Fachhochschule
Nordwestschweiz

Project track IP6
Real-time synchronization with an immersive, digital showroom

Submitted by:
Timon Tschanz and Timon Keller

Submitted to:
Prof. Dr. Hilko Cords
Dr. Marcel Gygli
Dr. sc. nat. Stefan Arizona

Client:
Pick-up-and-go GmbH / Inside Reality

Computer Science Course (iCompetence) HS21
at the School of Engineering, University of Applied Sciences
North-western Switzerland

Friday, 19. August 2022

Abstract

Marketing and selling a very technical product through a simple online store is no longer the option these days. Therefore, this thesis has dealt with the question if it is commercially worthwhile to develop a digital twin between a digital showroom and an immersive showroom.

The following technologies were used: Firebase for the web solution interface and Unity for the digital showroom. The products marketed in the digital showroom are the CapTop from pick-up-and-go GmbH, and the technology for the immersive showroom came from Inside Reality AG.

It was investigated whether Firebase could be used to generate an interface between the immersive and the digital experience in real-time. In addition, we looked at how such a digital showroom compares to an actual showroom. The thesis showed that Firebase is a perfect option to create such an interface and that this interface can also be very well maintained and adapted to various situations. Furthermore, this solution could be scaled very well, and we have established a proof of concept that this problem can be solved very well.

1. Introduction

Technical products and products that are not so visually but very convincing in the technical field, for these products it is not easy to use an online store as a good sales portal. Selling products that are not self-explanatory over the Internet is a challenge in this day and age. Our task in this project was to find a solution to offer customers a solution to explore such technical products in an online experience. The ability to build a virtual showroom into an online store, release it in real-time, and make it an immersive and virtual experience would create that opportunity and reinvent the sale of very technical and complex products.

This goal posed the following problems we want to solve in this project:

- Creating a digital twin of a store
- Marketing products virtually in this Showroom
- Integrating that showroom into an immersive and virtual real-time experience
- Making it commercially worthwhile

Much has already been reported and researched about the unique technologies we are trying to combine in this project. The individual technologies or tools are market leaders in their field or are at the forefront of their development. There are already approaches where these technologies are used individually in different ways for similar use cases, but never in combination with each other and certainly not in real-time and in the approach of a digital twin in the different "worlds." The given is not enough for our project because we want to combine the techniques to achieve the desired result.

In plain language, we need a way to store and retrieve data such as the camera's position in real-time. So, we need a so-called interface that we can call up via a website, a database, or something similar, and that we can call up from anywhere in the world and display the same data in the immersive, digital, and virtual showroom all in real-time.
That is the only way we can offer the customer the ability to enter that experience from anywhere in the world, whether through the digital immersive or the virtual showroom, and offer someone else the ability to have that displayed in real-time in their showroom which the former does.

As we will use it in this project, the immersive showroom is available in different versions. Inside-Reality uses three beamers, each projecting an image onto the opposite wall. These three images are projected so that the computers to which these beamers are connected calculate the image so that the overlaps are smooth. This means that the point where one projection ends and the other begins is so fluid that the naked eye cannot see it—more about this under point 3.1.2 Inside-Reality.

As mentioned in a previous section, with this project, we want to develop the ability to provide an interface that our customers can access from their different technologies, connecting their worlds in real-time. The technologies and products involved in this project are.

- Unity
- The CabTop of pick-up-and-go
- The immersive showroom of Inside Reality
- Firebase as a web solution

In practice, we have developed an interface that can be accessed from our digital showroom, with a script to push and pull data. So, one can imagine user A sitting in Australia looking at our client's products in an online digital showroom. Salesperson B can connect with Customer A and experience what Customer A is seeing in real-time in an immersive showroom in Switzerland and explain it in more detail via a phone call or other means as seen in Figure 1 Sketch from scenario. From a technical perspective, here is what happens. While Customer A is looking around the digital showroom, data from the Unity showroom hidden behind it is pushed to our interface. This is data such as camera position, viewing angle, and more. Vendor B or the immersive showroom now fetches the data pushed

by customer A in real time and now moves the first-person viewer to the same position and aims the camera at the same angle.

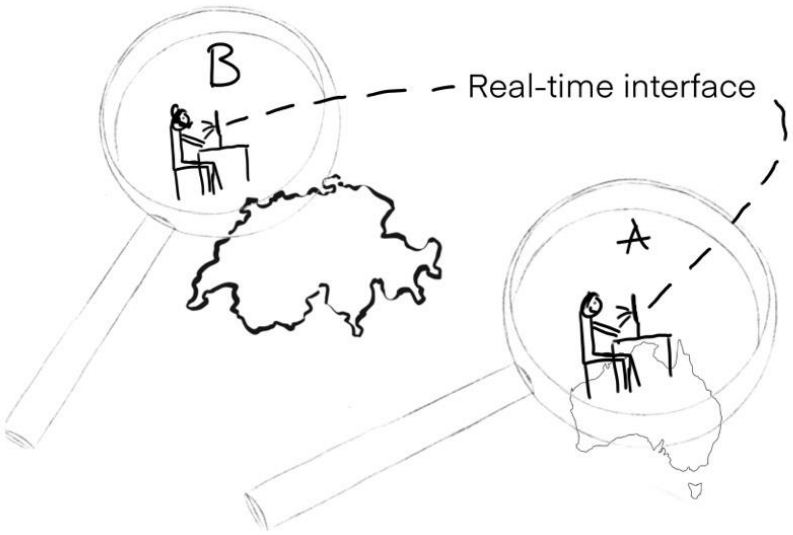

Figure 1 Sketch from scenario

At the end of this work, we delivered a fully functional and implemented prototype for a digital showroom in Unity. The prototype provided customers with an experience where users could experience and discover the products with animations, and the products were presented in different environments. With the showroom prototype, we also delivered a prototype of our interface that could capture and display the showroom data in real-time. In addition, we submitted this paper, which focused on creating a digital twin of the showroom and the capabilities of such an interface. The documentation in this paper explained and justified our approach and described our observations. Finally, we conclude by going into detail about our findings, making recommendations based on our findings, and discussing what might need to be done differently.

2. Background

In this chapter, we would like to discuss the various topics on which our project is based, which we have developed in our research, which we will return to again and again in this document.

2.1. Digital / Virtual Showroom

A digital or virtual showroom is nothing more than a showroom where one can appealingly present products and introduce them to customers. In a showroom, potential customers can take a thorough look at a product before making a purchase decision and receive advice accordingly. The advantages are easy to explain: Virtual showrooms depend on time and place. They can be visited from anywhere at any time, provided, of course, that one has an internet connection. Physical showrooms are increasingly being replaced by virtual showrooms, whether for cost, space, or other reasons that a physical showroom does not have. One can access most virtual showrooms directly via a web browser or with VR glasses. Another great advantage is that products can often be modified and adapted directly in the showroom. In addition, animations can be inserted so that it comes as close as possible to reality.
Today, many companies like IKEA or several fashion labels, have already developed a virtual showroom for their products and are continuously improving and expanding it. Also, for example, the Swedish car manufacturer Volvo. They have created a showroom to display their different car models. The customers can choose the color, the rims, various options in the interior, etc., according to their needs. One can look around the interior, open doors, or even switch on lights. [1] [2]

Figure 2 IKEA Virtual Showroom


Figure 3 Volvo digital showroom

2.2. Three.js

Three.js is a JavaScript-based WebGL library known for rendering 3D scenes in the browser. With three.js, one can create realistic 3D scenes without a compelling gaming computer, which can be integrated into websites to give them an artistic look. Ricardo Cabello launched Three.js on GitHub in April 2010. In our project, three.js is used for the immersive showroom of Inside Reality. [3] [4]

2.3. Caves

Cave Automatic Virtual Environment is a virtual reality environment where the walls and part of the floor and ceiling are projection screens as seen in Figure 4 Cave Automatic Virtual Environment in practice. One can imagine the room like a cube, where one is in the middle and the screens around one form a picture. So, one feels like one is in the projected world without virtual reality glasses. In some cases, one can also put on stereoscopic eyewear, which makes specific models appear 3D in the cube. This way, one combines the 2D space of the walls with the 3D space of the models and gets depth in a 2d environment. In addition, one can use data gloves, joystick, or other input devices to intervene in the experience and, for example, move forward. The Cave comes to us in the project in the form of the immersive showroom of Inside-Reality. [5] [6]

Figure 4 Cave Automatic Virtual Environment in practice

2.4. Engines

We used a game engine in this project to create a digital showroom. Game engines are used in most applications for producing games. However, these game engines can also be used in many other ways, such as creating a digital showroom.

2.4.1. Unreal Engine

Unreal Engine, like the other two engines, is a game engine and was first demonstrated in 1998 in the first-person shooter game unreal. Unreal differs from Unity, however, in that it is more commonly used to create more giant games, as some users point out that Unreal makes for more beautiful graphics. [7] [8] [9] [10]

2.4.2. CryEngine

CryEngine, like the other two, is a game engine. CryEngine was built by the development corporation Crytek. Games like Far Cry were created in this game engine, which shows how well this engine performs in the market. CryEngine is on the same level as unreal in graphics, so it outshines unity.

However, compared to unity and Unreal, it is a bit more complicated and not best suited for beginners. [11] [12] [13]

2.4.3. Unity

Unity is a free runtime and development environment mainly for PC, console, mobile and web browser games. It is one of the most widely used game engines in the world, especially for mobile games. However, Unity can also be used to create interactive 2D or 3D graphics applications and simulations. It allows one to create almost any object and any environment according to one's ideas. These objects can be equipped and moved with scripts and physical forces. Unity can also be extended via various plug-ins to include even more functions and objects. These can be easily downloaded via browser in the unity asset store or directly in Unity with the Package Manager. Ready-made models, figures, materials to create a scene are also available from the asset store. Unity is therefore ideal for creating the digital showroom envisaged for this project. [14] [15] [16]

2.4.3.1. Skybox

The skybox in unity can be thought of as a 6-sided cube. This cube is drawn behind every Object and contains a blue background with a sun as default as in Figure 5 Skybox. However, one can define what is displayed in the skybox. All that needs to be done is to insert a suitable HDRI, change it to a skybox cube and drag it into the skybox. [17]

