Announcements

- Sunday, November 22nd at 11:59pm:
  - Homework Project 3 due
- Wednesday, November 25th at 1pm:
  - Discussion Project 4
- Sunday, November 29th at 11:59pm:
  - Homework Project 3 late deadline
Particle Systems
Particle Systems

- Used for:
  - Fire/sparks
  - Rain/snow
  - Water spray
  - Explosions
  - Galaxies
Internal Representation

- Particle system is collection of a number of individual elements (particles)
  - Controls a set of particles which act autonomously but share some common attributes

- Particle Emitter: Source of all new particles
  - 3D point
  - Polygon mesh: particles’ initial velocity vector is normal to surface

- Particle attributes:
  - position (3D)
  - velocity (vector: speed and direction)
  - color + opacity
  - lifetime
  - size
  - shape
  - weight
Dynamic Updates

- Particles change position and/or attributes with time
- Initial particle attributes often created with random numbers
- Frame update:
  - Parameters: simulation of particles, can include collisions with geometry
    - Forces (gravity, wind, etc) accelerate a particle
    - Acceleration changes velocity
    - Velocity changes position
- Rendering:
  - GL_POINTS
  - GL_POINT_SPRITE
  - Point shader

Source: http://www.particlesystems.org/
uniform mat4 u_MVPMatrix;
uniform vec3 u_cameraPos;

// Constants (tweakable):
const float minPointScale = 0.1;
const float maxPointScale = 0.7;
const float maxDistance = 100.0;

void main()
{
    // Calculate point scale based on distance from the viewer
to compensate for the fact that gl_PointSize is the point
// size in rasterized points / pixels.
float cameraDist = distance(a_position_size.xyz, u_cameraPos);
float pointScale = 1.0 - (cameraDist / maxDistance);
pointScale = max(pointScale, minPointScale);
pointScale = min(pointScale, maxPointScale);

    // Set GL globals and forward the color:
gl_Position = u_MVPMatrix * vec4(a_position_size.xyz, 1.0);
gl_PointSize = a_position_size.w * pointScale;
v_color = a_color;
}
Demo

- Particle system in WebGL:
References

- Tutorial with source code by Bartlomiej Filipek, 2014:
  - [Tutorial](http://www.codeproject.com/Articles/795065/Flexible-particle-system-OpenGL-Renderer)

- Articles with source code:
    - [Article](http://www.darwin3d.com/gamedev/articles/col0798.pdf)
  - John Van Der Burg: “Building an Advanced Particle System”, Gamasutra, June 2000
    - [Article](http://www.gamasutra.com/view/feature/3157/building_an_advanced_particle_.php)

- Founding scientific paper:
  - Reeves: “Particle Systems - A Technique for Modeling a Class of Fuzzy Objects”, ACM Transactions on Graphics (TOG) Volume 2 Issue 2, April 1983
    - [Paper](https://www.evl.uic.edu/aej/527/papers/Reeves1983.pdf)
Collison Detection
Collision Detection

- **Goals:**
  - Physically correct simulation of collision of objects
    - Not covered here
  - Determine if two objects intersect
  - Slow calculation because of exponential growth $O(n^2)$:
    - $\#$ collision tests $= n^2(n-1)/2$
Intersection Testing

- **Purpose:**
  - Keep moving objects on the ground
  - Keep moving objects from going through walls, each other, etc.

- **Goal:**
  - Believable system, does not have to be physically correct

- **Priority:**
  - Computationally inexpensive

- **Typical approach:**
  - Spatial partitioning
  - Object simplified for collision detection by one or a few
    - Points
    - Spheres
    - Axis aligned bounding box (AABB)
  - Pairwise checks between points/spheres/AABBs and static geometry
Sweep and Prune Algorithm

- Sorts bounding boxes
- Not intuitively obvious how to sort bounding boxes in 3-space
- Dimension reduction approach:
  - Project each 3-dimensional bounding box onto the x, y and z axes
  - Find overlaps in 1D: a pair of bounding boxes can overlap if and only if their intervals overlap in all three dimensions
    - Construct 3 lists, one for each dimension
    - Each list contains start/end point of intervals corresponding to that dimension
    - By sorting these lists, we can determine which intervals overlap
    - Reduce sorting time by keeping sorted lists from previous frame, changing only the interval endpoints
Sweep and Prune Example
Collision Map (CM)

- 2D map with information about where objects can go and what happens when they go there
- Colors indicate different types of locations
- Map can be computed from 3D model, or hand drawn with paint program
- Granularity: defines how much area (in object space) one CM pixel represents
Collision Map Examples