VP1: Simple Start for Vpython, Projectile [the basic of Python, while]

I. Follow the steps in “安裝” at http://tcjd71.wixsite.com/vpython/install to install either (1) python3 + vpython7, or (2) anaconda + vpython (http://tcjd71.wixsite.com/vpython/copy-of-python-2-vpython-6), or both(1) + (2).

Video: https://goo.gl/6F8uZp

II. Free Fall:

Type (type instead of “cut-and-paste” can help you learn coding) and then run the codes. *** Hold the right mouse button and move mouse to change view angle. Hold both buttons and move the mouse to zoom in or out.

```
from vpython import *
g=9.8              # g = 9.8 m/s^2
size = 0.25        # ball radius = 0.25 m
height = 15.0      # ball center initial height  = 15 m

scene = canvas(width=800, height=800, center =vec(0,height/2,0), background=vec(0.5,0.5,0))    # open a window
floor = box(length=30, height=0.01, width=10, color=color.blue)                                # the floor
ball = sphere(radius = size,  color=color.red, make_trail = True, trail_radius = 0.05)          # the ball
msg =text(text = 'Free Fall', pos = vec(-10, 10, 0))

ball.pos = vec( 0, height, 0)          # ball center initial position
ball.v = vec(0, 0 , 0)                 # ball initial velocity

dt = 0.001                             # time step
while ball.pos.y >= size:              # until the ball hit the ground
   rate(1000)                          # run 1000 times per real second

   ball.pos = ball.pos + ball.v*dt
   ball.v.y = ball.v.y - g*dt

msg.visible = False
msg =text(text = str(ball.v.y), pos = vec(-10, 10, 0))
print(ball.v.y)
```

Code explanation:

1. *** Declaring using VPython module. All our simulation programs will have this first line

```
from vpython import *
```

2. Setting constants. For convenience, all physical quantities in the simulation world are always with SI units. *** Texts after # are not parts of the program, they are remarks.

```
g=9.8              # g = 9.8 m/s^2
size = 0.25        # ball radius = 0.25 m
height = 15.0      # ball center initial height  = 15 m
```

3. Opening a window.

```
scene = canvas(width=800, height=800, center = vec(0,height/2,0), background=vec(0.5,0.5,0))
```

*** Open a window named scene with 800 horizontal pixels and 800 vertical pixels. In the simulation world, before changing the view angle, +x axis points to the right is, +y to the top, +z pointing out the screen. center is the position vector of the center of the simulation world. vec(x, y, z) means a 3D vector.

*** background sets the background color to vec(red, green, blue), which indicates the strength for red, green, and blue, respectively, scaled from 0.0 to 1.0,. **Always set this attribute to some background color, otherwise the background defaults to black, making results difficult to see, especially with a projector.**

4. Objects in simulation world.

floor = box(length=30, height=0.01, width=10, color=color.blue)   # the floor

This draws a box of length = 30 (in x), height = 0.01(in y), and width = 10 (in z) called floor. You may use floor.pos to assign its center, e.g. floor.pos = vec(1,0,0). Without this, the center defaults at vec(0,0,0).
*** In Python, A.B means the "attribute B of A". The color of the floor is blue.

*** vector() or vec() is used to present a vector, such as a=vector(1, 2, 3), in which all three components are float (i.e here 1 is 1.0,…). More, a.x means the x component of a. We can use print(a.x) to show the x component of vector a or a.x = 5 to set the x component of a to 5. floor.pos is also a vector, therefore floor.pos.x is the x component of floor.pos. Similarly, for y and z.

ball = sphere(radius = size, color=color.red, make_trail = True, trail_radius = 0.05)

This draws a sphere called ball, with radius = size and color=color.red. Later, we may assign the center position of the ball, such as ball.pos = vector(1, 0, 0), and we can also attach more attributes to ball, such as ball.v = vector(2, 0, 0). Attribute make_trail = True makes a trail of the object. The thickness of the trail is set by trail_radius.

msg =text(text = 'Free Fall', pos = vec(-10, 10, 0))

This code shows a message with a text content 'Free Fall' at pos = vec(-10, 10, 0).

