Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №2.2
«Реализация модели вселенной. Трехмерная визуализация. Модель Солнце-Земля»
по курсу: «Языки и методы программирования»

Выполнил:
Студент группы ИУ9-21Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Реализовать модель вселенной. Каждый элемент вселенной должен быть объектом некоего публичного класса, который инициализируется вспомогательным публичным классом порождающим эту вселенную. При инициализации экземпляров класса частиц моделируемой вселенной необходимо подсчитывать количество частиц вселенной используя статичное экземплярное поле защищенное от изменения из объектов внешних классов путем реализации статичного метода. Сформировать исходные данные и определить необходимые экземплярные поля для хранения состояния объектов частиц вселенной в соответствии с условием задачи и реализовать расчет.

Моделировать движение материальных точек под действием силы гравитационного взаимодействия. Полученные данные визуализировать в виде трехмерного графика с помощью `gnuplot`.

Задачи

Расчет траекторий материальных точек вселенных и их визуализация с использованием `gnuplot`. Построение модели Солнце–Земля.

Решение

`Test.java`

Объявим класс для тестирования:

```
public class test {
    public static void main(string[] args) {
```

Создадим новую вселенную:

```
Universe universe = new Universe();
```

Добавим в эту вселенную материальную точку — модель Солнца:

Начальные координаты, м	Начальная скорость, м/с	Масса, кг
$(0, 0, 0)$	$(0, 0, 0)$	$1,9885 * 10^{30}$

```
universe.add(0, 0, 0, 0, 0, 0, 1.989e30); // солнце
```

И модель Земли:

Начальные координаты, м	Начальная скорость, м/с	Масса, кг
$(147098290 * 10^3, 0, 0)$	$(0, 30270, 0)$	$5,9726 * 10^{24}$

```
universe.add(147098290e3, 0, 0, 0, 30270, 0, 5.972e24); //
↪  земля
```

Далее, построим траектории движения планет за 1 год и запишем в файл:

```
        universe.dump("plot.dat", false);
        for (int i = 0; i < 1*24*365; ++i) { // год
            universe.recalcCoords(60*60); // час
            universe.dump("plot.dat", true);
        }
    }
};
```

Метод `recalcCoords` (Δt) обновляет координаты материальных точек вселенной спустя Δt.

Universe.java

Импорт необходимых пакетов и объявление класса Вселенной:

```
import java.util.Random;
import java.io.*;
import java.lang.Math;
import java.util.ArrayList;

public class Universe {
```

Объявим вспомогательный класс `Point`, реализующий геометрическую точку.

```
public class Point {
    private double x;
    private double y;
    private double z;
```

```
    public Point(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public double calcDist(Point p) {
        return Math.sqrt(Math.pow(this.x - p.x, 2)
                         + Math.pow(this.y - p.y, 2)
                         + Math.pow(this.z - p.z, 2));
    }

    public String toString() {
        return x + " " + y + " " + z;
    }

    public double getR() {
        return Math.sqrt(Math.pow(this.x, 2)
                         + Math.pow(this.y, 2)
                         + Math.pow(this.z, 2));
    }

    public void add(Point p) {
        this.x += p.x;
        this.y += p.y;
        this.z += p.z;
    }

    public void mult(double m) {
        this.x *= m;
        this.y *= m;
        this.z *= m;
    }
}
```

Далее, объявим класс `Particle`, реализующий частицу:

```
public class Particle {
    private double m;
    private Point pos;
    private Point v;
```

Поля класса: масса, координаты, вектор скорости.

Будем хранить точки всех вселенных в списке `particles`. Это необходимо для расчета силы гравитационного взаимодействия в случае нескольких Вселенных.

```
private static ArrayList<Particle> particles = new
 ↪  ArrayList<Particle>();
```

Объявим конструктор:

```
public Particle(double x, double y, double z, double vx,
 ↪  double vy, double vz, double m) {
    this.pos = new Point(x, y, z);
    this.v = new Point(vx, vy, vz);
    this.m = m;
    particles.add(this);
}
```

Функция преобразования к строковому типу:

```
@Override
public String toString() {
    return m + " " + pos + " " + v;
}
```

Метод для пересчета координат всех точек спустя Δt:

```
public static void recalcAllCoords(double time) {
```

Считаем вектор равнодействующей силам гравитационного взаимодействия для каждой точки:

```
Point[] forces = new Point[particles.size()];
for (int i = 0; i < particles.size(); ++i) {
    forces[i] = particles.get(i).calcOverallF();
}
```

С учетом времени и равнодействующей силы обновляем координаты каждой точки:

```
    for (int i = 0; i < particles.size(); ++i) {
        particles.get(i).recalcCoords(time, forces[i]);
    }
}
```

Метод для нахождения силы гравитационного взаимодействия, оказываемой одной точкой на другую:

```
private Point calcF(Particle p) {
```

Гравитационная постоянная: $G = 6,67430 * 10^{-11} \text{м}^3 * \text{кг}^{-1} * \text{с}^{-2}$.

```
final double G = 6.67430e-11;
```

Считаем расстояние между точками:

```
double r = this.pos.calcDist(p.pos);
```

И, наконец, модуль силы гравитационного взаимодействия:

$$|F| = G * \frac{m_1 * m_2}{r^2}$$

```
double modF = G * this.m * p.m / (r*r);
```

