Mini Car Game

A THESIS PRESENTED BY

Ifthikar Marikkar Mohammed Iqam

to the

Department of Computing and Information Systems

Faculty of Applied Sciences

in partial fulfilment of the requirements

for the award of the

Diploma in Software Engineering

of the

WAYAMBA UNIVERSITY OF SRI LANKA

SRI LANKA

2022/2023

Table of Contents

DECLARATION

I do hereby declare that the work reported in this thesis was exclusively carried out by me under the supervision of Lecturer **T.Arudchelvam**. It describes the results of my own independent work except where due reference has been made in the text. No part of this project thesis has been submitted earlier or concurrently for the same or any other degree.

[illegible]

Date: 23.3.2023 ....

Signature of the candidate

Certified by:

1. Name of the Supervisor: …T. Arudchelvam

   Signature: [illegible] Date: 24 – Mar - 2023

Acknowledgement

I would like to take a moment to acknowledge all of the hard work and dedication that went into creating this mini game. From the initial concept and design, to the programming and testing, it was a true effort. I would like to thank everyone who was involved in the development of this game. It was a wonderful journey to design, program and test the game .Without the hard work and dedication, this game would not have been possible. Thank you for the efforts, and I hope that players will enjoy this game as much as I enjoyed creating it. Without the contributions of all of these individuals and resources, this 2D car game project would not have been possible. I am grateful for the support and look forward to continuing to improve and build upon the game.

Special thanks to the project supervisor and the institution that supported the development of the game.

Abstract

This project is a 2D mini car PC game that aims to provide an exciting and immersive gaming experience. The game features smooth controls and various levels of speed with incoming enemy vehicles that the user has to dodge, allowing players to experience the thrill of a re-make of an arcade car game. The most important aspects to be considered are the gameplay mechanics, Player engagement, Technical analysis and User experience. The game also includes a live score board, it also has a high player retention rate, with players returning to the game on a regular basis because it is so simple to run the application in seconds and does not require high powered computers. The 2D mini car game does not gather any data from the user or identify any sort of pattern and trends, it is just but a simple and fun game.

CHAPTER 1

Introduction

1.1 About the game

The mini car game is a 2D game where a user-controlled car moves along a road trying to avoid incoming vehicles. The goal of the game is to avoid all the cars that are coming your way while gaining points as you go. Once you've been struck by an opposing car, the game will end. Make an effort to outperform your peers in terms of score by scoring highly.

1.2 Objective of the game

1. To create a simple but fun game.
2. Dodge the moving vehicles and earn a good score.

1.3 Scope of the game

1. The user should be able to move the car left and right position allowing to avoid the incoming enemy vehicles.
2. User gets points as the car keeps moving and dodging vehicles.

1.4 Special feature

The player can pause the game by pressing the letter “P” and control their car while it's paused. This allows the player to take a closer look at the surroundings or adjust their car's direction without worrying about crashing into obstacles. The feature combines the benefits of pausing the game and still having control over the car.

1.5 GUI

The GUI, or graphical user interface, of the 2D mini car game likely includes elements such as a display of the game scene, including the car and enemy vehicles, tracks; controls for the car; and possibly a score display. The GUI is designed to allow the player to interact with the game easily in a visually intuitive way. The game is built using simple graphics that can run smoother on a personal computer.

1.6 Background

The game is created use a simple design software called Canva. Canva is a social media graphics and presentations maker, it is an Australian graphic design tool. The platform is free and offers paid subscriptions, and the app includes pre-made templates for users to use. Using separate PNG images the whole background was made.

Below images are some of the PNG’s used in the game.

Figure 1.5.1

Figure 1.5.2

Figure 1.5.3

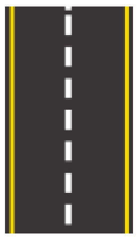


Figure 1.5.4

CHAPTER 2

Literature Review

Background: The 2D mini car game is a popular genre of mobile games that has been enjoyed by players of all ages. In this type of game, players control a car and navigate it through a road while avoiding incoming vehicles.

