



gd

GAME DEVELOPER MAGAZINE



sid meier's civilization v

SEAPINE AGILE EXPEDITION™

eXplore Agile in an eBook Adventure

The Seapine Agile Expedition is a **free** eBook-based learning series that provides a fun and informative overview of **Agile development**.

New to Agile? Discover how Agile boosts the performance of your entire team—developers, quality assurance, product owners, and ultimately, your customers—through timely communication and collaboration.

Already using Agile? Learn how to meet deadlines, respond to change, reduce risk, and deliver what you promise through Agile development powered by Seapine ALM.

Join the Adventure Today!



www.seapine.com/gamebook

gd

GAME DEVELOPER MAGAZINE

GAME DEVELOPER MAGAZINE

CONTENTS.0311

VOLUME 18 NUMBER 3

POSTMORTEM

13 HOUSEMARQUE'S DEAD NATION

Marrying a twin stick shooter with the Zombie genre was a perfect match. So much so that Housemarque's enthusiasm for the project resulted in ever increasing feature creep. What was supposed to be a small-scale downloadable title was starting to look like a full-on retail game.

By Harri Tikkanen and Ilari Kuitinen

20 FIRAXIS' SID MEIER'S CIVILIZATION V

When the team began work on CIVILIZATION V the technology for the new game was not yet in place. This forced them to run a development plan that put design and engineering on separate but simultaneous tracks. It wasn't until the final year of CIVILIZATION V's almost four year production that the two came together to create a new CIV game that was as pretty to look at as it was forward-thinking in its design.

By Dennis Shirk

FEATURES

7 TENSION MAPS

Ideally, by the time a game ships, it's various systems should be in a state of careful balance. Of course, during development, systems will need to be cut or modified and the closer a game is to its ship date the riskier those changes can be. Here, designer Simon Strange presents a decision making approach for identifying and defining low-risk design changes.

By Simon Strange

26 RED MEANS BOOM!

Putting exploding barrels in your game should be pretty simple, right? Well, not quite, as this anecdote from behind the scenes of BULLETSTORM illustrates.

By Arcade Berg

DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]
Surprise! Your Game Is Canned
- 4 **HEADS UP DISPLAY** [NEWS]
Cowlickification, Japanese game market woes, and Nathan Fouts' Developer Notebook.
- 28 **TOOL BOX** *By Seth Gibson and Damian Kastbauer* [REVIEW]
Autodesk Maya 2011 and BaseHead 2.5
- 32 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]
Water Finds a Crack
- 35 **THE INNER PRODUCT** *By Dave Cowling* [PROGRAMMING]
Idle Threads
- 39 **PIXEL PUSHER** *By Steve Theodore* [ART]
Blink of an Eye
- 43 **AURAL FIXATION** *By Jesse Harlin* [SOUND]
Three Erroneous Concepts
- 44 **THE BUSINESS** *By Kim Pallister* [BUSINESS]
A War of Attrition
- 46 **GOOD JOB!** *By Brandon Sheffield* [CAREER]
Manveer Heir Q&A, Who Went Where, and New Studios.
- 48 **EYE ON GDC** [GDC]
2011 GDC Classic Game Postmortem Lineup
- 50 **EDUCATED PLAY** *By Jeffrey Fleming* [EDUCATION]
Richard Flanagan's FRACT
- 56 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]
To The Writer



SURPRISE! YOUR GAME IS CANNED

SO YOUR BIG GAME IS CANCELLED. WHAT THE HECK DO YOU DO NOW?

UNTIL LAST WEEK, I WAS narrative director for a mid-tier budget, big publisher-backed original IP for retail Xbox 360 and PlayStation 3. Then, suddenly, one executive shuffle later, projects were getting canned left and right, mine among them. It's happened to nearly everyone in the industry, but it's the first time for me, so I'll delve into this a little bit.

WELL, THAT SUCKS!

» Nobody saw it coming—the game's alpha was approved and paid for, response from the publisher was positive, and all our ideas were really coming together. The story made sense and had a flow to it. All gameplay elements existed in the alpha, and our unique look was showing through. Five days after that, word came down that it was over. Ultimately, it came down to publisher finances, there was nothing anyone could do about it, and frankly I don't even blame the publisher. Still, one can't help but think—what if we'd put a little more effort in? What if we'd worked longer hours, or made our ideas more obvious in the alpha build? Would it have helped? Doubts and misgivings are difficult to avoid.

The game took a lot of calculated risks, which for NDA reasons I obviously can't divulge—did those risks keep us afloat longer, or did they kill us prematurely? Will I get to see these risks taken eventually in some other

game, and burn at my missed opportunity? These questions are unanswerable, but always lurking in the back of my mind.

Luckily, nobody's losing their job over this—the team absorbed all its core members into other projects, and life goes on. As an external contractor though, my journey with this team has ended. It was a good group we had, with solid ideas, crazy work ethic, and a willingness to take a stab at something new. I suppose everyone feels that way about their teams, don't they?

WHAT NOW?

» When you've put months of your life and much of your brainpower into one project, it's difficult to derail. I keep thinking of things I could do to make the story better, the characters more believable, the subtext more subversive. Since this does no particular good for anybody, I've got to redirect my thoughts to something else. And that's one of the silver linings to a project cancellation—if you do have other ideas, a canceled game is obviously a huge disappointment, but it's also a weight lifted. To make this magazine run and also be narrative director on a decent-sized game, I made myself work some crazy hours. Essentially, I never stopped. That may be a bad example to set in an industry plagued

by crunch, but I wasn't actually crunching in either job per se—I was crunching in life, by doing two full-time jobs simultaneously.

Suddenly, I find myself with reams of free time. No weekly meetings, no deliverables, no misunderstandings to clear up, and no critical issues to think through. But the energy remains. I still need to "do," and to "create."

I've talked a lot in these pages about the merits of smaller-scale teams and development, so I'm going to put my money where my mouth is, and work on a few iPhone and XBLIG projects I've been considering for some time. There's something to be said for working at your own pace, on a project that's very much your own, even if the financial rewards are uncertain.

TALENT BLEED?

» I outlined all this so explicitly because I think my experience is not unique. When a project you're passionate about slips away from you, the passion doesn't go away. The need to create remains, and must be channeled somewhere, or be lost. I think this experience is what turns a lot of modern-day developers away from the traditional publisher-backed game industry. This is good for players, because they get more varied experiences across more platforms—but it's bad for traditional game developers and publishers, because they

may lose some of the people that will fight for risk and new experiences in traditional games.

I'm certainly not saying I won't work in a traditional game development setting again—I most likely will, when the opportunity presents itself. There are certain kinds of narrative experiences that are difficult to put forward with a small team, and those ideas still intrigue me. But there are persons for whom rest and repose hold only limited appeal—I think many game developers feel this way, and it's what pushes us to keep creating and make the games we make. For all of you out there who may have recently lost your job, had your game canceled, been scaled to half-time, or anything of that nature, I urge you to find out what you really want to express, and find a way to make it happen for yourself, even if that's not within games.

Rather than looking at this as a time to mope and feel sorry for oneself, the best thing to do is to work on those smaller or weirder games you've had floating around in your head, write that screenplay, code that application, or finish that novel. That's my challenge to myself, and to those of you in similar situations, I pose the same challenge. Let's see what we can come up with before we get back to the big grind, if indeed we ever do. ☺

—Brandon Sheffield
twitter: @necrosotoy

United Business Media
303 Second Street, Suite 900, South Tower
San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090

SUBSCRIPTION SERVICES

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

FOR DIGITAL SUBSCRIPTION INFORMATION
www.gdmag.com/digital

EDITORIAL

- PUBLISHER**
Simon Carless | scarless@gdmag.com
- EDITOR-IN-CHIEF**
Brandon Sheffield | bsheffield@gdmag.com
- PRODUCTION EDITOR**
Jeffrey Fleming | jffleming@gdmag.com
- ART DIRECTOR**
Joseph Mitch | jmitch@gdmag.com
- PRODUCTION INTERN**
Tom Curtis
- CONTRIBUTING EDITORS**
Jesse Harlin
Steve Theodore
Kim Pallister
Dave Cowling
Soren Johnson
Damion Schubert
- ADVISORY BOARD**
Hal Barwood Designer-at-Large
Mick West Independent
Brad Bulkley Neversoft
Clinton Keith Independent
Brenda Brathwaite Lolapps
Bijan Forutanpour Sony Online Entertainment
Mark DeLoura THQ
Carey Chico Independent

ADVERTISING SALES

- GLOBAL SALES DIRECTOR**
Aaron Murawski | amurawski@think-services.com
t: 415.947.6227
- MEDIA ACCOUNT MANAGER**
John Malik Watson | jmwatson@think-services.com
t: 415.947.6224
- GLOBAL ACCOUNT MANAGER, RECRUITMENT**
Gina Gross | ggross@think-services.com
t: 415.947.6241
- GLOBAL ACCOUNT MANAGER, EDUCATION**
Rafael Vallin | rvallin@think-services.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER
Pete C. Scibilia | peter.scibilia@ubm.com
t: 516-562-5134

REPRINTS

WRIGHT'S MEDIA
Ryan Pratt | rpratt@wrightsreprints.com
t: 877.652.5295

AUDIENCE DEVELOPMENT

TYSON ASSOCIATES Elaine Tyson
e: elaine@tysonassociates.com

LIST RENTAL Merit Direct LLC
t: 914.368.1000



around the Arab world
there are more than
180 million people
under the age of 25*.



yet Arabic game development
is still in its infancy.

your opportunity is right here, right now at twofour54° in the heart of Abu Dhabi.

The Arab world is one of the world's fastest growing media markets. With a young population of 180 million under the age of 25, more than 80% with mobile phones*, strong broadband take-up and new gaming innovations, it's a prime opportunity for Arabic gaming businesses.

We empower businesses across all media platforms from production, gaming, digital, animation, broadcast and publishing – with world-class training from **twofour54° tadreeb**, state-of-the-art production facilities with **twofour54° intaj** and venture funding and support for Arab creative entrepreneurs from **twofour54° ibtikar** – to seize every media opportunity the region has to offer.

It's all part of our vision at **twofour54°**, creating a centre of excellence for Arabic content creation in Abu Dhabi.

we are twofour54°. are you?

find us. join us. create with us.

+971 2 401 2454 **twofour54.com**

twofour54°
Abu Dhabi

content creation community



cowclickification rising

Building upon his popular social game satire Cow Clicker, Ian Bogost has launched a Cow Clicker platform offering a programmable API, a new Facebook game, an iPhone app, a Google search engine, and more.

Initially designed as a parody of shallow social games and launched last July, COW CLICKER "challenges" users to click a virtual cow every six hours to earn clicks and publish their accomplishment onto their Facebook feed. Players can then spend those clicks to buy custom premium cows and in-game items.

The game has since taken off and convinced more than 50,000 people to click over 50 breeds of cows at least 5 million times.

Game designer, professor, and author

Bogost says he's realized through this success that people want "as many opportunities as possible to click a cow every six hours."

Thus he's expanding the COW CLICKER Platform to include a COW CLICKER Connect web-widget that allows content publishers to "cowclickify" their properties by integrating clickable bovines into their sites, and a programmable COW CLICKER API allowing developers to build their own cow clicker applications.

"It's called cowclickification—the application of cow clicking mechanics to non-cow clicking applications," Bogost explains. "Now everyone can graze on the sweet grasses and step in the pungent pies of COW CLICKER's pasture. Businesses can



COW CLICKER BLITZ

employ new cow clicking mechanics such as clicking a cow to distract customers from the vapid pointlessness of their

products and services."

To show the COW CLICKER API in action, the developer announced a new COW CLICKER BLITZ

puzzle game on Facebook, a COW CLICKER MOOBILE iPhone app (Mac App Store edition forthcoming), and a Google search engine enabling users to search the web by clicking on their own virtual cow.

He announced that companies will be able to "moonetize" their cowclickified application with an upcoming app store called The Stockyard, which will provide a hub "where developers and businesses can publish their cowclickified applications, and where eager cow clickers can find new opportunities to click."

Bogost adds, "Moove over gamification, it's time for gamooification. The COW CLICKER platform is here to turn your marketing and branding initiatives into a thunderous stampede of clicks!" —Eric Caoili

japanese retail falling

A report from Japanese Famitsu publisher and research firm Enterbrain found Japan's retail video game market declined 9 percent in 2010 to ¥493.66 billion [\$5.94 billion], down from ¥542.64 billion

(\$6.53 billion) in 2009, according to a translation on Andriasang.com.

The drop marks the third year in a row of Japanese video game retail decline. Falling hardware revenues drove the 2010 decline, as

the category fell 18.9 percent year-on-year to ¥175.59 billion [\$2.11 billion], according to the report. Software sales dropped 2.5 percent to ¥318.08 billion [\$3.83 billion].

Nintendo's DS line of handhelds led game-specific hardware sales with nearly 3 million units sold, followed closely by Sony's PSP line with 2.9 million units.

The Nintendo Wii led home console unit sales with 11.2 million units, followed by PlayStation 3 (1.6 million) and Xbox 360 (209,000).

Portable games topped the software charts, as Pokemon Co's September 2010 release POKEMON BLACK & WHITE for DS sold 4.9 million units during the year, and Capcom's PSP game, MONSTER HUNTER PORTABLE 3RD, sold 3.5 million units after just being released on December 1.

While Japanese video game retail revenues continue to shrink, Japanese consumers seem to be spending more money in emerging game markets such as mobile and social gaming, and using devices such as PCs, phones and

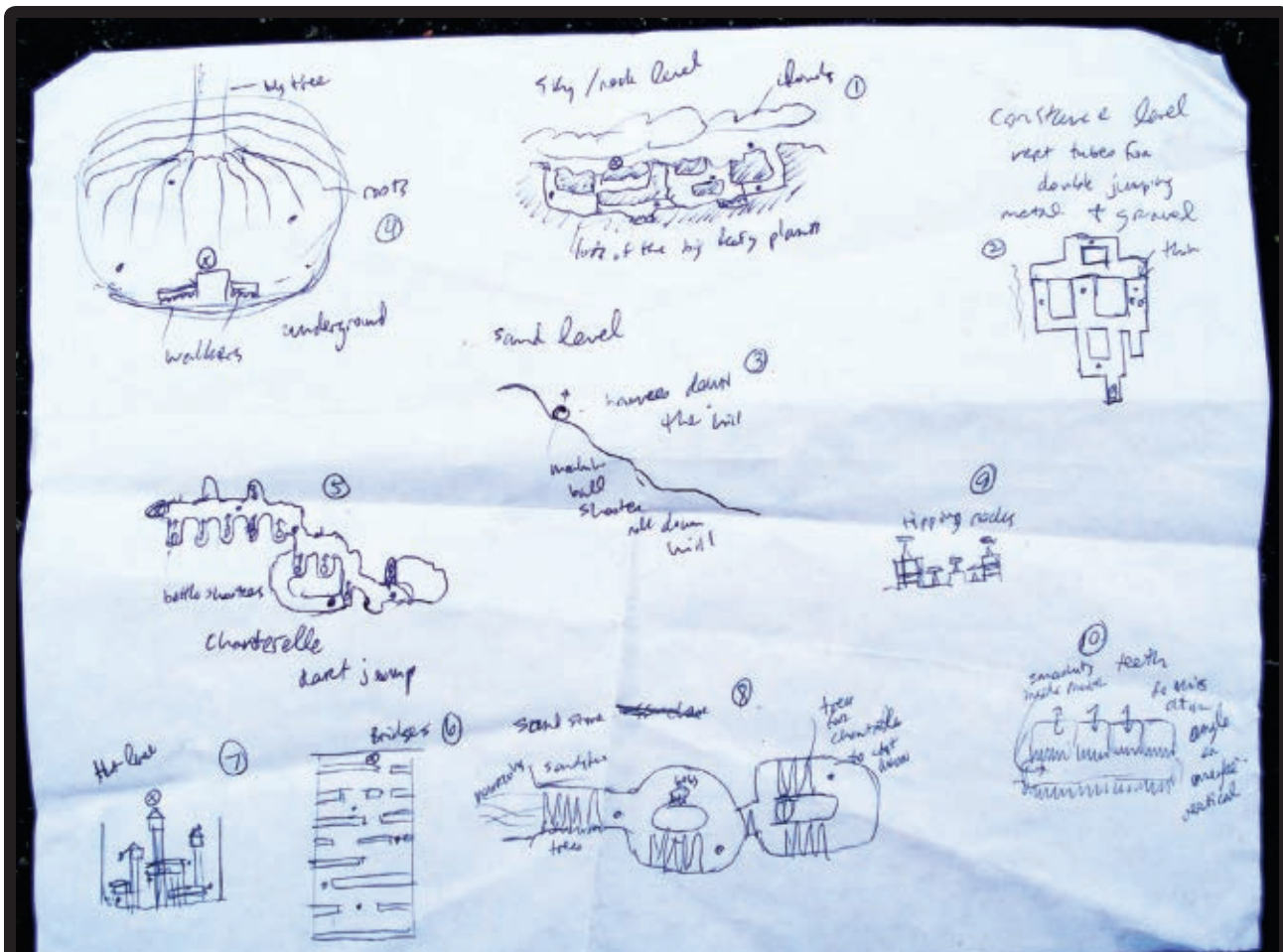
smartphones for gaming, much as in the West.

In November last year, Japanese mobile social network operator and game creator DeNA reported a 216 percent year-over-year jump in its second quarter sales to \$336.4 million.

The sales marked rapid growth for the firm, which acquired U.S.-based mobile game company Ngmoco for up to \$403 million last year. DeNA's mobile social network, Mobage-town (MobaMingle in the West), reportedly has over 20.5 million registered users.

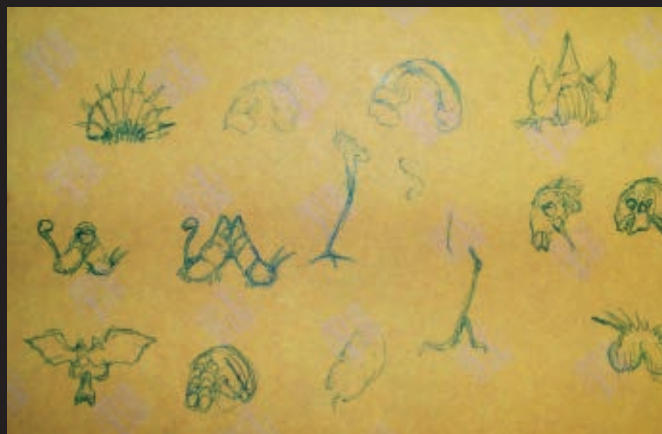
—Kris Graft





designer's notebook: nathan fouts

As the artist, designer, and primary coder behind Mommy's Best Games, Nathan Fouts has the freedom to create an index of visual obsessions that both disturbs and delights in equal measure. As seen in titles like WEAPON OF CHOICE, SHOOT 1UP, EXPLOSIONADE, and the upcoming GRAPPLE BUGGY, his hand-drawn art presents game players with an efflorescent riot of aberrant biology. Here we get a look at some of Fout's early design sketches from WEAPON OF CHOICE and GRAPPLE BUGGY that show an Albertus Seba-like fascination for organic forms.



THE 2011
gametree™ TV
DEVELOPER COMPETITION



WIN GLORY
WIN \$50,000

ENROLL | COMPETE | DOMINATE

GET IN THE GAME at [HTTP://GAMETREETV.COM/COMPETITION](http://GAMETREETV.COM/COMPETITION)



a process for identifying low-risk
design opportunities

tension maps

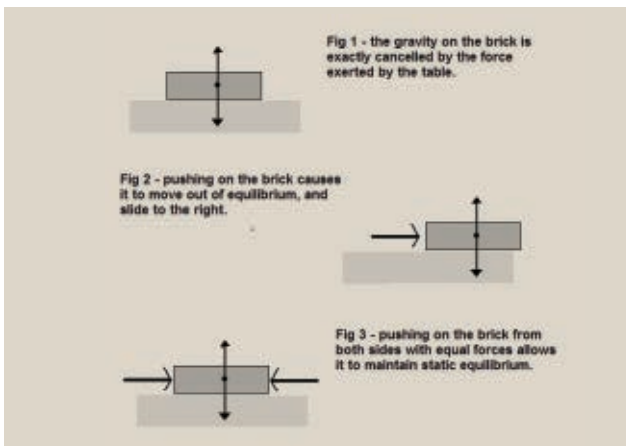
S I M O N S T R A N G E

All systems are fundamentally in one of three states: growth, decay, or equilibrium.

For a video game, which can be viewed as a system of systems, growth and decay both happen during development. As systems are added, removed, and adjusted, the game more and more resembles its final shape. By the time you ship, your game is (hopefully!) at equilibrium.

Of course, designers cannot simply tweak and tune game systems on a whim. The target equilibrium point (the final state of the game's systems) needs to be identified fairly early on, so that individual systems (and their supporting assets) can be locked down during development. This is a very practical way to reduce risk and manage a project. Unfortunately, this tends to create an antagonistic relationship between a designer's ability to effect change and the amount of development time left. This reduces the designer's ability to work on game systems during the latter half of the project, which can be very frustrating.

Over the last few years, I've developed a system for identifying and defining low-risk design changes. My goal is to allow design changes during the majority of a project instead of being forced to lock down design elements early on. By identifying low-risk options in a systematic way, using charts and visual aids (which I discuss in parts 2 and 3), I have been able to describe to producers and publishers in advance exactly why certain changes pose little to no risk to the project's long-term stability. This has afforded me almost twice as much time for fine-tuning our game's core systems, which has resulted in better, and more balanced, more polished products.



PART 1 DEFINING TENSION

]]] The key concept is “equilibrium tension.” A system in equilibrium feels the effect of many sub-systems, but each “pull” is balanced by an inverse “pull” of equal magnitude. In the simplest cases, this means two sub-systems opposing one another’s effects, but in most cases a combination of sub-systems must be considered. This is exactly analogous to the force diagrams you might have drawn in physics classes.

Imagine a brick resting on a table. The gravitational force on the brick is exactly opposed by the table, so the brick remains in motionless equilibrium. [See Figure 1.] Now imagine your left hand on the left side of the brick. If you press on the brick, the brick will slide to the right. [See Figure 2.] If you use both hands, one on either side, and apply an equal amount of force, you can re-establish equilibrium. [See Figure 3.] The point is that the brick can remain in equilibrium with any magnitude or combination of forces, so long as each force is counteracted by other forces.

This does not mean that all equilibrium states are the same! The “squeezed” brick can absolutely feel the tension from your two hands. In the same way, you can make significantly different experiences within a video game by changing the “tension” on that game’s equilibrium state.

