PHAS0007: Final assignment 2018/19

Version 1.2a — December 6, 2018 — Louise Dash

Contents

1. Introduction

You will take the programming skills you have developed throughout the course and use them, combined with your knowledge of classical mechanics, to code a game with mechanics similar to *"Angry Birds"*: The user will launch a projectile at a target to try to topple it. In order to do this, you will need to calculate:

- the path of the projectile (which you have already done in session 8);
- the momentum of the projectile at the point of impact with the target;
- the impulse transferred to the target
- the torque on the target due to the collision, and whether or not this is greater than the restoring torque.

The physics you need is included in detail in Section 2. An outline algorithm is provided in Section 3.1, and programming strategies and hints in Section 4.

The vast majority of the marks for this assignment are for correctly coding the *physics*, although some marks are reserved for visual impact. For this assignment you should produce a Jupyter notebook using the specification given in Section 3.

2. The Physics

2.1 Projectile motion

The path of a ballistic projectile, ignoring the effects of air resistance, can be expressed fully by two starting parameters: the initial speed v_0 and the angle of launch θ (note that this is equivalent to specifying the launch velocity $\mathbf{v_0}$).

The position $\mathbf{r}$ of the projectile at any time t is then given by:

$$r_x = x_0 + v_0 t \cos(\theta) \tag{1}$$

$$r_y = y_0 + v_0 t \sin(\theta) - \frac{gt^2}{2} \tag{2}$$

Since momentum is defined as

$$\mathbf{p} = m\frac{\mathrm{d}\mathbf{r}}{\mathrm{dt}}, \tag{3}$$

we can express the momentum of the projectile as

$$p_x = m v_0 \cos(\theta) \tag{4}$$

$$p_y = m v_0 \sin(\theta) - mgt \tag{5}$$

2.2 Collisions

If the ball hits the target, we need to decide how to treat the collision. In the interest of simplicity, we make the following assumptions:

- The collision between the ball and the target is completely elastic, i.e. no energy is lost.
- The ball comes completely to rest, i.e. it transfers all its momentum to the target.
- The contact time between the ball and the target is finite, and is the same for all collisions. It can thus be represented as a fixed parameter $\Delta t_{collision}$.

We can use this to calculate the force applied to the target as a result of the collision, and calculate if it is enough to topple the target.

2.3 Toppling the target

In order to tip the target over, we need to apply a torque sufficient to rotate the target clockwise (assuming the projectile is arriving from the left) around its bottom right corner, overcoming the restoring torque which operates through the centre of mass, as shown in Figure 1.

Again, we need to make some assumptions, as follows:

- The target cannot slide on the ground;

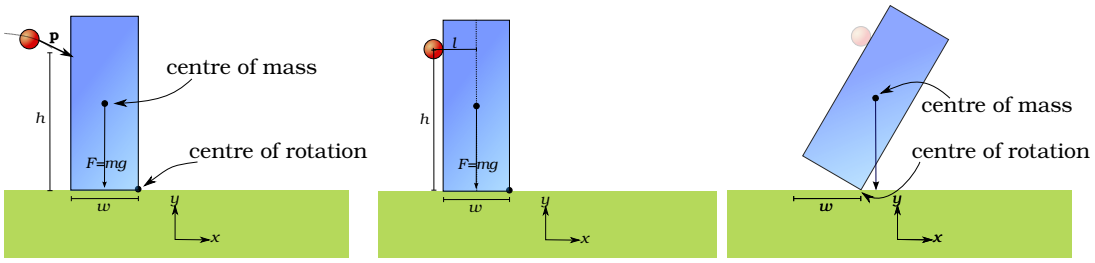

Figure 1: *Target before (left), during (centre), and after (right) the collision. The bird impacts the left side of the target with momentum* $\mathbf{p}$ *at a height* h *above the ground, with a horizontal distance* l *between the centre of the bird and the central axis of the target. The force applied to the target will provide a torque around the centre of rotation at the bottom right of the target. After the collision, if the applied torque is greater than the restoring torque (operating through the centre of mass), the target will topple.*

- The target has constant density, so that its centre of mass is at the geometric centre of the target.

