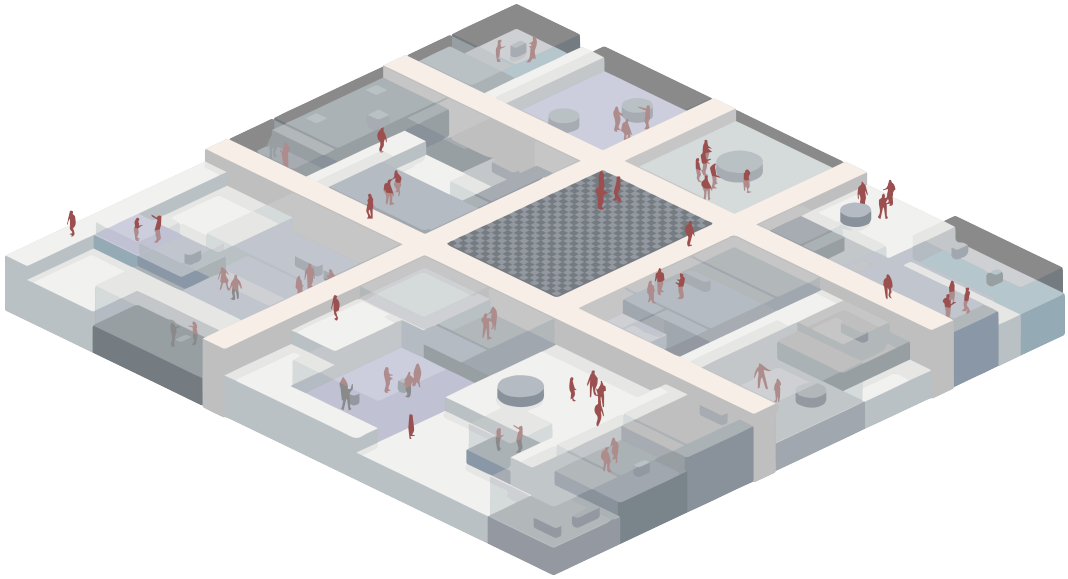
HYPERSPATIAL:

CONSTRUCTING VIRTUAL SPACES

Matthew Lee Jin Young

National University of Singapore



21 November 2020

A digital version of this report is available at

jy-matt.github.io/constructing-digital-space/hyperspatial1.0.report.pdf

Project website can be viewed at

jy-matt.github.io/constructing-digital-space/hyperspatial1.0.html

Contents

List of Figures

1 Introduction

Despite the fact that virtual environments (VEs) have been around for more than 20 years and there has been continued research into their design and implementation, they have been regarded largely as entertainment and as accessories to physical interaction. More recently, virtual reality through the use of head-mounted displays (henceforth referred to as VR) has introduced yet another way in which such virtual environments might be experienced. While VR remains somewhat of a niche market, improvements in technology and accessibility in recent years has made it increasingly accessible to the average consumer[1].

In the midst of the post-digital revolution and the ongoing pandemic, we are increasingly looking at VEs as substitutes for real-life interaction, whether it be for social reasons, education, work, or leisure. Thus far, the design of most VR environments has tended towards imitative environments that mimic physical spaces – in fact, there is often an emphasis on physical cues and architecture to make users feel at ease within the virtual environment, reminiscent of skeuomorphic design during the earlier years of the digital era (Norman, 2013).

However, within virtual environments, traditional architectural conceptions of scale, spatiality, movement, framing and interaction take on vastly

[1]The VR industry as a whole is growing at a fast pace, with the market size of consumer virtual reality hardware and software projected to increase from 6.2 billion U.S. dollars in 2019 to more than 16 billion U.S. dollars by 2022. (Alsop, 2020)

different forms. While the architectural conventions of the real world might be comforting, it is clear that the spatial design of spaces in virtual reality warrants a critical re-examination of its foundations.

1.1 Hypothesis

In this thesis, I hypothesize that by moving beyond conventional, real-world preconceptions of physics and spatial configuration, the design of virtual environments can be made more efficient and comfortable for users, especially for the purposes of peer-to-peer interaction.

2 Research Approach

While the project remains in progress and many of these efforts have been taken in parallel, the core of the research methodology can be traced in a roughly linear fashion. This following order will also serve as the structure for this report.

Firstly, I began with a review of the existing literature surrounding virtual environments. At the beginning of the project, I had not settled on the use of VR as a medium, having equally considered developing a 'web app' that would necessarily be presented in screen-space. Hence, this research investigated the design of both VR and non-VR virtual environments, providing the theoretical and contextual underpinnings for my project as well as ultimately influencing my decision to work with VR. After reviewing the existing literature, I undertook precedent studies of several existing

virtual spaces to understand and critique the existing state of spatial design in VR.

Subsequently, I distilled several parameters and strategies governing the design of virtual spaces, developing and designing various test cases in VR. Learning the requisite programming skills and developing the virtual environment in Unity also took place in parallel to the design and research work. Several users were then recruited to test and give feedback on the new spatial designs, collecting both qualitative and quantitative data that would inform subsequent design choices.

Finally, these inputs and iterations were synthesized to create a set of design parameters and possibilities that would serve as the foundation for the next stage of the thesis project – designing the overarching generation of these responsive spaces and the user experience within them.

3 Review: Virtual Environments

From Vitruvius to Le Corbusier and Neufert, architects have sought to design spaces with the scale and capabilities of the human body in mind, as well as attempting to express the tectonics of material and function in construction. In the same way, the design of virtual spaces ought to reflect the different rules that govern both user and environment in the virtual world (Hansen, 2012). Since the late 90s, researchers have called for "a new theory and practice" due to the nature of space in VEs being "fundamentally

different from the nature of real space" (Bourdakis & Charitos, 1999). While the design of virtual spaces remains largely on the fringe of architectural practice, the ability to experience these spaces in full 3D warrants a re-examination of these fundamental design principles.

3.1 Possibilities of Virtual Environments

For the most part, VEs are differentiated from physical spaces due to their freedom from physical and structural constraints. While this affords a significant degree of freedom to create almost anything the designer might imagine, the following are the most significant implications of VEs from an architectural perspective.

1. Unless consciously implemented, there is an inherent lack of physical rules such as friction, gravity or sound propagation (Bridges & Charitos, 1997). This affects not just the user experience but also the design of spaces, which are not bound by the usual considerations of load-bearing or material properties.

2. Spaces might be multi-dimensional or non-contiguous (Bridges & Charitos, 1997), being connected by portals or other means of translation, or even completely non-Euclidean in nature (Coulon, Matsumoto, Segerman, & Steve, 2002).

3. Scale of the user, objects and environment is unconstrained, allowing for unconventional experiences even of conventional spaces at different scales.

4. Modes of interaction with the environment are explicitly determined by the designer of the virtual space – this allows for new modes of experiencing space, either by restricting interaction to certain actions such as climbing or modes of movement, or allowing the user to perform actions that go beyond the experience of conventional architecture, such as creation, manipulation, destruction, and so on.

3.2 Limitations of Virtual Environments

Despite the seemingly limitless nature of virtual environments, the design of such spaces comes with its own set of constraints that govern design choices. These constraints can be seen as analogous to the physical and resource considerations that limit architectural design in the real world.

1. Resource intensiveness, both from a development and user performance perspective, remains a key concern. Although the gaming and simulation industry seem to have moved towards ever more realistic renderings over the years, such detailed environments require both extensive development time as well as advanced hardware to run. VR is especially taxing, requiring two separate renders to achieve stereoscopic vision – as such, most current VR environments have been designed to minimise graphical complexity (Sundstrom, 2015).

2. No matter how abstract or realistic the environment is designed to be, it ought to be designed in a way that allows the user to easily understand the space for (1) navigation and (2) interaction. While

navigational concerns are similar to those of real-world architecture, interaction presents a novel challenge as the visual representation of virtual objects might be entirely divorced from their function. Designers need to provide users with the affordances necessary to understand the environment, both at a macro and micro scale (Ellis, 2019).

3. Finally, comfort is a concern that applies to most immersive spaces but is especially a concern in VR. Especially for users new to the technology, movement in VR can cause discomfort and nausea due to the brain's attempt at processing the artificial spatial visualisation (Thompson, 2020). Spaces and interfaces should be designed so that the user feels at ease within the space, both in terms of physical discomfort as well as subjective unfamiliarity with the space.

4 Precedent Studies in VR

In order to gain an understanding of the current state of spatial design in VR, I undertook a series of studies of virtual spaces, focusing on archetypal VR spaces that exemplify either (1) player navigation in VR space, or (2) peer-to-peer interaction in VR.

4.1 Navigation and Design of VR Spaces

Several 'home base' environments were studied to gain an understanding of the current state of spatial design in VR, especially with regards to

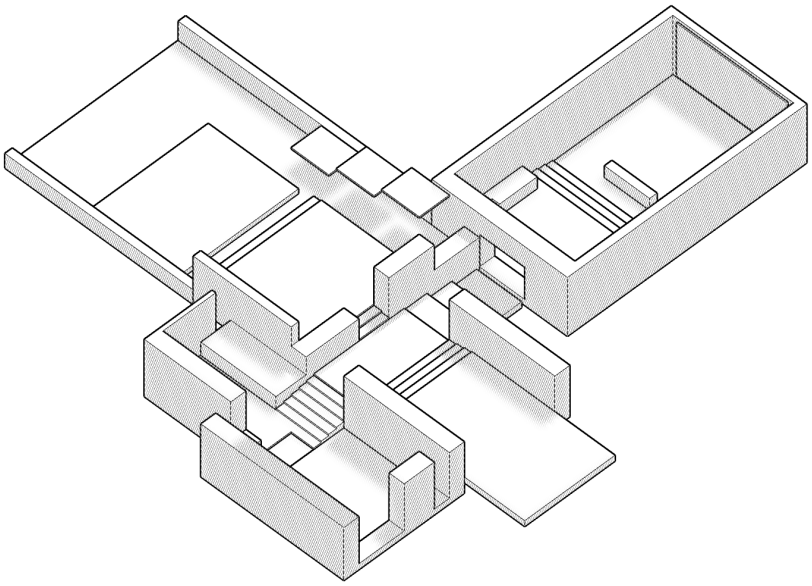


Figure 1: Illustration of the Windows Mixed Reality Cliff House

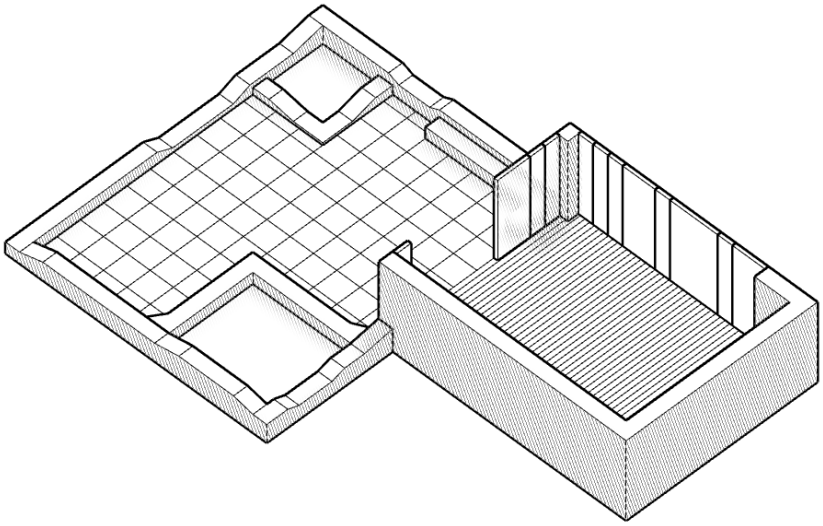


Figure 2: Illustration of the SteamVR Summit Pavilion

Figure 3: Home environment in AltSpaceVR

navigation and representation. These would be the first environments a user encounters when they put on a headset or launch particular apps, making it likely that they would have been carefully and intentionally designed to suit the VR medium. These spaces are the two default houses for the Windows Mixed Reality (WMR) and SteamVR platforms – 'Cliff House' and 'Summit Pavilion' respectively – as well as the two starting 'house' environments for the social applications AltSpaceVR and VRChat.

Within all of these environments, there is a clear trend towards realism – skeuomorphic details help the user acclimatise to the virtual world and show off the immersive capabilities of VR technology. In both the WMR and SteamVR houses, photorealistic textures such as concrete and timber flooring are used to imitate real-world surfaces, while the latter features architectural elements such as window frames and thick structural walls

Figure 4: Home environment in VRChat

that convey a sense of structural realism. In contrast, the home spaces in VRChat and AltSpaceVR are designed in a much more cartoonish manner with skewed lines and exaggerated proportions in some cases – however, despite the cartoonish aesthetic, the design of both environments features furniture and decorations that mimic a real-life environment.