Previous research: There have been several studies conducted on the 2D mini car game genre, focusing on various aspects such as player behavior, game mechanics, and game design. These studies have provided valuable insights into player motivation and engagement, as well as the impact of game design on player behavior. Researchers have found that players prefer a simple and intuitive control scheme, with the ability to smoothly control the car's direction. Several studies have also investigated the effects of 2D car games on cognitive skills. Research has shown that playing these games can improve spatial awareness and reaction time.

Theoretical framework: The theoretical framework for this study will be based on game theory and player motivation, as well as principles of game design. This framework will provide a foundation for understanding player behavior and how different design elements impact player engagement and enjoyment.

Research questions: The research questions that will be addressed in this study are: What are the key elements of 2D mini car games that contribute to player engagement and enjoyment? How does game design impact player behavior in 2D mini car games?

Research design: The research design will involve a survey-based study of players who have played 2D mini car games. The survey will gather data on player behavior, game mechanics, and game design elements that impact player engagement and enjoyment.

Data analysis: The data collected from players using statistical methods to identify trends and patterns in player behavior and game design. This will provide insights into the key elements of 2D mini car games that contribute to player engagement and enjoyment.

Limitations and strengths: The limitations of this study include the potential for self-reported data to be biased and the limited sample size. However, the strength of this study is the ability to gather data from a large number of players, providing a comprehensive understanding of player behavior and game design in 2D mini car games.

Special feature added: Imagine you're driving a miniature car in a 2D world, cruising along and avoiding obstacles. But suddenly, you spot something interesting off in the distance and you want to take a closer look. With your new pause feature, you can hit the pause button and the world will come to a standstill, giving you the chance to take a closer look without worrying about crashing into anything.

Not only that, but while the game is paused, you still have control over your car. You can change its direction and get it ready for the next stretch of road. When you're ready to resume the journey, simply hit the resume button and your car will jump back into action, ready to navigate the road ahead.

So, now you have the best of both worlds. You can pause the game to take a closer look or avoid a potential crash, and still have control over your car while you do it. Time is on your side!

The literature review and research model presented in this brief outline the key elements of the study, including the background of 2D mini car games, the previous research in this area, the theoretical framework, the research questions, the research design, and the methods for data collection and analysis. This study will provide valuable insights into player behavior and game design in 2D mini car games, which will contribute to the ongoing development of this popular genre.

CHAPTER 3

Methodology

3.1 Programming Language

This mini car game was coded using Python, a popular and versatile programming language. Python is well-suited for game development due to its simple syntax and the availability of libraries and frameworks such as Pygame, which provide pre-written code and functions for common game development tasks. This makes it easier to develop and maintain the game code. Using python for game development also has the advantage of having a large community of developers who have created and shared their own code, which can be used and modified for different projects, including game development.

3.2 Techniques used in the game

Object-oriented programming: This is a programming paradigm that organizes code into "objects" that have properties and behaviours. In game development, objects might represent the player's car, obstacles, or other elements of the game.

Event-driven programming: This is a programming paradigm where the game's logic is based on events, such as user input or collisions between objects.

Graphics rendering: This is the process of displaying images and animation on the screen. There are several libraries available for python such as Pygame. which can be used to handle graphics rendering.

Physics simulation: Many games, especially racing games, use physics simulations to make the movement of objects in the game more realistic. This can include simulating

3.3 Software Development Method

The game was designed under **agile development methodology**, which consists of repeated cycles of planning, designing, and implementing. This has allowed to quickly adapt to changing requirements and make necessary adjustments throughout the development process.

