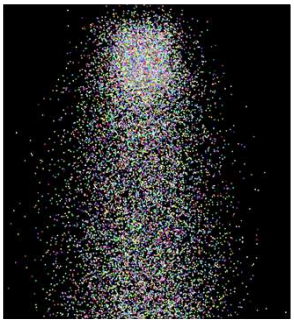
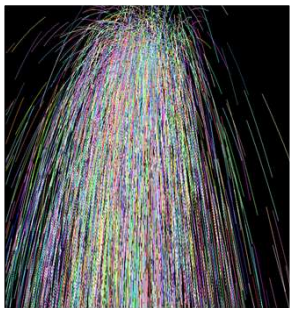
Particle Systems





Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

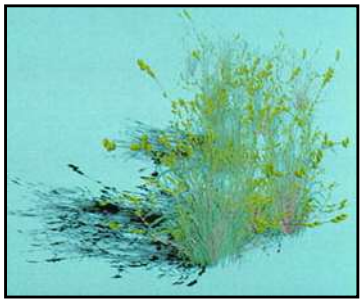
Particle Systems

- Are used to simulate the appearance of particulate, hairy, or fuzzy phenomena.
- Involve the animation of large collections of (perhaps tiny) particles which have various graphics characteristics.
- Were originally developed by Pixar's Bill Reeves for the "Genesis Sequence" in the movie *Star Trek II: The Wrath of Khan*
- Have been used to create effects of fire, smoke, rain, snow, fireworks, disintegration, dust, sand, explosions, flow, waterfalls, stars, comets, plants, hair, fuzz. Surely many more.

Oregon State University
Computer Graphics

3

Particle Systems Examples

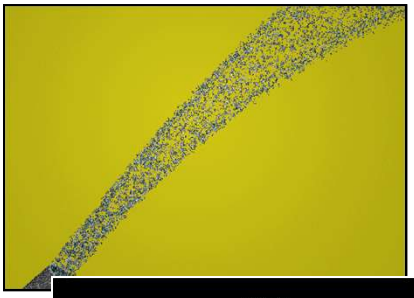


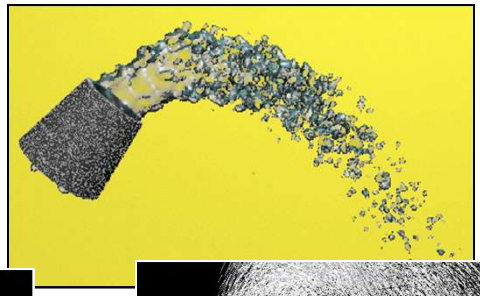
Oregon State University
Computer Graphics

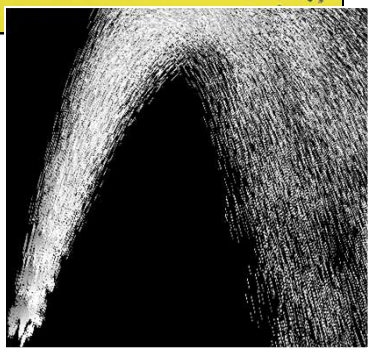
mjb –August 25, 2022

4

Particle Systems Examples







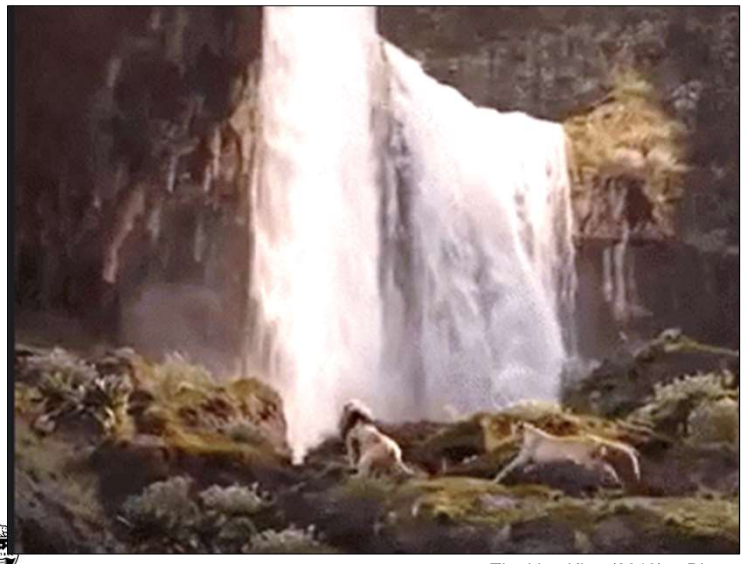
Oregon State University
Computer Graphics

