2D Space Shooter Game Project Report

1. Introduction

This report outlines the development of a 2D space shooter game using C++. The game allows players to control a spaceship, navigate through space, and engage in battles with enemy ships while avoiding obstacles.

1.1 Objectives

Develop a fun and engaging 2D space shooter game.

Implement basic game mechanics such as player movement, shooting, and enemy AI.

Utilize graphics and sound to enhance the gaming experience.

2. Game Design

2.1 Game Concept

Players control a spaceship in a 2D environment, facing waves of enemy ships. The objective is to survive as long as possible while achieving a high score.

2.2 Game Mechanics

Player Controls: Use keyboard inputs for movement and shooting.

Enemy AI: Simple behaviors that vary in speed and shooting patterns.

Scoring System: Players earn points by destroying enemies and collecting power-ups.

Levels: Progressively harder waves of enemies.

2.3 Visual and Audio Design

Graphics: Pixel art style for spaceships and backgrounds.

Sound: Background music and sound effects for shooting and explosions.

3. Implementation

3.1 Development Environment

Language: C++

Libraries:

SFML (Simple and Fast Multimedia Library): For graphics, audio, and input handling.

Box2D: (Optional) For physics if needed.

3.2 Code Structure

Main Class: Handles game loop, initialization, and rendering.

Player Class: Manages player input and behaviors.

Enemy Class: Defines enemy characteristics and actions.

Bullet Class: Controls bullet movement and collision detection.

Game Manager Class: Oversees game state, score tracking, and level progression.

3.3 Key Features Implementation

Player Movement:

cpp

Copy code

void Player::move(float deltaTime) {

if (isMovingLeft) position.x -= speed \* deltaTime;

if (isMovingRight) position.x += speed \* deltaTime;

if (isShooting) shoot();

}

Enemy Behavior:

cpp

Copy code

void Enemy::update(float deltaTime) {

// Move towards player or patrol

position.x += speed \* deltaTime;

}

Collision Detection: Implemented using bounding box checks between bullets and enemies.

3.4 Challenges Faced

Physics and Collision: Ensuring accurate collision detection without complex physics libraries.

Balancing Gameplay: Adjusting enemy difficulty to maintain player engagement.

4. Testing and Evaluation

4.1 Testing Methods

Unit testing for individual components (player, enemy, bullets).

Playtesting sessions to gather user feedback on difficulty and enjoyment.

4.2 Results

Feedback indicated that the game is enjoyable but could benefit from:

Enhanced enemy AI for more challenging gameplay.

Additional power-ups and weapon types.

5. Future Improvements

Graphics Enhancement: Consider using more detailed sprites and animations.

Level Design: Introduce diverse backgrounds and obstacles.

Multiplayer Mode: Implement a co-op mode for added fun.

6. Conclusion

The 2D space shooter game successfully demonstrates the application of C++ in game development. The project provided valuable experience in coding, game design, and problem-solving. Future work will focus on expanding features and improving overall gameplay.