# Documentation for the SeestadtVision AR-App

The project focuses on offering the people living in the Urban Lakecity (Seestadt) in Vienna the possibillity to design the city themselves in a way and to express how they would like to redesign certain aspects.

SeestadtVision is an Augmented-Reality (AR) application which doesn't just offer people the chance to express their imagination that way, but also features modes where all kinds of useful information can be found and where a Now/Future-view of buildings which currently get built is accessible. All of the features of the application are easily adaptable, adjustable, and expandable.

The application has been programmed in "Unity" by using multiple plugins and tools which provide the possibility to enhance application and configure them for AR/VR.

While I'll describe the details of the project as good as possible in this document, I will try to keep it less technical and more user-related so that everything can be understood easily. I'll also explain a little bit about the plugins and tools I tested on my path to creating the finished application.

Following one is the link to the GitHub-Repository: https://github.com/IneffableChris/SeestadtVision

## Table of Contents

## What can you do with the application?

The application provides people living in the Seestadt the possibility to shape the open space and to redesign the city how they would like. This helps making the city more user-friendly and can embrace sharing ideas on how to use the available space in a better way. While the project focuses on the people living in the Urban Lakecity (Seestadt), in Vienna, users all around the world have the possibility to use the application for their needs and to place objects, information and other features of the app wherever they live themselves.

SeestadtVision is able to show information about the buildings, spaces, places, and for example even streets of the Urban Lakeside City (Seestadt) such as for example about the woman after which the streets were named. Furthermore, a mode is included which, in future versions, should be able to provide a Present-Future-View about how the city currently looks like and how it should or will look like in the next few years. The application therefore combines the opportunity to reshape the city according to the user's needs while also giving insight into how the current plans would enhance the cityscape.

To do all these things, SeestadtVision uses AR-technology which gets provided by the development-environment called "Unity".

One of the most important factors of developing this application was that it should be easily adaptable to add more objects which can be placed, to adjust certain settings about the models, heights, and information and to build upon the possibilities which currently exist by using the different modes. Therefore, the application is adaptable, adjustable, and expandable.

## Plugins, Tools, Assets & Installation

To deploy and work on the software, following tools are needed:

- ARKit (to deploy for iOs)
- ARCore (to deploy for Android)
- more specific tools, depending on the version of your phone - those might be installed automatically after switching to "Build for xyz" in the Build Settings.
- the AR+GPS Plugin by Daniel Fortes, which you can read more about here: [https://docs.unity-ar-gps-location.com/#main-features]

The code used for the project runs on Unity, last tested on version 2020.3.23f1. For more information about the firmware see [https://unity.com/de/download]

If you only want to use the application itself, it is enough to install the project .apk, which you can find here: https://github.com/IneffableChris/SeestadtVision to your phone. You are ready to start the application afterwards immediately.

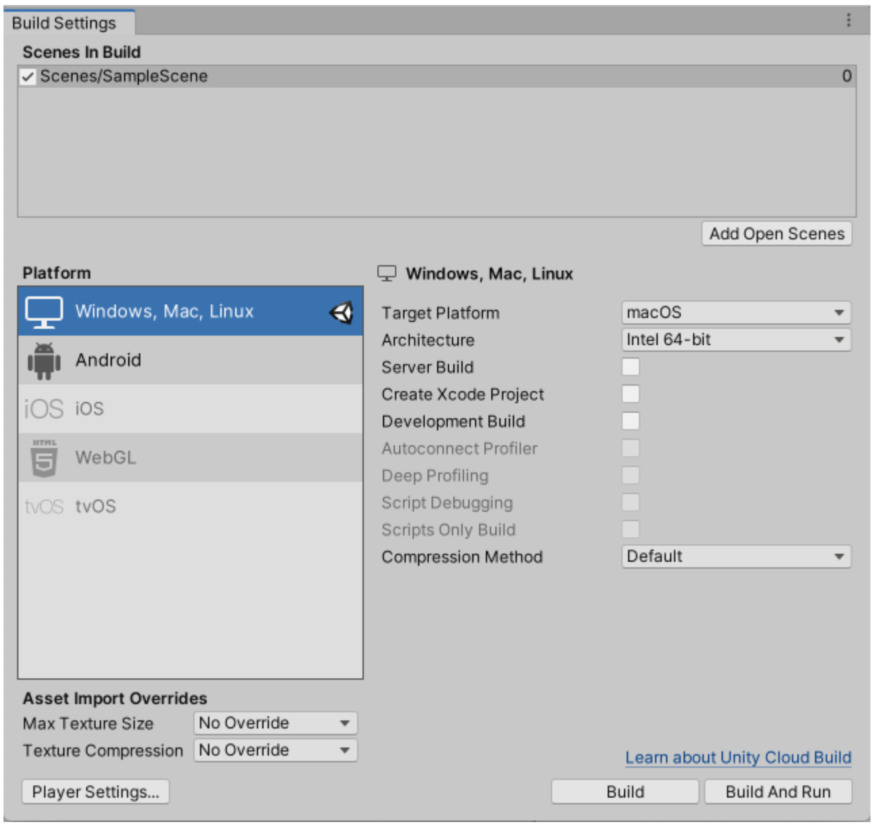
Besides that, for developers, it is necessary to install the development environment "Unity".

