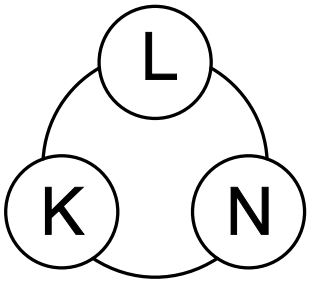


Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Wolfgang Kellerer

Bachelor's Thesis

An augmented reality-based mobile application for visualizing robot models and robot states

Author:	Knestel, Philipp Georg
Address:	Horscheltstraße 2
	80796 Munich
	Germany
Matriculation Number:	03688135
Supervisor:	Prof. Dr.-Ing. Wolfgang Kellerer
	Prof. Dr.-Ing. Eckehard Steinbach
	Dr.-Ing. Daniel Wahrmann
	M.Sc. Hasan Furkan Kaynar
Begin:	15. June 2020
End:	15. November 2020

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 10.11.2020	
Place, Date	Signature

München, 10.11.2020	
Place, Date	Signature

Kurzfassung

Das Ziel dieser Arbeit ist es, folgende Frage zu beantworten: Verbessert eine Visualisierung von Roboterzuständen mittels Augmented-Reality die Kommunikation zwischen Mensch und Maschine?

Für die Erörterung der wichtigsten Funktionen einer solchen Augmented-Reality Anwendung ist eine Umfrage unter den Mitarbeitern von Franka Emika durchgeführt worden. Anschließend wurde in Unity eine Augmented Reality Anwendung für iOS entwickelt. Anhand der Umfrage wurde eine Demo des Franka Emika Prototypen 'Garmi' sowie eine Demo des ebenfalls von Franka Emika stammenden 'Panda' Arms implementiert. Hierfür erkennt die Anwendung auf dem Smartphone die Oberflächen in der Umgebung. Als Mixed-Reality Ansatz wurde die dritte Funktion, 'Panda Inside', entwickelt. Hier wird die Anwendung mit live Daten des 'Panda' Arms versorgt und bildet diese Daten auf dem realen 'Panda' Arm ab um eben diese Daten dem Nutzer näherzubringen. Dabei wird ein QR-Code vor dem 'Panda' Arm platziert, erkannt und als Anker für das digitale Modell verwendet. Abschließend wurde eine Nutzerstudie ausgerichtet, welche die Anwendung und ihre Auswirkungen auf die Arbeit mit dem Roboter sowie die Aussichten für die Zukunft erfragt. Hierfür konnten die Teilnehmer die Augmented Reality Anwendung am echten Roboter, dem 'Panda' Arm, testen. Die Antworten in dieser Nutzerstudie fielen überaus positiv aus, was die Annahme bestätigt, dass Augmented-Reality Anwendungen überaus nützlich sind um Zustände von Maschinen zu visualisieren.

Abstract

This thesis aims to answer the following question: Does the visualization of robot states using augmented reality improve communication between man and machine?

To discuss the most important functions of such an Augmented Reality application, a survey was conducted among the employees of Franka Emika. Afterward, an Augmented Reality application for iOS was developed in Unity. Based on the survey, a demo of the Franka Emika prototype 'Garmi' and a demo of the 'Panda' Arm, also from Franka Emika, was implemented. For this purpose, the application on the smartphone recognizes the surfaces in the environment. The third function, 'Panda Inside', was developed as a mixed-reality approach. Here, the application is supplied with live data from the 'Panda' arm and displays this data on the real 'Panda' arm to bring this data closer to the user. A QR-code is placed in front of the 'Panda' arm, recognized, and used as an anchor for the digital model.
Finally, a user study was carried out to find out about the application and its effects on working with the robot and the prospects for the future. The participants were able to test the augmented reality application on the real robot, the 'Panda' arm. The answers in this user study were extremely positive, which confirms the assumption that Augmented Reality applications are beneficial for visualizing machines' states.

Preface

I had already worked with Franka Emika for half a year when I inquired which subject areas would be suitable for a Bachelor's thesis. Daniel Wahrmann suggested that I extend my education in the field of Augmented Reality in order to find a suitable topic for my thesis. Afterward, I started looking for a fitting development environment and tried out different approaches and frameworks. This work was exciting because I went through different technical approaches, and after some setbacks, I found the best solution for me. Then, I started to search for a suitable application for the 'Panda' Arm. After some adjustments, I specified the topic and the goal of the thesis. I was able to register the work with the help of the Chair of Media Technology and the Chair of Communication Networks.

Acknowledgement

First of all, I would like to thank my supervisor at TUM, Hasan Furkan Kaynar, for his consistently helpful and active support in this thesis. I was able to ask any questions and discuss the current events every week, which helped me a lot and provided a red thread during the thesis's development.

I would also like to thank Daniel Wahrmann Lockhart, who made sure to write my thesis at Franka Emika. Daniel helped me a lot, in the beginning, to get an overview of the topic Augmented Reality. Afterward, he was accommodating with all his experiences regarding the exact topic and the execution of the work.

A special thanks to Thore Goll, who was always able to help me with technical questions and spared no effort to solve a problem together with me.

I would also like to thank the participants of my surveys with Franka Emika. This work could not have been accomplished without them. Their willingness to provide information and their interesting contributions and answers to my questions contributed significantly to the survey's success.

Special thanks also go to my family, who accompanied and supported me throughout the entire thesis. Above all, I want to thank Meltem for your patience during the whole thesis and support in form questions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interpersonal communication often proves to be difficult. The phrase "Say what you feel - people cannot see inside yourself" comes up again and again. However, in contrast to interpersonal communication, man and machine rarely resemble each other in the way they feel and build up and perceive the environment. Thus it is all the more difficult for humans to put themselves in the position of a machine. Therefore, the key to good human-machine communication is to clearly and intuitively transmit as much data as possible [13]. So that this physical barrier can be bypassed, and a smooth workflow can be enabled. With the development of HoloLens 2, Microsoft has taken a big step towards human-machine communication [4]. With the dynamic 365 program, the understanding of a machine should be significantly increased. The HoloLens 2 spectacles are designed to guide the user step by step through an instruction manual, digitally demonstrating the next work steps and displaying data directly on the machine.
But this is only the beginning of this technology. Many daily work problems with machines and robots could be solved if machines could communicate better and thus better understand people [8]. Not only can errors be avoided or productivity in work steps be increased, but the quality of the work can also be improved. Safety aspects also play a role. Where do autonomous vehicles move to, or is a person even recognized as such in order to respond to them. Augmented reality or mixed reality is an interesting option of communication because information can be displayed directly in the person's field of view without being bound to little dynamic displays. Displays or representations could be projected directly onto a machine to describe the state of the machine visually. Using the example of the 'Panda' arm of Franka Emika, an augmented reality application will be built and tested. On the one hand, it shall be found out if this kind of application is already technically realizable. On the other hand, it shall be found out if this kind of human-machine communication can be useful for a robot. The goal is to implement an Augmented Reality application for a mobile device, interpret live data of the robot, and present it in a meaningful way. A survey in Franka Emika will show which functions are offered in the application. Franka Emika manufactures the 'Panda' robot arm. This com-

pany's employees have some expertise regarding the sources of errors and information not transmitted by the robot. Subsequently, an AR application will be implemented based on the more precisely defined goals. This is to be done platform-independent since, on the one hand, several operating systems are possible. On the other hand, a further development based on this thesis is conceivable. A user study will be conducted to determine whether the communication is facilitated and whether this type of use of AR makes sense. To test all functions extensively, the participants of this study will be provided with 'Panda' arms to make statements.

Chapter 2

Background

Augmented Reality has been removed from Gartner's Hype Cycle 2019 [7]. However, this step forward came much faster than expected. Apple, Google, and Co. developed the first basics quickly to make Augmented Reality usable for developers starting in 2018. Apple released iOS 11 in the spring of 2018 and brought the API ARkit with significant improvements to the iPhones and iPads [21]. Nevertheless, what is AR, and how does it work?

In general: Digital images or models are placed in the real world via a screen, beamer, or, as most common, via a mobile device like an iPhone using the camera image. The user can thus insert digital elements into his environment. In order to prevent these models from being placed arbitrarily in the camera image, the AR application has to recognize points in space and place the models there. To do this, there are two hidden possibilities.

One is marker-based AR [20]. For this, the camera image is grayed out to increase the processing speed. The AR application is implemented with pre-defined images it looks for. The best way to do this is to use QR codes because they show unique shapes with high contrast. The application tries to detect similarities between the live image and the stored images [20]. If an image is detected, it is used as the base of the model. So this image serves as an anchor for the AR application. Markerless AR is more complicated [20]. There are several methods to place an anchor in the environment without placed images. You can use geographic data like maps or GPS to get an idea of the environment or the AR app tries to recognize areas and objects by colors [20]. A larger surface with the same color is a good target. This makes it possible to place the digital models on a surface and make them look natural.

The robot used in this thesis is the 'Panda 'Arm by Franka Emika. For this thesis, it is necessary to know that the 'Panda' arm has 7 axes, i.e., seven degrees of freedom [6]. These axes cannot rotate 360 degrees but have different joint limits [6]. For example,

the first axis can rotate from -166/166 degrees, but the second axis can only rotate from -101/101 degrees [6]. The 'Panda' arm has almost 100 sensors because the forces acting on the arm are also measured [6]. The 'Panda' arm's position and measurement data are sent regularly to Desk in the form of a matrix. Desk is the user interface of the 'Panda' arm [5]. This is where this data is received and processed. At the same time, Desk has apps to give the 'Panda' arm various tasks. In this thesis, the matrix sent by the 'Panda' arm is also used. The position data, as well as the measurement data, are essential for the live digital image of the 'Panda' arm and the forces acting on it.

The interaction of real-world elements with digital elements is called mixed reality and was first described by Paul Milgram and Fumio Kishino [12]. Nevertheless, today, especially, Microsoft is working on this technology in the course of the HoloLense 2 development [2]. Basically, the term describes a connection between the human being, the computer, and the environment. The computer, as well as the human, recognizes the environment and usually receives different data. The computer should now communicate the data it recognizes visually and descriptively so that the human has more information at his disposal [2]. This thesis tries to apply this kind of technology to the 'Panda' arm.

Unity does the implementation. This is a development environment designed to be as cross-platform as possible [22]. This development environment allows the developer to export the same program for different operating systems and hardware relatively effortlessly [22]. At the same time, Unity offers some advantages for this thesis. The integration of AR-Foundation, an API for augmented reality, is far advanced and allows us to use the features of AR-Kit and AR-Core [19]. As described above, both typical techniques of AR technology are used.
A typical Unity program is divided into so-called scenes. A change from the main menu to a menu item would be a scene change. Within a scene, the developer brings related functions underneath, which are played until the scene is finished. Within a scene, so-called GameObjects are inserted. A GameObject can be anything from a UI element to a digital light source [25]. These game objects can also be imagined as classes. Information and functions are implemented within these GameObjects as components [26]. A component can be, for example, a shader, a script, or an animation. Everything that influences the game object is stored as a component [26]. In Unity, the scene is always shot by a camera. This is very figurative. One or more cameras record the implemented scene, and the user sees the action through one of these cameras [24].
Within Unity, the term raycast is important. This technique is used to assign touch input to a UI element [23]. The user taps on the screen, raycasts are shot in the direction of the scene. As soon as a raycast hits a surface, it is checked if this object is relevant for the user input and if the raycast should fly further through the scene or if the surface stops it Raycast. This technique is beneficial if several UI elements are on top of each other, and a specific one should be used.

Chapter 3

Pre-Survey

In order to discuss which functions are particularly useful for AR applications, a study with a questionnaire was set up at the beginning of the thesis. The study was held within the development department of Franka Emika and was, first of all, to check if a benefit is seen in an Augmented Reality application. There were 12 employees involved, all of whom routinely work with the 'Panda' and know the daily operation problems well.

It was checked which functions are considered to be especially important. Specifically, they were asked which problems often occur when working with the 'Panda'. Problems with the joint limits and problems with the payload were queried. As you can see in Figure 3.1, it turned out that errors due to joint limits were a medium to the large problem for all those questioned.

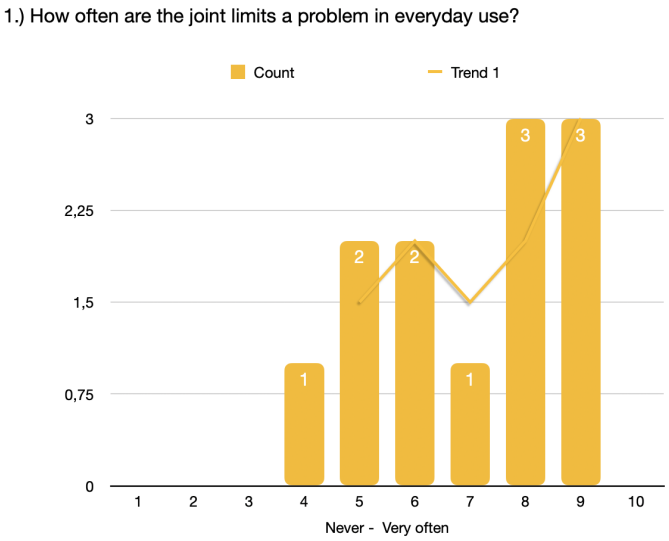


Figure 3.1: Pre-Survey: Question 1; Source: Own figure

In the case of errors caused by the payload, the camps are divided into the group with no problems, and the larger group that regularly receives errors here (Figure 3.2).

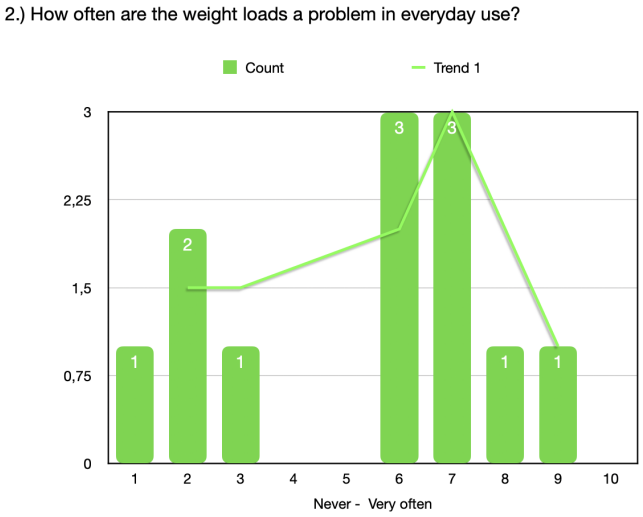


Figure 3.2: Pre-Survey: Question 2; Source: Own figure

Afterward, it was asked how useful a "Task-Preview" function and a recording function would be. The "Task-Preview" function would mean that the 'Panda' arm's digital model performs a movement sequence to show the user what the movement would look like and if it would work. The recording function should remember the individual joints' angles per frame and play them back if desired. As Figure 3.3 illustrates, almost all respondents found the "Task-Preview" function useful to very useful.

