Car Simulator Game – Traffic Rules using Unity 3D.

INFO 5900 - Special Problems

Dr. Sharad Sharma

Project Group 1

Krishna Rahul Mathamsetti

Vineeth Akula

Uday Kiran Reddy kotha

Contents

Abstract: 3

Introduction 3

  Designed Environment: 4

Implementation 5

Acknowledgement 5

References 6

Goal and Objectives 6

  Goals: 6

  Modeling: 6

  Application: 6

  Modeling in 3ds Max: 6

Programming 7

Functionality 14

  Functionality: 14

  Vision: 14

  Sound: 14

  Animation: 14

  Interactivity: 15

  Characters/Avatars: 15

  Sensors: 15

  Player: 15

  AI Implementation: 15

  Interface elements: 15

Explanation on behaviors implemented. 16

Importance of VR for this project 17

Future Work 17

Future Modes of Game: 18

Abstract:

The need for a VR-based game that teaches and promotes the following of traffic rules is significant for several reasons. Firstly, road accidents are a major cause of injuries and fatalities worldwide, and many of these accidents are caused by human error or negligence. By providing an immersive VR environment that simulates real-life traffic scenarios, drivers can practice safe driving behaviors and develop their driving skills in a safe and controlled environment.

Secondly, many new drivers may not have access to real-world driving experiences or may lack confidence in their driving abilities. A VR-based game can help bridge this gap by providing a realistic driving experience that allows users to practice and develop their skills in a simulated environment before they hit the road in the real world.

The VR environment in this game could show a variety of different scenarios, such as city driving or rural roads, and could include various weather conditions, traffic density, and road hazards. The game could also provide feedback to users on their performance, such as highlighting areas where they need to improve, and could track progress over time.

The target audience for this VR application would be new drivers, as well as experienced drivers who want to refresh their knowledge of traffic rules and develop their driving skills in a safe environment. Additionally, driving schools and educators could also use this game as a teaching tool to supplement traditional driver education courses.

The intention of this game is to promote safe driving behaviors and reduce the incidence of road accidents caused by human error. By providing a realistic and engaging environment for users to practice safe driving behaviors, the game could help create more responsible and skilled drivers.

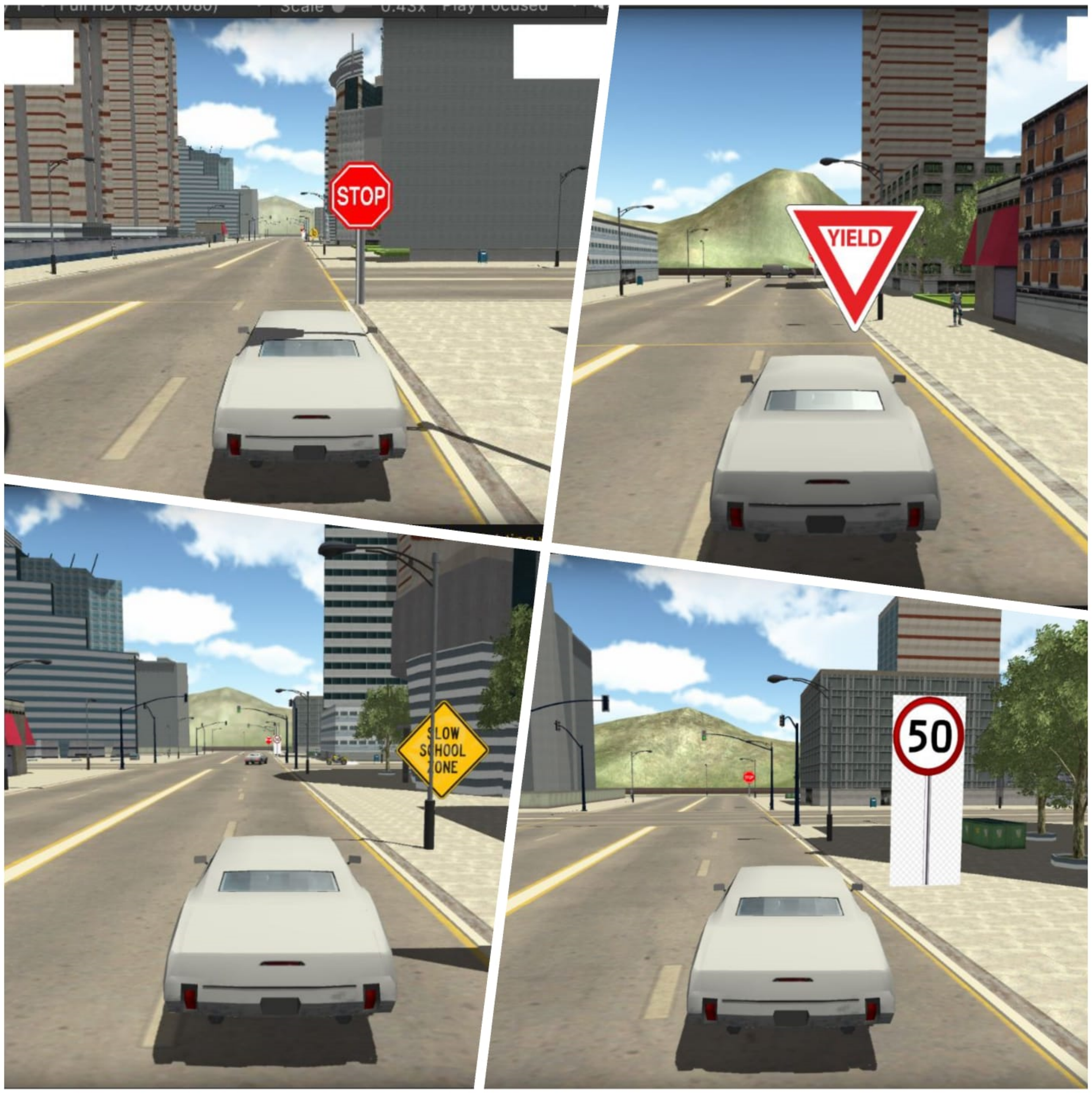
Overall, this VR-based game would be useful for promoting safe driving behaviors, reducing the incidence of road accidents, and improving driving skills for new and experienced drivers. It could serve as a valuable tool for driving schools, educators, and anyone looking to improve their driving skills and knowledge of traffic rules.

Introduction

The goal of the project is to design and develop a VR-based game that promotes safe driving behaviors and improves driving skills for new and experienced drivers. The objectives of the project include:

1. Creating a realistic and immersive VR environment that simulates real-life traffic scenarios.
2. Developing a game that is engaging and interactive, and that provides users with feedback on their performance.
3. Teaching and reinforcing safe driving behaviors, such as following traffic rules.
4. Improving driving skills, such as steering, braking, and accelerating, in a safe and controlled environment.

5. Providing a fun and enjoyable experience for users that encourages them to continue practicing and improving their driving skills as shown below.



Designed Environment:

The designed environment for the VR-based game would be a realistic simulation of real-world traffic scenarios. The environment would include different types of roads, such as city streets and rural roads, as well as different traffic densities.

The environment would be designed to be immersive and engaging, with high-quality graphics and realistic sound effects. Users would be able to interact with the environment using VR controllers, allowing them to steer the vehicle and adjust their speed.

