Movie Ticketing System

Mid-Term Project Report

PROGRAMMING IN PYTHON [B]

Submitted by:

Name	ID
OMI, MD. IFAJ HOSSAN	20-42387-1

Submitted to:

DR. AKINUL ISLAM JONY
Assistant Professor
Faculty of Science and Technology
American International University-Bangladesh (AIUB)

Project Overview

This project "**The Falcon Attack**" is a desktop-based graphical interface application. This application is a game. It is a space shooter game. It is meant to be an enjoyable space game. Here the player is portrayed as a spaceship that tries to go through a journey. But along the way, it meets challenges in the form of enemy ships coming to destroy it. The player can go complete the challenges in a level and go to the next level. After completing the levels, the player has managed to complete the game.

Solution Design

This game gives challenge and adventure to the player. The Player must defeat the enemies in order to complete a level. The Player has 3 lives at the start. Various types of enemies come to shoot the spaceship. If the player crashes with an enemy or its bullets, then they lose a life. The levels get progressively harder. This makes the game more challenging and fun. After completing all the levels, the player has completed the game and all the challenges.

Implementation

This game has been implemented by python programming language and one of its Module named Pygame. Pygame is a cross-platform set of Python modules designed for writing video games. Below the code and the screenshots are attached from the application:

Code

```
import pygame
import os
import time
import random
pygame.font.init()

WIDTH, HEIGHT = 1000, 750
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("The Falcon Attack")

BG = pygame.transform.scale(pygame.image.load(
    os.path.join("assets", "background-black.png")), (WIDTH, HEIGHT))
```

```
RED_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_red_small.png"))
GREEN_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_green_small.png"))
BLUE_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_blue_small.png"))


YELLOW_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_yellow.png"))


RED_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_red.png"))
GREEN_LASER = pygame.image.load(
    os.path.join("assets", "pixel_laser_green.png"))
BLUE_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_blue.png"))
YELLOW_LASER = pygame.image.load(
    os.path.join("assets", "pixel_laser_yellow.png"))


class Laser:
    def __init__(self, x, y, img):
        self.x = x
        self.y = y
        self.img = img
        self.mask = pygame.mask.from_surface(self.img)

    def draw(self, window):
        window.blit(self.img, (self.x, self.y))

    def move(self, vel):
        self.y += vel

    def off_screen(self, height):
        return not(self.y <= height and self.y >= 0)

    def collision(self, obj):
        return collide(self, obj)


class Ship:
    COOLDOWN = 30

    def __init__(self, x, y, health=100):
        self.x = x
        self.y = y
```

```
        self.health = health
        self.ship_img = None
        self.laser_img = None
        self.lasers = []
        self.cool_down_counter = 0

    def draw(self, window):
        window.blit(self.ship_img, (self.x, self.y))
        for laser in self.lasers:
            laser.draw(window)

    def move_lasers(self, vel, obj):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            elif laser.collision(obj):
                obj.health -= 10
                self.lasers.remove(laser)

    def cooldown(self):
        if self.cool_down_counter >= self.COOLDOWN:
            self.cool_down_counter = 0
        elif self.cool_down_counter > 0:
            self.cool_down_counter += 1

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1

    def get_width(self):
        return self.ship_img.get_width()

    def get_height(self):
        return self.ship_img.get_height()


class Player(Ship):
    def __init__(self, x, y, health=100):
        super().__init__(x, y, health)
        self.ship_img = YELLOW_SPACE_SHIP
        self.laser_img = YELLOW_LASER
        self.mask = pygame.mask.from_surface(self.ship_img)
        self.max_health = health
```

```
    def move_lasers(self, vel, objs):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            else:
                for obj in objs:
                    if laser.collision(obj):
                        objs.remove(obj)
                        if laser in self.lasers:
                            self.lasers.remove(laser)

    def draw(self, window):
        super().draw(window)
        self.healthbar(window)

    def healthbar(self, window):
        pygame.draw.rect(window, (255, 0, 0), (self.x, self.y +
                         self.ship_img.get_height() + 10, self.ship_img.get_width(),
10))
        pygame.draw.rect(window, (0, 255, 0), (self.x, self.y +
self.ship_img.get_height() +
                         10, self.ship_img.get_width() *
(self.health/self.max_health), 10))


class Enemy(Ship):
    COLOR_MAP = {
        "red": (RED_SPACE_SHIP, RED_LASER),
        "green": (GREEN_SPACE_SHIP, GREEN_LASER),
        "blue": (BLUE_SPACE_SHIP, BLUE_LASER)
    }

    def __init__(self, x, y, color, health=100):
        super().__init__(x, y, health)
        self.ship_img, self.laser_img = self.COLOR_MAP[color]
        self.mask = pygame.mask.from_surface(self.ship_img)

    def move(self, vel):
        self.y += vel

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x-20, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1
```

