A Comparative Study of AR Authoring Tools for Prototyping AR Experiences

Vanessa Nugaela

DEPARTMENT OF DESIGN SCIENCES
FACULTY OF ENGINEERING LTH | LUND UNIVERSITY
2026

MASTER THESIS

A Comparative Study of AR Authoring Tools for Prototyping AR Experiences

Vanessa Nugaela

LUND
UNIVERSITY

A Comparative Study of AR Authoring Tools for Prototyping AR Experiences

Published by
Department of Design Sciences
Faculty of Engineering LTH, Lund University
P.O. Box 118, SE-221 00 Lund, Sweden

Subject: Virtual Reality and Augmented Reality (MAMM20)

Supervisor: Günter Alce
Examiner: Joakim Eriksson

Abstract

Augmented Reality (AR) combines digital content with the physical environment, but creating AR prototypes often requires programming skills, 3D knowledge and interaction design experience. AR authoring tools can lower these barriers through visual interfaces and low-code workflows. This thesis compares how Unity and Snap Lens Studio support rapid prototyping of a reminiscence-based AR experience in a palliative care design context. The same AR concept was implemented in both tools through authoring workflows. In Unity and Snap Lens Studio, a virtual room was created and customised using imported memory images, interactive memory frames and slideshow-based interaction logic. The development process was documented and complemented by a qualitative evaluation with six participants, who followed a guided test procedure in both tools and reflected on the authoring process. The results show that Unity offers greater flexibility and technical control, but requires more setup and coding. Snap Lens Studio enables a faster and more visual workflow, making it suitable for early-stage prototyping, but becomes more limited when customised interaction is needed.

Keywords: Augmented Reality (AR), AR authoring tools, Rapid Prototyping, Reminiscence

Sammanfattning

Augmented Reality (AR), eller förstärkt verklighet, används för att kombinera digitalt innehåll med den fysiska omgivningen, men utveckling av AR-prototyper kräver ofta teknisk kunskap. Detta examensarbete jämför hur Unity och Snap Lens Studio stödjer snabb prototypframtagning av en reminiscensbaserad AR-upplevelse i en palliativ vårdkontext. Samma AR-koncept implementerades i båda verktygen och jämfördes utifrån utvecklingstid, arbetsflöde, teknisk komplexitet, kodningskrav, interaktionsdesign och begränsningar. Studien kompletterades även med en kvalitativ utvärdering där deltagarna fick bygga och reflektera över processen. Resultaten visar att Unity erbjuder större flexibilitet och teknisk kontroll, men kräver mer tid. Snap Lens Studio har ett snabbare och mer visuellt arbetsflöde, men har fler begränsningar vid avancerade interaktioner. Studien visar därmed att Unity är mer lämpligt när kontroll och flexibilitet prioriteras, medan Snap Lens Studio passar bättre för snabb och visuell AR-prototypframtagning.

Nyckelord: Förstärkt Verklighet, AR verktyg, Snabb Prototypframtagning, Reminiscens

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Günter Alce, for his guidance, support and valuable feedback throughout this thesis. His encouragement and advice have been greatly appreciated and have helped shape the direction of this work.

I would also like to thank all participants who took part in the evaluation of this study. The time, reflection and honest feedback were essential for understanding the prototyping process and contributed well to the results of this thesis.

Finally, I would like to thank my family and friends for their encouragement and support throughout this process. Their confidence in me helped maintain my motivation during the challenges of this thesis.

Lund, May 2026
Vanessa Nugaela

Contents

List of acronyms and abbreviations

AR	Augmented Reality
GDPR	General Data Protection Regulation
HCI	Human-Computer Interaction
iOS	iPhone Operating System
SDK	Software Development Kit
SDG	Sustainable Development Goals
UN	United Nations
VR	Virtual Reality
WHO	World Health Organisation
XR	Extended Reality
3D	Three-dimensional

1 Introduction

The following chapter aims to build a foundation for the research presented in the thesis. It first introduces the background of Augmented Reality (AR) and AR authoring tools, followed by a description of the problem that motivates the study. The aim and purpose of the thesis are stated, together with research questions that guide the work. The chapter also discuss the relevance of the project.

1.1 Background

Augmented reality (AR) is an increasingly important area within human-computer interaction (HCI), design and immersive media. Unlike fully virtual environments, AR extends the physical world by integrating digital objects, images, information and interactions into real-world settings. Azuma's definition describes AR as a system that integrates virtual elements with the real world, has real time interaction and aligns digital content with a three-dimensional (3D) physical environment [1].

AR has been applied in fields such as education, design, entertainment, training and healthcare [2]. However, AR development application often requires programming skills, knowledge of 3D environments and interaction design. This technical complexity can make AR development challenging for researchers or other non-expert developers who want to explore AR concepts but have limited programming experience [4].

AR authoring tools aim to reduce these barriers by offering visual interfaces, templated

and low-code or even no code functionality. These tools allow users to place digital objects, define interactions and test prototypes without building an application entirely through traditional programming [4]. This is especially useful in early design stages and research contexts where the goal is to explore ideas and test interaction concepts.

In this context, rapid prototyping is particularly important as it allows designers and researchers to create functional prototypes in a short period of time. In AR, this can include prototyping object placement, media visualisation, touch-based interaction and mobile testing. However, different AR authoring tools may support these tasks in different ways. Some may be easier to learn but less flexible, while others may offer more technical control but require more time and coding knowledge.

1.2 Problem Description

Although AR authoring tools are designed to make AR development more accessible, it is still important to understand how well they support rapid prototyping in practice. Tools that are described as visual, easy to use or low-code may still involve technical challenges [5]. This can include limitations in interaction design, difficulties with 3D object placement or challenges when testing the prototype.

For designers and researchers, the choice of tool can influence the prototyping process [6]. It may affect what type of AR experience can be created, how much time the process requires and what level of technical knowledge is needed. Consequently, this creates a challenge when selecting suitable tools for AR prototyping.

The problem addressed in this thesis is the lack of structured comparison between AR authoring tools from a rapid prototyping perspective. Without a clearer understanding of how different authoring tools support prototyping tasks, it can be difficult to select a tool that fits the needs of specific projects.

1.3 Aim and Purpose

The aim of this thesis is to evaluate and compare how two AR authoring tools, selected through a pre-study, support the rapid prototyping of AR experiences. The AR experience used in this study is based on reminiscence therapy, with a particular focus on the development process rather than on clinical outcomes.

The purpose is to investigate how each selected tool supports the creation of a reminiscence-based AR prototype. This includes examining how the tools enable the placement of digital content, media visualisation, interaction design, coding or non-code functionality and lastly testing on mobile devices. By implementing the same scenario in both tools, the study provides a structured comparison of their strengths, weaknesses and suitability for AR prototyping.

The reminiscence-based scenario is used as a design case [6], since it offers a meaningful interaction context involving personal media and memory prompts. The scenario functions as a practical case through which the selected authoring tools can be tested, analysed and compared.

1.3.1 Research Questions

The study is guided by the following research questions:

- How do the selected AR authoring tools differ in terms of development time, technical complexity, workflow structure, coding requirements and limitations?
- How well does each tool support the implementation of core AR prototype features?
- Which of the selected tools provides the most appropriate balance between ease of use and flexibility for rapid AR prototyping?

1.4 Relevance for sustainable development

The 2030 Agenda for Sustainable Development is a global framework established by the United Nations (UN) [7] and focuses on addressing social, economical and environmental challenges. The agenda consists of 17 Sustainable Development Goals (SDG), which aim to support a more sustainable and inclusive future. These goals encourage countries, organisations and researchers to contribute to development in areas of health, education and innovation. For the following thesis, the most relevant SDG are:

- SDG 3: Good Health and Well-Being
- SDG 4: Quality Education
- SDG 9: Industry, Innovation and Infrastructure
- SDG 10: Reduced Inequalities

1.4.1 Good Health and Well-Being

The third goal aims to ensure healthy lives and promote well-being for everyone at any age [7]. This goal is relevant to the thesis because of reminiscence-based AR scenario, which is connected to well-being, memory and personal experiences. Although this thesis does not evaluate clinical outcomes or effectiveness of reminiscence therapy, the chosen design focuses on how digital technologies may support emotional and social experiences in care related settings.

1.4.2 Quality Education

This goal focuses on ensuring inclusive and equitable quality education as well as promoting lifelong learning opportunities [7]. The thesis investigates AR authoring tools that can make the development of AR experiences more accessible. By examining low-code and visual authoring tools, the study contributed to understanding how people with limited programming knowledge can learn, experiment and prototype with immersive technologies.

1.4.3 Industry, Innovation and Infrastructure

AR authoring tools are part of the broader development of digital innovation and interactive technologies [7]. By comparing how different tools support rapid prototyping, the study contributes knowledge about how innovative digital experiences can be developed more efficiently and with lower technical barriers.

1.4.4 Reduced Inequalities

Goal number ten aims to reduce inequalities within and among countries. Technical complexity can create barriers for people who do not have advanced programming skills or access to development resources. AR authoring tools may help reduce these barriers by making AR prototyping more accessible to non-expert developers, researchers and designers. In this way, the thesis relates to broader questions of accessibility and inclusion in digital technology development.

1.5 Ethical considerations

Although the study does not evaluate the clinical effects or patient outcomes, there are still several ethical aspects to consider. These aspects include privacy, accessibility, the use of personal data and limitations of AR technologies. As for the following study, no personal data or sensitive media was collected.

Under the General Data Protection Regulation (GDPR), personal data must be processed fairly and only data that is necessary for the purpose should be collected. GDPR also requires a lawful basis for processing personal data, such as consent.

During the evaluation of the following study, all participants were informed about the purpose of the study, what type of data would be collected and how the data would be used. Participation was voluntary and participants had the right to withdraw from the study at any time. Since the evaluation focused on tool usage and authoring workflows, the collected data consisted solely of observation notes and reflections

rather than sensitive information. The results are therefore presented in a way that avoids identifying individual participants.

1.5.1 Usage of AI

The memory images used in the prototype and the user evaluation setup image in Figure 3.1 were AI generated. The memory images served as placeholders. This allowed the prototype to be evaluated without using real personal photographs.

1.6 Related Work

This section summarises previous findings and insights that form the foundation of this project. The thesis builds on research in four main areas; AR as an interactive medium, AR authoring tools, rapid prototyping and immersive technologies within healthcare. Together, these areas provide the background for comparing how different AR authoring tools support the development of a reminiscence-based AR prototype.

Previous research has shown that AR and VR development can be difficult, even when development tools are becoming more accessible. Ashtari et al. studied AR and VR creators and identified several recurring barriers, including difficulties in early prototyping and dealing with technical uncertainty [4]. Their findings show that the challenge is not only whether an AR application can be built, but how the development process is shaped by the tools available.

A similar issue is discussed by Nebeling and Speicher, who argue that many AR and VR authoring tools do not fully support the needs of non-technical designers and end users. Their work highlights a gap between the growing interest of immersive experiences and the lack of tools that allow users to quickly prototype without advanced technical knowledge [8]. This is especially relevant to the present study, since the comparison of the authoring tools concerns the balance between technical control and accessibility.

Research on AR authoring has also explored how visual and low-code approaches can support users who are not professional programmers. Work on visual scripting based AR authoring tools such as TrainAR shows that domain experts can create AR training experiences through node-based authoring rather than traditional programming [9]. This indicates that visual authoring can lower development barriers and make AR creation more accessible. However, such tools may also limit flexibility when the prototype requires more detailed control over the experience. This finding between accessibility and customisation is directly connected to the comparison carried out in this thesis [9].

Technology-supported reminiscence has mainly focused on how digital media can support memory and storytelling according to previous research. However, Lazar et al. reviewed the use of information and communication technologies in reminiscence therapy and found that digital systems often use photographs and other material to support memory related conversations. They conclude that technology can make reminiscence material easier to access and organise while also supporting more active participation in memory based interaction [10].

This thesis addresses the gap between by comparing two different AR authoring tools through the development of the same reminiscence-based AR concept. Existing studies often focus on AR and VR development in general or on specific authoring tools. Less attention has been given to how different AR authoring tools affect the practical process of creating a reminiscence-based AR prototype.

2 Theory

This chapter presents the theoretical foundation of the thesis. It introduces key concepts related to immersive technology, with a particular focus on Augmented Reality and its technical components. The chapter also explains the key concepts that are essential for understanding the comparison in this study. Reminiscence-based interaction and palliative care are also introduced.

2.1 Immersive technology

Immersive technology refers to digital systems that alter or replace the user's perception of the surrounding environment. These technologies include AR, virtual reality (VR), mixed reality (MR) and extended reality (XR). Although these terms are sometimes used similarly, they represent different relationships between the physical and the digital worlds. Milgram and Kishino's reality-virtuality continuum describes this relationship as a spectrum that has a range from the completely real environment to the completely virtual environment. Within this, AR is positioned closer to the real environment as it preserves the user's view of the physical world whilst adding digital elements to it [11].

2.1.1 Augmented Reality

Augmented Reality is commonly defined as a technology that extends the physical world with digital content. Rather than replacing the real environment, AR overlays

virtual objects, images, sounds or information into the user's perception of the real world. Azuma's definition is one of the most widely cited in AR research. According to this definition, an AR system has three main characteristics; it combines real and virtual content, it is interactive in real time and it is registered in three dimensions [1].