Different Aspects of the Environment:

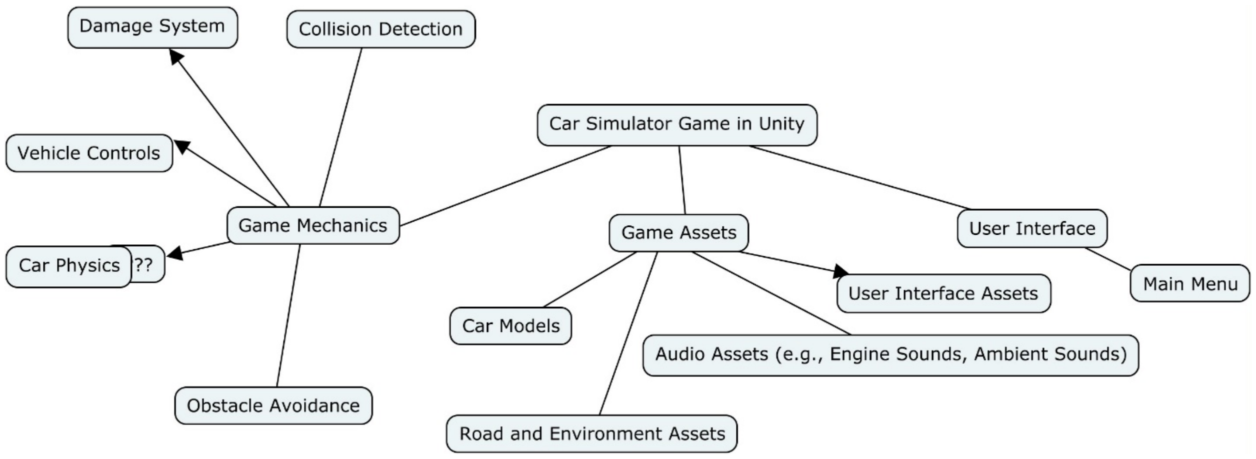
The different aspects of the VR environment would include:

1. Roads: The environment would include different types of roads, such as city streets and rural roads, each with their own unique characteristics and challenges.
2. Traffic: The environment would simulate traffic.
3. Traffic rules: The environment would include traffic different rules.

Overall, the designed environment would be a realistic and engaging simulation of real-life traffic scenarios, aimed at teaching and reinforcing safe driving behaviors and improving driving skills for new and experienced drivers.

Implementation

Car Simulator Game in Unity



The game mechanics, game assets, user interface, and game modes are some of the key elements of an automobile simulator game that are covered in this concept map. It's vital to keep in mind that this is only a basic blueprint and that, depending on the particulars of your game, there may be many extra elements and ideas involved.

Acknowledgement

Drive Sim - A driving simulator with traffic rules by Mustafa Karagoz. GitHub, 2021, https://github.com/mkaragoz/drive_sim.

References

https://github.com/mkaragoz/drive_sim

https://play.google.com/store/apps/details?id=com.tgf.real.car.simulator.game.traffic.rules

https://store.steampowered.com/app/493490/City_Car_Driving/

https://assetstore.unity.com/packages/tools/ai/traffic-rules-ai-for-unity3d-53356

Goal and Objectives

Goals:

- Educate players on traffic rules and safe driving behaviors
- Help players improve their driving skills through simulation and practice
- Encourage players to adopt safer driving habits and reduce the risk of accidents on the road

Modeling:

- The virtual environment should be a realistic representation of a typical urban or suburban setting, with roads, intersections, traffic signals, signs, and other objects commonly found in a real-world driving environment.
- The geometry of the environment should be accurate and proportional, with appropriate textures and lighting to create a realistic atmosphere.
- Animations such as moving traffic, pedestrians can be added to create a more immersive experience.
- The functionality of the game should include realistic traffic rules and scenarios that players must navigate, such as following speed limits, obeying traffic signals, and reacting to other vehicles on the road.

Application:

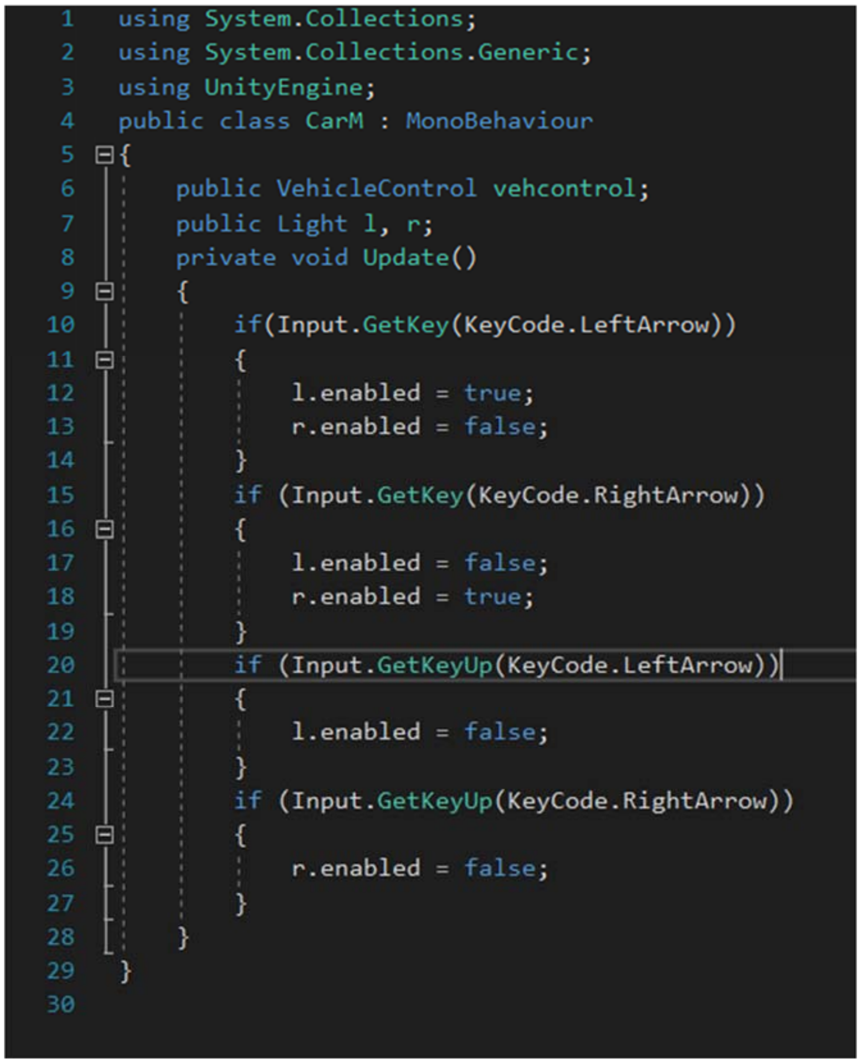
- The target audience for this game could be novice or inexperienced drivers, as well as anyone looking to improve their driving skills and knowledge of traffic rules.
- Players can interact with the game using a variety of inputs with a keyboard.
- Navigation can be done using a map, and interactions can include braking, and accelerating, as well as reacting to other vehicles or hazards on the road.

Modeling in 3ds Max:

- 3ds Max can be used to create a 3D model of the virtual environment, including the roads, intersections, and other objects.
- Accurate measurements and proportions should be used to ensure that the virtual environment is a realistic representation of a real-world driving scenario.

- Textures and lighting can be added to create a more immersive experience, and animations can be included to simulate moving traffic, pedestrians, and other objects.
- The 3D model can be exported to a game engine such as Unity or Unreal Engine to create an interactive game.

