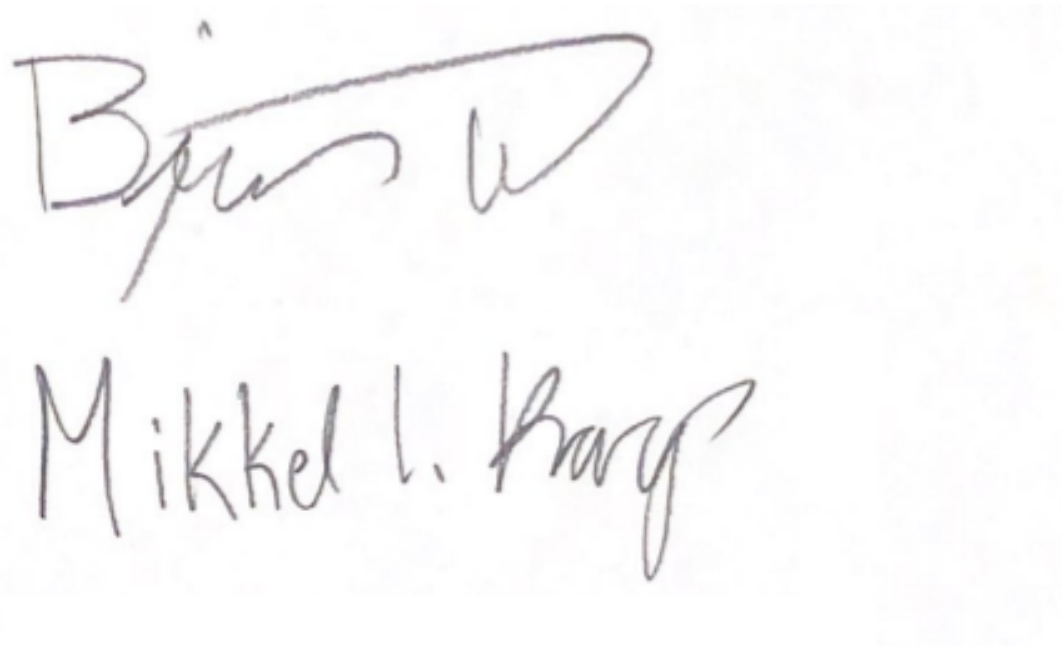
The Effects of Co-Location and Task Dependency in a Collaborative Multiplayer Virtual Reality Environment

A Study of Social Presence and Collision Behaviour

Bjørn Christian Winther, Mikkel Lynggaard Krarup

Supervisors: Carlos Mauricio Castano Diaz, Niels Christian Nilsson

Master's Thesis
Medialogy, Copenhagen, Spring 2023



The Technical Faculty of IT and Design
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
The Effects of Co-Location and Task Dependency in a Collaborative Multiplayer Virtual Reality Environment

Theme:
Co-Location and Task Dependency in Virtual Reality

Project Period:
Spring Semester 2023

Project Group:
N/A

Participant(s):
Bjørn Winther
Mikkel Krarup

Supervisor(s):
Niels Christian Nilsson
Carlos Mauricio Castano Diaz

Copies: 0

Page Numbers: 58

Date of Completion:
June 9, 2023

Abstract:

This thesis seeks to investigate the impact of co-location and task dependency in terms of co-presence, attentional allocation, perceived behavioral interdependence and collision avoidance. A mixed factorial design, incorporating both a within-subject and a between-subject design study including 32 subjects was used to form the conclusion. Collision avoidance and most of the sub-categories of social presence showed no significant differences between the conditions according to the Wilcoxon Signed Rank test, however, some conditions indicated that a significant difference existed, specifically: attentional allocation in the co-located distributed vs non co-located distributed condition. The co-located shared vs co-located distributed condition and the co-located shared vs non co-located shared condition indicated significant differences existed between them in terms of perceived behavioral interdependence. Similarly, collision avoidance did not show significant differences between any of the conditions. In conclusion, co-location and task dependency did not prove to have a significant effect overall when comparing the four conditions.

Contents

1 Introduction

Virtual Reality (VR) is experiencing a surge in popularity [1][2]. VR affords a highly immersive experience that transports users into virtual worlds, making it appealing for entertainment, education, training, and other applications. Technological advancements have also contributed to VR's popularity, with improved hardware and software capabilities. Additionally, the incorporation of social interaction in VR, allowing multiplayer experiences and social platforms, has further enhanced its appeal[2]. Finally, the entertainment and gaming industry has played a significant role in driving VR's popularity, offering immersive gaming experiences and introducing VR to a broader audience[2].

Previous work conducted by the researchers [3] found that natural walking was a compelling way to navigate Virtual Environments (VE) as well as increasing immersion, engagement and confidence in spatial navigation of the user. Based on the knowledge of the powerful effects of natural walking and preliminary research into a relevant paper [4] which motivated the researchers of this thesis to include task dependency as an independent variable for evaluation.

'Task dependency' in this thesis is defined as: Whether game objectives encourage solving in a shared or a distributed space among players. Task dependency is interesting to examine as it indicates whether collision behavior is a genuine concern in co-located environments and to what extent it is reflected in subjective evaluations of social presence and collision avoidance.

Preliminary research [5][4], in addition to personal aspiration to investigate co-located and non co-located multi-user VR games that afford natural walking, formed the motivation for this thesis. In the context of this thesis, co-location is defined as sharing the same physical play-space, contrary to non co-location where players are in physically separated locations. This research is interesting due to the common perception that VR experiences are typically non co-located. Therefore exploring the potential effects of co-location, particularly in terms of social presence is relevant as it might produce valuable insights into the interplay between physical proximity and immersive virtual environments.

In order to properly evaluate the effects of co-location within virtual environments (VE) it was decided to use social presence as the foundation for the evaluation as it strongly correlates with the objective of the thesis. Harms and Bioccas' validated standardized questionnaire [6] is used to measure and evaluate three subcategories to social presence: Co-presence, attentional allocation and perceived behavioral interdependence. Upon further preliminary research it was also deemed beneficial to include collision avoidance as another dependent variable to be measured and investigated. These measures will be employed in a mixed factorial design to assess the influence of co-location and task dependency, respectively.

1.1 Research question

This thesis aims to investigate the following research question:

To what extent is social presence and collision behaviour affected by co-location and task dependency in a collaborative multiplayer (VR) environment.

2 Related work

The following section will explore existing literature that is relevant to this thesis. This exploration aims to present the combined knowledge from previous research and investigate studies in related scientific fields. By examining the related work, we can design, implement and evaluate the developed prototype to address and attempt to answer the research question as effectively as possible.

2.1 Natural Walking in Virtual Reality

Naturally interacting within a VR environment is crucially important in order to create compelling and immersive experiences [7]. Specifically locomotion, referring to movement or the ability to move from one place to another, is one of the most common and universal tasks which is performed inside a VE. Real walking is the most natural locomotion technique and has shown a greater sense of presence in users compared to other techniques that does not include realistic body motion [8]. Furthermore, real walking has also shown improvement in regards to memory and attention [9]. Despite this, real walking is challenging to implement and design for as physical space limitations inhibits the ability to walk freely in the VE and must therefore be considered [7].

2.1.1 Challenges related to Natural Walking in VR

A review from 2018 conducted by Nilsson et al. [10] presents and defines the associated challenges with natural walking in virtual environments. Furthermore, in the review, research was conducted to understand how proper multisensory stimuli should be incorporated as feedback for interacting with the environment. In their book, Bowman et al. [11] define virtual travel as one of the most common and universal forms of interaction. Nilsson et al. further explains how virtual travel is usually secondary to other tasks which further underlines why it is crucial for users to be able to efficiently navigate the environment and remove their attention from the act of travelling itself [10].

Nordahl et al. [12] identified two interconnected challenges related to walking in virtual environments. The first challenge involves developing a travel technique that replicates the sensation of real walking, even when physical space limitations are present and may differ from the size of the virtual environment. The second challenge involves providing appropriate multisensory feedback to users, directly linked to their interactions within the virtual environment [12].

The publication discusses various approaches to address the challenge of achieving natural walking sensations in virtual environments, these are: repositioning systems, proxy gestures and redirected walking[10]. These techniques all enable users to replicate the sensation of natural walking in a virtual environment regardless of any physical limitations. Repositioning systems work by counteracting the movement of the user by means of physical or virtual equalization. Proxy gestures deals with elementary gestures performed by the user that are interpreted by the system and executed by the virtual avatar. Finally, redirected walking subtly redirects the users movement within a virtual environment in order to utilize the physical play-space more efficiently and translate this movement to a bigger environment [7].

Vision serves as a primary and precise source of spatial information, crucial for an individual's perception of movement in a virtual environment [10]. However, a phenomenon arises where the individual underestimates the virtual walking speed when visually presented speeds align with their physical walking speeds. While vision dominates spatial perception, Waller and Hodgson [13] describes how auditory information also plays a role in providing the moving observer with details about the virtual environment, including its size, object

positions, and ongoing events. In a virtual environment, auditory cues can be employed to inform observers about the presence of other users, both within and outside their field of view, often achieved through the use of footstep sounds. These auditory cues also offer insights into the surface characteristics being traversed. Additionally, haptic feedback serves as the third external source of sensory information for observers during virtual travel, actively reproducing forces and movements sensed through touch when in contact with objects [10].

2.1.2 The Effects of Natural Walking

In a paper that was published in 2023 [3] two different locomotion techniques were evaluated in terms of their effect on the player experience. One condition used VR controllers where inputs were translated to movement in the virtual environment and in the other condition natural walking was utilized to allow free movement within a confined physical play-space. The results of the study indicated that natural walking afforded significantly higher social presence as well as lowered simulator sickness. Results also showed that participants in the natural walking condition were more concerned with colliding with other players as well as forgetting that they were using controllers [3].

In a study conducted by Ruddle et al. [14] navigation performance in a VE was evaluated based on different types of body-based information as input. Different locomotion techniques were used as the independent variable of the study and consisted of: natural walking (tracking hall/omnidirectional treadmill) and controller input for translation and rotation within the VE [14]. A small-scale and a large-scale virtual marketplace were designed for two different experiments and in both sessions it was found that translational body-based information significantly increased the accuracy of participants' cognitive maps (a mental image of the layout of one's physical environment). Furthermore, in the small-scale environment participants' navigation performance was also improved with translational body-based information as input [14].

2.2 Social Presence

Social presence has been defined by Short, Williams, and Christie in 1976 as "the degree of salience of the other person in the interaction and the consequent salience of the interpersonal relationships" [15]. In 1995, Gunawardena redefined social presence as "the degree to which a person is perceived as a 'real person' in mediated communication" [16].

Oh, Bailenson and Welch [17] conducted a study on the concept and implications of social presence, as well as predictors of social presence. They found that social presence is a subjective experience that can be influenced by contextual and individual factors. The study found that a crucial component of interactions in virtual reality is the feeling of "being there" with a "real" person. Additionally, the perceived psychological distance between interactants has an impact on social presence. The study further divided the predictors of social presence into three overarching categories: immersive qualities, contextual properties, and individual traits, and identified subcategories such as visual representation of the communication partner, visual realism, high levels of interactivity, and haptic feedback. The article noted that physical proximity has a positive impact on interactants and their sense of social presence [17].

In their study, Oh, Bailenson, and Welch consistently found a positive relationship between physical proximity and perceptions of social presence [17]. They also noted that vivid perceptions of another person lead to greater enjoyment and social influence. In conclusion, the study found that when developing multiplayer virtual reality applications, there should be an emphasis on the immersive qualities, contextual properties, and individual traits in order to create a sense of social presence that maximizes the benefits of heightened social presence.

2.2.1 Dimensions of Social Presence

A consistent and internally valid measure was proposed in 2004 by Harms et al. [6] which was divided into six sub-dimensions that needs to be considered when measuring social presence. The following six sub-dimensions will briefly be described:

- **Co-presence**
  The observer's perception of their level of fellowship and aloneness, their extent of peripheral or focused awareness of the other individual, and their sense of how much the other person is aware of them peripherally or directly.

- **Attentional allocation**
  The level of attention that the user directs towards an interactant and the corresponding attention received from the interactant

- **Perceived message understanding**
  The user's capacity to comprehend the conveyed message from the interactant, along with their perception of how well the interactant's message understanding is.

- **Perceived affective understanding**
  The user's capacity to comprehend the emotional and attitudinal states of an interactant, along with their perception of the interactant's capacity to comprehend the user's emotional and attitudinal states.

- **Perceived affective interdependence**
  The level of influence and reciprocal impact between the emotional and attitudinal states of the user and the interactant.

- **Perceived behavioral interdependence**
  The degree to which the actions and conduct of a user impact and are influenced by the behavior of the interactant

2.3 Co-Presence

The following section establishes a foundation for understanding the importance of co-presence in the context of VR. Co-presence in VR has a potential to revolutionize social interactions and communication in virtual spaces. By bridging the gap between physical and virtual realities, co-presence in VR opens up new possibilities for shared experiences, team-building, and new forms of interaction.

In 1962, Goffman [18] defined the concept of "co-presence" as "a sense of being together in an environment where individuals become 'accessible, available, and subject to one another'". Furthermore, he explains that co-presence extends beyond physical proximity and includes mutual awareness among individuals, emphasizing the sensory aspects of the virtual environment. Co-presence includes two key dimensions: firstly, the ability to perceive and sense the presence of other individuals, and secondly, the feeling of being actively perceived by others and belonging to a cohesive group [18].

2.3.1 Co-Located vs. Remote Gameplay

In a study conducted by Born et. al. [4] the effect of physical co-presence in a virtual reality environment was evaluated. The study compared two different setups with a shared (co-located) and a separated play space. Several variables were considered and measured, these include, task awareness, spatial presence, player communication, interaction and performance. Born et. al. notes that physical co-presence seemingly benefits the player experience due to the inherent social interaction that these experiences afford. Furthermore, traditional co-located experiences also provide a higher sense of social presence (the sense of being together with another) which leads to increased enjoyment, involvement, engagement, competence and positive affect. However, in a related study Podkosova and Kaufmann [5] examined the impact of a multi-user navigation task in a room-scale VR environment and found that sharing a physical play space in VR can lead to undesirable effects such as reduced focus on the task at hand.

Born et. al. [4] chose a between-group design with a sample size of 92 participants collecting player experience related constructs. The results suggest that physically co-locating users in a virtual environment must be carefully considered as it can have a degrading effect on the player's experience. First and foremost, it is noted how wired head-mounted displays introduce the first potential interference in gameplay as cable management becomes a factor. Born et. al. further specifies two challenges related to the design of multiplayer VR experiences, these are spatial desynchronization (virtual play space is bigger than the physical demanding redirection techniques to be used) when there is a discrepancy in the size of the virtual and physical world there is a chance that spatial desynchronization may occur leading to unintentional physical collisions and threatening spatial orientation of the users [4]. Overall, studies have shown that co-location leads to an increased focus on collision avoidance in terms of greater clearance distances between users and thereby reducing the focus on the task at hand.

Results from the study [4] indicate that there is a comprehensible shift in user focus between the two conditions, however, further research is needed to document that these findings apply in more user-engaging interactions. Therefore, there is a need to develop VR multiplayer experiences where navigation is not the central objective and evaluate if and how physical collisions affect the experience. The researchers also noted how 'the awareness of co-location' of participants is a potentially confounding variable and future research should compare presence related constructs between co-located and remote participants [4]. Furthermore, the fact that participants could hear each other in physical space also presented a confounding variable as this allowed the participants to estimate their physical social surroundings. The study assessed two aspects of player communication, a subjective evaluation of the quality and the efficiency of the communication between participants and an objective measurement of the quantity of communication by recording audio during the test and performing analysis on the resulting audio files [4].

According to the findings of the study, the physical separation or co-location of two players in a multiplayer virtual reality game can have a significant influence on the player experience [4]. Consistent with anticipated outcomes and prior research, engaging in co-located play tends to undermine the cooperative social presence, communication quality, and performance. Therefore, the results expand upon previous investigations by demonstrating that unconscious processes, stemming from the physical co-presence of the other player and the technical properties of the VR setup, exert a significant impact on the cooperative experience and effectiveness. However, it is imperative to exercise caution when interpreting the findings, considering the limitations that have been addressed. Depending on the specific domain of application, such as educational or entertainment games, the VR interaction objectives may prioritize either the experiential or the performance related outcomes [4]. To enable informed design decisions within these domains, it is imperative for future research to delve deeper into multiple facets of player experience and performance in the context of multiplayer VR.

2.4 Collision Avoidance

Collision avoidance plays a crucial role in creating immersive and safe virtual reality (VR) experiences. In VR, users are transported to digital environments where they can interact with objects and navigate through simulated spaces. However, without proper collision avoidance considerations, users may face challenges such as unintentionally walking into physical objects or colliding with other users, disrupting the sense of presence and potentially causing discomfort or injuries.
On the basis of the work of Gerin-Lajoie et al in 2005 [19], Podkosova and Kaufmann [5] writes that collision avoidance behavior is widely understood to prioritize the preservation of the personal space surrounding an individual. This personal space is commonly modeled as an elliptical zone that must remain unobstructed to ensure the individual's psychophysical well-being. When engaged in locomotion, the personal space functions as a safeguarding zone that introduces a delay in response to unexpected obstacles and facilitates proactive planning of the locomotor trajectory.

Olivier et al. conducted a study in 2012 [20], on the pairwise interactions between walkers in a natural setting, and found that individuals were capable of predicting and adjusting their movement trajectories in response to potential collisions only as necessary. Based on the original source of Olivier et al, Podkosova and Kaufmann [5] states that walkers' collision avoidance behavior could be characterized as a mutual function of their states, denoted as the minimal predicted distance (i.e., the estimated distance at which they would cross paths at each moment in time). The results of the user study revealed that participants altered their walking trajectories to avoid collisions only when the minimal predicted distance was less than 1 meter. Additionally, it was observed that the collision avoidance strategies of individuals depended on their respective roles when walking on intersecting trajectories [Olivier et al. 2013]. Specifically, the individual who yielded to the other made more substantial modifications to their trajectory by adjusting both their heading and walking speed. Thus, it was concluded that collision avoidance was a collaborative process that involved intricate interactions between individuals.

2.4.1 Collision Avoidance In Virtual Reality

A study by Fink et al (2007) [21] research human perception and action by observing participants in virtual environments(VE), based on the assumption that locomotor behavior in a VE is similar to that of a real environment. The main objective of the study is to conclude whether VE's prove to be a reliable research tool, when investigating locomotor behavior.

The study [21] conducted an experiment that measured the following three factors: Locomotor path, larger obstacle clearance and slower walking speed. Subject of the experiments were instructed to walk towards a stationary destination goal while avoiding a static object in the path, that was matched in the physical– and Virtual environment. The results of the experiment showed an average of 0.16m larger maximum deviation (Locomotor path), 0.16m greater obstacle clearance and 0.13m/s slower walking speed. The models that were fit separately, accurately represented the mean virtual and physical paths with a high degree of correlation ($R^2 > 0.98$), suggesting that the models were very accurate in capturing the virtual and physical paths.

The study suggests a small but reliable difference in locomotor behavior in VE's compared to real environments. This research by Fink et al [21], conclude Virtual Environments to be a valuable and reliable research tool, when evaluating locomotor behavior.

2.4.2 Mutual Collision Avoidance with natural walking in VR

A study conducted by Podkosova and Kaufmann [5] in 2018 investigate mutual collision avoidance in virtual reality. Mutual collision avoidance is the definition of two moving individuals, both avoiding collision with one another, as opposed to regular collision avoidance where an individual avoids static objects. The effect of mutual collision avoidance is examined in two scenarios: co-located and non-co-located physical environments. Sensory abilities, cues, and perception of space in VR differs from that of real life. The common task of collision avoidance is therefore a valid concern to account for in VR.

This study [5] sets up an experiment to investigate locomotor behaviour of users in pairs, moving around in a matched physical and virtual space. To achieve an ecological simulation of mutual collision avoidance, participants were given a destination to walk towards. This creates the most realistic scenario, in which mutual collision avoidance occurs. The experiment was a within-subject design with subjects in pairs [5]. The goal was for subjects to pick up an object diagonally across from them when a walk-indication triggered. They would then be confronted with mutual collision avoidance, as their paths crossed. Two collision types were tested, frontal – and crossing, relating to the path of the subjects. These two collision types were then tested across three co-location conditions; Real, virtually co-located, and virtually distributed. An interesting find by the study, was that the collision behaviour was highly affected by which condition was tried first by the subjects [5]. The participants who tried the distributed condition first, tended to walk straight through each other's respective avatars and completely disregard any personal space. Whereas the subjects who tried the co-located condition first, tended to respect their partners avatar to a much higher degree across both virtual conditions and would furthermore not even attempt to go through each other, when trying the distributed condition [5].

