Pygame Workshop

Pygame - It is a python library used for creating video games using Python. It consists of various functions which support in adding graphics to your game.

About the Ws - We'll be making a 2 player game using this library!

Requirements for the Workshop -

1. Python installed in your PCs: you can install python into your PCs, if you haven't until now, from the link given below. Make sure to install the one with the correct OS. Also check for 32 or 64 bit.

https://www.python.org/downloads/

2. Install pip for your PC: wondering what pip is? It stands for Preferred Installer Program. For further info, look at this link :

https://www.makeuseof.com/tag/install-pip-for-python/

3. Install pygame: now that you have installed pip, installing pygame is easy. Make sure that pip is in your environment variables. Open cmd and type

```
pip install pygame
```

This should install the pygame framework.

Basic functions for Pygame:

(imported pygame as py hence , py.”function name”)

1.py.init() - Initializes all the functions required for later use in the game.

2.py.display.set_mode((l,b)) - Sets a window for our game of length ‘l’ and breadth ‘b’.

3.py.display.update() - Updates all changes on the window.

4.screen.blit(image,(x,y)) - Adds an image to the (x,y) co - ordinates where x and y represent the top left corner of the image.(Here screen is my display windows)

5.image.get_rect() - returns a rectangular surface of the image.
Also , we can change the center of the rectangle by
‘rectobj’.center =(x,y)
which set’s the center of the rectangular surface to x and y.
Everywhere in the game , we set the rect.center instead of specifying the x and y co - ordinates of the image as it is easier to get the rectangular surface of the rotated image.

Game Description:

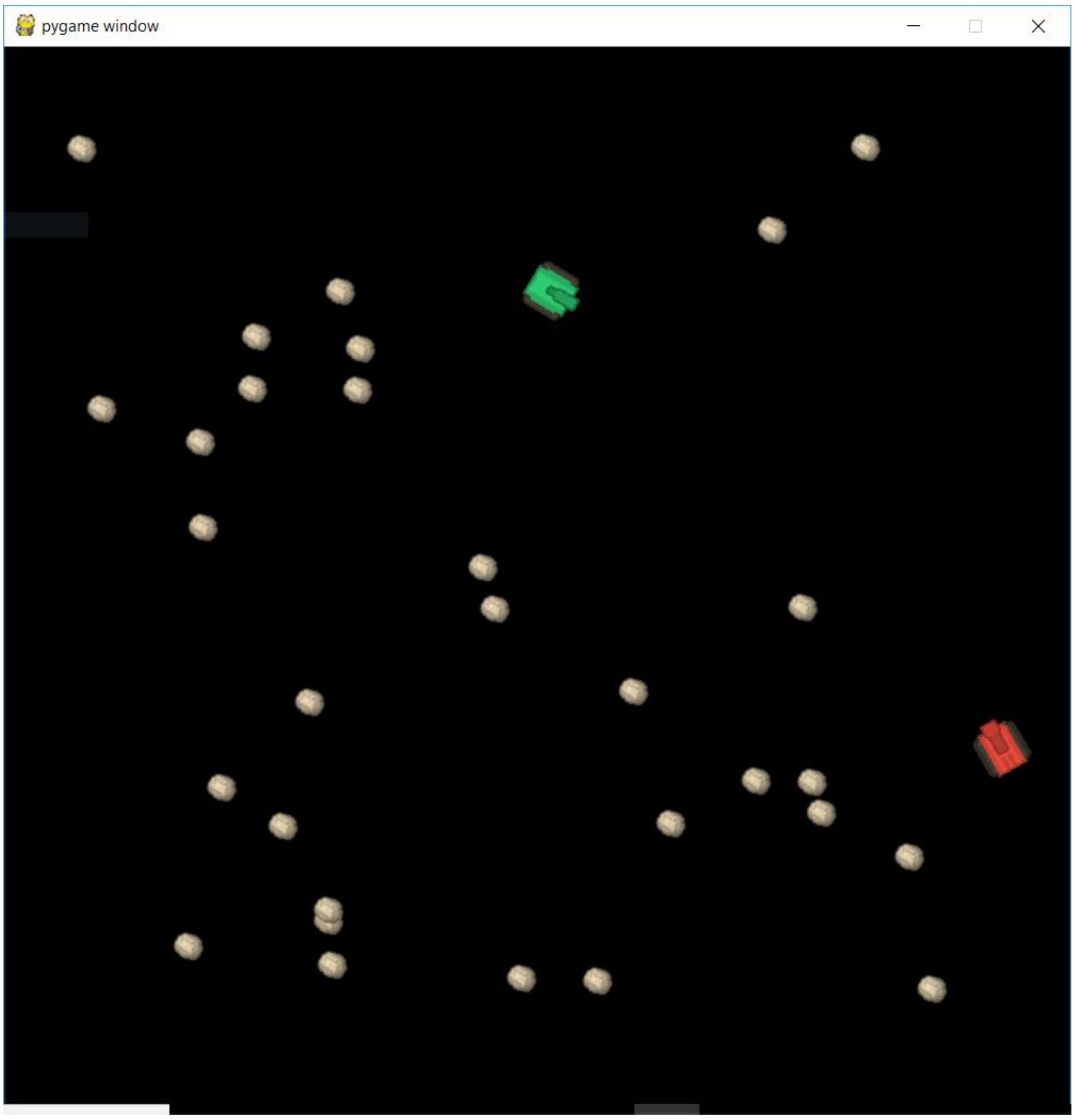
Our game , is a 1v1 game which consists of two tanks .
The tanks have three options:

