CMP
United Business Media

>> **PRODUCT REVIEWS** SIDE EFFECTS' HOUDINI 9

NOVEMBER 2007

gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

>> **PORTING WITH POWER**
BRINGING A GAME ENGINE FROM PC TO NINTENDO DS

>> **AURAL FIXATION**
AUDIO LOCALIZATION AROUND THE GLOBE

>> **Q/A'S 10 COMMANDMENTS**
HOW TO BEST UTILIZE YOUR TESTING FRIENDS

POSTMORTEM:
2K GAMES' BIOSHOCK

$5.95US $6.95CAN

11>

0 71486 02133 9

DISPLAY UNTIL DECEMBER 18, 2007

CREATE
In *Assassin's Creed,* Ubisoft used Autodesk® 3ds Max® software to create a hero character so real you can almost feel the coarseness of his tunic.

ANIMATE
Autodesk® MotionBuilder™ software enabled the assassin to fluidly jump from rooftops to cobblestone streets with ease.

INTEGRATE
Using Autodesk® HumanIK® middleware, Ubisoft grounded the assassin in his 12th century boots and his run-time environment.

HOW UBISOFT GAVE
AN ASSASSIN HIS SOUL.

autodesk.com/Games

Image courtesy of Ubisoft

gamedeveloper

CONTENTS

NOVEMBER 2007
VOLUME 14, NUMBER 10

20

COVER ART: 2K Games

JAPAN'S SLIPPED GRIP

THE OVERARCHING STATEMENT I HEARD FROM Japanese game developers at the recent Tokyo Game Show was this: The market is changing and shrinking, and nobody knows what to do about it.

When I was a kid, all the best console games came from Japan. That's just how it was. While Atari was fiddling around with the 7800, Konami was releasing Contra, and Capcom was bringing out Megaman. Based on this, my Japanese video game snobbery continued through the Saturn and PlayStation eras, on through the Dreamcast days and into the beginning of the PlayStation 2 reign. Then, something happened.

WHAT RISES IN THE EAST

These days, for every Katamari Damacy or Killer7 you get five cookie cutter dating sims, or bare-bones fighting game upgrades, or any number of generic action adventure games with plodding stories about saving the world (a world which exists pretty much exclusively in Tokyo, I might add). Western developers are at the top of the heap now, not just from a technical standpoint, but also in many ways from a design standpoint. It used to be that we Western folks were known for our shoddy ports and sports games more than our innovative blockbusters. This was especially true in the U.S. The Europeans had a very nice creative scene surrounding the Amiga which has, unfortunately, yet to be matched.

Granted, we still have our share of shoddy ports and franchise-branded sports games, not to mention an intense publisher-driven case of "me too"-ism when it comes to popular genres, but Westerners weren't making many games like Halo, Grand Theft Auto, Beyond Good and Evil, Call of Duty, Flow, and BioShock back in the 8- and 16-bit eras.

Certainly, that statement is incredibly debatable—not everyone in this office even agrees with me. LucasArts' adventure games were enormously influential in terms of story for instance, and American arcade games like Robotron, Breakout, and Gauntlet continue to spawn imitators. But the fact is, if you wanted an experience like Super Mario Bros., Sonic the Hedgehog, Street Fighter II, or Final Fantasy, Japan was it.

SETS IN THE WEST

The development community has no doubt noticed that Japan currently has its eyes set squarely on Western markets. The fact is the higher powered next-gen consoles just aren't taking off over there the way anyone hoped they would.

With that in mind, companies like Capcom are making games like Dead Rising and Lost Planet specifically for the West. Most other large companies are looking to do the same.

Developers in Tokyo told me over and over that their native market was stagnating. Consumers bought Wiis and DS systems, but as NanaOn-Sha's Masaya Matsuura indicates in an accompanying news story (see pg. 4), many feel the DS software bubble has burst. It used to be that anything on the console would sell; now in a glut of low-quality games, only Nintendo products are selling well, once again.

Japan's game market has simply not kept up from a technology standpoint. Companies were incredibly late to adopt middleware or engine technologies for fear of losing a competitive edge. A lack of communication with other developers (the kind *Game Developer* magazine strives for or the kind found at the Game Developers Conference) means Japanese developers are all trying to solve the very same problems in isolation.

I even heard once of teams working on two similar games simultaneously, one for the American market and one for Japan, who were not allowed to share assets, engine, code—nothing. And these games were the same genre, made by two different teams within the same company! It's not easy to convince the higher-ups to share information. Western developers have been able to do it because they always have, and perhaps because in a sense they came from behind. It takes cooperation to get ahead.

SUN FOR EVERYONE

The upshot of this is that there's a lot more opportunity now to work with Japanese publishers on games for your own market. I think we're going to see a lot more cross-cultural collaboration to this end. Japanese design sensibility and Western technology could yield a much-needed sense of renewal.

Japan isn't going anywhere. Nintendo and Sony's in-house teams continue to release games that push the envelope, and companies like Capcom, Konami, Square Enix, and Namco-Bandai all have excellent products coming out. But they're all working harder than ever to get Western attention. It will be very interesting to see how these companies adjust to the changing climate, or whether Japan will just become the land of Nintendo.

Brandon Sheffield
Senior Editor

gamedeveloper
www.gdmag.com

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

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

EDITORIAL
PUBLISHER
Simon Carless *scarless@gdmag.com*
SENIOR EDITOR
Brandon Sheffield *bsheffield@gdmag.com*
SENIOR CONTRIBUTING EDITOR
Jill Duffy *jduffy@gdmag.com*
ART DIRECTOR
Cliff Scorso *cscorso@gdmag.com*
ASSISTANT PRODUCTION EDITOR
Jeffrey Fleming
CONTRIBUTING EDITORS
Jesse Harlin *jharlin@gdmag.com*
Noah Falstein *nfalstein@gdmag.com*
Steve Theodore *stheodore@gdmag.com*
Mick West *mwest@gdmag.com*
ADVISORY BOARD
Hal Barwood Designer-at-Large
Ellen Guon Beeman Microsoft
Brad Bulkley Neversoft
Clinton Keith High Moon Studios
Ryan Lesser Harmonix
Mark DeLoura Ubisoft

ADVERTISING SALES
MEDIA ACCOUNT MANAGER
John Watson e: *jmwatson@cmp.com* t: 415.947.6224
GLOBAL SALES MANAGER, RECRUITMENT & EDUCATION
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227
SALES REPRESENTATIVE, EDUCATION & RECRUITMENT
Gina Gross e: *ggross@cmp.com* t: 415.947.6241
DIRECTOR OF SALES
Steve McGill
SR. EVENTS ACCOUNT MANAGER, SOUTHWEST
Jasmin Davé
SR. ACCOUNT MANAGER, EAST COAST, EUROPE & EASTERN CANADA
Cecily Herbst
ACCOUNT MANAGER, WESTERN CANADA, INDIA, AUSTRALIA, & ASIA
Amanda Mae Miller

ADVERTISING PRODUCTION
ADVERTISING PRODUCTION MANAGER Kevin Chanel

REPRINTS
PARS INTERNATIONAL
Joe Nunziata t: 212.221.9595 e: *reprints@parsintl.com*

MARKETING
MARKETING MANAGER
Hilary McVicker e: *hmcvicker@cmp.com* t: 415.947.6207

CMP GAME GROUP
SENIOR VP, GAME, DOBB'S, ICMI GROUP Philip Chapnick
GROUP DIRECTOR Kathy Schoback

AUDIENCE DEVELOPMENT
GROUP DIRECTOR Kathy Henry e: *khenry@cmp.com*
DIRECTOR Kristi Cunningham e: *kcunningham@cmp.com*
LIST RENTAL Merit Direct LLC t: 914.368.1000

INTERNATIONAL LICENSING INFORMATION
Mario Salinas t: 650.513.4234 f: 650.513.4482
e: *msalinas@cmp.com*

CMP TECHNOLOGY MANAGEMENT
PRESIDENT AND CEO Steve Weitzner
EXECUTIVE VP AND CFO Adam Marder
PRESIDENT, CMP BUSINESS TECHNOLOGY GROUP Tony Uphoff
PRESIDENT, CMP CHANNEL Robert Faletra
PRESIDENT, CMP ELECTRONICS GROUP Paul Miller
PRESIDENT, CMP GAME, DOBB'S, ICMI GROUP Philip Chapnick
CORPORATE SENIOR VP SALES Anne Marie Miller
SENIOR VP, HUMAN RESOURCES Marvlieu Jolla Hall
SENIOR VP, MANUFACTURING Marie Myers
SENIOR VP, COMMUNICATIONS Alexandra Raine

Game Developer is BPA approved
CMP United Business Media

You've got
the looks.

We've got
the brains.

Kynapse A.I. Middleware

Kynogon's A.I. middleware, Kynapse, is the brains behind the work of some of the leading game developers out there. In fact, **6 of the top 10 game publishers** use Kynapse in developing their best-selling titles.

INTEGRATED UNREAL ENGINE PARTNER

www.kynogon.com

France ■ United Kingdom ■ Canada

kynogon
no mind no game

TGS 2007: DEVELOPERS SPEAK

Masaya Matsuura

President of NanaOn-Sha

on the current state of the Japanese market

It's getting much more difficult. The game market, especially in this country, is still very conservative. Many people know that the DS has very unique titles like BRAIN TRAINING, or things like that. But it's not for the younger market, more for old folks like me. The young age market is still very conservative (in Japan).

We have to focus on the worldwide market simultaneously at the start of development. Many developers didn't have to care about the overseas market, because the Japanese market used to be powerful enough. But now it's not so powerful. This is a very big point. Actually the offer to make new game titles is increasing from overseas publishers. So we have to think about American kids, and European kids.

Some people say that the DS software bubble has already burst. I heard about a case with some sort of learning game—the first one sold over 200,000. The second one sold 8,000. These kinds of things are a big concern.

Yasuhiro Wada

Managing Director and President of the digital contents company within Marvelous, and creator of the HARVEST MOON series

on the paradoxical use of technology to promote a pro-environmental idea

Technology is just a tool, just a medium to get the message. It is a bit of a dichotomy, but not contradictory. I'm not pushing everyone to go back to country life, but just to realize that nature is important too.

I want to let people know that the countryside is important—and everyone wants to move to the cities, and nobody wants to live in the country to take care of the environment there. In Japan, several people have played this game, and because they liked it, they went to Hokkaido to see what the farm life is like.

Kazuhiro Yamao

President of Killaware (maker of the upcoming original DS title LUXPAIN)

on starting up a new independent company in Japan

Naturally it's not easy at all, and you need a good creator involved who has past results. If you look at Killaware, we don't have a long past. So if companies just look at our past games, there's nothing to see, but since we had established creative talent (Kiyotaka Ueda, previously of Atlus), Marvelous invested in us for this title.

Of course it's riskier to invest in original titles, and we know that. We can always try to make some movie licensed titles, or sequels of another company's games, but we know we won't go that far making the same thing over and over again. We knew at some point we had to take a risk, so we just said yeah, let's give it a try.

Tetsuya Mizuguchi

Chief Creative Officer of Q Entertainment

on making a true artistic game in the vein of Kandinsky

We need to cover two directions. The realism, and the organic and abstract—the softer direction. The console is like a canvass for us, so if I got a new canvass—high def, high res—and have lighter and softer vivid colors, we'll draw and create a new world.

But this is not a painting. We have to think of one more layer, which is that something is moving all the time, with music. It's really tough—we have to think about all the time something moving, the color changing, the sound and the music through the interactive process—with cause and effect.

Interviews and photo by Brandon Sheffield.

IGF LAUNCHES MOBILE COMPETITION

THE INDEPENDENT GAMES FESTIVAL HAS LAUNCHED a new IGF Mobile competition to encourage creative, independently developed games for mobile devices such as cellphones and Palm OS, as well as handhelds like the Nintendo DS and Sony PSP. Running parallel to the main IGF competition, IGF Mobile will award a total of $20,000 in prizes to the most innovative entries, and all the finalists' games will be playable in a special pavilion at the 2008 Game Developers Conference.

The winnings include $2,000 for best in Audio Achievement, Technical Achievement, Achievement In Art, Innovation In Augmented Design, and Innovation In Mobile Game Design, while the overall IGF Mobile Best Game will be awarded $10,000. Three finalists will be announced in each category in December 2007, along with five finalists for the IGF Mobile Best Game. Winners will be honored during the main IGF Awards on Wednesday February 20, 2008.

The competition's 'Augmented Design' award will be presented by IGF Mobile Founding and Platinum Sponsor Nvidia in recognition of games that best utilize mobile-centric technology such as GPS, camera, motion-sensing, Wi-Fi, or Bluetooth features in gameplay.

According to IGF Chairman (and *Game Developer* publisher) Simon Carless, the award should be of particular interest to independent developers. "There are all kinds of cool game design things you can do if you have a handheld device and other add-ons such as GPS, a camera, Internet connectivity and so on," he said.

"There are great, innovative indie games out there which use the unique advantages of handheld hardware," he continued, speaking of the event in general, "from Gamevil's NOM and SKIPPING STONE for cellphones through DS games such as 5th Cell's DRAWN TO LIFE or even group games such as PAC-MANHATTAN, and we're delighted to set up a new awards to help promote titles such as these."

—Jeffrey Fleming

EA ACQUIRES PANDEMIC/BIOWARE

IN A WELL-PUBLICIZED MOVE, ELECTRONIC ARTS HAS purchased Pandemic and BioWare parent company VG Holding Corp for well over $800 million, which brings the two influential and previously independent firms under the EA Games label, run by Frank Gibeau.

"We're really excited about the chance to work with John (Riccitiello) again," said BioWare co-CEO Greg Zeschuk. "We had a great working relationship in the early days of BioWare and Pandemic, and we have a strong vision for what we want to do. We think that BioWare/Pandemic joining EA gives us the chance to do even more stuff. Everyone knows BioWare's very focused on making quality games. We think this gives us even more people to interact with, more things to learn, more technology, more know-how to share and to do things even better."

Pandemic attempted to allay fears that this is an act of hegemony on EA's part, with chief production officer Greg Borrud saying, "Nothing is changing internally in terms of management, nothing is changing in terms of the way we develop games, the kind of games we want to develop, our core pillars of building big brands and having event launches, and attracting the best talent in the industry. Those have been kind of core to who we are at Pandemic for years, and we think that actually EA allows us to take that to the next level."

In the investor conference call that followed the announcement, EA CFO Warren Jensen mentioned that EA "had [its] eye on these studios for several years," indicating that the acquisition fills a gap in EA's lineup—namely RPGs and action-adventure titles, presumably referring to original IP.

An on-message Ray Muzyka (Co-CEO of BioWare) confirmed this idea: "I think BioWare is known for a lot of things, and it's associated with quality," he said. "We have great people here that make it happen, and that's only going to be enhanced by this. (EA is) looking for excellence in games, the best story-driven games in the world. We're proud to be part of that team."

With this acquisition, EA has cemented its position as the largest game-related publishing and developing organization in the world, adding some 800 employees to its growing roster.

—Brandon Sheffield, Christian Nutt, Brandon Boyer

CALENDAR

Independent Game Conference: Austin 2007
Austin Convention Center
500 E Cesar Chavez Street
Austin, TX 78701
November 29-30, 2007
Price: Approx. $125
www.independentgameconference.com

11th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games
Université de La Rochelle
La Rochelle, France
November 21-23, 2007
Price: 450 Euro
www.scit.wlv.ac.uk/gamecentre/cgames07_lar/index.html

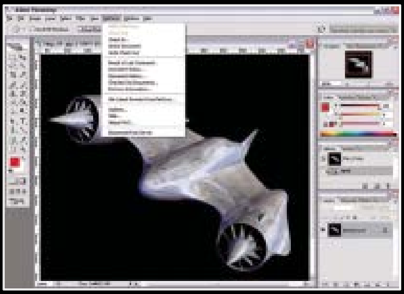
GRAPHITE 2007: 5th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia
Perth
Western Australia, Australia
December 1-4, 2007
Price: $1,050 AU
www.graphite.siggraph.org

Australasian Conference on Interactive Entertainment
RMIT University
Melbourne
Victoria, Australia
December 3-5, 2007
Price: $520 AU
www.ie.rmit.edu.au

Perforce | *Fast* Software Configuration Management

Introducing P4GT,

a productivity feature of Perforce SCM.



P4GT

The Perforce Plug-in for Graphical **Tools,** P4GT, makes version **control** painless by **seamlessly integrating Perforce** with **leading graphical tools. Drop-down menus allow access to Perforce from within 3ds Max, Maya, Softimage XSI, and Adobe Photoshop.**

Art and development teams can standardize on Perforce to version and manage both source code and digital assets. Enhanced collaboration during the design process helps teams to work together in real time to release small patches or create whole new worlds.

P4GT is just one of the many productivity tools that comes with the Perforce SCM System.

PERFORCE
SOFTWARE

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

stefan reinalter

SCALING SMALL

PORTING PC TECHNOLOGY TO THE NINTENDO DS

AFTER FINISHING OUR PC-ADVENTURE GAME UNDERCOVER: OPERATION WINTERSUN (released in the U.S. on August 28), Sproing Interactive Media started to work on its successor UNDERCOVER: DUAL MOTIVES, a prequel to the original for the Nintendo DS.

Up to this point, all our adventure technology was completely PC-based and had to be ported to the DS. Most of the taxing tasks centered on the fact that the DS has far less main memory and CPU resources than a standard PC. Because of this, we had to put quite a bit of effort into managing the memory resources, avoiding memory fragmentation, and coming up with new algorithms because the existing ones simply didn't work on the handheld platform.

