



BSIT-S22-006

03-135191-016 Umer Saif

Detective Cop Game

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Information Technology

Supervisor: Ms. Summaira Nosheen

**Department of Computer Sciences
Bahria University, Lahore Campus**

Certificate



We accept the work contained in the report titled
“Detective Cop Game”

Written by
Umer Saif

as a confirmation to the required standard for the partial fulfilment of the degree of
Bachelor of Science in Information Technology.

Approved by:

Supervisor: Ms. Summaira Nosheen

Dec 23, 2022

(Signature)

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-135191-016	Umer Saif	

Date: Jan 10, 2023

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, **Ms. Summaira Nosheen** for her invaluable advice, guidance, and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

Umer Saif

ABSTRACT

The video game business has expanded significantly in recent years. Not only independent games, but also indie detective games, are becoming incredibly popular right now. To determine who committed a crime, detectives frequently employ deductive reasoning. These games enhance players' mathematical and problem-solving abilities.

Additionally, this game helps players strengthen their cognitive abilities, logical reasoning, and inner investigator. A project called "Police Detective Cop" aims to give users a realistic 3D simulation game experience. A police detective cop will wrap up the murder case's plot, which is tied to the drug lord, and gather the evidence needed to apprehend the perpetrator.

The player receives incentives and has their level raised after the investigator gathers the pieces of evidence. After accomplishing the objectives at each level, the player's customization will also get better. There are three levels in the game, and by completing all the objectives, the player has correctly identified the culprit. In Unity 3D, I design and create a police detective simulation game. Visual Studio, C#, Story Blog (Sounds), and SketchFeb are the tools used to construct the game (for 3D probs).

TABLE OF CONTENTS

TABLE OF CONTENTS	v
DECLARATION	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF SYMBOLS / ABBREVIATIONS	xi
CHAPTER 1	1
INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	1
1.3 Aims and Objectives	2
1.4 Scope of Project	2
CHAPTER 2	3
SOFTWARE REQUIREMENT SPECIFICATION	3
2.1 Introduction	3
2.1.1 Overview	3
2.1.2 Functional Requirements	4
2.1.3 Non-Functional Requirements	4

CHAPTER 3	5
DESIGN AND METHODOLOGY	5
3.1 Methodology	5
3.2 Feature Driven Development	5
3.3 Research	6
3.4 Unity 3D	6
3.5 Planning	6
3.6 Design	7
3.6.1 System Case Diagram	7
3.6.2 Sequence Diagram	8
3.6.3 Class Diagram	9
3.6.4 Class Diagram	11
3.7 Use Cases	12
3.7.1 Use Case 1 (New Game)	12
3.7.2 Use Case 2 (Rate Us & More Games)	13
3.7.3 Use Case 3 (Exit Game)	14
3.7.4 Use Case 4 (Sound Button)	15
3.7.5 Use Case 5 (Level 1)	16
3.7.6 Use Case 6 (Level 2)	17
3.7.7 Use Case 7 (Level 3)	18
3.7.8 Use Case 8 (Pause Button)	19
3.7.9 Use Case 9 (Level Complete & Level Fail)	20
3.8 Feature Build-Development, Testing, Sample	21
CHAPTER 4	22
IMPLEMENTATION	22
4.1 Introduction	22
4.2 Story Board	22
4.3 Game Design and Development	23
4.4 Third Person View	24
4.5 Realistic Car Controller Package	24

4.6	Do Tween	25
4.7	Cinemachine	25
4.8	Control Freak	25
4.9	Urban Traffic System	26
CHAPTER 5		27
RESULTS AND USER MANUAL		27
5.1	Results and User Manual	27
5.1.1	Splash Screen	27
5.1.2	Main Menu Screen	28
5.1.3	Level Selection	29
5.1.4	Game Controller	30
5.1.5	Gameplay	30
5.1.6	How to Play	31
CHAPTER 6		34
CONCLUSION AND RECOMMENDATIONS		34
6.1	Conclusion	34
6.2	Recommendations	34
REFERENCES		36
APPENDIX		37

LIST OF FIGURES

Figure 1: System Diagram	8
Figure 2: Sequence Diagram	9
Figure 3: Class Diagram	11
Figure 4: Use Case (New Game)	12
Figure 5: Use Case (Rate Us & More Games)	13
Figure 6: Use Case (Exit Game)	14
Figure 7: Use Case (Sound Button)	15
Figure 8: Use Case (Level 1)	16
Figure 9: Use Case (Level 2)	17
Figure 10: Use Case (Level 3)	18
Figure 11: Use Case (Pause Button)	19
Figure 12: Use Case (Level Complete & Level Fail)	20
Figure 13: Splash Screen	28
Figure 14: Main Menu Screen	29
Figure 15: Level Selection Screen	29
Figure 16: Controller Screen	30
Figure 17: Gameplay Screen	31
Figure 18: Use this to move arounds	31
Figure 19: Use this to stop the car	32

Figure 20: Use this to turn left	32
Figure 21: Use this to run fast	33
Figure 22: Activity Script	37
Figure 23: Activity Manager Script	37
Figure 24: Timeline Events Script	38
Figure 25: Timeline Controller Script	38
Figure 26: Environment Scene Load Manager Script	39
Figure 27: Main Menu Script	39
Figure 28: Splash	40
Figure 29: Task Manager Script	40
Figure 30: Button Assign Events Script	41
Figure 31: Level Selection Script	41
Figure 32: Trigger Events Script	42
Figure 33: UI Manager Script	42
Figure 34: Camera Focus Script	43
Figure 35: Camera Input Script	43
Figure 36: Camera Manager Script	44
Figure 37: Player Manager Script	44
Figure 38: Plagiarism Report	45

LIST OF TABLES

Table 1: New Game	12
Table 2: Rate Us & More Games	13
Table 3: Exit Game	14
Table 4: Sound Button	15
Table 5: Level 1	16
Table 6: Level 2	17
Table 7: Level 3	18
Table 8: Pause	19
Table 9: Level Complete & Level Failed	20

LIST OF SYMBOLS / ABBREVIATIONS

<i>TPV</i>	Third Person View
<i>UTS</i>	Urban Traffic System
<i>CF</i>	Control Freak
<i>RCC</i>	Realistic Car Controller
<i>DT</i>	Do Tween
<i>ITPC</i>	Investor Third Person Controller
<i>FDD</i>	Feature Driven Development
<i>SRS</i>	Software Requirements Specification

CHAPTER 1

INTRODUCTION

1.1 Background

Everyone loves detective games! A good mystery is always just around the corner (if you know where to look that is). If your love for the genre is only matched by the thrill of hunting said mystery, there is a high chance you've watched at least one crime TV show, played some CSI games, or read a nail-biting book [1].

Games that challenge our strategic thinking and perception of details are arguably more than just games. Detective games are one such example, and just like you might expect, they come in a multitude of shapes and forms. To name a few, you have some rich story-driven games, the ones that cleverly combine intricate puzzles with mysteries and hidden object games.

1.2 Problem Statement

Despite being a perennially popular genre in television and cinema, detective stories are less common in video games. This is due to how challenging it is to successfully translate the investigative process into a playable form. Even while the plots of adventure games

frequently involve secrets, most of the player's time is still spent fighting, shooting, and leaping across chasms. Game designers must perform a delicate balancing act when they decide to omit action altogether and instead focus on observation and deduction. The player must be given enough hints to be able to solve the case, but not too many that they feel patronised.

1.3 Aims and Objectives

The objectives of the thesis are shown as following:

- To create a game that will spark the audience's curiosity about solving mysterious crime cases.
- Knowing a client's mental capacities and the ratio of their strengths and limitations based on how quickly they react to any actions is very useful for physiologists.

1.4 Scope of Project

I developed this project to convert real-life scenarios into 3D simulations game. After gathering the stats of current crime ratios, this project develops to appreciate the work of the police department and the approach of detectives to catch synchronized crime. The audience will experience the real-life experience of a detective. In this project, the multiple real-life waste scenarios will be covered and all aspects of chasing the criminal. We can add unlimited real-life crime chase stories in upcoming seasons.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

2.1 Introduction

The **Software Requirements Specification** chapter is intended to give a complete overview of Android game project (Detective Cop Simulator). The SRS chapter details all feature upon which Detective Cop Game have decided with reference to the matter and importance to their implementation. SRS of game have functional and non-functional requirements.

2.1.1 Overview

The player receives incentives and has their level raised after the investigator gathers the pieces of evidence. After accomplishing the objectives at each level, the player's customization will also get better. There are five levels in the game, and by completing all the objectives, the player has correctly identified the culprit. In Unity 3D, I design and create a police detective simulation game.

2.1.2 Functional Requirements

- Start of the game, we have a splash screen (Loading Screen)
- Game have projected on an android mobile
- Game provide Third person view to the player
- Game provides sound effects, ON or OFF
- Main Menu allows the player to play the game, exit the game, and play more games by using MORE GAMES button and review about us the game
- Level selection allows the player to select the level and play, in every level player can resume the level, restart the level, and go back to Level Selection/Main Menu
- We have left or right movements of player, detect directions of player and gives a tutorial to user (How to play)
- User can easily understand the tasks in the game and trigger actions
- User can pause and resume the game
- User can view tutorial or demo

2.1.3 Non-Functional Requirements

- The mobile display should have a high resolution
- The maximum number of click for any function in the system does not exceed 2 limits
- The average response time between click and reaction is always less than 0.5 seconds
- The game can run smoothly with 4 GB of Ram which is every common now in every Android phone
- The code written for the game is maintainable hence in return the system is maintainable to add new features to the newer versions
- Since the last recoil game does not create profile for users, it does not need to access user credential, therefore, security is not a concern in this application
- The game can be extended with other functionalities required

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 Methodology

The methodology we are using in our project is feature-driven development, which is driven by an agile framework.

3.2 Feature Driven Development

Feature Driven Development is an **agile methodology** that organises software development around achieving progress on features, as the name implies [2]. However, features in the FDD context are not always product characteristics in the conventional sense. They're more equivalent to user stories in Scrum. In other words, in the Feature Driven Development methodology, "finish the login procedure" might be considered a feature.

3.3 Research

A good mystery is one of the most entertaining aspects of a piece of literature. Those implausible crimes that get your head churning. Inexplicable killings, enigmatic disappearances, and intriguing puzzles. While reading or watching such stories in novels, TV series, or movies is entertaining, games allow you to go one step further and participate in the investigation yourself, feeling the joy of solving a case firsthand. The complete knowledge of the scope of the project and the current of domain targeted and analyze. Unity 3D is a powerful cross-platform 3D engine and a user-friendly environment. We developed our 3D game on unity 3D in which the first developed the interactive design and further coding.

