UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Karl - Walter Sillaots

Önwall AR Tool

Bachelor's Thesis (9 ECTS)

Supervisor:
Raimond-Hendrik Tunnel, MSc

Tartu 2019

Önwall AR Tool

Abstract:

This thesis describes the development and testing of an Android augmented reality application for the company Önwall. The application allows a user to view a room with Önwall's mosaic tiles virtually placed on a selected real wall. Alternative applications are first analyzed and compared in the thesis. Then the user experience design of the requested application is developed and described, required features specified. After that, Unity and ARCore are brought up as the tools used to develop the application. This is followed with the details about the architectural implementation of the application. The thesis ends with descriptions of the user testing, found usability issues and potential future improvements.

Keywords:

Augmented reality (AR), mobile, application, mobile app, smart device, 3D graphics, computer graphics, Android, Unity, ARCore

CERCS: P170 - Computer science, numerical analysis, systems, control

Önwall AR Tööriist

Lühikokkuvõte:

Antud bakalaureusetöös kirjeldatakse Android liitreaalsuse rakenduse arendamist ja testimist Önwall jaoks. Rakendusega saab vaadata ruumi, kus valitud seinale on paigutatud virtuaalsed mosaiikplaadid. Sarnaseid rakendusi on analüüsitud ja võrreldud. Järgnevalt on soovitud rakenduse kasutajakogemuse disain loodud ja kirjeldatud. Pärast seda tutvustatakse lühidalt Unity ja ARCore, millega loodi rakendus. Sellele järgneb rakenduse süsteemiarhitektuuri kirjeldus. Bakalaureusetöö lõppeb kasutajatestidega, leitud probleemidega ja võimalike edasiarendustega.

Võtmesõnad:

Liitreaalsus, mobiilirakendus, nutiseade, 3D graafika, arvutigraafika, Android, Unity, ARCore

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

1. Introduction

Augmented Reality (AR) is the technique of visually enhancing the physical world with virtually added objects in an AR application on a device. One method to implement this is by using the camera on devices such as smartphones or tablet computers that take the feed from the world as an input. Then, new information is overlaid on top of it and the result shown on the device's screen. This creates an *augmented* perception where the user can see objects that do not exist in the real world. According to R. T. Azuma's 1997 paper *A Survey of Augmented Reality (Presence: Teleoperators and Virtual Environments)*, the three characteristics that define AR are that (1) it combines real and virtual environments, (2) is interactive in real time, (3) is in 3D [1, 2]. AR characteristics are unique and AR technologies are continuously developed to be more accessible. Because of that, different companies see AR as a useful tool for advertising their products. Examples of such uses include placing virtual furniture in a person's home via AR, displaying instructions to the user while they are working, or placing down a virtual vehicle that the user can walk around and look inside [3].

Önwall is an Estonian company that creates and sells mosaic wall tiles that are made out of wood. These tiles form different unique patterns (Illustration 1).

Illustration 1. Mosaic tile, wall with mosaic tiles

The company would like potential clients to be able to preview how their room would look like with specific wall tiles before they make a purchase. Right now their options are to either take the customer to an area that already has specific mosaic tiles set up, or to use photo manipulation to manually add these tiles inside a picture provided by the customer. Since both these options are time-consuming for Önwall, the company started looking for alternative solutions and decided that an AR application would be the best option for them. The user of an AR application could actively move around a room, seeing a visual representation of what their wall would look like when Önwall's tiles would be placed on it.

During the work of this thesis, an AR application for devices that support the ARCore SDK (See chapter 4.1.2) was created[1]. It can detect walls using the device's camera and then display 3D models of mosaics on the detected walls. This allows the user to see what a wall would look like when specific mosaic tiles are placed on it.

Chapter 2 describes existing applications that are similar to what Önwall has requested for smartphones, but are not suitable for use. Chapter 3 describes the created design for the Önwall AR application, as well as setting up quality requirements for the final version of the application. Chapter 4 is about the implementation of the application and the tools used to achieve it. Chapter 5 goes through the tests that were done on the finished application. Chapter 6 goes through further improvements done on the application and what could be done to it in the future. Appendix I includes instructions on how to use the application. Appendix II includes images of the user feedback questionnaire that participants took during user testing (See chapter 5.2.3). Appendix III is a glossary of terminologies that were used but not defined within the thesis.

[1] https://developers.google.com/ar/discover/supported-devices

2. Similar Software

Önwall wanted an AR application where the user can place 3D models of mosaic tiles on their walls. The main purpose for using AR is so the user could experience their room with virtual Önwall tiles on their wall. Placing and moving tiles around or changing their amount should be easy to learn for the user. Also, the application should support taking screenshots of what is currently displayed, so the user could save the result as an image to look at it later.

There are various AR applications available to use that perform similar tasks to what Önwall wants, such as interacting with the environment and placing objects inside it. In the following subchapters 3 applications are described and their functionality analyzed. For various reasons, these applications were found unsuitable. These applications were found in the Google Play Store or Apple Store and are the closest examples found to what the company has envisioned.

2.1 PictureThat

PictureThat[2] is an iOS[3] application created by the company Nano 3 Labs[4] and released in 2017. It allows the user to display images on their wall by selecting an image and placing their smartphone physically on the wall[5]. When those conditions are met, pressing the *Hang Photo* button places the image where the device is. Moving back with the device now shows a visible image on the wall through the device's camera (Illustration 2). Through the use of the Apple ARKit SDK[6], Apple's equivalent for Google's ARCore (See 4.1.2), the user can now view that image from any angle, and place new images on the same wall.

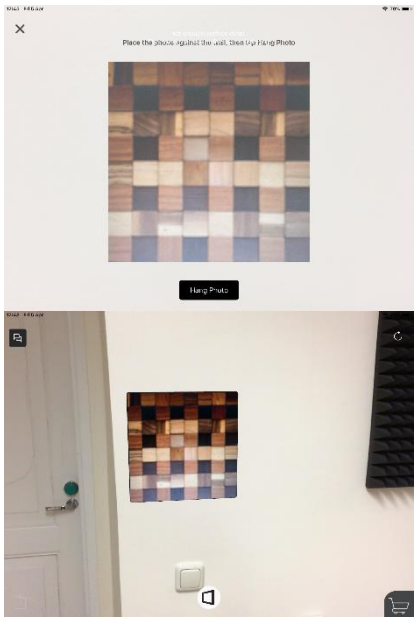

Illustration 2. Placing an image

There are 3 issues with this application that make it unsuitable for Önwall.

1. It only uses two-dimensional images. Önwall is using 3D meshes that are made to look like their tiles. Those would not be compatible

[2] https://www.picturethat.io/
[3] https://en.wikipedia.org/wiki/IOS
[4] http://nano3labs.com/
[5] https://youtu.be/CCGrhdP78MY
[6] https://developer.apple.com/arkit/

with PictureThat. Just a 2D image of a mosaic tile would not be able to convey its depth and shape.

2. This application does not support placing multiple images simultaneously in a grid format. This means that the user would have to place down and position each image on the wall manually one by one. This would end up taking too much time and the end result would be inaccurate (Illustration 3).

Illustration 3. Multiple images

3. This application has difficulties detecting texture-less walls, which is common for applications that use ARCore or ARKit. This means that it either will not be able to place images on certain walls, or the placed image might be displayed further from the wall than it is supposed to.

Based on these issues, this application would be too inconvenient for the user to use and the result would not be accurate enough to give an idea what the wall would look like with mosaic wall tiles placed on it.

2.2 ColorSnap Visualizer

ColorSnap[7] is a system of colors created by the company Sherwin-Williams[8] to help a customer decide on what color they would like to use in various projects. They have created an application to it called ColorSnap Visualizer, which is available on the web[9], iOS[10] or Android[11] platforms.

The main function of this application is to take any image and find colors on it that can be associated with this company's color codes. These color codes are then shown to the user. The

[7] https://www.colorsnap.com/
[8] https://www.sherwin-williams.com/
[9] https://www.sherwin-williams.com/visualizer
[10] https://itunes.apple.com/us/app/colorsnap-visualizer-iphone/id316256242?mt=8.
[11] https://play.google.com/store/apps/details?id=com.colorsnap&hl=en_US

web and, according to their advertisements, mobile versions of the application come with an additional feature called the *Paint a Scene* mode. This mode takes an image uploaded by the user and uses computer vision[12] algorithms to recognize edges of various objects that it can see. After that, the user can select a color from the system of colors available in ColorSnap to edit the color of a wall (Illustration 4).

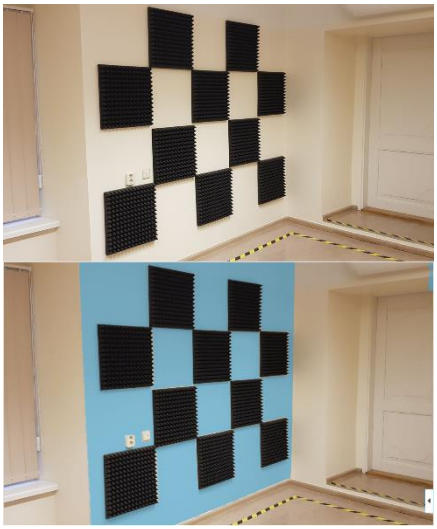

Illustration 4. Web version of ColorSnap

It would be possible to create an additional feature for this application to be able to use textures instead of flat colors. However, because it works on a single image, the Paint a Scene feature cannot be used to actively move around the room to see the results from different angles. Önwall's mosaic tiles are also 3D, not a simple texture or wallpaper, coloring the wall with a texture of the mosaic tile does not give the same representation as placing virtual 3D tiles on the wall would.

2.3 Augment and Augment Sales

Augment[13] is a company that has made an application with the same name for Android and iOS devices as well as a web version. Released in 2011 and with over 1 million downloads on the Google Play Store, the Android and iOS version is a general purpose AR application designed to visualize 3D models in a real world environment. Users can select any public 3D model uploaded in the Augment website and display it on the device, using the camera display as the background (Illustration 5). For a monthly fee, they can upload their own models either publicly or privately. In addition to this, they can choose to attach an object to a custom *tracker*, such as a pamphlet or a magazine. The object is

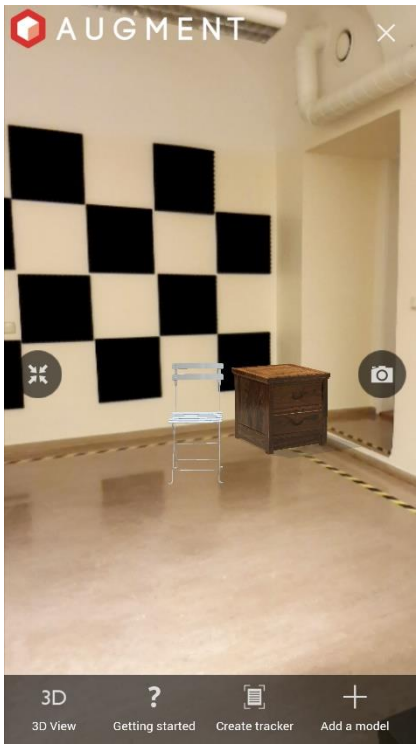

Illustration 5. Augment

[12] https://en.wikipedia.org/wiki/Computer_vision
[13] https://www.augment.com/

then displayed on the tracker when viewed through the device's camera in this application (Illustration 6). This is useful for smaller scale objects, as the tracker can be moved around, which moves the virtually created object on it as well. By paying a monthly fee, the application user can attach 3D models to a specific trackable, such as a custom pamphlet. Then the user's potential customer could take the pamphlet from the user. The customer could place the pamphlet anywhere they want to, and by using this application, the model is automatically made viewable to the customer.

Illustration 6. Trackable object in Augment

If Önwall decides to use a trackable to display the mosaic wall tiles, it would be time consuming for the end-user. Each trackable object would have to be manually placed on a wall for the created tiles to be displayed in a grid. Also, the Augment application might not recognize the trackable if it is too far away or at an angle, causing some 3D models to become displaced (Illustration 7).

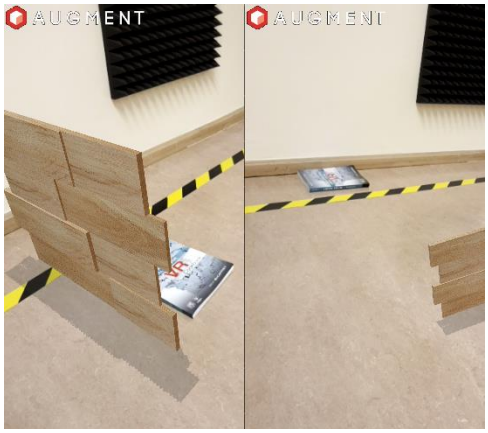

Illustration 7. Misplaced trackable in Augment

Without the trackable, this application would not work at all. Augment is designed to place objects on the ground. The height is not changeable. In addition to this, the user has to place each tile one by one, without any way to align objects next to each other. With no trackable objects to attach to, the user would not be able to move around the room without the created objects moving themselves.

In 2018, an alternative application named *Augment Sales* was released for Apple devices. This version requires users to pay for the monthly fee, and only supports using models uploaded by the user. Augment Sales uses ARKit to detect floors and place objects on it. Unlike the regular version of the application, once an object is placed down, the user can move around the area and each object would remain in their position relative to world position (Illustration 8).

Illustration 8. Augment Sales, placing tiles

As with Augment, Augment Sales can only place down objects one by one, with no ability to change the height. Another problem is that Önwall would have to send a person over to their client that has access to this application, while Önwall would like every customer to be able to use the application on their own.

