Particle Systems

And Other Notable Render Paths

by Russell Gillette (Skybox Labs)

Systems for Rendering in Game Engines

- Decals
- Gobos
- Billboards/Imposters
- Foliage
- Sprites
- Hair
- Volume Rendering
- Sky
- Point Rendering
- Voxels
- Signed Distance Fields
- Fonts
- Mesh Rendering
- Particle Systems

Why are there so many render paths?

- Generated meshes
- Performance tradeoffs
  - hacks to do something in a faster way, often with visual tradeoffs
- Memory or data transfer efficiency:
  - data packing
  - transmission of draws to the GPU (eg: execute indirect, instancing)
- Data might be represented in a different manner
  - eg: fluid simulations, lines, etc
- Data inputs may be computed differently

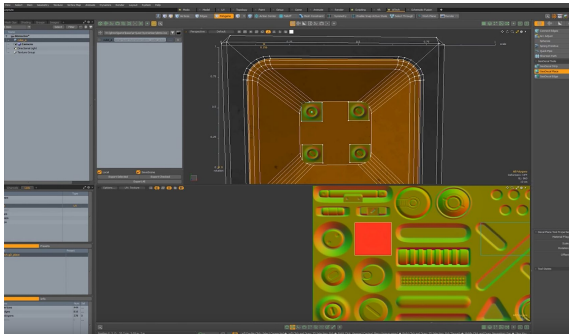
Disclaimer

- There are often many ways to do things.
- Each implementation is different based on use case, engine, and author whims.

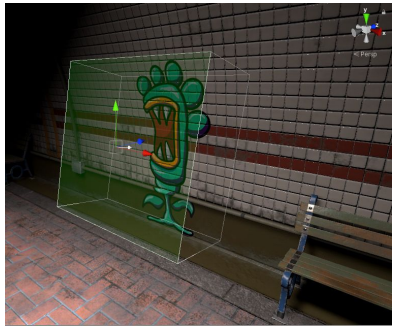
Rendering Decals

- Decals are images projected onto a mesh to provide more detail
  - usually implemented as a projected cube
  - A mesh can be generated for the decal if static
    - avoids custom shaders, provides perf improvement
    - generating meshes is slow and done offline
  - for a much more elaborate use of decals:
    - https://advances.realtimerendering.com/s2020/RenderingDoomEternal.pdf

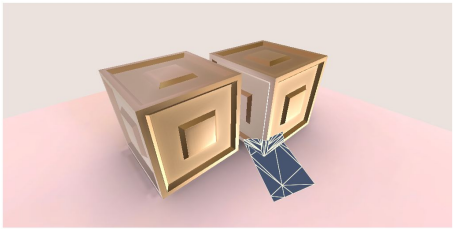
https://samdriver.xyz/article/decal-render-intro



Doom Eternal Review by Digital Foundry:
https://www.youtube.com/watch?v=UsmqWSZpgJY&t=396s



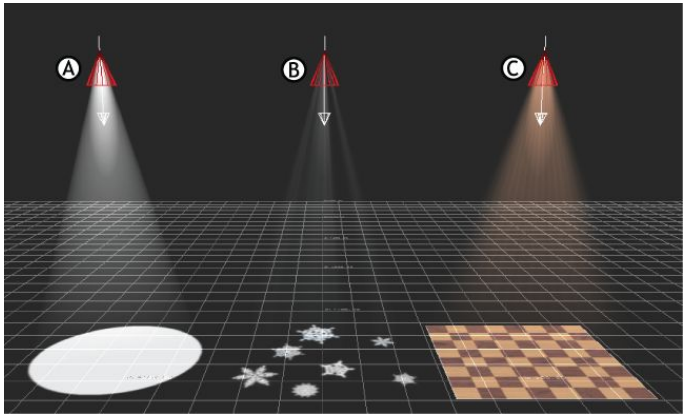
https://devforum.roblox.com/t/forward-rendering-decalsdecal-projecting/1154955



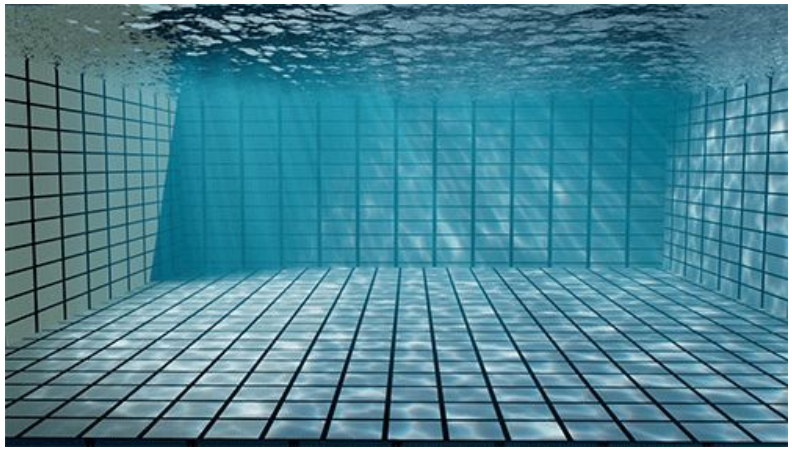
https://samdriver.xyz/article/decal-render-intro