3.4 Unity 3D

Unity3D is a robust cross-platform 3D engine with an easy-to-use development environment. Unity should appeal to everyone who wants to simply create 3D games and applications for mobile, desktop, the web, and consoles. It is simple enough for beginners yet powerful enough for experts [3].

3.5 Planning

We create the feature list and plan for the features of the project. System requirements and tools required are gathered. Features Selection of the game and planning.

3.6 Design

All possible UML diagram is designed here e.g., System Case, Class, and Sequence diagram design also, phase deals with the interface designing and prototype.

3.6.1 System Case Diagram

A system case diagram in the **Unified Modeling Language** can describe the specifics of your system's users (also known as actors) and their interactions with the system. To make one, you'll need a particular collection of symbols and connections. A good use case diagram can assist your team in discussing and representing:

- Interactions between your system or application and people, organisations, or external systems
- Goals that your system or application assists those entities (referred to as actors) in achieving
- The extent of your system

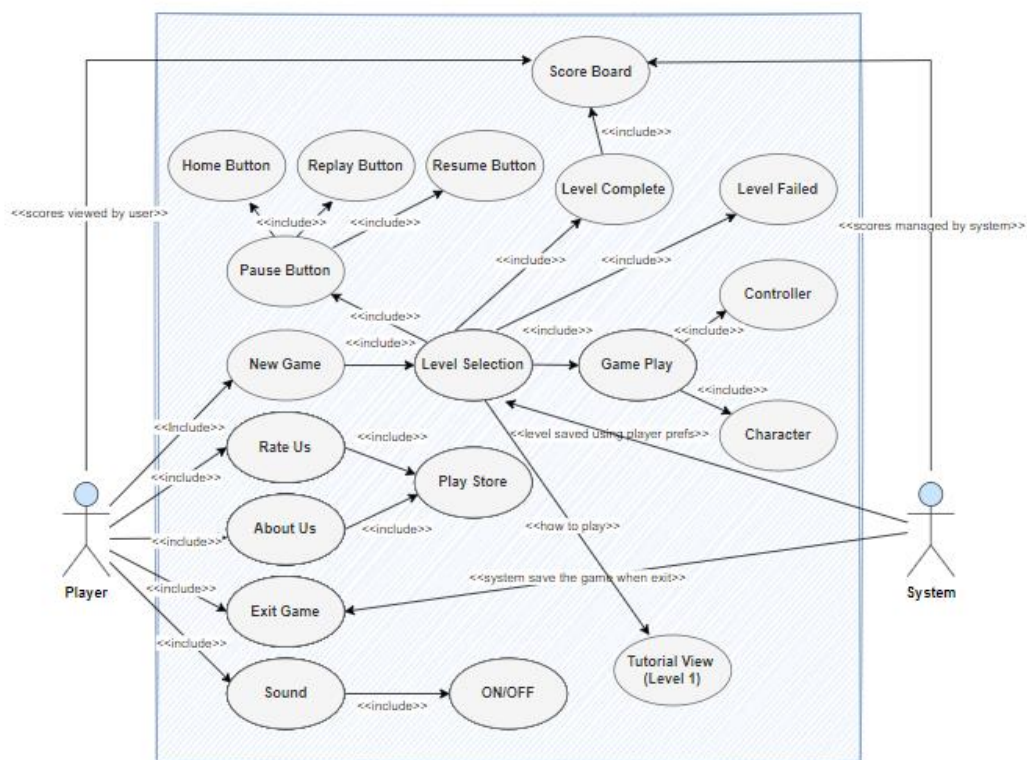


Figure 1: System Diagram

3.6.2 Sequence Diagram

UML Sequence Diagrams are interaction diagrams that show how processes are performed. They record the interaction of items within the framework of a partnership. Sequence Diagrams are time focused, and they graphically express the sequence of the interaction by utilising the vertical axis of the diagram to indicate time, what messages are conveyed, and when.

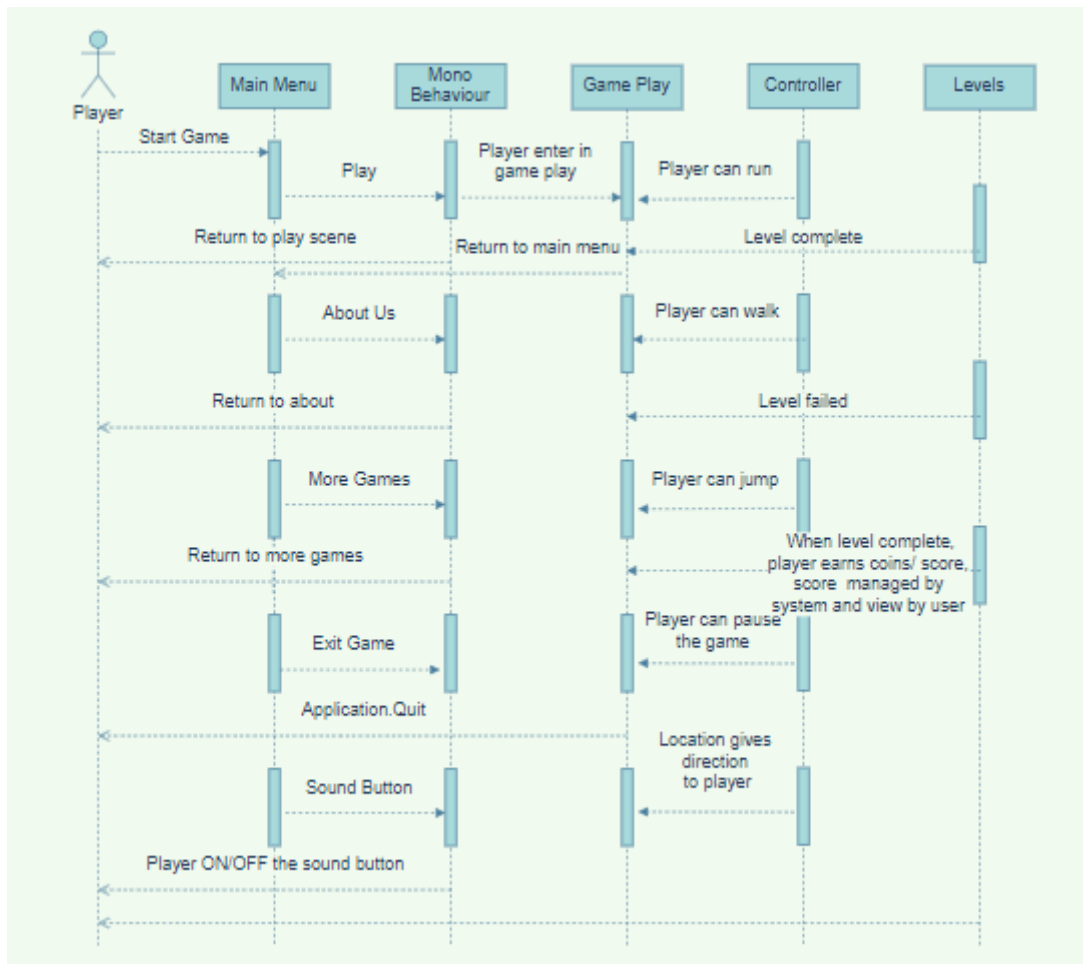


Figure 2: Sequence Diagram

3.6.3 Class Diagram

A **class diagram** in the Unified Modeling Language is a form of static structural diagram in software engineering that depicts the structure of a system by displaying the system's classes, their attributes, operations (or methods), and the connections between objects. Classes of detective cop simulator are:

1. Task Manager
2. Trigger Events

3. Activity Manager
4. Level Selection
5. Main Menu
6. Splash
7. Activity
8. Camera Focus
9. Player Manager
10. Camera Input
11. Camera Manager
12. Loading
13. Fading
14. Environment Load Scene Manager
15. Delay Event Trigger

