GDC 2013 **Visual Effects Artist** Roundtable Summary

The second, (now annual), VFX Roundtable was awesome! We had an even better showing than last year, with a full room on all three days. Attendees came from a ton of different backgrounds: Programmers, artists, tech artists, students, teachers, and of course VFX artists. Although, surprisingly, the number of people who considered themselves a straight up "VFX Artist" was actually pretty small - seemed like about 20%. Just goes to show how many different people are involved and interested in VFX, even if it's not specifically on their business card.

We also polled the audience asking what target platforms they were developing for. Again there was a lot of variety, evenly distributed between mobile, current gen, next gen, and MMO's. Also similar variety when we asked about engines and team size: Indies and AAA devs were represented, along with people using Unity, Unreal, other commercial engines and lots of proprietary internal engines.

Ok, enough background, on to the details!

How can you effectively work with designers, and how do you coordinate and balance design goals with VFX aesthetics?

There were some really creative ideas that came up during this question, and if you didn't catch it, look for **Julian Love's "The VFX of Diablo"** in the GDC Vault. It's a wonderful presentation with a lot of great high level design suggestions for crafting your VFX, as well as some super cool low level, implementation details.

- **It's important that the VFX provide accurate feedback about the gameplay** - matching explosion radius, intensity, etc...
- Because we all know that design tuning can happen up until the day a game ships, some studios have clever tricks where **the visual radius of an explosion is actually driven by an explosion radius** specified by designers. In some cases, VFX artists could expose specific elements of their effects to designers, and in another, VFX artists would actually author 2 entirely different explosions. Say a 1 meter explosion, and a 10 meter explosion. By cleverly interpolating the effect parameters between those two explosions, any size in between could also be generated.
- There was also a lot of support for simply **sitting VFX artists and designers together** in situations where the two departments aren't necessarily operating on the same wavelength.
- It's important to think with both VFX and design in mind from the beginning.

How do you use concept art?

There was a large consensus that **using concept art directly is not particularly helpful for VFX.** On the other hand using concepts to clarify the mechanics, intensity, style, and inspiration for the effect seemed pretty effective. Many VFX artists would make their own concept arts to help solve VFX problems - not necessarily with the intention of ever showing someone their work, but just as part of their own personal process. Short version: concept art is awesome, but trying to replicate someone else's concept art in a real time engine doesn't always yield the best results. In addition it can be time consuming, not very performant, and and often frustrating for the artists.

Do designers use VFX tools, place VFX in levels, tweak VFX in game?

- There were definitely **a lot of studios who had their designers add VFX to levels.** It seemed to be almost inevitable with very large projects (think Skyrim and MMO's).

- **Sometimes designers will mock up VFX with placeholders**, and the VFX artists will polish them up later. My favorite example of this was a process where designers would place a camera facing billboard sprite with the text of the desired effect on it. i.e. “HELLO I AM A WALL OF FIRE!”. The artist would later see the billboard, and replace it with a cool looking wall of fire.
- Another approach was to have **VFX artists build effect “kits”** that contained all the elements necessary to create various effects. A waterfall kit might contain mist, spray, splashes, falling water, rainbows, etc... And the VFX artist can teach a level designer how those pieces are supposed to fit together.
- **Documenting how, when and why to place the environmental VFX didn’t always help** (no one reads them!) The kits seemed to be more effective.

What are your favorite tips and tricks?

- **BlendAdd / Premultiplied Alpha.** In Julian’s session he describes this in awesome detail. Basically, he talks about how you can use the premultiplied alpha blending mode to create effect textures where a mask channel defines what parts of the texture are additive, and what parts are alpha blended in the traditional way. That allows you to effectively *knock out* colors from the destination image and replace them with your own, but only in very specific parts of your particle texture. Other parts of the texture can simply add onto the destination image, or smoothly blend between additive and alpha using those grayscale mask values. The point is that you can retain fantastic color saturation and texture detail, even against a bright background, while still getting the bloom and glow that’s so necessary for sweet looking fx.
- Similar to the earlier technique where you let designers blend between two different effects to create an explosion of arbitrary size, you can also **blend between two different effects** to easily create variety. Fading from a soft ember fire to blazing fire for example
- **Making videos of your VFX** to use across the company and across projects.
- **Using a vertex shader to get an awesome organic beam effect.** This was done by making a unit long ribbon with very dense vertices. Then the unit beam can be oriented to face a target, and scaled in the vertex shader by the distance to the target, effectively making the ribbon reach out to the target’s position. At this point sweet multi directional sine waves can be used to get wonderful, organic, wiggly awesomeness.
- **Combining flip books with UV displacement / flow map techniques**. This lets you use fewer frames, because even though the animation might not be super smooth, the UV displacement is always running smoothly which hides those artifacts.
- A**nimated UV lookup, flip-book texture.** Yup that’s a mouthful. But it’s a very cool trick - basically you create an animated flip book texture, but instead of being composed of the color information, it contains UV lookup information. In the shader, you fetch the color from the animated UV lookup texture, and then you use that color to sample the color from your actual particle texture. Note that this is a dependent texture fetch so it’s a bit more expensive than traditional flipbook animation. However it does mean that you can now animate any of your particle textures using a single animated lookup texture. The specific example for this was being able to shatter any of your particle textures.

What should we be teaching students who want to go into VFX?