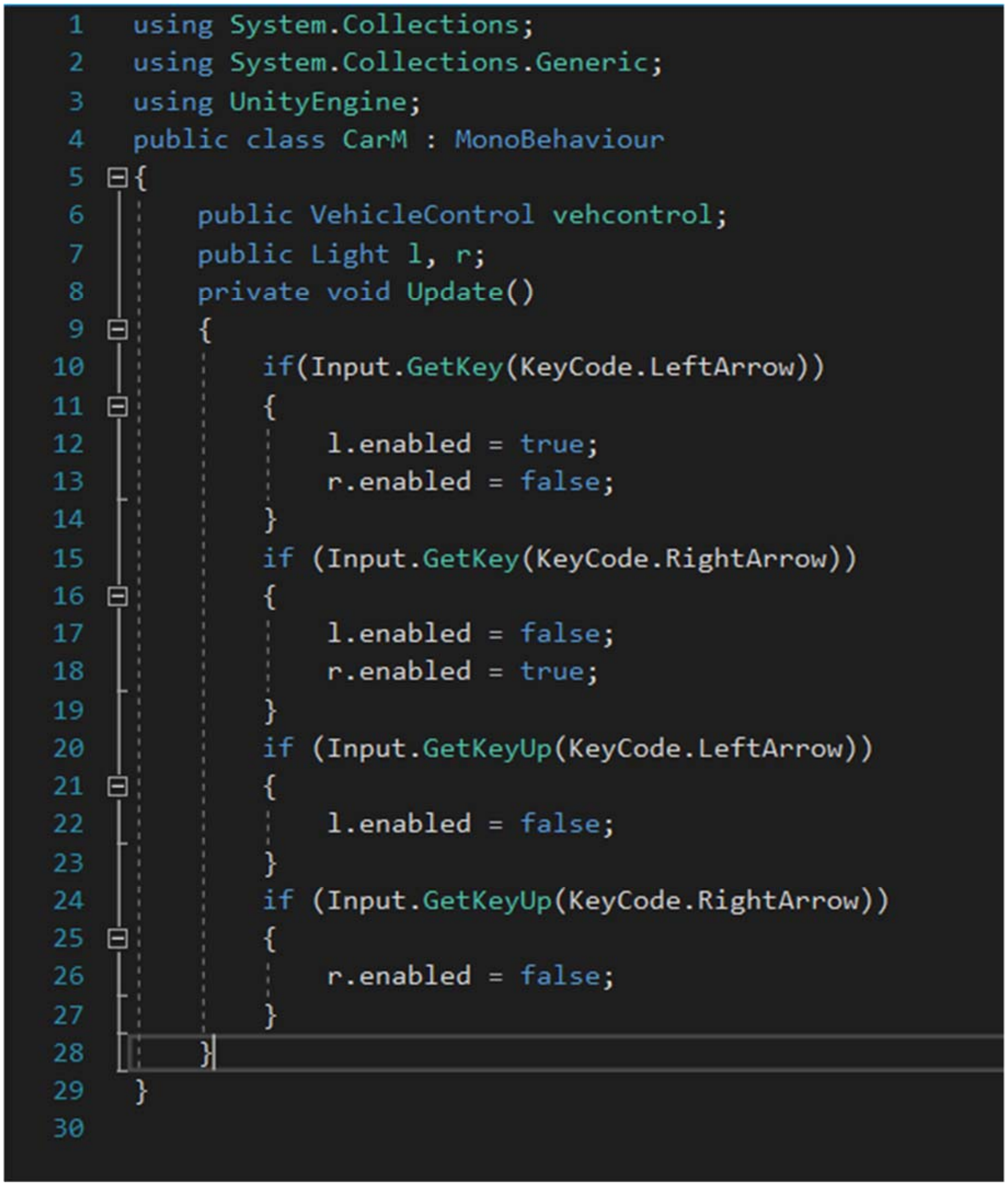
Programming



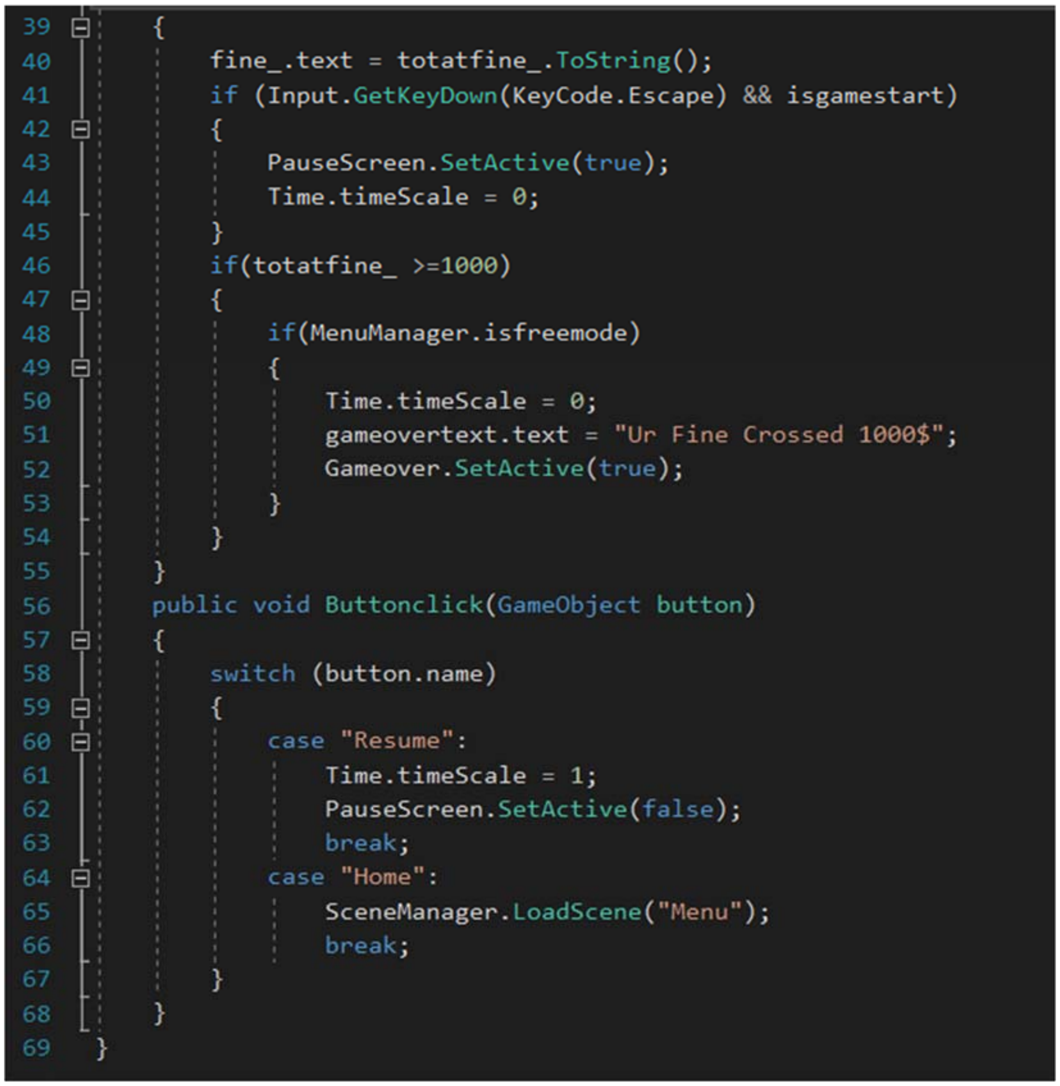
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CarM : MonoBehaviour
{
    public VehicleControl vehcontrol;
    public Light l, r;
    private void Update()
    {
        if(Input.GetKey(KeyCode.LeftArrow))
        {
            l.enabled = true;
            r.enabled = false;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            l.enabled = false;
            r.enabled = true;
        }
        if (Input.GetKeyUp(KeyCode.LeftArrow))
        {
            l.enabled = false;
        }
        if (Input.GetKeyUp(KeyCode.RightArrow))
        {
            r.enabled = false;
        }
    }
}

```

This script is for a Unity game's automobile object. The keyboard's Left and Right arrow keys are used to control the car's turn signal lights. The left turning signal light turns on and the right turning signal light goes out when the Left arrow key is depressed. The right turning signal light comes on when the right arrow key is pressed, and the left turning signal light goes out. The appropriate turning signal light goes out when either key is released. For the turning signal lights, the script needs two Light objects and a reference to a VehicleControl script.

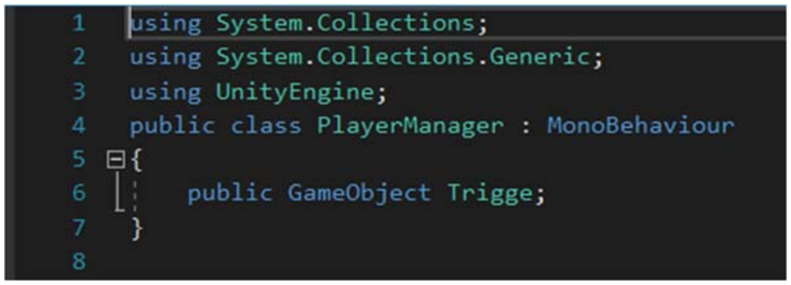


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CarM : MonoBehaviour
{
    public VehicleControl vehcontrol;
    public Light l, r;
    private void Update()
    {
        if(Input.GetKey(KeyCode.LeftArrow))
        {
            l.enabled = true;
            r.enabled = false;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            l.enabled = false;
            r.enabled = true;
        }
        if (Input.GetKeyUp(KeyCode.LeftArrow))
        {
            l.enabled = false;
        }
        if (Input.GetKeyUp(KeyCode.RightArrow))
        {
            r.enabled = false;
        }
    }
}
```



```
    {
        fine_.text = totatfine_.ToString();
        if (Input.GetKeyDown(KeyCode.Escape) && isgamestart)
        {
            PauseScreen.SetActive(true);
            Time.timeScale = 0;
        }
        if(totatfine_ >=1000)
        {
            if(MenuManager.isfreemode)
            {
                Time.timeScale = 0;
                gameovertext.text = "Ur Fine Crossed 1000$";
                Gameover.SetActive(true);
            }
        }
    }
    public void Buttonclick(GameObject button)
    {
        switch (button.name)
        {
            case "Resume":
                Time.timeScale = 1;
                PauseScreen.SetActive(false);
                break;
            case "Home":
                SceneManager.LoadScene("Menu");
                break;
        }
    }
}
```

