DECEMBER 2007

gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

>> **MUD-DY WATERS**
LESSONS LEARNED
FROM TEXT-BASED MMOS

>> **NOT-SO-SILENT HILL**
AN INTERVIEW WITH
AKIRA YAMAOKA

>> **VIRTUAL CASH COW**
THE FREE TO PLAY, PAY
FOR ITEMS GOLDEN GOOSE

POSTMORTEM:
5TH CELL'S
DRAWN TO LIFE

$5.95US $6.95CAN

1 2>

0 71486 02133 9

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

Autodesk

Image courtesy of Ubisoft

UBISOFT

gamedeveloper

[: CONTENTS :]

22

FEATURES

9

16

31

POSTMORTEM

DEPARTMENTS

COLUMNS

COVER ART: PAUL ROBERTSON

[GAME PLAN]

SOUTH KOREA'S GOT SEOUL

I'M FRESH OFF THE PLANE FROM SOUTH KOREA, having just attended the GStar expo and met no less than 13 different Korean game developers, either at the show or in their offices—and I have the very distinct impression that I've just been schooled.

It's easy to feel like you know what's going on in the world. I've been following the Korean market since 1999, when I picked up my first Game Infinity brochure (representing Korean game companies as a whole) at E3. I've long had faith in the region's entertainment industry, and time has proven me right in that regard—but actually going there, meeting the developers, visiting the studios, and feeling the vibe, there turned out to be a whole lot I just didn't know.

DIFFERENT ENVIRONMENTS
The first developer I visited was Blueside—pretty much the biggest console developer in Korea, with titles like KINGDOM UNDER FIRE and NINETY NINE NIGHTS under its belt. Shockingly to me, the office is incredibly small, with over 100 people crammed into a small space that was clearly a paper-shuffler sanctuary before Blueside occupied it. Developers there commented to me how difficult it was to get competent coders and designers, since everyone wanted to go off and work on the MMOs. It's hard to imagine that happening to the largest console developer in America.

The next place I visited was Gamevil (NOM), the second or third largest mobile developer in the country. Gamevil also had just over 100 people, but the space was easily twice as large, in a nicer building in a much swankier part of town, with gorgeous company-green walls and frosted glass partitions for the big bosses' offices. The difference in vibe (and revenue) was palpable.

Next there was Gravity, an upper-middle tier player in the online space, which had early success with RAGNAROK—success that hasn't yet been repeated, as RAGNAROK lumbers on with higher rates of use than its sequel. Gravity occupies five floors of the Meritz Tower in the Gangam part of Seoul—one of the nicest areas around. 280 people are spread across those floors, with plenty of space for all. The difference in scale was just staggering, and almost seemed inverse to the situation in the West, where console still rules.

DIFFERENT REVENUE
Korean developers sometimes get much larger revenue shares than we do on this side of the pond. Gamevil gets 85 percent of the profit revenue from its games, while the carrier gets only 15 percent. For games that need servers (like mobile MMOs), it's more like 60/40.

Then there's N-LOG Soft, developer of B.O.U.T., which Acclaim brought to western markets quite recently. N-LOG is relatively small for an online developer, with (again) around 100 people, so I asked if it was difficult to keep their IP since they always partnered with publishers. They looked at me like I was crazy. "You mean developers in the West don't keep their IP? It seems like you would really want to." I thought I'd never get my jaw off the floor.

DIFFERENT SKILLSETS
Korean companies are expanding into the West, so naturally they're toying with the idea of entering the console market. Trouble is, the industry grew up so fast around online games, which are never finished, that very few companies know how to make finalized, complete products. Blueside and Softmax (MAGNA CARTA) are the only companies that have really done it in the current era, and I didn't previously realize what a large problem it was. The consensus was; "If we move into the console space, we need help to figure out how to make a complete product out of our property." Their pipelines aren't even set up for it, given the different nature of the online space.

In terms of those who have done it, Blueside had help from Microsoft. Softmax had help from Bandai. Nexon is getting help from Nintendo for its MAPLE STORY DS port. Ntreev's PANGYA for Wii was handled by Tecmo. It's basically ubiquitous. No matter how creative and lucrative these companies are, they grew in a very different direction, and while they're ahead of us in network infrastructure and online logistics, they're behind in other ways.

CULTURE SHOCK
The fact is, Western developers have ignored or discounted the South Korean market for far too long, and now we're face to face with reality. WORLD OF WARCRAFT aside, they've taken a genre we invented and perfected it to the point where an online dance game like T3 Entertainment's AUDITION has tens of millions of subscribers worldwide. That's rather a lot of people and most of us over here have never heard of it. When I first noticed that company in 2002, they were making a game called RAPHAEL that was so rudimentary I thought it was freeware. The Korean industry has come a long way. It's time to start paying attention to what we can teach each other.

Brandon Sheffield, senior editor

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

DEVELOPERS SPEAK:
SOUTH KOREA

Bong Koo Shin
Head of Shin's Game Studio, Gamevil (NOM series)
on the far future of games

I think we could invent new emotions or feelings that you get from playing games. But that new emotion can't be explained in words, because it's never been experienced before. Score will have no more meaning, it won't be meaningful after you get that kind of new feeling. It's a feeling like joy from playing the game, and the joy of experiencing that feeling—so it'll be some sort of art convergence, with games in movies, and movies in games.

If you take a brain image of game addicts and drug addicts, those images are very similar. So there's some sort of addictive point, where you get addicted to the game. So in the future, games will be cut down to the barest essentials, just some sights, sounds, and colors, and it could be more like a digital drug. It could be used in bad ways, like implanting images in peoples' brains, but if we could use it in good ways, we could do psychotherapy and things like that.

It's not about virtual reality. There's this brain device, mind-controlling device called MCSquared in Korea, and you wear a headset and it kind of controls and soothes your brainwaves with colors and sounds. That's very preliminary stuff—it's an infant level of this digital drug. Eventually we won't need game characters and themes, like fight to find the princess, or save the world, that's all not related to the core, it's more peripheral. If we can pull out just those addictive points, and put them all in one, they'll be addicted immediately. That'll be the digital drug I expect to see in the future.

Joe Cho
Marketing, N-LOG Soft (B.O.U.T.)
on the publisher/developer relationship in Korea

The infrastructure is all developed on the development side, then customers service and game master service is done by the publisher. But we have co-production of GM tools and things like that.

I think most of the publishers have an initial one-time payment, but also some revenue after that. So if a developer gets an initial fee, they get worse revenue share. Currently we're doing initial payment with some revenue share and profit incentives if we reach certain numbers of users.

(If a publisher releases the game in the U.S.) we still get paid again. In the case of B.O.U.T. they have a service license, not a full license. So if the users pay for some cash items, we have revenue share for that.

If we sell our game source and things, maybe publishers get the IP. But I don't think any developers think that way. Because it's their project, not the publisher's! So we just need the publisher's big servicing power, not for selling our intellectual property.

—*Brandon Sheffield*

Sang Woon Yoon
Global Management team/Development Strategy, WebZen (HUXLEY, APB)
on the company's increasing console focus

(It's) the company's vision ... especially the CEO's vision, he's originally an animator. So he looks at a lot of the Western games, and he really likes the high-end graphic style. He had been working on online games for PC, but still had a lot of interest in the western style, the console graphics, which are more polished and things like that. In addition, the online world is quite busy. In console games, the U.S. market is big, and generally it sells a lot, so you can calculate return on investment easier than online games. Also next-gen consoles like 360 have great graphical and CPU capabilities.

But up until now, it's not really been easy for us to get into the console market, since we have such a strong MMO PC base. But we're trying.

So far for those console technologies within the company we don't really have the original staff working on those versions. So far the strategy was to try to collect people from other companies, and sometimes we tried to buy whole development teams, but now we have a different strategy—probably in the future we'll look to an outside company to make things work. Until now we were trying to solve everything within the company.

Jamie Kang
General Manager, International Business Division, Ntreev (PANGYA)
on the U.S. online game space

In terms of the online market in the States, aside from the big guys, there aren't many companies involved in the industry. So for us it was really hard to find somebody who could run our games in a better way. We figured it'd be easier to open a subsidiary.

I think in the future more companies will try to participate in the online industry in the States, but I think it'll take some time. We can't afford to wait for them to come. And one of the reasons we entered into the States is that there's a lot of potential. I mean the original online games come from there, right? But now the Korean companies are booming, and have expanded the market. But the originals come from the States, so I think it's still promising.

What I've heard from some of our staff in the States, as well as some of our friends, is that the item-based games have a lot of potential to grow. A lot of signs indicate that the revenue model goes up. For NHN with Hangame.com, their revenue is still growing, and a lot more users are coming. I thought that was a good sign for us, or other Korean companies.

2007 FRONT LINE AWARD FINALISTS

WHETHER YOU ARE WORKING CLOSE TO THE METAL, APPLYING INVERSE kinematics to a lich king, or modeling the impact velocity of a depleted uranium shell, the tools honored by Game Developer's 2007 Front Line Awards can help you get there.

Given annually to the top game development products, the Front Line Awards pay recognition to the software tools and books that help game creators stay on the cutting edge of the entertainment industry. Front Line winners will be announced in the January 2008 issue of Game Developer and one tool with enduring utility and influence will be inducted into the Hall of Fame.

Advancing the state of the art is never easy and Game Developer congratulates all of the finalists whose hard work enables and inspires game developers around the globe.

ART

3ds Max 9 Autodesk

modo 301 Luxology

Mudbox 1.06 Autodesk/Skymatter

Softimage XSI 6.01 Softimage

Zbrush 3.1 Pixologic

ENGINES

CryEngine 2 Crytek

Gambryo 2.3 Emergent

Hero Engine Simutronics Corporation

Unreal Engine 3 Epic Games

Vision Game Engine Trinigy GmbH

PROGRAMMING/ PRODUCTION TOOL

Hansoft 5.0 Hansoft AB

NIVIDIA PerfHUD 5 NVIDIA Corporation

Perforce 2007.2 Perforce Software

ReplayDIRECTOR Replay Solutions

XNA Game Studio Express Microsoft

MIDDLEWARE

Euphoria Natural Motion

Gameface Studio Anark

Havok Complete Havok

Kynapse Kynogon

PathEngine SDK PathEngine

BOOKS

Game Design: From Blue Sky to Green Light, **Deborah Todd** A K Peters

Game Writing Handbook, **Rafael Chandler** Charles River Media

GPU Gems 3, **Hubert Nguyen (ed.)** Addison-Wesley Professional

Second Person, **Pat Harrigan and Noah Wardrip-Fruin (ed.)** MIT Press

XNA Unleashed, **Chad Carter** Sams

AUDIO

Fmod Firelight Technologies Pty, Ltd.

Miles Sound System RAD Game Tools, Inc.

Vivox Precision Audio Vivox, inc.

Voice-O-Matic Di-O-Matic, Inc.

Wwise 2007.2 Audiokinetic

CALENDAR

2008 GAME DEVELOPERS CONFERENCE
SPEAKERS ANNOUNCED

THE ORGANIZERS OF THE 2008 GAME DEVELOPERS Conference have revealed the initial line up of speakers to be featured at the upcoming event.

Run by the CMP Game Group (as is Game Developer), GDC 2008 will take place at San Francisco's Moscone Convention Center February 18-22, 2008. With more than 100 lectures planned for the conference some of the highlights include:

Chris Foster, senior designer at Harmonix Music Systems will present "Your Music Is the Game: Designing the OTHER Project at Harmonix" in which he discusses adapting Harmonix's "beatmatch" formula to a new IP.

In "Storytelling In BIOSHOCK: Empowering Players To Care About Your Stupid Story," Kenneth Levine, President and Creative Director at 2KBoston/ Irrational insists that "Game stories can matter, even in first person shooters. But first, we're going to have to give up a lot of our preconceptions about what people care about when playing a game."

"E Pluribus Unum: Matchmaking In HALO 3" by Chris Butcher, the Engineering Lead at Microsoft/Bungie will describe the algorithms behind HALO 3's peer-to-peer multiplayer matchmaking model and the impact of matchmaking on the HALO online community.

Garry Taylor, audio/video manager at Sony Computer Entertainment Europe will describe the challenge of managing audio assets produced by multiple companies across the globe for a next-gen project in "At The Cutting Edge - Audio Production For HEAVENLY SWORD."

Naughty Dog's AI and animation programmer Christian Gyrling will present "Creating A Character In DRAKE'S FORTUNE" in which he will examine the variety of novel animation and AI techniques used to create the game's expressive characters.

In "CRYSIS' Next-Gen Effects" Tiago Sousa, effects and graphics programmer at Crytek, will talk about CryEngine2's technology and the creation of cinematic-quality effects and shading for CRYSIS.

Tom Gioconda, Bungie's web development engineer, will present "Make The Community Part Of HALO" in which he examines the history and development of Bungie.net from MYTH to HALO 3.

"New AI Techniques For SIMS 3" by Electronic Arts' Richard Evans will demonstrate some of the unusual AI techniques used in SIMS 3 and how they serve high-level design goals.

Derek French, associate technical producer at BioWare will discuss the evolution of the BioWare Live Team over the years and its role in future game development in his presentation

"The BioWare Live Team: Building Community Through Technology."

Other announced talks include "Do, Don't Show - Narrative Design In FAR CRY 2", "Creating Scalable And Dynamic Graphics For WORLD IN CONFLICT", "Lighting And Material Of HALO 3", "Pollinating The Universe: User-generated Content In SPORE", and the return of talks including the Game Design Challenge, the Experimental Gameplay Workshop, as well as tutorials and in-depth workshops.

In addition to the line-up of speakers making presentations at the main conference, the Independent Games Summit will host a series of talks centered on indie game development.

In "Evolving AQUARIA" Alec Holowka and Derek Yu of Bit Blot will trace the development their IGF grand prize winning game AQUARIA.

For "A Tale of Two Kyles" Kyle Gabler of 2D Boy and Kyle Gray of Electronic Arts compare their experiences developing games with and without the security of a large publisher.

Dylan Cuthbert of Q Games takes a look at self-funding and creating a PlayStation 3 title in his "Postmortem: PIXELJUNK Series."

Worlds In Motion will also host a two-day, pre-GDC summit, which will feature a number of speakers focused on the intersection of online worlds and games.

Areae's Raph Koster will discuss the ways virtual worlds are increasingly relevant to the ways we play and the evolution of the medium, as the worlds of online spaces, social networking and gaming converge.

Chris Romero of Worldwide Biggies, who did the original prototyping and led the full team for the design and development of Nickelodeon's NICKTROPOLIS, discusses the building of a successful online play space that has garnered users by the millions.

Conduit Labs' Nabeel Hyatt will present a lecture on social gaming with a discussion of games for Facebook.

Senior producer and head of Relic Labs, Adrian Crook will discuss the free-to-play business model and how it's helped evolve the face of online play.

More information on the conference is available on the official web site: www.gdconf.com ✷

—Jeffrey Fleming
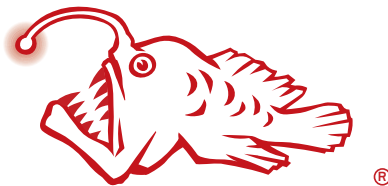
NETDEVIL

Now Hiring

for

LEGO Universe

Tech
Lead Programmer
Senior Programmers
Graphics Programmers
Programmers
Database Administrator

Art
Motion Graphics/Flash Animators
LEGO Digital Designer Artists
Special F/X Artists
3D Artists

Audio
Audio Director
Sound Designer

Design
Lead Scenario Designer

Production
Associate Producer

Check the latest postings and submit your resume online:
www.netdevil.com/employment/

They're back to save the world.

