DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Improving photogrammetric 3D Reconstruction by Augmented Reality and Gamification based User Guidance

Simon Stolz

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Improving photogrammetric 3D Reconstruction by Augmented Reality and Gamification based User Guidance

Optimierung photogrammetrischer 3D Rekonstruktion durch Augmented Reality und Gamification gestützte Nutzerführung

Author:	Simon Stolz
Supervisor:	Prof. Gudrun Klinker, PhD
Advisor:	M.Sc. Linda Rudolph
Submission Date:	2/15/2021

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 2/15/2021 Simon Stolz

Acknowledgments

I would like to thank my advisor M.Sc. Linda Rudolph for continuously providing me with valuable advice and feedback during the thesis and also for being frequently available when I had any questions.

I am grateful to Prof. PhD. Gudrun Klinker, for supervising this thesis and thereby making my bachelor's final hurdle possible.

Big thanks to Süleyman Yasir Kula for providing me with example code on how to use his Unity plugins, that I then adapted to my thesis needs.

Thanks to the TUM English Writing Center for advising me on how to improve my English.

Abstract

Existing research has shown that augmented reality has the potential, to be used for user guidance in several fields of application. So far, there do not exist many examples, for applying augmented reality guidance to the field of photogrammetry, especially if applications that use video material instead of photos for reconstruction are not considered. This paper builds an augmented reality guidance concept for the photogrammetry picture acquisition process by combining existing research about photogrammetry and gamification. The concept is implemented with Unity 3D. The paper then shows that the application created can be used to improve scans but there still exists room for improvement.

Kurzfassung

Aus bestehender Forschung geht hervor, dass Augmented Reality Potential zur Nutzerführung in einigen Fachbereichen hat. Jedoch im Bereich der Nutzerführung für den Fachbereich der Photogrammmetrie gibt es noch wenige Beispiele, insbesondere wenn man 3D Rekonstruktionen aus Videomaterial außenvor lässt. Diese Arbeit recherchiert Grundlagen aus Photogrammmetrie und Gamification um daraus dann ein Augmented Reality Nutzerführungskonzept in Unity 3D als Forschungsgrundlage zu implementieren. Die Applikation zielt dabei besonders auf Nutzer ab, denen der Bereich der Photogrammmetrie noch unbekannt ist. Die Applikation soll ihnen zeigen, aus welchen Blickwinkeln Fotos von einem Objekt gemacht werden müssen und sie dazu ermutigen einen kompletten Scan durchzuführen. Zur Evaluation wird Feedback von einzelnen Testern herangezogen. Zudem werden deren Scan Ergebnisse mit der Applikation gegen deren Ergebnisse mit lediglich wörtlichen Anweisungen verglichen. Dabei ist die Anzahl der Bilder, die pro Scan von Meshroom erkannt werden ein Bemessungsparameter. Die Arbeit kommt zu dem Schluss, dass die Anwendung von AR und Gamification erfolgreich war, jedoch Potential zur Verbesserung besteht.

Contents

(a) Iron Man augmented Helmet from Iron Man 3 [2]

(b) Terminator Vision from The Terminator [1]

Figure 1.1.: AR in Movies [2][1]

1. Introduction

This chapter specifies the goals of this thesis and lays a knowledge foundation for the fields of photogrammetry and augmented reality (AR).

1.1. Motivation

AR has huge potential to simplify everyday life, it can provide additional information about the present environment and guide to goals of spatial, social, learning, or research-oriented nature. Movies provide many visions about the potential AR could have. The Terminator's information collection and object recognition capabilities used to help the T-1000 track down its targets and help the machine to learn about and adapt to its environment like in figure 1.1b, is one example [1]. Another can be seen in figure 1.1a showing Tony Stark in one of his suits that features an AR interface helping him to maneuver and manage his invention.

While universal recognition and guidance systems like these still lie in the future, working solutions for certain problems already exist. Two examples for that are Google Maps Live View, a tool to illustrate your google maps route in the real world with the help of AR [3], visualized in figure 1.1, and "NAVIG", research aimed at guiding vision-impaired users to their target destination [4]. Another field of research AR can potentially be applied to is photogrammetric 3D reconstruction. In that scenario, AR would then guide users to create 3D models from real-world objects, by taking pictures of them. In subsequent AR projects, these 3D models, amongst other things, can then be placed in their environment or be used as a reference for object tracking. This concept is called Augmented Reality for Augmented Reality (AR4AR) and was first introduced by Frieder Pankratz who applied AR to help users calibrate AR devices in his dissertation "Augmented Reality for Augmented Reality" [5].

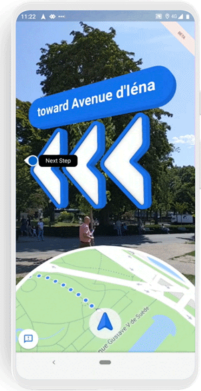


Figure 1.2.: Google Maps Live View in Action [3]

1.2. Thesis Goal

The goal of this thesis is to research a concept for user motivation and utilize existing publications and experience reports to model an optimal photogrammetry scan progress.

An AR-powered mobile application then puts the findings into practice. The application is used to assess if a positive effect on scan results can be achieved by guiding users with the help of AR combined with gamification. An additional performance indicator is how well the application motivates users to complete the scan. The guidance application later competes against verbal instructions in the evaluation process.

1.3. Augmented Reality and Mobile

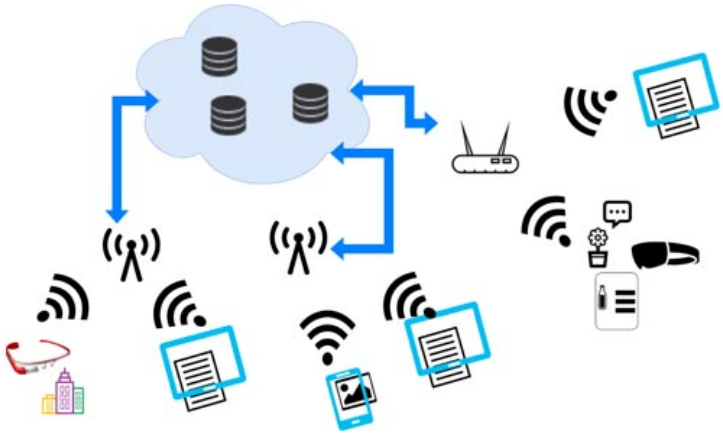
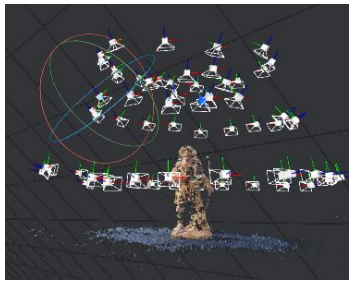


Figure 1.3.: Server Support for AR Devices

Pokemon GO is a good example of the success a mobile handheld AR application can have. Research in this field has continuously advanced in the last years with mobile phones

(a) Photographic Image



(b) Feature Points detected by Meshroom

(c) Mesh generated by Meshroom imported to Mehslab

Figure 1.4.: Photogrammetry Scan conducted for this Paper with the P20 Pro Front Camera and Meshroom

increasing in computing power every year. They are promising devices to bring AR to the masses since a big portion of the population owns one and they come with many sensors, needed for orientation in the device's environment. Already today AR is used in fields like spatial guidance, education, entertainment, museums, and more. A big limitation of mobile phones is that, by design, they need to be quite small to be handy, this limits their quick access memory, computational power, and storage size, which are necessary to execute consuming algorithms processing big amounts of data, needed to enable precise spatial orientation and object detection. Figure 1.3 illustrates a promising solution for the performance problem. The approach uses a cloud server to support mobile applications. The server receives data collected by the mobile devices, executes required computations on high-performance machines, and sends the results back to the AR devices [6]. This of course is limited by the bandwidth of the devices for which the availability of high-speed wireless access is needed. Once 5G, which is expected to be between 100x to 1000x faster than its predecessor 4G, is globally established as the new standard, it is to be expected that also huge advances for AR are in prospect [6][7].

Therefore, many problems this paper will run into in terms of detection and scene stabilization might be already obsolete soon, since for now all computations for this research are still done on the device.

1.4. Photogrammetry

A short introduction to photogrammetry is provided in the following, to establish required knowledge about the topic.

1.4.1. Basic Concept

Photogrammetry is the process of generating 3D models by capturing images of real-world objects, from different angles around the target. Once the needed images are captured, photogrammetric software can use the image data to calculate a 3D representation [8].

There are already several photogrammetric 3D reconstruction software on the market, like Agisoft Metashape, Alice Vision Meshroom, and Autodesk ReCap. This paper focuses on the picture acquisition part and Meshroom is used for the computation part. Taking the right pictures is crucial for the success of generating good quality 3D scans, this requires experience and knowledge about how and from which positions pictures need to be taken.

Figure 1.4.1 shows an example scan conducted in an environment with natural light, using the camera of the P20 Pro. In figure 1.4a, can be seen what the scanned object looks like in real life and in figure 1.4b the feature points and camera position estimations created by Meshroom are displayed. The positions show how many images needed to be acquired for the scan to result in a Mesh like in figure 1.4c. In this context, it also has to be noted that when the lighting differs too much in the different images, the environment contains many reflective or transparent objects, or the images contain too little overlap with the other images, the reconstruction software will just discard them.

1.4.2. Fields of Application

There are many fields of industry where photogrammetry comes in handy. It is an effective and easy to access means to analyze object structures, inspect the condition of a certain aperture, and translate real-life constellations into digital form for the fields of automotive development, wind energy and oil pipeline maintenance, subsea applications, and mining [9]. In addition to the manufacturing industry photogrammetry is also an essential way to generate assets for video games, like for example in the Battlefront franchise, Battlefield 1 and Assassins Creed Unity [10] [11]. Photogrammetry can also be used to help preserve and restore cultural heritages like the Notre-Dame that partially burned down in 2019. Photos and already existing photogrammetry scans from research, tour guides, and material collected for the game Assassins Creed used for the 3D Model shown in figure 1.5 are all helping to reconstruct Notre-Dames ceiling [11]. In contrast to restoring destroyed sites, it can also be used to assess the natural decay of antique cultural sites and help protect them from the traces of time or at least provide a 3D snapshot of their current beauty for future generations [12].

1.5. Structure of the Thesis

This thesis begins with analyzing related research aiming to determine research gaps and to inform about existing findings relevant for this paper in chapter 2.

After that the guidance concept is derived from existing publications in sections 3.2 and 3.1 it is further elaborated and adapted to this researches uses in section 3.3.

Chapter 4 focuses on how the guidance application was structured and which functions were implemented. Subsequently, the functionality and usability of the mobile application are evaluated in chapter 5 to solve this research´s question.

Eventually, in chapter 6 this paper's problems and findings are going to be wrapped up.

Figure 1.5.: Notre Dame in Assasins Creed Unity by Ubisoft [11]

2. Related Work

This chapter sheds light on related research and projects, to determine the research gap this project is going to fill, for the current state of the art.

2.1. Improving Museum Experiences with Augmented Reality and Photogrammetry

As mentioned in subsection 1.4.2 photogrammetry is used for preserving cultural heritage. Supported by AR, it can also make encounters with ancient cultures, civilizations, and art more memorable and enjoyable. The Google Tango project is a fitting example of this. Tango makes the museum experience of visitors more interact-able, by providing additional information about the exhibits. Figure 2.1 shows how Tango uses a regular mobile device to provide visitors with x-ray vision into a mummies sarcophagus [13].

Another example is the application named Whole, by Valentin Josef Beck. It shows visitors how exhibited statues have looked like before they suffered from natural decay, as depicted in figure 2.2 [14]. Projects like these, show the potential of what AR and photogrammetry joined together can achieve. They also indicate how advanced AR environment tracking is today, which is going to be crucial for this paper's research.

