Lecture 29 of 41

Lab 5b: Particle Systems

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:

Today: Particle System Handout
Next class: §5.3, Eberly 2e – see http://bit.ly/ieUq45; CGA Handout
Wikipedia, *Particle System*: http://bit.ly/hzZofl

Lecture Outline

- Reading for Last Class: §9.1, Eberly 2e; Particle System Handout
- Reading for Today: Particle System Handout
- Reading for Next Class: §5.3, Eberly 2e; CGA Handout
- Last Time: Collision Response, Particle Systems
  - Collision handling, concluded: response
    - Impulse *vs.* force
    - Compression & restitution
    - Bounce
    - Friction
  - Simulation of Processes, Simple Physical Bodies
  - Events: birth (emission), collision, death
  - Properties: mass, initial velocity, lifetime
- Today: Lab on Particle Systems; Dissection of Working Program
- Next Class: Animation Part 3 of 3 – Inverse Kinematics

Where We Are

21	**Lab 4a: Animation Basics**	**Flash animation handout**
22	Animation 2: Rotations; Dynamics, Kinematics	Chapter 17, esp. §17.1 – 17.2
23	Demos 4: Modeling & Simulation; Rotations	Chapter 10[1], 13[2], §17.3 – 17.5
24	**Collisions 1: axes, OBBs, Lab 4b**	**§2.4.3, 8.1, GL handout**
25	Spatial Sorting: Binary Space Partitioning	Chapter 6, esp. §6.1
26	**Demos 5: More CGA; Picking; HW/Exam**	**Chapter 7[2]; § 8.4**
27	**Lab 5a: Interaction Handling**	**§ 8.3 – 8.4; 4.2, 5.0, 5.6, 9.1**
28	Collisions 2: Dynamic, Particle Systems	§ 9.1, particle system handout
	Exam 2 review; Hour Exam 2 (evening)	**Chapters 5 – 6, 7[2] – 8, 12, 17**
29	**Lab 5b: Particle Systems**	**Particle system handout**
30	**Animation 3: Control & IK**	**§ 5.3, CGA handout**
31	Ray Tracing 1: intersections, ray trees	Chapter 14
32	Lab 6a: Ray Tracing Basics with POV-Ray	RT handout
33	Ray Tracing 2: advanced topic survey	Chapter 15, **RT handout**
34	Visualization 1: Data (Quantities & Evidence)	Tufte handout (1)
35	**Lab 6b: More Ray Tracing**	**RT handout**
36	Visualization 2: Objects	Tufte handout (2 & 4)
37	Color Basics; Term Project Prep	**Color handout**
38	Lab 7: Fractals & Terrain Generation	**Fractals/Terrain handout**
39	**Visualization 3: Processes; Final Review 1**	Tufte handout (3)
40	**Project presentations 1; Final Review 2**	**–**
41	Project presentations 2	–
	Final Exam	**Ch. 1 – 8, 10 – 15, 17, 20**

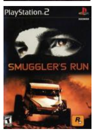
Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review; and the green-shaded entry, that of the term project.

Green, **blue** and **red** letters denote **exam review**, **exam**, and **exam solution review** dates.

Acknowledgements: 3-D Particle Systems

Hubert Pak Ho Shum
Postdoctoral Researcher
Advanced Science Institute
RIKEN (理研)

http://hubertshum.com/info/



Steve Rotenberg
Visiting Lecturer
Graphics Lab
University of California – San Diego
CEO/Chief Scientist, PixelActive
http://graphics.ucsd.edu

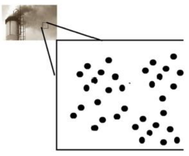
Xiaoyu Zhang
Assistant Professor, Computer Science
California State University – San Marcos
http://public.csusm.edu/xiaoyu/

Rahul Malhotra
Senior Software Engineer, Overstock.com

Review [1]: Particle Emitters & Attributes

Each new particle has the following attributes:

- ☐ initial position
- ☐ initial velocity (speed and direction)
- ☐ initial size
- ☐ initial color
- ☐ initial transparency
- ☐ shape
- ☐ lifetime



Adapted from slide © 2008 H. P. H. Shum, RIKEN (理研)
Computer Animation, http://bit.ly/ig6KTK

