THE LEADING GAME IND

FEBRUARY 2012 INSIDE

gd

GAME DEVELOPER MAGAZINE

10 YEARS OF SALARY SURVEYS

tweaks and balances: tuning an open-world rpg

stacking postmortem

THE LEADING GAME INDUSTRY MAGAZINE VOL 19 NO 2

FEBRUARY 2012 INSIDE: THE DEATH OF THE DEDICATED NETWORK PROGRAMMER

UBM

RED 5 STUDIOS

firefallthegame.com

Autodesk®

GAMEWARE

Webinar Wednesdays

Join our weekly Webinar every Wednesday to learn more about our Gameware products:

- » **Scaleform** (UI)
- » **Beast** (Lighting)
- » **Kynapse** (AI)
- » **HumanIK** (Animation)
- » **Cognition** (Behavior)
- » **Population** (Crowds)

CONTENTS.0212

VOLUME 19 NUMBER 2

POSTMORTEM

FEATURES

DEPARTMENTS

GAME DEVELOPER MAGAZINE

FREEDOM'S DOWNWARD SPIRAL

WHY YOU SHOULD BE ANGRIER ABOUT NDAA THAN YOU WERE ABOUT SOPA

By now, everyone's aware of the U.S. House bill SOPA, the Stop Online Piracy Act. And the related PIPA—the Senate's PROTECT Intellectual Property Act. Internet denizens have correctly rallied against these bills, which had vague, overly broad language that would have given the U.S. Government and rights-holders the means to block certain web sites deemed to be "primarily dedicated" to copyright and trademark infringement.

These would have bypassed due process, and given the U.S. Government and giant media corporations the power to be judge, jury, and executioner over what U.S. internet users are allowed to access online.

A GREATER THREAT

» The internet blacklist bills have pretty much united everyone in game development against them, and after a recent day of internet blackouts, voting on them has been postponed. But another, far more dangerous bill passed in late 2011, that could limit our freedom of expression in an even grander scope than SOPA/PIPA ever would.

No citizen voted for or against this law. Much like with SOPA, that opportunity was not afforded us. The article in question is the National Defense Authorization Act, and it is the greatest threat to intellectual and personal freedom in America.

NDAA is not new—the bill has been in effect in various forms for almost 50 straight years. But the big deal is some new language that was added to Title X, Subtitle D sub-sections 1021–1022 of the bill, which I encourage you to check out for yourself. (Alternately, you can read Salon.com's excellent article, "Three myths about the detention bill.") This new language allows for a) the indefinite detention without trial of anyone, American or otherwise, who is perceived to support terrorism, and b) an expanded view of what the war on terrorism entails (which can now be continued until "the end of hostilities").

The ability to imprison anyone without trial or appeal should make you angry enough—but before I lose my audience, let me get straight to how this relates to games. Consider COUNTER-STRIKE. One team plays as the terrorists, one as the counter-terrorists. In this game, Valve allows players to take on the role of terrorists, and encourages them to win. Is this supporting terrorism? You and I understand the concepts of fiction and role play, and the power and intrigue of imagination. But does our government?

Laws have a very curious tendency of serving whomsoever has the authority to twist them, and governments worldwide have been in a mad race to remove freedoms in the name of national security. Under the new NDAA laws, COUNTER-STRIKE's designers could absolutely be detained until the concept of "terrorism" no longer exists, which, as things are going, seems a long way off.

How many of our games allow players to see war from both sides, even if briefly? Quite a few, really. And while President Obama says that he definitely won't use this against Americans, even though he could, who's to say situations won't change? And even if he doesn't, what if the next president in line does?

Here's an example from another angle. Tahadi Games has brought the Korean FPS POINT BLANK to the Middle East. I think of the Middle East as the birthplace of much of ancient culture and religion. Others though, think of the Middle East as a hotbed of terrorism, and would swear up and down that giving Arabic-speaking countries access to an FPS is akin to training up a legion of terrorists. If you sell your game in Arabic-speaking countries, are you at risk for detention in America? The incredibly vague law leaves that possibility open, should someone have played your game before committing some malicious act.

COULD IT HAPPEN HERE?

» While these laws haven't yet been used to the effect described, they could, and that should scare you. Former U.S. Marine Amir Mizra Hekmati was recently sentenced to death in Iran for aiding in the development of KUMA\WAR, which lets players engage in "real life" scenarios, including an episode titled "Assault on Iran." The Iranian government says KUMA\WAR was funded by the CIA to help "manipulate public opinion in the Middle East." His sentence has not been carried out yet. Tommy Vietor, a spokesman for the White House's National Security Council, says that "The Iranian regime has a history of falsely accusing people of being spies, of eliciting forced confessions, and of holding innocent Americans for political reasons."

And that is precisely the power that NDAA's new language grants our own government. What if Hekmati had worked on a game like KUMA\WAR that showed what might happen if Iranian forces landed on U.S. shores? We could legally give him nearly the same treatment we're now decrying.

FREE GAME DEVELOPMENT

» As of 2011, video games are protected as free speech, but that only goes so far as the letter of the law, as SOPA, PIPA, and NDAA prove. When laws are so open ended, an official with an agenda could cause serious damage to our industry, and to America at large. With the outcry against SOPA and PIPA, we could be on our way to winning back the freedom of the internet—now let's fight for our freedom of expression and humanity.

If we want to make games that address issues like war and military occupation, we need to fix NDAA. I urge you to write to your representatives, or do whatever you can. Remember though; even if we fix NDAA and kill SOPA/PIPA, we only get back to square one. There's lots more work to be done to support our industry and our society's freedoms. But if we don't fix these problematic laws, we all lose. Every one of us.

—Brandon Sheffield
twitter: @necrosofty

GAME DEVELOPER MAGAZINE
WWW.GDMAG.COM

UBM LLC.
303 Second Street, Suite 900, South Tower
San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090

SUBSCRIPTION SERVICES

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

FOR DIGITAL SUBSCRIPTION INFORMATION
www.gdmag.com/subscribe

EDITORIAL

PUBLISHER
Simon Carless e: *scarless@gdmag.com*
EDITOR-IN-CHIEF
Brandon Sheffield e: *bsheffield@gdmag.com*
PRODUCTION EDITOR
Jade Kraus e: *jkraus@gdmag.com*
ART DIRECTOR
Joseph Mitch e: *jmitch@gdmag.com*
DESIGNER
Cliff Scorso e: *cliff.scorso@ubm.com*
CONTRIBUTING WRITERS
Tom Curtis
Ari Silvennoinen
Carey Chico
Ryan Consell
Damian Kastbauer
Soren Johnson
David Edery
Matthew Wasteland
ADVISORY BOARD
Mick West **Independent**
Brad Bulkley **Microsoft**
Clinton Keith **Independent**
Brenda Brathwaite **Loot Drop**
Bijan Forutanpour **Sony Online Entertainment**
Mark DeLoura **THQ**
Carey Chico **Globex Studios**
Mike Acton **Insomniac**

ADVERTISING SALES

GLOBAL SALES DIRECTOR
Aaron Murawski e: *amurawski@ubm.com*
t: 415.947.6227
MEDIA ACCOUNT MANAGER
Jennifer Sulik e: *jennifer.sulik@ubm.com*
t: 415.947.6227
GLOBAL ACCOUNT MANAGER, RECRUITMENT
Gina Gross e: *ggross@ubm.com*
t: 415.947.6241
GLOBAL ACCOUNT MANAGER, EDUCATION
Rafael Vallin e: *rvallin@ubm.com*
t: 415.947.6223

ADVERTISING PRODUCTION

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

REPRINTS

WRIGHT'S MEDIA
Jason Pampell e: *jpampell@wrightsmedia.com*
t: 877-652-5295

AUDIENCE DEVELOPMENT

AUDIENCE DEVELOPMENT MANAGER Nancy Grant
e: *nancy.grant@ubm.com*
LIST RENTAL Peter Candito
Specialist Marketing Services
t: 631-787-3008 x 3020
e: *petercan@SMS-Inc.com*
ubm.sms-inc.com

UBM
WWW.UBM.COM

intel® Software

Intel® GPA† helps Zombie Studios deliver *Blacklight*:*: *Retribution*

ZOMBIE STUDIOS OPTIMIZES WITH INTEL® GRAPHICS PERFORMANCE ANALYZERS

The Intel® Graphics Performance Analyzers (Intel® GPA) is a powerful graphics tool suite for analyzing and optimizing your games, media, and other graphics-intensive applications. With Intel GPA, you can conduct in-depth analysis from the system level all the way down to individual elements, allowing you to maximize the performance of your applications.

" Ultimately, the results of optimizations we did with Intel GPA tools made the Blacklight: Retribution *experience better and better for our players."*

– CHANCE LYON, LEAD DEVELOPER, ZOMBIE STUDIOS

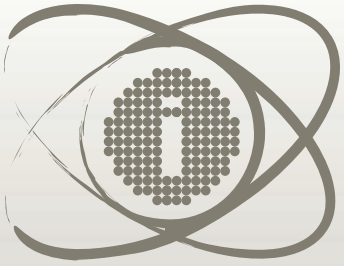
Intel® GPA System Analyzer.
Learn whether your game is CPU- or GPU-bound. Quickly analyze game performance and identify potential bottlenecks.

Intel® GPA Frame Analyzer.
Optimize graphics performance through deep frame analysis of elements at the draw-call level.

Intel® GPA Platform Analyzer.
Visualize performance of your application's tasks across the CPU and GPU.

DOWNLOAD INTEL® GRAPHICS PERFORMANCE ANALYZERS FOR FREE at www.intel.com/software/gpa

independent games festival announces finalists



The Independent Games Festival (IGF) juries have announced the Main Competition finalists for the 14th annual presentation of its awards, celebrating the brightest creatives and the most influential game designs to come out of the independent community in the past year.

All finalist games, which were voted on by a discipline-specific set of juries, will be playable at the IGF Pavilion on the Game Developers Conference 2012 Expo floor from March 7–9, 2012, at San Francisco's Moscone Center, as part of a week of independent game-related content that also includes the Independent Games Summit (March 5–6), and the IGF Awards ceremony itself.

The IGF Awards, where this year's IGF winners will be unveiled, will be held on the evening of March 7, alongside the Game Developers Choice Awards. IGF award recipients will receive $60,000 worth of prizes in various categories, including the $30,000 Seumas McNally Grand Prize.

The full list of finalists for the 2012 Independent Games Festival, with jury-picked "honorable mentions" to those top-quality games that didn't quite make it to finalist status, is as follows:

Excellence In Visual Art

- Botanicula (Amanita Design)
- Dear Esther (thechineseroom)
- Lume (State Of Play Games)
- Mirage (Mario von Rickenbach)
- Wonderputt (Damp Gnat)

Honorable Mentions

- Beat Sneak Bandit (Simogo)
- Dustforce (Hitbox Team)
- Fader (Chris Makris)
- Proun (Joost van Dongen)
- Toren (Swordtales)

Technical Excellence

- Antichamber (Demruth)
- Fez (Polytron)
- Prom Week (Expressive Intelligence Studio, UC Santa Cruz)
- Realm of the Mad God (Wild Shadow Studios and Spry Fox)
- Spelunky (Mossmouth)

Honorable Mentions

- Frozen Synapse (Mode 7 Games)
- Nitronic Rush (DigiPen Institute of Technology)
- Reflow (Xymatic)
- Super T.I.M.E Force (Capy)
- Tiny and Big -Grandpa's Leftovers (Black Pants)

Excellence In Design

- Atom Zombie Smasher (Blendo Games)
- English Country Tune (Stephen Lavelle)
- Frozen Synapse (Mode 7 Games)
- Gunpoint (Tom Francis, John Roberts, and Fabian van Dommelen)
- Spelunky (Mossmouth)

Honorable Mentions

- Faraway (Steph Thirion)
- FTL (Justin Ma & Matthew Davis)
- Johann Sebastian Joust (Die Gute Fabrik)
- SpaceChem (Zachtronics Industries)
- Where Is My Heart (Die Gute Fabrik)

Excellence In Audio

- Botanicula (Amanita Design)
- Dear Esther (thechineseroom)
- Pugs Luv Beats (Lucky Frame)
- To The Moon (Freebird Games)
- Waking Mars (Tiger Style)

Honorable Mentions

- Beatbuddy (THREAKS)
- Beat Sneak Bandit (Simogo)
- Fez (Polytron)
- Proteus (Ed Key and David Kanaga)
- Where Is My Heart (Die Gute Fabrik)

Best Mobile Game

- ASYNC Corp (Powerhead Games)
- Beat Sneak Bandit (Simogo)
- Faraway (Steph Thirion)
- Ridiculous Fishing (Vlambeer)
- Waking Mars (Tiger Style)

Honorable Mentions

- Fingle (Game Oven Studios)
- Hundreds (SemiSecret and aeiowu)
- iBlast Moki 2 (Godzilab)
- Temple Run (Imangi Studios)
- Tentacles (Press Play ApS)

Nuovo Award

[Designed to honor abstract, shortform, and unconventional game development.]

- At a Distance (Terry Cavanagh)
- Dear Esther (thechineseroom)
- Fingle (Game Oven Studios)
- GIRP (Bennett Foddy)
- Proteus (Ed Key and David Kanaga)
- Johann Sebastian Joust (Die Gute Fabrik)
- Storyteller (Daniel Benmergui)
- Way (CoCo & Co.)

Honorable Mentions

- Deep Sea (WRAUGHK)
- Four Letter Word (Terry Cavanagh);
- GlitchHiker (Aardbever)
- Hundreds (SemiSecret and aeiowu)
- POP (Rob Lach)

Seumas McNally Grand Prize

- Dear Esther (thechineseroom)
- Fez (Polytron)
- Frozen Synapse (Mode 7 Games)
- Johann Sebastian Joust (Die Gute Fabrik)
- Spelunky (Mossmouth)

Honorable Mentions

- Antichamber (Demruth)
- FTL (Justin Ma and Matthew Davis)
- Proteus (Ed Key and David Kanaga)
- SpaceChem (Zachtronics Industries)
- Where Is My Heart (Die Gute Fabrik)

In addition, the IGF announced that the festival program has entered a new multi-year partnership with Microsoft Studios and its Xbox LIVE Arcade publishing team, to offer a new prize to support notable indies. A stand-alone jury of independent game creators is working with Microsoft to identify possible choices for the "XBLA Prize."

The prize includes a guaranteed first-party publishing deal to release the selected title on Microsoft's LIVE-enabled platforms, including the Xbox LIVE Arcade service, Windows Phone, and Windows, and full game funding to complete the title, if desired, and its winner will be announced on-stage during the IGF Awards this March.

The Independent Games Festival was established in 1998 by UBM TechWeb's Game Network to encourage the rise of independent game development and to recognize the best independent game titles, in the same way that the Sundance Film Festival honors the independent film community.

For more information on the Independent Games Festival, please visit the official IGF website at www.igf.com—and for those interested in registering for GDC 2012, which includes the Independent Games Summit, the IGF Pavilion, and the IGF Awards Ceremony, please visit the Game Developers Conference website www.gdconf.com.

the future of *game developer*

/// Astute readers may have already noticed, but there have been a number of changes in the *Game Developer* suite of products. First, by popular demand, there is now an app for iOS devices. The September issue is available to try for free—and while normal subscriptions are available, the magazine can also be delivered to Apple Newsstand. An Android version is in the works as well, which will be coming in the near future.

In the meantime, we've also fully revamped our web site, www.gdmag.com. It's more than just a cosmetic update—in addition to showing the newest magazine content, we've also got a blog, which is updated weekly. Through the blog, we'll be bringing to light a variety of curious items, but perhaps most notable are the galleries—we get a lot of extra art when making postmortems, and don't have the pages to show it all, so we'll be displaying some of the best art there.

Elsewhere on the site there's a fresh new resources tab, with source code from articles and past issue summaries, and we've revamped our contributor page with updated info on submitting articles, and added a lot more features and information about our new and improved subscription offers.

Talking of subscriptions, *Game Developer* has been a print magazine for over 15 years, but as our readers change how they want to interact with media, we've embraced the digital landscape with our digital edition, which is viewable in web browsers, and then there's our aforementioned iPad/iOS version.

With all that work done, it's time to revise our subscription strategy. All our digital offerings have now been wrapped up into one subscription package for $29.95 per year, which gives you 11 digital issues per year, the iPad app, and digital archives going back to 2004. It's important to note that the iPad app only extends as far back as October 2011, but the digital archives can be downloaded as PDFs, and easily read on an iPad or iPhone with an appropriate reader.

For our other subscription option, a lot of people still enjoy the feeling of a print magazine,

and that includes us. The print version isn't going away any time soon. For $49.95 per year (for U.S. residents—$59.95 for Canada/Mexico, $69.95 worldwide), you'll get the print magazine, plus all the digital content described above. As a bonus incentive, you'll get our special Best of Postmortems issue, which is only available to new (or renewing) paid print subscribers via this offer, and only available while supplies last.

This special issue has a glossy cover and perfect binding, with nearly 100 pages that showcase 14 postmortems from the last 10 years, including games such as the original DEUS EX, NEVERWINTER NIGHTS, KATAMARI DAMACY, ROCK BAND, and FINAL FANTASY XIII. The issue also gives a glimpse into how vastly the art direction of *Game Developer* has changed over the years.

The new subscription offers are available immediately, and will help *Game Developer* continue to bring you the award-winning content that has helped game developers do their jobs since 1994. And if you feel inclined, please feel free to follow us on Facebook and Twitter (#gamedevmag), and send us your feedback to editors@gdmag.com!

crysis 2 named most-pirated game of 2011

/// Data from download tracker TorrentFreak.com shows the PC version of EA and Crytek's CRYSIS 2 was the most pirated video game of 2011, with more than 3.92 million confirmed BitTorrent downloads as of December 30, 2011.

This total falls just shy of the downloads accumulated by last year's top game, CALL OF DUTY: BLACK OPS, which saw more than 4.1 million confirmed downloads between its November 2010 launch and December of that year. By comparison, CRYSIS 2 launched in March, and thus had much more time to reach such a high number of downloads.

On the Wii, Nintendo's MARIO GALAXY 2 took the top spot despite launching in 2010, with more 1.28 million downloads. The game was followed by MARIO SPORTS MIX and the highly acclaimed XENOBLADE CHRONICLES, which was presumed exclusive to Europe and Japan until Nintendo announced it for the U.S.

The Xbox 360's most pirated title was Epic's GEARS OF WAR 3, which saw 890,000 downloads—considerably less than the top games for the Wii or PC. Pirated games for PlayStation 3 were not included in TorrentFreak's data.

TorrentFreak.com's complete lists of illegally torrented games for 2011 is as follows:

PC Unauthorized BitTorrent Downloads in 2011

(through December 30, 2011)

1. CRYSIS 2 (3,920,000)
2. CALL OF DUTY: MODERN WARFARE 3 (3,650,000)
3. BATTLEFIELD 3 (3,510,000)
4. FIFA 12 (3,390,000)
5. PORTAL 2 (3,240,000)

Wii Unauthorized BitTorrent Downloads in 2011

(through December 30, 2011)

1. SUPER MARIO GALAXY 2 (1,280,000)
2. MARIO SPORTS MIX (1,090,000)
3. XENOBLADE CHRONICLES (950,000)
4. LEGO PIRATES OF THE CARIBBEAN (870,000)
5. FIFA 12 (860,000)

360 Unauthorized BitTorrent Downloads in 2011

(through December 30, 2011)

1. GEARS OF WAR 3 (890,000)
2. CALL OF DUTY: MODERN WARFARE 3 (830,000)
3. BATTLEFIELD 3 (760,000)
4. FORZA MOTORSPORT 4 (720,000)
5. KINECT SPORTS: SEASON TWO (690,000)

—TOM CURTIS

RED 5 STUDIOS

firefallthegame.com

10 years of salary surveys

an aggregate analysis of game developers salaries from 2001~2010

ARA SHIRINIAN

Game Developer's annual salary surveys have measured and reported industry salary data since 2001. Given that it's been 10 years now, we wondered if there were any interesting patterns that the salary survey data could show across the years. In the coming pages, we'll show the results of a collective analysis of the data from the past ten years of salary surveys, from the first edition in 2001 to the tenth edition published in 2011.

METHODOLOGY AND CAVEATS

The data collected by salary surveys has been adjusted to various degrees year over year. In this analysis, only data categories that were present for enough years to see an interesting trend were considered. For example, data about "writer" salaries was not included because they were only collected from the fifth through eighth editions. Also, only actual reported data is considered here. No attempts were made to derive values from the computation of other reported values.

For the first and second editions of the salary survey, it appears as though data was collected in the same year it was published. However, for the third edition and onward, data was collected in the year before that of publication. The years displayed in all of the charts refer to the year of data collection, not that of publication. For this reason, "2001" refers to the first edition, where data was both collected and published in 2001, but "2010" refers from data from the tenth edition, where data was collected in 2010 but was published in 2011.

Whenever data is missing for a particular year, it is omitted from the chart. For this reason, some chart lines may have gaps here and there. Also, certain types of data only were collected after a certain year. For example, average salaries per discipline for Europe and Canada began in 2005.

Depending on the year, salaries above a certain threshold were not considered for inclusion. From 2001 to 2005, salaries above $300,000 were not considered. In 2006, this threshold was reduced to $200,000. For the succeeding years, 2007 through 2011, this threshold was changed again to $202,500. Even though there was a huge jump between 2005 and 2006, it does not appear as though there were enough salaries reported in the $200,000 to $300,000 range in 2005 and earlier to make a noticeable impact in the overall trends.

The margin of error for salary data also varied from year to year. The smallest margin of error was 1.7% at the 95% confidence interval in 2005 and 2006, the years where the salary survey enjoyed the largest sample sizes. The largest margin of error was 3.06% in 2009.

There are some occasional large spikes in salary data. These spikes are most likely caused by an unusual distribution of samples combined with an unusually low sample size for a particular data category. It is also possible that some of the spikes reflect incorrect data. The accuracy of data in this report is wholly dependent on the accuracy of data in previously published salary surveys and other reports. It should also be noted that the results come from self-supplied information from working developers. It was up to the individual developers to represent their salaries accurately, or otherwise.

In some of the filled-area charts, any isosceles triangle shapes you see are artifacts of having data for one year, (the top of the triangle) but no data for the next or previous year. It does not mean that the data was 0 for the next and previous years.

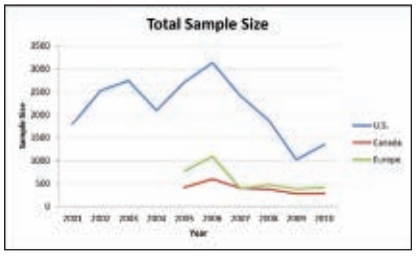
Finally, the way "years of experience" have been delineated has also shifted somewhat over the years of salary surveys. In this report, to reduce the amount of labeling, years of experience are always labeled as "<3 y." (less than three years), "3–6 y." (three to six years) and "6+ y." (more than six years). However, in 2001, the three categories were actually 1–2 y., 3–5 y., and 6+ y. Between 2002 and 2004 this became <2 y., 2–5 y., and 6+ y. It was only in 2005 and onward where the delineations settled on <3 y., 3–6 y., and 6+ y.

DISCIPLINE CATEGORIES

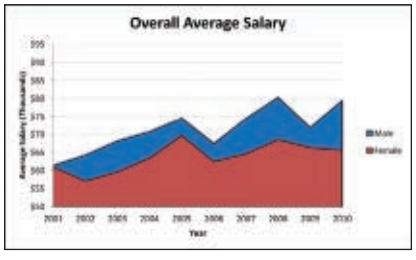
Game Developer has traditionally organized the industry into five discipline categories with slightly changing nomenclature over the years: Programming, Art (and Animation), (Game) Design, Production, and Audio (Developers/Professionals). In 2003, a sixth category was introduced, Quality Assurance (QA), which was renamed to "QA Testers" in 2007. In 2004, the seventh category, "Business and Legal" was introduced, which was renamed to "Business and Marketing" for just 2007.

THE CHARTS

Overall Data

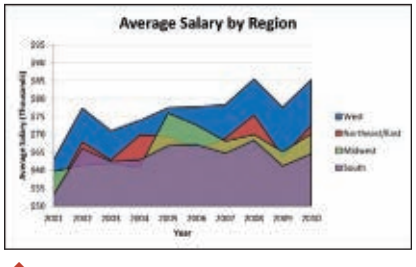


This represents the total sample size for each salary survey over the years. Starting in 2005, salary surveys began to collect European and Canadian data, although this data is completely separately reported. All of the sections that follow use only U.S. data except where noted. The larger the sample size, the smaller the margin of error.



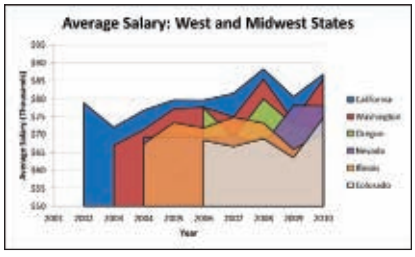
This chart shows the overall average salary across all disciplines, divided by gender. If we just look at the endpoints of data, the average overall increase in salary for males is about $1,800 per year, or roughly 2.6%. For females, the average overall increase in salary is about $460 per year, or roughly 0.7%.

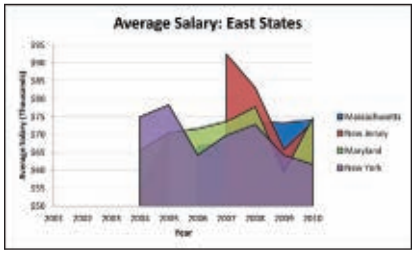
Salary Data by State and Region

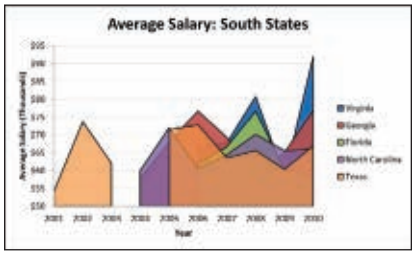


Salary surveys have divided the U.S. into four major regions. The "East" region has also been called "Northeast" depending on the survey edition, but the way states are grouped into each

region has never changed across the surveys. The greatest disparity between regions was reported in 2010, where workers in the West region earned over $20,000 more than those in the South.

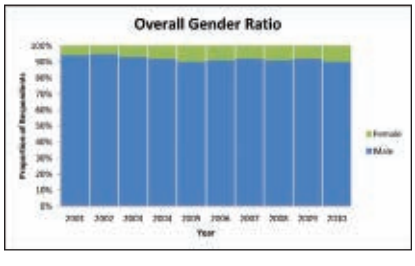


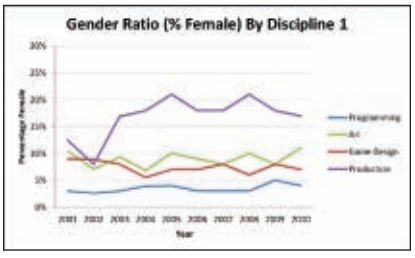


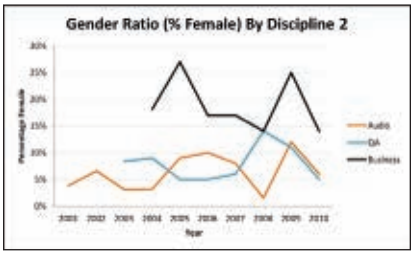


These charts show the average salary reported from specific states, generally those that boasted the highest average salaries. There appears to be a rather large variance of individual state data from year to year. This may either be an indication of a large amount of movement from state to state between years, or a small sample size on a per-state basis. Interestingly, California's graph is very close to that of the entire West region, which seems to suggest that it carries the great majority of respondents for its region. California is also the highest-paying state overall, apart from a few unusual spikes.

Gender Ratio





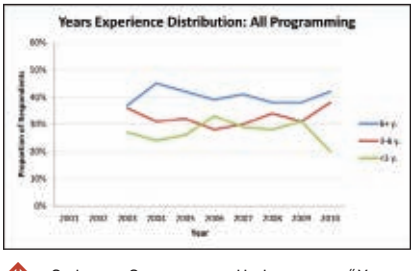


Overall, 2005 and 2010 showed the highest proportion of female workers at 10%. 2002 saw the lowest value at 5.2%. However, the "Business" category was not polled until 2004, which may have affected the particularly low gender ratio numbers during the first few years.

When we break out the gender ratio by discipline, we see several interesting results. Production and Business show a dramatically higher proportion of female workers than any of the other disciplines. Programming showed not only the lowest proportion of females, but also the most consistently unchanging proportion of gender. On the other hand, Business, Audio, and QA all showed wildly fluctuating gender ratios over the years, by as much as 10% from one year to another.

DATA BY DISCIPLINE

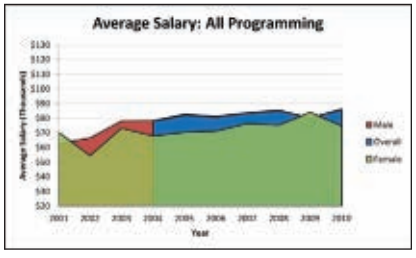
Programming

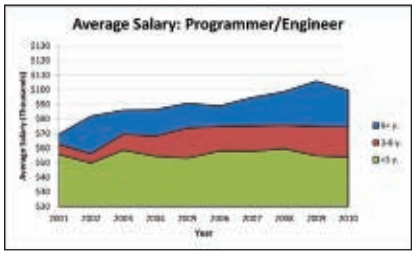


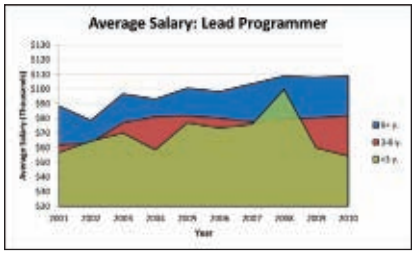
Salary Surveys all have a "Years Experience in the Industry" for each discipline, broken up into three strata of experience. By charting this data, we can see overall trends about the general age makeup of each discipline. The programming field shows a larger proportion of experienced workers, although

Programming showed not only the lowest proportion of females, but also the most consistently unchanging proportion of gender.

it is the most balanced and tightly grouped of all disciplines in terms of the distribution of experience. Ostensibly this kind of grouping is good, as it means that new talent is steadily coming into the discipline and is staying in it for at least several years. In 2010 this seems to have taken a turn for the worse, with a large drop in the proportion of new talent.







These charts show average salaries by job title, further broken down by years experience in the industry within each chart. It appears that while salaries are increasing gradually for highly experienced programmers, salaries for less experienced programmers are flat or decreasing slightly.

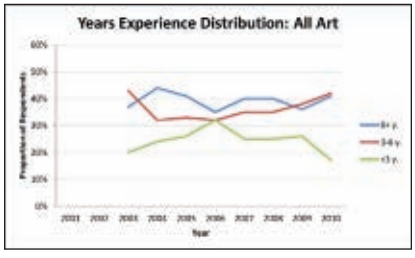
GAMERS HAVE FOUND A NEW AND THRILLING WAY TO INTERACT AROUND THE GAMES THEY LOVE.

Every month more than 12 million gamers come together on the world's #1 video game broadcasting and chat community.

