INFO6022 - Physics 2

Project #1

Due Date: Sunday, February 11th at 11:59 pm

This is an individual assignment. One submission is expected per person.

The submitted code must compile in Visual Studio 2015. If it does not compile, then the mark assigned will be zero.

# Spheres!

For this project, you will create a program that demonstrates collision detection and reactions between sphere-sphere, and sphere-plane pairs.

You'll create a scene of an enclosed world (enclosed by planes!) In this world are 10-20 spheres that can be controlled. The user can take control of any one of the spheres and "drive" it around the scene to witness all the marvelous physics in action!

|  |  |  |
| --- | --- | --- |
|  | Item | Marks |
|  | **MUST HAVES: ( Just do it, or you get zero ☹ )**  **Compile:** It’s gotta compile.  **Decoupling:** Except for where you instantiate a concrete physics factory, only your interface library may be referenced in your code. Almost all your marks from upcoming Project 2 will depend on this!  **External File:** The scene is defined via an external file, all entities’ initial physical values should be able to be manipulated here.  **Reasonable Rendering:** The rotation of the spheres must be visible, so wire frames at the very least will do, but textures and shading is best. No solid circles on solid planes – how do you expect to see all the cool physics happening??  **A darn good camera:** The camera is at a constant height, and points at the sphere that is being controlled just like a 3rd person camera… ok, it’s a 3rd person camera.  You can zoom IN and OUT.  Note: set some reasonable max and min zoom levels for your camera so that I can see most of the scene by zooming all the way out |  |
| 1 | Integration scheme is RK4, or equivalent 4th order method. | 5 |
| 2 | There is 1 plane that acts as a floor, and 4 planes acting as walls enclosing the scene; all with appropriate collision detection and reaction. (note the planes are static so they don’t interact with each other) | 5 |
| 3 | There are 10-20 individual spheres, **each with a different radius** (and corresponding mass). They behave as free, rigid bodies. | 5 |
| 4 | Reactions are between sphere-sphere pairs, and sphere-plane pairs. These reactions are *elastic* – that means some energy is lost when they interact.  **It "looks right"** - meaning I should see what I would expect to happen in the real world: small spheres being pushed around and bouncing off of larger ones, while larger spheres more easily push past smaller ones, spheres rolling around! | 15 |
| 5 | A single sphere is under the control of the user, and has a distinct color to show it.  There is a hot key that can decide which one is currently under control.  The controls are the ASDW keys, s.t.:  A - applies a force left  S - applies a force backward  D - applies a force to the right  W - applies a force forward  Where "forward", "backward", "right", and "left" are **relative to the camera and orthogonal to the "floor" plane**! | 5 |
| 6 | **BONUS :** You’ve implemented a debug renderer. There’s a hot-key that can switch the rendering to some physics “debug” mode, ie. It is driven entirely by the positions and shapes in your physics engine. Ideas: Include lines for radii, translucent or wireframe spheres, change colors during collisions. | 5 |
|  | TOTAL: | 35 + 5 |