CS 439 Task 3: What-If Thinking on Celestial Bodies

Description

This task addresses what-if thinking about a scenario involving celestial bodies interacting through basic classical Newtonian mechanics, also known as gravity. Our world has only two dimensions. This theme is based on the Task 0 preferences and the discussion in Lecture 20.

This time we are looking primarily at the Q and A in our QMSVA framework. The M, S, and V are provided. You will have to tweak S and implement A to answer a variety of and variations on a number of Qs.

Details on V were provided in Lecture 22.

Specifications

Part I: Analysis

Add your implementation to `Analyzer` so that it calculates the following metrics from the behavior of all bodies in the simulation. You may do this however you want, but it must have at minimum the provided structure.

In particular, `printResults` calculates and prints the metrics of minimum, maximum, average, and standard deviation for both velocity and acceleration resulting from the first update through the most recent. It also reports total time and distance. In other words, your output should look something like this for each body:

```
Body: myBodyID

Time: 0.0

Distance: 0.0

Velocity

  min: 0.0
  max: 0.0
  avg: 0.0
  std: 0.0

Acceleration

  min: 0.0
  max: 0.0
  avg: 0.0
  std: 0.0
```

You need only one analyzer because it collects data for all bodies. The simulator sets it up, does the updates, and calls `printResults` itself.

Be very careful not to run the simulation longer than necessary because it generates a huge amount of data. It is possible to fill your hard drive!

Part II: Simulation for Everyone

The provided `Task3` class sets up the following tests. Modify the parameters as needed.

<u>Test 1.A: Colliding Bodies A</u>

Bodies A and B are initially stationary and are pulled toward to each other. The solution is already provided. This is a test of your analyzer only.

Test 1.B: Colliding Bodies B

Repeat 1.A, but they are initially moving toward each other. Choose a reasonable speed.

Test 1.C: Colliding Bodies C

Repeat 1.A, but they are initially moving away from each other. Choose a speed such that they do not leave the view.

Test 2.A: Passing Bodies A

Repeat 1.B, but they are on opposite parallel paths. How far apart (y) do they need to be to avoid an initial collision?

Test 2.B: Passing Bodies B

Repeat 1.B, but they are on opposite parallel paths. How fast do they need to be to avoid an initial collision? Use the same speed for both.

Test 3.A: Launch to Orbit A

The rocket is launched on a tangent to Earth. How fast does the rocket need to go to reach a reasonable orbit without using thrusters? What is the issue in the results?

Test 3.B: Launch to Orbit B

Repeat 3.A, but thrusters are allowed.

Test 3.C: Launch to Orbit C

Repeat 3.A but straight away from Earth, what happens?

Test 4: Leave Orbit

The rocket is already in orbit around Earth. How fast does the rocket need to go to break orbit? Set the initial velocity; do not use thrusters.

Test 5.A: Enter Orbit

Set the rocket up to enter a reasonable orbit around Earth.

Test 5.B: Miss Orbit

Repeat 5.A, but with an initial speed too high to be reasonably captured. (The rocket may assume a comet-like orbit eventually, but do not simulate that far.)

Test 6.A: Bad Day in Dinosaur City A

Assume Earth is full of dinosaurs and a meteor is coming. Set up and execute a glancing (tangential) blow.

Test 6.B: Bad Day in Dinosaur City B

Repeat 6.A, but use a direct hit (i.e., perpendicular to Earth).

Part III: Simulation for Graduate Students Only

The provided `Task3` class sets up the following test. Modify the parameters as needed.

Test 7: Apollo Program

Recreate a one-way moon mission:

1. Launch from Earth and enter orbit.
2. Make at least one complete orbit.
3. Leave orbit and fly to the moon.
4. Enter orbit around the moon.
5. Make two complete orbits.
6. Crash land on the moon.

Notes

Use `scheduleThruster` to increase or decrease the velocity of a body by 10%. The first argument is true or false, respectively. The remaining arguments are the time(s) when the thruster is to be fired. Duplicate times are allowed for more extreme thrust. Time is shown in the viewer.

Deliverables

Submit the following:

- Your `Analyzer.java` and `Task3.java`
- A nicely formatted PDF document describing in English the procedure and showing the results for each test. Include screenshots from the viewer. Do not submit log files.