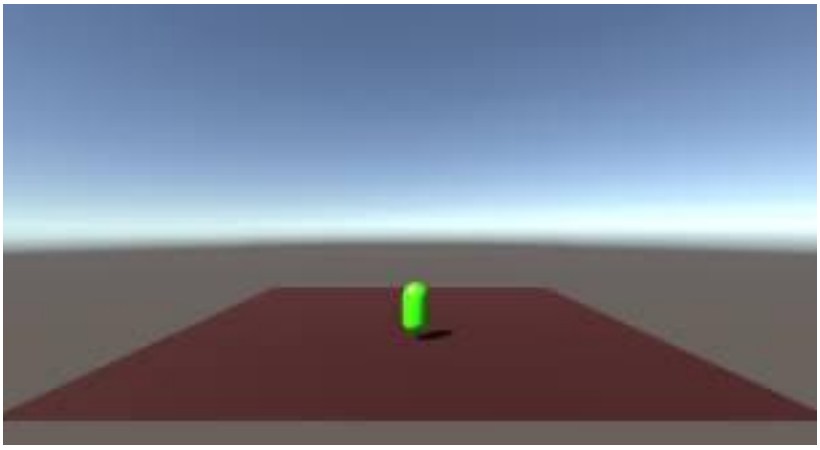

Figure 5 Skybox

2.4.3.2. Sky & Fog volume

Volumes are used in Unity to add fog to a scene or to change the colour of a scene, as seen in Figure 6 Unity volumes in use. In addition, volumes allow the creator of a scene in Unity to add different moods to a scene. Since we changed our project to HDRP, we could not use a Skybox as described in 2.4.3.1 Skybox, but we had to use the sky volume to insert our 360-degree image. [18]

Figure 6 Unity volumes in use

2.4.3.3. Unity packages

Unity packages are collections of files and data or entire elements of a Unity project. They can be easily compressed and saved as one file. For example, the downloaded assets from the Asset Store are saved as packages. Editor tools or runtime tools (graphics pipelines) are also stored as packages. Packages are therefore the key to creating and extending an advanced Unity project. To manage the packages, Unity provides the Package Manager. With the help of the package manager, one can see the packages already installed, add new packages and it gives suggestions for packages that work well together to create the best possible project setup. [19]

2.4.4. Render Pipelines

In Unity, there are four so-called render pipelines by default. Each of these pipelines has unique properties and performance characteristics. One chooses a suitable pipeline depending on the type of game, scene, application, or platform one is using. These pipelines are used to perform a series of operations before a scene is displayed on a screen. In summary, these are operations such as "culling" (discarding/cutting off objects that are not visible), rendering (the process of creating an image based on three-dimensional data contained in computer memory), and post-processing (merging all effects such as light, shadows, surface gradients, etc.). The pipelines described above are called: [16]

- Built-in Render Pipeline: Unity's standard render pipeline for general purposes.
- Universal Render Pipeline: Scriptable pipeline with quick and simple customization and optimizes graphics for a wide range of platforms.
- High-Definition Render Pipeline: Scriptable pipeline for high-resolution rendering on high-end platforms and the one used in this project.
- Custom Render Pipeline: With the Scriptable Render Pipeline API it is possible to create oner own custom render pipeline. [16]

2.4.5. Scripts

In Unity it is possible to attach scripts to the individual components in a project. These scripts are written in the programming language C# (C sharp) by default and the code is executed during gameplay. Scripts can be used, for example, to control the behavior of an object, trigger certain actions or create reactions to user input.
Two of the most important and basic functions that can be inserted with such a script are the Start() and Update() function. The Start() function is activated before the first frame when the script is used successfully. In this way, for example, the connection to a database can be established before continuing with the actual activity. The Update() function, on the other hand, is called once per frame. This is important in this project in order to update the camera position coordinates. In addition, there are countless other functions etc. that can be used to customise a project according to requirements. The functions and how they were used in this project will be explained in detail in the coming chapters. [20]

2.4.6. Asset stores

An asset store is a library of countless assets ranging from models, textures, audio assets, tools and animations to entire projects and extension assets. Many assets are available for free, but there are also some that must be purchased. For example, if someone wants to create a scene or a game and a game character or a house is needed for it, developers can simply search for them in an asset store and download them. This way people who only want to work with Unity do not have to model each object individually and can put together the desired scene. There are many different asset stores available on the internet. These are constantly being expanded with new content, as practically anyone can make assets available to others. This way it is not necessary to model each object individually and it makes it easier to put together the desired game scene. Often game engines have their own asset store, for example Unreal Engine has the Unreal Marketplace or Unity has the Unity Asset Store. [21]

2.4.7. Materials

Materials in a game engine define the surface properties of the objects and other surfaces in a scene. The material is a type of paint that can be applied to objects to make them look more realistic. So, walls can be made of concrete, floors of wood or mechanical parts can be given a metallic look. Materials are often depicted on a sphere by default, as can be seen in the images below.

Figure 8 Wooden material

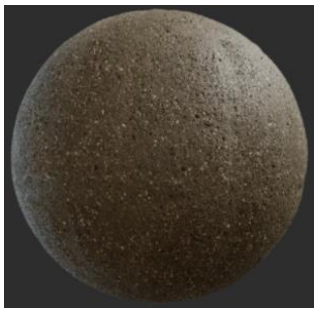

Figure 7 Concrete material

2.5. HDRI's

HDRI are High Dynamic Range Images, mostly 360-degree images, mainly used as background. These HDRI were taken in our case with a 360-degree camera in which every second a picture was taken, and all light tones to all dark tones were covered, and from all these images finally, a 360-degree image was put together. [22]

2.6. File formats for 3D scenes

There are several ways to export 3D scenes or models in other environments or for storage. However, since the .gltf or .glb format was recommended to us by Inside Reality, the decision was made quickly. A .glb file is basically a binary version of a .gltf file, with the same properties. The company has worked with this format in the past and it allows an easy integration into their software. GLFT stands for Graphics Language Transmission Format, and it is generally good for a fast, simple transmission and loading 3D scenes. Such a file has the advantages of a compact file size and fast loading times. Moreover, as complete 3D scenes, they are runtime-independent and expandable for further development.
In Unity there is an extension called "glft-Exporter". With this extension it is possible to export the created scenes directly from Unity to use them elsewhere. [23] [24] [23]

2.7. Firebase

Firebase is a Baas (backend-as-a-service) from the house of Google. Firebase supports developers with some services like a real-time database, also used in our project. Firebase is categorized as a NoSQL database, which means that Firebase stores the data in JSON-like documents and does not use a data manipulation language like an SQL database. In addition, Firebase provides services like authentication, the mentioned real-time database, hosting, test lab, and notifications. In principle, Firebase is specialized in mobile apps, but as in our case, Firebase can also be applied very well to web applications and other technologies such as Unity. [25] [26] [27]

3. Concept / Methodology

The points acquired in chapter 2 Background are now incorporated into the following subcategories and show how we have adapted them to our project. In the following subchapters, all requirements for our concept are covered and adapted to the individual components of our solution.

In the following storyboards we show how our concept should be applied and used in the work environment. In the first story board, we show how the game engine showroom functions without the Web solution. In the second, we include the Web solution and the immersive showroom. [28]

Figure 9 Storyboard 1 customer using digital showroom [28]

Figure 10 Storyboard 2 customer using digital and immersive experience [28]

3.1. Requirements

In order to plan and structure our project well and finally create a suitable concept, we have listed some requirements in the following paragraphs.

3.1.1. Pick-up-and-go

The company manufactures various interchangeable systems for pick-ups for work or leisure. The product we focus on is called "CabTop." This construction is especially suitable for excursions and overnight stays in nature. It consists of a large main cabin with a tailgate and two side flaps. This cabin has a cooker, a sink with a water tank, a battery, drawers, and much more storage space. In addition, there is a pull-out tent on the roof which is used for sleeping. The systems are nevertheless very flexible, because it only takes a few minutes to load and unload the construction onto the pick-up. The compactness and low weight support this change even more. These characteristics also apply to the other products of pick-up-and-go GmbH. [29]

Figure 11 CapTop from pick-up-and-go

As described, our customer's product is very technical and challenging to make palatable and bring closer to possible customers in a simple online store via pictures and simple texts. Therefore, the following requirements stand out:

- The customer must get to know the product through interactions and the possibility of seeing the product in 3D and action.
- The customer must be able to see the individual components of the product in detail and explore them in virtual use.

3.1.2. Inside-Reality

The company "inside-reality" specializes in creating so-called immersive rooms. This technology allows a group of individuals to experience a project or an application in an immersive way without VR glasses as seen in Figure 12 Product from Inside-Reality. It is intended to provide new perspectives in complex situations and facilitate group discussions with everybody in the same room. To present projects in an immersive room, only three beamers and one computer each are used for projection. Each projector displays a third of the situation on a wall, creating the entire experience together. The three beamers harmonize while moving around in the immersive room to create fluid, dynamic images.

Figure 12 Product from Inside-Reality

To include Inside-reality's immersive solution in our project we have established the following basic requirements.

- Our customer's technology must be able to access the data referred to by the game engine, such as the position data, the angle of view, and the positions of the components, in real-time.
- The data must match the game engine’s, to the last decimal place. [30]

3.1.3. Showroom

In point 2.1 Digital / Virtual Showroom, we have already dealt with what a digital showroom should look like and what it must offer. To apply this to our project, we defined the following requirements.

- The digital showroom must display our customer's pick-up solution described in 3.1.1 Pick-up-and-go in the center.
- The customer who wants to view the product in the digital showroom must be able to move freely around the showroom. This way, we guarantee the customer to explore the product from all sides.
- He must be able to control the camera and freely explore the given environment. This allows us to offer the customer an environment where he can view the product and thus get a more authentic feeling.
- The digital showroom should represent the pick-up-and-go workspace and a second environment that corresponds to a potential application environment of the product. This was very important to our customers because the customer feels much more connected to the product and pick-up-and-go.
- The customer must be able to explore the product through all interactions. The customer can get to know the product better through these interactions, and the "shopping" experience seems much more realistic.
- The product should be in proportion to the original and, if scaled, should only be scaled in numbers with one at the end or beginning. For example, a scale of 1:10 is allowed, but a scale of 1:15 is not, or a scale of 1:0.1 is allowed, but a scale of 1:0.2 is not. The customer must also be able to compare the product in connection with the other assets, and if the scale does not end with a 1, the product is no longer scaled correctly.
- The showroom must have an optic that is as real as possible. For this light, shadow, and natural objects must be considered and built in.

We have set up our showroom as follows, considering the abovementioned requirements. First, we designed two environments, the first one should reflect our client's Workplace, and the second one we created a forest look. The latter should allow the customer to view the product in a possible application environment. This way, we will enable the customer to get a better idea of the product and can already develop a more emotional connection to the product. Furthermore, we have decorated the two showrooms with assets so that the impression of a potential natural environment comes across better and the experience of experiencing the product increases.

Story telling

To bring the user closer to the product, and not only through the visual, but our goal is also for the user to experience a story in both environments that would not be possible in a regular online store.