Unlike VR, which replaces the user's surroundings with a simulated one, AR keeps the physical environment visible. Digital elements are then added as layers on top of the environment or even within. This means that the user can still see the room, table or other physical surroundings while interacting with virtual prompts. AR systems require several components to work together. The device camera captures the physical environment, tracking algorithms, estimating the position and orientation of the device [2].

2.1.1.1 Spatial Tracking and Registration

One of the most important technical aspects in AR is spatial tracking. Spatial tracking allows the device to understand where it is in relation to the physical environment [12]. In mobile AR, this is usually obtained through the camera, motion sensors and software algorithms that estimate the device's movement. The AR system tracks both the position and the rotation of the device so that digital content can be displayed from the correct viewpoint [15]. Registration refers to the alignment between digital content and the physical world. Azuma identifies registration as one of the core characteristics of AR, since virtual and real content must be aligned in three-dimensional space [1].

2.1.1.2 Plane Detection

Plane detection is a common AR feature that allows the system to identify flat surfaces in the physical environment. These surfaces include both horizontal planes, such as floors, as well as vertical planes, such as walls [12]. Plane detection is important as it gives the user a meaningful way to place digital media in the real world. Instead of floating around in space, a digital object can be positioned on for instance a table.

However, plane detection also has limitations. It depends on lighting conditions, surface texture, camera quality and the amount of visual detail in the environment. Plain white walls, reflective surfaces or dark rooms can make detection less reliable

[12].

2.1.1.3 Object Placement and Anchors

Object placement is a central part of AR because it determines where digital content appears in relation to the physical environment. Object placement affects both the technical function and the user experience. If an object is misplaced, it may feel disconnected from the environment hence why this is crucial for memory frame prompts [8].

Anchors are used in AR to keep digital objects fixed to a position in the physical world. Without anchors, digital objects may move unpredictably when the AR session is updated together with the environment. Anchors are therefore especially important when the user moves around the object [13].

2.1.1.4 Transformations and Materials

In AR prototyping, digital objects are controlled through transformations: position, rotation and scale. Position defines where an object is located in the scene, rotation defines the direction it faces and scale defines the size [14]. These properties are important as AR content must appear in a way that feels spatially meaningful and easy to interact with.

Materials define how a digital object appears in the AR scene. They can include colour, texture and transparency. Materials connect visual assets to objects in the scene [12]. In this thesis, materials are relevant as they affect how the reminiscence images are presented and how clearly the memory frames are understood.

2.2 AR Interaction

Interaction in AR involves users engagement with digital content that is connected to the physical world. Interaction therefore includes tapping, dragging, rotation, scaling, selecting and moving. More advanced AR systems may also include other interactions such as hand tracking, gaze based interaction and voice commands [2].

2.3 AR Authoring Tools

An AR authoring tool is a software application that supports the creation of an AR experience. Instead of requiring users to build an AR application entirely through code, authoring tools often provide visual interfaces, templates, drag and drop functionality, asset libraries or low-code scripting. The purpose of these tools are to reduce the technicalities of AR development and make it possible to create AR prototypes more efficiently [17].

Authoring tools can vary significantly in complexity. Some tools are designed for non-programmers and focuses on usability, while others are integrated into professional development environments and offer greater flexibility but require more technical knowledge [17]. The difference between ease of use and flexibility is crucial for this thesis as it compares AR authoring tools.

2.4 Rapid Prototyping

Rapid prototyping is the process of creating early versions of a system quickly in order to test ideas, evaluate the interaction and discover limitations. In interaction design, prototypes range from low fidelity sketches to functional digital systems. The purpose is not necessarily to create a finished product, but to learn through making [2].

In AR development, rapid prototyping can be challenging because AR systems often require spatial tracking and object placement [6]. Authoring tools can support rapid prototyping by simplifying these tasks and allowing designers to test ideas without building a full application from scratch.

2.5 Descriptive Statistics

Descriptive statistics were used in this thesis to summarise the participant completion times during the evaluation. Since the evaluation involved a small group of participants, the calculations were used to provide an overview of the observed data rather than to make statistical generalisations. The main measures used were mean, variance and standard deviation [3].

The mean represents the average completion time and was calculated by summing all participant times and dividing the result by the number of participants:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.1}$$

where $\bar{x}$ is the mean value, x_i represents each participant's completion time and n is the number of participants.

The sample variance was used to describe how much the completion times differed from the mean. Since the participant group represents a sample, the variance was calculated using $n-1$ in the denominator:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} \tag{2.2}$$

where s^2 is the sample variance.

The standard deviation was calculated as the square root of the variance:

$$s = \sqrt{s^2} \tag{2.3}$$

2.6 Reminiscence-Based Interaction

Reminiscence-based interaction refers to interactive experiences that support the sharing of personal memories. Butler described reminiscence as an important psychological process, where individuals look back on earlier experiences and integrate them into their understanding of the present [18]. Reminiscence therapy is therefore described as the discussion of memories and past experiences, often supported by prompts such as photographs, music and other familiar objects [19]. These prompts help individuals remember earlier life events that support conversation and storytelling.

Digital reminiscence refers to the usage of digital technology to present memory related materials. Digital approaches are relevant for interaction design since they show how personal memory material can be structured and presented through technology [10]. The interaction can range from moving between photographs to recording a narration as an audio prompt [20]. The reminiscence-based scenario therefore functions as a meaningful design context for the following thesis.

2.7 Palliative Care

Palliative care is an approach taken by the healthcare that aims to improve the quality of life of patients and families facing life-threatening illness. The World Health Organization (WHO) defines palliative care as a support that aims to prevent and relieve suffering through early identification and treatment. This makes palliative care a highly sensitive context [21]. Common parts of palliative care include pain relief, psychological support and communication with relatives. This broadens the focus beyond physical pain and instead shifts to the emotional and social aspects of illness [22].

3 Method Setup

This chapter presents the pre-study done to select the AR authoring tools, as well as the concept generation process that reflects the prototype design. The method involves both practical prototyping and reflective comparison in order to later compare the AR authoring tools through the same reminiscence-based experience.

3.1 Development Plan

The previous chapters present that AR has significant potential for supporting reminiscence-based experiences. However, the process required to create interactive AR experiences remains underexplored. To answer the research questions presented in Chapter 1.3, this study follows a comparative prototyping methodology. A central method in this thesis is the development of prototypes as it functions as a practical example of how reminiscence-based AR experiences can be designed. The prototypes also acts as research instruments through which the authoring processes of different AR tools can be compared. This approach is connected to Design Science Research, which is a systematic and iterative approach that focuses on the development of a practical prototype. Rather than producing knowledge only in theory, it involves creating, testing, evaluating and refining solutions through a continuous design process [23].

The research begins by identifying the problem of how AR authoring tools influence the development process, followed by a technical pre-study of different platforms in order to select two suitable tools for comparison. The comparison is structured

around the same conceptual AR experience using two different authoring tools. The development process was then evaluated using criteria such as workflow, technical complexity, limitations and development time. By developing the same concept in both tools, the study enables a comparative analysis of how different AR authoring tools support the design of interactive AR experiences.

3.2 Initial Phase

The initial research problem emerges from the combination of AR, reminiscence therapy and palliative care. As discussed in Chapter 2, AR has increasingly been explored in healthcare contexts, including rehabilitation and mental health support [22]. At some places reminiscence therapy has been used to support memory and emotional connection through stimuli.

However, while both AR and reminiscence therapy have been studied separately, there is still limited research on how AR can be used for reminiscence-based experience. In this context, AR offers the possibility of placing digital memories and objects into the user's physical environment, creating a more interactive form of memory engagement.

The research problem therefore focuses on how an interactive AR experience can be designed to support reminiscence in a care context and how different AR authoring tools enable or limit this development process. The study considers the practical process of creating the experience which includes how each tool supports interaction design, spatial placement, usability and the overall workflow required to develop an AR based reminiscence experience.

This leads to the central problem of the thesis, there is a need to better understand how AR authoring tools can support the design and development of meaningful, interactive AR experiences for reminiscence therapy in palliative care. By developing the same concept in two different tools, the study aims to identify both the possibilities and limitations of each tool, while also contributing to a broader understanding of how AR may be applied in meaningful healthcare contexts.

3.3 Choosing compatible software

There are many tools available for authoring AR content. These range from simple template-based applications to advanced game engines. The goal of the pre-study was to identify tools that could support the creation of an interactive reminiscence experience while also representing different levels of authoring complexity.

The tools considered during the pre-study included Unity, Snap Lens Studio, Adobe Aero, 8th Wall and ZapWorks. These tools were reviewed based on their ability to support interactive AR content, asset integration, user input, accessibility and rapid prototyping.

MindAR was considered for the study because it offered a lightweight and accessible path into prototyping. It is a web-based AR library that supports image tracking and face tracking, meaning AR experiences can run directly in a mobile browser [24]. MindAR also integrates with common web-based frameworks, making it suitable for simpler AR experiences such as scanning a printed image or overlaying digital content. However, the core functionality is mainly limited to image tracking and face tracking. MindAR would be too limited as the study required placing digital objects freely into a physical environment, detecting surfaces and creating interactions [24].

8th Wall, which was a WebAR platform, was also considered because it supports browser-based AR and 3D experiences. This made 8th Wall accessible since users do not need to download a separate app. However, 8th Wall was not considered as an alternative because its workflow is more closely connected to web development rather than visual AR authoring. Building custom interactive experiences in this platform involves web technologies and 3D JavaScript frameworks [25]. This would shift the study from comparing AR authoring workflows to also comparing web development. In addition, 8th Wall retired on the 28th of February 2026. Thus, 8th Wall was considered as a less suitable platform [26].

The next AR authoring platform to be examined was ZapWorks as it provides several tools for creating AR experiences, including no code and more advanced development

options [27]. ZapWorks Designer is described as a browser based, drag and drop AR development tool that supports image tracking. However, ZapWorks' tool ecosystem is broader and less directly aligned with the intended comparison. The no code workflow would mainly support simpler AR experiences, while the more advanced ZapWorks tools and SDKs would introduce a different technical development workflow [27].

Further into the pre-study, Unity was selected for this study as it represents a high control, game engine based approach to AR development. Unity's AR Foundation package enables developers to create multi-platform AR applications in Unity [28]. Google's ARCore further describes AR Foundation as a cross platform framework that allows developers to write an AR experience once and build it for either Android or iOS devices [29].

For this study, Unity was chosen because it offers flexibility and full control over functionality. It allows the developer to define custom interaction logic and build more advanced systems if needed [30]. This makes it suitable for testing how a more technically advanced tool supports the development of a reminiscence-based AR prototype.

However, Unity also requires a more technical knowledge than many dedicated AR authoring tools. Setting up an AR project involves installing packages, configuring build settings and writing scripts [30]. This makes Unity useful for examining the gap between flexibility and complexity.

Snap Lens Studio was selected as a contrasting platform to Unity as it represents a more accessible rapid prototyping approach to AR authoring. This platform supports visual scripting which is described as node-based and useful for users who are new to interactive platforms [31]. Snap Lens Studio also explains that logic in visual scripting is defined through a series of nodes and responds to interactions such as touch input. Snap Lens Studio was chosen because it supports a faster and more visual workflow. Its node-based and template driven, making interaction design more approachable, especially for designers and researchers who do not have programming experience [32]. The tools considered and the reasons for including or excluding them

are summarized in Table 3.1.

Table 3.1: Overview of considered AR authoring tools.

Software	**Platform Support**	**Advantages**	**Limitations**
MindAR	Web Browser, JavaScript	Easy to use. Supports image tracking which enables mobile use.	Less suitable for examining complex authoring workflows such as analysing detailed interaction logic.
8th Wall	WebAR, 3D web frameworks	Suitable for web based 3D and AR experiences. Has no requirements on a separate application.	Workflow is connected to web development shifting the focus from comparing AR authoring tools. The platform was also retired.
ZapWorks	ZapWorks Designer, SDK-based	Includes no-code tools and relevant advanced development options for the study.	Supports simpler AR experiences and SDKs have different technical workflows.
Unity	Unity Editor, AR Foundation, ARCore	Flexible and high control over AR experience. Suitable for examining more technically advanced authoring workflow.	Requires more technical knowledge. Requires scripting and a steeper learning curve.
Snap Lens Studio	Lens Studio, Snapchat	Supports rapid prototyping and a more visual approach. Has template driven workflow. Suitable as a contrast to Unity.	Less control and flexibility. May limit more advanced custom functionality.

3.3.1 Final Selection

Unity and Snap Lens Studio were selected because they represent two different approaches to AR authoring and provides interesting findings based on their differences. Unity offers deep technical control and scalability through a game engine based workflow. Snap Lens Studio offers speed, visual scripting and built-in AR prototyping.

3.4 Creating a Scenario

After the software had been selected, a conceptual scenario was created as the foundation of the prototypes. The concept was based on literature on reminiscence therapy, memory, palliative care and interaction design. The scenario was developed to provide a shared design case that could be implemented in both Unity and Snap Lens Studio, making it possible to compare the tools under similar conditions while remaining simple enough to be implemented within the scope of the study.