1.To rotate at its position
2.To fire a bullet.
3.To translate in a direction in which it stopped rotating.

The first player to hit the other one , wins!

Also , there are obstacles in the way which stop the movement of bullet as well as the tank.

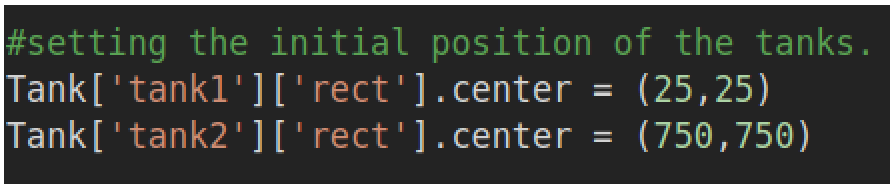
After the game is finished , it would look like this:



Basic flow of the game:

For every update on the screen , we would have to call the draw() function . In this function , we set the window , obstacles , the position of tanks.

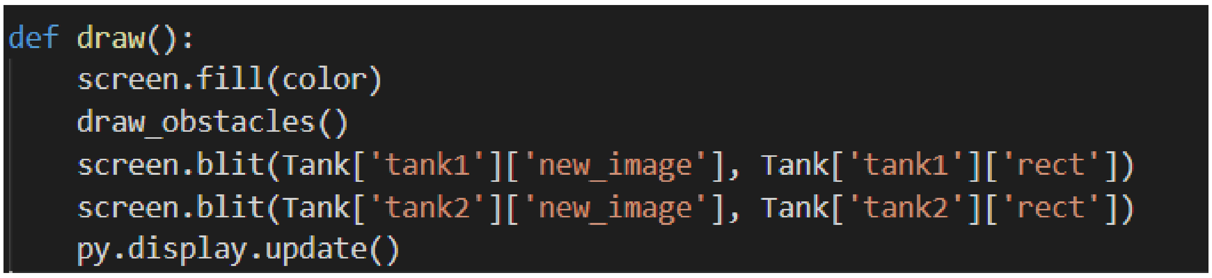
Setting the initial position of the tanks:



```python
#setting the initial position of the tanks.
Tank['tank1']['rect'].center = (25,25)
Tank['tank2']['rect'].center = (750,750)
```

Here , we are setting the center of their respective centers to an initial position . Refer the ‘Basic functions’ section for further elaboration on pygame.get_rect() and it’s properties.

