Data structure and algorithm analysis project 2021

N-Body dynamics simulation under gravitational force and collision

1. Introduction

In this project, you are going to build a simulation system of rigid spherical objects. You should write a program to simulate some rigid balls in a vacuum space so that they can move and collide with each other.

The basic aspects of the problem are as follows:

- All of the objects are rigid body. When they collide with each other, they will not change shape. They will not break into pieces either.
- Each object has its own size, mass, position and color. You can add as many attributes to the objects as you wish. The size, mass and color of an object do **NOT** change. But the position can change with time.
- Since the objects move their position, they can collide with each other and bounce off with each other. When this happens, the energy and momentum are preserved.
- There are gravitational forces between objects. This affects the movement of the objects.
- You need to write programs to simulate this process.

The difficulty of this problem can vary a lot with different requirement of the program. In order to simplify this job, we add the following simplification:

- All of the objects that move in space are spherical balls with uniform density, which means the centroid of mass of a ball is the the geometry center of the same ball.
- Ignore relativistic effect, quantum effect, air resistence (as the space is vacuum), friction.
- The gravitational force between two objects is denoted as: $\frac{Gm_1m_2}{R^2}$, where G is the gravitational constant, approximatly $6.67 \times 10^{-11} Nm^2/kg^2$. You can use that number in this project. m_1 and m_2 are the mass of the two object, respectively. R is the distance between the centroid of the two object.
- The objects are smooth. And they do **NOT** rotate at the initial state of the simulation. So they will not rotate at any time.
- The objects are in a two dimensional space. There are four walls around the space. The walls are also rigid. When the objects collide with the wall, they can bounce off the wall. The mass of the walls are very small, so the gravitational force between the walls and the objects are ignored. The walls do not move, so that you can simulate a finite size of a square space where x,y $\in$ [0, length] instead of from negative infinity to positive infinity.

- The objects cannot intersect with each other. The objects cannot intersect with the wall.
- All collisions happens instantly.

Your program should display the result using a graphic window. The window should display how the balls move as time goes.

You should also write some test code to test the correctness of your program as well as the performance. Note that in real world project, this could be equally important as writing a good algorithm.

2. Project Requirement

The actural work of the project contains many parts:

- Write the system described above. (50 Points)
- Test the accuracy and performance of your system. (10 points)
- Write your report. (10 points)

Write the system described above. First of all, you should make such a system.

- (10 points) Can customize all collision balls.(This means that the program should be able to read from standard input the initial state of all the balls.)
- (20 points) The program should be able to run and show the movement of the objects with GUI.
- (10 points) To collide and bounce according to the energy and momentum.
- (10 points) Consider the influence of gravity.

Test the accuracy and performance of your system.

- (5 points) Show that your program can produce the correct result.
- (5 points) Show the performance of your program.

Programming mistakes can happen anywhere in such a project. Some mistakes will make your program produce the wrong result but still be able to run. Try to prove that your program can produce the right result.

You could also find ways to show that your program runs fast, especially when you have some performance improvement in your project. Try to run your program with large number of objects instead of just 10 objects. Maybe you can test the frame rate of your program. You may also find hints in the reference material at the end of this document.

Write your report.

Include everything you want us to know about your project. Such as the basic structure of your project, the data structures you have used in the project, the way you optimize the performance, and anything that makes your project different from others.

Clearly state how to run your program in your report. It's necessary when we test your program.

You may also include some statistics of your program, such as those mentioned in the reference links.

However, we have no requirement on the length of the report. Do **NOT** try to make it unnecessarily long. Remember to include group members and student numbers. Also remember to upload your report in "pdf" format.

Extra test

You may noticed that the above parts add up to 70 points. In the last we will run some test we have prepared on your program. That is the last 30 points of your project. This can only be down after your code is submitted. We'll test your program by some cases. By specifying the number of objects and each object's size, mass, position and color, your program's result should be the same with the standard answer. The difficulty of each test cases is different. Some examples contains only a few objects, e.g. only 3 or 4 balls in the space. However, some cases contains much more objects than that. Brute force algorithms may not pass all of them, and you can try to minimize the time complexity of your algorithm. Each test case has some score, and your score depends on the number of test cases you pass. Your algorithm has to be able to show us the process through the GUI.

There are restrictions on what you can use and cannot use in this project.

You could use various tools to advance your project, but third party physics engines, game engines are not allowed. For instance, you could use OpenGL, and OpenCL, but Unity and Unreal are not allowed to use in the project.

You are allowed to use algs4 library. If you want to use some library and you are not sure about it, you can ask.

3. Input & Output Standard

Here's a template input file in the block shown as below.

```
terminal
100
3
0.5 0.5 0   0       0.1     1000000 150 30  10
0.1 0.5 0   0.01    0.03    1       50  50  50
0.8 0.5 0   0.01    0.01    0.1     50  50  50
3
4   2
10  0
14  4
```

There are 2 choices of line 1st: `terminal` and `gui`. `terminal` means you should not show a window. Instead, you should print the required information in the terminal. `gui` means you should show a window.

The number of line 2nd indicates the side length of the square space, all the objects are supposed be in the square space. For example, the second line contains `100`, then the x and y coordinate of everything in your space is in [0, 100].

The number of line 3rd `num` is the number of objects.

For each of the next `num` lines, there are 9 numbers represent the initial state of one object. The 9 numbers are:

- x-coordinate `x`
- y-coordinate `y`
- initial velocity in the x direction `vx`
- initial velocity in the y direction `vy`
- radius of the square `radius`
- mass of the square `weight`
- color of the square `r` `g` `b`.

The unit of `x`, `y` and `radius` is meter. The unit of `vx` and `vy` is meter/second. The unit of `weight` is kilogram.

The objects are indexed. The first object is the 0th and the second object is the 1st,..., the last object is `num-1`.

The number in the next line `query` indicates the number of querys. For each of the next `query` lines, there are 2 numbers. The first is the time `t` in seconds and the second is index of the object. You should print a line containing the `x`, `y`, `vx`, `vy` of the object at the time `t` seconds after the begining of the simulation.

Sample Input:

```
terminal
100
1
3 5 1 0 1 100 150 150 150
3
4 0
5 0
6 0
```

Sample Output:

```
7 5 1 0
8 5 1 0
9 5 1 0
```

In the above example, the object's initial location is (3, 5), the velocity is (1, 0). So after 4 seconds, the location is (7, 5). The speed remains the same in the above example.

Please follows strictly the above requirement. When we test your program (as mentioned in the last section, 30 points), we will run your program with our input data and compare your result with the standard output data. If your program does not follow the above restriction and prints something else, the comparison will fail and it will not be good.

It you have any question about the input output format, you can ask.

Reference

N-Body Simulation:
https://introcs.cs.princeton.edu/java/assignments/nbody.html

N-Body Simulation, with Barnes-Hut tree:
https://introcs.cs.princeton.edu/java/assignments/barnes-hut.html

Molecular Dynamics Simulation of Hard Sphere:
https://algs4.cs.princeton.edu/61event