Car Parking Simulator

**A Project Report**

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

**BACHELOR OF SCIENCE (COMPUTER SCIENCE)**

**By**

**Faiz Majaz Khan**

**(32)**

**Under the esteemed guidance of**

**Mrs. Pramitha Santhumayor**

**DEPARTMENT OF COMPUTER** **SCIENCE**

Royal College Of Arts, Science and Commerce

(Affiliated to University of Mumbai)

**MIRA ROAD 401107**

**MAHARASHTRA**

**2022 – 2023**

****

**DEPARTMENT OF COMPUTER SCIENCE**

Class: **TYBSc**  Roll. No. 32

Seat No.\_\_\_\_\_\_\_

**Certificate**

Certified that **Faiz Majaz Khan** of TY. BSc Semester-VI has successfully completed the project as prescribed by the University of Mumbai as partial fulfilment of requirement for completing Bachelor’s Degree in Computer Science during the academic year 2022-2023.

Signature of Project Guide

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of Examiner

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

H.O.D

Dept. of Computer Science

**Abstract**

It is a 3D Simulator parking game where the user has to pass through various checkpoints, obstacles and other cars and then park his car. It’s a complete strategy game with different levels. The main mission of the user is to park the car properly without Colliding with other obstacles present in the scenario. There are many levels with each level difficulty increases. The game is developed using Unity. Unity is a game engine which allows us to create models, games, infrastructures in 3 Dimensional environment using the required assets. Unity provides toolkit and the interface to create realistic 2D, 3D games, models, GUI, alongside audio and video configuration as well.

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help from many individuals and organizations. I would like to extend my sincere thanks to all of them. My deepest thanks to Mrs. Pramitha Santhumayor, Assistant Professor of the Computer Science Department, for guiding, supervising, providing me with necessary information about the project and correcting various prototypes and documents of mine with attention and care. She has gone with me through the project and without his help I was not able to complete my project. I also would like to express my gratitude towards my department Faculty who have given me a great chance to undergo this project and also provide me with knowledge to complete it. My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

**DECLARATION**

I hereby declare that the project entitled, “Car Parking Simulator” done at Royal College of Arts, Science and Commerce, Mira Road, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE) to be submitted as final semester project as part of our curriculum.

(Faiz Majaz Khan)

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Contents** | |
|  | Abstract | |
| **Chapter 1** | Introduction | |
|  | 1.1 | Introduction |
|  | 1.2 | Proposed System |
| **Chapter 2** | 2.1 | System Requirements |
|  | 2.2 | Feasibility Study |
|  | 2.3 | Gantt Chart |
| **Chapter 3** | System Analysis | |
|  | 3.1 | Use Case Diagram |
|  | 3.2 | E-R Diagram |
|  | 3.3 | Activity Diagram |
|  | 3.4 | Class Diagram |
|  | 3.5 | Sequence Diagram |
| **Chapter 4** | 4.1 | System Coding |
|  | 4.2 | Form layouts |
|  | 4.3 | Test Cases |
| **Chapter 5** | 5.1 | Conclusion |
|  | 5.2 | Future Enhancements |
| **Chapter 6** | Reference and Bibliography | |

**Chapter 1**

**1.1 Introduction**

This project presents a car parking simulator game, in which the human player parks a car properly in a three-dimensional environment. The objective of the game is to safely park the car and to provide the player with a challenging and enjoyable experience. The game Consists of many levels with each level a challenging one. The game has been created in a 3D environment where the user feels he’s in a realistic environment. The game starts with a car in a parking lot, you have to follow the right path through checkpoints and park in correct manner without colliding with other cars, obstacles and poles. The game is created in Unity with all the corresponding libraries of Unity3D.

**1.2 Proposed System Advantages**

1. Exploration: Simulator games allow you to explore the real-world environment giving the user an amazing experience through a game.
2. Skill development: Simulator games can help players develop their culinary skills and knowledge by exposing them to new real-life methods.
3. Problem-solving: Simulator Parking games require players to think critically and solve problems by choosing the correct speed, turn, when to accelerate and when to apply brakes for proper driving and adapting to changing game conditions
4. Entertainment: Simulator games provide an entertaining and engaging way for players to learn about how things work in real life scenario.
5. Accessibility: Simulator games are widely available on various platforms, including mobile devices and gaming consoles, making them easily accessible to a large audience.
6. Revenue generation: Simulator Parking games can generate revenue through in-game advertising and microtransactions, allowing game developers to continue improving and expanding the game over time.

**Chapter 2**

**2.1 System Requirements**

**2.1.1 Software Requirements**

|  |  |
| --- | --- |
| **Operating systems** | Windows/ Linux / Mac, Android compatible. |
| **Game Engine** | Unity |
| **Languages** | C# |
| **IDE** | Visual Studio |

**Table 1.1 Software Requirements**

|  |  |
| --- | --- |
| **CPU** | 5th generation Intel Core or newer |
| **Memory (RAM)** | 4 GB RAM or more |
| **Hard disk space** | 4 GB of available disk space minimum |
| **Screen** | 1280 x 800 minimum screen resolution |

**2.1.2 Hardware Requirements:**

**Table 1.2 Hardware Requirements**

**2.2 Feasibility Study**

**2.2.1. Technical Feasibility**

Building this system is technically feasible. The hardware and software needed are all available, it is not difficult to get them. In brief I can say the necessary resources needed for the development and maintenance of the system are available in Unity.

**2.2.2 Operationally Feasibility**

The project I am developing is operationally feasible as there is no need for users to have good knowledge of computers before using them. The user can learn and use the system with ease, he just needs to read the manual or tutorial from the developers.

**2.2.3. Economic Feasibility**

Besides being technically feasible, developing this system is economically feasible as well. The development of the system does not require the developers to spend a lot of money. The tools I will be using to develop the system are not expensive and the software’s open source. All I need is time. Even the maintenance of the system will not be expensive. The system is indeed economically feasible.

