2023

Project Report

AZGHAN AHMAD 22I_2667
ARSLAN SHABBIR 21I-1739

Table of Contents

Car Racing Game

1. Introduction:

The 2-D Car Racing Game is a console-based application developed in C++. It offers an engaging and interactive experience where players control a car through a dynamically generated obstacle course. The game includes features includes different features scoring, levels, obstacle avoidance, and high-score tracking. Basically this game is implemented in order to efficiently use data structures concepts in the game.

2. Features:

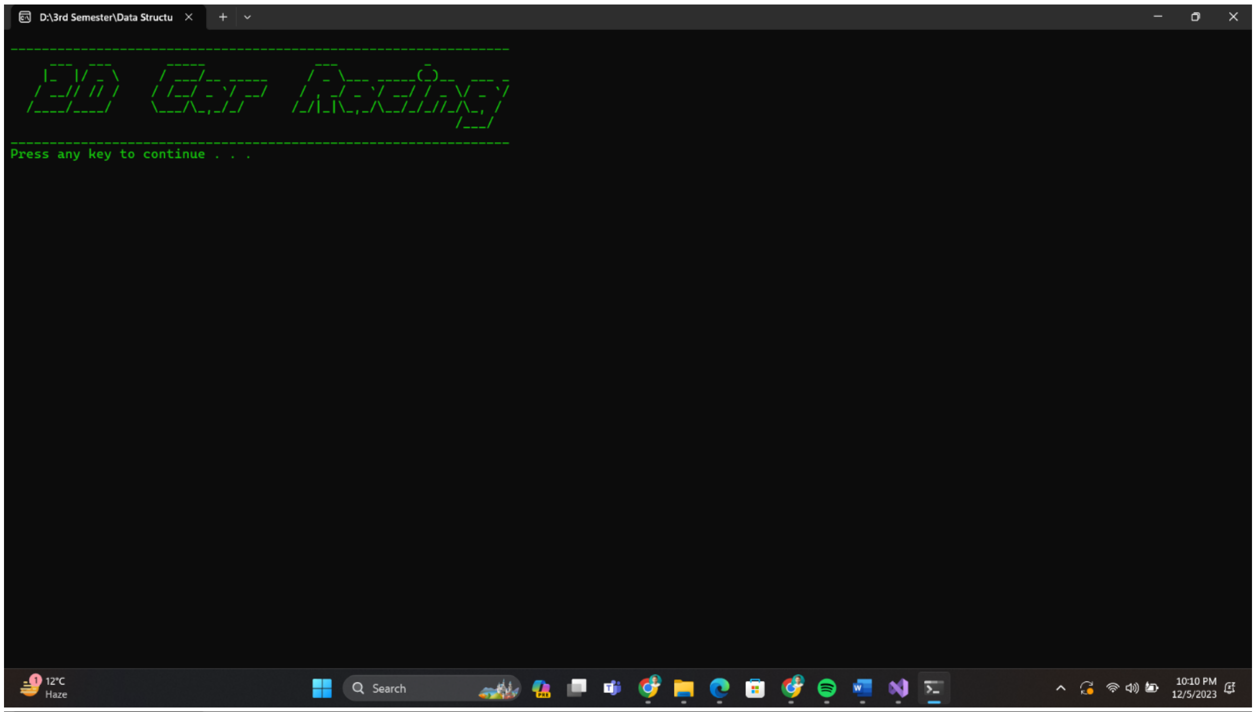
Obstacle Generation: The game generates obstacles randomly, including walls of the grid, creating a challenging environment for the player. This engages the user towards the game.

Scoring System: Players earn points by collecting coins represented by the letter 'S' on the grid. The score is displayed in real-time at the top of the game window.