3. The Design

Apart from the initial idea, Önwall did not have set requirements on how an AR application suited for their needs should work. The first goal was to search for various solutions to solve this. Two ideas were presented:

1. An application that can detect walls and directly place mosaic tiles on them. This approach would be easier to implement and use, but has issues with detecting unicolor walls (Chapter 4.2.2).
2. A solution created by Ott Saar, who also made an AR application as their Bachelor's thesis project [10]. This version focuses on detecting the ground instead of walls. The user would point their device at the starting location and click on a button to mark that area. Then they either raise or lower their device to indicate the height of the wall and click on the same button again. After that, clicking on the ground would build a virtual wall between it and the 2 previously created points. Overall, this is more accurate and works better in more environments compared to the first solution, but is more difficult to use.

In the end, Önwall decided to choose the first option, as being user-friendly was important for them even with limitations. The next part was to visualize the general operation of the application.

From this point on the following terminology is used:

- **Tile** – A 3D mesh that looks like a mosaic wall tile made by Önwall.
- **Panel** – An invisible rectangle that is used as a distinct portion of a wall to be filled with tiles.
- **Wall** – The real-world wall.

Chapter 3.1 is about the design created for the application, while Chapter 3.2 is about the quality requirements that the final product should fulfill.

3.1 Application Operation

As the application is being used, specific interactions with it can only be done when certain conditions have been met. Illustration 9 shows the general operation states of the application.

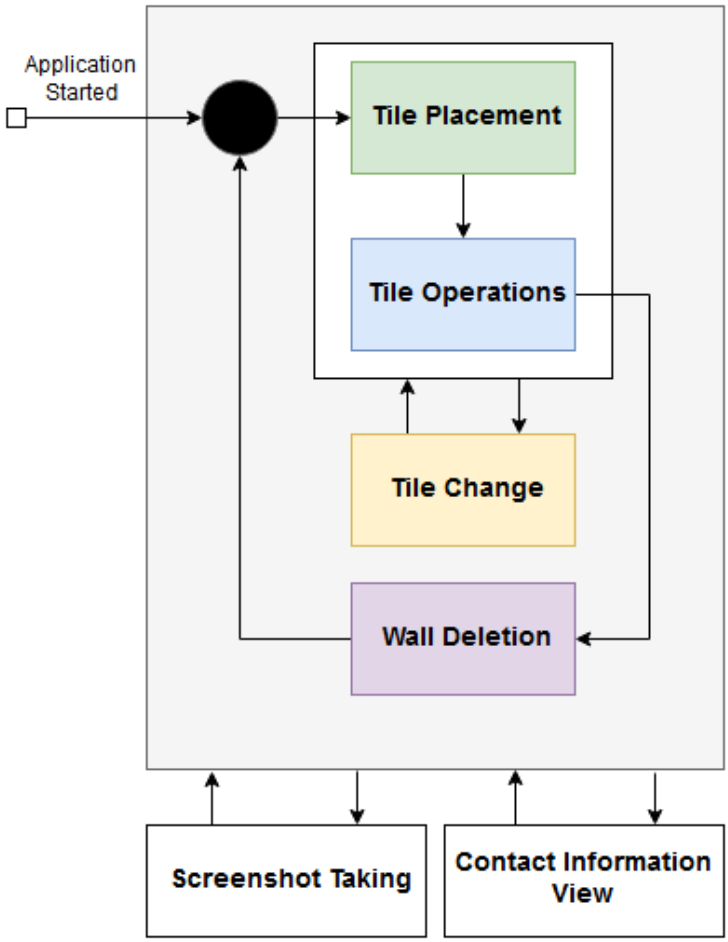

Illustration 9. Application operation

Chapter 3.1.1 describes how a user can place down a new tile in the *Tile Placement* state once the application has been started. Chapter 3.1.2 describes what the user can do with a tile in the *Tile Operations* state once the tile has been placed down. At any point, the user can change the currently used tile in the *Tile Change* state, which is described in chapter 3.1.3. Chapter 3.1.4 describes how to change the lighting of tiles. Chapter 3.1.5 shows how the user can delete their currently placed panel in the *Wall Deletion* state, if they want to place to another wall. Chapter 3.1.6 shows how the user can take a screenshot of their current view in the *Screenshot Taking* state.

There is also a separate button to view Önwall's contact details in the *Contact Information View* state, however, as it only contains their contact details and nothing else, it is not talked about more in-depth in this thesis.

3.1.1 Tile Placement

Illustration 10 shows how placing a tile on a wall should look like. The first screen depicts the user pointing their device camera towards a wall they want to place a tile on. The second screen shows that the application has recognized the wall and displays an overlay on where a tile can now be placed. When the user taps on the overlay, a panel is placed on the area the user tapped, a tile is placed on the panel and the wall overlay turns invisible.

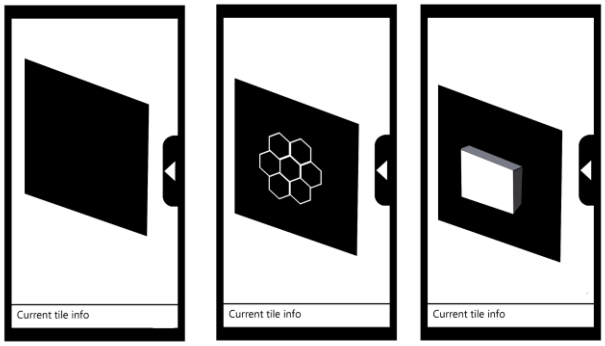

Illustration 10. Adding a tile

3.1.2 Tile Operations

Illustration 11 shows that the user can move the panel with tiles on it around by holding 1 finger on the screen, then panning it in any direction. When more tiles are visible, they are moved at the same speed as well.

Illustration 12 depicts operations when using 2 fingers. The first screen shows the user zooming horizontally with 2 fingers, which causes the panel to increase in size horizontally, until it can fit 3 tiles on it. Pinching 2 fingers together would reduce the panel size. The second screen shows the same operation, but vertically instead. The third screen shows the result, which is several tiles being visible to the user in a grid formation.

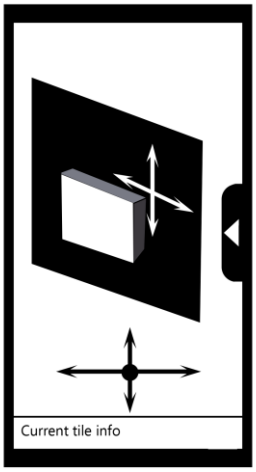

Illustration 11. Moving a panel

These two operations can be performed simultaneously when the user pinches or zooms 2 fingers diagonally. While the user is resizing the panel, the current tile info text at the bottom of the screen is changed to the current panel measurements. When let go, the information switches to the tile info after 1 second.

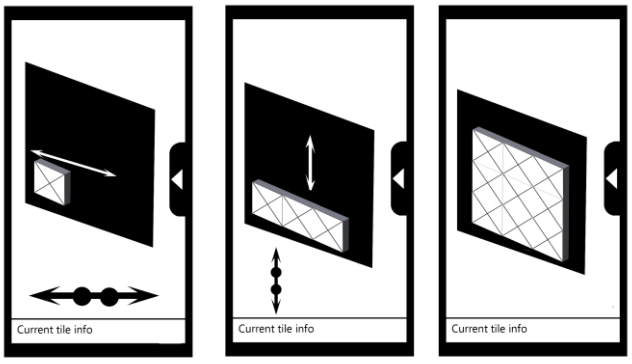

Illustration 12. Resizing a panel

3.1.3 Tile Change

Illustration 13 shows an arrow button. Tapping on it reveals a sub-menu with all selectable Önwall tiles available in this application. Each tile shows the name of the tile and its measurements. If no tiles have been created, tapping on a tile icon changes the default tile used in the application. When tapping a wall overlay after selecting a tile, a wall is created with the newly selected tile. If a wall already exists, selecting a new tile also replaces the current panel with a new one using the newly selected tile. When a new tile has been selected, its name and measurements replace the current tile info at the bottom of the screen.

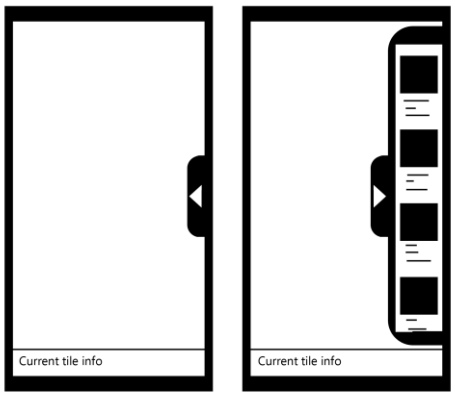

Illustration 13. Tile Change

3.1.4 Lighting

Illustration 14 shows a lightbulb button. Tapping it reveals a menu with 5 buttons. Clicking the middle lightbulb button rotates the lighting to the same rotation as the virtual camera is pointing to. When clicking any of the arrow buttons, the lighting is rotated by 2 degrees to the direction that the arrow shows – the right button rotates the light 2 degrees clockwise, the left button 2 degrees counter-clockwise, the up and down buttons 2 degrees upwards and downwards.

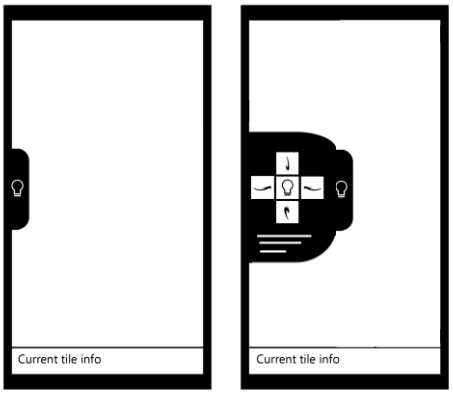

Illustration 14. Lighting

It should be noted that if both the Tile Change and Lighting menus are open, closing one closes them both. This is so that the user would not need to tap to close both menus separately.

3.1.5 Wall Deletion

The application supports only one wall at a time. Whenever a wall is created, a button to delete the wall becomes visible on the bottom of the screen. Illustration 15 shows that tapping this button deletes the panel and makes any invisible areas to place a tile on visible again. The application keeps track of what tile has been currently selected, so creating a new panel after a panel has been deleted creates a new panel using the same tile as before.

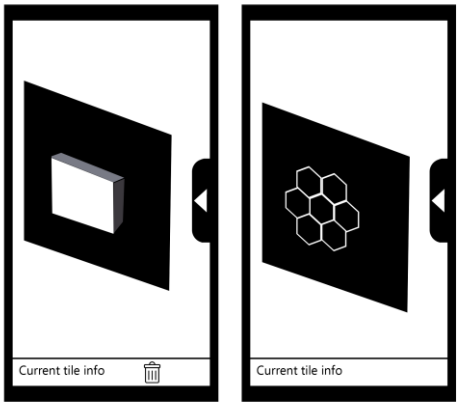

Illustration 15. Tile deletion

3.1.6 Screenshot Taking

The application supports taking screenshots. Illustration 16 shows a camera icon in the bottom right corner. Clicking it takes a screenshot of the current screen and initially saves it under the applications internal *files* folder. Considering that it would be difficult to find the image here, it is then moved over to the *Pictures* folder of an Android device inside a new folder titled *Onwall*. When the image is successfully moved, the user is notified that the screenshot has been taken at the bottom of the UI for 1 second, before switching back to the current tile info.

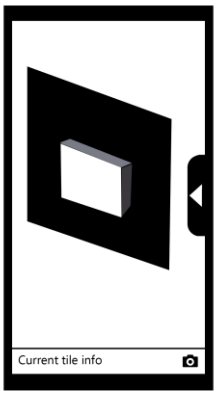

Illustration 16. Screenshot taking

3.2 Quality Requirements

Based on the requests from Önwall and the conditions the finished application would be used in, quality requirements were made to agree what the ideal conditions for this would be:

1. The application should take less than 5 seconds to start up.
2. The application should take less than 100 ms [4] to change the currently viewable tiles.
3. An average Android device user is able to place a tile on a wall in less than 1 minute.
4. The application works without issues in a subjectively well-lit environment.
5. The application works without issues on walls with textures (e.g. wallpapers).
6. The application works on any device that supports the ARCore SDK.
7. The application supports 10 different tiles to use.

These quality requirements cover the basic conditions for the application to be comfortable to use for any user. The results of these requirements can be read at chapter 5.4.

4. The Implementation

Once the quality requirements of the application, as well as the design, were set, the next goal was to start implementing the design. Chapter 4.1 is about the technologies used in order to create this application, the rest of Chapter 4 goes into more details on how each operation of the application in chapter 3.1 has been implemented.

4.1 Used Technologies

During the first discussion with Önwall, it was decided that the application should be developed for the Android platform, as according on Önwall, their target audience mostly uses Android devices. To develop the application on Android, two external tools were used: the Unity game engine and the ARCore software development kit (SDK).

4.1.1 Unity

Unity is a game engine made by Unity Technologies. This engine, while primarily designed to create 2D and 3D games, can also be used to create general purpose applications. It is also supported on a variety of different platforms[14] and frequently gets bug fixes and updates[15]. Products made with Unity are royalty free[16], however using Unity is free as long as the annual income of the user/company is less than 100000$[17].
Unity was chosen as the environment to develop the AR application for this thesis. An alternative would have been to use the Sceneform framework, which was created specifically for developing AR applications on Android devices. Unity was preferred since it is easier to create a user interface in it and work with 3D meshes.

4.1.2 ARCore

ARCore is Google's current platform for building augmented reality applications primarily for Android devices.[18] When using ARCore, it actively tracks the device location as well as uses the device camera to detect the size and location of surfaces, which can be interacted with.

[14] https://unity3d.com/unity/features/multiplatform
[15] https://unity3d.com/unity/whats-new
[16] https://unity3d.com/unity/faq
[17] https://store.unity.com/
[18] https://developers.google.com/ar/discover/

ARCore itself is supported on a wide array of devices[19] and is usable as an SDK for Java, C, Unity (C#), Unreal Engine and, to a limited degree, iOS.

