The Hitchhiker's Guide to XR

Novel Methods to Lower the Barriers for Novice Developers

Anil Gencay Erdem

DEPARTMENT OF DESIGN SCIENCES
FACULTY OF ENGINEERING LTH | LUND UNIVERSITY
2024

MASTER THESIS

The Hitchhiker's Guide to XR

Novel methods to lower the barriers for novice developers

Anil Gencay Erdem
anilgerdem@gmail.com

June 12, 2024

Master's thesis work carried out at
the Department of Design Science, Faculty of Engineering, Lund University.

Supervisor: Günter Alce, gunter.alce@design.lth.se
Examiner: Joakim Eriksson, joakim.eriksson@design.lth.se

The Hitchhiker's Guide to XR
Novel methods to lower the barriers for novice developers

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

Subject: Virtual Reality and Augmented Reality (MAMM15)
Division: Department of Design Sciences
Supervisor: Günter Alce
Examiner: Joakim Eriksson

Abstract

The growing popularity of Extended Reality (XR) attracts new people to enter the field. However, novices face challenges at the beginning of their journeys. Previous studies have identified the key difficulties based on real-world evidence and found that the barriers to entry were high. This thesis aims to guide novice XR practitioners in overcoming these challenges.

To achieve this, empirical insights were collected from experienced XR practitioners, visual scripting was investigated thoroughly, and Generative AI tools were tested. The findings revealed that starting the XR journey with a complete authoring tool could be an effective approach for beginners. Visual scripting could then enable novices to develop XR applications without extensive knowledge on coding, and Generative AI tools could serve as an efficient educational tool to enhance learning and skill development.

Keywords: Virtual Reality, Augmented Reality, Barriers to Entry, Visual Scripting, Generative AI

Acknowledgements

As this thesis concludes my master's degree at Lund University, I would like to express my deepest gratitude to several individuals who have been instrumental throughout this journey over the last three years.

First and foremost, I extend my heartfelt thanks to **Günter Alce**, **Mattias Wallergård**, and **Joakim Eriksson** for initiating the master's programme and fostering a friendly, supportive environment that has been incredibly conducive to my learning. Their mentorship and guidance have been invaluable.

I am also profoundly grateful to my former manager, **Bo Gabrielsson**, whose support has been crucial throughout my journey of learning and development. His trust in me empowered me to learn and grow, both academically and professionally.

My sincere appreciation goes to **David Fritz** and **Vijay Krishnan** for their flexibility and understanding, which have allowed me to effectively balance my work and academic commitments. Their support has made this journey much more manageable.

Last byt not least, I thank my beloved **Tety** for her patience and encouragement. Her support has been invaluable. I am deeply grateful for her presence throughout this journey.

Thank you all for being a part of this achievement!

Contents

Chapter 1
Introduction

This chapter begins by providing essential background information that guides the reader in identifying the research problem. Then, the research problem, objectives, contribution, and ethical and legal concerns are outlined.

1.1 Background

It was 1968 when Ivan Sutherland introduced the first head-mounted display (HMD) system [51]. However, it took 45 years until the technology became available to mass consumers at an affordable price thanks to Oculus Rift DK1 [44]. Two years after the success of Oculus Rift, Meta (Facebook at that time) acquired Oculus VR Inc. for US$2 billion in 2014 [14]. Since then, the field has seen noticeable technological improvements and commercial progress. At the time of drafting this thesis, Apple [4] officially released Vision Pro in February 2024, and two months later, Meta [32] announced that the operating system of Quest devices would be opened to third-party hardware makers. All these developments are signs of an exciting and promising future in technology.

Besides technological breakthroughs, people have witnessed numerous impactful incidents during the last decade. One such monumental event was the COVID-19 pandemic. This global crisis altered people's lives, norms, and interactions and influenced many industries, including Augmented Reality (AR) and Virtual Reality (VR). The World Economic Forum Global Future Council on Augmented Reality and Virtual Reality [9] has assessed the COVID-19 pandemic's impact on virtual and augmented reality. They concluded that the pandemic **changed the perception of VR and AR**, in particular, as an alternative to in-person interaction. It **has sped the adoption of immersive technologies by several years** [9]. Today, the VR and AR market is expected to grow at an annual rate of 10.77% (CAGR[1] 2024-2028) and is projected to reach US$58.1 billion by 2028 (see Figure 1.1) [56].

[1]Compound Annual Growth Rate

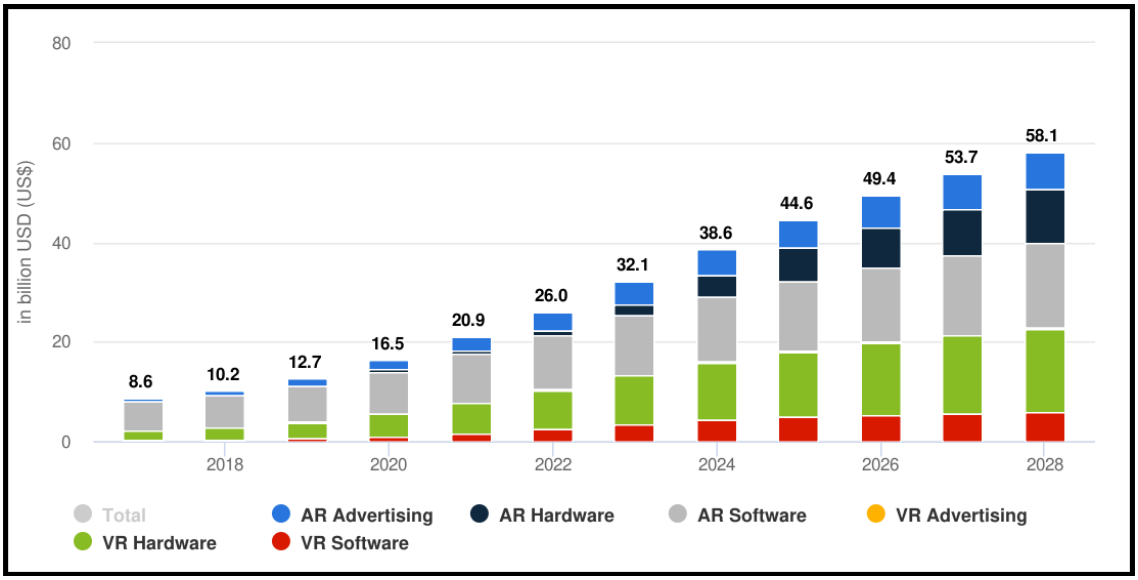


Figure 1.1: AR and VR revenue combined by market segments [56]

The author envisions that the need for skilled individuals proficient in developing, designing, and implementing these immersive technologies will be more critical than ever because the growth of the VR/AR market will surge the demand for professionals in the field. To fulfill such increasing demand, new professionals should enter the market, and people already in the sector should contribute more. However, the barriers to entry in the VR and AR field still seem to be high [5, 36].

The author believes that the high barriers to entry may discourage people, cause extended time and effort to acquire new skills, and make it difficult to scale the number of practitioners in the industry. The shortage of skilled people may slow the growth and adoption of VR/AR. Therefore, it is crucial to lower the barriers to entry. Once the VR/AR community identifies the hurdles of newcomers, those problems can be solved and frictionless developer onboarding and new skill acquisition can be achieved. Thus, reaching out to a bigger pool of talent will be easier, and more people will be able to join and contribute to VR/AR.

1.2 Research Problem and Objectives

The VR and AR technologies are still in their growth phase and trying to find the path to become more saturated and consolidated. Previous research shows [5, 26, 24, 36] that the field is highly fragmented, it suffers from the lack of standard guidelines, and it changes at a fast pace that tools become obsolete in a short time. This situation creates a moving target problem for professional and amateur practitioners because they must learn these tools and acquire new skills to adapt to the ever-changing industry dynamics. Lowering the entry barriers will not only widen the gates for newcomers but also give more opportunities and possibilities to professionals. This is because it was found that professionals and amateurs face common problems [26, 24].

This study aims to provide empirical insights from professionals, amateurs, and learners to capture the current VR/AR sector situation, the required skills to enter the sector, and

how practitioners could adapt to the fast-paced changes. It investigates whether the existing research successfully lowered the entry barriers and influenced the complete authoring tools to become more beginner-friendly. Lastly, to the author's best knowledge, only a few academic studies have examined the impact of artificial intelligence (AI) tools on VR/AR development. The latest Generative Artificial Intelligence (GenAI) tools are evaluated in this thesis to see if they could lower the barriers to entry.

The overall ***Research Objectives*** are:

- To explore the hurdles in learning Extended Reality[2] (XR) as a developer and the best practices to overcome these challenges.
- To explore novel approaches in XR and how these can lower the barriers to entry.

The research objectives are achieved by addressing the following ***Research Questions***:

- **RQ1:** What are the initial challenges and highlights when entering the XR field? How can a fresh entrant overcome these challenges?
  - What are the fundamental skills required? How effectively can people transfer their skills from other domains to XR?
- **RQ2:** What is the status quo of the authoring tools? How does the research on code-free XR authoring tools influence the current tools?
  - How effective are the visual (node-based) scripting in lowering the barriers for new entrants?
- **RQ3:** How effective is the GenAI tools in supporting novice XR practitioners and lowering the barriers to entry ?

1.3 The Sustainable Development Goals

The 2030 Agenda for Sustainable Development was adopted by all members of the United Nations in 2015, and it has 17 Sustainable Development Goals (SDGs) at its heart, from poverty to education to climate action [58]. VR/AR can help to pursue the SDGs because these immersive technologies have unique affordances such as embodied learning, social collaboration, and remote participation and presence [12]. The author believes that The SDGs #4 Quality Education and #9 Industry, Innovation and Infrastructure can benefit most from VR/AR technologies (See Figure 1.2).

1.3.1 Quality Education

The COVID-19 pandemic impacted our modern life in an unprecedented way. Digitization in many fields, such as education, medicine, entertainment, and commerce, was accelerated significantly [9]. Social and physical interaction has been altered and being remote (or online)

[2] See Chapter 3.1 for definition

Figure 1.2: SDG number 4 and 9 [58]

has become a regular part of people's lives. New ways of learning and medical care have emerged. For instance, 60 healthcare workers were trained on how to operate ventilators for COVID-19 patients by using Microsoft HoloLens in the Israel Center for Medical Simulation [9]. 185 first-year medical students took their anatomy class using Mixed Reality (MR) at Case Western Reserve University, and 85% of them reported that it was "equivalent to" or "better than" the in-person class [9].

These are a few instances of how VR/AR technology can revolutionize educational practices. The author believes that the immersive technologies can potentially turn traditional face-to-face education into remotely accessible and scalable interaction. This aligns with the SDG #4 Quality Education, which aims to "*ensure inclusive and equitable quality education and promote lifelong learning opportunities for all* [58]."

This thesis does not aim to develop an educational XR tool. But it contributes to the learning strategies needed to become an XR specialist. It aims to help novices become successful XR practitioners by lowering barriers to entry and making XR equitable. One step further, hopefully, some of these practitioners will develop educational XR applications to make education accessible.

1.3.2 Industry, Innovation, and Infrastructure

The SDG #9 is to Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation [58]. The author thinks that novel technologies like VR/AR or AI can improve efficiency through innovation. For example, manufacturing and maintenance operations can be optimized by real-time data visualization in VR/AR and by predictive modeling that uses AI/GenAI. Optimized resource usage can reduce the environmental

impact and enhance sustainability. Developing these applications is not the scope of this thesis. Instead, this thesis aims to demonstrate how it is possible to promote innovation by combining the latest technologies: VR/AR and GenAI. This study can inspire and support other developers, institutions, or organizations by showing novel approaches to innovate XR solutions for more sustainable industrialization.

1.4 Ethical and Legal Concerns

Immersive technologies collect and use a variety of unique data that cannot be captured by other technologies, such as physiological data, neural activity, surrounding details, or more [29]. All these data help the XR system to become more immersive and realistic. However, collecting massive amounts of data raises privacy concerns and brings more responsibility to the designers and developers. McGill [29] describes that such unique types of data may erode anonymity and privacy because those data can be used for user profiling and discrimination. It could be worse if the anonymous profiles in XR were connected to the user's real-world identity, and if this information was exposed, unwanted incidents might happen in the real world [29].

To protect people from undesired incidents, prevent the misuse of technology, and avoid legal breaches, it is crucial to understand the ethical and legal concerns. This thesis does not investigate ethical and legal concerns in particular. In this section, it was desired to raise awareness about data protection, privacy, and the use of Artificial Intelligence (AI) in the immersive technology domain. the author encourages readers to learn and follow responsible design/engineering principles. **It is a collective duty to create and maintain a safer *real* and *virtual* world!**

1.4.1 GDPR

Along with the ethical concerns, some laws regulate data collection, processing, and storage practices. The General Data Protection Regulation (GDPR) is a well-known European Union (EU) legislation. The GDPR mandates companies and developers to perform **data protection by design and by default** [54]. The author observed that collecting and processing more data sounds tempting for engineers and developers in the pursuit of creating better user experience. However, considering the data protection by design principle, the author recommends VR/AR practitioners to change their perspective to work with minimum required data. Collecting more data exposes more risks by expanding the possible attack surfaces. In this manner, the developer's principle should be **less is more**.

1.4.2 EU Artificial Intelligence Act

On 13 March 2024, the EU Parliament approved the AI Act [52]. This act aims to ensure that AI systems used in the EU are safe, transparent, traceable, non-discriminatory, and environmentally friendly [53]. For this purpose, AI systems are analyzed and classified according to the risk they pose to users [53]. And the regulations on risky systems are stringent. For example, AI tools that fall into unacceptable risk are considered a threat to people and will be banned [53].

The author notices that today's sophisticated systems, like AI, are inseparable from High-Tech industries and people's daily lives. AI systems are used to develop other technologies. Mobile phones, computers, or even simple applications take advantage of AI. As AI touches every part of life, lawmakers strive to enforce accountable use of AI. The author advocates that regulations like the AI Act put more responsibility on designers and developers to use AI more carefully. Developers should think twice before using any AI tools, and the design should prevent possible misuse. To create safer systems, **ethical AI should be prioritized**.

Chapter 2
Related Work

After an extensive investigation of the literature, the author decided to study the following three themes in this thesis: (1) barriers to entry, (2) authoring tools, and (3) GenAI tools. This chapter endeavors to synthesize and critically analyze the previous research related to these themes. Then, these findings builds the connection to the research problems.

2.1 Barriers to Entry

Ashtari et al. [5][1] classified and analyzed eight key barriers (See Table: 2.1) that VR/AR practitioners face from the perspective of three user groups:

- **Hobbyist** who tried VR/AR projects for their personal interest.
- **Domain experts** who tried VR/AR in their field as a new method, i.e., cognitive scientist uses VR to understand human behavior.
- **Professional designers** who create VR/AR products as part of their jobs.

They [5] explained that although the severity of problems could vary, these eight key barriers are common across all three user groups. Most of these issues are caused by the highly fragmented structure of the sector, the lack of best practices and guidelines, and the short life span of start-ups and tools [36, 24, 5, 26]. In such a chaotic ecosystem, hobbyists and domain experts have to put more effort than professionals at every stage of their development because they do not have the resources that professional teams have, such as previous project examples, technical materials, or guidance from expert colleagues [24, 26]. This situation may hinder the growth of the VR/AR community as the people who want to enter the field of immersive technologies may be intimidated. Moreover, professional teams also suffer

[1]Best paper award, Association for Computing Machinery (ACM) Conference on Human Factors in Computing Systems (CHI), 2020

Table 2.1: Eight Key barriers in creating VR/AR application [5].

Barriers in Understanding the VR/AR Landscape
1. Difficult to know where to start
2. Difficult to make use of online learning resources
3. Lack of concrete design guidelines and examples
Barriers in Designing and Prototyping VR/AR Experiences
4. Difficult to design for the physical aspect of immersive experiences
5. Difficult to plan and simulate motion in AR
6. Difficult to design story-driven immersive experiences
Barriers in Implementing and Testing VR/AR Applications
7. Too many unknowns in development, testing, and debugging
8. User testing and evaluation challenges

from these problems. Therefore, solutions to these hurdles would be beneficial for all XR community members: hobbyists, newcomers, professionals, and other stakeholders [26, 5].

To wrap up, previous studies have identified numerous reasons behind the barriers and proposed possible solutions. Although those studies identified issues based on real-world evidence, their proposed solutions were not based on real-world data but based on the personal opinions of the authors.

In this thesis, the author assumes these three main barrier categories: (1) barriers in understanding the VR/AR landscape, (2) barriers in designing and prototyping VR/AR experiences, and (3) barriers in implementing and testing VR/AR applications. These main categories were taken in a broader sense in this thesis and are the three pillars of the research conducted. For example, the investigation of authoring tools was divided into three parts: understanding, designing, and implementing. *Understanding* relates the general issues and solutions, *designing* elaborates how authoring tools may influence application design, and *implementing* focuses on the actual development phase.

This thesis builds on the literature and elaborates on the in the field previous research by adding empirical insights from the real world. These empirical insights will be presented in three main themes: understanding, designing, and implementing.

2.2 Authoring Tools

The current VR ecosystem was mainly established by the gaming industry members, such as game developers, designers, and software developers [24]. The influence of the gaming industry can be seen in the tools used for XR development. For example, Unity and Unreal Engine[2] are the most popular game development tools. They also dominate in the XR. Even though

[2]Unreal Engine is usually shortened as Unreal in this thesis

gaming practices can help with VR game development to some extent, those practices are inadequate for developing more generic enterprise VR experiences because VR development differs from traditional software and game development [24].

Designers, content creators, or non-technical novice practitioners have difficulty in prototyping and developing XR products because these popular game engines, i.e., Unity and Unreal, are geared toward programmers [36]. Previous studies [36, 5, 8] claimed that no-code/low-code authoring tools would enable novices to develop their XR applications and lower the barriers to entry **(Argument 1)**. In this context, no-code tools are the code-free tools that the practitioner can create VR/AR applications without writing any code. In addition, low-code can be defined as the requirement of basic or simple coding during the VR/AR application development.

Authoring tools in XR were investigated by Nebeling and Speicher [36]; they examined the main issues with the existing tools and found that there are significant gaps within and between tools. These gaps make the design cycle and content creation difficult. The available tool-chain somewhat works in an upward direction, from low-fidelity (lo-fi) tools to high-fidelity (hi-fi) tools, but not the other way around. This makes design iteration difficult, especially if the designer needs to go from the hi-fi to the lo-fi stage [36]. Even in the upward direction, the transition between lo-fi and hi-fi prototypes is not smooth. Medium to high fidelity transition often requires programming proficiency [8]. Nebeling and Speicher [36] proposed tools that make content creation and gesture interaction easier without coding. Their narrative was to build a no-code tool like PowerPoint but for AR. For example, ProtoAR enabled users to sketch an object on paper and then transfer this sketch into AR as an object [35].

The benefits of no-code tools cannot be denied for non-technical individuals. However, later studies [5] argued that even though no-code instruments help designers to create prototypes easily, they will make it harder for individuals to shift into more complete authoring tools like Unity and limit their outputs **(Argument 2)**. The author argues that this situation may not be an issue if a practitioner does not have the motivation or reason to develop complex systems on his/her own. However, suppose one aims to become a fully equipped VR/AR developer to land a job in the industry or wants to create usable XR products with minimum external dependency. In that case, working on complete authoring tools like Unity or Unreal (or similar) is inevitable **(Argument 3)**.

When these findings (**Argument 1**, **Argument 2**, and **Argument 3)** are put together, one might notice the problem: novices can start with no-code systems but they are not able to produce complex artifacts with these tools. Then, they shift to complete authoring tools for better outputs, however they cannot use the complete authoring tools because of programming requirement. Subsequently, they may return to no-code tools where they started their journey. This can be seen as a circle. The author called this as a vicious circle. In other words, there is a vicious circle between no-code tools and complete authoring tools. The author described this phenomenon as **the vicious circle of a novice XR developer** (See Figure 2.1).

Ashtari et al. [5] brought two ideas for a successful shift to complete authoring tools: (1) integrating no-code tools into complete authoring tools, and(2) adaptive user interface on complete authoring tools; personalizing the user interface's complexity according to the user's expertise. These two ideas are examined below.

Although various studies have been conducted on developing no-code tools, most were not evaluated in a realistic development cycle. For example, tools developed from research

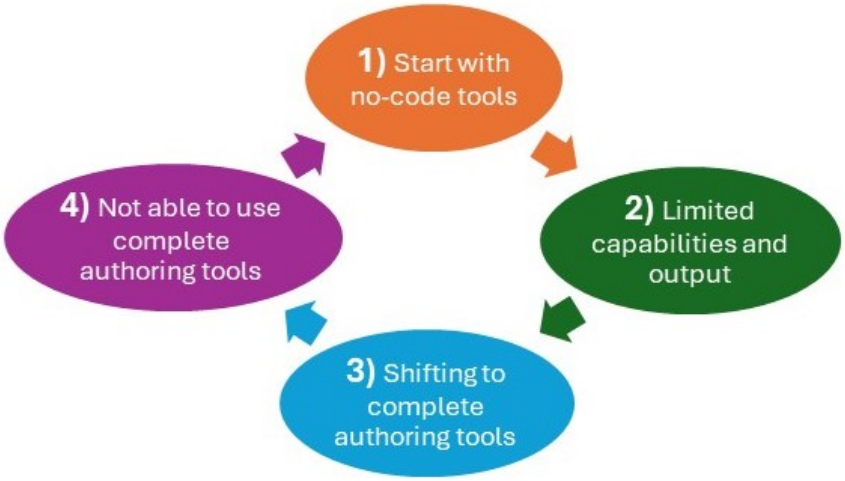


Figure 2.1: The vicious circle of a novice XR developer.

were not tested for their interoperability with complete authoring tools. This thesis addressed the first suggestion from Ashtari et al. and evaluated the integration of no-code instruments.

The second suggestion, personalizing the user interface according to expertise, did not materialize on popular authoring tools. For instance, a user cannot select a beginner or expert interface. The user can personalize Unity's or Unreal's interfaces by changing the location of windows or displaying/hiding sections. However, there is no personalization option to adjust the interface according to the user's competence level at the time of drafting this section. Moreover, the author could not find any research on interface personalization either.

Considering the user interfaces, one may ask if the complex user interface of game engines impedes the smooth development process. Usability assessment could answer this question; however, not much research was conducted on the usability of authoring tools that designers and developers use [30]. The limited number of research [30, 2] on the usability of Unity and Unreal Engine from the developer's perspective did not find any noticeable usability problems. In light of these findings, it could be concluded that the problem was not the user interface of complete authoring tools that hindered the development, as there are no significant usability problems. This conclusion would undermine Ashtari et al.'s recommendation for an adaptive user interface.

Yet, the hurdle of coding remains. The game engines offer visual scripting tools to tackle this coding obstacle and widen their target user groups. Unity Visual Scripting and Unreal Blueprint Visual Scripting have node-based user interfaces that allow users to create gameplay elements and interaction mechanics without writing any code [60, 61]. Figure 2.2 shows the node-based visual scripting interface.

Visual scripting could be an excellent option to break the vicious circle. It could help transition from no-code to a complete development environment or enable newcomers to start from complete authoring tools immediately. To the author's best knowledge, no research has been conducted on visual scripting, nor has it been conducted on whether visual scripting features could allow fresh entrants to create a sophisticated VR/AR artifact. This study systematically investigates to what extent these tools can enable novices to develop essential XR experience.

