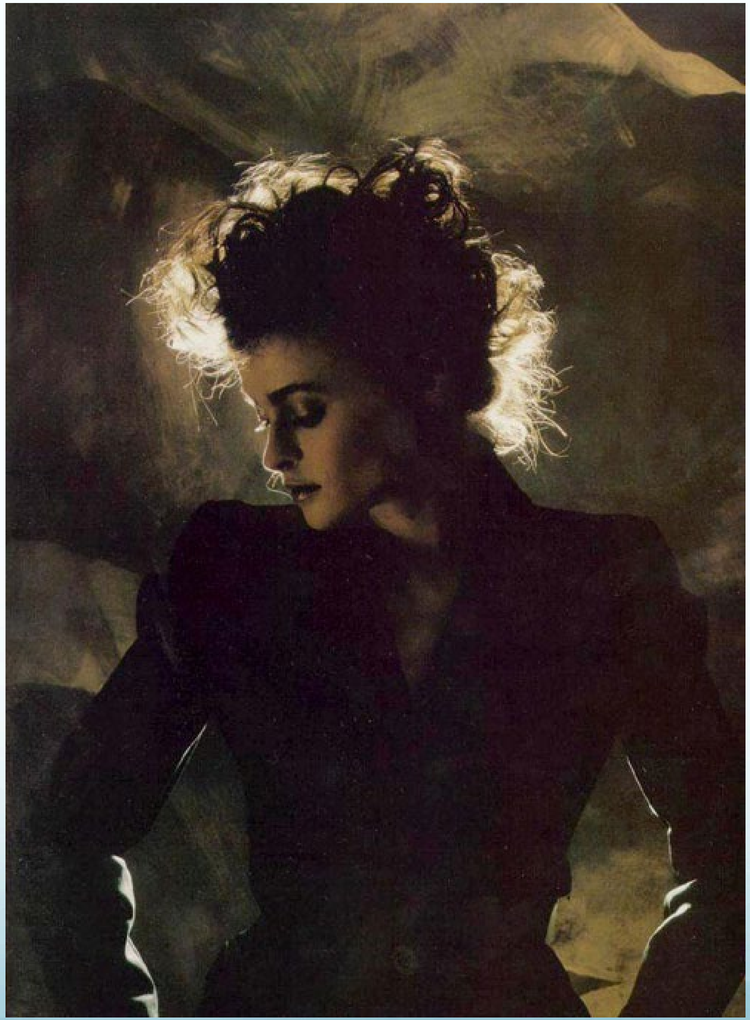
CSC 240 Computer Graphics
Video 21: Subsurface Scattering & Particle Systems

Nick Howe
Smith College

Subsurface Scattering


http://capsulesofenergy.tumblr.com/post/26433135511

Subsurface Scattering

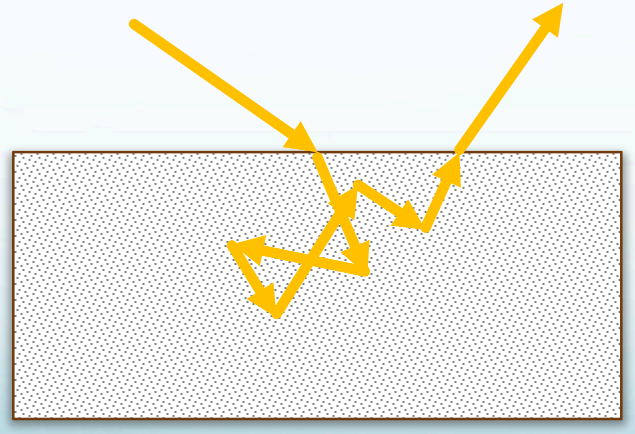
For some materials, SSS is necessary for realistic rendering