In terms of spatial configuration and movement, teleportation is the predominant means of locomotion in all cases. Aside from the WMR house, the three other houses feature relatively flat plans with no changes in elevation, likely for ease of navigation. In contrast, the WMR house has a fairly complex layout with level changes between each section, highlighting the variation of spaces and ability to move up and down with teleportation. To the left of the starting area, there is even a series of platforms that lead up to the roof – encouraging the user to experiment with the

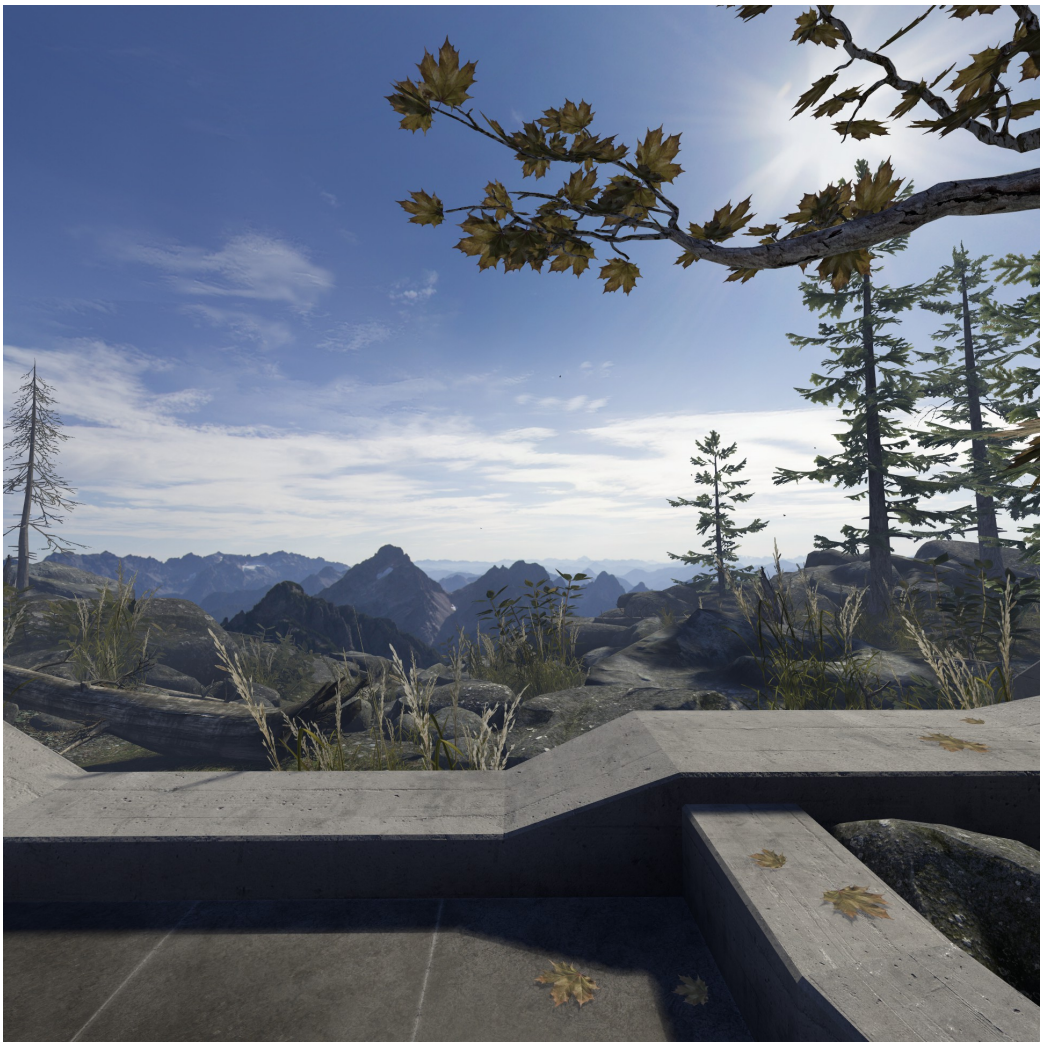


Figure 5: Photorealistic textures in the Summit Pavilion

unconventional means of experiencing the space. Regardless, 'realistic' features such as stairs appear to serve no purpose in the VR space, as there is no need for such mediation to jump between levels.

Finally, the scale of the environments are mostly designed in accordance with real-world proportions, with a few notable exceptions. Boundary walls tend to be lower, as there is no need to consider safety at the expense of views or aesthetic preference. Doorways are either wider than usual (1.5-2m in the VRChat/AltSpaceVR home), or take the form of wide open apertures such as in the WMR and SteamVR houses. This is a key design feature for teleportation, as providing line of sight and minimising turns and jumps between two points becomes a key consideration when planning movement.

4.2 Superhot VR: Embracing Teleportation

In VR, teleportation as a means of locomotion completely changes how we navigate spaces. SUPERHOT VR, one of the most acclaimed games for VR, takes place as a series of action vignettes that the player has to complete. Particularly relevant is the fact that locomotion is taken out of the player's hands, as the player is automatically teleported to the next location upon defeating all the enemies at the previous one. Deliberate positioning of these successive points of view allow the player to understand that the scenes are taking place in the same overall environment by recognising common environmental features or figures. Here, teleportation

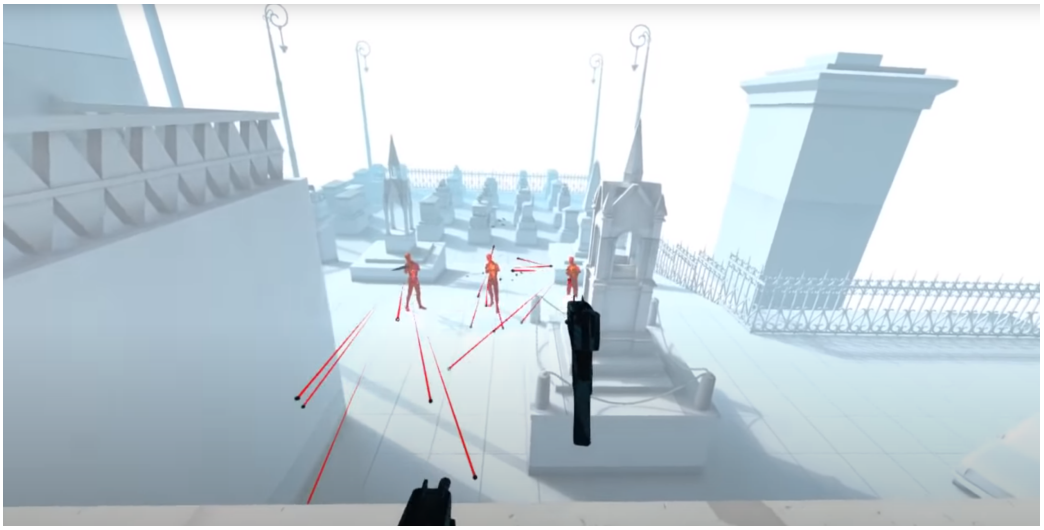


Figure 6: Screenshot of SUPERHOT VR

is embraced as a core feature of the VR experience, with spaces, framed views, and action sequences designed around that fact.

In addition, while the 3D modeled spaces appear to simulate real-life environments, they are presented in monochrome white, with enemy figures colored in bright red and interactable objects in black. Despite the lack of realistic texturing, the player is clearly able to understand the nature of the architectural space via depth perception, while their attention is drawn to the key objects of importance. These two design choices demonstrate the potential of designing VR environments that go beyond copying real-life environmental conditions.

4.3 Peer-to-Peer Interaction in VR

As the focus of this project is on designing spaces for multi-user interaction, I also studied several spaces in social VR platforms, namely, 'The Room of the Rain' and 'The Black Cat' in VRChat, and the 'Campfire' space in AltSpaceVR.

In the first two rooms, the design was once again imitative of real-life spaces, featuring photorealistic textures and elements such as non-interactable furniture and even a toilet. It is possible that users feel more comfortable with analogues of real spaces, even if their use of the space is entirely different. This was especially evident in 'The Black Cat', a virtual bar and restaurant, where users ignored the imitative restaurant seats and bar and instead congregated at the virtual 'mirrors' (allowing the user to observe themselves and others) for social activity.

In all three cases, audio was modulated based on distance from speakers as in real life, meaning that people naturally congregated to hear each other better, and moved away from other groups so as not to be distracted by other conversations. To access other rooms or instances, users have to bring up a 2D interface where they can select another room of their choice, as all the rooms are isolated within their own spaces.

From the above case studies, we see that the current iterations of VR spaces are largely derived from our conceptions of physical space and

Figure 7: Screenshot of Campfire space in AltSpaceVR

navigation. While this might afford a degree of comfort and accessibility to new users, there is a clear opportunity for spatial design to embrace the modes of locomotion and interaction inherent to the VR medium, moving beyond physically-imitative design. In addition, menus and abstracted representations that appear to be remnants of screen-space design have the potential to be conveyed spatially rather than relying on cumbersome menus navigated via controller buttons.

5 HYPERSPATIAL: A New Conception of Virtual Space

In science fiction, 'hyperspace' is used to refer to alternate dimensions that do not obey the regular laws of physics, usually used for

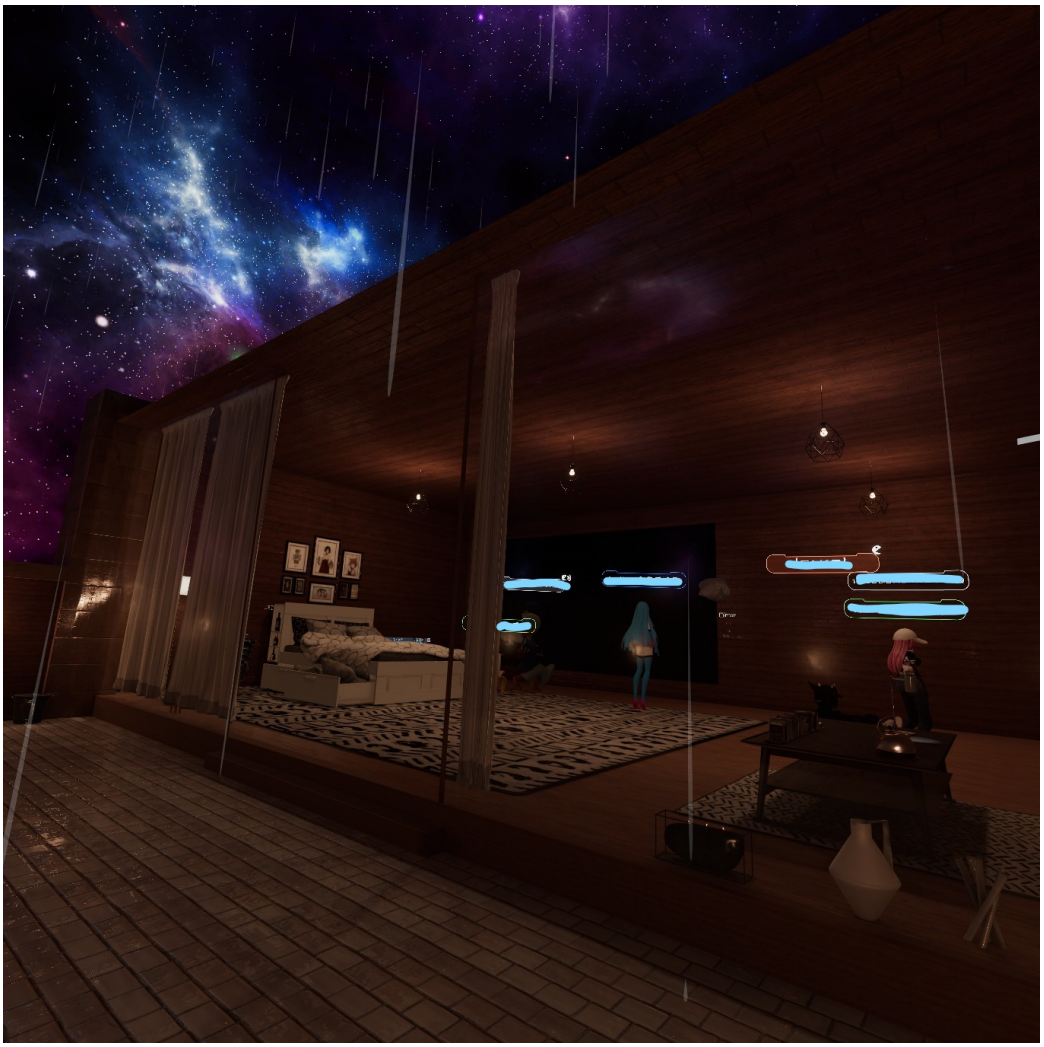


Figure 8: Screenshot of "The Room of the Rain"

Figure 9: Screenshot of "The Black Cat"

faster-than-light travel. Similarly, this thesis project aims to create a virtual platform that embraces the multi-dimensional, dynamic nature of virtual space as a key architectural strategy. This platform will be a multi-user space that facilitates multiple levels of peer-to-peer interaction, from small group discussions to large conference settings. Unlike the current generation of VR social platforms, Hyperspatial will be a dynamic, procedural space that responds to the needs of its users, generated in accordance with a new set of design guidelines for VR.

5.1 Platform: Unity3D

Due to the complexity of building a procedural 3D environment in VR, Hyperspatial is built in Unity3D, allowing for the use of a variety of graphical assets and advanced scripting support with C#. The Unity XR Management Framework also allows for easy exporting for both the Windows Mixed Reality and Oculus VR platforms, although the current industry standards for interoperability are still at an early stage and subject to glitches and gaps in support.

5.2 Broad Investigation - Parameters for VR Design

In order to derive a set of general guidelines for spatial design in VR as well as discover which aspects of virtual environment design were worth focusing upon, I distilled a number of key parameters that could be varied in virtual space. These parameters formed the basis for a broad-based investigation into the merits of so-called 'virtual physics', testing out which variations

might be significant and which ones turned out to be redundant. The goal was to create a basic set of rules that would guide the eventual generation of environments in the interactive platform.

1. Teleportation as primary means of locomotion

   After analysing the various means of locomotion in existing VR applications as well as in existing literature, it was understood that while less immersive than other methods, teleportation was the preferred means of locomotion due to the minimisation of nausea compared to continuous joystick movement (Thompson, 2020). Implementing and testing both modes in a test environment proved this to be the case, as such, it was decided that the spatial design would embrace teleportation as the primary means of experiencing space.

