TJHSST Computer Systems Lab Senior Research Project Proposal
Three Dimensional Collision Detection for N Solids Using OpenGL
2006-2007

Richard Hooper

November 5, 2007

Abstract

Collision detection is a very useful concept, it is used in various applications from surgery to manufacturing to video game design. My project aims to create an efficient algorithm for detecting collisions so that it can be used in a gaming environment. The objects in collision will be simple solids, and multiple will be put in a space to monitor their interactions. The first step is simple 2D collisions followed by more complex 3D collisions.

Keywords: 3D, graphics, collision, collision detection

1 Introduction - Elaboration on the problem statement, purpose, and project scope

1.1 Scope of Study

The project would start simple, with a 2D collision detection algorithm, then increase in complexity to a 3D algorithm and finally increase to a optimized algorithm. The final goal for the optimization would be to have the program capable of handling 1000 solids accurately without experiencing any reduction in frame rate due to processing time. The project would require extensive knowledge and research in the field of collision detection

1.2 Expected results

I would hope to end the year with a working algorithm that could handle 1000 colliding solids without any slowage. The results of this research could be useful in any environment where handling large numbers of collisions in real-time is important. The ideal application is video games, or other simulations, because real time collisions are essential in gameplay.

1.3 Type of research

This project is an example of use-inspired research, because the final algorithm could have real-world application, and I have videogames in mind, but the algorithm would not have much use outside of implementation in a larger program.

2 Background and review of current literature and research

This is a field looked into fairly often, usually with the application of Computer Assisted Machinery or medicine, or more specifically, surgery. Another approach is an optimization approach, trying to design the most efficient code possible. An example of this is "Fast continuous collision detection among deformable models using graphics processors," a study done which aimed at optimizing the collision detection process. This was a study in a more specific field called Continuous Collision Detection, or CCD.

3 Procedures and Methodology

In order to reach my objective, I would need to first get a working two-dimensional version working, then a simple three-dimensional version, even if it uses a different algorithm from the final version, and finally, a fully working and extremely fast algorithm would be necessary. I would accomplish this using the C programming language and OpenGL to handle the graphics.

Graphics and tables could be used to help analyze the data. Screenshots would not be helpful graphics, because the algorithm models motion, and screen shots can not accurately represent this. Useful information to print out would be the framerate of the graphics, because this would show how well the algorithm is representing a realtime environment. another set valuable data would be the time of each collision and a record of which solids it occurred between, along with a list of every single coordinate of every solid used, because although this may be unwieldy, I could construct a simple algorithm that could check possible collisions against actual collisions using simple bounding spheres. The comparison could be made with visual data, or it could be made against the actual collision printout data, and the similarities could be compared to a reasonable degree of accuracy that could be calculated. Hopefully, the main focus of my analysis would be run time.

A useful method of testing would be dynamic testing. Combining the above techniques with random sets of solids would provide useful data to analyze the efficiency and accuracy of the code.

I would expect the collision detection to work with one hundred percent accuracy, and it could be tested as described above, but this is not the main goal of my project. The true goal is to create an algorithm that is extremely efficient algorithm that would be able to handle large numbers of solids, preferably in the thousands, without experiencing any frame rate slowing at all. this could be measured by recording the framerate at every second and checking it against the goal, or graphing it against the goal.

A particular algorithm I am looking at right now is raytracing, although I don't think that that would be the algorithm i would use in the end, because although it works consistently and relatively efficiently, I don't think it will meet my goals at the end.

4 Expected Results

I would hope to end the year with a working algorithm that could handle 1000 colliding solids without any slowage. The results of this research could be useful in any environment where handling large numbers of collisions in realtime is important. The ideal application is video games, or other simulations, because real time collisions are essential in gameplay.

Ways of analyzing results include printouts of collisions, at what time they occur, and what response occurred, to be compared against visual data and lists of solid-positions, could be printed. Also, printouts of how many frames per second the code is running in could be printed out, so that the speed could be more easily monitored would be used. Other than that, any way of analyzing would require a previous solution to the problem.