COMP 5823M: Animation & Simulation 2023-2024

ASSIGNMENT 3: Collision Response **[40 marks]**

You have been provided with a simple interface derived from Assignment 1. For this assignment, you will implement collision detection and response with physics.

For full marks, you will need to complete ALL of the following.

**Task I: Bouncing Ball [10 marks]**

You have been provided with a spheroid model (it’s actually a subdividied icosahedron, but it’s close enough to a sphere for our purposes). Start the ball at a height of 10 m above the ground, with a lateral velocity of (5.0, 0.0, 0.0). Thereafter, apply gravity and impulse forces, and use a coefficient of elasticity of 0.6.

You should see the ball bouncing upwards repeatedly. Note that a sphere will always contact the ground at a single point at the lowest point of the sphere, which greatly simplifies the problem. Also, since the ball is spherical, you do not need to worry about implementing rotations.

For the collision test with the ground, it is acceptable to test whether the z-component of the centre of the sphere is above the terrain by less than its radius.

**Task II: Uneven Surface [10 marks]**

The scene has been provided with three different surfaces for you to test with. Use the ‘l’ key to switch between them. The rollingLand surface is the same one we used in Assignment 2, while the stripeLand surface is a single stripe of that surface combined with flat land elsewhere, which gives some interesting sideway bounces.

Modify your scene to use an uneven terrain, bouncing with respect to the local normal of the surface. Once this is working, use the ‘<’ and ‘>’ keys to change the angle of the launch direction.

For collision testing, note that the radius of the sphere is 1m, while the terrain is in 3m squares. It is therefore acceptable to test the sphere only against the ground directly under the sphere’s centre of gravity.

**Task III: Bouncing Dodecahedron [20 marks]**

Substitute a dodecahedron for the spherical ball, and get arbitrary bounces working. You will need to implementation both impulse forces and rotation. It is easiest to get the linear impulse forces working first, then add the rotational impulse afterwards.

Again, you can simplify the collision test by comparing only against the ground directly beneath the centre of the dodecahedron, but you will need to detect which vertex collides with the ground first.

**USER INTERFACE CONTROLS:**

In order to make it feasible to mark, you **MUST** use our choice of controls, even if you think you have a better solution. The following keys are defined:

WASD: Translate the camera position (like Assignment 1)

RF: Move the camera position up and down

QE: Rotate the camera position

P: Reset the character position

L: Switch land between flat, stripe & rolling

M: Switch model between sphere and dodecahedron

<>: Rotate launch angle

All code should compile on the School's Linux machines without installation of any extra libraries or applications. You should include a makefile and a readme.txt file with any additional instructions.

**FILE NAMING:**

In order to make the marker’s job easier, please make sure that you rename the directory from A3\_handout to xxxxxx\_A3 where xxxxxx is your userID. For example, since my userID is scshca, I would rename it to scshca\_A3.

**PENALTIES:**

Poorly structured or badly commented code may be penalised by up to 25% of the marks available.

Code without a readme may be penalised by up to 10% of the marks available.

Code that does not compile properly will be assigned a mark of 0, but I will usually give the student one chance to correct this.

**DUE DATE:** Thursday, January 18, 2024, 10:00am