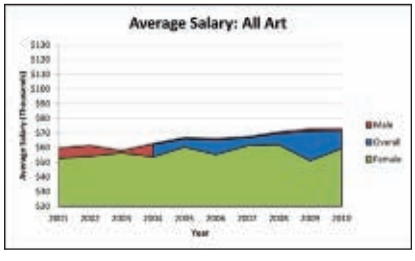
Learn more about video game broadcasting with TwitchTV

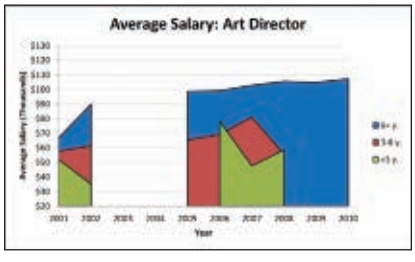
www.twitch.tv
jonathan@twitch.tv

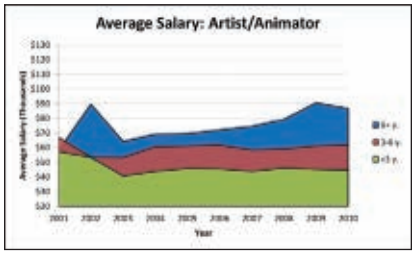
Art

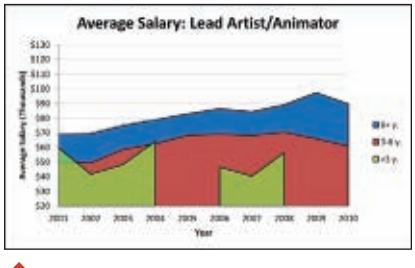


↗ The art discipline's distribution of experience seems similar to that of programmers, with a slightly larger proportion of workers with 3–6 years experience, and slightly fewer inexperienced workers. Again, the trend over the past year or so suggests a reduction of entry-level artists in the market.



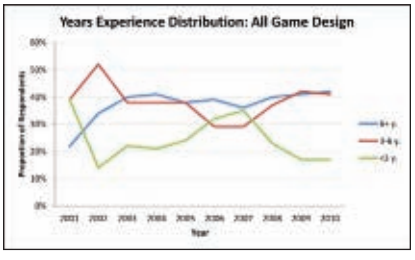




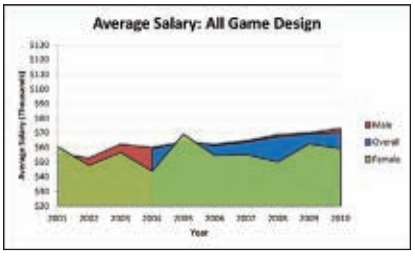


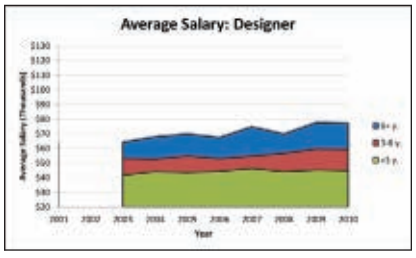
↗ More so than any other discipline, there seems to be a gradually increasing gap between male and female artist/animator salaries. Some of the unusual patterns in the early history of artist/animator salaries can be explained by changes in the way job titles were defined; for the first three editions of the salary survey, artists and animators were considered separately, until they were finally merged into one category in 2004. These charts use the "Artist" data for 2001-2003, and then "Artist/Animator" data for 2004 onward. Also, starting in 2009 the "Lead Artist/Animator" nomenclature changed to "Lead Artist/Tech Artist."

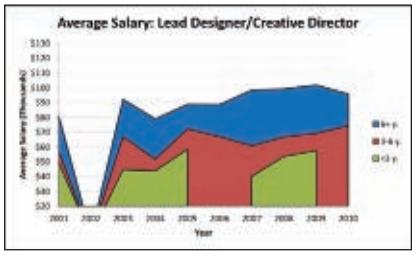
Game Design



↗ Unlike most of the other disciplines, it seems that the game design discipline enjoyed a relatively large infusion of new talent around 2006-2007, which has quickly dropped off since then.

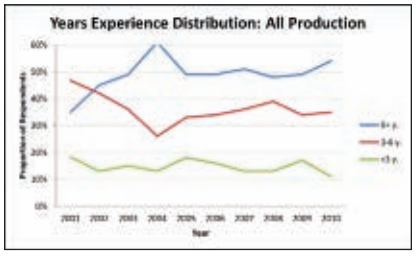




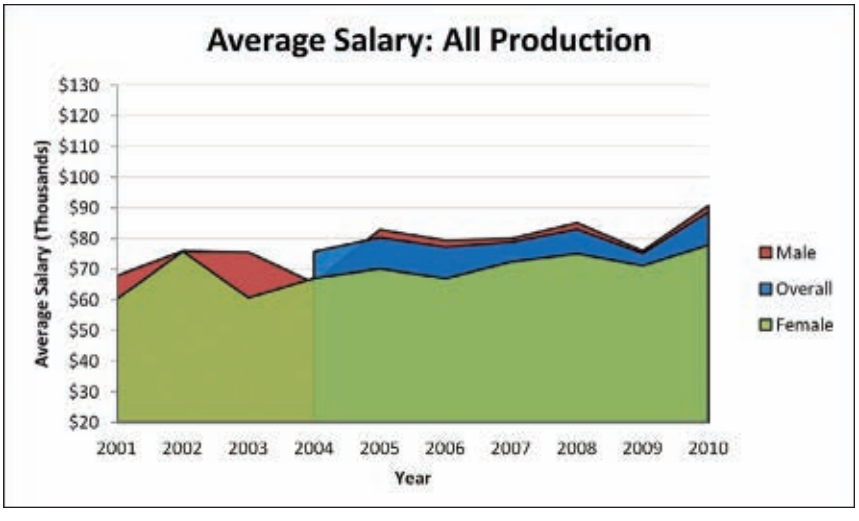


↗ Salaries for males appear to be steadily increasing, but for females salaries are not really trending in an obvious direction, while simultaneously showing lots of variance from year to year.

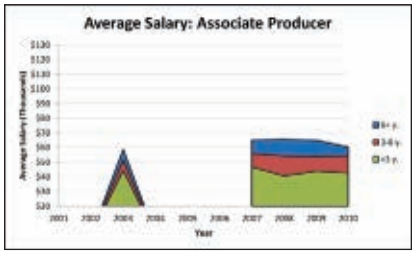
Production

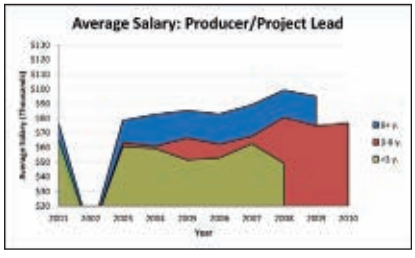


↗↘ Producers as a discipline have the most industry experience—those with more than six years of experience make up over half of



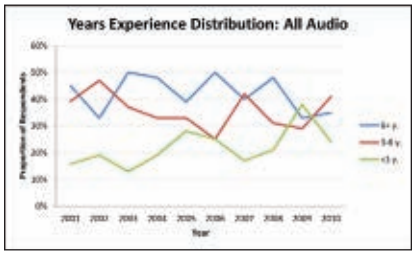
the population. Correspondingly, we find that there are fewer less-experienced workers in production than any of the other disciplines, and this is trending further downward.



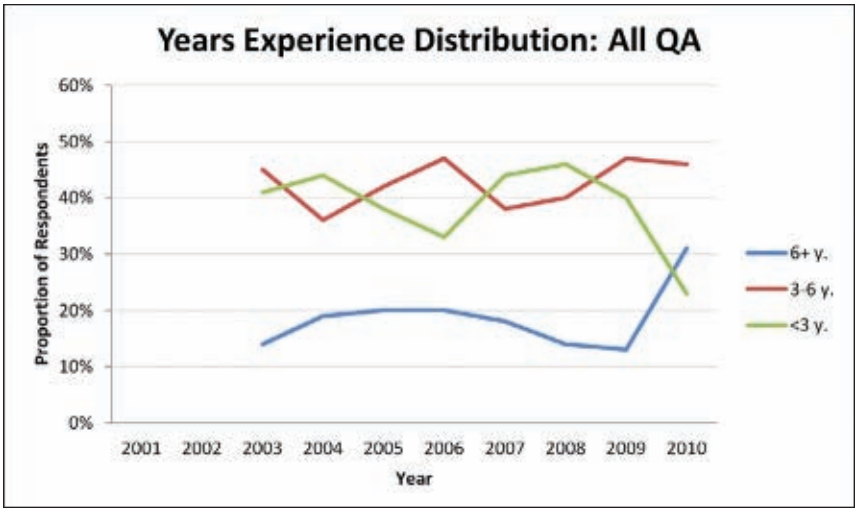


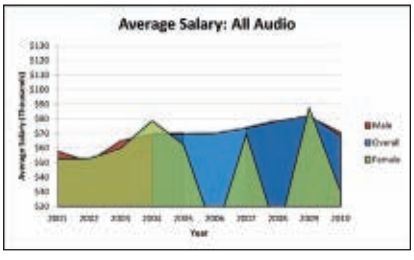
For experienced producers, there is a marked dip in 2009 which also corresponds with a dip in the Business category. This could be a reflection of the large number of studio layoffs around that period of time, or some other correction from the steep rise of Producer salaries from 2006–2008. Remarkably, salaries in some subcategories here appear to actually be trending downward, especially for associate producers.

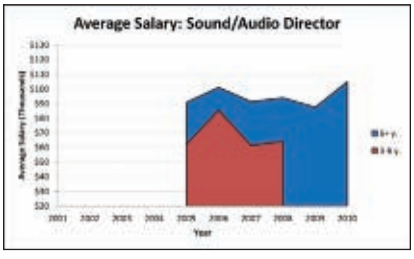
Audio

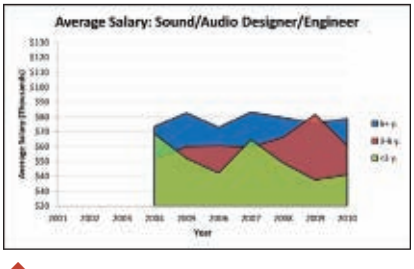


The audio discipline in terms of experience distribution seems to defy categorization - there is more variance from year to year with audio professionals than with any other category. This is almost certainly the result of the audio discipline being the least reported in the survey across the years.





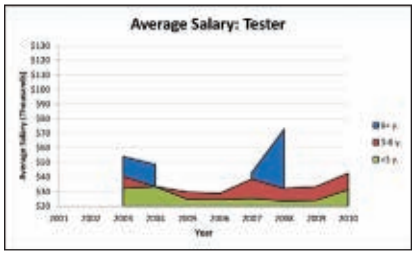


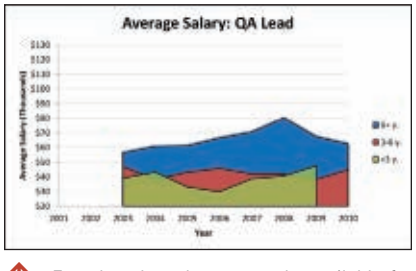


Audio salary data was not split into separate title divisions until 2004. Apart from QA, Audio is also the only other discipline where female salaries have exceeded male salaries within the past five years.

Quality Assurance

As might be expected, QA is largely made up of less experienced workers, although in 2010 we see this pattern start to reverse itself quite suddenly. Now, it seems as though workers are continuing to stay in QA for many more years than they used to.

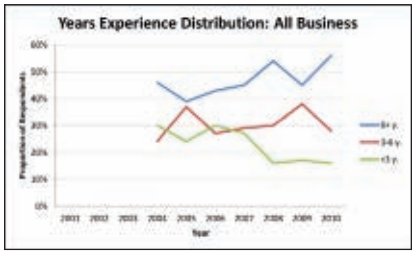




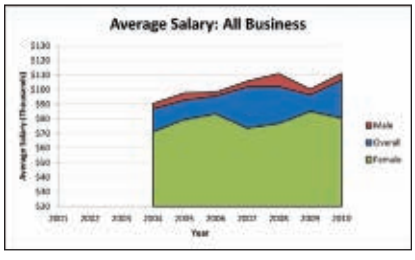
Female salary data was only available for five years, but for every year measured except

2007, female salaries exceeded male salaries in QA. QA Lead salaries for those with a lot of experience have dropped off quite dramatically since 2008.

Business



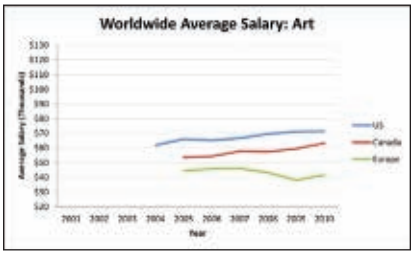
↗ The business category's experience distribution is most similar to that of production, but there are even more workers in the 6+ years category—a whopping 56% in 2010. It appears that the business category is becoming increasingly resistant to new talent over the years.

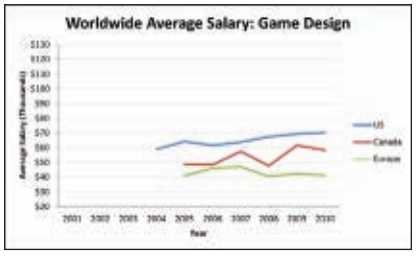


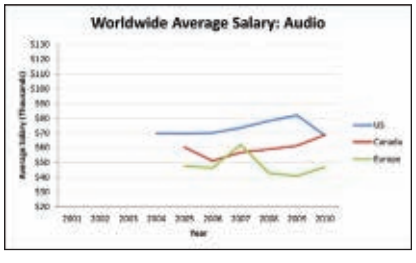
↗ The pay disparity between males and females is the largest in this category, roughly $10,000 larger than any other discipline. The average salaries are also larger than in any other discipline. Title-specific data was no longer reported in this category after 2008. There is also a lot of variance of salaries from year to year here, which may be partially explained by the fact that the job categories are much more broadly defined here than in other disciplines.

SALARIES WORLDWIDE

↗↘ These charts compare the average salary for each discipline between the U.S., Canada and Europe, all in American Dollars. For the most part, American workers enjoy better salaries. It appears as though the better paying the job is, the larger this disparity becomes—compare programming to art to QA, for example. It also seems as though European salaries in general have become particularly depressed since 2008. Most interesting is the giant difference in business and programming salaries between U.S. and Europe - it's around $35–40,000 in recent years.







Introducing

fmod® studio

The game changer for audio professionals

- New DAW inspired multi-track music and event editing interface
- Fully featured mixing desk with pro effects for mastering
- Create, add, edit and mix audio content **live** in game
- Integrated profiler captures game events and audio output
- Hardware control surface support, mixer snapshots, VCAs, sends, nested events, source control/multi user collaboration and more!

See us at GDC 2012, Stall 1532

Effects supplied by

iZotope

mDSP

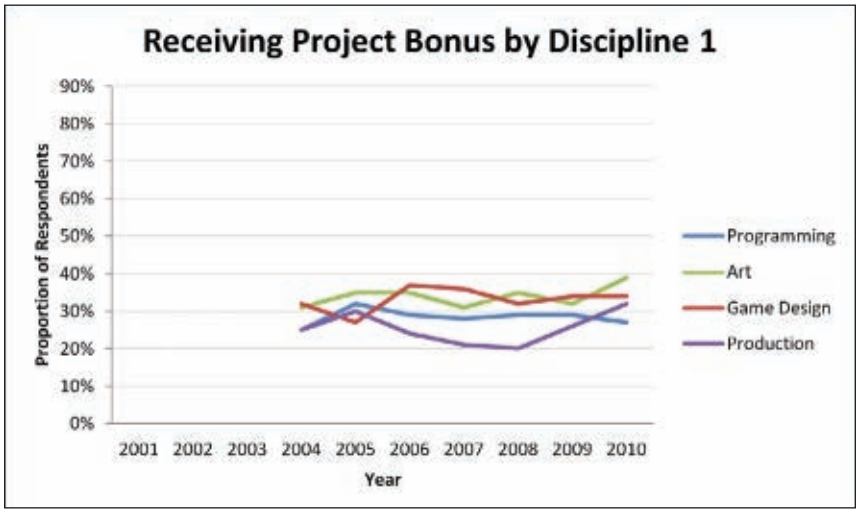
AudioGaming
NATURAL BORN INTERACTIVE

LITTLE ENDIAN

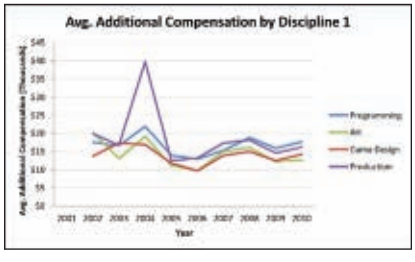
To get involved contact us at sales@fmod.org.

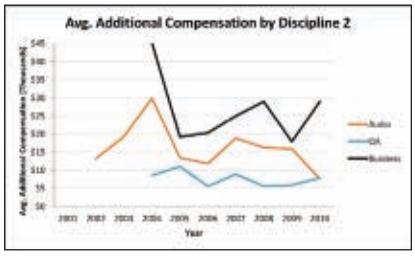
Firelight Technologies Pty Ltd.

www.fmod.org

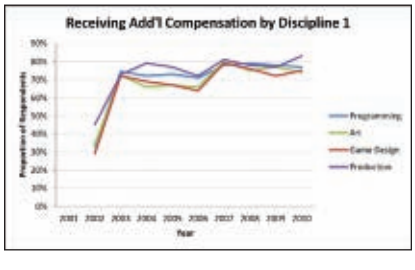


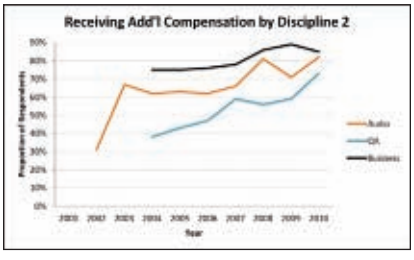
ADDITIONAL COMPENSATION BY DISCIPLINE



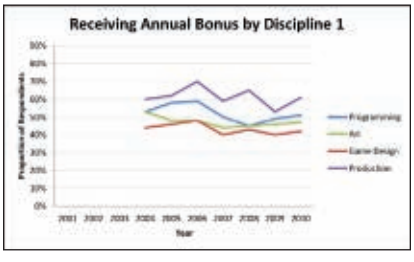


All compensation charts are presented here in pairs, for ease of reading. Across all disciplines, we see that most everybody is receiving some manner of additional compensation over time. Business and Production lead this category, with QA bringing up the rear. However, it does appear that all disciplines are also trending toward parity for the most part.

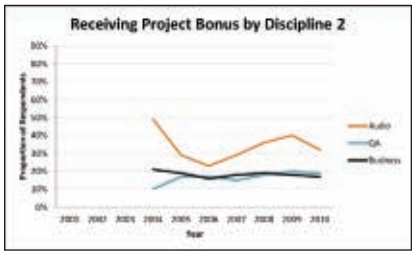




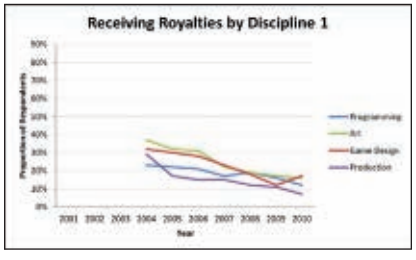
Business people receive the largest share of additional compensation, while QA receives the smallest.

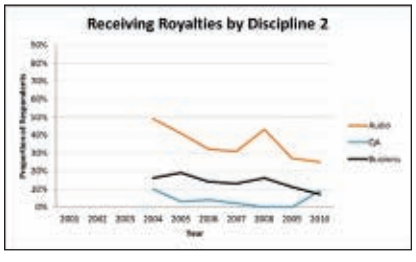


The remainder of these charts break down the manner of additional compensation received by each discipline. Production, Business, and QA most frequently receive annual bonuses, while designers are the least frequent recipients.

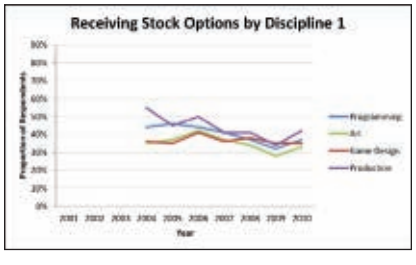


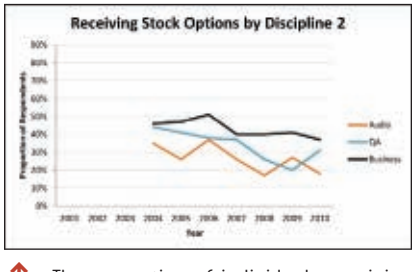
It seems as though project bonus distribution is essentially the opposite of annual bonus distribution. The rank of each discipline is for the most part reversed compared to the previous charts!





Royalties are not particularly common, and indeed they are quickly becoming even rarer for almost every discipline. However, across the board a higher percentage of audio workers receive royalties than any other discipline.



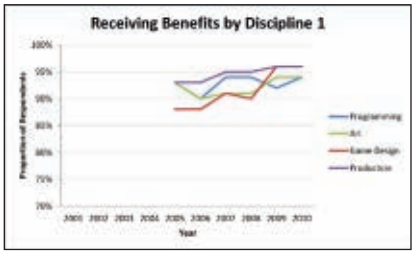


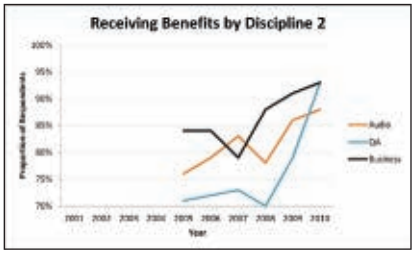
The proportion of individuals receiving

stock options is also trending downward across the board, except for a brief uptick for most disciplines in 2010.

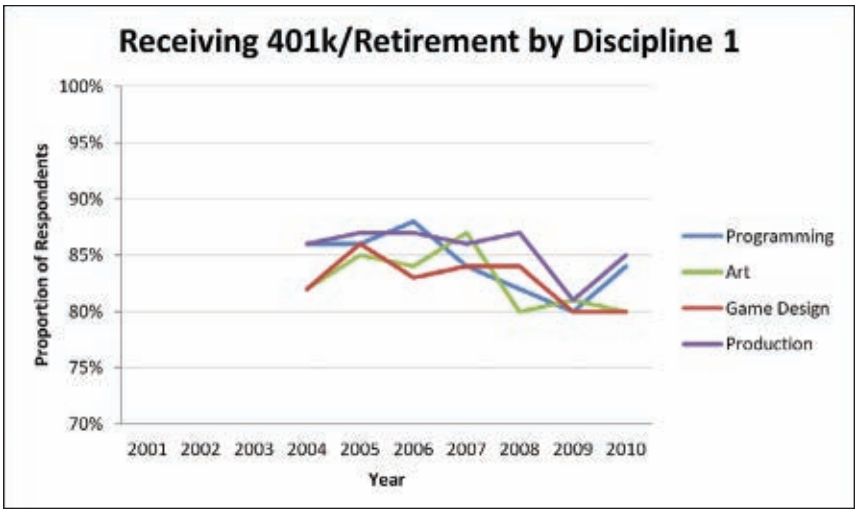
Profit sharing is perhaps the least common form of additional compensation overall. Business and audio workers are in the lead in this category, while QA and production bring up the rear.

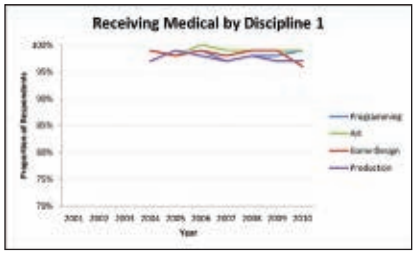
BENEFITS BY DISCIPLINE

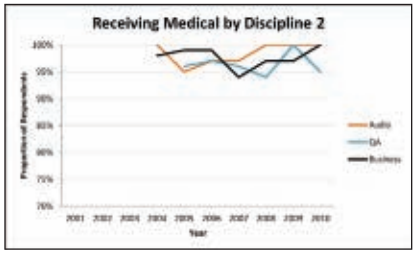


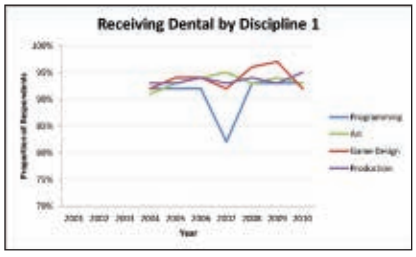


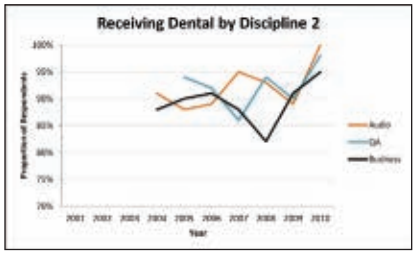
As with additional compensation,it also appears that more and more workers across the board are receiving some form of benefits and all disciplines are trending toward parity.



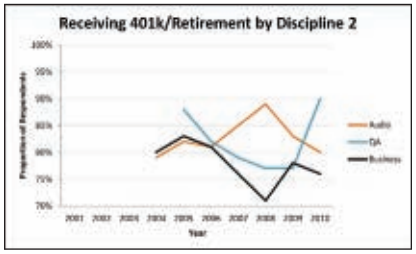








Nearly everybody in the industry receives some manner of medical and dental insurance. Dental insurance benefits are slightly less frequent than medical, particularly for business and QA.



Slightly fewer people receive 401(k) or some other retirement benefit. For most disciplines except QA, this type of benefit actually seems to be slightly trending downward.

SHOW ME THE MONEY

Over the last decade, though our industry has experienced ups and downs, overall performance appears to be roughly in line with the economy's rate of inflation. As expected, the groups, regions, and disciplines that already make the most money appear to be the most economically resilient and productive. Across nearly every discipline, in recent years we find that we are getting older as a whole—workers who have been in the industry for several years tend to stay there, and the number of workers who are relatively green seem to be thinning out. Is it a sign of a maturing industry? Strong competition? Overstaffing? Maybe some combination of all three.

ARA SHIRINIAN *is a game designer, currently working at DreamRift on its latest secret project.*

red5studios.com/jobs

UNREAL ENGINE 3 POWERS THE STYLISH, UNTAMED COMBAT OF ASURA'S WRATH

Capcom and CyberConnect2 have combined forces for the much anticipated brawler, *Asura's Wrath*, set for release on February 21. The action adventure game is set in an alternate reality called Gaea, blending science-fiction elements with Asian mythology, and featuring innovative gameplay that combines shooter action with grand-scale combat. Capcom has positioned *Asura's Wrath* to launch a new franchise, and they chose to develop their flagship game using Unreal Engine 3 (UE3).

Kazuhiro Tsuchiya, producer of *Asura's Wrath* at CyberConnect2, said of Epic's game engine, "This was actually the first time CyberConnect2 developed a game using Unreal Engine 3, and it provided the most effective environment for developing a game, while managing resources most efficiently."

Tsuchiya said UE3 equally reinforces the three core concepts of the studio: idea, knowledge and technology. He explains, "These are all important, and although one factor may weigh more than the other at one given point, overall it should be even as all are essential."

CyberConnect2 has been making games for more than 15 years, most notably fueling hit franchises such as *.hack* and *Naruto: Ultimate Ninja*. With such a distinguished record of development, the team had no problem diving into Unreal Engine technology to bring this new original IP to life.

The game itself follows the eponymous hero, the demi-god Asura, one of Eight Guardian Generals known for his victory against the evil Gohma. But in spite of his impressive military career, including slaying the monstrous Gohma Vlitra, Asura is framed as a traitor by his fellow generals. Asura's wife is killed and his daughter taken from him, leaving him expelled from Heaven, cast to earth, and left for dead. But after 12,000 years, Asura awakens and is consumed by the need for revenge and to reclaim his daughter.

Tsuchiya describes the storytelling approach of *Asura's Wrath* as more akin to a TV drama than a traditional game, and the developers worked to intertwine the mechanics of the game with the story. "During development, we wanted to bring something new to the action genre and to blend action and drama in a new way," said Tsuchiya. "We also wanted to blend cut scenes and action scenes seamlessly throughout the whole game.

"*Asura's Wrath* offers different play styles in each episode, rather than simply repeating the same action again and again in different maps. Instead, *Asura's Wrath* allows the player to be fully immersed in the game world and experience the drama Asura is going through. This means there are different game systems like combat action, shooting elements, quick time events, and so on, as Asura encounters different situations."

Tsuchiya's team utilized UE3 to incorporate a unique art style into the game's science fiction and Asian mythology-fused environments and characters, which he said "stylize our crazy over-the-top scenes." The game's many enemies include the Shinkoku Army and the hero's former mentor Angus, who trained Asura in his former life.

Tsuchiya said UE3 empowered his team to craft a truly immersive experience that entices the player into the game world the same way that Hollywood TV shows hook viewers. *Asura's Wrath* was conceived much like a serialized drama, complete with over-the-top situations, high stakes, and cliffhangers. The game's episodes, Tsuchiya hopes, will leave players wanting to come back for more. Using UE3, the team was able to add a sense of malleability to the game, and even cinematics can be interrupted, impacting the outcome.

The gameplay stands out from other action games. Battles, such as one waged between Asura and Angus, are huge, boss-style engagements. Gameplay involves combat with katanas and swords, as well as shooting and evading beams of energy bullets. Depending on the fight and Asura's rage, he can fight with as many as six arms or none at all.

Fights take on superhero proportions, with combatants being sent through walls and into the planet's atmosphere with powerful punches and kicks. Aiding players along the way is a Rage meter, which allows players to power up to take on the game's many enemies.

CyberConnect2 is hoping that good things come in threes. With UE3 bringing this visually impressive, action-packed game to life, the developer could have a third successful global franchise on its hands.

WWW.UNREAL.COM

Canadian-born Mark Rein is vice president and co-founder of Epic Games based in Cary, NC. Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award eight times, including entry into the Hall of Fame. UE3 has won four consecutive Develop Industry Excellence Awards. Epic is the creator of the mega-hit "Unreal" series of games and the blockbuster "Gears of War" franchise. Follow @MarkRein and @UnrealEngine on Twitter.

UPCOMING EPIC ATTENDED EVENTS

D.I.C.E. Summit
Las Vegas, NV
Feb. 8-10, 2012

GDC 2012
San Francisco, CA
March 5-9, 2012

Gadget Show Live
Birmingham, UK
April 10-15, 2012

EPIC GAMES

Please email **licensing@epicgames.com** for appointments

always online

NETWORK SYSTEMS IN INSOMNIAC GAMES' *OVERSTRIKE*

PETER KAO

/// Insomniac has a long history of creating online games, from RATCHET & CLANK: UP YOUR ARSENAL on the PlayStation 2 to the RESISTANCE franchise on the PlayStation 3. However, features were not always developed with online play in mind. In many cases, features were implemented offline first with online functionality added afterwards, usually by a separate team of network specialists. This process was both error prone and costly, often requiring major refactoring to get features working online. Beyond that, iteration on features in an offline environment would often break the existing networked implementation.

It was clear that we needed to change our development philosophy, and the change started with OVERSTRIKE, the studio's first multiplatform title. Online functionality is an integral part of the development process, and features aren't considered complete unless they work in an online environment.

ALWAYS ONLINE

/// One of the key problems Insomniac has faced on past projects was the need to maintain separate code paths for the online and offline implementation of a feature. This was a result of the networking API only being available in online games. OVERSTRIKE, however, provides an API that's always available, allowing us to use the same implementation in both online and offline modes.

OVERSTRIKE separates its network systems into two categories: a high-level API and a low-level network layer. The high-level API contains basic message-passing functionality and a synchronized object-management system known as sync host, which we'll discuss later. The low-level layer includes the data and interfaces for managing a true online session; network connections, NAT traversal, and hooks into platform-specific services (e.g. Xbox Live/PSN) all live in this layer.

This separation allows the high-level API to run independently of the low-level layer, making it safe for use in both online and offline games. Constructs such as clients, which were once only valid in online games, are now also available when offline.

Game logic interacts almost exclusively with the high-level API and has little knowledge of the low-level layer. For example, one of the functions in our high-level API allows the user to send a message to the server:

```
void SendToServer( ... );
```

When running an online game, this function determines when the network connection is being used to communicate with the server and sends the message over it. In an offline game, the local machine is considered the server, so calling this function will route the message back to the local machine, giving us behavior consistent with that off an online session.

