DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Authoring Tool for Industrial Augmented Reality

Kağan Batuker

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Authoring Tool for Industrial Augmented Reality

Autorensystem für Industrielle Erweiterte Realität

Author:	Kağan Batuker
Supervisor:	Prof. Gudrun Klinker
Advisor:	Jan Heitger, Sandro Weber
Submission Date:	15.05.2020

I confirm that this master's thesis in Informatics: Games Engineering is my own work and I have documented all sources and material used.

Munich, 15.05.2020 Kağan Batuker

Acknowledgments

I would like to thank all the people who made this project possible. First of all I would like to thank Jan Heitger and the company RE'FLEKT for the opportunity to write my thesis in RE'FLEKT and under their supvervision. Second, I thank Professor Gudrun Klinker and the Chair of Computer Aided Medical Procedures and Augmented Reality for accepting my thesis proposal. Additionally I would like to thank both Jan Heitger and Sandro Weber for their supervision and support throughout the thesis duration. Finally I would like to thank all the people below who participated in the user study and helped with various aspects of my work within six months (in alphabetical order):

Steven Babb
Martin Karg
Moritz Kartheuser
Mehdi Khadir
Özge Kilimci
Andreas Lay
Kadir Tandogan Tilki

Abstract

Augmented reality is a key technology that is going to leave it's mark on the world in the next decade. Since it's inception in the 19th century and it's growth in popularity in the last decade, the technology has provided new use cases in the latest industrial revolution as well, creating the term industrial augmented reality.

Some of the main hurdles for the growth of this sector are scalability and lack of supporting tools for faster development and content creation. Although the technology is advancing, supporting authoring tools still have a long way to go.

In this master's thesis, the goal is to conduct a broad search to find the best practices for constructing industrial authoring tools for augmented reality, and then implement the findings into a creation of a new authoring tool, fit for industrial needs. The thesis will be supported by the industrial AR solutions company RE'FLEKT, and the development of the application will be influenced and evaluated by different departments of the company. The final aim will be to integrate this authoring tool to the RE'FLEKT ONE production pipeline in the future, which is their biggest product.

Kurzfassung

Augmented Reality, oder Erweiterte Realität, ist eine Schlüsseltechnologie, die in den nächsten zehn Jahren ihre Spuren in der Welt hinterlassen wird. Seit ihrer Einführung im 19. Jahrhundert und ihrer zunehmenden Beliebtheit in dem letzten Jahrzehnt hat die Technologie auch in der letzten industriellen Revolution neue Anwendungsfälle bereitgestellt und den Begriff İndustrielle Erweiterte Realität"geschaffen.

Einige der Haupthindernisse für das Wachstum dieses Sektors sind Skalierbarkeit und das Fehlen unterstützender Tools für eine schnellere Entwicklung und Erstellung von Inhalten. Obwohl die Technologie weiterentwickelt wird, ist die Unterstützung von Authoring-Tools noch weit entfernt.

Ziel dieser Masterarbeit ist es, eine umfassende Suche nach besten Anwendungsoptionen für die Erstellung industrieller Authoring-Tools für Erweiterte Realität durchzuführen und die Ergebnisse dann in die Erstellung eines neuen Authoring-Tools umzusetzen, das den industriellen Anforderungen entspricht. Die Arbeit wird vom industriellen AR-Lösungsunternehmen RE'FLEKT unterstützt und die Entwicklung des Programms wird von verschiedenen Abteilungen des Unternehmens beeinflusst und bewertet. Das endgültige Ziel wird sein, dieses Authoring-Tool in die Zukunft in die RE'FLEKT ONE-Produktionspipeline zu integrieren, das größte Produkt des Unternehmens.

Contents

1 Introduction

An introduction to the subject of this thesis is made in this chapter. The structure of the thesis will be explained first, followed by the problem description, motivation and goals of the thesis.

1.1 Structure

The second chapter will consist of general concepts and the research of related work relevant to the thesis' goals. The next chapter focuses on the company RE'FLEKT, their product RE'FLEKT ONE and the pipeline revolving around that product, including the authoring tool developed in this thesis. The fourth chapter outlines the implementation work which is comprised of the proposed work, the approach taken in the implementation and the detailed explanation of every step and component of the authoring tool application. The fifth chapter discloses an user study in which participants from the company evaluate the usability and performance of the authoring tool. Next is the discussion and future work for the application and the pipeline. Finally, a conclusion is drawn in the final chapter.

1.2 Problem Description

We have seen the term augmented reality slowly become a part of our lives in the last decade. With the recent technological advancements in tracking, rendering and hardware devices, augmented reality has benefited from it the most. Since the beginning of the 21st century, its popularity is on the rise and it has already broken out of the Gartner Hype Cycle for Emerging Technologies in 2019[1] . Although this is the case, augmented reality has found more success in industrial and commercial use cases than entertainment ones. Today, it can be said that industrial needs are still the driving force in the supply and demand cycle, seeing that the industrial AR market is steadily growing and hardware manufacturers like Microsoft and Vuzix are prioritizing business needs meanwhile Magic Leap One, first launched with a focus on entertainment, is getting its software adjusted for enterprise needs[2] .

Producing and maintaining sustainable software in augmented reality, whether it be for personal or enterprise needs, requires platforms that are scalable in the first place. In industrial augmented reality, this means that the AR application will be used by numerous people and include tens of different use cases. The application should be easy and flexible enough to realize any needs for any specific company.

Authoring, the process of content creation, plays a crucial role in this matter. One of the reasons augmented reality is on the surge for enterprise use cases is that it is one of the

milestones of Industry 4.0, allowing digitalization and better visualization of any data that was in manuals before. As a matter of fact, authoring for augmented reality is the process of transforming the data on the manual to a digital content that fits the use cases and shows the feasibility and the strength of the technology.

Even though the technological limit has been breached, support and adoption for augmented reality applications has fallen short, and one of the reasons is the lack of support on the side of content creation. Research shows that widespread adoption can occur only with the improvement of tools that are used to author augmented reality content [3].

1.3 Motivation

Is is clear that the research and development for authoring tools has not caught up to the main research aspects of augmented reality yet. Despite the fact that augmented reality is a proven concept, creating content for it was always a problem throughout the technology's history. Even now, there is not a lot of research that goes into how an authoring tool for augmented reality should be designed.

The motivation for this thesis is to do a research about the essential concepts of an industrial augmented reality authoring and try to come up with an authoring tool that enables content creators to effectively author anything using augmented reality interaction techniques and ideas. This means building a standard authoring tool for 3D content creation and modifying it with elements that supplement augmented reality technologies.

The company RE'FLEKT is supporting this research in which this authoring tool will be created using their resources and expertise. Their flagship product, RE'FLEKT ONE, is a network of programs that also uses authoring tools, but these are not fine tuned to create augmented reality content. Instead, they are standard 3D content creation programs. This thesis will also be a challenge on their part by finding ways to conceptualize and integrate a lightweight tool to their pipeline.

1.4 Goals of This Thesis

The goals of this thesis can be divided into two. The first part revolves around the research question of what is defined as good guidelines in authoring for augmented reality. An extensive research is undergone to find the definition for AR authoring, which methods and entities are used to enhance AR elements and the alignment of company products with contemporary examples.

The second part is to use the information gained in the previous question and apply this knowledge into creating an authoring tool for industrial purposes, which is RE'FLEKT's specialty. This part includes determining the needs of the company, how the tool could fit in the current workflow and how sustainable this approach would be.

For the implementation, a patented concept from the company, Augmented Photos, will be integrated into the product, along with other internal modules. Other approaches such as external asset integration and addition of augmented reality elements will be handled as well.

After the implementation, the authoring tool will be evaluated by people from R&D (Research and Development) and UX (User Experience) departments and a conclusion will be drawn for the current state of the program, along with possibilities for future iterations.

2 Concepts and Related Work

This chapter takes a deep dive into topics that are essential for the whole thesis and displays the thorough investigation about industrial augmented reality, existing authoring tool implementations and evaluations of augmented reality interactions within authoring tools and applications.

2.1 Critical Research Topics

Starting with the thesis, the first goal was to investigate what defines a efficient authoring process in terms of augmented reality. This leads into researching the previous authoring tools that were developed and looking into their design and their key points. The problems these applications tried to solve and how they managed to propose solutions are going to be looked into. Another question is how much research was done about AR authoring in the past and how it holds up for today. Lastly, common findings from these research are going to be evaluated against RE'FLEKT's own implementation to see whether they align or not.

2.2 Industrial Augmented Reality

First, an introduction to the augmented reality and its use in the industry will be made, explaining the research topics thoroughly.

2.2.1 Augmented Reality

Augmented reality is "a technology that superimposes a computer-generated image on an user's view of the real world, thus providing a composite view"[4]. While this is the definition from the dictionary, the most common accepted explanation comes from Azuma in 1997, characterizing augmented reality systems with three components[5]:

1. Combines real and virtual
2. Interactive in real time
3. Registered in 3D

Nowadays, we can observe many applications that enhance the real world with virtual content in real time, and it is not restricted to visual enhancement. AR can also play on our auditory, haptic and olfactory senses, although it is not that common.

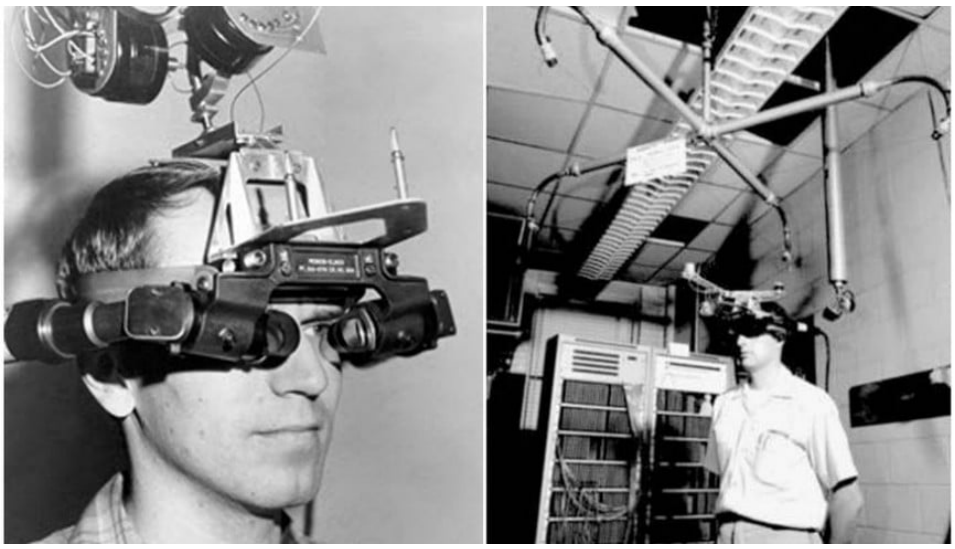

Figure 2.1: Ivan Sutherland with the first head mounted display, "The Sword of Damocles". Source: [6]

History of Augmented Reality

Augmented reality came into being when Ivan Sutherland created the first head mounted display (HMD) in 1968. In 1974 the laboratory called "The Videoplace" was opened, in which video projectors and cameras created silhouettes on screen and provided an interactive experience for the users.

In 1990, a Boeing researcher, Tom Caudell, came up with the term "Augmented Reality". Louis Rosenburg, a researcher in the USAF Armstrong's Research Lab, created 'Virtual Fixtures' in 1992, which was one of the first completely functional augmented reality systems. The system allowed military personnel to virtually control and guide machinery to perform tasks like training their US Air Force pilots. In 1994, the first Augmented Reality theater production was created as the performers danced with and around real and virtual objects.[6]

With the beginning of the 21st century, augmented reality started to emerge as a supplement in entertainment sector. NFL games started to include the "yellow line", an AR line projected onto the field. Researchers in USA began working on a Battle Augmented Reality System for naval battle training. Hirokazu Konta created ARToolKit, an open source software library to help developers create augmented reality programs. ARToolKit is still widely used today. Paper based media used AR for the first time in 2009, using a book's cover as a tracking target[7]s.

With the 2010s, AR began to become a part of everyday life. Volkswagen created MARTA (Mobile Augmented Reality Technical Assistance) to help technicians with step-by-step technical instructions, thus paving the way for the usage of augmented reality in industrial settings. In 2014, Google announced Google Glass, a glass improved with augmented reality to immerse its users, starting the wave of wearable AR. Microsoft and Magic Leap followed suit, as the former published its own wearable hardware HoloLens in 2016 and the latter

Figure 2.2: HoloLens2 used in an industrial setting. Source: [10]

starting its own AR hardware called Magic Leap One in 2014 after a 50M$ funding round. Meanwhile, Pokemon GO, a mobile game using augmented reality, caught the world by storm by reaching around 70 million users in 2017[8].

In the last years, we saw tech giants also invest in augmented reality and produce supporting software like ARKit (Apple), ARCore (Google), MixedRealityToolKit (Microsoft) and we saw the release of HoloLens 2, along with general AR support for mobile devices like smartphones and tablets. In 2019, over 4.1 billion dollars were invested into AR and VR combined [9] and this trend is keen to be going upwards in the next decade.

Tracking Methods

The backbone of the augmented reality technology comes from computer vision algorithms that ensure consistent overlapping of virtual objects onto real ones while counting in the effects of lighting, occlusion etc. This process is called tracking. There are different types of tracking that are being utilized in the industry. Below are the most used tracking methods both in personal and enterprise use:

- **Marker Tracking:** Marker tracking is one of the first methods used in augmented reality for input recognition. Black and white pictures with distinct sharp edges, called markers, are placed on a surface in the real world. Computer vision algorithms take the camera feed as input and process the marker through a series of steps including gradient extraction, classification of black and white parts and then a transfer to binary input. The position and orientation of the marker is then used as a starting point to render virtual content correctly on top of the real world. Today, advancements in processing technology allows us to use any image to act as a marker and render the result in real time. Although reliable, marker tracking is somewhat limited in flexibility

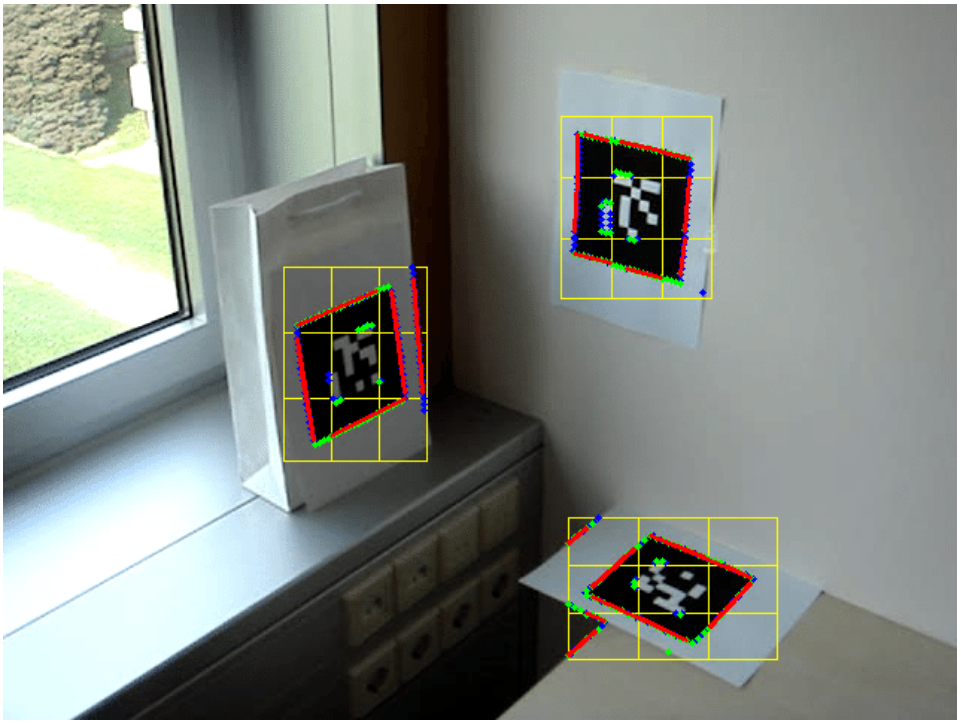

Figure 2.3: Marker tracking. Source: [12]

and must rely to an external 2D object.

- **SLAM Tracking:** This method derives from the algorithm called Simultaneous Localization and Mapping (SLAM), which was used for robotics applications to estimate real time position and orientation of robots [11]. This discovery was then applied for AR, in which local feature points were found and then mapped to previous frames to find the position and orientation information of objects in the scene. The unknown environment is virtualized with 3D point mapping and can be used with augmented reality.
- **Feature Tracking:** Also called Natural Feature Tracking, this technique detects some characteristic points from the image in real-time by the AR system and the virtual objects' pose is calculated based on these points. Popular applications include edge based tracking, where edges are extracted from the image as input. Another method is to track from the 3D object shape and match the object shape to the edges extracted from the camera feed.

2.2.2 Industrial Augmented Reality

The definition of industrial augmented reality (short: IAR) is using augmented reality technology to support an industrial process [14]. The first attempts at this were carried out by Boeing in the early 1990s [15]. Several attempts were made in incorporating AR into the industry, with the most significant one called KARMA, a maintenance helper for printers [16].

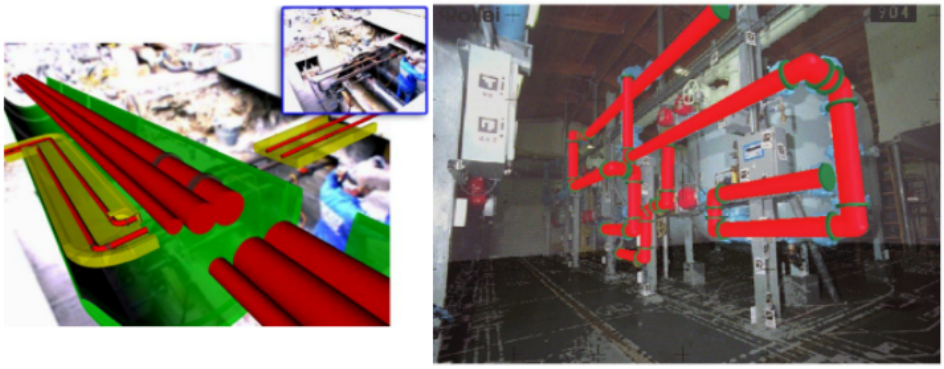

Figure 2.4: Examples of 3D static superimposition on the real environment through model based tracking. Source: [3] and [13]

With the growing interest and possibilities for augmented reality, research projects started to form around the globe in the first decade of the 21st century. The most ambitious and scaled project was ARVIKA, found with the aim of research and implementation of AR into German industries [17]. This led into another project called ARTESAS, specializing in the development of AR for automotive and aerospace maintenance [18]. There was also the collaboration between US and European companies named under the banner STAR [19].

With the 2010s, industrial augmented reality applications took a leap from being research projects and prototypes to being feasible implementations and scaling solutions. Accessibility of AR with handheld devices and the release of new HMD devices like HoloLens and Vuzix opened up new opportunities for companies to increase the speed and efficiency in their pipelines. Research states that AR is presented as a powerful tool to improve flexibility and process efficiency in the last decade [11].

Volkswagen's MARTA was followed by projects from Airbus, MAN and Siemens [20][21]. New companies only specializing in AR also emerged, with PTC founding Vuforia for their AR applications and others competitors like RE'FLEKT and DIOTA being founded. These and other companies provide all-in-one solutions for enterprises looking to modify their productions.

Application Areas

There are different applications for augmented reality when it comes to industrial pipelines and product life cycles. These are listed below.

Assembly

Assembling is the process of putting together several separate components in order to create a functional product [22]. This can happen multiple times in a product's life cycle. Even though most assemblies are now automated, some still need to be assisted by human operators. Using augmented reality to support assembly guidance, training and simulation has concrete benefits like increased concentration, efficiency and reduced human error.

Maintenance and Repair
Maintenance is crucial for a product's life cycle as it means the controlling of the product at regular intervals. By repair operations it is meant the actions of restoring the functional properties of a device. Similar to assembly procedures, AR helps maintenance workers to concentrate and have access to instant information about the object in question and cut time spent in looking into the paper manual. The same goes for repair tasks as well. Both maintenance and repair play an important role in a product's lifecycle as shorter time spent on these tasks means a longer uptime for the machinery.
Training
Many industries and processes require intensive training for operators before they begin working, as these processes are complex and need precision. The technicians need to be trained in in multiple areas to learn the intrinsics of the process, which can be challenging. Augmented reality can ease this process by visualizing the training content on the machinery in real time [23].

This method also creates an immersive learning experience, which leads to overall better results from the training session. Training can be enhanced with visual, audio and haptic elements to interact with virtual objects to achieve this.
Quality control and Inspection
Inspections are aimed at assessing the current status of the product and analyse the causality of deterioration and functional degradation [24]. Operators can apply augmented reality to an inspection process to fasten the procedure while automating the process and providing an error-prune method. Supplemented by computer vision, machine learning or IoT (Internet of Things) algorithms, AR applications have the ability inspect the product with efficiency and accuracy at the same time.
Monitoring and Visualization
The use of AR have been proposed to interact with scientific data in shared environments. An augmented environment can be set up collaboratively to share visualizations with multiple users in real time. Combined with IoT technology, any data (temperature, uptime, latency etc.) can be accessed by the operators and any other operation can be carried out simultaneously.
Remote Collaboration
The last topic that profits from augmented reality is particularly a new one: remote collaboration. During a downtime for a machinery, remote collaboration programs are used to connect technicians to maintenance experts. Augmented reality can also be applied to this interaction, in which both parties can use AR annotations to draw on the live camera feed and communicate using various virtual elements.

Industrial Augmented Reality Today

There are some research trying to determine the scope, impact and characteristics of the IAR market for today. One of those reports written last year points out that while this sector shows promise, AR is still not present in most industries [25]. A systematic literature review is carried out for papers and patents between 2012 and 2018. Research concludes that proposed solutions have majorly focused on manual assembly processes, which are a part of

maintenance, inspection and training activities. Palmarini [26] suggests the same trend in his foundings, citing assembly, maintenance, repair and inspection as four general topics. Two process innovations that realize these goals divide into two: Firstly, the creation of generalized solutions that can adapt themselves according to the user experience and company. Secondly, the use of remote assistance with AR that reduces the need of having experts on-site and, consequently, cost and time. The latter actually makes up 69% of industrial AR patents.

Another survey carried out in 2018 by Palmarini [26] looked into literature review between 1997 and 2017. Results say that there is a high fragmentation among hardware, software and AR solutions which leads to high complexity problems when selecting and developing AR systems.

The Lisboan research [25] again indicates that 52% of these implementations are applied in general industries (not developed for a specific industry). This number is followed by automotive, mechanical, electronics and aerospace industries. Palmarini's research [26] is similar in the field of maintenance: Mechanical maintenance is 29%, plant maintenance covers 21%, with aviation and consumer technology both covering 8% each.

The most hardware used for industrial augmented reality is found out to be head mounted displays with 40% in [25]. Tablets and smartphones get a combined share of 31% while monitors follow with 23%. It also becomes clear that HMDs have an upward trend even now, with research showing that 54% of AR applications in 2018 utilize this kind of hardware and 86% of patents in the same year utilize HMDs. However, as useful as it may seem, they still have issues like narrow field of view and weight. An additional research points out that reducing weight and improving autonomy for HMDs will increase their usability [27].

From identified HMDs, just three are currently commercially available in 2018, which suggests that this hardware technology is still in early stages of evolution. Palmarini [26] lists HMD (30%), handheld devices (27%) and desktop PC (27%) as the most common devices utilized in development of an AR application.