## Build settings

"Building" means to export the project from Unity to, for example, a phone. Users can build their project for PC-usage, for the phone, for the Web, and so on, simply by choosing "Build Settings" in the main menu. Here, one button-click on the according modes is enough to change this choice.

You can read more about the different settings and options when building the application to your phone here:

- for Android: [https://docs.unity3d.com/2020.1/Documentation/Manual/android-BuildProcess.html]
- for iOs: [https://docs.unity3d.com/2020.1/Documentation/Manual/BuildSettingsiOS.html]
- general information: [https://docs.unity3d.com/2020.1/Documentation/Manual/PlatformSpecific.html]


Build Settings

Scenes In Build

✓ Scenes/SampleScene 0

Add Open Scenes

Platform

Windows, Mac, Linux

Android

iOS

WebGL

tvOS

Windows, Mac, Linux

| | |
|---|---|
| Target Platform | macOS |
| Architecture | Intel 64-bit |
| Server Build | ☐ |
| Create Xcode Project | ☐ |
| Development Build | ☐ |
| Autoconnect Profiler | ☐ |
| Deep Profiling | ☐ |
| Script Debugging | ☐ |
| Scripts Only Build | ☐ |
| Compression Method | Default |

Asset Import Overrides

| | |
|---|---|
| Max Texture Size | No Override |
| Texture Compression | No Override |

Learn about Unity Cloud Build

## ARKit (iOS) und ARCore (Android)

First of all, to enable AR for iOS and Android the respective Plugins must be implemented into the project. There are numerous tutorials on the internet on how to do so, although the easiest way is to either install the tools when downloading the Unity version here: https://unity.com/de/download

When downloading, users can choose which version they would like to use, and which add-ons get installed with that version. It is recommended to install the ARKit and ARCore Plugins during this step. Besides that, there is the possibility to open the Package Manager via the menu in Unity, which is on top of the screen, where ARKit and ARCore can be installed within just a few mouse-clicks.

The implementation into Unity is pretty simple, as can be read about here: (https://developers.google.com/ar/develop/unity/quickstart-android), and it is a great way to get results fast while also "configuring" the device for AR-usage for the very first time.

By using the ARCore Depth API it is possible to place objects in the real world. ARCore and ARKit understand depth – at least if the device has the necessary technical features. You can read more about the technical requirements and tutorials about installing the necessary software here: https://developers.google.com/ar/develop/unity-arf/getting-started-ar-foundation

There is an app for iPhone called ARKit which shows some demos that can be tested out on the device instantly.

There is also ARFoundation which can be used in combination with ARKit and ARCore and Unity. With Foundation visual effects can be created by using the Unity shaders and a few other Unity options.

## AR+GPS Plugin by Daniel Fortes

A specific plugin that stood out to me is the AR+GPS Plugin for Unity by Daniel Fortes. At first, I was very surprised that there were seemingly not many "Tutorials" to the whole concept of placing AR objects in the real world while being able to interact with them and so on. By using Daniel's Plugin, those actions became possible. The AR+GPS Plugin by Daniel Fortes is the main tool used to create the SeestadtVision application.

Furthermore, he has created a little guide to his plugin: https://docs.unity-ar-gps-location.com/#main-features

The plugin works by using AR Foundation, ARKit/ARCore and Vuforia. GPS data and AR tracking gets mixed.

By using his plugin, I was able to place all kind of objects, from zombies to self-modelled Blender objects, on the street right in front of me, but not just that, I also was able to create a path on which the objects kept moving! In his plugin are numerous features, following ones are mentioned in the guide:

- Place 3D Objects in geographical positions defined by their latitude, longitude, and altitude.
- AR Hotspots that are activated when the user is near a given location.
- Place 3D Text markers on real-world points of interest (example using OpenStreetmaps is included.)
- Smooth movements on device location and heading updates.
- Move objects or place them along paths (Catmull-rom splines) on the map.
- Augmented reality floor shadows.
- General purpose Catmull-rom curves and splines.

I have tested out mostly all of them – placing objects in their geographical positions worked out great. I could simply open Google Maps, place a marker on the map and copy the latitude and longitude numbers into the Unity project. The altitude can be set to a specific amount, otherwise the objects placed at a spot are automatically at the same height as the device.

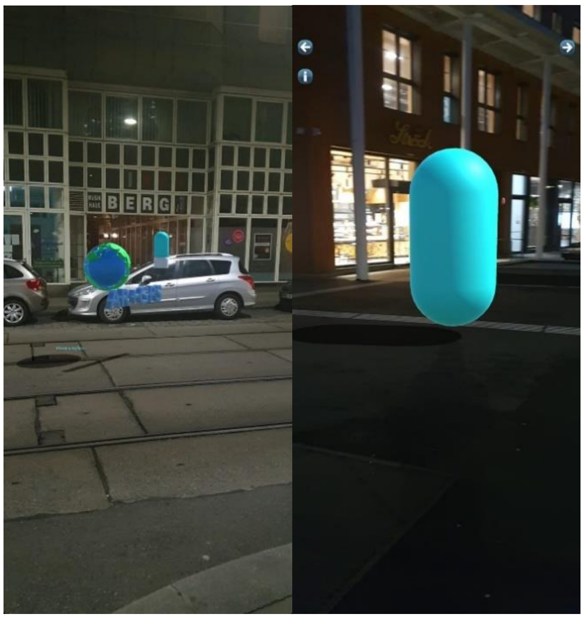
AR Hotspots are a great feature too. By placing numerous objects in different distances, I was able to test the feature and slowly see all objects appearing, according to my positioning – whenever I came closer more objects appeared in front of me.

