POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Improving training and learning methods in extended reality: user attention in the learning environments

Supervisors:

Prof. Andrea **Bottino**
Prof. Francesco **Strada**

Candidate:

Agnese **Serafino**

December 2024

Abstract

Technologies made for supporting learning are advancing very rapidly, especially applications that use virtual reality (VR) and augmented reality (AR): some meaningful examples are pilots training, tutoring of novice medical students in the operating room and firefighters' preparation. These innovative tools make the learning process more engaging and interactive, making it possible for people to dive into environments which are not always easily accessible. The goal of this thesis is to study and improve applications that use procedural learning with VR/AR headsets, focusing especially on the user attention for optimizing the user experience and enhancing the level of assimilated knowledge. However, sometimes the prolonged exposition to these complex virtual environments could induce the user to lose focus and lead to cognitive overload, with the risk of compromising the learning effects and making this goal unattainable. Starting from the examination of the existing literature and theoretical research, the aim of this thesis is to improve the current VR/AR applications that use procedural learning and reduce the factors that could induce excessive mental effort. To do so I developed a Unity library aimed at implementing and enriching of the techniques identified during the research. In particular, I implemented a method to verify in real-time that the user's focus is at the explanation in progress, with the possibility of guiding them towards the object of interest if they become distracted. This is obtained through a directional arrow that points toward the target object that is also highlighted by other UI components. The testing phase of this project will be carried out with a sample of twenty-four people divided into two clusters of twelve people each: the first group will conduct the experience without the integration of the user attention functionalities, while the second one will have the full experience with no limitations. The testing scenario has been chosen carefully to minimize the possibility of distractions due to the complexity of the tasks or the environment. The aim of this thesis is to assess if, and in which measure, the usage

of these techniques can enhance learning-oriented applications. This work has been carried out using Oculus Quest 2, but the proposed solutions can be also integrated in augmented reality applications using, for example, the Hololens2 headset.

Contents

List of Figures

Listings

List of Tables

Chapter 1

Introduction

In today's tech-driven world, education and training require innovative methods that actively involve learners, making it enjoyable and easier to follow procedures without feeling overwhelmed. With procedural learning people acquire knowledge through the repetition of ordered tasks, so it differs from the traditional declarative learning in which the concepts can be abstracted and summarized.

Ganier identified a pattern in how people learn: "as users inspect the instructions, they form a mental model of the task and its steps. Forming this mental model involves integrating information from the instructions and equipment, as well as retrieving prior knowledge from the user's long-term memory storage" [2]. After this step, learners then devise ways to achieve their goals using the knowledge and tools they have. Summarizing this model, users build a permanent knowledge by setting a goal, creating a mental model (so thinking of an action plan, considering the equipment), taking information from the long term memory and integrating it with the working memory. After all these steps they will have learned something that will go to the long term memory.

1.1 Training procedural tasks: procedural learning

1.1.1 Kolb's experiential learning cycle

The experiential learning cycle, influenced by the work of psychologists and educators such as John Dewey, Jean Piaget, and Kurt Lewin, describes the process that takes a person with no knowledge to achieve experience. This progression starts with concrete experience, where the user learns by doing, then it goes through the reflective observation where they reason about the experience they just had and draw their conclusions (such as finding errors and understand how to improve it). In the abstract conceptualization phase, learners develop concepts from these conclusions, and in active experimentation, they apply their new knowledge toward achieving a goal, continuously cycling through these steps as they practice and refine their skills. [3]

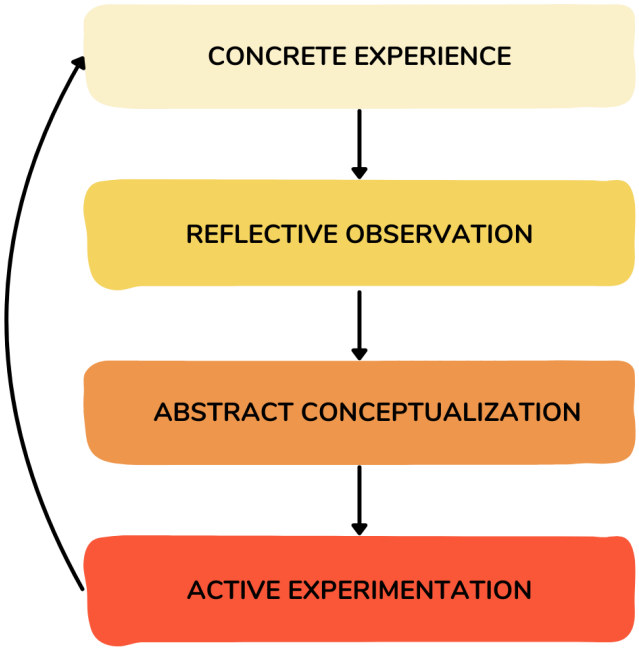

Figure 1.1. Kolb's experiential learning cycle [1]

1.2 Cognitive theories

When considering how people learn and use their brains, it is essential to address potential complications; an example is the importance of cognitive overload, which happens when the requests made to the active memory exceed its maximum capacity. Miller's law assumes that an average person can hold between five to nine requests to the active memory at a given time [4].

1.2.1 Cognitive demands

There are three main kinds of cognitive demand:

- Essential processing: the cognitive processes necessary for understanding the presented material.
- Incidental processing: it refers to cognitive processes that are not required for making sense of the presented material but are primed by the design of the learning task.
- Representational holding: in this category there are the cognitive processes aimed at holding a mental representation in working memory over a period of time

The total processing intended for learning consists of essential processing plus incidental processing and representational holding. Cognitive overload occurs when the total intended processing exceeds the learner's cognitive capacity [5]. Although reducing accidental processing might seem like a good solution, would minimizing details still provide an engaging user experience? Richard E. Mayer and Roxana Moreno addressed these questions in their cognitive theory of multimedia learning.

1.2.2 Mayer and Moreno's cognitive theory of multimedia learning

In their study, Mayer and Moreno presented five scenarios involving cognitive overload in multimedia learning and proposed nine different solutions to these problems.

1. In the first scenario, a student must watch a video while also reading an on-screen text at the bottom of the screen. "This situation creates what Sweller (1999) called a split-attention effect because the learner's visual attention is split between viewing the animation and reading the text" [5].
   Solution: narrating the text instead of displaying it on-screen; in this way the learner can focus their attention to the real scope of the experience. This technique is called off-loading.

2. The second one is a scenario in which the information is presented in too many forms (images, text, audio).
   First solution - Segmenting: Providing more time between bits of information so that the user can process each of them.
   Second solution - Pretraining: The second method to solve this problem could be offering the student a set of key-words or chunks of information beforehand to help them understand the content more quickly.

3. In another scenario, peripheral material such as animations and sounds distracts the learner from the core task leaving less cognitive capacity for essential processing [5].
   Solution 1 - Weeding : this technique consists in "simplifying the narrated animation to make it as clear and focused as possible, so the learner isn't distracted by unnecessary information and can concentrate on the essential content." [5].
   Solution 2 - Signaling : highlighting key concepts and signaling them with graphic cues such as arrows.

4. This problem can be similar to other already mentioned and it involves a confusing presentation of the material.

Solution 1: This solution involves having all the materials close together for easier understanding.

5. In the final scenario, learners try to process both essential and representational information simultaneously, for example, by listening to a narration while viewing an animation.
   Solution: An approach in this case could be to synchronize the material, so that animations and narration are presented together.

These nine ways of reducing cognitive overload have been deeply analyzed and, taking inspiration from these advices, in this project I decided to use only verbal instructions and visual cues to help the user understand the real goal of the experience without being distracted by the text. Moreover the learner will be helped in understanding the real focus of each task by an arrow that will indicate the direction of the goal.

1.3 User attention

This thesis centers mainly on user attention, specifically on ensuring that the player stays engaged with both the task explanation and the task itself. In a VR world getting distracted can be very easy, especially for people who have never used a VR headset before; this could mean that gaining user attention in this environment could be harder than doing so in a non-VR application. Researchers from the School of Sofware at Tsinghua University (China) have proposed SpatialGaze, "a spatial gaze tracking approach based on the realistic parallax-contingent visual model" [6]. Starting from an attentive study of the human vision, "SpatialGaze is designed based on the realistic parallax-contingent model, which enables the inference of visual axes' virtual and invisible parameters and accurately determines the user's gaze in space" [6]. SpatialGaze is described as a lightweight and accurate method to enhance the accuracy and overall usability of 3D displays and interactions in XR [6]. In particular, cameras are used to acquire eye images: initially, the researchers developed a Convolutional Neural Network (CNN), which is

a type of machine learning model, which processes the eye images as input and identifies and marks pupils, iris and eyelid. The same images are then put as input in an encoder-decoder network, which segments the pixels into the three previously mentioned categories. After these steps, they located the center of the pupil and the eyeball's rotation center to generate a 2D to 3D mapping. From those results they determinated the spatial gaze [6]. A limitation of this research is the need for two additional cameras to track eye movements, which may not be accessible to everyone. Another system for detecting attentional behavior in Augmented Reality (AR) is ARtSENSE. This project was developed to monitor the attention of museum visitors by creating an algorithm capable of estimating pupil position from low-resolution eye-tracking images. The algorithm is designed to analyze eye movements and facilitate interaction with the system through gaze, providing insights into attention estimation. [7] Although user focus can be assessed through eye-tracking mechanisms and computer vision algorithms, none of the works observed used the results obtained from such attention-checking mechanisms to improve learning experiences. Furthermore, the limited studies on this topic has prompted me to conduct a more in-depth analysis.

1.4 Spatial awareness

"Spatial awareness in our context is the ability to infer one's interaction potential in a complex environment from one's continuous sensorimotor assessment of the surrounding virtual environment" [8]. Consequently, thanks to the spatial perception we are able to navigate one's environment in relation to various obstacles and objects. In relation to this topic, Wallgrün, Bagher, Sajjadi and Klippel from The Pennsylvania State University conducted a study on the most efficient form of target-finding mechanism; in particular they based their research on visual mechanisms, which they further divided into contextual cues and overview+detail approaches: "the first group using some visual overlay to guide the user to the target, while in the second group a sort of overview is added in addition to the more detailed main view, specifying

the location of the target relative to the user's current view" [9]. They studied three main interfaces using an arrow, an halo and a wedge which come from solutions proposed by various researchers such as Burigat and Chittaro. Another important method analyzed is the firefly approach, which consists in an object that comes into the user view and moves towards the off-screen target [9]. The study compared three guidance mechanisms (an arrow, a "butterfly guide", and a radar) against an application with no guidance at all. The results are summarized as follows:

- Arrow: This was the preferred method, described as "intuitive, easy to learn and use, aesthetically pleasing, helpful, and least annoying" [9]. It was also classified as "the best candidate among the compared options to serve as a default option that everybody will be able to understand and use" [9].
- Butterfly Guide: Opinions on the butterfly guide were extremely polarized. Some users loved it, while others hated it and found it too distracting. Some suggested that it might have been more effective it if it used another animal instead of a butterfly [9].
- Radar: Same goes for this as it was for the butterfly guide; some people appreciated "that the mechanism directly tells them where exactly the target is located relative to their current view direction" [9]. Some critiques about this methodology were the positioning and the size.

Other researchers have also explored methods to achieve similar goals. Irlitti, Jackson and Thomas tested various types of visual cues and evaluated their performance based on user accuracy, mental effort, and task completion times, both in single and multiple-collaborator settings. They categorized visual cues for spatial awareness into five main types:

- Physical Attachment, Local Animation: "This cue is represented as a 2D arrow projected at the physical user's feet, pointing in the direction of each virtual user. The arrow cue is presented in 3(a). The cue is attached to an invisible circle which surrounds the physical user, using this guide for rotation purposes while the virtual user moves throughout the scene" [10].