Before ARCore, in 2014 Google released Project Tango as their first way to bring AR to smart devices. While more accurate than ARCore, only 2 commercially released devices supported Tango[20]. On 1. March 2018, Google shut down Tango and officially released ARCore [5, 6].

Alternatives for building an AR application would have been either ARKit, which is designed for Apple devices and mostly functions similarly to ARCore, or Vuforia, which can be used on both Android and Apple devices, but needs to rely on ARKit or ARCore to use its ground detection markerless functionality.

ARCore was chosen since Önwall wants the application to be usable on as many devices as possible and due to the amount of time needed to develop an application for 2 platforms, Android was settled as the final choice. This means that ARKit would not be suitable. As for Vuforia, while developing an application on it is free, deploying it would require Önwall to, at minimum, either spend 99$ a month or purchase the classic version for 499$[21].

4.2 Wall Detection

In order for the Önwall AR application to work, the application needs to be able to detect walls. The ARCore's built-in wall detection is used to achieve this. The following subchapters will talk about how this wall detection works and what limitations it has.

4.2.1 Detecting Planes

ARCore keeps track of every virtual object inside an application by using *planes*. Every virtual object is anchored to a plane by transferring the real world device movement and orientation over to the virtual world. However, the application needs to understand where to create a virtual plane such that it would accurately correspond to a real-world planar surface.

ARCore uses a process called concurrent odometry and mapping (COM) in order to understand where the phone is relative to its surroundings.[22] ARCore searches for distinct features in the captured camera image called *feature points* and uses these points to help calculate the device's position and orientation. These feature points can be, for example, visible corners on tables or patterns on the ground.

[19] https://developers.google.com/ar/discover/supported-devices
[20] https://en.wikipedia.org/wiki/Tango_(platform)#Devices
[21] https://developer.vuforia.com/vui/pricing
[22] https://patents.google.com/patent/US20170336511A1/en

In order to create a plane, ARCore searches for clusters of feature points that appear to lie on horizontal or vertical planar surfaces [7]. Usual examples of such surfaces are floors, tables and walls. Once it has recognized a valid surface, a plane is automatically generated and viewable to the user. Since the system is not able to immediately recognize the entire surface, the plane created on it may initially be of limited size. In this scenario, the user should move their device around the area, so that ARCore can detect new feature points and combine them together with either existing planes or create a new one.

4.2.2 Wall Detection Limitations When Using Planes

Since ARCore searches for distinct features in the captured camera image, it cannot detect unicolor surfaces that lack any features for the algorithm to use as feature points. The ground is somewhat less likely to have this issue, because it usually has a unique pattern or a rug on it. Plainly colored grounds are uncommon. While walls may be decorated with wallpapers, they can usually also be painted with one color. This means that it is likely that a user might need to do some additional work to make a wall detectable. This problem can be rectified by placing uniquely identifiable objects on the wall that differ from the surface color wise. One example would be using sticky notes or masking tape that the application would then recognize on a surface (Illustration 17).

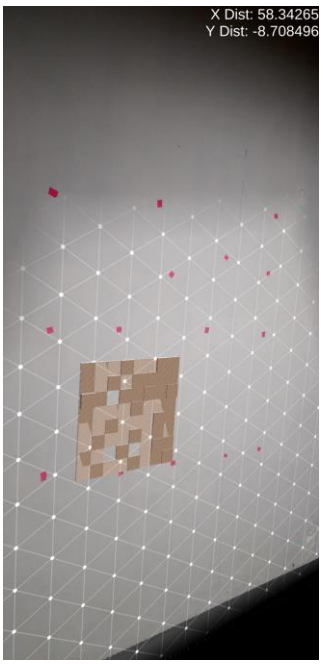

Illustration 17. Wall

4.3 Placing Tiles

Once the Önwall AR application has detected walls and placed planes on them, the next step is to create a panel. The following subchapters talk about how tracking objects with ARCore works, as well as the process of events to place down a panel with the first tile.

4.3.1 Object Tracking

As stated in chapter 4.2.1, ARCore searches feature points and uses these points to calculate the device's change in position and orientation. This visual information is combined with the device's inertial measurement units (IMUs) to estimate, in relation to the world, the camera position and orientation, also known as a *pose* (transform) in computer vision.
By aligning the pose of the virtual camera with the pose of the physical device's camera, virtual content can be rendered from the correct view. This rendered content is overlaid on top of the image that the device camera captures, giving the illusion that the content is a part of the world. This also means that the device can not differentiate if something from the real world is in front of the created content. So virtual content can still be incorrectly seen from behind occluding real world objects or walls.
In ARCore, planes and points are special types of objects called a *trackable*, which are the various feature points and planes that ARCore tracks in the real world. As ARCore is updating its information of the environment, the poses of scene objects should change to accommodate for these changes. For example, it initially detects the ground to be 5 centimeters above the actual ground and places a plane there, but when moving around the area, it detects that the ground is lower than the plane is. It then moves the plane closer to the ground. To ensure that every created scene object also moves when ARCore updates its understanding of the environment, an anchor needs to be defined for them. Virtual objects can be anchored to a specific trackable to ensure that the objects placed on a ground or a table will stay there, when ARCore adjusts the pose of these areas [7].

4.3.2 Object Placement

In order to place down an object, the user needs to tap on a plane that they can currently see. Illustration 18 shows what the process looks like.

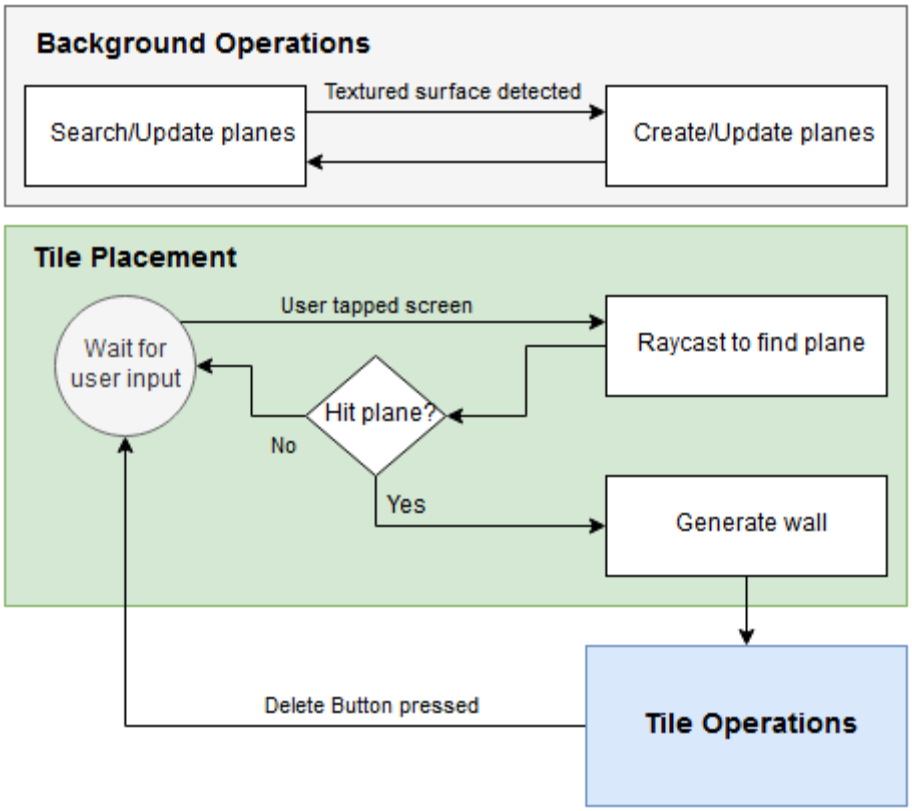

Illustration 18. Tile Placement

Whenever the user taps the screen, a ray is cast into the scene. If it does not hit anything, the application starts waiting for a new input. If it hits a detected plane, then the application collects the hit information, generates a panel with a tile in it, anchors the instantiated panel to the hit plane and turns all planes in the scene invisible. The tile itself is aligned vertically relative to the world, while the panel takes the vertical and horizontal measurements of the tile and sets it as its own measurements. After the panel is placed, all planes are turned invisible so that the grid used to display the plane would not distract the user.

4.3.3 Editing Panel Size

After a panel is created and a tile has been placed inside it, the application goes into the Tile Operations mode. The user can now move the panel around or increase/decrease its size, which causes the panel to be filled with as many tiles as possible. Illustration 19 shows what the process looks like.

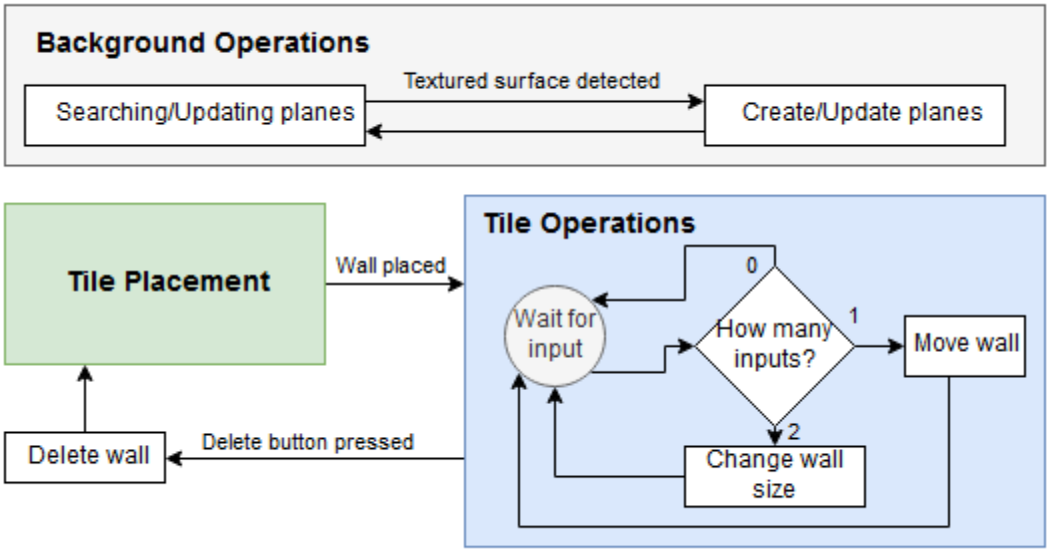

Illustration 19. Tile Operations

During the Tile Operations mode, the application is constantly waiting for input. When one finger is detected, the panel is moved around based on the direction the finger is moved in. For example, when the user moves their finger right on the screen, the panel moves right relative to the plane it is placed on. When two fingers are detected, the user can resize the panel by pinching or zooming their fingers. The horizontal and vertical size of the panel is used to calculate how many tiles can be fit inside the panel. The tiles are anchored to the bottom left corner of the panel. If the panel horizontal size is smaller or larger than the sum of the tiles horizontal size, a column of tiles is removed or added on the right side of the panel. The same thing happens with the vertical component and the topmost tile row.

As stated in chapter 3.1.2, the bottom area of the application displays the current tile information under normal circumstances. However, while the user is resizing the panel, the application displays the panel measurements in meters for the user to see. When the user is no longer resizing the panel, the information switches back to the current tile info after 1 second.

4.3.4 Changing the Current Tile Used

Whenever the user is placing down a tile or moving around/changing the size of the created panel, they might want to change the selected tile to another tile. Illustration 20 shows the logic for changing the current tile.

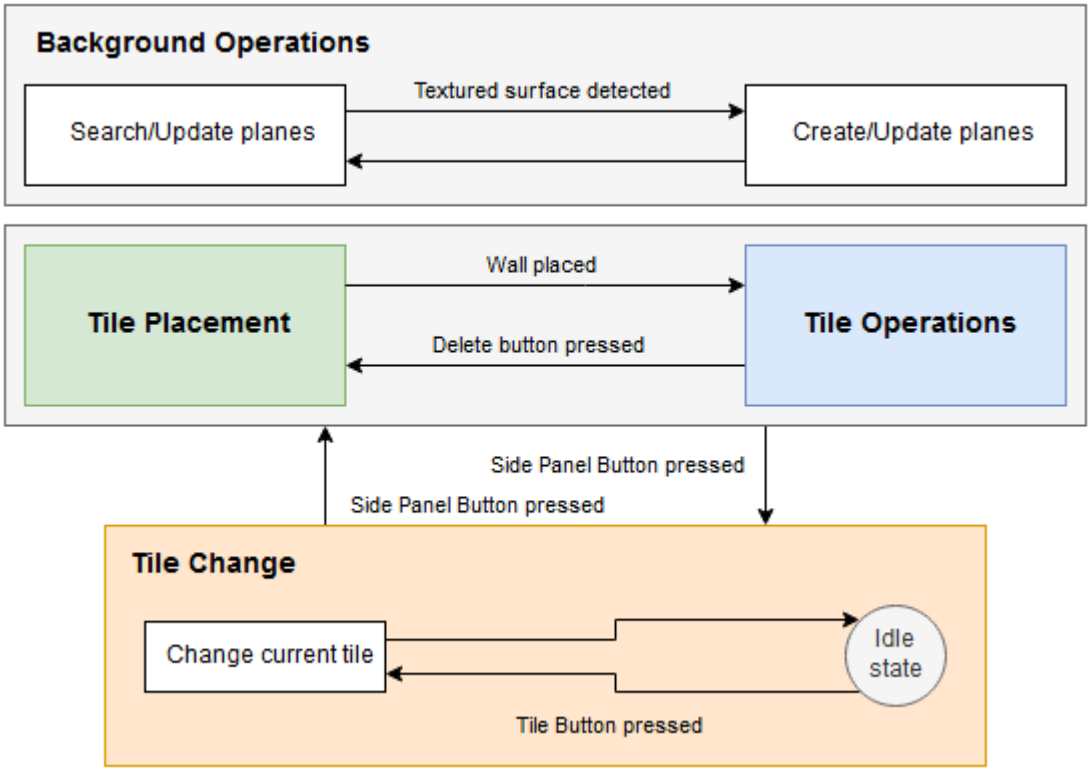

Illustration 20. Tile Change

Whenever the arrow button on the right side of the screen is pressed, the menu to change the current tile used is opened. While this menu is open, *Tile Placement* and *Tile Operations* modes are disabled, so that the user does not accidentally place tiles / move instantiated tiles around while selecting a new tile to replace the current ones with. There are 10 mosaic tiles to choose from, with an image, name and measurements. Tapping on a tile changes the tile used when creating a new wall, as well as replacing tiles on the currently instantiated wall. If the new tiles are of different size than the old ones, the application makes sure to fill the current wall area with as many tiles as it can in a grid pattern.

