COLLEGE OF ENGINERING AND COMPUTER STUDIES

PERFORMANCE TASK 4
Introduction to Game Design

Complete the Python Game Farm Invaders

Submitted By

John Brent David

Course & Section

BSCS 3-1

Date
December 15, 2022

TABLE OF CONTENTS

I. DESCRIPTION

A. Problem Statement

In developing a game, a developer must finish the game with complete sprites and functionalities. In this performance task, the game that will be documented will have a complete game sprites and functionalities.

B. Objectives

- Spawn your enemy (set the minimum)
- Put Leveling (at least two levels in your game)
- Additional object/sprite that can reduce player health (i.e. flying object)
- Player power capability (i.e. add bullet power)

II. GAME DESIGN (Mechanics/Characters)

This game version is still not a final game version. Meaning that some game features will be subject to change in the future.

For the **walking** mechanic, W,A,S,D keys will be used to make the character position go up, left down, right respectively. While the **jumping** mechanic will use spacebar to make the character move vertically in an instant and go back immediately, consecutive jumps is not allowed.

Attacking mechanic will be assigned to left click of the mouse button to release a projectile that can hit and damage the enemy. **Collision** mechanic is used for the projectile and the enemy.

Health/Life mechanic is used for both the player and the enemy. Life is for the player; this acts as a limit on how many enemies the player could let before losing. Health is used for the enemies to indicate how many more hits it could take before the enemy would die. Player also has health which will be deducted when it collided with an enemy.

Scoring mechanic is with the help of the health indicator of the enemy. If the enemy died (no health remaining), then the player will gain 1 point. Every enemy will be subjected for the health and scoring mechanic.

III. GAME FLOW/RULES

Final Game Version

The game will consist of game rules to create a challenging feature to the game. The following are the game flow/rules currently present at the final version of this game:

• The player will be spawned on the left part of the game window.
• The player can use W,A,S,D keys to move but if they are near the edge of the game window they will not be able to move towards that particular direction.
• The character will always face the right part of the game window if it is not moving(default position).
• The character can only move on one direction at a time. It wouldn't move diagonally.
• The character cannot jump while hitting and vice versa. The same thing also happens with walking and attacking. The character prioritizes walking instead of attacking.
• An enemy will be spawning on the right part of the game window and will move towards the leftmost part of the game window.
• A character will prioritize attacking instead of moving if clicked simultaneously.
• If the bullet projectile shot by the player hits, the enemy will respawn at the rightmost part of the game window. Continuing to reach the leftmost part of the screen.
• Enemy health will be randomized, and the upper limit will be increased based on the score progression (10,20,30,40).
• Enemy speed will be inversely proportional to the enemy's health. More health means slower and vice versa.
• The player will have 3 lives indicated by the heart image/s on the top leftmost part of the game window.
• If an enemy reaches the leftmost part of the screen the player's lives will be deducted 1 point. If there are no more lives left and an enemy reaches the leftmost border, then it is GAME OVER.
•Player to enemy collision is added and if they had collided reduce the player's health by 1 with total of 10 health.
•If the player has no health remaining reduce the total life to 0 leading to the player ending the game.
•If the player and enemy collided reset the position of the enemy to the starting point.

IV. RESULTS AND DISCUSSION (Game UI/UX with brief descriptions)

Enemies Spawning

Fig 1.1 Spawned Enemies

Figure 1.1 features two enemies spawned facing the left part of the game window. The enemy starting position is just outside of the right side of the game window. They would approach the right part of the screen with a constant speed based on its randomized initial speed.

Fig 1.2 Dead Enemy Sprites

An improvement for the spawned enemies is the addition of the death animation. Which includes 13 total sprites figure 1.2 shows the 9 of those animations. This animation will be drawn at the position where the enemy's hp became less than or equal to 0.