The 3D Text markers worked out too, although it has to be said that the altitude has to be considered more than it had to be with Mapbox.

## Different Modes

### NoteMode or "AR-Information-Mode"

In the case of this project nearly everything worked by using the AR+GPS Plugin. In the NoteMode models and information can be placed on the real-life position of a map. The pictures below this text show some spheres, a little bit of text and the "AR+GPS"-Logo which I on the one hand placed in front of a building and on the other hand in front of another building, namely the "Ströck", which is already located in the Seestadt.



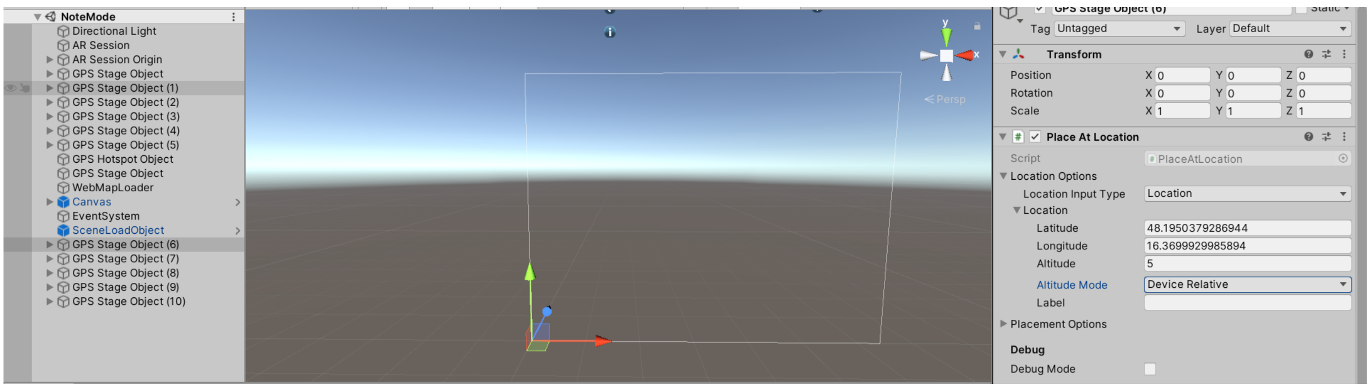
*Graphic 1 & 2: Unity NoteMode*

In the Unity project the NoteMode can be accessed with following path: Assets -> Modes -> ARLocation.

Here, numerous files can be found – it is recommended to take a look at the Demos in the Samples folder to get a better overview over the possibilities of the tool in that regard.

The easiest way to add new objects to the real-world, following steps have to be made:

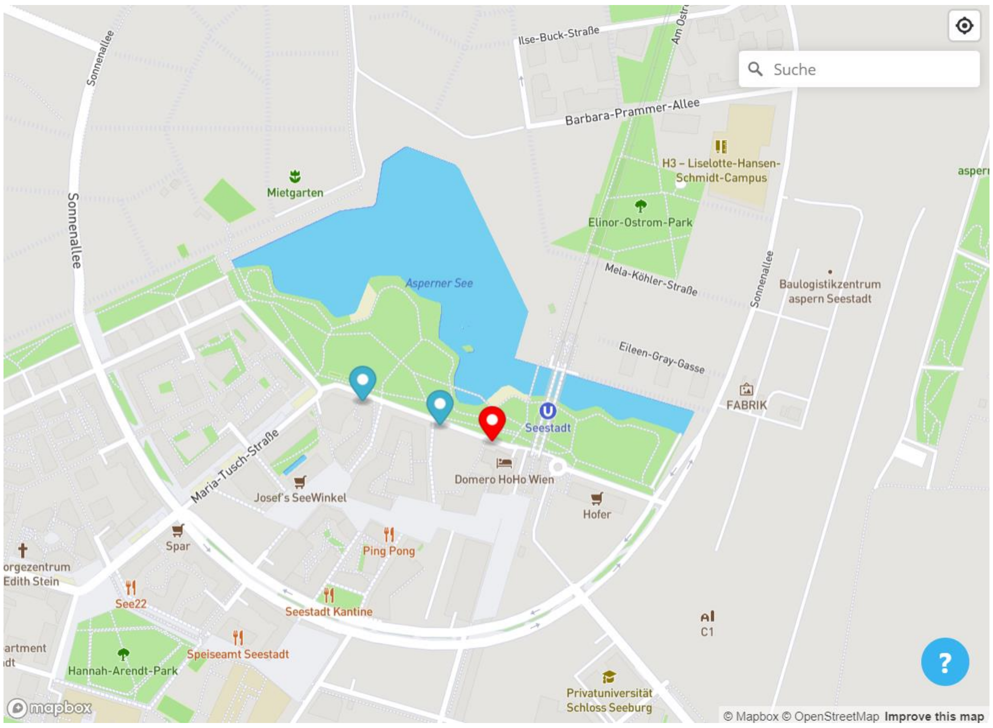
1. Open either the web-plugin to set markers and pins on the Web-Map and duplicate the locations of those to fill them into the models you would like to show in Unity, or open Google Maps and get the location of a certain position by pressing the right mouse button on a point of the map. Web-Editor: https://docs.unity-ar-gps-location.com/map/
2. Copy the longitude and the latitude of the position.
3. Open Unity, create a Scene, delete everything from the Hierarchy and create a new Basic Scene Structure by pressing right click -> AR+GPS -> Create Basic Scene Structure.
4. Add a GPS Stage Object.
5. Open the properties of the Object and find the "PlaceAtLocation" Script options at that object
6. Open the Placement Options, toggle up the Location settings and fill in the Longitude and the Altitude which you copied or saved in Step 2. You could also now start with step 2 again if you'd like to add new objects (and of course skip 3-5 since you already created the scene now).
7. Decide regarding the Altitude Mode – Device Relative for example means, that the object you placed will be on the same height as the device which opens the app is.
8. Build the application to your phone and start the application; you should now be able to see the object!



```
NoteMode
  Directional Light
  AR Session
  AR Session Origin
  GPS Stage Object
  GPS Stage Object (1)
  GPS Stage Object (2)
  GPS Stage Object (3)
  GPS Stage Object (4)
  GPS Stage Object (5)
  GPS Hotspot Object
  GPS Stage Object
  WebMapLoader
  Canvas
  EventSystem
  SceneLoadObject
  GPS Stage Object (6)
  GPS Stage Object (7)
  GPS Stage Object (8)
  GPS Stage Object (9)
  GPS Stage Object (10)
```

```
GPS Stage Object (6)                          Static
Tag  Untagged              Layer  Default
Transform
  Position    X 0   Y 0   Z 0
  Rotation    X 0   Y 0   Z 0
  Scale       X 1   Y 1   Z 1
Place At Location
  Script                 PlaceAtLocation
  Location Options
    Location Input Type  Location
    Location
      Latitude           48.1950379286944
      Longitude          16.3699929985894
      Altitude           5
      Altitude Mode      Device Relative
      Label
  Placement Options
  Debug
  Debug Mode             ☐
```

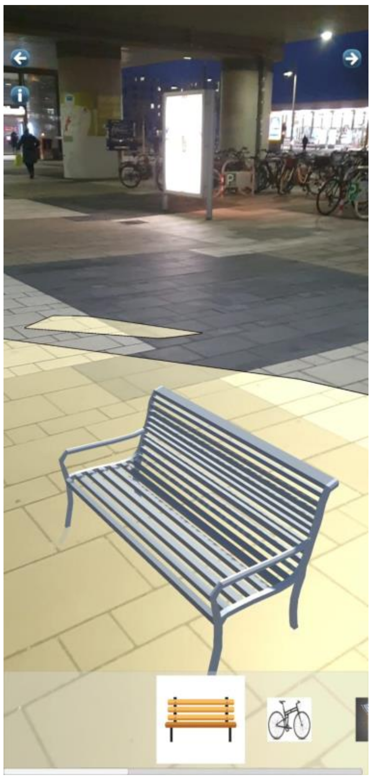
*Graphic 1: Unity UI NoteMode*



*Graphic 2: Web Editor to set markers*

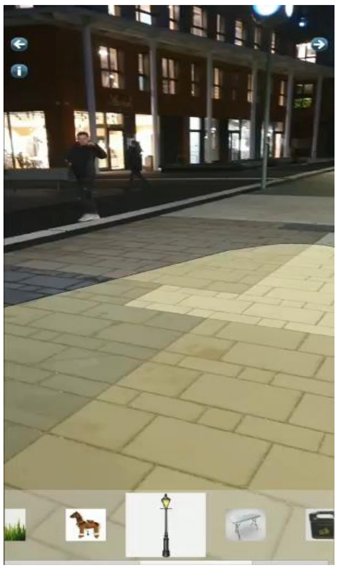
## ARPlacement-Mode or "AR-Objektplatzierungsmodus"

This is the chapter which describes the mode where people can insert the objects according to their needs right in the application, instead of seeing information which has already been placed. Since this application hasn't been user-tested yet, I decided to stick with usual objects and items for now, which can be placed in the real world. Examples would be a bench, a tree, grass, and so on.


Graphic 3: Placement-Mode Bench

The mode works by using AR Foundation and doesn’t use the AR+GPS Plugin. I found a great guide which allowed me to create the UI which you can see on the screenshot, which fitted my needs perfectly. The mode is perfectly adaptable and adjustable since whenever new objects are added to the mode, the UI adapts automatically and provides a new item to hover over. Furthermore, the bar which is used to slide through the objects can then move even further and adjusts itself to the new item.