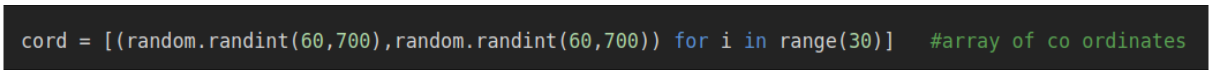
Draw() function :



```python
def draw():
    screen.fill(color)
    draw_obstacles()
    screen.blit(Tank['tank1']['new_image'], Tank['tank1']['rect'])
    screen.blit(Tank['tank2']['new_image'], Tank['tank2']['rect'])
    py.display.update()
```

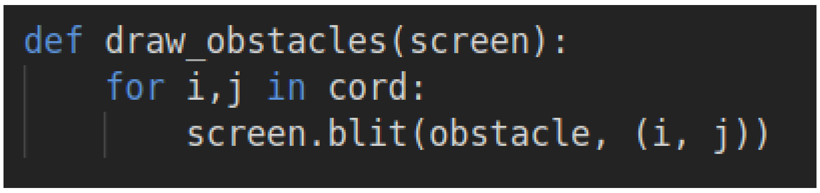
As we can see above , the screen is filled every time the draw function is called and the updated position of tanks 1 and 2 are set.
Note: Here , ‘Tank’ is a dictionary which holds the properties of both the tanks - tank1 and tank2 .(properties such as their rectangular surface , their image , their positions ,etc).

Generation of co - ordinates:



```
cord = [(random.randint(60,700),random.randint(60,700)) for i in range(30)]   #array of co ordinates
```

The above array , stores the random number pairs between 60 and 700. We have added 30 such pairs .
Now a simple draw_obstacles method will place the obstacle images on these co - ordinates . This can be done by simple looping.



```
def draw_obstacles(screen):
    for i,j in cord:
        screen.blit(obstacle, (i, j))
```

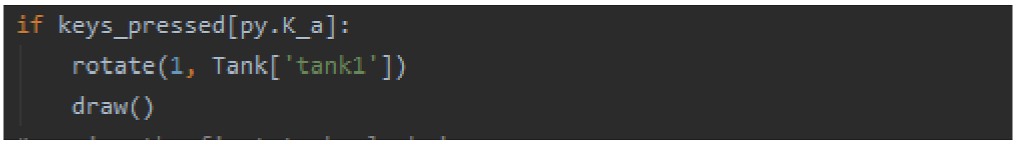
As we can see above , (i,j) are the (x,y) co - ordinates of the obstacles . Hence , we use the blit function to place the image at those positions.
Note: Here, ‘obstacle’ is an image of an obstacle in our game.

Code for the tank to rotate:-

Here, we first check if key ‘a’ is pressed. If yes than we pass 2 parameters into the rotate function:

1)integer 1 or -1 where -1 indicates clockwise rotation and 1 indicates anticlockwise rotation.

2)Tank[name]-as on which tank should the rotation function be applied.



```
if keys_pressed[py.K_a]:
    rotate(1, Tank['tank1'])
    draw()
```

Then we call the draw function to blit and flip the changes made onto the screen.

Next ,we rotate the tank using the following steps;

1) Then, we change the angle property of the respective tank by 3 degrees clockwise or anticlockwise depending upon the clock value(-1 or

1). Mod 360 because on adding 3 everytime it might end up being 363 degrees sometimes so mod 360 will make it 3 degrees.

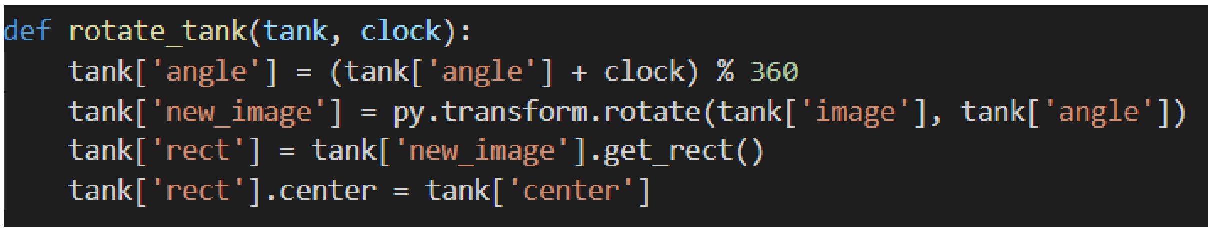
2)Then we take the original image, we rotate it with the angle of the given tank and then we store it in the new_image attribute of the tank.

3)Using py.transform.rotate function we rotate the tank image by that much angle calculated above.

4)The new image thus is formed.

5)We get the coordinates of the new image using get rect() function.