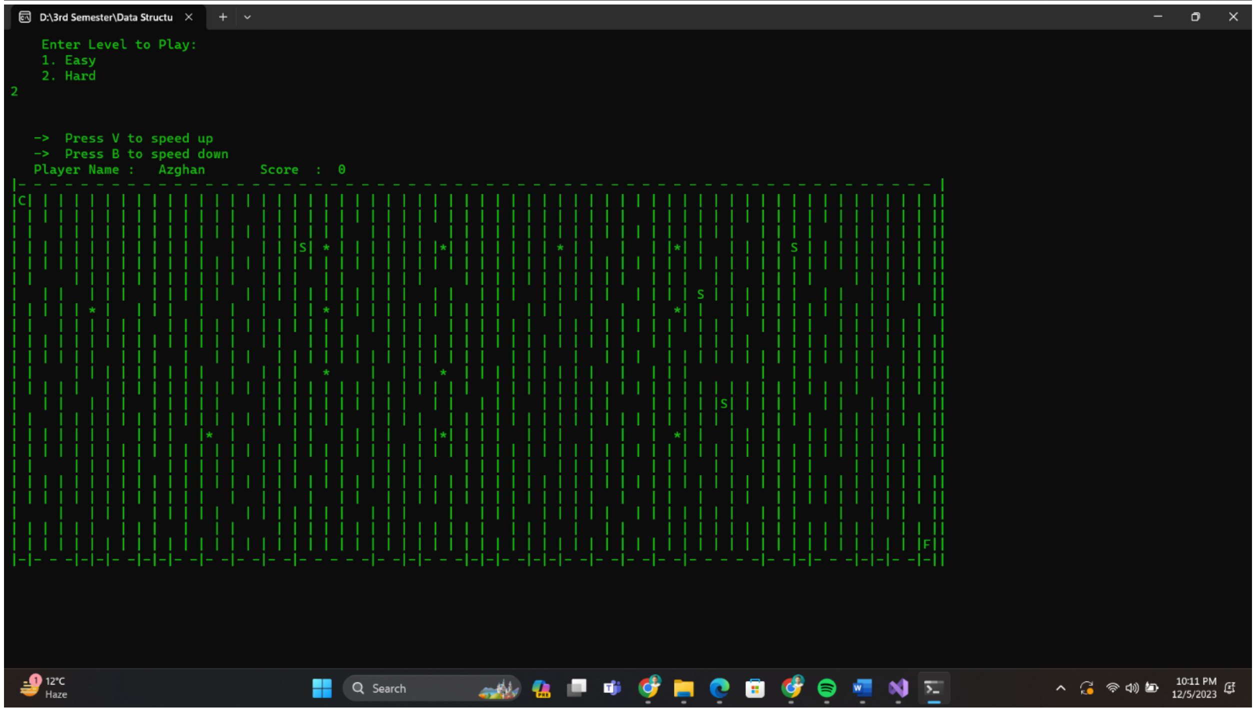
Speed Control: Players can adjust the speed of the car using the 'V' and 'B' keys, allowing for a customizable gameplay experience.

High Score Tracking: The game maintains a binary tree data structure to track and display the highest scores achieved by players.

Different Levels of Difficulty: The game offers varying levels of difficulty, providing players with options such as "easy" and "hard." This feature allows users to play the gameplay experience to their skill level or preference. “Easy” is for beginners and “Hard” level is for the expert user.



```
-------------------------------------------------------------------
    ___  ___      _____        ___             _
   |_  |/ _ \    / ___/__ _____  / _ \___ _____(_)__  ___ _
  / __// // /   / /__/ _ `/ __/ / , _/ _ `/ __/ / _ \/ _ `/
 /____/____/    \___/\_,_/_/   /_/|_|\_,_/\__/_/_//_/\_, /
                                                     /___/
-------------------------------------------------------------------
Press any key to continue . . .
```