The general results of the experiment showed that the crossing collision type produced greater clearing distance and slower walking speed, across all conditions, compared to frontal collision [5]. The results further showed a significant difference in locomotor behaviour between real and virtual environments. Greater caution was displayed across virtual conditions relative to the real condition. Moreover, greater caution was displayed when subjects were co-located as opposed to being distributed [5]. Although an overall more cautious locomotor behaviour was measured in the co-located condition, it should be noted that co-location did produce a higher score of co-presence, compared to the distributed condition [5].

2.5 Game Design Theory

Understanding the process of transforming an initial concept into a fully realized and enjoyable game experience is crucial in game design [22]. In order to achieve an optimal game design tailored to the specific target group, it is essential to involve the target players throughout the development process, maintaining a play-centric design approach [22]. Additionally, conducting iterative prototyping and playtesting at an early stage allows developers to gather feedback on aspects of the game system that may have been overlooked and redefine the implementation of features [22].

2.5.1 The Structure of Games

In game development, certain essential components are necessary for a game to provide users with an engaging interactive experience. These components, outlined in [22], will be summarized in this section and utilized as a guiding framework for the design process of the prototype.

Fullerton [22] provides a description of the components that form the structure of any game, which are as follows:

- **Players**
  The design of every interactive experience revolves around the players as the target audience.
- **Objectives** - The level of involvement that players exhibit in an experience can be defined by their desire to accomplish the objectives. Objectives, which can either be intrinsic (set by the player) or extrinsic (set by the system), serve as the primary motivators and driving force behind the experience.
- **Procedures** Procedures encompass the instructions provided by the system to guide players on how to navigate and interact within the experience, ensuring that the intended design and vision of the developers are effectively implemented.
- **Rules** The rules of a game establish a framework in which players operate, ultimately defining the limits and possibilities of their actions in order to achieve the objectives set by the game.
- **Resources** Resources refer to the assortment of tools, currencies, or other assets that are accessible to each individual player within the game.
- **Conflict** Conflict emerges in games, necessitating players to make strategic choices and resolve them in a manner that benefits their own objectives.
- **Boundaries** Boundaries play a crucial role in games as they prevent players from evading obstacles, objectives, or conflicts that may arise. It is essential to establish these boundaries to ensure that players are compelled to make decisions and engage with these scenarios instead of unfairly circumventing them.
- **Outcome** The outcome of a game determines whether players emerge as winners or losers, providing valuable feedback on their performance, highlighting areas for improvement, and offering insights into alternative strategies or actions that could have been pursued.
- **Challenge** Challenges presented in the game should gradually increase in difficulty. This will ensure a rising sense of tension, engagement, flow, immersion, etc. In order to have a successful game, there should be a difficulty curve, or players will lose interest in the challenges presented and the experience will become tedious.
- **Play** Play can be described as the freedom or flexibility that players are afforded within the structured framework of the game system. It represents the space for exploration, creativity, and personal agency within the established rules and mechanics.
- **Premise** The premise of a game serves as a means to captivate and engage users. It includes elements such as the setting, backstory, conflict escalation, and other compelling aspects of storytelling. By establishing these elements, the premise draws players into the immersive universe of the game, fostering a sense of immersion and engagement.
- **Characters** Players have the opportunity to embody various characters within the game's universe, enabling them to engage in dramatic experiences. Assuming these roles allows players to make choices and select solutions that align with the background or role they are portraying, enhancing their immersion and connection to the game.

When developing the prototype, it is crucial to thoroughly consider all of these components to ensure an engaging and immersive experience for the users. Each of these components, when combined, form the building blocks of a game, complementing one another to create a vibrant and dynamic experience. They provide both structure and limitations that guide the progression and course of the overall gameplay, resulting in a rich and captivating user experience as well as the most optimal game design. It is essential to establish

clear and well-defined objectives that can be collaboratively achieved using the available resources, while adhering to the boundaries, rules, and procedures set for the players. Additionally, it is important to provide players with a distinct and measurable outcome, accompanied by a difficulty curve that considers the players' expertise, ideally incorporating dynamic scaling of difficulty. Throughout the course of the experience, conflicts should arise within the game, serving the dual purpose of fostering learning and collaboration, as well as moving the overall experience forward.

2.5.2 VR Multiplayer Games

As the popularity of VR has surged in recent years [2], there has been a corresponding rise in the number of VR games available. Consequently, there is an increased need to comprehend the effects and implications of incorporating VR into gaming experiences.

A paper released in 2018 [2] sought to evaluate the differences between VR and non-VR multiplayer gaming through the use of a comparative study. A short multiplayer game was developed to be tested in both VR and non-VR in order to answer what difference VR makes in terms of player experience (Game Experience Questionnaire (GEQ)). Three different version of the game were tested with different independent variables being: Non-VR, simple VR and Full VR. The simple VR condition included a regular head mounted display (HMD) with an Xbox controller and the full VR condition included a HMD with two HTC Vive controllers and real walking [2]. In total 30 participants were recruited in the age range of 20-25 all being university students.

The hypotheses of the paper were that Full VR would produce the best results in terms of game experience as well as better results in the simple VR condition compared to the non-VR condition [2]. The results of the testing supported the hypotheses by showing significantly increased scores in all but two of the components of the Game Experience Questionnaire (core module) for the full VR condition. Furthermore, the results showed that simple VR also had better scores for all but one component compared to the non-VR condition[2].

2.6 Subconclusion

In conclusion an extensive review of existing literature and studies regarding: natural walking, social presence, co-location, collision avoidance and game design theory has been conducted and provided the researchers with valuable insights as well as established a foundation for the subsequent research. The exploration of various theories, methodologies, and findings has shed light on potential shortcomings and areas for further investigation. By examining the key contributions and limitations of prior works, this study is able to build upon existing knowledge and attempt to make meaningful contributions to the field. The comprehensive understanding gained from the related work section will serve as a solid framework for the subsequent research, design and implementation.

3 Methods

3.1 Study Design

This study investigates the effects on social presence and collision avoidance when in a multiplayer virtual reality game that is either co-located or non co-located physically as well as the impact of either a shared or distributed task dependency on the previously mentioned variables. The task dependency is implemented as two different versions of the game with either a shared task focus that encourages participants stay together in close proximity to each other throughout the experience and exchange information. On the other hand a distributed task focus encourages players to be separated throughout the experience and also maintain distance within the environment. These configurations will be further explained in terms of level design, game mechanics, feedback etc. in the implementation and design sections.

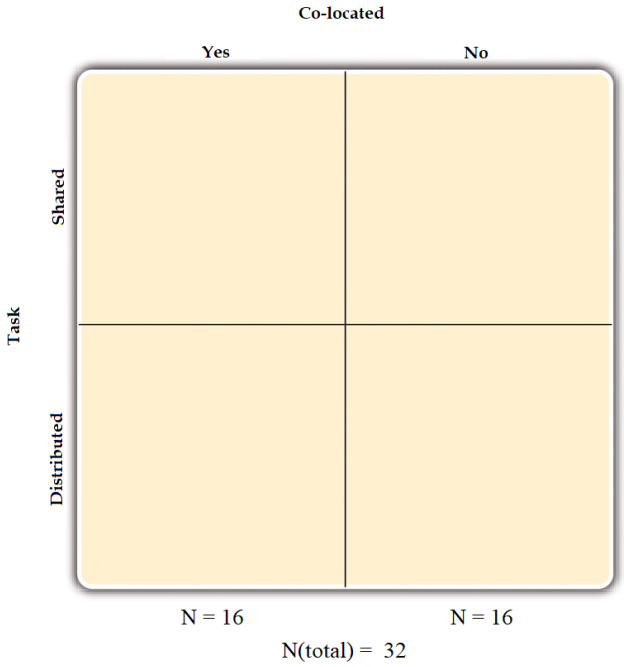


Figure 1: *Mixed factorial design*

For this thesis a mixed factorial design was used to evaluate the effects on the dependent variables caused by manipulating the independent variables. As can be seen in **figure [1]** the independent variables are co-location and task dependency, this is relating to the layout of the tasks and the environment in order to encourage either distributed or shared play-space navigation and interaction. 16 participants (8 groups) were recruited for both the co-located and the non co-located condition and each group tried both the shared and the distributed condition in a counterbalanced condition order.

By examining multiple factors we are able to investigate the effects of multiple independent variables simultaneously. Furthermore, by using this design it is possible to explore both within-subject and between-subject factors and how these factors interact with each other. In order to get a better understanding of the study design further explanation on how these conditions were specifically implemented will be presented in the design [4] and implementation [5] sections.

3.2 Participants

In total 16 groups were recruited with two participants in each group, totalling 32 participants, 21 male (66%) and 11 female (34%). Eight groups were recruited in the first days of the testing for the co-located condition and another eight groups were recruited in the last days of the testing for the non co-located condition.

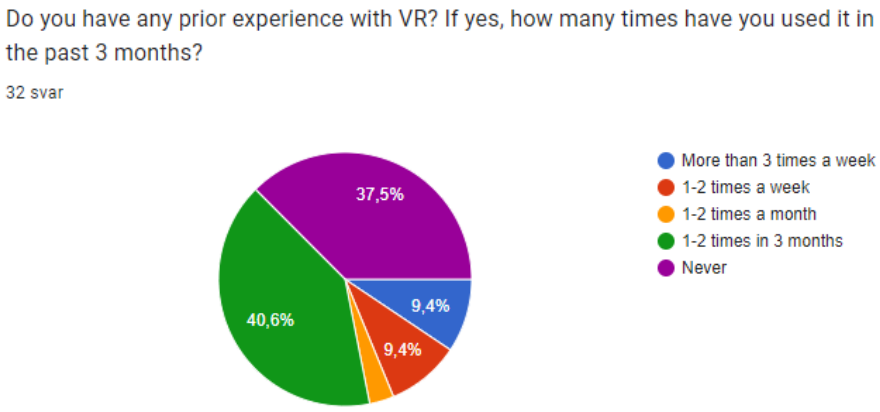


Figure 2: *VR experience distribution among participants*

As can be seen from the above **figure [2]** only 6 (18,8%) of the participants have regular experience with VR. 12 (37,5%) participants had no experience with VR at all. The remaining 14 (43,7%) participants were relatively inexperienced VR users and only used it rarely.

In order to recruit participants for the experiment convenience sampling was chosen as well as voluntary response sampling as both are convenient and efficient ways to recruit a sample of the population which are immediately available, which was necessary due to the scope of the project and time restrictions [23]. Participants were recruited from group rooms and common areas at Aalborg University and were followed to the testing rooms. Some participants would also encourage their fellow students to participate which resulted in even more test participants. It should be noted that convenience sampling will potentially introduce bias and confounding variables in the collected data, however, to mitigate this potential limitation, an effort will be made to incorporate voluntary response sampling, which will help acquire feedback that is more objective in nature [23].

3.3 Apparatus

Two available rooms were located and prepared for testing at the university in cooperation with the information desk that informed on the availability of the rooms each day of testing. As it was not possible to directly reserve the rooms due to economic restrictions it was necessary to regularly inquire about potential reservations and structure the testing sessions accordingly. On the first days of testing only a single room was required in the dimensions corresponding to the virtual environment (6x8 meters) as the co-located condition were to be tested and following this two rooms were required with the same dimensions of free physical space, it was possible to find two rooms adjacent to each other and clear the needed space.

For the hardware it was chosen to use two Meta Quest 2 HMDs as these headsets do not require a cabled connection and allows for the software to run directly on the headset affording unrestricted natural walking

and orientation in all directions. It was necessary to disable the guardian system on the Meta Quest 2 which is responsible for creating the safe boundary in the virtual environment as it did not allow for us to define a safe play space in the required dimensions. This obviously created some security concerns as the system would have to be very precisely calibrated to not create desynchronization between clients resulting in collisions with the physical environment or other users. Prior to each intervention the researchers will calibrate the HMDs based on marked positions in the physical play space and ensure that the environments are aligned both physically and virtually before participants are given the headsets for testing.

3.4 Procedure

The step-by-step procedure used during the testings will be listed here, from the point where participants entered the room, until they concluded their testing session. The given instructions were consistent in all conditions:

- Participants were first instructed to sit at a computer that had the first part of the questionnaire ready to be filled out. This part included a consent form, demographic information, VR experience and a preliminary simulator sickness evaluation.
- **Co-located condition** - Participants were given one headset each.
- **Non co-located condition** - One randomly selected participant from the group were assigned and directed to the adjacent room where a headset was prepared and ready to use.
- Participants were instructed to stand still and await instructions from both the in-game narrator as well as the researchers while the game started.
- Participants were instructed by the in-game narrator to "help each other cross to the heavens gate, you can only win if you trust and help each other"
- One of the researchers clarified the information by stating that participants had to help each other to make it across to the gate, that they could not do it on their own and that they should avoid exceeding the virtually presented play-space at all costs. A final hint was given that they should pay attention to their assigned color.
- Participants then began to communicate with each other and interact with the environment to figure out how the game mechanics worked and ultimately how to win the game. Researchers did not assist participants during this.
- When entering the virtual elevator the participants were instructed by the researchers to stand still until it had completely stopped due to consistent desynchronization issues with the elevators movement. No further interruptions were made to the game experience.
- The test subjects then filled out the second part of the questionnaire measuring the dependent variables: co-presence, attentional allocation, perceived behavioral interdependence and collision avoidance.
- Participants were given the headsets again to try the second condition respective of whichever condition they tried first, this time researchers simply instructed that the mechanics and objectives were the same except for some slight differences.
- Upon completing the second intervention the participants filled out the third and final part of the questionnaire which evaluated the same constructs as the second part, however, this time after the new condition had been tested.

- Subjects were thanked for their participation by the researchers and encouraged to grab snacks and drinks that were provided after the testing session. Any questions regarding the experience or thesis that the participants had were also answered by the researchers.

3.5 Measures

3.5.1 Likert scales

Likert scales are commonly used in questionnaires to measure participants' agreement with specific statements. They are psychometric response scales that help researchers evaluate the effectiveness of interventions [24]. The most widely used Likert scale is the 5-point scale, ranging from "strongly disagree" to "strongly agree," with a neutral option in the middle. However, researchers can also employ 7-point or 9-point scales to capture more diverse and accurate responses. Alternatively, a 4-point Likert scale can be used as a "forced choice" scale, eliminating the possibility of a neutral response [24].

The advantages of Likert scales include their simplicity and ease of comprehension for participants. They generally yield reliable results when evaluating a prototype [24]. However, Likert scales are not without their drawbacks. One limitation is the central tendency bias, where respondents tend to choose responses near the scale's midpoint, avoiding the extreme ends [24]. Additionally, it is crucial for participants to remain objective and provide honest opinions, rather than trying to please the experimenters or interviewers. To achieve this, researchers must account for and seek to eliminate any confounding variables that could influence participants' responses as much as possible [24].

For this thesis a 7-point likert scale will be used ranging from 1 (not at all) to 7 (very much/well) allowing for easy comprehension, neutral responses and a more diverse data gathering. Confounding variables will be considered and mitigated as much as possible by taking them into account, not omitting them from the results as well as using reliable and valid measures.

3.5.2 Social Presence

To evaluate social presence based on self-report measures, it is crucial to utilize a valid and internally consistent measure proposed in a relevant study. One such study was conducted by Harms and Biocca in 2004 [6], where they identified six subdimensions of social presence that need to be considered when measuring it. These subdimensions are listed in the related work section [2.2.1]. Harms and Biocca's study provides a comprehensive framework for understanding and measuring social presence in virtual environments.

For this thesis, it was chosen to focus on the following three sub-dimensions of social presence [6]: Co-presence, attentional allocation and perceived behavioral interdependence. These sub-dimensions are the most relevant and accurate in nature in respect to the aim of the thesis. Furthermore, to avoid fatigue in participants, it was necessary to condense the questionnaire as much as possible and limit the measure, to the most relevant dependent variables.

3.5.3 Collision Avoidance

In order to evaluate collision avoidance behavior Born et al. [4] formulated three subjective questions which participants would answer post-intervention. The following three questions were formulated and were rated on a 5-point likert scale [4]:

- Did the presence of your teammate restrict your freedom of movement?

- Did you deliberately avoid physical contact with your teammate?
- How much did you pay attention to where your teammate was and what she/he was doing?

This self-report measure is entirely focused on the participants' focus and attention to their teammate and collision avoidance in the different conditions.

3.5.4 Telemetry data

Telemetry data relating to the participants' headsets will be logged during the testing to support observations and data collected by the researchers. At fixed time intervals the computer will log positional and rotational information about each headset and write it to a text document on the pc. After each testing session the logged data would be categorized and sorted according to the condition that the participants were exposed to. This data is objective compared to the subjective measures collected after each testing and any potential trends and tendencies in participants' movements can be analyzed and perhaps even support other claims.

3.6 Validity and Reliability

To ensure the validity of the test results, internally valid and standardized questionnaires were employed as well as considering all data points, including outliers [25]. Additionally, ecological validity was ensured by designing tests that closely resemble real-life scenarios in which the system would be used, thereby ensuring accurate data collection [25]. To ensure reliability and validity, the standardized questionnaires included multiple variations of questions related to the same subject [26]. Consistency in testing conditions, instructions, and circumstances was maintained to enhance the reliability of the data [25]. Efforts were made to minimize external factors and confounding variables that could influence the outcomes by adhering to a standardized approach [25].

3.7 Scrum

The Scrum management framework [27] was partly used to structure and coordinate the development effort into several iterations/sprints. Firstly, a product backlog was created with 'Trello.com' which contained all the relevant features that could potentially be implemented in the game to reach the desired prototype. Furthermore, the backlog would keep track of which person were responsible for implementing which feature, deadlines and included notes from the researchers that would be discussed during meetings. A sprint was then defined by the researchers and all vital features were decided upon and included in that sprint backlog based on research and relevance to the final prototype. Regular sprint reviews were conducted and evaluated internally by the researchers and deadlines would be established for implementing certain features. If the deadline was exceeded for any number of reasons, then that feature would be re-evaluated and re-designed or the deadline would be extended to allow including that feature.

4 Design

4.1 Design requirements

The design requirements of this thesis plays a crucial role in guiding the development and implementation of the proposed application. In order to design an effective and efficient system, it is imperative to establish a clear set of requirements that will serve as a foundation for the entire design process. This section aims to define the specific objectives, constraints, and functional specifications that must be addressed and met in order to fulfill the research question. The design requirements of the system are derived from the findings presented in the preceding sections, and the relevant research will be referenced along each item.

- **Natural Walking**
  The system should support natural walking in the virtual environment, allowing users to move intuitively and seamlessly within the virtual space [2.1] [2.1.2].

- **Multiplayer Functionality**
  The application must support multiplayer capabilities, allowing multiple users to connect over a network and synchronize their experiences in real-time [2.5.2].

- **VR Implementation**
  The system should be designed for VR, utilize VR headsets and immersive virtual environments to provide an engaging game experience [2.1] [2.5.2].

- **Real-time Position and Rotation Tracking**
  Accurate, real-time tracking of user position and rotation is required to ensure precise and responsive interactions within the virtual environment and among the interactants [2.1.1].

- **Co-location**
  The system should facilitate co-location, allowing users to share the same physical space while engaging in the virtual environment [2.3.1].

- **Incorporation of Game Mechanics and Theory**
  The application should incorporate game mechanics to enhance engagement and provide interactive elements that motivate and challenge users within the virtual environment [2.5] [2.5.1].

- **Concise and Consistent Feedback**
  Clear and consistent feedback should be provided to users, conveying relevant information about their actions, progress, and state of the game [2.1.1].

- **Condition Toggling**
  The game should consist of two versions that can easily be switched between, where the task focus is either shared or distributed among users to allow evaluating collision avoidance, offering distinct collaborative experiences [2.4] [2.4.1].

- **Communication and Cooperation**
  The system should facilitate communication and cooperation among users, enabling effective collaboration and interaction within the virtual environment [2.5.2].

- **Mitigation of Simulator Sickness**
  Game mechanics should be implemented to minimize simulator sickness and provide a comfortable VR experience, considering factors such as motion sickness and discomfort [2.1.1].

4.2 Game Design

To test the hypotheses, a virtual environment was developed, incorporating game mechanics and essential functionalities. The application was specifically designed for the Meta Quest 2 VR headset, leveraging its capability to track the user's physical position and accurately translate it within the virtual environment. The virtual setting includes a map divided in two levels, where height is the separation factor. These levels are equally defined by a boundary that defines the play-space. Calibration is a prerequisite for every user who starts the application and participates in the game. Once a player completes the calibration process from a predetermined physical location, the virtual map is aligned with the physical play-space, for which the map was originally constructed. Additionally, once calibrated, players are able to accurately track each other's positions, allowing them to effectively engage with one another without being concerned about potential collisions. The level design section will further explain the design in categorized subdivisions, to properly convey the thought process behind the most crucial game design choices and proposed solutions.

4.3 Level Design

A consistent and stable gameplay loop was required and designed by the researchers that would limit unintended behavior as much as possible. Several considerations went into the map design, the stages of the game, feedback, consideration to players and playability. Numerous internal evaluations of the prototype and feedback from peers, helped the creation and common thread of the game.