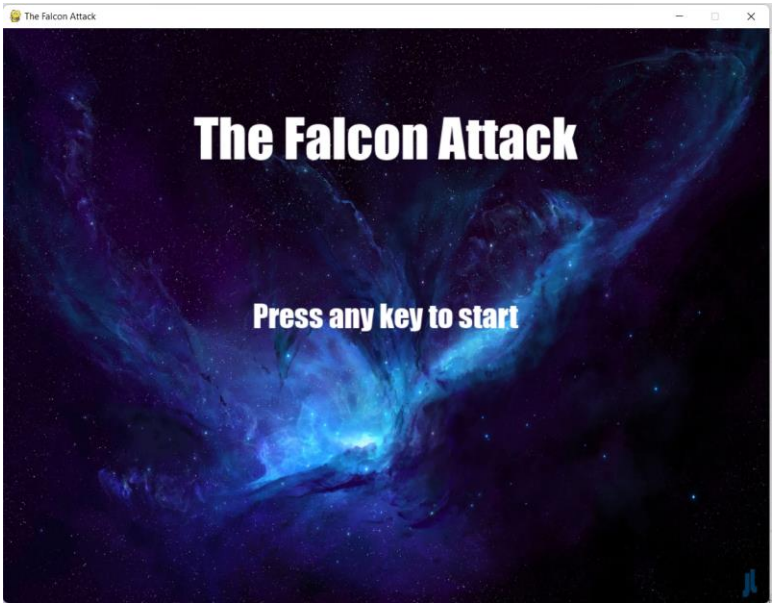
```python
def collide(obj1, obj2):
    offset_x = obj2.x - obj1.x
    offset_y = obj2.y - obj1.y
    return obj1.mask.overlap(obj2.mask, (offset_x, offset_y)) != None


def main():
    run = True
    FPS = 60
    level = 0
    lives = 3
    main_font = pygame.font.SysFont("impact", 30)
    lost_font = pygame.font.SysFont("impact", 60)

    enemies = []
    wave_length = 5
    enemy_vel = 1

    player_vel = 5
    laser_vel = 5

    player = Player(450, 630)

    clock = pygame.time.Clock()

    lost = False
    lost_count = 0

    def redraw_window():
        WIN.blit(BG, (0, 0))

        lives_label = main_font.render(f"Lives: {lives}", 1, (255, 255, 255))
        level_label = main_font.render(f"Level: {level}", 1, (255, 255, 255))

        WIN.blit(lives_label, (WIDTH - level_label.get_width() - 10, 10))
        WIN.blit(level_label, (10, 10))

        for enemy in enemies:
            enemy.draw(WIN)

        player.draw(WIN)

        if lost:
            lost_label = lost_font.render("Game Over!!", 1, (255, 0, 0))
            WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))

        pygame.display.update()
```

```
    while run:
        clock.tick(FPS)
        redraw_window()

        if lives <= 0 or player.health <= 0:
            lost = True
            lost_count += 1

        if lost:
            if lost_count > FPS * 3:
                run = False
            else:
                continue

        if len(enemies) == 0:
            level += 1
            wave_length += 5
            for i in range(wave_length):
                enemy = Enemy(random.randrange(
                    50, WIDTH-100), random.randrange(-1500, -100),
random.choice(["red", "blue", "green"]))
                enemies.append(enemy)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()

        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT] and player.x - player_vel > 0:
            player.x -= player_vel
        if keys[pygame.K_RIGHT] and player.x + player_vel + player.get_width() <
WIDTH:
            player.x += player_vel
        if keys[pygame.K_UP] and player.y - player_vel > 0:
            player.y -= player_vel
        if keys[pygame.K_DOWN] and player.y + player_vel + player.get_height() + 15
< HEIGHT:
            player.y += player_vel
        if keys[pygame.K_SPACE]:
            player.shoot()

        for enemy in enemies[:]:
            enemy.move(enemy_vel)
            enemy.move_lasers(laser_vel, player)

            if random.randrange(0, 2*60) == 1:
                enemy.shoot()
```

