Optimization of Finite-Element Physical Simulations

Adam Herbst

April 6, 2006

Abstract

Modelling behavior of and interactions between macroscopic solid objects is applicable to almost any type of macroscopic engineering, including architecture and machine design. It is also widely used in other forms of computing, including video gaming and astronomical simulation. One problem common to many such simulations, however, is the loss of precision when objects are represented by uniform, rigid, indivisible models. These models ignore the microscopic interactions that actually combine into macroscopic physical effects. The finite element method, however, treats each object as a collection of smaller units whose interactions are directly determined by the fundamental physical laws of the system. I propose a finite-element simulation that is optimized so as to allow large-scale simulations that can be computed in real time.

1 Introduction

The finite element method (FEM) of modelling objects assumes each object is composed of a certain number and variety of substructures with well-defined physical properties. One example is the creation of a train car from an array of straight metal rods in a crash simulation by the SRI Center for Fracture Physics. Each rod is considered indivisible within the simulation, and logical laws govern the interactions between individual rods. Using this method, an accurate simulation of a multi-car crash was computed. Any structure can be used as an element, and any laws can be implemented to determine their behavior. The method is inspired by the notion of matter being constructed from finite particles whose interactions are derived from constant mathematical relationships.

While FEM simulations are commonly used in scientific research and development, my proposed project focuses on FEM as a means of efficiently computing behavior of large-scale models so that the method can be applied to real-time simulation in video games, for example. The elements will resemble atoms in that they will be bonded within objects (chemical bonds), experience electrical repulsion and weak gravitational attraction as an inverse of the distance between them. In this case, however, the atomic scale will be much larger relative to the macroscopic scale than in reality, so that the memory and computation time required for a certain space state will be kept within reasonable limits.

The accuracy of the simulation is optimized by giving each element certain numerical attributes that affect its electrical, chemical or gravitational properties. This is acceptable because the computation time for a given space state grows with its size much faster than

the amount of memory it requires, so that even with added memory for each element, the simulation will exceed allowable computation time limits before it runs out of usable memory. In fact, by increasing accuracy in the form of memory, some accuracy resulting from added computation can be reduced with no net loss.

2 Optimization Techniques

I have attempted two different optimization techniques to improve the speed of the program while maintaining efficiency. The first involves separating particles into cells according to their position. Under this method, all particles in the system whose positions are within a given cube of space are grouped into a single cell. This associates particles with those that are near them spatially. This sorting takes place at the beginning of each iteration or time step of the simulation; during that step, each particle interacts only with those particles that are in its cell or an adjacent cell (its interaction with adjacent cells depends on its position relative to the center of its cell). Since the physical relationships in the system diminish as inter-particle distance increases, this restricts the computations to those that cause the most significant changes in the simulation state.

The second technique is largely an extension of the first. In each time step, a predetermined percentage of particles from each cell are selected randomly. During that step, only the interactions that involve these particles are calculated. The hope is that, over enough iterations, all particles' interactions are tested. The major problem with this algorithm is that when interactions are ignored for some time, the system can become unstable and even-

tually explode. I found that this occurred unless the fraction of particles tested in each step was nearly one, which, because of the random way of selecting particles, actually caused it to be slower than a straight iteration through all particles.

3 Findings

The first technique implemented, which operates by partitioning the spatial domain of the simulation system, proved to be significantly more efficient than a blind calculation of all possible interactions, with little loss in accuracy even for cell boundaries close to the size of the individual elements. Its relative efficiency actually grows exponentially with the size of the simulation, though it also depends on a sufficiently compact system. Although this is a tribute to the technique itself, it is also an indication that, in a physical system with forces inversely related to distance between particles, almost all interactions can be ignored without noticeable divergence from an ideal simulation. This suggests that the optimal such program does make use of approximative collision equations precalculated from momentum conservation, rather than tracking collisions through distance-dependent forces. Therefore, future searches for a perfect real-time simulation might focus on ways of representing objects and space instead of minimizing unnecessary calculations.

4 Voxelation

Voxelation is a discretization of space into volume elements (voxels). In this case, each voxel contains a stored amount of mass which is estimated to have uniform velocity throughout that cell. An object, instead of a collection of particles, is made up by a contiguous set of such voxels. During a time step of the simulation, the mass from a given voxel is distributed throughout neighboring voxels in its native object, according to that voxel's (non-discrete) velocity. After a step, any empty voxels are discarded, and new ones are created wherever mass has been transferred to cells that are not already included in the object.

Forces between voxels are dependent on their relative velocities. These are not calculated as accelerations, but as finite changes in this relative motion. The strength of these relationships depends on the rigidity of the object, so that forces between voxels of a gaseous object are extremely weak compared to those in a solid model.

A significant problem quickly rises from the lack of accuracy in positional data. The mass transferred from one voxel to another is effectively assumed to be distributed uniformly in each voxel, although this is usually not the case. Because of this, the mass in an object may spread out as the object moves. The solution is to store not only the amount of mass in a given voxel, but also the mean position vector of all mass within it. This provides much greater accuracy with only a few added calculations per voxel, and tends to eliminate gross decentralization of mass, though some still exists.

Using a spatially discrete simulation space is useful in that, while individual voxels behave very much like particles, there is no overhead in finding the significant interactions among all

those possible; each voxel automatically interacts only with those immediately surrounding it. Also, since there is an assumed distance between any two cells, there is no risk of simulation instability that is caused by particles coming too close to each other. One potential issue for large simulations is the time spent searching through the voxels in an object in order to pair newly created ones with their neighbors. However, this might be reduced by careful indexing and effective search algorithms.

References

[1] Structural Simulation, SRI International. www.sri.com/psd/fracture/fe_program.html, 2005.

[2] Fedkiw, Ron. Research. graphics.stanford.edu/ fedkiw, 2005.

[3] Finite Element Analysis. ceaspub.eas.asu.edu/structures/FiniteElementAnalysis.htm, 1994-2004.

[4] Tsap, Goldgof, Sarkar. Efficient Nonlinear Finite Element Modeling of Nonrigid Objects via Optimization of Mesh Models. http://figment.csee.usf.edu/ sarkar/PDFs/femcviu.pdf. October 15, 1996.