Figure 2.1.: X-Ray View into a Sarcophagus [13]

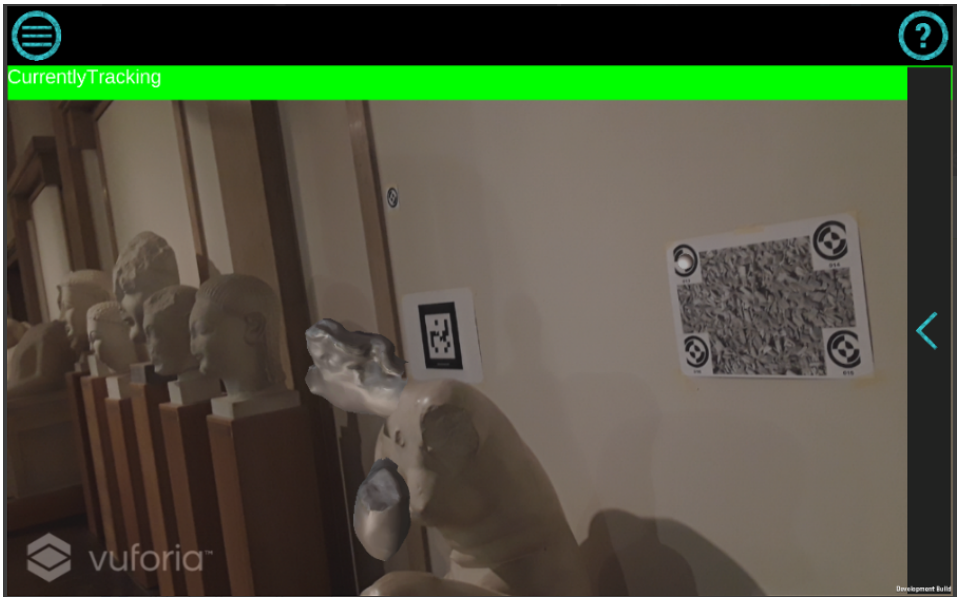


Figure 2.2.: AR reconstructed Statue located at Museum für Abgüsse klassischer Bildwerke in Munich [14]

2.2. display.land Photogrammetry Assistant

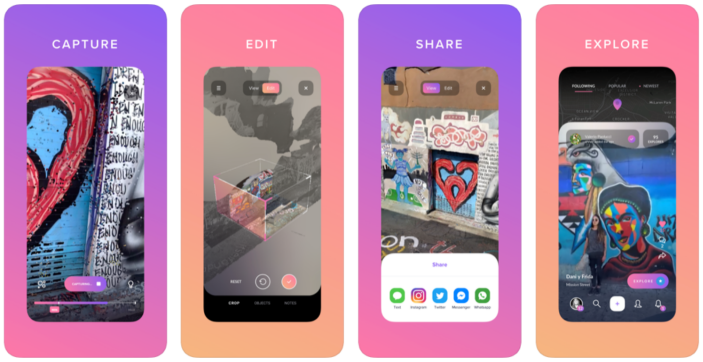


Figure 2.3.: The basic Functions of display.land [15]

The next candidate, display.land, is even more closely related to the thesis than the preceding examples. It guides users to film videos of real-world objects. The video material is then uploaded to a server that uses the material for photogrammetric 3D reconstruction and provides the results as a download.

The main difference between display.land and this research is the way the scanning process is conducted. While display.land relies on videos to create 3D objects this research uses single pictures to reconstruct an object. This also leads to different guidance concepts since Display.land needs to steer users in certain paths around the objects while here only the

viewpoints the pictures are taken from matter. Besides, this paper focuses on guiding the picture acquisition process for only smaller objects, instead of a universal guidance style [16]. Display.land was sunset on August 11th, 2020, to make room for future products. Therefore, during this research, there was no possibility to investigate functions of the application in detail, in particular getting more specifics on how the guidance concept worked is hard. The shutdown can also be interpreted as an indicator of how experimental this field of research still is [17][15]. In figure 2.3 an illustration of the basic functions of display.land can be seen. Next to guiding users to capture photogrammetry material, the app also provided functions to edit generated models on the phone and share created art with other users over social media and the app [15].

2.3. Scanning and Detecting 3D Objects with ARKit Demo

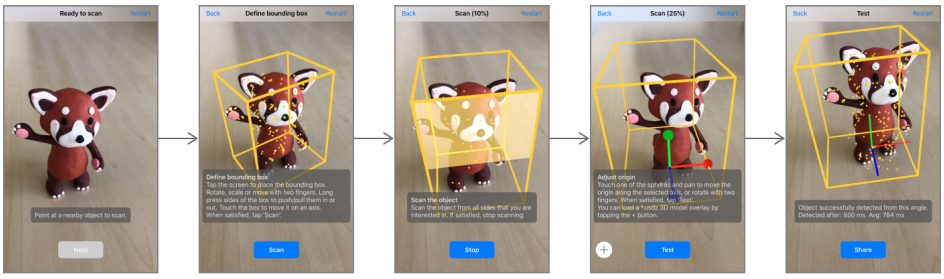


Figure 2.4.: Scan and Detection of 3D Objects with Apple Demo [18]

For iOS 12 Apple released the ARObjectScanningConfiguration function. It is used to reconstruct real-world objects, the models of which then can be imported into AR projects as ARReference objects, to be detected in the environment. Screenshots of this application can be seen in figure 2.4.

The first difference to this research is, that similar to the related work mentioned before it does not use single images, but continuous environment capture to create models. Photogrammetry is not mentioned explicitly in the documentation, therefore also other sensors might come into use for example a LiDAR Scanner that some Apple devices have.

The second difference is that the target of the function is not to create high-quality models, but ones good enough to be used for real-world object recognition. Third, because the scanned objects do not require the highest possible quality, the 3D model is generated on the device from depth data, while this research will use Meshroom as external reconstruction software that only uses image data [18].

Similar is how the scan process is structured. The Apple Demo uses a bounding box to specify where the scanned object stands and then guides the users around it and shows them from what angles and heights features are captured, this is in general similar to the goal of this paper, despite that this paper is using fixed viewpoint positions relative to the target.

3. Guidance Concept

To put existing findings to use they are projected onto this research's specific field of application. In detail, aspects from gamification and existing picture acquisition guidelines, for photogrammetric reconstruction, are taken and combined, to help users find the optimal camera angles and keep them motivated during the process.

3.1. Optimal Picture Acquisition Process

General Findings Finding the right equipment and settings for optimal 3D photogrammetric reconstruction is a complex field on its own yet in this paper, the usage of mobile phone cameras will be assumed. This way of taking pictures is not ideal for any scenario but is the best fit for this research, because it works well enough to yield decent results and is the only equipment that can be integrated into a mobile AR application, without additional effort [19].

The target object's surface and response to light are important factors for photogrammetric 3D reconstruction as well for example objects with reflective or transparent surfaces are nearly impossible to detect, as displayed in figures 3.2a, 3.2b, and 3.2c. On the contrary, objects with rough surfaces and no natural movement, like leaves in the wind do, are excellent scan-subjects. For example, a fabric shoe or a house, with curtains covering the windows, have the potential to result in quality scans, like those shown in figures 3.2e and 3.2d [20].

The optimal composition of viewpoints, needed for a successful scan, is modified by the scan target's size and level of detail. Capturing the right photos for a scan process gets increasingly difficult the bigger the object is, for several reasons. First of all more pictures need to be taken second areas with different degrees of detail need an adequate amount of attention third the height in which pictures need to be taken continuously increases with object size [19]. A house for example is too tall to be scanned without a drone or several higher outposts, also details, like the railing of a veranda, need additional attention. How this complicates the picture acquisition, can be seen in figure 3.2e which visualizes the positions images were taken from for reconstruction. This project only considers target objects fitting into cuboids with 0,05 m to 1m side lengths, so that all pictures needed can be taken, with only using a mobile phone. On the downside, this decision restrains the flexibility of this paper's research foundation. On the upside, once the application can guide users to scan smaller objects, experience gained during app usage is expected to apply to larger targets as well, since according to the expertise of the journalist Ben Kreimer experience gained by scanning small objects is easily transferable to scanning larger objects as well [19].

Environmental influences like weather, lighting, and movement of the object and surroundings will not be investigated, since the focus of this research is the scanning process and view

angle positioning.

Findings regarding the optimal Scanning-Process During researching documentation on optimal scanning processes, two combine-able principles were discovered.

The first one usually splits the scanning into three phases, wherein each phase pictures are taken from a different height relative to the object. In other words, the angle between a horizontal plane, through the object's center, and the view positions increases with each phase, in the possible range from 0° to 180°. During each phase pictures are taken from around the object in the phase's specific confines, all the images should overlap for about 70-80 percent [20, 21, 22]. In figure 3.1 can be seen how the described scheme looks for a medium-sized object. For smaller objects, that do not have the level of detail and spatial complexity of for example a tree trunk like in figure 3.1, three narrow rings of picture positions suffice. The basic concept stays the same for objects of all sizes [19]. If a very huge and detailed object is scanned, like a house, the explained process also needs to be completed before further scanning. Taking pictures around the object from an adequate distance is necessary because the photogrammetry software needs that to be able to orient itself, to approximate the position of additional closeup shots [19].

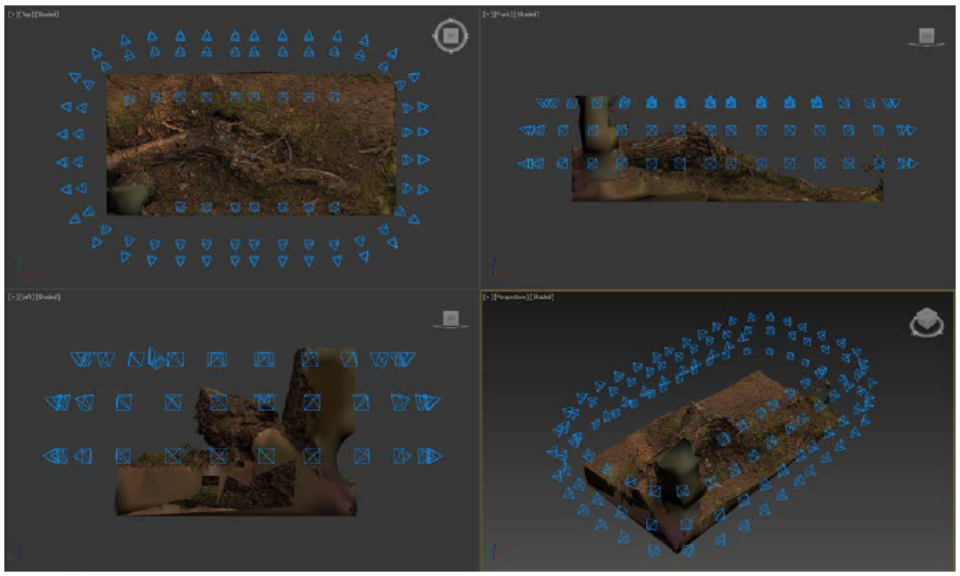


Figure 3.1.: View Angles for a uniform Scan Process from "Photogrammetry Workflow" by Unity [20]

Second, especially for larger objects and those for which important features are covered from the sides and above, like the slats of a mushroom are, the quality of photogrammetry results can be increased when images from closer viewpoints and irregular angles are captured. For larger objects, this is also needed because some features might have a higher degree of detail than others. To scan those optimally, more and also pictures from a closer distance need to be taken [19, 22].

The difficulty of this approach is that the amount of detail varies individually for each object, therefore there is no uniform way to guide for this approach, without a method of run-time object detail detection.

3.2. Gamification as Motivator

This section establishes basic knowledge about the topic of gamification and then proceeds to describe how it can be used to motivate users in a serious context.

3.2.1. Introduction to Gamification

According to a study, conducted by Limelight Networks, the average person plays about seven hours of games per week [23]. The idea to translate this extraordinary retention onto other fields, like for example projects in the field of education [24] or nutrition regulation assistance [25], is called gamification [26]. Before gamification can be used on an application to stimulate a certain behavioral result, the elements of a game that increase user-motivation have to be extracted. For this purpose, Legner et al. [27] particularly investigated mobile games. Since they are often free to play, to earn money they focus more on user captivation and motivation, towards in-app purchases, than actual gameplay quality, compared to pay to play products. In their finding process, they searched for application elements that had motivating effects and categorized them into several psychological stimuli concepts, for example, the urge of loss aversion and that fixed schedules make it easier for humans to adapt behavior. All elements found were barely reliant on a genre-specific game-play concept, but were rather based upon virtual reward and resource systems [27]. This is an important revelation since it shows that the elements that agitate us to spend time and money on games, can potentially be applied to any field of application that adds evident value to people's quality of life, without a huge design effort investment.

3.2.2. Gamification Elements

The design elements found were: introduction of a player level system, quests, resources, item acquisition, gifts, rewards, and content unlocking, shown in more detail in figure 3.3 [27].