Another aspect to look at is the used tracking methods. Among the reviewed literature from Cardoso et. al.[25], 57% of them utilized marker tracking, while only 16% used 3D recognition algorithms. 2D recognition algorithms (wherein an environment bi-dimensional feature is used for identification) followed with 13%. 10% of tracking methods were not specified to fit a category. Palmarini et. al. [26] came to a similar conclusion, putting marker tracking as the most used method in his research with 52%. Feature based tracking and model based tracking followed with 19% each, while other solutions represented the rest with 10%.

In terms of software development, 63% of applications declared the use of an AR framework [25]. All implementations run on Android, iOS, or the Windows platform. Unity 3D and Vuforia, one of the most popular AR SDKs, were responsible for most of the development among all prototypes. OpenCV libraries combined with 3D programs like Blender are also widely adopted. For development, the other survey reported usage of mid-low level languages (44%), libraries like OpenCV (23%), SDKs (14%) and lastly game engines (10%) [26].

Analyzing the primary benefits, 90% of the literature speaks of 6 main benefits, which can be seen in figure 2.5. According to primary studies; operational time decrease, production flexibility improvement, production quality learnability increase are factors that contribute to

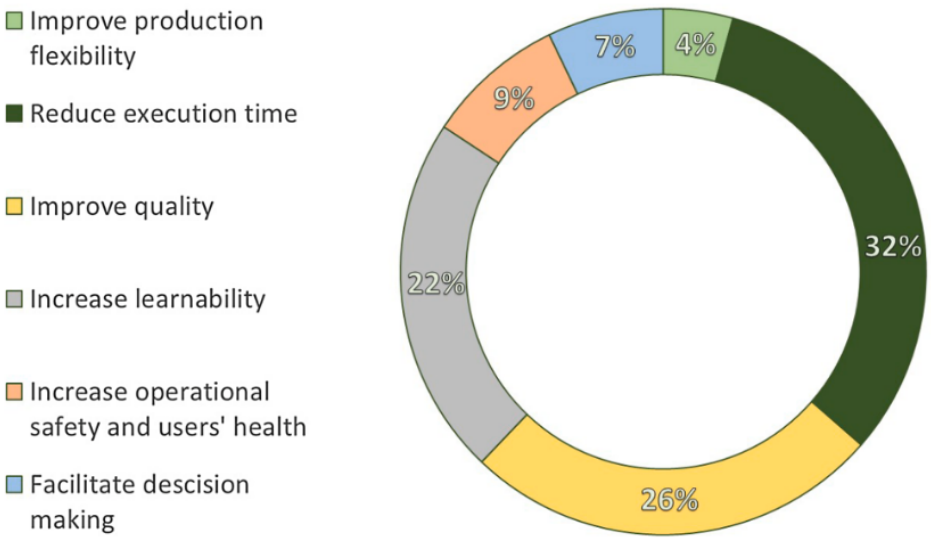

Figure 2.5: Percentage of main benefits of industrial AR use on production environment. Source: [25]

costs reduction. User safety and mental workload reduction are also other benefits.

Main challenges of applying augmented reality were listed as hardware (25%), projection quality, accuracy and interaction (25%), tracking methods (24%), user's health and acceptance (17%) and lastly, development complexity (9%). According to Syberfeldt[28], users need to have their hands and visual field free and thus using handheld devices and projectors actually limit productivity. One other hardware and health concern is that HMDs can cause fatigue and dizziness after long usage times.

Other concerns were about marker tracking not being too much viable in production environments, due to marker size and occlusion. Markerless methods were also investigated but found not mature enough to be used in bigger scales[29], along with virtual object instabilities[30].

The main category discussed was the quality and accuracy of the virtual elements. Around 22% of primary studies declared that accuracy plays a big part in product development and efficiency. Challenges faced in this area are rendering algorithm optimizations and tracking objects not having distinct features and virtual objects not being detailed enough.

Future Work for Industrial Augmented Reality

Although augmented reality has been around for more than 2 decades, literature review suggests that the industry has not developed to its full potential yet. "Even though the advantages of the AR technology have been proven at an academic level, improvements are required in several fields in order to provide a robust, reliable and flexible solution for practical implementation", says Palmarini [26]. Navab adds to this statement, indicating that commercial IAR applications should be robust and reproducible with accuracy, user friendly

and scalable beyond simple prototypes [3].

The same paper from Palmarini declares that there are three main areas that needs to develop in order for the sector to mature: Hardware, Tracking algorithms and User-AR interaction, authoring solutions, content management tools and visualization.

HMDs and handheld devices are reported as promising, although HMDs have ergonomic issues and hand held devices have limited field of view. It has been debated whether long-term use of AR hardware causes stress and strain on the user and whether solutions without AR might be better fits [31]. Future trends will hopefully include strong implementations of sensors and haptic devices, along with better ergonomics [26].

Future development in tracking will also hopefully overcome issues with lighting, occlusion and material mismatches. In 2018, Sarga et. al. published a paper in which they share that markerless tracking techniques still require maturing and robust tracking scenarios are limited to test environments [32].

The research concludes with need of improvement on the topic of user-AR interaction. It is declared that in order to implement AR in the industry, the AR system has to be easy to maintain and modify. A separate research for the understanding of best visualization techniques needs to be carried out. Lastly, Palmarini talks about the adaptability of future AR systems, incorporating artificial intelligence elements and interacting with the operator in real time through IoT technology.

In 2016, Wang et. al. concluded that areas of improvement are in tracking, registration, knowledge representation and context-aware systems [33]. Nee et. al. concluded that the accuracy required in tracking and registration in AR applications are the main limitations in applicability in manufacturing [34]. In another case study by Fast-Berglund et. al, they concluded that limitations in hardware and application design are the leading cause of underutilization of AR technology in manufacturing [35].

2.3 Authoring for (Industrial) Augmented Reality

This section will go over the concept of an authoring tool, discussion of known authoring tools, ending with literature review about authoring tools for augmented reality.

2.3.1 Authoring Tool

An authoring tool is a software program that enables users to create content using text, media, and interactions. It can also be called an authoring system.

An authoring program has pre-programmed elements included for the development of interactive multimedia software. This enables the user to interact with the program without requiring the programming knowledge, among with any other low-level knowledge about augmented reality or networking etc.

An authoring system usually includes an authoring language, a programming language built specifically for the authoring system. Examples of this include DocBook, DITA and XML

editors. These languages enable a high code abstraction and use without any programming code. High-level visual tools are commonly used for this matter.

The output of the authoring tool can then be used as the content in the augmented reality application. There are also different output sources to encapsulate the data.

Authoring can also be used to combine the created content and the tracking configuration, while adding increased layers for networking and security applications along the pipeline.

2.3.2 Authoring Tools in Industrial Augmented Reality

Authoring in augmented reality plays a big part in a successful application. It has been proven time and time again to be an essential part of the pipeline. It encapsulates content creation: The more flexible and usable an authoring tool, the better and faster the virtual content. Ease of use and speed also resurfaces in another aspect: Scaling. When AR companies are selling their all-in-one products, they also need to support it with various multimedia content and examples, so that their workflows can be replicated on the seller side with efficiency. This is where a need for a reliable and simple authoring comes in.

The literature review also shows the importance of an authoring tool for any AR enterprise workflow. Santos argues that authoring tools is one of the AR content-related issues, together with instruction design and content management tools [36]. Langlotz et. al. points to authoring tools as the AR solution to the widely known contents problem [37]. Bae et. al. lists it as one of the two key components of mobile AR development along with pose estimation [38].

The need for better authoring tools seems to be unanimously agreed upon by multiple acticles: "In order to implement AR in industry, the AR system has to be easy to maintain and modify. New authoring solutions and contents management tools are required", Engelke points out [39]. "Reconfigurability of future AR system is a must, hence the flexibility of authoring tools must improve", says Lamberti [40]. One of the conclusions from project ARVIKA was that "the creation of AR systems has not yet outgrown the stage of engineering-intensive lab solutions and science and industry must enforce the design of AR-assisting authoring systems" [17]. Another research from Knöpfle et. al. in 2005 [41] states that the existence of AR authoring tools is crucial for the success of the technology in the targeted application domain.

Palmarini also put authoring tools under the scope in his survey from 2018 [26]. He found 64% of authoring solutions to be manuals. This means that content, whether it be 2D or 3D, is manually generated. There is no authoring tool present in these solutions, just adaptations to the tracking configuration in terms of content. Manual authoring is expensive due to the amount of time and skills required in performing it. The professional skills involved int the process are listed to be programming, modeling and animation [42].

Some researchers have tried to generalize this process and provide more practical solutions. Klinker et. al. first came up with an idea when investigating augmented reality interactions for power plants. They generalized 2D virtual annotations into a set of primitive tasks: highlight, label, display information (text), clear information, edit information, set compass and hide/show [43]. Alvarez et. al. utilized annotations by combining them with SLAM

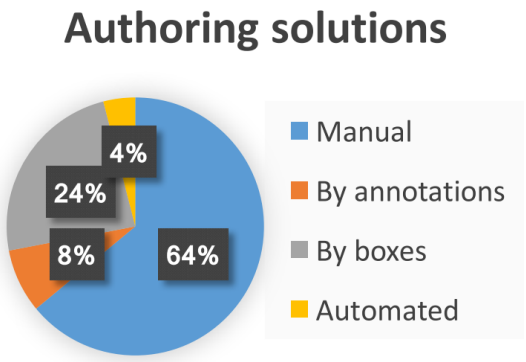

Figure 2.6: Distribution of authoring solutions by Palmarini et. al. Source: [26]

tracking for better registration for the real world [44]. Jung came up with a web-based annotation system in which the author can attach these annotations onto a 3D model [45]. Annotation techniques cover 8% of proposed authoring solutions.

Palmarini categorizes another 24% of authoring tool solutions by the concept of "boxes". These techniques aim to build AR solutions without the need for a deep computer programming knowledge. The process of achieving this is to generalize elements in an AR application and encapsulate them into entities called "boxes". Havard came up with the concept and his generalization led to the model based on these concepts below [46]:

- Entity, smallest part of a system
- External Entity, smallest part outside of the AR system like tools
- Actions, activities like push and pull
- Maintenance, a set of actions
- Operation, a list of maintenance operations

This abstraction should theoretically leads to the author only configuring the "boxes" and their relations with one another, providing a black box implementation. An example of this can be found in figure 2.7 There were other research that aimed for similar goals. Fiorentino et. al. designed a similar authoring tool with abstractions [47]. So did Zhu when he tried to create a context-aware AR system, even though most of the information was in text format [48]. Another attempt for a context-aware AR system was made by Erkoyuncu when he developed and tested an authoring solution which uses real-time data from sensors to help building new authoring procedures [49].

The remaining 4% is for automated authoring. An ambitious task, this was only applied to assembly procedures at that time. These procedures are automatically created from CAD models and assembly planning theory. Alvarez was successful to extract the correct procedure from every step possibility using CAD model constraints and assembly planning module [44].

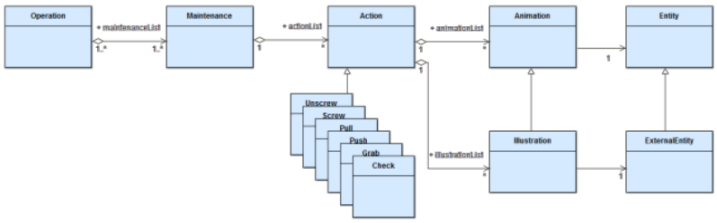

Figure 2.7: UML diagram that represents an example from Havard's boxing model. Source: [46]

2.3.3 Additional Literature Review for Authoring Tools

There were other literature work on the subject that flew under the radar for surveys, but they still play an important role in showing how authoring tools have evolved over the years.

Some of the earlier authoring tools that have seen the light of day in the 2000s are AMIRE [50], DART [51], and ComposAR [52]. These were simple authoring tools and their biggest accomplishment was to include a GUI for authoring AR content and use markup languages like XML and later X3D, languages that were going to be standard for AR authoring applications in the future.

There was another authoring tool made by Zauner in 2003, which was based on AMIRE and utilised another tool called PowerSpace [53]. PowerSpace mimics a 2D presentation program as the basis of the created content. Combining these two, Haringer was able to create a tool in which 2D elements are easily created and added in an AR context [54].

Another effort was made by Knöpfle et. al. in 2005 where he abstracted the work flow into templates (operations), work steps and tasks. Then he visualized this work flow using GUI windows [41]. Platonov et. al. implemented a similar proof-of-concept system featuring markerless tracking [55].

In 2006, Hampshire came up with a graph editor to do authoring. He also listed 4 components to achieve successful authoring: Tracking, marker manipulation, video capture and 3D output components [56].

In 2010 Wang et. al. constructed an authoring tool for examination purposes using the AR library ARToolKit [57]. The tool supported 2D text, 3D content editing, and finger-based interactions. This toolkit would then become one of the most-used API throughout the world for creating AR applications.

Gimeno et. al. created their own AR system (including authoring) and named it SUGAR in 2011. They utilized the depth sensors of Kinect to work around the occlusion problem that is widely encountered in AR [58].

Engelke et. al. proposed a generalized solution for the authoring problem in 2015. His implementation was based on web standards like XML, CSS and X3D. A task is divided and categorized into sub tasks and each of them can be supported with virtual annotations, along with supplementary text information. There is also the ability to freeze the camera image with tracking intact, allowing for more flexibility [39].

In the same year, Petersen et. al. tried a very ambitious technique. Wanting to eliminate

Figure 2.8: The content from the desktop application is viewed as holograms in the HoloLens. Source: [61]

the 3D content creation process and building up from their previous research in 2012 [59], they designed an authoring tool where the video of a maintenance operation is extracted into step-by-step guidelines: A version of automated authoring previously discussed in section 2.3.2. This way, an experienced operator would create a guide through the application without any authoring interference. Visualizations included annotations, assessing overlays, animations and feedback of the operating task by comparing hand positions in the created guide and the camera feed [60].

2.3.4 Contemporary Authoring Tool Implementations

Today, with industrial augmented reality reaching maturity, there are lot of tools that are widely used by software developers and enterprises alike. Learning from the past successes and mistakes, these applications are more or less the standard right now when it comes to creating 3D content for augmented reality.

Dynamics 365 Guides

Dynamics 365 Guides is a mixed-reality application for Microsoft HoloLens and HoloLens2. It enables operators to create step-by-step mixed reality guidances for any task, without requiring programming or 3D skills [61]. The content is authored on two separate steps. In the first step, PC authoring tool is used. Here the author can create hologram cards, each representing a step in the whole procedure. The author enhances these steps with images, video and 3D holograms. Authors can use the default library provided by Microsoft or import their own 3D data.

Authors need to create a structure in cards, step-by-step instructions to be carried out by the operator. These steps can have sub steps as well, each indicated with another instruction card. To support these instructions, video, audio, image and text material are present (See Figure 2.8).

Figure 2.9: In the second step of authoring, author needs to tether instruction cards to real objects. Source: [61]

After creating the guide on a desktop, the user uses HoloLens or HoloLens2 to connect the instruction cards and holograms from the desktop guide to the physical work space. A marker has to be printed out from desktop application and placed on the tracked object, so that tracking can be initialized. Next, holograms like instruction cards and 3D objects are placed in the environment. Then these holograms are "tethered" to the responsible objects. This means a dotted augmented reality line is created between the instructions and the object in reference (see figure 2.9).

After this step, the authoring process is complete. What Microsoft offers in addition is to track the user progress for each step by data analysis. After some usage, the author has access to information like step completion time, performance and effectiveness of a task. This way, they can gather feedback and improve the authoring by doing modifications.

It has to be noted that to use Dynamics 365 Guides, the user has to own HoloLens / HoloLens2 and agree to a monthly subscription.

Vuforia Studio

Vuforia Studio is the authoring tool provided by PTC, the owner company of Vuforia, for its users and customers. The content creating tool for Vuforia is a web based editor, similar to authoring tools used in e-learning. Everyone can use it, as long as they create an account.

As can be seen in figure 2.10, the screen is divided into 5 major parts. This is because both the tracking and content authoring is handled in this program.

On the most left column, the user has access to the scene and the authoring setup. User can configure themes, styles and other information that can vary between enterprises.

On the next column, tracking methods and 3D objects are listed. The author can change the tracking method from here, in addition to adding 3D objects and "widgets", 2D objects that provide real time information with IoT technology. The connection is done with "Things"

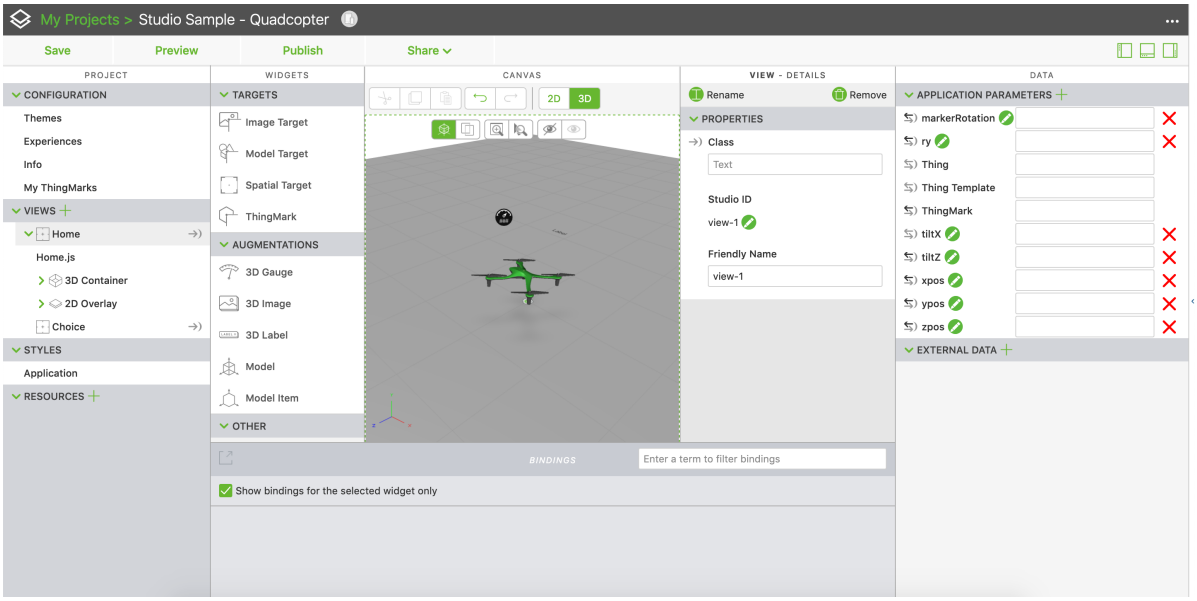

Figure 2.10: The interface of Vuforia Studio. Source: [62]

and ThingMarks", the name given to IoT abstractions by PTC. On the center of the screen is the 3D view of the scene, called the canvas. Objects in the scene have 6 degrees of freedom for movement and can be controlled by a mouse or the 3D widgets that are on the corners of this screen. The scene camera has the same flexibility, along with zoom options.

There is also an option to switch between 3D and 2D authoring. While in 3D authoring the augmented reality scene can be set up, user interface for hand held devices is configured in the 2D option.

On the right side of the canvas the user can access the general information panel for the selected object. Its attributes like name, size and type can be manipulated using this windows. At the last column the author can tweak the application parameters like scene position, orientation or parameters connected to IoT components.

The application also supports general cut/copy/paste functionality. There is also the option to show and hide any of these columns to avoid visual clutter.

In addition to Vuforia Studio, the company has released a new way to author their IoT-heavy procedures, called Vuforia Spatial Toolbox. An example of editing-in-AR, this program combines IoT data from the tracked objects to combine them into spatial hardware programming. Featured is a drag and drop AR interface that enables the visualization of and interaction with the data and logic of connected objects and machines. The authoring is mainly done through widgets that are pre-defined to track the IoT data and display them in a compact manner. The user can than combine these widgets to create a logical system between different components. [63].

Figure 2.11: Reality Composer can be used on any iOS device. Source: [64]

Reality Composer

Reality Composer is an application published by Apple and is available on iOS devices that support the required version of ARKit, which is available with iOS version 11. But it does not mean that it can't be used on a Mac. Since the application is based on Swift, the programming language for iOS applications, the current state of the created content can be exported to a eligible iOS device and viewed in an augmented reality view without much effort.

Reality Composer supports importing 3D elements into the scene. Nevertheless, the library provided by Apple has an abundance of resources, all located on the left side of the screen. The objects and camera angle can be configured easily by the attributes column on the right side of the screen. To add more functionality to objects, the author can click on the object in the scene. Animations are also one of these functionalities, created with ease using the frame-based animation widget on the bottom part of the application.

There is also a clipping feature to move objects with less precision but with more flexibility. A virtual grid on the ground helps with this interaction, as objects move clamped to grid lines when clipping is active.

All in all, Reality Composer is an authoring tool suited for commercial use, as there is a lack of tools to integrate step-by-step instructions or additional videos and images, which are essential for industrial scenarios. Without these use cases, the chances of this application to be used for industrial augmented reality is slim.

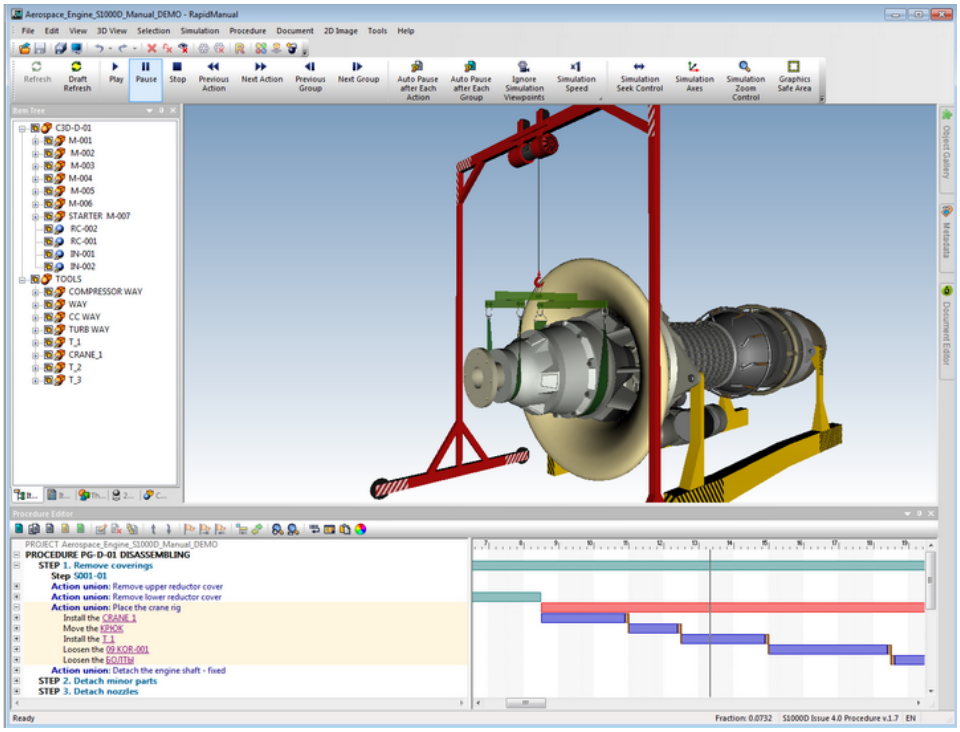

Figure 2.12: An example scene from RapidManual. Source: [65]

Others

While not specialized for augmented reality, authoring can also be completed through other 3D authoring tools. These programs also work on a step-by-step instruction basis, but augmented reality is not their focus and sometimes AR is not supported at all, unless some interim program connects the gap.

