AVR Project Report

MeasureIt

AR markerless measuring application

Created by:

Michal Jurewicz (266892)

Software Engineering

6th semester

12.04.2020

Table of content

1. Brief Description

Due to the scope of the project and all of its delimitations, the idea could not involve too many complex parts in order to make sure that it can be successful. Although the product is a quite simple application that performs one task with a few extensions to it, it can be undoubtfully considered to be a fully working prototype that is usable in real-life scenarios.

The purpose of the application is to measure distances marked on a plane, which is generated using AR plane detection algorithms from AR Foundation. The user is able to place multiple points that are used to calculate how far away they lay from each other. This can be done both on vertical and horizontal surfaces on any plane that can be recognized.

The final product was built for devices with Android system. There are many devices that are able to run ARCore and therefore the application can be used on any of them that fills these two requirements. However, it is needed to be remembered that time and resources were restricted and the only device it was tested on was a mobile phone in portrait mode, which the application is designed for primarily.

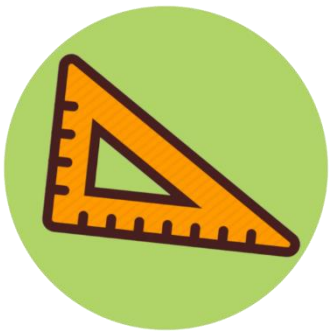

Figure 1 - The application's icon

2. Specification

The whole project was created in Unity Engine and the scripts are written in C# programming language.

To get augmented reality included, AR Foundation, AR Subsystems and ARCore XR plugins were imported. They provide objects and scripts ready to be put in a project that allow to add both marker-based and marker-less AR features.

In this case, there were needed AR Session and AR Session Origin objects. The second one was supplemented with AR Plane Manager and AR Raycast Manager scripts. The plane prefab contains AR Plane and AR Plane Mesh Visualizer. Thanks to these components, it is possible to detect planes and generate a visible mesh that indicates the play area for the user, who can then cast rays by touching the screen and place points on the surface.

When it comes to instantiating objects, first there is a marker that can be moved around the detected plane which purpose is to indicate for the user where the object will appear after pressing the button.

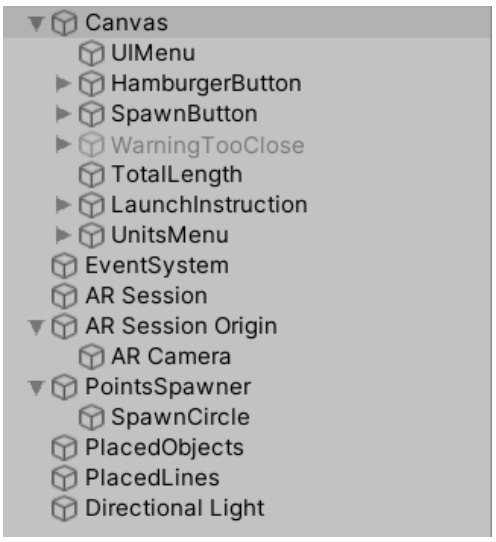

Figure 2 - The project's hierarchy

3. Application UX

As said before, the main task is to measure distances in the real world using AR technology and it can be performed on any device capable of working with ARCore. Due to these facts, it can be said that the application is targeted to a very broad spectrum of people and there are no restrictions or recommendations who either should or should not use it.

This being said, it becomes even more important for the product to be as user-friendly as possible. It is assumed for the user to be familiar with the mobile device they use and how the general mobile applications work.

The whole interface can be divided into two parts: the view and the menu. The first one is basically what the device’s camera registers and where the result of the operations is seen. The other one is where all the functional buttons are located.

