GDC 2014 **Visual Effects Artist** Roundtable Summary

The **Third VFX Roundtable** was excellent as usual! Lots of great topics and not nearly enough time to get through all of them, but we still did quite well. As with previous years, pretty much every area of game development was represented (programmers, artists, students, teachers, middleware providers, etc...).

Attendees were interested in discussing all platforms - mobile, steam (pc/mac/linux), last gen consoles and current gen consoles, although most of the conversation did center around PC / PS4 / XBONE. With the introduction of OpenGL ES 3.0, I think there's more overlap than ever between those high end devices and mobile and hopefully many of the talking points apply to all target platforms.

The usual "*what engine are you using*" question was asked as well, and the majority seemed to be internal, proprietary engine and VFX systems. Unity and Unreal followed closely behind - about even. There didn't appear to be anyone actively using a middleware solution.

One big difference between this year and previous years is that we tried to turn Thursday's session into one focused on "Lighting and Rendering". This was actually the smallest turn out - but I'm not sure if it's because people weren't that interested in the topic, or if there was simply scheduling conflicts. We should discuss whether we want to do this again next year, or try something different.

Last thing is a huge shout out goes to David Johnson for organizing the VFX meet (*drink*) up! Tons of people came out for that and many clever tips and tricks were shared. It was *#@$@% awesome.*

Is anybody using a 3rd party fx middleware, and if so how is it working out?

- Very few people (0?) were using particle middleware solutions.
- Many people felt like they could learn what they needed to know by testing a middleware tool, but that it would be easier to implement the features that they liked from scratch, rather than integrate the full middleware package.
- Particles themselves may not be the hard problem that we're actually trying to solve: the hard problem could be creating a core engine framework that allows for new features to be added without fundamentally changing the engine. This would make the challenge of creating amazing visual effects and visual effect features a tech art problem, rather than a core engine one.
- Another example of the above point is noting that sounds/lights/camerashake/post fx/etc… are a huge part of what makes up an "effect", but many middleware solutions will only address the particles themselves. And a middleware that does add the plethora of features listed above touches almost every part of the engine, making it difficult to integrate and maintain.

What are some best practices for asking for new tech on a live project?

- Scotch!
- Demo the feature in another realtime package - get people excited about it and use the demo to isolate specific features showing exactly how they work, and what they can add.
- As FX artists it's our job to make things awesome, and so we should be in a good position to get people excited about these new features.
- Do a time breakdown - show that the feature will save X number of hours/days.
- A dedicated vfx engineer can save money in the long run - but it was agreed that it's a tricky argument to make. There was at least one dedicate VFX engineer at the session, and everybody

was very, very jealous.

What are some easy ways to communicate with the more artistically inclined people at the start of the project? *From the an engineer's perspective*

- Complaining about features is universally unproductive.
- Artists who can learn the language of the programmer (and vice versa) tend to have a much happier and successful collaboration. i.e. programmers will have a tendency to help those they can communicate with most easily.

What are the industry expectations when it comes to your freelance reel?

This was specifically about whether it's better to include awesome VFX that look great, but don't necessarily run that well, or VFX that are locked at 60FPS, but don't look as amazing.

There was a clear consensus that making things look awesome is absolutely most important. But the ideal solution might be to make awesome VFX and include levels of detail that show how it will perform on lower end systems, and why the effect was optimized in those ways.

Ideas for making 2D stylized effects on mobile?

- A link to this fascinating **Animation Paper** software was shared: http://plasticanimationpaper.dk/
- One studio hired a 2D artist to solely do hand animated 2D VFX in flash.
- Few people had "shortcuts" for this.
- Hire Michel Gagne when in doubt.

Ideas for training?

- The excellent https://www.imbuefx.com/ site was shared. I believe it was the only suggestion.

New consoles and the future of realtime vfx! How can we get the most out of the next generation of hardware?