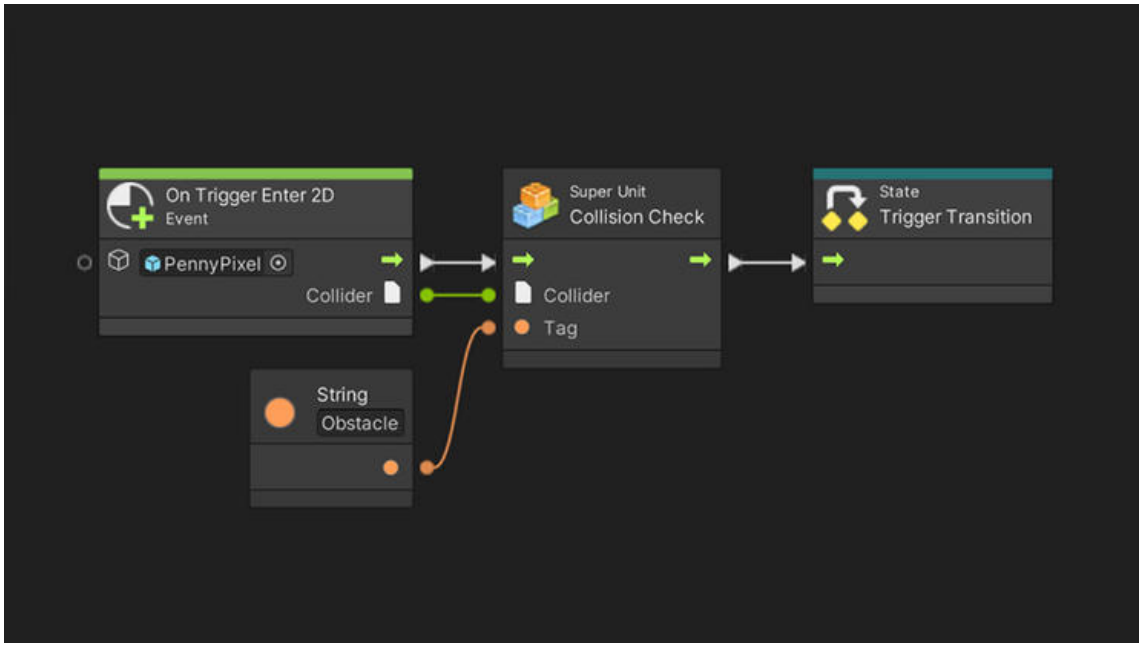


Figure 2.2: An example from Unity Visual Scripting [60].

To wrap up, no-code authoring tools can potentially enable people without programming skills to create simple VR/AR artifacts and prototypes. However, one may need complete authoring tools like game engines to develop more complex projects. In this study, the author reviews the recommended tools from previous research and checks whether they can be used for complete VR/AR development. He also assesses the visual scripting tools to see if they can enable individuals without programming skills and game engine knowledge to create VR/AR solutions and break the vicious circle.

2.3 Generative AI Tools

Each industrial revolution had its defining breakthroughs: first steam-powered mechanization, second electricity-powered mass production, and third computer-powered automation. Today, the fourth is defined by machine intelligence powered by unprecedented access to terabytes of data [11]. In the age of Industry 4.0, Artificial Intelligence (AI) has not only been hyped but become people's reality and an integral part of their daily lives.

AI was a significant leap for humans. And within AI, the introduction of ChatGPT can be seen as one of the most disruptive technological breakthroughs. ChatGPT is a conversational AI model that allows a user to interact with it in a natural dialogue format, and the user can accomplish various tasks through open conversation [40]. It is one of the most successful product launches of modern times. It reached 1 million users in 5 days and 100 million in 60 days, which was an unprecedented market penetration [19]. (See Figure 2.3)

The inauguration of ChatGPT raised great interest in AI from the general public. This can be seen in Google search trends. For instance, *artificial intelligence* had somewhat stable Google trends before ChatGPT. The trend tripled in a year with ChatGPT's introduction (See Figure 2.4) [17]. Today, millions of people are asking questions to AI tools, generating artistic images or completing various tasks with the help of GenAI. Institutions pub-

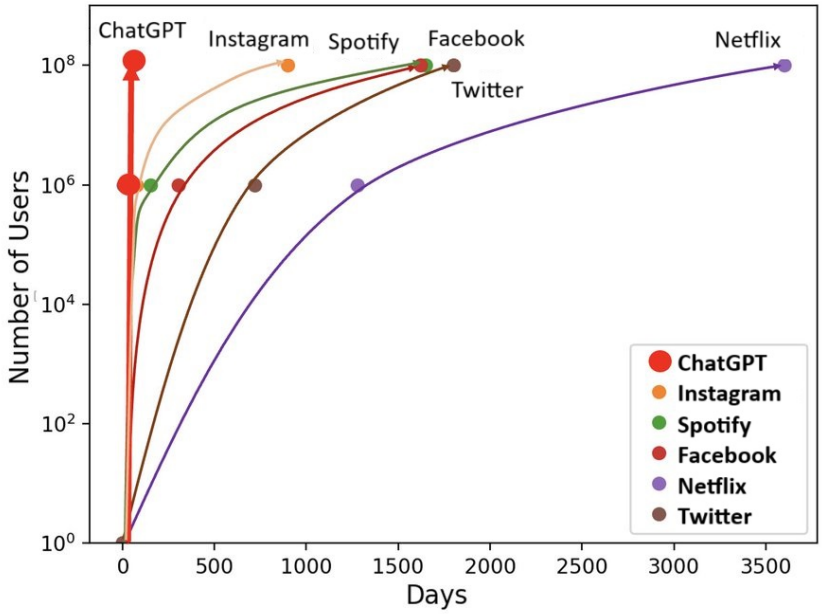


Figure 2.3: ChatGPT reached 100M users in 60 days [19].

lish guidelines and rules on using ChatGPT. For example, Lund University [57] published an article *"Four things to keep in mind before using ChatGPT"* one year after the introduction of ChatGPT.

GenAI was adopted quickly and is already part of people's lives. Despite the tremendous interest from the public, there are also concerns about AI. There have been rumors circulating about whether AI will take jobs or eradicate humanity. NVIDIA CEO Jensen Huang [6] said that traditional programming skills will become unnecessary in the future, and AI will handle the hard work of writing code during his keynote speech at the Global GPU Technology Conference. Stanford Professor and AI pioneer Andrew Ng [37] debunks these concerns and recommends using more AI to solve our problems instead of limiting or banning it. He advocates that AI is an excellent tool to support humanity and that AI will not take our jobs; however, people who do not use AI will fall behind those who use AI [37]. In other words, he believes that people who use AI as a supportive tool will become more successful than others.

Various AI tools have been developed to assist people in the last decade. These tools were adopted rapidly and widely used in almost all industries. At the time of writing this chapter, OpenAI introduced its latest GenAI model, Sora. It can generate realistic videos from text prompts [41]. Before Sora, the text-to-image model DALL·E was at the center of public attention. The latest version of DALL·E 3 could understand nuances and details significantly better than its predecessor, and it could generate exceptionally accurate images [39].

Another example in computer graphics is NVIDIA instant NeRF (neural radiance field), an AI-based rendering tool that can create a 3D volumetric scene from 2D images in a couple of minutes [49]. 3D scene generation from 2D images was possible before. But it was a lengthy and heavy process. This instrument could accomplish the heavy lifting of realistic scene generation and streamline the development pipeline in VR [42]. It is an excellent instance of a supportive AI tool.

Researchers [42] believe that with the introduction of accurate GenAI models such as

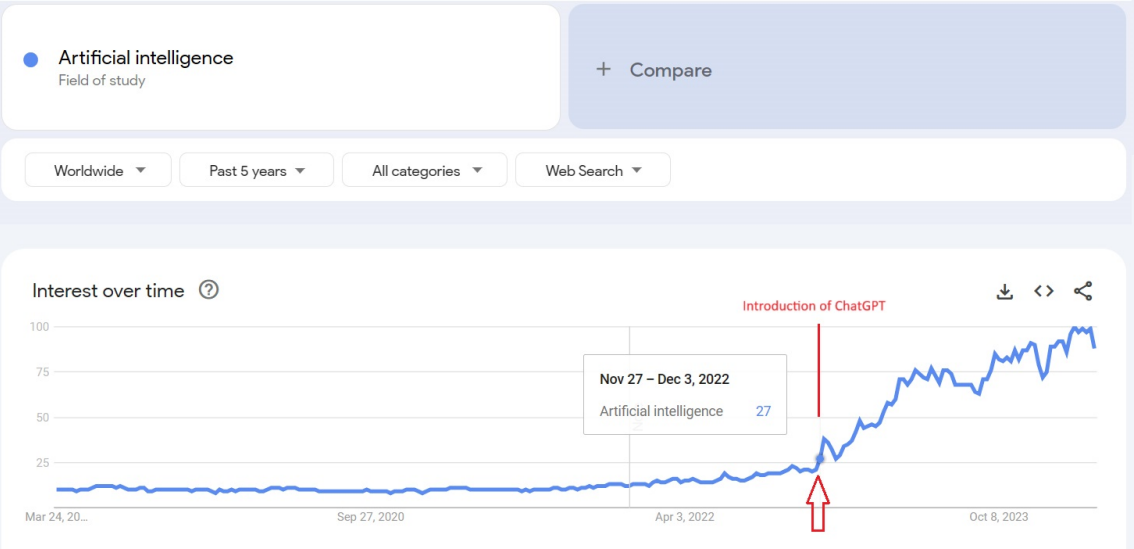


Figure 2.4: Worldwide Google search trend of "Artificial Intelligence" over the past five years [17].

DALL·E 3 or Stable Diffusion, one could generate 2D images, which will be later fed into NVIDIA instant NeRF to create 3D scenes without going through the traditional cumbersome 3D modeling pipeline. Such a process would make it possible to produce 3D scenes almost without limits and democratize content creation by enabling the general public to produce 3D artifacts with simple prompts [42].

Independent researchers [3] developed 2D games in three different genres on Unity by using ChatGPT and then compared these games against baseline games and human-designed games. Subsequently, they created an online platform where people could play and rank these games. Although human-designed games consistently ranked higher than baseline and ChatGPT-designed games, GenAI was able to produce entirely satisfactory games [3]. Even though AI failed to create a consistent and complete game narrative itself when human designers evaluated the AI-designed games, they admitted that AI had a broader scope of imagination, i.e., designers had their own patterns and biases and failed to *think outside the box* [3].

The AI is a relatively mature technology compared to GenAI, which is still very early in its adoption phase. However, it is already clear that AI-related technologies will change how people work in many industries. The VR/AR field will inevitably be impacted by these developments. The author foresees that the wave of changes is approaching and will alter the practices in the XR domain. The first disruptive news had already arrived. Unity [63] announced its new AI tools, Unity Muse and Unity Sentis, in June 2023. Unity Muse is an integrated AI-assistance platform that can be used to write code and create animations or more [63]. Unity Sentis is a tool to integrate AI capabilities into gameplay to create AI-powered characters or interactions [63].

As Andrew Ng said, [37], people who use intelligent systems will be more successful than others who do not. The author envisions that this also applies to the people in the VR/AR industry and those who intend to enter. The newcomers who use GenAI will be able to produce higher quality results quicker than the ones who do not.

Despite its great potential, a limited number of systematic and academic studies have been conducted on this topic. Few works have been done on using ChatGPT in game devel-

opment, and no study has been conducted on Unity Muse.

GenAI can support VR/AR professionals in more generic content/object creation, design, coding, debugging, customization, and optimization [3, 20, 10]. The author believes AI can be seen as an open ocean with many opportunities waiting for exploration. In this research, he attempts to sail into this ocean and explore it. This thesis will be one of the early works focusing on GenAI usage in XR.

2.4 Research Significance

Previous research could identify several major hurdles one faces when entering the VR/AR industry. These issues were also consistent across multiple studies. This thesis assumes these hurdles exist and does not attempt to discover new problems or validate them again. The objective is to find the best practices from other learners and industry. These best practices can extend the recommendations from previous works.

Another point was that researchers developed new instruments to tackle specific problems according to their research objectives, i.e., an AR content creation tool and a software development framework. Unfortunately, these efforts could not be unified, leading to further fragmentation of the VR/AR domain. For instance, more than 40 authoring tools were identified while reviewing the literature in Section 2.2. Most of these tools did not become viable products to support creators. Moreover, it was also found that neither professionals nor hobbyists use research-style tools in actual VR/AR development [5, 26]. The author interpreted this situation as another evidence of fragmentation. To tackle this, the author intends to review all the authoring tools mentioned in previous research. Creating a comprehensive list of tools and conducting a systematic and thorough review can be a reference for future studies. It can also guide novice users when they search for tools.

One of the fundamental outputs of previous research on lowering the barriers to entry was recommending authoring tools that did not require programming skills. However, as mentioned above, previous works did not trigger the snowball effect to unify or integrate various tools to develop a complete authoring tool without coding. Consequently, *the vicious circle of a novice XR developer* (See Figure: 2.1) has emerged. This research contributes to the field by addressing two instruments to break out of this vicious circle: Visual scripting and GenAI.

Popular game engines Unity and Unreal offer node-based visual scripting tools. Unfortunately, little research exists on how effective these tools are in lowering barriers to entry into VR/AR. This study aims to fill this gap by investigating if visual scripting could assist novice XR practitioners.

Finally, this thesis contributes to the field by exploring novel approaches to address the research problem. Since GenAI is a fresh area, there is almost no systematic research on combining GenAI and VR/AR development. This study would be one of the early examples of GenAI usage to lower the barriers to entry in XR development.

Chapter 3
Theory

This chapter defines the technical terminology, elaborates on data collection techniques, and explains theoretical background of these techniques.

3.1 Definitions

Although VR and AR became prominent in the last decade, these technologies have a history of over half a century. Sutherland's [51] head-mounted display (HDM) marked the beginning of a new era. Almost 25 years after Sutherland's HDM, Milgram and Kishino [34] introduced the reality-virtuality continuum (See Figure 3.1). This was a theoretical framework to visualize and compare different technologies according to their immersion capabilities. It can be seen as a spectrum of immersion where one end is 100% real/physical environment with no immersion, and the other end is 100% virtual environment in which the user is fully immersed.

The virtuality continuum has been very useful in creating a common language and representing immersive technologies. However, science has advanced tremendously since its introduction; new technologies emerged, some concepts changed, and some practices became obsolete over time. In order to accommodate these updates and represent today's technology,

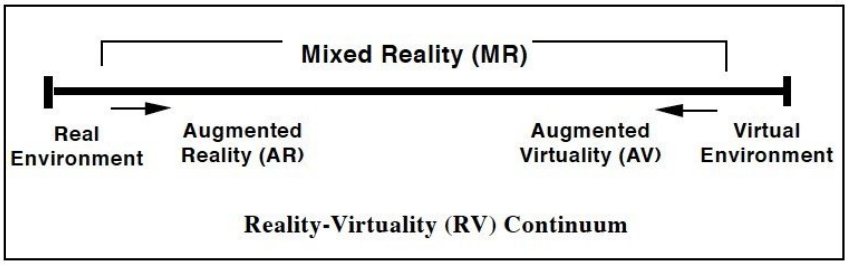


Figure 3.1: Representation of Reality-Virtuality Continuum [34]

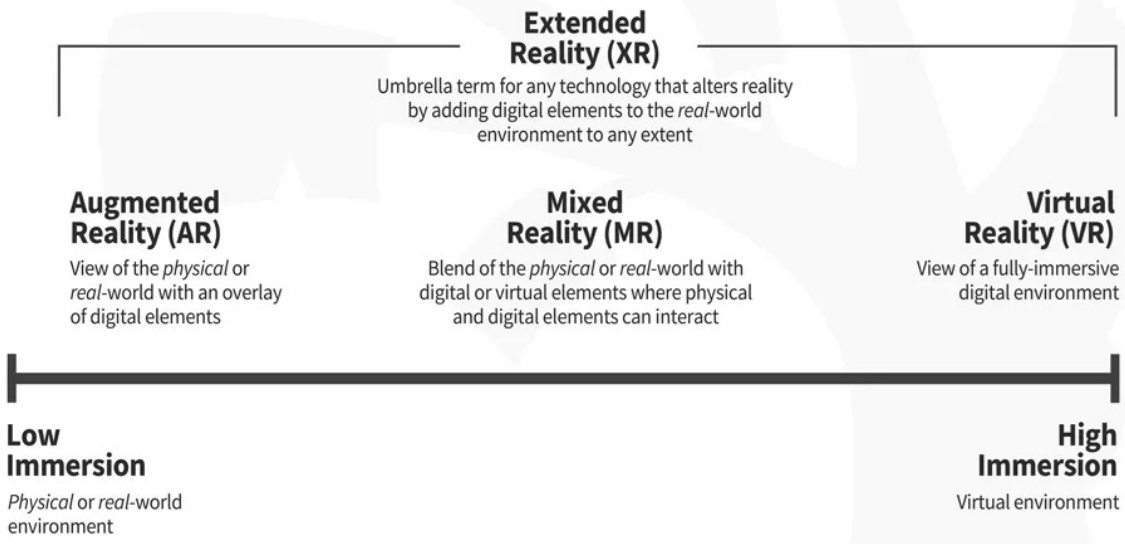


Figure 3.2: Current XR technologies according to the spectrum of immersion [55].

Interaction Design Foundation [55] proposed an updated version of this immersion spectrum (See Figure 3.2) using the following terms:

- ***Augmented reality (AR)****: a view of the real/physical with an overlay of digital elements.*
- ***Mixed Reality (MR)****: a view of the real/physical world with an overlay of digital elements where physical and digital elements can interact.*
- ***Virtual reality (VR)****: a fully-immersive digital environment.*
- ***Extended reality (XR)****: an umbrella term that covers all these different technologies, including AR, MR, and VR.*

The author conceptualizes the difference between AR and MR according to the user interaction. The AR is considered when the user interacts with the virtual objects through the 2D screen, i.e., touch screen interaction on a hand-held mobile device. Whereas MR is the technology that enables the user to interact with the virtual object in the real 3D space, i.e., interacting with virtual objects by using a controller or hand gesture while using a see-through HMD.

It is also worth mentioning that the terms XR and VR/AR (VR and AR) were used interchangeably throughout the thesis, implying a broader spectrum of immersive technologies.

In addition to these four -R abbreviations, there is another popular term in the field: **metaverse**. It is the combination of two words, meta and universe [43]. Meta[1] is a Greek-originated prefix with meaning "beyond" and "after." With the help of induction reasoning, one would say metaverse means beyond or after the universe.

Stephenson [50] used this term for the first time in his science-fiction book Snow Crash. This dystopian novel tells the story of a collapsing postmodern civilization, where people move into a virtual successor of the Internet, the Metaverse [43]. However, this term received

[1]Andronicus organized Aristotle's books; named the books after physics as metaphysics.

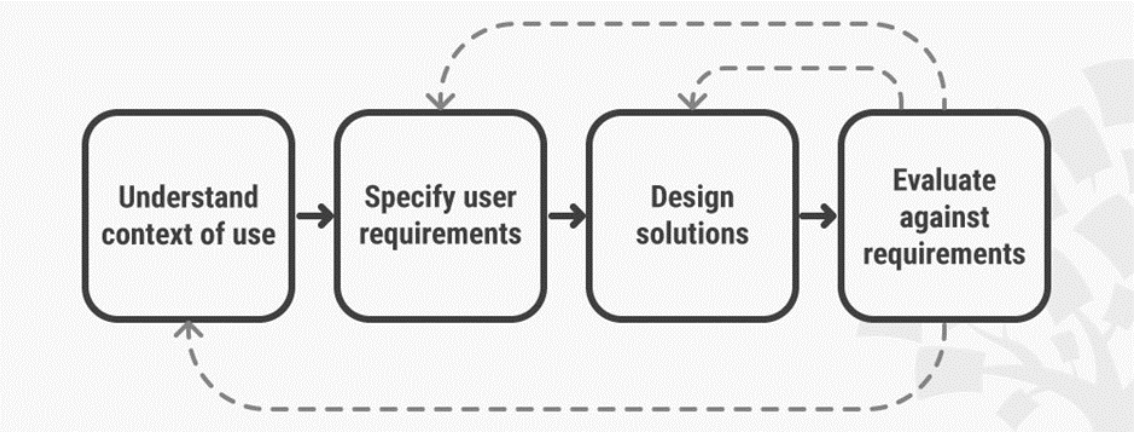


Figure 3.3: 4 phases of iterative user-centered design [23].

the attention of almost everyone in the world when Facebook changed its brand name to Meta. They defined metaverse as an embodied internet where users are in the experience, not just looking at it [31].

Today, there is no definite consensus on the definition of metaverse, and it is continually evolving [27, 43]. World Economic Forum [27] defines it as interconnected physical and digital worlds with a sense of presence. One way or another, it is evident that the metaverse and immersive technologies have started a new era in human interaction.

3.2 User-Centered Design

User-Centered Design (UCD) is built on three pillars: early focus on users and tasks, empirical measurement, and iterative design [47]. This approach assumes that the user knows the best and the designer should translate these needs into a solution [45]. This translation is an iterative process [23]. The first goal of the design team is to understand the context in which the user interacts with the product, system, or solution, followed by need-finding. After identifying the requirements, the actual design phase begins. The designer and other team members assess the design against the user's needs. This assessment may influence the previous three steps. Then, the team applies the necessary changes and re-evaluates the solution. This iterative process (See Figure 3.3) continues until the final design reaches a satisfactory level.

In XR, the experience is immersive, and user interaction is not limited to 2D small screens or mouse and keyboard. This causes issues in XR interaction design and user testing because traditional 2D design and testing principles cannot cover all possibilities of 3D interaction where the user is free to move [26, 5]. Besides, in such a free world, different users may execute similar actions differently. Their sense of perspective and interaction preferences may vary. These issues can be addressed by understanding different user profiles and conducting persona-based evaluation [24]. The user-centered design framework can bring persona-based development processes into the XR industry, address the issues with interaction design and user testing, and subsequently raise the bar higher.

3.3 Methodology

This thesis is exploratory research that aims to understand the challenges in the XR domain and seeks modern solutions from real-world evidence. Therefore, qualitative research methods that would be more suitable for exploratory research were chosen. There are five methods used in this thesis: semi-structured interviews, focus groups, diary studies, user research, and empirical research. The ultimate goal is to approach the research problems from different angles and generate reliable information by triangulating various data. In the final step, thematic analysis is conducted on the collected data. These themes were grouped into the three categories mentioned in Section 2.1 whenever it is suitable: *understanding*, *designing*, and *implementing*.

3.3.1 Semi-Structured Interview

Semi-structured interviews combine the features of structured and unstructured interviews; the interviewer can follow a basic script as a guide and then probe the interviewee to learn more by asking follow-up questions [47]. This script is prepared before the interview according to the research goals. The interviewer can change the order of the questions or decide how much time to spend on each question.

This approach aims to gather insights from participants by addressing both open and closed questions during the interview. More details can be captured with probing questions that would not be possible during a structured interview. It also mitigates possible distractions since there is a basic script to follow. On the other hand, there are some disadvantages. For example, it would be difficult to analyze and compare the answers to open questions due to their subjective nature. In addition, probing questions may lead to an observer bias, steering the participant in a particular direction [16].

To avoid biases, the interviewer should stay neutral throughout the interview and refrain from affirmative comments or leading signals. These signals may be verbal or nonverbal; therefore, the interviewer should consider the language, questions, gestures, and body posture.

Another aspect is data recording. Different data recording techniques can be used with interviews, such as video recording, voice recording, and note-taking. It is crucial to inform the participants about data recording and ask for their consent. Personal data should not be recorded without the participant's consent.

3.3.2 Focus Group

The Focus group is a group interview in which the discussion is led by a facilitator [47]. It creates a moderated open discussion platform. It allows participants to argue, inspire, and debate each other to pursue further insights that may have been missed otherwise. The participants should represent the sample of the target group. During group interviews, it is more appropriate to address shared issues rather than individual topics [47].

Both semi-structured interviews and focus groups are cross-sectional studies. They gather the data in a short period. This can be seen as a snapshot of subjective data during the event. These two methods have many practices in common because of their similarities.

Figure 3.4: Focus Group: Step-by-step guide [15].

For example, Figure 3.4 shows the stages of the focus group from initial preparation to final analysis. All these stages apply to semi-structured interviews as well. The only difference is that the focus group consists of multiple participants.

The focus group questions are also very similar to the semi-structured interview questions. During the focus groups, the facilitator can ask closed or open questions and then use follow-up questions to dive into the details. Previous section 3.3.1 explains that the facilitator should sustain neutral verbal and non-verbal communication throughout the focus group.

An essential requirement for this type of group research, which is different than individual interviews, is that the facilitator should guide the discussion and balance the participation among people, i.e., encourage quiet people to participate and slow down the verbose ones from dominating the discussion [47].

The same data recording techniques used in semi-structured interviews are also available for focus groups. The participants should always give informed consent before beginning the focus group.

3.3.3 User Test

User test is one of the observation techniques to capture the user's behavior while interacting with the application or the product. Preparing a test plan, including a guidance script that explains the study goals, rules, and tasks, is crucial for successful user testing. *"The use of a script ensures that each participant will be treated in the same way, which brings more credibility to the results obtained from the study"* [47].