- Physical Attachment, World Animation: A variant of the local animation method in which the arrow animation extends beyond the player and is integrated into the 3D environment.
- Virtual Attachment, Local Animation: It is represented by some concentric circles with increasing transparency towards the center. They wanted to simulate the act of breathing through this animation. "This allows the local collaborator to maintain vision of the virtual collaborators feet and also allows multiple collaborators to be represented by the glow while maintaining adequate spatial understanding." [10]
- Virtual Attachment, World Animation: Similar to the previous method, but it is represented by a animated, dashed circular ring which pulses throughout the environment until it reaches the other player.
- Exocentric Visualization: This approach creates a "reproduction of the physical environment inside a reduced rendering visualized beneath the floor." [10]

The results of this study showed how using virtual attachment cues, especially those with local animation, could minimize mental load and ensure high accuracy. Both studies validated the arrow method as the most effective approach for conveying spatial information to users. This conclusion is based on the method's ability to balance two critical factors: the minimal mental effort required for users to comprehend and interpret the spatial cues, and the relative ease with which users were able to locate the target. The arrow method consistently demonstrated higher performance in guiding users to their final destination, making it a reliable and efficient tool for spatial communication.

1.5 Goals and expected outcomes

This research will involve an in-depth investigation of visual cues currently used to direct attention in VR environments. Special attention

will be given to understanding how these cues can reduce cognitive load while maintaining the user's focus on key tasks. The effectiveness of different techniques will be assessed through user studies that examine ease of use, and user satisfaction. Furthermore, the study will explore how these methods apply to procedural training scenarios, where maintaining spatial awareness and user engagement are critical for successful learning outcomes. By conducting user studies, this thesis wants to understand if and in which measure these approaches can help improving learning and training applications. The expected outcome of this research is a comprehensive evaluation of visual guidance mechanisms, with a focus on how to check user's attention in each moment. Ultimately, this thesis aims to contribute to the improvement of immersive learning environments by demonstrating how targeted visual cues can enhance user experience, increase task efficiency, and support long-term learning in virtual reality.

Chapter 2

Literature Review

Currently, procedural learning is predominantly applied in academic, military, and industrial fields, where its structured and step-by-step approach is well-suited to complex training scenarios. In this research, I have narrowed my focus to its implementation within VR/AR environments, specifically examining the state of the art in methods for attention guidance and spatial awareness. By analyzing how these techniques are utilized across various applications, my goal is to identify recurring patterns and principles that can enhance user interaction and effectiveness in immersive learning experiences.

2.1 Academical uses

In the academic field, procedural learning has been adopted to enhance educational tasks and improve skill acquisition in virtual environments. In particular, these approaches are widely used in the medical field due to their cost-effectiveness and accessibility to all. VR/AR applications provide a more affordable alternative to traditional training methods, allowing medical professionals to practice in a controlled environment. Additionally, these systems can meticulously track and assess each action, ensuring that procedures are performed with precision and offering real-time feedback to ensure that tasks are completed in the most accurate way possible .
Through my analysis, I explored how VR/AR technologies are being

used to simulate real-world scenarios, providing students with interactive, hands-on learning experiences that promote deeper understanding and retention. I analyzed these works to gain a deeper understanding of which spatial awareness techniques and UX feedback mechanisms were most commonly utilized, helping to highlight the most effective methods for enhancing user experience and interaction within these environments.

2.1.1 TrainAR

TrainAR is an Augmented Reality application developed by Blattgerste, Luksch, Lewa and Pfeiffer from the University of Applied Sciences Emden. Its main goal is to teach procedural tasks through an interactive AR experience on mobile devices. One of the main use cases developed is for midwifery training, specifically preparing a tocolytic injection. The interaction concept includes selecting and manipulating virtual objects and receiving feedback, but also important is the possibility of having custom actions that could be quizzes or mini-games. Important to the scope of this thesis is noting that in this experience "correct actions always trigger visual feedback, e.g. in form of a green blinking outline of the object and auditory feedback" [11]. In their opinion the feedback for a wrong action must be more complex than the one used for correct answers. One specific feedback mechanism they implemented involves highlighting the selected object with a green line if the answer is correct, or a red line if it is incorrect.

2.1.2 HoloAnatomy

"Case Western Reserve University School of Medicine developed a holographic anatomy programme that includes all of the anatomy a preclinical medical student is required to learn in the dissection laboratory on a holographic male and female" [12]. It is used thorugh the Microsoft Hololens headset. This experience features a verbal explanation of the topics, accompanied by real-time highlighting of the object currently in focus. Notably, the experience is voice-activated, allowing the user to interact without the need for buttons. Another key aspect is the use of

static tags that display the name of the object being highlighted.

2.1.3 Extended Reality Advanced Trauma Life Support (ATLS) simulator

Another notable example is the ATLS simulator, developed by Virginia Tech in conjuction with Penn State Health, where "the trainee is immersed in an XR environment and asked to correctly diagnose a virtual patient following the ATLS guidelines" [13] and their goal is to provide medical training for emergency trauma response. In this project, the user experience incorporates a comprehensive checklist that summarizes all the tasks the user must complete. This checklist serves as a guiding tool, providing progressive assistance throughout the experience, ensuring that the user knows what actions are required to be done to progress. Similar to the previous example, this project also uses verbal instructions to deliver key information, offering real-time guidance and reinforcing the tasks at hand.

2.2 Human Anatomy VR

Human Anatomy VR , developed by Virtual Medicine, is available for a range of VR headsets, including Meta Quest, Apple Vision Pro, Playstation VR and Pico. In this application users take on the role of a students learning the human anatomy by interacting with a detailed body replica, where MRI scans can be analyzed in detail, and even shrink down to a smaller scale to navigate within the body itself. When a body part is selected, it is highlighted in color and a label appears next to the selected item identifying the part. Additionally, a static text box on the left side of the screen provides a detailed description of the selected body part, enhancing the learning process through visual and textual cues.

2.3 Industrial uses

With the rise of Industry 4.0, there is an increasing demand for new and more efficient methods to train workers. VR/AR applications offer a solution by providing immersive training experiences across various fields. For instance, these technologies can be used to train workers in industrial settings or pilots in aviation. Below are a few examples.

2.3.1 AR repair for industrial sewage

The AR experience developed by Cyent simulates the maintenance of an industrial sewage system by integrating various feedback mechanisms, as discussed in section 1.4. In this simulation, users are guided through a series of tasks that are outlined in an interactive checklist, ensuring they stay on track and understand the objectives at each stage of the process. To facilitate task completion, the objects relevant to the maintenance procedure are visually highlighted, drawing the user's attention to key elements. Furthermore, the steps required to complete each task are presented through a variety of instructional formats, including textual descriptions, 3D demonstrations, and video tutorials. This multimodal feedback allows users to engage with the content in multiple ways and enhancing comprehension.

2.3.2 Augmented Reality for aircraft maintenance, remote support and training

One field where AR training has seen significant adoption is the aircraft industry. AR provides an innovative solution for training new technicians, allowing them to acquire essential skills more quickly and effectively. This not only accelerates the training process but also ensures that mechanics are better prepared for real-world tasks. Both the company and the trainees benefit from this approach- trainees can gain confidence and proficiency faster, while companies reduce training costs and minimize disruptions to operations. In particular, Microsoft has developed an AR application, aimed at training new mechanics, which is made accessible through the HoloLens 2. In this application,

users are guided through each task with the help of a textual interface that explains the steps to be followed. What makes this application particularly relevant to the focus of this thesis is its use of visual guidance: as the trainee progresses, they are directed toward the target object or location by a dashed arrow that appears in the AR environment. This arrow dynamically updates as the user moves, creating a clear path from the user's current position to the target, ensuring that the trainee stays on track and knows exactly where they are meant to be.

2.3.3 WorkLink by ScopeAR

WorkLink is an AR application available through Microsoft HoloLens or through a handheld device. It has been developed by ScopeAR and it has three main fields of application: aerospace and defense, aviation and industrial. Its distinctive trait is that it does not use any 3D model to represent objects but it "makes use of a combination of stock content such as arrows and tools, images, and a model of the work space scanned with a 3D camera equipped smartphone to provide very effective instructions" [14]. In particular, it uses these shapes for drawing attention to the object of interest, making the user understand where to focus.

2.3.4 DigitalNauts pharmaceutical training

DigitalNauts is a British company that specializes in VR application for industrial use. Very interesting is their Pharmaceutical VR Training, which is "a virtual reality (VR) training solution for pharmaceutical companies to safely learn how to use the Chromaflow column packing station" [15]. The application is designed for use with the HTC Vive Puck, using teleportation as the locomotion method. A key focus in its development was strengthening the user experience (UX), led by the belief that in VR users are not meant to read text but to experience and interact with the environment in ways that reflect real-life scenarios, and "within the field, it is widely known now that VR learners tend to dismiss text quickly, unless short and tailored to the experience." [15]

For this reason, they used guided assistance such as AI guides, implemented through the use of arrows and visual cues that show exactly the movement that needs to be executed, because they create a more natural interaction and a more immersive experience. After each task is completed, "the results are presented to the user along with a list of good and bad practices. This offers the opportunity to reflect on choices made while also giving direct and instantaneous feedback for better knowledge retention. Upon completion, the user is also given the chance to continue or repeat the exercise. " [15]

2.3.5 HoloLens Maintenance for ŠKODA AUTO

This HoloLens application, developed by Brainz Immersive, was made specifically for Škoda employees to train them in machinery maintenance. The creators describe this experience as the perfect alternative to a textual manual, allowing users to engage in hands-on scenarios to solve various challenges. [16] The application provides brief textual instructions, supplemented by images of the relevant objects and arrows that not only highlight the item of interest but also indicate the path to follow. However, a limitation of the application is that users must interact with the UI to confirm the completion of each task, which may increase the cognitive demands and potentially divert from the overall learning experience.

2.4 Other use cases

2.4.1 Evacuation training

In the last few years, a lot of companies have opted for AR evacuation trainings, which are more effective than theoretical courses about the topic. Researchers from the Wageningen University (The Netherlands) and from Istanbul Kultur University (Turkey) have realized an AR experience "designed to make feel the disaster scenes for educational purposes. [It] includes three scenarios regarding the evacuation in the event of an earthquake, fire, and chemical attack. The general task of this software is to provide appropriate views based on the underlying

scenario and use artifacts by matching measured values from a mobile platform with the corresponding actions." [17]. Interesting to notice that they created an indoor GPS system to steer the user towards the target based on the user's coordinates in the world. Billboards and signs with names are placed in the 3D environment to bring more spatial awareness to the users and help them in finding the current mark.

2.4.2 AR system for maintenance of mountain bike brakes

Jorge Martin-Gutierrez from the Universidad de La Laguna (Spain) developed an Augmented Reality (AR) system aims at "making this technology approachable for applications and tasks quite common to everybody, so two prototypes have been developed : one smartphones-based mobile AR system and another video see-through based AR system which will guide the user step by step through installation of the V-brakes system, change of cartridge shoes and adjustment of tension cable" [18]. The system was developed in two prototypes: a smartphone-based AR application which allows the user to perform tasks using their device as an AR viewer and a head-mounted display (HMD) option to offer an immersive experience. Both versions incorporate 3D markers that are overlaid onto the real world environment [18], highlighting relevant components of the bike like brake parts, and guiding users step-by-step through the process. This overlay guides users' attention to key parts of the task, making it a more efficient learning experience. This structure allows users, even those without prior experience, to successfully perform maintenance tasks.

