Çankaya University

Department of Computer Engineering

Software Design Description

for

CAMPUS KART RACING GAME

(CKRG)

Version 1.0

Prepared By:

Esra ŞAHİN

Ozan ÇETİNER

Buğra DOĞAN

Cem ALTUNBULDUK

Change History

This was the first version of document . The document was prepared on 24 December 2020.

List of Figures

Table of Contents

1 INTRODUCTION

1.1 Purpose

The purpose of this Software Design Document is to provide detailed technical information about the project named Campus Kart Racing. In order to provide a better comprehension, this SDD includes various diagrams such as UML diagram of the project, activity diagram and block diagram.

1.2 Scope

This document contains details of CKRG.

Unity Game Engine is used to combine the game's physics, sound, artificial intelligence and game scenes. Unity is widely used because it offers many options such as low system requirements and physics calculations.[1] Unity uses the C# infrastructure as its coding language. It also offers cross-platform support. For these reasons, we chose Unity as the game engine.

We will be using 3Ds Max and Blender as 3D modeling tools. Blender is an open source 3D modeling tool that requires low system requirements and allows you to model in any category. 3Ds Max, on the other hand, is a powerful 3D modeling tool developed by Autodesk, one of the leading companies in the industry.

C # programming language will be used to create the script. C # is an object oriented programming language that offers a C-like experience developed by Microsoft.[2]

We will use the Photon Network infrastructure that is compatible with Unity, as the game offers in multiplayer mode other than single player mode. Photon Network is used by over 500,000 developers because it is easy to implement, self-documented and partially free.

1.3 Glossary

Expression	Meaning
CKRG	Campus Kart Racing Game
AI	Artificial Intelligence
CPG	Cross Platform Game
NPC	Non Playable Character (Controlled by AI)

GUI	Graphical User Interface
SDD	Software Design Document
UML	Unified Modelling Language (Used in Software Engineering)
PUN	Photon Unity Networking
MP	Multiplayer
OS	Operating System

1.4 Motivation

We are a group of seniors interested in gaming in the computer engineering department. As a group, our goal is to provide players with a playable, fun and competitive racing game experience. For this purpose, we chose Unity as the game engine we are familiar with as a group.We did research and purchased courses to make up for our shortcomings in racing game mechanics. We learn to use 3D modeling tools for designs. We are considering getting professional help in the points we are missing.

2 ARCHITECTURAL APPROACH

2.1 Simulation Design Approach

We used the Agile Management system in this project. In this form of project management, there is a continuous form of a gradual solution from the start date of the project instead of presenting the product after completion. In Agile system, projects are divided into smaller parts. It is then ranked according to importance. It is then delivered and checked in smaller weekly cycles called "Sprints." The goals are set by the group before the cycle begins. In some special cases the team must contact the customer directly and answer questions. Project workers and leaders can predict roughly how long the project will take. Feedback is received and recorded from the customer after each presentation and these notes can be used usefully in later stages. This system aims to continuously develop and improve the product.