The aim is to collect qualitative data to gain better insights into novice users' thought processes. However, *"one of the problems with observation is that the observer doesn't know what users are thinking and can only guess from what they see"* [47]. For this reason, participants can be encouraged to follow the think-aloud technique. This technique requires participants to share their feelings, opinions, understanding, and what is going on in their heads while doing the test.

Observations can take place in the field or in a controlled environment. *"In the former case, individuals are observed as they go about their day-to-day tasks in the natural setting. In the latter case, individuals are observed performing specified tasks within a controlled environment such as a usability laboratory"* [47].

"Observation in a controlled environment inevitably takes on a more formal character than observation in the field, and the user may feel more apprehensive" [47]. In order to help the participants feel more comfortable, the host should mention that the participant's skills or personal capabilities are not tested, but the application is tested. The purpose is to assess whether the application can enable users to accomplish specific tasks.

Various data collection techniques can be used during direct user observation in a controlled area. Some examples are video recording, photographs, screen recording, interviews, or note-taking. It is crucial to repeat that no personal data should be recorded without the participant's consent.

3.3.4 Diary Study

Nielsen Norman Group [46] defines diary studies as a qualitative user research method used to collect insights about user behaviors, activities, and experiences over time and in context. This type of method can capture the long-term behavior development of users in a particular context that could not be observed in a single session interview.

Adopting new skills is a long-term process that develops over time as the learner practices various activities like reading, watching, and experimenting. In this regard, a diary study is a suitable longitudinal research technique to capture behavioral changes and the acquisition of new skills over time.

During a diary study, both quantitative and qualitative data can be extracted. Although this type of data collection is powerful, it also has disadvantages. For example, third-person participants may miss logging important details, exaggerate, or understate their feedback [47].

One of the critical aspects of a diary study is its duration. *"Determining how long to run a diary study can be tricky. If the study goes on for too long, participants may lose interest and need incentives to continue"* [47]. The researcher should design the study in a way that overcomes this problem. The sampling technique (e.g., survey, personal check-in, logging), sampling range (e.g., hourly, daily, weekly), and the study length (e.g., day, week, month) should be optimized to maximize the reliability and quality of data.

3.3.5 Empirical Research

Empiricism is a philosophy which advocates that knowledge is acquired from the experience gathered through the senses. *"Empirical research is research that is based on observation and measurement of phenomena, as directly experienced by the researcher. The data thus gathered may be compared against a theory or hypothesis, but the results are still based on real-life experience"* [13]. In other words, empirical research can be seen as an umbrella term for systematic analysis of real-world evidence.

Empirical research can be conducted using various techniques such as observations, experiments, and surveys. It can be deductive or inductive. The inductive approach collects the real-world evidence and then derives a theory from this evidence, whereas the deductive

	Usual raw data	Example qualitative data	Example quantitative data	Initial processing steps
Interviews	Audio recordings. Interviewer notes. Video recordings.	Responses to open-ended questions. Video pictures. Respondent's opinions.	Age, job role, years of experience. Responses to close-ended questions.	Transcription of recordings. Expansion of notes. Entry of answers to close-ended questions into a spreadsheet
Questionnaires	Written responses. Online database.	Responses to open-ended questions. Responses in "further comments" fields. Respondent's opinions.	Age, job role, years of experience. Responses to close-ended questions.	Clean up data. Filter into different data sets. Synchronization between data recordings.
Observation	Observer's notes. Photographs. Audio and video recordings. Data logs. Think-aloud Diaries.	Records of behavior. Description of a task as it is undertaken. Copies of informal procedures.	Demographics of participants. Time spent on a task. The number of people involved in an activity. How many different types of activity are undertaken.	Expansion of notes. Transcription of recordings.

Figure 3.5: Quantitative and qualitative data can be collected from various methods [47].

approach tests the evidence against assumed theory [13]. The researcher must determine the study objectives at the beginning of empirical research. Then, the research methodology can be designed according to these objectives. Empirical research offers flexibility as it is possible to follow various methods to collect qualitative and quantitative data.

3.3.6 Quantitative and Qualitative

"*Quantitative data is in the form of numbers, or data that can easily be translated into numbers... Qualitative data is in the form of words and images, and it includes descriptions, quotes from interviewees, vignettes of activity, and photos*" [47].

It is possible to collect both quantitative and qualitative data from data gathering methods mentioned above. These data can be processed by different techniques to derive meaningful results. The Figure 3.5 summarizes typical data collection methods and the captured data types with initial processing steps.

3.3.7 Thematic Analysis and Categorization

Collected qualitative data should be processed and analyzed to have overall understanding and gain the insights. There can be various qualitative data analysis methods. For example, themes can be identified from the data or the data can be categorized into groups.

Thematic analysis can be seen as a general framework to identify the certain topics within the collected data. "*More formally, a theme is something important about the data in relation to the study goal. A theme represents a pattern of some kind, perhaps a particular topic or feature found in the data set, which is considered to be important and relevant ... to the goals driving the study*" [47]. It is also possible to have the analysis frame (the set of categories used) in advance. In other words, perform the analysis considering the categories and have in depth insights within these categories to answer study goals.

These two techniques (e.g., thematic analysis and categorization) can be combined as well. "*Identifying themes (thematic analysis) takes an inductive approach, while categorizing data takes a deductive approach*" [47]. One can also apply thematic analysis to the collected data in the first step, and then these themes can be categorized under pre-existed groups. Sharp et al. [47] argues that such iterative approach is a common practice to find patterns and represent these patterns systematically.

Chapter 4
Study Design and Results

This chapter presents the study design and the results of the various research techniques. It explains what was observed in a descriptive format. Through this multi-stage approach, the study aims to contribute to a deeper understanding of the barriers to entry and propose novel solutions to these hurdles.

The study design for this thesis entails an exploratory research approach utilizing qualitative research methods. The research journey begins with a comprehensive literature review, which serves as the foundation for identifying research gaps and areas for improvement. Then, necessary data is collected from the selected research methods. The collected data is analyzed to answer the research questions.

The data collection started with two methods: the semi-structured interviews and the focus groups. These were employed to gather insights from experts and experienced practitioners in the field. Real-world experiences, best practices, and recommendations were observed. These two research methods required the longest time to recruit more participants. Then, the diary studies were conducted to delve deeper into authoring tools and visual scripting. Building upon these findings and experiences, the user tests on visual scripting were designed and conducted. Cross-sectional research was also carried out to review a list of

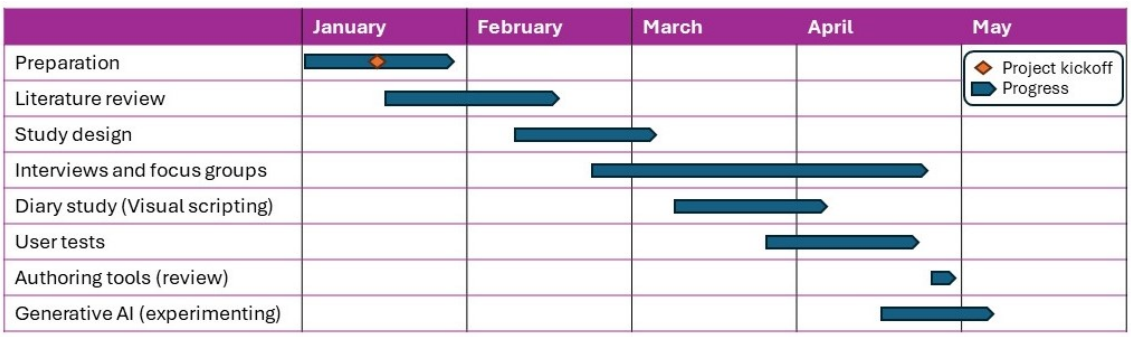


Figure 4.1: Study timeline.

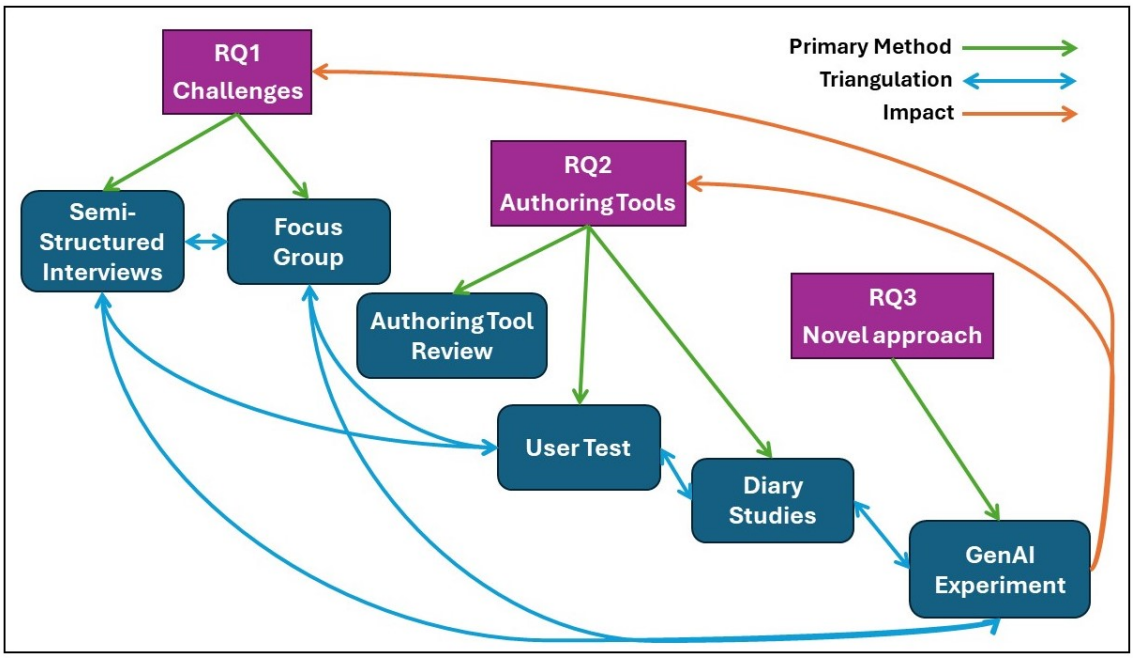


Figure 4.2: Study design overview: Research Questions and Methodologies

authoring tools from the literature. The final method was the empirical research focusing on using GenAI within the context of VR/AR development. Figure 4.1 shows the overall degree project timeline for the each method.

As analyzed in Chapter 2.1, previous studies categorized barriers to entry into three groups considering XR development stages: *understanding*, *designing*, and *implementing*. These categories drew the outline in this section for the consisted representation. The results from each method were grouped under these categories whenever possible.

4.1 Study Overview

This exploratory thesis used five different research techniques to answer the research questions. Figure 4.2 visualizes the relationship between the research questions and the research methods.

RQ1 was addressed primarily by semi-structured interviews and focus groups. During these studies, the author could capture the current hurdles in the field and some best practices from those participants with experience creating XR applications. Moreover, these interview studies could collect insights related to the other research questions because of their flexible and open nature of exchanges with the participants. The participants shared their views and experiences on difficulties, authoring tools, visual scripting, and GenAI. The data from individual and group interviews was later triangulated with the user test and GenAI experiments.

RQ2 was tackled by multiple methods: an empirical review of authoring tools from previous studies captured the current situation about no-code/low-code tools. Diary studies explored the visual scripting features of Unity and Unreal. User tests aimed to understand

the novice user's thought process while using visual scripts. Learning from diary studies laid the groundwork for designing a suitable scenario for the user tests. Subsequently, user tests brought different perspectives that might have been missed during diary studies. It was also possible to observe the real-world difficulties during the user test. These difficulties were also examined during the semi-structured interviews and focus groups. Therefore, the data from these studies could be triangulated.

RQ3 was investigated by another empirical study. The author experimented with GenAI tools ChatGPT and Unity Muse during this study. The aim was to evaluate if GenAI could be an overarching solution to the problems that newcomers face. These problems were mainly related to the difficulty in finding information and the difficulty in writing code. A similar scenario from the diary study and user test could be tested during the GenAI experiment. This gave comparable insights about the advantages and disadvantages of GenAI.

4.2 Semi-Structured Interview

Semi-structured interviews were conducted to understand the difficulties and best practices in VR/AR development. The data collected from these interviews was subjective and qualitative. The participants shared their personal experiences and perceptions of VR/AR development. The data was recorded by voice recording and note-taking. In the end, a thematic analysis was conducted on the collected data.

This method primarily aimed to answer RQ1 (barriers). Thanks to its flexible and open nature, the participants shared valuable insights also to answer RQ2 (authoring tools) and RQ3 (GenAI).

4.2.1 Preparation and Setup

First, an interview plan consisting of the participant profile, the interview protocol, and the interview questions was prepared (See Appendix A). This plan ensured that a certain quality standard level was achieved for each interview. Second, the target group was people with XR development experience. There was no limitation on their experience level, occupation, or background. Last, the interview questions were prepared and written in the plan.

Once the plan was completed, the recruitment of participants started. The candidates were recruited from acquaintances, related events ("Best of Europe: XR, AI, Immersive Tech" by VR and AR Association), LinkedIn, and online searches (developer forums or communities). Advertisements and flyers were distributed in local communities. The recruitment and interviews were done in parallel and lasted over two months.

The interviews were conducted on *Zoom*, an online video conferencing platform. It provided time and location flexibility. However, Zoom did not allow voice recording; it only allowed video recording because of license limitations. Video recording was considered intimidating for the participants and challenging to analyze. Therefore, a separate mobile device was used for voice recording.

The interviews started with a brief including a short introduction and interview protocol. The interview was recorded using a voice recorder upon the participant's consent. These recordings were used for further analysis after the interviews whenever necessary. The author also took notes during the interviews.

Table 4.1: Semi-structured interview participant details.

	Gender	Age	Background	Occupation	Experience
P1	M	23-27	Engineering	Game developer	Learner
P2	M	23-27	Occupational	Technical artist	Experienced
P3	F	18-22	Computer sci.	Student	Learner
P4	M	33-37	Engineering	XR developer	Expert
P5	M	23-27	Computer sci.	XR developer	Experienced
P6	F	28-32	Engineering	XR developer	Limited
P7	M	38+	Eng. Doctorate	Software dev.	Limited
Not in chronological order					

4.2.2 Participants

Seven interviews were conducted; most participants (4/7) had an engineering background. The author used engineering as an umbrella term for technical fields such as electrical, civil, mechanical engineering, or ICT, excluding computer science or software engineering. Computer science and software engineering were categorized under computer science. Two participants had computer science backgrounds. One had a high school diploma but occupational training in game and VR development.

Four participants worked in the VR/AR industry: three as XR developers and one as technical artist. One participant was a game developer, one was a software engineer, and another was a student studying in a VR/AR master program. Their VR/AR experience levels also varied. Experience levels were categorized based on the years of professional experience. Learners did not have any professional experience. Juniors had up to three years of professional experience. Experienced ones had three to nine years, and experts had 10 or more years of professional experience. Please see Table 4.1 for a detailed overview.

4.2.3 Results

The participants shared their own journeys, experiences, and views on VR/AR development. The challenges identified in these interviews were mainly similar to those mentioned in the previous studies. Thematic analysis was conducted on the collected data. The challenges, best practices, and relevant insights are presented under three categories which were identified during the literature review (See Section 2.1): *understanding*, *designing*, and *implementing*.

Understanding

All participants tried some VR/AR applications, and they were impressed with the technology before starting their VR/AR development career or learning journey. They suggested that newcomers should experience VR/AR and try to understand it as users before jumping into VR/AR development. Observing the XR system and application as a user would be beneficial when the person attempts to develop an XR application or experience.

Once a user decides to become an XR practitioner, participants recommended that selecting a complete authoring tool,i.e., a game engine, could be a good start. The general feedback was that Unity would be a better fit for a newcomer. Learning Unity was more

straightforward than learning Unreal; on average, Unity was also more commonly used in the industry. Unreal was usually used by highly sophisticated teams and companies. Practicing 3D game development on Unity could be an efficient way to learn the fundamentals. Then, transitioning to VR/AR would be easier.

The highly fragmented XR industry has already started moving towards consolidation. Frameworks, instruments, and practices are converging. However, despite the unification efforts, the lack of standardization is still a problem. Tools are being updated often, new instruments emerge, and some become obsolete. Keeping up with all those changes in the industry is an impossible task. In this regard, it was mentioned that there is no magical way or shortcut to handle this situation. VR/AR practitioners must put effort into keeping themselves updated. Being selective about the information intake and learning only relevant topics can be a good practice.

Another often observed issue was that users, clients, or people without sufficient understanding of XR systems often had wrong expectations. They would usually overestimate current XR technology's capabilities and expect too much from it. In the professional field, this would create problems between clients and XR developers. At an individual level, a novice practitioner might be disappointed or discouraged by the mismatch between his/her expectations and reality.

Designing

Most of the participants (6/7) had technical education and acquired high mathematics and analytical thinking skills. One participant had occupational training in game development at a young age and developed similar skills without formal education. The interviews revealed that understanding 3D concepts is crucial to succeed in the field. It is not only about 3D space and locations of the objects in the space but also about the observer, the perspective, the scale, and the interplay between the objects and the user. Besides, participants mentioned that designing an immersive 3D experience differed from 2D design, and transferring 2D know-how into the 3D domain was not easy.

Another common comment was that to create an immersive experience, more than 3D geometry is required. Realistic visuals and physical simulations are also desired in an immersive experience. Photo-realistic rendering is no longer a big problem, as specialized tools exist for this purpose. However, the correct physical simulation requires technical and analytical skills to be implemented. Without correct visual and physical representation, motion sickness may occur because of sensory mismatch. Another technology that may improve the immersion is digital twin technology. With this technology, an exact copy of a physical system can be simulated in a virtual environment.

The participants said that once they understood 3D concepts and immersive systems, technicalities in design are not an issue. Various free and open-source materials and tools are available, which could help practitioners design and prototype easily.

Another critical aspect of the design was sound. One participant, P4, was an esteemed expert in the field and he mentioned the importance of auditory design by quoting prominent director George Lucas: *"Sound is 50 percent of the moviegoing experience."* According to him, companies and developers usually neglect sound design as they focus on visual design. However, correct sound design would make the VR experience incredibly immersive. Sound should not be at the bottom of the priority list. He also added that finding a skilled sound

designer for VR/AR projects can be challenging.

Implementing

It was found that all participants used game engines to develop VR/AR applications. Unity was the primary authoring tool of all participants. Three participants mentioned that they also had previous experience with Unreal Engine.

Although coding is an inseparable part of VR/AR development, according to the participants, it is a small detail in the big picture. Therefore, coding should not be seen as a significant hurdle. Most of them (6/7) had technical education, and they did not have any issues with coding. They claimed that if one knows the terminology well then the necessary code could be generated with the help of GenAI. It may not be 100% correct in the first attempt but GenAI could be used for troubleshooting as well.

All participants made the same points about GenAI: (1) GenAI could be an excellent assistant to develop applications and write code. (2) GenAI would never replace a human developer as it was lacking certain comprehension features at the time of the interview. To get the most from AI assistance, the user should know what he/she is doing and be able to review the code generated by the GenAI. Otherwise, unknown code blocks could create more problems like undesired behaviors or unknown operations. They also emphasized that most of the companies in the field does not allow copying the code from ChatGPT and then pasting into the company environment without review.

Visual scripting was also asked during interviews. Although Unity was the primary platform for all participants, none actively used visual scripting. Some participants tried it briefly but did not continue using it. Three participants were not aware of Unity Visual Scripting (UVS). However, it was mentioned that designers usually use visual scripting to create shaders. This tool was Shader Graph, which enables users to create shaders visually and is different from UVS. Contrary to UVS, most of the participants (6/7) knew about Unreal Blueprints.

According to the participants, the main issues with UVS were the performance and not having complete control over the script. They preferred to have full control over their code and optimize its performance. Although Unreal Blueprints could offer better performance, it was stated that performance optimization would not be possible with visual scripts in general. They could be used in rapid prototyping or demonstrations but not in a professional product with complex interaction or game mechanics.

Testing was another challenge mentioned during the interviews. The test did not only mean testing the code and fixing the errors but also testing the whole system and the experience. User testing was costly in various aspects. A comprehensive user test would require know-how, resources, time and money. The teams working on XR projects usually did not have all of these simultaneously. When they wanted to outsource the user test, it would cost more money and there was a risk that the test might not align perfectly with the project requirements in the end. Some participants mentioned that they were using user-centered design principles to tackle these issues.

4.3 Focus Group

The focus groups aimed to deepen the findings from semi-structured interviews by bringing different perspectives from open discussions. During the focus group research, it was also possible to observe the communication between various roles, such as designers and developers. Similar to the semi-structured interviews, subjective qualitative data was collected during the focus groups. The participants shared their personal experiences and responded to each other's comments. The same data recording techniques used for the semi-structured interviews were selected: Voice recording and note-taking. Using similar techniques streamlined efforts for easier data collection, maintenance, and analysis. Thematic analysis was conducted at the end of the study.

The focus groups essentially sought answers to RQ1 (barriers) like the semi-structured interviews. Thanks to its flexible and open nature, valuable insights could be captured to answer RQ2 (authoring tools) and RQ3 (GenAI).

4.3.1 Preparation and Setup

A semi-structured interview plan also guided the focus group study (See Appendix A). The interview questions were slightly modified during the focus groups to accommodate group discussion dynamics, like allowing people to respond to each other, but the essence of the questions remained unchanged.

The same recruitment strategy was followed: acquaintances, events, online search, and advertisements in the local communities. However, the focus group recruitment was noticeably more difficult. The main problem was to find multiple people from the target group available simultaneously. Then, the author changed the recruitment strategy and approached XR companies. Companies were not willing to participate. Although it was not explicitly said, the author had the impression that managers and executives did not accept a team spending one-hour time slot during working hours. A free lunch was offered to the participants to tackle this and encourage participation. By doing so, the focus group did not interfere with the working hours, and more participants could be recruited. Two months were dedicated to the recruitment and organization of the focus groups.

While planning the group interview, the author requested team managers and company executives not to join the session. Since the purpose was to discuss any difficulties encountered, the presence of managers could affect the discussion negatively. For example, the team members might not feel comfortable to discuss their problems in front of their managers.

Two focus groups were conducted: one organized over Zoom and the other face-to-face. The purpose and protocol were explained at the beginning. With the participants' consent, a mobile voice recorder recorded the discussions. The facilitator also took notes during the discussions.

4.3.2 Participants

One focus group was organized with a professional team of eight at their company facilities. Five participants had XR developer roles, two 3D designers, and one designer. The developers had a technical background like computer science and engineering. Two developers and the

Table 4.2: Focus group participant details.

	Gender	Age	Background	Occupation	Experience
P1	M	23-27	Graphics	3D designer	Experienced
P2	M	28-32	Engineering	XR developer	Expert
P3	M	23-27	Engineering	XR developer	Limited
P4	F	38+	Fine arts	Designer	Expert
P5	M	28-32	Engineering	XR developer	Experienced
P6	M	33-37	Computer sci.	XR developer	Expert
P7	M	23-27	Engineering	3D designer	Limited
P8	M	18-22	Computer sci.	XR developer	Limited
P9	M	28-32	Computer sci.	Software dev.	Learner
P10	M	28-32	Computer sci.	Software dev.	Learner

designer were expert professionals with more than 10 years of experience. This group would be referred as a professional group later in this section when necessary. See Table 4.2 for a detailed overview.

The other focus group was conducted with two participants. They had a computer science background and worked as software developers. In addition to their personal interests, they studied in a two-year VR and AR master program and worked on various VR/AR projects during their studies. Later in the text, this group would be referred as an amateur group.

4.3.3 Results

The focus group studies could successfully achieve their purpose by creating an open discussion environment where participants dig deeper into the topic and respond to each other's comments. The discussions started slowly with one-to-one question-answer, but gradually, the participants started commenting actively without waiting for direct questions from the facilitator.

Conducting focus groups with amateurs and professionals revealed some differences. For example, the amateur group had to manage everything (e.g., designing, prototyping, and implementing) independently. On the contrary, the professional group had roles such as designers and developers. Thus, the development dynamics were different for these two groups. For instance, the amateur group had no issues implementing their design. However, the professional team experienced a mismatch between design and implementation from time to time.