2.5 Literature analysis

After a deep analysis of numerous works in the field of AR training, I gathered the most used UX components for spatial awareness and attention focus in the graph in Figure 2.1. As it can be inferred the 18% of experiences used an approach in which the explanations of the tasks is obtained thorough textual descriptions, most of the case this

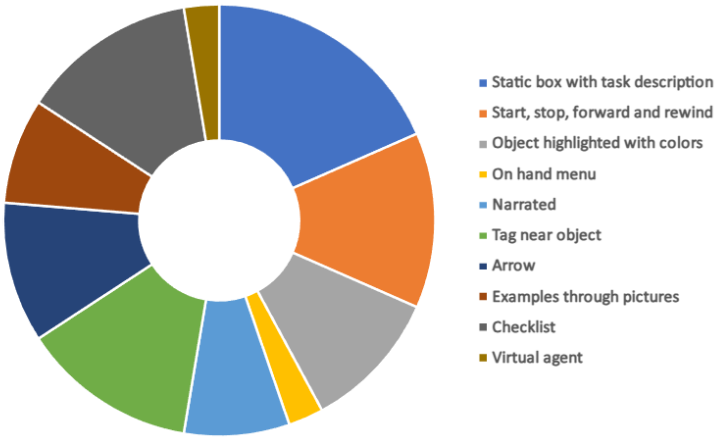

Figure 2.1. Main UX components

was helped by a checklist that recorded the completion of the tasks. The 8% of applications used images or videos to show how to complete the assignment but, as discussed in Section 1.2.2 this could lead to cognitive overload if the multimedia is too far from the object of focus or if it is too complex; in each of this cases I could not find the reason behind this choice and I could not infer if the creators studied the possibility of cognitive overload. An alternative to images as an example, as studied in the literature, is the usage of virtual agents represented by a 3D full-body avatar or by a simple 3D animation that reproduces the exact movements to be executed. Despite this, in my research this was used just 3% of the times, making it the least used solution.
Having examined these results, in this thesis I decided to implement the arrow method for spatial awareness, which was used the 11% of the times. In addition, I chose to use narrated explanations of the tasks, as this method distracts users less from the core focus and reduces the cognitive load required to read the text. Finally, I opted for the halo method to highlight the exact target object, replacing the common usage of colors to obtain the same result.

2.5.1 Gaps and open problems

Given that the primary ambition of these application is to teach how to get to a certain objective, it is crucial to monitor potential distractions. If the users become distracted, even the most well-designed application may fail to deliver the intended learning outcomes. In this analysis I came across a possible solution to this problem which is giving the capacity of pausing, replaying or navigating back to a certain point in the explanation. This feature enable learners to revisit any details they might have missed or listen to the same explanation more than once for better comprehension. Even though this could be an effective solution, I believe that integrating this methodology in a more automated and seamless way could be more straightforward and efficient. In particular, in the course of an on-going experience, the presence of buttons to advance, pause or rewind the explanation may induce distractions, which is exactly what we aim to avoid. A potential way to resolve this could be through the implementation of a narrated voice that immediately pauses the explanation when the users are not paying attention. This approach would help maintain focus without the need for manual interaction, further reducing cognitive load.

Chapter 3

Methodology and material

3.1 Research design

The purpose of this thesis is to evaluate whether, and in what extent, the application of algorithms that assess users' attention and the use of techniques that enhance spatial awareness can improve training efficiency in XR applications that use procedural learning. This project is followed by a qualitative assessment aimed at determining whether an application that does not use the library implemented performs similarly to one that does. The experiment is set up in a 3D scene created with Unity 3D, consisting of three rooms, each with a different set of tasks; users will face one task at a time in a predefined order and cannot move to another room until they have completed all the tasks in the current room. After having completed the experience, participants were asked to fill out a subjective questionnaire.

3.2 What is a Game Engine and what is Unity

3.2.1 What is a Game Engine

An important concept to grasp for the correct comprehension of this works is what is a game engine. "A gaming engine is a software development environment, also referred to as a "game architecture" or "game framework", with settings and configurations that optimize and simplify the development of video games across a variety of programming languages. A gaming engine may include a 2D or 3D graphics rendering engine that's compatible with different import formats, a physics engine that simulates real-world activities, artificial intelligence (AI) that automatically responds to the player's actions, a sound engine that controls sound effects, an animation engine, and a host of other feature" [19]. When building an application, the first decision that has to be made is whether a third-party game engine can satisfy the scope of the project or if we need to make one from scratch using, for example, programming languages like ARM. Taking into account that "third-party engines are designed to support as many different genres and playstyles as possible" [20], usually they are the preferred choice. At this time, the three most used game engines are Unity, Unreal Engine and Godot. In Table 3.2.1 some brief descriptions.

3.2.2 Unity Engine

As mentioned in Section 3.2.1 Unity is a cross-platform 2D and 3D game engine, developed by Unity Technologies since 2005. Its creators' aim was to design a tool accessible to developers of all skill levels, indeed one of its peculiarities is that it is well-suited for first-time game programmers due to its easy learning curve. A contributing factor to this is the educational support available through their platform *Unity Learn* where one could find tutorials for each proficiency level, courses, sample projects and also live sessions where experts in the field illustrate

	Programming Language	Learning Curve	Graphics and Performance	Target Platform
Unity	C#	Easy to medium	Good graphics, medium to high performance	Desktop, Mobile, Extended Reality (XR), Consoles, WebGL
Unreal Engine	C++, Blueprints	Very steep	High-fidelity graphics, very high performance	Desktop, Consoles, Mobile, XR, SteamDeck
Godot	C++, C#, GDScript	Easy to medium	Basic to medium graphics, Best performance for 2D games	Desktop, Mobile, Consoles, WebAssembly

Table 3.1. Current main third-party game engines

Figure 3.1. Unity Engine Symbol

how to navigate real-world projects and answer questions. Additionally, another factor that contributed to its notoriety is that Unity has an active community of developers who share their solutions to common problems and help each other. Another helpful tool they offer is their asset store, where one could find pre-made assets such as scripts, visual effects, AI integration, level design templates and much more.

Render Pipelines

Let's start by defining what is the definition of rendering: "Rendering is the process involved in the generation of a two-dimensional or three-dimensional image from a model by means of application programs" [21]. To do so, the render pipeline performs at each frame a series of operations which take the contents of a scene and shows them on the screen:

- **Culling** : the pipeline decides which elements are visible in the scene, so it removes objects that are outside of the camera frustum or that are hidden from other objects.
- **Rendering** : it draws the objects through pixels.
- **Post-Processing** : "the pipeline modifies the pixel buffers to generate the final output frame for the display. Example of modifications include color grading, bloom, and depth of field" [22].

In Unity, there are three types of render pipelines :

- **Built-In Render Pipeline** : this is the standard and predefined general purpose pipeline in Unity. It has limited flexibility and performance, but can be used both in 2D and 3D projects.
- **Universal Render Pipeline (URP)** : The URP can be used in 2D and 3D projects; it is easier to extend than the Built-In Render Pipeline and it is also more performing than the latter. This pipeline includes VFX graph, which can be used to design visual effects and particle behavior. It is thought for realistic and stylized lighting.
- **High Definition Render Pipeline (HDRP)** : it is the most heavy and performing pipeline. It is designed for 3D projects that require photorealism and high-quality rendering. Due to its complexity it can be used for high-end platforms. As the URP, HDRP also contains the VFX graph.

[23] As a result of its substantial demands, the High Definition Render Pipeline (HDRP) is not used for Virtual Reality or Extended Reality

applications, which require a lower graphical complexity to accommodate the hardware limitations of headsets. For this reason, this project employs the Universal Render Pipeline (URP), which is better suited for achieving optimal performance in its VR environment.

Shaders

"A shader is a piece of code that is executed on the Graphics Processing Unit (GPU), usually found on a graphics card, to manipulate an image before it is drawn to the screen" [24]. There are three main types of shaders:

- **Pixel shaders** - also known as *fragment shaders*, calculate the color and various properties of each individual pixel. They "take input data such as color, texture, lighting, and other properties and applies complex mathematical operations to calculate the final color and behavior of each pixel on the screen." [25] Pixel shaders are written using languages such as High-Level Shading Language (HLSL) and openGL Shading Language (GLSL).

- **Vertex shaders** - While *pixel shader* apply complex operations to calculate the final appearance of pixels on the screen, vertex shaders work directly on the vertices of the 3D objects in the environment. Vertex shaders can perform various operations: from modifying color, position, normal vector to creating new vertices from scratch. Its job is to take the vertices of the original 3D object from its coordinate system and transform it into the coordinate system of the camera.

- **Geometry shaders** - *Geometry shaders* take as input a set of vertices and transform them into a geometrical figure. They can generate more vertices and modify existing primitives. For example, they can be used to create or modify particle systems or convert a solid geometry into a wireframe rendered object.

These shaders are part of the graphic pipeline, which "is a conceptual model that describes what steps a graphics system needs to perform to

render a 3D scene to a 2D screen. " [26] The process begins with the vertex shader, followed by the geometry shader, which processes the output from the vertex shader, and finally, the fragment (or pixel) shader computes the final appearance of each pixel. In Unity, other then the previously mentioned shaders, there are *compute shaders* which "perform calculations on the GPU, outside of the regular graphics pipeline" and *ray tracing shaders*, which "perform calculations related to ray tracing" [27]. Unity allows users to use built-in shaders, such as those in the pipelines specifies in Section 3.2.2, and to create personalized shaders using ShaderLab, which is a specific language for writing shaders, or Shader Graph, a tool used for creating shaders without writing any code. [27]

3.2.3 GameObjects

GameObjects are the fundamental entity in Unity, they can represent characters, lights an so on. They act as containers for *Components*, which define the GameObject's functionality. For example, if a gameObject contains a light component, it will behave as a light source. The main components of a GameObject are the following:

- **Transforms** - store the position, rotation and scale of a GameObject. It can also give information about parent and children relationships.

- **Quaternion and euler rotations** - Unity uses both quaternions and Euler angles to represent rotations and orientations. Typically objects are rotated considering the rotation as an Euler angle, but Unity stores it as a quaternion, "which can be useful for more complex motions that might otherwise lead to gimbal lock" [28], which is "*The loss of one degree of freedom in a three-dimensional space that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space.*" [29]

What specifies the shape of the GameObject are the *mesh filter* and *mesh renderer*: the mesh filter holds a reference to a mesh, then the

mesh renderer displays it on screen. Alongside these components, *colliders*- which can have different shapes depending on the object being represented- give the object physical presence, preventing other objects with a RigidBody component from passing through it. The *RigidBody* "provides a physics-based way to control the movement and position of a GameObject" [30]. Important to mention are **Prefabs**. Prefabs in Unity are reusable GameObjects that contain components, properties, and children of a specified GameObject which act as a singular asset within the project. Changes made to a prefab will automatically apply to all instances of that prefab, making it a very useful tool for managing repeated elements in a project. For example, Figure 3.2 shows the Manager prefab, composed in turn by the ArrowParent prefab and CircleTargetPrefab.
GameObjects may be separated in different **layers** based on various reasons, such as behaviors or purposes. These may be useful when given the possibility to decide which layer to render, when trying to detect a collision or when using raycasting, which will be explained further in Section 3.4.1.

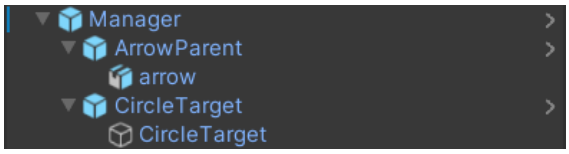

Figure 3.2. Manager Prefab

3.2.4 System Requirements for Unity Editor

In order to ensure optimal performance, the system requirements listed in Table 3.2.4 must be met. [31]

Minimum Requirements	Windows	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	Mojave 10.14+ (Intel editor) Big Sur 11.0 (Apple silicon Editor)	Ubuntu 20.04 and Ubuntu 18.04.
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor.	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)

Table 3.2. System requirements for Unity Editor version 2022.3

3.3 Project structure

This project has been realized using Unity 2022.3.11f1 and a Meta Quest 2.

The goal of this thesis is to create an environment in which users' learning curves are analyzed based on the presence or absence of a user attention-checking algorithm and its associated methods. In particular, this algorithm is integrated with methods such as an arrow indicating the direction of the target object and an halo pointing out the exact focus entity; additionally, a system that stops the execution of the vocal explanation whenever users lose focus is implemented. The experience spans across three rooms:

- The first room users encounter is the first level environment, where a fire training exercise is simulated. Users must recognize and locate a fire, turn off the fire alarm, and locate the fire extinguisher to put out the fire.
- Upon completing the first task, a door to the second level room opens. Here, users must wright several objects on a scale and then sort them into the corresponding boxes based on their weight.
- The third room simulates a grocery store environment, where users must identify items that are incorrectly placed on shelves and move them to the correct locations. Afterwards, users are asked to check whether certain objects are listed in the store's database and to take appropriate actions based on the results.