3.6.4 Class Diagram

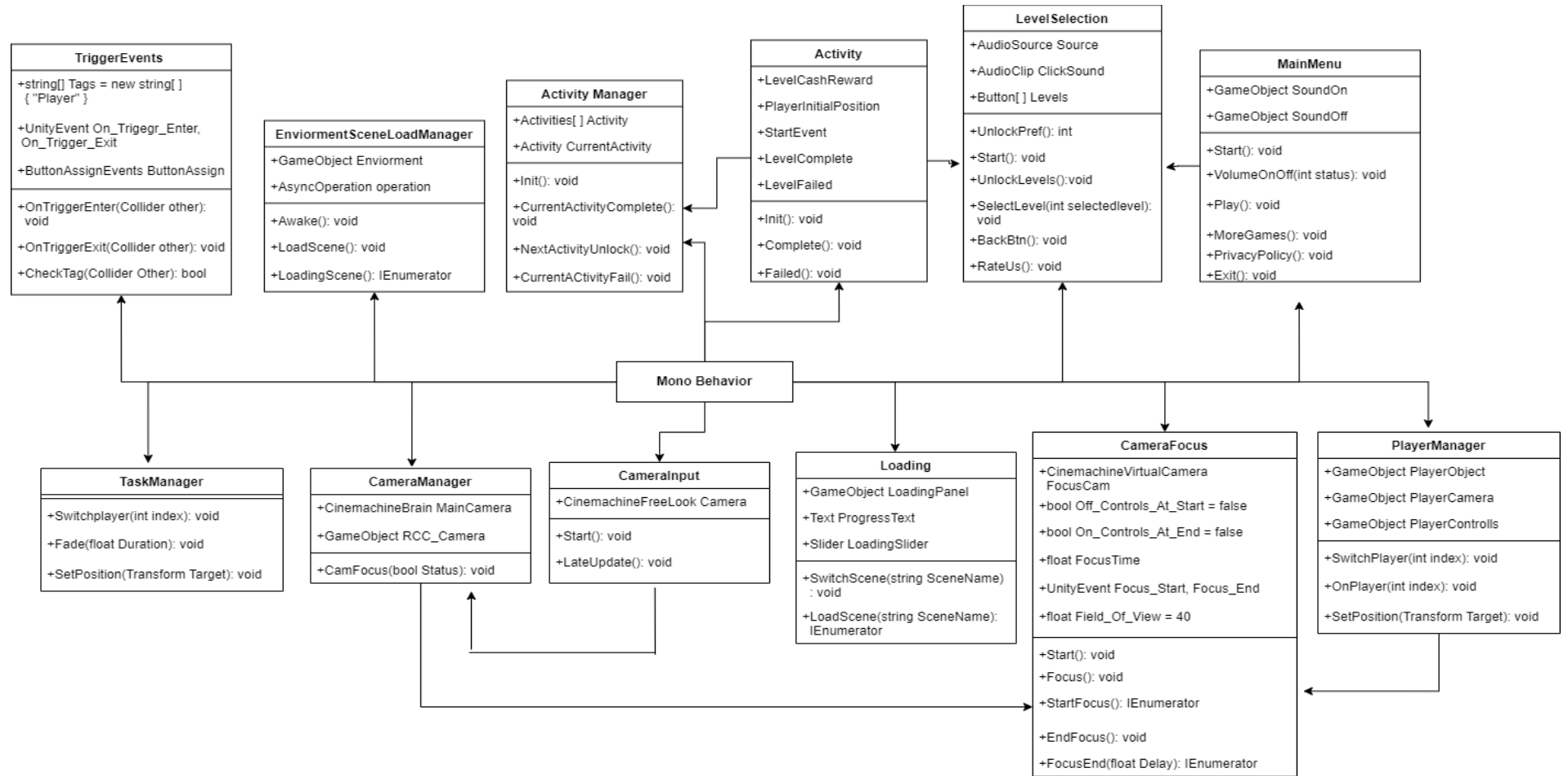


Figure 3: Class Diagram

3.7 Use Cases

3.7.1 Use Case 1 (New Game)

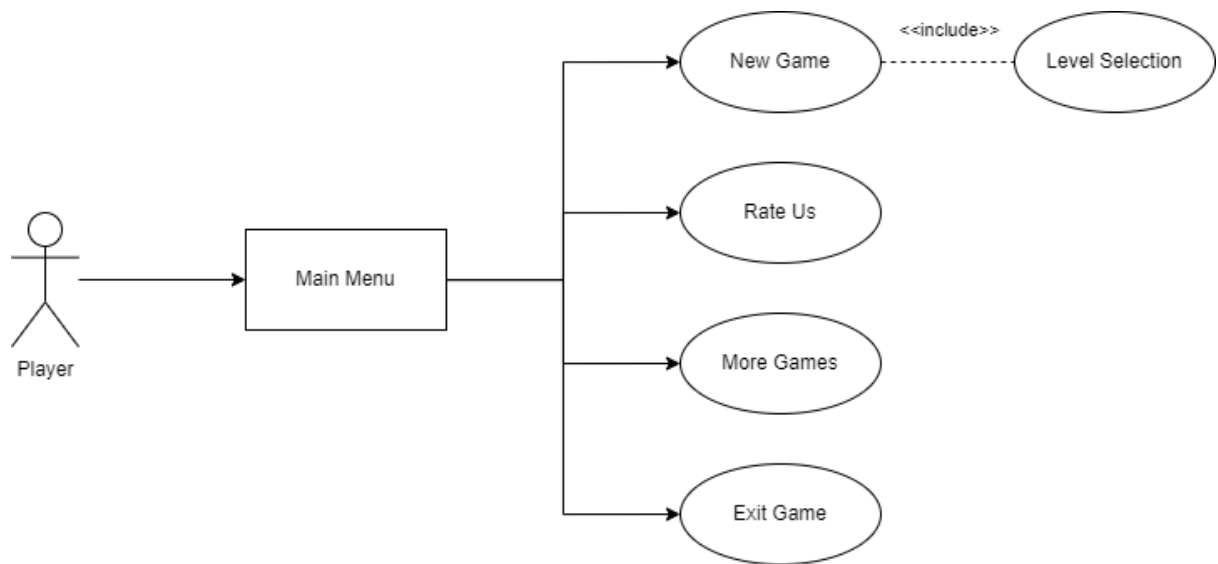


Figure 4: Use Case (New Game)

Name:	New Game
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	Player initiates new game function by clicking the new game button.
Exit Conditions:	Game is now in a playable state.

Table 1: New Game

3.7.2 Use Case 2 (Rate Us & More Games)

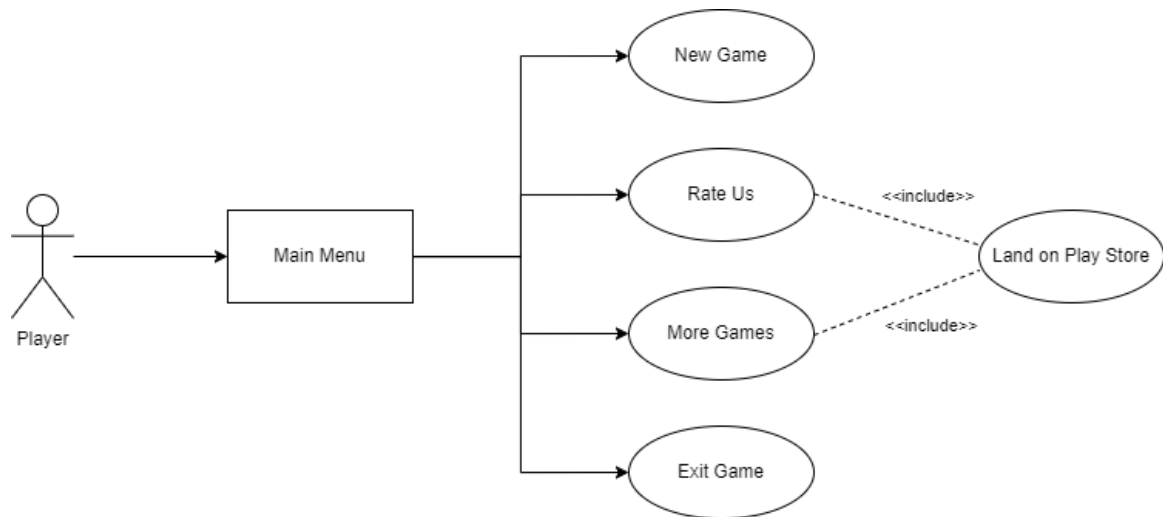


Figure 5: Use Case (Rate Us & More Games)

Name:	Rate Us & More Games
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	Player clicks on rate us button to rate the game by giving stars and click on about us button to land on the play store to check about the game.
Exit Conditions:	Game and developers' info is displayed.

Table 2: Rate Us & More Games

3.7.3 Use Case 3 (Exit Game)

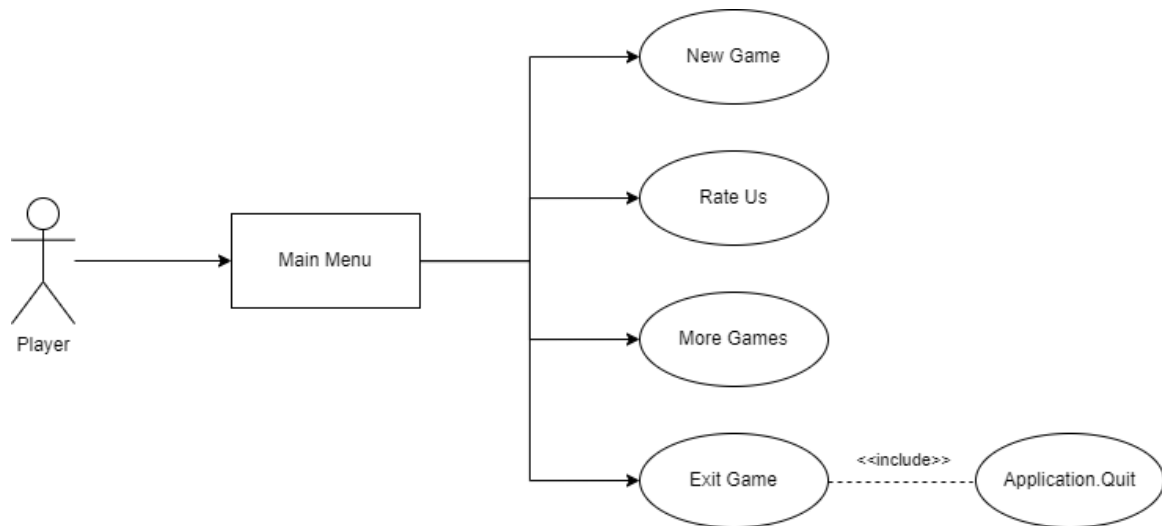


Figure 6: Use Case (Exit Game)

Name:	Quit
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	Player selects quit function by clicking the exit button. Player is asked to confirm his selection.
Exit Conditions:	Game is not in running condition anymore.

Table 3: Exit Game

3.7.4 Use Case 4 (Sound Button)

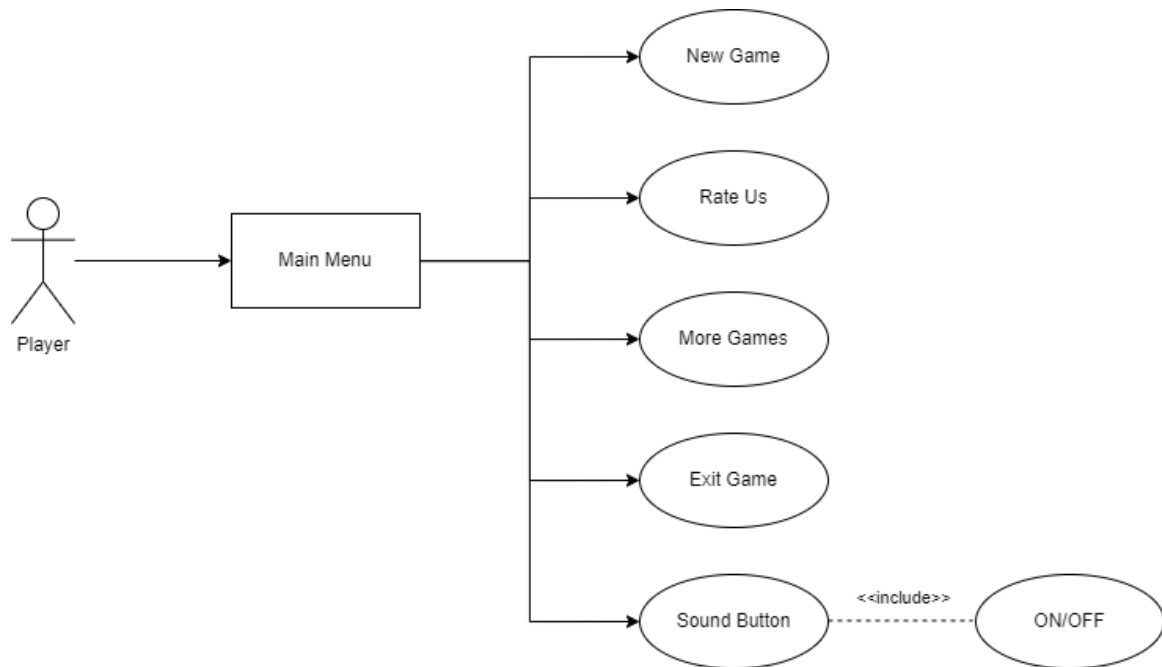


Figure 7: Use Case (Sound Button)

Name:	Sound
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	Player can ON/OFF the sound of the game by clicking sound button.
Exit Conditions:	Game is in a playable state.