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase Title** | **Expected Date of Completion** | **Actual Time of Completion with Guide’s Signature** | **Remarks** |
| **I. Preliminary Investigation** |  |  |  |
| (i) Organizational Overview | 15/7/22 |  |  |
| (ii) Present System and its advantages |  |  |  |
| (iii) System Requirements |  |  |  |
| (iv) Feasibility Study |  |  |  |
| (vi) Phase Title | 29/7/22 |  |  |
| (vii) Gantt Chart |  |  |  |
| **II. System Analysis** |  |  |  |
| (i) Use Case |  |  |  |
| (ii) ER diagram | 1 2/8/22 |  |  |
| (iii) Activity Diagram |  |  |  |
| (iv) Class Diagram |  |  |  |
| (v) Sequence Diagram |  |  |  |
| (vi)Sequence Diagram |  |  |  |
| **IV. System Coding** |  |  |  |
| (i) System Coding |  |  |  |
| (ii) Form Layouts |  |  |  |
| (iii) Test Cases |  |  |  |
| **V. Future Enhancements** |  |  |  |
| **VI. Reference and Bibliography** | 24/9/22 |  |  |

**Phase Title**

TYBSC Computer Science Semester 6

2022-2023

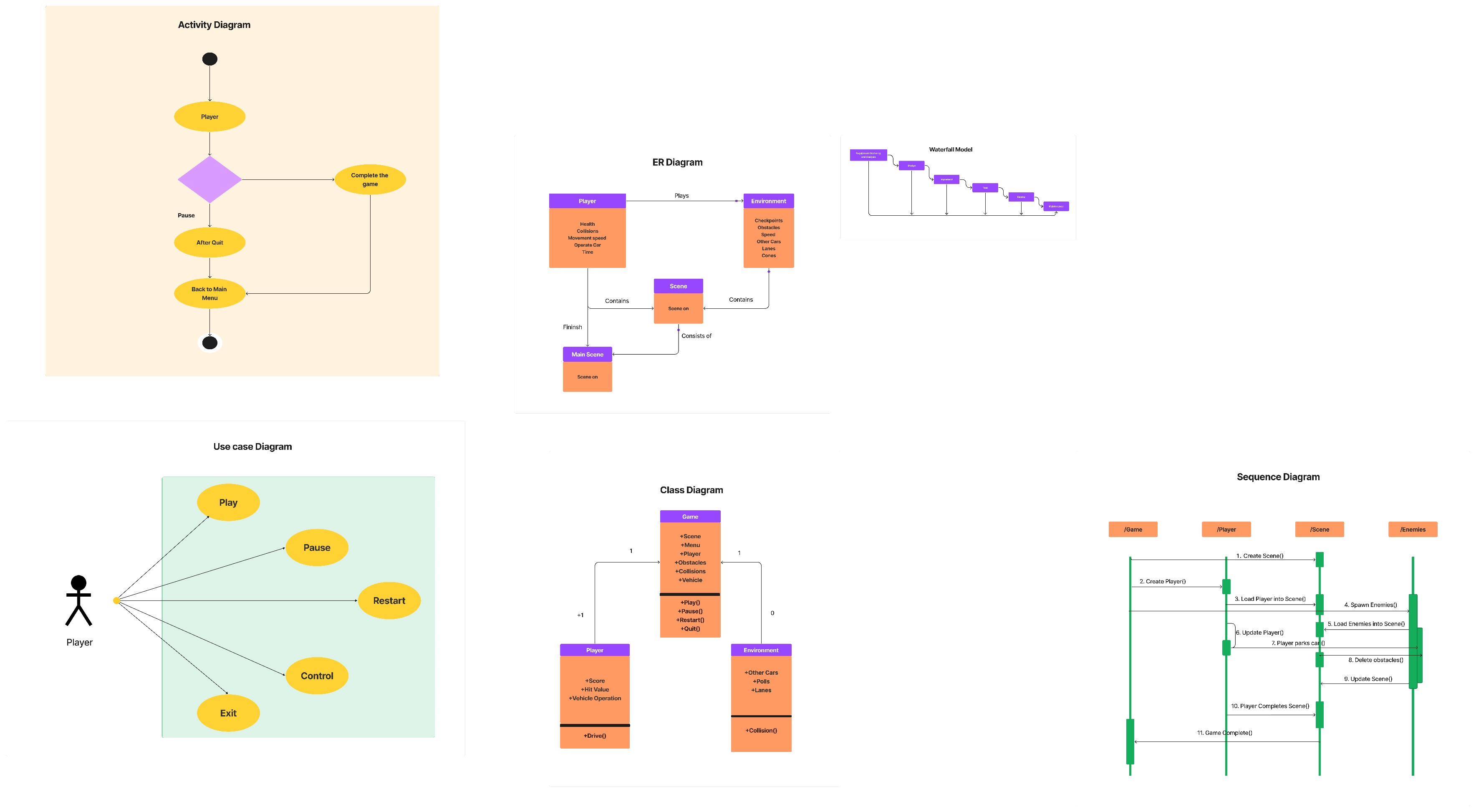
**2.3 Gantt Chart:**

**Gantt chart**

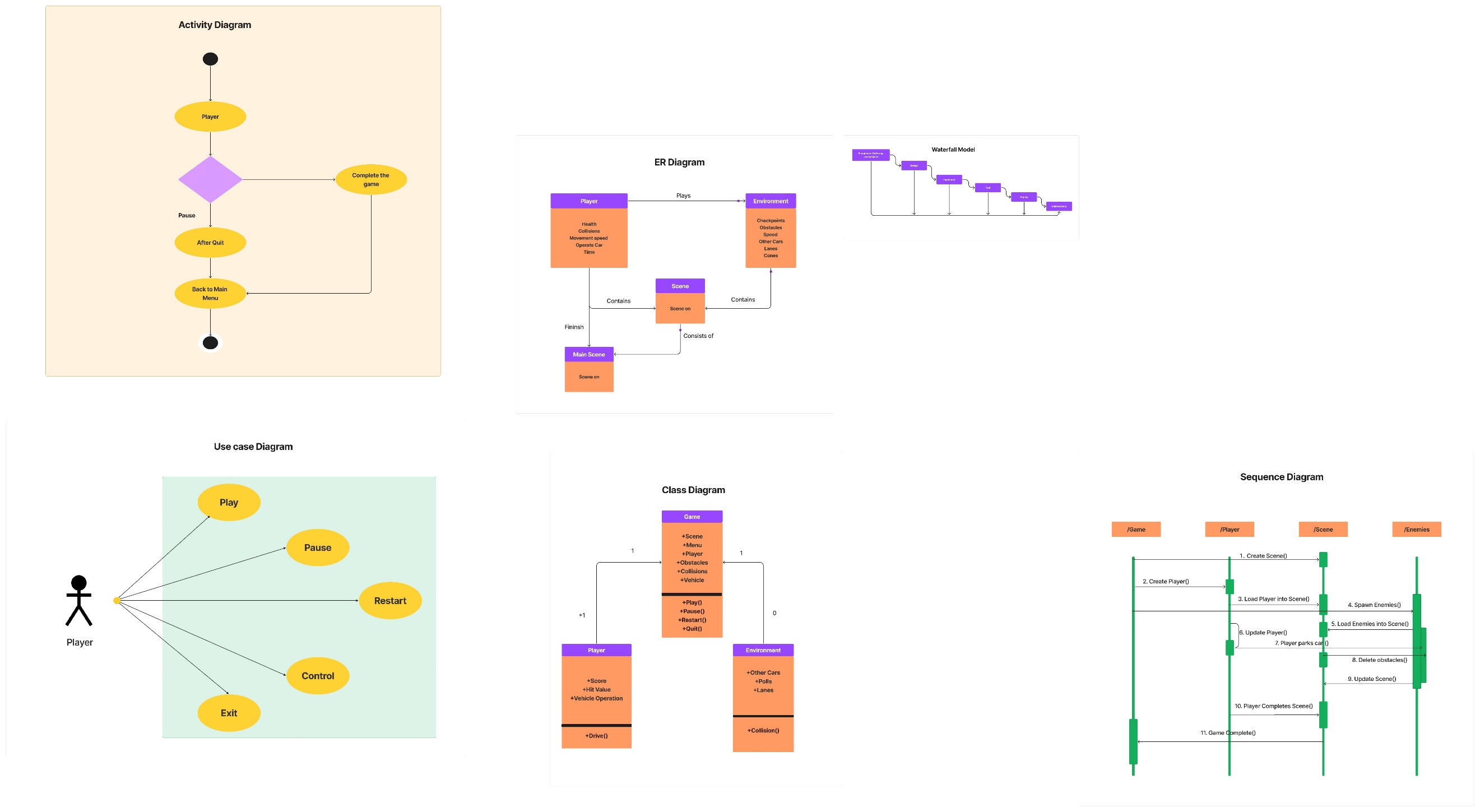
