SPACE INVADERS GAME USING PYTHON

Created By:- Sanket Prashant Marathe

Seat No.:- 1906542

Class:- SYBSC IT

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide Ms. Leena Shewale for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project.

I would like to thank my College Library, for having provided various reference books and magazines related to my project.

Lastly, I would like to thank each and every person who directly or indirectly helped me in the completion of the project especially my Parents and Peers who supported me throughout my project.

INDEX

SYNOPSIS

Introduction

Space Invaders is a 2-D fixed shooting game. Space Invaders is an arcade video game created by Tomohiro Nishikado and released in 1978. It was originally manufactured and sold by Taito in Japan, and was later licensed for production in the United States by the Midway division of Bally. Space Invaders is one of the earliest shooting games and the aim is to defeat waves of aliens with a laser cannon to earn as many points as possible. In designing the game, Nishikado drew inspiration from popular media: Breakout, The War of the Worlds, and Star Wars. To complete it, he had to design custom hardware and development tools. It was one of the forerunners of modern video gaming and helped expand the video game industry from a novelty to a global industry. When first released, Space Invaders was very successful.

Requirements

Software Requirements

Operating Systems:- Windows, Linux and Mac OS

Application Software:- Python IDLE, PyCharm(Optional)

Language:- Python(version 3.8)

Hardware Requirements

Hard Disk:- 32GB

RAM:- 128MB

Processor:- Any intel core version

Modules (Used in Code)

1. PyGame
2. Sys
3. Random

GAMEPLAY

Space Invaders is a fixed shooter in which the player controls the laser cannon by moving it horizontally across the bottom of the screen and firing at descending aliens. The aim is to defeat five rows of eleven aliens—although some versions feature different numbers—that move horizontally back and forth across the screen as they advance toward the bottom of the screen. The player's laser cannon is partially protected by several stationary defence bunkers—the number also varies by version—that are gradually destroyed from the top and bottom by blasts from either the aliens or the player.

The player defeats an alien and earns points by shooting it with the laser cannon. As more aliens are defeated, the aliens' movement and the game's music both speed up. Defeating all the aliens on-screen brings another wave that is more difficult, a loop which can continue endlessly. A special "mystery ship" will occasionally move across the top of the screen and award bonus points if destroyed.

The aliens attempt to destroy the player's cannon by firing at it while they approach the bottom of the screen. If they reach the bottom, the alien invasion is declared successful and the game ends tragically; otherwise, it ends generally if the player's last cannon is destroyed by the enemy's projectiles.

RULES

The basic rules of the game are as follow:

1. The Player can only move along the X-Axis on the ground.
2. The player will fire bullets at the Aliens. These bullets have limited velocity and limited rate of fire, both of which may vary during the game play.
3. The player has an unlimited amount of bullets.
4. If a bullet hits an Alien, the Alien will die.
5. The bullets cannot travel through the shields.
6. The Player will receive certain point values for killing each Alien.
7. The shields can be destroyed both by the Aliens bombs and Players Missiles.
8. The Aliens will also move along the X-Axis.
9. The Aliens shall drop down the Y-Axis closer to the Player as time passes.
10. The Aliens shall drop bullets on the Player.
11. If the Aliens reach the ground or Players live is 0, the Player shall lose.