4.3.1 Virtual Environment

The map was designed for a physical play-space of 8x6m. Finding a balance of what was logistically plausible to test and provide a proper physical space that allows players to roam free with natural walking. The map is designed in two separate levels, distinguished by disparity in height. The idea was to exploit the physical play space, by having various virtual settings for the players, interchanged by different stories.

The theme is a heaven/hell scenario, where players start in heaven with the task of safely crossing to the heaven's gate by cooperating. During the game the players will eventually fall into the hell environment, with new objectives to complete. This allows for the same physical play-space to be exploited once again with a new virtual environment. After completing another cooperative task in the hell environment, an elevator will transport players back up for another try at the heaven section. The two environments, heaven, and hell are approximately the same dimensions. The reason why they are not identical, is because a precaution had to be made. Since players are able to roam free at any point during the game, stepping over any edge of the heaven environment purposefully, had to be accounted for. As a result, the elements in heaven conclude a slightly smaller virtual play-space. By introducing this change, any player that would step off the edge in heaven, would still be inside the physical play-space, when falling into hell, although they would be on the edge. Both the upper and lower levels visually defined a clear edge of the map, to suggest where the play-space concluded.

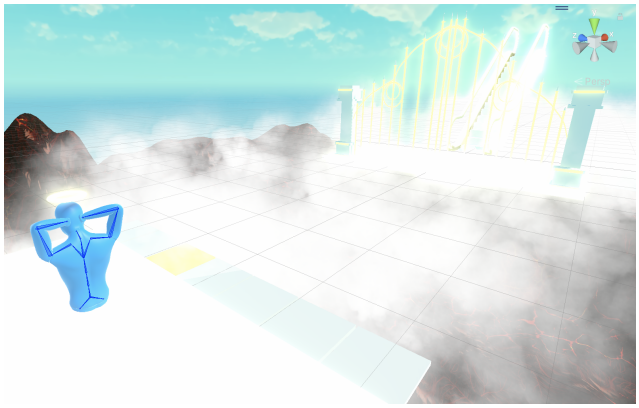


Figure 3: The heaven environment

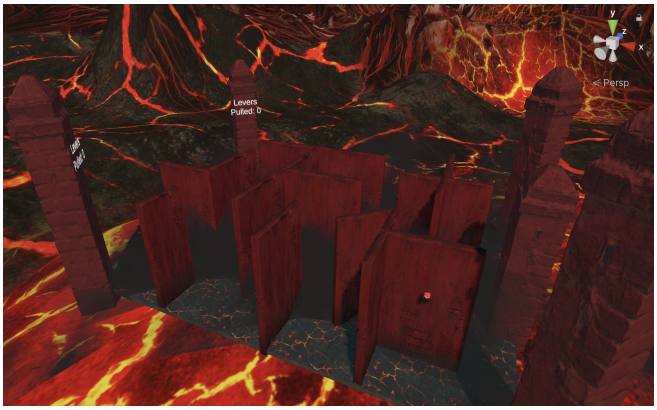
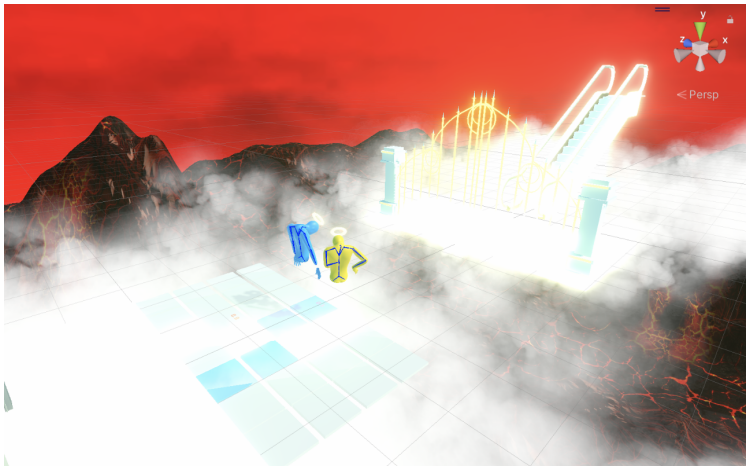


Figure 4: The hell environment (shared condition)

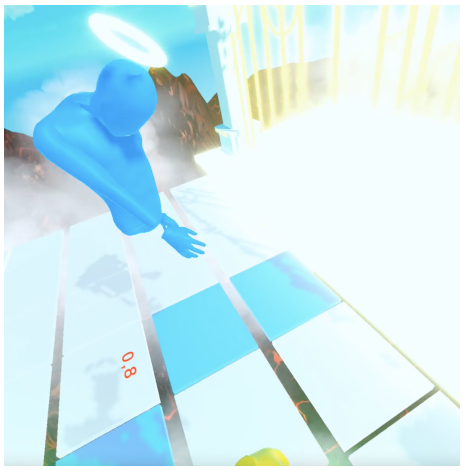
4.3.2 Game walk-through

The game requires calibration upon launch for both clients, to accurately synchronize the play-space to the virtual play space, in addition to synchronizing players physical and virtual positions. The game will not start until it has registered three connected clients, ensuring the connection of two headsets and a computer to log positional data. Upon game start, players material will change, platforms spawn and narration begin shortly after. The objective is explained by the narrator, who incentivizes players to cooperate to cross safely to the heavens gate. Players are only able to view which platforms their teammate can safely step on and are therefore forced to cooperate to complete the game. The players colors match the platform they are assigned to, which is a hint for the players. The game helps players with tips and information but circumvents directly providing the solution. The design is purposefully created with a problem-solving, cooperating intent to encourage teamwork and communication.

Upon stepping on any platform, a timer displayed on the platform, will count down from one second. After standing on the platform for one second, the platform exhibits one of two behaviors, depending on whether it is assigned to that specific player or not. If the platform is solid for the player, it will change its material, increasing its emission, and trigger a success audio clip. If on the other hand, a player steps on an unassigned platform, it will trigger a vanish sound and be removed temporarily alongside with all other surfaces in heaven. This game mechanic was essential, to prevent a potential collision scenario where players in the same physical play-space would no longer be able to perceive each other, as they would find themselves separated in different virtual environments, differentiated in virtual height. In terms of safety, it was a vital requirement to always have players in the same VE, to avoid any collisions.



(a) Screenshot during gameplay



(b) In-game footage

Figure 5: Heaven gameplay

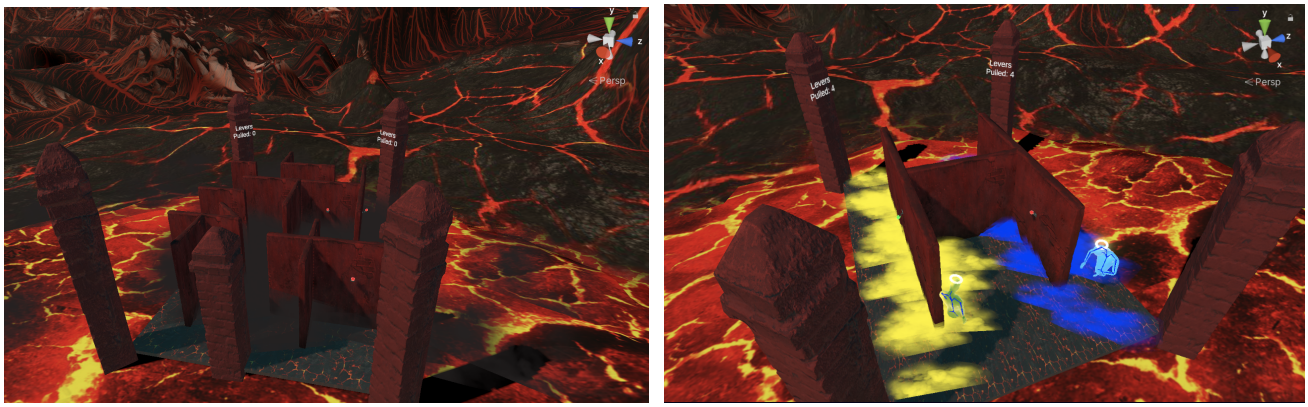
Finding the hidden pathway through the matrix of platforms by collaborating and communicating is the main objective at hand. If players are on the path of succeeding on their first try at the platforms, they will be forced into the hell environment. This will happen if they reach the halfway point on the platforms in their first go. In this case, a narrator will inform the players, that they are simply being forced into hell to experience it once, as to not cause any doubt or confusion about the game mechanics. Falling down into hell triggers a fade effect, that completely fades the screen, to counter simulator sickness. Upon landing in hell, the fade is reversed. The hell setting is a maze with walls and levers. The objective is to find all levers and pull them, to call for the elevator. All levers are only interactable by a specific player. If player1 interacts with a lever designed for player2, information will appear in front of player1's view. This approach was implemented to further encourage and promote collaboration and communication among players. Pulling a lever would of course prompt visual and audio feedback.

In case a player tries to walk through any wall, the canvas will fade to black, and text will appear, to instruct the player, to move out from the wall. This functionality is also key in preventing players from thinking they are stuck, if they happen to land inside a wall, after falling from heaven. As seen in **figure [6]**, visual feedback is provided to inform players on the amount of pulled levers, in addition to whether the elevator is on its way. This feedback was also accompanied by a narrator voice-over.

4.3.3 Condition altered design

Given the objective of testing shared versus distributed environments, the application innately had to account for this. In heaven, the pathway changes according to which condition is tested (shared or distributed). If the condition is shared, the path of assigned platforms relative to each player will encourage them to cross each other several times, in addition to constantly being virtually close in proximity to one another. If the condition is distributed, players will consistently maintain a greater virtual distance from each other, while also ensuring that their paths do not intersect. Likewise, the hell environment diverges, given what condition is tested as seen in **figure [6]**. In the shared condition, the hell environment is a maze, where players are encouraged to work together to find and pull all levers. In this condition, the levers are exclusively associated with either player 1 or player 2, thereby restricting their interaction to a specific player, to further incentivize teamwork and communication. Feedback is naturally provided, in case player1 attempts to interact with a player2-lever.

If the condition is distributed, the hell elements are rearranged for a new layout, that is designed to separate the players. Two color-matched particle effects, appearing as misty fogs, help intuitively guide players to their intended space **figure [6b]**. Additionally, feedback will appear on the canvas, in case a subject finds themselves on the wrong side of the play-space, which will further help navigate them to the correct space.



(a) Shared condition **(b)** Distributed condition

Figure 6: The Hell environment in the two different conditions

5 Implementation

The following section will cover the implementation of the game, the utilized platform alongside the most significant libraries, tools and assets used for development. It will present and explain how the developers have proceeded to implement solutions to satisfy the game design requirements for the application. Excluding tools, libraries and built in packages by unity, this project features 25+ scripts, that was written/implemented by the researchers. The most crucial scripts will be presented during this section of the thesis.

The application was built in Unity with XR as the most noteworthy library. More specifically, the extension used, was the XR interaction toolkit. This library provides functionality to interact with objects in a scene using Oculus controllers as input. As the network solution, Normcore was chosen. Normcore provides a free and straightforward solution for hosting a server and synchronizing data between clients.

5.1 XR Interaction Toolkit

The XR Interaction Toolkit is a powerful library that Unity offers to create immersive experiences in VR [28]. It provides a collection of user-friendly components and features that make it easy to build interactive interfaces and interactions in VE's. With this toolkit, developers can add hand and controller tracking, object manipulation, teleportation, and gesture recognition to their VR applications. By using the XR Interaction Toolkit, developers can create engaging and immersive VR experiences with intuitive user interactions.

5.2 Normcore

Normcore is a Unity plugin specifically designed to facilitate networking and collaborative features within Unity-based projects [29]. This plugin streamlines the implementation process by providing developers with a range of pre-built networking components, scripts, and asset management systems. By integrating Normcore into a project, developers can easily incorporate robust networking capabilities such as real-time collaboration, synchronized manipulation, and version control. This plugin empowers developers to enhance multiplayer experiences, enable seamless communication between players, and optimize the networking infrastructure of their Unity projects, allowing for efficient and scalable networking solutions.

5.3 Multiplayer application with Normcore

As mentioned, the project was setup in Unity using XR and Normcore. It is rather simple to get started with Normcore in a Unity project, although there are things to consider, in order to best streamline synchronization between both clients. Normcore does not appoint a master to any client. In some scenarios this will cause conflicting data between the clients, which in turn can break any multiplayer game. Normcore leaves it up to the developer, to solve this issue. The solution for this project was simply to appoint one of the clients the "master", thus only have the master execute certain parts of the code. Both clients will check if a master has been assigned and if not, the first connected device, becomes the master of the server. Moving forward the term "master" will be used to describe the master client, and client to describe the additional regular clients.

5.4 GameManager

The GameManager script essentially takes care of the entire game-loop and interacts with a number of other scripts as well as setting game variables which certain scripts depend on. The script is run on the master as well as the client, yet most code in the gamemanager is exclusively executed or set by the master to avoid conflicting data. Several checks are made in the update function to clarify whether the client is the master, if all players are connected and whether the platforms have been instantiated. The platforms are instantiated

once these conditions are true and from there, a constant cycle of checks begin. Nearly all functions that are called in the GameManager, are effectively called by a reference to the PlatformManager script. This was done, in order to have all platforms spawn under a parent gameobject that can easily be positioned, as well as encapsulate all functionality to manage the spawned array of platforms.

When the platforms are spawned, the game commences. A narrator will trigger and instruct the players upon the game mechanics. At this point, two methods are called continuously to check whether the correct platforms are collided with, in addition to whether the next row should appear. A counter, to keep track of failed platforms, ergo platforms that should not be stepped on, is checked to determine if all surfaces should disappear, effectively sending the players to the hell section. If this is the case, a sequence of coroutines are triggered to reset the playing field, effectively rearranging the platforms while generating a new path sequence. The gameloop then returns to call the two aforementioned methods, that fundamentally checks whether the players are crossing correctly, by stepping on the right platforms.

```
122         CheckAndSetAvatarArray();
123         AssignPlayerNumbers(); // also sets IsServer!
124         if (CheckIfServerExist())
125         {
126             // Only the server executes the following:
127             if (!IsServer) { return; } // Only do the following if client is Server/Master:
128             if (!CheckAllPlayersConnected()) { return; } // check if all players are connected, before realtime spawning objects:
129             if (!isPlatformsInstantiated)
130             {
131                 PlatformManagerScript.RealtimeInstantiatePlatforms();
132                 PlatformManagerScript.SetRandomSequence();
133                 isPlatformsInstantiated = true;
134                 syncedGameVariables._backupBool = true; // Platforms Instantiated = true
135             }
136 
137             if (Platform.NumberOfPlatformsDestroyed > 0) // IF FAIL:
138             {
139                 PlatformManager.isResetFinished = false;
140                 syncedGameVariables._backupFloat = Platform.NumberOfPlatformsDestroyed; // To reset material
141                 PlatformManagerScript.DestroyAllSurfaces(); // Also sets forcedIntoHell = true
142                 PlatformManagerScript.ResetAllPlatforms(); // Resets: ((Vars:) Rowindex, ActivatedPlatformsInRow), PlatformActivated and
143                 PlatformManagerScript.StartCoroutine(PlatformManagerScript.ActivateNextRowIE(PlatformManagerScript.RowIndex,3)); // enable all surfaces again after X sec
144                 PlatformManagerScript.StartCoroutine(PlatformManagerScript.SetRandomSequenceAfterXTime(5));
145                 PlatformManagerScript.StartCoroutine(PlatformManagerScript.ResetLocalPlatformVariables(7));
146                 PlatformManagerScript.StartCoroutine(PlatformManagerScript.ResetMaterial(8)); // Only for server // Also Sets IsResetFinished = true // if this timer is ad
147                 runOnceOpen = false;
148                 runOnceClose = false;
149                 Platform.NumberOfPlatformsDestroyed = 0;
150             }
151             else if(PlatformManager.isResetFinished) // RUNNING ALL THE TIME IF PLAYERS HAVE NOT FAILED:
152             {
153                 PlatformManagerScript.ActivateNextRowV2(PlatformManagerScript.RowIndex);
154                 PlatformManagerScript.CheckCorrectPath(PlatformManagerScript.RowIndex);
155                 if (PlatformManagerScript.RowIndex > 9) // if win
156                 {
157                     syncedGameVariables._sequenceIndex = PlatformManagerScript.RowIndex; // if 10
158                 }
159             }
160 
161         }
162         else
163         {
164             AssignServer();
165         }
```

Figure 7: *A snippet of the GameLoop exclusively handled by the master client*

5.5 PlatformManager

The PlatformManager class/script holds all the functionality for arranging, managing, and manipulating the platforms. This script primarily handle all platforms as a joined unit as the majority of algorithms in the class, loop through each gameobject in the array; “PlatformArray”, to execute the correct response. The platform class, which will be described in the next section, is responsible for the functionality of each individual platform. The platformManager class however, regularly checks the platform script of each platform, when looping through the array to check for collision, activation and unique attributes. It also does the exact opposite, as it will loop through the entire array and set a solid-variable, to identify a path of solid platforms, that can be stepped on by the respective player.

The PlatformManager class holds six two-dimensional integer arrays, with hardcoded inputs, to set a random pathsequence, ergo a pathway for players to follow to win the game. An example of this is shown in **figure [8]**. Each input is one of three values, 0, 1 or 2. The numerical value indicates whether the platform is fragile, solid for player 1 or solid for player 2. In other words, the interaction and movement of player1 is confined exclusively to platforms with a value of 1, while player 2 abides by the same restriction. In case either player engage with a "fragile" platform, all platforms will withdraw and reset with a new path sequence. The two-dimensional path sequence arrays are divided in shared and distributed manner. That is, the first three arrays will have pathways that force crossing and a close proximity relation among players, contrary to the last three arrays, which feature pathways designed to separate players.

```
private int[,] pathSequence1 = new int[ColoumnLength, RowLength] {          {0, 0, 1, 0, 2, 0, 0},
                                                                            {0, 0, 1, 2, 0, 0, 0},
                                                                            {0, 0, 2, 1, 0, 0, 0},
                                                                            {0, 0, 2, 1, 0, 0, 0},
                                                                            {0, 2, 1, 0, 0, 0, 0},
                                                                            {0, 1, 2, 0, 0, 0, 0},
                                                                            {0, 1, 2, 0, 0, 0, 0},
                                                                            {0, 2, 1, 0, 0, 0, 0},
                                                                            {0, 2, 0, 1, 0, 0, 0},
                                                                            {0, 0, 2, 1, 0, 0, 0}

};
```

Figure 8: *One of six hardcoded two-dimensional integer arrays to set the path sequence of the platform matrix*

The script contains numerous methods and coroutines that traverse through each platform and impose functionality accordingly. To avoid repetition and redundancy, only the most important algorithms of the class will be presented.