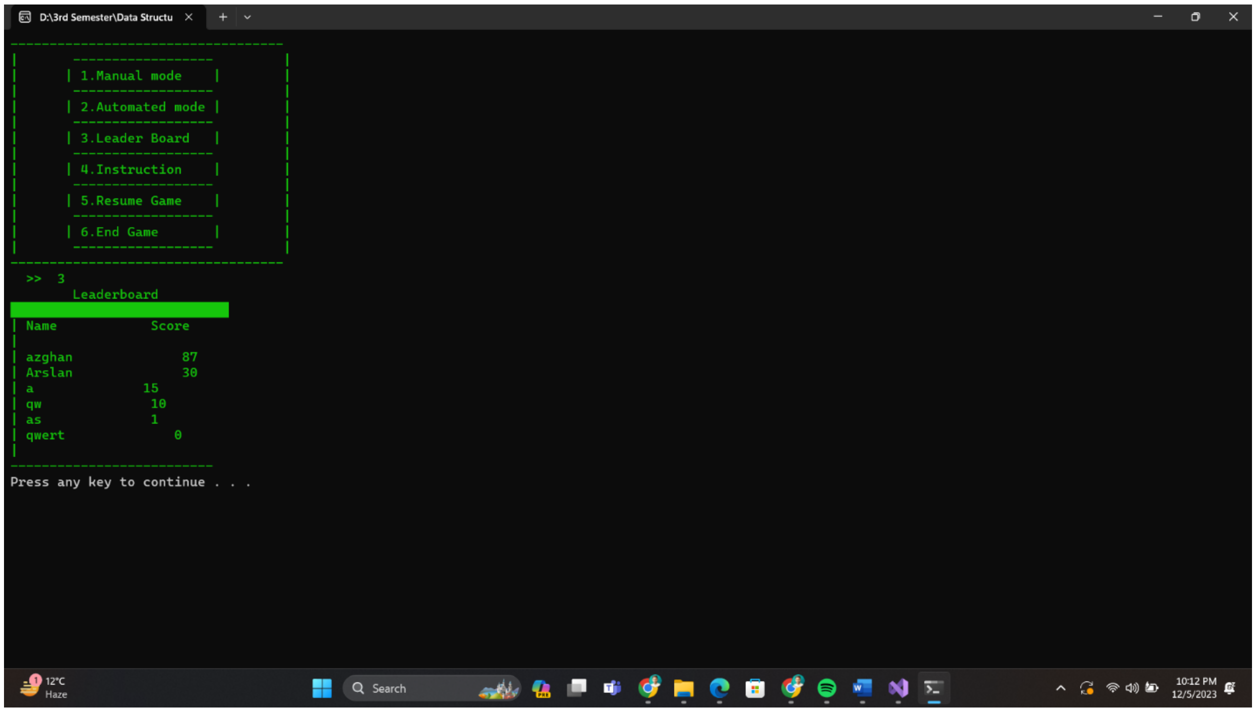
```
    Enter Level to Play:
    1. Easy
    2. Hard
2


   ->  Press V to speed up
   ->  Press B to speed down
   Player Name :   Azghan        Score  :  0
```

3. Game Logic:

The game's logic revolves around the movement of the car on a grid-based system. This grid is implemented from edges and nodes. Players control a car within this grid and navigates through obstacles, collects coins represented by “S”, and tries to reach the finishing point. The speed of the car is adjustable and can be increased and decreased, and the game ends when the player reaches the destination. The player can easily resume the game, enabling players to pick up where they left off in a previous session.

4. Binary Tree for Leaderboard:

A binary tree structure is used to store and manage scores. Each node in the tree represents a player's score, with the left child containing scores lower than the parent and the right child containing higher scores. This is used to keep track of the score at every step. Also, Leaderboard is maintained through this Binary tree that displays all the players that played the game after reading the game players and their scores from the file saved in the user computer.



```
-----------------------------------
|       ------------------        |
|      | 1.Manual mode    |       |
|       ------------------        |
|      | 2.Automated mode |       |
|       ------------------        |
|      | 3.Leader Board   |       |
|       ------------------        |
|      | 4.Instruction    |       |
|       ------------------        |
|      | 5.Resume Game    |       |
|       ------------------        |
|      | 6.End Game       |       |
|       ------------------        |
-----------------------------------
  >>  3
        Leaderboard

| Name             Score
|
| azghan               87
| Arslan               30
| a               15
| qw               10
| as               1
| qwert               0
|
--------------------------
Press any key to continue . . .
```

5. Obstacle Implementation using Queue:

The obstacle class implements a queue-based system for managing stones and sticks in the game environment. Utilizing linked lists with `nod` as the node structure, stones and sticks are enqueued at the rear of their respective queues during dynamic generation, with randomized positions enhancing gameplay variability. The queue's FIFO order is employed for efficient obstacle handling, and the deque_stone() and deque_stick() methods facilitate obstacle removal from the front of the queues, mimicking the car's movement. The Is_stone_quey_empty() and Is_stick_quey_empty() methods check for queue emptiness, aiding in conditional checks. The randomized count and positions, along with the dynamic generation of obstacles, contribute to an engaging and challenging gaming experience, making each session unique. The queue-based approach ensures organized and efficient obstacle management in the grid-based game environment.

