VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", BELAGAVI – 590018

A MINI PROJECT REPORT ON

"ROCKET GAME"

Submitted in partial fulfillment of requirements for the *course*

Computer Graphics and Image Processing Laboratory [21CSL66]

of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2023-24.

Submitted By

UDAY G
[4MH21CS114]

SUMANTH G N
[4MH22CS408]

Under the Guidance of

"Prof. Pallavi. Y"
Assistant Professor,
Dept. of CS&E,

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
Belawadi, S.R. Patna (T), Mandya (D) – 571477.

2023-24

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna (T), Mandya (D) – 571477.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini project work entitled **"ROCKET GAME"** is a bonafide work carried out by **UDAY G [4MH21CS114]** and **SUMANTH G N [4MH22CS408]** in partial fulfillment for **Computer Graphics and Image Processing Laboratory [21CSL66]** prescribed by the Visvesvaraya Technological University, Belagavi during the year 2023-2024 for the fifth semester B.E in Computer Science and Engineering. The mini project report has been approved as it satisfies the academic requirements.

Signature of Guide

(Prof. Pallavi .Y)
Assistant Professor, Dept. of CS&E
MIT Mysore

Signature of HOD

(Dr. Shivamurthy R C)
Professor & Head, Dept. of CS&E
MIT Mysore

Name of the Examiners	Signature with date
1.....	….....
2.....	….....

ACKNOWLEDGEMENT

We sincerely owe our gratitude to all the people who helped and guided us in completing this mini project work.

We are thankful to **Dr. B.G. Naresh Kumar, Principal, Maharaja Institute of Technology Mysore**, for having supported us in our academic endeavors.

We are extremely thankful to **Dr. Shivamurthy R C, Professor & Head, Department of Computer Science and Engineering,** for his valuable support and his timely inquiries into the progress of the work.

We are greatly indebted to our guide **Prof. Pallavi. Y, Assistant Professor, Department of Computer Science and Engineering,** for the consistent co-operation and support.

We are obliged to all **teaching and non-teaching staff members** of the **Department of Computer Science and Engineering** for the valuable information provided by them in their respective field's. We are grateful for their co-operation during the period of our mini project.

Uday G [4MH21CS066]

Sumanth G N[4MH21CS067]

TABLE OF CONTENT

1. INTRODUCTION:

The code initializes Pygame, sets the screen dimensions, and defines color constants. It creates a Pygame window with the title "Rocket Game." Represents the player-controlled rocket. The rocket can move in four directions (up, down, left, right) using the arrow keys or the numeric keypad. It has a polygonal shape drawn on the screen. Represents the bullets fired by the rocket. Bullets move upward and disappear when they reach the top of the screen. They are created when the spacebar is pressed. Represents enemy objects that appear randomly from the top of the screen. Enemies move downward. Their health depends on the player's score. If an enemy collides with the rocket, the game ends. The main game loop runs continuously until the player quits. It handles user input, updates rocket position, bullet movement, enemy spawning, and collision detection. The player's score is displayed on the screen. Controls (arrow keys and spacebar) are shown as white triangles.

1.1 Aim of the Project:

The aim of this project is to create a simple 2D game using the Pygame library. The game involves controlling a rocket, shooting bullets, and avoiding enemies. The player's objective is to survive as long as possible while accumulating points by shooting down enemies.

1.2 Overview of the Project:

- **Game Elements:**
  - Rocket: The player controls a rocket using arrow keys (or W, A, S, D).
  - Bullets: The rocket can fire bullets when the spacebar is pressed.
  - Enemies: Enemies (represented as red rectangles) descend from the top of the screen.
  - Score: The player's score increases when enemies are shot down.
- **Game Mechanics:** The rocket moves within the screen boundaries. Bullets move upward and disappear when they reach the top. Enemies move downward and disappear when shot or when they reach the bottom. Colliding with an enemy ends the game.
- **Controls:** Arrow keys (or W, A, S, D) control the rocket's movement. Spacebar fires bullets.
- **Scoring:** Points are awarded for each enemy shot down. Enemy health increases as the player's score rises

2. SYSTEM ANALYSIS AND DESIGN

2.1 System Analysis:

Installation and Setup: Ensure you have Python and Pygame installed. Create a game directory named "Platformer-Game." Download game assets (sprites, backgrounds, etc.) or use your own. Organize your files and folders within the game directory.

