UBM

GAMEDEVELOPERMAGAZINE

THE LEADING GAME INDUSTRY MAGAZINE VOL18 NO7 AUGUST 2011

INSIDE: TILING PERLIN NOISE: A NEW TECHNIQUE

gd

AUGUST 2011

TERMINATOR SALVATION

ARCADE

TOTAL WAR SHOGUN 2

POSTMORTEM

THE LEADING GAME INDUSTRY MAGAZINE VOL18

INSIDE: TILING PERLIN NOISE: A NEW TECHNIQUE

AUGUST 2011

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

LEADING GAME DEVELOPERS

CONTENTS.0811

VOLUME 18 NUMBER 07

gd

AUGUST 2011

POSTMORTEMS

FEATURES

DEPARTMENTS

MOST LIKELY TO ACHIEVE

A DISCUSSION OF WHEN AND WHY ACHIEVEMENTS MATTER

LIKE MANY OTHERS, I SCOFFED AT the idea of achievements when they were introduced to the mainstream through Xbox Live. Now, there are times where I actually miss them in games where they're absent. This is especially true of games that are good, but need that extra push to be great. And more than that, if I'm going to try to suffer through something I actively dislike, such as DUKE NUKEM FOREVER, I at least want my friends to be able to know about it.

This is a strange new relationship to have with electronic media. In the olden days, you'd have to send your high score in to a magazine via a screenshot in order to get any recognition for doing something spectacular in a game. Now, with achievements and leaderboards, anyone can know that you got the "Seriously 2.0" award in GEARS OF WAR 2 (for killing 100,000 enemies). For many people this kind of reward is quite compelling, and just increasing the number of points they have in their gamerscore can be enough motivation to play a game.

But in my opinion, achievements just for the sake of achievements are not worthwhile. I like to see numbers go up as much as anyone—but I want to feel that I've earned them. Receiving 200 points at chapter endings doesn't feel like something I've really achieved. If I'm playing the game through, passing a chapter point is inevitable, if the game is fun. So how interesting is that, really? The more interesting achievements are those that encourage alternate paths or play styles, or reward exploration. But you have to do it right.

Recently I was playing DUNGEON SIEGE 3, a passable dungeon crawler with a middling story that I continued through because it got the loot mechanic right. Many of the achievements, rather than being secret, were visible to the player if they cared to look. This is fine, and in my case it compelled me to try to fulfill the requirements. But you have to be careful with even these, especially in how you describe them. For example, in one DUNGEON SIEGE 3 boss encounter you get an achievement for "defeating 50 automatons" before taking the boss down. I counted some 120 automatons defeated before I finished off the boss, just to be sure. But I received no achievement. The game was not supplying me with the correct information or feedback, and I felt like I was getting cheated out of something. After jumping through the flaming hoops, I did not get the treat at the end of the performance. On the other hand, getting new rare loot was compelling enough on its own, and I didn't feel like I needed an achievement at all here. Getting a lightning-infused Spear of Magnificence was its own reward.

Then there are games like DRAGON AGE. In this massive RPG, achievements encourage exploration of alternate narrative paths, leading players to more content that they otherwise might not see. They are actually partially responsible for improving (or extending) the player's experience. This also serves the developers' best interests, because it means less of their hard work will go unnoticed.

Now that we're all so used to these sorts of systems, what happens when we aren't provided with them? No Nintendo console has ever had a proper achievement system built in, and at times, I actually miss them. In GHOST TRICK, for example, you're moving linearly through an adventure, and achievements would necessarily be of the "progress" type, so achievements are unnecessary. But in MONSTER TALE, which is a smart beat-em-up combined with a monster raising sim, I've felt their absence. I can consistently get over 30 hit combos, for instance, which is somewhat difficult. It feels like I should be rewarded for that, or compelled to push myself further. Likewise, there are rare forms of monster you can raise, which I also felt warranted an extra award. In the case of the combos, I simply wanted to show off. In the case of the monster raising, the new forms weren't sufficiently amazing on their own, and I felt I needed an extra reward. It seems that while achievements can't save a bad game, they can give a bit of a boost to game that's 80 percent of the way there.

If achievements are indeed extrinsic rewards, not an intrinsic part of the fun, this leads me down another path. Sony is starting to charge for its online experience in used games, giving download codes for newly-purchased titles, to discourage used game sales. EA gives bonus content to those who purchase games new. What if platform holders universally decided that only purchasers of new games would have their gamerscores displayed, through a code entry, or some other such method? Would people still feel that achievements were extrinsic rewards? Would used game buyers feel cheated? This will likely never come to pass, but it's certainly food for thought.

As Nintendo prepares to release Wii U, the company is also reconsidering its online strategy. One wonders whether Wii U will continue to buck the achievement trend. Nintendo already lets you see the how often and how long you've played games on the 3DS and Wii, but doesn't let you share that information. Will Nintendo be the lone hold-out in the race to gamify the world? It seems an inevitable issue to address. Nintendo has the opportunity to provide, rather than arbitrary achievements, actual data about the games you've been playing. This could be an even more meaningful achievement system of sorts, and one which wouldn't require developer implementation. When achievements move into hard data, I think the game will be changed for the better.

—Brandon Sheffield
twitter: @necrosofty

GAME DEVELOPER MAGAZINE
WWW.GDMAG.COM

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

SUBSCRIPTION SERVICES

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

FOR DIGITAL SUBSCRIPTION INFORMATION
www.gdmag.com/digital

EDITORIAL

PUBLISHER
Simon Carless | *scarless@gdmag.com*
EDITOR-IN-CHIEF
Brandon Sheffield | *bsheffield@gdmag.com*
PRODUCTION EDITOR
Jade Kraus | *jkraus@gdmag.com*
ART DIRECTOR
Joseph Mitch | *jmitch@gdmag.com*
DESIGNER
Jessica Chan
CONTRIBUTING WRITERS
Tom Curtis
Jesse Harlin
Paul Laska
Bijan Forutanpour
Jaime Griesemer
Steve Theodore
Christian Nutt
Frank Cifaldi
Matthew Wasteland
ADVISORY BOARD
Hal Barwood **Designer-at-Large**
Mick West **Independent**
Brad Bulkley **Microsoft**
Clinton Keith **Independent**
Brenda Brathwaite **Lolapps**
Bijan Forutanpour **Sony Online Entertainment**
Mark DeLoura **THQ**
Carey Chico **Independent**
Mike Acton **Insomniac**

ADVERTISING SALES

GLOBAL SALES DIRECTOR
Aaron Murawski e: *amurawski@think-services.com*
t: 415.947.6227
MEDIA ACCOUNT MANAGER
John Malik Watson e: *jmwatson@think-services.com*
t: 415.947.6224
GLOBAL ACCOUNT MANAGER, RECRUITMENT
Gina Gross e: *ggross@think-services.com*
t: 415.947.6241
GLOBAL ACCOUNT MANAGER, EDUCATION
Rafael Vallin e: *rvallin@think-services.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
Ryan Pratt e: *rpratt@wrightsreprints.com*
t: 877.652.5295

AUDIENCE DEVELOPMENT

TYSON ASSOCIATES Elaine Tyson
e: *Elaine@Tysonassociates.com*
LIST RENTAL Merit Direct LLC
t: 914.368.1000

United Business Media
WWW.UBM.COM

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

yet Arabic game development is still in its infancy.

your opportunity is right here, right now at twofour54º in the heart of Abu Dhabi.

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

We empower businesses across all media platforms from production, gaming, digital, animation, broadcast and publishing – with world-class training from **twofour54º tadreeb**, state-of-the-art production facilities with **twofour54º intaj** and venture funding and support for Arab creative entrepreneurs from **twofour54º ibtikar** – to seize every media opportunity the region has to offer.

It's all part of our vision at **twofour54º**, creating a centre of excellence for Arabic content creation in Abu Dhabi.

we are twofour54º. are you?
find us. join us. create with us.
+971 2 401 **2454** **twofour54**.com

follow us on:

twofour54
Abu Dhabi

content creation community

*Sources: Arab Media Outlook 2010. Media on the Move 2009. A.T. Kearney. Introduction to Gaming. Michael Moore. Screen Digest. IDC.

twofour54º is an initiative of the Abu Dhabi Government.

arcade!

a mixture of video games and performance art

\\\ Seven years ago Nicolas Rosette's housemate brought home a projector to screen films on their lounge room wall. Not wanting to limit the use of the new household gadget, Rosette plugged a console into the projector, turned up the sound to match the size of his wall, and fired up REZ. Standing in the middle of the room (his controller cable was too short for him to be seated) as the music pumped and his character floated from side to side, he felt like he was playing a game that he had played so many times before for the first time in his life.

"It felt like it was meant to be played this way, on a scale of 1:1," says Rosette. "Not only was the image wide, it also felt open—there was organic and architectural feedback from the image being projected and reflected into a physical space; it didn't feel restricted by a monitor."

Little did Rosette know that this one gaming session would be a seed that would blossom into Arcade!—a full-scale touring exhibition, launching in one of Paris' most cutting-edge theatres six years later.

After taking on the role of artistic advisor at the Théâtre de l'Agora—an institution known for pushing boundaries in theatre, dance, and music—Rosette began his push for a theatre program to allow people to experience video games the way he was able to back in 2004. Although now, it wasn't just about playing the game on a big screen with loud music. After years of research into how people played games, he wanted to look at game playing as a performance.

"Performing video games can mainly be understood in two ways: as a sport performance, or as a performing art proposal," says Rosette. "While the activity itself may be the same, the intention is not. As a sport performance, it is about showing skill and mastery of the game. As an art performance, it is more about interpreting the game piece through an act of performance."

"It can be considered a bit like playing a musical score (the game) with the dedicated musical instrument (the game device). And then comes the choreographic part: the way the player 'plays' (in the choreographic or dramatic sense) creates a dynamic shape that has its own narrative (even if abstract) drive, and aesthetic value."

Rosette experimented with a number of games but eventually settled on AUDIOSURF, SPACE GIRAFFE, REZ, GEOMETRY WARS RETRO EVOLVED 2, OSMOS, and SUPER LASER RACER, because he felt these games were best suited as performance pieces that were easy to understand and spectacular to watch. His team found that while most players slouched and were sedentary when they played these games in a regular gaming environment, in the Arcade! setting they became more aware of their bodies and played differently.

"Nowadays people are playing at home, laying on a sofa and grasping their game pad, or sitting at their desk with mouse and keyboard ... All these gaming experiences involve the body, but in such a mundane way that people don't even notice they are bodily engaged in the game," says Rosette.

"Having them standing in front of a huge controller displaying a wide angle image is forcing them to be more conscious of their body. This also allows them a freedom of movement, and as the games have an important musical part, they can also start moving in rhythm ... it's like dancing."

The reception to Arcade! has been overwhelmingly positive, with theatre goers who would normally have little engagement with video games being given a chance to consider games as dynamic experiences.

Arcade! will be returning to France in November 2012 at the StereoLux in Nantes, before commencing a European tour in fall of 2012.

—Tracey Lien

the good old days

remembering the still-existing challenges of arcade games

ALONG WITH OUR TERMINATOR SALVATION POSTMORTEM, GAME DEVELOPER *PRESENTS THIS DISCUSSION OF WHAT MAKES DEVELOPMENT OF A MODERN ARCADE GAME DIFFERENT FROM A MODERN CONSOLE OR PC GAME. PLAY MECHANIX MANAGER OF GAME SOFTWARE DEVELOPMENT MARK MACY SHARES HIS THOUGHTS.*

\\\ Designing and developing high end video games is a challenging and competitive endeavor. Developing coin-operated arcade games brings its own unique set of challenges on top of that. Here are a few key concepts an arcade game developer must keep in mind.

INSERT COINS TO PLAY

\\\ When player picks up a new console game, he makes one decision to purchase it, then plays it as much as he likes. With arcade games, the player must choose to pay each time he plays. The average arcade session typically lasts 2–3 minutes, so the challenge to the arcade game developer is to create a gameplay dynamic that is so fun, and so compelling, that players will play again and again. This also means that the experience has to be divided into smaller chunks of play. This might mean a single race in a driving game, or a five-site hunting trek. In the case of an action gun game like TERMINATOR SALVATION, (see Pg. 20) your game lasts as long as your health bar.

GET TO THE GOOD PART

\\\ If a player ever grows bored with your game, she'll move on to the next one. You have to constantly confront the player with a stream of new and exciting challenges, because if the first two minutes don't deliver, she probably won't stick around to see the next two minutes. Every moment must be engaging. You can't make the player hunt around to find the key to unlock the treasure, or present a puzzle that requires trial and error to solve.

ONE SIZE FITS ALL

\\\ When a player starts an arcade game, they may be playing for the very first time, or may be a seasoned veteran at your game. This means you need to find a way to challenge the experts without sacrificing accessibility to the beginner. In a console game, you can put very difficult or challenging gameplay 20 or 30 hours into the game, separating the expert play from the beginning play. But in an arcade game even the beginning must be layered so that there are challenges for the expert, and accessibility for the beginner at the same time. The presentation of the challenge must be obvious for the first time player, but not monotonous for the expert player. This is an extremely difficult equilibrium which often involves tweaking and balancing the gameplay again and again. Your core gameplay dynamic must be easy to learn, yet difficult to master.

THE WHOLE PACKAGE

\\\ In addition to designing and developing a game that appeals to all levels of players, an arcade game team needs to design, procure, and often build the hardware as well. A good game will go completely unnoticed in a bland or boring game cabinet. An eye-catching cabinet will get players to walk up and try the game—then it's up to the game to deliver the goods. The careful planning of the game cabinet, the computer, all the electronic wiring and interfaces inside, and any peripherals such as a plastic gun or steering wheel are critical to a game's success. No matter how fun the game may be, if the gun doesn't work, no one can play it. Electronic or mechanical features such as aiming the plastic gun at the screen, or adding a clip-style reload to the gun can greatly enhance the gameplay experience, but each new technology needs to be designed, built (usually from scratch), often redesigned, and heavily tested in order to assure a reliable experience for the player.

PLAYERS VOTE WITH QUARTERS

\\\ While we wish we could create games just for the fun of it, there is a business side (or dark side) as well. Sometimes the arcade game is owned and operated by the pizza joint or movie theater that houses the game, but more typically, the game is owned by a company that places games in various locations and works out an arrangement with the location to split the money taken in. This means that even if your game is the coolest, if it doesn't earn enough quarters, it will be replaced by that game that tries to pick up stuffed animals with a flimsy claw. The hope is that great games will deliver great earnings. That way everybody wins.

So how can you tune your game to maximize earnings? Dropping in a quarter-eating super boss may seem helpful, but is more likely to result in players feeling helpless or cheated by the game. It might help in the short run, but won't result in lasting earnings month after month. A better approach is to try to find a dynamic where the player feels he "just missed" the goal, that magic balance where he got 29 out of 30 targets and barely missed the last one. You want him to feel challenged, not frustrated. If he's challenged, but seeing his skill progress, he'll come back again and again.

—Mark Macy

RANDOM STRUCTURE

CREATING SEAMLESSLY TILING PERLIN NOISE FOR PROCEDURAL GENERATION

JOSHUA TIPPETTS

Procedural generation of assets, including textures and geometry, has become an integral part of many modern game development processes. Implementing procedural methods as part of the pipeline can free up artistic manpower for the creation of all the other rich details that are required to give the game world life. There has been a lot of research into various methods of procedural generation, and since its invention by Ken Perlin, Perlin noise has become an irreplaceable staple of the field. Coming in a number of variants and flavors, Perlin noise is commonly combined using various fractal methods to create continuously varying noise patterns suitable for use in generating random yet meaningful textures, pieces of heightmap geometry, maps to model the distribution of vegetation, precipitation, and other systems. It can even be used to add levels of jitter or variation to physically modeled systems (see Creator of Worlds, April 2011). In short, Perlin noise is found almost everywhere these days.

This article is not intended to be an in-depth explanation of Perlin noise; that topic has been covered exhaustively in other places. Implementations of both classic and improved Perlin noise have been written in countless languages, and have even been implemented as part of procedural texture shaders on GPUs. Instead, this article will talk about a topic that hasn't received quite as much coverage: making Perlin noise (and other arbitrary noise functions) tile seamlessly. Seamless noise is useful, for example, in creating landscape textures intended to be tiled across a terrain. The borders of the textures must align so that no visible seams or edges appear, and thus the functions that model the texture must themselves be made to tile. Tiling can, of course, be performed by an artist using post-process methods to edit the seams out once the texture has been modeled. But much work can be saved if the seamless tiling is included as part of the procedural process.

The key to performing seamlessly tiled mapping of a noise function is to minimize distortion, discontinuities, blurring, and muddling of the final image. A number of methods are available to tile a function, but most involve some sort of blending or filtering to perform the seamless mapping, which may result in visual artifacts in the final output. >>>

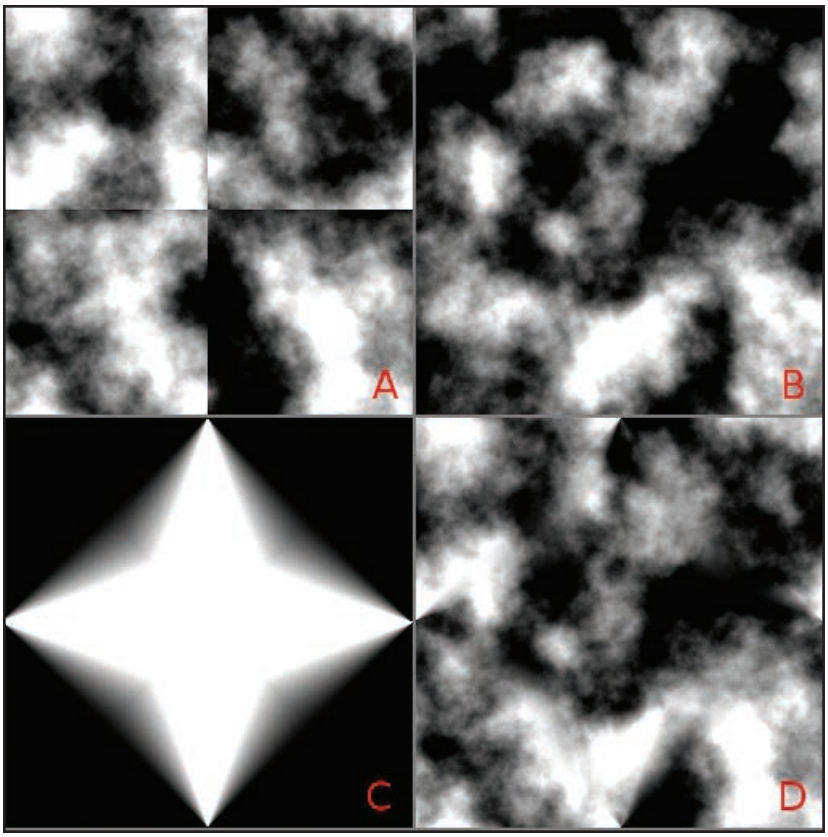


Figure 1: A texture created by blending an offset texture with a non-offset texture using a mask.

I'll discuss two commonly used techniques that utilize blending of different layers. The first involves offsetting a layer of the texture and blending it with another layer using a mask. The second entails blending four regions of the noise function together using a weighted blend. I'll also touch on a final technique which involves a procedure for producing the seamless mapping as a product of a domain transformation applied to a higher-dimensionality variant of the noise function, in order to avoid the artifacts produced by blending.

Offset and Blend

¶ This method of seamless mapping requires the creation of two separate layers of noise from the function, and offsetting one of the layers by half the image dimensions in each direction, wrapping the image around so that the seams appear inside the image while the edges of the images tile perfectly. Then a blend operation is performed, which in effect snips out regions of the non-wrapped image and overlays them onto the wrapped image in such a manner that the interior seams are hidden.

Mask creation is performed so that the artifacts caused by superimposing the regions onto the wrapped image are minimized, otherwise inappropriate visual defects may occur. For certain functions, this method works well. The mask may be created sharp-edged, soft-edged, regular, or irregular as needed. However, many other functions do not work well with this method, because the blending that is performed to hide the edges can cause artifacts to crop up that can disrupt the shape and flow of the texture or create obvious patterns that stand out.

This method tends to work best when performed manually, as a post-process pass performed by an artist, with an eye toward explicitly blending and modifying regions in order to enhance the final result from an artistic standpoint. Figure 1 demonstrates this technique. In Figure 1A, you see a texture mapped from a function and offset by half. Figure 1B shows another texture mapped from the same function, not offset. In 1C there's a sample mask that can be used for blending, which takes into account where the edges are in the offset texture. And finally, 1D shows the final texture after blending. Greater care taken in mask creation can serve to eliminate many of the artifacts you see in 1D, primarily around the apexes of the mask. Unfortunately, this method requires additional development time for the creation of an appropriate mask texture or function.

Weighted Regional Blend

¶ Another commonly used algorithm for computing tiling 2D noise samples four different locations within a region of the function, spaced according to the dimensions of the output texture, then uses weighting or interpolation to blend between these values to obtain the final output value. To visualize this in operation, imagine a 2x2 block of regions sitting within the domain of the function to be mapped, as in the left side of Figure 2.

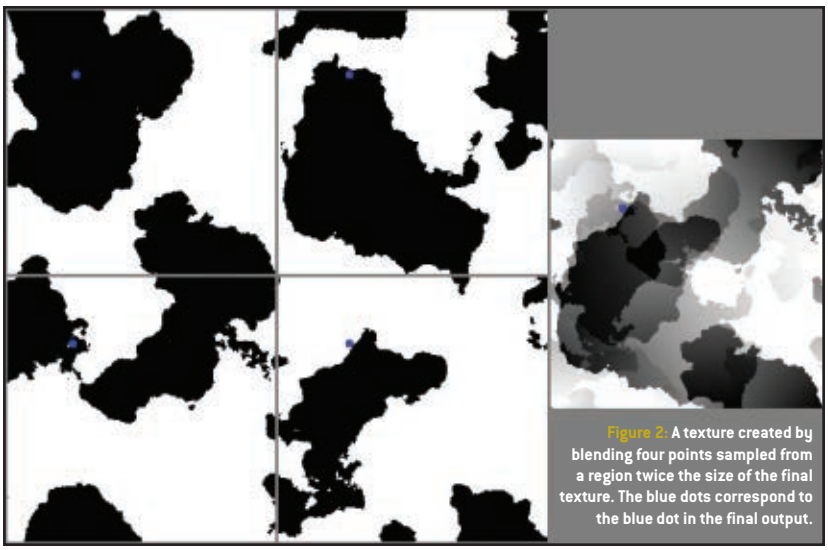


Figure 2: A texture created by blending four points sampled from a region twice the size of the final texture. The blue dots correspond to the blue dot in the final output.

LISTING 1

```
-- f: Noise function (2-dimensional)
-- x,y: Coordinate
-- w,h: Dimensions of output image/texture
function noiseTileBlend(f, x, y, w, h)
   return (f:get(x+w,y+h)*(w-x)*(h-y) +
         f:get(x,y+h)*(x)*(h-y) +
         f:get(x,y)*x*y +
         f:get(x+w,y)*(w-x)*y)/(w*h)
end

-- Function to map a noise function to a buffer
-- f: Noise function (2-dimensional)
-- buf: 2D array data type
function mapFunctionSeamless(f, buf)
   local mw,mh=buf:getWidth(), buf:getHeight()
   local x,y
   for x=0,mw-1,1 do
         for y=0,mh-1,1 do
               buf:set(x,y,noiseTileBlend(f,x,y,mw,mh))
         end
   end
end
```

LISTING 2

```
-- f: Noise function (4-dimensional)
-- x,y: Coordinate
-- w,h: Array dimensions

function noiseTileCircular(f,x,y,w,h)
   -- Calculate parameters of the circles
   local s=x/w*pi*2
   local t=y/h*pi*2

   -- Calculate point on circle on X/Y plane corresponding
   to s, the distance along the image X axis
   local nx=cos(s)
   local ny=sin(s)

   -- Calculate point on circle on Z/W plane corresponding
   to t, the distance along the image Y axis
   local nz=cos(t)
   local nw=sin(t)

   return f:get(nx,ny,nz,nw)
end
```

Each region is the same size as the final output image. A point is sampled within each region: all points correspond relative to one another, and a final mapping is performed by interpolating or blending these values. The interpolating factors are calculated based on the location of the given point relative to the region's boundaries such that, when the location being sampled is nearer the left edge of the image, the point samples from the right-hand regions gain greater weight; however as the sampling draws near the right edge, the point samples from the left-hand regions gain greater weight. Correspondingly, when a point is near the top edge, it gains greater contribution from the bottom point samples, and as it nears the bottom edge it gains greater contributions from the top point samples. When the final blends are performed, the final output will seamlessly tile due to the way the interpolation is performed. The most common algorithm used to perform this blend is shown in Listing 1.