Imagine our brick as a playable character in a simple 2D platform game. A player could move the brick left or right, jumping over small obstacles to progress through the world. If the player puts the controller down to take a break, nothing would happen to disturb the brick, just as you might have removed your hands from the physical brick and left it lying on the table.

Now let’s add a sub-system, and start throwing fireballs from the right side of the screen every three seconds. The fireballs are a new force which, if unbalanced, would “push” the brick out of equilibrium. To balance this force, the player must simply “push” back by jumping over the fireballs as they appear. So long as the player jumps properly, the game remains in the same equilibrium as before. The difference is that the player is now actively working to balance the game’s equilibrium. The more we demand of the player, the more tension we place on our equilibrium state.

Let’s look at a few examples of how tension might be increased or decreased in some well-known games, without passing judgment in regard to whether this increase or decrease would be a good thing.

PORTAL

]]] The central system in PORTAL can be boiled down to “Where do I place the blue portal?” and “Where do I place the orange portal?”

Not every surface in PORTAL is smooth enough for the player to open a portal through. Since the possible solutions to each room increase or decrease with the number of possible portal locations, reducing the number of smooth walls makes the solution more readily apparent to the player. So

fewer smooth walls equals lower tension.

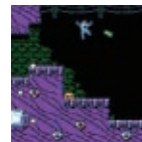
As PORTAL exists now, there is no penalty for creating more portals than necessary. But if we were to track the number of shots used in each room—by limiting the gun’s ammo, or simply telling the player that we expected them to be more prudent (16 shots / 8 expected)—players would become aware of each shot as a resource to be managed. Managing more resources means higher tension.

MEGA MAN

]]] MEGA MAN games have always included difficult jump sequences. In most cases, missing a jump means you have to start the sequence over again. But in a few spots, missing a jump means instant death. Jumps always provide opportunities to progress through the level but jumps over spikes or pits also offer an opportunity to fail the level entirely.

Spikes and pits have absolutely no consequence on a successful jump, as they are simply an extra harsh penalty for failure. Reducing these cases allows players to master the tricky jumps without fear of penalty. This changes the aggregate angle of the jump “tension” and reduces the player’s fear of failure. Less fear equals lower tension.

MEGA MAN games have always been moderately difficult due to tricky jump timing, instant deaths, and limited attack options. But there have never been strict time limits within individual stages. Simply adding a countdown clock would give players yet another way to fail, increasing the tension of every action in the game. Time limits mean higher tension.



ALIENS VS. PREDATOR

]]] Rebellion’s 1999 PC game ALIENS VS. PREDATOR had some wonderfully austere levels, in which only a handful of enemies would appear. The clever twist—nearly unprecedented for a FPS at the time—was that enemies were randomly distributed, and their locations were reset whenever you loaded or re-loaded the level. This meant that while you could learn the geography of the level, you never knew when or from where the enemies would be coming. This was a dramatic departure from the norm, and was a big part of

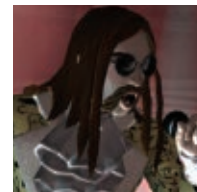


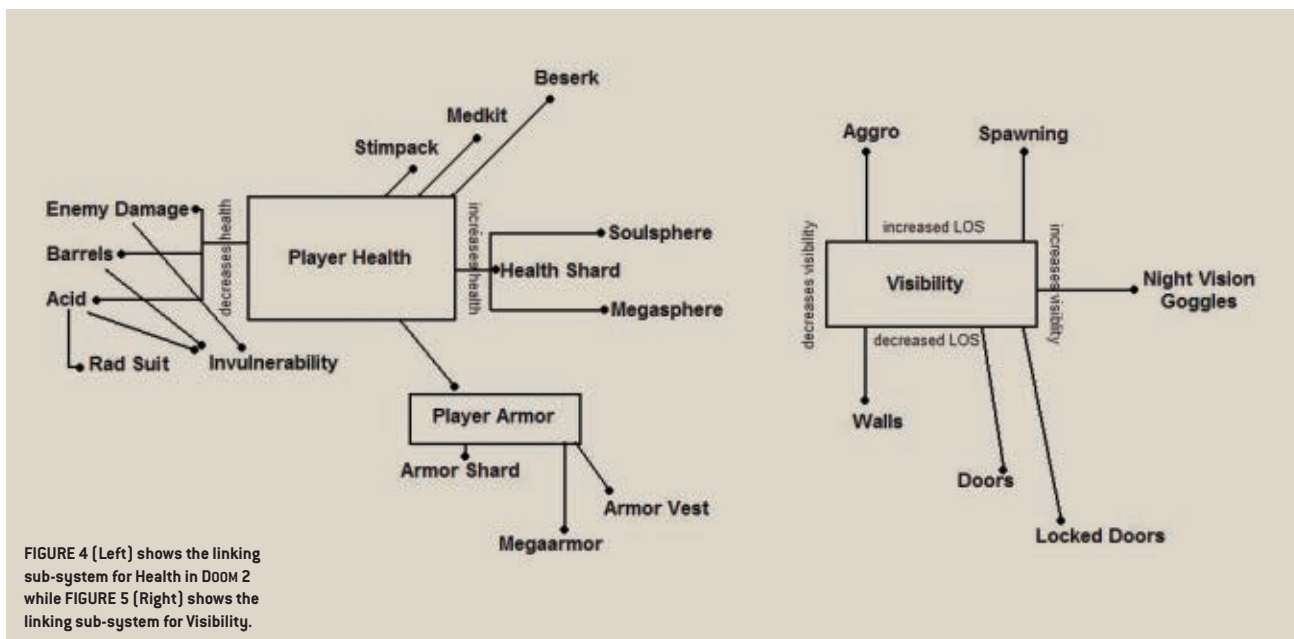
AvP’s reputation as a very scary game. Unpredictability equals higher tension.

Another innovation in AvP was the asymmetrical balance between the three characters. Predators and Marines each had a suite of powerful weapons, but they were all strictly limited by ammunition. This forced the characters to constantly guess which sort of threat they would be facing at any moment, and to switch weapons accordingly. The Alien, on the other hand, had no equipment, ammunition, or other limiting factors at all. Aliens were always working at 100 percent effectiveness. This cognitive simplicity was a big source of the appeal for playing as the Alien. Fewer gameplay choices means lower tension while playing as an alien.

ROCKBAND

]]] ROCKBAND’s core gameplay has never really changed. That’s actually an important principle of the franchise, because they want to sell additional tracks to customers, and breaking backward-compatibility would be a big deal. But the RB games have managed to change in other





significant ways, all of which are excellent examples of maintaining system equilibrium at different levels of tension:

Failure. RB1 forced players to fail if they missed too many notes. RB2 allowed quick games to be played in a special “no fail” mode, but separated those sessions from “real” RB2 sessions. LEGO ROCKBAND weakened the impact of failure by allowing failing players to save themselves. RB3 introduced a menu option which simply turns off failure at no penalty.

Group composition. RB1 forced bands to keep essentially the same members in every session. RB2 allowed players to lead multiple bands, but locked your selected band once you moved past the main menu. RB3 allows a group to seamlessly change bands at any time, and to add or remove players mid-song.

Achievement/Trophy criteria. In order to get all achievements, RB1 required players to master each instrument, expected all players to play at expert level, and required playing with multiple local players. RB2 required players to master each instrument, and even asked players to play for seven hours without pausing or disconnecting controllers. RB3 rewards you for playing in practice mode, gives rewards at every difficulty level, and tracks individual progress in every song (so you can fail or excel independently of your group).

ROCKBAND has clearly been moving toward lower and lower tension states. Some people consider that an improvement, while others are less happy with the change. But Harmonix has done an excellent job of adjusting its games without breaking their core equilibrium, and has done so by managing player tension levels.

The point of all these game-specific examples is not that these design elements necessarily make the games any better or worse. The point is that all of these proposals do nothing to break the equilibrium of the game’s established systems. The game might be more challenging, more frustrating, or less compelling, but it would mechanically be the same game. I’m not trying to evangelize creating games with more or less tension—that question needs to be addressed by each game’s individual design team. I’m simply posing that by thinking in terms of tension and system equilibrium,

we can identify design opportunities which do not threaten the stability of the project but still impact the player’s experience in significant ways.

PART 2 DRAWING TENSION DIAGRAMS

))) I’ve made a point that some system-level changes in your game can be made with minimal risk, and I’ve identified a few examples of such changes. But how, exactly, does one go about separating equilibrium-preserving changes from equilibrium-destroying changes? Without a formal process to repeat these results, it’s meaningless to simply point out examples. To that end, I’ll share my step-by-step process of drawing a Tension Diagram.

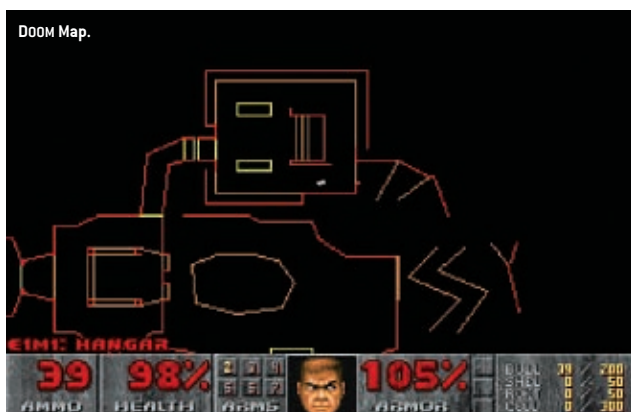
1. Identify as many sub-systems as possible.
2. Identify as many player actions as possible.
3. Link sub-systems together.
4. Link player actions to the sub-systems.
5. Identify missing sub-systems and player actions.
6. Repeat steps 3–6.

For this exercise, I’m going to create a Tension Diagram for DOOM 2. DOOM 2 is pretty primitive by today’s standards, but the limited systems will assist in our completion of the diagram. If you’re already familiar with DOOM 2, you may want to perform steps 1 and 2 on your own without looking at my list. You’re welcome to use online resources and so forth.

DOOM 2’S SUB-SYSTEMS

FPS fundamentals: levels, time counter, visibility, health counter, ammo counters, armor counter, projectiles, item collection, enemy spawning, victory condition(s), difficulty settings, load game, save game, and deathmatch.

Enemies: Soldier, Sergeant, Nazi, Heavy Weapon Dude, Imp, Demon, Specter, Revenant, Mancubus, Arch-vile, Cacodemon, Pain Elemental, Lost Soul, Hell Knight, Baron, Cyberdemon, Arachnotron, and Spiderdemon. Each enemy has the following sub-systems: movement, health, attack, aggro. Because the game supports deathmatch, we add “enemy player” to this list, though enemy players have more sub-systems than NPC enemies.



Weapons: Fist, Berserk Fist, Chainsaw, Pistol, Shotgun, Chaingun, Rocket Launcher, Plasma Gun, and BFG9000.

Environment: Doors, locked doors, moving columns, crushers, acid floors, and exploding barrels.

Collectibles: Health shard, stimpack, medkit, armor shard, armor vest, ammo, soul sphere, megaarmor, megasphere, invulnerability, night goggles, backpack, berserk, invisibility, map, radiation suit, and keycards.

DOOM 2's player actions: Four-way movement, turn, strafe, run, open door, shoot, and change weapon.

Linking sub-systems: Health [see Figure 4]. For no particular reason, I've started with Player Health at the core of my diagram. I listed all the sub-systems which interact with Player Health, and then arranged them flow chart-style around that mechanic. There are four sub-systems which decrease health, and seven which increase health. Of course, most of those systems have more subtle effects, which we also need to represent. In this starting diagram, moving to the right means increasing health while moving to the left means decreasing it.

Three items (health shard, soul sphere, and megasphere) directly increase health, so they lay to the immediate right. Stimpacks, medkits, and berserk only increase health up to 100 percent, so in some cases, they don't increase health as well as the first three items mentioned. To indicate this, those items are represented as mostly pulling against Player Health, but also partially pulling against the reduced maximum health for those items.

Armor does not increase or decrease Player Health, but it does mitigate damage, which I have represented by allowing it to "pull" on the four damage sources. Invulnerability (and to a lesser extent, Rad Suit) play a similar role.

Linking sub-systems: Visibility

(see Figure 5). When I use the term "Visibility," I literally mean what can be seen. There are three sub-systems which manipulate visibility: light, obstacles (such as walls), and aggro. Light affects whether the player can actually see something in front of them, and it is affected only by Goggles. Doors and walls block line of sight as well as prevent enemy aggro. Similarly, unspawned enemies cannot be seen.

So around Visibility, we have two orthogonal groupings of systems. Though they do not interact, they are linked by a common metric.

ADDING PLAYER ACTIONS

))) All of the player's movement options allow us to circumvent obstacles such as walls and doors. Moving eventually leads us to victory as we navigate each level. Moving also allows us to avoid enemy projectiles (see Figure 6).

Shooting can (hopefully!) reduce the enemies around us, at the expense of ammunition. I could complicate the diagram by taking into account how efficiently each weapon combats each type of enemy, but that's not strictly necessary, and would require overlapping lines or a 3D model. Since enemies in DOOM 2 did not have any sort of resistance system, all weapons work reasonably well against most enemies.

ARE WE MISSING ANYTHING?

))) This diagram does not include any systems from the story, theme, menus, or taunts, and it ignores player expectations which might have been built-up from an overexposure to the original DOOM. Those all have an important place, but they generally do not impact the central system mechanics that we are considering here. In fact, you can look at Raven Software's excellent HERETIC to see how a nearly identical system design can spawn a game with an entirely different feel.

PART 3 ISOLATING VARIABLES

))) I've explained my ideas concerning system equilibrium, tension, and



UNREAL TECHNOLOGY NEWS

BY Mark Rein
Epic Games, Inc.



UDK PUTS MORE MONEY IN YOUR POCKET

We demonstrated what is possible with Unreal Engine 3 on mobile devices with the award-winning *Infinity Blade*, which took the iTunes App Store by storm in December. We also released iOS support for the Unreal Development Kit (UDK), which brought the first UDK-powered App, *Chicken Coup*, to the App Store in February.

Developed by Trendy Entertainment, *Chicken Coup* is like a cross between *Angry Birds* and *Flight Control*. Trendy has even released the full source materials for the game so that UDK developers can use it as an example of how to build a puzzle game for iOS.

We recently modified the terms of the UDK license by increasing the revenue threshold for royalty payments. Under the new license agreement, developers don't pay any royalties until their total revenue exceeds \$50,000 (US). Beyond that developers keep 75 percent of each dollar they receive and Epic receives 25 percent.

The first step to taking a UDK game commercial is to pay a \$99 (US) fee at www.udk.com when it's time to start realizing financial benefit; typically, this is when a game is about to go on sale. Prior to that point, you can use UDK for free. This is a one-time license fee for your entire company for an unlimited number of games and apps across multiple platforms.

Having such a large royalty threshold makes it possible for developers to use UDK to establish themselves financially, keep overhead low and leverage the latest game engine technology.

Let's look at some real-life examples to understand what this means to a developer looking to establish themselves using UDK. These examples assume you've paid the UDK commercial licensing fee of \$99 and your digital store keeps 30 percent of gross revenue, so the most you're ever seeing on a \$1 sale is 70 cents. We noticed some people on our forums were confused and thought our percentage was applied to the price paid by end users. It is not. We only take a percentage of the portion of the revenue that you receive. In retail terms, this would be known as the "wholesale" amount as opposed to the "retail" amount.

Example 1: You release an app with a retail price of \$7 and it sells 10,000 units. The total retail sales are \$70,000 but your digital store pays you the wholesale amount of \$49,000. Therefore, you owe Epic no royalties as you have not crossed the \$50,000 threshold for revenue yet, so your total cost for using this world-class game engine technology has been \$99 and you're on your way to establishing yourself as a professional game developer.

Example 2: You sell 15,000 copies of your \$4.99 app. The total sales in the store are \$74,850. You receive \$52,395 of this as your revenue. Subtract the \$50,000 threshold and multiply the remaining \$2,395 by 25 percent and you owe Epic a sum just shy of \$600 with your total cost of using UDK in this situation being under \$700. You've kept 71 percent of the total money paid by end users and Epic has only received one percent.

Example 3: Now you're cooking. You sell 30,000 copies of your \$4.99 app for total app sales of \$149,700. In this situation, you'd be paying Epic \$13,697.50 and keeping \$91,092.50 of the \$104,790 paid out by the app store. That means you'd be keeping approximately 61 percent of the sales earned by your game and Epic would earn nine percent.

Once you go beyond these numbers you're likely on your way to becoming an established developer and we can work with you to transition to a full Unreal Engine 3 source code license with support so you can take your business to the next level. We have surprisingly economical licensing plans to suit nearly any size of project and budget. In addition, a source code license opens up the possibility of exploring a larger variety of platforms made possible by having Unreal Engine 3 source code including: PC, Mac, Xbox 360/XBLA, PlayStation 3/PSN, Sony's Next Generation Portable (NGP), iOS and Android.

Simple math shows that it's realistic to build a business with our technology. Epic is committed to regularly releasing free, updated versions of UDK. These updates give the community access to the underlying technology used to build *Infinity Blade*, *Bulletstorm* and *Gears of War 3*.

If you haven't tried Unreal Engine 3 yet, please check out the latest UDK release. There's nothing to lose. If you're ready for the next step, contact us.



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

Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award four times along with entry into the Hall of Fame. UE3 has won three consecutive Develop Industry Excellence Awards.

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

Follow @MarkRein on Twitter.

UPCOMING
EPIC ATTENDED
EVENTS


E3 Expo
Los Angeles
June 7-9, 2011

GDC
San Francisco
Feb. 28-Mar. 4, 2011

Please email: mrein@epicgames.com for appointments.

WWW.EPICGAMES.COM





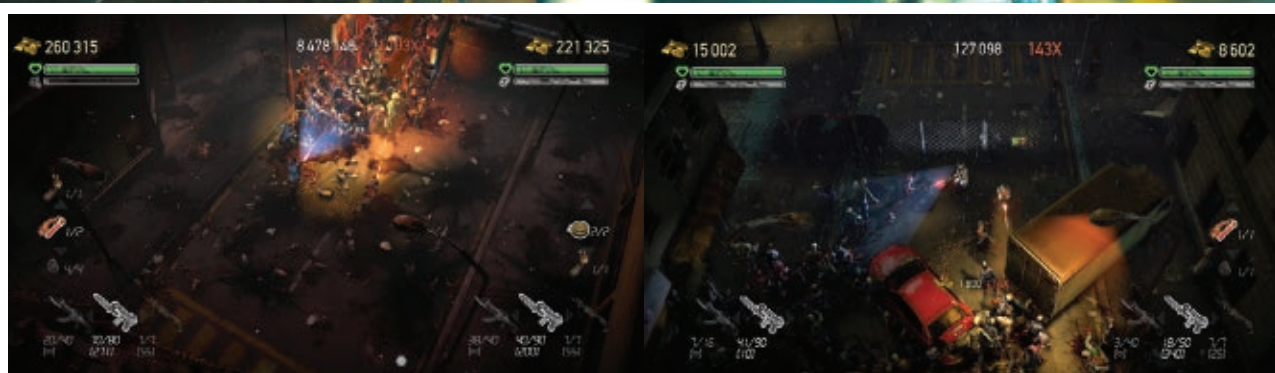
postmortem

dead nation

HARRI TIKKANEN AND
ILARI KUITTINEN

Three years ago, after the release of our PlayStation Network exclusive *SUPER STARDUST HD*, we began discussions with our senior producer at SCE, Phil Gaskell, regarding what we could do next. The hope was to make a *COMMANDO/TOTAL CARNAGE*-inspired twin stick shooter, again for PSN. Our “rumble in the jungle” idea turned into a zombie-infested apocalypse, as Phil suggested that the game should feature the undead. We went nuts over this idea, so it was quickly agreed that this was the direction to go!

We got into the prototyping after finishing off DLC work on *SUPER STARDUST HD* in the summer of 2008. The prototype used low fidelity art, with cubes representing the zombies that roamed the streets—our main goal was to prove that the core shooting mechanism was fun. The big change in twin stick controls from *SUPER STARDUST HD* was the addition of a separate button to shoot while using one stick to aim, rather than just using a single stick to



aim and shoot simultaneously. The prototype convinced Sony that we had a solid foundation, and we began preproduction in autumn.

As usual, plans started small but easily got out of hand. As a project, DEAD NATION grew larger during its second year of development as we (both us at Housemarque and Sony) wanted to implement co-op gameplay and a deeper upgrading system. Early user testing also suggested that players would like a story element, so we added a light storyline to frame the gameplay progression. We also extended the scoring system to further motivate score hunters.

As a whole, we have great interest and love both for twin stick shooters and for zombies. It was a no-brainer for us to combine two of our greatest loves together. Without further ado, let's delve a little deeper.

WHAT WENT RIGHT

1 /// PUBLISHER RELATIONSHIP. Sony as a publisher was one of the best "what went right" elements we had. After SUPER STARDUST HD, Sony had great trust in us. They put forth a lot of effort promoting DEAD NATION whenever possible, from regular trailers to a live action one, and frequent PlayStation Blog postings.

Sony also arranged user testing sessions for us, which provided lots of insights early on in development. Also, we found a very extensive closed beta test to be really useful as it generated tons of very good PR (players talking to each other later on) and feedback which

helped us iron out bugs and tweak gameplay further. We were given plenty of time to work on DEAD NATION without being forced to cut many corners, or jam fixes in.

2 /// ONLINE CO-OP AND METAGAME. One of the most important "late process" features we decided to add was also one of the most important. Some of us were really hoping for online co-op, and originally we thought that we would only have time to implement the feature as DLC after the game's release, if at all. Developing the feature required a lot of man hours from our coders, and we had to hire a new multiplayer programmer to make the dream a reality. We think it was well worth it. Matchmaking was included for quick online co-op games, and there was also, of course, co-op with specifically selected friends.

Another major online feature is the metagame, which doubles as the extended leaderboards. The metagame is essentially a detailed scoring system, and shows "progress reports" for anyone tracking how well their own country is doing compared to other nations around the world. People are also matched against their friends in addition to their countries, but the country the player represents is also matched against the other countries of the world. The metagame adds one extra layer of depth. We have more plans for the metagame to make it even more interesting in the future.

3 /// TWIN STICK GAMEPLAY. Since we felt we really understood twin stick controls quite well,

we definitely wanted to use the control scheme for DEAD NATION. It takes a bit of time to adjust for players who are unaccustomed to it, but it's very powerful, and allows precise action and aiming without auto-aim or cheated correction to player's aim.

