**OFFICE WORKER**

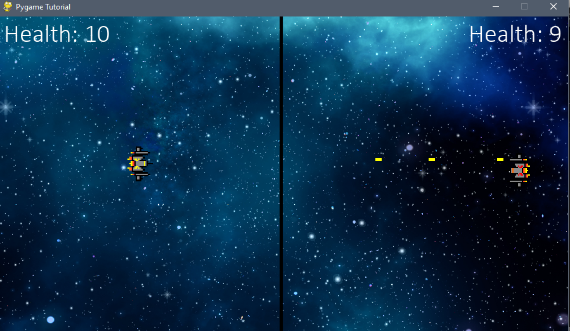
Goals:

* Produce a working and fun game using the pygame libary.
* Improve my programming skills in python using pygame.
* Provide employers with the evidence of making my own fun projects.
* HAVE FUN.

Initial ideas:

I want to make a simple game with simple graphics where the player works as an office worker and wants to collect as many briefcases as possible and dump that at an office desk. I hope to be able to include a database of some sort with the highest scores of a user, as well as basic animations and random events. I also hope to include basic audio and visual effects for certain events in the game.

Stage 1: Basic framework of the game

To get started into pygame, I had to watch certain basic tutorials venturing into the mechanics of pygame. I started off with replicating a space shooter game where two players would shoot each other, until the other player dies.

By following a tutorial online, I was able to create this basic game, which included motion, audio and a score system.

Through this basic tutorial, I was able to start the basic development of my office worker game.

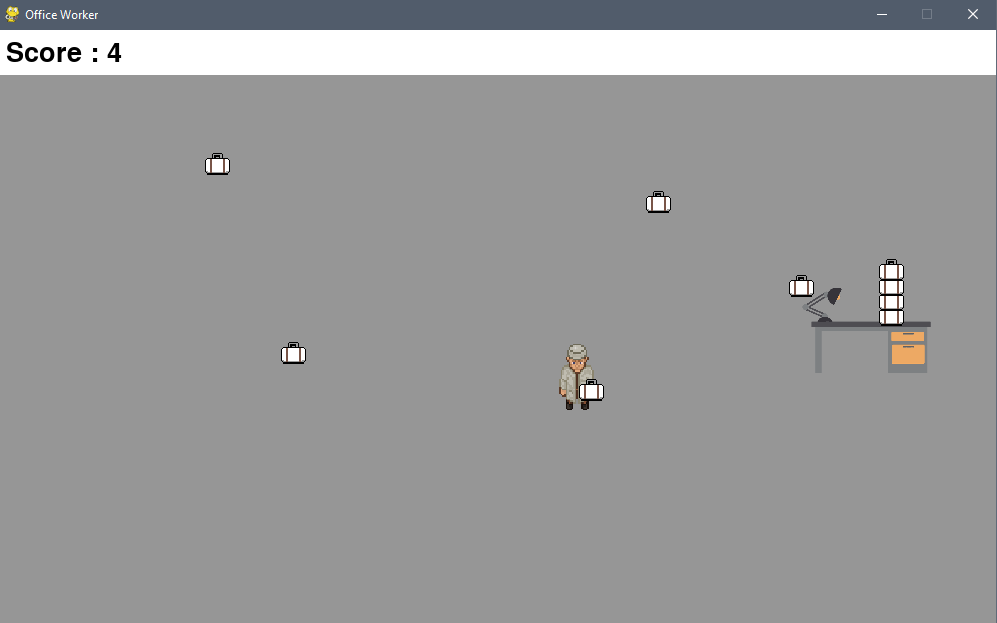
Using pygame, I am able to create an object of type Rect, which defines the dimensions of a certain object. I instantiated this for each briefcase and the character, as well as the desk. I also separated the game’s functionality into separate function, to make it easier to organise. These include:

* Main() – where all other functions are called, and where the main game is set up.
* drawWindow() – where all functionality to do with drawing my objects is held.
* workermotion() – where I could deal with the movement of my character
* handleworker() – where I could deal with events to do with the worker. i.e collisions, etc.