Direct Subsurface Combined

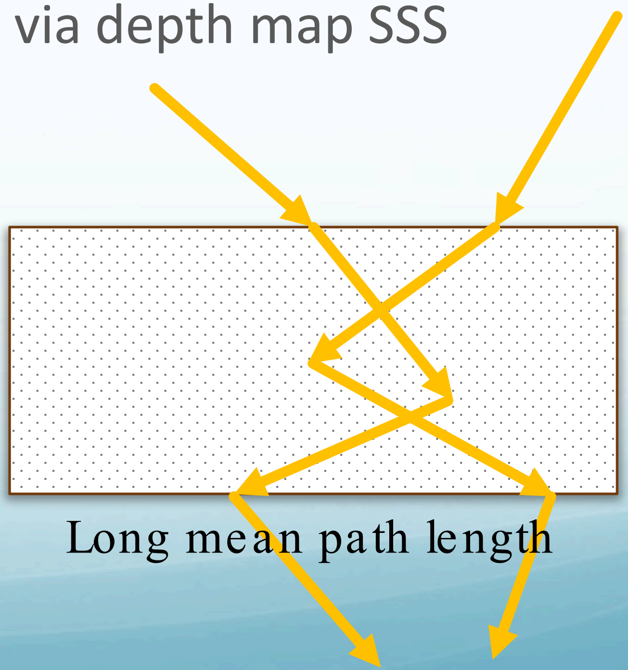
Subsurface Scattering

Many opaque objects still allow some light to travel through them

- The light travels an unpredictable path via **subsurface scattering (SSS)**
- Differing effects & techniques depending on the *mean path length*
  - Short mean path length: Localized SSS via texture space diffusion
  - Long mean path length: Simulated backlighting via depth map SSS