6. Adding Score:

In game scoring mechanism, this approach is implemented to keep track of the player's score. As the player collects coins represented by the letter 'S' on the grid, each successful collection operation results in a score increment. This dynamically adjusts to reflect the current score, pushing and popping elements as the player gains or loses points. This stack-based scoring system adds a layer of flexibility and efficiency to the score tracking process, facilitating real-time updates and contributing to the overall interactive and dynamic nature of the gameplay.

7. Obstacle and Graph Implementation:

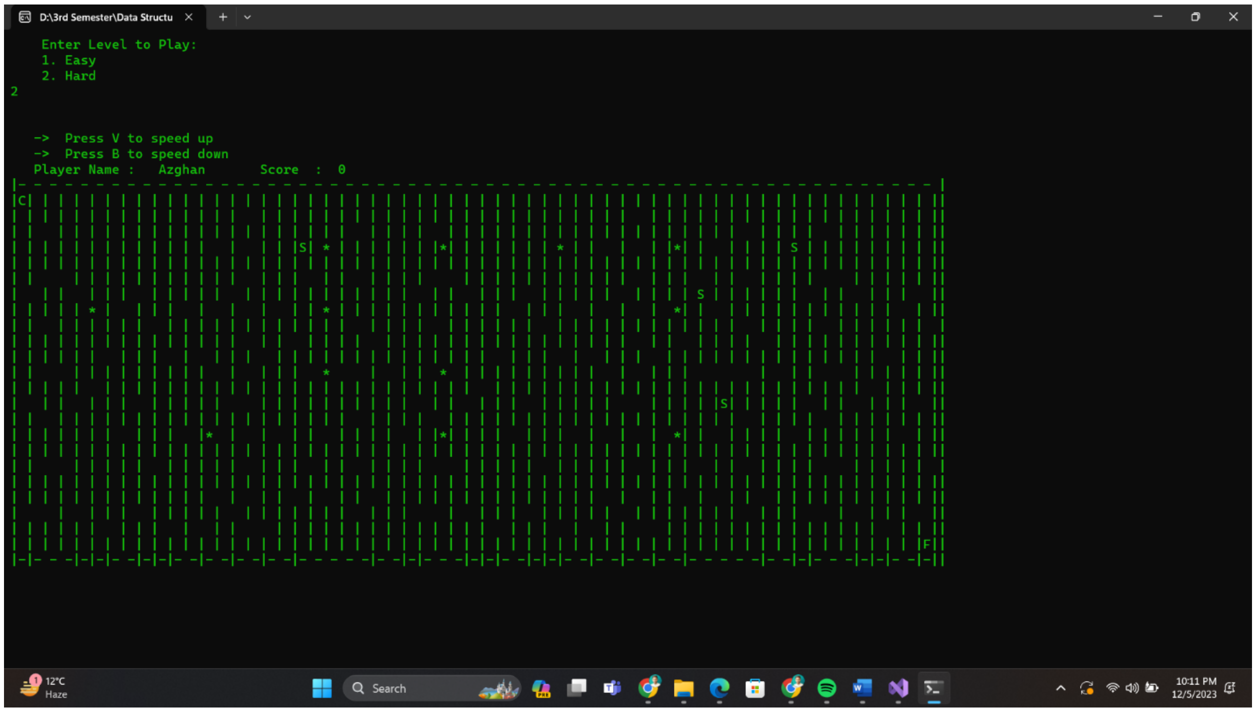
Obstacle Class: The obstacle class manages the generation and insertion of stones and sticks. The obstacles are strategically placed in a manner that introduces challenges for the player. The placement is randomized, adding an element of unpredictability to the gameplay and ensuring that each gaming session offers a unique and engaging experience.

