**SOFT10101 Computer Science Programming**

**Coursework 2020/21**

Title: **Crazy Racer – Endless Race Game**

Student ID: N0954970

Name: Adeeb Sait

**Specification**

This project is a video game where the user must keep changing lanes with their car to avoid oncoming traffic and collect coins to gain a score.

Crazy Racer is a 3D game that has been completely programmed in C++ and developed with the help of Unreal Engine 4. *Unreal Engine is a game engine developed by Epic Games that support scripting in C++ to develop new gameplay elements.*

The game project file EndlessRace can be accessed and edited with the help of Visual Studio, but to edit the game file, Unreal Engine 4.26.1 needs to be used. The game has also been packaged to run with any Window OS systems. The program will run a new window console without any further installations.

When the game is loaded, the user is shown the main menu with the option to Start or Exit. As soon as the user selects Start, the game starts running and ends with a death-screen when collided with any obstacle except the coins. The death screen then gives the user an option to restart or go back to the main menu.

Only two input keys are used in this program to control the player’s car: “A” to shift left and “D” to shift right. The player’s car will not be able to move beyond the specified lanes.

**Design and Implementation**

**Program Design**

Each component that is used in the game has been bound appropriately in the CPP files of the project. All components are programmed to display the graphics accurately in each frame according to the player’s input. The game has been built with separate modules for every gameplay element.

Separate source codes exist for the functioning of each Actors along with other components in this game. The three actors in this game are Player, Obstacles and Coins. These codes are run on all the actors and components at regular intervals to perform tasks and ensure the smooth running of the game. These runs are controlled within Tick() and the initialisation of all components during BeginPlay() call.

Coin: The mesh and collision component are attached using UPROPERTY. For each frame tick, FRotator function is used to make the coins spin slowly.

EndlessRace: Implements gameplay module. Mandatory for the running of the game.

EndlessRaceGameModeBase: Attaches Spawns the road in the form of tiles every frame. Once the new piece of road is spawned, the next spawn point is attached to the end of it as a vector so that a straight road can be made endlessly.

MyCharacter: Attaches the camera component with the user’s car, controls the scores and changes lanes. TimeLine function is used for changing the lane in a particular time using Linear Interpolation(Lerp) according to the player’s input. OnOverLap checks for any collisions between Player and any other actors; if the Player overlaps or collides with a coin, the point increases and, that coin is also destroyed immediately. The total points keep getting updated on the UI overlay that displays the score constantly. If the Player collides with any of the obstacle cars, the game stops running and, the DeathScreen box containing the Restart and Main Menu button is displayed.

ObstacleCharacter: Instead of the player’s car moving frontways, all the obstacles keep moving towards the user. AddMovementInput controls this movement.

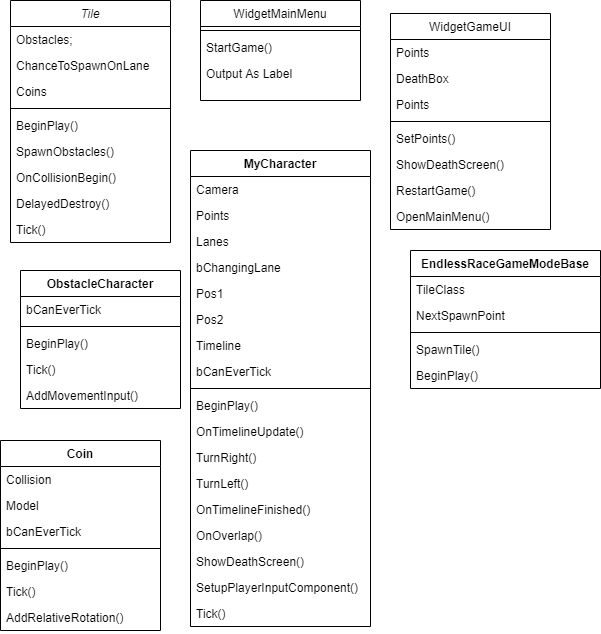
Tile: Controls the spawning of all the obstacles and coins on the road tiles. Each end of the road tile has an invisible trigger attached so, when the Player collides or overlaps with the trigger, new obstacles and coins get spawned. The location of the spawning occurs randomly for both the actors. Only a maximum of two cars gets spawned adjacently to ensure that the player can evade the obstacles.

WidgetGameUI: Controls the UI that is always displayed while the game is being played. This UI holds the DeathBox that contains the buttons for Restart and Main Menu.

