

**DEPARTMENT OF COMPUTER SCIENCE  
OBJECT ORIENTED PROGRAMMING**

**SEMESTER PROJECT**

AIRBORNE AMUSEMENT

**(A REMAKE OF FLAPPY BIRD)**

**GROUP MEMBERS**

|  |  |
| --- | --- |
| Muhammad Hassan | 01-134242-081 |
| Syed Hasnain | 01-134242-122 |

**PROJECT NAME**

**TABLE OF CONTENTS**

1. Introduction

2. Game Overview

3. Key Features

* Difficulty levels
* Plane control
* Tower movement
* Collision control
* Scoring system
* Audio feedback

4. Additional Features

* Credit page
* Increasing speed as score increases

5. Technical details

* Development environment
* Classes used
* File handling

6. Object-Oriented Principles Used

* Inheritance
* Classes and objects
* Composition
* Polymorphism

7. Screenshots

9. Conclusion

10. References

**Introduction**

This document provides a comprehensive overview of the semester project titled 'Airborne Amusement', a Flappy Bird-style game developed using C++ and the SFML library. The goal of this project was to apply object-oriented programming concepts in a real-world application and develop a fully functional game with interactive features.

**Overview**

Airborne Amusement is a 2D game developed using the SFML (Simple and Fast Multimedia Library). The objective of the game is to navigate a plane through an endless series of towers while avoiding collisions. The game features multiple difficulty levels, a scoring system, and a high score tracking mechanism. This report outlines the key features and functionalities of the game.

**Key Features**

**1. Multiple Difficulty Levels**

* **Easy**: Default tower speed.
* **Medium**: Slightly increased tower speed.
* **Hard**: Higher tower speed.
* The difficulty level can be selected by the player at the start of the game.

**2. Plane Control**

* The player controls a plane that can navigate vertically to avoid obstacles.
* The plane's position is reset at the beginning of each game.

**3. Tower Movement**

* Towers move from right to left across the screen.
* Towers reset to the right side of the screen once they move out of view, creating an endless obstacle course.
* Tower speed varies based on the selected difficulty level.

**4. Collision Detection**

* The game checks for collisions between the plane and towers.
* On collision, the game over screen is displayed.

**5. Scoring System**

* Players earn points for successfully navigating through the towers.
* The score is displayed on the screen during gameplay.
* The game over screen shows the final score of the player.

**6. High Score Tracking**

* High scores are tracked and saved.
* The high score list is updated whenever a new high score is achieved.
* High scores are persistent across game sessions.

**7. Audio Feedback**

* Background music plays during the game.
* A sound effect is played when the game is over.

**8. Additional Features**

* The game increases speed once the score crosses 6 and again after 11, increasing the challenge.
* High scores are saved, and only the top 5 are retained for display.
* A Credits page is included, displaying the names of the developers and the supervising teacher.

**Technical Details**

**Development Environment**

* **Language**: C++
* **Library**: SFML (Simple and Fast Multimedia Library)

**Classes and Responsibilities**

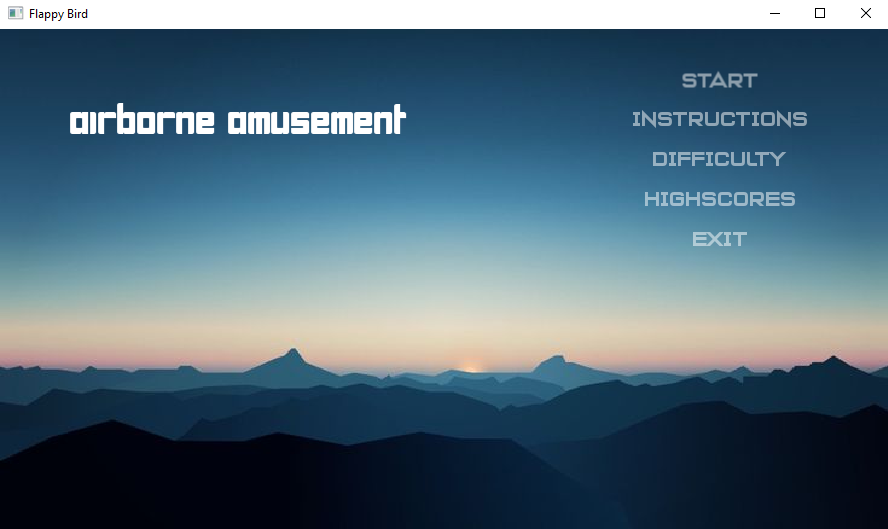
* **Plane**: Handles plane properties and movement.
* **HighScore**: Manages high scores, including adding and saving scores.
* **Game Over Screen**: Displays the final score after a game over.
* **Difficulty**: Manages the game's difficulty level.
* **Game**: Core game logic, including updating game state, handling collisions, and resetting the game.
* **Menu**: Manages the main menu and user interactions before the game starts.