MEMORY LANE

The Nintendo DS has 4MB of main memory in which to fit the program executable and all application memory. With UNDERCOVER: DUAL MOTIVES, the executable was approximately 1.5MB, leaving us only 2.5MB for the program itself. After deducting another 900KB for playing music and sound effects, we were left with 1.6MB for all game elements, characters, text, and animations.

At some points in the game, there are up to seven characters visible on both screens, with each character taking up 64x128 pixels in VRAM and main memory (see Figure 1). The animation data needed for that number of characters would never fit into the available memory, so all in-game animations had to be streamed from the cartridge and decompressed in real-time.

Animation data consists of single rendered frames for each of the characters' animations. These single frames were put into a large strip for each animation, which in turn were palletized and compressed by one of our tools. We used simple Run Length Encoding (RLE) packing for our compression scheme

STEFAN REINALTER

was lead programmer on UNDERCOVER: OPERATION WINTERSUN for PC and UNDERCOVER: DUAL MOTIVES for the Nintendo DS and built the adventure technology for both platforms. Email him at **sreinalter@gdmag.com**.

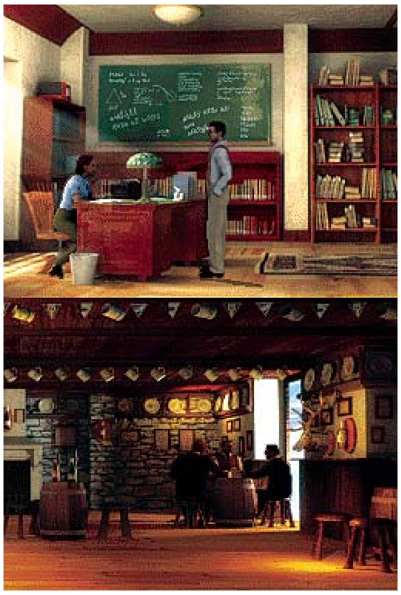


FIGURE 1 As many as seven characters must appear on screen during the game, and each character eats up 64x128 pixels in VRAM and main memory.

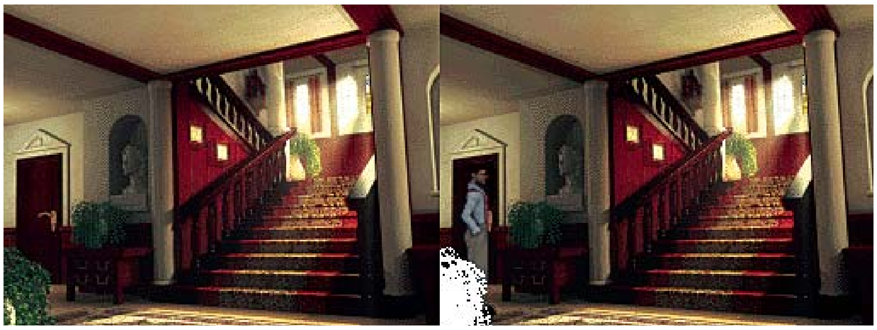


FIGURE 2 The decal sprite mask (in white) shows where objects and characters will be occluded.

because, for one, it provides a good tradeoff between data size and CPU-time needed for decompression, and two, each frame can be accessed individually, which was a must. Because of this, we couldn't use other more conventional compression techniques, such as zipping.

Because streaming many chunks of data per frame can quickly consume precious milliseconds, we implemented asynchronous streaming for all animation data. At the beginning of each frame, the asynchronous calls to the file system are initiated so that the streaming process can take place while all the other game objects are being updated. At the end of each update cycle, the animations are decompressed and uploaded into VRAM. Decompression of the RLE data was done using standard C++ code, additionally utilizing the SDK's fast copy-routines for larger blocks resulted in a general speed-up. It is possible to squeeze a few more cycles out of the code by using assembler or resorting to a different compression algorithm, but luckily we had no need to go that far.

In addition to having little memory, memory fragmentation can be an absolute killer. Because our base technology (and some game-code, too) used Stereo Lithography (STL) vectors and strings, we had to be extra careful about memory usage. We cut down STL usage as much as possible, and used a separate heap for all allocations smaller than 256 bytes, in which most of the STL allocations fit. Using about 150KB for this heap proved reasonable and didn't fragment our main heap where all the other allocations resided. Of course, we could have gotten rid of the STL completely, but our implementation shows that it's possible to use the STL on such platforms too, without any major hassle.

Other developers who attempt this approach should know that if you end up having too little memory during development, you should check the compiler settings to favor either compiling for speed or code size. When there are still clock cycles to waste, it might be better to opt for a smaller executable and free up some memory.

AN INTEGRATED WHOLE

For rendering in UNDERCOVER: DUAL MOTIVES, we used the Nintendo DS' dual 3D rendering mode, giving us 256KB available video memory in which all the textures used per frame had to fit. With up to seven animated characters at 64x128 pixels and two simultaneous scenes at 256x192 pixels, that left us little room for anything else, such as text, sprites, icons, and whatever else might go into an adventure game.

Our main strategy for VRAM allocation was to avoid fragmentation at all costs. Specific assets like backgrounds, in-game text, and small menu icons, which were used throughout the game, were assigned their own fixed region in our texture manager. In addition, through careful organization of all VRAM allocations along with automatic management of releasing and reloading the required assets, we ended up having zero VRAM fragmentation for the entirety of the game! At some points, for example when initiating a dialogue between characters or switching to the in-game menu, some assets are temporarily freed from VRAM and reloaded later when they are needed. Admittedly, some freeing up and reloading had to be done manually by giving hints to our texture manager, but it was worth it to do so.

As mentioned previously, there was not much VRAM to spare, so we had to cut down wherever we could. Developers who want to repeat some of this work should know that if standard 8-bit textures are used, the displaying text can lower the available VRAM pretty fast.

In DUAL MOTIVES, we used a screen-sized (256x192 pixels) 2-bit texture in which all currently displayed text was tightly packed. Whenever new text was to be displayed, the corresponding string was streamed from the cartridge and rendered and packed into the texture.

Rendering the text with the given font only had to be done once using this method (upon the first frame being displayed); we were also able to render whole sentences with just a single rectangle. Plus, we could render the text in different colors using flat-shaded rectangles without having to upload new palette data, and we could add outlines to the text with only four additional rendered rectangles.

SCROLLING, BITS, SPRITES, AND SHADOWS

Most of the backgrounds in DUAL MOTIVES are scrollable and do not need to be power-of-two in size. However, the Nintendo DS does demand such formatting, thus requiring us to adapt the memory layout of our background images.

All the background images are rotated clockwise by 90 degrees and are split into two parts: one 128 pixels wide and the other 64 pixels wide to meet the power-of-two restriction. Using this layout, only one line of new data has to be copied to the texture for each scrolled pixel. Furthermore, every line is aligned on 16-bit boundaries, which allows for faster copying. Upon rendering, the textures are then rotated back 90 degrees.

The characters in the game should be able to walk behind objects in various scenes, but we had tight restrictions with the VRAM. The easiest solution would have been to use 8-bit sprites and just render them with a smaller z-position than the characters, but then we would have used up four times as much VRAM as we really needed to. Instead, we used the Nintendo DS' decal sprite functionality and 2-bit masks for all occluding sprites.

Every occluding object is already contained in the background (see Figure 2) and is rendered using the background image with a binary mask at its desired z-position. The DS hardware then takes the source image (the background) and rasterizes the pixels into the z-buffer only where the binary mask is set; the remaining pixels are untouched.

At run-time, no 3D information about the scene is available, so we had to fake the character shadows in some believable manner. Our approach was to take the character's current texture, scale it and skew it, and then render it using alpha-blended rectangles. This resulted in nice smoothed shadows. For additional fine-tuning, the scale and skew values as well as the shadow's color and offsets can be edited in real time, which gave our designers greater control over this feature.

One obvious flaw of this method is that shadows can't properly creep up on walls, so you either have to come up with a solution for this problem or be on a team with clever artists and designers.

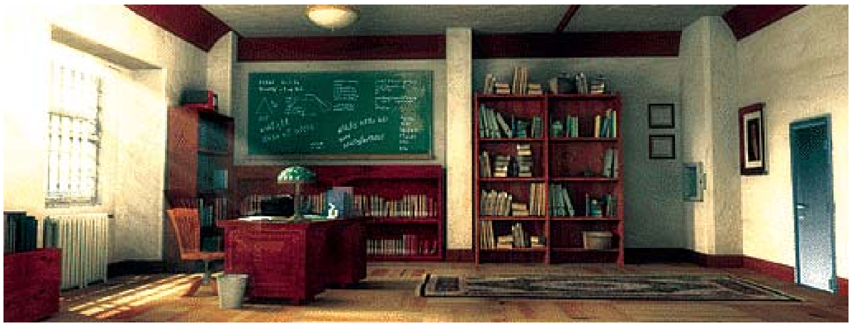


FIGURE 3 A typical background in which pathfinding is necessary.

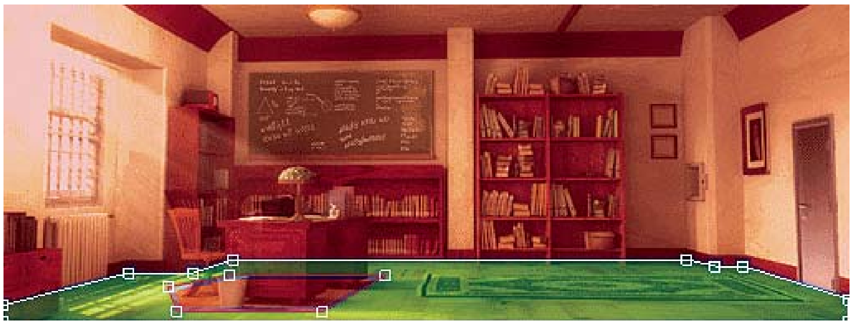


FIGURE 4 Areas where the player can walk are shown here superimposed on the background. Traversable areas are depicted in green, where as blocked areas are shown in red. Polygon vertices are shown using white rectangles.

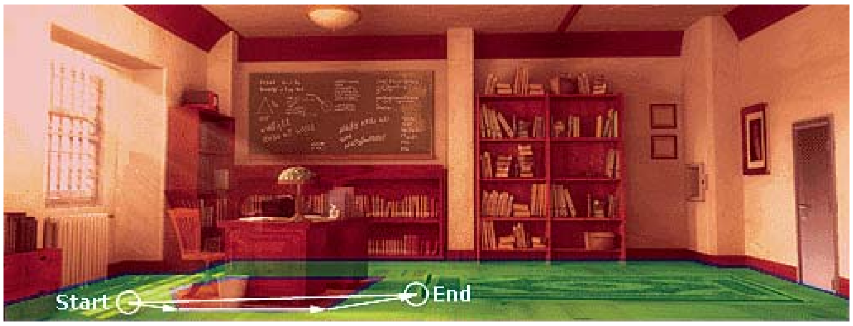


FIGURE 5 The virtual ray cast by our pathfinding algorithm.

PATHFINDING

Our adventure technology on the PC uses the familiar A* algorithm for pathfinding. However, with the Nintendo DS's memory and CPU limitations, this is not feasible in real time, so we had to come up with a different solution. We used a combination of pre-computed paths and ray-casting at run time. (See Figures 3–5.)

First, the level designer adds polygons for traversable or blocked areas in our editor. Then, using these polygons, the editor computes a binary mask of traversable and blocked areas (see Figure 4).

Every vertex is used as a node in our pre-process for calculating paths between them. This pre-process uses a modified Floyd-Warshall algorithm to determine the shortest paths between all pairs of nodes. We use the distance between the pair of nodes as their weight and set it to infinity if there's no connection between them. The shortest path for all nodes can be conveniently stored in a matrix, where each matrix element depicts the previous node to use on this path. By constraining the number of nodes to 255, each element in the matrix takes up one byte and is just an index into an array of nodes. Thus, the amount of memory consumed by our pathfinding algorithm is kept really low.

At run-time, the pathfinding algorithm casts a virtual ray (shown in Figure 5) from start to end in our binary mask to check whether the path is blocked. Ray-casting is done using a standard implementation of the Bresenham line-drawing algorithm.

If the path is blocked, our algorithm finds the nearest node to the first and last points of intersection along the ray. Using our pre-computed path matrix, the shortest path between these nodes can be determined with just a few array lookups.

The last step of our algorithm then optimizes the path by throwing away redundant nodes, which can be reached from the start and end points without hitting any blocking area. Again, this happens by casting rays in the binary mask image.

CONTINUED ON PG 10

CONTINUED FROM PG 9

It should be possible to further enhance our algorithm to account for dynamic objects as well, although it was not necessary for UNDERCOVER: DUAL MOTIVES.

SCRIPT COMMANDS AND INPUT CONTROLS

Our scripting system consisted of small script commands, which can be called from scripts within the code. Scripts themselves are similar to self-contained state machines that only use memory for states and parameters. Because we're not using co-routines or something similar for script commands, the scripting system from our PC games instantly worked on the Nintendo DS without any changes.

One thing to keep in mind is that all the scripting code will make your executable bigger, essentially eating up main memory.

We wanted to exploit the input mechanisms of the Nintendo DS as much as we could; one of the highlights is a mini game that requires the player to fire birdshot by blowing into the microphone. Unlike PC adventure games, there's nothing like mouse-over events on the Nintendo DS, unless you want the user to constantly move the stylus over the screen. So we had to come up with different methods of input—and there was a multitude of iterations during development. In the end we settled for hotspots, pop-up menus, and gestures for quick interaction with interactive game elements and characters.

Existing technology made it easier to build a complete editor for our game. We simulated the Nintendo DS's rendering capabilities using OpenGL, and most everything else was already available through existing technology.

During development, it proved handy to have a fully functioning working environment running on the PC. This way, not everybody on the team required a development kit, making turn-around times much shorter. Of course there were specific features of the Nintendo DS (for example, the microphone input and certain mini games) that could only be implemented and tested using a proper dev kit.

ROOM FOR IMPROVEMENT

Once we had the streaming of sound effects, animations, and texts in place, the cartridge was already becoming quite stressed. However, we believe that there is still room for improvement for other developers who are working on this problem. One possible solution is to stream ADPCM-compressed audio files from the cartridge to achieve far superior music quality compared to using MIDI files. Another option might be to stream background animations—our team would need a better suited compression algorithm than simple RLE-coding for that, otherwise there might not be enough CPU-time left for the remaining tasks with all the streaming and decompressing going on.

All in all, we're happy with how the development of UNDERCOVER: DUAL MOTIVES turned out. The Nintendo DS is a fine piece of hardware and is easily capable of doing more than a developer's first impressions might suggest.

The author wishes to acknowledge his co-workers for their valuable suggestions for this article.

Lyon GDC 2007

Lyon GameDevelopers® Conference

REGISTER BY 10/30 AND SAVE!

WWW.LYONGDC.COM

Attend Lyon GDC . . .
And gain access to world class conference content and network with industry leaders

Sponsor Lyon GDC . . .
And position your company as a key driver of global game development

3-4 DECEMBER 2007
LYON, FRANCE

PRESENTED BY:
CONNECTION EVENTS
CMP

ADVERTISEMENT

Unreal® Technology News

by Mark Rein, Epic Games, Inc.

Canadian-born Mark Rein is Vice President and Co-Founder of Epic Games based in Cary, North Carolina. Epic's Unreal Engine 3 has won Game Developer Magazine's Frontline Award for Best Game Engine for the past three years and Epic was recently awarded Best Studio at the Spike TV Video Game Awards. Epic's Gears of War, won Gamespot's overall Game of the Year and sold over 4,000,000 units on Xbox 360. Epic is currently working on the Unreal Tournament 3 for publisher Midway and a PC version of Gears of War for publisher Microsoft Game Studios.

Upcoming Epic Attended Events:

Lyon GDC
Game Connection
Le Palais des Congrès de Lyon
December 4-6, 2007

GDC 2008
San Francisco, CA
February 18-22, 2008

Please email:
mrein@epicgames.com
for appointments.

UMBRA JOINS THE UNREAL ENGINE 3 INTEGRATED PARTNERS PROGRAM

Umbra Software Ltd., the visibility optimization experts, today announced they have joined Epic Games, Inc. in the prestigious Integrated Partners Program for Unreal® Engine 3. The industry-standard Umbra(tm) visibility optimization middleware has been fully integrated into Unreal Engine 3 and is now available for licensing directly from Umbra Software.

UE3 includes occlusion culling, using a mix of automated and manual processes which has been used successfully in many games. With the integration of Umbra Software's hierarchical occlusion culling technology, developers will have a fully automated system that is scalable to very large worlds. This can result in time savings in content creation, and allow for much more complex content. Umbra middleware also optimizes player-created content, effectively allowing the players to modify worlds on the fly. Massively Multiplayer online (MMO) games in particular benefit from the increased freedom of movement brought on by visibility optimization.

Unreal Tournament 3 demonstrates the great scalability of Unreal Engine 3

"Several Unreal Engine 3 licensees already use Umbra's middleware and have been delighted with the results," said Michael Capps, President, Epic Games, Inc. "They're a perfect fit for our partners program, and now it's easier than ever for Unreal licensees to integrate Umbra software's technology."

"We're thrilled to have the Umbra technology integrated into Unreal Engine 3," says Farhad Taherazer, Umbra Software's VP of Marketing. "Our rock-solid middleware and the fantastic Unreal Engine 3 are a boon to all game developers."

AMERICAN MCGEE'S SPICY HORSE GAMES LICENSES UE3 FROM EPIC GAMES CHINA

Spicy Horse Games, a Shanghai based game developer, together with Epic Games China announced that it has entered into a multi-title agreement to license Unreal Engine 3. Spicy Horse Games is using UE3 for "American McGee's Grimm" along with upcoming PC and console game titles. "Grimm" is a part of the GameTap original lineup, expected to launch starting in 2008.