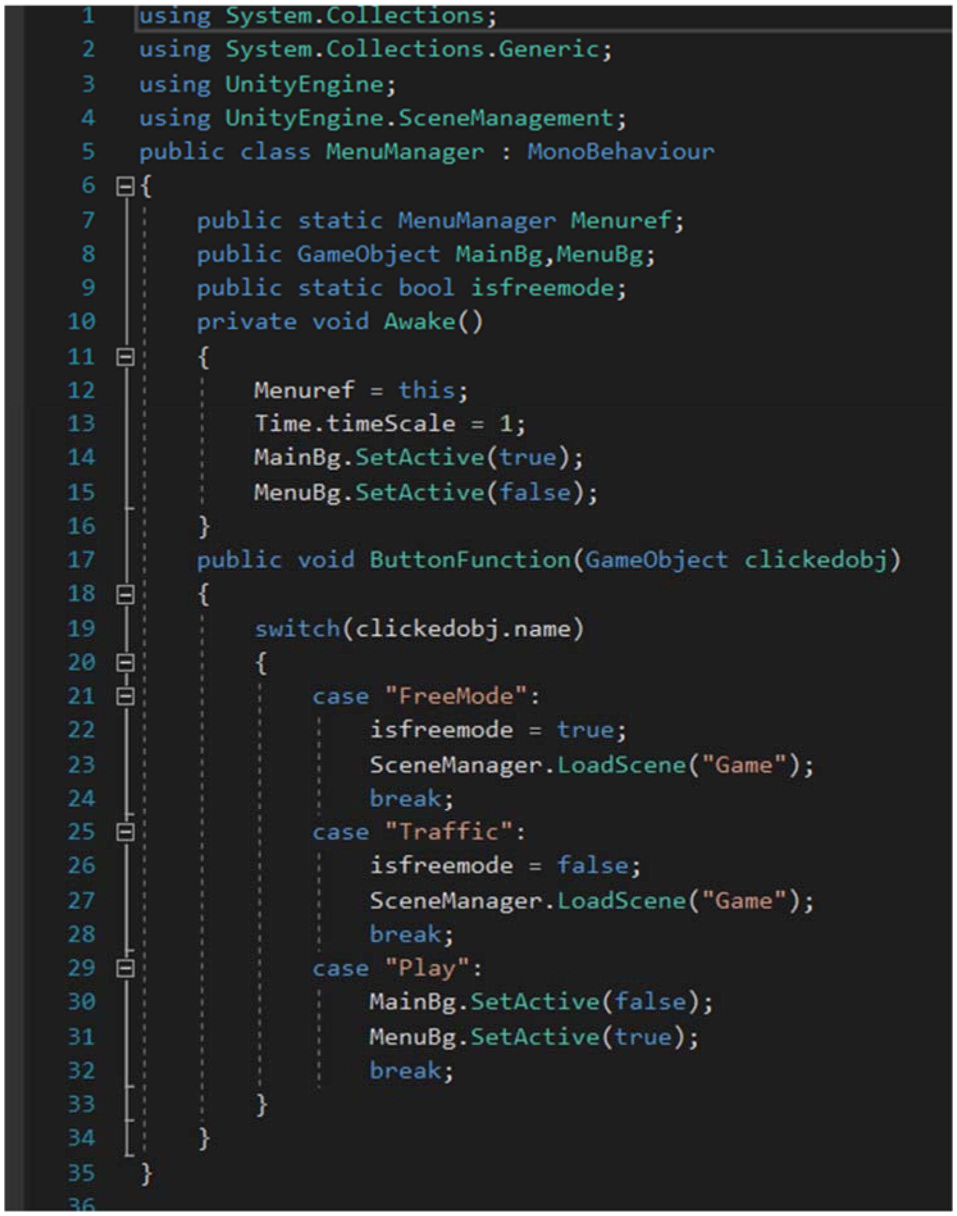
This script is for a Unity game's GameManager object. It controls the game's pace and keeps track of the player's advancement. The script includes variables and instructions for processing user input, controlling game objects, managing fines, and displaying text on screen. A timer, success and game-over screens, as well as a stop menu, are also included in the script. Depending on whether the game is in free mode or traffic rules mode, the script decides which game objects to activate. The game will finish and show a game over screen if the player has accrued more fines than a predetermined threshold. The script has procedures for dealing with button clicks, including restarting the game or going back to the main menu.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerManager : MonoBehaviour
{
    public GameObject Trigge;
}

```

This script is for a Unity game's PlayerManager object. There is only one public variable in it, and that refers to the GameObject "Trigge." Without more information, the purpose of this script is unclear, however it is likely that the "Trigge" object is used to start an event or behavior in the game.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MenuManager : MonoBehaviour
{
    public static MenuManager Menuref;
    public GameObject MainBg,MenuBg;
    public static bool isfreemode;
    private void Awake()
    {
        Menuref = this;
        Time.timeScale = 1;
        MainBg.SetActive(true);
        MenuBg.SetActive(false);
    }
    public void ButtonFunction(GameObject clickedobj)
    {
        switch(clickedobj.name)
        {
            case "FreeMode":
                isfreemode = true;
                SceneManager.LoadScene("Game");
                break;
            case "Traffic":
                isfreemode = false;
                SceneManager.LoadScene("Game");
                break;
            case "Play":
                MainBg.SetActive(false);
                MenuBg.SetActive(true);
                break;
        }
    }
}
```

This script is for a Unity game's MenuManager object. It has a number of static boolean variables as well as the public variable "isfreemode." The script's main functions are to control the game menu and take care of button clicks.