Leveling Difficulty

```
def repos(self,score):
    self.x = 1280
    self.y = random.randint(10,444)
    if score<10:#5 vel for this hp
        self.health =2
    elif score>=10 and score<20:#4 vel for this hp
        self.health = random.randint(2,4)
        self.vel= self.speed(self.health)
    elif score>=20 and score<30:#3 vel for this hp
        self.health = random.randint(2,6)
        self.vel= self.speed(self.health)
    elif score >= 30 and score<40:#2 vel for this hp
        self.health = random.randint(2,8)
        self.vel= self.speed(self.health)
    elif score >= 40:#1 vel for this hp
        self.health = random.randint(2,10)
        self.vel= self.speed(self.health)
```

Fig 2.1 Enemy health based on level

```
def speed(self,health):
    if health ==3 or health ==4:
        return 4
    elif health ==5 or health ==6:
        return 3
    elif health ==7 or health ==8:
        return 2
    elif health ==9 or health ==10:
        return 1
    return 5
```

Fig 2.2 Enemy's speed based on its health

According to figure 2.1, depending on the score, the enemy's health will have a potential to have higher hp indicated by the upper limit of the *"random.randint()"* function. The maximum health points '10' will be available if the player's score reaches 40. On figure 2.2 it was indicated that based on the enemy's health its speed will be affected. Speed and health relation will be inversely proportional to each other. This relation was utilized to improve the overall game balance.

Player Power Capability

```
#Game Balancing
if score<10:
    if(random.randint(0,1000)==9):
        buffs.append(buff(random.r
    bulctr =0
elif score>=10 and score<20:
    if(random.randint(0,800)==9):
        buffs.append(buff(random.r
    bulctr =1
elif score>=20 and score<30:
    if(random.randint(0,600)==9):
        buffs.append(buff(random.r
    bulctr =2
elif score >= 30 and score<40:
    if(random.randint(0,400)==9):
        buffs.append(buff(random.r
    bulctr =3
elif score >= 40:#This is the buff
    if(random.randint(0,200)==9):
        buffs.append(buff(random.r
```

Fig 3.1 Player Power

Figure 3.2 Double Damage Buff

With relation to game balance, figure 3.1 shows the potential player upgrades if the score reaches a certain value which will be similar values with the enemy conditionals. The use of power ups would help the player challenge tankier (more health) enemies as the game progresses. There are two player power ups included in this game version. First would be the amount of bullets will be increased every 10^{th} score with the max value of 3. Second, is the introduction of double damage buff as seen in figure 3.2.

Double damage buff will be dropped at a chance using random number generator. Figure 3.1 shows that the drop chance would increase as the score progresses. The buff after reaching its assigned y-position will then disappear after 5 seconds of not picking it up. The double damage buff after picking up would last for total of 10 seconds. The timer would reset if a player would pick up another double damage buff while the double damage is still active.