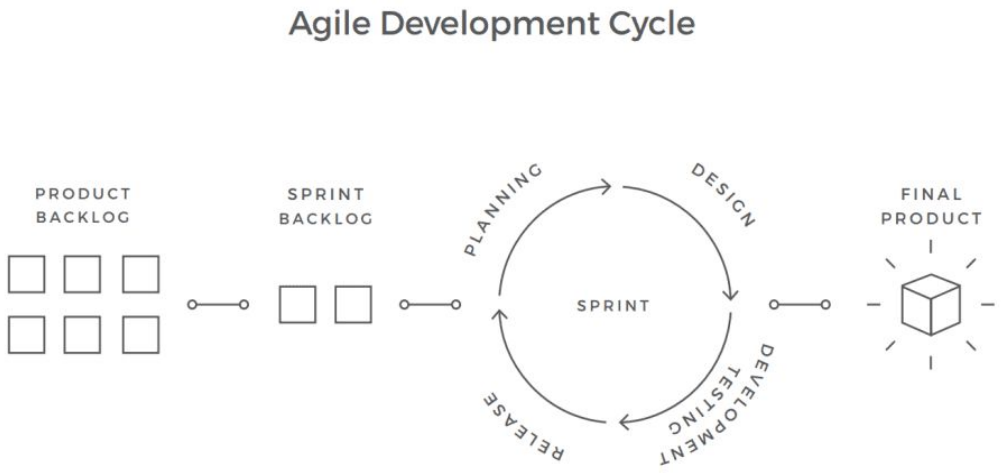

Figure1: Agile Development Cycle

2.1.1 Benefits of Using Agile Method

During the project, user participation is encouraged, making it easier to identify problems and make the experience more systematic for the team. There is feedback throughout the whole process, so progress is made.Thanks to this methodology, development is cared for from the very beginning of the project and major problems are avoided.

Teamwork: Team members work together. Everyone knows everyone's duty. The probability of problems is minimized.

Improvements: Various improvements are continuously made and tested in cycles to reveal the final product.

Client Side: A client is closely involved in development and can change requirements or accept the team's suggestions.

Distribution of Work: The project consists of small cycles (Sprints).

Flexibility: The scope of the project can be changed and transferred according to the client's request or variables. Flexibility can be provided according to feedback or other situations.

Gantt charts are used to plan projects of all sizes and are a useful way to show what work is planned on a given day.

