ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report

CENG 407

Innovative System Design and Development I

Car Racing Game using AI to Control Competitor Cars

Ataberk Baran Sütcü 201711060

Hasan Can Bayram 201711005

Umutcan Dansık 201711018

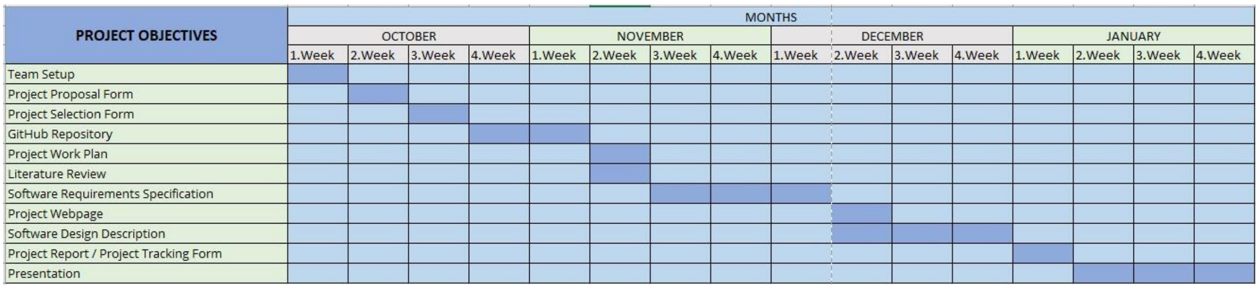
Advisor: Dr. Instructor Serdar ARSLAN

CONTENTS

1.Introduction

A racing game is a video game genre in which the player engages in a battle in which land, air, or watercraft compete in a first-person or third-person perspective.The subject we work on in our project is the car racing game. Car racing games are developed day by day nowadays. The main reasons for its development are the use of artificial intelligence and the graphics getting closer to reality day by day. The main purpose of our project is to use artificial intelligence effectively in a car racing game and to increase the enjoyment of the people who play the game. Detailed description of our project is included in this document.

2.Project Work Plan



PROJECT OBJECTIVES	MONTHS															
	OCTOBER				NOVEMBER				DECEMBER				JANUARY			
	1.Week	2.Week	3.Week	4.Week	1.Week	2.Week	3.Week	4.Week	1.Week	2.Week	3.Week	4.Week	1.Week	2.Week	3.Week	4.Week
Team Setup	■															
Project Proposal Form		■														
Project Selection Form			■													
GitHub Repository				■	■											
Project Work Plan						■										
Literature Review						■										
Software Requirements Specification							■	■	■							
Project Webpage										■						
Software Design Description										■	■	■				
Project Report / Project Tracking Form													■			
Presentation														■	■	■

3.Literature Review

3.1.Özet

Oyun geliştirme alanı büyüdükçe oyun severlerin de beklentileri arttı. Bu beklentiler sonucunda oyun geliştirenler kullanıcıların beklentilerini karşılamak için oyunlarda değişik teknikler kullanmaya başladılar. Tek oyunculu oyunlarda kullanıcıların oyundan aldıkları keyfin artması için oyun içinde bulunan rakiplerinin güçlendirilmesi yoluna başvurdular. Bu onlara, oynadıkları oyunu yapay zekaya karşı değil de gerçek bir oyuncuya karşı oynuyormuş gibi hissetmelerini sağladı. Oyun geliştiriciler oyunlarını hazırlarken yapay zekayı etkili bir biçimde kullanmaya başladılar. Bu yazıda, yapay zekanın bir araba yarışı oyununda nasıl çalıştığını değerlendirmek için geniş bilgisayar bilimi literatürünü araştırıyoruz. Bu konuyla alakalı makalelerden alıntılar yaparak konuyu detaylı bir şekilde inceleyeceğiz.

3.2.Abstract

As the game development field has grown, the expectations of gamers have also increased. As a result of these expectations, game developers started to use different techniques in games to meet the expectations of the users. In single-player games, in order to increase the enjoyment of the users from the game, they resorted to strengthening the opponents in the game. This made them feel like they were playing the game against a real player, not an artificial intelligence. Game developers have started to use artificial intelligence effectively while preparing their games. In this article, we search the vast computer science literature to evaluate how artificial intelligence works in a car racing game. We will examine the subject in detail by quoting from articles related to this subject.

3.3.Introduction

The love of playing games, which started from childhood, has become an addiction of young people, adults and even the elderly. Nowadays a simple game usually presents a situation where “players engage in an artificial conflict, defined by rules and results in a quantifiable outcome” [1]. Artificial conflicts used within games often require players to achieve certain achievements to earn points. For example collecting something, overcoming an obstacle, passing a vehicle etc... By playing such games, the younger generation has gained competencies such as easily obtaining information at a younger age and using this information effectively. Thanks to these skills, they tend to easily solve the problems they encounter in daily life.

The artificial intelligence (AI) community has witnessed a similar transition from the “classical AI games” such as Samuel’s Checker Player and Waterman’s Poker Player to the contemporary AI techniques adopted in electronic games [2]. As a result, the game developers improved the games and elaborated the point collection a little more. Different features have been added to the games, the environments have changed and a fluent progression method has been applied to the games. As the game progressed, dynamic developments began to be presented to the players. The developers have made these changes to increase the pleasure of the players they take from the game, and these changes has offered different experiences to the players.

The game WASD is an up-to-date car racing game where you compete against artificial intelligence on different maps offered to you, with moderate difficulty, giving you fun moments. The main goal of the player is to reach the finish before his opponents(AI). While striving for this purpose, different models on the map will meet him/her.

3.4.Literature Review

Research has been done to bring the performance of artificial intelligence to the level that it behaves like a human. Artificial intelligence in games is often called bots. Developing bots, that is, increasing the performance of artificial intelligence, causes a user to feel more happy while playing games and to think of himself as playing against a real user in the face of the power of artificial intelligence. In fact, users who play against powered artificial intelligence constantly express how hard it is to beat the bot against them. This powered bot usually represents artificial intelligence used at the end of games or episodes. To give an example of a game with strong artificial intelligence, it is FIFA [3], one of the most popular sports games where the objective is to score goals by passing all the players in the opposing team. To give an example of a simpler bot system, PacMan's artificial intelligence controlled ghost enemies can be given as an example. Ghost enemies can come towards the user because improved artificial intelligence algorithms are used.

There are various artificial intelligence algorithms used in games. These are decision trees, finite state machine, and fuzzy state machines. To give an example of games that use the finite state machine algorithm, the ghosts in PacMan can be an example of this algorithm. A finite-state machine is a model used to represent and control execution flow [5]. It is perfect for implementing AI in games, producing great results without a complex code [5]. This tutorial describes the theory, implementation and use of simple and stack-based finite-state machines [5].

To give different examples, in car racing games, users constantly make different maneuvers while driving to get their car to the finish line in minimum time. That is why, bots need to do more than just go straight and turn left and right to look like a real user. For this reason, artificial intelligence needs to make constant and minimal maneuvers in order to be good in the race.

3.5.Game System Design

The Waypoint System with Vector Calculations, which is used to move the artificial intelligence forward, is a coordinate consisting of points on the race track that are invisible to the user and determine the position of the artificial intelligence.

Another method used by artificial intelligence is trigger detection and artificial environment perception. Trigger detection, which is popular today, is one of the different mechanisms used in many games. It allows the objects in the game to perceive each other. Thanks to this mechanism, the vehicles in the game determine what they will do when they encounter an obstacle.

3.6.Implementation of Game System

Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X-exclusive game engine[6]. The engine has since been gradually extended to support a variety of desktop, mobile, console and virtual reality platforms [6].

MonoDevelop (also known as Xamarin Studio) is an open-source integrated development environment for Linux, macOS and Windows [7]. Its primary focus is development of projects that use Mono and .NET Framework [7].