2. Standardised gravity

   Although one of the possibilities early on was to play with gravity and disorientation that could not be achieved in the real world, a simple test environment where the user was able to flip gravity in a room showed the experience to be extremely nauseating in VR. As such, changes in gravity or involuntary movement of the user should be minimised.

3. Human scale

   In order to maintain a sense of immersion and relatability between participants, it was also decided that virtual avatars in this environment should maintain a roughly equal proportion to each other and to real-life human proportions.

4. Virtual spheres of influence

   Despite the adherence to real-life human scale, the mode of interaction and spheres of influence of an avatar in VR differ greatly from real life. Although advanced controllers and suits are available, most commercially available headsets only track the user's head position and their two hands, meaning that perception and interaction are centered around these key nodes of the virtual avatar. The hands are empowered by the ability to grab objects at a short distance, or even interact with objects far away through the use of projected beams, extending the user's reach beyond human proportions.

   In contrast, other parts of the body become irrelevant as the sphere of influence is centered around the upper body. Interaction with objects, sightlines, and other points of interest should thus be laid out in easy reach of the user's sphere of influence.

5. Width of entrances and apertures

   The lack of collision or a necessarily solid body in the virtual space makes spatial allowances a matter of comfort and perception rather than necessity. The key concern regarding the width of entrances, however, arises from the need to navigate via teleportation. In most of the case studies, having a wide doorway or even an entirely open face allows the user to see and teleport with ease, minimising the number of jumps and degree of head turning. By testing various opening sizes in the test environment, it was found that an entrance

width of about 1.5m is the minimum size for comfortable navigation.

6. Interactable surface height

   While there is little need for virtual furniture such as chairs and beds to be ergonomic since the user cannot physically interact with them, the sphere of influence around the upper body means that convenient surfaces for placing and observing objects become an integral part of the environment. This is further complicated by the fact that VR users can either be seated or standing in real life, although the reach provided by the virtual grip allows for some additional flexibility. Testing with objects and pedestals of various heights revealed that the comfortable height of surfaces was about 700-1100mm for standing users, and 600-800mm for seated ones – giving a recommended height of 700-800mm to accommodate both users.

7. Text size and distance

   Text in virtual reality is generally harder to perceive, due to the constant movement and the relatively lower resolution of most currently available headsets, resulting in pixelation known as the 'screen-door effect'. As such, text needs to be clear and legible if it is to be presented in 3D. Taking reference from existing work, several font sizes were tested at varying distances and it was concluded that a perceived font height of 3.5° was suitable for comfortable viewing at any distance.

8. Audio perception

   At present, existing VR chat rooms largely have audio modulated by

proximity, imitating the real world and creating a natural incentive for groups to cluster or spread out. While this might be ideal for regular conversations, there is an opportunity for audio to be transmitted differently in asymmetric situations such as a lecturer speaking to an audience, or a small discussion group where everyone wishes to hear each other clearly.

5.3 Targeted Investigation - Spatial Configurations

In addition to the above parameters, the most promising series of test designs focused on the spatial configuration of various rooms and how they affected the user experience of navigation and perception of comfort. These form the basis of a more targeted investigation as well as shaping the current design proposal for Hyperspatial.

Due to the vastly different way in which people experience space through teleportation, it was clear from the beginning that traditional architectural features such as doors, stairways and corridors do not serve their original purposes in VR – in many cases, they are redundant or even obstructive. Thus, I sought to find new ways by which spaces could be composed and aggregated that were better suited to this new mode of locomotion. Furthermore, on a larger scale, existing social platforms dealt with different 'rooms' by having them accessed through menus and lists, making for a cumbersome experience in VR. Is there a way to represent and navigate these rooms spatially, removing the need for text interfaces as well as

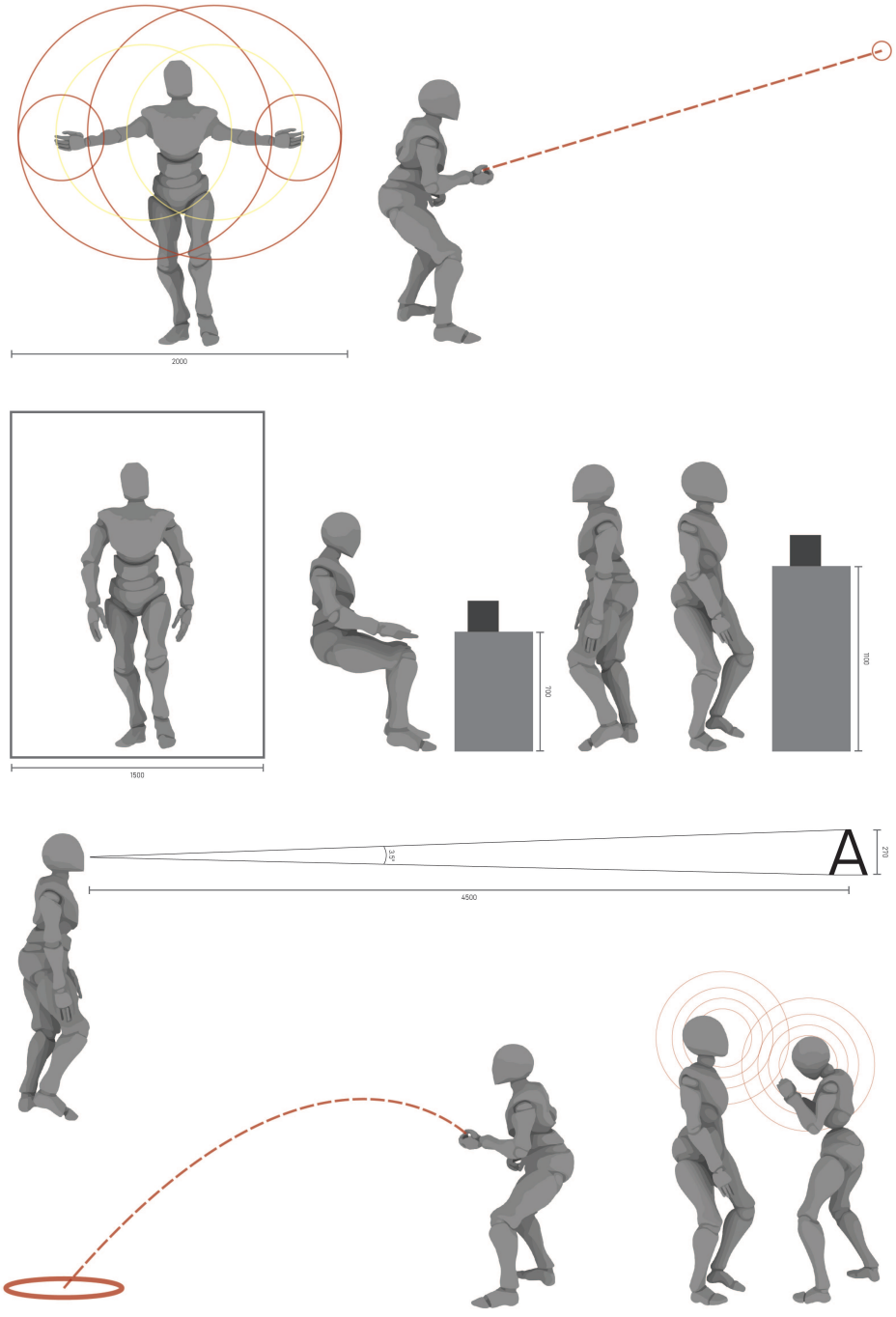


Figure 10: Diagram illustrating derived proportions and relationships in VR

offering the user an intuitive understanding of the overall space?

After a series of conceptual iterations, three test cases were developed that demonstrated different means of navigation:

1. A 'traditional' layout with walled rooms, doorways, and a central corridor

2. An experimental layout, comprising rooms with transparent walls – the user is able to appraise all the rooms at a glance, but when they step into a particular room, the walls turn solid to give a sense of privacy. This layout has the added benefit of rooms allowing entrance/exit from all sides.

3. Rooms without walls but separated instead by level differences

Of these three, the first case highlighted clearly the poor experience of navigating conventional spaces in VR, while the latter two offered interesting opportunities to leverage the potential of virtual space. The 'virtual rooms' in the second case proved to be easily navigable, while the (at present) illusion of privacy felt surprisingly comforting despite knowing other users could still see in. The last case did not make use of any virtual tricks, but demonstrated the ease of navigating between levels in VR and the sense of privacy and differentiation afforded by level changes.

These cases were thus developed further for an experimental trial to gather feedback from participants, to assess the detailed implications and user reception of the various spatial conditions.