Some of these programs are RapidManual from Cortona3D, QuadriSpace and Simulink3D. Among them, RapidManual is of special interest since it is the authoring tool that is used at RE'FLEKT at the moment. Details on this program will be discussed in chapter 3.

Unity's Solutions

Unity3D (or Unity) is one of the leading game development engines in the world, and is one of the go-to programs for creating virtual and augmented reality experiences. According to the company, 60% of AR and VR applications are created with Unity [66].

Although AR companies utilize Unity for their end products, authoring is seldom done in Unity, but created content is imported into it. Until recently, it was still hard to develop an AR application within Unity since Editor was not enough to test augmented reality implementations and required deployment to the AR hardware.

Nevertheless, Unity Labs' Authoring Tools Group has worked on this subject and delivered some milestones with regards to mixed reality, beginning with the start of 2020. It started

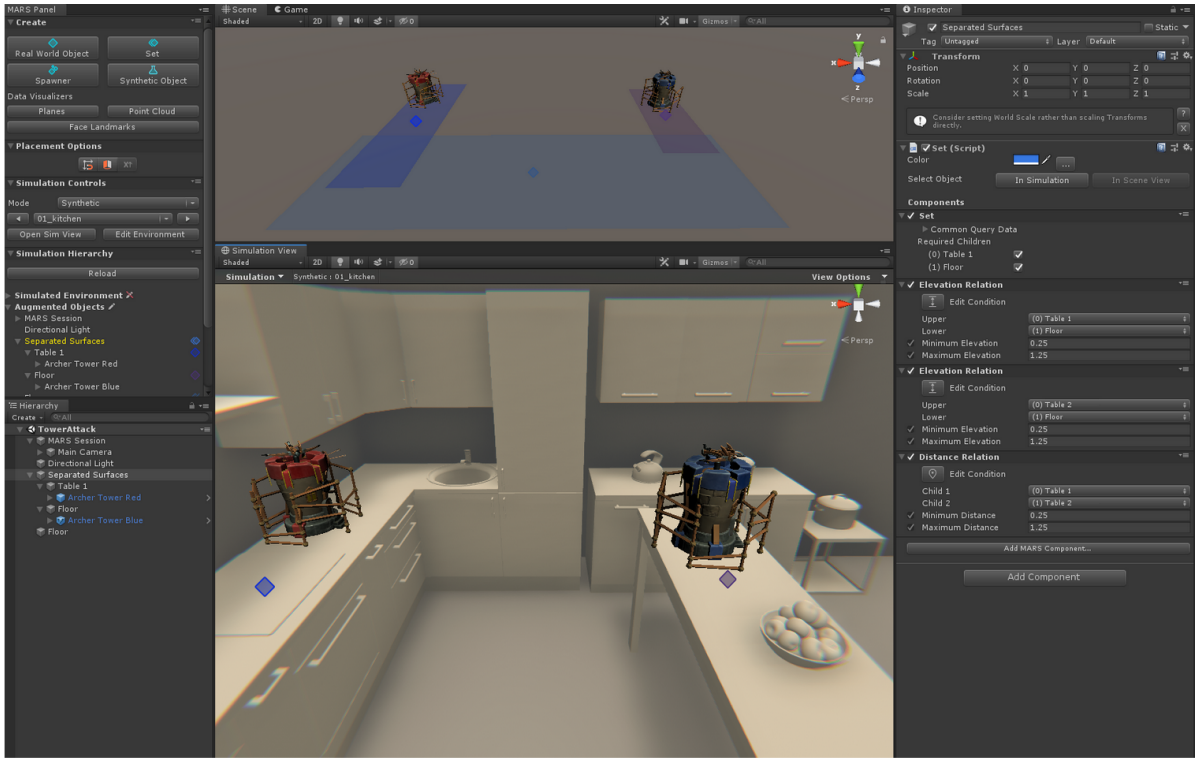

Figure 2.13: An inside look to Unity's plugin MARS. Source: [69]

with Unity publishing its own XR plugin framework to code on an uniform platform [67]. They followed with XR Interaction Toolkit, a developer toolkit for mixed reality programmers (although it favored VR more than AR) [68].

The most beneficial update in terms of authoring is the release of MARS: Mixed and Augmented Reality Studio [70]. It is a toolset for Unity developers to connect and design AR elements into the game more easily than before. It has two key parts: a Unity extension and companion apps for phones and AR head-mounted displays (HMDs).

The authoring team's starting point was that augmented reality apps created with Unity need to work in the real world, but developers also needed to get more information about the real world back into the Editor. Thus, they needed easy ways to tell Unity what's real and what's not and to let them design, develop, and test their real-world applications more easily [70].

MARS's first feature, the simulation view, aligns the Unity scene view and the real world view by simulating a real-world space in Unity Editor. It is similar to a virtual reality experience but with the focus of designing AR applications. The plugin includes tools and UI to see, prototype, test, and visualize robust AR apps as they will run in the real world [71]. The user can also simulate the device moving around the real world and see the "output" from the device in the editor. This method is called the simulation device view.

Along with those features, MARS offers new workflows for AR development in Unity and

advanced data manipulation [71]. MARS is still in development and not available as of late March 2020.

2.4 Research over AR Interactions

One of the research questions was to investigate what classifies as good authoring in industrial contexts. Another one was to find out which AR interactions are commonly used in authoring tools. This leads to finding out if there are any interaction elements that can be added to the authoring process, which may help with the platform incompatibility and lack of AR in authoring tools themselves.

Authoring tools are mainly run on desktop devices. However, the content is usually not viewed there, but on an AR-supported device. It can be hard for the author to realize 3D content on a program that does not provide required cues for augmented reality. Some examples may be visualizing annotations or objects on the real world or having difficulty determining virtual content size and orientation.

To find which AR techniques were used in the past and how efficient they were, another literature review on this matter is carried out. Following sections are some of the highlight points when it comes to developing applications specifically for augmented reality.

2.4.1 Best Design Practices

The best design practices for AR software and its solutions is suggested by Santos et. al. [36]. They are listed below:

- Enabling Exploration: Designing content that is non-linear and encouraging further study allows the users to be intrigued by the process. However, this is not one of the main goals for IAR.
- Promoting Collaboration: Allowing participants to work together (if the process is a collaborative one).
- Ensuring Immersion: Making the users concentrate more and be engaged at a constant level should be the aim with augmented reality. This is also one of the reasons AR applications reduce complexity and worktime for the operators.

The article also states that augmented reality is a step way to ubiquitous computing: A system of real and virtual objects replace images and virtual texts and audio elements replace words. It brings the world closer to a state where the human computer interaction is not done through a GUI but through interactions that enable anyone to effortlessly use any software.

AR enables the operator to learn the basics about the task by observing the augmented instructions and trying to perform the instructed subtask, to develop behaviour and movement patterns when performing the subtasks [72]. This explanation shows how AR is applied for the steps of ubiquitous computing as well.

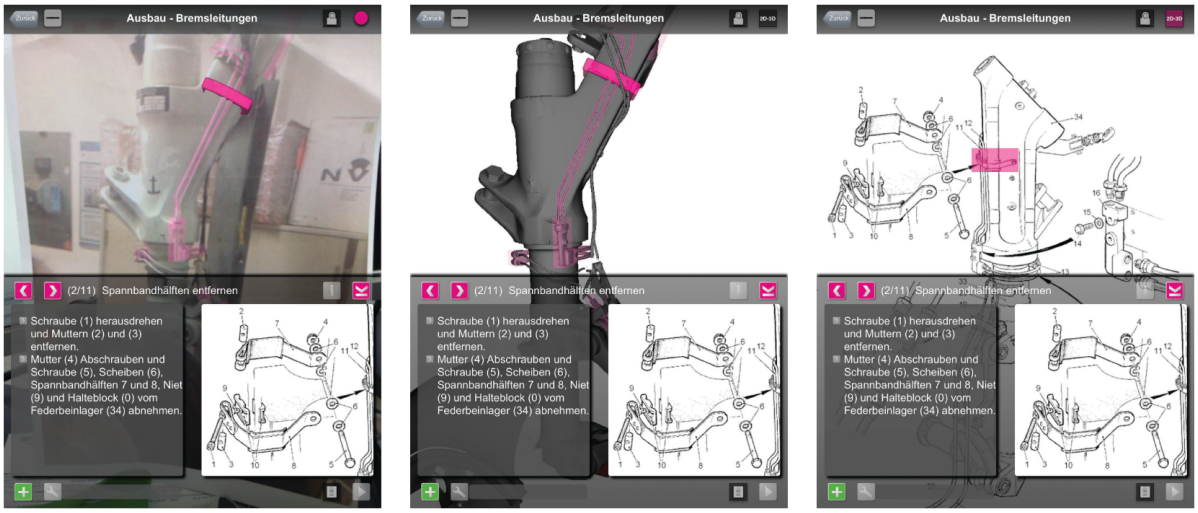

Figure 2.14: Engelke's solutions allows for different visualizations in runtime. From left to right: camera feed, CAD model only, areas of interest in 2D drawing. Source: [39]

2.4.2 Visualization Methods

In his survey from 2018, Palmarini compares the 3D content that was used in the reviewed literature [26]. The most common method utilized and mentioned is through dynamic 2D and 3D elements, with 40% of the cases. These visualizations mostly show how a task should be done with animated holograms. It is mostly regarded as successful as this method is superior to paper-based instructions [47].

In 26% of the applications, static 2D and 3D elements were used. These are not animated but remain in one state. Studies show that in some cases providing animations is not necessary at all and static elements can still provide enough information and accessibility [3].

Another less intrusive way to provide information is through text elements with 24%. Overlaying text information does not obstruct the field of view like virtual objects most of the time, and texts can be created and modified with ease. Still, their use cases are limited and they are not immersive as other solutions.

The last 8% consists of audio content. Audio guidance is supported in some of the systems. However, it is seldom used in step-by-step instructions and more on situational alerts, like giving user small feedback about environment hazards or application status.

In general, it is worth mentioning that the contents and context requirements have to be considered in order to develop the best AR solution. Engelke et. al.[39] believes that the operators should be able to visualize the instructions in the form of which is more suitable to them. In his research, he introduced the capability of switching between different visualization methods (see figure 2.14).

2.4.3 User Attention

Maintaining the user's attention and directing it to where it is needed is again where augmented reality excels. The form to do it is however debated and different solutions were found over the last couple of years.

Feiner, et. al. fixed this problem by using a 3D line drawn from a screen-fixed label to an offscreen target object or location [16].

Biocca and colleagues developed the "Attention Funnel" [73], a vector tunnel drawn to a target from a heads-up view and showed that it reduced search time compared to world-fixed labels or audible cues.

Tönnis and Klinker demonstrated that an egocentrically aligned screen-fixed 3D arrow projected in AR was faster at directing a car driver's attention than an exocentric alternative [74]. Schwerdtfeger and Klinker [75] studied AR attention-directing techniques to help users find and pick objects from stockroom storage bins. According to the results, their frame-based technique outperformed static 3D arrows and variants of the Attention Funnel.

Henderson et. al. used 2D and 3D arrows in combination to navigate the operator's gaze in the required direction. If the target component is behind the operator, a screen-fixed arrow points the user in the shortest rotational direction to the target. After the operator's view aligns with the target, another slimmer arrow points to the object of interest [76].

2.4.4 Occlusion

Occlusion of real and virtual objects on the camera feed is a fairly encountered problem in augmented reality. If the marker or tracked object becomes partially or fully obstructed, tracking algorithms can no longer work. On the other hand, holograms on the screen can occlude real world objects and make an operator's job harder, contrary to its purpose to make the process easier.

For the first variant, some solutions were tried out and implemented using real time depth information [58] or combining marker or markerless tracking with SLAM techniques [77].

For the second one, there were some research that was promising. Making virtual objects transparent and applying specialized shading has delivered good results, as these techniques are standardized in devices like HoloLens. Mohr et. al. built an AR interface in which AR animations appeared as "video phantoms" to reduce clutter on screen. If the selected element is hidden in the real world, the user is provided with a "ghost view" visualization, which indicates that the virtual object is occluded [78].

2.4.5 AR Interactions

There are a lot of interaction methods that can be used for augmented reality. In industrial augmented reality, only a handful of them have been implemented.

The most thorough review done on this matter was for augmenting the inside of a armored personnel carrier turret for the US forces [76] in 2009. The application was tested and reviewed with turret mechanics on-site and brought interesting results. The augmented contents were divided into 5 sections:

- Attention-directing information in the form of 3D and 2D arrows
- Text instructions describing the task and accompanying notes and warnings
- Registered labels showing the location of the target compo- nent and surrounding context
- A close-up view depicting a 3D virtual scene centered on the target at close range and rendered on a 2D screen-fixed panel
- 3D models of tools (e.g., a screwdriver)

3 RE'FLEKT and Pipeline Integration

This chapter is about RE'FLEKT, the company that is co-supporting this thesis together with Technical University of Munich. An introduction to the company, their products and vision are laid out, as well as important components in their production pipeline and how the authoring tool developed in this thesis fits in the ecosystem.

3.1 RE'FLEKT

3.1.1 Overview

RE'FLEKT is a Munich based technology company that enables any business or industry to create their own in-house Augmented and Mixed Reality applications [79]. The goal of the company is to produce scalable and affordable AR (and MR) applications for businesses. Their expertise allows them to infuse their industry knowledge into customized mixed reality solutions.

Figure 3.1: The banner and logo of RE'FLEKT. Source: [80]

Founded in 2012, the company's focus shifted from entertainment and one-time projects to enterprise use cases and scalable solutions over time. Their first product RE'FLEKT ONE, an AR viewer application for enterprises, was co-funded by Bosch and more investors followed after it was a success along with other custom projects.

RE'FLEKT's presence in trade shows like Augmented World Expo (AWE) was growing as they launched their second app RE'FLEKT Remote for remote collaboration solutions in 2017. The company expanded in 2018, opening a second office in San Francisco. They contributed to AWE Europe coming to Munich in three consecutive years with their influence and sponsorship.

Today, with two robust products and occasional customer projects, RE'FLEKT is one of the leading solution providers for industrial augmented reality. Some of their investors are Bosch,

BASF and Prosegur, while those are joined by customers and partners like Audi, Siemens, BMW, Eon, Hyperloop, Hyundai, Leybold, Microsoft, Porsche, Seepex and Thyssen Krupp. The team has also closely worked with Microsoft on their latest HMD HoloLens2, and are their official seller in Germany. It can be said that they are the dominant force of industrial augmented reality in Europe and they are looking to extend their influence.

3.1.2 Augmented Reality Solutions at RE'FLEKT

Apart from customer projects, RE'FLEKT has two major, customizable and subscription based products for industries.

The first product is RE'FLEKT ONE, the first scalable augmented reality platform that enables companies to independently create their own augmented reality applications. The award-winning platform lets companies create visuals and guides on smartphones, tablets, and head mounted displays to simplify production, training, maintenance, and repair. RE-FLEKT ONE's advanced technology leverages existing design and documentation data and transforms it into interactive 2D and 3D visualizations for real-time viewing on real objects [81].

Their second product is REFLEKT Remote, which connects technicians on site with the right support experts and allows real-time video support with augmented reality. RE'FLEKT's intelligent routing technology identifies and connects technicians directly with a support expert. The expert can assist the field technician by simply drawing steps or placing visual instructions; the visual engineer sees the visual direct in his field of vision. The platform provides technicians with access to a global database of issues already resolved to accelerate repair and maintenance processes [81].

Since the authoring tool is potentially going to be integrated into RE'FLEKT ONE pipeline, a deeper look into the intrinsics of this program will follow.

3.2 RE'FLEKT ONE & Pipeline

Although the name of the product and the application is RE'FLEKT ONE, it actually consists of multiple applications that work together in an ecosystem. The information that is provided in the sections below stem from RE'FLEKT's own knowledge base, which means the resources can't be traced or published in the context of this thesis.

3.2.1 RE'FLEKT ONE Ecosystem

RE'FLEKT ONE ecosystem consists of four major components. Those are frontend, authoring, publication, license management, a distribution server and an user feedback server. Since our focus is the task of authoring, all parts will be explained briefly and relevant ones will be explained in detail.

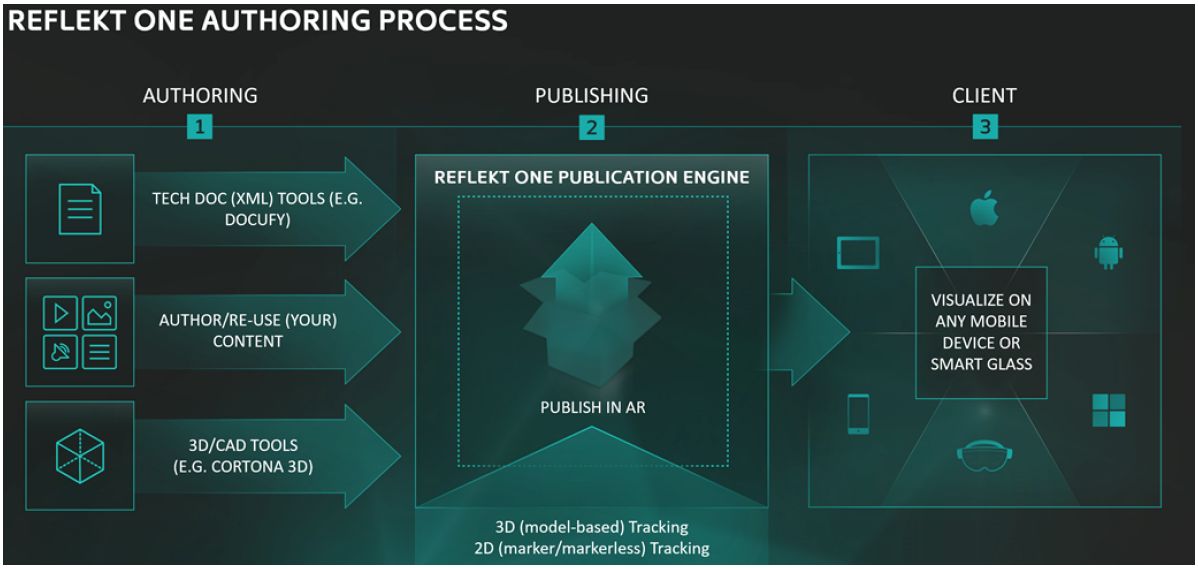

Figure 3.2: An overview of the authoring process at RE'FLEKT. Courtesy of RE'FLEKT.

Authoring Client

The authoring client that is used in the process is RapidManual, a CMS program infused with CAD capabilities. It has the ability to create step-by-step instructions to support product life cycle management operations such as inspections or repairs. Technical information is transformed into a digital manual called a scenario. Scenarios consist of steps, which may or may not have animations attached to them. Animations are the 3D visualizations that highlight or manipulate a part of the visualization in each step, played in loops. Videos, images, text and additional material can be added to steps as well.

Tracking Input

For the tracking information/configuration, RE'FLEKT SYNC is used. SYNC is a program which enables the users to create or modify their own tracking configuration through an user interface. The application supports marker based and object based tracking techniques. For marker tracking, a marker or an placeholder image is added into the scene and additional information like marker size, tracking quality or threshold are assigned.

For object tracking, a CAD model of that object has to be imported first. After that, the starting position and the orientation of the visual object is shaped. The object tracking technology used in RE'FLEKT comes from VisionLib, which is fully integrated into SYNC. To align the virtual model with the the real one to start tracking, a low-polygon version of the object is used first. This is called the visual aid model. After the tracking is initialized, the scenario content from RapidManual is activated. Here, models with higher polygons and more details are utilized. This version of the virtual content is called the scene model. In SYNC, both models can be configured with ease. Detailed information like distance from

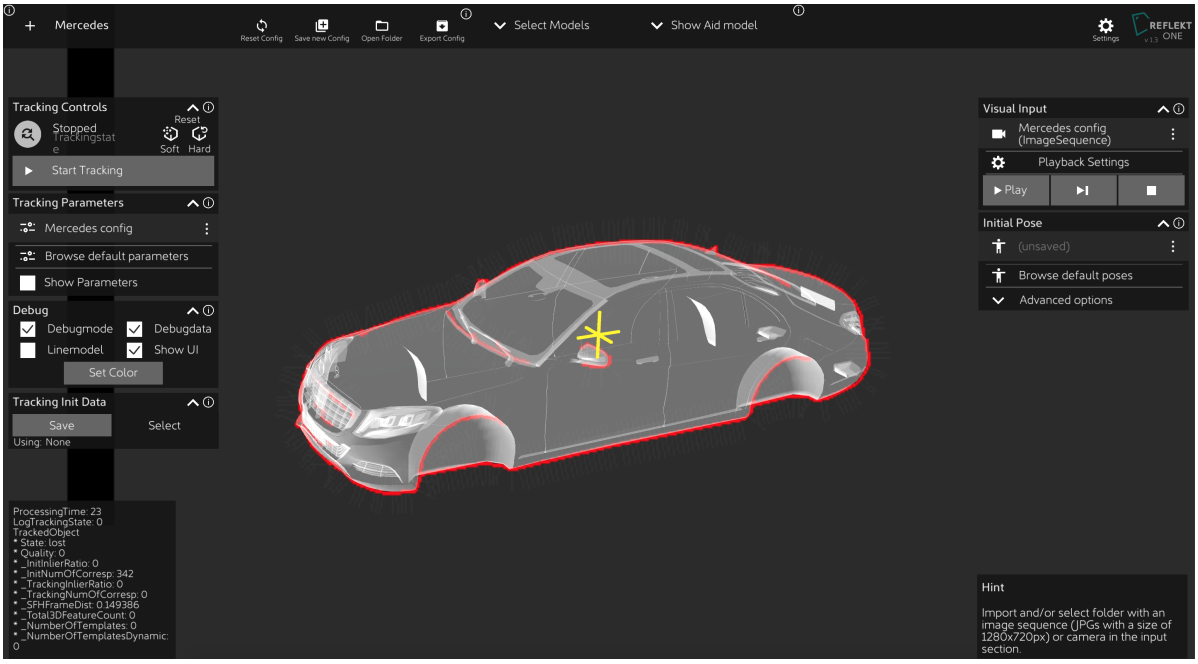

Figure 3.3: A screenshot from RE'FLEKT SYNC demonstrating the object based tracking configuration. Courtesy of RE'FLEKT.

camera, object size, animations, Laplace thresholds etc. can be programmed here. The output consists of two files: The tracking configuration (a .vl file) and the correctly configured object model (usually an .obj file).

RE'FLEKT ONE Author

The scenario content and the tracking information is merged in a program called RE'FLEKT ONE Author, which is the publication engine that merges the authored content with the tracking configuration. After the authoring is done in RapidManual, the created scenario content can be outputted in a format called DITA, which is an XML-based architecture for documents intended primarily for consumption by humans[82]. The 3D data in RapidManual is parsed into a X3D file, a XML format supporting 3D visualizations. At last, if there were any animations added to the scenario, they are stored in an interactive XML file. Publication engine then turns these three files into a C3D class model, which is ultimately converted into a standard X3D output.

The tracking configuration and the model is added here as well. If the scenario output from RapidManual already contains the object model, it is modified with the output from SYNC. After the merge of the tracking data, the publication result is saved to a local or online repository for the clients. The publication can be accessed on the client side (RE'FLEKT ONE VIewer) and used immediately. Scenarios can also be saved locally for offline use.

Frontend (RE'FLEKT ONE Viewer)

The frontend, also known as RE'FLEKT ONE Viewer, is the application which is used by the clients of the company. Users interact with the application using the built in UI systems and do not need to worry about the technical details. All they need to do is sign in, open their preferred scenario and initialize the tracking by aligning the marker or the visual aid model with the tracked object. The program works on different platforms such as iOS, Android, Windows Standalone and Universal Windows Platforms, which is used by HoloLens devices.