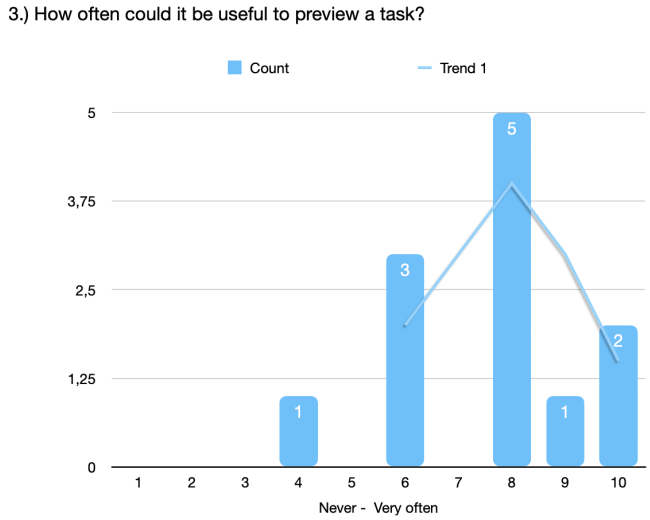


Figure 3.3: Pre-Survey: Question 3; Source: Own figure

The recording function, on the other hand, was rated rather neutrally. Figure 3.4 depicts that.

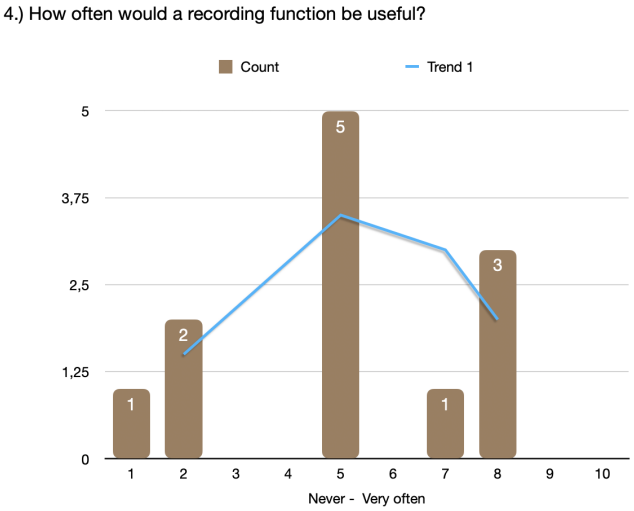


Figure 3.4: Pre-Survey: Question 4; Source: Own figure

These four functions could be ordered by preference in the next question. This resulted in the following Table 3.1 from important to worthless:

Functions	**Preference**
Joint limit ads	1.
Task preview	2.
Weight load ads	3.
Recording function	4.

Table 3.1: Preference of functions

The overall question was also asked for how great the benefit of an AR application is evaluated. Here in Figure 3.5, 9 out of 12 respondents said they saw a benefit or a great benefit in such an application.

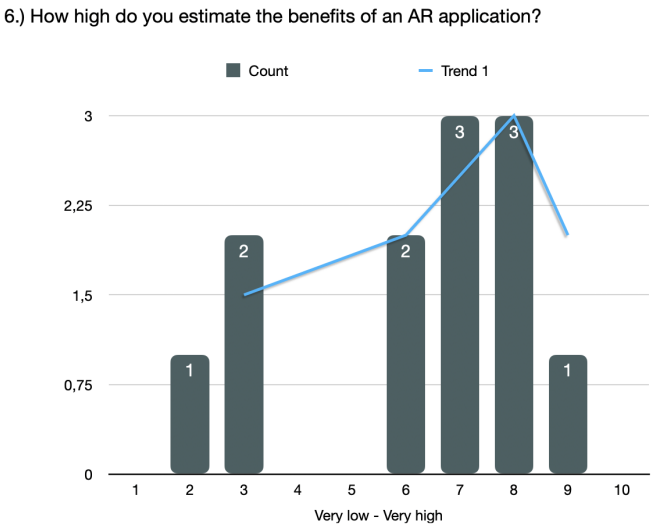


Figure 3.5: Pre-Survey: Question 5; Source: Own figure

Finally, the participants of the study could give personal feedback. In doing so, constructive suggestions were given. The most asked for was a 'Panda' arm model that can be placed in the world and manipulated afterward. Furthermore, all effects on the 'Panda' arm should be visible in the application. Not only the weight to be carried but all forces acting on the 'Panda'. A 'Garmi' demonstration was also suggested.

This study has not only confirmed the broad benefits of an AR application. However, it has also clearly pointed out that the 'Panda' arm's daily users may need certain functions to improve the work with a collaborative robot.

Chapter 4

Implementation

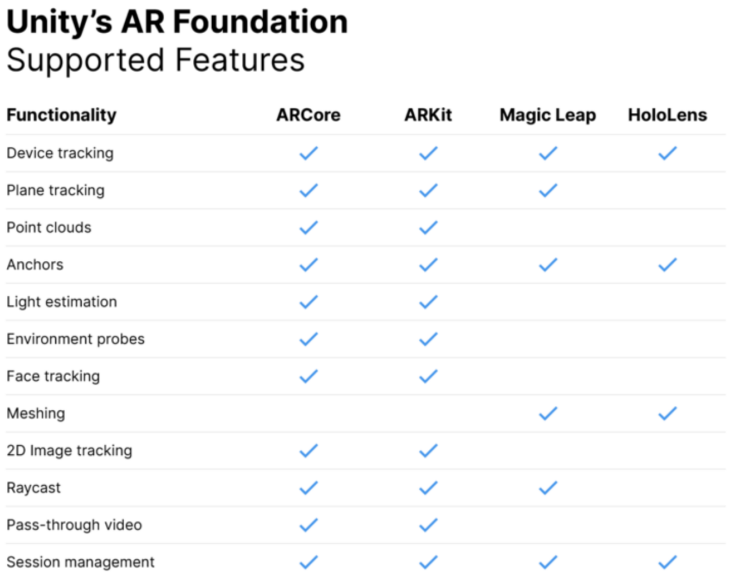
4.1 Development environment

First of all, the question of which development environment is the most suitable for this project arose. The requirements of the study are clear. The application has to recognize surfaces in order to place a digital model in the world. It must also be able to recognize an object or an image to place a model at this specific location. To provide the application not only on one system, but it also makes sense to implement it cross-platform. Support for iOS and Android is desirable as these are currently the most common operating systems for mobile devices [9]. Desired is this feature as it is not yet certain how the hardware for AR will develop in the future. Therefore it makes sense to keep as many options as possible open for further development. Since Apple and Google each provide their own APIs for AR, a comparison at the beginning is useful. It can be said that the differences are trivial for this application. Both APIs provide motion tracking, plane detection, light estimation, 2D, and 3D object tracking, anchors, and they both will support smart glasses once the hardware is invented [17]. There were no significant advantages or disadvantages of any particular API. So it is not necessary to implement the application for only one operating system.

For this reason, a platform-independent implementation seems to be a good option. It is obvious to use an engine so that one does not have to reinvent the wheel. The current leading open source engines are Unity and the Unreal Engine [14]. Here, however, it becomes apparent that the two engines specialize very differently. The Unreal Engine tries better to implement photorealism and graphic effects such as raytracing and clearly focuses on the optical in digital worlds and surfaces [29]. Although there is broad support for AR / VR APIs, the focus is not on building cross-platform applications [14]. However, this is the focus of Engine Unity [18]. The easy start with extensive tutorials and an active community and documentation [28] is convincing. Nevertheless, a big argument for Unity in the area of AR is the framework AR-Foundation. This is the framework of Unity,

which was created especially for platform-independent implementation in AR and VR. It combines the APIs of Google, Apple, HoloLense, and MagicLeap [19].

As you can see in Figure 4.1, AR foundation supports all the necessary functions required for this thesis. These are plane tracking and device or 2D image tracking. Also useful and available in AR foundation are the anchors that are set automatically. The application identifies specific and unique points in the real world that are easily recognizable. These are stored, and these anchors are used to increase the digital models' stability in space. The positioning of these models in space is usually done concerning known surfaces and these anchors. The more anchors are recognized, the more reference points the application has to find itself in space and calculate the models' positioning more precisely and stably [3]. AR Foundation is structured only to implement the AR application once using the framework AR Foundation and then publish it for all leading platforms like iOS, Android, and HoloLense. All these points spoke for using Unity with AR Foundation for this thesis.



Unity's AR Foundation
Supported Features

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

Figure 4.1: Development environment: AR-Foundation features; Source: https://unity.com/unity/features/arfoundation

4.2 Structure

4.2.1 Structure of the Unity editor

(Figure 4.2, A) The toolbar provides access to the main working functions. It contains the tools for manipulating the scene view and GameObjects. You can also start and pause the scenes and go through them step by step to test them [28].

(Figure 4.2, B) The hierarchy window is a hierarchically arranged list of all objects in the selected scene. You can see all direct dependencies between the objects [28].

(Figure 4.2, C) The game view simulates the rendered view from the camera, which eventually becomes the user's view. To do this, you have to press the play button in the toolbar [28].

(Figure 4.2, D) Here the built world or scene is shown in 2D or 3D. You can manoeuvre and manipulate the scene to make adjustments [28].

(Figure 4.2, E) The inspector window displays all information on the selected GameObject. Here you can see basic information like position, rotation, and scaling and all scripts, shaders, and effects related to the GameObject [28].

(Figure 4.2, F) The project window shows the entire available library of the project. It contains all models, shaders, scripts, and other file formats included in the project. Imported assets and models will end up in this structure [28].

(Figure 4.2, H) As soon as the scene is started with the toolbar's play button, live errors, warnings, or console output is displayed here [28].

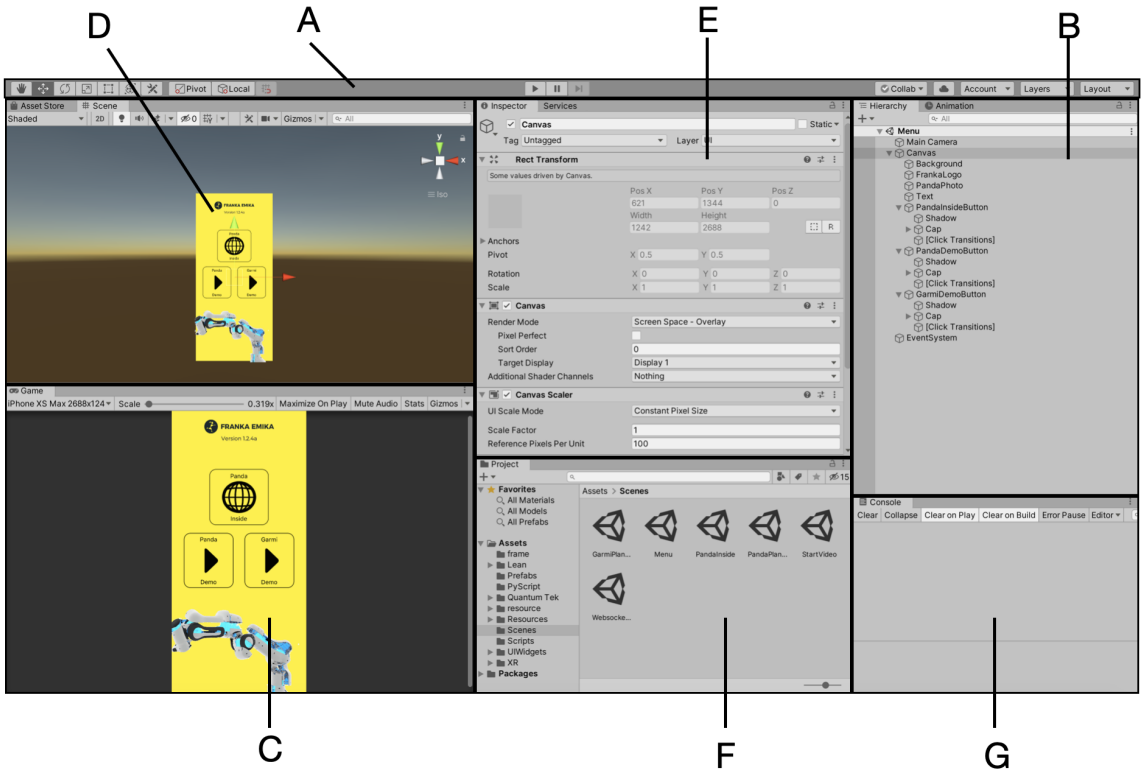


Figure 4.2: Structure: Unity Editor; Source: Own Figure

4.2.2 Presets in Unity

It needs some preparation to implement and export the Unity project as an xCode file with AR Foundation support. First of all, you can download the program Unity Hub for free at Unity.com. Once here, you can download a version of the Unity Engine under the tab "Installs". It is important to install the right modules for iOS, Android, or other platforms (Figure 4.3).

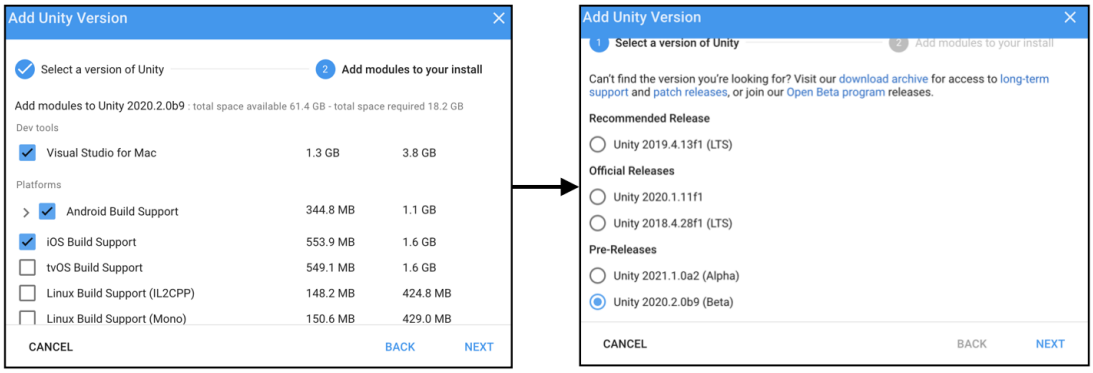


Figure 4.3: Structure: Unity Installation; Source: Own Figure

Afterward, you can create a new project in the Unity Hub under the tab "Projects" with the "Add" button. To do this, select the desired installed Unity version and create a 3D project for an AR application. Unity opens the project, and because the iOS/Android modules are installed, it is now possible to make some settings that allow exporting an xCode project for iOS. To do this, you have to select the iOS tab under File/Build Settings and switch to this platform. Afterward, the settings "Run in xCode" should be set to "latest version" and the setting "Run in xCode as" to "Release" (Figure 4.4).