The selected concept is an AR reminiscence experience in which the user points a mobile device towards a physical surface. When the AR scene is activated, a virtual memory space appears in the user's physical environment. This space contains visual memory cues such as photographs placed in frames, familiar objects and simple interactive elements. The user can move around the scene, view the memory material and interact with selected elements by tapping on an image to change the displayed photograph.

This concept was chosen because it combines several important elements to reminiscence such as physical context, visual stimuli, interaction and emotional engagement. When translating these images into an AR setting, the concept creates a simple but meaningful experience. By implementing the same scenario in both tools, it became possible to evaluate key authoring features for AR prototyping while also comparing the development processes in terms of workflow, complexity, flexibility, and ease of prototyping.

3.5 Evaluation Method

In order to evaluate how the selected AR authoring tools support rapid prototyping, a qualitative test was conducted. The purpose of the evaluation was to investigate how understandable, efficient and usable the prototype building process was when working with the selected tools. The test focused on the authoring process rather than the final prototype experience.

The evaluation was designed to complement the implementation as it provided additional insight on how users perceived the process. This made it possible to identify difficulties, recurring patterns and differences between the tools from a rapid prototyping perspective. This test was then followed by a short interview about the process. The evaluation setup is illustrated in Figure 3.1, showing the participant, the guided test procedure, the consent form and the setup.

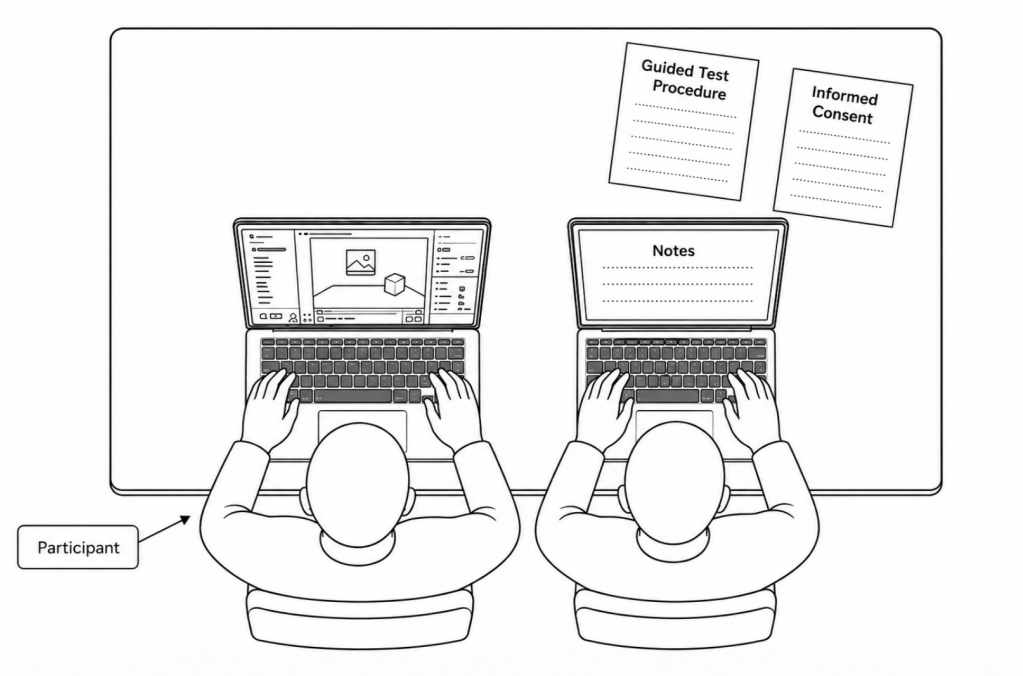

Figure 3.1: Visual of the evaluation setup with the participant to the left.

3.5.1 Participants

Six participants took part in the evaluation, consisting of three women and three men with an average age of 26 years. All participants had some form of technical background and were familiar with digital tools or basic technical concepts. Three of them were students at LTH currently pursuing engineering degrees, while the remaining participants had a personal interest in technology or other technical aspects. Their role in the evaluation was limited to testing the prototype and reflecting on the process of building a simple AR experience. Three participants began the evaluation in Unity before proceeding to Snap Lens Studio, while the remaining participants completed the evaluation in Snap Lens Studio first and then followed by Unity.

Table 3.2: Overview of participants' backgrounds and relevant skill sets.

Participant	**Background**	**Skill set relevant to the study**	**Starting tool**
P1	Computer Science Student	Programming experience and familiarity with technical software	Unity
P2	Test Engineer, Sony	Technical testing experience but less familiarity with software workflows	Unity
P3	Biomedical Engineering Student	Technical education and general digital tool experience	Unity
P4	Technical salesperson	Practical technology experience, but no prior AR development experience	Snap Lens Studio
P5	Electrical Engineering Student	Engineering background and has experience within AR development	Snap Lens Studio
P6	Technology enthusiast	General interest in technology and digital tools, limited AR experience	Snap Lens Studio

3.5.2 Test Procedure

The evaluation was based on observations and a short interview. During the test, participants were asked to follow a guided prototype building task and verbalise their thoughts while executing the tasks. The purpose was not to assess the participants' technical performance, but to examine how they experienced the authoring process. The task followed the same structure for all participants. This session was limited to 5 hours for each participant. This included introduction, guided tasks in both tools, questions and post-test interview. Figure 3.2 summarises the evaluation flow, including participant background, tool order and the main stages of the guided tasks.

At the beginning of the session, the participant was introduced to the purpose of the test and was asked to read the guided instructions carefully. The complete guide and interview questions are included in the Appendices. First, the participant opened the software together with the project file and was asked to familiarise themselves with the interface, which included panels such as the scene hierarchy, asset browser, inspector, preview panel and scene view. They then examined the basic scene structure by locating and expanding a parent object representing the room environment. As part of the task, they created central scene objects such as the floor, walls and the memory frame.

The next part of the task focused on modifying and understanding the AR scene. Participants inspected objects in the scene and adjusted or reviewed properties such as position, scale and material. They then imported three memory images into the project and applied one of these images to a material placed on the memory frame.

The final part of the task focused on interaction and previewing. Participants reviewed or created a slideshow manager object and connected the imported memory images to the interaction logic. However, they were not required to create a script, instead the script was provided alongside with an explanation. The participants tested the prototype in the preview panel by using simple interactions which allowed the evaluation to include both the visual authoring process and the setup of the interactions.

The data was collected through observation notes during the test and focused on both

verbal comments and observed behavior. Particular attention was given to moments where participants experienced confusion, required clarification, made mistakes or commented on the complexity of the tool.

The collected notes were analysed thematically. First, the notes from each participant were reviewed individually. Then, recurring comments, difficulties and reflections were grouped into themes. These themes were compared among the participants to identify common patterns and deviations. The analysis focused on the participants' experiences related to the research questions.

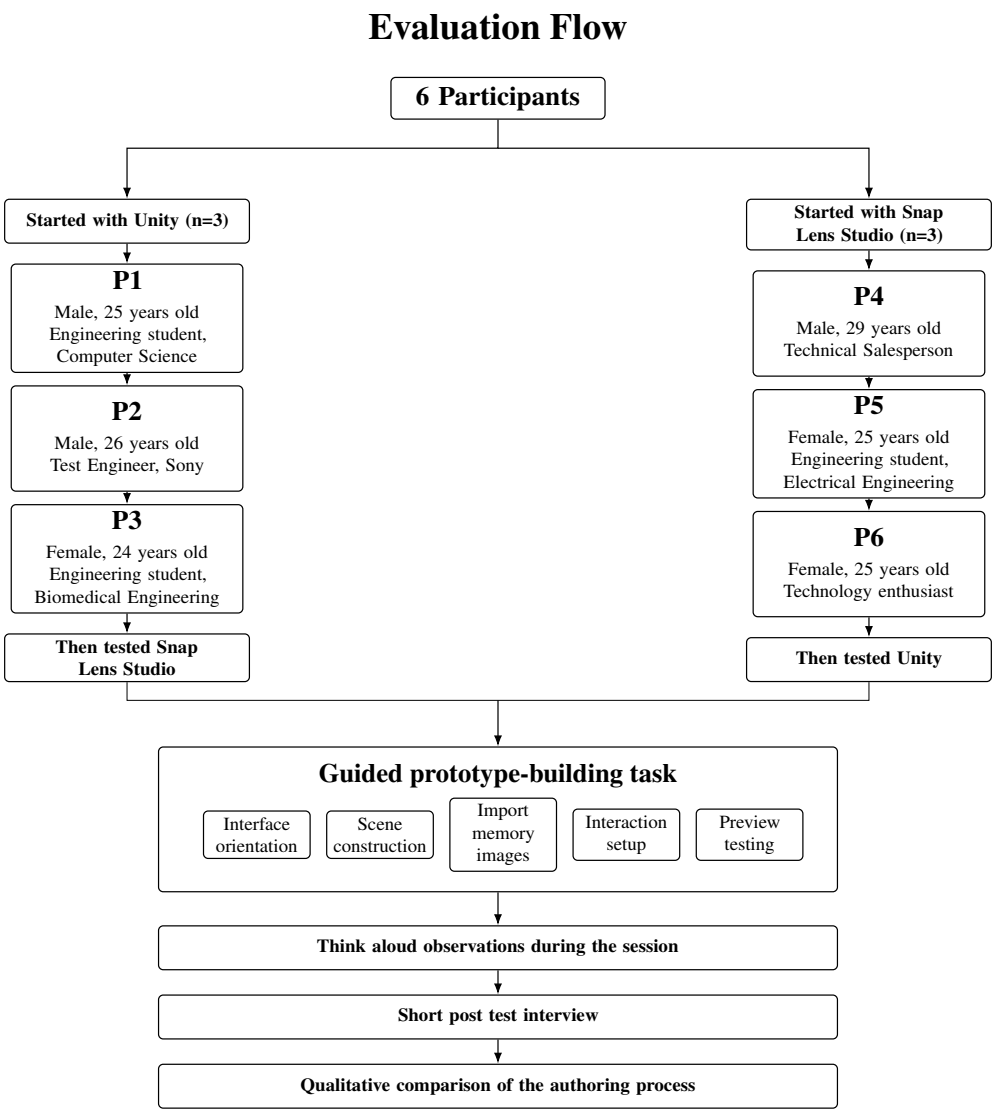

Figure 3.2: Evaluation flow showing the participant background and main stages of the test procedure.

4 Implementation

This chapter presents the implementation of the AR scene in the two selected AR authoring tools. It provides a detailed description of the development process in Unity and Snap Lens Studio, including the scene structure and interaction design.

4.1 Unity

The Unity interface is organised around four central windows that support the development workflow, see Figure 4.1. One of the most important windows is the Hierarchy, which displays all objects including the current scene. When an object is selected, its settings and attached components are shown in the Inspector Panel. The Scene view allows the developer to position, rotate and scale objects before running the application. In contrast, the Game view shows how the application behaves when it is executed, including effects of the scripts.

A Unity scene is built from different components, such as GameObjects, scripts, cameras, lights and media elements. GameObjects are the basic building blocks of a Unity project and can represent 3D models, text, images or other interactive objects. Scripts are written in C# and are used to define or extend the behaviour of GameObjects. Through scripts, objects can respond to input, move, change or trigger other events in the scene and make the prototype interactive.

Unity also supports the use of prefabs. A prefab is a reusable version of a GameObject that can be saved and used multiple times within a project. This is useful when the

same object or interaction is needed in several places without recreating it manually.

Moreover, a central part of the Unity interface is the Project Window, where imported assets are stored and organised. Assets can include 3D models, which can be dragged from the Project window into the Hierarchy or Scene view in order to add them to the active scene. Unity also supports external asset packages, such as *unitypackage* files, which can contain models, scripts or prefabs.

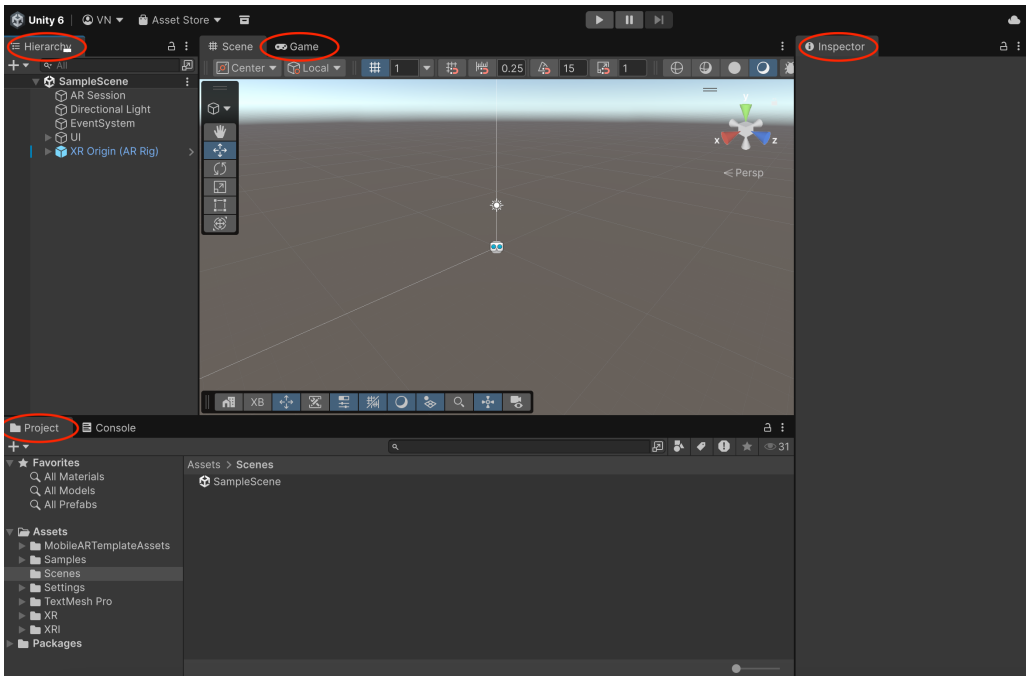

Figure 4.1: The project layout in Unity3D. Hierarchy is placed on the upper left. The project view is placed below and the Inspector Panel is placed to the right.

