**Advanced Games Engine Creation Tutorial 18**

In this practical you will practice implementing various aspects of game physics

* Writing some overloaded operators for the Vector3 struct or class
* Using variable timestep rather than fixed timestep updates
* Adding acceleration due to gravity so the teapots fall and bounce
* Adding a bit of drag so they gradually slow down
* Modelling inelastic collisions
* Adding collision response to the teapot-teapot collisions
* Refactoring your code to make a more flexible physics engine

We will start with the collision detection example code we used last week. You can carry on where you left off, and or download a fresh copy from Blackboard.

1. **Write some overloaded operators for the Vector3D struct or class**

Add overloads for the operators + and += to your definition of Vector3D, as discussed in the lecture. Test each overload by using them in the Teapot Update method.

Also add and test overloads for the - , -=, \* and \*= (scalar multiplication) operators

2. **Variable timestep updates**

As discussed in the lecture, modify your application so that the parameter **deltaTime** of the Update method in GameScreenLevel1 is passed to the Teapot Update method. Modify the teapot Update method to use deltaTime in calculating the new teapot position, rather than assuming a fixed timestep.

Run your application. You will probably need to change the range for the random initial velocities set in the constructor.

3. **Acceleration due to gravity**

Add another member variable, acceleration, to the Teapot class. Initialise this value to

(0, -1.0, 0) in the teapot constructor.

Change the Teapot Update method so that both velocity and acceleration are used to update the teapot position:

position += velocity\*deltaTime + acceleration \* (0.5 \* deltaTime \* deltaTime);

velocity += acceleration \* deltaTime;

Adjust the acceleration value if the teapots are falling too fast or too slow.

You should see the teapots bounce. Examine the code to see why this occurs.

Change the Teapot constructor so that the initial velocity has a random z component instead of zero. Adjust the position of the camera to get a better view of the teapots bouncing off the group (change the parameters of glLookAt in the GameScreenLevel1 Render() method).

4. **Modelling drag**

Use a very simple model to slow down the teapots each frame – simply multiply the final velocity by 0.98. Observe the effect with and without any acceleration due to gravity

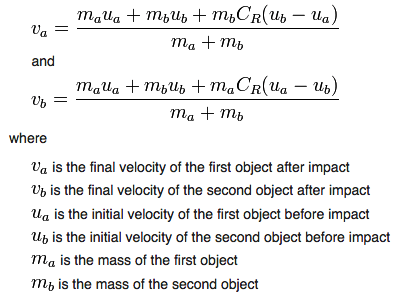
**5. Modelling inelastic collisions**

Find the code that handles collisions of the teapots with the “walls” of the world. The collision is assumed to be elastic, with the coefficient of restitution ℇ = 1. Change the code so that ℇ is variable. Experiment with different values between 0 and 1.

**6. Adding teapot-teapot collision response**

Experiment with adding collision response to your teapots. You need to:

* Move the colliding objects apart so they are no longer touching (the easiest way to do this is to store the previous position each frame, and back up to the previous position if a collision is detected
* Calculate the new velocities of each teapot, using the formula



And CR is the coefficient of restitution.

**7.** Refactor your application so it uses variables and method parameters rather than hard-coded values. Move the collision response code out of the Teapot class and into a static Collision method that takes the two colliding objects as parameters. Write a static Collision method that checks for a collision between a sphere and a plane. By using this method, move the code that checks and responds to a collision between the teapot and the “walls “ of the world out of the Teapot class.