Graph Class: The Graph Class serves as a representation of the game grid. It plays a fundamental role in managing the structure and functionality of the grid. This class is responsible for tasks such as displaying the grid, creating nodes (points on the grid), and establishing connections between these nodes. Nodes are crucial for defining the potential paths the car can take and determining the locations where obstacles, such as stones and sticks, can be placed.

8. Game Controls:

Car Movement: Players control the car using the 'A' to move “left”, 'S' to move “right”, 'W' to move “up”, and 'Z' to move “down” keys.

Speed Control: The game incorporates a speed control feature to enhance player engagement and customize the gameplay experience. By pressing the 'V' key, players can increase the speed of the car and the 'B' key is used to decrease the car's speed.

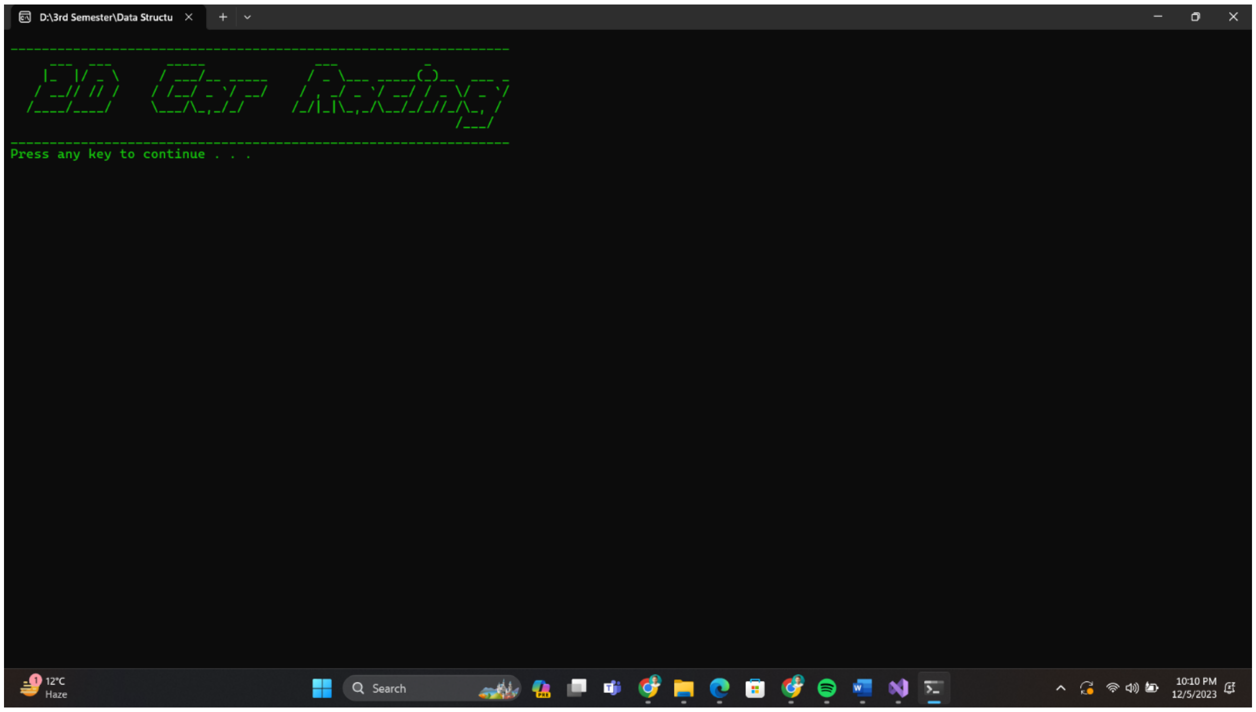


```
    Enter Level to Play:
    1. Easy
    2. Hard
2


   ->  Press V to speed up
   ->  Press B to speed down
   Player Name :   Azghan       Score  :  0
```