As previously stated, this part of the code in PlatformManager, is exclusively executed by the master client. The initial method of the script is called RealtimeInstantiatePlatforms().As seen in **figure [9]** a double for-loop fills the array platformArray[], while realtime-instantiating prefabs at incrementing positions. Within the same scope of the loop, each platforms' spawn position is logged and afterwards relocated to its despawn point, which is furthermore logged. This functionality is executed by two methods included in the script, and simply ask for permission to manipulate the passed gameobject as well as storing the transform data.

```
127     public void RealtimeInstantiatePlatforms()
128     {
129         platformArray = new GameObject[ColoumnLength, RowLength];
130         for (int i = 0; i < ColoumnLength; i++)
131         {
132             for (int j = 0; j < RowLength; j++)
133             {
134                 platformArray[i,j] = Realtime.Instantiate("PlatformV2", new Vector3(transform.position.x + i * coloumnMultiplier -coloumnOffset, 0, transform.position.z
135                 {
136                     ownedByClient = false, // True?
137                     preventOwnershipTakeover = false,
138                     destroyWhenOwnerLeaves = false,
139                     destroyWhenLastClientLeaves = true
140                 });
141                 MoveAndSetStartSpawnPosition(platformArray[i, j].transform.GetChild(0).gameObject); // Changed argument child obj instead of parent obj of Prefab.
142                 MoveAndSetDespawnPosition(platformArray[i, j].transform.GetChild(0).gameObject);
143                 Debug.Log("All Platforms Instantiated and Despawned");
144 
145             }
146         }
147     }
```

Figure 9: *This method instantiate and reposition all platforms realtime, effectively synchronizing what appears to all clients*

The next method utilized is called SetRandomSequence() **figure [10]**. Its purpose is to establish a sequence of solid platforms that the corresponding player can successfully step on. It begins by examining a static

game variable to determine whether the condition is shared or distributed. Depending on the variable's value, one of two switch cases is triggered. Within these switch cases, a random integer is generated within three specific intervals to select the path sequence for the platforms based on the corresponding condition. As mentioned earlier, the PlatformManager script contains six distinct pre-defined two-dimensional integer arrays, each holding a value of 0, 1, or 2. The switch case selects one of these arrays based on the randomized integer. Once a random path sequence is chosen, the array is traversed, and each index is inspected for its value. If the value is 1, a networked/synchronized boolean called "isSolidPlayer1" associated with the platform is set to true. Likewise, if the value is 2, "isSolidPlayer2" is set to true. If the value is 0, both variables are set to false. These assigned values determine which platforms can be stepped on and by which player.

```
275         public void SetRandomSequence()
276         {
277             int[,] pathSequence;
278             if (GameManager.IsTaskSharedStatic)
279             {
280                 int randomChance = Random.Range(0, 3);
281                 Debug.Log("Shared Task - Path: " + randomChance);
282                 switch (randomChance)
283                 {
284                     case 0:
285                         pathSequence = pathSequence1;
286                         PlatformSequence = 1;
287                         break;
288                     case 1:
289                         pathSequence = pathSequence2;
290                         PlatformSequence = 2;
291                         break;
292                     case 2:
293                         pathSequence = pathSequence3;
294                         PlatformSequence = 3;
295                         break;
296                     case 3:
297                     default:
298                         pathSequence = pathSequence1;
299                         PlatformSequence = 1;
300                         break;
301                 }
302             }
303             else...
327             for (int i = 0; i < ColoumnLength; i++)
328             {
329                 for (int j = 0; j < RowLength; j++)
330                 {
331                     if (pathSequence[i,j] == 1)
332                     {
333                         platformArray[i, j].gameObject.GetComponentInChildren<PlatformData>()._isSolidPlayer1 = true;
334 
335                     }
336                     else if (pathSequence[i, j] == 2)
337                     {
338                         platformArray[i, j].gameObject.GetComponentInChildren<PlatformData>()._isSolidPlayer2 = true;
339                     }
340                     else
341                     {
342                         platformArray[i, j].gameObject.GetComponentInChildren<PlatformData>()._isSolidPlayer1 = false;
343                         platformArray[i, j].gameObject.GetComponentInChildren<PlatformData>()._isSolidPlayer2 = false;
344                     }
345                 }
346             }
347         }
```

Figure 10: *Sets a random path sequence according to the played condition (Shared/Distributed) and a randomized integer. The contracted code in the else conditional, line 303, is similar to the first conditional, yet with a random range from 3-6.*

Two crucial methods, which operate continuously throughout the game, are responsible for managing row activation and verifying correct platform interactions. Both methods rely on the variable "RowIndex," which serves as an indicator of the current row to inspect and determines whether the subsequent row should appear.

The initial method, "ActivateNextRow()" **figure [11]**, is part of the game's main loop. It takes the "RowIndex" as a parameter, which specifies the targeted row within the platform array to activate or move to its spawn

position. Before triggering the activation process, the method compares the current "RowIndex" with its previous value to determine if a change has occurred. This check ensures that the method is only invoked when necessary. When a change is detected, the method proceeds to activate the corresponding row in the platform matrix based on the updated "RowIndex" value. Furthermore, audio feedback is initiated with each method call, excluding the first one when "RowIndex" is less than 1. To prevent unnecessary repetitions, the "PreviousRowIndex" variable is updated accordingly. This variable keeps track of the previous "RowIndex" value and enables the method to identify changes effectively. By continuously tracking changes in the “RowIndex”, the method optimizes performance by avoiding redundant activations and ensures the efficient execution of subsequent calls.

```
166     public void ActivateNextRow(int rowToActivate)
167     {
168         if(PreviousRowIndex == RowIndex) { return; }
169
170         for (int targetRow = rowToActivate -1; targetRow < rowToActivate; targetRow++)
171         {
172             for (int j = 0; j < RowLength; j++)
173             {
174                 platformArray[targetRow, j].transform.GetChild(0).gameObject.GetComponent<Platform>().SpawnPlatform();
175                 Debug.Log("SpawnPlatform -100x");
176             }
177         }
178         if (RowIndex > 1) // Trigger Audio for each "New Row Call", except the first.
179         {
180             Realtime.Instantiate("RealtimeAudioObj", new Vector3(5, 0, 0), Quaternion.identity, new Realtime.InstantiateOptions
181             {
182                 ownedByClient = false, // True?
183                 preventOwnershipTakeover = false,
184                 destroyWhenOwnerLeaves = false,
185                 destroyWhenLastClientLeaves = true
186             });
187         }
188
189         PreviousRowIndex = RowIndex;
190     }
```

Figure 11: *The "ActivateNextRow()" method checks for an increment of "RowIndex" to determine if the next row should be displayed along with audio feedback*

The second method, “CheckForCorrectPath()” **figure [12]**, runs continuously by traversing through the platform-array, while tracking the number of activated platforms in the given row, in respect to “RowIndex”. Each object of platform-array is checked and in case of activation, “NumOfPlatformsActivatedInRow” is updated. If this variable surpass or is equal to two, “RowIndex” is incremented, effectively prompting the previously presented method “ActivateNextRow()”. In case “RowIndex” exceed four and players have not already been forced into hell, a coroutine will be triggered to accomplish this task.

```
226     public void CheckCorrectPath(int rowToCheck)
227     {
228         for (int i = 0; i < rowToCheck; i++)
229         {
230             for (int j = 0; j < RowLength; j++)
231             {
232                 if (platformArray[i, j].transform.GetChild(0).gameObject.GetComponent<Platform>().GetPlatformActivated())
233                 {
234                     NumOfPlatformsActivatedInRow++;
235                     Debug.Log("NUmberOFPlatformsActivatedInRow INCREMENTED: " + NumOfPlatformsActivatedInRow);
236                     platformArray[i, j].transform.GetChild(0).gameObject.GetComponent<Platform>().SetPlatformActivated(false);
237                     if (NumOfPlatformsActivatedInRow >= 2) // if 2 or more
238                     {
239                         RowIndex++;
240                         NumOfPlatformsActivatedInRow = 0;
241                         Debug.Log("CheckCorrectPath Method - Row Index increased! : " + RowIndex);
242                     }
243                     if (RowIndex > 4 && !forcedIntoHell) // Forced into hell:
244                     {
245                         Realtime.Instantiate("RealTAudio_ForcedHell", new Vector3(5, 0, 0), Quaternion.identity, new Realtime.InstantiateOptions
246                         {
247                             ownedByClient = false, // True?
248                             preventOwnershipTakeover = false,
249                             destroyWhenOwnerLeaves = false,
250                             destroyWhenLastClientLeaves = true
251                         });
252                         StartCoroutine(ForcedIntoHell(13));
253                         forcedIntoHell = true;
254                     }
255                 }
256             }
257         }
258     }
```

Figure 12: *This function continuously checks for successfull interactions with all platforms to determine when "RowIndex" should be increased.*

5.6 Platform

The Platform script is a component attached to the platform prefab. Thus, every instantiated platform includes the platform script. It has various properties and methods for managing the behavior of each platform. As opposed to the "PlatformManager", The Platform class is responsible for handling the functionality of individual platforms in the game. It maintains properties such as "IsSolid", (indicating whether the platform is solid) and "timer" (used for tracking time). It also manages the audio and visual aspects of the platform, including playing sounds and changing materials. The class utilize and relies on components such as colliders, audio sources, and mesh renderers to achieve its functionality. It interacts with other scripts, such as the GameManager, to access game data and synchronize the state of the platforms.

The Platform class includes methods for setting the material of the platform, resetting the material timer, and triggering actions based on player interactions. These actions include checking if the platform is solid for specific players, triggering success conditions, and handling platform destruction. Additionally, the class contains methods for despawning and spawning platforms, as well as managing countdown timers and text objects associated with the platforms. The Platform class is a crucial component of the game, handling the logic and behavior tied to each individual platform.

In the following paragraph, code snippets will showcase and highlight the most crucial aspects of the Platform class. These snippets will demonstrate key functionalities, properties, and methods.

"CheckPlatformForPlayers()" is a method called in the "OnTriggerStay()" function of the platform. A guard clause in the Ontrigger-method will ensure that colliders contain the tag "Player", before proceeding with further execution. "CheckPlatformForPlayers() in **figure [12]**" is repeatedly called if a player collides with any given platform. Part of the functionality is solely performed by the master client to avoid inconsistency in the game loop. This functionality includes repositioning of the platform and tracking successful interactions

which relate to changes in certain game variables. Common functionality for both clients involve tracking interaction time to provide proper audio and visual feedback simultaneously.

```
304     public void CheckPlatformForPlayers(bool isPlayerServer)
305     {
306         if (!GameManager.IsServer)...
334         if (GameManager.IsServer)...
358         if (!GameManager.IsServer) { return; } // only server checks following.
359         if (isPlayerServer && syncedPlatformVariables._isSolidPlayer1)
360         {
361             successTimer += Time.deltaTime;
362             if (successTimer >= TimerThreshold -1)
363             {
364                 Success();
365                 setActivatedMaterial = true;
366             }
367         }
368         else if (!isPlayerServer && syncedPlatformVariables._isSolidPlayer2)
369         {
370             successTimer += Time.deltaTime;
371             if (successTimer >= TimerThreshold - 1)
372             {
373                 Success();
374             }
375         }
376         else
377         {
378             timer += Time.deltaTime;
379             GlassCracking();
380             if (timer >= TimerThreshold)
381             {
382                 PlatformFall();
383                 NumberOfPlatformsDestroyed++;
384             }
385         }
386     }
```

Figure 13: *A method that checks the interaction between player and platform to decide the correct action of the given platform*

Nested method calls have been used for optimized organization of the code throughout the implementation. “PlatformFall()” is an example of this as it is called within “CheckPlatformForPlayers()”. “PlatformFall()” furthermore invokes another method to reposition the platform by calling a Hdespawn method. As stated in the former paragraph, altering the transform of the platform is exclusively performed by the master client although it will appear similar to both clients as all transforms of any platform is networked and Tthus synchronized. The remaining part of this method simply resets dependent variables and trigger one of three audio clips by a classic switch statement.

```
249         public void PlatformFall()
250         {
251             int randomAudio = Random.Range(0, 3);
252             switch (randomAudio)
253             {
254                 case 0:
255                     audioSource.PlayOneShot(VanishSounds[0]);
256                     break;
257                 case 1:
258                     audioSource.PlayOneShot(VanishSounds[1]);
259                     break;
260                 case 2:
261                     audioSource.PlayOneShot(VanishSounds[2]);
262                     break;
263                 default:
264                     audioSource.PlayOneShot(VanishSounds[1]);
265                     break;
266             }
267             DespawnPlatform();
268             timer = 0;
269             successTimer = 0;
270             CountDownSuccessTimer = 0;
271         }
```

Figure 14: *In case of wrongful interaction PlatformFall() is invoked, effectively despawning the platform and trigger audio feedback accordingly*

5.7 Levers

Levers were used as a core mechanic in the hell-environment which required all six to be activated in order for the elevator to bring the players back up to heaven. Half of the levers were assigned to one player and the other half for the other player. The levers could only be pulled by their assigned player and thereby encouraged cooperation and communication in the shared task focus conditions and in the distributed conditions the levers were divided on two separated halves of the VE. The levers worked as a simple yet effective mechanic that would fit in the gameplay loop and worked as a great way to promote the intended kinds of interaction between all conditions.

```
Unity Message | 0 references
void Update()
{
    CheckElevatorPosition(); // Sets resetCondition for lever to true.
    CheckForResetLever(); // Resets levers

    if (syncedLeverData._leversPulled == 1 && !colorSet)
    {
        mat = meshRenderer.material;
        mat.SetColor("_EmissionColor", Color.green);
        audioSource.PlayOneShot(pulled, 0.7f);
        colorSet = true;
    }
    else if (syncedLeverData._leversPulled == 0)
    {
        mat = meshRenderer.material;
        mat.SetColor("_EmissionColor", Color.red);
    }
}

Unity Message | 0 references
private void OnTriggerStay(Collider other)
{
    if (other.GetComponentInParent<PlayerData>()._isServer && gameObject.GetComponent<LeverIdData>()._leverId == 1)
    {
        if (!wasPulled && other.tag == "Hands" && (other.GetComponentInParent<PlayerData>()._backupBool || other.GetComponentInParent<PlayerData>()._isReady))
        {
            gameObject.GetComponent<RealtimeTransform>().RequestOwnership();
            if (this.gameObject.name == "Lever_front(Clone)")
            {
                this.transform.rotation = Quaternion.Euler(135, 180, 0);
            }
            else if (this.gameObject.name == "Lever_back(Clone)")
            {
                this.transform.rotation = Quaternion.Euler(135, 0, 0);
            }
            else if (this.gameObject.name == "Lever_left(Clone)")
            {
                this.transform.rotation = Quaternion.Euler(135, 90, 0);
            }
            else
            {
                this.transform.rotation = Quaternion.Euler(135, 270, 0);
            }
            if (!other.GetComponent<RealtimeTransform>().isOwnedLocallySelf) { return; }
            syncedLeverData._leversPulled = 1; // Now means that its pulled and should set color.
            GameManagerReference.GetComponent<GameManagerData>()._level++; // A variable to keep track of how many levers has been pulled.
            wasPulled = true;
        }
    }
    if (!other.GetComponentInParent<PlayerData>()._isServer && gameObject.GetComponent<LeverIdData>()._leverId == 2)...
}
```

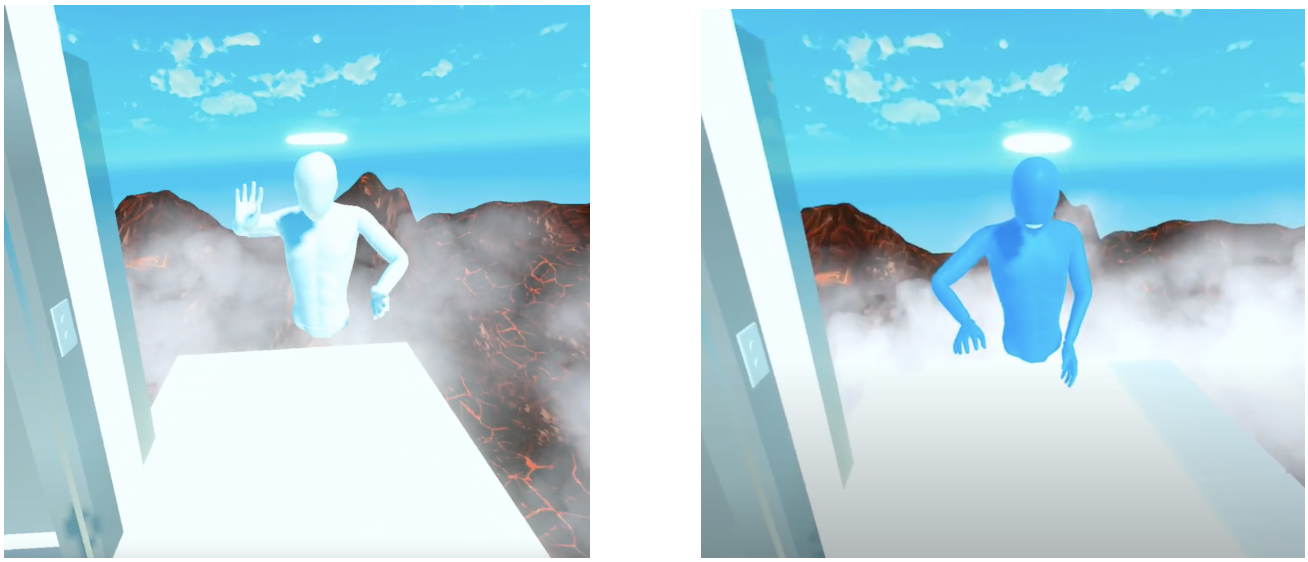
Figure 15: *This script is responsible for checking for collisions with users, update and sync state between clients (position, rotation and color) as well as resetting when the condition is met.*

The lever behaviour script would continuously check for collisions with the players' hands, if the 'grab' button was being held down and if the player was assigned to that lever, if so, the lever would rotate towards its final rotation, change material color and play a small sound to signify that it was activated. A synced game variable would increment for every lever pulled and once all were pulled the elevator would descent and the narrator would instruct the players.

5.8 Feedback

5.8.1 Avatar

Animating the players' avatars were done with inverse kinematics meaning that the virtual rig followed the physical position of the user based on the HMD and controller inputs. Furthermore, synchronized animations were needed when pulling the 'trigger' or 'grab' button on the controller in order for the other player to see the change occur visually. This was done with synced game variables that checked for player1 or player2 input and updated the state accordingly when a change was detected, allowing both users to receive the information. A mouth mesh attached to the head of both avatars was also synced and animated according to the input of the headsets microphone to allow for the users to see when the other person was talking.



(a) Avatar before game start (plain color)

(b) The Avatar after game initialization

Figure 16: The player avatar created with inverse-kinematics

5.8.2 Animations

A few simple custom animations were created with Unity's animation system. These included the elevator transporting the players back up to heaven as well as the gate into heaven. These animations provided visual feedback to the users allowing them to detect and verify changes in the VE that could help them better understand the state of the game. Ideally, this feedback would also assist the users in completing their goals more efficiently without misinterpreting each other unnecessarily.

5.8.3 Canvas

Unity's canvas was used for fading the screen to black as well as providing on screen information about the state of the game. The fading effect was done by overlaying a black image in front of the camera and interpolating the alpha value from 0% to 100% in a few seconds. As can be seen in **figure [17]**, a few checks are made to determine if the fade effect should trigger. It only triggers if the client is owned locally and then invokes another method "fallinDownwards()", to check if there is a negative change in y-position of the player. Finally, it checks a hardcorded y-interval, to determine whether the canvas should continue to fade. The fade effect was needed when users were falling from heaven to impede motion sickness as this could potentially have undesired effects in terms of simulator sickness that would have confounded the results. Furthermore, the fading effect worked as a boundary [2.5.1] when users moved into walls to dissuade them from walking through obstacles in the VE as there would be no physical way to prevent this.

```
98      public void ActivateFadeWhenFalling()
99      {
100         if (!GetComponent<RealtimeTransform>().isOwnedLocallySelf) return;
101         if (!fallinDownwards()) { return; } //Check if falling = Negative Y change
102         if (transform.position.y > -89 && transform.position.y < -3)
103         {
104             if (imageColor.a < 1f)
105             {
106                 fadeTimer += Time.deltaTime;
107                 imageColor.a += (fadeTimer / 0.5f) * Time.deltaTime;
108                 imageComponent.color = imageColor;
109             }
110         }
111
112     }
```

Figure 17: *A method to determine if the local player itself is falling, and if so, fade the canvas.*

The canvas was also used to deliver on-screen information about the state of the game or instructions on how to proceed when necessary. A TextMesh would appear on the canvas and prompt players to "move out of the wall" when stuck or letting them know that "the elevator has arrived" to guide them back to the starting point. Additionally, TextMeshes were also present in the hell environment and indicated how many levers had been pulled to guide users towards their objective.

5.8.4 Audio

Unitys 3D spatial audio was used to provide feedback within the VE. Unity's 'AudioSource' component was attached to game objects in the scene allowing them to play audio clips when needed. The following game objects had auditory feedback when interacted with: platforms, levers, elevator and button. These objects were able to deliver instant auditory and visual feedback when interacted with in order to guide the users and reinforce understanding of the game mechanics and state of the game. The narrator provided auditory instructions to the users several times throughout the experience. Furthermore, ambient sounds was also present to create a soundscape that would compliment the setting (whether in heaven or hell) and ideally to increase immersion and engagement. Lastly, When the win condition was true, the game would also play a song as well as write to the canvas to notify users about the outcome of the game.

5.8.5 Visuals

The art-style of the assets ad the environment is stylized but semi-realistic, this was deemed the optimal solution as complex trigonometry of models heavily impacted the performance of the HMDs. Therefore, simpler representations of objects with fewer polygons became the most optimal solution without sacrificing too much in terms of visual impression and realism. Visual effects were also implemented to give feedback to the users about several aspects relevant to the state of the game. Particle effects were implemented in both environments, and acted as cosmetics but also to provide information about the intended positions of the users in the distributed condition of the hell environment. Post-processing effects were used along with Unity's universal render pipeline to create glowing materials that would represent props of importance to the games progression. Unity's built-in terrain component was used to create an immersive environment consistent with the theme of the game. Additionally, textures and texture-painting was also heavily used to obtain the right impression about the environment and style of the game. The skybox within the VE would dynamically change depending on the position of the user (y-value) to update the background depending on the location of the user. To represent the users within the VE, a neutral mannequin model was chosen to represent the users in an objective way without inferring anything in terms of the player model.