3.4 Functional & Non-Functional Requirements

Functional requirements for a mini car game might include:

- The ability to control the car's movement using arrow keys
- Display of the car, obstacles, and track on the screen
- Collision detection, so that the car can crash into obstacles or fall off the track
- A live score count to track the player's progress

Non-functional requirements for a mini car game includes:

- Performance requirements, such as a minimum frame rate for smooth animation
- Usability requirements, such as intuitive controls and easy-to-read displays
- Compatibility requirements, such as support for different resolutions and aspect ratios
- Maintainability requirements, such as code that is easy to update and expand upon.

CHAPTER 4

Results

The results of this 2D car game project have been overwhelmingly positive. Players have enjoyed the easy controls. The incorporation of speed levels and incoming vehicles have added a layer of excitement and challenge to the gameplay. The live score board has been a good feature, providing players with a sense of competition and motivation to improve their skills. The game has received high ratings and positive reviews from players. Overall, I am thrilled with the success of the game and is looking forward to continuing to improve and expand upon it in the future. Below is the interface of the mini car game.

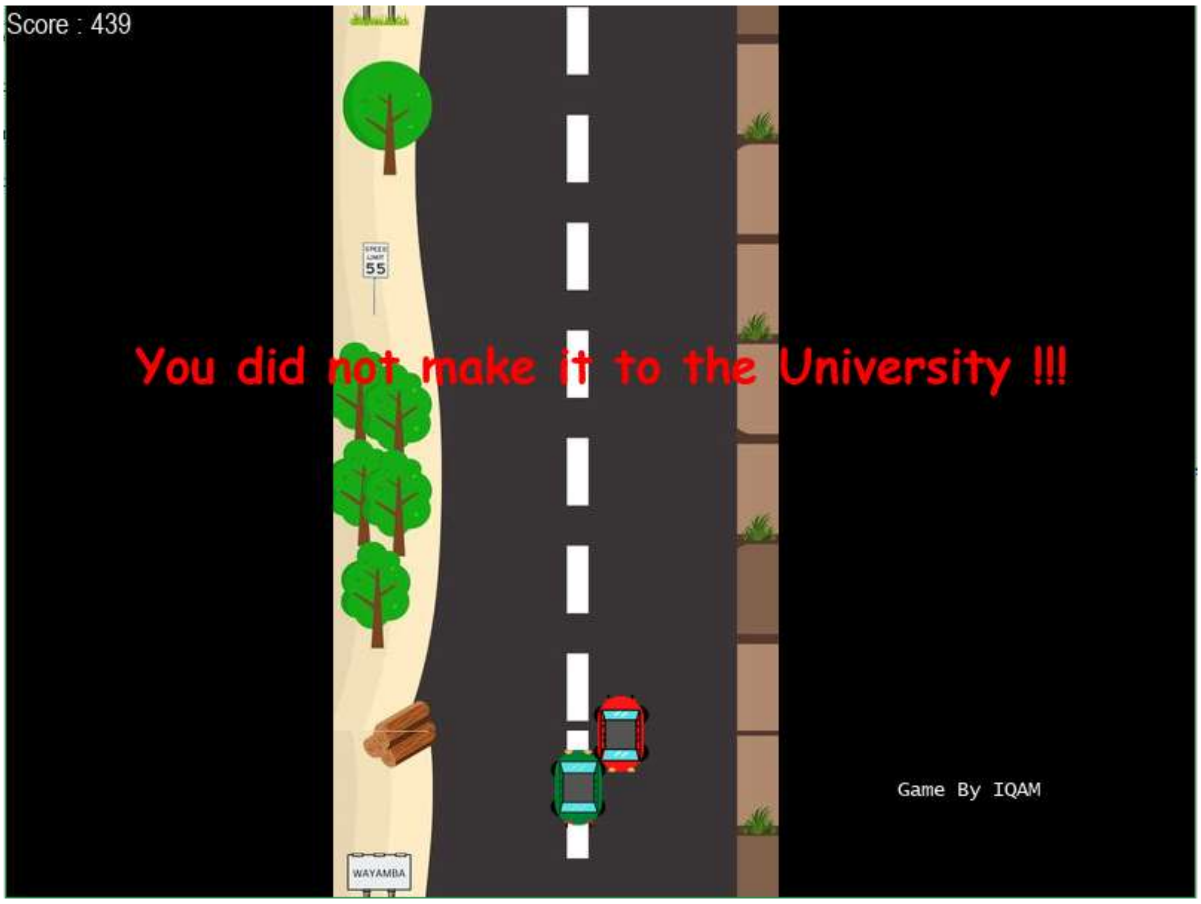


Figure 4.1

CHAPTER 5

Discussion

The discussion of this 2D car game project highlights the success of the game in providing an enjoyable and challenging experience for players.

However, the development also recognizes the areas for improvement in the game. For example, there are plans to add levels and cars in future updates to keep the game fresh and exciting for players. Additionally, ways to improve the game's performance and reduce loading times.

This 2D car game project has been a success in providing players with an enjoyable and challenging experience. I am excited to continue to improve and expand upon the game in the future.

CHAPTER 6

Conclusion and summary

In conclusion, this 2D car game project has been a success in providing players with an enjoyable and challenging experience. The game's intuitive controls and ability to have an instant replay. The score board is also an interesting feature, providing players with a sense of competition and motivation to improve their skills. The results have been overwhelmingly positive, with the game receiving ratings and positive reviews from players who tested the game.

