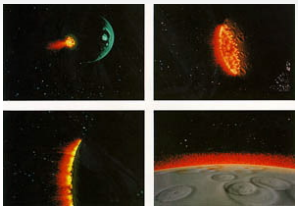
Particle Systems

Particle Systems

- Early method for representing amorphous, "fuzzy" objects
- Fire, water, clouds, etc.
- Develop by William Reeves at the Lucasfilm Computer Graphics Project
- Used in Star Trek II: The Wrath of Khan in 1982



Video

Idea

- Control the movement of simple "particles"
  - Small dots on the screen
- Many, many particles can be combined to create objects
- Particles have properties:
  - Color
  - Transparency
  - Position
  - Velocity
  - Etc.

Particle Systems

- Particle Emission
  - Particles are "born" or emitted from one or more source
  - Randomness is often used
- Particle Life
  - Particles move through the scene
  - Forces like gravity can act on the particles
- Particle Death
  - Particles may fade or be removed from the scene

Examples

- Karl Sims
- Daniel Franke and Cedric Kiefer

Simple Particle Class

```
class NParticle
{
  //Particle position
  float m_xPos = 0;
  float m_yPos = 0;
  //particle velocity
  float m_xVel, m_yVel;
  //particle acceleration.  Default to gravity
  float m_xAccel = 0, m_yAccel = 9.8;
```

Simple Particle Class

```
//size of particle
  float m_diameter = 10;

  //scale allows you to adjust the overall speed (basically
  //changing the metre to pixel mapping)
  float m_scale = 60;
```

Simple Particle Class

```
  //empty constructor
  NParticle(float xPos, float yPos, float xVel, float yVel)
  {
    m_xPos = xPos;
    m_yPos = yPos;
    m_xVel = xVel;
    m_yVel = yVel;
  }
```

Simple Particle Class

```
  //draw the particle
  void drawMe()
  {
    ellipseMode(CENTER);
    ellipse(m_xPos, m_yPos, m_diameter, m_diameter);
  }
```

Simple Particle Class

```
  //Update the velocity and position of particle
  //timeStep is the amount of time that has passed since
  //the last update
  void update(float timeStep)
  {
    //Euler integration of system state one time step
    m_xVel += m_xAccel * timeStep;
    m_yVel += m_yAccel * timeStep;
    m_xPos += m_xVel * timeStep * m_scale;
    m_yPos += m_yVel * timeStep * m_scale;
  }
}
```

Using Particle Class

```
void draw()
{
  background(255);
  line(0, gGroundPlane, width, gGroundPlane);

  testParticle.update(1.0/frameRate);
  testParticle.drawMe();

  float edgePos = testParticle.m_yPos +
testParticle.m_diameter/2.0;
```

Adding In Ground Reaction

```
float edgePos = testParticle.m_yPos + testParticle.m_diameter/2.0;
if(edgePos > gGroundPlane)
{

  testParticle.m_yPos += gGroundPlane-edgePos;
  testParticle.m_yVel *= -gRestitution;
}
```

Examples

- Single particle
- Multiple particles with different emitters