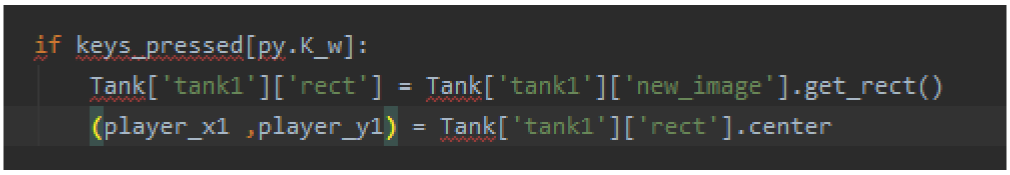
6)As in rotation the tank rotates about the same point hence the coordinates of the center needs to be brought back to the initial positon .Hence we made that old center to keep the center intact.



```
def rotate_tank(tank, clock):
    tank['angle'] = (tank['angle'] + clock) % 360
    tank['new_image'] = py.transform.rotate(tank['image'], tank['angle'])
    tank['rect'] = tank['new_image'].get_rect()
    tank['rect'].center = tank['center']
```

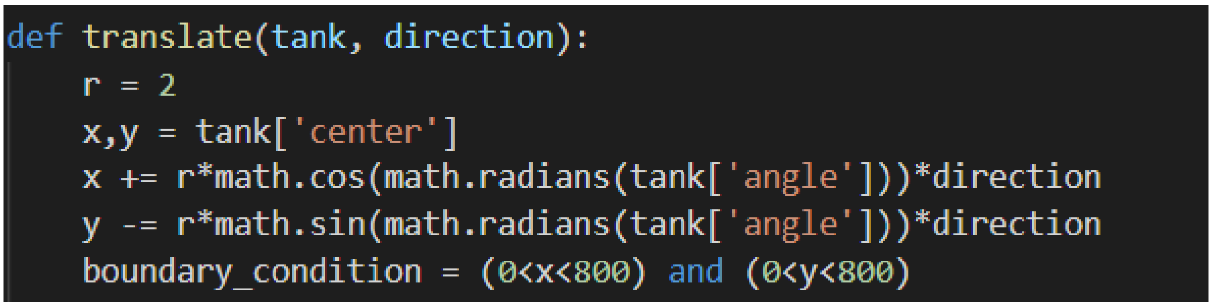
Code for the tank to move:-

 Here, we first check if key ‘a’ is pressed. If yes then we assign the co-ordinates of the tank to the current co-ordinates.



```
if keys_pressed[py.K_w]:
    Tank['tank1']['rect'] = Tank['tank1']['new_image'].get_rect()
    (player_x1 ,player_y1) = Tank['tank1']['rect'].center
```

Next is a little maths for translation:-

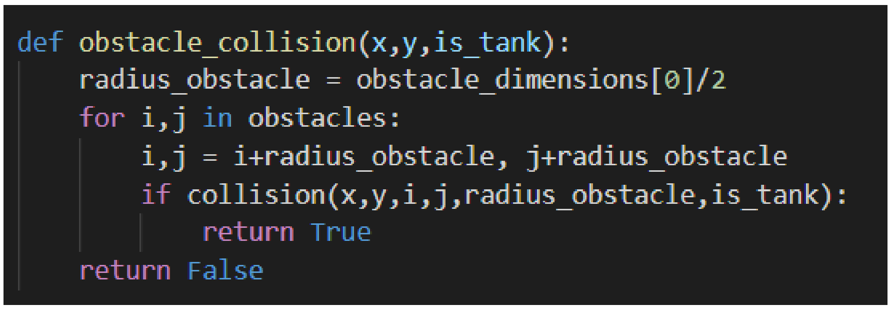


```python
def translate(tank, direction):
    r = 2
    x,y = tank['center']
    x += r*math.cos(math.radians(tank['angle']))*direction
    y -= r*math.sin(math.radians(tank['angle']))*direction
    boundary_condition = (0<x<800) and (0<y<800)
```

We store the current center of the tank in the variables x and y. Now, using a little trigonometry. If we say an object is moving straight with a velocity v, then we say that it travels v units in 1 second. If it is moving at an angle, the horizontal motion (x) increases by distance*cos(angle) and the vertical motion (y) increases by distance*sin(angle)

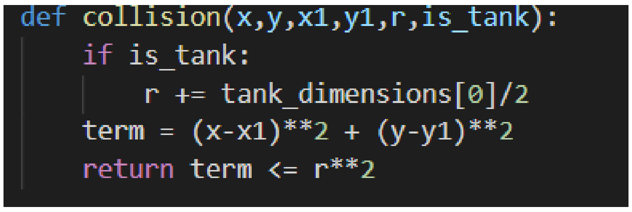
Sine and cosine functions are available in math library of python but the angle should be provided in radians.

