Physics Name _____
Keller
VPython programming 1

1. Get a laptop according to your roster number. Get a mouse and plug it into one of the side USB ports. Click the Windows icon in the bottom-left and in the *Search programs and files* box, type VIDLE and hit enter.

2. All programs will begin with same initial line:

*from visual import **

this command simply imports all of the Visual Python features into the Python program

3. Skip a line and type:

scene.width=1000
scene.height=800

these commands define the size of the visual window that will appear when the program is run

4. Skip a line and type:

scene.autoscale=0
scene.range=(400,400,400)
scene.center=(0,0,0)

the first of these disables the automatic autoscaling of VPython, allowing you to determine the scale and keep it fixed
the second command sets that scale as having four-hundred units in the x, y, and z planes
the third command puts the position (0,0,0) at the center of that three-dimensional set of planes

5. Skip a line and type:

sphere1=sphere(pos=(0,0,0), radius=50, color=color.blue, material=materials.rough)

this command creates an object called "sphere1" and defines it to be a sphere with a given radius, color, and texture. You could also write it in block format, as such:

```
sphere1=sphere(
pos=(0,0,0),
radius=50,
color=color.blue,
material=materials.rough
)
```

This is a more standard programming format which makes it easier to find errors.

6. Now that we have set the scene and put an object in it, we can see how it looks. Click "Run" at the top and then select "Run Module". This will open both the VPython window in which you will see the ball you've made and also a *Python Shell* window with program information in the header.

7. In the VPython window, hold down both mouse keys and move the mouse up and down. This should zoom into and out of the scene. You can also pan around the scene by holding down the right mouse key and moving the mouse up and down, left and right. For now, it's only going to look like you're causing the sphere to rotate.

8. Close the Vpython window and the shell window, leaving only the program open. Skip a line and type:

print("This appears in the shell window")

9. Run the program again and you will see that the sphere is still created in the VPython window and the print command has been followed in the shell window. Close both again to return to the program.

10. Make the following changes to the program and run it after each change to notice the effect it has on the scene:

a. change the scene width and height to 600

b. change the scene range to (100,100,100)

c. change the scene center to (20,30,0)

d. change the scene center back to (0,0,0) and move the sphere to position (10,20,0)

e. shrink the sphere to a radius of 30

f. change the color to one you like (red, green, yellow, orange, cyan, magenta, white)

g. change the material to one you like (wood, plastic, marble, diffuse, emissive, shiny, chrome, blazed, silver, bricks)

11. Go to "File" then "Save As..." and save the file to your desktop as "Your_Name_Python1a.py" and then email it to akeller@longmeadow.k12.ma.us. If you make the next program without emailing this one first, there's a good chance it will be overwritten.

12. Now use what you have learned to write a program that opens a VPython window that fills the left half of the computer screen and displays a square of four different spheres, named sphere1, sphere2, sphere3, and sphere4. They must be arranged in the shape of a square and each sphere should have a unique position, size, color, and material. Save this file as "Your_Name_Python1b.py and then email it to akeller@longmeadow.k12.ma.us. Also, try zooming and panning as described in step 7.

Bonus: Go to http://vpython.org/contents/docs/primitives.html and replace the four spheres with a cone, an ellipse, a pyramid, and a ring. Submit using the file name "Your_Name_Python1B.py"

Physics Name _____
Keller
VPython programming 2

1. Sign-out your laptop. Create a visual window that fills the left half of the computer screen. Turn off the autoscale and make the scene range 800 in the x, 1000 in the y, and 800 in the z. Set the center of the scene to be (0, 0, 0). [Use Python 1, steps 1-4 for guidance]

2. To create the floor on which objects will slide around, create a box with the following line:

floor = box(pos=(0,-300,0), length=600, height=10, width=600, material=materials.wood)

3. Create a cube that can slide around on the floor with the following line:

cube1 = box(pos=(-290,-285,290), length=20, height=20, width=20, color=color.blue, material=materials.rough)

Because the floor has a height of 10 and is centered on a y-position of -300, the actual top of the floor has a position of -295. And if the cube has a height of 20 and is centered on a position of -285, the bottom of the cube is at -295. This puts the bottom of the cube on the top of the floor.

4. To make the cube slide around on the floor, you'll need to create a loop in the program. Type only what is italicized.

cube1.velocity = vector(1,0,0)

This tells the program that the cube will move with a velocity of one unit in the x-dimension, zero units in the y-dimension, and zero units in the z-dimension.

time = 0
time_step = 1

This just sets the starting time as zero and sets the cube to move in one second intervals

while cube1.pos.x < 290:

This creates a loop that will loop until the cube reaches the right edge of the floor. It reads, "While the cube's position in the x-dimension is less than the right edge of the floor's x-position, do the following".