Rendering Gobos

- Projective Textures attached to a light
- not usually sampled for Global Illumination
- common uses: caustics underwater
- can be animated for more realism



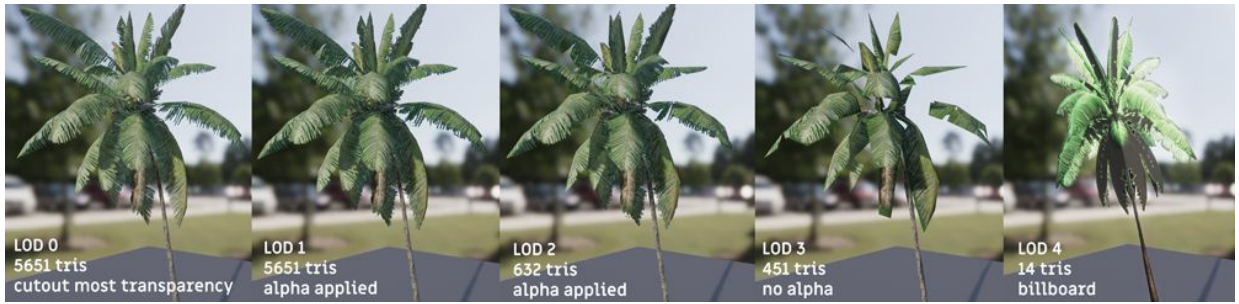
https://download.autodesk.com/global/docs/motionbuilder2014-tutorial/index.html?url=files/Custom_lights_Attaching_a_gobo_to_a_light.htm,topicNumber=d30e85749



https://docs.arnoldrenderer.com/display/A5AFMUG/Caustic+Effect+Using+Cell+Noise%3A+Pool+Scene

Rendering Billboards and Imposters

- Flat textures with transparency to imply detail
  - often camera aligned
- Usually done after the lowest LOD in an LOD chain for far objects that need to remain visible
- Imposters allow for a wider range of angles with reasonable plausibility



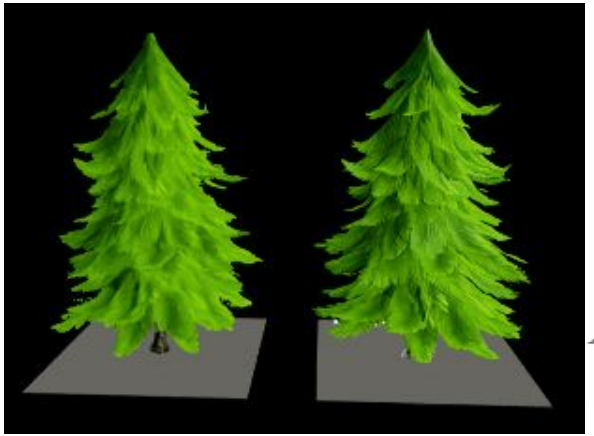
http://phos.ph/2017/11/21/creating-dense-foliage-in-vr-part-2/

https://shaderbits.com/blog/octahedral-impostors

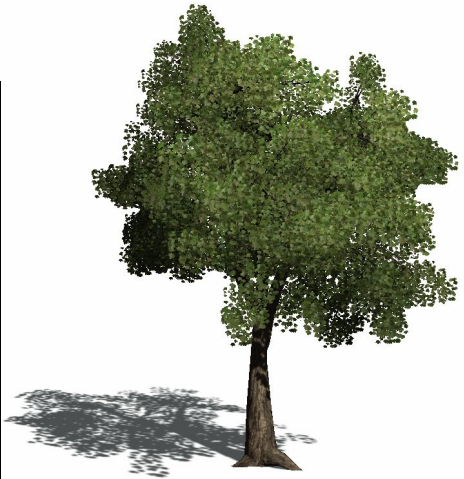
Rendering Foliage

- Often rendered as a series of billboards
  - Not always a separate render path
- Also often a simplified mesh
  - Trees have a lot of detail, can still be very costly to render many of them
- See http://www.stephanmantler.com/files/star1021.pdf for more detailed tree rendering

https://gonintendo.com/stories/275514-miyamoto-is-a-big-fan-of-tree-climbing-in-the-legend-of-zelda-br