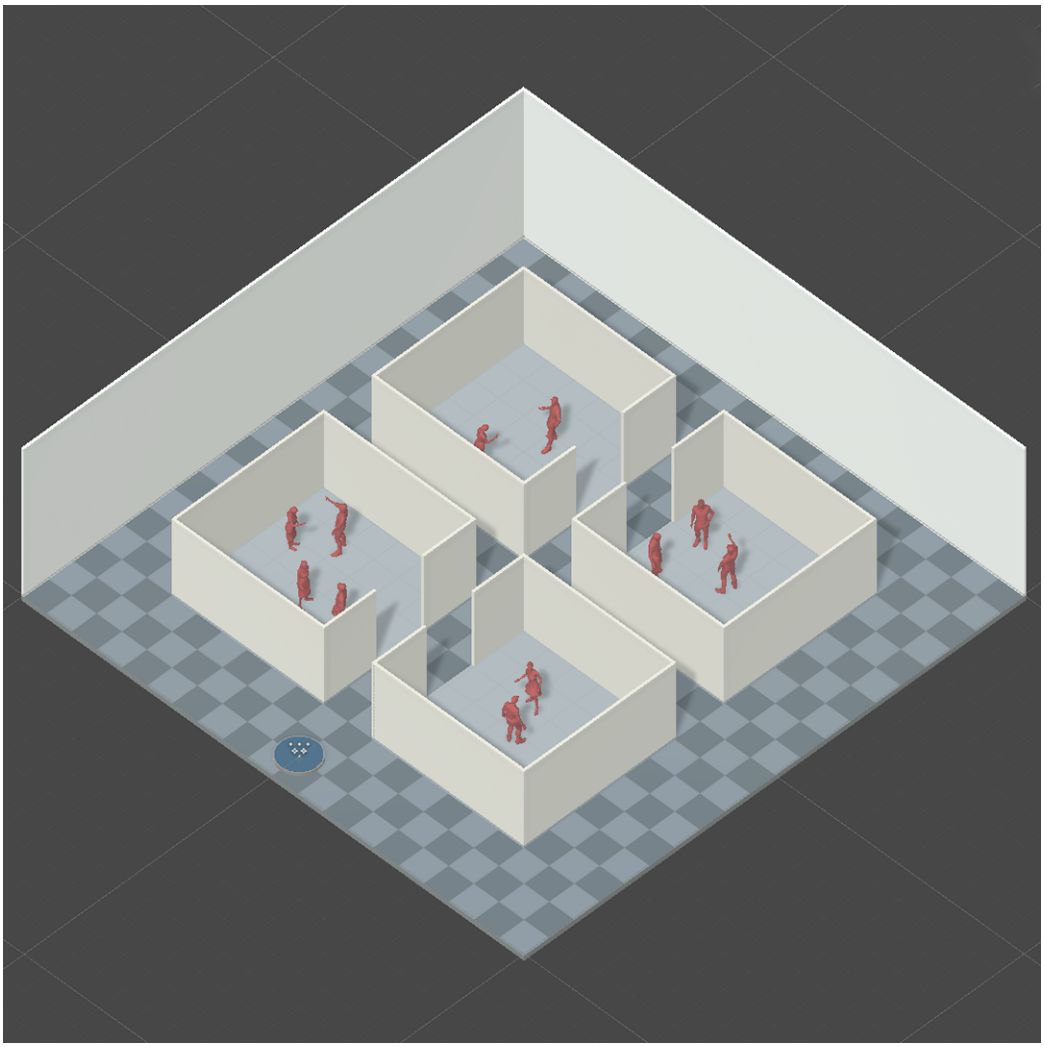


Figure 11: Spatial configuration 1

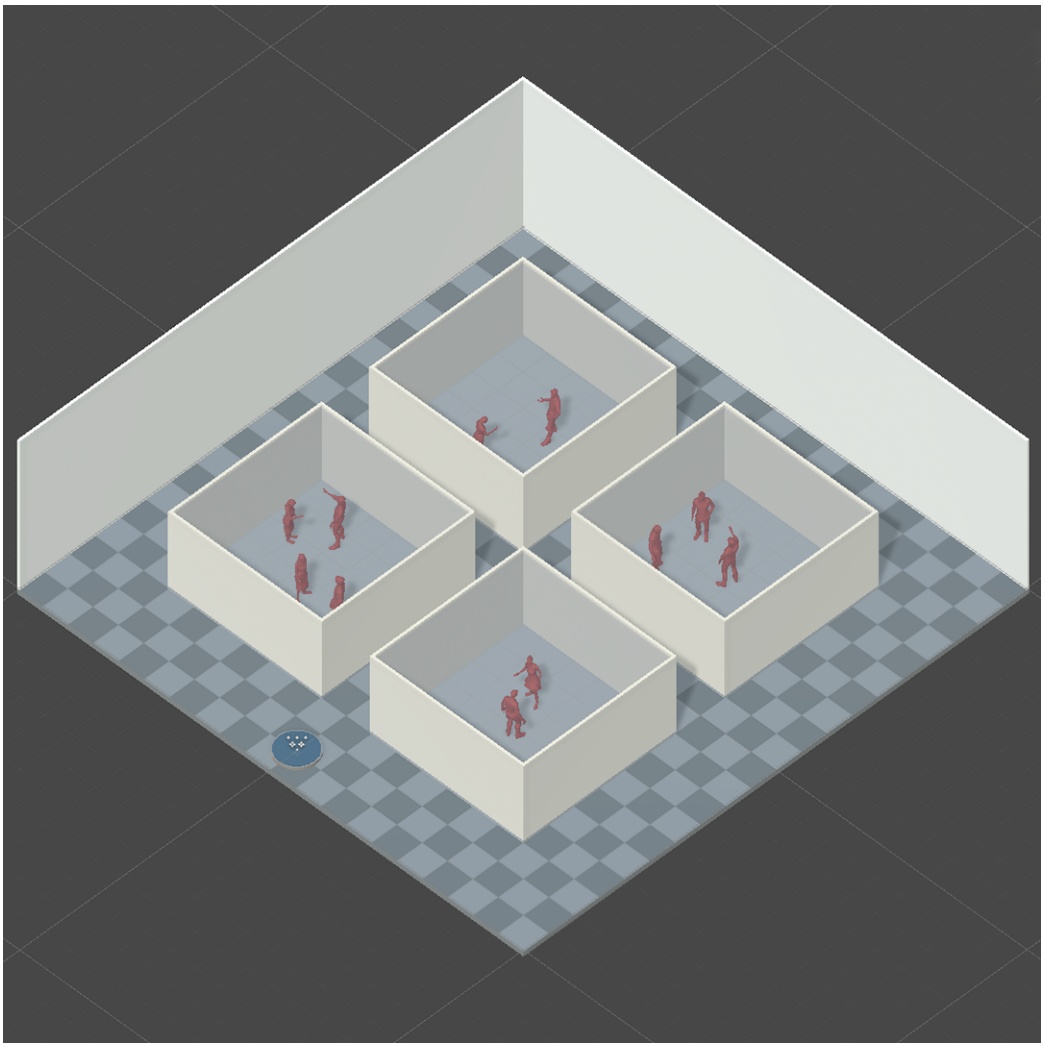


Figure 12: Spatial configuration 2

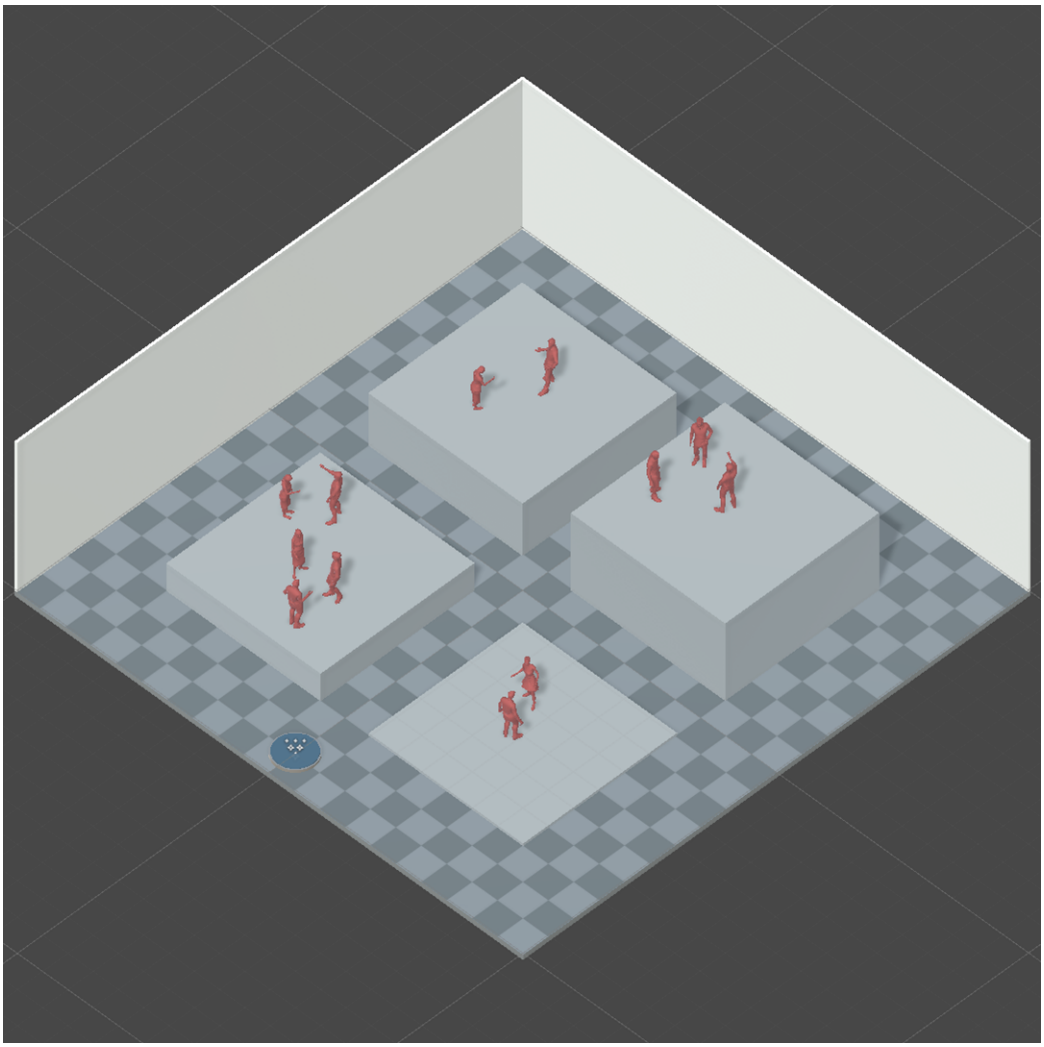


Figure 13: Spatial configuration 3

6 User Testing

6 participants were recruited for a short experimental survey, in order to verify the hypothesis that these non-standard virtual spaces could indeed contribute to user comfort and efficiency of navigation. Although the original intention had been to devise tests with multiple users to assess the dynamics of peer-to-peer interaction, the limited pool of participants with VR headsets and the difficulty of scheduling group meetings between the diverse participants meant that the experiments had to be devised for individual participants using character stand-ins to represent other users instead. The focus was thus on the quality of spaces and ease of navigation, rather than the actual interaction between participants.

In addition to the 3 aforementioned test environments, participants were presented with a 4th room containing a different 'levels' configuration with a central platform overlooking the others. In each scenario, participants were tasked to visit 6 groups of characters scattered around the room, reporting back two pieces of information – the letter missing from the sequence 'ABCDEF', as well as the letter belonging to a highlighted character. These simple tasks were devised as a proxy for actual interaction with other users in a real scenario. After each test, participants reported 4 scores along with their answers:

1. Time taken

2. Ease of navigation

3. Degree of physical discomfort

4. Perceived comfort of rooms as a prospective user

The test also included 2 additional mini-scenarios, asking participants to resize a room to their liking so as to asses what people perceive to be a 'comfortable' room scale for two people.

The full results of the survey as well as the full questionnaire can be found in the appendices, but the most relevant results of the study are as follows:

1. The 'virtual wall' scenario scored highest for both navigation and room comfort, followed by scenario 4 with the overlooking platform in both categories

2. Scenario 4 scored the lowest for physical discomfort

3. Scenario 1 was described as awkward, narrow, and difficult to navigate, scoring badly for navigation and physical discomfort

4. The 'virtual walls' were perceived as the best for a sense of privacy

5. Open plan layouts minimise navigational challenges and turning around, which can be uncomfortable

6. Having an elevated perspective to view rooms is a major aid for navigation

Unfortunately, the results for the room size experiment were too much of a spread for a conclusive interpretation. Ideally, a subsequent experiment

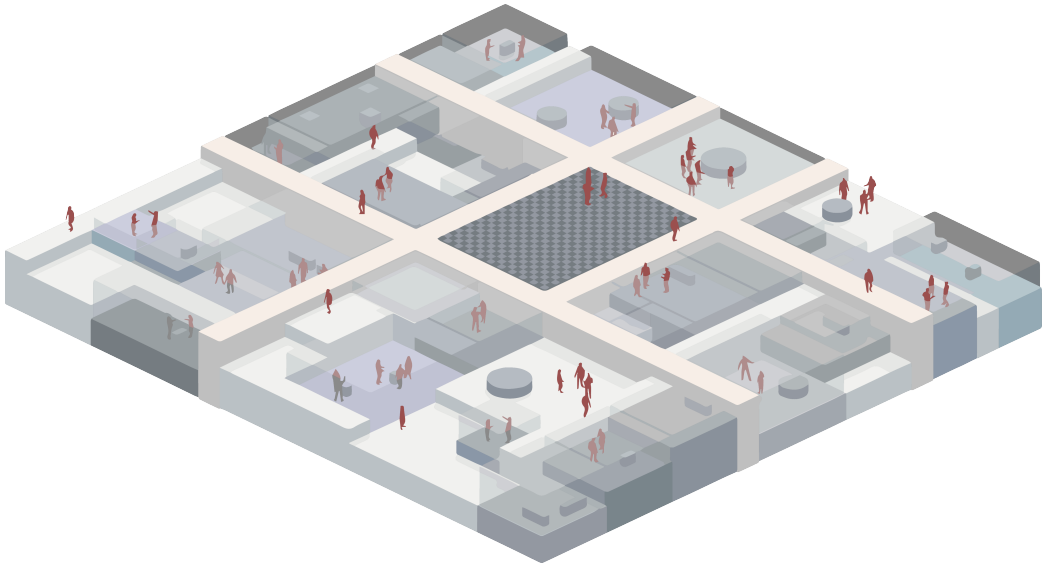


Figure 14: Prototype of "Hyperspatial"

could be conducted with varying group sizes as well, in order to measure ideal dimensions for various activities.

Overall, the results showed a fairly strong preference for the dynamics of the 'virtual wall' rooms, as well as the navigational advantages and variety offered by changes in levels. It was clear that the 'traditional' layout emulating rooms and corridors was not at all suited to the VR experience, while the new design suggestions offered promising opportunities – supporting the initial hypothesis that going beyond real-life conventions could offer a better user experience.

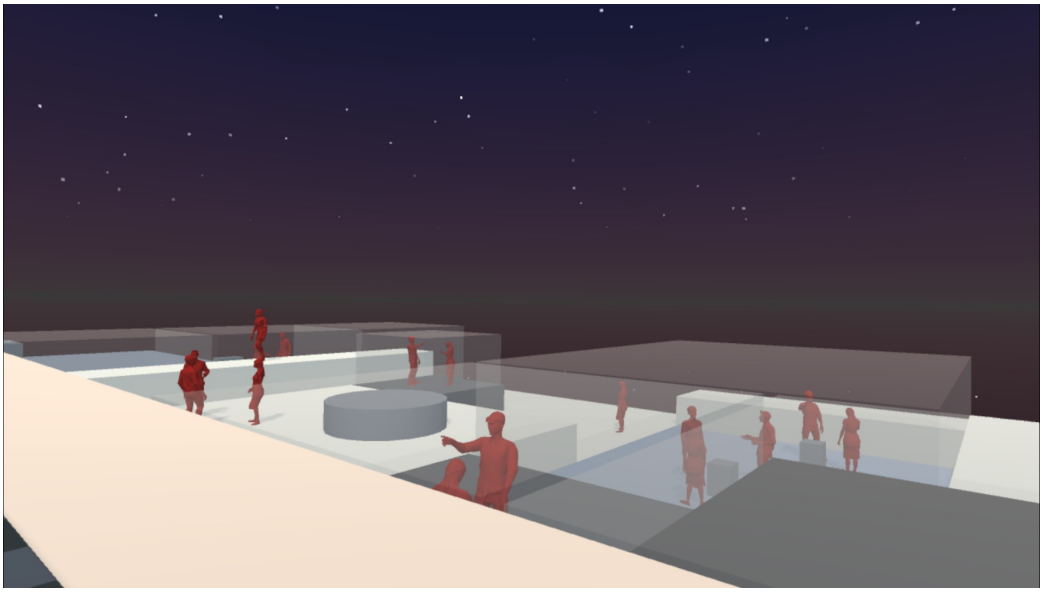
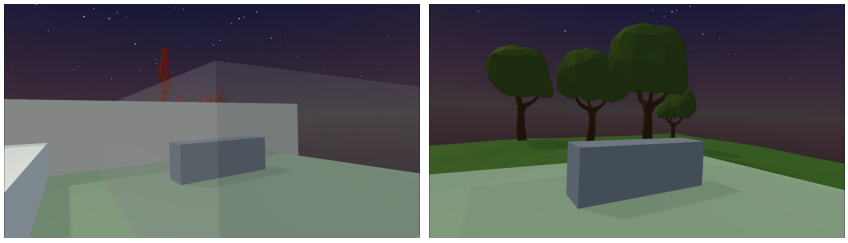


Figure 15: View within "Hyperspatial"

7 Conclusions

Based on the user experiment as well as the other investigations into design parameters, I have designed a functioning prototype of the final space, incorporating the use of levels, virtual rooms, and multi-dimensional spaces. In this design, the landscape is comprised of many different rooms and platforms suited for various activities – some are open, some screened when you enter, and some leading to different spaces entirely. These rooms are arranged at varying heights but not with too big of a gap, allowing users to look across the various rooms and jump easily from one to the next without intermediate 'stairs'. At the highest level, the landscape is crossed by long platforms serving as navigational arteries, offering an overview of the spaces on either side. Instead of traditionally enclosed corridors and rooms, it could be that the organisation of space in VR inverts such notions



(a) Frame 1 **(b)** Frame 2

Figure 16: Transition for 'Alternate Dimension' Platforms

of space, favouring open spaces and elevation changes for ease of navigation.