Gameplay required a lot of time for us to adjust and tweak, as we didn't want DEAD NATION to work and act like a simple action shooter but to have a more realistic feel, more than just button smashing (although how realistic can a zombie game really be?). We went by the old game development axiom "Easy to play, but difficult to master," and we think we succeeded. It takes only a few minutes to get used to the controls, but from there, you're exploring, adjusting to the situation and threat level, and finding a way to counter the hordes of zombies. With this in mind, we also included the ability to almost fully remap your controller. The scoring system in the game is also quite deep and, like the gameplay, takes time to master and requires that you know what you're doing at all times.

4 /// ENGINE AND TECHNOLOGY. Our in-house Housemarque engine has never let us down, and with the PlayStation 3, we're able to draw even more power from it. There's no point for us to even consider any other engine solutions. DEAD NATION with fully dynamic lighting and shadows is a good example of what we can draw from the PlayStation 3. Although we had a really solid engine that we used for SUPER STARDUST HD, we were able to squeeze even



o
e
r
r
r
r
r

Scotland.
Famous for the
Loch Ness Monster
and digital expertise.



We've got quite a reputation for invention, innovation and creating some unbelievable stuff. MRI scanning. Radar. The microwave oven. Whisky. It's a long list. And it's still growing. That's why companies are doing business in Scotland. Our passion for success and

hunger to win, combined with our world-class academic institutions, outstanding research and superb facilities is financially irresistible. Scotland could be your next great discovery and the ideal place to develop new products and expand your business.

To see what we can do for your business, visit www.sdi.co.uk/nessie

SCOTLAND. SUCCESS LIKES IT HERE.



CONTINUED FROM PAGE 14

more out of our proprietary tech and added significant enhancements to performance during development.

Our toolsets are also very quick (and familiar), and they allow us to tweak and build levels in real time, instantly seeing changes without crossing our fingers and hoping for the best. The improvements we made to our tools as the project went along greatly helped development run smoothly.

5 /// AUDIO AND ATMOSPHERE. Considering our co-operation with AriTunes in the past with SUPER STARDUST HD, he was our top choice for implementing audio design and music for DEAD NATION. With his expertise and skill, we were able to really live up to the visual atmosphere we had built. Full surround sound really makes people squirm in their chairs as they mow down zombie hordes.

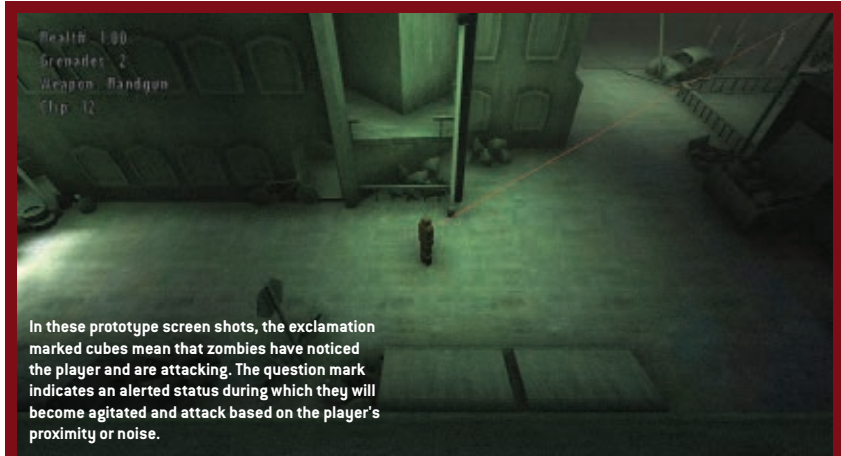
WHAT WENT WRONG

1 /// TIMING OF RELEASE. As the DEAD NATION project progressed from napkin notes, to proper project papers, to code, and finally to a running product on the PlayStation 3, zombie games grew and rose from their graves like mushrooms. When the DEAD NATION project started, we hadn't heard from zombie games for ages! By the time DEAD NATION was going gold, we were already battling against hordes of zombie games, both large and small.

Even though ours was the first zombie twin stick shooter released during 2010 on the PlayStation Network, it was also the third zombie-themed top-down shooter in total, so reviewers were getting tired of zombies in video games. Because of the theme of the game, some overlooked its more unique aspects, and seemed not to dig deep enough to find out what made DEAD NATION different from the others. Oddly enough, DEAD NATION was sometimes compared against full-priced zombie games instead of other downloadable games.

On the other hand, timing was both good and bad for us, due to the amount of active zombie activity both in and outside of games. Even though some were getting tired of the whole zombie genre, the success of the newly launched *The Walking Dead* TV series helped to reignite interest in our shuffling pals. We were also lucky that there weren't any comparable titles released on the PSN during the past 12 months or so before our game finally came out.

2 /// ESTIMATION OF THE SCALE OF THE PROJECT. We underestimated the scale of the project, especially as it grew iteration by iteration toward a larger, longer, and more complex game. At the beginning of production, the game was supposed to have more survival elements with a greater emphasis on looting and procedurally generated



In these prototype screen shots, the exclamation marked cubes mean that zombies have noticed the player and are attacking. The question mark indicates an alerted status during which they will become agitated and attack based on the player's proximity or noise.



levels for exploration. Eventually, we were able to create a more ambitious and complex title with more features than we originally had hoped to include in the game.

We stuck with the original idea for fairly a long time, testing the creation of procedurally generated levels, but found that we wouldn't be able to complete the development of the level generation system within our schedule or resources. Even though the system had taken us

quite a long time and a lot of effort to implement, we decided to scrap it in favor of ten individually designed, tuned, and polished levels.

Originally our upgrades were more geared toward a simpler item drop-based system, but after multiple iterations, we made a move toward a monetary upgrade system. The new system allowed players to make more choices with their hard-earned game currency, and made room for an upgrade path to suit individual playing styles.



3 /// TOO MUCH CONTENT FOR A DOWNLOADABLE GAME. In hindsight, our levels should have been smaller in scale. After all, we had a fairly small team for the game we were making, but we still wanted to make a title that would take hours to play through, with lots of replayability. As gameplay changed toward a more tactical direction, and as we increased emphasis on necessitated use of different items and weapons, we realized too late that we should have made the levels shorter and used the saved time to create even more variation and unique gameplay moments within the game.

As the amount of content required per level increased exponentially, we decided to outsource some of the level assets. Luckily, the whole outsourcing process was very straightforward and eased up some of the production pressure toward the end of the project. Sony helped us find some really good outsourcing companies—Pearl Digital Entertainment and Virtuos—and they proved to be very resourceful and experienced.

4 /// TUTORIAL SHOULD HAVE BEEN BETTER. Probably the biggest feature we didn't polish enough was a proper tutorial section. We managed to cram in a lightweight system with simple overlay text, which suggested to players what they should do to progress in the beginning of the game. It was relatively easy to ignore tutorial messages, and this made DEAD NATION a bit hard to get into for some players.

Unfortunately, the importance of the weapon upgrade system and use of items wasn't emphasized enough. One of the key aspects

of the game is crowd control of a vast number of zombies, so a player who doesn't fully use all available options will find the game much more difficult than it was designed to be. When players realize how to properly use items and the environment to their advantage, the game rewards them with deeply satisfying gameplay moments and feelings of accomplishment.

5 /// ALPHA TO MASTER CANDIDATE CHAOS. Although we passed Alpha with relative ease, we were stuck in Beta much longer than anticipated, and were unable to progress to Master Candidate submission due to a couple of missing non-critical features and assets. We also got stuck in the Master submission phase for quite a long time due to localization bugs and the large number of languages the game needed to support.

The online co-op mode was really the only major feature that gave us a hard time. Originally we were planning on only including offline co-op, as online would require much more time to do

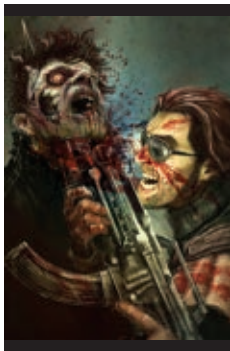
properly, but in the end, we felt we needed to get online co-op in for the launch.

CONCLUSIONS

/// We're pleased with what we were able to achieve with DEAD NATION, and hope to continue updating and adding new features alongside the downloadable content.

At the moment of this writing, we can't know for sure how DEAD NATION will fare on the PlayStation Network in the long run, but we're really pleased with the initial reception. Thanks to all our fans, and thank you for reading the DEAD NATION postmortem. ☺

HARRI TIKKANEN is creative director of Housemarque, and **ILARI KUITTINEN** is CEO. The rest of the team also contributed to this article. Both Harri and Ilari have been in game development since the early 90s and they co-founded Housemarque in 1995. The company is the oldest game development studio in Finland.



DEVELOPER Housemarque
PUBLISHER Sony Computer Entertainment Europe
PLATFORM PlayStation Network
RELEASE DATES November 30, 2010 (North America), December 1, 2010 (Europe)
TEAM SIZE AT THE BEGINNING OF THE PROJECT 4
TOTAL TEAM SIZE AT THE END OF THE PROJECT 12
R&D TEAM RESOURCES USED ON THE PROJECT 50 percent
NUMBER OF ZOMBIE TRANSPORT BOXES ORDERED TO THE OFFICE 1
TEAM MEMBERS ON SUMMER HOLIDAY 3 (in mid-December at the time of writing this article)
NUMBER OF ZOMBIES AT THE DEAD NATION RELEASE PARTY 43
NUMBER OF CONFIRMED SEVERED LIMBS AT THE RELEASE PARTY (WITH PHOTO CONFIRMATION) 1
TOTAL NUMBER OF ZOMBIES KILLED WORLDWIDE AFTER LAUNCH, AS OF DECEMBER 2010 Over 300,000,000

Launch fast. Land safe.

For visibly smart, unbelievably responsive performance, choose a 2nd gen Intel® Core™ i7 processor with an Intel® Solid-State Drive 510 Series. intel.com/go/ssd



Get visibly smart performance and twice the storage interface bandwidth of a traditional hard drive with the top-of-the-line 2nd gen Intel® Core™ i7 processor and a 6.0 Gb/s Intel® Solid-State Drive 510 Series in your PC.

It's your answer for an amazing PC experience. And, because there are no moving parts, you get extraordinary ruggedness to keep your data safe.

With the industry-leading technology of the 2nd gen Intel® Core™ i7 processor and an Intel® Solid-State Drive 510 Series, you have arrived.

old mario's

sid meier's civilization v
Sid Meier's Civilization V



civilization

postmortem

DENNIS SHIRK

ALMOST FOUR YEARS AGO, JON SHAFER LAID OUT HIS VISION FOR THE NEXT ITERATION OF SID MEIER'S CIVILIZATION TO OUR PROTOTYPE TEAM (WHICH CONSISTED OF JON, SEVEN ARTISTS, AND THE UBER-MODDABLE CIVILIZATION IV ENGINE). SOME OF THE MORE EXCITING FEATURES PROPOSED AT THE TIME INCLUDED SWEEPING CHANGES LIKE ONE UNIT PER TILE, HEXES, COMPLEX FULL-SCREEN LEADER ENVIRONMENTS, AND A NEW SCALE INVOLVING MORE UNITS ON-SCREEN THAN WE'VE EVER DISPLAYED BEFORE IN A CIVILIZATION GAME.

On the engineering side, the excitement surrounding the creation of a new engine to accommodate these systems had an amazing effect. The entire engineering team had the opportunity to create something unique, built from the ground up for Civilization V. On the art side, the team was challenged to create a completely believable world, and to produce leader scenes composed of fully fleshed-out characters greeting players in their native languages. We were setting out to create a completely new Civ experience, which got the team excited to bring the design to life. It's been a long road from prototype to final product, and as the vision was implemented, the challenges of delivering new concepts built on a new engine started to present themselves—but never in the places we expected them.

WHAT WENT RIGHT

1 /// CLEAR BOUNDARY BETWEEN GAMEPLAY AND ENGINE. Two of our major goals for the project were to support ambitious new gameplay changes (one unit per tile, hexes, and so forth), and to elevate our target for the visuals. The first priority was obvious. We were going to need to create an entirely new graphics engine to take advantage of features we wanted to use from Direct X 11. Given our schedule, this plan meant that our new engine wouldn't come online until 18 months before release—far too late for us to start testing these gameplay ideas.

Our solution was to enable a parallel development track for gameplay using the existing CIVILIZATION IV engine as the graphics component. We



needed to keep a very clear interface between gameplay and the engine so that we could do a quick swap of engines without having to halt development on either side. In the end, we were able to run gameplay with both engines for a few months as the swap took place, which ensured as seamless a transition as possible. Once we had everything back together in the new engine, we already had a game that had been refined for almost two years in its CIVILIZATION IV incubator.



PUBLISHER
2K Games

DEVELOPER
Firaxis Games

NUMBER OF DEVELOPERS
50 full-time, 6 contractors

LENGTH OF DEVELOPMENT
3 years, 3 months

RELEASE DATE
September 21, 2010

SOFTWARE
Visual Studio, 3D Studio Max, Perforce, multiple in-house tools

PLATFORM PC

2 /// OUR WONDERFUL, WONDERFUL FRIENDS. In January of 2009, our engineering team was still hard at work creating a completely new engine, including a custom renderer, for CIVILIZATION V. Broad testing across many different hardware platforms was something we had identified as a risk early on. As many of you may already know, PC development is tricky, to say the least. You can test on 500 different combinations of hardware, but once you release the game to millions of fans, your careful testing has a tendency to go out the window.

ATI, Intel, and nVidia were all instrumental in making sure this risk was minimized. Intel brought an engineer on-site to assist our graphics team with optimization for the new Core series of processors, and provided us

not only helped us with optimization, but also provided us with cutting-edge AMD systems for testing. Both nVidia and ATI made sure that we had enough video cards for the entire team so we could test on both older and newer hardware in-house, and most importantly, both gave us access to their amazing compatibility labs. This was instrumental in minimizing hardware compatibility issues when we launched.

That's not to say there weren't problems. As with any new engine, there were issues that needed correcting once we released the game into the wild, and our vendors were there with us, constantly updating their drivers, and continuing to make the end-user experience better with each iteration.

3 /// CO-LOCATING SUB-TEAMS. One of the new tactics we employed on this project was the co-location of our entire project team. From the start, the whole team sat on the same side of the building to increase ease of access to all teammates. Further, each smaller discipline—such as concept artists, modelers, animators, and the like—shared offices. Not only did we group certain disciplines together, we also clustered together people that frequently interacted with each other. For example, the lead designer, lead programmer, and lead artist shared an office. A graphics programmer was paired with certain artists to ensure that the group's technical needs were met and that the artists followed protocol. This daily face-to-face interaction improved communication among team members. Access to quick answers from coworkers sitting close by allowed the team to problem solve and overcome obstacles in a timely manner, which helped us reach our goal.

Office culture is defined by the people who work in a certain setting. One department or office may have an overall culture, but if you look beneath that, groups who sit together frequently form their own microcosmic tribe. In a video game company, you often find a clear dividing line between artists and programmers. When all artists sit on one side of the building and all programmers on the other, tensions can build between the two disciplines.



By simply arranging the office by project, the walls between artists and programmers begin to break down and communication improves. Instead of setting disciplines against each other, aligning people by project tends to reduce tribal behavior, encouraging people to be loyal toward the project they are working on rather than to the discipline they are working within.

Furthermore, there are other benefits that go largely unnoticed, such as informal conversations that naturally happen in an office setting throughout the day. What may start as an offhand comment can lead to incremental adjustments. Information and knowledge is often shared unofficially among those individuals who sit together. Decisions are made in these day-to-day interactions and are often not passed on formally because they sprung from casual conversation. Therefore, by allowing team members to sit among one another, we avoided redundancy and unnecessary backtracking as decisions became formal throughout the course of a day, a week, and a project.

4 /// EXPERIENCED AND DIVERSE ART TEAM. We had an experienced art team with a diverse skillset, which allowed us to solve problems quickly and effectively. At the outset, strong generalists were able to concept, model, and prototype their ideas quickly. This gave us a tremendous amount of flexibility to test concepts without a huge expenditure of man hours, and allowed the game designers to have a good idea of the visual direction we were taking early in the project.

Creating a unique title like CIVILIZATION requires a different perspective on iconography and problem solving. We had an advantage in that the majority of our artists had worked on previous iterations of the series and had good ideas to build upon. The interface lead incorporated many of the lessons learned from the console development of CIVILIZATION REVOLUTION. The concept artist had spent much of his career learning the costuming and design from different eras in history, and took the opportunity to show off his vast knowledge.

We also benefitted from having members of the team that had been lead artists at Firaxis in the past. These were artists who could be relied on to meet deadlines, be mature in conflict resolution, and be responsible with large aspects of the game. The project's art director had a wise and seasoned group of advisors upon whom he relied upon to point out mistakes and to lend a hand to help fix them. They were also understanding of the conflict between visual direction and needs of the game design, keeping morale high when compromises needed to be made.

Complementing the learning from this seasoned group were the junior artists. Their excitement level about being in the games industry, and getting an opportunity to work on a franchise as important as CIVILIZATION, also helped motivate the team during production. The passion and creativity of artists desiring to make their mark gave our interface exciting illustrations of the icons as well as the memorable landscape of our game.

We also had a strong variety of people that came from other art disciplines. We had an artist with a film background, one that had studied industrial design, and another that came from traditional 2D animation. Combining this diversity to achieve a singular goal made this a fun team to work with, and they were also effective at fitting the history of civilization into a single game.

5 /// IF ONLY THERE WERE ZOMBIES. When Jon Shafer originally laid out his plans for the modding systems in CIVILIZATION V, everyone on the team was excited by the prospect. With past versions, modding was extremely popular with our hardcore fan base, but most of our casual players never even knew that many of these epic mods existed. Enter Shaun Seckman, our modding lead. The systems he created and implemented will forever be used as an example of what we need to have in any game where we consider modding to be important.

The system we designed allows any person playing the game to search for and download mods directly into the game. There's no restarting, no technical knowledge needed, and it's simple to use. Marry this with the tools that we created, like the standalone World Builder, and suddenly anyone can become a scenario designer. As a result, our download numbers for mods have already surpassed the million mark, something we could not have imagined when we released the game. Special kudos need to go out to GameSpy and their Special Projects team. They provide the back-end server infrastructure that makes it all possible.

WHAT WENT WRONG

1 /// CLASH BETWEEN DESIGN CHANGES AND COMPLETING EXPECTED FEATURE SET. CIVILIZATION IV: BEYOND THE SWORD was as fleshed out a CIVILIZATION title as one could hope for. Expectations for a new version of the game would be extremely high, especially among our hardcore fan base. Since this was the fifth iteration of CIVILIZATION, our team came to the drawing board looking to do something profoundly new with the series. Our vision for CIVILIZATION V included many risky changes that would require a significant amount of new tech, and an even larger role for design and gameplay than in past versions.



The design radically changed three of the four types of victory from previous versions, and while this was exciting to us on paper, the challenges of designing and balancing it were numerous considering the schedule we had to keep.

One unit per tile was perhaps the biggest, most noticeable change. Whereas a player in previous versions would work with large stacks of units, one unit per tile was more about expanding the tactical game to make it more interesting and engaging. While I think we succeeded in this concept, the time commitment to this system needed by our design team was fairly costly, and it had a very real impact on the other core components of the game. An entirely new AI system also had to be created, and while great strides were made, we underestimated the time needed to make such a large system work in a consistent, competitive manner.

The reality is that the more we focused on brand-new systems to create a brand-new experience, the more we had to trim systems that players had come to expect from previous versions. We ultimately had to focus on making sure our core systems and new concepts were working well, sacrificing some of the less critical features. Some of our hardcore fans have been disappointed by the lack of certain features, but this prioritization has given us a solid foundation to build on, and we're restoring or improving most

of that functionality and more, as we continue to support the game moving forward.

2 /// OUR EXTERNAL DESIGN TEAM WAS NOT BROUGHT ONLINE UNTIL VERY LATE IN THE PROCESS. During CIVILIZATION IV's development, we utilized an external design group called "Frankenstein," which was primarily made up of some of the most hardcore fans of the series who know the game inside and out. Once they received NDAs, we passed them regular builds, and they provided extensive gameplay testing and feedback to the design team. For CIVILIZATION IV, we strongly believe that the working relationship between the Frankenstein group and our team was one of the key reasons for its success.

For this reason, we set up a new team with community veterans from the previous team, along with some new additions, and Jon started working with them early in the original CIVILIZATION V prototype process. At this point, we were still delivering builds that used the CIVILIZATION IV engine married to Jon's new game core. Because the engine had already been released DRM free, there were no issues pushing out regular builds to our external testers. The issues cropped up when the new engine was finally ready to make its debut.

When we were finally ready to move the game core to the new engine, our DRM solution (via Steam) was not yet approved, nor was

it integrated into the build system. Because this was a brand-new technology, there was significant work needed to get it to a place where we felt comfortable allowing the builds to start propagating out to our testers. The unfortunate thing is that the implementation took close to two months. Two months with no new builds going out to our external gameplay and design team. Two months with zero feedback. For a game that needs a tremendous amount of balance to perform well over the course of a 12-hour session, this was a painful process to go through.

Ultimately, we did get the external team back on track, but the lost time could not be recovered. Thankfully, Frankenstein is filled with possibly the most dedicated people on the planet. CIVILIZATION, for them and for us, is something that can never be left "as is." They've never stopped working, and continue to provide invaluable feedback that makes CIVILIZATION V a better game, even post release. As we march through the DLC process, we've put in place an aggressive patch schedule that allows us to incrementally improve almost every core mechanic of the game.

3 /// CRITICAL POSITIONS WERE STILL MISSING ENTERING PRODUCTION. Today's game development environment is heavily focused on the multiplayer component. Facebook is huge and MMOs are going strong. As a result, finding





qualified networking programmers has become akin to spotting a unicorn in your backyard. One of the biggest challenges we had to overcome was not having a staffed-up multiplayer team until well into production. We were fortunate to have a solid example with CIVILIZATION IV, but the amount of gameplay changes coupled with a completely new engine meant that much of it had to be coded from the ground up.

This is an area where 2K QA and our internal QA team and engineers adapted very well. With the compressed timeline, we had to put together an aggressive testing schedule to get multiplayer functioning well and ready to ship. Once we had core functionality set up, the multiplayer play sessions became extremely important. We organized a strike team composed of our networking engineers and two gameplay engineers to float around the office during the sessions. This way, as individuals ran into out-of-sync issues, we were able to identify the exact nature of each problem, correct the problem, and deploy new builds quickly.

While this successfully got us to a point where we were able to ship the game, the multiplayer experience was lacking many features that were present in previous versions of CIVILIZATION. We absolutely do not consider CIVILIZATION V's multiplayer to be a "closed book," and as with other aspects of the game, we are continuing to improve the experience to meet our standards, as well as those of our fans. And for the record, I think our engineering team still holds the edge for "games won" over 2K QA. I'm not, of course, including the last session where I was knocked out in 30 turns by a horsemen zerg.