Review [2]: Impacts

- When two solid objects collide (such as a particle hitting a solid surface), forces are generated at the impact location that prevent the objects from interpenetrating
- These forces act over a very small time and as far as the simulation is concerned, it's easiest to treat it as an instantaneous event
- Therefore, instead of the impact applying a force, we must use an impulse

Adapted from slides © 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005, http://bit.ly/f0ViAN

Review [3]: Impulse

- An impulse can be thought of as the integral of a force over some time range, which results in a finite change in momentum:

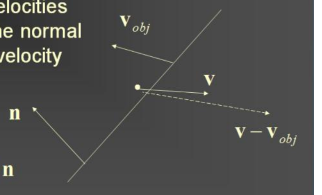
$$\mathbf{j} = \int \mathbf{f}\,dt = \Delta\mathbf{p}$$

- An impulse behaves a lot like a force, except instead of affecting an object's acceleration, it directly affects the velocity
- Impulses also obey Newton's Third Law, and so objects can exchange equal and opposite impulses
- Also, like forces, we can compute a total impulse as the sum of several individual impulses

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

Review [4]: Final Velocity & Collision Impulse J

- We take the difference between the two velocities and dot that with the normal to find the closing velocity



$$v_{close} = (\mathbf{v} - \mathbf{v}_{obj})\cdot\mathbf{n}$$

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

Review [5]: Impulse given Velocity (Frictionless)

- Let's first consider a collision with no friction
- The collision impulse will be perpendicular to the collision plane (i.e., along the normal) and will be large enough to stop the particle (at least)

$$\mathbf{j} = -(1+e)mv_{close}\mathbf{n}$$

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

Review [6]: Dynamic Friction Equation (Coulomb)

- As we are not considering static contact, we will just use a single dynamic friction equation
- For an impact, we can just compute the impulse in the exact same way as we would for dynamic friction
- We can use the magnitude of the elasticity impulse as the normal impulse

$$\mathbf{j}_{dynamic} = \mu_d \left|\mathbf{j}_{normal}\right| \mathbf{e}$$

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

Review [8]: Position Adjustment Options

- Moving the particle to a legal position isn't always easy
- There are different possibilities:
  - Move it to a position just before the collision
  - Put it at the collision point
  - Put it at the collision point plus some offset along the normal
  - Compute where it would have gone if it bounced
- Computing the bounced position is really the best, but may involve more computation and in order to do it right, it may require further collision testing…

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

Review [9]: Data Structures for Collisions

- BV, BVH (bounding volume hierarchies)
  - Octree
  - KD tree
  - BSP (binary separating planes)
  - OBB tree (oriented bounding boxes- a popular form of BVH)
  - K-dop tree
- Uniform grid
- Hashing
- Dimension reduction

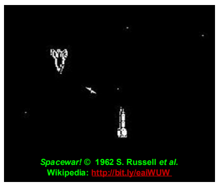
Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD

How Are Particle Systems Used?

- Explosions
  - Large
  - Fireworks
- Fire
- Vapor
  - Clouds
  - Dust
  - Fog
  - Smoke
  - Contrails
- Water
  - Waterfalls
  - Streams
- Plants

Command & Conquer 4: Tiberian Twilight
© 2010 Electronic Arts, Inc.
Wikipedia: http://bit.ly/gFGMjO

History of Particle Systems [1]: Spacewar!


Spacewar! © 1962 S. Russell *et al.*
Wikipedia: http://bit.ly/eaiWUW

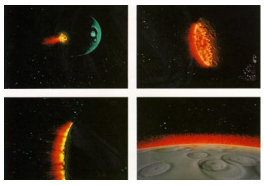
- Developed in 1962 on Digital Equipment Corporation PDP-1
  - Steve "Slug" Russell, Martin "Shag" Graetz, Wayne Witaenem
  - Trig functions by DEC
  - Other features, Dan Edwards & Peter Samson
- Used Pixel Clouds as Explosions

History of Particle Systems [2]: Asteroids

Asteroids © 1979 L. Rains & E. Logg
Wikipedia: http://bit.ly/hwfEQk

- Short Moving Vectors for Explosions
- Probably First "Physical" Particle System (Collision Model) in Games
- *Hey, Hey, 16K* © 2000 M. J. Hibbett, Video © 2004 R. Manuel http://youtu.be/Ts96J7HhO28

