GDC 2015 **Visual Effects Artist** Roundtable Summary

This year was the **Fourth VFX Roundtable**, and as always it was fantastic to see everybody together - new faces and old. There was an abundance of great networking, great conversations, an awesome VFX Mixer (organized by David Johnson and sponsored by Side Effects Software), and 3 days of *raw information mind melding!*

We kick off the roundtable with questions designed to get a bit of info about the makeup of attendees. This year we had about 50% VFX Artists, 30% VFX Programmers, a smattering of people from other art and engineering disciplines, and lucky us even a hardware guy! That was a treat. On engines, there was a surprisingly even split between Unreal, Unity, and Proprietary engines (i.e. Frostbite). Cryengine had representation although a little bit less than the others. About 75% identified themselves as console/pc focused, 25% mobile. One person (me!) was real geeked up about VFX in virtual reality.

This year we tried something new, and that was to take topics from the Facebook page (https://www.facebook.com/groups/realtimevfx/), print them out, and bring them to each session. This ended up being quite useful as it gave us a lot of material to burn through and focus the conversation. As always, the goal of the three days is to maximize information throughput, and we did pretty good this year.

Thanks to Alex Radkov for helping to put these notes together!

Physically Based Rendering and Lighting of VFX

Physically Based Rendering (PBR), has had a huge impact on games over the last year or two. It's clearly a powerful tool for unifying materials and lighting, but how does VFX figure into that equation?

- **How do you measure PBR values for VFX?** Is it even possible? The consensus was that for solid surfaces, you can get reasonable approximations. For transparent objects (particles, haze, liquids, gases, etc…) this is much more difficult. For example, there is no real physically based correlation between a "billboard sprite" representing smoke, and the actual smoke particles that scatter light in the real world.
- Some teams had different pipelines for rendering "solid" VFX vs. "transparent" VFX. In this case the solid pipeline supports PBR, and the transparent one is, as usual, based on a bunch of magic numbers and artist tweaking.
- Another team had a system for approximating "Rayleigh Scattering" to light transparent billboard particles realistically under different lighting conditions.
- Most studios are working with **HDR effects**, but are faking the emissive values that drive those effects. This leads to problems with PBR engines. For example as camera exposure changes from area to area, the look of the effects can change dramatically. This was usually solved with *more* magic numbers.
- Most **particle lighting** was being approximated with image based lighting (IBL) and/or ambient lighting including spherical harmonic (SH) terms.
- **Success stories:**
    - **Voxel-based lighting in Fable.** Spawn particles on the CPU, simulate and light them on the GPU. If anyone has more info on this, please post!
    - **UE4 indirect lighting.** Another team customized UE4 to use a screen space voxel grid to approximate lighting on particles. It was reported to be very effective!
    - See the great "**Lighting Skylanders SWAP Force"** talk by Sean Murphy form GDC 2014.
- Hope to hear many more success stories on this next year. Generally speaking, it sounds like it's

still difficult to unify PBR and VFX.

Global Wind

How do people handle wind motion in general? Is it a global solution? A per effect solution? Etc..

- Generally global wind seems to be useful, and most teams had a parametrized “wind influence” so that particle systems could ignore it, or modulate it, as necessary.
- A few teams could use the “amount of wind” value as an input to their particle systems so that custom behaviors could be created in response to wind changes.
- Many, many people had anecdotes of wind influenced effects that simply fall apart and look terrible as the wind intensity increases. This was clearly a source of frustration, and one of the risks of depending too heavily on an external wind system.
- **SCE Japan Studio:** We have a global wind field, which isn’t hooked up to the new system yet. Might use, might not. Biggest benefit comes from the fact that our system is on the GPU and having the global force as a texture seems like an intuitive solution. Designers still want more control, so we might work in that direction instead.

Forces and Fields

This was a general question about **vector fields, attractors, repulsors**, and how people are using fields in general right now.