After collecting all the data, a thematic analysis was conducted. The findings of thematic analysis can be seen below.

Understanding

The participants claimed that technical education like engineering or computer science would equip individuals with the necessary mathematical and programming skills. Once these skills were present, learning Unity and C# was not a problem. However, finding a job as an XR developer would require more than basics as it is still a relatively small market with unique hurdles. Participants commented that moving from gaming to VR and mobile development

to AR would be easier than finding an entry-level XR job. Both the gaming and mobile industries were considered bigger than XR at the time of the interviews. There would be more job opportunities in these fields compared to XR for junior candidates.

The amateur group emphasized that a mathematical background with 3D knowledge is necessary in VR/AR. For example, basic object manipulation (Translate, Scale, and Rotate) in 3D requires matrix operations. Without this knowledge, creating a VR/AR experience would be almost impossible.

Professional team members could help each other when one does not understand a topic, and they could also create their internal knowledge base to share the know-how and information with each other. Others did not have such support. Fortunately, new technologies like chatbots could help individuals in this matter. The majority of the participants said that they were using ChatGPT as an advanced *Google* to find information. Instead of searching blogs, knowledge-base articles, or lengthy documentation, they used ChatGPT for their inquiries. This helped them to find the correct information in a shorter time with little effort.

Designing

The participants stressed that analytical skills were required to create a well-designed XR experience. Without understanding of mathematics, one could not simulate physical behaviors or create immersive scenes where the environment behaved realistically. For example, the amateur group developed an AR application with a specific feature based on precise calculations. The use of correct mathematical model enabled them to produce an excellent AR experience. The audience who tried the application was impressed by that specific feature.

According to the participants, in XR, it could be possible to design something that has not been designed before. This was exciting for some of them but there were also some drawbacks. It was difficult to troubleshoot such a unique design and fix the issues because there were no other examples to guide the developers and designers. Such situations were described as an adventure with many unknowns. There was no straightforward solution to this. In this case, it was recommended that VR/AR practitioners should research, read, learn, and experiment. Over time, with more experience, they would acquire the necessary skills to cope with the unknowns.

The professional team's challenge during the design phase was the mismatch between design and implementation. Two types of mismatch were common. One was that the desired environment, object, or interaction would be too hefty to implement for developers. The other one was that the result would differ from the initial plan. Better communication, having a common language, and becoming familiar with each other could minimize these problems over time. Additionally, the designer started using AI tools to create storyboards and other materials to communicate her design with other stakeholders more effectively.

Implementing

All participants had heard about Unreal, but their main development platform was Unity. Unity was considered as easy to learn and sufficient for developing products in most VR/AR use cases. Nobody used UVS actively in their projects, but two developers attempted to use it in rapid prototyping. They believed that UVS could be helpful in some specific small-scale projects, which would be a tiny portion of the VR/AR market. Having complete control

over the application was crucial for optimizing the performance and developing complex algorithms. Therefore, traditional coding was still the preferred approach in the professional market. They also mentioned that C# is a high-level language[1] which is easier to learn. According to them, one did not need to be an expert in C# to create applications successfully. A programming mindset, especially being familiar with the OOP paradigm, would be enough to begin with.

As mentioned above, developers used AI tools like smart *Google*. Tools like ChatGPT and GitHub Copilot helped them with the heavy lifting of coding. They said that Copilot was their primary tool for writing code, and ChatGPT was mainly used for searching information. When they did not know how to code, or when they knew what to code but to avoid extended writing, or when they debugged their code, they took advantage of these tools. Moreover, these tools could explain the function of the code and help developers understand it. Moreover, one developer mentioned that he was using AI tools to refactor[2] his code. However, it was mentioned that blindly applying AI-created code was considered risky. The developer should be cautious about the GenAI-generated code. He/she should know what he/she is doing and be able to evaluate whether the answer serves the purpose and is free from unnecessary code blocks that would perform undesired operations without noticing it.

4.4 Diary Studies: Visual Scripting

The objective of this method was to acquire the necessary skills to develop Real-Time 3D (RT3D) scenes without writing code but using visual scripting. As learning new skills was a long-term process, a contextual and longitudinal research method, a diary study, was chosen.

The author decided to conduct a self-diary study to evaluate visual scripting on Unity and Unreal over a month in a home-office setup. The reasons for a self-diary study in a home-office environment are: (1) he average VR/AR development setup would be similar to the home-office environment; (2) to some extent, the author represented the target group, i.e., no prior experience with Unreal Engine and no previous experience with Unity Visual Scripting (UVS). He had a basic understanding of programming paradigms but minimal coding experience; (3) diary study is an expensive technique, and recruiting participants who would commit to this long-term study was difficult; (4) observations from the first person's perspective could ensure high-quality data collection.

The first diary study was on Unreal and lasted approximately 20 days. Since the author had not used it before, it took him longer to learn the platform before working on the visual scripts. The second study was on Unity and lasted approximately ten days. The author was already familiar with Unity at a basic to moderate level, and he could focus on visual scripting early during the study.

These diary studies were conducted to answer RQ2(authoring tool). In addition, insights related to RQ1 (barriers) could be observed as the author progressed through the journey.

[1] In the spectrum, machine language is at the bottom and human language is at the top. Low levels are closer to the machine language. High-levels are closer to human language. Therefore, high-level programming languages are regarded as easy to understand by humans.

[2] Code refactoring: the process of improving readability, structure, and maintainability of a code without changing its functionality

4.4.1 Preparation and Setup

A hands-on approach was decided as a self-learning method. The initial aim was to learn and implement the fundamentals such as object manipulation, 3D interaction and mechanics, and game design with visual scripting. Official documentation and tutorials were the primary source of information. If the official documentation was unavailable or insufficient, the author sought help within the official developer communities. Third-party resources, e.g., YouTube, other forums, or blogs, were used if that was not enough.

The following resources were used in the home-office setup: a gaming laptop with internet access, an additional monitor, game engines (Unity 2021.3.17f1 LTS and Unreal Engine 4.27), Visual Studio 2022, and logging applications such as Google Sheets and Docs, and note-taking materials. The most recent versions of Unity and Unreal were not selected, considering software stability and their popularity. The assumption was that more materials could be found for the previous version than the latest version as it takes time to adopt the newer versions. Selected versions were the latest versions of the previous generation at the time of study.

Before starting the study, it was important to read the license terms. Both game engines had free license options for the individuals and companies below certain funding thresholds. In this research, the author's conditions and usage purposes allowed him to use free licenses. Both engines could be installed easily and did not require specific expertise other than general computer knowledge. The instructions and the guides available were sufficient to complete the installation successfully.

4.4.2 Results

Although visual scripting interfaces were similar in both game engines,i.e., connecting graphical nodes, learning and development experiences differed because Unity and Unreal have different system architectures.

"Unreal Engine is built by developers, for developers" was the very first sentence on the main page of the Unreal website [62]. This could give a hint about their target user group: developers. To widen their potential user group beyond developers and to support different stakeholders in the industry, Unreal offers Blueprint Visual Scripting[3], which is a game-play scripting system using a node-based interface instead of writing code [61].

Unity promoted its popularity (82 of the top 100 games) and VR/AR capabilities on its website [59]. However, the visual scripting was not part of Unity by default. It is offered as an additional package. The user should download and install it manually. Whereas, Unreal Blueprints was one of the default development environments.

It is worth mentioning that the author had a basic understanding of Unity but no prior experience with Unity Visual Scripting (UVS). To mitigate possible biases from his previous knowledge, the author put extra effort at every step to evaluate if his actions could be achieved by a novice user or not. For example, the author knew how to use variables on Unity, but the documentation and tutorials were followed blindly to see whether they could guide the user correctly. Notes were taken if the author needed to do something different than what was explicitly written in the documentation.

[3]Blueprint in short

Figure 4.3: The beginner courses were not available [61].

Understanding

Both game engines offer various project types and templates, such as game, industrial, or cinematic projects. Unity project started right after the template selection, whereas Unreal had one more step to complete. The user was asked to configure render quality, ray tracing, and scripting method (C++ or Blueprints) in this additional step. One could understand the function of these settings from their description without technical knowledge. Therefore, a novice user could start a project without significant hurdles.

As mentioned above, RT3D game projects were selected for this research. The author initially thought the game engines' interfaces were difficult to understand; there were too many options, settings, and components to grasp. Even though the author understood Unity, his knowledge was not useful on Unreal. It was apparent that an inexperienced person could not create anything on these game engines without training. Each game engine would require its own training, as transferring know-how from one to another was not straightforward.

Numerous tutorials and training options were offered by both engines. Unity's materials for novices had more coverage and better structure than Unreal's material. Actually, the beginner tutorials of Unreal Engine 4 were removed completely, although there were still references to them in the official documentation (See Figure 4.3).

Regarding visual scripting, **getting-started** tutorials were available from both engines. UVS tutorials offered different paths that one could select according to his/her knowledge level. For inexperienced users, the fundamentals of Unity visual script were explained with more examples. More information in connection with C# scripting was given as well. Although tutorials were helpful in the basics, there were not many examples to guide the users in reaching the next level. The situation was the same when third-party resources were examined. Overall, the tutorials were helpful in learning the basics of UVS, but there was insufficient material to improve the visual scripting skills further.

The Unreal tutorial was brief and directly delved into the practical use of Blueprints, with minimal explanation. Yet, the project templates had examples of Blueprint visual script. Once the user acquired a basic understanding of Blueprints, existing scripts could be modified or manipulated for experimentation. However, this was not useful when the user wanted to create something new which is not similar to these examples. Despite limited official re-

sources, many materials were available from third-party resources on more advanced topics.

Overall, Unity visual scripting had useful materials suitable for novices to start with but few materials for experienced users. On the other hand, Unreal Blueprint materials targeted more experienced users and beginners.

Designing

The most significant difference between the engines was their general system architecture. This directly impacted the visual scripting design and implementation. Unity had a more intuitive system that could be understood by novices, as the visual scripts could be attached to objects almost freely without considering the whole hierarchy at a basic level. On the contrary, Unreal clearly stated in its documentation that Blueprint follows the Object Oriented Programming (OOP) paradigm [61] and it is a complete class structure rather than a simple script attachment. The author had difficulty comprehending this system despite his fundamental programming skills.

Because of this structural difference, expertise in one game engine could not be directly transferred to the other. To make it easier, both game engines offered guides and tutorials on how to transition from one to another. The resources for transitioning from Unreal to Unity were outdated. On the contrary, the Unreal Engine guide for Unity users was updated and more comprehensive.

Apart from the system structure, when the visual scripting was concerned; the major challenge was that predefined functions had to be used. There were plenty of functions for almost all basic operations, however, it required familiarity with the terminology. Functions that perform the same operations would have different names. For example, the **Hit** function on Unreal was named as **Collision** on Unity. Therefore, to use it efficiently, tool-specific knowledge was required. There were a few instances in which the author had to spend hours or days to find the correct nodes to complete the script. Official documentation and third-party resources were not helpful because the author could not find the correct information due to missing terminology knowledge or using the wrong terms.

Another aspect was that predefined functions were restrictive and made some simple tasks complicated. For example, basic mathematical operations were hefty to design. Many nodes were required to make a game object with three lives that would be reduced by one after being hit by another object (see Figure 4.4). Spending a significant amount of effort on simple operations or not even being able to find the correct function was the biggest challenge of visual scripting.

Despite the difficulties, it is indisputable that one could create visual scripts for game mechanics, object manipulation, or state machines without writing code. Online resources were sufficient for simple algorithms as long as the user was familiar with the terminology.

Implementing

Programming skills were not necessary to implement a visual script, but algorithmic thinking was needed to create the flow using inputs/outputs. The implementation was straightforward once the right nodes were found and the algorithm was designed.

However, when the visual script had multiple branches, various data types (e.g., Integer, Vector 3, object, camera), and methods (e.g., events, GET/SET methods, states), it became

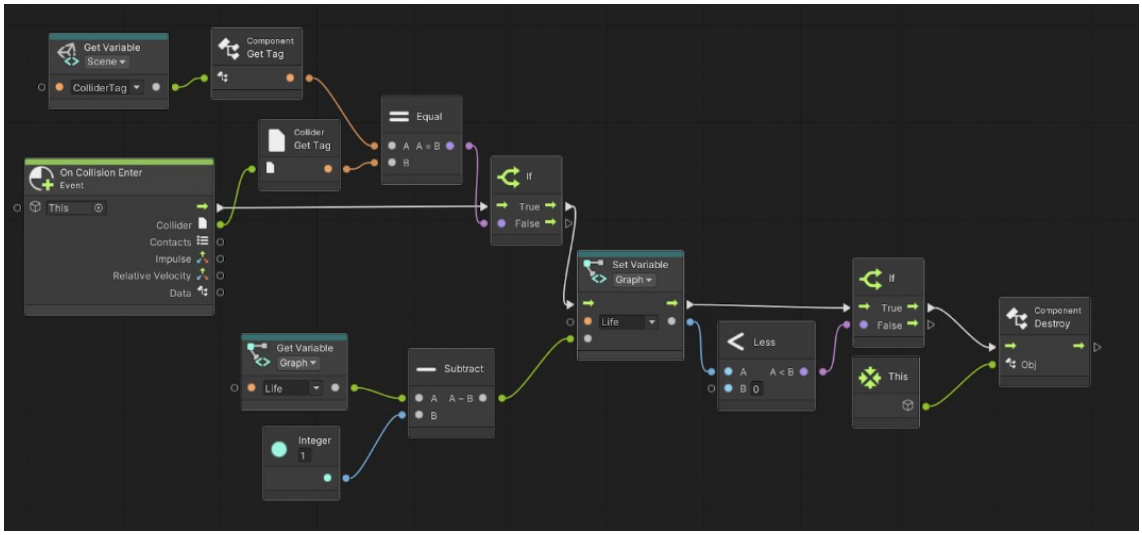


Figure 4.4: Unity Visual Script: Destroying an object when its life is zero.

burdensome to manage. These issues could be grouped into two categories: readability and troubleshooting.

The visual representation of the script made it easy to understand. But, when the code becomes too big or complex, it becomes unreadable as the reader cannot follow the node connections. This phenomenon is known as **Spaghetti Code** 4.5. The author observed that the risk of spaghetti code is inevitable with the visual scripting as the script grows (See Figure 4.5).

Visual script debugging and troubleshooting were not different than traditional code debugging and troubleshooting. Eventually, graphical nodes were nothing but the visual representation of pre-written code blocks. The error messages were related to the actual C++ or C# code when there was an error in the script. Interpreting these error messages demanded programming knowledge. For example, wrong data types could be passed into the nodes, and class inheritance might prevent certain operations, or runtime errors might occur. Without programming skills, it would be very difficult to fix these problems as they were related to the code. When these error messages were searched in the online platforms, C++ or C# solutions could be found, but not visual scripting solutions. Therefore, a novice user without programming knowledge could not use those solutions in his/her visual script.

The author also spent time on the transition guide from Unity to Unreal. This guide helped him better understand the Unreal Engine and Blueprint structure. After this, he felt more comfortable with Blueprints and could develop more complex experiences or game mechanics. However, the transition guide from Unreal to Unity was outdated.

While the author was experimenting with visual scripting, he observed that system performance metrics like frame per second and responsiveness were degraded more severely on Unity than Unreal. Performance optimization was outside the scope of this research, so it was not investigated further, but it could be observed clearly.

To summarize, despite its restrictive nature, visual scripting could enable users to write programs to perform actions or calculations. Designing and implementing a visual script for basic programs was easy; however, as the script grew, it became more difficult to maintain

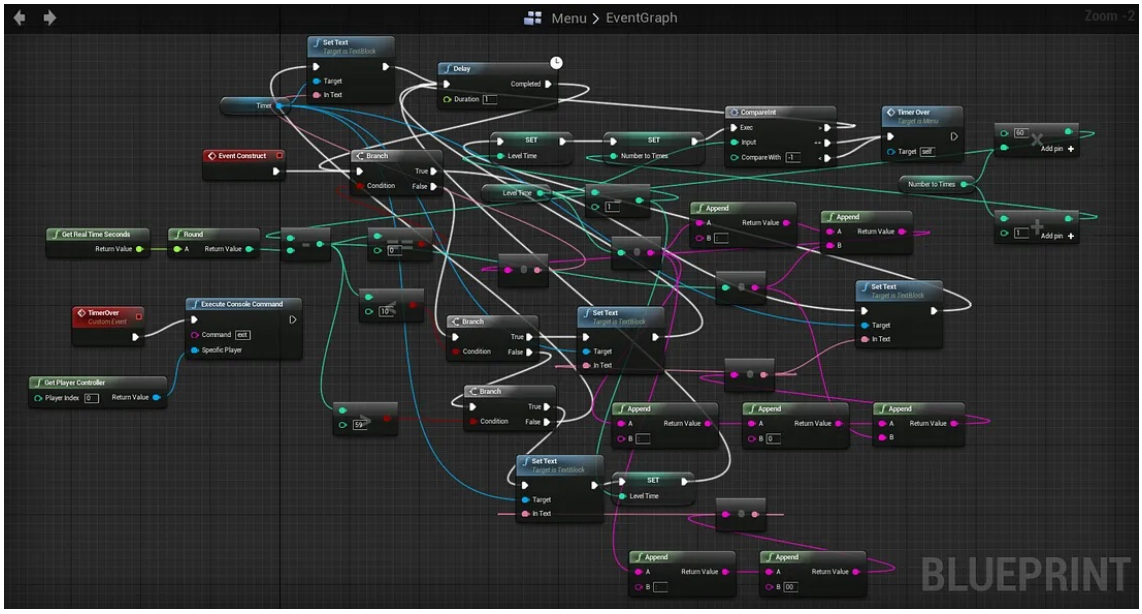


Figure 4.5: Relatively readable Blueprint, before becoming a spaghetti code [7].

and troubleshoot.

4.5 User Test

Visual scripting could be one of the ways to break the vicious circle of a novice XR developer (See Figure 2.1). It can empower non-technical newcomers to develop VR/AR applications without writing code on a complete authoring tool. This study intended to assess whether inexperienced people could mentally grasp the use of visual scripting and develop 3D interaction.

The user test implies direct user observation in a controlled area in this thesis. The purpose of this observation was not to identify usability problems but to understand the comprehension of visual scripting and 3D concepts. Subjective qualitative data was collected from users to achieve this. This data reflected users' thought processes, personal experiences, and perceptions. Voice recording and note-taking were used for data logging. Thematic analysis was conducted on the collected data. These techniques were not intrusive and allowed the author to streamline efforts, as the same techniques were used with other methods, such as semi-structured interviews and focus groups.

User tests attempted to answer the RQ2 (authoring tools) in the first place. Since the participants were novices, observing their challenges provided data related to the RQ1 (barriers).

4.5.1 Preparation and Setup

A test plan was prepared to standardize the testing process. This plan defined the test protocol, participant profile, data collection, and test procedure. The target group was defined

as people who did not have game or VR/AR development experience, regardless of their background, occupation, gender, or age.

As Section 3.3.3 mentions, user test means direct observation in a controlled area in this research. The controlled environment was a communal study room in the residential building where the author lived. The room had a semi-formal ambient, which was suitable for testing. It was not an intimidating lab environment, or very casual place with distractions. The following hardware and software resources were used for testing: A gaming laptop with internet access, an extra monitor, keyboard, mouse, and game engines (Unity 2021.3.17f1 LTS and Unreal Engine 4.27).

The participants were recruited from personal acquaintances, community advertisements, and flyer distribution at local game events. The testing was in-person, so the participants had to come to the test environment. To attract people, a cinema ticket voucher with popcorn and a drink was offered as a token of gratitude.

During Diary Studies 4.4, it was observed that the game engine editors were overwhelmingly difficult to understand without prior knowledge. Since the target group had no experience with game or XR development, onboarding was needed to eliminate frictions related to the game engines. A detailed guide was prepared to familiarize the users with the tools (See Appendix B the guide for Unity test and Appendix C the guide for Unreal test). This guide explained the game engine editor interface, navigation in the scene, and test procedure.

The main goal of this test was to capture the users' thought processes rather than measure their task completion success. This was the essential difference compared to a usability test. For example, in a usability test, the success measure would be task completion, whereas, in this study, tasks were used as tools to explore novice users' perceptions and thought processes.

The author's learning from the diary studies were instrumental to design the test scenario. A Real-Time 3D (RT3D) first-person shooter game was selected for the test scenario. In this scenario, the player could shoot a projectile from his/her gun and hit other objects in the scene (See Figure 4.6). The author utilized his learning from the diary studies (See Section 4.4) to create similar test environments on both Unity and Unreal. This RT3D environment allowed the player to move freely and interact with other objects.

There were some missing features in the test scenario and participants were expected to complete these missing features by using visual scripts. Given that the participants did not have a prior game or VR/AR development experience, the test scenario was developed in a progressive and didactic way, allowing participants to use the information provided in the previous task to complete the next task. For example, the test started with Task 0. It explicitly showed how to rotate a game object using visual scripting. Then, Task 1 introduced a new function, *destroy*. Yet, the connections between nodes were not explicitly shown. The user was expected to connect the nodes correctly and complete the visual script, which destroys the game object when a projectile hits. Task 2 required using the *destroy* function from Task 1, but it was not instructed explicitly.... Task overview can be seen in Table 4.4.

The test started with welcoming the participant and briefly explaining the process and the protocol. After the participants gave their informed consent, the voice recording started. The facilitator explained the test procedure and shared the detailed instructions as both soft and hard copies. He also supported the participants with the game editor usage, navigation in the scene, or other technical aspects that were not directly related to the test scope. Moreover, the participants were allowed to use any online resources when they had a problem or question related to the visual scripting. By doing so, two things could be achieved: (1) a more

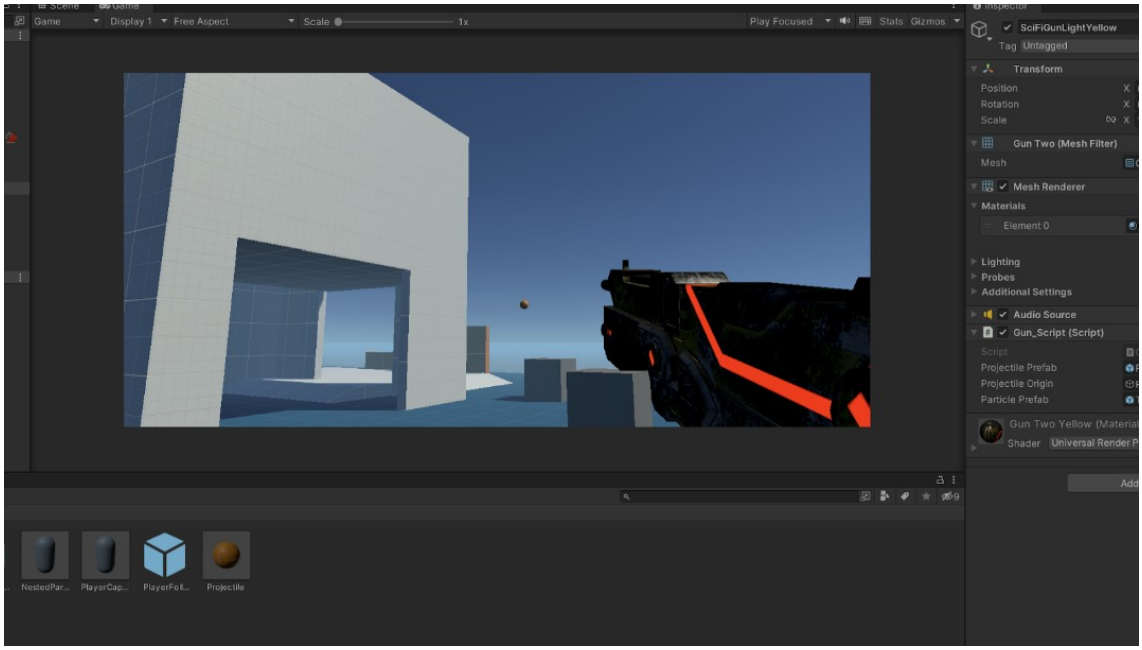


Figure 4.6: RT3D first-person shooter test scene on Unity.

realistic development environment and (2) capturing insights related to RQ1 (barriers).