When the group was asked how many people were trained in VFX, not one of the 50 or so people in the room raised their hand. However here are some skills that we agreed would be very good to foster in the students:

- An understanding of both the art and the tech behind it
- Being resourceful – we often have to do a lot with extremely limited tools
- Communication is key
- Learn about performance profiling for your game
- Learn to sometimes be pushy and reach outside your discipline
- The ability to learn quickly and pick up new tools
- Being able to isolate the essence of an effect, making it look cool, while keeping it performant.
- Show students how cool VFX are and then explain to them that this is a job they can get!

Particles and Time of day... Discuss!

We kicked off Thursday with this juicy topic, asking how different studios make their particle effects reusable in different lighting conditions - for example night vs. day, or out in the open w/ ambient light vs. inside a dark cave. One common solution was to bake ambient lighting color information into trigger volumes or other gameplay metadata, and then pipe that color into the shader for your particles. In some cases a simple tint can go a long way.

For open world games specifically, there were some interesting workflows that addressed this too. For example some artists would have a comprehensive set of lighting environments that they can quickly swap through, seeing how the VFX look in each setting. Another variation on this was to actually author all effects for a very specific reference lighting environment. That consistency allows the team to experiment with other tech solutions (often shader solutions) to make the particles look good in those other times of day.

Also asked folks for a show of hands on how many people had shadow casting particles. Only two or three lucky artists had them.

Have people come up with good solutions for scaling VFX between mobile, mid-range, and high-end targets?

Specifically, if you have a game that runs well on a mid range target, have you had success porting those VFX to mobile? What about to a high end PC? Not surprisingly, there were no silver bullets on this one. The brute force solution of manually ripping out particles and features to get to mobile performance specs seemed to be the only realistic approach. To some extent you can procedurally reduce particle count and features as well, but most people still needed an artist to supervise the process.

One solution for making assets look better for next gen was aptly named “Next Gen Multiplier” - which reduced particle opacity by 50% and also increases emission rate by 200%.

This conversation led to the interesting notion of **Physically Based Effects**. Inspired by the movement toward **Physically Based Materials** in recent lighting and rendering techniques. Although it’s unclear exactly what a physically based effect would be - here were some ideas:

- Artists don’t define specific particle counts. Instead they specify things like emission volumes, speed, desired overall opacity.
- Instead of choosing additive or alpha blending, artists would actually define physical characteristics like how emissive a glow is, how does the effect/particle block sunlight and cast shadows? How does the effect/particle reflect incident light?

The thought is that if the artist authored how they *want* the effect to look, rather than what the effect actually *is* (the actual raw data that drives the effect), talented engineers would be able to recreate the desired look using whatever technique is appropriate for the platform. For example a fire on mobile might just be some

additive particles. In next gen those additive particles might be replaced with emissive particles that can actually emit light and shadows in addition to glowing.

In general, the idea of a physically based effect is certainly pretty exciting! Even if it's just the seed of an idea at this point.

What do people want from next gen?

We're still using quads even on next gen. That's not so surprising, but what are the other features that people wish they could get their hands on?

- **Sorting**! Just give us properly sorting particles pleaeeeaaaase.
- **Procedurally generated geometry** could be very cool. Although no one seemed to be doing much with that yet beyond just procedurally generating ribbon trails.
- **Soft body physics** simulations
- The ability to combine **meshes, rigs, particle effects and other techniques in a single effect**. This was more of a tools issue - as it's often challenging to have creative control over all those different assets in a single level editing tool.
- It would be cool to get more out of **mesh based particle emitters**. Perhaps instead of having to have quads for everything, we can have particle sprites that are tessellated to more closely match our textures so we can spend minimize overdraw.

What is your workflow like on large projects (like MMO's)?

- MMO's have a MASSIVE asset bank, and so it's a requirement that the VFX are modular.
- Lots of use of kits here. This really helped standardize FX across the project.
- Artist education was big too. Especially when it comes to performance. MMO's require a big team effort in order to get VFX populated through the world - and if you don't want a terrifying, final optimization pass to land in your lap take the time to teach the other artists how to profile VFX, and how they can impact performance.

On that note, how do you enforce performance-centric thinking?

- Take the time to build tools that can display shader complexity, overdraw, etc... in game.
- Visual debugging of this and all kinds of other data is enormously helpful.
- Have a responsibility for your own optimization and create a culture of performance monitoring, starting with the designers.

Poll on FX tools

- ***How many people are using VFX Middleware?*** Very few!
- ***Who's creating VFX in a 3D Package (Maya/Max)?*** A handful of folks. Maybe 20%
- ***Who's creating VFX in their own engine with an internal tool?*** Vast majority of people.

This question spawned some conversations about creating VFX in a DCC package versus a proprietary tool. A few people really liked the power they were able to get out of creating all the VFX in a DCC app like 3DS MAX. The tradeoff that concerned people here is that you aren't actually looking at your VFX in engine, and most people wanted to be able to edit their VFX in the context where the player would see it. There was also a risk of having artists use features in the DCC app that aren't supported by the engine (or vice versa). In the end, there was a bit of a split on which approach was best, though the majority of people seemed to

prefer using an in house, proprietary VFX tool.

Ok - That was a lot of exciting stuff! See you all next year!!!

Special thanks to these folks for helping to arrange the Roundtable and organizing our awesome community!

- **Mark Teare** for organizing the AMAZING vfx drink up
- **David Johnson** for leading the charge remotely and continuing to get us represented inside the VES
- **Rachel Day** and **Robert Campbell** for helping me pull together notes from the roundtable
- **Jeff Hanna** and **Lee Petty** for helping to organize the VFX Round Table
- **Jeff Hanna** and **Steve Theodore** for organizing the Tech Art Bootcamp