Table 4: Sound Button

3.7.5 Use Case 5 (Level 1)

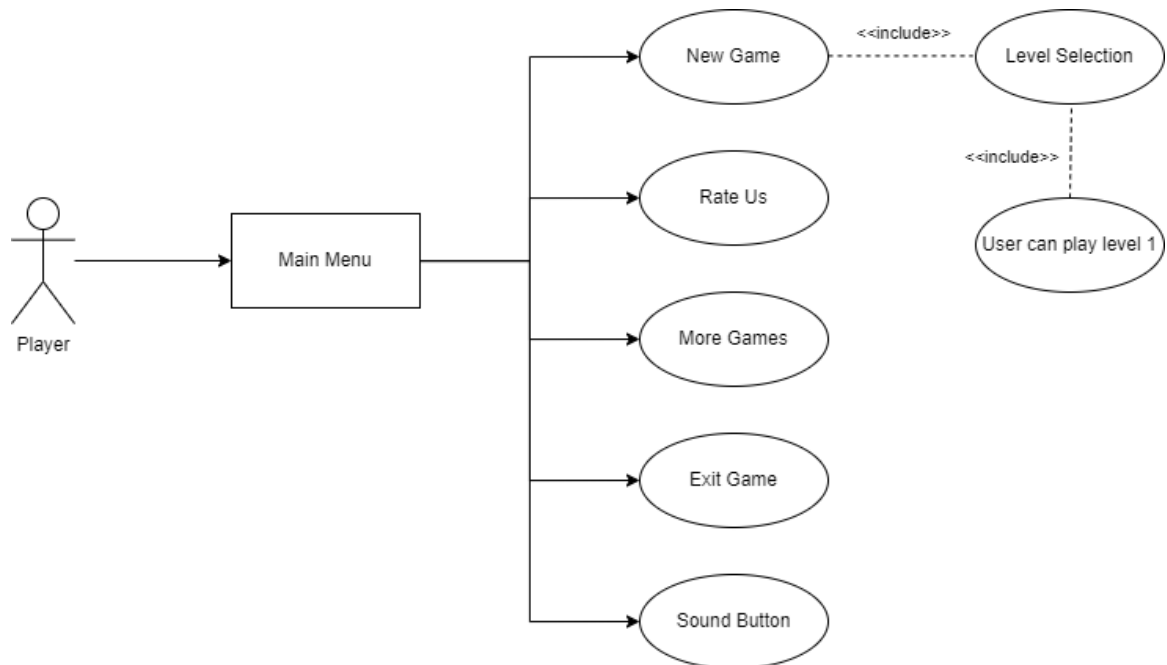


Figure 8: Use Case (Level 1)

Name:	Level 1
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	In first level, we guide the Player how to play and player can follow the tutorials, the first level is tutorial based and have check points to complete the level.
Exit Conditions:	Player must complete level, otherwise level failed.

Table 5: Level 1

3.7.6 Use Case 6 (Level 2)

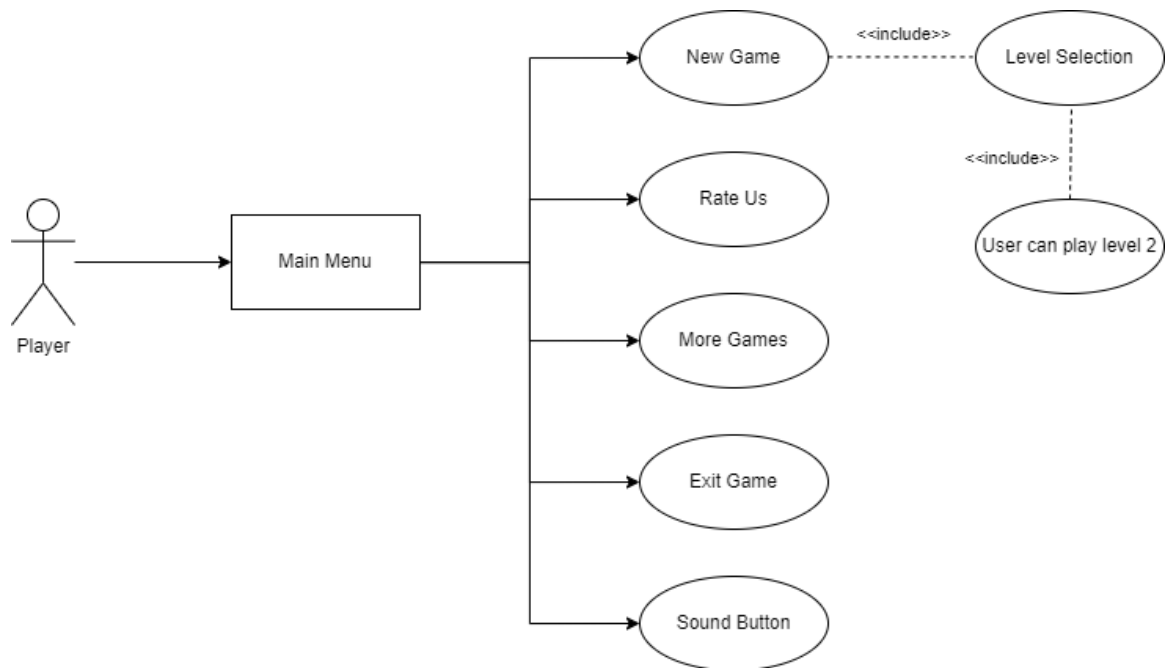


Figure 9: Use Case (Level 2)

Name:	Level 2
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	In the second level, player chase the criminal and caught the criminal by hitting car with the criminal AI car.
Exit Conditions:	Player must complete level, otherwise level failed.

Table 6: Level 2

3.7.7 Use Case 7 (Level 3)

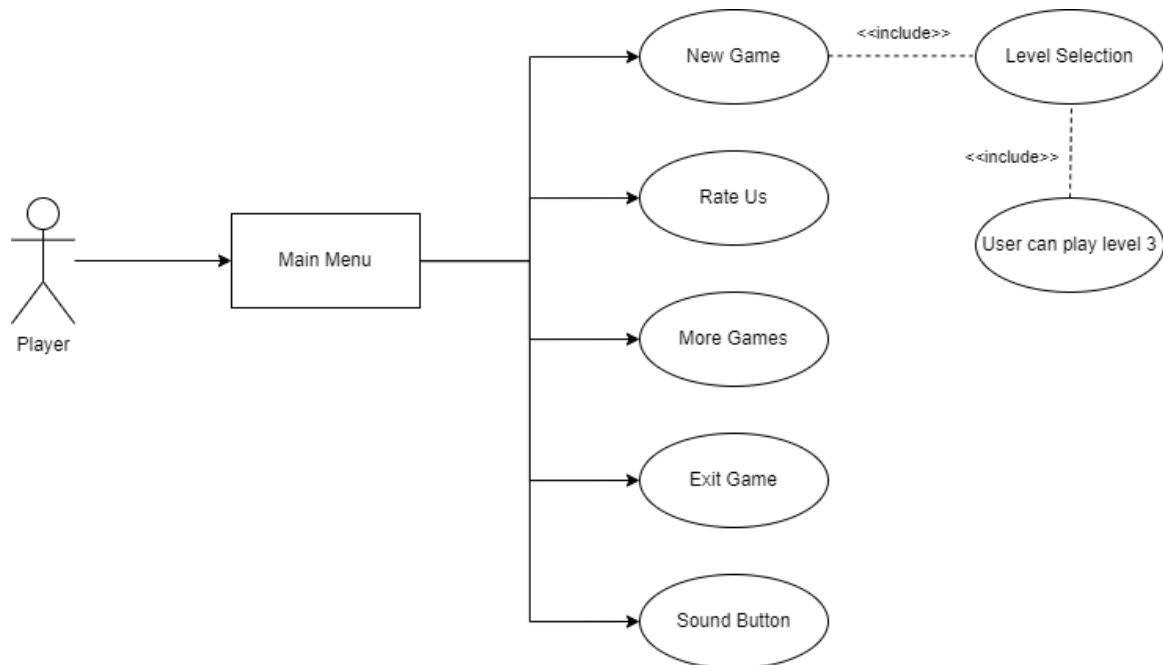


Figure 10: Use Case (Level 3)

Name:	Level 3
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	In the third level, we have 2 Ai cars that stops the player to reach the trigger point.
Exit Conditions:	Player must complete level, otherwise level failed.

Table 7: Level 3

3.7.8 Use Case 8 (Pause Button)

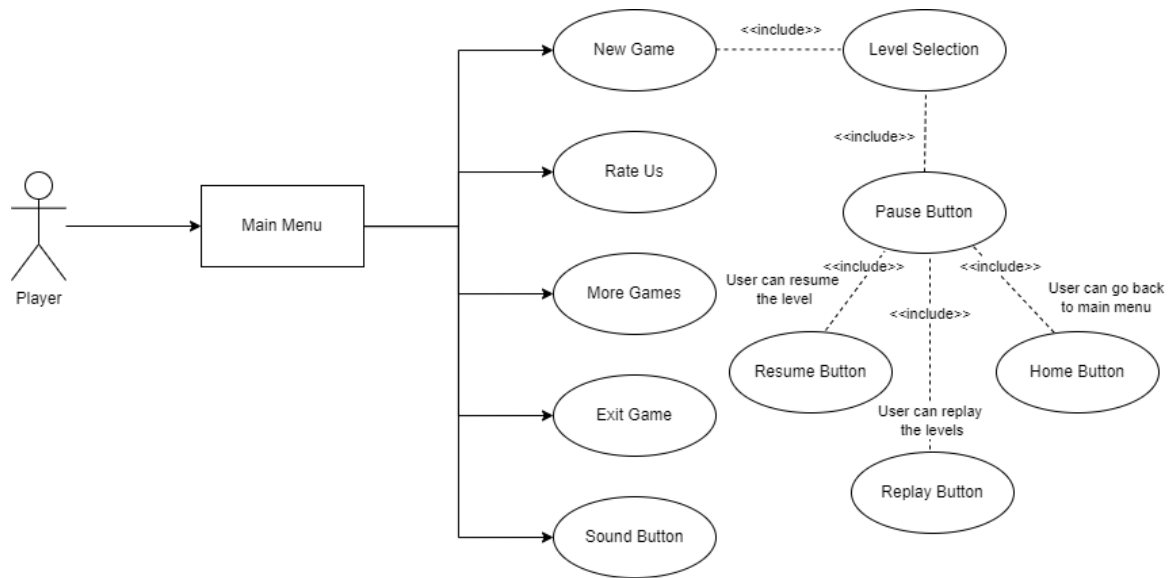


Figure 11: Use Case (Pause Button)

Name:	Pause
Actor:	Player
Entry Conditions:	Application is running. A Game is not currently in progress.
Flow of Events:	Player can pause the game by clicking pause button. In which, player can go back to main menu, replay the level, and resume the level.
Exit Conditions:	Game is not in a playable state.