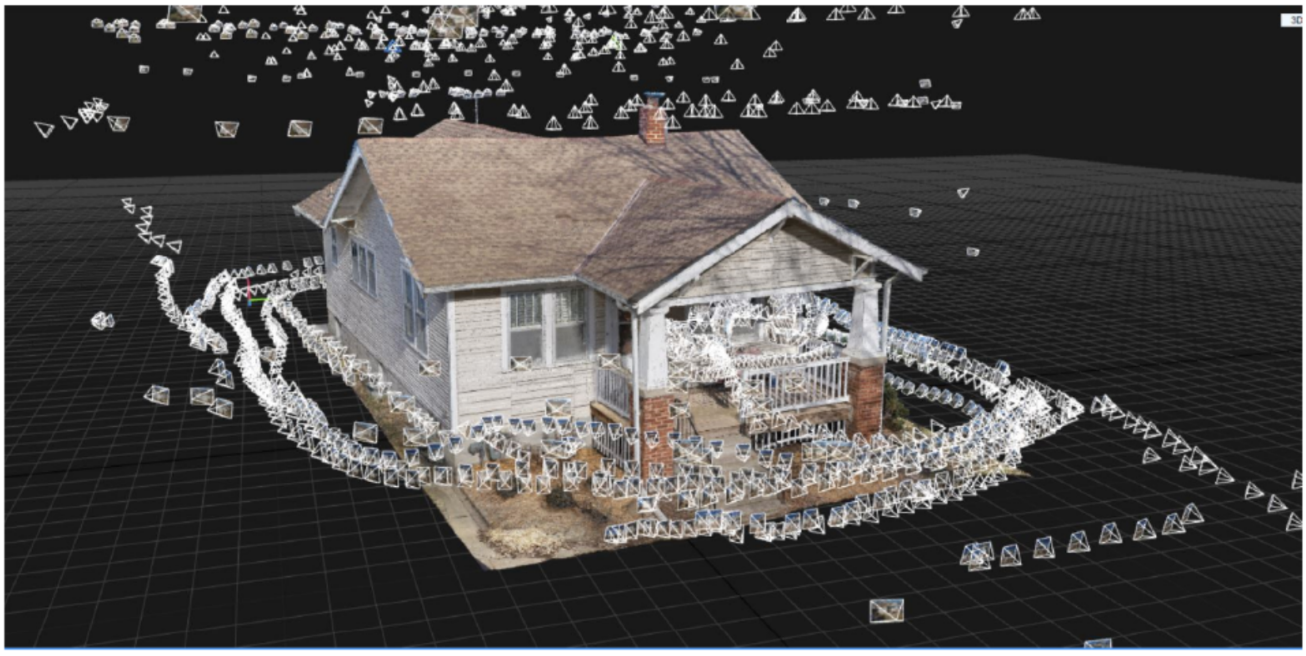
To determine which extractions fit this research, it needs to be determined which elements are most effective for a guidance use case. Legner also researched the effectiveness of the elaborated elements for the language learning app Duolingo as his test subject [24]. He found that gamification methods that punish users for failing can on the one hand motivate some users to spend more time learning, but on the other hand they are also prone to frustrate users. Examples of such elements are streaks that require the daily use of the app to increase in progress and a competitive leader board. The systems that were received the most positive, were success tracking, and positive feedback functions like milestones, lesson mastery, levels, progress feedback, and streaks. Streaks, being part of both categories, are an example that rewarding and punishing users, relative to their success, has a polarising effect on users.

(a) French Press

(b) Mug

(c) Tiffin

(d) Shoe

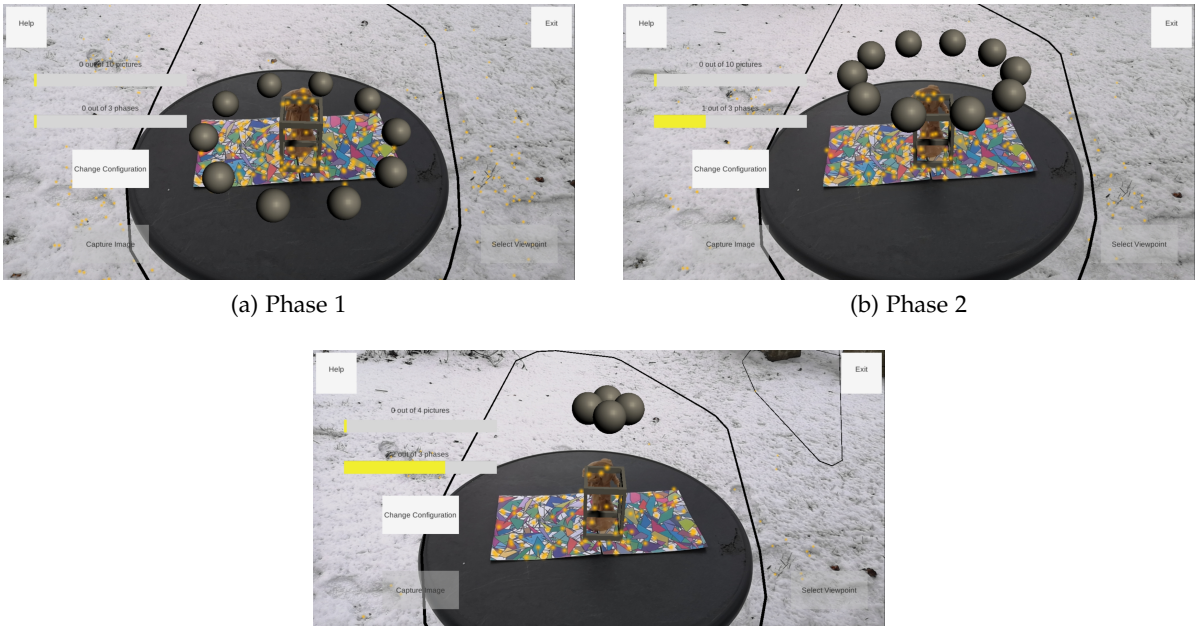


(e) House

Figure 3.2.: Scans conducted by Ben Kreimer [19]

Base Mechanic	Persuasive Mechanic	Psychological Theory	Explanation
Player Level	Level-Up Rewards	Goal-Gradient Effect	Players are motivated to work towards reaching the next level.
	Dynamic Experience	Endowed Progress Effect Goal-Gradient Effect	Experience gain is accelerated at the start of the level and slowed down towards the end to maximize the impact of motivating effects.
Quests	Daily Quests	Fixed Interval Schedule Loss Aversion	Players check back on the game once a day to avoid losing rewards from the daily quest.
	Quest-Quests	Goal-Gradient Effect	Completing a number of of quests is made more appealing through a bonus given out at the end.
	Retroactive Q. Introduction	Endowed Progress Effect	Quests given out with part of it already completed are more likely to be finished.
Resources	Interval Resource Collecting	Fixed Interval Schedule	Players can set up tasks (e.g. crops or material production) that require them to come back after a fixed amount of time to collect the reward.
	Energy System	Hedonic Treadmill Loss Aversion	Limits the duration of a play session and creates an incentive for people to come back to the game after some time to not lose out on energy regeneration.
Item Acquisition	Random Items	Variable Ratio Schedule	Random rewards motivate players to engage in an activity with persistent effort to obtain a desired item.
	Rotating Shop	Fixed Interval Schedule	Players are incentivised to check back on the game regularly to see if the shop has an item they want to purchase.
Gifts	Welcome Gifts	Endowed Progress Effect	Gifts early in the game generate the feeling of a headstart and players are more motivated to progress into the game.
	Fixed Interval Schedule Login Bonus	Fixed Interval Schedule	Players are incentivised to come back to the game regularly to collect the free reward.
Rewards	Reward Satiation	Hedonic Treadmill	Limited rewards within a play session lead towards putting the game aside and coming back later to keep the playing experience fresh.
	Time-Limited Rewards	Loss Aversion	Players are more likely to play the game until the reward duration has run out, as wasting it by not playing the game would cause dissatisfaction.
Content Unlocking	Exhibition	Goal-Gradient Effect	Future content is displayed in play space to give players a goal to work towards and therefore increases motivation to progress in the game.

Figure 3.3.: Persuasive Mechanics from "Persuasive Mobile Game Mechanics for User Retention" by Legner et ali. [27]



(a) Phase 1

(b) Phase 2

(c) Phase 3

Figure 3.4.: Viewpoint Constellations for each Phase

Logically it is advisable to avoid user frustration, as when losing a streak frustration can lead to demotivation.

For Duolingo, in-app resources were rated as the least satisfying gamified feature, possibly because there is no real goal to achieve with the help of these currencies, like in a building game where users need them to progress through the game [24].

3.3. Derived Guidance Concept

The guidance process was split into three phases. In the first phase, the target is photographed from even height, to capture the sides of the object in detail.

Starting below even height might be useful like in the mushroom example from section 3.1, but cases like this should be rare and require individual perspectives to take pictures from, therefore it is neither easy nor reasonable to attempt to standardize a scanning process for them in this paper.

The second phase takes pictures from a 40° angle and the third from an 80° angle. After every phase has been completed successfully, the user is returned to the menu. In figure 3.4 can be seen, how the viewpoint markers are positioned in practice.

The guidance application contains three motivational elements. Because regarding gamification, it was found in subsection 3.2.2 that rewarding the user for achievements and keeping them up to date with their progress proofed to be most effective, the gamification elements all serve this purpose. Besides that, also avoiding unnecessary sources of frustration was found

to be essential to keep users motivated.

For the guidance application, the focus lies on short-term motivation since the primary goal is to captivate users during the scan process.

The first element comes in the form of two progress bars, their purpose is to give users feedback about how much of the scan process they have already completed. One bar displays the number of phases completed, the other one shows the number of images taken so far relative to how many need to be taken to complete the current phase, these two bars can also be seen in figure 3.4.

Second, the application congratulates the user every time he has completed a phase, this belongs to the category of positive feedback elements and intends to make the user feel successful every time he completes a phase.

The third and last element is using a reoccurring sound that plays every time the user has successfully captured an image. This element on the one hand is a form of positive audio feedback and on the other hand, makes use of the mere exposure effect. The mere exposure effect constitutes that the more familiar people are to a phenomenon the more likely it is that they like it [28]. Playing a sound repeatedly with every success, therefore makes it feel increasingly rewarding with every time the user hears it, in theory. Deciding which sounds are the best for this purpose on a scientific level is hard to do since user studies are not possible during the Covid-19 pandemic. Having a distinct sound playing when a phase was completed might be something that intuitively should have a positive impact as well, this idea can be confirmed or debunked in future research.

As already mentioned for motivation it is also important to keep the user from getting frustrated. Sources of frustration can be for example the UI and technical errors. For this research, a help function is intended to reduce frustration. It has two functionalities, on the one hand, it displays information about the scan process and provides instructions on how to use the UI on the other hand it can be used to skip the remaining view positions in the current phase that comes in handy if some viewpoints are not reachable. More information on the design of the help function can be found in section 4.8. To reduce frustration due to technical problems, a possibility to reconfigure the scene should exist for the application.

By slightly shifting this paper's goal from only guiding users on how to conduct scans to adding the goal of motivating users to practice, what the guidance system teaches them, by conduction several scans, also long-term motivation becomes important. Regarding long-term motivation, a level system, achievements, and positive social interaction are promising methods.

To lay a foundation for future work a level system, that rewards experience for capturing images was implemented. In addition to that, a shop that allows to buy and equip new models for the view position indicators from in-app currency gained on leveling up exists. The build being evaluated in chapter 5 does not implement any of the long-term motivation functions mentioned before. In the same directory [29], the evaluation build is provided, also exists a build with the basic level system and the items for currency shop. It is located on the branch called "longterm motivation".

While for Duoling, according to Legners findings [24], the shop was seen as an ineffective

motivator, it can instead be used as a connector for different long-term motivation approaches, used in future work. For example, the shop could be modified to display rewards from level-ups and achievements, instead of being a traditional currency-run store. It could also provide access to new social interactions, like unlocking the possibility to share your successful scans with friends on reaching a certain level.

4. Mobile Application

The mobile application puts user motivation concepts and the blueprint for an optimal photogrammetry-based 3D scan into practice. In particular, it guides users to capture images from optimal positions, so that good reconstruction results are achieved. During the process, the user is continuously encouraged to continue capturing the remaining required images.

The app has been developed with Unity 3D using AR Foundation and ARCore to bring AR-related functions to the target platform Android and was subsequently tested on a Huawei P20 Pro. Switching to iOS would be with ease possible since all plug-ins used are viable for both platforms and with the help of AR Foundation, the AR implementations in Unity are cross-platform compatible as well, but chapters 4 and 5 exclusively talk about the Android scenario.

4.1. Main Functionality

The application provides two groups of functions first those dedicated to guiding users through the 3D scanning process second those keeping the users motivated during each step of the scan process. The functions are aimed at meeting the requirements specified in chapter 3. Section 4.4 provides more detail on the structural nature of the guidance process. How motivation focused requirements were modelled is discussed in sections 4.7 and 4.8.

4.2. Scene Structure

Before getting more in-depth with the specific functionalities, the application's scene structure is discussed.

As visualized in figure 4.1, on initializing the application the menu scene is booted. The menu is responsible for enabling navigation to the scenes settings and scan. Having both a menu and a settings scene is primarily a design decision. The existence of a menu makes it easier to expand the application for future work, like long-term motivation elements or new guidance scenarios as helping users to scan a room instead of an object. Currently, the menu's main task is to provide access to the settings, so the scan scene does not require an additional button for it. The settings scene is design-focused as well, during existing and future evaluations it can be accessed to quickly change the app's parameters without editing code and also allows the user to modify their experience. More details on configurable parameters in section 4.9.

The image capturing process takes place in the scan scene, it implements the requirements found in chapter 3. Users return to the menu on successful completion or opt-out, back to the

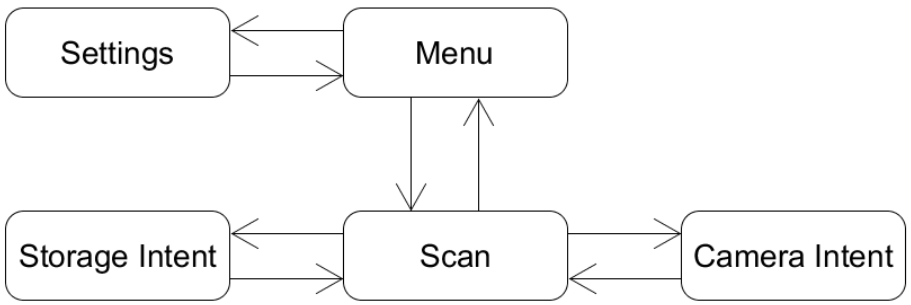


Figure 4.1.: Constellation of the Apps different Scenes

menu, if they want to leave earlier.