4.1.1 Project Setup in Unity

The implementation started by creating a new Unity project for mobile AR development that included an AR Session and an XR Origin. These components are central to mobile AR applications in Unity.

The AR Session controls the lifecycle of the AR experience and manages tracking

during runtime. The XR Origin represents the relationship between the physical device and the virtual Unity scene. Together, these components make it possible to connect the device camera and motion tracking to the virtual environment.

In the scene hierarchy, the main Unity scene was structured around several main objects. These included the AR Session, XR Origin, RoomRoot, Managers and MemoryCanvas. The RoomRoot object was used as a parent object for the virtual room which makes it easier to organise, move and scale the entire room environment as one unit.

4.1.2 Creating the Virtual Room Environment

After the project setup, the virtual room was created inside the Unity scene. The purpose of the room was to provide a familiar and emotionally meaningful environment for the reminiscence prototype. Since the prototype focuses on memory, a bedroom environment was chosen, as it can be associated with privacy, personal objects and individual memories.

The room was built using several GameObjects that included a floor, walls, furniture and decorative elements. In the hierarchy, objects such as *Floor*, *BackWall*, *RightWall*, *LeftWall*, *Furniture*, *Bed* and *Pillow* were used to form the environment. These objects created a simple but recognisable room structure.

The different parts of the room were positioned manually in the Unity scene view. The floor and walls were placed first in order to define the basic spatial layout. After that, furniture and decorative objects were added to make the room more complete. The objects were scaled and rotated so they matched the proportions of the room.

Materials were then added to the room objects in order to make the environment visually clearer. Separate materials were used for the floor, bed, wall, wood and frame elements. These materials were stored in the Materials folder in the Unity project.

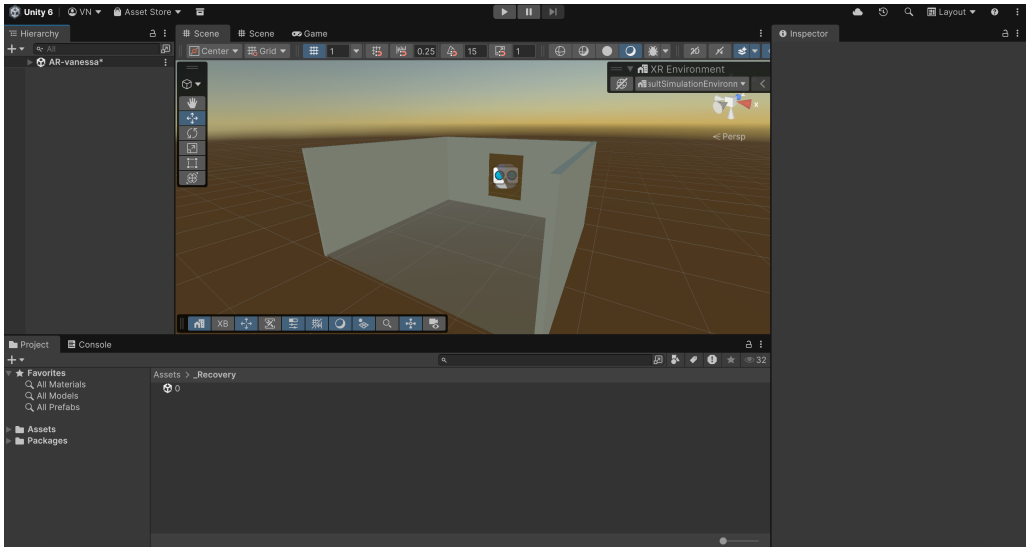

Figure 4.2: Unity scene showing the initial virtual room structure with a memory frame positioned inside the AR environment.

4.1.3 Organising the Scene Hierarchy

A clear hierarchy structure was important during the implementation as the scene contained several objects and interactive elements. The main objects were organised into parent objects to make the project easier to manage.

The *RoomRoot* object was used as the parent for the virtual room environment. This meant that the room could be adjusted as one complete unit rather than moving each object individually. This was useful during development because the room needed to be positioned correctly in relation to the AR camera and the user's viewpoint.

The *MemoryCanvas* object was used to organise the interactive memory content. Inside this object, a *PanelRoot* object contained the visual image elements and the click zones used for interaction. This structure separated the memory interface from the physical room objects, which made the project easier to understand and modify.

The scene also included a *Managers* object. This object was used to store scripts or logic components that controlled the behaviour of the prototype. Keeping the scripts

on manager objects helped separate interaction logic from the visual objects in the scene.

4.1.4 Adding Memory Frames

The memory frames were one of the central parts of the prototype. These frames represented digital memory objects placed inside the virtual room. Their purpose was to present reminiscence related images in a way that felt connected to the environment.

The memory frames were added as GameObjects in the room. They were positioned on the walls, appearing as framed pictures inside the bedroom. This design choice was made because framed photographs are commonly associated with personal memories and storytelling. By placing the images inside frames, the digital content became part of the room rather than appearing as separate floating media.

The frame objects were given specific materials to make them visually different from the wall and other room elements. A separate frame material was applied to create a clearer border around the images.

4.1.4.1 Importing Memory Images

The visual memory content was imported into Unity as image assets. These images were stored in the project folder named *MemoryImages*. Organising the images in a separate folder made it easier to replace and add new memory content during development.

The images were then applied to image objects inside the memory frame system. In the hierarchy, objects such as *SlideshowImage* and *SlideshowImage (1)* were used to display the current image in the memory frame. These image objects formed the visual part of the slideshow.

The use of a slideshow structure made it possible to include several memory images without needing to create a separate frame for each one. Instead, the same frame could display different images depending on the user's interaction.

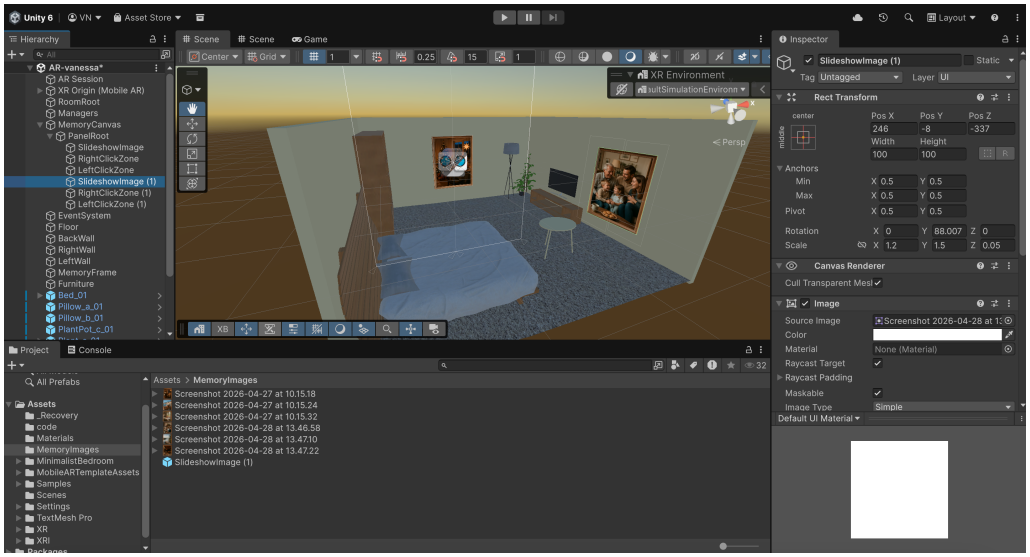

Figure 4.3: Visual of Inspector Panel of SlideshowImage (1).

4.1.4.2 Creating Interactive Click Zones

To make the memory frames interactive, invisible click zones were added to the image interface. These were placed on the left and right sides of the memory image and were named *LeftClickZone* and *RightClickZone* in the hierarchy.

The purpose of the click zones was to detect where the user touched the screen. If the user tapped on the right side of the image, the slideshow moved to the next image. If the user interacted with the left side, the slideshow moved back to the previous image. This interaction design was chosen because it is familiar to mobile applications, where users often swipe or tap left and right to navigate between images.

The click zones were placed as children under the *PanelRoot* object. This made them follow the position and scale of the memory panel.

4.1.4.3 Implementing Image Swipe Interaction

The slideshow interaction was implemented using a script that controlled which image was currently visible. The script stored an array of memory images and used an index value to determine which image should be displayed. When the user interacted with

the right click zone, the index increased and the next image was shown. When the user interacted with the left click zone, the index decreased and the previous image was shown. This interaction made the memory frame more dynamic. Instead of only viewing one photograph, the user could browse through several memory prompts.

4.1.5 Testing the Scene

The scene was tested continuously during development. The Unity Game view was used to preview the composition of the room and the memory frame interface. This allowed the scene layout, object positions and image display to be checked before testing on a mobile device.

4.2 Snap Lens Studio

Snap Lens Studio was used as the second authoring tool in the comparative study. While Unity represents a game engine based workflow with a high level of technical control, Snap Lens Studio represents a more visual and template based approach to AR authoring. Snap Lens Studio is designed for creating AR experiences, also referred to as lenses, that can be previewed and tested through the application Snapchat. The tool provides an interface where digital content and scene objects can be created and modified visually.

The implementation in Snap Lens Studio followed the same general concept as the Unity prototype. The goal was to create a reminiscence-based AR experience in which memory images could be displayed and interacted with. The prototype focused on importing visual memory content.

4.2.1 Project Setup

The implementation started by creating a new project where the project interface was organised around several important panels. For Snap Lens Studio the panels

were called differently but served similar functions; object panel, resource panel, inspector panel and preview panel. These panels support the main workflow of the implementation.

The Object panel was used to organise the scene hierarchy and manage the objects included in the prototype. Similar to Unity's Hierarchy window, this panel made it possible to structure the different parts of the AR scene. The Resources panel was used to store imported assets, such as image files and materials. The Inspector panel displayed the properties of selected objects, including position, scale, rotation and assigned materials. The Preview panel made it possible to test the prototype during development without immediately deploying it to a mobile device.

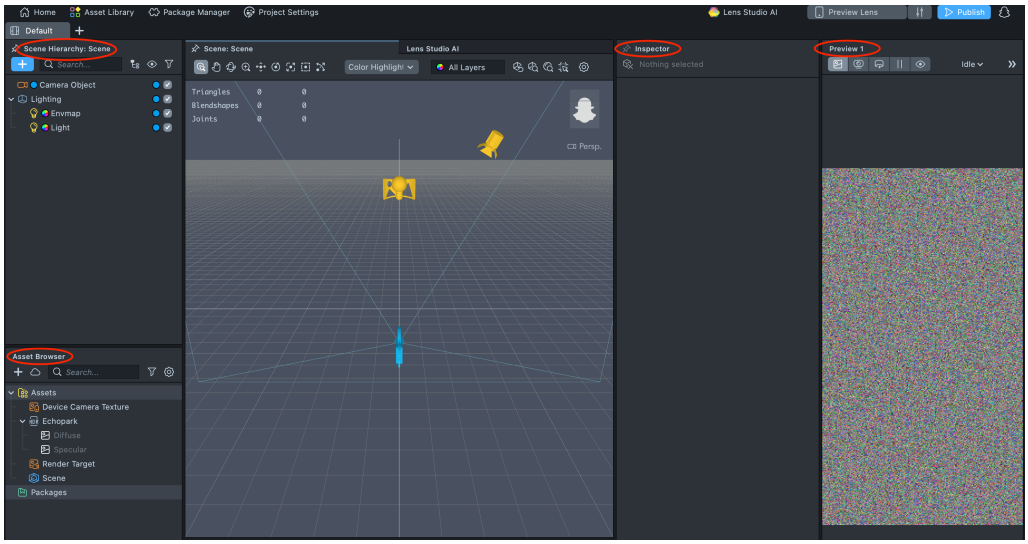

Figure 4.4: Snap Lens Studio project layout showing the Scene Hierarchy, Asset Browser, Inspector panel and Preview panel

4.2.2 Creating the AR scene

After the project had been created, the AR scene was structured around a camera based experience. Since the prototype was intended to present memory related visual content, the main focus was on creating a clear scene where images could be displayed and interacted with.

A main parent object was created to organise the visual memory content just as for Unity. This object functioned similarly to Unity where under this parent object, image objects were added to display the memory material. The scene was kept relatively simple in order to evaluate how efficiently Snap Lens Studio could support the rapid prototyping of the core concept.

4.2.3 Memory Frame

The memory images were imported into the Resources panel as image assets. Organising the images in the Resources panel made it possible to reuse them in the scene and assign them to visual objects. Each image represented a memory cue within the reminiscence-based prototype.

