3. WRITTEN RESPONSES

3 a.

3.a.i.

The purpose of the program is to entertain and to educate students by deepening knowledge of projectiles through an accurate physics simulation.

3.a.ii.

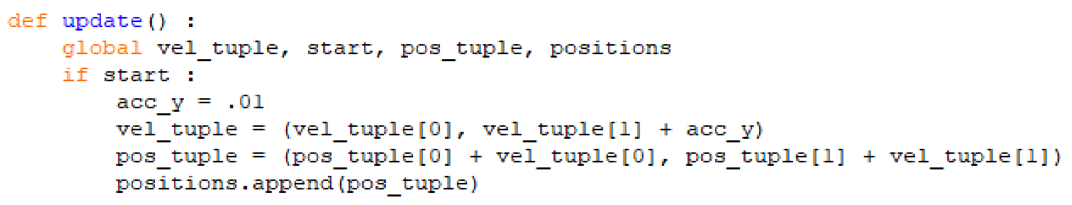
A realistic projectile simulation is displayed based on user inputs changing the initial angle and velocity. The projectile shoots across the screen, and after it goes off the screen, dots trace in with density showing speed. The entire simulation works to increase knowledge of projectile motion.

3.a.iii.

The user inputs the up and down arrow to increase or decrease the initial angle by 15 degrees each press respectively. The user can also input the right and left arrow to increase or decrease the starting velocity respectively. When the angle and velocity are where desired the user can input the enter button to run the simulation. Upon pressing enter, an output of a black ball animates across the screen with accurate motion; once the ball is off the screen, the program outputs dotted lines with density of lines showing speed. The user can then input the space bar to clear the screen and reset.

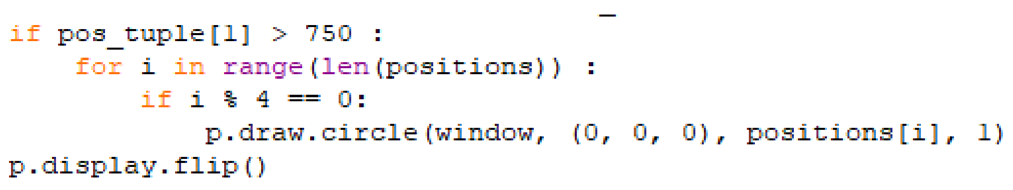
3 b.

3.b.i.



```
def update() :
    global vel_tuple, start, pos_tuple, positions
    if start :
        acc_y = .01
        vel_tuple = (vel_tuple[0], vel_tuple[1] + acc_y)
        pos_tuple = (pos_tuple[0] + vel_tuple[0], pos_tuple[1] + vel_tuple[1])
        positions.append(pos_tuple)
```

3.b.ii.



```
if pos_tuple[1] > 750 :
    for i in range(len(positions)) :
        if i % 4 == 0:
            p.draw.circle(window, (0, 0, 0), positions[i], 1)
p.display.flip()
```

3.b.iii.

positions

3.b.iv.

It represents a list of tuples with each tuple representing the position the projectile has been at each frame. All together the list represents every position the object has been at this run.

3.b.v.

The program utilizes the list by iterating over the list and drawing a dot at the position tuple provided by every fourth index. This is produced by only drawing a dot when the index modulo 4 is zero. The program only draws the dots once the object is off the screen, so a list is used to store positions so they can be accessed once the object is off screen. Without a list it would not be able to iterate over the list drawing a dot at every fourth element in the list once the object is off the screen. Instead, without a list, the program would have to somehow create a new variable for each tuple at each frame and create new variables in real time while figuring out how to wait and draw a dot at each of these; this would be extremely complex and messy if possible.

3 c.

3.c.i.

```
def render(positions, pos_tuple) :
    window.fill(p.Color(255, 255, 255))
    #Used Pygame Library for Font Syntax
    font = p.font.SysFont("Times New Roman", 20)
    degree = round(ang * (180 / m.pi), 0)
    str_ang = str(degree)
    str_vel = str(vel)
    vel_print = font.render(str_vel, True, (0, 0, 0))
    ang_print = font.render(str_ang, True, (0, 0, 0))
    window.blit(vel_print, (5, 730))
    window.blit(ang_print, (5, 750))
    p.draw.circle(window, (0, 0, 0), pos_tuple, 10)
    if pos_tuple[1] > 750 :
        for i in range(len(positions)) :
            if i % 4 == 0:
                p.draw.circle(window, (0, 0, 0), positions[i], 1)
    p.display.flip()
```

3.c.ii.

```
positions = []
pos_tuple = (10, 800)
start = False
while running :
    input_angle(ang, vel)
    update()
    render(positions, pos_tuple)
```

3.c.iii.

The identified procedure clears the previous screen by covering it with white. It then displays the angle and velocity before firing. While firing, the procedure draws the circle at each position determined by the update function producing an animated ball movement. Once the ball is off the screen, the procedure also draws a dot at the position the ball was at every fourth frame. Since it draws at every fourth frame the closer the dots are the slower it is moving and the farther they are the faster. Displaying the angle and velocity add to the functionality of the program because it allows the user to knowingly control them and see how they affect the objects movement. Displaying the object adds to the functionality by showing the realistic movement of the projectile in a parabolic motion and how changing velocity and angle transform the parabola. Displaying the dots after firing adds to the functionality by highlighting the realistic parabolic motion and showing the speed based on how close the dots are together which furthers the understanding of projectile motion.

3.c.iv.

The algorithm first fills the screen white to clear the screen. Next, the function defines a font variable to use when drawing the velocity and angle. Then, the function coverts the angle in radians represented by variable ang to degrees and rounds to the nearest whole degree to account for small errors. The procedure then turns that new degree and the velocity variable into strings so they can be printed to the screen later. After that, the procedure makes two new variables for the font image for the degree and velocity string. Next, the function draws both font images to the screen. After drawing the starting velocity and angle, the function draws a black circle at the position tuple parameter location. After drawing the object, the procedure checks to see if index 1 of the position tuple parameter is greater than 750. This determines if the ball is close to going of screen. If yes, the procedure iterates over the positions list and draws a dot at every fourth element in the list which are position tuples. If not greater than 750 it does nothing. Finally, the function flips the screen.

3 d.

3.d.i.

First call:

render(positions, pos_tuple) when pos_tuple = (50,50)

Second call:

render(positions, pos_tuple) when pos_tuple = (1000, 760)

3 d.ii.

Condition(s) tested by first call:

The first call tests if the pos_tuple index 1 is greater than 750 to check if the object is close to going off bottom of screen.

Condition(s) tested by second call:

The second call tests if pos_tuple index 1 is greater than 750 to check if the object is close to going off bottom of screen.

3.d.iii.

Results of the first call:

Since the pos_index 1 is not greater than 750, the program skips the code under the conditional and does not draw the dots.

Results of the second call:

Since the pos_idex 1 is greater than 750, the program executes the code under the conditional and iterates over the programs list drawing dots at every fourth position tuple contained in the list.