Chuck Evans

mjb –August 25, 2022

5

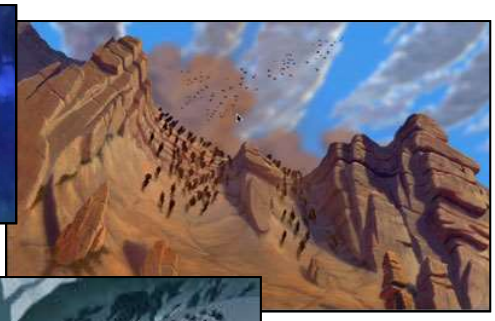
Particle Systems Examples



The Lion King (2019) -- Disney

6

"Particles" Don't Actually Have to Be Particles

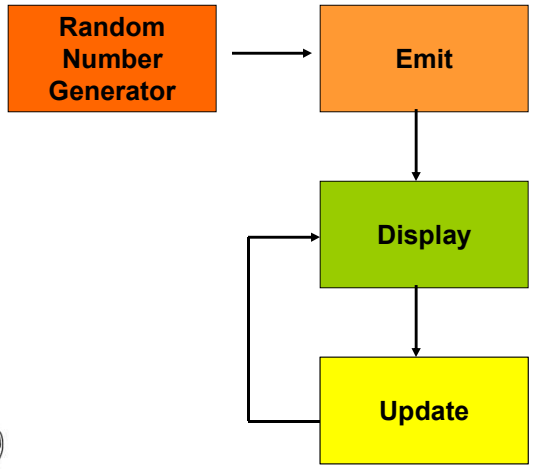


Starlings in Flight

https://www.wired.com/story/stunning-images-of-starlings-in-flight/?bxid=5bd6774e24c17c1048017a0f&cndid=39783882&esrc=AUTO_PRINT&mbid=mbid%3DCRMWIR012019%0A%0A&source=EDT_WIR_NEWSLETTER_0_ENGAGEMENT_ZZ&utm_brand=wired&utm_campaign=aud-dev&utm_content=WIR_Classics_022622&utm_mailing=WIR_Classics_022622&utm_medium=email&utm_source=nl&utm_term=WIR_Daily_TopClickers

The Process

The basic process is this:



The Emitter

The Emitter gives each particle a:

- Birth time
- Death time
- Start location
- Start velocity
- Start color
- Start size
- Start alpha (blending factor)

$$Color = (1-\alpha)Color_0 + \alpha Color_1$$

Plus, any information about how these quantities change over time

Oregon State University
Computer Graphics

Creating Random Values for the Emitter

```
#include <stdlib.h>


float
Ranf( float low, float high )
{
        float r = (float) rand( );              // 0 - RAND_MAX
        float t = r  /  (float) RAND_MAX;       // 0. – 1.

        return   low  +  t * ( high - low );
}


int
Ranf( int ilow, int ihigh )
{
        float low = (float) ilow;
        float high = ceil(  (float) ihigh  );

        return (int) Ranf(low,high);
}
```

Oregon State University
Computer Graphics

The Displayer

And the displayer draws the scene using different graphics techniques such as:

- Dots
- Small line segments
- Polygons with billboarding
- Quads with textures and billboarding
- Sprites
- Spheres, Cubes
- "Rockets"
- Lighting
- Blending
- Smearing

