Design Project Report

Co-located Extended Reality Game and Short-Term Memory Test

Group 12:

Alexander Fourie s2509253
Bora Ciftci s246476
Arda Duyum s2425688
Sophie van der Dong s2497255

Supervisor: Gwen Qin

University of Twente.

Table of Contents

Abstract

Extended reality (XR) technologies can alter one's perception of reality by creating external stimuli such as virtual objects that don't actually occur within nature. These technologies can have a negative impact on their users, typically referred to as cybersickness, characterised by nausea, dizziness and eye strain as some of the possible effects. For our project, we aim to create an application that enables researchers to conduct experiments on the effect that latency has on users, exploring the possible link between short-term memory capabilities and the extent of cybersickness experienced. There exist research attempts on latency as tried by Nguyen for MR technologies, however current research similar to this project is mostly related to other XR technologies, such as virtual reality (VR), where Stauffert noted cybersickness occurring from latency within its use. Therefore, the purpose of this report is to create an open-source application that allows researchers to test and see how latency affects multi-user collaboration with MR technologies. Our application consists of two parts: a short term memory experiment and the latency influenced game experiment. In the first part the user's short term memory capabilities are measured by having to select the correct colour order for the generated objects. In the second part two users have to collaboratively work on 'popping' virtually generated balls while their gameplay is influenced by varying latency. The first part can be conducted on the Hololens 2, whereas the second part can be conducted on the Meta Quest Pro or 3. Possible future additions such as a heart rate monitor could be implemented with this application for additional information to be gained during the experiments such that a more accurate understanding of the full effect latency has on cybersickness could be established.

Keywords: Extended reality, mixed reality, augmented reality, cybersickness, latency, short term memory

CLXR-G	Co-located Extended Reality Game
STM-T	Short-Term Memory Test
AR	Augmented Reality
VR	Virtual Reality
MR	Mixed Reality
XR	Extended Reality
PUN	Photon Unity Networking
MRTK	Mixed-Reality Tool Kit
QoE	Quality of Experience
UVC	Unity Version Control
RPC	Remote Procedure Call

Table 1: Abbreviations for frequently used terms

1 Introduction

This report presents the designs of a short-term memory test developed for the Microsoft Hololens 2, and a co-located multi-user mixed reality game compatible with the Meta Quest Pro and Meta Quest 3. The project is undertaken as part of the Design Project Module in collaboration with our supervisor, who is pursuing her PhD and aims to study the effects of network latency on user experience, specifically cybersickness, in mixed-reality environments. Unlike typical design projects, which often involve direct collaboration with companies to create or improve products, this project serves a research-oriented purpose: To develop a functional prototype that can facilitate an experimental study.

The game is designed to meet specific criteria set by our supervisor, who acts as the primary stakeholder. They will utilise this game as a core component in their experimental research on how latency influences collaborative user experiences in MR environments. Our task is to ensure the game mechanics, networking conditions, and functionality meet her requirements for valid experimentation.

The evolution of MR and XR technologies has dramatically transformed gaming and collaborative virtual environments, allowing players to interact in both AR and VR settings. Co-located multi-user experiences, where users engage together in the same physical space, are increasingly popular as they blend digital and physical interactions, enriching immersion and collaborative dynamics. However, achieving seamless interaction in these environments, especially under varying network conditions, presents technical challenges, including latency's impact on user experience and performance.

1.1 Background

1.1.1 Co-located Immersive Gaming in AR and VR

The increasing availability of MR headsets and immersive technologies has transformed modern gaming, particularly in collaborative and co-located experiences. These applications use AR and VR platforms, enabling players to interact within digital and physical hybrid environments. Important to note are the differences between these platforms and how they are defined. AR, as a technology, involves overlaying virtual objects with the real world. Comparatively, VR involves the user being immersed in a fully digitalized world which they can interact with. From these technologies MR exists as the mix between these technologies where physical and digital objects coexist and can be interacted with. XR is the term that can be considered to encapsulate all these technologies that alter a person's perception of reality (Vega, 2020).

Existing research on XR technologies in immersive gaming has primarily investigated how VR and AR environments independently impact player presence, co-presence, and engagement. For instance, Ghoshal et al. compared user experiences between AR and VR during collaborative gameplay and found that VR typically enhances the player's sense of presence due to their full immersion in virtual spaces, while AR environments enhances co-presence due to its integration with the physical environment, where players are allowed to interact with both virtual and physical elements, including each other, in real-time (Ghoshal, 2022). These distinctions are crucial for understanding how the player's physical and digital surroundings influence their overall experience using these technologies.

1.1.2 The Role of Short-term Memory with Cyber Sickness

The Short-Term Memory Test (STM-T) aims to explore the relationship between cybersickness and short-term memory performance in mixed reality (MR) environments. While there is no direct research on this specific correlation, the existing literature on cybersickness and motion sickness, as well as their respective impacts on cognitive performance, provides a foundation for understanding potential effects.

Cybersickness is a phenomenon characterised by symptoms such as nausea, disorientation, which arise from a sensory conflict between visual and vestibular inputs within virtual environments (Rebenitsch & Owen, 2016). These symptoms overlap with those of motion sickness (Kennedy et al., 1993), as both conditions result from sensory dissonance which, besides the previously mentioned symptoms, can also lead to a decrease in cognitive performance.

Studies investigating the effects of motion sickness on cognitive performance have highlighted a decrease on short-term memory (Dahlman et al, 2017). They investigated how motion sickness induced by an optokinetic drum affected short-term memory performance, where their study revealed that short-term memory performance declined as motion sickness symptoms progressed.

Although cybersickness differs from motion sickness in its mechanism—arising from visual stimuli without corresponding vestibular input—it induces similar physiological and psychological symptoms. This similarity suggests that the effects of cybersickness on cognitive performance, including short-term memory, may parallel those observed in motion sickness. However, there is not much existing research on the personal characteristics that people possess and how this affects the rate and extent at which cybersickness affects them. While there is research done to some extent on factors such as mental rotation ability and ethnicity (Lawson et al, 2021), short term memory has not been included. Therefore, as it is a basic cognitive function that is impacted by cybersickness, this project aims to aid in the research towards learning the extent a person's initial short term memory capabilities affects the extent to which they would experience cybersickness.

1.1.3 The Role of Latency in User Experience

Another area of study involves the influence of latency on multi-user collaborative experiences. According to Van Damme et al., multi-user VR collaboration significantly depends on low latency to maintain QoE and prevent negative effects on player dynamics and task performance (Van Damme, 2019). Their findings indicate that while VR requires more stringent latency management to avoid user discomfort, AR's transparency with the physical world could make it less susceptible to cybersickness, even with moderate delays. Latency levels exceeding 60 ms can disrupt the sense of presence and impair user interactions, especially in high-engagement tasks (Caserman, 2019). For co-located applications, this effect is compounded by the need for synchronised interactions between players, where latency impacts both the sense of agency and physical coordination.

However, few studies have explored the influence of network latency on QoE in MR environments, where AR and VR components are mixed. Our project aims to address this gap by developing a mixed-reality game suitable for co-located collaborative play in which network latency can be simulated.

1.2 Project Goals

There are two main goals of this project. First, to develop a co-located, collaborative mixed-reality game compatible with the Meta Quest that will serve as a tool for studying the effects of network latency on user experience in immersive environments. Second, to develop a short-term memory test compatible with the Hololens 2. Our work will serve as the foundation for our supervisor's experimental setup, aiming to provide a well-designed game that simulates realistic latency scenarios, and to provide a tool to assess the short-term memory capacities of the participants. Unlike prior research, which often examines AR and VR environments separately, this project offers a unique perspective by examining the effects of latency in a mixed-reality collaborative game format.

While our contribution is focused on the technical aspects of building a reliable and interactive game, the broader purpose is to facilitate a comprehensive experimental study on latency and its impact on user experience. Our findings will inform future game development strategies, particularly in latency management for mixed-reality environments.

1.3 Project Scope

This project's scope involves designing and developing two separate applications, the STM-T and the CLXR-G. The STM-T which is developed on the Hololens 2 and the CLXR-G which is developed on the Meta Quest 3. The specific components of the project include.

1.3.1 Scope

1.3.1.1 Game Development

- Develop an application involving one user, wearing a Hololens 2 headset, who interacts with virtual spheres and the available colours. The user must select a colour and color the spheres to go through the rounds. This application will measure the user's performance to gain measurements indicating their short-term memory capabilities (STM-T).
- Develop a collaborative application involving two users, each wearing Meta Quest 3 headsets, who interact with virtual spheres in a shared physical space. Users must touch the same sphere simultaneously to make them disappear. The application will simulate network latency (CLXR-G).

1.3.1.2 Latency Simulation

- For the CLXR-G, introduce artificial network latency for each player with different patterns of delay.

1.3.1.3 Data Collection

- Implement mechanisms for logging data relevant to research in STM-T and CLXR-G.
- STM-T: log time taken for users to finish a round and a user's colouring accuracy.
- CLXR-G: log time stamps of the simulated latency and the amount of latency throughout the duration of the gameplay for each participant.

1.3.1.4 Experiment Facilitation

- The applications serve as a research tool for our supervisor and will become open source. Therefore, it must be robust, easy to set up, and flexible to accommodate the changes required by the supervisor or other researchers.

1.3.2 Exclusions and Constraints

1.3.2.1 No Direct Experimental Research

- The project does not include conducting any experimental research. The task of obtaining users to partake in the experiments, creating surveys for measuring user's physiological state after partaking in the CLXR-G, running the experiments and analysing their subsequent results will all be carried out by our supervisor as part of her PhD research.

1.3.2.2 No Long-Term Support or Updates

- The project will not include long-term maintenance or post-release support for the applications. Once the functional application prototypes are delivered, the

responsibility for further iterations and improvements will fall outside our group's scope.

1.3.3.1 Time Constraints

- The project must be completed within the timeframe of the Design Project module. This constrains the amount of additional functionality that can be developed, emphasising the importance of focusing on core features critical to the research objectives.

1.3.3.2 Hardware and Platform Limitations

- Due to discontinuation of shared spatial anchors, a service provided by Microsoft for the Hololens 2, the CLXR-G had to be developed for the Meta Quest Pro and Quest 3.
- The STM-T will be developed specifically for the Hololens 2. No development or testing will be done for other hardware platforms.

1.3.3.3 Networking Constraints

- All networking aspects will be implemented using a cloud-based solution like Photon's PUN, which may introduce additional latency that we do not have control over.

1.3.3.4 Data Privacy and Ethics:

- Since metrics on the user's performance is collected, privacy must be ensured. All data collected will need to be anonymized to comply with ethical standards. Personal identifying information will not be stored or processed.

1.3.3.5 Technical Expertise of Team Members

- The development team comprises four students with varying levels of experience in Unity and networking. Consequently, design decisions may be influenced by our existing knowledge base and learning curve.

2 Project Organisation

2.1 The Team's roles and responsibilities

Name	**Roles/Responsibilities**
Alex Fourie	STM-T system integration, CLXR-G system integration, report
Arda Duyum	STM-T system integration, system design, report
Bora Ciftci	STM-T research, CLXR-G research and system design, report
Sophie van der Dong	STM-T system integration, system design, report

Table 2: Team Roles and Responsibilities

To ensure a software development project is completed successfully, it is important for every project member in this team to be aware of their assigned roles and responsibilities. This is to maintain a proper organisation of workflow in which tasks are fulfilled in a way that nothing important necessary to the completion of this project is omitted. All members are capable of programming and were assigned work related to this. To keep work efficient members are delegated to complete various components of the applications necessary for proper experimentation to occur. This is indicated by the Gantt chart key represented in Figure 1.It represents a planned timeline where members are delegated either to the STM-T or CLXR-G application, with the division of testing, documenting, etc being done accordingly to an extent that covers their programming and their process thereof.

2.2 Risk Analysis

2.2.1 Schedule pressure

The proposed project schedule could be unrepresentative of real life as external factors and unexpected circumstances can bring forth deviations to the initially planned schedule. In addition, with the limited time available any such deviations could cause a major impact on the subsequent tasks needed to occur. Consequently, this could cause adjustments to the deliverable product to be necessary.

2.2.2 People's interpersonal relationships

Within any project a proper work environment is necessary to keep productivity high. To maintain this, positive group dynamics and clear communication is necessary. Potential conflicts, high stress factors and strong differing opinions can lead to issues that would decrease this, breaking communications and causing a drop in productivity and motivation within the team.

2.2.3 Lack of Technical Documentation

Technical documentation allows the applications to be easily understandable and adaptable by future developers. Doing this improperly would cause people unrelated to the initial team to have difficulty in further expanding upon this work.

2.2.4 Secondary innovations / Software deviations

Deviations to the software can occur through human error with inadequate research and implementing programming solutions improperly. Additionally, incomprehension can cause misinterpretation, leading to include unnecessary features or functionalities to be implemented. Other factors, such as an impulse to improve solutions or wanting to expand upon a requirement when unneeded would also lead to such innovations and deviations from the original planned product to occur.

2.2.5 Requirement changes

Throughout the development phase of our project, as we increase our understanding of the software we develop, new skills and an improved understanding of our project and its scope, the requirements are bound to change. This will lead to some requirements becoming redundant or reprioritized from its initial standpoint.

2.2.6 Absences

Absences caused by illness, unexpected emergencies or other circumstances would cause a drop in productivity. If this occurs extensively, it will accumulate to the point of delays to an extent that milestones are not reached within the expected timeframe.

2.2.7 Lack of Equipment

As we develop an application for the Hololens 2 and Meta Quest 3, it is necessary to have these devices available to us, for their development and testing out certain

functional requirements. However due to limited availability, it can occur that we do not have access to either of these devices, causing possible bugs or Quality code to remain undetected for longer than if the devices were otherwise available to us.

2.4 Project Plan

The Gantt chart presented in Figure 1 outlines a timeline of our intended workflow for this project, where we develop a co-located mixed reality experiment (CLXR-G) and a short term memory experiment (STM-T). This timeline spans a duration of ten weeks from September through October and specifies the various tasks we have determined necessary for our project. These tasks are stemmed from our various development phases which includes a preparatory phase in addition to a phase associated to each application. Each individual task is assigned to one of the existing team members (Alex, Sophie, Arda, and Bora) or is worked on in a collaborative effort from all team members, as is represented by color-coded bars which meaning is indicated in the Gantt's chart key.

Initially, the project covers foundational tasks such as preparing the project proposal, setting up the Unity project environment, and establishing version control. The STM-T phase focuses on the implementation and validation of core game logic components, specifically addressing aspects such as block visibility, differentiability, and interaction accuracy. This phase ensures the reliability and functionality of the game environment prior to experimental deployment.

During the CLXR-G phase, unforeseen challenges related to spatial anchors caused a significant deviation from the planned timeline. These issues impacted the development and integration of multi-user capabilities, ultimately delaying the entire project schedule.

Optional expansions, such as heart rate recording, are not included in the final product due to time constraints. The final stages of the project focus on documentation, the development of a user manual, and the preparation of a project report and presentation.

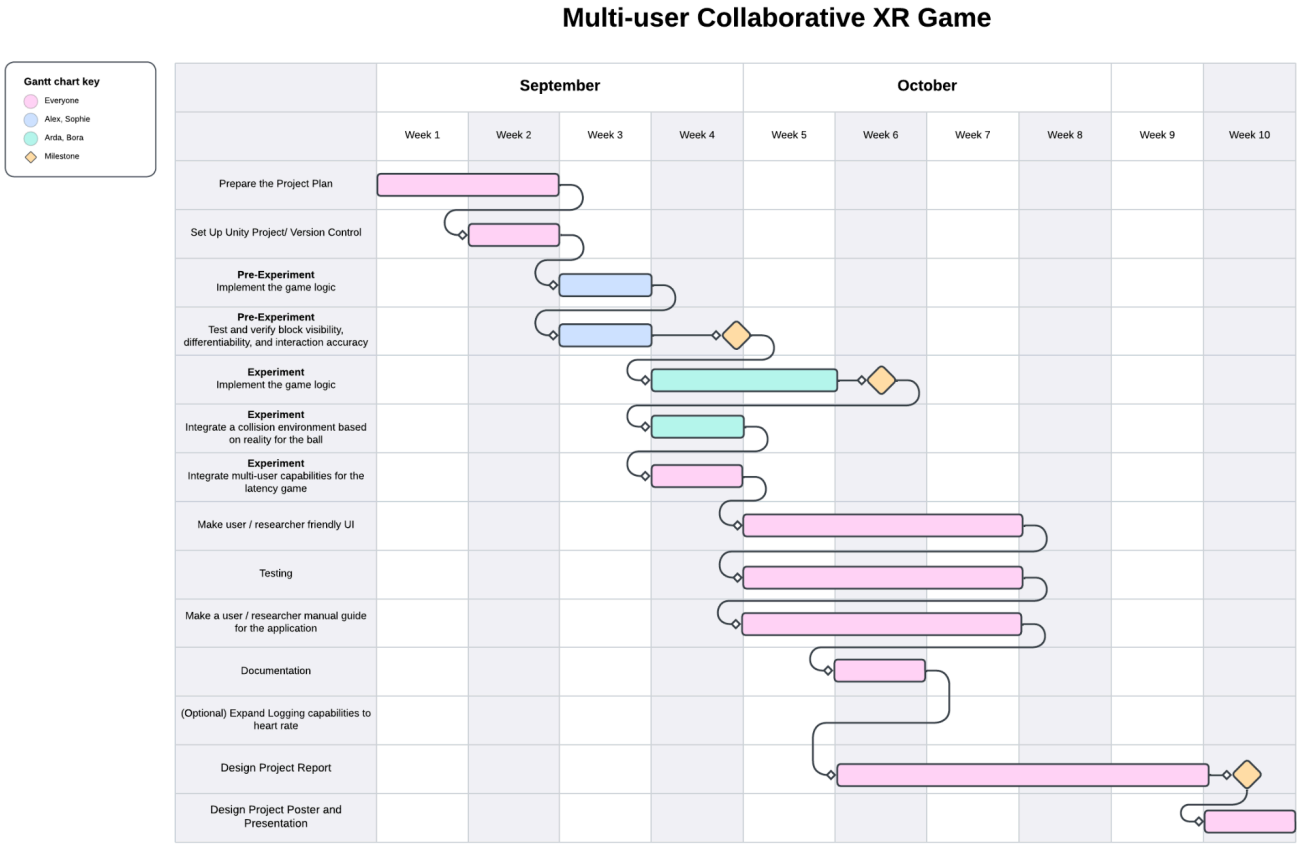


Figure 1: Project plan

2.5 Stakeholders

Stakeholder	**Details**
Client - Supervisor	The primary stakeholder of the project. Gwen Qin has been a member of the Pervasive Systems group within the Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS) at the University of Twente since 2023. Graduating with a computer science master's degree from the University of Tsukuba in 2023. Her current work focuses on collaborative Augmented Reality and Extended Reality experiences, particularly in the context of predictive maintenance in industrial settings. She aims to address challenges such as spatial inconsistency and latency in collaborative Extended Reality environments, while also exploring effective multi-user interaction and remote collaboration methods. The game will serve as an experimental tool for her research.
Players - Participants	Individuals who will participate in the gameplay sessions, providing experimental data related to user experience under varying latency conditions.
Development Team	A group of four students responsible for designing, developing, and implementing the game. They bring varying levels of

	experience in Unity and networking technologies.

Table 3: Stakeholders

3 System Specification

3.1 Requirements

3.1.1 Introduction

From the initial meetings conducted with the client, several key requirements were established to facilitate the research goals. The table (Table 4) categorises the various requirements and has defined the key modal verbs that are used throughout the specification of requirements, highlighting the different levels of obligation and priority assigned to each requirement, as has been established using the MoSCoW method.

Term	**Definition**
Functional Requirements	These are the requirements that the end user specifically demands as basic facilities that the system should offer.
Quality Requirements	These are the quality constraints that the system must satisfy according to the project contract.
Must	Mandatory for the project.
Should	Adds significant value but is not vital.
Could	Small impact, nice-to-have.