Users can add the items by pressing on the according item on the bottom and then pressing onto wherever they want to place the object. They can move through the menu and add more objects.


Graphic 6: Users can slide through the elements

How to add new items:

1. Direct to Assets -> Modes -> ARPlacement -> Prefab
2. Open Google and download a free usable Model of the object you would like to add to the scene and also download an image which looks alike. I liked to use https://sketchfab.com/feed a lot since they provide many different models.
3. Drag the downloaded model into the Prefab folder.
4. Press the model, quickly check if it gets displayed correctly and decide upon the “Scale Factor” – this defines how big the model will look like in the scene. As reference, I believe that 0.5 is a good scale for the bench that I used.
5. Add the image of the object which you downloaded as well to the Prefab Folder and set the Texture Type to “Sprite (2D and UI)”.
6. Now direct to Assets -> Modes -> ARPlacement -> Resources -> Items.
7. Create a new Item by either copying an existing one or by pressing the right mouse button -> Create -> Create Item
8. Choose your Item Prefab, which is the model you inserted and also add the Image under Item Image.
9. When you now start the application, you should already be able to see the image of your object in the bar. By pressing onto the object and then pressing somewhere into the scene, the model should already be getting placed in the real world.

### AR-Location-Mode

The AR-Location-Mode should provide more information about how the city currently looks like and how it will look like in the future. Sadly, due to not having enough time and information regarding how the city will look like, I haven’t been able to program this mode efficiently enough yet.

My idea would be relatively simply though, namely, that I’d like to toggle the mode on and off to see the current status and the future status. You can direct to Assets -> Modes -> ARFuture to see what I currently programmed, although this mode is definitely not finished yet.

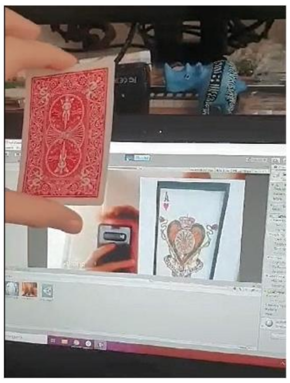
## Further Plugins which have not been used for the finished project:

There are numerous plugins which are very interesting and offer features which are not included in the current project, which is why I offer a small overview about them and their possibilities.

### Vuforia

Vuforia in combination with Unity stood out to me when it comes to image recognition. It is very easy to import into Unity and offers a large number of features, including occlusion handling (at least when it comes to very small margins such as a hand between the object and the camera and so on) and it is great to use for all kind of scanning. I personally tried out an experiment where I loaded an image into the Vuforia database which is usable by using the web service and inserting the “key” in the unity project, by which the database and the project more or less get connected.

By showing a specific playing card to the camera the image of the card got switched out by another one in AR. In addition to that I was able to start a video after the image got scanned.



On the picture the Ace of Hearts is visible – although I hold a different card. The swap happened after the camera saw my card which is the same, I saved in the database. The video to the explained effect is added to the folder in which this document is.
The advantages of Vuforia are that it is relatively simple to create good looking visualisations and tracking's. There are numerous functionalities, even with the free version, and it is nicely importable to Unity.

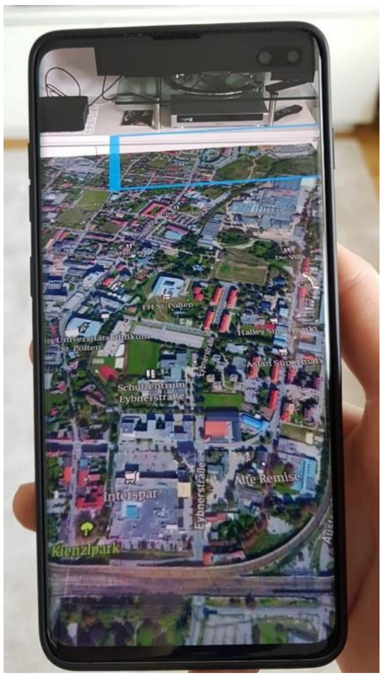
On the downside projects with Vuforia can become very complicated and massive quite fast if more than "just" a more or less simple tracking should be done. The free version is limited in its functionalities and does not allow much more than said effects.

While this Plugin isn't necessary to understand for the current version of the project, it is a good idea to look into it since it was and is planned that the current application builds upon more features which are made possible by using Vuforia.

## Mapbox

Mapbox creates opportunities to create maps of all kinds, edit them, visualize data, create routes and to use a traffic system for a car and on top of it Mapbox features AR too.

With the combination of Mapbox and AR I was able to create custom maps and display them via AR right in front of me, which I have even used for my bachelor thesis.



Thanks to the GPS Location Service I was able to place tags on the street I live on – I placed a label onto the house next to me which, when I held the camera of my phone into the right direction, was clearly visible.

On the downside with Mapbox it is awesome to create custom maps, display them and much more, but the more complex the data I wanted to display got, more and more problems appeared. I sadly have not tried out every aspect of Mapbox since I soon afterwards found a Plugin which seemingly suited my needs even better, but I will surely come back to Mapbox in the near future.