"Unreal Engine 3 has given our team the tools they need to express their creativity in a rapid and predictable way," said American McGee, Creative Director of Spicy Horse. "Working with the great people at Epic Games China allows us capture rapidly expanding business and creative opportunities in Chinese and global gaming."

Paul Meegan, CEO of Epic Games China said, "We're very happy to be working with American McGee and his team at Spicy Horse. American has a reputation for being highly creative, and his is one of the first independent teams to make games for the global marketplace entirely in China. We look forward to seeing what they do with the technology."

EPIC'S UE3-BASED PC GAMES ALMOST GOLDEN

The Windows versions of Gears of War and Unreal Tournament 3 will soon be on their way to stores world-wide.

Both releases are important milestones for UE3. Gears of War supports Microsoft's Games for Windows LIVE platform, and Unreal Tournament 3 uses GameSpy's multi-player platform. Our licensees have a fast path for integrating either solution. Plus, our multi-platform approach required a network abstraction layer which makes it easier than ever for licensees to use other networking libraries, or to create their own in-house solution.

We've just released the UT3 beta demo, generating feedback from thousands of end-users ahead of our retail release. In addition to gameplay feedback, the demo helped us gauge our efforts to create a highly scalable graphics engine. We've spent considerable time and effort optimizing the engine for older PC configurations. Feedback from the demo shows this was time well spent, because a surprisingly wide range of PCs can run this cutting-edge game. In fact, UT3 runs smoothly on hardware that was generally available when the last Unreal Tournament game was released, back in September 2003.

UT3 is also in development for PS3, Xbox 360, Mac and Linux. Gears of War is also in development for Mac.

POWERED BY UNREAL TECHNOLOGY

EPIC GAMES

For UE3 licensing inquiries email:
licensing@epicgames.com

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

W W W . E P I C G A M E S . C O M

NETDEVIL®

Now Hiring for LEGO Universe

Tech
Lead Programmer
Senior Programmers
Graphics Programmers
Programmers
Assistant Technical Director
Database Administrator

Art
World Art Lead
3D Character Animators
3D Artists
Concept Artists
Motion Graphics/Flash Animators
LEGO Digital Designer Artists
Special F/X Artists

Community
Assistant Community Manager

Design
Senior Content Designer
Systems Designer
Scenario Designers

Production
Platform Producer
Associate Producer

Submit your resume online:

www.netdevil.com/employment/

Chuck McFadden

TEN COMMANDMENTS OF QUALITY ASSURANCE

1. Employ the Scientific Method.
2. Understand the difference between playing a game and testing a game. Spend most of your time doing the latter.
3. Be flexible.
4. Find and report bugs as early as possible.
5. Think like a hacker. Be creative in finding problems with the game.
6. Put in as much effort with your regression testing as you do with your initial testing.
7. Don't let Q/A members test designs they've created.
8. Don't write sloppy bugs. Spell and grammar check everything.
9. Test everything you can reasonably test.
10. Work under the assumption that most (if not all) bugs can be consistently reproduced.

CHUCK MCFADDEN *is an associate producer at Factor 5. He was previously Q/A manager at Namco Bandai Games, and a member of the IGDA Q/A SIG. Email him at* **cmcfadden@gdmag.com**.

IF Q/A HAD A SET OF 10 COMMANDMENTS, WHAT WOULD THEY BE? Approximately two years ago, a member of the IGDA's Q/A Special Interest Group (SIG) posed that very question, which ignited the SIG into lengthy discussions and friendly arguments until, weeks later, we settled on 10. They were compiled and listed on the SIG site, where they have remained in relative obscurity ... until now.

This article lists the commandments in no particular order and examines each in detail. Each has equal value; no one commandment is more important than any other. But ignore them and the god of game development will smite you.

CONTINUED ON PG 14

CONTINUED FROM PG 13

1. EMPLOY THE SCIENTIFIC METHOD.

When you boil it down, this is what separates a good tester from a bad tester: the good tester—consciously or unconsciously—uses the Scientific Method. Make this a part of your Q/A team's training process, just in case one of your testers paid as little attention during science class as this author. If trained well in the Scientific Method, upon observing a bug, a tester will:

a) *Observe and describe.* At the least, the tester should take notes on what happened. Ideally, your tester is recording everything (with a VCR or digitally), which helps the tester with the next few steps. Either way, this step simply states what happened, for example that the game crashed at the second save point.

b) *Formulate a hypothesis.* The tester should speculate on what might have caused the bug. Did they try to save when the game was streaming off the disc? Was a character talking when the save action triggered? Did the character "die" at the same time the save initialized? Based on the observed behavior, a tester can develop a testable hypothesis to repeat the bug. (An inexperienced or less-thorough tester will end the process here, writing up the bug based on his/her speculation.)

c) *Experiment.* The hypothesis should then be tested to see if it results in the initially observed behavior/bug. Re-experiment to narrow down the steps to repeat the bug. If the hypothesis is true, move onto the next step. If the hypothesis is false, refer to step A and start again at step B.

d) *Draw a conclusion and communicate the results.* Once the tester has verified that they can repeat the bug with the minimal number of steps (in other words, once they have verified that the hypothesis is true), they should write it up. How your tester writes up the bug is up to you and your established procedure. Regardless of how they write up the bug, if they use this method, the bug will be solidly researched and consistently reproducible.

2. UNDERSTAND THE DIFFERENCE BETWEEN PLAYING A GAME AND TESTING A GAME. SPEND MOST OF YOUR TIME DOING THE LATTER.

The most common misconception I've seen with new hires (and with industry colleagues who don't know much about Q/A) is they believe testers "play games all day." If this is 100 percent true with your Q/A team, you release terribly buggy games. Quality assurance is not just a matter of testing how a game is supposed to work, but how the game is not supposed to work.

The difference is simple. A good tester knows that losing needs to be tested as thoroughly as winning. He understands that getting the best lap time in a racing game is only half the job. Otherwise, a bug as simple as a crash at the "game over" screen will never come up if the tester is always playing to win.

This commandment also speaks to another philosophy: Don't exclusively hire "hardcore" gamers. While employing hardcore

a production perspective

MY EXPERIENCE WITH Q/A OVER THE years has been interesting. I started out as a tester in 1993, knowing nothing about software development. I was an enthusiastic video game player, a writer, and a pen-and-paper RPG fanatic. This list of commandments would have instantly accelerated my learning by a year.

Now, as a creative director, I crave more polish in games relative to everything else, which usually stems from good development and tech practices. Best practices impact gameplay much more than people often acknowledge. Even in cases where development teams are trying something new and ambitious, a fast frame rate, a solid interface, stability and lots of feedback are the real key to enhancing the player's experience.

I especially like number five from this list, about inspiring testers to play creatively. There's always one person on the test team who thinks like that—someone who covers the basic route that 90 percent of mainstream players will take, then tries five or six creative alternatives. After a while, you develop this as an instinct. "Hmm, this front door triggers a scripted door-opening sequence ... I wonder what happens if I break the side window and skip the door."

HEARTS ARE MADE FOR BREAKING

Q/A people know that most games can be broken (or will at least expose something that looks silly) if you try hard enough. Citing alt-path problems, then speculating on their likelihood is very useful; sometimes we take things like that and run with them creatively. We might say "If five percent of testers think to try this, what if we bullet-proof it, then make it more attractive or obvious, so that it occurs to maybe 20 percent of players." Working on the DEUS EX games, which were incredibly free-form, this happened a lot. Even working on an FPS like BLACKSITE, over the last year we've seen lots of cases where testers approach a combat scenario in a way we didn't expect—routes which might be under-supported and cause frustration. In all cases, having great dialogue with Q/A will make the game better.

THE FINAL PUSH

The very concept of "testing" has evolved a lot to include a bunch of best practices for the genre, such as blind usability tests and post production (as a serious phase of development). The game never gets good as fast as it does in the final months, so it pays to invest in post production testing. Ideally you'll finish the game from beginning to end as soon as you can, then spend a ton of time pounding on it, observing new players stumble through it, and looking for opportunities to pay off dramatic moments. Bringing in round after round of people who have never seen BLACKSITE has taught us invaluable lessons about where players get stuck, run out of ammo, get lost, and have the most or least fun. I wish we had three to six more months of that sort of thing—it would make a tremendous difference in the final quality of the game. I think the difference between great and mediocre publishers is the wisdom to invest in the final period of testing and tuning; the discipline to avoid cheating into this phase, even if a game is late.

—Harvey Smith, Midway Austin creative director

havok™

"We have been extremely impressed with the functionality and design of Havok Behavior™ and its associated tool. By building our system on top of Havok Behavior™, we have been able to save man-years of effort, and it has allowed our engineers to focus on the technology that will allow Blue Fang to create the most compelling animal behavior ever seen in games – or anywhere else, for that matter. We have been especially impressed with the quality of support provided by the folks at Havok. Thanks Havok!!!"

Bruce Blumberg, Senior Scientist, Blue Fang Games

Havok Behavior™ is intuitive composition tool for artists and designers, and a run-time SDK for game programmers. Together, the Behavior tool and SDK provide "what you see is what you get" results, accelerating development of cutting edge character performances for Wii™, PLAYSTATION®3, PlayStation®2, Xbox 360™, Xbox™, GameCube™ and the PC.

The Havok Behavior tool provides an intuitive user interface and editing paradigm that lets non-technical users master these concepts in a matter of days. The flexible Project scheme allows users to self-organize and share common animation assets in work groups, without corrupting each other's work.

Havok Behavior integrates out of the box with Havok Physics™ and Havok Animation™ which must be licensed, and is backed by our industry-leading support and customer commitment.

Havok Behavior Tool Includes:

- A flexible project scheme and asset viewer that organizes 1000's of character assets including rigs, skin/mesh bindings, animation clips, events, variables and behaviors
- A graphical editor for authoring character behaviors as Hierarchical Finite State Machines with blend trees
- Sophisticated Blend Trees per state that match in-game results
- Storage of Behavior assets in XML and as Havok serialized binary files that load directly in the game
- Unique event-driven simulation mode that enables interactive traversal of arbitrarily complex state machines
- A Sequence editor that provides repeatable behavior testing through automated playback of event and variable manipulations
- Variable declarations and bindings within Blend Trees and animation clips, enabling two-way interaction between the game's AI and the Havok Behavior SDK
- A "GameView" test application for verifying all character behaviors, including event generation and variable tweaking, independent of the Behavior tool

Havok Behavior SDK Includes:

- Hierarchical finite state machines on current and next-generation game platforms
- Full support for fast loading of serialized character and behaviors assets
- Compatible with earlier serialized character assets
- Provides run-time integration with animation and physics techniques – enabling pose matching, powered ragdolls, and IK fix-up
- Introduces new behavior controllers including grab, tackle, and climbing
- Supports AI-driven run-time variable changes at arbitrary points in the Blend Tree and in animation clips
- Supports user-supplied custom Blend Tree nodes – enables open-ended customization within the Havok Behavior framework (requires Tool source code extensions)
- Enables unified representation for entire character performance – including facial and hand-based animations

Havok™
North American Sales
25 Jessie Street
San Francisco, CA 94105

Tel: (415) 543-4620
Fax: (415) 543-4621

www.havok.com
salesteam@havok.com

UPCOMING EVENTS

- **Montreal International Games Summit**
  November 27-28, 2007
- **GDC San Francisco**
  February 18-22, 2008

HAVOK PRODUCTS

- Havok Behavior™
- Havok Animation™
- Havok Physics™
- Professional Services

Contact us today for more information on Havok Behavior™, or any of the Havok products. (415) 543-4620 or salesteam@havok.com

gamers is important, it's equally important to cultivate more casual players. As a lead or manager, you'll be tempted to hire the most dedicated gamers in your neighborhood. Unfortunately those testers invariably have the hardest time losing. Moreover, the more they "test" your game, the less likely they are to lose.

You can avoid this problem almost entirely if you write good test plans. But if your tester is also conscious of the difference between playing and testing, he's sure to find bugs beyond the scope of what the average test plan can predict. Occasionally remind your testers that you're paying them to do a job. They can play games on their own time. While they're on the clock, they are there to test.

3. BE FLEXIBLE.

Video games represent a truly collaborative art form. Everyone involved in the game development process needs to work with at least one other person and working with someone else requires flexibility. So, how can Q/A teams be flexible?

Q/A teams are always doing more for their game (and their company) than simply testing the latest build for bugs. This commandment recognizes and encourages that. Be flexible enough to allow your testers to demo their game for the press. Help your marketing and PR teams with screenshots and/or videos. After all, who knows how to avoid all the embarrassing bugs better than the people who found them? The Q/A team knows the workarounds better than anyone. If your company is flexible enough to allow the Q/A team to help out with this work, the results might be surprising.

Be cautious not to let your Q/A team take on too much work, however. Don't take on so much extra work that you find your team testing less than they should. If your Q/A team ceases to find important bugs, all those screenshots and videos won't amount to much.

Your Q/A team should also be flexible with the bugs they report. As a game nears completion, Q/A will invariably disagree with the production/development team on some of the bugs

usability research commandments

THOSE IN THE TRADITIONAL usability/user research field (outside of the games industry) have often created golden rules, commandments, or heuristics for practitioners to adhere to. Many times they focus on principles of interface design that strive to make things simple, efficient, easy to use, and so on. These lists haven't really had as big an impact in the games industry. Why? It could be that usability as a functional discipline is too new to the games industry. It could be that the usability golden rules just aren't as applicable. It could be that the heuristics are perceived as a threat to the "art" side of game design.

Several of the commandments in this article jump out loud and clear as having just as much relevance to the usability practitioner as the Q/A tester. Employ the scientific method—usability and user research methods were built on experimental psychology and experimental research methods. To take it even further, usability engineers I work with (myself included) were all trained in behavioral science research, so that's an easy one. Be flexible—it is our job to try and account for as many sources of bias and influence when we run usability tests and collect data, but sometimes the ideal is simply not practical. Don't let Q/A members test designs they've created—usability engineers will often have recommendations to solve usability problems, which is great. However, as McFadden points out, it is hard to be objective about one's own design.

USABILITY COMMANDMENTS

So what else? As a practicing usability engineer in the games industry who has shipped nearly fifteen titles over the course of seven years now, what other commandments would I put forth for fellow usability engineers? Here are a few unrefined ideas I've had.

Users have opinions, but designers make the call. During your research and testing, users will always have opinions on things they do or don't like. Your job isn't to adhere to user whims—your job is to identify areas where user behavior is not consistent with the designer's vision. What do you from there will be context-dependent.

No one likes an ivory tower academic (especially in crunch). Most developers aren't interested in the classic "it depends" answer to something. They also aren't interested in inferential statistics, hypothesis testing, or the number of users you need for a valid test. When asked to do something or answer a question, do your research and testing, and give it your best shot. Don't be afraid to have an informed opinion, even if your research wasn't suitable for a scientific peer-reviewed journal.

Usability engineers don't own the market on data analysis. Let's be clear, just because a usability engineer may have a Ph.D. in Experimental Psychology, that doesn't mean he or she is the only one who can analyze data.

Show passion and dedication equal to your partners. Working in this industry can be a labor of love. In my experience, the most successful games I've worked on have been the ones where the entire team was equally passionate about what they were making. The more passionate you are, the more trust you'll gain, and the more effective you'll be.

Iterate, iterate, iterate! It doesn't matter how many usability issues you may uncover in a usability test. What is more important is how many issues you identified, then fixed.

When you think you're done—you're not. As long as designers have questions or new ideas, you'll always have something worth researching. You can always go the extra mile and provide more feedback and more data to your partners that will help them make better decisions.

LEARN AND LET LEARN

The usability/user research discipline in games has clearly not been around as long as Q/A, so I don't feel too bad stealing some of their good ideas, and I don't feel too bad that my discipline doesn't have our own commandments list ... yet. Whether you agree or disagree, I've thrown out a handful for us to start with. If you've got any more, then let's get to it!

—Randy Pagulayan, Microsoft user research lead

they want to close out. Be flexible enough to let less important bugs go (all games ship with some bugs, after all), but insist on fixing the bugs that make the biggest difference.

How do you know the difference? If your Q/A team isn't already using some sort of prioritizing scheme in their bug database, get one started. It's as simple as assigning a letter (or number, or both!) to a kind of bug. When it comes time to close out the bugs, it can save you hours upon hours of headaches. (For more on this, see 'Setting the Bar,' January 2007.)

4. FIND AND REPORT BUGS AS EARLY AS POSSIBLE.

This is a tricky one. Q/A is usually employed at the end of the development process, which is absolutely the right way to build a game. This commandment isn't condoning full-blown testing before your alpha milestone. Doing so would only clog the bug database with annoying bugs like "game-has-no-sound."

However, it is helpful to have a lead tester review the save flow (for example) long before it's implemented in the game. If an experienced tester can look at a save flow system design and tell you what will fail a technical requirements checklist, it'll save both the Q/A team and the development team hours or even days of work down the line.

You could also ask a few testers to take a look at the first iterations of your control scheme. Don't tell them how to control the game, just give them the controller and ask them to play around a little. If they can intuitively understand how to interact with the game, then you know you're on the right track with your controls. If they have problems with certain aspects of the controls, you know what to work on next. This won't give you the feedback you need if you're wondering how intuitive a casual gamer would find your controls, but it's a good place to start.

You can have your lead tester look at the game's text to verify correct usage of naming conventions long before you've implemented the text into the game. These are things they'll test for anyway once the text is in-game, so why not have them look at it before doing difficult implementation?