https://shaderbits.com/blog/octahedral-impostors

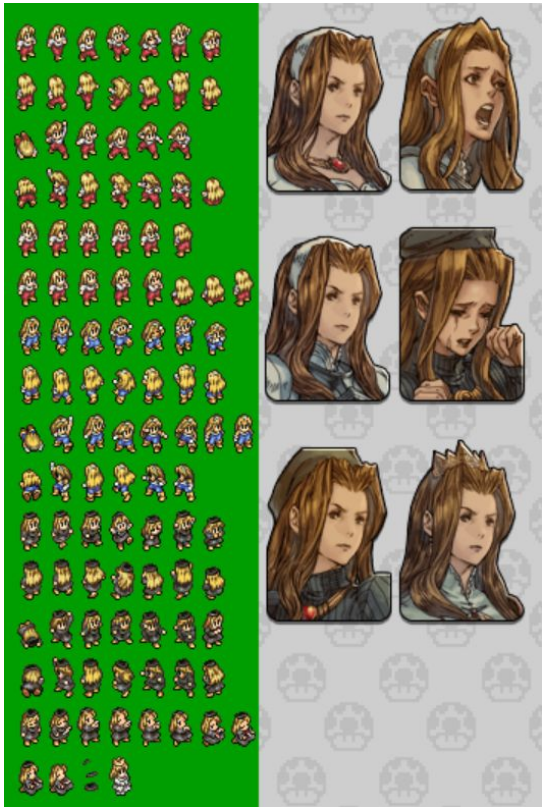


https://sites.cs.ucsb.edu/~holl/pubs/Candussi-2005-EG.pdf

Rendering Sprites

- quads with a texture sampled from a texture
- often a large number of draws can be concatenated into a single triangle strip
- quads are often in screenspace
- many images to be sampled are all in a small number of sprite atlases

http://www.hardcoregaming101.net/tactics-ogre/



https://www.spriters-resource.com/fullview/74463/

Rendering Hair

- Often rendered using a different material model called a BSDF
  - BSDF models light on a full sphere rather than a hemisphere
  - modified to properly represent a number of internal Transmittance and Reflection events
- Splines used to as “key hairs” to guide generation of additional hairs
- Hair Mesh can be used as a volumetric stand in
- Also for speed may use surface models, though they are hard to make look nice

Image from the game Heavenly Sword

http://www.cemyuksel.com/courses/conferences/siggraph2010-hair/S2010_HairCourseNotes-Chapter2.pdf

http://www.cemyuksel.com/courses/conferences/siggraph2010-hair/

Rendering Volumes

- Model the propagation of light through some media
  - as opposed to normal material evaluation stopping at the surface
- Can be used for smoke, fire, clouds, water, the sky, and more
- Frames problems in terms of scattering theory on participating media
  - mathematically modelling the behaviour of light in the media
- Often involves some form of ray casting or marching

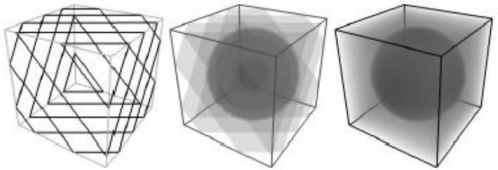


Figure 1.7: Volume rendering by 3D texture slicing (WESTERMANN and ERTL 1998): 3D texture slices are generated from the volume, perpendicular to the viewing direction (left); the textures are mapped onto the screen (middle); blended textures of previous slices (right).

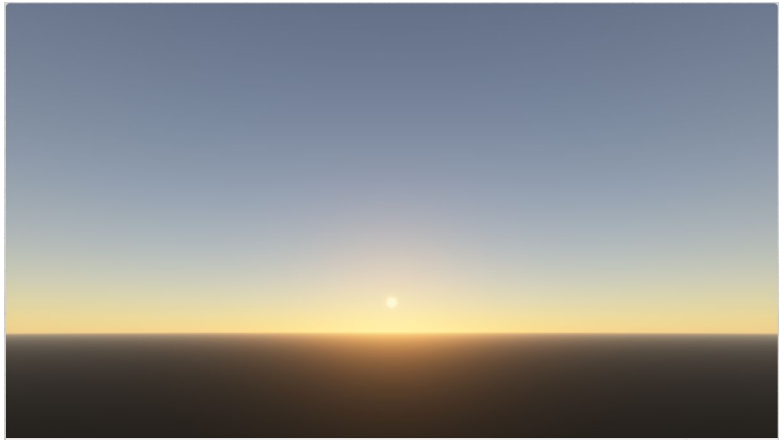
Older approach to volume rendering as 3d slices

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.682&rep=rep1&type=pdf

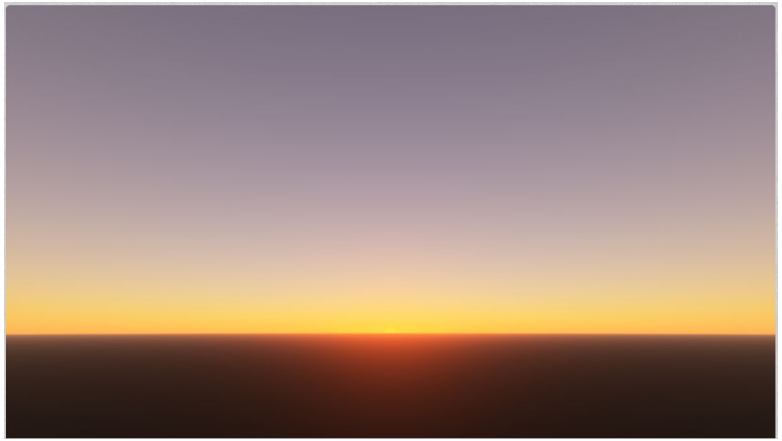
https://www.diva-portal.org/smash/get/diva2:1223894/FULLTEXT01.pdf