Want In?

>> taiyoung ryu and kyuhwan oh

PURCHASING POWER

Item-based payment models in Korean online games

KOREAN GAMES OCCUPIED ABOUT 45 PERCENT OF THE ENTIRE Asia-Pacific online game market in 2005 (excluding Japan), according to U.S. market intelligence provider IDC. Hundreds of new online games are launched every year, but only a few dozen survive due to the limited scale of the Korean market. In order to survive in this fierce environment, new online games are experimenting with a variety of payment methods in order to be priced competitively.

The majority of MMORPGs in the past used monthly subscription fees as their primary form of revenue. As the market became overcrowded, they quickly began transforming their payment models, either by discounting the service fee or providing the service for free but charging for items in the game. A free model occurs in the U.S. in some Sony Online Entertainment games (and a handful of others) but without as much emphasis on microtransactions.

EXISTING PAYMENT MODEL FOR ONLINE GAMES

In the West, online game companies generally adopt a subscription model, providing the game for free during an open beta and occasionally offering a free trial period. In Japan, subscription-based online services are provided to both PC and console platforms, as seen in FINAL FANTASY XI and PHANTASY STAR UNIVERSE. But there are limitations to this method.

Subscription-based models can burden both the game developers and the players. Game consumers who pay monthly dues for games might be less likely to try new games, since their payment commitments have already been established. This in turn creates a barrier of entry for new game companies trying to break into the market. As a result, many companies have had to cut their prices—to the point where even WORLD OF WARCRAFT lowered its service price in Korea in April 2006.

Additionally, as free casual online games have increased their market share in Korea, subscription-based models have lost their competitive pricing edge. Casual games typically offer players a choice of a few different payment methods, such as necessitating the purchase of certain in-game items to advance in the game, or by opting into a game that contains ads, or leading the player to subscribe to an affiliated company by paying service fees, or having the players purchase game money as a means to a settlement for buying items on specific online game web sites. The latter is the most pervasive.

ITEM-BASED PAYMENT MODELS

The most distinguishing feature of the item-based payment model is that it's up to the consumer how they choose to pay their fees. The two main ways are by paying for in-game items or various other game elements, and by purchasing in-game points.

In-game items and applications. In-game items make up the biggest part of the item-based payment model. Game players pay per item according to its price and use the items in the game. These purchases might include accessories, clothing, weapons, or items that allow the player to increase his or her

TAIYOUNG RYU is currently pursuing a graduate degree in interactive media at USC. Before coming to the US, he worked as a designer on a number of MMO and mobile games in Korea, such as TAKE DOWN ONLINE for Hanbit Soft, and LORD OF THE RINGS MOBILE for SK Telecom. **KYUWHAN OH** is an assistant professor in the Media division of Ajou University in Seoul, and is a technical advisor for Nexon Inc. His credits with Nexon include KINGDOM OF THE WINDS, and LEGENDS OF DARKNESS. Email Taiyoung Ryu at **tryu@gdmag.com**.

PURCHASING POWER

experience level. Some items can only be used once while others expire after a set amount of time, and still others become permanent virtual property of the player.

MAPLE STORY developed by Nexon is an example of a successful item-based payment model. The items for purchase usually have a decorative function but have little affect on the practical game function itself, such as a pet, which does not have any gameplay role (for example, pets cannot participate in combat with their owners).

In some games, players can buy applications that have multiple functions but do not affect the balance of the game. Most paid applications have an expiration time that's proportional to the cost.

SPECIAL FORCE (released in the U.S. as SOLDIER FRONT), a first-person shooter, was one of the most successful games in Korea in 2005. At the time the game saw on average more than 100,000 simultaneous connections, and once reached 1 million players in a single day. SPECIAL FORCE has adopted an item-based payment model—the tool used to save game replays is available for purchase, alongside various other in-game items. The replay item is what we would refer to as a paid application.

Prepaid game point system. Most of the assets in online games are usually sold for very small sums, ranging from a few hundred Korean won (KRW) to a couple thousand (tens of cents to low dollar values), and games with item-based payment models have to process every single one of those transactions. To more efficiently settle all those payments, some games require players to instead purchase game points or in-game currency in one single transaction; then the currency can be

The game design of MAPLE STORY considers whether or how each purchased in-game item will affect the character and the gameplay.

Because Nexon's games appeal to young players, there are ways to purchase in-game money other than just via credit card.

KART RIDER has two in-game currency systems; one that can be purchased and one that is earned through play.

spent in smaller amounts in the game without the company having to process each transaction.

Another benefit of using the game point system is that game companies with multiple online games can use the same in-game currency across their products. For example, most games from Nexon accept "Nexon Cash", encouraging players to try the company's other games.

Game points allow online game players to have diverse payment routes. In general, casual online games that levy a partial charge have a different target group than MMORPGs that charge monthly subscription fees. MMORPGs generally target relatively older men whereas casual online games target the 20-and-under crowd. Companies that take into consideration all these factors can often avoid problems such as young players not having regular access to a credit card. For example, players can buy game points with cash at a game shop. Game points may even be given to the players as a reward scheme for attending various events online or as a bonus when the player purchases a related product.

CASE STUDIES

Crazy Racing: Kart Rider. CRAZY RACING: KART RIDER has a very successful item-based payment model. It began selling items in

August 2004 after hosting an open beta test in June of the same year.

The game blossomed after beta: It had a record-breaking 10,000,000 unique users in Korea within eight months of launch. In addition to the service provided in Korea, Nexon began its beta test in China in the first quarter of 2006, which had a record 500,000 players simultaneously online. Since January 2005, Nexon has earned approximately 3 billion KRW (around U.S. $3.2 million) per month in Korea through in-game item sales in KART RIDER.

KART RIDER's target audience is young online players. The game is visually cute and its gameplay is straightforward and simple. Due to the young audience, implementing a payment system that required credit cards would not have made much sense. Instead, KART RIDER uses Nexon Cash, which is purchased in advance either with a credit card or at a convenience store or internet café, which are easily accessible to young people in Korea. And since Nexon cash can be bought in small or large amounts, it's readily within reach for most players.

But KART RIDER has another method of payment for in-game purchases as well, called "Lucci." Lucci is virtual money earned in-game, and what one can buy with Lucci is completely separate from what one can buy with Nexon Cash. KART RIDER has found a good balance between these two monetary systems.

KART RIDER is different from other games that let players buy the same items with either real money or in-game currency. Some games give players the option of purchasing in-game items at a discounted price when they pay in cash, creating an incentive for players to spend real money; the short-term effect is that the user doesn't have to play for hours upon hours to accumulate game points, but the long-term effect could be a drop in player participation. KART RIDER's item-based payment model has effectively resolved that problem.

Furthermore, the game design is such that the hundreds of items available for purchase have a synergy effect on each other, resulting in an increase in item sales and keeping players active at the same time. Items purchased with pre-paid Nexon Cash (or

Cash items) mainly function either as ornaments in the game or enhance the player's capabilities. Items purchased with Lucci on the other hand, generally enhance certain capabilities directly related to game balance but also provide visual flair with special characteristics or special effects which do not have an effect on the game itself, and allow players to boast to other players. Players are inclined to show themselves off to other players online, therefore, players pursue Lucci items.

Most Lucci items have high prices that are relative to the attractiveness of the item, and there are other restrictions, such as the player's rank, that make it difficult for players to purchase highly-valued Lucci items. So players purchase Cash items to enhance their character's capabilities, to obtain more Lucci, to upgrade their rank, and to purchase desirable items. Futhermore,

PURCHASING POWER

A selection KART RIDER items.

MABINOGI sells some of its in-game items in bundles.

players lose their competitiveness advantage without these good weapons), so players have good reason to purchase Cash items.

GAME DESIGN FOR ITEM-BASED MODELS

What do game designers need to consider when building a game around an item-based payment scheme?

Balancing pre-paid items and game money items. Of course, it's in the developer's interest to keep the player playing over the long term, rather than take an immediate yet negligible profit in the short term as happens when cash and in-game money items are the same.

Luckily, game designers can balance pre-paid items and game money items at the design stage. Differentiating the two types of items is the first step. It's also essential to think about what kind of synergy can be created between the two types of items so that they mutually enhance one another. Pre-paid items and game money items should be designed to facilitate sales of pre-paid items, and with an objective to increase gameplay.

Methods of selling items. Introducing a wide variety of in-game items that can be purchased with real money will help increase an online game's profits. Consumers appreciate having choice, so it helps to create items that will appeal to different people with different tastes. It also helps to create expiration dates for different items based on their function. For example, an item that is purely ornamental should have no expiration. Items purchased with real money and bearing durability should be designed to have some set period of use, such as one day, three days, one week, and so on, depending on the item's price. And some items purchased with real money should be designed with a limit to the number of times a player can buy them. The longer the duration selected, the more the average cost should decrease per day. The same principle can be applied to items with a limited number of uses.

Items that serve the player's experience level should be available for purchase only with game money earned through play—not with currency purchased with real money.

Selling items in bundles is another sales tactic. Items with similar functions or having mutual supporting functions might be bundled. It's also useful to coordinate items with high sales volume alongside items with low sales volume.

MABINOGI by Nexon sells items having similar functions in a bundle and with expiration dates under subscription-based payment. Similar to buying in bulk, bundled items cost less than the sum cost of purchasing all the items separately. Some of the

all Lucci items are permanent whereas Cash items have expiration dates, hence players are inclined to consume more Nexon Cash to obtain more Lucci.

SPECIAL FORCE. SPECIAL FORCE by NeoWiz is a first-person shooter and is the first online game to have successfully introduced the item-based payment model. The game has monthly sales of 5 billion KRW ($5.4 million), and has made around 20 billion KRW (around $21.7 million) from its initial service through January 2006.

There are two types of items that can be purchased in SPECIAL FORCE: "Cash items," which can be purchased with pre-paid game points, and "SP items," which can be purchased with the game money earned as a result of doing well in the game. Cash items are mainly used for the purpose of allowing players to obtain SP more conveniently. But unlike what happens in KART RIDER, Cash items in SPECIAL FORCE increase one's capability to obtain SP and experience levels, and never have a purely decorative function.

Purchasing Cash items directly endows players with SP or experience levels. It would take a considerable amount of time to accumulate enough SP to buy good SP items and weapons through normal game play (and

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 Front Line 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 Unreal Tournament 3 for publisher Midway and recently shipped the 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.

POWERED BY
UNREAL
TECHNOLOGY

HDFILMS TAPS UNREAL ENGINE 3 FOR DEVELOPMENT OF 3D ANIMATED SERIES

HDFilms, Inc. has selected Unreal Engine 3 to develop "Chadam," its new short-form 3D animated Web series to be distributed by Studio 2.0, the digital production and distribution arm of the Warner Bros. Television Group.

"Chadam," the brainchild of acclaimed artist Alex Pardee and character icon for the popular rock band The Used, uses his power of imagination to physically change his environment and, therefore, his world . . . to save it from evil. The series will find Chadam in a place of refuge, where he and others have retreated to plan their attack against Viceroy, a serial killer who threatens their way of life.

Using Unreal Engine 3 to render original character models and landscapes designed by Alex Pardee, this action-horror-adventure story filled with self-discovery serves as an incubator for future expansion of the "Chadam" universe across all entertainment mediums.

Jason Hall and HDFilms will use Unreal Engine 3 to unleash the hyper-stylized, exaggerated world of "Chadam" in a new animated series.

"As film, television and video games continue to evolve, the convergence of their associated tools, techniques and audiences are providing a fantastic opportunity for creative, cost-effective and compelling storytelling such as 'Chadam,'" said HDFilms President and Producer Jason Hall, who is also the founder and former CEO of Monolith. "Utilizing the Unreal Engine 3 as part of our overall production process has enabled us to realize our full vision for the 3D series."

"Jace Hall's access to leading-edge technology like the 3D animation engine of Epic is one of the reasons we are excited to be in business with his company HD Films," said Craig Hunegs, executive vice president, business management, Warner Bros. Television Group. "We look forward to working with Epic in what we hope is a long-term relationship. The "Chadam" project is a great example of Studio 2.0's commitment to provide the most compelling and exciting user experience in online and wireless video."

"Being able to work with such an advanced technology to finally bring my abstract artwork into the 3D realm is like a chance for me to play Dr. Frankenstein, except our monster is Chadam, whose skin is my art, and whose heart is the Unreal Engine 3," said Alex Pardee, creator of "Chadam."

DESTINEER LICENSES UE3 FOR NEW GAME

Destineer Publishing Corp. recently announced that it has licensed Unreal Engine 3 for a yet to be announced project. Destineer is the studio formed by former Bungie Software executive vice president, Peter Tamte.

Epic encourages companies to be imaginative and think outside the box with Unreal Engine 3.

"The Unreal Engine 3 is an impressively versatile piece of technology, and we are very excited to utilize it for our unannounced project," said Peter Tamte, president of Destineer. "We are using the Unreal Engine to make a game for a genre in which you would not expect to see it. The engine is performing marvelously and Epic's support has been top-notch."

SOUTHPEAK OPENS DOORS FOR INDEPENDENT DEVELOPERS THROUGH UE3 STUDIO LICENSE

SouthPeak Games has purchased an Unreal Engine 3 studio license for the development of its upcoming PC and console titles worldwide. The agreement is in effect throughout the life of UE3 technology across all major platforms including downloadable content. SouthPeak's latest release planned under the new studio license is "Monster Madness EX: Battle for Suburbia," currently under development by Psyonix Studios for PlayStation 3.

"Independent developers need more opportunities to bring their games to market with next-generation technology," said Katie Morgan, executive vice president at SouthPeak Games. "UE3 provides developers all the tools and resources necessary to build great games and SouthPeak is giving talented developers a chance to focus on creating their games without spending the resources to create an engine to support it."

"Our focus on UE3 technology has been key to our success as an independent studio. This is an ideal opportunity for us to work with SouthPeak on our projects," said Dave Hagewood, president of Psyonix Studios.

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

PURCHASING POWER

bundles offered in MABINOGI are the "extra storage service" pack, a variety of items plus an increase in storage space; the "support service" pack; which assists the character during play; the "advanced play service" pack, composed of various items that help the character to grow; and the "fantasy life club," a super bundle of the other three bundles that costs 30 percent less.

Abstracting the function of items. If an in-game item purchased by real money can practically enhance gaming capability of a game character in a casual online game, it is efficient not to disclose the actual data of increment to the player. If the data is disclosed, other players who are not using the item may be deprived of their sense of achievement. In addition, if the distinct differences between such items are disclosed to the players, there may be a rush on a specific item, and this may have an adverse effect on the balance of the game.

To avoid such cases, developers can explain an item's function using descriptive text, rather than reveal the hard data.

Strengthening events and communities. Game developers and designers should consider selling in-game items during specific events. It would be an especially good way to sell items that do not have specific functions other than decorative effect. For instance, Christmas-themed items for ornamental purposes can reap a large volume of sales in a short period of time.

MMORPGs can take advantage of selling items to whole communities. Guilds are a typical representation of communities, and selling related in-game items should be designed to assist the existing guild system or help promote guild plays. Guild-related items in MMORPG could be categorized as shown below:

Guild item: An item providing special effects to all members of the guild.
Guild emblem: A guild emblem inscribed in garments or equipment for members of guild.
Guild PR: Advertisement to recruit members for a guild all across the server or the map.
Guild equipment: Equipment with high functions and special effects to be used in guild-level quest.