Basic Game Components: Set up basic configurations in settings.py (e.g., window dimensions, tile size). Create the game window. Define the player's character (rocket) using the Rocket class. Add movement controls for the player.

Adding Platforms: Create platforms (blocks) for the player to jump on. Implement collision detection between the player and platforms. Ensure the player stays within the screen boundaries.

Enemies and Scoring: Create enemy characters (e.g., aliens, obstacles). Assign health points to enemies based on the player's score. Detect collisions between bullets (fired by the player) and enemies. Update the score when enemies are defeated.

Game Loop: Continuously update the game state (player position, enemy movement, etc.). Handle user input (keyboard controls). Display the game elements (player, enemies, score) on the screen. Implement game over conditions (e.g., player collides with an enemy).

Controls: Use arrow keys or WASD for movement (up, down, left, right). Press the space bar to fire bullets.

2.2 Algorithm:

Initialization and Setup: You've initialized Pygame and set up the game window with dimensions (800x600). Defined colors (WHITE, RED, BLACK, BLUE, GREEN) for drawing game elements.

Rocket Class: Represents the player character. Handles movement (up, down, left, right) within screen boundaries. Draws the rocket using a polygon.

Bullet Class: Represents bullets fired by the player. Bullets move upward and are removed when they reach the top of the screen.

Enemy Class: Represents enemies descending from the top. Enemy health is determined based on the player's score. Enemies are removed when their health reaches zero.

Game Loop: Continuously updates the game state. Handles player input (keyboard controls). Manages bullets and enemy movement. Updates the score.

Controls and Display: Displays controls (arrow keys, space bar). Shows the player's score.

3. REQUIREMENTS

3.1 Software Requirements:

Python: The game is written in Python using the Pygame library. Ensure you have Python installed on your system.

Pygame: Install the Pygame library using pip install pygame. Pygame provides the necessary functionality for creating games, handling graphics, and user input.

IDE or Text Editor: You'll need an integrated development environment (IDE) or a text editor to write and run the Python code. Popular choices include Visual Studio Code, PyCharm, or Jupyter Notebook.

3.2 Hardware Requirements:

Computer: Any modern computer with sufficient processing power and memory should be able to run this game.

Input Devices: You'll need a keyboard to control the rocket (using arrow keys or WASD) and a mouse for interaction (e.g., closing the game window).

Display: A monitor or screen to display the game graphics. The game window size is set to 800x600 pixels in the provided code.

Operating System: The game should work on Windows, macOS, or Linux.