You can see the result of using this method to tile a noise function that has very high contrast on the right side of Figure 2. The blue dots all correspond to the same location in the output image. As you can see, the method results in a large amount of averaging of the texture, when compared to the non-seamless version, due to the blending of four samples. This averaging does not occur in an even manner across the texture; samples toward the center of the image are the most averaged, while samples toward the edges, and especially the corners, are the least averaged. This may have serious side effects, especially for textures that will be tiled many times over, resulting in highly visible repeating patterns. The alteration to the fundamental characteristics of the function is another highly undesirable side effect of this method, and it may render it unsuitable for many applications.

Multi-dimensional Function Mapping

¶ The final technique is an algorithm that maps a N*2-dimensional noise function to a N-dimensional buffer in such a manner that the seamless tiling effect occurs as a result of a mathematical domain transformation, rather than a blend of samples. The easiest way to understand this is to start with the idea of using a two-dimensional noise function to obtain a looping one-dimensional buffer of noise. We could use any of the blending methods above, altered for one-dimensional functions, in order to blend multiple samples and create the tiling noise. Or we could imagine starting with a two-dimensional function and tracing a circle through it at some location, as seen in Figure 3.

We start at Point A and evaluate points spaced evenly around the circumference, one point per location in the one-dimensional buffer. By the time we near the end of the buffer, we are drawing near to Point A once more. Thus, the final buffer will tile seamlessly with itself, without any blending or interpolation being performed. We can use the parametric form of a circle with unit radius in order to perform the domain transformation as below.

```
x=cos(t)
y=sin(t)
```

Here, `t` is in the range of [0,2*pi). We can calculate `t` as a function of the length of the buffer, by dividing the current coordinate by the total buffer length to get a value in the range [0,1), then multiplying by 2*pi. The result gives us the following function:

```
-- f: Noise function (2-dimensional)
-- x: Coordinate
-- w: Length of array
function noiseTile1DCircular(f, x, w)
   local t=x/w*2*pi
   local nx=cos(t)
   local ny=sin(t)
   return f:get(nx,ny)
end
```

To extend the idea to two-dimensional noise for textures, we need to extend the circle-tracing

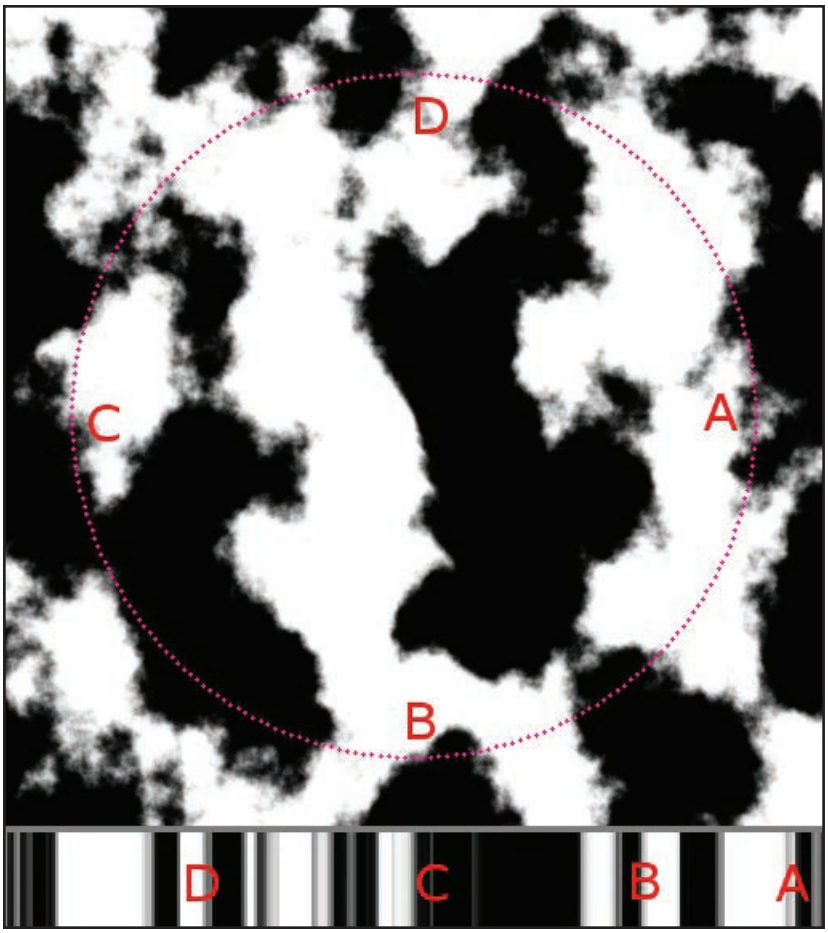


Figure 3: Seamless one-dimensional buffer (bottom) generated by tracing a circle clockwise starting at Point a through a two-dimensional noise function.

metaphor so that each axis of the image has its own conceptual circle in the input function. For each axis of dimensionality in the output image, we need two axes in the input function, forming a plane upon which we trace a circle to correspond to the image axis. In order to obtain a two-dimensional seamless output, then, we need a four-dimensional input function. We assign the X/Y plane in the input function to the circle traced for the X axis in the output image, and the Z/W plane in the input function to the Y axis in the output image, computing circles along these planes and calling the 4D function with the calculated coordinates to obtain the final output. The code to perform this circular mapping for a 2-dimensional image is seen in Listing 2.

Figure 4 shows the outcome of applying the technique to our high-contrast noise function. As you can see, the final result is seamless, yet includes none of the mixing or blurring of the interior regions that the previous blending algorithms produce. While some distortion of the function is introduced as a result of the circular mappings, this distortion is applied evenly across the whole texture and does not result in the occurrence of visual artifacts in the final result, nor does it have a severe impact upon the overall character of the function. The advantage of this method is that it can be applied to functions of arbitrary complexity, as long as that function can be constructed in the proper dimensionality.

The technique can be extended to creating sequences of looping noise textures, each frame of which will tile with themselves, by adding two more dimensions to the input function while tracing another circle upon the plane they form to correspond to the Z axis of the output sequence. This is useful for the creation of seamless textures that must animate in a looping sequence over time.

Of course, this technique comes with the computational overhead of implementing higher-dimensional functions to support it. Still, given that the blending algorithms require the corresponding function to be called two or four times per sample, this additional computational cost is mitigated somewhat by requiring only one call to the function. Going beyond three looping dimensions in a real-time application using classic Perlin noise, with the attendant exponential increase in complexity of the Perlin function, may not be efficient. And even three-dimensional seamlessness, requiring six-dimensional noise, may be unsuitable for real-time applications if the underlying functions are not heavily optimized. There are papers which detail the production of noise functions of extended dimensionality; Perlin's simplex noise variant may be especially suited for higher orders, given the decreased computational complexity of simplex noise versus Perlin's classic gradient noise.

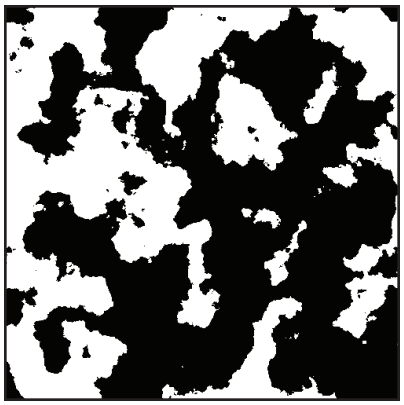


Figure 4: Seamless mapping of a high-contrast noise function using domain transformation of a four-dimensional noise function.

As production pipelines grow ever more complex, procedural generation of assets becomes an increasingly valuable instrument for creating detailed textures and models with a significant savings in time spent. Seamless noise is useful in a wide variety of applications in the field. Given a comprehensive set of noise functions, these methods for generating seamless noise may prove to be powerful tools in your toolbox for creating quality seamless textures and other procedural data.

JOSHUA TIPPETTS *has been an indie game developer for almost 15 years. He currently lives in the mountains of northern Wyoming. You can email him at* **vertexnormal@linuxmail.org**

resources

A Perlin noise math FAQ
http://webstaff.itn.liu.se/ffstegu/TNM022-2005/perlinnoiselinks/perlin-noise-math-faq.html#loop

Perlin's simplex noise desmystified
http://webstaff.itn.liu.se/ffstegu/simplexnoise/simplexnoise.pdf


JAMES RUSSELL

total war ..shogun 2

postmortem

THE TOTAL WAR SERIES BEGAN with the release of SHOGUN: TOTAL WAR back in 2000, spawning a run of PC strategy games that have visited the medieval world, ancient Rome, and more recently, the colonial era and beyond. Along the way, the team has grown from a small unit of 20–30 into a set of teams totalling almost 90 developers. This has partly been a reflection of modern AAA development requirements, and partly born from a desire to reliably release more TOTAL WAR content in a tighter timeframe, and it has presented many challenges. The team grew faster than we were able to develop a company culture, and a set of processes had to be devised in order to manage such a large group efficiently across such complex projects.

These problems were most acute during the development of EMPIRE: TOTAL WAR. Following its completion, we put a new team structure in place, and added processes to smooth production with a much larger team. That the studio was able to develop a full-scale TOTAL WAR game on time in little more than a year whilst maintaining quality in an ever-more discerning review environment (at the time of writing, SHOGUN 2's metacritic score stands at 90 percent) is testament to the success of the team's transformation. But the new approach has not come without its pitfalls. >>>

WHAT WENT RIGHT

1 DOING MORE WITH LESS

/// In many respects, EMPIRE: TOTAL WAR represented the series reaching its geographical apex. We included most of the world, since we were portraying the 18th century expansion of European powers across the oceans, from the Americas to India and beyond. We also put the player immediately in charge of major nations and extended empires. It's 1700, here's the British Empire, now "Go!" While we did try to ease the player into the game, its size and scale could definitely intimidate newcomers. The scope of the game world and feature set also stretched the team's ability to polish and tighten the game.

For SHOGUN 2, we resolved to change all that, and the setting of feudal Japan was a perfect choice for pushing the design goals we wanted to achieve: to do more with less—something we called the Zen of TOTAL WAR. What exactly did this mean to us, practically?

The contained setting (Japan, rather than most of the world) meant we could devote ourselves to a single iconic period by delving much deeper in our portrayal of one unique culture, rather than spreading our efforts more shallowly across many different cultures.

We infused Japanese art and audio styles right across the game, from the UI icons, loading screens, game event pictures and movies, writing style, and the look of the campaign game map itself, down to the parchment style of areas shrouded by the fog of war. This hopefully meant a much greater sense of immersion for the player, and much higher production values, without the work required to portray many different cultures.

From a gameplay perspective, we used fewer, better-integrated features, with a focus on delivering greater gameplay depth than we did in previous TOTAL WAR titles. This meant better counterpoint and meaningful trade-offs in the content itself. There are far fewer unit types in SHOGUN 2 than there were in EMPIRE, but the differences and relationships between them are much clearer. The player can also customise their units to a greater degree and create different combinations of specialities.

The historical situation was also a huge help in that the playable clans started the game in a simple, immediately understandable situation: they had control of just one region. This allowed players to come to grips with their starting assets and shape their own destiny. It made the game feel much less intimidating and much more accessible. Similarly, we could revert to a fog-of-war shroud where the player had to explore outward. The rest of the game world was hidden, unlike the 18th-century setting of EMPIRE, where the location of European capitals could not be hidden at the start, and players could see the entire scope of the setting in all its intimidating glory.

This over-arching "Zen" design goal was right for the setting of feudal Japan, and in the end, it really helped to ease the development effort as well as make the game better as a result. We will need to make sure these lessons are learned, even if the geographic scope of future projects extends outward again.

2 BUILDING EMPOWERED TEAMS AROUND GAME AREAS

/// We used to be a single team. Sure, people had their own areas of focus and their specializations, but there was no formal structure. This was fine for a small team, but during Empire, the team size ballooned, and it was clear by the end of the project that it had become necessary to divide the team into feature-focused sub-units. During the development of NAPOLEON: TOTAL WAR (between EMPIRE and SHOGUN 2), we executed this transition and refined it. We now have three core gameplay teams focused on specific game areas: campaign, battle (dealing with both land and naval), and multiplayer. Each of these has a design lead, a programming lead, and an art lead. Assisting these game area teams are a number of support teams, each with their own leads: UI, graphics, tools, text, and audio.

While they still reported to overall leads, each team had a great deal of genuine autonomy and control over the day-to-day details of how they operated and the decisions they made. Tighter teams focusing on specific feature areas have definitely helped improve the quality of the game across the board, driven by the greater sense of ownership, as well as by structurally locking people to feature sets. Management bandwidth had become a real problem for us, the games and the projects were becoming too big to centrally control, and design leadership was a particular bottleneck.

One added bonus has been the fact that there are many more (area) lead roles to go around. As a result, there is a greater feeling of potential career advancement, compared to a flat, monolithic structure where people feel there is nothing tangible to measure their development against, and where promotion opportunities are not so apparent.

Delegation of responsibility and splitting up the teams like this might sound obvious, but it felt like a big and risky step for us at the time. In hindsight, we should have done it sooner, and going back would be unthinkable.

3 GUNG-HO CHANGES VERSUS RISK-AVERSE CONSERVATISM

/// With the iterative nature of game development, there is always a balance between a willingness to make changes, even late in the project, and an often sensible desire to avoid making late changes which could introduce bugs and unknown balance issues. Making great games that push boundaries always requires change, and the later in the development cycle, the stronger the basis is for making a change, because it's decided on the back of a more complete version. On the other hand, a big game is a very fragile beast that can be upset in unpredictable ways by the slightest tweak. In many ways, making changes (and sometimes even fixing bugs) very close to release is folly for a multi-million-pound project. This is a dilemma game developers face every day.

We believe that we got the balance about right with SHOGUN 2. We are perhaps a bit more gung-ho about making late changes than many

development teams, and we certainly frustrate our programmers as a result! We have a design-led culture, and are always striving to push the game forward to be the best it can be. If we're not all constantly terrified, then we're not trying hard enough! But it's an eternal balancing act, and to walk the tightrope requires systematic questioning: Does the change solve a specific gameplay problem, or is it just an improvement? Is the solution obviously better across many circumstances, or an arguable opinion? Have ripple effects into other gameplay areas been identified and thought through? Does it require additional changes (UI, AI, text, tutorial advice, and so on)? How risky is the change, how much new code is involved, and how deeply integrated is that code with other areas?

Of course, not all the side effects can be predicted; there are unknown unknowns as well as known unknowns! But on the whole, constantly asking these kinds of questions as we made late changes allowed us to balance benefits versus risks fairly successfully.

4 THE BLESSINGS OF A MATURE ENGINE

/// The saying "If it ain't broke, don't fix it" applies to design as much as to code. A successful game represents years of daily problems encountered and solved by smart people. We underestimated this when we re-wrote the entire codebase for EMPIRE: TOTAL WAR, and this meant it took years to reinvent the wheel, even if the wheel was considerably shinier than before. Only then could we get on to making the new game.

With SHOGUN 2, we were going to enhance the core engine as required to support what the new game was going to be. This meant all the effort could go into making a better game instead of redesigning core systems which weren't broken in the first place.

Working on a brand-new (and therefore immature) engine was a key part of what made EMPIRE such a challenging project. Working with a much more mature framework and developing the components in a more modular, contained manner was without doubt a major factor in making SHOGUN 2 a relatively smooth development experience.

5. AI: A TRUE DESIGN-PROGRAMMING COLLABORATION

/// The fans have consistently been calling for stronger AI, and we were determined to deliver it. Writing AI for a campaign and battlefield for a game as complex and intricate as TOTAL WAR is an immense challenge. Here, the nature of SHOGUN 2 helped. On the battlefield, management of melee combat presents fewer degrees of freedom versus the predominantly ranged combat of EMPIRE, where spatial factors dominate, and more complex and open-ended attack-arrangements are possible. In addition, the maturity of the game engine meant the game systems and rules that the AI had to successfully navigate were not quite the fast-moving target they had been.

But perhaps the most essential factor was the much greater design focus on collaborating to improve the AI. Concerted day-to-day design attention was devoted to AI performance in battles (especially sieges) and on the campaign map. Designers were able to pore over detailed logs of AI actions and experiment with many campaign-AI parameters. This required great care as the AI's mind is constantly at war with itself, balancing competing desires that fight for priority. Changing a bias here or there could improve behavior in one situation, but could prove disastrous in other unpredicted contexts. This necessitated dedicated and careful collaboration between AI design and AI programming, as the power given to designers meant the power to mess things up very easily if they didn't understand exactly how each parameter worked.

We were very pleased with the results. The game changed: before, it was a struggle to create a challenge, and then suddenly the AI began running amok, even on easier difficulty levels. Complaints across the team used to be about the AI doing silly things and the game being too easy on hard difficulty settings. Then the complaints changed: people felt the AI didn't give them a chance, was constantly kicking their asses, and was far too difficult for new players. The balancing task was completely reversed, and we had to hold the AI back and cheat in favor of the player.

While the AI is not perfect, and there is plenty more to do, it is much, much better than it has been in the past—a testament to very successful, close, collaborative work between designers and coders.

WHAT WENT WRONG

Doubling the team and halving the time does not equal the same output! We knew this, but we went ahead with it anyway, and gave ourselves a number of headaches in the process.

1 MINIMAL PRE-PRODUCTION

/// One of the great things about creating a tools team was that we actually had a group of people who could make and maintain tools to enable our content creators to work their magic. During EMPIRE's development, tools were maintained by devs who also had to worry about specific game features at the same time. This meant that we really didn't start with the toolset we felt we needed.

The problem for SHOGUN 2 was that with such a tight schedule (a triple-A game in about a year), the tools had to be developed while production was in full swing. This was a recipe for some real difficulties, and early on, the tools group had the thankless task of dealing with portions of the team rendered unproductive by unstable tools.

Despite sometimes feeling like we were fighting tools as they were developed, and messing around with the pipeline while trying to make content, things were going much more smoothly by the end of the project, and in the end, we made major progress developing tools to set us up for an easier time in the future.

2 LATE DESIGN

/// A big game project is an all-consuming endeavor for the team, and this does not diminish as release approaches. Quite the contrary. This makes it a real challenge to peel even senior design attention away from one project that's about to release and onto another that should have started some time ago! We have always found this very hard, and the transition from NAPOLEON to SHOGUN 2 was no exception.

A significant portion of the team (artists especially) were finished with NAPOLEON many months before release, and they needed to start work on SHOGUN 2. The problem was there was very little thorough design. It was frustrating and risky for the artists to work with limited guidance, but we had little appetite for taking design attention away from making NAPOLEON the best it could be during such a critical phase in its development.

This definitely caused some disagreements within the team. Some did valiantly try to start thinking about SHOGUN 2 up front, but there wasn't the capacity to get into detail. The intention was to ensure that any finished artists at least had valid work to do, so a limited design effort fleshed things out at a high level so that work could proceed and no one was blocked. It was an admirable plan, but without a more detailed design, some cul-de-sacs were inevitably entered, and there were quite a few things that had to be retrofitted. Not an ideal outcome.

Partly, this was the result of such a time-constrained dev cycle, but we do need to get better at planning. The transition from NAPOLEON to SHOGUN 2 was probably one of our smoothest, but as teams and costs grow, and time constraints get more severe, we will have to up our game. The delegated team structure does help this, and hopefully we can learn from our mistakes.

3 GET NOOBS PLAYING EARLY!

/// Playtesting can present a challenge in very tight development cycles. Either it's too early and the game is not in a fit state for useful player feedback, or if the game is ready, it can be too late to do very much about issues that are raised by playtesting. We are generally very ambitious with what we try to do with each project, and new features and content come together as a playable whole frighteningly close to release. Getting the game into a state when it is truly playable as early as possible is vital, but this was a huge challenge in such a short project, and getting a true "vertical slice" of a grand-strategy game isn't really feasible until much of the game is actually finished; it's not like a level-based game.

When it was in a decent, playable state, the whole team could finally play the whole game as opposed to everyone just focusing on their own piece of it. Suddenly, a deluge of revelations appeared that might be familiar to any designer who has ever watched new, naïve users get their hands on initial code. The feedback from team outsiders was absolutely invaluable in making improvements. The problem was that this happened so late. Coordinating development better is vital so that each feature becomes visible and playable without delay.

4 BRANCHES, BRANCHES EVERYWHERE

/// Two games became three, which became four! In the beginning, there were battles: a super-realistic RTS game with thousands of soldiers fighting it out on screen at once. The turn-based strategy campaign was initially a wrapper to generate varied battles and give them context. That was the original SHOGUN in 2000. Over time, the campaign game has evolved into a full-fledged strategy game that is, in itself, a rival to series like CIVILIZATION. TOTAL WAR's unique formula involved two full games. When development began on EMPIRE in 2006, we knew that, since it was set during the great age of fighting sail, it was the perfect project to introduce full-scale naval battles: three games in one. Not to be outdone, in SHOGUN 2, we added game number four into the TOTAL WAR formula: an online career campaign in its own persistent environment to give context to players' progress in multiplayer battles. We didn't quite intend it that way, but revolutionizing the multiplayer experience ended up diverging into its own unique game world.

This meant more code branches (yay!). SHOGUN 2 was the first title where we branched the game code in earnest, and we really went to town. We had a main branch and branches for campaign dev, multiplayer, UI, battle dev, audio, and tools. We branched for E3, for alpha, for beta, for the demo, for release, and for patch one. Of course, branching was essential, and meant that development could proceed at pace inside sub-teams' branches without the risk of breaking the build for everyone, and always having a stable main branch to test was invaluable.

But perhaps we took things too far, with too many branches. Integration was a constant headache, especially where the database was not itself branched, meaning game data had to work in all branches at the same time. Fixes took a long time to propagate to all branches. We could have coordinated things and communicated better for sure, and we can only hope that as we rationalize the number of branches and get more used to working with them, we can work smarter and avoid some of the pitfalls and collisions that got in the way during SHOGUN 2's development.

5 COMMUNICATION AND CHANGE CONTROL

/// The feature focus that we gained when splitting people into multiple sub-teams did not come without cost, and the biggest danger was entrenching the separation between game areas, so that while each area might be improved, the game as a whole could drift into a disconnected patchwork. We strove to ensure consistency and coherency across the entire game, and while communication within the small, focused sub-teams was fantastic, communication between teams on a larger scale was not always as good as it should have been.

Problems were perhaps most obvious later in the project, when tweaks and changes were being made at a furious pace to improve and balance the game. Our origins were as a much smaller team, where word of mouth was sufficient to communicate information about changes. We need to be nimble and able to make quick changes without the protracted delays that are possible with bureaucratic with cumbersome processes, but in a large team, change-control and diffusion of information needed to be much better handled. What might seem the simplest feature tweaks can have tentacles across many areas and teams: UI, display, audio, text, VO, advice and tutorials, and more. We need to manage changes with an improved and more formal awareness of dependencies.

THE FIVE RINGS

/// The challenge for the future is to keep the series fresh and new—to allow constant innovation. We are determined not to let the increased team size diminish creativity, and we always look to push the boundaries on every product without fail.

As the team grows, we inevitably need processes to keep things organized. We are determined to maintain our creative, craft-driven culture while getting better at organizing the talent's focus. We need to get the best of both worlds if we are to continue to make fantastic games as each title and the team that makes it grow ever larger.

JAMES RUSSELL *is the lead designer for the TOTAL WAR series of games.*

DO YOU WANT DEEPER INSIGHT INTO YOUR GAME'S PERFORMANCE?

Pre-Instrumented Middleware Makes Performance Analysis Easier

Why are game developers industry-wide adopting Intel® Graphics Performance Analyzers (Intel® GPA) into their development pipeline? Primarily because Intel has collaborated with the middleware community to offer pre-instrumented code that works with Intel® GPA. When you are using one of the products from participating game middleware companies, you automatically see performance metrics and can rapidly identify bottlenecks and task-level efficiency characteristics across various game sub-systems.

Whether you use middleware from SpeedTree, Havok, Geomerics, Allegorithmic, Autodesk Scaleform, Confetti, Umbra, or even an engine like Unity, you can see into all of these game subsystems, which previously would have required hands-on, custom instrumentation work. In addition, you can experiment with changes to optimize your game code, and see results in real time.

Intel is proud to collaborate with these middleware companies to ease developer adoption costs, and continues to work with new companies every day to further broaden the pre-instrumented middleware portfolio.