V. CODES

```
#PT4 - OBJECT COLLISIONS
import pygame
from Player import player #from for file name the import can be the class
name
from Projectile import projectile
from Projectile import buff
from Enemy import enemy
import random

pygame.init()
pygame.mixer.init()
wx = 1280
wy = 576
win = pygame.display.set_mode((wx,wy))#set the size of the window
pygame.display.set_caption("PT4 Game version")#set the title
pygame.display.update()

#IMPORT THE SPRITES AND SOUNDS
bg= pygame.image.load('images/bg3.jpg')
lifeCtr = pygame.image.load('images/heart.png')

#SOUNDS RESOURCE
pygame.mixer.music.load('sounds/bgm.wav')
pygame.mixer.music.set_volume(.1)
pygame.mixer.music.play(-1)

step = pygame.mixer.Sound('sounds/step.mp3')
jump =  pygame.mixer.Sound('sounds/jump.wav')
hit = pygame.mixer.Sound('sounds/hit.wav')
ddamage = pygame.mixer.Sound('sounds/ddamage.wav')
gover = pygame.mixer.Sound('sounds/gover.wav')

jump.set_volume(.10)#.1 for headphones
step.set_volume(.3)#.3 for headphones
ddamage.set_volume(.05)
gover.set_volume(.05)
score = 0
bulctr = 0 #counts the amount of bullets that should be present on the
game window
lives = 3
damage = 1
buff_timer =0
```

```
once = True
mdpx=500 #monster dead position
mdpy = 250
mdp1x= 500 #monster1 dead position
mdp1y = 250

#INSTANTIATION OF THE CLASSES
play = player()#instance of player class
monster = enemy(1280,410,200,162,0)
monster1 = enemy(1280,110,200,162,0)
bullets = []
buffs = []

def redrawWindowGame():
    win.blit(bg, (0,0))#draw the bg at the position indicated
    for buff in buffs:
        buff.draw(win)
    play.draw(win)
    monster.draw(win,mdpx,mdpy)
    monster1.draw(win,mdp1x,mdp1y)
    life_x = 80
    life_y = 10
    for i in range (0,lives):
        win.blit(lifeCtr,(life_x,life_y))
        life_x+=50
    text = font.render('Score : '+str(score),1,(255,255,255))
    text2 = font.render('Lives: ',1,(255,255,255))
    win.blit(text,(1150,40))
    win.blit(text2,(5,20))
    for bullet in bullets:
        bullet.draw(win)
    pygame.display.update()

def drawEndGame():
    win.blit(bg, (0,0))
    play.drawDied(win)
    monster.draw(win,mdpx,mdpy)
    monster1.draw(win,mdpx,mdpy)
    creds = fontTitle.render('Game Over!',1,(255,0,0))
    fnlScore = fontTitle.render('Your total kills :
'+str(score),1,(255,0,0))
    win.blit(creds,(550,260))
    win.blit(fnlScore,(475,310))
    pygame.display.update()
```

```
clock = pygame.time.Clock()
run = True
font = pygame.font.Font('fontstyles/SummerPixel22Regular-jE0W7.ttf',25)
fontTitle = pygame.font.Font('fontstyles/SummerPixel22Regular-
jE0W7.ttf',40)
#MAIN GAME LOOP
while run:
    clock.tick(45)#frame rate

    #Game Balancing
    if score<10:
        if(random.randint(0,1000)==9):
            buffs.append(buff(random.randint(0+play.width,wx-
play.width),random.randint(0+play.height,wy-play.height)))
        bulctr =0
    elif score>=10 and score<20:
        if(random.randint(0,800)==9):
            buffs.append(buff(random.randint(0+play.width,wx-
play.width),random.randint(0+play.height,wy-play.height)))
        bulctr =1
    elif score>=20 and score<30:
        if(random.randint(0,600)==9):
            buffs.append(buff(random.randint(0+play.width,wx-
play.width),random.randint(0+play.height,wy-play.height)))
        bulctr =2
    elif score >= 30 and score<40:
        if(random.randint(0,400)==9):
            buffs.append(buff(random.randint(0+play.width,wx-
play.width),random.randint(0+play.height,wy-play.height)))
        bulctr =3
    elif score >= 40:#This is the buff for 40+ score line randomly drops a
double damage buff on the window that a player can pick up
        if(random.randint(0,200)==9):
            buffs.append(buff(random.randint(0+play.width,wx-
play.width),random.randint(0+play.height,wy-play.height)))
    if buff_timer >=450:#10 seconds
        damage =1
        buff_timer = 0
        ddamage.stop()
    #Game Player Lives counter
    if(monster.x >= -199 and monster.x <= -150):
        lives -=1
        monster.x =1280
```

```
    if monster1.x >= -199 and monster1.x <= -150:
        lives-=1
        monster1.x= 1280
    for event in pygame.event.get():
        if event.type == pygame.QUIT:#conditional to stop the game if X is
pressed
            run = False

    #check for bullet and monster collision
    for bullet in bullets:
        if bullet.x+10 >= monster.hitObject[0] and bullet.x+10 <=
monster.hitObject[0] + monster.hitObject[2]:
            if bullet.y+10 >= monster.hitObject[1] and bullet.y+10 <=
monster.hitObject[1] + monster.hitObject[3]:
                bullets.pop(bullets.index(bullet))
                if monster.hit(score,damage):