SYNC HOST

\\\ As mentioned earlier, part of OVERSTRIKE's high-level networking API is a synchronized object-management system known as Sync Host. It's divided into two major systems: a server and a client. While the client runs on all machines, the server runs only on the machine designated as the host.

SYNC OBJECTS

\\\ The Sync Host server contains a list of all network-aware sync objects allocated for a game. This list is game-agnostic; the server has no information about the nature of the objects that have been allocated. Instead, the server treats a sync object as a block of synchronized data and identifies it using a unique object ID (see **Listing 1**).

Sync objects can be created by any client through an asynchronous call to the client API. The user begins by creating a custom block of data that defines the context necessary to create the game actor on all machines. This data block is an input argument to the client API function that requests the server to allocate the sync object.

The server responds by sending a message to all clients containing the newly allocated object ID and the custom data block. This, in turn, triggers a predetermined callback function on each client, with the data block and object ID as arguments. The client then uses the data block to create the game actor which it then associates with the given object ID. Since the object ID is shared across all clients, it can be used to reference the created actor across machines.

```
OBJECT ID = 5
Field 0:
  Size = 4
  Data = 0x01234567
Field 1:
  Size = 2
  Data = 0x0123

OBJECT ID = 7
Field 0:
  Size = 4
  Data = 0x01234567
Field 1:
  Size = 2
  Data = 0x0123
Field 3:
  Size = 3
  Data = 0x012345

OBJECT ID = 1000
Field 0:
  Size = 0
  Data = NULL
Field 1:
  Size = 1
  Data = 0x01
```

LISTING 1 SYNC HOST SERVER OBJECT LIST.

SYNC FIELDS

\\\ The state of a sync object is described by a set of sync fields, each of which is referenced by a field id. On the client, a field ID is associated with a particular memory address of the game actor. Since the server has no knowledge of the memory layout of the actor, the associations between field ids and memory addresses are entirely determined by the client (see **Listing 2**).

Once a sync object is allocated, updates to individual fields are routed through the server. The Sync Host client sends a message to the server that contains the object id, field id, and data for the particular field change. The server uses the message to make the necessary changes to its local state before forwarding the change to other clients in the game (see **Listing 3**).

AUTHORITY MANAGEMENT

\\\ Routing state changes through the server allows the server to resolve contention for a sync object among clients. The server does this by selecting a client, known as the authority, to give exclusive rights to update the sync fields of a given sync object. If a client other than the authority attempts to change a sync field, the server will automatically drop its request.

Normally, the server selects the client that requested the creation of an object as the authority. If the authority is dropped from the game, the server will then select a new authority from the remaining clients by default. Most sync objects in OVERSTRIKE use this form of authority management.

Sync Host also provides a second form of authority management, where a sync object is automatically destroyed if the authority drops from the game. This is useful for actors whose existence is tied exclusively to a particular client, such as the player's character.

JOIN-IN-PROGRESS SYNCHRONIZATION

\\\ Sync Host was originally developed to automate the process of join-in-progress synchronization. This is the process of updating the game world of a client that has connected to a game that has already started.

Routing state changes through the Sync Host server ensures that the server always has most up-to-date copy of each sync object, allowing it to service join-in-progress requests without actually being the authority of these sync objects.

Join-in-progress synchronization happens in two phases. The server first sends a series of sync object-creation messages to the new client. Each creation message contains the object ID and custom data block required to create the original sync object. The data block can be used by the incoming client to spawn the appropriate actor, just as the existing clients did when the sync object was originally created. Once the creation messages are sent, the server then sends the client a series of sync-field changes to bring the contents of the spawned actor up to date.

Once the synchronization is complete, the new client receives normal sync-field changes to objects, just as any other client would.

DECOUPLING FROM THE GAME ENGINE

\\\ As mentioned earlier, the Sync Host server is game-agnostic in the sense that it has no knowledge of the nature of the game actors associated with the sync objects it allocates.

One of the advantages of this decoupling is that it the server is relatively lightweight. Its memory footprint is largely dependent on the number of allocated sync objects and amount of data that is being synchronized. In practice, this overhead is only a small fraction of the memory requirements for the game actors themselves. In addition, this separation avoids the processing cost of running the game logic for these actors.

As a result, we can run the server on an external dedicated machine without incurring the cost of running the full game, or we can run it in parallel with the Sync Host client on a console without having to run a separate instance of the game world.

DECOUPLING FROM A NETWORK CONNECTION

\\\ As part of OVERSTRIKE's high-level network API, Sync Host is required to operate in both online and offline sessions. Instead of interfacing directly with a network connection, a Sync Host client writes its operations to an intermediate message queue and relies on an external system to determine the appropriate delivery mechanism for contents of the queue.

In an online session, a message from a Sync Host client is sent over a network connection to the host machine. The host machine receives the message and then writes it back to the server (see **Listing 4**). In an offline session, messages are instead directly sent to the server after being read from the queue (see **Listing 5**).

In either case, the client API is asynchronous on both the server and client machines. This allows the network synchronization logic to be identical on all machines and causes game code to be written with fewer timing guarantees, which in turn makes it more robust in a true online environment.

SYNCHRONIZED COMPONENTS

\\\ The Insomniac Engine utilizes a component-based architecture for constructing actors, where the logic for a particular actor is split across multiple sub-objects called components. An actor might have a component that stores data about its health, another to manage its state machine, and yet another to run the logic of its current state. A component can be allocated on more than one type of actor, providing a flexible means of code reuse.

Because an actor is built from multiple components, the data needed to properly synchronize an actor is also spread across its components. Each component now needs to map a particular field ID to a specific memory address that it owns without knowing about other components allocated on the same actor.

OBJECT ID = 5 — CLIENT VIEW

```
// Registered as sync field 0
float m_Health = 100.0f;

// Member variable, not synced
float m_Scale = 1.0f;

// Registered as sync field 1
uint16_t m_State = 1;
```

OBJECT ID = 5 — SERVER VIEW

```
Field 0:
  Size = 4
  Data = 0x42C80000
Field 1:
  Size = 2
  Data = 0x0001
```

LISTING 2 CLIENT VS. SERVER VIEW OF AN OBJECT.

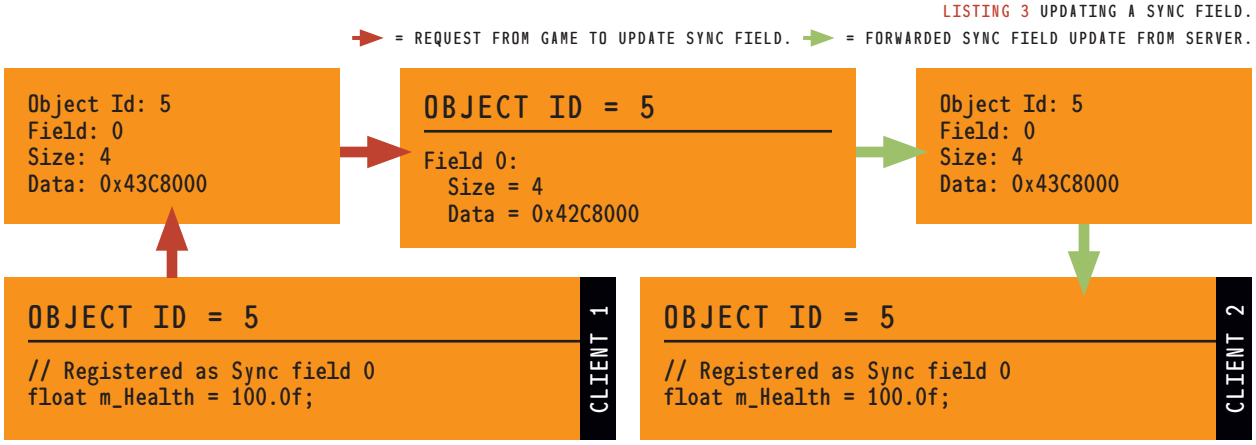
We accomplish this task by having a system that allocates a component ID for each component that has sync fields. Said components then have their component field IDs remapped to a field ID on the sync object (see **Listing 6**).

SYNCHRONIZED STATE MACHINES

\\\ One of the most useful network synchronization constructs in OVERSTRIKE is a specialized component called a Sync-State Machine. This component is used when an actor requires a high degree of fidelity when synchronizing its state.

The sync-state machine component registers a single sync field, which is a block of information called state transition data. Transition data contains the type of state to transition to, as well as additional information necessary to bootstrap that state. For example, if an actor has a generic state to play an animation, then the transition data might contain the ID of the animation to play (see **Listing 7**).

LISTING 3 UPDATING A SYNC FIELD.



IT'S YOUR CAREER.
PLAY IT FOR ALL ITS WORTH.

There has never been a better time to join IGT! After all, it's not just our award-winning games or the 3D online gaming technology that will reach out and grab you by the eyeballs. It's also the unrivaled, industry-leading career opportunity of a $2 billion-plus global gaming leader with the world's largest, most popular lineup of licensed games.

Interested? Stop by **booth #1414** in the West Hall of the Moscone Center (located in the Career Pavilion) to learn more about your next career move.

And be sure to connect with us online:

IGT IGT.COM/CAREERS /IGT You Tube /IGT /COMPANY/IGT

IGT BUILT TO PERFORM™

DRIVER STATES AND PROCESSOR STATES

\\\ In many cases, a state's behavior can be completely pre-determined based on the transition data that's used to initialize it. For states that aren't sufficiently deterministic, more fine-grained control is needed. Such states are split into three separate components: a processor, a local driver, and a remote driver (see **Listing 8**).

The processor component runs on both authority and non-authority clients and usually controls the visual and audio aspects of the state. By separating this functionality into a separate component and sharing between the authority and non-authorities, we simplify the task of making sure that such aspects of an actor are the same on all machines.

The local driver component runs only on the authority client, and it is responsible for running the decision-making logic for the state. This might be the decision tree of an AI or a system that polls device input to manipulate the player character. The local driver communicates changes to the processor component through a message called an action.

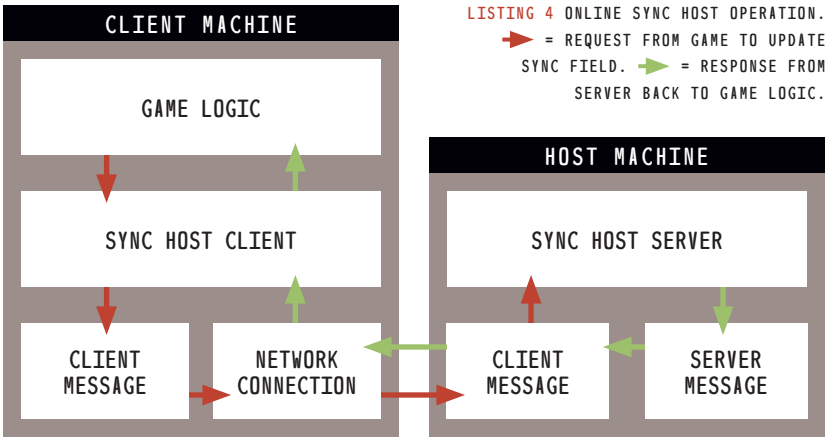
In order to communicate these changes to a non-authority client, the local driver can send a message called a state update. The non-authority allocates a remote driver, which reads and translates the state updates into actions that it then submits to its own processor component.

ACTIONS VS. STATE UPDATES

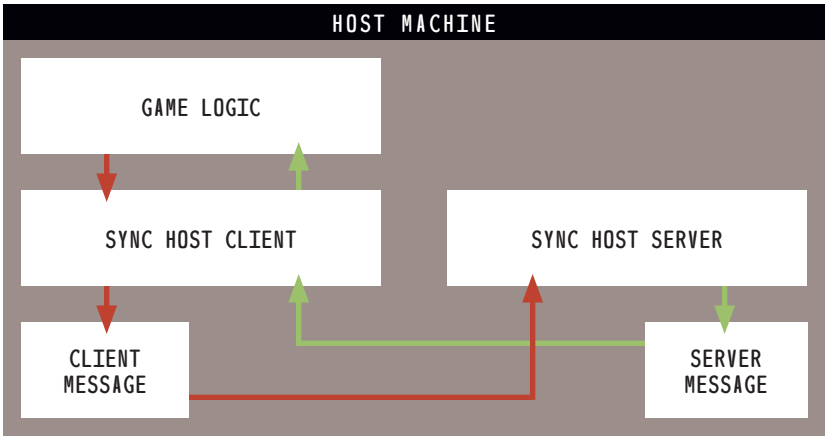
\\\ One might wonder why it's necessary to have separate constructs for actions and state updates. In many cases, there is a one-to-one correspondence between the two.

We make the distinction because the two types of messages serve different purposes. An action is a single operation that makes a specific change to the state. A state update contains the information needed by a non-authority machine to generate those changes. For example, in Listing 8, the action for the player walk state is a position delta submitted each frame. The state update, on the other hand, is a world position that the remote driver uses to smoothly interpolate the character's position by generating multiple actions.

Sending the actions directly over the network would be bandwidth intensive, since they are submitted each frame. Such a scheme would have very little tolerance for latency and packet jitter, as the quality of the simulation would be dependent on the non-authority machine receiving the actions at a predictable rate (ideally, the frame rate at which it's running). In addition, because the actions are position deltas, this system runs the risk of getting permanently out of sync, should any of the actions be dropped due to packet loss.



LISTING 4 ONLINE SYNC HOST OPERATION. = REQUEST FROM GAME TO UPDATE SYNC FIELD. = RESPONSE FROM SERVER BACK TO GAME LOGIC.



LISTING 5 LOCAL SYNC HOST OPERATION. = REQUEST FROM GAME TO UPDATE SYNC FIELD. = RESPONSE FROM SERVER BACK TO GAME LOGIC.

IMPROVING SYNCHRONIZATION FIDELITY

\\\ In some cases, changing states on a non-authority machine immediately upon receipt of a network update will cause artifacts in the simulation, such as animation popping, position jitter, and so forth.

For example, let's say that on the authority machine a player goes through three states. The player begins in the walk state and moves toward a low wall. The player then transitions to the vault state, where he plays an animation to mantle over the low wall. The player finally transitions back to the original walk state once the animation completes.

Unfortunately, packet latency and jitter can cause a non-authority machine to receive the state change to transition from vault back to walk before the animation completes. If we were to perform the transition at this point, we would see a noticeable discrepancy where the character pops from the middle of the vault animation directly into the walk animation.

To combat this problem, individual states have the ability to defer transitions until some condition is met (such as the completion of an animation). In the meantime, the state machine

In order to communicate these changes to a non-authority client, the local driver can send a message called a state update. The non-authority allocates a remote driver, which reads and translates the state updates into actions that it then submits to its own processor component.

HEALTH COMPONENT (COMPONENT ID = 0)

```
// Registered as field 0
float m_Health;

// Registered as field 1
float m_MaxHealth;
```

STATE MACHINE COMPONENT (COMPONENT ID = 1)

```
// Registered as field 0
uint8_t m_State;
```

WEAPON INVENTORY (COMPONENT ID = 2)

```
// Registered as fields 0-3
uint8_t m_WeaponTypes[4];
```

FIELD REMAPPING TABLE

COMPONENT ID	COMPONENT FIELD ID	TRUE FIELD ID	MEMORY ADDRESS
0	0	0	0x0F83ECA8
0	1	1	0x0F83ECAC
1	0	2	0x00E3FAC0
2	0	3	0x0AC15FB0
2	1	4	0x0AC15FB1
2	2	5	0x0AC15FB2
2	3	6	0x0AC15FB3

LISTING 6 COMPONENT FIELD ID REMAPPING.

maintains a queue of new states that the existing state can transition to once it is ready.

ALTERNATIVES TO SYNC HOST

\\\ Not all systems in Overstrike are synchronized through Sync Host. Actors or events that are short lived tend to use basic network messages to perform synchronization. Systems that are extremely sensitive to network latency also tend not to use Sync Host, as doing so would incur the delay of routing the message through the server.

For example, most projectiles in OVERSTRIKE aren't created as sync objects. Bullets tend to be extremely short lived and have no state to update once they are created, so there is little justification for allocating a sync object for them. Minimizing latency is crucial, as projectiles are created rapidly and en masse. Instead of using Sync Host, projectiles are immediately spawned on the client that chose to fire the weapon, and a basic network message is then broadcast to other clients to spawn the same projectile.

OVERSTRIKE's damage system is also synchronized using basic messaging. The event-driven nature of damage messages means there is no persistent state to speak of that needs synchronization. Actors that take damage also need react in a timely manner, thus minimizing latency paramount.

BANDWIDTH OPTIMIZATION WITH MESSAGE DEFINITIONS

\\\ Due to their sheer quantity, projectiles tend to account for a significant portion of the network traffic in the game, making it necessary to optimize their bandwidth usage. Message definitions were created to simplify the task of defining and optimizing these kinds of packets.

A message definition can be thought of as a dictionary of key-value pairs, where the key is a field identifier and the value is metadata about that field. In its most basic form, a field on a message definition is registered using a string name and its size (see **Listing 9**).

Once a definition is created, a message instance can then be constructed using the definition, at which point fields can be set or retrieved. Once the fields have been set, the contents of a message instance can then be serialized to a network buffer using the `Pack()` function. This function, however, will only write fields that have been set. This allows us to use a message definition with more fields than we may need without incurring the overhead of sending those additional fields (see **Listing 10**).

Message definitions also support optional compression/decompression callbacks when registering a field. The compressed size of the field is inferred from the size of the parameters to the provided callbacks, which are automatically triggered when a message instance write to or reads from a network buffer. This allows the programmer to quickly define the compression policies for the fields of a given message without having to write a custom serialization function (see **Listing 11**).

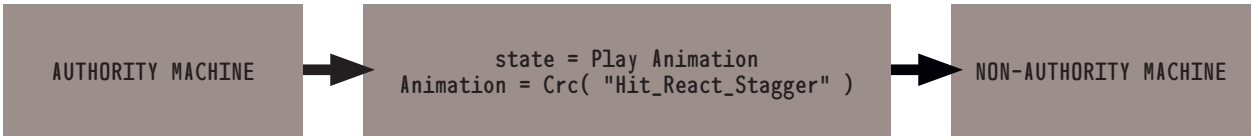
MOVING FORWARD

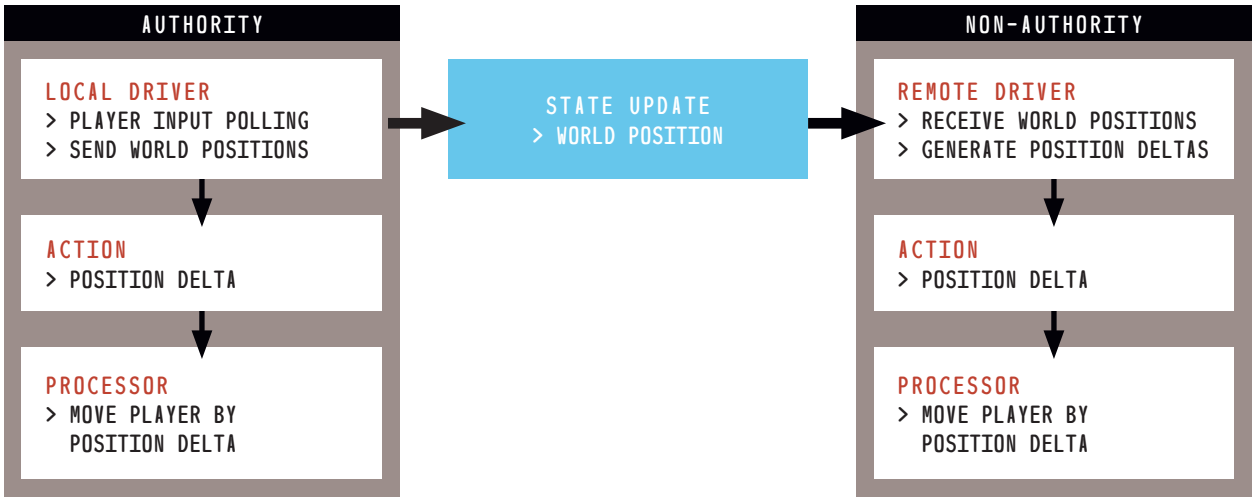
\\\ OVERSTRIKE provides a robust and flexible set of networking technologies while relieving programmers of the complexities unrelated to synchronizing the actual feature they may be working on. By making our network API available in both online and offline sessions, we no longer need separate code paths for each. Sync Host provides a powerful state synchronization system without burdening the programmer with the details of object ID allocation, authority management, and join-in-progress synchronization.

These tools have enabled our programmers to shoulder the responsibility of ensuring that their features work online. Insomniac no longer has a dedicated multiplayer team; on OVERSTRIKE, we are all network programmers.

PETER KAO *is a senior gameplay programmer at Insomniac Games with an emphasis on network systems. Prior to OVERSTRIKE, he has contributed to all three RESISTANCE games as a network and multiplayer programmer.*

LISTING 7 BASIC SYNCHRONIZED STATE TRANSITION.





LISTING 8 STATE DRIVERS AND PROCESSORS FOR THE PLAYER WALK STATE.

LISTING 9 SAMPLE MESSAGE DEFINITION.

```
// Vector structure, for reference
struct Vec3
{
  float x;
  float y;
  float z;
};

DECLARE_MESSAGE_DEF(BulletMessage)
  REGISTER_MESSAGE_FIELD("AimVector", sizeof(Vec3));
  REGISTER_MESSAGE_FIELD("Speed", sizeof(float));
END_MESSAGE_DEF(BulletMessage);
```

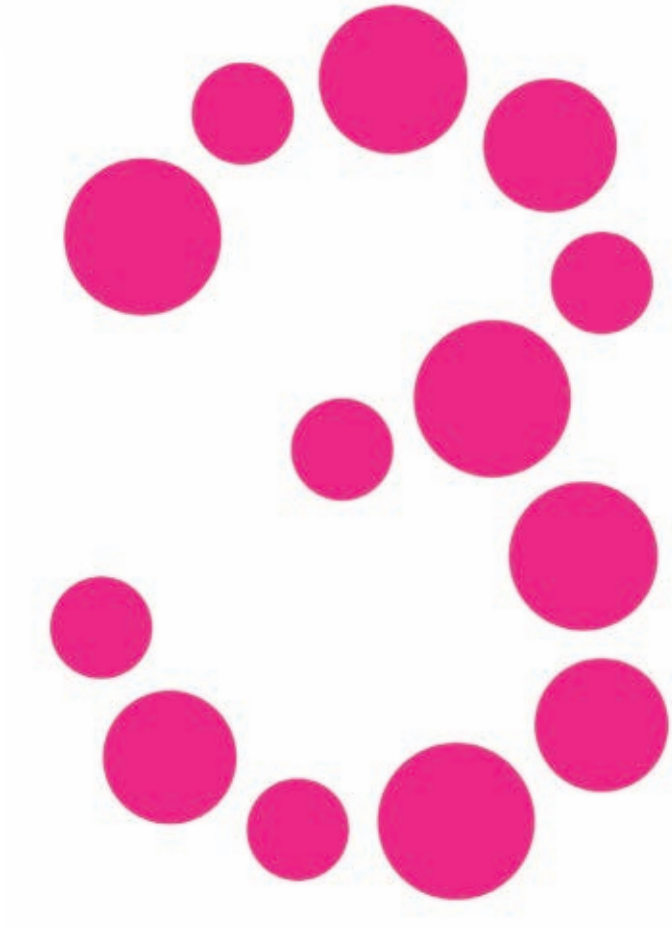
LISTING 10 MESSAGE INSTANCE CONSTRUCTION.

```
// Construct message instance
MessageInstance message(GET_MESSAGE_DEF(BulletMessage));

// Set aim vector
Vec3 direction;
direction.x = 1.0f;
direction.y = 0.0f;
direction.z = 0.0f;
message.SetField("AimVector", direction);

// Allocate some buffer space for network serialization
// Buffer is 12 bytes, not 16, since only the "AimVector"
// field has been set
uint32_t buf_size = message.GetRequiredSpace();
void* buf = alloca(buf_size);

// Write to network buffer
message.Pack(buf, buf_size);

// Read from a network buffer
message.Unpack(buf, buf_size);
```

LISTING 11 MESSAGE DEFINITION WITH COMPRESSION.

```
// Unit vector compress/decompression functions
void CompressUnitVec(uint16_t* dest, const Vec3* src);
void DecompressUnitVec(Vec3* dest, const uint16_t* src);

// Float compression using half (16-bit) precision
void FloatToHalf(int16_t* dest, float* src);
void HalfToFloat(float* dest, int16_t* src);

// Bullet message with compressed fields
DECLARE_MESSAGE_DEF(BulletMessage)
  REGISTER_MESSAGE_FIELD_PACKED("AimVector",
CompressUnitVec, DecompressUnitVec);
  REGISTER_MESSAGE_FIELD_PACKED ("Speed", FloatToHalf,
HalfToFloat);
END_MESSAGE_DEF(BulletMessage);

// Construct message instance
MessageInstance message(GET_MESSAGE_DEF(BulletMessage));

// Set aim vector
Vec3 direction;
direction.x = 1.0f;
direction.y = 0.0f;
direction.z = 0.0f;
message.SetField("AimVector", direction);

// Set speed
float speed = 100.0f;
message.SetField("Speed", speed);

// Allocate some buffer space for network serialization
// Packed size is 4 bytes after compression
uint32_t buf_size = message.GetRequiredSpace();
void* buf = alloca(buf_size);

// Write to network buffer
message.Pack(buf, buf_size);
```

umbra



UMBRA 3 RENDERING OPTIMIZATION

"Umbra's technology is playing an important role in the creation of our next universe, by freeing our artists from the burden of manual markups typically associated with polygon soup."

Hao Chen
Senior Graphics Architect

BUNGiE

UMBRA 3.COM

balancing a big huge rpg

LESSONS FROM *KINGDOMS OF AMALUR: RECKONING*

IAN FRAZIER

If you've ever played an open-world RPG, you know that these games are big. Really, really big. KINGDOMS OF AMALUR: RECKONING is no exception.

This is a game with hundreds of hours' worth of quests, dozens of combat abilities and skills, 40 different "destinies" (essentially classes), an obscene amount of dynamically generated loot, and several hundred hand-crafted unique items. And, being an open world RPG, content can all be consumed (or ignored) by the player in countless different ways.

With the sheer amount of content in a game of this size, the variety of character build options available in an RPG, and the complete freedom to go anywhere and do anything that an open-world game provides, we knew that overall game balance was going to be one of our biggest challenges. As there are very few games of this type being made, I thought it would be valuable to share with you—especially the systems designers among you—some of what we learned during the development of RECKONING. Note that much of what I'm going to talk about could be applied to more linear RPGs or even MMOs just as easily as it could to other open-world RPGs. >>>

balancing a

GETTING STARTED

/// At the start of developing a new RPG, it's reasonable to ask "where do I begin?" The temptation can be strong to immediately dive in to Excel and start plotting out tables of values for everything from XP to enemy armor, but plotting out actual values before you've worked out all the rules for the parameters and underlying math that make sense for your game is setting the cart before the horse. Instead, let's start at the beginning.

SETTING UP THE BASICS

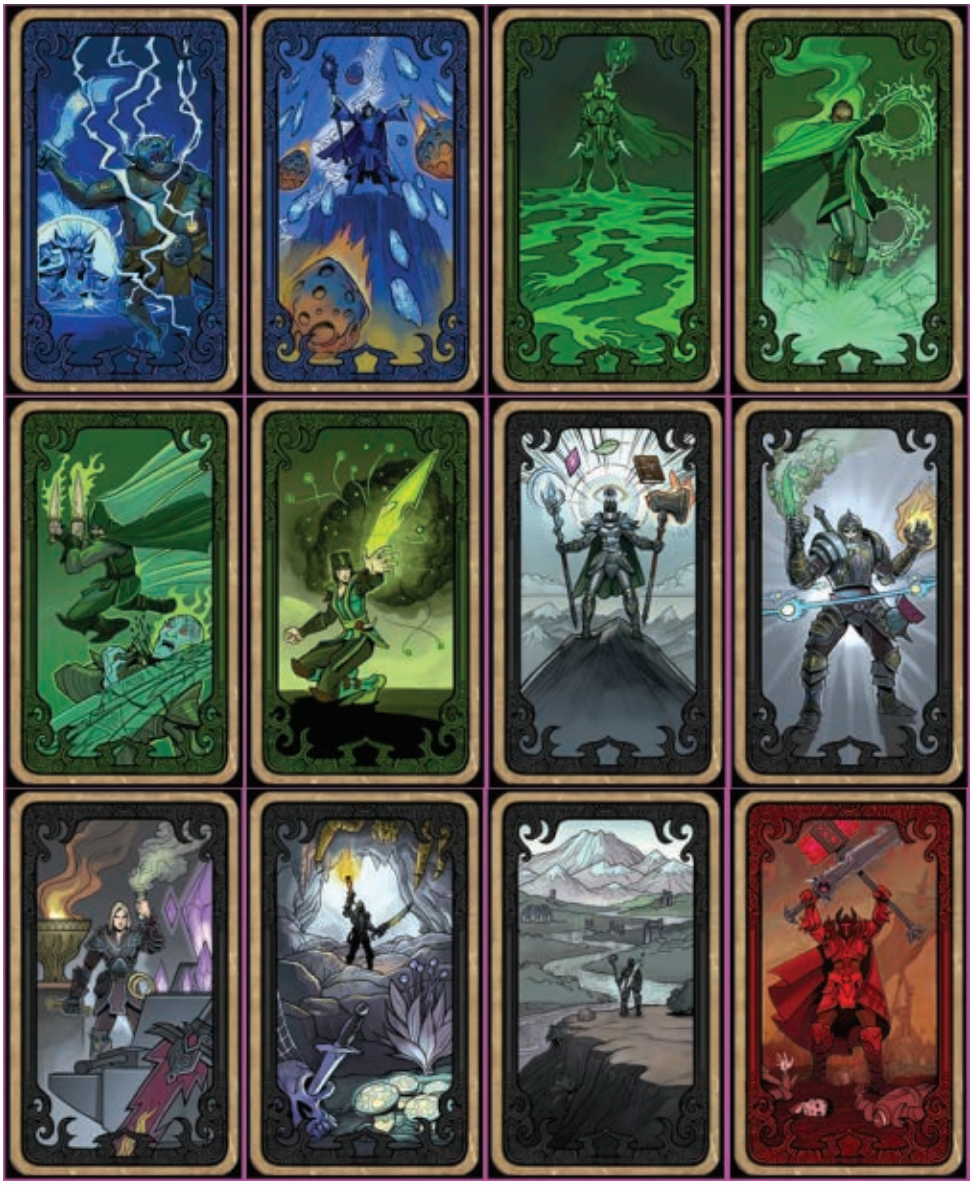
/// The first step toward balancing an open-world RPG is understanding what the core components of your game will be, and how they will relate to one another. In other words, know the variables before you start thinking about the numbers. Before production begins, you want to know what your damage types are, how damage can be mitigated, what the core attributes of the player character and NPCs are (health, mana, dexterity, and so forth), by what means these values can be buffed/debuffed, what rounding rules you're going to use, and what the limits are on buff operations. Also, don't forget to plot out how buffs will affect projectiles fired by either the player or NPCs.

Damage.

What are your damage types? Can all damage types be mitigated? Are there any damage types unique to the player or to NPCs?

Damage Mitigation.

Are you using percentage resistances or an abstraction (such as an armor rating)? If the latter is true, does that apply only to certain damage types (like physical damage) or to all of them? How do percentage resistances work against state-based effects such as slowing or stunning? Does 50% Stun Resistance mean that your chance to be stunned is reduced by 50% or that you'll still always get stunned but the duration of the stunning effect will be cut in half?



