**Software design document**

**Proof of concept**

**Tutorial**

**Author**: Stela Trencheva

**Student number**: 4520095

**Date**: 09.01.2023

Contents

[**1.** **Introduction** 3](#_Toc124691624)

[**2.** **Technical overview** 5](#_Toc124691625)

[**3.** **Technical details** 6](#_Toc124691626)

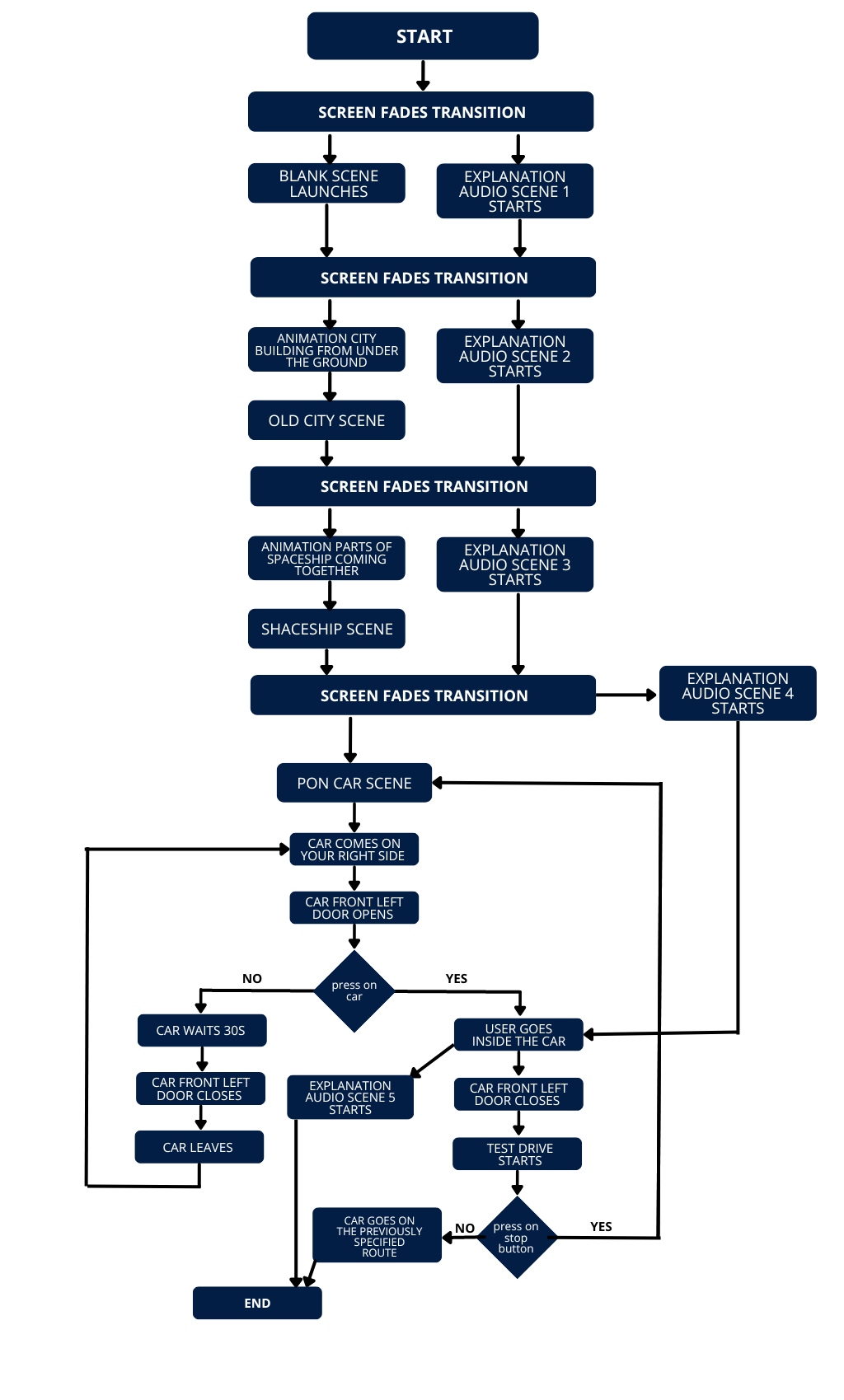
# **Introduction**

This proof of concept (POC) presents the idea of advertising in the metaverse through an interactive and self-explanatory tutorial. Based on positive feedback from a previous event with POC about the same concept, this experience has been retained for future use and further study. In the past, additional assistance from the team was required to help users fully understand the concept after experiencing it. Therefore, the main goal for this POC was to create a self-explanatory experience that does not require any other help and explanation.