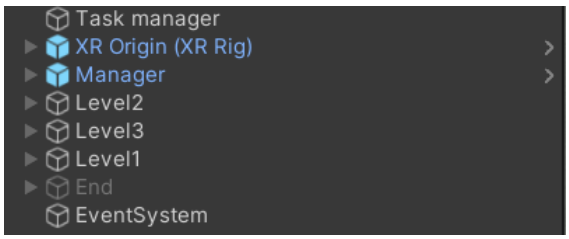

Figure 3.3. Scene Hierarchy

The tasks were deliberately made easy to understand and execute in order to avoid overstimulating the user. In fact, an excessive amount of information could have distracted the user from the goal, altering the outcome and making the sample inconsistent. Furthermore, the simplification allowed for a clear assessment of the method's effectiveness, as any loss of attention could be attributed solely to the user and not to the difficulty of the task.

3.3.1 Packages

The most important package used in this work is the **XR Interaction Toolkit**, which is "a high-level, component-based, interaction system for creating VR and AR experiences" [32]. It was chosen because of its wide range of compatible devices, which include Augmented Reality (AR) and Virtual Reality (VR) headsets and controllers. The package version used is the 3.0.3 which is compatible with Unity editor versions from 2021.3 and up. To function properly, the XR Interaction Toolkit has several dependencies that are automatically installed in the project, including the Input System, Mathematics, Unity UI and XR Core Utilities. If the project is intended for an AR device, then the AR Foundation must also be installed. The backbone of the XR Interaction Toolkit is the XR Rig, which contains fundamental elements such as the camera and the controllers Input Action Manager - used for binding controller inputs to actions.

Figure 3.4. XR Rig

Other used packages include *ProBuilder*, which can be used to build,

edit, and texture custom geometry, and *TextMeshPro*, which provides more advanced text formatting options compared to the deafult Unity method.

3.3.2 Prefabs and shaders

After discussing shaders and prefabs in Unity in Sections 3.2.2 and 3.2.3, their usage in this project will be examined. Significant to remark is the Manager prefab alredy shown in Figure 3.2. In particular, this prefab contains in turn two others, which are the *ArrowParent* and the *CircleTarget* prefabs. ArrowParent holds the *ArrowPointing* component, which is the script responsible for the arrow's rotating behavior, and it is the parent of the *arrow* GameObject. The arrow is particularly important as it contains the mesh representing the arrow that indicates the direction of the target object, shown in Figure 3.5. Given its importance to the purpose of this project, users must always be able to see it. Therefore a custom shader was developed to render the arrow after all other elements, allowing it to always appear on top of everything. The *CircleTarget* prefab contains a canvas component

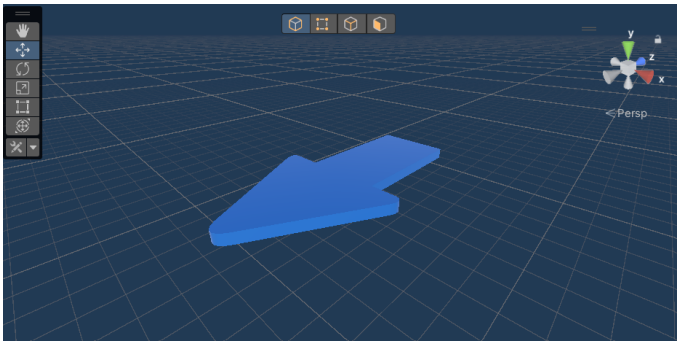

Figure 3.5. ArrowParent prefab in isolation

which, in turn, holds a child Image GameObject representing the halo that overlays the target object (Figure 3.6). Additionally, two scripts, CircleRotation and CircleTarget, are bound to the Image GameObject, responsible respectively for the halo's rotation and for its position on the scene. The circle, like the arrow mentioned previously, must always remain in front and not be obstructed by other objects in the scene. In

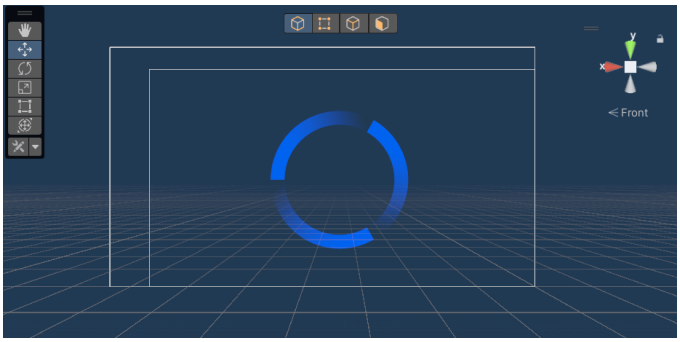

Figure 3.6. CircleTarget prefab in isolation

this case, a **UICamera** has been created. As shown in Figure 3.7, this camera only renders the objects in the *UI layer*, such as the CircleTarget prefab. It is an overlay type camera, meaning that its render will appear on top of the MainCamera's render. The UICamera prefab is a child of the MainCamera GameObject, which is itself a child of the XR Rig.

3.4 Code architecture

As discussed in Section 1.3, the existing methods for checking user attention rely on external objects, such as cameras that track the pupils movement. In this research, I considered that not everyone has access to external equipment, which may be expensive, unavailable for specific devices, or not sufficiently reliable.

3.4.1 User Attention and Spatial Awareness

Checking user attention

To check user, I began by considering that the human eye's binocular field of view (FOV), which is "the open, observable area a person can see through their eyes or via an optical device, such as a camera" [33], is approximately 120°. Before explaining the reasoning behind the designed method, the concepts of **raycast** and **spherecast** must be described: a

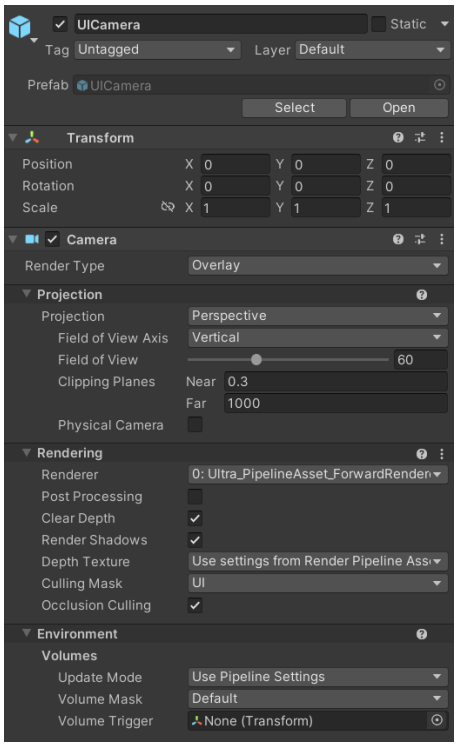

Figure 3.7. UICamera components

raycast is essentially a ray originating from a specific point and extending in a certain direction up to a maximum distance. A layermask is used to specify whether only certain layers should be considered when detecting hit. Spherecast is an extension of raycast, where instead of a single ray, a series of spheres are cast along the path. The latter is useful for our purpose because the complexity of the human vision cannot be reduced to a single point, but must account for peripheral vision. Since the FOV can be considered as a cone, the developed method for checking user attention utilize the existing SphereCastAll() function. This function first creates a spherical raycast which travels in the direction of the camera, then checks if the hit objects fall within the angle specified in the function. A limitation of the Spherecast method is that it does not detect colliders when the sphere overlaps with the collider. Specifically, objects entirely inside the initial sphere are not detected. To address this blind spot, an additional RaycastAll method was used, configured to detect only objects on the appropriate layer.

In this setup, objects that are very close to the origin are detected by twenty RaycastNonAlloc (ten in an horizontal line and other then in a vertical line), spanning angles of sixty degrees. The raycastNonAlloc are a form of raycast used for optimizing the creation of these rays without generating any garbage.
The **coneCast** method takes the following inputs: a *Physics* class, a *Vector3* indicating the origin point, a *float* for the maximum radius of the sphere, a *Vector3* which indicates the direction of the cast, a *float* to indicate the maximum reachable distance, a *float* specifying the cone angle and a *LayerMask* value indicating the layer to check. It returns an array enclosing all the objects hit by the conecast.

```
1  RaycastHit[] sphereCastHits = Physics.SphereCastAll(
     origin, maxRadius, direction, maxDistance, layer);
2  List<RaycastHit> coneCastHitList = new List<RaycastHit>()
     ;
3  if (sphereCastHits.Length > 0)
4  {
5      for (int i = 0; i < sphereCastHits.Length; i++)
6      {
7
8          Vector3 hitPoint = sphereCastHits[i].point;
9          Vector3 dirToHit = hitPoint - origin;
10         float angleToHit = Vector3.Angle(direction,
     dirToHit);
11         if ((angleToHit < coneAngle && IsPartiallyVisible
     (sphereCastHits[i], origin)) || (IsPartiallyVisible(
     sphereCastHits[i], origin) && isNear(origin, hitPoint,
     maxDistance)))
12         {
13             coneCastHitList.Add(sphereCastHits[i]);
14         }
15     }
16 }
17
18 float horizontalStep = 60 /*degrees angle*/ / (10 /*rays
     in horizontal*/ - 1);
19 float verticalStep = 60 / (10 - 1);
20
21 float halfHorizontalRange = 60 / 2f;
22 float halfVerticalRange = 60 / 2f;
```

```
23  float rayDistance = maxRadius * 2 + Vector3.Distance(
      Camera.main.transform.position, origin) + 2;
24
25  RaycastHit[] hits = new RaycastHit[20];
26  for (int v = 0; v < 10; v++)
27  {
28      float verticalAngle = -halfVerticalRange + (v *
      verticalStep);
29
30      for (int h = 0; h < 10; h++)
31      {
32          float horizontalAngle = -halfHorizontalRange + (h
       * horizontalStep);
33
34          Vector3 rayDirection = Quaternion.Euler(
      verticalAngle, horizontalAngle, 0) * direction;
35          int hitCount = Physics.RaycastNonAlloc(Camera.
      main.transform.position, rayDirection, hits,
      rayDistance, layer);
36
37          for (int i = 0; i < hitCount; i++)
38          {
39              coneCastHitList.Add(hits[i]);
40          }
41
42      }
43  }
44
45
46  RaycastHit[] coneCastHits = coneCastHitList.ToArray();
47  return coneCastHits;
48  }
```

Listing 3.1. coneCast method

As shown in Listing 3.1, a custom function was added within the if statement. The **IsPartiallyVisible** function, takes a *RaycastHit* and a *Vector3* value as parameters and returns a boolean value. This method provides an additional check on the visibility of the hit object in the scene: if the objects is obstructed by a larger item which is located in front of it from the camera's perspective, then the function returns *false*, otherwise it returns *true*. To achieve this, it considers the boundaries of the hit object and, using a raycast, it checks if the ray hits another

object before reaching any point on the boundary of the target object. If all the points considered are not reachable, the function returns false, and so the object is not completely visible, otherwise it returns true.

```
            int counterPointsObstructed = 0;
        Vector3[] pointsToCheck = new Vector3[]
        {
            hit.collider.bounds.center,
    // Center
            hit.collider.bounds.min,
    // Bottom-left corner
            hit.collider.bounds.max,
    // Top-right corner
            new Vector3(hit.collider.bounds.min.x, hit.
collider.bounds.min.y, hit.collider.bounds.max.z), //
Other corners
            new Vector3(hit.collider.bounds.min.x, hit.
collider.bounds.max.y, hit.collider.bounds.min.z),
            new Vector3(hit.collider.bounds.max.x, hit.
collider.bounds.min.y, hit.collider.bounds.min.z),
            new Vector3(hit.collider.bounds.max.x, hit.
collider.bounds.max.y, hit.collider.bounds.min.z),
            new Vector3(hit.collider.bounds.max.x, hit.
collider.bounds.max.y, hit.collider.bounds.max.z)
        };

        int usefulLayerMask = LayerMask.GetMask("Useful");
        foreach (Vector3 point in pointsToCheck)
        {
            Vector3 directionToPoint = point - origin;
            RaycastHit obstructionHit;

            if(Physics.Raycast(origin, directionToPoint,
out obstructionHit, directionToPoint.magnitude))
            {
                if (obstructionHit.collider.gameObject !=
hit.collider.gameObject && obstructionHit.transform.
gameObject.layer != usefulLayerMask)
                {
                    counterPointsObstructed++;
                }
            }
        }
```

```
28          if (counterPointsObstructed == 8)
29              return false;
30          else
31              return true;
32      }
```

Listing 3.2. IsPartiallyVisible() method

The *coneCast* method is then used in the **RaycastManager** script, which sets a variable _hit to true if the function finds any elements that match the criteria, and false otherwise. Additionally, it stores the objects just hit by the conecast in the list hitObjects.