**Chapter 3**

**3.1 System Analysis**

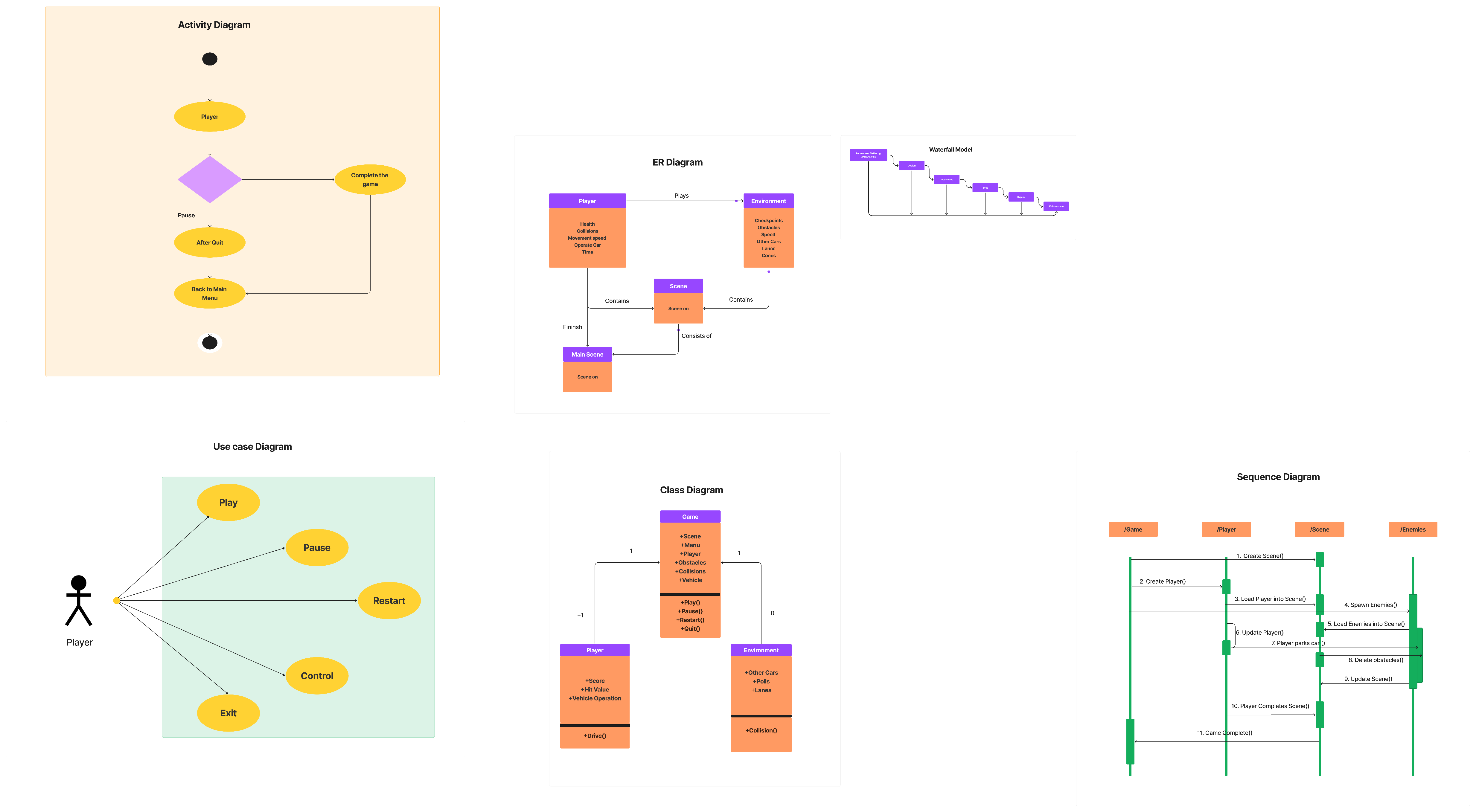
**Use Case Diagram**

****

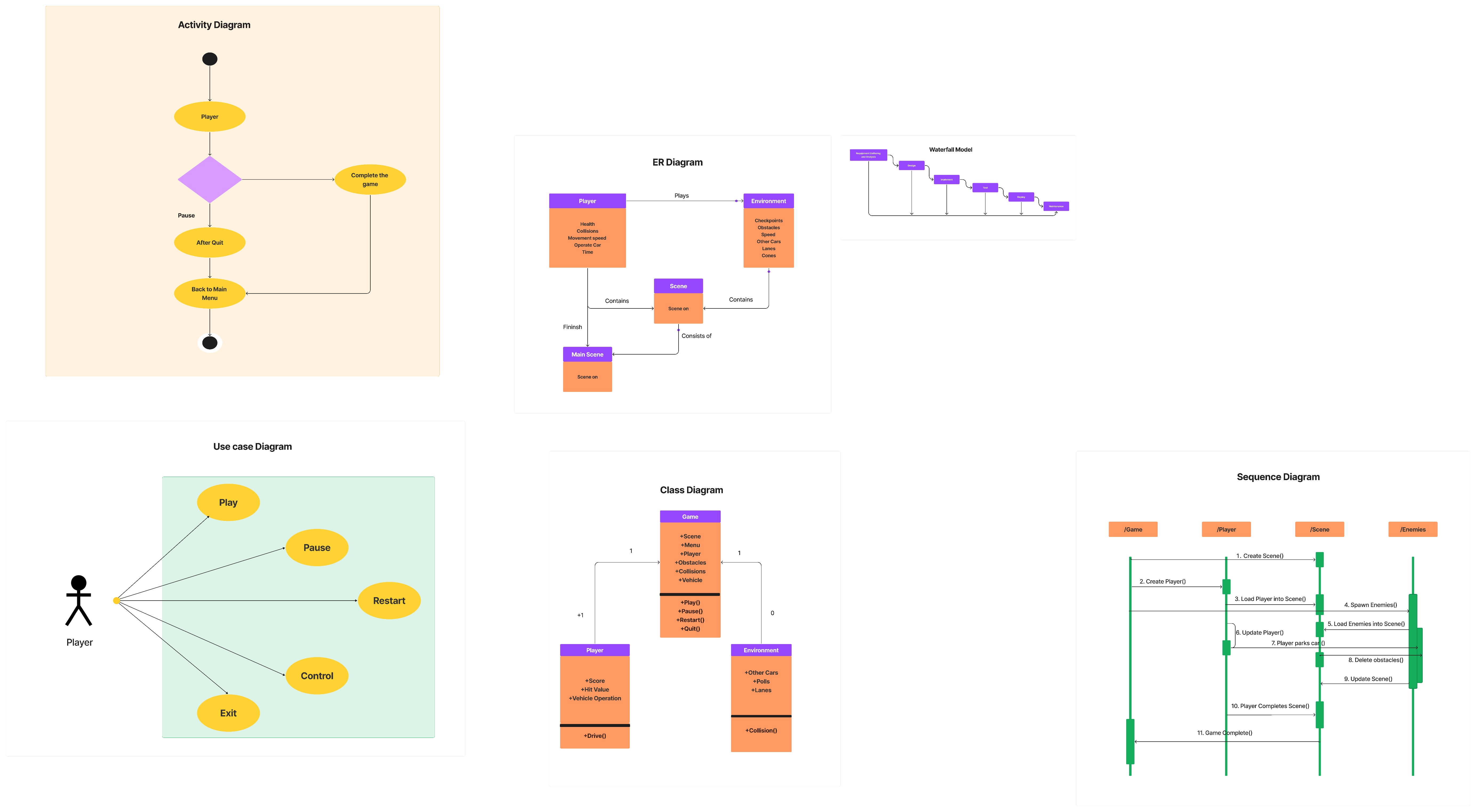
**ER Diagram**

****

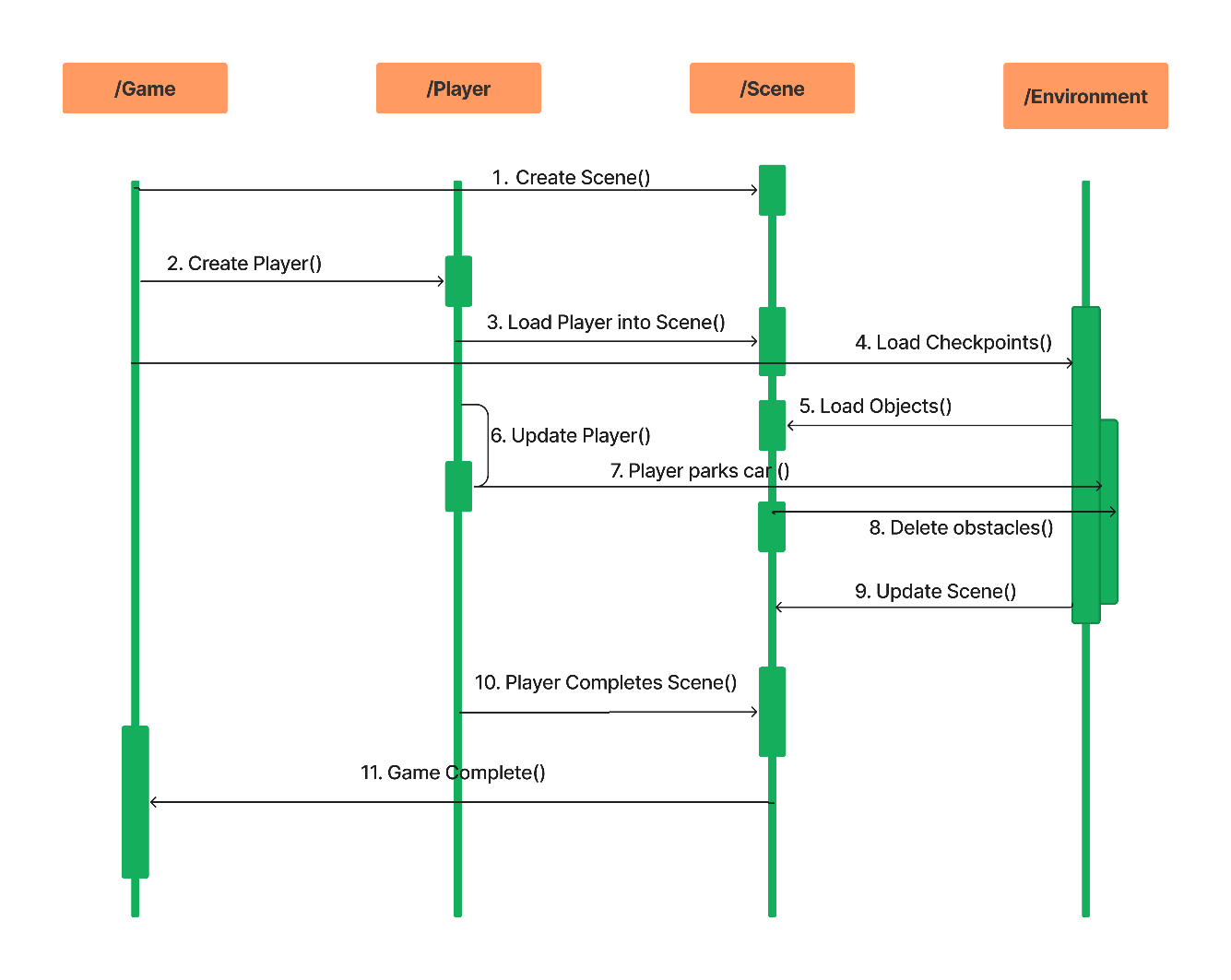
**Activity Diagram**

****

**Class Diagram**

****

**Sequence Diagram**

****

**Chapter 4**

**4.1 System Coding**

using UnityEngine;

using System.Collections;

public enum InputType

{

Keyboard,

Mobile

}

public class VehicleController2017 : MonoBehaviour {

public bool canControll;

[Header("Wheels")]

public WheelCollider[] Wheel\_Colliders;

public Transform[] Wheel\_Transforms;

public Transform steeringWheel;

// public Transform centerOfMass;

float currentSpeed;

[Header("Vehicle Setup")]

public float enginePower = 1400f ;

public float brakePower = 1400f;

public float[] gearsPower;

public Vector3 COM;

public float maxSteer = 43f;

public float maxSpeed = 74f;

// Input values

float throttleInput;

float steerInput;

// Also used in SmoothFollow component

[HideInInspector]public bool handBrake;

// Used for detecting reverse mode (if localVel.z <0 => reversing, if localVel.z>0 => is not reversing)