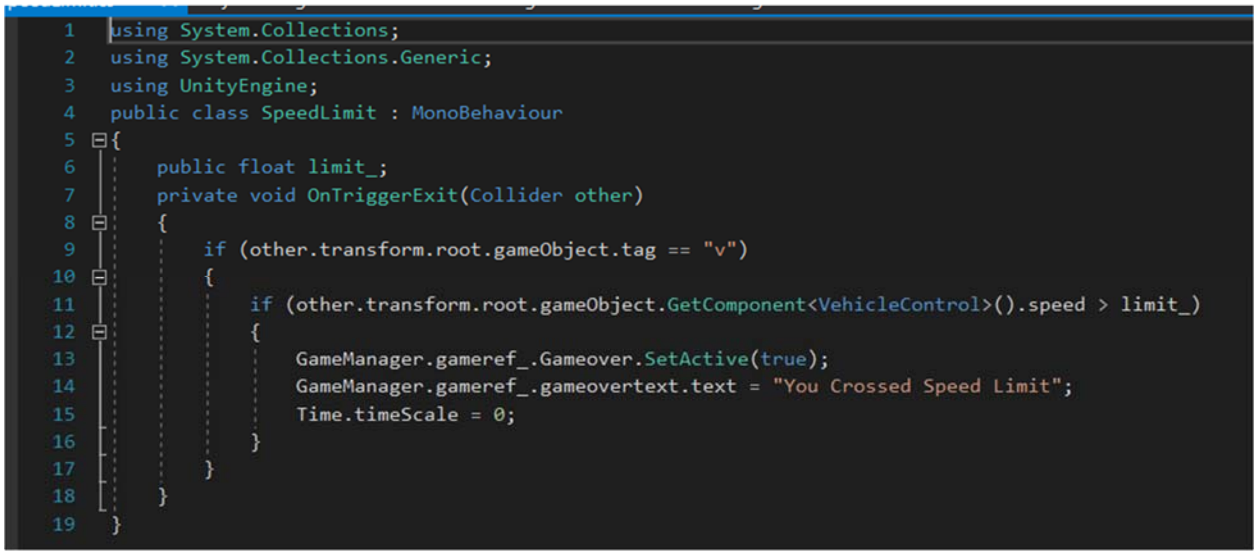
The Time.timeScale is set to 1 in the Awake() method, the MainBg object is turned on, and the MenuBg object is turned off.

Three different types of button clicks are handled in the ButtonFunction() method:

1. The "Game" scene is loaded when the "FreeMode" button is pressed, setting isfreemode to true.

2. The traffic button, which loads the "Game" scene and sets isfreemode to false.

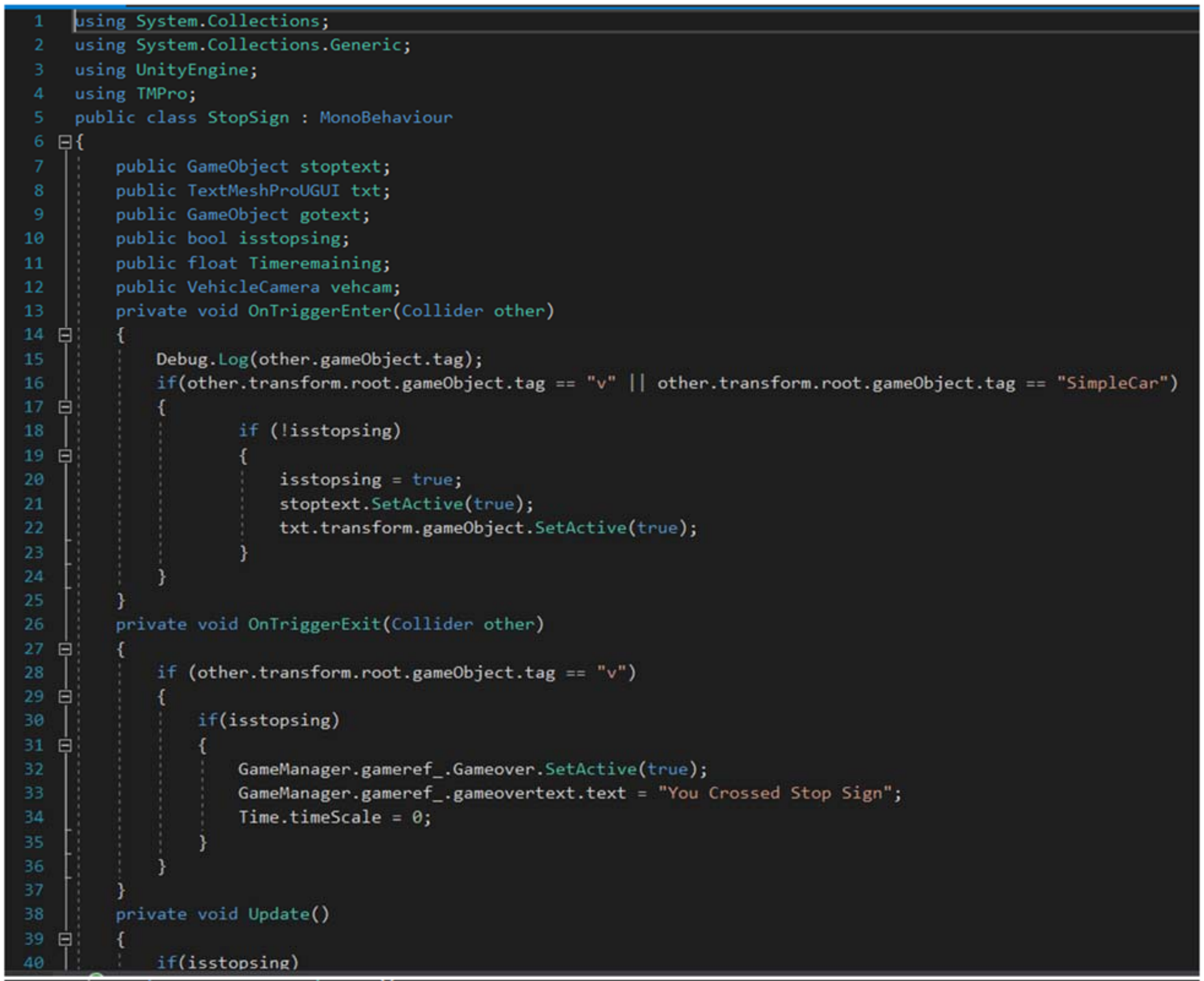
3. The Play button reveals the MenuBg object while hiding the MainBg object.