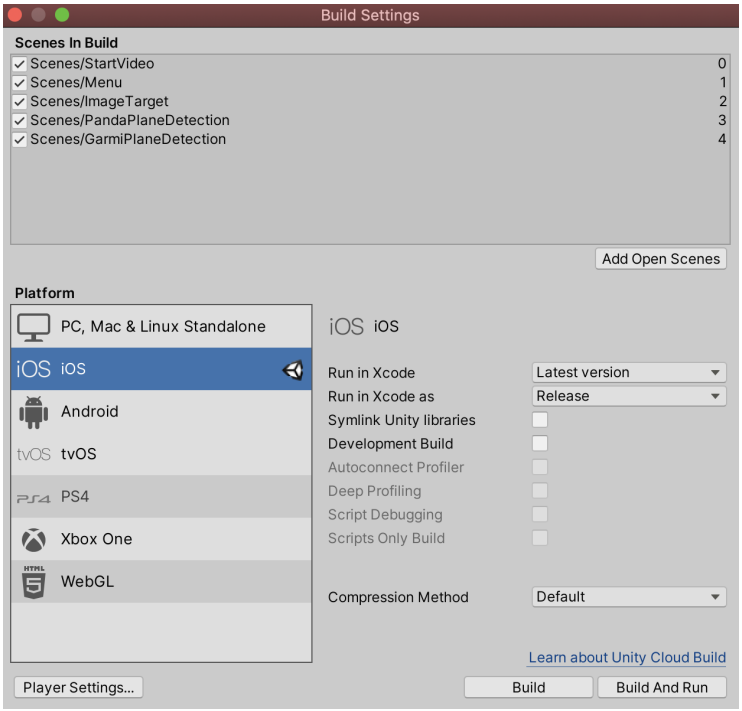


Figure 4.4: Structure: Unity build settings; Source: Own Figure

Now use the "Player Settings" button to open a window for detailed settings (Figure 4.5). Here it is necessary to create a company name and a product name. This is needed to create a signature afterward and in the future. Signatures should identify the developer. Further down in this window, you can choose between desktop, iOS, and Android. Since this thesis is developed for iOS, only these settings are important. Under the tab "other Settings" in the iOS settings, you have to enter the bundle identifier in the identification fields. This consists of "com.[Company Name].[Product Name]" (Figure 4.5). Further down in the settings, you must select the target devices. Since it is not important for the thesis on which mobile device from Apple, the application will run at the end, "iPad + iPhone" is selected here. The desired SDK is that of the end device, and the minimum requirements must be set to 11.0 due to the required AR capabilities of the end device [1]. As shown in Figure 4.5, the architecture must be set to "ARM64" as this is the supported capability of the newer iPhones and iPads [16]. These are all necessary settings to export an xCode project.

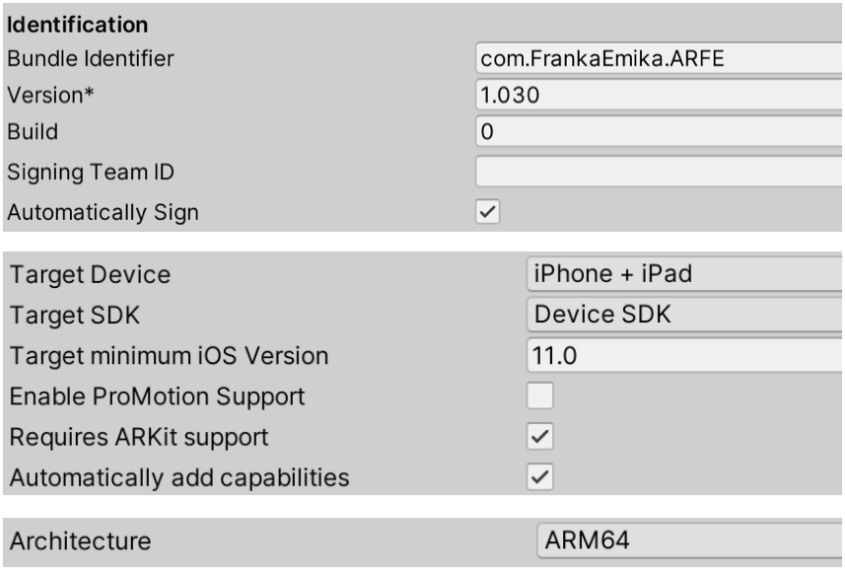


Figure 4.5: Structure: Unity player settings; Source: Own Figure

Since AR-Foundation is required for this thesis, this package must be added to the project. To do so, open Window/Package Manager and see all the already integrated packages. To use AR-Foundation for iOS, you have to install the following packages: AR-Foundation, AR Kit XR Plugin, XR plugin Management (Figure 4.6). The AR-Foundation and AR Kit XR plugin must be running on the same version because they are compatible. If this is not the case, any test of the application will result in an error.

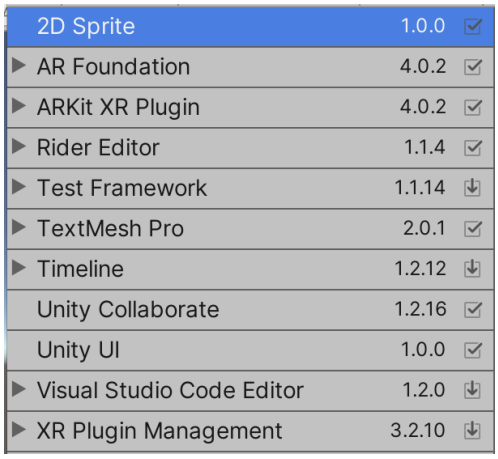


Figure 4.6: Structure: Unity packages; Source: Own Figure

4.2.3 Structure of the FEAR application

The AR application implemented in this thesis is called "Fear" in the following. The name is composed of the initial letters of Franka Emika and Augmented Reality.

The application is divided into six scenes. The scene "WegsacketTesting" is only used to test the connection between the application and the panda arm and does not appear in the final FEAR app. So there are only five scenes in the Fear app. To arrange the scenes or define the first scene, you have to add the Build Settings scenes in the correct order. The 0. scene is the scene the app starts with. The Fear app starts with the scene "StartVideo" which plays an intro video of Franka Emika (Figure 4.7). After 5 seconds, the app switches to the scene "Menu". Here the center of the application is implemented. You can choose between three scenes (Figure 4.7). From all three scenes, you can return to the main menu.

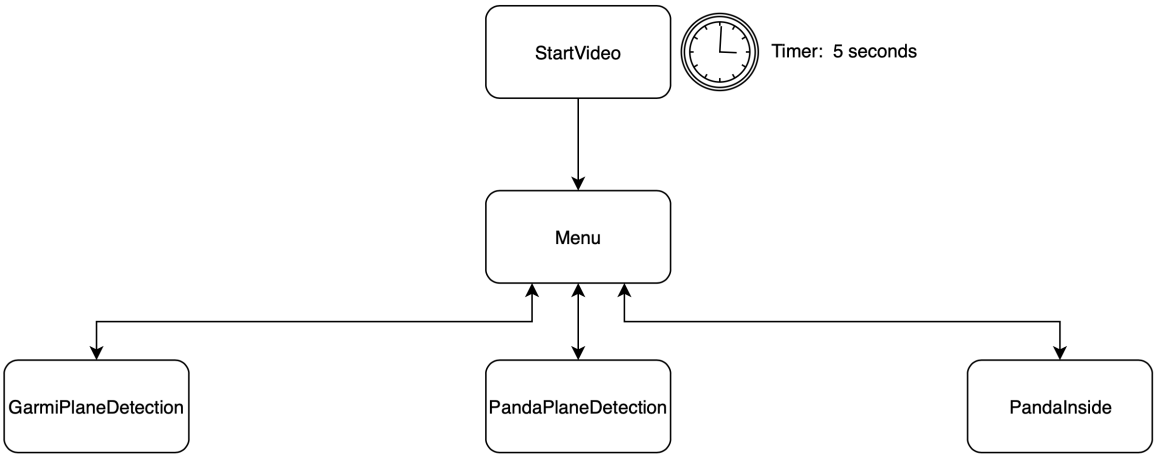


Figure 4.7: Structure: Structure of the Application ; Source: Own Figure

4.2.4 Which main functions are located in which scene?

These three scenes represent the three main functions of the application. All functions discussed in the study are located here. The "PandaPlaneDetection" scene allows you to detect vertical and horizontal surfaces and place, move, and manipulate a panda arm model on these same surfaces. The "GarmiPlaneDetection" scene detects vertical and horizontal surfaces in the same way as "PandaPlaneDetection". The only difference is that you can place a model of the prototype "Garmi" by Franka Emika in the world. Here you can also move the model but not manipulate it. The model of "Garmi" plays a loop animation. In the last scene, "PandaInside" you can scan a QR Code, place a transparent model of a panda arm over this QR Code and connect this model with a real 'Panda' arm. The app

then receives live data of the 'Panda' arm and visually displays joint limits, payload, and external influences on the 'Panda' arm.

All scenes are equipped with a canvas. This means that a user interface is implemented that corresponds to the display size of the terminal device. In the case of this thesis, this is the iPhone XS Max in portrait format with 1242x2688 pixels. It is still possible to play the application on other devices with other resolutions, but then the user interface is automatically distorted because it is adapted to the device.

4.3 Start Screen

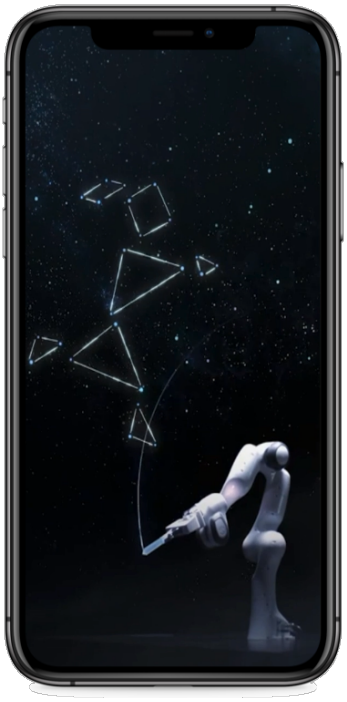


Figure 4.8: Start Scene: Screenshot; Source: Own Figure

This scene's primary function is to bridge the app's loading time during which usually only a white screen would be visible. A timer is integrated, which automatically switches to the scene "Menu" after 5 seconds. As you can see in Figure 4.9, the scene consists mainly of the canvas and the video player.

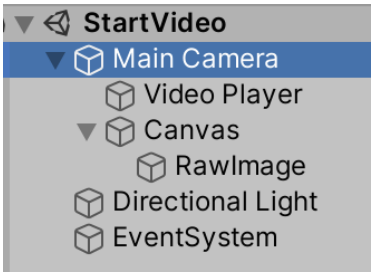


Figure 4.9: Start Scene: Hierarchy; Source: Own Figure

Video Player is a function provided by Unity to integrate videos into a scene quickly. Here you can insert a video in mp4 format and make settings. In the Fear app case, you should wait with the video until the first frame has been loaded. Also, frames that could not be loaded should be omitted. Since it is only a bridging video, it is not bad if frames are lost (Figure 4.10) On the other hand, a further delay by single loading frames would be

unhelpful. A script starts the video automatically after one second and makes sure that it is loaded. Inside the canvas, there is only a raw image over which the video is placed. On the main camera, there is only a single C# script in this scene. This script ensures that the scene is automatically changed after the timer has expired.

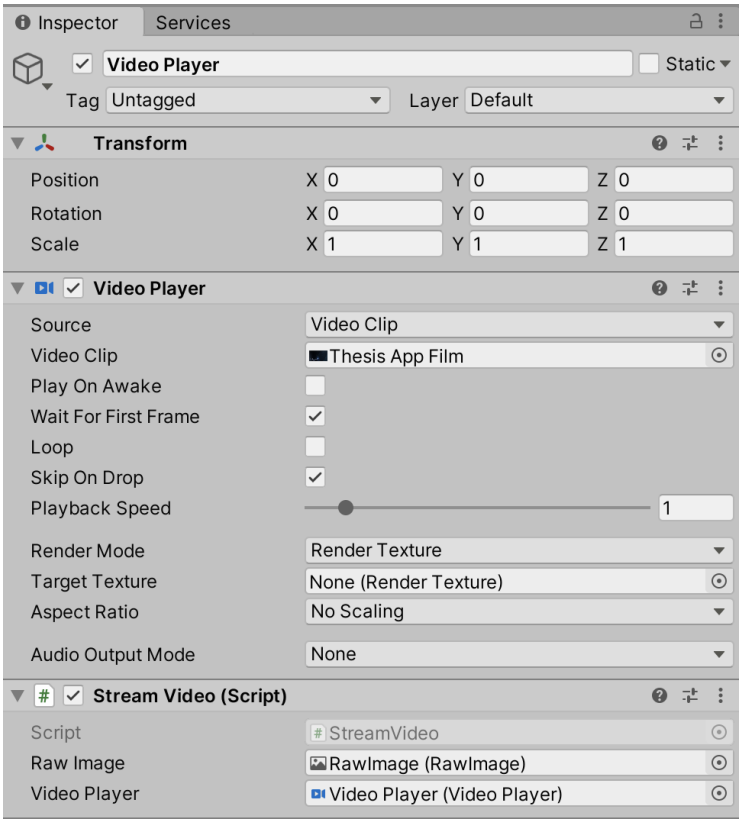


Figure 4.10: Start Scene: Inspector; Source: Own Figure

4.4 Menu

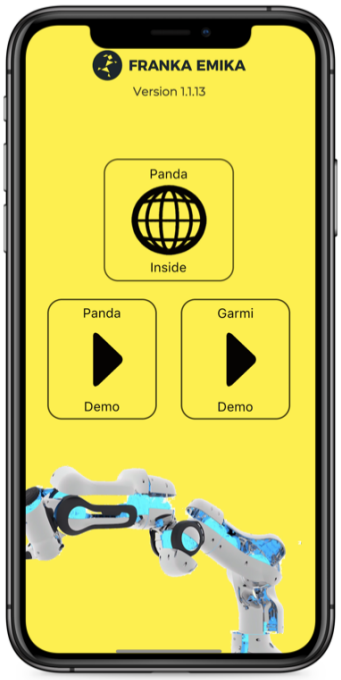


Figure 4.11: Menu: Screenshot main menu; Source: Own Figure

As shown in Figure 4.11 and Figure 4.12, the main components of the scene 'Menu' are:

- Main Camera
- Canvas
- Background
- Franka Emika Logo
- 'Panda' arm visualization
- 3x Lean Button

The main menu is the center of the FEAR App. From here, all three main functions, "Garmi Demo", "Panda Demo" and "Panda Inside" can be reached via built-in buttons. The design is clearly arranged. The background is golden yellow (RGB: 255;240;0;255) and inserted in the canvas as a normal image. The Franka Emika logo and the version number have been placed over the background image, and an illustration of the 'Panda' arm has been placed in the lower part of the menu. Franka Emika provided the logo and the illustration. The three buttons in the center of the menu are taken from the Lean GUI Package by Carlos Wilkes. This can be downloaded free of charge from the Unity internal asset store. The buttons in this GUI package are not fundamentally different from the UI

buttons already included in Unity. However, the Lean Buttons already have an animation implemented that indicates when the button is pressed.

