

# Lab 2 - VPython, simple mechanics

Thursday 6 September - Due: Thursday 13 September

In this lab we will experiment with solving some simple mechanics problems using Python/VPython.

## 1 Simple Vpython codes

### 1.1 Harmonic oscillations in 2D

1. Open up a file window (see earlier).
2. Go to the labs link from the PHY307 homepage (<http://www.phy.syr.edu/courses/PHY307>) and click on the two links to the codes `euler.py` and `force.py` discussed in class.
3. Cut and paste these codes into *separate* Python file windows. Save these codes to your filespace.
4. Run this program by selecting **Run Module** from the **Run** menu at the top of the `euler.py` file window. Record what you see in your lab report. This motion corresponds to the 2D harmonic oscillation we discussed in class.
5. Now we will add some code to draw the track of the ball.
  - Replace the line starting with `ball=sphere(...` with `ball=sphere(radius=0.5,track=curve(radius=0.1))`  
The variable `track` is the name I am using here for an object of type `curve` which will contain the path of the ball. `curve` is a standard Vpython object with many attributes of which the most important perhaps are its color and its radius (width).
  - Add the following line of code after the line `ball.vel=...`  
`ball.track.append(pos=ball.pos)`  
This is the python command to append the latest position of the ball to the track object. *Make sure it is indented at the same level as the position and velocity update lines.*
6. Now rerun the code. Record what you see. What is the shape of the orbit ? You may want to cut and paste the graphics into your lab report.
7. Also, include your modified code in your lab report in the following way. Select the whole program by finding **Select all** on the **Edit** menu
8. Copy your selection using **Copy** under **Edit**
9. Paste your selection into your report (**Paste** under the **Edit** menu in Word)

10. Experiment with changing the color of the ball and its track by using the `color` attributes of `sphere` and `curve`. For example, to change the color of the ball just write
 

```
ball=sphere(radius=0.5,color=color.red,track=...)
```
11. Again copy your modified code into your lab report.

## 1.2 Projectiles

1. Open up the `force.py` code in a python file window. We will modify this code to use it to describe 2D projectile motion. Lets make sure we save the old version of `force.py` by selecting **Save As** from the **File** menu and saving the code with a new name - say, `force_projectile.py`.
2. Modify this force function to handle a *constant* downward (y direction) force of magnitude  $-1$ . Careful: what should the x-component of the force be ? This models gravity near the Earth's surface (in some funny units). Also, add the following line to the bottom the `euler.py` program. The `if` *must* preserve the previous level of indentation so that it is run every time through the while loop

```
if(ball.pos.y<0):
    run=0
```

3. What is the purpose of this python statement in terms of physics. Hint: what is the y-coordinate of the ground ?
4. Also change the condition `while(1):` to `while(run):` at top of loop. You will need to set `run=1` somewhere after you create the `ball` object also. This `if` statement is a standard Python construct and allows us here to break out of the infinite update loop if a suitable logical condition is satisfied. In this case the variable `run` is a logical (sometimes called Boolean) variable which can take the values either 1 (true) or false (0). Notice also that the attribute `pos` of the `ball` object is accessed with the code `ball.pos` and that the y-component is further accessed with another dot operator i.e `ball.pos.y`.
5. Finally replace the statement `from force import *` by the new one `from force_projectile import *`. Set the initial coordinates to  $(-9,0,0)$  and the initial velocity to  $(3,3,0)$
6. Run the code and make a screen capture of the result.

### 1.3 Bouncing

1. Modify the program `euler.py` so that the projectile bounces when it hits the ground. The simplest way to do this is to flip the sign of the y-component on the velocity once the projectile hits the ground (i.e `if ball.pos.y < 0`). You will need to change the value of `scene.range` to see the new motion. Also the `while` command in `euler.py` needs to be modified so that the loop continues indefinitely again.