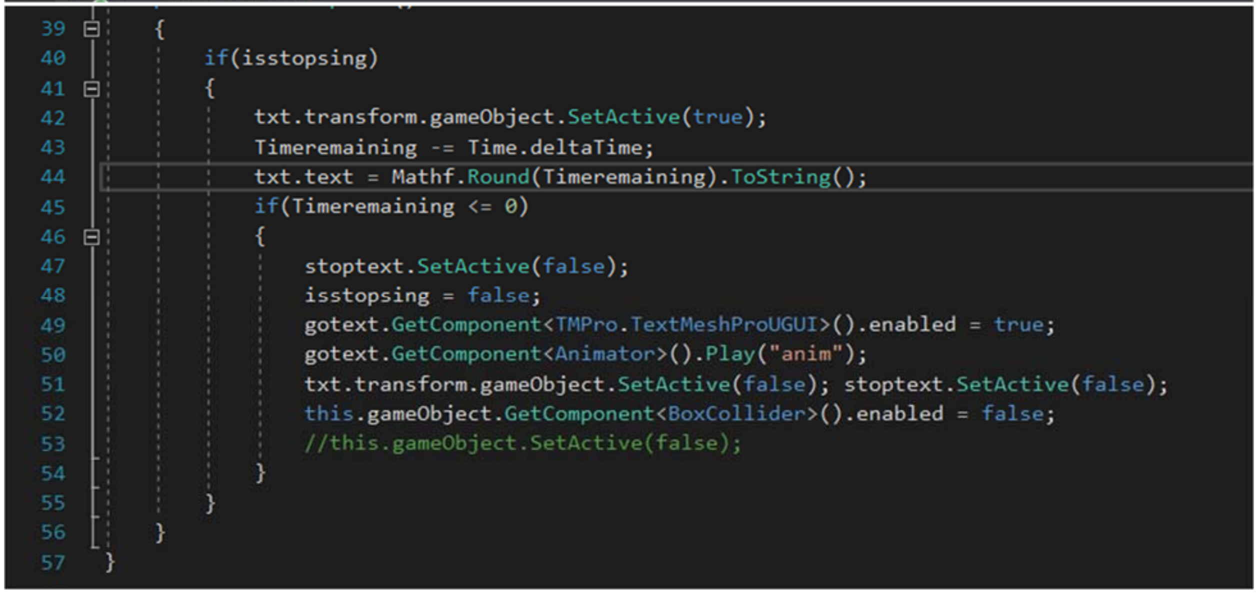
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SpeedLimit : MonoBehaviour
{
    public float limit_;
    private void OnTriggerExit(Collider other)
    {
        if (other.transform.root.gameObject.tag == "v")
        {
            if (other.transform.root.gameObject.GetComponent<VehicleControl>().speed > limit_)
            {
                GameManager.gameref_.Gameover.SetActive(true);
                GameManager.gameref_.gameovertext.text = "You Crossed Speed Limit";
                Time.timeScale = 0;
            }
        }
    }
}
```

The "SpeedLimit" script is used to determine whether the car exceeds the posted speed limit in a specific location. Its public variable "limit_" specifies the region's top speed limit. Any vehicle with the "v" tag that exits the collider attached to the script triggers a check to see if the speed is higher than the predetermined limit. The game ends and a notification is shown in the Game Over UI if the vehicle's speed is greater.

The trigger collider is positioned in the area where the speed limit has to be enforced, and the script is attached to it.



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
public class StopSign : MonoBehaviour
{
    public GameObject stoptext;
    public TextMeshProUGUI txt;
    public GameObject gotext;
    public bool isstopsing;
    public float Timeremaining;
    public VehicleCamera vehcam;
    private void OnTriggerEnter(Collider other)
    {
        Debug.Log(other.gameObject.tag);
        if(other.transform.root.gameObject.tag == "v" || other.transform.root.gameObject.tag == "SimpleCar")
        {
                if (!isstopsing)
                {
                    isstopsing = true;
                    stoptext.SetActive(true);
                    txt.transform.gameObject.SetActive(true);
                }
        }
    }
    private void OnTriggerExit(Collider other)
    {
        if (other.transform.root.gameObject.tag == "v")
        {
            if(isstopsing)
            {
                GameManager.gameref_.Gameover.SetActive(true);
                GameManager.gameref_.gameovertext.text = "You Crossed Stop Sign";
                Time.timeScale = 0;
            }
        }
    }
    private void Update()
    {
        if(isstopsing)
```



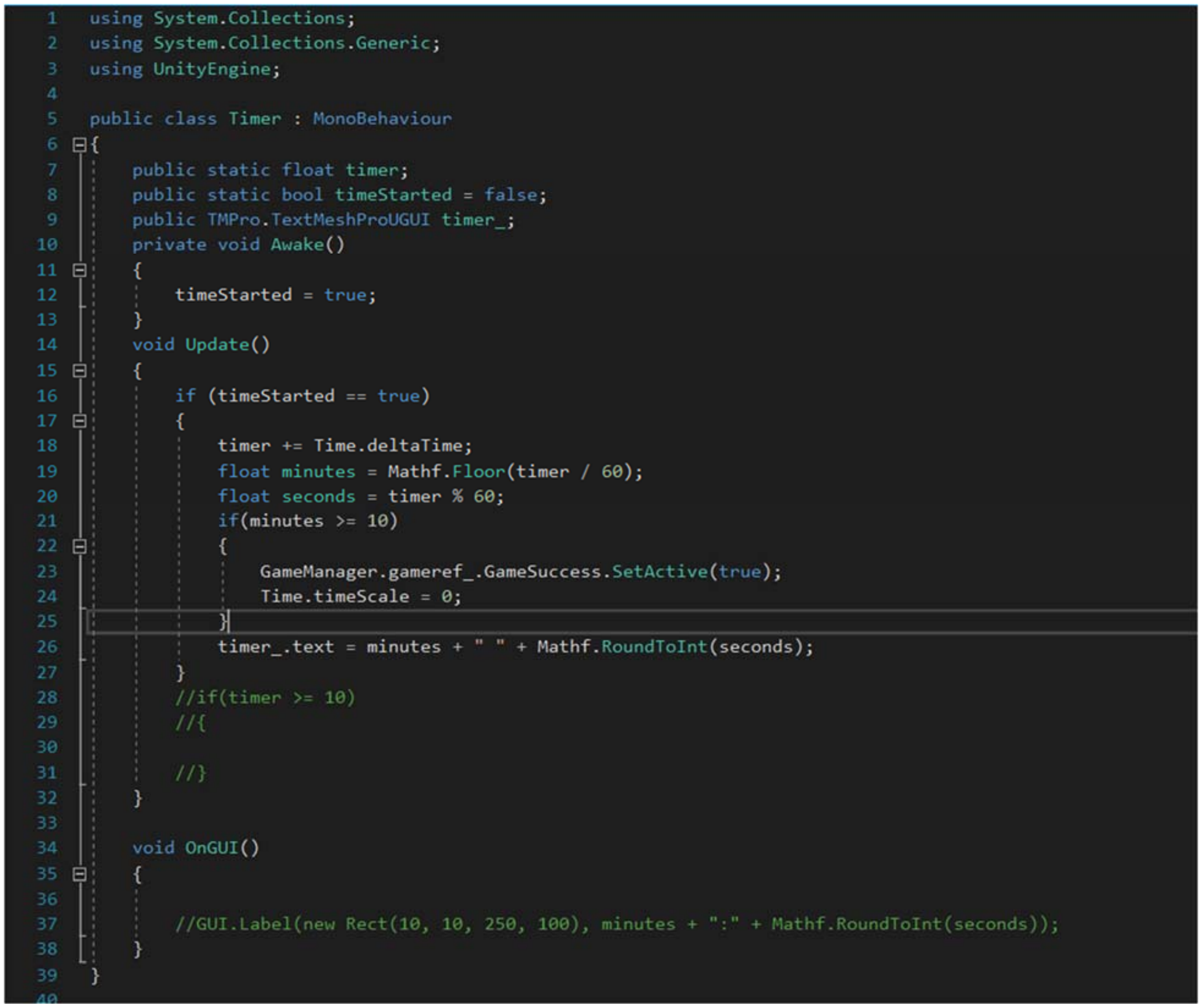
```csharp
    {
        if(isstopsing)
        {
            txt.transform.gameObject.SetActive(true);
            Timeremaining -= Time.deltaTime;
            txt.text = Mathf.Round(Timeremaining).ToString();
            if(Timeremaining <= 0)
            {
                stoptext.SetActive(false);
                isstopsing = false;
                gotext.GetComponent<TMPro.TextMeshProUGUI>().enabled = true;
                gotext.GetComponent<Animator>().Play("anim");
                txt.transform.gameObject.SetActive(false); stoptext.SetActive(false);
                this.gameObject.GetComponent<BoxCollider>().enabled = false;
                //this.gameObject.SetActive(false);
            }
        }
    }
}
```