When guild play is actively conducted in the game, players are inclined to be more loyal to the game, and game companies can expect to see an increase in sales of other items irrelevant to the guild.

EFFECTS OF THE ITEM-SELLING MODEL ON THE INDUSTRY

As item-based payment models expand in online games and beyond them, the game industry will most likely be affected in terms of how it handles game design, sales, and distribution. We expect game designers will take a serious interest in deciding which items to sell in a game, how they will be sold, and with what currency, as much of a game's balance relies on how these issues are managed. We also expect to see more research on payment systems, particularly those that can handle a high number of small transactions in a short period of time. ✴

items for purchase in kart rider and special force

Plazma PT. Plazma PT is a vehicle that lower-level players often purchase; it increases your odds of winning in the game. Expiration dates vary from seven days to one year, depending on price. (KART RIDER)

Red Missile Balloon. Red missile balloon is one of the decorations that players can add to their vehicle in Kart Rider. Once this item is mounted on the vehicle, it will automatically obtain Lucci when attacked by missiles. Single purchase may be used up to 500 times. (KART RIDER)

Lunar Rabbit. Lunar Rabbit is a vehicle that represents a traditional figure from Korean literature; this item can only be purchased during Chusok, the holiday of thanksgiving. Lunar Rabbit is an attractive vehicle that enhances the player's capabilities and has a special function that deletes items that would otherwise impede the player during the game. The vehicle has a high price tag and other restrictions that make it difficult for players to obtain. (KART RIDER)

Replay camera. The replay camera allows a player to record his or her racing during the game. Since it provides a function irrelevant to the game itself, it could be purchased at a relatively low price. (KART RIDER)

Double up. This is a Cash item with an expiration date. Once this item is purchased, SP earned will be doubled during the game, thus allowing players to earn SP quicker and buy better weapons. (SPECIAL FORCE)

Cross hairs. This is a Cash item with an expiration date that lets players modify the form and color of the cross hairs on a gun. Considering the characteristics of a first-person shooter, enabling players to modify cross hairs can have a significant effect. (SPECIAL FORCE)

jobs.ea.com
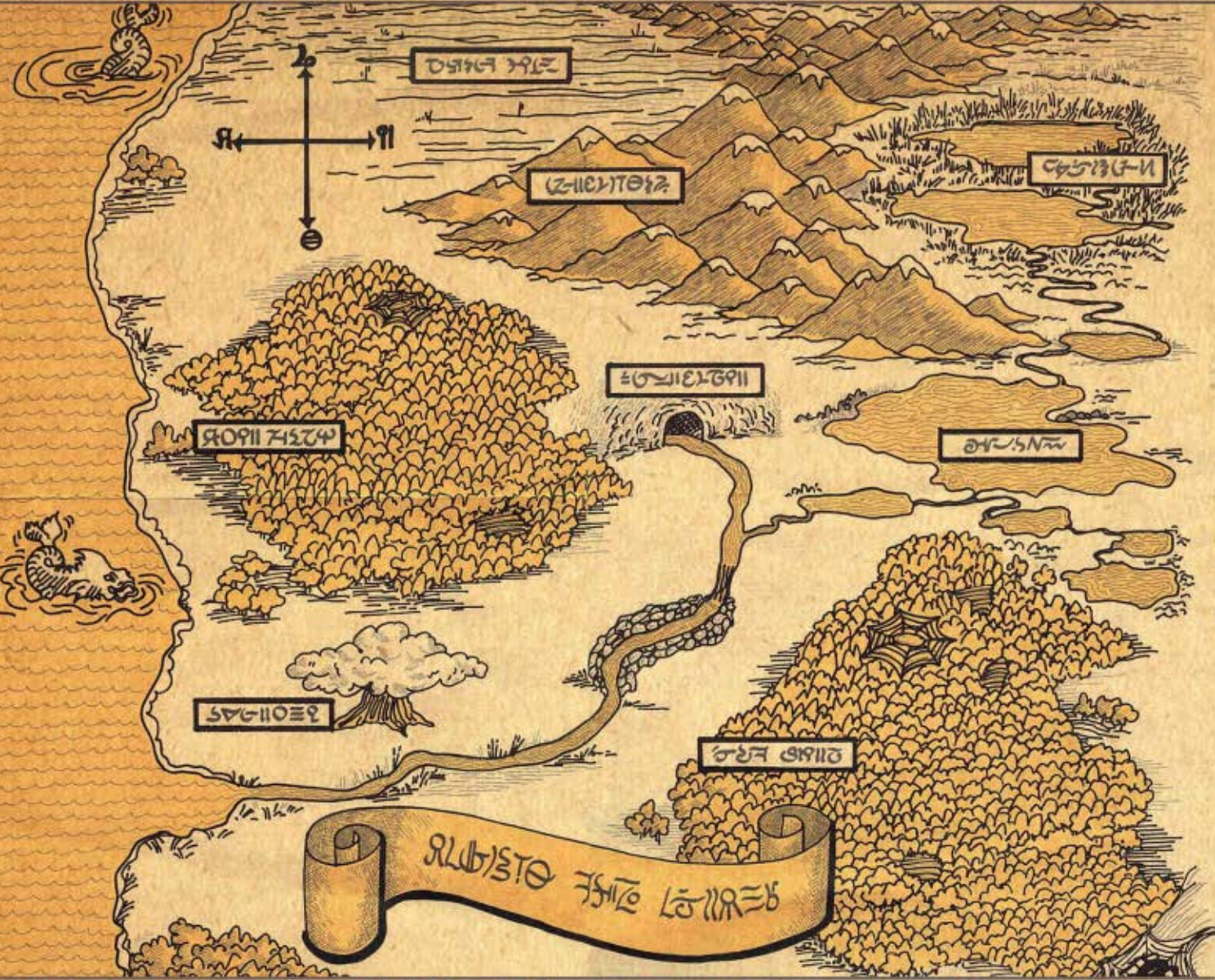
TEXTUAL HEALING

What Graphical MMOs Can Learn From Their Text-Based Ancestors

» stephanie shaver

IN THE BEGINNING, THERE WERE TEXT-BASED GAMES: ZORK, WISHBRINGER, and everything Infocom. Then someone came up with the brilliant idea of putting them online. Not long after, someone else slapped graphics over it and gave it a name. They called it an MMORPG—because heaven knows, the gaming industry needs more acronyms.

In spite of all the bells and whistles, you can still find the humble roots of the MMORPG in text-based online gaming. You just need to know where to dig.

That's where I come in, as a lead designer for Simutronics, which still makes text-based MMOs. Since we're all based on the same structure, there are still some lessons traditional MMOs can take by stripping away all the extras and staring directly at their roots. Though I can't possibly touch on all the lessons I've learned throughout years of playing and creating these kinds of games, I'm going to try to give you a few.

I'll be referring to my company's text-based games by their engine name—the IFE or Interactive Fiction Engine. This will save me some

Through strange, savage zones your way will be shown by the magical stone called **WISHBRINGER**

Your greatest challenge lies ahead–and downwards. **ZORK I**
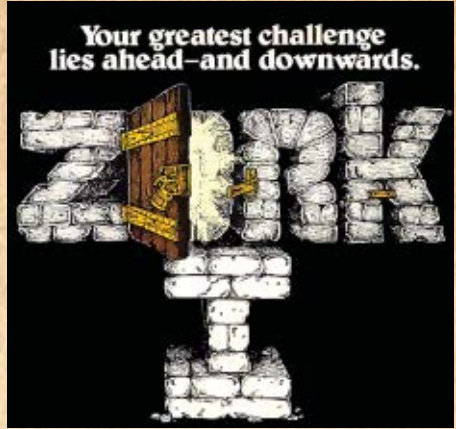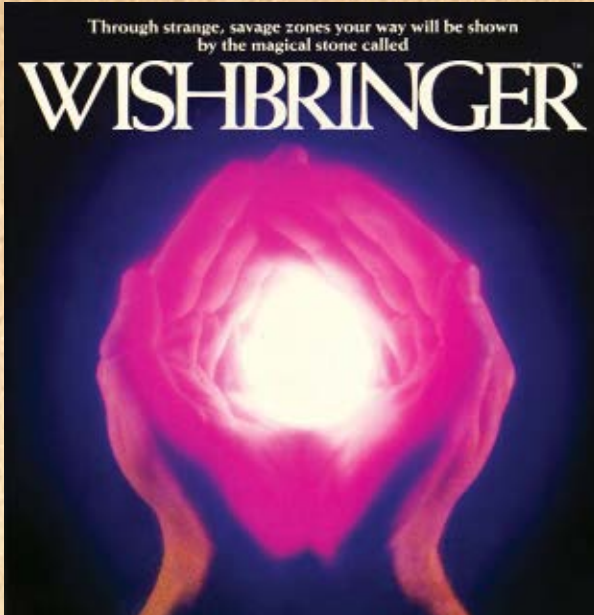
typing and your eyes from the monotony of seeing "text-based games" a zillion times. This also separates them from MUDs (Multi-User Dungeons) and MOOs (MUD Object Oriented), the text-based adventure engines that are beloved of time-killing university students everywhere.

COMMON GROUND

Simutronics GameMasters are not the GameMasters you see in other MMORPGs, or the Wizards in MUDs. Though our GameMasters do resolve support issues, they are also responsible for building in-game content like quests, areas, and code. Many of the game developers currently working full-time at Simutronics started as a GameMaster—essentially, a player who chose to take the next step and develop the game they loved rather than play it. So when I talk about a GameMaster in this article, presume I am referring to my company's definition of them.

When I refer to an MMO(RPG), I speak in the current definition of the genre: a graphical online game that usually involves levels, creatures, a hunt cycle, phat lewt, 14-year-old boys pretending to be nubile elf/alien chicks, some measure of grinding, and so on.

This is despite the fact that I consider IFEs (and games like them) to be the original MMOs. Strike the graphical part of that definition above, and you may as well be describing GEMSTONE IV or DRAGONREALMS or one of our late, lamented competitors (like DRAGON'S GATE or ISLAND OF KESMAI). The only difference? A rendering engine.

Well. Mostly. The fact is, you can't port an IFE into an MMO. There are things you can do in text that you just can't do graphically. I equate it to trying to adapt a book to movie format. Both have their benefits and their drawbacks. Let's look at some of the lessons that do translate.

SOLO IS THE MORTAR

We say this all the time in design meetings, and we're all pretty much in agreement: solo is the mortar of our IFEs, and I consider

that to be a shining universal truth that translates to MMOs. Your game may be visually stunning with the greatest Looking For Group tools anywhere, but if I can't chug along on my own while Chuck Norris[1] references fly by in the chat panel, I'm going to get bored.

Players—especially new-to-the-genre players—frequently start solo and graduate to groups. Even if they get brought in at the insistence of a friend, eventually they're going to venture out alone. We've never built a text-based game where advancement of your character required a group. No matter how organized and drama-free your guild is, there will be times when you want to play and no one is on or available. If a game doesn't make it possible for me to jump in and kill on my own, it's going to lose my subscription.

COMMUNITY IS EVERYTHING ELSE

Solo's the mortar, but the reason people keep playing an MMO is simple: interaction with other people. The smart MMO game designer knows that the game isn't as simple as taking a console or PC game and throwing in server-side technology. The smart MMO game designer builds mechanics into the game to encourage people to meet people.

"Massies" in GEMSTONE III are an example—a mid-level spell that a character could mass-cast to defensively buff everyone in the character's group. And since "groups" in GEMSTONE III are essentially limitless, it's also a great way to make friends and influence people. A cry of "MEG in TSC, 5 min!" was gamer shorthand for "Get your butt back to town in five minutes, because I'm going to cast a spell that will boost your defense by a significant margin for the next three hours." On top of that, within those groups you'd have people who'd start to channel mana to the caster so that he or she could cast more spells. It's the kind of community mechanic that is unique to these types of games.

Certain aspects of some classes were also refined to encourage interaction. If you found a locked box, you also needed to find a Rogue to pick it. If you got a wound, chances were good you were going to need an Empath to heal you. Unidentified item? Get a Bard. Wand low on charges? Get a Wizard.

Some of these dynamics must change in the MMO space, where player attitudes are different due to the wider audience and broader community. For instance, the global player "chat" in our IFEs can only be reached if you have a special item. MMO players, however, expect zone-wide chat to be free and available. It's a genie you can't put back in the bottle, and

1. Not that there were Chuck Norris references back when GEMSTONE III was on the late GEnie network. In fact, due to the luxury of the small-game environment, even today in those games the mere breath of an OOC (Out of Character) comment on the game-wide channel will cause you to get bombarded by the roleplay cops. Ignore them and you'll get warned or—even better!—tossed into The Cell, a special room in the game where the Gilligan's Island theme song plays in the room description. I always wished we could get it playing over the user's speakers, too, but that would have required licensing and money and lawyers, and tempting though it may be, it probably wouldn't be worth it. Well, maybe a little.

STEPHANIE SHAVER is lead designer for Simutronics Corporation's HERO'S JOURNEY. Email her at shaver@gdmag.com.

A shady tree in the Grenmeer Lowlands, part of HERO'S JOURNEY's early experience.

shouldn't try to—because, quite honestly, it's just too darn helpful to be able to chat anytime, anywhere.

Framerate also must be taken into account. The mass-cast spell mentioned above, though great in a text-based game, would be a disaster if directly ported into an MMO. A general rule of thumb is that anything that brings scores of players rushing into the same zone all at once is a bad thing. Some games solve this by intentionally authoring their textures to be lower resolution so they can fit more characters on the screen, or with instancing (creating different copies of the zone based on load). Theoretically, you could even solve it by making the mass-cast zone-wide rather than within a radius of the player. With the proper messaging, balance, and event effects, that would be one spell people would be thrilled to see. It would also bring attention to the players who are actively interested in contributing to the community, making it stronger, better, and a nicer place to play.

YOU CAN'T GREET THEM ALL

Once long ago, when the games only pulled in a hundred players on a good night, we used to send a GameMaster down to personally greet each and every new player who logged into the game.

Back then, the only way to access our games was by logging onto the GEnie network. The Internet as we know it was still gestating in the hideous womb of progress. Our game was quieter and events

and interactions more intimate and personal. We got one, maybe two newbies a night. Greeting them was no problem—until we became a content provider for AOL. Then everything went to hell.

In the late 90s, AOL used to run "banners" that advertised our games. Usually we got a warning that this was going to happen so we could be ready for it. The influx typically brought anywhere from ten to twenty times the number of players we were used to in the game, a mind-boggling number for us back then.

I remember fondly the hours I spent frantically greeting the masses. A third of these were mute. Another third stood around screaming things like "How do I talk?" or "How do I save the game?" or "I am the great Cornholio!" A final third would bypass my greeting completely, making a run for the game at large and escaping the Beavis and Butthead references.

Ultimately, out of a strong sense of self-preservation, staff developed automated systems that have been refined time and time again. Nowadays, we know this process as the "tutorial", and it's turned into a monster all its own. Tutorials can be elegant, but too often they're an overbearing mother smothering you with help.

You can't greet them all, but you do need to greet them with something—just so long as that something can kindly take a hike when the player wants it to.

PIMP MY AVATAR

Ultimately, the thing that set us apart was that we customized like crazy. I suspect this is one of the little things that compels our community to stick around.