4.4 Lighting

Whenever tiles are placed down in the scene, they are affected by the built-in lighting that Unity provides. The user has the ability to change the orientation of this lighting to, for example, simulate light shining in from a real world window. Illustration 21 shows how to change the lighting.

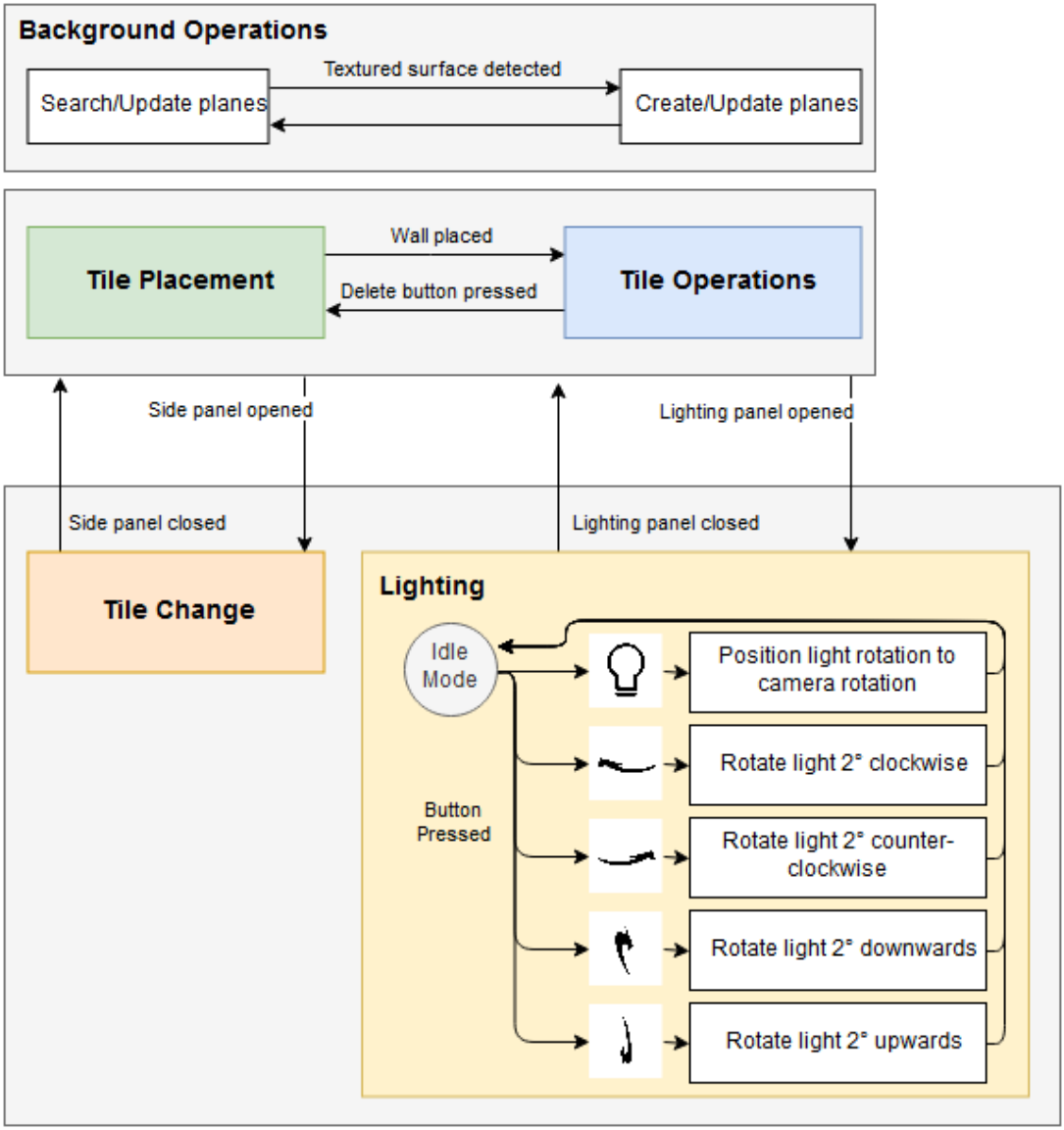

Illustration 21. Lighting

The lighting menu on the left side of the screen opens similar to the tile selection menu. The user can still open the tile selection or lighting menu if the other is open, but closing one closes them both.

The application uses a directional light source, which lights the area it is pointed towards. While the lighting menu is open, the user can click the lightbulb button to rotate the lighting based on

the virtual camera's rotation. Alternatively, they can use the arrows to manually change the lighting rotation 2 degrees at a time.

4.5 Other Commands

The user also has the ability to take screenshots of what is currently displayed on the device, as well as check Önwall's contact information at any point. Illustration 22 shows what the corresponding buttons do.

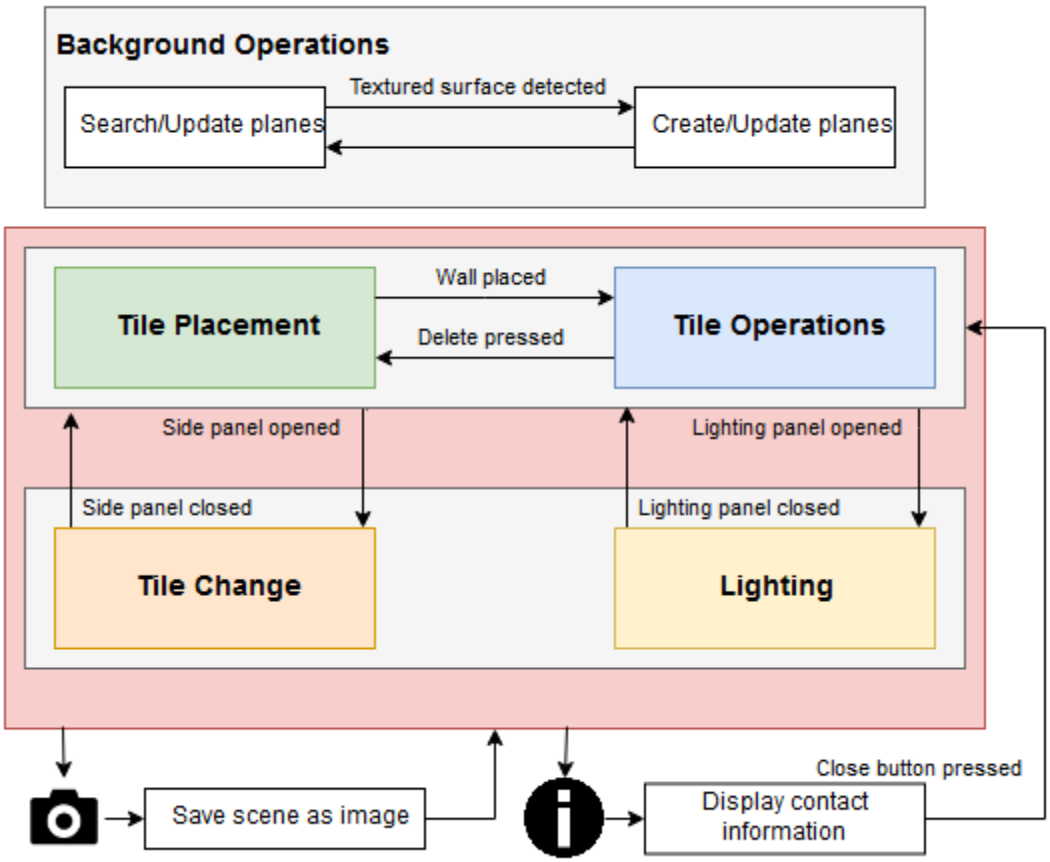

Illustration 22. Screenshot and Contact Info buttons

The screenshot button saves the current display into the device memory. Once the image is saved and the user has given permission for the application to access their device memory, the application creates a separate folder in the "Pictures" folder, where it moves over the saved image. As said in chapter 3.1.6 it then notifies the user that the image is taken at the bottom of the UI for one second. The contact information view shows information on how to contact Önwall.

5. Testing

As the AR application developed for this thesis is dependent on where and how it is used, there needs to be separate testing to determine the optimal environmental conditions for usage, as well as user testing to see how people understand and use this application. Chapter 5.1 is about testing the AR application in different environments, while chapter 5.2 goes through on what was tested during user testing, as well as the received feedback. Chapter 5.3 describes the platform testing done to see how the application works on supported devices. Chapter 5.4 looks back at the quality requirements in chapter 3.2 to see if these requirements have been met.

5.1 Environment Testing

This chapter talks about the ARCore built-in wall detection, introduced in chapter 4.2, and testing it with different textures, lighting and the maximum distance of feature points before the application starts having difficulties detecting a wall between them.

5.1.1 Textures

As stated before, ARCore has difficulties detecting unicolor walls, so the first test with textures was to see if it can detect minor texture changes. Illustration 23 shows a white surface with masking tape placed on it. Due to the lighting, the application had troubles detecting small changes such as this. Only at an angle did the application detect the masking tape, but it took a long amount of time before the lighting conditions were good enough for the application to detect it (Illustration 23).

Illustration 23. Hard to detect surface

A more distinct color, such as black, on a white wall yields much better results, as the application is able to quickly find the wall. Even then though, the application might mistake the wall to be further away than it actually is. By moving around and viewing the textured surface at different angles, the application corrects the created plane location (Illustration 24).

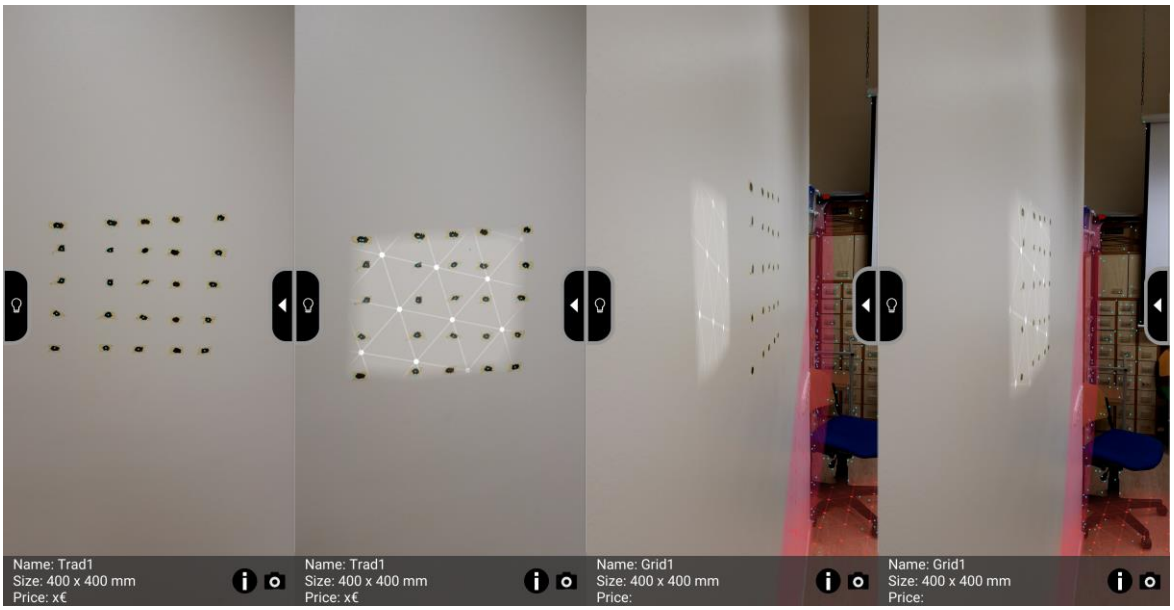

Illustration 24. Detecting a wall, correcting depth

The application can also detect uneven surfaces as walls, as long as there is a discernable pattern on it (Illustration 25). The application did not have trouble detecting walls in a room with unique textures in it (Illustration 26), but when creating multiple new planes, eventually old ones started being removed. It appears to have a limit of between 10-15 planes at a time, any more than that and the oldest detected planes are deleted first. This also means that if a panel has been placed down, then all the tiles would disappear while the application continues to detect new planes in the background. Since these planes are invisible, the user would very likely not understand what might have happened. On the other side, this is also somewhat unlikely of happening if the user would be using this application in only one small area, since there would not be enough detectable surfaces to reach this situation.