Attributes.

What are the basic parameters affecting your characters? While there is some room for adjustment over the course of development, the overwhelming majority of your basic parameters for both the player and NPCs should be determined from the start, so that you can more effectively plan how these elements will affect each other. Bear in mind that you'll also want to know up front what sorts of values these attributes will have (integers or floats, etc.) Having 2.5 health regeneration per second may make sense, but having 27.9 health probably does not.

Buffs/Debuffs.

How are you going to buff the various parameters on your characters? Flat mods (+10 health), or percentage (+25% health)? Do you buff values on the gear the player is wearing or on the player herself? What's the order of operations on these modifications (do we add flat modifiers and then multiply by the percentage modifiers, or the other way around)? What parameters can be modified with negative values? You also need to answer the all-important question of how multiple simultaneous buffs/debuffs to the same parameter will stack.

Rounding.

How and when will you round your fractional values? As a general rule, the easiest way to handle all rounding is truncation, but there may be instances where this doesn't give you the end result you want. In either case, it's important to call out up front exactly how and when rounding will occur.

Limits.

What limits do you want to set on your parameters? Will you allow damage output to be debuffed down to zero, or do you want to set a floor value that ensures players will always be able to

deal some amount of damage? Can any of your parameters go negative, and if so, what happens when they do? Will negative amounts of health regeneration cause a character to slowly lose hitpoints? Will negative amounts of fire resistance make a character take extra damage from fire attacks? Are you going to allow the player (or NPCs) to be 100% resistant to certain types of damage or are you going to hard-code in a limit preventing that from being possible?

Projectiles.

Buffs on a character in melee combat are one thing, but projectiles can cause unique issues. If I launch an arrow at a Kobold and then put on my Helm of Kobold Slaying (+10 dmg vs. Kobolds) before the arrow finds its mark, do I expect the damage buff to be applied? Or are all conditional buffs checked and applied at the moment I fire the projectile?

SHARE THE RULES

/// After answering the questions above and laying out the basic rules for the game and its underlying math, it's important to document this information in a single comprehensive form that's accessible to the team. Then make sure that everyone invested in implementing or tuning the game's systems reads and understands the document before you proceed, as content you'll build later in production will rely on this information. The greater the understanding of the systems behind the game, the easier it will be for other designers to create content based on those systems.

TEST CASES

/// After establishing, documenting, and sharing the basic rules for your system, the next step is to plot out a variety of test cases that describe the expected results of those systems in actual gameplay. You'll want test cases for everything from the simplest situations (the player attacks an enemy with 100 HP using a sword that deals 10 damage) to the obscenely complex (several stacking buffs and debuffs on both the player character and the enemy, where some of the buffs are applied only when specific conditions are met).

The most important thing about these test cases is that you need to keep doing them. Don't just set them aside after you've verified that the core systems are built to spec. Over the course of a project, a lot will change. New systems will emerge, new content will be built that stretches the systems in ways that weren't originally envisioned, and the increasing level of overall complexity in the game can cause older functionality to break. Because of these inevitable issues, it's vital to revisit your test cases on a regular basis to ensure that the systems continue to function as designed.

The reason it's so important to keep running these tests is because, as we discovered late in the production of RECKONING, subtle changes to these core systems won't cause any obvious breakage during general play (and are thus unlikely to be caught by QA testers), but can cause serious balance problems in various edge cases (like when the player is buffing the same parameter from a dozen different sources). In our case, we discovered some changes to the way we stack buffs very late in production, which necessitated the rebalancing of a large amount of content because it was too late in production to risk changing

fundamental code. Please, save yourself a lot of pain and run test cases on a regular basis.

FIGURING OUT THE NUMBERS

/// After determining all your basic variables, how they're modified, and how they play together, we can now start thinking about actual numbers. Most of the numbers we use in game systems, such as health, damage, and experience points, are entirely arbitrary. What matters is the relationships between those values. We can make the player's starting health 853,521 if we want to, provided that the starting enemies are dealing 50,000 damage. Because the base numbers are arbitrary, it can be difficult to establish your initial values. My solution may be unusual—I start with UI. I look at the UI and think about what values we're actually going to let the player see and which ones are going to be behind the curtain.

In RECKONING's case, the numbers we show are values on the player himself: how much health he has, how much damage his hits are dealing, and so on. Thanks to some early usability feedback, we knew that we didn't want to show NPC health or damage numbers. Because we knew this, we could start establishing the baseline values with things like player health and player weapon damage, and then base hidden values like enemy damage and enemy health based on their relationship with the displayed values. So, for example, we would say that the player is going

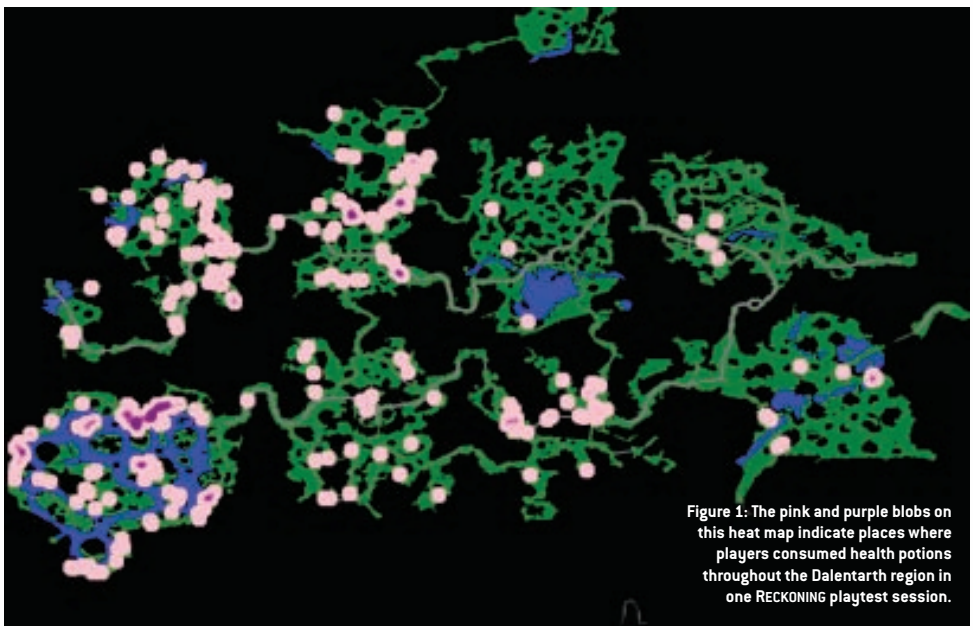


Figure 1: The pink and purple blobs on this heat map indicate places where players consumed health potions throughout the Dalentarth region in one RECKONING playtest session.

Scotland.
Home to software innovation and highly skilled graduates.

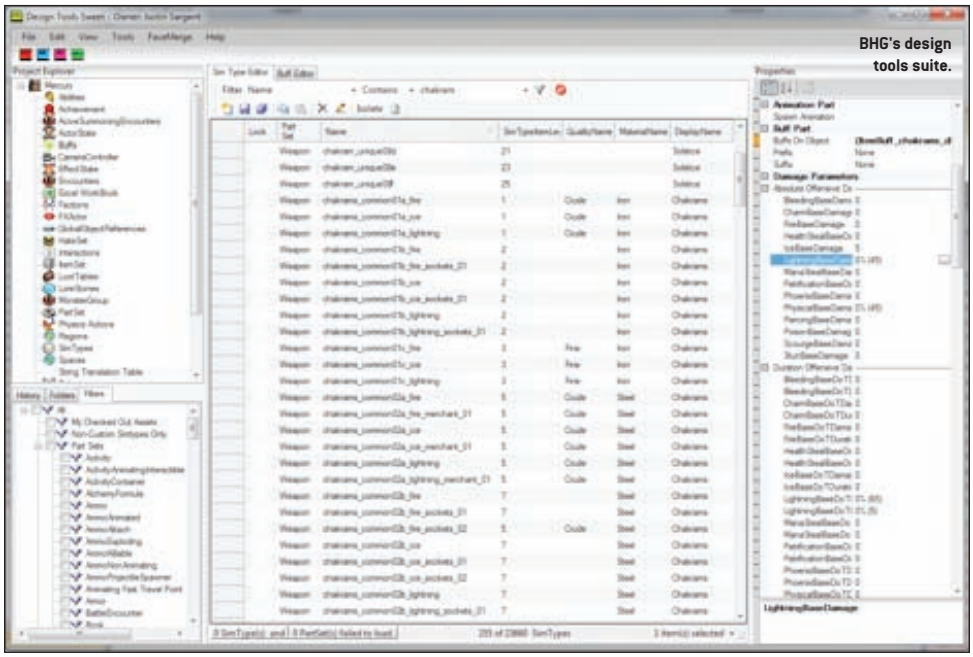
We could teach you a thing or two.

We've got quite a reputation for education, technology and invention. With the highest concentration of universities in Europe, we have produced innovations from the ATM to MRI scanning. Our education system has been the platform for our contribution to the world. It has helped build one of the world's most competitive, reliable and dynamic technology infrastructures, with particular strengths in implementing intelligence in to communication and software technologies. Whether it's informatics, cloud computing or games development, we strive to set standards. Our passion for success and hunger to win, combined with our world-class academic institutions, outstanding research and superb facilities make Scotland truly irresistible location. We can develop your products and help shape your business. And that's what makes Scotland such a secure investment.

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

SCOTLAND. SUCCESS LIKES IT HERE.

SCOTTISH DEVELOPMENT INTERNATIONAL



BHG's design tools suite.

to start with 50 health, and since we want an enemy to have to hit the player 10 times to kill him, that naturally gives us a starting point of 5 damage for enemy attacks.

Once you've established the broad umbrellas you want to start with, such as player health and weapon damage, I recommend establishing baselines using your most "average" content first, then basing everything else off of that. For us, that means determining how much health a Rogue (the middle-ground build between Warrior and Mage, with a balanced amount of health and mana) will have at any given level. And for damage output, we plotted how much damage a level 1 longsword would do. This gave us a starting point to base everything else off of. In general this helps get everyone on the same page with regard to what the base combat experience is. For instance, say the average enemy dies in 5 hits, and the player dies in 10 hits from the average enemy. In turn, this established a frame of reference for determining what stats a boss enemy might have or what sort of damage a "weak but fast" weapon should deal.

As for how to determine the actual numbers for whichever parameters you decide to display to the user, there are really only two things to bear in mind:

Aesthetics: As I said earlier, you could start the player with 853,521 health ... but players would roll their eyes at you. I'd recommend starting with nice round numbers, such as 100, that will let players see some digit-expanding improvements to their characters in the early game. Who doesn't love going from a 7 damage sword to a 12, or from 93 health to 109?

Granularity: Although you can make most of these initial values whatever you want, you do need to bear in mind the granularity of those values. If the player starts with 5 hitpoints, unless you want to allow fractional health/damage values, your fights are going to have to get resolved in only a handful of 1 damage hits, or have lots of healing/regen going on. You'll want to make these initial values big enough to give you a bit of breathing room with the pacing of your combat.

CONSTRAINING YOUR CONTENT

/// You've established the basics, communicated them to the team, and plotted out your starting variables. Now what? Well, now's the fun part: building the content! It's time to design all the swords, bows, monsters, abilities, and so forth, and start determining the hard numbers on all of them, from the damage of an iron battle-axe to the mana cost of Magic Missile. Before you craft a thousand swords, though, you should establish some constraints.

PARAMETER CAPS

/// There are some values that you're going to want to carefully constrain. For instance, if you allow the player to accumulate 100% damage resistance, she's going to be invincible, which stops being fun rather quickly. If you allow the player to have a -100% cooldown and/or -100% mana cost, you've suddenly lost a major resource management aspect of gameplay, which provides a brief surge of fun but is quickly followed by eternal boredom. Thus, it's a good idea to decide up front exactly how high and how low you want each variable to get, and clearly document that. Does the game simply become less fun when your damage resistance goes from 60% to 70%? If so, you might want to consider capping it at 60%.

After you've decided what you want your parameter caps to be, you need to decide how you want to implement them. I propose two methods, which can be used either independently or in conjunction.

OPTION 1: HARD-CODE THE CAPS

/// The easiest and safest way to prevent individual values from being buffed/debuffed outside of the optimal range is by directly setting floor/ceiling values for these parameters in code.

There are at least two downsides to this solution, however:

1. You're likely to incur player wrath when they combine multiple effects to make an über effect, and then learn that this doesn't actually work. Few things make a player happier than feeling smart, but few things irritate them more than discovering that a game won't let them be smart.

2. You'll likely run into specific situations where these caps prevent you, the designer, from doing something you want to do. Want to buff the player's resistances to 100% for a few seconds to ensure they don't die during your cool scripted sequence? Oops, turns out they can't get above 60%! Long story short: If you do go down the hard-coded cap road, make sure you've got an easy means to circumvent those caps with dev-side buff operations when needed.

OPTION 2: DESIGN WITH CAPS IN MIND

/// The other way to deal with parameter caps is to design all your content with the caps in mind. This approach boils down to plotting in advance exactly what different elements of the game will be "allowed" to buff different variables, then settling on individual parameter caps for the buffs coming from each of those sources, so that the grand total of all the stacked buffs can't cross your global cap for that parameter.

For instance, let's say you never want the player to have greater than 60% cooldown reduction. If the only things in the game that are allowed to grant cooldown are weapons, helmets, boots, and abilities, then we can set up sublimits for those types of equipment. We can draw up specific limits and say that we'll allow a max of 10% cooldown reduction on two-handed weapons, 5% on one-handed weapons, 5% on helmets, 5% on boots, and 40% for a single passive ability at max rank (with no other abilities allowed to affect this parameter). Assuming these buffs stack additively, we've

Top: Quest XP chart. Bottom: This chart indicates the minimum frames of animation needed for each basic attack in RECKONING. From here we can calculate the approximate DPS of each weapon and use that to balance the base damage of weapon classes.

now guaranteed that the player will never get above 60% cooldown reduction even if he goes out of his way to get the best possible cooldown gear available.

As you can imagine, policing restrictions like this can be quite time-consuming and prone to human error. Fortunately you can combine this approach with option 1, setting extremely lenient parameter caps in code to catch the truly egregious scenarios, but relying on option 2 to prevent players from ever actually hitting those invisible walls.

THE BEAUTY OF TELEMETRY

/// Now you've got all your rules and constraints in place, and you're merrily building content and making balance tweaks to it as you go. But how do you know if your balance tweaks are actually working? Did your decision to nerf the Warrior-Rogue hybrid just make the pure Rogue completely ineffective? Did your change to the XP-to-level curve make the early game more fun but plunge the mid-game into the depths of tedium? In addition to the usual response ("play the game, play the game, play the game!"), there's also another answer: Use telemetry data.

On RECKONING we used EA's proprietary Juice system for gathering data on the game as it was played—by QA, the development team, and in extensive multiday testing sessions with focus testers. This system, when combined with our internal tool for aggregating and studying the data, allowed us to use actual player data in our balancing efforts. By capturing key facts about what players are doing and what states they're in while doing it, you can easily evaluate where the balance of various pieces of content currently stands. You need quite a bit of user testing to make this viable, but with a huge open world game, there's no getting around the need for a great deal of playtesting.

Over the course of development, we gathered a large amount of data from our players, some of which proved to be useful and much of which did not. Here are some of the Events and States that we found especially useful (I'll leave out the ones that are entirely specific to RECKONING).

EVENTS

/// Each time one of these events occurs, the system stores the timestamp, the location of the event, and a list of all currently active states, as well as other event-specific data.

Player death.

If the player dies, we store where she was and what object or actor killed her. Our QA spy tool then plots these event coordinates on the game map, so we can see a heat map that will show us where players are consistently dying.

Potion use.

When the player drinks a health or mana potion we want to know about it. Like player death, this is a great event to see heat-mapped as it gives us a good sense of overall difficulty in each area (See Figure 1).

Player movement.

This is a simple event, polled every few seconds, that notes the player's current coordinates. We then can view these events on the map and see which locations are never (or seldom) being discovered and what locations draw the most attention.

Player kills NPC.

This event reports the NPC that the player killed as well as the particular attack or ability that served as the killing blow. Generally when you see the same ability or weapon being used overwhelmingly by a particular build/class, you can assume that either it's too good, or you haven't given the player enough options.

Level up.

This reports when the player has gained a level. We can use this event to see the rate at which players are leveling up as they move through the world. When viewed along with skill use, quest complete, and player kills NPC events, this gives us a good sense of how effectively different types of players (explorers vs. talkers vs. warriors) are leveling up.

Item acquired.

This reports whenever the player acquires a new item, listing both the item itself and the means by which it was acquired (purchased, looted, awarded, or stolen). It also tracks the rarity of each item, so we can see the rate at which players are acquiring different qualities of loot.

WHERE YOUR GAME MEETS GAMERS

VISIT US AT GDC BOOTH 416

GAME.minder

GAME.minder is designed to bring developers and gamers together. With GAME.minder, you can keep your audience updated so they **never miss a game release.**

GAME.minder is the perfect platform for you to quickly and easily inform fans about release dates and availability of all your games, making it easy to build a fan base and excitement around your next game release.

We welcome EVERY game – indie games and major releases. All platforms and systems covered!

To get your game into GAME.minder for free, contact us at:
newgame@minder-app.com
or visit handelabra.com/gmdev

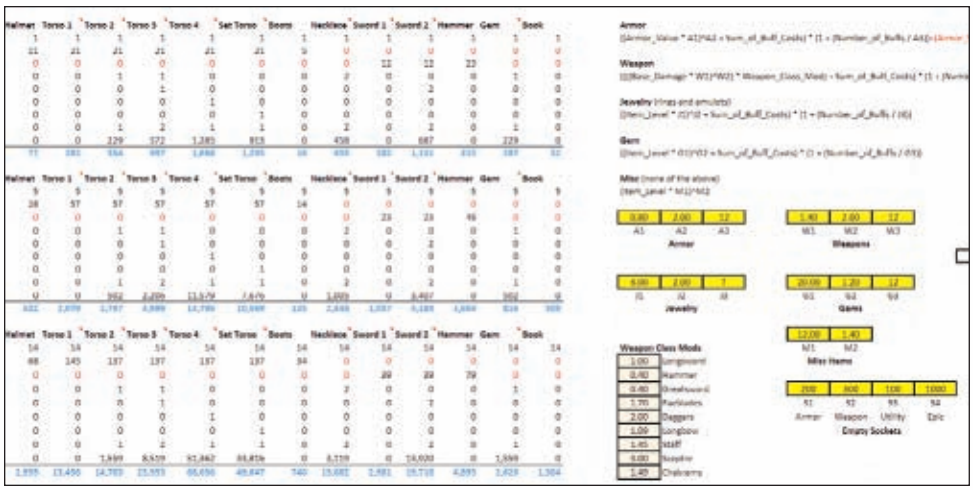
FOR iOS AND COMING SOON ON THE **MOBILE WEB**

Download GAME.minder FREE today

game.minder-app.com

Available on the App Store

	Warrior				Rogue				Mage				Enemy	Hits to Kill		
Level	ArmorPoints	Mitigation	MaxHealth	RealHealth	ArmorPoints	Mitigation	MaxHealth	RealHealth	ArmorPoints	Mitigation	MaxHealth	RealHealth	Damage	Warrior	Rogue	Mage
1	63	12.3%	50	57	47	9.5%	50	55	34	7.0%	50	54	5	11	11	10
2	90	16.7%	54	65	66	12.8%	54	62	46	9.3%	54	59	6	11	11	10
3	117	20.6%	58	73	85	15.8%	58	69	58	11.4%	58	65	7	12	11	10
4	143	24.2%	62	82	103	18.7%	62	77	70	13.5%	62	72	7	12	11	10
5	170	27.4%	67	92	122	21.5%	67	85	82	15.4%	67	79	8	12	11	10
6	197	30.4%	72	104	141	23.8%	72	95	94	17.3%	72	87	9	12	11	11
7	223	33.2%	78	116	159	26.1%	78	105	106	19.1%	78	96	10	13	12	11
8	250	35.7%	83	130	178	28.3%	83	117	118	20.8%	83	105	10	13	12	11
9	277	38.1%	90	145	197	30.4%	90	129	130	22.4%	90	116	11	13	12	11
10	303	40.3%	97	162	215	32.4%	97	143	142	24.0%	97	127	12	13	12	11
11	330	42.3%	104	180	234	34.2%	104	158	154	25.5%	104	140	14	14	12	11
12	357	44.2%	112	201	253	36.0%	112	175	166	26.9%	112	153	15	14	12	11
13	383	46.0%	120	223	273	37.6%	120	193	178	28.3%	120	166	16	14	12	11
14	410	47.7%	130	248	290	39.2%	130	213	190	29.7%	130	184	18	14	12	11
15	437	49.2%	139	275	309	40.7%	139	235	202	31.0%	139	202	20	15	13	11
16	463	50.7%	150	304	327	42.1%	150	259	214	32.2%	150	221	21	15	13	11
17	490	52.1%	161	337	346	43.5%	161	286	226	33.4%	161	242	23	15	13	11
18	517	53.4%	174	373	365	44.8%	174	314	238	34.6%	174	266	25	15	13	11
19	543	54.7%	187	413	383	46.0%	187	346	250	35.7%	187	291	28	15	13	11
20	570	55.9%	201	456	402	47.2%	201	381	262	36.8%	201	316	30	16	13	11
21	597	57.0%	216	503	421	48.3%	216	419	274	37.8%	216	348	33	16	13	11
22	623	58.1%	233	555	439	49.4%	233	460	286	38.9%	233	381	36	16	13	11
23	650	59.1%	251	612	458	50.4%	251	505	298	39.8%	251	416	39	16	13	11
24	677	60.1%	270	675	477	51.4%	270	555	310	40.8%	270	455	43	16	13	11
25	703	61.0%	290	743	495	52.4%	290	609	322	41.7%	290	498	47	16	14	11
26	730	61.9%	312	818	514	53.3%	312	669	334	42.6%	312	544	51	17	14	11
27	757	62.7%	336	900	533	54.2%	336	735	346	43.5%	336	594	56	17	14	11
28	783	63.5%	361	990	551	55.1%	361	804	358	44.3%	361	649	61	17	14	11
29	810	64.3%	389	1,089	570	55.9%	389	881	370	45.1%	389	708	66	17	14	11



Top: Armor scaling chart. Bottom: Gold economy scaling chart.

Merchant visited.

This shows whenever the player visits a merchant, what they bought and sold, and the exact amount of gold exchanged. Perhaps most importantly, it also reports the ratio of items the player can afford to items the player can't afford each time they visit a merchant. This helps us ensure that players always have a good mix of items within their price range, but also items to strive for as they move throughout the world.

STATES

/// Every time an event occurs, we store all currently active states. We can then filter events by individual players, by checking which events are active, or a combination of the two. The filtering ability is critical, because without it there is far too much data to parse in any meaningful way. Some of the most useful states we tracked are listed here.

Difficulty.

If your game has difficulty settings, you'll want to have a state that tracks it. If your data about Mage deaths comes from an easy playthrough and you're comparing it with warrior deaths from a hard playthrough, that's not a valid comparison. Make sure that you are comparing two similar challenges.

Player build.

You'll want states that cover the player's basic information, such as race and class. This is the most common state we used for filtering.

Player level.

This is another great one for filtering, as it lets you very easily break down data between the early game and later stages of gameplay.

Gear loadout.

We've got one state for each equipment slot on the player character. This enables us to compare the kill/death rates of players in, say, plate mail vs. those wearing robes, as well as comparing overall effectiveness of various weapons.

Skill ranks.

Whatever your system for skills/abilities/talents, you'll want a state for every possible rank the player can have in them. This will help you isolate and weed out particularly over/underpowered skills. Notice the death rate for players with a maxed out Heavy Armor skill is nearly zero? You may have gone a bit overboard with that one.

TOOLS

/// A huge component of balancing a game of this size and complexity comes down to the tools that you use. You need the means to easily view and edit vast swaths of data, a way to keep your data manageable, and a good way to understand the relationships between the different pieces of content. Fortunately for our design team, Justin Sargent and the rest of our amazing tools programmers have done an excellent job letting us do just that. Our tools were the key to providing us with the ability to build and balance the game systems and content for RECKONING, and featured a number of key components that allowed us to manage a huge amount of data and content.

MASS DATA VIEWING AND EDITING

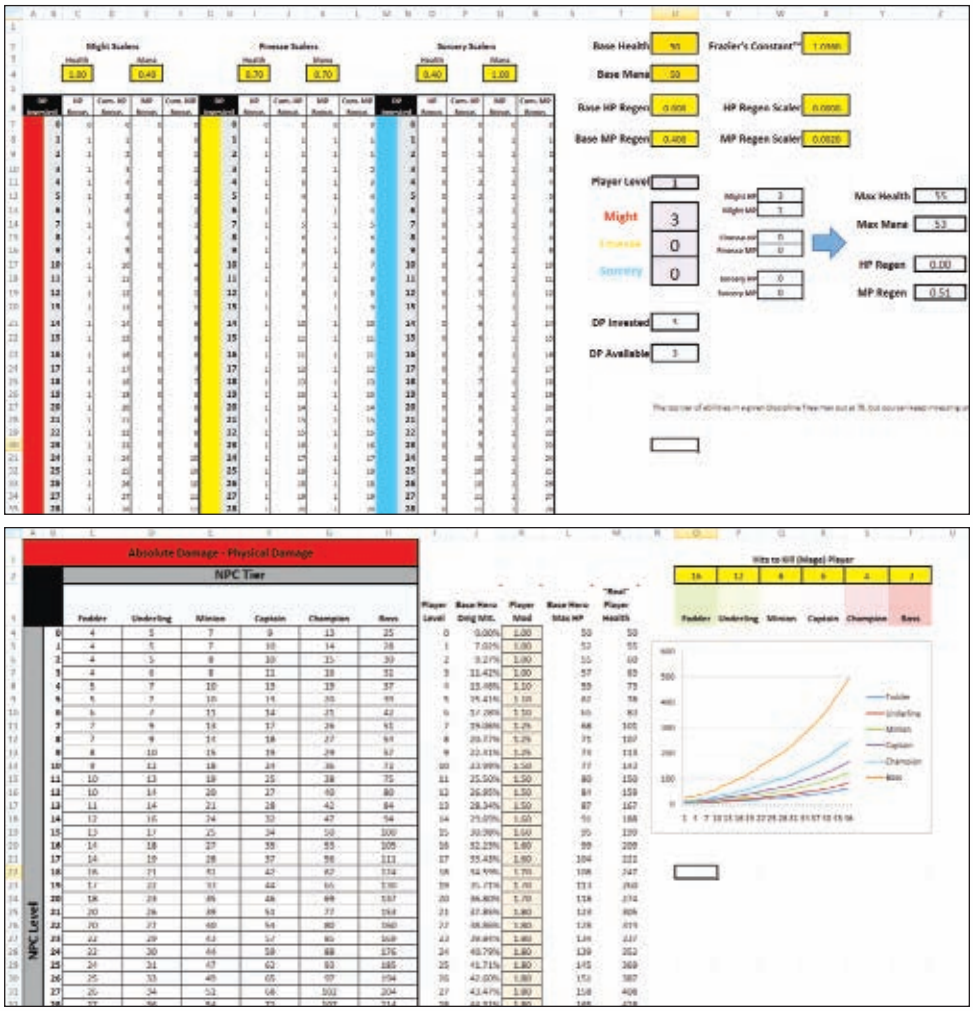
/// One of the trickiest parts of balancing a game is that you always need to visualize the relationships between variables. To some extent you can just keep these relationships in your head, but after the game reaches a certain size and level of complexity, that will no longer suffice. At this point you can resort to referencing docs and spreadsheets while working in your tools, or you can give the tools themselves the ability to let the user see the relationships between data. We've tried to do the latter as much as possible, and it's proven to be extremely valuable.

Using the standard .NET PropertyGrid, our design tool allows us to add any value from a given piece of content—whether that value be a number, a string, or even a texture—into the central viewing pane, where we can then see it in context with whatever other values we want. Data can then be directly edited—Excel style—within that central pane for rapid updates.

Additionally, the ability to make mass edits has proved incredibly

helpful. Not only can we do standard mass editing, such as selecting a bunch of assets and setting a field to the same value across all of them, we can also perform a specific operation to particular fields across a swath of assets. For example, we can select every dagger and increase the amount of piercing damage on all of them by -20%.

Finally, supporting data inheritance for many of our asset types has made our lives infinitely easier, allowing us to edit only the values that are unique to a given object rather than those shared with dozens or hundreds of other assets. For instance, all hammers share the same set of combat animations, so if that set of animations were to change, we need only edit a single asset (the "parent" hammer) and the change will propagate down to all the individual hammers.



Top: Health mana scaling chart. . Bottom: NPC stat scaling chart.

EXCEL WORKBOOK ASSETS

/// The crowning glory of the systems tools at Big Huge Games is our Excel workbook assets. These lightweight assets let us point to an actual Excel Workbook file, indicate the range of cells we want to use and which worksheets we want to include, then import the values from that workbook directly into the engine. At any given time, a single button press will re-import the values from that workbook. We use these assets for anything from armor equip requirements, to enemy hitpoints, to the matrix of items the player can potentially make with our crafting systems.

This method of quickly getting data from Excel directly into the game often allows us to avoid editing data in the tools at all, focusing our efforts instead on Excel, a long-established tool built specifically for viewing and managing mountains of data in an efficient manner.

In addition to the boon this provides to our workflow, it enables us to prevent a lot of potential oversights by linking together the various spreadsheets we use to balance the game. For instance, all the weapon damage in RECKONING is pulling from a spreadsheet that contains all standard weapon damage per level for each weapon class; meanwhile the amount of hitpoints each NPC has is driven from another spreadsheet which contains all NPC stats. When I increase the base damage of hammers in that weapon's spreadsheet, every hammer in the game will now be more effective—I'm making hammers more effective relative to the base weapon, the longsword.

In this instance, enemy stats won't change at all. If I change the longsword's damage, though, the health of every enemy in the game will now go up, maintaining the ratio of hits-to-kill that was previously working. Meanwhile if I want to leave weapon damage alone but universally make enemies easier or harder to kill, I can go into the enemy stats spreadsheet and just change that hits-to-kill number to adjust all enemy health up or down without impacting the relationship between the different weapon classes.

This network of interconnected spreadsheets (and the ability to easily import their values into the game) ultimately allows us to very quickly make minute balance changes or global ones, without needing to worry that we forgot to account for the downstream effects of any one particular change. Over the course of development on RECKONING, this has been our single most effective tool in the process of balancing the game.

BIG HUGE RESPECT

/// If you're setting out to create and balance a new RPG, especially one of the open-world variety, we at Big Huge Games salute you! You've got a very long road ahead filled with a terrifying array of challenges, but if you can pull it off, it's an incredibly rewarding experience. I hope that you can take a little something away something from this article that will make your life a little easier along the way.

IAN FRAZIER *is the lead designer on KINGDOMS OF AMALUR: RECKONING at Big Huge Games, but in his heart, he'll always be a systems designer. Good luck prying Excel from his cold, dead hands.*


STACKING
[stacking postmortem]

FOR THE FIRST 10 YEARS OF ITS LIFE, DOUBLE FINE PRODUCTIONS SHIPPED EXACTLY TWO GAMES: PSYCHONAUTS AND BRÜTAL LEGEND. ALTHOUGH WORKING ON SUCH ORIGINAL AND CRITICALLY ACCLAIMED TITLES WAS REWARDING, LONG DEVELOPMENT CYCLES CAN LEAD TO CREATIVE STAGNATION. TO RE-ENERGIZE THE STUDIO, DOUBLE FINE PRESIDENT TIM SCHAFER CAME UP WITH THE AMNESIA FORTNIGHT, A TWO-WEEK GAME JAM THAT SPLIT THE COMPANY INTO SEVERAL SMALL TEAMS, EACH GIVEN THE CHANCE TO CREATE THEIR OWN GAME IN THAT TIME.