7.1 Future Work

While the current prototype is functional and integrates many of the lessons learnt from this first semester of research, it remains a relatively conceptual model, offering a possible vision of what the final platform could look like. Having completed most of the research into the fundamental design principles for VR and the 'building blocks' for designing spaces, the next major stage would be to critically examine the agglomeration of spaces and the nature of procedural generation – how exactly should these spaces respond to users? Are some features persistent, or is the entire landscape ever changing according to users needs? What governs the placement of different modules next to each other, their relative heights, or openness? In some areas such as room sizes for larger groups, supplementary research and experiments could still be done to make the data-driven approach more robust.

The final product would be a dynamic, responsive space, fully designed for

the VR environment and taking advantage of its many quirks and opportunities. As in the current design prototype, this landscape would comprise a variety of rooms for a variety of social interactions, while at the same time being easy to navigate and comfortable to use. By embracing the multidimensional, non-physical nature of virtual spaces and the new modes of interaction arising from VR, this thesis project hopes to offer a new architectural paradigm for VR space.

References

Alsop, T. (2020, Jun). *Topic: Virtual reality (vr).* Retrieved from https://www.statista.com/topics/2532/virtual-reality-vr/

AltspaceVR. (2013). *Altspacevr.*

Binkovitz, L. (2019). *Revisiting the social life of small urban spaces.* Retrieved from https://kinder.rice.edu/urbanedge/2019/08/06/revisiting-social-life-small-urban-spaces

Bourdakis, V., & Charitos, D. (1999). Virtual environment design – defining a new direction for architectural education. *Architectural Computing: Virtual Environments*.

Bridges, A., & Charitos, D. (1997). On architectural design in virtual environments. *Design Studies*, *18*(2), 143–154. doi: 10.1016/s0142-694x(97)85457-9

Coulon, R., Matsumoto, E. A., Segerman, H., & Steve, T. (2002). Non-euclidean virtual reality iii: Nil. *arXiv*.

Deocadiz, Z. (2019, Jun). *How to design social vr spaces.* Virtual Reality Pop. Retrieved from https://virtualrealitypop.com/how-to-design-social-vr-spaces-fc06f532ef4a

Ellis, M. (2019, Mar). *How to design for virtual reality: basics and best practices for vr design.* 99designs. Retrieved from https://99designs.com.sg/blog/trends/virtual-reality-design/

Friedberg, A. (2009). *The virtual window: from alberti to microsoft.* Mit Press.

Gaylor, G., & Joudrey, J. (2013). *Vrchat.*

Hansen, M. B. N. (2012). *Bodies in code interfaces with digital media.* Routledge.

Kurbatov, V. (2019, Nov). *10 rules of using fonts in virtual reality.* Inborn Experience (UX in AR/VR). Retrieved from https://medium.com/inborn-experience/10-rules-of-using-fonts-in-virtual-reality-da7b229cb5a1

Ltd, T. S. (2020, Feb). *Virtual reality uses in architecture and design.* TMD STUDIO's Insights. Retrieved from https://medium.com/studiotmd/virtual-reality-uses-in-architecture-and-design-c5d54b7c1e89

Majmudar, K. (2020, Jul). *Function of vistas and views in game design.* NYC Design. Retrieved from https://medium.com/nyc-design/function-of-vistas-and-views-in-game-design-5bd069cfc05f

Neufert, E., Neufert, P., Kister, J., Sturge, D., & Luhman, N. J. (2019). *Architects' data.* Wiley Blackwell.

Nitsche, M. (2005). Games, montage, and the first person point of view. *Georgia Tech.*

Norman, D. A. (2013). *The design of everyday things.* MIT Press.

Rubin, P. (2020). *Future presence: how virtual reality is changing human connection, intimacy, and the limits of ordinary life.* HarperOne.

Samuel, F. (2010). *Le corbusier and the architectural promenade.* Birkhauser.

Schwarzer, M. (2004). *Zoomscape: architecture in motion and media.* Princeton University Press.

Soh, T. (2017). *Oculus platform design: Case study.* Retrieved from http://6thsense.xyz/work/oculus-platform/

Sundstrom, M. (2015, Apr). *How to design for virtual reality — backchannel.* Conde Nast. Retrieved from https://www.wired.com/2015/04/how-to-design-for-virtual-reality/

Team, S. (2019). *Superhot vr.*

Thompson, S. (2020, Apr). *Motion sickness in vr: Why it happens and how to minimise it.* VirtualSpeech. Retrieved from https://virtualspeech.com/blog/motion-sickness-vr

A Test Questionnaire

The test environment was developed for an individual test, due to the difficulty of getting multiple groups together for testing. About half the test participants were asked to test in person with a borrowed headset, while the others responded remotely with their own headsets.

A.1 Survey Instructions

The following instructions were given to participants, along with the file for testing.

Hi! Thanks for taking the time to help with my VR design project :) The test environment contains 6 scenarios, the first 4 testing different spatial configurations and the last 2 are a simple test of room scales. It should take about 15-20 minutes to complete and answer the questionnaire, depending on your speed. Download the test scenario here.

It should work with either a Windows Mixed Reality headset or an Oculus; other headset providers have not been tested. If you find yourself stuck in the floor, it might be an issue with the headset tracking - restarting the application or the headset might help. Apologies in advance for any other bugs or errors.

Scenarios 1-4:
In each scenario, there are 6 groups of figures labeled ABCDEF, with one of

the letters replaced with an X. There is also one group with a highlighted figure. Your task is to find two letters in each scenario - the missing letter X, as well as the letter belonging to the highlighted figure.

Activate the timer (right grip button) before entering each portal and hit it again when you leave. Do note down the answers and time taken after each scenario so you can fill up the form later, or you can fill up the form between scenarios.

Scenarios 5 and 6:
Use the left analog stick (might vary by controller) to control the size of the platform. Stop when you arrive at what you feel is a comfortable space for interaction and report the displayed number.

VR Controls (might vary by controller):
Left grip button to teleport
Left analog stick (left-right) to rotate view
Left directional pad to move
Right grip to interact (activate timer)
Right analog stick (up-down) to resize platforms

After you complete the scenarios, please help answer the following questions!

A.2 Survey Questions

1. How much experience do you have with VR? (1-5)

The following questions were given for each scenario 1-4, with the exception of the locomotion question which was only asked for the first two scenarios, as it is impossible to walk between platforms in the other two scenarios.

1. What was the missing letter? (A-F)
2. What was the highlighted letter? (A-F)
3. Time taken: (Open ended)
4. Primary method of locomotion (Teleportation vs Walking)
5. Ease of navigation
6. Physical discomfort e.g. nausea/dizziness
7. If you were one of the users in such a room, how comfortable would you feel using it for interaction/discussion?
8. Please elaborate on the above question, if possible.

Scenario 5 and 6 only had a single open ended question each, requiring participants to input the scale they settled on for the room size.