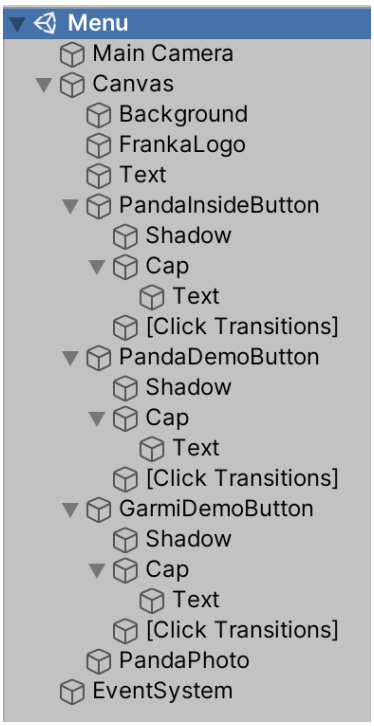


Figure 4.12: Menu: Hierarchy in unity; Source: Own Figure

Figure 4.13 illustrates the implemented options of the Lean Button. It can be seen that different states of the button can easily be equipped with animations. OnDown() and OnClick() represent two different states. OnDown() is good if you want to perform an action as long as the button is held down. OnClick(), however, performs a single action when the button is clicked. Because in the Fear app, the buttons are always used for one-time actions like a scene change, all C# scripts are included under the tab OnClick(). In the main menu scene, all three buttons are there to switch to the respective scene. All buttons in the Fear App use the Lean GUI Packages. All buttons in the Fear App have the same animations and differ only by the attached C# scripts. The Event System (shown in Figure 4.12) is automatically added by Unity if there is an UI element in a scene. This event system manages the users' input and assigns objects like buttons [27].

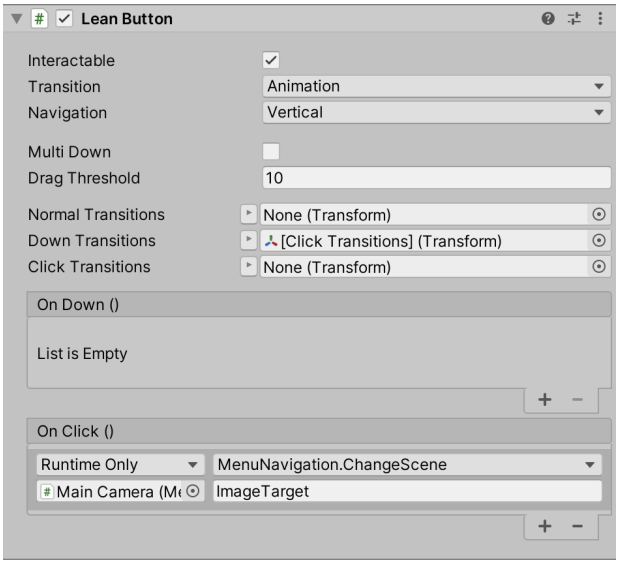


Figure 4.13: Menu: Inspector of a Lean Button; Source: Own Figure

4.5 Garmi Demo

Figure 4.14: Garmi Demo: Screenshot of the Garmi Demo; Source: Own Figure

The scene "Garmi Demo" basically has a function. You can enter the scene via the main menu and return to it via the Butten menu (see Figure 4.14). The application recognizes

surfaces and marks them as bright areas. Within these light planes, the user can now tap to place the 'Garmi' model in the world. This 'Garmi' model can be moved within the light planes and has an animation on the right arm to waves. The whole scene is shown in Figure 4.14.

The main components of this scene are:

- AR session
- AR Session Origin
- AR Camera
- 3x Directional Light
- Canvas
- Lean Button
- Plane
- 'Garmi' model

Unlike the main menu, the scene is not equipped with a Unity main camera. To use AR-Foundation and the package features, you need to add an AR-Session, AR-Session Origin, and an AR Camera to the scene (Figure 4.15). To ensure that the 'Garmi' model is well lit after placement, three light sources have also been added to the scene. The lighting conditions are not taken from the real world but have to be simulated in the Fear app. In the canvas, i.e., the user interface, only a panel, and a menu button is implemented to get back to the main menu. As the animation of the 'Garmi' arm is done automatically, no further UI element is necessary.

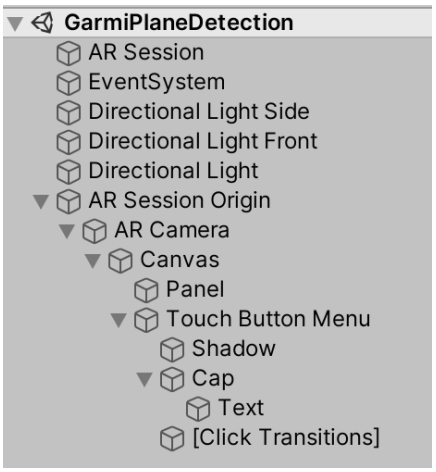


Figure 4.15: Garmi Demo: Hierarchie; Source: Own Figure

As the AR-Foundation package was integrated into the project, the components AR-Session, AR-Session Origin, and AR Camera are available. Also included are some scripts that use the Fear App. First of all, AR Session is implemented into the scene. As shown in Figure 4.16, there are two components included. AR Session (script) is a component provided by AR Foundation. Here, the settings made (Figure 4.16,) ensure that the models are always updated to match the entire prevention's framerate. It is also specified here that rotation and position are decisive for tracking surfaces, objects, or 2D images. Also given by AR-Foundation is the next component. The AR Input Manager (script) manages the lifetime of the XRInputSubsystem. This means that the position of the device is available. This is crucial for the correct display of the digital models in the room. The unit's position and movement are used to calculate the size and position of the model.

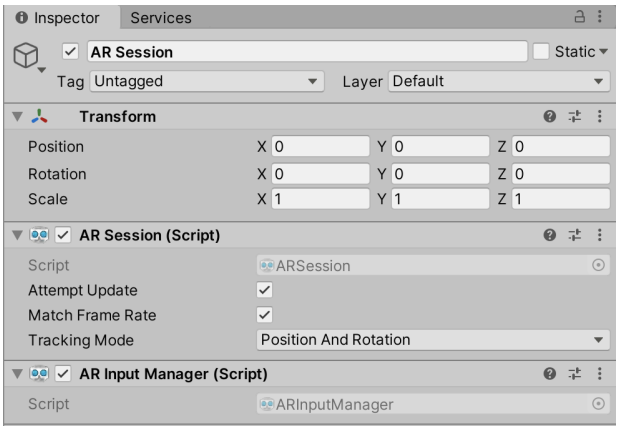


Figure 4.16: Garmi Demo: Inspector of the AR-Session; Source: Own Figure

While AR Session provides basic AR scene settings, AR Session Origin includes the applications and components specific to the scene (Figure 4.17). For an AR action to occur, the AR Session Origin (script) is used to select the AR camera that is the main camera for the scene. This camera is the center for all model calculations. The most important component for the "Garmi Demo" scene is the AR Plane Manager (script) provided by AR Foundation. This script ensures that horizontal and vertical surfaces are detected. You can also choose between horizontal or vertical. However, this is not relevant for this scene. Only the plane has to be transferred as prefab. You could give any prefab, but a plane with a slight transparency and edge marking is suitable for this application. As you can see in Figure 4.14, the borders and the surface are clearly visible, but the space behind it is still easy to see, so there is no limitation of the field of view. In this function's tests, it has been noticed that non-transparent marking of surfaces harms the user experience, as the user may feel that he cannot see everything and is therefore restricted. It is therefore advisable to cover as little of the camera image area as possible.

The AR Raycast Manager is also provided in the package AR Foundation. This manager is used to throw raycasts against tracked surfaces without giving them a physical presence. In this scene, this is bright panels that mark the vertical and horizontal surfaces. Since the

model is only to be placed on these panels, the panels must be accessible using raycasts. As these panels do not behave like normal user interfaces but are AR objects, a special script must be used. Also shown in Figure 4.17 is the script AR Tap to place Object. This function is one of the typical applications in an AR application. Basically, the script looks for the touch position on the device screen and then determines with the raycast manager if an AR plane was hit. If one of these surfaces is hit, the GameObject (in this case, the 'Garmi' model) is initialized with the correct position and rotation.
The 'Garmi' model consists of several composite prefabs: the head, the torso, and two 'Panda' arms, used in the rest of the project. Franka Emika provided the 'Garmi' 3D model.

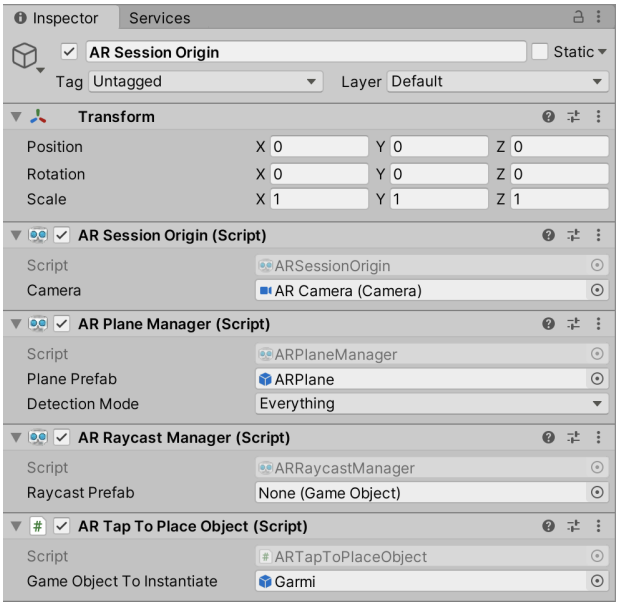


Figure 4.17: Garmi Demo: Inspector of the AR-Session Origin; Source: Own Figure

Finally, the AR camera is essential in this scene (Figure 4.18). The component AR Pose Driver is again provided by the package AR Foundation and can be added easily from the search function. The script tells you in which position the camera is in relation to the game object. Thus, by calculating distances, proportions can be calculated, and thus a realistic visualization is possible. The script "menu navigation" is stored as usual for the menu button. The AR Camera Manager is a script from the AR Foundation package and stores some main camera settings. Autofocus is user friendly for the Fear app, as otherwise, you would have to constantly click on the surfaces to be detected to get a sharp image. This could cause collisions with the Tap to place function, and the user experience could suffer. Besides, the camera can be used as an additional light source, and the orientation of the camera can be determined. The AR Background manager is only implemented in the FEAR app to deactivate a background. However, you could insert a virtual background here. Nevertheless, since this application is about placing models in the real world, this would be inappropriate.

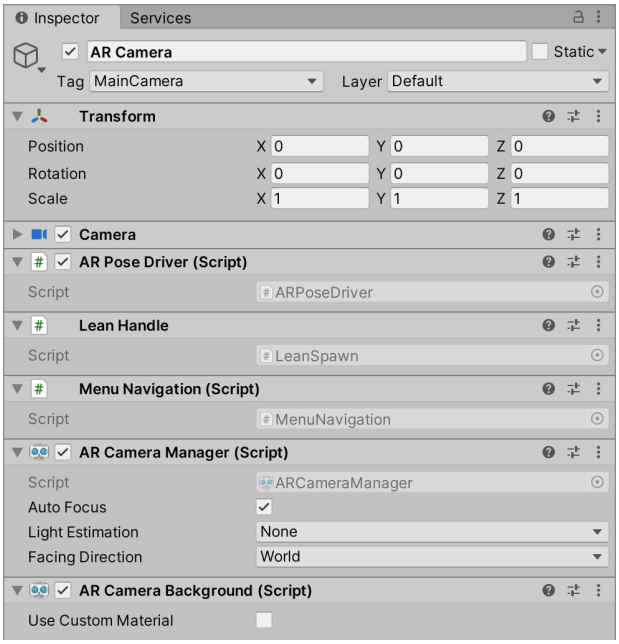


Figure 4.18: Garmi Demo: Inspector of the AR-Camera; Source: Own Figure

The 'Garmi' model consists of several composite prefabs: the head, the torso, and two panda arms, used in the rest of the project (Figure 4.19). Franka Emika provided the 'Garmi' 3D model.
Each joint of the 'Panda' arm is represented by a "link" in the Unity model of the 'Panda' arm. The rotation of each link on the z-axis causes the joint to move. On "Link4", the 4th joint of the 'Panda' arm, you can see an animation called "WaveLink4" in Figure 4.20. This animation consists of a rotation on the z-axis from 15 to -32 degrees and back. The whole thing is played as an endless loop so that the 'Garmi' model waves as soon as it is placed in the room. Only when the scene is finished, the animation will be finished.

Figure 4.19: Garmi Demo: 'Garmi' Model; Source: Own Figure

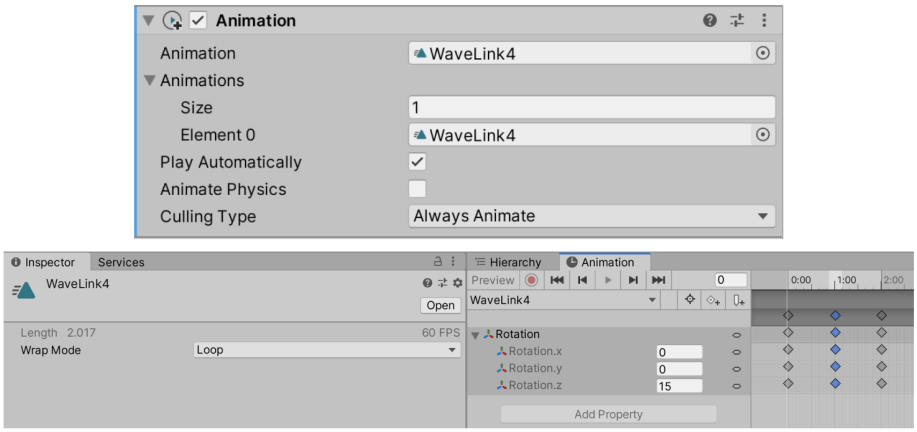


Figure 4.20: Garmi Demo: 'Garmi' wave effect; Source: Own Figure

4.6 Panda Demo

The "Panda Demo" scene is similar to the "Garmi Demo" scene in many respects, as, in both scenes, the surfaces in the real world are recognized and marked to place a model on them. Therefore this section will only go into the differences in detail. The big difference between the two scenes is that the 'Panda' arm model in this scene can be manipulated by the user by seven sliders. Every single joint can be moved and still shows the joint limits. The whole scene is shown in Figure 4.21.

Figure 4.21: Panda Demo: Screenshot of the 'Panda' Demo; Source: Own Figure

The main parts of the scene are:

- AR session
- AR Session Origin
- AR Camera
- 3x Directional Light
- Canvas
- 2x Lean Buttons
- Plane
- 'Panda' Arm model
- 7x UI Slider

To make the manipulation by seven slider user-friendly and clear, they are placed in a hidden submenu. Therefore a second Lean Button is inserted in this scene at the bottom right of the screen. As soon as the user clicks on this button, a panel with the seven sliders opens, as shown in Figure 4.21. As you can see in Figure 4.22, these sliders and the corresponding panel are gray. This means that these GameObjects are deactivated and not visible for the user or other functions.