Table 8: Pause

3.7.9 Use Case 9 (Level Complete & Level Fail)

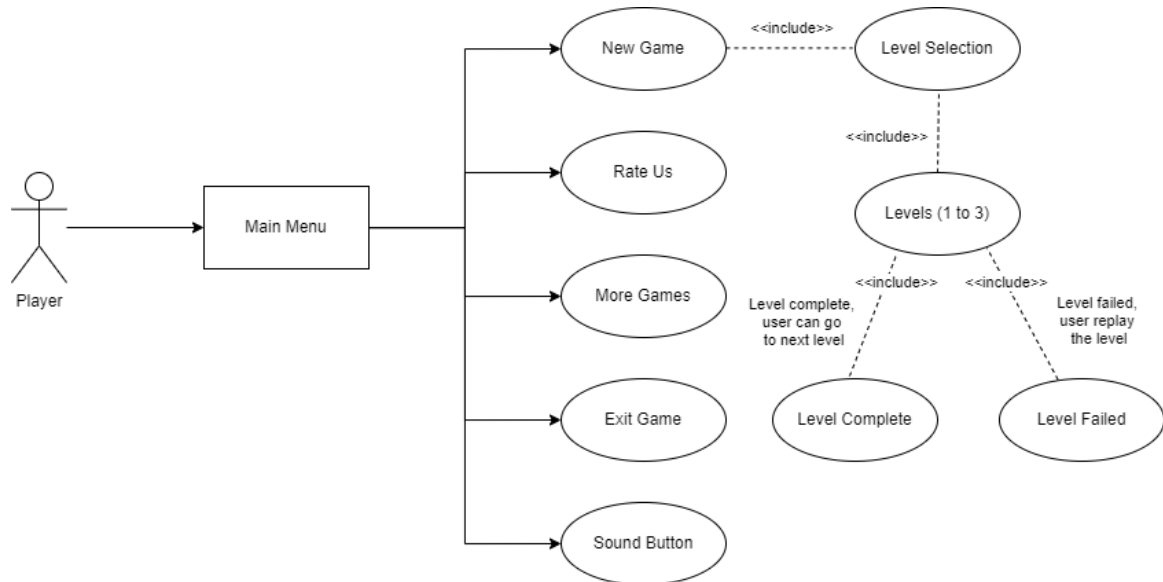


Figure 12: Use Case (Level Complete & Level Fail)

Name:	Level Complete & Level Failed
Actor:	Player
Entry Conditions:	Application is running.
Flow of Events:	Player can complete level to unlock next level otherwise, level failed, and player must play again.
Exit Conditions:	Game is in a playable state.

Table 9: Level Complete & Level Failed

3.8 Feature Build-Development, Testing, Sample

Functions are developed based on the feature list. Game Engines such as Game Physics, Graphics. The interactive approach of feature building results in the final product.

Testing: Each module is tested after its development and these modules are written in implementation chapter and results are discuss in results and user manual chapter, where you can see every development process and work.

Setup: Tested modules are integrated here to build a complete working system. Integration testing is where software modules are integrated logically and tested as a group. A typical software project comprises of several software modules written by various programmers. The goal of this level of testing is to identify flaws in the interaction of various software components when they are combined.

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

The game is the simulation game where we have a story line. In which player perform multiple tasks given in the levels and when level completes, user get score. We are adopting agile development approach using Feature driven development to ensure deadlines and goals are met on time. The development stages are outlined below.

4.2 Story Board

In the start of the game, we have splash screen that takes 5 seconds to go to next scene, basically it's a loading screen, then the next scene is the main menu scene. In main menu we have a play button, more games button, rate us button and exit button to quit the game. When the user clicks on Play game button, user moves to new scene which is level selection scene, in which user can play three levels. In the more games button, user automatically switch to play store and app store where he/she can play more games. Rate us button also switch user to play store and app store, where user can rate the game and in exit button, user exit the game.

On the level selection screen, user has three levels to play. In which every level has different task to perform or to complete the level. First level is a tutorial base level, where we tell user “How to Play” the game where user follow the tutorial and then user must complete some checks points to complete the level.

In the second level, user must stop the drug dealer and hit the Ai car until the slider come down, when user hit the Ai car continuously, the slider comes down and the drug dealer captured. The slider value is the of Ai car, when user far away from the Ai car, slider goes to high value and when user comes to the Ai car, the Ai car slider decreases and mission complete otherwise not.

In the third level, user must reach on the target point, where two Ai cars stop the user to reach the target point. In this level we have a timer, user must reach on target spot on the given time otherwise the level fail.

4.3 Game Design and Development

The game is developed in unity 3D and the game is a simulation game. In the development of game, first we make the main menu of the game using unity SCI-FI package and put functionality in the menu items using scripts. Second, the game has a level selection scene that provide the user to play levels, user can play three levels in the game. We make a controller for character movements and camera movements in which multiple scripts are written. Game environment and RCC car controller package download by Unity Asset Store.

4.4 Third Person View

A third person view is a camera behaviour in which the camera is detached from the primary character's focus [4]. This camera behaviour is also known as a bird's eye view because the game world is viewed from an external viewpoint rather than the primary character's eyes. The shortcomings of the first-person camera are the advantages of the third-person camera: In contrast to the first-person view, the player gets a considerably wider field of view with this camera behaviour, allowing him to observe his primary character as well as view the 3D scenery from various perspectives. If a combat occurs in the video game, the third-person perspective is used.

A third-person camera has a significant impact on the visual depiction in the game. If the user has complete control over the camera and can thus precisely explore the game world, the game environment is no longer just the "background" of an image.

4.5 Realistic Car Controller Package

Stylized Vehicles provides ready to use drag-and-drop vehicle prefabs. The package does not include any Stylized Vehicles assets. The Stylized Vehicles bundle offers drag-and-drop vehicle prefabs. It's never been easier to design your own realistic automobile. It only takes approximately 5 minutes to create a fully functional automobile. Switch controllers, switch behaviors, adjustments all with a single click. Simple to use and highly adaptable. It has ten pre-configured vehicle behaviors. Platforms tested include PC, Mac, Android, and iOS. With just one line of code, you can instantiate additional cars at a specific point, customize, operate, or alter behavior [5].

4.6 Do Tween

Do Tween is a fast, efficient, fully type-safe object-oriented animation engine for Unity that is optimised for C# users. It is free and open-source, and it has a plenty of advanced capabilities. It is also a development of my earlier Unity tween engine, HO Tween. DO Tween is more than 400% quicker, more efficient, type-safe, saves unnecessary GC allocations, and provides new shortcuts and features [6].

4.7 Cinemachine

Cinemachine is a modular camera tool suite for Unity that provides AAA game quality controls for each camera in your project. It's a simple plugin that allows you to add functionality to existing cameras or create new ones with astonishing characteristics [7].

Cinemachine is intended to be the full unified camera system in your project, but it can also be used in conjunction with your existing cameras. There's no problem if you already have a bunch of camera stuff running and simply want to use Cinemachine for cutscenes or something. However, when used throughout your project, it allows you to blend any camera to any other camera in a seamless gameplay-to-cutscene-and-back transition.

4.8 Control Freak

Control Freak is a cross-platform input system that works with multi-touch, accelerometers, keyboards, mice, and gaming controllers. Create a cross-platform game without touching a single line of code! Controls on-screen can be made up of buttons, joysticks, steering wheels, track pads, and super touch zones. Touch control states can be assigned to virtual

keys, named buttons, and axes. Super Touch Zone, the pack's most advanced control type, allows you to bind every imaginable finger gesture [8].

For example, a single-finger tap can emulate a right mouse button click, swiping horizontally with three fingers can emulate a mouse scroll wheel, and a two-finger twist gesture can be bound to an analogue axis that rotates the camera!

4.9 Urban Traffic System

The asset consists of 31 lowpoly models of various modes of transportation, each with one LOD (Level of Detail), and 11 models of humans, each with four LODS, which are required for the construction of urban traffic [9]. Examples include:

1. Mountain bike
2. Moped
3. Gyro Scooters
4. Vans
5. Motorcycle
6. Cars
7. Trucks
8. Semi
9. Buses

This item supports mobile platforms, but only if there are a few automobiles and people in the area. Otherwise, poor phones can't handle as many polygons. Because all the low-poly models have high-quality textures. We strive to create a better traffic system and the city's population. Furthermore, all vehicles have the LOD.

CHAPTER 5

RESULTS AND USER MANUAL

5.1 Results and User Manual

In results, we show the outputs of the splash screen, main menu, level selection, and game play and show the output of controller. In user manual, we show the user “how to play the game”.

5.1.1 Splash Screen

First, we have a splash screen when start game, basically it’s a loading screen, in which you have wait for seconds and then it goes to main menu scene.