The save flow, controls, and text check examples are just that, examples. There are many little things an experienced Q/A tester can do for you before the alpha milestone. If you schedule these things out ahead of time, and make sure the tester knows what he/she should not work on, your development team will have more time to work on more important things and the inevitable crunch time will be less ... crunchy.

5. THINK LIKE A HACKER. BE CREATIVE IN FINDING PROBLEMS WITH THE GAME.

It's relatively easy to find a spelling error or a level load bug. The best testers flex their mentality toward the test cases at hand, ranging from a technically skilled hacker to an anti-intuitive four year old. Crash bugs can be found from altering core files and manipulating fifteen different user interface screens (hacker) to smashing your palm on the keyboard and causing a buffer overrun (four year old). And there are plenty of bugs in between.

Hardcore gamers love to bend the game rules as far as possible. Keeping this commandment in mind can help you avoid shipping with those nasty "exploit" bugs found in multiplayer games (both console and PC). But this commandment doesn't just apply to multiplayer. Thinking like a hacker simply means looking for the flaws buried deep in the game, not just on the surface.

Don't only play the game the way you think the average consumer will. Think of ways to play that no one else will. If the main character is supposed to exit a room by using the door, ask your tester to find other ways out of the room. Can the character use the window? Can he jump through the ceiling? Can he walk through a wall? Approaching a seemingly mundane task and finding a creative way through it can result in a lot of surprising bugs.

6. PUT IN AS MUCH EFFORT WITH YOUR REGRESSION TESTING AS YOU DO WITH YOUR INITIAL TESTING (HALO TESTING).

Finding and reporting the initial bug, as well as properly regressing that bug, are two important and essential job responsibilities for every Q/A tester. However, there is a third area of testing that occurs: Halo testing. No, this isn't about the Bungie game. Halo testing is when a tester checks for newly uncovered or added bugs resulting from the fixed bug they are regressing. Successful Halo testing requires the tester to possess Q/A experience (or an innate talent for finding bugs), intimate knowledge of back-end systems, how the product is built or developed, and guidance by the Q/A management team overseeing the title.

Regression testing is more than just reproducing the initially reported issue. When a fix for a bug is checked in there is always a risk associated with that fix. That risk must be evaluated not only by the programmer, designer, or artist but also by the Q/A lead and tester. Ensuring the bug at hand is fixed is only the first step—the tester must then Halo test around that fix looking for new bugs that could have been a result of that fixed bug.

> The most effective testers are not those with the highest Gamerscore, but those possessing exceptional written and verbal communication skills.

7. DON'T LET Q/A MEMBERS TEST DESIGNS THEY'VE CREATED.

Otherwise known as the conflict of interest commandment, this one can get you in trouble with your ambitious testers. Every tester, at some point, has a brilliant idea. Be very careful when this happens.

It's human nature to be biased against one's own ideas. Therefore, when a tester's suggestion makes it into the game, don't allow her to regress it. By seeing her idea become reality, she cannot effectively Halo test the new feature. She will be less likely to observe any bugs resulting from this feature. Get another tester to put the new feature through its paces. This will ensure that the new feature gets the same objective attention as any other.

This commandment does not discourage testers from making the leap to level designer or some other position in the larger organization. Many industry luminaries started their careers in Q/A, and this commandment recognizes that. But it also serves as a reminder to resist allowing your tester's objectivity to be overshadowed by a really good idea.

8. DON'T WRITE SLOPPY BUGS. SPELL AND GRAMMAR CHECK EVERYTHING.

The most effective testers are not those with the highest Gamerscore, but those possessing exceptional written and verbal communication skills. This is because clearly-communicated bugs get fixed faster and better than sloppy, confusing bugs. Moreover, as a group whose fundamental responsibility is to find other people's mistakes, it's simply embarrassing and unprofessional when Q/A's own work is full of errors.

This commandment doesn't insist that every tester hold a Ph.D. in English. Even if your testers lack perfect grammar skills, the least they can do is spell check their work. Most bug databases have a spell check function built in. Insist that your testers use it.

9. TEST EVERYTHING YOU CAN REASONABLY TEST.

Don't ignore a feature just because it gives a good first impression. If it's in the game, test it (and test it often) to ensure it works as designed.

For example, upon initially testing the newest gun in a FPS, a tester will notice that it fires correctly, the correct sound plays, it deals the correct amount of damage to the enemy, and it depletes its ammunition according to spec. Oh! And it's really fun to shoot! At this point, your inexperienced tester (having finished the test plan for this gun) may think he's done. He'll move onto the next gun.

> While employing hardcore gamers is important, it's equally important to cultivate more casual players.

But, what if that new gun, when fired at an explosive crate, crashes the game? It seems tedious, but every feature needs to be tested in every reasonable way. That means firing weapons at every interactive target (and most non-interactive targets). Make sure everything from the smallest decal to the biggest explosion trigger and play correctly, lest you prematurely approve a new feature.

As another example, at some point we've all heard a developer say, "I only fixed *X* bug. I didn't touch anything else, so don't worry about testing the whole *Y* feature." An inexperienced tester will take that developer at his word and only test the change as reported. He would miss the bugs that unexpectedly resulted from the change to "*X* bug."

A good, thorough test plan (one that includes, for example, causing an explosion with every weapon type) can help you avoid this problem, but no test plan can predict every possibility. Your testers need to be aware that they are ultimately responsible for the stability of the game. They need to make sure everything works to spec in every situation. They also need to understand that a "simple change" to a seemingly innocuous feature can have unexpected results. Never settle for spot-checking and don't say you're "done" testing a feature until you're ready to ship it.

10. WORK UNDER THE ASSUMPTION THAT MOST (IF NOT ALL) BUGS CAN BE CONSISTENTLY REPRODUCED.

It's only a question of how difficult the bug is to repeat. Even the most "random" bug is repeatable given enough time and effort. Granted, it's not always wise to spend time attempting to consistently repeat an elusive bug, but if you can, it's much easier to fix.

This commandment ties into the first commandment (Scientific Method), but also begs the question: How much time should one spend trying to consistently repeat a bug? You might work under the assumption that every bug is repeatable, but is it advisable to consistently repeat every single bug?

Ask yourself—how important is this bug? If the tester "randomly" crashed the game after beating the first boss, isn't it justified to spend a day attempting to consistently repeat it? On the other hand, if the tester found that the credits scrolled unusually quickly once out of 10 viewings, you probably don't need to research the bug for more than a few minutes.

Consider the severity of the bug when you determine how much time to invest in getting it repeated. You know it can be consistently repeated, but you need to use your judgment to determine if it's worth the effort. If your testers can't get the bug to repeat, list how many times they attempted to against how many times they successfully repeated it (i.e. 1 out of 10 times, the credits scrolled at 5x speed). That way, the development team can address the bug knowing it's not yet consistently repeatable.

A GOLDEN RULE

Summing up these 10 commandments into one all-inclusive statement might look something like this: Test scientifically, creatively, and thoroughly enough to catch all the bugs, but not so obsessively as to jeopardize your ship date.

Whether you follow one golden rule or 10 commandments, Q/A work is as much an art as a science. It's at its best when a diverse team passionately comes together to work on a common goal; to do everything they can to help the development team make the best game possible. Hopefully these commandments can inspire you and your Q/A team to better serve that common goal.

The author acknowledges Rob Thompson, Q/A manager at Sony Online Entertainment for his contributions to this article.

gamedeveloper FRONTLINE FINALIST '05

"With ReplayDIRECTOR, we only had to find the bug once. Our developer just pressed play and saw the crash in the debugger… We had a fix the next day."

John Chowanec, *Lead Producer, Eidos Interactive*

BUGS. TOASTED.

RECORD. REPLAY. FIXED.

ReplayDIRECTOR™ gives you **Deep Recording**. This is much **more than just video capture.** Replay records every line of code that you execute and guarantees that it will Replay with the same path of execution through your code. Every time. Instantly Replay any bug you can find. Seriously.

DEEP RECORDING. NO SOURCE MODS.

download today at www.replaysolutions.com
email us at info@replaysolutions.com

REPLAY SOLUTIONS
1065 E. Hillsdale Blvd. Suite 307 Foster City, CA, 94404 - Tel: 650-472-2208 Fax: 650-472-2210

CRYSTAL DYNAMICS eidos XBOX 360 nVIDIA XBOX

POSTMORTEM

THE STORY OF DEVELOPING BIOSHOCK IS AN EPIC ONE AND ISN'T easily expressed in 10 postmortem points. The team and the game changed remarkably over the course of development. A company was acquired. The team size doubled. The product focus changed from RPG hybrid to shooter.

It's easy to talk about the processes we used to develop the game, but it's harder to describe the creative spark that somehow managed to turn the most unlikely of premises (a failed underwater art deco utopia set in the 1960s) into a marketable shooter. It took a visionary to make the creative choices to guide the game, and an incredibly talented and hardworking team to bring that vision to life.

2K GAMES' BIOSHOCK

WHAT WENT RIGHT

1 **EVERY DEMO TELLS A STORY.** Demos were galvanizing moments for BIOSHOCK. They led to a unified team vision, identification of problems and solutions, external excitement, and internal support. For example, the project was signed after GameSpot ran an exclusive feature based on a single-room graphics demo.

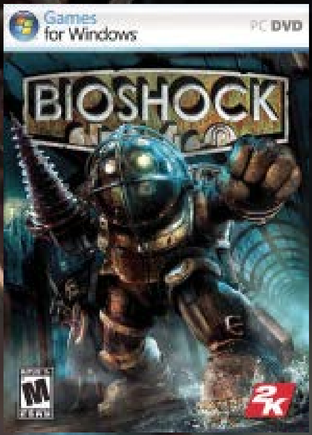
Since BIOSHOCK was a relatively unknown IP outside the game development community, the public's impression of it would be critical to building the buzz we needed to make it a commercial success. As a result, every time we took the game out in public, we put great thought into the message we wanted the demo to deliver and the level of polish of the presentation.

Our first public presentation was at E3 2006. We had developed a great deal of content before that point, but hadn't yet built a space that really demonstrated the game experience to our satisfaction. The E3 demo forced us to focus the whole team on what the user experience should be. We defined a message for the demo—player choice—and built a narrative around that message. Even though the experience was highly scripted at the time, it effectively demonstrated the feel of the game we wanted.

Another example of demo-inspired development was the "Hunting the Big Daddy" demo. Though Big Daddies and Little Sisters had been part of the game in some form since the beginning, initially

ALYSSA FINLEY *was the project lead on BIOSHOCK for 2K Boston. During her 15 years of experience in the game industry she has also worked as a lead programmer, technical director, and producer. Email her at* ***afinley@gdmag.com***.

GAME DATA



DEVELOPER
2K Boston and 2K Australia

PUBLISHER
2K Games

PLATFORM
Xbox 360 & PC

RELEASE DATE
August 21, 2007

DEVELOPMENT TIME
3 years

NUMBER OF FULL TIME DEVELOPERS AT PEAK
93 in-house developers, 30 contractors, 8 on-site publisher testers (see the sidebar on pg. 22 for details)

HARDWARE
PC; AMD Athlon X2 dual core or Pentium 4 Intel-Duo dual core processors; NVidia Geforce 8800 graphics cards; Xbox 360 dev and test kits

SOFTWARE
Microsoft Visual Studio 2005, Perforce, Xbox 360 SDK, Xoreax Incredibuild, Visual Assist X, Araxis Merge, BoundsChecker, Purify, VTune, 3ds Max 8, Photoshop CS2, ZBrush, Flash 8, SoundForge 8, Sony Vegas, Acid, Ableton Live

TECHNOLOGY
Unreal Engine, Bink, Havok, Fmod

NUMBER OF FILES
3,775

LINES OF NATIVE C++ CODE
75,8903

LINES OF SCRIPT CODE
187,114

having only large, constrained physics actors. This would have allowed us to spend much less time tuning the physics of individual objects while allowing the world to seem somewhat dynamic. However, doing so would have removed a huge level of interactivity from the game, so that decision was corrected relatively quickly.

In terms of design, we created a depth and density of game systems that fit into a game about character building and choice, but would not have been competitive as an FPS. Around the time that the game went into alpha, we took a hard look at that gameplay and realized that, although there were many choices, they weren't very compelling. This was because we hadn't been thinking as much about making a shooter as we should have, and many of our key interactions (weapons tuning, plasmids, length of AI engagement) were designed and tuned for a slower and more cerebral experience. To put it another way, nerdy RPG-like stat changes just didn't seem meaningful in the vibrant and dangerous world of Rapture.

Once we recalibrated the game to be more like a shooter, we simplified many of the deeper systems tremendously so that the user would be able to understand them. We also put more polish time into the core interactions of the game, such as the weapons, plasmids, and user interfaces. We ended up with fewer choices overall, but each one of those choices was infinitely more functional, understandable, and fun than the previous ones.

It was inevitable that we lost some progress due to these major corrections. But the team's ability to pull together and address the fundamental problems was amazing, and the results were well worth it.

TEAM BREAKDOWN

IN THE BOSTON STUDIO:

PROGRAMMER
1

ARTISTS AND ANIMATORS
15, plus 2 borrowed from Firaxis

DESIGNERS
6 in-house, 1 contract

AUDIO DEVELOPERS
2 in-house, 7 contract

PRODUCERS
3 in-house, 2 contract

TESTERS
13 contract, plus 8 on-site publisher testers

IN THE AUSTRALIA STUDIO:

PROGRAMMERS
12

ARTISTS AND ANIMATORS
10

DESIGNERS
5

AUDIO DEVELOPER
1

PRODUCERS
2

TESTERS
1 in-house, 7 contract

IN THE SHANGHAI STUDIO:

ARTISTS AND ANIMATORS
12

DESIGNERS
3

the player could confront Little Sisters directly without necessarily needing to dispatch the Big Daddy that protected them. During the development of this demo, the team discovered that with some polish and tuning changes the act of dealing with a Big Daddy could be a truly epic battle in itself. This led to the realization that Big Daddy battles should be the key to player growth, essentially providing a roving boss battle that players could undertake at a time and place of their choosing.

Another example is the graphical effects on the player's hands when using plasmids, which came out the first BIOSHOCK trailer created with Blur Studios. In that cinematic, the player uses a hypodermic needle to make his arm into a weapon; after the injection the protagonist's skin blackens and swells and angry hornets burst out of it to attack the Big Daddy. When working with Blur to develop the trailer, we knew that the sequence didn't accurately reflect the game's visuals, but we did it because it really captured the vibe of what the "genetic modification" part of the game was all about.

2 **COURSE CORRECTIONS.** One of the true successes of BIOSHOCK's development was our ability to identify and react when the game was not shaping up to become what it needed to be. For example, the first vertical slice prototype we built was an non-navigable linear corridor shooter that looked like it took place in an abandoned box factory. It didn't provide a compelling experience as either an RPG or a shooter. In response, we threw away that prototype and started again from scratch with the goal of building a single room that felt like the ruined underwater utopia we were trying to build.

First we did concept art passes. Once we got a concept that worked, we built it. Then we used it as a demo space. We used that single room (now Kashmir Restaurant in the first level of the game) as an artistic reference that guided us in creating an aesthetic unlike any other game on the market. (For more about the artistic style of BIOSHOCK, see the free art book download at www.2kgames.com/cultofrapture/artbook.html.)

Each department went through a similar crisis moment over the course of the project. These frequently came as the result of the demos, but not always. At one point, when facing a shortfall of programmers and an overflow of tasks, we proposed removing physics objects from the game entirely in favor of

3 **INPUT FROM OUTSIDE.** The first external BIOSHOCK focus test was meant to be a sanity check: to get a better sense of what was working well but needed polish and what wasn't working at all.

At this point we had already done one small round of internal focus testing with friends of friends, which had turned out mostly positive feedback. So, just after the first beta, the entire design team plus a contingent of 2K producers headed off to see how a group that knew nothing about our company or BIOSHOCK would react to the first level.

It was brutal.

The first level, they said, was overly dense, confusing, and not particularly engaging. Players would acquire new powers but not know how to use them, so they stuck to using more traditional weapons and became frustrated. They didn't interact with the Big Daddies, and they didn't understand (or care) how to modify their characters. They were so overwhelmed by dialogue and backstory that they missed key information. A few of the players did start to see the possible depth of the game, but even they were frustrated by the difficulty of actually using the systems we had created.

Based on this humbling feedback, we came to the realization that our own instincts were not serving us well. We were making a game that wasn't taking the initial user experience into account, and we weren't thinking enough about how to make it accessible to a wide variety of players.

CONTINUED ON PG 24

Mild-Mannered ALM, **Super Quality**

Developing quality software requires a heroic effort, from tracking thousands of the tiniest details, to keeping team communication flowing smoothly.

TestTrack Studio 2008 powers the application lifecycle, automating processes and keeping track of issues, change requests, test cases, and test results. With TestTrack Studio 2008, you have the tools and the time to prioritize, communicate, and track the status of your projects more effectively, without breaking a sweat.

Let TestTrack Studio 2008 do the heavy lifting–Be a superhero!

Seapine Software

Download your fully functional evaluation software now from www.seapine.com/gd08

The 2K Boston BIOSHOCK team.

CONTINUED FROM PG 22

After the focus test, we went back to the drawing board for the entire learning sequence of the game. We scrapped the gameplay in the first two levels entirely and re-architected them to be a much slower paced experience that walked the player through the more complicated gameplay verbs, such as "one-two punch"—combining weapons and plasmids. We changed the medical pavilion from having sandbox-style gameplay to using a series of locks and keys that were set up to ensure that the player knew how to use at least a few key plasmids. And we made a development rule that future changes would be data-driven, not based solely on our own instincts.