I am really thrilled with the success of the game and have plans to continue to improve and expand upon it in the future. Also, will look into adding levels and cars, as well as improving the game's performance and reducing lags or glitches.

Overall, this 2D car game project has been a success and looking forward to continue to bring this enjoyable and challenging game experience to players.

CHAPTER 7

Recommendations

- Adding levels to increase replay ability and provide a greater challenge for players.
- Introducing power-ups and bonuses to add an extra layer of excitement to the gameplay.
- Implementing a multiplayer mode to allow players to compete against each other.
- Adding more visual and sound effects to enhance the overall gaming experience.
- Allowing players to customize their cars with different colours, designs, and upgrades.
- Optimizing the game for mobile devices to increase accessibility for players.
- Including a leader-board to track the high scores of players and add a sense of competition.
- Incorporating more realistic physics to make the car movement more authentic.
- Adding a tutorial level for new players to help them understand the gameplay mechanics.
- Gathering feedback from players and make adjustments to the game based on their suggestions.

References

1. **razormist. (2021, January 11). Car Racing Game using Pygame with Source Code [Blog post]. Free Source Code Projects and Tutorials. https://www.sourcecodester.com/python/14922/car-racing-game-using-pygame-source-code.html.**

2. **Parisi, P. (2021, February 17). How to Fix "No Module Named..." Error in Python [Video]. YouTube. https://www.youtube.com/watch?v=bZpHIxgWYj8.**

3. **Canva. (n.d.). About Us. Retrieved March 24, 2023, from https://www.canva.com/about/.**

4. **Refsnes Data AS. (n.d.). Python Conditions. W3Schools. Retrieved March 24, 2023, from https://www.w3schools.com/python/python_conditions.asp.**

Appendices

The appendices this 2D mini car game would likely include several different sections:

Technical details: The language used to code is Python, Python is considered an easy language to learn because of its simple and straightforward syntax, its high-level nature, its large and active community, its versatility and its open-source status. While coding using python I have used the Pygame library as it is a set of Python modules designed specifically for creating video games. It includes functionality for displaying graphics, playing sounds, and handling input events, and it is built on top of the SDL library.