I already know though, that if we would plan to use Mapbox, the storing and the usage of the data has to be quite carefully. On top of that I would recommend using another plugin with potentially better GPS Features. While Mapbox is undoubtedly the best when it comes to creating maps, I trust other Plugins/Tools more when it comes to the GPS aspects, not to mention that many more features have to be considered when it comes to that.

## Kudan SDK and Wikitude AR

Kudan SDK for Unity offers the option to place objects in the real world, to animate them and to apply different rendering options.

While I did try out some features of Kudan, I won't go to deep in the discussion about its positive and negative aspects since I simply think, that there are better options than using this not as known SDK as others – meaning that there aren't a lot of sources to find solutions for appearing problems. While Kudan is quite good when it comes to markers and markerless tracking I believe that the tools I mentioned above are equally as effective.

The same pretty much counts for Wikitude, even though Wikitude has a little bit more "expertise" and focuses on more features rather than purely tracking. On top of that a nice documentation and a diversity of YouTube videos exist which help drastically in getting used to Wikitude.

## Photon Engine & Unity in combination with Firebase

I have also informed myself about possibilities to save data, create “Multiplayer-Scenes” and so on – therefore I tried a few different tools, although I surely tried out the most with firebase. It is possible, and not too difficult, to import firebase into a Unity project which was awesome to test.

The Photon Engine works great too, but in hindsight it may not be too relevant this project at all.

## AFrame

Here I managed to create projects in a very short amount of time, mostly markers were easy to create and even AR objects were perfectly displayable right in front of me.

While I also had high hopes for the GPS Location part I sadly didn’t get even close to as good results as I had with the “very near” AR applications.

Funnily I even found a Unity Plugin which combines AFrame with Unity – here the projection of an image or a video became possible, setting markers became even easier and I truly enjoyed the usage of Unity tools and AFrame. While it is quite a weird combination since GPS data such as longitude/altitude and so on can be inserted with html/java and AFrame without Unity totally fine it is more or less meaningless to combine it with a game engine. Therefore the other advantages of an engine have to be considered before combining both.

Maybe AFrame would be the best choice when using a Web-based tool.

On top of that there are numerous properties which can help with testing such as “simulateLatitude”, “simulateLongitude” and “simulateAltitude” with which small tests can be created quite well.

## AR.js

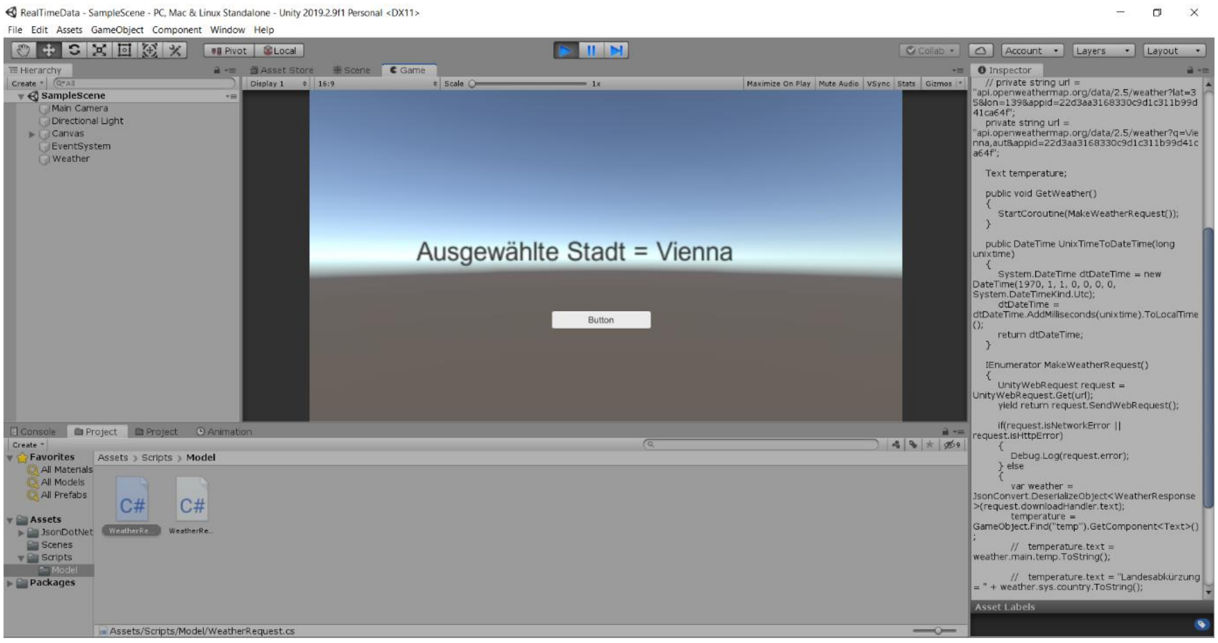
AR.js is an interesting case as well – image tracking, placing of objects, creating trackers in general and so on is possible too, although I do like AFrame more so far. With AR.js it is, again, really easy to create projects quite fast, even though the very early beginning might be a little bit rough.

It is possible to create AR projects with just a few lines of code, but everything becomes much harder when it comes to GPS locations, being exact about the positions, viewshed, and so on.