                    score+=1
                    mdpx = monster.deadx
                    mdpy = monster.deady
                    monster.dead= True
                if monster.health <=0:#removes the scoring bug double
checks the health of the enemy
                    score+=1
                    mdpx = monster.x
                    mdpy = monster.y
                    monster.dead = True
                    monster.repos(score)
        if bullet.x+10 >= monster1.hitObject[0] and bullet.x+10 <=
monster1.hitObject[0] + monster1.hitObject[2]:
            if bullet.y+10 >= monster1.hitObject[1] and bullet.y+10 <=
monster1.hitObject[1] + monster1.hitObject[3]:
                bullets.pop(bullets.index(bullet))
                if monster1.hit(score,damage):
                    score+=1
                    mdp1x = monster1.deadx
                    mdp1y = monster1.deady
                    monster1.dead= True
                if monster1.health <=0:
                    score+=1
                    mdp1x = monster1.x
                    mdp1y = monster1.y
                    monster1.dead = True
                    monster1.repos(score)
```

```
        if bullet.x < wx and bullet.x > 0:#checks if the bullet is still
on the window
            bullet.x += bullet.vel
        else:#if the bullet is not the remove it from the array
            bullets.pop(bullets.index(bullet))

    #checks for player and buff collision
    for buf in buffs:
        if buf.x+14 >= play.hitObject[0] and buf.x+14 <= play.hitObject[0]
+ play.hitObject[2]:
            if buf.init_y+25 >= play.hitObject[1] and buf.init_y+25 <=
play.hitObject[1] + play.hitObject[3]:
                damage = 2
                buff_timer = 0
                ddamage.stop()
                ddamage.play(0)
                buffs.pop(buffs.index(buf))
        if buf.pop:#checks if the bullet expires
            buffs.pop(buffs.index(buf))
    buff_timer += 1
    keys = pygame.key.get_pressed()
    mkeys = pygame.mouse.get_pressed()

#2nd condition sets the boundaries where u can only walk
    if lives >0:
        if(mkeys[0] and not play.isJump and len(bullets) <=bulctr):
            if (play.hsoundCount == 33):
                bullets.append(projectile((play.x +
play.width//2),round(play.y+play.height//2),6,(0,0,0),facing))
                hit.play(0)
                play.hsoundCount = 0
                play.a = False
            else:
                play.hsoundCount +=1
            play.l = False
            play.r = False
            play.a = True
            facing = 1
        elif (keys[pygame.K_a] and play.x> play.speed ):
            if(play.ssoundCount ==0):
                step.play(0)
                play.ssoundCount = 30
            else:
                play.ssoundCount -=1
```

```
            play.x-=play.speed
            play.l = True
            play.r = False
        elif (keys[pygame.K_s]and play.y < wy - play.speed- play.height+37
and not play.isJump): #not isJump is added as a condition to avoid getting
out of bounds
            if(play.ssoundCount ==0):
                step.play(0)
                play.ssoundCount = 30
            else:
                play.ssoundCount -=1
            play.y+=play.speed
            play.l = False
            play.r = True
        elif (keys[pygame.K_w]and play.y > 10 and not play.isJump):#370 is
the initial val of sprite
            if(play.ssoundCount ==0):
                step.play(0)
                play.ssoundCount = 30
            else:
                play.ssoundCount -=1
            play.y-=play.speed
            play.l = False
            play.r = True
        elif (keys[pygame.K_d] and play.x< wx - play.speed - play.width ):
            if(play.ssoundCount ==0):
                step.play(0)
                play.ssoundCount = 30
            else:
                play.ssoundCount -=1
            play.x += play.speed
            play.l = False
            play.r = True
    #if the character is not moving
        else:
            play.l = False
            play.r = False
            play.a = False
            play.walk_counter =0
            play.ssoundCount = 0
            play.hsoundCount = 0
#CONDITIONAL FOR JUMPING
        if not(play.isJump):
            if (keys[pygame.K_SPACE]):
```

```
                play.isJump = True
                play.right = False
                play.Left = False
                play.walk_counter =0
        else:
            if(play.jumpCount >= -11):
                if(play.jumpCount == 11):#condition to check if code
should play the sound
                    if (play.jsoundCount ==30):
                        jump.play(0)