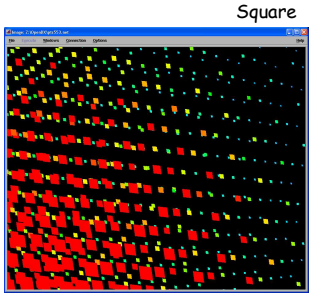
Easiest way:

```
glBegin( GL_POINTS );
        glColor3f(   r0, g0, b0 );
        glVertex3f( x0, y0, z0 );
        . . .
glEnd( );
```

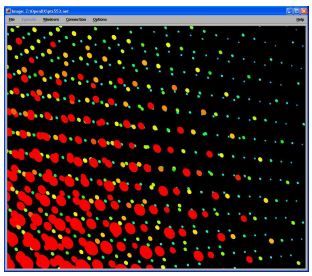
Most efficient way: Vertex Buffer Objects

OpenDX Scalar Glyphs

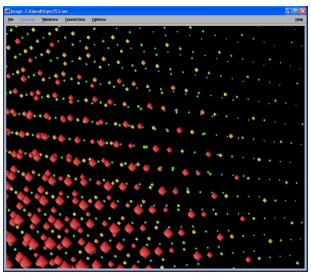
Square



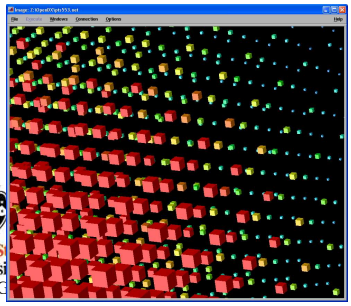
Circle



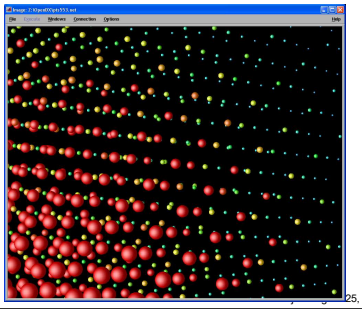
Diamond



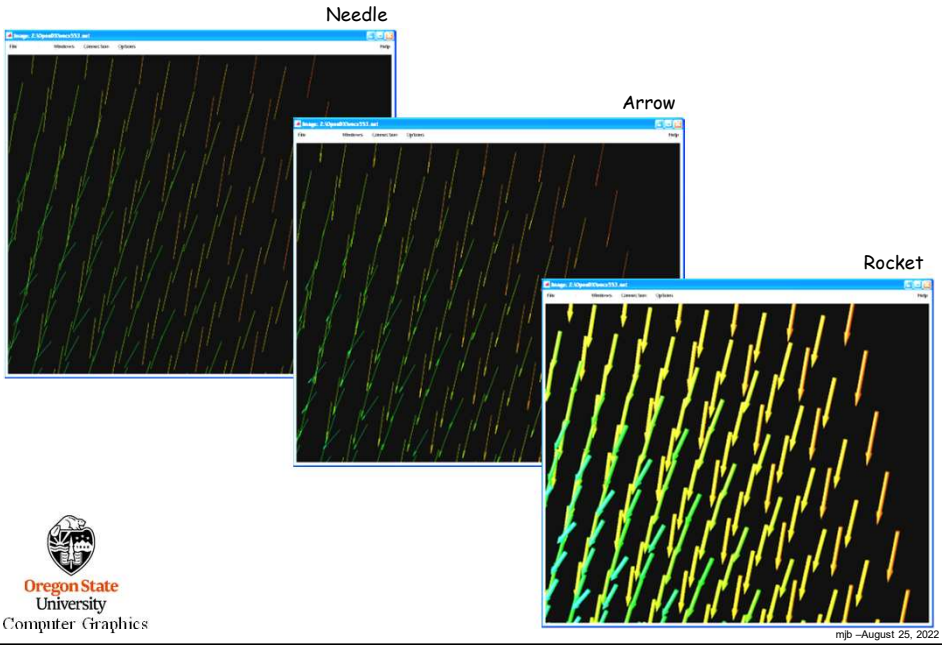
Cube



Sphere

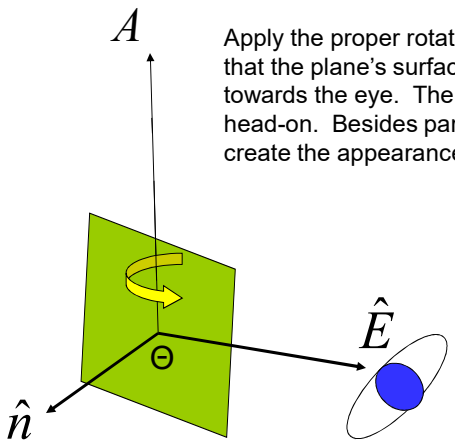


OpenDX Vector Glyphs



Oregon State University
Computer Graphics

Billboarding