CODE

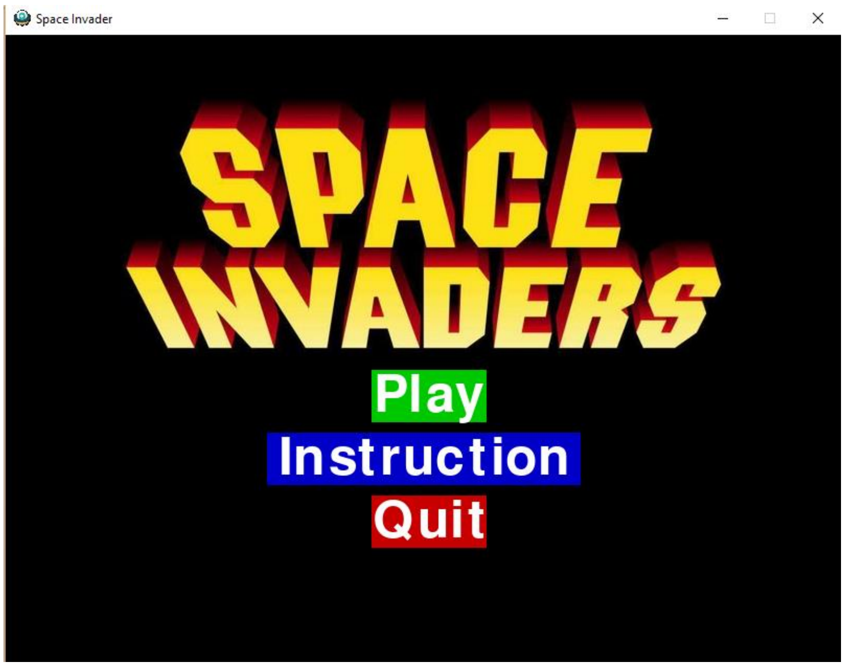
```
import pygame
from pygame.locals import *
import sys
import random
from pygame import mixer


class SpaceInvaders:
   def __init__(self):

        # Intialization
        pygame.mixer.init()
        self.score = 0
        self.lives = 2
        self.string=""
        pygame.font.init()
        self.font = pygame.font.Font("assets/space_invaders.ttf", 15)
        self.font1 = pygame.font.Font('freesansbold.ttf', 50)

        # Barrier Design
        barrierDesign =  [[],[0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
                             [0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0],
                             [0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0],
                             [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],
                             [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
                             [1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1],
                             [1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1],
                             [1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1],
                             [1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1],
                             [1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1]]

        # Screen Size
        self.screen = pygame.display.set_mode((800, 600))
        pygame.display.set_caption("Space Invader")

        # Icon
        icon = pygame.image.load('assets/ufo.png')
        pygame.display.set_icon(icon)
        self.enemySprites = {
            0:[pygame.image.load("assets/a1_0.png").convert(),
pygame.image.load("assets/a1_1.png").convert()],
            1:[pygame.image.load("assets/a2_0.png").convert(),
pygame.image.load("assets/a2_1.png").convert()],
            2:[pygame.image.load("assets/a3_0.png").convert(),
pygame.image.load("assets/a3_1.png").convert()],
            }
        self.player = pygame.image.load("assets/shooter.png").convert()
        self.title = pygame.image.load("assets/title.jpg")
        self.arrow_left = pygame.image.load("assets/arrow_left.png")
        self.arrow_right = pygame.image.load("assets/arrow_right.png")
        self.spacebar = pygame.image.load("assets/spacebar.png")
        self.animationOn = 0
        self.direction = 1
        self.enemySpeed = 20
        self.lastEnemyMove = 0
        self.playerX = 400
        self.playerY = 550
        self.bullet = None
        self.bullets = []
        self.enemies = []
        self.barrierParticles = []
        self.textSurface = None
        startY = 50
        startX = 50
        for rows in range(6):
            out = []
            if rows < 2:
                enemy = 0
            elif rows < 4:
                enemy = 1
            else:
                enemy = 2
            for columns in range(10):
                out.append((enemy,pygame.Rect(startX * columns, startY * rows, 35, 35)))
            self.enemies.append(out)
        self.chance = 990

        barrierX = 50
        barrierY = 400
        space = 100

        # Barrier Positions
        for offset in range(1, 5):
            for b in barrierDesign:
                for b in b:
                    if b != 0:
                        self.barrierParticles.append(pygame.Rect(barrierX + space * offset, barrierY,
5,5))
                barrierX += 5
            barrierX = 50 * offset
            barrierY += 3
        barrierY = 400

    # Enemy Update(After bullet hits the Enemy)
    def enemyUpdate(self):
        if not self.lastEnemyMove:
            for enemy in self.enemies:
                for enemy in enemy:
                    enemy = enemy[1]

                    # Enemy Bullet hits Player
                    if enemy.colliderect(pygame.Rect(self.playerX, self.playerY,
self.player.get_width(), self.player.get_height())):
                        self.lives -= 1
                        self.resetPlayer()
                    enemy.x += self.enemySpeed * self.direction
                    self.lastEnemyMove = 25

                    # Enemy Boundary