4 /// INDUSTRIAL AND MODERN ERAS WERE NOT AS POLISHED AS THE FIRST HALF OF THE GAME.

Because of the amount of attention the new combat system demanded, a significant amount of time was spent fine-tuning and iterating this concept throughout the early eras of the game. As changes were made, new games were started and concepts were tested. The problem this introduced is that CIVILIZATION by its very nature is a long and involved game. The time requirements to test a game like this are significant. You cannot just test a single system, you have to constantly test to make sure the system works throughout the length of an entire game. What may work wonderfully for the Ancient era may not work as well for an Industrial or Modern era. While we do have the ability to start a game in the Modern era, for purposes of balance and gameplay, there is no substitute for playing through a full game.

Ultimately, there ended up being a large disparity between the amount of playtime invested in the first half of the game versus the time spent testing the second half of the game. When you're early in development, each new build had the potential to break earlier saves, so testers frequently had to start over from scratch, not always able to complete a game before the next build would rear its head. Because of this, there are some imbalances that were not revealed until the game made it into the hands of our fans. It really reinforced the notion that above all, we have to find a way to make sure that save file compatibility from build to build is always maintained. This can be really difficult when you're in the middle of design, but we learned a hard lesson about what can result from this.

5 /// LAYOFFS AND THE OBVIOUS EFFECT ON MORALE.

You cannot underestimate the effect a layoff has on team morale, especially when it lands in the final weeks of Beta. The realities of last year's recession unfortunately touched our team when we were at our busiest, trying to finalize all the features and submit our Gold candidate. We lost some critical team members. It's a situation that you can't possibly prepare for.

Dev teams become a tight-knit group three years into the making of a game, so there is lost productivity as people adjust to friends having lost their jobs. Bouncing back from this was challenging, but to the team's credit, we were able to regroup and refocus on the work that needed to be done. In the end, through some very hard work and extra help from our other development team, we were able to hit our street date and deliver a quality game. So, I suppose this can be considered both a What Went Wrong as well as a What Went Right.



MORE SETTLERS!

/// We're incredibly proud of what we accomplished in CIVILIZATION V. We overcame significant challenges in production to ship a critically acclaimed game on time, and one that is both completely moddable (with more content arriving every day, and DLL source code on the way), and that has introduced CIVILIZATION to a new generation of players. We're aware of the high expectations of our fan community and are glad that overall, they feel we've delivered another great CIV game. We're committed to supporting the game as we move forward and are thankful for the support and feedback from the community in that effort. 🙌

DENNIS SHIRK joined Firaxis in 2005 and was the lead producer for SID MEIER'S CIVILIZATION V.



red barrels

why

We're trying really hard to make an awesome and innovative game with BULLETSTORM. There's no shooter out there like it, and to support that idea, I want to discuss the process of one of our most groundbreaking features.

EXPLODING RED BARRELS

\\\ I know, I know, it sounds crazy! But tests have shown that they work, and people "get the idea." You shoot them, and they go boom! In fact, we tried something else at first ...

It seems like it's a game development curse. Pretty much every action game has red barrels that will explode if you shoot them. One could easily argue that going with the tried and proven way shows the developer's lack of imagination, to go with the tried and proven way. As it turns out, it's not that simple. This was our first stab at trying something different with barrels instead of going with the cliché.

GREEN BARRELS!

\\\ Not only was it somewhat refreshing because it wasn't playing along with the stereotype, it also made sense as far as the story goes. You see, there's actually a valid reason for the barrels to be green in the world of Stygia in BULLETSTORM. The city Elysium is protected by a gamma filter from deadly storms. This creates a massive amount of toxic waste that is stored and contained in ... barrels! Those barrels are stored underground, but there's a catch: the toxic waste causes people to get sick, die, and mutate if they survive exposure. You'll get to meet these folks when you play the game.

We had the green barrel for a long, long time. We're talking years here (in fact, the first time they appeared was in a level that was cut early in development—see Figure 1). It worked because everyone who had tried the earlier versions was a mid to hardcore gamer, so they were pretty quick to test things out and basically shoot at everything. But as the game started to actually resemble something like its final form, it underwent a lot of testing and prototyping, and a lot more people got their hands on it.

Don't get me wrong, the old barrel used several tricks to make it look explosive in spite of the color. For example, it featured windows revealing the contents—a mass of waste that looked very unstable and dangerous.

Also, the use of black and yellow stripes along the edges signaled "warning" in a very classic manner. (See Figure 2.)

Most people realized that the barrel was dangerous and interactive after looking at it. And almost all players eventually figured out how to use it and what its attributes were. The issue is that we want people to immediately understand all that. The player shouldn't be required to invest time and dedicate brainpower toward deciphering the barrel's purpose: it should be obvious right away! There's a lot of stuff going on simultaneously in BULLETSTORM, so it's vital for us that the player be able to quickly read the environment and act accordingly. This is especially true since you can also leash and kick objects, so barrels can be flying all over the place. There's no time to analyze objects on a detailed level, so the shape and color have to be enough.

And I must admit, there were quite a few people who didn't pay attention to the barrels at all. They were completely ignored by some players, and none of them guessed or assumed that they were explosive.

Why not? Because they weren't red. Everyone knows that only red barrels are explosive! It became apparent for us that the most efficient way to communicate their purpose was to make them red.

Red not only makes them more obviously explosive, it also helps them stand out from the environment and background. There's already a lot of green and yellow in the world, so using a completely different color for the objects we want you to see makes sense.

ONLY RED WILL DO

\\\ "Just make it red" seems like a straightforward approach (see Figure 3), but apparently it wasn't as simple as I imagined. I've done some research into the process of how we reached the "final red barrel" look. Barrels aren't my main area of expertise, so I had to check with the rest of the team.

I asked Karolina, our art producer, to show me a picture of both the old and new ones. She sent me a picture with several different barrels, and when I inquired why there were multiple variants, the story finally broke loose.

Looking at the picture she sent (Figure 4), there are four different versions. One would assume that the barrel to the right is the one that got made, considering the circle and the check mark? But nooo, guess which one finally made it into the game?



we used red barrels in **bulletstorm** instead of green.

ns boom!

I bet you guessed wrong. The correct answer is “none of them.” Number four was the intended barrel, the one the team decided on after a meeting. That change would only require a new texture.

So the very first try was to change the texture of the original barrel, making it red and adding some flammable symbols. It was quick and easy. But then our artists suggested that we change the model as well, “while we’re at it, anyway.” They had new and cooler ideas than way back when, apparently. Normally, you can’t just say, “Let’s redo this,” but they saw the opportunity and grabbed it! And somehow, production allowed it.

This would mean they made either of the two left most barrels, right? Wrong again.

It ended up being almost in the shape of the two barrels left with some minor tweaks, and the texture was a mix between the flame logo from the left most one and the window from its neighbor, instead of the grating. (See Figures 5 and 6.)

People may ask us, “Why not just make a regular barrel and paint it red? What’s the deal with all the details like the handle, wheels, grating, and so on? And the new barrel is even more “sci-fi” than the old one! What gives?” There are a couple of reasons for it.

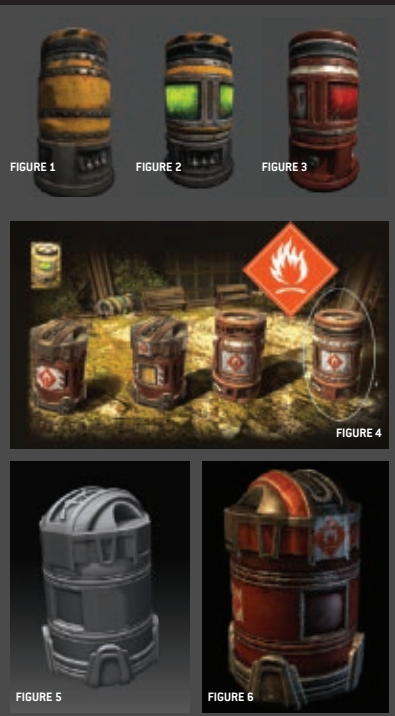
First, we want it to look like it works for its intended purpose. Amusingly, some say that it’s too “unrealistic,” but to be honest, the whole shape is based on a weird kind of Japanese fish canister that one of our artists found while exploring the vast universe known as the Internet. Admittedly, that one is made out of plastic; our barrel is metal.

However, the main reason is because it looks cooler! While it resembles a regular barrel in its basic cylindrical shape, **BULLETSTORM** is meant to be a joyride, and we want everything to be interesting! The worst thing that could happen for us as developers would be if players got bored with it. We don’t want people to play our game and talk to their friend the day after and say, “Oh, the game was awesome, but man, the barrels were extremely dull looking.”

In addition to barrels, we have other explosive objects, one of them being trash cans. These incinerate trash with something like a built-in futuristic furnace. What’s funny is that even after we’d changed the old barrel, the green trash can remained green for a few more months. I don’t know why. Maybe we kept clinging to the hope of having something green explode. Nevertheless, we later changed it to red as well, making sure to use a unified and intuitive visual language in our game. Is it red? Yeah. Well then, shoot it!

I’m glad I’m a game designer and not an artist when it comes to finding the best match of shape, logos, and colors. For me, “red cylinder” is enough. Regardless, the artsy people did a great job in the end, as now everyone expects the barrels to blow up—and they do!

I never expected that I’d be writing an article about the barrels in our game, but then again, the topic is one that’s been brought up in the industry since the dawn of time. We tried not doing red barrels, but as the old adage goes, the customer is always right! 🍌



ARCADE BERG is a game designer on **BULLETSTORM** and *People Can Fly's* community manager. He legally changed his first name on his 18th birthday. It's no surprise that he later started working on games.



AUTODESK Maya 2011

Game artists know that there are three sure things in life: death, taxes, and Autodesk's release schedule. The 2011 version of Autodesk Maya is chock-full of new and updated features, including a long awaited user interface overhaul, so let's take a look at some of these features and find out if 2011's beauty is more than skin deep.

IN THE BEGINNING

» As the new Qt framework-based UI is the most obvious update and the new feature that users will encounter first, let's start there. In addition to the new carbonized look, which Autodesk claims will help reduce color bleed from the UI to the viewport, Qt brings a new level of UI customization to Maya. Some elements can now be torn off and docked in new locations and other elements can be resized by dragging. Have you ever needed to pull out the Attribute Editor just a little further to see that super long attribute name? Well, now you can!

Several individual windows have also been updated. Those that rely on custom shelves will love the new options provided by the updated Shelf Editor. New icon formats, including GIF and JPG, are supported, and individual icon elements such as background color and transparency can be customized. Maybe it's time to hire that full-time icon artist you've been considering! 2011 also includes a new File Browser, which allows users to set bookmarks, manage projects, and even set file options such as referencing and namespaces. Speaking of namespaces, Maya 2011 includes a new Namespace Editor that allows users to view and manage a scene's namespaces, as well as manipulate the contents. Overall, Autodesk wasn't content to just make the

new UI more attractive; they've also made it more functional as well. That said, it will take some time to adjust to the new layout.

DO MY LITTLE TURN

» Now for some of the new modeling features in 2011. Past versions of Maya may not have been at the top of most artists' list when they were discussing preferred modeling packages, but Autodesk has been taking great strides to improve that perception.

The first major modeling feature is Export To Mudbox. Many studios have probably scripted their own interop pipelines by now. Even so, Autodesk's solution is pretty slick. Tell Maya where Mudbox is located, select some objects, hit File > Export To Mudbox, and Mudbox launches with your selection in view. Interoperability is a big theme for 2011, and this feature is a shining example.

2011 ships with many other compelling additions to the modeling palette, and while space constraints keep us from discussing them all, there are a few that stand out. The first two are extremely useful transformation tools: Arbitrary Scale and Object Level Soft-Selection. Arbitrary Scale scales an object along one of several presets outside of the standard defaults, including a "Custom Axis" option. This option derives a scale axis from a selected component's orientation. This remains in effect until a new axis is selected or the axis is reset to one of the defaults. Object Level Soft-Selection takes Maya's existing soft selection paradigm and applies it to transforms, allowing multiple objects to be transformed with a falloff, which is great for adding a random—but not TOO random—element to object placement.

Another feature to receive expanded functionality is Transfer



AUTODESK Maya 2011

Autodesk, Inc.
111 McInnis Parkway
San Rafael, CA 94903
www.autodesk.com/maya

PRICE

> \$3,495

SYSTEM REQUIREMENTS

> Microsoft Windows Vista Business (SP2 or higher), Microsoft Windows XP Professional (SP3 or higher), Microsoft Windows 7 Professional operating system. Intel Pentium 4 or higher, AMD Athlon 64, AMD Opteron processor, AMD Phenom processor. 2 GB RAM. 4 GB free hard drive space. Qualified hardware-accelerated OpenGL graphics card.

PROS

- 1 New features are genuine workflow enhancers
- 2 Interoperability with other Autodesk packages is much improved
- 3 Native PyMEL

CONS

- 1 Some new features could use additional functionality
- 2 New UI will take time to adjust to
- 3 Some features may not be used by specialists

Attribute. For 2011, Transfer Attribute adds topology-based transferring for identical meshes. No more having to fear the loss of component-based information due to component re-ordering by other packages.

Last but not least, Autodesk throws a nod toward 2D packages with the inclusion of Bezier Curves. If you've ever used Photoshop, you know what to expect. Click to set an

anchor point, drag to pull tangents, and switch into component mode to edit. Simple, effective, and familiar, what more could one want?

LIKE TO MOVE IT, MOVE IT

» Animation has been Maya's bread and butter since its inception, and 2011 continues that tradition. Autodesk has taken an "if it ain't broke, make it better" approach to animation tools, thus many of the familiar standbys have received major overhauls. Let's start by taking a look at the updated Graph Editor. Again, the list of changes is too extensive to describe fully, so I'll focus on some of the more impactful features. First up, two features from Maya's modeling toolset come to the animation world: Pre-Selection Highlighting and Pick-Walking. Both curves and key-frames can be pre-selected, though they tend to work better with curves.

In future versions, it would be cool to see tangents display in pre-select mode, but this is a great start. With keys selected, the left and right arrows can be used to pick-walk along the parent curve. It's a small feature, but animators will wonder how they ever got by without it. Again, it would be cool to see some additional functionality here. Perhaps the up and down keys could be used to cycle between all the displayed curves? In any case, it's encouraging to see workflow concepts shared across disciplines.

Maya 2011 simplifies Graph Editor work further with the addition of new display modes and filters. Tired of looking at all those overlapping curves? Set your second monitor to portrait mode, enable Stacked Curve display, and enjoy squint-free curve editing. Need to compare those curves to curves on another character?



The dark interface is intended to provide better contrast with the user's viewport image.

Individual channels can now be pinned in the Graph Editor, leaving you free to select other objects safely while keeping your original selection in view. Furthermore, 2011 provides the ability to filter specific attributes. For future versions, a bit more granularity would be appreciated (rotateX vs. rotate), but even this initial offering is quite useful.

Character technical directors will find that the Paint Skin Weights tool has also gone through some major changes, but all for the better. Sort By Hierarchy has been replaced by a proper tree view, though the old Flat display also remains available. Fixing minor errant weighting values is less painful, as influence weights can now be displayed as a color gradient instead of simple black and white. Wondering where that last .01 bit of weighting is? Set the left side of your gradient to neon green, and never again hunt for those weight spikes. 2011 also includes a 3ds Max-style Interactive Binding tool complete with preview volumes, as well as support for dual quaternion skinning. Weighted meshes now have the option of being linear, DQS, or blended, which can be painted on using Paint Skin Weights. The end result is skinning that's quicker to set up and easier to edit than in any previous version of Maya.

For cinematic animators, Maya 2011 borrows a few pages from MotionBuilder with the addition of the Camera Sequencer

and Time Warp Effects. Fans of MotionBuilder's Story Tool will feel right at home with both of these features. The Camera Sequencer provides a quick way to block camera shots and sequences independent of the timeline, similar to Trax, while Time Warp Effects allow a scene's timing to be controlled by a single animation curve. Rounding out the cinematics toolset is the option to display Time Code in scene, either as a HUD widget or in the time and range sliders.

Of all the new feature sets in 2011, animation feels the most polished, with some minor room for improvement. Hopefully the community will adopt these new features, give them proper trials by fire, and provide Autodesk feedback for future versions.

ROLLING YOUR OWN

While Python has been available in Maya since version 8.5, many users have been asking for a proper object-oriented scripting solution. This came in the form of a project called PyMEL. Lack of official support was a sticking point for adoption by many developers. Autodesk heeded the community's feedback and has now included PyMEL with 2011. This is one of the most significant scripting updates since the inclusion of Python. Porting existing pipelines does take time, but it's time well spent.

If PyMEL's inclusion weren't enough to keep technical artists on their toes, Maya's adoption

of Qt provides yet another new development paradigm: Qt UIs can be scripted using helper libraries such as PyQt or PySide; likewise, tool developers can create Qt UIs in their custom Maya API plugins. As Autodesk states, "The use of Qt is fairly transparent," and this is for the most part true. Custom UIs may exhibit minor differences on first run, but usually nothing show stopping.

The addition of PyMEL is as significant as the addition of Python itself. Autodesk has a long history of integrating community-developed features, and this one was an excellent choice.

PRAISE THE NEW ORDER!

With Maya 2011, Autodesk has stated loud and clear that they're not content to rest on their laurels. We've been hearing about things like Maya/MotionBuilder

convergence and improved interoperability for a bit now, and if Maya 2011 is any indication, the future is bright indeed! Granted, switching to 2011 in the middle of development may not be the best plan, but if the project is at a stable point, it's time and effort well spent. Studios should allow time for users to gain a solid understanding of the new features and how best to leverage them. A 30-day free trial of Autodesk Maya 2011 is available at: www.autodesk.com/mayatrial. As cliché as it sounds, there really is something in Maya 2011 for everyone, so whether your studio is looking to update its existing software or switch to a new package, this application should be at the top of your shopping list. 🎮

SETH GIBSON is a technical artist at Microsoft Game Studios.



FINALBUILDER

www.finalbuilder.com/game



BASEHEAD INC.

Basehead 2.5

In Sound Design, the challenge is often finding the right sound or combination of sounds to represent—either realistically or with abstract emotional impact—the drama unfolding on-screen. This quest for the secret ingredient to sell the moment with an appropriate and engaging sound is rife with challenge and uncertainty at the onset.

BRING THE NOISE

» The fundamental aspect of all sound design is, put simply: sounds, or the building blocks of sound that are used to create new sounds.

The most common way to interact with a growing library of sound files is through the use of a Digital Asset Management system. While everyone is familiar with the ability to navigate files using the search functionality provided at the folder level by your operating system, audio-specific solutions come with additional features to further enable the creative process.

If you can think of an audio asset management system that organizes and navigates files similar to the way iTunes manages music, then you're most of the way toward understanding how a specialized tool can help in this process. With a bit of extra pipeline for editing and manipulating the source files, this combination of searchability and

functionality forms the basis for the BaseHead SFX Database for PC and OS X.

The installer comes zipped in a slim 692kb file directly from www.baseheadinc.com. An initial 30-day trial is available and, once you decide you can't live without it, a license and CodeMeter CMStick can be purchased online to continue use. Inside the re-skinnable interface, a jump to the preferences will allow you to specify an already existing database location, your external wave editor of choice for file editing, as well as other description, playback, and workflow tweaks to help speed up usage.

THIS IS IMPORT(ANT)

» Several fields of metadata are selectable to be imported along with your sound files, and used with BaseHead's search capability; things like bit rate, sample rate, file type, channels and so on (see Figure 1). BaseHead also supports editing the universal broadcast WAV description metadata field.

In addition to the metadata embedded within your files, BaseHead parses the file names for possible descriptions, or keywords. It's here that the strength of file naming pays off, especially in cases where time has been spent to include relevant information such as a naming standard across different types of content.

I imported a large collection of

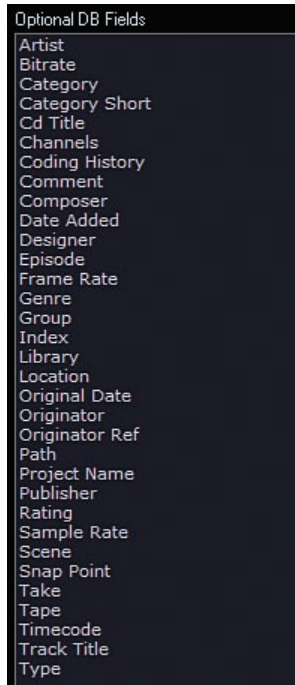


FIGURE 1 Optional database fields.

around 80,000 files from several libraries and waited patiently while the program slowly digested the information. It eventually stopped responding and after force quitting and re-opening the program I found my collection only partially imported. I changed strategies to the drag and drop methodology and selected multiple folders and dropped them on the Import window. After all this was

done I found that I had created many duplicate file entries in the database, which were quickly dispatched using the "Remove Duplicate Database Entries" feature.

FIND YOUR BLISS

» When it comes down to quickly parsing a massive library of sounds, a content management system lives and dies on its ability to quickly refine a search into a resulting list of appropriate potentials. With BaseHead, the flexibility is at your fingertips. All entries are sortable by column header including any of the metadata fields you specified on import.

Each of the three fields in the Search Bar (see Figure 2) is configurable, including arguments or Booleans for advanced look-up techniques and a chronological history of searches for each field. Keywords can be further augmented by using general expressions such as "-metal" to remove descriptions with the word metal in them. There is also a handy visual logic connector that goes above and beyond simple search and the ability to hack away at the forest of files is made easier by this extended functionality.

One of the handy tweaks in the preferences allows you to limit the number of records returned by a search. This speeds up the return of relevant information, and puts you two clicks away from more records

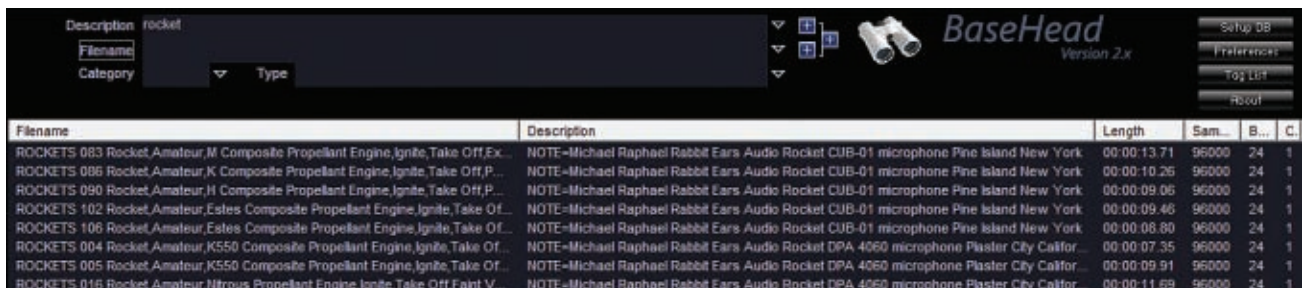


FIGURE 2 Search bar.

