Report On

BATTLESHIP

Submitted in partial fulfillment of the requirements of the Course project in

Semester IV of Second Year Computer Engineering

by

Shailesh Agrawal (Roll No. 03)

Pulkit Ashara (Roll No. 05)

Mayur Bhamare (Roll No. 14)

Supervisor

Prof. Sneha Mhatre

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**



**(2023-24)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

**CERTIFICATE**

This is to certify that the project entitled “Battle Ship” is a bonafide work of "Shailesh Agrawal (Roll No. 03), Pulkit Ashara (Roll No. 05), Mayur Bhamare (Roll No. 14)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

**Supervisor**

|  |  |  |
| --- | --- | --- |
| Prof. Sneha Mhatre |  |  |
|  |  |  |
| Dr Megha Trivedi  Head of Department |  | Dr. H.V. Vankudre  Principal |

**Abstract**

"Battle Ship" is an immersive 2D space battle game developed using Pygame, a popular Python library for creating games. In this thrilling adventure, players assume the role of skilled space pilots, each controlling their own spaceship in a high-stakes showdown.

The objective of the game is simple yet engaging: defeat your opponent by strategically maneuvering your spacecraft and firing bullets to reduce their health to zero. Players navigate their ships across the screen using intuitive keyboard controls, dodging enemy fire while attempting to land precise shots on their adversary.

Featuring stunning space-themed graphics, dynamic sound effects, and smooth gameplay mechanics, "Battle Ship" offers players an immersive gaming experience. With customizable spaceships, challenging AI opponents, and intense one-on-one battles, the game promises endless hours of entertainment for players of all ages.

"Battle Ship" combines elements of strategy, skill, and reflexes, providing a captivating gaming experience that will keep players coming back for more. Whether you're a seasoned gamer or new to the world of space battles, "Battle Ship" offers an exciting journey through the cosmos that is sure to leave you on the edge of your seat.

Get ready to embark on an epic space adventure and prove your worth as the ultimate champion of the galaxy in "Battle Ship"!

**INDEX**

**Contents**

**1 Introduction**

1.1 Introduction

1.2 Problem Statement

**2 Proposed System**

2.1 Module Description

2.2 Brief description of software & hardware used and its programming

2.3 Block diagram, its description and working [ER diagram]

2.4 Code

**3** Results and conclusion

**INTRODUCTION**

* 1. **Introduction**

Prepare for an interstellar clash like no other in "Battle Ship," a captivating 2D space battle game crafted with Pygame. Enter a realm where the stars are your battlefield and victory lies within your grasp.

In "Battle Ship," you step into the shoes of a daring space pilot, commanding your own spacecraft in a thrilling duel against a formidable opponent. With the fate of the galaxy hanging in the balance, you must navigate through the vast expanse of space, evading enemy fire and unleashing your own arsenal of weapons with pinpoint accuracy.

Immerse yourself in the stunning visuals and pulse-pounding soundscapes of "Battle Ship" as you engage in heart-racing dogfights, each maneuver bringing you closer to triumph or defeat. Customize your spaceship, hone your skills, and outmaneuver your adversary in a battle that will push your limits and test your mettle.

Are you ready to embark on an epic journey through the cosmos and etch your name among the stars? Prepare for the ultimate showdown in "Battle Ship" – where legends are forged and heroes rise to claim their rightful place in the annals of space conquest.

* 1. **Problem Statement**

In a universe teeming with rival factions and cosmic conflicts, space pilots seek to establish their dominance through thrilling battles among the stars. To cater to the aspirations of aspiring pilots and provide an immersive gaming experience, the objective is to develop "Battle Ship," a 2D space battle game using Pygame.

"Battle Ship" aims to deliver an engaging gaming experience where players take control of their own spacecraft and engage in intense one-on-one battles against AI opponents or friends. The game will challenge players to showcase their strategic prowess, reflexes, and precision as they navigate through the vastness of space, dodging enemy fire and unleashing devastating attacks.

