CSE 167:
Introduction to Computer Graphics
Lecture #16: Particle Systems

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2012

Announcements

- Wednesday, Nov 28: Last day for late grading of project 6
- Thursday, Nov 29: Midterm exam #2
- Friday, Nov 30: Final project summary due
- Thursday, Dec 13: Final project presentations in EBU-3B room 1202, 3-6pm
- Looking for TAs and Tutors for CSE190: 3D UI

Demo

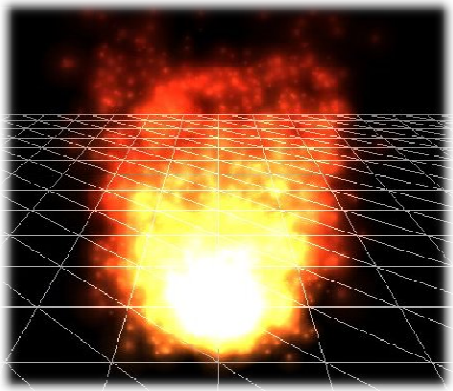
- Geisel Returns Home
  - By Robert Pardridge, Christopher Jenkins, Kevin Reynolds
  - “It is well known that Geisel Library resembles a huge spaceship. Almost every UCSD student has this thought at least once while walking past the library.“

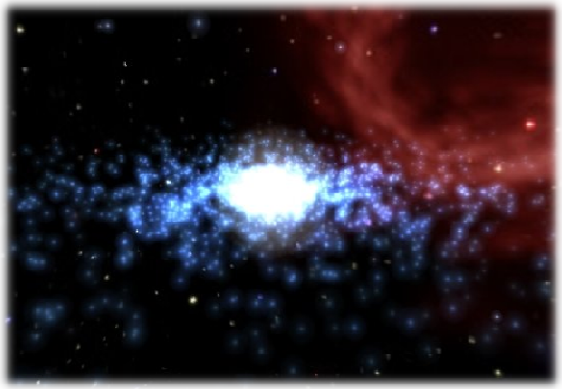
Lecture Overview

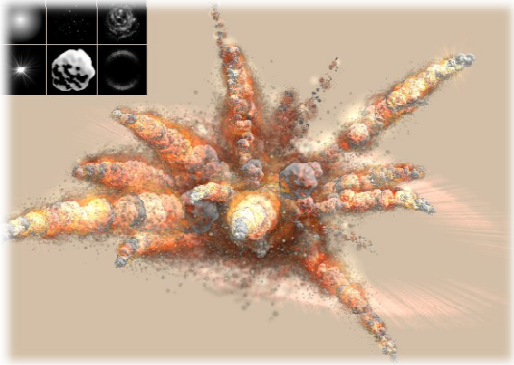
- Particle Systems
- Collision Detection

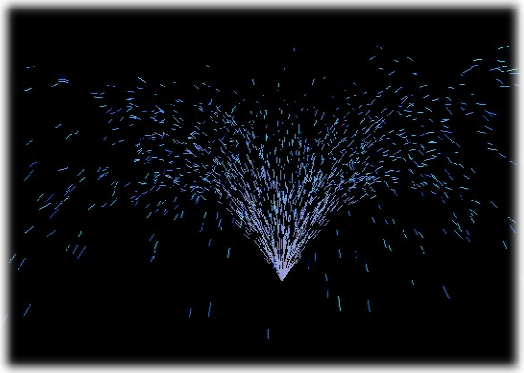
Particle Systems

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







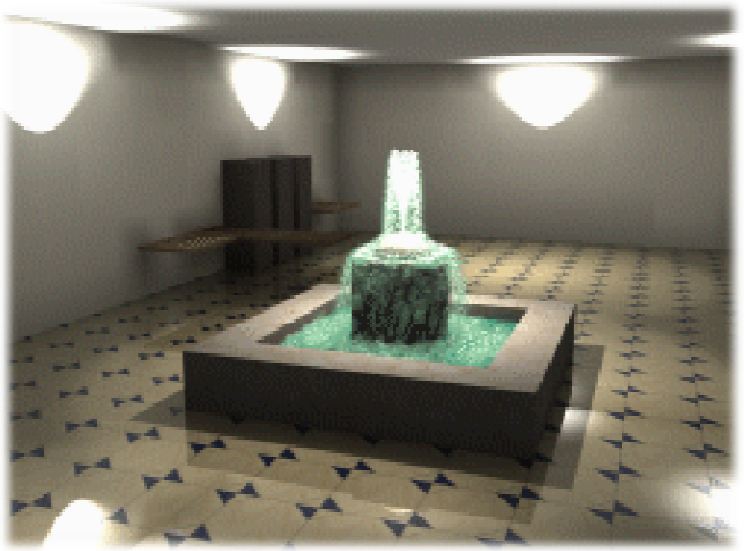


Internal Representation

- 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: display as
    - OpenGL points
    - (Textured) billboarded quads
    - Point sprites