Short mean path length

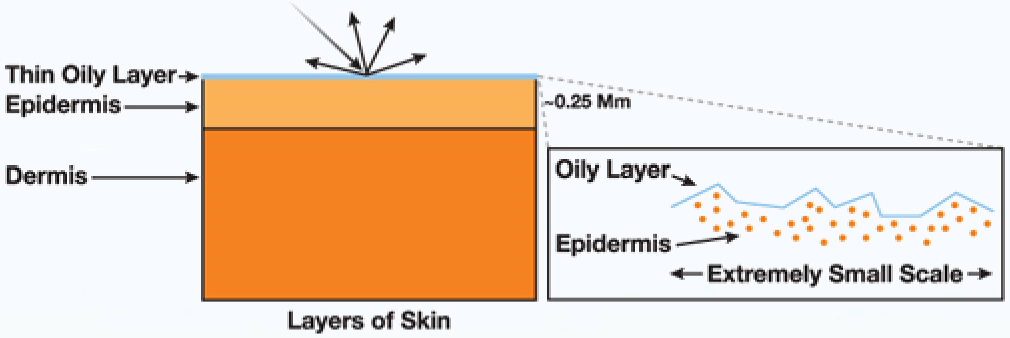


Long mean path length

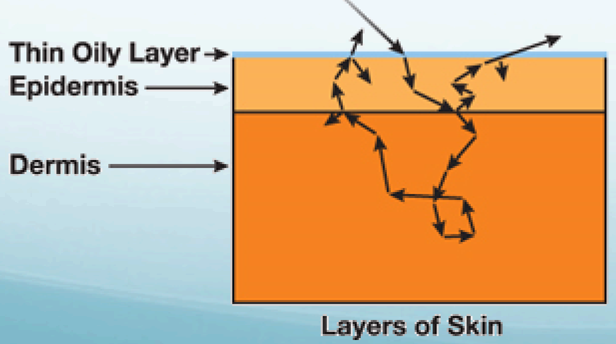
Subsurface Scattering

Human skin interacts with light in complex ways

- Some light reflects directly off oil layer



- Some penetrates and scatters

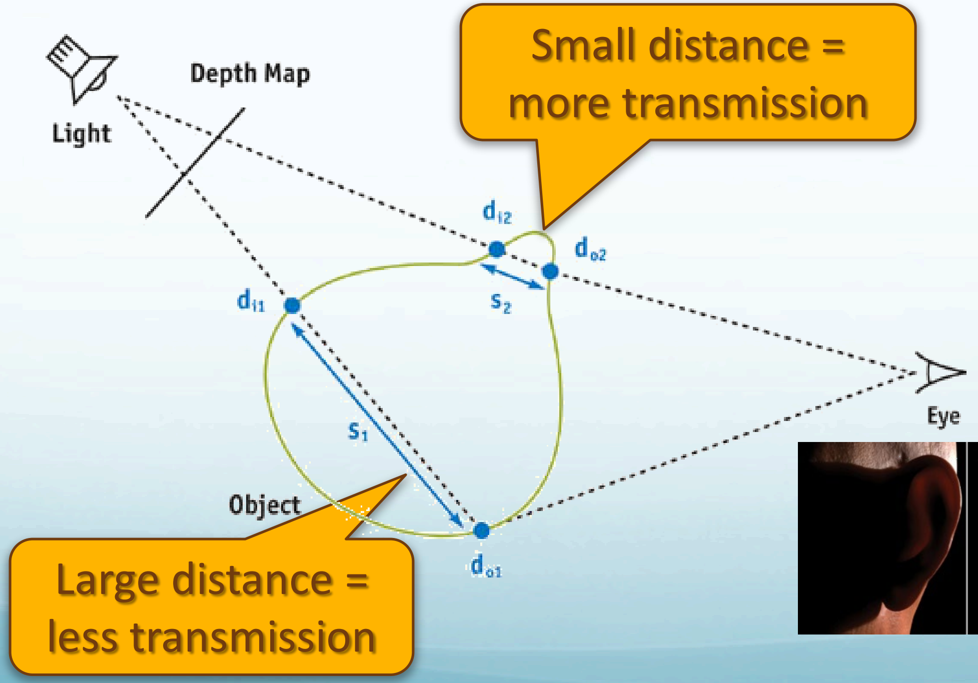


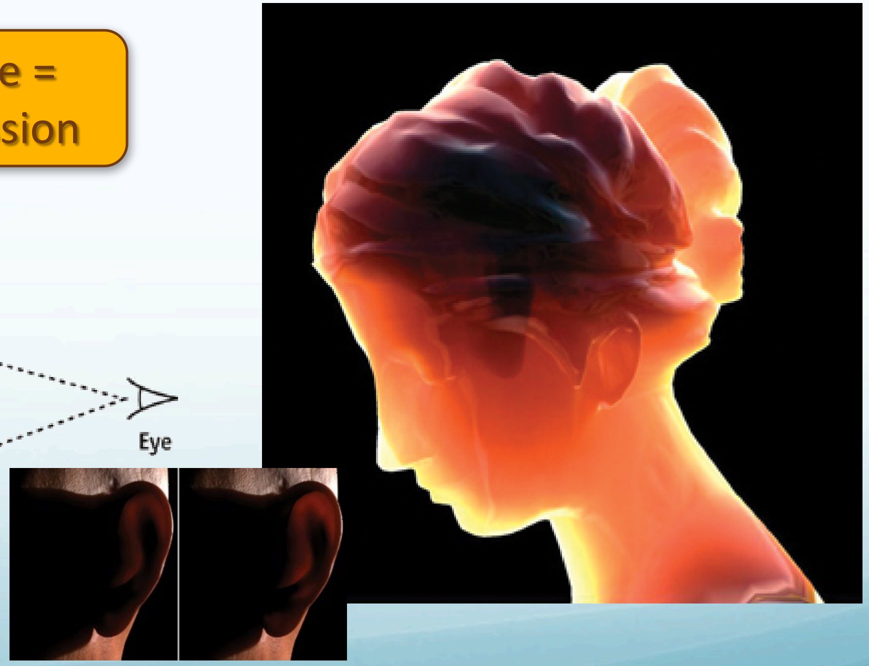
https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch14.html

Subsurface Scattering

Depth map SSS works similar to shadow calculation

- Render scene from both light and camera; look at z buffer
- Distance between surface points gives material thickness