In the IFEs, the cream of the customization crop is the alteration. That is to say taking an item from a player's inventory and uniquely altering either the description (changing a blue sword to a rune-etched falchion for a small fee), functionality (adding fire damage for a larger fee), or both (ca-ching!). On some weekends, I would spend up to eighteen hours as a special merchant character, altering, roleplaying, and getting intimate with my dictionary.

These were no small affair. If we'd had a CLAW command, players would have torn our doors to shreds waiting to get into the merchant tents. They waited hours for the chance to get a sword, a shield, a single piece of armor—of which there could be twenty or more—changed.

To this day, they still wait for those chances. Unique alterations are by far one of the biggest draws of the text-based festivals. This yearning for character uniqueness is something some MMOs brush up against, but rarely follow through on. One notable exception: CITY OF HEROES—I could play with their character creator for days, and know people who have.

There are quite a few reasons why one might not want to go the crazy customization route. For one, texture and memory space. Never mind the monumental task of organizing the artists needed to create all that art—keeping the memory load down so that the game is playable is a major technical issue that has to be addressed from the get-go.

Design-wise, the choice and limits of customization must be part of the game from the beginning, and some designers just don't see the value or urgency in allowing players to look the way they want. This is an attitude that came out of single-player games (where wardrobe and appearance can often be limited to the character you're supposed to be playing) and the very earliest MMOs where one piece of armor could display five different ways depending on what model your character was. Didn't want your helmet to show up as a snarling wolf's head? Well, damn, you shouldn't have created a Barbarian!

But that's the thing with an MMO—when you're one character in a world that's full of Barbarians, looking different is vital. I could tell you in detail about the way I dressed and trained a GEMSTONE III character from 1996 and why I did it. With the aforementioned exception of CITY OF HEROES, I can't say the same for any of my MMO characters, because the option to make them truly unique was taken from me. I don't care about the character in an MMO now, I care about the items and their power—and those are two very different things.

All designers believe (or should believe) in the need for great gameplay, and don't get me wrong—I agree in the value of a really fun, really playable game. But part of great MMO gameplay is that a pattern is established—and then it's broken. The visual repetition of avatars can be just as ruthless as a bad grind. Breaking up the monotony is good, and years of experience with the IFEs have proven that players love it.

INVASIONS AND WEDDINGS AND WEDVASIONS

You know what else players love? When you drop a ravenous hellhound on their head. Invasions are gleeful, evil fun for players and GameMasters. No matter what game you play, if you throw the ravenous hordes at a town, the denizens will charge back to defend it. That sometimes results in a server crash, even in the IFEs. The same consideration must be taken here as for anything that coaxes players to run back to town en masse.

Most of the tools we use to run invasions have translated to HERO'S JOURNEY, our upcoming graphical MMO. Creature "possession" (when a GameMaster can control a creature), creature spawning, system-wide messages, special treasure—these are cheap tricks that yield big fun. Of course, there's also a staff calendar for tracking who's invading when, and why, and with what, and there are long-term storylines. None of this is brain surgery—it just requires a staff with the tools, vision, and the proper attitude to pull it off.

Organized weddings are more deliberate, and we haven't yet determined exactly how we'll do them in HERO'S JOURNEY, but very likely it'll be through a ticketed system on our site—which is to say, it's something players will have to pay for. Choose a package, choose a locale, choose a dress, choose a ring, start sounding like the spiel from *Trainspotting*, designate your wedding party, and spend the next two hours roleplaying out your fantasy wedding.

"Wow, that's ... dumb," you might think. "Why would anyone waste time writing tools for that?" The answer is simple: because people want it. Put another way—at one time in the

One of the Broken shambles through the Quillmore sewers of HERO'S JOURNEY.

TEXTUAL HEALING

A Gearknight Human showing off her shiny Gear-armor in HERO'S JOURNEY's Grenmeer Lowlands.

IFEs we had a super deluxe wedding package of awesomeness. Your own area, clothing, the personal attention of two GameMasters, a major amount of work. We priced it at $1,000 in real money, uncertain if anyone would ever buy it. It sold in less than 24 hours.

IF YOU BUILD IT, THEY WILL EXPLOIT IT

Another AOL experience—the day someone logged in with the name "The". This was bad, because the IFE parser stripped out that word, which meant the character could not be referred to by other players. Naturally, I went down to deal with it.

"Hi!" I said. "You need to reroll and pick a different name!"

"Why?" The asked. "I like this name." He smiled at me.

I attempted to explain. After a while, The said, "I see. Do you know who I am?"

"Nope."

The proceeded to tell me. I vaguely recognized The's name being that of a high-level AOL exec. I confirmed with my supervisor and an account name check that The was, indeed, said exec.

"I'm sorry, sir, but you still need to change your name."

"Also," I typed on the GameMaster-only chat channel, "'The' is a stupid name for a Giantman Warrior."

And then The said something that has stuck with me to this day, because it broke open the shell of my small-game naiveté.

"Well," said The, tossing me another smile emote, "maybe your system shouldn't have let me pick it."

The's charming attitude is also true. We can lament the death of chivalry all we like, but as long as sites like Something Awful and Fark exist, the theme of "if you let me do it, I will" remains a pervasive sentiment.

Today, the character naming systems of the IFEs have also been bulletproofed and enhanced. You won't find players named "Brownnote" or "Britneyspears" in our games. We have tools galore for dealing with that, and our GameMasters are empowered from the start with ways to deal with them. As a company, we know game developers can't be omniscient, and as the ESRB likes to remind us, game experience may change during online play.

Of course, bulletproofing that experience is an artform in and of itself, and no one can ever get it 100% right—which is why we'll need logs. Lots of logs.

YEAH, WE TRACK THAT, TOO

Here's an area where the IFEs keep improving: the game and staff's ability to track, compare, and verify a customer's history. We keep extensive logs about player activities—money spent, money gained, special items acquired, IP addresses used, special achievements met, warnings issued. Our game staff who handle player issues document every official conversation they have with our customers. The guys who handle our phone lines in-office lean on these logs extensively to do their job as customer service representatives. It's vital to our operation.

Recently I had a friend lose his character in an MMO. It took a month for him to get it back. All this time, he was writing customer service over and over again, and each time got flipped back to the start of the loop. They didn't ask him to clear his Internet cache or check to see if his monitor was plugged in, but they might as well have. The experience was frustrating, and only got resolved by him finding an error log on his own computer and sending it to them—but not because customer service asked him to do it. My friend just happened to care enough to take matters into his own hands.

Said friend has gone from an avid player of that game to a disgruntled customer who'll probably cancel before the next billing cycle. The loss is to the MMO, which didn't provide its customer service with enough information for them to do their job properly.

LESSONS LEARNED

MMO players are a double-edged sword. On one hand, they will always, always think up crazy-ass stuff to do with your systems that you never planned on, and you have to respect that mightily. On the other hand, they can be annoying jerkwads that post your home address and phone number on the forums, pushing you to fantasize deathtrap contraptions to buckle them in, a la the *Saw* film franchise.

The MMO experience, more than any other, has created a rapport between developer and customer that is part of this whole crazy age of communication. Learning to harness that and use it to its best end—not getting lost in it, but also not ignoring its value—is a lifelong negotiation. Excellent tools and optimism will get you through it all. Good luck. ✄

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

GameDevelopers®

08

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
- Participate in industry specific forums

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

2008

CMP
United Business Media

Learn more about myGDC at www.gdconf.com

POSTMORTEM

**JOSEPH
M. TRINGALI**

*is general manager and
executive producer at 5TH
Cell Media, LLC. He's been
with the company since
2003 as a founder, along
with Jeremiah Slaczka and
Brett Caird. Email him your
DRAWN TO LIFE characters at*
jtringali@gdmag.com.

5TH CELL'S DRAWN TO LIFE

WE FOUNDED 5TH CELL BACK IN 2003 TO develop quality, original games with a positive message. This was easy to do at the start, since we had no office, no official employees, and three fun game concepts for mobile phones (a growing market at the time). Once those were done and we secured a publisher for them, our company landed an opportunity to develop our first licensed game. This was a necessary step at the time, but one that would take us further away from our core vision. We rented an office, hired employees and stumbled through the first few years of being an independent game developer. It was a period to learn tough, but valuable lessons on human resources, development process and running a small business.

I remember the first time Jeremiah, our creative director, explained the concept behind DRAWN TO LIFE. Most of us didn't understand it, but eventually we came around to see how brilliant the concept was and the Nintendo DS as being the perfect platform. Our company was still

busy working in the mobile game space, developing licensed properties and enduring the brutal "pitch, prototype, pitch" state of the contract market.

We put our technical director, Marius Fahlbusch on testing the feasibility of the concept. At the time, we were not official Nintendo DS developers, so he utilized his background in hardware engineering and went about programming directly to a hacked system. It took him 6 months, but we finally had a basic prototype of our Draw system. We proved the concept could be executed.

While we prototyped DRAWN TO LIFE, the mobile game market took a turn south and we ended up cutting most of our staff. This was the most difficult period we had experienced thus far and most of our focus was spent trying to stay in business. It was March of 2006 when Jeremiah and I sat down to talk about the future. We had some opportunities in the mobile space, but both desired a return to original work. We still weren't authorized by Nintendo to develop on the DS, we had no money, and our technical staff consisted of a single programmer, who started as an intern the year before and had just finished his Master's from DigiPen.

Despite these obstacles, we decided to stop mobile contract work and focus completely on DRAWN TO LIFE, which was getting interested queries from several publishers. Later that month we received a royalty check for our original mobile games which was eight times larger than normal and gave us the funds to prototype DRAWN TO LIFE to a stage where we could land a publisher. We mailed a video of our homebrew demo to Nintendo and they approved us as an official developer. We purchased one development kit and started porting over the prototype. In May of 2007 we hired some programmers that agreed to help us work on expanding the prototype to include some gameplay.

One evening, a project manager from THQ Wireless whom we worked with previously, sent an MSN message to Jeremiah, asking what we were up to. Jeremiah told him about DRAWN TO LIFE, and loving the concept, he put us in touch with THQ Europe's business development department. They put us in touch with THQ North America and one day (ironically, the same day we had flown to discuss a deal with another major North American publisher) we received a call from Sean Heffron, business development manager at THQ North America. He loved the concept and wanted to discuss how we could bring the vision to life.

Months went by, and eventually we signed a global publishing deal with THQ.

WHAT WENT RIGHT

1 **A GREAT GAME HOOK.** The hook of "Draw your own character and play with it" was the perfect elevator pitch. The fact that our game could be reduced to such understandable concept helped us gain both initial press and interest from several major game publishers. Since this was our first handheld title, we needed a concept that could take the focus away from our lack of shipped titles.

The entire game experience was built around supporting this concept. We decided early on that our desire was to build a game with purpose, not just expand upon the Draw mechanic and turn it into a gimmick. We set out to build three game modes, the central Draw experience, a village that was populated with the player's creations, and a platforming adventure where they would collect objects for the village. Each of these allowed the user to experience drawing in some way. The Draw experience acted as a test-bed of sorts, allowing the user to experiment with various characters and styles. The village was created to motivate the player to rescue the lost villagers and restore to them through drawing various buildings, objects and locations. The platforming adventure was where the drawing impacted gameplay most, and we packed in as many gameplay mechanics with the drawn objects as we could.
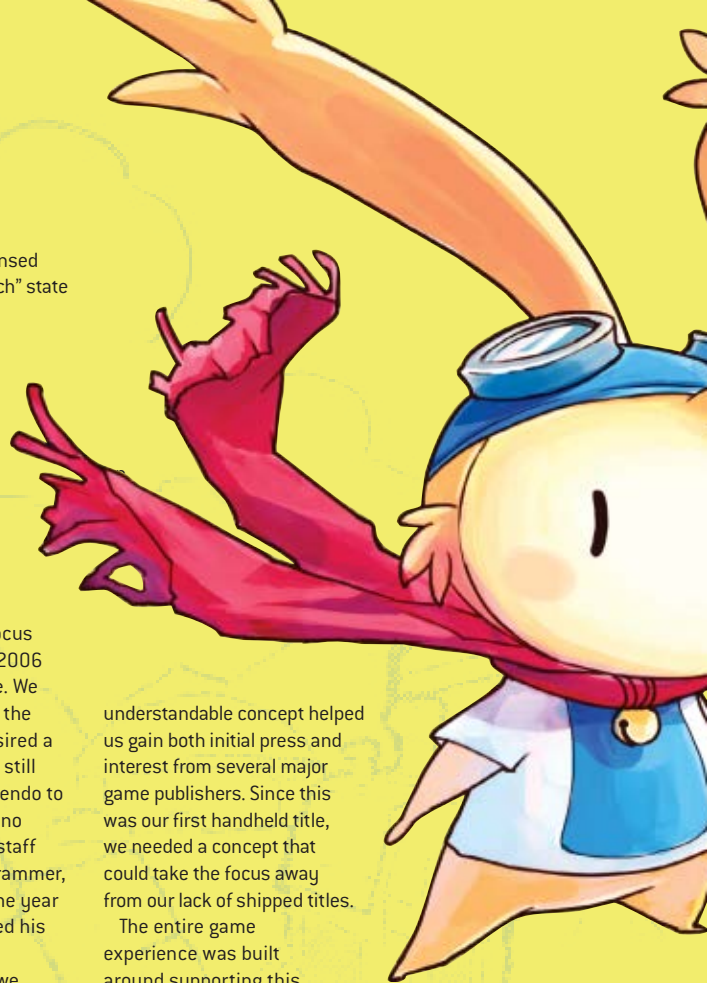
2 THE RIGHT PUBLISHING PARTNER. Choosing the right publishing partner is a luxury that most independent developers don't have. Often, financial or logistical pressures will force a developer to sign with whatever deal is currently on the table. This was the case for us, but thankfully we had an offer from a publisher that was willing to do everything possible to bring our vision to life—THQ. From the first conversations, it was apparent that THQ was serious about publishing original IP on the Nintendo DS.

THQ has an interesting setup, which differs from many publishers. They separate the project management responsibilities from the creative management responsibilities, so there was no overreaching executive producer on the publisher side to make unqualified creative decisions. This worked out great, and our creative manager, Scott Rogers, brought years of experience to challenge our creative vision. On the project management end, we had the experienced George Collins, who helped us make the difficult choices to stay on schedule. Both George and Scott represented DRAWN TO LIFE at THQ Corporate.

Arguably the most important publisher decision on DRAWN TO LIFE was the size of the save card. Larger save cards cost more money, and it's a common decision with publishers to go with the smallest possible save card in order to make more money per unit. THQ made the decision to allow us the largest save card possible so the user would be able to save everything they drew as well as two separate save profiles. This decision was critical to the success of the title, and proves their dedication to the quality of the game.