Learn more about Intel® GPA by visiting www.intel.com/software/gpa.

DOWNLOAD NOW!

Get the latest version of Intel® Graphics Performance Analyzers:

www.intel.com/software/gpa

WHAT DEVELOPERS ARE SAYING ABOUT INTEL® GPA

"To earn our customers' trust we've always had to focus on performance. The integration of our run-time component with the Intel GPA tool provides critical performance data right out of the box. Seeing not only the performance characteristics of SpeedTree, but how it interacts with other middleware packages, is just a fantastic step forward."

CHRIS KING
CEO, IDV INC.

"Using Substance smart textures has tremendous benefit for real-time texture map generation, but one always has to look toward optimizing performance. The integration work we did with the Intel GPA tool and the Substance run-time is certainly going to aid game developers when they do their performance tuning. Now with holistic, integrated performance, data developers can confidently deliver fast, beautifully textured content generated at runtime."

SÈBASTIEN DEGUY
CEO

Geomerics

"The visualization provided by Intel's GPA allows us to clearly understand the temporal relation of tasks. This can be really powerful when seeking performance optimizations. The dependencies and utilization patterns of Enlighten tasks are clear, as are their interactions with other tasks such as rendering or game logic."

CHRIS DORAN
COO

Autodesk Scaleform®

"The first time we used the GPA Platform Analyzer, we saw that our AMP instrumentation was lacking in some code parts and was giving too much information in others. The ability to visualize the events within each game frame is surprisingly effective in pointing you in the right code-optimizing direction."

ALEXIS MANTZARIS
PRINCIPAL ENGINEER

"Now with Umbra 3 being instrumented for Intel's GPA, when developers profile their culling code, they will see, in detail, the various culling tasks that the Umbra runtime is responsible for. Furthermore, this integration enables developers to visualize the clear flow of these culling tasks, whether they come from multiple cameras, AI or other entities."

TEPPO SOININEN
COO

"GPA is outstanding as a profiler, showing the data the way you would expect it. We use it for pretty much all the GPU optimizations we work on, on every piece of hardware we target."

WOLFGANG ENGEL
FOUNDER

"Intel GPA is the best GPU performance tool on PC."

ARAS PRANCKEVIČIUS
LEAD GRAPHICS PROGRAMMER

INTEL® GPA ALSO SUPPORTED IN: havok™

DOWNLOAD NOW! Get the latest version of Intel® Graphics Performance Analyzers: **www.intel.com/software/gpa**

TERMINATOR SALVATION

SCOTT MATOTT

Licensed properties don't come much stronger than Terminator. Generations of teenagers and adults have grown up watching the Terminator films, reading Terminator comic books, and playing Terminator video games. The language and images of the film series don't just resonate with audiences worldwide, they're part of our cultural lexicon.

Phrases like "I'll be back" are referenced and used in everyday conversations. For the team at Play Mechanix and our partners at Raw Thrills, the connection to the Terminator license is doubly strong. Play Mechanix was founded by George Petro, lead programmer and designer on Midway's "TERMINATOR 2: JUDGEMENT DAY," an arcade classic from 1991 which more than a few of us harbor fond memories of pumping quarters into as kids. Living up to this legacy was both an inspiration and a challenge for our team when making TERMINATOR SALVATION — THE ARCADE GAME.

Making a game worthy of the Terminator name was a daunting task, one which we met head on, pushing ourselves on all fronts to make the most kick ass action gun game we possibly could. We're not just proud of the final result, but we're also proud of how we got there: by setting ambitious goals for ourselves and accepting nothing less than surpassing them.

WHAT WENT RIGHT

1 SWEET-LOOKING CABINET.

/// In addition to all the famous and talented humans that have been associated with the Terminator license over the years, the Terminator robot itself has become an iconic figure, instantly recognizable to audiences the world over. From the moment the license was secured, we knew the best way to evoke that icon and draw people to our game was to put a life-sized bust of the Terminator endoskeleton right on top of the cabinet. Even if there was no game to play and the rest of the cabinet was made of unpainted plywood, we knew people would stop and investigate when they saw a giant Terminator head staring out at them.

To create such a head, we turned to a material we'd had plenty of experience with over the years: vacuum formed plastic. Vacuum formed plastics are created by using vacuum suction to pull a flat sheet of plastic onto a hot aluminum mold. This stretches and forms the plastic into the shape of the mold. It's a terrific way to add custom shapes and contours to an arcade cabinet.

Still, we had never attempted anything as complex as the Terminator bust before. In some ways, the high-profile status of the Terminator head made things harder, because people know what a Terminator robot looks like, and if our bust didn't hit the mark, they'd notice right away. We found that getting the general shape of the Terminator right came pretty quickly, but the hard part proved to be getting the printed graphics to distort correctly onto the stretched plastic. Since the graphics are printed onto the flat plastic sheet before it is vacuum formed, they distort as the plastic stretches to fit the mold. The


ON
postmortem

printed graphics have to be distorted beforehand—stretched in some places and shrunk in others—in order to compensate for this. Computer modeling software can aid in this process, but we found that it was no substitute for trial and error with human eyes. In one early trial, our mean looking Terminator robot came out looking more like Bugs Bunny. It was pretty humorous, but definitely not what we were going for!

Of course, we didn't stop with just putting a Terminator head on the top of the cabinet. We gave the rest of the vacuum formed plastics on the cabinet a slick black finish. We added clear etched plastic with red edge lighting around each gun holster. We tilted the gun holsters up and out to better show off the guns themselves. We covered the rest of the cabinet in artwork from the film. We even went so far as the make the red LED eyes on the Terminator bust blink in unison with the thumping of the theme music that plays during the game's attract mode. All this was done to make sure the Terminator theme really carried throughout the entire cabinet. The end result speaks for itself, as the cabinet is an absolute head turner that instantly connects players to the license.

This unification was made more challenging by the fact that there wasn't just one design to consider. In fact, we actually developed three different flavors of the TERMINATOR cabinet: a 42" Deluxe cabinet, a 32" Mounted Gun cabinet, and a projector-based 100" Super Deluxe cabinet. The 42" model was our main platform from the start, the 32" version came about after release to accommodate smaller spaces, and the Super Deluxe cabinet was made for locations with a lot of space that really wanted to show off the title. Of course, each cabinet design presented its own set of engineering challenges. The projector for the Super Deluxe cabinet, for example, had to be carefully tuned and tested to make sure it could survive being on for days at a time. Ultimately though, having all three of these designs available was a real boon to the project, as it allowed us to offer a version of the game to suit any buyer.

2 THE WAY OF THE GUN.

/// One of the great things about making arcade games is that you get to make use of custom hardware to create a truly unique experience for the player. With TERMINATOR, we really wanted to take advantage of this by creating a new gun controller that would make the player feel like a regular badass just by holding it. To accomplish this, we modeled our gun controller after the real M4 assault rifle featured prominently in the film, and we gave it a sliding reload switch underneath the magazine clip. The player reloads by slapping this switch, making something like the motion of actually inserting a new magazine clip into a real rifle.

Believe it or not, this simple, intuitive motion was one of our greatest points of concern when designing the gun. Nearly all prior action gun games used some quirky reloading scheme, such as shooting off-screen or stepping on a pedal. For TERMINATOR we were adamant that we didn't want to go down this route; we wanted a reload mechanic that added to the player's immersion, taking the player further into the experience rather than distracting them from it. We wanted a mechanic that would make the player feel more like they were firing the real deal rather than just reminding them that they were playing a video game. We knew that putting the reload switch under the magazine clip was the best way to achieve this, but we worried that players, trained as they were by the reloading schemes of the past, wouldn't get it. If there's one thing you don't want to do in an arcade game, it's ask players to unlearn something they've spent years learning to do. To avoid this we compromised; the game accepts both the reload switch and shooting off-screen to reload. This way, players trained to shoot off-screen can continue to play that way if they wish, while everyone else can enjoy smacking the magazine clip to reload.

Another fun touch we added to the gun was a strong solenoid knocker which fires in sync with the game to give it a somewhat realistic kick when it's fired. Tuning the strength of this knocker was definitely a challenge. In an early revision, the knocker was tuned very high, giving the gun a satisfying, powerful firing kick. Unfortunately, it also gave the player a serious shoulder ache after a few minutes of play! With some fiddling, we got the knocker tuned to a mix of being strong enough to feel powerful, but weak enough to feel comfortable.

Of course, a cool design wasn't our only consideration when making the gun mold. Early in the design process, we polled game owners and operators about their experience with competitors' gun games. Their universal response: gun simulators are cool, but they break too frequently and cost too much to repair. Hence we set a development goal to make the construction of the gun as reliable as possible. To accomplish this we partnered with Suzo Happ, the main parts supplier to the arcade industry, which was able to implement the best practices based on experience manufacturing hundreds of thousands of guns over the past 25 years. Suzo Happ life tested the gun for an estimated 1,000,000 operations, which simulated approximately five years of continuous usage, to look for stress or failure points in the gun shell or the mechanical parts. We also did a fair bit of our own testing, including a four-foot "drop test" onto hard concrete. Thanks to all this testing, we were able to produce a gun that worked well not just for the player but for the owner of the cabinet as well.

3 TOUGH DECISION.

/// The licensing deal for TERMINATOR SALVATION was finalized in April of 2008, with the movie to be released in May 2009. Allowing for manufacturing time, this gave us roughly nine months to produce the completed title. This was not a generous amount of time. But if we took any longer, we'd miss out on the opportunity to synchronize with the multi-million-dollar marketing campaign for the movie, and risk arriving in arcades after consumers had forgotten all about John Connor and his epic struggle against the Terminator robots. So we drafted a schedule for making the game in nine months and got to work.

Six months later, we had the core of a game ready. Basic character models and environments were completed, and a first pass of game dispatch had been roughed in. The trouble was, no one was particularly happy with anything. The gameplay was, to put it kindly, monotonous. The art assets were serviceable but lacked pop and polish. Additionally, by that point it was more than clear that there was no way our custom gun and cabinet designs would be ready for manufacture by the movie's launch. Still, with three months to go, we could have cranked out the rest of the environments and dispatch, settled for an off-the-shelf gun and a bare bones cabinet, and we'd be done. We would have had a game—maybe not the greatest game, but something—and it would have been ready in time for the movie's May 21st

DEVELOPER Play Mechanix
MANUFACTURER Raw Thrills
DISTRIBUTOR Betson Enterprise
NUMBER OF DEVELOPERS 40
LENGTH OF DEVELOPMENT 22 months
RELEASE DATE February 25, 2010
BUDGET $4 million
LINES OF CODE 591,797
SOFTWARE
Microsoft Visual Studio
GNU gcc, gdb, gperf
valgrind
PhysX
Maya
SolidWorks
Crazy Bump
Mudbox
Adobe Photoshop
Adobe After Effects
GIMP
Mantis Bug Tracker
Subversion
WinMerge
SQLyog
PLATFORM Arcade
T600S TERMINATED 999,999+

premiere. Beyond the licensing advantages, there was an additional consideration driving us toward completing the project as originally scheduled: both the programming and art teams had been staffed up to take on the larger-than-expected workload of producing a modern "next-gen" title. The result was that the project was already over budget, and even if it arrived on schedule, it was set to be the most expensive game Play Mechanix had ever produced. Couple that with the global economic downturn that started in 2008, and there was considerable financial pressure to complete the project as quickly as possible.

Still, for the team at Play Mechanix, the decision was an easy one. The goal of meeting the movie's release was abandoned, and the entire schedule was extended by six months. The rationale was simple: without a great game, having a great license and great marketing wouldn't mean anything. We also knew from years of industry experience that, although the pain of missing the movie's release and cost of another six months development could be overcome, the stink of releasing a turd would linger forever.

In the end, the decision proved more than justified. This was true even though the game ended up taking a full 22 months to complete and cost over $4 million to develop, more than double what we'd originally planned to spend. The extra time was used to redefine and re-tune our gameplay. We quickened the pace, upped the cinematic elements, and added more variety to the enemy types and dispatch scenarios. We also took time to overhaul the game's visuals, bringing everything—lighting, character models, environments, particles, and so forth—up to a higher standard. Finally, the extra time allowed the new gun and cabinet designs to be completed.

If we hadn't taken the extra time to get it right, we'd have been releasing a mediocre title in a weak economic climate. Instead we were able to release a title so strong that it re-ignited the market for arcade action gun games. It also opened up overseas sales opportunities for us in markets like Japan, China, and Russia that we hadn't even thought possible before.

4 GUI SCRIPTING TOOL.

/// Going into the project we knew we'd need a new scripting tool. All of our existing tools had been created ad-hoc in spare programmer cycles, and were either too narrowly focused or too onerous to use (or both) to be useful for TERMINATOR. We considered using a high-level scripting language such as LUA or Python, but we didn't want to limit the scripting process to only programmers. Instead, we wanted a tool that would set the bar for creating dispatch as low as possible, so that anyone at the company could contribute to the creative process, even those with little or no programming background.

To that end, we developed an in-house system which combines a basic scripting language and a simple trigger-and-event system with a GUI front end. The scripting language is easily extensible, but deliberately somewhat simplistic. Typical commands include things like "Travel along this path," "Begin attack cycle," and "Wait for trigger." Only simple flow control (branching and looping) is implemented, and no generic variable interface is provided. The goal is for the game objects' logic and AI to be written in C/C++, while the scripting language provides the gameplay and dispatch patterns for those objects to execute. Again, the motivation for this was to keep the dispatch instruction set as simple as possible so that even someone with no programming background could use it. Keeping with this thought, the trigger-and-event system is similarly basic, providing little more than a method for scripts to set and respond to named global triggers. For example, the script controlling one T800 actor might wait for the "T800_1_Attack" trigger to begin attacking the player and then signal another T800 to begin attacking by setting the "T800_2_Attack" trigger when it's done. Finally, the GUI front end provides a point-and-click way to author scripts and assign them to game objects. The GUI runs in-engine, which allows scripts and scenes to be quickly tested and iterated upon.

Ultimately, this combination proved very potent. The dispatch team quickly learned to combine the scripting language's simplistic commands and triggers to produce much more complex results. Moreover, by keeping the scripting tool simple and easy to use we opened up the design process, allowing anyone at the company to contribute dispatch ideas and scenarios. The net result was the dynamic, layered scenes and varied dispatch that give TERMINATOR its cinematic and visceral punch.

5 TUNING THE "RED BULLET."

/// One of the toughest parts of arcade game design is that you have an excruciatingly short time in which to hook a player. Buyer's remorse might keep a player spelunking away at a $60 home title for hours to get to the good stuff, but a player who has only invested $1 into an arcade title demands some serious immediate gratification before they'll put in another. At Play Mechanix, our rule of thumb is that you have about 10 to 20 seconds to grab a player's attention and start showing them a good time. After that, the eyes start to glaze over and the player is lost to you. They might play their dollar out, but they'll never put another coin in again.

This is a challenge we ran smack into when developing the gameplay for TERMINATOR. From the outset, we knew we wanted the game to follow in the grand "red bullet" tradition of arcade action gun games past, such as TIME CRISIS and ACTION HERO. In this style of game, the screen is full of bad guys to shoot, but only the specially highlighted enemy shoots the magic red bullet which actually hurts the player. Everyone else fires white bullets, which shoot harmlessly past the player. Core gameplay consists of picking the highlighted enemy out of the crowd and shooting him before he can fire his red bullet.

On top of this, we had the idea that the Terminator robot in the game should feel like the same unstoppable killing machine people are familiar with from the movie franchise. Instead of scoring one-hit kills, we figured the player could have fun slowly shooting the Terminator apart bit by bit, first shooting off an arm, then a leg, then maybe a section of his chest, and so on.

We put both of these ideas together and envisioned a game where a handful of Terminators on-screen took turns firing red bullets at the player who slowly chiseled them down to bits. The core skill of the game would be in chasing the highlight from Terminator to Terminator, aborting the red bullets. We quickly learned that this vision would need some serious tweaking to pass the ten-second test.

First, it became evident pretty quickly that an indestructible robot made for an insufferable bore when it came to gameplay. Playtesters did enjoy damaging the Terminators using the destructible character model system we'd built, but even though they reported having a good time, we could see them tune out as they stared at one scene for

20–30, seconds pummeling away at the same robots. To amend this, we reduced the number of hit points for the Terminator robots to something that, while still making them as strong as befits the license, was far down from our original vision. We also sprinkled more gun power-ups and grenades throughout the game to help the players clear out the Terminators faster.

Second, while our original vision for the game's red bullet gameplay did turn out to be fun to play, it had two critical failings. Even though it is a long-standing convention of the genre, we found that the red bullet idea took too long for players to grasp. Players would be hit four, five, even as much as ten times before they'd catch on to what was going on with the highlights and the red bullets. This was exacerbated by the fact that in our original scheme every time a Terminator would fire a red bullet, the highlight would move to a different Terminator. This made it very hard for players to get the connection between the highlight and the red bullet. Secondly, the system asked the players to do something they intuitively didn't want to do, which was shift targets from one Terminator to another before they were done killing the first one. Because the Terminator was supposed to take a ton of hits to kill, the idea was for the moving red bullet highlight to alleviate the tedium of pummeling on one enemy for so long, but in the end, it just fought with the player's better pummeling instincts.

Once players understood what was going on, they could enjoy the anticipation of the red bullet bouncing around and the skill of knowing when to shift fire from one Terminator to another, but for most players this understanding simply came too late. By the time they'd figured out what to do, they'd have already spent their first critical 10 seconds getting pasted and not understanding why. It was a recipe for people to walk away. We altered our red bullet scheme several times to combat this, finally settling on a setup where for most enemy types, once an enemy was highlighted, it would continue firing red bullets until it was killed. We also sped up the red bullet enemy's rate of fire (turning down, of course, the damage done by each bullet). The result of this was to make it painfully obvious to the player which enemy they needed to shoot. The skill test of chasing the red bullet was lost, reduced to merely identifying the highlighted enemy, but players could quickly comprehend what they were supposed to do, and thus start having a good time before their 10 seconds were up.

As it turns out, both of these deviations from our original vision were for the better. Reducing the Terminator's hit points not only sped up the gameplay but also allowed the level designers to pack more of them into each scene, giving the game an epic sense of scale that players really enjoyed. Simplifying our red bullet scheme actually allowed the level designers to be more controlled in their use of the red bullet, assigning it to specific enemies at specific times for maximum effect. This led to scenarios that were more deliberate and cinematic, attributes that grabbed the players' attention more quickly and fully than our original mechanic ever could have.

In the end, our experience tuning TERMINATOR's gameplay to meet the ten-second rule is a classic case of working within a constraint that at first seems onerous but ultimately drives you towards a better game.

DEDICATION

Play Mechanix lead animator, Ryoichi (Roy) Yanase, passed away on February 3, 2011, at the age of 39. Game players all around the world have been entertained by Roy's unique talent and style while playing such games as BIG BUCK HUNTER PRO, BIG BUCK SAFARI, JOHNNY NERO ACTION HERO, ALIENS EXTERMINATION, and TERMINATOR SALVATION. Roy's life was cut short much too early, and he will be missed by all those that knew him and those that were exposed to his art.

WHAT WENT WRONG

1 REALITY BITES.

/// Anyone who's ever been through it knows that the development cycle of a game can often feel like you're living Murphy's Law day in and day out. Everything that can go wrong is going wrong, all of the time. Arcade games, being a physical, real-world product, bring an extra dimension to this experience. During development, it's nearly impossible to have all the pieces of the puzzle (software, electronics, cabinet) in exactly the right state at all times. More often than not, you have to make do with prototype hardware or beta software until the very end of the project. Then after release, there's always some bizarre real-world circumstance that comes up that you couldn't have foreseen. TERMINATOR certainly had its share of issues.

A great example of working with incomplete hardware came with the gun molds. The iterative nature of the design process coupled with long turnaround times to build test models meant that the rest of the game was ready for field testing well before final gun molds were available. For our first round of field testing, which consisted of just a single cabinet at one location, we ended up sending the game out with two of our four prototype gun models. Not only were we entrusting the public with half the world's stock of our new gun model, the prototype models were made with a fragile clear plastic which had to be painted over with automotive paint in order to match the rest of the cabinet. The paint had a harmless but distinct odor that refused to dissipate, even after we held up testing for several days while waiting for it to clear up.

Later when we moved onto wider field testing, we needed more guns, but the final molds were still months away. Needing a quick solution, we opted to have castings made from one of our prototypes. From these, we were able to make around 30 working guns for our extended field testing and for showing the game to operators at the various industry trade shows. Unfortunately, the screw fittings from this rough casting were not correct, meaning the test guns had a bad habit of splitting apart after a few days of use. Not having any better options, we ended up using zip ties to hold the split guns together. In summary, during the entire time we were field testing our game, gathering critical earnings data, and showing it to operators to create pre-sales buzz, we were using guns that either stunk like fresh auto paint or were held together by zip ties. Fortunately for us, players and operators were both able to look past these "features" and the game earned and sold well anyway.

One of the odder problems turned up only after the game started appearing in some of Japan's busiest arcades. The gun in TERMINATOR uses a camera and light array setup similar to, though much more sophisticated (and expensive!), than the one in the Wii Remote. An IR light array sits inside the cabinet marquee and a camera mounted in the gun interprets where the player is aiming based on which lights it can see and where. As it turned out, in some of these densely packed Japanese arcades, the cabinet lights on the games opposite the TERMINATOR games would literally flash so brightly and so frequently that their reflections coming back off the glass of the TERMINATOR monitor would confuse the gun camera into thinking they were part of the light array. This would cause shots to go wild as the gun code tried to account for all these extra lights! It took some pretty serious retooling of our gun's light recognition algorithms to get them to deal with these phantom lights.

In our experience, these types of problems are ultimately par for the course when developing arcade games, but they're still never anything you'd describe as going right.

2 CREATE FIRST, LOCALIZE LATER.

/// Given how much work needed to be done and how far we were pushing ourselves in every other direction, it might seem understandable that we readily fell into the mindset of "get the game done first, and worry about translating it later," but this approach came back to haunt us at the end of the project. Based on our market research and previous experience, we didn't expect to sell too many games outside of the U.S. (and other countries where an English game with a localized manual is sufficient). Anything beyond that would come only after the game had proven itself in the U.S., and then again only after lengthy contract negotiations and so on, giving us plenty of time to go back and kludge in some kind of multilingual support.

Fortunately for the company, but unfortunately for the programming staff, things didn't work out that way. Distributors in Japan were so impressed with early builds of the game that they signed on to receive the second batch of games leaving the factory floor. This gave us about a month after the

game's initial release to have it fully translated into Japanese. This shouldn't have been too huge a task, except that our engine had no concept at all of "translation." All on-screen text was either created via ASCII strings embedded in the codebase or buried in the texture maps for various in-game plaques. So we were caught with just a month to design a localization system, implement it, and convert all the in-game text to using it. Ultimately, we got the job done and with some time to spare, but not without some serious He-Manning on the part of two of our programmers, and we definitely learned our lesson about assuming that localization can wait for another day. The flat truth is that, in today's global marketplace, there really isn't a place for treating localization as an afterthought; going forward, we've committed ourselves to developing and using a robust localization system on all of our projects.

3 FROM ENGINE 1.5 TO 2.0.

/// Since the original schedule was so tight, we had planned to make TERMINATOR using a minimally upgraded version of the engine that powered our popular hunting game BIG BUCK HUNTER PRO. While that engine was older and featured only OpenGL fixed-pipeline functionality support, everyone on the team was very comfortable using it, and we somewhat naively thought we could get by with just a few quick, surgical changes, such as adding limited shader support and tweaking the animation system slightly. Over time, it became evident that much more extensive engine work would be required to meet the visual and gameplay goals of the project. One by one, pieces of the engine were re-written or replaced, some systems more than once. The engine's render loop, for example, was re-written not once but twice, first to add support for multi-pass rendering and again later to better optimize object culling and sorting in order to meet our performance targets.

The end result was that we spent a lot more time retooling the engine than we would have if we had planned to do a major rewrite from the start. We were still dealing with a lot of the quirks and limitations of the BBH engine right through the end of the project.

4 UNDERESTIMATING ASSET CREATION TIME.

/// Again, driven by our initial nine-month schedule, we started the project with the idea of utilizing new visual tech sparingly, adding dynamic lighting and normal maps to the main character models but sticking to baked-in lighting for environments and secondary characters. The hope was to add some sizzle to the game's look, while at the same time, minimizing development time by sticking mainly to tech we were already familiar with.

It quickly became obvious that this wouldn't be enough to give the game the visual impact it needed. A dynamically lit, normal-mapped Terminator simply looked out of place in a vert-lit world. We realized that we'd need to dynamically light and normal map almost everything, and on top of that we'd want bloom, glow, and a host of other post effects in order to help connect and composite everything on the screen into one unified image.

