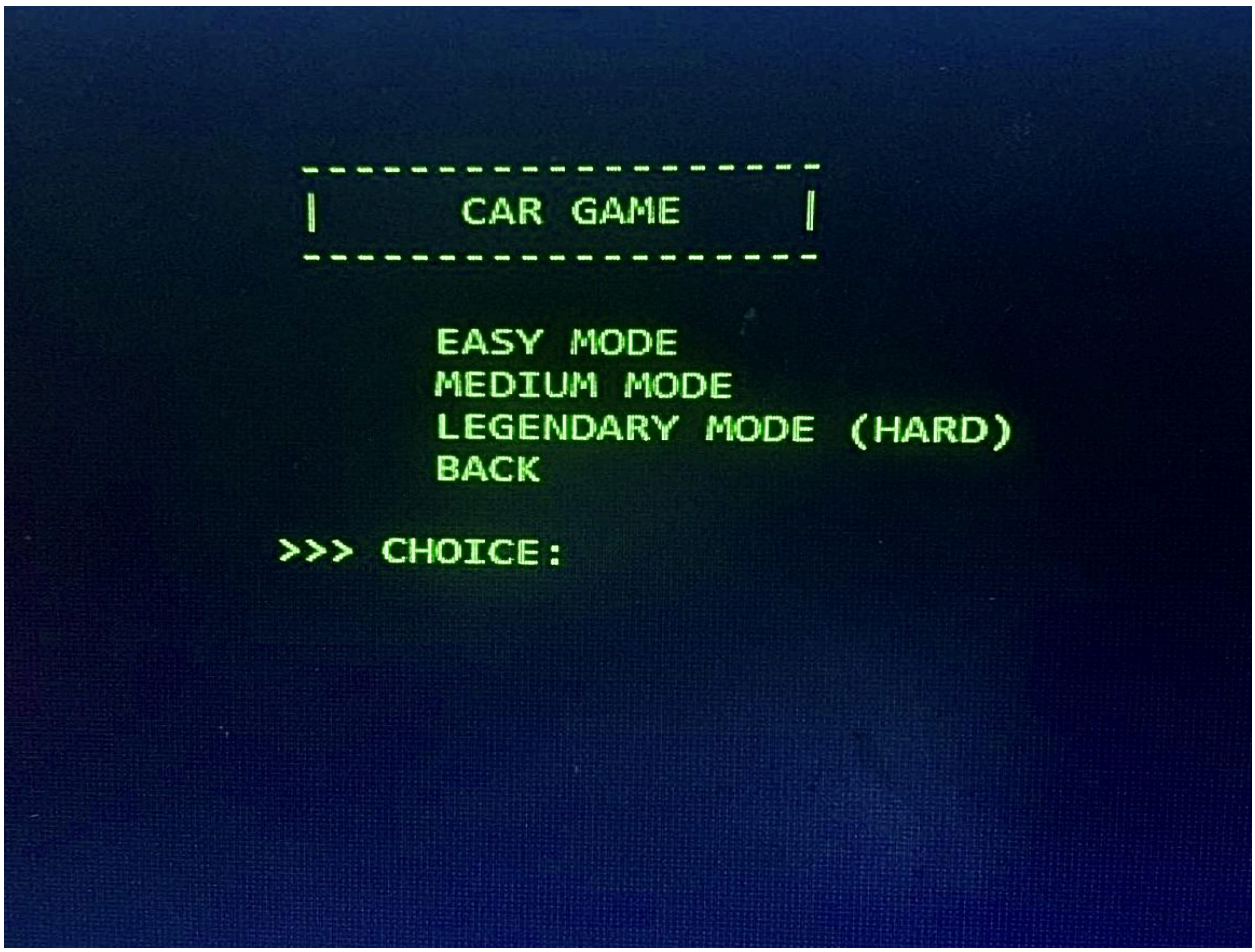
OBJECT ORIENTED PROGRAMMING

CAR GAME

FINAL PROJECT - 2024



Members:

- **Rayyan Merchant [23K-0073]**
- **Riya Bhart [23K-0063]**
- **Syeda Rija Ali [23K-0057]**

ABSTRACT

This project focuses on the creation and execution of a captivating car racing game leveraging fundamental and advanced Object-Oriented Programming (OOP) principles. Drawing inspiration from classic arcade games, the objective entails crafting an immersive gaming experience where players navigate a dynamic race track, competing against computer-controlled opponents or other players. The game's core mechanics involve controlling a virtual car through various terrains, evading obstacles, and strategically overtaking competitors to secure victory. By employing OOP methodologies, such as encapsulation, inheritance, and polymorphism, the development process ensures modularity, extensibility, and maintainability of the codebase. Through meticulous design and implementation, the project aims to deliver an engaging gaming experience that resonates with both novice and seasoned players, showcasing the versatility and power of OOP in game development

Project Overview

- **Objective:** The project aims to create a car racing game implemented using C++ and leveraging Object-Oriented Programming (OOP) principles.

