3D Collision Detection Using Non-static Psuedo-bounding Surfaces for N Solids

TJHSST Computer Systems Lab, 2007-2008

Richard Hooper

Abstract

Collision detection is a very useful concept, it is used in various applications from surgery to manufacturing to video game design. The purpose of this project is to create an efficient algorithm for detecting collisions for use in a gaming environment. The objects in the simulation are simple solids, and the algorithm is designed to handle many solids at once without a slowage of frame rate.

Introduction

The purpose of this project is to create an efficient algorithm for 3D collision detection. This project has value because there are many different applications for collision detection, and in game development, as with all other fields, efficiency is of extreme importance.

Collision detection is the concept of first detecting possible collisions, then contact, and then determining how to react to the collision. My algorithm is designed to detect multiple collisions without slowage. The first step in the development was to create a simple 2D algorithm that would model collisions as a prototype, followed by a simple 3D algorithm. This was then redesigned, and then the number of solids in the given space was increased, and the time taken and accuracy were tested. The results were graphed and analyzed.

Discussion

The algorithm itself is an extension of the bounding solids concept. It simplifies it by creating 360 vectors to simulate the presence of bounding sphere. The algorithm would iterate over the points and check to see if there was another solid there, and how close it was. In that way, collisions are detected. The algorithm was further optimized by having the vectors constantly change in length. The vectors start by having one vector at full length, with ever other vector decreasing slightly as they get further away, until they reach the minimum. The vectors then all decrease in length until they reach the minimum, where they increase.

Results

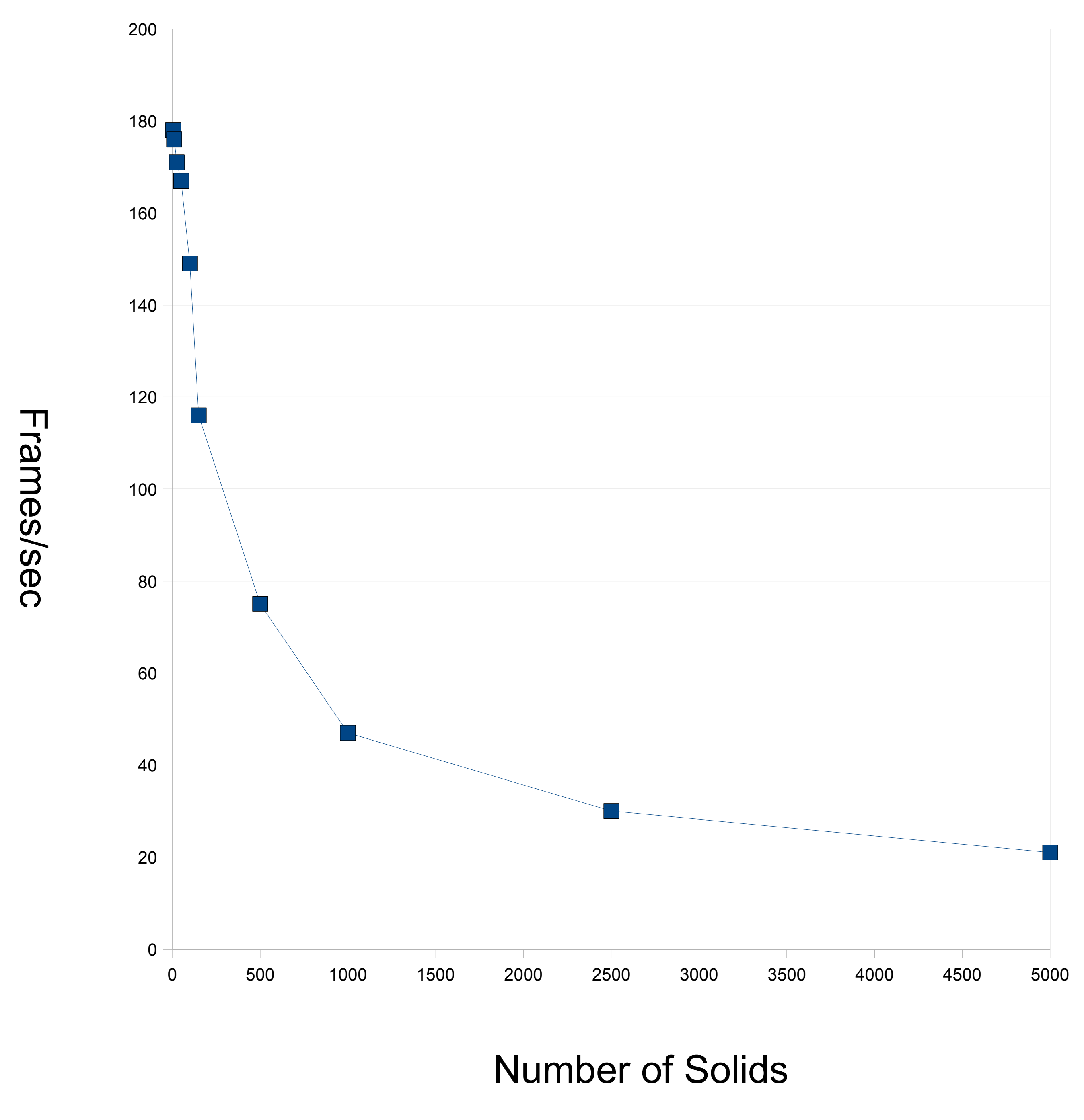
The frame rate was shown to experience exponential decay as the number of solids increased. The frame rate stays relatively high throughout however, which is a very good result, the algorithm was also found to be 100% accurate for spheres and 98% accurate for other solids.

Conclusion

The project was deemed a success. The frame rate stayed at an acceptable level (above 30 frames/sec) all the way up to 3217 solids, which is a better result than expected. The shape of the graph was to be expected, because the algorithm was designed to take a $O(N^2)$ function and reduce it to a smaller $O(N^2)$ rather than to create a different algorithm that worked in different time.

An interesting direction that a new project may be taken in would be to implement a tree hierarchy or other similar data structure to try to bring the algorithm into logarithmic time.

Frame rates of Algorithm for tested levels of solids


Frames/sec
Number of Solids