At this first stage, I also created a timer class, to keep track of time and to allow for random spawning delay of the briefcases. Also, I tried to use variables for constant values such as the widths and heights of the character, so that if I needed to change them, it would affect the whole program.

pygame.event allowed me to create two events, one which was when the worker collided with a free briefcase, and the other when the worker collided with the desk. By this, I can judge when the score should increase by 1.

At this stage, I have created basic animation, in that after collecting the briefcases, the worker holds the briefcase and when the worker drops the briefcase, the briefcase appears at the desk. This is done through WIN.blit and using the desk coordinates as a basis of where to draw the briefcase rect object.

This is what my initial prototype looks like.

Stage 2: Adding a win/lose factor:

To win or lose and not just continue to keep collecting briefcases, I need to add a win/lose factor. I will do this by adding a bar at the top of the game, resembling health (in this case coffee energy). Once, the energy runs out the game ends. I will also have to add a coffee station which will only refill every x amount of time. Coffee energy can be affected by time progression and briefcases collected.

if event.type == DROPBRIEF:

                score+=1

                health-=5

            if event.type == COLLECTBRIEF:

                health-=5

By utilizing user events, I can deduct the global variable health by 5 each time the worker interacts with a briefcase. I can then pass this variable to a drawBorder() function, which will animate a health bar as such:

def drawBorder(score,health):

    #drawing white border

    pygame.draw.rect(WIN,WHITE,(0,0,WIDTH,BORDERHEIGHT))

    #drawing the health bar with a black outline

    pygame.draw.rect(WIN,BLACK,(295,6,310,37))

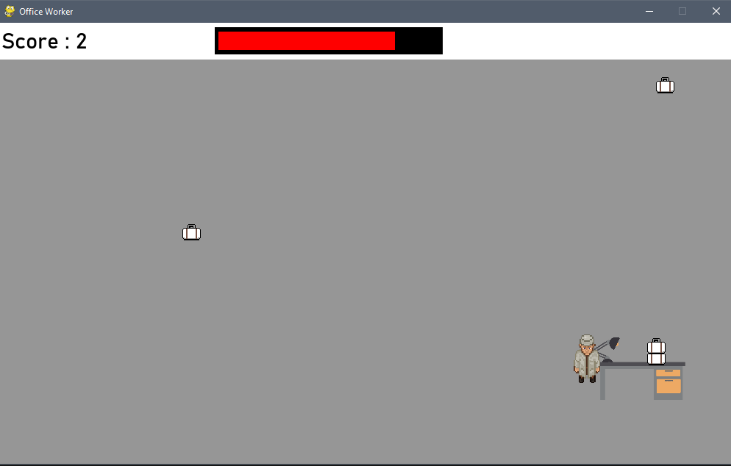
    pygame.draw.rect(WIN,RED,(300,12,health\*3,25))

    #drawing score

    scoreText = scoreFont.render("Score : "+str(score),1,BLACK)

    WIN.blit(scoreText,(5,5))

    pygame.display.update()

In this basic implementation, whenever I drop/collect a briefcase, the worker loses health, observable by the health bar.

Now I need to implement a method to add health. This led me to the idea of a coffee machine, which I implemented as another Rect() object and whenever it collided with the worker, the userevent COFFEE would be added to the queue, thus adding health to the worker.