5.9 Shared vs Distributed Toggle

One of the requirements for the application was to account for the two conditions that should be tested. These conditions should furthermore be easy to switch between, to streamline testing. A static boolean "IsTaskShared" in the "GameManager" class, was created and used throughout the implementation. The variable is checked in several scripts to determine whether the application should include/exclude certain environment designs, scene elements and functionality. For example, the sequence of the "solid" platforms is altered according to which condition is played. In the shared condition, players are obliged to cross each others pass multiple times, as opposed to the distributed condition, where a certain distance between players are maintained. The GameManager furthermore contains a reference to two GameObjects, that include two different editions of the hell-scene environment, as can be seen in **figure [6]** . These scene elements are toggled according to the played condition. To streamline testing, two different builds were created, where the only modification was a toggled boolean, that fundamentally changed the game from shared functionality to distributed.

5.10 Telemetry Data

The Telemetry script in **figure [18]** was created to track and store telemetry data of players. Data logging has been reduced solely to head position as this includes the most accurate measurement of players position relative to each other. The class contains two string variables, "headPosPath1" and "headPosPath2," representing the file paths where the telemetry data will be stored. These file paths are constructed by appending the current datetime string to the directory paths.

```
8   public class TelemetryData : MonoBehaviour
9   {
10      public Dictionary<int, RealtimeAvatar> avatars;
11      static string format = "Mddyyyyhhmmsstt";
12      static string datetime = DateTime.Now.ToString(format);
13
14      string headPosPath1 = @"/Users/bjornwinther/Desktop/TelemetryData/1headPosPath" + datetime + ".txt";
15      string headPosPath2 = @"/Users/bjornwinther/Desktop/TelemetryData/2headPosPath" + datetime + ".txt";
16
17      void Start()
18      {
19          if (Application.platform == RuntimePlatform.Android) { return; }
20      }
21      void Update()
22      {
23          if (Application.platform == RuntimePlatform.Android) { return; } // only computer does following:
24          if (!GameManager.GameStarted) { return; }
25          avatars = GetComponent<GameManager>().Avatars;
26          for (int i = 0; i < avatars.Count; i++)
27          {
28              RealtimeAvatar player = avatars[i];
29              if (player.isOwnedLocallySelf) { return; }
30              int playerNumber = player.gameObject.GetComponent<PlayerData>()._backupInt;
31
32              Vector3 headPos = player.gameObject.transform.Find("Head").transform.position;
33
34              if (playerNumber == 1)
35              {
36                  File.AppendAllText(headPosPath1, Time.time.ToString() + " : " + headPos.x + " : " + headPos.y + " : " + headPos.z + "\n");
37              }
38              if (playerNumber == 2)
39              {
40                  File.AppendAllText(headPosPath2, Time.time.ToString() + " : " + headPos.x + " : " + headPos.y + " : " + headPos.z + "\n");
41              }
42
43          }
44      }
45  }
```

Figure 18: *The Telemetry data script responsible for tracking and storing the head positions of all connected android devices*

As with any script in Unity, the "Update()" function runs continuously throughout the game. The initial guard clause will check the application platform, to ensure that the data is being stored on a computer and not an android device, as well as circumventing unnecessary code execution on the Meta Quest 2 device.

If the game has not started yet, indicated by "GameManager.GameStarted", the method returns and does not proceed further. This counters a potential null reference bug, as the avatar dictionary otherwise would be empty. The avatar dictionary/list is of course a key component as it provides access to the positional data of each player. A loop iterates over each avatar in the "avatars" dictionary. For each avatar, it checks if the avatar is controlled by the local player ("player.isOwnedLocallySelf"), and if so, it skips to the next iteration of the loop. As mentioned, only the computer executes this part of the code. By checking if it is locally owned, storing the computers own stationary avatar data is prevented.

A reference to retrieve the player number from the "PlayerData" component is then made to distinguish each player. Subsequently the position of the avatar's head is obtained by finding the child gameobject named "Head" and accessing its position. Depending on the player number, the method appends the current time and head position coordinates (x, y, z) to the corresponding telemetry file. The data is appended using the "File.AppendAllText()" function. The script successfully captures and stores the head position of all connected non-computer devices each frame of the game.

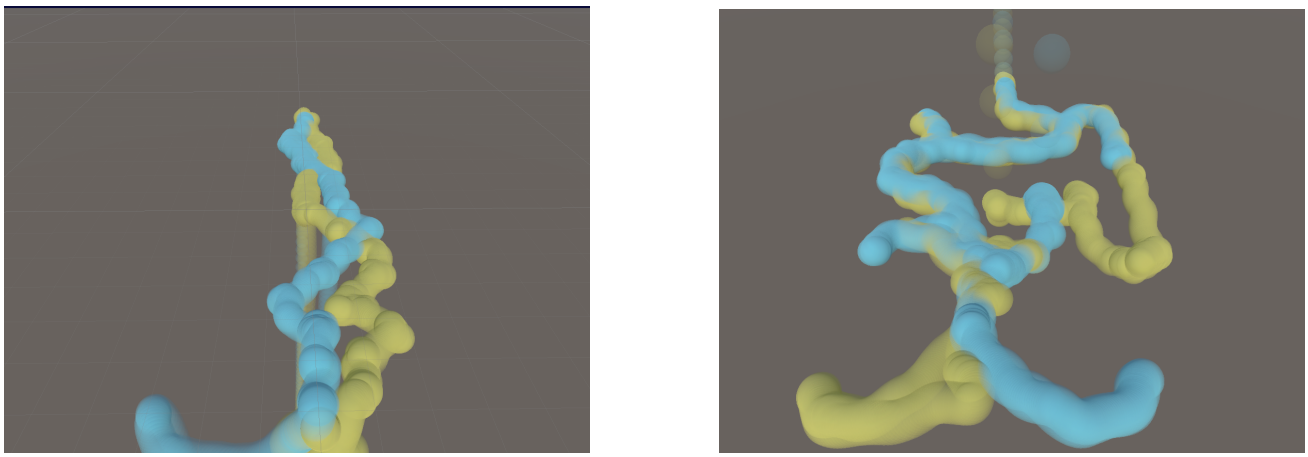
5.10.1 Showcasing the telemetry data

A new separate unity project was created to visualize the data collected by the telemetry script in **figure [18]**. This project is plain in nature, only including one script and two sphere prefabs with individual materials. The materials are yellow and blue respectively (to represent each player), with a low alpha value, to appear semi-transparent. When several semi-transparent objects are spawned on top of each other or in a cluster, transparency will decrease. This procedure allows for visualizing players movement, as well as prominent and less prominent areas of the map. The method to visualize the positional data can be seen below, in **figure [19]**.

```
19      void ReadTelemetryData(string filePath, GameObject prefab)
20      {
21          if (File.Exists(filePath))
22          {
23              string[] lines = File.ReadAllLines(filePath);
24
25              for (int i = 1; i < lines.Length; i += 2)
26              {
27                  string line = lines[i];
28                  string[] data = line.Split(':');
29                  float time = float.Parse(data[0]);
30                  float x = float.Parse(data[1]);
31                  float y = float.Parse(data[2]);
32                  float z = float.Parse(data[3]);
33                  Instantiate(prefab, new Vector3(x, y, z), Quaternion.identity);
34              }
35          }
36          else
37          {
38              Debug.LogError("Telemetry file not found: " + filePath);
39          }
40      }
```

Figure 19: *The method parses the telemetry data from a text file and creates an object for every second line, using the corresponding values provided in the file.*

The script to visualize the data, contains a single method, shown in the figure above, which is called twice in the "start()" function. Each method call includes a specific filepath and a unique prefab corresponding to each player. An example of the result can be seen below in **figure [18]**



(a) Heaven (Shared condition)

(b) Hell (Shared condition)

Figure 20: Heaven and Hell telemetry data (Non-colocation shared condition)

The figures above showcase the output. The outcome closely resembles a heatmap representation. The telemetry text files contains positional data for every frame. To limit the amount of instantiated gameobjects, the script only spawns an object for every second line. The utilization and results of the telemetry data will be discussed further in section [7.2.7]

6 Results

This chapter presents the results of data collection and statistical analysis. The data collected is quantitative and includes subjective self-report measures of co-presence, attentional allocation, perceived behavioral interdependence and collision avoidance. In order to account for confounding variables when testing a randomized condition order will be implemented. Likert scales were used ranging from 1 (Not at all) to 7 (Very much/well) and all variables were measured this way. The data was collected from a sample of the target population, consisting of 32 individuals (16 pairs). The participants were exposed to a collaborative multiplayer game in a virtual reality environment. A mixed factorial design will evaluate the effects of co-location and task focus between all four conditions. The "IBM SPSS Statistics" software was used for descriptive statistics and statistical analysis of the data. Given the modest sample size, the Shapiro-Wilk test was chosen to assess the normality of the data, using an alpha level of 0.05. The conducted normality tests reveals that the data generally departs from normality, however, some isolated measures appear to be normally distributed but as the whole category will be evaluated non-parametric tests will be used to check for significant differences in the data. Upon preliminary inspection of the boxplots of the data it becomes clear that the distribution of the data is generally either very right-skewed or left-skewed depending on the variable measured. This further supports the claim that the data is not normally distributed, however, several Shapiro-Wilk tests will be conducted in the different conditions to check for normality regardless. The non-parametric test used is the Wilcoxon Signed Rank Test.

6.1 Social presence

6.1.1 Co-Presence

A Shapiro-Wilk test was used to test for normality in the four groups in regards to co-presence. It was shown that for the co-located shared condition ($W = 0.405$, $p = < 0.001$), the co-located distributed condition ($W = 0.644$, $p = < 0.001$), the non co-located shared condition ($W = 0.405$, $p = < 0.001$) and the non co-located distributed condition ($W = 0.398$, $p = < 0.001$) that all departed from normality. As the value $p < 0.05$ in all conditions the null hypothesis that the data is normally distributed must be rejected. In the case of co-presence the data departs from normality and therefore a Wilcoxon Signed Rank test will be used to check if there is a statistically significant difference in the mean co-presence measures between all four groups.

The Wilcoxon Signed Rank Test showed that for the co-located shared vs distributed conditions ($z = -0.736$, $p = 0.461$). Since the p-value is greater than 0.05, we can not reject the null hypothesis. We have sufficient evidence to conclude that the task focus did not have a statistically significant effect on the co-presence between these conditions. In the case of the non co-located shared vs distributed conditions ($z = -0.962$, $p = 0.336$) reveals that there is no significant difference in terms of co-presence in these two conditions either. For the co-located shared vs non co-located shared it shows that ($z = -1.633$, $p = 0.102$) and furthermore the co-located distributed vs non co-located distributed showed that ($z = -1.382$, $p = 0.167$) as both of these tests have $p > 0.05$ we must also conclude that there are no significant difference in co-presence across all four conditions when compared in a mixed factorial design.

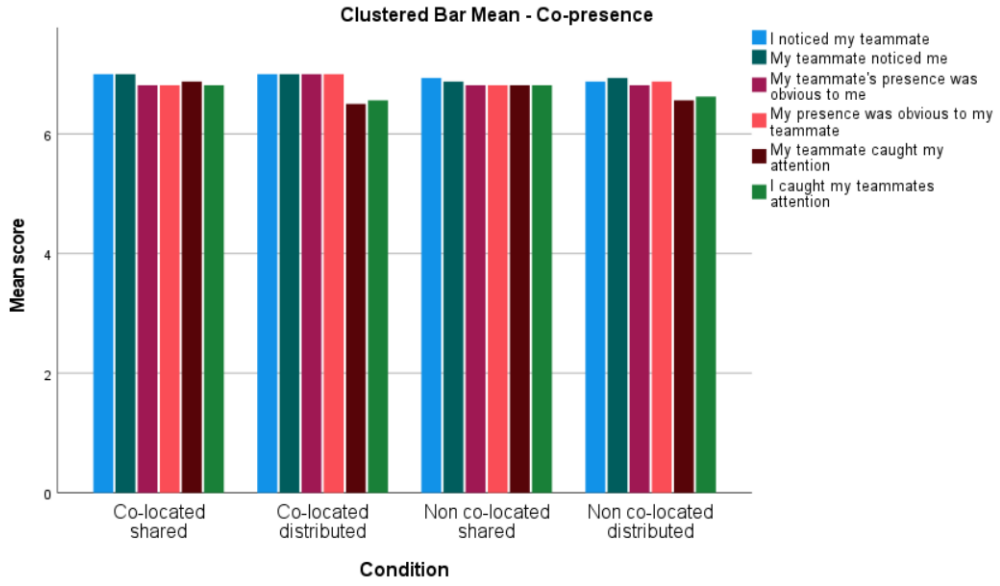


Figure 21: *Co-presence distribution - Clustered bar mean*

6.1.2 Attentional Allocation

In relation to attentional allocation it was shown that for the co-located shared condition (W = 0.826, p = 0.006), the co-located distributed condition (W = 0.823, p = 0.006), the non co-located shared condition (W = 0.843, p = 0.011) and the non co-located distributed condition (W = 0.887, p = 0.049). Once again the data appears to be departing from normality as $p < 0.05$ in all conditions. Once again the Wilcoxon Signed Rank test will be used to check for any significant differences in the data.

The Wilcoxon Signed Rank Test is once again used to check for significant differences between the four conditions in the attentional allocation category. The co-located shared vs distributed conditions showed that (z = -1.682, p = 0.093) which shows no significant differences. The non co-located shared vs distributed condition (z = -1.782, p = 0.075) again showing no significant differences. In the non co-located shared vs co-located shared conditions (z = -0.954, p = 0.340) there are no significant differences. Finally, in the co-located distributed vs non co-located distributed it shows that (z = -1.997, p = 0.046) which reveals a significant difference between these two conditions in terms of attentional allocation.

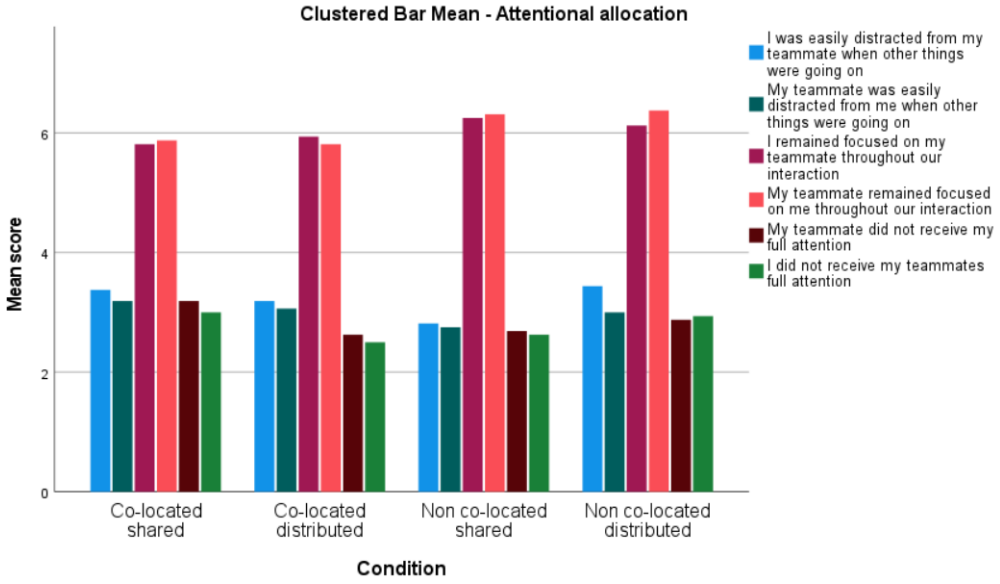


Figure 22: *Attentional allocation distribution - Clustered bar mean*

6.1.3 Perceived Behavioral Interdependence

In the case of perceived behavioral interdependence the co-located shared condition showed that (W = 0.845, p = 0.011) the co-located distributed condition (W = 0.796, p = 0.002) the non co-located shared condition (W = 0.771, p = 0.001) and for the non co-located distributed condition (W = 0.697, p = 0.001). These results show that the data departs from normality once more as all $p < 0.05$ in all conditions. The Wilcoxon Signed Rank test will also be used for this variable.

The Wilcoxon Signed Rank Test revealed that in the co-located shared vs distributed conditions (z = -2.201, p = 0.028) which shows that between these two conditions the means of the perceived behavioral interdependence is significantly different. In the non co-located shared vs distributed conditions (z = -0.707, p = 0.480) no significant difference exists. In the non co-located shared vs co-located shared conditions (z = -2.207, p = 0.027) a significant difference exists as $p < 0.05$. Lastly, in the non co-located distributed vs co-located distributed (z = -0.316, p = 0.752) no significant difference is found in terms of perceived behavioral interdependence.

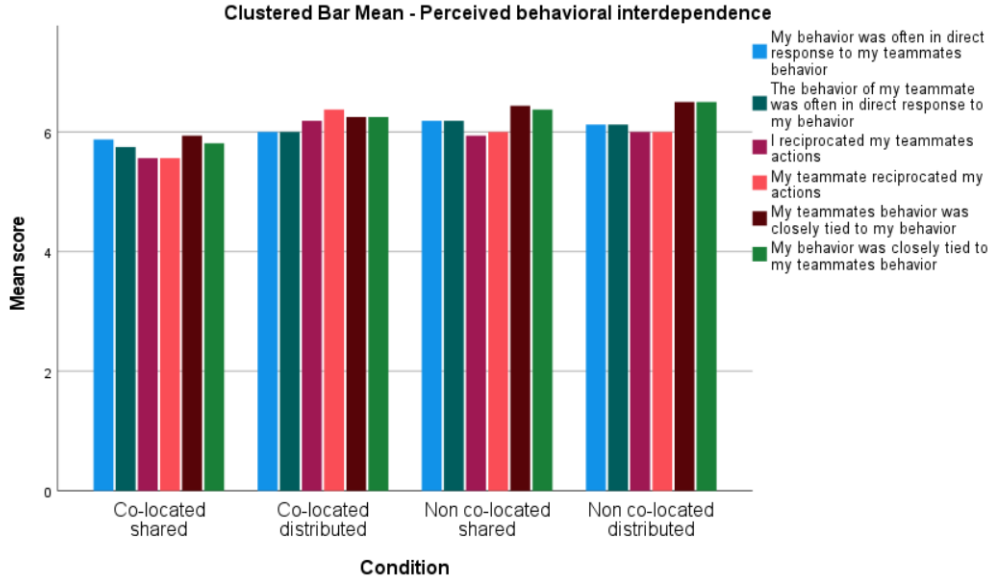


Figure 23: *Perceived behavioral interdependence distribution - Clustered bar mean*

6.2 Collision Avoidance

For collision avoidance, the co-located shared condition showed that (W = 0.880, p = 0.038) the co-located distributed condition (W = 0.873, p = 0.031) the non co-located shared condition (W = 0.883, p = 0.042) and for the non co-located distributed condition (W = 0.816, p = 0.004) all conditions show that the data departs from normality and a Wilcoxon Signed Rank test will be used to check for significant differences.

The Wilcoxon Signed Rank Test revealed that in the co-located shared vs distributed conditions (z = -1.633, p = 0.102) the non co-located shared vs distributed conditions (z = -1.604, p = 0.109) the non co-located shared vs co-located shared conditions (z = -1.633, p = 0.102) and lastly for the co-located distributed and non co-located distributed conditions (z = -1.604, p = 0.109) that no significant differences exists between the means of the data.

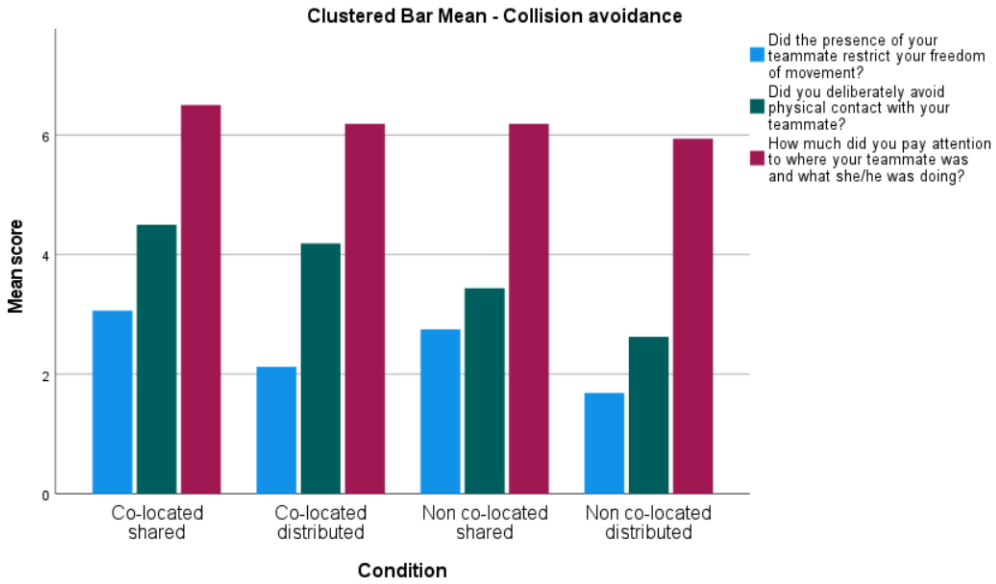


Figure 24: *Collision avoidance distribution - Clustered bar mean*

7 Discussion

7.1 Results discussion

This section aims to elaborate on the study's findings. The gathered results will be analyzed and interpreted, while potential causes will be discussed. The section will include an examination of the results regarding, co-presence, attentional allocation, perceived behavioral interdependence and collision avoidance.