In our case, the goal is that by trying to make the garage look like our customers, the user already understands the environment in which the product was built and designed and feels more connected to the product. In addition, by discovering the product in a potential future application, the customer will be more inspired and possibly motivated to buy the product than if they only see pictures of it. The goal was to carry out the customer through an experience he probably would have gone through in the real world. As mentioned above, he started in the pick-up-and-go workshops where he could design the product as it suited him. Then, he would have left the workshop with the product, tried it out in a potential use case, and got to know it better.

The points mentioned above allow the user to develop initial feelings towards the product.

3.1.4. Interface

To display our showroom simultaneously in the digital and the immersive showroom, we needed a real-time web solution that served as an interface to which the changes in position, viewpoint, and animation transformations were saved. Furthermore, this web solution should have served as an

interface that can be accessed from anywhere in the world and was able to update and display the data in real-time. Finally, the interface should have allowed the game engine and three.js to fetch data.

We go over the implementation in point 4 Implementation.

3.2. Engine

In order to develop a digital showroom, we had to agree on a game engine that we could use to implement the showroom. This decision was already heavily influenced by our customers, but we'll come back to that later.
The engine we decided on had to fulfill several constraints, which we have listed in the table below.

3.2.1. Constraints

Title	Description	Reasoning	Evaluation Questions
C.01 Real-Time	The engine must have a possibility to push the data of the first-person controller or similar in real time to a web solution.	To guarantee real time transmission, the data must be able to be pushed in real time.	How easy is it to use? How reliable is it? How fast is it?
C.02 Showroom possibilities	The engine must offer possibilities with which the creation of a showroom is possible.	The main feature of our project is the showroom, which is why this must be given. Otherwise, too much time is lost.	Has this option been used more than once? How easy is it to use?
C.03 Assets	The best case is an asset store or similar available in minimum must be able to import assets.	To make the showroom realistic and to display the product from the customer, assets must be available and importable.	Has this option been used more than once? How easy is it to use?

Table 1 Engine requirements with description, reasoning, and evaluation questions

We looked at the following engines and checked them against the above constraints.

3.2.2. Unreal Engine

Unreal Engine has with the latest release of the Unreal Engine 5 to the current state the probably most modern and best suited game engine brought out. The engine covers C.01 very well with the new release. Figure 13 Showroom Unreal Engine shows that Unreal is also a suitable option for designing a digital showroom of any kind and thus C.02 is also covered. Through the Unreal Engine Marketplace, C.03 is also very well covered. [10] [11] [9]

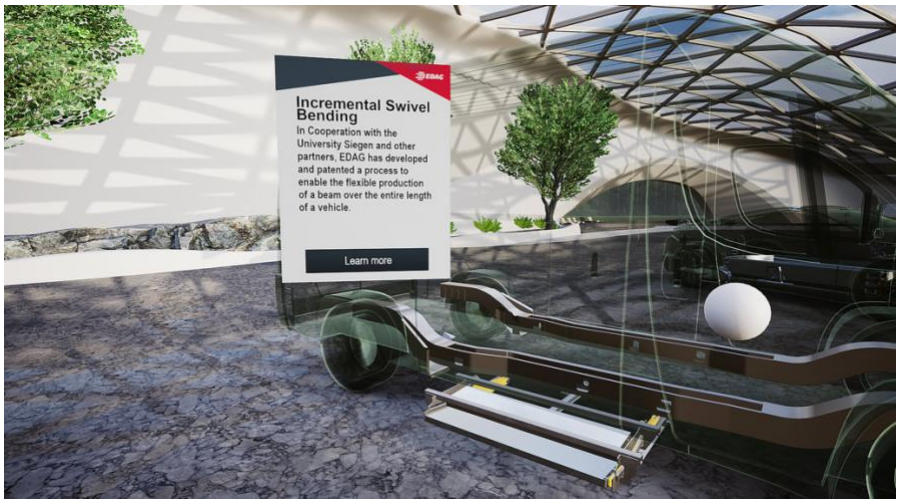

Figure 13 Showroom Unreal Engine [31]

3.2.3. Unity

Unity covers constraint C.01 entirely by the scripts described in 2.4.5 Scripts. Because of the Update() function, Unity exports two times per second. C.02 is covered by Unity since companies such as Volvo have already created their showroom in Unity, as described in 2.1 Digital / Virtual Showroom. C.02 covers Unity through the Unity Asset store. [14] [11]

3.2.4. CryEngine

Cry Engine covers constraint C.01 by exporting the data in CryEngine and Unity via the scripts. C.02 is covered in CryEngine as well, but we have not found a showroom example in the time we have been researching and therefore assume that there are no existing tools to help with this. Finally, C.03 is covered in CryEngine by the CryEngine asset Database. [12] [11]

3.2.5. Evaluation

Criteria	**Options**		
	Unreal Engine	Unity	CryEngine
C.01 Real-Time	2	2	1
C.02 Showroom possibilities	2	2	1
C.03 Assets	2	2	2
Total	6	6	4

Table 2 Evaluation of engine options (2: best, 1: acceptable, 0: worst) [7] [9] [11] [10]

As shown in the table above, Table 2 Evaluation of engine options (2: best, 1: acceptable, 0: worst) [7] [9] [11] [10], is Unity and Unreal Engine on par. Previously, however, we mentioned that our customers have already greatly affected this decision. In fact, our customers decided early in the project that we needed to work with Unity.

3.2.6. Decision

Unity was one of our technologies already defined at the beginning of the project. Unity was a requirement that came from our customers. This is also why we used Unity to create the digital showroom.

However, Unity also brings some advantages to our project. Unity offers the possibility to develop a simple primary showroom with basic knowledge by using some packages, which we have already discussed in point 2.4.3.3 Unity packages and below and try out the first features. In addition, Unity offers the possibility of using the HDRP pipeline from point 2.4.4 Render Pipelines to make the showroom look realistic and accurate to nature. Unity does not support the file formats in which our customer could export the CAD, but we were able to circumvent this through the pipeline described in

point 3.3 File format pipeline. The ability to create scripts in Unity C# is described in 2.4.5 Scripts also enabled us to capture and export the data generated by our First-Person Controller Minimal.

With these advantages, Unity is very well suited for our project and offers us the possibility to fulfill the requirements for our showroom of 3.1.3 Showroom and to represent the showroom as realistically as possible. [9] [11]

3.2.6.1. Structure

We have planned the structure of the showroom in unity as follows. The user starts in the simulated garage of our customer. There, he can modify the product and view it in the environment where it is also being processed and built-in in our customer's garage. In addition, he can let individual features drive out and back in through an animation or open and close flaps. The user can move freely in this garage and has a first-person view of the product, a pickup truck, and the simulated garage of our customer.

After the customer has assembled the product the way he wants, he can change the environment by clicking a button or by a similar interaction. So, after this action, he will find the product simply in a forest environment.
In this environment, he can no longer modify the product. All other possibilities like the animations are open to him here as well.

3.2.6.2. Realism

As mentioned above Unity allowed use to include the HDRP pipeline. The HDRP, as described in point 2.4.4 Render Pipelines, allows us to make the showroom look much more natural. Through this pipeline, the light and shadow in our showroom look much more realistic, and the objects we have placed in the showroom also give a shadow. Without this pipeline, the showroom would look much more unrealistic. The differences are clearly visible in the Figure 14 Standard vs. HDRP [32]. On the left is the showroom without the HDRP and on the right the same with the HDRP. There are clear differences in the light and shadow, and the terrain looks much more realistic. The magic behind HDRP is called deferred rendering. With deferred rendering, most of the heavy rendering is postponed to a later phase. The rendering is done in two passes. In the first pass, all possible geometric information from the objects in the scene is stored in the so-called G-buffer. In the second pass, the data previously stored in the G-buffer is used to calculate the lighting of the objects. The advantage of deferred rendering is that the scene geometry is decoupled from the lighting calculations. Each light is only calculated for the pixels it affects, which saves a lot of computing power, but it is still a high-definition rendering.

Figure 14 Standard vs. HDRP [32]

In addition, we have searched Unity's asset store for assets that already have a realistic touch. For example, we downloaded the package containing the trees we used to create the forest in the forest environment. We also chose a package to include grass in the environment bin. Also, through the terrain builder, we could create a natural setting with hills and valleys, which makes the showroom look similar to nature. Through this terrain builder, a standard in Unity, we could also load terrain-like materials from the asset store and incorporate them into the environment with so-called "brush tools." [16] [33] [32]

3.2.6.3. Animations

To make the storytelling even more substantial, we decided to provide the customer with some interactions in the form of animations. In addition, we decided with the customer pick-up-and-go that it is essential for the customer to be able to change some components of the product himself so that he develops a more vital interest.

So, we decided that the customer must have options such as opening and closing the flaps and drawers of the product. Further animations could be the opening of the sunshade or the decoupling of the entire CabTop from the pick-up, which would reinforce the potential of the product even more. Furthermore, we thought with the customer that it would be exciting to incorporate the components of virtual reality better and offer the customer the possibility to open and close the flaps and drawers through virtual reality glasses. We also wanted to incorporate the physical resistance to these changes.

3.3. File format pipeline

The company Pick-up-and-go GmbH draws and constructs its 3D models with the help of a CAD (Computer Aided Design) program. Therefore, they could provide us with their products in many different file formats. The task was to find out which format could be imported and displayed best in Unity. Instead of a long search, we tried to import the different file formats directly into Unity to see which was the cleanest. However, none of the file formats provided can be imported directly into Unity. Therefore, an intermediate solution is needed, which brings the files into a format supported by Unity.
A tip from one of our supervisors gave us the idea of importing a suitable format into Blender and then exporting it from Blender into a format suitable for Unity. Blender is free software used to create and edit existing 3D models. Blender allows it to import .stl files, a format provided to us by pick-up-and-go. So, the first step is to import the .stl file into Blender. The .stl format is a rather old and widely used format when working with CAD programs.
The next step was to find out which file formats Blender and Unity have in common. There are mainly two formats that come into question: .fbx and .obj. A .fbx file contains many properties and is compatible with many platforms. However, the first tests showed that the model was not displayed cleanly. Looking more closely, triangle-like structures instead of smooth surfaces are visible as seen in Figure 32 Bad quality CAD import. This is usually a sign that something was not done correctly during the import.
As a logical consequence, we switched to the .obj format. The .obj format is also a widespread standard format for 3D models and has many similarities with the .fbx format. The results were much better than before as seen in Figure 33 Good quality CAD import and met our requirements, so we finally decided on the .obj format. [24]

Figure 15 File format pipeline

3.4. Scenario

We came up with the following 3.6 Software Concept, firstly so that we could develop the software concept and then understand it better.