RE'FLEKT ONE Viewer has a complex architecture which is a mixture of MVC (Model View Controller) and MVVM (Model View Viewmodel). Each function is handled by a separate module which communicate within themselves and the core program using interfaces. A module that is interesting to the thesis is the X3DModule. This module accesses the X3D API and parses the X3D input from the publication into a 3D visualization. It has to be noted that X3D API is separate from the ONE repository and can be detached and used in other programs as well.

Figure 3.4: RE'FLEKT ONE being used on an iPad for an assembly procedure. Source: [83]

3.2.2 UX Department Brainstorming

In order to identify the needs of the users of the authoring client, a survey was conducted with the UX (User Experience) department of RE'FLEKT. These colleagues are mostly responsible for creating 3D and scenario content, authoring them and then presenting the results to the customers, among other things. Sometimes customer training is also done by people in

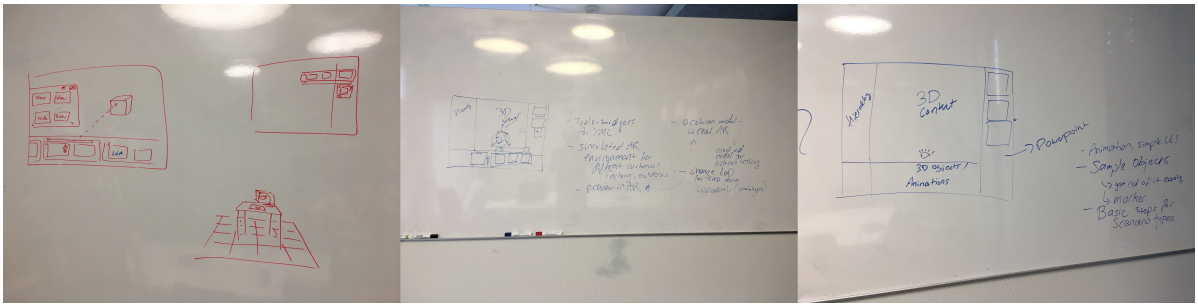

Figure 3.5: Some pictures from the UX brainstorming session.

this department.

This leads to the conclusion that the input and feedback from these colleagues would be essential to shape the authoring tool. Since they are also familiar with the pipeline and had a conceptual understanding of all the tools used in it, they could also walk me through the pipeline processes and comment on where the new tool could fit in.

The survey was done with 4 people. The highlights of this survey goes as follows:

- An easy way to import 3D annotations and other 3D visualizations is needed.
- A method should be found that helps visualize what is going to be in AR and what is not.
- The process should be more tangible, the author should understand the medium and place in which the content is consumed.
- There could be a better way to visualize the scenario steps used in RapidManual. Although the XML format used in RapidManual is human readable, it could be transformed into a better conception.
- Animations should be realized with the help of a simple user interface.
- Sample 3D objects should be easy to access.
- 3D content should be in the center.
- A grid on the 3D space should be necessary.
- Side windows like hierarchy and scenario information could be hidden when authoring 3D content.

4 Implementation

This chapter goes through the entire implementation process. The proposed work after the research will be explained first. After that, the implementation approach is explained. This section is followed by a description of the external programs and tools used. At last, the editor implementation is demonstrated, followed by the addition of augmented reality interactions into the application.

4.1 Proposed Work

4.1.1 Research Results

The research shows that there is definitely a need for an uncomplicated and intuitive authoring tool which should handle the input and the output straightforwardly. This means fitting into the current pipeline as well.

In addition, we can confirm that there is somewhat a "status quo" in terms of AR authoring. The research has shown that there is not enough information gathered on an contemporary authoring tool that also benefits from addition of augmented reality utilities. The applications that fit in this criteria are scarce, so it makes it harder to pinpoint what can be defined as a successful authoring for AR.

Nevertheless, there is enough information from past implementations in the last 20 years that can help to identify the best practices and previous mistakes. We can conclude that industries are not focusing on developing specific applications to solve their problems, but they are interested in finding standard applications for adaptation to their production scenario. Plus, AR applications in industrial environments are directly associated with complex processes. The technology used is a critical factor to improve process flexibility, which is the key to a company's success. Process flexibility is achieved by not only providing information to manufacturing, but also by improving inspection, providing more efficient logistics, support maintenance and process monitoring [26].

Regarding the support for augmented reality utilities, the demand for precision stands out. Accuracy in industrial augmented reality applications was argued in 40% of the studies to be an essential feature in the application, while sharpness in 23% [26]. This means tracking stability and correct alignment of real and virtual scenes is of utmost importance. Other than that, there are implementations in fields of IoT, tangible interactions and augmented reality previews for authoring. All of them are praised in the literature.

As a final thought, much of the key points revealed by the initial research is implemented in the RE'FLEKT ONE ecosystem already. A majority of the implementation fields like maintenance, repair and inspection can be handled by RE'FLEKT ONE. The need for scaling

authoring solutions can be covered with the flexibility of RE'FLEKT in terms of branding and individual content creation for customers. In terms of tracking, RE'FLEKT one offers both marker-based and object based tracking. Both are industry standard methods and both work with decent FPS and without critical overload of the hardware. Supporting content in ONE consists of text, videos and images in step-by-step fashion, all of them frequently used and proven in other applications in the AR industry.

4.1.2 Expectations

The revelation that RE'FLEKT ONE is within industry standards on most aspects leads to the fact that a new authoring tool could potentially be integrated into the pipeline without much effort. The expectations from the company would be to create a step stone for a future authoring tool than can replace the current one, RapidManual, or be used alongside it. This application should be done with Unity to be easy to integrate into the pipeline by using Unity's packaging and API system. It should be noted here that both RE'FLEKT ONE Viewer, RE'FLEKT ONE Author and RE'FLEKT SYNC are made with Unity, which leads to a quick data transformation between these programs.

A new authoring tool should counter the deficiencies of RapidManual on the topic of AR interactions and possibly replace an outsourced program with an in-house one. An in-house program also allows for fast bug fixing and new feature implementations.

The new authoring tool should display an interface to get the 3D data as an input, make changes and additions and finally output it in a type that is consumable by the Viewer application.

On the topic of AR interactions, a selection of interactions deemed most useful will be implemented into the authoring tool. At the end of the thesis, a second survey will be organized with the UX and R&D departments to review both the editor and augmented reality features.

The full integration into the pipeline is thus not expected due to the workload and short period of time.

4.2 Approach Taken

The research revealed the key features of industrial augmented reality, along with the tools that are needed. Another conclusion would be how to best realize the cross-device editing between desktop and AR use and what additions authors would need to effectively transport the information to the user.

But first, it will make more sense to explain the coding environment, used APIs and other assets since they will be frequently referred in the rest of the thesis.

4.2.1 Coding Environment, APIs, Assets

Unity

It is determined that the development is going to be carried out in Unity3D (short: Unity), a game engine which became the go-to platform for mixed reality development in the last couple of years. Unity also provides a detailed online manual and a dedicated forum for any type of discussion, which helps with the development process. The development language used in Unity is C# most of the time, with additional shader languages ready to use.

Another advantage of Unity3D is the Asset Store and Package system. Other Unity users can create their own features and publish them in an online store, either for free or for a compensation. These can be added as packages to the project and can be added and removed from the project without much trouble. Some aspects that are not the focus of the thesis (building the editor program, object manipulation) will be integrated through these assets. This will allow for more time to focus on tasks that are relevant to the research questions of this thesis.

The literature review suggested that game engines were used regularly in the field, and RE'FLEKT also uses Unity3D for the development of ONE along with other helper applications such as SYNC and RF ONE Author. It paves the way for effortless data transition between Unity-based programs thanks to Unity's packaging and bundle systems. The natural choice for the development platform is Unity3D because of these reasons.

Mixed Reality Toolkit

Mixed Reality Toolkit (short: MRTK) is a Unity plugin developed by Microsoft to assist the creation of programs for mixed reality. At the start, the toolkit was aimed for HoloLens development, but later it was expanded to support various mixed reality devices, mostly head mounted ones (HMDs). MRTK offers a lot of features for MR development, consisting of input gestures, pre-built game objects, example scenes, camera configurations and more. This tool is also integrated and frequently used in the development of ONE Viewer.

Augmented Photos

Augmented Photos is a fairly new addition to the features of ONE Viewer. After a development duration of 8 months by the R&D team, it was integrated into the product. A new version followed in February 2020.

Augmented Photos Module is an extension to scenarios in which edge-based or marker-based tracking is used. The user can take photos, browse them in the augmented photo gallery and display them in full screen. The photos are augmented, which means the user has access to AR functionalities on the photo.

In addition, user can enable "auto capture" to take photos automatically when continuing the tracking. While browsing photos, the user can swipe through them chronologically or spatially. If spatial selection is active, the next photo is selected depending on the swipe direction. The most relevant photo in that swipe direction is selected and displayed. This

way, the user does not have to scroll through all the photos to find the one with the desired angle. Other functions of ONE remain integrated and work seamlessly with the addition.

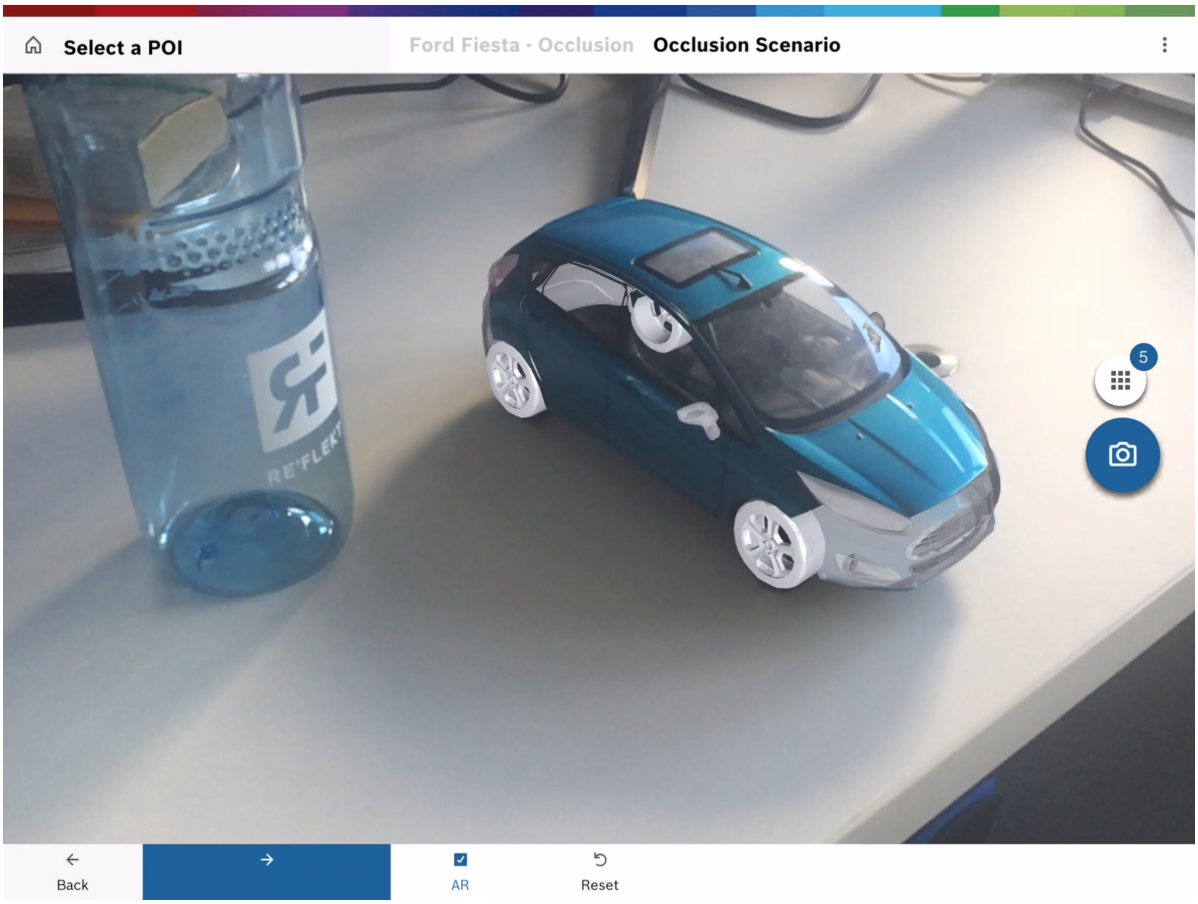

Figure 4.1: A screenshot displaying the Augmented Photo feature in RE'FLEKT ONE.

Runtime Transform Handles

Runtime Transform Handles is an Asset found in Unity Asset Store. It provides runtime 3D controls that are used to manipulate items in the scene. It is compatible with every platform Unity supports. There are three built-in transform tools to position, rotate and scale object via transform component. Supplementary controls such as scene gizmo, selection gizmo and grid allows to change viewing angle and projection mode, identify selected objects and orientate in scene space [84].

Runtime Transform Handles is a lite version of a bigger asset called Runtime Editor. However, some functionality of this program is also included in Runtime Transform Handles. These are Undo, Redo and Delete functionalities, switching between Editor and Preview modes, dragging and dropping objects from a menu selection, snap to grid option and changing the pivot of the object.

Runtime Hierarchy and Inspector

Runtime Hierarchy and Inspector is another Unity Asset that is essentially a runtime inspector and hierarchy solution for Unity 3D. It works on any platform that Unity supports, including mobile platforms [85]. Its main functionality is creating a copy or a sub-copy of the scene hierarchy of Unity Editor and access it on runtime. It also integrates inspector support, mimicking the inspector behaviour in Unity, and providing accessibility to the scripting components attached to the GameObject.

Standalone File Browser

Standalone File Browser is a Unity asset that eases the burden of importing any file into the Unity scene. It is a wrapper for native file dialogues on Windows, Mac and Linux [86]. The API can be adjusted for any file type and can be used on multiple occasions in the same scene setting. Another important part is that it can both work on Editor and on Runtime in Unity. It can also be used to export files and scenes out of the authoring tool. This asset was already used in SYNC, which adds to its credibility in the company and one of the reasons why it is utilized in the thesis as well.

X3DModule

X3DModule is taken from ONE development and was already discussed earlier. Import, visualization and modification of the X3D data is all handled by this module. The module is specialized for ONE, which has its up- and downsides: The downside is that it has to be reconfigured a bit to be used in the thesis. The upside is that due to its use in ONE and in the pipeline, importing and exporting X3D data (if needed in the future) should not be a major issue. This module is also maintained and updated regularly by ONE development team, so any updates can be transferred in here as well. Any required changes on the module to fit the authoring tool can also be developed jointly with the ONE development team in the future.

4.2.2 Mindmap

The next step is to identify to be implemented features according to the needs and suitability of RE'FLEKT ONE pipeline and the authors. For this matter, a diagram was created first to categorize the possible features the authoring tool is going to have. They are divided into four sections: Input, output, editor and AR specific authoring. The literature review and the feedback from the UX department helped to shape out the items. At last, each item was sorted by importance in each branch after a final review.

Input

The input branch consists of CAD/X3D data, XML, preset backgrounds, template scenes, Augmented Photos, audio and video recordings. We already know that we require the X3D data from RapidManual, so this is a must. Pure XML integration is not required since the

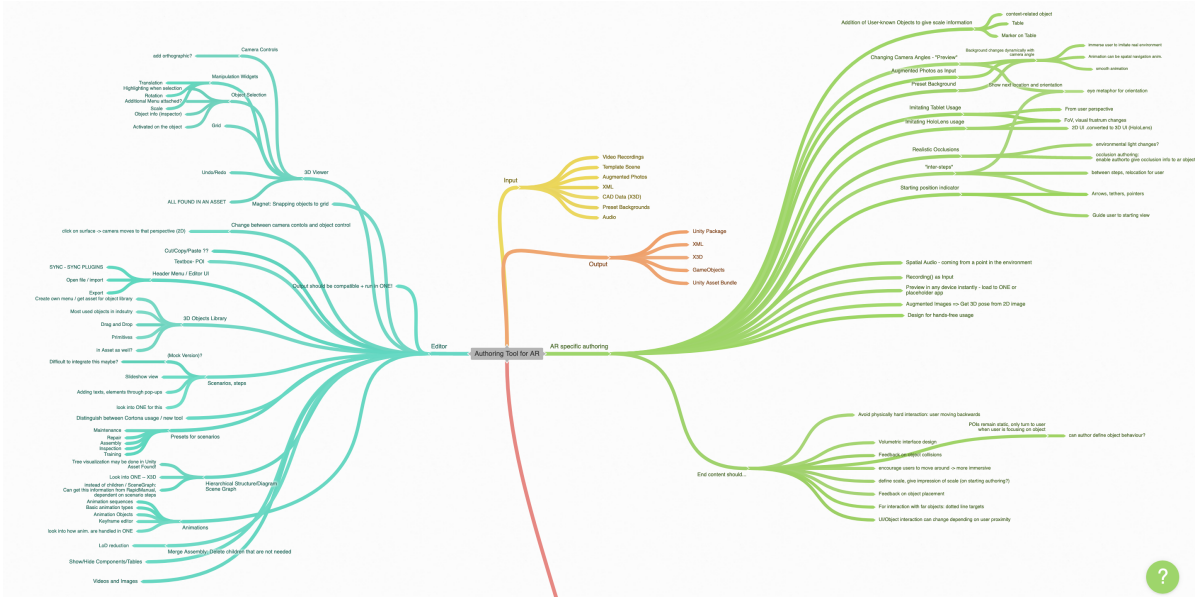

Figure 4.2: An overview of the diagram for the authoring tool features. Main branches of the diagram are input, output, editor, AR specific authoring and literature review (not included in the screenshot).

additional data like videos, text and images are added into the final X3D package in the publication engine.

Preset background addition can be done quickly and with fast results, so this will be a priority. More on this topic will be explained in section 4.2.2.

Augmented Photos is a new feature of ONE which can be an essential asset to the authoring process. It's featured were explained in section 4.2.1. Implementation of this feature makes sense from the product perspective: Authors can also benefit from the utilities Augmented Photos provides as well as clients.

Output

The output branch represents the possible output values for the authoring tool. This section has 4 possibilities: XML, X3D, Unity Package or Unity Asset Bundle.

It was already stated previously that the goal of this thesis is not a major integration yet, but to see which possibilities are optimal. Considering previous research done on this matter in the R&D department, Unity Package solutions will be investigated as an output option. This method was researched and tried in the company some time ago for a different project and was found promising.

Outputting the authoring solution as XML and X3D are still viable options, however, it makes more sense to transfer data as packages will be transferred between Unity environments. The other option would need the encapsulation of XML/X3D as output and parsing again in ONE Viewer, which is not necessary thanks to the other option.

Editor

On the editor branch there are a lot of concepts that are covered by the two assets mentioned in section 4.2.1. These concepts are: Camera controls, changing camera mode, object selection and manipulation, undo/redo functionality, a grid, snapping to grid, and drag&drop functionality. Most of them are essential features that are must-have in any editor, while others are a nice bonus.

It was determined in brainstorming sessions that the layout of the whole program should be similar to other editor programs. This implies a header menu, hierarchy and additional information on sides on the screen and the 3D data in the center. Any additional parts can be put onto the bottom side. Following this design, the question there came up about what each section should contain.

For the header menu, it was concluded that out of the options, the import button and buttons showcasing primal editor functionalities should be put on the top side. Runtime Editor's assets occupy most of these, plus my additions for importing X3D data, background pictures and other augmented reality elements.

For the left and right side of the screen, runtime hierarchy and inspector are the go-to options. Regarding the hierarchy, there were other options considered: Building a hierarchy from scratch to match the X3D data, displaying everything in a tree structure and pruning the tree for unnecessary X3D objects. Since the asset was compatible with other parts of the program and provided lots of utility, it was selected instead of other options. However, modifications can be applied for future work.

The center of the screen is reserved for 3D data and visual authoring, it being the most crucial part of the process. To avoid clutter, the design is kept as simple as possible. Apart from the 3D objects, only the grid is visible, which can be toggled. The main focus here is to provide a robust interaction system for selecting and manipulating objects plus providing a smoother experience with services like deletion and undo/redo.

This leaves us with the bottom half of the screen. The implementation options that are not yet mentioned from the mindmap include scenario and step logic, object library, animations, presets for scenarios and hiding specific components. Scenario and step logic is a crucial part of industrial AR operations which is a must, so this is selected for implementation. An object library is also necessary to provide the adequate 3D tools for authors. Most common used objects in the industry like screws and corks will be coupled with widgets like arrows and floating texts and images.

Although animations are also a crucial part of the authoring, the X3D input from RapidManual has its own animation API. This is not directly transferable to Unity's Animator component and due to time circumstances it was not deemed necessary by the company for the initial integration. This is however still a very important step in creating a independent authoring tool and will be investigated in the future.

What is meant by preset scenarios is to have some example Unity scenes for different industry operations like maintenance, inspection, training etc. The idea was to incorporate dedicated object libraries to each operation to decrease the search time for objects related to that tasks, making initiation steps a bit easier. Since this is another nice-to-have feature, this

idea will be saved among future work.

Likewise, hiding components on the screen is an useful but not necessary utility. Still, there is some room needed to fit all the features onto the screen. For this reason, a tab-system will be integrated to the bottom side of the screen, allowing step and object library control on different tabs.

AR Interactions

This branch of the mindmap is the most interesting part. It includes all AR specific authoring methods that can be implemented for the thesis. This list was put together with a combination of the literature review that was undergone for this thesis and UX & R&D department inputs. Afterwards, it was sorted by importance and relevance, revealing the first AR features of the authoring tool.

The list, sorted by priority, is down below:

- Addition of user-known objects to give scale information: The most encountered problem in an authoring scene (even in AR and VR applications to some extent) is the initial scale of the object. If there is only one object in the scene, it can be hard to understand the dimensions of it without another object or any indicators for its size. Two solutions will be provided for this problem. The first is a direct one: Through a button click in the editor the dimensions of the current selected object will be displayed. The second one is a bit more vague, but it is still helpful. Since the author has an object library at its disposal, the objects from that library can be used to have an idea on the dimensions of the initial object. Here, it is assumed that the author is already familiar with the objects from the library and roughly knows their size. The two objects in the scene create a size comparison from which the author should deduce necessary information. Additionally, AR related objects like markers are also useful tools for authoring for marker tracking scenarios.

- Preview Mode and Imitating Handheld/HoloLens Usage: One of the things that has been missing in the current pipeline is a visualization of the authoring outcome. Authoring is done on desktop, but the final content is consumed mostly on handheld and HMD devices. A preview mode enables the author to view the authoring outcome on a visualization that resemble these platform environments. For this, two different options will be implemented: One for handheld devices and one for HMD devices. An environment creation inspired from RE'FLEKT ONE will allow the author to see the final outcome on the client and make required changes without having to do another iteration after the authoring is already published. For handheld devices, a replica of RE'FLEKT ONE screen will be created. For HMD, two variations will be present. The field of view for HoloLens and HoloLens 2 will be shown in the authoring tool. This way, authors can readjust the size and location of virtual objects. Again, this is for the ease of the authors and prevents any more iterations of the authoring due to issues explained above.

- Augmented Photos as Input: Augmented Photos as a feature was already handled in section 4.2.1. Augmented Photos permits authors to gain access to previous content and have a better spatial understanding of the virtual object when authoring. Having access to augmented photos from different angles, the user can use this information to place virtual objects with more accuracy in the authoring scene.