History of Particle Systems [3]: Genesis Device in *Star Trek II*


"Wall of Fire" effect from *Star Trek II: The Wrath of Khan* © 1983 Evans & Sutherland
Wikipedia: http://bit.ly/eXwrhb

- Particle System for Genesis Bomb: http://youtu.be/Qe9qSLYK5q4
- Part of Planetary Fly-By "Visualization"
- One of Earliest Cinematic Uses

Definition & Basic Particle System Physics

- A particle system is a collection of a number of individual elements or *particles.*
- *Particle systems control a set of particles that act autonomously but share some common attributes.*

- Particle is a point in 3D space.
- Forces (e.g. gravity or wind) accelerate a particle.
- Acceleration changes velocity.
- Velocity changes position

More Attributes of Particles

- Position
- Velocity
- Life Span
- Size
- Weight
- Representation
- Color
- Owner

Methods of Particle Systems

- Initialize
- Update
- Render
- Move
- Get/Set force

Implementation [1]: Particle Struct

```
struct Particle
{
  Vector3  m_pos;           // current position
  Vector3  m_prevPos;       // last position
  Vector3  m_velocity;      // direction and speed
  Vector3  m_acceleration;  // acceleration

  float   m_energy;         // how long particle is alive

  float   m_size;           // size of particle
  float   m_sizeDelta;      // change in size per time unit

  float   m_weight;         // how gravity affects particle
  float   m_weightDelta;    // change over time

  float   m_color[4];       // current color
  float   m_colorDelta[4];  // change over time
};
```

Implementation [2]: Particle System Class

```
class ParticleSystem
{
   public:
      ParticleSystem (int maxParticles, Vector3 origin);
      // abstract functions
      virtual void  Update(float elapsedTime)     = 0;
      virtual void  Render()                      = 0;
      virtual int   Emit(int numParticles);
      virtual void  InitializeSystem();
      virtual void  KillSystem();
   protected:
      virtual void  InitializeParticle(int index) = 0;
      Particle *m_particleList;     // particles for this emitter
      int       m_maxParticles;     // maximum total number of particles
      int       m_numParticles;     // indices of all free particles
      Vector3   m_origin;           // center of the particle system
      float     m_accumulatedTime;  // track when last particle emitted
      Vector3   m_force;            // forces (gravity, wind, etc.) on PS
};
```

How to Represent Particles?

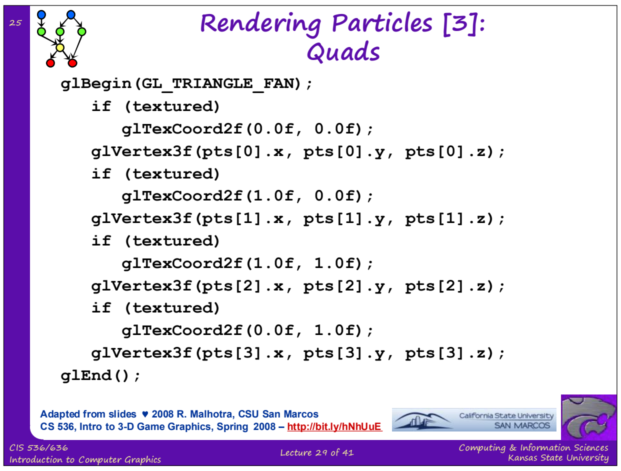
- Points
- Lines
- Texture-mapped quads
- Point Sprites

Rendering Particles [1]: Points

```
glBegin( GL_POINTS );
   glVertex3f
       (m_position.x,
        m_position.y,
        m_position.z);
glEnd();
```

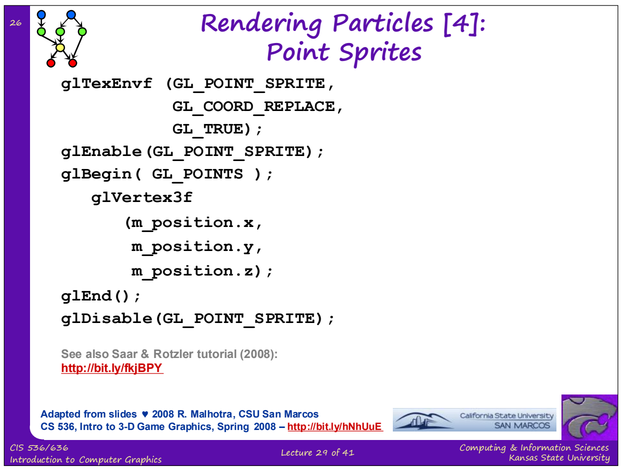
Rendering Particles [2]: Lines

```
glBegin(GL_LINES);
   glColor4f( r, g, b, 0.1f );
   glVertex3f
       (m_position.x,
        m_position.y,
        m_position.z);
   glColor4f( r, g, b, a );
   glVertex3f
       (m_position.x + m_direction.x,
        m_position.y + m_direction.y,
        m_position.z + m_direction.z);
glEnd();
```