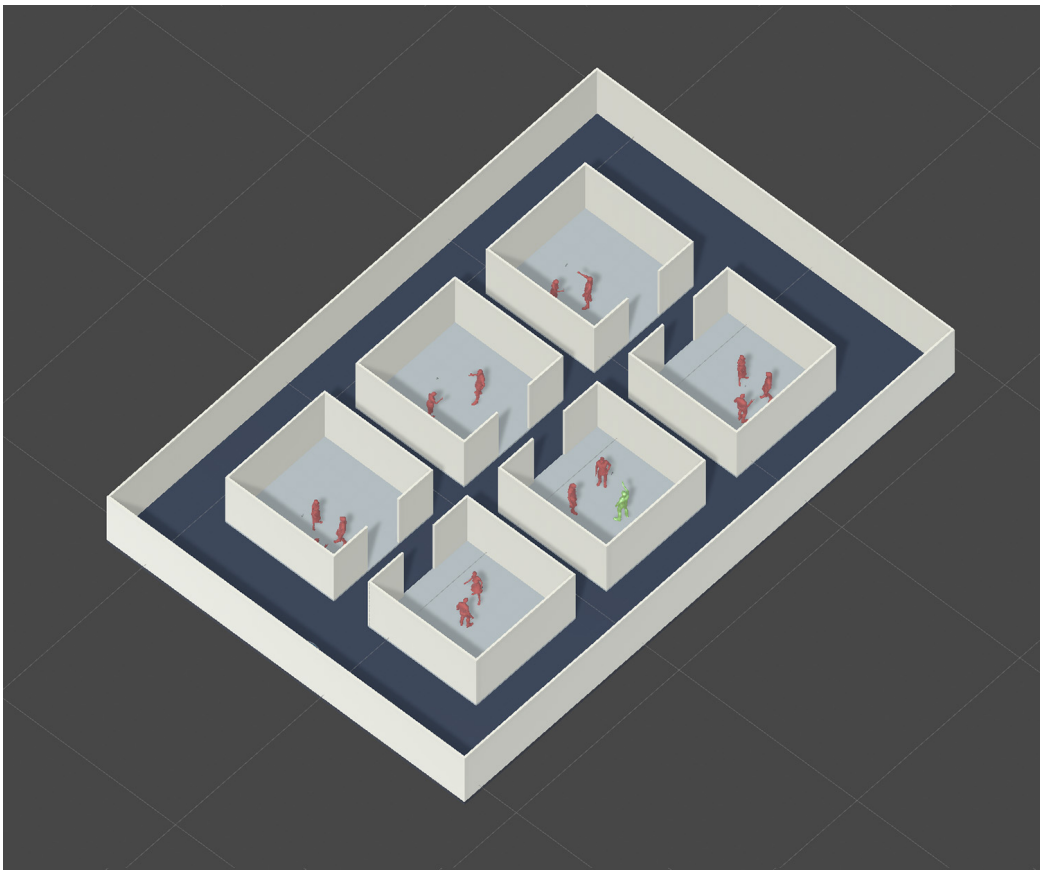


Figure 17: Test Scenario 1

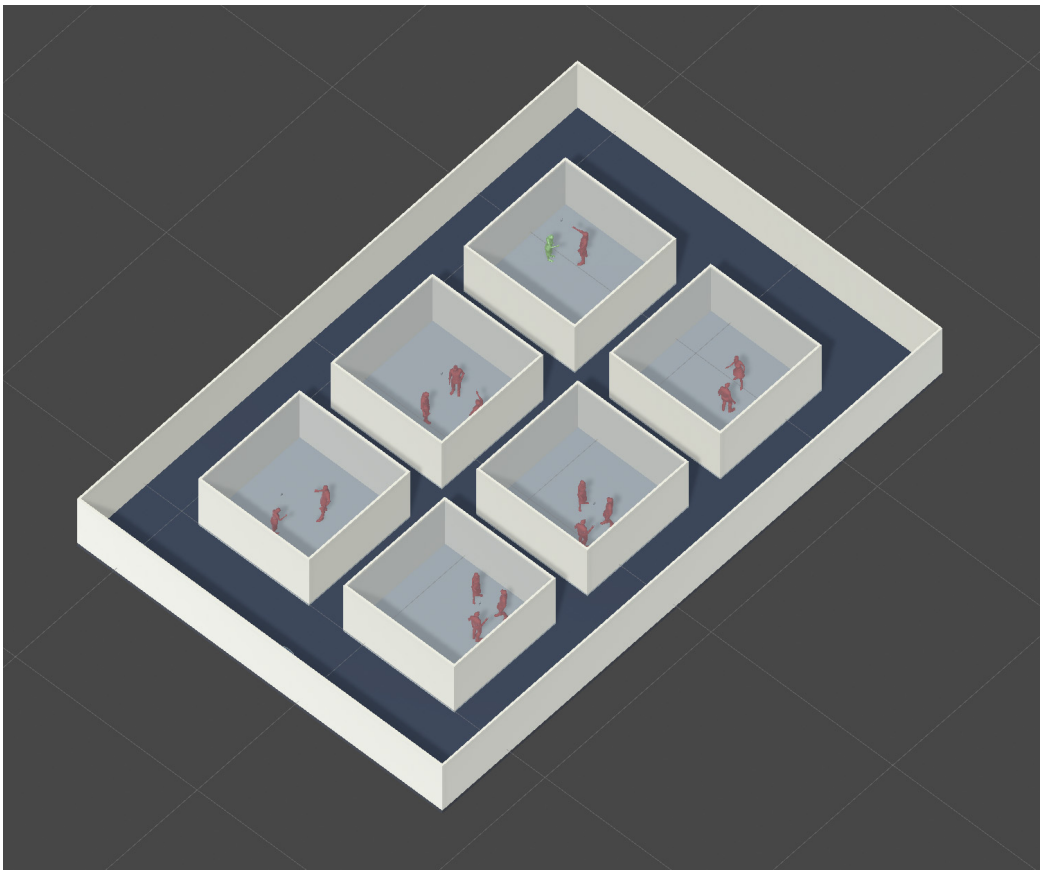


Figure 18: Test Scenario 2

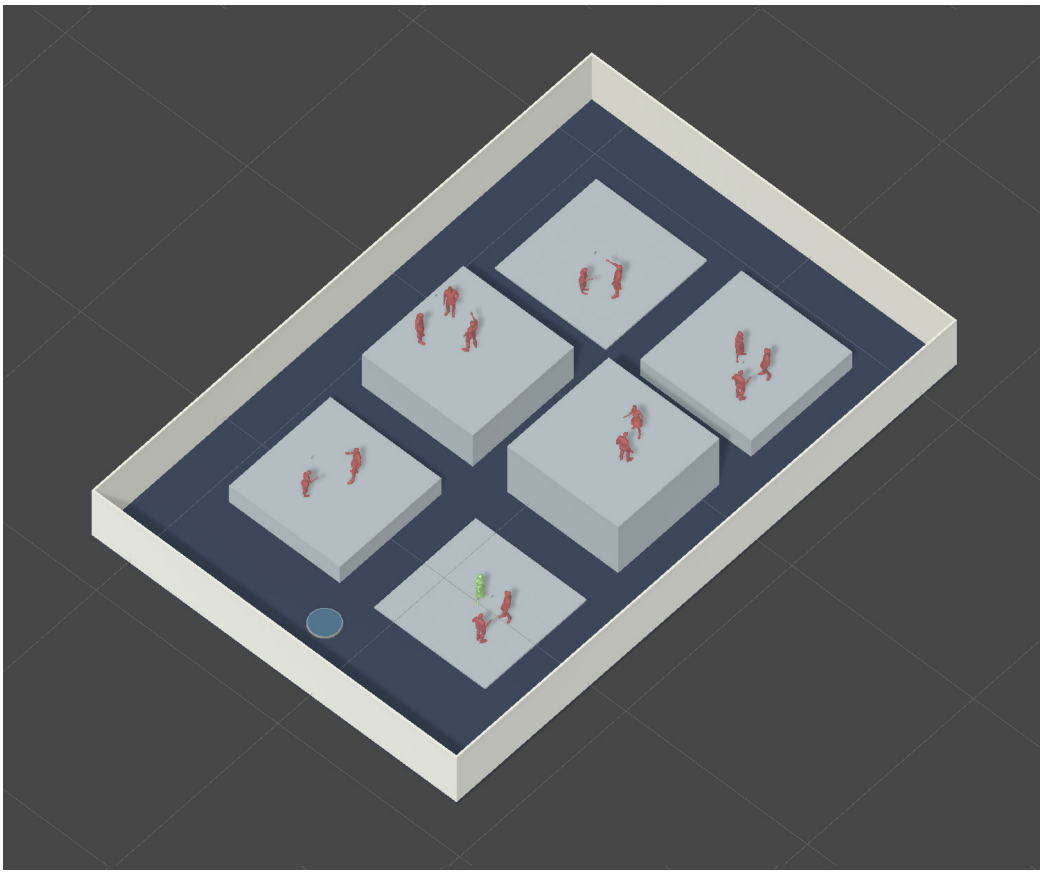


Figure 19: Test Scenario 3

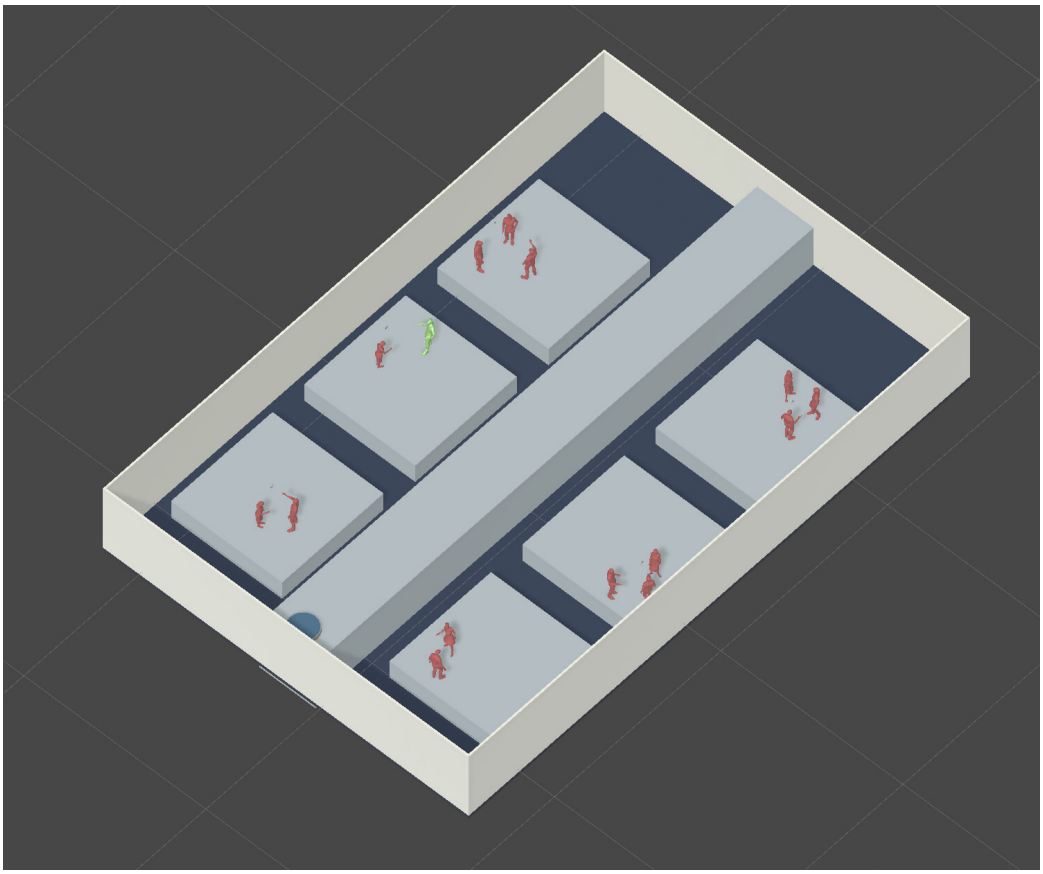


Figure 20: Test Scenario 4

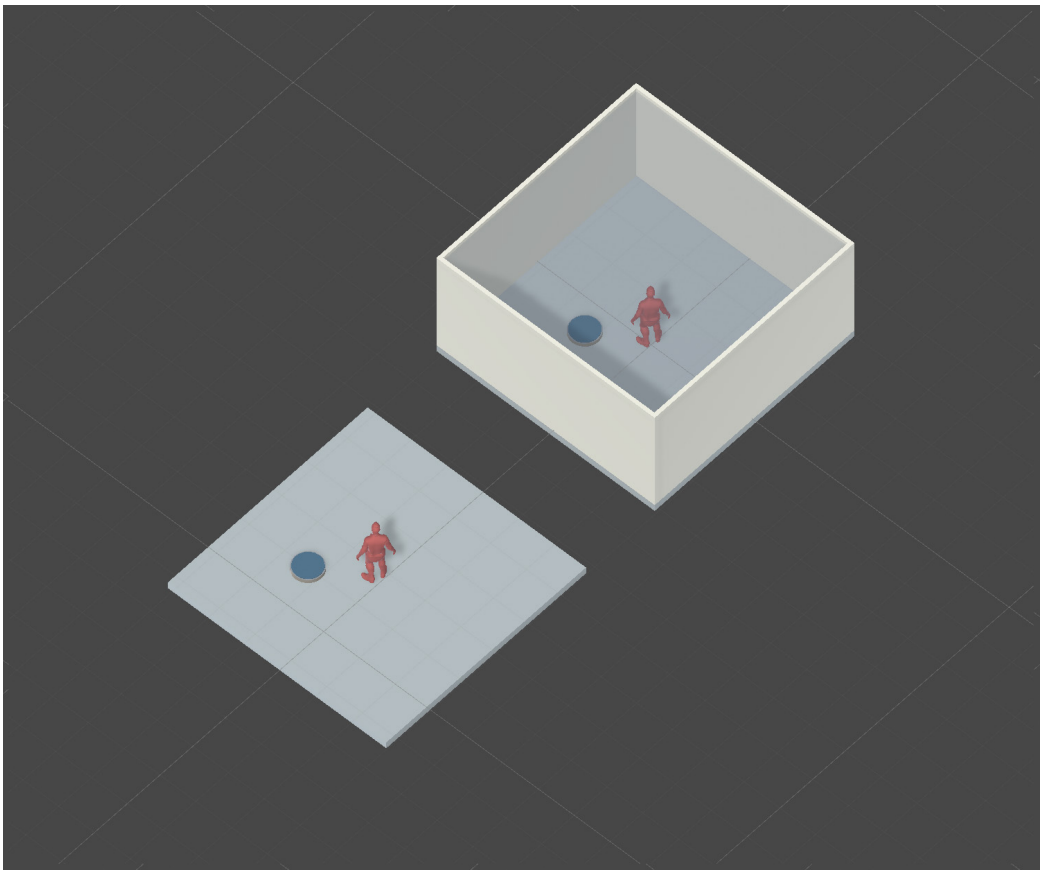


Figure 21: Test Scenarios 5 & 6

A.3 Test Environments

B Test Survey Responses

Scenario 1	Time (s)	Navigation	Discomfort	Conducive
Participant 1	73	3	8	6
Participant 2	150	4	7	6
Participant 3	54	4	8	8
Partitipant 4	103	5	9	7
Participant 5	137	5	9	6
Participant 6	110	7	8	6
	104.5	4.67	8.17	6.50

Comments:

1. Experience to get there is poor but the room itself feels ok.
2. I think the enclosed private space may be quite conducive for discussion. But the experience to get into the room wasn't very good.
3. Closed room feels good for discussion, but the lack of a door detracts from the privacy.
4. Room feels a bit small especially with the narrow corridor.
5. Not much to say about the regular room, might be better with more decoration and objects.
6. I feel like there are too many walls overall. the room itself it quite ok but walking through the corridor feels a little bit uncomfortable, maybe if the walls are not so plain it might not feel as bad

Scenario 1	Time (s)	Navigation	Discomfort	Conducive
Participant 1	41	9	5	10
Participant 2	130	6	6	7
Participant 3	40	8	4	9
Partitipant 4	90	7	5	8
Participant 5	120	10	4	10
Participant 6	114	8	5	9
	89.17	8.00	4.83	8.83

Comments:

1. Room with invisible walls feels great for discussion.

2. I like how the walls become opaque when the user is in the room for perceived privacy. I also like how I was able to walk through the walls to get towards the target (green person) quickly. Overall, I did not have to turn my head often, which helps with navigation comfort.

3. The enclosed space gives a good illusion of privacy, but in reality people would be able to see in?

4. The walls are good for a sense of privacy.

5. Disappearing walls are interesting and feel more comfortable.

Scenario 1	Time (s)	Navigation	Discomfort	Conducive
Participant 1	49	7	8	8
Participant 2	138	6	4	7
Participant 3	31	7	3	6
Partitipant 4	84	6	5	8
Participant 5	129	5	3	7
Participant 6	97	7	5	5
	88.00	6.33	4.67	6.83

Comments:

1. The platforms are difficult to navigate up and down because you have to aim the teleportation

2. would be good if teleportation arc can be tweaked with joystick. Changes in elevation became an issue for me when getting back. I had to turn my head to point in the direction where I need to teleport to, which is pretty nauseating. I wasn't able to just walk backwards as before.

3. Open platforms feel a bit awkward, feels like you might fall off the edge.

4. I feel uncomfortable having to look up and down a lot, also no sense of privacy with the open space.

Scenario 1	Time (s)	Navigation	Discomfort	Conducive
Participant 1	52	7	8	10
Participant 2	106	9	2	7
Participant 3	36	8	4	7
Partitipant 4	75	9	5	8
Participant 5	95	7	3	8
Participant 6	94	7	4	7
	76.33	7.83	4.33	7.83

Comments:

1. Less confusing than the previous room.
2. Similar navigation issue as before. But I like that I have a good overview of all the rooms from an elevated point.
3. Open platforms are still awkward but the overall layout is good with the vantage point.
4. Easier to navigate than the previous room, platforms are still too open.
5. Sense of privacy with the big wall in the middle.

	Scenario 5 Size	Scenario 6 Size
Participant 1	111	187
Participant 3	70	100
Participant 4	85	80
Participant 5	105	143
Participant 6	90	140
Average	92.2	130

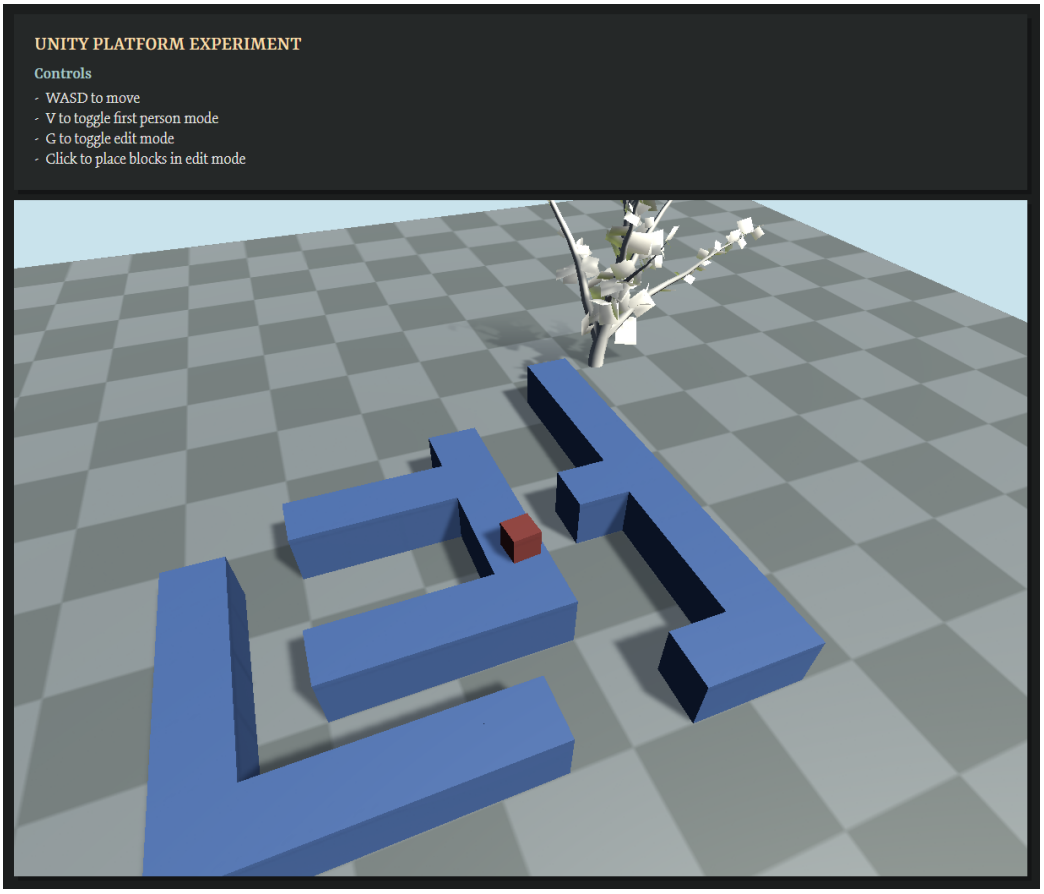


Figure 22: First experiment with Unity3D platform

C Progress and Early Work

The following are a collection of screenshots from earlier iterations and coding experiments that did not fit into the final research narrative.