  One downside to this approach is that in order to have an augmented photo, the content should already be created and used in RE'FLEKT ONE. So Augmented Photos can be used when the author needs to make modifications to the content or uses the same virtual object for another scenario.

- Preset Backgrounds: The authoring is mostly done in a virtual environment. What lacks here is a spatial information regarding the environment in which the client is used: ONE is mostly used in industrial environments consisting of factories, workshops and production lines. When the background information is disregarded in content creation, it can lead to issues like color mismatches between the AR content and the background. For example, if the client is put to use in a setting where most of the colors are tones of gray and the AR content is of the same color, it will be hard for the user to distinguish between separate elements of the scene. To avoid this, the author will have the ability to import various background pictures to the scene and adjust the authoring accordingly.

- Starting Position Indicator: An indication for the starting position for the user is a nice idea as well. The idea is to direct the client user to a specific spot to start the operation after initializing the tracking. This could be done with an object like an arrow that indicates a specific location. This can still be done in the authoring tool by adding an initial step and placing the object that gives location information. However, there is no official support from RE'FLEKT for this feature as this feature would require a change in the pipeline and the redesign of previously authored content.

- Inter-steps: Sometimes during the operation, the user needs to change their location to continue with the task or have a better angle for see virtual content in a better way. The idea of inter-steps is to guide the user between tasks by virtual arrows. Adding a separate entity as an inter-step would again need a change in the pipeline, similar to the starting position indicator. A workaround for that option is to add new steps in a scenario and author the logic of inter-steps in that new step.

One feature that would make an impact is realistic occlusions. This feature is about making the tracking more robust and resistant to environmental changes. These include hiding virtual object behind real ones if the real object (partially or wholly) occludes the tracked object. Although this is implemented in ONE to some extent, it needs to be updated with the latest technological advancements, especially for the handheld client.

This would be a fine addition to the authoring system and to the client in general. Although adding realistic occlusions to the authoring program and provides better immersion and accuracy overall, this task requires a joint development with the ONE Client team. Therefore, it is postponed to the later stages of development of the authoring tool.

There are some other features that came up: Spatial audio solutions, recording implementations, preview solution in device, augmented images (getting 3D pose from 2D images) and hands-free usage. Even though some of these are interesting research aspects and have the potential for integration, they did not make the cut unfortunately, due to RE'FLEKT's prioritization and time limitations.

4.2.3 Authoring Tool Overview

This section boils down to the selected features and development track of the authoring tool. To summarize: Primary input format will be X3D data. The output would then be a compressed Unity package, ready for consumption by RE'FLEKT ONE or RE'FLEKT ONE Author. The editor is designed as any other editor in the industry with essential features like object selection, manipulation, hierarchy and inspector access, object library, undo/redo, scenario steps addition, configuration etc. Additionally, a handful of AR specific authoring elements will be integrated into the program. These include bounds visualization, preview modes, Augmented Photos, preset backgrounds and inter-steps.

4.3 Editor

The implementation starts off with the scene setup and integration of the assets explained in section 4.2.1.

The used APIs and company features are each stored in their individual folder in the project setup, under the Assets folder. Each folder is further divided into Resources, Scenes, Scripts and Prefabs folders, if any of these are present. The major folders are as follows:

- Authoring Tool: The core folder of the implementation. Contains the scripts handling the connection and management of other components plus the AR feature scripts.
- Augmented Photos: This folder in located inside the Authoring Tool folder, since the assets in this folder is a specific collection of Augmented Photos features and is unique. Features were extracted from different implementations of Augmented Photos. Detailed information will follow in section 4.4.2.
- BattleHub: The folder containing the Unity Asset "Runtime Handles".
- RuntimeInspector: The folder containing the Unity Asset "Runtime Hierarchy and Inspector".
- Standalone File Browser: The folder containing the Unity Asset with the same name.
- X3D: Contains the complete X3D logic, taken from the latest RE'FLEKT ONE implementation.

Only one scene will be used for the application, called the `"MainScene"`.

4.3.1 Events and EventController

`UnityEvents` are a way of allowing user driven callback to be persisted from editor time to runtime without the need for additional programming and script configuration [87]. UnityEvents are also used to decouple systems and scripts, meanwhile being easy to use as an alternative to the C# delegate system.

A centralized event system is implemented to ease the communication of different scripts with each other. This way, events can be invoked in a script that triggers other scripts sequentially.

All of the custom events are derived from the events in the script `Events.cs`. For example, `Events.cs` contains an Event called `BoolEvent`, which derives from `UnityEvent` with the type `<bool>`.

These events are instantiated in a script called `EventController.cs`, which is attached to an empty GameObject in the scene. All the events that are included derive from the base classes in `Events.cs`. As an example, an event which indicates whether an augmented photo is displayed is implemented like this:

```
public Events.BoolEvent isAugmentedPhotosActive = new Events.BoolEvent();
```

Each script which makes use of events, has a `private EventController _eventController` variable inside them. This attribute then instantiated on Unity's `Awake()` function, which is called once when the application is run:

```
_eventController = FindObjectOfType<EventController>();
```

As an example, when an augmented photo is toggled on or off, this event is invoked with a `bool` parameter. Other scripts, who are listening on that specific event, can use the parameter for their internal logic:

```
_eventController.isAugmentedPhotosActive.AddListener(isActive => {_isAugmentedPhotosActive
= isActive});
```

4.3.2 Transform Runtime Handles

The backbone of the editor is the Transform Runtime Handles asset, which can be accessed under one prefab called `TransformHandles` (see figure 4.3). Some of its children are the main scene camera, EventSystem, Transform Handles UI and Transform Runtime Handles components.

The UI elements are the buttons on the top left part of the screen (see figure 4.5). These buttons basically provide the essentials for an Editor, along with some additions for authoring. The buttons are:

- Focus: Focuses the camera to the selected object(s).
- View(Q), Move(W), Rotate(E), Scale(R): These buttons mimic the Hand, Move, Rotate and Scale tools of Unity Editor. The selected object can be manipulated exactly like and object in an Unity scene. The shortcuts, same as Unity tools, can be used for manipulation as well. The letters next to the buttons are the shortcut keys for each individual button.

- Select Parent: Selects the parent of the selected object in the hierarchy. If more than one parent is selected, it selects the parent that is highest in the hierarchy.
- Grid: Toggles the grid on the scene on and off. The grid is only applied on the ground floor, in other words, the X-Z plane. The grid also scales and adapts to the camera's height difference from the origin of the scene.
- Undo, Redo: Each action in the scene is recorded on an internal history. These buttons are used to revert any changes in the scene.
- Delete: Deletes the selected object and removes it from the hierarchy.
- View HMD, View Tablet: Toggles the tablet or the HMD emulation respectively.

Another set of buttons are to the top right. These buttons are mostly used for activating AR features of importing files into the scene. They are listed below:

- Bounds: Calculates and displays the bounds of the selected object.
- Visual Perimeters: Toggles the visual perimeters. This opens additional buttons for toggling the separate perimeters.
- Coordinate System: Toggles the coordinate system (This feature, along with the two features listed before, are illustrated in section 4.4.3.
- Open X3D: Opens a file dialog to select an X3D file and import it into the scene.
- Open Augmented Photos: Opens a file dialog to select an Augmented Photos folder and import the photos into the tool.
- Activate Background: Toggles the background feature (section 4.4.1) on and off.
- Change Background: Opens a file dialog to change the background image. This does not toggle the background, it only changes the image that is used.

In order to allow manipulation of any object in the scene, that object must have the script `ExposeToEditor.cs` attached to them. When the application starts, all objects in the scene that have this component will be automatically registered for manipulation. This way, other elements in the scene which should not be interactable (like the coordinate system) can't be selected in any way.

`MainSceneEditor` is the class that controls most of the UI, the access point to the editor functionality and the application state. This class is adapted from the Transform Runtime Handles Editor example script. Since some of the buttons are also carried over from that, they are untouched. The newer buttons are added here for functionality. The trigger events are added to the buttons in the Unity Editor.

On startup, the editor state is registered into the Runtime Transform Handles internal data. The functions used here are mostly callback functions that forward new selected objects, UI input and similar changes to the core of the asset, where it is handled internally. Most useful

functions can be accessed from the object called `Editor`, as it can be seen in the two examples in listing 4.1.

More complicated features of the asset, such as Undo, Box Selection, runtime handles etc. are each controlled by components attached to their own GameObjects under the Transform Handles prefab. As an example, box selection is handled by the GameObject called `BoxSelection`, whereas the rotation tool is controlled by the GameObject `RotationHandle`. As previously said, these are all children under `TransformHandles` GameObject.

Some of the components are activated in runtime, such as the SceneGizmo, which mimics the camera widget on the top right of the scene window in Unity Editor. The user can click on different axes on the widget to align the camera to that axis, click on one surface of the widget to align the camera to that respective 2-axis plane. For example, clicking the Y axis on this widget shifts the camera to an Euler rotation of 90 degrees in the X axis. This emulates the bird's eye view where the camera is looking from top down to the X-Z plane.

The camera can even be toggled between perspective and orthographic projections, which can be handy for some specific authoring cases.

```
protected override void Awake()
{
    base.Awake();
    _currentAppState = ApplicationState.Editor;
    IOC.Register<IRTEState>(this);
    m_resourcePreview =
        gameObject.AddComponent<ResourcePreviewUtility>();
    IOC.Register<IResourcePreviewUtility>(m_resourcePreview);
    _eventController = FindObjectOfType<EventController>();
    _rthManager = FindObjectOfType<RuntimeHierarchyManager>();
}

protected override void Start()
{
    base.Start();
    Editor.IsOpened = true;
    Editor.IsPlaying = false;
    OnPlaymodeStateChanged();
    Editor.PlaymodeStateChanged += OnPlaymodeStateChanged;
    Editor.Selection.SelectionChanged += OnSelectionChanged;
    _eventController.onFocusClick.AddListener(OnFocusClick);
    m_editorCamera.transform.position =  new Vector3 (7, 9, -7);
    m_editorCamera.transform.eulerAngles = new Vector3(45, -45, 0);
    _displayingBounds = false;
}
```

Listing 4.1: After the application's launch, the main scene starts up using the functions above.

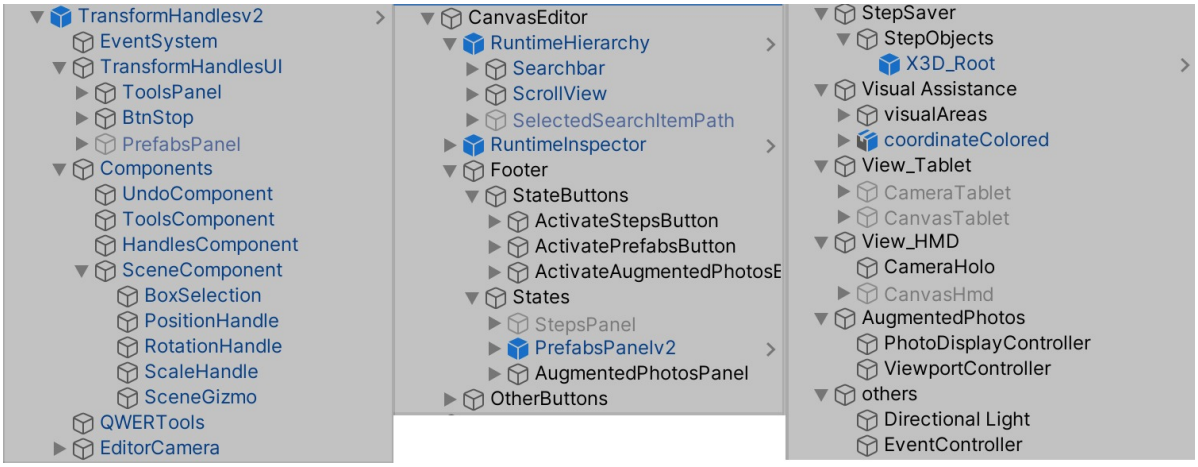

Figure 4.3: The complete scene hierarchy.

4.3.3 Runtime Hierarchy and Inspector

Next up on the list is the asset Runtime Hierarchy and Inspector. This asset is controlled by two prefabs respectively: `RuntimeHierarchy` and `RuntimeInspector` (see figure 4.3). These two components are connected with each other and emulate the hierarchy and the inspector from Unity. When the selection in the hierarchy changes, the inspector reflects on that change and displays the inspector window for that selected element.

The hierarchy costs 1 SetPass call and around 5 batches, meanwhile the inspector costs 1 SetPass call and around 10 batches. Both panels are heavily optimized to prevent unnecessary garbage collection. This asset also includes a color picker and a reference picker, which are instantiated at runtime and toggled whenever the user wants to change a reference to an entity like a material or a mesh, or change the color of an object.

The inspector displays all the Unity components the currently selected GameObject has. However, these components can be filtered out to prevent the user from changing component values that he is not supposed to, or just to avoid visual clutter. The author can even drag and drop objects from the hierarchy to the variables in the inspector to assign these objects to those variables.

The hierarchy can display the whole Unity scene hierarchy at will, or just list a portion of it. This is done by instantiating "pseudo-scenes". The objects that are going to be displayed in runtime hierarchy must be added into the pseudo-scene through scripting.

Since both of these panels are automatically synced, it is enough to just manage one of them via scripts.

4.3.4 Runtime Hierarchy Manager and EditorAndHierarchyConnector

Runtime Hierarchy's logic is handled by `RuntimeHierarchyManager.cs` which is attached to the prefab `RuntimeHierarchy`. This script controls the instantiation of X3D files and what will be displayed in the hierarchy depending on the application state.

When the author clicks on the "Open X3D" button, the function `LoadX3DClicked()` is called (see listing 4.2). Through the file dialog, the file path to the X3D file is received. If any file was loaded previously, the corresponding GameObjects will be deleted. Next, the X3D file is imported through `LoadX3DFile()`. Here, a new GameObject is created, and this is used as the root for the rest of the X3D visualization. The X3D factory utility from X3DModule is put to use here as the factory object handles the parsing of the .x3d file and converts the hierarchy of this file into a transform hierarchy under the root GameObject.

In the X3D file conversion, GameObjects are only added to leaf nodes of the X3D hierarchy tree, meanwhile other nodes are mainly used for structure. To enable interactibility with the GameObjects, each leaf node is found recursively and stored in a list called `List<GameObject> _x3DLeafObjects`. Then, leaf nodes are added with the component `ExposeToEditor`, which was explained in section 4.3.2. This way, all GameObjects included in the X3D are accessible by the editor functionality of Runtime Transform Handles.

```
1  public void LoadX3DClicked()
2  {
3      var paths = StandaloneFileBrowser.OpenFilePanel("Choose your X3D
           file", Application.streamingAssetsPath, "x3d", false);
4      if (paths.Length > 0 && paths[0].Length > 0)
5      {
6          if (X3DRootObject.transform.childCount > 0)
7          {
8              _x3DLeafObjects.Clear();
9              foreach (Transform child in X3DRootObject.transform)
10             {
11                 Destroy(child.gameObject);
12             }
13         }
14         var path = new System.Uri(paths[0]).AbsoluteUri;
15         LoadX3DFile(X3DRootObject, path);
16         AddX3DDataToRuntimeHierarchy();
17         GetLeafObjectsRecursive(X3DRootObject.transform.GetChild(0));
18         AddInteractabilityToObjects(_x3DLeafObjects);
19         FocusCameraOnSelectedObject();
20         _eventController.onObjectBoundsChanged.Invoke
21         (_hierarchy.CurrentSelection.gameObject.CalculateBounds());
22         _eventController.onDocumentLoaded.Invoke();
23     }
24 }
25
26 private void LoadX3DFile(GameObject parentGo, string scenarioPath)
27 {
28     var root = new GameObject {name = Path.GetFileName(scenarioPath)};
29     root.transform.SetParent(parentGo.transform);
30     string[] separator = {"file://"};
31     var curbedPath = scenarioPath.Split(separator, 2,
```

```
        StringSplitOptions.RemoveEmptyEntries);
32      Stream StreamFactory() => File.OpenRead(curbedPath[0]);
33      IX3DConfiguration config = new CAPX3DSingleThreadConfiguration();
34      var x3dFactory = CreateFactory(config);
35      x3dFactory.Load(StreamFactory, root);
36  }
```

Listing 4.2: The logic for importing X3D files into the scene.

The other problem that needs to be tackled is the synchronization of Editor and Hierarchy assets. Both of the components needs to be aware of the other one's object selection logic and notify the other one if there is a change.

The connection between those entities is handled by `EditorAndHierarchyConnector.cs`. Both assets have a delegate function which can be listened to if the current selection changes. The script subscribes to these events and handles each one separately. When the selection in the hierarchy changes, the function `OnSelectionHierarchyChanged()` invoked, meanwhile any change of selection in the editor triggers `OnSelectionEditorChanged()`. To avoid these functions from firing at the same time, a `bool` called `_lock` is introduced. If there is no lock, the function sets `_lock` to `true` and only sets it back into `false` at the end of the function, after all the logic is done. If there is a lock, the function does not continue.

`OnSelectionHierarchyChanged(Transform trs)` is a simple function that checks if `trs` is `null` or not and updates the editor selection accordingly. Its counterpart, `OnSelectionHierarchyChanged()`, is more complicated since the editor can have more than one object selected, while only one can be displayed as selected in the hierarchy. This function first syncs deselection with the hierarchy, if any objects were deselected in the editor. If one object is selected, the selection is reflected to the hierarchy. If more than one object was selected, the transform with the lowest tree depth is selected in the hierarchy instead.

"Select Parent" button's logic is also in this script. When clicked, the root transform of the current selected GameObject is retrieved and it is selected both in the editor and hierarchy.

4.3.5 Footer Features

Three features are implemented into the "Footer", the UI at the lower part of the screen. It has three tabs for Scenario Steps, Object Library and Augmented Photos respectively. Changing between the tabs enables the UI connected to that tab.

The footer logic is handled by `FooterController.cs`. Each tab is a button that activates its respective UI and deactivates all others. The activated tab is highlighted so it is more clear to the user.

Next, two tabs that are implemented (object library and scenario steps) are discussed. The other tab (Augmented Photos) is explained under section 4.4.2.

Object Library

One of the main goals of the authoring tool is to allow adding and editing new objects into the library. A footer tab is reserved exclusively for this feature. The main functionality is

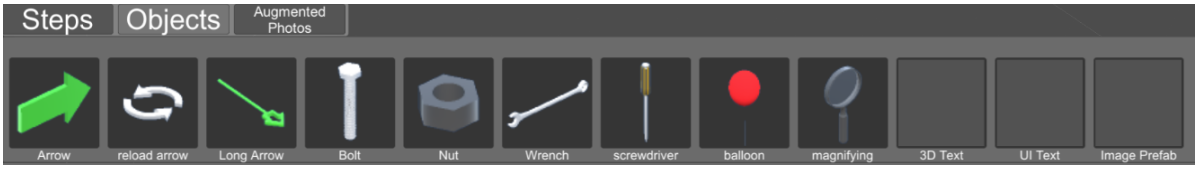

Figure 4.4: The object library found in the footer.

taken from Runtime Transform Handles, where the logic for dragging and dropping prefabs was already set. This is handled in the `PrefabSpawnPoint.cs`.

When the user clicks on one of the objects, a new GameObject is instantiated from the prefab on the location of the cursor. While the user is still dragging, the GameObject is translated on the X-Z axis while avoiding collision with other object in the scene. When the dragging ends, the object is placed on that spot, given interactability through `ExposeToEditor` and added into the hierarchy, under the currently active pseudo-scene.

Since the authoring tool will mainly be used for industrial content, a selection of 3D objects are added which can be beneficial for content creation. These consist of bolt, nut, wrench, screwdriver and magnifying glass objects. A balloon GameObject is also added for the demo. Other than these, three arrow objects, as well as two different text prefabs and an image prefab is ready to use as well.

The concept of inter-steps is already explained at the start of this chapter. The logic here is to guide the user to a position with the help of visual indicators. This can be achieved by creating a new step and inserting arrow objects into the scene. Text and image prefabs can also be additionally good for this matter. Visual cues like a movement widget or a text giving out instructions can be beneficial in assisting the user with his position and orientation in the environment during tracking.

The difference between the two text prefabs is how they react to the camera's position. This is commonly stated under the term billboarding. What it means is an object changing its orientation depending on the position and the orientation of the camera and the affecting coordinate planes. Here, it is used in two ways. The first way is from MRTK, is used on 3D Text Prefab and allows the user to select the pivot axis of the billboarding functionality. By default, billboarding is pivoted from the X and Y axes. This creates a effect that the text is on 3D space, as it is also three dimensional and can change it's rotation on any axis.

The second prefab, called UI Text prefab, acts as a 2D text on the scene, while always directed in the direction of the camera. This creates the effect that the text is attached to the screen view and seems like an 2D UI object, when in reality it is also positioned in 3D space.

The image prefab has the same billboarding effect with the 2D Text prefab. The user can change the source image from the inspector and insert any image that is included in the project library.

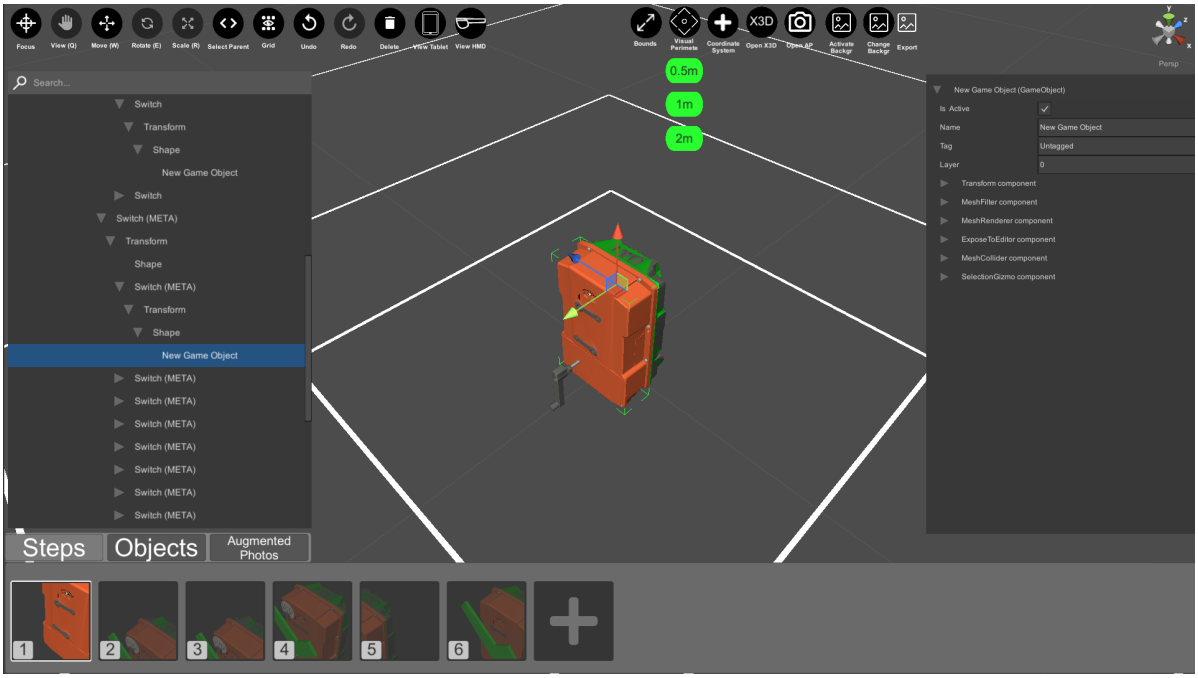

Figure 4.5: A screenshot from the authoring tool.

