Problem Session: Workshop 1 - Particle Cannon

Goals

- Understand the fundamentals of a complete OpenGL program in UNIX/LINUX
- Understand how first person movement works in OpenGL
- Understanding fundamental particle movement with gravity

Students may work in groups of 1, 2, or 3 (ideally 2 per desktop computer)
You should ask the teaching assistants for help!

Problem 1 (low difficulty):
(a) Add functionality to the first person movement so that the "k" key moves the person backwards and the "h" key (home) resets the user to his original position and orientation.
(b) Allow quick changes to the number of particles by adding: #define MAXPARTICLES 20 and adjust current code accordingly - Look for the loops which end at 10. For fun, try it out with 1000.
(c) Currently, we draw immediately to the screen. Double-buffering means that it first writes everything to an unseen buffer and then swaps the unseen to the display buffer. To do this, in main(), after glutInit(), add this line: glutInitDisplayMode (GLUT_DOUBLE). In update(), replace glFlush() with glutSwapBuffers(). Now the rendering should be nicer.
(d) Make the "r" key fire all of the particles from a single point above the ground (each time you hit "r", it should select a random value for y between 10 to 60 and a random color – all MAXPARTICLES will have the same starting point and the same color) in all directions, not just upwards. This should look a bit like exploding fireworks. Try it with MAXPARTICLES 1000.

Problem 2 (medium difficulty):
The "g" key should toggle gravity on/off - the first time you hit "g", gravity should turn on; the second time gravity should turn off, third time on, etc.
In principle, recall that gravity is the force, f=m*a, exerted between two objects: $f = G * (m_1*m_2)/(r*r)$
where m_1 is the mass of object 1, m_2 is the mass of object 2, r is the distance between them and G is the gravitational constant, http://en.wikipedia.org/wiki/Gravitational_constant
This means that we will *approximate* gravity using updates to the velocity in the Y axis using *roughly*

NewVelocity = OldVelocity + a * t (where a = acceleration from gravity and t = time)

Because the gravity is working against the direction of the initial particle movement,

NewVelocity = OldVelocity - a * t (where a = acceleration from gravity and t = time)

Thus, within the context of this problem, gravity is simply a change in the velocity in the Y direction. Visually this will mean that the particles will initially move upwards, slow down, and then eventually fall to the ground. *Note that when objects hit the ground they should stop moving.*

Problem 3 (high difficulty):
The "w" key should toggle a gravity well on/off and should automatically turn off the gravity from Problem 2. In Problem 2, the gravity was 1D, just on the Y axis. Now we will look at 3D gravity like the gravity well of a star in space around which the particles should move like planets. The only difference is that you must compute the direction, V, from the well to the particle and the gravity should be exerted in that direction on the X, Y, and Z velocities. *When objects hit the ground with "w" enabled, they should continue moving as if the ground did not exist.*

Within the context of this problem, the gravity well should appear as a white particle above the ground "roughly" at x=0, y=50, z=0). This white particle does not move and should exert 3D gravity (now gravity works in X, Y, and Z). This means that when you fire the particle cannon, the cannon particles should clearly move toward the white particle using 3D gravity. When the "w" key is toggled on, objects should not stop when hitting the ground. It should be possible for objects to go into stable near-circular orbits.

Submission Checklist

The deadline is at the end of class, but if necessary, they may be submitted by 6pm of the day of the Workshop. Place in a ZIP file and submit on the LML Course Manager: *The top level of the zip file should contain a directory called* ***firstname.lastname.project*** *as described below*: In a directory called "***firstname.lastname.project***" (e.g. ***mary.smith.project***)

(1) a file named "**AnswerJournal.txt**" which should list
- Your full name and student ID
- Mention which of the problems you solved.

(2) The source code in C/C++ on Linux, Makefile (the project must compile using "make") and
(3) Working executable of your solution
(4) All source code must compile and execute on the Linux OS in the Snellius computer rooms (e.g. rooms 302, 303, 306)

Good luck!