CMPT466-766 Computer Animation

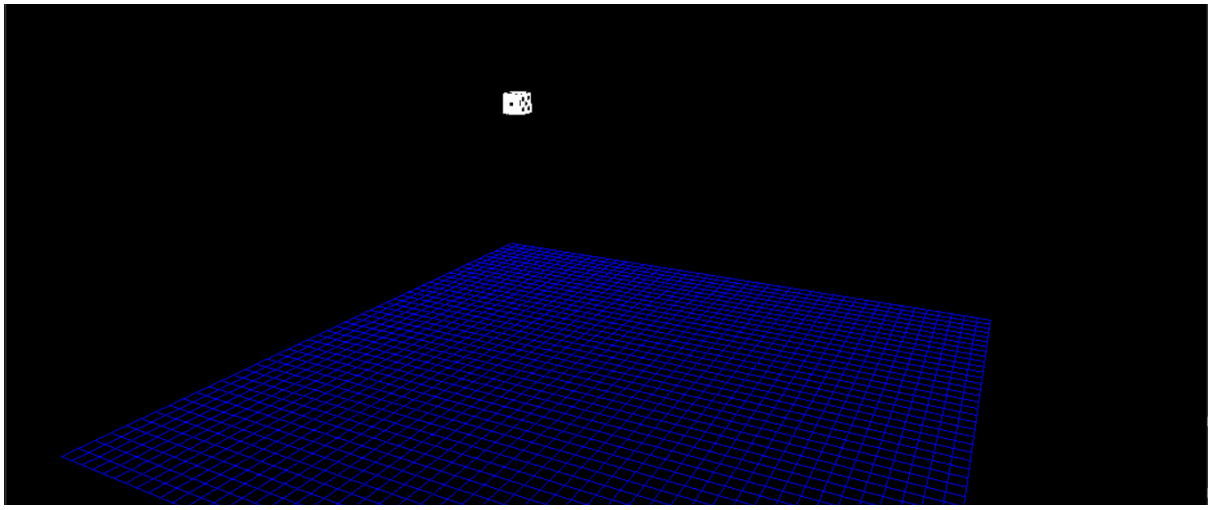
Programming Assignment 3: Rigid Body Simulation (15 points)

Due date: **23:59 Sunday March 27, 2022**

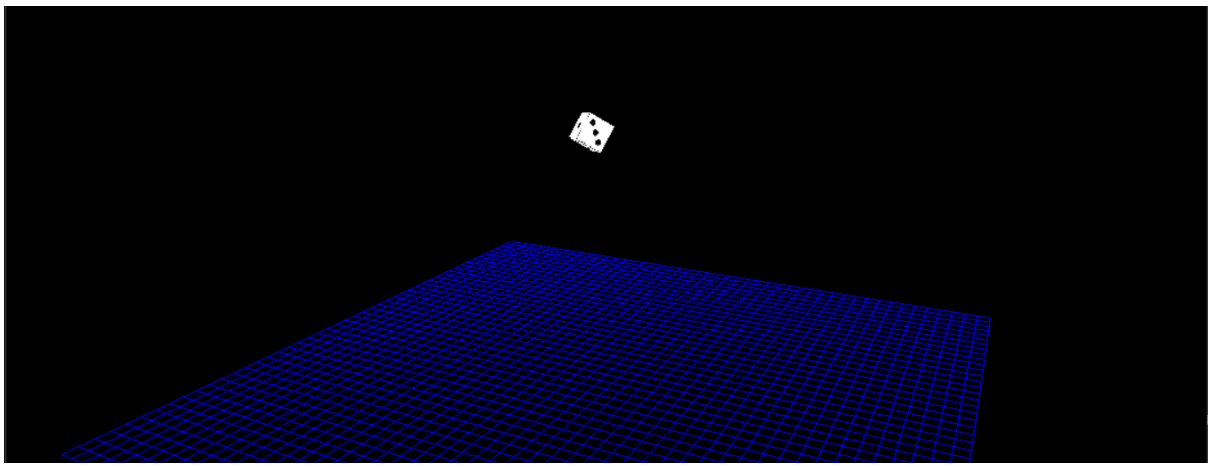
In this assignment, you are going to implement a Rigid Body Simulator using Forward Dynamics. You need to simulate the motion of a dice when its thrown in the mid-air with an arbitrary initial velocity, and then the dice should drop and bounce on the ground floor and rotate etc. The default coding environment is using Python, PyGame and PyOpenGL.

Introduction to the project

As you run the `rigid_body_sim.py`, a pygame window will pop up and you will see a dice floating at the top of the floor as shown in the figure below.



When you first run the template code you would just see a static scene and nothing would move. Once you complete your coding, you can start the simulation by pressing **'s'** on your keyboard. This would run the simulation till you explicitly close the window, or restart the simulation using key **'r'**. Each time key **'n'** is pressed, one more timestep is simulated and displayed. You can also pause the simulation by pressing **'p'**.



Where to code?

The template is given as `CMPT466-766_Program_Assignment_3.zip`. Extract it. Open the project in your favorite editor.

1) `rigid_body_sim.py` Do not edit this file.

   This file is the entry point to the code. It renders the frames and hold the necessary OpenGL code.

2) `quaternion_computation.py` Do not edit this file.

   Contains utility functions for quaternion. You can visit these files to reuse code.

3) `render_dice.py` Do not edit this file.

   This file loads the textures for the dice and renders it.

4) `rigid_body.py` The only file you need to modify.

There are **four sub-functions** be completed within `rigid_body.py`

```
simulate_one_substep
|-- collision_detection
|-- calculate_forces_and_torques
|-- calculate_accelerations
|-- integrate
```

The `simulate_one_substep` function is responsible for simulating one substep for the rigid body. It should compute the updated values for the four state variables –

1) `velocity` - 3-dimension ndarray
2) `position` - 3-dimension ndarray
3) `angular velocity` - 3-dimension ndarray
4) `quaternion` - 4-dimension ndarray

Coding Part

Part 1) `collision_detection()` (3 /15 points)

Input:
  None
Output:
  Boolean: to determine whether a collision has occurred
  2D List: List of vertices that collided with the floor

You need to check when and where a collision occurs between the eight vertices of the dice and the floor.

Part 2) `calculate_forces_and_torques(isCollision: boolean, collision_vertices: ndarray)` (4 /15 points)

Input:
- **Boolean:** to indicate whether a collision has occurred
- **2D List:** List of vertices that collided with the floor

Output:
- **Total force and total torque that need to be applied to the rigid body**

In this assignment you can simply use the penalty method for collision response. You need to aggregate per-vertex quantities including the penalty forces and torques. and then return the resultant total force and torque for the dice.

Part 3) `calculate_accelerations(force: ndarray, torque: ndarray)` (5 /15 points)

Input:
- **Total Force**
- **Total Torque**

Output:
- **Linear acceleration and Angular acceleration**

Based on the total force and total torque calculated in the previous part, you need to calculate the linear and angular acceleration for the dice here.

Part 4) `integrate(linear_acceleration: ndarray, angular_acceleration: ndarray)` (3 /15 points)

Input:
- **Linear Acceleration**
- **Angular Acceleration**

Output:
- **Updated state variables (velocity, position, angular velocity, quaternion)**

Integrate accelerations to update the state variables of the rigid body.

If everything that you implemented is correct so should see a smooth moving dice. You can play with the initial state variables to see if your implementation still functions correctly. You can also observe different simulated behaviours using different simulation parameters.

Submittion

Please submit a zip file with student number and your name (i.e., **300000001_TerryFox.zip**). The zip file should contain all the python files (edited or unedited) as well as the texture files.