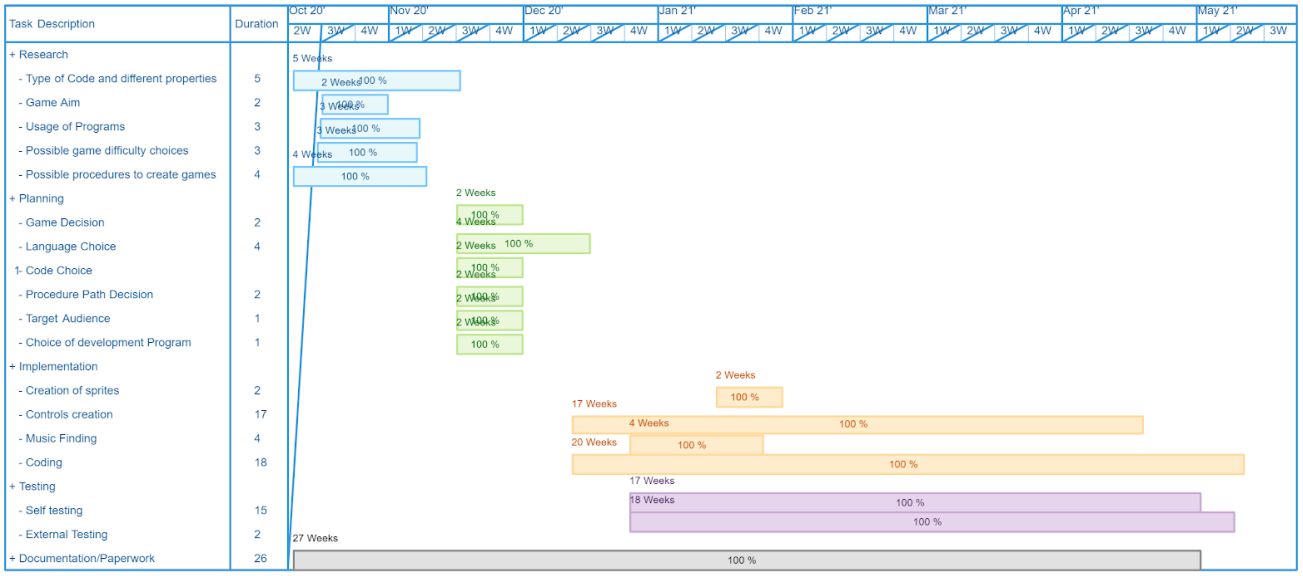

Figure 2: Gantt Chart for Our Project

2.1.2 Class Diagram

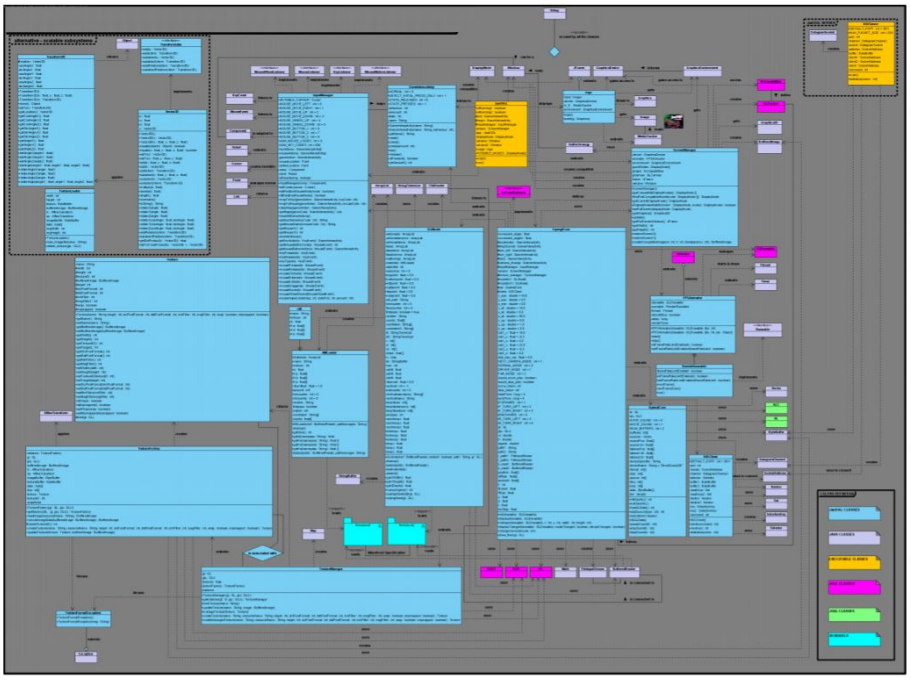

Figure 3:Class Diagram

In the figure above, the connections between classes are shown clearly. But this shape does not meet every need. This diagram, which is currently prepared only to show the in-game racing dynamics, will change in future sprint meetings and will continue to be developed. The CGame class retains the main functions, while the map and coordinates classes control the location of the tool and work with the pathfinding algorithm. Highscore class keeps the high scores and rankings as can be predicted.

2.2 Architectural Estimated Work Patterns

2.2.1 Physics and Movement

The physics of the car is calculated using three-dimensional vectors (Vector3 [3]) in the CarController script. The physical attributes of the car are:

Mass (float mass)

Direction of movement and speed (Vector3 velocity)

Position (Vector3 position)

Forward (Vector3 forward)

Upward (Vector3 upward)

A car also has some motion attributes. An AnimationCurve [4] is a curve that can be edited directly in the graphic. The following are the car's motion attributes:

Acceleration / backward acceleration (float acceleration / backAcceleration)

Inertia of acceleration (AnimationCurve accelerationDrag)

Acceleration of rotation (AnimationCurve torque)

Inertia of rotation (float rotateDrage)