We also had the opportunity to work with a passionate public relations specialist, Karen Fujimoto. She loved the game, and arranged major press for us in magazines, web and trade shows. THQ marketed the game well, with TV and print ads as well as numerous regional contests. Adam Affrunti from first party helped bring our product through Nintendo LOT Check and we had a great Q/A staff, led by Andrew Frederiksen and Josh Gertz, which worked overtime to find the bugs in what was a lengthy and content-rich handheld game.
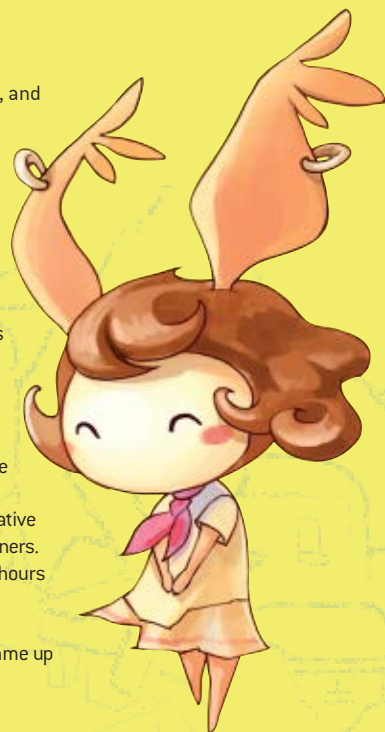
3 PERSISTENT DEVELOPMENT TEAM. We would probably still be working on DRAWN TO LIFE if it weren't for the extra time put in by everyone on the development team. This was led by the example of our lead programmer, Brian Firfer, who put in countless hours of overtime to get the job done.

We ended up making significant changes to certain game systems, which resulted in our programmers coding the same system multiple times. This was taken in stride, and we ended up with a very polished final project. Our crunch time started before first playable, and continued until we submitted for Nintendo Certification. All our developers gave up their Saturdays and worked late at night to meet all our milestones.

Despite the challenges, we made the most of it, and grew closer as the project continued on. We joined a Flag Football league in the dead of winter and nearly captured the championship. Activities like this—game tournaments and company-provided food helped alleviate some of the stress of extra hours.

Remarkably, everyone remained positive and focused on making DRAWN TO LIFE the best game possible. Our team is truly amazing and everyone's camaraderie makes each workday a blessing for management.

4 FOCUS ON THE CREATIVE ELEMENTS. After spending years developing mobile games, where quality and attention to detail took a backseat to branding and license, DRAWN TO LIFE unleashed a creative explosion among our artists, sound and game designers.

Our creative director, Jeremiah, spent countless hours going over the game with a fine-toothed comb, which resulted in a high level of polish for the final product. Our sound and music designer came up

DRAWN TO LIFE allows players to design their own characters.

with nearly fifty unique music tracks, totaling over an hour and seven minutes in length. We recorded a full vocal track for the ending.

We have always worked closely with Sherman 3D, an art studio based in Malaysia which is run by an associate and close friend of ours. They staffed up and handled the art load for DRAWN TO LIFE professionally and the final result was something everyone is genuinely proud of.

Finally, we decided to create an expandable intellectual property from the get-go. The village characters (Raposa), enemies, and story were created to build DRAWN TO LIFE as a brand, which could be expanded easily in any medium. The story was written with a strong emphasis on the character dialogue and creating attachment to the characters.

5 THE DRAWING SYSTEM. The drawing system was the first game system we worked on, and we continued refining it until the very end of development. The core of our game was drawing, and if we didn't get that right, the rest of the game would go down in flames.

From the start, we wanted a system that allowed the user to sketch a character, and not have to worry about self-animating or any complicated technique. We also wanted to add a layer of depth, so a trained pixel artist could create a professional-quality game character. We had the various levels of talent within our company continuously creating characters to see the potential of the tool.

During development we added the ability to draw on a per-pixel basis, and refined the stylus controls after receiving feedback that it wasn't accurate. We added various palettes, stamps, patterns, and a lock color feature that allowed the user to effectively shade their character.

One of the most unique aspects of the drawn character wasn't in the draw system at all. Due to the way the creation animated, we had gaps in the joints, since the users couldn't fill in space they couldn't see. We ended up making two passes at developing a system that extrapolates pixels from each side of the joint to fill in the breakage. The system succeeded in making the character animations appear fluid and is one of the more impressive technical feats in DRAWN TO LIFE.

WHAT WENT WRONG

1 NOT ENOUGH PRE-PRODUCTION TIME. Our early development was driven by the need for a publishing deal. We had a huge list of features we wanted to show off, and made the decision very early on to do whatever it took to get those features in the game, even if it meant cutting corners.

We also converted our proprietary level editor to use in building the game content. We did not take the time to optimize it for the Nintendo DS and the massive scope of the levels was more than it could handle. We lost time due to the sluggish performance of the editor, and in hindsight should have taken the time to refine it for the DS (which we did at the start of our current, unannounced title).

The lack of pre-production came back to haunt us during the final stages of the project. We finally had all the features in the game, but the core technology had reached the limit of what it could handle and we ended up adding more shortcuts to cover issues that were caused by the previous ones. We passed LOT check on the second attempt, and we're proud of the final state of the game from a technical perspective, given the raw number of assets and features crammed in. But we could have saved ourselves a lot of trouble by taking the time at the start of the project to build a scalable and robust foundation. Suffice to say, we secured enough pre-production time on the second project to ensure we didn't make the same mistake twice.

2 UNPREPARED FOR THE ART CONVERSION PROCESS. There was only one scenario where I will say our lack of platform experience really hurt us—the required art conversion process for the Nintendo DS. Coming from a mobile background that focused on a simple and straightforward method of using compressed .PNGs, the art management for DRAWN TO LIFE was a complete nightmare.

This problem was compounded by the raw amount of art we were cramming into the game. DRAWN TO LIFE ended up with over 35 unique enemies, 4 huge bosses, 30 unique Raposa villagers with very detailed and intricate animations, hundreds of level objects and 12 unique tile sets. This art needed to stay consistent to a specific palette, and making changes to the art resulted in spending extra time changing palettes and re-converting art.

We eventually had to put one of our level designers on the art conversion process nearly full time to keep everything up to date. We lacked the experience and established pipeline to manage this process. By the end of the project, we had over 15,000 unique pieces of art. Formalizing a process to deal with conversion and art implementation became the first priority for our current project.

Concept art for DRAWN TO LIFE.

Despite the positive attitudes, we could tell that everyone was burnt out at the end of DRAWN TO LIFE. As a company, we gave everyone some time off after we shipped, and we are now refreshed and hard at work on a second original project.

5 BALANCING THE GAME DIFFICULTY. We delivered our first prototype of the platforming mechanic fairly early in the development cycle. The word came back from THQ that the game was too difficult and we needed to make significant changes for the game to appeal past the core gamer.

As attached to the game as we were, it was smart to have an outsider evaluate the difficulty. Elements of the game that we found non-challenging, due to the extent we knew the behavior and timing of the enemies, made our prototype level unbeatable by evaluators at the publisher.

We also knew the game would have an appeal past the core gamer because of the draw mechanic and large number of DS owners that are casual and even non-gamers. We had to ensure that these gamers didn't get frustrated due to the difficulty being too high. We made a final pass late in development, adding more health power-ups and extra lives. We left the final level and boss difficulty the same.

Once the game reviews started coming in, we saw the flipside of our balancing decisions. Most reviewers, although enjoying the concept, said the game was too easy.

IT'S ALIVE!

Sometimes I contemplate all the decisions we made to bring DRAWN TO LIFE from a concept on paper to a commercial game on the shelves of every major retailer. The fact that we even completed a commercial, original Nintendo DS game given our starting point, was a miracle in itself. Despite the years in business and shipped products, DRAWN TO LIFE was the first game of ours that started as a real dream—something we were so passionate about seeing come alive.

I've been in the game industry for a number of years and there is an overriding sense of negativity about original games, especially from new developers. Don't buy it. Smart independent developers with a marketable concept have a huge logistical advantage over larger companies with a tremendous monthly burn. Hire developers that believe in your vision and do whatever it takes to move your concept forward. Be creative, outsource, and although there will be time spent off track, stay true to your core dream.

The industry needs more quality original games. We want to be part of a larger trend of independent studios with a creative vision that bring innovative and fun games to market. Our company's motto, "Advancing Entertainment," just about sums it up. ⠿

3 PREVIOUS INTERNAL PROJECT MANAGEMENT PROCESS DIDN'T SCALE. DRAWN TO LIFE was a learning experience in terms of our internal project management process and procedures. Previously we had very small, independent teams that worked on mobile projects. The largest single team we had was five people, which would triple for DRAWN TO LIFE. For the first time, we had multiple people working in each field, and we had programming tasks directly dependent on the timely delivery of art assets, level content tied with game mechanics, and a 300 page design document which underwent significant changes early in the development cycle.

Also, we had one producer (myself), and due to the budget and my background in writing, it became apparent early on that I would be spending most of my time scripting the village scenarios and writing character dialogue. We had no formalized asset tracking software and assignments were often given verbally or via email. This resulted in no central location to gauge progress and extra time spent communicating with each individual to get a sense of where the project stood as a whole.

Thankfully, our team was small enough that this wasn't a project stopper. However, it became a main priority for our second project to introduce better schedule and asset tracking, and we've put in place project tracking software and better communication channels to ensure a more structured process.

4 CONSISTENT OVERTIME WAS REQUIRED TO HIT OUR MILESTONES. This was touched upon in the positive section, but only because our team reacted so well to it. Consistent overtime is not fun, nor is it something that should be assumed when developing a game. We ended up working late weekdays and Saturdays for the entire second half of the project.

Our studio had the perfect storm of factors that contribute to crunch—lack of experience on the platform, an ambitious game, and a schedule that was too short. Combined with so little pre-production, we were playing catch up from day one, taking far too many shortcuts to implement the required game features.

We probably could have used more level designers and programmers, but as a smaller developer under budget, you have to make do with what you have. The extra effort required by our team would have been less with additional staff.

FILTER FORGE

BY JENNIFER SPARKS

IN THIS GAME DEVELOPMENT BUSINESS, you've got to have brains.

And guts.

And a whole lot of rusty diamond plate with black-and-yellow hazard stripes.

Yeah, you need a lot of textures, and painting them in Photoshop can be a huge pain. Thankfully, Filter Forge can make this a lot easier.

Released this spring, Filter Forge is a node-based filter editor that works as a Photoshop plug-in or standalone. When you think "filters," you may think of stuff like "lens flare," "plastic wrap," and "twirl." Well, Filter Forge does that kind of filter, but its real value is in creating high-resolution, seamless procedural textures. You can output bump, diffuse, specular, and normal maps (Direct X and OpenGL-style) for use in 3D, including support for HDR and floating-point image formats.

Filter Forge comes with access to thousands of filters, including magma, chain-link fence, crates (Warning: overuse of crate textures can lead to dizziness, nausea, and public humiliation), wood, stone, leather, tire treads, star fields, radar screens, animal fur, camouflage, parchment, eyeballs, brains, guts, and...rusty diamond plate with black-and-yellow hazard stripes. The Filter Forge user community is adding more filters every day—some incredibly useful, some incredibly weird, and everything in-between.

Finding and organizing your filters can be tough though, as the categories are limited and you can't yet add your own. With 3500 filters and counting, you can spend a lot of time wading through generic categories like "Misc" and "Creative."

You create a filter by connecting simple components in the drag and drop editor. Components include image adjustments (gamma, hue, blur, etc.), noise, and pattern generators. Other components, like sliders and checkboxes, let you create the user interface for your filter. Unfortunately you can't group components, so some processes that should be simple macros (rotate, scale, tile) must be built out of many little tedious steps. Other than that, it's pretty easy to get the hang of.

BUT WE... NEED...BRAINS!

Filter Forge comes with lots of organic textures—Figure 1 is Brains, by Michael Tannock. You can see the individual components that make up this filter. One of the nice things about Filter Forge is that filters are open—you can view and edit their inner workings. It's a good way to learn how to

FIGURE 1 Brains texture by Michael Tannock.

FIGURE 2 User-level controls are shown.

build your own.

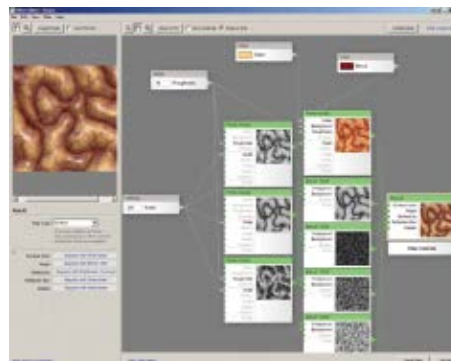Luckily, you don't have to dig into the components just to make simple adjustments. Figure 2 shows the filter's
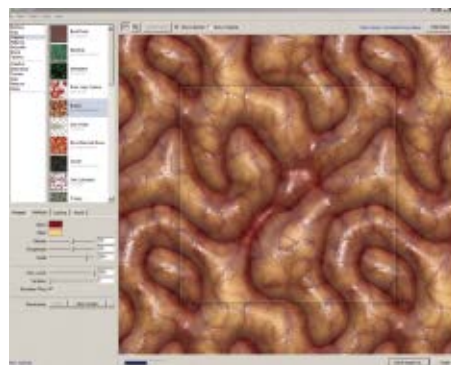

FILTER FORGE

★★★★

STATS

Filter Forge
901 North Pitt Street, Suite 325
Alexandria, VA 22314
www.filterforge.com

PRICE

$99 Basic Edition (filters only, no editor)
$199 Standard Edition (filters and editor, but no advanced image format support)
$299 Professional Edition (filters, editor, advanced image format and multi-core CPU support)

Note: Demo users can get a discounted or free Pro Edition as a reward for creating popular filters (see web site for details).

SYSTEM REQUIREMENTS

Windows 2000/XP/Vista.
Pentium III 733 MHz or better (dual core recommended.
128 MB RAM.
50 MB free disk space.
Screen resolution of 1024x768 or higher
Internet connection to download filters.

PROS

1. Can find a filter for virtually any texture or effect you might need.
2. If not, you can make one pretty easily.
3. Good online support and active user community.

CONS

1. Some components can be tricky to use.
2. Difficult to find and organize so many filters.
3. No way to save multiple maps at once.

FIGURE 3 Abnormal Brain—Do Not Use!
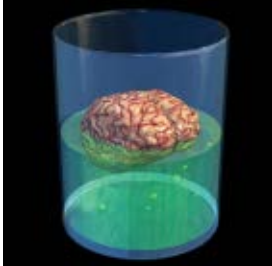
user-level controls: color, texture roughness, scale, etc. I've checked the "Seamless Tiling" option—the center square shows the actual texture, while the border gives you an idea of how it tiles.

The lighting in our brain's preview window is coming from Filter Forge's HDRI (High Dynamic Range Imaging) lighting.

This built-in renderer lets you make 2D images with realistic lighting and depth.

… AND SQUISHINESS

But if you're going to generate textures for 3D, you'll need texture maps. Filter Forge outputs a variety of maps—unfortunately, you can't automatically output multiple maps.

For the texture in the example below, I only needed diffuse and bump maps. Figure 3 shows the result in 3DS Max.

For the cost of a stock texture library or two, Filter Forge lets you create an unlimited variety of seamless, resolution-independent textures. As the old saying goes, "Give an artist one decaying flesh bitmap and they can texture one faceless zombie nurse. Give an artist a procedural texture tool, and

Filter Forge gets the old hairy eyeball.

they can texture infinite hordes of the ravening undead."

Or something like that.

*SPARKY is an independent game developer, graphic designer, and occasional writer of stuff. Send comments to her at **sparks@gdmag.com**.*

product news

SN Offers Free Distributed Build System For Sony Hardware
SN Systems
Sony subsidiary SN Systems announced that version 2 of its SN-DBS distributed build system has been released as a license-free download to all licensed PlayStation 3, PlayStation 2, and PlayStation Portable developers. SN Systems says the system "significantly reduces compile times by distributing source code builds between cooperating PCs on a local area network," and can build assets via the command line.

