Practical Gaming 2024

# Name of Student: Rani Wheeza

# T Number: T00244793

# Name of Project: Speedster

# Gameplay

The player controls a high-speed car on both city and highway tracks. The objective is to complete laps in the shortest time possible while collecting speed boosters scattered along the route. The player uses the **arrow keys or WASD** to steer, accelerate, and brake. Speed boosters are randomly generated across the track and collecting them temporarily increases the car’s top speed or decreases their speed, or they could get a health booster that would add on to their health bar.

**Keys:**

* W / Up Arrow: Accelerate
* S / Down Arrow: Brake/Reverse
* A / Left Arrow: Turn Left
* D / Right Arrow: Turn Right

**The game features:**

* Speed and health boosters that enhance the racing experience.
* Collision detection with the environment to ensure realism.
* Two environments: city and highway tracks and a practice track.

# Coding

* Frame Rate Independence

I used Time.deltaTime to ensure all movement calculations are frame rate independent. This means the game runs smoothly across different devices regardless of frame rate.

* Interfaces

I used interfaces to define behavior that can be shared across game objects. For example, speed boosters implement a simple interface like ICollectible, which allows flexibility in how objects respond when collected.

* Inheritance

Inheritance is used in the car controller logic. City and practice cars derive from a base CarController class that contains common physics and movement functions, reducing code duplication. Each has its own assigned movement script to prevent the code from messing with the movement of each car.

* Case pattern

CamelCase and PascalCase are used throughout the codebase for naming consistency.

* + Variables: currentSpeed, boostAmount
  + Methods: IncreaseSpeed(), ResetSpeed()
  + Classes: CarController, SpeedBooster

Observer Pattern

Used in the UI update system. For example, when a health booster is collected, a Healthbooster script notifies the healthBar UI to update.

* Polymorphism
* Communication between scripts/game objects

The player car communicates with the game manager and UI controller through script references. For example, when a booster is collected, it sends a message to the BoosterManager script to temporarily activate the booster for 5 seconds for the speed boosters.

* Instantiation and Prefabs

Speed boosters were created as prefabs and instantiated randomly at game start. This is done via a spawning script that ensures boosters appear in diverse positions each round with the help of spawn points placed throughout the game.

* Magic Numbers

I avoided magic numbers by defining constants or public variables with descriptive names like boostAmount, boostDuration, etc., allowing easy adjustment through the Unity Inspector.

* Model Animation

At the moment, the game does not feature rigged character animations,

* Self made models and or animations

The car models were modified from asset packs, and the tracks (city and highway) were fully imported from the Unity Asset Store. I didn’t create custom models but assembled and edited components to fit the game's theme and functionality.

* Interactions between objects/scripts
  + Boosters interact with the player car through trigger colliders. When collision happens, they notify the car controller to increase or decrease speed or add health and destroy themselves afterward.
* Propper code placement

Code is separated into clear components:

* + Movement logic is in CarController.
  + Booster logic is in SpeedBooster.
  + Spawning is managed by BoosterManager.
  + UI updates are handled in UIManager.
* Code repetition
* Feature 1 (Boosters)
  + Implemented speed boosters that increase the car's speed temporarily when collected
  + Implemented speed boosters that increase the car's speed temporarily when collected
  + Implemented speed boosters that increase the car's speed temporarily when collected
* Feature 2 (Random Spawning System)
  + Boosters are randomly generated across the track using a spawn script to make gameplay more dynamic.
  + BoosterManager to assign random boosters to random spawn points throughout the game.
* Feature 3 (Dual Track Selection)

The game includes two separate tracks a city and a practice environment, both offering different driving challenges. Players can choose which track to race on before the game starts.