Next the **obstacle_collision()** function is called wherein this equation is checked:-



```python
def obstacle_collision(x,y,is_tank):
    radius_obstacle = obstacle_dimensions[0]/2
    for i,j in obstacles:
        i,j = i+radius_obstacle, j+radius_obstacle
        if collision(x,y,i,j,radius_obstacle,is_tank):
            return True
    return False
```

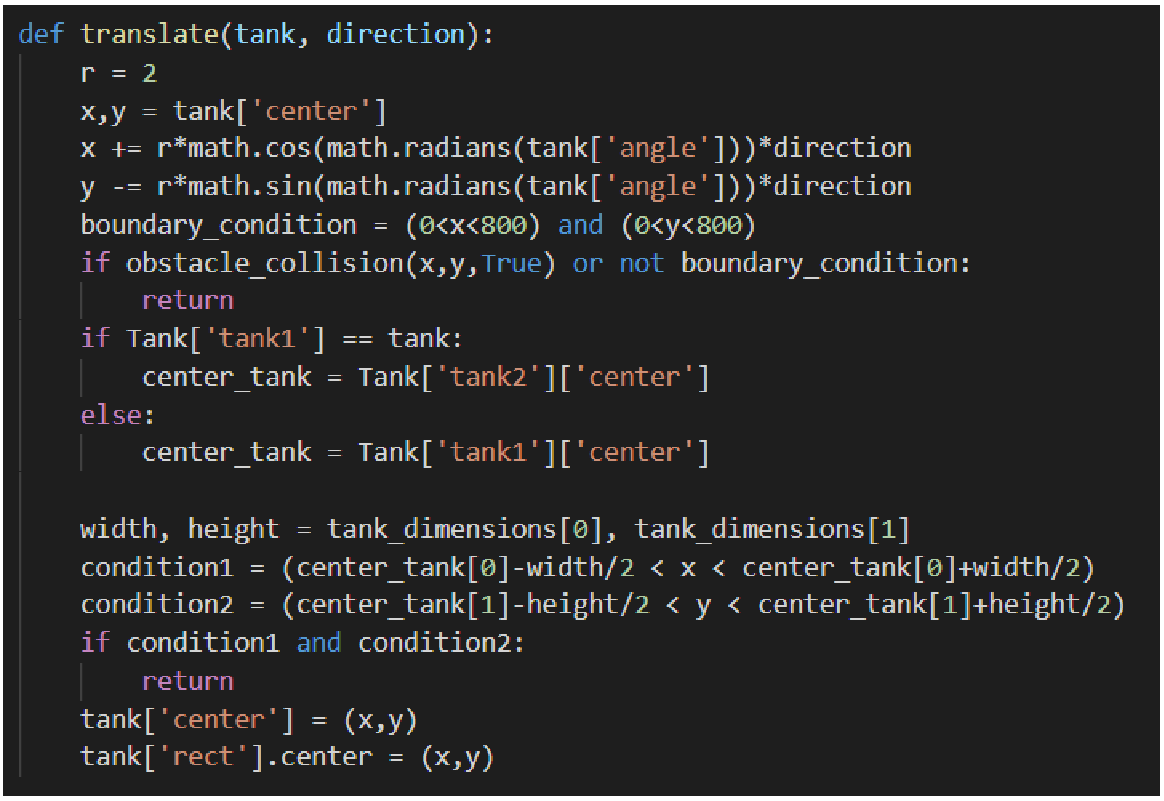
We know the top-left corner coordinates and so to get the coordinates of the center, we add the radius to the coordinates.The collision function is



```python
def collision(x,y,x1,y1,r,is_tank):
    if is_tank:
        r += tank_dimensions[0]/2
    term = (x-x1)**2 + (y-y1)**2
    return term <= r**2
```

We check the distance between the two points. If the distance is less than the radius, we say that the two objects collide and hence it will return true. Else false. If the object is the tank, we are going to add half the width of the tank as well because we do not want that half to be overlapped with the rock.