3.7.Conclusion

Artificial intelligence used in game development has become an indispensable subject of this market today. This system, which is used in almost all games, is very important for users not to feel alone while playing single-player games and to think of their opponents as a real player. In addition, in multiplayer games using artificial intelligence, users can be more successful in multiplayer platforms by training themselves in artificial intelligence mode to improve themselves. In our article, we discussed the subject of game development with artificial intelligence and made a detailed literature search on this subject.

3.8.References

[1] K. Salen and E. Zimmerman, Rules of Play: Game Design Fundamentals, vol. 1, MIT Press, 2004.

[2] Marvin T. Chan, Christine W. Chan, and Craig Gelowitz, Development of a Car Racing Simulator Game Using Artificial Intelligence Techniques , Received 29 June 2015; Revised 30 September 2015; Accepted 8 October 2015.

[3] FIFA 22: https://www.ea.com/games/fifa/fifa-22.

[4] J. DeNero and D. Klein, “Teaching Introductory Artificial Intelligence with Pac-Man,” http://www.aaai.org/ocs/index.php/EAAI/EAAI10/paper/viewFile/1954/2331.

[5] Finite-State Machines: Theory and Implementation: https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867.

[6] Unity Game Engine: https://en.wikipedia.org/wiki/Unity_(game_engine)

[7] MonoDevelop: https://en.wikipedia.org/wiki/MonoDevelop

4.Software Requirements Specification(SRS)

4.1.Introduction

A racing game is a video game genre in which the player engages in a battle in which land, air, or watercraft compete in a first-person or third-person perspective. Competitions may be based on real-world racing leagues, or they may have a completely fictitious setup. Such games can have high (Simulation) or low (Arcade) realism. Racing games can also be classified as sports games [1]. The subject we work on in our project is the car racing game. Car racing games are developed day by day nowadays. The main reasons for its development are the use of artificial intelligence and the graphics getting closer to reality day by day. The main purpose of our project is to use artificial intelligence effectively in a car racing game and to increase the enjoyment of the people who play the game.

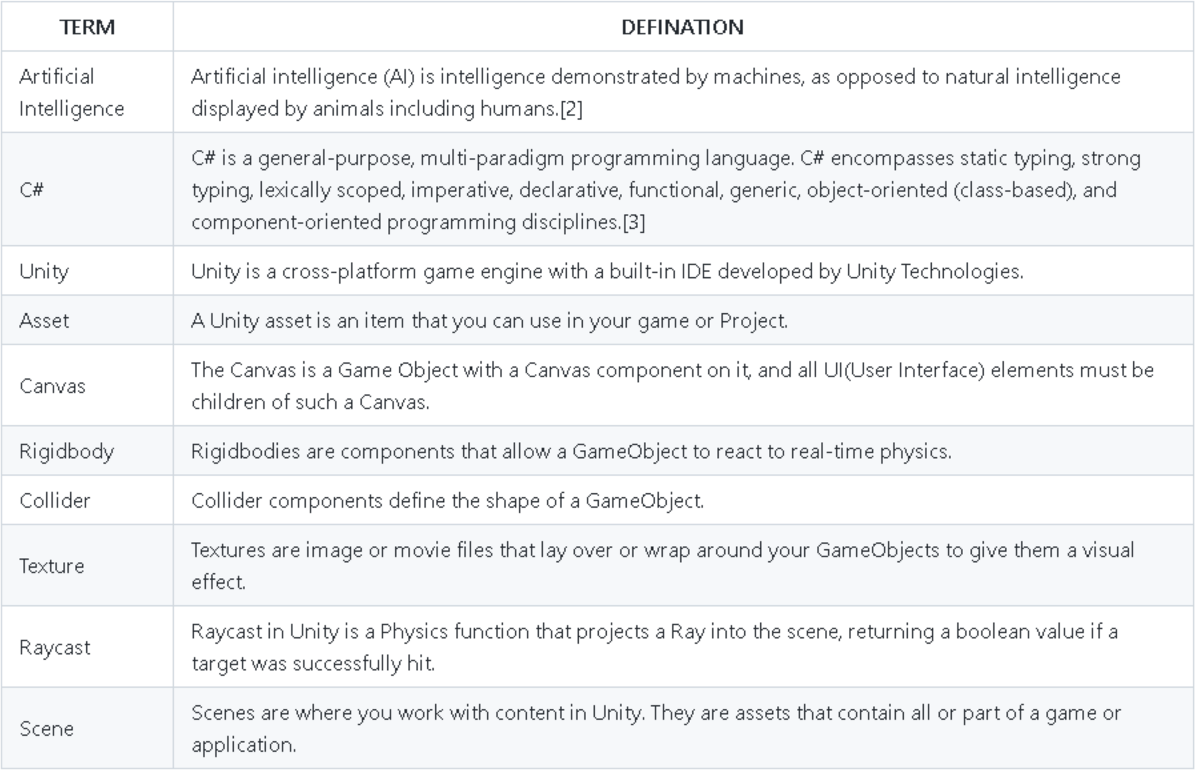
4.1.1.Purpose of This Document

In this project, our purpose is to develop a car racing game with artificial intelligence using Unity, a game development platform. In this game, you will encounter strong opponents controlled by artificial intelligence and your purpose is to complete the runway before them. While doing the project, we will use the assets and external models provided by Unity. We will create scripts using object-oriented programming and with these scripts we will direct the models we have received from outside. This SRS document contains the project requirements and software requirements specification for car racing game using AI to control competitor cars.

4.1.2.Scope of This Project

The people we want this project to reach are people who like car racing games and want to compete against artificial intelligence. The game will be designed to appeal to all age ranges. Thanks to its user-friendly interface, people will enjoy playing the game without any difficulty from the moment they enter the game.

4.1.3.Definitions, Acronyms, and Abbreviations



TERM	DEFINATION
Artificial Intelligence	Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to natural intelligence displayed by animals including humans.[2]
C#	C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.[3]
Unity	Unity is a cross-platform game engine with a built-in IDE developed by Unity Technologies.
Asset	A Unity asset is an item that you can use in your game or Project.
Canvas	The Canvas is a Game Object with a Canvas component on it, and all UI(User Interface) elements must be children of such a Canvas.
Rigidbody	Rigidbodies are components that allow a GameObject to react to real-time physics.
Collider	Collider components define the shape of a GameObject.
Texture	Textures are image or movie files that lay over or wrap around your GameObjects to give them a visual effect.
Raycast	Raycast in Unity is a Physics function that projects a Ray into the scene, returning a boolean value if a target was successfully hit.
Scene	Scenes are where you work with content in Unity. They are assets that contain all or part of a game or application.

4.1.4.Overview

This document includes:

- General information about the document.
- The goal of the project.
- General information about the project.
- Detailed explanation of the requirements.

4.2.Overall Description

In this section we will go through the main features and requirements of the car racing game.

4.2.1.Product Perspective

Our system includes only the software part. In our project, we will use artificial intelligence, which is a branch of computer science. We will use the C# programming language while using it. We will use Unity and Visual Studio throughout the development process. Where the models in Unity are insufficient, we will benefit from other sites. A computer is required for this project to run. We use UnityEngine and UnityEngine.UI libraries to include Unity in our project while working through Visual Studio. After downloading the game to your computer, there is no need for an internet connection.

4.2.2.Product Functions

- **Open Game:** To open the game, you need to click on its icon on the desktop 2 times.
- **Play Game:** When you click on the play the game button, it directs you to make a map selection.
- **Select Map:** The scene where you will choose one of the 2 different maps you want to play.
- **Select Car:** The scene where you can choose your car to play.
- **Play Again:** It allows you to replay the map.
- **Main Menu:** The button on the canvas that comes after the race is over. Allows you to return to the main menu.

4.2.3.User Characteristics

In this section actors of our project are listed.

User

The user is responsible for starting the game.

- Starts the game by clicking the icon.
- Clicks the "Play Game" button.
- Selects map.
- Selects car.
- Directs the car by pressing inputs.

WASD Car Game

The objectives of the game are as follows.

