Object-Oriented Design — Lab 9

Jeff Epstein

November 7, 2011

Introduction For this exercise, you must work in pairs. Choose a partner from the class. You will submit your programs together. If you can't find a partner, ask your lab instructor.

If you need help with the exercises, ask your lab instructor.

Do not copy code from other groups. Do not copy code from the internet. You must be able to answer any questions about your code.

Rules For these exercises, you must use good programming style. Specifically:

- Use comments for all functions and classes.
- Each class has a constructor and, if appropriate, a destructor.
- Use good and consistent indentation.
- Create a header file for each class.
- Use good, accurate variable names.
- Use C++ data types (like `string`) instead of C data types (like `char*`).

Introduction The *n-body problem* (`http://en.wikipedia.org/wiki/N-body_problem`) is a famous problem. We will write a program to simulate the gravitation between a number of bodies.

The *gravitational force* between two bodies is $F = G\frac{M_1 \times M_2}{d^2}$ where

- G is the gravitational constant
- M_1 is the mass of the first body
- M_2 is the mass of the second body
- d is the distance between the two bodies

We will focus on a two-dimension simulation. Therefore we are interested in the horizontal force (F_x) and vertical force (F_y) on each body.

The *net force* on a body is the sum of all forces on it.

The *acceleration* of a body is given by $a = F/m$ where

- F is the force on that body
- m is the mass of that body.

Similarly, the horizontal and vertical forces are given by $a_x = F_x/m$ and $a_y = F_y/m$.

Program You will create a class named `Body`. This class represents a single body, and will store its current position, acceleration, and mass.

You will also create a class named `Universe`. This class controls the simulation. It will store a `vector` of all bodies.

We read data about the bodies from a file. The file should be named `nbody.txt`. Use the `ifstream` class to read the file. Here is an example file:

```
.00005
S 500 500 0 0 100000000
P 255 255 2 0 1
A 745 745 -2 0 2
M 270 740 -1.5 -3 12000
Q  253 450 0 -2 50000
R 500 0 3 0 1000000
V 504 0 4 4 0.000001
```

The first line contains a value for G, the gravitational constant.

The other lines contain information about a body. Each line says:

- The body's name (a `string`).
- The body's initial horizontal position (a `double`).
- The body's initial vertical position (a `double`).
- The body's initial horizontal velocity (a `double`).
- The body's initial vertical velocity (a `double`).
- The body's mass (a `double`).

The x and y coordinates of position are between 0 and 1000. So, in the above file, the body "S" (the sun) is in the middle of the universe and is not moving initially.

For a demonstration of this universe, watch the movie `http://www.ductape.net/~eppie/lab9.ogv` You may need to install VLC.

You can download example files from `http://www.ductape.net/~eppie/lab9-examples.zip`. You can also create your own universes.

Your program will have a simple structure. Basically:

```
read data file "nbody.txt"
repeat forever:
   display bodies
   update position of bodies
```

To `display bodies`, show each body as a letter on the screen.

To `update position of bodies`, calculate the force on each body. Then you can calculate the bodies' acceleration. When you know the acceleration of each body, you can calculate its new velocity. Finally, when you know the velocity, you can calculate the new position. Each iteration of the loop, we will update each body's position and velocity.

Remember that the velocity is the integral of the acceleration. The position is the integral of the velocity.

Extra functions Download `http://www.ductape.net/~eppie/lab9.zip` and add the files to your project. There are two useful functions:

- `void sleep(long ms)` will wait `ms` milliseconds. You can use this to slow down the animation.
- `void putText(string text, unsigned x, unsigned y)` will display the string `text`, starting at column `x` and row `y`.

File access You can use the `ifstream` (for *input file stream*) class to read data from a file. It is similar to `cin`.

```
/*
  The file "integers.txt" should contain
  integers separated by whitespace, for example:
     34 435 65 234  345 4
*/
#include <fstream>
using namespace std;
int main()
{
  ifstream thefile("integers.txt"); // create new file object for reading
  while (thefile) // repeat until we reach end of file
  {
     int someint;
     thefile >> someint; // read an integer from the file
     cout << someint << endl; // display the int
  }
  return 0;
}
```

References For more information about using the `ifstream`, `string`, or `vector` classes, consult `http://www.cplusplus.com/reference/`

Submission Each file must contain your full name and email address in a comment.

Put all your source code in a ZIP file and send it to `epcz@centrum.cz`