The experience is taking place in the Oculus Quest VR Headset. It guides the user through the current settings of the headset and illustrates how, in the future, advertising can be integrated into the VR headset or metaverse by expanding upon these capabilities. During the whole time of the experience there is a voiceover that is explaining the things the user sees and interacts with.

In the first scene, the user is placed in an empty black environment facing an illustration of the Oculus Quest menu. Everything behind it is blank and dark. This is the homepage of the Oculus Quest without any interesting world as a background. In the next scene, the user is again facing the menu, but the background is an old city from a historical movie. The third scene, the user is inside a spaceship. During the whole time of transferring from one to another scene the voiceover is explaining that currently, it is possible to customise the background of your homepage in the Oculus Quest but the only thing that you can do is look at it and enjoy it. All the transitions are done smoothly by fading from scene to scene, building a city from underground or different parts coming from everywhere forming a spaceship. The last scene is an improved version of the POC PON from a previous event. There the voiceover is explaining how in the near future, the user will be offered an advertisement about something they are interested in and they will be able to interact and take part of the advertisement without even knowing it. The takeover advertisement in the metaverse will not be the annoying takeover advertisement that people are used to see over the websites, but it will be done in a gentle and nice way that brings value both to users and advertisers.

The POC builds upon the results and feedback from the previous event, incorporating improvements to the PON POC to enhance the overall experience.

* **Process diagram**

# **Technical overview**

The application consists of front-end and back-end part.

The frontend of Unity includes the Unity Editor, which was the main tool utilized to develop the application. The Unity Editor is a standalone application that operates on the Unity engine, and provides a vast array of features such as a visual scene editor, animation editor, game object inspector, and more. Additionally, it also grants the ability to expand its functionality by providing a scripting API that can be used to automate regular tasks and workflows.

The frontend part of the project consists of all the 3D assets that are modelled together in each of the scenes. All of the assets in this project are downloaded from the Unity assets stores due to the short time for development. The car in the last scene (PON experience) is the only asset that was modified in a 3D software – Blender. The interior of the car was changed. Some parts were removed and others were modelled so that the desired result is achieved. Then it was imported in Unity. The placement of the assets in the world/project in a good-looking way forms the frontend of the project.

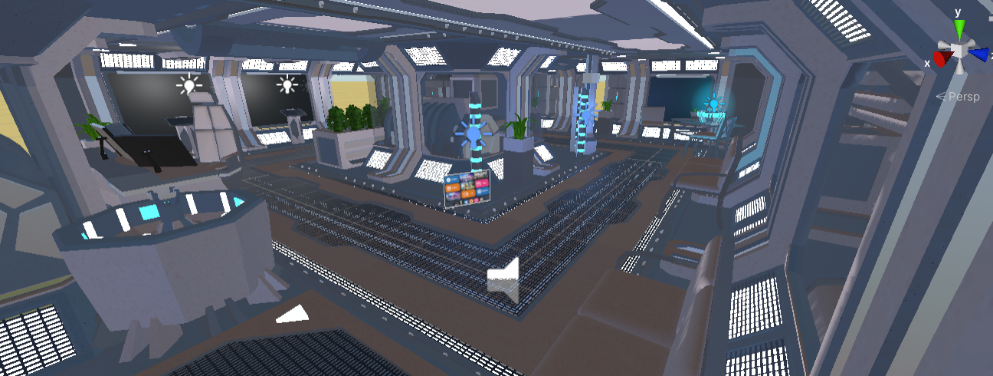
This backend was responsible for managing features such as rendering, physics, audio, and input. In this project the animations and audio play was implemented in the backend.