The true value of the idea was put to the test in late 2009 when, shortly after the second Amnesia Fortnight had wrapped, a massive publisher bomb was dropped on the studio: the sequel to BRÜTAL LEGEND had been canceled. What was once just a creative exercise for the company became its best hope for survival, and the decision was made to develop four of the Amnesia Fortnight projects as commercial games. Within a few months, all four games were signed: COSTUME QUEST, STACKING, IRON BRIGADE (formerly known as TRENCHED), and ONCE UPON A MONSTER.

Now all we had to do was make them.

In STACKING, the player is transported to a world of living Russian nesting dolls. The game focuses on the adventures of the world's tiniest doll, a young chimney sweep named Charlie Blackmore. His diminutive size turns out to be his greatest strength, as he discovers his ability to stack into and control larger dolls. Charlie is then able to harness the "stacked" doll's special ability to solve adventure game-style puzzles on his quest to save his family from the black-hearted industrialist known as the Baron.

Making STACKING presented several new challenges, including adjusting to a major shift in the studio's production model and finding ways to maximize the efficiency of a very small team to finish the game on time. STACKING and the other three projects were also the first games that Double Fine would make without game industry luminary Tim Schafer directly leading the project. This was a decision by Tim to promote other creative voices in the company, but it also meant that we had to find a way to make sure that all four of these games continued and extended the Double Fine brand.

In other words, we needed to make an original high-fidelity game that also felt undeniably Double Fine in a very short time frame with a small team and a new leadership structure for new digital platforms.

What could possibly go wrong?

WHAT WENT RIGHT

1/ amnesia fortnight.

/// At the heart of Double Fine's new process for making games is the two-week rapid-prototyping period that we call the Amnesia Fortnight (because we forgot what we were working on for two weeks). Most games begin their life cycle with a concept document, concept art, and a long, labor-intensive "proof of concept" milestone that is followed by more documentation and planning. Amnesia Fortnight throws all that out and instead super charges a small team to create the heart of an entire game in a short amount of time. If all goes well, the team winds up with a strong initial direction for the mood, art style, and core mechanics.

Coming from a long background of multiyear projects, I thought these goals sounded a bit ambitious to say the least—but the process absolutely worked. Going into Amnesia Fortnight, I had created very little, just a short presentation, a few sketches, and an idea of how the core mechanics would work—and that was it.

But after two weeks, the team created a visually appealing, charming, playable demo that demonstrated many of the core mechanics like stacking, special abilities, and multiple-solution adventure game-style puzzles. The demo also emphasized both the diorama and silent film influences, both visually and in terms of narrative devices.

This was, by far, the best way to start a project that I have ever experienced. Not only did we create a playable, fun proof of concept quickly, but the whole team got excited about the game and their contributions to something new. By the time the demo was done, everybody was on board with the vision and felt they'd had a role in realizing it.

The other key benefit of this process is that we now had a demo that we could pitch to publishers to secure financing for the game; and because STACKING was an unusual game, it was even more important that we could clearly demonstrate to publishers what we wanted to make.

We were fortunate to sign STACKING rather quickly, and a lot of the credit goes to the strength of the demo that was created during Amnesia Fortnight.

2/ economy of design.

/// One of our goals in making smaller downloadable games was to create high-fidelity experiences with unusual gameplay, visuals, and personality that might be deemed too risky for large retail titles. A lot of those characteristics are expressed in both the details and the overall presentation, but with such short timelines and limited resources, how could we create a polished game with a distinct sensibility? The answer was to approach the game with economy in mind and look for clever ways to express that personality.

Many games start out with a "kitchen sink" approach—huge, overly ambitious, and with a giant feature and systems list. As the game is developed, the team focuses its efforts, inevitably trimming or cutting features to stay on track. Player verbs are reduced, and the game eventually focuses on a smaller, more polished feature set.

For STACKING, we started small from day one, expanding the idea over time instead of shrinking it. As one means to accomplish this goal, we limited our gameplay ideas to those that could be directly expressed with the simple core idea and action of stacking. The way the player solves puzzles and interacts with the world and other dolls were all directly derived from the player character's ability to stack inside other dolls. This approach kept us very focused and also ensured that the game had a thematic cohesiveness and distinct personality, which are key ingredients to the Double Fine brand. In addition, by focusing on fewer mechanics, we felt we would be able to both increase the overall accessibility of the game and the amount of polish for those mechanics in our short production cycle.

The best ideas evolve.

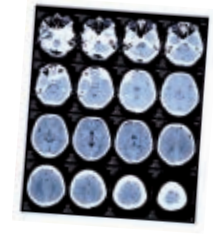
Great ideas don't just happen. They evolve. Your own development teams think and work fast.

Don't miss a breakthrough. Version *everything* with Perforce.

Software and firmware. Digital assets and games. Websites and documents. More than 5,500 organizations and 400,000 users trust Perforce for enterprise version management.

Try it now. Download the free 20-user, non-expiring Perforce Server from perforce.com/try20

Or request an evaluation license for any number of users.



PERFORCE
Version everything.

CONTINUED FROM PAGE 37

This approach really paid off when it came to the art production. The nature of the game meant that we could represent the 100+ unique denizens of this world with essentially one character model. Some early experimentation found that the personality, variation, and gameplay possibilities of the dolls was dramatically enhanced with the inclusion of simple accessories, such as top hats, bunny ears, or deranged clown wigs.

As we began to add layers of personality to our dolls, we considered adding some form of facial animation, but we eventually decided to try to convey our dolls' emotions purely through simple rigid body animation because of the potential production time involved. There was something immediately charming about this approach, and it had the added benefit of pushing us to emphasize our silent film theme even more. It also meant that our dialog recording budget could be put into the game's music, which was used to great effect by our genius audio team to express the game's personality in ways we couldn't have imagined.

Another benefit of having such economical characters is that their low bone and polygon count meant that we could put a lot of them on screen. That really opened up the possibilities for us to create a rich fantasy world for the player to explore. We found in early play tests how much players enjoyed just exploring the world; experimenting with dolls and their abilities. This feedback encouraged us to add a collection-and-reward system for player experimentation, giving meaning to exploration and ensuring that the game experience wasn't just moving from challenge to challenge.

The gameplay also directly benefitted from our economical approach to the characters. STACKING's challenges are essentially adventure game-style puzzles, but all feature multiple solutions. As the game was tested, players would often try to solve the puzzles in ways that they thought should work but didn't. This is a classic problem in adventure games, where players often follow a logical but incorrect (for the game's purposes) chain of thought in trying to solve puzzles.

In most adventure games, this can only be addressed by adding a few signs or some dialog to dissuade the player from trying these things. Although we employed these techniques to some extent, we were also able to easily add new solutions to existing challenges based on player feedback because the cost of adding a new doll to drive that solution was so low. The total cost of creating a new doll and solution, including all the modeling, texturing, animation, and ability programming was often only a few days of work.

3/ rapid prototyping and iterative development.

/// Since the start of BRÜTAL LEGEND, Double Fine has incorporated agile development methodologies into its production practices. Although different projects have used it in varying ways, at the heart of all of our practices is keeping as small a loop as possible when developing new features and ideas. Although we do write design documentation, create paper maps, and plan our productions, we focus a lot of our energy on creating "quick specs" for new features and working collaboratively across departments to get a rough version in the game as quickly as possible. For us, the flexibility to constantly evaluate what is working in a production, and then to shift direction if it isn't, represents the true power of agile development.

This approach was especially crucial on a small game like STACKING because we really had to maximize the efficiency of a small team. Although Amnesia Fortnight answered a lot of fundamental questions about the game, it was effectively the only "pre-production" period of the project. This meant that we still had a lot of unanswered questions going into production, which was especially evident when looking at the game's challenges.

These adventure game-style puzzles were the foundation of the game's structure. We knew early on that we wanted all our challenges to support multiple solutions, both to encourage replayability and to increase accessibility by trying to avoid some of the classic adventure game single-solution shortcomings. In our Amnesia Fortnight demo, challenges could be replayed to find additional solutions, but we were unsure of how to "reset" them. When should players be able to try to find additional solutions? A second playthrough? After they've experienced the primary narrative? On return visits to the location?

We decided that we wanted the player to be able to choose to immediately replay any challenge after finding a solution if they wanted. Although the player had to find only one solution to most challenges to progress, we wanted to encourage them to find more. The challenges would immediately "reset," fading the screen down, placing the player to be directly in front of the challenge, and activating a UI element that would show the player how many and which solutions they had found.

Although this solution sounds relatively straightforward, it took lots of rapid prototyping to create satisfying challenge solutions of varying difficulty that had no direct dependence on each other, since the player could choose to do them (or not) in any order. An initial write-up for a challenge would always include at least three solutions, intentionally leaving room to add or remove solutions based on playtest feedback.

We learned to rapidly implement quick versions of solutions and challenges, because seeing all the multiple solutions together was the only reliable way to identify things that didn't work. We stayed flexible through the whole process, cutting solutions that were boring or confusing and replacing them with ones that worked.

4/ established tech.

/// Many years ago, Double Fine embarked on the long, arduous task of creating its own technology from scratch for the game BRÜTAL LEGEND. BRÜTAL LEGEND was a large, ambitious game featuring an open-world single-player campaign, a multiplayer battle mode, scores of unique units that the player could "double team" and a ridiculous amount of over-the-top visual effects and custom sequences.

Starting STACKING with the BRÜTAL LEGEND engine not only meant that we had an extremely optimized engine with a diverse and rich feature set, but also a senior team who knew everything the engine was capable of and where it could be

pushed further. Combining that experience and engine horsepower with the more constrained scope of STACKING let us focus our energies on creating exciting visuals and gameplay, confident that we had more than enough technology to ship a competitive title on time.

5/ all for one and one for all.

/// Until recently, Double Fine had always been a one-team studio. People here are used to helping each other and working toward common goals. Although shifting to a multiple-project studio caused a fair amount of adjustment in our processes and interactions, everyone worked to continue to foster a cooperative atmosphere. Everyone felt, rightfully so, that the success of the studio depended on the success of all of the projects.

Throughout its history, Double Fine has held mandatory HOF ("hour of fun") meetings at the end of every milestone for its projects. At these times, everyone is required to play the latest build of the game and gather together after an hour to share their feedback on the game. We extended this practice in our multi-team structure by making sure that everyone, regardless of which project they were on, got an opportunity to play the other games. We also added online surveys for more detailed feedback and to track the comments over time. At one point, we also added beer to the mix, creating our first Hof-brau.

Not only are these practices good for keeping everyone aware of and involved in all of the studio's projects, it also makes it easier for people to help out on other projects if they have any downtime between project cycles. This helps us push the quality of our games in ways we can't budget for. This not only works because of the shared vision of the studio but also because we use a common toolset and engine. There is literally no ramp-up time for someone to jump on a project and help for a few days.

This gives an advantage that many multiple-team or smaller-project studios don't have, and STACKING definitely benefitted from extra art, animation, and programming help, right when it counted the most.

WHAT WENT WRONG

1/ stretched thin.

/// Transitioning from one big team to several small teams created a lot of growing (or would it be shrinking?) pains. The most immediate issue was that several teams were resource starved in certain roles. We couldn't really afford to make many new hires during this transition, so we had to come up with other solutions, all of which were far from perfect.

For some of the gaps, we "departmentalized" people with certain skill sets—such as UI, visual effects/technical art, and audio—in order to better share their high-demand skills among the teams. But what was the best way to share these people? They needed to be fully dedicated to a project at predictable times for those projects to hit their milestones, but at the same time, if any of the projects had no coverage, then these shared individuals would not be able to provide important input at critical stages. To make matters worse, many of our initial project schedules were bunched up on top of each other, making it hard to find natural times during the game's production cycle to move shared people between projects.

The worst part of the whole experience is that several talented people were stretched way too thin as they moved from project to project. Although everyone in the company worked very hard, these people unfortunately missed out on some of the creative flow that results when you are singularly focused on the task at hand.

However, these growing pains did help a few positives to develop. Because each of the new smaller teams needed more leads, it provided more growth opportunities for people who were ready to transition to a more senior or lead role. These games were shipped with several first-time leaders who wouldn't have had those opportunities if Double Fine were still one large team.

In addition, the studio began to embrace the idea of people developing secondary and tertiary skills to increase coverage across projects. On larger projects, people tend to take on more specialized roles, but for these smaller projects, people who are strong in multiple areas are very helpful in making the timelines and budgets work.

By the end of STACKING, the studio had improved its internal processes for sharing people. Departments were given more internal authority for controlling the flow of their shared work, taking advantage of efficiencies that only they could see. Different shared people were also assigned to be the primary contact for a particular project, so that, even if they were currently off the project, the other members knew who to contact for questions or feedback on a particular issue. Perhaps most importantly, we also learned as a studio how to plan better, and we take great pains to spread out our production schedules and needs on these shared departments as much as possible.

2/ our first date at the digital prom.

/// Shipping games on digital platforms was a new experience for Double Fine. Having already shipped a large game this generation with multiple DLC packs (BRÜTAL LEGEND), we thought we had a pretty good understanding as to what was involved—but we were a little off.

We underestimated the impact of things like digital platform-specific TCRs, and profile and marketplace integration. Dealing with the nuances of these took a lot of time, right at the end of our production, and stole away some of the systems polish work that we would've otherwise been able to do.

Even though we had just shipped COSTUME QUEST on the same digital platforms, we hadn't yet had the opportunity to create libraries of code, and most of our in-house expertise was tied up with other tasks. This made small technical issues really hard to track down.

In addition to dealing with platform-specific issues, the bug testing and patching process for DLC didn't go quite as expected. With a smaller production budget comes fewer testing resources, making tracking down all the bugs at the end of the project way less efficient than on larger projects. We had to use most of the artists, programmers, and designers to help do bug testing instead of polishing features.

By the end of STACKING, our in-house expertise in dealing with digital platforms had greatly increased, but the chaotic learning process caused a lot of extra stress and overworked some of our most senior programmers. The lack of resources for playtesting reminded us of how important QA is, and also reinforced that it pays to diligently address known bugs as early as possible.

3/ design tools.

/// Even though Double Fine has great technology and solid tools, some of the design tools can be slow to iterate with and hard to debug if something goes wrong. Our design tools

CONTINUED FROM PAGE 41

facilitate implementation of basic gameplay, such as repetitive tasks like spawning entities, applying game states, or adjusting the dialog response of a character in a particular situation.

Because Double Fine relies on gameplay programmers for lots of the harder gameplay implementation, the design side is still almost entirely based on Lua script. While this choice gives our games a lot of agility and flexibility in gameplay implementation, it can cause problems with some of the more basic gameplay tasks.

As an example, to place a simple AI character on a patrol path in STACKING required designers to edit several different text files. First, locators to define points in the patrol path were placed in our world editing tool and exported to the game. Then, a "task list" text file would be created by hand that would string together those points and tell the AI to do things like "idle," or "patrol," or "use ability." Lastly, another text file was edited to spawn that particular entity in the world and assign it the newly created task list.

While none of this work was difficult, if there was a small syntax error anywhere down the chain, such as a misplaced semicolon, things would not work. Most of the time the error messages that these problems generated were not verbose enough to be useful. This unfortunately meant that a gameplay programmer's valuable time would be wasted helping a non-programmer track down some of these problems.

In the future, we plan on defining our design workflow more specifically, developing better tools for repetitive design tasks, and investing in a Lua debugger. This will not only empower our designers and speed up their workflow but also allow gameplay programmers to stay more focused on their tasks.

4/ stacked-up scheduling.

/// Early in the production of STACKING, we made the decision to keep the entire team focused on the same level at the same time. We had little time to spend on organization, so we tried to line up one "phase" of a level's production with exactly one long sprint (about 3.5 weeks) that also equaled one milestone delivery to the publisher. The idea was that this would not only remove the need for advanced dependency planning, which would've been highly inaccurate anyway due to our lack of pre-production, but also assured that any level-specific needs and opportunities that emerged as the game developed could always be addressed.

While this approach did achieve some of those goals, it also resulted in some people being blocked while they waited for work that was upstream from theirs to be completed. When the entire team jumped onto a particular level all at the same time, some people would be immediately blocked. This was especially true when we kicked off a brand-new level. We dealt with this as best we could by frequently meeting about the blocking tasks and moving around the workload to try to unblock people. We also worked in passes, roughing in any dependent assets so that those downstream could start their work.

However, despite these efforts, people (usually gameplay programmers) were still unable to begin some of their work until later in a level's production than was optimal. We would've been able to capture more of the team's efficiency if we had done a more detailed dissection of our production process and staggered the times when people would start and end work on a level.

GAME DATA

PLATFORMS XBLA, PSN, PC
NUMBER OF DEVELOPERS 13 full time, and some polish help from others in the studio
LENGTH OF DEVELOPMENT 11 months
RELEASE DATE February 2011
TECH USED Double Fine Production's proprietary engine, originally developed for BRÜTAL LEGEND. Scaleform middleware for UI; FMOD middleware for audio.
DID YOU KNOW According to a database search, the word "taint" is used six times in the game.

5/ playtesting.

/// One of our goals with STACKING was to appeal to many different types of gamers, across the spectrum from casual to core. This is a notoriously difficult task, and if we were to succeed, we would need lots of playtesting to help guide our decisions. Unfortunately, we didn't really have a formal playtesting process at Double Fine. We would quite often invite friends and colleagues to play our games and give us feedback, but these people aren't a broad enough group to be our only playtesters.

Because most people working at a game company and their immediate friends are usually classified as core gamers, I asked for help bringing in lots of "casual" gamers—people who only occasionally played games. The production staff did a great job finding these types of gamers, bringing them in, and taking lots of helpful notes on their play sessions. This information provided us with lots of good ideas to help tune our game to suit the more casual player.

Unfortunately all this focus on the casual gamer unintentionally caused us to neglect some of our more core audience. We were so busy making sure that people didn't get lost, always had helpful solution hints, and weren't confused, that STACKING didn't quite have the mechanical depth that it could have. Some of our hardest challenge solutions should've emphasized things like combining doll abilities or more specific timing skill, but instead, they wound up just being more obscure. STACKING was sometimes critiqued as not having enough "skill challenge," and while that sort of dexterity-based gameplay wasn't what the game was striving for, it could've benefitted from offering the core side of our audience more gameplay depth to sink their teeth into.

If anything, this process has reminded us of how valuable playtesting is, and in the future, we'll strive to get a greater diversity of playtesters in front of our games.

STACKING CASH!?

/// Creating STACKING was one of the highlights of my career. Despite the major challenges involved, the team created an appealing game with a unique visual and gameplay style, which shipped on time to great reviews and very respectable sales. And more importantly, the flood of new games and creativity rejuvenated the studio and created opportunities for new creative voices.

Even though the game industry is rapidly changing, creating lots of uncertainty, it's an exciting time to be an independent developer. There is more diversity in games than ever before, a multitude of platforms to choose from, and a broader audience to make games for. In some ways, it feels as if there is endless possibility for those who can put together an idea and work hard enough to make it happen.

At the time of writing, all four of the Amnesia Fortnight projects have been completed in a short 16-month period. All have shipped on time, under four different project leads, with four different teams, with three different publishers, and on three different platforms. I think that's a testament to Double Fine's culture, creativity, and independent spirit.

LEE PETTY *was project lead on STACKING, and art director for BRUTAL LEGEND at Double Fine Productions. He has been a texture artist, 3D modeler, concept artist, lighting artist, effects artist, lead artist, and art director, since he entered the game industry in 1996.*

REVIEW BY CAREY CHICO

IDV INC.
SpeedTree 6.0

With the introduction of SpeedTree version 6, IDV Inc. offers an improved experience over the previous version of its tree-creation system. IDV is one of the only developers of software focused completely on tree creation and tree rendering for both real-time and rendered applications. In fact, apart from Woody3D, which was announced last year, there is no other rendering system that offers anything similar.

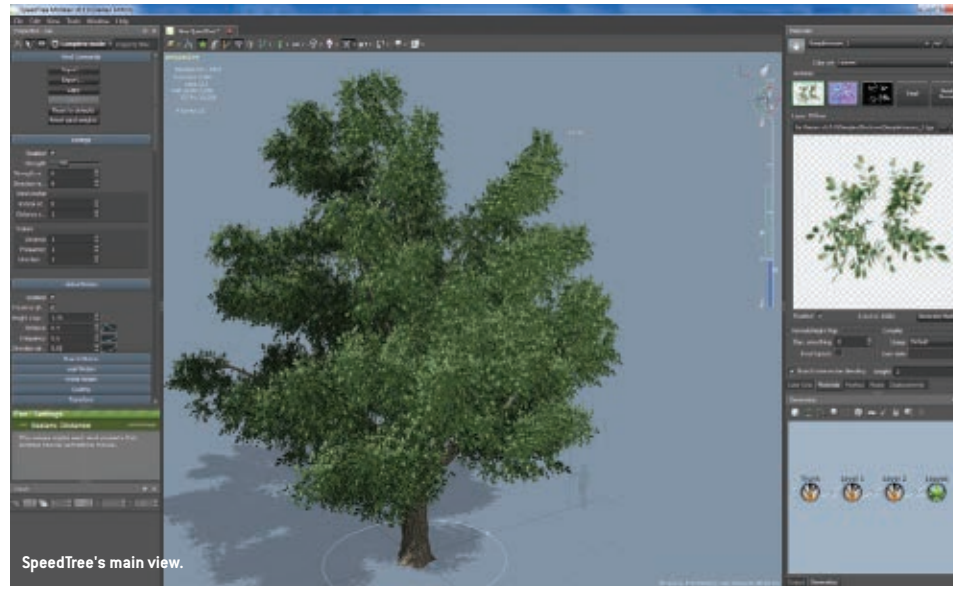
In my experience, trees have always been one of those asset classes with larger-than-normal iteration times. Not only are they resource hogs for your tech—since you want them to sway with the breeze, burn down when shot, and so forth—they can also be tedious work for your art team. Since trees provide familiar visuals for players and add volume to worlds, it's crucial that you find ways to make them quickly, populate your game efficiently, and give them dynamics to make your game come alive.

VERSION 6

» While version 5 introduced the revamped interface for the modeler that made making trees much easier and more intuitive, this new version does offer up a few new tricks. My last version was 4.2, so to me, version 6 represents a whole new ball game.

Once I got through the Simple Mode tutorials to get accustomed to the UI, I was able to very quickly generate a tree trunk, branches, and leaves through the node-based UI. I have to say that I really liked this generator's interface: it's clear, logical, and very polished. All the elements have a nice drop shadow, the icons are attractive, and the mouse-driven usability is quite fluid. It contextualizes the tree in small bite-sized pieces, which makes navigating through what you wish to do much easier.

The same can't be said for the left tabbed side of the UI. It reminds me of the frustration I had when I first used 3ds Max and discovered that much of the UI was hidden, and that I had to grab the UI and drag it upward just to find it. It's just a bit tedious, and you'll find that one of your primary actions when using version 5 or 6 is opening and shutting the tabbed sections of the UI to get to the one you want to see.



SpeedTree's main view.

BRANCH INTERSECTION BLENDING

» In this version, SpeedTree gains a new key feature called Branch Intersection Blending. This helps to make seamless transitions between branches and tree trunks automatic and easy.

In the previous version, the software would take the open edges of each branch and weld them to the trunk. This helped make the geometric transition from trunk to branch more natural and more pleasant than the old "shove into tree" method, but this didn't do anything to hide the texture seams at the junction point. That's what branch blending does.

Now, when you turn this function on inside the materials tab of the UI, the trunk texture borrows the coordinates from the branch and maps the UVs to match the trunk. This texture is then faded off so that the original branch UVs take over, and then presto: you have seamless texture transitions from the trunk to the branches.

In the end, what you get is a natural and lovely blend between the branches and the trunk. As I mentioned, I've used 4.2 before, and the lack of a branch solution did affect visual quality. Provided you support this SpeedTree system in your engine, this feature will definitely improve your game's visual fidelity.

The other good news is that branch blending is supported by the FBX exporter, so you can get this blending goodness out to your other modeling programs for cinematics and other uses. The data is saved out in texture layers 3, 4, and 5 in the exported format.

OTHER NEW FEATURES

» While branch intersection blending is one of the larger features present in version 6, IDV was still able to add a host of other tweaks to improve the user experience.

- Automatic online updates through built-in software updater.
- New optional Maya- or 3ds Max-style navigation modes.
- FBX support gets additional export features in this version. Apart from being able to export the branch blending, you can also export the camera-facing leaf cards and meshes via

FBX, as well as extract and invert alpha maps.
- You can now add templates to the scene via the right-click context menu.

All these tweaks help to improve usability.

WORTH IT?

» So the question for many might be, is it worth investing in SpeedTree now? My answer would be yes. There are so many other higher-priority assets that need attention in a game. The amount of work that would have to be done just to duplicate what the SpeedTree system does is not worth the effort—they are only trees, after all. Additionally, of all the third-party systems that could be adopted into your game, SpeedTree's costs are quite economical in comparison. The bang for the buck here seems clear.

SpeedTree 6 picks up where version 5 left off, and still offers the best UI for this system to date.


Branch intersection blending.

If hindsight were 20/20, I would have said hold off on 4.2 until 5 and 6 came along—it's that large of an improvement in the overall system.

Lastly, the Branch Intersection Blending feature in this latest release really does make the end result shine. The fidelity and natural-looking quality it adds to trees is obvious.

WHAT'S NOT TO LIKE?

» The tabbed UI on the left is a bit tedious to operate. It's not a deal killer, but it is something for the company to improve upon. Anything else I might add would just be mincing words. The darn thing makes trees after all!

MAKING TREES

» All in all, SpeedTree continues to do what SpeedTree does: turn the construction of trees into one of the easiest tasks your team has to manage. The system alleviates the iteration burden that these types of assets can have on your schedule, which allows your team to focus on the core assets that are front and center for your game. But let's face it, when we are able to make trees look as good as these do, we may be spending more time wandering the forests than we will fighting the bad guys.

CAREY CHICO *is president/chief creative officer of Globex Studios and a 17 year veteran of the game industry. Prior to Globex, Mr. Chico was a founding member of EA/Pandemic Studios where he oversaw art direction, production, outsourcing and tech R&D for multiple projects as executive art director. He is credited on 16 game titles including* Star Wars: The Clone Wars *and* Battlefront *series,* Mercenaries, *and* Full Spectrum Warrior.

IDV INC.
SpeedTree 6.0

5446 Sunset Blvd., Suite 201
Lexington, SC 29072
http://www.speedtree.com

PRICE

› Contact IDV for quote

SYSTEM REQUIREMENTS

› 2GB Ram, 235MB HD space, Shader model 2.0 graphics card or better
› The SpeedTree SDK requirements depend on the type of integration and platform. Contact IDV for specific information.

PROS

1. Look, it makes trees!
2. Branch Intersection Blending offers up a much more natural and seamless looking tree.
3. Additional FBX export support increases the value and usability of the software.

CONS

1. Left tabbed side of the modeler is a bit tedious to work with.

WE'RE READY FOR YOU TO MAKE SOMETHING EPIC

Apply Today

www.epicgames.com/careers

CHAIR
UTAH

EPIC GAMES
NORTH CAROLINA
SEOUL • TOKYO

PEOPLE CAN FLY
WARSAW

WHAT'S MINE IS YOURS

THE COMPLEXITIES OF REVENUE SHARING

Some lessons are harder to learn than others. One of the toughest lessons you may ever learn is that granting someone a generous share of the revenue from your game in exchange for a service (assistance with development, publishing, etc.) does not mean you can assume your incentives are properly aligned.

Say that you give a publisher 50% of the revenue from your game in order to promote it, to handle customer service, or for some other reason. Or perhaps you've agreed to develop a game in tandem with a few other individuals and split the future revenue equally. In either case, you're making the important assumption that a significant percentage of future profits will ensure that all parties will do their "best" to make the game a success. Sometimes, that's exactly what happens—but not always, unfortunately.

REVENUE SHARES AND PUBLISHERS

» There are many situations in which even a large share of your game's revenue may not result in the behavior you need or expect from your business partners. A publisher, for example, may view your self-funded game as just one of a great many small gambles in their portfolio: something worth putting a few hours of effort into and/or maybe a few thousand dollars, but certainly no more than that until the game "proves" itself. The fact that you've given them (for example) 50% of your revenue after paying for development yourself may mean very little to certain publishers because they view your game as a lottery ticket ... and you don't win a lottery by spending large sums on a single ticket.

If giving a publisher 50% of your revenue isn't enough to get them to really get behind it and help you in significant ways, then what else can you do? The answer, in some cases, may simply be "nothing." If you aren't effective at pitching your game and your studio, you may discover that the only publishers who are interested in getting a piece of the game are those who want that piece for free. If, however, you have a competent pitch and seem like a developer worth building a long-term relationship with, there are certain demands that you can and should make of any publisher.

For example, you may demand a recoupable advance against your future royalties. If the publisher believes strongly in your game, this theoretically costs the publisher very little (just the interest payments they would have received from that money during the time period in which they advanced it). It also gives the publisher a good reason to get behind the game: they want to recoup that advance at bare minimum! An alternative is to ask for commitments, such as 300,000 downloads. If the publisher can't get your game downloaded at least that many times, then their revenue share should be reduced, or in the case of an extreme shortfall, the publishing contract could be terminated.

REVENUE SHARES AND DEVELOPMENT PARTNERS

» Things get significantly more complicated when sharing revenue with individuals or companies with whom you have partnered to co–develop a game. Such arrangements are a big leap of faith for everyone involved, and you absolutely cannot assume that healthy revenue shares will keep everyone on the same page. Here are just some of the reasons why a co–development partner might disappoint you (or vice versa!) despite the fact that you're splitting revenue:

- Different financial needs. Someone on the team may not actually care much about money. Maybe they are independently wealthy. Maybe they simply aren't motivated by money no matter how little or how much they have. In either case, a revenue share is no guarantee that a person will share your goals.
- Different financial goals. Even if two people on a team have exactly the same level of financial need, one might be satisfied with a ten thousand dollar payoff while another might be dissatisfied with anything under a million. There's a good chance the latter person is going to become frustrated with the former if the project is marginally but not largely successful right off the bat (and of course, most projects aren't that successful at all).
- Different priorities. Even if two parties have the exact same financial needs and goals, there are other priorities to consider. How important is the success of this game to each party involved? If it's a make-or-break project for one party but something that could easily fail without consequence for another party, there may be problems down the road. If one party needs income from the project in three months to survive while another can plug away for years without income, conflicts could easily result, especially if this isn't discussed before the project is kicked off. If one party owns the IP the game is based on while another does not, once again, there may be a vast difference in motivation to perform.

The aforementioned examples are just a tiny slice of all the possible differences between people (and companies) that can result in serious disputes down the line, especially once real money is involved. And unfortunately, there's no perfect way to predict and prepare for all possible disputes. To some extent, when you're splitting ownership of a game and/or its revenue, you're always taking a big gamble. The best way to reduce the risk of all parties involved is to carefully talk through your goals, priorities, and commitments before kicking off a partnership, and document in writing the results of those conversations.

TALKING IT THROUGH

» Would you be disappointed if your partner didn't work at least 20 hours per week on the game, on average? Discuss it, and put something in the contract that specifies exactly what happens if someone doesn't meet their commitments (i.e., their revenue share drops from X% to Y% after a given period of time). What if someone gets sick and can't work for a month? Agree to something and put it in the contract, too. What happens if the game launches and is not successful? How long are you all willing to keep working on it? What happens if someone bails on the project before this time period has elapsed? Talk it all through, and put it all in a contract.