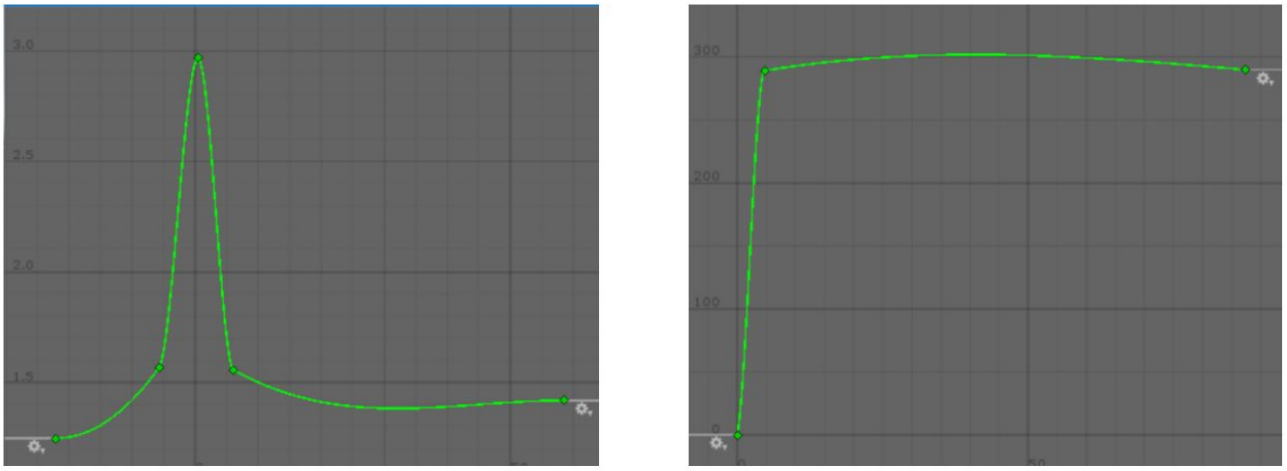

Figure 4: Acceleration Drag and torque

The calculation of the movement:

velocity + = InputY () * forward * acceleration * deltaTime (deltaTime is the time between each frame, InputX () / InputY () is the horizontal / vertical input of the player between -1 and

AddTorque (InputX () * upward * torque.Evaluate (velocity.magnitude) * deltaTime)

(AddTorque () is a function that adds acceleration to the rotation of the car. Evaluate () takes the value of the x-axis and returns the value of the y-axis of the curve. Magnitude is the length of a vector)

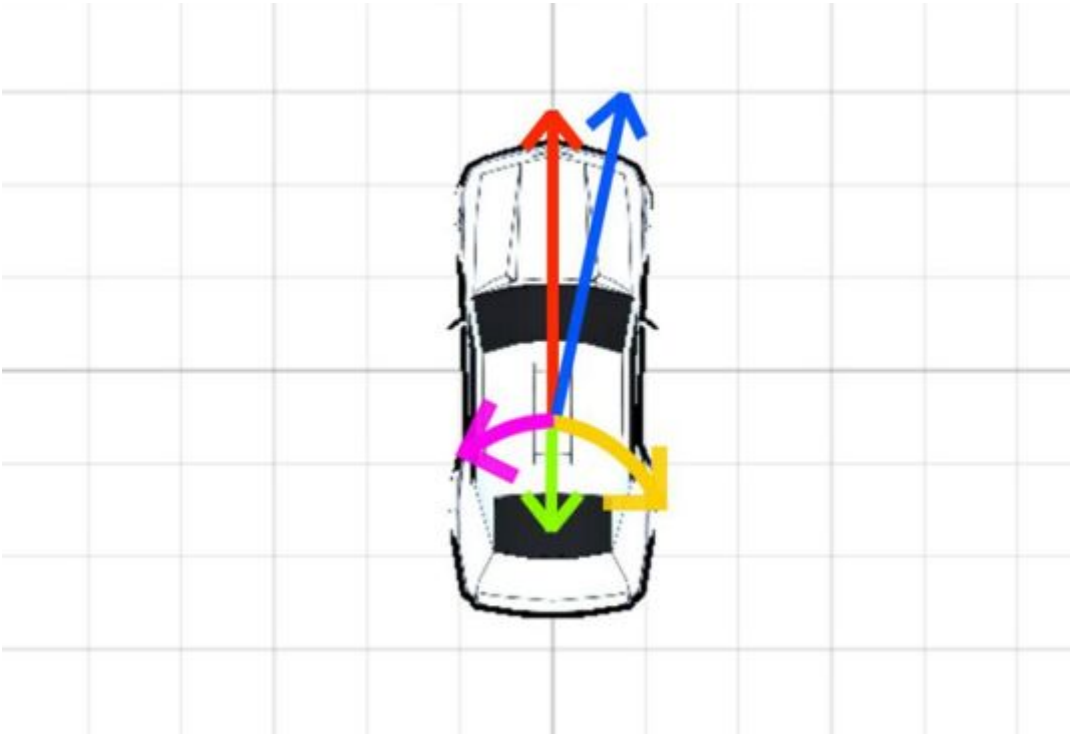

Figure 5: Car Vectors

Red: initially velocity, green: accelerationDrag, yellow: torque, purple: torqueDrag, blue: new velocity

After each frame the friction is calculated:

Velocity * = (1-accelerationDrag.Evaluate (velocity.z)) * deltaTime

angularVelocity * = (1-rotateDrag) * deltaTime

2.2.2 Network and Synchronization

PUN is a Unity package for multiplayer games. Offering flexible matching, custom properties, fast and reliable connection, PUN is used by many developers.