- More **memory** means bigger textures! Higher resolution flipbooks, more frames!
- Overdraw is still a problem. As usual, we have a higher resolution framebuffer so some of the new GPU simply goes to more pixels.
- Check out siggraph papers from previous years for techniques that could be applicable to games. The super useful "**Curl Noise**" came from a siggraph paper from several years ago for example.
- Refer to the *super incredible mega awesome* GDC talks by **Bill Rockenbeck** (particle system architecture) and **Matt Vainio** (vfx artist). They show how compute shaders can be used to create custom particle motion that runs entirely on the GPU - in fact every particle system gets compiled to its own, unique compute shader. This is the future lady's and gentlemen! p.s. also sort your particles in a compute shader so that they can all be drawn in a single pass - hotness.
- For ingenious ways of baking offline particle fluid simulations down into threshold textures that can be smoothly animated in a shader, see Eben Cook's **Surface Tension: Liquid Effects in The Last of Us.**
- For ingenious ways of applying flow maps to more than just water - check out Keith Guerrette's **Moving the Heavens: An Artistic and Technical Look at the Skies of The Last of Us.** What else can we do with flow maps that we haven't thought of!?
- **Vertex animation!** There is lots of untapped potential here. Vertex animation for particle quads, or

vertex animation for more complex meshes. One idea was that vertex animation on a subdivided quad could be worth exploring.

- **Geometry shaders?** It didn't look like anyone was using these yet, although everyone agreed they have great potential for VFX.
- A **full screen triangle** technique was shared - although I missed the details on this one. Perhaps it was referring to something like this: http://www.opengl.org/discussion_boards/showthread.php/175492-FXAA-and-Fullscreen-Triangles
- **Lit particles!** Lit particles are going to be a huge part of physically based rendering and next gen effects. In particular the use of Bump Mapped particles is currently underutilized.
- **Particle Lights!** Particles can illuminate your deferred lighting directly. Refer to Bill Rockenbeck's talk for details on this.
- **Realtime fluid sims l**ike Apex have been used in Call of Duty Ghosts. Although the performance hit was very significant. These solutions still seem a bit like a novelty, and not necessarily ready for prime time.
- **Baked simulations.** With the added memory of next gen, it should be possible to bake entire particle simulations out, and play them back at runtime. This type of solution dovetails nicely with the idea of custom Compute Shaders per particle system, as you could just have a unique compute shader that plays back a baked simulation stored in memory. We should look to the film industry for examples of this type of thing.
- **Post processing.** We should all ask for more tools that expose the power of post processing effects. Now we can do more than just color correction, motion blur etc… Custom shaders can sample world space positions, do ray intersections, and lots lots more. A great example brought up were the rain splashes in Last of Us - every rain splash was generated from a single **Screen Space Post Effect**.
- Post effects can also be applied at the geometry level - for example the buff effects in Bioshock Infinite were created this way. This has a huge benefit as well of creating effects that don't have to rely on specific UV layouts. Bug **Jeremy Griffith** if you're curious about this approach.
- **Order Independent Transparency?** Heard crickets when this was asked. Don't hold your breath waiting for it.

Are there ways that full screen Post Processing can play nicely with Particles?

For example, depth of field has been a huge problem for all transparently rendered objects including particles. There is no way for the post processing shader to separate opaque background objects from midground/foreground transparent objects, and blur them uniquely. Some solutions were shared:

- Apparently one of the talks on Ryse had an interesting solution for this. Maybe it was this one? We'll have to check when it's on the GDC vault. **Moving to the Next Generation: The Rendering Technology of Ryse**
- **Alpha to Coverage.** We need to do some research on this one. It was presented as a possible solution for post processing transparent objects. http://en.wikipedia.org/wiki/Alpha_to_coverage. Buy your most hardcore engineer some scotch and bug 'em about this one.
- Dust motes in the Last of Us were done in an incredibly clever way. They **fake DOF in the shader**, lerping between a regular dust texture, a out of focus dust texture, and a Bokeh dust texture based on distance to the camera. Genius.

What are the pros and cons of creating a brand new Node Based particle system for next gen?

As with the similar conversation about shader writing (HLSL vs. Node Based), preferences on this topic go deep, although when it gets to specifics there are pros and cons on both sides. If you go with a node based system (and it's unclear if you necessarily should) be careful of

- **Node size**. You can have micro nodes (i.e. add/multiply), but that doesn't simplify anything. Or mega nodes (i.e. BlinnPhongShading), but then what's the point of having it in the first place. The amount of encapsulation in your nodes, and the ability to share them, template them, etc… all seem to determine whether a node based system will be successful.
- It was agreed that no one wants to lose the ability to write custom HLSL or custom simulation code. It seems imperative that a node based system supports **custom nodes**.
- The film industry has been down this road and is still struggling to get it right. Lots can be learned from them.
- **UE4 seems to be *crazy* node based now**. Nodes for everything. Might be worth messing with that and seeing how it feels for those trying to make this decision.
- Prepare yourself for node networks that are absolutely mind bogglingly huge - and as a consequence very difficult to visualize and debug. We've all seen shader networks like these, so we should all know what to expect.