- Some teams have a “global” field that drives particles (similar to the wind solution above). Others have forces like attractors and repulsors that are local to specific particle systems.
    - One advantage of a “global” field was that it plays very nicely with GPU particles as everything is now on the GPU and can be parallelized.
    - The more traditional attractors/repulsors had a lot of support as well, they tend to be easier to craft and a bit more consistent. Anecdotes of Attractors/Repulsors being used for interesting wheel dust effects were shared.
- **Constraints**. People were curious if anyone had a system for constraining particle simulations so that they can’t be “pulled apart” by external forces and fields. Don’t think anyone had a good solution for this, but it was a problem shared by all. It’s no fun seeing your beautiful smoke plume turned into a bunch of isolated billboards by some external force.
- **SCE Japan Studio:** My designers wanted something like the particle forces in Maya, which are pretty good. Once you substitute the basic bounding shape (sphere) with something else (box, torus, cylinder) you can get some crazy effects.

Flipbooks

The bread and butter of VFX! Flipbooks! Yay!

- **Distortion & Optical flowmaps** to improve quality of flipbooks and reduce size. (Somebody mentioned Guerrilla Studios were using flowmaps in Killzone 3). These seem to work well with flipbooks containing hard surface/rigid elements, but fall apart when you are trying to represent overlapping transparent surfaces (i.e. smoke).
- A few teams were using an **After Effects Plugin** to general these optical flow maps for compression (not sure which plugin it was).
- Helpful tool for authoring flipbooks: ToonBoom.
- **2D Arrays** can speed up texture fetching and to better compress grayscale flipbooks (ask your programmers).

- There were discussions about packing all 4 channels of a DXT5 compressed texture with 4 grayscale flipbook frames. This worked in some cases, but compression artifacts look terrible when there are large changes between the frames. I think we agreed we should call this technique **RGB Cramming**. Texture arrays are better, but not supported on older hardware. See the next section for more details on texture compression for VFX.
- Also see **Bill Rockenbeck's** talk last year was about using 1 Material for all particles and sorting/batching all of them together, I think they used 2D Arrays for this to work.

Flipbook Compression

From Alex: *Big miss on my side, I said DXT5 when I meant BC7. The formats are here:* https://msdn.microsoft.com/en-us/library/windows/desktop/hh308955%28v=vs.85%29.aspx

Basically

Dx11	Dx10	Quality	Size(bytes per 4x4 block)
BC3	DXT5	RGBA (5:6:5:8 bits per channel)	16 ?
BC4	ATI1	R (8 bits per channel)	8
BC5	ATI2	RG (8:8 bits per channel)	16
BC6H	none	RGB (16:16:16 bits per channel)	16
BC7	none	RGB or RGBA (varied)	16

We use BC4 for grayscale (best quality), BC5 for normals (distortion, normal maps for particles etc.) and BC7 for RGBA (best compression). BC7 offers better quality than BC3 (former DXT5), there is no reason to use DXT5 if the hardware supports BC7 (Dx11 and PS4 do, not sure about mobile) BC6H is good for HDR (no alpha).

Another option: **ASTC** (Adaptive Scalable Texture Compression by ARM Ltd.)
Wikipedia says there is an extension for OpenGL/ES as of Aug 6 2012.

VFX Artist required skills and suggestions for students

A question was asked about the appropriate type of course work for VFX students. This launched us into a discussion about the skillset of VFX artists, hiring practices, and relevant tools.

- One question was: "*In a 6 week VFX course, should students spend any time doing VFX in Maya?*" The, not that surprising, unanimous response was NO. Students should not spend any time doing VFX work in maya, but instead should be authoring VFX in an engine.
- **Sketchbooks**. Some teams looked specifically for sketchbooks from VFX applicants when hiring.
- A **sense of art and the fundamentals of art** are more important than actual technical knowledge which can be taught later. Obviously students still need an aptitude and interest in the technical side.
- **Timing!** Animation timing = hugely important. (Not news to most people reading this).
- Being able to demonstrate skills for hand **authoring/painting textures is important**.
- Some teams don't draw by hand, and instead **generate flipbooks and other textures from offline simulations**. This can also be very effective, and people seemed to approach VFX authoring from both of the simulation, and hand painting, directions.