Delays:

One issue I encountered was how to implement delays in pygame. Because you cannot simply tell the pygame to delay for a set amount of time, I had to use my previous Timeclock class as well as pygame.set\_timer() to introduce delays into certain events. In this case, I added a delay to how often a worker could drink coffee, as well as adding a random delay between spawning briefcases. Here is the implementation:

#user events

COLLECTBRIEF = pygame.USEREVENT+0

CREATEBRIEF = pygame.USEREVENT+1

pygame.time.set\_timer(CREATEBRIEF, 2000)

DROPBRIEF = pygame.USEREVENT+2

COFFEE = pygame.USEREVENT+3

COFFEEDELAY = pygame.USEREVENT+4

if event.type == COFFEEDELAY:

                if healthTimer.delay(dt,3000):

                    startHealth = 0

            if event.type == COFFEE:

                if health >= 100:

                    health = 100

                elif startHealth<15:

                    health +=1

                    startHealth+=1

                elif startHealth >=15:

                    pygame.event.post(pygame.event.Event(COFFEEDELAY))

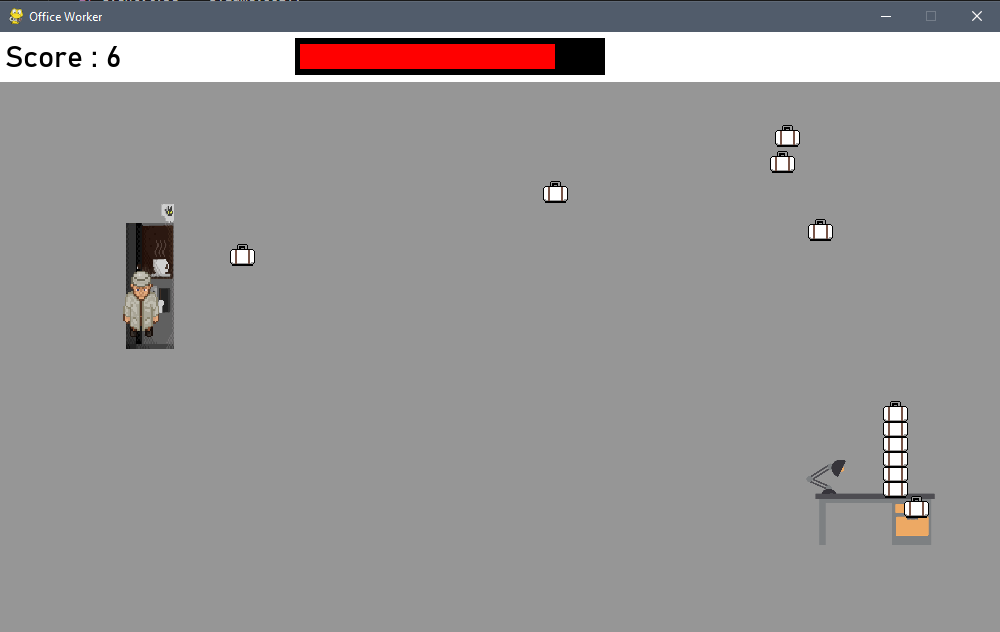
if event.type == CREATEBRIEF:

                #placing briefcase at random times, at random locations

                briefcase = pygame.Rect(random.randint(0,WIDTH-BRIEFCASEWIDTH),random.randint(BORDERHEIGHT,HEIGHT-BRIEFCASEHEIGHT),BRIEFCASEWIDTH,BRIEFCASEHEIGHT)

                briefcases.append(briefcase)

                pygame.time.set\_timer(CREATEBRIEF, random.randint(1500,3000))

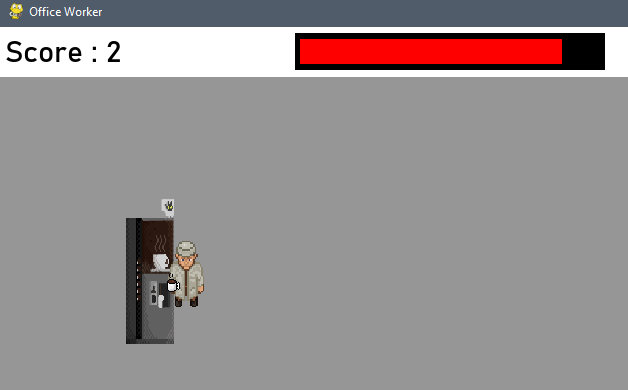


Animating the coffee:

Now having implemented the coffee machine, I wanted to animate the worker to drink the coffee. This would be done simply by drawing the coffee at the free hand and after a certain time, it would disappear. To do this I created a global Boolean variable, to tell drawWindow() function whether or not the worker was drinking coffee. When they were drinking coffee, the function would draw the coffee image onto the worker’s hand to resemble them drinking coffee.

 if coffee == True:

        WIN.blit(coffeeImg,(officeWorker.x,officeWorker.y+30))



Degradation of health:

Over time, I wanted the worker to gradually lose health, to resemble hunger/energy of a real office worker. I did this through user events, which would appear on the queue every second. If the user event was on the queue, health would decrease.

HEALTHDEGRADE = pygame.USEREVENT+5

pygame.time.set\_timer(HEALTHDEGRADE,100)

One thing I had to correct, was that the health would continue to degrade when taking coffee. I needed to use a Boolean to correct this, which would affect the user event queue. I did this below:

if event.type == COFFEEDELAY:

                healthdegrade = True

                coffee = False

                if healthTimer.delay(dt,3000):

                    startHealth = 0

            if event.type == COFFEE and event.type!=COFFEEDELAY and officeWorker.colliderect(coffeeStation):

                if health >= 100:

                    coffee = False

                    health = 100

                elif startHealth<20:

                    healthdegrade = False

                    coffee = True

                    health +=0.2

                    startHealth+=0.2

                elif startHealth >=20:

                    coffee = False

                    pygame.event.post(pygame.event.Event(COFFEEDELAY))

            else:

                coffee = False

                healthdegrade = True

            if healthdegrade == True and event.type == HEALTHDEGRADE:

                health-=0.1

Adding a boss:

Overtime, the briefcases would get stacked up without being removed. Thus, I needed to add an animation of a boss collecting the briefcases one by one. I did this by creating a boss motion and an object of type pygame.rect. Every time a briefcase is dropped, the boss would come to collect it. This was simply a few if statements, whether or not there were briefcases on the desk and whether or not the boss was collecting a briefcase.

def bossMotion(boss,desk,bossbriefcases,deskbriefcases):

    if len(deskbriefcases)>=1 and boss.x>desk.x+60 and len(bossbriefcases)==0:

        boss.x-=2

    elif len(bossbriefcases)==1:

        boss.x+=2

def handleBoss(boss,desk,deskbriefcases,bossbriefcases):

    if boss.x <= desk.x+60 and len(deskbriefcases)>=1:

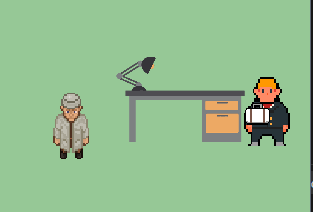
        briefcase = deskbriefcases.pop()

        bossbriefcases.append(briefcase)

    if boss.x>=WIDTH and len(bossbriefcases)==1:

        bossbriefcases.pop()

        bosscollect = False

Changes to health degradation and coffee animation:

I had previously used user events to judge health degradation and coffee animation, however this proved to be very inefficient. Instead, I used the clock cycle to decrease health and Boolean values to judge when either event occurred.

if event.type == COFFEEDELAY:

                healthdegrade = True

                coffee = False

                dt = clock.get\_time()

                if healthTimer.delay(dt,2000):

                    startHealth = 0

            if  event.type!=COFFEEDELAY and officeWorker.colliderect(coffeeStation):

                if health >= 100:

                    coffee = False

                    health = 100

                elif startHealth<20:

                    healthdegrade = False

                    coffee = True

                    health +=1

                    startHealth+=1

                elif startHealth >=20:

                    coffee = False

                    pygame.event.post(pygame.event.Event(COFFEEDELAY))

            else:

                healthdegrade = True

                coffee = False

#outside of for events in pygame.event.get()

if healthdegrade == True:

            health-=0.025

Adding a main menu

In order to make this game feel more user friendly, I want to add a main menu with options that the user can select. First I am creating a separate file called main menu. Then, by using w and s, the user can scroll through the different options.