Vector3 velocity;

Vector3 localVel;

[HideInInspector]public bool reversing;

// Catch rigidbody

Rigidbody rigid;

// Gear values to control engine sound based on gears

public int numberOfGears = 10;

int currentGear ;

float GearFactor;

[HideInInspector]public float Revs;

public float GearShiftDelay = 0.3f;

VehicleAudio2017 audioCar;

[Header("Lights")]

// Vehicle lights

public Light[] brakeLights;

public Light[] reverseLights;

public Material lightMaterial;

void Start()

{

StartCoroutine (GearChanging ());

// I see this in unity car sample script, i don't khow what is do this

Wheel\_Colliders [0].attachedRigidbody.centerOfMass = new Vector3(0,0,0);

rigid = GetComponent<Rigidbody> ();

// used to smoothing smooth follow camera movement behind vehicle

rigid.interpolation = RigidbodyInterpolation.Interpolate;

// Set center of mass to center of mass transform localposition

rigid.centerOfMass = COM;

audioCar = GetComponent<VehicleAudio2017> ();

}

void Update ()

{

VehicleEngine ();

// Update current speed and multiply to 3 for better understand

currentSpeed = rigid.velocity.magnitude \* 2.23693629f;

// Find vehicle reversing state

velocity = rigid.velocity;

localVel = transform.InverseTransformDirection(velocity);

if (localVel.z < 0)

reversing = true;

else

reversing = false;

// Align wheel mesh across wheel collider rotation and position

for (int i = 0; i < Wheel\_Colliders.Length; i++)

{

Quaternion quat;

Vector3 position;

Wheel\_Colliders [i].GetWorldPose (out position, out quat);

Wheel\_Transforms [i].transform.position = position;

Wheel\_Transforms [i].transform.rotation = quat;

}

if (steeringWheel)

steeringWheel.rotation =

transform.rotation \* Quaternion.Euler (34f, 0,(Wheel\_Colliders[0].steerAngle ) \*-1f );

}

public void VehicleEngine()

{

CalculateRevs ();

if (canControll)

{

if (currentSpeed >= maxSpeed)

rigid.drag = 0.3f;

else

rigid.drag = 0.1f;

Wheel\_Colliders [2].motorTorque = enginePower\*throttleInput\*gearsPower[currentGear];

Wheel\_Colliders [3].motorTorque = enginePower\*throttleInput\*gearsPower[currentGear];

Wheel\_Colliders [2].motorTorque = Mathf.Clamp (Wheel\_Colliders [2].motorTorque, -enginePower/2, enginePower);