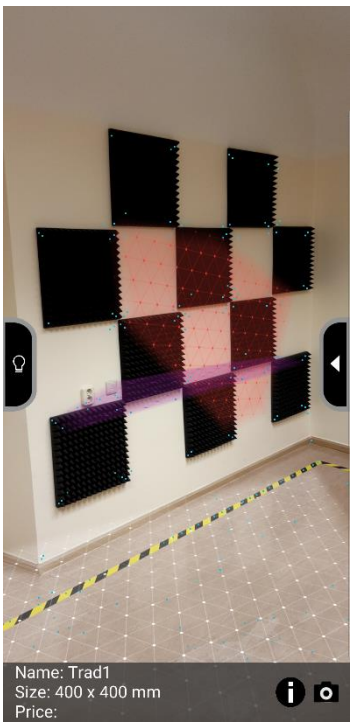

Illustration 25. Uneven surface

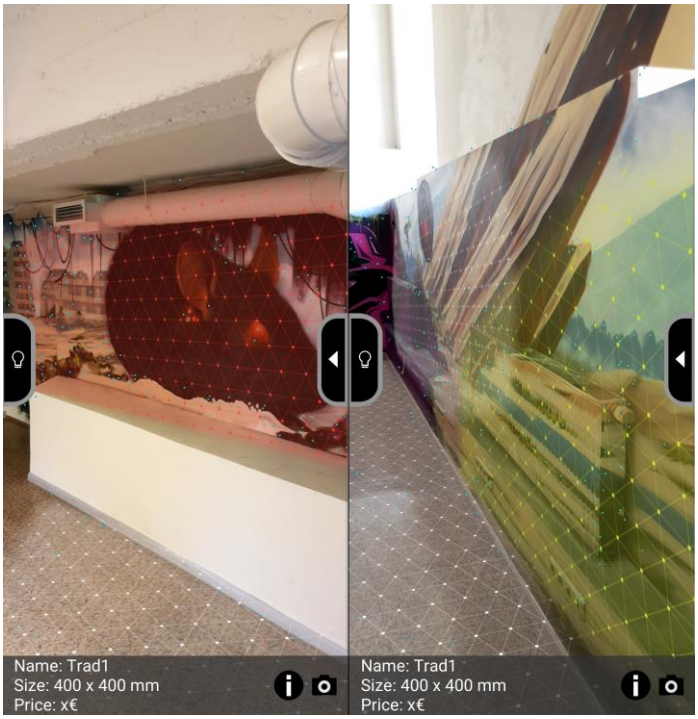

Illustration 26. Textured surfaces

In conclusion, as long as the surface is not unicolor or the textured wall colors do not look too similar to each other, the application should not have difficulties detecting a wall.

5.1.2 Lighting

Testing the application's wall detection capabilities during different lighting scenarios was done in the same room. The first test was done by having the room lighting turned on (Illustration 27, first image). The application had no issues detecting walls while the room was well lit. For the second test, the lights in the room were turned off, however a curtain was slightly opened to let a partial amount of light into the room. During this, the application still detected the wall without any particular issues (Illustration 27, second image).

Once all the lights were turned out and the curtains were covering any light from the outside coming in, the application did not detect the wall, due to it being too dark. However, while moving around, the application detected feature points a lot better, even while in the dark, which allowed it to detect the wall in the end (Illustration 27, third and fourth image). The amount of time needed to move around until the application detected the wall makes using the application in the dark unsuitable for use.

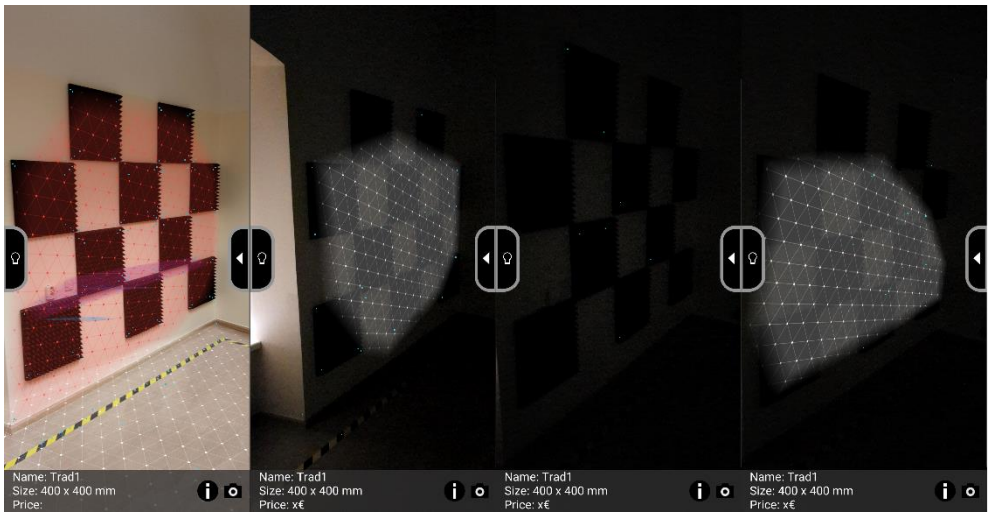

Illustration 27. Viewing the wall in different conditions

In the end, as long as there is even partial lighting, the application should be able to detect walls, but proper lighting is going to give the best results.

5.1.3 Distance

Under distance testing, 2 aspects were tested, how far away the device using the application needs to be in order to detect a surface and how far feature points can be from each other before the application starts having trouble detecting them.

When testing the distance from the wall, the maximum distance the user can be away from the wall was found to be between 1 to 2 meters. Any further than that and nothing is detected.

When testing the distance between feature points, the application needs at minimum 9 feature points to be able to create a plane between them (Illustration 28).

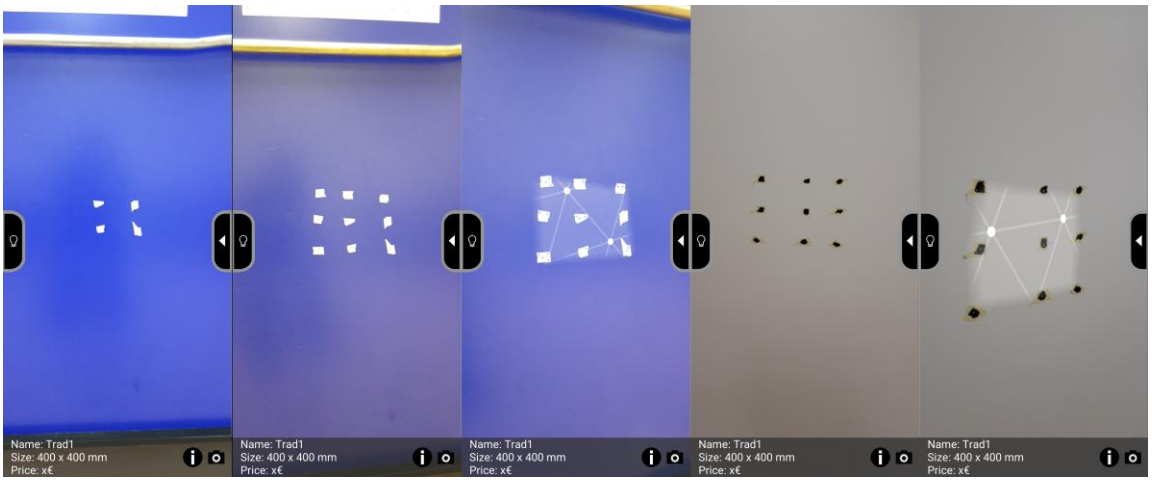

Illustration 28. Minimum points to detect a plane

Testing with a larger area, for the first test, feature points were placed with a distance of 10 cm between each point. When viewing the wall directly, the application did not detect the wall. Slightly moving over to the left however was enough for the application to recognize that it was viewing a textured surface (Illustration 29).

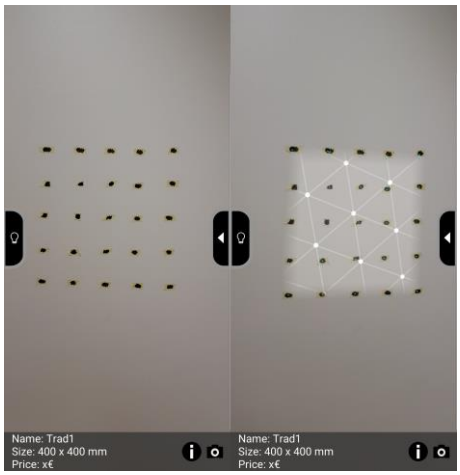

Illustration 29. 10×10 cm gaps

When placing the feature points 15 cm apart from each other, the application started to have more difficulties detecting the textured surface. The wall was only detected when viewed at a larger angle compared to the previous test (Illustration 30). Furthermore, once moving back to the initial location, the wall had disappeared, meaning that the application could only recognize the wall when viewing at this angle.

Based on this, if the feature points are too far away from each other, the application cannot detect a plane between them.

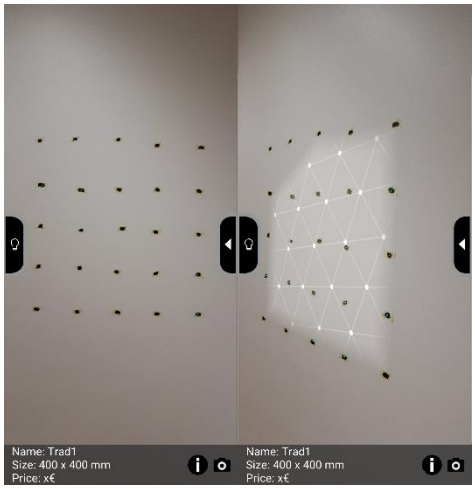

Illustration 30. 15×15 cm gaps

5.1.4 Discovered Limitations (Conclusion)

- The application detects distinct features better during movement
- It cannot find surfaces in the dark, unless there is very partial lighting and the user is moving around.
- It detects surfaces better at an angle, rather than when directly facing the wall
- If a plane is created, the user should view the surface at different angles, so that the application can align the surface to the wall
- If the application had problems detecting a wall when directly facing it, a plane created on it can still disappear when the user goes back to face it head on
- The application can keep track of between 10-15 planes at a time
- If the application has detected the floor beforehand, it has an easier time keeping track of walls
- The application starts having difficulties detecting planes when features points are more than 15 cm apart from each other.

These limitations were formulated as guidelines for the user of the application in Appendix User Instructions.

5.2 User Testing

While the Önwall AR application is designed with the intention of being easy to understand, the regular user might still have trouble with certain features, so the application was tested for feedback.

5.2.1 Testing Area and Tests

The user testing was done on a white wall that had been covered with pink masking tape, so that the wall would be detectable (Illustration 31).

Illustration 31. Test area

The test scenario consisted of 7 tasks and filling out a feedback questionnaire (Appendix II) once finished with the tasks. The tasks were:

1. Detect the wall using the application and place down a tile.
2. Resize the panel size to fit 3✕4 tiles.
3. Change the lighting so that it would shine from the real world light source.
4. Take a screenshot of the current result and find it on your device.
5. Delete the panel and create a new one.
6. Pick out the Oaknut tile.
7. Find the contact information button.

The next chapter shows the results from these tasks based on time and how each task was generally done by the participants, while the chapter after that goes into the feedback of the participants.

5.2.2 Test Results

For each participant, the time taken to solve each task is shown in Table 1.

Table 1. Test results based on time

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Total
Participant A	2:30	1:00	1:37	NA	0:15	0:03	0:01	5:26
Participant B	2:00	1:30	0:57	NA	0:05	0:07	0:01	4:40
Participant C	1:40	0:55	1:45	NA	1:23	0:12	0:01	5:56
Participant D	0:10	0:31	0:51	NA	0:22	0:20	0:04	2:18
Participant E	0:25	0:32	1:42	NA	0:41	0:26	0:05	3:51

None of the participants were told anything about the application, except that they will be using an AR application. The first task, having the application detect the wall and placing a tile on it, had the most varying results. Participants D and E are more regular smartphone users, while A, B and C, are not. Therefore D and E had a much easier time to figure out how to place down a tile, while the others either did not understand how the wall detection worked, or tried to use different methods for placing the tiles. Having some form of instructions could have given likely given better results.

Task 2, resizing the panel, went mostly well for everyone except for participant B, who did not think about zooming 2 fingers on the screen to increase panel size, but tried to drag 2 fingers in various ways instead. Having instructions for this as well would probably solve the issue.

Task 3, changing the lighting, was solved by everyone correctly, however none of the participants really understood how the lightbulb button exactly worked (changed the light rotation to the device camera rotation), nor noticed any major differences when tapping the lighting arrows. Apart from having instructions for this as well, the arrows would likely function better if they moved the lighting every frame while the button was being held down, instead of tapping the buttons many times.

Task 4, taking a photo and finding it on the device, was not finished by anyone. The taking a photo part was understood by everyone, there is a camera icon, and clicking it makes a flash in the screen, indicating an image being taken. However, finding the image was nearly impossible, due to a bug where every image displaying application on the device did not immediately show these newly saved images. The only way to find it was to go inside the device memory, and from there, the *Pictures* folder, to find the folder where all of the application images are stored

in. When checking later, all the programs are suddenly able to find these images, which means there might be a function that is supposed to notify these applications that there are new images to view and currently the device itself does it after some time has passed, when it should be notified of this upon the image being taken.

Task 5, deleting the panel and placing a new one, was easy enough for every participant. Participants C and E took considerably longer because they were giving feedback about various things during this point, but they did not have any troubles actually performing the task.

Task 6, selecting the Oaknut tile, did not cause issues for anyone. Task 7, finding the contact information button, was also not difficult, however, since 3 of the participants discovered the location of this button during the first task, they knew where it was immediately. The application could use an icon to make it more understandable that it leads to the contact information view.

In conclusion, most of the issues prevalent here came from the participants not being introduced to any form of instructions on how to operate with the application. Since one the goals stated in chapter 2 of the thesis was that this application would be easy to learn for the user, having a short user manual or in-application instructions would likely make it easier to understand for users. For this, appendix I briefly goes through all the functions of the application.

The feedback of these participants for the general design of the application is written in the following chapter.

5.2.3 Feedback

Once each test participant had finished all the tasks, they were asked to answer questions regarding their smartphone usage habits and feedback on specific features of the created AR application. Some questions also had additional fields so that each participant could write their reason for a rating (See Appendix II).The first question was how often the participant uses their smartphone, shown on Figure 1.