Spatial Awareness

Once the method for checking user attention is made, a methodology is needed to lead users towards the target if they lose focus. As shown in Figures 3.5 and 3.6, an arrow and an halo were designed as guiding mechanism for users.
Before describing how their behavior is implemented, another important script must be discussed: the **ChangeTarget** script. This script includes a public array of GameObjects called *targets*, which is populated by dragging and dropping the target elements, which are the objects where users need to focus their attention during the experience. For this purpose, a new **TargetPlaceHolder** empty script was created. The sole purpose of this script is to check and find the target objects within the scene.

```
1       public void changeTarget()
2       {
3           if (i < targets.Length)
4           {
5               targets[i].layer = 0;
6               TargetPlaceholder targetPlaceholder = targets[
    i].GetComponent<TargetPlaceholder>();
7
8               // Check if the TargetPlaceholder component
    exists, then destroy it
9               if (targetPlaceholder != null)
10              {
```

```
11                  Destroy(targetPlaceholder);
12              }
13              SpatialAwarenessManager.NeverLooked = false;
14              if(i !=  targets.Length - 1)
15                  i++;
16          }
17      }
```

Listing 3.3. ChangeTarget() method

As displayed in Listing 3.3, when the changeTarget() method is called, the layer of the current target object is set to *Default* and the TargetPlaceHolder component is removed from its GameObject. The next target is set in the Update() method, as demonstrated in Listing 3.4. Here, the layer of the new target object becomes *Useful*, and the TargetPlaceHolder component is added to its GameObject and in each of its children, if any.

```
1           if (i < targets.Length)
2           {
3               if (targets[i].layer != LayerMask.NameToLayer(
    "Useful"))
4               {
5                   targets[i].layer = LayerMask.NameToLayer("
    Useful");
6                   if (targets[i].transform.childCount > 0)
7                   {
8                       for (int j = 0; j < targets[i].
    transform.childCount; j++)
9                       {
10                          targets[i].transform.GetChild(j).
    gameObject.layer = LayerMask.NameToLayer("Useful");
11                      }
12                  }
13                  targets[i].AddComponent(typeof(
    TargetPlaceholder));
14              }
15          }
```

Listing 3.4. Update() method in the ChangeTarget script

The arrow's behavior is implemented in the *ArrowPointing* script in Listing 3.5. Here, if a GameObject with the TargetPlaceHolder is

found, then that it is designated as the target, and the arrow is rotated towards it using Quaternions.

```
        float singleStep = speedRotation * Time.deltaTime;
        Vector3 position;
        TargetPlaceholder placeholder = GameObject.
FindObjectOfType<TargetPlaceholder>();
        if (placeholder == null) return;
        target = placeholder.transform;
        position = target.position;
        arrow.gameObject.SetActive(true);
        Vector3 targetDirection = target.position - arrow.
position;
        Vector3 newDirection = Vector3.RotateTowards(arrow
.forward, targetDirection, singleStep, 0.0f);
        arrow.rotation = Quaternion.LookRotation(
newDirection);
```

Listing 3.5. Update() method in the ArrowPointing script

While the arrow gives the direction of the target object, the halo indicates its exact position in the environment. To obtain this, the method first checks if an object has been hit using the *getHitObject()* function in the RaycastManager script, then it sets the position of the image representing the circle to the position of the found target. It uses the boundaries of the target object on the screen to determine the size of the circle: the biggest of the x and y dimensione is used to set the circle's size, ensuring the entire object can fit inside it. Using this method, the size of the circle varies according to the distance from the object, because closer objects appear larger on screen and vice versa.

```
        if (raycastManager.getHitObject() != null)
        {
            target = raycastManager.getHitObject().
transform;
            if (target != null)
            {
                imageRect.position = target.position;
                Bounds objectBounds = target.GetComponent<
Renderer>().bounds;
                Vector3 objectSize = objectBounds.size;
```

```
9                 Vector3 objectScreenMin = _camera.
  WorldToScreenPoint(objectBounds.min);
10                Vector3 objectScreenMax = _camera.
  WorldToScreenPoint(objectBounds.max);
11                Vector2 objectScreenSize = objectScreenMax
   - objectScreenMin;
12                if (objectScreenSize.x > objectScreenSize.
  y)
13                    objectScreenSize.y = objectScreenSize.
  x;
14                else
15                    objectScreenSize.x = objectScreenSize.
  y;
16                imageRect.sizeDelta = new Vector2(
  objectScreenSize.x * sizeMultiplier.x, objectScreenSize
  .y * sizeMultiplier.y);
17
18                image.enabled = true;
```

Listing 3.6. Update() method in the circleTarget script

The **circleRotation** script then configures the rotation of the circle to make it more dynamic. The **SpatialAwarenessManager** script links the functionalities of the arrow and halo methods, coordinating the conditions under which each of these methods is activated or deactivated.

3.4.2 Audio tutorial

To minimize the risk of cognitive overload from written instructions, as discussed in the previous chapter, all tasks in this project are delivered through voice instructions. The previously discussed methods are used not only for setting the arrow and halo indicators but also to manage the interruption of vocal explanations. Specifically, if users are not looking at the designated point of focus, the explanation pauses and resumes once they regain focus. This behavior is implemented in the *CheckAttention()* method within the *SpeechManager* script. The script interacts with the RaycastManager to access the *_hit* variable. If this variable is false and users fail to regain focus within three seconds, the explanation halts. It will restart as soon as attention is regained and the variable is set to true. The implementation of this method is shown in Listing 3.7. A critical aspect was determining whether the audio

had been paused by the method or had finished playing on its own. To address this, a list of integer values was utilized. The list's length matched the array of *AudioClips* used in the project, with each value representing the playback status of a corresponding clip. If an audio clip completed (i.e., all its seconds had elapsed), the associated value in the list was set to 1. By checking these values, the program could accurately determine whether a specific audio clip had ended.

```
1         public void CheckAttention()
2 {
3          if (!RaycastManager._hit)
4          {
5              wasDistracted = true;
6              timeOutOfFocus += Time.deltaTime;
7              if (source.isPlaying && timeOutOfFocus >=
    delayBeforeStoppingAudio)
8              {
9                  source.Pause();
10             }
11
12         }
13         else
14         {
15             if (!ended)
16             {
17                 timeOutOfFocus = 0;
18                 if (!source.isPlaying && wasDistracted)
19                 {
20                     if (source.clip == audioClips[index])
21                     {
22                         playAudio();
23                         ended = false;
24                     }
25                     else
26                     {
27                         source.clip = audioClips[index];
28                         playAudio();
29                         ended = false;
30                     }
31                 }
32                 wasDistracted = false;
```

```
33                  }
34              }
35  }
```

Listing 3.7. CheckAttention() method in the SpeechManager script

3.4.3 User attention manager

The **Manager** prefab was created to coordinate the simultaneous operations of both user attention-checking and spatial awareness methods. It brings together the *RaycastManager* component for attention tracking with the SpatialAwarenessManager and ChangeTarget script, ensuring they work in a synchronized way and without conflict.
As shown in Figure 3.8, each of these scripts uses a variable from another script in the manager: ChangeTarget takes the NeverLooked boolean value from the SpatialAwarenessManager, which gets the boolean value _hit from the RaycastManager.

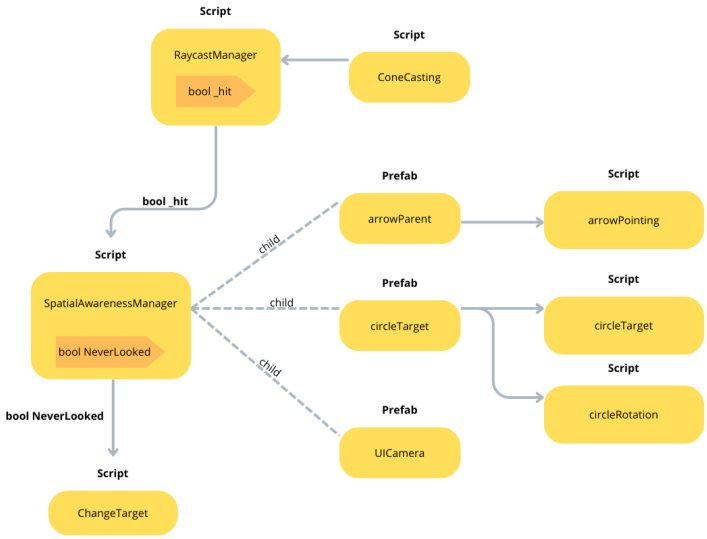

Figure 3.8. Components of the Manager prefab

3.4.4 Levels and tasks implementation

The creation of the **TaskManagers** namespace played a key role in managing the level and taks. Within this namespace, the public class **Task** has been created, which contains the task's name, description, status which is an enumeration made by *NotStarted, OnGoing* and *Completed* values. In line 10 of Listing 3.8 a reference to the ChangeTarget script, defined in Listing 3.3, is instantiated. This variable is used to change the focus target as soon as a task is completed. Upon creation of a Task, its status is initially set to *NotStarted*. Then, methods such as *completeTask()* or *StartTask()* allow other scripts to change the task's status. The **Level** class contains a list of Tasks, along with the level number and its status. When instancing a Level, a new empty list of Tasks is created and the status, as in the Task class, is set to *NotStarted*.

```
1  namespace TaskManagers
2  {
3      public enum Status { NotStarted, OnGoing, Completed };
4
5      public class Task
6      {
7          private string taskName;
8          private string taskDescription;
9          private Status status;
10         private ChangeTarget changeTarget = GameObject.
       FindObjectOfType<ChangeTarget>();
11
12         public Task(string taskName, string
       taskDescription)
13         {
14             this.taskName = taskName;
15             this.taskDescription = taskDescription;
16             this.status = Status.NotStarted;
17         }
18
19         public void completeTask()
20         {
21             this.status = Status.Completed;
22             changeTarget.changeTarget();
23
```

```
24          }
25
26          public Status GetStatus()
27          {
28              return this.status;
29          }
30
31          public void StartTask()
32          {
33              if (this.status == Status.NotStarted) this.
    status = Status.OnGoing;
34          }
35      }
36
37      public class Level
38      {
39          private List<Task> tasks;
40          private int levelNumber;
41          private Status status;
42
43          public Level(int levelNumber)
44          {
45              this.tasks = new List<Task>();
46              this.levelNumber = levelNumber;
47              this.status = Status.NotStarted;
48          }
49
50          public void EndLevel()
51          {
52              status = Status.Completed;
53              Debug.Log("LevelFinished");
54          }
55
56          public void StartLevel()
57          {
58              status = Status.OnGoing;
59          }
60
61          public void AddTask(Task task)
62          {
63              tasks.Add(task);
64          }
65
```

```
66          public List<Task> GetTasks() { return this.tasks;
    }
67      }
```