Table 4: Terms and Definitions for the Requirements

3.1.2 Short term memory experiment

The requirements for the STM-T can be found in Table 5. The functional user requirements, described in requirements 1 to 7, focus on the actions the user must take, such as performing gestures, selecting items from the hand menu, and achieving a specific accuracy threshold to advance rounds. The system functional requirements, described in requirements 8 to 10, ensure that the system provides appropriate feedback through voice and text, displays the necessary guidance, and manages the progression of the experiment. The quality requirements for the user are in requirements 11 to 15, ensuring that the interface is intuitive, accessible, and responsive. The system quality requirements, in requirements 16 to 18, ensure performance and user immersion by managing latency, providing understandable instructions, and ensuring virtual objects do not interfere with real-world objects

Nr.	Description	Type
1	The user must make the Palm Up gesture to reveal the Hand Menu when guided by the Left Hand Coach.	Functional
2	The user must click on the buttons in the hand menu to reveal their functionality through text and voice.	Functional
3	The user must select a colour from the colour tray and click	Functional

	on the sphere to change its colour during the trial round and the STM-T when the tray type is set to 0.	
4	The user must select a colour from the hand menu and drag and drop the coloured sphere into the sphere they want to colour in the trial round and the STM-T when the tray type is set to 1.	Functional
5	The user must select a colour from the hand menu and click on the sphere to change its colour during the trial round and the STM-T when the tray type is set to 2.	Functional
6	The user should press the "Start next round" button on the hand menu or click the button on the left of the spheres to proceed to the next round.	Functional
7	The user should achieve 50% accuracy or higher to generate an additional sphere in the next round.	Functional
8	The system must reveal the functionality of hand menu buttons with text and voice once clicked.	Functional
9	The system should display the start button for the trial round and the actual experiment while guiding the user with text and voice.	Functional
10	The system must display a message indicating that the real experiment will start after the STM-T concludes.	Functional
11	The coloured spheres on the colour tray must be clearly visible and easy to click when the tray type is set to 0.	Quality
12	The coloured spheres on the colour tray must be clearly visible and easy to drag and drop when the tray type is set to 1.	Quality
13	The buttons on the hand menu must be clearly visible and easy to click when the tray type is set to 2.	Quality
14	The colours displayed on the spheres should be differentiable for users with colour blindness.	Quality
15	The Palm Up gesture detection should be responsive with minimal delay.	Quality
16	The system must ensure that virtual objects are not placed within real objects.	Quality
17	The system must maintain a latency of 10-15 ms for interaction responsiveness.	Quality
18	The robot avatar’s instructions should be simple and understandable.	Quality

Table 5: Requirements for the STM-T

3.1.3 Co-located mixed reality experiment

The requirements for the CLXR-G can be found in Table 6. For the experiment, the functional requirements described in requirements 1 to 7 focus on system actions that handle the interaction between users, such as introducing global latency, logging data, and managing the game mechanics like the ball movement. The quality requirements, defined in 8 to 9, ensure that the system maintains user immersion by minimising spatial inconsistencies and simplifying the UI for easy interaction.

Nr.	Description	Type
1	The system must spawn 10 networked spheres in random positions in the room the users are in.	Functional
2	Spheres must disappear upon simultaneous touches by both players.	Functional
3	Spheres must react to the physical environment similar to a basketball, i.e., the spheres bounce and make contact with the walls and the floor.	Functional
4	The system could provide a tutorial before the start of the game for new users to familiarise themselves with the way of interacting with virtual objects using the Meta Quest 3	Functional
5	The system must simulate network latency during gameplay	Functional
6	Every 200 ms, the system must simulate a network latency of 100 ms with a chance of 50%, the latency experienced by each player should be determined locally and should not affect the other player	Functional
7	The system must log the simulated latency along with the timestamp of the simulated latency for analysis	Functional
8	The system must simulate a maximum latency of 100 ms to ensure the well-being of the candidates	Functional
9	The system must allow for an initial calibration phase to ensure all participants' devices are aligned in the same physical space before starting the game.	Functional
10	The game should maintain at least 90 frames-per-second to ensure smooth user experience	Quality
11	Latency simulation should add minimal overhead without disrupting core game functionalities.	Quality
12	The system must ensure that data collection complies with privacy requirements and ethical standards.	Quality
13	The onboarding process should be completed within two minutes, with clear visual or verbal instructions to minimise learning time.	Quality

14	The system must support two concurrent users without degradation in performance.	Quality
15	The system must maintain consistent synchronisation of the virtual objects between users.	Quality
16	The system must support room creation and entering for multiplayer purposes.	Functional
17	The spheres should be able to be manipulated/realistically interactable (e.g., picking up and moving around).	Functional

Table 6: Requirements for the CLXR-G

3.2 User Stories

3.2.1 Short-term memory experiment

The user stories below describe the user and system perspective of the experiment.

User Stories

- **US1:** As a player, I want to make the Palm Up gesture to reveal the Hand Menu when guided by the left-hand coach, so that I can access the interaction options and understand how to interact with the interface.
- **US2:** As a player, I want to click on the buttons in the hand menu, so that I can understand their functionality through text and voice feedback, ensuring that I am aware of what each button does.
- **US3:** As a player, I want to select a colour from the hand menu and apply it to the sphere during the trial round, so I know how to play the short-term memory game and familiarise myself with the mechanics of the interaction.
- **US4:** As a player, I want to press the "Start next round" button on the hand menu or click the "Next Round" Button on the left of the spheres, so that I can advance to the next round when I am ready and feel confident in my understanding of the game.
- **US5:** As a player, I want to achieve at least 50% accuracy in each round, so I can challenge my short-term memory with an additional sphere and gauge my performance in the STM-T.
- **US6:** As a player, I want the buttons in the hand menu to be clearly visible and easy to click, so that I can interact with the interface efficiently, even during the trial phase.
- **US7**: As a player, I want the colours on the spheres to be distinguishable even if I have colour blindness, so I can complete the task successfully and not be hindered by visual limitations.
- **US8:** As a player, I want Palm Up gesture detection to respond with minimal delay, so that I can interact smoothly and without frustration, allowing me to focus on learning the game.
- **US9:** As a player, I want the game to appear in front of me without having to move my head up or down, so that I can comfortably view and interact with the game without straining my neck.
- **US10:** As a player, I want all interactive game objects to have a voice effect during inter-actions, so it is clear when an action has been successfully completed.
- **US11:** As a player, I want the robot avatar’s instructions to be simple and understandable, so that I can easily follow the task without confusion and be well prepared for the main experiment.
- **US12**: As a system, during the trial round, I want to reveal the functionality of the hand menu buttons with text and voice once clicked, so that users can quickly learn how to interact with the game elements.

- **US13**: As a system, I want to have the Hand Coach displayed properly and synchronised with user actions, so that the user knows exactly how to interact with the game objects and avoid confusion.
- **US14**: As a system, I want to display a message indicating that the real experiment will start after the STM-T concludes, so that users are mentally prepared for the main experiment and understand the transition.
- **US15**: As a system, I want to ensure that virtual objects are not placed within real objects, so that user immersion and safety are maintained, preventing any disruptive overlap during the STM-T.
- **US16**: As a system, I want to maintain a latency of 10-15 ms for interaction responsiveness, so that the user experience remains fluid and accurate, providing realistic feedback during the STM-T.
- **US17**: As a system, I want to guide the user step by step through the STM-T with minimal intervention needed, so that the user can quickly become proficient in the experiment's tasks.

3.2.1.1 Acceptance Criteria

User Story 1: As a player, I want to make the Palm Up gesture to reveal the Hand Menu when guided by the left-hand coach, so that I can access the interaction options and understand how to interact with the interface.

Scenario: Revealing the Hand Menu using the Palm Up gesture.

- **Given** the player is guided by the left-hand coach,
- **When** the player performs the Palm Up gesture,
- **Then** the Hand Menu should appear, allowing access to interaction options.

User Story 2: As a player, I want to click on the buttons in the hand menu, so that I can understand their functionality through text and voice feedback, ensuring that I am aware of what each button does.

Scenario: Receiving feedback upon clicking hand menu buttons.

- **Given** the Hand Menu is open,
- **When** the player clicks on a button in the hand menu,
- **Then** the system provides text and voice feedback explaining the button's functionality.

User Story 3: As a player, I want to select a colour from the hand menu and apply it to the sphere during the trial round, so I know how to play the short-term memory game and familiarise myself with the mechanics of the interaction.

Scenario: Applying a selected colour to a sphere during the trial round.

- **Given** the trial round is active and the Hand Menu is open,
- **When** the player selects a colour from the hand menu and clicks on a sphere,
- **Then** the sphere changes to the selected colour, confirming the interaction.

User Story 4: As a player, I want to press the "Start next round" button on the hand menu or click the "Next Round" Button on the left of the spheres, so that I can advance to the next round when I am ready and feel confident in my understanding of the game.

Scenario: Advancing to the next round of the game.

- **Given** the current round has ended,
- **When** the player presses the "Start next round" button on the hand menu or clicks the "Next Round" button,
- **Then** the system initiates the next round of the game.

User Story 5: As a player, I want to achieve at least 50% accuracy in each round, so I can challenge my short-term memory with an additional sphere and gauge my performance in the STM-T.

Scenario: Increasing game difficulty based on user performance.

- **Given** the player has completed a round,
- **When** the player achieves 50% accuracy or higher,
- **Then** the next round includes an additional sphere to increase the challenge.

User Story 6: As a player, I want the buttons in the hand menu to be clearly visible and easy to click, so that I can interact with the interface efficiently, even during the trial phase.

Scenario: Ensuring hand menu buttons are user-friendly.

- **Given** the Hand Menu is displayed,
- **When** the player views the buttons,
- **Then** the buttons are clearly visible and sized appropriately for easy clicking.

User Story 7: As a player, I want the colours on the spheres to be distinguishable even if I have colour blindness, so I can complete the task successfully and not be hindered by visual limitations.

Scenario: Providing colour accessibility for users with colour blindness.

- **Given** the spheres are displayed in various colours,
- **When** the player has a colour vision deficiency,
- **Then** the colours are selected to be distinguishable through patterns or shades to accommodate colour blindness.

User Story 8: As a player, I want Palm Up gesture detection to respond with minimal delay, so that I can interact smoothly and without frustration, allowing me to focus on learning the game.

Scenario: Responsive gesture detection for smooth interaction.

- **Given** the player performs the Palm Up gesture,
- **When** the gesture is made,
- **Then** the Hand Menu appears with minimal delay (within 10-15 milliseconds).

User Story 9: As a player, I want the game to appear in front of me without having to move my head up or down, so that I can comfortably view and interact with the game without straining my neck.

Scenario: Positioning game elements within the user's comfortable field of view.

- **Given** the player is facing forward,
- **When** the game begins,

- **Then** the game elements are positioned directly ahead at eye level within the user's natural line of sight.

User Story 10: As a player, I want all interactive game objects to have a voice effect during interactions, so it is clear when an action has been successfully completed.

Scenario: Audio feedback upon interacting with game objects.

- **Given** the player interacts with a game object (e.g., colouring a sphere),
- **When** the action is successfully completed,
- **Then** a voice effect or sound plays to confirm the successful interaction.

User Story 11: As a player, I want the robot avatar's instructions to be simple and understandable, so that I can easily follow the task without confusion and be well prepared for the main experiment.

Scenario: Clear and concise instructions from the robot avatar.

- **Given** the player started the game,
- **When** the avatar provides instructions,
- **Then** the instructions are simple, step-by-step, and easily understandable.

User Story 12: As a system, during the trial round, I want to reveal the functionality of the hand menu buttons with text and voice once clicked, so that users can quickly learn how to interact with the game elements.

Scenario: Educating the user about hand menu buttons during the trial round.

- **Given** the player clicks a button in the hand menu during the trial round,
- **When** the button is clicked,
- **Then** the system provides immediate text and voice feedback explaining the button's purpose.

User Story 13: As a system, I want to have the Hand Coach displayed properly and synchronised with user actions, so that the user knows exactly how to interact with the game objects and avoid confusion.

Scenario: Proper display and synchronisation of the Hand Coach.

- **Given** the player needs guidance,
- **When** the Hand Coach demonstrates an action,
- **Then** it is displayed correctly and synchronised with the user's perspective and timing.

User Story 14: As a system, I want to display a message indicating that the real experiment will start after the STM-T concludes, so that users are mentally prepared for the main experiment and understand the transition.

Scenario: Informing the user about the transition to the main experiment.

- **Given** the player has completed the STM-T,
- **When** the STM-T ends,
- **Then** the system displays a clear message stating that the main experiment will begin shortly.

User Story 15: As a system, I want to ensure that virtual objects are not placed within real objects, so that user immersion and safety are maintained, preventing any disruptive overlap during the STM-T.

Scenario: Avoiding overlap between virtual and real-world objects.

- **Given** location of where the player's facing toward,
- **When** virtual objects are positioned,
- **Then** they are placed in front of user.

User Story 16: As a system, I want to maintain a latency of 10-15 ms for interaction responsiveness, so that the user experience remains fluid and accurate, providing realistic feedback during the STM-T.

Scenario: Ensuring low latency in user interactions.

- **Given** the player performs an interaction (e.g., gesture, button press),
- **When** the action occurs,
- **Then** the system responds within 10-15 milliseconds to maintain fluidity.

User Story 17: As a system, I want to guide the user step by step through the STM-T with minimal intervention needed, so that the user can quickly become proficient in the experiment's tasks.

Scenario: Providing step-by-step guidance throughout the STM-T.

- **Given** the player is starting the STM-T,
- **When** they progress through tasks,
- **Then** the system offers clear, sequential instructions requiring minimal additional support.

3.2.2 Co-located mixed reality experiment

The user stories in this section represent the needs of the players and researchers with an aim towards smooth collaboration. Compared to the STM-T here the application is made to allow collaboration between users without issues. Each user story in this section describes a specific goal encompassing the researcher's goal and the functional gameplay.

User Stories

1. **US1:** As a player, I want to see the spheres disappear when both players interact with them simultaneously, so that I understand when an interaction is successful.
2. **US2:** As a player, I want to see the same sphere position as my teammate in real time, so we can collaborate accurately during the game
3. **US3:** As the researcher, I want to have access to a log of all the simulated latency data for analysis after the experiment
4. **US4:** As a player, I want the spheres to bounce and react realistically to the room's physical environment, so interactions feel natural and immersive.
5. **US5:** As a player, I want a tutorial to understand how to interact with virtual objects in the game, so I am comfortable with the controls before starting.
6. **US6:** As the researcher, I want the players to experience a predetermined amount of latency at random times during gameplay, so the research captures how latency affects their experience.
7. **US7:** As a player, I want any captured data to be collected anonymously, so my privacy is respected.

8. **US8:** As a player, I want each sphere to have a unique colour so I can coordinate with the other player to touch a specific sphere at the same time

3.2.2.1 Acceptance Criteria

User Story 1: **As a player,** I want to see the spheres disappear when both players interact with them simultaneously, so that I understand when an interaction is successful.

Scenario: When both players simultaneously touch the same sphere, the system registers the event and makes the sphere disappear.

- **Given** two players in the game,
- **When** both players touch the same sphere simultaneously,
- **Then** the sphere should disappear, indicating a successful interaction.

User Story 2: As a player, I want to see the same sphere position as my teammate in real time, so we can collaborate accurately during the game
Scenario: Before gameplay begins, there is a calibration phase in which the play spaces of the two players are aligned.

- **Given** two players in the game,
- **When** a sphere moves in the environment,
- **Then** both players should see the sphere in the same position in real time.

User Story 3: As the researcher, I want to have access to a log of all the simulated latency data for analysis after the experiment
Scenario: The system logs latency data and timestamps, storing them for analysis after the game session.

- **Given** an experiment with simulated latency,
- **When** latency data is generated during gameplay,
- **Then** the system should log all latency data, making it accessible for post-experiment analysis by the researcher.

User Story 4: As a player, I want the spheres to bounce and react realistically to the room's physical environment, so interactions feel natural and immersive.
Scenario: The spheres should exhibit realistic physics, bouncing when they hit walls or the floor.

- **Given** spheres moving in a simulated room,
- **When** spheres bounce off walls or interact with physical boundaries,
- **Then** they should react and move realistically, in line with the room's physical properties, for a natural and immersive experience.

User Story 5: As a player, I want a tutorial to understand how to interact with virtual objects in the game, so I am comfortable with the controls before starting.
Scenario: Before the game starts, the system provides a quick tutorial that explains how to interact with spheres and basic controls.

- **Given** a player starting the game,
- **When** the player enters the tutorial,
- **Then** the game should guide them in learning how to interact with virtual objects, ensuring they are comfortable with controls before beginning.

User Story 6: As the researcher, I want the players to experience latency at random times during gameplay, so the research captures how latency affects their experience.

Scenario: Before the game starts, the system provides a quick tutorial that explains how to interact with spheres and basic controls.

- **Given** players participating in gameplay with simulated latency,
- **When** different levels of latency are introduced,
- **Then** players should experience varying latency effects to capture data on how latency impacts their experience.

User Story 7: As a player, I want any captured data to be collected anonymously, so my privacy is respected.
Scenario: The system does not collect any personal information and ensures compliance with ethical standards for data privacy.

- **Given** data collection during gameplay,
- **When** the system captures player data,
- **Then** all data should be collected anonymously, ensuring player privacy is respected.

User Story 8: As a player, I want each sphere to have a unique colour so I can coordinate with the other player to touch a specific sphere at the same time
Scenario: The system ensures each spawned sphere has a unique colour.

- **Given** multiple spheres in the game environment,
- **When** the players observe the spheres,
- **Then** each sphere should have a unique colour, allowing players to easily identify and coordinate touching specific spheres together.

3.3 Risk Analysis

The development of the applications involves several potential risks that can affect the quality and usability of the systems. Therefore, in this section multiple of these risks in addition to an insight towards mitigating their said consequences are stated.

3.3.1 Identified Risks

3.3.1.1 Cybersickness

Cybersickness is a well-known issue in immersive environments such as virtual reality and augmented reality. VR-related sickness is primarily caused by the dissonance between real-world body movement and virtual stimuli, which can cause overloading in the brain (Pinchuk et al., 2020). Cybersickness can reduce the psycho-physiological resistance of the user, leading to nausea, dizziness, and discomfort.

- **Mitigation**: In order to minimise cybersickness, the system should maintain consistent frame rates and low latency. However, since our research investigates the effects of latency on cybersickness, latency will fluctuate in the CLXR-G. To reduce the extent of the effects these fluctuations bring forth, it is kept below 110 ms. Additionally, we will provide users with regular breaks and offer additional guidance in case of extreme effects.

3.3.1.2 Physical Discomfort

Extended use of MR devices can cause discomfort due to improper ergonomics or repetitive gestures. Prolonged interaction in a standing or static position can also cause neck or eye strain. Adequate device positioning and supervised familiarisation with the MR device are crucial to preventing discomfort (Cometti et al., 2018).

- **Mitigation**: Encourage proper posture through initial instructions, recommend frequent breaks, and ensure that devices are properly fitted to the user to reduce strain. Familiarisation sessions with the device will be supervised to ensure autonomy and safety.

3.3.1.3 Cognitive Overload

XR experiences can overwhelm users with too many stimuli or complex instructions, leading to confusion or frustration. This has been observed in studies where high cognitive demands negatively impacted user performance (Tran et al., 2023).

- **Mitigation**: The system will simplify the interface, limit the number of simultaneous tasks, and provide clear and concise instructions to minimise the cognitive load. Directive cues will be introduced to help participants recover from errors without exacerbating cognitive overload.

3.3.1.4 Real-World Hazards

Given the physical aspect of XR, there is a risk that users may bump into real-world objects or trip over obstacles.

- **Mitigation**: The play area will be clearly defined, with virtual boundary warnings to alert users if they approach real-world hazards. Supervision will also be encouraged, particularly during activities that require significant movement or engagement with the environment, to ensure safety (Cometti et al., 2018).