Теперь можем найти вектор силы. Для этого вектор из одной точки в другую, домножим на отношение модуля силы и длины этого вектора:

$$F = \frac{|F|}{|\vec{AB}|}\vec{AB}$$

```
    Point F = new Point(p.pos.x - this.pos.x,
                        p.pos.y - this.pos.y,
                        p.pos.z - this.pos.z);
    F.mult(modF / F.getR());
    return F;
}
```

Метод для вычисления вектора равнодействующей силы:

```
private Point calcOverallF() {
```

Тут просто суммируем силы, с которыми точка притягивается к остальным:

$$F_{\text{равнодействущая}} = \sum_{i=1}^{|particles|} F_i$$

Где F_i — сила гравитацонного взаимодействия данной точки и точки i.

```
    Point F = new Point(0, 0, 0);
    for (Particle p : particles) {
        if (p != this) {
            F.add(this.calcF(p));
        }
    }
    return F;
}
```

Метод для пересчета координат точки спустя Δt, под действием силы F:

```
public void recalcCoords(double time, Point F) {
```

$$\begin{cases} x = x_0 + v_{x0}\Delta t + \frac{a_x \Delta t^2}{2} \\ y = y_0 + v_{y0}\Delta t + \frac{a_y \Delta t^2}{2} \\ z = z_0 + v_{z0}\Delta t + \frac{a_z \Delta t^2}{2} \end{cases}$$

```
        Point a = F;
        a.mult(1 / this.m);
        this.pos.x += v.x * time + a.x * time*time / 2;
        this.pos.y += v.y * time + a.y * time*time / 2;
        this.pos.z += v.z * time + a.z * time*time / 2;
        this.v.x += a.x * time;
        this.v.y += a.y * time;
        this.v.z += a.z * time;
    }
};
```

Далее объявляем члены класса `Universe`.

Массив частиц и конструктор:

```
private ArrayList<Particle> particles;

public Universe() {
    particles = new ArrayList<Particle>();
}
```

Метод для добавления частицы во Вселенную:

```
public void add(double x, double y, double z, double vx,
 ↪  double vy, double vz, double m) {
```

```
    particles.add(new Particle(x, y, z, vx, vy, vz, m));
}
```

Метод для получения количества частиц, принадлежащих Вселенной:

```
public int getCount() {
    return particles.size();
}
```

Статический метод для обновления координат всех точек всех Вселенных:

```
public static void recalcCoords(double time) {
    Particle.recalcAllCoords(time);
}
```

Метод для вывода текущих координат точек Вселенной в файл. Флаг append выбирает режим записи: дописывать в конец или перезаписать.

```
    public void dump(String path, boolean append) {
        try {
            FileWriter file = new FileWriter(path, append);
            for (Particle particle : this.particles) {
                file.write(particle.pos + " ");
            }
            file.write("\n");
            file.flush();
        }
        catch(IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
};
```

universe.plt

Скрипт на gnuplot для 3D визуализации траекторий частиц.

```
#!/usr/bin/env -S gnuplot

set key off
file='plot.dat'
```

```
stats file nooutput
splot 'plot.dat' using 1:2:3 with dots dt 2, for
 ↪  [i=4:STATS_columns:3] 'plot.dat' using i:i+1:i+2 with
 ↪  lines ls 1
pause -1
```

plot.dat - пример исходных данных для графика

```
0.0 0.0 0.0 1.4709829E11 0.0 0.0
0.11936727519528922 0.0 0.0 1.470982502442213E11 1.08972E8
 ↪  0.0
0.47746906704009967 8.842856399831395E-5 0.0
 ↪  1.4709813097689645E11 2.1794397054849064E8 0.0
1.0743052405701388 4.421427895138887E-4 0.0
 ↪  1.470979321980704E11 3.269158527424634E8 0.0
1.909875525856707 0.001237999658732987 0.0
 ↪  1.4709765390783307E11 4.3588758767894816E8 0.0
2.984179518006644 0.002652855929754301 0.0
 ↪  1.4709729610631927E11 5.448591164550495E8 0.0
4.29721667716237 0.004863568027833623 0.0
 ↪  1.4709685879370883E11 6.538303801679827E8 0.0
5.848986328502027 0.00804699193662818 0.0
 ↪  1.470963419702265E11 7.628013199151102E8 0.0
7.639487662239712 0.012379983089440712 0.0
 ↪  1.4709574563614197E11 8.717718767939777E8 0.0
9.668719733625807 0.018039396260463417 0.0
 ↪  1.470950697917699E11 9.807419919023507E8 0.0
11.936681462947405 0.025202085456021827 0.0
 ↪  1.4709431443746988E11 1.08971160633825E9 0.0
14.44337163552884 0.03404490380581874 0.0
 ↪  1.4709347957364648E11 1.1986806611999884E9 0.0
17.188788901732295 0.044744703454178296 0.0
 ↪  1.470925652007492E11 1.3076490975862074E9 0.0
20.172931776958528 0.05747833545129029 0.0
 ↪  1.4709157131927252E11 1.4166168565959122E9 0.0
```

График модели Солнце-Земля