def mainMenu():

    WIN.fill(WHITE)

    drawText("Main Menu",mainMenuFont,BLACK,WIN,WIDTH//2,30,True,False)

    button1 = drawText("PLAY GAME",buttonsFont,BLACK,WIN,WIDTH//2,200,True,True)

    button2 = drawText("CREDITS",buttonsFont,BLACK,WIN,WIDTH//2,300,True,True)

    button3 = drawText("QUIT",buttonsFont,BLACK,WIN,WIDTH//2,400,True,True)

    buttons = [button1,button2,button3]

    currbutton = 0

    arrow = pygame.Rect(button1.x-50,button1.y,50,50)

    while True:

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

            if event.type == pygame.KEYDOWN:

                if event.key == pygame.K\_w:

                    currbutton-=1

                    if currbutton<0:

                        currbutton+=1

                    arrow.y = buttons[currbutton].y

                if event.key == pygame.K\_s:

                    currbutton +=1

                    if currbutton>2:

                        currbutton-=1

                    arrow.y = buttons[currbutton].y

                if event.key == pygame.K\_SPACE:

                    if buttons[currbutton] == button3:

                        pygame.quit()

                        sys.exit()

                    if buttons[currbutton] == button1:

                        main.main()

                    if buttons[currbutton] == button2:

                        creditsFunc()

        #updating display

        WIN.fill(WHITE)

        drawText("Main Menu",mainMenuFont,BLACK,WIN,WIDTH//2,30,True,False)

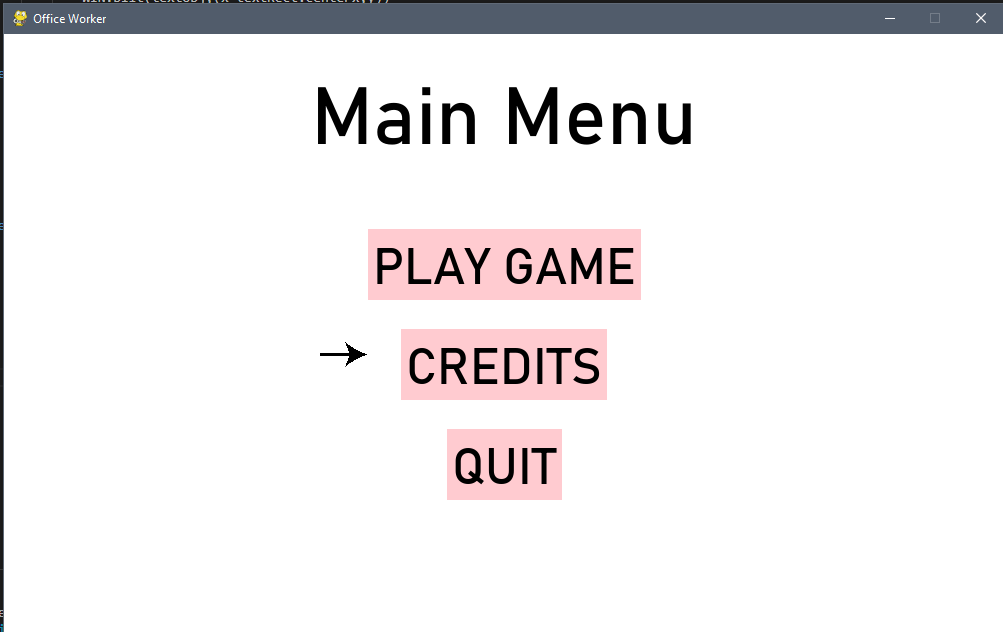
        button1 = drawText("PLAY GAME",buttonsFont,BLACK,WIN,WIDTH//2,200,True,True)

        button2 = drawText("CREDITS",buttonsFont,BLACK,WIN,WIDTH//2,300,True,True)

        button3 = drawText("QUIT",buttonsFont,BLACK,WIN,WIDTH//2,400,True,True)

        WIN.blit(arrowImg,(arrow.x,arrow.y))

        pygame.display.update()



From pressing space, the user can choose which option will run. And using w and s will scroll through the buttons.