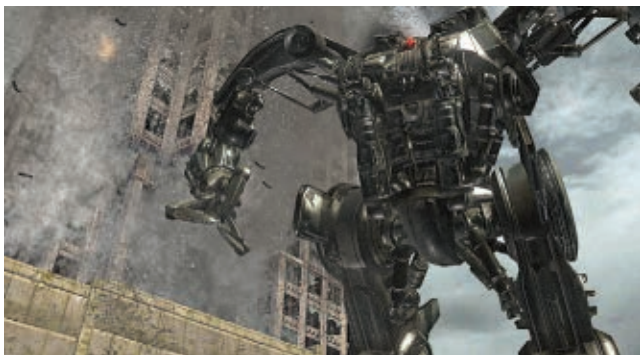
Of course, as great as all this new tech was, it'd be a serious understatement to say that our initial schedule was unprepared for the tenfold increase in asset complexity and creation time required. Aside from the sheer volume of work, the task of maintaining asset continuity proved to be a much larger challenge than we'd initially expected. Simply matching in general style and look was no longer sufficient, as each new layer of tech added a new aspect that the art assets had to match.

Normal and specular map resolutions had to be balanced across the entire scene, so that objects with sharp lighting details didn't appear directly adjacent to objects with softer features. Specular maps, especially those for character models which appear in many different scenes, had to be tuned to match relative intensities across all objects and to look correct in a wide variety of lighting conditions. All of these were new dimensions for us to work in, and they required much tighter coordination between all members of the art team than our previous titles had. In the end, and thanks largely to our decision to extend the project by six months, the look of the game surpassed even our own internal goals and expectations; however, we learned a hard lesson about what it takes to produce a modern-looking title, and why our original production schedules were way off target. A silver lining to all this was that the experience really helped grow and forge our art department into a stronger team, and taught them a whole new set of skills that they've been able to apply to even greater effect on subsequent projects.

5 RELEASE MANIA.

/// Whoever said "when it rains it pours" must have been around the Play Mechanix and Raw Thrills studios during the release window for TERMINATOR. When we put TERMINATOR out, we also had three other new titles preparing for release, plus a slew of updates ready to roll out for our existing BIG BUCK HUNTER PRO ONLINE and BIG BUCK SAFARI ONLINE products. Adding to this math was the fact that, from a production and testing standpoint, we treat each different cabinet as a separate release, so TERMINATOR represented three different releases on its own. Similarly, each title is tested and released as a separate build for each language and region it's sold in. Overall, we've found this level of thoroughness is the only way to ferret out cabinet or locale-specific issues and make sure they don't make it into the final game. Still, it's an approach that certainly makes for a busy QA department. Moreover, the fun doesn't stop with just the first release of a title.

We are always listening to customer feedback about our games, and we try to respond to it as quickly as possible. This means adding new features and bug fixes, sometimes weeks, sometimes years after a game's launch. A good example of this is the optional aiming reticule and tracer bullet effects in TERMINATOR, both of which were added within a few weeks of our first release based on feedback from distributors in Japan. In all, TERMINATOR saw nine such releases in the first five months after initial release. Managing so many releases across so many different titles in such a short span of time was a Herculean task of the first order for both our software and QA departments. Somehow we survived the storm, but not without learning a few lessons and making some changes along the way.



First, we established clearer, more formal lines of communication between QA and software development, so that everyone is always on the same page about what titles are being released and when. Secondly, we've shifted to a more advanced revision control system in order to better track and manage software changes on new projects. One thing we were not willing to commit to was scheduling fewer releases. Despite the difficulty that managing so many releases presents, we feel strongly that having multiple titles in simultaneous development, testing them rigorously, and being quick to respond to customer feedback are all critical to our continued success.

SALVATION

/// Making TERMINATOR SALVATION - THE ARCADE GAME was an amazing experience for the whole team. While there was plenty of hard work to go around, getting the chance to add our own small piece to the Terminator legacy was a real treat. Moreover, pushing ourselves to create a game worthy of that legacy has only left us more excited about the possibilities for our future projects.

SCOTT MATOTT *is a game developer at Play Mechanix. Email him at smatott@playmechanix.com. Contributions to this article were also made by Mark Ritchie, Nick Mangiaracina, Mark Macy, Kevin Uskali, Matt Davis, and Bob Yoest.*

TOKYO GAME SHOW 2011

GAME - Dancing Your Heart

9/15 [THU] 16 [FRI] BUSINESS DAYS

17 [SAT] 18 [SUN] PUBLIC DAYS

Venue: Makuhari Messe (Chiba, Japan)

■Business Days: 10:00-17:00, September 15(Thu) & 16(Fri), 2011
Admission Fees: Business Days - Ticket sold in advance: 5,000 yen
Online Ticket Service is available at our official website ※Admission on Business Days is restricted to those in the game industry and press only.

■Public Days: 10:00-17:00, September 17(Sat) & 18(Sun), 2011
Public Days -Advance Ticket/1,000 yen per day, On-the-day Ticket 1,200 yen per day. Online Ticket Service is available at our official website.
Children (elementary school age and under): free of charge ※ Please note that Public Day ticket is valid once for one day (September 17 or 18) only.

■Official Tour:
Official Tour Packages departing from Chicago, Los Angeles, New York and San Francisco are available.
Please access to our official website for more information.

TOKYO GAME SHOW 2011
Organizer: Computer Entertainment Supplier's Association (CESA)
Co-Organizer: Nikkei Business Publications, Inc.
Supporter: Ministry of Economy, Trade and Industry

Official Website:
http://tgs.cesa.or.jp/english/

CESA
日経BP社
CoFesta 2011

GDC ONLINE
{PREVIEW GUIDE}

FRANK CIFALDI

IT'S AN INTERESTING TIME FOR CONNECTED GAMES. TITLES RANGING FROM CITY OF HEROES TO TEAM FORTRESS 2 ARE SUDDENLY EMBRACING THE FREE-TO-PLAY MODEL, "TRADITIONAL" PUBLISHERS LIKE CAPCOM ARE FORECASTING DRAMATIC WORLDWIDE GROWTH FOR ONLINE GAMES BY 2015 OR SO, AND WE'VE COME TO FIND OUT THAT ZYNGA IS APPARENTLY WORTH ONE BILLION DOLLARS. EVEN IF YOU'RE NOT CURRENTLY EMBRACING CONNECTED GAMES, THERE'S OBVIOUSLY A LOT TO LEARN FROM THOSE WHO ARE.

The Game Developers Conference Online may be a smaller event than GDC proper, but it is by no means "GDC Jr." This annual spin-off conference is focused much more on online games, be they casual, MMOs, virtual worlds, or games on social networks.

The event, which is held in "weird" Austin, attracts more than 3,000 attendees to its over 120 sessions, with tracks catered to professionals specializing in business and marketing, game design, customer experience, production, and programming.

Among the many panels scheduled, this year's show will see former WORLD OF WARCRAFT designer Tom Caldwell discussing the causes of poor game design, CCP's Valerie Massey sharing how she prepares for and deals with audience crises, Zynga's Rober Zubek giving an engineering postmortem on the immensely popular CITYVILLE, and PopCap's Giordano Bruno Contestabile helping to make the difficult iOS vs. Android decision just a little bit easier.

The Game Developers Choice Online Awards at GDC Online.

And no, this isn't a "social only" club, either. Other speakers this year include folks from Eidos Montreal (DEUS EX: HUMAN REVOLUTION), BioWare Austin (THE OLD REPUBLIC), Volition (SAINTS ROW), and more.

Due to demand, this year's show adds a new Customer Experience Track, which finally provides a dedicated series of panels for what is quickly becoming one of the most important sectors of the online games business. Topics will include building and managing online communities, policing user-generated content, anti-fraud tactics, and more.

As always, the show will host dedicated summits in addition to the main conference. The Game Narrative Summit will focus on best practices for interactive storytelling, the Virtual Items Summit will explore design and monetization techniques for virtual goods, and the Smartphone & Tablet Games Summit will discuss the future of gaming on both established and emerging portable platforms.

To prepare you for the show, which takes place October 10–13, we've highlighted a handful of staff picks from the sessions that have been announced so far. As always, head over to GDCOnline.com for the most up-to-date information on what's happening at the show.

Building the Story-driven Experience of DEUS EX: HUMAN REVOLUTION

MARY DE MARLE — EIDOS MONTREAL

Like the legendary original game in the series, DEUS EX: HUMAN REVOLUTION manages to tell a pointed, compelling story while giving its players the creative freedom to shape their own destinies. In this hour-long lecture at the Game Narrative Summit, lead writer and narrative designer Mary De Marle will share her team's approach to story-based game development, revealing how they managed to keep a tight grasp on the game's various branching paths.

Designers are Human Too— Causes of Poor Design Decisions

TOM CADWELL — RIOT GAMES

The road to bad games is paved in good intentions. In this design lecture, former WORLD OF WARCRAFT designer Tom Cadwell (now at Riot Games) will explore the single most common flaw amongst game designers: natural human tendencies. Cadwell will share some of the methods he's developed for identifying bad decisions before they become costly, and how to combat what he calls "these all-too-common forces of evil."

DEFCON: A Basic Guide to Crisis Management

VALERIE MASSEY — CCP GAMES

It's hard to imagine a better test case for online community crises than EVE ONLINE. The game is designed at its core to be about as open to freedom and emergent gameplay as is feasible for a major MMO, and as a result players are gambling with their real-world dollars. CCP's PR and community director Val Massey will share her experiences, and offer tips and tools to stay calm and get things back to normal when inevitable community crises happen.

Engineering CITYVILLE

ROBERT ZUBEK — ZYNGA

In this talk aimed at social game programmers, Zynga's principal software engineer Dr. Robert Zubek will discuss the server-side engineering techniques that allowed his team to rapidly iterate and grow CITYVILLE into the most popular game on Facebook. The talk will also cater to those who have little experience in the social space, and will expose some of the challenges that await the green Facebook game programmer.

How Metrics Are Ruining Your Game; Common Pitfalls and Uncommon Solutions

IAN WANG — ZYNGA

Despite the title of this talk, Zynga product manager Ian Wang admits that good metrics can complement the game design

process. The problem comes when a designer looks at metrics as scripture, rather than a tool. This talk will give a practical view of how metrics should be used, and how to avoid having your creative design process hindered by cold, hard, unforgiving data.

Live Game Disasters: How to Prepare for the Worst Before It Happens

CRYSTIN COX — NEXON AMERICA

As anyone who has worked on an online game knows, actually launching your game is the beginning—not the end—of your troubles. MAPLESTORY producer Crystin Cox promises to give several real world examples of the horrors that occur after the servers are set live, including hackers, forum trolls, gold farmers, and network outages. The goal of the talk is to help producers overcome little disasters so that their teams can focus on the next update, instead of putting out fires.

Rapid MMO Content Iteration and Validation with Spatial Analysis in STAR WARS: THE OLD REPUBLIC

GEORG ZOELLER — BIOWARE AUSTIN

In the late stages of MMO development, there is perhaps nothing more important than the ability to rapidly iterate on content. In this design lecture, BioWare Austin's Georg Zoeller shows how his studio uses spatial analysis to track player behavior on the ground level, showing off the studio's homegrown HoloProjector visualization toolkit and giving tips on how to begin applying spatial analysis to your game, even if you don't have a HoloProjector to call your own.

SETTLERS ONLINE: Moving a Traditional European Boxed Game to a Worldwide Free to Play MMO Experience

BENEDIKT GRINDEL AND CHRISTOPHER SCHMITZ — UBISOFT BLUE BYTE

All eyes will be on Blue Byte, after SETTLERS ONLINE lead designer Teut Weidemann's controversial talk at GDC Europe last year asserted that microtransaction-based games should "exploit human weakness" in order to be profitable. At GDC Online, the game's head of production and head of live operations will give a mini-postmortem of the game's development, with a particular focus on how to best bring a successful franchise to the free-to-play world.

Successful Publishing on Smartphone Platforms: iOS vs. Android

GIORDANO BRUNO CONTESTABILE — POPCAP GAMES

You've got a great idea for a smartphone game, but what platform do you put it on? iOS? Android? Both? PopCap's head of mobile strategy Giordano Bruno Contestabile hopes to help you answer that question by sharing how these decisions are made at his company, with a particular emphasis on managing games as a service and leveraging in-app purchases.

The Future Is Now—Emergent Narrative Without Ridiculous Tech

MATTHEW WEISE — SINGAPORE-MIT GAMBIT GAME LAB

According to MIT GAMBIT Game Lab researcher Matthew Weise, replayable emergent narrative is often looked at as a tech problem, rather than a design problem, which he doesn't think is right. Weise will outline a method for tackling this issue with creative design, showing off the research he's done and the prototypes developed at MIT as part of an ongoing research project into experimental narrative design methods.

Threat Modeling for Game Developers

STEPHEN BEEMAN — GIZMOCRACY

"Threat Modeling" is likely not a term you've heard a lot about before, but with all the recent security hacking woes, it's one you're likely to be hearing more and more of. Threat modeling is a process for identifying and preventing security risks that may threaten the personal information of your players, and as Beeman hopes to show, the same processes that are used on major operating systems can still apply to 99-cent smartphone games.

The Year in Social Games 2010–2011

STEVE MERETZKY AND DAVE ROHRL — PLAYDOM

Self-described "grizzly" social game veterans Steve Meretzky and Dave Rohrl of Playdom will present what is sure to be an entertaining annual review of the industry's ups and downs. Topics from the always-entertaining duo will include the latest trends, the most interesting games, and notable new viral and monetization techniques. The aim of the talk will be to separate social gaming myths from the truth because, as they explain it, "a lot of the conventional wisdom about the sector is actually wrong."

EVERQUEST II Extended: Streaming a Non-Streaming Game

JOSHUA KRIEGSHAUSER — SONY ONLINE ENTERTAINMENT

Sony Online's technical director Joshua Kriegshauser will reflect back on the Herculean task of switching EVERQUEST II from a client-side install requiring several Gigabytes of hard drive space to a streaming system. Kriegshauser says that attendees will "learn the intricacies, initial assumptions and lessons learned of taking a live system from file-based storage to real-time asynchronous streaming of assets from the Internet."

Sins of the Past

STEVE JAROS — VOLITION

As Woody Allen once said, "If you're not failing every now and again, it's a sign you're not doing anything very innovative." In what he hopes will be a roundtable confessional, Volition senior writer and designer Steve Jaros will swap war stories with attendees about failures stemming from "bad planning, over eagerness, or sometimes just a straight up lousy idea that sounded good at the time." As he explains it, these conversations often happen in after-hours parties, but not everyone gets invited to those.

From Subscription to Hybrid—Introducing Virtual Item Sales into EVE Online

BEN COCKERILL — **CCP GAMES**

When EVE Online introduced "vanity" virtual goods earlier this year, it caused outrage amongst the game's most dedicated players that was only resolved after a committee of players went to Iceland for face-to-face talks with CCP. In this talk at the Virtual Items Summit, CCP's Cockerill will discuss shifting the game's business model from subscription-only to a subscription and virtual item hybrid after eight successful years, and will be catered to those looking into moving into virtual goods for their games.

Fundamental Multiplayer RPG Math

SARA JENSEN SCHUBERT — **KINGSISLE ENTERTAINMENT**

KingsIsle's Sara Jensen Schubert will bring her RPG system expertise from Shadowbane and DC Universe Online to help attendees implement RPG-style character stats into their games. Special emphasis will be placed on creating an initial, flexible framework for basic RPG systems, and will focus on data-driven spreadsheets that allow easy visualization and iteration of data. Topics will include NPC attributes, items, cash economies, and experience curves.

Get Over Yourself: Making Someone's Else's Game

LARALYN MCWILLIAMS — **LOOT DROP**

Even the best designers know that the audience comes first, and sometimes that means making a game that isn't necessarily your cup of tea. McWilliams will call upon her experiences designing Full Spectrum Warrior and the MMO Free Realms to help you learn about your players and view the world (and by extension, your game) from their eyes. "By shifting the focus from the developer to the player, and from the specific game to the art of entertaining with interactive media, designers can find inspiration and joy in their craft," says McWilliams.

Marvel Super Hero Squad Online Postmortem—An MMO For the Whole Family in Under Two Years

JAY MINN AND JASON ROBAR — **THE AMAZING SOCIETY**

Creative director Jay Minn and studio manager Jason Robar say they've learned a lot from their experiences developing Marvel Super Hero Squad Online. How to work with a major licensee, how to use Unity as a client engine, how to "capture the fun" in two weeks of prototyping, and how to create an MMO for people of all ages are just some of the topics these two promise to discuss during their postmortem of the game, which they say will be "highly interactive, full of humor, war stories, and audience participation."

Emerging Trends In Games-as-a-Service

ATUL BAGGA — **THINKEQUITY**

Trusted research analyst Atul Bagga will discuss emerging trends across the entire gaming industry, including social, mobile, online, and consoles, with a particular emphasis on the maturing games-as-service markets in Japan and social-mobile games in China. The intended takeaway here is to get a head start on international trends that most analysts feel will be replicated in Western markets sooner than others might be anticipating.

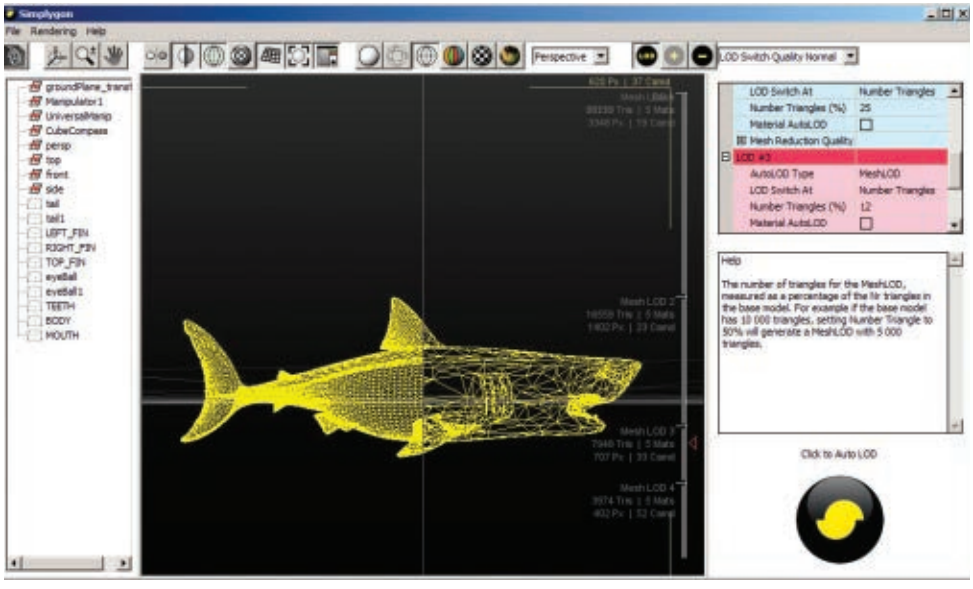
REVIEW BY BIJAN FORUTANPOUR

DONYA LABS

Simplygon

THE HUMAN BRAIN WEIGHS ABOUT 3 POUNDS, CONTAINS ABOUT 100 BILLION NEURONS, AND generates almost 25 watts of power. Although that is enough to light a light bulb, it seems like a shame to spend an ounce of energy creating Level Of Detail (LOD) models by hand in a game production pipeline. Artists bring creativity, imagination, and vision to a team. It is a waste of valuable resources to use any part of their time and talent performing mundane tasks that can be done automatically using intelligent software. The good news is that there are always people looking to improve the machinery of game development. Simplygon from Donya Labs claims to do just that, and today we will investigate whether Simplygon can actually lighten the load artists have to bear.

Simplygon is a tool that takes a textured polygonal model and generates new lower resolution models that can be used as LOD models in-game. The model may be a skinned and animating character model, or an environment object. Simplygon uses proprietary algorithms to intelligently decimate the model and produce high quality results.



Simplygon LOD generation

WHAT'S IN THE BOX

» The Simplygon package consists of a standalone application, which can natively load .OBJ, .FBX, and Collada files, and there are also Maya and Max plug-ins. Most importantly, there's a C++ SDK which accesses all Simplygon functionality, allowing tighter integration into production pipelines.

One of the great strengths of Simplygon is its ease of use. Files can be loaded directly into Simplygon, or indirectly through Maya or Max. Let's look at the Maya workflow. First, the model is loaded into Maya, and the desired geometry nodes are selected. Next, the MEL command "Simplygon" is run. This invokes the Simplygon application, and the model is automatically transferred over to Simplygon. The next step involves some decision making, such as quality settings, and the number of LODs needed. During my tests, I never changed the settings from "Normal" quality, and frankly, the results were great out of the box. One decision that has to be made is how much to reduce the base mesh for each LOD. Simplygon does have a standard option for choosing the desired number of triangles, which is done by setting a percentage (e.g., LOD #1 should have 50 percent fewer triangles than the base, LOD #2 should have 25 percent, and so on). However, it's easier and more intuitive to think of it as the number of pixels you think a particular LOD should occupy on the screen. Thinking of it this way lets the computer decide how many triangles are necessary.

MESH LODS VS PROXY LODS

» In traditional LOD systems, focus is placed on reducing the triangle count in a mesh, which Simplygon calls Mesh LODs. The textures, shaders, and materials remain the same, and are shared between all LODs. If the triangle count is reduced significantly, fewer and fewer triangles are submitted in a rendering batch. This means the cost of rendering the triangles becomes less significant than the overhead cost of the render state changes. So the problem becomes twofold. Not only is the amount of geometry rendered reduced, but the number of render state changes is reduced as well.

To address this problem, Simplygon can generate a Proxy LOD, which is a new mesh with a new set of textures that are visually almost identical to the original. It's not a simple copy, but rather the result of an extensive process of analyzing the model, sampling it, filling holes, removing interior parts, merging triangle geometry, and re-meshing to generate a new mesh with massive geometry reduction. You could think of it an intelligent shrink-wrap of the original.

The next automated step in the process involves the creation of Proxy Textures. Simplygon automatically does an unwrap of the UVs, then re-packs them to create the smallest texture map possible while also getting as close as it can to the original mesh. It's important to note that the new UVs generated will not have the same layout as the original UVs. They're packed for optimal space saving and display. After the new UVs have been generated, the textures are baked into a single texture sheet, including color, normal, occlusion, and specular.

One interesting feature of Proxy LODs is that they are not limited to single objects. A group of hundreds

of objects that happen to be near to or intersecting with each other, such as buildings in a cityscape, can be replaced by a single proxy mesh.

HEAVY LIFTING

» I decided to do a stress test to find the limits of Simplygon's processing power. I found a few models created in ZBrush, which is known for its ability to create heavy models. The largest model I had at hand was a human figure consisting of 50 geometry mesh nodes totaling 850,000 triangles. I tasked Simplygon with generating three LODs by completely re-meshing them, creating Proxy LODs. It managed this in less than five minutes on a dual core 2.4 GHz CPU. This success brings to mind all the other uses for Simplygon, such as 3D scanning data cleanup, or optimizing scene generation for rendering data in the VFX industry. Last but not least, there's the ability to transfer art assets from a digital film production facility to a game development company for use in-game.

UNIQUE FEATURES

» One unique feature of Simplygon (aside from Proxy LODs) is the ability to handle animation and skinned models. The skinning information is taken into account, and the polygon reduction ensures that the joints in LOD models can still animate correctly without artifacts. This is obviously an extremely important feature when it comes to 3D game characters.

Another clever feature is the 3D viewer. There is a vertical "split screen" mode, which allows the quality of the polygon reduction to be more easily seen. This is done by splitting the 3D view window into two halves: the left side of the screen shows the left half of the base model at full resolution, and the right side shows the right half of the model, drastically reduced in polygon count. In wireframe mode, the results are very interesting, as the quality of the silhouette is easy to judge. The results are uncanny, and the silhouette is preserved extremely well.

With the 3D viewer, you can also switch between LODs as the camera is dollied in and out relative to the object. The switch between LODs is extremely subtle, if at all noticeable.

WHY SIMPLYGON ?

» One natural question that comes to mind is, why not use the polygon decimation tools inside a robust 3D package such as Maya? I decided to compare, thinking that at least for static objects Maya would perform the task easily. If successful, Simplygon's solution would be much less compelling.