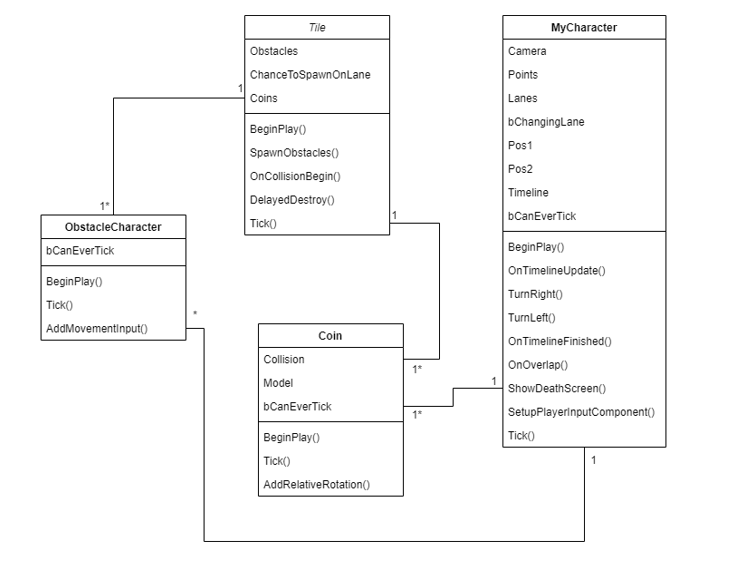
WidgetMainMenu: Controls the Main Menu UI that displayed when the game begins at default. Contains buttons to Start or Exit the game.

**Classes**

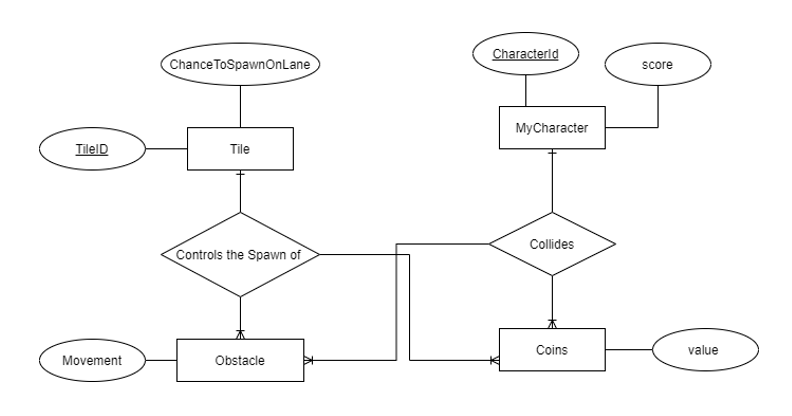
This project contains seven classes, as follows:



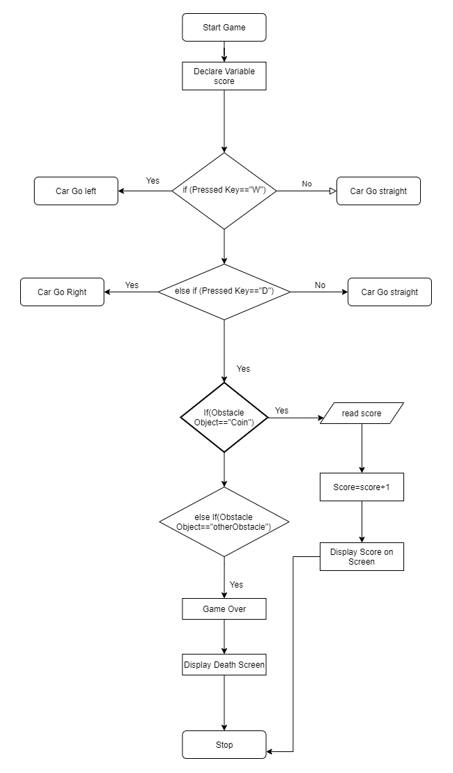
**Class Diagram**



**Entity Relation Diagram**



**Flowchart**



**File Structure**

The last score of the game played before exiting is always saved in a text file called “save.txt” and is available inside the game directory just in case the user wants to recollect their score later.

**Unreal Engine Properties**

* CoreMinimal.h – Included to make sure certain components from the hierarchy do not go missing during build.
* Kismet/XX.h – Contains useful gameplay utility functions.
* Components/XX.h – Special type of Object that can be used to attach to the Actors as sub-objects. Useful for representing certain behaviours, display visuals or play sounds.
* Misc/FileHelper.h – File input/output.

**Test Plan**

The following is a text plan for the project:

|  |  |  |
| --- | --- | --- |
| **Action** | **Test Method** | **Success Criteria** |
| Start game | Click on Start button | *The program can run the game as soon as the player presses the button. Starts with the player in the centre lane.* |
| Move right | Press a or A | *Player’s car shifts right. When in the rightmost lane, the car is restricted from going further right.* |
| Move left | Press d or D | *Player’s car shifts left. When in the leftmost lane, the car is restricted from going further left* |
| Restart game | Click on Restart button | *Program restarts directly without leading to the Main Menu UI. The position of the player’s car also resets.* |
| Main Menu | Click on Main Menu button | *The game does not run, and the Main Menu UI is displayed correctly.* |
| Exit | Press Esc or Exit button | *Program ends.* |

**Functional Testing**

****

**Demo**

