Shaun McThomas 13828643
Matt Yefima 37442442
CS 114 Project proposal
May 12, 2016

Introduction:

One interesting aspect of computer graphics is scene management for interactive applications. As the number of objects in a scene increases, the amount of collision detection that must be performed in each frame increases. Although several techniques exist to reduce the performance load of real-time collision detection, for this project we will focus on Octrees. Octrees reduce performance costs by spatially partitioning objects in the scene into tree nodes. Objects that are far apart occupy more distant nodes in the Octree, and objects that are nearby occupy closer nodes or the same node. By partitioning objects in this manner, collision detection can be ignored for objects that are far apart and thus have no chance of interacting with each other, improving overall performance of the application.

Project Proposal:

We intend to implement our own collision detection and Octree in the Unity3D game engine, comparing our implementation of collision detection to that provided by Unity3D for accuracy. Our final scene will show the subdivisions of 3D space provided by the Octree, and highlight any candidate objects for collision detection. As an additional goal for this project, we will use any remaining time we have to write our own shaders for the scene.

Proposed Timeline:

Week of 5-16:

Install Unity, research various collision detection, and begin implementing the Octree in Unity.

In this week we will mainly be focusing on getting a good working understanding of how Octrees work and how collisions should be handled. This includes the Physics involved on how to transfer the forces of the collision.

Week of 5-23:

Finish Unity implementation of the Octree and begin to implement our own implementation of collision detection for simple shapes (cubes, spheres, planes...).

By this point, we should have a pipeline to create scenes using Unity's built-in collision detection methods and be well on our way to having our own implementation written to do the same.

Week of 5-30:

Design and render scene(s) with somewhat complex models using both Unity's built-in collision detection and our implementation to create a side by side comparison.

After this stage we should have enough implemented to produce a demonstration of how Octrees work and how they are used for collision detection.

Week of 6-6:

Fix any bugs in code and catch up on any areas we fell behind. If time permits, write own shader code.

Past experiences tell us that delays or other setbacks may occur in working with any new technologies. Because of this, we have decided to plan for delay. This time will be set aside for getting back on schedule. However, if we are on schedule, we plan to write our own shaders for the scene.