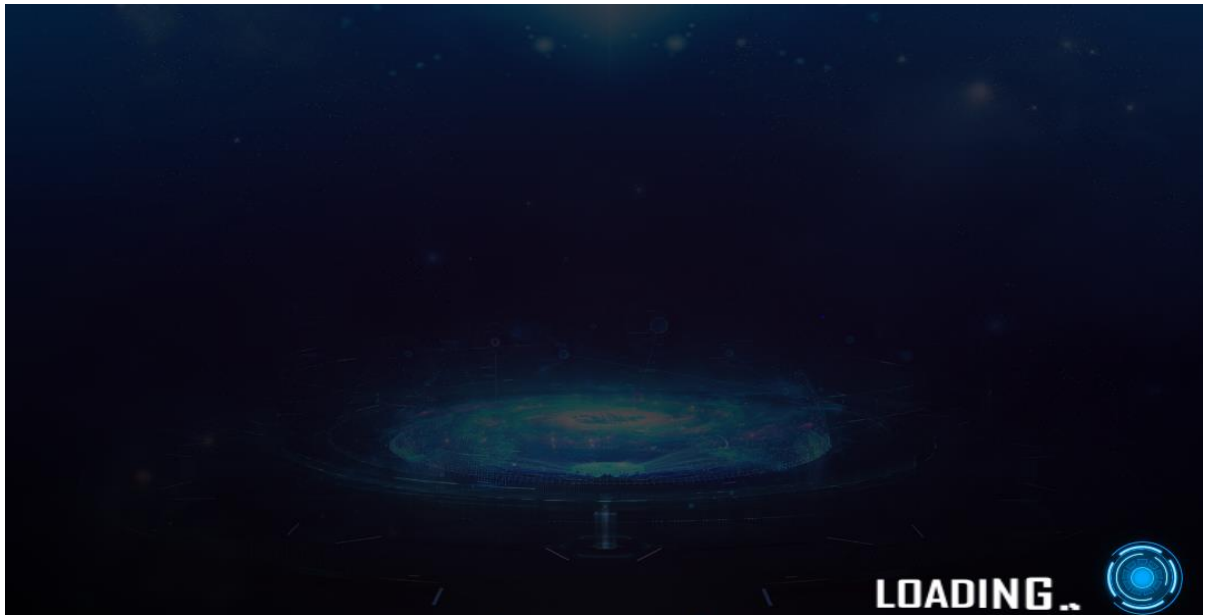


Figure 13: Splash Screen

5.1.2 Main Menu Screen

Main menu screen has four buttons in which buttons perform their actions and goes to next scene. User can play game, rate the game, play more games using more games buttons and exit the game. The buttons of main menu given in below figure.



Figure 14: Main Menu Screen

5.1.3 Level Selection

In level selection screen user can select level from 1 to 3. Users must complete the first level to go on next level. Every level has different tasks to perform, like meet the checkpoints.



Figure 15: Level Selection Screen

5.1.4 Game Controller

The Controller have multiple scripts that control the movements of player, movements of camera, user moves the character anywhere and its third person view in which user can view around the environment. User can see his/her location while chasing criminal, pause the game and subtitles that's guide the user.



Figure 16: Controller Screen

5.1.5 Gameplay

Gameplay is the way in which players engage with a game, particularly video games. Gameplay is the pattern set by the game rules, the connection between the player and the game, the problems and how to overcome them, the plot and the player's relationship to it.



Figure 17: Gameplay Screen

5.1.6 How to Play

In this scene, user follow the instruction given move around anywhere in the environment, these instructions give help to user “**How to play**” the game.

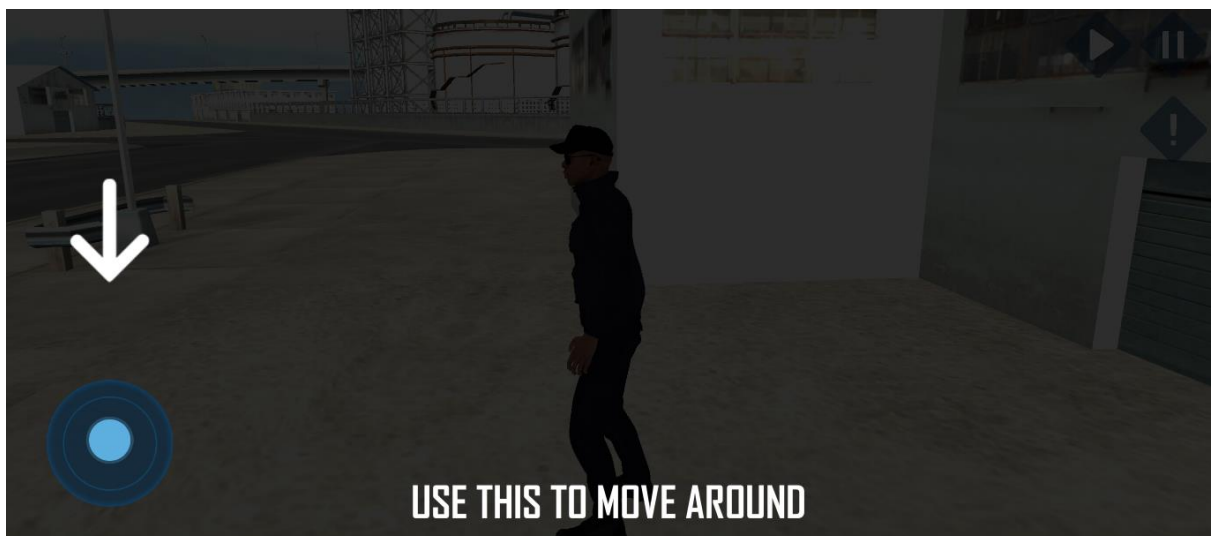


Figure 18: Use this to move arounds

In this scene, when user get in the car, player can move the car left to right and press the stop button to stop the car and these instructions give help to user.



Figure 19: Use this to stop the car

In this scene, user learn how to turn the car, tutorials tell user to move car left to right by given instructions, which display in the screen.

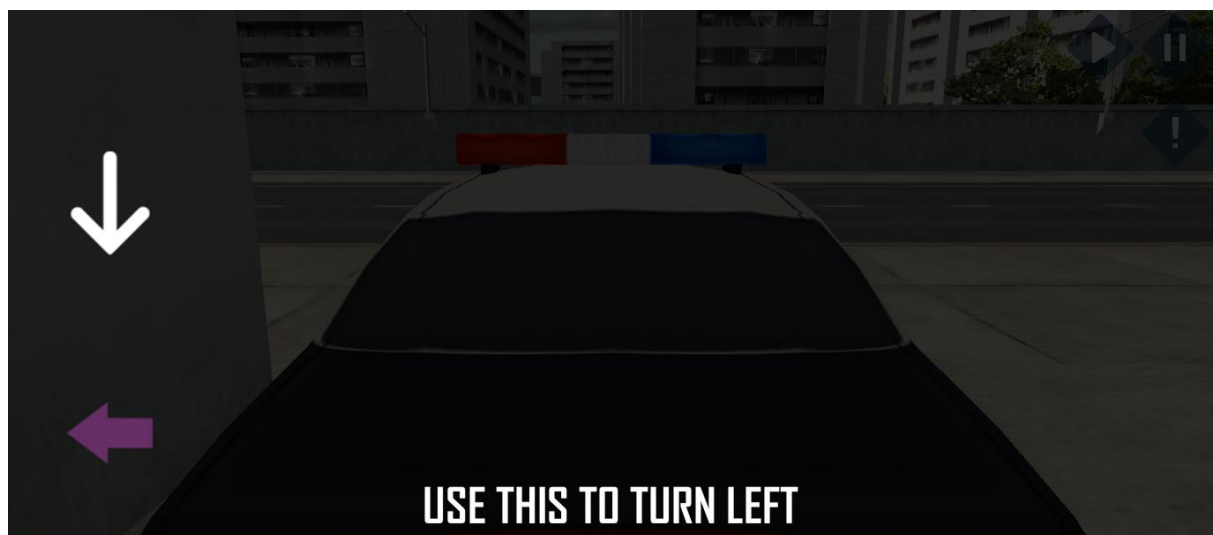


Figure 20: Use this to turn left

In this acene, user use this button to run fast and cover distance fastly and complete tasks easily. These instruction gives help to user.

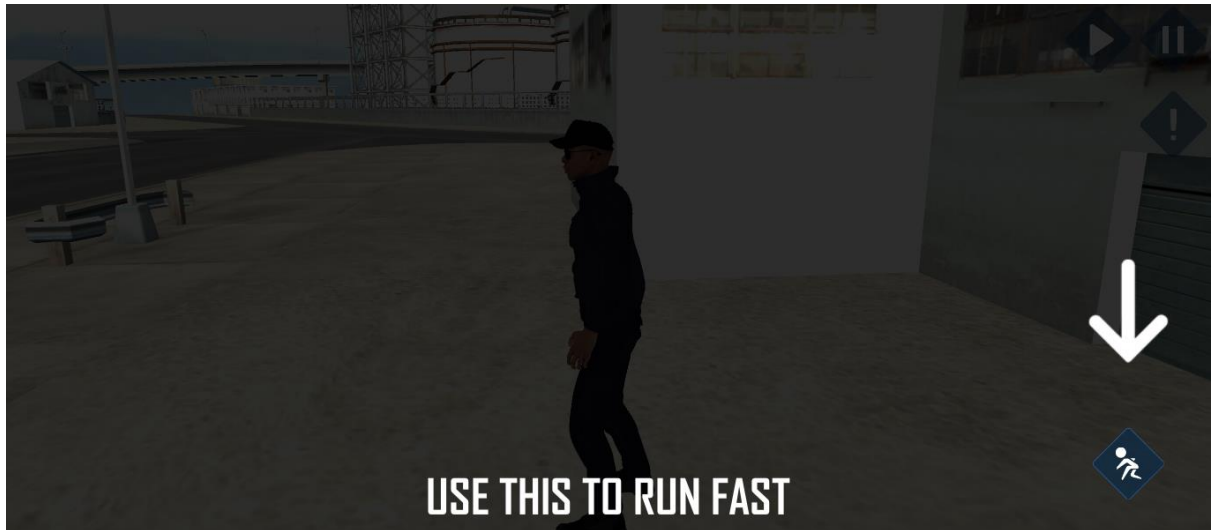


Figure 21: Use this to run fast

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

I developed this project to convert real-life scenarios into 3D simulations game. After gathering the stats of current crime ratios, this project develops to appreciate the work of the police department and the approach of detectives to catch synchronized crime. The audience will experience the real-life experience of a detective. In this project, the multiple real-life waste scenarios will be covered and all aspects of chasing the criminal. We can add unlimited real-life crime chase stories in upcoming seasons. Game has main menu, level selection, loading screens and have multiple levels to play, in which user complete levels and earn score/coins.

6.2 Recommendations

As per our recommendations, designed 3D virtual game that captures and interpret the hand gestures of the player and act accordingly in the game. The hand gestures are captured using leap motion device. Leap motion captures the bone structure and joints of hands to trace the position of hand. It is very precise and accurate. The game runs in PC and mobile device acts as a display for the game in VR device. Any android phone can be used to act as a display for the PC game.

Uniquely designed gadgets give real feeling and sensations of the game while it is being played. The player not only give input to the game but also get feedback from the virtual world. This makes a two-way interaction between the player and the virtual game environment.