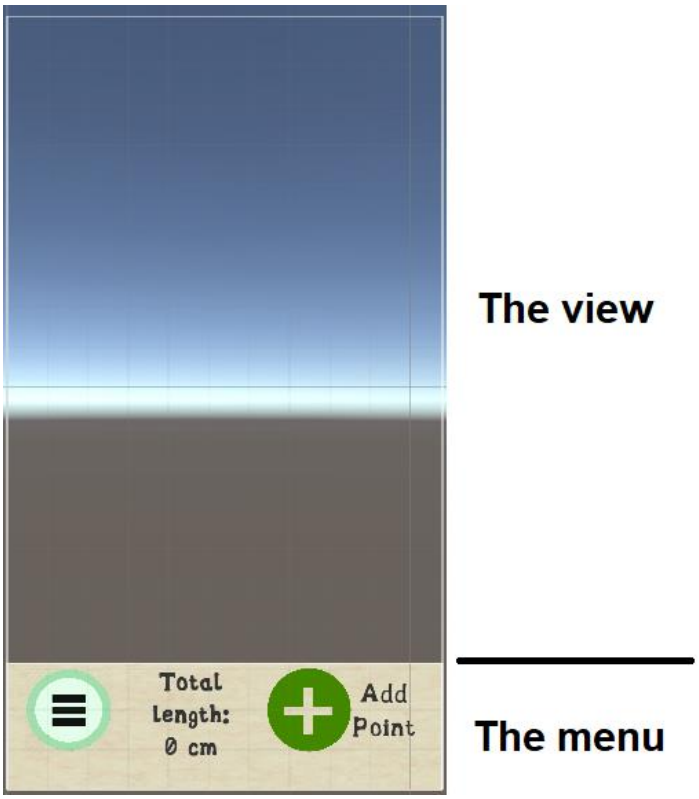

Figure 3 - The application's interface parts

Just after starting the application, a quick guide is visible in the view part. It stays there until it is pressed on. Then, when a surface is detected, an orange plane with pluses and dots pattern and red borders appears. Then, the user can press on it anywhere, causing a light blue circle to be moved to that position. After the spawn button is pressed, a small diamond-shaped object is created on the circle position and indicates where the chosen point is located. If there at least two of them, a green line with its distance written over it is created and connects them.

The most important button is the spawn button that is responsible for generating the new points in the chosen position. It is located in the right part of the menu. To the left, close to the center, there is a text showing the total measured length and to the left of it, there is a hamburger button. After pressing the last one, five other buttons are displayed / hidden. These are:

- Exit button – for closing the application
- Plane toggle – for enabling and disabling the planes
- Sounds toggle – for enabling and disabling the sounds
- Units button – for opening the menu where the user can choose what unit they want to see the measurements in
- Clear button – for deleting all the points and measurements, allowing to start from the beginning

4. Implementation and prototyping

Besides scripts that come from plugins mentioned before, there are also 9 other ones that are created specifically for this application. All the logic is divided into reasonable parts that are connected with each other. Most of the SOLID principles were kept in mind while writing the scripts, which are the following:

- ChangeUnits – for opening the units menu and getting input from units buttons that is being sent to PlacePoints script
- DisableInstruction – for disabling the launch instruction after pressing on it
- ExitApp – for closing the application
- HamburgerOpener – for displaying / hiding the menu buttons
- LabelRotator – for rotating the labels (measurement values above the lines on the planes) towards the user
- PlacePoints – the core script containing the most important parts, responsible for moving the marker, instantiating both points and lines, calculating the distances, changing the units and clearing all the measurements
- PlaneToggle – for disabling / enabling planes
- ScaleUIMenu – for scaling the menu part of the screen so that it is always the bottom 1/6 of it
- SoundsToggle – for enabling / disabling sounds in the application

The whole system is written in a modular way, which makes it quite easy to modify or add more to it. Comments and naming in code should make it easier for the reader to understand its logic.

Throughout the whole process, all the new changes were constantly tested to make sure that all the functionality works as intended and there are no conflicts. Due to the type of the application, a build was needed to be created to check how the whole logic works together each time. Every feature was added and tested one after another until the final product was developed, and the biggest milestones were:

- Setting up an AR-ready Unity project
- Detecting planes and placing the marker on them
- Instantiating points and lines
- Calculating distances and displaying them
- Removing all the spawned items and changing the units
- Other (plane toggle, audio toggle)
- Final version of all the models and graphics

Most of the assets used in the application was gotten either from Unity Asset Store (fonts, 3D models) or the internet (textures, other graphics). As there are not that many objects used in the system and they are also quite simple, it was rather easier and surely a way faster to obtain them from some source than to create them.

The assets that are crucial are the point, the line, the plane and the marker. They are a visual representation of the core functionality and the system could not perform its task without them. All the other ones are kind of an addition to make the whole experience more appealing and intuitive for the user.

As said before, every interaction (except placing the marker and disabling the launch instruction in the beginning) is performed by the user by pressing buttons that can be found in the bottom menu part of the screen. An explanation on ScaleUIMenu script might be needed. It is important to have the menu at 1/6 height of the screen because it is the area that blocks raycasting to the AR plane. This is done to prevent the user from spawning a point in the wrong place if there is a plane behind the spawn button, causing to move the marker on press.

All the functionality is simple enough to be assumed that it does not need an additional explanation to it, as the names combined with appropriate symbols should be intuitive for most of the users.

On top of assets, there are also a few animations which are supposed to make all the transitions look better and let the user to notice a change happening on their device’s screen.

5. User path

This part of the report purpose is to explain the flow of the application for the reader both in words and graphically.

The first screen that is seen after launching the app contains a quick guide explaining briefly how the system works. After the user presses on it, the message disappears and the user needs to scan the surroundings in order to find a plane that can be detected:

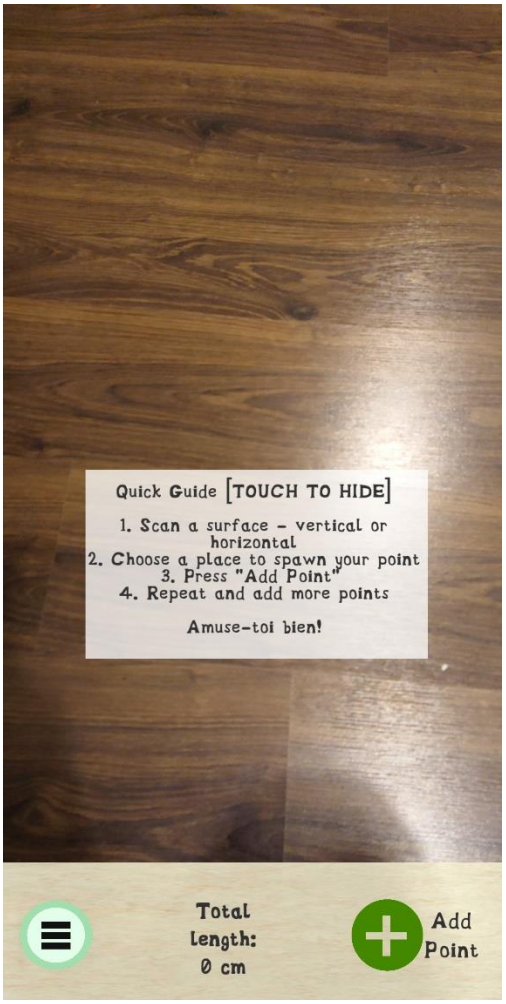

Figure 4 – The launch instruction

Figure 5 – Plane detection

Let’s assume a scenario that we want to measure the length of the radiator from the photo above. We, as the user, can touch anywhere on the plane to place the marker and when a position is chosen, a point can be instantiated by pressing the “Add Point” button:

Figure 6 – Placing the marker

Figure 7 – Spawning the first point

We can notice that a diamond representing our point is spawned. The next step that should be taken is setting a second one which lays on the other side of the distance that is being measured, so the other side of the radiator in our case. We can notice a line connecting the two points is spawned, as well. This way the task is already completed and we can just continue adding more points, if we wish to:

Figure 8 – Spawning second point

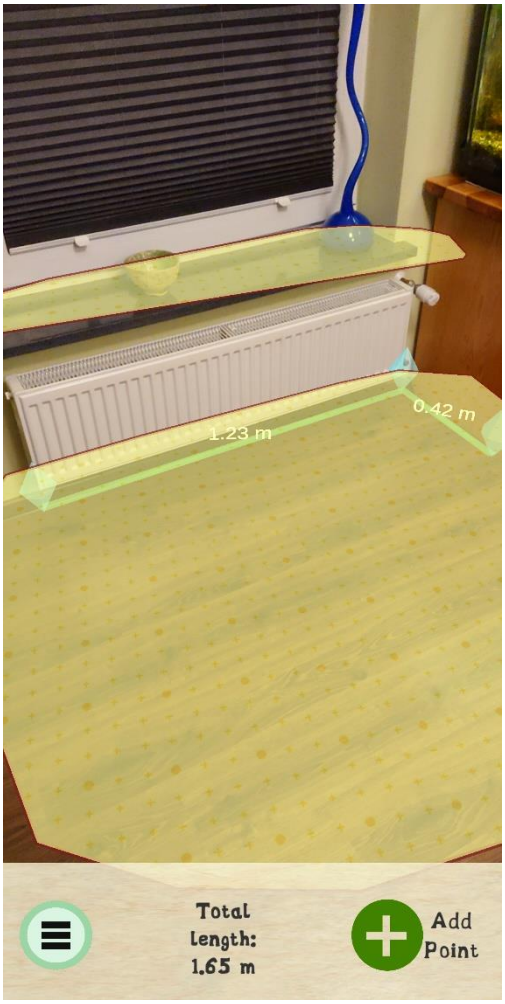

Figure 9 – Spawning more points, changing units

When there are more than just two points, another feature comes in handy In the middle of the menu part, there is total length value that is updated with every next point spawned. As shown above in the right picture, the distance is in a different unit – meters instead of centimeters, as before. This can be adjusted in the menu, where there are four units in total: milimeters, centimeters, decimeters and meters. Another feature that can be helpful when the user for example might want to take a photo of the measurements is the option to turn off the planes:

Figure 10 – Units menu

Figure 11 – Turning off the planes

When it comes to restrictions, besides the fact that the user can obviously place points only on a plane, the points are not allowed to be too close to each other. The minimum distance is set because of the fact that below it the measurement is too inprecise. When the user tries to place a point too close, an appropriate warning appears for a few seconds. Moreover, there is also an option to remove all the measurements and to turn off the sounds of the button clicks. The open menu looks as follows:

Figure 12 – The warning

Figure 13 – All the menu buttons

These are all the possibilities the user has in the system and the last not mentioned button is the one closing the application.

6. Summary

Although the plane detection algorithms sometimes can struggle with detecting planes correctly, most times they manage to do their tasks if the surface is not too shiny and there is enough light. Because of this the measurements can differ from the reality slightly but they are accurate enough to let the user know the real approximate value, especially in cases, when the length is neither too small nor too big.

During the testing the results gotten from the application were compared to the ones from a classic measure tape and this way it was made sure that the values are correct.

To conclude, the whole system is a result of 2-weeks work and therefore its purpose is quite simple and not overcomplicated. Nevertheless, all the requirements are met and the product is a fully functional application that can be used in real-life scenarios, so the whole project can be considered as successful.