# **Technical details**

**Frontend** - The components in the frontend of this project are 3D assets downloaded from the Unity assets store or other platforms that offer 3D assets like Sketchfab. The menu of the Oculus Quest was created in Photoshop by me and imported into the project in Unity.

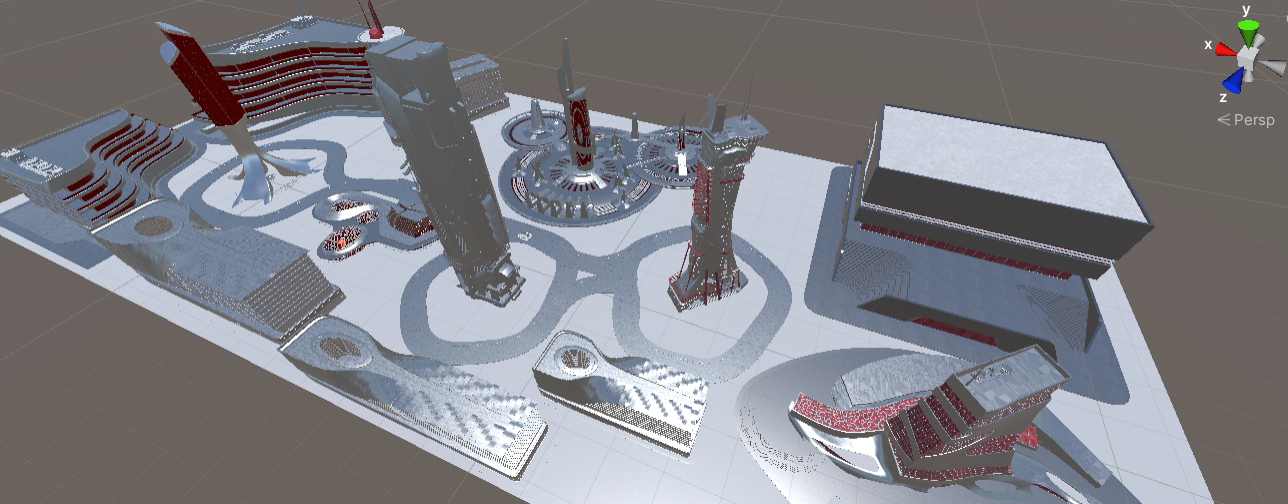
The first scene of the experience is a black environment. This was done by changing the material of the Skybox in the render settings of the project. A skybox in Unity is a feature that allows you to add a background to your 3D scene that appears to be infinitely far away. It is a six-sided cube that surrounds the camera and has a texture applied to each face, creating the illusion of a sky and environment around the scene. The skybox is rendered behind all other objects in the scene, giving the impression that it is far away and providing a sense of depth and realism to the scene.



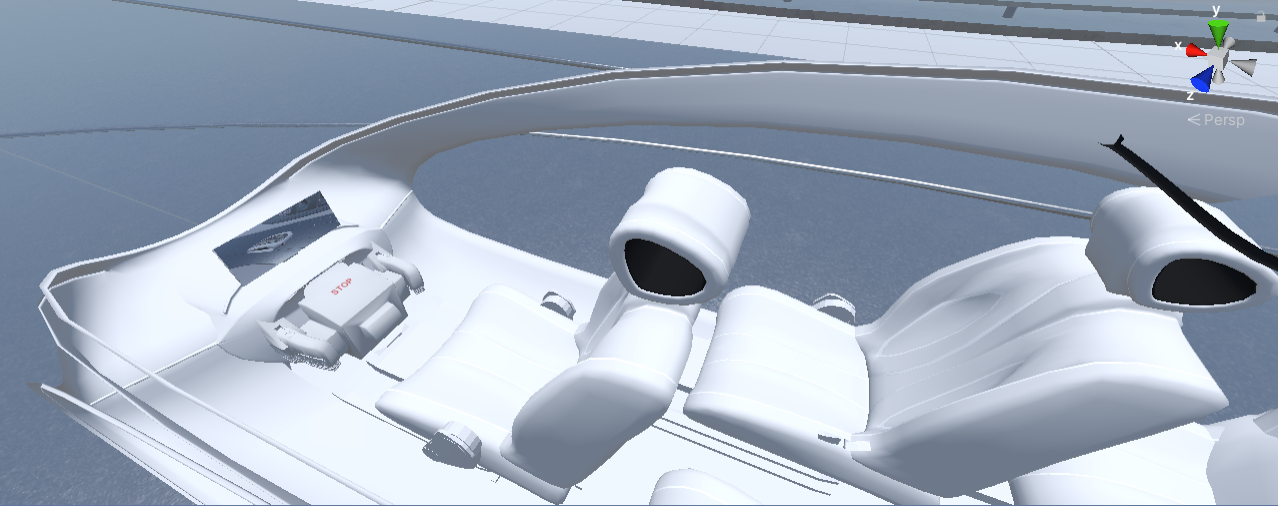
The assets of the historical city and the spaceship in the next two scenes were downloaded from 3D websites and placed properly inside the project.