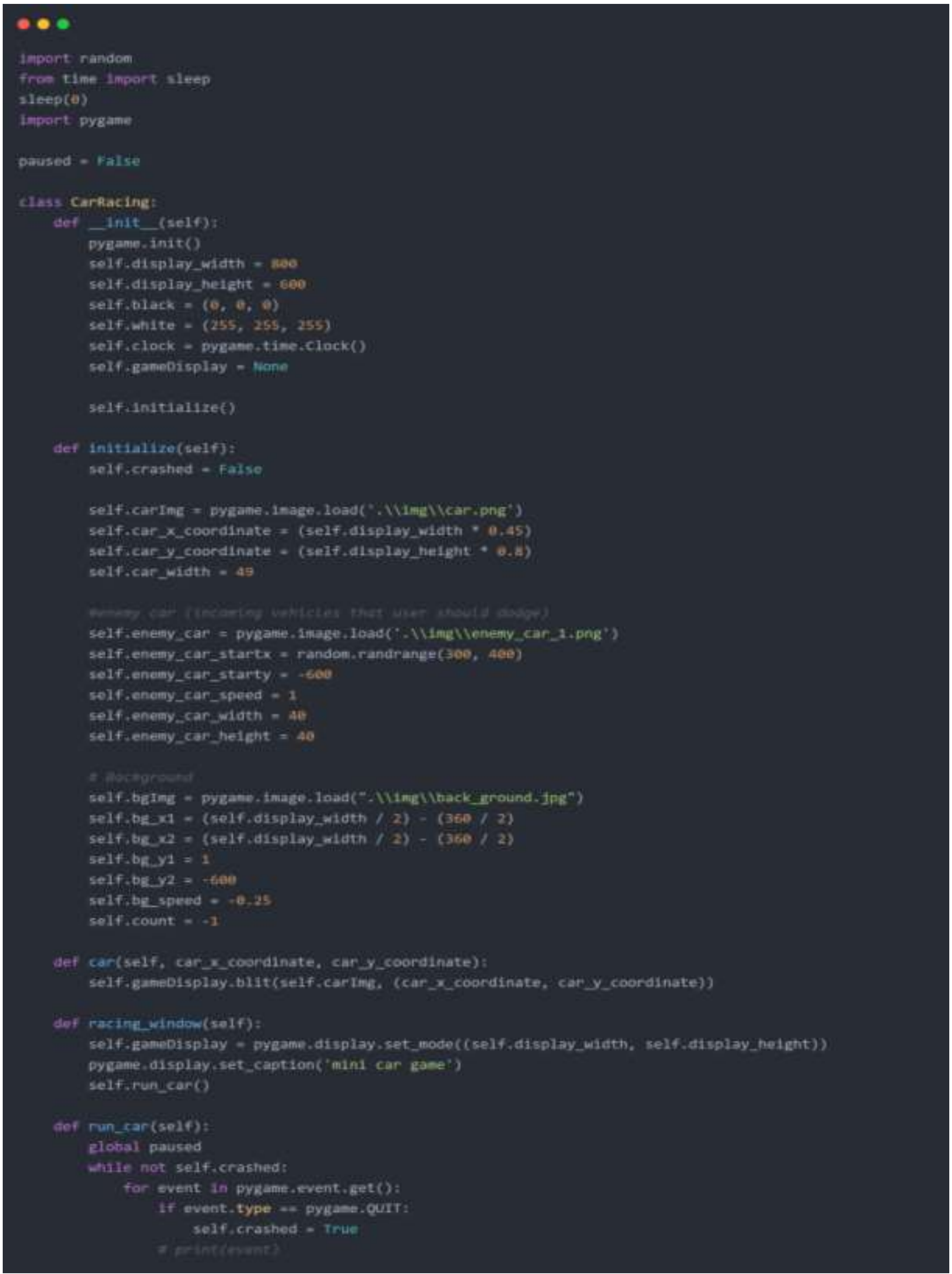
Art assets: Using Canva and its stock images I was able to create PNG images and combine all to an attractive background.

Level design: There is no specific level designed in this game but the speed increases as the car keeps moving, also the randomly generated enemy cars keep coming from the opposite direction.