A potential customer of the pick-up-and-go GmbH product is very interested in the product and would love to review it, and the problem is that he lives in Australia and does not want to take a flight to Switzerland just for this. So, the customer joins the meeting in the digital showroom, which he shares with the salesperson of pick-up-and-go via a screen transmission. During the meeting, the customer moves through the showroom with the help of the company's mini controller and moves individual

product components through the animations. The customer's interest increases, and he wants to get a better explanation of the product in an immersive experience. The customer connects an immersive showroom in Australia with the database. The pick-up-and-go GmbH salesperson moves around the digital showroom, and these movements are transmitted via the real-time database directly to the immersive showroom, which the customer can see simultaneously.

3.5. Real time data storage

As mentioned in chapter 1 Introduction, in short, the goal is for two people in two different locations to see the showroom from the same perspective and position. For this to work, both clients access a database or something similar where data can also be synchronized quickly enough. This data store or database takes the coordinates of the camera position from one and displays the updated coordinates on the other. The whole process is supposed to happen in real time, of course. In order to make a wise decision, we have done some research and selected three options. We examined these three options more closely and evaluated them according to various criteria that are essential for our project. The detailed criteria and evaluation are described in section 4.1 Tech stack.

3.6. Software Concept

Based on the above requirements and in order to be able to fulfill the imagined scenario from the 3.4 Scenario, we have created the following concept.

As already mentioned, our concept consists of three axes. First is the digital showroom, created in Unity and extended by scripts in C#. In addition, the CAD files of our customers are integrated. Finally, we took special care to make the showroom as realistic as possible and to share the data with the web solution in real-time.

The second axis is the immersive showroom, which mirrors the virtual showroom. These are based on three.js. Three.js as described in 2.2 Three.js is the programming language used by our second customer to develop their product. We needed to make it as easy as possible for inside-reality to integrate the data. The goal was to provide one with a configuration or a JavaScript library that makes it very easy to include the data.

The third and probably most important pillar in this project is the interface. The plan was to use an HTML and JavaScript solution, with the possibility of a database also in mind. Here we put the most significant emphasis on the fact that the real-time transmission was given at any time because this is also one of the main aspects of this project.

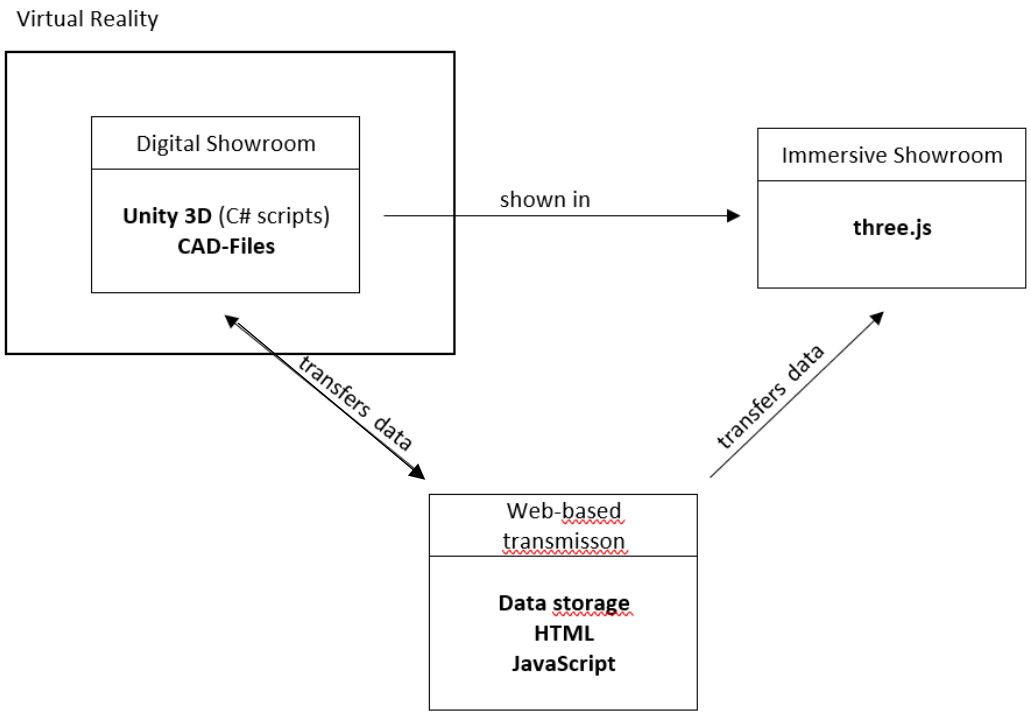

Figure 16 Software Concept

3.7. Discussion

From the beginning of the project, some details were already known about what was required of our project, what we should approach, and how. The two sides of pick-up-and-go and Unity were inside reality, and the immersive showroom, so three.js clearly showed what technologies were involved. The part of the web solution, the interface between the two sides, was still unclear and was the part we had to work on and do the research for. However, it was clear that both technologies had to be able to communicate with the web solution to create a real-time solution. In addition, the web solution should have been scalable since we did not know which data should be sent back and forth at that point.

We also researched how we could create the file format pipeline and how we should bring the CAD files from pick-up-and-go into Unity. So, after this point, we are ready for the 4 Implementation.

n

4. Implementation

In this chapter, we describe the tech stack we used for the web solution and why we chose the tech stack. Further, we describe the scripts in Unity, how the materials were used, and which animations we used.

4.1. Tech stack

A project's technology stack is the group of programming languages, frameworks, and libraries used for a project. In our case, we had a list of requirements and constraints that limited our choices. Based on these requirements, we investigated different options and selected the most suitable one.

4.1.1. Constraints

Title	Description	Reasoning	Evaluation Questions
C.01 Real-Time	The database or other solution must be a real-time solution. It should be a real-time database or at least support real-time.	To connect the showrooms and see the same behavior in all active rooms, the place where the data comes from and where the data is fed into must be in real-time.	How easy is it to use? How reliable is it? How fast is it?
C.02 Unity compatibility	Data can be fetched and pushed from Unity.	Unity is one of our essential technologies for this project, so there is no way around this criterion.	Has this option been used more than once? How easy is it to use?
C.03 JavaScript compatibility	Data can be fetched and pushed from JavaScript.	Three.js is one of our essential technologies for this project, so there is no way around this criterion.	Has this option been used more than once? How easy is it to use?
C.04 Scalability	The possibility of adding and removing more data is given.	Presumably, we only need a handful of data to be shared for this project. However, since this is very uncertain, it must be possible to expand and contract this data indefinitely.	How easy is it to perform CRUD operations?

Table 3 Tech stack requirements with description, reasoning, and evaluation questions

4.1.2. Options

We have selected the following options as possible variants.

4.1.2.1. Firebase

Firebase from Google is a possibility that solves three main problems. First, Firebase offers the ability to quickly create an app, share and monitor an app, add users, store data in a database, store data in a real-time database, and more. Second, Firebase covers the C.01 constraint with the real-time database. The requirements of C.02 are covered by Firebase, which provides a custom component for Unity. C.03 is covered because the Firebase real-time database already provides a JavaScript configuration. Finally, the C.04 constraint is covered because the firebase real-time database can be scaled to any request. [34] [35] [36]

4.1.2.2. Excel sheets

Excel is a spreadsheet program created by Microsoft. Excel does not cover the first requirement C.01 since the new release of the new Google sheets API. The C.02 constraint is covered since one can save the data in Excel. The C.03 constraint can be covered by the TableExport library. The scalability of Excel is given because it is possible to create almost infinite rows in Excel. [37]

4.1.2.3. Local File

By serializing data, Unity can be used to save the data to a local file on the computer. The constraint C.01 is covered because Unity allows to store the data in the update() function as described in 2.4.5 Scripts, which is stored twice per frame. The C.02 constraint is fulfilled; for this, only Unity scripts are needed. When the data is saved in the local file in the same folder as the JavaScript code, it is theoretically also accessible. The problem is since in requirement C.01, in each frame, approximately two new files are created. The file must be declared, from which the data must be pulled always again in the JavaScript file. The problem can only be avoided by overwriting the file each time. C.04 the scalability is given in any case because the file can be scaled as desired, but C.01 suffers from this. [38] [39]

4.1.3. Evaluation

We have rated these three options according to how well they meet the requirements. The rating questions in 4.1.1 Constraints were used to determine "how well" a tech stack option met a requirement. We ranked the options from first to third, with the first option meeting the requirements "best" and thus receiving 2 points for that requirement, while the third option met the requirements "worst" and thus received 0 points for that requirement.

Criteria	**Options**		
	Firebase	Excel	Local File
C.01 Real-Time	2	0	1
C.02 Unity compatibility	2	1	2
C.03 JavaScript compatibility	2	1	1
C.04 Scalability	2	2	2
Total	8	4	6

Table 4 Evaluation of tech stack options (2: best, 1: acceptable, 0: worst)

As shown in the table above, Table 4 Evaluation of tech stack options (2: best, 1: acceptable, 0: worst), Firebase is the most suitable for us with eight points. In addition, Firebase has excellent documentation online with which we can read up on the topic and quickly connect Unity with Firebase.

4.2. Concept

From the tech stack defined above, we developed the following concept. We set up a database and connected it to Unity. In addition, we created scripts in Unity that push the data, such as the position of the viewpoints and the positions of the animated objects, to the database and fetch from it when necessary. The connection from the digital showrooms to the Firebase is bidirectional, the connection to the immersive showroom is unidirectional. The React app only displays the data mentioned and is mainly used for testing purposes. We also developed a JavaScript variant for fetching the data. After researching a suitable tech stack, we planned and implemented the above developments according to the specifications defined in the project contract.