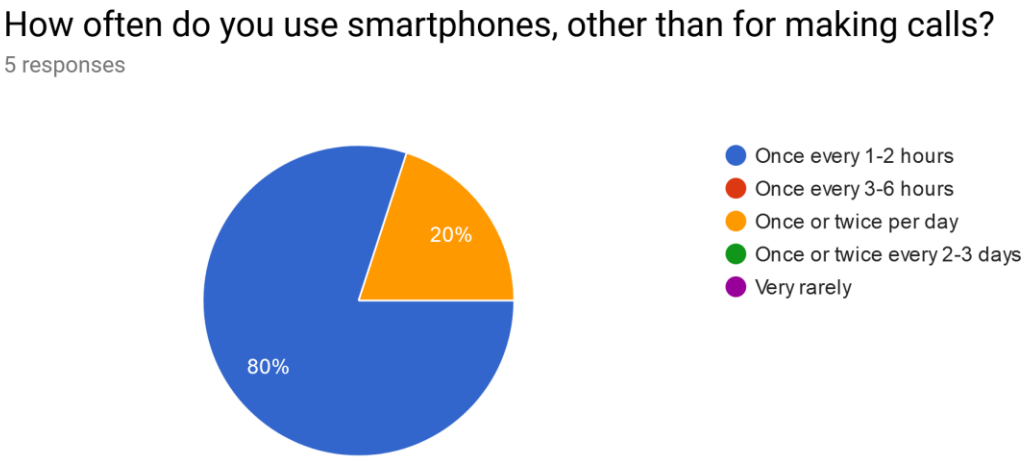

Figure 1. Smartphone usage

As can be seen in Figure 1, the majority of participants use their smartphone regularly every day. This means that most of the participants should be familiar with different smartphone functions and what to do with them.

Figure 2 shows what each participant commonly uses their smartphone for.

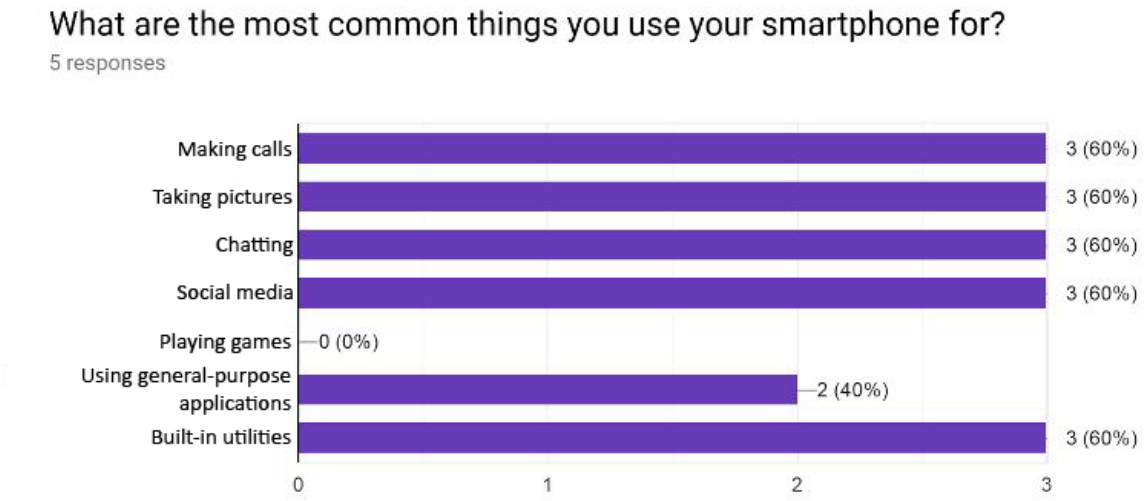

Figure 2. Smartphone most common uses

The results on Figure 2 are mostly balanced, each participant has specific needs when they are using their smartphone, but nothing that everyone has in common. This makes it more difficult to gauge a general level of skill that all participants have.

Figure 3 shows how often each participants uses AR applications.

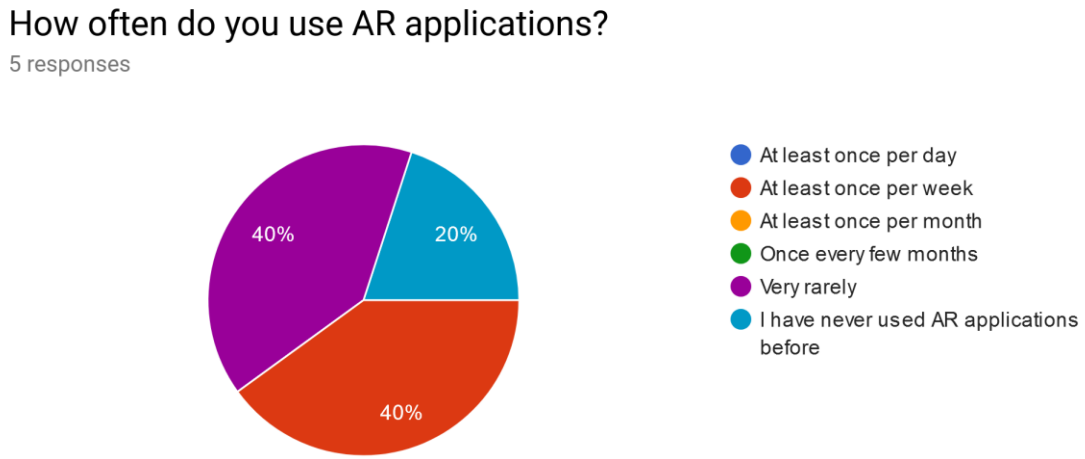

Figure 3. AR application usage frequency

Two of the five participants use some form of AR application semi-frequently, while the others are mostly unfamiliar with AR applications. This means that some issues are more likely to happen due to lack of experience on how AR applications work.

Figure 4 shows each participants opinion on how difficult it was to use the application, 1 being easy and 10 difficult.

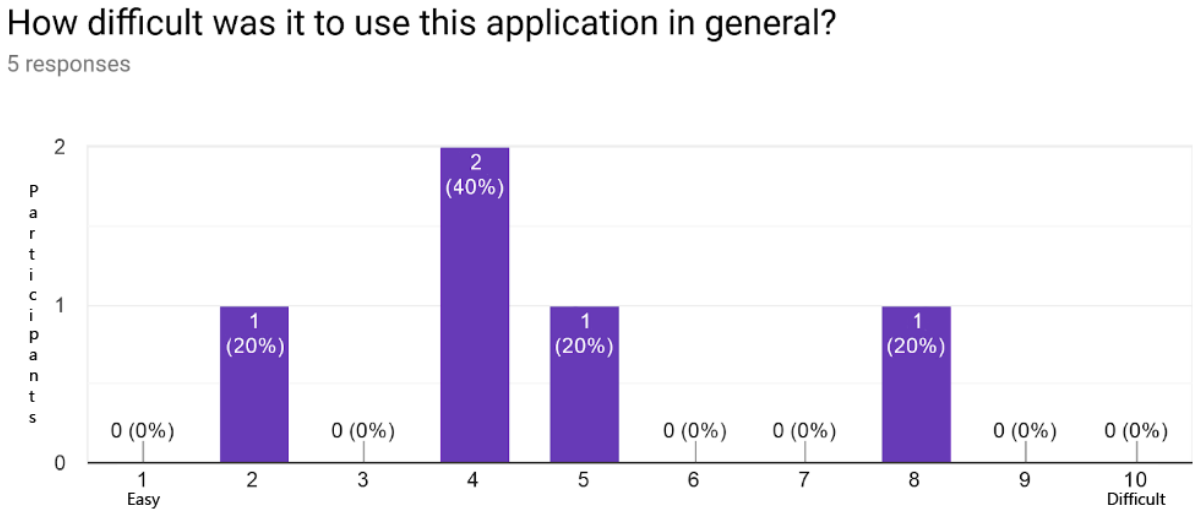

Figure 4. Application usage difficulty

One participant found the application to be easy to use, while others found some parts of the application to be difficult. While these details are gone more in-depth into in the following

questions, placing down the first tile and understanding how the lighting works were the main issues brought up here.

Figure 5 shows how difficult each person found the wall detection to be, 1 being easy and 10 difficult.

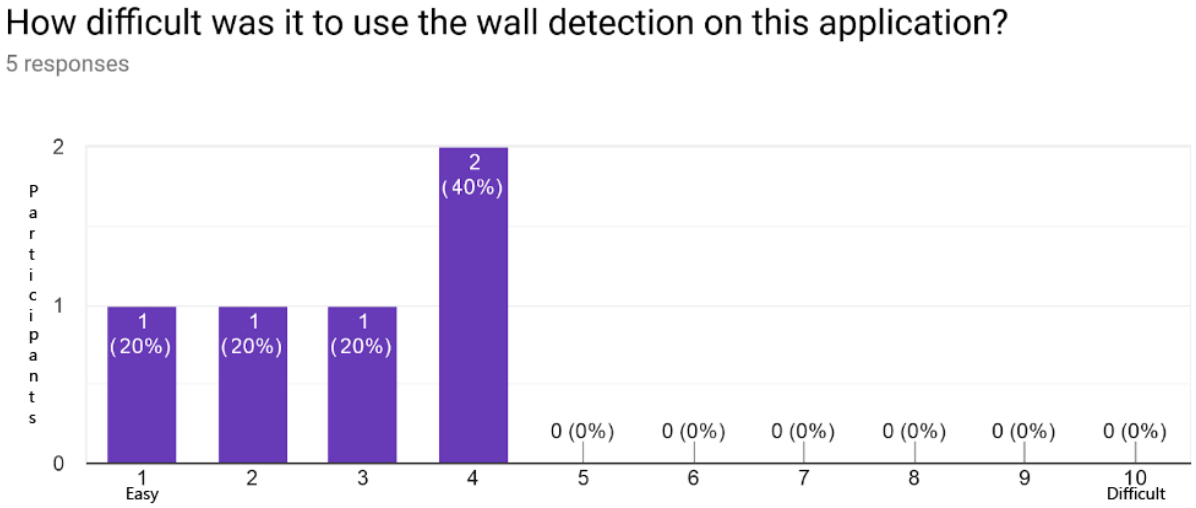

Figure 5. Wall detection usage difficulty

In general, none of the participants had too much difficulty in using the wall detection. Based on the feedback, the application should have some form of indicator to tell the user if they are not close enough for the application to detect a wall, since some participants did not think that they had to move closer to the wall for the application to detect the wall.

Figure 6 shows how much each participant liked changing the panel size and panel movement, 1 being they disliked it and 10 being they liked it.

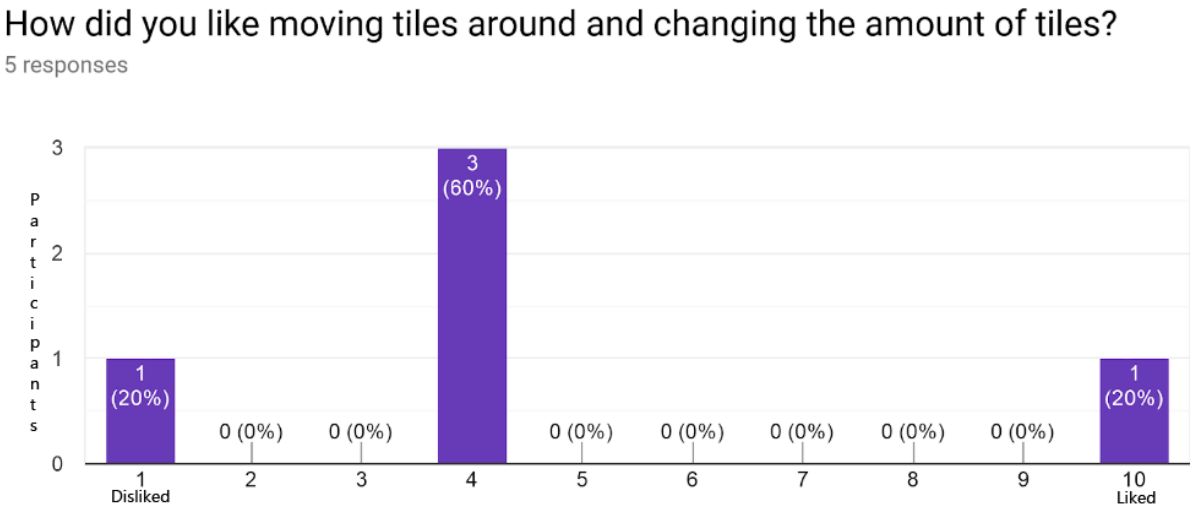

Figure 6. Tile movement and resize opinion

The results here were mostly below average. Once the participants understood how it works then it was fine, but figuring out how to change the amount of tiles at first was rather confusing for them. One suggestion was to swap moving the panel around to 2 fingers and resizing the panel with 1 finger. Another person did not like that all the tiles that were being displayed suddenly disappeared. This is due to the plane the anchored panel was attached to no longer being detected, so everything attached to the plane disappeared with it, until the app detected the plane again. This can be easily fixed by not anchoring panels to planes, however then the application would lose the ability to correct the panel positioning, when the application detects that the plane should be positioned differently than it currently is.

Figure 7 shows how each participant liked changing the lighting, 1 being they disliked it 10 being they liked it.