REFERENCES

- [1] G. Lees, The best detective games, 2022, January 7.
- [2] Feature driven development (FDD), December 14, 2022.
- [3] I. Zamojc, Introduction to Unity3D, 2012, May 17.
- [4] Opsive, Third Person Controller, December 14, 202.
- [5] BoneCracker, Realistic car controller, December 14, 2022.
- [6] Demigiant.com, DOTween, December 14, 2021.
- [7] Unity3d.com, About Cinemachine, March 10, 2022.
- [8] T. I. M. Easy!, Control Freak, 2020.
- [9] Unity.com, Urban Traffic System, October 12, 2021 .
- [10] N. Ozcelik, “The 10 best detective games for Android,” Mobile Marketing Reads,” 2022. [Online].

APPENDIX

APPENDIX A: Computer Programme Listing

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Events;
5
6  public class Activity : MonoBehaviour
7  {
8      public int LevelCashReward;
9      public Transform PlayerInitialPosition;
10
11     public UnityEvent StartEvent, LevelComplete, LevelFailed;
12
13     1 reference
14     public void Init()
15     {
16         PlayerManager.Instance.SetPosition(PlayerInitialPosition);
17         StartEvent.Invoke();
18     }
19
20     1 reference
21     public void Complete()
22     {
23         LevelComplete.Invoke();
24     }
25
26     1 reference
27     public void Failed()
28     {
29     }
30 }

```

Figure 22: Activity Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ActivityManager : MonoBehaviour
6  {
7      [Singleton]
8
9      public Activities[] Activity;
10     [HideInInspector]
11     public Activity CurrentActivity;
12
13     1 reference
14     public void Init()
15     {
16         for (int i = 0; i < Activity.Length; i++)
17         {
18             Activity[i].Activity.gameObject.SetActive(false);
19         }
20         // On The Current Level
21         Activity[PlayerPrefs.GetInt("CurrentActivity")].Activity.gameObject.SetActive(true);
22         CurrentActivity = Activity[PlayerPrefs.GetInt("CurrentActivity")].Activity;
23         CurrentActivity.Init();
24     }
25 }

```

Figure 23: Activity Manager Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Events;
5
6
7  @ Unity Script | 0 references
8  public class TimelineEvents : MonoBehaviour
9  {
10     public bool CanSkip = false;
11     public GameObject Cameras;
12     public UnityEvent OnTimelineStart, OnTimelineEnd;
13
14
15     @ Unity Message | 0 references
16     private void OnEnable()
17     {
18         Cameras.SetActive(true);
19         OnTimelineStart?.Invoke();
20     }
21
22     @ Unity Message | 0 references
23     private void OnDisable()
24     {
25         Cameras.SetActive(false);
26         OnTimelineEnd?.Invoke();
27     }
28 }

```

Figure 24: Timeline Events Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Playables;
5
6
7  @ Unity Script | 0 references
8  public class TimelineController : MonoBehaviour
9  {
10     public PlayableDirector Director;
11
12
13     0 references
14     public void Play()
15     {
16         //if(((Time)Director.playableAsset).
17     }
18 }

```

Figure 25: Timeline Controller Script

The screenshot shows the Visual Studio IDE with the 'EnvironmentSceneLoadManager.cs' script open. The script is a Unity MonoBehaviour class that manages scene loading. It includes several using statements at the top, followed by a class definition with public fields for 'Environment' and 'operation', a private 'Awake()' method that calls 'LoadScene()', and a public 'LoadScene()' method that starts a coroutine. A reference to 'LoadingScene()' is also shown.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class EnvironmentSceneLoadManager : MonoBehaviour
7  {
8
9      public GameObject Environment;
10     AsyncOperation operation;
11
12     private void Awake()
13     {
14         LoadScene();
15     }
16
17
18     void LoadScene()
19     {
20         StartCoroutine>LoadingScene();
21     }
22
23     IEnumerator LoadingScene()
24     {

```

Figure 26: Environment Scene Load Manager Script

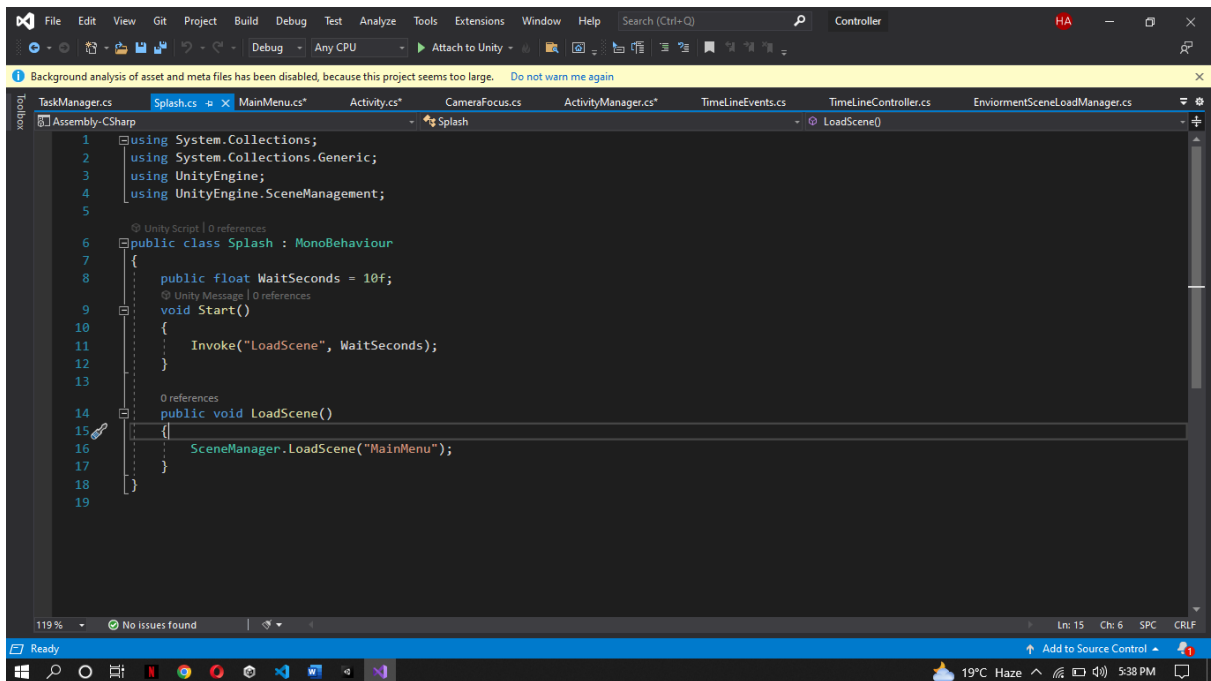
The screenshot shows the Visual Studio IDE with the 'MainMenu.cs' script open. The script is a Unity MonoBehaviour class that handles menu sounds. It includes using statements for 'UnityEngine.UI'. The class has public fields for 'SoundOn' and 'SoundOff', a private 'Start()' method that calls 'VolumeOnOff()', and a public 'VolumeOnOff()' method that uses an if-statement to toggle sound settings and adjust volume.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class MainMenu : MonoBehaviour
7  {
8      public GameObject SoundOn;
9      public GameObject SoundOff;
10
11     private void Start()
12     {
13         VolumeOnOff(PlayerPrefs.GetInt("SoundPref"));
14     }
15
16     public void VolumeOnOff(int status)
17     {
18         if(status == 1)
19         {
20             SoundOn.SetActive(true);
21             SoundOff.SetActive(false);
22             AudioListener.volume = 1f;
23         }
24         else
25     }

```

Figure 27: Main Menu Script

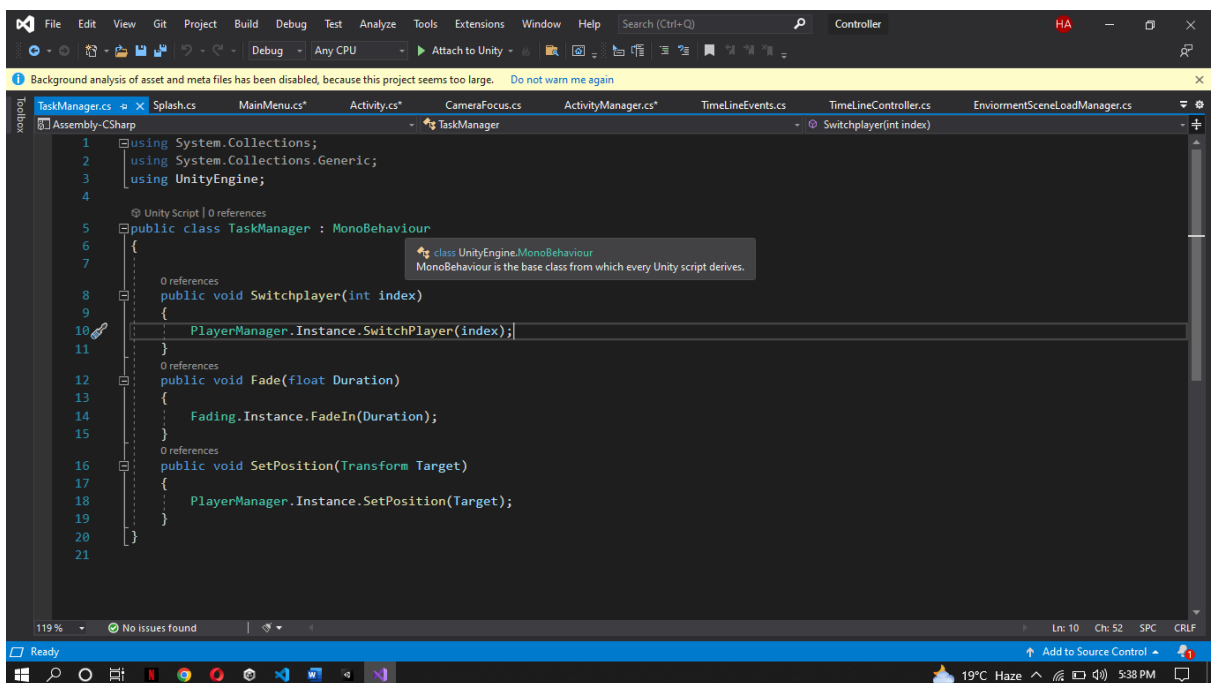


```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class Splash : MonoBehaviour
7  {
8      public float WaitSeconds = 10f;
9      void Start()
10     {
11         Invoke("LoadScene", WaitSeconds);
12     }
13
14     public void LoadScene()
15     {
16         SceneManager.LoadScene("MainMenu");
17     }
18 }
19

```

Figure 28: Splash



```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class TaskManager : MonoBehaviour
6  {
7      public void SwitchPlayer(int index)
8      {
9          PlayerManager.Instance.SwitchPlayer(index);
10     }
11
12     public void Fade(float Duration)
13     {
14         Fading.Instance.FadeIn(Duration);
15     }
16
17     public void SetPosition(Transform Target)
18     {
19         PlayerManager.Instance.SetPosition(Target);
20     }
21 }