7.1.1 Social Presence Evaluation

The following three sub-dimensions of social presence were evaluated:

- Co-presence
- Attentional Allocation
- Perceived Behavioral Interdependence

All above sub-dimensions were evaluated in a mixed factorial design (four conditions) and all the overall measured differences between them in terms of co-presence were deemed insignificant. Co-presence refers to the level of presence experienced by the participant and their partner within the social context. It encompasses the extent to which the partner feels present in the interaction and how present the participant perceives themselves to be in the same social setting. It can therefore be argued that co-location and a shared task focus does not provide a better social connection with the partner in this context.

Attentional allocation refers to the extent of participants' focus on social dynamics. It describes the level of attention allocated towards comprehending and actively participating in the interpersonal elements of a specific situation or context. Based on the attentional allocation data gathered, there is insufficient evidence to suggest that three of the conditions encourages participants to allocate a significantly different amount of attention towards social dynamics. However, in the co-located distributed vs non co-located distributed condition a significant difference is found in terms of attentional allocation. This indicates that in a co-located physical and virtual environment with a distributed task focus participants allocate a significantly different amount of attention to their teammate.

Perceived behavioral interdependence focuses on the social behaviors exhibited between partners. It pertains to the way in which the actions of your partner influence your own behavior and vice versa. Based on the data collected in this category it can be argued that participants did not exhibit a higher tendency to either follow or reciprocate their partners actions in the non co-located shared vs non co-located distributed condition as well as the co-located distributed vs non co-located distributed condition as no significant differences were found. However, in the co-located shared vs co-located distributed condition and in the co-located shared vs non co-located shared condition significant differences were found suggesting a higher tendency to either follow or reciprocate their partners actions in these conditions as $p < 0.05$.

7.1.2 Collision Avoidance Evaluation

Collision avoidance relates to assessing individuals' subjective experiences related to collision avoidance behavior inside the VE. Based on the data collected in this category it can be argued that none of the participants experienced a higher focus on avoiding collisions in any of the conditions as there are no significant differences between any of the conditions ($p > 0.05$).

It is interesting that there was no significant difference among any of the conditions in terms of self-reported collision avoidance behavior. It should be kept in mind that the collision avoidance measure was not a validated and standardized questionnaire, which is why the data is potentially ambiguous.

7.1.3 Significant differences in findings

There were a few significant differences found between sub-conditions as can be seen in **figure [25]**.

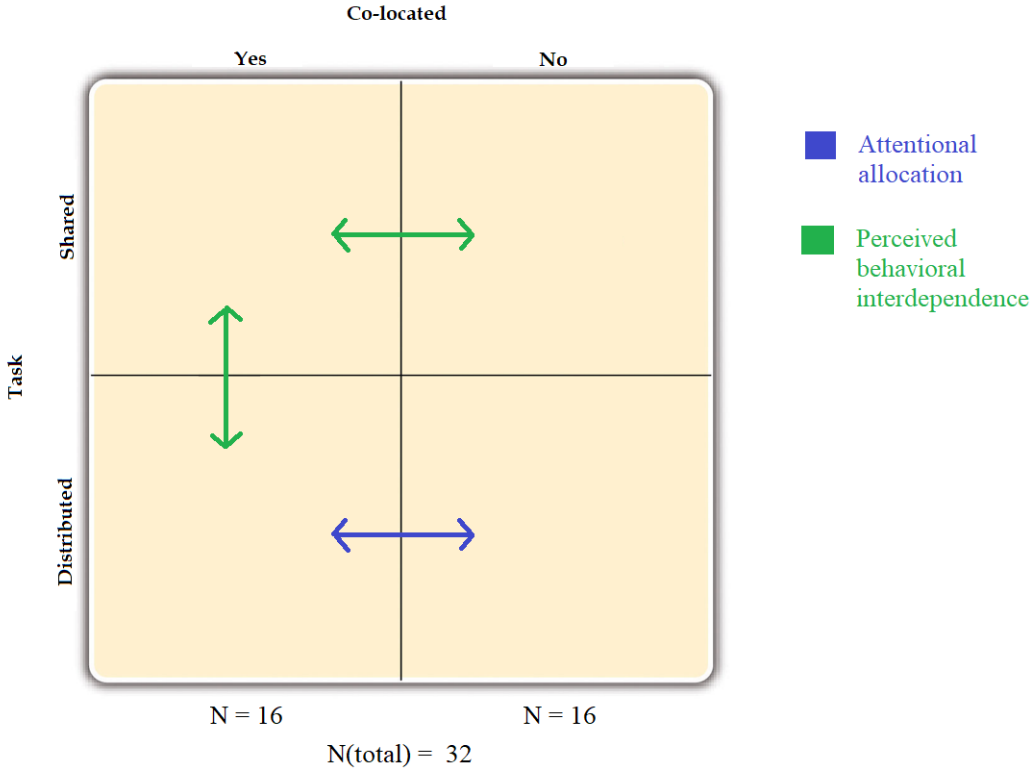


Figure 25: *Significant differences between the conditions visualized*

Attentional allocation showed a difference between co-location and non co-location in the distributed (task dependency) condition. The results indicate increased attentional allocation in the non-colocated condition, as compared to co-located.

Perceived behavioral interdependence showed a significant difference between the co-located shared and distributed condition. This indicates that the condition altered design of the system works. Interestingly, the users exhibit improved behavioral interdependence in the distributed condition, compared to the shared condition.

As depicted in the above figure, a significant difference was also found between the co-located shared and non co-located shared conditions when measuring perceived behavioral interdependence. Although a modest sample size, the results indicate slightly better feelings of perceived behavioral interdependence and increased attentional allocation in the non co-located condition which aligns with what was found in section **[2.3.1]**.

7.1.4 Motion sickness confound

A preliminary measure of prior motion sickness was conducted to account for confounding variables tied to motion- and simulator sickness when exposed to VR. As can be seen from **figure 26** the distribution of prior experience with motion sickness reveals that 24 participants (75%) has never or rarely experienced motion sickness when exposed to VR. The remaining eight participants indicated that they sometimes or frequently experience motion sickness when exposed to VR. However, it was deemed that there was no reason to suspect that motion sickness would differ between conditions as input, control and visual environment stayed consistent throughout testing. Test observations also supported this claim as there were no visual or auditory indications that participants experienced a higher degree of simulator sickness during exposure.

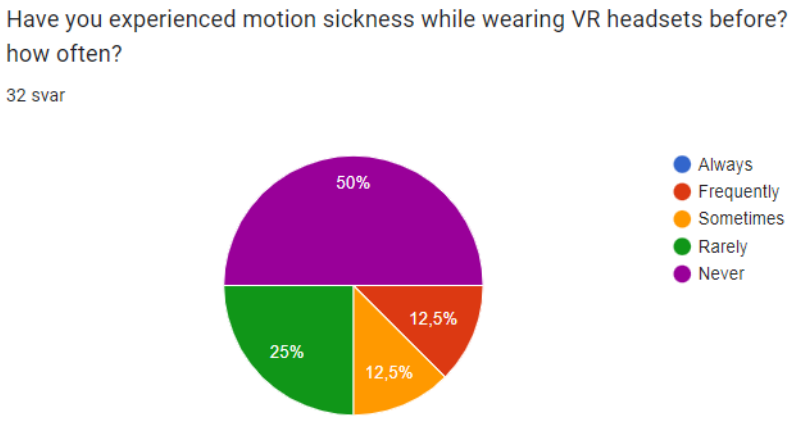


Figure 26: *Preliminary motion sickness experience measure*

7.1.5 Test Observations

Several observations were made during testing that potentially affected the game experience. These observations will be listed here as mediating factors that potentially played a role in positively or negatively shaping the final results. Some of these observations might not be detrimental to the game experience or the results but will be listed regardless as considerations of the testings.

Observations during the interventions ascertained that participants communicated both through gestures and verbal communication. Gestures identified included pointing, crouching and touching platforms to clarify to their teammate which one to step on next. Participants also interacted with one another inside the VE by dancing, boxing, waving and touching each other which presumably sensitized them to the VE to a certain extent. Participants also heavily utilized the 3D spatial audio in both the co-located and non-colocated conditions to locate and guide each other inside the VE to their respective objectives. Observations by the researchers determined that there was a high degree of confidence in terms of spatial navigation, the users' movements clearly indicated trust in the system as well as the representation of objects and the other user's avatar inside the VE.

When navigating the platforms different forms of communication were used to identify and guide the other user to their assigned platforms. Some participants indexed the platforms from either the left or the right side, some participants simply resorted to give directional information (e.g. "up and left", "beside me", "to my right" etc.). Furthermore, when crossing paths on the platforms the users often coordinated who went

first and took turns prompting each other to cross before or after the other, this was especially true in the co-located condition but also occurred in the non co-located condition. Interestingly, in the non co-located condition the participants also avoided physical collisions despite being physically separated into different rooms. Some participants thought that they should synchronize stepping on the platforms or that there was a time limit to successfully complete that section, perhaps enforced by the fact that there were visual timers on every platform.

In the shared hell-section of the game most participants chose to split up and call upon each other when they encountered a lever that could only be pulled by their teammate. Some participants also resorted to stick together and switch pulling the levers depending on which player was assigned to it. Some participants felt that they needed to synchronize pulling the levers while some participants thought that pulling the levers would have a detrimental effect on the games progression, perhaps enforced by the fact that the levers were red when inactive. Upon discovering that there were no other interactive elements in the hell-section the participants figured out that they had to interact with the levers in order to progress.

7.2 Technical discussion

The technical discussion will delve into the specific challenges related to the technical implementation of the system. These complications will be listed to give an overview of the issues encountered, be transparent about possible mediating factors as well as potential solutions for future work.

7.2.1 Low-fidelity prototyping

It would have been beneficial to employ a more play-centric approach by recruiting small numbers of participants and evaluating several iterations of low-fidelity prototypes along the development process. This would allow for valuable insights and early feedback on various aspects of the development, the core mechanics and direction of the game. From this preliminary data the target group would influence and steer the development towards the most crucial mechanics and visual impression, promoting usability, understanding and generally, an improved user experience.

7.2.2 Testing limitations

The sample size for this evaluation was rather modest (N = 32) and must be considered when examining the results. It would have been ideal to recruit more participants for testing as increasing the sample size would have increased the statistical power and generalizability of the findings as well as making the sample more representative of the population. Therefore, the statistical conclusions presented in this study are relatively ambiguous and the question remains whether a true effect actually exists.

7.2.3 Microphone input

In the co-located conditions the microphones on the HMDs were unintentionally left active, this introduced some potential mediating factors in the sense that participants could hear each other twice both in the real-world and VE. The active microphones were discovered too late by the researchers and as a result it was not possible to change the setup in future sessions as this would interfere with the reliability of the findings.

7.2.4 Restricting natural walking

In regards to the technical implementation it was a balance between affording the users with real walking but simultaneously restricting unintended movement inside the VE. It was not possible to impede the physical

movement of the participants in the real world and altering the position of their virtual avatars would only introduce desynchronization issues between clients. These limitations were constantly considered during development and often required game design modifications that were compatible with these requirements. One participant in the non co-located condition realized that walking directly through the virtual walls was a viable strategy as there were no physical obstructions to impede their movement. Another participant also realized that it was possible to skip the last row of platforms when moving fast enough as the timer would not trigger a reset and the win condition would become true. All other participants were observed to respect the virtual environment as well as the established boundaries and navigated it as intended by the researchers.
Furthermore, the system required extremely precise calibration between each condition as well as utilizing the fade effect on the canvas [5.8.3] to attempt to prevent any further physical movement of the user when colliding with unintentional virtual objects. Additionally, the participants were instructed to not exceed the virtual play-space [3.4] during the introduction segment of the game.

7.2.5 Lever count desynchronization

The synchronized variable responsible for keeping track of the amount of pulled levers were susceptible to desynchronization issues if two were pulled within a short time frame of each other. This proved to become an issue with two of the groups that mistakenly thought that they should pull two levers simultaneously resulting in the variable being accessed by two clients at the same time and only being updated once. This issue would be fixed with the 'Wizard of Oz' method where the researchers would manually increment the synchronized variable as the third client to match the in-game lever count.

7.2.6 Hardware issues

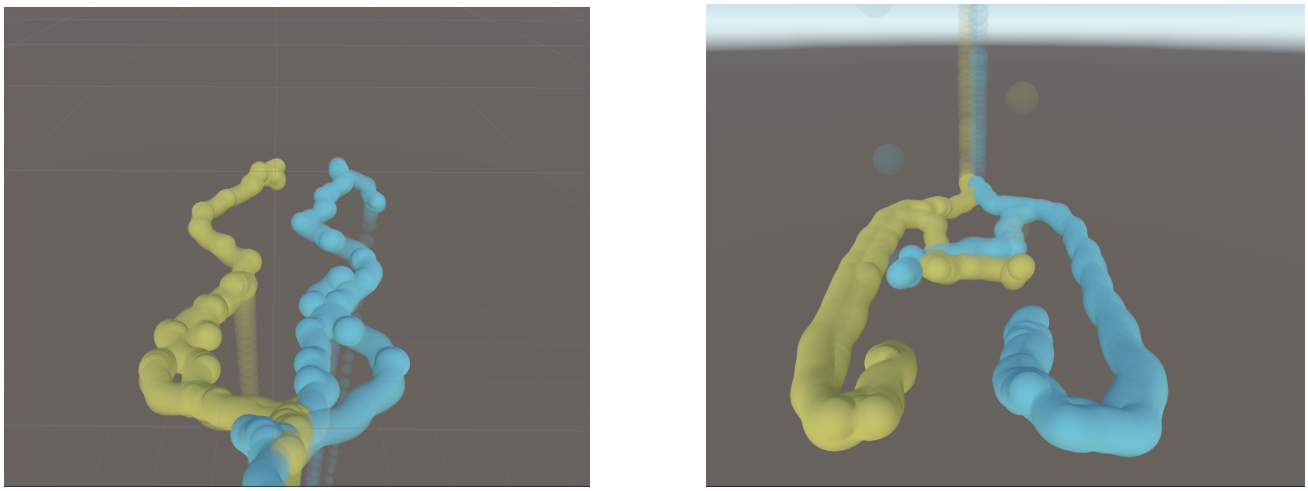
Both of the Oculus Quest 2 HMDs used for the evaluation experienced some difficulties in the co-located testing sessions. This was presumably caused by the physical lighting conditions which disrupted the sensors on the headset resulting in extremely low frame-rate, poor tracking and desynchronization. This however was fixed before every testing session and only once resulted in desynchronization between clients but quickly fixed itself again.

7.2.7 Telemetry Data

The collected telemetry data was not used for its intended purpose because of failed considerations. Initially it was meant to showcase movements of all tested subjects in an averaged heatmap-like representation. The intention was to identify navigational tendencies, with actual data representation to verify any claim made from the thesis. During the visualization process of the telemetry data, it was realized that it did not make sense to visualize movement data across different test sessions, as this would only provide a confusing, inconclusive result. The pathway in heaven alters between six paths that are randomly chosen, why it would only provide a chaotic and ambiguous result. The same goes for the hell environments, although one could argue that the hell distributed environment, might show a somewhat accurate representation, as it heavily encourage players to only roam on their half of the play-space.

Although the telemetry data is not able to show general tendencies by comparing data across all sessions, it can indicate whether game mechanics and functionality worked as intended by observing visualized telemetry data for each individual game session. As can be seen in the below **figure [27]**, the movement of both players are clearly depicted. It is clear that players understand and oblige the game mechanics in this session. In **figure [27a]**, the players are crossing the platforms to make their way to the heavens gate. Interactants paths do not cross and a certain distance is maintained between them, demonstrating the envisioned behavior and dynamic. It can clearly be seen that the subjects fall down around halfway through, indicating that they have triggered

the mechanic of being forced down into hell. In the hell environment in **figure [27b]**, the subjects are clearly seen, respecting each others play-space, which indicate that the implemented game mechanics, feedback and environment has worked as intended as well.



(a) Heaven (Distributed condition)

(b) Hell (Distributed condition)

Figure 27: Heaven and Hell telemetry data (Non-colocation distributed condition)

(a) Heaven (Shared condition)

(b) Hell (Shared condition)

Figure 28: Heaven and Hell telemetry data (Colocation shared condition)

In **figure [28]**, the portrayed movement data of the shared condition is depicted. Contrary to **figure [27]**, it is clearly displayed how subjects engage with each other in this condition. This again verifies the implementation of the changed environment, paths and mechanics that is toggled according to what condition is initialized. **Figure [28a]** clearly shows how the participants pathways cross several times in heaven, in addition to constantly being in close proximity to each other virtually and physically (as it is co-located). **Figure [28a]** shows the interactants movement in hell in the shared condition. The observed behavior clearly shows how they

exchange paths and even appear to follow each other. This of course can not be stated with absolute certainty, as the representation does not provide any way of evaluating the time of each instantiated object. Players are subsequently clearly seen making their way to the elevator in every figure.

One should keep in mind that the above four figures solely depict one game session out of 32, why the observed tendencies can not necessarily be generalized. One would have to examine telemetry data for each game session and develop a suitable evaluation method, in order to make a any definitive conclusive statements of the game mechanics. The visualized telemetry data did coincide with what was generally observed during the test sessions. Subjects generally obliged and used all game mechanics as intended.

7.3 Game design discussion

In this section, the impact of the game design on the player experience will be discussed, along with potential improvements for future development. It is important to note that the application's final state was a high-fidelity prototype, resulting in certain shortcomings. By closely observing the test sessions, significant flaws and areas where implementation was lacking were identified, and their implications will be explored in the following subsections.

7.3.1 Elevator

The elevator which was responsible for transporting players up to heaven had several issues related to it. Firstly, when changing the position of the elevator with an animation, desynchronization between clients would occur, attempting to synchronize the animation playback with a synced game variable yielded similar results. It was also attempted to use Unity's physics engine to move the elevator towards a target position with the players inside, collision logic with the avatars collider and the elevators collider were responsible for "pushing" the players towards the target position, but still introduced a tiny delay between clients. In order to solve this issue the researchers instructed players to be stationary after entering the elevator and pressing the button.

Furthermore, a slight oversight was the fact that upon calling the elevator one of the participants could theoretically take the elevator up to heaven and leave the other player behind. This only occurred with one group but introduced a severe issue where players would be desynchronized and would have no way to avoid collisions as they were on different levels (y-axis). However, the game design took this into account as the now solo player in heaven had no way of progressing without their partner. This meant that he/she were forced to go back down to hell in order to progress the game along with their teammate.

Additionally, the elevator also risks posing as a confounding variable, as participants would always be close in virtual proximity when using the elevator, no matter the condition. Ideally, the distributed condition would include two elevators, one for each player, to keep the virtually, and thereby physically separated (For the co-located test subjects). Alternatively, the elevator could have been wider and divided the floor in two confined spaces, respectively for each player. Although the elevator ride is only a brief part of the experience, it might have influenced the results.

7.3.2 Narrator

Upon starting the game the narrator would verbally instruct the participants on their objectives. Participants consistently missed the verbal instructions from the narrator presumably because they were focused on interacting with their teammate or becoming sensitized to the environment. To solve this issue, the researchers consistently briefed all participants on their objectives after the narrator had finished as participants would

have been clueless on how to progress if they did not receive this information. Furthermore, upon being forced into hell the narrator would similarly verbally instruct the players that they would have to face hell at least once in order to make it, this went relatively unnoticed as the game progressed despite this change of events. A solution to this issue would have been to stop incrementing the variable "RowIndex" when the forced-into-hell event started as to stop progression and force their focus towards the narrator. Similarly, this was mitigated by consistently instructing all participants on what had just happened after being forced into hell and that this was intended as a game mechanic.

7.3.3 Tutorial

The inclusion of a small introductory tutorial could have been beneficial in facilitating the understanding of fundamental game mechanics. This, in turn, would have allowed players to focus more on their objectives, the environment, and their teammate. Another approach could have involved incorporating a test round to familiarize players with the basic mechanics. However, it should be noted that such a test round might have introduced bias by providing additional exposure time in the VE in one condition. While a tutorial would have been the most optimal solution, it was not given priority due to the time constraints of the thesis.

7.3.4 Platform feedback

A rather infrequent confusion arose from the timers of the platforms. A few users thought that these timers indicated a need to hurry through the platform-section resulting in unintended failures by the participants, they did however discover that this was not the case upon re-entering heaven once more and took their time positioning themselves in the correct row and index.