Scenario Steps

Another important assessment in the research section is that step-by-step instructions are the main usage of augmented reality scenarios. This is also the approach in RE'FLEKT ONE and RapidManual. The step information is also preserved in the X3D files that are outputted from RapidManual. This information can be accessed in Unity on the first child of the X3D object, in the component named `CAPDocumentController`. Along with step information, additional data like filters, display mode and additional details can also be found here.

In ONE, there are two types of scenarios, which are Animation and POI (Point of Interest) scenarios. Animation scenarios consist of animation steps and have a different animation assigned to each step. These steps coincide with scenario steps. In POI scenarios, there is only one step. The user can interact with different parts of the object, which might have text, image or other materials attached to them.

This duality also translates into the X3D configuration when it is imported. `CAPDocumentController.cs` has a variable called `public AnimatorBehaviour Animator`, which is instantiated only if the X3D is an animation scenario. If that is not the case, this variable is not instantiated at all.

After the document is loaded, `ScenarioStepManager.cs`, which controls the scenario step creation and management, checks whether if the `Animator` variable is `null`. If `Animator` is present, one thumbnail for each step is instantiated. These thumbnails are visual representations for steps with on click functionality added to them. For their `OnClick()` event, each button activates the corresponding animation step. An example of this can be seen in figure

4.5, on the bottom part of the screen.

When a new step is activated, a screenshot of that current step is recorded internally and the screenshot is assigned as the texture for the thumbnail. This way, authors can quickly identify the step differences by looking into the thumbnails. The camera is focused on the current selected object before a screenshot is taken.

Screenshot functionality is taken care of by `ScreenshotHandler.cs`, which is attached to the main camera. The script has a public function which, when accessed, calls on the internal function for taking a screenshot. The texture after the last camera frame is saved into a .png file for further use. When the function is complete, a new event is fired indicating that the screenshot is saved and ready to use. `ScenarioStepManager.cs`, which is listening on the event, assigns the screenshot as the activated thumbnail's texture and enables the outline of the thumbnail. This way, user can know which step is active at that time.

The last step index is saved into a private variable, so that it's thumbnail outline texture and the scene objects connected to that step can be deactivated as well.

At the end of the step list, a new thumbnail with a plus icon is present as well. This thumbnail is for adding new step buttons, as clicking this button copies the current scene state into a new step, adds a new animation step into X3DDocument's `Animator` and activates that step, also intanstiating a new step thumbnail. The new step thumbnail is then positioned between the last step thumbnail and this thumbnail. The implementation for the step buttons can be found in listing 4.3.

The explanations in the previous paragraphs were taking an animation scenario into account. When it comes to a POI scenario, a new `Animator` has to be created and the current scene state has to be saved as the first animation step. This only occurs when the user wants to add a new step to a POI scenario.

```
1  private void InstantiateStepButtons()
2  {
3      var stepButtonOriginal = transform.GetChild(1);
4      for (var i = 0; i < _stepCount; i++)
5      {
6          var stepButtonCopy = InitializeButton(stepButtonOriginal, i,
               _buttonName, stepButtons, this.transform);
7          var i1 = i;
8          AddButtonFunctionality(stepButtonCopy, i1);
9      }
10
11     //insert AddNewsStepButton at last with different functionality
12     var stepButtonAdd = Instantiate(stepButtonOriginal, transform);
13     stepButtonAdd.name = "AddNewStep";
14     stepButtonAdd.GetChild(0).gameObject.SetActive(true);
15     stepButtonAdd.GetChild(1).gameObject.SetActive(false);
16     stepButtonAdd.GetComponent<Button>().onClick.AddListener(() =>
17         {
18             AddNewStep(stepButtons.Count);
```

```
19          });
20      stepButtonAdd.transform.GetChild(0).GetComponent<RawImage>().
21      texture = _plusIcon;
22      stepButtonOriginal.gameObject.SetActive(false);
23  }
24
25  private void AddButtonFunctionality(Transform button, int index)
26  {
27      button.GetComponent<Button>().onClick.AddListener(() =>
28      {
29          if (_lastActiveStepIndex != index)
30          {
31              SaveStepObjects(_lastActiveStepIndex);
32              if (_scenarioHasAnimator)
33              {
34                  X3DDocument.Animator?.Play(index);
35              }
36              LoadStepObjects(index);
37              _lastActiveStepIndex = index;
38          }
39          _eventController.onFocusClick.Invoke();
40          _eventController.onScenarioStepActivated.
41          Invoke(_lastActiveStepIndex);
42          _screenshotHandler.TakeScreenShot(index);
43      });
44  }
```

Listing 4.3: The code for instantiating thumbnails for each scenario step and the additional button for adding new steps.

Saving the scene information for different scenario steps was also another matter of concern. Originally, the imported X3D file was stored under one root GameObject, which changed depending on the animation step. But this did not have the same effect in POI scenarios. Additionally, other objects that are added into the scene also did not behave in relation to their respective steps when the active scenario step was changed.

The solution is to create a StepObject for each scenario step and add a root GameObject to each step respectively, which is also the parent for all the objects that are active in the scene for that scenario. This behaviour is handled by `RuntimeHierarchyManager.cs`, which listens to the event `onScenarioStepActivated()` for this matter. The related GameObjects in the hierarchy are the StepSaver and it's children. Each children holds the scene objects connected to that step, under the X3D root object. The GameObjects in question can be observed in figure 4.3.

After a step change, the current pseudo-scene is deleted and a new one is created with a name containing the scenario step index. After that, a new pseudo-scene containing the new step's objects is created. Since the last active GameObject is also deactivated, the hierarchy's current selection changes to the root of the active X3DObject in the scene (see listing 4.4). The hierarchy is refreshed between these operations to process each step correctly.

When the scenario step is changed, the last state of the previous step is saved and deactivated in the hierarchy, while the GameObjects of the current step will be activated in `ScenarioStepManager`. If a new step is created, the last active step's scene objects, which are stored in a parent container, are duplicated and added as the new step's objects. This procedure ensures that new steps carry on from the last active step and simplifies the authoring process. Since authors do not have to start over with each step, they can continue from their last scene state and finish their tasks more quickly.

```
1  void Start()
2  {
3      /**/
4      _eventController.onScenarioStepActivated.AddListener
5      (OnScenarioStepActivated);
6      /**/
7  }
8
9  private void OnScenarioStepActivated(int stepIndex)
10 {
11     _hierarchy.DeletePseudoScene(sceneName);
12     _hierarchy.Refresh();
13     sceneName = "ScenarioStep" + (stepIndex + 1);
14     _hierarchy.CreatePseudoScene(sceneName);
15     _hierarchy.Refresh();
16     ActiveX3DObject = AuthoringToolUtilities.GetLastChildAsGameObject
17     (_stepManager.stepObjectContainerList[stepIndex]);
18     _hierarchy.AddToPseudoScene(sceneName, ActiveX3DObject.transform);
19     _hierarchy.Refresh();
20     _hierarchy.Select(ActiveX3DObject.transform);
21     _hierarchy.Refresh();
22 }
```

Listing 4.4: The procedure of changing pseudo-scenes after a scenario step has been activated, in RuntimeHierarchyManager.cs

4.4 Augmented Reality Additions

Up until this section, discussed features were mostly related to editor utilities. The building blocks of most of the editors/authoring tools were implemented into the application. This next part is about the features that are mostly uncommon and enhance the authoring process specifically for augmented reality applications.

Following the goal of creating a tool that benefits from AR, the main focus of the thesis was always the AR specific authoring techniques and additions rather than building the editor itself. In this section, 5 features are demonstrated which eases the authoring procedure for augmented reality.

4.4.1 Preset Backgrounds

The preset backgrounds add an additional layer of realism to the authoring. Normally, what the author sees in the editor is a virtual environment, but having the option to toggle background presets and being able to import additional background images bring a lot of usability.

There are two buttons associated with this feature, called "Toggle Background" and "Change Background Image". The first button enables and disables the background image. Having a simple toggle button enables the author to change modes swiftly. The second button opens a Explorer dialog with the help of StandaloneFileBrowser. The user can choose any file with a .jpg extension as input. The next time background is toggled on, the new selected image will be resized and fit onto the screen. This behaviour can be examined in listing 4.5.

```
1  public void ToggleBackgroundClicked()
2  {
3      var canvas = editorCam.transform.GetChild(0).gameObject;
4      canvas.SetActive(!canvas.activeSelf);
5  }
6
7  public void ChangeBackgroundClicked()
8  {
9      var backgroundImagesPath = Application.dataPath +
           "/AuthoringTool/Resources/Background";
10     var paths = StandaloneFileBrowser.OpenFilePanel("Choose your
           background file", backgroundImagesPath, "jpg", false);
11     if (paths.Length > 0 )
12     {
13         var path = new System.Uri(paths[0]).AbsoluteUri;
14         string[] separator = {"Resources/", ".jpg"};
15         var curbedPath = path.Split(separator, 3,
               StringSplitOptions.RemoveEmptyEntries);
16         Debug.Log("into the resources: " + curbedPath[1]);
17         var texture = Resources.Load(curbedPath[1]) as Texture2D;
18         if (texture != null)
19         {
20             Sprite sprite = Sprite.Create(texture, new Rect(0, 0,
                   texture.width, texture.height), new Vector2(0.5f,
                   0.5f));
21             var targetImage =
                   editorCam.GetComponentInChildren<Image>();
22             targetImage.sprite = sprite;
23         }
24     }
25     else
26     {
27         throw new Exception("could not find image!");
28     }
```

```
29 }
```

Listing 4.5: Two functions that are attached to preset background buttons. It also demonstrates the plugin StandaloneFileBrowser.

4.4.2 Augmented Photos

As previously said, Augmented Photos implementation is in its own folder, which is a mixture between previous Augmented Photo implementations. The center of the implementation is the Augmented Photo structure itself: These are instantiated in RE'FLEKT ONE when the user takes a photo of the tracked object. In ONE, the tracked object is always in the center of the scene while it is the camera that changes its position.

When the photo is "taken", the camera's translation and rotation are saved along with the background image from the camera feed. The photo id, camera translation and rotation are stored in the name of the background image, which is saved as a .png file. An example would be so: The image with the name `0002#0.010,0.560,-0.506,0.356,-0.008,0.008,0.934.png` corresponds to an photo id of 2, a translation of `Vector3(0.010f, 0.56f, -0.506)` and a rotation of `Quaternion(0.356f, -0.008f, 0.008f, 0.934f)`. Although there was an updated notation which saved the photo information under a JSON file instead of in the name, this approach was selected because of its simplicity.

It is crucial that the edited X3D and the Augmented Photos align. When taking augmented photos, the credibility of the solution comes from the tracking itself. If the tracked object used for Augmented Photos and the object in the editor do not align, the scene does not make much of a sense and the user can not get much information from the feature. A good alignment can be see in figure 4.6.

Augmented Photos feature is managed by the script called `AugmentedPhotosFeatureController.cs`. The user can load a folder full of augmented photos using the "Open AP" button on the top right of the screen. This button opens a file dialog to select the augmented photos folder. Afterwards, .png files are read from the folder and then for each one, an augmented photo will be instantiated. Next, thumbnails for each loaded photo will be instantiated on the Augmented Photos tab in the footer. The first photo will be activated and displayed (see listing 4.6).

Finally, an event is fired which indicates that all augmented photos are successfully loaded. `PhotoDisplayController.cs`, which is listening on the event, initializes a `PhotoContainer` prefab, assigns the photo and activates the object. `PhotoContainer` is a parent prefab, which has its own cameras: One for rendering the objects and the other for rendering the background texture which is the photo texture. It's script `PhotoContainer.cs` creates a quad mesh, sets it's size to fit the screen and loads the texture from the .png file onto it. It also applies the rotation and translation from the Augmented Photo to the camera. The main camera properties are also copied into the first camera for consistency. Use of layer masks for augmented photos creates the illusion that the photo is augmented, as the user can interact with the content on top of a real world background.

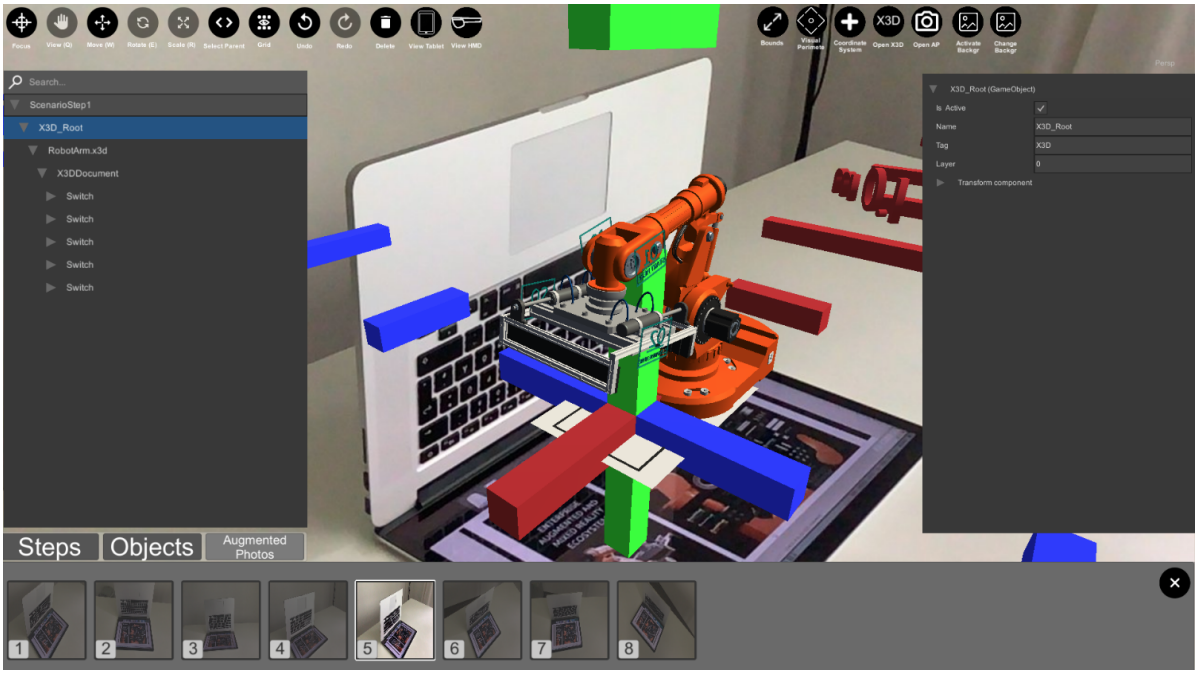

Figure 4.6: Augmented photos in use in the authoring tool. Here, the photos are taken from a marker tracking scenario, which is display on the Mac screen in this case. It can be seen how the photo and the virtual object aligns ideally. Out of the 8 photos, the fifth is displayed currently.

```
1  public void LoadAugmentedPhotosFromFolder()
2  {
3      var paths = StandaloneFileBrowser.OpenFolderPanel("Choose the
           Augmented Photo Folder", CurrentFolderPath, false);
4      if (paths.Length > 0 && paths[0].Length > 0)
5      {
6          if (AugmentedPhotos != null && AugmentedPhotos.Count > 0)
7          {
8              ClearPreviousButtons(Thumbnails);
9              AugmentedPhotos.Clear();
10         }
11         CurrentFolderPath = paths[0];
12         ReadAllPhotos(CurrentFolderPath);
13         InstantiateAugmentedPhotoThumbnails();
14         _eventController.onAugmentedPhotoFolderLoaded.Invoke();
15     }
16 }
17
18 /// <summary>
19 /// Reads all augmented photos under the folder rootPath.
20 /// </summary>
```

```
21 private void ReadAllPhotos(string path)
22 {
23     AugmentedPhotos = new List<AugmentedPhoto>();
24     string[] files = Directory.GetFiles(path);
25     foreach (string file in files)
26     {
27         if (!(file.EndsWith("jpg") || file.EndsWith("png")))
28         {
29             continue;
30         }
31         AugmentedPhoto newAugmentedPhoto = new AugmentedPhoto(file);
32         AugmentedPhotos.Add(newAugmentedPhoto);
33     }
34 }
```

Listing 4.6: The code handling the augmented photos initialization.

4.4.3 Bounds Visualization

The importance of giving the user accurate information on scene objects was stated previously. Previously discussed solutions to this problem are object library elements for size comparison and the grid on the plane. Three new additions for visualizing the size and bounds of objects will be demonstrated in this chapter. One way is to extract size information is to display the selected object's bounds. The other two ways are to insert objects that are only visual cues themselves, a visual perimeter and a coordinate system, respectively.

Showing Bounds Information

There is a button present in the header, called "Show Bounds". When clicked, it invokes `DisplayBoundsClicked()` function, which in turn calculates the object bounds through the object extension method `CalculateBounds()` and assigns the values to the text fields on the UI. `CalculateBounds()` makes the calculation using Unity core functions.

After the buttons is clicked, each dimension of the object, which can be accessed by `bounds.size`, is rounded first, then assigned to the `Text` element from UnityUI. The related code snippet is listing 4.7.

```
1  public void DisplayBoundsClicked(Text text)
2      {
3          var selected = _hierarchy.CurrentSelection.gameObject;
4          var bounds = selected.CalculateBounds();
5          text.text = "X: " + Math.Round(bounds.size.x,3) + "\nY: " +
               Math.Round(bounds.size.y,3) + "\nZ: " +
               Math.Round(bounds.size.z,3);
6      }
7
```

```
8  public static Bounds CalculateBounds(this GameObject g)
9      {
10         Transform t = g.transform;
11         Renderer renderer = t.GetComponentInChildren<Renderer>(true);
12         if (renderer)
13         {
14             Bounds bounds = renderer.bounds;
15             if (bounds.size == Vector3.zero && bounds.center !=
                   renderer.transform.position)
16             {
17                 bounds =
                       TransformBounds(renderer.transform.localToWorldMatrix,
18                 bounds);
19             }
20             CalculateBounds(t, ref bounds);
21             if (bounds.extents == Vector3.zero)
22             {
23                 bounds.extents = new Vector3(0.5f, 0.5f, 0.5f);
24             }
25             return bounds;
26         }
27         return new Bounds(t.position, new Vector3(0.5f, 0.5f, 0.5f));
28     }
```

Listing 4.7: The process of calculating and displaying bounds.

Visual Area Perimeter

The second method actually stems from the idea of inserting known objects into the scene to display visual information. These objects can be varying: From a table to a tracking marker or to a machinery known to the users. Object library helps with this approach, but these objects still give information indirectly. The users still can have a hard time figuring out the exact bounds of the object.

To mitigate this, one concept is to create a measured perimeter around the world origin. This perimeter is in shape of a square, is on the X-Z plane and is repeated every 0.5, 1 and 2 meters. Unity's world scaling is correspondent to real life, which means an object that is 1 meter high, 1 meter wide and 1 meter deep has the exact same dimensions in an Unity scene. This meshes well with the authoring procedure, since these objects are going to be relevant in size in an AR app.

This feature will not be active when the program starts. To activate it, the button "Visual Perimeter" is used, which toggles the feature. When enabled, these perimeter lines will be visible and each of them can be toggled through the additional UI responsible for each of the lines. A representation of the feature can be seen in figure 4.7.

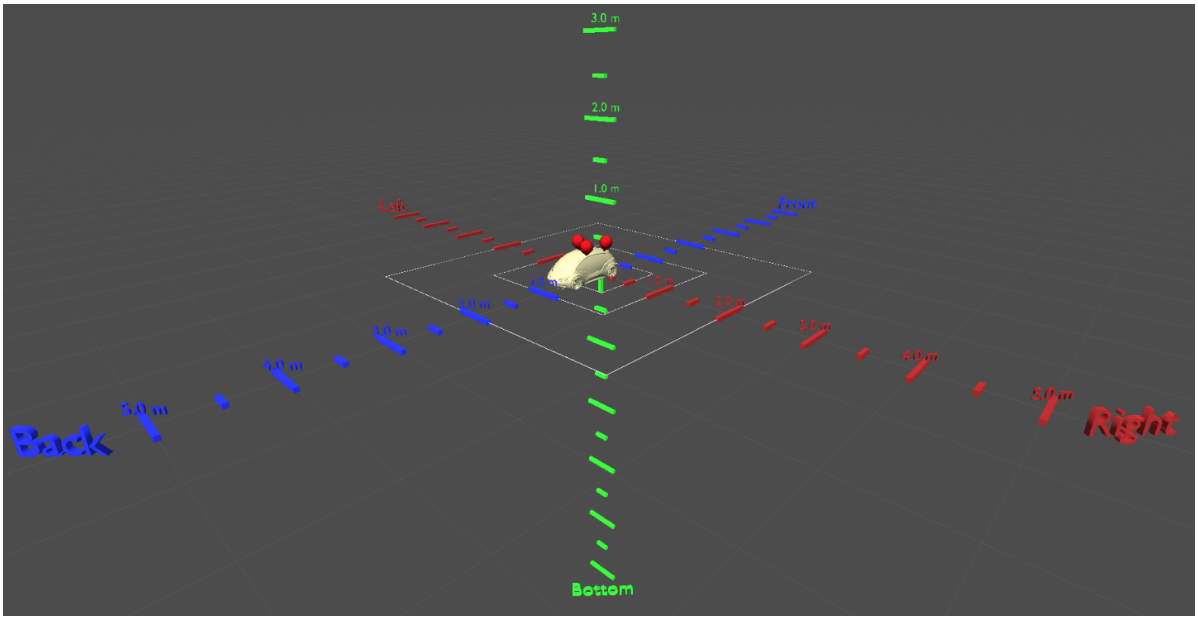

Figure 4.7: The visual perimeter (in white colors) and the coordinate system used with a car model.

Coordinate System

The third option is similar to the prior in terms of execution. Again, we want to make additions that give visual cues and size information for the object. An enlarged coordinate system, suited for industrial tracking objects and vehicles is the choice here.

Unlike the visual area, which only covers the ground, the coordinate system spans in each axis direction for 10 meters. So in total, it covers a volume of 1000 m^3. This area should be more than sufficient for any object in question.

For the color selection, the standard Unity colors for up, forward and right vectors are used. These are green, blue and red respectively. To adjust the user orientation easily, 6 directions are additionally labeled as Left, Right, Up, Down, Front and Back. This can be clearly seen in figure 4.7.

When the authoring tool starts, the active camera automatically adjusts to the coordinate system and positions itself to see the guidance labels in the right angle. This corresponds to an Euler rotation of 45 degrees around the X axis and -45 degrees around the Y axis.

4.4.4 Preview Modes

As stated in section 4.2.2, preview mode will be divided into two parts. The first one is handheld or tablet preview, while the second one is HMD preview. For the first one, the latest versions of iOS and Android tablet devices will be used as foundations. Smartphones will not be taken into account, since ONE is mostly used with tablets. For the HMD emulation, HoloLens and HoloLens are selected, again, due to heavy usage of these hardware in the

company.

In the project, only one scene is used. Although Unity's multi scene editing was tried out, it made more sense from an architecture perspective to separate preview elements into a parent GameObject that can be activated through UI logic. Moreover, the different states of the scene is represented as an public enum called the ApplicationState. It has three states: Editor, Tablet and HMD.

The preview modes can be directly accessed from the header buttons "View Tablet" and "View HMD", which are controlled by the script `MainSceneEditor.cs`. When pressed, the event that is linked to that button is invoked, resulting in the application state change. The camera state of the editor is saved as well, so when the author returns to the editor he/she can continue working from the same viewport. An example of this can be seen in listing 4.8. To exit the preview mode, user has to press the "X" button on the up right corner of the screen.