Rendering The Sky

- Rendered Volumetrically
  - modelled statistically to avoid heavy computation of ray bounces
  - math heavy computation to account for transmittance, scattering, and phase
  - data often stores as look up tables on altitude and view angle relative to zenith
- Sky can also can be rendered using environment maps



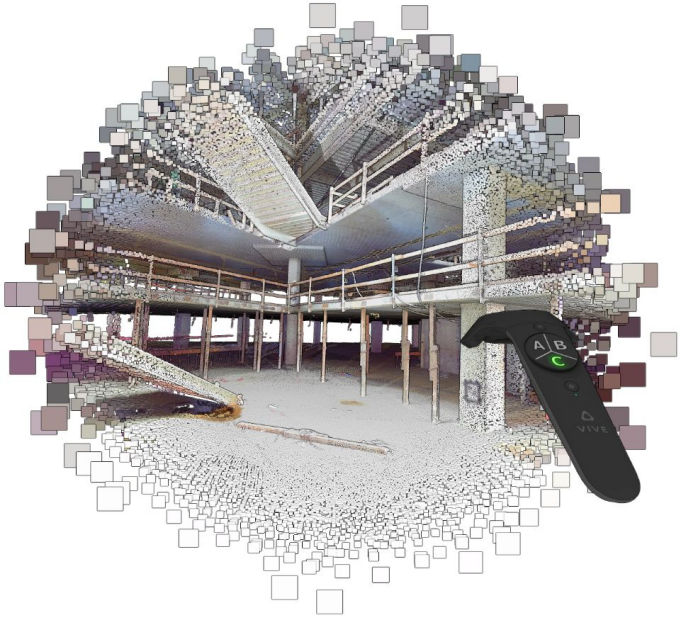
https://www.shadertoy.com/view/slSXRW



https://www.shadertoy.com/view/slSXRW

Rendering Point Clouds

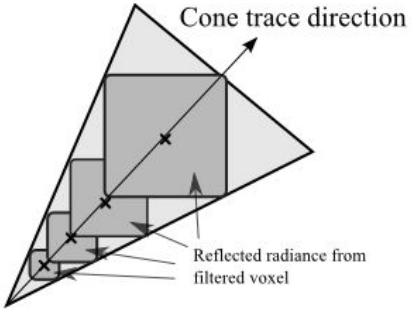
- Used for medical imaging often
- Hard to animate
- Can be rendered by:
  - splatting circles to the screen
  - rendering particles
  - constructing a mesh from the points
  - voxelizing the space and rendering that
- lighting can be challenging depending on what data is available per point



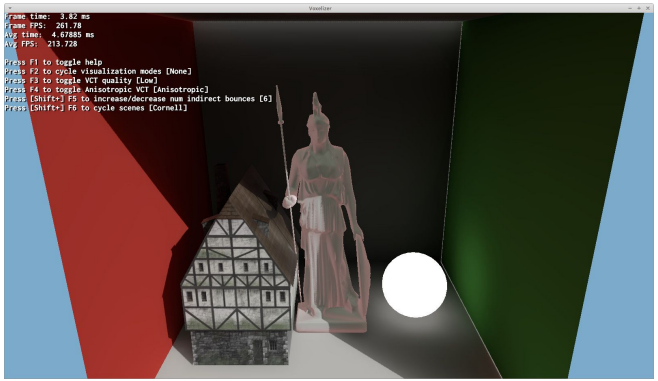
https://www.cg.tuwien.ac.at/research/publications/2019/schuetz-2019-CLOD/schuetz-2019-CLOD-screenshot_1.png

Rendering Voxels Pt 1

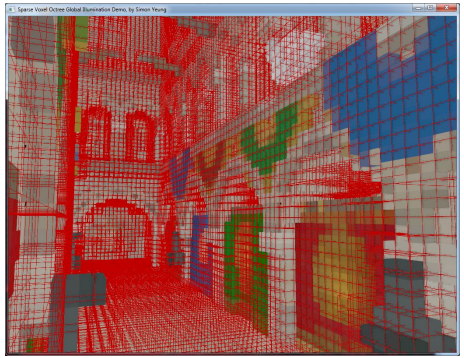
- Voxel meshes can be used
  - to store lighting information on the world
    - Voxel cone tracing can be used to render area lights
    - cone tracing is also used in come computation of GI
  - for sims using cellular automata
  - in finite element analysis to determine tensile strength



Voxel Cone Tracing
http://simonstechblog.blogspot.com/2013/01/implementing-voxel-cone-tracing.html



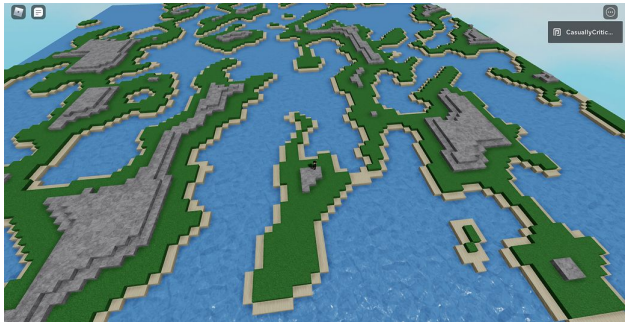
Voxel Cone Tracing in Ogre 3D
https://www.ogre3d.org/2019/08/05/voxel-cone-tracing