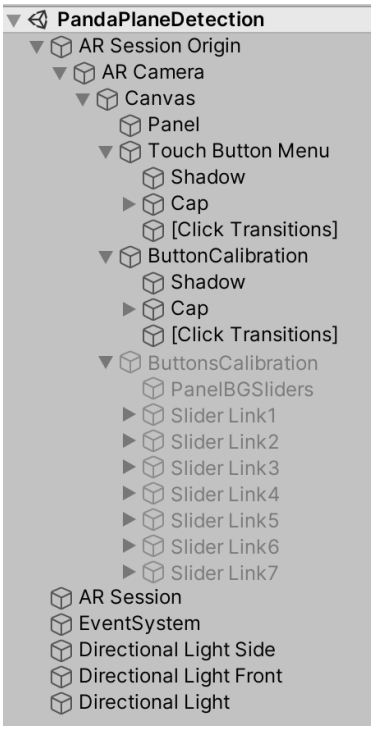


Figure 4.22: Panda Demo:Hierarchie of the 'Panda' Demo scene; Source: Own Figure

Because it sometimes caused the problem when tapping the calibration button so that the touch input was taken as input for the "Tap to place object" script, and the 'Panda' model was unintentionally moved. It is standard in this scene that the Lean Button and the seven sliders have an event trigger. This is shown in Figure 4.23. An event trigger is a tool provided by Unity to react to certain events with a script. As soon as the pointer, i.e., the button or the slider, is activated, the script "Tap to place Object" is deactivated. This prevents unwanted touch inputs. As soon as you release the button or slider, the "Tap to place object" script is activated again. Also shown in Figure 4.23 is the "Panel Opener" script. The only function of this script is to activate the "ButtonsCalibration" tab (Figure 4.22). Thus all children of this tab are also activated, and the panel with the seven sliders will open or close when the button is pressed again.

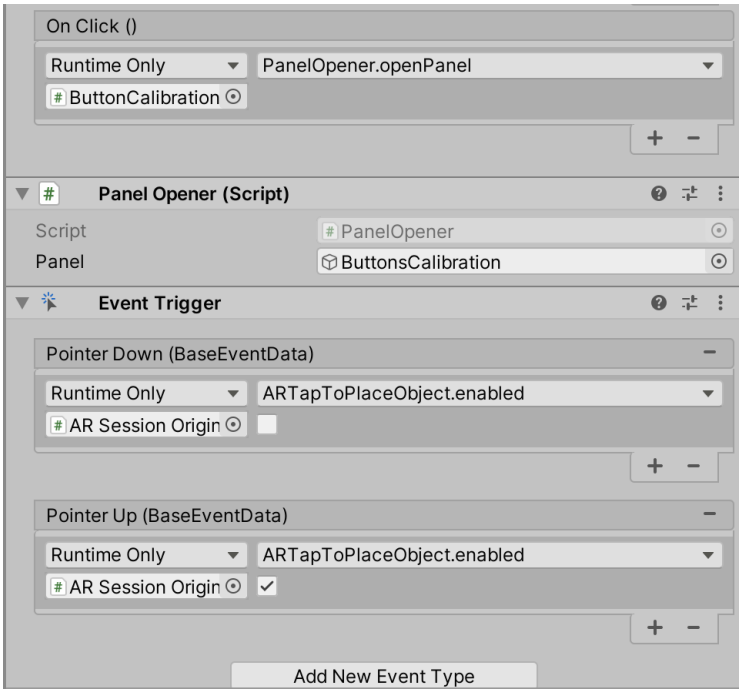


Figure 4.23: Panda Demo: Inspector of the calibration button; Source: Own Figure

Of course, in this scene, the digital model is the main difference. Here the 'Panda' arm is shown. Franka Emika also provided the 3D shapes for the prefabs. However, as you can see in figure 4.24, a few things have been added.

1. colored bubbles have been added to each joint with the same color as the sliders to make the control as easy as possible for the user.
2. the joint limits of each joint have also been visualized. This is shown as a round disk (Figure 4.24). The red indicator on the disc is the area within the joint limits. The joint cannot rotate within this area. The white pointer visualizes the current position of the joint.

For each joint, a slider has been implemented. Each slider has its own setting because the joint limit differs from joint to joint. By default, the Inspector of a slider looks like Figure 4.25. The maximum (Max Value) and minimum (Min Value) values that the slider can assume are important. Another script is required to transfer the values that the user has changed by moving the slider to the 'Panda' arm. This is the "NavRotationZ" script (Figure 4.25). The script "NavRotationZ" is responsible for this (Figure 4.25). The script needs the UI slider once to get a value that is generated by the slider. Second, the script needs a link tag to find the desired slider in the 'Panda' arm models' spanning tree and manipulate the local z-axis.

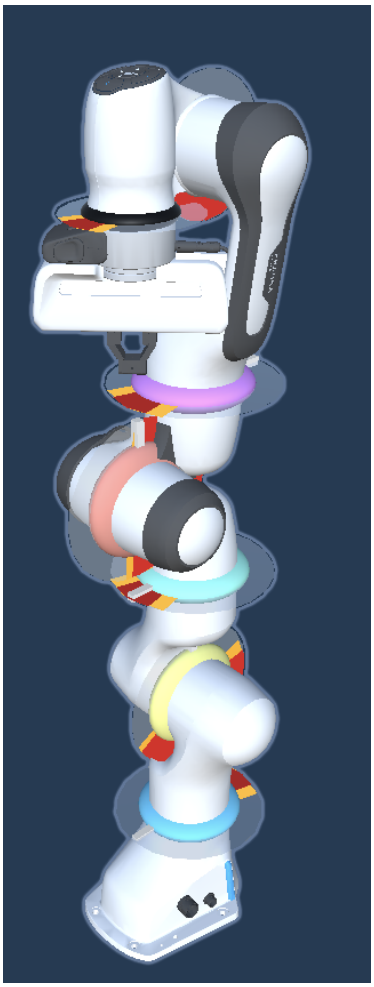


Figure 4.24: Panda Demo: Model of the Panda Arm; Source: Own Figure

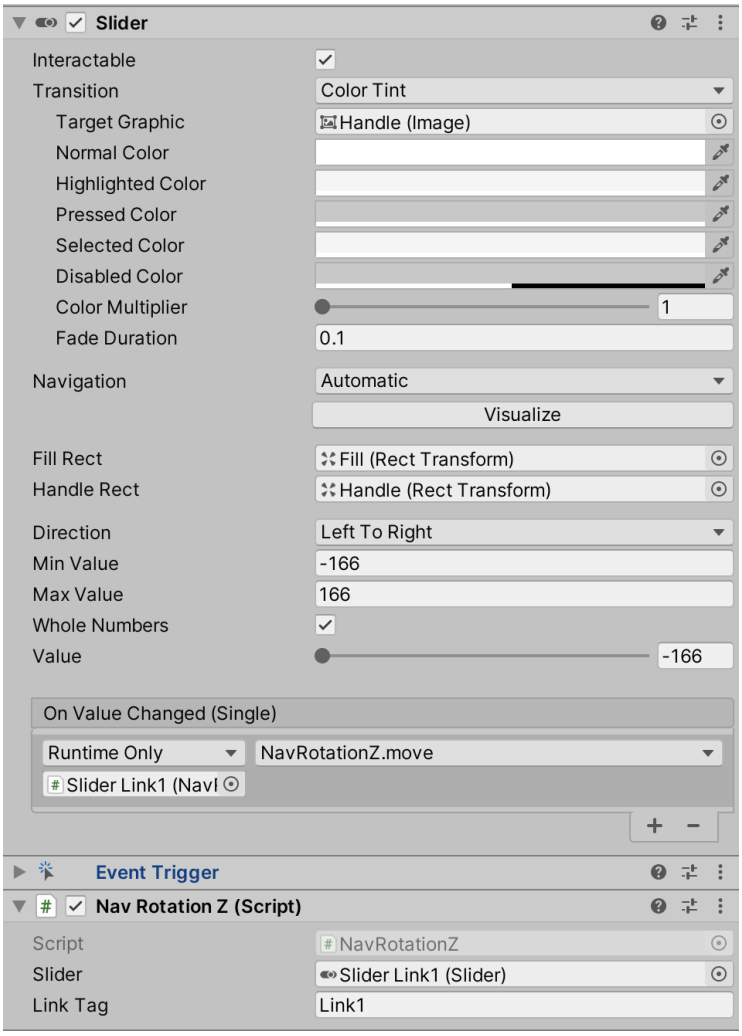


Figure 4.25: Panda Demo: Model of the Panda Arm; Source: Own Figure

4.7 Panda Inside

The scene panda inside is the most extensive part of the Fear App. Here a QR code is recognized, and a partially transparent model of the 'Panda' arm is projected onto its surface. Then the connection with a real 'Panda' arm is to be established, and the position is to be adjusted live so that the semi-transparent model is displayed over the real robot (Figure 4.26).

Figure 4.26: Panda Inside: Screenshot of scene Panda inside; Source: Own Figure

Important parts of this scene are:

- AR session
- AR Session Origin
- AR Camera
- 3x Directional Light
- Canvas
- 13x Lean Buttons
- 3x Planes
- 3x Input Fields
- 'Panda' Arm model
- ReferenceImageLibrary

In Figure 4.27, you can see that several GameObjects are disabled at the beginning of the scene. This is for clarity, as before. The button "Calibrate" activates and deactivates the calibration buttons. The button "Reconnect" activates and deactivates a panel with three input fields and two buttons that are needed for the connection.

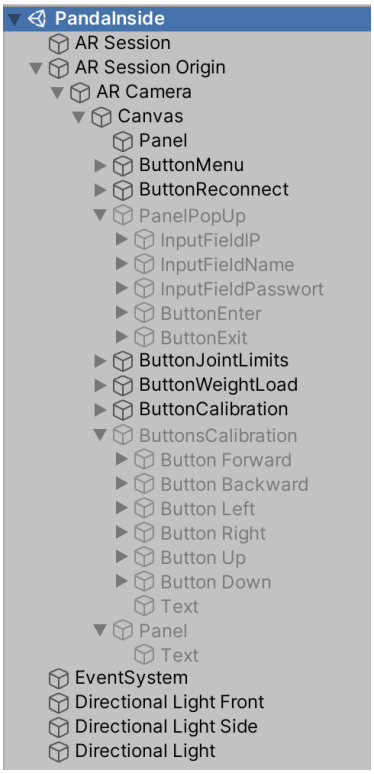


Figure 4.27: Panda Inside: Hierarchie Source: Own figure

The first difference to the other two main functions can be found in Object AR Session Origin. Instead of the Plane Manager, here you find a component called "AR Tracked Image manager". This component is included in the AR Foundation Package. To do this, create a library with photos (Figure 4.28). The dimensions of the 2D images are given in reality so that the camera can adjust the digital models' size to the distance to the image. The maximum number of images should not be set too high as this is a burden on stability. Also, you pass the prefab of the model you want to project on the tracked image to the script.

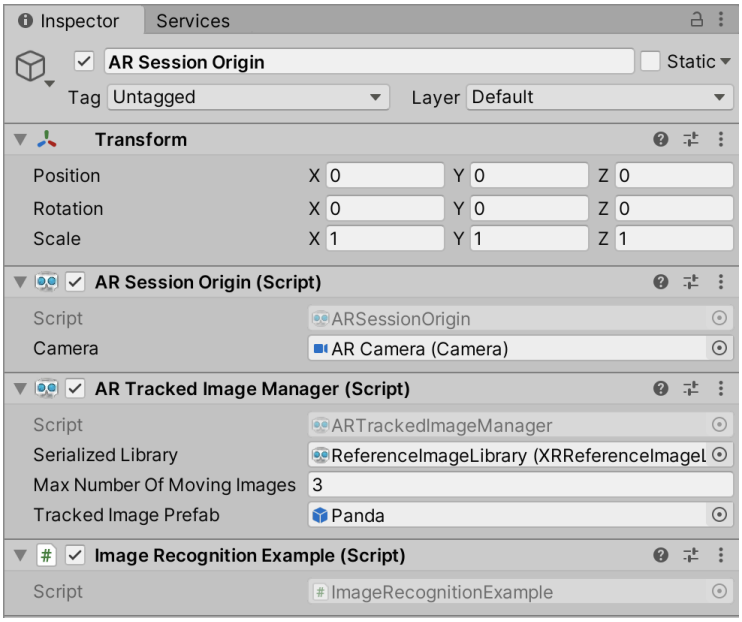


Figure 4.28: Panda Inside: Inspector of AR Session origin; Source: Own Figure

Since you work together with the robot and, therefore, always get into the picture with your own hands, the AR camera has been supplemented with a further component that significantly improves the optics while working with the 'Panda' arm. In Figure 4.29, the object "AR Camera" is shown. The only difference to the other scenes is the component "AR Occlusion Manager". This feature is included in this package since version 4.0 of AR-Foundation and tries to detect human hands and forearms. Normally, when intervening in the image, the digital model would be projected in front of the arm that came into the image. However, this is very annoying. Therefore AR-Foundation tries to recognize these body parts and if they are located in the front or behind the digital model. The result of this function can be seen in Figure 4.30. Surprisingly well, this function recognizes the distance to the hand and to the model itself.

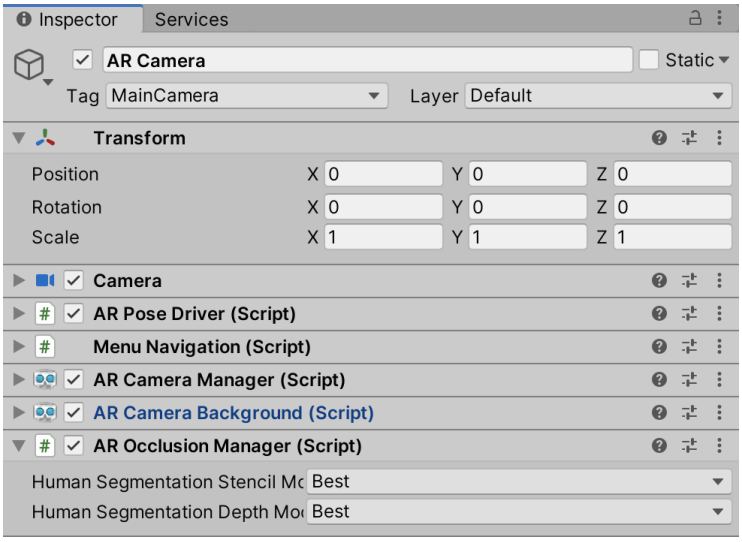


Figure 4.29: Panda Inside: Inspector AR Camera; Source: Own Figure

Figure 4.30: Panda Inside: Occlusion Manager demonstration; Source: Own Figure

So far, the Fear app recognizes the QR code and can project a semi-transparent model of the 'Panda' arm and a human hand and realistically place the model in front or behind it.

The only thing missing is the function to bring the whole thing to life with the real 'Panda' arm's live data. For this, the Fear app has to be in the same wifi network as a router

connected to the 'Panda' arm or its master controller and a PC with a running proxy server.

The Fear app will then make a request to the proxy server in this form:

"ws://fepc-win002.dev.kbee.lan:7070/desk/api/robot/configuration?user=franka&pass=franka123&hos

This address consists of several pieces of information:

- the prox server: fepc-win002.dev.kbee.lan
- the port: 7070
- the request for registration: /desk/api/robot/configuration?
- username: user=franka
- Password: franka123
- IP address of the 'Panda' arm: host=192.168.0.3

The first three components are hard-coded in the FEAR application for simplicity. For the use of real customers, you would have to find a remedy here. But this is not part of this thesis. Because the connection with different robots should be possible, this function was implemented. To establish a connection, the library WebsocketSharp is used in Unity. This can be easily imported within a script. With this library, WebSockets can be used, and in the case of this thesis, requests can be sent to the proxy server. As soon as a connection is established, a WebSocket with the URL created above is built and sent. If a connection is not successful, a standard model of the 'Panda' arm is placed in the scene until the connection is possible. As soon as the Fear app can connect to the master controller of the 'Panda' arm, a matrix with all the 'Panda' arm's current data is sent at regular intervals of 10Hz. Below are the joints' current positions and the forces acting on the joints themselves. This matrix is then readout, and the respective data is transmitted to the corresponding GameObjects in the scene.