Apply the proper rotation about the proper axis such that the plane's surface normal is always pointed towards the eye. The eye always sees the surface head-on. Besides particle systems, this is often used to create the appearance of 3D trees from 2D tree images.

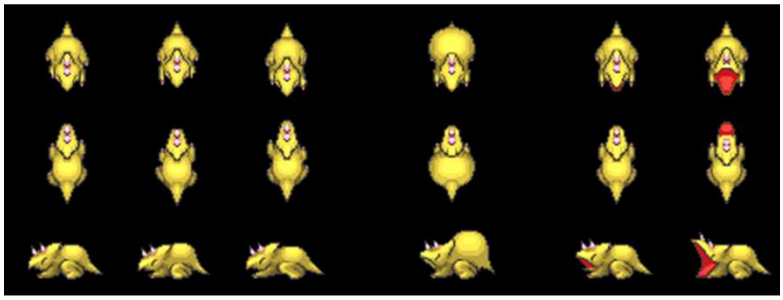
$$A = \hat{n} \times \hat{E}$$

$$\theta = \cos^{-1}\left(\hat{n} \bullet \hat{E}\right)$$

Oregon State University
Computer Graphics

Sprites

A “sprite” is a 3D object pre-rendered to a flat 2D texture and “slipped” into a certain depth in the scene.



http://sdb.drshnaps.com

The Updater

And the simulation updates the:

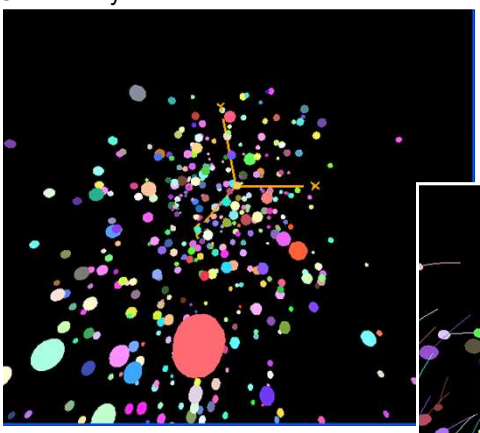
- Position
- Color
- Size
- Alpha
- Interaction with other particles and other objects