Unfortunately, doing this will not guarantee that you avoid disappointment or drama. If you partner with the wrong folks (or even with the "right" folks but under the wrong conditions), no contract is going to help you—but going through this process is vital. Most importantly, it may help you avoid getting into the wrong partnership. Additionally, it will give you a framework to rely on in the event that disagreements arise between you and your partners.

DAVID EDERY *is the CEO of Spry Fox and has worked on games such as Realm of the Mad God, Steambirds, and Triple Town. Prior to founding Spry Fox, David was the worldwide games portfolio manager for Xbox LIVE Arcade.*

DevTest Studio

The industry's #1 choice for test management and defect tracking

DevTrack

Use DevTrack to track defects/issues

- Track each issue through a definable workflow
- SCM integration-track fixes against their source code deliverables
- Deploy a resolution across multiple releases, versions and products
- Reporting and metrics to illustrate the entire defect lifecycle

DevTest

Use DevTest to manage your testing

- Create a central repository for your test cases, Knowledge items and automation scripts
- Schedule releases and test cycles using a wizard-driven interface
- Execute test assignments and submit defects from the same interface
- Track results with real-time dashboards and reports

TestLink

Use TestLink to automate your testing

- Add automated tests to the DevTest library
- Schedule automated tests along with manual tests
- Launch automated tests from the DevTest interface
- Track automation results with real-time dashboards and reports

Try DevTrack and DevTest live. Watch a recorded overview. Request an online demo.

www.techexcel.com
1.800.439.7782

CHASING SHADOWS

UPPING YOUR SHADOW MAP PERFORMANCE

Shadow mapping is a well-known technique for rendering shadows in 3D scenes. With the rapid development of graphics hardware and the increasing geometric complexity of modern game scenes, shadow maps have become an essential tool for real-time rendering.

Though they're popular, shadow maps are notoriously hard to get working robustly. One of the biggest issues with shadow mapping is poor scalability, both in terms of quality and performance, with respect to the increasing complexity of real world game scenes. In theory, we should need only one shadow map sample per pixel and the shadow map generation step should be output sensitive, i.e., the running time of the shadow map generation step should depend only on the number of objects that cast visible shadows, instead of the number of objects in the scene. In practice, however, everyone has their own bag of tricks to cope with shadow map resolution management and shadow map rendering issues.

There exists an ever-growing body of literature on different shadow mapping techniques, most of which focus solely on improving shadow quality (see Eisemann in References). Indeed, given a sufficiently high shadow map resolution, these methods are able to achieve shadows of very high quality. Hence, in this article our focus is on scalable shadow map performance.

We already have a good understanding of how to render large environments in a scalable fashion, at least without shadows. Occlusion culling helps us to get rid of all hidden objects, and we can apply level-of-detail techniques to the remaining visible ones in order to bound the geometric and shading complexity to an acceptable level. Thus, it is not uncommon to find out that the performance bottleneck shifts to shadow mapping (see Silvennoinen in References).

Shadow map performance can be characterized by two things: generation cost, and sampling and filtering cost. Sampling and filtering are essentially independent of the geometric complexity of the scene since they are texture space operators. On the other hand, shadow map generation, i.e., the rendering of shadow casters to the shadow map, can consume a big portion of the rendering budget, unless we take care to bound the number of rendered shadow casters somehow. This effect is potentially amplified when using cascaded shadow maps because some shadow casters could be rendered multiple times during the shadow map generation phase.

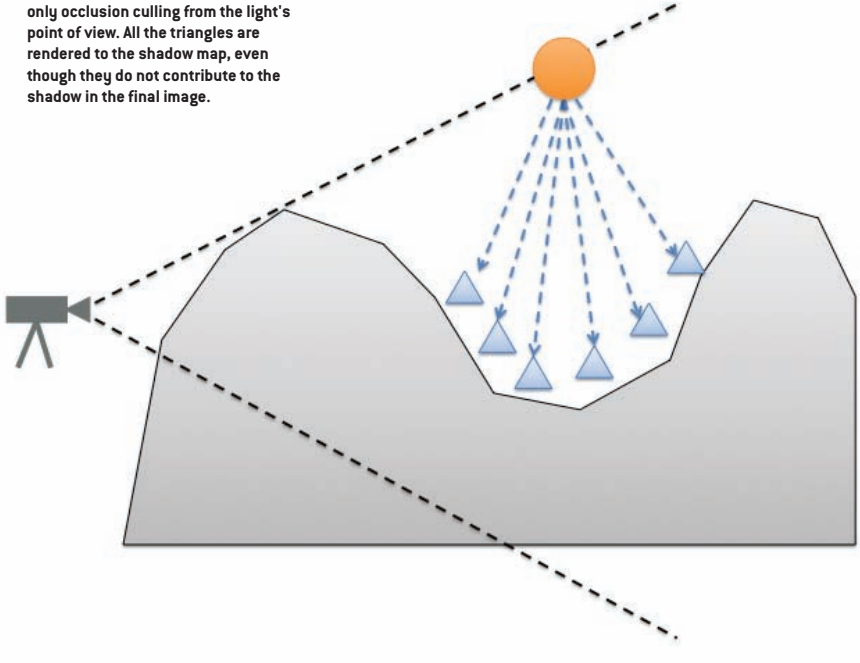


FIGURE 1: The problem with using only occlusion culling from the light's point of view. All the triangles are rendered to the shadow map, even though they do not contribute to the shadow in the final image.

BACKGROUND

» In this article, our goal is to speed up shadow map generation. A naive solution to this problem would again use occlusion from the light's point of view to reduce the number of rendered shadow casters when rendering the shadow map. However, as Bittner et al observed (see Bittner in References), the effectiveness of this approach is completely dependent on the light view depth complexity.

Large outdoor environments combined with a global shadow casting light source such as the sun or the moon might not gain much from occlusion culling alone. In the worst case, we might end up rendering all the potential shadow casters contained in the intersection of the view frustum and the light frustum, even though most of the shadow casters do not actually contribute to the shadows in the final rendered image (see Figure 1).

Bittner et al observed that in addition to using occlusion culling from the light's point of view, it is essential to cull shadow casters, which do not cast a visible shadow. The group first rendered the scene from the camera's point of view and used occlusion culling to identify all the visible shadow receivers. Second, they rendered the visible shadow receivers to a light space shadow receiver mask to mark the visible parts of the scene as seen from the light's point of view. Finally, they applied occlusion culling from the light's point of view—together with the shadow receiver mask—to identify the potential shadow casters, which should be rendered to the shadow map (see Figure 2).

Compared to the naive approach, the main overhead in this method comes from the shadow receiver mask generation, which is not guaranteed to be optimal in all cases. In the worst case all visible shadow receivers are already in shadow, and the receivers are rendered to the shadow receiver mask, since the visibility status from the light's point of view is determined after the shadow receiver mask is created. In addition, they rely on an efficient hardware occlusion culling algorithm which limits the applicability of their method.

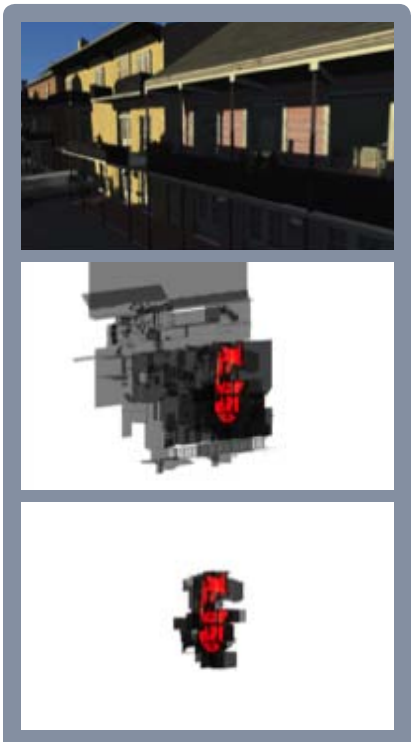


FIGURE 2: (top) A view of a game scene rendered with shadows. (middle) A shadow map with shadow casters rendered using a naive application of occlusion culling from the light's point of view (gray) and visible pixels (red). (bottom) A shadow map with shadow casters rendered using our method (gray) and visible pixels (red). The scene is part of LEFT4DEAD 2 (courtesy of Valve Corp.).

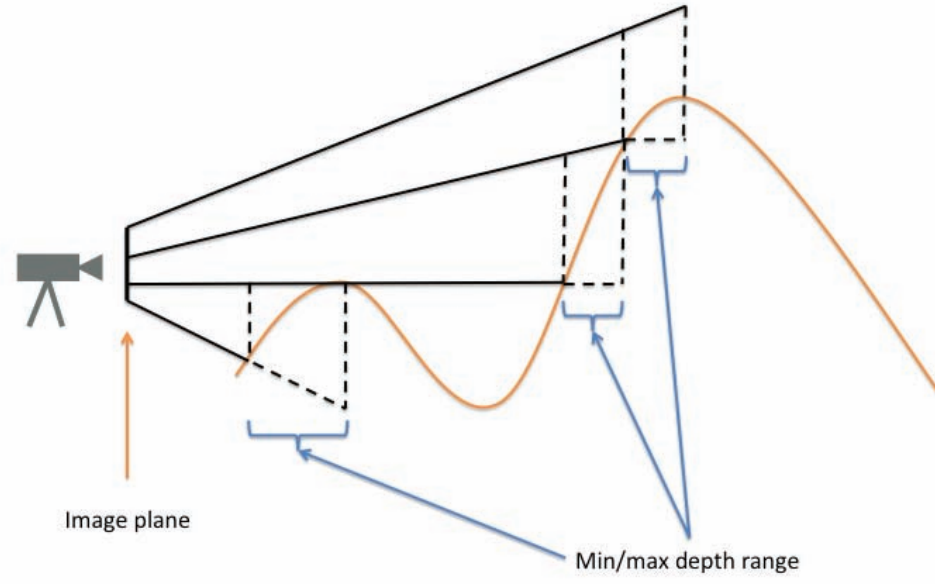


FIGURE 3: An illustration of a min/max depth pyramid. We compute the minimum and maximum depth for each screen space tile and construct a bounding frustum based on the depth range.

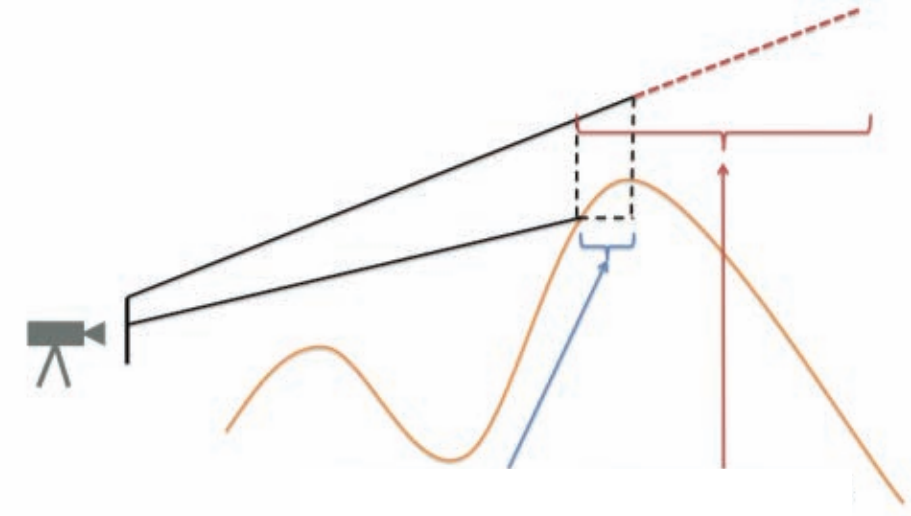


FIGURE 4. An illustration of the horizon problem. (right) Using only depth values that correspond to geometry, we avoid extruding the frustum. (left) Otherwise the mask will be overly conservative.

In this article we introduce a practical variant of the shadow caster culling algorithm based on the ideas of Bittner et al with the aim of making the method a more viable option in a wider array of scenarios. In particular, our method only assumes the availability of the main view depth buffer, and we will demonstrate a technique for generating the shadow mask directly from the depth buffer. Furthermore, the shadow mask generation step is independent of the geometric complexity of the scene and solves the worst case scenario with the method described above.

SHADOW MASKING

» The first thing to do is identify all light space shadow map texels that will contribute to the shadow in the final image. A shadow map texel T will contribute to the final image if T gets sampled during the deferred lighting pass. We call the set of contributing shadow map texels a shadow mask. There is a direct connection between the main view depth buffer and the shadow mask since each visible pixel in the main view will generate shadow map lookups during the deferred lighting pass, and we will use this property to generate the shadow mask directly from the depth buffer.

Given a fully initialized shadow mask, the culling part is relatively simple; for each shadow caster we rasterize the light space AABB of the shadow caster, and if the rasterized bounding box does not contain a contributing shadow map texel, we can safely cull the shadow caster. Otherwise, we have a potential shadow caster, which should be rendered to the shadow map.

A straightforward method for generating the shadow mask from the main view depth buffer would be to reproject the depth buffer pixels to light space, by treating each depth buffer pixel as a world space point and rasterizing this point cloud to the shadow mask. However, this approach has two obvious drawbacks. First of all, the number of points generated from a full resolution depth buffer creates a non-trivial amount of work to be executed on the GPU. Second, we lose all information about the topology of the original geometry by using the point cloud approximation, which means that all bets are off when considering the connectivity of the projected point cloud. In particular, small holes or cracks are likely to appear in the shadow mask, which could lead to false occlusion and missing shadows. Point splatting could fix the light space topology, but it adds another burden to our already-overworked GPU.

We propose a scalable method for shadow mask generation by subdividing the depth buffer

TECHNICAL SCHOOL

"Coming in from VFS, I was ready to hit the ground running. VFS prepared me very well for the volume and type of work that I do, and to produce the kind of gameplay that I can be proud of."

David Bowring, Game Design Graduate
Gameplay Designer, SAINTS ROW 2

VFS

VANCOUVER FILM SCHOOL
200-198 West Hastings St
Vancouver, BC V6B 1H2
Canada
Phone: 604.685.5808 or 800.661.4104
inquiries@vfs.com
www.vfs.com/gamecareer

Vancouver Film School

Game Design at Vancouver Film School is an intense one-year program that covers everything you need to join the game industry as a designer or producer, from theory to hands-on practice to the production of a professional-quality portfolio. There's a reason why the L.A. Times called VFS one of the top 10 schools favored by video game industry recruiters.

VFS Game Design students learn more than just one side of game design – they experience the full scope of this varied and rewarding career through an in-depth curriculum that includes:

- >> Interactive Narrative
- >> Analog Games
- >> Interface Design
- >> Scripting
- >> Level Design
- >> Pre-Production
- >> Project Management
- >> Flash
- >> Mobile & Handheld Design
- >> Game Audio
- >> The Business of Games

Led By Industry

In VFS Game Design, you're mentored by a faculty of respected industry pros – your first crucial connections to the professional world. At the helm is veteran Dave Warfield, who, as a Senior Producer for EA, helped produce and design the NHL franchise for 10 years. His many other credits include titles like EA's NBA LIVE and Konami's TEENAGE MUTANT NINJA TURTLES. An Advisory Board of industry leaders, including luminaries from Activision, Microsoft, Nokia, and LucasArts, keeps the curriculum on the cutting edge.

A Studio Environment

In a process that closely mirrors a real-world studio environment and production pipeline, you work in teams to take games from concept to completion. Toward the end of your year at VFS, you get the chance to present your final playable games to an audience of industry representatives and recruiters: a unique chance to prove yourself and make valuable professional contacts.

Living & Creating in Vancouver

In VFS Game Design, you have the advantage of learning in Vancouver, B.C., Canada. Along with its strong film, TV, and animation industries, Vancouver is a world center of game development, meaning that VFS is always industry-current, hosts many guest speakers, and provides you with vital mentorship and feedback opportunities throughout your year. It's the perfect place to get your career started.

The Results

Our graduates have gone on to earn key design and production roles at top studios around the world. A small selection of their recent and upcoming titles includes: PROTOTYPE 2, MASS EFFECT 3, DEUS EX: HUMAN REVOLUTION, WARHAMMER 40,000: DAWN OF WAR II, DRAGON AGE II, PUNCH-OUT!!, FIFA 10, SKATE 3, TRON: EVOLUTION, DEAD SPACE 2, STAR WARS: THE OLD REPUBLIC, DEAD RISING 2, and MODNATION RACERS.

Find out about VFS Game Design and begin your career at **vfs.com/gamecareer**.

into a screen space tile grid. Given the screen space tile grid, we compute a world space bounding frustum for each tile based on the minimum and maximum depth values in the tile. Note that after this process each world space tile frustum contains all the world space points corresponding to the screen space depth buffer pixels that reside inside the screen space tile. Then, instead of rasterizing the point cloud resulting from the full resolution depth buffer we only need to render the frusta to the light space shadow mask buffer to obtain a conservative approximation of the shadow mask (see Figure 3).

It turns out that min/max depth pyramids offer an efficient way to compute the world space tile frusta. Given the full resolution depth buffer, we compute a min/max depth pyramid by successively downsampling the original depth buffer until we obtain the grid resolution we want. We store the depth values in the R and G channels of a single texture.

In order to guarantee that we do not lose any information in the original depth buffer—and to make sure the tile grid matches the last mip level in the chain—we round up the depth buffer resolution to the next multiple of 2^N in each dimension for the lowest mip level (i.e., highest resolution) in the min/max pyramid, where N is the number of mip levels in the min/max pyramid, and N-1 is last mip level in the mip chain with the same resolution as the tile grid. Then, we bootstrap the min/max pyramid downsampling by upsampling the depth buffer to the lowest mip level of the min/max pyramid.

Another subtle consideration to make while performing the min/max pyramid construction is how to correctly downsample the maximum depth values. Suppose we have an outdoor scene consisting of terrain and a visible skybox at the infinity with depth 1.0. Now, if we would simply take the maximum depth value of the four samples in the lower mip level, it would mean that there would almost surely be a set of bounding frusta spanning the whole terrain at the horizon (see Figure 4). The correct way to handle this case is to consider only maximum depth values less than one during the downsample operation (see Listing 1).

After we have obtained the min/max depth pyramid that corresponds to the screen space tile grid of the original depth buffer, the next step is to rasterize the frusta defined by the grid to the light space shadow mask. In our implementation, we chose to utilize the geometry shader stage and stream the emitted triangles from the geometry shader directly to the rasterizer stage, eliminating the need to explicitly compute the bounding geometry or read back the results to the CPU.

To feed the geometry shader we render an immutable point list, where each point corresponds to a single tile in the tile grid. During each geometry shader invocation we emit the frustum triangles by looking up the minimum and maximum depth values from the last mip level of the previously constructed min/max depth pyramid. In addition, we disable depth and color writes, and write out only to the stencil buffer associated with the shadow map, setting the stencil value to one for each passed fragment. The final shadow mask will then consist of all the shadow map texels with an associated stencil value of one (see Figure 5).

In addition to obtaining the binary shadow mask, we could also prime a conservative depth buffer for subsequent occlusion culling passes by rasterizing the backfacing triangles of the view frustum. This might be especially useful in cases where the light depth complexity is high.

LISTING 1:

Pixel shader code for the min/max pyramid downsampling pass.

```
float4 MinMaxDownsample_PixelShader(in float4 Position : SV_
Position, in float2 UV : TEXCOORD) : SV_Target

{

     float4 Samples[4] = {

          Texture.SampleLevel(PointSampler, UV, 0),

          Texture.SampleLevel(PointSampler, UV, 0, int2(1,0)),

          Texture.SampleLevel(PointSampler, UV, 0, int2(1,1)),

          Texture.SampleLevel(PointSampler, UV, 0, int2(0,1))

     };

     // Use only valid depth values in the downsampling filter

     for (int i = 0; i < 4; i++)

          Samples[i].y = Samples[i].y < 1 ? Samples[i].y : 0;

     float MinZ = min(min(Samples[0].x, Samples[1].x),
min(Samples[2].x, Samples[3].x));

     float MaxZ = max(max(Samples[0].y, Samples[1].y),
max(Samples[2].y, Samples[3].y));

     return float4(MinZ, MaxZ,0,0);

}
```

SHADOW CASTER CULLING

» Now that we have obtained a fully initialized shadow mask constructed from the original depth buffer, we can now run an occlusion culling pass from the light's point of view using the shadow map as our depth render target, in a spirit similar to Bittner et al. Depending on the circumstances, however, we might not need a fully hierarchical occlusion culling pass to obtain significant performance gains at the shadow map generation step. In some cases, especially when dealing with low depth complexity from the light's point of view, it is more beneficial to avoid all forms of GPU read-backs in order to eliminate any potential synchronization issues, such as GPU starvation, CPU stalling, or latency, which are usually associated with hardware occlusion queries.

In our implementation, we keep all the data on the GPU at all times, and choose to use predicated rendering for shadow caster culling. In particular, we issue predicate queries for each shadow caster candidate in the intersection of the light frustum and view frustum by rendering the shadow caster candidate's world space AABB and checking for intersection with the shadow mask. We disable both depth and stencil writes during this pass and set the stencil test for equality with one (i.e., the shadow mask stencil reference value). Then, in a second pass we issue predicated draw calls for each candidate and profit each time a predicate culls a shadow caster candidate.

references

Real-Time Shadows - Elmar Eisemann, Michael Schwarz, Ulf Assarsson and Michael Wimmer, CRC Press 2011

Occlusion Culling in Alan Wake - Ari Silvennoinen, Hiding Complexity, SIGGRAPH 2011 Talks

Shadow Caster Culling for Efficient Shadow Mapping - Jiri Bittner, Oliver Mattausch, Ari Silvennoinen and Michael Wimmer, Symposium on Interactive 3D Graphics and Games 2011

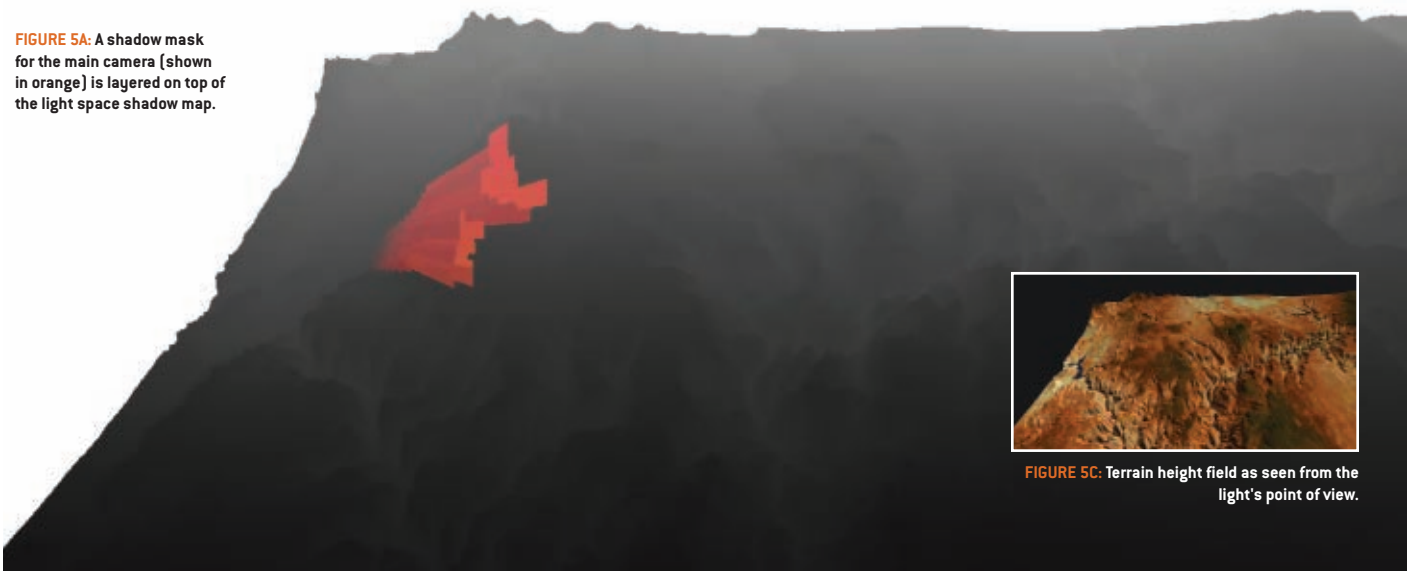


FIGURE 5A: A shadow mask for the main camera (shown in orange) is layered on top of the light space shadow map.

FIGURE 5C: Terrain height field as seen from the light's point of view.

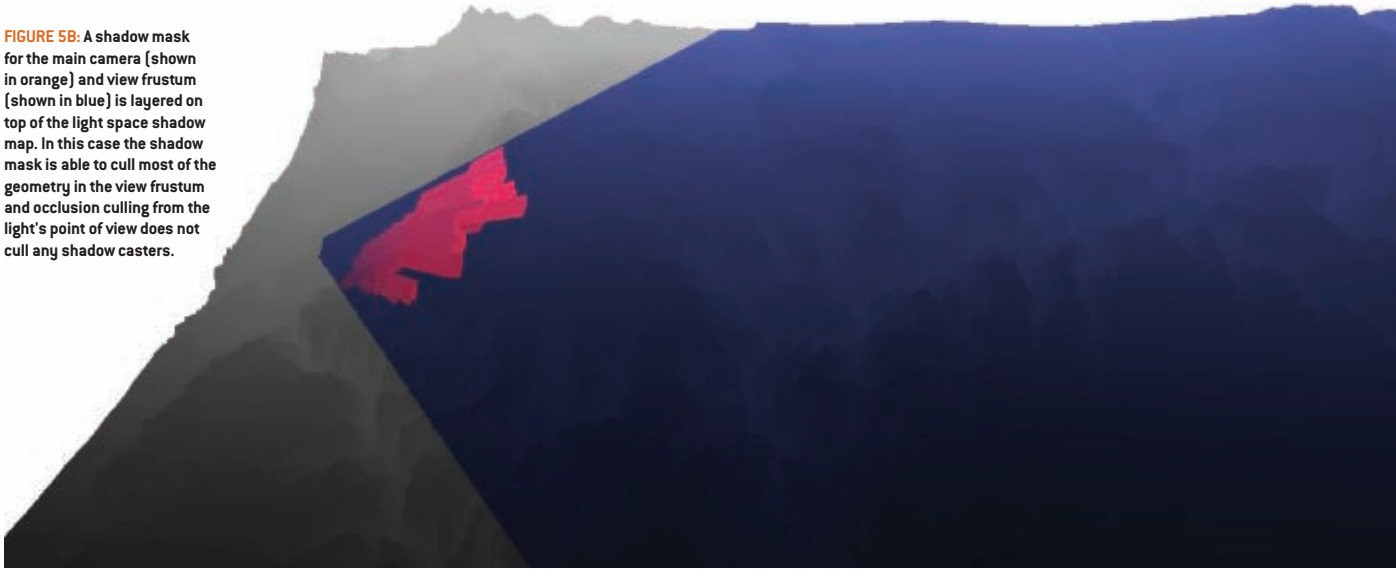


FIGURE 5B: A shadow mask for the main camera (shown in orange) and view frustum (shown in blue) is layered on top of the light space shadow map. In this case the shadow mask is able to cull most of the geometry in the view frustum and occlusion culling from the light's point of view does not cull any shadow casters.

Since the query geometry consists only of a few vertices, the predicate initialization pass is completely fill-bound on the GPU. As an additional optimization to reduce the fill cost of this process, we could conservatively downsample the shadow mask to a lower resolution for the predicate rendering pass.

SOFTWARE OCCLUSION SYSTEMS

» Software-based visibility systems have recently regained popularity, and there are several high-end game engines and AAA-titles that have adopted this approach. Regardless of whether the system is based on potentially visible sets (PVS), cells and portals, or a more straightforward software rasterizer with custom occluder geometry, it is possible to output a conservative depth buffer based on the visibility query results.

The good news is that as long as we have a conservative depth buffer available, we can compute the shadow mask using the min/max pyramid approach—as described above—by using a trivial minimum depth bound of zero. This operation and the subsequent shadow caster culling is thus perfectly suited for a software implementation.

CONCLUSIONS

» Our shadow caster culling method is compatible with both hardware and software-based visibility systems. We assume only the availability of a (conservative) depth buffer and hence believe that the presented technique is easy to integrate into an existing rendering engine.

The shadow mask concept has applications beyond shadow caster culling. One potentially interesting direction of future work is to generalize cascaded shadow maps with the combination of hardware-supported sparse textures together with the shadow mask, aiming for a more flexible level-of-detail management for shadows. In particular, the shadow mask could be used to select which tiles in the sparse shadow texture need updating as well as aid in selecting the correct resolution for each tile.

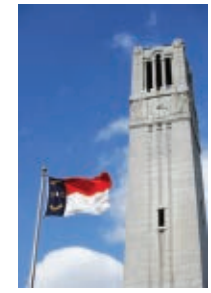
ARI SILVENNOINEN *is the research lead at Umbra Software, where he is currently focusing on next generation rendering technology. His research interests include all things related to light, shadows, and visibility in general. Ari can be found online on twitter @AriSilvennoinen and his blog www.clownfrogfish.com.*

FDG

Foundations of Digital Games 2012

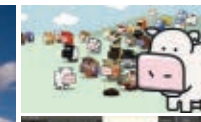
Workshops: 29th May 2012

Conference: May 30th to June 1st, 2012



Raleigh, North Carolina

www.fdg2012.org



FDG 2012, the International Conference on the Foundations of Digital Games, first held in 2006, brings together experts and researchers in all areas of game design in one place to share new ideas and contributions to their respective fields, including:

Game Studies - studying games, players, and their role in society and culture
Game Design - methods, techniques, studies
Serious Games - building and evaluating games for a purpose; learning in games
Games Education - preparing students to design and develop games
Artificial Intelligence - agents, motion and camera planning, navigation, adaptivity, dialog
Game development - tools, frameworks, networking, databases
Graphics and Interaction - rendering, modeling, animation, understanding players and interfaces, interaction techniques

FDG promotes direct dialogue and information exchange between industry and academia concerning the scientific foundations of digital games, technology used to develop digital games, and the study of digital games and their design. This year **FDG** is hosting for the first time an experimental game festival to feature new ideas in game design. These will be playable games judged to have design innovations by game design judges.

Sponsors: Microsoft Research, Microsoft Studios, Graphisme, animation et nouveaux médias GRAND Graphics, Animation and New Media

Organized by: Society for the Advancement of the Science of Digital Games (SASDG)

DUDE, HISTORY?

HOW I FOUND THE FUTURE BY LOOKING BACKWARD

Game Developer magazine has been publishing sound-related articles since its inception in 1994. It makes sense; sound has always been an integral part of the game experience. An interesting article in the first year by Jon Burgstrom broke down a new audio specification for DOS platforms which included "digital (WAVE) audio (8 or 16 bit, mono or stereo), MIDI (not just FM synthesis), volume control, minimal three-dimensional sound effect positioning." At the time, our industry was moving away from "just FM synthesis" and into a sample playback mentality that has since taken root. We can now not only play back compressed or uncompressed audio files, but we can also stream uncompressed music and sound effects in 7.1 and play hundreds of simultaneous sounds every frame as a matter of course.

YOUR HISTORY

» Jesse Harlin packed up the Winnebago and hit the dusty trail last month after six years of scribing the Aural Fixation column. Having had the pleasure of working alongside him during my time at LucasArts, and through the splendor of his writing here each month, I know that this change will find him focusing his creative energy into other aspects of his life and career. I'm not sure what that means for the future, but I hope that at some point he is able to take a step back and survey the kingdom of game audio he has helped create through these pages, knowing that he has helped us all through the trials and tribulations of this console generation in style.