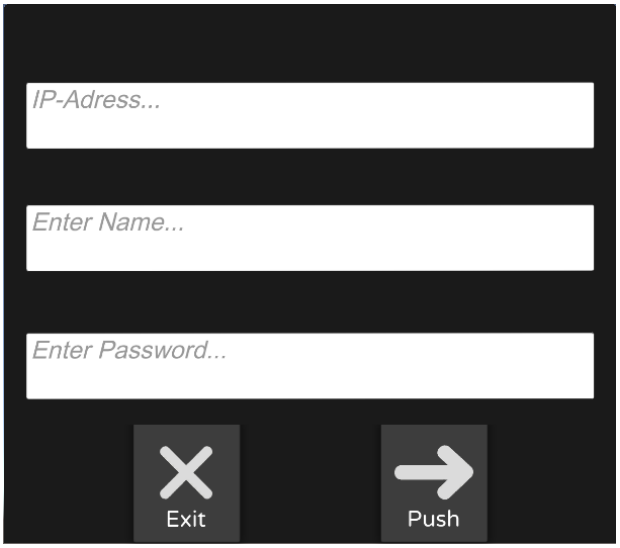


Figure 4.31: Panda Inside: Inspector input field; Source: Own Figure

When the user clicks on the button "Reconnect", a panel with three input fields opens (Figure 4.31). These are standard UI elements provided by Unity. Here the user must enter the username, password, and IP address of the 'Panda' arm.

As soon as you click on "Push" a string is built up, which is composed of the hardcoded parts and the variables. This string will be sent to the proxy server as a request. The whole thing happens in the "Reconnector" script (Figure 4.32). The button "Exit" closes the pop-up panel and has no other function.

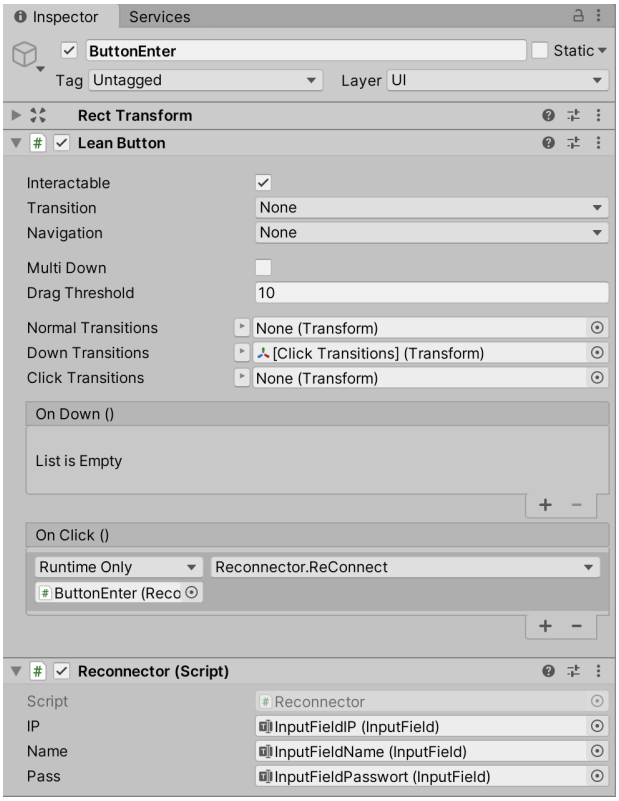


Figure 4.32: Panda Inside: Inspector Enter Button; Source: Own Figure

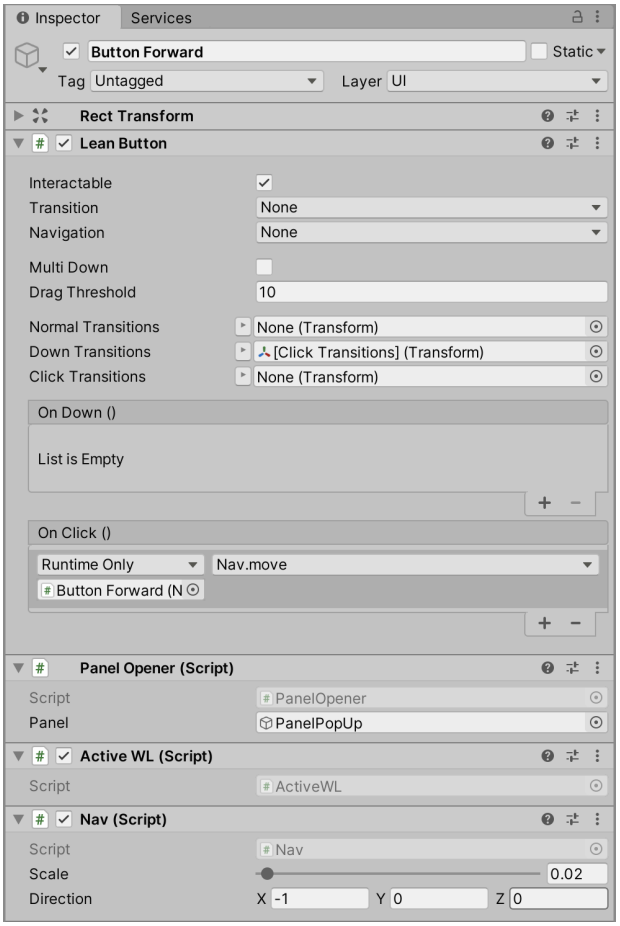


Figure 4.33: Panda Inside: Inspector calibration button; Source: Own Figure

Now the robot can be connected and is fed with live data. However, testing has shown that it is not user friendly to put the QR code in the same place. The models are not 100% accurate, and users may not always place the QR code in exactly the same way. Therefore another Lean Button was added. This activates six small buttons that move the model of the robot arm 2mm in the real world with every click (Figure 4.33). The "Nav" script is given a multiplier on a certain axis and a "Scale" set to 0.2 in the Fear app, which is 2mm real world. This value has proven to be practical. On the one hand, it is twice as fast as 1mm/click, and on the other hand, it is still accurate enough to make fine adjustments.

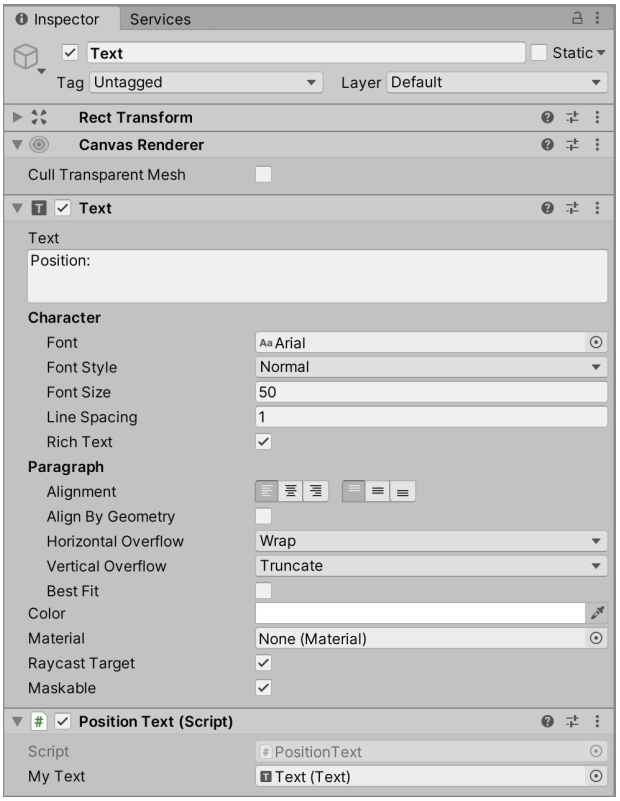


Figure 4.34: Panda Inside: Inspector PositionText; Source: Own Figure

Finally, a small text field is displayed in the calibration panel to show the user exactly where the bottom part of the 'Panda' arm is currently positioned. This has proven useful once you have attached the QR code and do not want to move the robot to the optimal position each time by fine-tuning. This way, you can find the optimal position once and pass it in Unity. As long as no 'Panda' arm has been initialized, "Position: -" will be displayed by default (Figure 4.34).

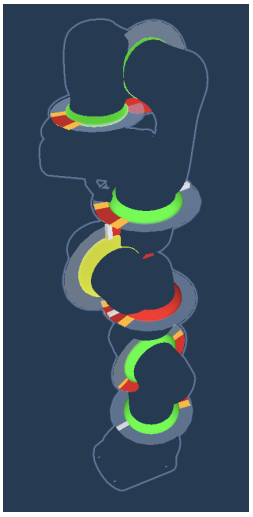


Figure 4.35: Panda Inside: Transparent model of the 'Panda' arm; Source: Own Figure

The partially transparent model is shown in Figure 4.35. The joint limits are shown in the same way as in the "Panda Demo" scene. Colored bubbles have been added around the joint limits in this model as well. This time they are not used to mark the slider but to show the power that is acting on the joint.

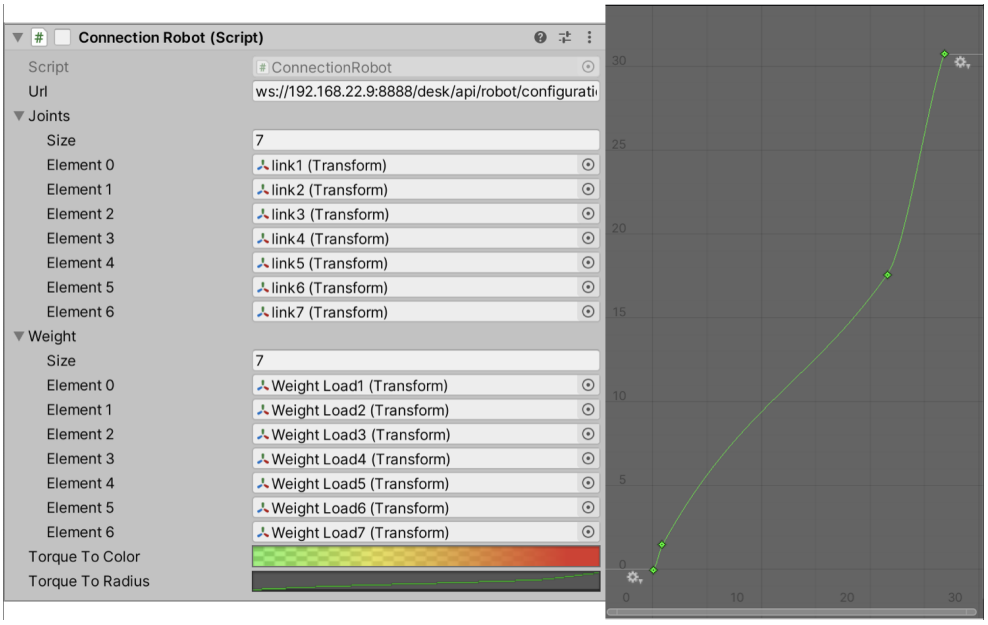


Figure 4.36: Panda Inside: Inspector 'Panda' arm; Source: Own Figure

Green stands for no or very little force and the more force added to the joint (0 - 30 Nm) the more the bubble turns red (Figure 4.36). This is possible because the script " ConnectionRobot" once implemented a gradient "TorqueToColor" and an animation curve "TorqueToRadius". As soon as the force values on a joint change, the color is adjusted accordingly.

Chapter 5

Methodology

5.1 User Study

5.1.1 Hypothesis

Almost all of the desired functions from the pre-study were implemented. The application is based on clearly recognizable and unambiguous symbols, and the application shows the possibilities that augmented reality could offer in the future. Specifically, how well otherwise hidden machine data can be visualized.
The hypothesis is, therefore, that the Fear application is easy and intuitive to use and brings value to the human-robot-interaction experience.

5.1.2 Topic and research question

Following the implementation of the AR framework, a user study was conducted to validate the results and assess the usefulness of such tools. It is also of interest which professional groups are participating in the study and how much experience the participants already have with the 'Panda' arm, as different functions like "Panda Demo" or "Garmi Demo" might be more or less useful for specific professional groups. This could be the basis for further development in this field. The participants are selected from different user groups. Each participant receives an introduction to the tool and is asked to test the different functionalities on a fixed structure. At the end of the demonstration, each participant is asked to fill in a questionnaire about their experience with the tool on a Likert scale of 1 (very poor) - 10 (very good).
The participants will also assess the usefulness of an application such as this in the future and how promising they think AR's technical possibilities in human-machine communication are.

5.1.3 Used materials

- 2x Franka Emika 'Panda' Arm
- 2x Franka Emika Master Controller
- 2x Franka Emika 'Panda' Hand
- 2x Franka Emika connection cable
- 2x Franka Emika pop-up box
- 2x LAN cable
- 2x FEAR App QR Code
- 1x Router
- 1x PC with Proxy Server
- 1x iPhone with the Fear application
- 1x iPad with the Fear application

5.1.4 Experimental set-up

The basic set-up of the experiment roughly consisted of a large open area and two 'Panda' arms placed on pop-up boxes (Figure 5.2 "C" "D"). Due to the current Corona Pandemic, the number of people in the room had to be limited to three. To do this, there was an entrance (Figure 5.2 "A") equipped with a disinfectant dispenser and a separate exit (Figure 5.2 "F") separated by a partition.

Further, in the room, the two 'Panda' arms are placed on top of the pop-up boxes (Figure 5.2). A QR code for the "Panda Inside" function is placed 5 cm apart on each 'Panda' arm (Figure 5.1). The free space for the functions "Panda Demo" and "Garmi Demo" was in the middle of the room (Figure 5.2 "B").

Figure 5.1: Experimental set-up : QR-Code: Source: Own Figure

After testing the application, participants could scan a QR code at the exit to take part in the survey (Figure 5.2 "E").