                    if enemy.x >= 750 or enemy.x <= 0:
                        self.moveEnemiesDown()
                        self.direction *= -1

                    chance = random.randint(0, 1000)

                    # Enemy Shoots Bullet
                    if chance > self.chance:
                        self.bullets.append(pygame.Rect(enemy.x, enemy.y, 5, 10))
                        self.score += 5
            if self.animationOn:
                self.animationOn -= 1
            else:
                self.animationOn += 1
        else:
            self.lastEnemyMove -= 1

    # Movement of Enemies
    def moveEnemiesDown(self):
        for enemy in self.enemies:
            for enemy in enemy:
                enemy = enemy[1]
                enemy.y += 20

    # PLayer's Movements
    def playerUpdate(self):
        key = pygame.key.get_pressed()
        if key[K_RIGHT] and self.playerX < 800 - self.player.get_width():
            self.playerX += 5
        elif key[K_LEFT] and self.playerX > 0:
            self.playerX -= 5
        if key[K_SPACE] and not self.bullet:

            # Bullet Sound
            bulletSound = mixer.Sound("assets/laser.wav")
            bulletSound.play()
            self.bullet = pygame.Rect(self.playerX + self.player.get_width() / 2- 2, self.playerY -
15, 5, 10)

    # Bullets Creation
    def bulletUpdate(self):

        # Player Bullet hits Enemy
        for i, enemy in enumerate(self.enemies):
            for j, enemy in enumerate(enemy):
                enemy = enemy[1]
                if self.bullet and enemy.colliderect(self.bullet):

                    # Enemy Collision Sound
                    explosionSound = mixer.Sound("assets/explosion.wav")
                    explosionSound.play()
                    self.enemies[i].pop(j)
                    self.bullet = None
                    self.chance -= 1
                    self.score += 100

        # Enemy Bullet Speed
        if self.bullet:
            self.bullet.y -= 20
            if self.bullet.y < 0:
                self.bullet = None

        for x in self.bullets:
            x.y += 20

            # Bullet Boundary
            if x.y > 600:
                self.bullets.remove(x)

            # Bullet hits Player
            if x.colliderect(pygame.Rect(self.playerX, self.playerY, self.player.get_width(),
self.player.get_height())):
                self.lives -= 1
                self.bullets.remove(x)
                self.resetPlayer()

        # Bullets Hits Barriers
        for b in self.barrierParticles:
```

```
        check = b.collidelist(self.bullets)
        if check != -1:
            self.barrierParticles.remove(b)
            self.bullets.pop(check)
            self.score += 10
        elif self.bullet and b.colliderect(self.bullet):
            self.barrierParticles.remove(b)
            self.bullet = None
            self.score += 10

# Reassign the position Player after been hit by Enemy's bullet
def resetPlayer(self):
    self.playerX = 400

# For Display Buttons on Screen
def button(self,msg,x,y,w,h,ic,ac,action):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    if x+w > mouse[0] > x and y+h > mouse[1] > y:
        pygame.draw.rect(self.screen,ac,(x,y,w,h))

        if click[0] == 1 and action != None:
            action()
    else:
        pygame.draw.rect(self.screen,ic,(x,y,w,h))

    TextSurf = self.font1.render(msg, True, (255, 255, 255))
    TextRect = TextSurf.get_rect()
    TextRect.center = ((x+(w/2)),(y+(h/2)))
    self.screen.blit(TextSurf,TextRect)

# Main Menu
def game_intro(self):
    clock = pygame.time.Clock()
    blue = (0,0,200)
    bright_blue = (0,0,255)
    green = (0,200,0)
    bright_green = (0,255,0)
    red = (200,0,0)
    bright_red = (255,0,0)
    while True:

        clock.tick(60)
        self.screen.fill((0, 0, 0))
        for event in pygame.event.get():
            if event.type == QUIT:
                sys.exit()
        self.screen.blit(self.title,(80,0))
        self.button("Play",350,320,110,50,green,bright_green,self.game_loop)
        self.button("Instruction", 250, 380, 300, 50, blue, bright_blue, self.instruction)
```