    *cube1.pos = cube1.pos + cube1.velocity*time_step*

This line tells the cube to move from its current position to a new position based upon its velocity and one second of time passing. Lines like this which occur inside the while-loop must be indented as shown.

    time = time + time_step

This simply advances the clock as the program loops through the loop.

    rate(80)

This slows down the rate at which the computer cycles through the loop (80 times per second). Otherwise, the cube would move too fast across the floor.

These programs tend to freeze when finished, so it's always good to add as the very last line (untabbed):

exit( )

This will make the program quit once it has run.

5. Run this program and watch what happens. When it works, save it as Your_Name_Python2a.py and email the file to akeller@longmeadow.k12.ma.us

6. Right now, you have a blue cube sliding from the bottom left corner to the bottom right corner. Add another cube (with a different size and different material) that slides from the top right corner to the top left corner. To do this, just realize that everywhere there is a cube1 statement in the first program, there must now be a cube2 statement as well.

7. Save this file as "Your_Name_Python2b.py" and email the file.

Bonus: Make two more cubes that move from corner to corner so that you have four cubes rotating counter-clockwise. Save and email this as "Your_Name_Python2B.py"

Physics Name _____
Keller
VPython programming 3

1. Create a window that fills the left half of the computer screen. Turn off the autoscale, set the scene range to be 400 in the x, 500 in the y, and 400 in the z, and set the scene center to be (0,200,0). [Use Python assignment 1, steps 1-4 for guidance.]

2. Create a wooden floor with a y-position of 0 and a blue sphere (called sphere1) with a y-position of 300. [Use Python 2, step 2 and Python 1, step 5 for guidance]

3. Create a *while* loop that moves the sphere down towards the floor and stops when it gets there. [Use Python 2, step 4 for guidance]

4. Save this program as Your_Name_Python3a.py and email it to akeller@longmeadow.k12.ma.us.

5. Now we are going to do something new by making the sphere accelerate downward like a falling object would.

6. Change your time step to a value of 0.05 and make your initial velocity (0,0,0). Beneath your velocity vector line, add:

sphere1.acceleration=vector(0,-10,0)

And beneath the "position equals position plus velocity times time step" line in the while loop, add:

*sphere1.velocity.y= sphere1.velocity.y + sphere1.acceleration.y*time_step*

This is saying final velocity equals initial velocity plus acceleration times time.

7. When the sphere accelerates down to the floor and stops, save this program as Your_Name_Python3b.py and email it to akeller@longmeadow.k12.ma.us.

---

8. Now we are going to add some graphs to the display which demonstrate the motion of the falling sphere.

Beneath *from visual import ** type the following:

*from visual.graph import **

This line imports the graphics capabilities of VPython

Beneath *scene.center=(0,200,0)* type the following:

y_position_graph = gdisplay(x=600, y=0, width=400, height=275, xmin=0, xmax=10, ymin=0, ymax=400, xtitle='time (s)', ytitle='y-position (m)')

This creates a graph window (called "y_position_graph") where the top-left corner of the new window has a position of (600,0) on the computer screen. The window has a width of 400 and a height of 275. The graph in the window will have an x-axis that goes from 0s to 10s and a y-axis that goes from 0m to 400m. The axes will have titles of "time(s)" and "y-position(m)".

One line below, type:

position_line = gcurve(color=color.red)

This line instructs the program that a position graph will be drawn along these newly created axes and the graph will be red.

Beneath *sphere1.pos.y = sphere1.pos.y + sphere1.velocity.y*time_step* type the following:

position_line.plot(pos=(time,sphere1.pos.y))

This line instructs the program to plot a position point along the axes every time the while-loop cycles. The x-coordinate of this point along the graph will be based upon the time. The y-coordinate of this point along the graph will be based upon the sphere's y-position. Run the program to check that the position graph is correct.

Notice there are three different variables involved:

y_position_graph is the name of the window in which the graph appears
position_line is the name of the line graphed and is used in both the *gcurve* and *plot* commands
sphere1.pos.y is the actual numeric value that is plotted every time the while loop cycles

9. Now add a green velocity graph and a blue acceleration graph that appear beneath the position graph. Choose appropriate scales and titles for the axes. When all three graphs appear correctly, save this program as Your_Name_Python3c.py and email it to akeller@longmeadow.k12.ma.us.

Bonus: Modify the program so that the sphere bounces off the floor. To do this, you only need to change the while condition to something like:

while time < 1000:

Also, inside the while loop, include the conditional statement:

if [the y position of the sphere is less than the top of the floor]*:*
[the y-velocity of the sphere equals the y-velocity of the sphere times negative one]

You need to translate what is written above in brackets into computer code.

Save this program as Your_Name_Python3B.py and email it to akeller@longmeadow.k12.ma.us.