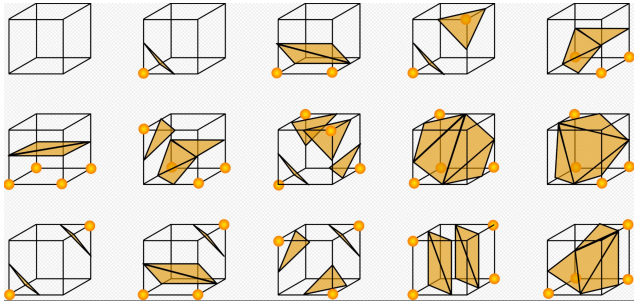
Voxelation of scene and storage in Octree
http://simonstechblog.blogspot.com/2013/01/implementing-voxel-cone-tracing.html

Rendering Voxels Pt2

- Can be used for geometries and destructibility
  - minecraft/roblox
  - point cloud data
    - each grid holds a bit specifying in or out of the object
    - perhaps also density or rate of flow
- Rendering of voxelized surfaces can be done by:
  - converting to a polygonal mesh using the marching cubes algorithm
- Ray tracing the voxels directly
  - well suited for GPU implementations



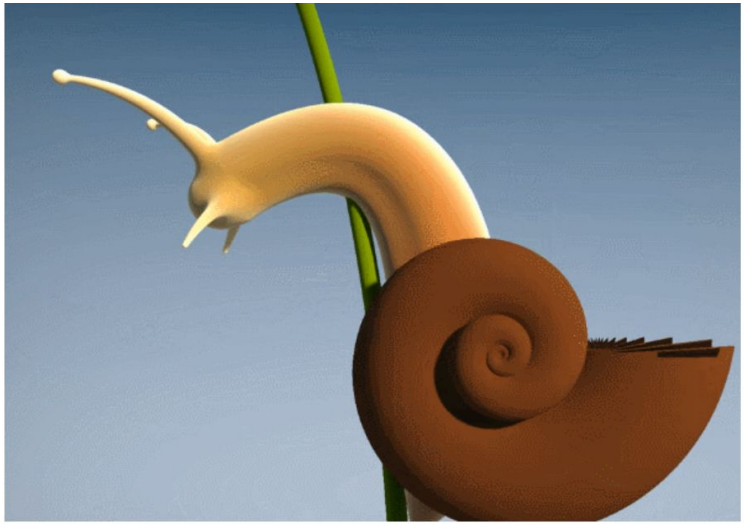
Voxel Terrain Generation in Roblox
https://devforum.roblox.com/t/cellular-automata-for-voxel-terrain-generation/594922



https://en.wikipedia.org/wiki/Marching_cubes

Rendering Signed Distance Fields

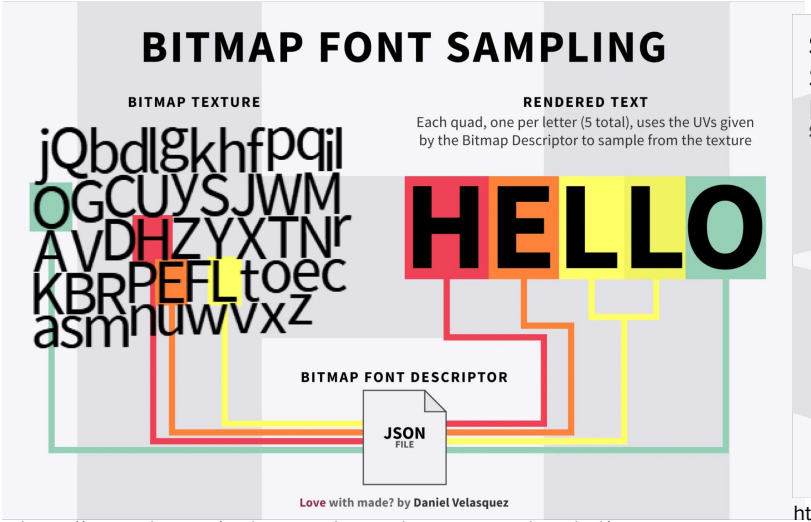
- can be used to represent objects as distances to a function
  - I’ve seen used in production for fonts and font-like symbols
- can be used to generate point cloud sets



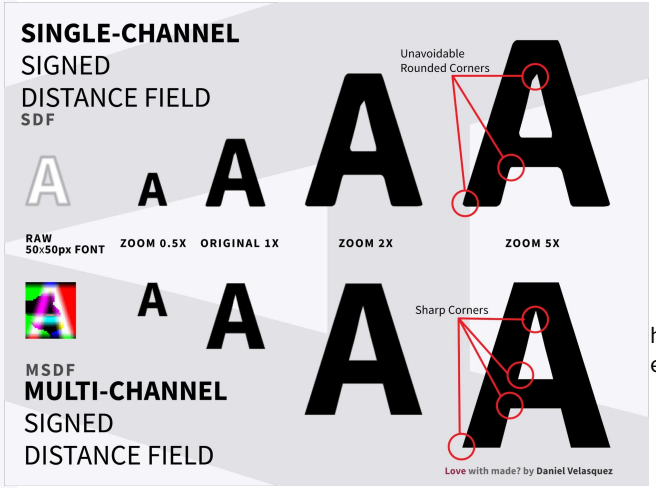
https://www.alanzucconi.com/2016/07/01/signed-distance-functions/