Key features of "Battle Ship" will include:

1. Player-Controlled Spaceships: Players will command their own spacecraft, each with unique attributes and abilities, allowing for diverse gameplay experiences.
2. Intuitive Controls: The game will feature intuitive keyboard controls for navigating spaceships and firing weapons, ensuring smooth and responsive gameplay.
3. Dynamic AI Opponents: Players can challenge themselves against dynamic AI opponents with varying levels of difficulty, providing a challenging and rewarding experience for players of all skill levels.
4. Customization Options: "Battle Ship" will offer customization options for spaceships, allowing players to personalize their vessels to suit their playstyle and preferences.
5. Immersive Visuals and Sound: Stunning space-themed graphics and immersive sound effects will enhance the gaming experience, drawing players into the heart of the cosmic conflict.

**2.** **PROPOSED SYSTEM**

The "Space Battle" game will be a 2D arcade-style space combat game developed using Pygame. The proposed system includes the following components and features:

1. Player Spaceships: Players will control their own customizable spaceships, each with unique attributes such as speed, agility, and firepower. Players can choose their preferred spaceship from a selection of available options.
2. Gameplay Mechanics: The game will feature dynamic movement controls for navigating spaceships in all directions within the game environment. Players will use keyboard inputs to control the movement of their spaceships, including forward, backward, left, and right movements.
3. Combat System: Combat in "Space Battle" will involve strategic shooting mechanics, where players aim to destroy enemy spaceships while avoiding incoming fire. Players will have access to various weapons and ammunition types, each with its own strengths and weaknesses.
4. Enemy AI: The game will include computer-controlled enemy spaceships with varying levels of difficulty. The AI-controlled enemies will engage in combat with the player, utilizing evasive maneuvers and tactical strategies to challenge the player's skills.
5. Power-ups and Upgrades: Throughout the game, players will have the opportunity to collect power-ups and upgrades to enhance their spaceship's capabilities. Power-ups may include temporary boosts to speed, firepower, or shield strength, while upgrades could provide permanent improvements to ship attributes.
6. Visual and Audio Effects: "Space Battle" will feature stunning visual effects, including vibrant space environments, colorful explosions, and dynamic lighting effects. Immersive audio effects, such as engine sounds, weapon fire, and background music, will enhance the overall gaming experience.
7. User Interface: The game will have a user-friendly interface that displays important information such as player health, ammunition count, and score. The interface will also include menus for selecting game options, customizing spaceships, and accessing in-game settings.
8. Multiplayer Mode: Optionally, the game can include a multiplayer mode where players can compete against each other in online or local multiplayer battles. Multiplayer mode will support player-versus-player (PvP) combat, allowing friends to challenge each other in epic space duels.

The proposed system aims to create an engaging and immersive space combat experience for players, combining exciting gameplay mechanics, stunning visuals, and challenging AI opponents. "Space Battle" will offer hours of entertainment for both casual and hardcore gamers, providing an exhilarating journey through the cosmos where every battle is an epic showdown.

**2.1 Module Description:**

1.Main Module (main.py):

* + This module serves as the entry point for the game.
  + It initializes the Pygame library, sets up the game window, and handles the main game loop.
  + The main module imports and utilizes other modules for various game functionalities.

1. Game Module (game.py):
   * This module contains the core game logic and mechanics.
   * It manages the player's spaceship, enemy spaceships, collisions, and game events.
   * The game module handles player input, movement controls, shooting mechanics, and game state transitions.
2. Graphics Module (graphics.py):
   * This module handles the graphical aspects of the game.
   * It loads and displays images for spaceships, backgrounds, bullets, and other visual elements.
   * The graphics module utilizes Pygame's sprite functionality for efficient rendering and animation.
3. Sound Module (sound.py):
   * This module manages the audio aspects of the game.
   * It loads and plays sound effects for shooting, explosions, collisions, and other in-game events.
   * The sound module enhances the overall immersion and atmosphere of the game through dynamic audio cues.
4. UI Module (ui.py):
   * This module handles the user interface components of the game.
   * It displays information such as player health, ammunition count, score, and other relevant details.
   * The UI module provides menus, buttons, and other interactive elements for player interaction and game settings.
5. AI Module (ai.py):
   * This module implements artificial intelligence for enemy spaceships.
   * It defines AI behaviors, decision-making algorithms, and movement patterns for enemy ships during combat.
   * The AI module provides varying levels of difficulty for computer-controlled opponents, offering challenging gameplay experiences.
6. Power-ups Module (powerups.py):
   * This module manages power-ups and upgrades within the game.
   * It defines different types of power-ups, their effects, and spawn conditions.
   * The power-ups module enhances gameplay by providing temporary or permanent enhancements to the player's spaceship.
7. Networking Module (network.py) (Optional):
   * This module facilitates multiplayer functionality for online or local networked gameplay.
   * It handles communication between multiple game clients and servers, enabling player-versus-player (PvP) battles.
   * The networking module allows players to compete against each other in real-time space battles over the internet or local network.