I began with a model of a great white shark created by Rhythm & Hues Studios, purchased from TurboSquid.com. Using Maya 2012 x64 Hotfix2, I loaded the model, combined the meshes, and ran Mesh->Reduce at 50 percent reduction. I got an error message, "Error: Cannot reduce polygonal object with non-manifold geometry. Cleanup the object using Mesh->Cleanup before reducing." Disappointed but still hopeful, I ran the cleanup tool, set it to remove non-manifold geometry, and clicked OK. Maya immediately crashed, and brought up the "Send error report to Autodesk." No thanks.

What may have caused the non-manifold geometry was that I had combined all the model pieces into a single mesh. I then ran the Mesh->Reduce menu again without the combine step, and was able to get a result. However, the result was very faceted, which led me to believe the normals had not been processed in an intelligent way. I then ran Average Normals to smooth things out, but the result wasn't nearly as good as what Simplygon had produced. Details in sharper, finer areas had been lost, and the silhouette quality was not as good.

It also became clear that, while Maya simply runs a 50 percent reduction on each individual piece, Simplygon takes a more intelligent global approach. Varying polygon model nodes that make up the whole are decimated by varying factors, leaving a larger poly count to more "important" pieces. Presumably, those decisions are based on the scale of the polygon mesh node.

QUIRKS AND GOTCHAS

» While Simplygon's great strength and secret sauce is its ability to easily generate high-quality LOD models, as with any software package, there are some minor issues where some polish is still needed. However, looking at current and past release notes, it's clear that Simplygon has an aggressive development schedule, with each version introducing significant new features and fixes. The current issues I encountered, as of version 3.5, are very minor.

In 64-bit versions of Maya (2011, 2012), loading the Simplygon plug-in takes a full 30–36 seconds. This delay is presumably due to a license check being done in the background. The delay is significant enough to where I would be wary of turning on Auto-Load, thereby increasing Maya startup times. This delay does not occur in 32-bit versions of Maya.

As mentioned earlier, when invoking Simplygon from within Maya using the "Simplygon" MEL command, the Simplygon standalone GUI application is launched with the models loaded. However, there is one crucial necessary step, which is that the models to be transferred must be selected first. On more than a few occasions, I forgot to do so, and Simplygon showed a list of nodes, mirroring the Outliner in Maya, but no geometry was present in the 3D viewer. Perhaps a feature should be added such that if no models are selected, then ALL the models are transferred over by default.

During the time Simplygon is running, Maya is completely blocked. The Maya application does not refresh, and moving windows in front of it leaves a trace of repeating window frames. One may misinterpret this as a crash, but in reality Maya is simply waiting for the "Return To Maya" button to be pressed inside Simplygon. When that happens, everything returns to working order.

DONYA LABS
Simplygon

Ostervagen 38, SE16955
Solna Stockholm, SWEDEN
http://www.simplygon.com

PRICE

- Available upon request

SYSTEM REQUIREMENTS

- Pentium 4
- 2 GB memory
- Windows XP
- Graphics card supporting OpenGL
- 10 GB free hard disk space

PROS

1. Produces great models
2. Reduces render batch counts
3. Takes character model skinning information into account

CONS

1. Maya waits until Simplygon is finished processing
2. 64-bit Maya plugin has a loading delay
3. Pricing may be out of range for smaller studios

PRICE

» The cost of Simplygon depends on several factors, such as the type of game, or whether a studio license is required. Prices are available upon request from Donya Labs. While the figure you receive may be the price of a small sports car, it is important to do the math first. The cost savings is enormous, should your game need LODs. A medium sized MMO can easily contain 7,000 models, and a large MMO can contain up to 25,000 if not more. If you estimate that it would take an artist four hours on average to generate several LODs per model, a small game will require 28,000 hours, or 14 man-years. A large MMO would take 100,000 hours, or 50 man-years. In this sense, Simplygon can pay for itself over time.

SIMPLY FASTER

» Simplygon attacks the difficult problem of LOD generation masterfully, and joins the ranks of smart tools that can have a huge impact on time savings in game development. Automatic LOD generation may very well be as significant in time savings as technologies such as motion capture, automatic lipsync, automatic animation rigging, and physics engines, to name a few. More importantly, it frees up artists' brains to work on what really matters—and then grab a few beers and kill a few brain cells afterward!

BIJAN FORUTANPOUR *is a senior graphics programmer at Sony Online Entertainment in San Diego and has 18 years' experience in the visual effects and games industries.. He is also the author of Enzo 3D paint for Photoshop (www.enzo3d.com). Email him at* **bforutanpour@gdmag.com**

product news

Valve's Source SDK Goes Free

HALF-LIFE and PORTAL creator Valve will release a software development kit for its proprietary Source Engine and modding tools free of charge. Normally, the Source SDK is only available to those who purchase a game built on the Source Engine. But now that TEAM FORTRESS 2 has gone free-to-play, the SDK has too.

In addition to offering users free development tools, the release also gives users access to a variety of free mods. Currently, the freely-available SDK appears to be more of a side effect than a planned initiative, as only certain mods work with the SDK provided, but Valve plans a more formal roll-out soon.

–FRANK CIFALDI

TEAM FORTRESS 2.

Sony Reveals PSP Engine for Remaster Series

The engine behind Sony's upcoming PSP Remaster series, which is set to offer enhanced versions of past PSP games for the PlayStation 3, has been detailed in full.

The PSP Engine, as revealed at the Game Tools & Middleware Forum 2011 event in Tokyo, and as reported by Andriasang, is the base engine for the upcoming series, and will act as a connection between the PS3 OS and selected PSP games.

Sony's Kentaro Suzuki explained that the new engine allows original PSP titles to access to high resolution rendering, as well as 3D stereoscopic output.

The engine also enables the use of PS3 wireless controllers with the remastered games, while supporting an ad-hoc mode through the appropriate application on the PS3.

A memory expansion feature handles the high resolution textures, allowing original PSP assets to be swapped out for the enhanced visuals. Common save data between old PSP titles and the new remastered versions also comes as part of the new engine.

The new series is being launched in Japan, with Capcom's MONSTER HUNTER PORTABLE 3RD HD VER. hitting the store first. Further expansion into the U.S. and European territories is planned.

–MIKE ROSE

GameSalad Adds HTML5 Publishing to Game Creation Tool

GameSalad has announced that users of its drag-and-drop game creation engine can now publish titles that work with the HTML5 standard, and embed them for play on any compatible web browser.

With most major web browsers now supporting HTML5's options for heavily interactive web apps, the company says the option gives a game a much bigger potential audience than plugins like Flash or Unity, and the company is helping promote many of those games through the GameSalad Arcade section of its web site.

While HTML5 web apps are playable natively on many mobile browsers, GameSalad Chief Product Officer Michael Agustin told Game Developer he doesn't see the option primarily as a way to do an end-run around the certification requirements and fees associated with mobile app stores.

"The App Store and the Android Marketplace still serve as the primary discovery model on mobile devices, where native apps often can provide a better overall experience," Agustin said. "That said, HTML5 has less friction because the content is delivered as you browse the Web, and there is no download or content to install."

GameSalad doesn't currently offer any built-in monetization options for HTML5 developers, but Agustin said he sees the web games as "the best way for players to learn about and experience mobile games before downloading them."

–KYLE ORLAND

Adobe Flash Builder Update Adds iOS, BlackBerry Tablet Support

Following up on its recently-debuted support for Android, Adobe has updated its Flash Builder development suite to support iOS and BlackBerry PlayBook devices, allowing developers to make cross-platform apps with one codebase.

The Adobe Flash Builder MXML editor (formerly known as Adobe Flex Builder) allows creators to build cross-platform applications for mobile, web, and desktop applications using ActionScript and Adobe's own Flex framework.

"Developers can quickly build and distribute apps through the Android Market, BlackBerry App World, and Apple App Store using one tool chain, programming language and code base—a first for developers!" Adobe's Puneet Goel wrote on Adobe's Flash blog.

While this doesn't bring iOS devices any closer to supporting Flash-based applications (Apple CEO Steve Jobs blames "technology issues"), it could ease development for those wishing to create cross-platform applications or port any existing Flash content to iOS.

–FRANK CIFALDI

Microsoft Releases Non-commercial Kinect SDK for PC

Microsoft has officially released a non-commercial beta version of a long-promised PC software development kit for its Kinect depth-sensing camera, for use in C++, C# and Visual Basic projects.

The SDK, which is available for download now, provides academics and hobbyists access to raw sensor streams from Kinect's RGB and depth-sensing cameras, as well as its directional microphones.

The software will also provide automatic skeleton tracking for up to two people and audio processing features such as echo cancellation and source recognition. It comes packaged with over 100 pages of documentation and a number of demos.

Hackers have been using homebrew Kinect drivers since just after the hardware's launch to develop everything from art projects to autonomous robotic helicopter guidance systems. Hardware maker PrimeSense released a set of open PC drivers for the hardware last December.

During E3, Microsoft released Kinect Fun Labs, an Xbox Live download providing a trio of tech demos that show off the Kinect's capabilities.

A commercial release for the PC SDK is planned for release "at a later date." Microsoft has also promised eventual XNA support for the Kinect.

–KYLE ORLAND

Game Developers Conference Online® 2011 (GDC Online) is the community meeting point and principle learning platform for game professionals in the online and connected games space, including cloud gaming, MMOs, virtual worlds, casual and social networking games.

Join us at GDC Online, October 10–13, for four days of world-class content across 120+ sessions, covering a wide-ranging scope of online and connected game development topics taught by leading industry experts.

SUMMITS

MONDAY–TUESDAY

- Game Narrative
- GDC Virtual Items
- Smartphone & Tablet Games

PLUS

WEDNESDAY

- Game Career Seminar
- Game Writing Tutorial

MAIN CONFERENCE SESSIONS

TUESDAY–THURSDAY

- Business & Marketing
- Customer Experience
- Design
- Production
- Programming
- Monetization (sponsored)

EXPO FLOOR

TUESDAY–WEDNESDAY

- Leading online game publishers
- Cloud-based gaming services
- Payment solution companies
- Development tool vendors
- Networking, playable games

and more!

2ND ANNUAL CHOICE ONLINE AWARDS

WEDNESDAY

- Best Online Game Design
- Best Social Network Game
- Best New Online Game
- Online Game Legend

and more!

GDC 11 Online

REGISTER BEFORE SEPTEMBER 1ST AND SAVE UP TO 40%!

Game Developers Conference Online® 2011

October 10–13, 2011 | Austin Convention Center | Austin, Texas

Visit **www.gdconline.com** for more information.

For the latest news, contests and promotions visit us on: twitter and facebook.

UBM TechWeb

READY, SET, ALLOCATE!

DYNAMIC MEMORY ALLOCATION WITH A MEMORY HEAP SYSTEM

ALLOCATING DYNAMIC MEMORY CAN BE A MAJOR SLOWDOWN in any game, but a memory heap system that replaces the built-in allocation system can help to overcome slowdown situations. A technique I've used to good effect is to allocate as much memory as possible into one heap, and then rewrite Malloc and Free to use that heap. This requires tracking and a bit of memory, but the speed increase has proven well worth the effort.

Presented here are some of the basics to a method I've used, as well as some metrics to show the speed increase. All my tests are run on an old desktop running Windows XP Pro Service Pack 3, with a Pentium 4 3.0GHz 32-bit processor, 800MHz FSB, with 512MB of RAM.

The first thing to do is set up the tracking. Following the tracking will be a breakdown of Malloc, and then Free. Afterward we'll discuss a simple high-resolution performance timer, test, and metrics.

TRACKING

» In my code I use an array of tracking units—each element is an unsigned 32-bit integer with each bit representing a page of tracked memory. The size of the array is determined by the amount of memory to be tracked, and the page size.

First we need to choose the amount of memory to track. The current console systems range up to 512MB of memory, but most of my work has been done on the PSP, which only has 32MB, so that's the size I chose to test with, even though the PSP only really allows use of about 20MB after the OS and volatile memory.

32 * 1024 * 1024 = 33554432 Bytes

Second, we must choose the page size for each allocation of memory to track. This will be the minimum allocation size, so we need to strike a balance that doesn't require too many small allocations for large files, which would make tracking a heavy burden. We also don't want to waste too much space for a reasonable number of small allocations. In my code, the page size should also be made a power of two, for speed considerations to be explained later. I've chosen 4096 Bytes.

Since each page is represented by a bit in a tracking unit:

number of pages per tracking unit =
4 Bytes per tracking unit * 8 bits (or pages) per Byte
(32 pages per tracking unit)

number of pages required to track the total memory =
33554432 Bytes / 4096 Bytes per page
(8192 pages)

number of tracking units =
8192 pages / 32 pages per tracking unit
(256 tracking units)

So at a cost of 256 tracking units, at 4 Bytes each, it will take 1KB to track 32MB with a 4KB page size. Using a similar setup for 512MB would require 16KB to track, and for 2GB would require 65KB to track. Pretty sweet eh?

I should mention a few typedefs I use:

```
u8 - unsigned char
u16 - unsigned short
u32 - unsigned integer
l64 - long long
```

To aid in tracking, a few constants are set up as below:

```
// Utility macro
#define _MB( size ) size * 1024 * 1024
#define TRACKING_UNIT u32
const TRACKING_UNIT kMemTrackingUnitAllPagesInUse = 0xFFFFFFFF;
const u32 kMemInUse = 0x01;
const u32 kMemNumPagesPerUnit =
   (sizeof(TRACKING_UNIT) * 8 /* Bits per byte */);
const u32 kMemPageSize = 4096;
const u32 kMemTotalTracked = _MB( 32 );
const u32 kMemNumPages = kMemTotalTracked / kMemPageSize +
   ((kMemTotalTracked % kMemPageSize)? 1 : 0);
const u32 kMemNumTrackingUnits = kMemNumPages / kMemNumPagesPerUnit +
   ((kMemNumPages % kMemNumPagesPerUnit)? 1 : 0);
```

And finally the tracking array and memory to be tracked are allocated:

```
TRACKING_UNIT aMemPageTrackingBits[ kMemNumTrackingUnits ];
const void* kpMemory = malloc( kMemTotalTracked );
```

Malloc and Free methods will maintain and use the tracking array.

MALLOC

» First, here comes the huge non-secret secret. Aligned memory is easier to allocate than un-aligned memory, because the addresses work nicely within this system. Since it can be assumed that all allocations will be made along the alignment boundary, there is no requirement to divide and track sub-alignment size blocks, which saves time in operations and space in tracking. The downside is the allocations potentially waste space and fragment memory. With some careful

planning and usage of design patterns such as the Object Pool, the wasted space and fragmentation can be minimized.

One of the dirty little secrets about the tracking system is that it only tracks whether a page of memory is in use, which doesn't inform the Free method about how much memory was previously allocated when Malloc was called. So all Malloc calls should mark the memory being returned with some sort of header to denote how much memory was allocated, so the corresponding Free call will have the information it needs to release the memory and update the tracking information. In my code I use a simple header that tracks the size and the alignment.

```
typedef struct _TAllocationHeader
{
   size_t  uSize;
   u32     uAlignment;
} TALLOCATION_HEADER;
```

Now, technically, a size_t, or 32-bit unsigned integer for the uSize member is overkill for a 32MB memory heap. 2ffi25 would suffice, but trying to go smaller isn't possible with the current data types; the next data type down, a u16, only goes to 65,535. The 32-bit unsigned integer can denote up to a 4GB allocation, which hopefully should be more than will be needed in one allocation. If not, adjust accordingly for your needs.

The 32-bit unsigned integer for the Alignment is also overkill, but instead of trying to conserve space using an 8-bit unsigned char, the u32

LISTING 1

```
TRACKING_UNIT uNumRemPagesNeeded =
((uNumPagesReq >= (kMemNumPagesPerUnit - uPreOffset)) ?
(uNumPagesReq - (kMemNumPagesPerUnit - uPreOffset)) : 0);

if(uNumRemPagesNeeded == 0)
{
  aMemPageTrackingBits[ uBegTrackUnit ] |= uPreBitMask;

  u32 uAddress = ((uBegTrackUnit * kMemNumPagesPerUnit) +
      uPreOffset) * kMemPageSize;

  memset((void*)((u32)(const_cast< void* >(kpMemory)) +
      uAddress), 0, uAllocHdrPadSize);

  ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
      uAddress + uAllocHdrPadSize))->uSize = uSize;
  ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
      uAddress + uAllocHdrPadSize))->uAlignment = uAlignment;

  return (void*)((u32)(const_cast< void* >(kpMemory)) +
      uAddress + uAllocHdrSize);
}

u32 uNxtTrackUnitToChk  = (uBegTrackUnit + 1) %
      kMemNumTrackingUnits;
if(uNxtTrackUnitToChk < uBegTrackUnit)
{
        return NULL;
}
```

Figure 1: **Big Endian bit field.**

is chosen because it keeps the address at the end of the header aligned within the 32-bit system. Keep in mind when modifying the allocation header's size that it should be a multiple of the alignment size. This means that memory addresses returned will be useable with operations that require memory alignment (such as those used in SIMD).

The Malloc function signature looks like this:

```
void* my_malloc( size_t uSize, u32 uAlignment )
```

uSize is the amount of memory in bytes being requested, and uAlignment is the size in bytes to use for calculating address boundaries. Calls to malloc can be redirected to use my_malloc by providing and using a define that can be swapped between Malloc or my_malloc and inserts whichever is chosen during the pre-processor phase of compilation.

```
#define MYMALLOC( uSize ) my_malloc( uSize, 4 )
```

There are several things Malloc needs to do, so let's go through them one by one. The first four steps are mostly housekeeping, so I've kept the explanations mostly to just a description of what is done in code.

1 // Add padding to compensate for alignment of the allocation.

If the allocation request doesn't match the alignment requirement, it's simple enough to fix. The additional space is calculated by taking the modulus of the size by the alignment, subtracting the remainder from the alignment, and adding the result back into the size.

```
uSize += (((uSize % uAlignment) > 0) ?
uAlignment - (uSize % uAlignment) : 0;
```

2 // Align the header so the beginning address of the memory will be aligned.

As an optimization, this step can be skipped if you know that your allocation header will always take enough memory to leave the next available byte on an alignment boundary. However, if someone calls my_malloc with an alignment size greater than the size of the allocation header, the result will be an unaligned address boundary. The size in bytes to pad the allocation header is obtained using a similar method to what was used to pad the allocation itself. The final size of the allocation header is also saved so it can be used to calculate the return address.

```
u32 uAllocHdrPadSize = ((sizeof(TALLOC_HDR) % uAlignment) > 0) ?
uAlignment - sizeof(TALLOC_HDR) % uAlignment : 0;
u32 uAllocHdrSize = sizeof(TALLOC_HDR) + uAllocHdrPadSize;
```

3 // Add the size of the header to the allocation.

Nothing big here, just keeping track of how much memory will be required.

```
uSize += uAllocationHeaderSize;
```

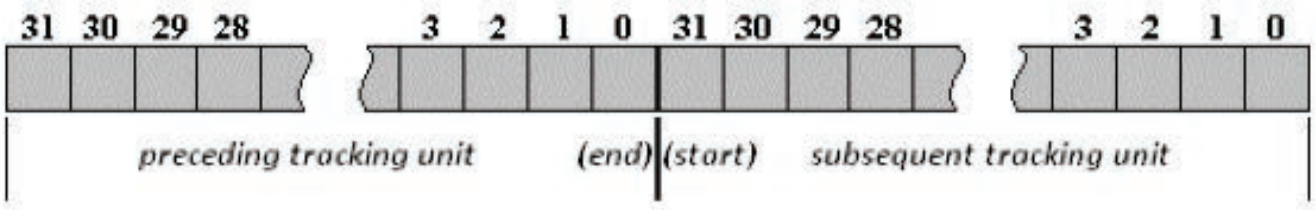


Figure 2: **Preceding/Subsequent tracking units.**

4 // Check the request to make sure it will fit in the memory being tracked.

The number of pages being requested is calculated and saved. First, the number of whole pages is obtained by dividing the allocation size by the size of a page of memory. Then any remaining memory that is required will fit into one page, so one page will be added if the allocation size modulus by the size of a page of memory is anything other than zero.

```
const u32 uNumPagesReq = uSize / kMemPageSize +
((uSize % kMemPageSize) ? 1 : 0);
```

Next the number of tracking units requested is calculated and saved in a similar fashion. The number of whole tracking units is the number of pages requested divided by the number of pages per tracking unit. Then any remaining pages will fit into one tracking unit, so one tracking unit will be added if the number of pages requested modulus by the number of pages per tracking unit is anything other than zero.

```
const u32 uNumTrackUnitsReq = uNumPagesReq / kMemNumPagesPerUnit +
((uNumPagesReq % kMemNumPagesPerUnit) ? 1 : 0);
```

Lastly, the number of tracking units requested is compared to the total number of tracking units. If there are more tracking units being requested than the total number, the allocation failure is reported by returning NULL.

```
if( uNumTrackingUnitsReq > kMemNumTrackingUnits )
{
   return NULL;
}
```

5 // Find enough contiguous pages of memory to satisfy the allocation.

The search for pages is broken up into three stages; starting point, whole tracking units, and remaining pages. The starting point looks for a tracking unit that either resolves the entire request or has pages available at the end, and is contiguous to the next tracking unit. The whole tracking units consist of 32 pages, at 4KB each, and by searching for available whole tracking units, the search is sped up by not having to do an individual search for each page internally. The remaining pages stage aims to fulfill the remainder of the request by checking the tracking unit that is contiguous to either the tracking unit used for the starting point, or the last whole tracking unit, depending on which was last used.

With the given machine architecture (Little Endian), I still think of the units of the tracking array like they are Big Endian. When the end of the tracking unit is referred to, this refers to the least significant bit. The beginning or start of the tracking unit is the most significant bit (see Figure 1).

The array of tracking units will position the end of each preceding tracking unit next to the start of the subsequent tracking unit as seen in Figure 2.

The search begins at the first tracking unit, and continues iterating over the array of tracking units until they have been exhausted.

```
u32 uBeginningTrackingUnit = 0;
while(uBeginningTrackingUnit < kMemNumTrackingUnits)
{
```

5a // Find a starting point of available pages within a tracking unit.

Four potential situations exist within the first tracking unit.

1. All the pages have been used.
2. Not enough free pages exist internal to the tracking unit to fulfill the request.
3. All the pages needed to fulfill the request are contained within the starting tracking unit.
4. Available pages exist starting at some point within the tracking unit, extending to the end, and may be used in conjunction with the next tracking unit to fulfill the request if enough available contiguous pages exist.

The first two situations are not useful for fulfilling the request, so the last two are what we should test for. If there are enough pages contained within a tracking unit to fulfill the request, they can be located by building a bitmask that represents enough pages, then using that bitmask to find a match with the available pages. The number of pages not needed in a tracking unit is calculated by subtracting the number of pages needed from the number of pages per tracking unit. The bitmask is then created by bit shifting the "all pages in use" constant toward the end of the tracking unit by the number of pages not needed; this leaves the number of pages that are needed as the bitmask.

```
u32 uPreOffset = 0;
 TRACKING_UNIT uPreBitMask = 0;
if(uNumPagesReq < kMemNumPagesPerUnit)
{
   uPreBitMask = kMemTrackingUnitAllPagesInUse <<
      (kMemNumPagesPerUnit - uNumPagesReq);
 }
```

If there are not enough available pages contained within a tracking unit to fulfill the request, then either the entire tracking unit (if all pages are available) or whatever available pages are at the end of the tracking unit can be used in conjunction with the available pages contained in the start, or the entirety of the subsequent tracking unit. The bitmask that represents all pages available is built from this.

```
else
{
   uPreBitMask = kMemTrackingUnitAllPagesInUse;
}
```

Once the bitmask is created it is repeatedly checked and shifted toward the end of the tracking unit until a match to the bitmask is found, or all

bits in the bitmask are exhausted. If a match isn't found prior to shifting the bitmask, then the bits that are shifted off the end of the tracking unit, representing needed pages, will need to be found in the next tracking unit, and this is kept in the uPreOffset counter used in a for loop.

```
for( ; uPreOffset < kMemNumPagesPerUnit; ++uPreOffset)
{
 if((~(aMemPageTrackingBits[ uBegTrackUnit ]) &
uPreBitMask) == uPreBitMask)
 {
    break;
 }
 uPreBitMask = uPreBitMask >> 1;
}
```

Next, a check is done to make sure the bitmask wasn't completely exhausted, and if it was, to move to the next tracking unit and start over.

```
if(uPreOffset == kMemNumPagesPerUnit)
{
 uBegTrackUnit++;
 while(aMemPageTrackingBits[ uBegTrackUnit ] ==
 kMemTrackingUnitAllPagesInUse)
 {
    uBegTrackUnit++;
 }
 continue;
}
```

If the bitmask wasn't completely exhausted, then the number of remaining pages needed is calculated and checked. If more pages are needed then a check is performed to ensure more tracking units are available. When no more tracking units are available, but more pages are needed, a null is returned. In the case that enough pages have been found to fulfill the request, the pages to be assigned are marked as in use, the beginning memory address is calculated, the memory used to pad the allocation header, if any, is cleared, the allocation header is written, and the memory address immediately following the allocation header is returned. By placing the allocation header immediately prior to the memory address returned, the size and alignment information can be easily retrieved during a free method call—or anywhere it needs to be viewed, such as in a debugger—simply by moving the memory address pointer backward. See Listing 1.