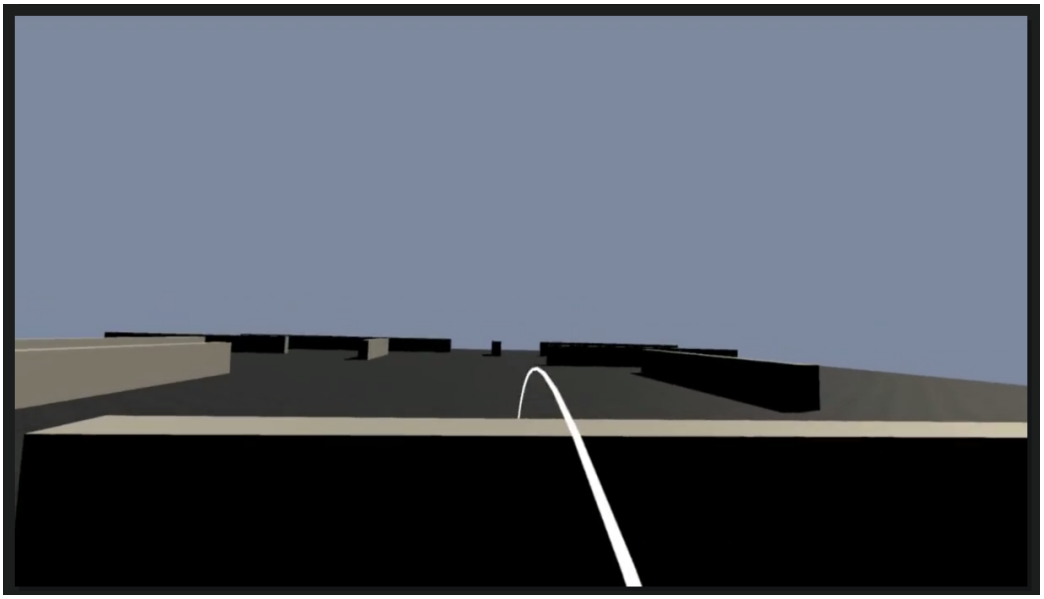


Figure 23: Second experiment with Unity3D platform - teleporting and VR control

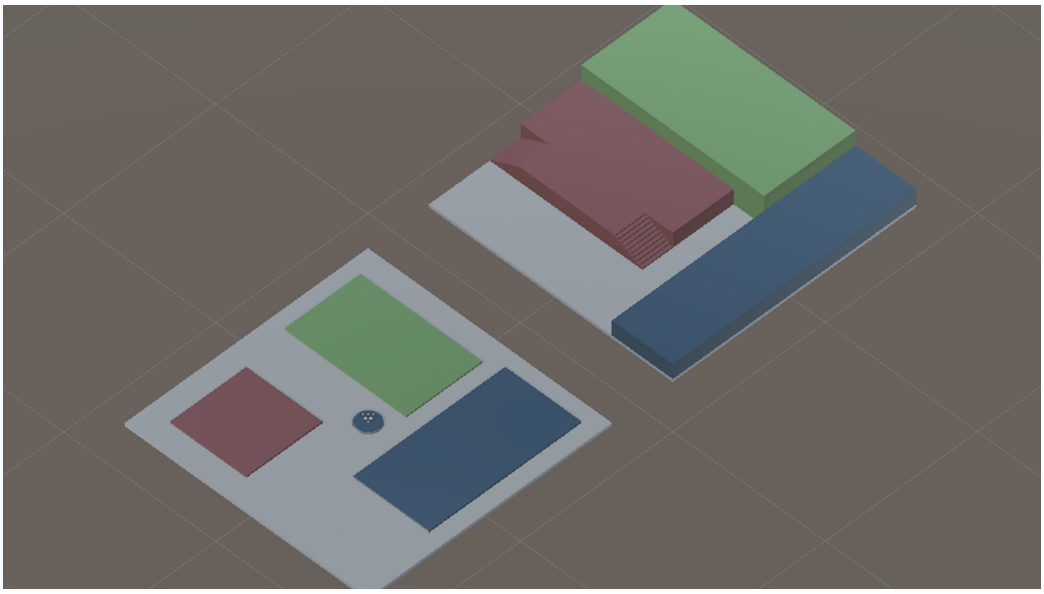


Figure 24: Early concept models of VR space - levels

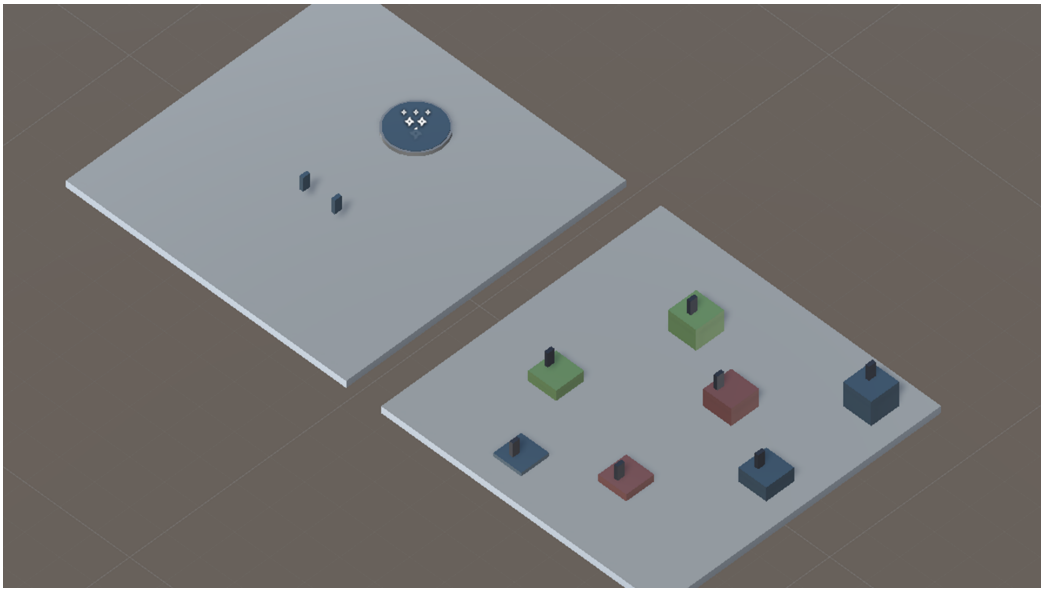


Figure 25: Early concept models of VR space - objects

Figure 26: Simple testing environment for interactables and VR mechanics

Figure 27: Multiplayer test run with other players connected

D Unity VR Scripts in C#

While the nature of the Unity platform means that much of the design and programming work takes place across various objects, variables, and linkages - not to mention plugin support - these are the bulk of the raw scripts manually coded to support the VR environment.

D.1 Player Movement Controller

Script controlling player movement and positioning based on controller inputs.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
```

```
4  using UnityEngine.XR;
5  using UnityEngine.XR.Interaction.Toolkit;
6
7  public class MovementProvider : LocomotionProvider
8  {
9      public float speed = 1.0f;
10     public float gravityMultiplier = 1.0f;
11     public bool gravity = true;
12     public List<XRController> controllers = null;
13     private CharacterController characterController = null;
14     private GameObject head = null;
15     private bool resetClick = true;
16
17     protected override void Awake()
18     {
19         characterController = GetComponent<
               CharacterController>();
20         head = GetComponent<XRRig>().cameraGameObject;
21     }
22
23     // Start is called before the first frame update
24     private void Start()
25     {
26         PositionController();
27     }
28
29     // Update is called once per frame
30     private void FixedUpdate()
31     {
32         PositionController();
```

```
33          CheckForInput();
34          if(gravity)
35              ApplyGravity();
36          else if(characterController.transform.position.y <= 9
                )
37              gravity = true;
38      }
39
40      private void PositionController()
41      {
42          //Get the head in local, playspace ground
43          float headHeight = Mathf.Clamp(head.transform.
                localPosition.y, 0.5f, 2);
44          characterController.height = headHeight;
45
46          //Cut in half, add skin
47          Vector3 newCenter = Vector3.zero;
48          newCenter.y = characterController.height / 2;
49          newCenter.y += characterController.skinWidth;
50
51          //Move the capsule in local space as well
52          newCenter.x = head.transform.localPosition.x;
53          newCenter.z = head.transform.localPosition.z;
54
55          //Apply
56          characterController.center = newCenter;
57      }
58
59      private void CheckForInput()
60      {
```

```
61          foreach (XRController controller in controllers)
62          {
63              if(controller.enableInputActions)
64                  CheckForMovement(controller.inputDevice);
65          }
66      }
67
68      private void CheckForMovement(InputDevice device)
69      {
70          if(device.TryGetFeatureValue(CommonUsages.primary2
               DAxis, out Vector2 position) && gravity)
71              StartMove(position);
72          if(device.TryGetFeatureValue(CommonUsages.secondary2
               DAxisClick, out bool click))
73              if(click && resetClick)
74              {
75                  if(characterController.transform.position.y <
                       = 12)
76                  {
77                      HighJump(true);
78                      resetClick = false;
79                  }
80                  else
81                  {
82                      HighJump(false);
83                      resetClick = false;
84                  }
85              }
86              else if(!click && !resetClick)
87                  resetClick = true;
```

```
88      }
89
90      private void StartMove(Vector2 position)
91      {
92          //Add forward vector
93          Vector3 direction = new Vector3(position.x, 0,
                position.y);
94          Vector3 headRotation = new Vector3(0, head.transform.
                eulerAngles.y, 0);
95
96          direction = Quaternion.Euler(headRotation) *
                direction;
97
98          //Apply speed and move
99          Vector3 movement = direction * speed;
100         characterController.Move(movement * Time.deltaTime);
101         XR_TestVariables.XRMoveCount += Vector3.Magnitude(
                movement * Time.deltaTime);
102     }
103
104     private void ApplyGravity()
105     {
106         Vector3 gravity = new Vector3(0, Physics.gravity.y *
                gravityMultiplier, 0);
107         gravity.y *= Time.deltaTime;
108
109         characterController.Move(gravity);
110     }
111
112     private void HighJump(bool jump)
```

```
113     {
114         Vector3 highJump = new Vector3(0, 11 -
                characterController.transform.position.y, 0);
115
116         if(jump)
117         {
118             characterController.Move(highJump);
119             gravity = false;
120         }
121         else
122         {
123             characterController.Move(-highJump);
124             gravity = true;
125         }
126     }
127 }
```

D.2 Teleporter Node Script

Script handler for teleportation pads between two nodes.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class XRTeleporter : MonoBehaviour
6 {
7     public XRTeleporter linkedTeleporter;
8     public GameObject telePad;
```

```
9       public GameObject teleFX;
10
11      private Vector3 telePosition;
12
13      private int playersColliding = 0;
14      private bool fxOn = false;
15      private Material defaultMat;
16      private Material redMat;
17
18      // Start is called before the first frame update
19      void Start()
20      {
21          defaultMat = telePad.GetComponent<MeshRenderer>().
                material;
22          redMat = Resources.Load("Materials/Select-Red",
                typeof(Material)) as Material;
23
24          telePosition = linkedTeleporter.transform.position;
25          telePosition.y += 2;
26      }
27
28      // Update is called once per frame
29      void FixedUpdate()
30      {
31
32      }
33
34      void OnTriggerEnter(Collider collider)
35      {
36          if (collider.gameObject.tag == "Player")
```

```
37          {
38              playersColliding += 1;
39              TeleporterFXOn();
40
41              collider.GetComponent<PlayerController>().
                    TeleportPlayer(telePosition);
42          }
43      }
44
45      void OnTriggerExit(Collider collider)
46      {
47          if (collider.gameObject.tag == "Player")
48          {
49              playersColliding -= 1;
50              if (playersColliding == 0)
51              {
52                  if (fxOn) TeleporterFXOff();
53              }
54          }
55      }
56
57      void TeleporterFXOn()
58      {
59          telePad.gameObject.GetComponent<MeshRenderer>().
                material = redMat;
60          fxOn = true;
61      }
62
63      void TeleporterFXOff()
64      {
```

```
65          telePad.gameObject.GetComponent<MeshRenderer>().
                material = defaultMat;
66          fxOn = false;
67      }
68  }
```

D.3 Player Controller

Supplementary movement script supporting the teleport function, allowing for a global cooldown to prevent the player from looping between the two tele pads upon landing.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerController : MonoBehaviour
6  {
7      public bool teleportCooldown = false;
8      private Vector3 startPosition;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         startPosition = transform.position;
14     }
15
16     // Update is called once per frame
17     void Update()
```