Physics Name _____
VPython programming 4

1. Sign-out a laptop. Create a visual window that fills the left half of the computer screen. Turn off the autoscale and make the scene range 800 in the x, 1000 in the y, and 800 in the z. Set the center of the scene to be (0, 0, 0). [Use Python 1, steps 1-4 for guidance]

2. Create a wooden floor near the bottom of the visual window. [Use Python 2, step 2]

3. Create a marble tower that sits on the left edge of the floor and is about as tall as the floor is wide. The tower can be a box just like the floor is, but with a much smaller length and width and a much greater height. [Use Python 2, step 3]

4. Create a colored sphere (called sphere1) that sits on the top of the tower. [Use Python 1, step 5]

5. To make the sphere move, we are going to give it an initial velocity and acceleration with the following lines:

```
sphere1.velocity=vector(50,0,0)
sphere1.acceleration=vector(0,-10,0)
```

In standard units, this gives the sphere an initial velocity of 50m/s to the right and an acceleration of -10m/s^2 downward, just like on the surface of the Earth.

6. Now, define the starting time as zero and give the upcoming while loop a time step.

```
time=0
time_step=0.02
```

7. To make the sphere move, create a while loop.

```
while sphere1.pos.x < 300 and sphere1.pos.y > -400:
    sphere1.pos = sphere1.pos + sphere1.velocity*time_step
    sphere1.velocity = sphere1.velocity + sphere1.acceleration*time_step
    time = time + time_step
    rate(80)
```

Here is what each of those lines is doing:

while sphere1.pos.x < 300 and sphere1.pos.y > -400:
tells the sphere to keep moving as long as it is to the left of the floor's right edge and above the floor;
you will probably need to adjust the numbers 300 and -400 to match your floor

*sphere1.pos = sphere1.pos + sphere1.velocity*time_step*
makes the sphere move from its previous position to its next position

*sphere1.velocity = sphere1.velocity + sphere1.acceleration*time_step*
makes the sphere change from its previous velocity to its next velocity

time = time + time_step
advances the internal clock of the while loop

rate(80)
slows the loop down to running only 80 times per second

Once you have a sphere that leaves the tower and stops when either it reaches the floor or the right boundary of the floor, save it as Your_Name_Python4a.py and email the file.

8. Now change the initial velocity of the sphere so that it only travels about one-quarter of the way across the floor before it hits the floor.

9. Also, change the while loop to having only one condition:

while sphere1.pos.x < 300:

Again, your number will probably be something other than 300. This will stop the program only when the sphere gets to the right side of the floor.

10. We are going to now make the sphere bounce when it hits the floor with this statement underneath the *rate(80)* statement:

```
if sphere1.pos.y < -400:
        sphere1.velocity.y = sphere1.velocity.y*-1
```

What this states is:

If the sphere hits the floor, take its downward velocity and make it positive. This is just like a ball bouncing. It strikes the floor at something like -10m/s and then almost instantly leaves the floor at +10m/s, which is its previous velocity multiplied by negative one.

Again, you will need to adjust the number -400 to match your own floor. When the sphere bounces across the floor and stops at the right edge of the floor, save as Your_Name_Python4b.py and email the file.

---

11. Now we'll add a few more options. One is to draw a vector which represents the instantaneous velocity of the projectile. Do this with the following line before the while loop:

velocity_vector=arrow(pos= sphere1.pos, axis= sphere1.velocity, shaftwidth=2)

And inside the while loop:

```
velocity_vector.pos= sphere1.pos
velocity_vector.axis= sphere1.velocity
```

The first of the three lines creates the arrow, the second places the base of the arrow at the center of the ball, and the third orients the arrow along the velocity of the ball.

12. We'll add a trail that shows the path of the projectile in flight. All it takes is adding the following to the properties of the ball after material=materials.rough:
make_trail=True

13. Now we'll add a display that shows the speed of the ball as it is in flight. Use this line before the while loop:
speed_label=label(pos=(0,0,0), height=10, color=color.yellow, border=4)

14. And these lines in the while loop:

```
sphere1_speed=round(mag(sphere1.velocity),1)
speed_label.text = "Speed is: " + str(sphere1_speed) + " m/s"
```

What the first of these two lines is doing is defining the speed of the sphere to be the velocity vector magnitude and then rounding it to one decimal place. The second line is defining the text to be inserted into the label created before the while loop.

When your program runs correctly, save as Your_Name_Python4c.py and email the file.

15. Now add four graphs on the right side of the computer screen. [Use Python 3, Step 8]
x-position vs. time y-position vs. time
x-velocity vs. time y-velocity vs. time

When your program runs correctly, save it as Your_Name_Python4d and email it.

Bonus: Remove the tower and adjust the program so that the sphere leaves the floor in an arc and then bounces a few times before reaching the right side of the floor. Save this as Your_Name_Python4B.py and email it.