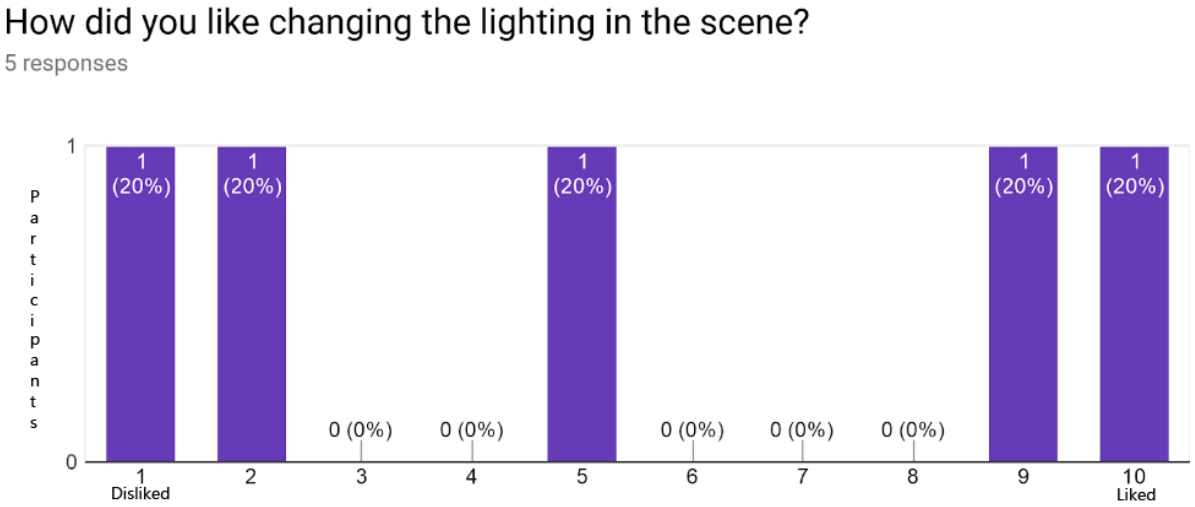

Figure 7. Lighting opinion

Some participants liked how the lighting direction affected the tiles, however, everyone had difficulties first understanding how the lighting works. None of the participants really understood that the lightbulb icon rotates the lighting at the same rotation as the device. Since arrow buttons move the lighting 2 degrees at a time, they also barely gave any kind of indication that the lighting had moved. There needs to be a clearer explanation on what the lighting buttons do, or an illustration of the light's direction.

Figure 8 asks each participant how they liked taking screenshots, 1 being they disliked it 10 being they liked it.

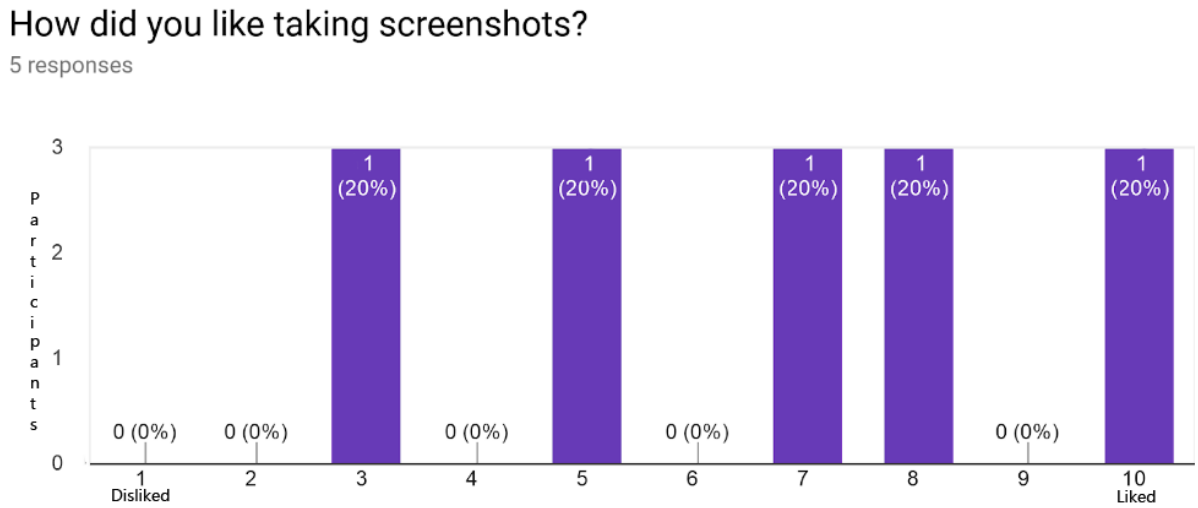

Figure 8. Taking screenshots opinion

Most participants did not have trouble taking screenshots, however, as stated in chapter 5.2.2, Task 4, the device did not recognize any newly created images, meaning they could only be

found by going into the device's internal memory. One participant suggested that the screen-shot button could be larger, as it was a little difficult for them to press it.

Figure 9 asks each participant how they liked the user interface (UI), 1 being they disliked it 10 being they liked it.

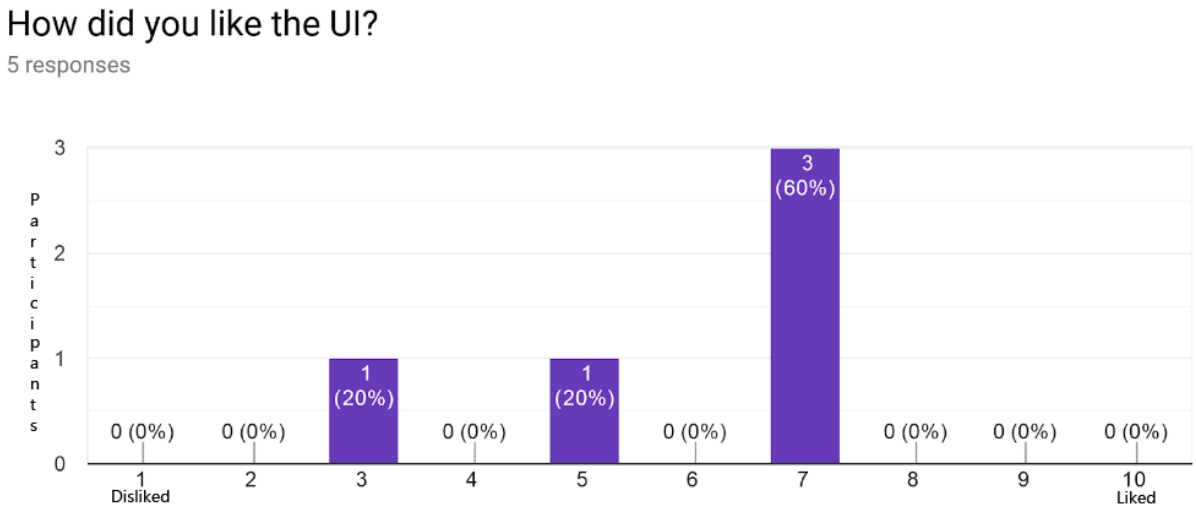

Figure 9. UI opinion

Most participants agreed that the UI is simple and easy to understand, but could use instructions on how certain parts of it function, such as the lighting. One participant also suggested having the Önwall logo be placed in one of the upper corners to understand what application is currently being used. Önwall did not find this necessary to currently add, as they already liked the minimalistic UI design.

Another participant decided to hold the testing device horizontally and found that the UI was too small when the application is used in this state. A separate scaling system should be made when the application is used horizontally, so that the UI remains the same size as when it is vertically.

Figure 10 and Figure 11 asks each participant how likely they would try out this application when they would plan to purchase Önwall's products and how they liked the overall experience with the application.

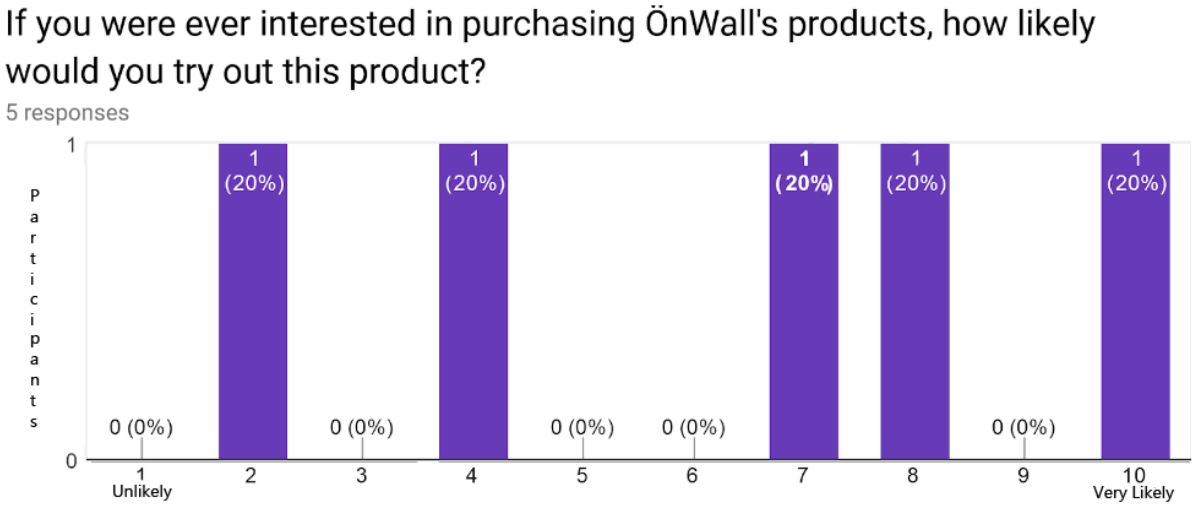

Figure 10. Likelihood of using the Önwall AR Tool

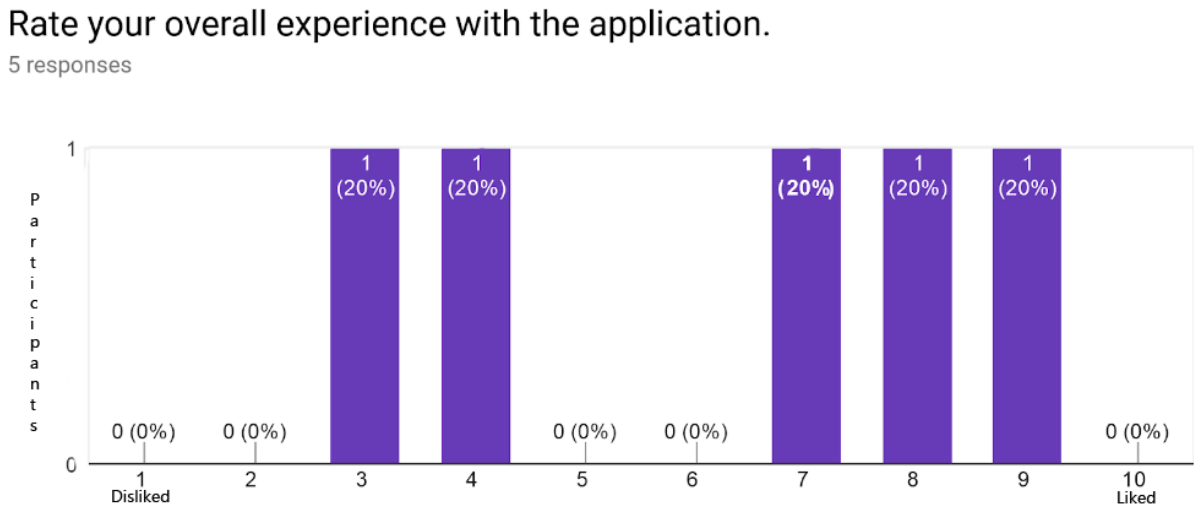

Figure 11. Overall experience

Overall, the application was seen mostly positively by the participants, but specific features should be improved on for a higher average. A few additional improvements were suggested after the testing had concluded. One thing was to improve the button reaction on click, as a few participants had longer taps when selecting tiles or changing the lighting which caused nothing to happen. Making the buttons react as soon as they receive input would fix this issue. The other thing suggested by a participant was to make any information changes at the bottom of the UI (screenshots, panel resizing) stay visible for a longer amount of time, as 1 second was not enough time for them to notice what was written.

5.3 Used Devices

The application was tested on different devices that support ARCore[23] to find out if these devices have any differences between each other when using the application. The following devices were:

- Samsung Galaxy Note 8 – The default device used to develop the application on, which will be used to compare with other devices.
- Xperia XZ1 Compact – Considerably smaller screen, which causes the panel resizing to be different due to it not being scaled with screen size, otherwise no noteworthy differences.
- Galaxy S7 Edge, OnePlus 5, Nokia 7 Plus – Similar measurements to each other, no noteworthy differences.
- Huawei Honor 10 – The camera display was flickering, which caused the application to have difficulties detecting the walls. It was not certain if the problem was related to the device itself or the application.
- Xiaomi Mi A2 lite – This device was not shown as a supported device on the ARCore supported devices webpage. The application installed successfully, but when starting the application, it requests to download ARCore. Accepting this takes the user to the Google Play Store page for ARCore, where it says this device does not support ARCore.

Based on these tests, the application should not have any problems when used, as long as the device supports ARCore and the camera on the device does not have any issues.

[23] https://developers.google.com/ar/discover/supported-devices

5.4 Quality

Looking back at the quality requirements in chapter 3.2, to see if their conditions have been met.

1. The application should take less than 5 seconds to start up.
   Result: 4.82 seconds, measured on the smartphone used for development of the application. This time mostly comes from the Unity's splash screen when starting up the application. This splash screen is used to load items in the background, but it is possible that the application might finish loading faster than the duration of the splash screen. The splash screen cannot be removed when making Unity applications for Android[24], so this cannot be improved on.
2. The application should take less than 100 ms to change the currently viewable tiles.
   Result: Depends on how many tiles are placed down. At around 25 tiles or more at a time, the time taken for a wall to be changed starts to take more than 100 ms.
3. An average Android device user is able to place a tile on a wall in less than 1 minute.
   Result: Based on user testing in chapter 5.2, if the user is less experienced with using smart devices, they are more likely to perform badly at this. This is due to the application not having any instructions to notify the user on what they should be doing.
4. The application works without issues in a subjectively well-lit environment.
   Result: No issues found.
5. The application works without issues on walls with textures (e.g. wallpapers).
   Result: No issues found.
6. The application works on any device that supports the ARCore SDK.
   Result: some devices need to have a more up-to-date version of Android, the ARCore supported devices page[25] has notes for these exceptions.
7. The application supports 10 different tiles to use.
   Result: 10 tiles are currently usable in the application.