The two preview modes will be explained in detail as follows.

Handheld Device Emulation

RE'FLEKT ONE is mostly used with Android or iOS devices, which makes this feature vital. The intention here is to create a visualization of the tablet client to help with the authoring. For this, the outcome should reflect elements of the tablet client like the UI and the resolution.

The tablet client has two modes: AR and VR. VR mode represents the virtual content in an environment similar to the editor. AR modes also renders the real world feed and is the more used mode overall. For this reason, recreating that experience would add a significant immersiveness to the emulation.

Preset Backgrounds feature was already showcased in section 4.4.1. This feature is again utilized in here to have a realistic background image for the tablet. While tablet preview mode is active, the preset background is always seen and cannot be cancelled. The repeating issue of object and background orientation mismatch can happen here as well, depending on the picture angle.

The author can manipulate the object as if in the editor minus some of the capabilities. In ONE tablet client the user can move, rotate and scale the object, so these interactions are kept intact in this mode. Other interactions are disabled.

For the ONE UI, a screenshot of the main screen UI is used here only as an image, so the UI does not have functionality. It shows the regular scenario main screen UI plus the Augmented Photos menu. Since various devices with different resolutions can run ONE, there is an option to change the resolution settings between preset values. This widget can be found on the top right corner. For the preset values, the latest iOS and Android devices were used [88]. These values are 1024x768(iPad Mini), 2048x1536(iPad, Samsung Galaxy Tab S3), 2224x1668(iPad Pro), 2560x1600(Samsung Galaxy Tab S4) and 1920x1080(Editor Resolution).

`ResolutionManager.cs` is responsible for changing the resolution. At initialization, a `Vector2` array is created and filled up with the resolutions above. When the resolution is changed through the arrows in the widget (visible in figure 4.8), this class changes the screen

Figure 4.8: A screenshot showcasing the Tablet Emulation.

width and height accordingly. The UI elements are scaled with the resolution to adapt to the new size.

```
1  private void ToggleTabletView(bool active)
2      {
3          //deactivate
4          Editor.Selection.activeGameObject = null;
5          m_uiTools.SetActive(!active);
6          sceneGizmo.SetActive(!active);
7          canvasEditor.SetActive(!active);
8          //activate
9          m_stop.gameObject.SetActive(active);
10         canvasTablet.SetActive(active);
11         ToggleTabletBackground(m_editorCamera.GetComponent<Camera>(),
               active);
12
13         if (active)
14         {
15             SaveEditorCameraPose();
16         }
17         else
18         {
19             RestoreEditorCameraPose();
20         }
```

```
21      }
```

Listing 4.8: This function toggles between tablet emulation and the editor mode. If the parameter is true, it deactivates editor related UI and GameObjects, whereas activating the GameObjects related to the tablet preview mode.

Head Mounted Device (HMD) Emulation

The aim for this feature is to create a possible emulation for a HMD usage. The devices used for steady development in RE'FLEKT ONE are HoloLens and HoloLens 2, so the focus will be shifted on those two.

Prior to the implementation, the main discussion about this feature was to select HMD components that are valuable in two ways: First, components should provide a sensible emulation for the HMD platform. The user should get a feeling of presence and usage of HMD and differentiate the experience from desktop usage to some extent. Second, it should make sense in from an authoring perspective. Features that are added for the emulation but do not bring much authoring value will make the feature bloated and harder to use.

For these purposes, it made the best sense to add these features:

- FPS Camera Controls: When using a HMD, the camera is the user itself. The manipulation of the camera happens when the user moves its head, providing an ubiquitous experience. Trying to recreate that experience makes the most sense in providing a believable emulation.
- HoloLens Screen Filters: When building a Unity project to the HoloLens, the field of view and depth is always mismatched. The author has little to no way of understanding the spatial positions of objects on the HoloLens before deploying their application. Trying to imitate the field of views in a Unity scene will help authors figure the right object position and scale.

FPS Camera Controls

It made the most sense to use a well-known preset for the control scheme, so camera controls in editor are deactivated, along with the camera itself. In it's place, a new HoloLens camera is activated, which has FPS (First Person Shooter) camera controls. User can move forward, left, back and right with W, A, S and D keys (in the same order). The camera can be moved up and down with Q and E buttons as well. Additionally, the user can look around using the mouse. By pressing shift, the mouse cursor is made invisible and locked in its place to avoid confusion. In this mode, the user can move the camera view through the changes in the horizontal and vertical mouse input per second. When the shift key is not held anymore, the cursor is unlocked and the user can move the cursor freely on the screen. This has to be separated because providing both functionalities at once causes an uncomfortable experience.

In addition, the movement and mouse factors are adjusted depending on the object size. The reason is that the camera and movement speed can be a bit too fast on small objects, while the same can happen on bigger sized objects. The starting position of the camera is also

changed depending on the object size again. This way it is ensured that at the start of the preview mode, the camera faces the object and has a relative distance to it.

HoloLens Screen Filters

Although Microsoft supports development with it's multiple plugins for Unity, it is still hard to transfer an Unity scene to the HoloLens visor, due to the vision changes.

HoloLens is unique with its hardware, having a revolutionizing technology for hologram rendering. However, this comes with costs. With the technology currently available, the HoloLens can only process so much visual data before heating becomes a problem. By limiting the size of the screen, Microsoft was able to limit the amount of data that needed to be processed without losing rendering quality [89]. Instead of opting for a larger display, Microsoft chose smaller displays and hence field of vision in return for higher resolutions, enough to render not pixelated but smooth holograms.

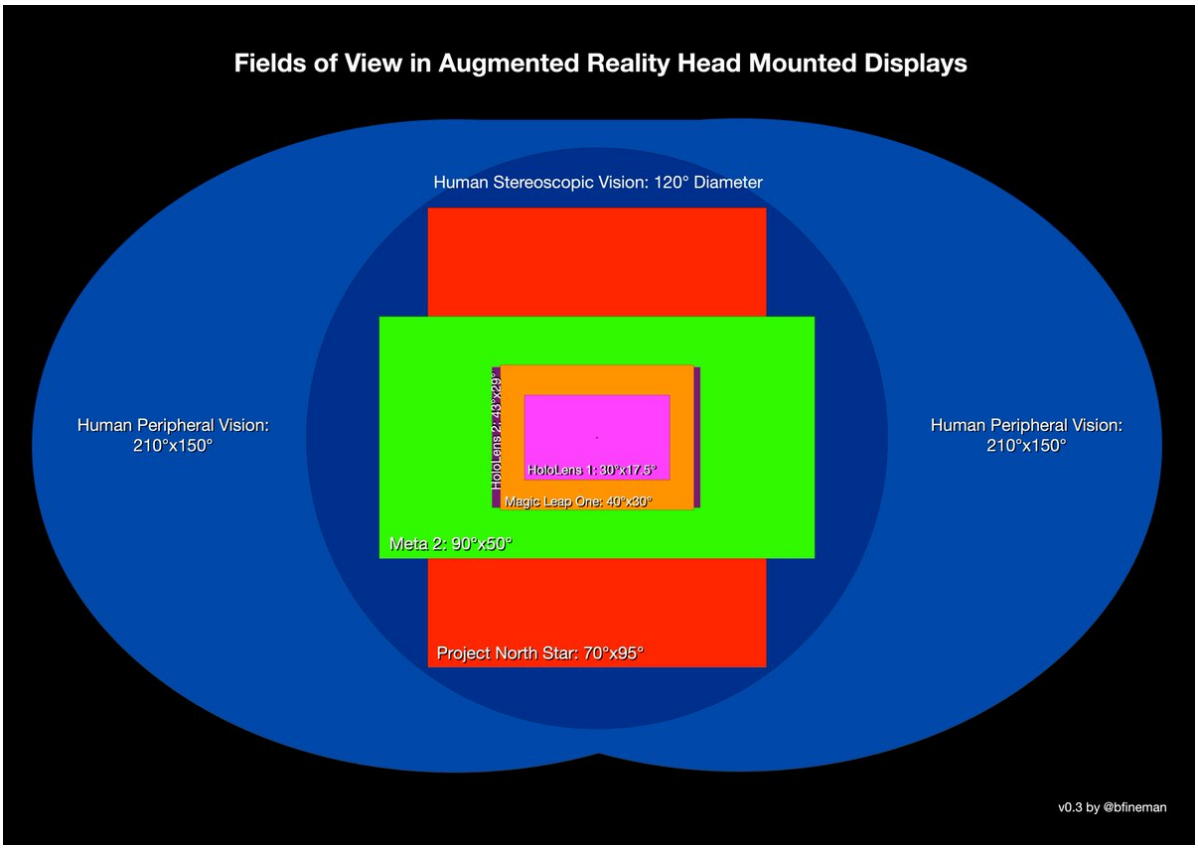

Figure 4.9: A graphic comparing the devices' fields of view in Augmented Reality and Human Vision. Listed from smallest to largest: HoloLens (pink), Magic Leap One (orange), HoloLens 2 (purple), Meta 2 (green) and Project North Star (red). Source: [90]

When working in an environment like Unity, the camera component used in the scene has drastic differences from what the user sees in a HoloLens. This is mainly due to the

display qualities of the headset discussed above. Usually, since the field of view of HoloLens is smaller than the Unity camera, objects seem nearer and bigger when the project is deployed. However, this is caused by the user seeing a smaller, much narrower display of the camera render output. This can be somewhat mitigated by using the recommended camera prefabs and settings from MRTK, but it still does not provide a consistent solution for this matter.

The goal here is to apply a filter on to the screen of the authoring tool, so that how much space the object covers when viewed through the HMD can be understood. For this to work, a semi-transparent filter will be used which can be toggled on an off. This filter will be present for both HoloLens and HoloLens 2 emulations.

The next question is the width, height and space these filters will cover on the original screen of the authoring tool. For this step, we need to calculate the exact visual translations from Unity to the headsets.

The camera used in the thesis as the emulated HoloLens camera has a solid color background, and a near near clipping plane of 0.3 and a far clipping plane of 1000. These are the recommended values from the MRTK, both for HoloLens and HoloLens 2 [91]. The tricky part is to get the FoV right. The FoV axis value in Unity is represented as a single float, and it can be chosen between the horizontal and vertical value. A vertical degree of 60 is the default choice, so the calculations will be carried out based on this value.

The next step is to find the correct FoV values for the headsets in Unity standards. Initially, the field of view values of both HoloLens devices were somewhat vague, given the different explanations from Microsoft before and after the product's release [92]. Right now, the values are exact and can be verified from different sources. The first HoloLens has a horizontal field of view (FoV) of 30°and a vertical field of view of 17.5°. The aspect ratio is 16:9. On the other hand, HoloLens2 has a horizontal FoV of 43°and a vertical of 29°. The headset's aspect ratio is 3:2 [93].

After some research, it was clear that the vertical FoV values that should be used are 16°for HoloLens [94] and 29°for HoloLens 2 [93]. Note that these are the values that translate into the Unity scene camera projection. After that, a test scene was created to verify the object proportions between 60°(Authoring Tool) and 16°(HoloLens) field of views. The same was carried out with 29°for HoloLens 2.

The screen resolution that is used in the authoring tool is 1920 pixels in width and 1080 pixels in height, shortly 1920x1080. Using the proportions between field of views and the aspect ratio of the headsets, it was found that HoloLens display covered an area of 480x270, while HoloLens 2 display covered an area of 864x486, both centered from the middle of the screen. In other words, HoloLens display covered 25% of the authoring tool screen and HoloLens 2 display covered 45% of it.

Finally, filters were created and applied to the scene to verify the calculations. The results of these tests can be found on figures 4.10 and 4.11.

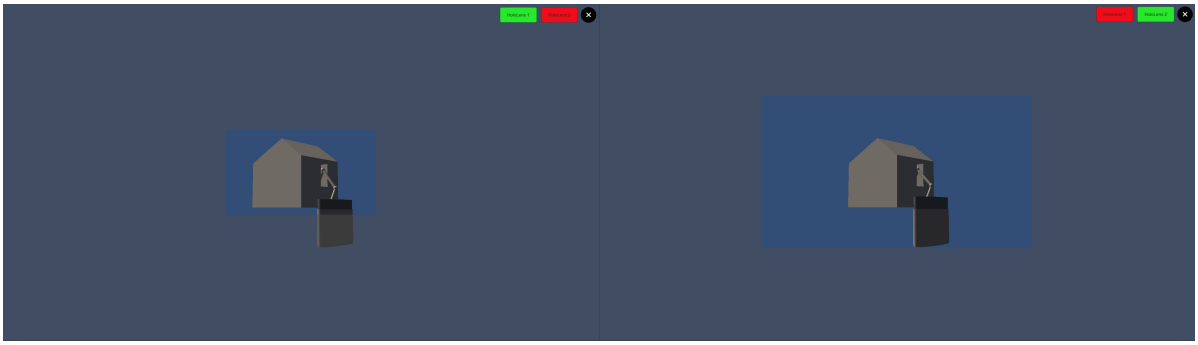

Figure 4.10: A demonstration of the field of view differences between HoloLens and HoloLens 2. Filters are colored grey and have and alpha value of 0.5 (1.0 being full opaque). The size difference here roughly fits the comparison found in figure 4.9. It is clearly visible how much of the authored content is visible compared to the authoring tool view.

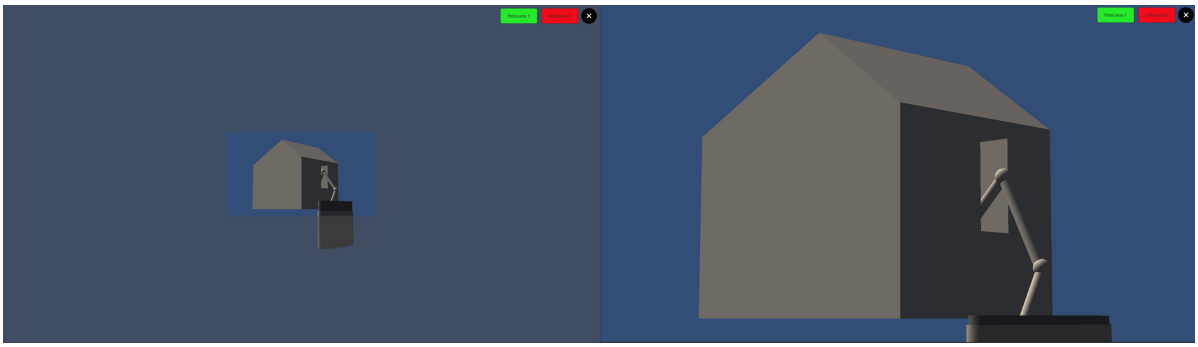

Figure 4.11: These images prove the correctness of the calculation. On the left, the HoloLens filter is applied onto the camera output with a vertical FoV of 60°. On the right, the filter is lifted and the FoV is set to 16°. The small image not filtered on the right image is identical to the image on the left.

5 User Survey

5.1 Introduction

In order to test the usability of the newly created authoring tool, an evaluation is necessary. This is essential since the implementation holds no value if it is approved by just the developer and the supervisors. As a consequence, the flaws of the project can be pinpointed and the project can be improved with the evaluation of results and gathered feedback.

Different approaches are evaluated for the user survey. Since we are evaluating the ease of authoring and the usability the features provide, the demo and the survey had to be built around this. Considering the fact that everyone authors on their own pace, tracking the time for each completed task is not a good option.

The aim here is to roughly measure how much new techniques can be used with authoring. So, the proposed approach is going to be a open interview format, where the user is guided by the moderator through the features and small tasks that they will accomplish throughout the demo. This gives the user much more freedom where they can explore the application on their own with some assistance. At the end of the session, the participants will fill out a questionnaire designed with a Likert Scale, which is a psychometric scale commonly involved in research that employs questionnaires and is the most widely used approach in survey research [95]. The main features and the general functionality of the authoring tool will be graded from a scale of 1 to 5.

5.2 User Survey

5.2.1 Participants

Regarding to the participants, members of UX and R&D teams were selected to test a demo application and then fill a questionnaire. Participants were chosen depending on their availability and their knowledge about RE'FLEKT ONE and the authoring process. This ensures that they are already comfortable with authoring content which leads to less time teaching the controls and the general idea of the application. Furthermore, the feedback is more valuable and participants can use their prior experience and additional references for this matter.

The questionnaire is divided into three parts: The first part evaluates the features added to the authoring tool. The second part goes through the general usability and potential of the program. Finally, the third question is a open one for writing additional feedback and room for improvement.

In total, 7 people took part in the survey with 4 of them being content authors from the UX team. The other two were from R&D who had adequate knowledge of RE'FLEKT ONE and its ecosystem.

5.2.2 Authoring Tool Demo

For the authoring tool demo, a build with an empty scene was created and built. Just like in a normal authoring environment, the users start from scratch. After getting the hang of the camera controls, the user has to import an X3D file to start authoring. This X3D file is interpreted to be coming from RapidManual. Here, two different X3D files were handed out: One is a 3D robot arm, which is used in the training scenarios for RE'FLEKT one. The other one consists of a machine made for the company Amplexor and is added here because it contains animation steps and it will help to showcase how these steps are translated into the authoring tool.

As supporting material, another folder containing Augmented Photos for the robot arm and an additional folder containing background image templates are included as well.

At this stage of the demo, the user is encouraged to explore the hierarchy of the scene, check individual GameObjects and change their properties. The visual assistance features like the visual perimeter and the coordinate system are also shown to them. Next, new steps and adding prefab objects into the scene are demonstrated. Some meaningful content should already be produced by now, so the next features that are demonstrated are tablet preview, HMD preview and finally Augmented Photos. If both parties are available, the session was extended where additional questions were answered and the demo was explored further.

Due to the Coronavirus situation, the user survey could not occur in the office. This had an impact on the pace of the survey since a joint session in the office, in which the features of the authoring tool would be explained, would be much more efficient and easy. Instead of this, an online session with each individual was conducted. A demo folder was provided to each individual containing the application and other resource folders mentioned above. Some people were using Windows and some were using Mac, so two different versions compatible to these operating systems had to be created and tested before the sessions. During the session, the participant shared their screen with me so I could guide them through the tasks and help them at the same time.

During the walk through, any feedback from the participants were saved. This included bug reports, general ideas and ideas for improvement. After the demo, each participant filled the questionnaire, which also had an open question for any feedback.

5.3 User Survey Results

5.3.1 Results

The survey took place in the span of a week and the process was overall successful. Each participant was able to try out the demo with assistance from my side. Generally, the overall reactions were positive and promising for future development.

The results for the first question can be found on table 5.1. The table lists the features in the program and how many participants voted for each option. Each row distributes the voting of the feature into 5 different categories, ranging from "Not Useful" to "Very Useful". The average score is calculated at the last column. Empty cells mean that no one has voted for that option.

The second question's results are demonstrated on table 5.2. Here, the prospect of the authoring tool is evaluated. The format is similar to question one, with one change on grade labels: They are ranked from "very unpromising" to "very promising".

The results from the third question, which is an open question for feedback, is incorporated into section 5.3.2 and chapter 6.

Authoring Tool Features	Not useful	Needs Improvement	Neutral	Useful	Very Useful	Average Score	Percentage
Importing Data			1	2	4	4.43 / 5	88%
Object Selection		1	3	1	2	3.57 / 5	72%
Object Manipulation		1	3		3	3.72 / 5	75%
Camera Controls			3	2	2	3.86 / 5	77%
Adding Objects		1		1	5	4.43 / 5	88%
Visualization Tools				4	3	4.43 / 5	88%
Scenario and Step Integration				4	2	4.15 / 5	83%
Tablet View			2	2	3	4.15 / 5	83%
HMD View		1	1	2	3	4 / 5	80%
Augmented Photos			1	3	3	4.29 / 5	86%

Table 5.1: The questionnaire results from feature evaluation.

5.3.2 Evaluation

It can be said that the majority of the participants were pleased with the results of the authoring tool demo. Almost all of them shared their enthusiasm after the session was done and were willing to give feedback for the next possible implementation steps.

For the feature questionnaire (table 5.1), the most useful features were data imports, visualization tools and adding new objects with an average score of 4.43 over 5. This translates to an 88% satisfaction rate. Augmented Photos feature follows just behind with a score of 4.29 and a rate of 86%. Scenario and step integration, tablet view and HMD view all received good acclaims as well with a rating over 80%.

Camera controls, although easy to use, have received some neutral responses due to some facts. The camera controls were implemented regarding the camera controls in Unity. However, some participants who had little experience with it have struggled to find the shortcuts since they were used to the controls in RapidManual, which are slightly different

Authoring Tool General Rating	Very Unpromising	Unpromising	Neutral	Promising	Very Promising	Average Score	Percentage
Editor Functionality			2	1	4	4.29 / 5	86%
Authoring Tool Features			1	2	4	4.43 / 5	88%
User Interface			4	2	1	3.57 / 5	72%
Usability			3	2	2	3.86 / 5	77%
Pipeline Integration Potential				2	5	4.72 / 5	94%

Table 5.2: The questionnaire results for the general evaluation of the authoring tool.

from Unity's. Some visual aids for the camera movement were also sometimes misleading for the users when tablet or HMD view was active.

The camera controls in HMD mode were also discussed. It was stated that FPS controls may not be familiar to users with an industrial background. Additionally, it was said that a transparent UI showing the camera controls and more adaptation from the MRTK camera components could come in handy.

Object selection and manipulation had the lowest grading, although both being above 70%. During the sessions, some bugs were discovered that prevented users from selecting objects in the editor scene, either by clicking or by box selection. Some of these bugs were found and fixed between the sessions. Adding to that, the hierarchy and inspector readability were also put into question. Since all of the components were visible in the inspector, it caused some clutter. But filtering the components that the user does not need access to in the inspector is a valid solution to this problem. This will possibly implemented in the future.

On some occasions, it was stated that getting the scale information can be hard in tablet or HMD modes. Adding some additional UI here to display the distance between the object and the camera, like a binocular widget or some text, is recommended.

For the hierarchy, the names of the children are directly imported from the X3D document, which do not provide a lot of readability. There are also objects that do not have an GameObject attached and are unnecessary apart from giving hierarchy information. Both of these can be solved in the next iterations of the product with improvements in the X3DModule.

Coming to the second question (table 5.2), it becomes clear on which areas the tool has proven it's potential and which areas need improvement. First of all, the pipeline integration potential has seen an overwhelming 94% rating. The editor functionality and the authoring tool AR features have received a rating of 86%, which is equally impressive.

Next on the list is the usability, which received an average score of 3.86 out of 5, leaving it at 77%. Some of the topics that were pointed out were the lack of changing scenario step order or deleting steps, adding a scene description and some bugs that hindered the usability.

The major critic seems to be on the user interface. A lot of UX designers pointed out rooms for improvement in their feedback, pointing out the dense UI blocking the application viewport, some UX processes being hard to interact and UI scaling issues. These are going to be addressed in an upcoming joint session with the UX team to discuss the future UI changes.

6 Discussion and Future Work

6.1 Discussion

It can be said that the thesis's goals are mostly reached. The expectation from the company was to lay a groundwork for a fully integrated authoring tool into the pipeline. The aim here is to have somewhat of a prototype for enhancing the company's authors by implementing the most useful augmented reality features in an authoring context. Although it still has some flaws as it is pointed out by the user survey, the same survey also shows the enthusiasm and the overall acceptance of such a new tool.

Since augmented reality features were mostly received positively by the study group, it is clear that they are beneficial for authoring and can be regarded as useful implementations. This can also be regarded as a step stone in defining which augmented reality interactions can assist industrial authoring the most.

6.2 Future Work

6.2.1 RE'FLEKT ONE Pipeline Integration

The next steps for RE'FLEKT would be to take the feedback gathered in the user survey and prioritize it into a roadmap of fixes and possible new features.

