

SPACE INVADERS

PYTHON PROGRAMING



Simón Bosque Goni

Mario Álvarez Gracia

SUMMARY

Introduction.....	3
Aim.....	3
Scope.....	3
Methodology.....	3
Theoretical Part.....	4
Game Design Theory.....	4
Practical Part.....	4
Implementation.....	4
Setting Up the Environment.....	4
Player Control.....	4
Alien Behavior.....	6
Collision Detection.....	6
Conclusion.....	6
Bibliography.....	6

Introduction

This project involves creating a simple 2D game using Python where a green spaceship controlled by the player must shoot and destroy invading aliens. The game utilizes the Pygame library to handle graphics and user input, offering an engaging experience reminiscent of classic arcade games.

Aim

The primary aim of this project is to develop a basic interactive game where the player controls a spaceship to defend against waves of alien invaders. The focus is on implementing game mechanics, including movement, shooting, collision detection, and scoring.

Scope

The scope of this project includes:

- Developing a player-controlled spaceship that can move left, right, and shoot.
- Creating alien invaders that move towards the player and must be destroyed.
- Implementing basic game mechanics such as collision detection and scoring.
- Using the Pygame library for rendering graphics and handling user input.

Methodology

The project is developed using Python, leveraging the Pygame library for game development. The process involved:

- Designing the game layout and mechanics.
- Coding the spaceship and alien behaviors.
- Implementing collision detection and scoring systems.
- Testing and debugging to ensure smooth gameplay.

Theoretical Part

Game Design Theory

Game design involves creating interactive experiences that engage players. Key elements include:

- **Player Control:** Ensuring responsive and intuitive controls for the spaceship.
- **Enemy AI:** Designing predictable yet challenging patterns for alien movement.
- **Collision Detection:** Implementing algorithms to detect interactions between the spaceship's bullets and the aliens.
- **Scoring System:** Rewarding players for destroying aliens and achieving high scores.

Practical Part

Implementation

Setting Up the Environment

We set up a Pygame window with a background image representing space. The spaceship and alien images are loaded and rendered on the screen. The spaceship is controlled using keyboard inputs, allowing it to move left, right, and shoot bullets.

Player Control

The spaceship's movement is controlled by capturing keyboard events. The left and right arrow keys move the spaceship horizontally, while the spacebar triggers shooting.

```

1 import pygame
2 import random
3
4 # Initialize Pygame
5 pygame.init()
6
7 # Screen dimensions
8 screen_width = 800
9 screen_height = 600
10
11 # Create the screen
12 screen = pygame.display.set_mode((screen_width, screen_height))
13
14 # Title and Icon
15 pygame.display.set_caption("Space Invaders")
16 icon = pygame.image.load("spaceship.png")
17 pygame.display.set_icon(icon)
18
19 # Player
20 playerImg = pygame.image.load('player.png')
21 playerX = 370
22 playerY = 480
23 playerX_change = 0
24
25 def player(x, y):
26     screen.blit(playerImg, (x, y))
27
28 # Enemy
29 enemyImg = pygame.image.load('enemy.png')
30 enemyX = random.randint(0, screen_width - 64)
31 enemyY = random.randint(50, 150)
32 enemyX_change = 0.3
33 enemyY_change = 40
34
35 def enemy(x, y):
36     screen.blit(enemyImg, (x, y))

```

```

37
38 # Bullet
39 bulletImg = pygame.image.load('bullet.png')
40 bulletX = 0
41 bulletY = 480
42 bulletY_change = 0.5
43 bullet_state = "ready" # "ready" means you can't see the bullet on the screen
44
45 def fire_bullet(x, y):
46     global bullet_state
47     bullet_state = "fire"
48     screen.blit(bulletImg, (x + 16, y + 10))
49
50 # Main game loop
51 running = True
52 while running:
53     screen.fill((0, 0, 0)) # RGB
54     for event in pygame.event.get():
55         if event.type == pygame.QUIT:
56             running = False
57         if event.type == pygame.KEYDOWN:
58             if event.key == pygame.K_LEFT:
59                 playerX_change = -0.3
60             if event.key == pygame.K_RIGHT:
61                 playerX_change = 0.3
62             if event.key == pygame.K_SPACE:
63                 if bullet_state == "ready":
64                     bulletX = playerX
65                     fire_bullet(bulletX, bulletY)
66         if event.type == pygame.KEYUP:
67             if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
68                 playerX_change = 0

```

```

69
70     playerX += playerX_change
71     if playerX <= 0:
72         playerX = 0
73     elif playerX >= screen_width - 64:
74         playerX = screen_width - 64
75
76     player(playerX, playerY)
77     enemy(enemyX, enemyY)
78
79     pygame.display.update()

```

Alien Behavior

Aliens are programmed to move horizontally across the screen and descend towards the player at intervals. The position of the aliens is updated in each frame, and they reset upon reaching the screen edges.

Collision Detection

Collision detection is implemented to determine when a bullet hits an alien. This is achieved by calculating the distance between the bullet and the alien and checking if it is below a certain threshold.

```
import math

def is_collision(enemyX, enemyY, bulletX, bulletY):
    distance = math.sqrt((math.pow(enemyX - bulletX, 2)) + (math.pow(enemyY - bulletY, 2)))
    return distance < 27
```

Conclusion

The goal of creating a simple, playable game where a player-controlled spaceship defends against alien invaders was successfully achieved. The implementation covers basic game mechanics, providing an engaging experience. Future improvements could include adding multiple levels, different enemy types, and enhancing graphics and sound effects.

Bibliography

- Pygame Documentation: <https://www.pygame.org/docs/>
- Python Official Documentation: <https://docs.python.org/3/>
- Game Development Tutorials: <https://www.youtube.com/user/sentdex>