The transitions to storage intent and camera intent symbolize intents to camera and storage functions that are already implemented in Android. Intents are Android-specific functions that allow the running activity to request another activity to full fill a task that is either completed in the foreground or asynchronously in the background of the app. Both types then deliver a result on completion notifying the caller of the success of the task. Android intents can also be used to switch the activity permanently similar to changing the current scene in Unity. For this paper's application, they always return a result and return to the scan scene on completion. The storage intent is asynchronous and the camera intent takes over the main thread on execution, the second also sometimes requires the scan scene to be re-calibrated.

The need for re-calibration is faulted to the fact that the scan scene and the camera intent require access to the same resources, like the camera and the main thread, so that the AR tracking in scan occasionally can not re-establish the previous orientation. More on how to tackle this issue in section 4.5. Because Unity does not contain a function to call Android native intents, assets by Süleyman Yasir Kula are used as a connector between Unity code and Android intents, they are called Unity Native Gallery for Android and iOS Plugin [30] and Unity Native Camera Plugin for Android and iOS [31].

To make data, like settings configured in the settings scene, globally accessible to all scenes, globally needed data is saved on the device's hardware with the help of Unity PlayerSettings, they persist as long as the application stays installed on the device.

4.3. Scan Process

The core functionality of the scan scene is the AR-guided scan process. At the beginning of the process the position of the target object has to be specified, this configuration should be maintained during the whole process and needs to be re-established by the user, if technical errors occur.

Once the configuration is completed the process of capturing images begins, it is split up

into three phases as described in section 3.3.

To capture an image from a certain position, its view position marker has to be selected, by hovering the device over its position in the AR scene and clicking n, then at least one picture needs to be taken from that position. Once the user confirms the captured image a new one can be selected. Images need to be confirmed so that users can capture several images at the same position if they want to. This is a design decision and no requirement from chapter 3, this discussion will be, among other things, re-evaluated in chapter 5.

This process repeats itself until, on confirming the last position of the current phase, a congratulation message appears, which on confirmation initiates the next phase. Confirming the last congratulation message returns the user to the menu.

Figure 4.2 depicts the class diagram for the scan scene. It utilizes the state pattern, with the StateInterface and its children acting as states that respond to the UI as their context. The states answer to on-click listeners assigned to UI buttons in the UtilityManager, in this case, the UI is no class in the traditional sense but consists of game objects implemented by Unity. The UtilityManager also acts similar to a facade with the difference that the state system, which resembles the client in this constellation, is not independent but is an aggregation relative to the UtilityManager.

4.4. State System

Focus now shifts onto how the scan process is modeled into the app's user experience, which is managed by the state system, visualized in figure 4.3.

To begin with, the scan scenes UI layout is described. Figure 4.4a shows the scene during its initial state. The user interface contains five buttons, the exit button at the top right is used to return to the menu scene. The help button on the top left opens the help panel. For capturing images, the bottom left button is pressed. The bottom right button is used for picking up viewpoint positions and confirming images. To confirm the configuration of the target object marker or initiate re-configuration, the center-left button needs to be clicked. More information on how the scan target is selected and tracked in section 4.5.

A state diagram for the scanning process can be seen in figure 4.3. The scan scene boots with the ConfigureScanState active, with the UI configuration from figure 4.4a. During this state the position of the target object needs to be configured, to be tracked by AR foundation. This is achieved by moving the camera around the object till the plane, the target is standing on, is detected. On detection, planes are marked for the user, like in figure 4.4b. Once a plane has been detected, clicking anywhere on that plane creates a target marker, as shown in figure 4.4c. If a scan marker was placed at least once during ConfigureScanState, the center-left button in-scripted with "Confirm Configuration" is active. Clicking on this button either returns the app to the previous state or initiates the SelectViewpointState if the current ConfigureScanState is the first one of the scene.

In figure 4.5a can be seen how SelectViewpointState initiates the viewpoint markers when it is called at the beginning of a phase. While this state is active only the buttons help, exit and re-configure scan are active continuously. Exit and help have their standard function,

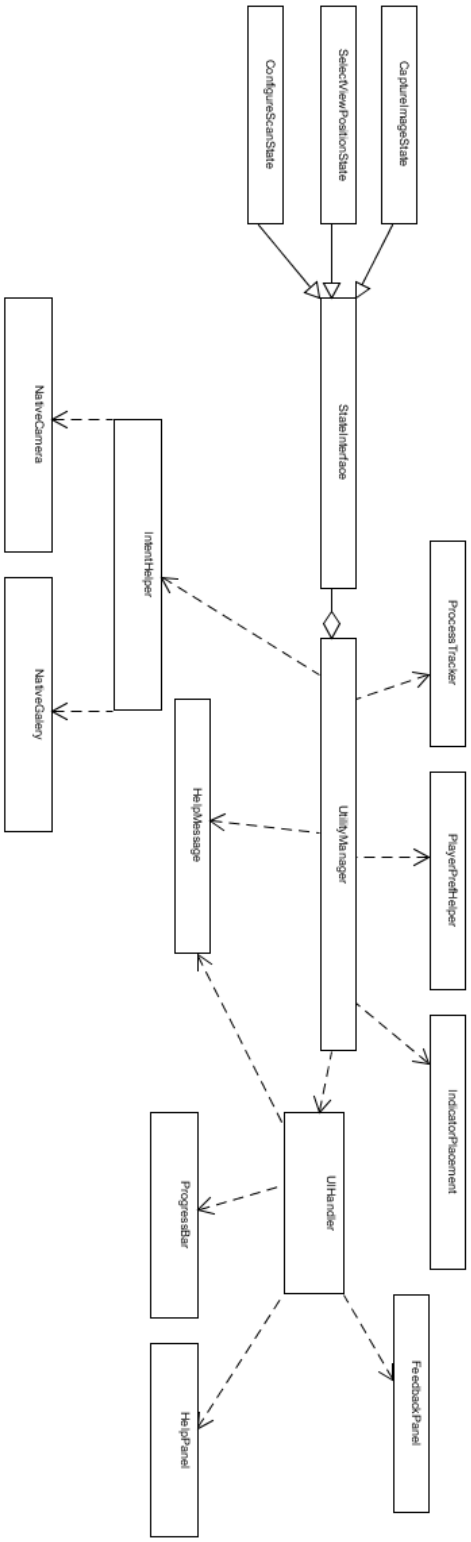


Figure 4.2.: Class Diagram of the Scan Scene

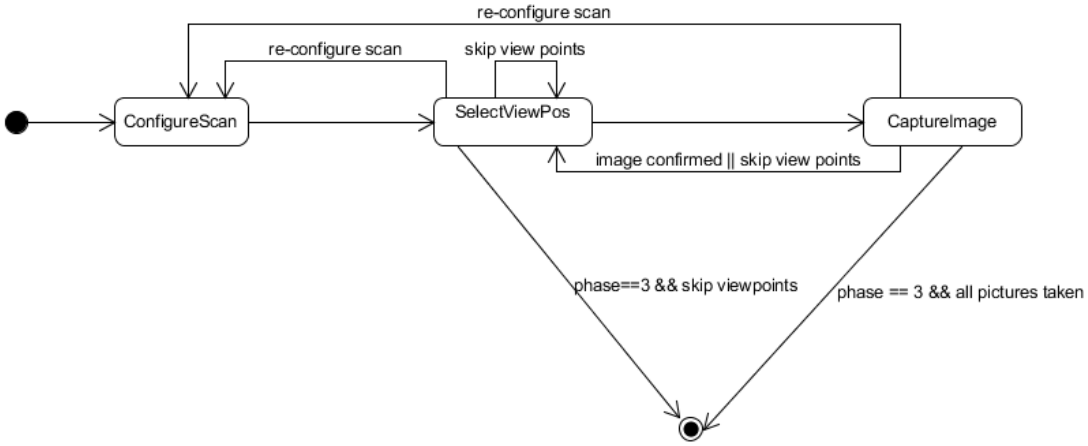
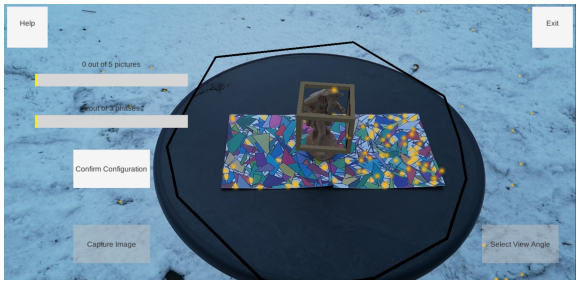


Figure 4.3.: States of the Scan Scene

(a) Scan Scene after Booting

(b) Detected Plane



(c) Target Marker placed

Figure 4.4.: Screenshots of ConfigureScanState

(a) Setup Viewpoint Selection

(b) Hover over Viewpoint

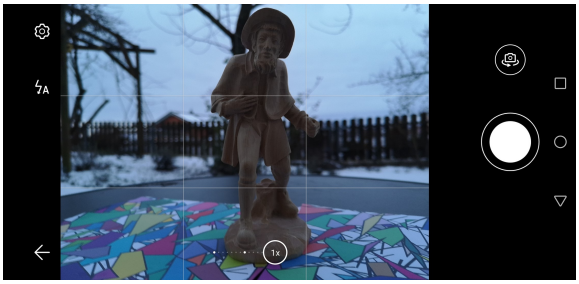
Figure 4.5.: Screenshots of SelectViewpointState

clicking on re-configure scan changes the applications state to ConfigureScanState. While the device is hovering in a view position of choice, the bottom right button is activated, which selects the hovered viewpoint and swaps the state to the CaptureImageState. If points are too close together or the device is not positioned precisely in a viewpoint position, the wrong viewpoint may be selected. Because the app always selects the view position detected first, users need to adapt to this occurrence. Figure 4.5b shows how the position accuracy of augmented objects deteriorates when the device is moved onto first phase viewpoints. Detailed viewpoint distinction was not expected to have a big impact, due to the unstable scene that makes it hard for users to determine if the device is positioned accurately. While for this research further distinction did not seem very reasonable, future approaches can on detection compare the distance of the device from the detected viewpoint and its neighbors to determine the closest one, without increasing run-time complexity. Reducing the selection distance, to improve selection accuracy, can make it hard to select any viewpoint at all.

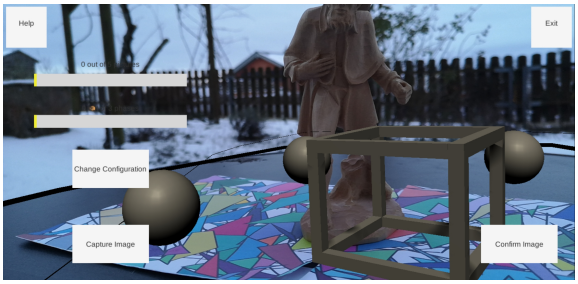
Figure 4.6a shows the CaptureImageState after initiation. Help and exit are active and behave as usual. Specific for this state the capture image button on the bottom left is activated, clicking on it takes a picture by calling an image acquisition activity by intent. In figure 4.6b the UI of the intent can be seen, it works similar to the standard Android camera but has fewer customization options. By pressing the intents bottom center button an image is captured and then is asked to be ticked off to return to the scan scene.

After the user has clicked the capture image button at least once the scan scenes bottom right button is activated. Clicking on it confirms that the user is done taking pictures at this position. If the user reached the end of a phase, they will receive an encouraging notification like in figure 4.6d. During the last phase, confirming the notification will return the user to the menu scene, otherwise, on confirming, the UI will be notified about the progress and the state changes to SelectViewpointState. When the phase is not completed no notification appears, but the progress is updated and the state is updated to the SelectViewpointState.