Lighting Topics (Thursday's Session)

Thursday's lighting session revolved heavily around **Physically Based Rendering**. For an overview of PBR, and more detailed answers to many of these questions, check out Sean Murphy's excellent **Lighting Skylanders SWAP Force** talk.

Are there good workflow methods for repurposing current gen assets for PBR?

- Few people had rock solid tricks for how to get last gen materials ported over. The general feeling was they would really need to be **rebuilt from scratch**.
- **PBR textures look different**, and often use a radically different color space than what artists are used to. For example PBR Albedo textures should never have full black or full white values.
- **128 is not middle gray** (due to gamma / linear, other effects)
- PSD files contain multiple layers specific for PBR rendering. This way artists generate all important maps from one photoshop file.
- Check out http://quixel.se/
- Consider **Physically "inspired" Shading,** it's OK to break the rules as long as you're doing it intentionally. The "physics" in Physically Based Shading really just sets a baseline so that when you deviate from it, you're doing it intentionally. And not just by accident.

What are some problems with PBR that we should be prepared for?

- **Art direction.** There was a lot of concern that PBR engines would be difficult to art direct.
- **Performance.** Although it sounds like there are solutions for making PBR fast, on slower PC's and mobile devices, it will still be a challenge. The sentiment was we'll believe PBR can run on slower mobile devices when we see it.
- Asset conversion can be prohibitive/impossible on a live project or a project deep in development.
- Harmonics didn't switch for some of the reasons above.
- Double Fine is currently deciding what route to take, in a similar quandry since our games are so stylized.

How can you actually switch to PBR?

- **Rewrite everything.** People laughed uncomfortably, because they knew this wasn't really a joke.
- Train Artists! This was hugely important. Consider weekly/monthly training sessions.
- Provide visualization tools in engine.
- Show them a **PIX Capture** and walk them through it. Scary but super helpful.

Whare are some concrete benefits of PBR?

- 2X faster for content creators to create assets! You don't have to worry about whether they'll feel like they "Fit" in the environment.
- Monster's university. QED.

Tricks for lighting particle systems? i.e. Self shadowing smoke volumes

- Exposure in PBR is difficult. Sucker Punch had a trick for modifying emissive material (fire) exposure based on time of day.
- Another idea was using average scene intensity (similar to adaptive exposure), to modulate those emissive material settings.
- You can fake some volumetric particle lighting by pretending the system is another shape (i.e. a big sphere) and projecting that lighting onto the particles.
- But beyond that, for faking self shadowing particles there really weren't any other killer solutions introduced.

Are there other good resources for lighting?

- Polycount lighting topics
- GDC Vault
- There doesn't seem to be another good resource for realtime lighting and rendering. It probably needs a home!

People's thoughts on Raytracing? Is it gonna happen!?

Consensus: It's like sasquatch. People see it occasionally but it's just a fantasy (for now).

Shadows!? Who has awesome shadows for the love of god!?

- Most people had a limit of 2 to 4 shadows at runtime.
- Except for one studio that had 8 hemispherical shadows! The 8th actually renders every three frames, bouncing between three different lights in the distance. So the actually shadow casting count is, gulp, 10! It goes without saying that everybody was jealous of this.

Unasked Questions

There were a ton of topics this year, obviously way more than we could get through. Good fodder for the **RealtimeVFX** Facebook group while we wait for next year!

- How does Oculus / stereo 3D interact with shaders/post processing, and how can VFX be made effectively in 3D?
- Tips and tricks for cinematic VFX pipelines?
- Profiling tools best approaches and best practices?
- How can you make yourself attractive to the industry?
- Best practices for particle systems on mobile devices?
- How have the new consoles changed your approach to creating VFX?

Ok - That was a TON of awesome stuff, yet again! If I got anything way off base let me know and I'll update this PDF. Catch up with everybody next year!