3.3.1.5 Privacy and Data Security

As the system may log user data during interactions, it is essential to handle this information securely.

- **Mitigation**: The project will follow strict data privacy protocols, including anonymization of logged data, secure storage, and compliance with GDPR standards (European Union, 2016).

3.3.1.6 Inconsistent Player Alignment

This is a risk specific to the CLXR-G, as the same scene is shared between two users. While shared between users, there can be a misalignment of the virtual objects between them, which can result in a non-coherent shared experience.

- **Mitigation**: With the use of linear algebra transformations both users will be aligned accurately within the shared virtual space.

4 Process

This section will detail a high level overview of the process for completing this project. Specifically, this will include the software and hardware used, team communication methods, and client communication.

4.1 Tools

To complete this project, a combination of software and hardware tools was employed, each serving a specific function in development, collaboration, and testing. Unity was utilised as the primary platform for creating and fine-tuning the application, while Unity Version Control ensured smooth collaboration and code management. The HoloLens 2 and Meta Quest 3 hardware enabled immersive testing and validation of the application correctness. For project management, tools like Trello and Discord supported effective communication and task tracking, enabling the team to stay aligned throughout the development process.

4.1.1 Game Engine

The development of our application required a platform for creating Extended Reality (XR) applications along with enabling realistic physics to be simulated in such environments. There are a wide range of platforms, and in our case we required game engines for developing applications like ours. The two most popular game engines are Unity and Unreal Engine. Unity was developed by Unity Technologies as a cross-platform game engine that supports development for desktop, mobile, console, augmented and virtual reality platforms. Similarly Unreal Engine is a computer graphics game engine developed by Epic Games that supports the same platforms as Unity, however it specialises in photo realistic visual effects (VFX) and particle simulations. We ultimately decided to use Unity despite its graphics capabilities being less refined, firstly because a few of the members already had experience developing with Unity and also because the primary programming language is C# which is most similar to Java, which we all have experience with. This contrasts with Unreal which offers development in C++ or visual scripting which we all have little to no experience using.

4.1.2 Version Control

Version control is essential for software development and collaborative projects as it manages and tracks changes in code and related files over the development lifecycle. It allows multiple developers to work on the same project simultaneously without overwriting each other's work. Additionally, team members can work on different parts of the project in parallel, where they have assigned themselves tasks that relate to the implementation of specific features. There are many version control tools available, the most popular being Git. However, when deciding on what version control tool to use, we came across Unity's own version control tool, called Unity Version Control (UVC). This tool is designed specifically for game development and is optimised for large binary files and is well suited for large binary assets such as textures and models. It also includes merging tools for Unity scenes and prefabs making it easier to handle changes without complex merge conflicts. Given that Git is a general purpose version control while UVC is tailored for applications like ours, we decided that UVC was the better option.

4.1.1 Hololens 2

Given the nature of our project, we naturally require an XR headset. The Hololens 2 is a Microsoft developed and manufactured mixed reality head-mounted display. Microsoft developed a framework called the Mixed Reality Tool Kit (MRTK) that allows developers to create mixed reality applications for their devices such as the Hololens and Hololens 2. MRTK contains pre-built UI components, interaction gestures, tools for hand tracking, spatial awareness and object manipulation. These components and tools make it easier for developers to create mixed reality applications. In addition, we also used the open source API OpenXR developed by Khronos. The OpenXR API defines an interface between the device and the developer tools which allows developers to write the same code that can be ported to different devices, instead of writing device tailored code.

4.1.2 Meta Quest 3

For reasons we will explain later, we also used the Meta Quest 3 which is Meta's version of the Microsoft Hololens. The device is a virtual reality (VR) headset that comes with its own line of software tools and frameworks developed by Meta to allow developers to create mixed reality applications. For development with the Meta Quest 3, we required the Meta XR All-in-One SDK, Oculus XR Plugin, and Meta Quest Developer Hub. The Meta SDK and the Oculus XR Plugin provide code samples, custom components, prefabs and more to allow developers to write software for Meta headsets. The Developer Hub is used for managing all the release builds and quickly uploading a build to the Meta Cloud so that all permitted users can have instant access to the application on their device.

4.1.3 Trello

Many large software development projects typically require the division of tasks and their status amongst the developers. After defining all the project requirements and objectives it may become difficult to meet them if they are not broken down into manageable tasks that we can execute. Therefore we used Trello which is a widely known web-based, kanban-style, list making application allowing us to manage tasks and place them into different categories that we define. In our case we used “Backlog”, “Todo”, “Doing”, and “Done” to keep track of tasks status.

4.1.4 Discord

With regards to communication, it was decided that Discord was to be used. This conference software allowed us to send messages, send diagrams, resources, meeting points (and times), and make calls. Most conference tools like Zoom and Microsoft Teams allow for the same or similar messaging features, however Discord also allows us to create different conference channels and messaging channels dedicated to specific discussions such as “Meeting”, “Resources”, “Report”, and “General”. On top of these features, it turned out that we all had the most experience using Discord, meaning we did not have to learn to use other conference and meeting tools.

4.2 Communication

Our high level objective is to deliver a product that meets the requirements of the client. Therefore it is imperative to have regular meetings with not only the team but also our client. In the first week we came to the agreement with the client that we would have weekly meetings on Thursday at 11:00. During these meetings, we would ask questions pertaining to the required features of the product, clarifying requirements, discussing issues encountered, and a progress discussion. These meetings were essential as they allowed us to ensure that we were on the right track and whether or not our progress was sufficient for that week.

Within the team, we decided to meet every day from Monday - Friday at 10:00 am- 17:00 pm to work on the project. If a given team member could not arrive on time, or work that day, they would have to let all the other team members know and provide a sufficient reason, such as being ill. It was ensured that every team member knew the status of every other member so as to minimise any miscommunication and mismanagement relating to the project development.

5 Implementation Approaches

For both the STM-T and the CLXR-G, we evaluated and implemented a range of approaches to meet stakeholder requirements. Our solutions focused on enhancing user experience, refining the user interface, optimising game networking, ensuring precise game alignment, and improving spatial awareness.

5.1 Short Term Memory Experiment

Throughout the STM-T design and implementation phases, several initial design choices were reconsidered based on testing results and feedback:

- **Radial View Implementation:** Initially, we planned to implement the game objects with a radial view component, which ensures whatever game object has this will follow the user’s field of view as they move their head. However, we ultimately kept

the spheres in a static position since the moving spheres frequently collided with other game objects, such as the Next Round button, as illustrated in Figure 22.

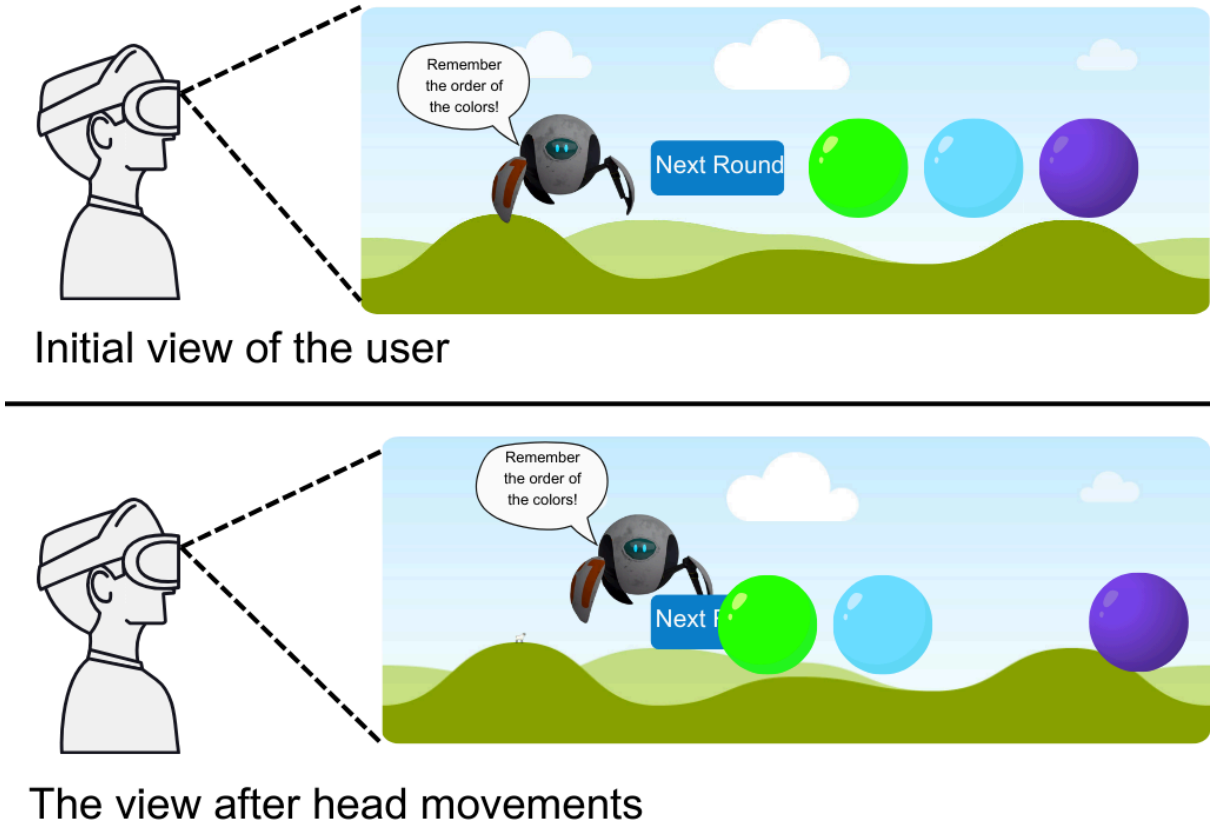


Figure 22: Inconsistency of game objects location with Radial View

- **Avatar Design:** Another planned feature was using a Japanese anime character as the avatar. We aimed to incorporate this character with free animations to increase the interactivity of the game, with reactions such as clapping when the user achieved high accuracy or crying when accuracy was low. However, after a peer review session, we replaced this avatar with a Robot Sphere (see Roku in Figure 8), as feedback indicated that the anime character may not be appropriate for a research setting.
- **Colour Selection Mechanism:** The initial game design required users to click on a sphere until they found their desired colour, repeating this for each sphere. Stakeholders expressed concerns that this design was time-consuming and lacked user-friendliness. To address this, we created three alternative versions that allow users to:
    - Choose colours via a hand menu (Figure 8).
    - Click directly from the colour tray (Figure 21).
    - Drag-and-drop from the colour tray (Figure 21).

  It's important to note that changing the colour tray type (Requirements 3 and 4) requires the researcher to modify the setting in the StateManager game object and redeploy the application. This means that users cannot switch between tray types within the application itself, rather it has to be adjusted on a development platform such as Unity. These functionalities have been made after initial feedback from our first iteration of the colour selection functionality.
- **Colour Set Reduction:** We initially had made 27 different colours available to be randomly chosen from upon the sphere's generation, however it was found that certain colours, such as white and beige, were too similar within the application and

could cause confusion. Therefore, we reduced the set to 9 distinct colours to improve usability and reduce cognitive load.

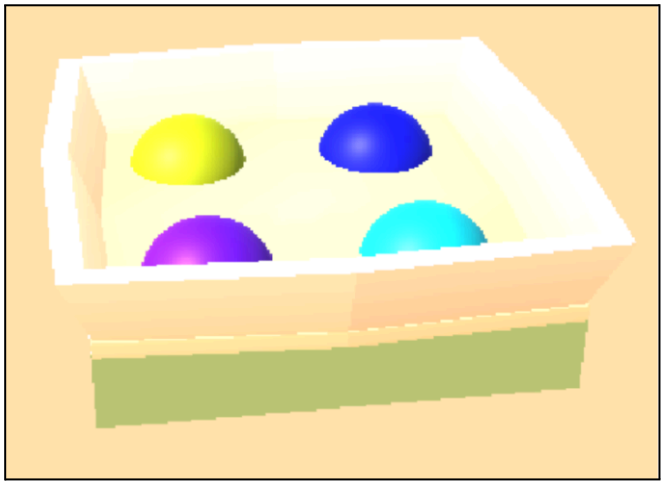


Figure 21: Tray type 0 and 1

- **Alignment of Game Elements:** Finally, the positioning of the Start button, text elements, spheres, and the Next Round button were initially misaligned. In our final version, we manually tested and aligned each component to ensure an optimal user experience.

5.2 Co-located XR Game

Development of the CLXR-G required in-depth research into the current networking solutions available, and the respective Meta and Microsoft solutions for spatial awareness and alignment. We also developed our own solution for spatial alignment to overcome some of the issues and setbacks experienced while using the ones developed by Meta and Microsoft.

5.2.1 Networking Solution

Choosing a networking solution is one of the most critical aspects of developing a co-located, collaborative mixed reality game. We have selected PUN 2 as our networking solution. Below, we provide a detailed explanation of why PUN 2 was chosen and how it fits within our specific project needs.

5.2.1.1 Chosen Solution

Initially, our project began with development targeted for HoloLens 2, using Microsoft's MRTK alongside PUN 2 as the networking solution. PUN2 is a well-established, widely-used real-time networking solution for Unity that enables low-latency, scalable multiplayer applications. When we switched to the Meta Quest 3 platform due to the discontinuation of the support for shared spatial anchors resources of Microsoft, we chose to adapt our existing networking implementation to support the Meta Quest Pro and Quest 3 platforms rather than switching to a new networking solution. This decision allowed us to minimise time spent rewriting and reconfiguring networking logic. This saved significant development time and reduced the learning curve for adapting to a new platform, and

allowed us to focus on the core gameplay mechanics and the experimental requirements of our supervisor/stakeholder.

5.2.1.2 Alternative Solutions

While PUN2 was chosen for its suitability, other networking solutions are available for Unity-based multiplayer development. In the following section, we compare some alternatives and justify our preference for PUN2.

5.2.1.2.1 Unity Multiplayer (Netcode for GameObjects)

Unity's new "Netcode for GameObjects" aims to replace the deprecated UNet directly. It is integrated with Unity and offers good performance and ease of use. Developed by Unity itself, this solution is deeply integrated into the Unity Editor and takes advantage of Unity's features, making it highly accessible for developers already familiar with the Unity ecosystem (Chandra, 2024). It also provides tools for real-time simulation and testing within the editor. However, this solution is still evolving. It lacks established cloud support and has fewer features than Photon (e.g., no matchmaking services). Thus, we would have had to set up our own back-end server infrastructure or rely on third-party providers. We needed a mature, stable, and feature-rich solution that would allow us to focus on the game rather than managing backend services. PUN2 has a well-documented SDK, reliable cloud infrastructure, and support for features like room management and synchronisation better suit our requirements.

5.2.1.2.2 Mirror

Mirror is an open-source networking library that builds on the legacy of Unity's UNet. It is designed to provide a reliable and efficient solution for small- to medium-scale games, focusing on simplicity and direct control over networking. It is highly customizable and offers great flexibility. Since it's open-source, it gives developers a high level of control over how networking logic is implemented. Mirror also integrates well with Unity's physics and game object architecture, making it very intuitive for Unity developers. However, unlike PUN2, Mirror does not provide built-in matchmaking, relay services, or cloud-hosted features (Mirror, 2020). This means the burden of managing network infrastructure would fall entirely on us, requiring server maintenance and additional time and resources, aspects that we did not have given the time constraints and limited experience of our team. Due to these reasons, Mirror did not meet our needs as well as Photon. Photon's cloud services significantly simplify setting up, scaling, and maintaining the networking infrastructure, allowing us to focus on game development.

5.2.1.3 Justification of Chosen Solution

We initially began developing for the HoloLens 2 with PUN2, and transitioning to the Meta Quest 3 allowed us to keep this core element intact. Maintaining the same networking solution significantly reduced the effort required for rewrites and reconfiguration. Our existing knowledge of PUN2 meant that the team did not have to spend additional time learning a new API or changing our networking logic, which was crucial given the project's time constraints.

PUN2 provides reliable cloud services, including matchmaking, room management, and relay capabilities. These features allowed us to create a seamless co-located multiplayer experience without the need to self-manage server instances or deal with complex networking configurations. The scalability offered by Photon ensures that we can add more features or extend the number of players in future iterations without being constrained by our infrastructure. PUN2 integrates smoothly with Unity, supporting both VR and AR applications. Its SDK is well-documented and widely used across the Unity developer community, making problem-solving easier through community support and available resources. PUN2 also provided robust synchronisation capabilities, which is critical

for maintaining a consistent game state in a co-located mixed reality experience, where players expect objects to appear identically across devices (Chandra, 2024).

Given that our supervisor's focus is on conducting experiments rather than managing backend infrastructure, Photon's managed cloud service was a significant advantage. It allowed us to meet our supervisor's reliability and robustness requirements without the additional overhead of server maintenance. Therefore, PUN 2 was the most suitable choice for our project's unique needs. It provided a reliable, scalable, and developer-friendly solution that facilitated our transition from the HoloLens 2 to the Meta Quest 3 and Quest Pro, supported the co-located multiplayer functionality, and aligned well with our team's existing expertise and project constraints.

5.2.2 Player alignment

In co-located mixed reality (MR) games, player alignment is crucial to ensure that all participants have a shared and synchronised experience of the virtual content in the physical environment. For players occupying the same physical space, each user must perceive virtual objects and interactions in the exact location in the real world. Without precise alignment, inconsistencies can break immersion, disrupt gameplay, and cause confusion or discomfort.

For example, if two players are expected to interact with the same virtual object, any misalignment in object positioning between their devices could make coordination impossible. Player alignment ensures that when one player reaches out to touch an object, the other sees the same event unfolding from their perspective. This consistency is fundamental to achieving an immersive and collaborative experience in MR.

Achieving accurate alignment involves aligning the playspaces of each player. This allows both players to clearly understand where virtual objects are located in relation to their shared physical environment. A key technology that facilitates this shared understanding is shared spatial anchors. In the following section, we describe what they are and our experience with using them.

5.2.2.1 Shared Spatial Anchors

Shared spatial anchors are a fundamental technology that aligns the virtual play space for multiple users in a co-located mixed-reality experience. A spatial anchor is a point in physical space tracked by the device and used as a fixed reference to position virtual objects. When multiple players use shared spatial anchors, they essentially share a common reference point that their devices can use to synchronise the placement of virtual content.

In practice, shared spatial anchors allow each user's device to understand where a particular virtual object should be positioned relative to the real-world environment. When an anchor is created by one device, the information about the anchor is shared with the other devices. This allows all players to see virtual elements in exactly the same place relative to the real-world context, even if they are using different hardware or starting from different positions (Pamistel, 2024). Using shared spatial anchors is particularly beneficial for maintaining the illusion of realism in a co-located MR environment. Ensuring that all users' views are anchored to the same reference points makes virtual interactions consistent across all devices (Meta Developers, 2024).

For example, when one player interacts with a shared object, spatial anchors ensure that the object appears to move or change in precisely the same way from the other player's perspective, regardless of slight differences in their physical position or device orientation. This is essential for synchronised actions, such as both players touching an object simultaneously.

Shared spatial anchors leverage each headset's spatial awareness capabilities to map the environment and share positional data across devices. Once an anchor is established, both devices use their respective sensors to recognize the environment, locate the anchor, and precisely align the virtual objects accordingly. This shared understanding

helps create a unified virtual experience for all participants, where everyone perceives the virtual environment in the same spatial context.

The importance of shared spatial anchors became evident in our project when we used the HoloLens 2, which heavily relied on this feature for player alignment. However, after switching to the Meta Quest 3 due to changes in Microsoft's services, we adapted our approach while maintaining the principles of accurate player alignment.

5.2.2.1.1 Meta Shared Spatial Anchors

Using and maintaining Shared Spatial Anchors (SSA) involve a number of steps, namely: Creating, Saving, Discovering, and Sharing. These steps are known as the anchor's lifecycle. The lifecycle starts with creating the spatial anchor, followed by saving the anchor to persist it across sessions. In later sessions, or when in a different space, the persisted anchors can be queried so that they can be found. For applications that involve multiple colocated users that share the same coordinated system, a given user can share a spatial anchor with all the other users to maintain a consistent perspective of the environment and alignment.