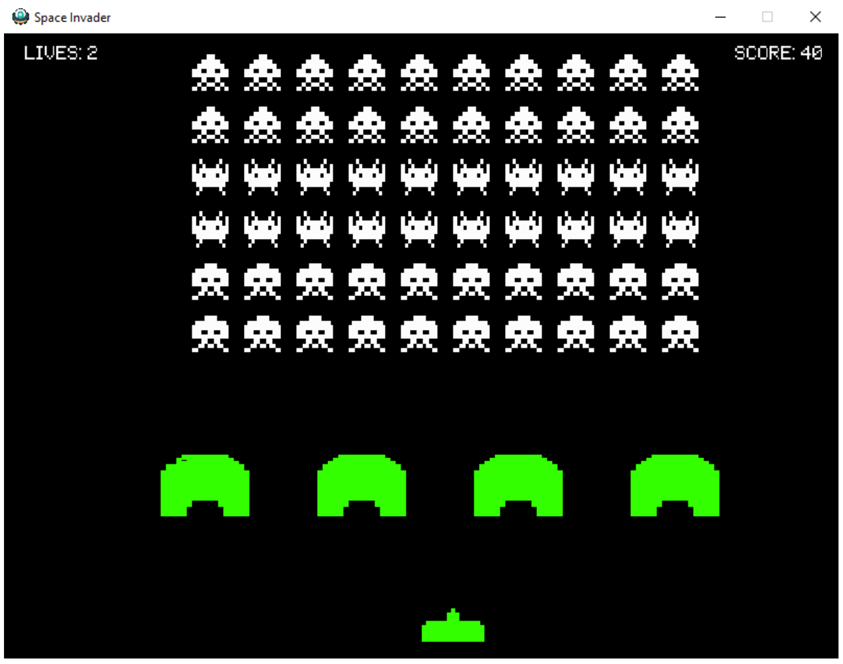
```
            self.button("Quit", 350, 440, 110, 50, red, bright_red, self.gamequit)
            pygame.display.update()

    # Screen after Win or Lose
    def game_end_screen(self):
        clock = pygame.time.Clock()
        green = (0,200,0)
        bright_green = (0,255,0)
        red = (200,0,0)
        bright_red = (255,0,0)
        while True:

            clock.tick(60)
            self.screen.fill((0, 0, 0))
            for event in pygame.event.get():
                if event.type == QUIT:
                    sys.exit()

            if self.string == "You Win!!!":
                self.screen.blit(pygame.font.Font("assets/space_invaders.ttf",
100).render(self.string, -1, (52, 255, 0)), (150, 90))
            else:
                self.screen.blit(pygame.font.Font("assets/space_invaders.ttf",
100).render(self.string, -1, (52, 255, 0)), (100, 90))

            self.screen.blit(self.font.render("Score: {}".format(self.score), -1, (255, 255, 255)),
(340, 215))
            self.button("Restart",300,250,190,50,green,bright_green,self.game_loop)
            self.button("Quit", 340, 310, 110, 50, red, bright_red, self.gamequit)
            pygame.display.update()

    # Instruction Menu
    def instruction(self):
        clock = pygame.time.Clock()
        green = (0, 200, 0)
        red = (200, 0, 0)
        bright_red = (255, 0, 0)
        while True:

            clock.tick(60)
            self.screen.fill((0, 0, 0))
            for event in pygame.event.get():
                if event.type == QUIT:
                    sys.exit()

            TextSurf = self.font1.render("INSTRUCTION", True, (255, 255, 255))
            TextRect = TextSurf.get_rect()
            TextRect.center = (400,100)
            self.screen.blit(TextSurf, TextRect)
            pygame.draw.rect(self.screen, green, (235, 255, 155, 50))
            TextSurf = self.font1.render("Move:", True, (255, 255, 255))
            TextRect = TextSurf.get_rect()
            TextRect.center = (312,283)
            self.screen.blit(TextSurf, TextRect)
            self.screen.blit(self.arrow_left,(400,250))
            self.screen.blit(self.arrow_right,(460,250))

            pygame.draw.rect(self.screen, green, (220, 330, 170, 50))
            TextSurf = self.font1.render("Shoot:", True, (255, 255, 255))
            TextRect = TextSurf.get_rect()
            TextRect.center = (305,358)