- Offers different map options after pressing play button.
- Offers car options.
- Starts the game according to the selected map and car.
- Directs AI cars.

4.3.Assumptions and Dependencies

In this section, we have specified the system requirements.

Environment Conditions:

After downloading the program, there is no need for an internet connection. The existing operating system is assumed to be windows.

Operating System:

The system must be supported by windows. The program must be installed on the computer.

Hardware:

Estimated minimum system requirements:

- CPU: Pentium III or Athlon XP
- CPU SPEED: 933 mhz
- RAM: 256 MB
- OS: Windows 98/98SE/ME/2000/XP
- VIDEO CARD: AGP Video Card with 32 MB and a DirectX 8.1 compatible driver (GeForce2+ /Radeon 7500+)
- TOTAL VIDEO RAM: 32 MB
- 3D: Yes
- DIRECTX VERSION: 8.1
- FREE DISK SPACE: 2 GB

Software:

No extra software is needed for the program to run.

4.4.Requirement Specification

4.4.1.Interface Requirements

User Interfaces

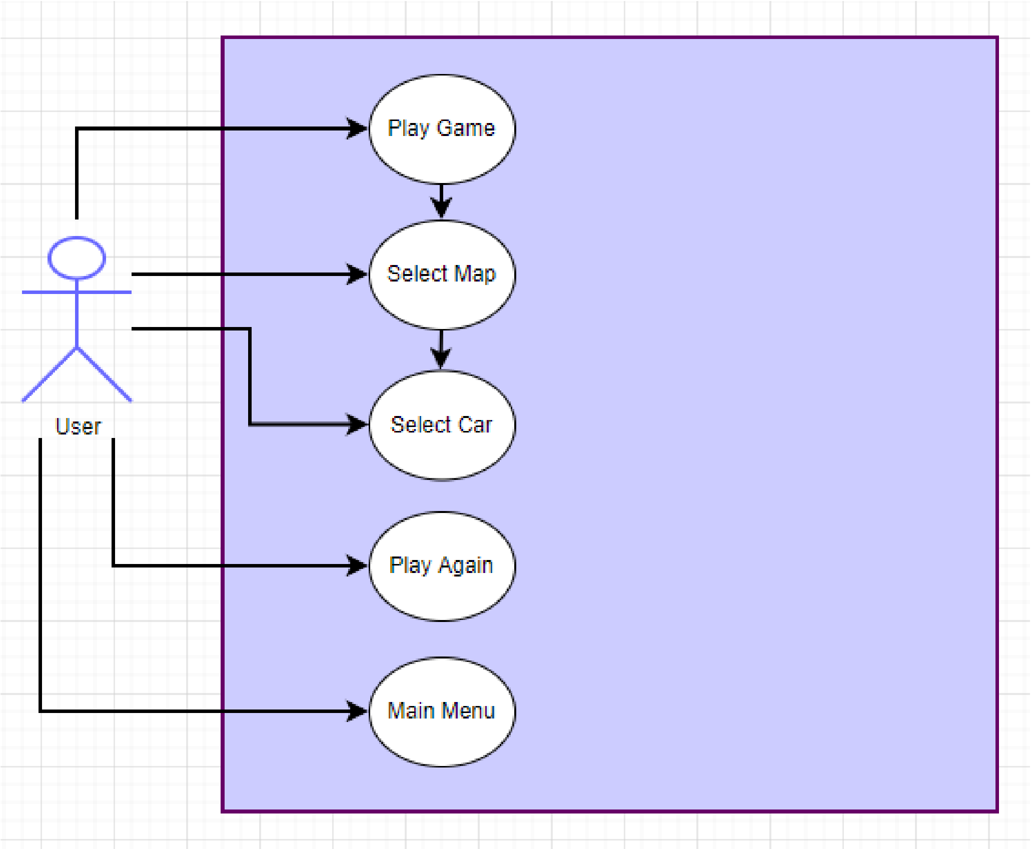
The user interface will be in a style that people are used to and comfortable with. While doing this, we will develop our design by considering human computer interaction theories. The language of the application will be English. After entering the application, there will be sections where various choices are made. There is a pause button so that the user can stop the game at any time.

Software Interfaces

We will use Unity 5.6.7 while preparing the software of the game. While developing the software, we develop it through Visual Studio and Unity. We will create the interface design manually with the canvases that Unity offers us.

4.4.2.Functional Requirements

4.4.2.1.Use Case Diagram



4.4.2.2.Use Cases

All cases for the user are given below.

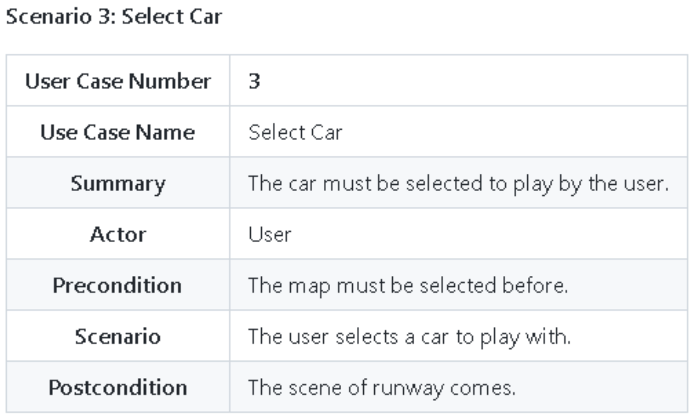
Use Case	Description
Play Game	The button to allow user to play the game.
Select Map	The scene that user should select his/her own map to want to play on.
Select Car	The scene that user should select the car to play with.
Play Again	The canvas element/button that shows up when the game is over and ask user to if he/she wants to play again or not.
Main Menu	The canvas element/button that shows up when the game is over and ask user to if he/she wants to leave the game and return to the main menu or not.

Scenario 1: Play Game

User Case Number	1
Use Case Name	Play Game
Summary	The game is started by the user.
Actor	User
Precondition	• The game must be downloaded to the computer. • The game must have been opened before.
Scenario	• The user clicks to icon to open the program. • Then clicks to the button "Play Game"
Postcondition	The button directs you to the map selection.

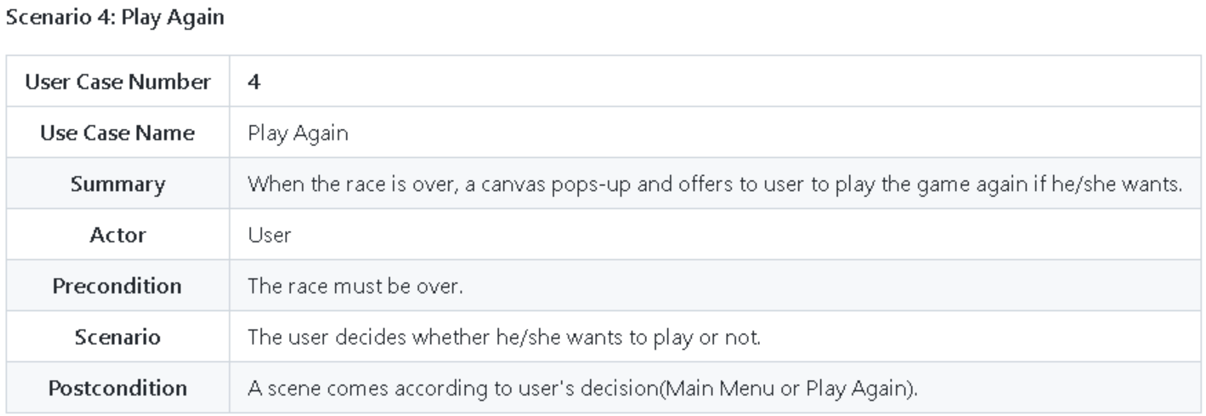
Scenario 2: Select Map

User Case Number	2
Use Case Name	Select Map
Summary	The map must be selected to play by the user.
Actor	User
Precondition	The game must be opened and the button "Play Game" must be clicked.
Scenario	The user selects a map out of 2 maps, then clicks the button "Next".
Postcondition	The scene of car selection comes.