After the first round of changes, we had two rounds of internal 2K play testing to gather more data about the user experience, releasing builds of the game to several 2K studios and soliciting feedback about how far people got and which weapons or systems they enjoyed. We received feedback from 2K game analysts, Microsoft, and a few other advisors.

When we brought the demo back to focus testing, which was barely a month before we were (then) scheduled to complete the game, the experience was very different. Although players still got stuck and frustrated at various spots, they understood the game systems and saw the potential inherent in them. While we still had work to do to make the game more accessible, at least now the problems were much more easily solvable.

4 **SMALL EMPOWERED TEAMS.** While developing our first internal demo, we realized just days before completing it that it was on the wrong track. By that point it was too late to take on all the problems in the demo, but we decided to try to improve the core interactions. We used a small, focused strike team approach to target and solve AI problems, choosing one problem at a time, analyzing and tackling it, then moving on. Although this approach wasn't enough to salvage the original demo, it was recognized in our internal postmortem of the demo as an effective process that we should do more often.

One of the most visible successes of the strike team system is the tuning of the weapons of the game. All the weapons had been in and working for several months, but as the game got closer to content lock, they still weren't feeling as good as they should. To tune each weapon, a team consisting of one designer, an animator, a modeler, a programmer, the effects specialist, and an audio designer held a kickoff meeting where they analyzed and brainstormed about each aspect of a single weapon. They came up with a task list for each team member, went off to work for a day or two on their tasks, then came reviewed all the results. When they were satisfied, they moved on to the next weapon.

Over the course of development, we created multidisciplinary strike teams to work on a wide variety of problems, including AI, animation, visual effects, and cinematics. The results of those teams were universally better than the previous non-iterative process.

5 **TALENTED PEOPLE, FLEXIBLE STAFFING.** BIOSHOCK was initially scoped to be developed in about two years with a small team of 30 people—25 in Boston focused on gameplay and five in Australia working on the core engine. As the team completed successful milestones and demos, and made strong cases for more development resources it became clear that we needed to tap into the Australian office.

Initially, Australia was intended to supply a small core technology team that worked on the renderer, engine, and core tools and processes for console development. The Australians had a tremendous impact on development because by taking care of the core engine and pipeline tasks, the Boston programming team was free to focus on gameplay systems and production.

One of the fastest and easiest ways to staff up any newly-opened position on the BIOSHOCK project was to pull from the Australian team. By the time BIOSHOCK went gold, almost everyone in the Australian office had worked on the game in one way or another.

The huge advantage to using the Australian team resources was that they already knew the engine and the game, and had easy access to the core technology team. They came up to speed incredibly quickly, and could be productive almost immediately upon getting project tasks. And although the time difference made communication a challenge, it also meant that critical bugs could be worked on literally day and night.

WHAT WENT WRONG

1 **EVOLVING PRODUCT POSITIONING.** The spec of BIOSHOCK changed so much over the course of development that we spent the majority of the time making the wrong game— an extremely deep game, and at times an interesting one, but it was not a

groundbreaking game that would appeal to a wide audience. We knew from the start that we'd have to make late changes to really bring the game to life—we had even built our original schedule to allow for six months of finalizing—but the amount of change that we ended up needing seriously exceeded our remaining schedule. Ultimately, we were very lucky to get an extension in the eleventh hour.

Part of the reason for the late course change came from not having our internal product message clear from the beginning. BIOSHOCK had initially been positioned as a hybrid RPG FPS. The decision to reposition the game as a focused FPS came later, after our initial production phase in summer of 2006. Had we been working with an FPS mentality earlier, we could have made better use of our time.

Another contributing factor to the late switch was that the game had been more or less proceeding according to plan throughout development, so there didn't seem to be any emergencies that needed intervention from higher levels of management. Milestones were completed, goals were met, development seemed to be proceeding uneventfully. But as the game neared alpha, key people began looked more closely and saw that BIOSHOCK wasn't on track to become an accessible and marketable game.

As mentioned in the first What Went Right point, the real turning point for BIOSHOCK came when we had to present the game to the outside world, which forced us to carefully consider the story and takeaway message. In retrospect, we should have tried to develop some of that thinking sooner.

2 **NARRATIVE CONTENT DEVELOPMENT HAPPENED LATE.** We had many drafts of the story over the course of development, but the final draft turned out to be an almost complete rewrite. To make matters worse, we failed to fully exercise the narrative production path in early versions, so once the final draft was complete and recorded, many implementation and pipeline problems appeared for the first time that should have been caught and resolved earlier.

The core issue was that a giant pile of content came online well past beta, and the team had to scramble to get that content correctly installed while also fixing bugs. Competing demands for time and resources meant that, unfortunately, some of the important narrative details of the game weren't created until the final rewrite, and therefore required quite a bit of work to retrofit them into an existing game.

To add to our woes, the first focus test feedback on the narrator's voice came back extremely negative. People found the character extremely off-putting, so we recast the part at the last possible moment. On the positive side, this allowed us to refine several areas of the game (including the intro sequence) to

ensure that the player knew what to do at the right time. On the other hand, it was difficult to get all the content in, debugged, and polished in the remaining timeframe.

3 **SCALING VISION TO TEAM SIZE.** Our goals and vision pretty consistently overreached our production capacity. Ideas that started out small turned out to require a tremendous amount of coordinated support to reach a polished state. Although we were able to add resources regularly and make some cuts late in the game, our ability to plan for how much work it would take to bring any single idea or space from concept to completion was poor.

In addition, we didn't have an effective internal review process set up until very late in the development cycle. We would work on levels to the definition of a milestone, get feedback, and then set it all aside for a while or leave it in the designer's hands to polish. It wasn't until we created a more regular review cycle, where all the key players sat in one room and watched the gameplay session and someone logged all the bugs, that we were really able to define the amount of remaining work to bring a feature or level to completion.

The ultimate reason we were able to pull off the game we made to the level of polish we did, was the sheer dedication of the team working on it. Even though we had padded the finalizing schedule, we still far exceeded it. It's to the team's credit that they stepped up to the challenge with incredible dedication. The final crunch period on BIOSHOCK was long and hard. People who had been pacing themselves for six more weeks of work had to reset to three more months of work rather suddenly. Had we understood our polish cycle requirements sooner, or had known about the additional time earlier, we could have paced ourselves better.

4 **INEFFICIENT PROCESSES AND TOOLS.** Many of the processes and tools we used to develop BIOSHOCK were inefficient or confusing in implementation, leading to slow iteration cycles and bugs. Using a modified version of the Unreal engine, which the team had already used to ship two previous games, gave us a huge head start in developing BIOSHOCK. The gameplay team was able to mock up a playable version of the core game mechanics in just a few months, and the team's familiarity with the tools allowed us to get new gameplay spaces up and running quickly.

However, the ease and familiarity of the workflow often led us to accept a solution that was faster to implement but slower to use rather than taking the time for a more efficient implementation. For example, there was no good convention for how to name script actions. Depending on the system, one script action might be called "Change<SystemProperty>" while another would be called "Set<SystemProperty>" or "Modify<SystemProperty>". With hundreds of scripting actions available, designers often spent way too much time searching for the right tool to use. This could have been avoided with a scripting code standard.

The content baking process for the console was time-consuming and difficult to troubleshoot. Frequently the only way to either identify or resolve a bake problem was to re-bake at the cost of up to an hour of work, and if the tools were actually broken in some way, it would take at least another bake cycle to be able to work effectively again. Once we reached crunch time, it was extremely painful to have to wait for the bake process to complete when people could have been working productively instead. We should have put more energy and time into speeding up the bake process sooner.

5 **POOR DATA COLLECTION.** One of the most frustrating things about our decision to be more data-driven in tuning the game was the lack of actual good data to base that tuning on. Our game log system was barely adequate to analyze single play-throughs and became completely unwieldy when trying to analyze a single log file containing data from multiple play-throughs. We had no good methodology to define what information was logged and at what level of detail, so the job of parsing out the logs into understandable "gameplay metrics" was painful, slow, and ended with inadequate results. To further complicate the problem, most people in the office used shortcuts or cheat codes at the start of a level rather than playing from the beginning of the game, which caused us to base a tremendous amount of early tuning on a shaky set of assumptions about how players would choose to build their characters.

BLOCKBUSTING

Our goal when we set out to make BIOSHOCK was very clear. We wanted to get to the next level, moving beyond our suite of critically acclaimed games to make a blockbuster. A lot of factors aligned to make this possible: the commercial backing of 2K; the game design knowledge we'd acquired from building SYSTEM SHOCK 2; the technological familiarity with our UNREAL-based engine that we'd built with previous games. But we still had to figure out how to make it all big—blockbuster big.

A lot of our problems came from underestimating how big the task of making a triple-A product for multiple platforms and multiple regions really is. And other problems came from over-estimating our capacity to solve those problems using our existing procedures and staffing levels.

If there's an over-arching theme of our development, it's that we, like many other developers, believe that ultimate success in this industry comes from iteration. You have to build, evaluate (and have others evaluate) and be prepared to throw things away and rebuild. The products we make are just too complex and our industry reinvents itself too rapidly to do anything else. But we believe that if you are truly prepared to turn a critical eye on your own product and honestly respond to that criticism you'll get quality at the end. As to whether you get a blockbuster, only time will tell.

jobs.ea.com

SIDE EFFECTS' HOUDINI 9

BY DAVID MARCH

HOUDINI 9

★★★★

STATS
Side Effects
123 Front St. West,
Suite 1401
Toronto, Ontario
Canada M5J 2M2
416.504.9876
www.sidefx.com

PRICE
$1,995

SYSTEM REQUIREMENTS
Windows and Linux operating system. Support for 64-bit Windows and Linux available. 512MB memory required. AMD or Intel compatible processor (PIII or later). 1GB disk space. Workstation-class OpenGL graphics card with a minimum resolution of 1,024x768; color depth of 16-bit and support for OpenGL 1.2 or later.

PROS
1. Construction history on crack.
2. Robust channel editing.
3. More familiarity in UI for new users.

CONS
1. Would like to see expanded Sub D modeling tools.
2. Would like quick click normal mapping tools.
3. Small community of users in game development.

BEFORE THIS REVIEW, I HAD ONLY HEARD here and there about Houdini, either in passing or when some 3D production magazine mentioned that the application was used on this or that big-budget film. Maybe it's just my particular circumstances, but I don't personally know any game developers who use it for modeling, texturing, or animating. Houdini has a reputation for being a tool only used by film artists for complex algorithms in special effects work. But there's no reason game developers shouldn't adopt it for special purposes—or for the whole enchilada, if they're not already married to one of the three packages that dominate our industry.

IF YOU LIKED THE ORIGINAL, YOU'LL LOVE THIS!

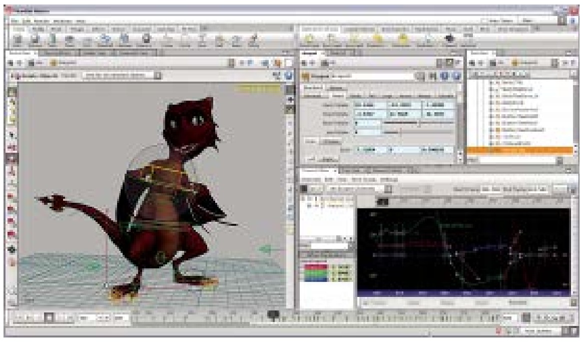
After I installed the software, I really had no clue what to expect. All I had read so far was a brief caption on Side Effects' web site about how the company had changed the UI and "workflow enhancements" to be way more "artist-friendly."

That being said, I decided to give Houdini 9 the old newbie test, and promptly skipped over the UI quick start videos. And there it was ... Maya. Just kidding. I mean Houdini. But it really did remind me of Maya's UI, with the shelves at the top and similar icons, and buttons on the right-hand side. Even the panels on the left side have a similar feel, too. The likeness to Maya made me feel completely comfortable with Houdini from an artist's standpoint. If Side Effects did this intentionally, I say, "Brilliant."

Without having seen Houdini's previous versions I cannot compare how much more artist-friendly the UI is now, but Side Effects sure has hit their mark with the first impression a new user gets from the design.

I immediately clicked the left mouse button while holding down the Alt key, then the control key, and then the shift key, saying to myself, "Hey wait a minute. How the heck do I navigate?" I felt quite silly shortly thereafter because you have to use the right, middle, and left mouse buttons to dolly, pan, and zoom. Yep, it's that simple.

Next I clicked the box icon and—shazam!—with no surprises there was a basic box with a gizmo around the object for translating and rotating. But after clicking around some more I got stuck.


Houdini 9's interface will be comfortable for users already familiar with Maya.

TIME FOR TV

I decided it was high time I watched those videos that I first saw upon booting up the program. Then, remembering that I had also seen a link on the web site to video tutorials, I instead jetted over to the Houdini 9 interface lessons where I found about 13 bullet tutorials with links to even more.

The only thing I initially stumbled on was the fact that you have to hold the space bar down to get to the point where you can move and dolly around in and out of the object mode. Even though I didn't like doing this for the first few minutes, it became very natural after a short time.

A NOD TO THE NODE

After watching quite a few tutorials and playing around with Houdini, the one feature that stands out from other 3D applications is the node-based workflow. It's construction history on crack. Every decision you make in creating a scene is created as a "network node" that you can come back to and edit.

The node workflow is extremely powerful, as it allows you to manipulate the changes you've already made and gives you much more control to explore new techniques. Basically it lets you back up and make a change to what you've created, plotting out your entire process.

I also felt quite comfortable with simple node tasks, though the type of artwork I was creating was very rudimentary. However, I can immediately see how the node-based workflow would benefit video game artists in the thick of things, creating complex models without remembering exactly how they got there and wanting to change one of the elements without disturbing the rest of the work.

I also really like a minor feature in the node window that color-codes different icons to represent the state of an object, such as "active node" or "node change." After toying around with Houdini, I must say I wish every 3D program had such a useful construction history— Houdini has one of the best, if not the best. Side Effects is clearly one of the few companies in the game market that has evaluated what works in the other software packages and has added those features into its own tools.

SPECIAL EFFECTS

The special effects abilities are what really piqued my interest in this software, but being a video game developer, I wondered, "What am I really going to do with them?" Maybe I could use Houdini for pre-rendered cinematics for cutscenes or pre-visualizations. But based on my professional experience, those things usually don't happen very often in games (or at least not the games I've worked on). Still, it's cool to have them on hand.

Judging by what Houdini has produced in films and by reading about it in production magazines, I'd say it has a one of the more advanced particle systems on the market today. Version 9

OUR RATING SYSTEM :

★★★★★ EXCEPTIONAL ★★★★ GREAT ★★★ FAIR ★★ POOR ★ UNFORTUNATE

Houdini 9's fluid dynamics solver allows liquids, fire, and smoke effects to be used with rigid bodies and particles.

also comes with an updated fluid dynamics solver, which can simulate liquids, fire, and smoke. Not only can you simulate these effects, but you can also use them with rigid bodies and particles.

ANIMATION AND TEXTURES

Houdini has a robust channel editor called CHOPS. The system gives you a world of control over the data in your channels, including layering animation, which is a must with any animation software. If you can't layer with your current animation software, it's time to get new software.

In addition to layering and blending motions, Houdini is procedural in nature so there are many other possibilities for things you may want to do. You can layer, shift, and mix motions and then blend them to any channel or mix them with other premade clips. And you can simulate motion dynamics on top of it.

All the basic texturing tools are right where they should be in Houdini, including one for pelt mapping. Pelt mapping is that tool most 3D apps added about a year or two ago that lets you to cut a seam and flatten your UVs, as if they were skinned animals, and slap your textures right on.

One thing that seems to be missing, though, is a quick way to create normal maps, although Houdini does have occlusions and displacement maps. But even in these areas, I felt like Houdini could use a bit more work.

IS IT A CONTENDER?

Houdini has covered all the bases, with a bit of an extra kick in the node-based workflow and special effects. Still, whether game developers decide to adopt the tool will depend on what they are trying to execute in a project.

I actually tried to dream up a few scenarios in which Houdini's node-based workflow would be unbelievably powerful for a game engine, but I struggled to come up with anything solid. However, it's quite obvious why Houdini is used for pre-rendered cinematics.

Technically, if your game world was set up to go straight from your 3D application into your engine, it could be quite useful with some of its procedural workflow. I don't know of any engine that could handle such a large input of work, but it could be created somehow to batch process out elements. And I do know of some engines out there that can take files straight from one 3D application right into the engine. (Basically, the 3D application is the engine, sort of.) Houdini seems like a candidate that could have quite some potential for this type of work.

Even though Houdini has all the nuts and bolts of any other major 3D package, I find it hard to imagine the game community jumping ship from their existing staples. Yet with Houdini's improved user-friendly interface and solid set of tools, it can stand on its own for sure. I can also see it being used for specific specialized purposes within a studio—for example, making reusable tools that can automate redundant tasks.

DAVID MARCH *is lead animator at Irrational Games in Australia. Send comments to him at* **dmarch@gdmag.com**.

product news

SUPPORT FOR PHYSX SDK EXPANDED

AGEIA

Ageia Technologies expanded its licensing and support model for the PhysX SDK, in which the SDK binaries will be offered royalty-free, while the SDK source code is licensed at $50,000 per application. Their support model covers game profiling and optimization for console, multi-core and PPU, augmented with both off-site and on-site support options. The PhysX software supports PC, PlayStation 3, Xbox 360, and Wii.

www.ageia.com

WII SUPPORT ADDED TO GAMEBRYO

EMERGENT

Emergent Game Technologies' Gamebryo game engine now supports the Nintendo Wii, enabling the development of titles for Nintendo's console using the latest version of the engine also used on Bethesda's OBLIVION.