First of all, consider the torque which will try to restore the target to its normal, stable upright position:

$$\boldsymbol{\tau}_{\text{restoring}} = \mathbf{F}_{\text{grav}} \times \mathbf{d}_r, \tag{6}$$

where $\mathbf{F}_{\text{grav}} = m\mathbf{g}$ is the force provided by gravity, operating through the centre of mass, and $\mathbf{d}_r = w/2$ is the *horizontal* distance between the point of rotation (i.e. the bottom right corner) and the centre of mass.

We are interested in the magnitude of this torque:

$$|\boldsymbol{\tau}_{\text{restoring}}| = -mg\frac{w}{2}\sin(\phi) \tag{7}$$

$$= -mg\frac{w}{2}, \tag{8}$$

as here the angle between the force and position vectors ϕ is 90°.

Now consider the torque supplied by the collision:

$$\boldsymbol{\tau}_{\text{applied}} = \mathbf{F}_{\text{applied}} \times \mathbf{d}_a, \tag{9}$$

where now $\mathbf{d}_a$ is the vector from the point of rotation (bottom right-hand corner of target) to the point of impact, on the left-hand side of the target at height h, as in Figure (1).

What is the applied force $\mathbf{F}_{\text{applied}}$? For this, we will need to use our assumptions about the collision, and use the definition of impulse

$$\mathbf{I} = \mathbf{F}_{\text{applied}}\Delta t = \Delta\mathbf{p}, \tag{10}$$

which we rearrange to give

$$\mathbf{F}_{\text{applied}} = \frac{\Delta\mathbf{p}}{\Delta t}. \tag{11}$$

Since we have assumed that *all* the ball's momentum is transferred to the target, $\Delta\mathbf{p}$ will be equal to the momentum of the ball at impact, i.e.

$$\mathbf{F}_{\text{applied}} = \frac{\mathbf{p}_{\text{ball}}}{\Delta t}. \tag{12}$$

The target will topple if

$$|\boldsymbol{\tau}_{\text{applied}}| > |\boldsymbol{\tau}_{\text{restoring}}|. \tag{13}$$

3. Specification

Your code *must* meet the following requirements. You will lose marks if you fail to follow the specification exactly.

3.1 Structure:

Your code should follow the basic outline algorithm for the game shown in the flowchart of Figure (2). You will want to plan the structure of your code carefully, and decide how best to implement this using loops, conditionals, and user functions.

The code should continue running until the user has successfully toppled the target.

3.2 Setup:

1. Your code must run on a normal Python 3 kernel (not a VPython kernel).
2. You will need to import Numpy, and the following functions from vpython:

   ```
   sphere, color, rate, canvas, vector, curve, label,
   box, cross, mag, random, arrow
   ```

   You may also import other vpython functions as necessary, but you *may not import any other modules*, i.e. you are restricted to base Python, Numpy and VPython. Your code should also not depend on any external files—you will only be able to upload one file.
3. Use the following commands to set up the vpython canvas:

   ```
   scene = canvas(width=640, height=480, center=vector(8,5,0),range=8)
   ground = curve(pos=[(0,0,0),(16,0,0)],color=color.green)
   ```

   This will give you a canvas where the origin is on the left hand side, and the maximum value of $x = 16$ at the right hand side, with the "ground" near the bottom of the canvas. You will be working in units where one screen unit is equal to one metre - i.e. the ground is 16m wide. You can change the visual properties of the "ground" object, but it should remain 16m wide.
4. The bird/projectile will be launched from a platform located at $x = 0$. The vpython function `random()` will create a random number between 0 and 1. Use this function to generate a random height for the platform between 0 and 1 m. Choose any sensible vpython object(s) to represent your platform on the canvas.
5. Use the vpython `box()` object to create an object with the variable name `target`. The target should be 2m tall and 0.5m wide and deep. Use the `random()` function again to place the target on the ground at a random x-position between 5 and 15 metres from the origin. The mass of the target should be set to 100 kg.
6. The bird/projectile should have a mass of 0.1 kg and should be launched from the random-height platform at $x = 0$. For the purposes of the calculation of the mechanics and detecting a collision (see below) assume the bird has a radius of 5cm.