Listing 3.8. TaskManagers namespace

The script that really coordinates tasks is the **TaskManager**. Here tasks, along with the levels, are defined. As shown in Figure 3.9, each level is implemented in its own self-named script: **LevelOne, LevelTwo, LevelThree**. In the *Start()* method of the *TaskManager* script, the levels and tasks are defined. To manage the various levels, **Actions** have been used. To understand *actions* it is helpful to first grasp the concepts of *polling* and *interrupts*, which usually refer to communication mechanisms between CPU and peripheral devices. An interrupt is an hardware mechanism that notifies the CPU when the device requires attention, whereas polling consists in a protocol in which the CPU continuously checks whether a device needs attention. Similarly, checking in every frame if one of the levels has finished (using an if statement in Update(), for example) would compare to polling. On the other hand, actions relates to interrupts. When using Actions, functions can subscribe to a certain action, meaning that when the action is invoked using *action.Invoke()*, the subscribed functions are notified and execute the predisposed function. In this way, when the task loading for level one ends, the LevelOne script starts. Correspondingly, LevelTwo and LevelThree Update() methods begin execution once the relative previous level has finished. // add things about audio stopping and starting

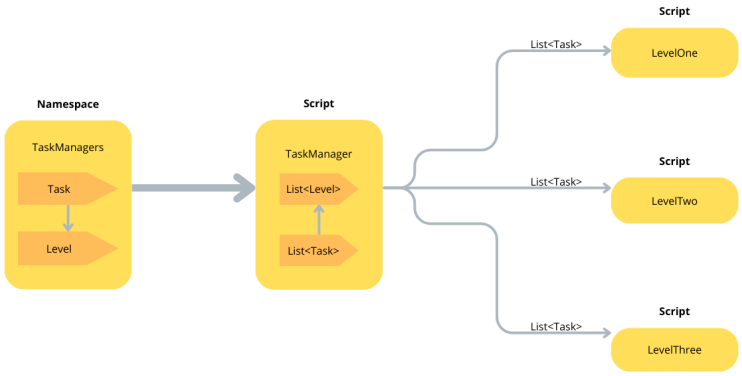

Figure 3.9. TaskManager prefab and TaskManagers namespace

Chapter 4

Testing

The objective of the testing phase of this project is to evaluate whether the proposed methodology designed to assess user attention and enhance spatial awareness improves learning outcomes in procedural learning-based XR applications. To estimate this, participants were divided into two distinct groups, each experiencing different versions of the application: one where audio explanations pause when users lose focus, from now on referred to as *guided version*, and another where the audio continues without accounting for user attention (from here referred to as *unguided version*).

4.1 Demographics

Participants were recruited on a voluntary basis through an online form, from which twenty-four individuals have been selected and divided into two groups of twelve. As shown in Table 4.1, the sample consisted of 58.8% females and 41.7% males. The majority of partecipants (87.5%) were aged between 18 and 25 years, while 12.5% were between 26 and 35 years old. Regarding educational background, most participants held a bachelor's degree (58.3%), 29.2% had an high school diploma and 12.5% had a master's degree. An important aspect of this demographic research was the frequency with which the sample used VR applications: among them, 54.2% had never used a VR device before, 37.5% used it rarely and 8.3% used it one or more times per week. This information

is particularly relevant, as participants' prior background in VR could influence their performance during the experience.

Characteristics	Total	Guided	Unguided
Gender	14 females (58.3%), 10 males (41.7%)	8 females (66.7%), 4 males (33.3%)	6 females (50%), 6 males (50%)
Age range	18-25 87.5%, 26-25 12.5%	18-25 91.7%, 26-35 8.3%	18-25 83.3%, 26-35 16.7%
Education level	Bachelor's degree 58.3%, High school diploma 29.2%, Master's degree 12.5%	Bachelor's degree 58.3%, High school diploma 41.7%, Master's degree 0%	Bachelor's degree 58.3%, High school diploma 16.7%, Master's degree 25%

Table 4.1. Demographic characteristics

4.2 Procedure

As previously mentioned, the testing phase involved twenty-four people, who were assigned to either guided or unguided version of the application in an alternating manner. In the initial session, participants had to completed a task-based experience where an audio tutorial guided them step-by-step, while visual cues, such as an arrow and a halo, signaled the object of focus. Participants were divided into two groups: those using the guided version, where the audio tutorial would pause if the user became distracted and lost focus, and those using the unguided version, where the audio continued uninterrupted regardless of user attention. After completing the first part, all users had to repeat the experience without any help, so without arrow and circle mechanisms or audio tutorial. Each level had a maximum execution time of three minutes. If participants were not able to finish a level in this amount of time, they

were automatically relocated to the next level, or if it was the final level, the experience was concluded. The goal of this second phase was to assess whether participants who used the A version would demonstrate faster task completion times and better retention of task instructions compared to those using the unguided version. Completion times were recorded for each task and for each level, across both versions A and B, as well as during the second, unassisted experience. This comparison aimed at evaluating whether the attentiveness-checking mechanism in the guided version led to improved learning outcomes and faster execution compared to the unguided version. The third part of the testing phase involved administering a questionnaire which aimed at gathering information about demographics, XR knowledge, and user experience. Specifically, two standardized tools were used: the IPQ (**Igroup Presence Questionnaire** and the **System Usability Scale** (SUS) were used. The *IPQ* aims at measuring the sense of presence experienced in a virtual environment, in particular it analyses *spatial presence*, defined as "the sense of being physically present in the virtual environment" [34], *involvement*, which evaluates "the attention devoted to the virtual environment and the involvement experienced" [34] and *experienced realism*, which captures "the subjective experience of realism in the virtual environment" [34]. Higher scores on the IPQ test indicate a stronger sense of presence, higher involvement, and greater perceived realism. The *SUS* measures the usability of the application. According to its guidelines, a score below 51 indicates poor usability, a score around 68 suggests average usability, and a score of 80.3 or higher suggests that users find the system highly usable and easy to understand. Additionally, to assess user attentiveness, the following questions were included to gather participants' self-assessment of their ability to maintain focus during the experience:

- *Was it easy to follow directions?*
- *How often did you get distracted?*
- *Did the implemented mechanics help you in regaining attention?*
- *Did you ever feel lost or confused?*

This self-evaluation complements the objective measure of attentiveness, which was captured through task completion times. Faster task completion times may indicate a better understanding of the tasks. By combining these subjective and objective measures, the study aims at assessing how the attention-checking mechanism affects task comprehension and performance.

Chapter 5

Results

This chapter presents the results of the study already discussed in Chapter 4. First, the results of the IPQ and SUS tests are presented and analyzed. Following this, the objective test results are organized to provide an analysis of how participants performed, with a focus on task completion time, improvements from the learning to the training experience, and an analysis of how their self-evaluation reflected on their performance, considering which version of the application they used. Additionally, the influence of prior XR experience is examined. This chapter highlights patterns in user behavior, performance and attentiveness between the guided and unguided versions (already described in Section 4.2).

5.1 IPQ and SUS scores

Before evaluating how users performed during the experience, it is useful to evaluate the overall effectiveness and usability of the application. The average *Igroup Presence Questionnaire (IPQ)* showed these results:

- A 5.65 score in terms of spatial presence, which means users had a moderately high sense of spatial presence.

- A 2.98 involvement score indicates a moderate level of engagement with the virtual environment.

- A 4.64 score in the realness compartment suggests that users perceived the virtual environment as quite realistic.

The involvement score of 2.98 could have been impacted by the fact that the majority of participants had never used an XR device before, causing them to be more aware of their surroundings and reducing their engagement in the virtual environment. Moreover, this value could be also due to the easy tasks and simplified environments. In Figure 5.1 the IPQ scores based on the application version.

The *System Usability Score* showed excellent results in terms of usabil-

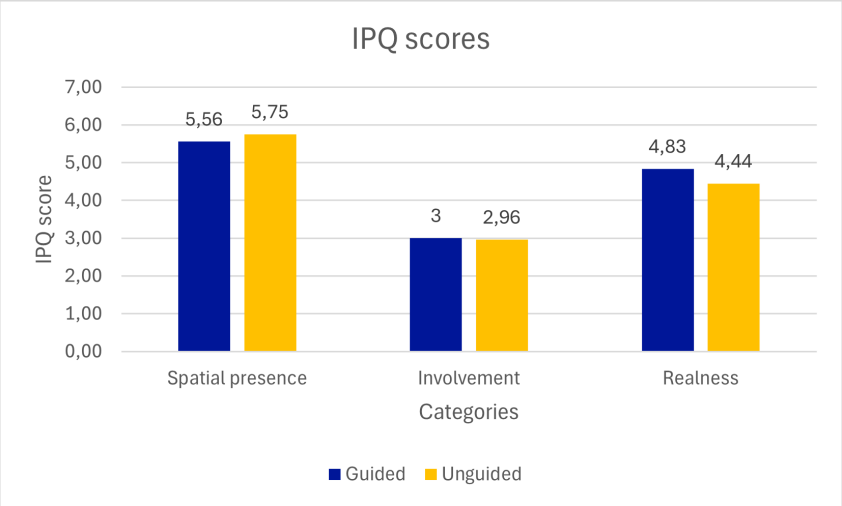

Figure 5.1. IPQ scores relative to the application version

ity (Figure 5.2): the overall score was 87.6, which indicates that users found the application easy to use and to understand. It is interesting to note that the guided version had an average score of 90 compared to the 85.2 of the unguided version. In order to assess the potential impact of the application version on the usability of the experience, a t-test for independent samples was conducted. The p-value for a two-tailed test was 0.19, which is greater than the accepted significance level of 0.05. This suggests that, even though the guided version has an higher average SUS score, there is no statistically significant difference between the guided and unguided versions, meaning that the SUS scores don't relate to the version of the application.

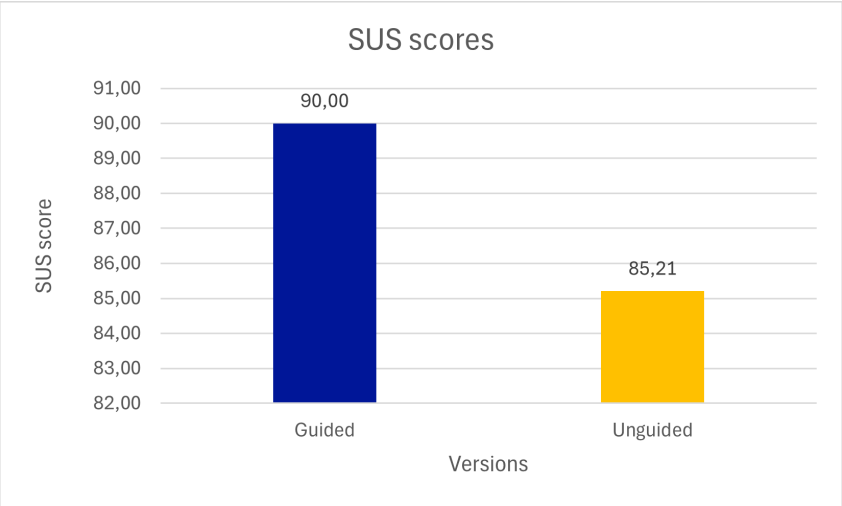

Figure 5.2. SUS scores for guided and unguided versions

5.2 Subjective attentiveness results

Part of the post-experience questionnaire included subjective questions regarding user distraction and confusion during the experience. The responses to these questions were analyzed separately for each application version, and represented in Figure 5.3:

- The question "*was it easy to follow directions?*" received an average score of 6.58 in the guided version and 6.25 for the unguided version. Although the difference is marginal, it suggests that users with the guided version felt more guided towards the objective than users with the unguided version.

- Participants, on average, responded very well to the question "*how often did you get distracted?*". Users with the guided version reported an average score of 1.17, while unguided users reported a higher average of 1.5, indicating that users with the unguided version experienced more frequent distractions.

- A substantial difference was observed in the responses to the question "*did the implemented mechanics help you in regaining attention?*": guided participants scored 5.75, while unguided participants scored 4.75.