8 Conclusion

This thesis attempted to answer the research question: "To what extent is social presence and collision behaviour affected by co-location and task dependency in a collaborative multiplayer (VR) environment. To investigate this research question, a virtual environment was developed and a mixed factorial design study including 32 participants was conducted. The collected data departed from normality and therefore a non-parametric test, the Wilcoxon Signed Rank test, was performed to evaluate whether a significant difference could be found.

Social presence can be broken down into six subcategories, three of which were deemed relevant to this study. These sub-dimensions were analyzed with the Wilcoxon Signed Rank test to find out whether or not social presence was improved/degraded in any of the conditions. Results indicated that the three subcategories of social presence did not prove a significant difference except in a few certain conditions. Attentional allocation showed a significant difference in task dependency between shared co-location and distributed co-location. This provides vague argumentation for prioritizing a distributed task focus when developing multiplayer VR environments. Attentional allocation also showed a significant difference between distributed co-location and distributed non co-location. Furthermore, perceived behavioral interdependence revealed a significant difference between shared co-location and shared non co-location. The remaining measures did not indicate any significant differences among co-location and non-colocation. The findings of this thesis indicate a slight improvement in the case of non-colocation.

Simulator sickness, a mediating variable, was also investigated pre-intervention in an attempt to assess any confounding degree of simulator sickness that could potentially interfere with the validity of the results, however, none of the participants indicated severe issues with simulator sickness that it necessitated intervention from the researchers at any point during the testing sessions. This thesis can ultimately conclude that minor significant differences in social presence were found and that no significant differences was found for collision avoidance between the four conditions.

Drawing a definitive conclusion regarding the research question proves challenging due to the limited number of measures exhibiting statistically significant differences across the conditions. The few instances are insufficient to establish a conclusive disparity in social presence between co-location and non co-location. An argument can be made in support of non co-location, considering the few measures that favored this condition. The impact of task dependency remains inconclusive, as only one out of six scenarios yielded a significant difference.

References

[1] Augmented reality (ar) and virtual reality (vr) headset shipments worldwide from 2019 to 2023.

[2] Jesper Vang Christensen, Mads Mathiesen, Joakim Have Poulsen, Ea Ehrnberg Ustrup, and Martin Kraus. Player experience in a vr and non-vr multiplayer game. In *Proceedings of the virtual reality international conference-Laval virtual*, pages 1–4, 2018.

[3] Bjϕrn Winther, Mikkel L Krarup, Patrick N Andersen, Ungyeol Lee, and Niels C Nilsson. Effects of walking together in a co-located virtual reality game. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 257–262. IEEE, 2023.

[4] Felix Born, Philipp Sykownik, and Maic Masuch. Co-located vs. remote gameplay: The role of physical co-presence in multiplayer room-scale vr. In *2019 IEEE conference on games (CoG)*, pages 1–8. IEEE, 2019.

[5] Iana Podkosova and Hannes Kaufmann. Mutual collision avoidance during walking in real and collaborative virtual environments. pages 1–9, 2018.

[6] Chad Harms and Frank Biocca. Internal consistency and reliability of the networked minds measure of social presence. 2004.

[7] Evan A Suma, Gerd Bruder, Frank Steinicke, David M Krum, and Mark Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. In *2012 IEEE Virtual Reality Workshops (VRW)*, pages 43–46. IEEE, 2012.

[8] Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P Brooks Jr. Walking> walking-in-place> flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 359–364, 1999.

[9] Evan Suma, Samantha Finkelstein, Myra Reid, Sabarish Babu, Amy Ulinski, and Larry F Hodges. Evaluation of the cognitive effects of travel technique in complex real and virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):690–702, 2009.

[10] Niels Christian Nilsson, Stefania Serafin, Frank Steinicke, and Rolf Nordahl. Natural walking in virtual reality: A review. *Computers in Entertainment (CIE)*, 16(2):1–22, 2018.

[11] Doug Bowman, Ernst Kruijff, Joseph J LaViola Jr, and Ivan P Poupyrev. *3D User interfaces: theory and practice, CourseSmart eTextbook*. Addison-Wesley, 2004.

[12] Rolf Nordahl, Stefania Serafin, Luca Turchet, and Niels Christian Nilsson. A multimodal architecture for simulating natural interactive walking in virtual environments. *PsychNology Journal*, 9(3):245–268, 2011.

[13] David Waller and Eric Hodgson. Sensory contributions to spatial knowledge of real and virtual environments. *Human walking in virtual environments: Perception, technology, and applications*, pages 3–26, 2013.

[14] Roy A Ruddle, Ekaterina Volkova, and Heinrich H Bülthoff. Walking improves your cognitive map in environments that are large-scale and large in extent. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(2):1–20, 2011.

[15] Williams E. Christie B. Short, J. The social psychology of telecommunications. pages 64–195, 1976.

[16] Charlotte N Gunawardena. Social presence theory and implications for interaction and collaborative learning in computer conferences. *International journal of educational telecommunications*, 1(2):147–166, 1995.

[17] Catherine S Oh, Jeremy N Bailenson, and Gregory F Welch. A systematic review of social presence: Definition, antecedents, and implications. *Frontiers in Robotics and AI*, 5:114, 2018.

[18] Erving Goffman. *Behavior in public places*. Simon and Schuster, 2008.

[19] Martin Gerin-Lajoie, Carol Richards, and Bradford McFadyen. *The Negotiation of Stationary and Moving Obstructions during Walking: Anticipatory Locomotor Adaptations and Preservation of Personal Space*. 2005.

[20] A. Crétual Olivier, A. Marin and J. Pettré. *Minimal predicted distance:A common metric for collision avoidance during pairwise interactions between walk- ers. Gait Posture*. 2012.

[21] Philip W Fink, Patrick S Foo, and William H Warren. Obstacle avoidance during walking in real and virtual environments. *ACM Transactions on Applied Perception (TAP)*, 4(1):2–es, 2007.

[22] Tracy Fullerton. *Game Design Workshop*. Taylor Francis Group, 2018.

[23] Shona McCombes. An introduction to sampling methods.

[24] Dane Bertram. Likert scales. *Retrieved November*, 2(10):1–10, 2007.

[25] Thomas Bjørner. *Qualitative Methods for Consumer Research*. Hans Reitzels Forlag, 2015.

[26] Carrie Scheel, Jim Mecham, Vic Zuccarello, and Ryan Mattes. An evaluation of the inter-rater and intra-rater reliability of occupro's functional capacity evaluation. *Work*, 60(3):465–473, 2018.

[27] Guru99.com. *Agile Methodology: What is Agile Software Development Model?* 2021. `https://www.guru99.com/agile-scrum-extreme-testing.html#3`, (visited on 2023/06/05).

[28] Unity. Xr interaction toolkit.

[29] Normal. Normcore homepage.

9 Appendix

9.1 Social Presence and Collision Avoidance

The following box plots display the results of the collected data in Co-presence, attentional allocation, perceived behavioral interdependence and collision avoidance.

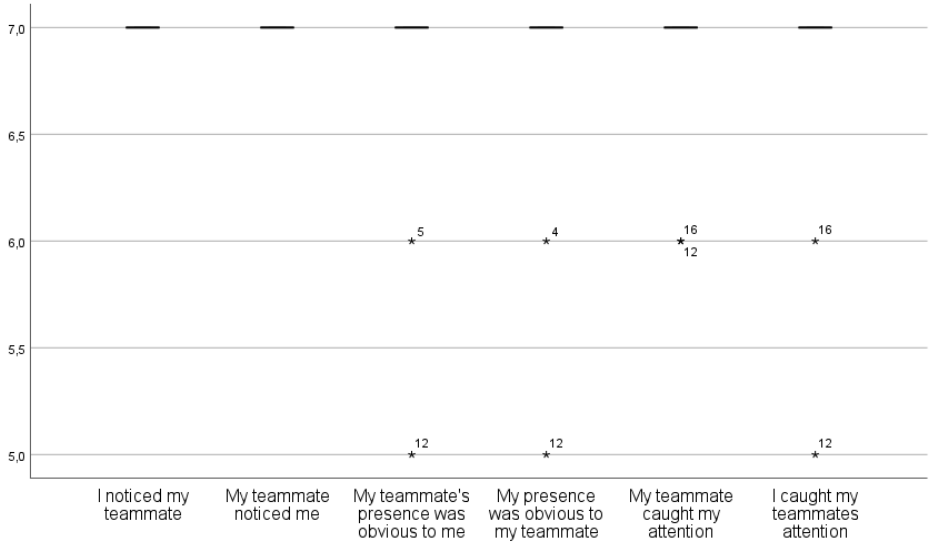


Figure 29: *Co-presence distribution in the co-located shared condition*

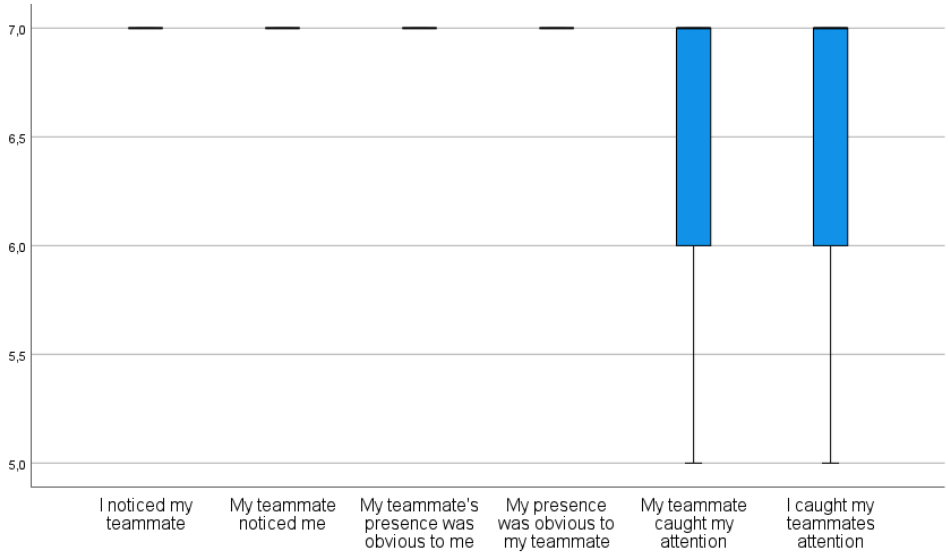


Figure 30: *Co-presence distribution in the co-located distributed condition*

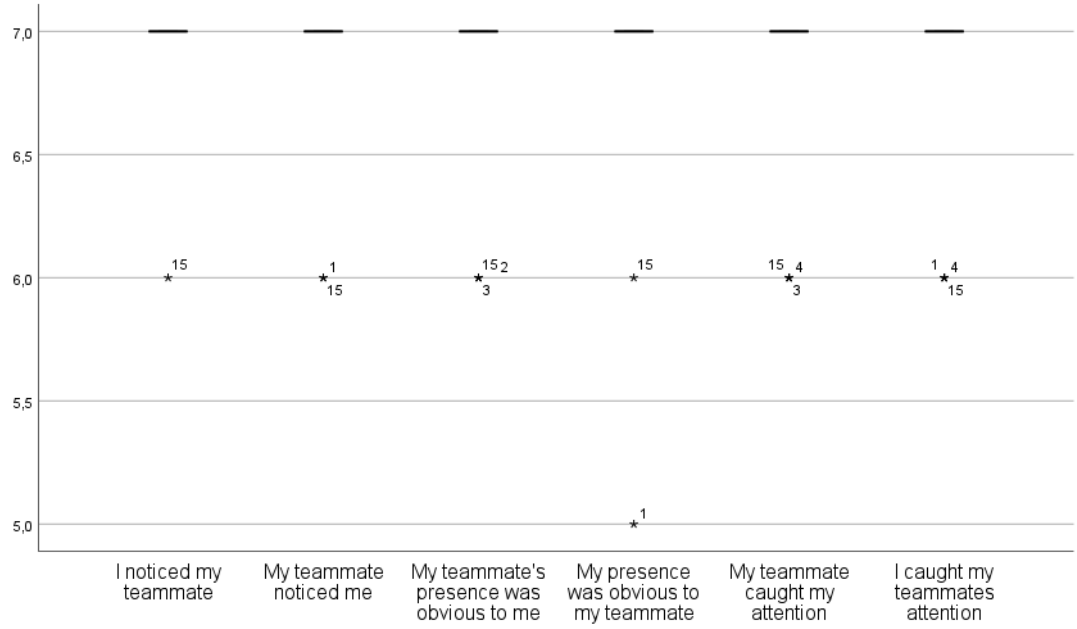


Figure 31: *Co-presence distribution in the non co-located shared condition*

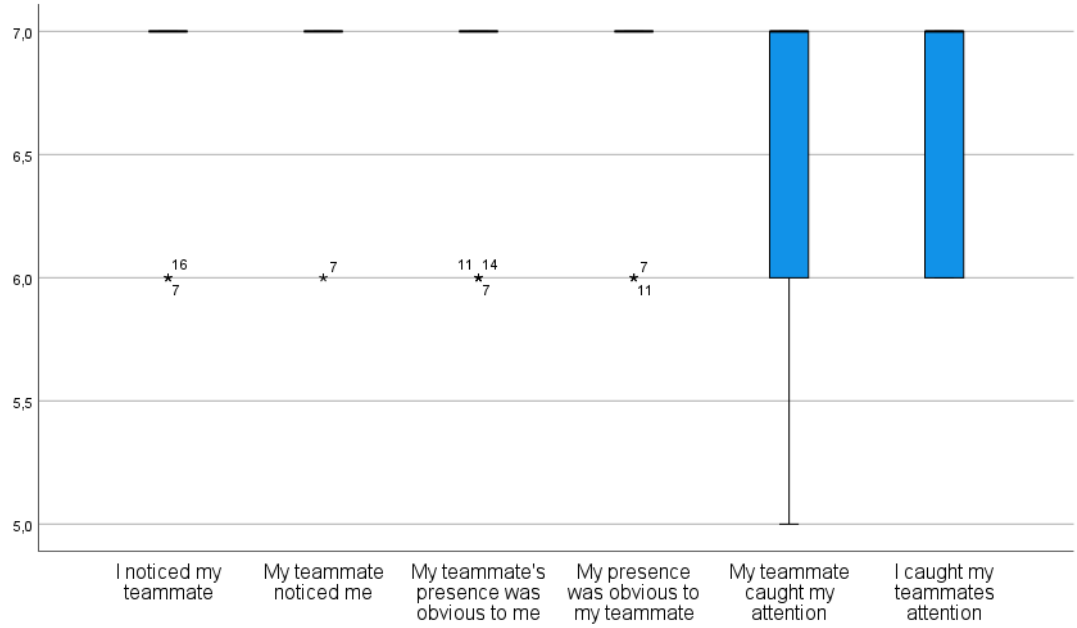


Figure 32: *Co-presence distribution in the non co-located distributed condition*

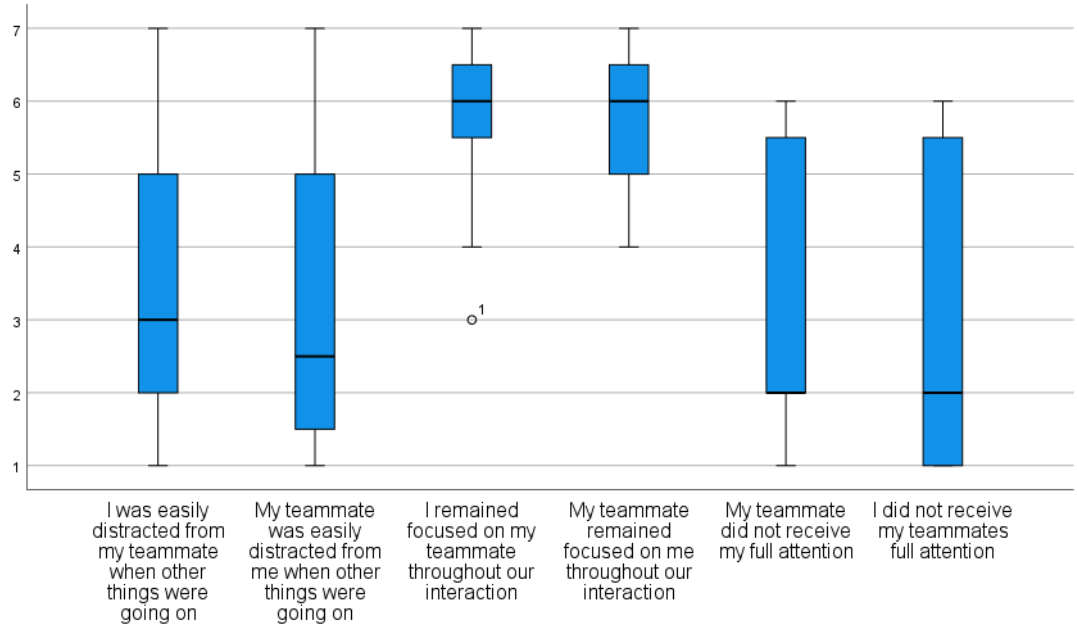


Figure 33: *Attentional allocation distribution in the co-located shared condition*

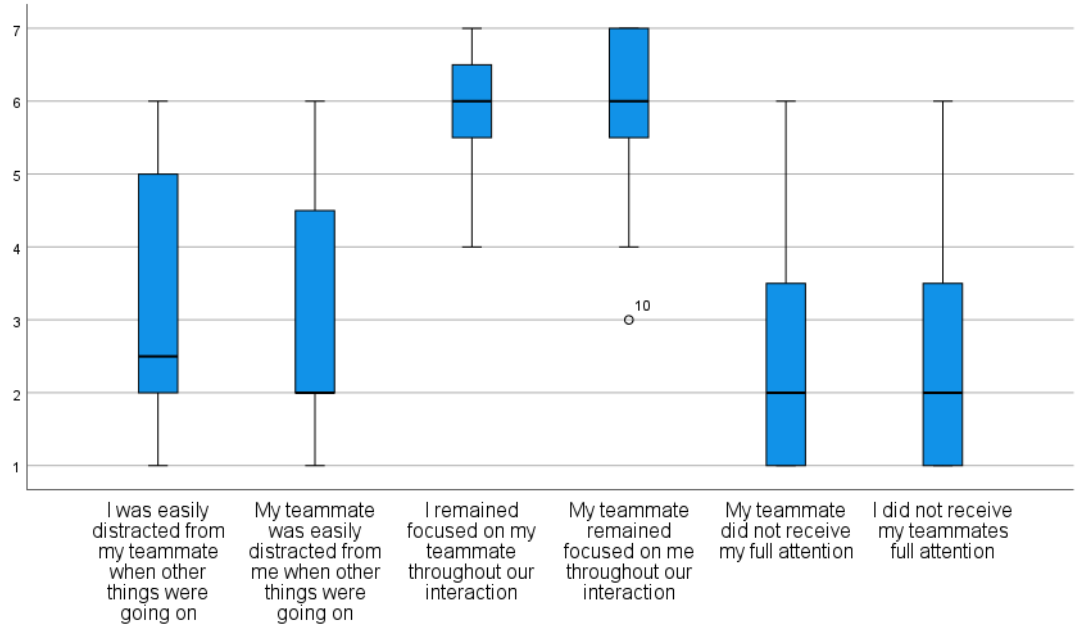


Figure 34: *Attentional allocation distribution in the co-located distributed condition*

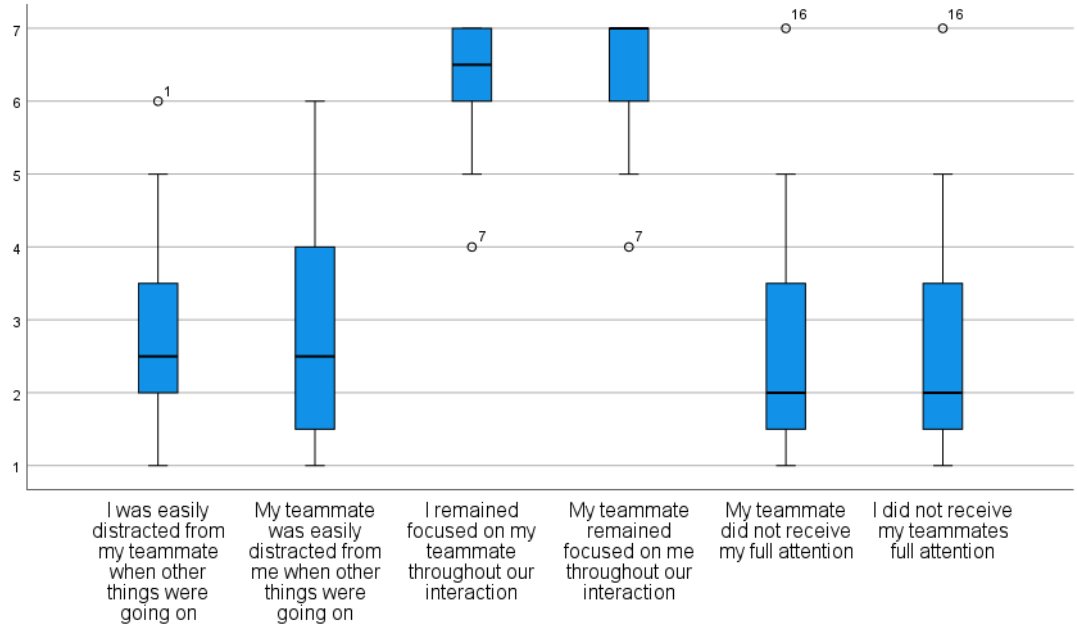


Figure 35: *Attentional allocation distribution in the non co-located shared condition*