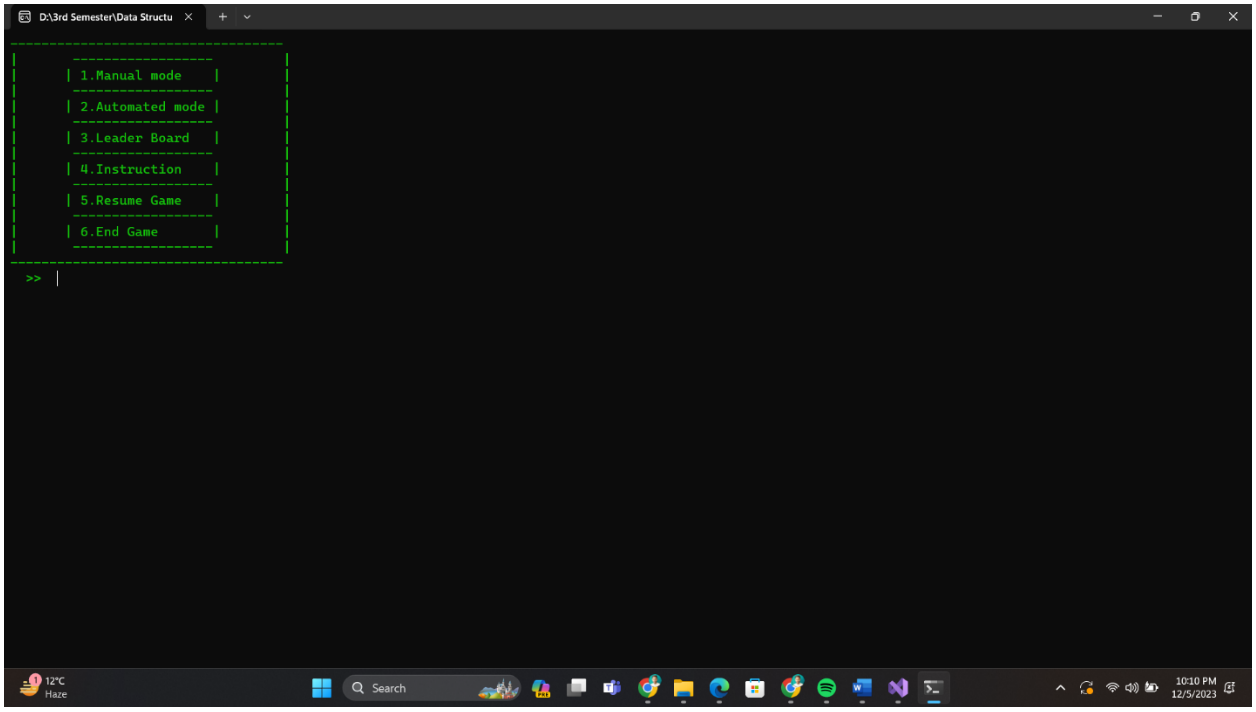
9. Menu and User Interface:

The game includes a menu system with various options:

- Manual Mode (Player controls the car)
- Automated Mode (Car moves automatically)
- Leaderboard (Displays the score of all the players)
- Instructions (Provides information on game controls)
- Resume Game (Resumes the game)
- End Game (Exits the game)