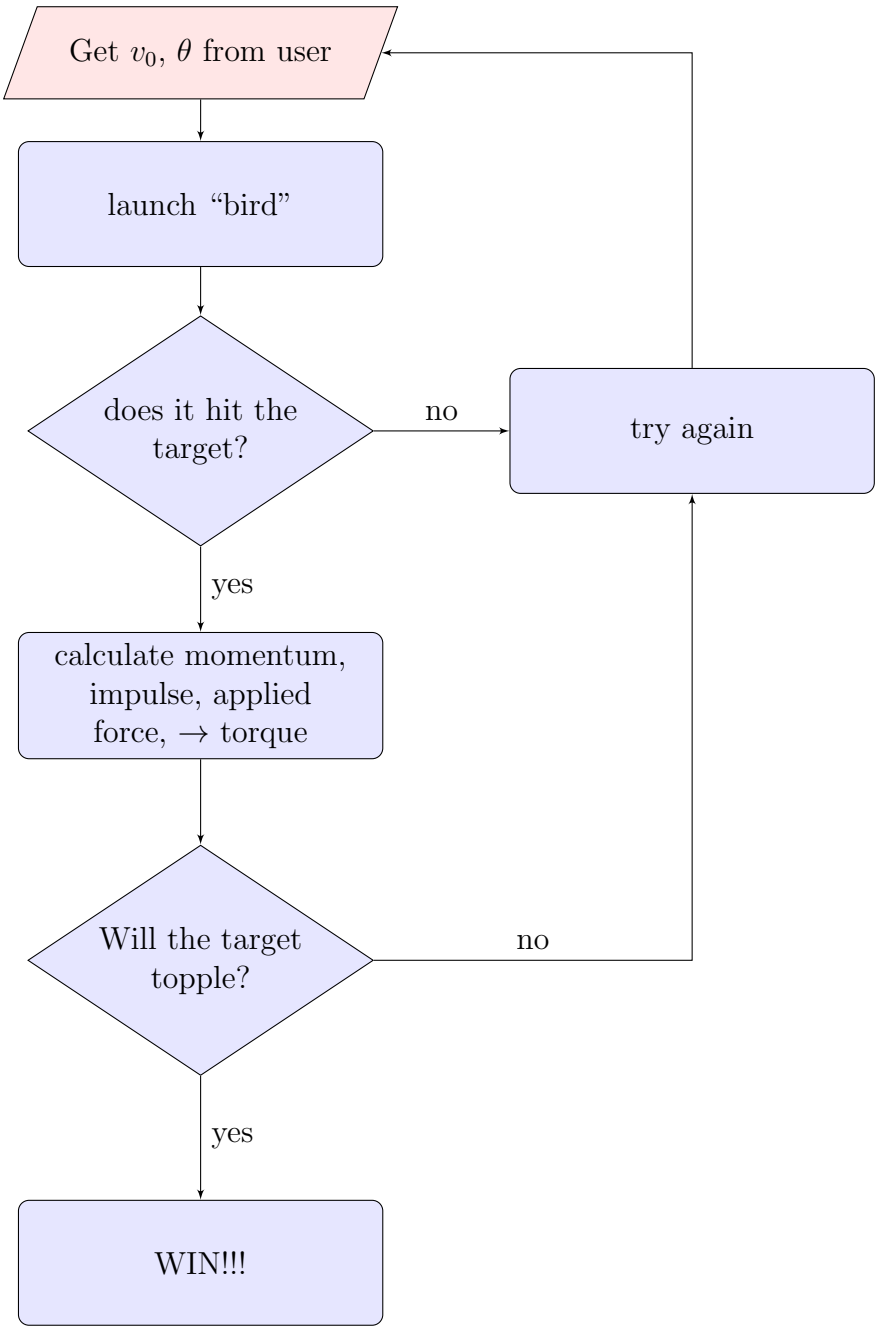

Figure 2: *Outline algorithm for the game*

For the purposes of the animation only, it is recommended that you use a vpython sphere with a radius of 0.3. You may, if you wish, choose a more complex bird-object for the animation once you have everything working.

