CG 2013-14
Assignment 3: Particle Systems

Out: Dec 10. Due: Dec 23.

Introduction

In this third assignment, you will animate a particle system in order to achieve a number of effects. As for the previous assignments, interactive rendering will be done through OpenGL directly into the provided template code and, to ease your debugging, we are providing the compiled version of the solution code, which can be used to compare your results with ours.

Template code

The provided template c++ code initializes an interactive window which draws the virtual scene 60 times per second. Furthermore, the template code provides the following functionalities:

Class Particle: represents a single particle. The state of the particle is composed by its position, velocity, acceleration, etc. Furthermore a particle can compute its dynamics via Verlet Integration by calling the method *Update()*.

Class ParticleSystem: represents a set of particles. An object of the class *ParticleSystem* initializes, updates and draw a set of particle.

- **Initialization** - method *Build()* – create a set of particles and initialize their position and velocity;
- **Update** – method *Update*() – apply forces to the particles (e.g., gravity) calling the private method *AccumulateForces()*, call the update method for each particle and then handles the collisions in *CollisionHandling();*
- **Draw** – method *Draw()* – draw the particles as round points and draw the collision geometric primitives.

Requirements

1. Firework. [10 points]

Change the code of the *Build()* function, defined in *particle_system.h.* Put 100 particles in the position (0, 0, 0). Each particle must be initialized with a random velocity within the semicircle lying on the xy plane, y > 0, center in (0, 0, 0) and radius 10.

Note that you can compute a random number by using the c function *rand()*:

```
int i = rand(); // integer number between 0 and RAND_MAX

float f = ((float)rand / RAND_MAX) * 10; // float 0 - 9.99…
```

The final result shall be similar, not identical, to the one showed in the executable *1.firework.exe*.

2. Attractor. [10 points]

Modify the code of *Build()* in order to produce a regular grid of particles centered into the origin. Make a grid of 50x50 particles with initial velocity zero.

Modify the code of *AccumulateForces()*, remove the gravity and add a force of magnitude 10 from each particle to the origin. The origin in this case works as an attractor: the particles will be always moved towards it.

Add a *Vector3* member to *ParticleSystem* called *attractor*, and initialize it to (0, 0, 0). Change its position to a random value in the bounding box (-5, -5, -5) and (5, 5, 5) every 5 seconds. Note that the *Update* function is called 60 times per second so once every 16.67 ms. So, the *attractor* position should be moved once every 300 iterations.

The result shall be similar to *2.attractor.exe*

3. Collisions. [10 points]

Modify the *Build()* function in order to shot a set of 1000 particles from (0, 0, 0) towards the positive y axis. Do this by setting the initial velocity with a random modulus between 8 and 12, a random angle around the positive y between –PI/20 and +PI/20. Let the particles fall under the effect of the gravity force.

Fill the body of the *CollisionPlaneXZ()* and *CollisionSphere()* functions in order to compute the collisions between the particles, the planes and the sphere defined inside *CollisionHandling()* (they must be uncommented). Note that the *CollisionPlaneYZ()* is already provided.

Define the corresponding functions to draw the collision primitives in *DrawCollisionPlaneYZ(), DrawCollisionPlaneYZ(), DrawCollisionPlaneXZ(), DrawSphere()*. The result shall be similar to *3.collisions.exe.*

Extra credits

Choose some of these items for extra credits.

- Move the attractor in task 2 in order to roughly follow the position of the mouse (e.g., when the mouse move towards right, so does the attractor). The position of the mouse and the attractor do not need to be exactly the same, it is enough to let them move in the same way. If you want to use exact positions, see *gluUnproject(…)*.
- Add a trail to each particle.

Submission

Send the code for each point ***within the 23th of December*** to cg.sapienza.1314@gmail.com