Note that these can change as a function of time, position, or anything else

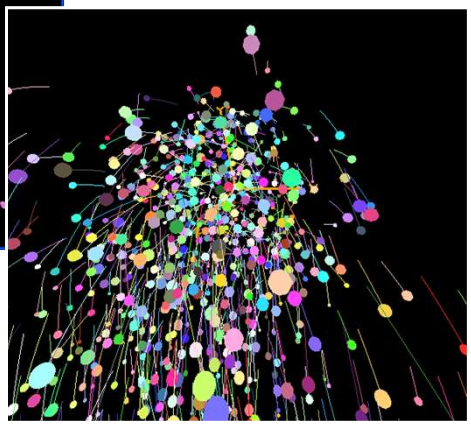
17

Particle Systems

Circles only



Circles with traces



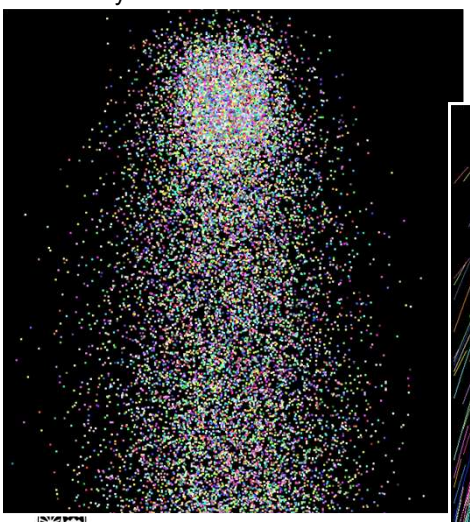
Oregon State University
Computer Graphics

mjb –August 25, 2022

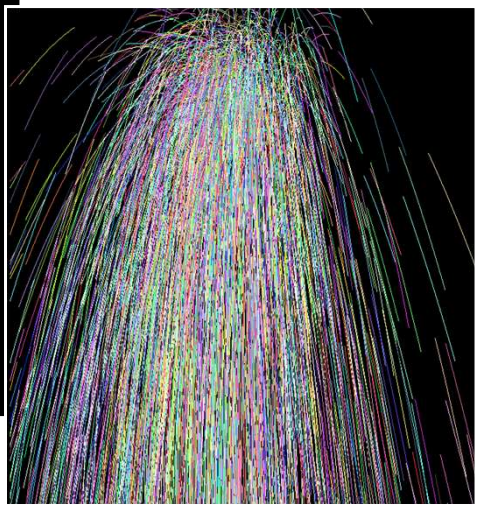
18

Particle Systems

Points only



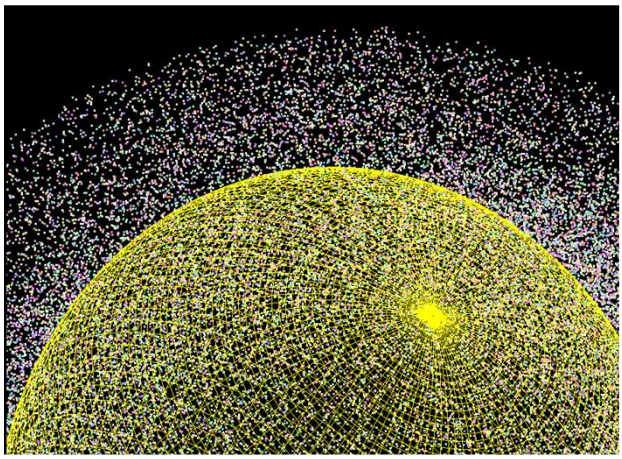
Points with traces



Oregon State University
Computer Graphics

mjb –August 25, 2022

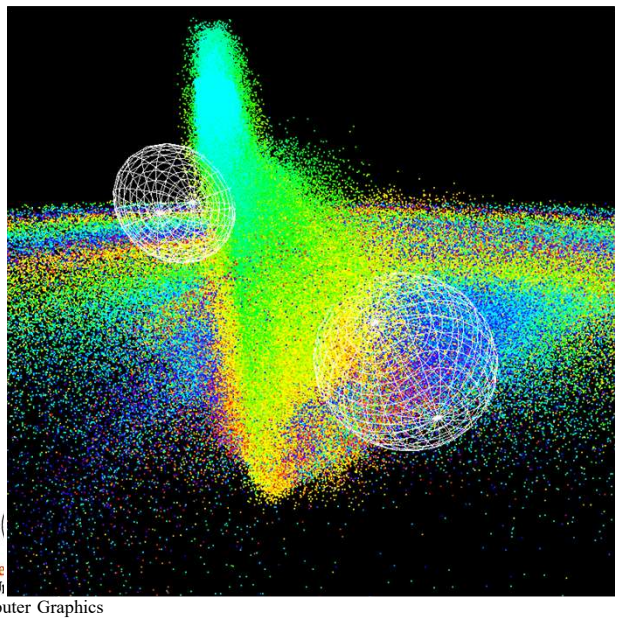
Particle Systems using OpenGL Compute Shaders