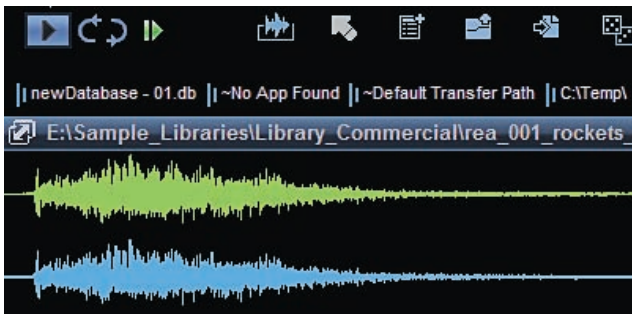


FIGURE 3 Waveform window.

if your initial batch of results feels too limited. Also of note is the ability to randomly select a file, for those situations when your creativity needs an oblique jumpstart strategy.

WORK FLOWING

» Once you've narrowed down your selections, you can listen to and skip around inside of an individual file using the Waveform Window (see Figure 3). It's not uncommon for a sound effect library file to contain multiple takes or samples, which makes visually skipping around within the file extremely helpful when identifying sections for later use. These individual sections can be highlighted in the waveform view and then added to the Tag List (see Figure 4). The Tag List is like an intermediate clipboard where you can assemble sounds in preparation for a mass export to your Digital Audio Workstation (DAW).

Support for all the major audio editors makes it easy to get your audio from BaseHead over to your

DAW of choice. The developer has gone a long way toward making this part of the process as brain dead simple and error free as possible in order to keep the creative flow going. The names of files or sections of files in the Tag List can be appended when exporting to your DAW with additional Pre or Post-Descriptions to help keep things straight once you enter the multitrack domain.

BACK TO THE FUTURE

» The online manual comes in handy for addressing several unique circumstances such as how to disable the Windows Bing sound that plays when pressing enter on any search, workflow speedups, and additional hot keys. It's important to note that version 2.8 of BaseHead has recently been released and that the company will be showing version 3.x of BaseHead at this year's GDC. The upcoming version promises additional features such as batch filename, a completely

new GUI design, labeling and color-coding, and VST plug-in support.

END OF THE ROAD

» When you find yourself faced with a steep descent into the mouth of cavernous silence, you need every tool at your disposal to mount the journey onward. Before settling on the course ahead it's good to feel prepared. While BaseHead comes with some quirks that can confound you on your sound design expeditions, there are still plenty of reasons to give it a shot in your creative pipeline. That random button just might help you out of a tight spot when you least expect it. [🔊](#)

DAMIAN KASTBAUER is a freelance technical sound designer working with Bay Area Sound. He is the co-founder of the Game Audio Podcast and writes a series of articles on audio implementation at DesigningSound.org.

BASEHEAD INC. Basehead 2.5

BaseHead Inc.
20413 Hamlin St.
Winnetka, CA 91306
www.baseheadinc.com

PRICE

> \$259.00

SYSTEM REQUIREMENTS

> Windows, OS X

PROS

- 1 Flexible and powerful search functionality
- 2 Exporting files to a DAW works smoothly
- 3 Waveform Window makes auditioning intuitive

CONS

- 1 Occasionally unresponsive when importing or searching large data sets

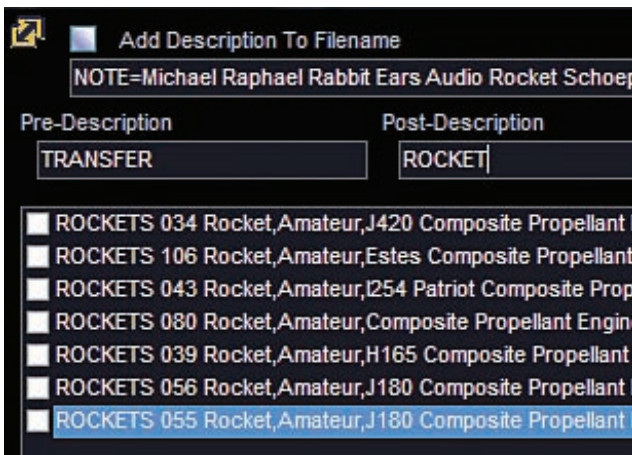


FIGURE 4 Tag list.

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

Become a
Unity 3D
Master!

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

unityworkshop

WWW.UNITYWORKSHOP.COM | 1.877.895.6882



WATER FINDS A CRACK

HOW PLAYER OPTIMIZATION CAN KILL A GAME'S DESIGN

This is what games are for. They teach us things so that we can minimize risk and know what choices to make. Phrased another way, the destiny of games is to become boring, not to be fun. Those of us who want games to be fun are fighting a losing battle against the human brain because fun is a process and routine is its destination.

—RAPH KOSTER, A THEORY OF FUN

MANY PLAYERS CANNOT HELP APPROACHING

a game as an optimization puzzle. What gives the most reward for the least risk? What strategy provides the highest chance—or even a guaranteed chance—of success? Given the opportunity, players will optimize the fun out of a game.

Games are so complex that it is difficult to anticipate exactly how players will optimize a game until after release, once thousands bang away at it and share their ideas with each other online. Often, designers don't even understand their own games until they finally see them in the wild.

A phrase we used on the CIVILIZATION development team to describe this phenomenon

was "water finds a crack," meaning that any hole a player can possibly find in the game's design will be inevitably abused over and over. The greatest danger is that once a player discovers such an exploit, she will never be able to play the game again without using it; the knowledge cannot be ignored or forgotten, even if the player wishes otherwise.

CIVILIZATION 3 provides a simple example with "lumberjacking"—the practice of farming forests for infinite production. Chopping down a forest gives 10 hammers to the nearest city. However, forests can also be replanted once the appropriate tech is discovered.

This set of rules encourages players to have a worker planting a forest and chopping it down on

every tile within their empire in order to create an endless supply of hammers. However, the process itself is tedious and mind-numbing, killing the fun for players who wanted to play optimally.

TANK-MAGES AND INFINITE CITY SLEAZE

» One of the dangers of players looking to optimize a game is that a single dominant strategy will emerge that drowns out all others. In the MMO world, the shorthand term for this predicament is the "tank-mage"—a reference to ULTIMA ONLINE, in which certain hybrid class builds could both wear heavy armor and cast powerful damage spells. This character served as both the damage absorber (the "tank") and the damage dealer (the "mage"), displacing most

other possible character builds. Almost every MMO has experienced some version of the tank-mage as players try to find the optimal build for all situations.

The CIV series has its own version of the tank-mage: the strategy of spamming settlers for “infinite city sleaze” (or ICS), a bane of the franchise since the beginning. The essential problem is that 50 size-2 cities are more powerful than 5 size-20 cities as a number of bonuses are given out on a per-city basis. For example, every city gets to work its home tile for free, which means that a size-2 city works 3 tiles with only 2 citizens (1.5 tiles per citizen) while a size-20 city works 21 tiles (only 1.05 tiles per citizen).

The problem is that while ICS makes beating the highest difficulty levels trivially easy, handling 100 cities is a management nightmare. Players who pursued this strategy—or even less extreme versions of it—were always aware that they were breaking the game, but they often simply couldn’t stop themselves.

Armed with knowledge from the earlier versions of the game, we were able to counter ICS ahead of time with CIV 4 by adding a per-city maintenance cost that scaled with the total number of cities. Thus, building too many cities too early crippled a player’s economy, killing ICS at long last.

The reason to kill tank-mages and ICS is that a single, dominant strategy actually takes away choice from a game because all other options are provably sub-optimal. The sweet spot for game design is when a specific decision is right in some circumstances but not in others, with a wide grey area between the two extremes. Games lose their dynamic quality once a strategy emerges that dominates under all conditions.

UNDERVALUING TIME

» When presenting players with a choice, games typically pair a specific reward with a certain level of risk. When gamers discover that one play style offers a trickle of rewards for little or no risk, they will inevitably gravitate toward that degenerate strategy.

In other words, players will trade time for safety, but they risk undervaluing their own time to the point that they are undermining their own enjoyment of the game. A classic example is the skill system from MORROWIND, which rewards players for repeating any activity. Running into a wall for hours increases the Athletics skill while jumping over and over again increases the Acrobatics skill. Many players couldn’t stop themselves from spending hours doing mindless activities for these cheap rewards.

Another example of players undervaluing their own time comes from growth, production, and research overflow in the CIV series. Every turn, cities produce food, hammers, and beakers, filling up various boxes. Once these

boxes are full, new citizens, buildings, units, and technologies are created.

For example, if a civilization produces 20 beakers per turn, and Writing costs 100 beakers, the technology will be discovered after 5 turns. However, if the same civilization produces 21 beakers per turn, the box for Writing will contain 105 beakers at the end of 5 turns. In that situation, after Writing is discovered, the extra 5 beakers are thrown away so that the box will be empty when the player starts researching Alphabet on the next turn. Players quickly realized that when they came close to finishing a tech, they could adjust their tax rate so that no beakers would be wasted (because those beakers are all potential gold at a different rate).

A similar dynamic exists with food and hammers for city growth and production. In this way, the game’s rules encourage players to visit every city every turn to rearrange their citizens to ensure no food or hammers will be lost. This micro-management is actually a somewhat interesting sub-game, but clearly not how the designers want the players to be spending their time, as it completely bogs down the game. We solved this in CIV 4 by simply applying the overflow food/hammers/beakers to the next citizen/unit/building/technology.

Players who adopt this strategy often refer to the game as being heavy on “micro-management” because they can no longer resist playing the game without squeezing every last drop out of their cities. The problem is even



worse in multi-player as gamers who don’t micro-manage their cities will always fall behind in the race for more growth and production.

The designers don’t want people to play this way; nonetheless, the rules inadvertently encourage it. Again, designers often don’t understand their own games as well as the players do. The problem with a gamer undervaluing her own time is that, while the easy

rewards may feel good at first, eventually the amount of time required will slowly seep away the fun per minute, until the game begins to feel like a grind.

GOOD EXPLOITS?

» Designers can also go too far trying to remove all exploits from a game. Often, the right choice depends upon the game’s context. Does the exploit drown out all other play styles, or is it a fun, alternative way to play? Does the degenerate strategy create an endless grind, or is it a quick shortcut for players who need a little help?

The famous endless free lives trick from SUPER MARIO BROS.—in which the player bounced a turtle shell repeatedly against a block staircase for long strings of 1UPs—was actually not a bug but a feature the team included. In exchange for mastering a small dexterity challenge, players can quickly mine all the free lives they need to progress in the game. Discovering and abusing a hole in a game’s design can be a fun experience—giving the player a unique sense of mastery—as long as the exploit doesn’t ruin the game for the player (or the player’s opponents).

If possible, designers should provide the ability to turn an exploit on or off, giving the players control over their worst instincts. For example, most games with save/load functionality can be abused by players to improve their odds; an RPG in which smashing a box produces random loot can be reloaded as many times as necessary until the best possible weapon or armor appears.

With CIV 3, we introduced a feature that preserved the game’s random seed in the save game file, guaranteeing that individual combats would play out the same way regardless of how many times the player reloaded the game. No longer were players tempted to reload every bad combat result, which could slow the game to a crawl.

However, the community response was not what we anticipated. Although some players appreciated that they were no longer tempted to reload combats, many others were frustrated that one of their old tricks disappeared. Indeed, some angry fans actually felt that the game was cheating on them by always reproducing the same combat result!

We solved this problem by turning this feature into an option on game start. Players who want the chance to reload a particularly unlucky roll can use the old exploit, but the game, by default, discourages this work-intensive strategy. Ultimately, the designer can’t go wrong putting the player in control of his or her own experience. 🎮

SOREN JOHNSON is a designer/programmer at EA2D, working on web-based gaming with strategystation.com and DRAGON AGE LEGENDS. He was the lead designer of CIVILIZATION IV and the co-designer of CIVILIZATION III. Read more of his thoughts on game design at www.designer-notes.com.



WE ARE THE

igda

Developers helping developers

www.igda.org/join



IDLE THREADS

SHOW THE DOOR TO LOAD-HIT-STORE

IF YOU'VE DONE MUCH WORK ON THE XBOX 360 OR PLAYSTATION 3, YOU may have heard how important it is to minimize load-hit-store (LHS) performance penalties. LHS is an issue that particularly affects in-order processors, and the PowerPC core(s) driving these consoles are no exception.

LHS is trivial to explain in principle: if you try to read back and make use of data from a memory location to which you have very recently written, your hardware thread will flush and you will sit idle for a while. (If you're wondering why you would ever read something back that you only just wrote, the practical examples will help to illustrate many of the situations where this can happen.)

For those who don't know the nitty-gritty, here's a more in-depth explanation of what's going on under the hood: When you write to memory, your store goes directly to the L1/L2 cache via a store queue. This is not an instantaneous operation; it can take on the order of tens of cycles for the store queue to flush and for your write to make it to cache. If you try to read that data back and make use of it before it has reached the cache, your thread will flush (sit idle) for around 40 cycles—this is the load-hit-store penalty. Once that flush has completed, the instruction pipeline is “rewound” to the instruction that caused the LHS, and execution commences again. This time around, the data is likely to have reached the cache, so it can be fetched and used successfully.

OKAY—WHY SHOULD I CARE?

» If you don't consider yourself someone who writes performance-critical code, you may be ready to skip this article (assuming you even read this far).

Here's the thing: despite being easy to describe, many of the situations where LHS issues appear are fairly non-intuitive, and they're certainly not limited to “low-level” code. The good news is that once you understand the basic mechanism of the issue and some of the common pitfalls, you should have a good sense for whether the code you're writing is likely to suffer from this problem. Better yet, in many cases, you should be able to make minor adjustments to greatly reduce LHS impact or even eliminate it altogether. I have made incredibly slight changes to code to eliminate LHS and seen that code double or triple in speed.

With such easy performance gains on the table—gains that can be achieved at a relatively low cost and without a need for complex code changes—why wouldn't you care?

IDENTIFYING LOAD-HIT-STORE ISSUES

» There are a few useful tools available for identifying existing LHS issues in your codebase. On the 360, you have PIX, trace dumps, and the PMCPB performance counters. On the PlayStation 3, you've got the pipeline analysis output and LHS performance counter tracking within the SN Tuner.

These tools are all well explained in their respective system documentation, and space (not to mention NDAs) prevents a more in-depth discussion here.

Of course, regardless of platform, you have another tool at your disposal: your finely honed engineering sensibilities! Take a look through the practical examples, and then try eyeballing some familiar code in your codebase; there's a good probability you'll turn up issues without the help of any additional fancy analysis tools.

PRACTICAL EXAMPLES

All the examples below are based on real-world issues I've come across in

various codebases. Many of the code examples I provide are somewhat contrived, but this is to get the idea across in as concise a way as possible, not because you have to write contrived code samples to generate LHS issues!

TYPE CASTING

» Type casting is the most frequently cited example of how to generate a load-hit-store issue easily on a PowerPC core. PowerPC is unable to move data directly between different register types (from integer to floating point, say, or from floating point to VMX). Instead, the data has to be written from one type of register to memory and subsequently read back from memory to a register of the destination type. Yikes—LHS! Remember, this applies to implicit casts just the same as for explicit casts (see Listing 1).

The easiest way to avoid this class of LHS penalty is to think carefully about the type of your variables and try to avoid the sorts of casts that will require data to take a round trip via memory.

In some cases, it can be advantageous to keep two representations of the same piece of data around if it will allow you to avoid casts (see Listing 2).

HAMMERING MEMBER VALUES

» It's all too easy to forget when you write something like this ...

```
while ( ++m_opCount < limit )
{
    DoSimpleOperation();
}
```

... that what the compiler “sees” is:

```
while ( ( this->m_opCount = this->m_opCount + 1 ) < limit )
{
    DoSimpleOperation();
}
```

The pointer dereference should be a clue here. In almost all cases, the compiler will shy away from loading an object's member variable values to temporary registers for a number of discreet operations prior to storing them back “into” the object. Instead, the compiler will just update the value where it sits in memory. Thus, you end up reading back a value on loop iteration n that you only recently wrote to memory in loop iteration $(n-1)$.

In our example, if `DoSimpleOperation()` completes in relatively few cycles, the chances are good that on successive loop iterations you will suffer LHS stall penalties since `m_opCount` is being read from memory so soon after being written.

The simple solution for many of these cases is to briefly decouple the member variable into a locally instantiated variable that the compiler *can* keep in a register.

```
int localOpCount = m_opCount;
while ( ++localOpCount < limit )
{
    DoSimpleOperation();
}
m_opCount = localOpCount;
```




LISTING 1

```
int i_dist = GetDistance();
float f_dist = 12.0f * i_dist; // Implicit cast of i_dist int->float (type promotion)

std r11, 50h(r1)    // Store our integer value i_dist off to the stack
lfd fr0, 50h(r1)   // Reload value directly back from the stack into a float register
fcfid fr13, fr0    // Convert from integer to floating representation - LHS!
```

LISTING 2

```
for ( int i_index = 0; i_index < 100; ++i_index )
{
    float f_dist = 12.0f * i_index; // Implicit cast - LHS!
    DoSomething( f_dist );
}
```

This code could change slightly and avoid the LHS:

```
for ( int i_index = 0, f_index = 0.0f; i_index < 100; ++i_index, f_index += 1.0f )
{
    float f_dist = 12.0f * f_index; // No cast - no LHS!
    DoSomething( f_dist );
}
```

LISTING 3

```
void LHS_NightmareAverted( float* __restrict p_ptr0, float* __restrict p_ptr1 )
{
    *p_ptr0 = 5;
    *p_ptr1 = 10;
    *p_ptr0 += 7;
}
```

```
*p_ptr0 = 5;
*p_ptr1 = 10;
li r11,105.7 // Load immediate value 10

*p_ptr0 += 7;
li r10,125.7 // Load immediate value 12 (aka 5 + 7)
stw r11,0(r4) // Store value 10 to p_ptr1
stw r10,0(r3) // Store value 12 to p_ptr0
```

LISTING 4

```
// Explicitly storing some general purpose register (GPR) values to the stack
stw r12,-8(r1) // Link register
std r30,-18h(r1) // Integer register 30
std r31,-10h(r1) // Integer register 31
stfd fr31,-20h(r1) // Float register 31

// Using a utility function to store Link register and GPRs 28 and above
bl savegpr28lr
```

BAD PARAMETER CHOICES

» It's a good idea to have a solid understanding of the calling convention for your target platform. This will help you determine in which situations the compiler will use registers to transfer parameter data between functions, and in which situations it will be forced to use memory for this transfer.

```
float f_health = 3.5f;
float f_healthAdjusted;

// Compute adjusted health into f_healthAdjusted.
DoHealthCalculation( f_health, &f_healthAdjusted );
if ( f_healthAdjusted > 2.5f )
// ...
```

From the Xbox 360 and PlayStation 3 PPE ABIs, we know that simple float parameters will be passed as registers and that float return values will also be returned in a register. What about the float pointer? The pointer value will be passed in an integer register, but `DoHealthCalculation()` is clearly required to use that value to store the computed result to a location in memory. By now, it should be clear that when we try to use `f_healthAdjusted` once the health calculation function has completed, we'll be loading from memory and making use of a value that was only very recently written.

Knowledge of the calling convention informs us that making a simple change to `DoHealthCalculation()` to return the computed value rather than store it to a provided address will eliminate the LHS condition here:

```
f_healthAdjusted = DoHealthCalculation( f_health );
```

POINTER ALIASING

» Pointer aliasing is a subject worthy of its own article. However, a simple example can illustrate how hamstringing the compiler can become at avoiding LHS unless you help it out.

```
void LHS_Nightmare( float* p_ptr0, float* p_ptr1 )
{
    *p_ptr0 = 5;
    *p_ptr1 = 10;
    *p_ptr0 += 7;
}
```

Okay, so where's the LHS here? Let's look at the disassembly to see what's going on:

```
*p_ptr0 = 5;
li   r11,5 // Load immediate value 5
stw  r11,0(r3) // Store value 5 to p_ptr0

*p_ptr1 = 10;
li   r10,10 // Load immediate value 10
stw  r10,0(r4) // Store value 10 to p_ptr1

*p_ptr0 += 7;
lwz  r11,0(r3) // Read back contents of p_ptr0
addi r9,r11,7 // Add 7 to contents - LHS!
stw  r9,0(r3) // Store sum back to p_ptr0
```

At first glance, it's hard to understand why the compiler is bothering to read back the value recently stored in memory at `p_ptr0` just to add another 7 to it and store it back to `p_ptr0` again.

But from the compiler's point of view, there is no alternative. There is nothing that says `p_ptr0` and `p_ptr1` can't point to the same spot in memory.

(Pointer aliasing can loosely be described as just this situation—when two or more pointers point to the same memory location, those pointers are said to alias each other.)

If it's possible that the pointers do point to the same location, you can see why the compiler has to reload the value stored to `p_ptr0`. It has no idea whether the store to `p_ptr1` was *also* a store to `p_ptr0`, so the only option is to reload and be safe.

What we need is a way to tell the compiler that the pointers won't ever alias. Fortunately, we have a mechanism to do this: the `restrict` keyword (see Listing 3).

In this modified code, armed with the additional knowledge that the pointers don't alias, the compiler is smart enough to just generate two immediate value stores.

It's important to recognize that the `restrict` keyword is not something that the compiler can sanity check. It is often described as a "promise" that you make to the compiler. You promise that the pointers won't alias and, in return, the compiler can perform instruction re-ordering and optimization that it couldn't otherwise do. If you break that promise, nasty things are very likely to happen.

ARRAY ALLOCATIONS ON THE STACK

» Even smart compilers seem to be very bad at optimizing locally instantiated small array use into discrete registers. Instead, everything goes via the stack, so reading and writing these array elements can lead to a large amount of LHS stall penalties.

```
enum locations
{
    INSIDE = 0,
    OUTSIDE,
    INTERSECTING,
    MAX_LOCATION
};

int locationCount[MAX_LOCATION] = { 0 };
for ( int i = 0; i < numberOfPointsToTest; ++i )
{
    int location = INTERSECTING;
    if ( PointIsOutside( point[i] ) )
    {
        location = OUTSIDE;
    }
    else if ( PointIsInside( point[i] ) )
    {
        location = INSIDE;
    }
    locationCount[location]++;
}
```

Clearly we are interested in tracking three independent values: how many points are inside, how many are outside, and how many intersect. Choosing to use a locally instantiated array to track this is asking for trouble. The compiler is more than likely to keep that array on the stack, which leads to problems with this line:

```
locationCount[location]++;
```

You can see that this translates to the sequence "read value from memory that was only recently written, increment the value and write the value back to memory" for each successive loop iteration.

