Project Report

CY5-1 Analysis of Algorithms

Names- Inam, Noroz Ali, M.Ashan, Sada Ullah

SAP ID - 45521, 45496, 43914, 45871

BS (Cyber Security**)**

Instructor

Muhammad Usman Sharif

RIPHAH INTERNATIONAL UNIVERSITY

Project Report: Space Invaders Game

Project Title:

Space Invaders Game

Introduction:

This project is a Python-based implementation of the classic arcade game **Space Invaders**. The game involves controlling a player spaceship to shoot and destroy alien invaders while dodging alien bullets. As the player progresses, the game dynamically increases difficulty by respawning aliens with faster movement and firing rates. A main menu allows the user to start a new game, view the high score, or quit.

Objective:

To create a scalable and interactive arcade game that showcases programming concepts such as event handling, collision detection, and performance management for a dynamically scaling game environment.

Features:

1. **Gameplay:**
    - Control a player spaceship with keyboard inputs.
    - Destroy aliens to earn points and survive.
    - Aliens fire bullets at the player, increasing the challenge.
2. **Dynamic Difficulty:**
    - Aliens respawn with faster movement upon being destroyed.
    - Alien firing rates increase over time.
3. **Main Menu:**
    - Start a new game.
    - View the high score.
    - Quit the application.
4. **High Score Tracking:**
    - Retains the highest score for the session.

Game Flow:

1. **Main Menu:**
    - The player selects an option: New Game, View High Score, or Quit.
2. **Gameplay Mechanics:**
    - The player moves and shoots bullets to destroy aliens.
    - Destroyed aliens respawn with increased speed and faster firing rates.
    - The game ends when the player is hit by an alien bullet.
3. **Game Over:**
    - Displays the score and returns to the main menu.
    - Updates the high score if necessary.

---

1. **Time Complexity (Per Frame)**: O(n2)O(n^2)O(n2).
2. **Space Complexity**: O(n2)O(n^2)O(n2).

Challenges Faced:

1. **Collision Detection:** Scaling collision detection efficiently for n2n^2n2 entities.
2. **Dynamic Difficulty:** Balancing alien speed and firing rates to maintain playability.
3. **Optimization:** Managing performance as the number of entities increases.

Future Enhancements:

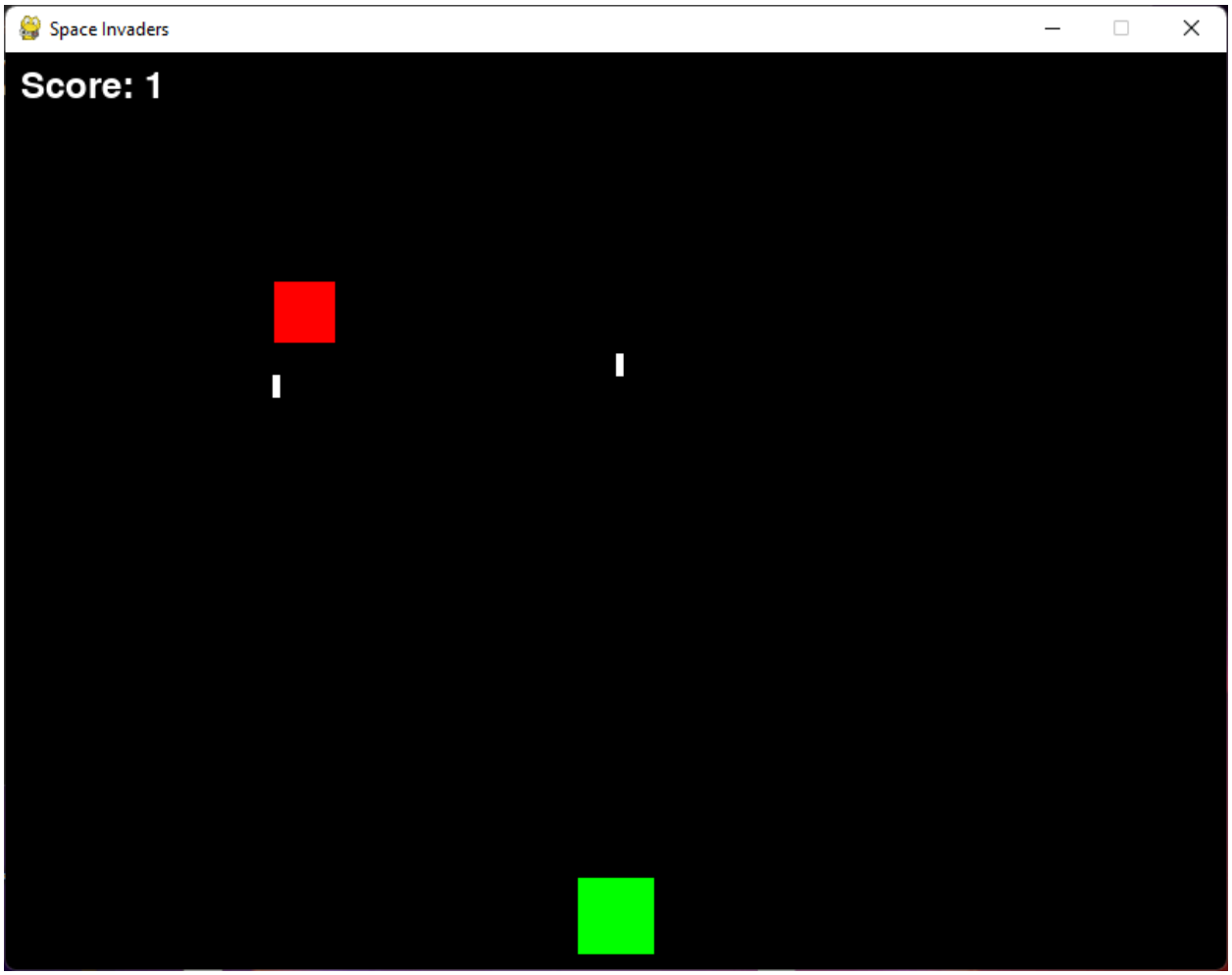
1. **Persistent High Scores:** Save scores across sessions using file handling or a database.
2. **Level Progression:** Add levels with varying alien formations and movement patterns.
3. **Enhanced Graphics:** Introduce animations, explosions, and more dynamic visuals.
4. **Power-Ups:** Add collectible items for shields, extra lives, or rapid fire.

Appendix:

Code Files: Python script (`space_invaders.py`).

Libraries: Install Pygame using `pip install pygame`.

Game Play:



CODE:

```
import pygame

import random

import sys

# Initialize pygame

pygame.init()

# Screen dimensions

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption("Space Invaders")

# Colors
```

```
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
# FPS
clock = pygame.time.Clock()
FPS = 60
# Load assets
player_width, player_height = 50, 50
alien_width, alien_height = 40, 40
bullet_width, bullet_height = 5, 15
# Game variables
player_speed = 5
bullet_speed = -7
alien_speed_increment = 0.5
alien_bullets = []
alien_fire_rate = 3000  # Milliseconds between alien shots
bullet_cooldown = 500  # Milliseconds between shots
max_bullets = 5  # Max bullets on screen
high_score = 0  # Persistent high score
font = pygame.font.Font(None, 36)


def main_menu():
    """Display the main menu and handle user input."""
    while True:
        screen.fill(BLACK)
        title_text = font.render("SPACE INVADERS", True, WHITE)
        new_game_text = font.render("1. New Game", True, GREEN)
        high_score_text = font.render(f"2. High Score: {high_score}", True, WHITE)
```

```
        quit_text = font.render("3. Quit", True, RED)
        # Display menu options
        screen.blit(title_text, (WIDTH // 2 - title_text.get_width() // 2, 100))
        screen.blit(new_game_text, (WIDTH // 2 - new_game_text.get_width() // 2, 200))
        screen.blit(high_score_text, (WIDTH // 2 - high_score_text.get_width() // 2, 250))
        screen.blit(quit_text, (WIDTH // 2 - quit_text.get_width() // 2, 300))
        pygame.display.flip()
        # Handle menu input
        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_1:
                    return "new_game"
                elif event.key == pygame.K_2:
                    return "high_score"
                elif event.key == pygame.K_3:
                    pygame.quit()
                    sys.exit()
def game_loop():
    """The main game loop."""
    global high_score
    # Player variables
    player_x = WIDTH // 2 - player_width // 2
    player_y = HEIGHT - player_height - 10
    last_bullet_time = 0
    bullets = []
    # Alien variables
```

```
aliens = [{"x": random.randint(0, WIDTH - alien_width), "y": random.randint(50, 150), "speed": 3}]
last_alien_fire_time = pygame.time.get_ticks()
# Game variables
score = 0
running = True
while running:
    screen.fill(BLACK)
    current_time = pygame.time.get_ticks()
    # Event handling
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # Player movement
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and player_x > 0:
        player_x -= player_speed
    if keys[pygame.K_RIGHT] and player_x < WIDTH - player_width:
        player_x += player_speed
    if keys[pygame.K_SPACE] and current_time - last_bullet_time > bullet_cooldown:
        bullets.append((player_x + player_width // 2, player_y))
        last_bullet_time = current_time
    # Update bullets
    for i, (bx, by) in enumerate(bullets):
        bullets[i] = (bx, by + bullet_speed)
        if by < 0:
            bullets.pop(i)
    # Check collisions with aliens
    for bullet in bullets[:]:
        for alien in aliens[:]:
```

```
        if check_collision(bullet[0], bullet[1], alien["x"], alien["y"], alien_width, alien_height):
            bullets.remove(bullet)
            aliens.remove(alien)
            score += 1
            # Spawn new alien
            aliens.append({
                "x": random.randint(0, WIDTH - alien_width),
                "y": random.randint(50, 150),
                "speed": abs(alien["speed"]) + alien_speed_increment
            })
            break
    # Alien movement
    for alien in aliens:
        alien["x"] += alien["speed"]
        if alien["x"] <= 0 or alien["x"] >= WIDTH - alien_width:
            alien["speed"] *= -1
    # Alien firing
    if current_time - last_alien_fire_time > alien_fire_rate:
        for alien in aliens:
            alien_bullets.append((alien["x"] + alien_width // 2, alien["y"] + alien_height))
        last_alien_fire_time = current_time
    # Update alien bullets
    for i, (bx, by) in enumerate(alien_bullets):
        alien_bullets[i] = (bx, by + abs(bullet_speed) // 2)
        if by > HEIGHT:
            alien_bullets.pop(i)
    # Check collisions with player
    for bullet in alien_bullets[:]:
        if check_collision(bullet[0], bullet[1], player_x, player_y, player_width, player_height):
```

```
            running = False
        # Drawing
        draw_player(player_x, player_y)
        for alien in aliens:
            draw_alien(alien["x"], alien["y"])
        for bullet in bullets:
            draw_bullet(*bullet)
        for bullet in alien_bullets:
            draw_bullet(*bullet)
        # Display score
        score_text = font.render(f"Score: {score}", True, WHITE)
        screen.blit(score_text, (10, 10))
        # Update display
        pygame.display.flip()
        clock.tick(FPS)
    # Update high score
    if score > high_score:
        high_score = score

def check_collision(obj1_x, obj1_y, obj2_x, obj2_y, obj2_width, obj2_height):
    return obj2_x < obj1_x < obj2_x + obj2_width and obj2_y < obj1_y < obj2_y + obj2_height
def draw_player(x, y):
    pygame.draw.rect(screen, GREEN, (x, y, player_width, player_height))



def draw_alien(x, y):
    pygame.draw.rect(screen, RED, (x, y, alien_width, alien_height))
def draw_bullet(x, y):
    pygame.draw.rect(screen, WHITE, (x, y, bullet_width, bullet_height))
```

```
# Main game loop
while True:
    choice = main_menu()
    if choice == "new_game":
        game_loop()
    elif choice == "high_score":
        screen.fill(BLACK)
        high_score_text = font.render(f"High Score: {high_score}", True, WHITE)
        screen.blit(high_score_text, (WIDTH // 2 - high_score_text.get_width() // 2, HEIGHT // 2))
        pygame.display.flip()
        pygame.time.wait(2000)
```