While I was very happy with the results after I have been through the troubles I had in the beginning, it was still pretty annoying that for example interactions did not get tracked accordingly, which is obviously a big no-go.

## Honourable Mentions

The IATK 1.0 (Mala): Immersive Analytics Toolkit (https://github.com/MaximeCordeil/IATK, https://www.researchgate.net/publication/331968628_IATK_An_Immersive_Analytics_Toolk it) and DXR (http://hvcl.unist.ac.kr/wp-content/papercite-data/pdf/jychoi_dxr_2019.pdf) provided interesting information, while the Totem-Plugin offered the possibility to place objects. While the Plugin didn't fit my needs, it was a good start to get in touch with AR applications.



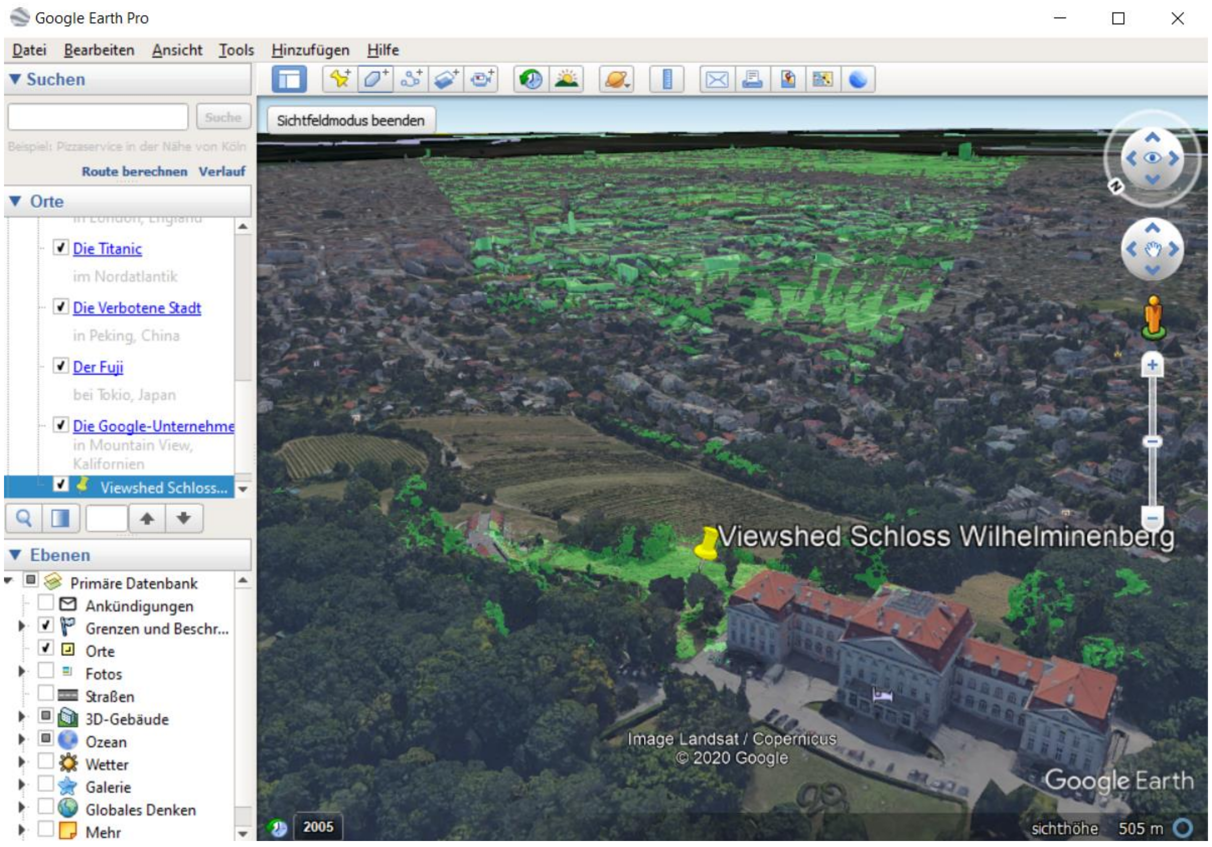
```
    // private string url =
"api.openweathermap.org/data/2.5/weather?lat=3
5&lon=139&appid=22d3aa3168330c9d1c311b99d
41ca64f";
    private string url =
"api.openweathermap.org/data/2.5/weather?q=Vie
nna,aut&appid=22d3aa3168330c9d1c311b99d41c
a64f";

    Text temperature;

    public void GetWeather()
    {
        StartCoroutine(MakeWeatherRequest());
    }

    public DateTime UnixTimeToDateTime(long
unixtime)
    {
        System.DateTime dtDateTime = new
DateTime(1970, 1, 1, 0, 0, 0, 0,
System.DateTimeKind.Utc);
        dtDateTime =
dtDateTime.AddMilliseconds(unixtime).ToLocalTime
();
        return dtDateTime;
    }

    IEnumerator MakeWeatherRequest()
    {
        UnityWebRequest request =
UnityWebRequest.Get(url);
        yield return request.SendWebRequest();

        if(request.isNetworkError ||
request.isHttpError)
        {
            Debug.Log(request.error);
        } else
        {
            var weather =
JsonConvert.DeserializeObject<WeatherResponse
>(request.downloadHandler.text);
            temperature =
GameObject.Find("temp").GetComponent<Text>()
;
            //   temperature.text =
weather.main.temp.ToString();

            //   temperature.text = "Landesabkürzung
= " + weather.sys.country.ToString();
```

Furthermore, I looked into performance issues with Unity and how to solve problems in that regard – there are numerous ways to do so, which made me even more sure that Unity would be a great fit for AR projects. With Unity many tools are import- and usable, data could get stored, AR objects displayed easily, and performance issues could be solved.