**Object-Oriented Principles Used**

The game design leverages Object-Oriented Programming concepts:  
**Encapsulation**: Data and behaviors are contained within classes  
**Inheritance:** Shared behavior among entities such as menus and screens  
**Composition:** Game is composed of smaller, modular components like Plane, Tower, and Score  
**Abstraction:** Simplified interaction via public methods  
**Polymorphism:** Used in UI handling and menu systems

**OUTPUT/SCREENSHOTS**

**Main Menu:**

****

**DIFFICULTY WINDOW:**

**A mountain range with a blue sky and white text

Description automatically generated**

**PLANE SELECTION WINDOW:**

**A group of airplanes in a row

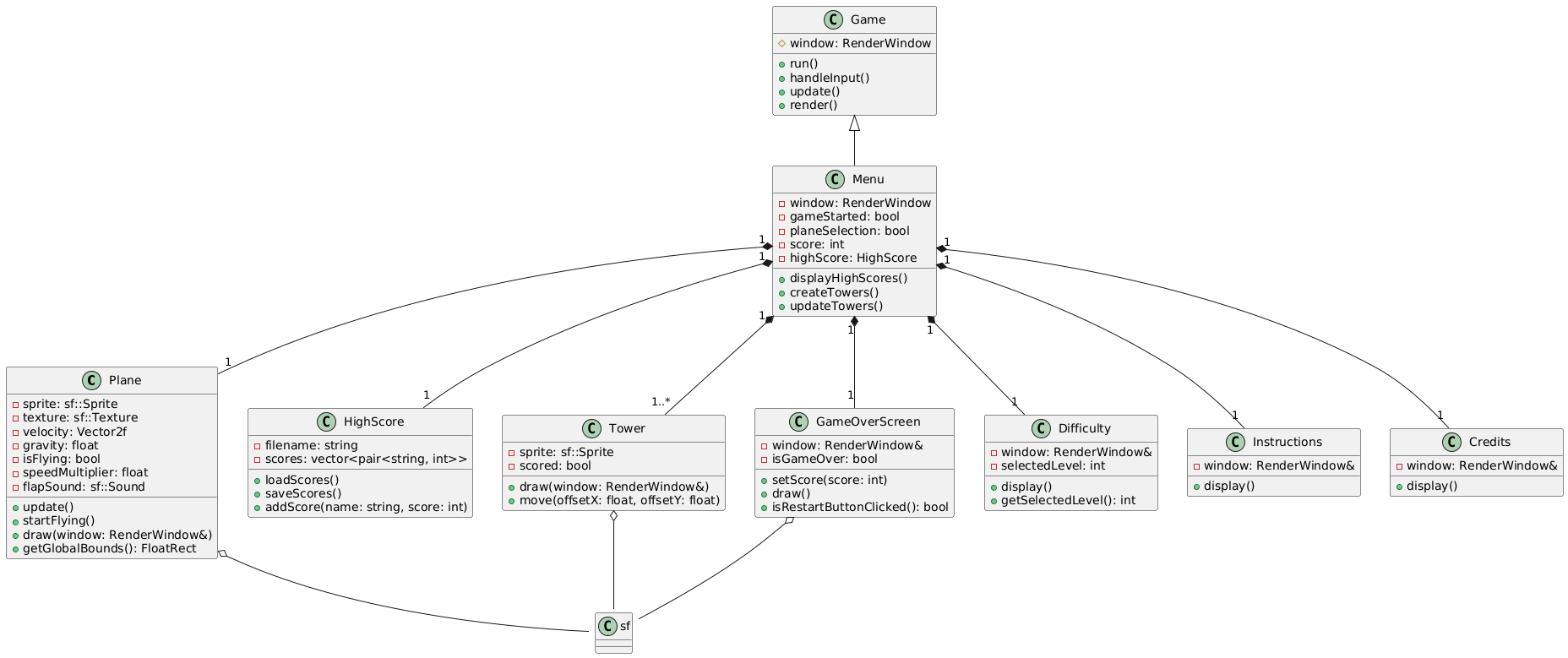
Description automatically generated**

**Conclusion**

Airborne Amusement successfully demonstrates the implementation of OOP principles in a real-time game. The inclusion of gameplay dynamics, user interaction, and persistent data storage makes it a well-rounded academic project.

**References**

- SFML Documentation: https://www.sfml-dev.org/documentation/  
- C++ Reference: <https://en.cppreference.com/>

**UML**