Version 2.0 includes the building of multiple projects in parallel, makes faster builds than its 1.4 version, has an improved GUI with project view, and improved support for asset builds. The developer notes the product is multiplatform and will work for projects on all Sony systems when used in conjunction with a platform-specific compiler. Finally, SN says support fees will no longer be associated with this product, but that it will continue to offer support and develop SN-DBS on an ongoing basis following feedback from game title developers.
www.snsys.com

Unity 2.0.1 Released
Unity Technologies
A free 2.0.1 update for the new Unity 2.0 game engine has been released to address issues introduced by Apple's new Mac OS X 10.5 "Leopard" operating system.

Unity 2.0 is a major update to the Mac-based game development engine and includes 50 new features and a variety of other product updates. Unity now has the ability to play video files as textures on any surface and a new way of developing content GUI systems. Unity's core engine has been optimized for speed and can make dynamic rendering adjustments to ensure the best performance across a variety of machines. DirectX 9 rendering is also included for standalone games running on Windows.

Workflow improvements include the new Unity Asset Server, which allows teams of developers to stay in synch and provide project source control capabilities.
http://unity3d.com

Monumental Offers Free MMO Middleware
Monumental Games
MMO middleware and online game developer Monumental Games will offer a fee-free limited license of its MMO

middleware dubbed the Monumental Technology Suite to developers for the purpose of creating prototypes. The suite is based on a number of component systems that form a complete toolset, including server technology, client technology, a graphics engine, and development and management tools.

Intended for smaller development studios, the new license allows full, unrestricted use of the Monumental Technology Suite for prototyping to help offset the cost of licensing a commercial game engine. Under the new model, studios can develop an operational prototype of their product to better position them to secure funding, publishing, and distribution deals.

The Monumental Technology Suite can be used throughout all stages of the online product development lifecycle, from prototyping, pre-development and development to live-operation and management. It acts as a complete standalone solution for product development requirements and is designed to integrate with existing production pipelines, development tools, systems, technology, and processes.
http://monumentalgames.com

INDEPENDENT GAMES FESTIVAL

NEW THIS YEAR!

INDEPENDENT GAMES FESTIVAL MOBILE

LEARN MORE AT
WWW.IGFMOBILE.COM

THE 10TH ANNUAL

INDEPENDENT GAMES FESTIVAL

FEBRUARY 20-22, 2008 • SAN FRANCISCO, CA

FINALISTS ANNOUNCED IN DECEMBER!

CHECK OUT THE GAMES PUSHING THE
INDIE MOVEMENT AT:

WWW.IGF.COM

PLAY THE BEST INDIE GAMES OF THE YEAR

THE IGF PAVILION AT GDC 2008
FEBRUARY 18-22, 2008
SAN FRANCISCO, CA

WWW.GDCONF.COM

CMP
United Business Media

>> **b r a n d o n   s h e f f i e l d**

HEAVEN'S NIGHT
AN INTERVIEW WITH AKIRA YAMAOKA

WHEN SILENT HILL WAS RELEASED IN 1999 IT LOOKED DIFFERENT from other video games. But perhaps more important, it *sounded* radically different.

With music and sound design by Akira Yamaoka, the sonic environment of SILENT HILL was an inseparable part of the game's sepulchral mood. As players explored SILENT HILL's fog-shrouded streets and decaying halls, Yamaoka draped the world in vast sheets of sound. Often suggestive of air raid sirens, background radiation, or the quiet hum of a dialysis machine, Yamaoka used guitar and electronics to create a sound that was alternately lush and bracing, with melancholy dreamscapes dissolving under a rain of lacerating distortion. Video game music could never sound quite the same again.

Although most recognized by his grim compositions for the SILENT HILL titles, Yamaoka isn't all serious, having contributed upbeat, crunchy pop tunes to Konami's BEATMANIA and DANCE DANCE REVOLUTION games as well as a convincing hair metal turn for RUMBLE ROSES—not to mention his CD tribute to 1980s fluff.

Looking forward, Yamaoka stepped into the role of series producer on SILENT HILL 4: THE ROOM and is now guiding SILENT HILL as it becomes one of Konami's key franchises.

Brandon Sheffield: *How did you get into composition from the beginning?*

Akira Yamaoka: I wanted to be a designer in the first place, then I started work with CG using a computer, and while working with CG, I learned that I could make music using the computer. So yeah, I was a designer working with CG stuff, but I began to play around with music and thought 'hey, this is interesting!' Granted this is a story from more than 20 years ago, but I guess that's how it started.

BS: *So you taught yourself music?*

AY: Yes.

BS: *When you were first making music 20 years ago, it was chip tunes? Like Famicom sound?*

AY: It was actually exactly Famicom sound. Only three sounds at the same time.

BS: *How different do you find working on the* Silent Hill *movie, versus working for the games? (Original SILENT HILL CG creator) Takayoshi Sato said that it was like looking into his past, when he watched the movie. I wonder if you felt something similar.*

AY: Well with this one, I really didn't feel like I was making a movie. The director Christophe Gans really wanted us to work together on it, so I went to Tornoto many times to really work closely with him, to help the movie do justice to the game. There wasn't really a lot of difference from working on the game.

BS: *And he used some of the same camera angles from the original SILENT HILL as well.*

AY: That's exactly what the director wanted to do. He had a 40-inch or so TV screen on the set, and he played the PlayStation 2 on the scene. When he gave instructions to the actors, he looked at the game, and gave instructions.

BS: *When you were composing the music—and I know you say it's an extension of the game—did you feel you had to put more into it because it's a different format, or did you have to*

BRANDON SHEFFIELD *is senior editor of* Game Developer *and* Gamasutra.com. *He lamentably has still not finished SILENT HILL 2. Email him at* **bsheffield@gdmag.com**.

HEAVEN'S NIGHT

treat it differently, maybe because of higher-fidelity sound and better speakers?

AY: Yeah ... I didn't want to do that. Basically I didn't want to change the music from the game's feeling. I actually used the sound that was similar to the PlayStation One games. I didn't want to make the game music like a movie, I wanted to make the movie more like the game.

BS: *Is there going to be a sequel to the movie, and are you going to be working on it if so?*

AY: Well there's nothing announced, but there's some production web site out there talking about it I think. If I got the opportunity I'd love to do it again.

BS: *I'm curious to know what you think of interactive music right now in video games. Is there more that we could be doing with it? Games like Q Entertainment makes, or all the other companies that are trying to do that.*

AY: The interactivity is what makes games different from other entertainment—music, movies, DVDs, or books. Interactivity makes the difference. People who design games are seen as creating interactivity, and I think musicians should also be recognized as part of that ... what should I call it—entertainment group? Something like that. It should be recognized in a different way. Maybe it should be a new category, but sound and music creators need to be thought of as part of that interactive creation process from the beginning. I think if you do that, something different will happen, just from a change of project perspective. I've always thought you should get everyone thinking about the interactive and creative aspects of the game together, from the very beginning.

BS: *In the SILENT HILL series, for instance, most of the interactivity of the music and background is either just sound effects or two tracks of music—one that's just normal state, and the other that is when you're in the SILENT HILL state. One goes down, and the other goes up, and then reverses. Do you think that you could go even deeper than that, and make something in which a player's actions really affect how the environment works and reacts to them?*

AY: Oh yeah, we could go way deeper. There's nothing to say that we need to just have static state changes all the time. There's no limit. You really should be able to make the sound respond to the players' actions or movements. It's not just like "battle music start," or "ambient music start" and then crossfades like you were talking about. I think it's really important to go beyond

that. I keep thinking I'd like to have the games and the graphics really and truly agree with each other. But it's still a game. I don't really want to make it virtual, I don't want to emulate reality.

BS: *It seems like you could have several similar tracks running simultaneously that could thread in at different times. Not even*

just music, but also ambient sound that will really bring the player inside of the environment.

AY: Yeah, I think that's good.

BS: *Will you be trying to do that on the next SILENT HILL, or is that even further in the future?*

AY: Hmm, after all, the next one's going to be on a next-generation platform, so we'll utilize Dolby surround sound of course. We're trying to do some new things, but it's nothing like the type of interactivity I was just talking about. The music presentation could be more detailed.

BS: *What is it like working with an American team for SILENT HILL 5? Is it different from having a team in-house with you?*

AY: It's completely different working with an American team. There are of course advantages and disadvantages, but overall, I'm really impressed with the American staff and their technology. Their graphical and technical ability is amazing. There's a huge gap, actually. They're very advanced. I'm Japanese, and I think this is not just with SILENT HILL but with the whole of the industry—I look at what American developers are doing and I think wow ... Japan is in trouble.

BS: *That's interesting, because I've been noticing this for a while. Technology in games in Japan is falling very far behind. Usually I have to ask people if they think so, but you said it without prompting. Why do you think it is? Is there just not enough communication between companies, and sharing of technical know-how?*

AY: There are two reasons I think. One is that the development environment in Japan is divided into developers and publishers. Publishers have to create a game in a short amount of time at low cost, and it's a lot of pressure on them in that respect, and they pass that on to the developers. So basically it has to be done as quickly and cheaply as possible. And the people doing this are getting old like me. And tired! And the salary isn't that great. So you've got pressure on these people to perform like they did when they were 20, and it's just not possible. I look at a game magazine, and I see interviews with the "important creators," like Mr. Sakaguchi. He's a great game creator, but he's not young. And I don't see many young game creators in Japan. Then I look at the West, and I see all these young guys coming up so fast, it's just amazing.

The second reason is that … well for example on another project we were in development for a while, and we realized that we needed a new driver for some graphics program. That happens of course. So we looked around for it, and we found it, ok. Same maker, same everything, should be fine right? But the problem is they're all in English. So we get this thing and we have to localize it into Japanese. We don't have a lot of people who can understand English deeply enough for something like that, so that reduces speed. And while we're waiting for that, we're already a step behind everyone else who can understand it intuitively. This sort of thing builds up, and we just fall further behind. I mean of course we can understand it once we know what it says, but this falling behind really affects the quality of what we can do. So that's the second big problem.

BS: *To that point, it seems like in Japan every company has the same problem, but they're all working on it separately by themselves. Here, we have people using the same engine, and if they're using the same engine, they'll compare ideas and problems and fixes and things like that. That's my perception of one of the problems.*

AY: Definitely, that's a big one. Like back in the Famicom days, people didn't want other companies to see what they were doing to maximize the console. And it's not even company to company, even the same company with two different projects, those two teams won't share driver research or resources like I was talking about before.

BS: *I've heard of other companies where two teams are working on the same type of game, but they didn't share an engine or assets or anything. Same company, two similar games, totally different tools. I've been feeling as though that is the reason why next-gen games are not taking off in Japan, because Japanese technology has not made it easy for Japanese game creators to make games for their own market.*

AY: Yes, that's absolutely true. There really aren't people who can use the tools. The people who are starting to learn this stuff in Japan are still rather green too, so they can't even meet the levels we need to get to. ✖

The author acknowledges Jeffrey Fleming for writing the introduction.

On Yamaoka's iFuturelist CD

iFuturelist, released in 2006, was Yamaoka's first non-game-related album release—a complete stylistic shift from his usual work.

Brandon Sheffield: *What was your inspiration for the project?*

Akira Yamaoka: It's different from SILENT HILL-type music. I think it has a lot of smiley humor. Smiley? Something like that. I started work on it about three years ago.

BS: *It has a lot of humor, yeah.*

Some of the songs are very strange, like Lionzuki (which means "likes lions"). What were you trying to say with those songs?

AY: I like '80s music very much, like electro-type music. It sounded cool and soulful. When I was a student, I thought this one song was really cool, and I checked the lyrics, and I found out they were the complete opposite of what I thought. For example, that song said something like, "If you smoke too much, you'll get lung cancer. So you should reduce tobacco." Something like that. It was very shocking to me, to find out that the lyrics didn't have any meaning. So I like those lyrics after all, when they don't really meet with your expectation of the music, and that's what I wanted to do with my music, like with Lionzuki. It's good music, but the lyrics don't have any meaning.

BS: *So do you choose words based on the sound of the voice and the pronunciation of the syllables?*

AY: Yeah, it's really just about the sounds. So like with Lionzuki, it just sounds good.

BS: *Actually, my favorite song on there was more like trance-type sound. I think it was number five. That was the most normal one, so I feel bad that it's my favorite.*

AY: But that trance-type music has some originality to it too, I hope. I wanted to add that smiley humor as well.

MICK WEST

>> THE INNER PRODUCT

REFLECTION FOR GAMES

REFLECTION IS THE PROCESS OF exposing information (called metadata) within code at run-time that allows the code to inspect, use, read, and modify objects without being hard-coded to do so. It both encapsulates objects in a modular fashion and makes aspects of them accessible in a general purpose manner.

There's a wide range of ways to implement something like reflection, from ad-hoc solutions using macros, to more complex and powerful systems involving custom compilation steps, or templates. Here I'll give an overview of a simple method that could form the basis for a reflection system. In the process I'll explain some uses for reflection in game development.

WHAT IS IT GOOD FOR?

In games programmed in C++ (as most currently are), the most common use of reflection is serialization. Serialization is the conversion of an object into binary data that can be saved somewhere (like on disk), and then later de-serialized to recreate the object. A typical use for this would be saving the game, where you would want to save the state of all the important game objects, and then at a later date, recreate them.

Often when doing something like saving the game, programmers will write a different function to save and restore the state of every object. If there's a reflection mechanism in place, then this becomes significantly easier, as common code can be used for every object. All the

programmer has to do is tell the reflection system what needs to be saved.

BASE TYPE

To add reflection information to a class, we simply derive that class from a simple base class, CReflectable. Listing 1 shows the definition of this class (the complete code is available at www.gdmag.com). At run time, we want to be able to get a data structure that describes the class, which we can use to implement reflection. We could implement this in a number of ways, but they all boil down to some kind of RTTI. We could have a database of classes and then have either a hard-coded class type as the first class member, or use an RTTI function such as typeid(object).name() to look up the type and then find the entry in the table.

What I do here is use a virtual function, CReflectable::GetReflectInfo(), which each class implements, to return a per-class static object of type CReflectInfo, which contains the class metadata. This is quite a cheap method of associating reflection type information with individual objects, as it allows you to directly access it via object instances, rather than looking up the type in a database. If your object already has virtual functions, then it adds no additional per-instance overhead.

REFLECTING CLASSES

Listing 2 shows how classes are registered in practice. There are two classes CGameObj1 and CGameObj2. These are dummy classes for illustration only. Each resembles a simple game object, having fields such as position, velocity, id, and so forth. CGameObj2 is derived from CGameObj1.

To make a class reflectable in this system, we need to derive it from CReflectable and add the reflection information and a couple of utility functions using the REFLECTION_INFO macro. This adds the declaration of the static CReflectInfo object and an

accessor function, and also defines a RegisterMembers function.

We then have to populate RegisterMembers with calls to the Register function via the REGISTER macro. This adds the member name and type to the static CReflectInfo object m_ref_info and calculates the offset of each data member relative to the start of the object, also storing that in m_ref_info.

INITIALIZATION PHASE

The final step in setting up our reflection information is to call each class's RegisterMembers function, which we do via ReflectAllObjects (see Listing 3). Although RegisterMembers is updating a static member object (m_ref_info), it's not a static function itself because it needs access to the this pointer to perform the pointer arithmetic to get the offsets of the data members. We could implement this by instantiating an instance of the object, calling RegisterMembers, and then deleting it. But that's rather untidy, and could have several side effects. Instead, we use the templated function InitReflect to cast a NULL pointer to the correct type and use that as the calling object. Since it is only used for limited pointer arithmetic, it does not actually cause any problems, but this may vary by platform and compilers.