Conclusion: mostly satisfying, but some form of instructions should be included with the application to reduce the overall average time to place tiles on a wall. If possible, the time taken to change the currently viewable tiles with new could also be improved on.

[24] https://support.unity3d.com/hc/en-us/articles/212165803-Why-can-t-I-remove-the-Splash-Screen-from-my-build-
[25] https://developers.google.com/ar/discover/supported-devices

6. Future Development

After testing had been finished, some minor improvements were made to the application:

- All buttons were changed to trigger as soon as they are held down. During testing, some participants had minor issues clicking buttons, so having these buttons trigger instantaneously should improve results.
- Based on the suggestion by one of the participants, whenever the information on the bottom of the UI is changed, it is now displayed for 2 seconds before switching back to the tile info.

In the upcoming 1-2 months, based on the feedback from the user testing and manual testing, the following issues will be addressed:

- Instructions – Either creating a separate manual that the user can read or making in-app notifications that tell the user what they should be doing.
- Lighting – Give the buttons a clearer indication on what they do, and change the arrow buttons so that they would continuously rotate the lighting when held down, instead of having to tap the button every time.
- UI – Make the screenshot, deletion and information buttons slightly larger, so that they would be easier to tap on.
- Tile Operations scalability, as right now, moving a panel with one finger moves it the same amount on every device, meaning users that have a smartphone with a smaller screen would have to let go of their finger more often.
- Improve UI scaling when the application is used on a mobile device horizontally.
- Research ways to make the device see newly saved images for other applications.
- Research ways to improve performance when changing the size of the panel and tiles are added or removed in the process.

Once these issues have been taken care of, most of the issues that the participants had would be significantly reduced. Any further improvements that could be made is to be discussed with Önwall in the future.

7. Conclusion

During the work for the present thesis an augmented reality (AR) application was developed for Android for the company Önwall. The name of this application is Önwall AR Tool. This application was made using the Unity game engine and ARCore SDK and can be used by most modern devices, usually needing only Android 7.0+ to operate. Different features and a user interface were added to make the application easy to learn for the user. The user can use this application to detect walls and place panels that are filled with mosaic tiles on these walls. The panel can either be moved around or resized, which moves the tiles inside it or adds/removes tiles based on the size of the panel. Other functions feature changing the lighting of the tiles, changing the current tile used, or taking a screenshot of the current scene.

Several types of tests were done to discover the limitations of the functions provided by ARCore, performance and compatibility issues. This includes testing the application on participants who were mostly familiar with using mobile devices, but had rarely used AR applications. The app was rated to be above average, however there were many areas that the participants were not fully satisfied with. The main reasons for this were a lack of instructions on how the application operates, as well as the lighting abilities not being clear to the user on what they do. Some minor issues were addressed after testing.

Apart from a minor scalability issue with resizing panels on devices with different screen sizes, the application works identically on any Android device that supports ARCore, due to it being developed in Unity. The performance between devices was nearly identical, only devices that had issues with their camera had difficulties working optimally. The major issues still left in the application were agreed with Önwall to be taken care of beyond the present thesis.

Learning how AR applications work was a big challenge, as well as figuring out how to develop one myself. Overall, I enjoyed learning how to develop an application in a field I was not familiar with. Special thanks go to my supervisor, Raimond-Hendrik Tunnel, who helped me out whenever I had any questions regarding the thesis, as well as Karl Nagel, head of Önwall, who offered the opportunity to do this thesis.

8. References

[1] Micheal Lanham. Learn ARCore - Fundamentals of Google ARCore. (30.03.2018)

[2] Ronald T. Azuma. A Survey of Augmented Reality (Presence: Teleoperators and Virtual Environments). (1997)

[3] Matt D'Angelo. Augmented Reality Check: Why Businesses are Embracing AR in 2018. (18.03.2018). [Online]. https://www.businessnewsdaily.com/9245-augmented-reality-for-business.html

[4] Jakob Nielsen. Powers of 10: Time Scales in User Experience. (05.10.2009). [Online]. https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/

[5] Brie Barbee, David Cogen. Tango vs. ARCore: Which is the future of augmented reality on Android? (17.12.2017). [Online]. https://www.digitaltrends.com/mobile/tango-vs-arcore-theunlockr/

[6] Jacob Kastrenakes. Google's Project Tango is shutting down because ARCore is already here. (15.12.2017). [Online]. https://www.theverge.com/2017/12/15/16782556/project-tango-google-shutting-down-arcore-augmented-reality

[7] Fundamental concepts of ARCore. [Online]. https://developers.google.com/ar/discover/concepts

[8] Alan B. Craig. Understanding Augmented Reality. (26.04.2013)

[9] Jonathan Linowes, Krystian Babilinski. Augmented Reality for Developers, (09.10.2017)

[10] Ott Saar. RoomMapperAR – A Mobile Augmented Reality Room Mapper

Appendix

I. User Instructions

Detecting walls

Point the device towards the wall you want to place mosaic tiles on. If the wall is unicolor, a pattern of 3x3 pieces of masking tape, sticky notes or something similar should be placed at minimum on the wall, up to 10 cm away from each piece. If the application still cannot detect the wall, try moving closer to it, view it from different angles or add more masking tape until it detects a distinct texture and generates a plane on it.

Placing a tile

Tap anywhere on a plane to place a panel, that starts with 1 tile.

Moving the panel

When a panel is created, hold down one finger on the screen to move the panel around.

Resizing the panel

To resize the panel, place two fingers on the screen, zooming your fingers will increase its size while pinching will reduce it. If the panel is large enough to fit more tiles in it, it will do so automatically. Same thing goes for when it is smaller than the total size of the tiles. The UI bar at the bottom shows the current size of the panel while resizing it.

Changing tiles

To change tiles, press the arrow button on the right side of the screen. A menu with all selectable tiles are presented in a scrollable bar. Pressing a tile button will automatically change any tiles to the newly selected one.

Deleting the panel

Whenever a panel is created, a button for deleting it becomes visible. Tapping that button deletes the wall and makes all planes visible again.

Screenshot

To take a screenshot, press the camera button on the bottom-right corner of the screen. A flash will indicate that an image has been taken. The UI bar at the bottom notifies the user once it is successful.

Contact

To view the contact information, press the “i” icon at the bottom right corner. The menu can be closed with the “close” button.

II. Feedback Questionnaire

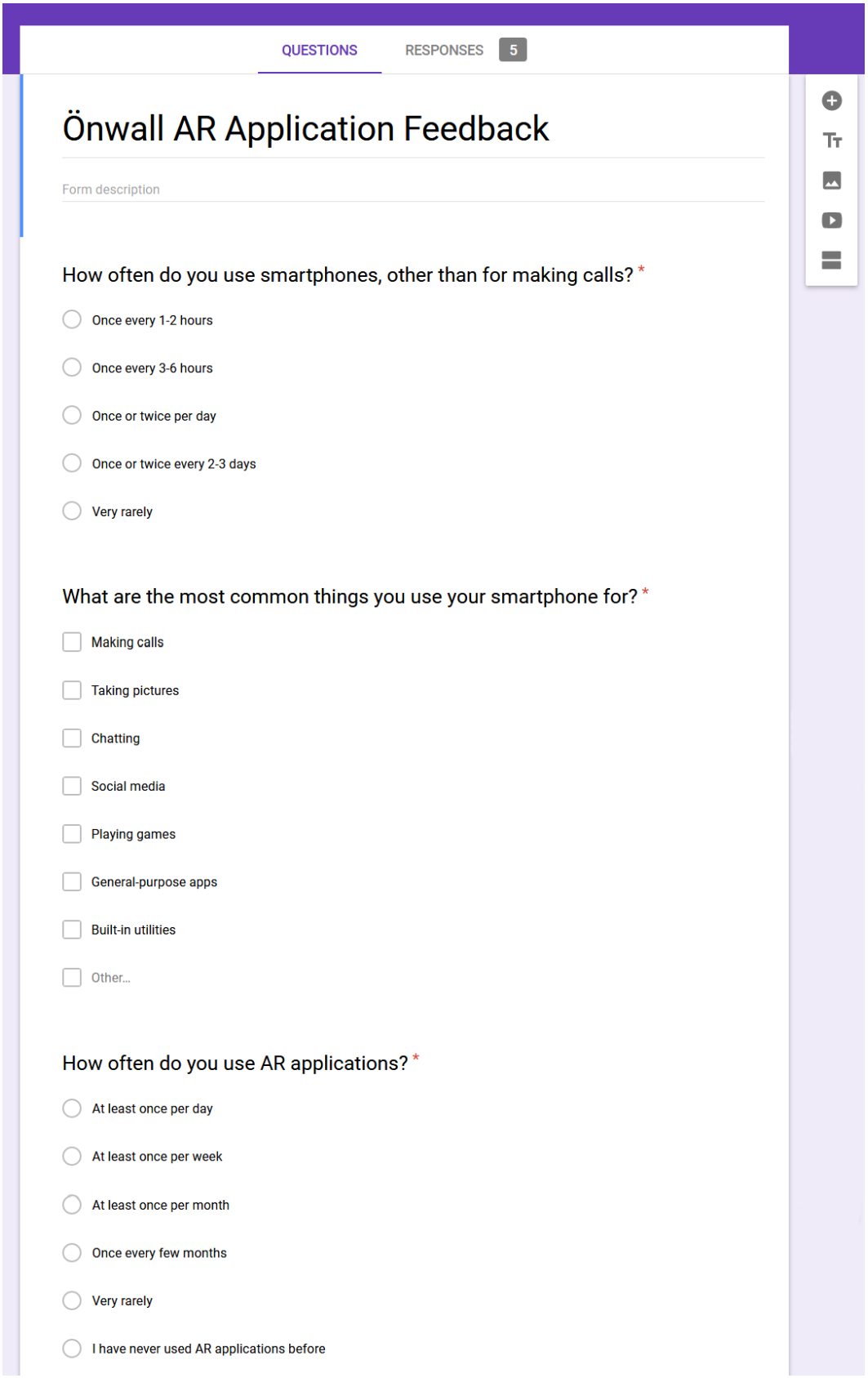

Illustration 32. Questionnaire, part 1

How difficult was it to use this application in general? *

	1	2	3	4	5	6	7	8	9	10	
Easy	○	○	○	○	○	○	○	○	○	○	Difficult

Why?

Long-answer text

How difficult was it to use the wall detection on this application? *

	1	2	3	4	5	6	7	8	9	10	
Easy	○	○	○	○	○	○	○	○	○	○	Difficult

Why?

Long-answer text

How did you like moving tiles around and changing the amount of tiles? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Why?

Long-answer text

How did you like changing the lighting in the scene? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Why?

Long-answer text

Illustration 33. Questionnaire, part 2

How did you like taking screenshots? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Why?

Long-answer text

How did you like the UI? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Why?

Long-answer text

If you were ever interested in purchasing ÖnWall's products, how likely would you try out this product? *

	1	2	3	4	5	6	7	8	9	10	
Not very likely	○	○	○	○	○	○	○	○	○	○	Very likely

Is there anything else you would like to be changed or added? *

Long-answer text

Rate your overall experience with the application. *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Illustration 34. Questionnaire, part 3

III. Glossary

Smart Device – Electronic device that is generally connected to other devices or networks through wireless protocols like Bluetooth, Wi-Fi, 4G, etc. Some examples are smartphones, tablets, smartwatches, and so on.

Software development kit (SDK) – Set of software development tools to give more functionality to an existing platform, such as ARCore providing premade AR related functions to Unity.

Game engine – A software-development environment that is designed to create video games. Typically includes a rendering engine for 2D or 3D graphics, a physics engine or collision detection, sound, scripting, animation, and so on.

Inertial measurement unit – An electronic device that uses a combination of accelerometers to measure acceleration and gyroscopes to measure its angular rate (rotation acceleration).

Transform – The position, rotation and scale of an object.

Pose – The computer vision term for a transform, to determine each object's position and orientation relative to some coordinate system.

Computer vision – A scientific field of computer science that deals with making computers understand digital images or videos / automate tasks that the human visual system can do.

3D Rendering – Generating an image from a model (e.g. a virtual 3D model)

Virtual camera – A camera that is used to display a 3D virtual world. In AR, the *real world* is used as a background and any virtually created objects are what the user can only see through the virtual camera.

Mesh – A collection of vertices, edges and faces that make up a 3D model.

Directional light source – A light source that defines a global direction where the light comes from. Usually used as sunlight in a 3D world when a ground is rendered.

IV. Additional Files and Repository

2 Additional files were provided in a separate file, an apk file and a video:

Onwall_ARCore.apk – This is the main program that can be installed on any Android device that has Android 7.0 at minimum. However, if the device does not support ARCore, it is still not usable. To use this program, copy it on the device you want to use it on. Under normal circumstances, going into the file manager and tapping on the apk file prompts the user to install the application. If this does not work, then the user might have to enable the developer mode on their device. Once the application is successfully installed and activated, it requests permission to use the camera, in order for the application to work, and storage, in order to move images from its internal files to the Pictures folder. It also asks permission to download ARCore, if the user does not have it already. Doing so takes them to the Google Play Store page, where they can download ARCore, if their device supports it. Once all of this is done, the application can be started normally at any point by simply clicking on the application icon.

Onwall_Demo.mp4 – This video is a demonstration on how the Önwall AR Tool works, in case the apk file cannot be installed. It shows everything that the application can do, apart from viewing the contact information.

An additional repository link to the project is provided here:

Repository link – https://gitlab.com/K-WS/onwall-arcore

Due to Önwall wanting the code to be private, for business reasons, separate permission will need to be asked in order to access it.

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I,
Karl – Walter Sillaots,
(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Önwall AR Tool,
(*title of thesis*)

supervised by Raimond-Hendrik Tunnel.
(*supervisor's name*)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl – Walter Sillaots
10/05/2019