Coursework Report

Paulius Bieksa
40216180@napier.ac.uk
Edinburgh Napier University - Physics-Based Animation (SET09119)

1 Abstract

This project aims to implement a physics simulation of a domino run. The main focus is on 3D collision detection and collision response. This has been achieved using bounding volumes; mainly object oriented bounding boxes.

2 Introduction

Project Aims The aim of this project is to show the understanding of various mathematical concepts that are required to implement physics simulations and show an ability to program a robust framework that allows the physics simulation run in real time.

3 Implementation

Bounding volumes While detecting collisions with information gotten directly from a rigid body is possible, a more elegant solution is to implement a class that deals with bounding volumes. The specific bounding volumes implemented in this project are spheres and OBBs. Each of them have their own class, which inherits from a bounding volume parent class. This has the benefit of not needing to handling collisions regardless of the type of bounding volume in a single class.

Detecting collisions To detect collisions the separating axis theorem *(Figure 1)* is used in this project. When dealing with spheres there is only one axis to check - the axis along the translation vector between two spheres. When dealing with OBBs however, to comprehensively check whether a collision has happened up to 15 axes have to be checked. The collision detection algorithm checks the radii of two OBBs projected onto each of the fifteen axes until it finds there is no collision or until all the axes have been checked and a collision is therefore found.

Collision plane and collision point detection To calculate the impulses required to do collision response a plane of collision normal and a point of contact are required. The collision detection algorithm has been modified to calculate these values if a collision has been found. The normal of the collision is just the axis where the intersection is smallest normalised. The point of collision is a complex process however.

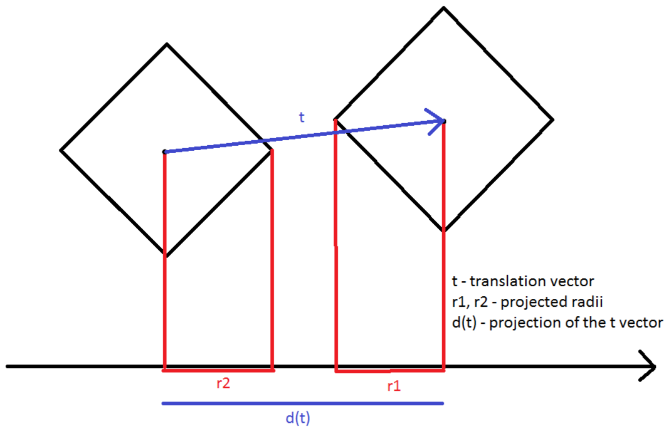

Figure 1: **Separating axis theorem**

Firstly, to calculate the point of contact, a point on the plane of collision is found to define the said plane. This is done by finding the radius of one of the objects projected on the axis of collision, subtracting half of the intersection distance, scaling the axis of collision to that length and adding it to the position of that object. At this stage all vertices that are beyond the collision plane *(Figure 2)* can be found with a help of a dot product, however some of those vertices may not be involved in the collision or in some face to face or edge to edge cases the average of only some of them is the collision point.

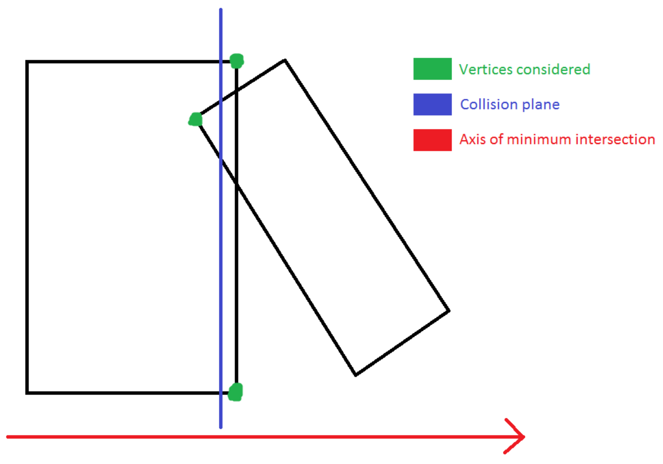

Figure 2: **Collision plane**

To find which vertices are required to calculate the collision point, this implementation finds the limits of the vertices of both objects tangential to the collision normal.

Figure 3: Impulse magnitude

$$j_r = \frac{-(1+e)v_r \cdot \hat{n}}{m_1^{-1} + m_2^{-1} + \hat{n} \cdot (I_1^{-1}(r_1 \times \hat{n}) \times r_1) + \hat{n} \cdot (I_2^{-1}(r_2 \times \hat{n}) \times r_2)} \quad (1)$$

The first step in doing this is constructing a rotation matrix with the collision normal as the x axis and getting the other orthogonal axes depending on the method of getting the normal of the collision. The tangential axes can be arbitrary as long as they complete the rotation matrix without skewing it. After getting the rotation matrix the vertices in model space are rotated using matrix just calculated. Now the limits on the rotated y and z exes are calculated and all vertices from each object that are beyond the limits of *both* of these axes are discarded from consideration. This ensures that face to face and edge to edge collision points can still be found. The number of collision points is calculated for both objects and then the average of the considered vertices for the object with lower number of vertices considered is used as a collision point.

Collision response Collisions are resolved using an impulse based method, using the formula in *(Figure 3)*. The collision response also features the Coulomb friction simple friction model for both ground and object to object collisions. This model calculates friction based on the impulse being applied to the surface, which means that heavier objects and objects hitting a surface with higher momentum experience more friction. This implementation also calculates the resting friction threshold.

Broad phase collision detection A simple broad phase collision detection technique has been implemented. The broad phase pass uses computationally cheap sphere bounding volumes to check whether a collision is possible between the two object before continuing to do the more expensive collision check. This should ensure that a large number of physics objects can be simulated in the scene as long as there aren't too many simultaneous actual collisions.

4 Future Work

Due to time constraints this project explored only the basic concepts of collision detection. Given more time collision detection between different types of bounding volumes could be implemented. Another avenue to explore is mass spring systems and their interaction with rigid bodies. Mass spring systems can be used in a plethora of ways - from cloth to soft bodies.

5 Conclusion

In conclusion, this project was a very educational undertaking. The maths required to implement the algorithms are well worth understanding and programming within such a minimal framework hones the coding skills well.