- **Game Modes:**
    - The game offers three modes: Easy, Medium, and Hard.
    - Players can select their preferred mode from the main menu.

- **Game Mechanics:**
    - Players control a car on a race track, moving left (A) and right (D) to avoid collisions with enemy cars.
    - Each mode offers varying difficulty levels, with faster enemy cars and more obstacles in higher difficulty modes.

- **Scorekeeping:**
    - The game keeps track of the player's score, incremented each time the player successfully avoids a collision.

- **User Interface:**
    - The game provides a simple text-based user interface utilizing the console window.
    - Instructions are displayed to guide the player on how to play the game.

- **Game Over:**
    - When the player collides with an enemy car, the game ends, displaying a "Game Over" message along with the final score.
    - Players have the option to return to the main menu or exit the game entirely.

- **Implementation Details:**
    - The game utilizes basic C++ features such as input handling, screen clearing, and text output to create a dynamic gaming experience.
    - OOP principles are applied through classes such as gamePlay, enemyClass, and subclasses (easyClass, mediumClass, hardClass) to manage game entities and behaviors.

Design Methodology:

- **Object-Oriented Design (OOD)**:
    - The project adopts an Object-Oriented Design approach, leveraging the principles of encapsulation, inheritance, and polymorphism to model game entities and behaviors.
    - The game is structured around classes representing different game components such as the player's car (gamePlay), enemy cars (enemyClass), and game modes (easyClass, mediumClass, hardClass).
- **Modularity**:
    - The codebase is modular, with each class responsible for a specific aspect of the game's functionality.
    - This modular design allows for easier maintenance, debugging, and extension of the game.
- **Abstraction**:
    - Abstraction is employed to hide implementation details and expose only relevant interfaces to interact with game entities.
    - For example, the gamePlay class abstracts the details of drawing the player's car and managing collisions, providing a clean interface for the game logic.
- **Encapsulation**:
    - Data and functionality are encapsulated within classes, ensuring that each class maintains its own state and behavior.
    - Private member variables and member functions are used to encapsulate internal details, preventing external interference and promoting data integrity.
- **Inheritance**:
    - Inheritance is utilized to establish relationships between classes and promote code reuse.
    - The easyClass, mediumClass, and hardClass classes inherit from both the gamePlay and enemyClass classes, inheriting their functionalities while also adding specific behaviors for each difficulty level.
- **Polymorphism**:
    - Polymorphism enables flexibility and extensibility in the codebase, allowing different classes to be treated uniformly through base class pointers or references.
    - The drawCar() and drawEnemy() functions are declared as virtual in the base classes (gamePlay and enemyClass) and overridden in the

derived classes (easyClass, mediumClass, hardClass) to provide specialized implementations based on the specific type of game entity.

- **Iterative Development**:
    - The game's development likely followed an iterative process, with continuous refinement and improvement based on feedback and testing.
    - Iterative cycles may have involved designing, implementing, testing, and refining various game features and mechanics to achieve the desired gameplay experience.

Future Enhancements:

1. **Graphics Upgrade**:
    - Implement graphical enhancements to improve the visual appeal of the game, such as adding textures, animations, and particle effects.
    - Transition the game from a console-based interface to a graphical user interface (GUI) for a more immersive experience.
2. **Multiplayer Support**:
    - Introduce multiplayer functionality, allowing players to compete against each other online or locally.
    - Implement features such as leaderboards, player rankings, and matchmaking to enhance the competitive aspect of the game.
3. **Customization Options**:
    - Provide players with the ability to customize their cars with different colors, designs, and upgrades.
    - Implement a progression system where players can unlock new customization options and upgrades as they progress through the game.
4. **Additional Vehicles and Environments**:
    - Introduce a variety of new vehicles with unique characteristics and handling traits.

- Expand the game's environments to include diverse landscapes, cityscapes, and themed tracks, offering a greater variety of challenges and visuals.
- Implement an achievement system with various milestones and challenges for players to accomplish.
- Reward players with in-game rewards, such as unlockable content, currency, or cosmetic items, for completing achievements and challenges.

Conclusion:

In conclusion, the development of the car game demonstrates the creative application of programming concepts to create an entertaining and engaging gaming experience. Through the utilization of C++ and console-based graphics, the game offers players the opportunity to navigate through challenging environments while avoiding obstacles and competing for high scores.

The project's implementation showcases various programming techniques, including input handling, collision detection, and game state management, highlighting the versatility and practicality of object-oriented programming principles. Despite encountering challenges such as input responsiveness, collision accuracy, and performance optimization, the development process fostered problem-solving skills and encouraged iterative refinement to deliver a polished final product.

Overall, the car game project exemplifies the fusion of creativity, technical proficiency, and iterative development processes inherent in game development. It serves as a testament to the potential of programming as a tool for bringing imaginative concepts to life and delivering immersive entertainment experiences to players worldwide.