Rendering Particles [3]: Quads

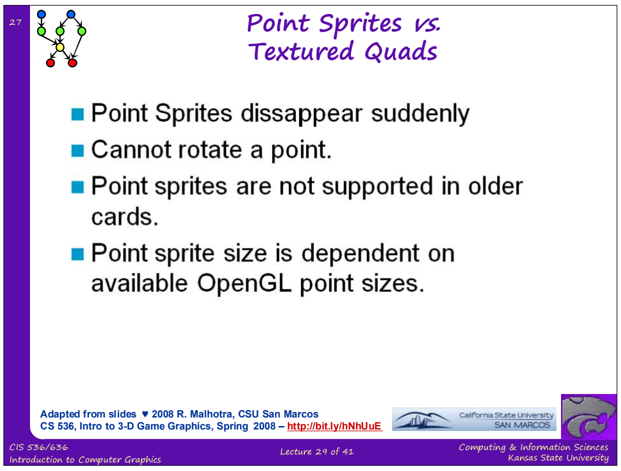
```
glBegin(GL_TRIANGLE_FAN);
   if (textured)
      glTexCoord2f(0.0f, 0.0f);
   glVertex3f(pts[0].x, pts[0].y, pts[0].z);
   if (textured)
      glTexCoord2f(1.0f, 0.0f);
   glVertex3f(pts[1].x, pts[1].y, pts[1].z);
   if (textured)
      glTexCoord2f(1.0f, 1.0f);
   glVertex3f(pts[2].x, pts[2].y, pts[2].z);
   if (textured)
      glTexCoord2f(0.0f, 1.0f);
   glVertex3f(pts[3].x, pts[3].y, pts[3].z);
glEnd();
```



Rendering Particles [4]: Point Sprites

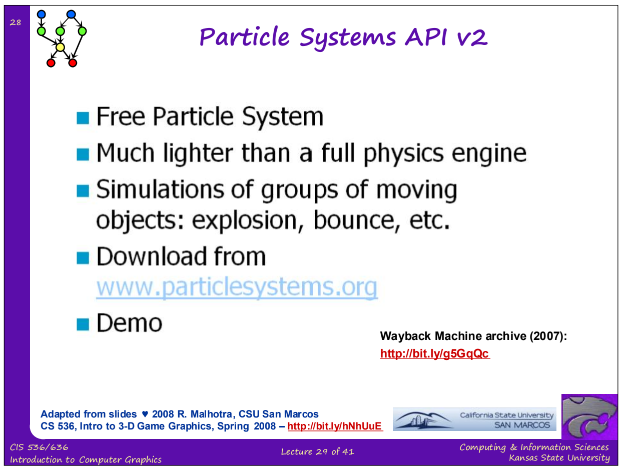
```
glTexEnvf (GL_POINT_SPRITE,
           GL_COORD_REPLACE,
           GL_TRUE);
glEnable(GL_POINT_SPRITE);
glBegin( GL_POINTS );
   glVertex3f
      (m_position.x,
       m_position.y,
       m_position.z);
glEnd();
glDisable(GL_POINT_SPRITE);
```