            self.screen.blit(TextSurf, TextRect)
            self.screen.blit(self.spacebar,(400,328))

            self.button("Back", 340, 490, 125, 50, red, bright_red, self.game_intro)
            pygame.display.update()

    # For Quit Button
    def gamequit(self):
        pygame.quit()
        quit()

    # for Checking Enemy list is empty or not
    def empty(self,seq):
        try:
            return all(map(self.empty, seq))
        except TypeError:
            return False

    # Main Game Loop(Game Play)
    def game_loop(self):
        self.__init__()
        clock = pygame.time.Clock()
        for x in range(1):
            self.moveEnemiesDown()
        while True:

            clock.tick(60)
            self.screen.fill((0, 0, 0))
            for event in pygame.event.get():
                if event.type == QUIT:
                    sys.exit()

            for enemy in self.enemies:
                for enemy in enemy:

self.screen.blit(pygame.transform.scale(self.enemySprites[enemy[0]][self.animationOn], (35,
35)),
                         (enemy[1].x, enemy[1].y))
            self.screen.blit(self.player, (self.playerX, self.playerY))
            if self.bullet:
                pygame.draw.rect(self.screen, (52, 255, 0), self.bullet)
            for bullet in self.bullets:
                pygame.draw.rect(self.screen, (255, 255, 255), bullet)
            for b in self.barrierParticles:
                pygame.draw.rect(self.screen, (52, 255, 0), b)

            if self.empty(self.enemies) :
                self.string ="You Win!!!"
                self.game_end_screen()
            elif self.lives > 0:
                self.bulletUpdate()
                self.enemyUpdate()
                self.playerUpdate()
            elif self.lives == 0:
                self.string ="You Lose!!!"
                self.game_end_screen()
            self.screen.blit(self.font.render("Lives: {}".format(self.lives), -1, (255, 255, 255)),
(20, 10))
            self.screen.blit(self.font.render("Score: {}".format(self.score), -1, (255, 255, 255)),
(680, 10))
            pygame.display.update()

    # For Running the Thread of a class
    def run(self):

        # Background Sound
        mixer.music.load("assets/background.wav")
        mixer.music.play(-2)

        self.game_intro()


if __name__ == "__main__":
    # Actual Execution of Code
    SpaceInvaders().run()
```

