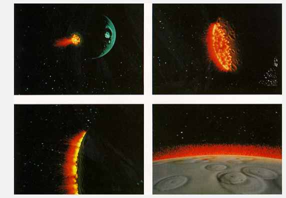
Particle Systems

Particle Systems

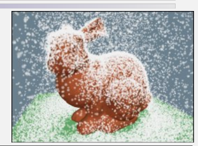
- pioneered in early 1980s
  - invented by William Reeves at ILM for the Genesis effect in *Star Trek II: The Wrath of Khan*

https://www.youtube.com/watch?v=NM1r37zIBOQ

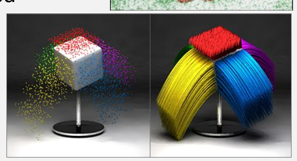


- applications
  - "fuzzy" objects without smooth, well-defined surfaces
    - fire, smoke, clouds, fog, fountains, waterfalls, spray of breaking waves, fireworks, …
  - complex solid objects or solid objects which change shape
    - trees, grass, fur, ...

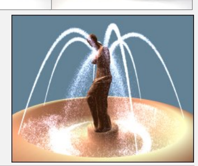
Particle Systems



- idea: entity is modeled as a large collection of individual particles
  - particle is (often) subpixel-sized
  - each particle is individually animated



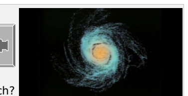
- can be *animated* or *static*
  - for static (or *structured*), entire lifetime of particle is rendered at once



- a different way to model objects
  - cloud of particles defines the volume of the object, rather than modeling the surface
  - objects are not static
  - objects are not deterministic

https://en.wikipedia.org/wiki/Particle_system
http://www.2ld.de/gh2004/

Some Milestones in Movies



- *Cosmos* (1979)
  - use of particles to model galaxies

https://www.youtube.com/watch?v=uWhsLmFFhTI

- *Star Trek II: The Wrath of Khan* (1982)
  - invention of particle systems for fuzzy object modeling

https://www.youtube.com/watch?v=NM1r37zIBOQ

- *Twister* (1996)
  - 5-10 million particles for the basic tornado
  - a mix of particle systems and hand animation

https://www.youtube.com/watch?v=IzYXUVmt9So

- *The Perfect Storm* (2000)
  - ocean simulation using particle systems
  - at times, billions of particles per frame

https://www.youtube.com/watch?v=yOMrXCT5nBg

- *Monsters, Inc.* (2001)
  - developments in algorithms and a physics engine to handle realistic fur and cloth on a large scale (2,320,413 hairs on Sulley's body)

https://www.youtube.com/watch?v=6tCxnHCqqxg

Particle Systems

- many things have been described as particle systems
- key characteristics
  - a *collection of particles*
  - a *random element* (within limits)
    - typically randomness in particle attributes such as position, velocity, color

Particle System Evolution

- repeat
  - generate new particles and add to system
    - initialize individual attributes for new particles
  - remove particles which have exceeded their lifetime
  - move/update current particles
  - draw current particles

  until done

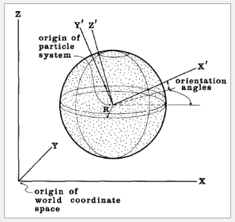
Particle Generation and Initialization

- the number of new particles introduced at a given frame is random, within limits
- initial values must be assigned for each attribute when particles are created
  - position and velocity (direction and speed)
  - size and shape
  - color and transparency
  - lifetime
  - path
  - ...

Particle Initialization

- initial position and direction of movement
  - defined by *emitter* associated with particle system

example: spherical emitter
initial position is within sphere
initial direction is away from center



W. Reeves, “Particle Systems – A Technique for Modeling a Class of Fuzzy Objects”, ACM Transactions on Graphics, April 1983

- initial speed, color, transparency, size typically determined randomly given average and variability
  - average and variability can be global or emitter-specific

Particle Initialization

- particle shape is a parameter to particle system
  - spherical
  - rectangular
  - streaked spherical (for motion blur)
  - ...

Particle Extinction

- particles can be removed from the system for various reasons
  - lifetime is exceeded
  - particle no longer contributes to image (e.g. intensity is too low)
  - particle has moved too far from initial position
  - ...

Particle Dynamics

- specifies how values of particle attributes change over time
- for position, can specify dynamics or other patterns of change
  - if dynamics, `position = position+speed`
  - ignore particle-particle collisions
- color, transparency, size typically change according to global rate-of-change parameters
  - steady change pattern

Additional Complexity

- system described so far is a bare-bones particle system
- many extensions and variations are possible
  - additional attributes
    - e.g. acceleration
    - vary rate-of-change parameters with each particle instead of being global
    - …
  - more complex emitters
  - hierarchical particle systems
  - solid particles
    - need to consider collisions – with other objects in the scene and/or other particles
  - ...

Plume Particle System

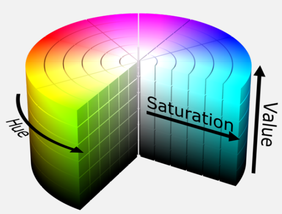
```
// spark properties
float x, y;                   // position
float xspeed, yspeed;         // speed
float hue, saturation, brightness;  // color (HSB)
float transparency;           // transparency
float t;                      // noise parameter

// emitter properties
float emitterX;                 // center of emitter
float emitterSpeed;             // speed of emitter
float emitterHue;               // hue of particles currently being emitted
```

single spark starter code – needs to be array-ified!