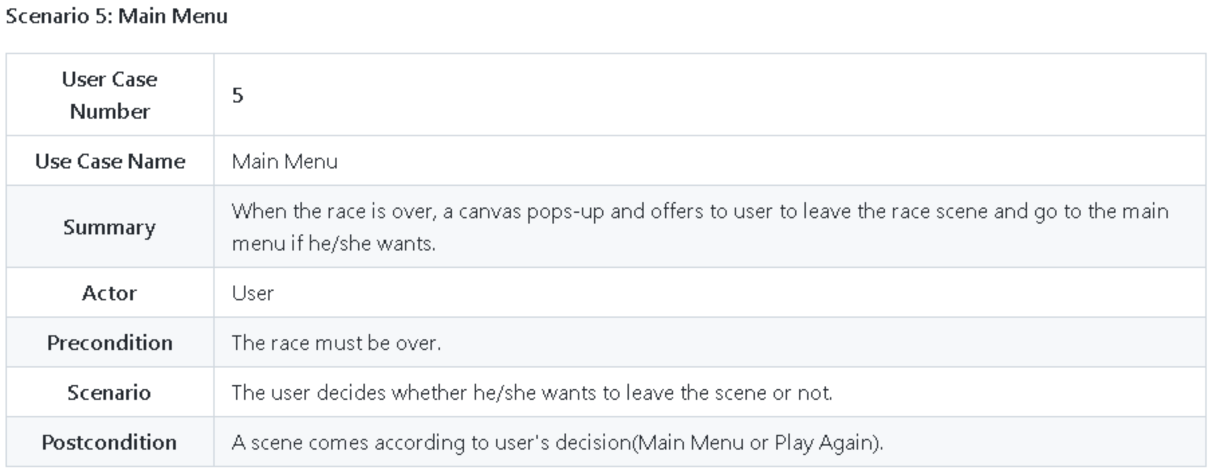
Scenario 3: Select Car

User Case Number	3
Use Case Name	Select Car
Summary	The car must be selected to play by the user.
Actor	User
Precondition	The map must be selected before.
Scenario	The user selects a car to play with.
Postcondition	The scene of runway comes.



Scenario 4: Play Again

User Case Number	4
Use Case Name	Play Again
Summary	When the race is over, a canvas pops-up and offers to user to play the game again if he/she wants.
Actor	User
Precondition	The race must be over.
Scenario	The user decides whether he/she wants to play or not.
Postcondition	A scene comes according to user's decision(Main Menu or Play Again).



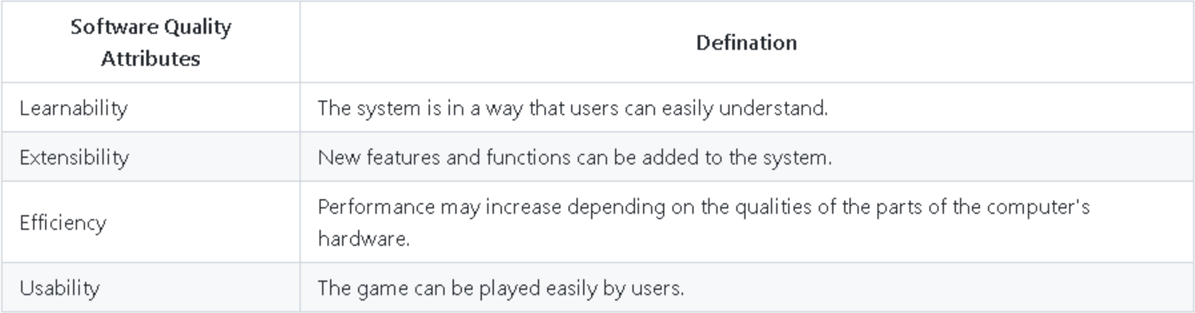
Scenario 5: Main Menu

User Case Number	5
Use Case Name	Main Menu
Summary	When the race is over, a canvas pops-up and offers to user to leave the race scene and go to the main menu if he/she wants.
Actor	User
Precondition	The race must be over.
Scenario	The user decides whether he/she wants to leave the scene or not.
Postcondition	A scene comes according to user's decision(Main Menu or Play Again).

4.4.3.Non-Functional Requirements

The non-functional parts of the project are listed below.

4.4.3.1.Software Quality Attributes



Software Quality Attributes	Defination
Learnability	The system is in a way that users can easily understand.
Extensibility	New features and functions can be added to the system.
Efficiency	Performance may increase depending on the qualities of the parts of the computer's hardware.
Usability	The game can be played easily by users.

4.5.References

[1] Racing game: https://tr.wikipedia.org/wiki/Yar%C4%B1%C5%9F_oyunu

[2] Artificial Intelligence: https://en.wikipedia.org/wiki/Artificial_intelligence

[3] C#: https://en.wikipedia.org/wiki/Artificial_intelligence

5.Software Design Description

5.1.Introduction

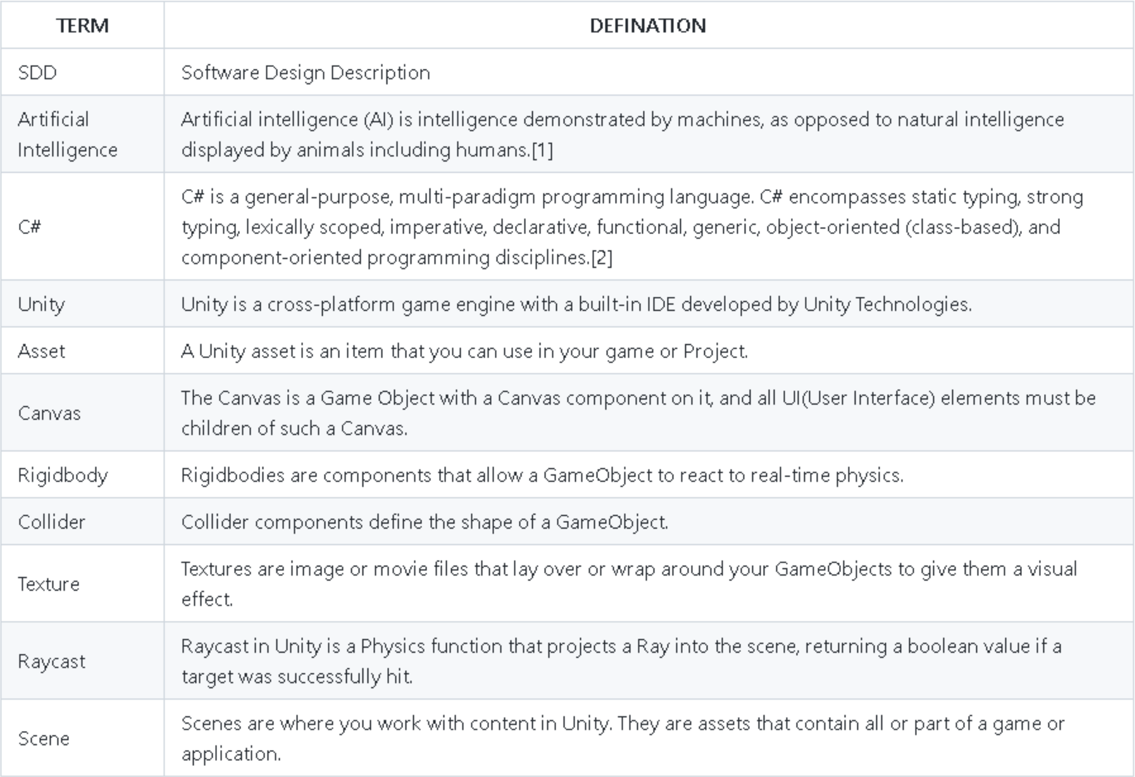
5.1.1.Purpose

This document (SDD) covers the design details of a car racing game project against artificial intelligence. The purpose of this document is to help people who will implement the project.