If it holds true then we actually update the co-ordinates of the tank and last the **draw()** function is called to display the changes made….



```python
def translate(tank, direction):
    r = 2
    x,y = tank['center']
    x += r*math.cos(math.radians(tank['angle']))*direction
    y -= r*math.sin(math.radians(tank['angle']))*direction
    boundary_condition = (0<x<800) and (0<y<800)
    if obstacle_collision(x,y,True) or not boundary_condition:
        return
    if Tank['tank1'] == tank:
        center_tank = Tank['tank2']['center']
    else:
        center_tank = Tank['tank1']['center']

    width, height = tank_dimensions[0], tank_dimensions[1]
    condition1 = (center_tank[0]-width/2 < x < center_tank[0]+width/2)
    condition2 = (center_tank[1]-height/2 < y < center_tank[1]+height/2)
    if condition1 and condition2:
        return
    tank['center'] = (x,y)
    tank['rect'].center = (x,y)
```

Now we also need to check if the two tanks are colliding with each other or not. To do so, we just check if the center of the current tank is in the rectangle of the other tank.

Code for bullet to move:-

Everything is similar to that of tank’s movement conditions just the catch over here is that once the bullet is fired it should keep moving until it hits an obstacle or a tank or has reached the end of screen…

So we assign the co-ordinates of bullet and next is a while loop and the limits here are same as the size of our screen……….

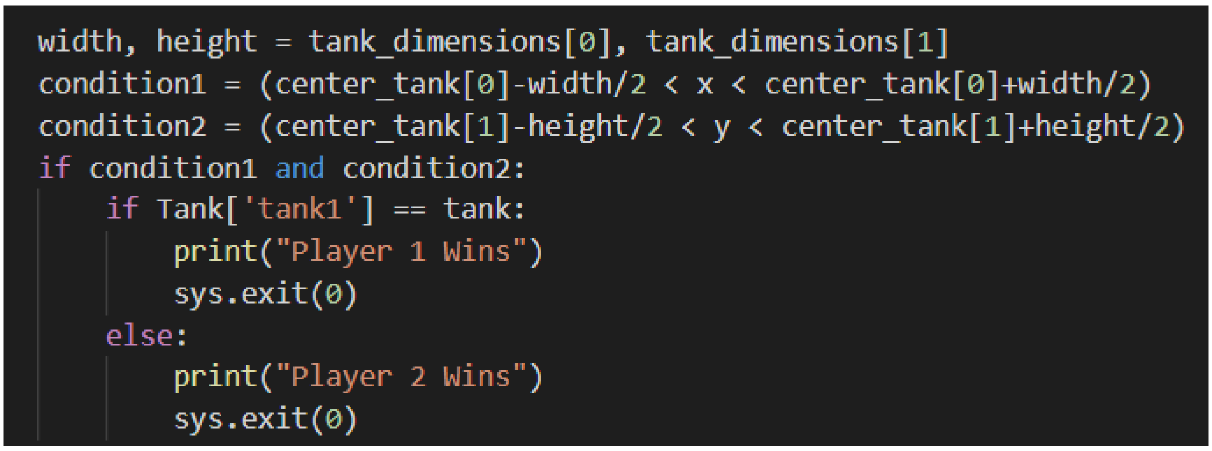
We do the same thing to check if the bullet is hitting the other tank or not. If it does, we end our game.

And under this while loop all conditions which were above there in tank come as it is….(with minor changes though)

Winning condition:-

So finally how is anyone going to win????

Well no worries!!! check this:-



```
width, height = tank_dimensions[0], tank_dimensions[1]
condition1 = (center_tank[0]-width/2 < x < center_tank[0]+width/2)
condition2 = (center_tank[1]-height/2 < y < center_tank[1]+height/2)
if condition1 and condition2:
    if Tank['tank1'] == tank:
        print("Player 1 Wins")
        sys.exit(0)
    else:
        print("Player 2 Wins")
        sys.exit(0)
```

What we pass in this function are the co - ordinates of bullet and the opposite tank to which the bullet will attack.

Next we check the same old equation and finally if that holds true we return true, which in turns say that the bullet has destroyed our opposite tank.

Else we keep moving forward