                        play.jsoundCount = 0
                play.y -= (play.jumpCount *abs(play.jumpCount)) *.5
                play.jumpCount -=1
            else:
                play.jsoundCount = 30
                play.jumpCount = 11
                play.isJump = False
        #CHECKS for character collisions between enemy and player
        if play.hitObject[0] >= monster.hitObject[0] and play.hitObject[0]
<= monster.hitObject[0]+monster.hitObject[2]:#check the left of the player
            if play.hitObject[1] >= monster.hitObject[1] and
play.hitObject[1] <= monster.hitObject[1]+monster.hitObject[3]:
                monster.x = 1280
                play.health -= 1
            elif play.hitObject[1]+play.hitObject[3] >=
monster.hitObject[1] and play.hitObject[1]+play.hitObject[3] <=
monster.hitObject[1]+monster.hitObject[3]:
                monster.x = 1280
        elif play.hitObject[0]+play.hitObject[2] >= monster.hitObject[0]
and play.hitObject[0]+play.hitObject[2] <=
monster.hitObject[0]+monster.hitObject[2]:
            if play.hitObject[1] >= monster.hitObject[1] and
play.hitObject[1] <= monster.hitObject[1]+monster.hitObject[3]:
                monster.x = 1280
                play.health -= 1
            elif play.hitObject[1]+play.hitObject[3] >=
monster.hitObject[1] and play.hitObject[1]+play.hitObject[3] <=
monster.hitObject[1]+monster.hitObject[3]:
                monster.x = 1280
                play.health -= 1
        if play.hitObject[0] >= monster1.hitObject[0] and
play.hitObject[0] <= monster1.hitObject[0]+monster1.hitObject[2]:#check
the left of the player
```

```
            if play.hitObject[1] >= monster1.hitObject[1] and
play.hitObject[1] <= monster1.hitObject[1]+monster1.hitObject[3]:
                monster1.x = 1280
                play.health -= 1
            elif play.hitObject[1]+play.hitObject[3] >=
monster1.hitObject[1] and play.hitObject[1]+play.hitObject[3] <=
monster1.hitObject[1]+monster1.hitObject[3]:
                monster1.x = 1280
                play.health -= 1
        elif play.hitObject[0]+play.hitObject[2] >= monster1.hitObject[0]
and play.hitObject[0]+play.hitObject[2] <=
monster1.hitObject[0]+monster1.hitObject[2]:
            if play.hitObject[1] >= monster1.hitObject[1] and
play.hitObject[1] <= monster1.hitObject[1]+monster1.hitObject[3]:
                monster1.x = 1280
                play.health -= 1
            elif play.hitObject[1]+play.hitObject[3] >=
monster1.hitObject[1] and play.hitObject[1]+play.hitObject[3] <=
monster1.hitObject[1]+monster1.hitObject[3]:
                monster1.x = 1280
                play.health -= 1

        if play.health == 0:
            play.health =10
            lives =0

    if lives <= 0:
        drawEndGame()
        pygame.mixer.music.set_volume(0)
        if once:
            ddamage.stop()
            gover.play(-1)
            once = False
    else:
        redrawWindowGame()

    #end of the MAIN LOOP

pygame.quit()
```

VI. LEARNING OUTCOMES

I have learned that in game development, there are many factors that a developer might consider. One of the difficulties that I personally had encountered was one with the sprites. As a game developer they should make sure that the assets should have no copyright issues. These includes sprites and sounds. Animation is one of the most interesting part of the games which gives excitement to the players. Meaning that the more animations the better. Sound should be in line with the game theme to increase game immersion.

The thing that I focused on the game is the game balance. I have realized that a good game balance will improve player retention in playing the game as they progresses. Of course, no one would continue to play the game if he/she would realize that the game would be too easy as they progress. It would also be the same if the game would be too difficult as the player progresses. A proper game balance would give a better player progression which leads to a better player experience overall.

There are many more things that is not about animations and game balance that should be considered. Quality of life changes for players should also be in the list to consider when updating a game. The most important factor in a game is how a player would feel about the totality of the game that a game developer must consider.

VII. REFERENCES

For the free Sprites that is used in the game
https://craftpix.net/freebies/free-satyr-tiny-style-2d-sprites/?num=1&count=2&sq=satyr&pos=0
For free music that is used in the game
https://pixabay.com/sound-effects/
Github Link
CSEL2L/PT4 code at master · Eusugii/CSEL2L (github.com)
YouTube Video