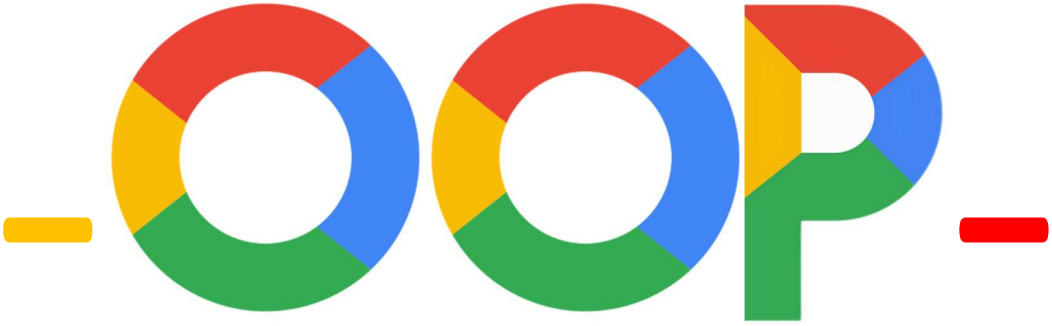

Course: CS1004 Object Oriented Programming.

Instructor: Lecturer Sara Rehmat.

Submitted by: Muhammad Rehan & Hassan Sardar

Roll no: 22P-9106 | 22P-9108

Class: BSE-2A

Date: May 17th, 2023.

Project Report

Department of Computer Science

Project Title:

Flappy Bird: An OOP Approach to Game Development 

Introduction:

We are thrilled to present our project submission report for "Flappy Bird: An OOP Approach to Game Development." This project has allowed us to delve into the game development world while mastering object-oriented programming (OOP) principles and exploring advanced concepts in C++. Our Team, consisting of Muhammad Rehan and Hassan Sardar, has poured their efforts into creating an engaging and visually appealing game, leveraging industry-standard tools and methodologies.

Project Overview:

"Flappy Bird: An OOP Approach to Game Development" focuses on building a Flappy Bird game using C++ and adhering to OOP principles. We have employed the SFML library (Simple and Fast Multimedia Library) to create an intuitive and user-friendly interface. The Clion IDE has been our primary development environment, enabling seamless coding. Utilizing Git and GitHub, we have employed version control and collaborative workflows to enhance project efficiency. The game encompasses a scoring system and interactive gameplay elements to ensure an immersive player experience.

Setup Instructions:

1. Clone the Repository:

- Open a terminal or Git Bash.
- Navigate to your desired directory.
- Execute the following command: git clone https://github.com/Rohtanza/Rohtanza-Flappy-Bird-An-OOP-Approach-to-Game-Development

2. Install Dependencies:

- Ensure that SFML and its dependencies are installed on your system. If not, kindly refer to the official SFML documentation for installation instructions tailored to your operating system.

3. Build and Run the Project:

- Launch your preferred C++ IDE (e.g., CLion).
- Configure the project settings to link the SFML library correctly.
- Build the project.
- Once the build is successful, run the game.
- https://youtu.be/37hXfy-Q8tY- Watch a demo of the game in action!

Game Instructions:

Control the bird's flight using the left mouse button to navigate the pipes. The objective is to pass through as many pipes as possible without colliding. Each successful passage through a pipe will increment the score. Collisions with pipes or the ground will result at the end of the game. Press any key or click the mouse to restart the game after it ends.

Technical Explanation:

The Game class is the central game controller, managing the game's overall flow and updating its state. It utilizes the GameState class, which represents different states of the game, such as the main menu, gameplay, game over, etc. Each GameState subclass encapsulates the logic and behaviour specific to that state. The game loop is the heart of any game, responsible for continuously updating the game state and rendering the graphics. It ensures a smooth and interactive gaming experience. In this project, the game loop is implemented in the Game class.

The game loop consists of three main steps:

1. Process Input:

The InputManager class captures user input events like mouse clicks. The input events are processed within the game loop, allowing for interaction with the game.

2. Update Game State:

The Game class updates the current GameState based on the input received and the game's logic. For example, it updates the bird's position, checks for collisions with pipes, and handles scoring.

3. Render Graphics:

The Game class renders the current game state to the screen, displaying graphics and sprites. This involves drawing the bird, pipes, background, and other elements using the SFML library.

Conclusion:

In conclusion, "Flappy Bird: An OOP Approach to Game Development" has been an exciting and educational project. By exploring the codebase and focusing on introductory classes such as InputManager, AssetManager, Game, and GameState, and understanding the game loop, we have gained a deeper understanding of game engine architecture, OOP principles, and the technical aspects of game development.