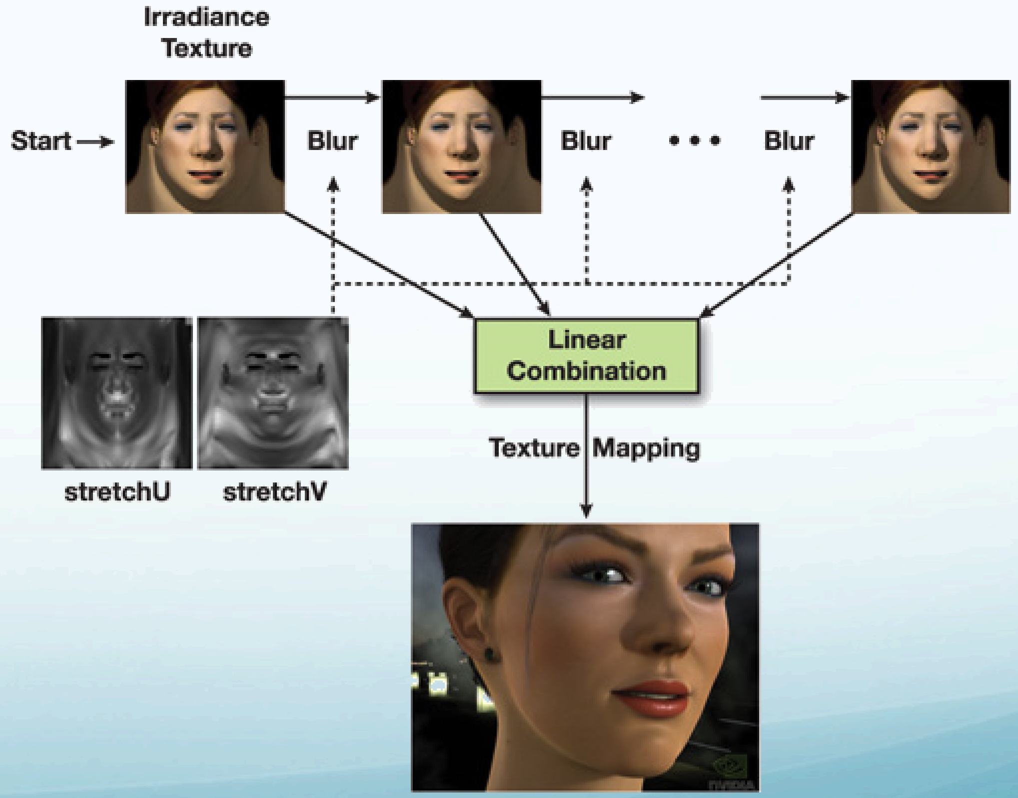


https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch16.html

Subsurface Scattering

Short-range texture space diffusion SSS uses blurring in unwrapped UV map

- Reverse map object lighting onto texture plane
- Blur illumination image and re-map onto object
- Process different colors separately