For the credits, I added a text file which I could edit in case I wanted to easily add details etc.

Adding end screen:

def gameEnd(score):

    WIN.fill(WHITE)

    drawText("GAME ENDED",mainMenuFont,BLACK,WIN,WIDTH//2,100,True,True)

    drawText("You Scored: {}".format(score),scorefont,BLACK,WIN,WIDTH//2,300,True,False)

    drawText("Press Space to continue to menu...",scorefont,BLACK,WIN,WIDTH//2,500,True,False)

    while True:

        for event in pygame.event.get():

            if event.type == pygame.KEYDOWN:

                if event.key  == pygame.K\_SPACE:

                    mainMenu.mainMenu()

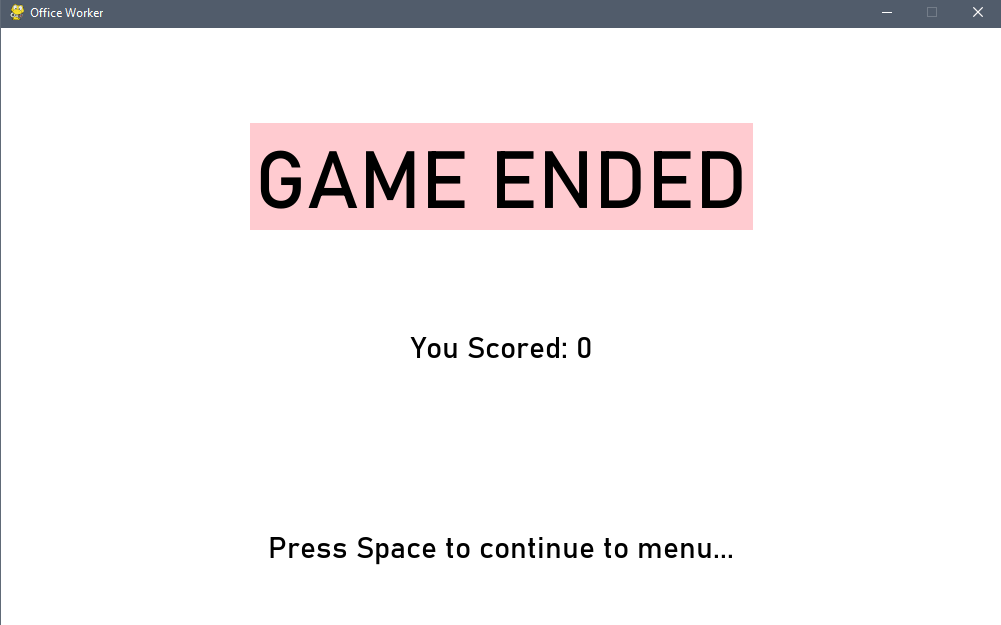
            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

        pygame.display.update()

I wanted to add an end screen after the player dies. I did this through a function within the main file, which would run when the health was below 0. It would simply display the score and give the user the option to go back to the main menu.

Adding an input box for the name:

To add an input box I used a class object, which had methods to update the text box displayed and to return whatever name was entered as a variable.

class InputBox:

    def \_\_init\_\_(self,x,y,w,h,text = ''):

        self.rect = pygame.Rect(x,y,w,h)

        self.color = PINK

        self.text = text

        self.txt\_surface = scorefont.render(text,True,self.color)

    def handleEvent(self,event):

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_RETURN:

                print(self.text)

                return self.text

            elif event.key == pygame.K\_BACKSPACE:

                self.text = self.text[:-1]

            else:

                self.text += event.unicode

            self.txt\_surface = scorefont.render(self.text,True,BLACK)

    def update(self):

        width = max(200,self.txt\_surface.get\_width()+10)

        self.rect.w = width

    def draw(self,WIN):

        pygame.draw.rect(WIN,self.color,(self.rect.x,self.rect.y,self.rect.w,self.rect.h))