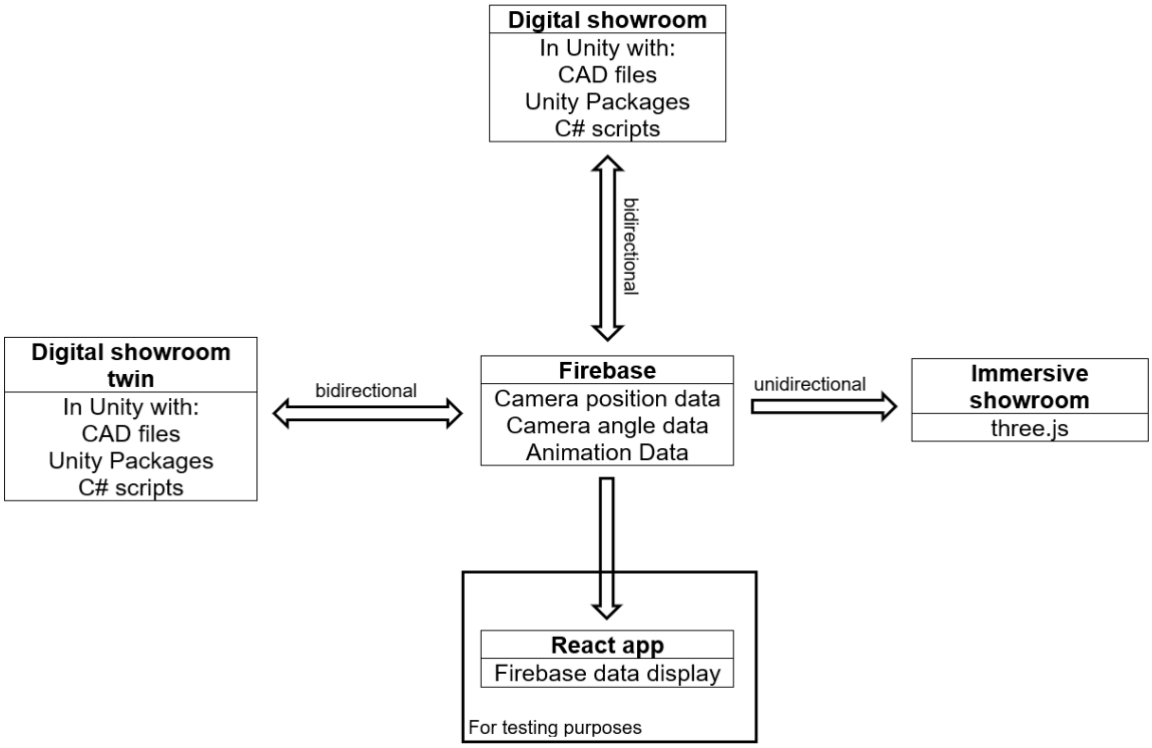

Figure 17 Detailed software concept

4.3. Firebase

As we learned above, Firebase is the most optimal solution for our project. So, after deciding to use Firebase, we created a project in Firebase and set up a real-time database. Then we used the configuration that Firebase provides for Unity to connect the real-time database to our Unity scripts and started pushing data to the database. Then we adapted the scripts provided by the first-person controller and were able to send the position and the camera angle of the user to the database.

Through Firebase, we were able to create the configuration for JavaScript, as shown in Figure 18 Firebase to JavaScript configuration. We used that to write a React app, which is shown in Figure 19 React App. We wrote this React app so that we could also check whether the changes we made in Unity were sent to the React app via Firebase. This configuration of Firebase we could then provide to our customer to write the data into their three.js program.

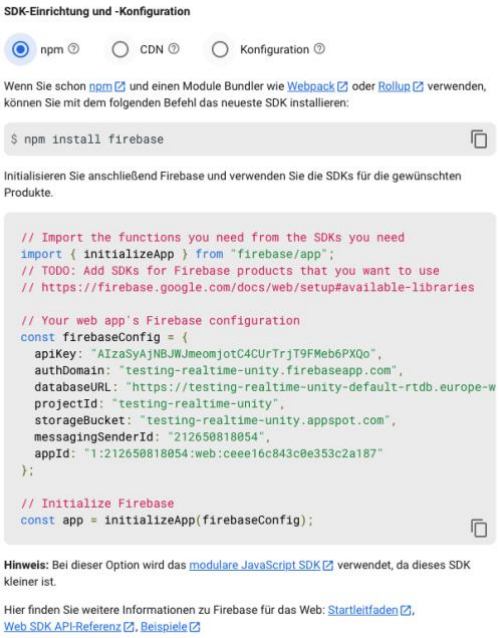

Figure 18 Firebase to JavaScript configuration

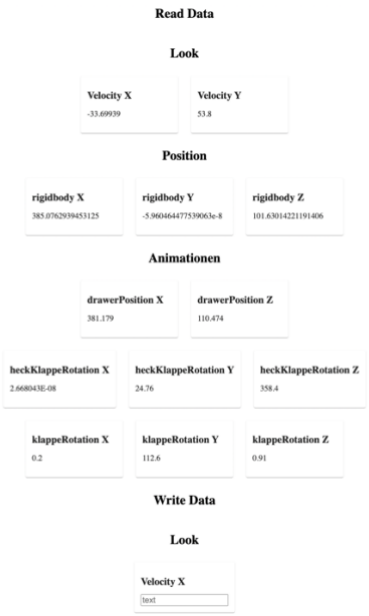

Figure 19 React App

In the end, we structured the Firebase real-time database to push the data into three groups, as shown in Figure 20 Firebase real-time database [34]. The first one is animation, in which we have stored the different positions that the different components of the pick-up-and-go product have. As an example, we can take the drawer which has in the database an "X," a "Y," and a "Z" position which the drawer also has in Unity. The other two groups are Position and Look, which have the position and viewpoint of the user or, more precisely, the first-person controller. [34]

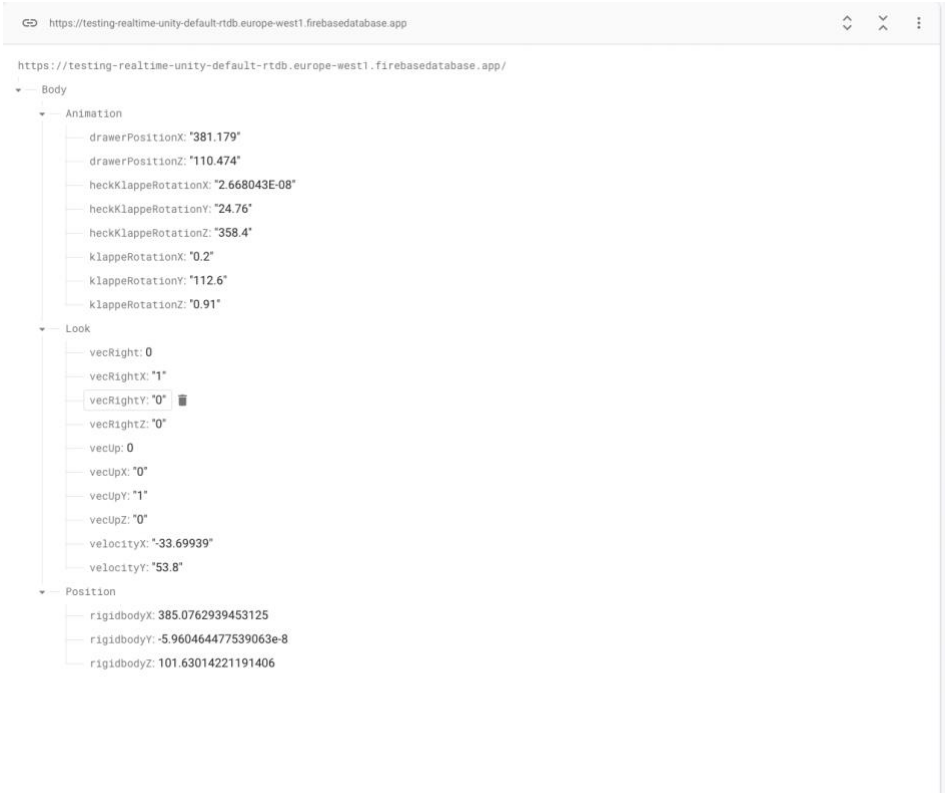

Figure 20 Firebase real-time database [34]

4.4. Data flow

The data flow in our project is realized by moving the Mini First Person-controller. This controller has a script for angle change and one for the position. We have adapted these scripts to listen to the event listener of the Space button, which we will discuss later, and send the changed data to the Firebase real-time database and listen for the changed data in the database when needed.

Unfortunately, we had the problem that the first-person controller took the absolute position and not the relative position when calculating. This meant that the camera movement in the second showroom was not the same as in the showroom in which the changes were made. To fix this, we took the position of the rigid body of the Mini First Person-controller and not the controller's camera so we could access the relative position and not the absolute one.

So, when a position is changed in the digital showroom, the position of the controller is split in “X”, “Y”, and “Z” directions and sent to Firebase as seen in Figure 21 Code snippet for pushing to Firebase. As mentioned above, the data is stored under the appropriate labels. If now a second showroom is added, one can listen to the changes from the database by pressing the space bar, and the Mini First Person controller will move to the changed position almost simultaneously with the change in the other showroom, as seen in Figure 22 Code snippet for pulling data from Firebase. If the immersive showroom is added, the Firebase database listens nonstop, and all the settings that affect the Mini First Person-controller are applied. Unfortunately, the animations in the immersive showroom do not work yet, since inside reality is not technically mature yet.

```
/* Getting the position of the rigidbody and storing it in a variable. */
rigidbodyX = GameObject.Find("First Person Controller Minimal").transform.position.x;
rigidbodyY = GameObject.Find("First Person Controller Minimal").transform.position.y;
rigidbodyZ = GameObject.Find("First Person Controller Minimal").transform.position.z;

/* Pushing the position of the rigidbody to the database. */
reference.Child("Body").Child("Position").Child("rigidbodyX").SetValueAsync(rigidbodyX);
reference.Child("Body").Child("Position").Child("rigidbodyY").SetValueAsync(rigidbodyY);
reference.Child("Body").Child("Position").Child("rigidbodyZ").SetValueAsync(rigidbodyZ);
```

Figure 21 Code snippet for pushing to Firebase

```
//Listen if spacebar is pressed
if(Input.GetKey(space))
   {

       /* Getting the value of the rigidbodyX from the database and storing it in a variable. */
       reference.Child("Body").Child("Position").Child("rigidbodyX").GetValueAsync().ContinueWith(task =>
       {
           if(task.IsCompleted)
           {
               DataSnapshot snapshot = task.Result;
               speicherX = snapshot.Value.ToString();

           }
           else
           {
               Debug.Log("not retrived");
           }

       });
       rigidbodyXkopie = float.Parse(speicherX);
```

Figure 22 Code snippet for pulling data from Firebase

We made the data flow between the different Unity showrooms bidirectional and unidirectional between the digital and the immersive showroom. Between the Unity showrooms, it was straightforward because we could use the same scripts and only had to put an event listener on the space button. By pressing the space bar, the first person Mini First Person-Controller no longer listens to the input it receives and sends data to the database but only listens to the real-time database and processes the data that the other user changes. As soon as the space key is released, the data adjusts himself again, and the controller listens to one's changes. We decided to use unidirectional communication between the digital and the immersive showroom because our customer, Inside-Reality, does not yet have an implementation that can deliver the same data that Unity needs to place the controller and adjust the position and viewing angles. We used a React app Figure 19 React App to ensure that the data changes made in the digital showroom arrived in a JavaScript solution. We ensured that Inside-Reality could use these configurations to incorporate the data into their immersive showroom. [40] [34] [41]

4.5. Showroom

4.5.1. Unity Packages

We used various packages from Unity in our digital showroom. We have described what a Unity package is in section 2.4.3.3 Unity packagesWe have integrated the following packages into our showroom:

Package	**Description and usage**
High-Definition RP	The High-Definition Render Pipeline is a scriptable high fidelity render pipeline developed by Unity. HDRP uses physically based lighting techniques, linear lighting, HDR lighting, and a configurable hybrid cluster deferred/forward lighting architecture. With HDRP, any Unity user can create very nice-looking games even if they do not have much experience. We applied this HDRP to make the showroom display more realistic. To use the HDRP pipeline, we installed the package via the HD Render Pipeline Wizard in the Configuration Checking and pressed the Fix All option to eliminate all problems with the configuration. After that, we had to upgrade all materials, which we did in the Render Pipeline menu again using the Render Pipeline Wizard. [33]
ProBuilder	With the ProBuilder package, it is possible to edit and texture custom geometry in Unity. For example, we used ProBuilder to create the first showroom. With this ProBuilder package, we could geometrically design the showroom the way we wanted. ProBuilder allowed us to draw the walls, floor, and ceiling of the showroom so that we knew the

	proportions exactly and could design the masses true to life. ProBuilder can be used simply by selecting ProBuilder from the Tools menu. [15]
Unity UI	Unity UI is a UI toolkit for developing user interfaces for games and applications. We used Unity UI to include the texts in the user's interface. To do this, we added text to the scene, placed it where we wanted and modified it. [42]
Visual Studio Code Editor	Code editor integration for supporting Visual Studio Code as code editor for Unity. Through this package we were able to edit the scripts in Visual Studio Code. Visual Studio Code is the text editor of choice. [43] [43]
UnityGLTF	Unity3D library for importing and exporting GLTF 2.0 assets. [44] Installed from [45] 4.6. Through this package, we could convert the scenes from Unity into the formats requested by inside-reality, namely the .glb format, which we have described in point 2.5 HDRI's HDRI are High Dynamic Range Images, mostly 360-degree images, mainly used as background. These HDRI were taken in our case with a 360-degree camera in which every second a picture was taken, and all light tones to all dark tones were covered, and from all these images finally, a 360-degree image was put together. [22] File formats for 3D scenes. [44]
Mini First Person-Controller	The Mini First Person-Controller is a package from the Asset Store. It makes it possible to view the showroom from a first-person perspective. With the controller, one can move around and discover the showroom or the product from different perspectives. [46]

Table 5 Packages used in Unity

4.6.1. Materials

The materials we used for our showroom where either from the asset store from Unity or imported from Poly Haven. Poly Haven is a website specialized on materials and HDRI's. The materials and HDRI's are free to use and thus ideal for our project. The process is very simple. One can select a suitable material in Poly Haven, download it, then drag and drop it into Unity. In Unity one must make sure that the material meets the requirements of the HDRP and then drag and drop it to the desired object. If the material does not meet the requirements of the HRDP, it is displayed in a magenta colour, and an upgrade must be done with the wizard so that it changes into the desired colours. [47]

For example, we needed a material we could use as asphalt for the road in the forest environment. In Poly Haven, we could easily find the asphalt material shown in Figure 23 Asphalt Material from Poly Haven under the "Road" subcategory. We downloaded this material and inserted it into Unity. Afterward, we selected the material with the Brush tool and "painted" it onto the terrain as shown in Figure 24. Since the cabtop of pick-up-and-go is not only made of one material, it was also important

to add these. Most of the parts are made of aluminum, which is why a light, metallic metal was chosen for them. The plastic parts are covered with black paint. [47]

Figure 23 Asphalt Material from Poly Haven

Figure 24 Asphalt material used in Unity showroom

4.6.2. HDRI's

To imitate the showroom of our client pick-up-and-go, we had two options that were there:

- Recreate the showroom through modeling
- To use an HDRI described in 2.5 HDRI's taken from the workshop.

Since the former would have been very time-consuming and probably used up this project's time, we decided to do the latter. We were lucky that Inside Reality had a 360-degree camera which we were allowed to use to capture the image from the Figure 25 360-degrees image from the pick-up-and-go workshop. Unfortunately, it turned out that we could not use this HDRI because we had placed the camera at the height of the head. Now, if Unity is set to HDRP, using a skybox is no longer possible. So, we had to use this HDRI as Sky and Fog Volume, which is how the image from the Figure 26 360-degree image in useoriginated. It is recognizable that the pick-up does not look natural, and unfortunately, the material could not be used that way. So, we agreed to concentrate on the Forest scene for the rest of this project.

Figure 25 360-degrees image from the pick-up-and-go workshop

Figure 26 360-degree image in use

4.6.3. Animations/Interactions

What was particularly important for pick-up-and-go was that the customer could view the product in such a way that he could also interact with the product and already develop initial emotions towards the product. Therefore, it was essential for us to present the product as it looks in real life and give the product some interactions that can be done with it in real life. This allows us to bring the product closer to the customer, and he feels more connected to the product if he can do something with it.

We have reduced the animations to the most important ones. So, the customer can click on one of the three keys, "1", "2", and or "3" to trigger one of three different animations. For example, the first key, the "1" allows the customer to drive out the kitchen drawer as seen in Figure 27 Code snippet for drawer animation. The "2" key allows the customer to raise the flap and the "3" key allows the customer to open the tailgate. We have implemented this in such a way that we have built in an event listener that listens for whether any of the three keypads have been pressed. When such an event occurs, the position of the tailgate is checked and changed to a position we left in the code. So, it always looks at which position is active and then changes to the other one.

In the beginning, we also had the intention that the customer could intervene in the event via virtual reality glasses, and we offered him the possibility to try out the physical forces and more. Unfortunately, due to time constraints, we could not include this, but we did go into it a bit more in the 5.4 Recommendations.

```
/* This is the code that is used to move the drawer back and forth. */
else{

    /* This is checking if the key 1 is pressed. If it is pressed, then the keyOneIsPressed is
    set to true. */
    if(Input.GetKeyDown(KeyCode.Alpha1)){
        keyOneIsPressed = true;
    }

    /* This is checking if the count is even and if the keyOneIsPressed is true. If it is true,
    then the drawer will move to the endPosition. */
    if((count % 2f == 0) && keyOneIsPressed){
        if(rb.position != endPosition) {
            Vector3 newPosition = Vector3.MoveTowards(rb.position, endPosition, movementSpeed);
            rb.MovePosition(newPosition);
            count++;
            keyOneIsPressed = false;
        }
    }
    /// <summary>
    /// If the player presses the key that corresponds to the first position, the player will
    /// move to the first position
    /// </summary>
    /// <param name="keyOneIsPressed">This is a boolean that is set to true when the player
    /// presses the key.</param>
    else if(keyOneIsPressed)
    {
        /* This is checking if the position of the drawer is not equal to the startPosition. If
        it is not equal, then the drawer will move to the startPosition. */
        if(rb.position != startPosition) {
            Vector3 newPosition = Vector3.MoveTowards(rb.position, startPosition, movementSpeed);
            rb.MovePosition(newPosition);
            count++;
            keyOneIsPressed = false;
        }


    }
    /* This is setting the value of the drawerPositionX and drawerPositionZ to the position of
    the drawer. */
    reference.Child("Body").Child("Animation").Child("drawerPositionX").SetValueAsync(rb.position.x.ToString());
    reference.Child("Body").Child("Animation").Child("drawerPositionZ").SetValueAsync(rb.position.z.ToString());
}
```

Figure 27 Code snippet for drawer animation

4.7. Discussion

We have completed the essential points of the implementation. The most important points were that when we change the Unity showroom, we will find these changes on the interface of the web solution and in the immersive showroom or additional Unity showrooms. The tech stack we have chosen is, in our opinion, the best choice for such a project after its completion. Firebase is perfectly designed to work with Unity and a JavaScript application. We also managed to create a forest environment as seen in Figure 28 Cabtop in forest environment scene and the workshop environment as seen in Figure 29 Captop in workshop environment scene in Unity, which allows the customer to experience the product in natural environments.

In the context of this project, we could not include any interactions that could be used in a virtual reality experience. For this, we had to change the animations to lifelike ones, not only position

changes. With Unity, the scripts with Time.deltaTime could be extended to make the whole showroom more realistic. Further recommendations we give, however, still later under 5.4 Recommendations.

Figure 28 Cabtop in forest environment scene

Figure 29 Captop in workshop environment scene

5. Validation and Discussion

Like any other software, our showroom is also tested to its requirements. This chapter describes the errors and problems that came to light during testing and the approaches used to solve them. We also ask how the digital showroom can replace an actual showroom or exhibition. Furthermore, the insights gained, and the goals achieved in the course of the project are described in this chapter. Finally, any extensions or improvements that could be made to the concept or implementation of the showroom are also discussed below.

5.1. Testing the showroom

What has already been tested with self-testing is the real-time synchronization between two desktops, which is actually the minimum goal. In this way, the camera position and the viewing direction are always the same and are constantly updated.
But initial tests have shown that the two views on the desktops are slightly offset from each other, so the camera positions do not match 100 per cent. This was very clear when comparing the two screens directly. The reason for this was that the values of the camera position were always calculated absolutely. This means that when moving in one showroom, the values in the other showroom are added to its current position. Due to the many decimal places and the partly rounded values, the more one moved in the showroom, the more differences there were in the camera position in both showrooms. To correct these offset camera positions, the values are recalculated relatively. The values are therefore always calculated from a fixed starting point, which means that the values are updated in the database and taken over exactly from both showrooms without any deviation.

Furthermore, the synchronization between a desktop and a showroom as an immersive showroom has been tested, which at the same time always confirms successful real-time database access.
To bring the entire showroom as an immersive experience with the hardware and software of inside reality, it is necessary to convert the showroom created in Unity into a .gltf or .glb file. With the Unity Package "UnityGLFT" described in section 2.6 File formats for 3D scenes it is possible to create the desired format. During testing, it was discovered that the export did not work 100%. This became apparent in the scene where the product is presented in nature or in a forest-like environment. The trees and other objects, which together create the forest, could not be displayed, which is one of the main problems at this point.

5.2. Comparison of a real and the digital showroom

The second chapter of this documentation already reported on the basic advantages of a digital showroom as seen in Figure 31 CapTop work in showroom compared to a real one as seen in Figure 30 CapTop explore in real life. Basically, these are the independence of place and time, which allow practically unrestricted access. Another is the configurability of the products, such as changing colours or different variations of the product with just a few clicks. However, what enhances the experience in the digital showroom are animations of drawers and flaps, for example, which demonstrate the functionalities of the product. Such small features have a great added value for customers to get a good impression of the product.
But there are also details that are difficult or impossible to show in the digital showroom. The big picture is recognisable, but not every detail, such as wiring, cables or general details that are more in the interior. If all the details were included, the files could be very large and it would take more resources to render them. In addition, the physical forces that are needed to pull out the drawers, for example, cannot be simulated. In this project, however, it is important to the client that he can demonstrate the most important functions three-dimensionally to his customers in the digital showroom and not just on a screen with photos.

