A Discrete-Space Physical Simulation

Adam Herbst
Computer Systems Lab 2006, Period 1

Abstract

A general simulation of macroscopic physical bodies calculable in real-time is a vital tool for game developers, physical theoreticians and other researchers. Such a simulation generally relies on division of objects into basic elements, usually particles of some kind. This project focuses on the use of discrete-space rather than discrete-matter or particle representation, as well as the relative utility of these two methods.

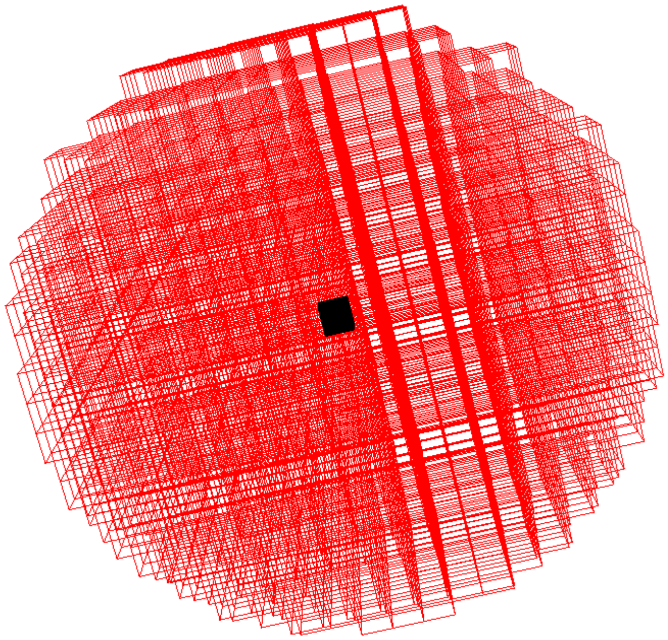
The Discrete-Space Method

In this discrete-space simulation, space is uniformly discretized in all dimensions. This results in elementary cells with individual attributes (mass contained in the cell, average velocity of this mass, average position of matter within the cell, coefficient of friction with adjacent cells). Large-scale objects are stored as contiguous groups of these cells.

During each time step of the simulation, mass is transferred between cells according to their properties; mass and velocity are recalculated to account for momentum changes. All empty cells are discarded after the completion of an evolution step, and new cells are created when mass moves outside of an object's existing bounds.

Dilemmas in Programming

Storing the cells that make up a given object is problematic, since this group is dynamic and has no clear spatial order (assuming a multi-dimensional simulation). Currently, the program uses a C++ *vector*, which is built so as to allow dynamic array alteration. Though easily coded, however, this is clearly an inefficient container. Removing cells is at worst an O(n) operation, and in adding a cell the entire list must be searched for a new cell's neighbors. A linked list with some sort of spatial order might be a better alternative.

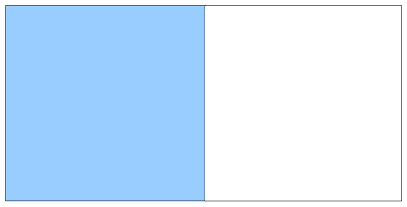


A radius-10 voxelated sphere

A Sample Time-Step:

mass: 1.0
position: (4, 3)
velocity: (0.5, 0)

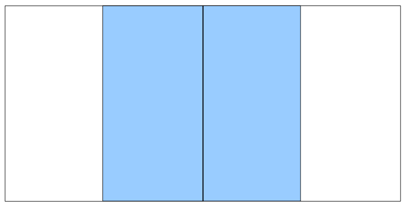
no existing
voxel at (5, 3)



t = 0

mass: 0.5
position: (4, 3)
velocity: (0.5, 0)

mass: 0.5
position: (5, 3)
velocity: (0.5, 0)



t = 1