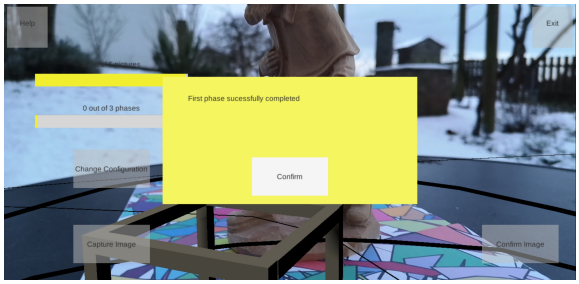
(a) Setup Capture Image



(b) Camera Intent

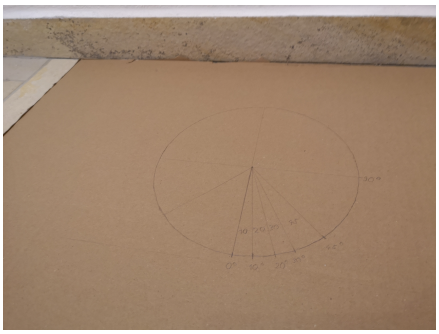


(c) At least one Image captured

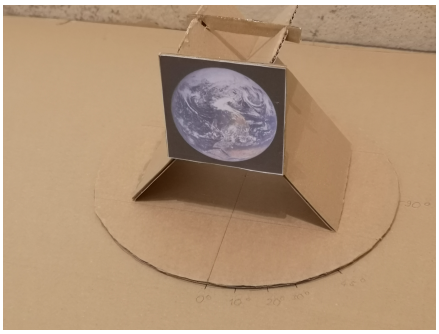


(d) Phase Completion Feedback

Figure 4.6.: Screenshots of CaptureImageState



(a) Base for the Experiment Setup



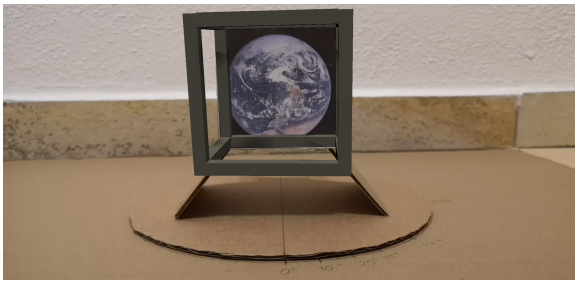
(b) Frame with the attached Tracking Target positioned on the Base

Figure 4.7.: Experiment Setup

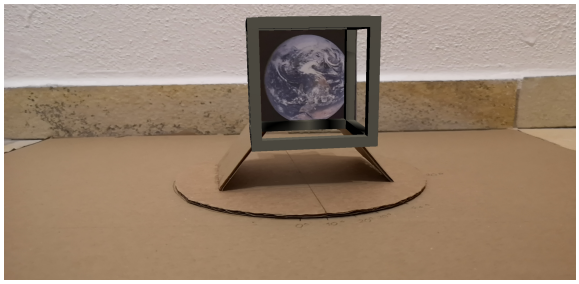
4.5. Object Placement and Tracking

At this point, it is already evident that tracking the position of the object that is being scanned is very important since all the positions users need to take pictures from are initiated and tracked relative to the scanned object that is configured at the beginning of the scan process.

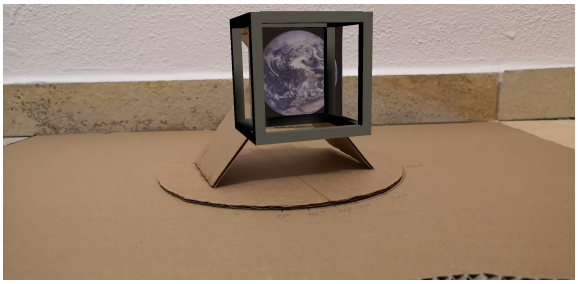
There are many different AR functions that contribute to stabilizing the scene. For this project there were considered two main approaches to design the tracking functionality.



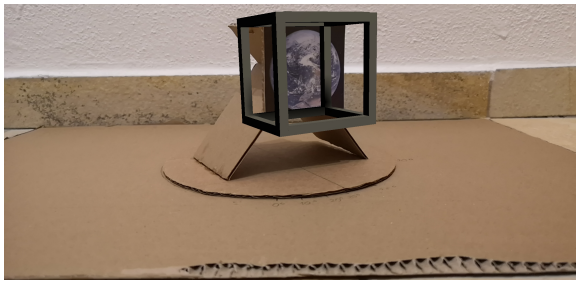
(a) 0 degree Rotation



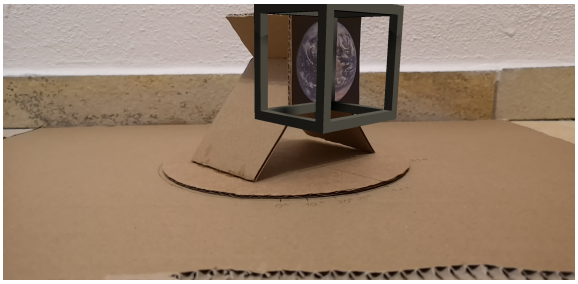
(b) 10 degree Rotation



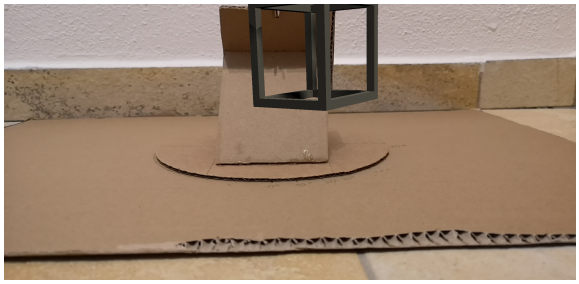
(c) 20 degree Rotation



(d) 30 degree Rotation



(e) 45 degree Rotation



(f) 90 degree Rotation

Figure 4.8.: Experiment Result

Per Frame Image Tracking On the one hand, image tracking sounded promising, the idea was to place several distinguishable images around the target object in a set order that the application then tracks every frame. The target's position is estimated by calculating the center of all used tracked images that are detectable from the current view angle. While this idea might be applicable with future updates of AR Foundation, during prototyping the current version of image tracking did meet this thesis's requirements, more details on findings from prototyping later. No existing projects and guides were found that use image tracking for more than projecting game objects onto different images, in addition to that judging from the documentation Unity image tracking is mainly aimed to be used like that. This thesis instead wants to track the rotation and position of different markers, to use them for further calculations.

The problem faced during prototyping was, that to use image tracking for the goal of this

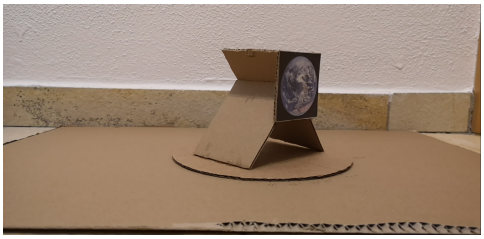


Figure 4.9.: Image Tracking does not recognize Image from initial acute Angle

paper, tracking needs to be stable even when the different images are faced from highly acute angles, because as already described in chapter 3 pictures need to be taken from different heights and angles relative to the object that is being scanned, the smallest one being 0°.

To test the limits of image tracking, a simple application was set up, it only includes basic AR Foundation image tracking, which consists of an AR Tracked Image Manager assigned to the AR Session Origin, the manager has both a Serialized Library and a Tracked Image Prefab assigned. In figure 4.7 the setup of the experiment can be seen, an image marker was attached to a frame that is rotated to simulate view angle derivations of 0°, 10°, 20°, 30°, 45° from a viewer looking perpendicular onto the plane. The square marker's side length attributes 6 centimeters, as tracking target an image from the official Google augmented images for Android best practices was used [32].

The experiment starts with the device's camera facing perpendicular onto the tracking target, which will be referred to as 0° rotation, as depicted in figure 4.8a.

During the experiment position and orientation of the device's camera stays the same, but the frame is rotated counterclockwise around its axis to the extent of the mentioned derivation stages.

Between the stages 0° to 20° shown in figures 4.8a, 4.8b, and 4.8c tracking is quite stable and the placing object is positioned right on the image. In stage 4.8c the placement prefab is misplaced the first time to the extent that it can be easily noticed by the human eye. This trend increases dramatically with a 30° rotation, as shown in figure 4.8d, where the tracking capabilities get continuously less precise. This leads to the conclusion that the image tracking AR Foundation provides is not fit to stabilize a photogrammetry scanning process that also needs to take pictures from highly acute angles. This test assumes that track-able images are placed on the same plane the scan target is placed on.

In figure 4.9 can be seen how the same application reacts when the starting derivation is 45°, which stands in comparison to figure 4.8e. As depicted in figure 4.9, the application did not detect any image, starting from this angle. In combination with the information that, if AR Foundation has detected the marker before, at 0° derivation, it has no problem to recognize the image at 45° and 90°, as shown in figures 4.8e and 4.8f, it can be concluded that there must be a mechanism that does not solely track the image, but also the features surrounding the target, even when the sight of the image was already lost.

This discovered feature is great for a game that wants to place AR objects to provide users with objects of reference, but it is counterproductive for scene stabilization in this thesis

scenario.

Target Configuration with Confirmation On the other hand, there is the option to configure the scan target's position once and then to try to maintain it. To do that, AR Anchors need to be attached to important objects so that AR Foundation knows, it needs to track their positions.

Three options came to mind deciding how to configure the position first of all the target marker could just be placed relative to a certain directional vector from the camera. This can be implemented with ease but is not very stable for a device that has no depth estimation capabilities since there is no real layer of spatial reference.

Second plane tracking is a fitting option for placing a reference object since with a plane manager the application always has a reference to where the object was placed. One problem is that when the planes in the environment of the device deliver barely any level of detail the plain detection might have issues to keep the orientation stable, this can be improved by adding some detail with high detail printouts. With this option, a cube can be easily spawned on a detected plane by clicking on the screen. If the user clicked on a detected plane, a ray cast can determine where the user wanted to spawn the object. An additional issue here is that the plane manager sometimes can not distinguish the plane from the target so that the target is treated as part of the plane, therefore the plane is then tracked poorly.

The third option is using the image tracker. There are strong indicators, this would be an easy-to-operate implementation if the user knows they have to position their device perpendicular to the target plane. This option though is a little more difficult to implement than the second one, which was already finished when the idea for the third option came up, therefore it is only an idea for the future, which then would have to be performance tested against plane detection.

Depth Sensing and Feature Tracking After deciding that plane detection based object placement is expected to be the most fitting method for this thesis purpose, by testing it was found that with only using AR Anchors and Plane Detection the scene often loses orientation when the camera intent is brought to the foreground and also when the surface, the target object stands on, is looked upon from different angles.

To diminish these issues the AR Point Cloud Manager and the AR Occlusion Manager prove successful. The AR Point Cloud Manager creates and tracks feature points to track the environment. The AR Occlusion Manager uses depth-sensing capabilities, if they exist for the device, to estimate the 3D structure of the scene. By providing this additional depth data a normal single camera struggles to provide, the AR Occlusion Manager helps to stabilize the scene.

4.6. Image Capture and Storage

Image capture is a crucial requirement for the guidance concept to function, three realization options were found during the implementation process. The first two can be implemented

with the help of Unity and AR foundation. The game engine can on the one hand directly capture a screenshot and on the other hand access the current camera image from the CPU. The Problem with the first option is, that any objects from the AR Scene that are displayed on the phone will also be in the screenshot because a clear shot on the scanned object and its environment is needed this is not an option.

The second option was disqualified because images produced with this method turned out to be too low in quality to be used for photogrammetry. One reason for this might be that if the photo is just taken from the CPU there is no additional focusing, light-capturing, or post-processing done, taking the photo. The third option is accessing the already implemented Android camera function to capture high-quality images, Android does this via intents, which can not be easily accessed via Unity functions. For this application, the Unity Native Camera Plugin [31] by Süleyman Yasir Kula was used to allow intents to an Android library that provides camera functionalities. This option works best for this paper's needs because it delivers a high image quality. On the downside, it is part of the reason a possibility of reorienting the scene must be implemented since the app often loses focus when it returns to the scan scene after taking a photo.

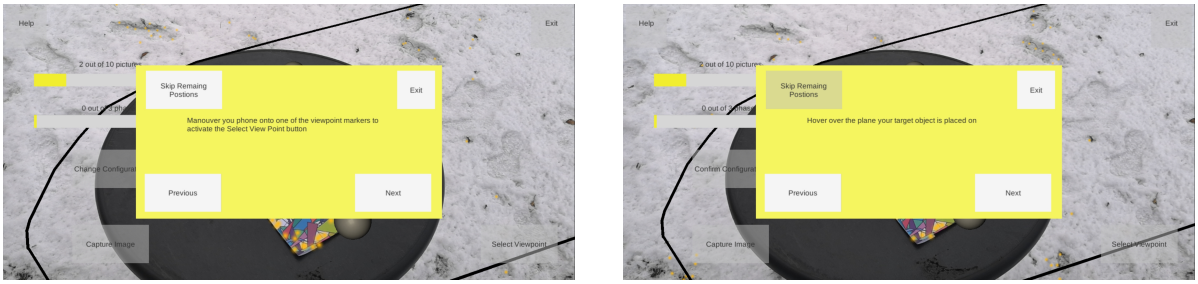
Also in the sense of image quality Unity Native Camera is not optimal, because ISO and shutter angle can not be modified by the user, but are selected automatically. That is a problem because according to experience, collected by conducting scans for Meshroom, the camera sometimes selects big differences in lighting depending on if the camera is facing towards or away from the light source. Importing these images into Meshroom might then cause Meshroom to discard some of the pictures, since their brightness deviation compared to the other pictures is too big. That also happens for images that are facing the sun, because they are too bright with a constant ISO and shutter angle, they do need manual brightness adjustments to not be discarded by the software, but not as much correction as the Unity Native Camera plugin on the Huawei P20 Pro applies.

This issue can be fixed if one would implement a new interface to Android for Unity. This possibility was discovered by looking at the source code of Unity Native Camera, it uses an Android library to provide access to the camera function and an Android library could as-well contain a more complicated camera implementation as long as it can be implemented by Android code.

Storing images is also done with the help of an asset from the Unity Hub called Unity Native Gallery Plugin [30], it saves images via the Androids gallery function, which allows users to easily find images, taken with the app, in their galleries others section. On the Huawei P20 Pro, the stored images can be found under storage/emulated/0/ARApp/<date><time>, for this phone storage/emulated/0 is the standard directory for all applications that capture images besides the standard Android camera.