4. IMPLEMENTATION

4.1 Source code:

```
import pygame
import random

# Initialize Pygame
pygame.init()

# Screen dimensions
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600

# Colors
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)

# Set up display
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Rocket Game")

# Rocket class
class Rocket:
    def __init__(self):
        self.width = 50
        self.height = 60
        self.x = SCREEN_WIDTH // 2
        self.y = SCREEN_HEIGHT - self.height - 10
        self.speed = 5
        self.rect = pygame.Rect(self.x, self.y, self.width, self.height)
    def move(self, dx, dy):
```

```
        self.x += dx * self.speed
        self.y += dy * self.speed

        # Boundary check
        if self.x < 0:
            self.x = 0
        elif self.x > SCREEN_WIDTH - self.width:
            self.x = SCREEN_WIDTH - self.width

        if self.y < 0:
            self.y = 0
        elif self.y > SCREEN_HEIGHT - self.height:
            self.y = SCREEN_HEIGHT - self.height

        self.rect.topleft = (self.x, self.y)
    def draw(self):
        rocket_design = [
            (self.x + 25, self.y),  # Top point
            (self.x, self.y + 30),  # Bottom left point
            (self.x + 12.5, self.y + 30),  # Bottom left wing point
            (self.x + 12.5, self.y + 50),  # Bottom left wing bottom point
            (self.x + 25, self.y + 60),  # Bottom center point
            (self.x + 37.5, self.y + 50),  # Bottom right wing bottom point
            (self.x + 37.5, self.y + 30),  # Bottom right wing point
            (self.x + 50, self.y + 30)   # Bottom right point
        ]
        pygame.draw.polygon(screen, BLUE, rocket_design)

# Bullet class
class Bullet:
    def __init__(self, x, y):
        self.width = 5
        self.height = 10
```

```
        self.x = x + 22.5
        self.y = y
        self.speed = 7
        self.rect = pygame.Rect(self.x, self.y, self.width, self.height)

    def move(self):
        self.y -= self.speed
        self.rect.topleft = (self.x, self.y)

    def draw(self):
        pygame.draw.rect(screen, GREEN, self.rect)

# Enemy class
class Enemy:
    def __init__(self, health):
        self.width = 50
        self.height = 50
        self.x = random.randint(0, SCREEN_WIDTH - self.width)
        self.y = -self.height
        self.speed = random.randint(2, 5)
        self.health = health  # Set enemy health based on score
        self.rect = pygame.Rect(self.x, self.y, self.width, self.height)

    def move(self):
        self.y += self.speed
        self.rect.topleft = (self.x, self.y)
    def draw(self):
        pygame.draw.rect(screen, RED, self.rect)
# Game loop
def game_loop():
    rocket = Rocket()
    bullets = []
    enemies = []
```

```
score = 0
clock = pygame.time.Clock()
font = pygame.font.SysFont(None, 36)
running = True
while running:
    screen.fill(BLACK)  # Fill the screen with black
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    keys = pygame.key.get_pressed()
    dx, dy = 0, 0
    if keys[pygame.K_w] or keys[pygame.K_UP] or keys[pygame.K_KP8]:
        dy = -1
    if keys[pygame.K_a] or keys[pygame.K_LEFT] or keys[pygame.K_KP4]:
        dx = -1
    if keys[pygame.K_d] or keys[pygame.K_RIGHT] or keys[pygame.K_KP6]:
        dx = 1
    if keys[pygame.K_s] or keys[pygame.K_DOWN] or keys[pygame.K_KP2]:
        dy = 1
    rocket.move(dx, dy)
    rocket.draw()
    # Fire bullets continuously when space bar is held down
    if keys[pygame.K_SPACE]:
        bullets.append(Bullet(rocket.x, rocket.y))
    # Determine enemy health based on score
    enemy_health = 3
    if score > 50:
        enemy_health = 5
    elif score > 100:
        enemy_health = 7
    elif score > 150:
        enemy_health = 10
    elif score > 200:
```

```
    enemy_health = 12

if random.randint(1, 20) == 1:
    enemies.append(Enemy(enemy_health))

for bullet in bullets:
    bullet.move()
    bullet.draw()

for enemy in enemies:
    enemy.move()
    enemy.draw()
    if enemy.rect.colliderect(rocket.rect):
        running = False
for bullet in bullets:
    for enemy in enemies:
        if bullet.rect.colliderect(enemy.rect):
            bullets.remove(bullet)
            enemy.health -= 1
            if enemy.health <= 0:
                enemies.remove(enemy)
                score += 1
            break

bullets = [bullet for bullet in bullets if bullet.y > 0]
enemies = [enemy for enemy in enemies if enemy.y < SCREEN_HEIGHT]

score_text = font.render(f"Score: {score}", True, WHITE)
screen.blit(score_text, (10, 10))

# Display controls
controls_text = font.render("Controls:", True, WHITE)
screen.blit(controls_text, (SCREEN_WIDTH - 120, 10))
```

```
        pygame.draw.polygon(screen, WHITE, [(SCREEN_WIDTH - 100, 40),
(SCREEN_WIDTH - 80, 40), (SCREEN_WIDTH - 90, 20)])
        pygame.draw.polygon(screen, WHITE, [(SCREEN_WIDTH - 100, 80),
(SCREEN_WIDTH - 80, 80), (SCREEN_WIDTH - 90, 100)])
        pygame.draw.polygon(screen, WHITE, [(SCREEN_WIDTH - 120, 60),
(SCREEN_WIDTH - 120, 80), (SCREEN_WIDTH - 140, 70)])
        pygame.draw.polygon(screen, WHITE, [(SCREEN_WIDTH - 80, 60),
(SCREEN_WIDTH - 80, 80), (SCREEN_WIDTH - 70, 70)])

        pygame.display.flip()
        clock.tick(30)
    pygame.quit()

if __name__ == "__main__":
    game_loop()
```