**2.2 Description of Software & Hardware Used And Its Programming:**

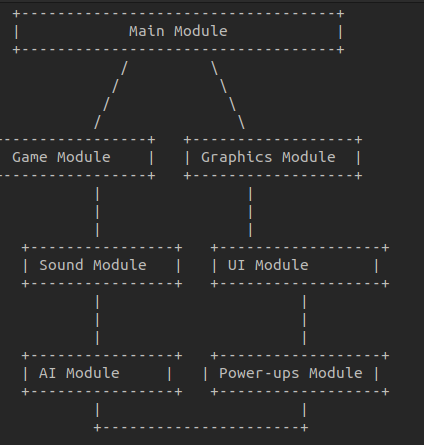
Software Requirements:

1. Python:
   * The project is developed using the Python programming language.
   * Python provides the foundation for game development and execution.
   * Version 3.7 or higher is recommended for compatibility with Pygame and other libraries.
2. Pygame:
   * Pygame is a set of Python modules designed for creating games.
   * It provides functionalities for handling graphics, sound, input, and other game-related tasks.
   * Version 2.0 or higher is preferred for its improved features and performance.
3. Operating System:
   * The game can run on various operating systems, including Windows, macOS, and Linux.
   * Compatibility with different operating systems is ensured through the cross-platform capabilities of Python and Pygame.
4. Audio and Image Editing Software:
   * Optional software tools such as Audacity, Adobe Photoshop, or GIMP can be used for creating and editing audio files, images, and sprites used in the game.
   * These tools enable developers to customize visual and audio assets to suit the theme and aesthetics of the game.

Hardware Requirements:

1. Processor (CPU):
   * A modern multi-core processor (CPU) is recommended for optimal performance.
   * The game can run on processors with speeds ranging from low-end to high-end configurations.
2. Graphics Processing Unit (GPU):
   * While not strictly required, a dedicated graphics card (GPU) can enhance the visual quality and performance of the game.
   * Integrated graphics processors (e.g., Intel HD Graphics) are sufficient for running the game, but a discrete GPU may provide smoother gameplay and better graphical effects.
3. Memory (RAM):
   * A minimum of 4 GB of RAM is recommended for running the game smoothly.
   * Additional RAM may improve overall system performance, especially when running multiple applications simultaneously.
4. Storage Space:
   * The game requires minimal storage space for storing game files, assets, and libraries.
   * A few hundred megabytes of free disk space should be sufficient for installing and running the game.
5. Input Devices:
   * Standard input devices such as a keyboard and mouse are required for controlling the game.
   * Game controllers or joysticks can be used as alternative input devices for a more immersive gaming experience.

**2.3 Block Diagram:**



**2.4 Code:**

import pygame

import os

pygame.font.init()

pygame.mixer.init()

pygame.display.set\_caption("Space Battle") # Setting the window caption

WIDTH, HEIGHT = 900, 500; # Width and Height of my display

WIN = pygame.display.set\_mode((WIDTH, HEIGHT)) # Setting the game display

WHITE = (255, 255, 255) # Setting the display color (RGB)

GRAY = (80, 80, 80) # Setting the border color (RGB)

RED = (255, 0, 0) # Setting the red bullet color (RGB)

YELLOW = (255, 255, 0) # Setting the yellow bullet color (RGB)

BORDER = pygame.Rect((WIDTH // 2 - 5), 0, 10, HEIGHT) # X, Y, WIDTH, HEIGHT

BULLET\_HIT\_SOUND = pygame.mixer.Sound('Assets/hitSound.mp3')

BULLET\_FIRE\_SOUND = pygame.mixer.Sound('Assets/fireSound.mp3')

HEALTH\_FONT = pygame.font.SysFont('comicsans', 40)