Suggested Books

- Elemental Magic: The Art of SFX Animation
- The Animator's Survival Kit

Suggested Tools and Middleware

- PopcornFX
- Shuriken (Unity)
- Cascade (Unreal)
- Particle Playground (Unity)
- ShaderToy
- ShaderForge
- ShaderFX
- Houdini

Pipelines for offline simulation

- **Point-cached effects** can be really good. Execution is the same every time, but particle to particle interactions can have offline simulation quality. Very heavy on the hardware (recommended for current gen AAA console titles). Good for explosions, destructions.
- **Looping.** Techniques for creating looping simulations include FFT wavelength techniques in Houdini (anyone have links for this?). Also mesh blendshape bouncing to ping pong between different keyframed shapes for endless looping.
- At the very end of the week, **OpenVDB** (http://www.openvdb.org/download/openvdb_toolset.pdf) and Sparse Data Structures were mentioned. It sounds like this is a very powerful technique for streaming offline simulation data into real time engines. Someone talk next year on this please!

Team Layout

A question was asked about people's experience working with a larger VFX team as part of central tech (services oriented) versus being embedded in smaller strike teams.

- **One response:** "*Sitting with the team you're working with now, rather than the effects group, gives better iteration results (Combat Team) and prevents Us vs Them mentality (artists vs game designers vs programmers).*"
- **SCE Japan Studio:** We have a core effects team (3 artists + 2-3 programmers) sitting together which helps for debugging, discussing new features.
- A **hybrid approach** was to have VFX artists embedded in strike teams, but have daily standups with the entire VFX team each morning.
- Generally speaking, there was an even split between people in favor of a "**Centralized VFX Team"** vs. "**Embedded VFX Artists in Strike Teams**".

Virtual Reality

This year Virtual Reality was a pretty big deal at GDC. Lots of activity and talk. This topic did a cursory examination of how VFX and VR play together.

- Deferred rendering is very expensive because of the *enormous* number of pixels involved in VR. **Expect to switch back to forward rendering**, and give up some of the deferred rendering tricks we have learned to love.
- **Post Effects are very expensive** for the same reason. On mobile, expect NO post effects. On desktop, limited post effects.
- Valve's talk on **Advanced VR** by **Alex Vlachos** is a must see for rendering programmers.
- **90FPS means 10 milliseconds…** *for everything.*
- Refraction/Framebuffer effects can be prohibitively expensive.

- Per-pixel lighting from points/spots can be prohibitively expensive. Directional is OK.
- **Biggest Takeaway**: Most VFX techniques work great in VR, especially when approached from the low end PC/Mobile mindset. In generally, **VFX look amazing in VR!**

Volumetric Rendering

- **Nobody in the room was using Raymarching for VFX** in real time applications, even though everybody was excited about it. Some uses cases where it is used are grass rendering, parallax occlusion mapping, fog, and screen space reflections. But not much for VFX. For example, as cool as pyroclastic noise seems to be, it's not being used in practice by attendees.

Misc Techniques

At the end of the roundtable, we dug into miscellaneous techniques that would be generally useful, interesting and/or valuable to bring back to our teams and our graphics programmers.

- Lots of people are using **Threshold Alpha / Alpha Crunch / Alpha Ramp / Alpha Erosion** / Etc… (many names for it!) to create texture animation without requiring multiple textures. This is a topic that's come up before, and continues to be very powerful.
- **SCE Japan Studio:** We use Spherical Harmonics for lighting centered either on the billboard center (for effects with fewer particles) or 1m above the emitter (for small scale effects).
- Using **macro textures** to give particles uniform pattern is also good, UVs either based on screen space or world space.
- **Motion blur can be faked** with blurred particles which write out to a distortion buffer, or read from an existing framebuffer. One example was for faked motion blur on wheels as they accelerate. Another was motion blur on sword trails.
- **Instancing geometry** to particle positions was a very powerful technique.
- **Trim particles with tessellation** to prevent heavy overdraw (saves about 30%). Multiple engines had this support - hugely important for mobile, VR, and traditional desktop.
- **DOF techniques for mobile** were discussed. No killer solution here, although switching to low res MIPS and/or rendering objects with replaced shaders were some mobile friendly ideas for post processing.