To display the images, plane objects were used as visual surfaces in the scene. The imported image assets were assigned as textures or materials to these objects. This made it possible to show the memory images as part of the AR experience. The prototype used one main display area where the visible image could be changed through interaction. This supported the same design idea used in Unity where one memory frame could contain several related memory prompts.

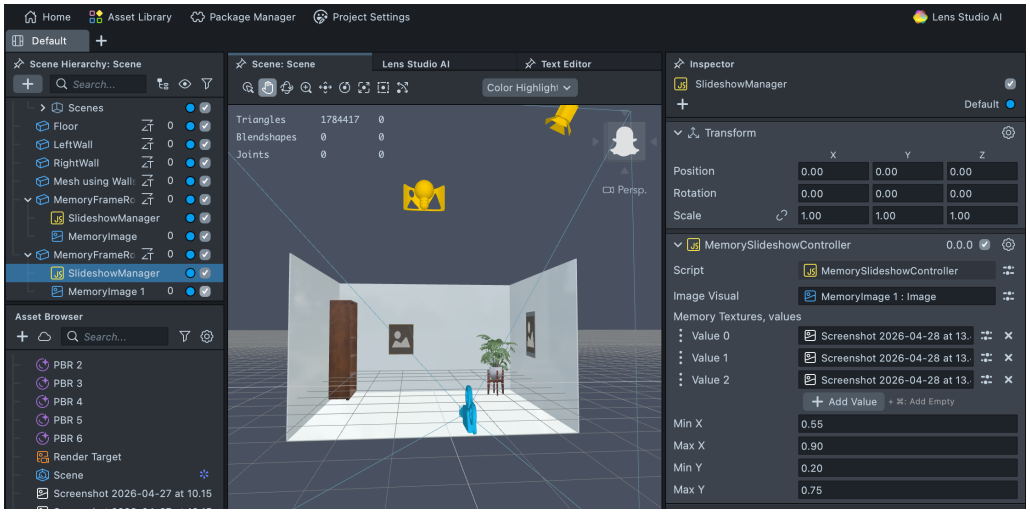

Figure 4.5: Visual of parent object MemoryFrameRoot with SlideshowManager.

A simple memory frame interface was created around the displayed image. The purpose of the frame was to make the image appear as a meaningful memory object rather than a detached visual element. In the Unity prototype, the memory images were placed inside a virtual bedroom as framed pictures on the wall. In Snap Lens Studio, the frame was implemented in a simpler way than for Unity, where images were placed inside a virtual bedroom as framed pictures on the wall.

The frame and image objects were grouped under the same parent object. This made the interface easier to move, scale and modify during development. By keeping the memory image and surrounding frame in one structure, the prototype remained organised and easier to adjust.

4.2.4 Interaction Logic

The interaction in the Snap Lens Studio prototype was created using the tool's visual scripting system. The goal was to allow the user to navigate between memory images through a simple tap interaction. In Snap Lens Studio, the interaction was visually represented through nodes. The visual scripting system made it possible to define events, connect them to actions and update the displayed image when the user interacted with the prototype.

The interaction logic followed the same principle as the Unity version. When the user interacted with one side of the image or triggered a forward action, the prototype changed to the next image. This created a simple slideshow interaction.

Testing of the interactions was done using the built-in preview function and mobile testing through Snapchat. The Preview panel allowed quick testing during development, which made it possible to check changes without performing a full build process.

5 Results

The following chapter presents the results of the study that consisted of implementation and verbal thinking with six participants that had some technical background. The prototypes are shown in the following chapter.

5.1 Unity

The Unity prototype was developed as a reminiscence-based AR room environment. The prototype consisted of a virtual bedroom scene containing walls, floor, furniture and memory frames placed inside the room. The memory frames functioned as interactive visual elements where the user could view and navigate between memory related images. The use of a bedroom environment was intended to create a familiar and personal context for the reminiscence experience.

The prototype was structured around several main objects in the Unity hierarchy, including *AR Session*, *XR Origin*, *RoomRoot*, *Managers* and *MemoryCanvas*. The *RoomRoot* object was used to organise the virtual room, while the *MemoryCanvas* object contained the memory image interface and interaction zones. This structure made it possible to separate the visual room environment from the interaction logic.

To make the frames interactive, invisible click zones were placed on both sides of the memory image. These zones allowed the user to move backward or forward between images. The interaction was controlled through a C# script that stored the memory images in an array and changed the currently displayed image based on index value.

5.2 Snap Lens Studio

The Snap Lens Studio prototype was developed as a second version of the same reminiscence-based AR concept. The goal was to create a prototype where memory images could be displayed and interacted with in an AR scene. Compared to Unity, the Snap Lens Studio prototype used a more visual and lightweight workflow. The implementation focused on importing memory images, placing them in the scene, creating a memory frame and implementing a simple slideshow interaction.

The project was organised using the main panels in Snap Lens Studio, including the Objects panel, Resources panel, Inspector panel and Preview panel. The Objects panel was used to manage the scene hierarchy, while the Resources panel contained imported image assets and materials. The Preview panel made it possible to test the prototype directly inside the authoring environment.

The memory images were imported into the Resources panel as image assets. Plane objects were then used as visual surfaces for displaying the images. Similar to the Unity prototype, the memory frame was designed as one main display area where the visible image could be changed through interaction. The frame and image objects were grouped together under the same parent object, which made the interface easier to move and adjust.

The interaction logic in Snap Lens Studio was created using visual scripting. Instead of writing C# code, the interaction was represented through nodes that responded to user input and changed the displayed image.

5.3 Comparative Data

During the implementation process, key notes were taken on the development time, technical complexity, workflow, structure, coding requirements and limitations. These results can be seen in Table 5.1 below and was documented prior to the evaluation test of participants.

Table 5.1: Development time for the implementation in Unity and Snap Lens Studio.

Implementation step	**Unity**	**SnapLens Studio**	**Included steps**
Project setup	3h 20 min	1h 50 min	Downloading software, creating project, configuring settings
AR setup	1h 30 min	50 min	Selecting template, AR origin, XR origin, camera setup
Scene structure setup	2h	1h 40 min	Creating parent objects, organising hierarchy
Building the AR environment	4h 50 min	3h 10 min	Walls, floors, scale and positioning
Importing image assets	2h 30 min	2h 10 min	Importing memory images, organising asset folder
Creating memory frames	1h 50 min	1h 20 min	Placing image surface, choosing materials
Creating slideshow/image switching logic	2h 20 min	1h 50 min	C# scripting, visual scripting
Adding interaction zones or input events	1h 50 min	1h 10 min	Click zones, tap input
Testing in editor/preview	50 min	30 min	Game View and Preview Panel iteration
Mobile testing	1h 20 min	1h	Device testing
Debugging and problem solving	2h 40 min	1h 40 min	Interaction errors, object placement issues
Final prototype review	40 min	30 min	Full prototype review
Total development time	**25h 40 min**	**17h 40 min**	

5.4 Prototypes

The AR experience created in Snap Lens Studio had its content built as a lens and connected directly to Snapchat through its own publishing and testing system. This made it possible to quickly open and test the AR function on a mobile phone through the Snapchat app. For the Unity project, an app based workflow was used. The AR scene was built in Unity with support for mobile AR where the project was then exported as an iOS project. The exported project was then opened in Xcode, where the application could be signed, connected to an iPhone and run directly on the mobile device. However, due to software update inconsistencies in the iOS phone, the prototype was primarily evaluated through Unity's Game view during development. The Game view is shown below to illustrate the prototype and its interaction setup.

Figure 5.1: Visual of the prototype in Snapchat with a rear view and front view.

Figure 5.2: The two interactive memory frames within the AR prototype.

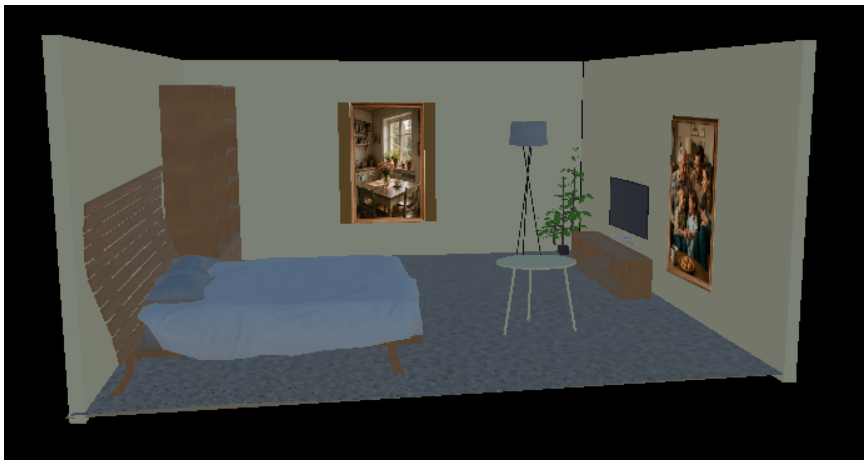

Figure 5.3: The prototype seen in Unity's GameView.

5.5 Cross Participant patterns

The evaluation consisted of six participants where three were female and three were male with a variety of technical skills. Three respondents studied in a technical field while the others had some knowledge of technology. Recruitment was done personally and all volunteered without incentives.

5.5.1 Interface Orientation

A recurring finding was that participants initially needed time to understand the structure of the authoring environment. Both Unity and Snap Lens Studio use several panels which was easily identified by all participants after some time. The Scene Hierarchy and Object panel was interpreted as the main place where the prototype was organised, while the Inspector panel became clearer once participants selected individual objects.

Before navigating the workspace, all participants asked for an additional explanation on the panels. After the explanation, four participants were able to understand the basic relationship between the scene view, the hierarchy and the asset browser. The remaining participants voiced that the interface was not fully understandable without prior experience in AR or 3D authoring tools.

Snap Lens Studio was perceived as more approachable during the initial orientation because the interface gave faster visual feedback through the preview panel. Unity, in contrast, required more explanation of each panel which in general made the interface orientation longer. Two participants had a hard time understanding the Game view which resulted in some tasks not being completed. However, observations show that the first three participants had an easier time understanding Snap Lens Studio after implementing in Unity first. Whilst the other participants started implementing in Snap Lens Studio first and then in Unity. These participants showed difficulties with Unity and needed more clarifications.

5.5.2 Scene Hierarchy

The scene hierarchy was one of the most important parts of the prototype building process but was also one of the most confusing parts according to three participants. Grouping objects under parent objects was easy to understand. A clear observation was that grouping the frame under a shared memory frame made it easier to move and scale the frame as one unite. However, several participants found it difficult to understand which objects belonged together and which object should be selected when making changes. A pattern that occurred among the participants, was when adjusting the memory frame. It was not clear at once whether to select the whole memory frame or only the image object inside which also made scaling parts of the prototype hard for some participants. Two participants voiced that selecting and adjusting the memory frame in Unity was easier as it supported a natural flow of the project setup. In general participants thought that the objects became more clear once the hierarchy was understood completely.

5.5.3 Importing Assets

Importing images was generally perceived as one of the easier parts of the task in which participants were able to understand that images had to be imported to Assets before using them in the scene. Once the images appeared in the asset list, all participants understood that could they be assigned to objects.

However, the step between importing an image and displaying it in the scene was less intuitive. Four participants expected that dragging an image directly onto the memory frame would automatically display it. Instead, the process required understanding the relationship between image assets, materials and textures. This was one of the first points where the authoring process became more technical according to two participants.

The image file having an object that displaces a certain material was not immediately clear to all participants. Two participants understood the process after one demonstration, while others needed repeated clarification. The task depended on the

ability to import and replace visual memory content efficiently. While Snap Lens Studio made image import relatively fast, the material and texture workflow still created confusion for participants. All participants agreed that changing an image file in Unity would be much more complicated as they associated the images to the required script. Two participants stated that the workflow was easier to understand with Snap Lens Studio and therefore made the importing assets easier because the objects were more visible.

5.5.4 3D Placement

The memory frame was one of the central elements of the prototype. All participants generally understood the idea of placing a framed image inside the room, especially because the metaphor of a photograph on a wall was familiar.

Positioning the memory frame in the 3D scene created more mixed reactions. One participant found it easy to move and scale the frame using the transform tools, while others found it difficult to align the frame correctly with the wall. The 3D space required participants to understand depth, camera perspective and object position. A participant voiced that this was not always intuitive, especially when the frame appeared correct from one angle but looked misplaced in the preview. A recurring difficulty was understanding the difference between the Scene view and the Preview view. Participants sometimes adjusted the frame in the Scene view and then became confused when it appeared differently in the Preview panel. Four participants observed that the GameView in Unity did not save changes on scaling and positioning which was confusing for most participants. Two participants agreed that 3D placement is a central challange in AR authoring, even when the idea is simple. Despite these difficulties, three participants were able to understand the purpose of grouping the frame and image together under one parent object. Once this structure was understood, it became easier to move the memory frame as a complete unit.

