Space Invaders Game using C++ and Raylib

Group Members

1. Obaidullah Khan – [24K-0788]
2. Hammad Siddiqui – [24K-0702]
3. Sahal Arif– [24K-0991]

1. Introduction

Background

The project focuses on developing a 2D arcade-style game, "Space Invaders," utilizing C++ and the Raylib library. This endeavor emphasizes object-oriented programming (OOP) principles, including encapsulation, inheritance, and polymorphism, to manage game entities and behaviors effectively.

Problem Statement

Many beginner programmers struggle to grasp OOP concepts due to a lack of practical applications. By creating a game that employs classes for players, enemies, bullets, and other entities, this project aims to provide a tangible context for understanding and applying OOP methodologies.

Objectives

- Develop a functional Space Invaders game using C++ and Raylib.
- Implement core OOP concepts to structure game components.
- Enhance understanding of game development processes and OOP design patterns.
- Produce comprehensive documentation detailing the development process and OOP applications.

2. Scope of the Project

Inclusions

- Implementation of player-controlled spaceship movement and shooting mechanics.
- Design and behavior of enemy invaders with basic AI movement patterns.
- Collision detection between bullets and enemies.
- Score tracking and display.
- Basic sound effects for shooting and explosions.
- User interface elements such as start screen and game over screen.

Exclusions

- Advanced enemy AI behaviors (e.g., adaptive strategies).

- Multiplayer functionality.
- Online leaderboards or networked features.
- Mobile platform compatibility.

3. Project Description

This project entails the creation of a classic Space Invaders game clone, emphasizing the application of OOP principles in game development. Each game entity (player, enemies, bullets) will be represented as objects with specific attributes and behaviors, facilitating modularity and code reusability.

Technical Requirements

- **Programming Language:** C++
- **Graphics Library:** Raylib
- **Development Environment:** Visual Studio Code
- **Operating System:** Windows

Project Phases

1. **Research and Planning**
    - Study existing implementations and tutorials.
    - Define game mechanics and requirements.
2. **Design**
    - Create class diagrams and game flowcharts.
    - Design user interface layouts.
3. **Implementation**
    - Develop game classes and integrate Raylib functionalities.
    - Implement game logic and user interactions.
4. **Testing**
    - Conduct unit tests for individual components.
    - Perform integration testing to ensure cohesive functionality.
5. **Documentation**
    - Compile a report detailing development processes and OOP applications.
    - Prepare user manuals and code comments for clarity.

4. Methodology

- **Approach:**
  The team followed a structured and incremental approach, directly inspired by our Lab-Instructor Sir Talha Shahid. Each member was assigned specific segments based on their comfort and skill level. This ensured a smooth workflow and allowed each member to take full ownership of their part.

Team Responsibilities

- **Member 1:** Lead developer responsible for implementing core game mechanics and integrating Raylib functionalities.
- **Member 2:** Focus on designing and developing enemy behaviors and collision detection systems.
- **Member 3:** Handle user interface design, sound integration, and documentation compilation.

5. Expected Outcomes

- **Deliverables:**
  - Functional Space Invaders game executable
  - Well-commented and structured C++ code
  - A short report and development documentation
- **Relevance:**
  This project demonstrates practical application of OOP, strengthening foundational programming concepts while introducing students to graphics/game libraries like raylib. It connects to ICT through file handling (high score), data logic (game states), and object-based development.

6. Resources Needed

- **Software:**
  - Visual Studio Code
  - raylib library
  - GitHub (for collaboration)
- **Other Resources:**
  - raylib documentation
  - Instructor support during integration and debugging phases