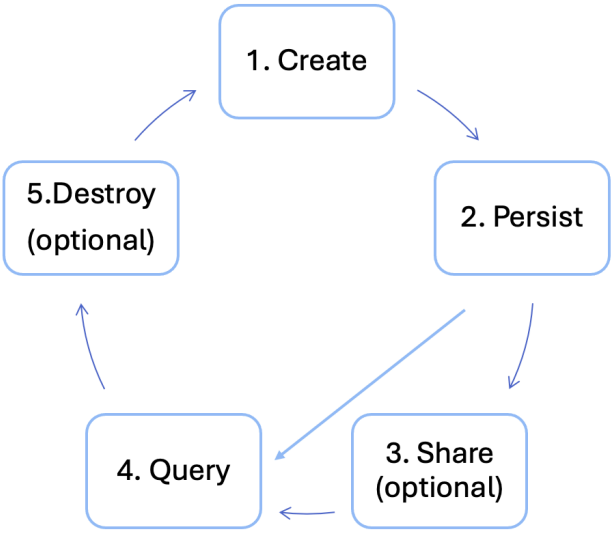


Figure 2: Figure of Shared Spatial Anchors

The implementation of SSA's can be summarised in a short list of steps. First the players must be in the same networked room, which can be achieved using any network solution, such as Photon. Photon provides functions for room creation, lobby joining, player management, and player network communication. The Meta SDK provides functions for creating spatial anchors using their custom made OVRSpatialAnchor component. After creation of the anchor, it may then be saved. Sharing a spatial anchor involves storing its point cloud data on Meta's servers along with which users are allowed to access it. The sharee is to specify which anchors are to be shared to whom they are to be shared with. After sharing the spatial anchors, the other users may load them so they can be rendered in their environment.

The anchor obtained after loading it is known as an Unbounded Spatial Anchor that needs to be associated (bound) with an OVRSpatialAnchor component. Before binding the Unbounded Spatial Anchors, it is required that the system localise each anchor. This process determines the anchor's position and orientation in the world, ensuring that the anchor is instantiated in the correct real world location that is aligned with all the other players. The Meta SDK provides functions for binding the anchors to this component.

However, the loading user must specify which anchors to be loaded, to do this knowing the anchor identification numbers (UUID) of each shared anchor is required. The sharee must therefore use the chosen network solution to share the UUID's with the permitted players.

5.2.2.2 Aligning to two known points

Another approach for aligning the virtual player spaces was to use linear algebra. Since the only objects in the scene are a number of spheres, we had to only ensure that their spawn points are the same in reality, meaning we could theoretically just translate and rotate the spheres in the virtual space of one player to align with the other. When the master client of the Photon room spawns the spheres at some point in the virtual space, they will be spawned in the same positions for the other client. However, since the virtual spaces of the master client and the client are not aligned, we thought that if we can find the exact rotation and translation offset between the two spaces, then we could take any point from the master client's virtual space and convert it into the corresponding aligned point in the client's virtual space.

Translation:

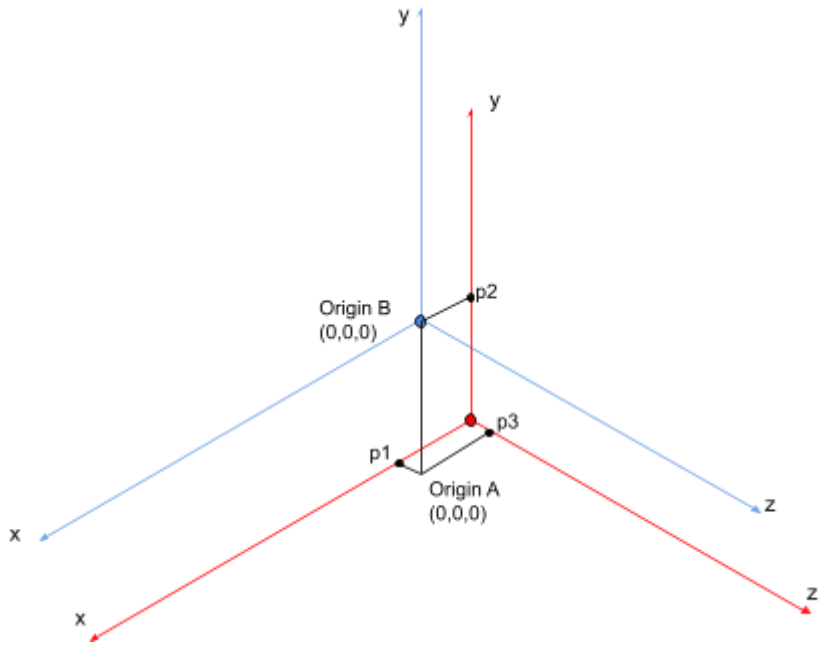


Figure 3: Alignment of two known points

In the case of translation, consider the above figure [NUMBER]. Suppose the master client's coordinate system is represented by A, and the other client is represented by B. Both have their own respective origin's at (0,0,0). However, the origin of B represented in A is given by $O_{B:A} = (p1, p2, p3)$, so to express any point (x, y, z) from A in space B, we just subtract $(p1, p2, p3)$ resulting in $(x - p1, y - p2, z - p3)$. To achieve this, both players follow a series of steps:

1) Player B instantiates a cube at the origin, denoted by O_B
2) Player B tells player A where O_B is relative to reality
3) Player A presses the right trigger of the controller at this point
4) Player A stores O_B relative to space A, denoted by $O_{B:A}$

Rotation:

To rotate a vector in A and express it in B, player B will instantiate an additional three cubes, one in the right (1,0,0), forward (0,0,1), and up (0,1,0) directions. These points are denoted by: $S_B = \{R_B, F_B, U_B\}$. Once again, player B will tell player A where these points are in the physical space they are in, so that player A can store them locally. The points that A stores are denoted by: $\{R_{B:A}, F_{B:A}, U_{B:A}\}$. The usefulness in these points lies in the fact that any set of basis vectors for a space S span that space, implying that one can generate any vector from the basis vectors. In this case, the basis vectors of space B are S_B, and the basis vectors that will take a point from A to B are $S_{B:A}$. Therefore, to express a point p_A in space B, given by p_B, we need to construct a transformation matrix of row vectors, $T = [R'_{B:A}\, U'_{B:A}\, F'_{B:A}]$ where $R'_{B:A} = R_{B:A} - O_{B:A}$ (same for other vectors). Thus we conclude that $p_B = Tp_A$, meaning p_B and p_A map to the same physical location but are different points in their local spaces (B and A).

5.2.3 Spatial Awareness

Spatial awareness is a critical aspect of any MR experience, especially in a co-located gaming where virtual elements must align seamlessly with the real world to maintain immersion. In this project, we had started with the Microsoft Hololens 2 as our target device for the co-located MR game. Then, have switched to the Quest 3 as the shared spatial anchors feature required for our purposes were discontinued for the Hololens 2. In this section we explain the spatial awareness capabilities of both devices and why Meta Quest 3's capabilities are much better for our purposes. During the development phase, we noticed several issues with the spatial awareness capabilities of the device. Sometimes gaps would get created in the spatial mesh that is generated by the device when looking around the room. Causing some objects adhering to physics in the environment to fall through gaps and exit the expected play area of the user which is the room that they are in. This issue would break the illusion of the spheres adhering to real world physics as expected. An example of such a gap created in the mesh generated by Hololens 2 can be seen in Figure 4, in such a case, the virtual sphere would fall through the hole in the floor of the generated model if it had physics enabled.

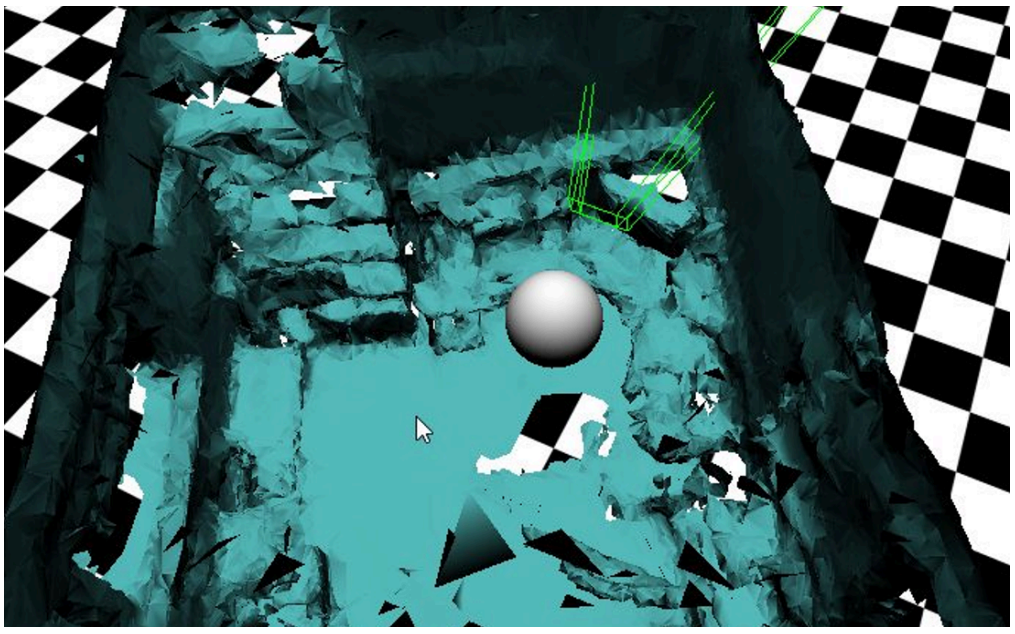


Figure 4: Screenshot of a 3D Model of a Room Generated by the Hololens 2's Spatial Awareness Capabilities (El Bruno, 2017)

In contrast, the Meta Quest 3 allows you to pre-scan your environment prior to playing a game. Ensuring that in a mixed-reality setting, the virtual objects with physics enabled always act as expected when colliding with the walls or the floor. The device relies on a video see-through system to provide users with a representation of their surroundings based on what cameras on the front of the device see (Meta Developers, 2024). This video-based passthrough feature is effective for displaying the real environment, however, it brings forth issues such as colour distortion and visual noise, particularly in settings without sufficient lighting. Although this limitation is overshadowed by a much more immersive gaming experience. Pre-scanning the room the users are in allows the virtual spheres the users are interacting with to bounce much more realistically off the floor during gameplay.

5.2.4 Sphere Logic

In addition to player alignment, it is required for their experience to be as realistic as possible. This means that the spheres instantiated must exhibit realistic physics that simulates the way they would behave in the real world. Furthermore, it is also required that the players be able to realistically interact with the spheres, namely being able to pick them up, throw them, move them around, and push them. However if the players both interact with a given sphere at the same time, the ball "pops" out of existence and disappears.

5.2.4.1 Interaction

Firstly, it is required that any given sphere be interactable. Fortunately, Meta SDK provides BuildingBlocks for a number of basic XR functionalities, such as hand tracking, controller tracking, grab interaction, networked objects, spatial audio, and alot more. For an object to be interactable, it suffices to add the "HandGrab" BuildingBlock as a child gameobject of the object itself. The BuildingBlock contains a "Grabbable" component, "Hand Grab Interactable" component and a "Grab Interactable", which all work together to add the interaction functionality. These components also allow developers to make adjustments to how a player is to interact with the object, for example specifying the hand gestures that can be used to engage with the sphere, changing the physics on the object when it is selected, and throwing it when unselected.

That was physical interaction, however it is required that there also be network interaction, which handles the way a sphere is to behave when being interacted with by two players. Photon Unity Networking implements a way for players to interact with networked objects. Specifically, it relies on the notion of "ownership", only one player at any given time can own an object, meaning no other player can manipulate the object unless ownership is transferred. Each sphere has to have a PhotonView component attached to it to enable the ownership handlers. By default, all the spheres are owned by the master client, the client who instantiated the spheres. We define two types of interaction, multi-user interaction, and single-user interaction. Single user interaction occurs when only one player is interacting with a sphere. For handling these interactions, each sphere contains a custom component we named "NetworkedSphere". When a player initiates an interaction with a sphere, our custom component will check the ownership status, if it belongs to the interacting player, then the physical interaction proceeds. Alternatively, when the interacting player does not own the sphere, then the component will make an ownership request that goes over the PhotonNetwork to the other player who automatically accepts the transfer.

With regards to handling multi-user interaction, the "NetworkedSphere" component also maintains the current number of players interacting with the sphere it is assigned to. Whenever this number equals two, then the sphere disappears, or "pops". Since this state management of the number of interacting players is local, we needed a way to synchronise

this information. This was achieved through Photon's Remote Procedure Calls (RPC), where whenever a player interacts with a sphere, a function that updates the current interaction state is called for both players. This function is automatically called for the non-interacting user, meaning it does not have to be invoked anywhere, and is only required to be defined.

5.2.4.2 Realistic Physics

Adding realistic physics is the other important part for ensuring an immersive collaborative environment. Unity being a 3D game engine implies that it is able to handle and simulate realistic physics using built in components. The first component is a "RigidBody" which allows developers to modify the mass, drag, and angular drag along with enabling gravity and specifying the type of collision mechanism (Discrete, Continuous, Continuous Dynamic, and Continuous Speculative). The table below details the use cases for each collision mode.

Collision Detection Mode	Useful for	Not useful for
Discrete	- Slow moving collisions.	- Fast moving collisions.
Continuous Speculative	- Fast moving collisions.	- Fast moving collisions that require a high degree of accuracy.
Continuous	- Fast moving linear collisions that require a high degree of accuracy. - Physics bodies that collide with static colliders.	- Collisions that happen as a result of physics body rotating. - Physics bodies that collide with moving colliders.
Continuous Dynamic	- Fast moving linear collisions that require a high degree of accuracy. - Physics bodies that collide with moving colliders	- Collisions that happen as a result of physics body rotating.

Table 7: Collision Detection

In our case, since the spheres collide with walls, the floor, and other spheres, it made sense to us to opt for the Continuous Dynamic collision mode which deals mainly with fast moving bodies that collide with each other.

The RigidBody component is what adds realistic physics to the object it is attached to. However, the RigidBody component does not specify anything regarding collision except for the collision type, which is why the "Sphere Collider" is required. This component adds a collider material around the object that defines how it is supposed to behave in the event of a collision. Additionally, the developer can specify properties such as the centre of the collider material and its radius, along with editing the collider material shape itself. Unity allows developers to create their own materials, which is what we did to allow the spheres to

bounce around the room. There are a number of properties that the material has, Dynamic Friction, Static Friction, Bounciness, Friction Combine, and Bounce Combine.

Property	Function
Dynamic Friction	The friction used when an object is moving. Value is in range [0,1]. Lower value will have less energy loss, whereas high value results in higher energy loss.
Static Friction	The friction used when an object is not moving. Value in range [0,1]. Lower value will allow the object to move easier, higher value will make it harder to move.
Bounciness	How bouncy is the object. Value in range [0,1]. Low value less bouncy, high value more bouncy. Higher value will also result in less energy lost.
Friction Combine	How the friction values are combined
- Average	Average two friction values
- Minimum	Take minimum of two friction values
- Maximum	Take maximum of two friction values
- Multiply	Multiply two friction values
Bounce Combine	How the bounciness of the two colliding objects are combined. Takes on the same modes as Friction Combine.

Table 8: Properties and their functions

We modified these values such that they allow the spheres to bounce and collide with the environment in an appropriate manner.

5.2.5 Latency Simulation

In order to simulate network latency within the game, we are using the PhotonPeer class from the Photon .NET Client API as we are already using PUN 2 as our networking solution (Photon .NET Client API 4.1.4.8) . This class provides a variety of methods that allow us to control the network simulation settings and simulate latency throughout the gameplay.

The property IsSimultionEnabled allows us to enable or disable network simulation for a PhotonPeer instance, which every user in the game will have. When this property is set to true, we can delay the network packets intentionally, which emulates the network latency. This is useful for our purposes as it allows us to replicate fluctuating network conditions that users will experience.

The NetworkSimulationSettings property provides access to detailed settings for configuring the latency characteristics. This includes defining parameters like the duration of the delay, packet loss percentage, and jitter. By adjusting these settings, we can simulate

various types of network impairments to study their effect on the user experience during gameplay.

We use the PhotonPeer to ensure that the latency settings are applied consistently across all connected players. By applying simulated delays when players interact with shared objects, we create realistic networking conditions that allow us to evaluate how latency impacts the synchronisation of in-game actions, such as touching or moving spheres. This approach helps us understand the player experience in situations where the network quality may not be ideal, providing valuable insights into how different levels of latency affect co-located multiplayer interactions.

5.2.6 Implementation Details and Setbacks

Over the course of the CLXR-G development, we encountered a number of setbacks, some of which severely hindered our progress and the number of requirements successfully met. The first setback relates to our usage and development with the Microsoft Hololens 2. It was discovered that Microsoft terminated its SSA services, meaning we were not able to develop our application using the Hololens2 and SSA's. We confirmed this directly by contacting a Microsoft employee (see Figure 1 in Appendix).

The second setback was that of Meta's SSA services. We made use of their SSA API documentation and managed to get creating, saving, and sharing of SA's working, however loading remained a problem. For this reason we came up with a more direct approach, which was to apply linear algebra to find the translation and rotation needed to switch between two virtual spaces, which in the end, proved to be successful.

The final setback relates to the user's ability to interact with the bouncing spheres. One of our functional requirements was to ensure that users are able to move around, pick up, and throw the spheres as if they were real (basketball, football, etc). While implementing these functionalities were not complex, sharing them over the network became problematic. A user who picks up a sphere changes its position when moving it around, meaning the transform data should be sent to the other user. While the data was being shared, the position of the sphere was not updating properly and resulting in the sphere making irregular and inappropriate movements. We therefore decided to omit these two functionalities as they would hinder the immersiveness of the game.

6 Global System Architecture

This section presents a comprehensive overview of the system architecture, offering both a high-level summary and detailed insights into each individual component of both the STM-T and CLXR-G systems.

6.1 Short Term Memory Test Architecture

In this section we present the global system architecture for the STM-T. We discuss the Unity hierarchy along with a system workflow detailing how the system is used, the relevant components, and how the system works for which diagrams are provided.

6.1.1 Overview

6.1.1.1 Unity Hierarchy

In Unity, the scene has a hierarchy that consists of a structured list of all the game objects within the scene which is arranged in a parent-child relationship. These relationships between game objects are important to consider as certain changes done to a child's (game object's) parent will also affect the child, such as transformations.

The STM-T application is made within one scene in Unity. This scene has pre generated game objects defined by Unity and the MRTK 2 toolkit we use. The Interaction Manager, handling the communication between interaction events triggered by hand movements. Directional light, which mimics sunlight, only comes from one direction. MixedRealityToolkit representing multiple settings for the environment, allows options such as enabling the diagnostics or adding extensions. AsyncCorountineRunner handles coroutine methods and async/wait operations. MixedRealitySceneContent is where most of our application is built. The main game objects it is constructed by are represented in Figure 5. Lastly, on the first hierarchy level in the CololerInteract Scene is the MixedRealityScenePlayspace where the main camera is located and determines the user's view.