Emergent also unveiled an upgrade to its Gamebryo engine focused on performance and feature enhancements for next-generation platforms. The latest release aims to enable dramatic rendering results while attempting to significantly reduce development resources required for production.

www.emergent.net

CARRARA 6

DAZ 3D

Carrara 6 is DAZ 3D's modeling, animation, and rendering application, with the latest version adding a host of features including the handling of morph targets, conversion of surface materials and rigging. New enhanced remote control grants users simultaneous control over multiple translation and transform dials.

Also new is non-linear animation, dynamic hair, and displacement modeling—in which the user can paint detail on a model using free-form brush tools.

Symmetrical modeling has been added, which allows content creators to edit both sides of a symmetrical object at the same time using a variety of editing tools; and in-scattering of light, which when combined with the new Ocean primitive, provides water surface and wave simulation.

www.daz3d.com

CMP
United Business Media

If you are going to attend one game industry event in 2008, this is the one.

GameDevelopers® Conference 08

Insider Tip:

The event dates are earlier this year so book your hotel asap!

GDC08: Transform Your Game.

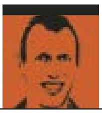
New for GDC08!

Expanded Summit Lineup:
- Casual Games
- Independent Games
- Game Outsourcing (new!)
- Serious Games
- Worlds in Motion (new!)

Simplified Pass Structure
- All Access
- Main Conference
- GDC Mobile
- Summits and Tutorials
- Expo

myGDC: Our New Professional Networking Resource
- Set up profiles
- Search and connect with thousands of game development professionals
- Share projects, images, and ideas

Save up to 30% when you register by January 16, 2008. www.gdconf.com



EMBEDDED SCRIPTING

A Closer Look at Ruby

AFTER DEVELOPING YOUR GAME ENGINE in C++ and implementing game content using C++ for a while, you might start to consider what options you have for using scripting languages. This article looks at various options for adding scripting to a game, with a focus on embedding Ruby into an existing engine.

Implementing game logic in C++ can work quite well in small games, but in large games, the rebuild time can become an issue. You may have to spend up to five minutes rebuilding and reloading the game just to see the effects of one change. Scripts, on the other hand, don't need to go through the entire compile and link process. You can modify a script and re-launch the game in seconds.

In an ideal situation, you would be able to make changes to the script and see the results in real time. This is not always possible, but it is something that should always be kept in mind as a goal.

A large proportion of one's time in game development is spent iterating possible ways of doing things and fine tuning them. Anything that can reduce the cycle time in this feedback loop is very important.

ROLL OR ADAPT

Do you roll your own scripting language or adapt an existing one? There are pros and cons of both approaches. If you roll your own, then you have full control over aspects of the script language that could affect your game's performance, specifically the speed of execution and how memory is allocated. This may be very important if your target platform has limited resources and if a more compact and targeted scripting language would work for your application.

Rolling your own also has the advantage that you will be very familiar with the inner workings of the script compiler or interpreter, and this will aid you greatly when it comes to debugging issues that have to do with low-level problems with scripts or script objects. The downsides of rolling your own are 1) the large amount of time it takes to get a fully featured language and 2) the inevitable number of bugs involved in developing something from scratch. Maintenance of the code may also be an issue if your script programmer leaves; then extending and debugging the script language may become problematic and messy.

“Adding a scripting language to an existing game engine can greatly aid the development process.”

If you choose to adapt an existing language, you benefit from getting the language up and running with minimal work. However, there's still a significant portion of code to be written: the interface between the script engine and your game engine. But once this interface code is written, you'll have a powerful scripting language with an extensive set of features and generally with a lot of documentation on how to write code using it. Using an existing language also allows you to hire script programmers who are already highly proficient in the language, avoiding a lengthy ramp-up time for a proprietary language.

CHOICE OF LANGUAGE

There are several available scripting languages than can be embedded in a game engine. Lua, Perl, Python, and Ruby are some of the more popular ones. Each comes with its own set of benefits and problems.

Perl and Ruby are quite well known, as they have been used for some time for writing web server applications and general-use scripts. Perl is the older of the two and hence is more widespread, but Ruby is increasing in popularity and is considered something of a “modern” language. Perl usually executes faster than Ruby, but both suffer from being slightly overweight and from sometimes taking up a lot of memory.

Lua and Python are more lightweight languages, sacrificing some power for simplicity, and possibly speed of execution. Lua in particular has been a

MICK WEST was a co-founder of Neversoft Entertainment. He's been in the game industry for 17 years and currently works as a technical consultant. Email him at mwest@gdmag.com.

popular choice for game scripting and is used in several commercial games including WORLD OF WARCRAFT and FAR CRY. Lua is a popular choice for defining the user interface.

Ruby is a powerful language with a very active user community. There are several books on the market which discuss how to program in Ruby, and many libraries exist for performing a wide variety of functions. Unfortunately, Ruby is a bit lacking in documentation regarding the actual internal workings of the language. If you want to incorporate Ruby into your game engine, you would benefit from a little understanding of what's going on under the hood. Here we'll take a very brief look at what's involved.

EMBEDDING RUBY

The basics of embedding Ruby into another application are relatively straightforward. At the time of writing the latest stable version of Ruby was 1.8.6, on which these examples will be based. Ruby is implemented in C and can be built into a DLL using the supplied makefile, and then linked in with your project. However, if you're planning to incorporate Ruby at a fairly low level, you may want to build the source directly into your project, which will let you customize the language, remove parts you don't use, and have finer control over debugging.

The first stage in embedding Ruby in your game is simply to get it to compile and link with your code. Then you need a mechanism for loading and executing scripts. The very simplest way is shown in Listing 1.

The first two lines initialize the Ruby system. The third line tells Ruby that we're running from an embedded script. The fourth line actually loads and compiles the script "hello.rb."

LISTING 1

```
RUBY_INIT_STACK
ruby_init();
ruby_script("embedded");
rb_load_file("hello.rb");
ruby_exec();
```

Loading and executing a Ruby script.

While Ruby is an interpreted language, it's actually compiled into a tree of "nodes" when the source file is loaded. This compilation can also happen off line if you don't want to ship with your source code. Here the call to rb_load_file() will load the source file and compile it into the Ruby node tree. It will also set the global variable ruby_eval_tree to point to this newly loaded set of nodes.

"Using an existing language also allows you to hire script programmers who are already highly proficient in the language."

The call to ruby_exec() then executes this compiled program, and then returns control back to your code.

With this simple example we can look at some of the immediate practical implications of using Ruby as an embedded language. The most obvious one is the use of memory.

Incorporating the full Ruby compiler and interpreter into a code base may add more than 500K to the executable size. This alone may be a prohibitive amount for certain platforms with limited memory. But for something like a PC game, or even a PlayStation 3 or Xbox 360 game, it's not an unreasonable amount—about 1 percent of the available memory. You're weighing several factors here, with more memory used, you get more power and flexibility in your scripting language.

LISTING 2

```
C++ Code:
   rb_funcall(Qnil, rb_intern("simple1"),
          1, (125<<1)+1);

Ruby Code
   def simple1( arg1)
     puts arg1
   end

Output:
   125
```

C++ calls a Ruby function directly, passing a parameter.

COMMUNICATION

Now that you've got Ruby to compile and run, what next?

Ruby comes with a fully featured C API that allows you to specify all manner of ways of communicating between Ruby and your C/C++ code, calling functions and passing parameters. The simplest way is show in Listing 2.

Everything in Ruby is an object. All Ruby objects are represented in C/C++ by a "VALUE" data type. A Ruby VALUE is a loosely typed object, and pretty much every Ruby function in the C API takes parameters of type VALUE, regardless of the actual nature of the object being passed. A VALUE can be anything from an integer to an array or hash table of other VALUE objects.

A VALUE is not an object in the sense of a C++ object. In C++, a VALUE is (usually) a 32-bit word, which typically is a reference to the actual object. However, a VALUE can also be one of several compact data types, the most common of which is a "small integer"; in this case, the value is simply shifted left and ORed with 1. This is what's happening in Listing 2.

It's a bit of a hack to illustrate what's happening at a low level when passing simple parameters to Ruby functions. You can do the same thing with the INT2FIX(n) macro. Encoding a concrete data object in a VALUE is a much more efficient way of manipulating objects (such as small integers) than requiring all objects be represented by reference.

Suppose you wanted to pass a string to Ruby. You would have to make an object of string type and pass that. Listing 3 demonstrates this. A new Ruby object is created called "ruby_string." Again, this is just a VALUE type, so for C++, it's just a 32-bit word (an unsigned long, to be precise, which is generally a 32-bit word). Passing this string to the Ruby function is exactly the same as passing the small integer, and the Ruby function

LISTING 3

```
VALUE ruby_string = rb_str_new2("Hello World");
             rb_funcall(Qnil,
rb_intern("simple1"), 1, ruby_string);
```

Passing a string object.

simple1 does not care if it's a string, or an integer, or a 20MB array of red-black trees, or whatever. This loose typing makes Ruby very flexible. The internal representation using 32-bit VALUES makes handling Ruby objects reasonably lightweight on the C++ side.

Functions (or methods) and variables are all referenced by an ID type. Like a VALUE, an ID is a 32-bit word. An ID represents a Ruby symbol or name. The second parameter to rb_funcall here is an ID, and in this case it's the ID of the function "simple1." This simple way of referring to functions makes it very easy to reference them and object methods.

Going the other way (calling C++ from Ruby) is slightly more complicated. To call a function from Ruby, we first have to tell Ruby that the function exists and what its name is. See Listing 4 for an example. First, we have the function we're going to use.

The function load_asset takes two parameters: an ID and a VALUE. The ID is the receiving object, which in this case will be the Ruby special value Qnil (4, in this implementation). The VALUE is whatever is passed in by the calling Ruby script. It could be any Ruby object. From the point of view of C++, it's just an unsigned long, and we'll need to call a Ruby function to get the contents—hence the call to rb_string_value_ptr. We can return a Ruby VALUE back to the Ruby calling function, but in this case we just return Qnil.

Now that we've got our function, we need to register it with a call to rb_define_global_function, which just takes as parameters 1) the name of the function (which need not be the same as the C++ name), 2) the address of the function (using the RUBY_METHOD_FUNC macro to cast to the correct type), and 3) the number of parameters (which is over 1, in this case).

Then we can call our function from Ruby. From C++, we call the Ruby script load_some_assets, which simply calls the C++ function load_asset() ten times, with a different string parameter each time, and then returns back to C++.

This barely scratches the surface of what's possible in interfacing Ruby and C++. You can also extend Ruby objects by adding C++ member functions. You can give Ruby members access to your C++ objects. And you can even do all this automatically using a pre-processor such as SWIG. These examples should give you a sense of how simple it is to incorporate a scripting language into an existing code base. You can couple them as loosely or as tightly as you want.

PERFORMANCE ISSUES

In games the two perennial performance issues are memory and frame rate. With a language like Ruby, these two issues combine in one particular issue: garbage collection.

In Ruby, as in many scripting languages, you do not specifically allocate and free memory. Instead, memory is allocated automatically for objects as needed and is also freed automatically when it's no longer being used, at least in theory. In practice what happens is unused objects are kept around until Ruby detects a need for "garbage collection," which happens when a heap gets full. You could be merrily sailing along, when all of a sudden the heap fills up and a massive garbage collection operation is initiated which takes a frame or so to process, resulting in a frame rate glitch.

All this is highly dependent on several factors, like how you use objects, how big they are, and how big your default heap is. To avoid garbage collection going off at an inopportune time, you can call it yourself manually with the function rb_gc(). You could call this every frame, giving you a fixed overhead, which might be quite reasonable if you don't have an immense amount of Ruby objects. You could also defer it until a less noticeable point in the game, such as a level transition. If you don't use it in the real-time potions of your game, then it's less important.

A GEM IN THE ROUGH

Adding a scripting language to an existing game engine can greatly aid the development process by allowing rapid iterations of game logic that would ordinarily require a rebuild of the C++ code. Adding a script engine creates an immediate resource loss by taking up additional memory, and the logic that was in C++ will now be running slower in script.

Generally, these are manageable problems, and the benefits of the additional flexibility of scripting should greatly outweigh the negatives over the long term.

Ruby is a mature and powerful scripting language. It might be too heavyweight for some applications, but it also could work very well in games that can afford to spare some memory and CPU cycles. Integrating Ruby with C++ is very straightforward and should allow you and the content creators on your team to begin using the language for scripting after only a few days' work. ⁙

LISTING 4

```
// C++ Function to be called from Ruby
// with one parameter
VALUE load_asset(ID recv, VALUE asset)
{
   printf ("Loading asset %s\n",
      rb_string_value_ptr(&asset));
   return Qnil;
}

// Code to register the C++ function
   rb_define_global_function("load_asset", RUBY_METHOD_FUNC(load_asset), 1);
// And run a script to test it.
   rb_funcall(Qnil,rb_intern("load_some_assets"),0);

# Example Ruby script calling C++
def load_some_assets
 for i in 1..10 do
   load_asset "asset_"+i.to_s+".jpg"
 end
end
```

Calling C++ from Ruby, which is called from C++.

RESOURCES

Thomas, D. and Hunt, A. "Extending Ruby," in *Programming Ruby: The Pragmatic Programmer's Guide*. Reading, Mass.: Addison-Wesley, 2001. (Includes the Ruby C API documentation.)
www.rubycentral.com/pickaxe/ext_ruby.html

Gutschmidt, T. *Game Programming with Python, Lua and Ruby*. Boston, Mass.: Premier Press, 2004.

MONTREAL INTERNATIONAL
GAME SUMMIT 07

November 27-28, 2007
Montreal Canada
Palais des congrès

Where the Game Begins!

Register on line now to Canada's Main Event for Game Development

New in 2007: The **Business Lounge** and the **Serious Game Symposium**

DISCOVER:

- The MIGS Exhibition Zone
- Some forty internationally renowned speakers:

YOSHIAKI KOIZUMI, Nintendo;
JONATHAN BLOW, Independent Game Developer;
DAVID PERRY;
+ Ian Bogost, Persuasive Games; George Borshukov, Electronic Arts WW Studio; Mike Chrzanowski, Vicarious Visions; Don Daglow, Stormfront Studios; Jon Goldman, Foundation 9; Clint Hocking, Ubisoft; Julien Merceron, Eidos; Samuel Rivello, Neopets; Duncan Wain, and many more...

w w w . s i j m . c a

Organised by : Alliance numériqc
Quebec's digital industry network

Presented by: Autodesk®

Mega Sponsor:

Artificial Mind & Movement · Beenox · EA Montréal · Humagade · Nintendo TOO MUCH FUN · Ubisoft™

STEVE THEODORE

EVERY PICTURE TELLS A STORY

Modeling and Narrative

WE GEEKS OF A CERTAIN AGE EXPERIENCED a little thrill of nostalgia during the blizzard of pre-*HALO 3* marketing. For most industry folks, the commercials featuring Stan Winston's mammoth "Believe" diorama were an intellectual exercise: a chance to speculate about the end of the trilogy, to nitpick about the details of the beautifully executed hand-built models, or to debate the marketing merits of the ad campaign. (See Figure 1.)

For the more retro among us, though, the mockumentary footage showing the painstaking modeling work resurrected some pungent memories, the lemony smell of polystyrene glue, the slimy slide of water-release decals, and the tedium of filing mold-marks off of various Panzer sprockets and Mustang manifolds.

The plastic modeling scene of 25 years ago might seem irrelevant to a magazine that specializes in whiz-bang next-gen game graphics. The technical challenges of modeling in plastic and in polygons are completely different, but the artistic demands of level design and asset modeling are actually quite similar to those facing diorama builders and other real-world model-makers, like effects houses and set dressers.

Physical and digital modelers both need to engage their audience in ways that differ from most of the other arts. Temporal media like animation or comics tell stories by controlling the audience's experience of time and sequence.

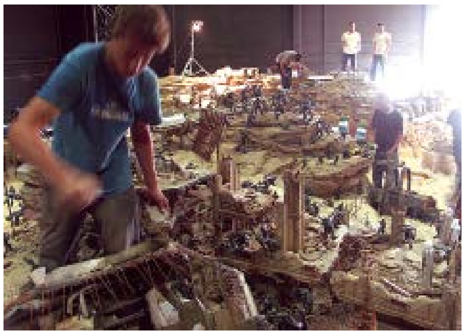


FIGURE 1 **Stan Winston's "Believe" diorama—the missing link between modern game graphics and old-school modeling skills.**

Traditional graphic arts like painting and illustration set the stage with a 2D composition that guides the eye and shapes the viewer's sense of occasion. Physical and virtual modelers, however, must both cope with a viewer who can inspect the finished piece from any angle or distance. Of all the disciplines, modelers face the toughest challenge in reaching the audience emotionally. Just as animators still find value in the works of Seamus Culhane or Preston Blair (even if they've thrown away their pegboards), modelers should ponder the lesson of the pioneering modelers of the 1970s and 80s, artists like diorama builder Shep Paine, miniaturist Bill Horan, or ILM's Lorne Peterson—even if we never need to know the right way to vacuum-form a new Messerschmitt canopy or how to unblock a dodgy airbrush.

PLASTIC TO PIXELS

The central task of any modeler, physical or virtual, is to give a static object or scene enough life that it can reach the audience emotionally without the kind of framing devices that other media have. Most of us deal with subjects that are basically anonymous: mass-produced vehicles, manufactured goods, generic architectural spaces. Only a fraction of our work is devoted to unique capstone designs that are strong enough to capture the imagination based on design alone. For every Death Star or TIE Fighter there are miles of faceless corridors, inevitable period vehicles, and necessary but uninteresting bric-a-brac.

