Project Report: Invaders – A 2D Space Shooter Game

1. Introduction

This report documents the development of a 2D arcade-style shooting game titled **"Invaders"**, built using **C++** and the **Raylib** graphics library. The game is inspired by the classic *Space Invaders* and aims to deliver a retro gaming experience while showcasing the use of modern programming principles such as object-oriented design, structured programming, and efficient graphics handling with Raylib.

*2. Objectives

The main objectives of this project were:

- To implement a simple but engaging 2D shooter game.
- To understand and apply core programming concepts in C++ such as classes, inheritance, and pointers.
- To learn game development basics including game loops, user input, collision detection, and sprite handling.
- To gain hands-on experience with **Raylib**, a simple and fast C library for game development.

3. Tools and Technologies Used

Tool / Technology	Purpose
C++	Core programming language
Raylib	Graphics, input, and sound handling
Visual Studio Code / Code::Blocks	IDE for development
Git & GitHub	Version control and collaboration
Paint.net / Piskel	For designing custom 2D sprites (optional)

4.1 Gameplay Description

- The player controls a spaceship at the bottom of the screen.
- Enemy invaders move horizontally and descend after reaching screen edges.
- The player must shoot the invaders while avoiding enemy bullets.
- The game ends if the player loses all lives or eliminates all enemies.

4.2 Game Features

- Player movement and shooting using keyboard input.
- Multiple waves of enemies.
- Collision detection between bullets and objects.

- Score tracking and game over screen.
- Basic sound effects and music (optional).

5.1 Game Loop

The core of the game relies on the standard game loop structure:

1. **Initialize** – Set up window, variables, load assets.
2. **Update** – Handle input, update game objects, check collisions.
3. **Draw** – Render all game elements on the screen.
4. **Unload** – Free memory and close the window.

6. Implementation Overview

6.1 Player Controls

- Arrow keys: Move left and right.
- Spacebar: Fire bullets.

6.2 Enemy Movement

- Enemies move in groups horizontally.
- Upon reaching the screen edge, they move down and reverse direction.

6.3 Collision Detection

- Implemented using bounding box (rectangular) collision checks between bullets and enemy/player sprites.

6.4 UI Elements

- Score display on top-left.
- Game Over screen with restart option.

6.5 Sound Effects

- Background music and shooting/explosion sounds (if implemented).

---

7. Testing and Debugging

- Manual testing was done for all modules: player movement, shooting, collision, and scoring.
- Debug messages and Raylib's inbuilt drawing functions were used during development for visualization.

- Edge cases (e.g., multiple bullets hitting the same enemy, bullets leaving the screen) were handled.

---

8. Challenges Faced

- Learning the Raylib library syntax and structure.
- Timing-based bullet firing and movement control.
- Collision precision and performance.
- Organizing code for scalability and readability.

---

9. Future Improvements

- Add multiple player lives and health bar.
- Implement advanced enemy patterns and power-ups.
- High score saving functionality.
- Improve UI and animations.
- Add levels and bosses for progressive difficulty.

---

10. Conclusion

The **Invaders** game project successfully demonstrated the fundamentals of game development using **C++ and Raylib**. It allowed for the application of theoretical programming concepts in a practical, engaging way. The project provided valuable experience in handling graphics, user input, game logic, and debugging. It also laid a solid foundation for more complex game development in the future.