# TJHSST Computer Systems Lab Senior Research Project Paper
# Three dimensional collision detection using non-static psuedo-bounding surfaces for N solids using graphics with OpenGL
# TJHSST Computer Systems Lab 2007-2008

Richard Hooper

June 11, 2008

**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 framerate.
**Keywords:** 3D, graphics, collision, collision detection

## 1 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.

## 2 Background

An important setting in which this would be used is in game development. In video games, it may be necessary for many objects to interact in space, and in video games, there can be no slowages as they are supposed to be a real-time simulation, and pauses for calculations cannot be accepted. Other applications include surgery, as simulations are used in the preparation, machining and animation.

Some alorithms used include raytracing, which creates vectors, or "rays" and uses them to detect possible collisions. Others include using bounding solids or using a simple point in polygon test, which is similar to raytracing. Another method uses bounding surfaces to estimate when collisions take place. This concept is also applied to more complex meshes by using separate surfaces for each unique part of the mesh.

After conducting research, the system that most applies to my particular project was determined to be variation on bounding surphaces where the number of points checked is minimized by having the the bounding solid constantly change its shape. This idea is called Non-static bounding solids.

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.

## 3 Development

This project is an effort to create a fast and efficient collision detection algorithm. Success is considered a working algorithm that can successfully detect collisions for one hundred solids without slowage. Anything less would be considered a failure.

The language used is C using OpenGL, because C is a powerful language that can accomplish all my needs efficiently and has strong object orientation built in. OpenGL is an easily accessible graphics library, which more than suits my need for simple graphics.

The workplan for the project was as follows: write a 2D algorithm, then write a 3D algorithm, then optimize the 3D algorithm or rewrite it to meet my time constraints.

An interface was also created to help with the visualization of the algorithm. This interface features a box which clearly marks the boundaries of the 3D area in which the solids interact. The box can be rotated, providing a better view of certain areas of interest if needed, and if a certain event needs to be re-examined,

the program can also pause, rewind, and fast forward the simulation.

# 4 Discussion

The algorithm uses non-static pseudo-bounding surfaces to detect collisions. These surfaces are supposed to simulate a constantly changing solid which is designed to add minor calculations as a replacement for the necessary iterations inherent in bounding surfaces. The algorithm further reduces the number of iterations by using equally distributed vectors all around the solid which represent the bounding surface. Hence the name pseudo-bounding surface.

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.

# 5 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 percent accurate for spheres and 98 percent accurate for other solids.

# 6 Conclusions

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(N2) function and reduce it to a smaller O(N2) 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.

# 7 References

1.The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC machining - T.D. Tang, Erik L.J. Bohez and Pisut Koomsap

2.Fast continuous collision detection among deformable models using graphics processors - Naga K. Govindaraju, Ilknur Kabul, Ming C. Lin and Dinesh Manocha

3.Deformable free-space tilings for kinetic collision detection - Pankaj K Agarwal, Julien Basch, Leonidas J Guibas, John Hershberger, Li Zhang

4.Fast Collision Detection based on nose augmentation virtual surgery - Kai Xie, Jie Yang, T.M. Zhu

5.Continuous Collision Detection for Articulated Models using Taylor Models and Temporal Culling - Xinyu Zhang, Stephane Redon, Minkyoung Lee, Young J. Kim

6. Collision detection between complex polyhedra - Juan J. Jimenez and Rafael J. Segura