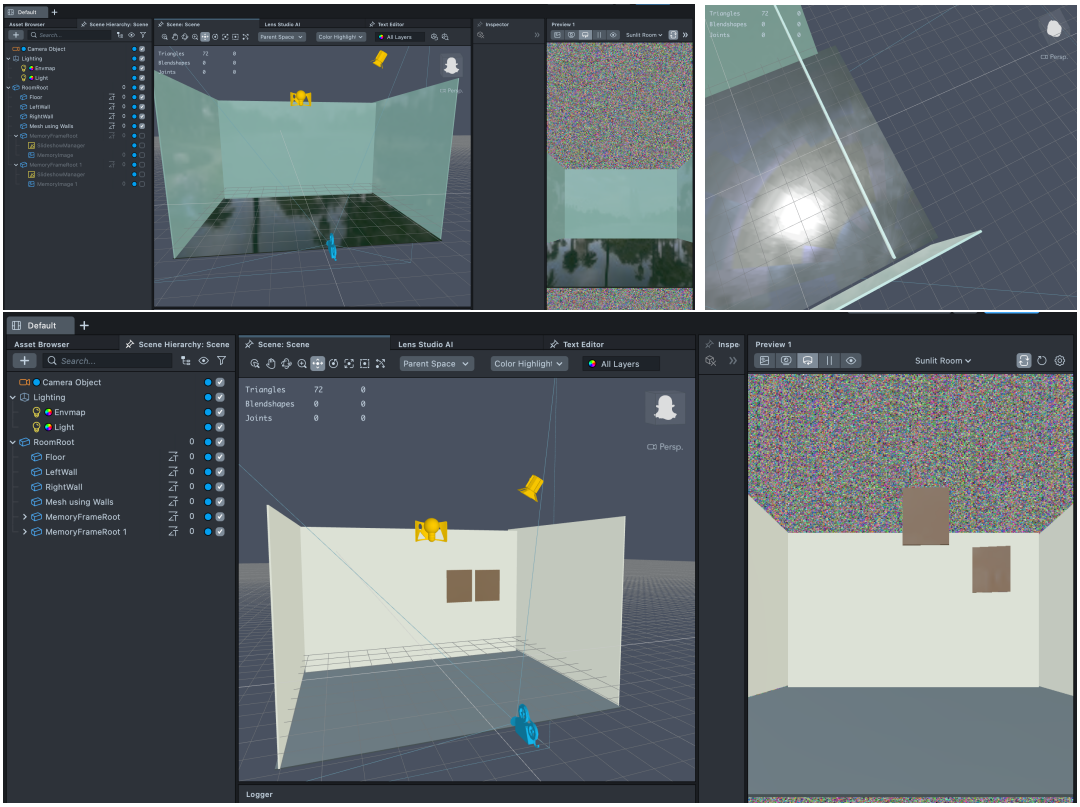

Figure 5.4: False alignment, where the room appears coherent from one view but is misaligned from another.

As shown in Figure 5.4, the participant experienced a mismatch between the Scene view and the Preview view in Snap Lens Studio. The frame appeared to be correctly aligned from one perspective, but the actual preview showed that it was placed at a different angle. This illustrates how 3D placement can become confusing when users have to interpret depth, perspective and object position at the same time. Similarly, participants experienced comparable difficulties in Unity, but expressed that the Game view panel made the process harder because changes in scaling and positioning were not always reflected as expected. This created uncertainty about whether the memory frame had been placed incorrectly or if it was caused by the difference between the editing view and the previewed result.

5.5.5 Interaction

The interaction logic was the most technically demanding part of the evaluation. All participants understood the intended interaction, of tapping either left or right which changed the image. However, the implementation behind the interaction was perceived as complicated since it involved scripts.

Several participants expected the interaction to belong directly to the memory frame, rather than being controlled through a separate manager object. This was handled by a script, and since the scrip was provided no participant was expected to code. This structure made the project more organised from a development perspective but was not immediately clear to all participants. The complexity increased when two memory frames were used, as both frames could react to the same tap if the script listened to the whole screen. However, only two participants managed to completely finish the task of placing an image and correctly applying the interaction logic of tapping.

Snap Lens Studio was perceived as more accessible than the traditional game engine workflow. Participants were more comfortable adjusting values in the Inspector, such as image lists or touch zones on Unity, than editing the script directly. The two participants expressed that Snap Lens Sudios' built-in code editor was more intuitive rather than opening a separate program for scripting in Unity. One participant added that script for Unity had to be saved in order to run the Game view, which confused the participant as the script could be assumed to be saved due to it being imported to the memory frame object. This was the task where participants spent the most time to understand and perform.

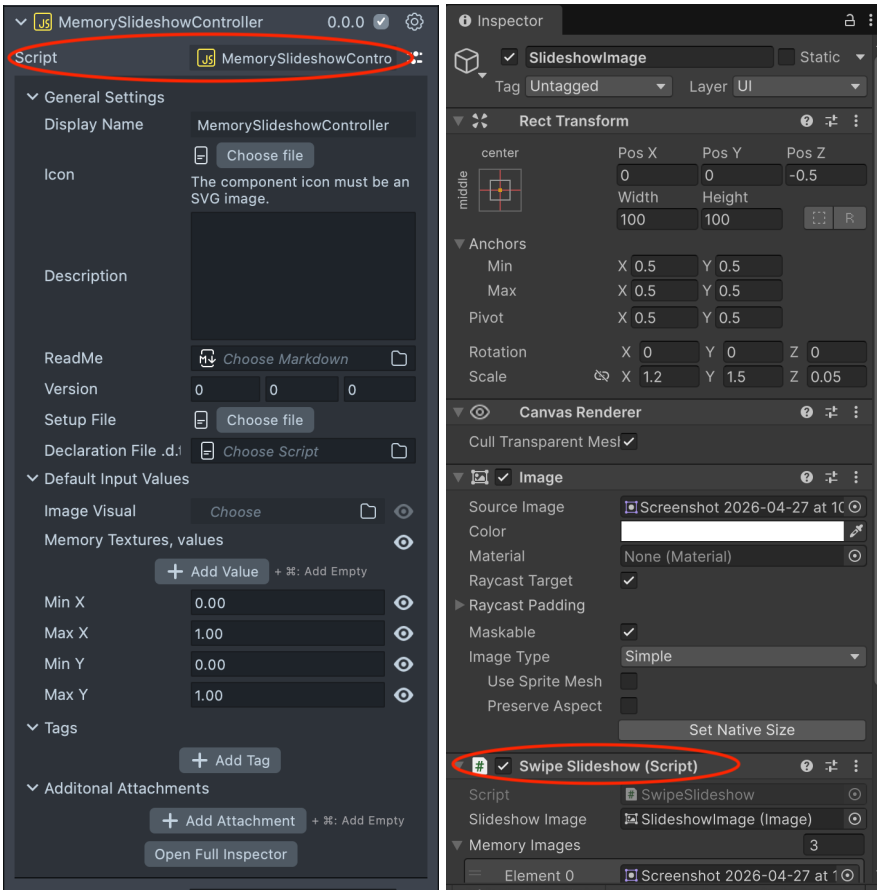

Figure 5.5: The interaction logic shown in the Inspector panel for Snap Lens Studio to the left and Unity to the right.

Figure 5.5 illustrates how the interaction logic was configured in Snap Lens Studio and in Unity. In Snap Lens Studio, the script code itself is not shown in the figure but instead the MemorySlideshowController appears as a script component where a text input panel is connected. The image visual component and less direct interaction with code made the interaction logic appear more configurable according to two participants. In Unity, the Swipe Slideshow script was attached to the SlideshowImage GameObject together with other components. According to participants, this required more understatement on how the imported script related to the user interface image component and the object hierarchy. Participants therefore found the interaction task

difficult not because they had to write code, but because they had to understand how scripts, objects and component references were connected. It was expected that the interaction would belong directly to the memory frame, which made the use of a script component less intuitive.

5.5.6 Participant Reflections

To further support the comparison between Unity and Snap Lens Studio, participants reflections were grouped according to how they described their experience with each tool. The citations were extracted from each session according to the software discussed by the participants.

Unity	Snap Lens Studio
"I could see that Unity was powerful, but I did not know where I was supposed to look at first." – P1	*"Snap Lens Studio was easier to understand because there were fewer distractions in the interface."* – P1
"Importing the image was easy, but making it show correctly was the confusing part." – P2	*"It was difficult to align the frame correctly inside the room."* – P2
"I expected the interaction to belong to the frame, not to a separate manager object." – P3	*"Importing images was faster, and changing images was easier because the objects were more visible."* – P3

> *"The hierarchy made sense after it was explained, but before that the objects felt hidden inside each other."* – P3

> *"The frame was easier to move once I understood how the objects were grouped."* – P4

> *"I understood that tapping should change the image, but I did not understand why the script had control."* – P5

> *"Duplicating the memory frame seemed easy at first, but the copied frame was still connected to the original."* – P5

> *"The moment materials were involved, it felt more like programming."* – P6

> *"It became difficult when two memory frames needed to work separately."* – P6

5.5.7 Overall Observations

Across participants, Snap Lens Studio was perceived as a faster and more approachable tool once the basic scene structure was understood. Three participants found the scene hierarchy in Snap Lens Studio easier to understand than in Unity, mainly because the interface contained fewer distractions and provided immediate visual feedback. This immediate visual feedback made the process feel accessible according to three participants as they could see progress early and understand how changes affected the prototype. Tasks such as selecting objects, adjusting room elements and testing changes in the Preview panel were completed more efficiently on both software.

However, the workflow became slower when the prototype required more complex activities. The main difficulties were not related to building the room or importing the images, but to connect the correct images and materials. Small mistakes such as assigning the wrong material or visual object to the script required careful thinking to avoid further confusion.

A recurring limitation was the management of duplicated objects. One participant successfully created two memory frames with the correct images. Duplicating a memory frame seemed efficient for the participant, but discovered that duplicated frames could still share materials and script references. This made it difficult to create multiple independent slideshows according to the participant.

Table 5.2: Overall observations based on evaluations.

Implementation	**Unity**	**Snap Lens Studio**
Interface orientation	Needed more explanation to understand interface. Relationship between the Scene view, Game view, Hierarchy and Inspector was not clear which made the initial stage of the workflow slower.	Easier to approach at the beginning. The Objects panel, Inspector and Preview panel gave faster visual feedback which helped participants understand the structure quickly.
Scene organisation	Provided a structured scene through parent objects however participants were unsure which object to select or edit. Took longer time to organise properly.	Perceived as more direct and less visually distracting. Participants found it easier to identify objects.
Importing and using assets	Applying imported assets correctly to objects was more difficult than understanding the concept. Participants needed to understand the relationship between materials and scripts.	Importing assets was faster and easier to understand. However, participants were still confused when images were not correctly connected.
3D placement	Allowed detailed control over object placement, but positioning frames and objects in 3D space was challenging. Struggled with camera perspective and the difference between the Scene view and Game view.	Placement felt more immediate because changes could be tested directly in the Preview panel. Precise alignment of memory frames still required careful adjustment.
Interaction logic	Most technically demanding part. Although participants understood the idea of clicking to change images, the C# script made implementation harder to understand.	Simple interaction testing through visual scripting and built-in previewing. However, more complex when creating multiple independent slideshows.

5.6 Participant Completion Time

The participant completion times were analysed to provide a descriptive overview of how much time was required to complete the guided prototype-building task in each tool. The times were converted into minutes before calculating the mean, variance and standard deviation. Since each evaluation session was limited to approximately five hours, the completion times should be interpreted as descriptive measures.

Table 5.3: Participant completion time for the guided prototype-building task.

Participant	**Unity**	**Unity, min**	**Snap Lens Studio**	**Snap Lens Studio, min**
P1	3h 45 min	225	3h 05 min	185
P2	4h 50 min	290	3h 50 min	230
P3	2h 55 min	175	2h	120
P4	5h	300	4h 30 min	270
P5	3h 05 min	185	2h 40 min	160
P6	4h 20 min	260	4h	240

The completion times shown in Table 5.3 were converted into minutes before the descriptive statistics were calculated. The mean, variance and standard deviation were calculated according to Equations 2.1, 2.2 and 2.3. These calculations provide a descriptive overview of the participant completion times for each tool and are summarised in Table 5.4.

Table 5.4: Descriptive statistics for participant completion time.

Tool	**Mean time**	**Variance**	**Standard deviation**
Unity	239.17 min	2794.17	52.86 min
Snap Lens Studio	200.83 min	3124.17	55.89 min

Figure 5.6 visualises the participant completion times for each tool. The graph shows that all participants spent more time completing the guided task in Unity than in Snap

Lens Studio.