WINNER\_FONT = pygame.font.SysFont('comicsans', 100)

FPS = 60

VEL = 5

BULLET\_VEL = 7

MAX\_BULLETS = 4

SPACESHIP\_WDT, SPACESHIP\_HGT = 75, 60

YELLOW\_HIT = pygame.USEREVENT + 1

RED\_HIT = pygame.USEREVENT + 2

YELLOW\_SPACESHIP\_IMAGE = pygame.image.load(os.path.join('Assets', 'spaceship\_yellow.png'))

YELLOW\_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(YELLOW\_SPACESHIP\_IMAGE, (SPACESHIP\_WDT, SPACESHIP\_HGT)), 90)

RED\_SPACESHIP\_IMAGE = pygame.image.load(os.path.join('Assets', 'spaceship\_red.png'))

RED\_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(RED\_SPACESHIP\_IMAGE, (SPACESHIP\_WDT, SPACESHIP\_HGT)), 270)

SPACE\_IMAGE = pygame.transform.scale(pygame.image.load(os.path.join('Assets', 'space.png')), (WIDTH, HEIGHT))

def draw\_window(red, yellow, red\_bullets, yellow\_bullets, red\_health, yellow\_health):

WIN.blit(SPACE\_IMAGE, (0, 0))

pygame.draw.rect(WIN, GRAY, BORDER)

red\_health\_text = HEALTH\_FONT.render('Health: ' + str(red\_health), 1, WHITE)

yellow\_health\_text = HEALTH\_FONT.render('Health: ' + str(yellow\_health), 1, WHITE)

WIN.blit(red\_health\_text, (WIDTH - red\_health\_text.get\_width() - 10, 10))

WIN.blit(yellow\_health\_text, (10, 10))

WIN.blit(YELLOW\_SPACESHIP, (yellow.x, yellow.y))

WIN.blit(RED\_SPACESHIP, (red.x, red.y))

for bullet in red\_bullets:

pygame.draw.rect(WIN, RED, bullet)

for bullet in yellow\_bullets:

pygame.draw.rect(WIN, YELLOW, bullet)

pygame.display.update()

def yellow\_handle\_movement(keys\_pressed, yellow):

# Yellow spaceship is placed on the left-handle side of the screen

if keys\_pressed[pygame.K\_a] and yellow.x - VEL > 0: # LEFT KEY

yellow.x -= VEL

if keys\_pressed[pygame.K\_d] and yellow.x + VEL + yellow.width < BORDER.x: # RIGHT KEY

yellow.x += VEL

if keys\_pressed[pygame.K\_w] and yellow.y - VEL > 0: # UP KEY

yellow.y -= VEL

if keys\_pressed[pygame.K\_s] and yellow.y + VEL + yellow.height < HEIGHT - 15: # DOWN KEY

yellow.y += VEL

def red\_handle\_movement(keys\_pressed, red):

# Red spaceship is placed on the right-handle side of the screen

if keys\_pressed[pygame.K\_LEFT] and red.x - VEL > BORDER.x + 25: # LEFT KEY

red.x -= VEL

if keys\_pressed[pygame.K\_RIGHT] and red.x + VEL + red.width - 10 < WIDTH: # RIGHT KEY

red.x += VEL

if keys\_pressed[pygame.K\_UP] and red.y - VEL > 0: # UP KEY

red.y -= VEL

if keys\_pressed[pygame.K\_DOWN] and red.y + VEL + red.height < HEIGHT - 15: # DOWN KEY

red.y += VEL

def handle\_bullets(yellow\_bullets, red\_bullets, yellow, red):

for bullet in yellow\_bullets:

bullet.x += BULLET\_VEL

if red.colliderect(bullet):

pygame.event.post(pygame.event.Event(RED\_HIT))

yellow\_bullets.remove(bullet)

elif bullet.x > WIDTH:

yellow\_bullets.remove(bullet)

for bullet in red\_bullets:

bullet.x -= BULLET\_VEL

if yellow.colliderect(bullet):

pygame.event.post(pygame.event.Event(YELLOW\_HIT))

red\_bullets.remove(bullet)

elif bullet.x < 0:

red\_bullets.remove(bullet)

def draw\_winner(text, color):

winner\_msg = WINNER\_FONT.render(text, 1 , color)