Animations in the form of transitions from scene to scene were implemented. In Unity animations could be considered both part of the frontend and backend depending on how they were done. If the animations are created using Unity's animation system, which uses animation controllers and animator components, then the animations are considered part of the backend. These animations are controlled by scripts and are triggered by events or variables that are set in the backend. If the animations are created using Unity's particle system or Unity's UI system, then the animations are considered part of the frontend, as they are visual elements that are rendered and displayed to the user

The animations created in this project to change from screen to screen are fading animations. In order to create them, a part of the frontend was combined with triggering from the backend. In the frontend, a white canvas was created and placed in front of the camera of the user. When the experience is started, the canvas is not visible. The fading effect is done in the backend and will be explained later in the report.

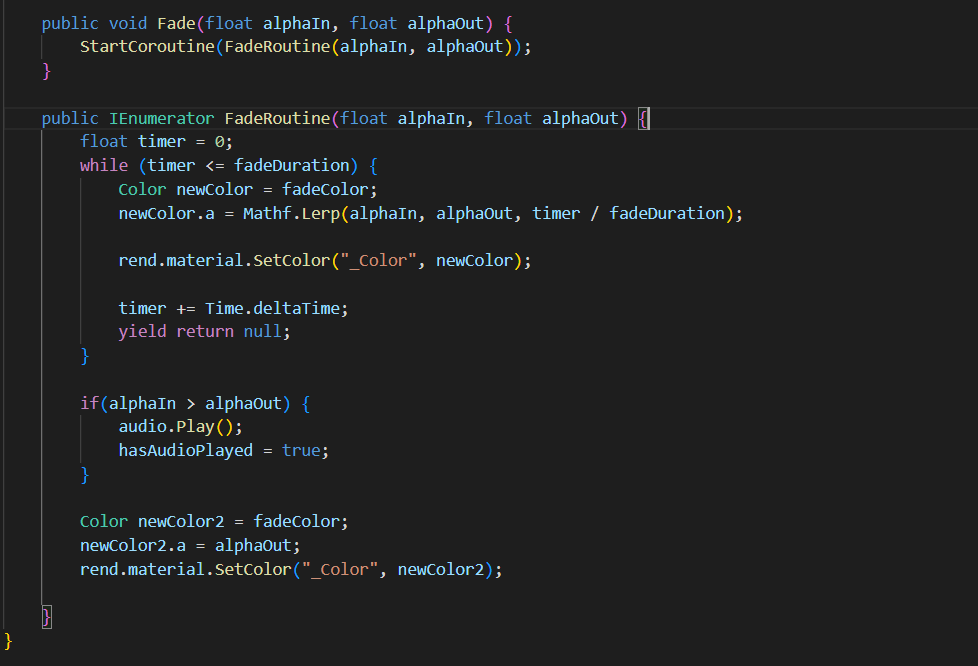
The city in the last scene of the tutorial that is part of the previous POC was saved as FBX 3D object and imported into another software - Substance 3D Painter that is used for adding textures and materials to 3D objects. Filmbox (FBX) extension was used because it is more advanced for example in comparison with OBJ and can store a lot of information about the 3D object. It is recognised by a lot of 3D software and is used in game engine software.