Final Pass: Combine Blurs + Specular

https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch14.html

Questions

PAUSE NOW & ANSWER

1. What key measurement governs the type of subsurface scattering in a material?

   Mean path length

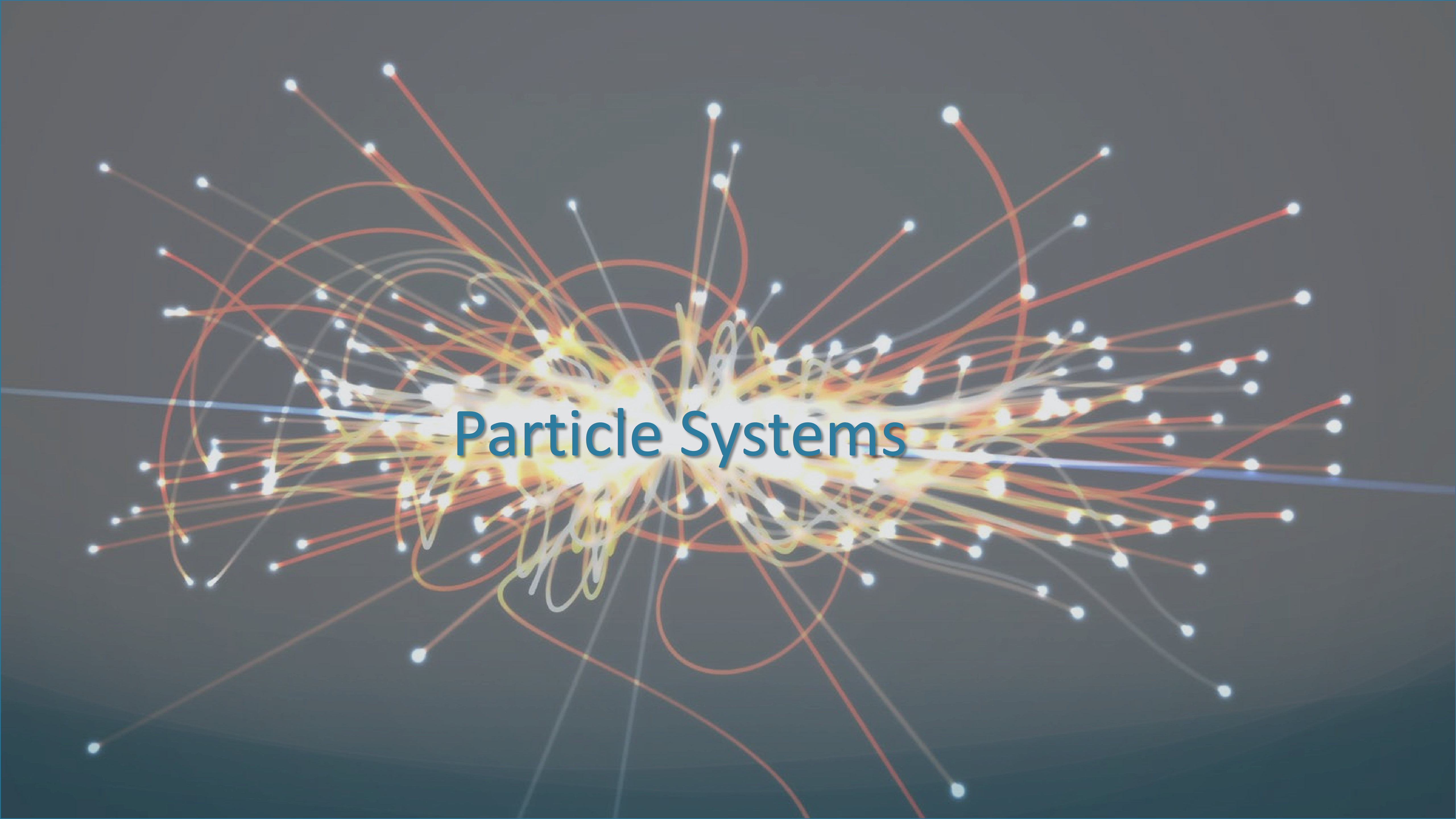
2. Which SSS technique is used to model light diffusing short distances though a material?

