3D Collision detection for N Solids in Open GL

Richard Hooepr

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.

Introduction

In this project, I plan 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. I intend to create an efficient algorithm that would detect collisions, so that the interactions of multiple solids could be modeled at once. The first step is to create a simple 2D algorithm that would model collisions as a prototype, followed by a simple 3D algorithm. This would then be optimized or redesigned, and then the number of solids in the given space would be increased, and the time taken and accuracy would be tested. The goal is to have the number of solids in space to be in the thousands, but the first benchmark would be in the hundreds.

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 (although one thousand would be preferable). Anything less would be considered a failure.

The language used is C using OpenGL, because C is a powerful language, and OpenGL is an easily accessible graphics library.

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

It has also become clear that the display previously used to show the simulation was inadiquate, and a new one was created. This new interface features a box which clearly marks the boundaries of the 3D area in which the solids interact. You can also rotate the box so that you can get a better view of certain areas of interest, and if a certain event needs to be re-examined, the program can also pause, rewind, and fast forward the simulation.

So far the 2D and 3D algorithms have been completed, as has the interface, and the next stage is to optimize the 3D algorithm. Unfortunately the current algorithm is not very robust and only works with certain solids. The new algorithm has been started, and is in its early developmental stages.