In this case, the code would benefit from being re-written thusly:



```
int locationInside = 0;
int locationOutside = 0;
int locationIntersecting = 0;
for ( int i = 0; i < numberOfPointsToTest; ++i )
{
    if ( PointIsOutside( point[i] ) )
    {
        ++locationOutside;
    }
    else if ( PointIsInside( point[i] ) )
    {
        ++locationInside;
    }
    else
    {
        ++locationIntersecting;
    }
}
```

Having discrete integer variables rather than an array is far more likely to persuade the compiler to keep those values in-register, bypassing writes and reads from memory, and eliminating the possibility for LHS.

STACK FRAME TEARDOWN

» When you enter a function, the compiler will often choose to store off to the stack some combination of the general-purpose integer and float registers, freeing them up for use within the body of the function. Sometimes you can see this happening explicitly, register by register, at the start of your function, and sometimes the compiler will make use of a utility function to perform the stores (see Listing 4).

When you *exit* the function, those values that were stored off need to be read back from the stack and placed into the correct registers, so the general register state on exit is the same as on entry.

If you're not spending much time in the function, you can see that there's a chance of encountering LHS during the stack frame teardown (or more correctly, subsequent to the teardown, when you come to use any of those values that were recently read back).

In many cases, this form of LHS can be hard to avoid. The one thing to consider, particularly when the called function is small, is more aggressive inlining. If the function is inlined, you'll be removing the need for the additional stack setup and teardown.

CASCADING LHS

» In some situations, you just can't eliminate LHS entirely. What you can do is use your knowledge of the mechanism of LHS to minimize the impact.

```
float LHS_Cascade( const int* p_ptr )
{
    static const float weights[] = { 0.1f, 0.2f, 0.25f, 0.45f };
    float sum = 0.0f;
    for ( int i = 0; i < 4; ++i )
    {
        float f = static_cast<float>( p_ptr[i] );
        sum += f * weights[i]; // LHS!
    }
    return sum;
}
```

```
int d[4] = { 12, 45, 34, -17 };
float sum = LHS_Cascade( d );
```

Okay, we already know that `float` to `int` casting is a terrible thing, but let's

assume in the situation above that there just isn't any alternative, and also that the compiler hasn't gone ahead and unrolled that loop automatically. Do we have a means to reduce the impact the LHS might have? We know the loop will iterate four times, and on each loop iteration, we are going to encounter an `int` to `float` cast and a subsequent operation which will incur a LHS penalty. Sum total LHS penalty is therefore 4x (single LHS penalty).

How about this approach, essentially an unrolled version of the simple loop:

```
float LHS_Cascade( const int* p_ptr )
{
    static const float weights[] = { 0.1f, 0.2f, 0.25f, 0.45f };

    float f0 = static_cast<float>( p_ptr[0] );
    float f1 = static_cast<float>( p_ptr[1] );
    float f2 = static_cast<float>( p_ptr[2] );
    float f3 = static_cast<float>( p_ptr[3] );

    float sum = f0 * weights[0]; // LHS!
    sum += f1 * weights[1];
    sum += f2 * weights[2];
    sum += f3 * weights[3];

    return sum;
}
```

We're going to incur an LHS penalty when we use a cast value for the first time, but we know that penalty will cause the thread to flush for sufficient time so that our remaining cast values should be safe to use without penalty. Sum total LHS penalty here is therefore 1x (single LHS penalty)—or a mere 25 percent of the cost of the original code!

"FALSE" LOAD-HIT-STORES

» I've included mention of false LHS here for completeness. There's really very little you can do about this issue except in particularly contrived examples—if this is your remaining LHS concern, you should be sleeping soundly at night!

False LHS occurs because the hardware that checks for LHS occurring is lazy! Rather than checking the full address from which data was read, it checks only the low *n* bits. On some CPUs, for example, only the low 12 bits are checked; so if you read from an address 4k away from an address that was just written, you will suffer the LHS penalty despite the fact that there was no "true" collision.

HIT THE LOAD, JACK

» I hope the practical examples above have given you a sense of the potential that LHS behavior has to infest your codebase (and the potential for easy optimization that awaits the enthusiastic engineer).

Although LHS penalties can be considered a low-level issue, they are often easy to avoid using simple, high-level changes to your code in tandem with a broad appreciation for the underlying mechanism. What's not to love about simple optimizations, especially when they can improve performance on more than one target platform?

One final word of caution: I think it's important to remember that fixing LHS is not a panacea for poorly performing code. Agonizing over every last LHS penalty is no substitute for proper algorithmic optimization or L2 cache management or, in many cases, deciding that function `foo()` occupies such a tiny fraction of your frame that time spent on optimization might be better spent on making a nice strong pot of coffee. ☕

DAVE COWLING has been coding games professionally for 17 years. He is currently the studio technical director for Neversoft Entertainment. At his age, effective optimization is one of a dwindling number of fun activities that doesn't leave him feeling sore the next morning.



BLINK OF AN EYE

USING FILM EDITING TECHNIQUES OUTSIDE OF CINEMATICS

GDGS COME AND GO, BRINGING WITH THEM A NEW CROP OF TALKS BY VISITING HOLLYWOOD LUMINARIES—AND, inevitably, mixed feelings from the developers in the audience. Whether it's fanboy enthusiasm or defensive grumbling, our industry can never quite free itself from the coils of sibling rivalry when it comes to the movie business.

Of course, that relationship has changed over the years. When Pixel Pusher first pondered the perilous partnership of film and games [see "Beg, Borrow, Steal," *Game Developer*, May 2004], we noted how deftly the first incarnations of the CALL OF DUTY franchise borrowed cinematic conventions in our low-fidelity medium. Fast forward seven years, and the sophistication of game cinematics has become truly jaw-dropping, even for those of us who work in the business. Pat yourselves on the back, gamers, it's been a long haul from the goofy 8-bit slideshows of ZERO WING to the moody psychodrama of DEAD SPACE 2.

We've assimilated a lot of the production values of cinema, but we haven't always made them truly our own. We've become very adept at mimicking the high style of Hollywood within the confines of fixed-camera cinematics. Once the black bars retreat and the player takes over, though, we don't always show the same degree of finesse. So this month, we're going to take a quick look at some of the ways in which basic cinematic techniques can work in an interactive setting. Much of this will be familiar to animators and cinema directors, but there's a lot that environment artists and level designers can learn from the Hollywood editors' book of tricks.

CAHIERS DU CINÉMA

» Film school grads and movie buffs divide the world into two main camps with impressive French names (remember to learn these, you can irritate your friends by dropping them at parties). "Mise-en-scène" is, literally, the "stuff in the scene." In a movie, that's the actors, sets, and action of the film. In our world, it's "content"—the actual stuff of the game world. The other half of the film vocabulary is "montage"—the assembly of images that connect the individual scenes into a larger narrative. In other words, the editing.

Effective storytelling requires both content and edits. The best looking models and animation can't grab

an audience effectively without good camera work and well-timed cuts. Until recently, game design orthodoxy has tended to keep the player's point of view locked into a single "shot" for long stretches of time. Though there are a few notable exceptions (ALONE IN THE DARK, for example, and later the RESIDENT EVIL series), games generally avoid sudden cuts while the player is in control. Game cameras don't always behave in any case, and sudden jumps in perspective can easily confuse players [See Figure 1].

In the last few years, though, many games have started bridling at the tyranny of the fixed camera. As open world games have become popular, the need for cameras to switch between vehicular and foot travel has accustomed players to quick transitions in perspective. Brawlers and melee-based adventure games are increasingly willing to enhance the impact of big moves with zooms and cuts, and in the



FIGURE 1 The RESIDENT EVIL series (above and far right) is the most famous early example of a game that tried to mimic cinematic camerawork in an interactive game.

post-GEARS OF WAR world, even shooters have begun to loosen up their points of view. Most importantly, games of every genre routinely include short vignettes where the camera focuses on an important landmark or a scripted event—moments which are "cinematic" in their use of camera and cuts but which don't necessarily involve animation or dialog. With all this going on, basic familiarity with the language and conventions of film editing is good for level artists and animators, as well as for the cinematics team.





ILLUSTRATION BY JUAN RAMIREZ



FIGURE 2 The conventions of editing make it easy for us to interpret these separate images as a continuous story. Although this sequence compresses time and space it makes intuitive sense—at least to audiences who have been trained by years of movie watching.

FILM THEORY 101

>> The basic job of a montage is to break the flow of time. When the camera is rolling, time unfolds naturally at 24 frames per second (don't patronize us, Angelenos, we're solid at sixty hertz! Nyah nyah). A sudden cut to another camera allows the editor to break the flow of time and space in the service of the story. Consider this classic film-school example: Two cops are riding in a car, talking to each other. Then (a second shot) we see the same car pull up in front of a building and the cops get out. Next (a third shot), we see them knocking on an apartment door. Finally (a fourth shot), we see somebody answer the knock and open the door from an angle that lets us see both the cops and the interviewee (See Figure 2).

We've all seen variations on this sequence hundreds of times—so many times that it's easy to miss how much is going on in those simple cuts. The most obvious effect is time compression. In "real life," the trip from the station to the suspect's apartment is probably twenty minutes, rather than twenty seconds long. Likewise, parking the car and walking up a few flights of stairs would take long enough to bore most audiences. Of course, we're also expected to know that almost no time passes between the knock and the opening door. The edit breaks the flow of time and asks the viewer to fill in the gaps.

This seems entirely natural to us, as heirs to a century of cinematic tradition. It's not "natural" at all, though. It's a learned set of conventions. In many parts of the world, it took several years for new audiences to assimilate the way films casually violate the ordinary laws of time and space. I once knew a woman who used to drive a truck around rural India, showing movies projected on the sides of buildings in remote villages. She recounted having to constantly stop the film to explain to her audiences how to understand the sudden leaps from place to place and time to time.

Interestingly, some types of edits are much easier even for completely untrained audiences. Apparently even first-time film viewers grasp the typical "three camera" dialog setup (alternating close-ups of two actors conversing with longer shots of both of them seen from farther away). This seems to be because the actors' faces help teach the audience what to expect. As each actor delivers a line and looks to the other for a reaction, their expression is telling the viewer "the next important thing is over here, where I'm looking." The cut, rather than surprising the viewer, actually fulfills the audience's subliminal need to see "what's next." This is how first time viewers manage to grasp it so easily.

The key lesson here is that cuts can't just happen for no reason (unless you're Michael Bay). To justify breaking the continuity of the shot, you need to prepare the audience for a change. This is even truer for us, since we have to preserve the player's sense of agency and control.

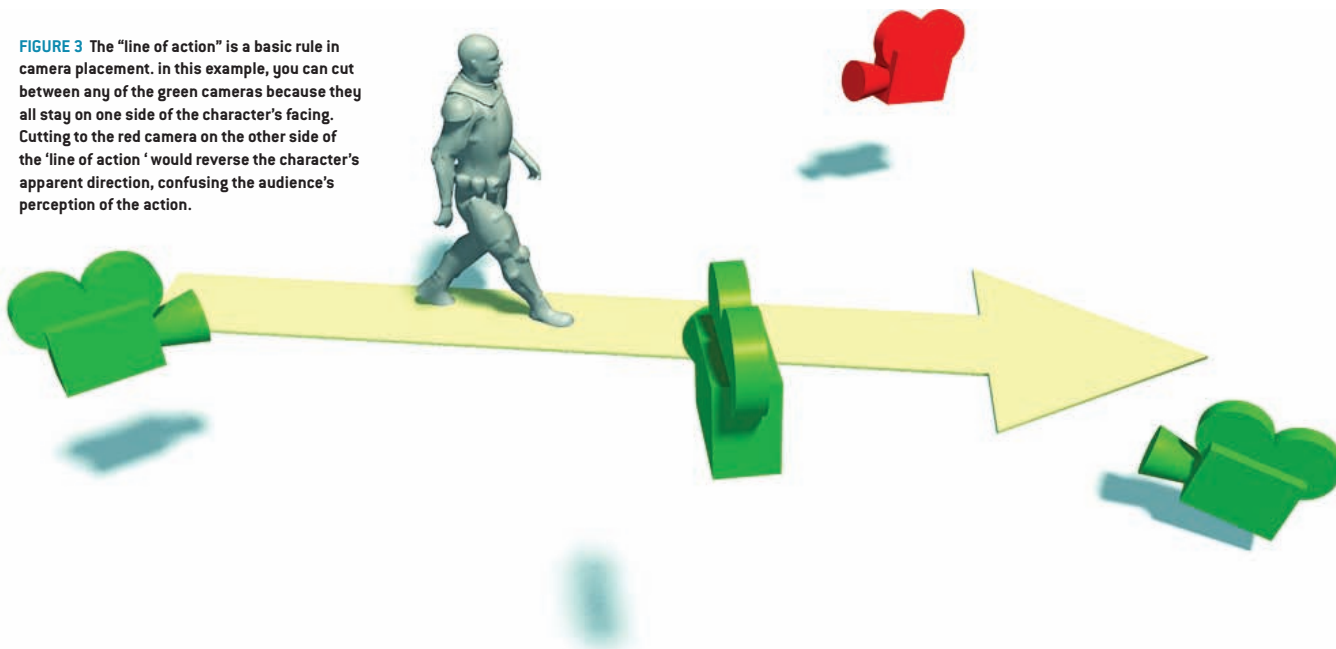
Roy Thompson's indispensable book *Grammar of the Edit* catalogs criteria that film editors use to weave coherent stories out of sudden jumps in time and space. It's a great introduction to a complex subject that also offers a lot of good advice to game artists, whether we're storyboarding animations or setting up camera vignettes in a game level. While the book is far too rich to sum up neatly, here are a couple of the highlights.

INFORMATION

>> Thompson's first, and most important, principle is the question: Does a cut add something to the viewer's understanding of the scene? In film terms, this is usually a plot point: a close-up of a newspaper headline, a close-up of an antagonist in a crowd, or the ominous opening of a creaky door. For level artists, the information in a camera cut might also be simple navigational information ("Hey, that lighted door at the top of the catwalk must be the exit from this level") or tactical knowledge ("Hmm, why in the world is the camera focusing in on those explosive barrels?"). Of course, vignettes are also a good way to reveal plot points without fully animated cinematics. A well-timed and framed close up of an important clue can tell a lot of story much more cost-effectively than a fully animated cutscene can.

The flip side of this idea is to use cuts sparingly. Many level artists are tempted to use vignettes just to show off their work—it's hard to labor for

FIGURE 3 The “line of action” is a basic rule in camera placement. In this example, you can cut between any of the green cameras because they all stay on one side of the character’s facing. Cutting to the red camera on the other side of the ‘line of action’ would reverse the character’s apparent direction, confusing the audience’s perception of the action.



weeks on a beautiful panorama only to have your players glue their eyes only to ground cover and exit routes. That’s understandable, but it’s a mistake; unless the cut really conveys important information, it’s not worth the cost of pulling the player out of continuity.

MOTIVATION

» Thompson’s second rule is that every cut requires a “motivation”—a visual or aural cue that explains to the viewer why the continuity is broken. For example, when an actor hears a noise off-screen and turns his head, the audience already expects a cut that shows the source of the noise. Without that motivation, the cut is jarring; the audience may assume it represents a lapse of time or shift in scene.

Motivated edits are the key to traditional Hollywood camerawork. By preparing the audience for the cuts before they happen, the cognitive cost of all those shifts in perspective is lowered. Good motivation is equally important for interactive vignettes. In fact, it’s even more necessary, because a game cut that disrupts the player’s interactive control is more jarring than a movie edit.

Unfortunately, games don’t always pay enough attention to the need to prepare the audience for cuts before they happen. We’ve all seen games that cut to a new camera setup when the player enters a new room. Unfortunately, we’ve also seen how often these kinds of cuts are triggered by nothing but a character inching over the boundary of a trigger volume. Think back to some of the early *RESIDENT EVILS* if you really want to understand the need for motivated camera cuts!

The key to selling these kinds of transitions is to capture the physicality of the motivation before actually moving the camera. Ideally, the

player character should actually do something that motivates the cut. Reaching for a door knob, pushing a button, or turning to face a new threat are all great physical ways to prepare the audience for the break in continuity. It’s important not to forget the audio either; when the camera is going to reveal a new boss monster entering the fray, a battle cry or a music cue that hits just before the actual cut will prep the audience for the new revelation.

SPATIAL SENSE

» One important motivation that games can offer which films cannot is player action. The act of pressing a button—say to switch between views in a driving game or using the hat switch in a flight sim—has a low “cost” for the viewer because it’s intended to cause a cut. At the same time, even intentional cuts can frustrate the player’s intuitions about control, as veterans of games as different as *METAL GEAR SOLID*, *RED DEAD REDEMPTION*, and *MADDEN 2010* can all attest. No matter what you want to do with the camera, you can’t confuse the players’ thumbs.

Film editors obviously don’t have this problem, but they also have to worry about the audience’s spatial sense. They use a principle known as the “line of action,” which is basically the major axis of a particular sequence. They try to always keep the camera on one side of that line [See Figure 3].

For example, when you film two armies marching into battle, you never allow them to switch sides. Purists will even cheat the shots that are right along the line of action just a hair so that even an over the shoulder shot of the left-hand army is moving slightly to the right on screen and vice versa.

The line of action is an important tool for keeping audiences clear about the physical space

in which the action unfolds. Our players generally have a better spatial sense than movie audiences, since they’re navigating 3D spaces under their own power. However, they still have years of cinematic training under their belts, through watching movies. If you violate the line of action, you’ll disorient players. So, if your player approaches the door moving from left to right across the screen, the reveal shot that triggers when the door opens should respect that line of action.

FINAL CUT

» Editing is the subtlest of all the cinematic arts. As Thompson says, the best cut is one so correct that it doesn’t feel like a cut at all. Though games aren’t always known for subtlety, a lot has changed in the last few years. Just compare gameplay footage from say, *GEARS 2* with an old-school TPS from 2003, and you can see how much more “cinematic” gameplay is becoming. While the vexing problems of control response aren’t going to go away overnight, we have made remarkable progress. And now that our cinematics are so freaking fabulous, it’s especially important for level art and animation to push the quality of our interactive experiences equally far. In the process, we’ll probably learn a few tricks we can pass back to our Hollywood cousins to repay them for all the wisdom we’ve borrowed over the years... although, seriously, after *Transformers 2*, I don’t really feel like I owe them anything. 🎬

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

SEEING MUTANT ZOMBIES
THROUGH THE DARK IS GREAT.



SEEING VIDEO GAME TRENDS
BEFORE ANYONE ELSE IS BETTER.

GameSpot Trax reads the pulse of the gaming community for data to help predict real world outcomes. Get reliable key performance indicators that help you see in the dark and make better decisions. Email us at trax@gamespot.com.



GAMESPOT
TRAX™



THREE ERRONEOUS CONCEITS

RETHINKING GAME AUDIO

THERE IS A CURIOUS KIND of memory that we all possess. I'm talking about the long-term memory that accumulates and exists beyond the life of a single individual. Carl Jung called it The Collective Unconscious. In animals, we call it instinctive muscle memory, and it's what drives monarch butterflies to the same mating grounds generation after generation. In business, this collective memory is the force that spreads common practices and unquestioned universal truths from one professional to the next.

Unfortunately, universal truths and long-standing best practices are sometimes simply obvious options that are in need of reevaluations. Game audio implementation has a number of these conceits which have passed from company to company through the migratory nature of audio professionals, and they are ripe for rethinking.

FOOTFAILS

» The first erroneous conceit is the notion that footsteps are important. As far back as PAC-MAN's wakka-wakka, we established the notion that a character in movement must call attention to itself—and loudly. But as games have matured and increased exponentially in their complexity, we have clung to this notion that all player-related sounds are somehow equally important.

Unless stealth is a specific game mechanic, most game protagonists jog or run through the entirety of their journey. From a purely logical standpoint, running



is the loudest of human locomotion, especially when wearing combat or moon boots, as many of our characters do. However, running—except during exercise—is a symptom of something greater. People and game characters run out of emotion, whether it's fear or anger or urgency. When under the extreme circumstances of a fight-or-flight response (which a vast majority of game interactions are), the brain is focused on threats and goals. Therefore, the oncoming footfalls of enemies are important information regarding changes to a threatening environment. The player's footsteps, on the other hand, are comparatively unimportant information.

As loud as we usually mix them, this creates an odd cognitive dissonance. Non-threatening footsteps—those of players and companion characters—might be

loudest to a microphone attached to the game's camera in that situation, but a game's microphone feeds information to the player's brain, which wants to ignore superfluous noise and focus on danger. As a result, implementors can treat footsteps like other gameplay-critical data and set volume ranges differently for Foley created by threatening and non-threatening characters.

WAIT FOR IT

» The second conceit is that implementation timing is tied to level designer-authored events. It's an all-too-common practice from gameplay designers to set scripted events throughout their levels, while audio implementors simply piggyback off of these events. Unfortunately, while this is easy and fast, the end results often feel unrealistic and "gamey."

It's a common scene: the player is on a path through the level when something catastrophic explodes/sinks/crashes and they're forced to change their plan of attack. As the large event is occurring, in-ear dialogue is already chattering away, directing the player with new information. There are two major problems with this.

First, these events are frequently audio showpieces. Lots of work goes into designing satisfying explosions and twisting-metal sounds. Unfortunately, the in-ear dialogue that accompanies these scenarios is often gameplay-critical voice. Since gameplay-critical voice trumps all other sound and therefore often triggers a dynamic ducking system, the voice turns down everything else at its expense. This means that the audio showpiece that took critical design and implementation time to create is ducked and lost beneath dialogue which is not tied via lip-sync to any of the visuals.

Secondly, the character giving the in-ear re-direction comes across as unnaturally omniscient. They seem to know everything as it occurs, instead of assessing and reacting to situations as believable storytelling demands. The player's attention is divided in these moments. Before they have even processed the changing landscape of their goals, their primary source of reliable information is drawing their attention elsewhere.

The solution is as simple as wait timers. By using delays as part of the scripted audio triggers, audio

is given space to breathe, impress, and add drama and pacing to showcase moments that should be memorable and exciting.