Thus every game modeler faces the same problem many times: How can I make my Sherman tank different from all the other 3D Shermans out there? How will my shipping container yard stand out from all the other container yards? Even if I'm lucky enough to work on a strong, unique design, how can I anchor that design in physical reality for the players? Those questions would be equally familiar to earlier generations of model builders.

Real-world modelers and scenarists try to compel the audience by presenting objects or spaces as slices of living history, not static images. By now most game artists have learned the obvious truth that the world isn't factory fresh, and most modelers today add a dash of

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

SHEP PAINE DIORAMA IMAGES COPYRIGHT MONOGRAM 1972

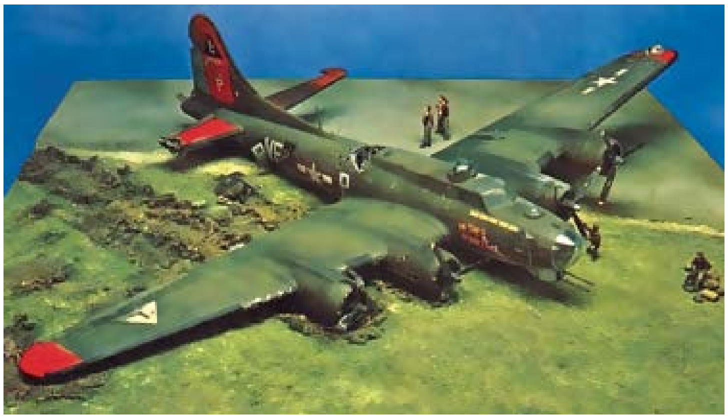


FIGURE 2 **Shep Paine's Monogram B-17 diorama is a study in the art of bringing backstory and personality to a static model.**

noise or a bit of wear as a matter of course. The classic models of artists like Paine and Francois Verlinden, or the miniature work in the pre-CG era *Star Wars* films, take this principle several steps further. They are built around details that can turn an empty room or a simple object into something like a character with a past and a distinct personality. Great models never let the viewer forget that every object or scene had a past that made it different from all the others of its kind. Whether it's a lucky pet name chalked on a tank turret, some jerry-rigged repairs on the Millennium Falcon, or just the litter of paperwork and coffee cups on a desk, classic modelers always remind the viewer that what they see is a moment in an ongoing story—not just a tank, a spaceship, or an office.

"Physical and digital modelers both need to engage their audience in ways that differ from most of the other arts."

BACKSTORY

Shep Paine's famous Monogram diorama series is a great example of how details can be chosen to create miniature narratives. The B-17 kit (see Figure 2) in particular, shows storytelling and modeling welded together into a single process. The diorama centers on a crashed bomber, a subject that could easily find a home in many game settings. The execution, though, shows how the artist's careful thought has transformed a simple premise into a unique form. Rather than simply layering on scorch marks and gibs, Paine has imagined the entire crash sequence: flak over the target, a limping flight home, the failure of the plane's landing gear(see Figure 3), and a final skid off the runway into the muddy verge of the landing field.

That story drives the details of the final model. Everything, from the streaks of oil smoke on the wings to the way the propellers were bent back asymmetrically by the climactic belly flop, helps support the background story. Instead of a forgettable icon that simply checks the "plane crash" checkbox, the imagined scene invites the viewer to envision the off-screen drama, as well as appreciate the final result. The viewer doesn't need to decipher the details of the story correctly to be affected by it. The logic of the imagined events gives the whole model an artistic unity and authentic presence that a random collection of brownouts and debris could not.

An example of the same principle rendered in up-to-the-minute shader 3.0 glory, is the artwork in 2K's BIOSHOCK. The drowned city of Rapture abounds in well-chosen narrative details, even aside from the important set pieces, which are important to driving the game's complex interwoven stories. Many of the stories are played out in very literal ways for macabre effect, bodies hung from meathooks, stores looted, and so on. But what really helps sell Rapture's unique feel are the hundreds of tiny stories scattered throughout the city. Even in out of the way corners, you'll find small stories that illustrate how or why the utopia failed: a barricaded spare room with a filthy mattress, a few bottles of booze, a couple of books; a policeman's office, buried under piles of paperwork from toppled book shelves; the wreckage of a garden party in Arcadia, complete with empty champagne bottles and an overturned tea table.

Perhaps the way the city itself is being destroyed by internecine warfare and the encroaching sea is the inspiration for the use of narrative details. Whatever the reason, the overall effect is compellingly immersive in ways few games can match.

VARIETY

There's an obvious cost to thinking about every modeling task or environment as a story. Inevitably, relying on shorthand symbolism is easier and quicker than imagining an entire history behind every

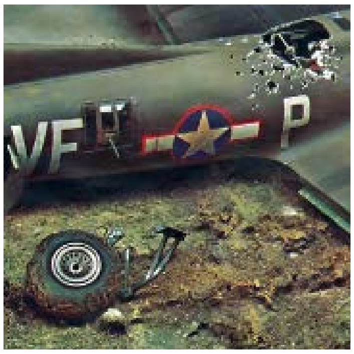


FIGURE 3 **The key to diorama is the artist's thorough conception of the events leading up to the crash. Here the bent and twisted landing gear, the torn control surfaces and the scarred landing field all help convey a complete narrative in a single moment.**

poly and pixel. The time differential may be a lot less than it seems. The actual amount of detail necessary to sell a story can be quite minimal. The mental effort, on the other hand, is considerable.

Artists who like to wade right into a project and start laying down polys will find it uncomfortable to put the building on hold while they ponder a bit of backstory. If the task seems daunting, though, it's worth remembering that the modeler or level designer doesn't suddenly have to become a screenwriter to make use of narrative detail. The stories don't need to have a lot of depth or character development; they simply need to respect and reward the player's latent powers of observation.

There is one practical drawback to focusing on narrative details. The essential point of a storytelling approach to modeling is to emphasize the individual history—the "personality"—of the subject. In games, unfortunately, we have to manage scarce runtime resources, and many of the assets we create have to be reused. The convincing detail that turns a model into a uniquely believable object can backfire when it shows up again and again. Asset modelers will have to plan carefully how to avoid undermining their own efforts through repetition. Even environment artists will find that key details will repeat, whether common elements like doors and fixtures or, more often, textures. Balancing the need for reusability with the power of hand-crafted individualizing detail is a tough trick.

RETREAD LIGHTLY

One common strategy is to build both individualized and generic versions of the same asset. For example, if you have a lot of stop signs in your city, you'll probably need to reuse them often. But if the mix is leavened with a couple of variants, such as a bent stem from being hit by a car or a vandalized version with a "'Stop' Eating Animals" sticker, the monotony is relieved and even the generic variants gain a touch of extra depth.

Ideally, the variants can share geometry or texture work with the generic versions so that the resource costs of the whole package aren't overwhelming. Designing assets from the outset so they support cheap color variations, part swaps, and decaling makes life much easier as you balance unique details with unobtrusive generics.

"Respect and reward the player's latent powers of observation."

Rigging assets for animation and then "re-posing" them to build variants cheaply is another good investment. Naturally, of course, the amount of energy you'll put into individualizing assets will scale with their relative importance. You probably want the details of the heroine's car to tell the player something about her and her history, but you probably don't want to spend too much time on the life and times of her toaster oven.

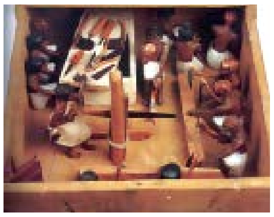


FIGURE 4 **This 4,000 year old Egyptian diorama displays many of the artistic techniques a modern environment artist or modeler would recognize.**

Although game technology seems to be at the height of information age modernity, the basic challenges of the working artist never really change. Learning some tactics from real-world modelers isn't an unreasonable stretch for the modern pixel pusher. We're the latest generation in a line of modelers that goes back at least to the days of the Egyptian Pharaohs, who passed into the next world accompanied by detailed hand-carved dioramas of daily life in this one. The 4,000-year old diorama in Figure 4, despite its simple execution, still conveys startling immediacy. You can almost hear the commotion and smell the sawdust in the crowded carpenter's workshop. Though we work in ways that anonymous 11th dynasty craftsman never dreamed of, we're still hoping to achieve the same things. Let's hope we do an equally good job.

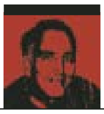
more on modeling

SHEPHERD PAINE'S BOOKS ***Modeling Tanks and Military*** *Vehicles* (Kalmbach Publishing, 1982) and *How To Build Dioramas* (Kalmbach, 1999) are great introductions to Paine's narrative approach to detailing. His web site, www.shepherdpaine.com, contains a gallery and reproductions of his famous Monogram series dioramas, which show the narrative principle applied to a number of historical military vehicles.

Bill Horan's *Military Modeling Masterclass* (Osprey, 1994) focuses on the details of painting technique rather than modeling per se, but it's still a good showcase for the way in which detail choices can take stock subjects and give them personality. Many of the best pieces can be seen online at www.kitpic.com/pf.php?fid=479.

Lorne Peterson's *Sculpting a Galaxy* (Insight Editions, 2006) is a lushly illustrated tour of the physical models from the early years of the *Star Wars* franchise, the most famous application of narrative detail in modeling.

The Association of Professional Model Makers is the professional group for modelers in the engineering and entertainment. Its web site, www.modelmakers.org, has a good bibliography of modeling-related books.



NOAH FALSTEIN

GAME SHUI

FISHY RULES

Water? What water?

I'VE RANTED BEFORE ABOUT HOW PEOPLE have told me I'm missing the most important rule of game design: "Make it fun!" I usually try to politely explain that it's such a basic rule that if it's news to someone, they should reconsider becoming a designer.

But is that necessarily true? I've written before about how I'm not crazy about the term "serious games" for games that have a purpose beyond entertainment because most—but not all of them are supposed to be entertaining and fun too. It's possible to conceive of a game that very effectively categorizes photographs or awakens people to take action against genocide that might not be fun per se, but it might be quite effective in its main purpose.

Will Wright's "software toys" such as SIMCITY 2000 may lack explicit goals but they often have strong implicit goals.

Perhaps stating the blindingly obvious is useful in getting us to challenge our assumptions, which is a useful rule. Like the old cliché about fish not being aware of the water they're swimming in, examining our most basic assumptions may reveal some important facts to which we've become blind.

THE FISHWATER RULES

What are these rules of game design that are so basic every game—or at least 99 percent of them—have them as part of their basic DNA? Are they rules that help us define what a game is?

I'm not sure I trust myself to even see them, as by definition they should be nearly invisible to me, so I'm working on instinct and feeling and am ready to be corrected by my readers. That said, here are a few that come to mind.

START WITH WHAT WORKED BEFORE

Starting with what has been proven successful in the past is not so much a rule as a truism. Virtually all current games are heavily inspired by previous ones. I've been a game developer long enough to remember the 1970s, when we had no idea what was even possible for video games, and tried all sorts of wild experiments.

But now, for people who have been exposed to games for decades, I doubt it's even possible to truly start fresh. Certainly when people begin with the pitch, "I don't play games, but I've come up with an idea so new and exciting that even I'd play it, so I know it's good!" then often the idea is one that has been done dozens of times before.

Some of the current masters of experimental gameplay manage to consistently break rules and come up with interesting things, but even their games often share a basic structure with preexisting titles.

INCLUDE INTERACTIVITY

Our medium is about setting up lots of interesting choices for the player. Is it possible to make a title without choices that can even be called a game? I can conceive of games that minimize choice, but if you eliminate it completely it seems that it's not a game anymore.

HAVE GOALS

I've written previously about more sophisticated rules for giving the player short, medium, and long-term goals, or how to use visual or implied goals. Will Wright is fond of so-called software toys that lack explicit goals, but even those have pretty strong implicit goals. It's hard to get people to define a game, but something fun without goals tends to be called a toy or perhaps a hobby or pastime.

NEAR-UNIVERSALS

I considered other possible rules that are common to many games, but with some significant exceptions. For example, the vast majority of games involve some sort of gradual increase in difficulty, or at least in scale or complexity. But somehow that doesn't seem inherent to actually being a game. I can even think of some fun games that maintain a pretty consistent difficulty like THE SECRET OF MONKEY ISLAND series, which entertained more through humor than through challenge.

Similarly, almost all games have a theme and often a story or narrative, but there are some very abstract puzzle games that are so lacking in this respect I hesitate to suggest it.

WHAT'S THE POINT?

Why is it worth questioning our basic assumptions about what must go into a game? I believe that if we are to continue advancing the state of the art, it's very helpful to know what assumptions we make in case we've been treading down a road for years without even noticing it's not the only option. Just because everyone does it doesn't make it right. Or to quote another saying I believe in, "Everybody knows that" is not a valid proof.

NOAH FALSTEIN *has been a professional game developer since 1980. His web site, www.theinspiracy.com, has a description of The 400 Project, the basis for these columns. Also at that site is a list of the game design rules collected so far and tips on how to use them. Email him at* ***nfalstein@gdmag.com****.*

February 18-22, 2008
San Francisco
Visit gdconf.com for more information & event updates

www.gdconf.com

New for 08! Purchase a Conference Pass and
Gain Access to

myGDC

The Ultimate
Game Developer's Networking Resource

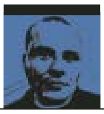
myGDC enables you to:

- Create and customize your profile on gdconf.com
- Search a database of thousands of game developers attending GDC
- Message and network with other members of the myGDC community
- Share your profile, projects, and ideas with your peers

Connect with the most influential and
internationally diverse audience of game industry professionals
before, during, and long after the conference!

CMP
United Business Media

Learn more about myGDC at www.gdconf.com



JESSE HARLIN

AURAL FIXATION

SPEAKING IN TONGUES

Localizing at home and abroad

THE GLOBAL ECONOMY NECESSITATES global communication, and for game development that means ensuring your final product is translated into a host of local languages. To help combat piracy and extend customer appeal to non-English speaking countries, international versions of games must be produced alongside domestic products to ensure a simultaneous worldwide release. This process, known as localization, can be complex with endless opportunities to introduce language-related bugs into the game.

Luckily, well-established practices are slowly developing to help teams steer clear of the potential pitfalls of localization. One aspect in particular that is seeing more and more best practices is the recording of international voice dialogue.

M'AIDEZ!

The most common process of localization involves translating game text and recorded dialogue from an original English source. It's common for tens—if not hundreds—of thousands of dollars to be spent on scriptwriters, voice talent, and audio professionals, all to ensure the highest quality source material during domestic development. However, outside of a few ancillary tasks, this critical domestic audio team rarely has much involvement in the creation of the international voice assets—and with good reason, as few of these teams include native speakers of a worldwide collection of languages.

The standard range of localization includes French, Italian, German, and Spanish (often abbreviated as FIGS) as well as Japanese. As more markets open up across the globe, though, additional languages are making their way into games, such as Mandarin, Arabic, Korean, and even Polish.

"Actors in Europe are often affiliated with recording studios rather than talent agencies."

Localizing a product begins with casting the domestic title. Once cast, a casting package is assembled by producers or a head writer who might have some input as well. The package includes concept art and descriptions of key characters; for example, "male, 17 years old, computer nerd with high voice and a slight lisp."

Ideally, the package should also include some audio reference of the domestic actors. Interestingly, actors in Europe are often affiliated with recording studios rather than talent agencies, so selecting a studio goes a long way toward casting the FIGS voice set.

The casting package is typically sent to the publisher's international partners, whether Sony Computer Entertainment Europe, Sega Europe, or domestic companies such as Babel Media or Medialocate, which specifically handle international localization. It's these international partners who then handle outsourcing the FIGS voice set to established European localization companies.

Luckily for domestic audio teams tasked with localization, the same companies used by the likes of Sony and EA are available for hire to smaller teams as well. Companies such as SideUK, France's La Marque Rose, or Germany's Effective Media can handle all aspects of localization from casting to recording to delivery, depending on the project's specific needs. These companies are also tasked with translating the full domestic script into the various target languages, a job undertaken by native speakers of the languages hired by the localization company.

MULTI-LINGUAL ACTORS

Once the casting and translation is completed, the various versions of the game's script go into the recording studio.

There are three main ways to record international dialogue:

1. Have the actors perform it as written without any other restraints.
2. Ask the actors to constrain themselves to an approximate time with an acceptable variance of about 10 percent of the original domestic line's timing.
3. Have the actors perform so as to be lip-synced for cinematics or important in-game animations.

A large percent of the actors employed by these European localization companies are Automatic Dialogue Replacement (ADR) actors from European film and TV, so they're well versed in the process of time syncing voice. If they feel too restricted by lip-sync or timing issues, the actors may improvise with more colloquial or truncated versions of the lines so as to make the dialogue fit the scene.

After the raw international assets have been recorded and edited, if the domestic audio team is going to be responsible for

JESSE HARLIN has been composing music for games since 1999. He is currently the staff composer for LucasArts. You can email him at jharlin@gdmag.com.

anything more before final delivery, it's now that they may find themselves involved in specific or complex creative processing or international cut scene mixing. Otherwise, the international voice files are delivered and implemented into the game.

Once everything has been implemented, the international voice set undergoes linguistic quality assurance. This testing is handled by either the publisher's international partners, the company contracted to create the FIGS assets, or both, and consists of native speakers playing through the game to make sure the dialogue makes sense or hasn't accidentally had the wrong language implemented into the wrong SKU.

ORIGINS IN THE EAST

When English isn't the initial source language, the process is relatively similar. For games from Japan, translation work is done by native speakers of English after which voice direction is frequently outsourced to Hollywood film and animation voice directors, such as Jack Fletcher (FINAL FANTASY X) and Ginny McSwain (RESIDENT EVIL 4).