Figure 30 CapTop explore in real life

Figure 31 CapTop work in showroom

5.3. Achievements

One of the goals of this project was to create a showroom for the company's product, clone it and synchronize it in real-time. A simple showroom was created using documentation, tutorials, and simple trial and error to achieve this goal. The list in Section 3.1.3 Showroom serves as a guide of the key requirements that the showroom should have. In this way, the showroom could be developed step by step. The result is two different environments where the product can be seen on a pick-up. Furthermore, the showroom was refined by using different packages and the Render Pipeline (3.6).

To find an optimal solution for real-time synchronization, we performed an evaluation 4.1.3 Evaluation. During the course of the project, it has been shown time and again that Firebase is an excellent database choice. With it, we can generate real-time camera data synchronization between two or more digital and immersive showrooms.

One part that was underestimated was importing the CAD data into Unity. Through research and some experimentation, we found an acceptable solution. Using the pipeline described in 3.3 File format pipeline, especially with the help of Blender software, it is possible to import the CAD objects cleanly into Unity. Without this pipeline, we had great difficulty importing the models into Unity at the beginning of the project; the quality of the import was poor, as shown in Figure 32 Bad quality CAD import. With the pipeline, we could eliminate this and have a good quality import, as shown in Figure

33 Good quality CAD import. The CAD model of the Cabtop product was initially imported as one large object. This made it impossible to apply different materials and colours to individual parts of the product, such as flaps and drawers.
Furthermore, only the entire object could be selected. This meant that the animations for the flaps and drawers could not be implemented. As a solution, pick-up-and-go GmbH provided us with the CAD files of the individual parts, such as side flaps, tailgate, and drawers. Thus, the parts can be individually equipped with materials, colours, and animations.

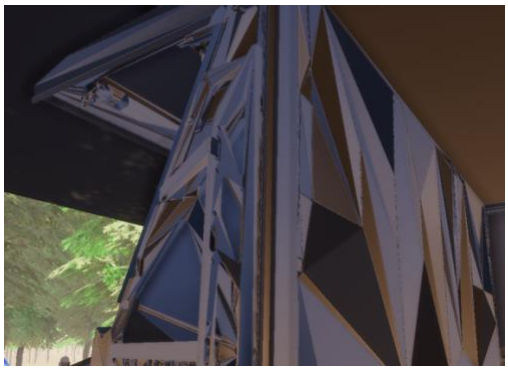

Figure 32 Bad quality CAD import

Figure 33 Good quality CAD import

With the support of inside reality, it has been shown that the showroom can also be presented as an immersive experience in an immersive showroom. The key to this result is the export and import with the 3D file format .gltf and .glb respectively. Despite the described complications during the testing 5.1 Testing the showroom, it could be ensured that the synchronisation of the camera position and thus the database access also works with this technology. Unfortunately, there were some minor problems: the terrain was not recognized correctly in the export and could not be imported into the immersive space see Figure 34 Current State of the Immersive Showroom.

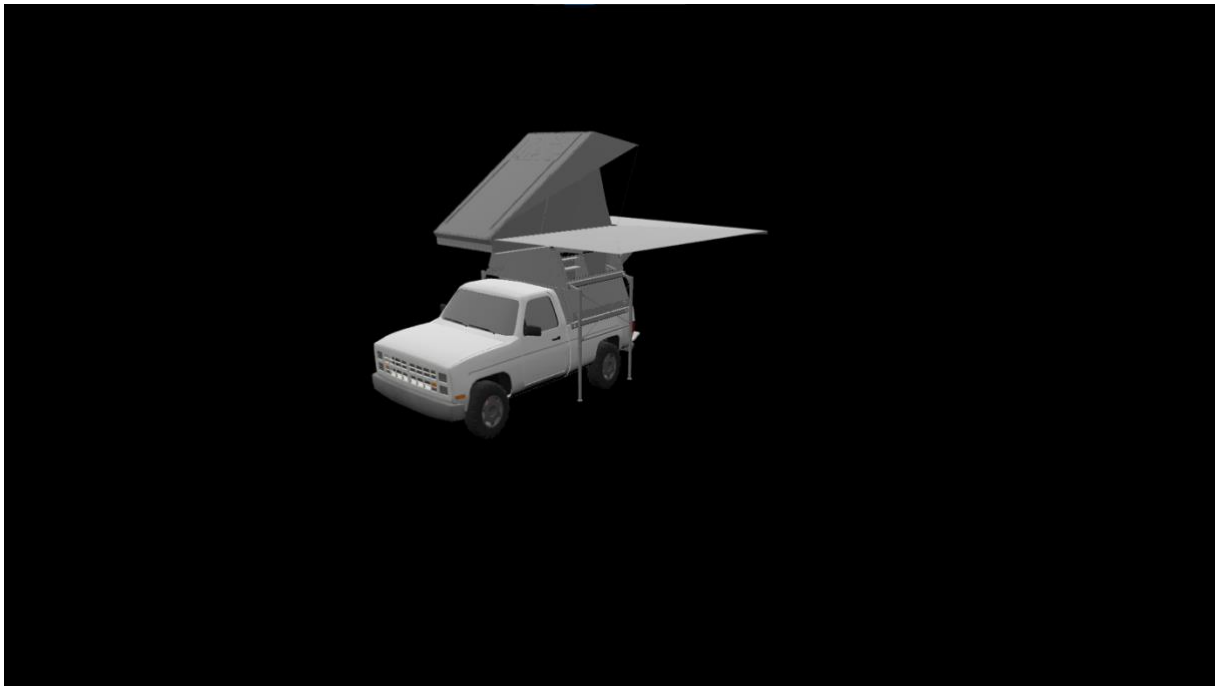

Figure 34 Current State of the Immersive Showroom

The main goal of this work was to find out whether it is worthwhile to create a digital twin for a digital showroom, which can then be used for commercial purposes. After completion of this project, we have shown a proof of concept that it is worth creating such a digital twin and displaying it in an immersive and virtual showroom. It is only necessary to find a real-time database solution compatible with the technologies used. If these technologies are covered in the chosen database, then nothing stands in the way of such a digital twin as we have worked on in this project.

5.4. Recommendations

Once the basic concept of the showroom is in place, there are endless possibilities for designing and expanding it. The environment in which the product is presented can be changed or completely redesigned. As an extension, some animations could be added to the product to demonstrate the load and unload of the changing systems on the pickup truck. It would also be interesting to add a feature to the showroom that allows to switch back and forth between multiple products and compare them. The possibilities for such a digital showroom in Unity are almost infinite. To make the experience even more realistic for the pick-up-and-go customer, one should figure out a way to add a 360-degree view to the showroom that corresponds to the pick-up-and-go workshop.

As mentioned before, the introduction of virtual reality glasses was also a topic in this project. Unfortunately, due to time constraints, we could not include this in this project, but we would like to briefly discuss it here and reveal our ideas about it and how this element can be included in future developments of this project.

We imagined that a person in the immersive showroom room could put on the glasses and thus experience the experience that the other person sees all around on three screens more accurately. In addition, the virtual reality glasses and the whole experience allow the user to open and close the flaps and drawers with the controller and thus also perceive the physical forces associated with it.

Regarding the web solution, a way must be found to provide a session to the customers. Otherwise, it will be challenging to provide the showrooms for several customers simultaneously because otherwise, the different customers change the data simultaneously, and one cannot use the immersive and virtual reality experience. Once a way has been found to give customers a session, it would also be possible to add a chat or a way to communicate via the showroom.

We would like to briefly share one of our ideas on what such a solution with a session could look like. If an interface is added at the beginning of the showroom, in which a number or something similar can be entered, one can assign the user and the associated supervisor to a room with this number. This room looks the same for all customers but differs in that the number that was previously entered when pushing to the database after each variable name is added. As an example, as shown in Figure 20 Firebase real-time database [34] from drawerPositionX to drawerPositionX123, one can push new values to the database for each new customer and delete them after the session. Because the consultant and the customer's number should be the same for the same session, both showrooms push and pull from the same values, such as drawerPositionX123. This way, several sessions can run simultaneously, and the different customers and consultants do not get in each other's way.

As for the immersive showroom, a solution should be found to change the digital and immersive showroom's unidirectional connection into a bidirectional one. This means finding out how to register the “X”, ”Y”, and “Z” coordinates and the animations in three.js and sending them to Firebase. Also, a way has to be found to export the whole Unity showroom and import it back into three.js so that not only the product as well as the pickup is displayed.

Suppose that the above recommendations are implemented correctly and combined with the findings of this project. In that case a digital twin that perfectly replaces an online showroom and it comes as close as possible to an actual showroom.

6. Summary

To get a better idea of how the showroom is used, two storyboards have been created that describe two classic use cases. Furthermore, the requirements of the two companies and the showroom have been written down. In the following, we explain why we chose the game engine Unity and what alternatives there would have been. It also covers how we imported the CAD files created by pick up and go into our showroom in Unity. As it was a time-consuming process, it is described with the help of a kind of pipeline. Finally, at the end of the chapter Concept / Methodology, there is a sketch of our software concept. This was created at an early stage to get a rough overview of our project

In order for two showrooms to display the same data at the same time, Unity needed a kind of database where the data is updated in real-time. In order to find a suitable tool for this, an evaluation with three different options was created. In the next step, we were going into more detail about our chosen database and explaining how we created the connection to the Unity showroom and how and which data is exchanged. Section 4.5 Showroom lists the essential packages that were used for the showroom. In addition, the render pipeline, the code editor, and details on the environment creation are explained and what else was done to make it as realistic as possible. Finally, based on the code snippet it becomes apparent how animations are inserted to understand the product's functions better.

We have described how we tested our converted concept and what changes were made afterward. On the one hand, we tested from desktop to desktop and, on the other hand, from desktop to the immersive showroom. Furthermore, a comparison was made between an actual showroom and the digital showroom, and the limits of the possibilities of both showrooms were described. Also described again were the achievements made at the end of the project. These included the real-time synchronization with the database connection, the import of the CAD files, and the display as an immersive showroom at Inside Reality.

7. References

7.1. Bibliography

7.2.

[1] N. Midderhoff, “Virtueller Showroom – Beispiele,” enra Blog, 2022.

[2] IKEA, “Virtual Reality Showroom,” [Online]. Available: https://present.digital/ikea/. [Accessed 07 2022].

[3] L. Blue, “Discover three.js,” [Online]. Available: https://discoverthreejs.com/. [Accessed August 2022].

[4] Wikipedia, “Wikipedia,” Wikipedia, 3 June 2022. [Online]. Available: https://en.wikipedia.org/wiki/Three.js. [Accessed August 2022].

[5] ST engineering Antycip, “VR Cave,” ST engineering Antycip, [Online]. Available: https://steantycip.com/vr-cave/. [Accessed August 2022].