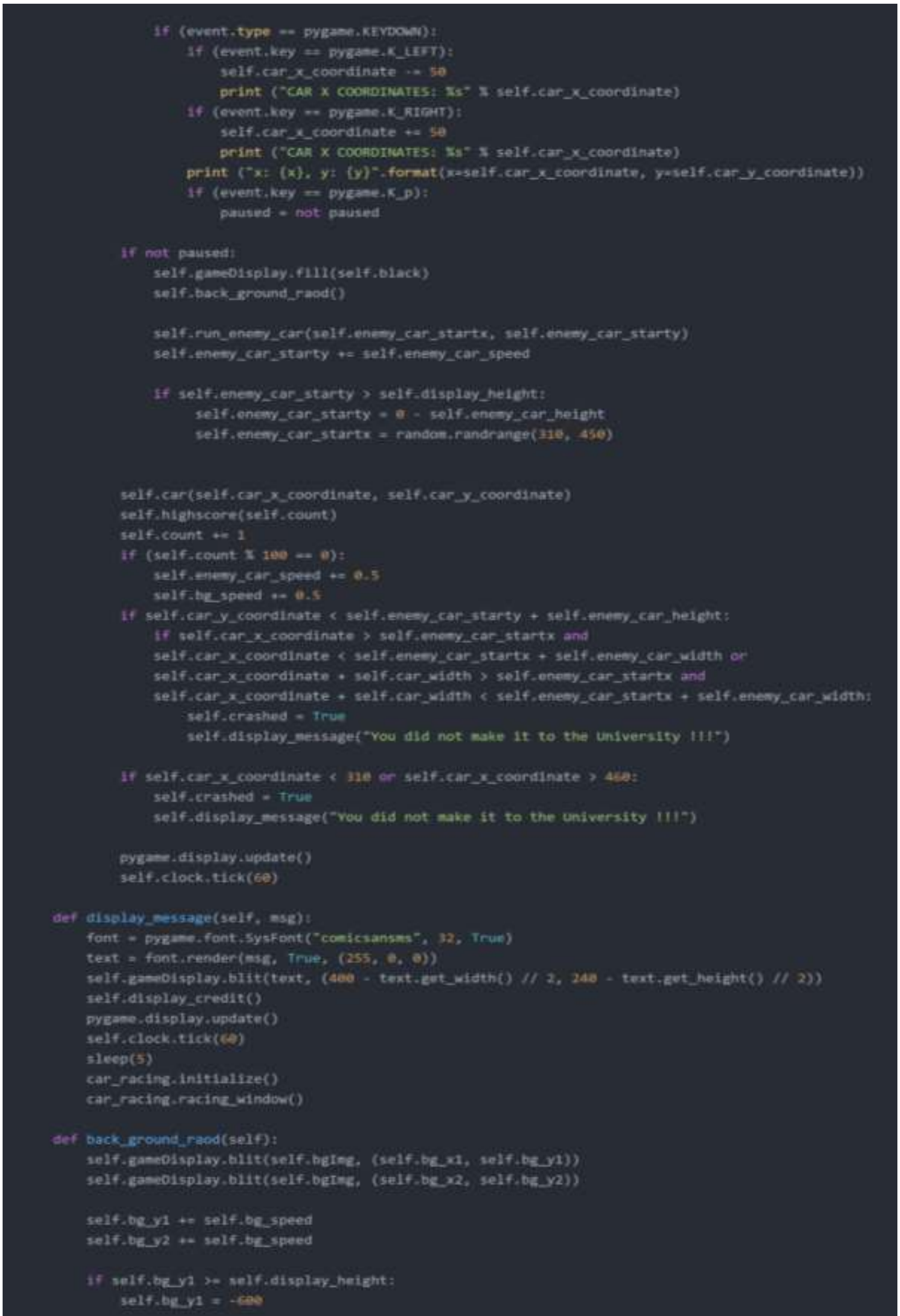
Game mechanics: The player controls the vehicle using the left and right arrow keys.