ACTION DETRACTION

» The final of the three erroneous conceits is that action sequences require action music. Many games continue to use music as dramatic wallpaper. It occurs in big sheets, covering wide areas of gameplay with an ever-increasing amount of interactive music. But look at some of the classic action sequences from cinema, like the speeder bike chase from *Return of the Jedi* or the car chases from *Bullitt* and *Ronin*. What's instantly clear is that these are sequences where sound design is given center stage.

In these moments, music frequently ducks out under the first squeal of tires while the sound effects are cranked in the mix. The punch of fists, the crunch of metal, and the whoomp of explosions can be used to create a soundtrack that's just as dynamic with the orchestration of action-oriented effects. Consider alternate mix groups that boost elements of the sound effects beyond their normal settings or a design approach that focuses on the impression that the player's character and their vehicle are being pushed to the breaking point. Music is an easy way to add a driving sense of urgency, but it's not the only solution. 🎧

JESSE HARLIN has been composing music for games since 1999. He is currently the staff composer for LucasArts.



A WAR OF ATTRITION

APP STORE FATIGUE AND THE FUTURE OF DIGITAL MARKETPLACES



COUNTLESS WORDS HAVE BEEN DEVOTED TO THE SUBJECT OF digital distribution by game industry publications. What does it mean for game developers and for the medium as a whole? Like any other disruptive innovation, the advent of digital distribution has come with some growing pains, as well as its share of winners and losers.

On the whole, digital distribution has been a boon for gaming. It's opened up channels for indie developers and those serving niche market fare, enabled new business models, and increased the efficiency of the supply chain, all while providing gamers with convenient access to a wider variety of games across a broader mix of devices and platforms.

The current quintessence of digital distribution is Apple's App Store, which, together with the iPhone and iPad devices it services, have changed the face of gaming remarkably since their introduction less than three years ago. As a result, iPhone and iPad comprise a game market of significance, with game-derived revenue likely to soon cross the billion-dollar mark. While this is still smaller than the traditional console markets, when viewed from the standpoint of potential profit margin (taking development costs into account), it's no wonder the App Store is the darling of developers.

Apple's success has not gone unnoticed by makers of other platforms and services. At CES this year, I counted no less than twenty different "App Store"-like storefront services across countless devices and platforms. There are the smartphone platforms from Microsoft, Motorola, Google, and others; tablets from a large number of companies; smartTVs with

the capability to run games—all are offering their own app stores. If this weren't confusing enough, add to the mix the aggregators with their catalog-within-the-store type offerings, as well as support within these platforms for remote-gaming approaches from the likes of Onlive and Vudu. It's started to feel awful crowded.

APP STORE FATIGUE

» The first assertion I'll make is that I don't believe consumers are going to have the trust, patience, or incentive to maintain an ongoing commercial relationship with every electronic device in their household. Even so, it appears that many of those devices are going to go ahead and ask for one anyway.

As a result, there are likely to be winners and losers, as consumers suffer app store fatigue and forgo some stores and services in favor of others. (You can already see a hint of this today: Many readers may already have two or more boxes in their living room that have a Netflix client, though chances are they favor one of these for everyday use.)

The successful app stores are going to be those that best address a number of factors, including offering a critical mass of games, how well suited and/or differentiated the applications are to the platform, how well the store helps users navigate which apps meet their needs and support their device, and which offer the fewest hurdles between them and the cash register (price, ways to pay, simplicity of purchase, and so forth).

Failed app stores are likely to either become either ghost towns (with

a lack of a critical mass of offerings) or flea markets (with tons of generic apps, but few that are noteworthy). These in turn may be less able to court developers. Be wary of those touting an installed base of platforms, but not sales numbers for their app stores.

Those launching app stores will have to think about how to get that customer in the door for the first time, and then how to keep them. I anticipate seeing lots of “Comes bundled with \$50 worth of App StoreBux” offers.

App store fatigue will likely plague developers, too, who will be faced with a huge choice of channels. Even with relatively lightweight click-to-accept agreements, most will not have the bandwidth to launch on every platform and customize for services/devices. This is already occurring, and in some ways is playing out exactly as the PC download casual games market did a few years ago. As it did then, it will likely lead to the following:

∞ Multi-platform development may mean developing for the lowest common denominator, allowing for titles to go to a wider base of platforms, but decreasing how well-tuned a title is to any given platform.

∞ There may be a rise of content aggregator/distributors telling developers “sign with me and I’ll get you on a thousand platforms,” at the expense of a good chunk of developer margin, putting pressure on the critical mass needed for viability.

∞ Many developers will sit on the sidelines for all but the proven app stores. Those duped into thinking that a platform will be big, if it then isn’t, will potentially complain loudly about it. Unsuccessful app stores have a “content canyon,” with a few mega-hits that are everywhere, and a glut of low-end fare, but nothing in between.

Ultimately, competition is good, though it can be painful. As this competition heats up, gamers and developers will both be watching intently and deciding which platforms and app stores are actually going to be successful. One of the deciding factors here will be developer tools, though not the kind we usually think of.

CRY HAVOC, AND LET SLIP THE DOGS OF ... COMMERCE?

» In his 2010 GDC keynote, NGMoco’s Neal Young pointed out his view of what it means to couple business models like free-to-play with tools like in-app purchases. He stated “For the first time, the business model is in the hands of the game

designers,” adding that this was a fundamental turning point for the industry. It was a prescient observation, and at the time, I thought it was limited to free-to-play, item-based sales business models. In truth, it’s much more broad-reaching.

As app stores compete to woo consumers into a relationship and developers to be their first choice, the winners are going to, in part, be determined by who can offer the best value for consumers while offering developers the greatest ability to extract maximum revenue out of their consumer base. If those seem like conflicting objectives, well, in some respects they are.

As app stores compete to woo consumers into a relationship and developers to be their first choice, the winners are going to, in part, be determined by who can offer the best value for consumers while offering developers the greatest ability to extract maximum revenue out of their consumer base.

Finding the optimal point (or points) between these two goals is going to require the ability to innovate, and doing so requires the ability to experiment. The tools and policies in place for online commerce in these app stores are going to determine the variety and rate of experimentation. For example, can developers try any price point, or are there fixed tiers? Can they set the price, or does the platform owner? Can one do bundles, gifting, giveaways, “golden tickets,” or any other type of promotion? Free with three cereal box tops? How about pre-orders? A/B testing? Platforms will have to balance giving developers the leeway to innovate versus presenting consumers with a confusing mess of business models, offers, and potentially the introduction of risk.

In addition to the tools giving developers the flexibility to innovate, there is an entire category of tools and services to measure the success of those experiments. Billing and reporting tools offer transparency, granularity, and timeliness of the data. What detail can a developer get on sales data? How fresh or stale is the data? What will the store’s position be on the tension between user privacy versus providing developers demographic information? What of their approach to merchandising? Is “shelf space” curated, democratized, or sold to the highest bidder? What payment methods are supported? Can developers see sales data per territory? Over time? Per day? Per hour?

These are the tools with which developers will compete, and platforms that give their developers better tools will have an advantage over those that do not. A complete free-for-all has an adverse effect on the end-user experience, so there’s a balance to strike. Still, one only has to look at the speed at which new business models

and promotional experiments are emerging on Apple’s platforms and compare it to the snail’s pace at which traditional consoles have taken in similar experimentation to see the contrast.

STANDING OUT IN A CROWD(ED PLATFORM)

» A final element is the degree to which developers can customize and tune their applications to take advantage of platform differentiators. The extent to which apps take advantage of differentiators is going to lead to a fulfilled or broken promise to the end-user from the device manufacturer. For

example, if someone offered a tablet today, and differentiated versus iPad by offering dual thumbsticks in addition to a touchscreen, it might be of interest to some gamers. However, if all the games in the device’s app store were ports from the iPhone/iPad, the takeaway might be “Yeah it has thumbsticks, but no games support them.” On the other hand, features targeting specific platforms that are push-marketed or store-filtered to users who have those platforms/features could encourage greater co-marketing opportunities for developers and device makers. The apps that best translate a hardware platform’s features into compelling experiences will be promoted by platform vendors.

PICK YOUR WEAPONS

» So what’s a developer to do? As usual, the answer is “it depends.” There is no one strategy to rule them all. However, it’s pretty clear that the platforms that provide developers with the ability to differentiate and innovate with their commerce will have an overall advantage. If, as a developer, you feel comfortable making that part of your game design and part of how you do business, then evaluate those tools and policies as part of your approach to the platform. If, on the other hand, you prefer to stick to making a game as a packaged piece of media that someone else markets and sells, then you may want to stick to traditional consoles or other platforms that aren’t moving as quickly in this direction. ☹

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



Heir Today, Gone Tomorrow

EX-RAVEN DESIGNER DISCUSSES HIS BIG MOVE TO BIOWARE

Manveer Heir got his start in the industry as an intern for BigHugeGames, but very quickly moved to a full-time programming position at Raven, where he remained for five years. He went on to lead and help build that company's design department, but eventually his interests diverged, and he left the company without a fallback plan. A few months later, he was at BioWare, now working as a senior level designer on MASS EFFECT 3.



Brandon Sheffield: After working at one game company for most of your professional life, how did you decide when it was really time to move on?

Manveer Heir: I realized one day that the needs of me as a creative and the needs of Raven as a business were no longer in-sync. Priorities shifted at Activision, which meant Raven was going to help Treyarch out with CALL OF DUTY: BLACK OPS. Personally, having just worked on a first-person shooter in WOLFENSTEIN for the first four years of my career, I was ready to do something completely different. CALL OF DUTY is an amazing franchise, but it wasn't necessarily the right game for me to work on next since I was burnt out on the genre.

I believe the most important thing in life is doing something you love and are passionate about. I got into this industry so that I could get to do what I love everyday. To work on something that I wasn't personally as enthused and passionate about would have been a disservice to my amazing coworkers at Raven, the developers at Treyarch, and the fans of CALL OF DUTY, as well as violate my own ideals. So it just felt like the right move to step aside and move on to the next chapter of my life. Clearly they didn't need me either, because everyone did a great job with BLACK OPS and it has earned a number of Game of the Year nods.

BS: Were you worried about leaving Raven during a time when the job market was constrained?

MH: I probably should have been, but I wasn't too worried. I decided to quit before having another job lined up, which most people would say is a big no-no (and I would agree with them). However, I was looking for very specific things in a job and decided that if I couldn't find that job,

then I would go indie, live in my parents' basement, and make something awesome with whatever money I could cobble together. There are enough indie game ideas in my head that I was looking forward to trying that route if needed. I thought doing that for a couple years would have scratched a nice itch, and that I could always go back to the AAA industry if I wanted. Fortunately, I found the right job that was the overall right fit for me, in the right type of city, so now I don't have to try to sneak girls into my parents' basement while they sleep.

BS: How did you feel about coming in on a project for which fans have extremely high expectations? Has that changed since you've actually been working on it?

MH: MASS EFFECT is one of my favorite current-generation games. I absolutely love the franchise, which is a big reason why I came here, so it's a bit daunting. A number of my friends basically told me, "Don't screw MASS EFFECT up," when I told them where I was going and what franchise I was working on. The truth is that I couldn't possibly mess the game up if I tried. There are far too many smart, talented people that would never let someone like me mess the game up. And they've been working on it for much longer than me. I'm just doing my small part to help make the game be as best it can be. Now that we've formally announced the game and had a teaser trailer go up, getting to read all the excited fans' comments really helps keep you enthused about going to work and realize you're a part of something really special. So, if anything, I'm even more enthusiastic and excited now that I've been here for a bit, because I've gotten a chance to see all the amazing stuff the team is working on that hasn't been revealed yet, and see how psyched the fans are for the game.

BS: How does compartmentalization work at BioWare? At Raven, your title was more generic, though Raven isn't traditionally a company of "designers." Now that you're a senior level designer specifically, does this change things for you?

MH: Within the design department of BioWare's MASS EFFECT, there is a writing team, cinematic design team, gameplay design team, and a level design team (all this across two studios, in Edmonton and Montreal). At Raven, while I was the lead designer, we were still only a team of twelve I think, mostly level designers. Raven came from the old QUAKE days where designer meant level builder. To answer your question, things are very different here, starting with just the sheer size and breadth of the design department.

The biggest thing it changes for me is responsibility, management, and day-to-day duties. Before, I was responsible for an entire department's performance. I had to make sure the department was on task, on schedule, and helping them improve day to day, while helping to guide the high-level vision of the game with the creative director. Most of my day was spent in meetings with the creative director and other department leads trying to fight different fires that came up. After that, it was evaluating work done by other designers and trying to give them valuable feedback to help them with their next iteration. In my last year at Raven, very little of my time was spent in the game actually building levels or writing code. That's basically the deal when you are a lead: you enable others to do a great job, not necessarily do the work yourself.

Now, I am much more hands-on and learning some new skills as a senior level designer. I haven't done pure level design professionally before, so I'm learning a lot of valuable skills in terms of how to build levels, how to handle technical constraints, and how to create high-quality scenarios on schedule and budget. I learn more each day, and I hope, that by the end of it all, I'm producing work that everyone feels is of the quality that the franchise deserves. I have a great network of talented designers around me to ask questions of and to help me learn and grow each day.

I still get to do some of the stuff I was used to doing as a lead. The real difference between a junior and senior developer in any department is that you expect a junior to need guidance and assistance the whole way while you expect a senior to seek out the answers and solve the problems on their own. So I still get to solve problems and improve processes, at a much smaller scale, but at the same time, I get to build content hands-on and learn every day. At the end of the day, I know that we will succeed or fail as a team and not individuals, so I try to do my part to not let the team down and help catch problems as they crop up during development.

BS: Not to peg you as “the race guy,” but you and I have talked about race in games a lot—how do you feel about racial portrayals in games where it’s abstracted by aliens, or when there’s a character creator, as in MASS EFFECT?

MH: If it keeps the discourse of the subject going, I'll gladly be known as “the race guy.” It's too important a topic to drop off the radar because it makes some people uncomfortable. This is the point in the interview where I state that my opinions are my own and do not represent the opinions of Electronic Arts or BioWare or minorities. I think, in general, games that are willing to discuss mature themes, such as race, in a fashion that gets people thinking are good. If that means they abstract the race discussion to alien races, instead of human races, then that is fine. I think part of the reason I like MASS EFFECT is that it deals with mature themes in a respectful, honest, and interesting way, but is still a AAA blockbuster. At the same time, I think there's still room for us to have games that specifically talk about race directly, instead of abstractly.


One of the games I was thinking about making when I was considering going indie would be what I would describe as *Do The Right Thing* in video game form. Not that I was trying to copy Spike Lee's seminal work, but rather, I wanted to handle the subject matter of race with the honesty and introspective lens that Lee did, but also take advantage of the fact that we are

an interactive, not linear, medium. I think there is room in this industry for someone to do that, and I think we'll have a really interesting, amazing experience if it's executed well. I still want to see and play that game. I hope developers in general will start trying to tell new types of stories and craft new types of experiences. It doesn't have to be revolutionary; incremental steps over time will get us there. Then, we'll have something totally new and different to talk about and discuss as a medium. So ultimately, any discussion of the topic is valuable, but I'll always want more, and from more sources.

BS: A Do the Right Thing-style approach sounds interesting, but how do you discuss race in an emergent rather than explicit way? Many attempts have been a bit heavy-handed, rather than discovery-oriented. How do you make sure it becomes more experiential?

MH: That's a great question and definitely the challenge. I think you need the player to experience the feeling, so that requires building a world that is hostile, but not hostile in the way we normally build our games. So if you had a game where you could build your own character, maybe certain color skinned characters would be treated differently by NPCs or charged more at stores (or even banned). Maybe certain factions and quests aren't available to you in a RPG. I think creating a world in which the player never quite fits in is difficult, but necessary, to sort of keep the player on edge about their role in the world; you never want the player to feel fully comfortable. You want them on edge, but in a very different way than a survival horror game.

There have been plenty of games that have let the player be multiple characters, so I could see a game where part of the game you are a minority dealing with whatever obstacles the game has and part of the game where you are a white man of stature, and all the barriers that exist for the other character are lifted (and likely traded for a different set of barriers). That juxtaposition in gameplay opportunities, whether it's where you can go, who you can talk to, actual game verbs you can perform, or the general demeanor of characters toward you could provide an interesting dynamic in a game. As you say, this is easy to do in a heavy-handed way, so execution is of importance.

At the end of the day, I think making our players think and consider more after playing a game is far more interesting than what normally occurs. I love when I play a game and I think back to some of the decisions I made and how that led to the outcome that I arrived at (RPGs are great at this). Exploring the depths of a topic as complex and challenging as race to me is exciting, as well as daunting, which is exactly why it is worth pursuing. 

who went where

Microsoft researcher Johnny Chung Lee, a major contributor to the development of the Kinect technology, has left the company to join a special Google team.

Former EA Redwood Shores CTO Tim Wilson and ex-Namco Bandai Games America VP of online development Robert Stevenson have joined cloud-based gaming company Gaikai.

Renowned alternate-reality game designer and researcher Jane McGonigal has signed on with Social Chocolate, a new company dedicated to creating social games based on psychology, neuroscience, and sociology.

ESA senior vice president Kenneth Doroshov, after working with the ESA to battle a California law that aims to place government control over video game sales, has departed from the industry trade group.

MOTOGP 09/10 developer Monumental Games announced the appointment of video game industry veteran Nick Wheelwright as CEO.

Rockstar London's studio head, Mark Washbrook, has resigned from the company after working with Rockstar since 2005.

Former Activision and Electronic Arts executive Kathy Vrabeck is leaving her latest position as head of Legendary Pictures' digital division, where since 2009 she's been responsible for activities including the film company's video game efforts.

Germany-based free-to-play browser game company Bigpoint announced it has hired on 37 artists and developers from Planet Moon Studios, whose talent joined Bigpoint's San Francisco development location.

Game industry veteran Rich Flier has left marketing company Secret Identity to join media production house Mothership as an executive producer.

new studios

Following the collapse of Midway, a number of former employees have formed Phosphor Games to develop AWAKENED, which hopes to build upon the now-defunct Midway title HERO.

Tales of the Future

GAME DEVELOPERS CONFERENCE REVEALS ALL-STAR CLASSIC GAME POSTMORTEM LINE-UP

ORGANIZERS OF GAME DEVELOPERS CONFERENCE 2011 HAVE REVEALED AN ALL-STAR LINE-UP OF GAME DEVELOPERS, FROM JOHN ROMERO (DOOM) THROUGH WILL WRIGHT (RAID ON BUNGELING BAY) TO TORU IWATANI (PAC-MAN) AND BEYOND, PRESENTING POSTMORTEMS ON THE MAKING OF SOME OF THE MOST FAMOUS VIDEO GAMES OF ALL TIME. THE SPECIAL ONE-OFF HOUR-LONG LECTURES ARE PART OF CELEBRATIONS FOR THE 25TH ITERATION OF GDC, AND WILL ALL TAKE PLACE DURING THE MAIN CONFERENCE OF THIS YEAR'S SHOW. THE FOLLOWING 11 SPECIAL LECTURES WILL TAKE PLACE AT GDC 2011:

Prince of Persia

JORDAN MECHNER

\\ Decades before it was a Hollywood film with tens of millions of dollars and hundreds of people working on its production, PRINCE OF PERSIA was mostly the project of a single man. Jordan Mechner rotoscoped the game's fluid and realistic character animations, designed its difficult puzzles, crafted its thrilling sword-fighting combat, and penned its captivating story. He will present a postmortem discussion on the landmark cinematic platformer.

Pac-Man

TORU IWATANI

\\ More than just the man who created and designed PAC-MAN, Toru Iwatani revolutionized an arcade industry filled with space shooters and PONG clones, introducing a new kind of game that was both immediately accessible and highly addictive. In this session, Iwatani will share how he created one of the world's most successful and beloved arcade games, centering around a circle with a wedge sliced off.

Elite

DAVID BRABEN

\\ When it launched over 25 years ago, ELITE amazed science fiction fans with its interstellar missions presented with wireframe 3D graphics, eight galaxies

to explore, and thousands of procedurally generated planets. Co-creator David Braben, a stalwart in the video game industry and founder/chairman of Frontier Developments, will discuss the genesis of the space-trading sim that went on to inspire titles like EVE ONLINE, FREELANCER, WING COMMANDER: PRIVATEER, and many other sci-fi epics.

Another World/Out Of This World

ERIC CHAHI

\\ Released across more than a dozen platforms since its 1991 debut, OUT OF THIS WORLD [a.k.a. ANOTHER WORLD] has long been a favorite among critics and sophisticated players, due to its cinematic cutscenes and atmospheric presentation. OUT OF THIS WORLD's creator Eric Chahi will reveal his process developing the innovative game and building its memorable scenes.

Marble Madness

MARK CERNY

\\ Mark Cerny is a legend in the games industry, working as a consultant, producer, and programmer on hits like RESISTANCE, RATCHET AND CLANK, JAK AND DAXTER, SPYRO THE DRAGON, and SONIC THE HEDGEHOG 2. Before building that near-incomparable resume, though, he designed Atari's MARBLE MADNESS, the addictive and maddening arcade game that ate

scads of quarters as players craved another spin of its trackball. Along with the game's catchy soundtrack and Escheresque graphics, Cerny will share insights on how he designed the classic title.

Doom

JOHN ROMERO

\\ Few games can match the ubiquity and legacy of DOOM, the seminal first-person shooter that ushered in thousands of mods, clones, and successors. Programmer, game designer, level designer, and DOOM II final boss John Romero will deliver a postmortem revealing never-before-seen material, memorializing its immersive but nerve-racking 3D environments, networked multiplayer deathmatches, Satanic imagery and themes, Barney WADs, exploding barrels, and BFG 9000.

Pitfall!

DAVID CRANE

\\ PITFALL! isn't just one of the most successful and cherished releases of the Atari 2600; it's also one of the progenitors of the modern platform/adventure genre. Industry legend David Crane, co-founder of Activision and Absolute Entertainment (A BOY AND HIS BLOB), will reminisce about PITFALL!'s vine-swinging hero Harry and the breakthrough game he famously designed

with just a blank sheet of paper and 10 minutes of brainstorming.

Populous

PETER MOLYNEUX

One of the first god games ever released, Bullfrog's POPULOUS beguiled players with its premise of playing as an all-powerful divine being capable of shaping the earth. POPULOUS' always-entertaining designer Peter Molyneux, who went on to found Lionhead Studios where he helped create the game's spiritual descendant BLACK & WHITE and popular RPG series FABLE, will talk about his work on the groundbreaking (and -raising and -lowering) isometric sim.

Bejeweled

JASON KAPALKA

\\ As one of the most popular puzzle games of all time, BEJEWELED and its spin-offs and sequels are everywhere. The man behind the addictive match-three game, PopCap co-founder and chief creative officer Jason Kapalka, will deliver a postmortem talk about designing the franchise that's seen more than 150 million downloads and sold over 25 million copies.