Rendering Fonts

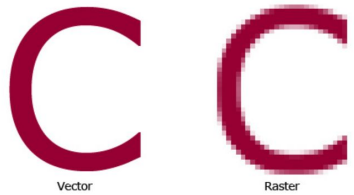
- Often rendered similar to sprites
  - Can also use vector fonts for clean fonts at all scales
    - (though often rasterized to the desired scale)
  - More complicated fonts and symbols can be represented using signed distance fields



https://css-tricks.com/techniques-for-rendering-text-with-webgl/



https://css-tricks.com/techniques-for-rendering-text-with-webgl/



https://www.digitemb.com/c-font-raster-to-vector.php

Particle Systems

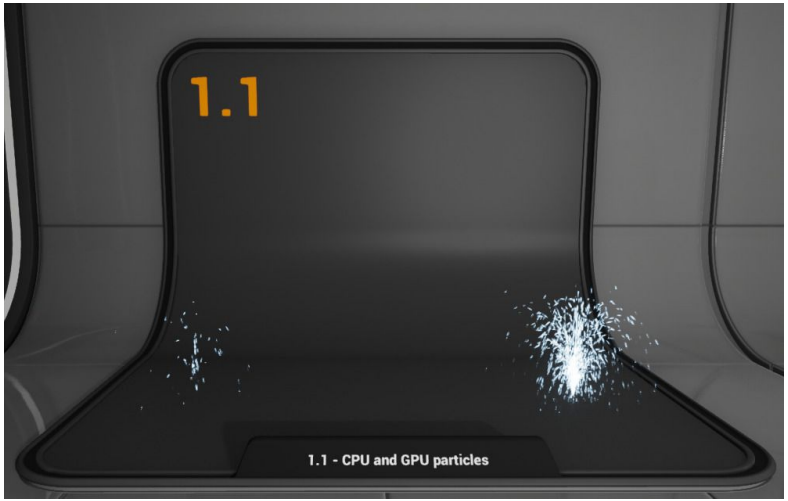
What is a Particle System?

- Loosely two parts:
- Generator/Emitter:
  - anchor point for system in the world
  - object that tracks
    - which particles are alive
    - how long they've been alive
    - when new particles spawn
- Particles:
  - objects with some spawn state (velocity, position, random see for animation, etc)
  - simulation behavior
  - conditions for death
  - can be rendered in any way depending on desired results (though sprites are most common)

https://cesium.com/learn/cesiumjs-learn/cesiumjs-particle-systems/

CPU particles or GPU Particles

- Cpu particles support more features
  - some features are limited to cpu particles (eg: much harder to emit light from gpu particles)
- Gpu particles allow for many more particles on screen at a time
  - GPU allows sim in parallel, usually done in compute shader



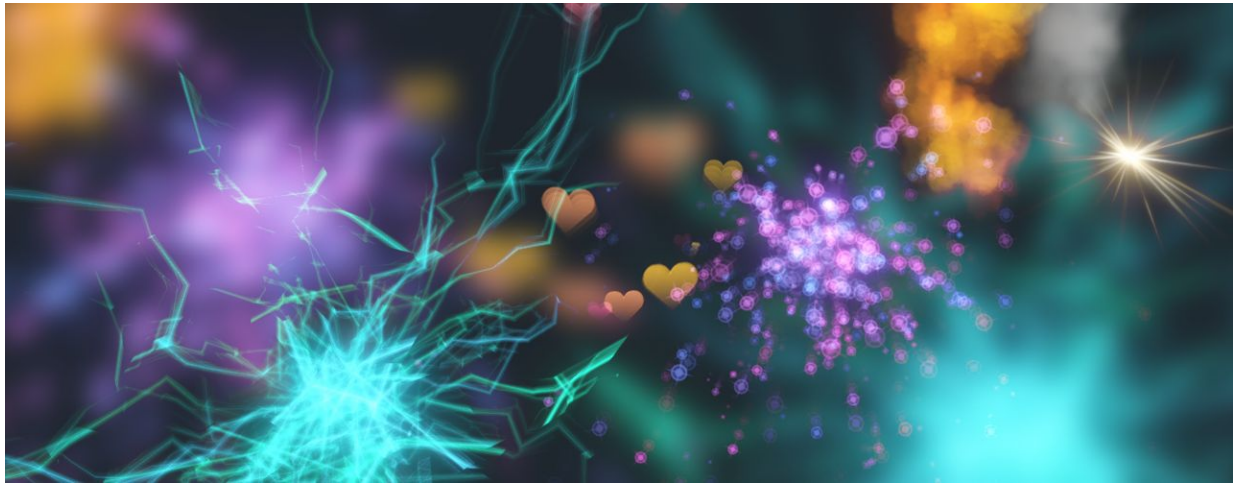
https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/EffectsGallery/1_A/