4.7. Progress Tracking and positive Feedback

The progress of the current scan can be tracked by looking at the progress bars at the top left, they get updated every time a new picture was confirmed. There is one displaying the



(a) Skip Remaining active

(b) Skip Remaining not active

Figure 4.10.: Help

number of images needed to complete the current phase and another for informing about the phase progress, they can be seen in figure 4.6. The user is also congratulated by a feedback message on completing a phase like in figure 4.6d. There is a sound used for audio feedback that plays every time an image was confirmed.

4.8. Help Function

Since the whole scanning process is quite confusing to a person that is completely new to photogrammetry, explaining the app only through UI design is hard. With access to additional information during the scan process, this can be avoided and the user is less likely to get frustrated. In figure 4.10 the help function can be seen, implemented to deliver adequate information depending on what state the user is in at the moment. The help function is used in the scan scene and can be accessed by pressing the help button on the UI. In addition to more information, there is also the possibility to skip the remaining markers of a scene if they are not reachable by the user, by pressing the skip remaining positions button. Users can maneuver through the help messages by pressing the previous and next buttons.

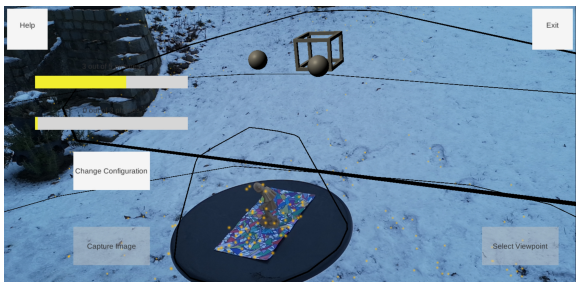
4.9. Configuration of App Settings

The settings scene is used to modify the scan process to the user's needs. Here the width, height, and depth of the scanned object can be set. The number of images taken in each phase can also be customized, this is a design decision to on the one hand allow users to customize their guidance experience as well as to quickly adapt the application for different testing scenarios. The optimal amount of view positions for this guidance application has not been determined yet, but the settings scene can help to test the app with different configurations. Phases one and two are downwards capped at 5 pictures and phase three at 3 pictures. The upward border for phases 1 and 2 is 15 images and 10 for phase 3. The borders have been set naively, to determine how many view positions per phase are optimal different ranges need to be user-tested. For the image capturing process there are two options for the user, first reading the images directly from the CPU and second using the Android Camera App. The

first option can be used if the user only wants to practice the positioning of the photos or when the application needs to be tested with fewer technical errors, the second option is using the Android camera, this is for taking quality pictures but sometimes needs re-configuration between images.

(a) Detection of non existent Planes

(b) Overshooting of detected Planes



(c) Detection Error after leaving the Foreground

Figure 5.1.: Detection Errors

5. Evaluation

To evaluate the application, it has to be determined if its guidance function helps users, new to photogrammetry, to conduct better scans than with just simple verbal instructions. The instructions will be closer specified in subsection 5.2.2. Further, it has to be determined, if user motivation through gamification helps users to stay focused and leads them to completion of the process. On the technical side, there will also be an evaluation on how stable the guidance process is and there is going to be a discussion on how some errors can be solved in future work. Because this thesis was composed during the Covid-19 pandemic, the possibilities for empiric testing are limited, therefore this publication resorts to self-evaluation and sample testing.

5.1. Obstacles found during Self Evaluation

Already during development there appeared many problems regarding the scene orientation. For example Unity sometimes detects planes that do not exist in the real world as shown in

figure 5.1a.

Further in figure 5.1b can be seen how Unity sometimes detects a plane correctly but estimates it's surface wrong, also when the scan scene returns from the camera intent, the scene orientation can be detected differently than at the beginning of the scan. Figure 5.1c shows, how the scan target is shifted up mid-air after the scan scene returned to the foreground.

With a closer look at figure 5.1c it can be observed that the orbs marking the view positions are not positioned around the target object in a clean circle anymore. It is expected this is because each orb contains its own anchor, therefore the Unity AR functions estimate the position for each object individually.

A complete misinterpretation of the spatial bounds like in figures 5.1b and 5.1c only happened every 15-30 times of reconfiguration during development testing. Smaller misplacement errors on the other hand are quite common and usually even happen just by walking around an object. When moving closer to the scan target the augmented objects also get misplaced which can be seen in figure 4.5, this error was taken into concern for adjusting the positions of the viewpoint-markers but has only been tested for the target object that will be used for the evaluation experiment in figure 5.2. The error in figure 4.5 often corrects itself once the user has again increased the device's distance to the target object, sometimes re-configuration is necessary.

Usually re-configuring the scene by again placing the marker onto the right position works quite well since the scene seems to memorize its gyrometric orientation. The viewpoint-markers are positioned relative to the environment's orientation and not relative to the center's orientation. While most of the time remembering the orientation of the scene works well, sometimes Unity still loses orientation, which then means, the scene can not be re-configured properly. At this point, there does not come anything to mind to fix this for Unity.

AR Point Cloud Manager and AR Occlusion Manager The AR Occlusion Manager contributes a lot to the stabilization of the scene. It utilizes depth estimating capabilities of the device if they exist. The Huawei P20 Pro, on which the app was tested, has no committed depth sensor but uses its triple camera for depth estimation, therefore newer devices that use more advanced depth estimation technology might produce even better results. The standard purpose of the AR Occlusion Manager is to make placed objects blend in with their environment. For example, if an augmented object is concealed by a real-life object it will also appear that way in the application.

For this research, it has the nice side effect that it helps the app to distinguish the target object from the plane, which makes the scene more stable. As a reminder, without the occlusion manager Unity sometimes treated objects like images drawn onto the plane instead of 3D objects. This especially manifests itself when the device is moved around the target and the foundation is not ground level. This was discovered only later in development since the first test runs were conducted with the objects placed on the ground, in these scenarios the AR Plane Detection Manager handles itself quite well without the help of the AR Occlusion Manager, but once the object is placed on a table like the one used in figure 3.4a, when

moving the device onto a viewpoint the scene loses orientation without occlusion. With occlusion, the augmented objects only get shifted a little like in figure 4.5.

The AR Point Cloud Manager does not have a big effect on all view angles since as already investigated in chapter 4.5 image detection, and therefore also feature tracking, starts to increasingly struggle from a 20° displacement on. The main idea of using this function is to track features of well distinguishable images to stabilize the scene. This function shines when the scene already lost orientation or when the device is looking onto the target from a beneficial angle because when the device's camera views the target from a steeper angle many feature points are detected and can be tracked to help stabilize or restore the scene. Tho get the maximum value out of this it helps to instantiate the target object viewing it from above.

Image Capture As already mentioned in section 4.6 the camera of the Unity Native Camera Plugin does not provide as many modification options as the Android camera from the test device, in particular manual ISO and shutter modification. Being able to configure these two manually can help increase the image quality since the user then can apply the values in a way that the images taken have a similar brightness, which according to experience, collected during this research, can help Meshroom discard fewer photos.

For future work, a new Android camera plugin for Unity could be implemented, by using Android libraries. Then it could be evaluated if for users without experience in photogrammetry the new implementation performs better since users then need basic knowledge about photography, to apply the according changes to ISO and shutter.

User Frustration On the one hand, making meaningful statements about the user experience without empirical evaluation of user feedback is barely possible. On the other hand that many detection errors and the need for reconfiguration for most viewpoints can quickly lead to frustration is quite obvious. For the future, it needs to be evaluated how big the impact of this frustration is and to what extent the gamification measures help to prevent frustration.

Ideas for the Future As already mentioned in section 5.1 implementing a new camera function might have some potential. In addition to that, as proposed in chapter 4.5 testing scene configurations evolving around one or multiple tracked images to instantiate the scene can be attractive, if reconfiguration is not needed very often. Another argument for testing this approach is that when using image tracking, markers can be instantiated with a constant orientation, which allows the viewpoint-markers to be oriented relative to the target markers, instead of relative to the scene origin. The problem with viewpoints being oriented relative to the scene's origin was that when the application loses track of the scene orientation, the viewpoint-markers get mixed up as well and the scene can not be re-configured anymore.

5.2. Experiment Setup

As already mentioned this research was conducted during the Covid-19 pandemic, therefore there are limited possibilities for empiric evaluation of the application. For this reason, it was decided to, instead of using a questionnaire and empirical evaluation, ask a small sample of testers to conduct scans with and without the application and share their experience during the process. Additional to the tester's impressions the results of the tester's two scans were compared in respect of how many pictures per scan did get discarded by Meshroom and how well the generated meshes turned out. The guided scan is restricted to 25 pictures. As target object, the same wooden object was used that in the introduction also serves as a demonstration for a good scan target seen in figure 1.4.1. Under the target high-detail images, downloaded from [33], were placed to efficiently use feature tracking.

5.2.1. Course of the Experiment

Before the experiment begins testers sign an informed consent form, to confirm that their feedback and scan results can be used for the evaluation. The forms are kept by the supervising faculty, meeting EU data protection regulations. Since a person, completely new to photogrammetry, can not be expected to comprehend how the application is supposed to work and what its purpose is, the test subjects receive verbal instructions at the beginning, which are closer specified in subsection 5.2.2. After the subjects have a basic understanding of what they are testing, the subjects start with the guided scan process and then have to scan the object without additional help using the Android camera function. From the pictures of each scan then a model is generated by Meshroom. It was decided to start with the guidance application to make sure users have a good impression of how they are supposed to capture images when they perform a scan without guidance. In the future, alternating the order of the two ways might help determine, if the guidance application also has an educational effect, in the sense that after using the app users now might also perform better, scanning objects on their own.

5.2.2. Verbal Instructions

Concerning the verbal instructions, users are told to take pictures of the object and try to aim at about 70 percent overlap between the pictures. They are hinted to move in several circles in alternating heights around the target while taking images and that about 70-90 percent of the picture should be covered by the target object. Their target picture count should be at least 25 but they can exceed that if they want to. Also, they get a short introduction into photogrammetry, similar to the one in section 1.4.

5.3. Experiment Results

The Experiment Result is composed of both feedback from the users as well as information extracted from the scan results.

User Feedback The biggest issues during testing were under-performance in intuitiveness and lack of information provided for the user. These issues manifested themselves in many ways. The first problem was that on starting the application there is no introduction to how the application works or what the user needs to do. While via the help button users can get part of this information it is somewhat hidden in the UI. Secondly, the function of re-configuring was a mystery for some users because at the beginning of the scan they of course had no idea that errors are likely to appear during the scan process, or that they need to be fixed by re-configuring. Resulting from this knowledge gap, the fact that the re-configure button is active during the majority of the scanning process, was an origin of the confusion. The third aspect is that exclusively written information turns out to be hard to comprehend without visual examples for some subjects. The fourth and last point of critique was that, for the test subjects, the whole scan structure seemed over-complicated, especially the part where viewpoints need to be picked up and that pictures need to be confirmed after they have been taken. While these functions might be more useful to users that are already familiar with photogrammetry, for the test subjects, which all did not know anything about it before, these processes made it harder to understand the UI. On the positive side, the subjects did not care as much about the few smaller technical issues that occurred during the process since it is not hard to adapt to them as long as the majority of the scene stays stable, as it does most of the time with the help of to the Occlusion Manager and the Point Cloud Manager.

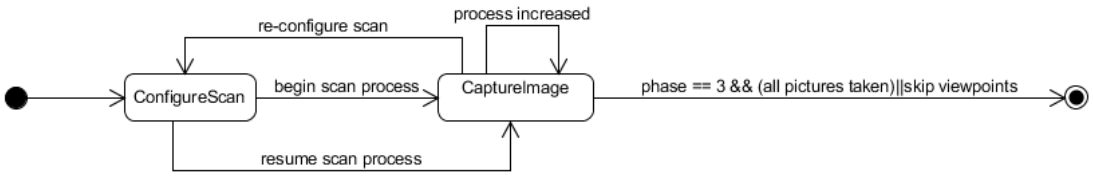


Figure 5.2.: Updated State Diagram

Simplification of the Scan Process In figure 5.2 an updated state diagram, for a possible reconfiguration of the scan process can be seen. The SelectViewpointState would then be removed, viewpoints now will be automatically picked up when a picture is taken and the capture image button is only active when the device is positioned over a viewpoint. Pictures now also do not need to be confirmed, they get confirmed automatically once an image has been successfully taken. This on the one hand makes the scan process easier to understand and makes it possible to discard one of the buttons, which also makes the UI easier to comprehend.