More realistic development environment could be achieved because, in real life, users search for information online all the time. This was also verified during the focus groups and the semi-structured interviews. Participants said that online resources are their primary source of information. Besides, it could reveal insights related to RQ1 (barriers) because difficulty using online resources was one of the critical barriers [5].

The facilitator sat beside the users and directly observed their actions on the screen. The users were encouraged to think aloud. When the user had an issue with the visual script, the facilitator allowed the user to try harder to find a solution. If the user requested guidance after failed attempts, the facilitator asked about the user's opinion on how to solve the problem. After a brief conversation, the facilitator tried to guide the user implicitly instead of fixing their problems. This approach could be seen as similar to probing questions used in interviews to delve into the topic and capture more details.

At the end of the test, a final interview session occurred. The facilitator asked participants about their experience, understanding, and feedback. The whole test procedure, from welcoming to the final interview, was scheduled to take approximately one hour.

4.5.2 Participants

Four participants could be recruited for the user test research. Two participants had engineering backgrounds. One participant studied computer science, and another one studied law. Their occupations varied; one worked as a software developer, two worked in engineering but not software development, and one was a lawyer. See Table 4.3 for an overview.

During diary studies and interviews, it was evident that learning Unreal was more difficult than learning Unity. To minimize the effect of different learning curves, Unreal was tested with the participant who had expertise in C++ software development. This was because

Table 4.3: User test participant details.

	Gender	Age	Background	Occupation	Tool
P1	M	28-32	Computer Sci.	Engineer	Unity
P2	F	33-37	Law	Lawyer	Unity
P3	M	33-37	Engineering, PhD	Researcher	Unity
P4	M	33-37	Engineering	Software Dev.	Unreal

Table 4.4: User test tasks

Task 0	Rotate a game object
Task 1	Destroy the game object when a projectile hits
Task 2	Create an explosion effect when the projectile hits an object
Task 3	Add life to the game object, destroy it when the life is zero

Unreal utilizes C++ OOP architecture, and it was assumed that learning could be easier for this participant. The other three participants ran their tests on Unity.

4.5.3 Results

All participants started their test by navigating the scene after initial onboarding. They wanted to know more about the scene and how it was working. On average, participants spent 25% of their time understanding the game engine editor, its interface, and the scene. After becoming familiar with the system, they started working on tasks. Task overview can be seen in Table 4.4.

Task 0 was a warm-up task, and the instructions provided the solution. All participants completed the task successfully. However, half of the participants asked for further clarification about the node connection and script flow.

Task 1 required the use of functions destroy and hit/collide. Function names were given in the instructions. All participants could find the correct functions; however, non-engineer participant, P2, had difficulty in connecting the nodes correctly. This participant said that she did not understand how the inputs and outputs of a function work. Other participants also requested more clarification about how game objects interact with each other and how the visual script impacts the objects. *Event Hit* function on Unreal was intuitive for P4. Conversely, *On Collision Enter* function of Unity was confusing. Participants wanted to know what it does and the difference compared to other collision functions like *On Collision Stay*.

Task 2 expected using a particle system for the explosion effect. Instructions implicitly mentioned the names of the functions like the following instance: *"Then you need to add another node which can **instantiate** an **original** copy of a prefab (like projectile) at certain **position** and **rotation**."* Half of the participants understood the cue, others could not find the correct function without the help of the facilitator. At that point, the facilitator also explained how the explosion effect could be created and what the particle system was. However, it was observed that the participants did not grasp the logic of particle systems quickly. All participants could include the particle system in the visual script with guidance. However, they all had difficulty setting the particle's correct **position**. The users were not familiar with the world position and relative position concepts. Engineers could understand it after explanation, whereas the

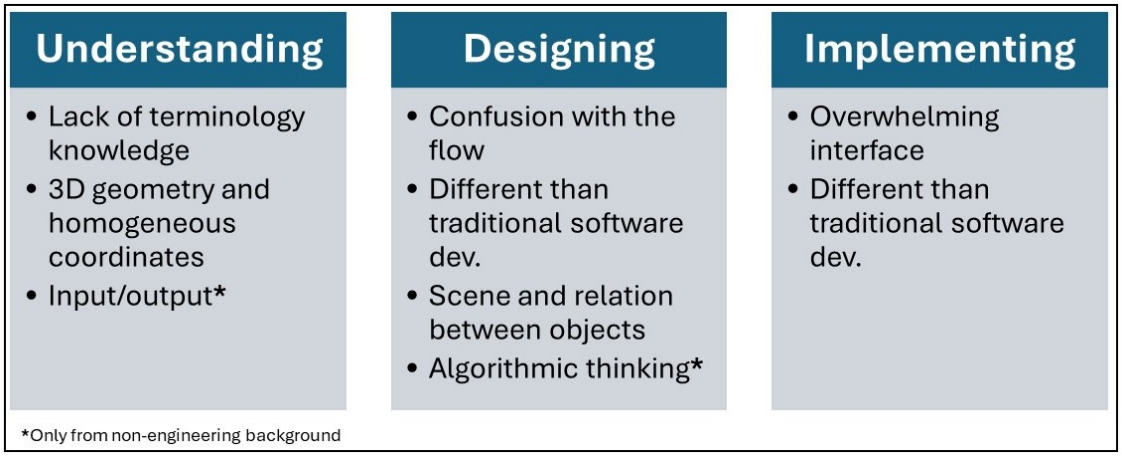


Figure 4.7: User test results: Major difficulties in visual scripting.

non-engineer participant, P2, could not.

The task was completed once the position parameter was fixed on Unreal, as the system could handle the rotation. However, the rotation parameter needed to be handled by the participants on Unity. The facilitator asked participants what they were thinking about the rotation. The engineers could comment on rotation in a generic way. But these comments were not related to the correct rotation concept (quaternion) which was required to instantiate a new object on Unity. It became even more confusing when they tried to pass a Vector3 value to the rotation input. Because the script did not compile successfully with Vector3 input, it gave an error. The error occurred because of wrong data type. **Instantiate** function takes **quaternion** type as a rotation input, not **Vector3**. At this point, it was almost one hour, and the Unity test was finished for P1 and P2 to have the final interview. P3 decided to stay longer to finish the test; it was not one hour yet for P4. The facilitator gave a cue about *get position* function on Unity to P3. Then, he could find the correct functions to pass the rotation input.

Task 3 was attempted by half of the participants, but no one could complete it without guidance. The main challenges were the math operation and logical operations. Math operation was subtracting one from the life value after each hit. Besides, there were two logical operations: one was to control the subtraction so that it would be performed only after hit, and the other one was to compare the life variable against zero to destroy the object (See Figure 4.4). Even after giving the correct nodes, it was difficult for the users to complete the whole script. Having multiple branches and then connecting these branches through the "**if**" condition was not intuitive. However, once the task was completed with the guidance of the facilitator, the participants who had engineering backgrounds said that the flow made sense and they could understand the algorithm.

General observations: The Figure 4.7 summarizes the overall findings from the user tests. Only two participants used online resources to find solutions, and they tried to search on *Google*. However, they could not find answers after several attempts. The other two participants who did not search stated that they did not know what to search, so they thought searching would not help them. P1 and P4 had considerable software development experience. They said they could have completed the tasks quickly by writing code rather than using visual scripting. Visual scripting did not allow them to be flexible and write their own

functions. They also mentioned that the visual scripting and game development were different from traditional software development, and they were unfamiliar with the terminology and practices.

Participants with engineering backgrounds tried to solve the problems by copying previous tasks and systematic testing. For example, when they had some errors, they changed some connections between the nodes and tested again. They performed such iterations multiple times. After a brief explanation, they could also comprehend the basics of 3D, such as relative position and world position.

The non-engineer participant could not understand the quaternion even after it was explained. She struggled with the input/output relation and the flow while connecting nodes. When the facilitator asked her to explain the solution in her mind, she could explain a possibly working solution narratively, but she could not translate it into an algorithm or pseudocode. She said that she could perform programming on Microsoft Excel by writing formulas, but this type of coding was impossible for her.

4.6 Empirical Research: Authoring Tools

To become a successful VR/AR practitioner, it is vital to be proficient in development instruments such as game engines and content/asset creation platforms. There are tons of tools available for various purposes. However, having such a broad range of options may be an issue. It was found that newcomers did not know where to start their XR journey [5]. In addition to this, there was another friction point: Coding requirement. It was claimed that no-code/low-code authoring tools would enable non-programmers to develop their VR/AR application [36, 5, 8].

Nebeling and Speicher [36] categorized various authoring tools according to their use cases, and Ashtari et al. [5] recommended some tools for inexperienced XR practitioners. This study aimed to extend the coverage of those studies and capture the current status. This extension was both horizontal and vertical. In the horizontal axis, more authoring tools were included, and in the vertical axis, the assessment criteria expanded.

In short, it was a cross-sectional study, and its goal was to take a snapshot of numerous tools with broader criteria to address RQ2 (authoring tools). This study would guide novices or researchers by providing a bird's-eye view of the available tools.

4.6.1 Preparation and Setup

The author created the list of tools (the complete list can be seen in Appendix D) by reviewing numerous publications during the literature review. In the end, 43 VR/AR development and content creation instruments were on the list. All of these were evaluated by their application type, availability, novelty, coding requirement, and interoperability (see Appendix D for details). This review was done by using online resources.

The author first examined the publication, searched for the tool, and attempted to test it briefly. If the tool could not be found, the author tried to find the researcher or developer who created it and traced it back to their public repositories to find the tool. During the investigation, the author recorded his observations instantly.

Figure 4.8: Hirokazu Kato developed an early example of marker-based AR [25].

4.6.2 Results

The list included various tools from advanced game engines to research applications to SDKs[4]. The list items were divided into three main categories: commercial (24/43), research (16/43), and from research to commercial (3/43). The research type described the items that were developed with scientific research intentions. Commercial type implied that the items were developed without scientific research intentions. Commercial did not necessarily mean paid or licensed product either. It could also be open source. The last category described the products that initially developed with research goals but later moved from the academic domain to business. These products eventually became commercial products.

DART (Designer's Augmented Reality Toolkit) [28] was an early example of a no-code rapid AR prototyping tool. During the literature review and this empirical research in which numerous tools were reviewed, it was apparent that DART inspired many researchers in the field and gave direction to the no-code studies. Unfortunately, none of these no-code research applications on the list could survive until today. They either became obsolete, were unavailable, or did not turn into usable tools.

Three successful research outputs later became usable products. Hirokazu Kato [25] developed ARToolKit in 1999. It was an early example of a marker-based AR (see Figure 4.8) and one of the first AR SDKs for mobile devices. However, despite its success during the early 2000s; today, it was maintained by three volunteers under *artoolkitX* name [22]. It was a usable SDK but it was not popular anymore. Another successful tool developed in 1999 was Teddy. Teddy allowed users to create 3D objects from 2D sketches quickly and easily [21]. Today, it can be purchased from the Unity Asset Store and used in VR/AR or 3D projects for rapid 3D asset creation. The last example that started from research and became a commercial product was HoloBuilder [48]. It began as an AR-based interactive user manual project. Now, it is an enterprise construction progress management platform. Therefore, it is no

[4]Software Development Kit

longer a VR/AR authoring tool but an excellent example of successful no-code research in the VR/AR domain.

From the commercial category, Adobe Aero was one of the no-code AR authoring tools. It enables users to create AR experiences without writing code [1]. Yet, when it was tested, it was observed that it had limited capabilities. Microsoft Maquette was an excellent example of an immersive VR authoring tool. Users could design VR scenes within VR and then import these scenes to Unity. Unfortunately, the beta tag was not removed from Microsoft Maquette, and Microsoft stopped actively developing it in 2022 [33]. It was stated that it would be removed from the store, and Microsoft had no plan to open any source code of it [33].

Although there were a few no-code authoring tools available, they had limited capabilities compared to the complete authoring tools like Unity or Unreal. In other words, the author could not find any no-code tool which had potential to replace Unity or Unreal. On the other hand, various no-code or low-code tools could be integrated to Unity or Unreal as an additional package. These packages could help novices to create XR artifacts.

To wrap up, DART has inspired many researchers to take the next step in developing no-code authoring tools. However, those attempts could not be unified to create a complete authoring tool or create momentum to support prototyping in an upward direction (from lo-fi to hi-fi). In contrast, complete authoring tools like Unity and Unreal offer flexibility, interoperability, and a unified end-to-end solution to VR/AR practitioners.

4.7 Empirical Research: Generative AI

GenAI-assisted programming can be a solution to overcome the barriers to entry. This study evaluated two GenAI tools, ChatGPT and Unity Muse, by experimenting with their assistance capabilities. As mentioned in Section 2.3, researchers [3] could develop 2D games using ChatGPT. This study extended similar research into RT3D development. ChatGPT and Unity Muse were not only tested for code generation but also for searching information, design, implementation, and troubleshooting.

Unity Muse was in the early access phase at the time of the study. It had five features: chat, texture, sprite, animate, and behavior. The chat feature could be seen as a general assistant that supports the user in various topics through conversational dialogue. In this research, the main focus was to use Unity Muse's chat feature for code generation, debugging, and general guidance like ChatGPT. In addition to this, the author could briefly test the texture features of Unity Muse to observe its capabilities.

This experimental study mainly collected qualitative data. This method sought answers to RQ3 (GenAI) in the first place. Since GenAI tools could cover a wide range of topics, the challenges from RQ1 (barriers) and RQ2(authoring tools) could be addressed by asking relevant questions.

4.7.1 Preparation and Setup

The same home-office setup was used for this study: a gaming laptop with internet access, an additional monitor, game engines (Unity 2021.3.17f1 LTS and Unreal Engine 4.27), and Visual Studio 2022. ChatGPT version 3.5 was tested in this study. However, the version information of Unity Muse was not available at the time of study.

Both tools were web-based which means it could be accessed via web browser separately from the Unity editor. Therefore, the user needed to change the windows during chat interaction. ChatGPT 3.5 was a free tool at the time of the study, while Unity Muse required a paid subscription. It was tested during a 15-day free trial. Another constraint was that during the free trial, only a limited number of interactions were allowed. The user had 1500 points, and each chat interaction takes 25 points. Therefore, the number of interactions was limited by this budget.

Various RT3D scenarios were tested. One of the scenarios was the first-person shooting game developed during the diary studies and tested with the user tests. By doing so, it was aimed to capture insights comparable to those of other studies.

These experimental tests were conducted in the following way. The author explained his task/goal to the GenAI tools and asked for its guidance. He followed the instructions or used the code provided by GenAI. When there was an error, the error message was shared with the tools for possible fixes and additional guidance. During the study, the author tried to follow general prompting recommendations such as precise descriptions and dividing long and complex tasks into smaller steps. He did not attempt to master prompting as it was out of the scope of this study.

4.7.2 Results

Although ChatGPT could guide the user on the basics, it could not provide precise instructions on Unreal visual scripting. Finding the right graphical node and using it correctly was not easy with the given instructions. Then, he attempted to develop C++ scripts. This could be a perfect fit for this research as the author was not proficient in C++ programming. However, despite successful code generation, ChatGPT could not provide precise instructions for an inexperienced user like the author. The author's progress was slow with the C++ scripting. As this method was not a diary study to observe the long-term learning, the author decided to stop the experiment on Unreal due to the time constraints to complete the research.

The rest of the study focused on creating functional RT3D scenarios on Unity with different interactions and operations. The results were presented in a generic way, such that if the name of the tool (i.e., ChatGPT or Unity Muse) was not explicitly mentioned, it was observed on both tools. If an observation was from one tool, then its name was mentioned.

Understanding

Understanding the GenAI tools was rather easy as it did not require special skills other than writing text prompts. However, using the correct terminology and precise description was crucial for getting the right answer. Otherwise, the answers could be misleading or wrong. If the same question was paraphrased or asked differently, it was observed that the tools could give different answers. In other words, the way the questions were asked influenced the answers.

The tools could guide the user in various topics, such as finding pieces of code, accessing relevant information, or general Unity usage. When the author was unsure about the terminology, he described what he was looking for, and the GenAI could provide helpful answers. It could be seen as an advanced Google search. Unity Muse was noticeably better in providing

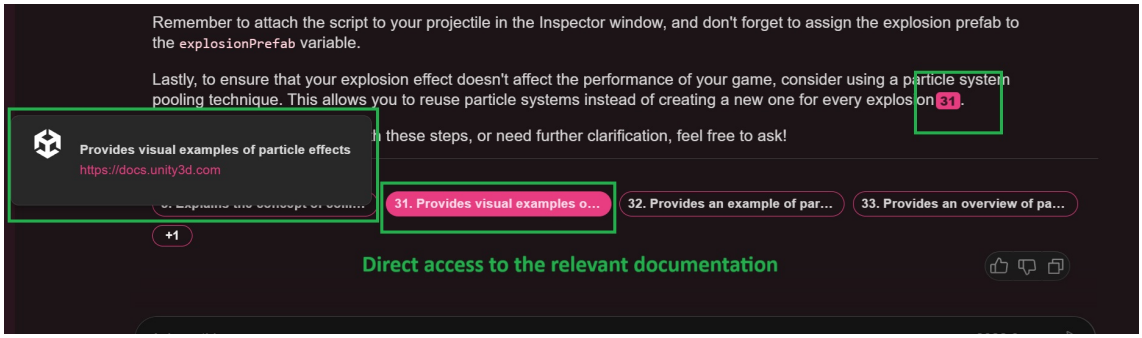


Figure 4.9: For every information, Unity Muse gave a reference to the official documentation

accurate information about Unity. It also shared references to the official documentation for every information it provided (see Figure 4.9).

On some occasions, the author already knew the answer, and the tools' reply was incorrect. In such cases, the author shared his concerns with the tools that the answers were incorrect. Subsequently, the answers were updated or changed. Sometimes, it could provide the correct answer after the interference from the author, but it was also observed that the correct answer could not be reached even after a few attempts. In another words, the GenAI tools were not 100% accurate (See Figure 4.10).

The answers were mostly comprehensive and long, covering many details related to the question. Follow-up questions could be asked to narrow down the focus or shift the attention to another topic. During the conversations, both tools could remember the previous interactions and provide answers considering the previous messages.

Designing

Both tools could guide the user on the game and algorithm designs. They could explain how to manipulate the game objects or change the scene. However, to utilize these instructions, the user should have had an understanding of Unity and its interface. Performing a game design without familiarity with the game engine would be challenging despite GenAI's assistance. The result would also depend on the user's interpretation. Because the instructions from the GenAI tools could be generic and open for interpretation. For example, instruction like "*add a 3D object*" might result in different scenes as it did not specify the object, its location, and its rotation.

Unity Muse and ChatGPT designed the same interaction in two different ways. For example, the author wanted to develop an interaction to destroy a game object when it was hit by a projectile. The same terminology was used with both tools, but the answers were different. ChatGPT suggested using *OnTriggerEnter* method, whereas Unity Muse suggested *OnCollisionEnter* for the same task. These two different methods influenced the overall object behavior, resulting in different game mechanics. In other words, tools made decisions that directly impacted the game design. The author found this problematic as GenAI tools could affect the design negatively by introducing undesired interactions or features. Therefore, the user needed to understand the answers and translate them into the project correctly to avoid such problems.

Further into developing more complex interactions, both tools could be used for design-

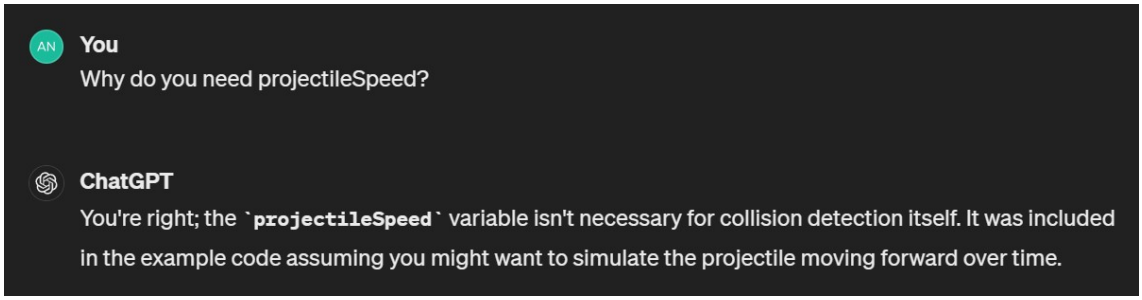


Figure 4.10: There were unnecessary parts in the code

ing specific actions or algorithms. GenAI could guide the user on more generic topics like how to make an explosion effect, how to simulate physical phenomena, or how to design an algorithm to accomplish more complex tasks. In addition to code generation, Unity Muse could generate textures and animations on versions 2022.3 LTS or later. The author experimented with texture generation. He could create textures by prompting his desired visuals. Unity Muse could create texture artifacts successfully. However, this was not investigated further as the focus was on code generation, game design, and algorithm design.

Implementing

None of the tools could generate a visual script that could be imported into the Unity project. They could guide the user on how to add visual scripts, but these instructions were not precise. It did not provide function names, required data types, or how to connect these nodes. Therefore, GenAI could not assist the author with the visual script. On the other hand, both tools could generate C# successfully. C# codes were easy to read and had correct indentation and inline comments that supplemented additional information. No syntax error was observed with the GenAI-generated codes during the study.

Coding-related answers included a simple description of the overall function and the methods used in the script. It was also possible to ask for more details on the code itself or instructions on how to use it. For example, when the user wanted to know more about a piece of code, the tool could elaborate on it. It was also observed that ChatGPT provided better-structured codes compared to Unity Muse.

A minor observation was that ChatGPT could provide answers in a couple of seconds, but Unity Muse was significantly slower. Depending on the complexity, sometimes, it took up to ten seconds to get the answer from Unity Muse. Simple code blocks from both GenAI usually worked right away. However, longer codes did not function as expected or had errors. In this case, error messages were also sent to the GenAI tools. They could guide the author to debug the code.

One of the biggest problems in implementation was that those tools were unaware of the project. Their knowledge depended solely on the prompt written by the user, resulting in occasionally unusable code generation. For example, GenAI-generated code might conflict with another project component, override another script, or be overridden by another script. It was apparent that the user needed to evaluate the code before using it.

Another aspect of not-project-aware answers was that proposed GenAI solutions might have assumptions without the user's input or include additional code blocks. For example, when the author asked ChatGPT's guidance on the hit and destroy function, there were

unnecessary code pieces in the answer. These code pieces did not have any functional value. When the author asked about the reason why unnecessary things were included in the code, GenAI said that it was part of the example and it was thought that the author might be interested in it.

Both tools added some unnecessary code blocks to their answers from time to time. These code pieces could have impacted the application performance directly or might not have had a noticeable effect. When the latter situation occurred, there was a high chance that the user had no control over the script and its actions. Once it was out of the user's control, there was a risk of harmful actions such as data breaches or cybersecurity issues. The author did not face such an issue, but he could notice the risk of blindly copying and pasting the use of GenAI-generated code. The author thought he needed to review the code carefully during GenAI-assisted development.

Overall, simple tasks like mathematical calculations or well-defined step-by-step actions could be implemented successfully with GenAI assistance. When errors were found in these simple scripts, GenAI-assisted debugging could fix them. However, complex code generation had more errors and the risk of undesired code blocks. Therefore, the author had to review the codes before using them.

Chapter 5
Discussion

This thesis is an exploratory research. Through meticulous examination, valuable insights emerged from empirical evidence. These insights and analysis results are presented in a descriptive format. The overall objective is to guide newcomers with advice based on real-world practices.

This chapter endeavors to interpret the results in a broader context by integrating insights from various research methods. It discusses three main research categories that were identified during the literature review: *Barriers to entry*, *Authoring tools*, and *Generative AI tools*. For a systematic and consistent representation, the discussion under each category is grouped into three themes: *Understanding*, *designing*, and *implementing*. This is followed by recommendations for beginners and answers to the research questions. Finally, research limitations and suggestions for future work are shared.

5.1 Barriers to Entry

The semi-structured interviews and the focus groups could capture the snapshot of today's XR industry. Moreover, the author could observe the difficulties from a first-person perspective during diary studies. This section discusses challenges, best practices, and required skills from a broader perspective.

5.1.1 Understanding

Previously, it was identified that understanding the XR landscape was one of the initial challenges for novice XR practitioners. The findings indicated that being an XR user before becoming a developer could help novices conceptualize the field. In other words, if a person uses and experiences XR, he/she can understand the concepts easier later when he/she starts his/her XR development.