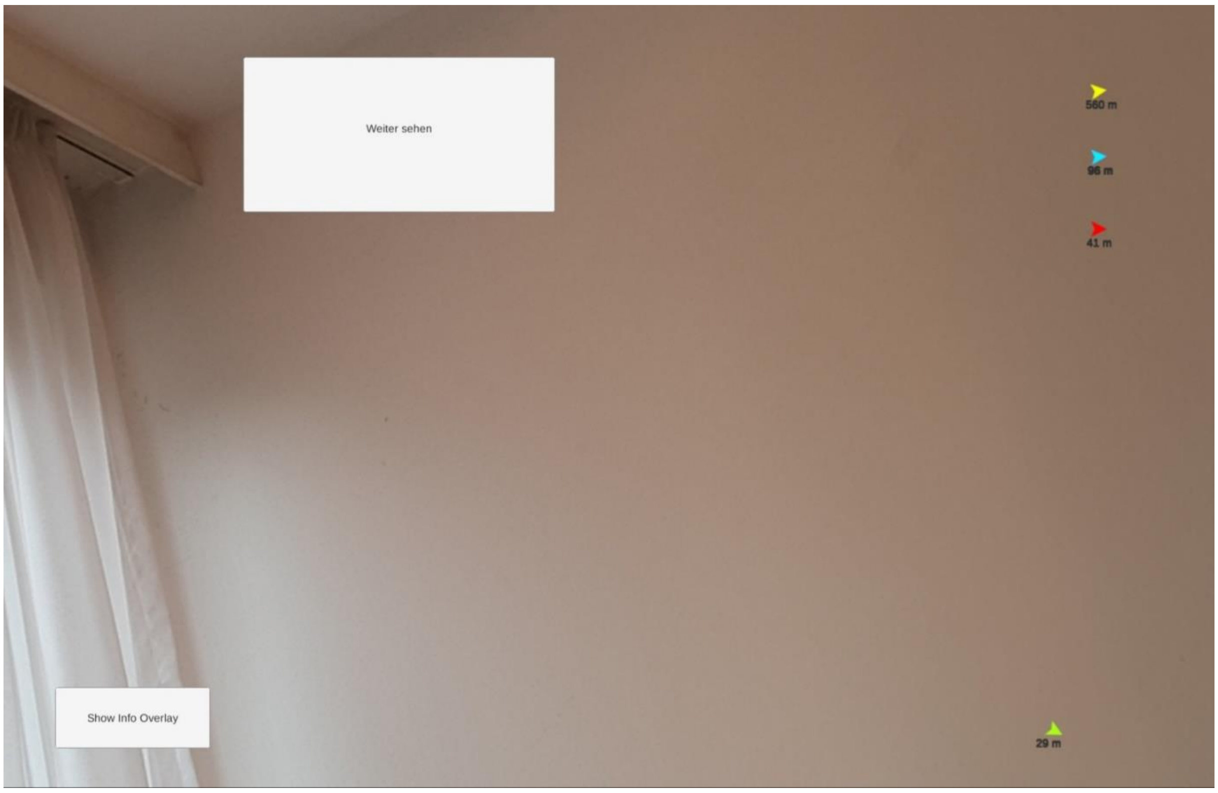
One of the problems of displaying objects is in regard to "Viewshed". A Viewshed is more or less the view that an object has, depending on where it is placed. For example, I should not be able to see an object if a wall is between us; I would have to go around the wall in order to have free sight at it, with which the object should become visible. Sadly, I have not been able to solve this massive problem yet. My current try is to make the shadows colourful and to write a script in which it gets declared that only the colourful "blocks" should be visible.

With the tool "Google Earth Pro" it is possible to create a viewshed yourself.

In the following picture the green parts are what is visible from the perspective of "Schloss Wilhelminenberg", or to be more exact in the field right beside it.



Furthermore, I wrote a program which combines the spawning of objects in AR with arrows that lead to the objects in the scene. Whenever they leave the scene, the arrows appear so that the user instantly sees where the objects he visually can't see on his screen are, which leads to him being able to simply move the device in the direction of the arrows and seeing the objects. Therefore, even such applications and features which are not included in the plugins and tools in general can be created.



In this screenshot you can see one of my first prototypes of that kind of application. On the right there are 4 arrows which also indicate in which distance the objects which are outside of the screen currently are. By pressing the button “Weiter sehen” I activate another camera with which the user can see further than with the usually activate camera. By using that function I was able to test a few options and location based features I tried to combine with this application.

## Support

For support or when questions arise, you are always free to email e51825978@student.tuwien.ac.at