        WIN.blit(self.txt\_surface,(self.rect.x+5,self.rect.y+5))

Within my gameEnd func, I initialised an object of type InputBox, which acted as my name input box whenever the game ended.

  inputName = InputBox(400,400,100,100)

name = inputName.handleEvent(event)

            inputName.update()

            inputName.draw(WIN)

            if name:

                writefile(name,score)

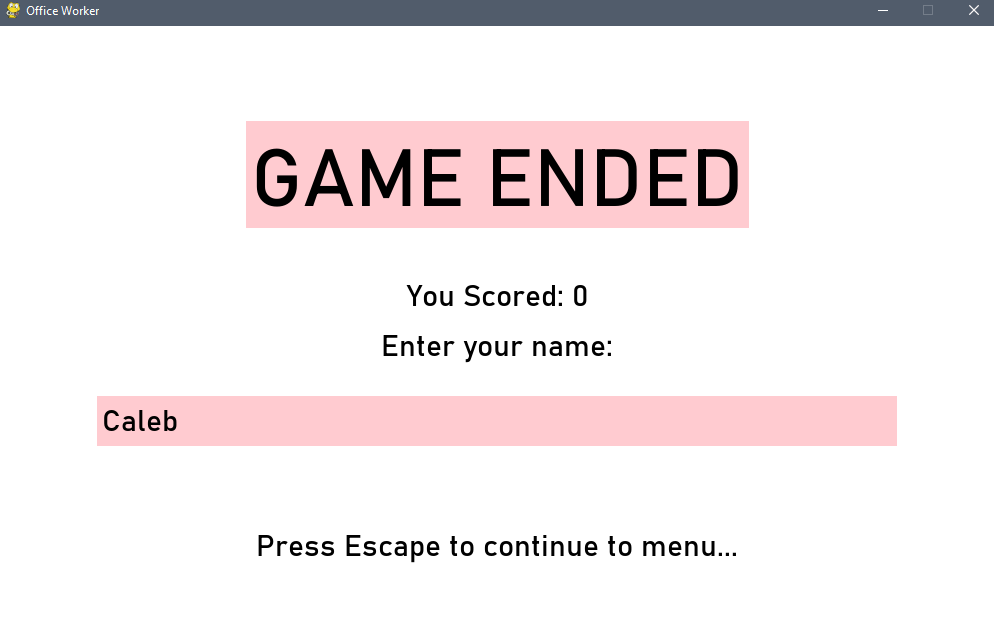
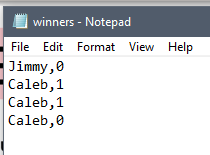
                mainMenu.mainMenu()

def writefile(name,score):

    f = open('winners.txt','a+')

    f.write("{},{}\n".format(name,score))

    f.close()

Then when the user entered the name, the name and their score would be written to a .txt file and the program returns to the main menu.

Adding a leader board:

Through the external .txt file, I can display the top 5 players with the top 5 scores in the main menu. I did this through a simple dictionary and sorting though the values to display the top 5 names and scores.

def leaderboard():

    myDict = {}

    winners = readfile("winners.txt")

    winners = winners.split("\n")

    winners.remove("")

    for winner in winners:

        winner = winner.split(",")

        myDict[winner[0]] = int(winner[1])

    myDict = dict(sorted(myDict.items(), key=lambda item: item[1],reverse = True))

    drawText("Top 5 Players:",subtitleFont,BLACK,WIN,50,175,False,True)

    for i in range (5):

        name = list(myDict)[i]

        score = myDict[name]

        temp = "{}, {}".format(name, str(score))

        drawText(temp,subtitleFont,BLACK,WIN,50,225+i\*50,False,True)