5b // Find whole tracking units where all pages are available and required in fulfilling the request.

Since the tracking units are 32-bit unsigned integers, any tracking units where all pages in the tracking unit are available will have a value of zero. The number of contiguous whole tracking units to find is calculated, and then the tracking units subsequent to the starting tracking unit are evaluated. If a partially or fully-used tracking unit is encountered before all the required whole tracking units are found, then the tracking unit to start searching from is moved up to the current tracking unit, and the search for whole tracking units is terminated. If a tracking unit is completely available, it is counted and evaluation continues until failure, or all required contiguous whole tracking units are found and confirmed, as below.

```
u32 uNumContgTrackUnitsAvail   = 0;
u32 uNumContgTrackUnitsToFind  =
 uNumRemPagesNeeded / kMemNumPagesPerUnit;
for(u32 i = uNxtTrackUnitToChk;
i < kMemNumTrackingUnits &&
```

LISTING 2

```
 if((~(aMemPageTrackingBits[ uNxtTrackUnitToChk ]) &
 uPostBitMask)
         == uPostBitMask)
 {
     aMemPageTrackingBits[ uBegTrackUnit ] |= uPreBitMask;
     for(u32 i = (uBegTrackUnit + 1); i < uNxtTrackUnitToChk
     ; ++i)
     {
 aMemPageTrackingBits[ i ] |=
 kMemTrackingUnitAllPagesInUse;
     }
 aMemPageTrackingBits[ uNxtTrackUnitToChk ] |= uPostBitMask;

 u32 uAddress = ((uBegTrackUnit * kMemNumPagesPerUnit) +
     uPreOffset) * kMemPageSize;

 memset((void*)((u32)(const_cast< void* >(kpMemory)) +
     uAddress), 0, uAllocHdrPadSize);

 ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
     uAddress + uAllocHdrPadSize))->uSize = uSize;
 ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
     uAddress + uAllocHdrPadSize))->uAlignment = uAlignment;

 return (void*)((u32)(const_cast< void* >(kpMemory)) +
     uAddress + uAllocHdrSize);
 }

 uBegTrackUnit = uNxtTrackUnitToChk;
}
```

```
uNumContgTrackUnitsAvail < uNumContgTrackUnitsToFind; ++i)
{
 if(aMemPageTrackingBits[ i ] == 0)
 {
    uNumContgTrackUnitsAvail++;
 }
 else
 {
    uNumContgTrackUnitsAvail = 0;
    uBegTrackUnit = i;
    break;
 }
}
```

Once the search has terminated or completed, the number of available whole tracking units found is compared to the number of contiguous whole tracking units required, to make sure the search wasn't terminated early. If the search was terminated early, then the search for the requested memory must begin again at the beginning.

```
if(uNumContgTrackUnitsAvail != uNumContgTrackUnitsToFind)
{
 continue;
}
```

The number of pages needed is recalculated, subtracting out the pages contained in the whole tracking units that were just located.

```
uNumRemPagesNeeded = uNumRemPagesNeeded -
  (uNumContgTrackUnitsAvail * kMemNumPagesPerUnit);
```

A check is then performed to determine whether all the pages needed have been found. If they have, then the same steps are taken as described at the end of the search for the starting point—except when the pages are marked as used, the whole tracking units used to satisfy the request are marked as well. If more pages are required then the algorithm continues.

```
if( uNumRemPagesNeeded == 0 )
{
   aMemPageTrackingBits[ uBegTrackUnit ] |= uPreBitMask;
   for(u32 i = uNxtTrackUnitToChk;
i < (uNxtTrackUnitToChk + uNumContgTrackUnitsAvail); ++i)
   {
      aMemPageTrackingBits[ i ] |= kMemTrackingUnitAllPagesInUse;
   }

   u32 uAddress = ((uBegTrackUnit * kMemNumPagesPerUnit) +
uPreOffset) * kMemPageSize;

   memset((void*)((u32)(const_cast< void* >(kpMemory)) +
uAddress), 0, uAllocHdrPadSize);

   ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
uAddress + uAllocHdrPadSize))->uSize = uSize;
   ((TALLOC_HDR*)((u32)(const_cast< void* >(kpMemory)) +
uAddress + uAllocHdrPadSize))->uAlignment = uAlignment;

   return (void*)((u32)(const_cast< void* >(kpMemory)) +
uAddress + uAllocHdrSize );
 }
```

5c // Find the remaining pages needed to fulfill the request.

Before checking for the last tracking unit to fulfill the request, a check is performed to make sure the index for the next tracking unit to check is not past the end of the array of tracking units. If the next tracking unit to check would be past the end of the array, then not enough contiguous memory exists, and null is returned. When the index for the next tracking unit to check is valid, then a bitmask is created for the remaining pages needed; the bits for this are shifted to begin at the start of the tracking unit, because the pages must be contiguous with the previously located pages.

```
uNxtTrackUnitToChk = (uNxtTrackUnitToChk + uNumContgTrackUnitsAvail)
% kMemNumTrackingUnits;
if(uNxtTrackUnitToChk <= uBegTrackUnit)
{
 return NULL;
}

TRACKING_UNIT uPostBitMask = kMemTrackingUnitAllPagesInUse <<
(kMemNumPagesPerUnit - uNumRemPagesNeeded);
```

One last check is performed to determine whether enough contiguous pages have been located to fulfill the request. The new bitmask is compared to the next tracking unit to check. If a match is found, then the same steps are taken as described at the end of the search for the starting point, except when the pages are marked as used the whole tracking units used to satisfy the request and the pages required in the next tracking unit to check are marked as well. If a match is not found, then the tracking unit from which the check will start is moved to the tracking unit that was evaluated for the end in this iteration, because it may be the start of the tracking units that will fulfill the request in the next iteration of the search over the tracking unit array. See Listing 2

If all the tracking units have been exhausted without locating enough pages to fulfill the request, then null is returned.

```
return NULL;
```

And that's my_malloc.

In the bigger picture, there are issues that still need to be considered and dealt with, such as fragmentation or frequent small requests. Issues such as those can be handled through other schemes, like memory heaps, or hashes containing lists of available memory with frequently used sizes and object factories that would be implemented on top of Malloc. The reason these other schemes are not handled in Malloc itself is because in my implementation I chose not to incur the costs associated with them trying to allocate larger blocks of memory. In other words, only incur the execution costs when necessary.

FREE

» To release memory back to the allocation system, we'll take three steps. First, retrieve the allocation header. Then determine the pages used to track the memory. Finally, release the pages back into the memory heap.

The Free function signature looks like this:

```
void my_free (void* pMem)
```

pMem is the memory to be released back to the allocation system.

Similarly to how Malloc was redirected, calls to Free can be redirected to use my_free by providing and using a define that can be swapped between Free or my_free and inserts whichever is chosen during the pre-processor phase of compilation.

```
#define MYFREE( pMem ) my_malloc( pMem )
```

1 // Retrieve the allocation header.

Recall that the allocation header was placed just prior to the address returned from Malloc. So to retrieve the allocation size and alignment information, the address pointed to by pMem is decremented by the size of an allocation header; this makes the address point to the beginning of the allocation header, so after a quick cast the allocation header members can be retrieved.

```
u32 uSize = ((TALLOC_HDR*)((u32)(pMem) -
   sizeof(TALLOC_HDR)))->uSize;
u32 uAlignment = ((TALLOC_HDR*)((u32)(pMem) -
   sizeof(TALLOC_HDR)))->uAlignment;
```

The amount of padding used before the header is determined to keep the memory aligned, and is used to move the pMem pointer back, so that when the memory is released, all the memory added for the allocation header is released as well.

```
u32 uAllocHdrPadSize = ((sizeof(TALLOC_HDR) % uAlignment) > 0) ?
   uAlignment - sizeof(TALLOC_HDR) % uAlignment : 0;
u32 uAllocHdrSize = sizeof( TALLOC_HDR ) + uAllocHdrPadSize;
pMem = (void*)( (u32)( pMem ) - uAllocHdrSize );
```

2 // Determine the Pages used to Track the Memory.

The first thing to do here is to figure out the "address," or offset from the beginning of the memory being tracked. Then the tracking unit where the allocation was started, as well as the offset for the beginning bit/page within the first tracking unit can be determined using the address.

```
u32 uAddress = (u32)(pMem) - (u32)(const_cast<void*>(kpMemory));
u32 uBegTrackUnit = uAddress / kMemPageSize / kMemNumPagesPerUnit;
u32 uPreOffset = uAddress / kMemPageSize % kMemNumPagesPerUnit;
```

The size information obtained from the allocation header allows the number of pages used to track the allocation to also be calculated.

```
u32 uNumPagesUsed = uSize / kMemPageSize +
   ((uSize % kMemPageSize)? 1 : 0);
```

3 // Release the pages back into the memory heap.

Clearing pages for each tracking unit follows the same basic pattern, a bitmask of the pages used is built, then bitwise inverted and logically AND-ed to the tracking unit, with results stored back into the same tracking unit.

If the entire allocation is contained within one tracking unit, then it is cleared and the function returns.

```
if(uNumPagesUsed < (kMemNumPagesPerUnit - uPreOffset))
{
   aMemPageTrackingBits[uBegTrackUnit] &=
      ~((kMemTrackingUnitAllPagesInUse <<
      (kMemNumPagesPerUnit - uNumPagesUsed)) >> uPreOffset);
   return;
}
```

Otherwise, the free operation is broken down into three parts (see Listing 3):

1. Free the partially-used tracking unit for the beginning of the allocation.
2. Free any whole tracking units used for the allocation.
3. Free a partially-used tracking unit, if used, for the end of the allocation.

And that's it, the Free method is done.

A SIMPLE HIGH-RESOLUTION PERFORMANCE TIMER FOR TESTING

» All testing was done using a simple high-resolution performance timer which uses Microsoft's QueryPerformanceCounter and QueryPerformanceFrequency methods located in WinNT.h. QueryPerformanceCounter returns the current value of a computer's high-resolution performance counter. The frequency of a computer's high-resolution performance counter varies from machine to machine, but will not change while a machine is running, so it must be obtained using QueryPerformanceFrequency. Dividing the difference of two values returned from QueryPerformanceCounter by the frequency returned from QueryPerformanceFrequency gives the elapsed time, and is accurate to one microsecond.

The timer needs three counters (start, paused, and total paused) and five methods (Initialization, Start, Pause, Resume and GetTime). Initialization zeros out the counters, and also determines the frequency. Start and Pause store the value returned from QueryPerformanceCounter to their respective counters. If the pause counter value is greater than zero, Resume subtracts the pause counter from the current value returned from QueryPerformanceCounter, adds the result to the total pause counter, and zeros out the pause counter—otherwise Resume calls Start. GetTime returns the amount of time elapsed by subtracting the start counter from the current value returned from QueryPerformanceCounter, then subtracting the total paused counter, dividing the result by the frequency. Many examples exist on the web, so I've forgone the code implementation here. Please be aware that with multi-processor systems the thread affinity should be specified for the timer using SetThreadAffinityMask

TESTS AND METRICS

» I've used three tests to show some of the differences between the heap system created with my_malloc, my_free, and the tracking array and their standard counterparts. The first test examines how long it takes to do a set number of allocations with a given size. The second test looks at how long it takes to exhaust 32MB of memory with 4KB allocations a set number of times. The third test investigates how long it takes to allocate a set number of times in a worst-case scenario.

1 // How long does it take to allocate X times at size Y?

Multiple sizes, from one byte to 32MB were tested to see how long it took to allocate each one, 1, 10, 100, 1,000, and 10,000 times. When the test was run using the heap system here, allocation times ranged from less than a millisecond (for one byte being allocated one time), to 27.3 milliseconds (for 32MB being allocated 10,000 times). However, when the test is run using the standard memory allocator, allocation times range from less than a millisecond (for one byte being allocated one time), to 9 minutes 17 seconds 442.7 milliseconds (for 32 MB being allocated 10,000 times). Now that's a pretty sensational performance improvement, but something a bit more realistic in real world use might be 64KB allocated 1,000 times, and with that the improvement still shows through with 0.7 milliseconds and 66.3 milliseconds respectively.

Allocating a bunch of objects of the same size can add up in time spent allocating, and if lots of them are done at the same time, it can be a performance issue. The problem becomes more prevalent with larger allocations when the system has to search for larger blocks of contiguous memory.

Windows XP cannot be limited to allocating the requested memory from a specific range of addresses, short of implementing a memory allocation system, which lends itself to increased allocation times while searching for available memory with the standard allocation system, and that is partly what is being demonstrated here.

2 // How long does it take to allocate all the memory X times at 4KB per allocation?

Memory was allocated 4KB at a time until the memory was exhausted, in order to test how long it took when the steps were repeated 1, 10, 100, 1,000, and 10,000 times. Only the smallest size possible from the system is tested, since larger allocations decrease the amount of allocations and time.

Exhausting 32 MB of memory at 4KB per allocation with the heap system here ranges in time from 10.1 milliseconds to do it one time, to 1

LISTING 3

```
aMemPageTrackingBits[uBegTrackUnit] &=
    ~(kMemTrackingUnitAllPagesInUse >> uPreOffset);

uNumPagesUsed -= (kMemNumPagesPerUnit - uPreOffset);

u32 uNextUnitToClear = (uBegTrackUnit + 1);
for( ; uNextUnitToClear <=
    (uBegTrackUnit + (uNumPagesUsed / kMemNumPagesPerUnit));
    ++uNextUnitToClear)
{
     aMemPageTrackingBits[uNextUnitToClear] &=
        ~kMemTrackingUnitAllPagesInUse;
}

uNumPagesUsed -= ((uNextUnitToClear - (uBegTrackUnit + 1)) *
        kMemNumPagesPerUnit);

if(uNumPagesUsed == 0 || (uNextUnitToClear >=
kMemNumTrackingUnits))
{
      // If this assertion fails, then the tracking information
      // is messed up, because it was believed that pages were
      // still in use that must exist past the array of
      // tracking bits.
      assert(uNumPagesUsed == 0);
      return;
}

aMemPageTrackingBits[uNextUnitToClear] &=
      ~(kMemTrackingUnitAllPagesInUse <<
      (kMemNumPagesPerUnit - uNumPagesUsed));
```

minute 39 seconds 995 milliseconds to do it 10,000 times. When using the standard memory allocation system, the times range from 101.5 milliseconds to exhaust 32MB of memory one time, to 15 minutes 52 seconds 593.3 milliseconds, to do it 10,000 times. Even more interesting is the increase in performance of the Free method. To free the same memory that was allocated, the heap system here ranges from 6.1 milliseconds to 1 minute 1 second 891.4 milliseconds, while the standard memory allocation system clocks in a range from 613.6 milliseconds to 1 hour 43 minutes 35 seconds 161.5 milliseconds. At first I didn't believe it took more than an hour and a half to free the memory, so I re-ran the test and came up with a similar time (1:41:58.453.5). While the idea of exhausting the memory 10,000 times in a row may not be a real-world problem, it still highlights the time gains that can be made over many allocations and de-allocations with a specialized system.

Allocating all the memory at 4KB per allocation demonstrates some good and bad allocation conditions, though not the absolute worst. When no memory is previously allocated, available memory is discovered quickly, and because each subsequent allocation is contiguous, whole blocks eventually end up getting examined with a single check, reducing the time to discover available memory. Performing lots of small allocations means that more searches are performed than if larger allocations were made, because larger allocations would exhaust the memory quicker, and more searches adds up to more time spent searching for available memory.

Allocating all the memory in Windows XP isn't the same as allocating all the memory in the system described herein, since Windows XP doesn't guarantee one contiguous chunk of memory for all the allocations, and Windows XP will also begin page-swapping once the physical memory has been exhausted. However, the time it takes Windows XP to allocate 32MB worth of memory at 4KB per allocation versus the time it takes the system described herein to do the same can still be compared for time to allocate a similar amount of memory.

3 // How long does it take to allocate 8KB of memory X times in the worst-case?

The worst-case scenario set up the memory to be entirely allocated in 4KB blocks, then went back through and freed up every other block. At the end of the memory the pre-test conditions ensured an 8KB block was available. An 8KB request was then made to the system and timed to determine how long it takes to find the available memory when the test was repeated 1, 10, 1,000, and 10,000 times.

The worst-case test gives a sense of the time an allocation can take with this heap system at its worst, and shows how important it is to avoid fragmentation of the memory. Times to perform the test range from 390.5 milliseconds to complete 1 time, to 1 hour 5 minutes 12 seconds 634.8 milliseconds, to complete 10,000 times. In the real world one hopes to never get into a fragmentation predicament like the one set up here, and careful planning of heaps is usually done to help avert this kind of situation.

Since there is no way to limit where Windows XP will allocate memory, and because Windows XP will start page swapping to the hard drive when it runs out of physical memory, this last test isn't something that can be fairly compared, so it has been skipped for the standard memory allocator.

HEAPS OF HELP

» While this heap system has its advantages, it can be improved upon in a number of ways. Three ideas spring to mind. The heap system could be made thread-safe. Small allocations, less than the block size, need to be handled at the sub-block level to avoid wasting space. One way that could be done is to allocate a heap containing a set number of blocks (e.g. 128 blocks (512KB)), then any time a small request comes in, it can be redirected to that heap. This method will require some overhead to track the sub-block allocations. Third, this heap system could be improved for debugging by incorporating tracking information to determine where an allocation was made.

I hope you find creating your own memory allocator as helpful as I do. Now go forth and Allocate!

Thanks go to Mike Acton at Insomniac Games for starting AltDevBlogADay.org and nudging me to start writing professionally, and to Lee Marshall from Google, formerly of Locomotive Games when I met him, for leading me back to memory management concepts and pointing me toward Doug Lea's work. Also a special thanks to Simon Lundmark of Pixeldiet Entertainment for digging in and providing corrections for a couple flaws in my initial work.

PAUL LASKA *is a game programmer that has worked on franchises such as Transformers, Wall-E, and Spider-man at Savage Entertainment and Treyarch, and is currently working on his own project. He is an active member in the game development AltDevBlogADay community and consults for game development companies. Paul can be found online at Twitter @paul_laska and LinkedIn www.linkedin.com/in/paullaska.*

THE 2ND ANNUAL

game developers CHOICE online awards

The Game Developers Choice Online Awards is the premier award ceremony for peer-recognition in the connected games industry. Taking place annually during GDC Online, the Choice Online Awards recognize and celebrate the creativity, ingenuity and innovation of the finest online developers and games created in the last year.

AWARDS ARE PRESENTED IN THE FOLLOWING CATEGORIES:

2011 AWARD CATEGORIES

- Audience Award
- Best Audio for an Online Game
- Best Community Relations
- Best Live Game
- Best New Online Game
- Best Online Game Design
- Best Online Visual Arts
- Best Online Technology
- Best Social Network Game
- Online Innovation Award

SPECIAL AWARDS CATEGORIES

- Online Game Legend
- Hall of Fame

Award finalists will be announced in August.

Stay updated at www.gdconlineawards.com

PRESENTED BY

GameDevelopers Conference

PRODUCED AND HOSTED BY

GAMASUTRA
The Art & Business of Making Games

gd GAME DEVELOPER MAGAZINE

UBM TechWeb

UNDERMINING ACHIEVEMENTS

THE REAL REASON GAMERS LOVE THEM—AND DEVELOPERS FEAR THEM

Why are you reading this? Hoping for a tip or two you can apply to your game? Trying to better understand your customers? Looking for proof that "cheevos" are destroying video games? Just needed some reading material while you wait for an appointment? What are you hoping to get out of it? More importantly, how would your motivations change if I paid you to keep reading? I'm not going to ... does that make you more or less likely to continue to the next paragraph?

Likely enough, apparently, because here you are. When Microsoft first introduced Achievements with the launch of the Xbox 360, they were met with a combination of confusion and derision. Who cares? They're meaningless, right? Now achievements of some sort are on almost every platform: PlayStation Network, Steam, World of Warcraft, and even your phone. They are an inescapable aspect of modern gaming. Clearly they mean something to someone, but can we harness their power, or are they destined to turn on their creators, reducing every game designer to a digital drug dealer and every game to a Skinner Box?

BEHAVIORAL BACKLASH

» First, let me lay out the "nightmare scenario" for how platform achievements might undermine the entire industry, based on a series of classic psychology experiments performed by Dr. Edward Deci as far back as 1969 (see Chris Hecker and Edward Deci in References). Deci maintains that humans are motivated to engage in an activity because it fulfills an unmet desire, usually for autonomy, competence, or relationships. Unfortunately, these implicit motivations can be replaced by more explicit rewards, ultimately demotivating the activity they intended to promote.

In Deci's experiments, two groups of participants are given an intrinsically rewarding task. The activity doesn't matter—it could be solving puzzles, drawing pictures, reading books, or even playing games—as long as it can be enjoyed for its own sake. Usually it fulfills the need to demonstrate competence, to test one's abilities and effectiveness. Unsurprisingly, most people appreciate the opportunity to show off in front of an audience of scientific observers.

On the second day, one of the groups was given a contingent reward for their task—a couple dollars, coupons for free pizza, whatever was lying around the lab—which acted as a secondary motivation for them, but not for the unpaid group. At this point, the paid group's desire to look smart and competent was replaced for the paid group by a more tangible need for financial gain, or pepperoni pizza, or a surplus lab coat.

On the final day, neither group was paid to solve puzzles. The participants that had never been paid usually worked just as hard as the previous days, seeking to show their competence as before. But the group that had been paid felt like the task is less rewarding—in fact, it was less rewarding—and this demotivating loss caused some of them to abandon the task. The loss of the contingent reward was more acute than their memory of their original intrinsic enjoyment.

The lesson to game developers seems clear. If a player is engaged in an intrinsically rewarding task, like playing a game, and we replace that intrinsic fun with mountains of points and piles of trophies, then they will react the same way as the test subjects: they'll abandon video games!

EXTRINSIC EXAMINATION

» Part of the difficulty of interpreting the results of these experiments lies in understanding the difference between intrinsic and extrinsic, especially since intrinsic has positive connotations and we are inherently suspicious of extrinsic influence.

First of all, extrinsic does not mean "external to the player." All rewards come from outside ourselves. If we already possess them, then they cannot, by definition, be desired. Extrinsic doesn't mean "explicitly promised," either. Telling someone that they will enjoy an activity doesn't reduce their enjoyment, and neither does noticing it afterward.

Extrinsic doesn't mean "materially beneficial or useful." Every need is directed toward accomplishing some goal, and every reward furthers some end, or we wouldn't value it. Extrinsic certainly does not mean "bribery" or "added with an immoral ulterior motive." In fact, the reasons for the offering of a reward are irrelevant to our appreciation of it. Fun has no providence.

An extrinsic reward is simply one that is "separable" or "not part of the essential nature" of the activity being rewarded. Since making money is a separable component of solving a puzzle, it can be removed, and suddenly puzzle-solving fails to meet the money-making need. At some point, parents will stop buying their children pizza for reading books, because eating pizza is not part of the essential nature of reading. This lack of reliability of the secondary, contingent reward is the true source of the demotivating effect. If the test subjects were tasked with baking a delicious pizza, and then allowed to eat the results of their handiwork, then there would be no problem, because eating a pizza is part of the essential nature of baking a pizza.

Every single Xbox 360 game is required to have Achievements. Every PS3 game is required to include Trophies. They are inseparable from the activity of playing games on those consoles. Platform achievements are no longer an extrinsic reward with the corresponding risk of removal: they are part of the essential nature of gaming on non-Nintendo consoles. (And increasingly a part of PC and mobile gaming as well.) They aren't going anywhere, which means the game industry, as a whole, is not at risk of suffering the catastrophic demotivating effects described in the psychological research. If a player's needs are met by demonstrating competence, games will continue to be appealing. Even if her needs are met by collecting and investing in Achievement Points, games will still continue to be appealing, as we have an unlimited amount to distribute.

Well ... actually, the fact that platform holders have an infinite number of achievements to dispense, is the real nightmare scenario for game developers, because we don't own them!