1,000,000 particles

1.3 Gparticles / sec

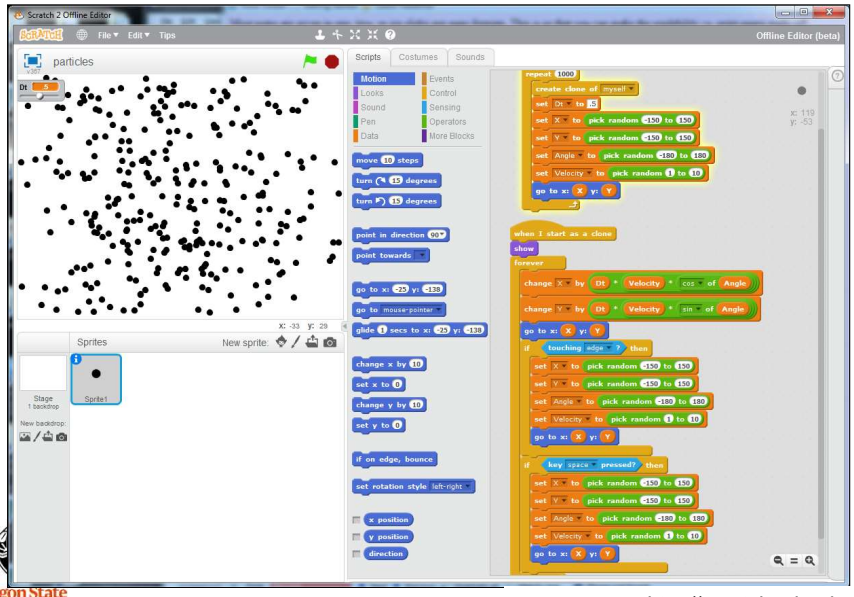
Particle Systems in the OSU Shaders Course



1,000,000 particles

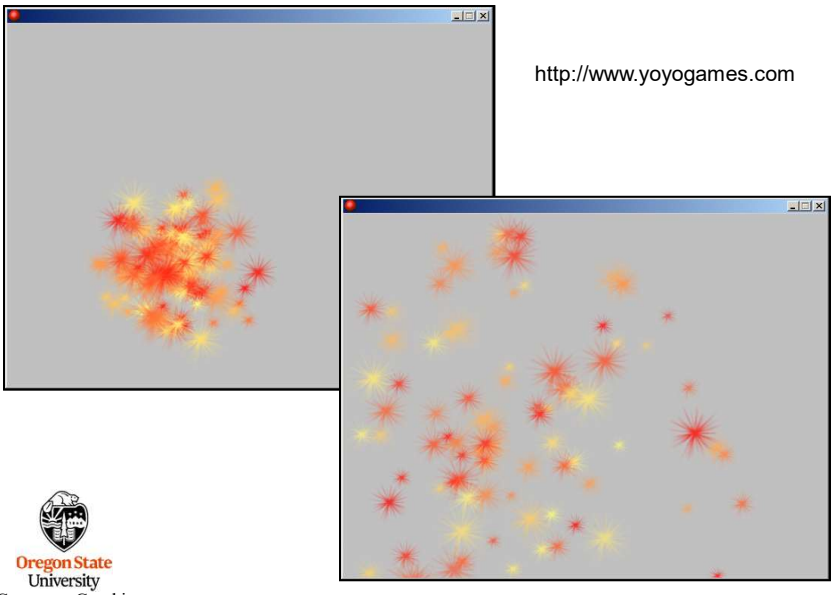
Ben Weiss

Particle Systems using Scratch



http://scratch.mit.edu

Particle Systems using Game Maker



http://www.yoyogames.com