CS134 Computer Graphics - Lab 9

Points

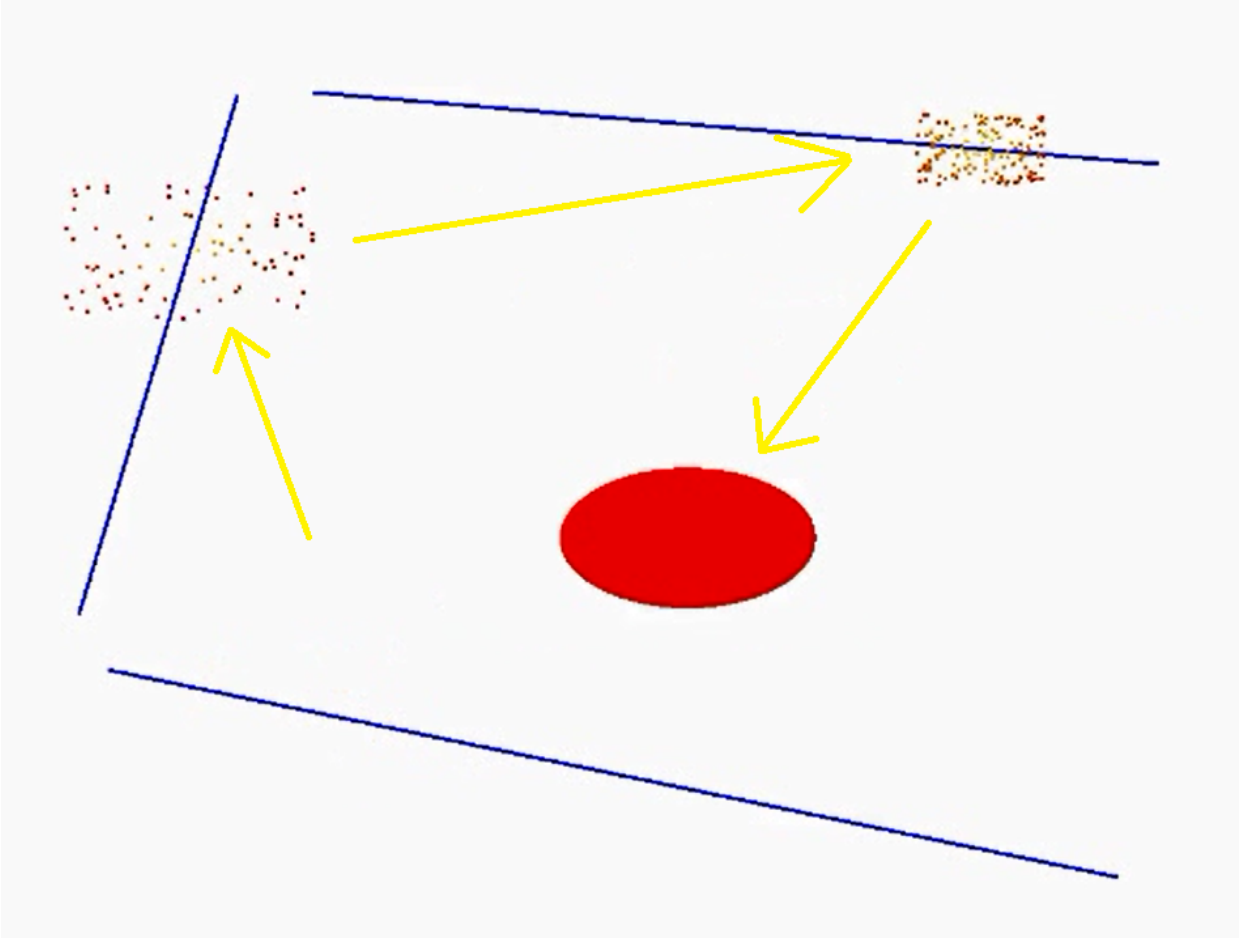
- Part 1 - 10 Points

Goals

- Complete the Particle System Framework
- Design your own Particle Effect

Introduction

Particle Systems are used to create simple special effects. An example is shown below of a ball colliding on walls causing sparks at the point of collision.



In this lab, we will develop the particle system framework provided the template below. The template includes the Vector3 and Color class from previous labs as well as the Particle and ParticleSystem classes which need to be completed.

Download the template file below:

https://drive.google.com/file/d/0B2dqioxbKNaAOFU0c1VPLU1qbVU/edit?usp=sharing

A particle has the following essential components:

- Position Vector, p
- Velocity Vector, v
- Sum for forces, f
- Mass, m

The aggregate sum of forces on a particle changes the velocity component at each time step. This allows freedom for the controller to add any additional forces with ease. The sum of forces with the mass can be reduced down to an acceleration vector since the acceleration component directly controls the velocity vector.

Therefore, the components can be simplified in this lab:

- Position Vector, p
- Velocity Vector, v
- Acceleration Vector, a

The particles will be animated along with the rest of the game engine provided with their own update() function. The update() function will update the vector components known as a Euler Step:

- $v(i+1) = v(i) + a(i) * \triangle t$
- $p(i+1) = p(i) + v(i) * \triangle t$

The Particle System is used to track all particles involved with it's associated effect. It spawns its own particles, updates them, and removes them (when expired).

Task:

In this lab, your job is to complete the particle system framework. The Particle class will also contain the representative components above as well as the following:

- The color of the particle
- The lifetime of the particle

A particle will be removed from the Particle System if it has expired. This will prevent unnecessary tracking. However, its initialization is left up to you such that you can design your own unique particles.

Complete the following functions:

- Particle: *isExpired()* - returns true if the particle has existed beyond its lifetime.

- Particle: *update(float elapsed)* - performs a Euler Step updating its timer, acceleration, velocity, and position.

- ParticleSystem: *init(const Vector3 & spawnPoint)* - populate the particle list provided a spawn location. This will depend on what kind of special effect you want to generate. An example would be an explosive spark: generate hundreds of particles with a random direction with a random color gradient from red to yellow.
  Note: you may use *randFloat(float min, float max)* to retrieve reasonable values within a certain range.

Once these three functions are completed, you can test your Particle System by calling its constructor and tracking it. The global list, *psystems,* is available to you and already updates all Particle System every frame.

After confirming the Particle System you generated works properly, write a mouse event handler such that when you click, it will generate your Particle System at the clicked location.

Next, have each individual particle change in representation besides position over time (color, speed, direction, or ect.) if you haven't done so already. This change in representation should be a pattern associated with all particles. The TA will not be looking for the "perfect animation" for this special effect, you can tune or modify your parameters until it looks desirable.

Reference:
http://www.pixar.com/companyinfo/research/pbm2001/pdf/notesc.pdf