5.1.2.Scope

This game project is a software that aims to compete against artificial intelligence by choosing different cars and different maps. This system will be designed for people who love to play games and more specifically car racing games. The game will be designed to appeal to all age groups. The main purpose is to give people a realistic gaming experience. For this purpose, strengthened artificial intelligence tools are used. The game consists of a main menu at the first entry. Then you will see the map and car selection screens. After the selections are made, the game will start the race by bringing random opponents from a group of cars according to your choices.

5.1.3.Glossary



TERM	DEFINATION
SDD	Software Design Description
Artificial Intelligence	Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to natural intelligence displayed by animals including humans.[1]
C#	C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.[2]
Unity	Unity is a cross-platform game engine with a built-in IDE developed by Unity Technologies.
Asset	A Unity asset is an item that you can use in your game or Project.
Canvas	The Canvas is a Game Object with a Canvas component on it, and all UI(User Interface) elements must be children of such a Canvas.
Rigidbody	Rigidbodies are components that allow a GameObject to react to real-time physics.
Collider	Collider components define the shape of a GameObject.
Texture	Textures are image or movie files that lay over or wrap around your GameObjects to give them a visual effect.
Raycast	Raycast in Unity is a Physics function that projects a Ray into the scene, returning a boolean value if a target was successfully hit.
Scene	Scenes are where you work with content in Unity. They are assets that contain all or part of a game or application.

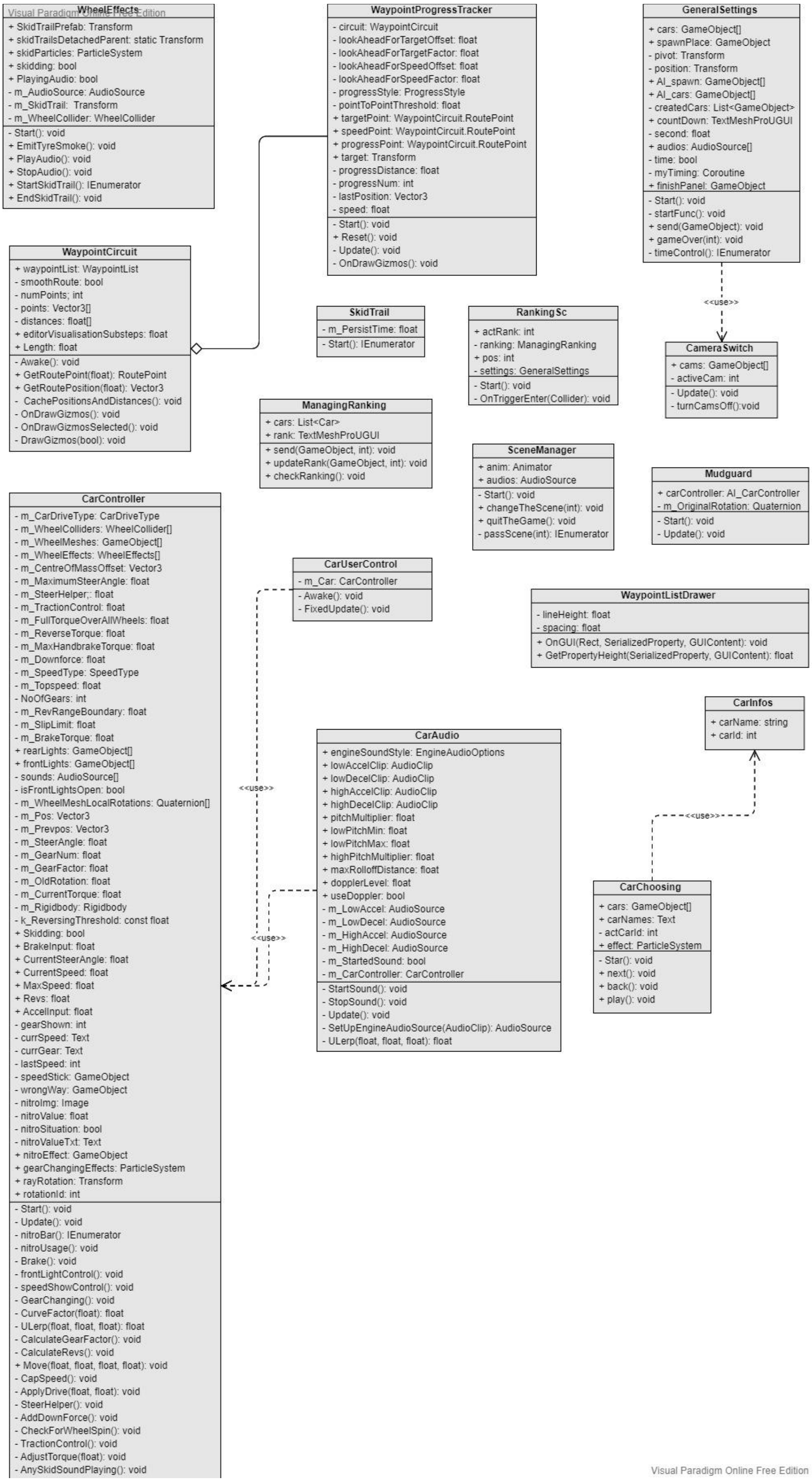
5.1.4.Overview

This Software Design Description document includes:

- General information about the document.
- The goal of the project.
- General information about the project.
- Detailed explanation of the design approach.
- Diagrams and the interfaces in the game.