After adding materials suitable for the futuristic look of the city in Substance 3D Painter, the textures and the 3d object were exported as GLTF file which was then grouped in a GLB file. GLB is the binary version of GLTF and contain the same information as GLTF but it is stored in a binary format which makes it smaller and faster to load than GLTF. Due to the fact that the application that was created is a VR app, the size of the assets is of an importance and therefore this is the reason why this file format was chosen for the 3D object of the city.

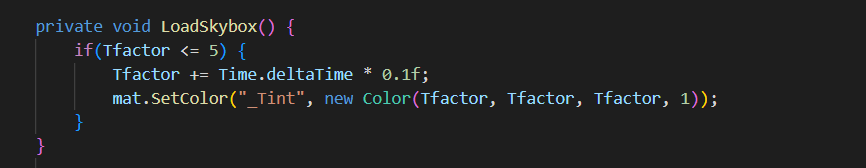
Another change to this POC was the interior of the car. It was refactored in Blender. While testing the PON POC with users, it was found out that some of them have difficulties seeing through the front left frame of the car. Therefore, it was decided to remove the two front seats and have just one that is placed in the middle. This intervention was done in Blender by me. In order for the car to look natural with this change, a change in the whole interior should have been done. The part that was dividing the two sides of the car in the middle was removed and another part in the front was modelled so that there is no hole in the front.

Another change to this scene is the lightning. Up until this moment a directional light had been used that is coming from above the car. However, this light was only going on the car and some parts of the city were dark. Therefore, the directional lightning was removed and environment lightning was turned on from the rendering setting. This type of lighting is illuminating the whole world.

**Backend -** The main part of the backend in this project besides the logic from the PON POC that was already created for the previous event, are the animations. All of them are executed by running methods from the backend.

The fading effect that is used in the transitions from scene to scene is created by creating classes *FadeScreen.cs* and *TransitionsManager.cs.* These classes are used in each of the scenes to manage the transitions and fading effects. In order to do the fading effect a method FadeRoutine(float alphaIn, float alphaOut) is implemented. It is getting two values that tell whether there will be fading in or out. Coroutine() method that allows you to perform a set of instructions over multiple frames was used to perform the action of fading due to the fact that it takes a long time to complete. Coroutines are implemented using the *IEnumerator* interface and the *yield* keyword. MathLerp(float a, float b, float t) is a method in *Unity's Mathf* class that is used to interpolate between two values, a and b, based on a third value, t. t is a value between 0 and 1 that represents the proportion of the interpolation. In this method FadeRoutine, the changing of color is implemented until the *fadeDuration* time is not reached. In this way the color of the screen in front of the user’s camera is being changed gradually making the effect of fading.

Other animations like the city in scene 2 that is building up from the group and the flying parts in scene 3 that are forming the spaceship are done in the animation editor that Unity provides. On a couple of timeframes, the position of the animated objects is changed so that when executed the animation runs smoothly. These animations are played once on the start of the scene.

The changing of skyboxes is another animation done in the backend. Depending on the scene, different skies are used as *Skybox* materials. The skies materials were downloaded from the Unity assets store. In the class where fading of the screen in front of the camera is done, the trigger of changing the skybox is also called. On each of the scenes the initial material of the skybox is set to black. When started and if the scene is different than the first (where the skybox needs to be black), the material is gradually changed to the desired one.

The playing of audios corresponding to different scenes was also implemented by checking which is the current scene. In the last scene, there are two audio sounds. One is played on start and one is played only when the used has interacted with the car. During the whole time of the tutorial, there is a background music playing.

In the last scene of the PON POC, a couple of things in the backend were changed. After testing the POC with users on the previous event, one of the things that was noticed was that the time of 10 seconds when the car opens doors and waits for the user to interact with it, is not enough for the users to understand what they have to do. Therefore, this time was changed to 30 seconds. Another thing was that a lot of people get motion sick while doing the test drive of the car. Except the fact that most of the time it is normal to get motion sick in VR, some changed were implemented to the path of the car so that it runs more smoothly than before. The location of the nodes was changed but in order to not check how it works with try and error which would have consumed a lot of time, unit tests were implanted that test how the car moves with different locations of the nodes and also finds the best variant.