It was also evident that XR development requires specific theoretical knowledge and skills such as math knowledge, analytical skills, and algorithmic thinking. Research findings showed that formal education in STEM[1] would be sufficient to build the necessary fundamentals.

Previous studies identified the lack of a common language in XR as one of the problems. Despite some improvements in the field, the problem is still present. One participant described this situation as follows: *"There is no bible in XR."* The results showed that, to tackle this challenge, it is advised to **pick a complete authoring tool** to start with (e.g., Unity or Unreal) and **then build the knowledge around that tool**. This would narrow the scope significantly and help the novices to find the necessary information easily.

Alternatively, one may develop the required skills and gain knowledge from a more mature field. Then, this person can transition from that field to XR. The transition from the gaming industry to VR and MR (head-mounted displays) and mobile development to AR (hand-held devices) is considered as smooth.

5.1.2 Designing

Due to the open 3D environment, designing the interaction and user experience (UX) in XR is difficult. Users can move freely and interact differently. There is no standardized way to tackle this problem; however, the persona-based approach, which could be achieved by User-Centered Design (UCD), can be a solution. The author argues that familiarity with UCD would put the XR practitioners one step ahead in the market.

Further studies indicated that understanding the 3D concepts is vital to designing an immersive environment and experience in VR/AR. The author categorizes these concepts into four and advises novices to become familiar with them.

1. **3D geometry/math**: Understanding how objects are manipulated, positioned, and interact with each other. For example, translating, rotating, or occluding objects in 3D.

2. **3D physics**: Animations, actions and manipulations in 3D geometry should follow correct physical phenomena. For example, if walking is not simulated correctly, it may cause motion sickness.

3. **3D visual effects**: Interaction with the light in 3D is different than it is in 2D. For example, unrealistic visual effects would easily break the illusion and reduce immersion.

4. **3D user interaction**: User interaction in 3D is more complicated than it is in 2D. For example, (x,y) coordinates on a 2D screen can easily be determined. Meanwhile, in 3D, sophisticated algorithms like ray-casting are required.

5.1.3 Implementing

According to the previous studies, the coding requirement was one of the significant hurdles. Coding is essential to implement an XR project. However, further results in this research showed that the coding part is a small portion of the XR development. Understanding the

[1] Science, Technology, Engineering, and Mathematics

programming paradigm seems to be more beneficial than just writing code as it equips the novices with the necessary skills to comprehend the overall system. The author believes that novices should not consider the lack of coding skills as a significant obstacle. Algorithmic thinking and the basic understanding of programming paradigms are sufficient to start the XR journey. Novices can improve their coding skills with various instruments while producing XR experiences.

5.2 Authoring Tools

The most extensive section of this thesis concerns the authoring tools. The primary data collection methods were the diary studies, user tests, and the reviews of tools from previous research. The semi-structured interviews and the focus groups also supported the investigation.

5.2.1 Understanding

DART[2] [28] has been recognized as a highly successful no-code authoring tool, inspiring numerous researchers to develop similar tools aimed at lowering the barriers to entry. The author acknowledges the utility of these tools for specific tasks but questions their effectiveness in genuinely reducing the barriers for the novice developers. He argues that to avoid *the vicious circle of a novice XR developer* (see Figure 2.1), one should start with a complete authoring tool instead of a no-code platforms. If this person has limited coding skills, then the visual scripting feature of the complete authoring tool can enable him/her to create XR applications without writing code.

Regarding development platforms, Unity and Unreal are the most popular authoring tools. This study found that Unity is considered the *de facto* standard platform with a shallow learning curve. It has comprehensive tutorials and guides for novices. Moreover, the transition from Unity to Unreal is well documented and could ne easier, but not the other way around. Thus, the author advocates that Unity is a more suitable authoring tool for the novices.

5.2.2 Designing

Asset creation and prototyping are essential details of the XR development cycle. The author believes that although no-code authoring tools are unsuitable for generic XR development, they can be helpful for asset creation. These tools can help novices to create assets rapidly and use these assets in prototypes and storyboards.

Another aspect to consider is the level design and the project structure. Unreal strictly follows the OOP paradigm not only for its scripting but also for its level design. Novices without expertise in programming would have difficulty in creating desired scenes. However, Unity's level design is a parent-child hierarchy that follows the tree structure, which is easier for the newcomers with limited programming knowledge to grasp. The author recommends Unity for novices, considering its more straightforward design structure.

[2]The Designer's Augmented Reality Toolkit

5.2.3 Implementing

It was mentioned above that Unity is considered to have a shallow learning curve. One of the components that leads to this is the scripting language: C#. The results indicated that learning Unity's scripting language C#, is more straightforward than learning Unreal's scripting language C++. The level design architecture and the scripting language influence the visual scripting features of these tools as well. The author found that although Unreal Blueprints offers a more robust system, using Unity Visual Scripting (UVS) was more straightforward, making UVS more suitable for novices.

The author considers visual scripting as an excellent **introductory instrument** for novices with limited coding skills. He emphasizes that the visual scripting can enable novices to generate outputs early in their journey, become familiar with the authoring tool, and improve their programming skills to transition to actual scripting.

The author believes that generating concrete results as early as possible would enable novices to produce prototypes quicker, allowing them to practice more iterations. During these iterations, novices also become more familiar with the authoring tool. This can help them learn the correct terminology and find the relevant information easily. Thus, it is advocated that completing such a positive feedback loop (learn—create—iterate) can effectively improve the learning experience and prevent the vicious circle of a novice XR developer.

The visual scripting has disadvantages as well. It is not performance optimized and not as flexible as code scripting; complex visual scripts may turn into spaghetti code, and some simple tasks like mathematical operations are hefty to implement. The author acknowledges these limitations yet still supports **the visual scripting as a valuable introductory instrument for novices to develop the fundamental skills**. Normal scripts and the visual scripts can coexist and empower novices to take advantage of both. For example, the author suggests writing mathematical operations with coding and then converting it into a visual node and using it in visual scripting to prevent spaghetti code. Because, basic programming skills would be sufficient to write most of the mathematical operations in C#. The combination of these two options helps novices to improve their coding skills and encourages them to learn and practice.

The final stage of the project implementation is the deployment. The user needs to deploy his/her solution to a hardware platform such as an HMD or mobile device. During the research, it was observed that Unity has more versatile cross-platform XR deployment capabilities. Its documentation, plug-ins, and plug-in management options make the cross-platform development and deployment more straightforward, making Unity more suitable for novices.

5.3 Generative AI Tools

Today, Generative AI technology supports people in different fields. This study investigated how GenAI could support novices in XR. The primary data collection method was empirical research in which the author experimented with ChatGPT and Unity Muse. The semi-structured interviews and the focus groups also provided additional information.

5.3.1 Understanding

Generative AI has generated hype and has been treated as a cure for every problem. However, the author observed that the GenAI is not magic. **It can amplify the user's capabilities to become more efficient and productive, but it cannot go beyond the user's potential.** In this regard, it can be a perfect instrument to support notices to grow gradually.

One significant limitation is that these intelligent tools are not perfect (yet). Therefore, the novice users are advised to adapt their GenAI usage according to their knowledge level to assess the GenAI answers so that false information can be filtered. Considering these limitations, one can still benefit from GenAI to find information, delve into details, clarify ambiguous points, or see examples for guidance. These benefits can speed up the learning process and make it more efficient considering the fragmented XR field where *there is no bible.* The author advocates that **GenAI can be an effective learning tool for novices to improve their skills and knowledge**

Another aspect to consider is the cost associated with these tools. Developer participants mostly used premium tools, such as GitHub Copilot, for code generation. In addition, the author tested Unity Muse (paid) and ChatGPT 3.5 (free). The author found that the additional benefits of premium tools would not make a significant difference for novices because of their limited XR knowledge and coding skills. He thinks that free tools would be sufficient for novice XR creators as there are abundant of them for various purposes.

5.3.2 Designing

The above limitations are also present in the design phase. The user has to review GenAI's answers before using them. With that in mind, the author recommends the novice XR practitioners to use GenAI in two design topics: game design and visual asset creation.

It was found that the GenAI can inspire the user in game design directly or indirectly. The direct inspiration can be achieved by asking direct design questions. And the indirect inspiration occurs as a by-product of another question. In both cases, the GenAI can enable the novices to leverage their potential by providing valuable examples or additional features.

The author does not recommend using chat-based general-purpose AI tools for complex/heavy 3D asset creation. These general-purpose tools are not specialized in 3D asset creation, and the artifacts would not be suitable for using in XR development. For example, the ChatGPT-created asset may not be compatible with Unity. However, findings showed that such GenAI tools, e.g., Unity Muse, can support novices with simple visual creations like textures.

5.3.3 Implementing

One of the biggest limitations of the GenAI instruments is that they are not part of the VR/AR project that is being developed. As the tools are not integrated into the project, they cannot give project-aware answers, which may cause errors or unexpected behaviors. Therefore, following the GenAI blindly may not be a good practice. The practitioner should be able to understand the answers and use them according to the project's needs.

When it comes to code generation, GenAI-generated code must be reviewed more rigorously than other GenAI-generated artifacts before use. Because there may be undesired

code snippets, that are by-products of GenAI-code-generation, in the script. These snippets may perform specific actions without the novice user noticing them. These code pieces may be harmless, but they can also pose cybersecurity risks. Therefore, novices should be very cautious when using the code from GenAI.

Cybersecurity aspect should be handled with care in XR because as mentioned, XR systems can collect unique personal data requiring the highest level of protection (See 1.4 Ethical and Legal Concerns). For novices, this may not be a common situation. They may never attempt to develop something risky, but the author takes a conservative approach in this topic. He encourages novices to perform responsible development practices from the beginning of their XR journey to build ethical habits. Therefore, **novices are advised to use GenAI generated code at a level that he/she can assess the risk associated with it**.

Having mentioned these risks and limitations, the author believes using the GenAI as a learning tool would be the safest option. In the previous section, the combination of visual script and normal script was recommended for novices to improve their programming skills. The GenAI tools can help novices generate such codes, which would be hefty in visual scripting but simple to review in actual code.

Converting the visual scripts to regular scripts was also recommended for novices to improve their programming skills. The GenAI tools can guide the novices in this exercise by explaining the code blocks and their structure. As the novices have already developed the application with the visual script before, they would be familiar with the solution and able to assess the code and its behavior.

To wrap up, the author takes a conservative approach and advocates using the GenAI as a learning tool to find information and generate basic codes that would not pose cybersecurity risks or are simple enough for novices to review.

Figure 5.1: Different stages in the journey of novice XR practitioner

5.4 Recommendations for Beginners

The overarching objective of this thesis is to guide the newcomers at the beginning of their XR journey to overcome the barriers to entry. This journey has different stages which are visualized and summarized in the Figure 5.1.

To progress in this journey, one needs to prepare himself/herself in advance. Experiencing immersive technologies and trying to understand them would be a good start. Without understanding what XR is, it would be difficult to create one.

Analytical skills are essential in XR. Understanding mathematics and 3D concepts is sufficient initially; one does not need to be an expert in these topics. Basic algorithmic thinking and design thinking always bring extra benefits at the beginning.

It is an excellent strategy to start with a complete authoring tool and develop knowledge around it. This narrows the focus and makes finding tutorials and project examples easier. Understanding the programming paradigm would allow novices to utilize the learning resources better.

The user-centered design principles can guide beginners through the design cycles. Besides, if the user is not experienced with coding, the visual scripting is a good alternative for starting the first project.

The novice users should aim for incremental growth. Seeing outputs as early as possible would keep beginners going. There are various resources to help with asset creation. These tools can be utilized. Then, GenAI can support and inspire the beginners by bringing the necessary information. And testing should not be forgotten; make sure that the project is working as expected.

In the final stage, it is time to repeat everything; add more things to the existing projects, convert the visual scripts into code, and improve programming skills. When the project is ready, deploy it to HMDs or hand-held devices. It's time to have fun.

Don't forget to share it with family and friends!

5.5 Research Questions and Answers

- **RQ1**: What are the initial challenges and highlights when entering the XR field? How can a fresh entrant overcome these challenges?

- The main challenge newcomers face is that there is no well-defined career or development path in XR to guide them. The highly fragmented sector also makes it difficult to access the resources. Fortunately, there are tools suitable for the novices. These tools can enable them to produce simple XR artifacts. Witnessing concrete outcomes early can keep them motivated to move forward.

  Basic mathematical skills, the understanding of 3D concepts, and algorithmic thinking are prerequisites for overcoming entry challenges. Based on these know-how, the novices can start working with complete authoring tools and expand their knowledge.

  - **RQ1_1**: What are the fundamental skills required? How effectively can people from different backgrounds transfer their skills to XR?
  - Four fundamental skills are required to enter XR successfully: General mathematical skills, understanding 3D concepts, algorithmic thinking, and design thinking. Depending on their background, people can transfer their skills from other domains into XR domain without a major issue. Transitioning from game development and mobile development to XR is smooth.

- **RQ2**: What is the status quo of the authoring tools? How does the no-code approach reflect on popular XR authoring tools?

- There are numerous authoring tools available for different use cases. Unity and Unreal Engine are the most popular complete authoring tools in XR. Unfortunately, the previous research on no-code tools did not influence the complete authoring tools. Most of the no-code authoring tools that was developed under research became obsolete. There are a few examples of research-originated tools becoming commercial products for specific purposes but not complete authoring tools.

  The complete authoring tools like Unity and Unreal offer visual scripting, which enables users to create game-play elements and interaction mechanics without writing any code.

  - **RQ2_1**: How effectively can visual scripting lower the barriers for novice users?
  - The visual scripting has the potential to lower the barriers to entry for novices as it enables the users to create scripts without writing code and encourages them to improve their algorithmic thinking and programming skills. It is a great tool to start with. Users can learn basics and become familiar with the complete authoring tools. Then, the user can gradually shift to regular coding.

- **RQ3**: How can Generative AI tools support novice XR practitioners and lower the barriers to entry?

- The GenAI is a promising novel technology that can support novice and experienced XR practitioners. Especially for the novices, tackling initial challenges with GenAI can effectively lower the barriers. It can help to find information, explain concepts, and guide them on design and implementation. These features can assist novices in developing and improving the necessary skills to progress in XR.

5.6 Limitations

Some limitations have occurred at different stages of this research due to time, resources, and availability constraints. This section explains these limitations.

The major limitation of this study was the number of participants in user research (interviews, focus groups, and user tests). Advertisements were published. Incentives (free movie tickets and lunches) were offered, but the participation could not be improved. Demographics were also limited to European residents with technical backgrounds. Although some participants were originally from different regions, they all resided in Europe at the time of study. Moreover, 80% the participants (17/21) had engineering or computer science backgrounds. The author attempted to mitigate the impact of these limitations by triangulating the data from various methodologies.

Despite all efforts and gifts, only four participants could be recruited for the user test. Although this test was not a usability test, the author thinks the essence is similar. In this regard, Nielsen [38] argues that increasing the number of user tests would reach a point of diminishing return and claims that five user tests would be sufficient to find most of the design problems. In light of Nielsen's suggestion, the author believed that significant friction points in visual scripting could be identified despite the limited number of participants.

In addition, a newer version of Unity Muse was published at the end of Generative AI experiments. The newer version could be installed as an in-editor package for 2022.3 LTS or later versions. This was a noteworthy update as it was claimed that the in-editor version could understand the actual scene and tailor its answers according to the active scene [18]. However, the author could not test it due to trial license limitations and time constraints to complete the research. Therefore, the in-editor version was not tested for this thesis.

Diary studies were conducted by the author only. Inevitably, such self-study results in certain limitations, such as lack of diversification or personal bias. The author put his best efforts into collecting objective data; however, it is impossible to evaluate the impact of invisible bias, if there is any. To tackle this, the insights were supported with data from other studies.

5.7 Recommendations for Future Work

People with different skills may experience the barriers differently. Identifying the challenges according to demographics and backgrounds and attempting to lower the barriers to entry for those particular groups may be a practical approach.

The VR/AR sector and GenAI technologies are evolving rapidly. Repeating similar studies over a specific period, e.g., once every five years, may provide valuable insights and keep academia and industry updated with the current trends. Project-aware GenAI tools might

radically change the XR field.

Standardization would be the most effective way to consolidate the XR market. Consolidation would naturally fix most of the issues and lower the barriers to entry. Researchers, non-profit organizations, and companies should unify their efforts to standardize the XR field.

Chapter 6
Conclusion

The adoption of Extended Reality (XR) was accelerated, and the demand has increased in the recent years. To fulfill this increasing demand and industry expansion, new people should join the XR field. However, the barriers to entry remain high, making it difficult for newcomers.

Previous studies identified key difficulties in the XR field and proposed no-code authoring tools as a way to overcome these difficulties. Unfortunately, these tools had limited capabilities and prevented the users from transitioning to complete authoring tools. The author described this issue as *the vicious circle of a novice XR developer* (Figure 2.1) where the novice XR developer is unable to create complex XR experiences because of the limitations of no-code authoring tools and also unable to use the complete authoring tools because of lack of skills and know-how.

This thesis aimed to lower the barriers to entry by finding empirical insights and the best practices from experienced XR practitioners, attempting to tackle the coding requirements with visual script programming, and exploring Generative AI (GenAI) assisted development.

The results indicated that mathematical skills, the understanding of 3D concepts, algorithmic thinking and design thinking are the required fundamentals in the field. Having these skills at a basic level is sufficient to start. To prevent *the vicious circle of a novice XR developer*, it is recommended to use a complete authoring tool with visual scripting at the beginning. Visual scripting can enable users to create XR experiences without writing code. As the novice user becomes more familiar with the tool and programming, he/she can improve his/her skills gradually. Moreover, Generative AI can help novices to learn more efficiently and improve their skills effectively by guiding them. The novices can utilize GenAI to fix issues, improve existing projects, or create assets.

Finally, a set of recommendations was identified for beginners (See Figure 5.1). The author believes that these recommendations can guide novices to overcome the barriers to entry to XR.

References

[1] Adobe. Augmented reality. it's everything you imagined. https://www.adobe.com/products/aero.html.

[2] Fabio Alexandrini, Adriano Gomes de Freitas, and Bruno Inacio. Comparative evaluation unity and unreal, using nielsen's 10 heuristics as an evaluation parameter. IEOM Society International, 12 2022.

[3] Asad Anjum, Yuting Li, Noelle Law, Megan Charity, and Julian Togelius. The ink splotch effect: A case study on chatgpt as a co-creative game designer. 2024.

[4] Apple. Apple vision pro available in the u.s. on february 2. https://www.apple.com/newsroom/2024/01/apple-vision-pro-available-in-the-us-on-february-2/, 2 2024.

[5] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K. Chilana. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. pages 1–13. ACM, 4 2020.

[6] Tim Bajarin. Nvidia's ceo on the democratization of coding. https://www.forbes.com/sites/timbajarin/2024/03/20/nvidias-ceo-on-the-democratization-of-coding.

[7] Bit Bandit. The quest to avoid spaghetti code. https://medium.com/@BitBandit/the-quest-to-avoid-spaghetti-code-c8913e1a5527.

[8] Oloff C. Biermann, Daniel Ajisafe, and Dongwook Yoon. Interaction design for vr applications: Understanding needs for university curricula. pages 1–7. ACM, 4 2022.

[9] Pearly Chen, Mark Griswold, Hao Li, Sandra Lopez, Nahal Norouzi, Gregory Welch, and Yu Jingyi. Immersive media technologies: The acceleration of augmented and virtual reality in the wake of covid-19 [white paper]. https://www3.weforum.org/docs/WEF_Immersive_Media_Technologies_2022.pdf, 2022.

[10] Josh Urban Davis, Fraser Anderson, Merten Stroetzel, Tovi Grossman, and George Fitzmaurice. Designing co-creative ai for virtual environments. pages 1–11. ACM, 6 2021.

[11] Enno de Boer and Federico Torti. Global lighthouse network: Adopting ai at speed and scale [white paper]. https://www3.weforum.org/docs/WEF_Global_Lighthouse_Network_Adopting_AI_at_Speed_and_Scale_2023.pdf, 2023.

[12] Cyan DeVeaux and Jeremy Bailenson. Learning about vr in vr. *XRDS: Crossroads, The ACM Magazine for Students*, 29:14–19, 9 2022.

[13] Emarald Publishing. How to... conduct empirical research. https://www.emeraldgrouppublishing.com/how-to/research-methods/conduct-empirical-research.

[14] Facebook. Facebook to acquire oculus. https://about.fb.com/news/2014/03/facebook-to-acquire-oculus/.

[15] Tegan George. What is a focus group | step-by-step guide & examples. https://www.scribbr.com/methodology/focus-group/.

[16] Tegan George. Semi-structured interview | definition, guide & examples. https://www.scribbr.com/methodology/semi-structured-interview/, 1 2022.

[17] Google. Google trends. https://trends.google.com/trends/.

[18] Stacey Haffner. All unity muse capabilities are now available in the editor, plus 3 new updates. https://blog.unity.com/engine-platform/unity-muse-ai-capabilities-in-editor-plus-new-updates.

[19] Kyle Hailey. Personal blog post. https://www.linkedin.com/posts/kylelf_bard-google-ai-activity-7029284181952131072-980r, 2023.

[20] Ken Huang, Yang Wang, Feng Zhu, Xi Chen, and Chunxiao Xing, editors. *Beyond AI*. Springer Nature Switzerland, 2023.

[21] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. pages 409–416. ACM Press, 1999.

[22] Ethar Inc. artoolkitx open-source, multi platform augmented reality. https://www.artoolkitx.org/.

[23] Interaction Design Foundation. User centered design (ucd). https://www.interaction-design.org/literature/topics/user-centered-design.

[24] Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. Is virtual reality product development different? pages 1–11. ACM, 2 2019.

[25] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. pages 85–94. IEEE Comput. Soc, 1999.

[26] Veronika Krauß, Alexander Boden, Leif Oppermann, and René Reiners. Current practices, challenges, and design implications for collaborative ar/vr application development. pages 1–15. ACM, 5 2021.

[27] Cathy Li and Kathryn White. Metaverse privacy and safety [white paper]. https://www.weforum.org/publications/privacy-and-safety-in-the-metaverse/, 7 2023.

[28] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. Dart: a toolkit for rapid design exploration of augmented reality experiences. *ACM Transactions on Graphics*, 24:932–932, 7 2005.

[29] McGill and Mark. Extended reality (xr) and the erosion of anonymity and privacy - white paper. *The IEEE Global Initiative on Ethics of Extended Reality (XR) Report*, 2021.

[30] Sahin Mercan and Pinar Onay Durdu. Evaluating the usability of unity game engine from developers' perspective. pages 1–5. IEEE, 9 2017.

[31] Meta. Founder's letter, 2021. https://about.fb.com/news/2021/10/founders-letter/.

[32] Meta. Introducing our open mixed reality ecosystem. https://about.fb.com/news/2024/04/introducing-our-open-mixed-reality-ecosystem/, 4 2024.

[33] Microsoft. Microsoft maquette beta overview. https://learn.microsoft.com/en-us/windows/mixed-reality/design/maquette.

[34] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, pages 282–292, 12 1995.

[35] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. pages 1–12. ACM, 4 2018.

[36] Michael Nebeling and Maximilian Speicher. The trouble with augmented reality/virtual reality authoring tools. pages 333–337. IEEE, 10 2018.

[37] Andrew Ng. Ai isn't the problem - it's the solution. https://www.ted.com/talks/andrew_ng_ai_isn_t_the_problem_it_s_the_solution?language=en.

[38] Jakob Nielsen. How many test users in a usability study? https://www.nngroup.com/articles/how-many-test-users/.

[39] OpenAI. Dall·e 3. https://openai.com/dall-e-3.

[40] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt.

[41] OpenAI. Sora: first impressions. https://openai.com/blog/sora-first-impressions.

[42] Jeremiah Ratican, James Hutson, and Andrew Wright. A proposed meta-reality immersive development pipeline: Generative ai models and extended reality (xr) content for the metaverse. *Journal of Intelligent Learning Systems and Applications*, 15:24–35, 2023.

[43] Georg David Ritterbusch and Malte Rolf Teichmann. Defining the metaverse: A systematic literature review. *IEEE Access*, 11:12368–12377, 2023.

[44] Peter Rubin. The inside story of oculus rift and how virtual reality became reality. https://www.wired.com/2014/05/oculus-rift-4/, 5 2014.