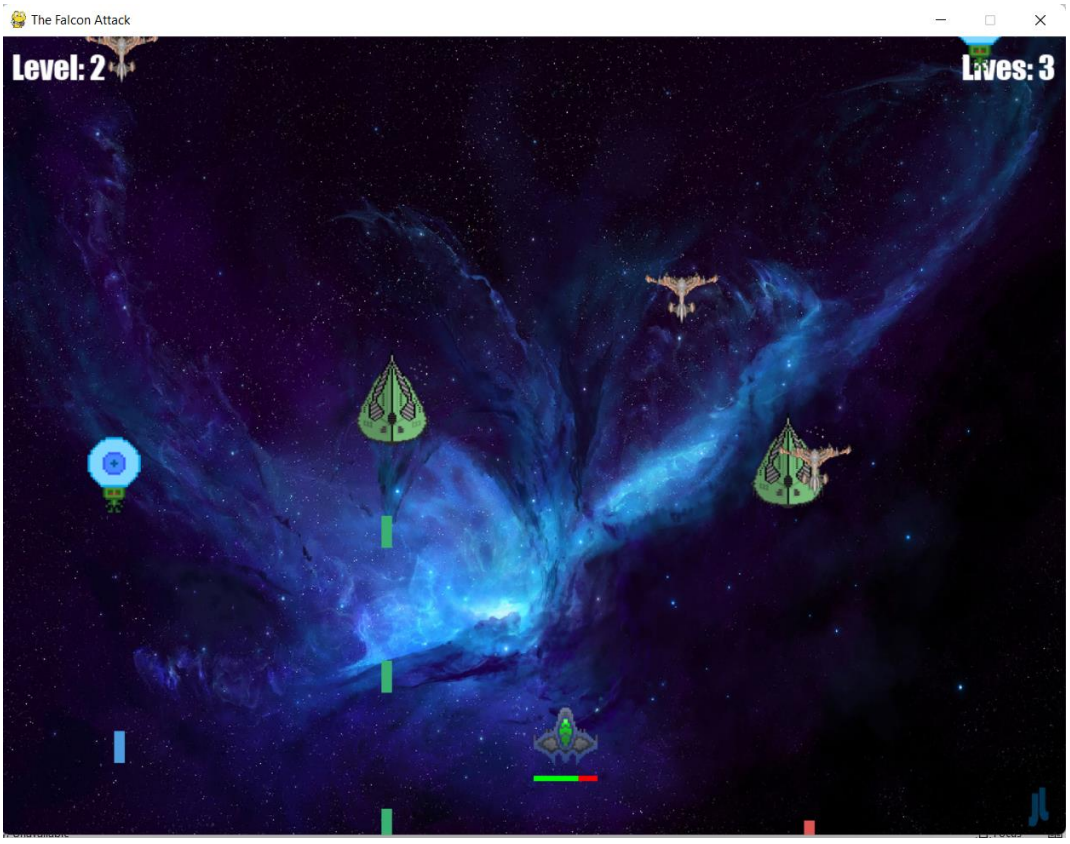
```
            if collide(enemy, player):
                player.health -= 10
                enemies.remove(enemy)
            elif enemy.y + enemy.get_height() > HEIGHT:
                lives -= 1
                enemies.remove(enemy)

        player.move_lasers(-laser_vel, enemies)


def main_menu():
    title_heading = pygame.font.SysFont("impact", 70)
    title_font = pygame.font.SysFont("impact", 40)
    run = True
    while run:
        WIN.blit(BG, (0, 0))
        heading_label = title_heading.render(
            f"The Falcon Attack", 1, (255, 255, 255))
        WIN.blit(heading_label, (WIDTH/2 - heading_label.get_width()/2, 100))

        title_label = title_font.render(
            "Press any key to start", 1, (255, 255, 255))
        WIN.blit(title_label, (WIDTH/2 - title_label.get_width()/2, 350))
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                main()
    pygame.quit()


main_menu()
```