Wheel\_Colliders [3].motorTorque = Mathf.Clamp (Wheel\_Colliders [3].motorTorque, -enginePower/2, enginePower);

Wheel\_Colliders [0].steerAngle = maxSteer \* steerInput;

Wheel\_Colliders [1].steerAngle = maxSteer \* steerInput;

Wheel\_Colliders [1].steerAngle = Mathf.Clamp (Wheel\_Colliders [1].steerAngle, - (maxSteer/(currentSpeed/10)), (maxSteer/(currentSpeed/10)));

Wheel\_Colliders [0].steerAngle = Mathf.Clamp (Wheel\_Colliders [0].steerAngle, - (maxSteer/(currentSpeed/10)), (maxSteer/(currentSpeed/10)));

if (handBrake)

{ // Hand brake state

Wheel\_Colliders [2].brakeTorque = brakePower;

Wheel\_Colliders [3].brakeTorque = brakePower;

LightIntensity (0, 1f);

lightMaterial.SetFloat ("\_Intensity", 0.1f);

LightIntensity (1, 0);

//Debug.Log ("1");

}

else

{

// Speed decreasing when motor input value is 0

if (throttleInput <= 0.07 && throttleInput >= -0.07f)

{//Debug.Log ("2");

Wheel\_Colliders [2].brakeTorque = brakePower / 5;

Wheel\_Colliders [3].brakeTorque = brakePower / 5;

Wheel\_Colliders [0].brakeTorque = brakePower / 5;

Wheel\_Colliders [1].brakeTorque = brakePower / 5;

LightIntensity (0, 0);

lightMaterial.SetFloat ("\_Intensity", 0);

LightIntensity (1, 0);

}

// Brake in forward moving

else if (throttleInput < 0 && !reversing)

{//Debug.Log ("3");

Wheel\_Colliders [0].brakeTorque = brakePower \* Mathf.Abs (throttleInput);

Wheel\_Colliders [1].brakeTorque = brakePower \* Mathf.Abs (throttleInput);

Wheel\_Colliders [2].brakeTorque = brakePower \* Mathf.Abs (throttleInput / 2);

Wheel\_Colliders [3].brakeTorque = brakePower \* Mathf.Abs (throttleInput / 2);

LightIntensity (0, 1f);

lightMaterial.SetFloat ("\_Intensity", 0.1f);

LightIntensity (1, 0);

}

// Brake in backward moving

else if (throttleInput > 0 && reversing)

{//Debug.Log ("4");

Wheel\_Colliders [0].brakeTorque = brakePower \* Mathf.Abs (throttleInput);

Wheel\_Colliders [1].brakeTorque = brakePower \* Mathf.Abs (throttleInput);

Wheel\_Colliders [2].brakeTorque = brakePower \* Mathf.Abs (throttleInput / 2);

Wheel\_Colliders [3].brakeTorque = brakePower \* Mathf.Abs (throttleInput / 2);

LightIntensity (0, 1f);

lightMaterial.SetFloat ("\_Intensity", 0.1f);

LightIntensity (1, 0);

}

// Release brake ( is now driving)

else {//Debug.Log ("5");

Wheel\_Colliders [2].brakeTorque = 0;

Wheel\_Colliders [3].brakeTorque = 0;

Wheel\_Colliders [0].brakeTorque = 0;

Wheel\_Colliders [1].brakeTorque = 0;

LightIntensity (0, 0);

lightMaterial.SetFloat ("\_Intensity", 0);

LightIntensity (1, 0);

}

}

if (reversing && throttleInput < 0) {

LightIntensity (0, 0);

lightMaterial.SetFloat ("\_Intensity", 0);

LightIntensity (1, 1f);

}

}

}

bool isMobile;

public void Move(float motor,float steer,bool hand)

{

throttleInput = motor;

steerInput = steer;

handBrake = hand;

}

void LightIntensity(int type,float value)

{

if (type == 0) {

for (int a = 0; a < brakeLights.Length; a++)

brakeLights [a].intensity = value;

} else {

for (int a = 0; a < reverseLights.Length; a++)

reverseLights [a].intensity = value;

}

}

// Engine sound system calculation

// Gear changing only used for sound system

IEnumerator GearChanging ()

{

while (true)

{

yield return new WaitForSeconds (0.01f);

if (!reversing) {

float f = Mathf.Abs (currentSpeed / nextGearSpeed);

float upgearlimit = (1 / (float)numberOfGears) \* (currentGear + 1);

float downgearlimit = (1 / (float)numberOfGears) \* currentGear;

// Changinbg gear down

if (currentGear > 0 && f < downgearlimit) {

// Reduce engine audio volume when changing gear

audioCar.audioSource.volume = 0.7f;

//audioCar.ChangeGear ();

// Delay time for changing gear down

yield return new WaitForSeconds (0);

audioCar.audioSource.volume = 1f;

currentGear--;

}

// Changing gear Up

if (f > upgearlimit && (currentGear < (numberOfGears - 1))) {

// Reduce engine audio volume when changing gear

audioCar.audioSource.volume = 0.3f;

//audioCar.ChangeGear ();

// Delay before changing gear up

yield return new WaitForSeconds (GearShiftDelay);

audioCar.audioSource.volume = 1f;

currentGear++;

}

} else {

if (reversing)

currentGear = 0;

}

}

}

// simple function to add a curved bias towards 1 for a value in the 0-1 range

private static float CurveFactor (float factor)

{

return 1 - (1 - factor) \* (1 - factor);

}

// unclamped version of Lerp, to allow value to exceed the from-to range

private static float ULerp (float from, float to, float value)

{

return (1.0f - value) \* from + value \* to;

}

public float nextGearSpeed = 300f;

// Used for engine sound system

private void CalculateGearFactor ()

{

float f = (1 / (float)numberOfGears);

// gear factor is a normalised representation of the current speed within the current gear's range of speeds.

