**Applied Physics - Practical 5**

**To be completed by January 18th**

**Purpose:** To extend practical 4 and allow a square to move **under gravity along the y-axis** (and bouncing off the ground) or move **under friction along the x-direction** .

**To Do:**

1. **Using the SFML starter code from the previous lab** create a new project called **Practical5** on your **G:\** drive in the folder **G:\Applied Physics\**.

Place a square on the ground instead of the circle using the RectangleShape SFML class.

1. When **the y-key is hit** by the user, move the square up the **y-axis** as if it was hit by an impulsive force that imparts a velocity of (0, **u**) m s-1 to the square in the y-direction only.

After the initial impulse, the square is to move under the **action of gravity only**, so that its position (and velocity) after each frame is given by the equations:

**Position = Position + Velocity\* time + 0.5\*acceleration\*(time)2**

**Velocity= Velocity + acceleration\* time**

(where time, is the time between frames, i.e. time since the last frame)

until it hits the ground again. Then it should bounce up again with a velocity of (0, **u\*m\_restitution**) m s-1 where m\_**restitution** is a value between 0 and 1 (0 means no bounce, 1 means it continues bouncing for ever with no loss of energy) and is called the **coefficient of restitution**.

The acceleration here is gravity only, **acceleration = (0, 9.81)**

**(20 marks)**

1. When **the x-key or z-key is hit**, move the square along **the positive x-axis** as if it was hit by an impulsive force that imparts a velocity of (**ux**, **0**) m s-1 to the square in that direction only.

After the initial impulse, the square is to move under the action of a frictional force only, so that its position (and velocity) after each frame is given by the equations:

**Position = Position + Velocity\* time + 0.5\*acceleration\*(time)2**

**Velocity = Velocity + acceleration\* time**

until it stops moving. The square should remain then at the point where it stops moving forward (do not allow it to return back again).

The acceleration here is friction along the direction of movement and according to physics the constant frictional force found from:

Force = ma = -mg (in the direction it is moving)

where g = the acceleration due to gravity and  = the coefficient of friction that depends on how rough the surface is. So in SFML dividing by mass use:

**acceleration = -coeffFriction\*g\*unitVelocity**

Note : unitVelocity is the normalized velocity vector. A normalized vector is the vector divided by its length. In the case of velocity vector , the length is sqrt(x\*x +y\*y), hence the normlised vector

x= x/ sqrt(x\*x +y\*y)

y=y/sqrt(x\*x+y\*y)

**Try  = 0.8** to start with and vary it around until you think it looks realistic.

**(50 Marks)**

Again output the maximum distance moved each time the program is run and also output the time taken until the object comes to rest. Check these values against what is predicted by physics for both the y- motion and the x-motion.

**(30 marks)**