OUTPUTS

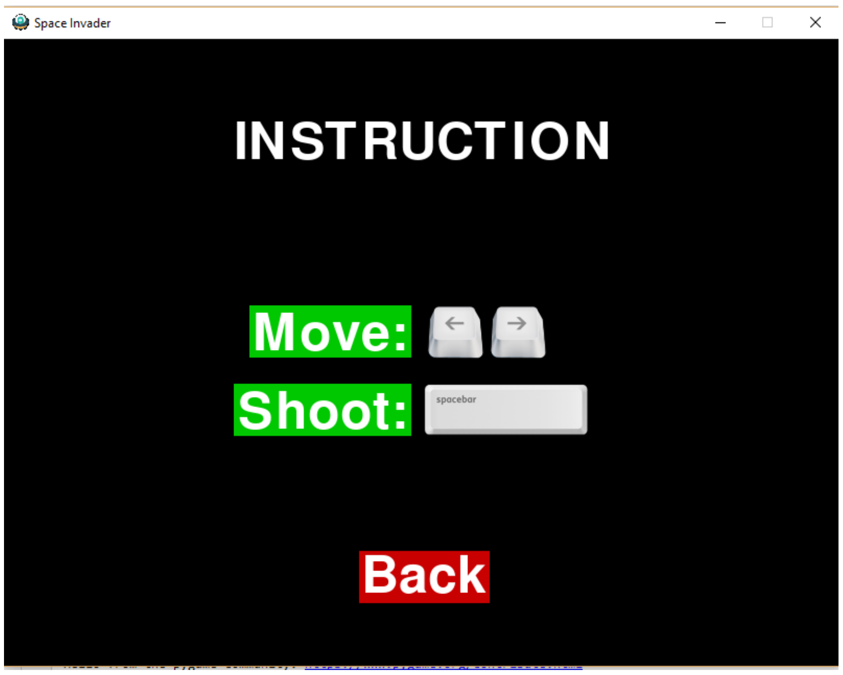
1. Main Menu



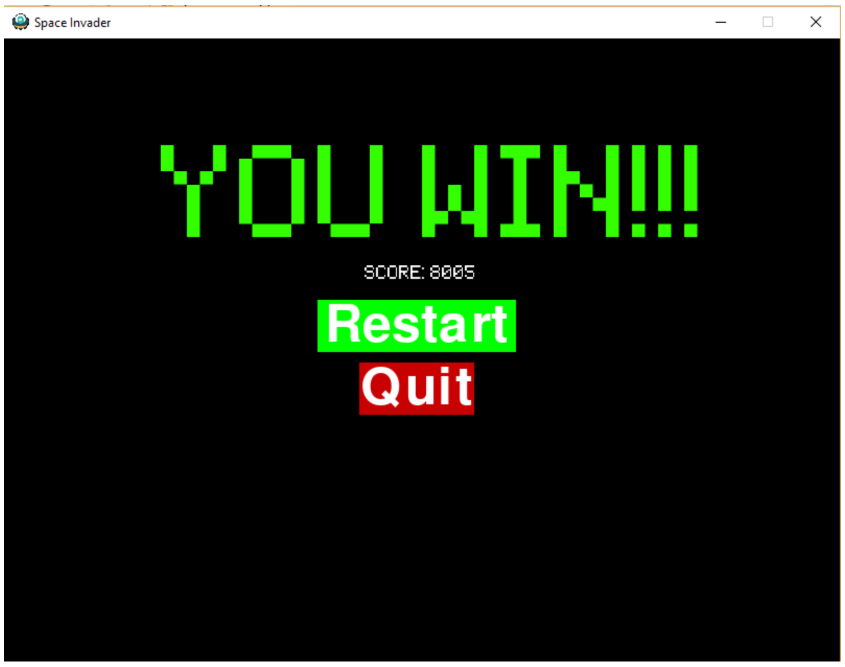
2. GamePlay



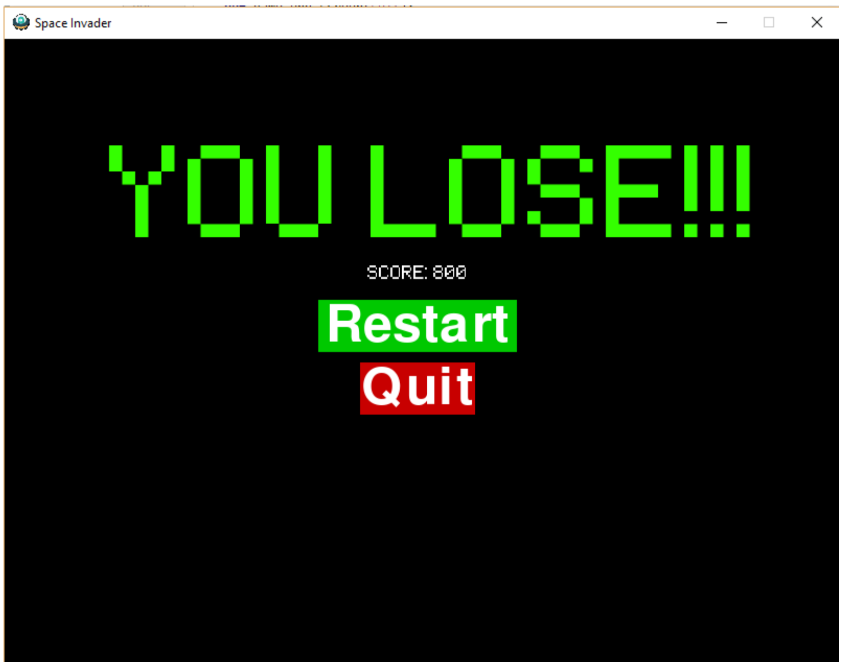
3. Instruction Menu



4. Winning Screen



5. Losing Screen



CONCLUSION

We want to create a game that has a very interesting mix of inputs. In order to ensure that the game is fun, we want the difficulty of the game to scale linearly with the length of time/ skill of the player. We also want to ensure that we have as many enemies on screen as possible by the end. We hope that once the dust settles, we won't have just a barebones game, but a fun and challenging experience that lives up to the name of the classic Space Invaders.

REFERENCES

1. Pygame Documentation :- https://www.pygame.org/docs/
2. Beginning Game Development with Python and Pygame Book by Will McGugan.
3. https://pythonprogramming.net/
4. https://www.python.org