Game testing: This game was tested multiple times by the users and the developer myself to ensure that the game was balanced and fun to play.

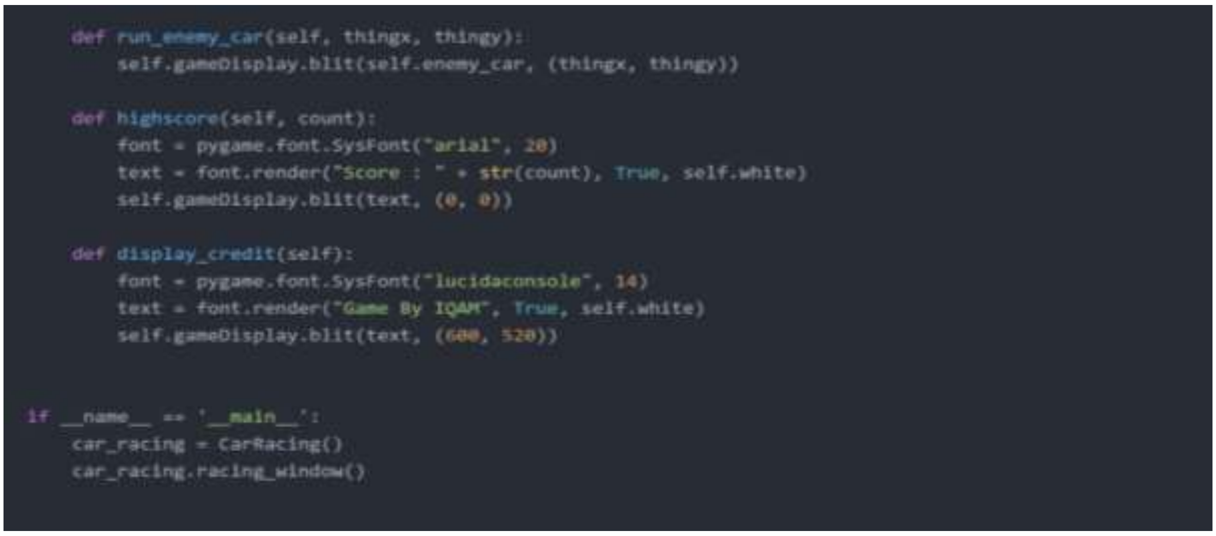
Below you will find the code of the game



```
import random
from time import sleep
sleep(0)
import pygame

paused = False

class CarRacing:
    def __init__(self):
        pygame.init()
        self.display_width = 800
        self.display_height = 600
        self.black = (0, 0, 0)
        self.white = (255, 255, 255)
        self.clock = pygame.time.Clock()
        self.gameDisplay = None

        self.initialize()

    def initialize(self):
        self.crashed = False

        self.carImg = pygame.image.load('.\\img\\car.png')
        self.car_x_coordinate = (self.display_width * 0.45)
        self.car_y_coordinate = (self.display_height * 0.8)
        self.car_width = 49

        #enemy car (incoming vehicles that user should dodge)
        self.enemy_car = pygame.image.load('.\\img\\enemy_car_1.png')
        self.enemy_car_startx = random.randrange(300, 400)
        self.enemy_car_starty = -600
        self.enemy_car_speed = 1
        self.enemy_car_width = 40
        self.enemy_car_height = 40

        # Background
        self.bgImg = pygame.image.load(".\\img\\back_ground.jpg")
        self.bg_x1 = (self.display_width / 2) - (360 / 2)
        self.bg_x2 = (self.display_width / 2) - (360 / 2)
        self.bg_y1 = 1
        self.bg_y2 = -600
        self.bg_speed = -0.25
        self.count = -1

    def car(self, car_x_coordinate, car_y_coordinate):
        self.gameDisplay.blit(self.carImg, (car_x_coordinate, car_y_coordinate))

    def racing_window(self):
        self.gameDisplay = pygame.display.set_mode((self.display_width, self.display_height))
        pygame.display.set_caption('mini car game')
        self.run_car()

    def run_car(self):
        global paused
        while not self.crashed:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.crashed = True
                # print(event)
```
]



```python
                if (event.type == pygame.KEYDOWN):
                    if (event.key == pygame.K_LEFT):
                        self.car_x_coordinate -= 50
                        print ("CAR X COORDINATES: %s" % self.car_x_coordinate)
                    if (event.key == pygame.K_RIGHT):
                        self.car_x_coordinate += 50
                        print ("CAR X COORDINATES: %s" % self.car_x_coordinate)
                    print ("x: {x}, y: {y}".format(x=self.car_x_coordinate, y=self.car_y_coordinate))
                    if (event.key == pygame.K_p):
                        paused = not paused

            if not paused:
                self.gameDisplay.fill(self.black)
                self.back_ground_raod()

                self.run_enemy_car(self.enemy_car_startx, self.enemy_car_starty)
                self.enemy_car_starty += self.enemy_car_speed

                if self.enemy_car_starty > self.display_height:
                     self.enemy_car_starty = 0 - self.enemy_car_height
                     self.enemy_car_startx = random.randrange(310, 450)


            self.car(self.car_x_coordinate, self.car_y_coordinate)
            self.highscore(self.count)
            self.count += 1
            if (self.count % 100 == 0):
                self.enemy_car_speed += 0.5
                self.bg_speed += 0.5
            if self.car_y_coordinate < self.enemy_car_starty + self.enemy_car_height:
                if self.car_x_coordinate > self.enemy_car_startx and
                self.car_x_coordinate < self.enemy_car_startx + self.enemy_car_width or
                self.car_x_coordinate + self.car_width > self.enemy_car_startx and
                self.car_x_coordinate + self.car_width < self.enemy_car_startx + self.enemy_car_width:
                    self.crashed = True
                    self.display_message("You did not make it to the University !!!")

            if self.car_x_coordinate < 310 or self.car_x_coordinate > 460:
                self.crashed = True
                self.display_message("You did not make it to the University !!!")

            pygame.display.update()
            self.clock.tick(60)

    def display_message(self, msg):
        font = pygame.font.SysFont("comicsansms", 32, True)
        text = font.render(msg, True, (255, 0, 0))
        self.gameDisplay.blit(text, (400 - text.get_width() // 2, 240 - text.get_height() // 2))
        self.display_credit()
        pygame.display.update()
        self.clock.tick(60)
        sleep(5)
        car_racing.initialize()
        car_racing.racing_window()

    def back_ground_raod(self):
        self.gameDisplay.blit(self.bgImg, (self.bg_x1, self.bg_y1))
        self.gameDisplay.blit(self.bgImg, (self.bg_x2, self.bg_y2))

        self.bg_y1 += self.bg_speed
        self.bg_y2 += self.bg_speed

        if self.bg_y1 >= self.display_height:
            self.bg_y1 = -600
```



```
    def run_enemy_car(self, thingx, thingy):
        self.gameDisplay.blit(self.enemy_car, (thingx, thingy))

    def highscore(self, count):
        font = pygame.font.SysFont("arial", 20)
        text = font.render("Score : " + str(count), True, self.white)
        self.gameDisplay.blit(text, (0, 0))

    def display_credit(self):
        font = pygame.font.SysFont("lucidaconsole", 14)
        text = font.render("Game By IQAM", True, self.white)
        self.gameDisplay.blit(text, (600, 520))


if __name__ == '__main__':
    car_racing = CarRacing()
    car_racing.racing_window()
```

If you have had a look into the code it is really simple to understand what’s going on in the program, the user is free to edit the colour of the text that appears, change the background, edit a new car image and rename it into the code, the user is also able to control the speed of the cars and the background in the game All these options will help to make the game more engaging and interactive, providing players with a great gaming as well as coding experience.

END