Almost 20 years in production, the pages of *Game Developer* have seen audio articles from Alexander Brandon, Rob Bridgett, Vincent Diamante, Aaron Marks, Bobby Prince, George "The Fat Man" Sanger, and a host of other wordsmiths intent on keeping you in the loop. Game designer Jonathan Blow

contributed articles on real-time sound filtering (DSP), Andrew Boyd speculated on the impact that Microsoft's DirectMusic would have on interactive music in comparison to audio engines like RAD Game Tools, Miles Sound System, and Headspace's Beatniks Audio Engine. In his February 2000 article on physically modeled audio, Mark Miller says, "According to some people in the game development industry, physical modeling synthesis is the 'next big thing' in interactive audio," and I still believe it!

Through three iterations of the PlayStation console, both Xboxes, and enough Nintendo hardware to fill every living room, there has always been someone here representing audio and attempting to keep everyone on the same page with the challenges of the day.

THAT WAS THEN

» I tend to lean heavily on the side of nostalgia when it comes to game sound. I spent my youth dropping quarters at local arcades, bars, and pizza parlors. I can still vividly remember beating the mother brain in METROID, trading off turns mapping and playing with my good friend and neighbor throughout the night. The slow crawl of 300 baud across the telephone lines waiting for the latest game to land on the 1541 of my executive C64; the slow intro crawl, all bouncing rasters and SID chip tonality, communicated directly into my adolescent skull.

I carry these memories with me (literally, I just unpacked every console and cartridge I've ever owned), and they help inform my view of how far we've come in game audio and games in general. They are the things that brought me here and, in some ways, help keep me focused on the future. Similarly, articles from the past can help us map well-trodden paths, dead ends, or secret passages that got lost along the way.

THIS IS NOW

» In the aforementioned article on real-time DSP from 1998, we learn some of the hows and whys of sound filtering and how it can be used to interactively enhance gameplay. These techniques are still valuable, and they may seem like veiled secrets if you don't know where to look for them. Things like obstruction and occlusion may have become commonplace in today's middleware engines, but they're less often used in the expansive ways described by Blow: "This cannon blast is happening on the other side of an echoey ravine, and it's passing through a damp fog bank on the way here, and the weather is very windy, and by the way I'm listening to the sound underwater."

Similarly, in the piece on physical modeling, while somewhat focused on the replacement of musical instruments, the Stacatto Systems SynthCore Mission Control tool is shown as part of a parametrically synthesizing vehicle engine. What might have sounded like a great leap forward from FM synthesis at the time would now sound like nothing more than a hungry Midwestern mosquito in your ear. The value of having this perspective comes from being able to see and hear the march of progress, and then using it to frame future possibilities.

Anyone looking to jump-start their own education would do well to peel back the pages of history and gain a better understanding of where we've been. It's been said that those who don't know history are destined to repeat it, and in a lot of ways we're still building on the challenges of the past. Even with increased fidelity, new resource challenges, and the demand for greater interactivity, we still need every trick in the book to make today's experiences sing. Knowing where we've been allows us to make a firm next step forward from the edge of a well-worn path.

In the spirit of those that have come before me, I'll do my best to keep things fresh and forward-thinking. On the shoulders of giants, guided by the stars, toward the next generation of game audio.

"...and in the end the love you take is equal to the love you make." —The Beatles

DAMIAN KASTBAUER *is a freelance technical sound designer and can be found musing on game audio at LostChocolateLab.com and on twitter @LostLab.*

IMAGE FROM WWW.ISTOCKPHOTO.COM. ART BY KELSEY KRAUS

BREASTPLATE vs BOOBPLATE

ON FANTASY ARMOR AND LADY BITS

A brilliant Tumblr feed, Women Fighters in Reasonable Armor, inspired me to add my personal thoughts on the subject of female armor in fiction.

Why does my opinion matter? I'm an armorer. I make actual armor that people wear when they hit each other with swords. When making armor I have to strike a balance between comfort, protection, range of motion, and appearance. My experience has made me more than a little opinionated on the subject of fantasy armor.

I intend to set the world straight. What follows is a discussion of how to do it wrong, how to do it right, and why you might care.

THE PROBLEM

» There is a commonly-held understanding in fantasy, game, and science fiction communities that female armor sucks. That is, it doesn't really cover any vital organs. It appears to follow the relationship described by equation 1.

FIGURE 1: *Red Sonja*, Dynamite Entertainment.

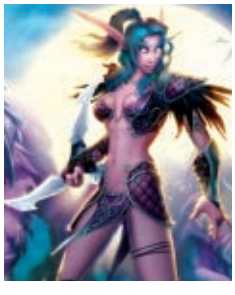


FIGURE 2: Night Elf, WORLD OF WARCRAFT, Blizzard Entertainment.

What does that relationship imply? It means that there is a wealth of fantasy artwork trying to sell the idea that chain bikinis (Figure 1) and steel push-up bras (Figure 2) are things that women might wear to a sword fight. Clearly these women are both poorly insulated and have no particular intention of keeping their vitals inside their bodies.

We know why these images exist. They appeal to a specific market. That, though, is a different discussion. All we want to establish here is that there is a rather strong trend toward dressing women in metallic lingerie rather than protective armor in fantasy combat.

To predict a counterpoint: There are men that wear next to nothing in fantasy art as well. Take Conan or He-Man, for example. Neither of them wears much in the way of protection. This is true, but they aren't meant to be armored. Both of the ladies described earlier are wearing armor, not barbarian-style loincloths. Their metal garments describe access to real armor, and their decision not to wear it.

What can be done?

THE HISTORICAL PROBLEM

» My first choice when armoring women is to draw from history. Unfortunately, there are a few problems with that:

» **Women have traditionally been restricted from fighting.**

» **The few that were allowed to fight would have mostly been commoners unable to afford quality armor.**

» **The period in history in which plate armor was actively used was very brief: in Europe, around 1400 to 1600 AD.**

This leaves us with barely any extant examples of women in armor. Even if there were women warriors, they would likely be wearing the same thing as the men: hardened lamellar leather, chain hauberks, or coats of plates.

Fully kitted in this stuff, they'd be indistinguishable from men. In combat that is just fine, but for artistic purposes, we usually like to have our characters clearly gendered. So, we can't just look at what real women wore and expect to get very much of value for our modern designs.

FIGURE 3: Joan of Arc, c. 1485.

FIGURE 4: St. George, c. 1504.

FUNCTIONALITY

» We can't pull much from historical examples of the appropriate gender, but we can still let the expertise of the ages inform us in terms of what would make sense.

Plate armor is the way it is largely out of necessity. The layout and articulations of the plates are the best solutions the designers could come up with to balance mobility with protection. Also, note that nobody was naked under their armor. There was always padding between

$$Defense \propto \left(\frac{Exposed\ Skin \times Cup\ Size}{Comfort}\right)^{Chance\ of\ Wardrobe\ Malfunction}$$

EQUATION 1: Governing relation for apparent female armor quality.

the metal and the flesh that absorbed the energy of the blows. That means the difference between male and female plate armor is relatively trivial, because once you've padded it out and left space for movement, you've all but erased the figure of the person inside. Let's grab some examples to show this in action. The first is an artist's interpretation of St. Joan of Arc (Figure 3), the second is of St. George (Figure 4), both of whom are wearing variations on German Gothic-style armor.

Note the differences in the armor as depicted by artists of the time period. There are none. Both are fully covered and both have prominent chests and narrow waists. This is pretty common because that is how armor worked; It was a functional necessity more than it was a style. Want another example? How about a contemporary interpretation on the theme?

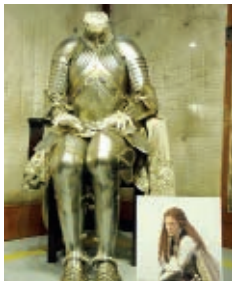


FIGURE 5: **Elizabeth's Armor, *Elizabeth: The Golden Age*, Universal Pictures.**

The armor shown in Figure 5 is from the film *Elizabeth: The Golden Age*. It is gorgeous. Modeled again on German Gothic plate, I have only a minor gripe with it: no neck protection. That's important stuff, but let's look more at what they did right.

They made the armor functional yet feminine with the detail work. The overall form could easily go on a man, but the trim, the collar, and the cuffs were character and period appropriate. Brilliant.

However, artists aren't always aiming for practicality or historical relevance. Style will often trump realism in costume design. Consider one of the most epic suits of armor ever worn, by Sauron from *The Lord of the Rings*. If this guy lifted his arms too high, he'd poke his eyes out with his own pauldrons. It's awesome but impractical armor, so why don't we deride this design? Because we believe that it's appropriate for the world and the character.

BREASTPLATES AND BOOBPLATES

» The comparison between breastplates and boobplates serves well to illustrate the competition between style and function. Breastplates are what you call the large metal shell worn over the torso that protects pretty much all of the important squishy bits. They're designed to deflect blows and distribute impact (see Figure 6).

FIGURE 6: **Italian Breastplates, Palace Armoury, Malta.**

Pretty much all plate armor uses variations on this design. Counter-examples like the Roman musculata are primarily decorative, worn by important folk that didn't much expect to actually be fighting in them.

FIGURE 7: **Male Commander Shepard, MASS EFFECT 2, BioWare.**

FIGURE 8: Female Commander Shepard, MASS EFFECT 2, BioWare.

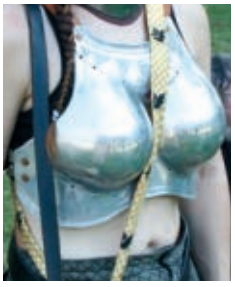


FIGURE 9: **Combat Archer Breastplate, Ryan Consell.**

Boobplates are ostensibly breastplates fitted to a female torso. That is, they have actual breasts dished out. Figure 9 shows a boobplate. I made that one myself. The woman in the photo asked for it to be like that. She fights in it. I worry constantly that she's going to fall hard and it will crack her sternum, even with the padding. Note also that it seems almost perfectly designed to guide sword points and arrows into her heart. They still have to penetrate the armor but, honestly, that's a design flaw. However, it looks good, serves her specific purposes, and makes her feel sexy and badass at the same time. That's important, too.

So we have a bit of a new problem: We want to make people look good. We want characters to be sexy. We want that more than we want realism in our fantasy art, but we also want to feel like what they are wearing makes sense. The armor should complement the character and setting, not distract from it. How do we do it?

RECOMMENDATIONS

1. Internal consistency.

Any science fiction or fantasy world runs on its own set of rules. The fashion, technology, values, and physics are all free to be laid out by the creative minds involved. Maintaining some logical consistency in what people wear for armor adds a lot to the world. If men and women are going to be fighting the same battles, afford them the same level of protection.

Comparing the armor for both the male (Figure 7) and female (Figure 8) versions of Commander Shepard's default armor from MASS EFFECT 2, we can see a good example of maintaining a consistent standard. In contrast, TERA ONLINE did this very, very badly (Figure 10).

FIGURE 10: **TERA ONLINE, Bluehole Studio.**

2. Go for the eyes, Boo.

Any artist working with human subject matter will tell you that the face is the most important part of the character. A headshot by itself can tell you everything you need to know about who a person is and how they feel. Sex appeal can come entirely from a beautiful face; the body doesn't need to be naked as well.

I argue that just a face (Figure 11) is more appealing, sexy, and descriptive of a character than an exposed chest (Figure 12) could ever be.

FIGURE 11: **NEVERWINTER NIGHTS, BioWare.**

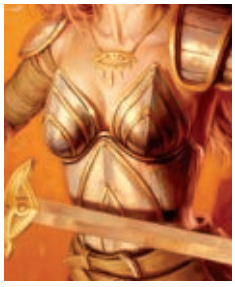


FIGURE 12: NEVERWINTER NIGHTS, BioWare.

The bare chest and metal bustier add nothing to the femininity, sexiness, or appeal of the character. In design, focus instead on the face for character appeal; let the armor be a reflection of the setting and her role within it (with the caveat that you'll likely need more close-ups).

3. Unwrapped Christmas presents aren't exciting.
So you still want your fantasy fighters to be sexy? How about a bit of a tease? Let our imagination run away with us.

People will always want to see more than they're allowed. An exposed ankle will make someone blush if they've always been denied access to them in the past. If your characters are naked, there's nothing to tease us with. A well-considered bare shoulder can be way sexier than full frontal nudity. Put a bit of thought into when and how you expose your characters. The anticipation and the idea can be more enticing than the full show.

Not convinced? Let's consider Tali'zorah (Figure 13). In two full MASS EFFECT games, we have not seen her out of her armor, and yet one of the most compelling moments in the game was when she took off her mask to make out with Shepard. Moreover, you didn't see her face even then; it was sexy because of the idea that she was revealing herself. She's alluring because of the idea of what she could be. The mystery is sexier than the reality could ever be.

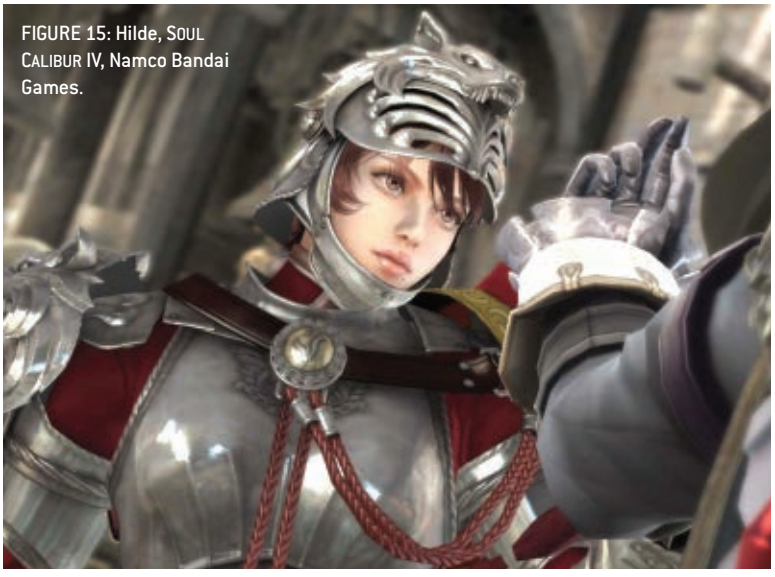


FIGURE 15: Hilde, SOUL CALIBUR IV, Namco Bandai Games.

4. Everybody is naked under their clothes.
Maybe you still want to make pictures of pretty girls in very little clothing. I won't stop you. There is a time and a place for such things, and I am not about to try to dictate terms on that front. This is just a plea for reasonable armor. So if you need to have a female warrior with exposed flesh, could you let her be in a state of undress rather than depict her default state as being mostly undressed?

Consider Figures 14 and 15, two women with rather substantial armor exposing their figures. We can have our cake and eat it, too. Wasn't that easy?

FIGURE 13: Tali, MASS EFFECT 2, BioWare.

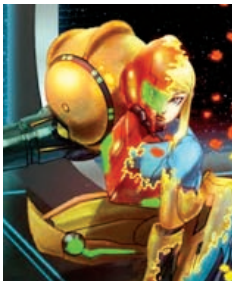


FIGURE 14: Samus, Sung Jin Ahn.

REVIEW

» Historically believable armor is good (Elizabeth). Exposed midriffs and miniskirts are bad (Figure 16). Stylized, figure-flattering suits are fine (ala female Shepard). Metal lingerie that can only be reasonably held in place with glue is just silly (Figure 17).

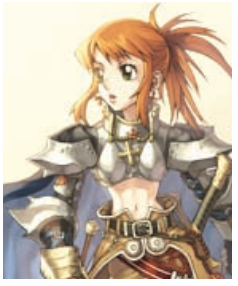


FIGURE 16: RAGNAROK ONLINE, Gravity.

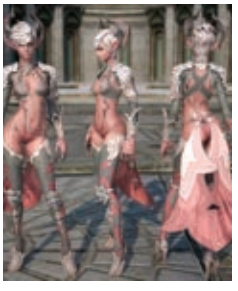


FIGURE 17: TERA ONLINE, Bluehole Studio.

A version of this article was originally published on MadArtLab.com.

RYAN CONSELL *is an armor smith, blogger, and avid gamer from Ontario, Canada. He has been making armor since 2001 for both costume and combat purposes.*

Now seeking

Game Faculty

fullsail.edu/jobs

Interested in leading the next generation of game designers, artists, and producers? Full Sail University is hiring passionate, qualified faculty for its degree programs.

Full Sail offers a creative, tech-driven working environment, competitive pay and benefits – and an opportunity to work among a team of enthusiastic industry professionals who have instructed some of the gaming world's brightest stars.

To learn about our faculty opportunities, visit us at the

GDC 12 Career Pavilion Booth #1228

or visit

fullsail.edu/jobs

FULL SAIL UNIVERSITY®

800.226.7625 • 3300 University Boulevard • Winter Park, FL 32792

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

To view detailed information regarding tuition, student outcomes, and related statistics, please visit fullsail.edu/outcomes-and-statistics.

Recruitment at GDC

WHAT ARE HR MANAGERS LOOKING FOR AT THIS YEAR'S CAREER PAVILION?

\\\The Game Developers Conference, held each year in San Francisco, has grown into one of the industry's best and most exciting venues to make new connections and find jobs within the game industry. The next show is set to take place March 5–9, 2011, and recruiters from throughout the industry are primed to seek out new talent from all over the world.

For developers looking to break into (or find a new job within) game development, GDC's Career Pavilion offers a perfect opportunity to connect directly with recruiters from all types of companies, ranging from big budget console developers to up and coming mobile studios. In anticipation of the show, we talked to a few recruiters from Nexon, Riot, Ubisoft, Wooga, and more, to hear their opinions about their recruitment practices at GDC 2012.

Game development now encompasses a wide swath of platforms, and distribution models, from console gaming, to Facebook and beyond. With this diversification, some companies have chosen to look for new recruits that have a firm grasp of the industry's flourishing markets and emerging trends.

"We are hiring far more people with experience in online games, including data analysts, economic designers, and network programmers, among others," said Ubisoft international events specialist Anne Rubio. "As the audience of gamers expands, so too do the types of jobs that are available, which is great news for people who want to work in the industry"

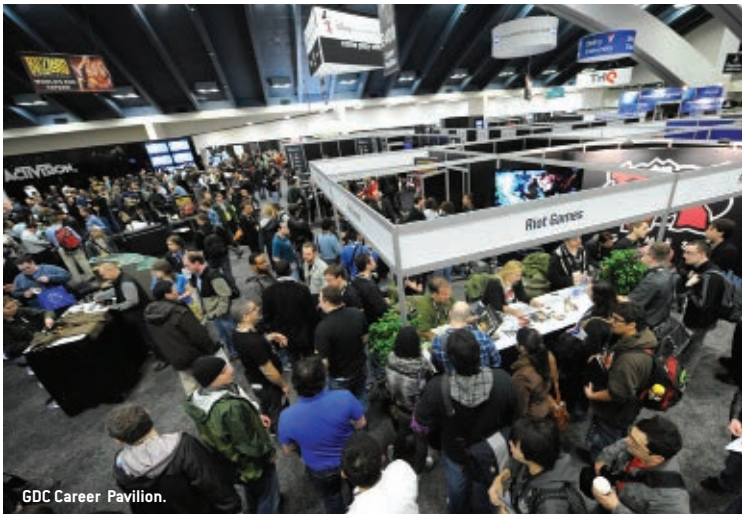
In particular, Rubio emphasized that even candidates for console development jobs should be aware of popular trends in other sectors, as these other markets are the most likely to change the way players want to consume their games.

"We think AAA games will increasingly adopt the best features from social and mobile games, and vice versa," she said. "To be successful, we have to adapt to the way people like to play."

An anonymous source from another major game developer agreed that candidates should be well-versed other markets, so long as their particular skills remain sharp and up-to-date with modern development. At times, developers can get slotted into a specific skillset and lose larger perspective.

"Trouble would arise if one has been in the social space or console for too long and looking to switch—are their skills on par for that particular segment of games? Moving from one 'genre' to another is generally not an issue, unless their skills have been left on the shelf too long and are outdated for what we are looking to do," the source said.

Crytek's HR manager Andrea Hartenfeller agrees."It depends on the individual's skills, abilities and work history," she says. "We can't say that moving between genres poses problems as a general rule. There are situations where the change works, and there are situations where it doesn't work."



GDC Career Pavilion.

When it comes to hiring people up from more junior roles, Wooga's head of HR Gitta Blatt had positive things to say, reflecting a trend toward hiring self-starters or self-taught developers, as was often done in the early days of games. "Expert experience today has nothing to do with age or number companies worked with," says Blatt. "Some of these main actors on the gaming stage are really such creative, ambitious, talented and successful that the old view of hierarchy is completely out of use."

Our anonymous source agrees. "I think that if the candidate is that outstanding—regardless of whether there is an opening or not—we would take the opportunity to hire them," said the source. "Skillsets that are rare or in high demand don't just appear every day, and you have to be able to seize the opportunity, and timing is always everything."

Interestingly, Riot's senior recruiter Valerie Lee takes the discussion one step further, valuing passion above all. "Ultimately, Riot values passion and aptitude above experience, degree, or institution," she says. "If we believe you have passion and aptitude, there is probably a role for you at Riot."

Students are another story. Game schools have become more plentiful in recent years, with many students of game programs going off to do great things, even forming their own companies after graduation. For Ubisoft, for example, the type of university a student goes to is important. Rubio says the company's policy is to "actively recruit from top universities that provide students with a strong education in both its theoretical and practical components."

But sometimes school isn't enough. A little time at a studio, in whatever capacity, can convince companies that a prospective hire is serious about working in the industry. Nexon's recruiting coordinator/sourcer Jennifer Di Pietrantonio explains. "We always look at students that come from game school programs, however, students that have had hands-on experience through part time work or an internship tend to have a better chance of success," she says. "These students understand the business side as well as the creative side."

Regardless of your experience level, in this rapidly-changing industry, it behooves any potential hire to be aware of the ins and outs of online games, as much of the industry is trending in that direction. Aside from that, talent is king! Happy job-hunting!

ONLINE PRICING ENDS FEB 29. **Game Developer readers, register with code GDC12GDM and receive an additional 10% off!***

GDC 12

learn / network / inspire

GAME DEVELOPERS CONFERENCE® 2012

SAN FRANCISCO, CA | MARCH 5–9, 2012 | EXPO DATES: MARCH 7–9, 2012

The **Game Developers Conference**® is the world's largest and longest-running professionals-only game industry event. Join us at GDC, March 5–9, 2012, for five days of learning led by some of the biggest names in the industry.

learn

SUMMITS AND TUTORIALS
(MONDAY & TUESDAY)

- AI
- Game IT
- Localization
- Games for Change @ GDC
- GDC Education
- Independent Games
- Smartphone & Tablet Games
- Social & Online Games

MAIN CONFERENCE SESSIONS
(WEDNESDAY–FRIDAY)

- Audio
- Business, Marketing & Management
- Game Design
- Production
- Programming
- Visual Arts
- Monetization (SPONSORED)
- Game Career Seminar (FRIDAY)

network

GDC Play
(TUESDAY–THURSDAY)

Business Center
(WEDNESDAY–FRIDAY)

Career Pavillion
(WEDNESDAY–FRIDAY)

inspire

INDEPENDENT GAMES FESTIVAL
(MONDAY–FRIDAY)

game developers choice awards
(WEDNESDAY)

Expo Floor
(WEDNESDAY–FRIDAY)

REGISTER ONLINE BEFORE FEBRUARY 29TH!

VISIT WWW.GDCONF.COM FOR MORE INFORMATION.

UBM TechWeb

*DISCOUNT CODE IS GOOD FOR ALL ACCESS, MAIN CONFERENCE, OR SUMMITS & TUTORIALS PASSES ONLY. DISCOUNTS CANNOT BE COMBINED WITH OTHER DISCOUNT CODES OR GROUP PASSES/DISCOUNTS. RESTRICTIONS APPLY AND DISCOUNTS ARE SUBJECT TO REVIEW. DISCOUNT CODE MUST BE USED BY THE END OF ONLINE REGISTRATION, ON FEBRUARY 29, 2012 AT 11:59PM ET.

THE COMING STORM

HOW CLOUD GAMING COULD CHANGE GAMES

Ever since OnLive's dramatic public unveiling at GDC 2009, the game industry has been watching and wondering about cloud gaming, at times skeptically, at times hopefully. The technology holds the potential to revolutionize the business, perhaps forever destroying the triangle that connects consumers with hardware manufacturers and software retailers.

Some of the immediate benefits are obvious. Instant, time-limited demos would allow every developer to showcase its games on demand with no extra work. Frictionless per-day, or even per-hour rentals would bypass Blockbuster and other rental chains, potentially meaning that more money goes directly to the people who actually make the games. Similarly, virtual ownership handicaps GameStop's ability to resell a single disc multiple times, again making sure that the money flows directly from the consumer to the developer. Further, if a publisher commits fully to the cloud—with no offline version available—piracy would be virtually impossible.

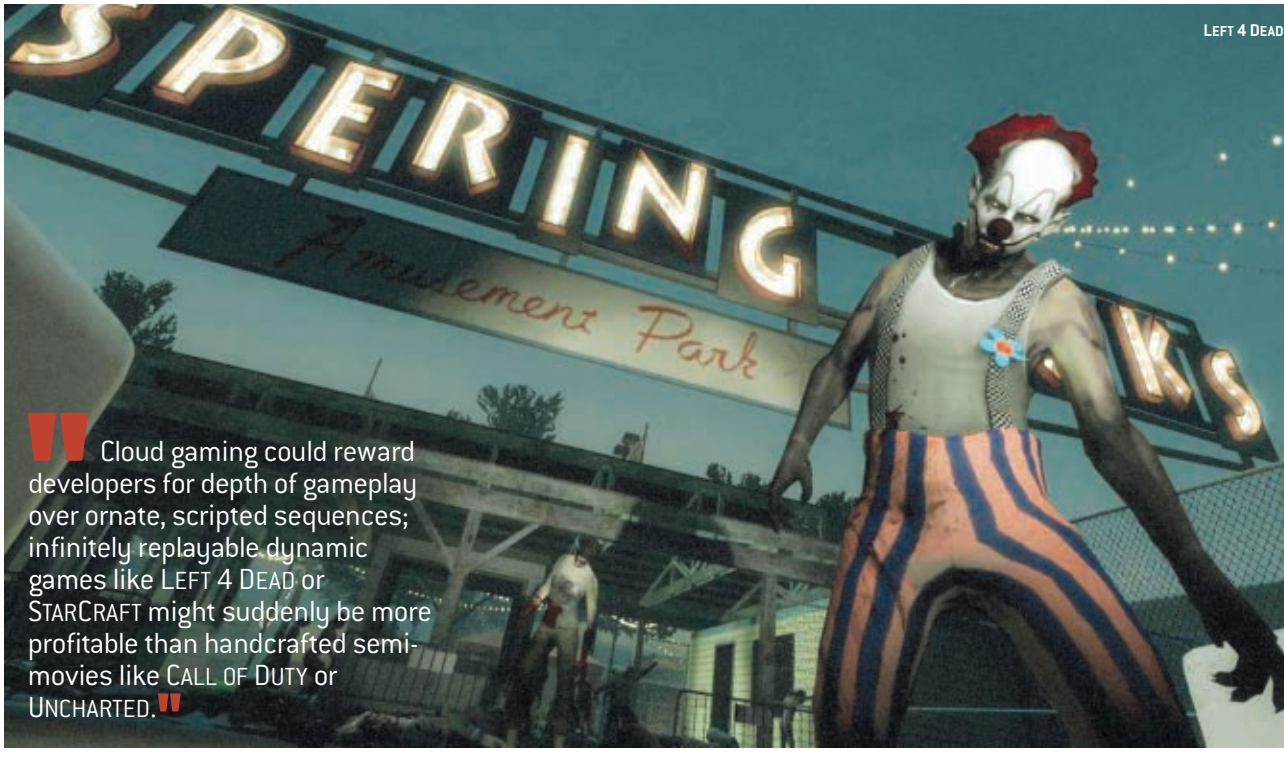
And for the consumer, cloud gaming enables cutting-edge graphics on any connected device, with no installing or patching ever. Although the system requires a constant internet connection, more and more games—even single-player ones—are demanding a connection anyway. Indeed, cloud gaming can handle internet stutters better than local gaming does; in DIABLO III, a dropped connection sends the player back to a checkpoint screen, while games on OnLive bring the player back to the exact frame they last encountered.

Most importantly, cloud gaming should change the economics of pricing. By removing the traditional retail middlemen, not to mention secondary drags on the system like rental and used-game sales, a developer could easily make as much money selling a game for $30 via a cloud service as they could selling it for $60 via a traditional retailer. The industry could finally approach a mainstream price point, with games priced comparably to movies, books, and music—instead of the $60 price point (after buying a $300 console), which is absurdly out of reach of the average consumer.

Indeed, the economics could change for developers, too. If entirely new business models emerge, with consumers paying for a game daily, weekly, monthly, or perhaps with a single subscription to all available games (a la Netflix), the design incentives change. Cloud gaming could reward developers for depth of gameplay over ornate, scripted sequences; infinitely replayable dynamic games like LEFT 4 DEAD or STARCRAFT might suddenly be more profitable than handcrafted semi-movies like CALL OF DUTY or UNCHARTED.

RETHINKING CONSOLES

» Cloud gaming also has important implications for the next generation of consoles. The ability to run games from the cloud gives the console makers a profitable



LEFT 4 DEAD.

"Cloud gaming could reward developers for depth of gameplay over ornate, scripted sequences; infinitely replayable dynamic games like LEFT 4 DEAD or STARCRAFT might suddenly be more profitable than handcrafted semi-movies like CALL OF DUTY or UNCHARTED."

alternative to both the rental market and the retail middlemen, all within their own closed systems. As a bonus, they could even sell inexpensive "cloud-only" versions of their next-gen console, without optical drives or hard disks. (Microsoft already allows saved games on the cloud with its 360.)

Going down this path, however, raises the thorny question of whether consoles are even necessary at all. OnLive is already selling a "MicroConsole" that provides a current-gen console experience via the cloud; there is no reason similar technology can't be included by default in new TVs, or even in any cable box or satellite receiver. OnLive and Gaikai both have services that don't require a console at all. What defines a console, after all? The three necessary elements are the controller, the screen, and the couch. Soon, anyone with those three things and an internet connection to the cloud will be seconds away from any game.

Also, because cloud servers already dwarf current-gen consoles in horsepower, they can bring a next-gen experience to consumers today by default. The cloud promises a "perpetual" virtual console that gets updated regularly as new, faster servers come online. Publishers should be receptive as a perpetual console promises to end the boom-bust cycle they experience with each new generation.

This year in the current generation's life cycle should be when publishers are making record profits, but instead, many are fighting just to stay in business; the next wave of forced upgrades could wipe them out. Anything that could prevent the gap years when consumers are forced to migrate between consoles would be a welcome change.

Thus, for the console makers, cloud gaming's promise is mercurial—it could break them free from the parasitic drag of traditional retail, but it could also destroy them by making the hardware itself irrelevant. The best defense against the latter is an active and direct relationship with the consumer, not tied to any one machine.

On this front, Microsoft, with its comprehensive LIVE service, is far ahead of Sony and Nintendo. Many gamers would be hesitant to leave behind their Gamerscores, Achievements, friends lists, and downloadable games for another eco system. However, a radical step could cement this bond.

Because a thin video-based client can run on almost anything, any one of the console manufacturers could start the next generation tomorrow by simply buying OnLive or Gaikai and embedding it in the next system update. Next, they could sell cloud-only versions of the current-gen consoles for almost nothing ($100? $50?), which could revolutionize the market and inoculate the company from the coming shift. Some are predicting that the next generation of consoles will be the last one, but it may not even be necessary at all.