5. RESULT ANALYSIS

5.1 Snap Shots:

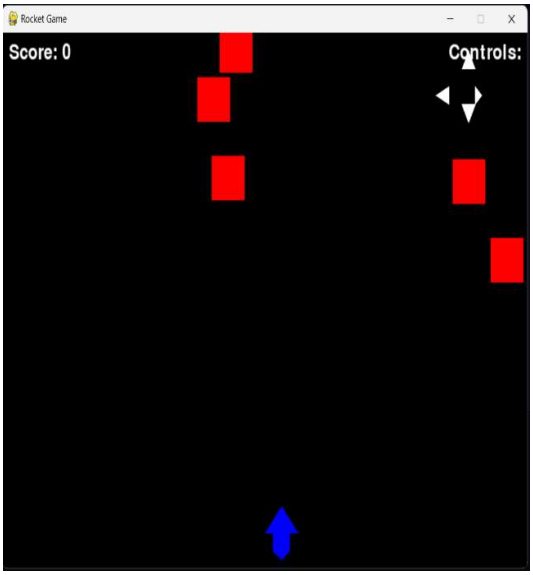

Fig1: Initial stage

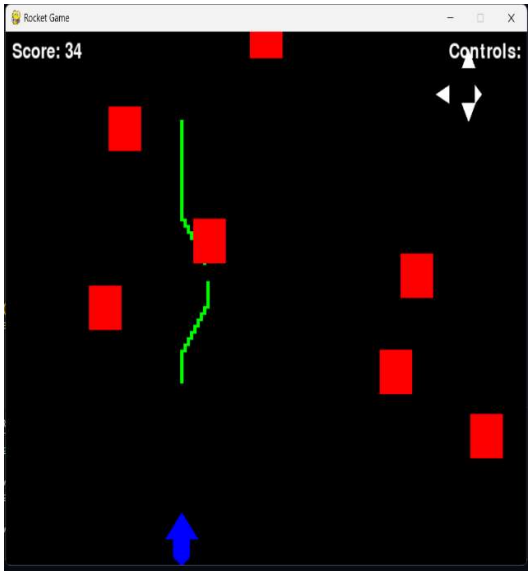

Fig2: Runing to rocket

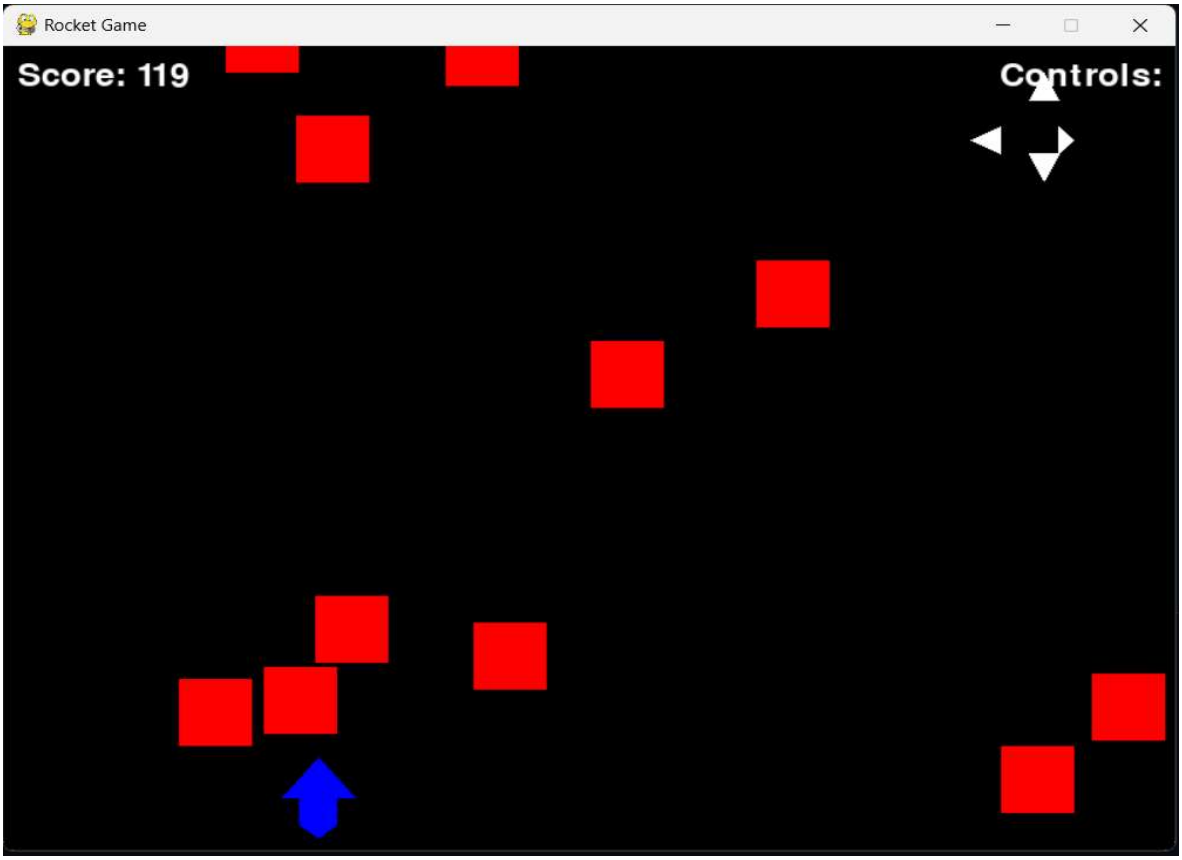

Fig3: Enemies attack to rocket

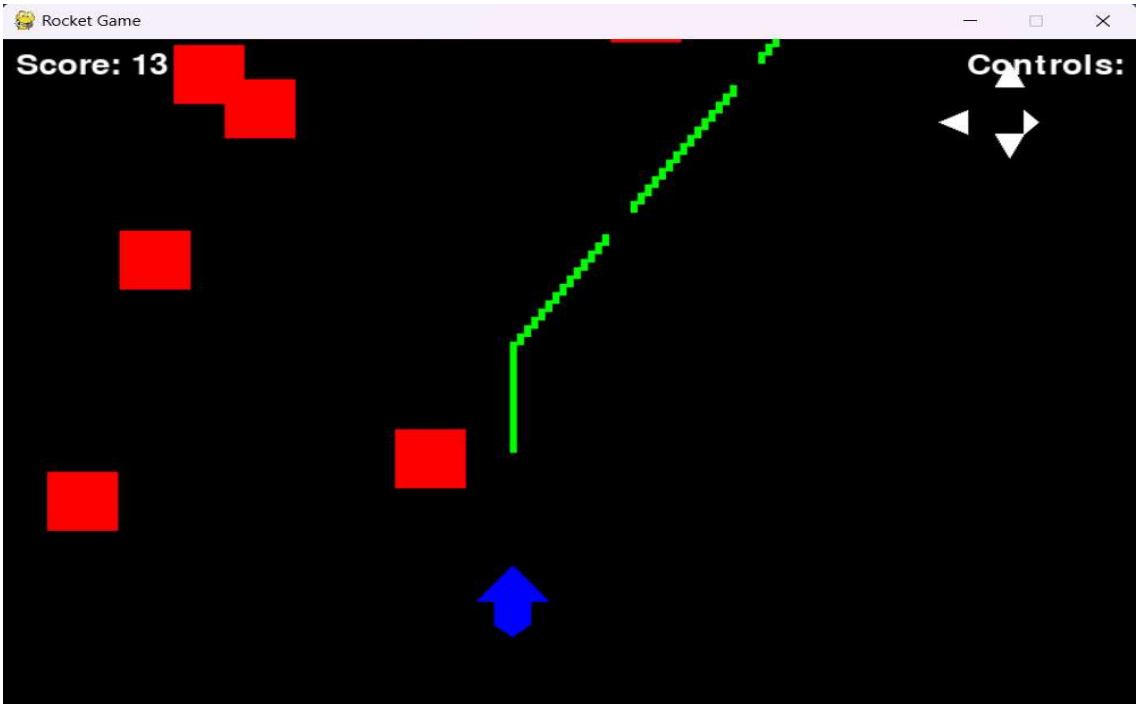

Fig4: Movement of rocket

5.2 Discussion:

This Python code snippet demonstrates a simple game using the Pygame library. Let's break down the key components:

Initialization and Setup: The code initializes Pygame, sets up the screen dimensions, and defines color constants. It creates a display window for the game.

Rocket Class: The Rocket class represents the player-controlled rocket. It defines properties like position, speed, and dimensions. The move method updates the rocket's position based on user input (arrow keys). The draw method renders the rocket on the screen as a polygon.

Bullet Class: The Bullet class represents bullets fired by the rocket. Bullets move upward and disappear when they reach the top of the screen. The draw method renders the bullets.

Enemy Class: The Enemy class represents enemy objects falling from the top. Enemies have random positions, speeds, and health (based on the player's score). The draw method renders the enemies.