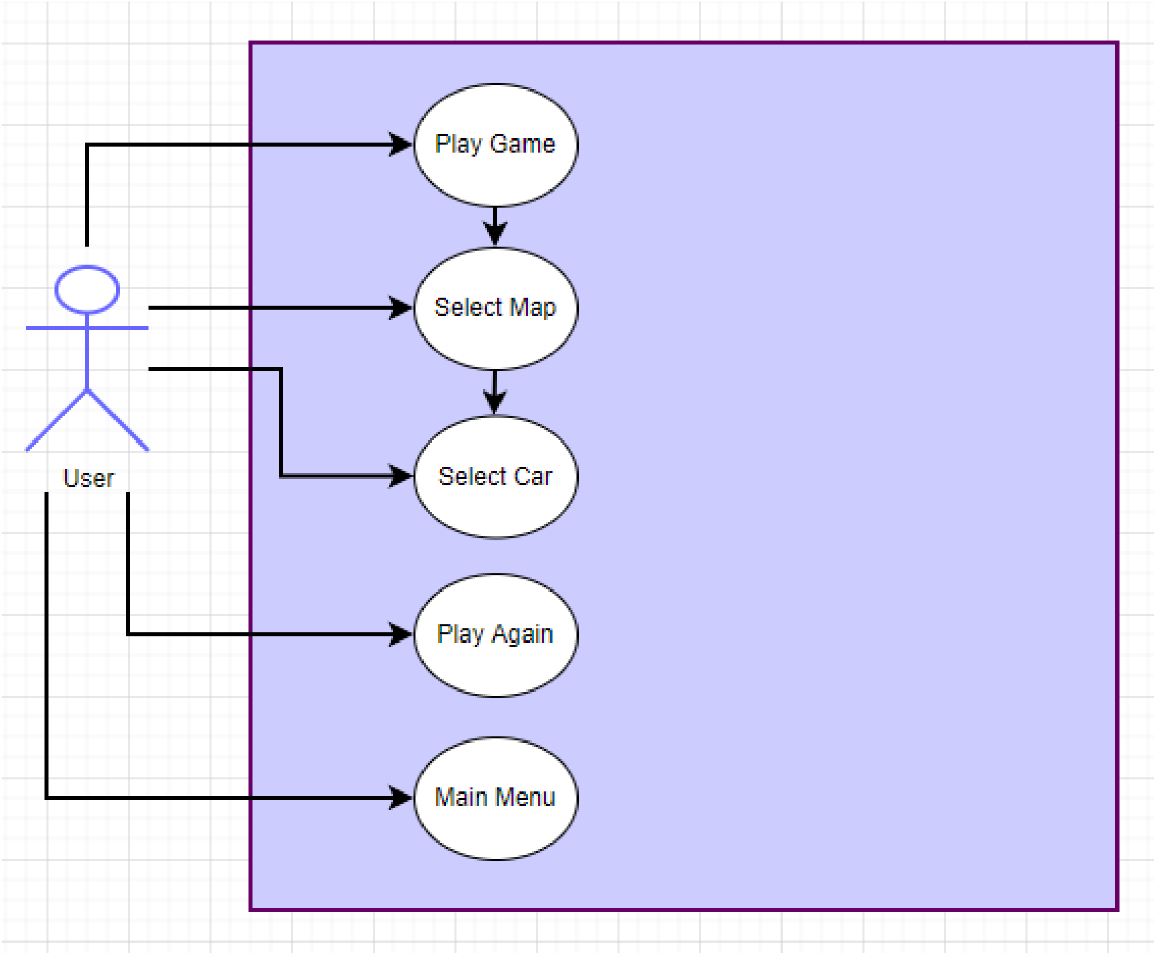
5.2.Architecture Design

5.2.1.Class Diagram


Visual Paradigm Online Free Edition
WheelEffects
+ SkidTrailPrefab: Transform
+ skidTrailsDetachedParent: static Transform
+ skidParticles: ParticleSystem
+ skidding: bool
+ PlayingAudio: bool
- m_AudioSource: AudioSource
- m_SkidTrail: Transform
- m_WheelCollider: WheelCollider
- Start(): void
+ EmitTyreSmoke(): void
+ PlayAudio(): void
+ StopAudio(): void
+ StartSkidTrail(): IEnumerator
+ EndSkidTrail(): void
WaypointProgressTracker
- circuit: WaypointCircuit
- lookAheadForTargetOffset: float
- lookAheadForTargetFactor: float
- lookAheadForSpeedOffset: float
- lookAheadForSpeedFactor: float
- progressStyle: ProgressStyle
- pointToPointThreshold: float
+ targetPoint: WaypointCircuit.RoutePoint
+ speedPoint: WaypointCircuit.RoutePoint
+ progressPoint: WaypointCircuit.RoutePoint
+ target: Transform
- progressDistance: float
- progressNum: int
- lastPosition: Vector3
- speed: float
- Start(): void
+ Reset(): void
- Update(): void
- OnDrawGizmos(): void
GeneralSettings
+ cars: GameObject[]
+ spawnPlace: GameObject
- pivot: Transform
- position: Transform
+ AI_spawn: GameObject[]
+ AI_cars: GameObject[]
- createdCars: List<GameObject>
+ countDown: TextMeshProUGUI
- second: float
+ audios: AudioSource[]
- time: bool
- myTiming: Coroutine
+ finishPanel: GameObject
- Start(): void
- startFunc(): void
+ send(GameObject): void
+ gameOver(int): void
- timeControl(): IEnumerator
WaypointCircuit
+ waypointList: WaypointList
- smoothRoute: bool
- numPoints; int
- points: Vector3[]
- distances: float[]
+ editorVisualisationSubsteps: float
+ Length: float
- Awake(): void
+ GetRoutePoint(float): RoutePoint
+ GetRoutePosition(float): Vector3
- CachePositionsAndDistances(): void
- OnDrawGizmos(): void
- OnDrawGizmosSelected(): void
- DrawGizmos(bool): void
SkidTrail
- m_PersistTime: float
- Start(): IEnumerator
RankingSc
+ actRank: int
- ranking: ManagingRanking
+ pos: int
- settings: GeneralSettings
- Start(): void
- OnTriggerEnter(Collider): void
<<use>>
CameraSwitch
+ cams: GameObject[]
- activeCam: int
- Update(): void
- turnCamsOff():void
ManagingRanking
+ cars: List<Car>
+ rank: TextMeshProUGUI
+ send(GameObject, int): void
+ updateRank(GameObject, int): void
+ checkRanking(): void
SceneManager
+ anim: Animator
+ audios: AudioSource
- Start(): void
+ changeTheScene(int): void
+ quitTheGame(): void
- passScene(int): IEnumerator
Mudguard
+ carController: AI_CarController
- m_OriginalRotation: Quaternion
- Start(): void
- Update(): void
CarController
- m_CarDriveType: CarDriveType
- m_WheelColliders: WheelCollider[]
- m_WheelMeshes: GameObject[]
- m_WheelEffects: WheelEffects[]
- m_CentreOfMassOffset: Vector3
- m_MaximumSteerAngle: float
- m_SteerHelper;: float
- m_TractionControl: float
- m_FullTorqueOverAllWheels: float
- m_ReverseTorque: float
- m_MaxHandbrakeTorque: float
- m_Downforce: float
- m_SpeedType: SpeedType
- m_Topspeed: float
- NoOfGears: int
- m_RevRangeBoundary: float
- m_SlipLimit: float
- m_BrakeTorque: float
+ rearLights: GameObject[]
+ frontLights: GameObject[]
- sounds: AudioSource[]
- isFrontLightsOpen: bool
- m_WheelMeshLocalRotations: Quaternion[]
- m_Pos: Vector3
- m_Prevpos: Vector3
- m_SteerAngle: float
- m_GearNum: float
- m_GearFactor: float
- m_OldRotation: float
- m_CurrentTorque: float
- m_Rigidbody: Rigidbody
- k_ReversingThreshold: const float
+ Skidding: bool
+ BrakeInput: float
+ CurrentSteerAngle: float
+ CurrentSpeed: float
+ MaxSpeed: float
+ Revs: float
+ AccelInput: float
- gearShown: int
- currSpeed: Text
- currGear: Text
- lastSpeed: int
- speedStick: GameObject
- wrongWay: GameObject
- nitroImg: Image
- nitroValue: float
- nitroSituation: bool
- nitroValueTxt: Text
+ nitroEffect: GameObject
+ gearChangingEffects: ParticleSystem
+ rayRotation: Transform
+ rotationId: int
- Start(): void
- Update(): void
- nitroBar(): IEnumerator
- nitroUsage(): void
- Brake(): void
- frontLightControl(): void
- speedShowControl(): void
- GearChanging(): void
- CurveFactor(float): float
- ULerp(float, float, float): float
- CalculateGearFactor(): void
- CalculateRevs(): void
+ Move(float, float, float, float): void
- CapSpeed(): void
- ApplyDrive(float, float): void
- SteerHelper(): void
- AddDownForce(): void
- CheckForWheelSpin(): void
- TractionControl(): void
- AdjustTorque(float): void
- AnySkidSoundPlaying(): void
CarUserControl
- m_Car: CarController
- Awake(): void
- FixedUpdate(): void
WaypointListDrawer
- lineHeight: float
- spacing: float
+ OnGUI(Rect, SerializedProperty, GUIContent): void
+ GetPropertyHeight(SerializedProperty, GUIContent): float
CarInfos
+ carName: string
+ carId: int
<<use>>
CarAudio
+ engineSoundStyle: EngineAudioOptions
+ lowAccelClip: AudioClip
+ lowDecelClip: AudioClip
+ highAccelClip: AudioClip
+ highDecelClip: AudioClip
+ pitchMultiplier: float
+ lowPitchMin: float
+ lowPitchMax: float
+ highPitchMultiplier: float
+ maxRolloffDistance: float
+ dopplerLevel: float
+ useDoppler: bool
- m_LowAccel: AudioSource
- m_LowDecel: AudioSource
- m_HighAccel: AudioSource
- m_HighDecel: AudioSource
- m_StartedSound: bool
- m_CarController: CarController
- StartSound(): void
- StopSound(): void
- Update(): void
- SetUpEngineAudioSource(AudioClip): AudioSource
- ULerp(float, float, float): float
<<use>>
<<use>>
CarChoosing
+ cars: GameObject[]
+ carNames: Text
- actCarId: int
+ effect: ParticleSystem
- Star(): void
+ next(): void
+ back(): void
+ play(): void
Visual Paradigm Online Free Edition

5.2.2.Use Case Diagram

The main form of the use-case diagram is as follows. The detailed version is explained in the SRS document.

