**Lab 4 SFML**

**Deadline 17th Jan 2018**

**Create a ‘pleasing’ walking and jumping motion.**

**The player can move left and right when jumping also.**

**To Do:**

1. **Jumping:**

When the square is on the ground and **space is hit** by the user, move the square up the **y-axis** as if it was hit by an impulsive force that imparts adds 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**

The square stops when it’s hits the ground.

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

In this simulation each pixel is a meter. To enable to simulation to be more useful for games etc.

use a scaling variable called **const** **float pixelToMeter**

set **pixelToMeter=20.0;**

**gravity(0**, **9.81**\* **pixelToMeter);**

Experiment with different values of pixeltoMeter to get a ‘nice’ jumping effect.

**(10 marks)**

**2. Moving and jumping:**

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

When **d-key is hit**, move the square along **the positive x-axis** as if it was hit by an impulsive force that adds a velocity of (**ux**, **0**) m s-1 to the square in that direction only. After the impulse the square moves under the influence of gravity as above.

**(10 marks)**

**3. Walking**

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

When the **d-key is hit**, move the square along **the positive x-axis** as if it was hit by an impulsive force that adds 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

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 has a movement that would work well in a game environment.

**(10 Marks)**