// We smooth towards the 'target' gear factor, so that revs don't instantly snap up or down when changing gear.

var targetGearFactor = Mathf.InverseLerp (f \* currentGear, f \* (currentGear + 1), Mathf.Abs (currentSpeed / nextGearSpeed));

GearFactor = Mathf.Lerp (GearFactor, targetGearFactor, Time.deltaTime \* 5f);

}

// Used for engine sound system

private void CalculateRevs ()

{

// calculate engine revs (for display / sound)

// (this is done in retrospect - revs are not used in force/power calculations)

CalculateGearFactor ();

var gearNumFactor = currentGear / (float)numberOfGears;

var revsRangeMin = ULerp (0f, 1f, CurveFactor (gearNumFactor));

var revsRangeMax = ULerp (1f, 1f, gearNumFactor);

Revs = ULerp (revsRangeMin, revsRangeMax, GearFactor);

}

}

using UnityEngine;

using System.Collections;

public class InputSystem : MonoBehaviour

{

// Select control type => Touch or keyboard

public InputType controllType;

VehicleController2017 controller;

// Automatically detect target platform input type (mobile keyboard)

public bool autoSwitch;

float motorInput,steerInput;

bool handBrake;

bool reversing;

[Header("Visuals")]

public UnityEngine.UI.Image reverseImage;

public Sprite reverseOffSprite,reverseOnSprite;

[Header("Components")]

public SteeringWheel steeringWheel;

bool sWheelControl;

public GameObject sWheel;

public GameObject arrowKeys;

// Accelerometer controlling

[Header("Accelerometer")]

public float accelSensibility = 10f;

public float accelSmooth = 0.5f;

Vector3 curAc;

float GetAxisH = 0f;

bool accelInput;

VehicleHorn hornComponent;

CameraManager camManager;

IEnumerator Start ()

{

if (autoSwitch) {

#if UNITY\_EDITOR || UNITY\_WEBGL

controllType = InputType.Keyboard;

GetComponent<PauseMen>().mobileControlls.SetActive (false);

#else

controllType = InputType.Mobile;

#endif

}

hornComponent = GetComponent < VehicleHorn > ();

camManager = GetComponent<CameraManager> ();

if (PlayerPrefs.GetInt ("ControlType") == 0) {

sWheel.SetActive (true);

arrowKeys.SetActive (false);

sWheelControl = true;

}

if (PlayerPrefs.GetInt ("ControlType") == 1) {

sWheel.SetActive (false);

arrowKeys.SetActive (true);

}

if (PlayerPrefs.GetInt ("ControlType") == 3) {

sWheel.SetActive (false);

arrowKeys.SetActive (false);

accelInput = true;

}

yield return new WaitForEndOfFrame ();

controller = GameObject.FindObjectOfType<VehicleController2017> ();

}

void Update ()

{

if (!controller)

return;

if (accelInput)

{

if(Input.acceleration.x > 0.2f || Input.acceleration.x <-0.2f)

steerInput = Input.acceleration.x;

}

if (sWheelControl)

steerInput = steeringWheel.GetClampedValue ();

if (controllType == InputType.Keyboard)

{

if (Input.GetAxis ("Vertical") > 0) {

if (reversing)

motorInput = -1f;

else

motorInput = 1f;

}

else

motorInput = 0;

if(controllType == InputType.Keyboard)

steerInput = Input.GetAxis ("Horizontal");

if (Input.GetKey (KeyCode.Space) || Input.GetAxis("Vertical") < 0)

handBrake = true;

else

handBrake = false;

if (Input.GetKeyDown (KeyCode.R)) {

reversing = !reversing;

if (reverseImage)

{

if (reversing)

reverseImage.sprite = reverseOnSprite;

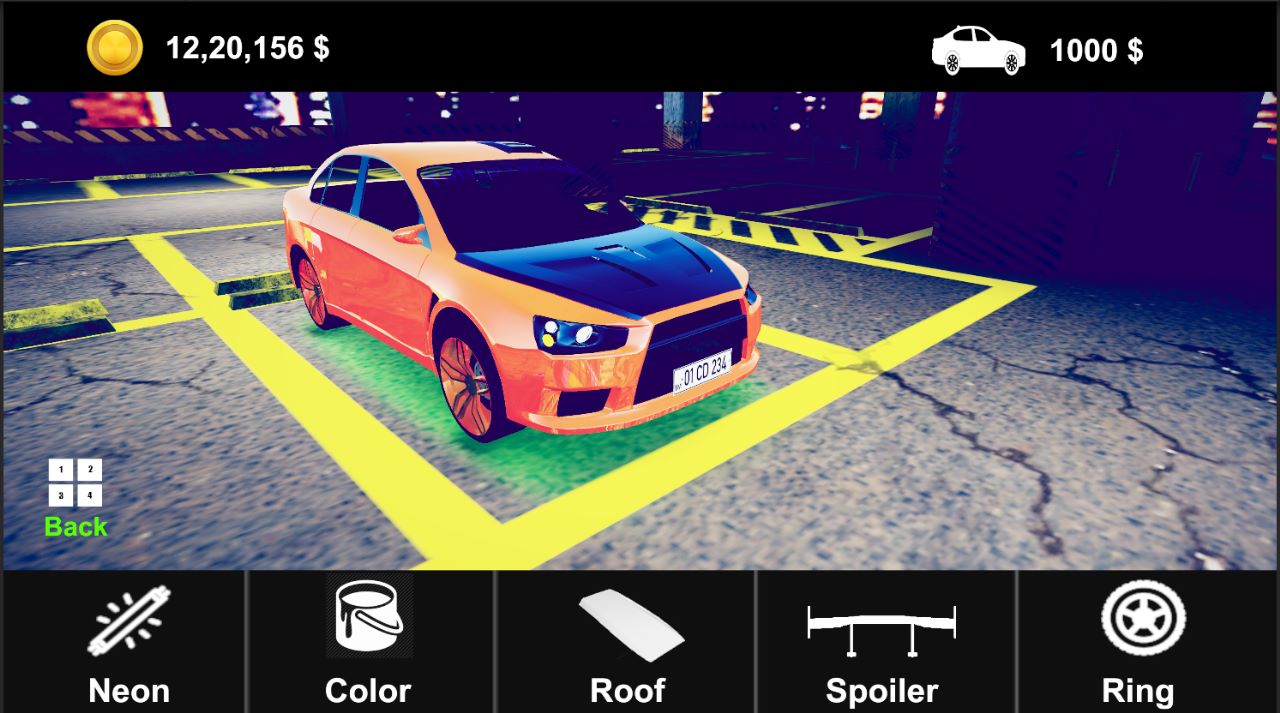
else

reverseImage.sprite = reverseOffSprite;