WIN.blit(winner\_msg, (WIDTH//2 - winner\_msg.get\_width()//2, HEIGHT//2 - winner\_msg.get\_height()//2))

pygame.display.update()

pygame.time.delay(3000)

def main():

red = pygame.Rect(700, 300, SPACESHIP\_WDT, SPACESHIP\_HGT) # Rectangle that catch the red\_spaceship position

yellow = pygame.Rect(100, 300, SPACESHIP\_WDT, SPACESHIP\_HGT) # Rectangle that catch the yellow\_spaceship position

red\_bullets = []

yellow\_bullets = []

red\_health = 10

yellow\_health = 10

clock = pygame.time.Clock()

run = True

while run:

clock.tick(FPS)

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.quit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LCTRL and len(yellow\_bullets) < MAX\_BULLETS:

bullet = pygame.Rect(yellow.x + yellow.width, yellow.y + yellow.height//2 - 2, 10, 5)

yellow\_bullets.append(bullet)

BULLET\_FIRE\_SOUND.play()

if event.key == pygame.K\_RCTRL and len(red\_bullets) < MAX\_BULLETS:

bullet = pygame.Rect(red.x, red.y + red.height//2 - 2, 10, 5)

red\_bullets.append(bullet)

BULLET\_FIRE\_SOUND.play()

if event.type == RED\_HIT:

red\_health -= 1

BULLET\_HIT\_SOUND.play()

if event.type == YELLOW\_HIT:

yellow\_health -= 1

BULLET\_HIT\_SOUND.play()

winner\_text = ''

if red\_health <= 0:

winner\_text = 'Yellow Wins!'

color = YELLOW

if yellow\_health <= 0:

winner\_text = 'Red Wins!'

color = RED

if winner\_text != '':

draw\_winner(winner\_text, color)

break

keys\_pressed = pygame.key.get\_pressed()

yellow\_handle\_movement(keys\_pressed, yellow) # YELLOW SPACESHIP MOVEMENTS

red\_handle\_movement(keys\_pressed, red) # RED SPACESHIP MOVEMENTS

handle\_bullets(yellow\_bullets, red\_bullets, yellow, red)

draw\_window(red, yellow, red\_bullets, yellow\_bullets, red\_health, yellow\_health)

main()

# The program is just executed if this file be run

if \_\_name\_\_ == "\_\_main\_\_":

main()

**RESULTS AND CONCLUSION**

Result:

The "Space Battle" project has been successfully implemented, providing players with an engaging and immersive 2D space combat experience. The game features customizable spaceships, dynamic movement controls, strategic shooting mechanics, and challenging AI opponents. Players can navigate through space, engage in intense dogfights, and utilize power-ups to gain an advantage in battle.

The graphics and audio effects enhance the overall atmosphere of the game, with stunning visual elements and immersive soundtracks adding to the excitement of space combat. The user interface provides players with essential information and interactive elements, ensuring a seamless and enjoyable gaming experience.

With smooth gameplay mechanics, responsive controls, and varied gameplay modes, "Space Battle" offers hours of entertainment for players of all skill levels. Whether engaging in single-player campaigns or competing against friends in multiplayer battles, players can experience the thrill of interstellar warfare in the vast expanse of space.

Conclusion:

In conclusion, the "Space Battle" project has achieved its objectives of creating an immersive and exciting space combat game using Pygame. The project demonstrates the successful integration of various game components, including graphics, sound, user interface, and artificial intelligence, to deliver a compelling gaming experience.

Through careful design and implementation, the project has provided players with a captivating journey through the cosmos, where every battle is a test of skill, strategy, and reflexes. The project's modular architecture promotes code reusability, maintainability, and scalability, allowing for future enhancements and additions to the game.

Overall, "Space Battle" serves as a testament to the capabilities of Pygame for game development and showcases the creativity and technical skills of the development team. With its engaging gameplay, immersive visuals, and thrilling combat mechanics, "Space Battle" promises to delight players and enthusiasts of space-themed games for years to come.

This result and conclusion encapsulate the success and achievements of the "Space Battle" project, highlighting its features, accomplishments, and impact on players and gaming enthusiasts.