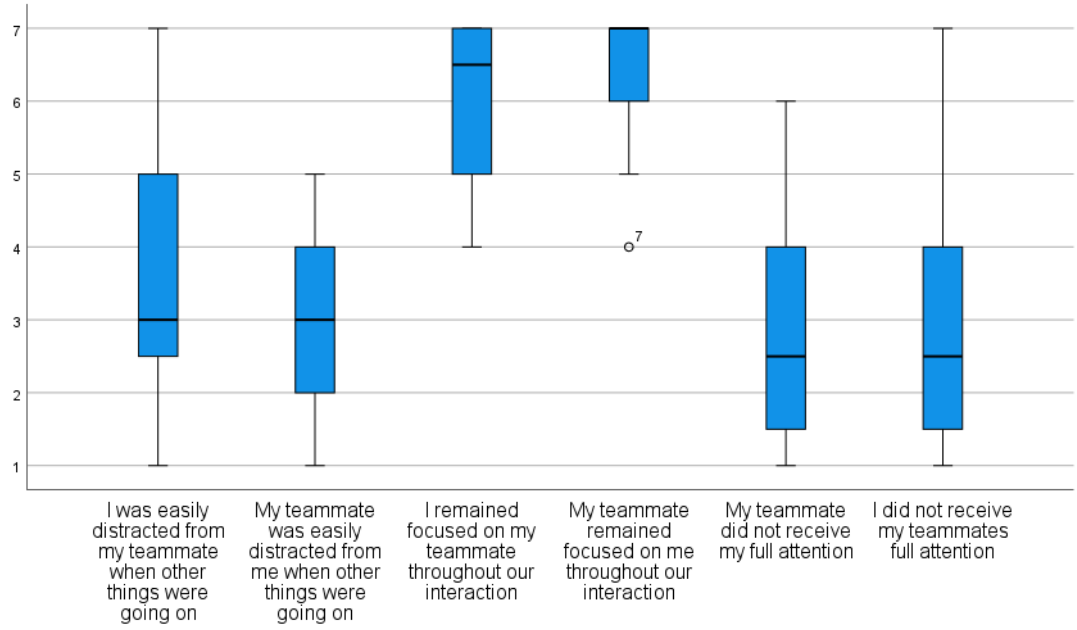


Figure 36: *Attentional allocation distribution in the non co-located distributed condition*

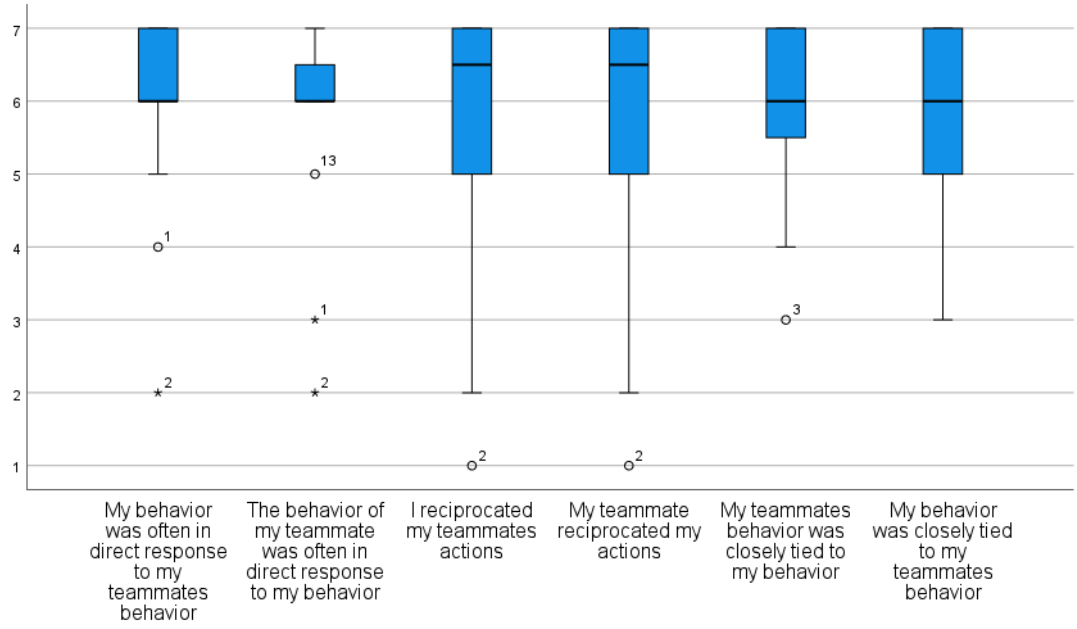


Figure 37: *Perceived behavioral interdependence distribution in the co-located shared condition*

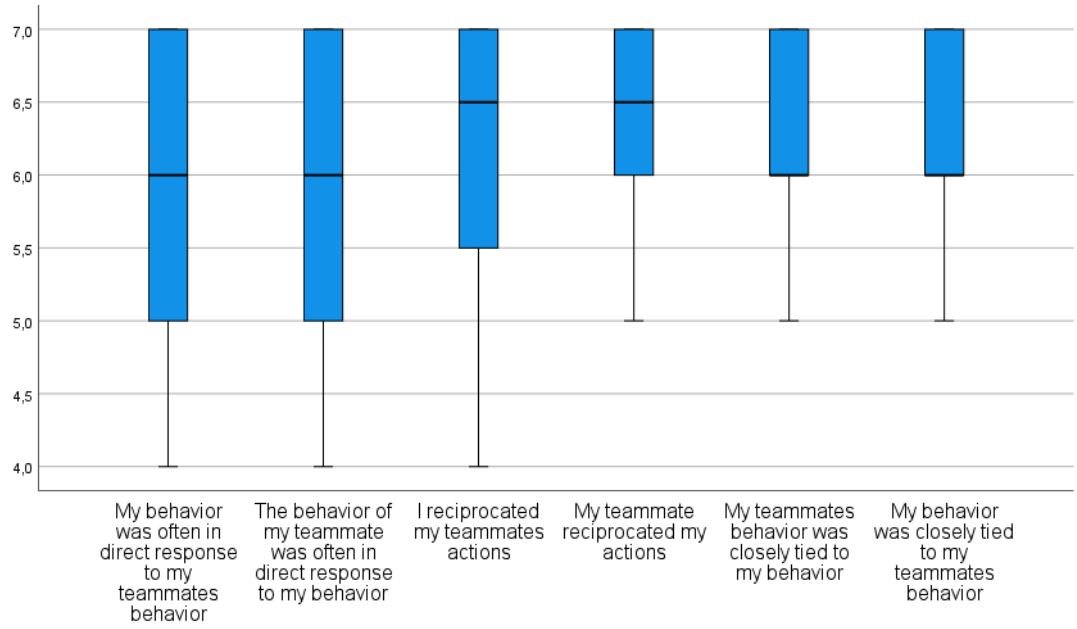


Figure 38: *Perceived behavioral interdependence distribution in the co-located distributed condition*

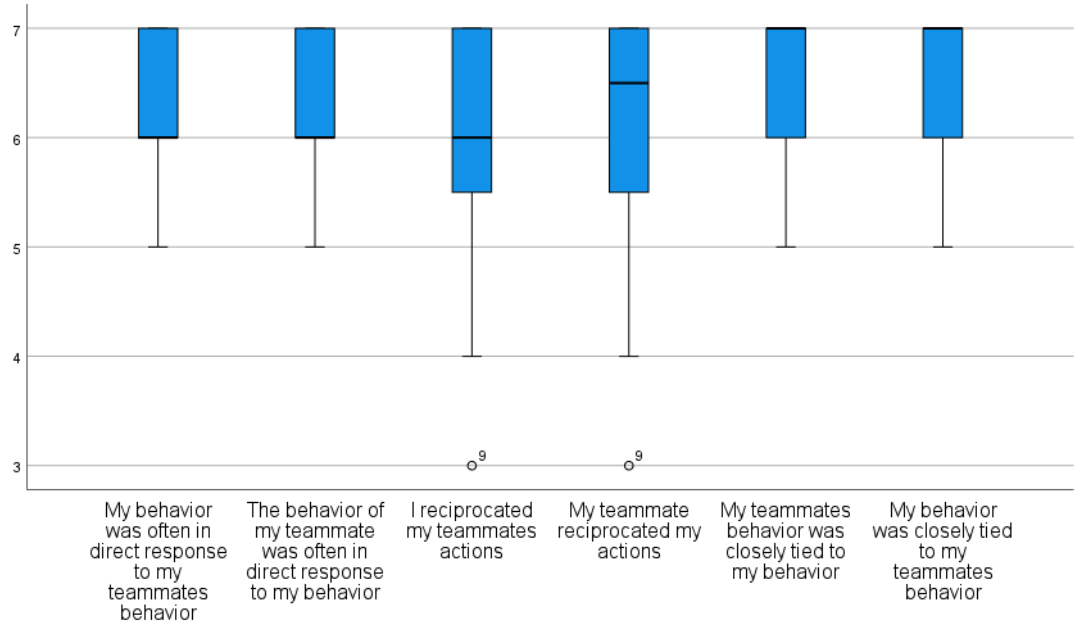


Figure 39: *Perceived behavioral interdependence distribution in the non co-located shared condition*

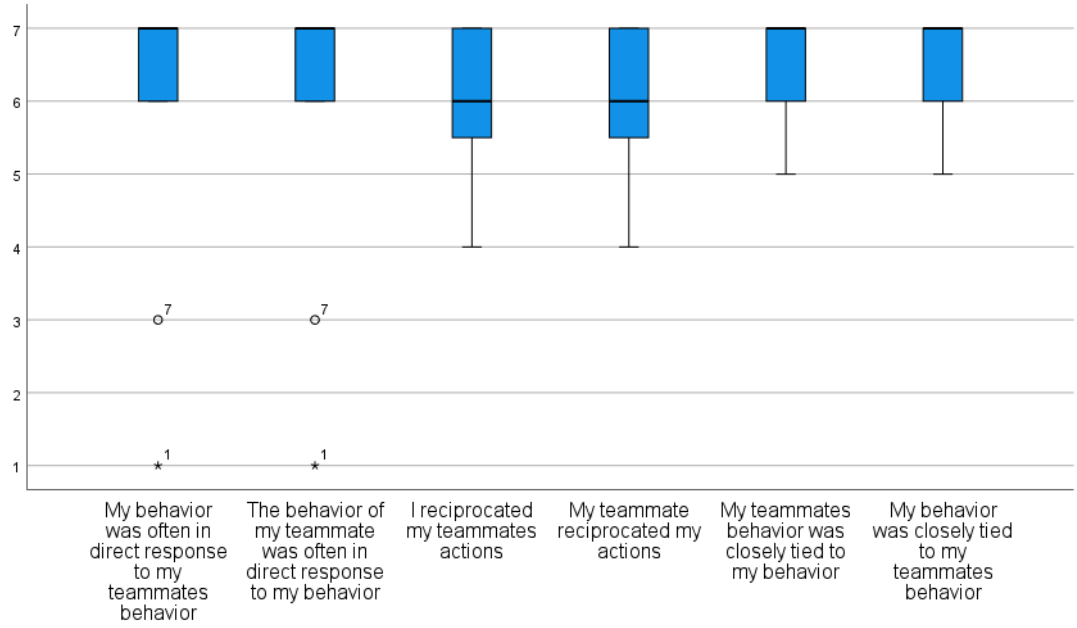


Figure 40: *Perceived behavioral interdependence in the non co-located distributed condition*

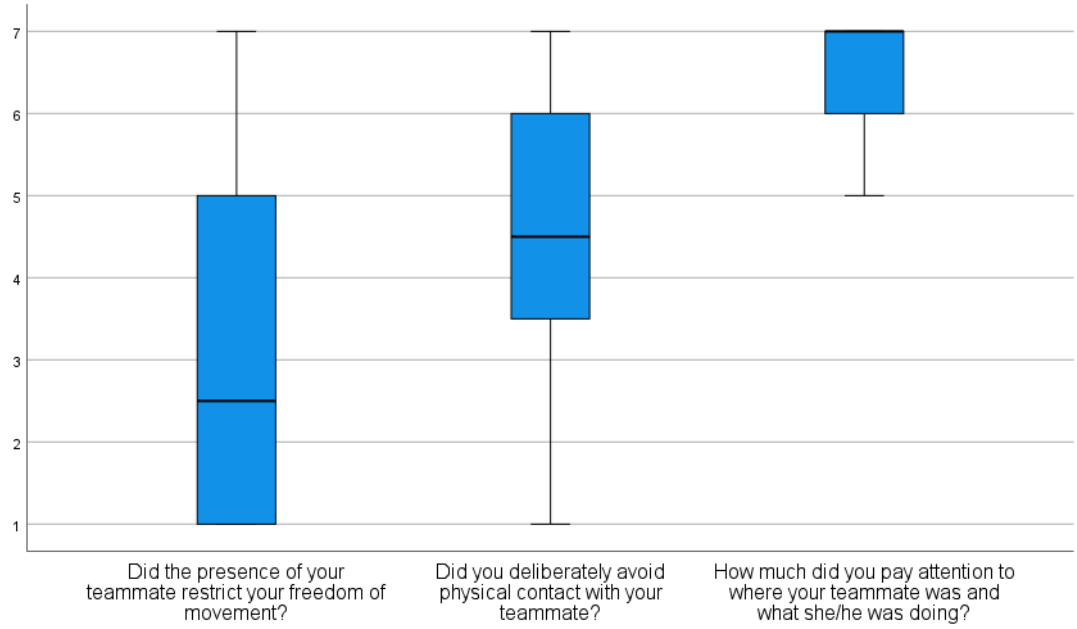


Figure 41: *Collision avoidance distribution in the co-located shared condition*

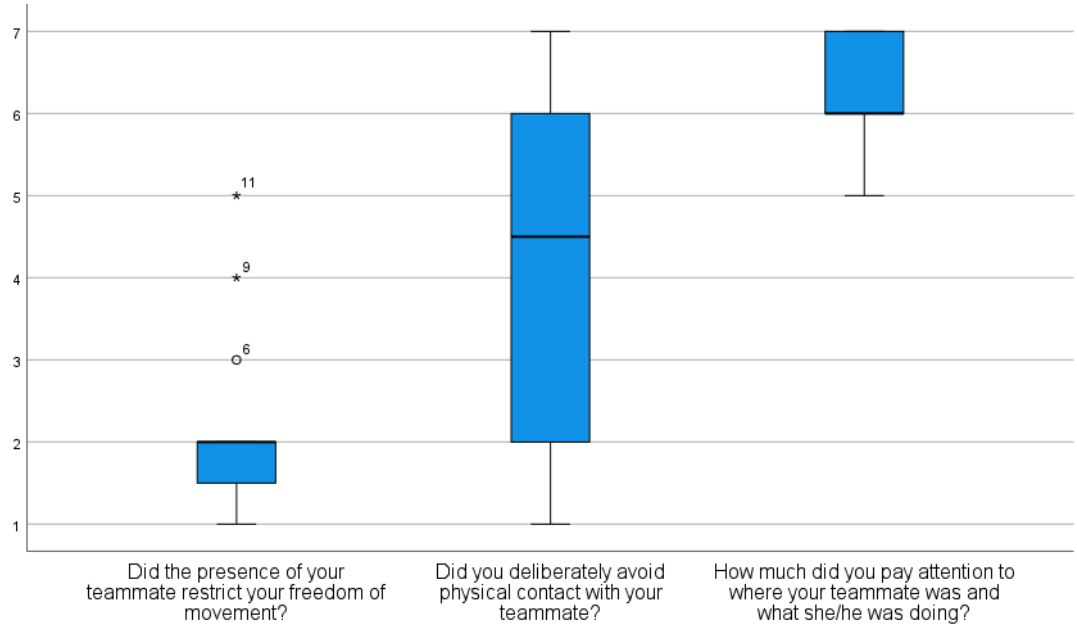


Figure 42: *Collision avoidance distribution in the co-located distributed condition*

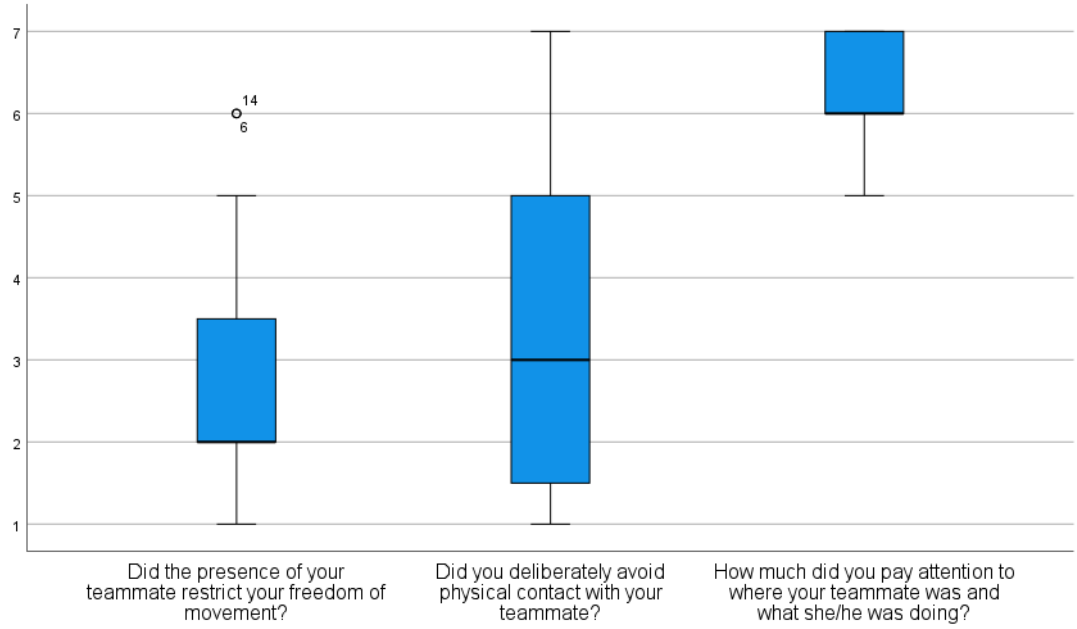


Figure 43: *Collision avoidance distribution in the non co-located shared condition*

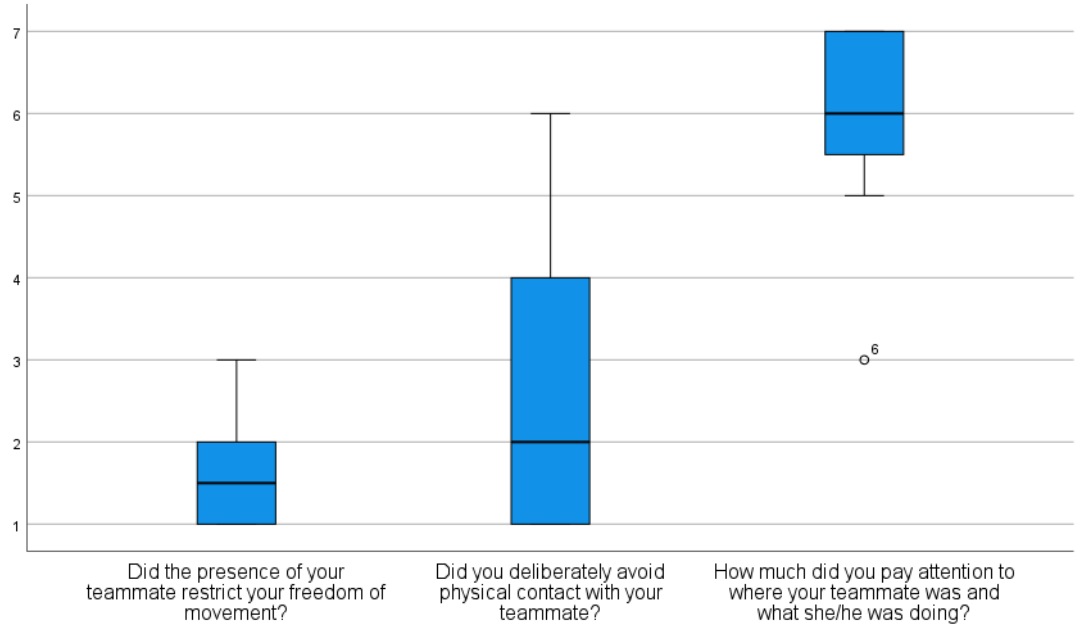


Figure 44: *Collision avoidance distribution in the non co-located distributed condition*