UI Improvements For this research, UI design has not been a big factor, which was a mistake because it has a big impact on how well users can adapt to the UI. Reducing the Amount of UI Elements as mentioned in section 5.3 is one way to make user interaction more

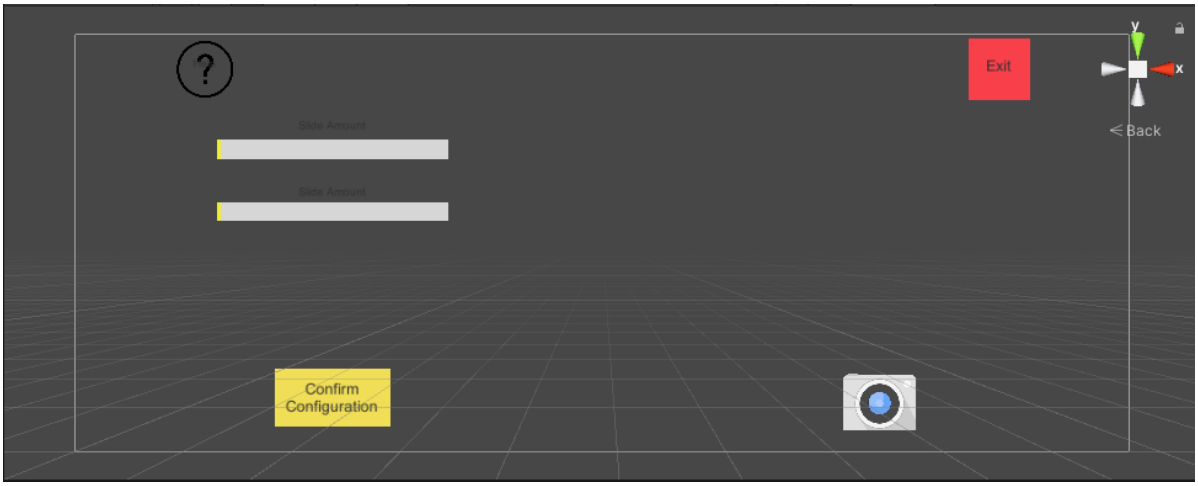
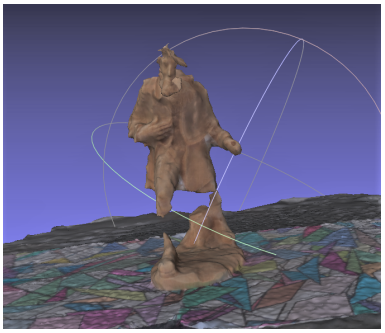


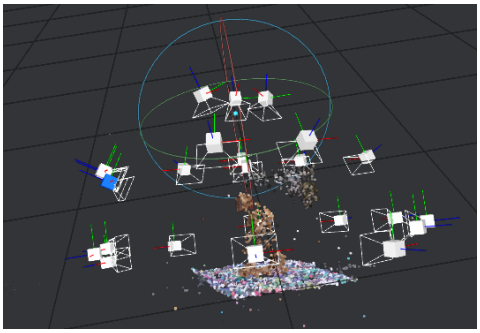
Figure 5.3.: Updated UI

intuitive. According to test subject feedback, also the fact that the differences between the buttons are mainly manifested by inscriptions makes it harder to quickly understand the UI. Adding more intuitive Images for the buttons, in combination with the UI complexity reduction, resulting from the new state system are promising tools to make the application more intuitive. The UI could then be structured similar to figure 5.3. Regarding the help function including tutorial-images needs to be considered and their effect has to be evaluated. Introducing pop-up tutorials to first-time users might also have a positive effect. It also came to mind that having the smallest phase at the end of the process might be counter-productive because then the scan process is expected to take longer than it does. To avoid that, the order of the phases should be reversed in the future. Also pressing the exit button should not instantly lead to leaving the app instead users should have to confirm to abort the scan process, for example by having to tick off a pop-up.

Scan Result Interpretation In total three different users tested the application. Figures 5.4 and 5.5 show the results generated for the images taken by the first tester. For this section images with the caption "Resulting Model" always show the .obj generated by Meshroom imported into Meshlab. Images with the caption "Detected Cameras" show the constellation of camera angles detected by Meshroom. For the first test run guided and unguided results were quite similar, Meshroom was able to detect 100 percent of the images for both scans. Regarding mesh quality, the guided version is a little better for the face quality in the same areas, but both generated figures have several holes and missing areas in their structure. This might be because the scans were conducted during a quite cloudy day and therefore not much light was present in the scanning environment. It has to be noted that for this tester, via the guidance app, only 24 instead of 25 images were taken because they once declined an image they took, which is not explicitly registered by the application since users can take additional pictures at each position. If the improvements proposed in the two previous paragraphs were to be implemented, the app needs to check if the user confirmed the images they captured.

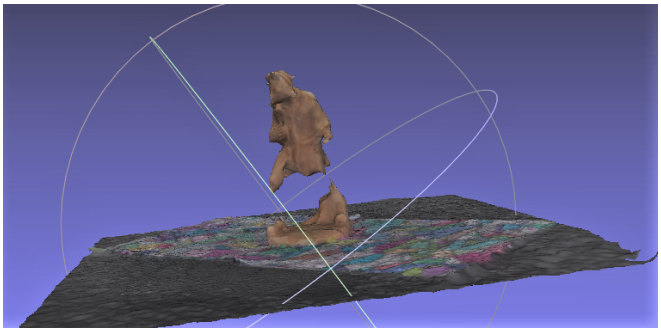


(a) Resulting Model

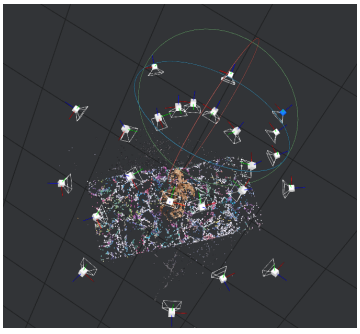


(b) Detected Cameras

Figure 5.4.: First Test Subject with Guidance

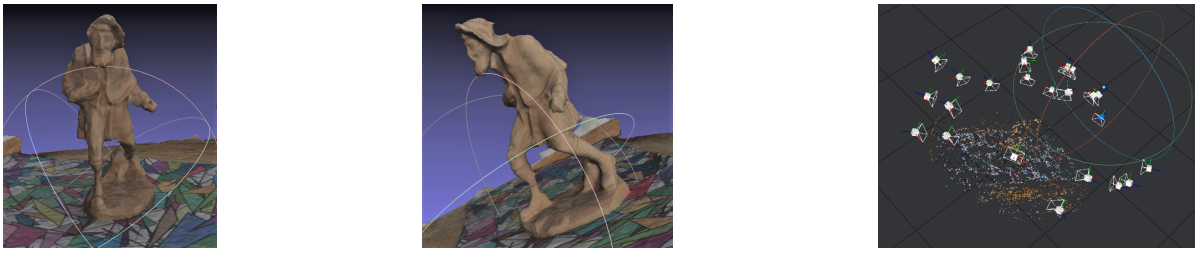


(a) Resulting Model



(b) Detected Cameras

Figure 5.5.: First Test Subject without Guidance



(a) Resulting Model (b) Resulting Model (c) Detected Cameras

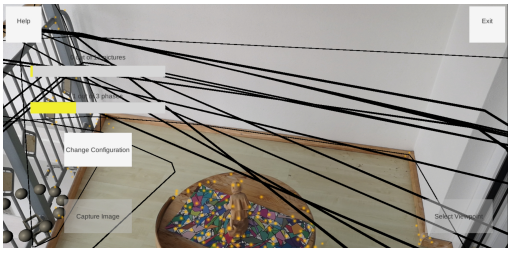
Figure 5.6.: Second Test Subject with Guidance

(a) Resulting Model (b) Detected Cameras

Figure 5.7.: Second Test Subject without Guidance

For the second tester in the guided scan 23 out of 25 cameras were detected and 20 out of 25 for the unguided scan. The results can be seen in figures 5.6 and 5.7. Even considering that for the unguided scan about three images might have been discarded due to the wrong focus of the camera, the guided scan again performed at least as well in this regard. The pictures of the generated 3D models show that the guided scan produced a way better model than the one without guidance. This can be traced back to the fact that, as shown in figure 5.7, the camera angles are not distributed well around the object and that several gaps are not covered at all. Also during this test, another issue unraveled, which occurs for plane tracking. As displayed in figure 5.8 Unity detected so many planes that it was not able to keep track of them in Update() so that the app froze, this also later happened with the third subject. It needs to be taken into account though that for these test scans both horizontal and vertical planes were tracked, which fueled this issue further. Tracking both horizontal and vertical planes was considered a bug that got fixed after the test was completed. The potential for described error still exists, but it is now less likely to occur.

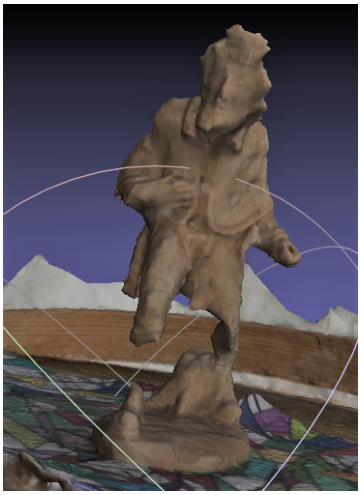
For the third test subject, the guided scan again outperformed. For both 25 out of 25 images were detected, but, as shown in figures 5.9 and 5.10, the model for the guided scan has better quality and fewer blank spots. Regarding the app an additional issue was discovered here, viewpoints are only select-able, when the user hovers over a viewpoint with their device,



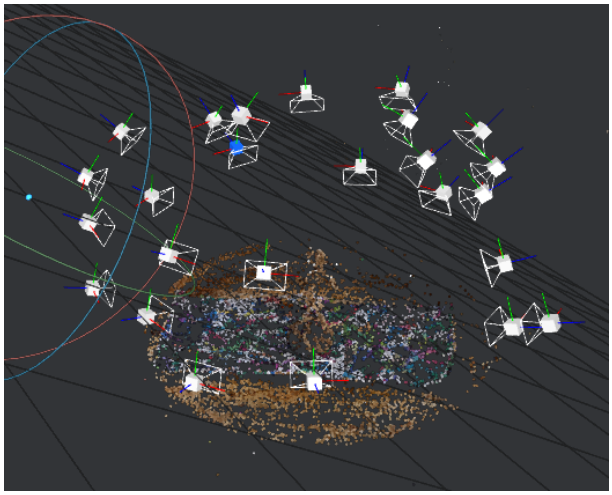
(a) Unity 3D detects large Amount of Planes

(b) Unity 3D tracks too many Planes and freezes

Figure 5.8.: Errors during second User Test



(a) Resulting Model



(b) Detected Cameras

Figure 5.9.: Third Test Subject with Guidance

viewpoints are detected by checking if the distance of the device to the viewpoint's center is close enough. The radius for these tests was set to 0.20f which should in theory equal to about 20 centimeters, the user ended up taking about three images at a very similar position, as already predicted in chapter 4.4. This value was set when depth and feature tracking were not used to stabilize the scene already, now it might be possible to tune the detection radius down to about 0.15f. If determining the viewpoint closest instead of taking the first one that fits the criteria is better performing, can be determined in the future. In the scan without guidance, the user tended to only take images from high positions, therefore the unguided scan under-performed.

5.4. Blueprint for Future User Studies

User studies are a promising tool to further evaluate the application on an empirical level, this is needed to be able to make scientific statements. These studies are especially useful

(a) Resulting Model (b) Detected Cameras

Figure 5.10.: Third Test Subject without Guidance

for examining how users perceive things like the UI and the structure of a process [34]. Proving that the application performs well in regards to the thesis goals and showing that the improvements proposed in this chapter perform better than previous results are both hypothesizes that can be evaluated in the future. To make this easier, a blueprint for the first hypothesis was already created and can be found in this thesis repository [29].

Survey Structure The survey focuses on motivational requirements from chapter 3 and is structured into three parts. The first one asks questions about the effect of elements implemented to motivate users. The second tries to determine weak points in the UI that potentially lead to user frustration. Because also technical errors can demotivate users or even make completing a scan impossible the third part asks users about technical errors that occurred during testing. Requirements regarding scan quality performance can not be evaluated by user studies, but material needed to mathematical evaluate them can be collected during user studies as well, by saving images the testers captured.

Question Types Mainly Likert scale type questions are used to get a weighted impression on the user's opinions [35]. Testers are also asked to elaborate on already suspected weak points, to help solve them. For cases, where new ideas for improvement are needed, like the UI usability, users are asked to provide written feedback on weak points they spotted. This question type is not empirical, as long as users do not detect similar issues, but they can be used to improve future user studies to cover these issues empirically as well.

6. Conclusion

This chapter first sheds light on this thesis's contributions, then provides an outlook onto future work, and eventually sums up the topics discussed in this thesis.

6.1. Contribution

In this section contributions of this paper are discussed.

Scan Process and Gamification For this thesis, a guidance concept for the photogrammetry scan process and a gamification-based motivation concept were created. They are applied to a Unity 3D application so that it meets the requirements of the thesis goal from chapter 1.2. The implementation shows that AR can be applied to this thesis purpose goal, with the current state of AR technologies for mobile phones.

Target Placement and Tracking Concept Different Placement and tracking approaches were discussed in section 4.5. Placing a marker in the scene and tracking it with the help of the AR Occlusion Manager, the Plane Manager, and the AR Point Cloud Manager was decided to be the most promising approach for this thesis. In the implementation, the target-marker is placed by clicking on a detected plane that was visualized by the application. Another viable approach would be using image tracking for initial instantiation.

Evaluation and User Feedback The application's performance was evaluated and several potential improvement ideas for future research were found. First evaluations indicated that creating scans with the help of the guidance application leads to results at least as good as the ones without guidance. In two sample-tests, the app outperformed verifiably. To improve this result several proposals for future adaptions were made in chapter 5.