   Texture space blurring

3. Which SSS technique is used to model backlight shining through a bulk material?

   Depth map SSS rendering

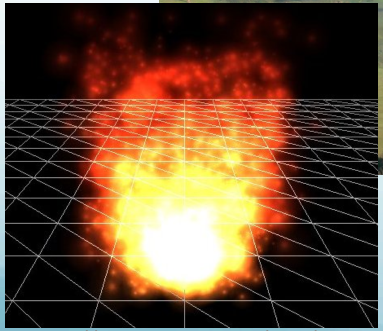
Particle Systems



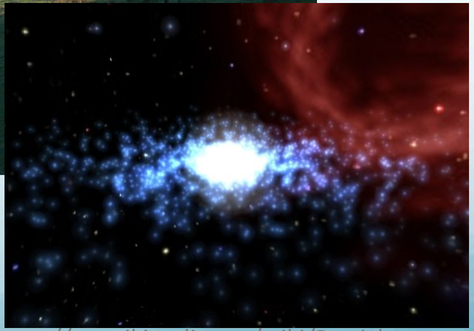
Particle Systems

We've studied lots about rendering solid objects

- What about liquids and gases? Flexible stuff?
- These are best rendered using many small "particles"



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

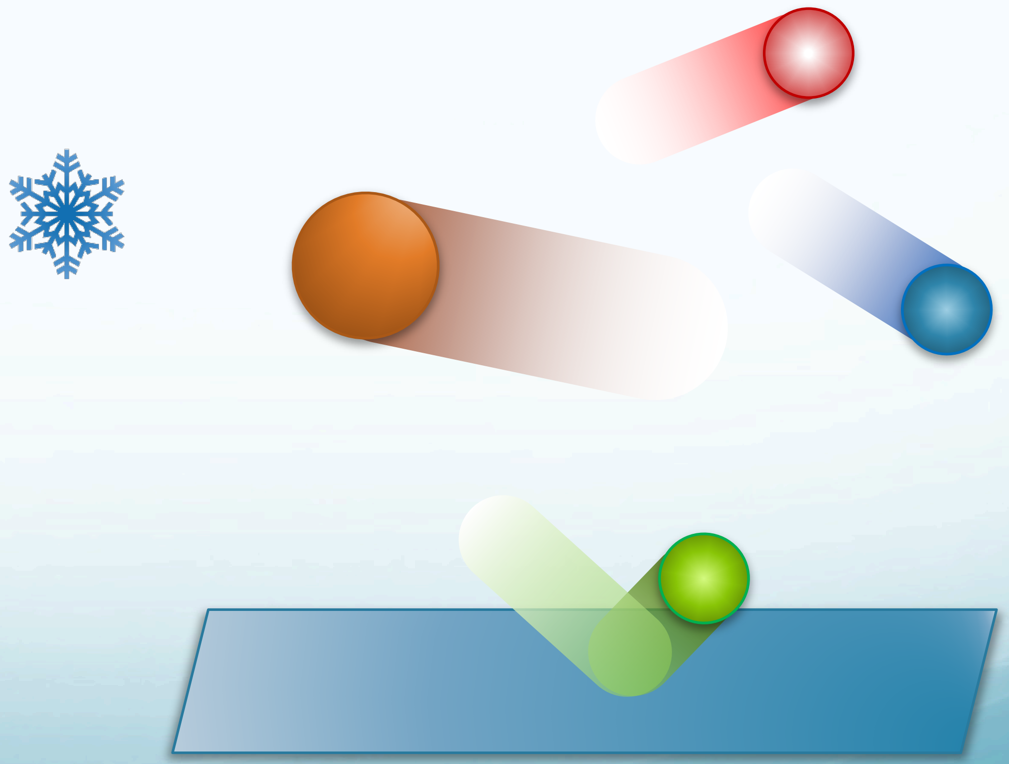


https://en.wikipedia.org/wiki/Particle_system

Particle Systems

Overall effect made up of many interacting particles

- Typical particle lasts for limited time
- Has properties relevant to desired effect
  - Position & velocity
  - Color & appearance
  - Size, turbulence, stiffness, etc.
- Can interact with the environment
  - Physics!
  - Cartoon or otherwise!



Particle Systems

Each particle has internal state, updated over time (e.g., each frame)

Some possible simple models:

- Position only, with random updates (Brownian motion)

$$\vec{p} = \vec{p} + \vec{r}$$

- Position & velocity, with randomly updated velocity

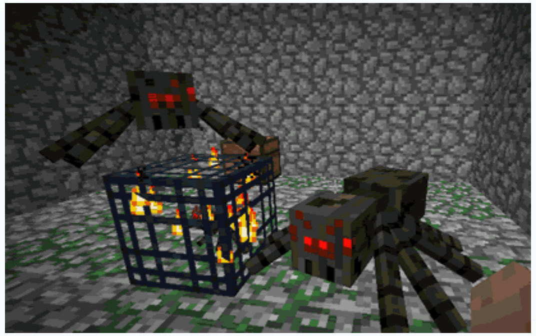
$$\vec{p} = \vec{p} + \vec{v}\Delta t \qquad \vec{v} = \vec{v} + \vec{r}$$

- Position, velocity & constant acceleration (gravity)

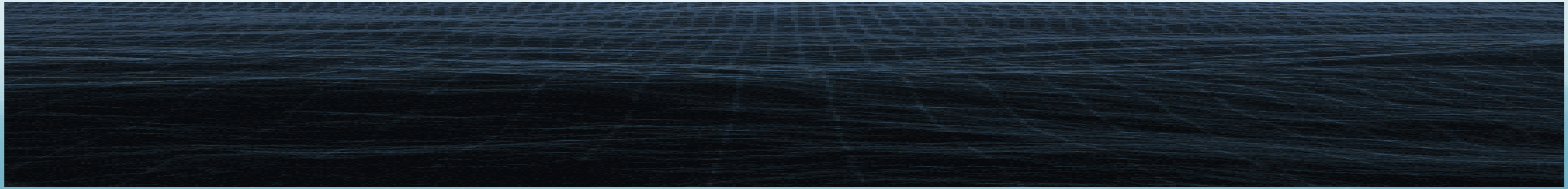
$$\vec{p} = \vec{p} + \vec{v}\Delta t \qquad \vec{v} = \vec{v} + \vec{a}\Delta t \qquad \vec{a} \text{ constant}$$

Particle Systems

Design choices affect the outcome of a particle system



- Emitter: Specifies where & how particles are created
  - Spawn rate, lifetime, properties
  - Usually randomly generated variation
- Simulation/update
  - Modify particle properties over time: follow trajectory, collide, etc.
- Render: use particles to create visual effect