- In the question "*did you ever feel lost or confused?*" guided participants gave on average a 1.5 score, while unguided participants scored 2.33, suggesting that users found the guided version more understandable than the unguided version.

These results suggest that the guided version provides better guidance and clarity than the unguided version, particularly in terms of helping users stay focused and regain attention. To statistically analyze these results, a T-Test was used. While the first three questions reported p-values higher than the typical significance level of 0.05 ($p1 = 0.167, p2 = 0.213, p3 = 0.214$ respectively), the question "*did you ever feel lost or confused?*" revealed a negative t-statistic and a p-value of 0.0057, which indicate that the unguided version led to higher confusion or disorientation compared to the guided version. It is also important to note that, during the testing phase, the majority of participants with the unguided version tended to get distracted during the explanation of the first task of level two, which led to an increased number of questions and requests for clarification.

5.3 Task performance results

This part of the study focuses on the objective measurements used to assess the efficiency of both application versions. Specifically, task execution times were recorded during both the first (guided) and second (unguided) parts of the experience. On average, guided participants completed the experience in 444,36 seconds, while unguided participants took 512,6 seconds. Similarly, in the second part guided participants completed the experience in 159,44 seconds, while unguided users took 230,79 seconds. The shorter completion time in the guided version could indicate that users were able to understand and navigate through

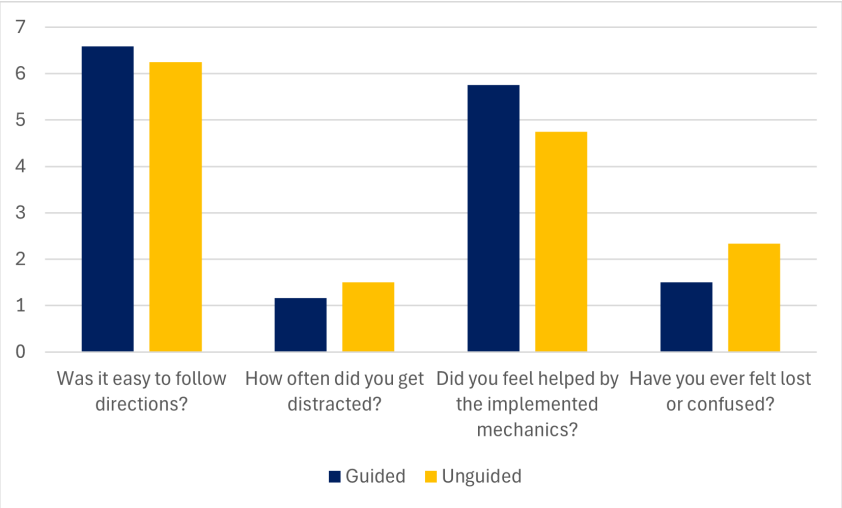

Figure 5.3. Results of subjective questions

the tasks more quickly and with less distractions, likely because the pause in the explanation allowed them to focus on the task. In contrast, the longer execution time for the unguided version could suggest that users needed more time to recall instructions or orient themselves due to lack of guiding mechanisms. Another important aspect to considerate is the standard deviation: the guided version resulted in a 30,49 seconds standard deviation, while the unguided version had a standard deviation of 49,46 seconds, also shown in Figure 5.4. The lower results for the guided version suggest that the more guided approach in this version led to more consistent task completion times. In contrast, the higher standard deviation in the unguided version indicates greater variability and a less predictable user experience.

A comparison has been made between the execution times of participants using the guided version and the unguided version across both the learning and training experiences. Participants using the guided version showed a 63.12% improvement in execution time between the two testing phases, while unguided participants improvement was about 52%, comparatively smaller than the guided version. While both versions of the application have the same tasks, the mechanisms in the guided version could have helped users achieving a deeper understanding of the tasks during the learning experience, leading them to a better

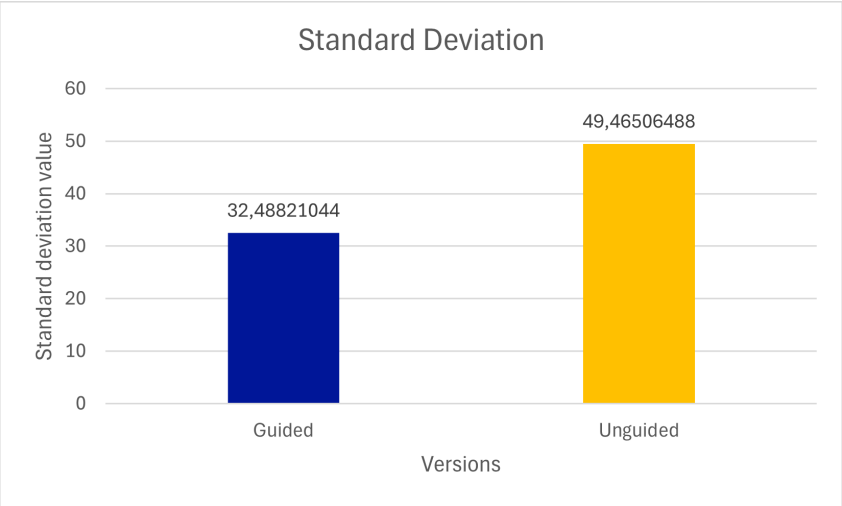

Figure 5.4. Standard deviations of the two versions of the application

performance in the second part of the testing process. To support this hypothesis, another T-Test has been executed taking into consideration the improvement percentages between the learning and training experiences in both guided and unguided. The results showed a p-value for the one-tailed test of 0.028, which is less than the typical significance level of 0.05, but a value of 0.056 for the two-tailed test which is slightly above the threshold. Considering that our goal is to prove that the guided version can lead to higher improvement percentage than the unguided version, the one-tailed test results can be taken into account. Therefore, it has been confirmed that guided participants had higher improvements between the learning and training experiences than those using the unguided version. Upon analyzing the data in the graph in Figure 5.5, one could notice that the only task where unguided users slightly outperformed those using the guided version is the second task of level two. In this task, participants were required to weight a block on a scale to determine the right container for the object. Several guided participants, recalling the block's weight from the learning phase, asked whether they needed to re-weigh the object. This hesitation led to longer execution times for these users.

The graph in Figure 5.6 illustrates the relationship between the responses to the question *Did you ever feel lost or confused?* and the

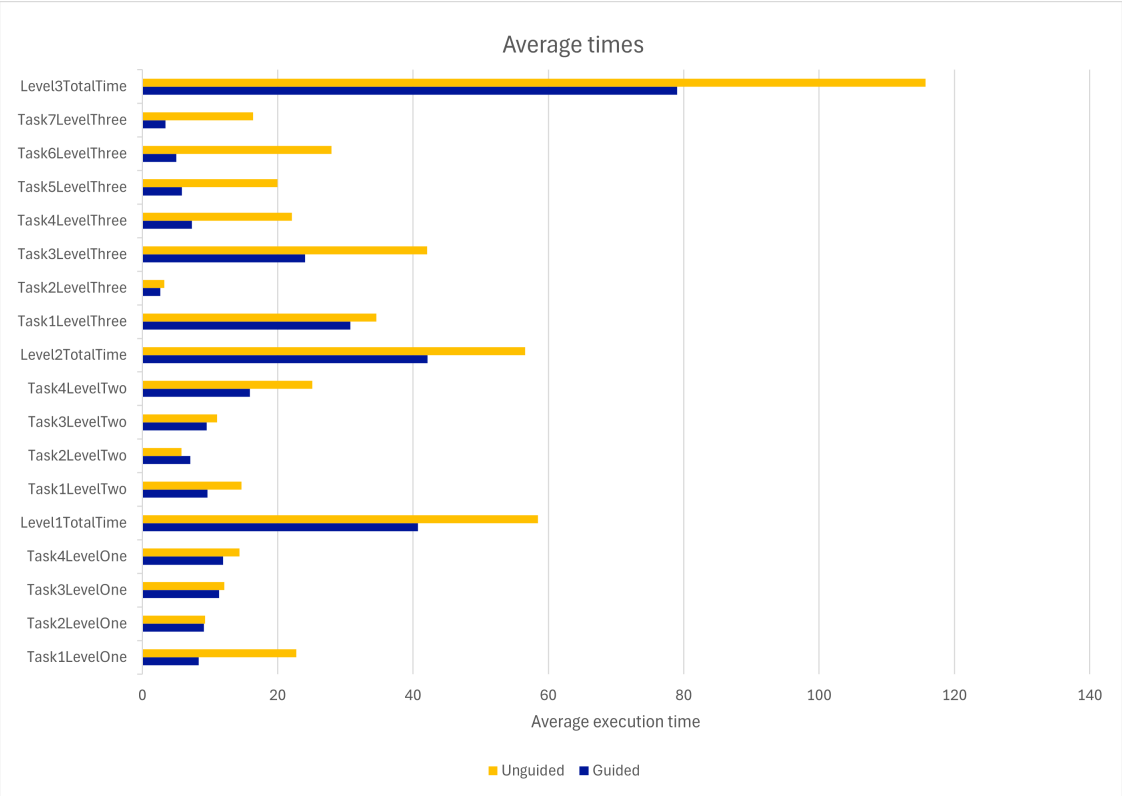

Figure 5.5. Average times for each task and version

execution times for the training experience for each participant. The results of this question have been multiplied by 100 to better align with the time scale; for instance, an answer of 1 is represented as 100 on the graph, and so on. One of the most striking features is the alternating pattern of peaks and valleys, which is related to the fact that both versions were included in this graph. Peaks represent guided participants, while the troughs indicate unguided users. By comparing the trends between execution times and feeling of confusion, it is easy to note that there is a clear pattern between the two values: when users felt more disoriented, their total execution times tended to increase accordingly. Indeed, the relationship between these two features was confirmed by a *Pearson correlation*: the analysis revealed a moderate positive correlation between the variables (r=0.5664), implying that higher values in the response of the questionnaire are associated to higher execution

times. To test whether this correlation was statistically significant, a t-test was conducted. The resulting t-statistic was t=3.22, with 22 degrees of freedom. Comparing this to the critical t-value of 2.074 and considering the p-value equal to 0.0011, we can conclude that there is a significant positive relationship between the question *Did you ever feel lost or confused?* and the execution times for the training experience.

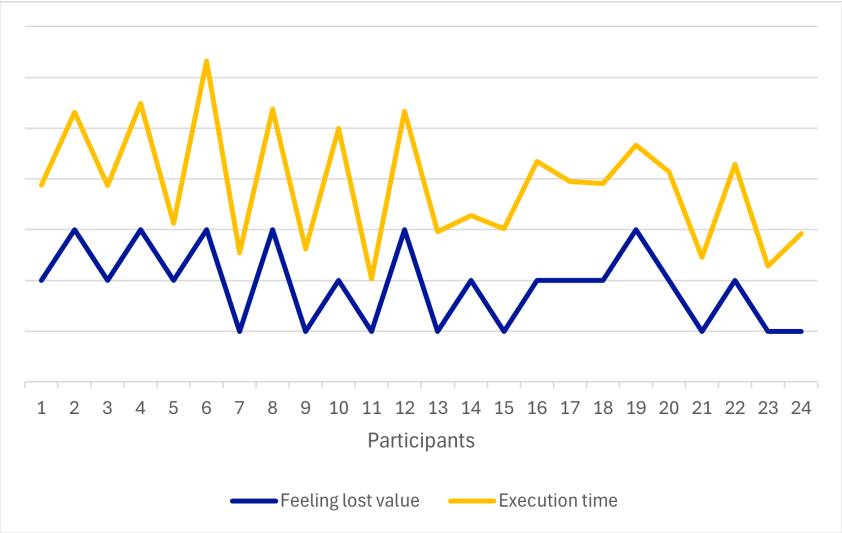

Figure 5.6. Comparison between the answers to the question "did you ever feel lost or confused?" and the execution times during the training phase.