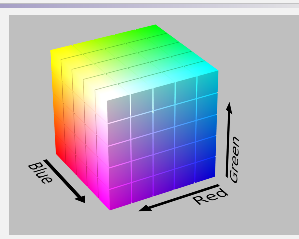

```
void setup () {
  size(1000, 400);
  colorMode(HSB);

  // start offscreen and invisible
  x = 0;
  y = height+10;
  xspeed = 0;
  yspeed = 0;
  hue = 0;
  saturation = 0;
  brightness = 0;
  transparency = -1;
  t = 0;

  emitterX = 10;
  emitterSpeed = 0.25;
  emitterHue = 0;
}
```

single spark starter code – needs to be array-ified!

About HSB Color



HSB/HSV color model



RGB color model

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

- HSB/HSV aligns more closely with how humans perceive and describe colors
  - *hue* captures the color itself (red, orange, yellow, etc)
  - *saturation* captures the intensity of the color (how far from gray)
  - *brightness* or *value* capture how light or dark the color is
- HSB is used for this sketch because it makes it easier to cycle through colors for the plume without affecting how intense or bright the color is


```
void draw () {
  background(0);

  // draw spark
  noStroke();
  fill(hue, saturation, brightness, transparency);
  ellipse(x, y, 2, 2);

  // update spark
  if ( transparency >= 0  ) {         // when spark is vis
    x = x+xspeed+map(noise(t), 0, 1, -.7, .7);   // noise to wiggle spark as it rises
    y = y+yspeed;
    xspeed = xspeed-.01*xspeed;
    transparency = transparency-1;
    t = t+.005;

  } else if ( random(0, 200) <= 1 ) {   // reset spar
    x = emitterX+random(-5, 5);
    y = height;
    xspeed = emitterSpeed;
    yspeed = random(-1.5, -3.5);
    t = random(0, 1000);
    hue = random(emitterHue-10, emitterHue+10);
    saturation = random(200, 255);
    brightness = random(200, 255);
    transparency = random(200, 255);
  }

  // update emitter
  emitterX = emitterX+emitterSpeed;
  emitterHue = emitterHue+emitterSpeed*255/width;  // so it cycles through colors across the
}
```

single spark starter code – needs to be array-ified!

draw current sparks

transparency is used instead of a separate lifetime – a spark expires when its transparency drops below 0

move/update unexpired sparks

an expired spark is replaced by a new one with a certain probability, allowing for a staggered start at the beginning when all sparks start expired

generate new sparks to replace expired ones

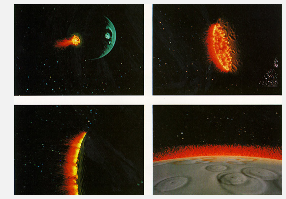
Particle Hierarchies

- each particle is itself an emitter
- parameters of parent particle are inherited (with variations) by child emitter
  - e.g. position of parent particle determines origin of child emitter
  - e.g. average color and variance of parent particle are used to pick random (within range) average color and variance for child emitter

Example: The Genesis Effect

https://www.youtube.com/watch?v=NM1r37zIBOQ

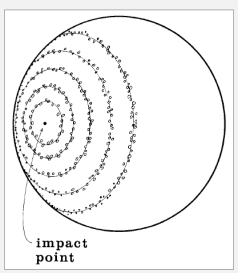
Star Trek II: The Wrath of Khan (1982)



- two-level particle system
  - first level distributes emitters around impact point on planet's surface
  - second level handles explosion out from surface

The Genesis Effect: Particles

- first level particle system distributes emitters around point on planet's surface

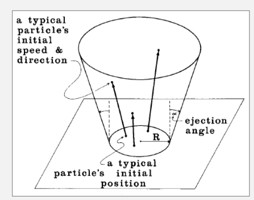


W. Reeves, "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics, April 1983

- radius of ring where emitters are produced depends on time (frame)
- number of new emitters in each ring is based on ring circumference and density parameter

The Genesis Effect: Particles

- second level particle systems handle explosion out from surface



W. Reeves, "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics, April 1983

- emitter is circle centered at first-level particle position
- number of new particles per frame depends on surface area covered by emitter
- initial velocity of particles is away from planet's surface, within cone defined by ejection angle
- parameters (average color, rate of color change, emitter radius, etc) based on parent particle system, but vary randomly

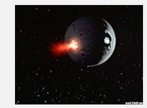
The Genesis Effect: Particle Extinction

- particles are removed when…
  - lifetime expires
  - intensity falls below minimum intensity
  - they fall below the surface of planet

The Genesis Effect: Rendering

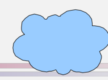
- color of second level particles is mostly red, with a small amount of green and blue
  - when only a few particles overlap a single pixel, pixel appears red
  - when many particles overlap a single pixel, green and blue contributions become significant
  - over time green and blue drop off faster than red

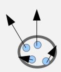
result is white hot in very center, cooling to yellow-orange and finally red towards edge



- motion blur
  - particles actually drawn as straight line between position at time *t* and at time *t+dt*/2 (halfway through frame)

Example: A Growing Cloud



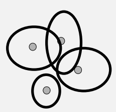


parent emitter defines where child emitters will be located and how they move

location and size of parent particles define origin and size of child emitters
actual cloud particles are generated within child emitters

early on, parent particles are small and close together

child emitters are also small



as the system evolves, the parent particles grow and move apart

child emitters also grow larger

newly-generated particles are within the shaded regions

how the cloud appears – particles have not moved much

newly-generated particles are within the shaded regions

cloud appears filled in because older particles still exist (and are also moving)

- child emitters control billowing of each region of the cloud
- parent emitter/particles control relationship between cloud parts and overall cloud movement