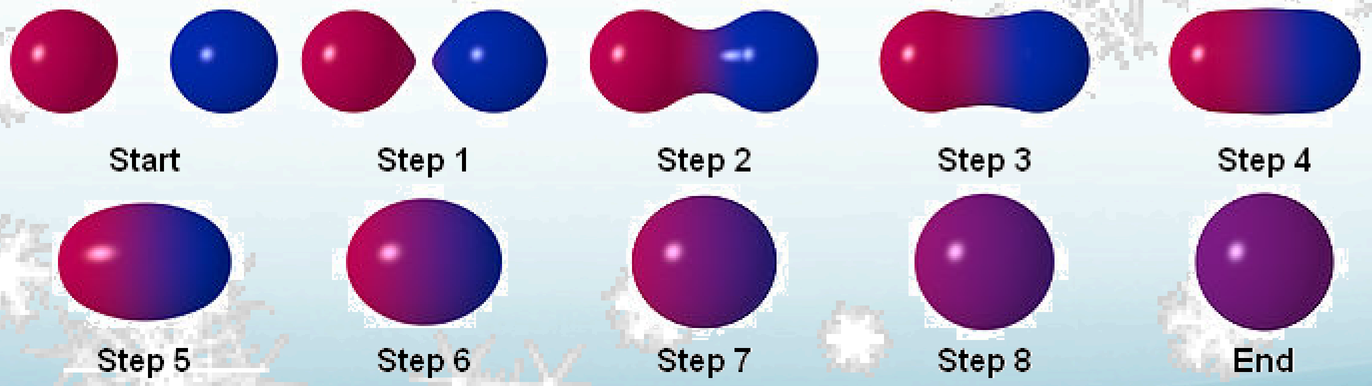
https://giphy.com/gifs/water-sea-Cym7wdsiqNZ2U https://www.digminecraft.com/decoration_recipes/make_monster_spawner.php

Particle Systems

Rendering choices can generate many effects

- **Textured billboard quad** or 3D mesh
- **Metaballs** simulate density
  - Add up density functions centered at each particle
  - Render **isosurface** for water/liquid simulation

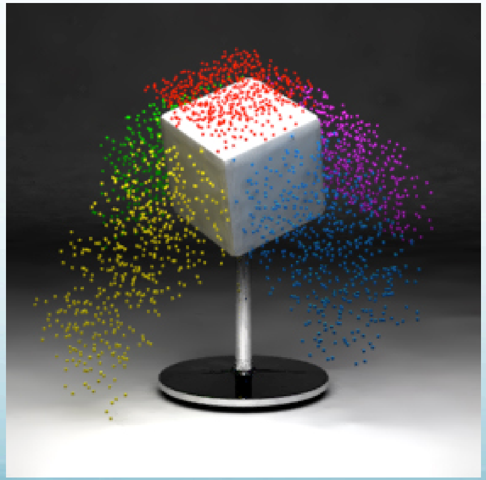
"Marching Cubes" algorithm



Particle Systems

Rendering may be **animated** or **static**

- Animated rendering shows one particle state per frame
  - Water, gas, fire, etc.
- Static rendering shows all particle states at once
  - Hair, fur, cloth, etc.

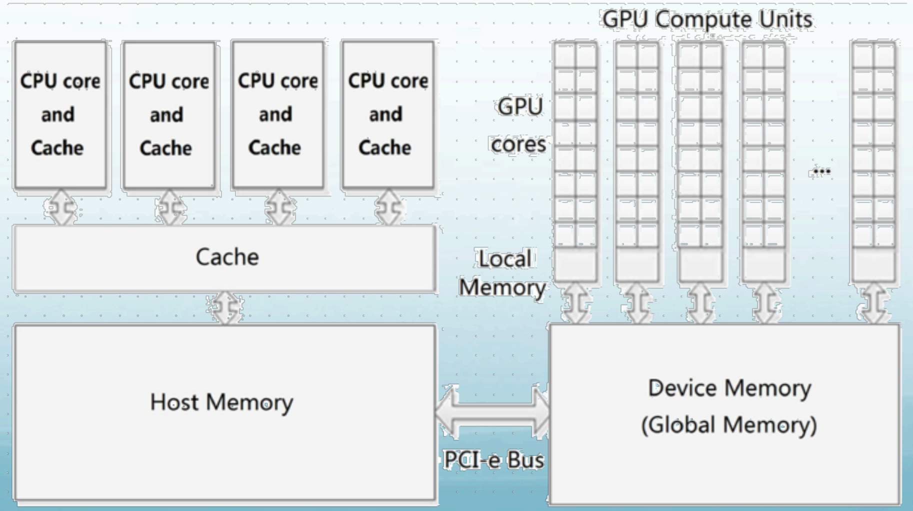
 

https://en.wikipedia.org/wiki/Particle_system

CPU vs GPU?

