Parker Whaley

Final

For this final I set out to create a simple robust physics simulator and graphics rendering. To accomplish this I set up new classes and a physics engine which is as fast as the user chooses to make it.

First let me describe the classes. There are two classes, objects, and instances of objects. In this code there are three objects, a sphere, a cube, and the landscape. An instance of an object has a texture, position, and velocity, among other traits. The position a instance holds is a matrix that maps the object, thus any contortions of an object, like a ellipsoid are simply instances of an object.

Setting up a new type of object is simple, make a new derived class with a constructor that initializes the list of points, and then calls the setup function. Then make the two required functions, one that can tell if a point is inside your object and one that can tell if a ray intersects the object, or returns -1 if the item is not selectable.

Once these are set up the code will do everything else, including physics. See the initialization in the code. If two objects are interacting too often and taking too much time the user can write a fast check function that will supersede the built in physics and allow that interaction to run faster. As a demonstration in my code I did this for the sphere ground interaction. Since both the sphere and the ground had many vertices it was taking the default checker too long to check if the two collided, the fast checker eliminated this problem and allowed the code to run faster.

As a demonstration of the physics observe how the cube and sphere impact each other, a very cool interaction as it sends the ball flying.

The selector is working however I don't have it doing anything interesting, if you right click on an objet the code spits out what objet you clicked and its distance.

The camera controls are relatively simple, wasd+qe and a grab and drag interface with the left mouse button.

The demonstration I present is not the impressive bit, the cool part of this code is actually how robust it is and how easy it is to add in new functionality.