PHAS1240: Final Assignment 2015

Version 0.99a — December 11, 2015 — Louise Dash

Contents

1. Introduction

For the 2015 PHAS1240 Final Assignment you will be extending some of techniques covered in sessions 8 and 9 to produce VPython code. Imagine you are in the following situation:

You have been hired as a consultant physicist-programmer for the upcoming Angry Birds sequel "Frozen Angry Birds Star Wars".

Your job is to produce a prototype of the game to give to the production department, who will deal with the artwork and the user interface—the physics of your code needs to be correct, but the graphics will be rudimentary.

Your predecessor in the role (who left for a better-paid job at Google) has provided some incomplete code that you can use as a starting point. The storyline department has provided the following, somewhat unlikely, scenario:

> Following an unfortunate lava-related incident, Olaf the Snowman has gone over to the Dark Side. Olaf is currently on the surface of his newly-built Death Snowball. An elite unit of Jedi Angry Birds has been despatched to defeat him. These Angry Birds explode on impact, and will destroy Olaf if their speed at the point of impact is sufficient. The evil Sith lord Darth Vader is most displeased with this development, and has teleported his Death Star into the region to support Olaf.
> The story continues ...

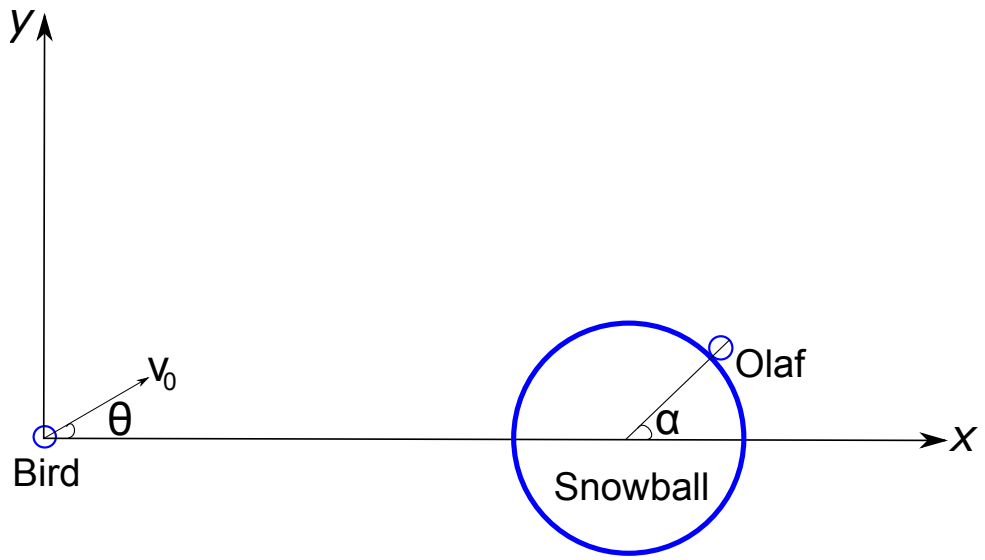

Figure 1: *A rendition of the situation just before the bird is launched with speed v_0 and angle θ*

2. The setup

Your code needs to follow the specifications of the brief exactly.

In this galaxy, all quantities are given in Galactic Natural Units (an arbitrary unit system in which, conveniently, the gravitational constant $G = 1$), and all the action takes place in the xy-plane. In the prototype, all the objects will be represented as spheres. You may include some additions or improvements to the graphics if you wish, but the physics should reflect the underlying sphere structures.

This setup is illustrated in Figure 1 and described in detail below. The initial parameters are mostly already set up in the template code, but please read the information below carefully.

2.0.1 The Bird

- The bird will be launched from the origin of the coordinate system at $(0, 0, 0)$ at an angle θ to the x-axis and initial speed v_0.
- The bird has a mass of 1 Galactic Mass Unit and a radius of 0.1 Galactic Distance Unit.
- *You do not need to program any form of user interface, these variables should be set within the code, as indicated in the template.*

2.0.2 Olaf and the Death Snowball

- Olaf's Death Snowball has a radius of 1 Galactic Distance Unit, and is initially located at rest at coordinates $(7, 0, 0)$.