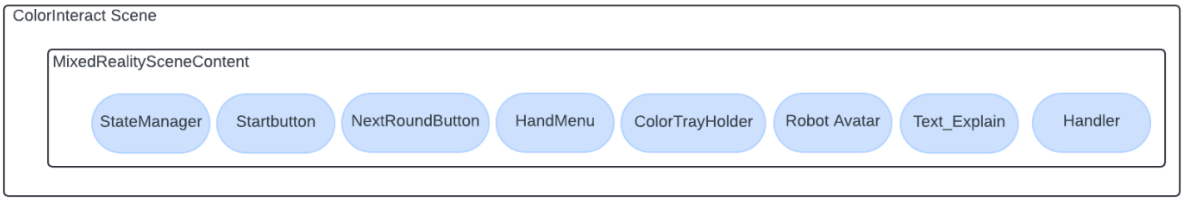


Figure 5: The game objects of MixedRealitySceneContent

Within the MixedRealitySceneContent defined by MRTK exists the StateManager object. The StateManager is a game object that consists of three scripts and an audio source with which it handles most functionality of the application. It is represented in Figure 6 where the blue nodes indicate the game objects, the green nodes the script components that contain the functions and red for any other component. The 'StateManager' script which activates/deactivates the spheres, checks the state of the game, handles the audio explanations with the audiosource component and acts as an in-between for communication for most other game objects. Its 'SphereGenerator' Script contains the methods and their functionality to generate the sphere objects and assign them a colour and location. 'BetterJson' is what handles all the measurements taken from the user doing the test, which at the end of the test converts the logged information into a Json file that is saved as a text file within the downloads folder of the Hololens 2.

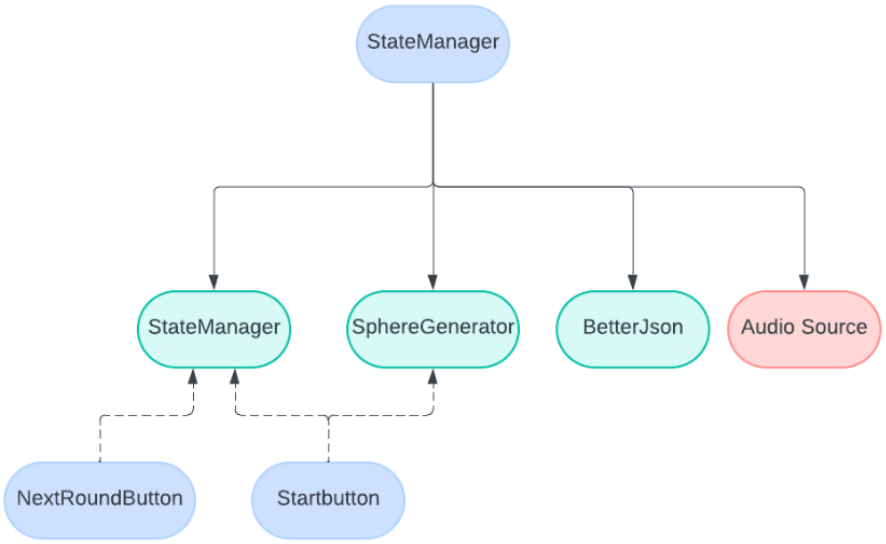


Figure 6: StateManager object hierarchy and button interaction

The start button is the object that initially communicates with the ‘SphereGenerator’ script to generate the spheres and tells the ‘StateManager’ script to change the type content that would appear when clicking the ‘help’ and ‘next round’ button on the handmenu. This is due to the initial functionality of clicking the handmenu buttons will provide an explanation towards their functions rather than providing their function. In any concurrent rounds the next round button will communicate with the ‘StateManager’ script (see Figure 6) to check if the next round can occur, which tells the ‘SphereGenerator’ to do so when the answer is yes. A ‘next round’ button also appears next to the spheres which provides the same functionality as the ‘next round’ button on the handmenu. The ‘help’ button will provide explanations to help a user.

The handmenu is an asset provided by the MRTK toolkit, it is a gameobject with multiple child gameobjects and scripts for several of them. The most important to note in regards to our application is the ‘buttoncollection’ object of which all its children are represented and visually visible by the user as a pressable button on the handmenu. From the start it contains a ‘help’ and ‘next round’ button, however if tray type two is selected, ColorSelectButton objects are created by the ‘StateManager’, these objects upon being pressed allow to give the generated spheres a colour when clicked.

Besides the handmenu, there is also a colorTrayHolder object that when the selected tray type is zero or one, is made visible with coloured spheres within, generated by its ‘GetColor’ script. When the type is zero, the user can click the coloured sphere and do the same to the spheres generated by ‘SphereGenerator’ to colour it. When one, the user can click and drag a coloured sphere through the blank spheres. The coloured spheres within the tray are generated and made as children by the colorTrayHolder. This tray is moveable by the user, however on occasion when a user does this in the next round the coloured spheres within the tray are aligned differently. A simplified overview of this is shown in Figure 7, where the tray type arrows indicate which objects and scripts will exist during the scene’s runtime if selected.

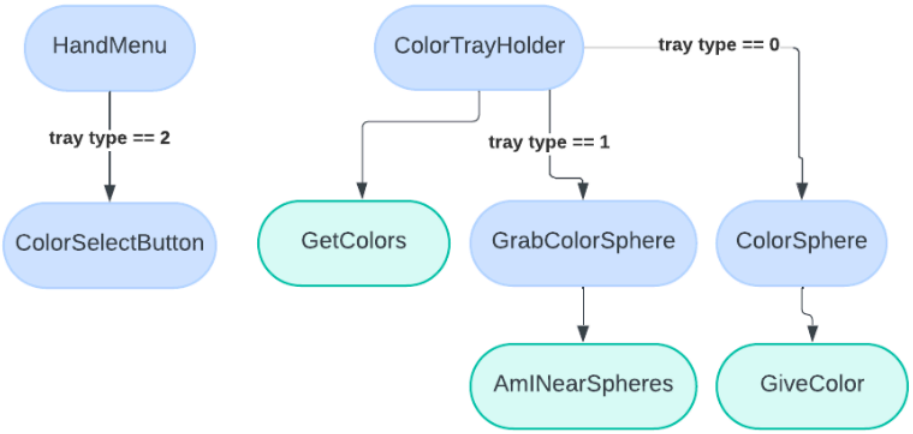


Figure 7: Tray type selection for STM-T

The sphere objects that are generated by the 'SphereGenerator' have the necessary components for colour to be rendered, collisions to be detected, etc, and a 'SphereInteractor' script. That script reacts to a user's click interaction or collision from a coloured sphere.

The robot avatar functions as an encouraging guide that acts upon certain events occurring. It has an animator component which allows the available animations to this object to be acted out.

The text explain object as the name gives indication towards, showcases all the textual explanations and information. Most of the textual information showcased by this object is determined through the 'StateManager'.

Handler is an object that contains the hand coaches that are dependent on the tray type. The hand coaches are objects that showcase which movements a user should make.

6.1.2 System Workflow

Design plays an important role in the digital experience of a user. To showcase the design and how this is displayed for the user, a step by step process of how a user would experience and go through the STM-T is portrayed. Additionally, a demo video of this has also been made.

Instructional Video:
The STM-T uses the Hololens 2 for participants. It is assumed that participants do not have any prior experience with this equipment, therefore before they are asked to wear the headset they are shown a video that shows the most basic hand movements needed for this experiment to interact with the 3D objects.

Menu explanation:
When wearing the headset they would be provided textual and auditory information about the handmenu (Figure 8), a menu that initially consists of a help and next round button. To ensure the participant understands how to see this menu a spectral representation (the palm up hand coach in Figure 8) of the action they need to perform is shown as additional visual information.

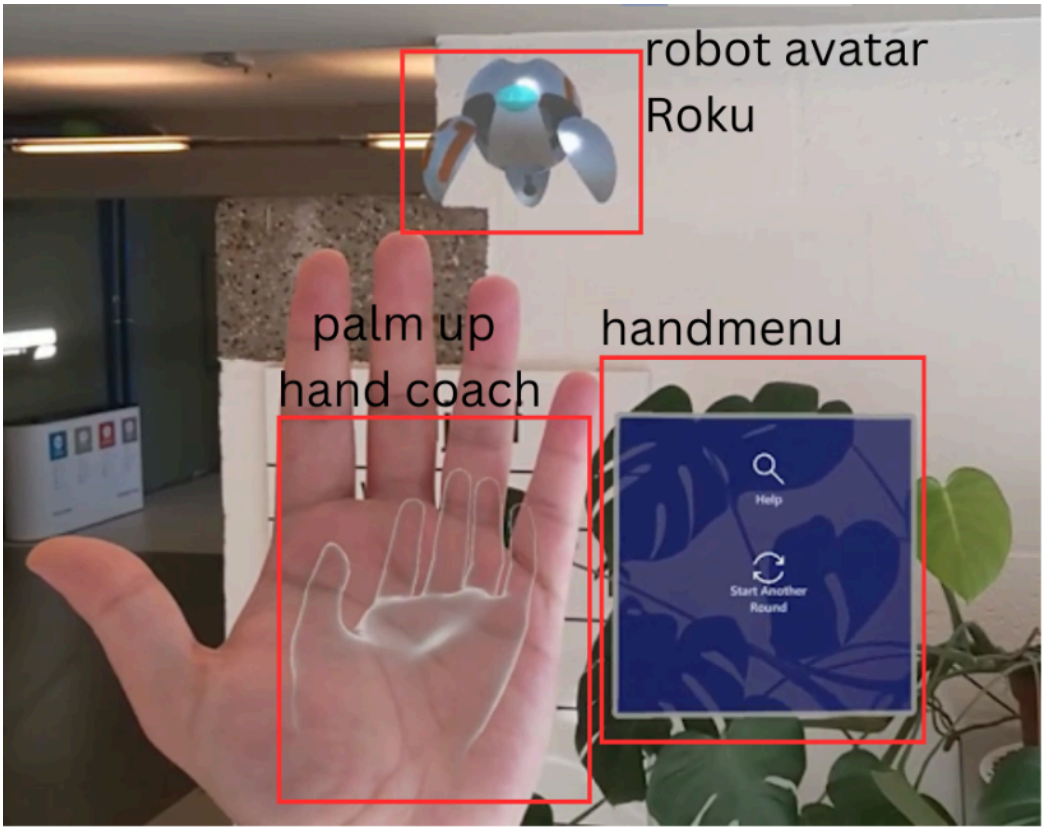


Figure 8: Spectral Palm, Hand Menu and Roku

Trial round:
Upon having interacted with these buttons the participant can start a trial round of the pre-experiment. This round is a simplified version of the actual experiment as only one coloured object will be generated (see Figure 9), with a thorough explanation for each step.

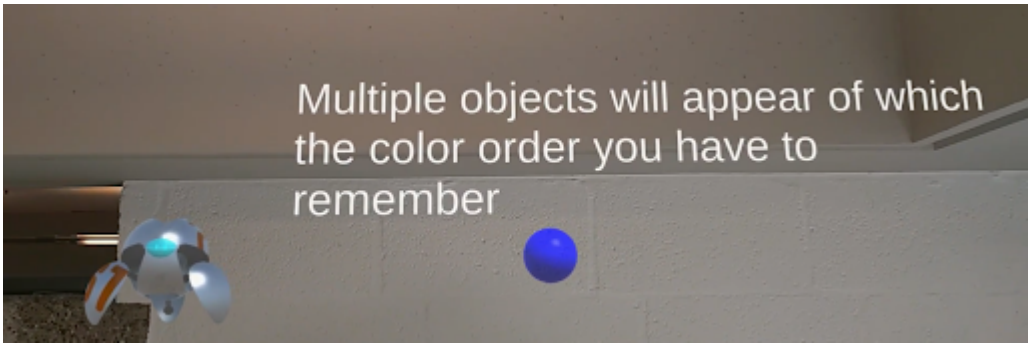


Figure 9: Explanation of experiment

Experimentation rounds:
In the actual experiment rounds, a minimum of four objects are generated to a maximum of nine. They appear for several seconds coloured until they revert to their default colour as represented in Figure 10, in which the participant is then able to interact and colour them as seen in Figure 11. A 3D “next round” object is spawned next to these that performs the same functionality of the button on the handmenu. When the participant coloured all objects they can continue to the next round. If they achieved an accuracy of 50% or higher in identifying the correct colour order, an additional object is generated the next round.

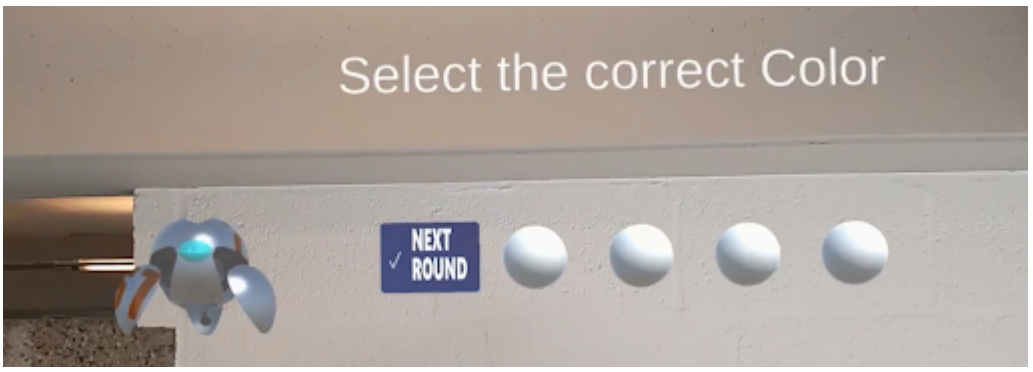


Figure 10: Uncoloured spheres (objects)

Figure 11: User colouring spheres (objects)

Experiment end:
A text message informing the participant that they finished and the recorded data of their accuracy score and time taken is saved as a json text file for the researcher within the headset's downloads folder.

6.1.2.1 Final Diagrams

The final diagrams presented in this section illustrate the structured logic and flow for the STM-T phase. The initial diagrams, and how the game evolved can be found in section 10.2.

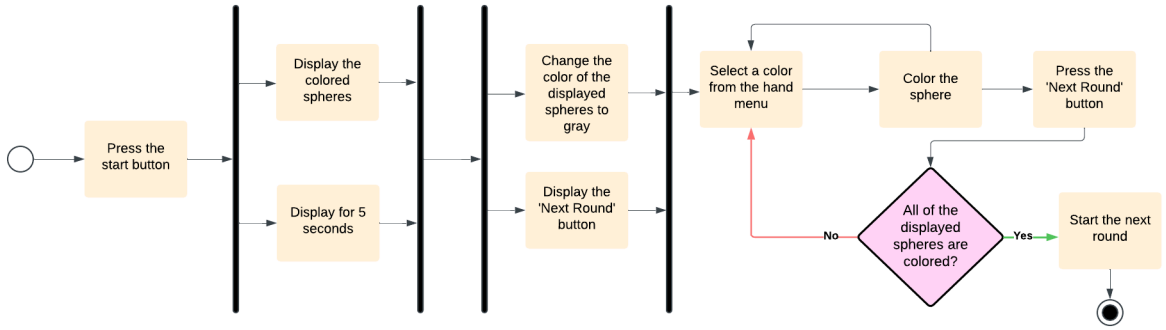


Figure 12: User colouring spheres (objects) in STM-T

Figure 12 shows the flow for a single round in the STM-T, focusing on the sequence of colour memorization and interaction. This process is designed to test participants' short-term memory retention while interacting with virtual objects in a controlled setting.

- **Colour Memory and Interaction:** Initially, the coloured spheres are displayed for 5 seconds, allowing users to memorise the sequence of colours. During this period, a

text prompt instructs the user to remember the colour order, reinforcing the memory task aspect of the experiment.

- **Guided Interaction:** After the spheres turn grey, participants use the hand menu to select and apply colours to the spheres, matching the original colour sequence. Roku and Hand Coach provide additional guidance during the trial round, which consists of colouring only one sphere, helping participants maintain focus and perform the task accurately. It should be noted that sound effects were integrated into all interactive elements during the STM-T phase to enhance user comprehension of the interaction.
- **Progression Logic:** The decision point in the diagram checks if all spheres are correctly recoloured before allowing progression to the next round. This check ensures the accuracy of task completion, which is critical for evaluating memory performance under virtual conditions

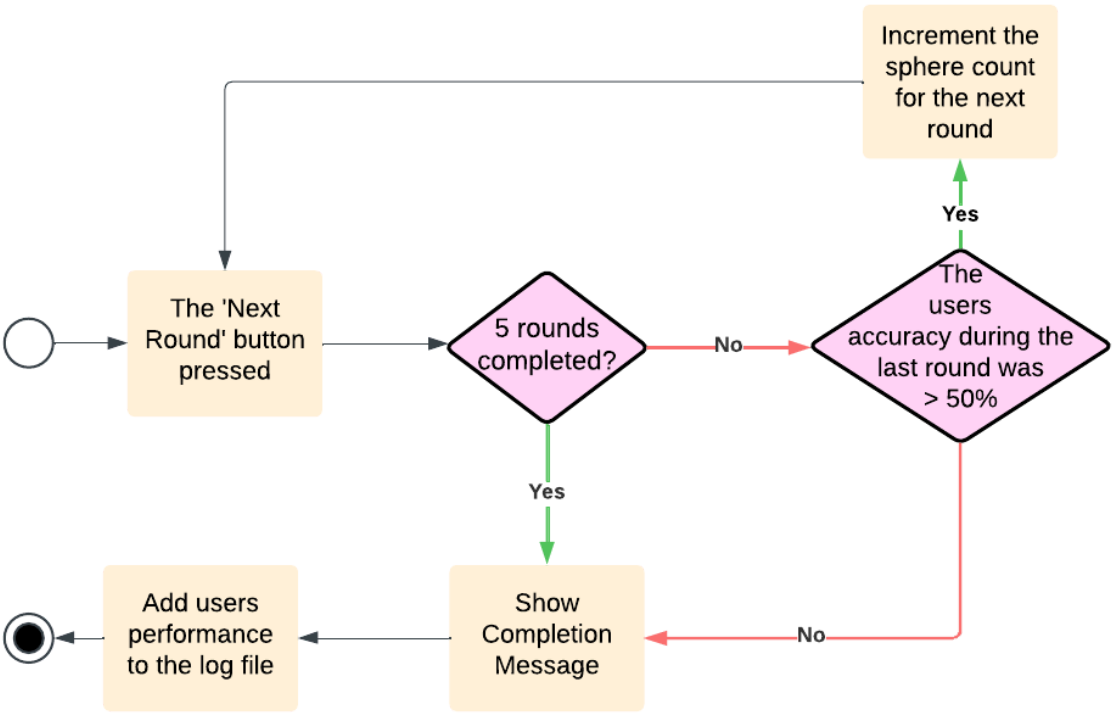


Figure 13: STM-T flow of the game

Figure 13 provides the overall structure of the game rounds in the STM-T. This phase adapts dynamically to participants' performance, adjusting difficulty based on accuracy to investigate the potential impact of increased cognitive load on cybersickness symptoms.

- **Adaptive Difficulty:** The decision node evaluates whether the participant's accuracy in each round exceeds 50%. If so, the number of spheres in the subsequent round increases by one, thus incrementally raising the cognitive challenge. This mechanism allows researchers to observe whether increased task difficulty correlates with changes in memory performance and cybersickness levels.
- **Looping Structure:** The experiment consists of five rounds and a trial round, each offering participants an opportunity to engage in repetitive memory tasks under gradually increasing complexity. If a participant's accuracy remains below the threshold, the number of spheres does not increase, allowing them to continue with a consistent level of challenge.
- **Completion and Data Logging:** After all rounds are completed, the system logs performance data—such as accuracy and time taken—alongside any observed symptoms of cybersickness. This data, saved in JSON and text formats, provides a

basis for analysing the relationship between cognitive load, memory performance, and cybersickness.

6.1.2.2 Initial & Final Design Choices

From our initial requirements and project plan an instructional video was not included to be in the final delivery. However, after conducting some initial user testing we found that providing as many possible measures to reduce the learning curve at adapting to the Hololens would allow the participants to have a much more likeable and usable application. This would allow the experiment itself to then occur much more fluidly when the users interact with the virtual objects in the actual short-term memory test portion of the STM-T and the later occurring latency affected experiment.