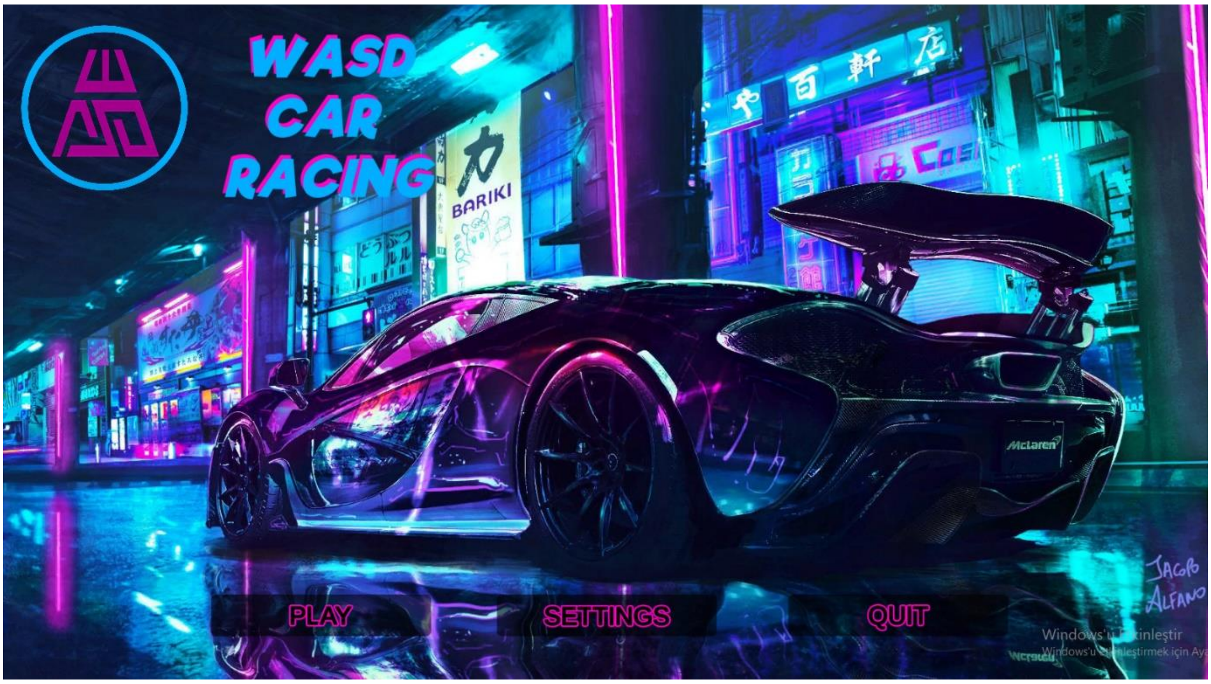


First, the user starts the game by clicking on the game icon. After the user starts the game, the main menu appears. If the user wants to play the game, s/he clicks the play button and the map selection scene appears. After selecting the desired map, the game directs the user to the car selection scene. After the desired car is selected, the race begins.

5.3.User Interfaces

5.3.1.Main Menu

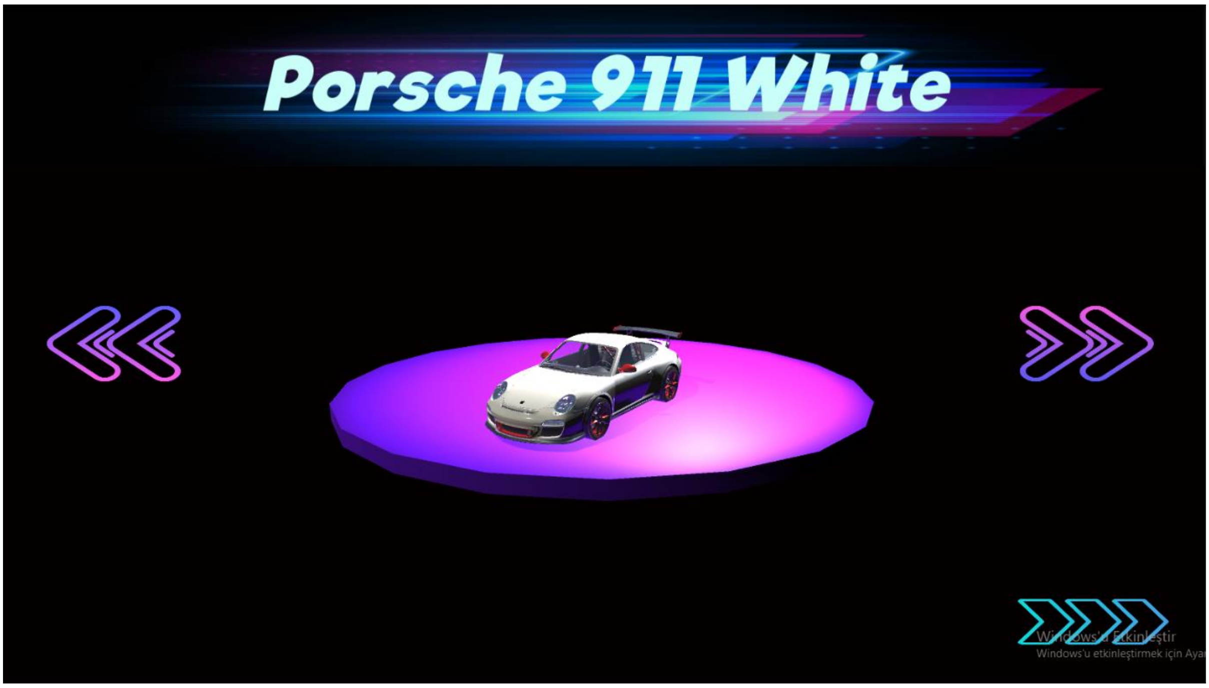
Below, the image shows the main menu of the game.



The user encounters this menu after starting the game. The menu allows the player to start or exit the game.

5.3.2.Car Selection

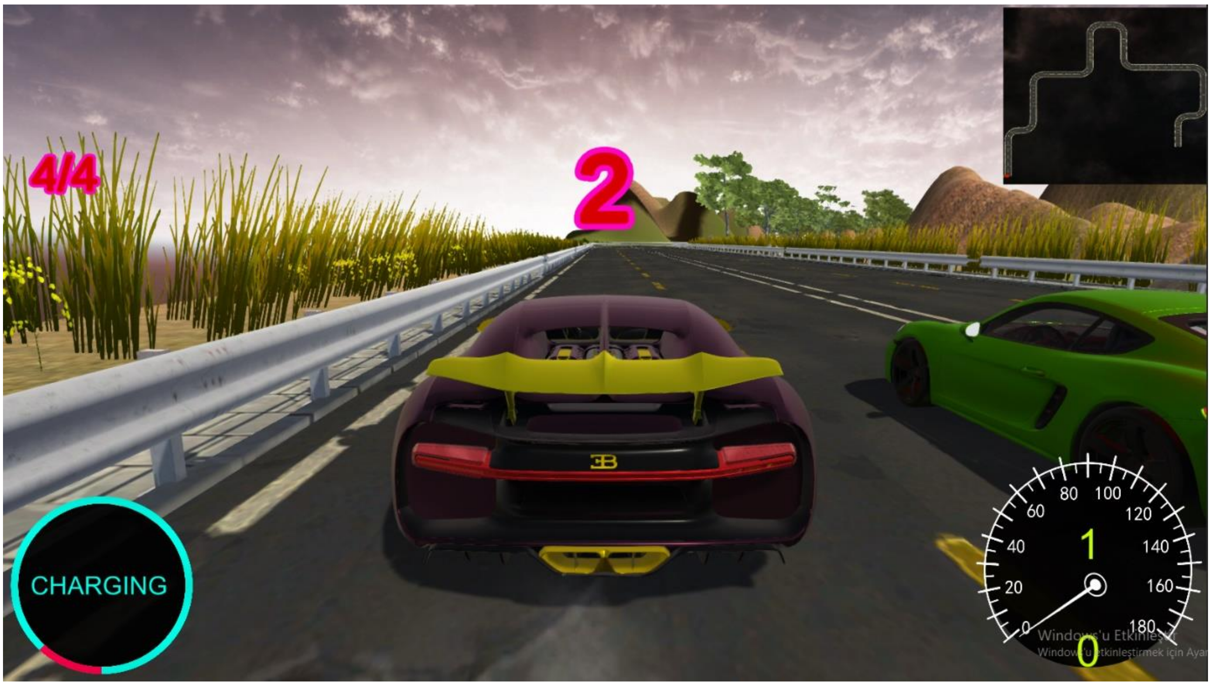
Below, the image shows the car selection scene of the game.