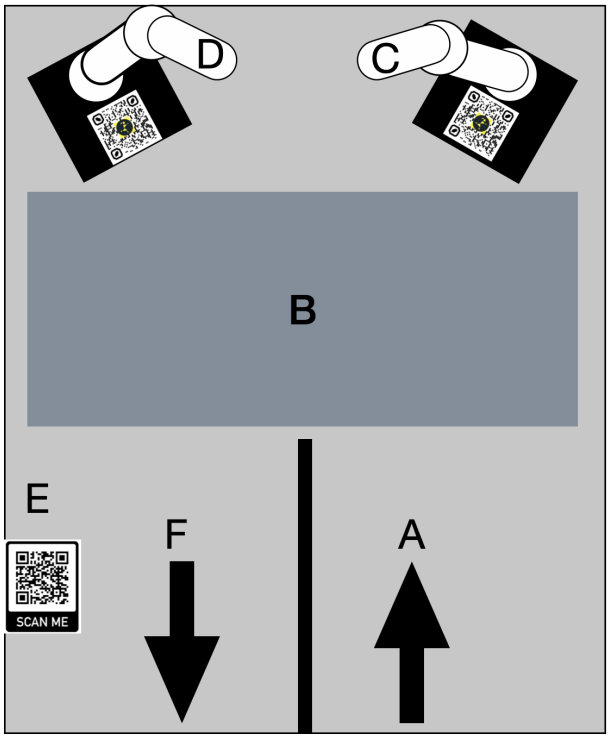


Figure 5.2: Experimental set-up: Set Up; Source: Own Figure

This experiment also requires a network to connect the application to the robots and display live data in the application. As shown in Figure 5.3, the router is the network where the two mobile devices (iPad iPhone), a PC, and both master controllers of the 'Panda' arms are located. The PC and the two master controllers are connected to the network with a LAN cable. The two master controllers' IP addresses are known to the proxy server, and the Fear app also needs this information. For the Fear app to establish a connection to a 'Panda' arm, a request must first be sent to the proxy server. This IP address and port must be hardcoded in the Fear application. The Fear application also sends the IP address together with the username and password for the master controller or 'Panda' arm. The proxy server is transparent to the Fear application, but it processes the requests and establishes the connection between the Fear application and the master controller. As soon as the connection is established, live data is sent from the 'Panda' arm to the Fear application at 10Hz.

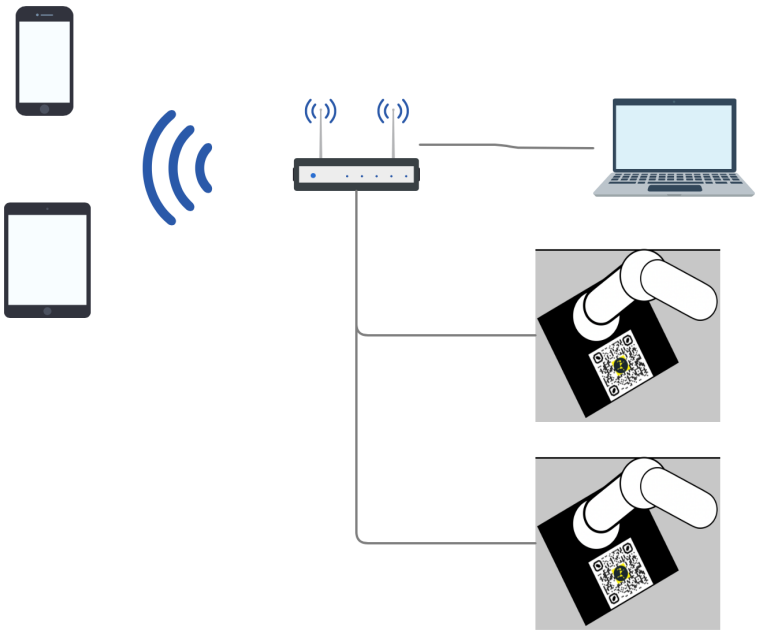


Figure 5.3: Experimental set-up: Network; Source: Own Figure

5.1.5 Experimental execution

The procedure for all participants is identical. The participant enters the room and disinfects the hands. The participant then receives the iPhone or iPad with the Fear App installed. Three tasks now follow for the participant of the study.

1. The participant should start the Fear App and get an overview of the main menu. Then the menu item "Garmi Demo" should be clicked on. Now the participant should

move the smartphone so that the application can find its way around the room and recognize its surroundings. As soon as the Fear app recognized the area, it will be colored. The participant should click on these colors to place the 'Garmi' model in the world. Finally, all functions, such as moving the model within the limits and stability, will be tested. Afterward, the stability of the models will be tested by rotating the end device. The participant has a few minutes to take a closer look at the function.

2. To test the second function, the participant should navigate back to the main menu and click on the menu item "Panda Demo". Again, the end device should be moved around so that the Fear app can find itself in the room and recognize surfaces. As with the "Garmi Demo" we should click on the colored, recognized areas to place the model of the 'Panda' arm in the world. Again, the model should be moved within the borders. However, in contrast to the "Garmi Demo" the 'Panda' arm's model is not animated. The participant should now click on the "Rotation" button to open a submenu with sliders. The user should now move these sliders to manipulate the model of the 'Panda' arm. The participant is reminded that the sliders and their respective joints have the same coloring. Besides, attention is drawn to the joint limits displayed. Afterward, the stability of the models will be tested by rotating the end device. The participant has a few minutes again to watch the function in peace and test it.

3. To test the last function, the participant must return to the main menu and click on the "Panda Inside" button. The QR code should then be scanned in front of a free 'Panda' arm to place the semi-transparent 'Panda' arm's model on this QR code. You can already see the standard model of the semi-transparent 'Panda' arm. To connect the Fear app to the 'Panda' arm, the participant should now click on the "Reconnect" button. Here the IP address, username, and password of the specific 'Panda' arm must be entered. This information will be given to the participant. The application automatically connects to the robot, and the semi-transparent model adapts to the 'Panda' arm's position. To place the 'Panda' model perfectly on the real 'Panda' arm, the participant must now click on the Calibration button and make any necessary changes. From this moment on, all preparations are done, and the participant can move the 'Panda' arm and, at the same time, see the changes in the joint limits in the Fear app. The user should then apply a force to the 'Panda' arm to see a color change in the Fear app. Afterward, the stability of the models will be tested by rotating the end device. Finally, the participant should move the 'Panda' arm to a joint limit to see a warning in the Fear app. Afterward, the participant has a few minutes to take a closer look at this function.

Before the participant leaves the room, the QR code should be scanned at the exit to participate in the online survey and to describe and evaluate the impressions the Fear app has left. Participants are also encouraged to leave constructive feedback in the online survey.

5.2 Results

5.2.1 Monitoring

Despite the long waiting times caused by the COV19 measure, 28 participants came between 2 pm and half past 3 pm to try out the Fear app. All participants were employees of Franka Emika and became aware of the experiment through a presentation in the weekly meeting two hours before. The first impression was that all participants were enthusiastic about the Fear app and the possibilities of the AR technology. In the middle of the experiment, there were problems with the network so that the participants could not test the "Panda Inside" function for a short time. To bridge this period, the 'Panda' arm was moved to the same position as the standard semi-transparent model of the "Panda Inside" function. In this way, the function could be shown even without the network function. The functions seemed to be mostly self-explanatory as most participants intuitively knew what to do. All in all, the feedback during the experiment was extremely positive.

5.2.2 Evaluation

As shown in Figure 5.4, most participants (40.9 %) rate themselves as 'Panda' experts, and 31.8 % of the participants claim to have average experience with the 'Panda' arm. This question was relevant for this thesis, as there might be differences in the Fear app's assessment depending on the participants' experience.

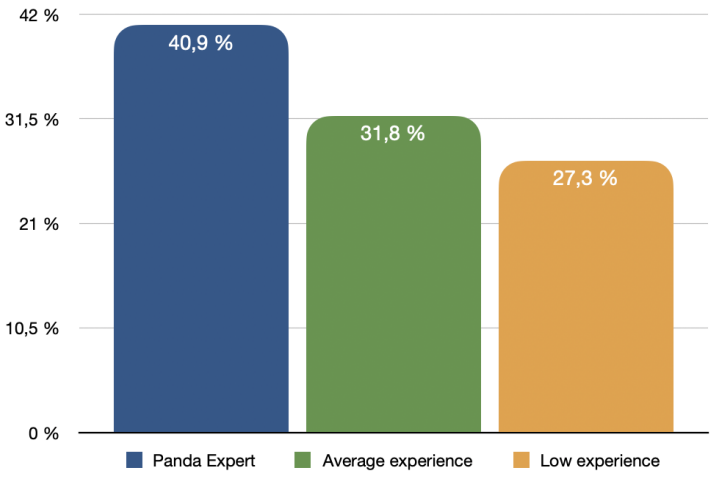


Figure 5.4: Results: How would you rate your experience with the panda?; Source: Own Figure

The same applies to the question: In which category of Franka Emika are you working? As Figure 5.5) shows, the distribution of employees was relatively balanced so that the statements from the study apply to the opinions of both professions.

However, in the following, it will be shown that all groups answered with approximately the same positive pattern in the questions.

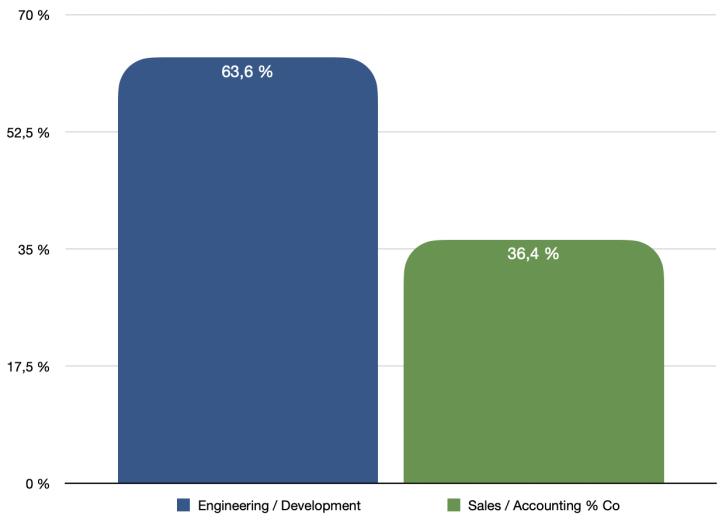


Figure 5.5: Results: In which category of Franka Emika are you working?; Source: Own Figure

In the following six questions, participants should give their opinion from 1 - 10 (very bad - very good).

As Figure 5.6 shows, 95.5% of all respondents found the Fear App useful or very useful. This is already a clear indication that almost all participants are satisfied with the app in general. Nevertheless, the next questions will ask for more detailed information.

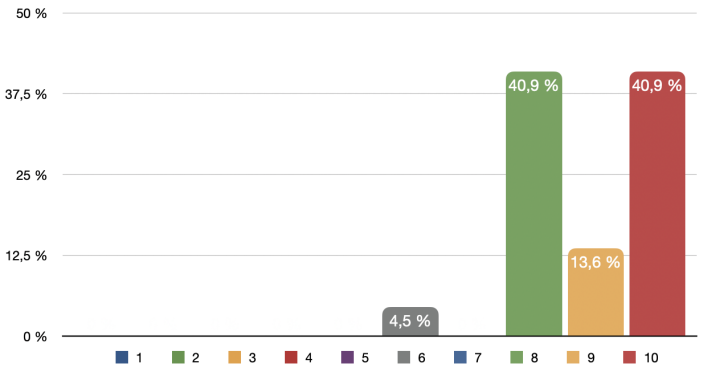


Figure 5.6: Results: How useful do you think the application is?; Source: Own Figure

The next question (shown in Figure 5.7) deals with the Fear app's benefits for partners and customers. Again, the benefit is seen to be positive to very positive. However, it is noticeable here that employees from the sales/accounting areas & co assess the benefits as even higher. Figure 5.8 shows this clearly.

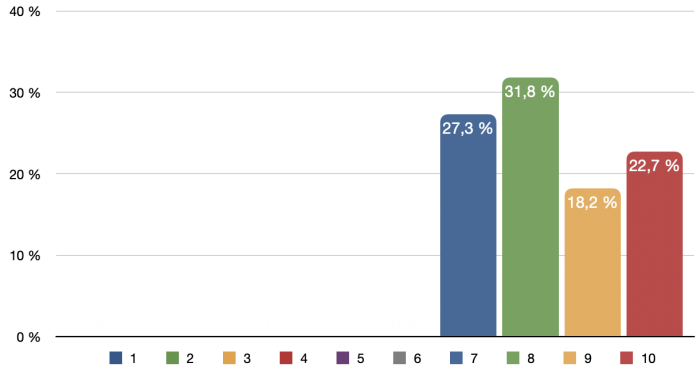


Figure 5.7: Results: How much potential and usability does an app like this have for customers and partners?; Source: Own Figure

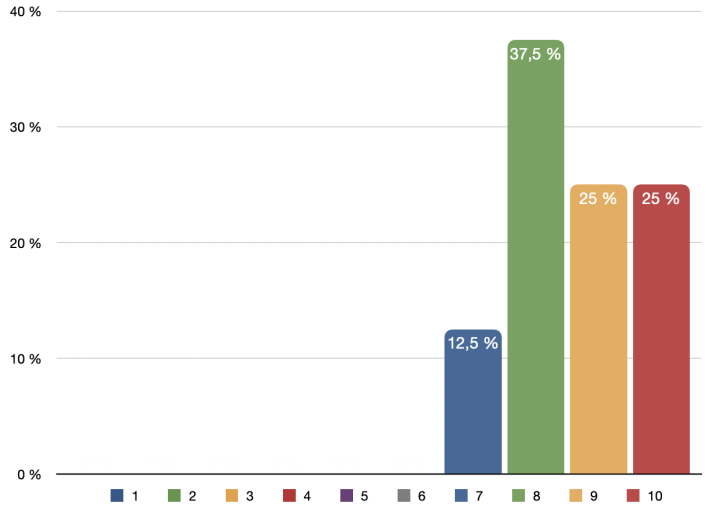


Figure 5.8: Results: How much potential and usability does an app like this have for customers and partners? Filter: Sales / Accounting & Co; Source: Own Figure

Since the study participants tested the models' stability by swiveling and rotating the end device, they could make a statement here. The result of this test and the subsequent feedback was extremely positive. Even if a participant turned away the device, the models remained in place and moved away from the digital model. Light to medium panning and rotating also had, if only minimal, influence on the models. Only with strong shaking did the accuracy fall behind. However, with the knowledge that all calculations only take place concerning the first detected area and some anchors, this result is quite impressive. The participants felt the same way. As you can see in Figure 5.9, 95.5% of the participants felt that the stability was good to very good.

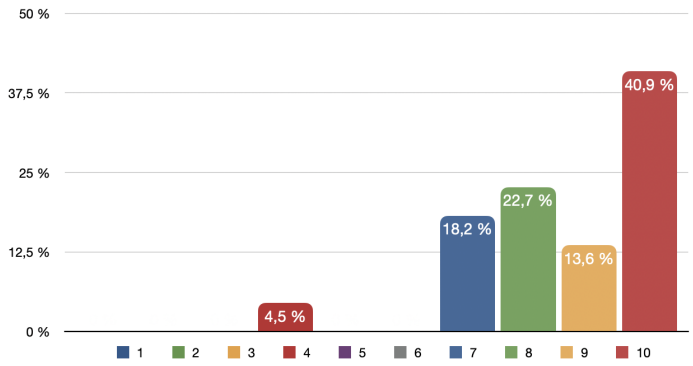


Figure 5.9: Results: How did you feel about the stability of the models?; Source: Own Figure

The sixth question of the study deals with whether the Fear App increases the 'Panda' arm's benefits. As Figure 5.10 shows, the results here look somewhat more neutral. As the Fear app is more supportive than function-enhancing, the participants' reaction was to be expected. However, the Fear app is intended to avoid mistakes in daily use and increase productivity, which explains the answers' positive orientation.