To extend from this point, during this testing and refining process we had decided that our initial idea of creating a chamber-like room where users are able to interact and learn the motions needed to use the hololens and interact with the virtual objects could be improved on. While the initial idea would allow users to become familiarised with the motions needed for interacting with virtual objects they would not be familiarised with the actual test. Therefore in its stead a simplified version of our actual short-term memory test would be made to prepare and familiarise the participant better. Through this when the participant would conduct the actual short-term memory test, the process and experience would be comparatively better as in the trial round a thorough step-by-step explanation would be given and tried out, allowing any issues that could occur during the test to occur in the initial trial round. Additionally, throughout the trial round and even in the actual rounds there is now guidance included through the use of spectral hands that perform the motions needed to interact with the virtual objects. This is a feature that can be implemented from the resources the MRTK tool provides, which we weren't aware of at the start of using this toolkit.

From all the design changes the actual testing part has undergone the most changes each iteration. The game objects that are generated for showcasing a colour order and to be coloured by the participants were initially cubes, these were after consideration converted to represent a spherical form to remain similar to the objects used in the latency affected experiment. This choice was made to keep as many factors as familiar as possible for the participant throughout the whole experience of the STM-T and the actual experiment. Menu functionality also experienced considerable changes where initial buttons were solely made as 3D objects nearby the participant and no existence of a help button. A handmenu that can be summoned by the participant was added and all buttons were changed to fit the same style, which the handmenu also upholds. These improvements towards consistency and betterment in UI design were all made for an increased positive user experience. For this in the final design all virtual objects to be interacted with during the experiment are generated nearby participants upon opening the application to allow near interaction, which is considered a more intuitive and easier form of interaction to achieve with any virtual object the Hololens creates. With these possible interactions the participant could colour the generated spheres. This was initially achieved by clicking a sphere and through that action it would switch to a different colour, whereas now they can achieve this by either clicking on a colour on their hand menu and then clicking the sphere. Another option existing separate to this allows them to grab a colour in the tray and drag it through the spheres. Which option is available to the user is determined by the researcher.

These design choices and all the iterative changes they underwent are expanded upon in section 5.1.

6.1.2.3 The GameObjects and their Functionality

Introduction:

All the existing virtual objects viewable by a user and their functionality is made by using Unity with C#. This section provides an informative view on the GameObjects created in Unity to establish this STM-T experience.

Asset use:
All assets were either obtained from MRTK 2 or the Unity asset store, references to the creators of the assets can be found in the appendix. The unity asset store is a library housing a large amount of assets created by those from Unity and users of Unity creating a large variety from 3D models, textures, templates and more. The MRTK 2 toolkit, is a software development toolkit provided by microsoft to help in developing MR and AR software that contains useful assets and scripts specifically for this purpose. All assets from the MRTK 2 toolkit are free and the Unity asset store contains free and paid assets.

Sound & voice:
While all virtually generated objects were made with unity or the MRTK 2 toolkit, some sounds were taken from a licence 0 website and the voice lines were generated with the use of a model on huggingface.com.

Statemanager:
An object created to manage most other objects and general flow of the experiment. It consists of three scripts, ‘StateManager’, ‘SphereGenerator’ and ‘BetterJson’. StateManager manages most functionality, activating and deactivating the spheres, the audio explanations as well changing the game’s states and more. SphereGenerator is the script related to the creation and rendering of the sphere objects and contains all the available colours supplied to them. BetterJson contains all the code that performs the logging and saving of the wanted user data.

Handmenu:
Handmenu did not initially exist as buttons generated virtually in real life performed the same functionality. However, it was determined to allow easy access to two essential buttons. ‘Help’ and ‘Next Round’. Help was a button created upon the first user testing cases that was not performed by any of the group’s members, whereas ‘Next Round’ existed from the start. The handmenu consists of assets taken from MRTK and modifying the buttons to fit our needs.

Buttons:
Initial buttons consisted of 3D cubes adapted to function as a button. In later iterations when looking through the assets, button assets provided by MRTK were used. Their actions trigger on being clicked or pressed.

Head Coach:
Handcoaches were not part of initial design as the idea of spectral hands was not thought of. However, upon finding them within the assets and the need to improve user experience they were added. Handcoach refers to the objects that consist of spectral hands indicating the necessary movement for a user to make during the experiment. The palm up facing head coach is attached to the camera (the view of the user) itself, all other hand coaches are kept under a handler object. The looks of the occurring head coaches depends on the tray type.

Tray:
Initially the selection of colour was achieved by clicking on the objects which would cycle through the available choices. Upon feedback, in later iterations three different options were implemented. A tray where a user clicks a colour within the tray and then clicks one of the objects. A tray where a user grabs the colour and drags it through the objects. A handmenu

(tray) where a user presses the colour on the handmenu and then clicks one of the objects. Pressing and clicking are different actions available to the Hololens 2 and are not considered interchangeable terms.

Sphere (the object):
The object being generated consisted of cubes that were changed to spheres to help in reducing unfamiliarity, as the actual experiment also uses spheres. The object has a 'SphereInteractor' script which functionality consists of detecting hand interactions and changing the objects colour.

Text:
All visual textual information is provided by this object.

Robot Avatar (Roku):
From initial human-like characters this avatar was adapted to a non-gender robot looking character deemed most appropriate to the research setting. This character named Roku was added to function as a mascot that gives encouragement to the user and to act as the guide with the explanations.

6.2 Co-located XR Experiment Architecture

6.2.1 Overview

The colocated application architecture is one that features three main high level components, the client, PUN, and the Photon Cloud.

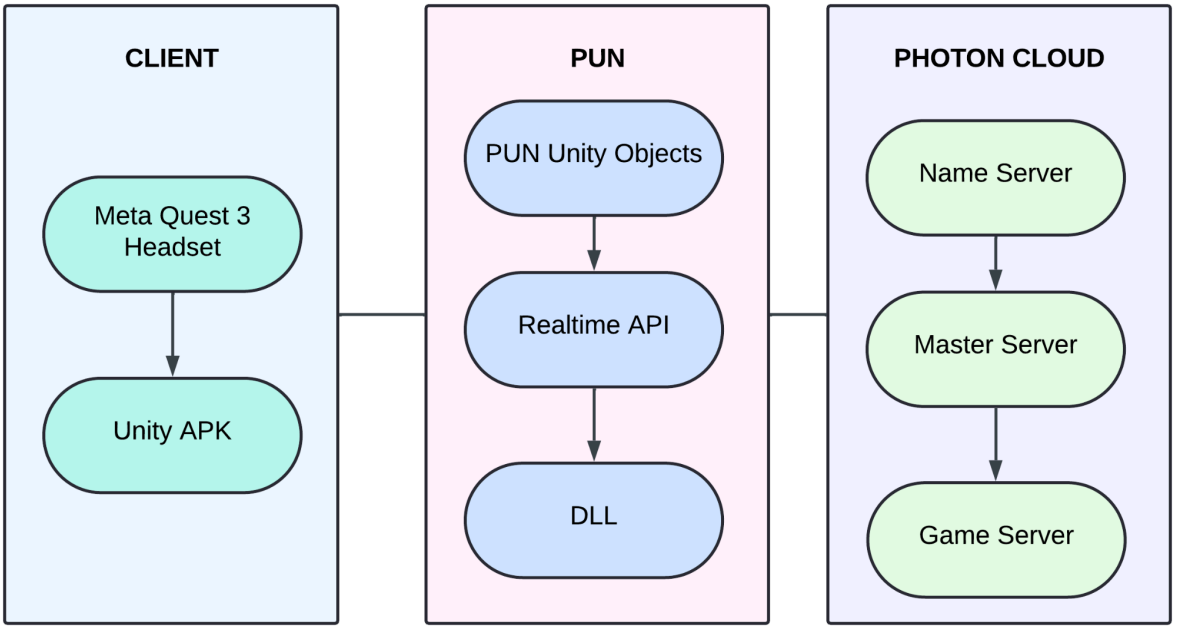


Figure 14: High Level System Architecture Diagram

The first component is the client which is broken down into two parts, the Meta Quest Headset and the Android Application Package (APK). The APK is obtained by building the application in the Unity Editor, this is the file that gets installed on the Meta Quest, allowing it to be executed. In order for players to be in the same colocated environment, PUN is used to connect them.

PUN provides high level Unity components that can be added to Unity gameobjects so that they can be instantiated on the Photon Cloud, synchronised, and run RPC's. PUN provides a middleware component known as the Realtime API which provides the high level network interface that is responsible for matchmaking, room creation, and callback functions. Finally the lowest level is made up of Dynamic Link Library (DLL) files — abstracted away from the user — which are responsible for handling processes related to converting data that can be transmitted over the network, packet reassembly, minimising network load, and the implementation of the network protocols, which define the rules for how data is sent and received between clients and servers. Photon's protocols handle reliability, ordering of messages, and packet delivery across the network, ensuring that messages arrive in the correct order and that lost messages are resent if needed.

The Photon Cloud is essentially a series of machines running the Photon server on them. When a user connects to the Photon server they get forwarded to what's called a "Name Server". The Name Server checks the application id, which gets provided in the Unity Editor before building the APK, and obtains region information. Then the client gets forwarded to the "Master Server", which is a hub for many regional servers. The Master Server knows all the rooms in a given region. When the client creates or joins a room, they get forwarded to one of the other machines called the "Game Server". The Games Server is where the application is hosted. It manages all updates for objects, RPCs (Remote Procedure Calls), events, and other data that need to be synchronised among players, ensuring that each client has the latest information about player positions, rotations, scaling and any other information to be synchronised. The Game Server is what facilitates real-time communication between players in the same room.

6.2.1.1 Unity Hierarchy

In Unity, the scene hierarchy is a structured list of all the game objects within a given scene that are arranged in a parent-child relationship, the organisation of which is essential because it dictates both how objects are rendered and how they interact. The order of parent objects does not affect the application, although it is useful to structure the hierarchy in terms of the types of gameobjects. However the order of the parent-child structures do matter, swapping a child for a parent will influence the behaviour of the application since child transforms are made to be relative to their parents.

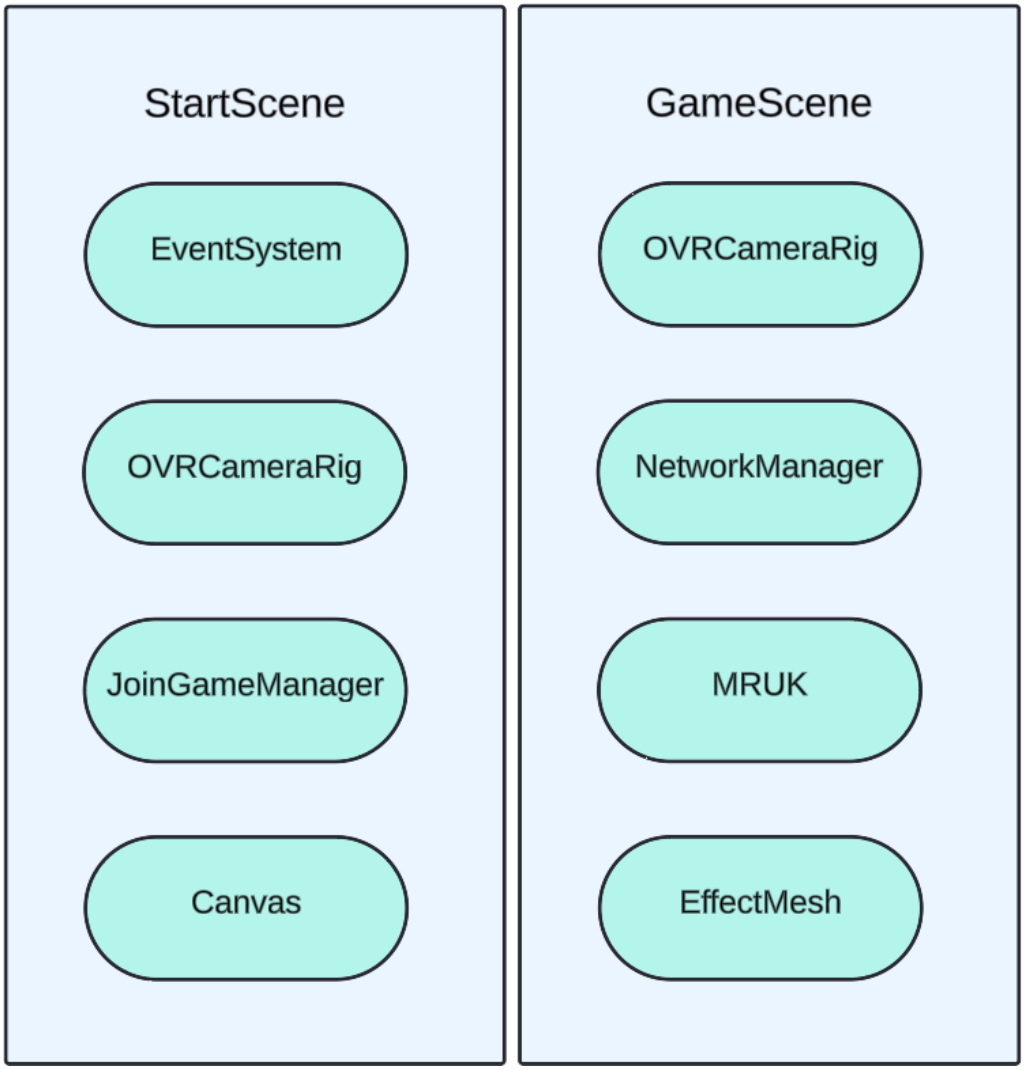


Figure 15: Start and game scenes of CLXR-G

In our application, we maintain two scenes, “StartScene” and “GameScene”. The StartScene is local and does not connect players yet. Instead it can be seen as a main menu, where a player can click on a button to join the game, at which point Photon matchmaking is initiated. The GameScene handles the player alignment and the game itself. The StartScene contains four gameobjects, the first two are defined by Unity, they are “EventSystem” and “OVRCameraRig”. The EventSystem manages input events and user interactions with UI elements. The input events include, but are not limited to, button clicks, pointer selections, and menu interactions from controllers, hands, and raycasters. In our case, we used this component to allow users to interact with the canvas which contains a button to join the game.

The OVRCameraRig is a specialised camera setup provided by the Oculus Integration SDK for VR. It contains cameras and objects for each eye and tracking anchors for positional tracking (eyes and hands), allowing the OVRCameraRig to track the headset, controller positions, and accurately capture head and hand movements.

The JoinGameManager component, made by us, is used to handle the captured event by the EventSystem when the user presses the join game button. The component makes use of Photon’s RealTimeAPI such that when the button is pressed, the user will be transferred to the Master Server lobby where the API will check if there are any existing rooms to join. If there is a room to join, then the user joins it, otherwise a room will be created. As explained

earlier, when a user joins a room, Photon will transfer the user from the Master Server to the Game Server.

Our second scene, the GameScene is hosted on the GameServer. This scene contains an identically structured OVRCameraRig that serves the same purpose as the one in the StartScene. To allow the bouncing spheres to interact (bounce off and collide) with the real world, the system requires the user to specify the walls of the room. The headset will guide the user through the process of selecting and defining the walls. The walls will be fixed throughout the course of the application and are persisted for later sessions. The Mixed Reality Utility Kit (MRUK) and the EffectMesh game objects are responsible for allowing this process to take place. MRUK allows the headset to query the scene within the real world, such as defining the floor level, scene boundary (physical space that renders the virtual content), and walls. The EffectMesh is responsible for adding a mesh to the scene objects specified, in this case, the walls will be wrapped in a mesh allowing for virtual objects to interact with the mesh. Specifically, the spheres will appear to be bouncing off the walls, but are in fact bouncing off the mesh.

The NetworkManager is the core of the GameScene. It is a custom made component that handles the instantiation of the balls, player alignment logic, and determines the number of rounds and when to start a new one. It has a photonview attached so that the component logic is shared between both the client and the master client and to allow the master client to invoke RPC's that the client can call.

6.2.2 System Workflow

The overall system workflow consists of a number of subsystems pertaining to networking, scene flow, latency simulation and sphere interaction.

6.2.2.1 Networking

Before players get forwarded to the Game server, the system needs to verify that their Oculus accounts have the required permissions to use the application. The Oculus SDK provides an API to check the accounts that the players are logged into on their respective headsets. It is also a requirement when using SSA's that the accounts are verified. If the player cannot be verified or fails to be verified, the application will force close, meaning they have to try again or ensure their account has the required permissions. A verified player will be forwarded to the Master Server in which the system will automatically check if there is a room to join or not. In the event that there is no room to join, a room will be created, otherwise they will simply join the already existing room. In either case, the player will be forwarded to the Game Server where the GameScene is hosted.

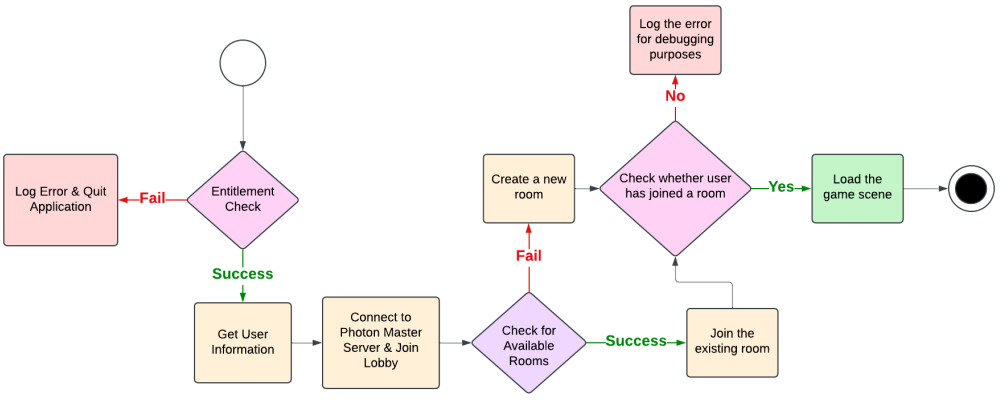


Figure 16: Flow Diagram for Joining a Room

6.2.2.2 Unity Scene Flow

Unity scenes provide a structured way to organise different phases or stages of an application. In multiplayer settings, they help ensure that both players have opened the application and joined the room before starting the game for either player as synchronisation between players is necessary for our scenario. Using scenes allows us to create isolated environments for specific tasks, such as waiting in a lobby. Managing scenes effectively ensures that all players experience the application in the intended sequence.

In our co-located game, the game is divided into two scenes, a Start Scene, and the Game Scene. When a user opens the game they start in the Start Scene, if no other games are detected in the same network, they are dedicated the Master Client and put forth into the Game Scene. When the second player opens the game while in the same network as the Master Client, the second player will be put through to the Game Scene as well and join the room created by the Master Client. Both players will then be in the Game Scene, after which the alignment phase will start. In this phase, the players select four physical points in the real world to align their virtual play spaces. This alignment involves the non-master client user transforming their virtual environment to match the master client's environment. Once the alignment is successfully completed, no Unity scene changes occur. Instead, the gameplayer phase begins within the same scene by spawning the interactive spheres. This approach minimises loading times and ensures the transformation information doesn't get overwritten by scene changes to keep the play spaces aligned. The described logic can be seen in the flow diagram below, showcasing the Unity Scene Flow of the game.

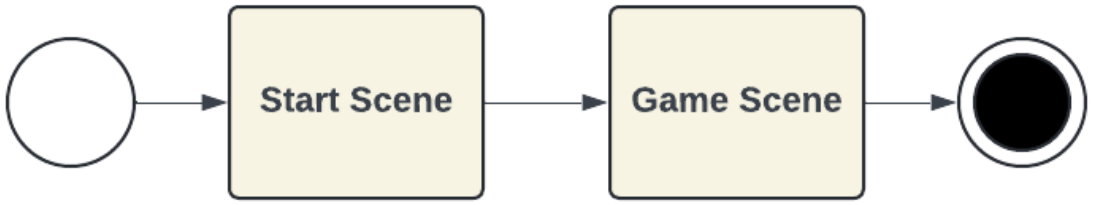


Figure 17: Flow Diagram showcasing Unity Scene Flow

6.2.2.3 Latency Simulation

The latency simulation logic is depicted in the flow diagram below, and demonstrates the systematic implementation of simulated network latency in the co-located mixed reality game. When a game session initiates, the Photon Peer instance responsible for managing network simulation is initialized. This enables latency simulation, allowing for a controlled introduction of network latency to each player. The core logic checks every 200 milliseconds to determine whether a 70 millisecond delay will be applied, based on a 50% probability. In our case, we generate a random number in between 0 and 100, if the generated number is greater than 50, the system simulates a network latency of 70 ms and logs it with the current timestamp. This process does not affect the co-player as each player has their own Photon Peer instance, which allows each participant to experience independent latency conditions.

