Edinburgh Napier UNIVERSITY

Coursework Report

Marco Moroni
40213873@napier.ac.uk
Edinburgh Napier University - Physics-Based Animation (SET09119)

Abstract

This coursework presents three types of gravity that are implemented in a physics-based simulation.

Keywords – Physics-Based Animation, gravity, Super Mario Galaxy

1 Introduction

The aim of this coursework is to explore different ways to implement gravity in a physics-based simulation:

- point gravity;
- Newton's gravity (gravity between any body with mass);
- *Super Mario Galaxy*'s gravity, a video game where the player can walk on planetoids of any shape.

2 Implementation

The starting project comes with a force `Gravity` towards the ground. This has been replaced with new forces for each gravity type.

2.1 Point gravity

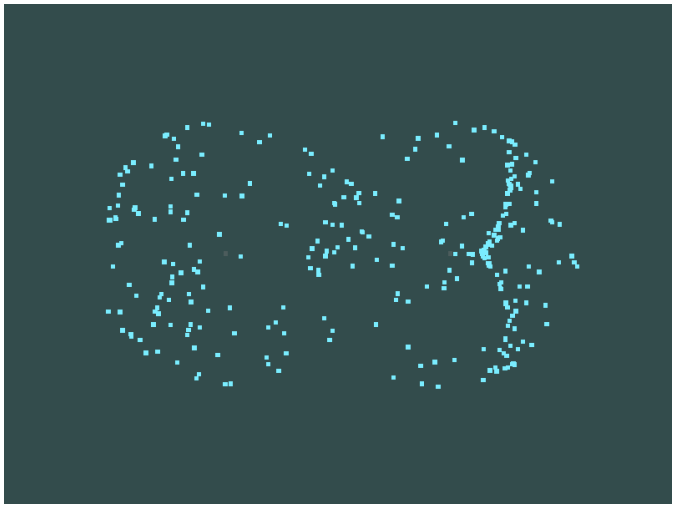

Figure 1: **Point gravity** - The particles are affected by two gravity points (grey)

Point gravity simplifies how bodies experience gravity on the surface of a planet: a body is pushed towards the "centre" of the planet, and it will be pushed with different intesities depending on the distance to it (the more the proximity the centre, the stronger the gravity is).
A force `pointG` that wants to simulate that needs to have:

- a position `c`, which represents the point where the gravitational accelleration is at its maximum;
- a value `g` for the gravitational accelleration;
- a radius `r`.

With these elements a sphere can be created. A body `body` can be either:

- inside such sphere. In this case the force applied to the body is made by the following:
  - direction: the normalised vector of `pointG.c - body.c`
  - scalar value: the distance `d` between `pointG.c` and `body.c` multiplied by a number between `0` (if the distance is more or equal to the radius) and `1` (if the body is at the centre of the sphere). This number is calculated by doing `1 - (d / r)`
- outside such sphere, in which case the force is `vec3(0.0f)`.

In this simulation (and the following) the body is a particle, therfore the body is considered inside the sphere when the distance between it and the centre is less or equal than the radius. Every particle has a force for each gravitational point applied.

2.2 Newton's gravity

This type of gravity is between any two bodies and it depends on their masses, the distance between them and a gravitational constant: $F = G\frac{m_1 m_2}{r^2}$. A force `newtonG` needs to know:

- the first body `b1` (or the body to which the force is added);
- the second body `b2`;
- a gravitational constant `g`, which will probably be different from the real one.

Newton's law of gravity is the same for both bodies (one is reversed). This useful information is unfortunately not used, because `newtonG` returns only the force to its corresponding particle.

Although the formula is easy to implement, it would be diffucult to see the intractions between the particles in the simulation because particles near each other can be pushed far away very quickly. The following adjustments have been made:

- the particles are set inside a box;
- the velocity cannot be higher than a certain amount;
- when two particles are very close to each other, `newtonG` is `vec3(0.0f)`. By doing so the particles will seem to form a conglomerate because they don't have enough force to "leave" a particle with a higher mass, which makes easier for the user to see the forces effects.

2.3 *Super Mario Galaxy*'s gravity

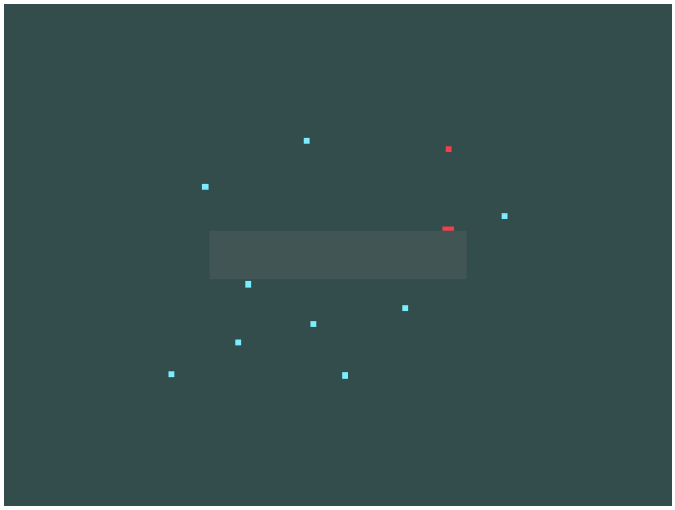

Figure 2: ***Super Mario Galaxy*'s gravity** - The gravity pushes particles towards the grey box.

In the video game *Super Mario Galaxy* the player can walk of planetoids of any shape, and this means all previous gravity types cannot be used. Instead, the gravity direction should be the inverse of the normal of the surface below the player. Usually this surface is the one closest to the player. The gravity is the same no matter the distance from the ground and a body is affected by only one gravity force at a time.
The force is calculated in the following way:

1. The closest point `q` to the the body on (or in) the planetoid is calculated [1] (see the red particle with the corresponding `q` point in fig. 2);
2. The force direction is the normalised vector `q - body.c`. The non-normalised vector is the closest distance between the body and the planetoid, and normalised vector corresponds to the normal of the planetoid's surface;
3. The force (direction multiplied by a constant) is returned.

By using the closest point method in step 1, the force will always have the correct direction, even if `q` is on an edge or a vertex of the planetoid.

3 Future work

The simulations could be more interesting if instead of particles, spheres (with collisions between them) were used.

4 Conclusion

Gravity can be implemented in many different ways, and depending on the goal, some of the foramlae used can be completely different from the classical physics ones.

References

[1] C. Ericson, *Real-time collision detection*.