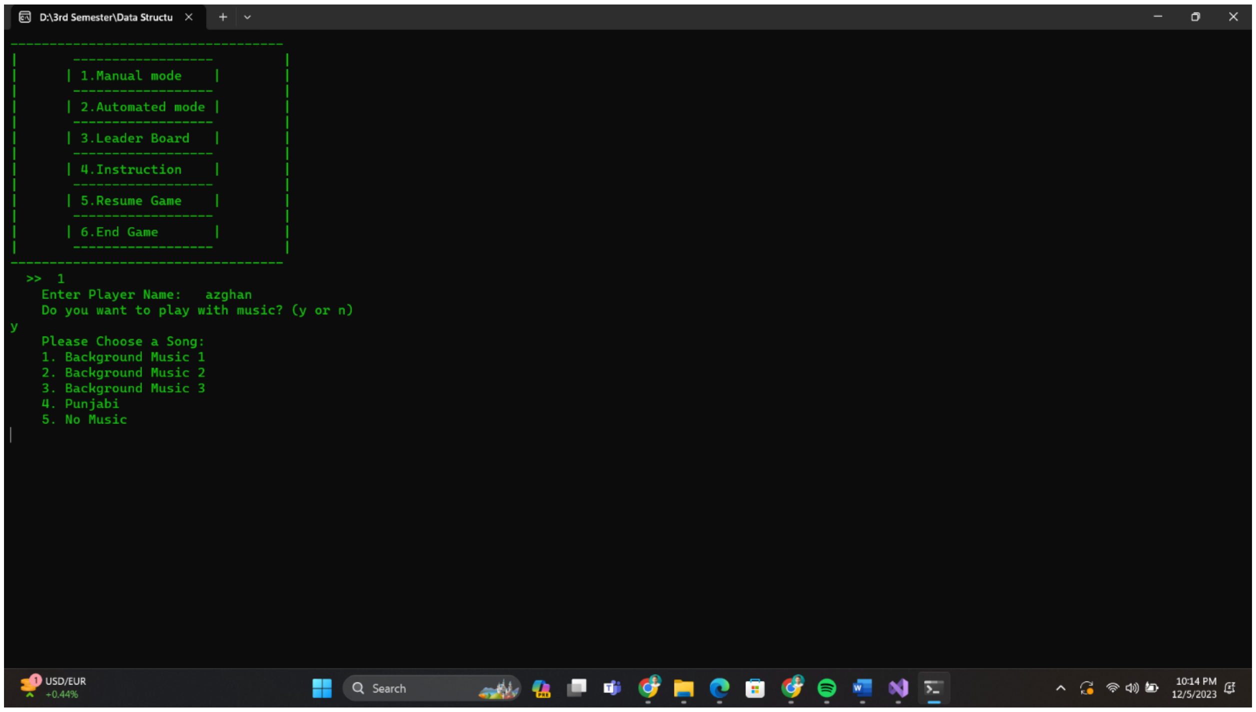
```
-----------------------------------------------------------------
   _____ ____     ______              ____             _
  |__  // __ \   / ____/___ ______   / __ \____ ______(_)___  ____ _
   /_ </ / / /  / /   / __ `/ ___/  / /_/ / __ `/ ___/ / __ \/ __ `/
 ___/ / /_/ /  / /___/ /_/ / /     / _, _/ /_/ / /__/ / / / / /_/ /
/____/_____/   \____/\__,_/_/     /_/ |_|\__,_/\___/_/_/ /_/\__, /
                                                           /____/
-----------------------------------------------------------------
Press any key to continue . . .
```



```
-----------------------------------
|       ------------------         |
|      | 1.Manual mode    |        |
|       ------------------         |
|      | 2.Automated mode |        |
|       ------------------         |
|      | 3.Leader Board   |        |
|       ------------------         |
|      | 4.Instruction    |        |
|       ------------------         |
|      | 5.Resume Game    |        |
|       ------------------         |
|      | 6.End Game       |        |
|       ------------------         |
-----------------------------------
  >>
```

10. Sound and Music:

The game enriches the overall gaming experience by integrating sound effects and background music. Sound effects and music contribute to creating a more immersive and engaging atmosphere, enhancing the player's connection with the virtual environment. The Play Sound feature is employed as a mechanism to play audio files during specific events within the game, such as the start of the game, the occurrence of a game-over situation, or when certain in-game conditions are met.



```
-----------------------------------
|        ------------------         |
|       | 1.Manual mode    |         |
|        ------------------         |
|       | 2.Automated mode |         |
|        ------------------         |
|       | 3.Leader Board   |         |
|        ------------------         |
|       | 4.Instruction    |         |
|        ------------------         |
|       | 5.Resume Game    |         |
|        ------------------         |
|       | 6.End Game       |         |
|        ------------------         |
-----------------------------------
  >>  1
    Enter Player Name:   azghan
    Do you want to play with music? (y or n)
y
    Please Choose a Song:
    1. Background Music 1
    2. Background Music 2
    3. Background Music 3
    4. Punjabi
    5. No Music
```

11. Conclusion:

In conclusion, the Car Racing Game is an entertaining console-based application with dynamic obstacles, adjustable speed, and a scoring system. The implementation of a binary tree for high scores, along with obstacle and graph classes, contributes to a well-organized and enjoyable gaming experience. The inclusion of sound effects and music adds an extra layer of immersion to the game.