Overview and in initial specification

My Physics for game assignment project is project that simulates physics collisions between spheres and planes. The OPENGL was supplied by another lecturer from the university and everything else was made by me. Physics uses Euler integration.

Research

The research I looked into was how to do the specific collisions and what formulae was needed. One thing researched was sphere to sphere to sphere collisions. Studio Freya’s article was useful for showing how to get the distance between spheres and why normalising the vector to create the normal was important. One this this article did that I wouldn’t and didn’t do was addressing the x and y coordinates as separate. This is easy for explanation and when you don’t have the code structure. But for readability and scaling was completely hideous.

High level description

First of all, all of the thing rendered in the simulations are an object Game Object. This is useful as everything is in the same format. The rendering updates are handled together in an Object Manager. In terms of Physics, a physics class can be applied to the Game Object. This class handles all the physics in the program, the physics classes are handled by a physics manager. Having the Objects managers is a very good way to handle all of the physics simulation and collisions. The more Objects that are put into the simulation and exponential number of collisions take place because every object needs to be tested against every other object. For each physics object a Euler integration is calculated. This is a simple calculation that calculates the velocity of the object dependent on previous velocity and force applied to the object. As well as the distance the object moves dependant on its velocity. The Physics manager goes through all the physics objects and handles the updating of the objects and the collision detection for those objects. The type of collision is determined, and different methods are used for each. The Sphere to Plane Collision is the simpler of the two collision types. It takes the direction of the objects force and multiply it by the normal of the plane. This results in the object bouncing. The more complicated of the two is Sphere to Sphere. This involves using a formula given in Physics for games slides. The Impulse of the objects is calculated by the force and direction of both objects, the normal of the collisions and then the mass of the objects. This gives a scalar force that then is then multiplied by the normal. This is then applied to the objects force.

Analysis

Testing the proformas with the simple example used in the demo video, three ball having on of each collision easily 60 fps adding in 5x more balls randomly placed resulted in no effect on performance. This is obviously a small sample size and the computer I used to test this is a very powerful one. If I were to do more testing it. Having said that, the number of tests for collision is going to be exponential meaning 255 collision detections. The strengths of my program are that I thought more about the structure of the program in terms of Game Objects and physics objects as a result adding objects and changing their statistics is very easy and quick as well as update functions. A weakness and area for improvement for my project would be how the collisions are handled. The collisions are done in a chain of if statements that check for both types and funnels into each algorithm. If I were to change this, I would simplify this and make it so that one object could have multiple collisions. Another area would be separation of the physics and Physics manager, this caused problems when wanting to transfer data from class to class efficiently and safely. Making all of maths done in one area would be both safer as less functions could be exploited and quicker as less function are called.

Reference

Studio Freya, 2018. SIMPLE SPHERE-SPHERE COLLISION DETECTION AND COLLISION RESPONSE[online]. Norway. Available from: <https://studiofreya.com/3d-math-and-physics/simple-sphere-sphere-collision-detection-and-collision-response/> [Accessed 01 January 2019].

Solar System Scope, 2018. Solar Textures [online]. <https://www.solarsystemscope.com/textures/> [Accessed 26 January 2018].