Figure 6: How PUN Works

Players connect to the server and start exchanging data. This data can be data such as the speed, location, color of the player.

The master server is a server set up for each game on the PUN system. Players can connect to this main server and can open, list and join rooms as desired. In our game, each room will represent a race.

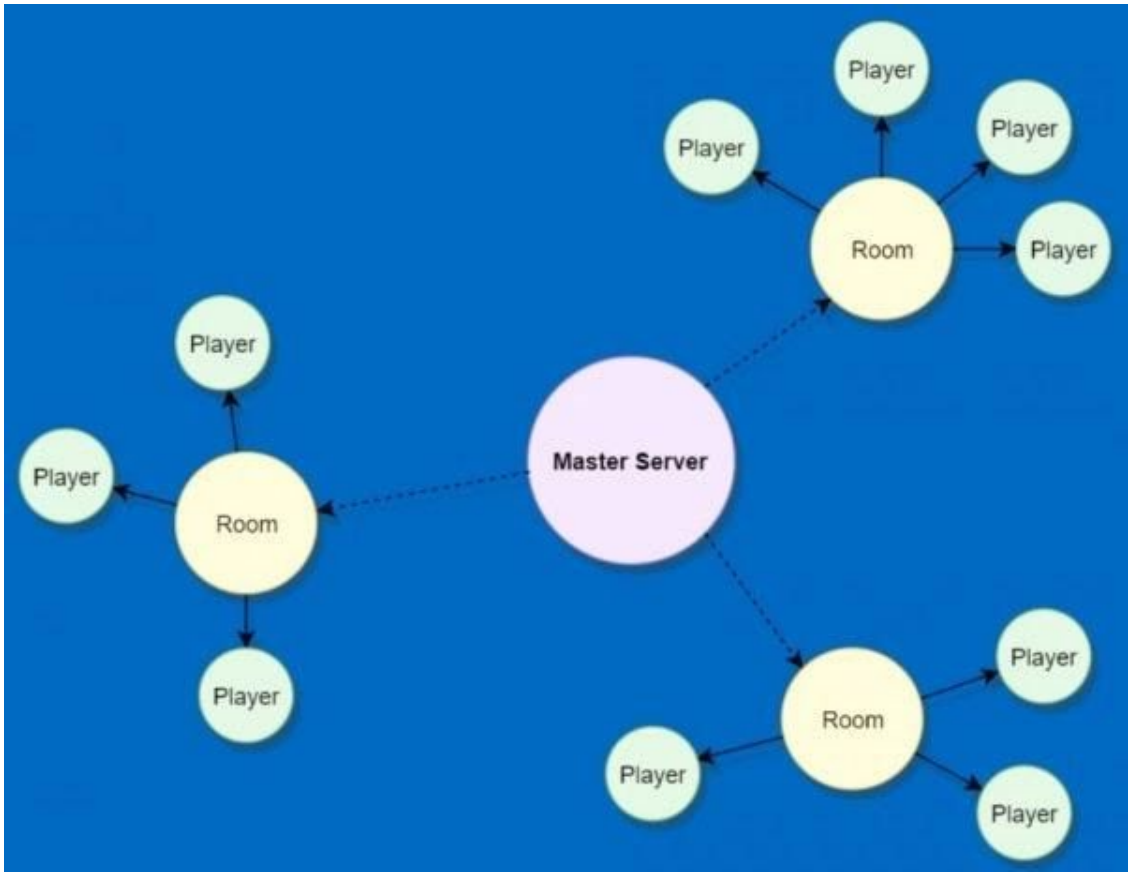

Figure 7: Master Server

3.USE CASE REALIZATIONS

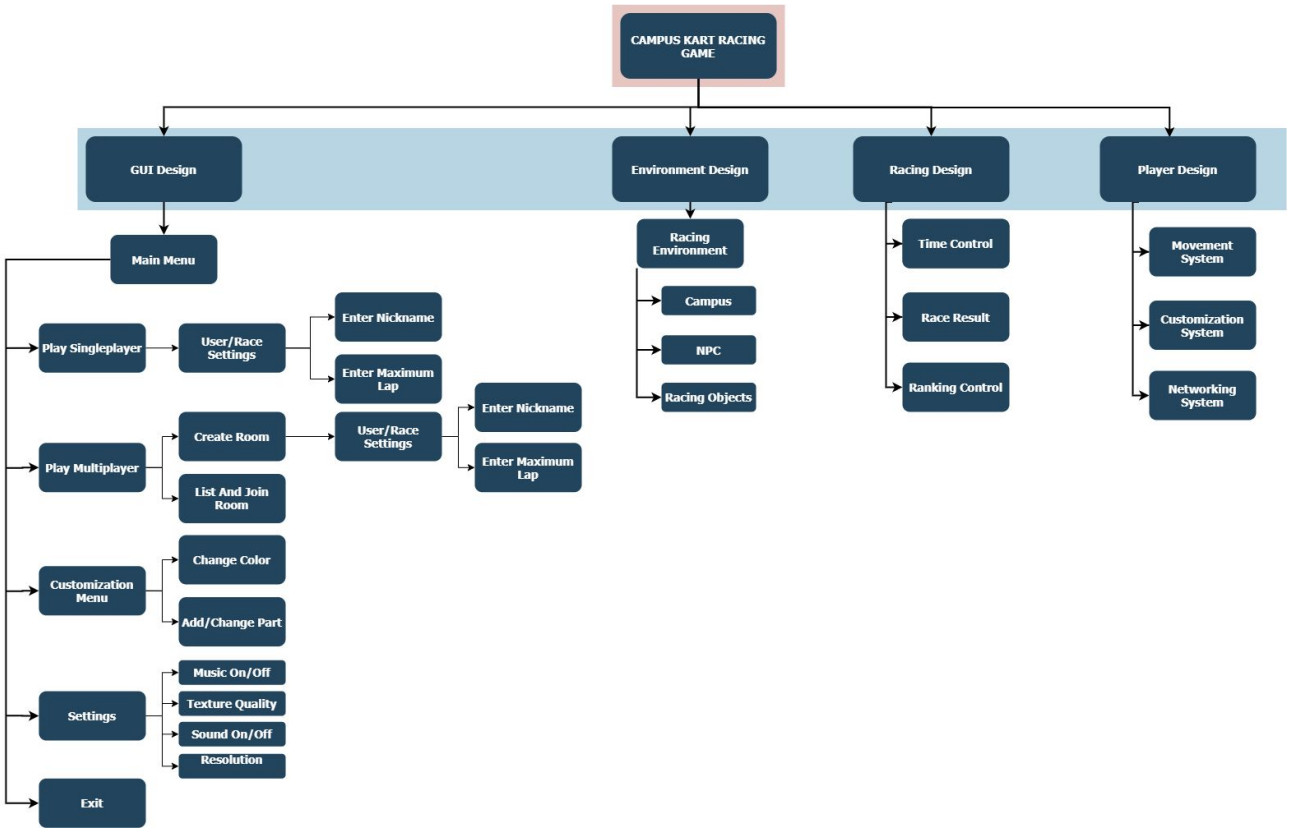

Figure 8: Project Components of Campus Kart Racing Game

2.3 Brief Description of Figure 6

The game consists of four main components. These are indicated in Figure 6.

2.3.1 GUI Design

GUI design is the main layer that provides the connection between the player and the game. The player uses the other components of the game with gui interaction.All players can access other systems from the main menu. The main menu is divided into five sub-menus. These are single player play, multiplayer play, customization, settings and exit buttons. The player can customize her vehicle in the customization system, and set her game to her own system on the settings screen.