7. Use an `arrow` object to represent the momentum $\mathbf{p}$ of the bird at each step of the animation.
8. In order to decide whether or not a collision has occurred between the bird and the target, you will need to consider the following:
   - The horizontal distance between the centre of the bird and the vertical axis of the target: is this less than the distance l in Fig. 1 (centre)?
   - The vertical distance between the centre of the bird and the top of the target: is this less than the physical radius of the bird (5cm)?
9. Use the following values:
   - 9.81 ms^{-2} for the acceleration due to gravity.
   - 0.01 seconds for the contact time Δt when implementing equation (12).
   - 0.001 seconds for the animation timestep.
10. Your code should ask the user to input the launch angle and speed each time the bird is launched, and keep the user informed of their progress via `label()` objects on the canvas.
11. In addition to the `label()` status messages to the user, your code should output the following quantities to the console in the event of the bird hitting the target, using appropriately formatted `print()` statements:
    - The height of the impact point (the distance h in Fig. 1);
    - The bird's momentum at the point of impact;
    - The applied torque (Eq. (9))
    - The magnitude of the restoring torque (Eq. (7))

3.3 Text cells

For this assignment, your notebook is not required to be entirely self-contained in terms of the physics (as the Reading Week task was), and you do not need to restate all of the physics in the script.

However, as a minimum, you should include the following as text cells:

- A clear introduction, stating your aims and strategy
- A discussion at the end of your notebook detailing:
  - How you would suggest improving the representation of the physics in the game
  - How you might go about implementing these improvements in Python.

- Other text cells as necessary (in addition to the code comments) to make it clear to the person marking your assignment exactly what you are doing in terms of implementing the physics, and why you've chosen to do it in that way.

Your text cells should be structured in complete, grammatically correct, coherent sentences and paragraphs.

4. Strategies and hints

You are recommended to break the problem down into sections as in the flowchart. Each section of the code will be marked separately, so if you are having severe problems, concentrate on getting one or more parts correct rather than attempting everything.

Make sure you explain all choices you make in your implementation, especially in the comments. If you use an unusual coding approach, you *must* explain it clearly in the comments, and state why you have chosen this approach, otherwise you may not receive full credit for it. As usual, you are recommended to comment your code as you go along, as this forms an essential part of the debugging procedure. You may also find it useful to enable line numbering in your code cells (press ESC-L in a code cell).

You are not required to *animate* the toppling of the target—a message to the user with a `label()` object is sufficient. You are however welcome to try this animation for bonus points if you have the time and inclination. Similarly, you are not required to provide a user interface beyond that provided by the console—the user can enter the initial angle and speed of the bird in exactly the same way as in the Session 8 task.

Think carefully about how you will use nested `if` and `while` structures in this code. Plan these well, as it is easy for the unwary to get in a mess. You will also find it useful to code functions for repeated tasks.

Much use is made of vectors in this task. You will be familiar with `vector` and `mag` from sessions 7, 8, and 9, but you will also find the command `cross` (to calculate the cross product of a vector) useful when calculating the torques.

You are strongly encouraged to make full use of the Jupyter VPython documentation available at http://www.glowscript.org/docs/VPythonDocs/index.html.

Before you submit your code, make sure that you have tested it fully for all outcomes.

5. Assessment

This is an individual assignment. You must work on this entirely on your own, without the assistance of other students, course staff or anyone else. You should make sure you are fully familiar with the UCL guidelines on avoiding plagiarism before you start this assessment.

Everything in your submission must be either completely your own work, or have the source explicitly acknowledged and suitably referenced. We will be using several methods to check for any plagiarised or copied assignments.

As for the Reading Week assignment, this assignment will be graded anonymously. Please make sure that:

- your name does not appear anywhere in your submission;
- your submission title starts with your student number; and
- you include your student number at the start of your submitted Jupyter Notebook.

Your assignment will be assessed using the following criteria:

- **Physics:** How well you have implemented the physics (e.g. the equations of motion, mechanics of the collision, torque, etc) (40%).
- **Structure:** How well your code is structured and how effectively you have implemented the basic algorithm of Fig. 2 using loops, conditionals and functions (12%).
- **Code style and comments:** General coding style and efficiency (12%), and the clarity of the documentation provided by your code comments (12%).
- **Specification:** Whether you have correctly followed the specification given in Section 3 (10%).
- **Text cells:** The quality of the text cells, as specified in Section 3.3 (8%).

The remaining 6% of the marks are reserved for any improvements to the visual impact of the animation and user interface that, while remaining consistent with the requirements of the specification given in Section 3.2, go beyond it.