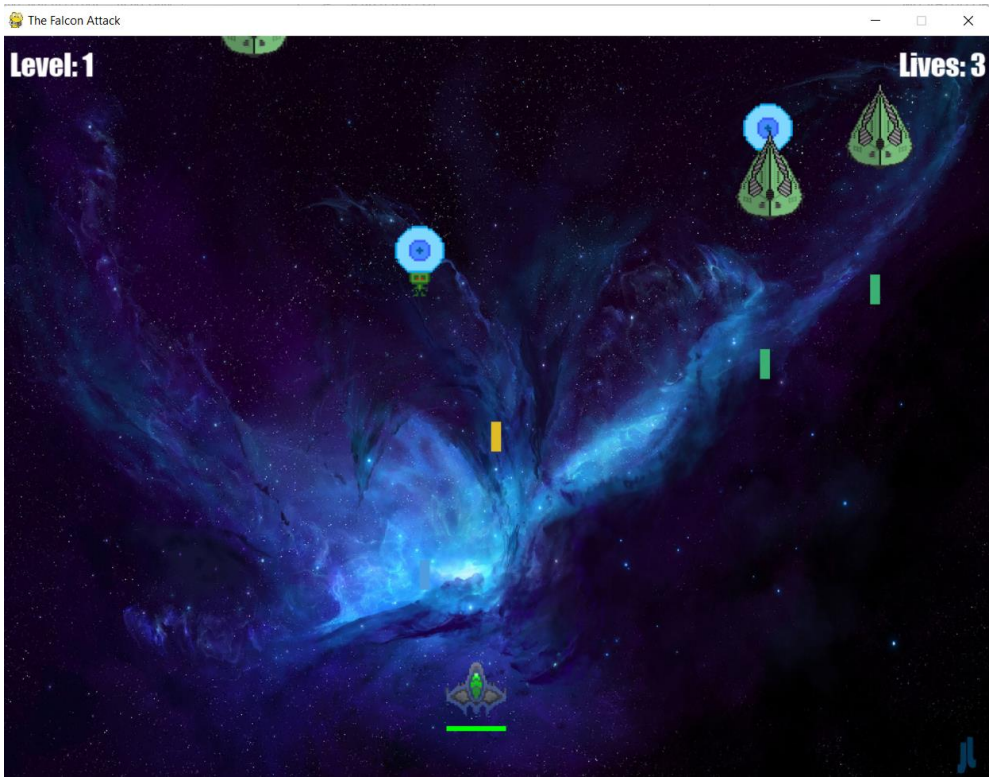
Screenshots

Main Menu

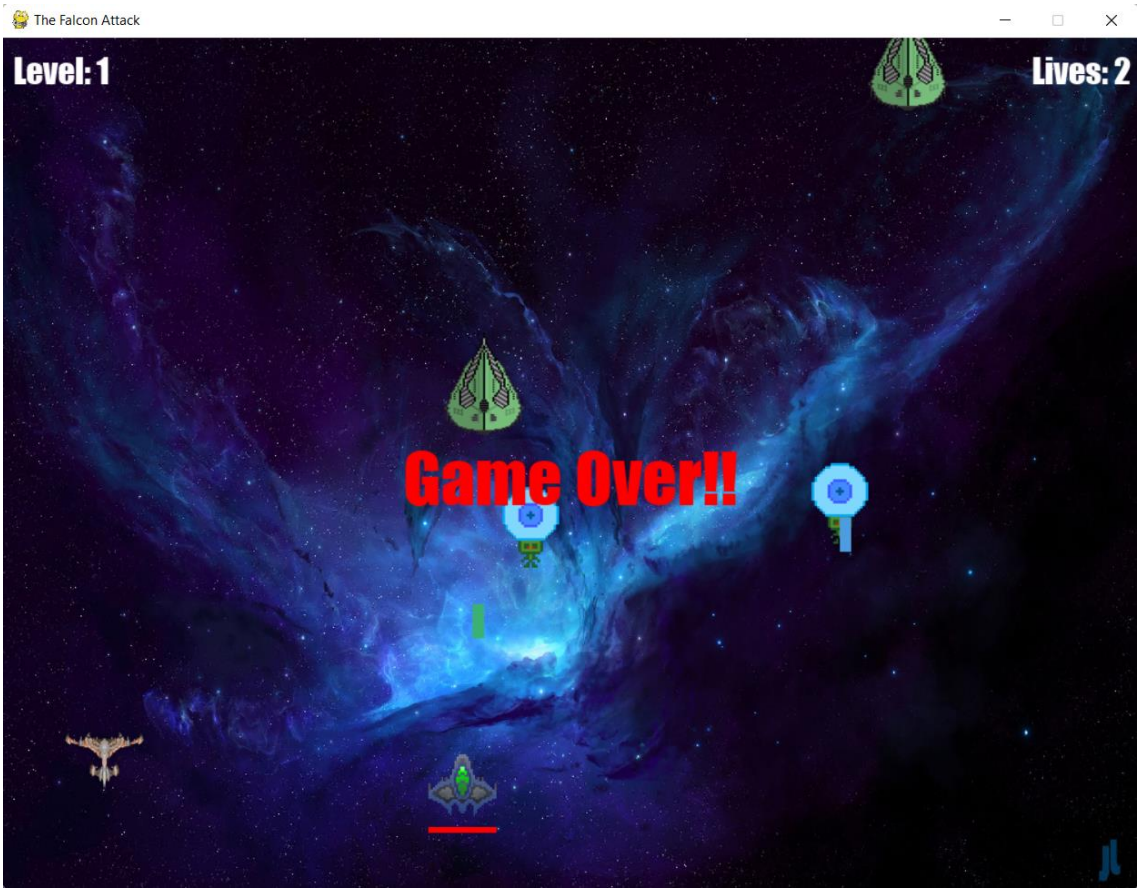


In Game





End of the game



Project Outcome

This project has been designed for entertainment purposes. Hopefully, this will be enjoyable for a player who is playing the game. This project increased our knowledge about Python and also the use of a graphical user interface in Python. This has given us some basic development experience. Overall, we can say that this has been a knowledge-building experience for the developer. Also hopefully it will be enjoyable as much for the player as it has been to the developer.

Project Report

This part will mainly be about the good and bad sides of the project and also about possible future development. This is a basic space shooter game. Its difficulty increases with level and various types of enemies make the game experience enjoyable. But there are some problems with the project as well. There is no boss fight in the game. Also, there are no spells in the game and no option for customization of the space shuttle. There can be a lot of future improvements. The game is currently very simple. The UI can be improved. More documentation is needed and the code can be made cleaner. This application could also the equipped with better graphics in the future.