2.3.2 Environment Design

Environment Design is the racing area where the race will take place. It includes the campus (Çankaya University) area, which is the main element of our game. It also includes many racing objects such as start / finish line, direction signs. Finally, if the player chooses to play in single player mode, it will also include NPC vehicles driven by AI.

2.3.3 Racing Design

Racing design is a system that includes the players' ranking, race times and race results.

2.3.4 Player Design

It is the section that contains the main mechanics of the player. Vehicle motion system, network system and customization systems are included in this field.

3 ENVIRONMENT

3.1 Modelling Environment

In this project, tools such as Blender and 3Ds Max were used to model the racing area. The polygon numbers of the models were tried to be kept as low as possible. Because the fewer polygons on the screen, the shorter it will take for the game engine to render the models. This means an increase in performance and FPS for the game. As mobile devices are not as powerful as computers today, it is very important to keep in-game performance as high as possible. We will use Blender and 3Ds Max modeling software to model a realistic environment such as cars, homes, barriers, map, level design, and other stuff.

In addition, another factor that makes a game look realistic is lighting. Skybox will be used for a realistic sky view in the game. Skybox method is to combine six different photos in skybox system to obtain a cuboid shape.

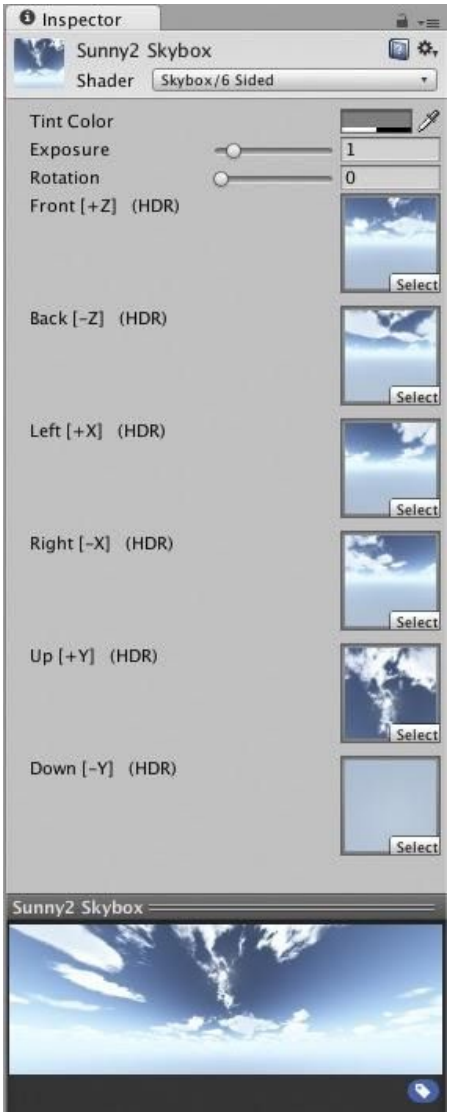

Figure 9: Example Skybox

Baked lighting system, which is better in terms of project performance, was used. Baked Lighting system, a light setting is made according to the positions of static (non-moving) objects in the scene and recorded according to these settings. In this way, the game engine memorizes the lighting and does not consume additional power for the lighting. This improves game performance.

The following objects are the design object samples from our game:

Figure10: Example Scene (Not in Unity Engine)

Figure: Tree objects

Figure 11: 3D Tree Objects

4 REFERENCES

[1] U. Technologies, “Unity User Manual (2019.4 LTS),” Unity. [Online]. Available: https://docs.unity3d.com/Manual/. [Accessed: 25-Dec-2020].

[2] “C Sharp (programming language),” *Wikipedia*, 04-Dec-2020. [Online]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Accessed: 25-Dec-2020].

[3] U. Technologies, “Vector3,” *Unity*. [Online]. Available: https://docs.unity3d.com/ScriptReference/Vector3.html. [Accessed: 25-Dec-2020].

[4] U. Technologies, “AnimationCurve,” *Unity*. [Online]. Available: https://docs.unity3d.com/ScriptReference/AnimationCurve.html. [Accessed: 25-Dec-2020].