Ross Adelman  
15468 A

Collision Detection on the GPU

For project 4, I decided to implement on the GPU using CUDA the collision detection code I developed for project 3 last semester in Animation of Natural Phenomena.

I have included a copy of the write-up for that project – see proj3descr.pdf. The goal of that project was to implement a rigid body simulator using signed distance functions (SDFs) instead of triangle meshes to represent the boundaries of the bodies. The way I did this was by having a base class, ra_body, which contained a virtual function, sdf(). I would extend this class and override this function to create various types of bodies, such as spheres and cubes. This worked great on the CPU using C++, but, unfortunately, as a C compiler, the CUDA compiler does not support the higher-level features of C++, such as virtual functions. Instead, when passing information about each body to the GPU, I included a few extra parameters, including the body type and some geometric properties. Then, I hard-coded the SDF calculations into the CUDA code.

In the CPU version of the simulator, this is how the collision detection routine worked: (1) do a broad-phase check using bounding spheres to determine which bodies could be colliding; and (2) for each pair of bodies that could be colliding, do a narrow-phase check to see whether they are actually colliding, and, if so, where. The GPU version works almost the same way. However, instead of checking each pair sequentially, I launch a thread for each one so that they run in parallel. Then, on the CPU, I inspect the results of these threads to determine which pairs of bodies could be colliding. For those that are, I launch a thread that performs the narrow-phase check. On the CPU, I inspect the results, and for the pairs of bodies that are colliding, I respond appropriately. Thus, in total, I wrote two CUDA kernels, along with one helper function, which calculated the SDF of a body.

On my computer, which has a quad-core Q6600 and a NVIDIA GTX 260, I achieved a 3x speedup from the CPU version to the GPU version. A graph showing this can be seen in proj4pres.pdf. I also ran the CPU and GPU version on one of the Gates machines. There, the CPU version ran much faster and the GPU version ran a tad slower – the CPU version was almost as fast as the GPU version.

In the future I would like to do several things. To begin with, the broad-phase detection algorithm is horribly slow. There are a variety of faster broad-phase algorithms, many of which run in close to O(n) and are parallelizable. In addition, I did not use __shared__ memory or any other GPU tricks – doing this would probably speed things up a tad. Also, I would like to implement in CUDA the other critical sections of code in the rigid body simulator, including the force accumulator, the integrator, and the collision response code.