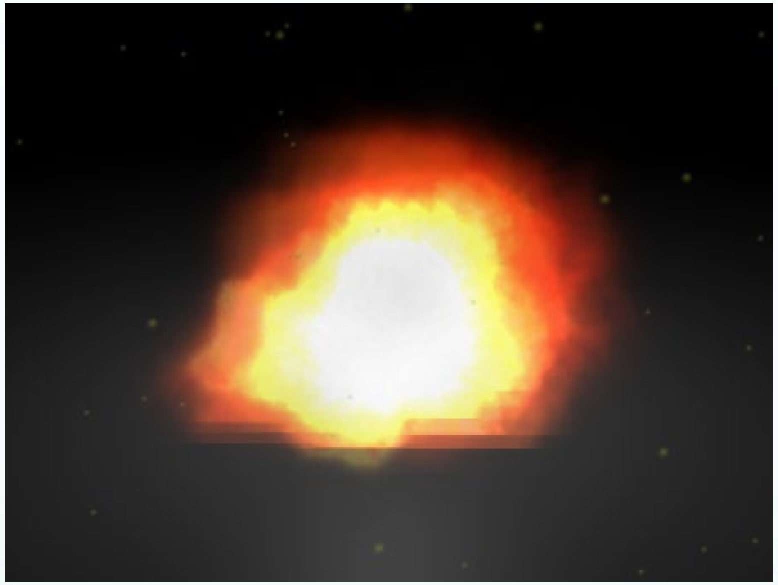
Particles = simple behaviors in large numbers

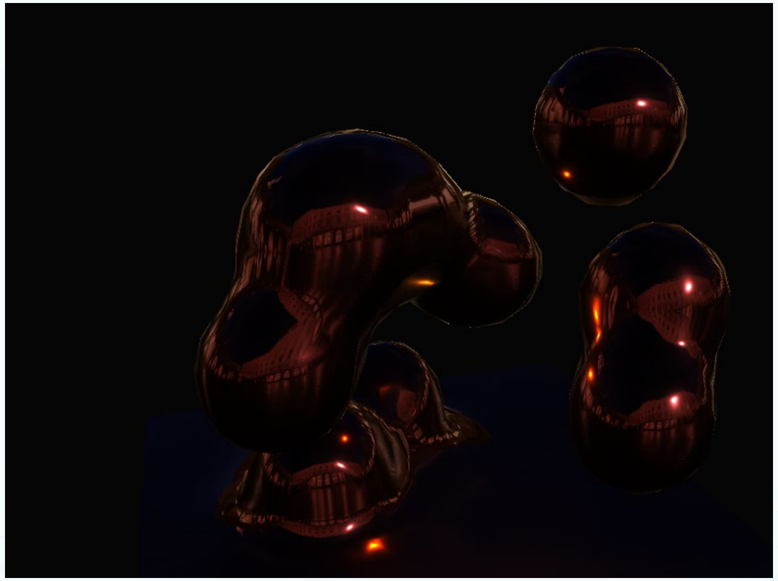
- CPU can maintain limited number of particles (~2k)
- Real power comes from allowing GPU to handle the particle updates
- Limited communication from CPU to GPU: simple mechanics, high-level update



Particle Systems

Best way to see is by demonstration!





https://aerotwist.com/static/tutorials/creating-particles-with-three-js/demo/

https://squarefeet.github.io/ShaderParticleEngine/

https://threejs.org/examples/webgl_marchingcubes.html

https://threejs.org/examples/#webgl_points_dynamic

http://www.spacejack.ca/projects/terra/

Questions

PAUSE NOW & ANSWER

1. How are particle states updated in a Brownian motion model?

   The position is modified by a small random amount

2. What is the advantage to using the same simple state updates across all particles?

   Updates can be computed efficiently on a graphics processing unit (GPU)

3. If we use particles to simulate hair, how can we control the hair properties (curliness, body, etc.)?

   Change the state equations that govern the hair particles

Review

After watching this video, you should be able to…

- Define subsurface scattering and identify cases where it is important for realism
- Describe two techniques for simulating subsurface scattering in rendered images
- Define a particle system in graphics and list three common uses
- Describe how the state model and its update governs the behavior of a particle
- Explain how similar mathematical processes can generate very different effects such as fire, water, hair, and cloth