See also Saar & Rotzler tutorial (2008):
http://bit.ly/fkjBPY



Point Sprites vs. Textured Quads

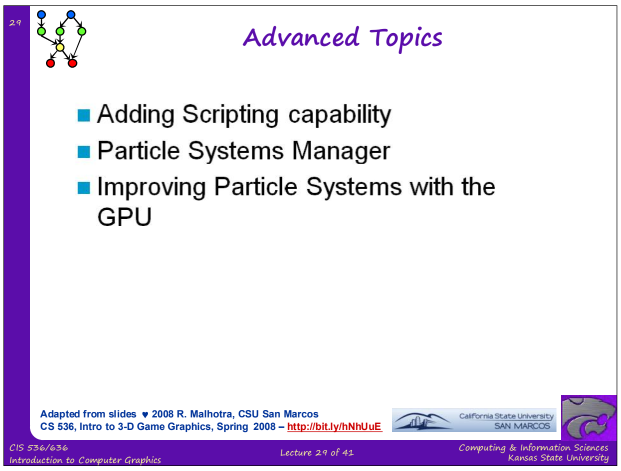
- Point Sprites dissappear suddenly
- Cannot rotate a point.
- Point sprites are not supported in older cards.
- Point sprite size is dependent on available OpenGL point sizes.



Particle Systems API v2

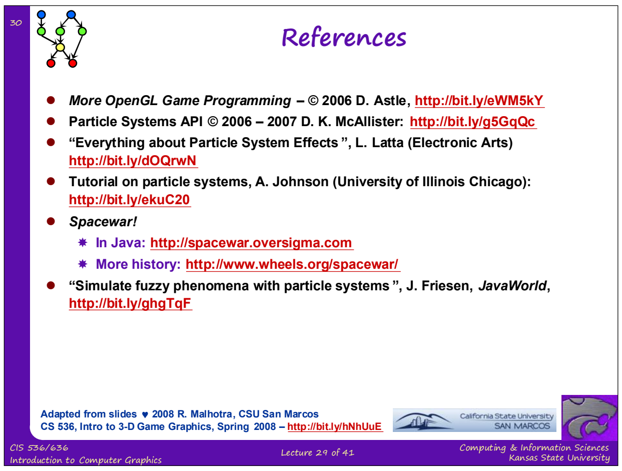
- Free Particle System
- Much lighter than a full physics engine
- Simulations of groups of moving objects: explosion, bounce, etc.
- Download from www.particlesystems.org
- Demo

Wayback Machine archive (2007):
http://bit.ly/g5GqQc



Advanced Topics

- Adding Scripting capability
- Particle Systems Manager
- Improving Particle Systems with the GPU



References

- *More OpenGL Game Programming* – © 2006 D. Astle, http://bit.ly/eWM5kY
- Particle Systems API © 2006 – 2007 D. K. McAllister: http://bit.ly/g5GqQc
- "Everything about Particle System Effects", L. Latta (Electronic Arts) http://bit.ly/dOQrwN
- Tutorial on particle systems, A. Johnson (University of Illinois Chicago): http://bit.ly/ekuC20
- *Spacewar!*
  - In Java: http://spacewar.oversigma.com
  - More history: http://www.wheels.org/spacewar/
- "Simulate fuzzy phenomena with particle systems", J. Friesen, *JavaWorld*, http://bit.ly/ghgTqF

Summary

- Reading for Last Class: §9.1, Eberly 2e; Particle System Handout
- Reading for Today: Particle System Handout
- Reading for Next Class: §5.3, Eberly 2e; CGA Handout
- Last Class: Particle Systems
  - Collision response
  - Simulation, events: birth (emission), collision, death
  - Properties: mass, initial velocity, lifetime
  - Changing properties: color, position (trajectory)
- Today: Lab on Particle Systems; Dissection of Working Program
- Next Class: Computer-Generated Animation Concluded
  - Autonomous movement in agents *vs.* hand-animated characters
  - Inverse kinematics
  - Rag doll physics
  - Minimization models
  - More CGA resources

Terminology

- Particle Systems – Simulation of Processes, Simple Physical Bodies
  - Events
    - Birth – particle generated based on shape, position of emitter
    - Collision – particle with object (including other particles)
    - Death – end of particle life, due to collision or expiration
  - Initial properties: mass, position, velocity, size, lifetime, color, owner
  - Change in properties: delta mass, position, *etc.*
- Emitter – Point, Line, Plane or Region from which Particles Originate
- Particle Fountain – Particle System with Directional Emitter
- Sprite (Wikipedia: http://bit.ly/gyInPg)
  - Definition: 2-D image or animation made part of larger scene
  - Point sprite
    - Screen-aligned element of variable size
    - Defined by single point
    - (Saar & Rotzler, 2008): http://bit.ly/fkjBPY