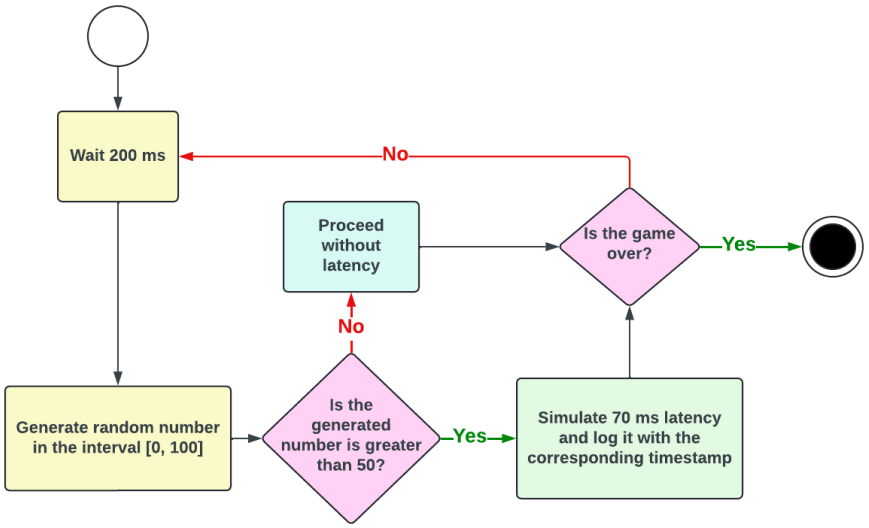


Figure 18: Flow Diagram for Latency Simulation Logic

Each instance of simulated latency is logged with precise timestamps, capturing the amount of simulated network latency that occurred for the player at a specific time. Each player will generate their own log file, so after the game, each player will have a log file that is different from each other. Having access to the amount of latency experienced at a specific time allows our supervisor to see the correlation between experienced latency and their cybersickness levels.

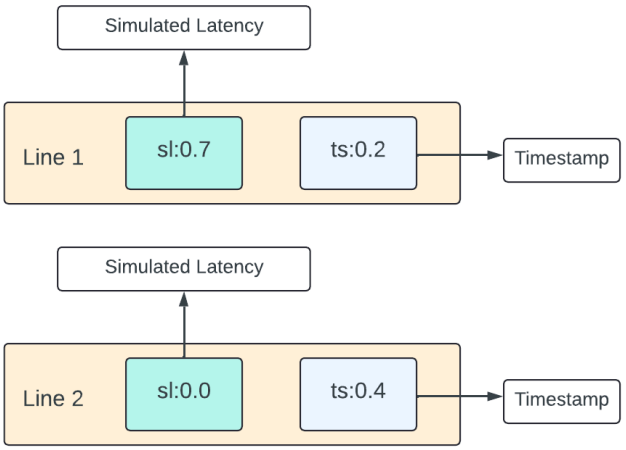


Figure 19: Contents of Log File for Simulated Latency Levels

6.2.2.4 Sphere Interaction

In order for the spheres to disappear upon simultaneous touches, we have to send some information whenever a user touches any sphere. When the sphere detects a touch by a player, it will call an RPC method to update the UserPressTimestamps dictionary. A new entry will be created in the dictionary with the userID as the key and the timestamp at which the time took place will be the value of the item. Each user does this action so that they have the same values as each other's dictionary at all times. After the dictionary is updated for both users, we check if the dictionary contains two items, and if the other entry has a different userID as it's key. If this is the case, that means the other user has touched the

sphere as well. We can get the time interval in between the timestamps of the entries to see if the users have simultaneously touched the same sphere. The time interval that we consider to be simultaneous is one second. So if the time interval in between the dictionary entries with differing keys is less than a second, we hide the sphere so that the users know they have successfully touched the sphere at the same time. The described logic is shown in the flow diagram below.

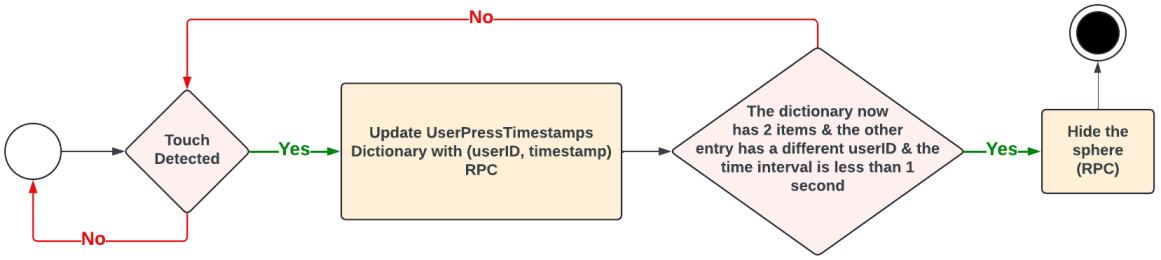


Figure 20: Flow Diagram showcasing Sphere Hiding Logic

7 Testing

Testing played a crucial role in ensuring the reliability and functionality of our project. We employed a comprehensive testing strategy that included unit tests, integration tests, and smoke testing. Unit tests were used to validate the correctness of individual components, ensuring they performed as expected in isolation. Integration tests were conducted to verify that different modules interacted seamlessly, maintaining consistency and functionality across the system. Smoke testing was used as a final checkpoint, quickly assessing the stability of the major application features and functionalities. This multi-layered approach allowed us to identify and address issues at various levels, ensuring a robust and high-quality end product.

7.1 Short-Term Memory Test

During the STM-T phase, we manually tested the implemented requirements. This testing was conducted within three different environments: simulation, holographic remoting and in the UWP app.

For simulation, no XR headset (Microsoft Hololens 2) was needed allowing us to conduct testing whenever this was unavailable. It also loads the application the quickest and does not suffer from connection problems, however not all actions available to a person wearing the headset are fully representative within this environment.

Holographic remoting allows previewing and debugging the application locally while using the headset. Testing was preferably done where one member of the team wore the headset, while the other adjusted the settings in Unity to ensure proper alignment of virtual objects with the real world. These adjustments were made in real time based on the feedback from the headset user. The focus in this environment was on the positioning and behaviour of interactive objects within the real life environment. Some issues that would occur were low frame rates or the connection between the Hololens 2 and the PC being broken, as it is connected through Wi-fi and uses the resources of the PC. Additionally, when starting up holographic remoting we found that the objects rendered in our application are rendered based on the location the Hololens was at the previous time holographic remoting occurred. Therefore, if there was a large difference in location between two instances of testing with holographic remoting, holographic remoting would have to be started, stopped and then started again for the correct position to be used.

Besides testing the functionality of the product, user testing was also conducted with a few volunteers. During this it was found that a stronger emphasis on teaching users how to use the headset was needed. Some common mistakes noted were moving solely the finger during near interaction or making quick imprecise movements unrecognisable to the Hololens for far interaction. Due to this it would be advised that anyone conducting experiments with this application has some basic proficiency in case further instruction remains needed for a participant.

Near the end of the development, UWP debugging occurred. A UWP app would be the final version of this application on the Hololens 2. For creating a UWP app the application had to be built and deployed to the headset while a user is logged in. For final user testing this app was built as a release version, which as a UWP app on the headset no connection errors or frame rate reduction occurred. For debugging file storage on the headset it had to be built as the debug version, as the other version does not support logging debug messages. This takes an extremely long time to build compared to all the other environments and reducing built time by reducing mesh optimization and other similar choices were performed to somewhat reduce this.

7.2 Co-located XR Experiment

Unit testing involves selecting a requirement and implementing it either by defining a single function, a series of functions, or a collection of gameobjects that interact with each other. This way we knew immediately if a given requirement was met in isolation. If the implementation of a requirement was too large to be described by a single function, it was broken up into single components that could at times be tested individually. A simple example was to allow users to join a room. This feature was defined simply by a script containing a series of Photon callback functions that would transfer users between the different Photon servers. A more complex example was implementing spatial alignment which was not implemented in a single function, and required the implementation and correctness of other features such as sending data using RPC’s, registering points when controller trigger is pressed, and transmitting transform data using Photon callback functions. In addition, a small function such as matrix multiplication was integral to the correctness of spatial alignment and was tested on its own in isolation. By ensuring that data transmission, point registration, and matrix multiplication worked, we were able to validate the correctness of the spatial alignment feature.

Validating the correctness of features that are made up of smaller components required integration testing. Unit testing only provides coverage for individual functions or smaller components in the system, we therefore used integration testing to test larger system features. Integration testing allowed multiple already tested units to combine in a larger component that are able to communicate and work together. For instance, the Photon callback function would invoke the matrix multiplication function and reference the variables that contain the stored controller points. Combining these smaller components would produce a larger one that we could then test to ensure that they all work correctly together and that operations are executed in the correct order. We used integration testing to verify whether the ball popping feature works. For this feature we had to ensure that ownership requests were fulfilled, since only one player can own a shared object, and only one player can interact with said object. In addition, we had to test the collision between the user's hand and the ball, the RPC that sends interaction data, and the function to make the sphere disappear. Overall, integration testing played an important part in ensuring that larger system components were functioning correctly.

Building up from integration testing was to ensure the correctness of multiple larger system features that combine to form basic functionalities of the application. This form of testing was to provide assurance that the major features of the system were working as expected. For example it was required that the application provide a realistic and immersive environment. Testing this required spatial alignment, realistic physics, and ball popping. Our tests involved two members putting on the headsets with the application installed and ensuring that these components together provided the larger functionality.

8 Evaluation

8.1 Short-Term Memory Experiment

8.1.1 MVP

The Minimum Viable Product (MVP) for the STM-T is the initial version of the game with fulfilled functional requirements. It provides users with an initial interactive experience. The user wears the Hololens 2 glasses, but the experiment does not start in front of the user, instead appearing in the location where the last user concluded the experiment. The user locates the "Let's Start the game" text and Start button to begin. Upon clicking Start, coloured balls appear for 2 seconds, and the user selects colours from the hand menu, applying them to the spheres by clicking on them. Successfully changed spheres visually respond by emitting a click sound and increasing in size.

There are three tray types that can be selected by researchers for the colour selection of the users. The tray type 0 provides the user with a tray filled with spheres that are coloured based on the colours revealed on the spheres. The user should first click on the coloured sphere followed by the sphere they want to colour. Similarly, tray type 1 provides the same tray as tray 0; this time, the user should drag and drop the coloured sphere into the sphere they want to colour. Finally, tray type 2 is the hand menu, offering the option to select colours, start another round, and a help button. The Help button is present in the hand menu, though it is Quality. An additional sphere appears when the user achieves over 50% accuracy, introducing an added challenge. If not all spheres are coloured, a message prompts the user to complete this task. The hand menu expands horizontally when the colour selection exceeds four colours. After five rounds, a message informs the user that they have completed the first part, with part two coming up.

8.1.1.1 Fulfilled Requirements

- **Requirements 1, 5, and 6** are fulfilled:
    - **Requirement 1:** The user initiates the hand menu by making the Palm Up gesture when guided by the Hand Coach.
    - **Requirement 5:** The user selects a colour from the hand menu and clicks on the sphere to change its colour when the tray type is set to 2.
    - **Requirement 6:** The user proceeds to the next round by pressing the "Start next round" button on the hand menu or clicking the button next to the spheres.
- **Requirement 7** is met, as achieving 50% accuracy unlocks additional spheres, challenging the user further.
- **Requirement 8** is partially met with sound and visual feedback upon colour changes, though specific text and voice guidance are limited.
- **Requirement 10** is met by providing a message signalling the end of the STM-T.
- **Requirements 13 and 15** are met:

    - **Requirement 13:** The buttons on the hand menu are clearly visible and easy to click when the tray type is set to 2.
    - **Requirement 15:** The Palm Up gesture detection is responsive with minimal delay, allowing smooth interaction.

8.1.1.2 Unmet or Partially Fulfilled Requirements

- **Requirement 2** is only partially implemented, as the Help button on the hand menu is currently Quality, and button functionalities have limited text and voice feedback.
- **Requirement 9**, which includes displaying the trial round start button with guiding text and voice, requires further refinement for users unfamiliar with MR headsets.
- **Requirement 14** is not met due to time constraints. The colours displayed on the spheres are not optimised to be differentiable for users with colour blindness.
- **Requirement 16** needs improvement. The initial placement of virtual objects (e.g., Start button) does not align ideally with real-world spatial orientation.
- **Requirement 18** is not fully met, as instructions are conveyed with limited guidance, missing clear voice or text-based assistance.
- **Requirement 17** was not formally tested. While no latency was observed during practical use, the system's latency was not measured to confirm it meets the 10-15 ms requirement for interaction responsiveness.

8.1.2 Final Product

In the final product, the user starts the application and is immediately greeted by the game character, Roku. Roku guides the user on how to open the hand menu by providing instructions both in text and voice. A Hand Coach is displayed to visually demonstrate the Palm Up gesture required to reveal the hand menu.

Once the hand menu is opened, the user is instructed to interact with two buttons: 'Help' and 'Start Another Round'. When the user clicks on these buttons, Roku explains their functionality through text and voice feedback, ensuring the user understands the purpose of each button.

After both buttons have been pressed, the Start button for the trial round appears. Upon pressing the Start button, a trial round begins with only one sphere displayed, along with a 'Next Round' button positioned next to the sphere. The user is guided on how to interact with the sphere using the Hand Coach.

The user selects a colour from the hand menu and then clicks on the sphere to apply the selected colour. Once the user successfully colours the sphere in the trial round, the Start button for the actual experiment appears.

The user then proceeds to play five rounds of the game. Each round starts with four spheres. If the user achieves more than 50% accuracy in a round, an additional sphere is added in the next round, increasing the challenge. This means that if the user maintains high accuracy, the final round will have up to nine spheres.

Upon completing the final round, a message is displayed to the user indicating that the second part of the experiment will begin soon.

8.1.2.1 Fulfilled Requirements

- **Requirements 1, 2, 3, 4, 5, 6, and 7** are fulfilled:
    - **Requirement 1:** The user makes the Palm Up gesture to reveal the hand menu when guided by the Hand Coach.

    - **Requirement 2:** The user clicks on the buttons in the hand menu, and Roku reveals their functionality through text and voice.
    - **Requirements 3, 4, and 5:** The system allows the researcher to set the tray type in the StateManager game object before deployment:
        - **Requirement 3:** When the tray type is set to 0, the user selects a colour from the colour tray and clicks on the sphere to change its colour.
        - **Requirement 4:** When the tray type is set to 1, the user selects a colour from the hand menu and drags and drops the coloured sphere into the sphere they want to colour.
        - **Requirement 5:** When the tray type is set to 2, the user selects a colour from the hand menu and clicks on the sphere to change its colour.

      Note: Changing the tray type requires the researcher to modify the setting before building the application; users cannot switch between tray types during runtime.

    - **Requirement 6:** The user presses the "Start next round" button on the hand menu or clicks the button next to the spheres to proceed to the next round.
    - **Requirement 7:** Achieving 50% accuracy or higher generates an additional sphere in the next round, increasing the challenge.
- **Requirements 8, 9, and 10** are fulfilled:
    - **Requirement 8:** The system reveals the functionality of hand menu buttons with text and voice once clicked, providing immediate feedback.
    - **Requirement 9:** The system displays the start button for the trial round and the actual experiment while guiding the user with text and voice.
    - **Requirement 10:** The system displays a message indicating that the real experiment will start after the STM-T concludes.
- **Requirements 11, 12, 13, 15, 16, and 18** are fulfilled:
    - **Requirement 11:** The coloured spheres on the colour tray are clearly visible and easy to click when the tray type is set to 0.
    - **Requirement 12:** The coloured spheres on the colour tray are clearly visible and easy to drag and drop when the tray type is set to 1.
    - **Requirement 13:** The buttons on the hand menu are clearly visible and easy to click when the tray type is set to 2.
    - **Requirement 15:** The Palm Up gesture detection is responsive with minimal delay, allowing smooth interaction.
    - **Requirement 16:** The system ensures that virtual objects are not placed within real objects, enhancing user immersion and safety.
    - **Requirement 18:** The robot avatar’s instructions are simple and understandable, aiding the user through the experiment.

8.1.2.2 Unmet or Partially Fulfilled Requirements

- **Requirement 14** is not met due to time constraints. The colours displayed on the spheres are not optimised to be differentiable for users with colour blindness, potentially affecting accessibility.
- **Requirement 17** was not formally tested. While no latency was observed during practical use, the system's latency was not measured to confirm it meets the 10-15 ms requirement for interaction responsiveness.

8.2 Co-located XR Experiment

8.2.1 Fulfilled Requirements

Requirements 1, 2, 3, 5, 6, 7, 8, 9, 12, 14, 15, 16 are fulfilled.

- **Requirement 1:** The system spawns 10 virtual spheres in random positions in the room the users are in when they are both in the same networked room.
- **Requirement 2:** The spheres disappear upon simultaneous touches by both players.
- **Requirement 3:** The spheres adhere to physics and bounce of the walls and the floor when contact is made.
- **Requirement 5:** The system simulates network latency.
- **Requirement 6:** The system simulates a network latency of 100 ms with a chance of 50% of occurring, which happens every 200 ms.
- **Requirement 7:** The system logs the simulated latency along with the timestamp at which it took place.
- **Requirement 8:** The system does not simulate latencies exceeding 100 ms.
- **Requirement 9:** The system features an initial calibration phase to align the two players' play spaces.
- **Requirement 12:** The system does not collect any personal data.
- **Requirement 14:** The system supports two concurrent users.
- **Requirement 15:** The system maintains synchronisation of the virtual objects in between users.
- **Requirement 16:** The system creates a room if no room is detected, and automatically enters a room if one is detected.

8.2.2 Unmet or Partially Fulfilled Requirements

Requirements 4, 10, 11, 13, 17 are either unmet or only partially fulfilled. These requirements were not met or partially fulfilled due to time constraints and setbacks caused by other more important requirements.

- **Requirement 4:** The system does not provide a tutorial before the start of the game for new users to familiarise themselves with the way of interacting with virtual objects using the Meta Quest 3.
- **Requirement 10:** The system does not consistently maintain 90 frames-per-second and does not always provide a smooth user experience.
- **Requirement 11:** Latency simulation may disrupt core game functionalities such as the synchronisation and alignment between spheres.
- **Requirement 13:** The system does not provide a tutorial therefore no instructions were defined.
- **Requirement 17:** The spheres cannot be interacted with beyond popping. Therefore no picking up or moving around of the spheres is supported.

9 Conclusion

The Short-Term Memory Test (STM-T) successfully achieved its primary objective of creating a game to test short-term memory, for researchers to use and possibly find a correlation between users' short-term memory and cybersickness in an XR environment. The implementation provided an engaging and interactive platform where participants could perform memory tasks while immersed in XR, allowing for valuable data collection on their short-term memory capabilities. The user interface and interaction mechanics were well-executed, with features like the virtual character Roku and the Hand Coach effectively guiding users through the experiment. However, there were areas that did not fully meet the

intended goals. The application did not optimise the colour palette for users with colour blindness, which could affect the accessibility of the game and potentially influence the accuracy of the collected data. Additionally, while latency issues were not observed during practical use, formal testing to confirm that the system meets the specified 10–15 ms latency requirement was not conducted. The limited functionality of the Help button and the need for more comprehensive voice and text guidance were also identified as aspects requiring improvement. For future work, it is recommended to enhance the application's accessibility by implementing colour schemes that are distinguishable for users with colour vision deficiencies, conduct formal latency measurement, and expand the functionality of the Help button.