Source: http://www.particlesystems.org/

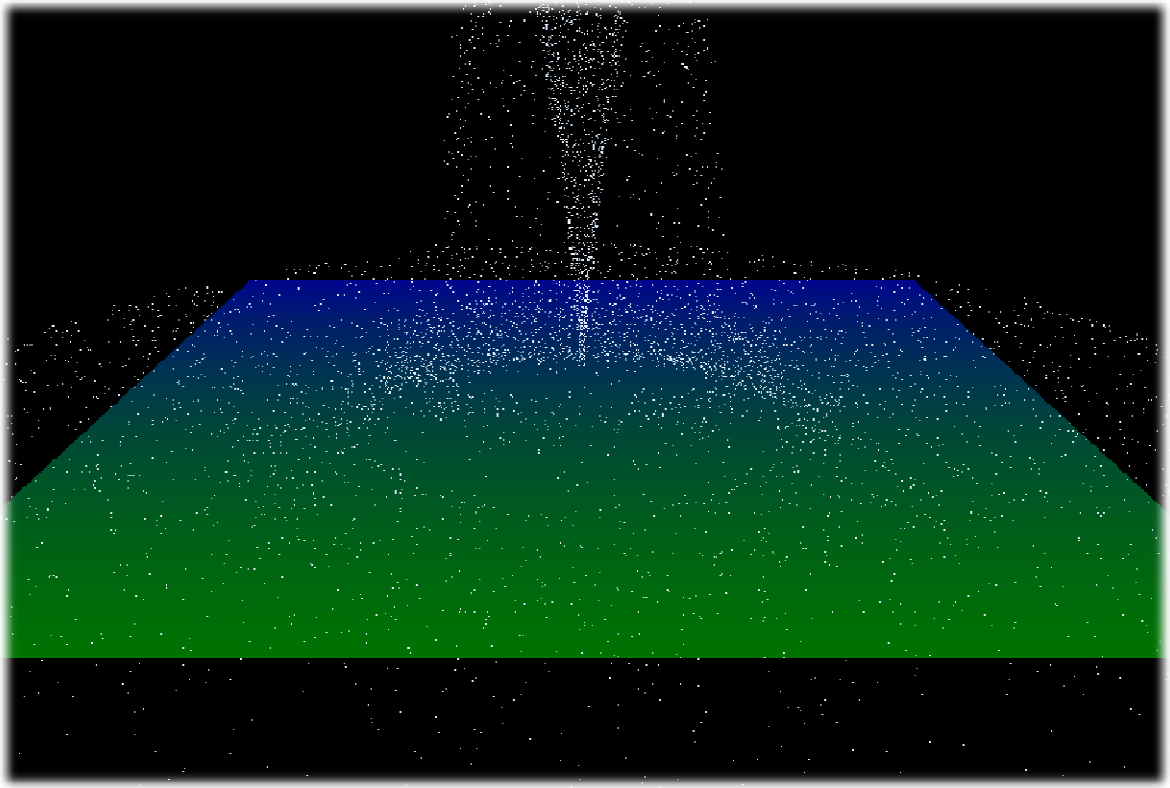
Point Sprite

- **Screen-aligned element of variable size**
- **Defined by single point**
- **Sample code:**

```
glTexEnvf(GL_POINT_SPRITE, GL_COORD_REPLACE, GL_TRUE);
glEnable(GL_POINT_SPRITE);
glBegin(GL_POINTS);
   glVertex3f(position.x, position.y, position.z);
glEnd();
glDisable(GL_POINT_SPRITE);
```

Demo

- Source: http://www.particlesystems.org/Distrib/Particle221Demos.zip



References

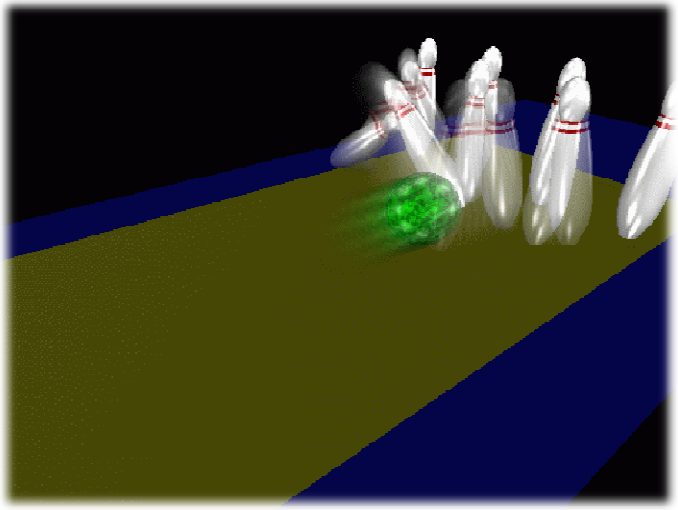
- Free particle systems API (not for final project):
  - http://particlesystems.org/
- On-line tutorial:
  - http://www.naturewizard.com/tutorial08.html
- Initial scientific paper:
  - Reeves: "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics (TOG) Volume 2 Issue 2, April 1983
- Article with source code:
  - Jeff Lander: "The Ocean Spray in Your Face", Game Developer, July 1998, http://www.darwin3d.com/gamedev/articles/col0798.pdf
- John Van Der Burg: "Building an Advanced Particle System", Gamasutra, June 2000
  - http://www.gamasutra.com/view/feature/3157/building_an_advanced_particle_.php