Maniac Mansion

RON GILBERT

\\ Cherished by adventure game fans and reviled by hamsters everywhere,

MANIAC MANSION was the first adventure game LucasArts developed on its SCUMM (Script Creation Utility for Maniac Mansion) platform—the beloved scripting engine used for subsequent classics like SAM & MAX and the MONKEY ISLAND series. Ron Gilbert will talk about his work on MANIAC MANSION, touching on the game's multiple endings, point-and-click interface, and oddball cast of characters.

Raid On Bungeling Bay

WILL WRIGHT

\\ Before he became a household name with gamers, Will Wright created RAID ON BUNGELING BAY, a helicopter action/strategy title for the Commodore 64, NES, and MSX. The unassuming game would serve as the inspiration for Wright's much, much, much more popular SIM CITY series, as it was during his tinkering with RAID ON BUNGELING BAY's editor that the designer discovered building complex cities was more fun and had more potential than destroying them.

All of the "classic postmortem" lectures will be held in the largest possible lecture halls at GDC 2011, and attendance will be on a first-come, first-served basis. In addition, the postmortems will all be video recorded for free post-show availability on the conference's GDC Vault online service.



take
control
of your
future

www.gamasutra.com

the art and business of making games

FRACT

www.richardflanagan.com

FRACT IS AN ATMOSPHERIC PUZZLE GAME INFUSED WITH SYNAPTIC GLITCH VISUALS AND A CRACKLING ELECTRONIC PULSE. WE SPOKE WITH RICHARD FLANAGAN TO FIND OUT HOW HE CREATED THIS 2011 INDEPENDENT GAMES FESTIVAL STUDENT SHOWCASE WINNER.

Jeffrey Fleming: *FRACT has a really fantastic look with abstract, generative-looking art. Did you use any procedural or random techniques to create its environments?*

Richard Flanagan: The creative process during the development of FRACT was almost entirely experimental, and the techniques I used evolved along with my understanding of game development workflow and asset management. My original hope was to use the terrain editor built directly into the Unity engine, but I could not manage to produce the angular and polygonal look I was going for. In order to achieve the harsh angular geography seen in the FRACT beta, I used the terrain generator found in Cinema 4D, which when coupled with a random seed and a series of definable parameters, gave me something closer to my initial vision. These results, however, still required a fair amount of hand sculpting in order to behave correctly within the world.

While some generative systems certainly helped me prototype and produce assets quickly, building a relatively cohesive world still required a fair amount of time.

JF: *How would you compare the Unity engine to other development environments that you might have used?*

RF: My experience with other development systems is relatively limited, with some time spent in Blender Game Engine, Unreal Editor, Adventure Game Studio, and Game Maker, to name a few. While I've seen inspiring examples of great games developed with these systems, they weren't the right tools for me at the time.

Unity surprised me, though, as I never found myself completely stonewalled by a development hurdle. While many of my

workarounds were inelegant at best, I still ended up with results close to my original intent. I think it's a testament to Unity's usability, great documentation, and very collaborative user community that I managed to create the FRACT beta in roughly three months of full-time work.

JF: *Sound is a big part of FRACT. Can you tell me a bit about how you created the game's aural landscape?*

RF: For the sounds in FRACT, I would record tones, pulses, or sonic textures from a trusty old analog synthesizer, sometimes combining them with iconic percussion samples from early electronic music to sync up with events in the game world. In order to add some subtle emotional inflection to many of the sounds, I used a simple white noise generator to lay the basis for tonal shifts and crescendos. Sound design for FRACT happened in parallel to modeling and animation, and even precedes some puzzles. I am fascinated with the principles of synesthesia and I try, where I can, to build contextual relationships between the sounds, sights, and interactions found in the world of FRACT.

Sound design is not only a very integral element in the FRACT beta, but also a major source of inspiration. I have a very strong emotional connection to my initial discovery and exploration of electronic music and sound design, and I wanted to explore some of these themes within a game space. The FRACT beta hints at how I would like to build creative tools for the player to create sound and music in non-intimidating ways. I hope to explore this further.

JF: *You don't often hear analog synthesis in games. What synthesizer are you using to generate FRACT's tones?*

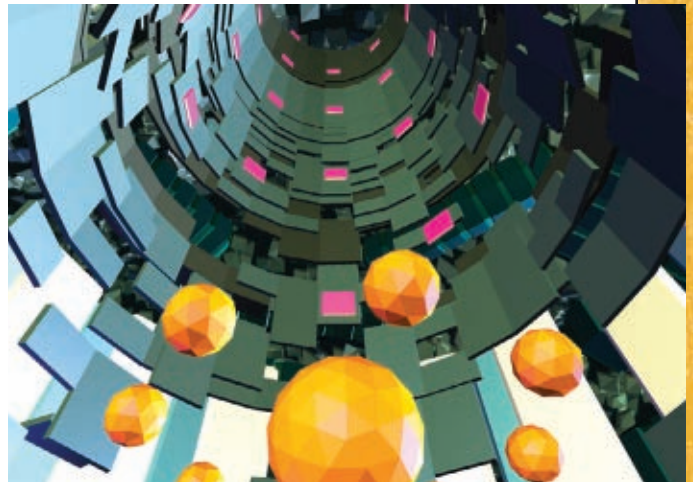
RF: Analog synths really are quite magical devices. While the only synth currently still in my possession is an indestructible Yamaha CS-15, it has proved quite handy on FRACT. The Yamaha is only duophonic, but still yields a remarkable range of sound from expansive soundscapes to percussive squelches and bleeps. A handful of recordings and samples from a Roland Juno-60 and a friend's Korg Monopoly also got mixed into a bit of the sound design, along with samples from Kawai R-100 and Alesis HR-16 drum machines.

I must humbly admit, though, some of the sound was also produced with analog modeling software, including the Arturia Arp 2600 VST and the remarkably versatile built-in synths found in Ableton Live. Given an unlimited timeline

RF: I think the best design is achieved when it is able to communicate a message or experience to the user without making them overtly aware of how it is being achieved. Good typography is especially capable of this, combining both aesthetic and purpose in a subtle but powerful way.

Similarly, I think the gaming moments that we remember most fondly occur when we forget the game design and just get lost in the experience. It is in these moments where gameplay, mechanisms, presentation, and feedback come together to show just how powerful games can be as a medium of communicating ideas.

While the methods in which game design and typography achieve their respective communication pathways are different, I think they are rooted



I'd build everything with these dusty, temperamental machines.

JF: *It was interesting to read on your blog about how important the study of typography was to your development as a designer. What might be a key lesson from typography that game designers should be considering?*

In a similar purpose. I've only managed to scratch the surface of typography in my work and studies, but constantly refer to it as a blueprint for good communication design. And as an aspiring game designer, I hope to be able to apply these principles in the games that I create. 🎮

—Jeffrey Fleming



ACADEMY *of* ART UNIVERSITY

FOUNDED IN SAN FRANCISCO 1929 BY ARTISTS FOR ARTISTS



TAKE CLASSES ONLINE OR IN SAN FRANCISCO

Advertising
Animation & Visual Effects
Architecture*
Art Education
Fashion
Fine Art
Game Design
Graphic Design
Illustration
Industrial Design
Interior Architecture & Design
Motion Pictures & Television
Multimedia Communications
Music for Visual Media
Photography
Web Design & New Media

ENROLL NOW

EARN

YOUR AA, BA, BFA, MA, MFA OR
M-ARCH ACCREDITED DEGREE

ENGAGE

IN CONTINUING ART EDUCATION COURSES

EXPLORE

PRE-COLLEGE SCHOLARSHIP PROGRAMS

WWW.ACADEMYART.EDU

800.544.2787 (U.S. Only) or 415.274.2200

79 NEW MONTGOMERY ST, SAN FRANCISCO, CA 94105

Accredited member WASC, NASAD, Council for
Interior Design Accreditation (BFA-IAD)

**Architecture BFA degree program not currently available online.*

Photo credit: Sungho Lee, Texture & Light Game Design Course

NEW YORK FILM ACADEMY

▶ WWW.NYFA.EDU
1-800-611-FILM

>> BRAND NEW PROGRAM

WORLD

PRESTIGE

Associate of Fine Arts Program
One Year Conservatory Program

NEW YORK CITY • UNIVERSAL STUDIOS, CALIFORNIA

All credits and degrees are solely granted by the New York Film Academy California.
All workshops are solely owned and operated by the New York Film Academy and are not affiliated with Universal Studios.

INVEST IN YOURSELF. BECOME A MASTER OF DIGITAL MEDIA.

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

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

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

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

VISIT US AT GDC IN SAN FRANCISCO | FEB 28 – MAR 4 2011

**CENTRE FOR
DIGITAL MEDIA**

Great Northern Way Campus



emily carr
university of art + design



TURN YOUR PASSION FOR GAMING INTO A CAREER

Game Art

Bachelor's Degree Program
Campus & Online

Game Design

Master's Degree Program
Campus

Game Development

Bachelor's Degree Program
Campus

Game Design

Bachelor's Degree Program
Online



Campus Degrees

Master's

- Entertainment Business
- ▶ Game Design

Bachelor's

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

Associate's

- Graphic Design
- Recording Engineering

Online Degrees

Master's

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

Bachelor's

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



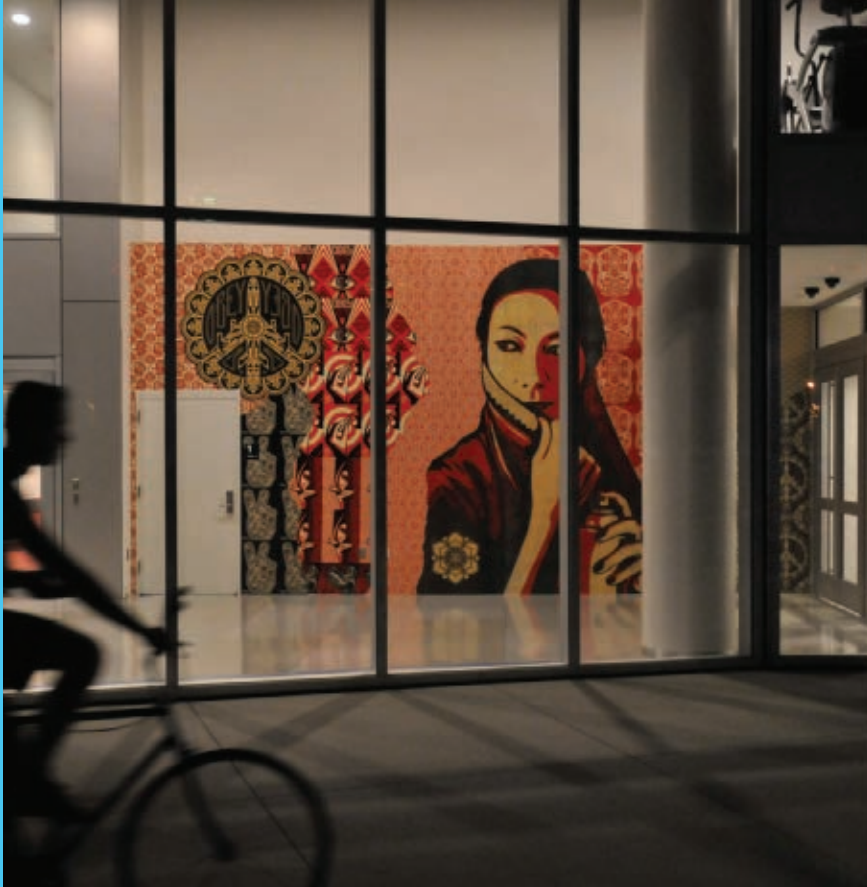
**FULL SAIL
UNIVERSITY.**

fullsail.edu

Winter Park, FL

800.226.7625 • 3300 University Boulevard

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



Game Design and Interactive Media in the heart of Boston

Our Creative Industries program strives to create, foster, and implement digital media innovation through research, education, and ground-breaking team-based interdisciplinary projects.

Collaborate and create with one of seven combined majors, or the Creative Industries minor.

For more information, go to:
www.ci.neu.edu



Northeastern

CREATIVE INDUSTRIES

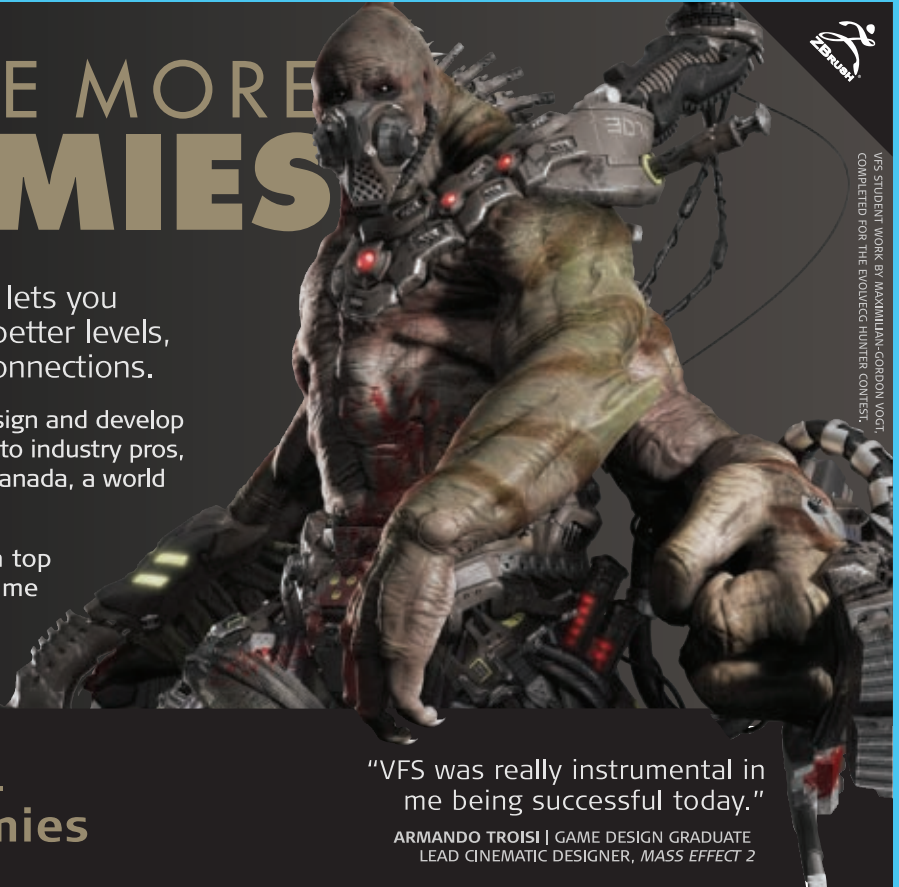
Northeastern University
360 Huntington Avenue
Boston, MA 02115-5000

MAKE MORE ENEMIES

Game Design at VFS lets you make more enemies, better levels, and tighter industry connections.

In one intense year, you design and develop great games, present them to industry pros, and do it all in Vancouver, Canada, a world hub of game development.

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



VFS STUDENT WORK BY MAXIMILIAN-GORON VOCT
COMPLETED FOR THE EVOLVED: HUNTER CONTEST.

VFS

Find out more.
vfs.com/enemies

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

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

GET YOUR GAME ON!

EVERYTHING YOU NEED TO KNOW TO GET INTO THE GAME INDUSTRY!

- News and features for students and educators
- Getting Started section – an invaluable how-to guide
- Message Boards



DIGITAL COUNSELOR

I'LL MATCH YOUR INTERESTS AND GOALS WITH THE RIGHT GAME RELATED PROGRAMS AND SCHOOLS FROM AROUND THE WORLD.

www.gamecareerguide.com



Download and Play the latest student Games!

Download your FREE digital edition of the 2010 *gamedeveloper* Game Career Guide online at: www.gamecareerguide.com



ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
Academy of Art University	51	Northeastern University	54
Epic Games	12	Rad Game Tools	C4
Full Sail Real World Education	53	Scottish Development International	16
GameSpot	42	Seapine Software	C2
IGDA	34	TransGaming	6
Intel	19	TwoFour54	3
Masters of Digital Media	53	Unity Workshop	31
NaturalMotion	C3	Vancouver Film School	54
New York Film Academy	52	VSoft Technologies	29

gd Game Developer (ISSN 1073-922X) is published monthly by United Business Media LLC, 303 Second Street, Suite 900 South, South Tower, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$69.95; all other countries: \$99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *gd Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *gd Game Developer* on any correspondence. All content, copyright *gd Game Developer* magazine/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



TO THE WRITER

A GAME STUDIO MANAGES ITS HOLLYWOOD GUY

HEY, MISTER WRITER!

» Thanks for sending over the latest revision of the game script right on schedule! Late last week I circulated your draft among an elite, select group of the team—call it the “brain trust”—of thirty-seven leads, senior employees, and basically anyone else at the studio who has an opinion about dialogue, story, films, or games. These folks sacrificed their weekends to go over your script very carefully, and would now like to give you some feedback. Don't worry, none of it is bad!

PACING

» One of the biggest concerns is that the scenes seemed to run long. I know you come from the film world where you can do anything you want, but it's different here in games. Our primary objective should be getting the cutscene over as quickly as possible. You have to imagine the player there with his finger on the “skip” button. All he wants to do is get on with it so he can join the Cyber Police and cap bad guys in a dark and gritty future.

Also, we wanted really deep, three-dimensional characters, but we aren't getting any sense of that from this script. Sergeant Guts in particular just seems like a cliché to us. His personality should have layers and complexity.

THE NURSE

» Now, I know we discussed this character in a lot of different meetings, the nurse who cares about all the death and so on, but—how can I put this without sounding bad—the female presence here just seems to distract from the, you know, military feel we're going for.

Don't get me wrong, we all love extremely strong female characters (I myself keep one around at home, if you know what I mean, ha ha ha!), but the nurse just seems a

bit too, well, to hear the team talk about it, she's just too much of a character. The fact that she does something important runs counter to our idea that the player should be the only one doing anything. We call this concept “fantasy.”

Nobody wants her cut entirely, of course—she needs to be around so it's not like a total bro-fest—but if she could just show up for a moment and then just disappear, we think that would be best. We don't need like an arc or anything for her. She can be there to ensure there's women around and prove that we aren't sexist or anything. Then she can drop out of the story and let the real stuff happen—the battle to save mankind.

THE MELODRAMA

» Which brings me to my next point. One of the other items that came up time and again was that we need to studiously avoid melodrama. By “melodrama,” we mean anything that's emotional or in any way breaks the hard, unsentimental atmosphere we're talking about creating here. Again, this is a serious future that is dark and gritty, where the survival of the human race is at stake. There's no room for any of that schmaltzy stuff about being sad, will I ever see you again, etc., nor should there be any humor. Funny situations are just going to detract from the serious atmosphere.

We're all really into military shows here—we watch a whole lot of them—and the ones that are just sort of documenting what's going on are really the ones we like the most and want to go for. You know, *cinéma vérité* kind of stuff. We want to take that approach to telling the story we have here. Imagine things as they *actually are* when the player enlists with the Cyber Police to save the world.



Not that any of this really matters—I mean, nobody's going to watch the cutscenes anyway. I don't mean to sound down on your work or anything, I'm just stating a fact.

ALL OF THE DIALOGUE

» Some of our guys had problems with the way the lines were written. Generally speaking, we're worried that they just aren't “cool” enough. We're also concerned that some of the lines are a bit difficult to understand. While this is a violent military-themed game that caters to the hardcore (and lets them pwn noobs), we do want everyone to be able to enjoy it, so you need to write lines that clearly explain everything, even to, say, a soccer mom.

Here's an example: we're just assuming players know what “tangoes” are. With just a quick, small change, you could rewrite the line to be: “Tangoes—that means bad guys!” I'm not trying to tell you how to do your job or anything, but just suggesting something along those lines. As the writer, we trust you to come up with something cool there.

THE STORY

» Our last point of feedback has to do with the way the story “wraps up” at the end. It's like all these plot threads sort of come together and are just resolved. Not only does

that not really leave the door open to sequels, which is important, but we also think it's too obvious of a storytelling device. We don't want players to think a story is being told to them. It should feel more like real life, like a real war.

In real war, you don't get closure. War is just one thing after another with very little logic to it. That's really what we're trying to capture in our storytelling: gritty, dark chaos that's hard and unsentimental.

Well, that's it for the feedback. I'd say it's pretty targeted and actionable, so I think a revision wouldn't be too difficult in the next couple of days. By the way, have you seen that new *TRON* movie? It's pretty good about giving you the plot in a way that kind of sounds like there's a whole other world out there and you don't really follow it, but you sort of do follow it at the same time. You know what I mean? You should watch that movie and you'll see what I'm talking about. We should make our script more like the *TRON* script.

Thanks for the good work and we'll be in touch soon! ☺

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

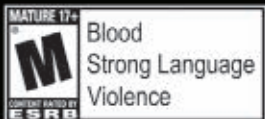


KAOS STUDIOS

USES MORPHEME

“In Homefront, we aimed to present the player with an authentic vision of an occupied America. NaturalMotion’s Morpheme empowered our animators to create believable characters that our audience could connect with, to help us convincingly portray the human cost of war.”

David Votypka, Creative Director/GM Kaos Studios



RELEASED MARCH 8th 2011

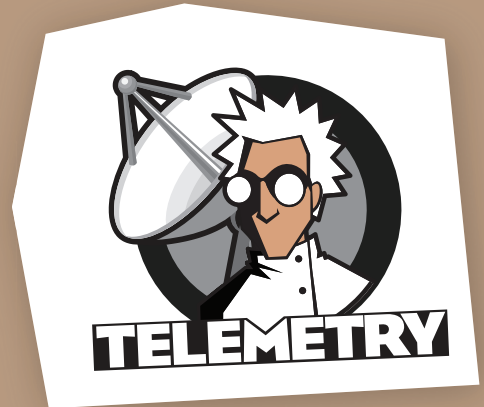


THE NEWEST RAD TOOL IS

yes, it's **NEW**

NOW SHIPPING

OUT
OF THIS
WORLD



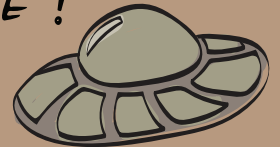
TELEMETRY

42

is a programming library and set of tools for instrumenting, profiling, tuning and visualizing application **PERFORMANCE!**



VISUALIZE real-time game performance,



see **WHEN** things happen — not merely **WHAT** happened!

PROBE the hierarchical display —

see thread interactions, context switches, and mutex locking!

THIS ISN'T JUST ROCKET SCIENCE, THIS IS **rad!**



www.radgametools.com/telemetry
(425) 893-4300