RETHINKING GAMES

» However, cloud gaming's potential is much, much greater than changing the economics of the industry; in fact, it could revolutionize the very way games are made. For starters, the cloud could solve the number one problem that plagues most teams: a lack of feedback from real players during the early stages of development when radical change is still possible. Most game projects grow slowly from fast and nimble speedboats to hulking battleships that can only change course at great effort and cost.

Using cloud technologies, a team could expose its game to fans as soon as it is playable, with almost no technological hurdles or security concerns. All players need is a browser and, if necessary, a password. Releasing games early for feedback and buzz is nothing new for indie developers (doing so is actually one of their major competitive advantages); nonetheless, for major publishers, the idea is fraught with the potential risk of leaked games and bad press.

As for prerelease buzz, the greatest danger, of course, is of simply releasing a bad game, and the surest way to do so is to isolate a development team from the oxygen of real players. Further, the cloud's inherent flexibility creates myriad ways to target players for testing: 24-hour passes, GeoLocked sessions, early press versions, pack-in codes, and so on.

Further, the cloud promises more from these early tests than some simple metrics or private forum comments. Because the output of a cloud server is a video feed, developers would have access to a recording of every minute ever played of their game. Wonder how players are handling a certain tricky boss? The designer can simply watch saved videos of many different players tackling the encounter.

DIABLO III.

Still, the greatest change cloud gaming could bring is the end of client/server architecture. Many online games have thin clients, with the "real" calculations being done on the server, largely to prevent rampant cheating. (As Raph Koster famously put it, "The client is in the hands of the enemy.") With cloud gaming, the client is so thin that it might be inappropriate to even call it a client, as it's simply a video player that takes input.

The upside of this system is that developers would no longer need to waste resources developing a traditional game client, plugging its security holes, worrying about peer-to-peer connectivity, and optimizing what minimal, yet necessary, sets of data need to be sent to the client. In other words, making a game multiplayer would now be essentially trivial.

Writing multiplayer games is a formidable challenge—keeping game state in-sync between servers and clients in a safe, fair, and accurate manner is no small feat. With cloud gaming, these issues evaporate because there are no clients anymore. Developers simply write one version of the game, run it on a single machine, and update it based on user actions, which is how single-player games are made.

Taking advantage of this feature would require some courage, as the developer would need to go all-in on cloud technology. Developing an online game with no client means that the game could be played only via the cloud. There are many benefits to being cloud-only—no piracy, for one—but the greatest benefit might be not even needing a network programmer.

Perhaps the group that has the most to gain from this new model would be small, independent developers, for whom the idea of building an indie MMO seems laughable given their tiny resources. One can't help wonder what Mojang could do with a cloud-based version of MINECRAFT, seamlessly updated, playable from any device or browser, that connects every world end-to-end. So far, the big question for cloud gaming is when will it be feasible, but ultimately, the more important question is what will it enable.

SOREN JOHNSON *was the co-designer of CIVILIZATION 3 and the lead designer of CIVILIZATION 4. He is a member of the GDC Advisory Board, and his thoughts on game design can be found at www.designer-notes.com.*


igda
Developers helping developers
www.igda.org/join

Alpay It Forward

KENAN ALPAY MOVES FROM UBISOFT OSAKA TO SUPERBOT

Kenan Alpay is a young designer with whom I've worked many times over the years, and who most recently joined SuperBot in Los Angeles after a long stint in Japan. He's now working with AAA designers for the first time, which has opened up a whole new world.

Brandon Sheffield: How different is in-the-trenches game design from Ubisoft Osaka to SuperBot?
Kenan Alpay: It's funny, because everything feels extremely new. But sitting down and analyzing it, I'm doing a lot of the same things I did at my old company, just in a larger context. I went from leading a team of 10 people to being one of 60+ members, but I have a core group that I work with and together we own a part of the project (as opposed to the entire game).

Ubisoft has a very specific process when it comes to game design. It sometimes felt constricting, but that exposure to such an established structure helped me become a better designer. At SuperBot you have this crazy mix of people from all over the place, so that experience keeps me grounded and focused.

BS: You've mentioned to me that you've learned a lot now that you're at SuperBot—can you give some examples of what and how?
KA: The biggest thing has just been the team. There's such a mix of different people here, many of whom have worked on hugely popular AAA titles. I've never been exposed to that before. So when I'm approaching a design problem, I can discuss it with them and see how they handled a similar problem in the past, and compare our ... designer instinct, I guess you could call it?

I play a lot of popular games, but I also play a lot of fringe, hardcore indie titles, which I love. In my work I try to take the spontaneous, experimental feel that many smaller titles can afford, and work it in so that it fits with our project's structure and language ... searching for the best of both worlds. Being able to bounce these ideas off my boss and team members has been refreshing.

BS: What good things did you take from your time as a designer in Japan?
KA: One huge thing was the importance of communication! With any second language, there is a realm of nuance and assumptions that come with your words that you might not be entirely familiar with, but even in English words can mean completely different things to different people. I usually have a pretty concrete understanding of what I want, so most of the effort goes into explaining it as precisely as possible. Unless you are coding the damn thing yourself, you can't expect someone to create the exact thing you have in your head. But I found that including drawn diagrams or example videos directly into my documents worked well. The more avenues of communication, the better.

Another thing was learning about working with different personalities. I found that some people really want to be part of the creative process, offer opinions, and contribute to fleshing out those opinions. And I found that other people simply want to be told exactly what to make. Once I understood which approach worked best with which person, things moved much faster.

BS: How have you found it culturally coming from east to west?
AP: Culturally I think there is a lot of indirectness in Japan. For the sake of clarity I tried to be extremely blunt when talking about work, or reviewing other people's work. Polite, but blunt. That took some people off guard, but I'm confident it contributed to better games.

Ubisoft Osaka is in a unique situation. It's made up of mostly Japanese developers who have suddenly been introduced to Ubisoft's unique brand of design philosophy. Seeing those two philosophies mix felt like some kind of wonderful science experiment. Working in the middle of that was extremely interesting.

who went where

As part of its planned "move into mass-market online gaming," RUNESCAPE maker Jagex has appointed Rockstar Games's former art and animation director Alex Horton as its new chief creative officer.

CCP Games has hired David Reid, formerly Trion Worlds SVP, as its new chief marketing officer ahead of the launch of its next big first-person shooter DUST 514.

Nokia Connect's former program management execution office head Kaj Haggman will join Digital Chocolate's Helskini studio as its new general manager. Sulake's former user and market insight director Emmi Kuusikko will join the same office as product management VP.

new studios

Chinese online/mobile game developer and operator NetDragon revealed that it is set to form a new social games studio with Tokyo-based social and mobile gaming giant DeNA, with the aim to develop mobile social games for the Chinese market.

Tapjoy co-founders Lee Linden and Ben Lewis, along with former Playdom executive producer Craig Dos Santos, have founded Andover Games, a new mobile gaming studio that raised its initial funding by selling itself to Ascend Acquisition Corp.

Pandemic Studios co-founder Greg Borrud hopes to find success in the growing social market with the foundation of L.A.-based Seismic Games.

BLIZZARD IS HIRING

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

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

jobs.blizzard.com

Follow us on Twitter: @blizzardcareers
Visit us in the GDC Career Pavilion at Booth #1408 and scan your GDC badge for a chance to win epic loot.

THE 14TH ANNUAL
INDEPENDENT GAMES FESTIVAL
AWARDS

WEDNESDAY, MARCH 7, 2012 | 6:30-8:30PM
MOSCONE CONVENTION CENTER | SAN FRANCISCO

IGF AWARD CATEGORIES

IGF MAIN COMPETITION
- » Seamus McNally Grand Prize
- » Excellence In Design
- » Excellence In Visual Art
- » Excellence In Audio
- » Technical Excellence
- » Audience Award
- » Best Mobile Game

IGF STUDENT SHOWCASE
- » Student Showcase Finalist
- » Best Student Game

NUOVO AWARD
NUOVO

CELEBRATING OVER 800 INNOVATIVE GAMES ACROSS THIS YEAR'S MAIN, STUDENT, AND NUOVO AWARD COMPETITIONS

VIEW THIS YEAR'S SUBMISSIONS AT WWW.IGF.COM
PLAY THE FINALISTS AT THE IGF PAVILION ON THE GDC 2012 EXPO FLOOR, MARCH 7-9, 2012

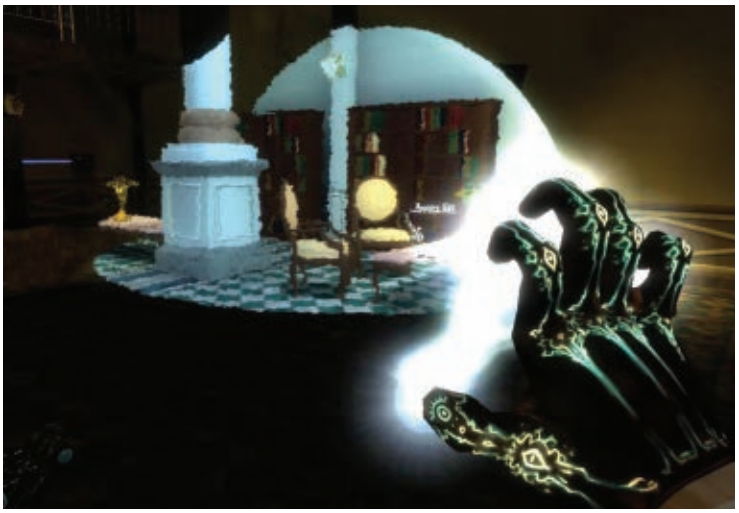
Platinum Sponsor: Microsoft Studios
Student Showcase Platinum Sponsor: DigiPen
Student Showcase Gold Sponsors: le cnam enjmin, ISART digital

UBM TechWeb

VOID

www.digipen.edu/?id=1170&proj=23876

IN VOID, PLAYERS MUST COMPARE DIFFERENCES BETWEEN A RUINED WORLD OF THE PRESENT AND A PERFECT PAST WORLD IN ORDER TO PROGRESS THROUGH THE WORLD. IT'S A NEW KIND OF FIRST PERSON PUZZLER—AND ONE THAT HAD A LOT OF POTENTIAL ROOM FOR EXPLOITS. THE TEAM, WHICH WON BEST STUDENT GAME AT THE IGF CHINA AWARDS, DISCUSSES THE MAKING OF THIS UNUSUAL TITLE.



***Tom Curtis:* Can you give a little background on the origin of the project? How did you come up with the unusual design?**

Mark Ravindran (designer): VOID was actually a student project created for the production phase of our course at DigiPen. I think its eventual design was actually a product of our limitations as well as some technological breakthroughs that we had while designing the project itself. For example, we didn't have much time [to work on] the project, and with a small team size, we figured that designing enemies with a substantial amount of AI would actually be rather difficult, so we kept things simple and created a first-person puzzler. Other things, like the dimension rip bubble, were a result of programmers discovering new ways of adjusting the engine.

***TC:* What technical challenges did you encounter?**

Chan Sin Huan (programmer): The biggest technical challenge was the Source SDK itself. While we had access to the code that made HALF-LIFE the game it is today, the SDK only provides the code and assets for the game layer itself [the AI, weapons, monsters, and levels]; it does not provide any access to the internals of the Source Engine itself. As the Source Engine is a "black box," this posed many challenges to us as we could not modify the engine to better suit our needs.

Leau Tat Sin (programmer): I was the graphical programmer on the team, in charge of creating the bubble effect. The first challenge I had was trying to figure out how to create custom shaders on the Source SDK. There were little bits of documentation, and most of them were a little outdated, since the Source SDK has undergone multiple updates since its release. There was a lot of trial and error involved.

Creating the effect also proved to be much more difficult than first expected. The basic effect was easy enough to create, but there were a lot of unforeseen artifacts, which took up most of the development time to fix. There were still a few issues that I never did solve, but unfortunately we ran out of time.

***TC:* Any design challenges?**

Tan Chee Ming (designer): The key challenge was the doubled workload for everything we wanted to design and create. For every level or room we created, the designers had to create two layouts, and the artists had to model two versions of assets: a past (perfect) version and a present (destroyed) version.

It was also a huge challenge for both Mark and I as designers, since we had to ensure the level design for both past and present complemented each other, and that there was not any easy bypass or walkthrough breaks by changing time dimensions.

***TC:* If you could go back in time and do one thing differently on this project, what would it be?**

LTS: I would have gone with the two-world approach right from the start. We essentially wasted the first half of the production time when we rebooted and threw everything away. The extra development time would have allowed us to fix the remaining issues, or at least find some kind of workaround.

TCM: On my part, I would have liked to engage this project with a clearer vision, so that the team wouldn't have had to go through so much trouble. Also, I would have liked to learn to filter feedback properly, since we received so much feedback that it actually caused us to become confused and lose our vision.

CSH: As this was part of a student course, for most of us, it was our first time creating games in an actual production environment, and it would definitely have been better to approach the whole project more methodically.

Zou Xinru (animator): I do sometimes wish we had the full five months to concentrate on the final iteration of VOID. It could have also gone down the same road of self-destruction, or it could have been much more awesome, with more levels, a third dimension, and maybe some monsters clawing at your back. At least, I would like to imagine that would be the case.

***TC:* How does it feel to earn the Best Student Game award at the IGF China? What's next for the team?**

LTS: It's surprising, really. To be honest, throughout development, we all thought that we performed badly. For the longest time, at least until the reboot, the game simply sucked. In fact, we were in danger of failing the course. Good thing it all worked out!

MR: It definitely makes up for all the pain we had with redesigning and scrapping the ideas (*Laughs*).

TCM: Now that the project is finished, all of us are at separate companies right now, but I definitely wouldn't mind coming back together to further develop VOID if someone or some company expressed interest in it.

Penn
University of Pennsylvania

Penn Engineering's

MASTERS IN COMPUTER GRAPHICS AND GAME TECHNOLOGY

offers recent Engineering and Computer Science grads:

A unique, one-year graduate-level program combining **art** with **engineering** and balancing **theory** with **practice**

A **wide array of interdisciplinary courses** from UPenn's School of Engineering and Applied Science, School of Design, the Wharton School of Business and the Annenberg School of Communication

Access to the **LiveActor Motion Capture Studio** for student game projects

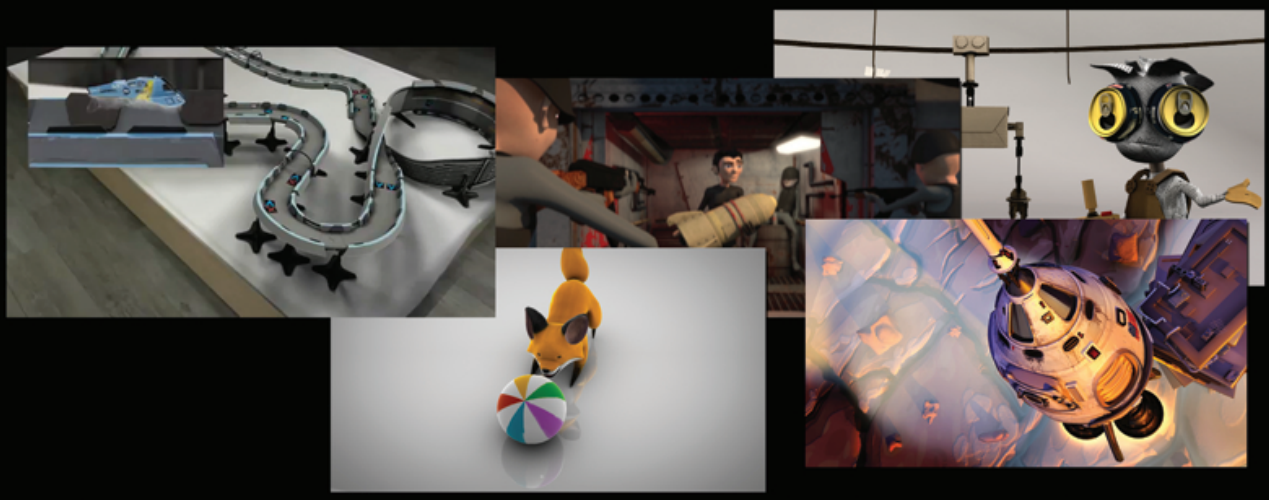
Cutting edge research opportunities at the **Center for Human Modeling and Simulation**

A world class education at University of Pennsylvania's Philadelphia-based **Ivy League campus**

GET IN THE GAME

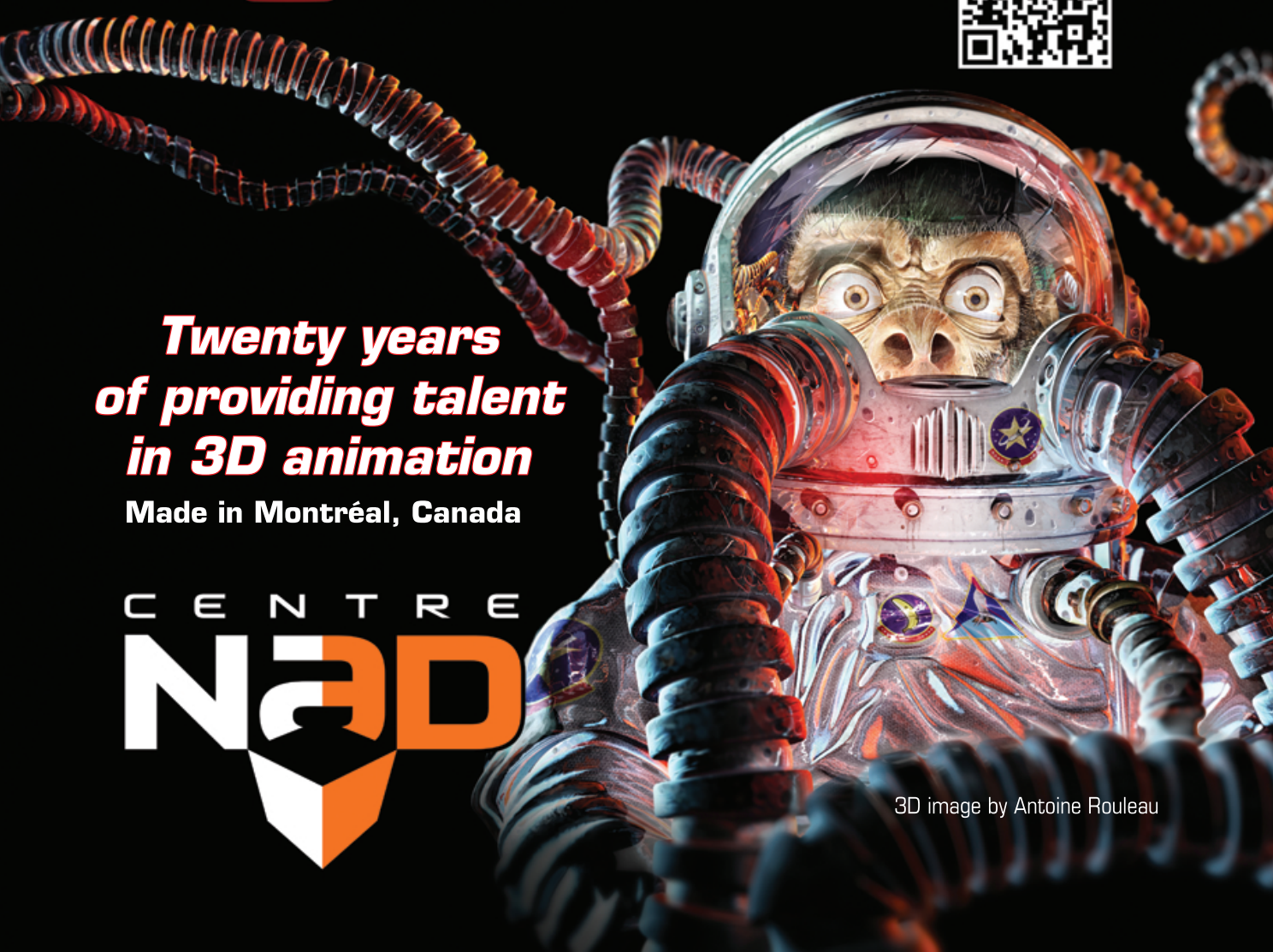
www.cis.upenn.edu/grad/cggt/

The BEST in 3D Animation...



See what **2.5M+** viewers have been amazed by:

youtube.com/CentreNAD



Twenty years of providing talent in 3D animation

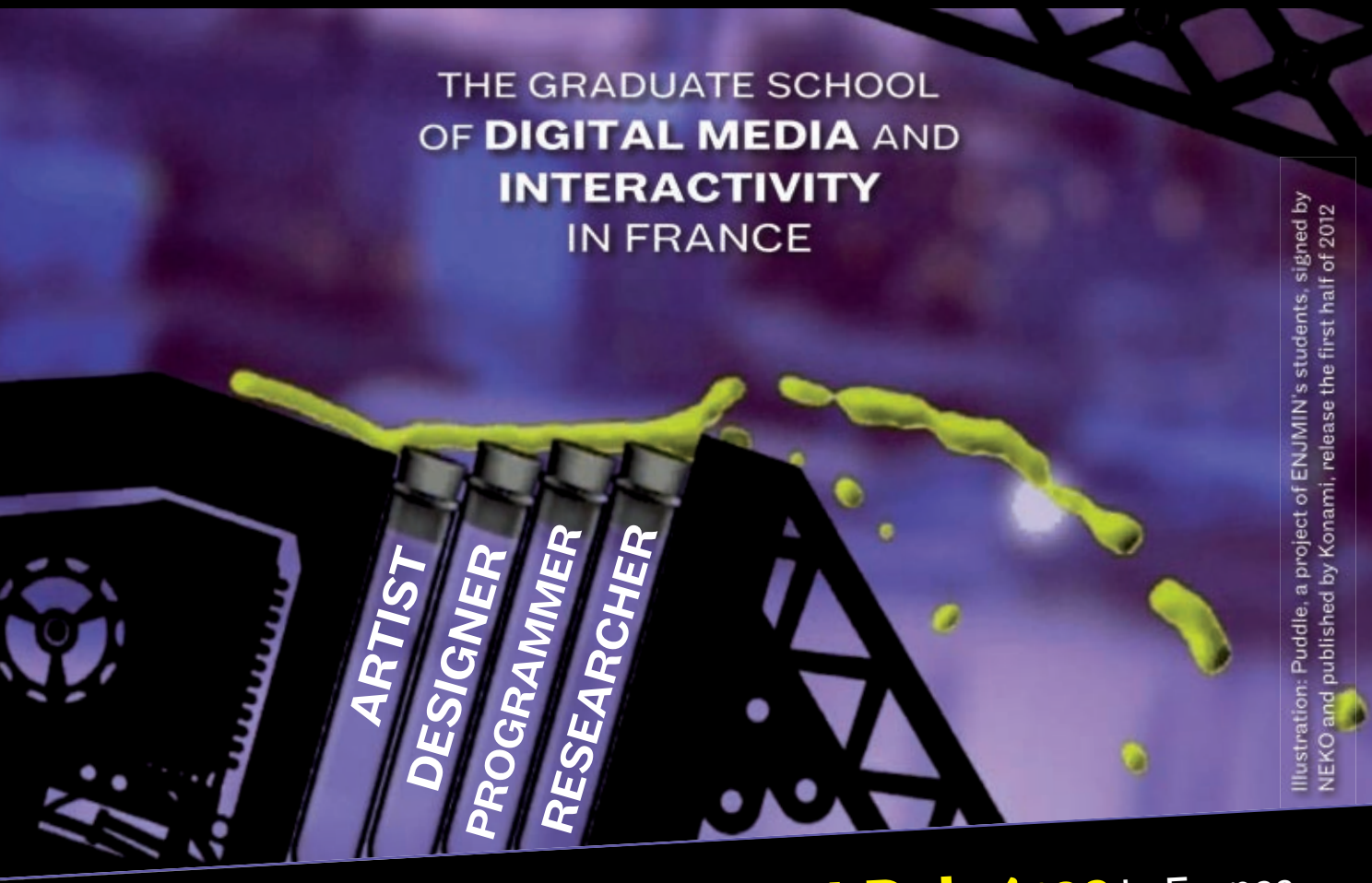
Made in Montréal, Canada

CENTRE NAD

3D image by Antoine Rouleau

le cnam

The graduate school of games and interactive media enjmin



Get a **Master**, **Engineering** or **PhD degree** in France, or get them as an **accreditation from your professional experience**.

- Game Design & Interactive Narration
- Graphic Design / 3D Art
- Sound and Music Design
- Programming and Architecture for Interactive Media Systems
- Production and Project Management
- Applied Cognitive Psychology and Sociology of Interactive Media

Take Advantage of the French Public Education System :
Low and subsidized cost, French social security System, Grants for Housing, intensive French courses during summer on the West Coast of France.

"ENJMIN, a one-stop shop for the games and new media profession !"

Visit us at GDC EXPO
Booth CP#1026

ENJMIN | The Graduate School of Games and Interactive Media
121 rue de Bordeaux 16000 ANGOULEME - FRANCE
Phone : +33 (0) 5 45 38 65 68 | contact@enjmin.fr

www.enjmin.fr

In the creative media campus Magelis Campus Image Angoulême

Screen shot from 2010 Senior Game Design capstone project, *Grunt Work*

He'd like to talk to you about majoring in Game Design.

game.colum.edu

Columbia
COLLEGE CHICAGO

INNOVATION IN THE VISUAL, PERFORMING, MEDIA, AND COMMUNICATION ARTS

Columbia College Chicago's Interactive Arts and Media Department offers majors in:

Interactive Arts & Media
Game Design

with concentrations in:

Game Art
Game Development
Programming
Sound Design

Books for any 3D Toolkit

Maya • Blender • 3ds Max + More

By Adam Watkins
$49.95

A complete end-to-end approach to teaching the entire 3D process in Maya.

By Eric Luhta
and Kenny Roy
$44.95

This is an animator's workflow with complete, step-by-step walkthroughs of essential animation techniques to increase your efficiency and speed.

By Lee Montgomery
$49.95

Explore Disney's 12 principles of animation — from squash and stretch to timing and appeal, while learning how to animate in Maya.

For a look at our 3D titles, visit www.focalpress.com

Focal Press

Creativity has an endless shelf life.

Focal Press Books are available wherever fine books are sold or through your preferred online retailer.

focalpress.com

ADVERTISER INDEX

gd Game Developer (ISSN 1073-922X) is published monthly by UBM LLC, 303 Second Street, Suite 900 South, South Tower, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as UBM LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. ***SUBSCRIPTION RATES:*** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $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.

ASK A FROST DRAGON

REAL TALK WITH AN ELDER SCROLLS V: SKYRIM STAR

Erp.

Today we're excited to present an interview with a dragon from—

Frost dragon, actually. You can tell by the spikes here, and the ice breath.

Okay—a frost dragon, then, from THE ELDER SCROLLS V: SKYRIM! We're very excited to have you!

Thank you. I'm excited to be here as well.

So to start out, what's it like being one of the most feared and recognized characters in one of last year's standout games?

Well, first let me say that SKYRIM is a pretty darned good game, and I'm proud to be a part of it. It's also really fun. I have a blast flying around attacking and freezing things, and the world has a ton of big, wide-open spaces. It really gives me this sense of total freedom. There are also a lot of ruins I can perch on, which is important for dragons. Most importantly, there are a lot of helpless villagers to terrorize. That's sort of what I do, you know—just swoop down and freeze folks. I love seeing them freak out and run around when I show up.

At the same time, I had a couple ... well, I guess I'd call them quibbles, with the way the team chose to present me. Not that I'm complaining, mind you—like I said, I do love the game, and the world and everything, and I, personally, have been a big fan of the ELDER SCROLLS series since MORROWIND.

However ...

However ... gosh, I don't know quite how to put it. [*Thinks for a moment*] Like, do you know how annoying it is to sit there breathing ice all over this player, and then they just gulp down a couple potions, or eat a bunch of goat legs and cheese wheels and they're totally fine? I mean, that kind of thing comes close to ruining the immersion for me. I want to freeze this human right now! It's not fair to just let them pause the game and recover.

Well, it *is* a video game, so one could argue that—

Sure, I get that aspect of it. I get that it's a game, and that it's about the player's experience. And I think I did a pretty good job of playing my role and all.

Don't get me wrong. I love the big skies and dense forests and watching those dead town-watch guys ragdoll down the cliff sides after I've frozen them—man, that last one is terrific. I could do that all day.

The problem for me really only starts when the player shows up. It just feels like there's way more power given to that character than anyone else in the world, and that kind of bothers me.

The player does seem to be the only one who's able to effectively combat dragons, but isn't that because the player is the Dragonborn? Doesn't it have to be that way, you know, because of the prophecy?

Yeah, I don't know if I believe in all of that determinism stuff.

Well ... alright.

I don't mean to sound dismissive, but what's the point in believing all your actions are preordained?

No point, I guess.

Anyway, I'm just saying I don't think it's a hundred percent balanced. Everything seems to skew toward the player, you know, "oh, what does the player think, what does the player want." Hello, I'm here too! I hate to be the one calling "bias," but in this case, I really think it's true.

Any words of feedback for your developers, then?

[*Considers the question*] No, not really on that, because like I said, I do know why they make those choices. What I would say to the people at Bethesda Softworks is, thank you all for giving me such good art, animation, sounds, and behaviors! Thanks for saving the cool music for me, too. And—oh yeah, I did have one question for you guys:

Why am I a frost dragon in a country composed almost entirely of people with an inborn frost resistance? Do you even know how disappointing it is when you go to breathe icy death on a whole bunch of people and they turn out to be mostly fine? Next time, put me in with a bunch of humans, trolls, daedra ... I don't care who as long as they have a crippling weakness to frost. I just want to have a proper rampage at some point. Is that so unreasonable to ask for?

Alright, got that off my chest.

Understandable, I think. Thanks for taking time out of your busy schedule to chat with us.

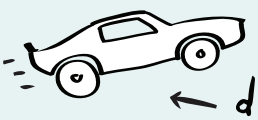
Of course. It's my pleasure.

Oh, one more note for you players. I know some of the Jarl's men came by and left that bounty letter to kill me. You should know they'll only give you a hundred gold for your trouble! It's probably not worth it.

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

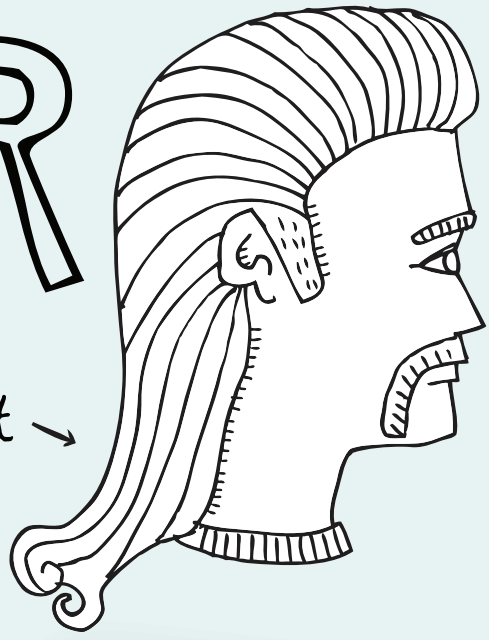
THERE ARE LOTS OF WAYS TO BE AN EVEN

COOLER GAME DEVELOPER



← drive a sweet old Camaro

like grow a mullet →



or you can use **BINK VIDEO**.

You get an amazing, super =FAST video and audio codec - all in a simple, clean API. And Bink Video runs on **EVERY PLATFORM**!

BINK VIDEO

USING BINK VIDEO NOT ONLY MAKES YOU cooler, IT MAKES YOU rad!

RAD GAME TOOLS ©

www.radgametools.com
(425) 893-4300