The user encounters this scene after clicking the "start" button from the main menu. Here s/he can start the game by choosing the car s/he wants.

5.3.3.Race Start Scene

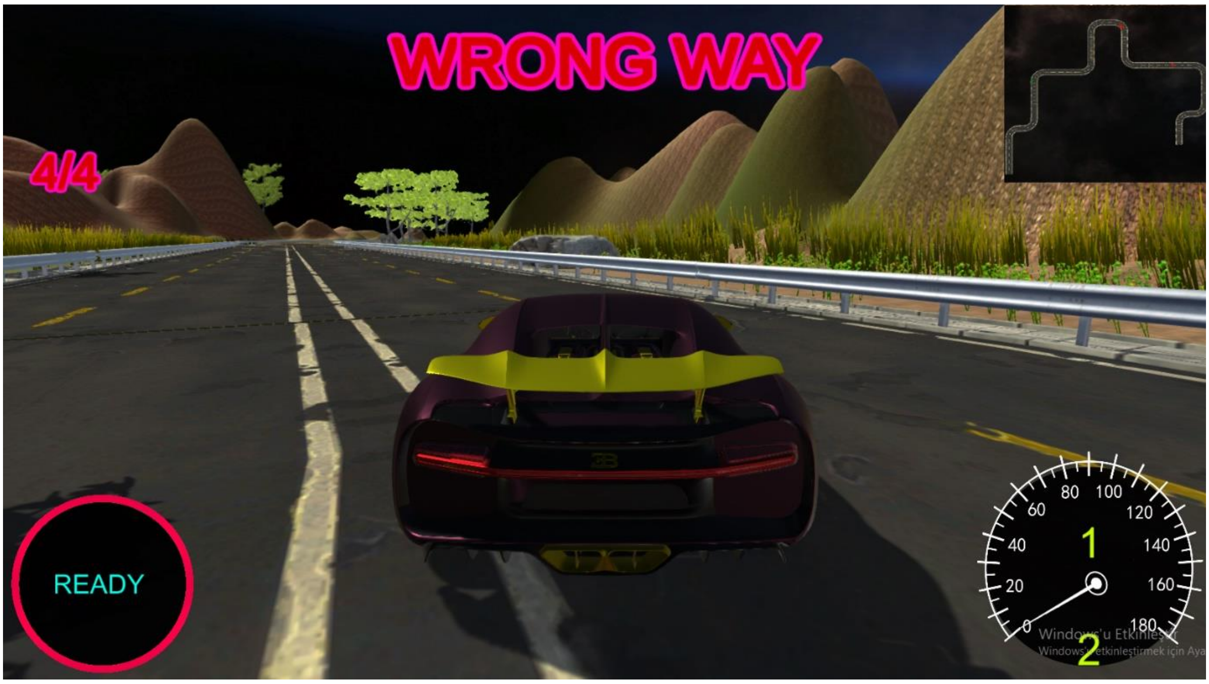
Below, the image shows the scene at the beginning of the race scene of the game.



After the user selects the desired car and map, he/she encounters this screen. After the countdown, the race begins.

5.3.4.Wrong Way Warning

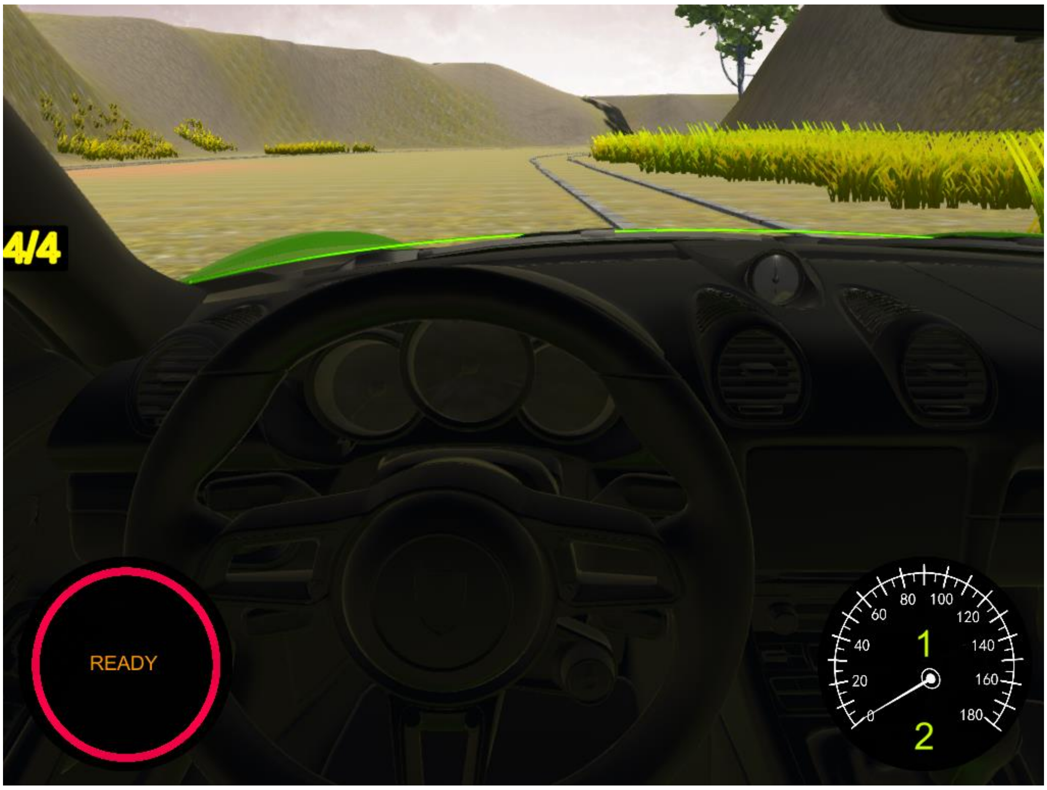
Below, the image shows "the user's car going wrong way" of the race scene of the game.



If the user goes against the given route, he will encounter this error scene.

5.3.5.Camera Switch Example

Below, the image shows the camera look 3 of the game.



Users have different camera look options to choose from. This is one of them.

5.4.References

[1] Artificial Intelligence: https://en.wikipedia.org/wiki/Artificial_intelligence

[2] C#: https://en.wikipedia.org/wiki/Artificial_intelligence

6.Conclusion & Discussion

With the development of artificial intelligence technology, the freedom of thought and diversity of movement continue to increase with the increase in the pleasure and functions that game-loving people experience while playing games. The fact that the technologies used are updated by adding new features provides a plus for many computer / software-based works, including the game world. In this project, which we tried to develop with new software tools, we tried to test artificial intelligence and add features such as freedom of movement for the user and overcoming their problems easily. While making this project, we tried to be different and to make it a long-term car racing game by using software techniques open to development by taking into consideration the previous projects.