Given that the user survey pointed out the usability of the features and the potential of the program, the next iteration would be to look for ways to integrate this authoring tool into RE'FLEKT ONE pipeline. A majority of participants finding the pipeline integration potential very promising is also an undeniable factor indicating this approach.

The first improvements should be made on to the user interface, which has the lowest average score from the survey. A lot of feedback was already given about this topic during the sessions, so related improvements will be integrated into the application.

The potential bugs should also be fixed at the same time, along with code refactoring. Some of the aspects of authoring for RE'FLEKT ONE have still not made it into the authoring tool yet: Deleting scenarios, adding scenario descriptions, switching scenario step sequence are some of those. Changing the names of the X3D hierarchy objects to something meaningful is also possible with some tweaking and can increase the readability of the hierarchy immensely.

One important task is to connect X3D's Animator and Unity's Animation logic. X3D comes with it's own animation logic, which covers a bunch of scripts. The animations are available in Unity but they are not connected to Unity's animation components. This means that they are not editable or accessible through UnityEngine's logic. The ability to add and edit animations in authoring is crucial. This aspect should be handled in the next iterations of the product.

The last important topic would be to export the created scene. X3D parsing can be done with the current X3DModule. However, parsing GameObjects back to X3D format is not part of the module. Adding to that, since the next step in the pipeline is RE'FLEKT ONE, which is Unity based, it makes less sense to convert the output into X3D. Creating an Unity package or an Unity asset from the scene data would be a better option. However, I found out that creating packages or assets in runtime is not possible. There is one way which includes opening up a terminal and using the UnityEditor.exe's commands to extract the scene information. Since this approach was not user friendly and because of time restrictions, the research on this matter is postponed.

Another question would be about where to place the authoring tool in the pipeline. One of the possibilities would be integrating this tool into RE'FLEKT SYNC, where tracking parameters are adjusted. Another feature of SYNC could be to polish AR scenarios using the authoring tool, combined with tracking information. The other option would be to turn it into a self-sustaining program what works alongside RapidManual, and if the pipeline evolves in that way, possibly replace that program. Since RapidManual is also used by RE'FLEKT's customers for content creation, the thesis project could also only be used internally.

There were also some ideas left in the mindmap from section 4.2.2. These included nice-to-have features, like adding scenario specific object libraries, the ability to hide components in the viewport or occlusion support. These will also be considered for necessity and complexity and will potentially be additional features for future iterations of the tool.

6.2.2 Research Perspective

The research has revealed that augmented reality is still emerging, which also shadows the development of supporting programs which also include the authoring tools. It can also be seen that the research and development for augmented reality has developed a faster pace in the last 10 years and is expanding to become a dominant field in the 2020s.

With expansion, support is also needed, and since beginning this thesis I have seen more and more companies like Unity, Microsoft and Vuforia releasing new applications to fill this gap for scaling and persistent solutions. With recent improvements in AR interactions and authoring tools, the process is just going to become easier and scalability is not going to be a problem anymore.

One of the research questions was to how to do good authoring for augmented reality. While the research has given some answers about general guidelines and patterns that have proven themselves, these were already important parts of RE'FLEKT ONE. Regarding AR interactions, it was hard to pinpoint which AR elements were the most beneficial to authoring, so it was decided after brainstorming sessions and the availability of the product.

Several AR interactions were tried out in the authoring tool and the feedback about the features were mostly positive. I can hope that this thesis can help other research in the future as an attempt to specify the best balances between AR authoring and easing the process with augmented reality elements.

7 Conclusion

There were two main goals of this thesis: Doing adequate research for the best practices for using augmented reality in industrial authoring scenarios and using this information for the development of an authoring tool for RE'FLEKT.

I can surely say that both of these goals are achieved to some extent. The research spans the definition of industrial augmented reality and it's use cases, designs and algorithms used for industrial solutions as well as currently utilized authoring applications and plugins used worldwide. It also investigates the proven methodologies for augmented reality based human computer interactions. The research, also backed by internal brainstorming sessions, gathers the information necessary to create an authoring tool best suited to the company's needs.

The creation of the authoring tool is based on different assets, both internal and external, connected together to shape an interactive editor which is designed to fit the ONE pipeline. On top of that, AR features aimed for covering the shortcomings of the existing authoring tool are added into the product as well. After an user survey done with the experts in the company, this tool has seen the green light to continue development and possibly be shifted into a major program in the pipeline, proving satisfactory usability and proficiency to the authors of RE'FLEKT and it's customers.

List of Figures

List of Tables

Bibliography

[1] *5 Trends Appear on the Gartner Hype Cycle for Emerging Technologies, 2019*. 2019. URL: https://www.gartner.com/smarterwithgartner/5-trends-appear-on-the-gartner-hype-cycle-for-emerging-technologies-2019/.

[2] *Magic Leap: Rewrite the rules of your industry*. (accessed on: 14.03.2020). 2020. URL: https://www.magicleap.com/enterprise.

[3] N. Navab. "Developing killer apps for industrial augmented reality". In: *IEEE Computer Graphics and applications* 24.3 (2004), pp. 16–20.

[4] *Oxford Dictionary*. (accessed on: 15.03.2020). 2020. URL: https://www.lexico.com/definition/augmented_reality.

[5] R. T. Azuma. "A survey of augmented reality". In: *Presence: Teleoperators & Virtual Environments* 6.4 (1997), pp. 355–385.

[6] *A Brief History of Augmented Reality (+Future Trends and Impact)*. (accessed on: 18.03.2020). 2019. URL: https://learn.g2.com/history-of-augmented-reality.

[7] *The Lengthly History of Augmented Reality*. (accessed on: 18.03.2020). 2017. URL: http://images.huffingtonpost.com/2016-05-13-1463155843-8474094-AR_history_timeline.jpg.

[8] *Pokémon GO Revenue and Usage Statistics (2019)*. (accessed on: 18.03.2020). 2019. URL: https://www.businessofapps.com/data/pokemon-go-statistics/.

[9] *Digi-Capital: Over $4.1 billion invested in AR and VR in 2019*. (accessed on: 12.05.2020). 2020. URL: https://venturebeat.com/2020/03/12/digi-capital-over-4-1-billion-invested-in-ar-and-vr-in-2019/.

[10] *Einsatzbereite Apps und Lösungen für HoloLens 2*. (accessed on: 18.03.2020). 2019. URL: https://www.microsoft.com/de-de/hololens/apps.

[11] P. Fraga-Lamas, T. M. Fernández-Caramés, Ó. Blanco-Novoa, and M. A. Vilar-Montesinos. "A review on industrial augmented reality systems for the industry 4.0 shipyard". In: *Ieee Access* 6 (2018), pp. 13358–13375.

[12] M. Hirzer. "Marker Detection for Augmented Reality Applications". In: (2008).

[13] G. Schall, E. Mendez, and Kruijff. "Handheld augmented reality for underground infrastructure visualization". In: *Personal and ubiquitous computing* 13.4 (2009), pp. 281–291.

[14] P. Fite-Georgel. "Is there a reality in industrial augmented reality?" In: *2011 10th ieee international symposium on mixed and augmented reality*. IEEE. 2011, pp. 201–210.

[15] P. Thomas and W. David. "Augmented reality: An application of heads-up display technology to manual manufacturing processes". In: *Hawaii International Conference on System Sciences*. 1992, pp. 659–669.

[16] S. Feiner, B. Macintyre, and D. Seligmann. "Knowledge-based augmented reality". In: *Communications of the ACM* 36.7 (1993), pp. 53–62.

[17] W. Friedrich, D. Jahn, and L. Schmidt. "ARVIKA-Augmented Reality for Development, Production and Service." In: *ISMAR*. Vol. 2002. Citeseer. 2002, pp. 3–4.

[18] A. Konsortium. *ARTESAS–Advanced Augmented Reality Technologies for Industrial Service Applications*. 2008.

[19] A. Raczynski and P. Gussmann. "Services and training through augmented reality". In: *1st European Conference on Visual Media Production (CVMP) 2004*. IET. 2004, pp. 263–271.

[20] *MAN's new app brings AR to training*. (accessed on: 28.02.2020). 2019. URL: https://shipinsight.com/articles/mans-new-app-brings-ar-to-training.

[21] *Stepping into the virtual world to enhance aircraft maintenance*. (accessed on: 29.02.2020). 2019. URL: https://www.airbus.com/newsroom/stories/stepping-into-the-%20virtual-world-to-enhance-aircraft-maintenance-.html.

[22] B. Lilly. "Mechanical Assemblies: their Design, Manufacture, and Role in Product Development". In: *Assembly Automation* (2006).

[23] S. Webel, U. Bockholt, T. Engelke, N. Gavish, M. Olbrich, and C. Preusche. "An augmented reality training platform for assembly and maintenance skills". In: *Robotics and autonomous systems* 61.4 (2013), pp. 398–403.

[24] S. Takata, F. Kimura, F. Van Houten, E. Westkämper, M. Shpitalni, D. Ceglarek, and J. Lee. "Maintenance: changing role in life cycle management". In: *Annals of the CIRP* 53.2 (2004), pp. 643–656.

[25] L. F. M. de Souza Cardoso. "A survey of industrial augmented reality". In: vol. 139. 2019.

[26] R. Palmarini, J. A. Erkoyuncu, R. Roy, and H. Torabmostaedi. "A systematic review of augmented reality applications in maintenance". In: *Robotics and Computer-Integrated Manufacturing* 49 (2018), pp. 215–228.

[27] S. Büttner, H. Mucha, M. Funk, T. Kosch, M. Aehnelt, S. Robert, and C. Röcker. "The design space of augmented and virtual reality applications for assistive environments in manufacturing: a visual approach". In: *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*. 2017, pp. 433–440.

[28] A. Syberfeldt, O. Danielsson, M. Holm, and L. Wang. "Visual assembling guidance using augmented reality". In: *Procedia Manufacturing* 1 (2015), pp. 98–109.

[29] N. Soete, A. Claeys, S. Hoedt, B. Mahy, and J. Cottyn. "Towards mixed reality in SCADA applications". In: *IFAC-PapersOnLine* 48.3 (2015), pp. 2417–2422.

[30] L. Hou, Y. Wang, X. Wang, N. Maynard, I. T. Cameron, S. Zhang, and Y. Jiao. "Combining photogrammetry and augmented reality towards an integrated facility management system for the oil industry". In: *Proceedings of the IEEE* 102.2 (2014), pp. 204–220.

[31] J. Tumler, F. Doil, R. Mecke, G. Paul, M. Schenk, and Pfister. "Mobile Augmented Reality in industrial applications: Approaches for solution of user-related issues". In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 87–90.

[32] Á. Segura, H. V. Diez, I. Barandiaran, A. Arbelaiz, and Álvarez. "Visual computing technologies to support the Operator 4.0". In: *Computers & Industrial Engineering* (2018), p. 105550.

[33] X. Wang, S. K. Ong, and A. Y. Nee. "A comprehensive survey of augmented reality assembly research". In: *Advances in Manufacturing* 4.1 (2016), pp. 1–22.

[34] A. Y. Nee, S. Ong, G. Chryssolouris, and D. Mourtzis. "Augmented reality applications in design and manufacturing". In: *CIRP annals* 61.2 (2012), pp. 657–679.

[35] Å. Fast-Berglund, L. Gong, and D. Li. "Testing and validating Extended Reality (xR) technologies in manufacturing". In: *Procedia Manufacturing* 25 (2018), pp. 31–38.

[36] M. E. C. Santos, A. Chen, T. Taketomi, G. Yamamoto, J. Miyazaki, and H. Kato. "Augmented reality learning experiences: Survey of prototype design and evaluation". In: *IEEE Transactions on learning technologies* 7.1 (2013), pp. 38–56.

[37] T. Langlotz, S. Mooslechner, S. Zollmann, C. Degendorfer, G. Reitmayr, and D. Schmalstieg. "Sketching up the world: in situ authoring for mobile augmented reality". In: *Personal and ubiquitous computing* 16.6 (2012), pp. 623–630.

[38] H. Bae, J. White, M. Golparvar-Fard, Y. Pan, and Y. Sun. "Fast and scalable 3D cyber-physical modeling for high-precision mobile augmented reality systems". In: *Personal and Ubiquitous Computing* 19.8 (2015), pp. 1275–1294.

[39] T. Engelke, J. Keil, P. Rojtberg, F. Wientapper, M. Schmitt, and U. Bockholt. "Content first: a concept for industrial augmented reality maintenance applications using mobile devices". In: *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM. 2015, pp. 105–111.

[40] F. Lamberti, F. Manuri, A. Sanna, G. Paravati, P. Pezzolla, and P. Montuschi. "Challenges, opportunities, and future trends of emerging techniques for augmented reality-based maintenance". In: *IEEE Transactions on Emerging Topics in Computing* 2.4 (2014), pp. 411–421.

[41] C. Knopfle, J. Weidenhausen, L. Chauvigne, and I. Stock. "Template based authoring for AR based service scenarios". In: *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE. 2005, pp. 237–240.

[42] H. Ramirez, E. G. Mendivil, P. R. Flores, and M. C. Gonzalez. "Authoring software for augmented reality applications for the use of maintenance and training process". In: *Procedia Computer Science* 25 (2013), pp. 189–193.

[43] G. Klinker, O. Creighton, A. H. Dutoit, R. Kobylinski, C. Vilsmeier, and B. Brugge. "Augmented maintenance of powerplants: A prototyping case study of a mobile AR system". In: *Proceedings IEEE and ACM international symposium on augmented reality*. IEEE. 2001, pp. 124–133.

[44] H. Alvarez, I. Aguinaga, and D. Borro. "Providing guidance for maintenance operations using automatic markerless augmented reality system". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE. 2011, pp. 181–190.

[45] T. Jung, M. D. Gross, and E. Y.-L. Do. "Annotating and sketching on 3D web models". In: *Proceedings of the 7th international conference on Intelligent user interfaces*. 2002, pp. 95–102.

[46] V. Havard, D. Baudry, A. Louis, and B. Mazari. "Augmented reality maintenance demonstrator and associated modelling". In: *2015 IEEE Virtual Reality (VR)*. IEEE. 2015, pp. 329–330.

[47] M. Fiorentino, A. E. Uva, M. Gattullo, S. Debernardis, and G. Monno. "Augmented reality on large screen for interactive maintenance instructions". In: *Computers in Industry* 65.2 (2014), pp. 270–278.

[48] J. Zhu, S. Ong, and A. Nee. "A context-aware augmented reality system to assist the maintenance operators". In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 8.4 (2014), pp. 293–304.

[49] J. A. Erkoyuncu, I. F. del Amo, M. Dalle Mura, R. Roy, and G. Dini. "Improving efficiency of industrial maintenance with context aware adaptive authoring in augmented reality". In: *Cirp Annals* 66.1 (2017), pp. 465–468.

[50] P. Grimm, M. Haller, V. Paelke, and S. Reinhold. "AMIRE-authoring mixed reality". In: *The First IEEE International Workshop Agumented Reality Toolkit,* IEEE. 2002, 2–pp.

[51] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter. "DART: a toolkit for rapid design exploration of augmented reality experiences". In: *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 2004, pp. 197–206.

[52] H. Seichter, J. Looser, and M. Billinghurst. "ComposAR: An intuitive tool for authoring AR applications". In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 177–178.

[53] M. Haringer and H. T. Regenbrecht. "A pragmatic approach to augmented reality authoring". In: *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE. 2002, pp. 237–245.

[54] J. Zauner, M. Haller, A. Brandl, and W. Hartmann. "Authoring of a mixed reality assembly instructor for hierarchical structures". In: *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society. 2003, p. 237.

[55] J. Platonov, H. Heibel, P. Meier, and B. Grollmann. "A mobile markerless AR system for maintenance and repair". In: *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2006, pp. 105–108.

[56] A. Hampshire, H. Seichter, R. Grasset, and M. Billinghurst. "Augmented reality authoring: generic context from programmer to designer". In: *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*. ACM. 2006, pp. 409–412.

[57] M.-J. Wang, C.-H. Tseng, and C.-Y. Shen. "An easy to use augmented reality authoring tool for use in examination purpose". In: *IFIP Human-Computer Interaction Symposium*. Springer. 2010, pp. 285–288.

[58] J. Gimeno, P. Tena, J. M. Orduna, and M. Fernández. "An Advanced Authoring Tool for Augmented Reality Applications in Industry". In: *Actas de las XXIII Jornadas de Paralelismo (JP 2012). Elche: Servicio de Publicaciones de la Universidad Miguel Hernández* (2012).

[59] N. Petersen and D. Stricker. "Learning task structure from video examples for workflow tracking and authoring". In: *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2012, pp. 237–246.

[60] N. Petersen and D. Stricker. "Cognitive augmented reality". In: *Computers & Graphics* 53 (2015), pp. 82–91.

[61] *Overview of Dynamics 365 Guides*. (accessed on: 23.03.2020). 2019. URL: https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/.

[62] *Vuforia Studio Web Version*. (accessed on: 23.03.2020). 2020. URL: https://www.ptc.com/en/products/augmented-reality/vuforia-studio.

[63] *Vuforia Spatial Toolbox - Augmented Reality*. (accessed on: 10.05.2020). 2020. URL: https://www.ptc.com/en/products/augmented-reality/vuforia-spatial-toolbox.

[64] *Reality Composer - Augmented Reality*. (accessed on: 23.03.2020). 2020. URL: https://developer.apple.com/augmented-reality/reality-composer/.

[65] *RapidManual*. (accessed on: 23.03.2020). 2020. URL: http://www.cortona3d.com/en/products/authoring-publishing-solutions/rapidauthor/rapidmanual.

[66] *What experts expect from enterprise augmented reality and virtual reality in 2020*. (accessed on: 23.03.2020). 2020. URL: https://blogs.unity3d.com/2020/02/12/what-experts-expect-from-enterprise-augmented-reality-and-virtual-reality-in-2020/.

[67] *Unity XR Platform Updates*. (accessed on: 23.03.2020). 2020. URL: https://blogs.unity3d.com/2020/01/24/unity-xr-platform-updates/.

[68] *XR Interaction Toolkit Preview Package is here*. (accessed on: 23.03.2020). 2020. URL: https://blogs.unity3d.com/2019/12/17/xr-interaction-toolkit-preview-package-is-here/.

[69] *Mixed and Augmented Reality Studio (MARS)*. (accessed on: 23.03.2020). 2020. URL: https://unity.com/unity/features/mars.

[70] A. Maneri. *Mixed and Augmented Reality Studio (MARS): Designing a framework for simplified, flexible AR authoring*. (accessed on: 23.03.2020). 2020. URL: https://blogs.unity3d.com/2020/01/03/mixed-and-augmented-reality-studio-mars-designing-a-framework-for-simplified-flexible-ar-authoring/.

[71] *Labs Spotlight: Project MARS - Unity Technologies Blog*. (accessed on: 23.03.2020). 2020. URL: https://blogs.unity3d.com/2019/10/02/labs-spotlight-project-mars/?_ga=2.67128850.222668468.1580313711-278837252.1565688036.

[72] S. Webel, U. Bockholt, T. Engelke, M. Peveri, M. Olbrich, and C. Preusche. "Augmented reality training for assembly and maintenance skills". In: *BIO web of conferences*. Vol. 1. EDP Sciences. 2011, p. 00097.

[73] F. Biocca, A. Tang, C. Owen, and F. Xiao. "Attention funnel: omnidirectional 3D cursor for mobile augmented reality platforms". In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006, pp. 1115–1122.

[74] M. Tonnis and G. Klinker. "Effective control of a car driver's attention for visual and acoustic guidance towards the direction of imminent dangers". In: *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2006, pp. 13–22.

[75] B. Schwerdtfeger and G. Klinker. "Supporting order picking with augmented reality". In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 91–94.

[76] S. J. Henderson and S. Feiner. "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret". In: *2009 8th IEEE International Symposium on Mixed and Augmented Reality*. IEEE. 2009, pp. 135–144.

[77] *Unique Model Tracking with VisionLib*. (accessed on: 25.03.2020). 2020. URL: https://visionlib.com/modeltracking/.

[78] P. Mohr, B. Kerbl, M. Donoser, D. Schmalstieg, and D. Kalkofen. "Retargeting technical documentation to augmented reality". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 3337–3346.

[79] *Enterprise OS for Augmented Reality by RE'FLEKT*. (accessed on: 25.03.2020). 2020. URL: https://www.re-flekt.com/company/.

[80] *RE'FLEKT Logo*. (accessed on: 25.03.2020). 2020. URL: https://media.licdn.com/dms/image/C4D0BAQF9_Dl_mBXcsA/company-logo_200_200/0?e=2159024400&v=beta&t=C54yAfsqWv-70ML_KWJdl9x_ZJFOTXRIEW1rg-w7uO8.

[81] *RE'FLEKT Press News*. (accessed on: 25.03.2020). 2020. URL: https://www.re-flekt.com/press-news.

[82] *What is DITA?* (accessed on: 10.04.2020). 2020. URL: https://www.xml.com/articles/2017/01/19/what-dita/.

[83] *Microsoft AppSource - REFLEKT ONE*. (accessed on: 12.05.2020). 2018. URL: https://appsource.microsoft.com/en-us/product/web-apps/reflektgmbh.reflekt_one?tab=Overview.

[84] *Runtime Transform Handles Documentation*. (accessed on: 15.04.2020). 2020. URL: http://rteditor.battlehub.net/transform-handles/.

[85] *Runtime Hierarchy and Inspector*. (accessed on: 15.04.2020). 2020. URL: https://github.com/yasirkula/UnityRuntimeInspector.

[86] *Standalone File Browser*. (accessed on: 17.04.2020). 2020. URL: https://github.com/gkngkc/UnityStandaloneFileBrowser.

[87] *Unity - Manual: UnityEvents*. (accessed on: 10.05.2020). 2020. URL: https://docs.unity3d.com/Manual/UnityEvents.html.

[88] *Device Metrics - A comprehensive resource for sizing, resolution and more across multiple devices*. (accessed on: 20.04.2020). 2020. URL: https://material.io/resources/devices/.

[89] *The HoloLens Might Have a Small Field of View, but That's Actually a Good Thing*. (accessed on: 18.04.2020). 27.05.2016. URL: https://hololens.reality.news/news/hololens-might-have-small-field-view-but-thats-actually-good-thing-0171349/.

[90] *Ben Fineman's (@bfineman) Twitter, Image Comparing Augmented Relity Devices' Field of Vision*. (accessed on: 18.04.2020). 2020. URL: https://twitter.com/bfineman/status/1100475267569598469.

[91] *MRTK Camera System*. (accessed on: 18.04.2020). 2020. URL: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/CameraSystem/CameraSystemOverview.html.

[92] *Microsoft Significantly Misrepresented HoloLens 2's Field of View at Reveal*. (accessed on: 18.04.2020). 27.02.2019. URL: https://www.roadtovr.com/microsoft-significantly-misrepresented-hololens-2s-field-of-view-at-reveal/.

[93] *HoloLens 2's Field of View Revealed*. (accessed on: 18.04.2020). 25.02.2019. URL: https://uploadvr.com/hololens-2-field-of-view/.

[94] *Default camera FoV is not set correctly, Issue #176*. (accessed on: 18.04.2020). 17.08.2019. URL: https://github.com/microsoft/MixedRealityToolkit-Unity/issues/176.

[95] *How Do You Pronounce "Likert?" What is a Likert Scale?* (accessed on: 05.05.2020). 2015. URL: http://core.ecu.edu/psyc/wuenschk/StatHelp/Likert.htm.