Game Loop: The main game loop runs continuously until the player quits. It handles user input, updates rocket position, fires bullets, and spawn's enemies. Collisions between bullets and enemies are detected, affecting enemy health and player score. The game displays the score and control instructions.

Controls: Arrow keys or WASD control the rocket's movement. Spacebar fires bullets.

5.3 Outcome of the Project:

When you run this code, a Pygame window will open, displaying the game. You'll be able to control the rocket, shoot bullets, and avoid enemies. The game will keep track of your score, and the difficulty will increase as your score grows.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion:

The provided Python code snippet creates a basic 2D game using the Pygame library. Here's a summary of its functionality:

1. **Rocket Movement:** The player controls a rocket using arrow keys (WASD). The rocket moves within the screen boundaries.
2. **Bullet Firing:** Pressing the spacebar fires bullets from the rocket. Bullets move upward and disappear when reaching the top.
3. **Enemy Spawning:** Enemies fall from the top of the screen. Enemy health increases based on the player's score.
4. **Collision Detection:** Bullets collide with enemies, reducing enemy health. When an enemy's health reaches zero, it disappears, and the player's score increases.
5. **Controls Display:** The game displays control instructions (arrow keys and spacebar).

6.2 Future Enhancement:

- Add more enemy types (with different behaviors or abilities).
- Implement power-ups or obstacles.
- Create levels with increasing difficulty.
- Enhance graphics and sound effects.
- Implement a game-over screen and high-score tracking.

7. REFERENCES:

1. **Pygame Documentation:**
   - The official Pygame documentation provides detailed information about the library, including tutorials, examples, and reference documentation for all its functions and modules1.
   - You can find the documentation here: Pygame Documentation
2. **Pygame Tutorials:**
   - The tutorials section of the Pygame website offers step-by-step guides for beginners and more advanced users. It covers various aspects of game development using Pygame2.
   - You can explore the tutorials here: Pygame Tutorials
3. **Pygame on Read the Docs:**
   - The Read the Docs page for Pygame provides an online version of the documentation, which stays up to date with the development version of Pygame on GitHub3.
   - You can access it here: Pygame on Read the Docs