[6] I. Wigmore, “CAVE (Cave Automatic Virtual Environment),” TechTarget, August 2016. [Online]. Available: https://www.techtarget.com/whatis/definition/CAVE-Cave-Automatic-Virtual-Environment. [Accessed August 2022].

[7] S. K. Arora, “Unity vs Unreal Engine: Which Game Engine Should You Choose?,” hackr.io, 18 November 2021. [Online]. Available: https://hackr.io/blog/unity-vs-unreal-engine#:~:text=Unity%20has%20a%20wide%20range,making%20the%20development%20process%20easier.. [Accessed July 2022].

[8] Unreal Engine, “Marketplace,” [Online]. Available: https://www.unrealengine.com/marketplace/en-US/store. [Accessed July 2022].

[9] A. Eldad, “Unity vs Unreal, What Kind of Game Dev Are You?,” INCREDIBUILD, 7 April 2021. [Online]. Available: https://www.incredibuild.com/blog/unity-vs-unreal-what-kind-of-game-dev-are-you. [Accessed July 2022].

[10] Unreal Engine, “The world’s most open and advanced real-time 3D creation tool,” Unreal Engine, [Online]. Available: https://www.unrealengine.com/en-US. [Accessed July 2022].

[11] Thinkwik, “CryEngine vs Unreal vs Unity: Select the Best Game Engine,” Medium, 20 April 2018. [Online]. Available: https://medium.com/@thinkwik/cryengine-vs-unreal-vs-unity-select-the-best-game-engine-eaca64c60e3e. [Accessed July 2022].

[12] CryEngine, “ASSET DATABASE,” CryEngine, [Online]. Available: https://www.cryengine.com/marketplace. [Accessed July 2022].

[13] CryTek GmbH, “CryEngine Docs,” 2015. [Online]. Available: https://docs.cryengine.com/. [Accessed July 2022].

[14] Unity, “Unity,” Unity Technologies, [Online]. Available: https://unity.com/. [Accessed 06 2022].

[15] Unity Technologies, “ProBuilder,” Unity, [Online]. Available: https://unity.com/de/features/probuilder. [Accessed 07 2022].

[16] Unity, “Render Pipelines,” Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/render-pipelines.html. [Accessed 07 2022].

[17] Unity, “Skybox,” Unity Technologies, 2 July 2017. [Online]. Available: https://docs.unity3d.com/560/Documentation/Manual/class-Skybox.html. [Accessed August 2022].

[18] Unity, “Volumes,” Unity Technologies, 2020. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Volumes.html. [Accessed August 2022].

[19] Unity, “Packages and feature sets,” Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/PackagesList.html. [Accessed 06 2022].

[20] Unity, “Creating and Using Scripts,” Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html. [Accessed 06 2022].

[21] Unity, “Asset Store,” Unity Technologies, [Online]. Available: https://assetstore.unity.com/. [Accessed 05 2022].

[22] Visao, “What is HDRI? Learn more about HDR images,” Visao, 13 January 2021. [Online]. Available: https://visao.ca/what-is-hdri/. [Accessed August 2022].

[23] File Format, “File Format GLTF File,” File Format, [Online]. Available: https://docs.fileformat.com/3d/gltf/. [Accessed 07 2022].

[24] File Info, “File Info,” [Online]. Available: https://fileinfo.com/. [Accessed 05 2022].

[25] educative, “What is Firebase?,” Educative Inc., 2022. [Online]. Available: https://www.educative.io/answers/what-is-firebase. [Accessed August 2022].

[26] podiluska, “Difference between JSON and SQL,” stackoverflow, 27 February 2014. [Online]. Available: https://stackoverflow.com/questions/22071735/difference-between-json-and-sql#:~:text=They%20are%202%20completely%20different,to%20talk%20to%20web%20services.. [Accessed August 2022].

[27] K. Bohn, “Atomic Object,” Atomic Object LLC, 20 March 2021. [Online]. Available: https://spin.atomicobject.com/2020/03/17/firebase-unity-setup/. [Accessed 20 06 2022].

[28] StoryboardThat, “StoryboardThat”.

[29] pick-up-and-go GmbH, “pick-up-and-go,” [Online]. Available: https://www.pick-up-and-go.ch/. [Accessed 05 2022].

[30] Inside Reality, “Inside Reality Products,” [Online]. Available: https://inside-reality.com/products/. [Accessed 05 2022].

[31] Third Aurora, “Youtube,” Third Aurora, 13 July 2021. [Online]. Available: https://www.youtube.com/watch?v=zgSES5PQNu8. [Accessed 07 2022].

[32] GabaLaba, “Standard vs. HDRP - Terrain comparison. Thoughts?,” 2021. [Online]. Available: https://www.reddit.com/r/Unity3D/comments/naedr5/standard_vs_hdrp_terrain_comparison_thoughts/. [Accessed August 2022].

[3] Unity, “High-Definition Render Pipeline (HDRP),” Unity Technologies, [Online]. Available: https://unity.com/de/srp/High-Definition-Render-Pipeline. [Accessed 5 08 2022].

[4] Firebase, “Realtime Database,” Microsoft, July 2022. [Online]. Available: https://console.firebase.google.com/u/0/project/testing-realtime-unity/database/testing-realtime-unity-default-rtdb/data. [Accessed July 2022].

[5] NBN MIinds, “NBN MINDS,” NBN MINDS, 1 April 2019. [Online]. Available: https://www.nbnminds.com/when-you-should-and-shouldnt-use-firebase/#:~:text=Automatic%20Scaling&text=Additionally%2C%20the%20API%20functions%20of,any%20change%20in%20the%20code.. [Accessed 07 2022].

[6] S. Ivanov, “back4app,” Back4App, 2022. [Online]. Available: https://blog.back4app.com/de/was-ist-firebase/. [Accessed 20 07 2022].

[7] DelfStack, “HTML-Tabelle mit JavaScript nach Excel exportieren,” DelftStack, 18 October 2021. [Online]. Available: https://www.delftstack.com/de/howto/javascript/exprt-html-table-to-excel-javascript/#:~:text=und%20sheet%20bereitgestellt.-,Verwenden%20Sie%20die%20Bibliothek%20TableExport%20%2C%20um%20HTML%2DTabellen%20in%20eine,%2C%20CSV%2D%20oder%20Textdateien%20expor. [Accessed 06 2022].

[8] C. Seibert, “I'M CODER,” 16 Dezember 2008. [Online]. Available: https://im-coder.com/lokalen-zugriff-auf-dateien-mit-javascript-2.html. [Accessed 07 2022].

[9] L. Talbert, “Saving Game Data with Unity,” Red Gate Software Ltd, 03 February 2020. [Online]. Available: https://www.red-gate.com/simple-talk/development/dotnet-development/saving-game-data-with-unity/#:~:text=Unity%20provides%20two%20ways%20to,%2C%20and%20you're%20done.. [Accessed 07 2022].

[10] T. Keller, “Read Data,” 07 2022. [Online]. Available: https://ip6-firebase.vercel.app/. [Accessed 07 2022].

[11] Google Developers, “Firebase,” Google Developers, 19 Juli 2022. [Online]. Available: https://firebase.google.com/docs/web/setup. [Accessed 18 Juni 2022].

[12] DocFX, “Unity UI: Unity User Interface,” Unity Technologies, 6 October 2020. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html#:~:text=Unity%20UI%20is%20a%20UI,position%2C%20and%20style%20user%20interfaces.. [Accessed 07 2022].

[13] Microsoft, “Unity Development with VS Code,” Microsoft, 8 March 2022. [Online]. Available: https://code.visualstudio.com/docs/other/unity. [Accessed 07 2022].

[14] KhronosGroup, “KhronosGroup/UNityGLTF,” Github, 3 April 2019. [Online]. Available: https://github.com/KhronosGroup/UnityGLTF. [Accessed 07 2022].

[15] prefrontalcortex, “prefrontalcortex/UnityGLTF,” Github, [Online]. Available: https://github.com/prefrontalcortex/UnityGLTF?path=/UnityGLTF/Assets/UnityGLTF. [Accessed 07 2022].

[16] Unity, “Asset Store Mini First Person Controller,” Unity Technologies, [Online]. Available: https://assetstore.unity.com/packages/tools/input-management/mini-first-person-controller-174710#description. [Accessed 07 2022].

[17] Poly Heaven, “Poly Heaven Textures,” Poly Heaven, [Online]. Available: https://polyhaven.com/textures. [Accessed 05 2022].

[18] S. Wenz, “A virtual showroom for the smart city vehicle of the future,” Unreal Engine, 17 March 2021. [Online]. Available: https://www.unrealengine.com/en-US/spotlights/a-virtual-showroom-for-the-smart-city-vehicle-of-the-future. [Accessed July 2022].

7.3. Table of Figures

7.4. Table of Tables

«We the undersigned declare that all material presented in this bachelor thesis is our own work and written independently only using the indicated sources. The passages taken verbatim or in content from the listed sources are marked as a quotation or paraphrased. We declare that all statements and information contained herein are true, correct, and accurate to the best of our knowledge and belief. This paper or part of it have not been published to date. It has thus not been made available to other interested parties or examination boards. »

Windisch, 19.08.2022

Name: Timon Keller

Signature: T. Keller

Name: Timon Tschanz

Signature: T. Tschanz
Timon Tschanz (Aug 19, 2022 12:16 GMT+2)

Appendix

GitHub for Showroom: https://github.com/timon7796/IP6-Immersive-Showroom
GitHub for React App: https://github.com/TimonKeller/ip6_firebase

Project track IP6 Real-time synchronization with an immersive, digital showroom

Final Audit Report 2022-08-19

Created:	2022-08-19
By:	Timon Keller (timon.keller@students.fhnw.ch)
Status:	Signed
Transaction ID:	CBJCHBCAABAAKW89Lo3kVOKIDEipboFjGr8bCEhsSbKw

"Project track IP6 Real-time synchronization with an immersive, digital showroom" History

Document created by Timon Keller (timon.keller@students.fhnw.ch)
2022-08-19 - 9:03:43 AM GMT- IP address: 85.0.13.122

Document emailed to timon.tschanz@students.fhnw.ch for signature
2022-08-19 - 9:04:23 AM GMT

Email viewed by timon.tschanz@students.fhnw.ch
2022-08-19 - 10:13:00 AM GMT- IP address: 172.225.190.146

Signer timon.tschanz@students.fhnw.ch entered name at signing as Timon Tschanz
2022-08-19 - 10:16:04 AM GMT- IP address: 178.197.204.200

Document e-signed by Timon Tschanz (timon.tschanz@students.fhnw.ch)
Signature Date: 2022-08-19 - 10:16:06 AM GMT - Time Source: server- IP address: 178.197.204.200

Agreement completed.
2022-08-19 - 10:16:06 AM GMT