The development of the CLXR-G aimed to create a collaborative and immersive experience that effectively simulates network latency for experimental research purposes. The game successfully met several critical requirements. It achieved real-time synchronisation between players, ensuring that virtual spheres appeared at the same location for both participants, thus supporting effective collaboration. The physics of the spheres, designed to mimic real-world interactions such as bouncing against walls and floors, added to the immersive quality of the experience, making interactions natural and intuitive. The system also ensured data logging, capturing timestamps and latency events during gameplay, which will be instrumental for later analysis. However, some aspects remained partially fulfilled. The process of maintaining a consistent spatial alignment across different devices posed challenges, leading to occasional discrepancies in virtual object positions experienced by the users. This inconsistency somewhat impacted the user experience by occasionally misaligning the shared virtual objects. Future work could look at further refinement in spatial alignment techniques to enhance the functionality and the robustness of the game, such as changing the four-point alignment method to a two-point alignment or using shared spatial anchors as a way to align the play spaces. Additionally, an onboarding process could be added to ensure the player is familiar with the controls of the system prior to participating in the game.

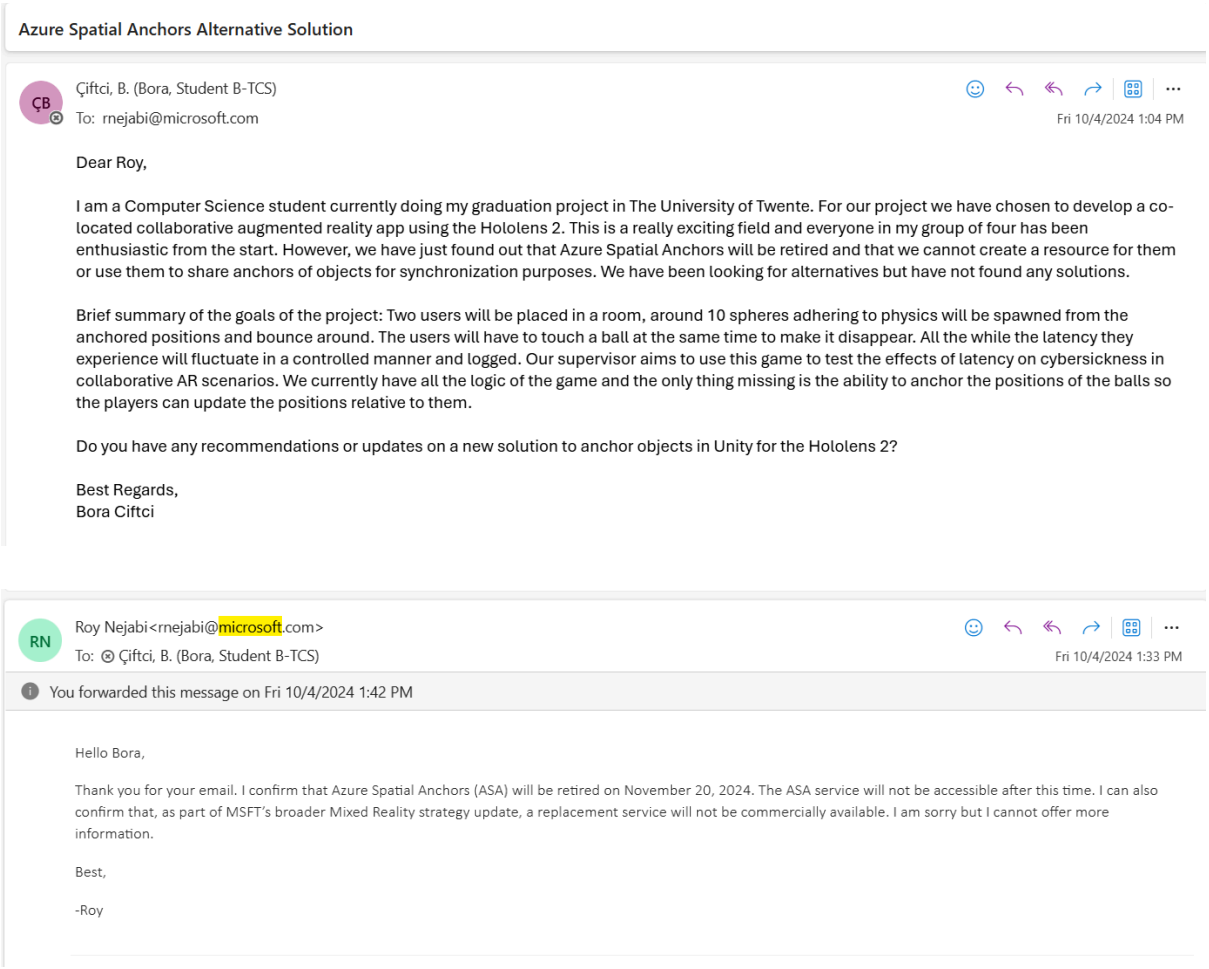
References

1. Caserman, P., Martinussen, M., & Göbel, S. (2019). Effects of end-to-end latency on user experience and performance in immersive virtual reality applications. In E. van der Spek, S. Göbel, E. L. Do, E. Clua, & J. Baalsrud Hauge (Eds.), Entertainment Computing and Serious Games. ICEC-JCSG 2019. Lecture Notes in Computer Science (Vol. 11863). Springer, Cham. https://doi.org/10.1007/978-3-030-34644-7_5
2. Chandra, R. L., Patrick Querl, Djamel Berkaoui, Koen Castermans, & Heribert Nacken. (2024). Comparison of Open-Source Networking Libraries for Unity Engine in Higher Education. Journal of Computer Science and Technology Studies, 6(3), 65–75. https://doi.org/10.32996/jcsts.2024.6.3.7
3. Cheng, L., Schreiner, M., & Kunz, A. (2024). Comparing tracking accuracy in standalone MR-HMDs: Apple Vision Pro, Hololens 2, Meta Quest 3, and Pico 4 Pro. In Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology (VRST '24) (Article 48, pp. 1–2). Association for Computing Machinery. https://doi.org/10.1145/3641825.3689518
4. Cometti, C., Païzis, C., Casteleira, A., Pons, G., & Babault, N. (2018). Effects of mixed reality head-mounted glasses during 90 minutes of mental and manual tasks on cognitive and physiological functions. PeerJ, 6, e5847. https://doi.org/10.7717/peerj.5847
5. Dahlman, J., Sjörs, A., Lindström, J., Ledin, T., & Falkmer, T. (2009). Performance and autonomic responses during motion sickness. Human Factors the Journal of the Human Factors and Ergonomics Society, 51(1), 56–66. https://doi.org/10.1177/0018720809332848
6. El Bruno (2017) #hololens – how to save a #3D model of the environment around the device, ElBruno.

https://elbruno.com/2017/07/10/hololens-howto-save-a-3d-model-of-the-environment-around-the-device/.
7. European Union. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, 119, 1–88. Retrieved from http://data.europa.eu/eli/reg/2016/679/oj
8. GASTRO: Sushi Food Pack FREE by Yaro.team | 3D Food | Unity Asset Store. (2023, November 18). Unity Asset Store. https://assetstore.unity.com/packages/3d/props/food/gastro-sushi-food-pack-free-by-yaro-team-266512
9. Ghoshal, M., Ong, J., Won, H., Koutsonikolas, D., & Yildirim, C. (2022). Co-located immersive gaming: A comparison between augmented and virtual reality. 2022 IEEE Conference on Games (CoG), 594–597. https://doi.org/10.1109/CoG51982.2022.9893708
10. Kyaw, N., Gu, M., Croft, E., & Cosgun, A. (2023). Comparing usability of augmented reality and virtual reality for creating virtual bounding boxes of real objects. Applied Sciences, 13(21), 11693. https://doi.org/10.3390/app132111693
11. Lawson, B., Proietti, P., & Burov, O. (n.d.). Guidelines for Mitigating Cybersickness in Virtual Reality Systems. In https://core.ac.uk/download/543485155.pdf.
12. Maria Torres Vega et al. 2020. Immersive interconnected virtual and augmented reality: a 5g and iot perspective. J. Netw. Syst. Manage, 28, 4, (Oct. 2020), 796–826. https://doi.org/10.1007/s10922-020-09545-w.
13. Meta Developers (2024) Shared Spatial Anchors. https://developers.meta.com/horizon/documentation/unity/unity-shared-spatial-anchors/?intern_source=devblog&intern_content=build-local-multiplayer-experiences-shared-spatial-anchors.
14. Mittelstaedt, J. M., Wacker, J., & Stelling, D. (2018). VR aftereffect and the relation of cybersickness and cognitive performance. Virtual Reality, 23(2), 143-154. https://doi.org/10.1007/s10055-018-0370-3
15. Mirror. 2020. A community replacement for Unity's abandoned UNET Networking System. github.com/vis2k/Mirror
16. Pamistel (2024) Azure spatial anchors overview - azure spatial anchors, Azure Spatial Anchors | Microsoft Learn.https://learn.microsoft.com/en-gb/azure/spatial-anchors/overview.
17. Parler-TTS - a Hugging Face Space by parler-tts. (n.d.). https://huggingface.co/spaces/parler-tts/parler_tts
18. Photon .NET Client API 4.1.4.8: ExitGames.Client.Photon.PhotonPeer Class Reference. https://doc-api.photonengine.com/en/dotnet/current/class_exit_games_1_1_client_1_1_photon_1_1_photon_peer.html.
19. Pinchuk, O., Burov, O., Ahadzhanova, S., Logvinenko, V., Dolgikh, Y., Kharchenko, T., Hlazunova, O., & Shabalin, A. (2020). VR in education: Ergonomic features and cybersickness. In S. Nazir, T. Ahram, & W. Karwowski (Eds.), Advances in Human Factors in Training, Education, and Learning Sciences (pp. 350–355). Springer. https://doi.org/10.1007/978-3-030-50896-8
20. Roku Sphere. (n.d.). Robot Sphere [3D character asset]. Unity Technologies. Retrieved from https://assetstore.unity.com/packages/3d/characters/robots/robot-sphere-136226
21. Tran, N., Grant, T., Phung, T., Hirshfield, L., Wickens, C., & Williams, T. (2023). Now look here! Mixed reality improves robot communication without cognitive overload. In J. Y. C. Chen & G. Fragomeni (Eds.), Virtual, Augmented and Mixed Reality (pp. 395–415). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-35634-6_28
22. Van Damme, S., Sameri, J., Schwarzmann, S., Wei, Q., Trivisonno, R., De Turck, F., & Torres Vega, M. (2024). Impact of latency on QoE, performance, and collaboration in interactive multi-user virtual reality. Applied Sciences, 14(6), 2290. https://doi.org/10.3390/app14062290
23. Waltemate, T., Senna, I., Hülsmann, F., Rohde, M., Kopp, S., Ernst, M., & Botsch, M. (2016). The impact of latency on perceptual judgments and motor performance in closed-loop interaction in virtual reality. In Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology (VRST '16) (pp. 27–35). Association for Computing Machinery. https://doi.org/10.1145/2993369.2993381

10 Appendix

10.1 E-mail conversation with Microsoft



Azure Spatial Anchors Alternative Solution

ÇB Çiftci, B. (Bora, Student B-TCS)
To: rnejabi@microsoft.com
Fri 10/4/2024 1:04 PM

Dear Roy,

I am a Computer Science student currently doing my graduation project in The University of Twente. For our project we have chosen to develop a co-located collaborative augmented reality app using the Hololens 2. This is a really exciting field and everyone in my group of four has been enthusiastic from the start. However, we have just found out that Azure Spatial Anchors will be retired and that we cannot create a resource for them or use them to share anchors of objects for synchronization purposes. We have been looking for alternatives but have not found any solutions.

Brief summary of the goals of the project: Two users will be placed in a room, around 10 spheres adhering to physics will be spawned from the anchored positions and bounce around. The users will have to touch a ball at the same time to make it disappear. All the while the latency they experience will fluctuate in a controlled manner and logged. Our supervisor aims to use this game to test the effects of latency on cybersickness in collaborative AR scenarios. We currently have all the logic of the game and the only thing missing is the ability to anchor the positions of the balls so the players can update the positions relative to them.

Do you have any recommendations or updates on a new solution to anchor objects in Unity for the Hololens 2?

Best Regards,
Bora Ciftci

RN Roy Nejabi<rnejabi@microsoft.com>
To: Çiftci, B. (Bora, Student B-TCS)
Fri 10/4/2024 1:33 PM

You forwarded this message on Fri 10/4/2024 1:42 PM

Hello Bora,

Thank you for your email. I confirm that Azure Spatial Anchors (ASA) will be retired on November 20, 2024. The ASA service will not be accessible after this time. I can also confirm that, as part of MSFT's broader Mixed Reality strategy update, a replacement service will not be commercially available. I am sorry but I cannot offer more information.

Best,

-Roy

Figure 1: E-mail Correspondance with Roy Nejabi about the Continuation of Shared Spatial Anchors

10.2 Initial Diagrams

The initial sequence diagrams illustrate the original structure and interaction flow of the short- term memory game. For clarity, the sequence has been divided into two parts: Part 1 details the setup phase, while Part 2 represents the repeated interactions throughout multiple game rounds. The following diagrams form the foundation for understanding the game mechanics and highlight key differences between the initial and final versions of the design.

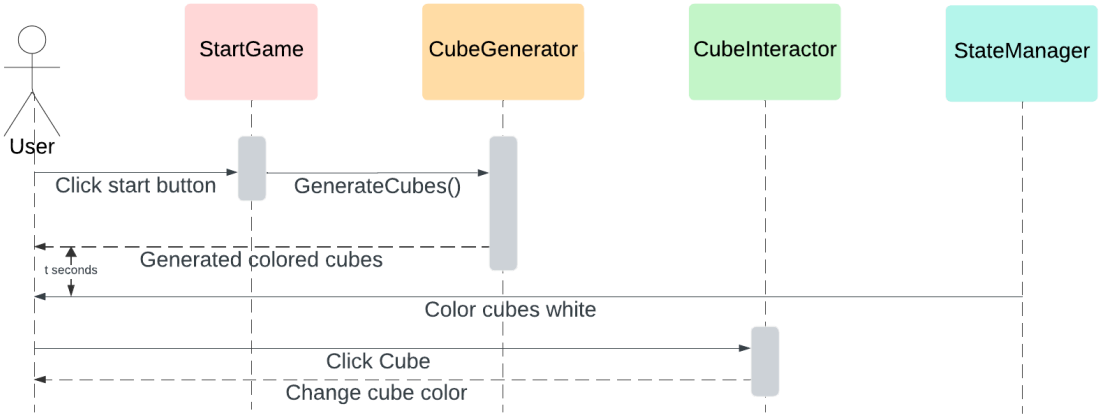


Figure 2: Initial Sequence Diagram (Part 1)

Figure 4.5 represents the setup phase of the short-term memory game, showing the initial steps leading up to the main loop of gameplay. The primary differences between this initial setup and the final version are as follows:

- **User colour Selection Mechanism**: In the initial design, the user repeatedly clicked on each cube until the desired colour was reached. In the final version, this was replaced with a colour selection mechanism in which the user picks a colour directly from a hand menu (or a colour tray if the tray type is set to 0 or 1). This change improves efficiency and reduces the complexity of the interaction.
- **Cube Replacement with Spheres**: The initial design used cubes as the interactive objects. In the final version, these were replaced with spheres to align better with the experimental design and provide a more intuitive interaction experience in the XR environment.
- **Memorization Time Specification:** The initial design had an unspecified memorization period denoted by t-seconds. In the final version, this period was explicitly set to 5 seconds, ensuring a consistent and controlled duration for short-term memory retention across all rounds.
- **Transition to Gray colour:** Initially, the cubes changed to white after the memorization phase to indicate readiness for interaction. In the final version, spheres turn grey instead, offering a neutral colour that avoids visual interference and maintains focus on the recolouring task.

These differences highlight adjustments made to enhance user experience and ensure a controlled, consistent interaction flow, particularly in the transition from memorization to interaction.

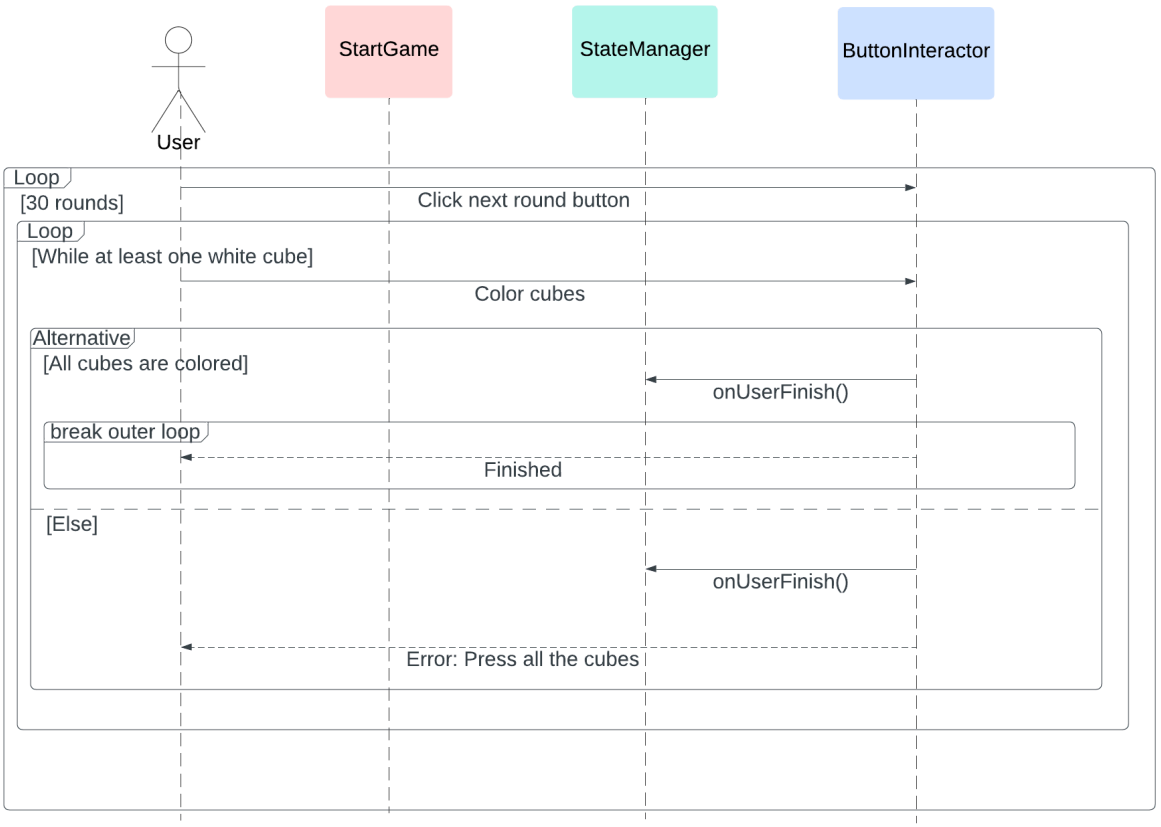


Figure 3: Initial Sequence Diagram (Part 2)

Figure 4.6 illustrates the main gameplay loop, capturing the iterative interactions across multiple rounds. Key differences between this initial version and the final design are as follows:

- **Reduced Number of Rounds:** In the initial design, the gameplay loop was set to repeat for 30 rounds. In the final version, this was reduced to 5 rounds to streamline the experiment and minimise user fatigue, focusing on quality of data rather than quantity.
- **Adaptive Sphere Count Based on Accuracy:** The initial design maintained a fixed number of interactive objects across all rounds. In the final version, an adaptive mechanism was introduced, where the sphere count increases by one in the following round if the user's accuracy is or exceeds 50%. This adaptation allows the game to dynamically adjust difficulty, providing a more nuanced evaluation of short-term memory performance under increased cognitive load.
- **Enhanced Feedback Mechanism for Accuracy:** While the initial version only logged basic completion data, the final design includes detailed performance tracking, particularly in relation to the adaptive sphere count mechanism. This additional feedback enhances the data collected and enables more comprehensive post-experiment analysis.