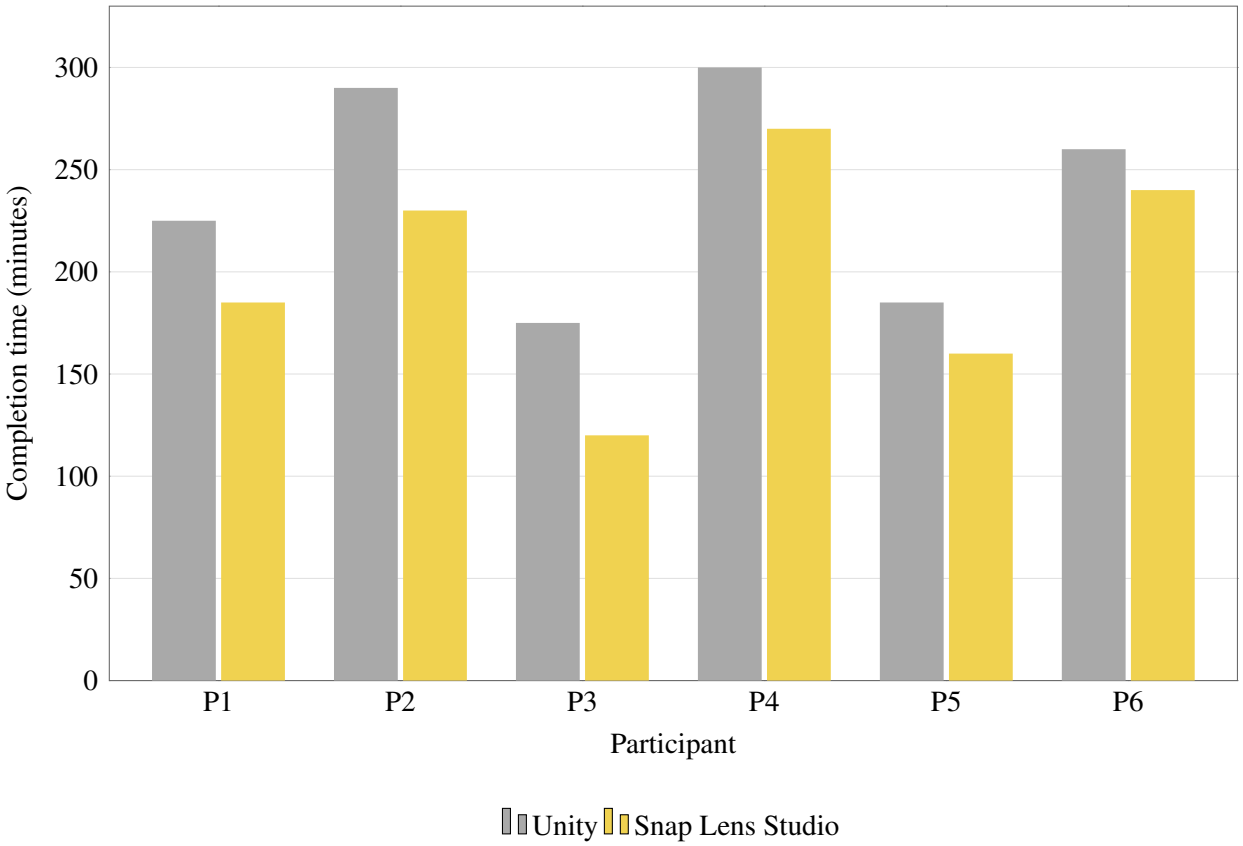

Figure 5.6: Participant completion time for the guided prototype-building task in Unity and Snap Lens Studio.

6 Discussion

This chapter discusses the key findings from the implementation and participant evaluation of the two AR authoring tools, Unity and Snap Lens Studio. The discussion is structured around the main comparison criteria of the thesis: development time, technical complexity, workflow and suitability for rapid prototyping.

6.1 The role of Authoring Tools

The findings show that the selected authoring tools did not only affect the time it took to build the prototypes, but also how the design process was experienced. Unity provided a more technical and component based workflow where the prototypes were constructed through GameObjects, materials and scripts. This made the development process more demanding but also provided great control over the scene.

The prototype building process for Snap Lens Studio was found different. Its visual interface, preview function and node-based interaction logic supported a more immediate prototyping form. This made it easier to move from concept to visual results. However, participant observations showed that the visual workflow did not remove complexity entirely as users still need to understand how objects were structured and how assets were connected.

AR authoring tools do more than provide technical functions. They influence what the designer pays attention to during the prototyping process. In Unity the workflow made the technical structure visible which allows the designer to focus on the components.

As a result, attention was directed towards structure and interaction logic. On the other hand, Snap Lens Studio had a workflow that instead directed the attention towards visual composition and previewing. Since many technical aspects were hidden behind templates and visual scripting, the focus was drawn more towards how the memory images looked and how the interaction felt. For this reason, the selected tools can affect not only the final prototype, but also influence where effort is placed during the design process.

6.2 Accessibility and Technical Complexity

A central aspect in the comparison was the relationship between accessibility and technical control. Snap Lens Studio reduced the need for coding and therefore made the prototyping process more approachable. This was especially visible in the early stages where importing images and placing objects could be done quickly. From a rapid prototyping perspective, this supported the fast exploration and made it easier to communicate the concept. Unity, on the other hand required more setup and a higher level of technical understanding. The use of C# scripts increased the complexity of the process where participants showed confusion at this step of the evaluation. However, this complexity also created more possibilities for interaction logic. A participant explained that the interactive zones in Unity were more comprehensible than the JavaScript code for Snap Lens Studio. The reasoning for this was that Unity had defined objects that solely supported the interaction logic which was then mentioned in the script. This made more sense to the participant as opposed to Snap Lens Studio where instead the scrip supported the interaction. Others disagreed and suggested that Snap Lens Studio was easier to understand because the visual workflow reduced the need to write code. For these participants, the node-based interaction felt more approachable. This shows that accessibility depends on the user's previous experience. Participants who were more comfortable with object based programming could understand Unity's interaction, while participants who preferred visual workflows found Snap Lens Studio more intuitive.

This difference is important because it shows that technical complexity is not experienced in the same way by all users. It is clear to state that coding can be a barrier, but visual scripting can also become confusing when the connection between objects and actions is not clear. The results show that a tool that is easier to start may become limited when the design becomes more specific, while a tool that is harder to learn may provide stronger support for later stages of development. In relation to rapid prototyping, this means that accessibility should not only be measured by how quickly a user can begin working in a tool. Snap Lens Studio supported a fast start into the prototyping process while Unity supported a deeper understanding of how the prototype was technically structured. The balance between accessibility and technical control therefore became one of the main differences between the tools.

6.3 Rapid Prototyping for an Iterative Process

As mentioned in the theory, rapid prototyping is not mainly about producing a finished product as quickly as possible but instead about creating an early version in order to test ideas and evaluate the interaction. In AR development, this is particularly important because the prototype must not only show the digital content but also connect this content to spatial placement. Therefore, speed is only one part of rapid prototyping. This process also depends on how easily the prototype can be tested, adjusted and developed.

The results of the development of the prototype as well as the evaluation support this understanding of rapid prototyping. Participants were able to see visible progress quickly in Snap Lens Studio as importing assets and positioning objects gave immediate feedback through the Preview Panel. The instant feedback supports the rapid prototyping and made the tool more approachable. This supports the theory of AR authoring tools stated earlier, where visual interfaces and templates are described as ways to reduce technical barriers. However, the results also show that Snap Lens Studio became less straightforward when the prototype did not involve visual placement. Table 5.1 clearly shows a difference in project setup time of the tools,

where Snap Lens Studio was notably faster. However, once the implementation involved building the AR environment, the gap between the tools became smaller.

Unity required more time and technical understanding at the beginning of the process which was also reflected in the participants evaluation. The interaction logic required more explanation for some participants which showed a steeper learning curve. Unity also supported another part of rapid prototyping which was the ability to refine and extend the prototype after the first version had been created. Although results present that this tool was more demanding, it shows that rapid prototyping is also a learning process where the developer gradually understands the structure of the prototype. The comparison therefore shows that the two tools support different forms of iteration within rapid AR prototyping.

6.3.1 Participant Completion Time

The participant completion times provide additional support for the qualitative findings regarding the prototyping workflow in Unity and Snap Lens Studio. As shown in Table 5.3, all participants spent more time completing the guided task in Unity than in Snap Lens Studio. The mean completion time was 239.17 minutes for Unity and 200.83 minutes for Snap Lens Studio which means that Unity required approximately 38 minutes more on average. The standard deviation was 52.86 minutes for Unity and 55.89 minutes for Snap Lens Studio, indicating a similar level of variation between participants for both tools.

These values support the qualitative results, where Unity was generally experienced as more technically demanding, while Snap Lens Studio was perceived as faster and more visually direct. The longer completion time in Unity suggests that its workflow required more time for interface orientation, scene organisation and interaction setup.

The difference between the two tools was also visible at the individual participant level. The largest difference was observed for P2, who spent 60 minutes more in Unity than in Snap Lens Studio. The smallest difference was observed for P6, where the difference between the two tools was 20 minutes. Although the size of the difference

varied between participants, the same pattern was present across all cases which was that Snap Lens Studio allowed participants to complete the guided task faster than Unity.

However, the completion time should not be interpreted as meaning that Snap Lens Studio is always the better tool for AR prototyping. Instead, the time data highlights the trade-off between speed and flexibility. Snap Lens Studio supported faster task completion because of its more visual workflow and Unity required more time but offered greater technical control and flexibility. This distinction is important because rapid prototyping is not only about completing the task quickly, but also about whether the tool supports the level of complexity needed for the prototype.

The mathematical results therefore strengthen one of the central conclusions of the thesis which is that the most suitable AR authoring tool depends on the desired balance between ease of use, flexibility and technical knowledge. Snap Lens Studio appears more suitable for early-stage prototyping when speed, visual feedback and accessibility are prioritised. Unity is more suitable when the prototype requires more advanced control and custom interaction logic.

6.4 Reminiscence as a Design Context

The created prototypes were not only a technical demonstration, but a memory oriented experience based on images. For this type of concept, early visual testing is valuable because it allows the designer to quickly see whether the memory frames communicate the intended idea. Snap Lens Studio therefore supports the initial phase of the design process. However, reminiscence-based AR also require careful adaptation to individual users as well as different media types which makes the technical control important in later stages of the design process. This makes Unity more suitable for a further development.

From this perspective, the results both support rapid prototyping and challenge the idea of rapid prototyping. Snap Lens Studio in particular demonstrates how immediate

previewing and visual workflows can support fast creations. However, when creating a more interactive prototype where connecting elements are needed, speed depends less on the interface and more on how clearly the tool supported adjustments. The findings show that the most suitable tool depends not only on which tool is fastest at the start, but on what kind of iteration the prototype requires over development time.

6.4.1 Sensitive Care Contexts

Although this thesis does not evaluate clinical outcomes, the palliative care context is important for interpreting the results. A reminiscence-based AR experience in this context would need to be understandable and adaptable to individual users. It would also need to handle personal memory material or other sensitive information. From this perspective, Snap Lens Studio is valuable for early concept generation as it allows quick understanding of the concept but Unity provides better support because of its greater control over interaction logic and content structure.

6.5 Limitations of the Study

This thesis focused on the authoring process rather than the clinical effect of the prototype. The participants were not patients and the prototype was not evaluated in a palliative care context. Therefore, the findings cannot be used to make conclusions about the effectiveness of reminiscence therapy of AR in care settings.

The number of participants was also limited to six people, which means that the findings should be interpreted as qualitative observations rather than generalised results. The participant sessions provided useful insight into workflow and tool usability, but a larger study would be needed to confirm other patterns.

A further limitation concerns the balance between ease of use, flexibility and technical knowledge in AR authoring tools. The study shows that this balance is difficult to evaluate in a general way, since it depends on the user's background, the complexity

of the intended prototype and the stage of the design process. Snap Lens Studio supported a faster and more visually accessible workflow, but became more limited when the prototype required customised interaction logic. Unity offered greater flexibility and technical control, but this came at the cost of more setup, scripting and technical understanding. Future AR authoring tools may need to combine the findings of Snap Lens Studio and Unity by offering beginner friendly visual workflows while still allowing users to access more advanced functionality when needed. This could be achieved through progressive complexity, where simple prototyping tasks remain accessible to non-programmers, while more advanced features can be introduced gradually as the user's needs and skills develop.

6.6 Answering Research Questions

The first research question was how the selected AR authoring tools differ in terms of development time, technical complexity, workflow structure, coding requirements and limitations. The results show that Snap Lens Studio required less development time than Unity and supported a more visual workflow. Its interface, preview function and visual scripting, made it easier to test and modify the prototype during development. Unity, on the other hand required more time for project setup as the software had more technical complexity. However, this offered more control over the scene structure and interaction logic as the development progressed.

The second research question asked how well each tool supports the implementation of core AR prototype features. Both tools supported the main features required for the prototype, including importing images, creating a virtual scene, displaying images and allowing interaction between the images. Unity supported features through a structured code-based workflow that was interpreted as more suitable contexts requiring detailed control. Snap Lens Studio was more efficient for creating and testing simpler prototypes with fewer interactive components.

The last research question asked which tool provides the most appropriate balance

between ease of use and flexibility for rapid prototyping. Based on the results, Snap Lens Studio provided the best balance for early-stage rapid prototyping because it reduced setup time and allowed the prototype to be created through a more visual workflow. Unity provided a better balance when flexibility and long term development were more important. Therefore, it is concluded that Snap Lens Studio is more appropriate for quick concept development.

6.7 Future Work

Future work can extend this study by testing the prototypes with larger and more diverse group of participants. In this thesis, the evaluation focused on users with some technical background. By including participants with less technical experience, such as designers and healthcare staff, a deeper insight into how accessible the tools are for different user groups could be studied.

Another aspect for future work can be to evaluate the AR reminiscence experience with stakeholder from healthcare professionals or patients that gives a comprehensive understanding of the design context and what concepts are more significant. Since this thesis did not evaluate the clinical outcomes, future studies could investigate how such an AR experience is perceived in a care context.

The prototype could also be developed further by adding more advanced interaction features. This could include voice narration, audio memories and music association. Such features would make it possible to study how different types of media and interaction influence the reminiscence experience.