So that's our setup phase. All we need to do now is call ReflectAllObjects during setup, and we're ready to use the reflection.

ACCESSING MEMBERS

In Listing 1, the definition of CReflectable contains two functions: RGet and RSet. Their implementation is shown in Listing 4. These functions are intended to mirror the more typical accessors Get and Set that are often implemented as a per-member instances, such as x=GetID(), SetID(int x), SetVelocity(Vector3 v)and so on. Instead of (or, as well as) all these accessors, we now just have the two functions, RGet and RSet, that can be

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**.

used as accessors or mutators for any reflected data member. Thus:

p_obj1->GetID()

and

p_obj1->RGet<int>("m_id")

are functionally equivalent, except that the latter takes about a hundred times as long to execute and can be called on any object derived from CReflectable.

But speed is not the issue here. What we have now is a way of accessing data members without needing to know anything about an object other than that it is derived from CReflectable. This allows us to do things like implement very straightforward binding with a scripting language. It also allows us to write custom debugging tools that can inspect and modify arbitrary objects. And it lets us loosely bind code modules at a very high level, opening up some possibility of reloadable modules.

DUMPING DATA

A common use for reflection is automated serialization. Serialization is the writing out of an object into a binary format than can later be used to reconstruct the object. It's used for saving data to disk and for sending data over a network. With our reflection system, dumping out the contents of an object is fairly trivial. See Listing 5.

DumpMembers is a member function of CReflectable, but you could quite easily do something similar as a utility function. All it does is iterate over the list of reflected data members and outputs (using printf for clarity) an ASCII representation of each member in a simple CSV format. This could very easily be used as the basis

LISTING 1

```
#define    MAX_REF 32
#define MAX_ID_LEN 128
enum        EReflectionType
{
                    REF_INT,
                    REF_VECTOR3,
                    REF_FLOAT
};
class CReflection
{
public:
        char                        m_name[MAX_ID_LEN];
        EReflectionType m_type;
        int                          m_offset;
};
class    CReflectInfo
{
public:
        void Register(EReflectionType type, const char* p_name, int offset);
        int        m_num_refs;
        CReflection m_ref_table[MAX_REF];
};
void      CReflectInfo::Register(EReflectionType type, const char* p_name, int offset)
{
        strncpy(m_ref_table[m_num_refs].m_name,p_name,MAX_ID_LEN);
        m_ref_table[m_num_refs].m_offset = offset;
        m_ref_table[m_num_refs++].m_type = type;
        printf ("Register(%d, %s, %d)\n",type, p_name, offset);
}

// Base type for all object that are to be reflected
class      CReflectable
{
public:
        void      RegisterMembers(CReflectInfo* p_info) {};
        void * GetMemberPointer(const char *p_member_name);
        template <typename T>
        T                    RGet(const char *p_member_name);
        template <typename T>
        void      RSet(const char *p_member_name,T value);
        void      DumpMembers();
        virtual CReflectInfo * GetReflectInfo()=0;
};
```

Classes to store the reflection metadata are shown.

LISTING 2

```
#define REFLECTION_INFO  static CReflectInfo m_ref_info; \
            virtual CReflectInfo * GetReflectInfo() {return &m_ref_info;} \
            void RegisterMembers(CReflectInfo* p_info);

#define REGISTER(TYPE,MEMBER)        p_info->Register(TYPE,#MEMBER,(int)((long)&MEMBER-(long)this))

class CGameObj1 : public CReflectable
{
public:
        REFLECTION_INFO;
        int GetID() {return m_id;}
        int                        m_id;
        Vector3            m_position;
        Vector3            m_velocity;
        float              m_power;
};


inline void CGameObj1::RegisterMembers(CReflectInfo* p_info)
{

        REGISTER(REF_INT,m_id);
        REGISTER(REF_VECTOR3,m_position);
        REGISTER(REF_VECTOR3,m_velocity);
        REGISTER(REF_FLOAT,m_power);
}

class CGameobj1 : public CGameObj1
{
public:
        REFLECTION_INFO;
        int                          m_count;
};

inline void CGameobj1::RegisterMembers(CReflectInfo* p_info)
{
        // Register parent class´s members
        CGameObj1::RegisterMembers(p_info);
        // Then this object´s
        REGISTER(REF_INT,m_count);
}
```

Two data types, showing how reflection info is added and the utility functions to register them are shown.

for a serialization (although you are more likely to output in a binary format).

A major advantage of implementing serialization via reflection is that once the reflection is set up, everything is automatic. It also makes it trivial to add the ability to deserialize (load) old data even after the data structure has changed, with new members added or old members removed. The deserialization code should be able to transparently handle this. You could even add simple code to support the renaming of data members and still maintain binary compatibility.

This ability to inspect the makeup of arbitrary objects is also useful for custom debugging tools. I mentioned how the accessors allowed you set and get data for debugging. But being able to iterate over the whole list of members of an arbitrary object allows you to write general purpose code that can expose all those members to some editing or debugging control object, allowing you, for example, to click on any game object, view all its members, and modify them or attach logging traces to them. This can be particularly useful when your game objects are composed of an aggregate of components. You simply reflect each component, and the same debugging tools become available for every component.

FINAL REFLECTIONS

This is just one way of implementing reflection. It's simple and relatively easy to understand, but it's also problematic in several ways. It's rather inefficient, both in terms of memory usage and speed. It also uses fixed sized structures with arbitrary limits on the length of the member name and on the number of members that can be reflected. All these issues can be addressed in several ways.

The creation of the `CReflectionInfo` structure for each class is essentially a compile time process, and we can in fact create simple versions of these structures at compile time. Given that an offset of a member m in class T is

`(int)&((T*)0)->m`

this can be compiled into a constant. But the run-time creation gives us more flexibility, for example in handling derived classes, and is more easily extensible.

There are other ways of implementing similar results using templates or additional custom-build steps to parse the source to extract the reflection information. Hopefully this simple implementation of reflection will clarify exactly what reflection is and how it can be useful in game development. It's a relatively simple technique to implement but can be very useful for debugging, encapsulating, modularizing, and creating content. ✖

LISTING 3

```
template <class T>
void InitReflect() { ((T*)NULL)->RegisterMembers(&T::m_ref_info); }

CReflectInfo CGameObj1::m_ref_info;
CReflectInfo CGameobj1::m_ref_info;

void        ReflectAllObjects()
{
    InitReflect<CGameObj1>();
    InitReflect<CGameobj1>();
}
```

Initialize the reflection information for each class.

RESOURCES

Filion, Dominic. "Using Templates for Reflection in C++," in *Game Programming Gems 5*. Boston: Charles River Media.

Vollmann, Detlef. *Metaclasses and Reflection in C++*, 2000. www.vollmann.com/en/pubs/meta/meta/meta.html

LISTING 4

```
void * CReflectable::GetMemberPointer(const char *p_member_name)
{
        CReflectInfo *p_info = GetReflectInfo();
        CReflection *p_ref = p_info->m_ref_table;
        for (int i=0;i<p_info->m_num_refs;i++) {
                if (strcmp(p_ref[i].m_name,p_member_name)==0) {
                        return (void*)((int)this +
(int)p_ref[i].m_offset);
                }
        }
        return NULL;
}

template <typename T>
T CReflectable::RGet(const char *p_member_name)
{
        return *(T*)GetMemberPointer(p_member_name);
}

template <typename T>
void CReflectable::RSet(const char *p_member_name, T value)
{
        *(T*)GetMemberPointer(p_member_name) = value;
}
```

Reflected access and modification of object member by name is shown.

LISTING 5

```
void CReflectable::DumpMembers()
{
        CReflectInfo *p_info = GetReflectInfo();
        CReflection  *p_ref  = p_info->m_ref_table;
        for (int i=0;i<p_info->m_num_refs;i++) {
                void *p = (void*)((int)this +
(int)p_ref[i].m_offset);
                printf ("%d,%s,",p_ref[i].m_type,p_ref[i].m_name);
                switch (p_ref[i].m_type) {
                case REF_INT:
                        printf("%d",*(int*)p);
                        break;
                case REF_VECTOR3:
printf("%f,%f,%f",(*(Vector3*)p).x,(*(Vector3*)p).y,(*(Vector3*)p).z);
                        break;
                case REF_FLOAT:
                        printf("%f",*(float*)p);
                        break;
                }
                printf("\n");
        }
}
```

Dump the members of a structure, the basis of a serialization function.

STEVE THEODORE

PIXEL PUSHER

REVELING IN REACTION

Are gut responses to games more informed than we know?

WITH THE HOLIDAYS UPON US, ALL THE merry olde game industry traditions are back again. The snow is on the ground and the holiday releases are on the shelves, so we gather around the welcoming glow of a cube-mate's monitor and ring in the season of good will by firing up the latest holiday goodie from EB or Gamestop—and, in the immemorial industry tradition, merrily ripping it to shreds. "Will you look at that lousy UV mapping?" "Check out that animation—dude looks like he needs more fiber in his diet!" "Oh boy, somebody went ZBrush crazy!"

The ritual criticism of the latest fresh-from-shrinkwrap disc is an interesting study in the game industry's mentality. You've got a range of attitudes to strike. You might be the thoughtful chin-tugging type who can speculate eruditely about the nicely written anisotropic shader that made that particular sheen. You could be the one who insists that nothing ever matches up to your favorite obscure Japanese-only Super Famicom title from 1992. Maybe you're a hardcore board warrior who can deliver internet-quality flames in real time, to the amusement of all your peers. Whatever your critical style, rampaging over the work of your hapless peers is one of the things that says you're a real pro.

Since it is the holiday season, we should probably pause for a few moments of guilt for the way we run roughshod over our competitors' titles. We all know (far too well) what an astonishing amount of human effort is required to make a game. Even a mediocre game represents a substantial investment of a lot of people's energies and talents. Dismissing it in 30 seconds does seem like a shame, or at least a dereliction of duty. You might hope we'd review the work of our competitors slowly, looking at it in detail, gradually building up a critical opinion like literary

> ## His basic hypothesis in the book is simple: snap judgements work.

critics poring over a complicated text. But in reality, we react like fans crowding around an E3 demo booth (back when E3 still had fans and booths).

SPLIT-SECOND THINKING
Is it unfair of us to jump to judgment so quickly? Perhaps not as much as you might think—or at least that's what Malcolm Gladwell would say. Gladwell, known best for his essays in *The New Yorker*, looked at the nature of snap judgments in his book *Blink: The Power of Thinking Without Thinking* (Little, Brown and Company, 2005).

His basic hypothesis in the book is simple: snap judgments work. Gladwell believes that people solve complicated problems holistically, that is, they comprehend many different factors at the same time, rather than carefully piecing their opinions together out of a mountain of evidence. To use a game-geek analogy, he thinks the brain works like a modern graphics card, processing lots of simple operations in parallel to form a picture quickly, rather than carefully calculating each element of the situation in turn.

The result, he argues, isn't just faster judgment—it's actually more likely to be accurate as well. He cites many examples, both anecdotal and empirical, of people solving hard problems more effectively with less time and less information rather than more. For example, emergency room doctors at a Chicago hospital tried to improve their diagnosis of heart attacks with more sophisticated equipment and better records. The extra information led to arguments, second guesses, and clinical mistakes. However a newer diagnostic test that considered only four simple factors improved the process by more than 70 percent, saving many lives.

This condition may seem paradoxical, but many game veterans would say the same thing about their own jobs. It's easy to find folks who believe that the tight time frame and limited options that come with crunch time make them work faster and smarter than they do during the leisurely hours of preproduction. Of course, plenty of game vets will also say that crunch time destroys their health and home lives, but we know they're just a bunch of whiners.

If you believe Gladwell's theory—and to be fair, it's hardly hard science—there is a lot of power in the global processing he calls rapid cognition. It certainly seems counterintuitive that going with your gut reaction would be a more sophisticated way of evaluating the world than a carefully thought out investigation. However, it may make more sense to

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**.

artists than to many others. Watching a piece magically come together thanks to a single inspired tweak—or one refusing to come together despite having no obvious flaws—is an education in the power of rapid cognition.

WORKING ON INSTINCT

Gladwell's theory offers an interesting slant on one of the great mysteries of the professional artist's life. When you're blindly groping for the missing "something" that will bring your work to life, the process can certainly feel like a kind of mystical quest. Since it's so hard to summon that magical ingredient by willpower or planning, we often say we work by instinct or inspiration.

Unfortunately, words like "inspiration" encourage the tendency of engineers and producers to think of artists as temperamental prima donnas or new age searchers. *Blink*, however, argues that this is actually undervaluing what we do. Gladwell believes there's less magic and more experience in those "instinctive" decisions than we realize.

He cites an experiment in which psychologists asked test subjects to play a rigged card game. One of the decks in the game offered big wins, but even bigger losses; the other provided frequent small rewards. If a subject knew what all the cards were, he or she would always choose from the safer deck since the other was a net loser. It took about 50 hands of the game for most players to figure this out, an ordinary example of learning from experience. The interesting twist is that polygraph tests showed the subjects reacting negatively to the loser deck very early in the game, in some cases in only 10 hands, long before the players could produce a theory about the odds in the game when asked. In other words, as the book puts it, the subjects "figured the game out before they figured the game out." And they did it by learning from experience in the ordinary way, but they reached the right conclusion emotionally before they could articulate it verbally.

If this is the right way to understand "instinctive" choices, it means we should think twice about some of the ways we

work. If gut choices are really a very fast, very smart application of things we've learned (rather than mysteries of the tortured artistic soul) then we need to pay much more attention to the way we work and the way we talk about work. That rapid-fire game dissection doesn't seem quite as crassly insensitive when you think that those instantaneous insights are summations of professional

> ## " Quick reactions include a lot of the accumulated wisdom we slowly and painfully pick up in the course our jobs. "

experience, not mere hip-shots. Every artist knows how important it is be able to listen to your own inner muse, but if you buy Gladwell's argument, that becomes an absolute necessity.

WILL THE DEFENSE PLEASE RISE?

There's an obvious objection to all this, of course. We've all known people with strong instincts who were also dead wrong about a lot of things (though of course, gentle readers, we're not thinking of you). Rapid cognition can get you into a lot of trouble, when it comes in the form of prejudice, stubbornness, or willful ignorance. More insidious, though, are the biases that creep in when we're called upon to justify our blink-of-an-eye judgments. Because there's such a gap between the lightning fast, holistic way in which rapid cognition sums things up and the plodding, step-by-step way we're accustomed to, it's easy to make mistakes about your own intuitions.

Gladwell argues that we often jump to the right conclusion, but when asked to support or defend that conclusion, we flail and grab onto anything that sounds convincing at the moment. Most working artists have been in critique sessions where good feedback gets randomized by efforts to find a "reason" for a very complicated overall mood or impression.

Most of us have at one time or another talked ourselves out of initial reactions when we can't find language to defend them, only to find much later that those initial reactions were basically right. And far too often we've seen good work get devalued, or lesser work given pride of place, because a plausible theory was allowed to run roughshod over the art team's instincts.