This script is for a driving game's stop sign item. A "stop" text and a countdown timer appear when a car enters the stop sign's trigger region. When the countdown timer reaches zero, the vehicle must come to a complete stop and remain there until a "go" text displays, allowing it to move forward. The game over screen appears if the car passes the stop sign before the clock runs out.

A "stop" text game object, a text mesh pro component, a "go" text game object, a boolean flag to determine whether the vehicle has stopped, a timer to clock down, and a reference to the vehicle camera are all public references in the script.

When a vehicle enters the trigger area, OnTriggerEnter displays the "stop" text and countdown timer if the vehicle hasn't stopped yet. The game over screen is shown in OnTriggerExit if a vehicle leaves the trigger region while the stop sign flag is still true.

The countdown timer is shortened and the timer text is updated in Update if the stop sign flag is set to true. The "go" lettering is shown and the stop sign collider is turned off once the timer reaches zero, allowing the vehicle to go.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Timer : MonoBehaviour
{
    public static float timer;
    public static bool timeStarted = false;
    public TMPro.TextMeshProUGUI timer_;
    private void Awake()
    {
        timeStarted = true;
    }
    void Update()
    {
        if (timeStarted == true)
        {
            timer += Time.deltaTime;
            float minutes = Mathf.Floor(timer / 60);
            float seconds = timer % 60;
            if(minutes >= 10)
            {
                GameManager.gameref_.GameSuccess.SetActive(true);
                Time.timeScale = 0;
            }
            timer_.text = minutes + " " + Mathf.RoundToInt(seconds);
        }
        //if(timer >= 10)
        //{