[45] Dan Saffer. *Designing for Interaction: Creating Innovative Applications and Devices*. Voices that matter. New Riders, 2010.

[46] Kim Salazar. Diary studies: Understanding long-term user behavior and experiences. https://www.nngroup.com/articles/diary-studies/.

[47] Helen Sharp, Jennifer Preece, and Yvonne Rogers. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Inc., 5 edition, 5 2019.

[48] Maximilian Speicher, Kristina Tenhaft, Simon Heinen, and Harry Handorf. Enabling industry 4.0 with holobuilder. volume 246, 2015.

[49] Jonathan Stephens. Getting started with nvidia instant nerfs. https://developer.nvidia.com/blog/getting-started-with-nvidia-instant-nerfs/.

[50] Neal Stephenson. *Snow Crash*. Del Rey Books, 1992.

[51] Ivan E. Sutherland. A head-mounted three dimensional display. *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, page 757, 1968.

[52] the European Parliament. Artificial intelligence act: Meps adopt landmark law. https://www.europarl.europa.eu/news/en/press-room/20240308IPR19015/artificial-intelligence-act-meps-adopt-landmark-law.

[53] the European Parliament. Eu ai act: first regulation on artificial intelligence. https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence.

[54] the European Parliament. General data protection regulation (gdpr). https://gdpr.eu/what-is-gdpr/.

[55] Laia Tremosa. Beyond ar vs. vr: What is the difference between ar vs. mr vs. vr vs. xr? https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr.

[56] Jonah Trenker. Ar & vr - worldwide, market insights by statista. https://www.statista.com/outlook/amo/ar-vr/worldwide, 2023.

[57] Unit for Educational Services. Four things to keep in mind before using chatgpt. https://www.campusonline.lu.se/en/studying-techniques/four-things-keep-mind-using-chatgpt.

[58] United Nation. The 17 goals. https://sdgs.un.org/goals.

[59] Unity. Unity real-time development platform. https://unity.com/.

[60] Unity. Unity visual scripting. https://unity.com/features/unity-visual-scripting.

[61] Unreal Engine. Introduction to blueprints. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/.

[62] Unreal Engine. We make the engine. you make it unreal. https://www.unrealengine.com/en-US.

[63] Marc Whitten. Introducing unity muse and unity sentis, ai-powered creativity. https://blog.unity.com/engine-platform/introducing-unity-muse-and-unity-sentis-ai.

Appendices

Appendix A
Interview Plan

Interview Plan for Thesis

Participant profile:

- **Age**
- **Gender**
- **Education & Background**
- **Occupation**
- **Total professional life**
- **Experience in AR/VR (or related fields)**

Interview Protocol:
Explain the research intent and the anonymity of the participants.
Voice will be recorded for transcription. This transcription will be used for analysis.
Only the author and the supervisor will have access to participant details and recordings
Participants can cancel and leave the interview at any time.
Do you give your consent to participate in this interview?

Expected time 30-40mins.
Tools: Voice recording and note taking.

Interview Questions:
Semi-structured focused interview. Aim is to understand

- the challenges at the beginning of VR/AR career
- The ways and strategies to overcome these challenges
- Skills needed in the sector
- Use of authoring/development tools
- Visual scripting and its applications in reality
- Recommendations/best practices

Focus Group:
Focus group interviews will also follow the same format and question content as individual interviews. For example, if the focus group consists of students and fresh learners, then their respective questions will be asked to initiate the discussion.

Questions for Leaders/Experts

- Can you introduce yourself? (Broadly: background and experience)
- How did you first meet AR/VR?
- Can you tell us the challenges and difficulties you had at the beginning of your AR/VR journey?
    - How did you overcome these challenges?
- Let's flip the coin, what about positive things, highlights? What kept you motivated to continue at the beginning of your AR/VR journey?
    - Follow up question related to main motivators.
- Throughout this journey, how could you transfer your skills from different domains like XXXXX (something from the first answer that they introduce themselves) into AR/VR?
- Talking about skills, what kind of skills/profiles do you have in your team/company? Or what kind of skills/profiles do you need in your team/company?
- Can you find talented people in the AR/VR sector? How do you see the industry from a human resources perspective?
- When you have a new team member, how do you observe their progress? How do they develop and contribute to your team over time?
- Let's move on to the tools your team is using. What kind of tools, programs, applications does your team use?
    - Specifically, authoring tools like Unity, Unreal Engine, Autodesk 3ds Max etc. How do you use them? In which parts of your product development process they are used?
        - Tell us more …
- These tools have visual scripting options (unity and unreal). Do you know such features? Does your team take advantage of such features?
    - If yes >> How do you use them, can you explain? What are the good things and and missing things about visual scripting? Advantages & disadvantages & things to improve?
- Last question, how do you see the future of the immersive technologies market (AR, VR, XR etc) ? What do you recommend to people who want to enter the AR/VR industry?
- Anything you would like to add related to AR/VR?

And some follow up questions according to their answers.

Blue: Common questions across participants
Green: Participant group specific questions

Questions for Developers/Designers/Creators.

- Can you introduce yourself? (Broadly: background and experience)
- How did you first meet AR/VR?
- Can you tell us the challenges and difficulties you had at the beginning of your AR/VR journey?
    - How did you overcome these challenges?
- Let's flip the coin, what about positive things, highlights? What kept you motivated to continue at the beginning of your AR/VR journey?
    - Follow up question related to main motivators.
- If we go more into your learning experience: How would you describe your journey until today? Can you draw the picture of your development journey?
- Throughout this journey, how could you transfer your skills from different domains like XXXXX (something from the first answer that they introduce themselves) into AR/VR?
- What kind of skills are required to become a successful VR/AR practitioner?
- Let's move on to the tools that you use. What kind of tools, programs, applications do you use?
    - Specifically, authoring tools like Unity, Unreal Engine, Autodesk 3ds Max etc. How do you use them? Can you elaborate?
- These tools have visual scripting options (unity and unreal). Do you know such features? Do you take advantage of such features?
    - If yes >> How do you use them, can you explain? What are the good things and and missing things about visual scripting? Advantages & disadvantages & things to improve?
- VR/AR market is changing rapidly, how do you keep up with changes?
- Can you summarize your day-to-day job? How do you see working as an AR/VR creator?
- What are your go-to resources/platforms to learn something new, or for troubleshooting or for better understanding?
- Last question, how do you see the future of the immersive technologies market (AR, VR, XR etc) ? What do you recommend to people who want to enter the AR/VR industry?
- Anything you would like to add related to AR/VR?

Blue: Common questions across participants
Green: Participant group specific questions

Questions for Students/Learners

- Can you introduce yourself? (Broadly: background and experience)
- How did you first meet AR/VR?
- Can you tell us the challenges and difficulties you had at the beginning of your AR/VR journey?
    - How did you overcome these challenges?
- Let's flip the coin, what about positive things, highlights? What kept you motivated to continue at the beginning of your AR/VR journey?
    - Follow up question related to main motivators.
- If we go more into your learning experience: How would you describe your journey until today? Can you draw the picture of your development journey?
- Throughout this journey, how could you transfer your skills from different domains like XXXXX (something from the first answer that they introduce themselves) into AR/VR?
- What is your development plan in this field? What kind of skills do you think that you should develop to become a successful AR/VR creator?

- Let's move on to the tools that you use. What kind of tools, programs, applications do you use or learn?
    - Specifically, authoring tools like Unity, Unreal Engine, Autodesk 3ds Max etc. how do you learn them? What kind of resources do you have?
- These tools have visual scripting options (unity and unreal). Do you know such features?
    - If yes >> How do you use them, can you explain? What are the good things and and missing things about visual scripting? Advantages & disadvantages & things to improve?
- What do you recommend to people who want to enter the AR/VR industry?
- Anything you would like to add related to AR/VR?

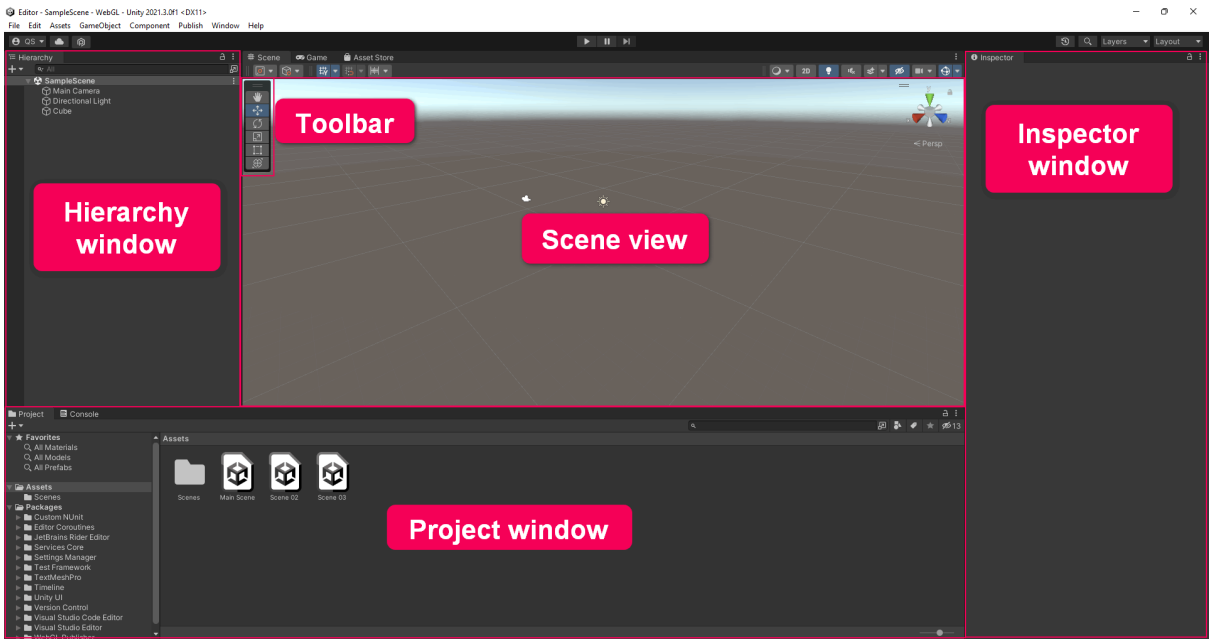
Blue: Common questions across participants
Green: Participant group specific questions

Appendix B
Unity User Test

Introduction to the Unity Editor

Find “UNITY HUB”, start it. Then open the project with your name. It will take couple of mins.

The default Unity Editor layout is organized with the most important windows you’ll need positioned to help you complete basic activities. Let’s start by reviewing them:



Scene view and Game view

In the center of the default Unity Editor layout is the Scene view. This is your interactive window into the world you are creating. You’ll use the Scene view to manipulate objects and view them from various angles.
In the default layout, the Game view also appears in this area; you’ll use the Game view to playtest your game.

Hierarchy window

The Hierarchy is where you can organize all the things in your project. These things are called GameObjects.
If you add a GameObject to your project in Scene view, it will be listed in the Hierarchy. If you delete a GameObject from the scene, it will no longer be listed.

Project window

The Project window is where you can find all the files (assets) available for use in your project, whether you use them or not.
The Project window works like a file explorer, organized in folders. You can drag assets directly from the Project window into the Scene view to add them to the scene.
Note the difference between the Project and Hierarchy windows: the Hierarchy contains all the GameObjects in the current scene, and the Project window contains all the assets available to your entire project.

Inspector window

The Inspector is where you’ll find and configure detailed information about GameObjects.
When you select a GameObject in Scene view or in the Hierarchy, you’ll see its components in the Inspector. Components describe the properties and behaviors of GameObjects.

Toolbar

Use the toolbar buttons to change your point of view in the scene, and start and stop Play Mode.
The scene navigation functions are hosted in a floating toolbar in your scene view. These functions allow you to move, rotate and scale your selected GameObjects.

Practice navigating the scene

When working in Unity Editor, navigating in the Scene view is very important. One way to think about navigating in this window is like operating a drone camera — it lets you examine your GameObjects from any angle or distance.

With practice, you can learn to navigate with ease. There are also more general settings you can use to configure the Scene view.

Let's quickly review the basics:

- **Pan:** Select the **Hand** tool in the Toolbar and click and drag in the Scene view to move your view.
- **Zoom:** Holding **Alt** (Windows) right-click and drag in the Scene view to zoom.
- **Orbit:** Holding **Alt** (Windows) left-click and drag to orbit around the current pivot point. Note: this option is not available in 2D mode.
- **Focus (Frame Select):** When a GameObject is selected, select **F** with your cursor in the Scene view to focus your view on that GameObject. **Note:** If your cursor is not in the Scene view, Frame Select will not work.

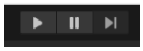
Flythrough mode

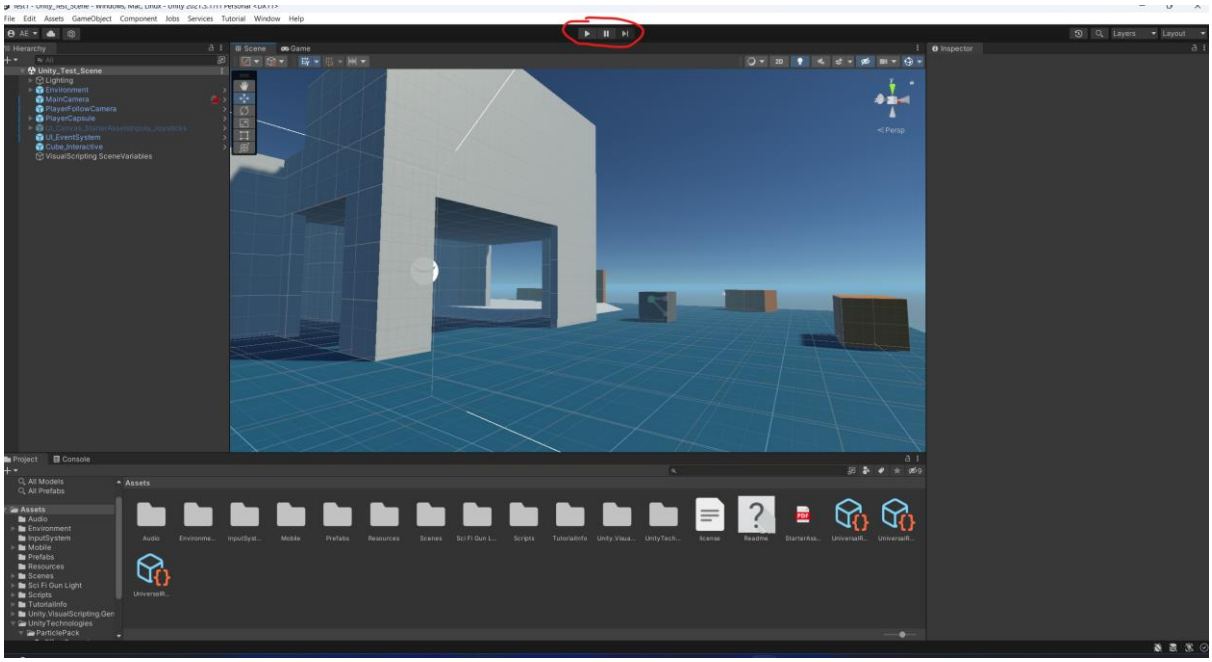
You can also use **Flythrough mode** to navigate in the Scene view by flying around in first person, which is common in many games. To do this:

- Click and hold the right mouse button.
- Use **WASD** to move the view left/right/forward/backward.
- Use **Q** and **E** to move the view up and down.
- Select and hold **Shift** to move faster.

Getting Started

You will see a screen like the following image. Try to right click your mouse in the scene and then move it around to see the 3D Scene in the scene view. You can move by using WASD. Top center, you will see Play and Pause buttons. Press play and walk in the scene.



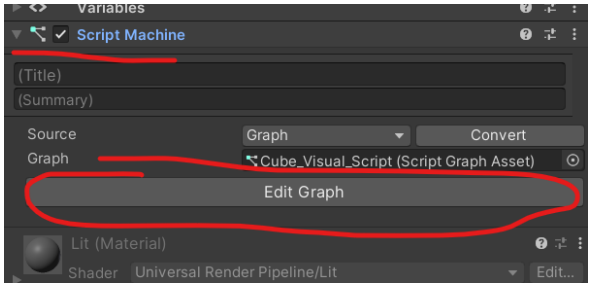


You can walk by WASD and shoot a projectile from your gun with left click. To exit game mode, press ESC and then deselect the PLAY button. You will add some interaction to make this simple scene more fun.

To create certain action, logical flow or interaction, Game Objects should be programmed. This is done by creating a script and then attaching this script to the game object that needs to be programmed. There are two scripting options: C# and Visual Scripting. In this study, we will assess how effectively Unity Visual Scripting enable people to program without writing any actual code.

Visual Scripting

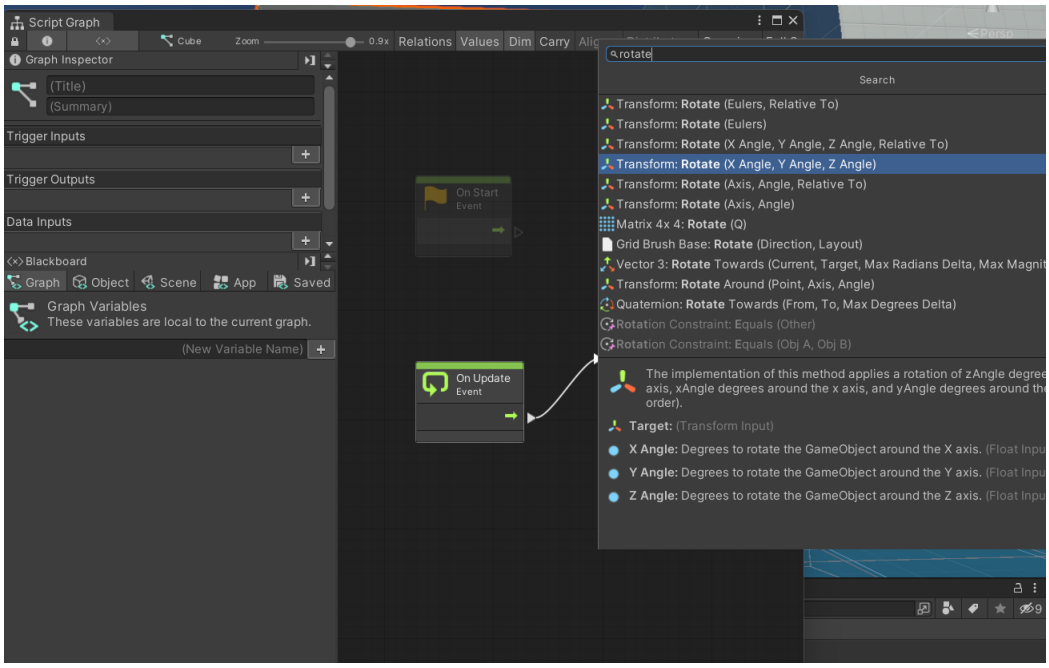
Let's learn about how to do it.



First, find the cube that is floating in the air in the scene. Alternatively, you can double click the "Cube" in the hierarchy. Then your view will be moved to the Cube's location.

When the cube (or any game object from hierarchy) is selected, you will see the details of the cube (or selected game object) on the right hand side, in the inspector window.

Locate the Script Machine and click Edit Graph. This is where the programming is done.



Task 0: Rotation

Visual Scripting is a tool that gives you opportunity to program game objects by connecting nodes.

You can navigate in the space by clicking middle mouse wheel. You can right click on the empty space, it will also show the menu. Alternatively, you can click and drag the white triangles, when it is released, the menu will appear.

You can type "Rotate" in the menu and select "Rotate (X Angle, Y Angle, Z Angle). Then type 1 in one of these variables. Save the script by "ctrl + S", close or minimize it. Navigate to the main editor and start the game mode by clicking Play button on top.

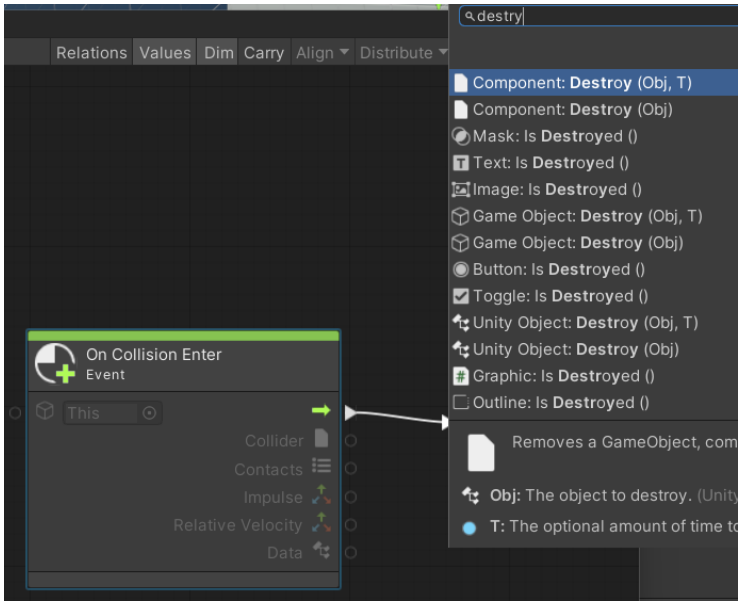
You should see the cube rotating now. What is happening? The "Update" is the function that literally updates the scene for every frame. It rotates the cube by 1 degree for every frame. Since there are not many things moving in the scene, the game runs at very highspeed like 100-200 frame per second (FPS).

Task 1: Destroy

Let's open the visual scripting again. To remove the connection between update and rotate, press "Alt" and then right click the arrow. The connection should disappear. If you want, you can select and delete the nodes that you do not want to use.

Now let's do something else, right click on the empty space and search for "collision" and select "On Collision Enter" from the options.

When one object collides with the game object (in this example the cube, because we are writing the script which is attached to the cube), the game object **enters** the **collision** state.



Pull the arrow from this event and search for "Destroy" then select Destroy (Obj, T). As the name suggest, this function destroys the object obj, in certain time T.

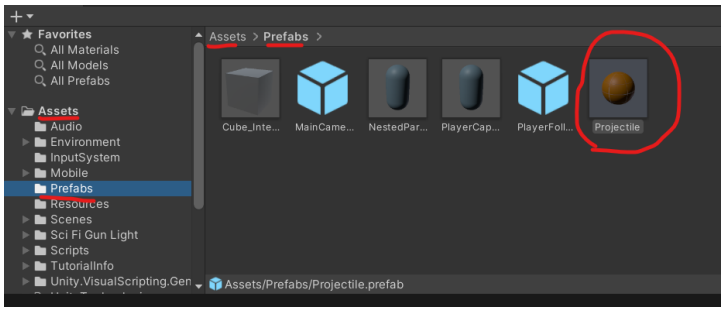
While the other functions are green, this Destroy function is orange. Because it is not ready yet. The destroy function requires two input: Object and time. We want to destroy the cube itself that we are programming. Then we can right click on empty space and search for "This".

"This" node points to the object that we are programming for. In this situation, it is the cube. Select the This node and connect it to obj on Destroy function. It is not orange anymore, because it has all the necessary input: Object to Destroy = "This" and Time to Destry = 0 secs. Let's make it 3 secs. Save the script ctrl + S, and then go into game mode and test your script.

When you are in the game mode, try to hit the box by shooting projectiles from your gun. What happens? You can exit from the game mode and make the time of destroy function to zero or small number make it more realistic.

Task 2: Explosion Effect

Navigate to project window, and open Assets > Prefabs > select Projectile.



This is Prefabricated copy of Projectile which is the object that our gun throws. For this game, the gun was programmed in a way that when the user clicks, a new copy of this projectile is created and then it was pushed in the direction that the gun facing.

When the projectile is selected, on the right hand side, you can see the Script Machine component like the cube had. Script Machine > Edit Graph. You can see it is empty now.

