CW1: Simple Start for Vpython, Projectile [basic of Python, while]

Notice: text relating to describing VPython feature for the first time is marked with "***"

I. Install python and vpython by "安裝" in http://VPhysics.ntu.edu.tw or http://tcjd71.wix.com/vpython. Run "VIDLE for Python". Type in the following:

```
print 1+1
print 1/2
print 1.0/2.0
print 'Hello world'
```

***Press F5 to run.

***In Python, 1, 2, ... are **integer type**. Thus, (1/2) is of integer type and is equal to 0. You can use 1.0 / 2.0 (called **float**) to avoid this problem.

II. "Free fall" simulation. Type (typing instead of cut-and-pasting can facilitate your learning to code greatly) the codes in your VIDLE, then press F5 to run. *** 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 visual 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 = display(width=800, height=800, center = (0,height/2,0), background=(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)                                            # the ball

ball.pos = vector( 0, height, 0)          # ball center initial position
ball.v = vector(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.v*dt
   ball.v.y +=  - g*dt

print 'end'
```

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

```
from visual 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 for remark.

```
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. Openning a window.

```
scene = display(width=800, height=800, center = (0,height/2,0), background=(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 is 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.

*** background sets the background color to (0.5, 0.5, 0), which indicates the strength for red, green, and blue, respectively, scaled from 0.0 to 1.0,. **Always set this attribute, otherwise the background defaults to black and is difficult to see, especially on a projector.**

4. Objects in simulation world.

```
floor = box(length=30, height=0.01, width=4, 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 = vector(1,0,0). Without this, the center defaults at (0,0,0).
*** In Python, A.B means the "attribute B of A".

*** vector() 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)                                    # the ball
```

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).

5. Beginning the simulation

```
ball.pos = vector( 0, height, 0)          # ball center initial position
ball.v = vector(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 not be correct. For free fall, an event of several seconds, dt = 0.001 is good. For atom collision in 10^{-11} seconds, dt should be 10^{-14}. For Earth to circle around the sun it takes about 10^{7} seconds, dt = 10^{3} is fine.

```
while ball.pos.y >= size:
```

*** We use this "loop" command all the time. The condition between while and colon(:) is tested. If it is satisfied, all **the indented codes** below colon are executed once. Then the condition will be tested again and the process will repeat until the condition is no longer satisfied (here, it means the y component of the ball's center position is no longer larger than the ball radius, meaning the ball touches the ground). At this moment, Vpython stops executing the while loop and its associated codes, but then continue to the next section of the codes (here, it is to print 'end').

*** In Python, **indentation of a section of codes** (you can do this by press one tab) 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 second. With dt=0.001, this simulation runs at a speed of 1000*0.001 = 1 of real time, meaning the result is presented at real time. If rate(500), 500*0.001 = 0.5, then the result is presented at a slow motion of 0.5 real time.

```
    ball.pos += ball.v*dt        Let ball.pos to increase ball.v*dt in dt
    ball.v.y +=  - g*dt          Let ball.v.y to increase -g*dt in dt.
                                 These two lines are the basic
print "end"
```

Practice:
1. Change the program from "free fall" to "projectile". Set ball's initial position at vector(-15, size, 0), and ball's initial velocity at vector(10,10,0), vector(10, 10, 3), vector(10,10,-3), or your own choice.

2. Continuing with practice 1, find the total time the ball is in the air. Hint: You can name a new variable t=0 before the while loop and increase it by dt for every loop execution and print t in the end.

3. In simulation, we usually use “arrow” to indicate vectors such as velocity or acceleration. a1 = arrow(shaftwidth=0.1) draws an arrow with width = 0.1. arrow has attributes like pos, axis, and color. E.g. a1.pos = vector(1,0,0) makes a1 starting at (1,0,0), a1.axis = (1, 2, 0) draws a1 as a vector of (1, 2, 0). ， a1.color = color.green makes a1 green.

Write a program, in which you have to set up two vectors A and B, and a scalar s. Then you have to print the results of A+B, A-B, sA, AxB, A.B, and you have to plot (use the arrow object mentioned above) A+B in red, A-B in blue, AxB in green. In the plot, the three unit vectors (1,0,0), (0,1,0), (0,0,1) in yellow must be presented as well.

Homework submission:
Modify the projectile program with ball’s initial position = vector(-15, 2, 0), initial velocity = (8, 8, 0). Add on the right edge of the ball an arrow, length of which is proportional (proportional constant at your choice) to and parallel to the velocity vector of the ball. In the end of the program, print how much time the ball stays in the air. Save your program as “CW1_b104xxxxx.py” and submit it on ceiba.