Workshop 6 (Version 2)

Do question 2 first (it's easier).

Question 1:

Imagine that you are sitting in your spacecraft, and you discover that you are 1000 km from a black hole of mass 8×10^{30}kg (roughly 5 times the mass of the Sun, which is small for a black hole). What should you do? Will you die, or can you get into an orbit around it? We will write a Python program to simulate the motion of your spacecraft.

1.1: Setting the Scene

Start a VPython program, using either glowscript or Spyder (glowscript is probably easier).

Create a spherical object to represent the event horizon of the black hole – the line of no return, inside which you would need to go faster than light to escape. The event horizon for a black hole of this mass should have a radius of 11km. Put this object at the origin (i.e. (0,0,0)). Create a spacecraft as a coloured sphere, located 1000 km away along the x-axis (i.e. (1000000, 0, 0)). Make it big enough to easily see – i.e. 10 km across.

The black hole can destroy your spacecraft even if it is outside the event horizon. The tidal forces will tear your spacecraft to pieces if it is closer than about 300 km to the black hole. Create a semi-transparent red sphere of radius 300km to mark the danger zone. Give your sphere the property "opacity=0.5" to make it semi-transparent.

Run your code to make sure this is all working, and fix any bugs.

1.2: Make your Spacecraft Move

Now make your spacecraft move. Give it a starting velocity of 10,000 km/s along the positive y axis. Give your spacecraft a property called "vel" with a command like "vel=vector(0,10000000,0)" inside the list of properties of yoru spacecraft.
Now add a "while" loop to your program, to move your spacecraft in a straight line for a few simulated seconds. Pick a time-step dt=0.0001 and a starting time t. In your loop, update your time ("t = t + dt"), and use the velocity to update the position of your spacecraft ("ship.pos = ship.pos + ship.vel*dt"). Check that your code runs as expected. Fix any bugs.

1.3: Adding Gravity

To add gravity, we need to come up with a vector form of Newton's gravity equation

$$F = \frac{GMm}{r^2}$$

Convert to vector form by replacing r with $|\vec{r}|$, where $\vec{r}$ is the vector pointing from the black hole to the spacecraft (i.e. it is equal to the spacecraft's position, as the black hole is at (0,0,0)). Multiply the force by $-\hat{r}$ to make it point back towards the black hole.

Up near the top of your code, define the constant G and the mass M of the black hole. Now inside your loop, calculate the acceleration using your vector equation. The magnitude of the vector can be calculated as mag(ship.pos), if you called your spacecraft "ship". The unit vector is norm(ship.pos).

Now use the acceleration to update the ship's velocity, and the velocity to update the position. Everything will happen very fast, so you need to set a very small time-step, such as dt = 0.0001. Run your code and make sure it is working as expected.

1.4: Escaping the Black Hole

Play around with your starting speed see what the minimum speed and best direction is to keep you out of the danger zone around the black hole.

2: Modelling Wind

Write a VPython program to model the motion of a beach ball (mass 300g, radius 50cm) on a windy day. The ball should be launched from the ground at an angle of sixty degrees to the horizontal at a velocity of 10 m/s. But it is a windy day – there is a steady 9m/s wind blowing horizontally in the opposite direction, which will blow the ball backwards.

2.1: Set the scene, with a flat green box for the ground and the ball resting on top of it. Run your code and make sure it is working.

2.2: Now give your ball a velocity. Define its starting velocity V and angle theta (angle to the positive x-axis). Note that angles are in radians, so 60 degrees would be written V = 60*pi/180. Then add "vel=vector(V*cos(theta), V*sin(theta), 0)" to your spacecraft's properties. Make sure you define V and theta *before* you set up your ball.

2.3: Now use the velocity to move the ball. Pick a time-step dt and a starting time t. In your loop, update your time ("t = t + dt"), and use the velocity to update the position of your ball ("ball.pos = ball.pos + ball.vel*dt"). Make sure everything is working as expected.

2.4: Now add gravity. Define a gravity acceleration vector ("g = vector(0,-9.8,0)"). Inside the loop, update the ball's velocity ("ball.vel = ball.vel + g*dt"), then update the position as before. Run and make sure it's working sensibly.

2.5: Now add drag. Write down a vector form of the drag equation $F = \frac{1}{2} C\ A\ \rho v^2$ by replacing the "v" with the magnitude of the velocity, and multiplying by a suitable unit vector to make the force point in the right direction. If your ball's velocity is ball.vel, the magnitude of this is mag(ball.vel) and a unit vector pointing in the same direction as the velocity is norm(ball.vel). Add the acceleration caused by drag to the line that updates your ball's velocity. Run and check. You may assume a drag coefficient of 0.7 and a density of air of 1.1 kg m^{-3}.

2.6: Now add wind. The drag force depends on the velocity of the ball *relative to the air*. You need to define a wind vector, using a command like "vwind = vector(-9,0,0)" and then, inside your loop, calculate the relative velocity "vrel = ball.vel – vwind". Then use vrel to calculate the drag force. Check that it behaves sensibly – the ball should be blown backwards.

3: Time Step Accuracy (Extension)

Going back to Question 1, we'll investigate how the accuracy of this numerical simulation depends on the time-step used.

Calculate the starting velocity needed to put your spacecraft in a perfectly circular orbit around the black hole. Give the spacecraft this velocity and simulate one orbit around the black hole. Print out the ratio of the final energy of the spacecraft divided by the initial energy. If the simulation was perfect this ratio should be 1.

Repeat this numerical "experiment" for at least ten different time-step values, ranging from 10^{-4} up to 10^{-1} s. Plot a graph of ratio as a function of time-step used. Based on this, comment on what is a good time-step to use in this situation. Explain your reasoning.