ACKNOWLEDGING ACHIEVEMENTS

» Achievements satisfy a very important human need to demonstrate competence; to do well according to a standard of excellence (see Andrew Elliot in References). Achievements provide an objective measure—however flawed

and arbitrary—of our ability to successfully play games. Even more than that, we seek social approval, and an "official" recognition of our accomplishments. This validates us and allows us to believe that we are making a worthwhile contribution to our community. Achievements may not have material value, but they are not meaningless. Denigrating them as "e-peen" swaggering mischaracterizes the near-universal aspirations they fulfill.

Achievements are particularly well designed as reinforcing rewards because we can anticipate them. Even before a game is on the shelves, its list of achievements is available on the internet, and a player can expect that if they meet the requirements, the achievement will be unlocked and the points awarded. There is no random chance, no potential for missing out. That is why there is a trend to award all the achievements over the course of a campaign, where every player can get them, instead of in multiplayer or for defeating exceptional challenges. A single game cannot provide that level of dependability.

The way achievements are allocated allows players to be efficient with their time by selectively choosing which to attempt, especially in a new game when the easy ones are available. The more time spent with a game, the harder the remaining challenges become. At some point it is going to be more efficient to quit the current game and buy another one, if achievements are the primary goal. Even a game that is worth playing to completion must eventually run out of its allotted total and stop meeting the needs served by achievements. Maximum efficiency drives players to purchase as many new games as they can afford.

Achievements are also essential to their platform, and are even required for certification. They have become such a fundamental aspect of games that even the most momentous event or amazing accomplishment seems less important without a corresponding achievement. Rewards that are inextricably linked to playing on a platform are more motivating than those only found in a specific game.

Finally, achievements are platform exclusive; they cannot (at present) be found anywhere else or liquidated and taken to another service. They are unlimited—new points can be added at no cost to the platform holder—but they maintain their value because there is an unbreakable ratio of achievements per game. This exclusivity ties the fulfillment of the player's needs to a specific source, increasing its power to drive behavior. And that source is the platform, not the game's developer.

In so many ways, achievements are the ultimate reward for investment—and that is the problem. The reason so many developers are uncomfortable with achievements is because they know that nothing confined to a single game can compete with a permanent, global, well-publicized achievement system. The need for validation has replaced whatever aspirations their game was meeting, and now it has been reduced to a vehicle for achievement hunting. Their game and the rewards it offers have become the extrinsic, separable part. Achievements are not a threat to games as a whole, but they are certainly a challenge to the lasting appeal of any one game.

DEVELOPER DECISIONS

» So now we are faced with an opportunity and a risk. How can we take advantage of the built-in appeal that achievements offer without reinforcing the same draw that entices players to move on? One answer is to exploit achievements by offering them for as little effort as possible. A cynical path, to be sure, but leaning hard on achievements can give any game —even a bad one—some value to its players. How many games have sold more copies than they deserved because reviews cited "easy achievements?"

Assuming your game aspires to be more than a trophy dispenser, the most common choice is to embrace achievements and try to use them to encourage players to fully enjoy the experience, as outlined below.

* Scale achievements to match the content. The player shouldn't run out of achievements before they have seen all the game has to offer.
* Pace achievements regularly and provide clear requirements. Arrange the rollout of achievements so the player gets a steady stream of rewards that are straightforward to earn. Awarding them too slowly may encourage a player to seek easier points in another game.
* Don't put achievements behind skill or investment barriers. Every player should be able to collect all the rewards, simply for playing normally.
* Reward dabbling in different gameplay modes. Use easy rewards to draw players toward content they might otherwise skip. But don't expect them to invest significant effort to learn entirely new gameplay modes.
* Focus on meeting the players need to demonstrate competence. This is the need that achievements fulfill, so you must align with them. If your game doesn't feature obvious challenges, then players will find the achievements unintuitive.

Approached in this way, achievements can bring players in and drive them through your content, but they will inevitably be swept on by that same powerful tide. The final, most ambitious option is to recapture your players by offering them something better:

* Meet needs other than competence or validation. You can't go toe to toe with achievements—they own competence and validation—so pick a different path. Focus on offering the player more autonomy or recognizing their creativity.
* Develop a community for your game. Achievements are powerful because they are publicized, but they are shown to a largely impersonal audience. The connections between members of a tightly knit community provide a more significant and personal validation.
* Don't add achievements to your existing rewards. Remember, achievements replace more subtle rewards when they happen simultaneously, so if your game is good at rewarding something, don't stomp on it. For example, if your reward for completing a mission is a great story moment, don't pop up an unlock notification during the cutscene!
* Add your rewards to existing achievements. Try to steal the thunder from existing achievements by linking them to one of your rewards, like unlocking new weaponry or receiving a mountain of in-game currency.
* Reward experimental or skillful play. Since your goal is to invest players in your game, scatter achievements at irregular intervals and behind difficult challenges. You will lose some players, but those that remain will be more excited that they overcame your challenge than for the achievements that originally motivated them.

As you can see, these techniques for replacing achievements are not compatible with those for embracing them, but the result of successfully superseding platform achievements will be a loyal and evangelical community.

JAIME GRIESEMER *was a game designer on Bungie's original HALO trilogy and is now working at Sucker Punch Productions. Read more at thetipofthesphere.com or follow him on Twitter @tipofthesphere.*

references

Hecker, Chris *Achievements Considered Harmful?* Game Developer Conference. San Francisco, CA. March 2010.

Deci, Edward. Flaste, Richard. *Why We Do What We Do: Understanding Self-Motivation*. Penguin, 1996.

Elliot, Andrew. Dweck, Carol. *Handbook of Competence and Motivation*. The Guildford Press, 2005.

ALL IN THE FAMILY

"GAME ARTIST" DOESN'T MEAN WHAT IT USED TO

IF YOU MISSED THIS YEAR'S GDC, YOU ALSO MISSED A REALLY INTERESTING moment in the ongoing evolution of the game business. The game design track always features a "Rant Session"—a chance for developers to blow off some steam and take public stands on controversial topics. This year's rant was titled "No Freakin' Respect! Social Game Developers Rant Back." (See References.) It was theoretically about design, but really it became a forum for social game developers to push back against dismissive or derisive attitudes from AAA developers. The mere existence of such a forum points out some intriguing changes in today's game business.

Social games, even though they are only a few years old, are expected to bring in around $2 billion next year in the US. That might seem small compared to the $18 billion or so from more familiar console and PC titles, but social and casual games are booming, while AAA development is treading water. The social game market expanded by a whopping 66 percent last year, while the traditional market shrank by nearly a third. Social and casual gaming has a lot of momentum right now, while the core games market is in a holding pattern.

Popcap's PEGGLE.

The contrast is pretty obvious to investors, who are firehosing money into social game development. Disney bought Playdom for more than half a billion dollars, and if the company hits performance targets, the total payout could be about as big as EA's purchase of BioWare/Pandemic a few years ago. But that's small change: FARMVILLE creator Zynga is now reportedly worth more than Electronic Arts. And as this article was going to press, EA just bought PopCap for $750 million.

This all makes for some pretty heady times. But even us poor working stiffs who aren't waiting for our new Lamborghinis are going to be affected by this sudden and massive change in the structure of the game business. It's hard to know exactly what the future will look like; after all, the MMO boom proved conclusively that big investments don't always translate into big successes. Even so, it's hard to imagine that the game industry of three or four years from now will be quite the same clubby, AAA-centric world we've known for so long. For artists, in particular, the rise of Casual, Social, and Mobile (CSM) gaming will upend a lot of things we thought we knew about our professional identities.

DIFFERENT STROKES

» CSM game studios don't match the standard game studio template most of us know. The teams tend to be much smaller and more fluid. Development cycles are measured in months rather than years, and budgets usually hit six digits, rather than eight. The skillsets for CSM artists tend to be different as well. High-poly modeling, 3D character animation, and shader work is rare, while graphic art stylings and traditional animation are in high demand.

Perhaps more importantly from the artist's perspective, CSM studios are driven by gameplay rather than content. A lot of big studios like to talk about how "gameplay is king," but in CSM studios the line between game design and art is often very fluid. With lower burn rates and looser tech requirements, many CSM studios expect their artists to actively shape their gameplay. As Popcap's John Vechey said in an interview with the London Telegraph:

"An artist on a PopCap title is never just an artist, or a programmer just a programmer, all are required to contribute to and have a vision for the game and what makes it fun."

That's not something a lot of traditional game companies can say anymore. It sounds a lot more like what our parents and friends think we do than the actual reality of toiling way in the midst of a $30 million project. It seems pretty attractive to the poor drone toiling away on a long Excel sheet of breakable crates. It's also the reason why a lot of veteran game artists are willing to walk away from the prestige of big-budget, as-seen-on-TV AAA games to work on CSM titles: It's a lot like the way games used to be before the age of next-gen bloat.

SQUARE PEGS

» There's another obvious difference between CSM games and the core business: demographics. If you're a GDC veteran, the change is pretty apparent just walking the hallways. Game studios have traditionally looked a lot like their "core gamer" audiences: young, white, and male. The audiences for CSM games are a lot more diverse, and so are the artists.

Some of this just reflects the gradual mainstreaming of gamer culture; now that pretty much everyone under the age of 30 has some experience with games, it's not surprising that you see more kinds of faces around

the office. The diversity of subjects and styles in casual and social games also attracts new artists who don't care about traditional games' focus on science fiction, fantasy, and the military. Plus, the CSM gaming world is a lot more geographically diverse than the core business. Take a look at GameDevMap.com, and you'll notice CSM studios in all sorts of places that are far from the centers of AAA power in California, Washington, and Texas. This, too, helps attract a wide range of artists with different tastes, styles, and outlooks.

All of this means that the stereotypes that go with the phrase "game artist" are changing rapidly. You're going to meet a lot of folks at your local IGDA chapter who won't chuckle at your Leeroy Jenkins T-shirt, or admire the fact that you do, in fact, haz Recon. You might get a blank stare if you tell them that all their base are belong to you. Our little club has gotten a lot bigger all of a sudden.

In most ways, this new diversity will be good for us; it'll help us break out of the stylistic rut we've been in since the era of near-photorealism that dawned with the current console generation. It doesn't hurt that small teams and short dev cycles let CSM games take a lot more risks than big, ponderous AAA projects; once somebody's scored a hit with a game like LIMBO, it's a lot easier to pitch a similarly risky-style project to your publisher. A wider set of perspectives can help us shake off some of our clichés, and it might also help us avoid some cross-cultural embarrassments like RESIDENT EVIL 5's white-heroes-shooting-black-zombies controversy.

There will, of course, be grumbling too. A lot of developers love the old school frat-house atmosphere and won't be happy to give it up. With a broader range of folks in the break room, you will have to be a little more corporate about your behavior. The language and swagger learned while deathmatching against the over-caffeinated and under-adjusted can land you in serious trouble with HR.

Even without the Employee Handbook issues, the gradual broadening of the game business brings with it a culture clash that will be hard to ignore. It's easy to snicker about the industry's genre obsessions, but for many working artists, those giant stomping mechs, badass spec-ops warriors, and buxom elf-maidens are really the reason they got into games. Those folks cannot be bribed to work on the next iteration of OH MY DOLLZ, not for all the credits on Facebook. As with any big, complex cultural shift, there's no one single story about how these changes will play out. Like the rest of the world, we'll muddle along in our new, more diverse reality. Some of us will find it less cozy than what we've known, and others will come out of hiding and cry "I'll never have to make another goddamn suit of power armor again! Hallelujah!"

THE OFFICE

» One thing is for sure, though: the ongoing re-definition of the term "game artist" will have serious impacts on your career.

For example, if you're a student, you're probably studying high-poly modeling and character design skills. These make great portfolio pieces when you're trying to land a spot on the next installment of GEARS, but they may not be as much help if all the actual jobs are for 2D Flash games. You need to think carefully whether you want to fight hard for one of those increasingly scarce AAA jobs or go where all the hot money is heading. Your chances of retiring early on your bonus as modeler #37 on a team of 400 making World of Call of Grand Theft Auto are pretty slim. Perhaps you want to swing for the fences with a scrappy little web game startup? If so, you'd better add some Flash, graphic design, and 2D animation to your repertoire. Naturally, the choice will depend on your taste, your talents, and your ambitions; however, there is an important new variable you need to consider, which is what, exactly, you mean when you say "I want to work in games."

If, on the other hand, you're a veteran with some hard-won skills and a few titles under your belt, you may worry that all this action is passing you by. The skills you've worked so hard to acquire may seem undervalued in the casual space. Veterans already know the danger of being professionally stereotyped: once you've shipped a couple of racing games, it's hard to change course and decide to make KINECTIMALS. Add in really radical differences of platform and technology—if you wanted to jump from GRAN TURISMO to, say, CALL OF BIEBER—and you'll have to work pretty hard to prove the relevance of your skills and experience.

FUTURAMA

» Over time, though, the boundary between AAA and CSM games will become more porous. While most of today's CSM games look pretty low tech to a jaded AAA developer, that will change as the space matures and gets more competitive. It's inevitable that some titles will try to stand out from their crowded fields with graphics, which means that a lot of casual developers will need to tap the skills of artists from the core games business. You can already see this process at work in mobile games, where the last six months have seen some really incredible graphics in games like REAL RACING and INFINITY BLADE.

Web and social media games haven't gone for lavish production values to the same degree, but that has a lot to do with the bandwidth limitations of browser-based gaming. Cloud gaming companies like Gaikai think they can break the bandwidth barrier by streaming games directly from their own servers, allowing even your parents' creaky old PC to play high-powered modern games inside a browser. If this works as promised, the graphics arms race will hit the casual-social space with a vengeance. That's good news for AAA artists who could use some cheering up after several years of doom and gloom in big-name development.

Even if the graphics armageddon doesn't hit tomorrow, AAA artists have a lot more to offer than just their skills with high-poly models or fancy shaders. Big projects have huge asset bases and complex toolchains. Keeping these beasts fed is a difficult job for the tech artists who build the tools and the leads who keep herding things along. As the new frontiers of gaming become big businesses, they're going to need hardcore production experience to make sure they don't repeat some of the mistakes that the core business has gradually learned to avoid. A lot of agile little studios would benefit from the wisdom of vets who know how important unsexy things like asset tracking, source control, and debugging skills are.

Predicting the future is really hard. If this column had been written in 2006, it might easily have gone out on a limb to predict that by 2011 we'd all be laboring on 2,000-person MMO teams or that all of our jobs would have already been shipped to China. Reality is a lot messier, but it's also always changing around us, and we have to keep our eyes open to deal with it. There's one thing we can predict with certainty: it's going to be hard to get by on just attitude. Maybe you did models for a CALL OF DUTY episode, or maybe you animated some awesome melee moves for Nathan Drake. Great! But when interviewing for a job with some group of twenty-somethings whose point-and-click Flash game has 35 million users, you'd better show—as the phrase goes—some freaking respect.

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

references

No Freakin' Respect! Social Game Developers Rant Back at GDC Vault
http://gdcvault.com/play/1014539/No-Freaking-Respect-Social-Game


take
control
of your
future
www.gamasutra.com
the art and business of making games
UBM
TechWeb

CROWD CONTROL

THE ADVENT OF CROWDSOURCING IN GAME AUDIO CREATION

BUZZWORDS ABOUND IN THE TECHNOLOGY sector, and from cloud computing to transmedia convergence, the concepts behind these buzzwords are at the forefront of the ever-evolving game industry. New styles of project management and experimental team building are being driven by a range of desires to lower development costs, maximize brand loyalty and exposure, and spread out development into a compartmentalized global workforce capable of a 24-hour work day.

Over the last five years, massive outsourcing companies have developed in foreign countries that are frequently working with game companies, particularly in the disciplines of art and animation. Aside from cheap Eastern European orchestras, audio never became a discipline that saw much of a shift to low-cost outsourcing in developing nations. European, Japanese, and North American contractors remain the vast majority of content creators handling the audio needs for the industry.

Recently though, outsourcing has begun to flirt with a new scattershot approach known as crowdsourcing, and its influence is already being felt in professional audio circles.

FACIN' THE CROWD

» Crowdsourcing is the process of presenting a large group of external users with a defined task or set of tasks. Those users then work toward the defined goal either collaboratively or as a large pool of individuals from which the originator of the task can select the input they'd like to use. Examples of crowdsourced work include everything from Wikipedia articles to the t-shirts of Threadless.com to logo design competitions for companies like Pepsi, Twitter, and Chiquita—yes, the banana people.

While an ever-growing list of industries toy with their first forays into the crowded space of crowdsourcing, its use in the video game industry isn't anything new. One of its most common uses in the game industry has been around for over a decade. In public betas, a group of unpaid and unaffiliated users are invited in to play the game, stress-test its servers, and offer feedback or report bugs.

Crowdsourced work can be tackled across the globe without the need for dedicated management, and with little impact on dedicated company resources. Of course, crowdsourcing requires a crowd, and if a company can't attract people willing to give their time or energy, the work goes unfinished. Two of the defining qualities of crowd creation—the content creation form of crowdsourcing—can further detract from the overall experience.

For one thing, those looking to have work crowdsourced should note that the quality of the work can vary greatly, since there are no restrictions on involvement, so you will get work from professionals and amateurs alike. The second point is that for those looking to do crowdsourced work, pay is not guaranteed. Many crowdsourced projects are effectively done as either volunteer work or simply for the opportunity of increased exposure. Submitting work to crowd creation projects does not mean that the work will be selected, and depending on the number of submissions, it may not even be reviewed.

ATTRACTING A CROWD

» Audio has been slow to develop as a crowdsourced discipline, perhaps due to the high bar of entry related to the gear, software, and training inherent to the field. Two companies are aiming to change that.

AudioDraft is a new company out of Finland that aims to establish itself as a hub of crowdsourced music and sound design specifically targeting game and video production companies. Through AudioDraft.com, those looking for game audio content establish what is referred to as a "contest." The contest outlines the brief describing the desired work, a timeline the work must be completed within, the amount of compensation, and a description of the kind of license the winner of the contest will be entering into with the contest holder. Once an entry has been submitted for a contest, individual entries can be listened to and voted on with a Facebook-esque "Like" button, though the winner of a contest is selected solely by the contest holder.

ILLUSTRATION BY KELSEY KRAUS

Minimum Noise is a Danish company founded in 2008 that aims to take crowdsourcing to the realm of music licensing. Much like AudioDraft, the company's web site is set up for people or companies to create "projects." Projects are then opened for submissions, and winning submissions are selected by project owners. While less focused on games than AudioDraft, Minimum Noise has had an increased number of independent game developers posting projects over the last year. However, there are currently no projects available, with the most recent project having finished within the last two months.

Both sites are similar in their approach, design, and goals, and make it very easy for those looking to crowdsource audio to find a ready and willing crowd. The nature of the sites also makes it easy for those with little or no previous game experience to connect with developers and begin building a credit list. Unlike other crowdsourced ventures, both sites are structured around the notion of paying for winning submissions. Unfortunately for content creators, pay is significantly lower than standard accepted industry norms. Creators can expect anything from $1,000 for a 30-second jingle, to game engine company Unity Technologies offering $25 iTunes gift cards in return for two minutes' worth of orchestral game underscore.

Crowdsourcing relies on making work available to the masses, and the masses being attracted to the available work. Whether this will ultimately mark an archetypal shift in the audio production and networking pipeline is still to be seen, but the crowds are gathering.

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

Join us at the **Game Developers Conference Europe™ 2011**, August 15-17, for three days of over 100 sessions, as we cover a comprehensive selection of European game development topics taught by leading domestic and international industry experts.

SUMMITS
MONDAY–WEDNESDAY

- Independent Games
- Smartphone & Tablet Games
- Social Games
- Community Management

MAIN CONFERENCE SESSIONS
WEDNESDAY–FRIDAY

- Business & Marketing
- Game Design
- Production
- Programming
- Visual Arts

EXPO FLOOR
MONDAY–TUESDAY

Explore the latest in game innovations and talk with industry experts.

GDC EUROPE VIP LOUNGE
MONDAY–WEDNESDAY

Broaden your networks and connect with fellow game developers and professionals from the European games industry.

ONLINE REGISTRATION ENDS AUGUST 9
REGISTER NOW!

GDC 11 Europe

Game Developers Conference Europe™

August 15 – 17, 2011 | Congress-Centrum Ost Koelnmesse | Cologne, Germany

Visit **www.GDCEurope.com** for more information.

*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 August 9, 2011 at 21:59 UTC.

Supported by

European Games Developer Federation

BIU

gamescom

UBM TechWeb

TURN A SQUARE

AN INTERVIEW WITH SQUARE ENIX'S MIKE FISCHER


WAKFU

IN JULY 2010, MIKE FISCHER was named the new president and CEO of Square Enix in the U.S., overseeing the business of localizing content from the Japanese offices and managing newly acquired Eidos properties like TOMB RAIDER and HITMAN. In addition, the company is looking to expand further into partnered publishing, as it is doing with WAKFU, a French-developed MMO that already has an established fanbase in Europe. Square Enix will be publishing the game in North America, while developer Ankama Studio will retain the rights for the rest of the world.

Game Developer discusses Square Enix's changing business, alongside Fischer's thoughts on the future, as framed by the North American publishing of WAKFU.

Christian Nutt: It's been almost a year now; what would you say you have learned during your time as Square Enix's North American president?

Mike Fischer: I think the one thing I've learned is that there is no such thing as a stable period in our industry. Our industry has been going through a period of transition for the entire 20 years I've been in it, so the first thing that I learned is that that's never going to change. I think that it's really not about getting your business to a point where everything's fine; it's about creating a culture where you're constantly driving the change and staying at the front of what the trends are.

CN: Can you give me some examples of things you see the company doing now that speak to that?

MF: I think you've seen announcements already about our continued focus on a more net-based business, and I am in a very unique position in my role here as the head of the U.S. organization because I have the great [content] coming out of the studios in Tokyo, but I also now, after the Eidos acquisition, have this lineup of Western games. That leaves me and my organization free to really pursue a lot of the more leading-edge opportunities in online and social games, and I'm obviously looking at not-massive games on an FINAL FANTASY XIV or FFXI scale but, you know, smaller, more agile opportunities. One of those is WAKFU [from Ankama Studio in France].

One nice thing about [WAKFU] is that it takes some of the strengths we have in terms of knowledge of our audience and the RPG world, but it's also helping us leverage it into a whole new place and partner that is very well entrenched there already. So that's the type of one-plus-one-equals-three synergy that I'm looking for in the projects that we either pick up or we drive.

CN: Is that free-to-play game or is it a subscription game?

MF: It's kind of a hybrid free-to-play [game]. So, there's a subscription component to it but there's also a free-to-play, microtransaction component of it as well.

CN: Are microtransactions something that interests you particularly from a Square Enix perspective?

MF: It does, but I don't believe that there's going to be any one single dominant model in the years ahead. I think if there's one thing that I see now, it's a diversification, and we're a big enough company so that we couldn't put all of our bets all on one business model if we wanted to anyway. And for us, it's not about picking winners in the different platforms; it's about having great IP and fitting that to the best platform for it. We want all of the platforms to succeed.

It's funny, I remember people saying, "I'm not sure if the industry can support three consoles." It's always been two consoles, two consoles, two consoles, I'm not sure if we can support three, and now I hear people worried that we might not have three big console platforms at one point. So it's kind of funny, you know, when it looks like the number of platforms are going [to] grow, people are going to worry. If there's speculation the number of consoles might shrink, you've got everybody worried. The fact of the matter is it's not the number of platforms that's important; it's the health of those platforms and the growth of the market. I don't see what's happening in the browser-based, free-to-play, or social market cannibalizing the overall industry sales, what I see is [it] adding to it.

CN: Yeah, absolutely. I mean, you've now got a situation where we have three consoles, two or three handhelds, smartphones, Facebook, web downloadable, browser-based, there's a tremendous variety.

MF: I think that to some degree there's a fallacy that some people are applying to the future of our industry based on what happened say, for example, to music. I'm not an expert on the music business, but there's a perception that the shift toward digital music downloads has hurt the overall growth of the recording industry, and let's just say for the sake of argument it has. I don't think that applies to games, because the switch to digital downloads for music didn't bring any more people into the "listening to music" market. Everybody was listening to music—everyone is still listening to music—but in the case of games, what this transition is bringing is a whole new audience and more ways to play. So in the past, if you had to sit down in front of your TV, now you can do it in front of your TV, your computer, your iPad in bed, your smartphone while you're outdoors, or your work PC while you're in the office and your boss isn't looking. So, I think the same trends that have threatened the other media are actually strengthening us.