        //}
    }

    void OnGUI()
    {

        //GUI.Label(new Rect(10, 10, 250, 100), minutes + ":" + Mathf.RoundToInt(seconds));
    }
}
```

The game timer is kept track of by this script, and it is displayed on the screen. It updates the amount of time that has passed since the game began using the 'Time.deltaTime' property, using a static variable named 'timer' to keep track of the passing seconds. Additionally, it displays the time using a TextMeshProUGUI component and formats it in minutes and seconds.

The status of the timer is determined by the value of the 'timeStarted' variable. This variable is set to true by the 'Awake()' method, indicating that the timer should begin to run as soon as the game launches.

The timer value is updated and formatted into minutes and seconds via the 'Update()' method. The 'GameSuccess' screen is displayed and the time scale is reset to 0 when the elapsed time is greater than or equal to 10 minutes. The game is then paused. Because the 'OnGUI()' method is commented out, it has no impact on how the script works.

Functionality

Functionality:

- The game was designed to simulate real-life traffic scenarios, where the player must follow traffic rules and regulations.
- The player must navigate through the environment using keyboard controls and avoid collisions with other vehicles and pedestrians.
- Points are not awarded for following traffic rules, such as stopping at red lights and using turn signals, and fines are added for breaking rules.

Vision:

- The environment was designed using 3D models of roads, vehicles, buildings, and pedestrians.
- Textures were used to provide realistic details, such as road markings, signs, and buildings.
- The environment was designed to resemble a real city, with intersections, traffic lights, and different types of vehicles.

Sound:

- Ambient sounds were added to the environment, such as vehicle sounds, pedestrian sounds, and background noises.

Animation:

- Animated objects were included in the game: traffic lights, pedestrians, and vehicles.
- The traffic lights change color at specified intervals, indicating when the player can proceed.
- Pedestrians walk along sidewalks and crosswalks and follow traffic signals.
- Vehicles move along the roads, following traffic rules and avoiding collisions.

Interactivity:

- Five user-triggered events were included in the game: starting the game, ending the game, and navigating the vehicle.
- The game can be started by clicking the “PLAY" button on the game menu.
- The vehicle can be navigated using keyboard controls.

Characters/Avatars:

- Animated agents were included in the game, such as pedestrians and vehicles.
- The behavior of the agents was programmed to follow traffic rules, such as stopping at red lights and yielding to pedestrians.

Sensors:

- Three different types of sensors were included in the game: speed sensors, time sensors, and touch sensors.
- Speed sensors were used to detect when the player is accelerating the vehicle.
- Time sensors were used to measure the time elapsed in the game.
- Touch sensors were used to detect when the player interacts with buttons on the game interface.

Player:

- A player controller was added to the scene, allowing the player to navigate the vehicle using keyboard controls.
- The player can accelerate, brake, and turn the vehicle using arrow keys.

AI Implementation:

- AI functionality was implemented for the behavior of agents, such as pedestrians and vehicles.
- The behavior of agents was programmed to follow traffic rules, such as stopping at red lights and yielding to pedestrians.

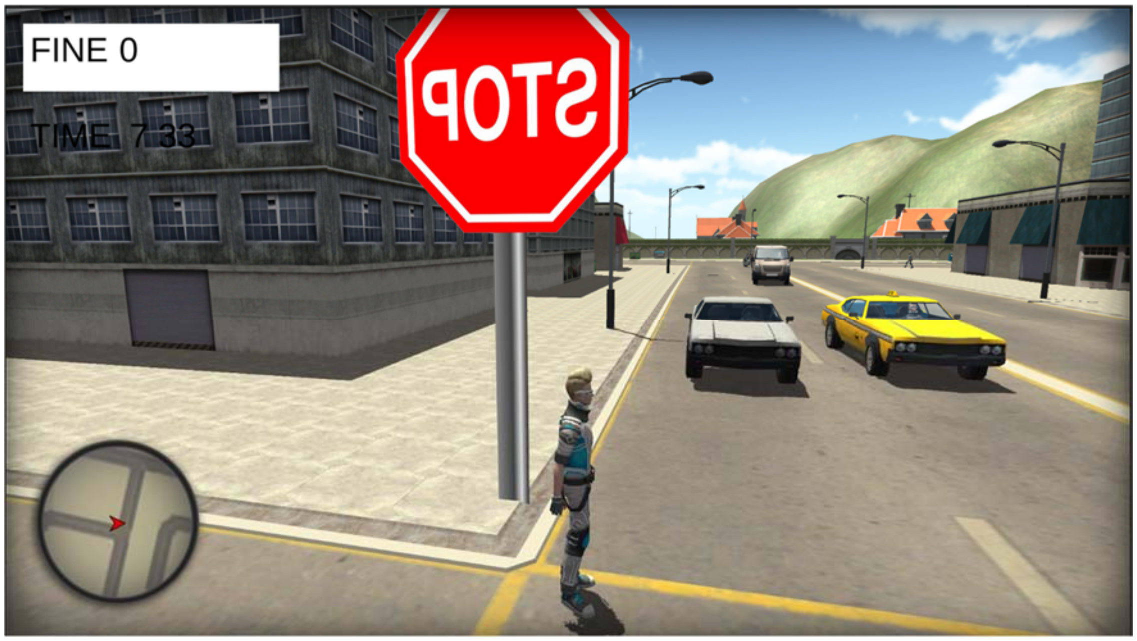
Interface elements:

- The game interface includes menu items such as a "PLAY" button and a timer.
- Points are deducted and are displayed on the game interface.

Explanation on behaviors implemented.

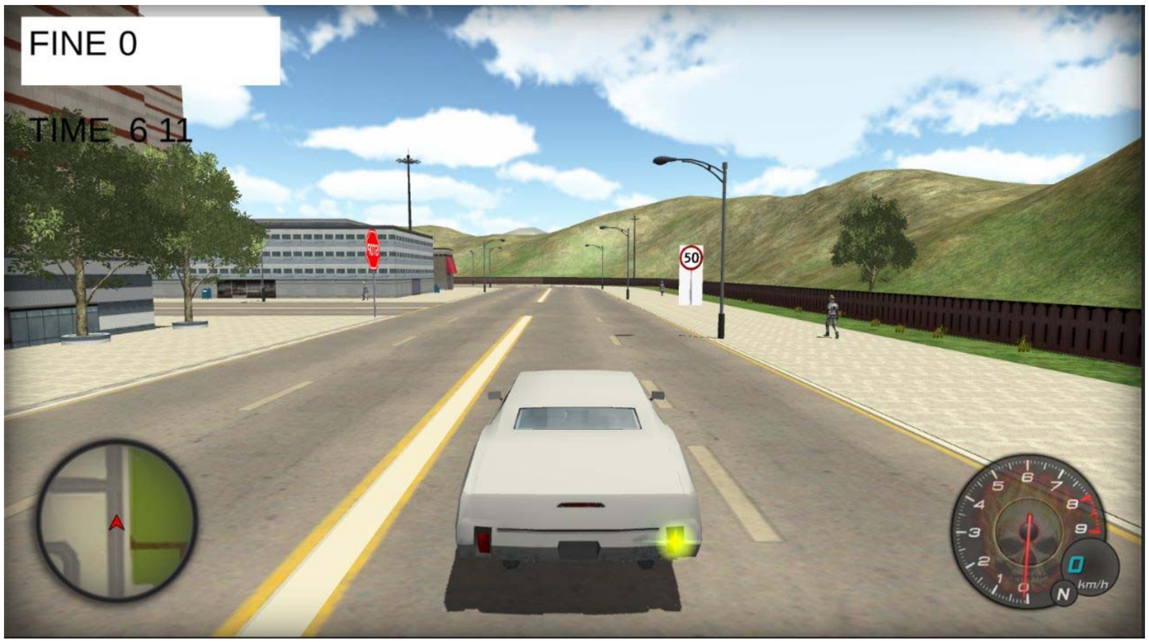
Here are some screenshots of these agents/avatars in action:

1. Pedestrian behavior:



In the screenshot above, the pedestrians are waiting at a traffic signal. They will start crossing the road only when the signal turns green.

2. Vehicle behavior:



In the screenshot above, the vehicles are following lane markings and avoiding collisions with other vehicles.

Importance of VR for this project

This traffic rules game in virtual reality has several benefits over traditional methods of learning about traffic rules. The use of virtual reality allows for a more immersive and engaging learning experience, which can help to increase knowledge retention and motivation to learn. It also provides a safe and controlled environment for learners to practice and apply their knowledge without the risks and dangers associated with real-world driving.

In addition, the use of virtual reality allows for a more dynamic and interactive learning experience, where learners can actively engage with the environment, make decisions and receive immediate feedback on their actions. This can help to develop critical thinking skills and decision-making abilities, which are crucial for safe driving.

Furthermore, the use of avatars and artificial intelligence in the virtual environment provides a more realistic and dynamic simulation of real-world traffic situations, helping learners to better understand and anticipate different scenarios that may arise while driving. The inclusion of sensors and interactivity also allows for a more personalized and adaptive learning experience, catering to the individual needs and learning styles of different learners.

Overall, the use of virtual reality technology provides a powerful and effective tool for learning about traffic rules and safe driving practices, making it a highly useful application for learners of all levels and ages.

Future Work

here are some common problems and shortcomings that could be encountered in a virtual reality traffic rules game and some suggestions for future work:

1. Limited interactivity: Although the game may have several user-triggered events, it may not provide enough interactivity with the environment or other characters. To improve this, future work could focus on adding more interactive elements to the game, such as traffic signals, pedestrian crossings, and obstacles.
2. Limited realism: The virtual environment may not fully replicate the real-world driving experience, which can limit the effectiveness of the training. Future work could focus on improving the realism of the environment, such as adding more realistic textures, sounds, and animations.
3. Technical limitations: The game may have technical limitations that can affect the performance and user experience. Future work could focus on optimizing the game for different hardware configurations, such as lower-end VR headsets or mobile devices.
4. Lack of variety: The game may lack variety in terms of different scenarios and challenges that can arise while driving. Future work could focus on expanding the game to include different driving conditions, such as different weather conditions, different types of roads, and different times of day.

5. Limited AI behaviors: The AI agents in the game may have limited behaviors, which can affect the realism and effectiveness of the training. Future work could focus on expanding the AI behaviors to include more realistic and varied driving scenarios, such as road rage, distracted driving, and emergency situations.

In conclusion, while virtual reality is an appropriate technology for a traffic rules game, there are still many areas that can be improved upon to provide a more effective and engaging training experience. By addressing these problems and shortcomings, the game can become a valuable tool for teaching safe driving practices and promoting awareness of traffic rules.

Future Modes of Game:

A traffic game with different modes, such as state-wise and country-wise modes, can be an engaging and educational tool for users to learn about different traffic rules and regulations in various regions.

In the state-wise mode, the game can focus on the specific traffic laws and driving conditions of different states within a country. Users can select the state they want to learn about and play the game accordingly. The game can also provide information on specific landmarks, road conditions, and other unique features of each state.

In the country-wise mode, the game can focus on the overall traffic rules and regulations of different countries around the world. Users can select the country they want to learn about and play the game accordingly. The game can provide information on specific landmarks, road conditions, and other unique features of each country.

Both modes can provide an interactive and immersive experience for users to learn about different traffic rules, signs, and signals. Users can navigate through different scenarios and challenges, such as driving through busy intersections, following road signs and signals, and dealing with unexpected road hazards.

Overall, this game can be useful for anyone who wants to learn more about traffic rules and regulations in different regions and can be especially beneficial for individuals who are planning to travel or relocate to a new state or country. Virtual reality is an appropriate technology for this project as it provides an immersive and interactive experience for users to learn and practice driving skills in a safe and controlled environment.