- Olaf is on the surface of the Death Snowball, located at an angle of $\alpha = 45°$ above the positive x-axis relative to the centre of the Death Snowball (as shown in Figure 1). You will need to add code to the template to calculate the x and y-coordinates of Olaf's position.
- Olaf (in his sphere representation) has a radius of 0.1 Galactic Distance Units.
- The combined mass of Olaf and his Snowball is 2000 Galactic Mass Units. You can treat Olaf and the Snowball as a single point mass from the point of view of implementing the mechanics.

2.0.3 The Death Star

- At the moment of the bird's launch, Darth Vader's Death Star teleports into the region at coordinates $(10, -2, 0)$ and velocity $(0, 15, 0)$.
- The Death Star has a mass of 2000 Galactic Mass Units.
- The Death Star is fitted with a tractor beam that utilises the Force to counteract all gravitational forces between the Death Star and the Death Snowball;
- however it does interact gravitationally with the Angry Bird.

All your code should continue to work as intended when any of the initial parameters given above are changed.

In particular, the production team has asked you to allow for the possibility of a super-dense Neutron Bird, with a mass of up to 1000 Galactic Mass Units, and you need to take this into account when calculating the gravitational forces.

3. Strategy

The gravitational forces should be calculated as in Session 9, using Newton's law of universal gravitation:

$$\mathbf{F}_{12} = -Gm_1m_2\frac{\mathbf{r}_{12}}{|\mathbf{r}_{12}|^3}, \tag{1}$$

to calculate the force $\mathbf{F}_{12}$ on the object with mass m_1 due to the object with mass m_2. You can use your completed Session 9 code as a starting point.

To summarise, your code *should* include the following gravitational forces:

- between the bird and the Olaf/Death Snowball system;
- between the bird and the Death Star.

Your code does *not* need to calculate the following gravitational forces:

- between the Death Star and the Olaf/Death Snowball system (this is cancelled out by the Force generated by the tractor beam on the Death Star).

You will need to find a way of detecting collisions between the bird and the other objects. For this, consider the equation of a circle of radius r centred on a point (x_0, y_0)—a point (x, y) will be within the circle if the following condition holds:

$$(x - x_0)^2 + (y - y_0)^2 \leq r^2. \tag{2}$$

If your timestep is well-chosen, you can use this to check if the bird has collided with the other objects—make sure to include any assumptions you make in these calculations in the comments of your code.

You are strongly recommended to start with a simplified version of the code, with just the bird, Olaf and the Death Snowball (i.e. no Death Star). Once you are confident that all is well with this simplified model, add in the moving Death Star.

Make sure that your code is clearly structured and includes concise, relevant comments that will allow the marker to easily decipher what your code is doing.

One Python statement that you may find useful, but which we haven't explicitly covered in PHAS1240 so far, is the `break` statement:
https://docs.python.org/2/reference/simple_stmts.html#break

Your code should output the following information to the console:

- Whether the bird collides with one of the other objects in the system (and if so which one), or whether it misses everything completely
- If the bird collides with Olaf, the speed of impact should be output to the console.

You can, if you wish, extend your code beyond the given scenario—however if you do this you *must* make sure your code stays consistent with the requirements specified above.

4. Submission and assessment

Before you submit your code, make sure that you have tested it with several variations of the initial parameters given in the template.

The submitted version of your code should include parameters that result in a successful collision between the bird and Olaf.

As for the Longer Task assignment, this assignment will be graded anonymously. Please make sure that your name does not appear anywhere in your submission. Make sure that the comment block at the beginning of your code contains your student number, and that your submission title starts with your student number.

4.1 How your code will be marked

We will first run your code exactly as you have submitted it.

However, we will also be replacing the header section of the code (as clearly marked in the template) with different header sections that set a variety of different parameters. For example, the masses, velocities, and positions of the objects in the system will be varied. This will enable us to check that the physics of your code is working as intended.

For this reason, ***do not change any of the variable names that are set in the header section of the code***, and do not add any lines of code without which your code will not run to this header section.

4.2 Assessment criteria

The assessment for your final assignment will be based on the following criteria:

- Successful implementation of the physics (40%)
- The structure, style and commenting of your code (40%)
- Whether you have followed the specifications of the brief (10%)

The final 10% of the marks will be reserved for showing imagination and going beyond the basic requirements given in the script, while still remaining consistent with the specification—for example, improvements to the basic graphics.