5. Start the simulation

```
ball.pos = vec( 0, height, 0)          # ball center initial position
ball.v = vec(0, 0 , 0)                 # ball initial velocity
```

These two set the initial conditions.

dt = 0.001

dt sets how much real time elapses in one step in the following while loop. The size of dt depends on the time scale of the simulation events. Too small, the simulation takes too long. Too large, the simulation will be too rough and cause incorrect results. For free fall, an event of several seconds, dt = 0.001 is just fine. For atom collision events in 10^{-11} seconds, dt should be 10^{-14}. For Earth to circle around the sun it takes about 10^{7} seconds, then dt = 10^{3} is fine.

while ball.pos.y >= size:

*** We use the "while loop" command all the time. The condition between while and colon(:) is tested. If it is satisfied, all **the indented codes (associated codes)** below colon are executed once. Then the condition will be retested again and the process will repeat until the condition is no longer satisfied (here, it means that the y component of the ball's center position is no longer larger than the ball radius, meaning the ball touches the floor). At this moment, Vpython stops executing the while loop and its associated codes, but then to continue to the next section of the codes (here, it is msg.visible = False)
*** In Python, **indentation of a section of codes** (you can do this by press tab key) means this section of codes is associated with the previous line of code with colon (:).

   rate(1000)

   *** This sets the while loop to run 1000 times per real-world second. With dt=0.001, this simulation runs at a speed of 1000*0.001 = 1 of real-world time, meaning the result is presented as in real-world time. If rate(500), 500*0.001 = 0.5, then the result is presented at a slow motion of 0.5 real-world time.

```
    ball.pos = ball.pos + ball.v*dt        Let ball.pos to increase ball.v*dt in one dt
    ball.v.y = ball.v.y - g*dt             Let ball.v.y to increase -g*dt in one dt.
```

These two lines are the most basic to describe kinetics of moving bodies

```
msg.visible = False
msg =text(text = str(ball.v.y), pos = vec(-10, 10, 0))
```

After the while loop stop running due to the unsatisfactory condition, the next two lines of code make the previous message text “Free Fall” invisible and then show at the same position the y component of the ball’s velocity. Here, str() transforms a number to a string text. For example, str(5.5) gives you a text string ='5.5' You can also print this value on the shell screen by

```
print(ball.v.y)
```

III. Arrow:

```
from vpython import *

scene = canvas(width=800, height=800, background=vec(0.5,0.5,0))                    # open a window
a1 = arrow(color = color.green, shaftwidth = 0.05)
b1 = arrow(color = color.blue, shaftwidth = 0.05)

a1.pos = vec(1, 1, 0)
a1.axis = vec(1, -1, 0)
b1.pos = a1.pos + a1.axis
b1.axis = vec(2, 1, 0)

c1 = arrow(color = color.yellow, shaftwidth=0.05)
c1.pos = a1.pos
c1.axis = a1.axis + b1.axis
```

In this program, it draws two arrows, a1 and b1, with color being green and blue, respectively and with shaftwidth = 0.05. arrow has attributes like pos, axis, and color. E.g. a1.pos = vector(1, 1, 0) makes the starting point of a1 at (1, 1, 0), a1.axis = vec(1, -1, 0) draws a1 as a vector of (1, -1, 0). Similarly, b1 starts at a1’s arrow tip and has an axis of vector vec(2, 1, 0). If you follow the codes for arrow c1, you can find that this is actually a representation of a vector addition, vector a1 + vector b1 = vector c1.

IV Homework:

You need to hand in your homework with a filename extension ‘.py’. If you are writing your homework in Jupyter, you need to extract the complete runnable codes and save them in just one ‘.py’ file.

(must) Modify the projectile program. Let the ball’s initial position = vec(-15, 5, 0) and with initial velocity = vec (6, 8, 0). Add at the center of the ball an arrow, which moves along with the ball and whose length is proportional (proportional constant by your choice) to and parallel to the velocity vector of the ball. In the end, show the displacement of the ball for the entire flight.

(optional) Add some codes to find the following values and show the values in the end.

1. The flying time in the air.
2. The length of the entire path. (Note: in python, x**p means x to power p, x^p)