}

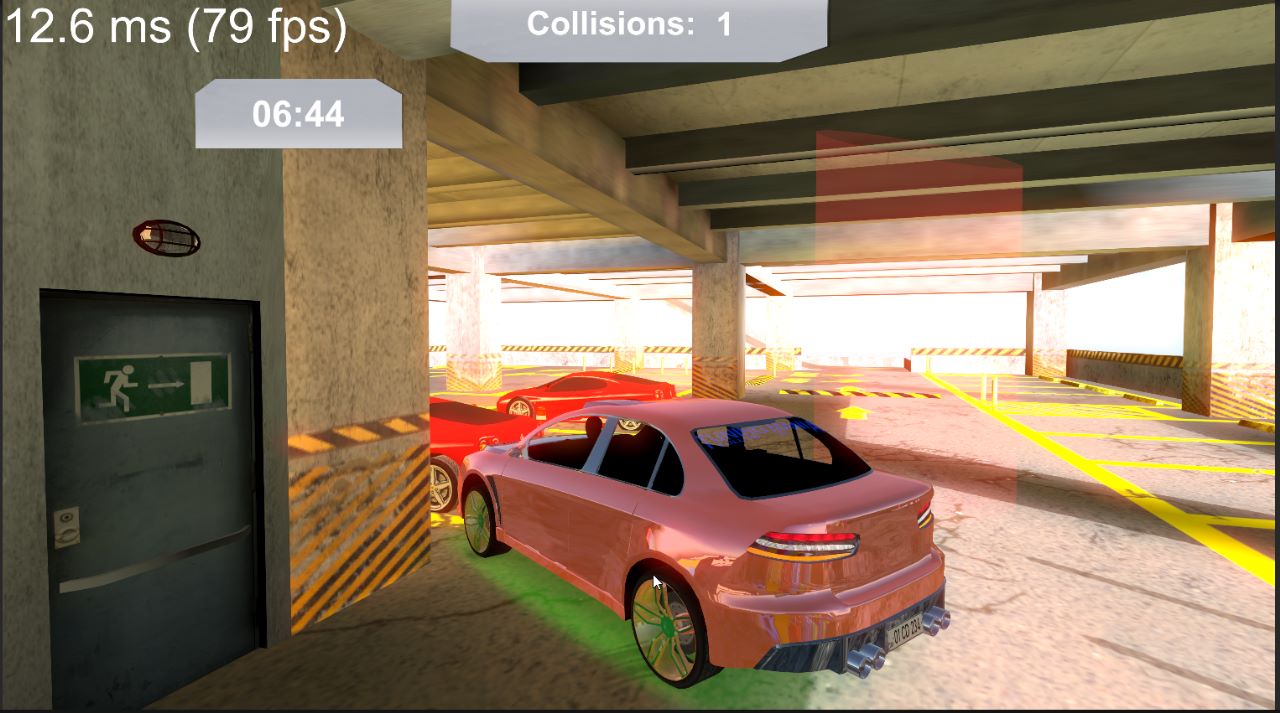
}

**4.2 Form Layouts**

****

****

**\**

****

****

**4.3 Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| Basic Movement | - The car moves smoothly in all directions | The car moves smoothly | Pass |
|  | - The car turns smoothly without jerking or sliding | The car turns smoothly without jerking or sliding | Pass |
|  | - The car's speed can be adjusted according to the player's needs | Car’s speed cannot be adjusted according to the player | Fail |
| Parking Challenges | - The parking lot has various parking spots of different sizes | The parking lot has various parking spots of different sizes | Pass |
|  | - Obstacles like barriers and parked cars are present | Obstacles like cars, cones, barriers, walls are present | Pass |
|  | - The parking spot is clearly marked and visible from a distance | User has to find the parking spot through checkpoints | Pass |
| Physics Simulation | - The car's movements are physically accurate | Car moves accurately | Pass |
|  | - The car behaves realistically when colliding with obstacles | Car crash happens, if collision is happened | Pass |
|  | The car responds to changes in the environment | car responds to changes in the environment | Pass |
| User Interface | - The player can easily navigate menus and settings | The player can easily navigate menus and settings | Pass |
|  | - The instructions and tutorials are clear and understandable | Game is user-friendly | Pass |
|  | - The player can save progress and resume the game later | Player can save his progress | Pass |
| Sound and Visuals | - The graphics are smooth and realistic | Graphics are realistic | Pass |
|  | - The sound effects are appropriate and realistic | User can hear engine, collisions, and horn sound | Pass |

**Chapter 5**

**5.1 Conclusion**

The objective of this project was to create a game for Fun and Learning purpose. With stunning graphics and realistic physics, this game will provide you with a thrilling and immersive experience that will keep you coming back for more. The game can be played on both Android and Windows. This required I had a fundamental knowledge about the Unity game engine and programming. A game engine is the core of creating a game. The integration of model design, level design and script design are the game engine, which is complex and powerful. The Unity game engine supports visualized design; thus, it is a strong game engine which is suitable for a beginner.

**5.2 Future Enhancements**

* Level Expansion
* Improve GUI
* Introduce more Features
* Introduce new Environment

**Chapter 6**

**Reference and Bibliography**

* + youtube.com
  + ijcaonline.org
  + mixamo.com
  + assetstore.unity.com
  + parallexsite.webflow.io
  + figma.com
  + Project report of “Nadial IIT” https:// [www.slideshare.net/NadiaIIT/](http://www.slideshare.net/NadiaIIT/)
  + www.unity3D.com