Lecture Overview

- Particle Systems
- Collision 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*(n-1)/2



Intersection Testing

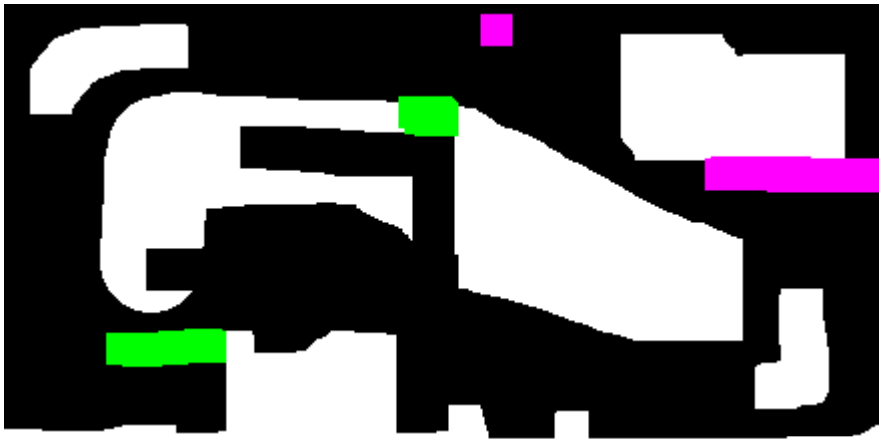
- Purpose:
  - Keep moving objects on the ground
  - Keep moving objects from going through walls, each other, etc.
- Goal:
  - Believable system, does not have to be physically correct
- Priority:
  - Computationally inexpensive
- Typical approach:
  - Spatial partitioning
  - Object simplified for 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

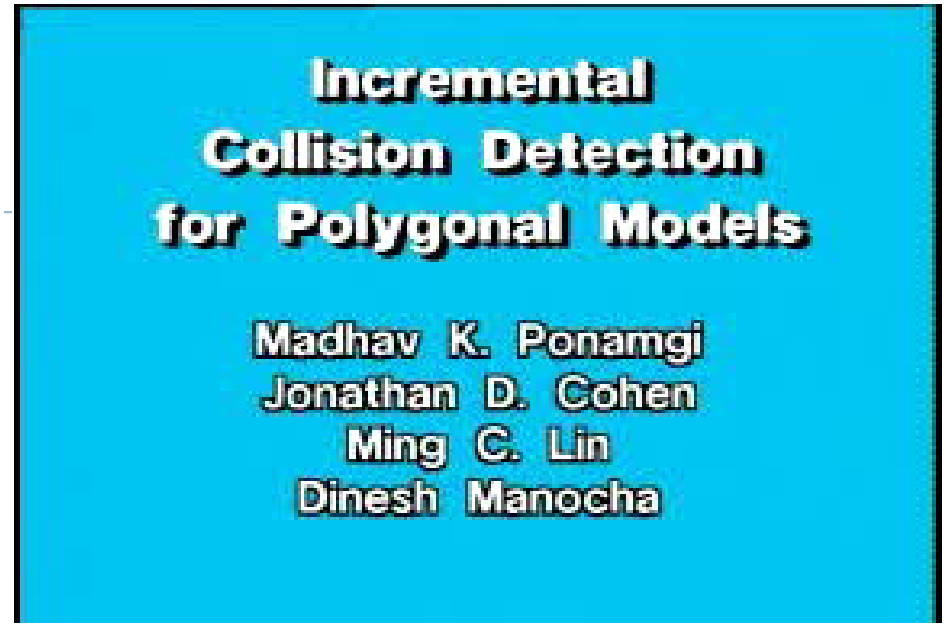
- Sorts bounding boxes
- Not intuitively obvious how to sort bounding boxes in 3-space
- Dimension reduction approach:
  - Project each 3-dimensional bounding box onto the x,y and z axes
  - Find overlaps in 1D: a pair of bounding boxes can overlap if and only if their intervals overlap in all three 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
- Alternative: project bounding boxes onto coordinate axis planes and look for overlaps in 2D

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



References



- I-Collide:
  - Interactive and exact collision detection library for large environments composed of convex polyhedra
  - http://gamma.cs.unc.edu/I-COLLIDE/
- OZ Collide:
  - Fast, complete and free collision detection library in C++
  - Based on AABB tree
  - http://www.tsarevitch.org/ozcollide/