Particle Pipeline Per Emitter

- Stage 1: Update the Emitter
  - emitter may need to follow an object, etc
- Stage 2: Terminate old particles
  - if any particle has reached the criteria for death, remove them from the list
- Stage 3: Spawn new particles
  - generate new particles for the system based on artist derived spawn criteria
  - spawning may be limited by perf concerns
- Stage 4: Simulate particles
  - Update particle positions, animation state, etc
- Stage 5: Render individual particles
  - render the particles using whichever pipeline is desired for the particles
  - when done on GPU, should still be separated into compute (update) and graphics (draw) passes
    - the update can be used to derive lighting data against other scene objects, etc

Update the Emitter

- Move the emitter to follow a target
- Split the emitter into multiple emitters
- Trigger the emitter for termination
  - stop spawning particles
  - maybe phase out particles before disappearing



https://developer.amazon.com/blogs/appstore/post/028909c9-86f6-446c-8ab4-5b6e67c02f30/drawing-particle-effect-sprites

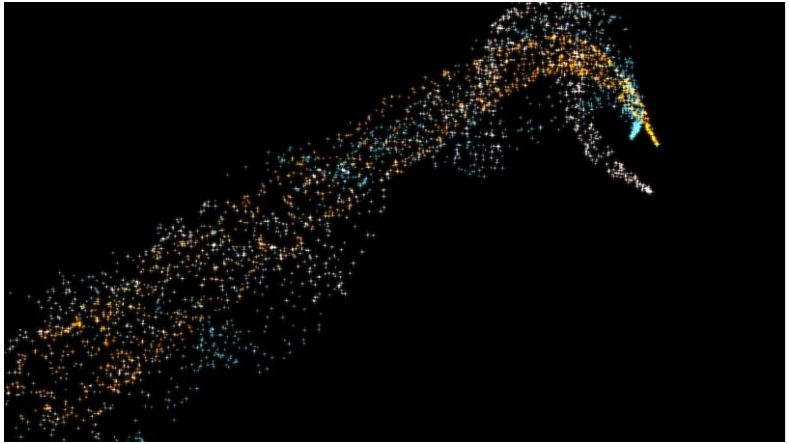
Terminate Old Particles

- Usually based on lifetime
- Other Ideas:
  - distance from emitter
  - number of active particles
  - randomly
  - after some particle animation plays it course

https://pikbest.com/video/qianku-golden-particle-line-disappearing-animation-free-pendant_2447004.html

Spawning Particles

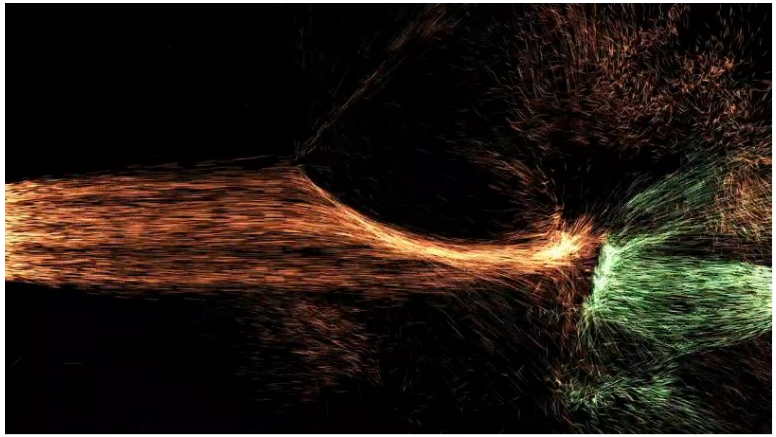
- Usually relative to emitter location though not necessarily
  - following the emitter allows you to pin an emitter to an object for effects
  - spawning particles on other particles allows effects like fireworks
- Spawning location and details plays a large part in effect



https://borisfx.com/products/particle-illusion/

Updating Particles

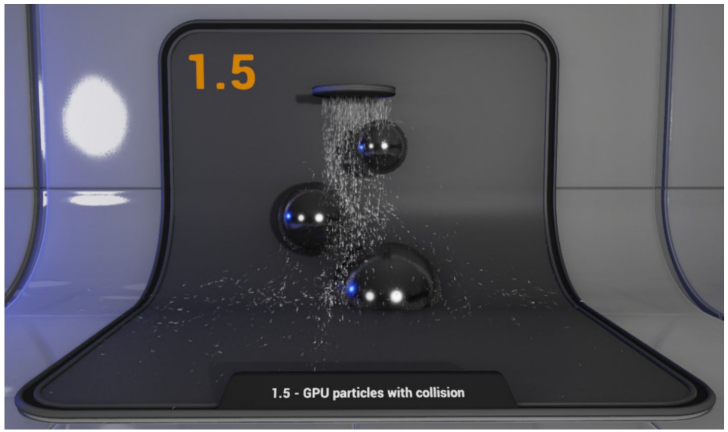
- Collision: Easy on the GPU by testing against the depth buffer
- Light: Easy on the CPU by populating point lights on particles after sim
- Movement:
  - can be directed by splines, signed distance fields, physics (spawned velocity), steps towards a target, random walk
  - External forces to control movement: gravity, wind, vector field, etc