Finally, future research could include additional AR authoring tools in the comparative analysis which would provide a broader understanding of how different authoring tools support rapid prototyping. All platforms could be compared again by using the same scenario with other findings which would help researchers or designers choose tools based on project needs and tool advantages.

7 Conclusion

This thesis compared Unity and Snap Lens Studio as AR authoring tools for rapid prototyping of a reminiscence-based AR experience in a palliative care design context. The findings show that both Unity and Snap Lens Studio can be used to create a reminiscence-based AR prototype, but support the process in different ways. The study focused on the development process by implementing the same AR concept in both tools. In this way, each platform was evaluated on how well they supported core prototyping tasks such as scene creation, asset import and workflow.

The findings show that Unity provided greater flexibility and technical control but also required more setup and technical knowledge. Snap Lens Studio provided a faster and more visual workflow with built-in preview tools and visual scripting that made the prototyping process more accessible. On the contrary, Snap Lens Studio was more limited when the prototype requires more detailed control or customised interaction logic. The participant completion times further supported this conclusion, as all participants completed the guided task faster in Snap Lens Studio than in Unity. However, the results also show that faster completion time should be understood in relation to tool flexibility.

Overall, the study shows that the choice of AR authoring tool depends on the purpose of the prototype. If the goal is to quickly explore an early concept or test a visual layout, Snap Lens Studio is more suitable. If the focus shifts to building more scalable and flexible prototype, Unity is the more appropriate choice.

References

[1] R. T. Azuma, "A Survey of Augmented Reality," Presence: Teleoperators and Virtual Environments, vol. 6, no. 4, pp. 355–385, 1997, doi:https://doi.org/10.1162/pres.1997.6.4.355

[2] M. Billinghurst, A. Clark, and G. Lee, "A Survey of Augmented Reality," Foundations and Trends in Human-Computer Interaction, vol. 8, no. doi: 10.1561/1100000049, Mar. 2015, Accessed: April 05, 2026, https://www.academia.edu/56839157/A_Survey_of_Augmented_Reality

[3] University of Southampton, "LibGuides@Southampton: Variance, Standard Deviation and Standard Error: Maths and Stats," library.soton.ac.uk, Apr. 12, 2024. https://library.soton.ac.uk/variance-standard-deviation-and-standard-error

[4] N. Ashtari, A. Bunt, J. McGrenere, M. Nebeling, and P. K. Chilana, "Creating Augmented and Virtual Reality Applications: Current Practices, Challenges, and Opportunities," Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, no. 9781450367080, Apr. 2020, doi:https://doi.org/10.1145/3313831.3376722

[5] N. Numan, G. Brostow, S. Park, S. Julier, A. Steed, and J. Van Brummelen, "CoCreatAR: Enhancing Authoring of Outdoor Augmented Reality Experiences Through Asymmetric Collaboration," Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, pp. 1–22, Apr. 2025, doi:https://doi.org/10.1145/3706598.3714274

[6] M. Beaudouin-Lafon and W. Mackay, "Prototyping Tools and Techniques." Accessed: April 06, 2026, https://www.kth.se/social/upload/52ef5ee4f2765445a466a28a/mackay-lafon-prototypes-52-HCI.pdf

[7] United Nations, "THE 17 GOALS | Sustainable Development," Un.org, 2015. Accessed: April 05, 2026, https://sdgs.un.org/goals

[8] M. Nebeling and M. Speicher, "The Trouble with Augmented Reality/Virtual Reality Authoring Tools," IEEE Xplore, Oct. 01, 2018. https://ieeexplore.ieee.org/document/8699236

[9] J. Blattgerste, J. Behrends, and T. Pfeiffer, "TrainAR: An Open-Source Visual Scripting-Based Authoring Tool for Procedural Mobile Augmented Reality Trainings," Mdpi.com, Apr. 03, 2023.

https://www.mdpi.com/2078-2489/14/4/219

[10] A. Lazar, H. Thompson, and G. Demiris, "A Systematic Review of the Use of Technology for Reminiscence Therapy," Health Education and Behavior, vol. 41, pp. 51S61S, Oct. 2014, doi:https://doi.org/10.1177/1090198114537067

[11] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: a class of displays on the reality-virtuality continuum," Telemanipulator and Telepresence Technologies, vol. 2351, Dec. 1995, doi: https://doi.org/10.1117/12.197321.

[12] S. Ullah and R. Ihsan, "A Survey of Augmented Reality Challenges and Tracking," Academia.edu, Feb. 09, 2014. Accessed: April 10, 2026, https://www.academia.edu/6012805/A_Survey_of_Augmented_Reality_Challenges_and_Tracking(accessedMay05,2026)

[13] "Spatial anchors," Unity Learn, 2026. Accessed: February 10, 2026, https://learn.unity.com/tutorial/spatial-anchors

[14] U. Technologies, "Unity-Manual: Transforms," Accessed: February 10, 2026, https://docs.unity3d.com/Manual/class-Transform.html

[15] M. Chmielewski, K. Sapiejewski, and M. Sobolewski, "Application of Augmented Reality, Mobile Devices, and Sensors for a Combat Entity Quantitative Assessment Supporting Decisions and Situational Awareness Development," Applied Sciences, vol. 9, no. 21, p. 4577, Oct. 2019, doi: https://doi.org/10.3390/app9214577.

[16] D. W. F. van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," Google.de, 2026. Accessed: March 08, 2026, https://scholar.google.de/citations?view_op=view_citation&hl=vi&user=4dIxTCwAAAAJ&citation_for_view=4dIxTCwAAAAJ:b1wdh0AR-JQC

[17] J. L. Bitter , "Interactive Storytelling in Immersive Augmented Reality: Supporting the Authoring Process with Software Tools," Jun. 2022. Accessed: March 05, 2026. Available https://hlbrm.pur.hebis.de/xmlui/bitstream/handle/123456789/159/ma_bitter_2022.pdf?sequence=4&isAllowed=y

[18] D. J. Hallford, D. Mellor, and R. A. Cummins, "Adaptive autobiographical memory in younger and older adults: The indirect association of integrative and instrumental reminiscence with depressive symptoms," Memory, vol. 21, no. 4, pp. 444–457, May 2013, doi:https://doi.org/10.1080/09658211.2012.736523

[19] S. H. Thorgrimsdottir and K. Bjornsdottir, "Reminiscence work with older people: the development of a historical reminiscence tool," International Journal of Older People Nursing, vol. 11, no. 1, pp. 70–79, Jul. 2015, doi: https://doi.org/10.1111/opn.12093.

[20] L. Xu, N. L. Fields, M. C. Highfill, and B. A. Troutman, "Remembering the Past with Today's Technology: A Scoping Review of Reminiscence-Based Digital Storytelling with Older Adults,"

Behavioral Sciences (2076-328X), vol. 13, no. 12, p. 998, Dec. 2023, doi: https://doi.org/10.3390/bs13120998.

[21] World Health Organization, “Palliative Care,” World Health Organization, Aug. 05, 2020. Accessed: March 10, 2026 https://www.who.int/news-room/fact-sheets/detail/palliative-care

[22] L. Radbruch et al., “Redefining Palliative care—A New consensus-based Definition,” Journal of Pain and Symptom Management, vol. 60, no. 4, pp. 754–764, 2020, Accessed: April 10, 2026, doi: https://doi.org/10.1016/j.jpainsymman.2020.04.027.

[23] V. Vaishnavi, B. Kuechler, and S. Duraisamy, “Design Science Research in Information Systems," Jan. 2004. Accessed: March 10, 2026 https://designsciences.org/design-science-research-in-information-systems.pdf

[24] “Overview | mind-ar-js,” Github.io, 2026. https://hiukim.github.io/mind-ar-js-doc/face-tracking-quick-start/overview

[25] “Welcoming WebAR Development Platform 8th Wall To Niantic,” Nianticlabs.com, 2022. Accessed: February 10, 2026, https://nianticlabs.com/news/welcome-8thwall

[26] “8th Wall - Open Source AR and 3D,” 8thwall.org, 2026. Accessed: February 10, 2026, https://8thwall.org/

[27] “Augmented Portal,” Docs and Tutorials, 2026. Accessed: February 10, 2026, https://docs.zap.works/studio/tutorials/augmented-portal/(accessedMay27,2026)

[28] Unity Technologies, "AR Foundation", Unity Documentation Available: Accessed: February 10, 2026, https://docs.unity3d.com/Packages/com.unity.xr.arfoundation%406.2/manual/index.html

[29] AR Core, “ARCore- Google Developers,” Google Developers.Accessed: February 12, 2026 https://developers.google.com/ar

[30] C. Li and B. Tang, “Research on The Application of AR Technology Based on Unity3D in Education,” Journal of Physics: Conference Series, vol. 1168, p. 032045, Feb. 2019, doi: https://doi.org/10.1088/1742-6596/1168/3/032045

[31] A. Ortega, S. Stumpp, and T. Knopf, “Augmented Reality (AR) Lenses: User Experience on Snapchat.,” Journal of User Experience, vol. 19, no. 3, Jun. 2024, Accessed: February 12, 2026, https://uxpajournal.org/augmented-reality-ar-lenses-user-experience-on-snapchat/

[32] “3D Object Export | Snap for Developers,” Snap.com, 2026. Accessed: February 12, 2026, https://developers.snap.com/lens-studio/assets-pipeline/3d/exporting-content/overview

[33] J. Carmigniani, B. Furht, M. Anisetti, and M. Ivkovic, "Augmented reality technologies, systems and applications," Springer Nature Link, Dec. 2010. https://doi.org/10.1007/s11042-010-0660-6

A Appendix

Evaluation Guide - Building an AR Prototype

In this session, participants follow a process to build a simple AR reminiscence prototype in Unity and Snap Lens Studio. The goal is not to create a finished application, but to explore how understandable, efficient and usable the authoring process is. You are asked to think out loud while working and describe what feels easy, confusing, unclear, slow or difficult. The study focuses on the prototype-building process, not on technical performance. The prototype includes:

- A simple room-like AR scene
- A memory frame placed inside the scene
- Imported memory images
- Tap interaction for changing images

Step 1: Open the Software and Project

1. Open the software.
2. Open the project file.
3. Look at the interface for a moment.
4. Identify panels such as:
   - `Scene Hierarchy / Objects panel`
   - `Asset Browser / Resources panel`
   - `Inspector panel`
   - `Preview panel`
   - `Scene view`

Step 2: Create the Room Structure

1. In the Scene Hierarchy, look for a parent object named `RoomRoot`.
2. Open it by clicking the small arrow.
3. Create objects named:
   - `Floor`
   - `LeftWall`
   - `RightWall`
   - `BackWall`
   - `MemoryFrameRoot`

Task 3: Adjust Object Properties

1. Click on `Floor`.
2. Look at the Inspector.
3. Change the position, rotation, scale, and material accordingly.
4. Repeat this for `BackWall`, `Bed`, and `MemoryFrameRoot`.

Task 4: Import Memory Images

1. Go to the Asset Browser.
2. Right-click inside the Assets area.
3. Choose `Import Files`.
4. Select three memory images from the provided folder.

Task 5: Create the Memory Frame

1. In the Scene Hierarchy, create an object named `MemoryFrameRoot`.
2. Create a child object named `SlideshowManager`.
3. Create a simple memory frame using a plane object for the image.

Task 6: Assign Image Material

1. Find the material assigned to `MemoryImage`.
2. Assign the first imported image as the texture.
3. Check that the image appears on the memory frame.

If a material is missing:

1. Right-click in the Asset Browser.
2. Create a new material.
3. Assign one imported image to the material.
4. Apply the material to `MemoryImage`.

Task 7: Add Slideshow Interaction

1. Create an object named `SlideshowManager` and insert the provided script.
2. In the Inspector, look for the slideshow script or interaction component.
3. In the Memory Textures list, add the imported images:
   - Value 0: `screenshot_01`
   - Value 1: `screenshot_02`
   - Value 2: `screenshot_03`

Task 8: Test the Prototype

1. Look at the Preview panel.
2. Press play or reset preview if needed.
3. Try the following interactions:
   - Tap right side of the frame
   - Tap left side of the frame
   - Swipe left
   - Swipe right

B Appendix

Interview Questions During Evaluation

Step 1: Open the Software and Project

- Do you understand where the objects are and edited?
- Do you understand where imported files are stored?

Step 3: Adjust Object Properties

- Is the scene hierarchy easy to understand?
- Is it clear which objects belong to the room?
- Is it clear how objects are positioned?

Step 4: Import Memory Images

- Was importing images easy to find?
- Was it clear where the imported images appeared?

Step 6: Assign Image Material

- Was it clear how an image becomes visible on the frame?
- Was anything confusing about connecting an image to an object?
- Was the connection between images and interaction understandable?
- Would this be difficult without guidance?

Step 7: Add Slideshow Interaction

- Did the interaction work as expected?
- Was it clear how to test the prototype?
- Did the Preview panel make testing faster?
- Did anything behave differently than expected?

Interview Questions After Implementation

- What part of the process was easiest?
- What part was most confusing?
- Did Snap Lens Studio feel suitable for rapid prototyping?
- Did the workflow feel visual and accessible?

- Did any part require technical knowledge or coding knowledge?
- Was the image interaction easy to understand?
- What limitations did you notice?
- Would you prefer this tool for building a quick AR prototype? Why or why not?