Rather than utilizing studios with a stable of ready talent as in Europe, these voice directors most frequently utilize the available talent pool of Los Angeles voice-over actors. These actors perform in the same manner as European actors, recording either with timing or lip-sync concerns, or freely without any restraints to their performances. However, it's much more common in Japanese game development than in English-language development to have the game's director or producer on hand at the localization sessions actively partaking in the direction of the voice talent.

In the end, the process of localizing game dialogue for a global market is a global endeavor, as it should be. Efficient international systems exist to ensure not only the widest reach of your game's script, but also your game's unique voice. :::

RESOURCES

Babel Media, U.K.
www.babelmedia.com

Medialocate, U.S.
www.medialocate-usa.com

SideUK, U.K.
www.sideuk.com

La Marque Rose, France
www.lamarquerose.com

Effective Media, Germany
www.effective-media.de

Daily Game Event Coverage

GAMASUTRA

The Art & Business of Making Games

Winner of 2006 and 2007 Webby Awards Winner

Your comprehensive Source for Game Industry

INFORMATION

{ Timely Business & Management Updates • Breaking News From Seasoned Editors
In-depth Features by Industry Leaders • Crucial Career & Education Resources }

CMP

INDEPENDENT GAMES FESTIVAL

THE 10TH ANNUAL

INDEPENDENT GAMES FESTIVAL

FEBRUARY 20-22, 2008 • SAN FRANCISCO, CA

ENTRIES ANNOUNCED

WWW.IGF.COM

WATCH FOR FINALIST ANNOUNCEMENTS IN DECEMBER

PLAY THE BEST INDIE GAMES OF THE YEAR

THE IGF PAVILION AT GDC 2008
SAN FRANCISCO, CA

PAVILION: FEBRUARY 20-22, 2008
GDC08: FEBRUARY 18-22, 2008

WWW.GDCONF.COM

CMP
United Business Media

ACTIVISION®

Great games start with great people.

We have immediate openings for:

Animators
Art Directors
Character Modelers
Designers
Environment Artists
Graphic Programmers
Lighters
Network Programmers
Producers
Scripters
Visual Effects Artists

ACTIVISION. VALUE PUBLISHING, INC. NEVERSOFT infinity ward Vicarious Visions® TREYARCH

RAVEN Luxoflux BEENOX ShABA redoctane® ACTIVISION. FOSTER CITY Toys For BOB

WWW.ACTIVISION.COM
GO TO "CAREERS"

Work for a studio with great character
AMAZE
BACKBONE
the collective
DIGITAL ECLIPSE
imaginengine
Shiny
Foundation 9 Entertainment
Emeryville, CA Boston, MA Newport Beach, CA Kirkland, WA New York, NY
Vancouver, BC Austin, TX Charlottetown, PEI Eugene, OR Los Angeles, CA
Go to www.f9e.com for current job listings

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

1. Publication Title: Game Developer. 2. Publication No.: 1073-922X. 3. Filing Date: September 17, 2007. 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): CMP Media LLC, 600 Harrison St., 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): CMP Media LLC, 600 Harrison St., San Francisco, CA 94107. 9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor: Publisher: Simon Carless, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107; Editor: Brandon Sheffield, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107; Managing Editor: Jill Duffy, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107. 10. Owner: CMP Media LLC, 600 Community Drive, Manhasset, NY 11030, an indirect, wholly owned subsidiary of United Business Media PLC, 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 Magazine. 14. Issue Date for Circulation Data Below: October-07.

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)	46,024	36,345
b. Paid and/or Requested Circulation		
(1) Individual Paid/Requested Mail Subscriptions Stated on Form 3541. (Include advertiser's proof and exchange copies)	29,727	29,728
(2)Copies Requested by Employers for Distribution to Employees by Name or Position Stated on PS Form 3541	317	321
(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid or Requested Distribution Outside USPS	4,094	3,450
(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)]:	34,138	33,499
d. Nonrequested Distribution (By Mail and Outside the Mail)		
(1) Nonrequested Copies Stated on PS Form 3541:	5,667	1061
(2):Nonrequested Copies Distributed Through the USPS by Other Classes of Mail	0	0
(3) Nonrequested Copies Distributed Outside the Mail (Pickup Stands, Trade Shows, Showrooms, and Other Sources)	5,290	981
e. Total Nonrequested Distribution	10,957	2,042
f. Total Distribution (Sum of 15c and 15e)	45,095	35,541
g. Copies Not Distributed	929	804
h. Total (Sum of 15g and 15h)	46,024	36,345
i. Percent Paid and/or Requested Circulation (15c Divided by 15f Times100)	75.70%	94.25%

16. Publication of Statement of Ownership: This Statement of Ownership will be printed in the November 2007 issue of this publication. 17. Signature and Title of Editor, Publisher, Business Manager, or Owner (signed): Simon Carless, Date: October 1, 2007.

Make More Enemies

Game Design at Vancouver Film School

The Leader.
Dave Warfield, 15-year veteran game designer for over two dozen titles.

The Know-How.
In just one year, you'll learn every aspect of game design. Your portfolio project is a playable, industry-ready video game.

The Results.
Our graduates work at top game companies including Backbone Entertainment, BioWare, EA Black Box, Next Level Games, and Propaganda Games.

vfs.com/enemies

VFS™

SCHOOL OF GAME DEVELOPMENT

COMPUTER ANIMATION > DIGITAL ARTS & DESIGN > ENTERTAINMENT BUSINESS > FILM > RECORDING ARTS > SHOW PRODUCTION & TOURING

Financial aid available to those who qualify • Career development assistance • Accredited School, ACCSCT

ONE OF THE TOP FIVE GAME-DEGREE PROGRAMS IN THE WORLD

— *Electronic Gaming Monthly*

FULL SAIL
Real World Education

800.226.7625
3300 University Boulevard
Winter Park, FL 32792

fullsail.com

© 2006 Full Sail Inc. All rights reserved.

Serious Game Design Institute

SANTA BARBARA CITY COLLEGE

Do you want to work on a Serious MMO?

NOW HIRING FOR ALL POSITIONS!

Visit: **sgdi.sbcc.edu** for current job listings

CENTER FOR DIGITAL IMAGING ARTS AT BOSTON UNIVERSITY

WALTHAM, MA | WASHINGTON, DC

CREATE your WORLD

cdia BOSTON UNIVERSITY

CERTIFICATE PROGRAMS:

3D ANIMATION | VISUAL EFFECTS | GAME DESIGN

Intensive full- and part-time programs for the skills and tools you need to turn your ideas into reality. Financial assistance and career services available. ***Apply today!***

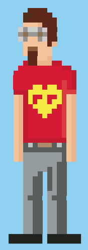
CALL **800-808-CDIA** EMAIL **INFO@CDIABU.COM** WEB **WWW.CDIABU.COM**

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
Activision	43	Perforce Software	6
Autodesk	C2	Rad Game Tools	C4
Center for Digital Imaging Arts at Boston University	47	Replay Solutions LLC.	19
Electronic Arts	27	Santa Barbara City College	46
Epic Games	11	Seapine Software Inc.	23
Full Sail Real World Education	46	The Collective Inc.	44
Havok	15	University of Advancing Technology	47
Kynogon	3	Vancouver Film School	45
Netdevil LTD	12	Vicious Cycle Software	C3

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as CMP Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **Subscription Rates:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **Postmaster:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **Customer Service:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/CMP Media, LLC, unless otherwise indicated. Don't steal any of it.

GET EDUCTATED



(pwned)

(pwner)

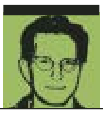
University of Advancing Technology UAT Learn. Experience. Innovate.

Please geek responsibly.

You may speak the language, but are you geeked? Here's a chance to prove it.

GAME DESIGN	COMPUTER FORENSICS
DIGITAL ANIMATION	NETWORK SECURITY
ARTIFICIAL LIFE	SOFTWARE ENGINEERING
DIGITAL VIDEO	TECHNOLOGY MANAGEMENT
WEB DESIGN	**GAME PROGRAMMING**

www.uat.edu > 800.658.5744



RUSSELL CARROLL

WHERE CREDIT IS DUE

QUICK, NAME A CASUAL GAME DEVELOPER other than PopCap! You didn't come up with anything immediately did you? Even for those who work in the casual game industry, naming more than a handful of developers can be difficult. This is true despite the fact that there are many casual game developers making a very lucrative living.

Why is it that many of the extremely successful developers in the casual space haven't gained any name awareness?

Though it is probably only a part of the explanation, an interesting phenomenon has occurred in the casual game industry. When you purchase a non-casual game in a retail store or at an online store such as Amazon, you will always see credit given to the creator. The creator's name and logo usually appear on the front, spine and back of the game box. The creator's name is credited on the product page in online stores.

However, this is not the case when you look at the majority of the casual game distribution portals. When visiting Real Arcade, MSN, iWin, Big Fish Games, Yahoo! Games and most other portals, there is no mention of either the developer or the publisher on the game product page. Clearly the lack of creator credit is not an oversight. Real Arcade, for example, formerly included credits on their pages, but has since removed them.

Why don't the portals want to give credit where credit seems due? Are portals marginalizing developers in order to try and keep another PopCap from rising up? This is a question that most portals avoid answering.

Now before going further I should mention that I work for Reflexive Entertainment, a game developer which also runs www.reflexive.com, a casual game distribution portal that does include creator credits. Take as many grains of salt as you need.

With that said, I have some thoughts on why portals are not crediting game creators. The reason is tied into the value of traffic to the portal. As is true in all businesses, casual game distribution portals need customers to survive. To accomplish this, the portals take every opportunity to keep all the eyeballs on themselves. Not crediting is only one of many things done to accomplish this goal.

While the portals don't credit the games' creators, they do require a portal credit of sorts inside of the games they distribute. You may have noticed that every casual game that you download has the logo of the distribution portal it came from inside the game. In fact, every major portal requires that their logo appear in equal size and prominence with the developer's logo, wherever the developer's logo is displayed in the game.

Imagine if this happened in other parts of the industry. Can you picture a Walmart splash-screen at the beginning of HALO 3 if you bought it from Walmart and a GameStop screen if the game were purchased at GameStop?

However, the fact that the mainstream game industry readily credits creators and not the point of purchase is not a good reason for the casual industry to change. They are different industries and you could argue that without the portals, there wouldn't be a casual games industry. The portals invested in the infrastructure and the marketing to create the customers. Don't the portals deserve to recoup on their investment in the way that makes the most business sense to them?

The answer to this question is 'yes' with one very important caveat. Misleading customers in regard to who has created a game is nigh unto plagiarism.

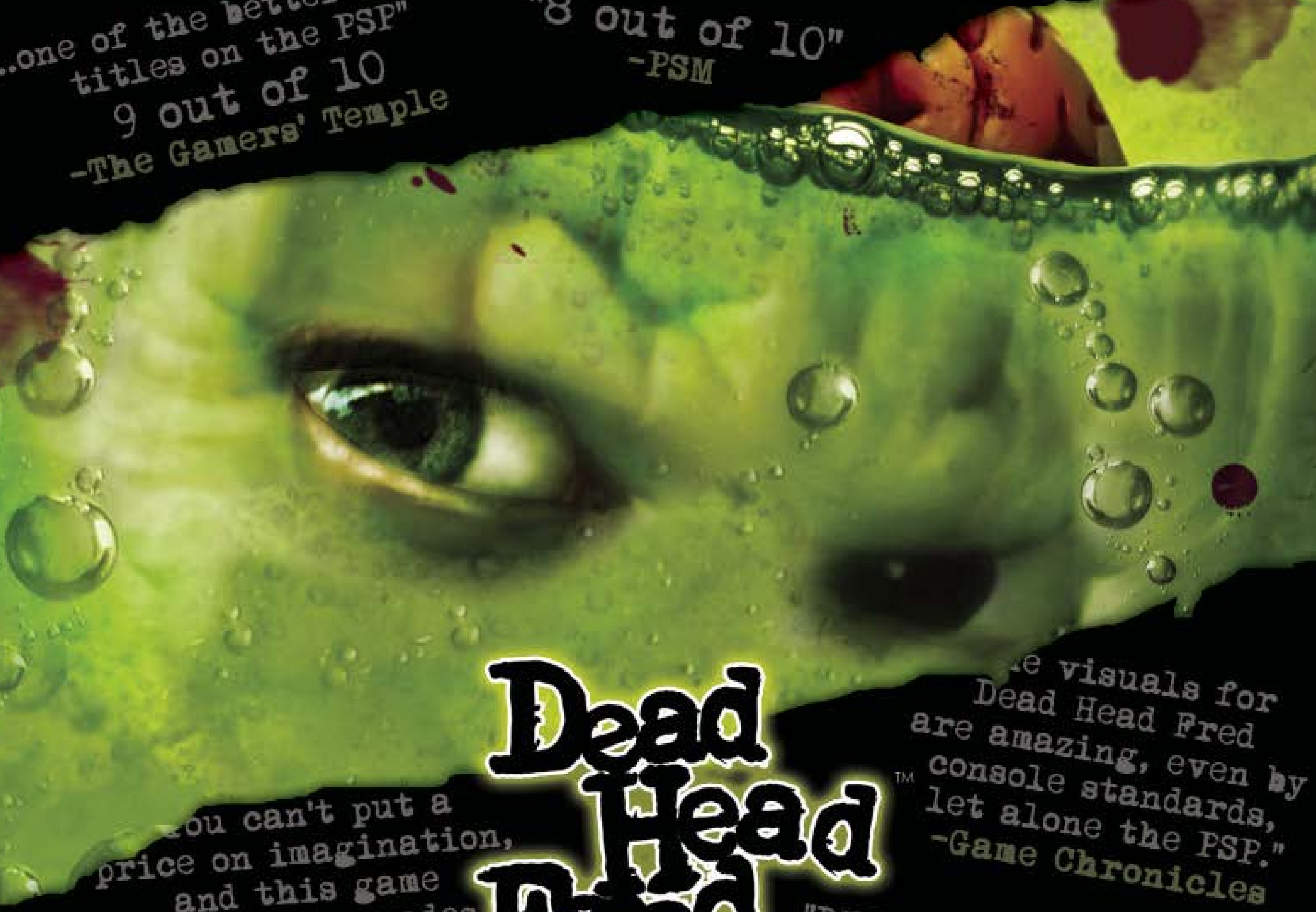
Imagine going to an art gallery where every painting had a plaque with a description next to it, but without any mention of the artist. Imagine further that each art piece had an imprint of the gallery's name on it equal in size and prominence to the artist's signature. Though this example may seem a bit extreme, a search through the forums on any portal finds that the customers believe the portal to be the author of the games they are enjoying.

In considering the problem, it seems clear that some changes are needed. The IGDA casual games steering committee is currently considering an initiative to standardize developer crediting to ensure that developers and publishers receive credit for the games that they create.

In considering different options, one suggestion may please both creators and portals: game credits that link to a page containing a selection of the developer's games within that portal. One of the most difficult things for portals to do as their catalog of games expands daily is help their customers navigate to relevant games. As different developers have different styles, using linked game credits could be used to direct customers to other games that might appeal to them. This provides credit to the game creator, and potentially creates revenue from increased cross-selling on the portal. Reflexive, and to some extent Big Fish Games, are both currently using linked credits on their pages, and I can say for the Reflexive side that the results have been positive.

The creator of a work deserves to be credited for it. While some portals are crediting developers, including PlayFirst, Shockwave, and a handful of others, most of the casual game distribution portals do not. With the IGDA casual games steering committee currently considering this issue, it's a good time for game creators to do likewise and push for the casual game industry to adopt guidelines that will help give credit where credit is due.

RUSSELL CARROLL *is the director of marketing for Reflexive Entertainment. Email him at rcarroll@gdmag.com. Email him at* ***rcarroll@gdmag.com.***



..one of the bette
titles on the PSP"
9 out of 10
-The Gamers' Temple

"8 out of 10"
-PSM

Dead Head Fred™

e visuals for
Dead Head Fred
are amazing, even by
console standards,
let alone the PSP."
-Game Chronicles

ou can't put a
price on imagination,
and this game
has that in spades.
-Cheat Code Centra

"DHF is slick on the PSP"
"8.3"
-IGN

The game looks simply
enomenal on the PSP."
★★★★
-Gamespy

"9 out of 10"
-GamerNode

"The game engine is silky smooth
You'll be hard-pressed
to find a better-looking
game on the PSP."
-Hardcore Gamer

"8.5 out of 10"
-Gamezone

Vicious Engine

making games. easier. faster. better.

Vicious Cycle®

The powerful & flexible technology behind the critically acclaimed PSP® (PlayStation® Portable) system title: "Dead Head Fred™."

www.viciousengine.com

MATURE 17+ M — Blood and Gore, Intense Violence, Mature Humor, Strong Language — ESRB

PSP PlayStation® Portable

D3PUBLISHER

RAD GAME TOOLS PRESENTS

MILES

Sound System

COMPLETELY HANDLES ALL ASPECTS OF SOUND

WWW.RADGAMETOOLS.COM/
MILES

MULTI-CHANNEL DIGITAL AUDIO, INTERACTIVE MIDI,
RED BOOK CD AUDIO, 3D AUDIO, INTERNET
VOICE CHAT, DECOMPRESSION, EAX, AND MORE!

TUNE IN AT:
425-893-4300

MP3,
OGG, ADPCM
DECOMPRESSION

RAD GAME TOOLS

ALL PLATFORMS:
PS3, XBOX 360,
Wii, PC, MAC

GRANNY 3D   BINK VIDEO   PIXOMATIC   MILES