Unfortunately, there's no magic formula for disentangling good rapid cognition from prejudice or mental laziness, so you can't simply say, "Go with the gut" and leave it at that. The big lesson of *Blink* is not to follow every impulse blindly or dismiss all efforts at rational analysis. The moral, rather, is that those quick reactions include a lot of the accumulated wisdom we slowly and painfully pick up in the course our jobs. If you believe that, it may be easier to trace those inspirations back to their sources in experience and examples, rather than just dismissing them as subjective tastes.

If there's a basic rule here, it's simply to respect the intuitions and instincts you have developed over the years—but also to remember that your colleagues have their own experiences to draw from, and that pig-headed jerk in the next cube who doesn't appreciate the brilliance of your latest work may actually have a point too, even if it's not being put as diplomatically as you'd like.

In a collaborative art like ours, we don't have the option of being lonely misunderstood geniuses. Our success at teaching and learning from each other is as important to our work as the sharpness of our eyes or the precision of our hands. In that spirit, a quick run through *Blink* over the holidays might be a good present to yourself. ⁝

GAMASUTRA

The Art & Business of Making Games

Don't Play
Games with Your CAREER }

GAMASUTRA IS WHERE TOP COMPANIES LOOK FOR TALENT AND WHERE PROFESSIONALS FIND THEIR NEXT JOBS

### Search Hundreds of New Job Listings Each Month

Gamasutra's job postings comprise the largest game development job board on the web from the industry's top game companies, including EA, Activision, Microsoft, Sony, and Nintendo. Job postings are updated daily—search by company, title, or location.

www.gamasutra.com/jobs

Post Your Resume and Be Discovered

Posting your resume on Gamasutra gets you in front of recruiters from leading companies like Rockstar, THQ, Blizzard, and LucasArts. Post your resume today and let these companies come to you!

www.gamasutra.com/resumes

Winner of
2006 and 2007
Webby
AWARDS
Winner
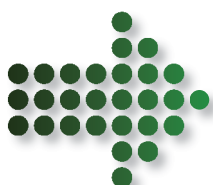
Best of all, these services are FREE } to all registered users.

NOAH FALSTEIN
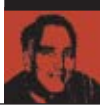
WHEN TALES WAG THE DOG

CONSIDER THE FOLLOWING STATEMENTS. "Games need storytelling to reach a wide audience."

"Storytelling and games are polar opposites, and any game designer who thinks otherwise is just a frustrated novelist or screenwriter."

"Games don't need story, but they need strong, archetypal characters to be really successful."

"Games thrive on narrative, not story—it's all about the player's story of what happened to her while playing the game; the rest is just backstory."

THE STORY OF CONTRADICTION

These are just four fairly contradictory points of view that are expressed frequently whenever game developers gather. And despite the contradictions, there are elements of truth to all of them. The larger truth is that games and their design goals can vary tremendously. Some succeed or fail based on the story included in them, as is the case with adventure games. Others need storytelling as much as the proverbial fish needs a bicycle. TETRIS and some match-three casual games fit that description. Some can make good use of storytelling elements, but are still primarily about the gameplay, like many first-person shooters and real-time strategy games. And some titles focus on the narrative that happens between players after the game, as with THE SIMS.

But it's easy to dodge the issue by saying that the interactive world is big enough to include all types of games. The questions that really should be of interest to game designers are, when do story elements enhance a game, when do they diminish it, and when and how should they be used?

These are big topics, but here are a few rules of thumb about the first two questions at least. I'll return to cover specifics in greater detail in future columns.

WHEN STORYTELLING HELPS

Emotional weight. Story and characters can add emotional weight to interactivity—that's probably the single biggest reason to include a story and strong characters in games. This statement is particularly relevant to games that have goals driven by the motivations of the characters, which helps explain why TETRIS or BEJEWELED don't really need emotional weight.

Rewards and respite. Story provides rewards or welcomed breaks from action, and can add compelling goals to a game. There is a lot of controversy about how best to implement story, but many games, from HALF-LIFE to STARCRAFT to GOD OF WAR, have shown that story progression can help with a game's tempo and engage players on levels beyond the core interactivity.

Entertainment. Storytelling can be an optional layer of entertainment on top of gameplay, so that players who care about the story can revel in it, and the others can hit escape and get back to the interactivity.

Characters as hooks. Strong archetypical characters can provide good emotional hooks without much story depth. Duke Nukem and Solid Snake may not have the depth of a Hamlet or Odysseus, but they're a lot more fun to play than "Generic Avatar No. 4."

WHEN STORYTELLING HURTS

Unnecessary. Some gameplay styles just don't need much story. Abstract puzzle games, sports titles, and embedded mini-games are often fine standing on the merits of their gameplay alone.

Unbefitting of the gameplay. Story elements must be carefully designed to complement interactivity, not clash with it. If a player is forced to sit through long cut scenes (and for some players, long means more than five seconds) that wrest control away at the most exciting points of the game, something is wrong. Seamless blending of story with gameplay is possible, and ruthless editing of non-interactive sequences is essential.

Stories by inexperienced writers. Good storytelling—even just good character creation—is hard and takes practice. A designer with no previous background in writing will probably create some pretty bad stories while learning his craft. So designers must listen to others to learn whether the story is working, as it is much too easy to fall in love with your own story and characters simply because you created them. Ignoring this maxim will often result in stories that actually diminish the game's entertainment value.

Out of character. The story theme and characters must align thematically with the gameplay. Lara Croft is a good fit as a personality for the kind of tasks she tackles in a game. The Prince from KATAMARI DAMACY, charming as he can be, would not be a good tomb raider. The humor of Guybrush Threepwood works great in MONKEY ISLAND, but would be a touch out of place in SILENT HILL.

PLAYING WITH FIRE

Like the old saying about fire, storytelling as an element of game design can be a good servant but a terrible master. It's better to first try it under some experienced or expert supervision, or failing that, start with a small, inconsequential part of a larger game. Do it right, and you'll warm the hearts of your players. Do it wrong, and you (as well as they) will be burned! :•:

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**.

JESSE HARLIN

>> AURAL FIXATION

MUSIC LICENSING 101

IN 1982, MUSIC LICENSING JOURNEYED into the world of video game development with the game JOURNEY ESCAPE for the Atari 2600—the game was a vertical-scroller that contained eight bars of Journey's hit song "Don't Stop Believing." Ever since then, a mutually beneficial relationship has evolved between record labels and game developers. With the rise in pirated music, labels these days are more interested than ever in finding additional revenue streams to help off-set slumping sales. Games can benefit by attracting the interest of a band's already-installed fan base or gain valuable street credibility by steeping a game in the culture of its target audience. Twenty-seven years on from JOURNEY ESCAPE, the success of games such as GUITAR HERO and GRAND THEFT AUTO prove that developers are still devising creative new uses for licensed music.

Knowing where to start and what to ask for can be very helpful at the outset. However, since music licensing essentially boils down to making money off of someone else's creative work, it is a century old topic with a well-established bureaucracy. Any amount of music licensing will inevitably lead to your company's business affairs department and legal contracts of some kind.

KNOW YOUR RIGHTS
All music licensing begins and ends with rights. The licensable rights for any piece of music break down into multiple categories, the rights that are applicable to us here are master use rights and publishing rights. Master use rights are the rights associated with a particular recording of a song. This means that a song as widely covered as The Beatles' "Yesterday" actually has over 3,000 separate sets of master use rights, each one specific to an individual version of the song.

Publishing rights, however, are harder to define. These are the rights associated with the most basic elements of what defines a given work as a unique piece of music. Typically this means its melody and/or lyrics, but only if a song has them. For instance, publisher Edition Peters owns the publishing rights to John Cage's all-silence "4'33"."

Which rights need to be licensed depends on the usage of the song in-game. A development team looking to use the original recording of Led Zeppelin's "Black Dog" for an in-game radio station would need to license both the master use and publishing rights. Meanwhile, a development team looking to re-record the song for an interactive karaoke game would only need to license the song's publishing rights.

Knowing which rights you need to obtain is the easy part. The hard part is tracking down and signing usage agreements with all of the rights holders. The master use rights are owned by the record label or music library company that owns the requested recording. Thankfully, master use rights cannot be divided up between multiple rights holders and are therefore fairly easy to get permission to use (called clearance).

Publishing rights, on the other hand, are divided between anyone involved in the writing of the song. This can mean a single songwriter, each member of a band, or any number of other complicated additional rights holders such as producers or managers. If a song contains samples of other recordings, the publishing rights holders of the original recording need to be tracked down as well. In practice, this can take a long time. One publishing company may only own 15% of the rights, leaving the remaining 85% ambiguous. Unless 100% of the publishing rights can be cleared, the song cannot be licensed. Plus, since asking for clearance on a song costs nothing, it makes sense to try and clear more songs than you might need in the hope that asking for 50 songs might result in a fully cleared list of 30.

KNOW YOUR BUDGET
Once clearance is requested, the rights holders will come back with the licensing fee. Both master use and publishing rights will have separate fees—the larger the artist or the more well-known the song, the larger the licensing fee. On average, licensing pop and rock music by an established artist can cost around $4,000 for the master use rights and $6,000 for the publishing rights per song. Some cost as little as $1,500 for each set of rights. Some can cost up to $10,000 for the publishing rights alone. Some publishers make ridiculous requests such as $250,000 and up for clearance.

Luckily, everything in the world of music licensing is open for negotiation. How a song is being used may affect its cost. If it's going to be featured in trailers and commercials, the cost could be higher over music used as generic background filler. Additionally, some rights holders will lower their fees if they are told what other rights holders are quoting for clearance. Some rights holders might make a special deal if you license music from them in bulk. The negotiation process is long and involves repeated phone calls and emails. Always make sure to plan for plenty of time when trying to license music.

Lastly, many of the big music publishers such as EMI, Sony, and Warner Chappell are embracing streaming Internet technology as a means of helping to streamline the music selection process. Sites like emifilmtv.com, onestoptrax.com, and sonyatv.com put the depth of their licensable catalogues online for quick perusal with filterable search terms, full lyric sheets, digitized audio clips, and contact information for beginning the long process of clearing your soundtrack. ::

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.

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

WWW.ACTIVISION.COM
GO TO "CAREERS"

GameDevelopers®
Conference Mobile 08

February 18-22, 2008
San Francisco

CMP
United Business Media

Save Up to 30%
When you register by January 16, 2008.
www.gdcmobile.com

The definitive game industry event for mobile game developers looking to learn and network within the exploding mobile entertainment sector.

NEW THIS YEAR!

INDEPENDENT GAMES FESTIVAL MOBILE

LEARN MORE AT
WWW.IGFMOBILE.COM

GameDevelopers®
Conference 08

CMP
United Business Media

New Summit Lineup for GDC08:

SERIOUS GAMES SUMMIT

INDEPENDENT GAMES SUMMIT

CASUAL GAMES SUMMIT

GAME OUTSOURCINGSUMMIT
NEW

WORLDSIN MOTIONSUMMIT
NEW

Speaker and session information are now available online.

InsiderTip:
Purchase the **All Access Pass** and gain entry to every summit, every conference session and every GDC sponsored event.

February 18-22, 2008
San Francisco

www.gdconf.com

2008

EELKE FOLMER

THE ACCESSIBILITY GAME

Considerations when creating games for people with disabilities.

GAMES HAVE DISTINGUISHED THEMSELVES from other forms of entertainment since they can provide something books, movies and music cannot: interaction.

Interaction and gameplay are closely related, of course. Gameplay focuses on finding fun ways to interact with a second party, be it AI or another player. Interaction design deals with doing this in such a way that the player can intuit it (usability), or how the player requires it (accessibility). In PONG, if the player uses a joystick, he or she most likely expects to be able to control the paddle by moving the joystick up or down, and not some exotic combination of movements.

If the player is disabled, they might not be able to provide the interaction as expected: what if they can't hold a joystick in their hands? In that case, if we want our game to be accessible to these people, we need to modify the game's interaction to accommodate the player's limitations. Disabilities vary—there are mobility, auditory, visual or cognitive impairments. So how do we need to modify the interaction for each particular disability?

In my research I try to solve some of these kinds of accessibility problems by trying to limit the amount of required interaction and finding alternative ways of interaction for what is required. Here I'll discuss two of them.

MOBILITY DISABILITIES

Players with mobility disabilities, such as quadriplegics, players missing hands or arms—or more commonly, players with repetitive strain injury—all face problems with being able to control the game using conventional control devices, such as a controller or keyboard and mouse. Alternative input devices exist, but they may limit the player in the amount of control and precision available to them.

One of my research questions was to find out whether game interaction can be reduced by automating most of it, while at the same time preserving some fun parts of the interaction. At the University of Nevada Game Engineering Research Group we explored our ideas by developing a first person shooter that can be played using only one button. We created a mod of HALF-LIFE 2's deathmatch mode in which we allowed the player to assist an AI-controlled bot. The player can use the space bar (or any sort of one-button controller) to fire the bot's gun, but all the other interaction is automated; our bot seeks out enemies, aims at them, runs away when the ammo or health gets too low, and finds ammo and health when possible.

We did several tests with our mod and it turns out to be quite fun to play (even for people without disabilities). It does change the gameplay significantly, but we think this is quite an improvement compared to most first person shooters which are more inaccessible to physically disabled players.

Creating an accessible game by automating most of the interaction turns out to be quite easy, since most of the functionality that we need, like pathfinding, is already present in the AI. It's merely a matter of hooking this up to the player.

Care must be taken to balance gameplay, however. The one-button requirement was kind of strict; players with severe disabilities can often provide more interaction, but as long as the one-button version is still playable we can offer more shades of grey in-between.

GAMES FOR THE BLIND

Visual disabilities include blindness, low vision, and color blindness. Developing games for the visually disabled is challenging, since the majority of games rely on visual output. Replacing visual stimuli with either physical stimuli or audio stimuli is another approach we explored in two projects: BLIND HERO and SECOND LIFE.

BLIND HERO. Audio games are extremely well suited for the blind. GUITAR HERO is essentially an audio game, but relies upon visual stimuli, which makes it inaccessible to blind players. We explored creating an accessible version of guitar hero by replacing the visuals. Because music is playing, it's hard to use audio cues, so we used physical stimuli instead. We developed a glove with four little pager engines at the top of the fingers, which start buzzing when you need to press a particular button. Since guitar hero is closed source, we used the open source clone FRETS ON FIRE, which is playable with a normal GUITAR HERO controller. A blind player can navigate the menus of FRETS ON FIRE using speech synthesis. Though it's still a prototype, we can play relatively simple songs without visual stimuli, though we need a rather expensive glove to do so.

SECOND LIFE. SECOND LIFE is a virtual environment, but does rely upon the ability to see. Our idea was to replace the visual output with text, which can then be hooked up to a screen reader. This basically turns SECOND LIFE into an audio-based adventure game. This project is technically a lot more challenging, since we need to be able to extract 'meaning' from the SECOND LIFE environment. Fortunately, objects in SECOND LIFE have names and can be recognized relatively easily. We have already achieved remarkable results with our current prototype, which we hope to make available to the blind community very soon.

All our projects have given us a better understanding into what makes games accessible, and we hope game developers may adopt some of the technologies we've developed (and will develop in the future) to make their games more accessible. ::

WEB RESOURCES
www.helpyouplay.com/blindhero.html
www.helpyouplay.com/gtf.html
www.igda.org/wiki/Game_
Accessibility_SIG

EELKE FOLMER is an assistant Professor at the University of Nevada's Game Engineering Research Group. Email him at efolmer@gdmag.com.