https://borisfx.com/products/particle-illusion/

Rendering Particles

- Most Commonly Sprites or Points
  - can be grouped into a single tri-strip or drawn through executeIndirect to avoid draw call overhead
- Not limited to any shape or geometry
  - eg: mesh streamers are common



https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/EffectsGallery/1_E/

https://docs.unrealengine.com/4.27/en-US/Resources/ContentExamples/EffectsGallery/1_D/

Rendering Particles Cont.

- May need multiple render passes
  - Almost certainly will need to be rendered in the Transparent pass
  - May need to be in the Opaque pass as well
  - Depending on how you define your passes, you may also need a separate pass for points
    - (You don’t want to be swapping render topology a lot)

https://www.unrealengine.com/marketplace/en-US/product/fireworks-particle-effects

Advanced Usage of Particle Systems

- Not just for effects
- Also used to model complex systems such as fluid sims
  - particles hold all sorts of data for the model such as velocity, rotational velocity, etc at that point in the fluid

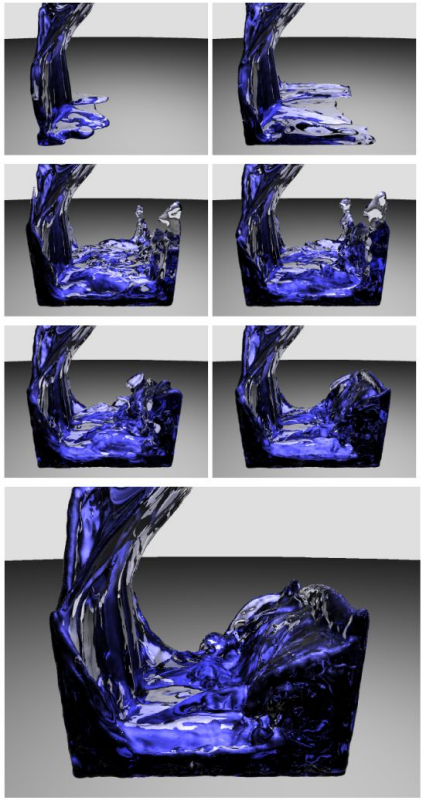
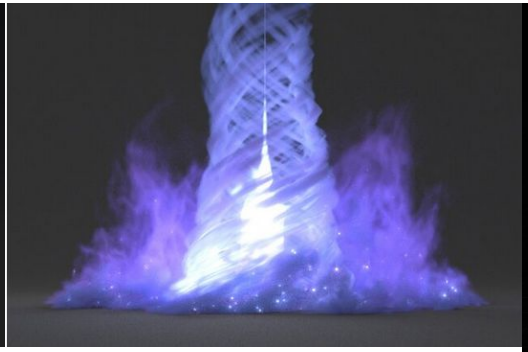


Figure 3: *A source with three nozzles filling a box. The fluid motion is simulated by 150,000 fluid particles. The fluid is being emitted from three nozzles that hit an obstacle surface set near the top of the box.*

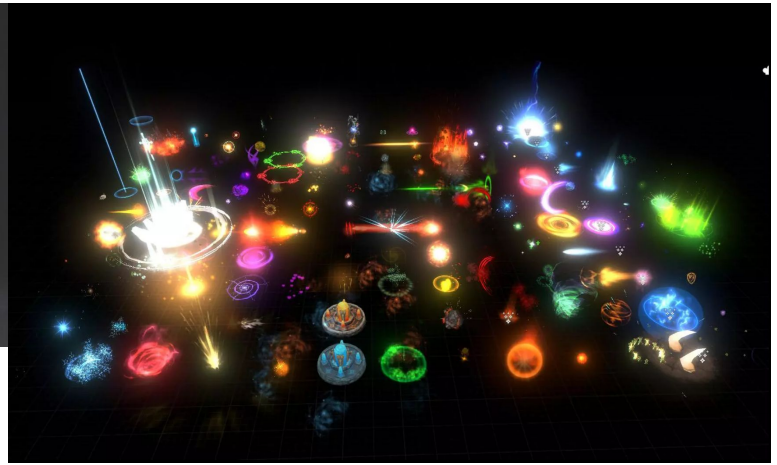
http://www.sci.utah.edu/~tolga/pubs/ParticleFluidsHiRes.pdf

Particle Effect Examples

Particle Generator with Petals
https://www.appliedhoudini.com/



Procedurally Generated Volumes
https://www.appliedhoudini.com/



Magic Effects for Games
https://assetstore.unity.com/packages/vfx/particles/epic-magic-particle-effects-pack-124834

Starburst with tails and glow effect
https://comptutorials.com/game-design/create-particle-effects-in-gamemaker-studio-2/

The End

Mandatory Plug: Skybox Is Hiring!!