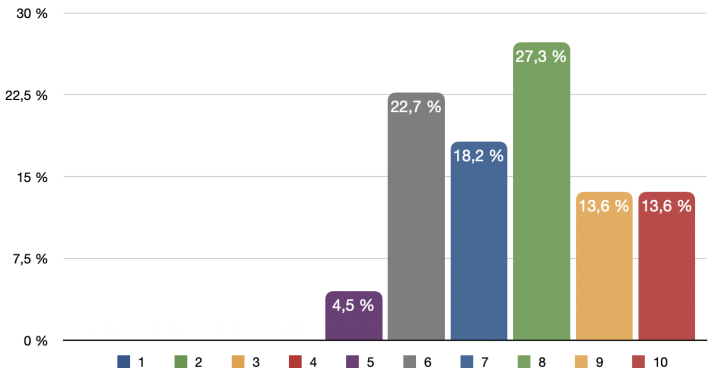


Figure 5.10: Results: How much does the application improve the use of the panda?; Source: Own Figure

In question seven, the study participants should evaluate the user interface's implementation, as was already evident in the experiment. Most participants had no problems with the application and had no problems at all. This is also shown in Figure 5.11. Here you can see an overwhelming majority of 63.6% who awarded 10/10 points for the user interface.

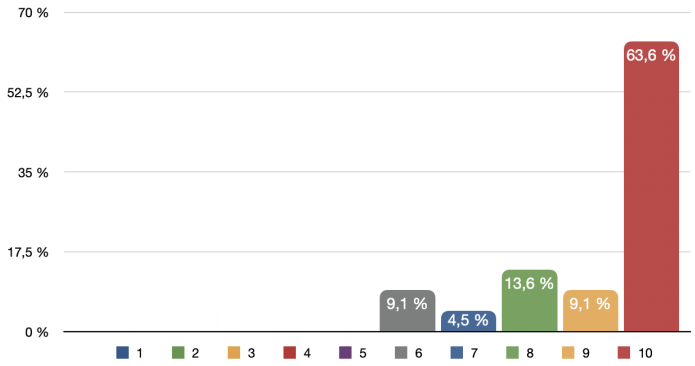


Figure 5.11: Results: Are the functions clearly presented in the application?; Source: Own Figure

In the eighth question, participants should sort the three options according to what they think is the greatest potential in the future. The results are given as an average. The lower the average, the more often the option with a high potential was attributed. The result (Figure 5.12) may seem trivial as the "Garmi Demo" function has the least actual benefit, and the "Panda Inside" function is likely to provide the most added value by providing new information while working with the 'Panda' arm. However, these interviews' assessments secretly confirm the usefulness of the "Panda Inside" function and thus the idea that much can be achieved with AR technology in the area of man-machine communication.

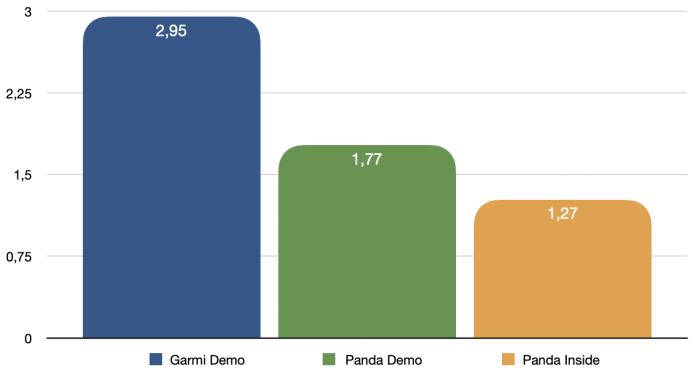


Figure 5.12: Results: Which function do you think has the most potential in future use?; Source: Own Figure

The final question of the study is to ask for an assessment of the future of AR technology. The respondents should indicate whether they think that AR plays a significant role in working with machines. They should also consider AR glasses. As shown in Figure 5.13, the result is clear, with 72.7% at 10/10. There is no answer worse than 7/10. The respondents consider AR technology to be fundamental, especially with AR glasses.

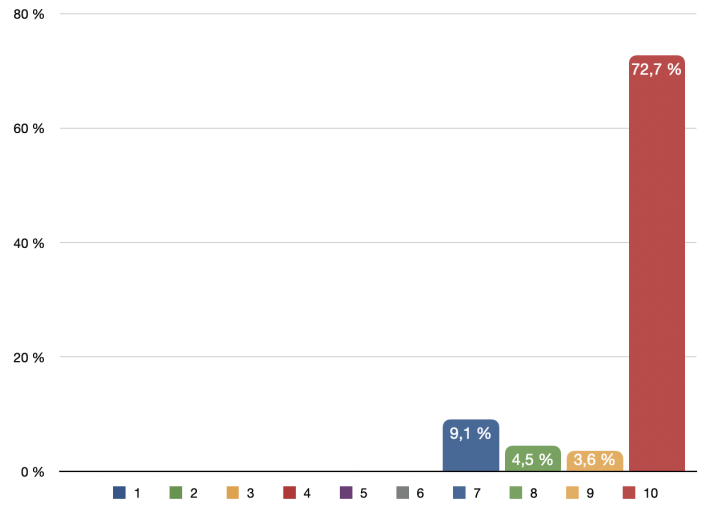


Figure 5.13: Results: Do you think that AR will play an important role in the future when working with robots and machines? Especially with regard to AR-Glasses.; Source: Own Figure

Finally, the participants should give personal feedback. Besides some short statements about the app's success, there were also technical remarks. It became clear that in the scene "Panda Demo" a possibility for cartesian motion would be good. Since the 'Panda' arm is characterized by the fact that the user does not have to move joint by joint but can simply be guided by the Franka hand. So it was suggested to integrate both possibilities of control in the scene "Panda Demo" to explain the difference to customers and interested people. The second most frequently mentioned was that an application with glasses like the HoloLense 2 would be optimal because you do not have to hold the device in your hand and have a larger natural field of vision. It was also mentioned that the color marking of sliders and joints in the scene "Panda Demo" should be highlighted more clearly. The topic of network connection was also taken up. In order to make this application really useful for the end customer or employee, integration with a normal Wifi connection without a proxy server should be possible. Another technical note was that the 'Panda' Arm has hard and soft joint limits. Hard joint limits are the joint limits implemented in the Fear app. Soft joint limits are the limits given to the 'Panda' arm so that it does not hit itself. These are important during the work with the robot as they also limit the 'Panda' arm's movement. Finally, it was suggested to implement a kind of tutorial that guides the user through the different functions.

Chapter 6

Conclusions and Outlook

The feedback provided by the user study was extremely positive and supported the expectations that arose during the thesis. The way the Fear app was implemented was well received and seemed to be intuitive. Almost all respondents got along well with the models and ads. The iPhone and the iPad as a platform which naturally limits this application. But it shows that even with the physical barriers these devices bring, excellent results can be achieved. This shows that Mixed Reality is well accepted by the end-users and is indeed useful. The study people were excited about the stability and possibilities that even simpler applications can bring. The feedback responses mentioned that this type of application should be unlocked for AR-Glasses to take full advantage. The question about AR's role in the future was also answered with a broad consensus for its importance. All this suggests that AR applications, combined with real machines, i.e., mixed reality, can fill a communication gap. Human perception meets and complements machine perception. In this way, sources of error could be avoided in the future, and productivity and safety in man and machine interaction could be significantly increased.

As already mentioned, this thesis is limited to mobile devices such as a smartphone. As a consequence, the stability of the models reaches the limits of the hardware. Since augmented reality is currently still based purely on optical effects such as image and surface recognition, it is not yet possible to display the data with millimeter accuracy. This thesis avoided this by keeping the digital model as transparent as possible to avoid direct comparison to the real 'Panda' arm. The first idea of the thesis was to look into the 'Panda' arm, but this idea was rejected due to the stability problem. Future models of smartphones and glasses like the HoloLens 2 will have LIDAR sensors so that a millimeter exact placement in space is physically possible [15]. Therefore, it can be expected that the stability of the models will be massively improved once again. Likewise, the previously mentioned network connection using a proxy server in the network is not optimally solved. Technically this problem is solvable, but it would have enlarged this thesis's scope and would not have contributed to the final result. Only for marketability, this function would

be necessary. The thesis question, however, could also be answered without a distinct network function.

For further research, I would recommend implementing a mixed reality application for devices like the HoloLens 2 and using the more suitable hardware to improve the user experience further. I would also recommend using the improved hardware to recognize the machine or robot as a model and not work with a QR code. If the machine is recognized as a whole, this should further increase the accuracy.

Developments of mixed reality in surgery [10], research by Microsoft [2], and this thesis's results give hope that mixed reality will find its way into daily working life. In a digital world, humans must also be able to expand their perception of this created world. Mixed reality applications like this could be one way to achieve this.

Appendix A

Notation and abbreviations

This chapter contains tables where all abbreviations and other notations like mathematical placeholders used in the thesis are listed.

AP	Access Point
AR	Augmented Reality
CQI	Channel Quality Indicator
DCI	Downlink Control Information
D-SR	Dedicated Scheduling Request
D2D	device to device
eNodeB	evolved Node B or E-UTRAN Node B
FDD	Frequency Division Duplexing
H-ARQ	Hybrid-Automatic Repeat Request
IoT	Internet of Things
LTE	Long Term Evolution
MCS	Modulation and Coding Scheme
OFDM	Orthogonal Frequency Division Multiplexing
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PRB	Physical Resource Block
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
RACH	Random Access Channel
SC-FDMA	Single Carrier Frequency Division Multiple Access
SR	Scheduling Request
SRS	Sounding Reference Signal
TDD	Time Division Duplexing
UE	User Equipment

Bibliography

[1] Apple. (o. J.). Augmented Reality. Apple (Deutschland). Abgerufen 30. Oktober 2020, von https://www.apple.com/de/augmented-reality/

[2] Apps, Services, and Solutions for HoloLens 2 — Microsoft HoloLens. (o. J.). Microsoft HoloLens 2. Abgerufen 30. Oktober 2020, von https://www.microsoft.com/en-us/hololens/apps

[3] AR Anchor Manager. (o. J.). AR Anchor Manager. Abgerufen 30. Oktober 2020, von https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.0/manual/anchor-manager.html

[4] Corporation, M. (o. J.). Guides — Microsoft Dynamics 365. Microsoft Dynamics 365. Abgerufen 30. Oktober 2020, von https://dynamics.microsoft.com/de-de/mixed-reality/guides/

[5] Franka Emika. (o. J.-a). Arbeiten mit Panda. Abgerufen 12. Oktober 2020, von https://s3-eu-central-1.amazonaws.com/franka-de-uploads/uploads/Arbeiten-mit-Panda-DE.pdf

[6] Franka Emika. (o. J.-b). Datenblatt ROBOTER ARM & CONTROLLER. Abgerufen 5. November 2020, von https://s3-eu-central-1.amazonaws.com/franka-de-uploads/uploads/Datenblatt-DE.pdf

[7] Herdina, M. (2020, September 28). Augmented Reality Disappeared From Gartner's Hype Cycle - What's Next? ARPost. https://arpost.co/2020/09/25/augmented-reality-gartners-hype-cycle/

[8] Immerman, D. (2020, Mai 14). How Augmented Reality Bridges the Gap Between Humans and Machines. PTC. https://www.ptc.com/en/blogs/corporate/augmented-reality-bridges-humans-machines

[9] Market share of operating systems in mobile internet usage in Germany 2009-2020. (2020, Oktober 5). Statista. https://www.statista.com/statistics/461981/mobile-operating-systems-internet-usage-share-germany/

[10] Marill, M. C. (2019, November 21). Hey Surgeon, Is That a HoloLens on Your Head? Wired. https://www.wired.com/story/hey-surgeon-is-that-a-hololens-on-your-head/

[11] Microsoft. (2020, August 26). What is Mixed Reality? - Mixed Reality. Microsoft Docs. https://docs.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality

[12] Milgram, P. M. & Kishino, F. K. (1994, Dezember 12). A taxonomy of mixed reality visual displays. A taxonomy of mixed reality visual displays. https://cs.gmu.edu/žduric/cs499/Readings/r76JBo-Milgram_IEICE_1994.pdf

[13] Niculescu, A. N. (2002, Januar 2). Mensch-Maschine Kommunikation. ethsis.net. http://www.ethesis.net/kommunikation/kommunikation.pdf

[14] Program-Ace. (2020, August 4). Unity vs. Unreal: What to Choose for Your Project? –. https://program-ace.com/blog/5-years-of-unity-vs-unreal/

[15] Puschen, F. P. (o. J.). Lidar scanner augmented reality wird salonfähig. Meedia. Abgerufen 31. Oktober 2020, von https://meedia.de/2020/10/14/lidar-scanner-augmented-reality-wird-salonfaehig/

[16] Required Device Capabilities - Support - Apple Developer. (o. J.). Apple Developer. Abgerufen 30. Oktober 2020, von https://developer.apple.com/support/required-device-capabilities/

[17] Shavel, T. (2019, August 13). ARCore vs. ARKit: Which Is Better for Building Augmented Reality Apps? Iflexion. https://www.iflexion.com/blog/arcore-vs-arkit

[18] Technologies, U. (o. J.-a). Multiplatform. Unity. Abgerufen 30. Oktober 2020, von https://unity.com/features/multiplatform

[19] Technologies, U. (o. J.-b). 's AR Foundation Framework. Unity AR Foundation. Abgerufen 30. Oktober 2020, von https://unity.com/unity/features/arfoundation

[20] The Science of Augmented Reality. (2019, Dezember 18). The Franklin Institute. https://www.fi.edu/science-of-augmented-reality

[21] Trepesch, S. (2018, Februar 6). Fünf Top-Features in iOS 11. GIGA. https://www.giga.de/downloads/ios-11/

[22] Unity. (o. J.-a). Unity. Abgerufen 22. Oktober 2020, von https://unity.com

[23] Unity. (o. J.-b). Unity Docs Physics.Raycast. unity Docs. Abgerufen 28. Oktober 2020, von https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

[24] Unity. (o. J.-c). Unity Manual Camera. Unity Manual. Abgerufen 25. Oktober 2020, von https://docs.unity3d.com/Manual/class-Camera.html

[25] Unity. (o. J.-d). Unity Manual Game Objects. Unity Game Objects. Abgerufen 27. Oktober 2020, von https://docs.unity3d.com/Manual/GameObjects.html

[26] Unity. (o. J.-e). Unity Manual Using Components. Unity Manual. Abgerufen 26. Oktober 2020, von https://docs.unity3d.com/Manual/UsingComponents.html

[27] Unity - Manual Event System. (o. J.). Unity - Manual. Abgerufen 3. November 2020, von https://docs.unity3d.com/2018.4/Documentation/Manual/EventSystem.html

[28] Unity Manual. (o. J.). Unity Manual Doc. Abgerufen 30. Oktober 2020, von https://docs.unity3d.com/Manual/index.html?_ga=2.139237367.573765990.1604249769-1437427150.1604249769

[29] Unreal Engine — Features. (o. J.). Unreal Engine. Abgerufen 30. Oktober 2020, von https://www.unrealengine.com/en-US/features