GDC ONLINE 2011 DEBUTS FIRST SESSIONS FOR GAME NARRATIVE SUMMIT

\\\ GDC Online has revealed the initial lineup for the show's Game Narrative Summit, which features speakers from Eidos on DEUS EX: HUMAN REVOLUTION's branching narrative, Microsoft on creating transmedia giants, and MIT on crafting education-based ARGs, among others.

Now in its sixth year, the Game Narrative Summit—formerly the Game Writers Conference—once again returns to GDC Online to showcase leading speakers on the many facets of interactive storytelling, with sessions ranging from roundtable discussions to postmortems and more.

The initial sessions and lectures featured in the two-day Game Narrative Summit include highlights such as independent producer and writer Aaron Linde offering examples of successful transmedia properties in "*The New World: Case Studies in Transmedia Narrative Design*." Linde, former GEARS OF WAR community manager at Microsoft, will look at projects surrounding titles such as PORTAL 2 and HALO: REACH to "explore the potential of engaging players across multiple channels of communication, and why we, as storytellers, must embrace transmedia as an essential component of narrative development."

Scot Osterweil and Caitlin Feeley—who work at MIT and the MIT Education Arcade, respectively—will explain how ARGs can be used to create an exciting learning environment for students. The talk, "*Capturing Children's Attention and Imagination with Investigative Play*," will outline the ARG Vanished, which was designed to help middle school students learn about science while they solve a large-scale mystery. Geoffrey Long, producer of narrative design at Microsoft, will note what it takes to build an IP into a powerhouse franchise in "*From Story to Universe: 10 Best Practices for Transmedia Franchise Design*." Long will look at franchises outside of games, including *Lost*, *Buffy the Vampire Slayer*, and *Star Wars*, to explain why these IPs became some of the most popular in the world, and how game developers can build their IP to do the same.

Augmenting the Narrative Game Summit, GDC Online will also feature a one-day Game Writing Tutorial for beginning writers following the two-day Summit, as well as Summits on Smartphone & Tablet Games and Virtual Items.

For more information on GDC Online as the event takes shape, please visit the official GDC Online web site (www.gdconline.com), or subscribe to updates from the new GDC Online-specific news page via Twitter (@GDC_Official, @GDC_Online), Facebook, or RSS. GDC Online is owned and operated by UBM TechWeb, as is this magazine.

GDC Vault Reveals Most-Watched GDC 2011 Talks As Views Top 250,000

\\\ As views of GDC Vault's video, audio, and slides from GDC 2011 top 250,000, the site has detailed the most-viewed sessions from the March show, spanning a postmortem of DOOM through HALO: REACH and beyond.

The specially constructed web site archives multimedia from the numerous lectures, panels, and keynotes at the multiple Game Developers Conference shows yearly, and a number of each show's most popular talks are now available for free.

The sessions available on the GDC Vault spanning the last 15+ years have attracted 155,000 unique viewers in the last year, and the content from Game Developers Conference 2011 alone has attracted more than 262,000 views since mid-March.

GDC 2011's Classic Game Postmortem series has proven the most popular by far, with video of these seminal talks making up six of the show's top 10 most-viewed sessions.

These lectures featured various industry legends reflecting on their most seminal classics, including John Romero and Tom Hall on DOOM, Eric Chahi on OUT OF THIS WORLD/ANOTHER WORLD, Ron Gilbert on MANIAC MANSION, and more.

Other popular talks included "*I Shot You First: Networking the Gameplay of* HALO: REACH," featuring Bungie's David Aldridge on the studio's approach to online infrastructure, and "*Life and Death and Middle Pair: Go, Poker and the Sublime*," a talk featuring Area/Code's Frank Lantz on some of the oldest and most influential games in history.

The following are the top 10 most-viewed GDC 2011 video sessions on the GDC Vault, including current view counts:

Top 10 Most-Viewed Sessions From GDC 2011

1. "Classic Game Postmortem - DOOM," John Romero & Tom Hall
27,970 views

2. "Classic Game Postmortem - OUT OF THIS WORLD/ANOTHER WORLD," Eric Chahi
19,960 views

3. "Classic Game Postmortem - MANIAC MANSION," Ron Gilbert
14,420 views

4. "Classic Game Postmortem - PITFALL!," David Crane
10,530 views

5. "Classic Game Postmortem - PRINCE OF PERSIA," Jordan Mechner
8,601 views

6. "I Shot You First: Networking the Gameplay of HALO: REACH," David Aldridge
8,390 views

7. "1-Hour Video Game MBA," Charlie Cleveland
6,656 views

8. "Life and Death and Middle Pair: Go, Poker and the Sublime," Frank Lantz
6,020 views

9. "Classic Game Postmortem - BEJEWELED," Jason Kapalka
5,940 views

10. "Automated Level of Detail Generation for HALO: REACH," Xi Wang
5,560 views

Several talks from Game Developers Conference 2010 have also seen similarly impressive audiences on the GDC Vault. Brenda Brathwaite's "*Train (or How I Dumped Electricity and Learned to Love Design)*" and Zynga's "*Engineering Scalable Social Games*," for instance, have both surpassed 20,000 views each.

Some of the most popular titles from GDC Online 2010 include Professor Richard Bartle's "*MUD: Messrs Bartle and Trubshaw's Astonishing Contrivance*" and Brian Reynolds' "*Bears and Snakes! The Wild Frontier of Social Game Design*," both of which have topped 5,000 views apiece.

Besides the multitude of free videos and slides, full GDC Vault access is available to GDC 2011 All-Access Pass holders, speakers, and All-Access Pass buyers to other GDC events for the rest of 2011.

Individual Vault subscriptions not tied to All-Access passes are now available for a limited time via a Beta invite process—those interested in signing up to be invited in on a first come, first served basis should sign up on the GDC Vault web site (www.gdcvault.com).

In addition, game-related schools and development studios who sign up for GDC Vault Studio Subscriptions can receive access for their entire office or company. More information on this option is available via viewing an online demonstration on the official GDC Vault web site.

who went where

David Maisel, formerly chairman at film company Marvel Studios, has joined ANGRY BIRDS developer Rovio as a special advisor, where he will help the studio shape the future of its popular franchise.

Following the closure of HOMEFRONT developer Kaos Studios in June, studio creative director David Votypka has found a new home at Ubisoft's Red Storm Entertainment.

Yasumi Matsuno, creator of high-profile RPGs such as TACTICS OGRE, VAGRANT STORY, and FINAL FANTASY XII, has been hired by PROFESSOR LAYTON developer Level-5.

Former Playdom executive producer Ethan Fassett has been named the new senior vice president of product at mobile gaming toolmaker OpenFeint.

new studios

Six core team members of Seattle, WA-based FAT PRINCESS creator Titan Studios have formed a new company, Carbon Games, following Titan's closure earlier this year.

Two former Ubisoft staffers who have worked on franchises including Tom Clancy's ENDWAR and SPLINTER CELL DOUBLE AGENT have formed a new French mobile dev studio, The Game Bakers.

Staff members from TEST DRIVE UNLIMITED developer Eden Games have formed Blossom Minds, a new French independent studio that aims to release its first game in 2012.

Sulka Haro, the lead designer on popular virtual world Habbo Hotel, has left developer Sulake to co-found the new startup studio Makielab, which aims to combine 3D printing with web and mobile gaming.

Unity Technologies, has opened a new office in Stockholm, Sweden led by Erik Hemming and Erland Korner, both of whom previously worked at DICE on the BATTLEFIELD franchise.

GLOBAL WARFARE and KINGDOMS OF CAMELOT developer Kabam has opened a new European office, which will focus on localization and customer support for its international market.

No Sleep 'Til Burbank

KEN FINLAYSON JOINS INSOMNIAC GAMES IN LA

Ken Finlayson came from the world of film visual effects because he wanted to work on characters, a chance that movies weren't affording him. After several years at BioWare, he's made the jump to Insomniac's Burbank studio, to become lead character artist for the company that (at least in theory) never sleeps.

BRANDON SHEFFIELD: Why the move to Insomniac from BioWare?

KEN FINLAYSON: There were several motivations for the move. I had been at BioWare a little over five years and needed some new challenges. I had worked on all of the major BioWare IPs (in some cases several times) as well as on an IP that never saw the light of day. Art styles were very established, especially on games going into their third installment. The character art team was an extremely talented and experienced team that didn't require a lot of guidance or instruction. If I was going to continue to grow as an artist and grow as a leader, I was going to have to leave those calm and familiar waters and find new challenges. I found those challenges at Insomniac. They had a younger character art team that was being rebuilt and hired into. I could help steer it. It was also an extremely interesting time to come here. [There are] new firsts for the studio to tackle like going multi-platform and a major update that's in the works for our proprietary toolset that also needed input. There is a lot going on, and it is pretty hard to be bored.

BS: In 2005 you left the film industry to work on games. How did you find the switch?

KF: Going to EA Sports and into the games industry was the result of a few things. I had always wanted to make believable creatures and monsters. In the early 2000s, only a handful of VFX shops in the world were doing that type of work at a high level. The VFX shop that I was at was small. I mostly worked on visual effects and lighting. Any character-based work that came in the door was immediately pounced on by the senior artists so there was very little opportunity for me to create characters in film.

In the game industry, dozens of CG humans or creatures are made per game versus maybe one organic CG asset per film. 2005 was also the eve of "next-gen." EA was working on launch titles for the Xbox 360, and by the time I got there, EA Canada had brought in many very experienced, top film industry people to explore what the new hardware was capable of. So in spite of moving to the game industry, the target of photo-realism was still the same as it was in film. It was an exciting time to be there. I always joked that coming to EAC was like a vacation because I was no longer working on weekends or holidays, and the sun was still up when I went home at night.

Tools-wise, I was still using Maya and Photoshop at EA, but I had given up compositing software like Shake and tracking software like Boujou. EA had hired me because I was very proficient with ZBrush. Back then that was hard to find. I was one of two people doing Zbrush work at the whole studio, and because of that I had a unique job. Now, even the most junior character artist today is expected to know 3D sculpting software inside and out.

BS: What, in your mind, are the most important tenets of a believable game character? How do you sell it to the player?

KF: Knowing the character's personality early on is important. A believable character is the result of everyone on the development team knowing who that character is and acting in unison to deliver a consistent message. The caveat is that the character needs to deviate at times from predictable behavior to keep players on their toes, and so the character doesn't feel one dimensional. It's a paradigm shift to not treat the character as just another mesh, but as a personality that is the result of many pieces.

You need someone who knows and owns the character, and makes sure that a consistent message is being delivered from everyone. How the character looks (art), how the character talks and sounds (audio), how the character moves (animation), and what the character does (story) all support or hurt the character's believability.

Imperfections are very important. Someone with a robotic personality that responds to all situations the same way is boring and totally unbelievable and will get no emotional investment from players. Imperfection in the design like asymmetry in the face and wear in clothing—even imperfections in the motion like a limp, or a lisp all add greatly to the level of believability. People are far from perfect, so our characters should follow suit.

SHOOT INTO THE FUTURE...

LEAD MULTIPLAYER DESIGNER • MULTIPLAYER DESIGN SCRIPTER
SR. LEVEL DESIGNER • LEVEL BUILDERS • CONCEPT ARTIST
SR. ENVIRONMENT ARTIST • SR. TECHNICAL ARTIST
LIGHTING ARTIST • PC PLATFORM LEAD ENGINEER

APPLY @ HTTP://WWW.NEVERSOFT.COM/SITE/HIRING.HTML

PaperPlane

http://paperplane-game.com

IGF FINALIST PAPERPLANE AIMS TO CAPTURE THE SPIRIT OF CHILDHOOD INNOCENCE, ALLOWING PLAYERS TO GLIDE THROUGH THE COUNTRYSIDE AND UNCOVER WHAT IT HAS TO OFFER. AS THE GAME PROGRESSES, THE INITIALLY BARREN LANDSCAPE BECOMES POPULATED WITH SWING SETS, FARMLANDS, AND MORE, DEPENDING ON WHERE PLAYERS GUIDE THEIR SMALL PAPER AIRPLANE. HERE, THE TEAM FROM FRENCH GAME SCHOOL ENJMIN DISCUSSES THE INSPIRATION AND WORK THAT WENT INTO CREATING THIS ATMOSPHERIC TITLE.

***Tom Curtis:* How did you all come together to work on this project?**
PAPERPLANE was our second year student project in our master's degree program. For these projects, you have to submit a pitch to the school's board, and they select some of them. Then, you have to build a team around your project. There were three of us behind this pitch, and quickly we found six other teammates. We had six months to develop a prototype and we had to deliver three presentations during this time, with the final one being in front of a jury for the end of the project.

***TC:* How did you all decide upon the premise for the PAPERPLANE?**
The proposed pitch was not quite like the final prototype. At first, it was more based on the concept of Rube Goldberg machines, and you would use the paper airplane as a means of chaining actions together. We got a little lost with this concept during the first month or two of development, and after a few meetings with some of our teachers, we were able to find a different direction that suited all of us. The idea of using childhood memories came at around that moment, and it was from there that everything fell into place.

***TC:* The game heavily emphasizes life in the rural countryside. Did any of you grow up in such an environment?**
Most of us did, or at least had parents or grandparents who grew up in the countryside, and we all have different memories of this type of environment. When you're a child, it's such a rich and enjoyable setting. To get a good feeling of the countryside we did some "project hikes" with the whole team. We went around the city we lived in: Angoulême, France, which is where our school is. That was really fun to do and it created a bond among the team. Also, mountain bike rides and books by French author Jean Giono were major inspirations during development.

We didn't want to show a kitsch, adult-idealized vision of the rural countryside. It's not all flowers and cows there; we also feature things like a highway and a tractor, because these sorts of things can still seem incredibly fun in a child's eyes.

***TC:* What sort of inspiration did you draw from your own childhoods?**
The central tree house is really a thing we all had in common—it also speaks to every child. Everyone had one that was more or less like the one in the game. It's a wonderful feeling to be on top of everything, to be higher than your parents, to have a private space where they can't go. Also, it's really hard to find people who never made a paper airplane; every one of us did it at some point either as a child, as a parent, or as a grandparent.

***TC:* What were your influences when developing the game's visual style? The game's intro sequence in particular seems to draw some cues from impressionist paintings.**
We opted for a visual style close to hand drawing. Nevertheless, it's pretty hard to obtain a convincing NPR render, especially with Unity, so we decided to work with low-poly models and flat textures, soft light, colored shadows, and pastel tones. The intro scene is a pre-rendered sequence, so we tried to obtain a painted visual style with a little program wrote by one of our graphic designers.

***TC:* What were some of the major challenges you all faced during development?**
At first, we wanted to integrate an "origami editor" into the game, which would allow the player to fold their own paper airplane and even other shapes like boats or cranes. That was a major feature in the original pitch, but we had to drop it because of the technical and usability challenges involved. It was a hard decision to make, but it was for the greater good—you need to keep it simple to be able to focus on the little things.

Another stressful challenge was when we had to do an art change about a month before the final presentation of the project. We lost our direction along the course of development and were harshly reminded during a presentation that our game looked too realistic, and it didn't quite feel right. That was enough for us to review all of the textures we had done so far and do them again—we did not regret it.

***TC:* What was it like working with Unity as your game engine?**
Unity is great for students; you have all you need right away. You don't have to waste time developing your own engine. Of course, the engine is quite generic, so you can make pretty much any type of game you want. Unfortunately, that's also its worst drawback, meaning you have everything to make a game but not everything is as advanced as you would like. For example, the sound engine is good but you can't have interactive music easily. However, some missing features can be developed with the help of plug-ins.

The fact that JavaScript is implemented is also of great help for the designers, and when it comes to balancing and playtesting. Using that, a non-coding team member can easily tweak some game variable without bothering a programmer.

***TC:* Now that PAPERPLANE has been featured in the IGF Student Showcase, what are your plans moving forward? Do any of you have plans for future projects?**
The IGF draws a lot of attention, and it's really a great way to get contacts. We have been contacted by several people with different opportunities relating to this project, and we are excited about all of them so far. We all would like to work some more on the game, but once you finish school, it's hard to find time for side projects.

ADVERTISER INDEX

gd Game Developer (ISSN 1073-922X) is published monthly by United Business Media LLC, 303 Second Street, Suite 900 South, South Tower, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *gd Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *gd Game Developer* on any correspondence.

THE BEST VIDEO GAME STUDIO IN THE WORLD

WELCOME, NEW EMPLOYEE.

You have arrived.

That's right. You have arrived at your new job at *The Best Video Game Studio in the World*. All your hopes and dreams and ambitions in life have led up to this moment. Now you are here, and it is time to start the real work, the reason you came into existence: the work you will do for us.

As you might imagine, we do things a little differently here. This isn't what you're used to at those other, terrible, awful video game development studios under whose jack-boot regimes you have labored in the past. Our studio is nothing like those clueless suit-infested businesses that populate the lower depths. Up here, our studio is a sanctuary, an oasis for nursing talent—Valhalla, really, though that might sound a little arrogant.

Here, we have no schedules. No producers, no managers, no stockholders. Actually, we do have those, but they will be far away. Very, very far. You will never know they exist. We will keep you in a bubble. A happy, comfy bubble of pure ideas: design as you always wanted to do it, as you always wished for. The irritating constraints that always dogged you in the past—milestones, technical ability, other people—have been removed entirely.

Impossible, you say? But it is true, fantastically true! Don't strain yourself too much thinking about it. Instead, look around you and drink in the care with which we have constructed the most perfect, nurturing cradle ever devised for game developers, one designed from the ground up to coax, gently, those droplets of pure unadulterated genius from the intricate folds of your brain.

See how our expansive office's multiple cantilevered travertine balconies allow raw creativity to hover, unfettered, through the air. Each employee gets his or her own temperature-controlled levitating work surface. We have our own specially made dev kits (at great expense) that emit no distracting noise or skin-drying heat. These three straws next to your desk here allow you enjoy sips of fair trade artisanal coffee, single malt Scotch, or puréed cuts of premium Wagyu beef at any time you desire. Hot face towels are distributed seven times a day to all employees. These mouse pads are fashioned of gold leaf and Ultrasuede and are emblazoned with your monogram (small) and our logo (large). That knob is pure tungsten.

Next, let me introduce you to your personal manservant. We all get one here, of course. Your manservant will take care of all your needs for you, so that you—thank goodness! are freed up to concentrate on your singular art. For example, if you need to make a change to the codebase, simply describe the change you want to your manservant, and he will make the changes and commit them into the repository for you. If the change turns out to be bad, you can tell everyone it wasn't your idea and then fire the manservant. Don't worry about it! That's what they're there for—we can always get more.

Directing the work of the help in a sustained manner is liable to quickly exhaust the delicate instrumentation of your finely honed mind, we realize. To that end, we have a range of rejuvenating activities available to you right here on the premises. To your left there are the tennis courts, recently resurfaced with an abalone shell mother of pearl finish. Our assistant to the associate art director doesn't like the iridescence, though, so it may be removed tomorrow. To your right you'll find the five-story climbing wall located inside what we call the Grand Rationalization Hall. The facility comes with its own personal trainers at your disposal—many of them are former game designers themselves! And nothing is more soothing on knotted shoulders than our Olympic-size hot tub, straight ahead. Its healing waters are imbued with hand-milled Himalayan sea salt and homemade egg noodles.

Now, you would be forgiven for thinking, "Wait a second—have I died and gone to game development heaven?" But no, you are very much alive. Don't act so surprised. You've known all these years that you were the Best Game Developer in the World, haven't you? So it was only a matter of time that you would find your way to us, *The Best Video Game Studio in the World*. Yes, it was preordained, written in texts so hallow their gilt pages sing with an angel's voice as they are turned. But enough of that for now. We'll save the religious part of the new employee process for later.

If you were the cynical sort—and I would not blame you for being one, after having survived the bleak reaches of game development out there in the big, scary world—you might ask, "What's the catch?" And I would be obliged to answer you truthfully: there is no catch. I repeat firmly: there is no catch. None. We expect you only to be yourself, where "yourself" is that apotheosis of brilliance that you always knew you were. And we know that, once you have applied that mind, that beautiful mind, to the vague problem statements that may happen to float your way, you will come away content, knowing that your work is never really done. Think not of ship dates, think not of technology or resources. Just reconfigure symbols in different pleasing ways—until we tell you to stop, that is.

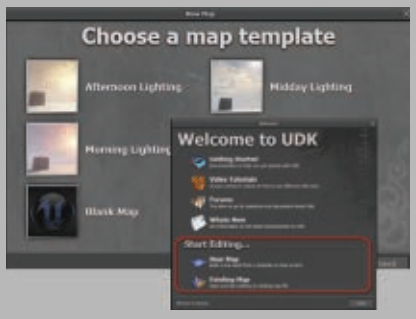
By the way. Anything you *can't* do with the world's greatest studio at your full disposal is your own freaking fault. We hope you understand that.

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

NEW FEATURES, ROBUST UPDATES TO UDK MAKE WAVES

Every month, Epic Games releases a new update to the free Unreal Development Kit (UDK) downloaded by more than 800,000 developers around the world. With the same triple-A toolset as the award-winning Unreal Engine 3, UDK is popular among professionals, fledgling developers, and educators alike for its industry-leading capabilities and flexibility.

And while every month brings about new improvements, the June 2011 release is particularly exciting for UDK users. The release introduces a series of robust updates as well as new technology integration that provides access to Simplygon.



The first major update is the brand new foliage system (the result of which is pictured above this article). The foliage editor is amazing in that it allows level designers to 'paint' foliage within their levels as easily as drawing a brush over a canvas. Foliage instance types can now specify a landscape layer for weighting, and foliage painting can be used for altitude-based painting, as well. The new foliage painting tool, which is focused on speed and quality, enables developers to quickly paint foliage and decorations such as grass, small rocks and bushes onto Landscape, static meshes or BSP. UDK is all about efficiently building games at triple-A quality.

Now, with our new foliage editor, designers can get the end result they need without investing a huge amount of time placing each individual object.

Other improvements in the June release echo this sentiment. With the new Unreal Kismet debugger designers can visualize the flow of their Kismet sequences while their game is running. Designers can also step through their sequences one frame at a time and set breakpoints. These features can help identify exactly what parts of a sequence are currently executing in the game and for what reasons.

The addition of fully customizable map templates (see image below), for example, lets designers immediately configure lighting for day, night, dawn, or sunset by dragging the appropriate thumbnail from the Unreal Content Browser into a level to populate settings and achieve the desired lighting effect. This gives developers a wide range of possibilities for more complicated lighting projects and saves a great deal of time.

There's now a single editor and game content tree for both PC games and mobile games. This allows developers to share gameplay logic and assets between PC and mobile more easily. We're making it much easier to build cross-platform games, while still allowing developers to accurately preview graphics and gameplay with a single button click.

Furthermore, Donya Labs is now a member of the Integrated Partners Program. Donya Labs' Simplygon is used to automatically generate game-ready Level of Detail models (LODs) for a specific pixel resolution.

Simplygon, which is known to save developers hundreds to thousands of hours of manual art adjustment, provides high-quality mesh reduction without having to leave the Unreal Editor. Developers can quickly simplify meshes, generate LODs, and immediately see the results in their maps. Simplygon is especially useful for converting high-end PC assets for deployment on mobile devices, which is a boon to Unreal Engine 3's cross-platform strengths.

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 five times along with entry into the Hall of Fame. UE3 has won three 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 on Twitter.

WWW.UNREAL.COM

UPCOMING EPIC ATTENDED EVENTS

GDC Europe	gamescom	GDC Online	Montreal Int'l Game Summit
Cologne, Germany	Cologne, Germany	Austin, TX	Montreal, Quebec
August 15-17, 2011	August 17-21, 2011	October 10-13, 2011	November 1-2, 2011

Please email: mark.rein@epicgames.com for appointments.

THERE'S NOTHING LIKE BEING A RAD GAME DEVELOPER

BINK VIDEO

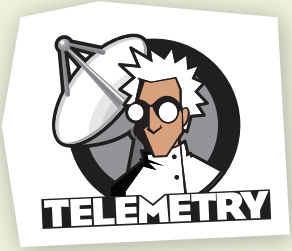
It feels like you have SUPER POWERS when you use Bink, our amazing, super =FAST video and audio codec.

You'll **LOVE** using Granny, our run-time dynamic 3D ANIMATION SYSTEM.

GRANNY 3D

There's nothing like Miles, our multichannel sound system.

And our newest tool, Telemetry, is a library for profiling, tuning and visualizing application =PERFORMANCE



So it's obvious, there's nothing like using RAD tools.

Well, nothing except having a killer MUSTACHE

GAME TOOLS ©

www.radgametools.com
425-893-4300