6.2. Future Work

In this section future research possibilities, with this paper as its foundation, are elaborated.

Long Term Motivation For future work it can be investigated, if long term motivation is relevant for the field, also in the sense that a guidance application could also have an educational effect on its users, so that on frequently using the application, they do not need guidance anymore do perform quality photogrammetry scans.

UI Improvement The UI needs to be reworked in sense of usability and inactivity. To do that, future research can start with adapting ideas from section 5.3 for a new application that is then tested against the version from this paper. The focus of refactoring should lie with making the UI less complex and providing more easily understandable information when the user is confronted with a new concept. Making the UI simpler is also connected to reducing the complexity of the scan process, which section 5.3 describes in more detail.

Using Image Tracking for Scene Instantiation Exploring new methods for scene stabilization and initialization is key to improving the usability and stability of the guidance system. One possible route to explore in this field is using image tracking to setup the target-marker or area. This procedure has the potential to increase scene stability and intuitiveness of the UI. After implementation, the new approach needs to be user tested against the existing one.

Improving the Guidance Concept The evaluation showed that there is room for improvement for the guidance concept. The placement of viewpoints was determined straightforwardly for this thesis. In future work, on the one hand, setting additional focus for areas that contain a high degree of detail can improve guidance performance, on the other hand re-evaluating the general positioning of viewpoints starting with the degrees the pictures are taken from should be studied further by using bigger scale user testing.

Empirical User Studies For this research, no statistical evaluation was possible, due to Covid-19 restrictions. In the future, it is essential to evaluate the gamification and photogrammetry concept on a larger scale and also test ideas for improvement from chapter 5 against the existing version to validate if this research is moving in the right direction.

Android native VS. Unity During development, it has been found that according to the author's subjective opinion, Googles Android native Hello AR project that uses Open GL for game-engine related functions, appears to outperform Unity implementations transferred to Android [36], regarding scene stability and plane detection accuracy. According to this experience, it is possible to improve the guidance performance, by using platform-specific implementations. This can be further investigated in the future.

6.3. Summary

The goal of this thesis was to elaborate if an AR application is capable of improving photogrammetry scan results of scans conducted by users new to photogrammetry, by guiding them along with an optimal scan process reference. Gamification elements were used to keep users motivated during the process. These requirements were implemented in Unity 3D using AR Foundation and AR Core. The result was then evaluated with the help of experience collected during development and sample testing. In general, the app proved successful in guiding users to conduct better scans but the user interaction concept has still room for improvement and technical errors are sometimes negatively influencing the user experience.

A. Apendix

A.1. Tools and Environments Version Numbers

Tools and Environments used for development were Unity 2020.2.1f1, AR Foundation 4.1.1, AR Core XR Plugin 4.1.1, Unity Native Camera for Android and iOS 1.2.7, Unity Native Gallery for Android and iOS 1.6.2, and Universal Rendering Pipeline 10.2.2. The device all scans and tests were performed on is a Huawei P20 Pro with Android version 10 and EMUI version 10.0.0.

For photogrammetry calculations, Meshroom 2020.1.1 was used, for displaying created meshes MeshLab 2020.12.

Images for tracking were taken from the sources [32] and [33].

A.2. Mobile Application Instructions

The mobile application can be installed by installing the .apk file found in this thesis's repository [29]. Sounds and images used for the project are found in the folder Additional-Material/ForeignAssetsAndSounds with sources provided in Sources.txt.

A.3. Export Results and Import to Mesh Room

To export the images captured from your device they can be found in the gallery in the others section. The Pictures can be imported into Meshroom by dragging them into the Drop Image Files/ Folders section. By pressing on Start the Mesh and Textures are calculated. On completion, the final results are saved where your current Meshroom project is located. From your Meshroom projects select the folder MeshroomCanche and then Texturing in that folder only one other folder should exist which contains mesh and texture data. To view the mesh open Mehslab and Select File>ImportMesh. With the file browser that opens go to the folder that contains your created mesh as described before and import the file "texturedMesh.obj".

A.4. Viewpoint Placement Implementation

To place the viewpoints a naive approach was used, the basic concept for that is already discussed in chapter 3. Regarding the implementation, a post on Stack Overflow was used as a reference and adapted to the left-handed coordinate system of Unity 3D [37]. In figure A.1 you can see the implementation that can be found in the Unity project folder under

```
2 references
public Vector3 GetAngleVector(float circularAngle, float planeAngleRad){
    Vector3 result = new Vector3();
    result.x = Mathf.Cos(circularAngle)*Mathf.Cos(planeAngleRad);
    result.z = Mathf.Sin(circularAngle)*Mathf.Cos(planeAngleRad);
    result.y = Mathf.Sin(planeAngleRad);
    return result;
}
```

Figure A.1.: Code Snippet of View Point Placement

/Assets/Scripts/Functionality/IndicatorPlacement.cs. It takes the angle relative to the plane the target is placed on and an angle on a circle drawn onto the plane around the target. They are used to calculate a direction vector which is then used to calculate the viewpoints positions by scaling the resulting vector and adding it to the target's center. The scaling of the vector and viewpoints is done relative to the target objects scaling.

List of Figures

Bibliography

[1] G. A. Hurd and J. Cameron. *The Terminator*. 1984.

[2] K. Feige and S. Black. *Iron Man 3*. 2013.

[3] F. C. Manager. *Neu in Google Maps: Augmented Reality mit Live View*. Ed. by Google. URL: https://support.google.com/maps/thread/12938628?hl=de (visited on 01/24/2021).

[4] B. Katz, F. Dramas, G. Parseihian, O. Gutierrez, S. Kammoun, A. Brilhault, L. Brunet, M. Gallay, B. Oriola, M. Auvray, P. Truillet, M. Denis, S. Thorpe, and C. Jouffrais. "NAVIG: augmented reality guidance system for the visually impaired". In: *Technology and Disability* 24 (Nov. 2012), pp. 163–178. DOI: 10.1007/s10055-012-0213-6.

[5] F. Pankratz and G. Klinker. "AR4AR Based on ARVIDA Reference Architecture: Application Demonstration". In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. 2016, pp. 348–349. DOI: 10.1109/ISMAR-Adjunct.2016.0115.

[6] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui. "Mobile Augmented Reality Survey: From Where We Are to Where We Go". In: *IEEE Access* 5 (2017), pp. 6917–6950. DOI: 10.1109/ACCESS.2017.2698164.

[7] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang. "What Will 5G Be?" In: *IEEE Journal on Selected Areas in Communications* 32.6 (2014), pp. 1065–1082. DOI: 10.1109/JSAC.2014.2328098.

[8] K. Kraus. *Geometry from Images and Laser Scans*. Photogrammetry. Karl Kraus, 2004. ISBN: 9781680152616.

[9] A. Lipman. *5 Industrial Photogrammetry Applications*. Ed. by Industry Today. URL: https://industrytoday.com/5-industrial-photogrammetry-applications/ (visited on 12/24/2020).

[10] T. Collins. *A Look Into Photogrammetry and Video Games*. Ed. by medium.com. URL: https://medium.com/@homicidalnacho/a-look-into-photogrammetry-and-video-games-71d602f51c31 (visited on 12/24/2020).

[11] K. McDonald. *Assassin's Creed creators pledge €500,000 to Notre Dame*. Ed. by The Guardian. URL: https://www.theguardian.com/games/2019/apr/17/assassins-creed-creators-pledge-500000-notre-dame-restoration (visited on 12/24/2020).

[12] A. Greenberg. *3D models help preserve landmarks like Notre Dame*. Ed. by Nova Next. URL: https://www.pbs.org/wgbh/nova/article/3d-model-point-cloud-notre-dame/ (visited on 12/24/2020).

[13] J. Quimby. *Explore museums in a new way with Tango*. Ed. by Google. URL: https://blog.google/products/google-ar-vr/explore-museums-new-way-tango/ (visited on 12/25/2020).

[14] V. J. Beck. "Using Photogrammetry and Augmented Reality for Virtual Reconstruction of Ancient Statues". In: Dec. 2017. URL: https://wiki.tum.de/display/infar/MA%3A+Using+Photogrammetry+and+Augmented+Reality+for+Virtual+Reconstruction+of+Ancient+Statues.

[15] G. Kumparak. *Ubiquity6's Display.land is part 3D scanner, part social network*. Ed. by Techcrunch. URL: https://techcrunch.com/2019/11/18/ubiquity6s-display-land-is-part-3d-scanner-part-social-network/ (visited on 01/05/2021).

[16] C. Shortcuts. *3D Scanning from your Smart Phone for free!* Ed. by Youtube. URL: https://www.youtube.com/watch?v=jyfktD1CtLM&feature=emb_logo (visited on 12/25/2020).

[17] D. Team. *display.land*. Ed. by Ubiquity6 Inc. URL: https://display.land/sunset (visited on 12/25/2020).

[18] Apple. *Scanning and Detecting 3D Objects*. Ed. by Apple. URL: https://developer.apple.com/documentation/arkit/scanning_and_detecting_3d_objects (visited on 12/19/2020).

[19] K. Ben. *A Guide to Photogrammetry Photography*. Ed. by Journalism 360. URL: https://journalists.org/resources/a-guide-to-photogrammetry-photography/ (visited on 12/14/2020).

[20] "Photogrammetry Workflow". In: Unity, July 2017. URL: https://unity3d.com/files/solutions/photogrammetry/Unity-Photogrammetry-Workflow_2017-07_v2.pdf?_ga=2.2025459.314872352.1607958942-690372025.1592319625.

[21] F. Photogrammetry. *A BEGINNERS INTRODUCTION to taking pictures for photogrammetry*. URL: https://www.youtube.com/watch?v=O2ZGNPbDhPI (visited on 12/15/2020).

[22] P. Josef. *Photogrammetry 2 – 3D scanning with just PHONE/CAMERA simpler, better than ever!* URL: https://www.youtube.com/watch?v=1D0EhSi-vvc&feature=emb_logo (visited on 12/15/2020).

[23] L. Networks. *Research Report Shows How Much Time We Spend Gaming*. URL: https://de.limelight.com/resources/white-paper/state-of-online-gaming-2019/#spend (visited on 01/24/2021).

[24] L. Legner. "User Perception of Game Elements in Duolingo". In: Oct. 2020.

[25] V. Berger, L. Miesler, and J. Hari. "The Potential of Gamification in Changing Consumer Behaviour Towards a More Sustainable Nutrition Behaviour". In: July 2014.

[26] Cambridge. *Gamification*. Ed. by Cambridge Dictionary. URL: https://dictionary.cambridge.org/dictionary/english/gamification (visited on 12/01/2020).

[27] L. Legner, C. Eghtebas, and G. Klinker. "Persuasive Mobile Game Mechanics For User Retention". In: Oct. 2019, pp. 493–500. ISBN: 978-1-4503-6871-1. DOI: 10.1145/3341215.3356261.

[28] R. Zajonc. "Mere Exposure: A Gateway to the Subliminal". In: *Current Directions in Psychological Science* 10.6 (2001), pp. 224–228. DOI: 10.1111/1467-8721.00154. eprint: https://doi.org/10.1111/1467-8721.00154. URL: https://doi.org/10.1111/1467-8721.00154.

[29] S. Stolz. *BAWS20 Stolz Simon Gamified Photogrammetry*. URL: https://gitlab.lrz.de/IN-FAR/Thesis-Projects/ba-ws20-stolz-simon-gamified-photogrammetry (visited on 02/02/2021).

[30] S. Y. Kula. *Unity Native Gallery Plugin*. https://github.com/yasirkula/UnityNativeGallery. Oct. 2020.

[31] S. Y. Kula. *Unity Native Camera Plugin*. https://github.com/yasirkula/UnityNativeCamera. Oct. 2020.

[32] Google. *augmented-images-earth.jpg*. Ed. by Google. URL: https://developers.google.com/ar/images/augmented-images-earth.jpg (visited on 12/26/2020).

[33] Brosvision. *AUGMENTED REALITY MARKER GENERATOR by Brosvision*. URL: https://www.brosvision.com/ar-marker-generator/ (visited on 02/11/2021).

[34] S. Streng. *Mensch-Maschine-Interaktion User Study Design*. University Lecture at Ludwig-Maximilian-Universität. Sommersemester 2009. URL: https://www.medien.ifi.lmu.de/lehre/ss09/mmi1/20090430_UserStudyDesign.pdf.

[35] J. Robinson. "Likert Scale". In: *Encyclopedia of Quality of Life and Well-Being Research*. Ed. by A. C. Michalos. Dordrecht: Springer Netherlands, 2014, pp. 3620–3621. ISBN: 978-94-007-0753-5. DOI: 10.1007/978-94-007-0753-5_1654. URL: https://doi.org/10.1007/978-94-007-0753-5_1654.

[36] G. Developers. *Depth API quickstart for Android*. URL: https://developers.google.com/ar/develop/java/depth/quickstart (visited on 02/10/2021).

[37] T. Valeev. *3D Vector defined by 2 angles*. URL: https://stackoverflow.com/questions/30011741/3d-vector-defined-by-2-angles (visited on 02/02/2021).