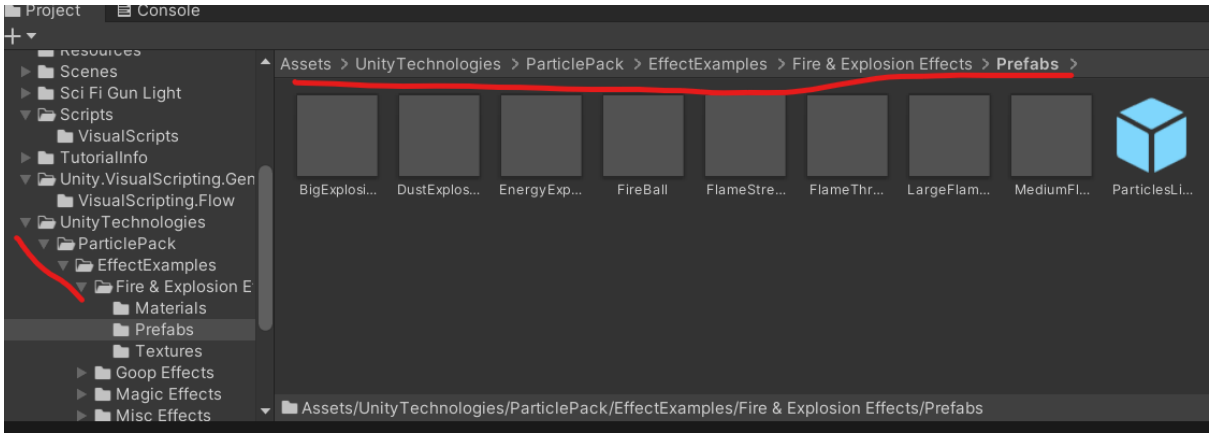
First, you need to add a node which is triggered when an object collides with another objects.

Then you need to add another node which can ***instantiate*** an ***original*** copy of a prefab (like projectile) at certain ***position*** and ***rotation***.

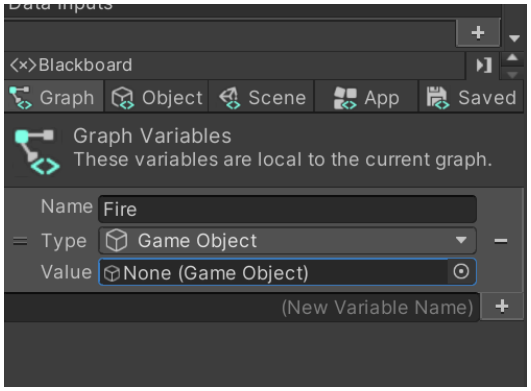
Explosion effect is achieved by particle systems. Our demo game already has this system installed.

See the screen shot in the next page to find the particle systems. We will use ***TinyExplosion*** in our test. You can navigate to the fire effects prefab folder and try other fire effects too.

We need to pass this TinyExplosion prefab to our Visual Script. Otherwise, the script itself does not know what to instantiate.



To pass this prefab to our script, we need to create a variable from the menu on left hand side. Please



type a name for your fire effect, here in this example, we call it “fire” for now. And then select the type as game object. And Value: click the circle with the dot inside. Type “tinyexplosion” and select the prefab it shows.

At this point, we defined our Game Object type as a variable named Fire. It will pass the TinyExplosion effect to our script.

Now, your task is to create a script that instantiates the TinyExplosion when the projectile enters a collision at at the place where the position and rotation comes from the projectile itself.

You are free to use any resources online, offline or anything. It is also good if you think aloud about the steps what you are going to do, what do you think about the task.

You can also ask questions to the host if you cannot find an answer or if you are not really clear about what to do. Please note that, we are not testing you, but we are testing how easy to use the visual scripting, how effective and intuitive it is. Do not hesitate to ask for help, but you need to try yourself first.

Try and test the fire effect, try to make it as good as possible by playing with it adding some extra thins or maybe by optimizing its effect.

Task 3 – Life

If you could successfully complete the Task 1, then let’s add one more logic in this game. Let’s add a life count to the cube that was in the scene. Then reduce the life by one when the player shoots the cube with the gun. When the life hits zero, the cube should be destroyed.

Not clear? Any question? Do not hesitate to ask!

Appendix C
Unreal User Test

Introduction to Unreal Engine Level Editor

Find “UNREAL ENGINE” and start it. Then open the project with your name. Give it a couple of minutes. The default Level Editor is organized to help you with the most important windows. When the editor is running, you will see an interface similar to this layout:

1. Tab and Menu Bar

Menu bar provides general tools and commands like edit, help etc.

2. Toolbar

It has group of commands and provides quick access to commonly used tools and operations. We will use Play option mostly.

3. Actos

This is where you can find Actors (object) like 3D objects, characters or other elements.

4. Viewport

The Viewport panel is your window into the worlds you create in Unreal Engine. This is the screen that you see and manipulate your level. In our example, you see the 3D scene that will be used for this test. You can place Actors (objects) in the viewport, adjust the 3D geometry or more.

5. Content Browser

You can navigate and access to the files and folders in your project.

6. World Outliner

The World Outliner panel displays all of the Actors within the scene in a hierarchical tree view. You can select and modify Actors directly from the World Outliner. Use the Info dropdown menu to display an additional column that shows Levels, Layers, or ID Names.

7. Details

This panel shows the details of the object (actor) that is selected in the level. For example, selecting a cube will show the location of the cube, its scale, material, texture etc.

Practice navigating the level

The viewport can be navigated just as you would in a game, There are various controls to enable you to navigate the scene, select and manipulate Actors, and change display options while working in the viewports.

Left Mouse Button	**LMB**
Right Mouse Button	**RMB**
Middle Mouse Button	**MMB**

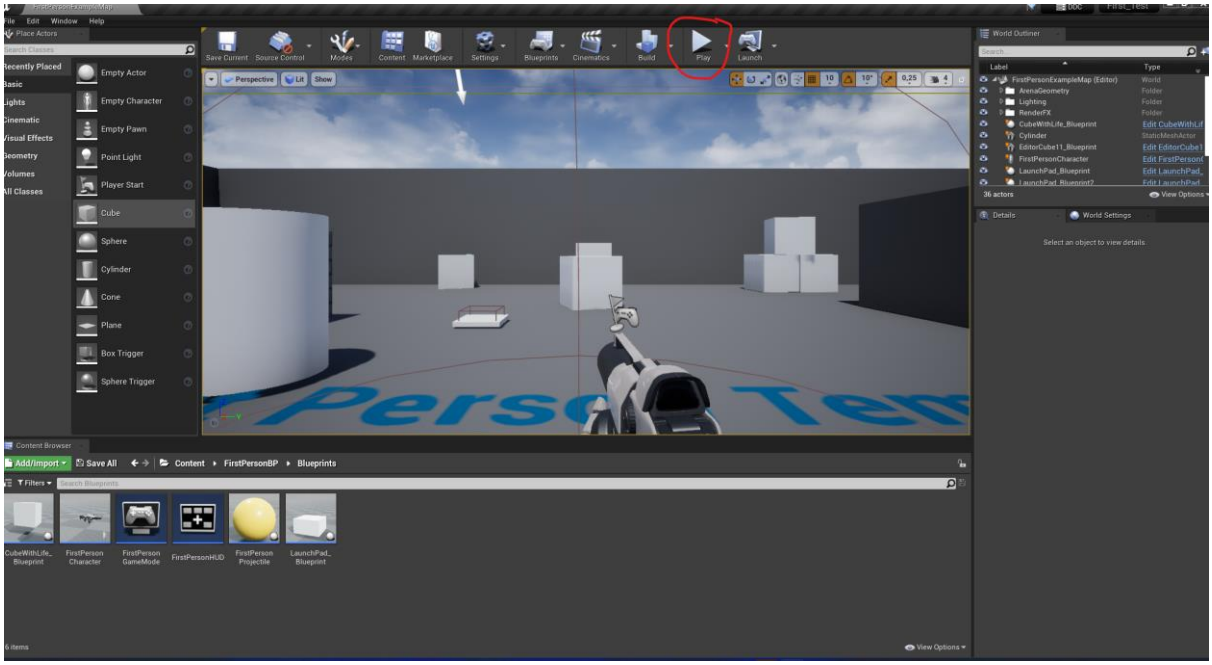
LMB + Drag	Moves the camera forward and backward and rotates left and right.
RMB + Drag	Rotates the viewport camera.
LMB + RMB + Drag	Moves up and down.

RMB + W/A/S/D	Moves the camera forward, left, backward and right
RMB + E/Q	Moves camera up and down
RMB + Z/C	Zooms the camera out and in

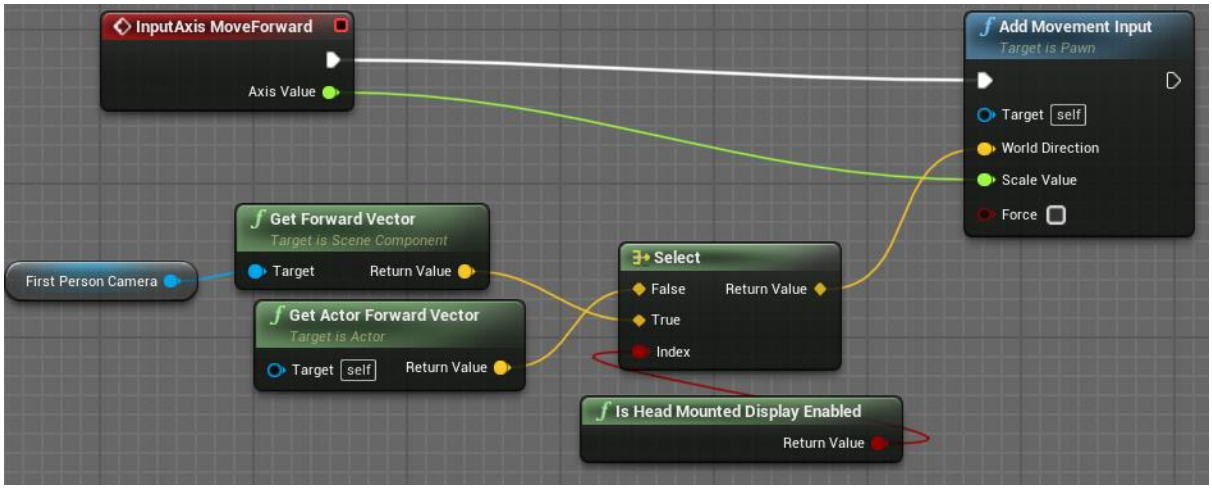
You can select Actors in the viewport individually simply by clicking on them.

Getting Started

You will start with a screen like the following image. You can move in the scene as it was explained above by using your mouse right click and WASD. If you click “Play” button in the top of the screen, you will simulate your game. Press play and try to explore the scene: walk around and shoot with your gun, interact with other objects. You can press “Stop” button or ESC to stop the simulation.



To create certain action, logical flow or interaction, Actors (Game Objects) should be programmed. This is done by creating scripts for actors. There are two scripting options: C++ and Blueprint. Blueprints are scripts that are developed by connecting visual nodes instead of writing C++ code. In this study, we will assess how effectively Unreal Engine Blueprints enable people to program without writing any actual code.

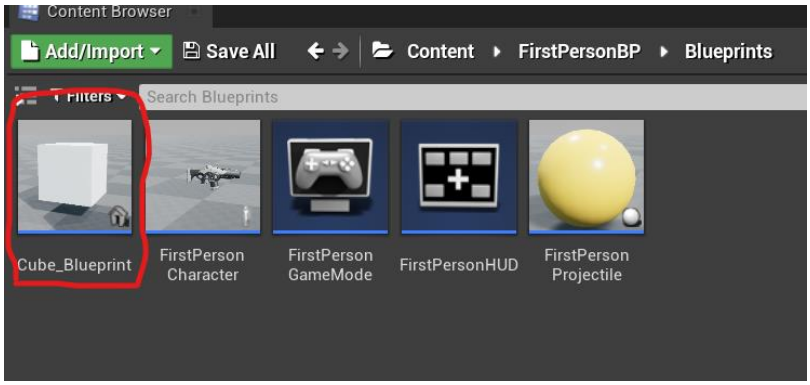


Blueprints example: programming an actor without writing code.

Blueprints

Let's do our first visual programming.

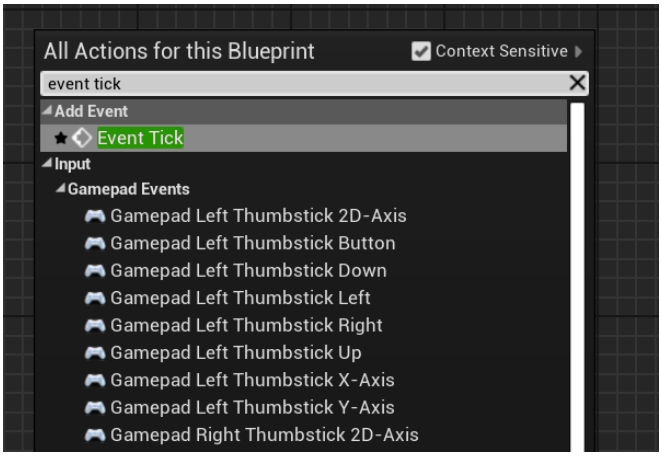
In the content browser, you will see "Cube_Blueprint". Double click on it and open the Blueprint editor.



Task 0: Rotation

We will create our visual scripting in this editor.

Right click on the empty space (you may also notice "*Right-Click to create new nodes"* message on top of the empty field)



Then in the dropdown menu, type "**Event tick**". Select this event and place the node in the field.

Once this node is placed, right click somewhere else in the empty field and search for "**Actor rotation**". And then select "**AddActorLocalRotation**" node.

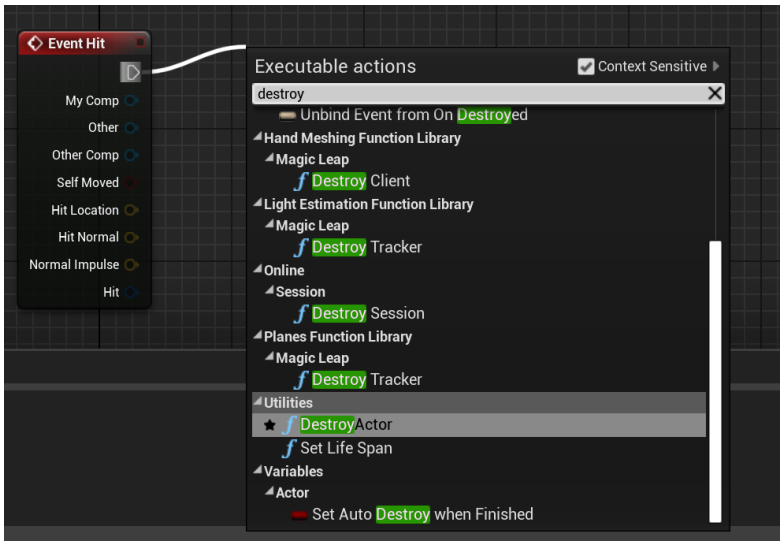
Left click on the white triangle on the Event Tick node and connect the node to the white triangle on the left of AddActorLocalRotation node.

Type 1 in one of the X Y Z values. Save the Blueprint by ctrl + S or from Save button on the top. Start the game simulation by clicking play button in the editor main page.

At this point, you should see the cube on left is rotating. What is happening here? The Event Tick node is triggered with every frame update during the game. And this node was connected to local rotation function. Result: 1 degree of rotation is added to the cube at every frame update.

Task 1: Destroy

Let's add an interaction to our game. We will destroy the cube when we hit with the projectile that we shoot from our gun.



First, we need to find the hit function in our Blueprint. Right click and search for "**Hit**", then you should find "**Event Hit**" option, add it to the scene.

After adding our event hit, you can drag the line from Event Hit node. This will automatically open the menu where you can search for "**Destroy**", Select "**DestroyActor**"

DestoryActor node is added to our script and it is already connected to the hit event.

Save the script, play the simulator and try to hit the rotating cube.

Task 2: Explosion Effect

We are going to add some visual effects to our game. When our projectile hits a target, we would like to have an explosion effect. To do this we need to program our projectile actor. Double click on the FirstPerson Projectile object in the same menu screen as Cube_Blueprint. (See the image on previous page). Explosion effects are handled by **emitters** in Unreal Engine. As the name suggest, emitter is an actor that produces emission of heat and light.

The logical flow is as follows: When a **hit event** occurs, we want to **Spawn Emitter at location.** In other words, when our projectile hits another object, a new emitter actor will be created in the scene at location. Now your task is to create this explosion effect.

You are free to use any resources online, offline or anything. It is also good if you think aloud about the steps what you are going to do, what do you think about the task.

You can also ask questions to the host if you cannot find an answer or if you are not really clear about what to do. Please note that, we are not testing you, but we are testing how easy to use the visual scripting, how effective and intuitive it is. Do not hesitate to ask for help, but you need to try yourself first.

Try and test the fire effect, try to make it as good as possible by playing with it adding some extra thins or maybe by optimizing its effect.

Task 3 – Life

If you could successfully complete the Task 1, then let's add one more logic in this game. Let's add a life count to the cube that is rotating in the scene. Then reduce the life by one when the player shoots the cube with the gun. The cube should be destroyed once it runs out of life.

Not clear? Any question? Do not hesitate to ask!

Appendix D
Authoring Tools Review

	A	B	C	D	E	F	G	H	I
1	**Tool Name**	**Type**	**Availability**	**Function**	**Novelty**	**Documention**	**Coding**	**Integration**	**Remarks**
2	**HoloBuilder**	From research to Commercial	License	AR based construction process management	View and use 360 images/point cloud etc.	Well	No-code	No integration,It is a complete solution now	Standalone niche commercial product not for AR Authoring tool anymore
3	**ProtoAR**	Research	Not usable but source code is available	Capture physical objects and use in AR	Rapid prototyping from physical to AR	Basic	No-code	Not possible	It did not work, probably it is outdated. Source code available, can be improved further
4	**GestureWiz**	Research	Not usable but source code is available	Gesture recognition tool	Gesture based interface	Basic	No-code	Not possible	No product to test but source code is available and can be improved
5	**InVision**	Commercial	Available	Mobile and web design	Design and prototyping tool	Well	No-code	Not for AR/VR but for other domains	Acquired by Miro
6	**Adobe XD**	Commercial	License	Mobile and web design	Design and prototyping tool	Well	No-code	Not for AR/VR but for other domains	
7	**Sketch**	Commercial	License	Mobile and web design	Design and prototyping tool	Well	No-code	Not for AR/VR but for other domains	
8	**DART**	Research	Obsolete	Content and AR object creation	Rapid AR prototyping	Obsolete	Low-code	Not possible	One of the very first example of low-code rapid AR prototyping
9	**Proto.io**	Commercial	License	Mobile and web design	Design and prototyping tool	Well	No-code	Projects from other tools can be imported	
10	**ARtoolKit**	From research to Commercial	Dissovelled after acqusition	Software library for AR prototyping	Rapid AR prototyping	Moderate	Moderate	Can be integrated to Unity	Open-source system was maintained under new name ARtoolKitX
11	**Tiles**	Research	Not available	Marker based AR authoring tool	Rapid AR prototyping	Not available	Moderate	Not possible	One of the very first example of rapid AR authoring tools
12	**Studierstube**	Research	Not usable but source code is available	AR authoring tool	Multi-user prototyping	Basic	Moderate	Not possible	Early example: Multiple users can interact on the same visual object independely.
13	**ComposAR**	Research	Obsolete	AR and MR authoring tool and framework	Supports both visual programming and interpretive scripting	Obsolete	Low-code	Not possible	Early example of visual scripting and game engine style interface
14	**Facebook AR Studio (Meta Spark Studio)**	Commercial	Available	AR effect creation tool for social media platforms	Complete tool	Well	low-code to high	No direct integration but complete platform for social media	
15	**Snapchat Lens Studio**	Commercial	Available	AR effect creation tool for social media platforms	Complete tool with GenAI assistance	Well	low-code to high	No direct integration but complete platform for social media	Available tutorial for Unity developers
16	**Teddy**	From research to Commercial	License (Unity asset)	3D content creation on Unity	Rapid 3D content	Moderate	No-code	Unity integration	
17	**Lift-Off**	Research	Not available	3D content creation in VR	Rapid 3D content	Not available	No-code	Unreal Integration	
18	**Google's SketchUp**	Commercial	License	3D modelling	Profesioal rapid 3D Modelling	Well	No-code for regular use	Unity and Unreal Integration	
19	**Google's Blocks**	Commercial	Free / obsolete	3D content creation in VR	Rapid 3D content	Obsolete	No-code	With in VR, no integration needed	
20	**3DS Max**	Commercial	License	Profesional 3D modelling and rendering	Robust modelling and high qualittyrendering	Well	Coding and visual scripting	Objects can be imported to Unity and Unreal	

	A	B	C	D	E	F	G	H	I
1	**Tool Name**	**Type**	**Availability**	**Function**	**Novelty**	**Documention**	**Coding**	**Integration**	**Remarks**
21	**Maya**	Commercial	License	Profesional 3D animation VFX	Real life like animation	Well	Coding	Objects can be imported to Unity and Unreal	
22	**Unity**	Commercial	License	Profesional game, VR/AR development	Complete authoring tool	Well	Coding and visual scripting	Import and export possible	
23	**Unreal Engine**	Commercial	License	Profesional game, VR/AR development	Complete authoring tool	Well	Coding and visual scripting	Import and export possible	
24	**A-Frame**	Commercial	Open source	WebVR development framework	Complete authoring tool	Well	Coding	From Unity to A-frame	
25	**Vizor**	Commercial	Open source	WebVR debelopment	Rapid development with visual scripting	Moderate	visual scripting	No integration, WebVR	In the article by Ashtari, it was mentioned as Visor which is another VR headset company
26	**Torch**	Commercial	Obsolete	AR development	No code AR development	Obsolete	No-code	Not available	Torch VR is not available anymore.
27	**Argon.js**	Research	Discontinued, source code available	Web AR development	Easy Web AR development	moderate	Coding	Not possible	
28	**360proto**	Research	Not usable but source code is available	Capture physical objects and use in VR/AR	Rapid prototyping from physical to VR/AR	Basic	No-code	Not possible	
29	**ARtalet**	Research	Not available	AR authoring tool for another research	AR Prototyping tool	not available	No-code	Not possible	
30	**iaTAR**	Research	Not available	AR authoring tool	Immersive authoring	not available	No-code	Not possible	
31	**Amire**	Research	Not usable but source code is available	MR Authoring tool	No code MR development	Well	No-code	Not possible	EU funded project
32	**XRDirector**	Research	Not usable but source code is available	VR/AR authoring tool	Immersive authoring	Basic	No-code	Not possible	
33	**ARKit**	Commercial	Available	API for Apple products	Official API from Apple	Well	Coding	Unity and Unreal Integration	
34	**ARCore**	Commercial	Available	SDK for Android devices	Official SDK from Google	Well	Coding	Unity and Unreal Integration	
35	**WebXR**	Commercial	Available	API using HTML5 and WebGL	Cross platform	Well	Coding	From Unity/Unreal to WebXR	
36	**Pronto**	Research	Not available	AR authoring tool	Rapid prototyping	Obsolete	No-code	Not possible	Adobe did not develop it further, probaböy prepared base for Adobe Aero
37	**PowerSpace**	Research	Not available	AR authoring tool	Rapid prototyping	Obsolete	No-code	From MS PowerPoint to AR	User presentation slides in AR
38	**ARVIKA**	Research	Not available	AR authoring tool	AR for industry	Obsolete	Coding	Not possible	Funded by German Ministry of Education and Research
39	**Adobe Aero**	Commercial	Available, free	AR authoring tool	No code authoring	Well	No-code	Standalone authoring tool	
40	**wiarframe**	Commercial	Not available	AR authoring tool	Rapid AR prototyping	Obsolete	No-code	Standalone authoring tool	

	A	B	C	D	E	F	G	H	I
1	**Tool Name**	**Type**	**Availability**	**Function**	**Novelty**	**Documention**	**Coding**	**Integration**	**Remarks**
41	**Microsoft Maquette**	Commercial	Available but discontinued	VR scene building	Immersive VR scene building	Well	No-code	Can be integrated to Unity	Microsoft phasing out and will completely remove
42	**Reality Composer**	Commercial	Free	AR Asset creation	Rapit 3D asset creation	Well	No-code	Possible to export AR assets	
43	**Gravity Sketch**	Commercial	Free for individuals	Collaboration and 3D design tool	Immersive 3D design	Well	No-code	Possible to export 3D assets	
44	**MRTK**	Commercial	Free	MR Software development kit	For Hololens, but supports others too	Well	Coding	Unity and Figma integration	