```

Figure 29: Task Manager Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Events;
5  using UnityEngine.UI;
6
7  public class ButtonAssignEvents : MonoBehaviour
8  {
9      public Button ActionButton;
10     public Sprite ButtonSprite;
11     public UnityEvent Button_Event;
12
13     1 reference
14     public void AssignEvent()
15     {
16         if (!ActionButton)
17         {
18             ActionButton = UIManager.Instance.ActionButton;
19         }
20
21         if (ButtonSprite)
22             ActionButton.GetComponent<Image>().sprite = ButtonSprite;
23
24         ActionButton.onClick.RemoveAllListeners();
25         ActionButton.onClick.AddListener(EventInvoke);
26         ActionButton.gameObject.SetActive(true);

```

Figure 30: Button Assign Events Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  0 Unity Script | 0 references
7  public class LevelSelection : MonoBehaviour
8  {
9      public AudioSource Source;
10     public AudioClip ClickSound;
11     public Button[] Levels;
12
13     2 references
14     public int UnlockPref
15     {
16         get
17         {
18             return PlayerPrefs.GetInt("UnlockedLevels");
19         }
20         set
21         {
22             PlayerPrefs.SetInt("UnlockedLevels", value);
23         }
24     }
25     0 Unity Message | 0 references
26     private void Start()

```

Figure 31: Level Selection Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Events;
5
6  public class TriggerEvents : MonoBehaviour
7  {
8      public string[] Tags = new string[] { "Player" };
9
10     public UnityEvent On_Trigrgr_Enter, On_Trigger_Exit;
11
12     ButtonAssignEvents ButtonAssign;
13
14     @ Unity Message | 0 references
15     private void OnTriggerEnter(Collider other)
16     {
17
18
19
20         if (CheckTag(other))
21         {
22             On_Trigrgr_Enter.Invoke();
23             ButtonAssign = GetComponent<ButtonAssignEvents>();
24         }
25
26         if (ButtonAssign)

```

Figure 32: Trigger Events Script

```

3  using UnityEngine;
4  using UnityEngine.UI;
5  using DG.Tweening;
6
7  public class UIManager : MonoBehaviour
8  {
9      #region Singleton
10     private static UIManager _Instance;
11
12     4 references
13     public static UIManager Instance
14     {
15         get
16         {
17             _Instance = FindObjectOfType<UIManager>();
18             return _Instance;
19         }
20         set
21         {
22             _Instance = value;
23         }
24     }
25
26     @ Unity Message | 0 references
27     private void Awake()

```

Figure 33: UI Manager Script

The screenshot shows the Visual Studio IDE with the CameraFocus.cs script open. The script is a C# class that inherits from MonoBehaviour. It includes several using statements at the top: System.Collections, System.Collections.Generic, UnityEngine, Cinemachine, and UnityEngine.Events. The class has a public property FocusCam of type CinemachineVirtualCamera. It also has two public properties of type CinemachineBlendDefinition: virtualCamBlendStart and virtualCamBlendEnd. There are several public boolean and float properties: Off_Controls_At_Start, On_Controls_At_End, FocusTime, End_Manually, Focus_Start, Focus_End, Field_Of_View, and a Range attribute for FocusTime. The class also has two private fields: managerUi of type UiManager and Camera_Manager of type CameraManager. The Start method is partially visible at the bottom of the script.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Cinemachine;
5  using UnityEngine.Events;
6
7  public class CameraFocus : MonoBehaviour
8  {
9      public CinemachineVirtualCamera FocusCam;
10     [Header("<---Blend_Modes--->")]
11     public CinemachineBlendDefinition virtualCamBlendStart = new CinemachineBlendDefinition();
12     public CinemachineBlendDefinition virtualCamBlendEnd = new CinemachineBlendDefinition();
13
14     public bool Off_Controls_At_Start = false;
15     public bool On_Controls_At_End = false;
16     public float FocusTime;
17     public bool End_Manually;
18     public UnityEvent Focus_Start, Focus_End;
19     [Range(0, 100)]
20     public float Field_Of_View = 40;
21
22
23     UiManager managerUi;
24     CameraManager Camera_Manager;
25     void Start()
26     {

```

Figure 34: Camera Focus Script

The screenshot shows the Visual Studio IDE with the CameraInput.cs script open. The script is a C# class that inherits from MonoBehaviour. It includes several using statements at the top: System.Collections, System.Collections.Generic, UnityEngine, and Cinemachine. The class has a public property Camera of type CinemachineFreeLook. It has a private Start method that calls GetComponent<CinemachineFreeLook>(). It also has a private LateUpdate method that updates the Camera_m_XAxis_m_InputAxisValue and Camera_m_YAxis_m_InputAxisValue properties based on the mouse X and Y axis values.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Cinemachine;
5
6  public class CameraInput : MonoBehaviour
7  {
8      CinemachineFreeLook Camera;
9
10     private void Start()
11     {
12         Camera = GetComponent<CinemachineFreeLook>();
13     }
14
15     private void LateUpdate()
16     {
17         Camera_m_XAxis_m_InputAxisValue = ControlFreak2.CF2Input.GetAxis("Mouse X");
18         Camera_m_YAxis_m_InputAxisValue = ControlFreak2.CF2Input.GetAxis("Mouse Y");
19     }
20 }
21

```

Figure 35: Camera Input Script


```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Cinemachine;
5
6  public class CameraManager : MonoBehaviour
7  {
8
9      [Singleton]
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32     public CinemachineBrain MainCamera;
33     public GameObject RCC_Camera;
34
35
36     public void CamFocus(bool Status)
37     {
38         //PlayerCamera.gameObject.SetActive(!Status);
39         RCC_Camera.SetActive(!Status);
40         MainCamera.gameObject.SetActive(Status);
41     }
42
43
44
45
46

```

Figure 36: Camera Manager Script

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using SicknesscoreGames.HUDNavigationSystem;
5
6  public class PlayerManager : MonoBehaviour
7  {
8
9      [Singleton]
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32     public Player[] Players;
33     public GameObject CurrentPlayer;
34
35
36     public void SwitchPlayer(int index)
37     {
38         for (int i = 0; i < Players.Length; i++)
39         {
40             Players[i].PlayerObject.SetActive(false);
41             Players[i].PlayerCamera.SetActive(false);
42             Players[i].PlayerControls.SetActive(false);
43         }
44         OnPlayer(index);
45     }
46
47     void OnPlayer(int index)
48     {
49         Players[index].PlayerObject.SetActive(true);
50     }
51
52
53
54
55

```

Figure 37: Player Manager Script

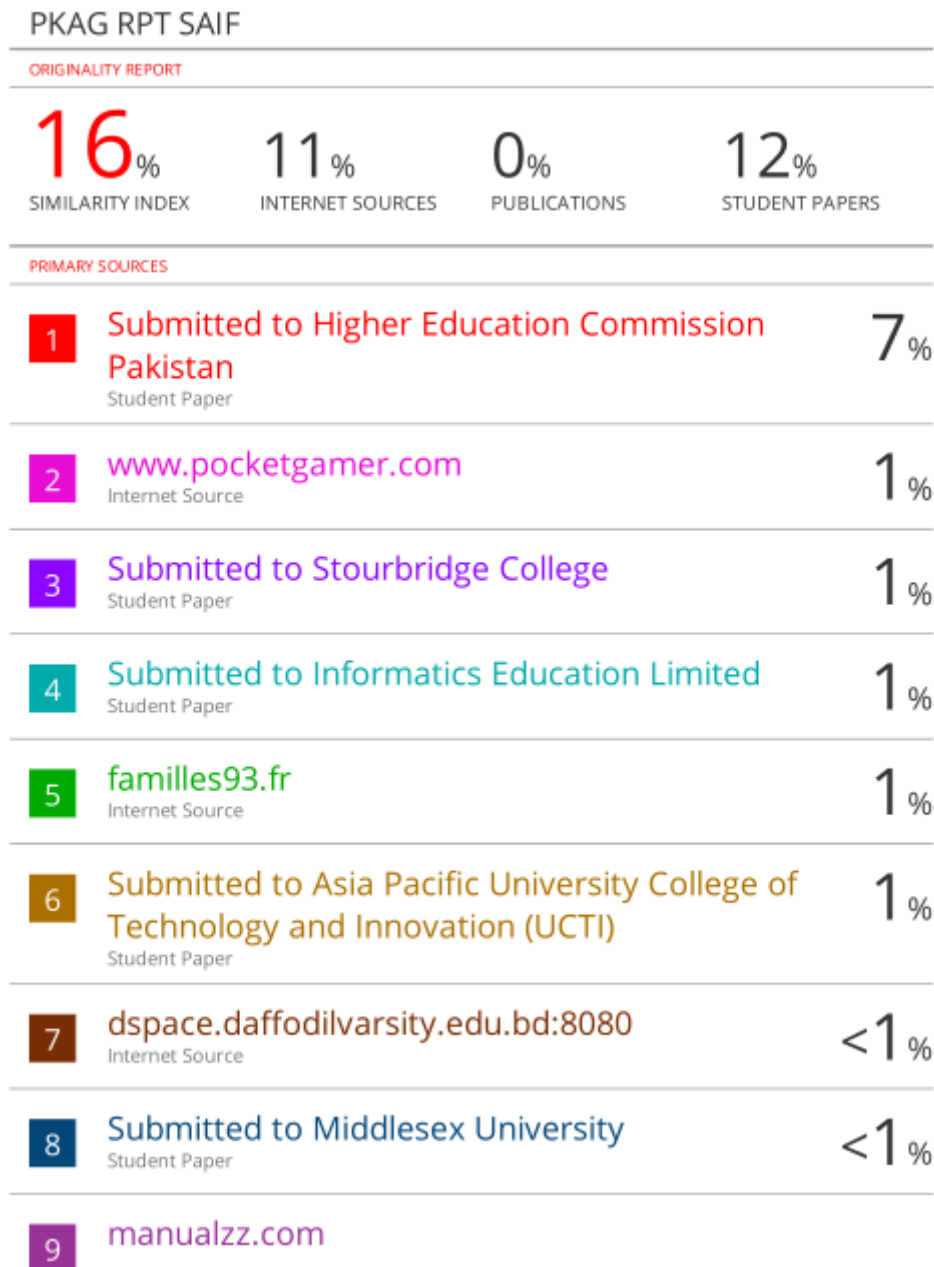


Figure 38: Plagiarism Report