The average times for each level are fundamental for the scope of this thesis, not only for understanding the varying difficulty throughout the levels but also for comparing different performance between the two versions of this project. In the training experience, the trends for both guided and unguided versions are quite similar: the first two levels have comparable execution times, while the third level shows significantly higher times (Figure 5.7). This was expected, as the third level was the longest and most complex, requiring users to recall multiple tasks and interact with numerous objects.
In the learning part of the experience, the average times for the second level are very similar between the two versions. However, in the training phase, on average guided participants performed better. This

could be explained by the fact that, in the learning experience, guided participants took more time in completing the level but they were more attentive, enabling them to complete the training experience faster than unguided participants. Notably, the only person who failed to finish the third level used the unguided version of the application; in particular this user could not recall all the objects to put in the right place. Performing a T-Test on the execution times of the various levels for the different versions led to find that, even though in the learning phase the unguided version has higher average execution time and greater variability, there are no differences strong enough to be considered statistically significant considering a p-value of 0.187. On the other hand, executing the same test on the execution times of the training phase, showed a negative t-statistic and a p-value of 0.0011 which is largely below the 0.05 threshold, confirming that users of the guided version performed more efficiently than unguided participants.

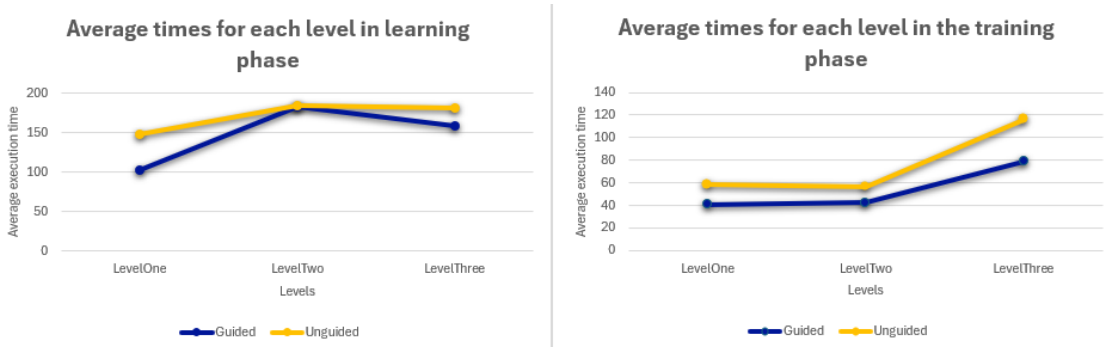

Figure 5.7. Average times for each level

On average, people with no prior experience using XR devices completed the training experience in 208,97 seconds, while those who used XR rarely completed it in 190,588 seconds, and those who used it weekly finished in 139,31 seconds. In particular, following the trends discussed up until now, people with the guided version who had never used any XR device took 163,57 seconds, compared to the same category in the unguided version that took 193,31 seconds. Similarly, among people who rarely used XR, guided participants took 151,19 seconds, and unguided participants completed the experience in 165,54 seconds. The only participants who used XR technologies weekly used the guided

version of the application, so no direct comparison can be made for that group.

5.4 Analysis summary

By analyzing the results from both subjective and objective measures, it is evident that participants using the guided version performed better overall. This aligns with the expected result, as mechanisms that help users regain or maintain focus, contribute to a better understanding of the explanations. While participants using the unguided version tended to ask many questions regarding the task they were executing, guided participants were generally more confident in completing every step of the experience. These results demonstrate that attention-monitoring mechanisms that adapt to user engagement, contribute in having better learning outcomes, fewer distractions and improved performance.

Chapter 6

Conclusion

The primary goal of this thesis was to investigate whether the use of attention-checking mechanisms could improve the learning experience in extended reality (XR) environments that use procedural learning, which involves acquiring knowledge through the repetition of ordered tasks. For an application that has learning as the final goal, it is essential to ensure that users remain focused, as distractions could interfere with their ability to effectively reach the intended learning objectives. By studying methods to reduce cognitive overload and guide users towards a target object, a VR application was developed using spatial awareness techniques such as an arrow indicating the direction of the object of interest and a circle highlighting it. The tasks for each level of the experience were carefully designed to be appropriately challenging without being overly difficult, ensuring that the test results depended primarily on the methods used rather than the difficulty of the tasks.
In the testing phase two groups were compared: one with adaptive pauses when user attention shifted away from the target object (which was the main focus of this thesis), and another without this feature. The results demonstrated that participants who used the version of the application with the attention-checking mechanism had significantly better completion times compared to the the unguided version, which did not include this characteristic. In addition to better completion times, users that used the guided version had a better learning experience compared to those using the unguided version. It was also shown how the self-evaluation of the participants closely matched their actual

performance, with users who reported feeling less distracted and more concentrated on the experience obtained lower completion times.
These findings suggest that integrating mechanisms to detect and respond to shifts in user attention can significantly enhance learning outcomes. By pausing instructions during inattentive moments, users are more likely to stay engaged and perform better in tasks. This could be particularly beneficial in complex training simulations such as those for medical students or pilots, where maintaining high concentration is critical for the success of the experience. Ultimately, the evidence gathered during this study has the potential to improve applications utilizing procedural learning across various fields, from educational to professional development.

6.1 Future works

The main limitation of this study is mainly given by the use of random sample. Several factors, such as participants' previous experience with VR/XR technologies, needed careful consideration when analyzing the results. Future studies could aim to expand the sample size and group participants based on predefined characteristics, such as prior experience with VR, age, or technical familiarity. This approach would enable a more detailed analysis of the results by comparing more homogenous groups. Moreover, prolonging the VR experience to include more complex or challenging tasks could further test how attention-checking mechanisms improve the learning outcomes. This could be combined with assessments of user knowledge before and after the experiment.
Lastly, future research could explore whether attention-tracking mechanisms can improve applications that rely on declarative learning. Investigating how these mechanisms influence memory retention or conceptual understanding could offer valuable insights into their wider application in various learning contexts.

Bibliography

[1] D. Kolb, *Experiential Learning: Experience As The Source Of Learning And Development*, 01 1984, vol. 1.

[2] F. Ganier, "Factors affecting the processing of procedural instructions: Implications for document design," *IEEE Transactions on Professional Communication*, vol. 47, no. 1, pp. 15–25, 2004.

[3] L. Daling and S. Schlittmeier, "Effects of augmented reality, virtual reality, and mixed reality- based training on objective performance measures and subjective evaluations in manual assembly tasks: A scoping review," *Human Factors The Journal of the Human Factors and Ergonomics Society*, vol. 66, 05 2022.

[4] D. Schofield, Z. Cai, O. Medonza, K. Ray, C. Le, and J. Tromp, *Human Factors for an E-Health Training System: UX Testing for an XR Anatomy Training App.* Wiley, 04 2020, p. 81.

[5] R. Mayer and R. Moreno, "Nine ways to reduce cognitive load in multimedia learning," *Educational Psychologist - EDUC PSYCHOL*, vol. 38, pp. 43–52, 03 2003.

[6] S. Yang, Y. He, and Y. Chen, "Spatialgaze: towards spatial gaze tracking for extended reality," *CCF Transactions on Pervasive Computing and Interaction*, vol. 5, pp. 430–446, 10 2023.

[7] T. Schuchert, S. Voth, and J. Baumgarten, "Sensing visual attention using an interactive bidirectional hmd," in *Proceedings of the 4th Workshop on Eye Gaze in Intelligent Human Machine Interaction.* New York, NY, USA: ACM, 2012, pp. 1–3.

[8] R. Boulic, D. Maupu, M. Peinado, and D. Raunhardt, "Spatial awareness in full-body immersive interactions: Where do we stand?" in *Motion in Games*, R. Boulic, Y. Chrysanthou, and T. Komura, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg,

2010, pp. 59–69.

[9] J. O. Wallgrün, M. Bagher, P. Sajjadi, and A. Klippel, "A comparison of visual attention guiding approaches for 360° image-based vr tours," 03 2020, pp. 83–91.

[10] A. Irlitti, T. Piumsomboon, D. Jackson, and B. Thomas, "Conveying spatial awareness cues in xr collaborations," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, pp. 1–1, 08 2019.

[11] J. Blattgerste, K. Vogel, C. Lewa, and T. Pfeiffer, "Trainar: A scalable interaction concept and didactic framework for procedural trainings using handheld augmented reality," *Multimodal Technologies and Interaction*, vol. 5, p. 30, 06 2021.

[12] S. Wish-Baratz, A. P. Gubatina, R. Enterline, and M. A. Griswold, "A new supplement to gross anatomy dissection: Holoanatomy," *Medical education*, vol. 53, no. 5, pp. 522–523, 2019.

[13] N. Chandrashekar, M. Manuel, J. Park, A. Greene, S. Safford, and D. Gracanin, *An Extended Reality Simulator for Advanced Trauma Life Support Training.* Springer International Publishing AG, 05 2022, pp. 31–44.

[14] ScopeAR. (2016) Worklink. [Online]. Available: https://www.scopear.com/

[15] DigitalNauts. Pharmaceutical manufacturing vr training. [Online]. Available: https://www.digitalnauts.co.uk/pharmaceutical-vr-training-case-study/

[16] K. Hanáčková. (2021) Is mixed reality the future of corporate training? [Online]. Available: https://medium.com/brainz-vr/is-mixed-reality-the-future-of-corporate-training-a97b15bddbc7

[17] C. Catal, A. Akbulut, B. Tunali, E. Ulug, and E. Ozturk, "Evaluation of augmented reality technology for the design of an evacuation training game," *Virtual Reality*, vol. 24, pp. 359–368, 09 2020.

[18] J. Martin-Gutierrez, "Generic user manual for maintenance of mountain bike brakes based on augmented reality," *Proceedings of the 28th International Symposium on Automation and Robotics in Construction, ISARC 2011*, 01 2011.

[19] ARM. What is a gaming or game engine? [Online]. Available: https://www.arm.com/glossary/gaming-engines

[20] University of Silicon Valley. What is a game engine? [Online].

Available: https://www.arm.com/glossary/gaming-engines
[21] M. Rouse. Rendering. [Online]. Available: https://www.techopedia.com/definition/9163/renderingl
[22] Unity Technologies. Introduction to render pipelines. [Online]. Available: https://docs.unity3d.com/6000.0/Documentation/Manual/render-pipelines-overview.html
[23] ——. Choose a render pipeline. [Online]. Available: https://docs.unity3d.com/Manual/choose-a-render-pipeline.html
[24] M. Hergaarden, "Graphics shaders," 2011.
[25] Lenovo. What is a pixel shader? [Online]. Available: https://www.lenovo.com/us/en/glossary/pixel-shader/?orgRef=https%253A%252F%252Fwww.google.com%252F&srsltid=AfmBOord47JD4DXef7nJouhpjStzPMAmBu76nL1TqSz-MdFHuhRz6D11
[26] Medium. Graphics pipeline. [Online]. Available: https://medium.com/@rakadian/graphics-pipeline-9e4bb2d28f58
[27] Unity Technologies. Introduction to shaders. [Online]. Available: https://docs.unity3d.com/Manual/shader-introduction.html
[28] ——. Rotation and orientation in unity. [Online]. Available: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html
[29] YourDictionary. Gimbal-lock definition. [Online]. Available: https://www.yourdictionary.com/gimbal-lock
[30] Unity Technologies. Rigidbody component reference. [Online]. Available: https://docs.unity3d.com/Manual/class-Rigidbody.html
[31] ——. System requirements for unity 2022.3. [Online]. Available: https://docs.unity.cn/2022.3/Documentation/Manual/system-requirements.html
[32] ——. Xr interaction toolkit. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html#requirements
[33] R. Awati. Field of view. [Online]. Available: https://www.techtarget.com/whatis/definition/field-of-view-FOV#:~:text=Field%20of%20view%20(FOV)%20is,much%20can%20the%20device%20see%3F%22
[34] H. R. Thomas Schubert, Frank Friedmann. Ipq. [Online]. Available:

https://www.igroup.org/pq/ipq/index.php