```
18      {
19          if (transform.position.y < -2)
20          {
21              ResetPlayerPosition();
22          }
23      }
24
25      private void ResetPlayerPosition()
26      {
27          transform.position = startPosition;
28      }
29
30      public void TeleportPlayer(Vector3 target)
31      {
32          if (!teleportCooldown)
33          {
34              transform.position = target;
35              StartTeleportCooldown();
36          }
37      }
38
39      void StartTeleportCooldown()
40      {
41          teleportCooldown = true;
42          StartCoroutine("DelayedTeleportCooldownReset");
43      }
44
45      IEnumerator DelayedTeleportCooldownReset()
46      {
47          yield return new WaitForSeconds(2);
```

```
48          teleportCooldown = false;
49      }
50  }
```

D.4 Early Script for Disappearing Walls

Prototype script in the early testing rounds that handled the collision detection and toggling of invisible walls.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class XR_FadeCollision : MonoBehaviour
6  {
7
8      public List<GameObject> fadeObjects;
9      public bool fadeIn;
10
11     // Start is called before the first frame update
12     void Start()
13     {
14         if (fadeIn)
15         {
16             foreach (GameObject i in fadeObjects)
17             {
18                 i.SetActive(false);
19             }
20         }
```

```
21          else
22          {
23              foreach (GameObject i in fadeObjects)
24              {
25                  i.SetActive(true);
26              }
27          }
28      }
29
30      // Update is called once per frame
31      void Update()
32      {
33
34      }
35
36      void OnTriggerEnter(Collider collider)
37      {
38          if (collider.gameObject.tag == "Player")
39          {
40              if (fadeIn)
41              {
42                  foreach (GameObject i in fadeObjects)
43                  {
44                      i.SetActive(true);
45                  }
46              }
47              else
48              {
49                  foreach (GameObject i in fadeObjects)
50                  {
```

```
                    i.SetActive(false);
                }
            }
        }
    }

    void OnTriggerExit(Collider collider)
    {
        if (collider.gameObject.tag == "Player")
        {
            if (fadeIn)
            {
                foreach (GameObject i in fadeObjects)
                {
                    i.SetActive(false);
                }
            }
            else
            {
                foreach (GameObject i in fadeObjects)
                {
                    i.SetActive(true);
                }
            }
        }
    }
}
```

D.5 Hyperspatial Handler for Platforms / Rooms

Multipurpose script for the rooms and platforms in the Hyperspatial prototype, handling a variety of functions including proper scaling / positioning of walls, collider, and platform, as well as the original disappearing wall script.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class XR_Hyperspace_PlatformManager : MonoBehaviour
{

    public List<GameObject> fadeWalls;
    public List<GameObject> fadeObjects;
    public bool fadeIn;

    public GameObject platform;
    public GameObject collision;
    public float ceilingHeight, floorHeight;
    public float wallMultiplier = 1;
    private Vector3 basePosition;

    // Start is called before the first frame update
    void Start()
    {

```

```
22          //Resize collision box, platform, and walls to
                dynamic height.
23          basePosition = transform.position;
24          Vector3 collisionScale = new Vector3(collision.
                transform.localScale.x - 0.2f / transform.
                localScale.x, ceilingHeight - transform.position.y
                , collision.transform.localScale.z - 0.2f /
                transform.localScale.z);
25          Vector3 platformScale = new Vector3(platform.
                transform.localScale.x, transform.position.y -
                floorHeight, platform.transform.localScale.z);
26          collision.transform.localScale = collisionScale;
27          platform.transform.localScale = platformScale;
28          collision.transform.localPosition += Vector3.Scale(
                collisionScale, new Vector3(0, 0.5f, 0));
29          platform.transform.localPosition -= Vector3.Scale(
                platformScale, new Vector3(0, 0.5f, 0)) - new
                Vector3(0, 0.1f, 0);
30
31          //Move and scale walls to platform perimeter.
32          Vector3 wall1Position = new Vector3(0, 0, 0.5f - 0.05
                f / transform.localScale.z);
33          Vector3 wall1Scale = new Vector3(1, collisionScale.y,
                 0.1f / transform.localScale.z);
34          Vector3 wall2Position = new Vector3(0.5f - 0.05f /
                transform.localScale.x, 0, 0);
35          Vector3 wall2Scale = new Vector3(0.1f / transform.
                localScale.x, collisionScale.y, 1);
36          fadeWalls[0].transform.localScale = wall1Scale;
37          fadeWalls[0].transform.localPosition = wall1Position;
```

```
38          fadeWalls[1].transform.localScale = wall2Scale;
39          fadeWalls[1].transform.localPosition = wall2Position;
40          fadeWalls[2].transform.localScale = wall1Scale;
41          fadeWalls[2].transform.localPosition = -wall1Position
                ;
42          fadeWalls[3].transform.localScale = wall2Scale;
43          fadeWalls[3].transform.localPosition = -wall2Position
                ;
44
45          foreach (GameObject i in fadeWalls)
46          {
47              Vector3 wallScale = new Vector3(i.transform.
                    localScale.x, (ceilingHeight - transform.
                    position.y) * wallMultiplier, i.transform.
                    localScale.z);
48              i.transform.localScale = wallScale;
49              i.transform.localPosition += Vector3.Scale(
                    wallScale, new Vector3(0, 0.5f, 0));
50          }
51
52
53
54
55          fadeObjects.AddRange(fadeWalls);
56          //Set fade on load
57          if (fadeIn)
58          {
59              foreach (GameObject i in fadeObjects)
60              {
61                  i.SetActive(false);
```

```
62              }
63          }
64          else
65          {
66              foreach (GameObject i in fadeObjects)
67              {
68                  i.SetActive(true);
69              }
70          }
71      }
72
73      // Update is called once per frame
74      void Update()
75      {
76
77      }
78
79      void OnEnable()
80      {
81          if (fadeIn)
82          {
83              foreach (GameObject i in fadeObjects)
84              {
85                  i.SetActive(false);
86              }
87          }
88          else
89          {
90              foreach (GameObject i in fadeObjects)
91              {
```

```
92                  i.SetActive(true);
93              }
94          }
95      }
96
97      void OnTriggerEnter(Collider collider)
98      {
99          if (collider.gameObject.tag == "Player")
100         {
101             if (fadeIn)
102             {
103                 foreach (GameObject i in fadeObjects)
104                 {
105                     i.SetActive(true);
106                 }
107             }
108             else
109             {
110                 foreach (GameObject i in fadeObjects)
111                 {
112                     i.SetActive(false);
113                 }
114             }
115             collision.GetComponent<Renderer>().enabled =
                    false;
116         }
117     }
118
119     void OnTriggerExit(Collider collider)
120     {
```

```
121         if (collider.gameObject.tag == "Player")
122         {
123             if (fadeIn)
124             {
125                 foreach (GameObject i in fadeObjects)
126                 {
127                     i.SetActive(false);
128                 }
129             }
130             else
131             {
132                 foreach (GameObject i in fadeObjects)
133                 {
134                     i.SetActive(true);
135                 }
136             }
137             collision.GetComponent<Renderer>().enabled = true
                ;
138         }
139     }
140 }
```

D.6 Hyperspatial Handler for 'Alternate Dimensions'

Special script for the current implementation of 'alternate dimension' platforms, allowing the player to stay in the same place but hiding all other objects according to tags and toggling on the hidden environment. Creates the illusion that the player has been transported to another world while standing still.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class XR_Hyperspace_AltDimension : MonoBehaviour
{
    public List<GameObject> fadeObjects;
    public List<GameObject> exceptionObjects;
    private GameObject[] allObjects;

    // Start is called before the first frame update
    void Start()
    {
        allObjects = UnityEngine.Object.FindObjectsOfType<
            GameObject>();
        foreach (GameObject i in fadeObjects)
        {
            i.SetActive(false);
        }
    }

    // Update is called once per frame
    void Update()
    {

    }

    void OnTriggerEnter(Collider collider)
    {
```

```
29          if (collider.gameObject.tag == "Player")
30          {
31              foreach (GameObject i in allObjects)
32              {
33                  if(i.tag == "Structure" || i.tag =="Dummy")
34                  {
35                      i.SetActive(false);
36                  }
37              }
38              foreach (GameObject i in exceptionObjects)
39              {
40                  i.SetActive(true);
41              }
42              foreach (GameObject i in fadeObjects)
43              {
44                  i.SetActiveRecursively(true);
45              }
46          }
47      }
48
49      void OnTriggerExit(Collider collider)
50      {
51          if (collider.gameObject.tag == "Player")
52          {
53              foreach (GameObject i in allObjects)
54              {
55                  i.SetActiveRecursively(true);
56              }
57              foreach (GameObject i in fadeObjects)
58              {
```

```
                i.SetActiveRecursively(false);
            }
        }
    }
}
```

D.7 Multiplayer Network Manager

Network manager to connect and monitor server activity.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class NetworkManager : MonoBehaviourPunCallbacks
{
    // Start is called before the first frame update
    void Start()
    {
        ConnectToServer();
    }

    void ConnectToServer()
    {
        PhotonNetwork.ConnectUsingSettings();
        Debug.Log("Try Connect To Server...");
    }
```

```
20
21      public override void OnConnectedToMaster()
22      {
23          Debug.Log("Connected To Server.");
24          base.OnConnectedToMaster();
25          RoomOptions roomOptions = new RoomOptions();
26          roomOptions.MaxPlayers = 10;
27          roomOptions.IsVisible = true;
28          roomOptions.IsOpen = true;
29
30          PhotonNetwork.JoinOrCreateRoom("Room 1", roomOptions,
                TypedLobby.Default);
31      }
32
33      public override void OnJoinedRoom()
34      {
35          Debug.Log("Joined a Room");
36          base.OnJoinedRoom();
37      }
38
39      public override void OnPlayerEnteredRoom(Player newPlayer
            )
40      {
41          Debug.Log("A new player joined the room.");
42          base.OnPlayerEnteredRoom(newPlayer);
43      }
44  }
```

D.8 Multiplayer Network Player Controller

Script that animates and syncs the player avatar position for other players, also hides the user's own character model so the view is not intersecting the mesh.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using Photon.Pun;

public class NetworkPlayer : MonoBehaviour
{
        public Transform head;
        public Transform mesh;
        public bool hideSelf = true;

        private GameObject rig;
        private Animator animator;
        //private int interval = 1;
        //private float animUpdateTime = 0;

        private PhotonView photonView;

    // Start is called before the first frame update
    void Awake()
    {
        photonView = GetComponent<PhotonView>();
```

```
24          rig = GameObject.Find("XR Rig");
25          animator = GetComponentInChildren<Animator>();
26      }
27
28      // Update is called once per frame
29      void Update()
30      {
31          if(photonView.IsMine)
32          {
33              if(hideSelf)
34              {
35                  mesh.gameObject.SetActive(false);
36              }
37              MapCharacterPosition(mesh, XRNode.Head);
38              Animate();
39          }
40      }
41
42      void MapPosition(Transform target, XRNode node)
43      {
44          InputDevices.GetDeviceAtXRNode(node).
                TryGetFeatureValue(CommonUsages.devicePosition,
                out Vector3 position);
45          InputDevices.GetDeviceAtXRNode(node).
                TryGetFeatureValue(CommonUsages.deviceRotation,
                out Quaternion rotation);
46
47          target.position = position;
48          target.rotation = rotation;
49      }
```

```
50
51      void MapCharacterPosition(Transform characterModel,
            XRNode node)
52      {
53          InputDevices.GetDeviceAtXRNode(node).
                TryGetFeatureValue(CommonUsages.devicePosition,
                out Vector3 position);
54          InputDevices.GetDeviceAtXRNode(node).
                TryGetFeatureValue(CommonUsages.deviceRotation,
                out Quaternion rotation);
55
56          Vector3 snapTurnOffset = rig.GetComponent<Transform>(
                ).rotation.eulerAngles;
57          Vector3 characterControllerOffset = rig.GetComponent<
                Transform>().position;
58          Vector3 headPosition = new Vector3(
                characterControllerOffset.x,
                characterControllerOffset.y,
                characterControllerOffset.z);
59          Vector3 headRotation = new Vector3(0, rotation.
                eulerAngles.y + (snapTurnOffset.y), 0);
60
61
62          characterModel.position = headPosition;
63          characterModel.rotation = Quaternion.Euler(
                headRotation);
64      }
65
66      private void Animate()
67      {
```

```
68          // Blend between walk/run
69          float blend = rig.GetComponent<CharacterController>()
                .velocity.magnitude;
70          animator.SetFloat("Move", blend);
71      }
72  }
```

D.9 Multiplayer Network Player Spawner

Simple handler for connecting and disconnecting players.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Photon.Pun;
5
6  public class NetworkPlayerSpawner : MonoBehaviourPunCallbacks
7  {
8      private GameObject spawnedPlayerPrefab;
9      public override void OnJoinedRoom()
10     {
11         base.OnJoinedRoom();
12         spawnedPlayerPrefab = PhotonNetwork.Instantiate("
               Network Player", transform.position, transform.
               rotation);
13     }
14
15     public override void OnLeftRoom()
16     {
```

```
17          base.OnLeftRoom();
18          PhotonNetwork.Destroy(spawnedPlayerPrefab);
19      }
20  }
```

D.10 Multiplayer Object Syncing

Script that allows multiple players to take control of objects during a single session.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR;
5  using UnityEngine.XR.Interaction.Toolkit;
6  using Photon.Realtime;
7  using Photon.Pun;
8
9  [RequireComponent( typeof( PhotonView ) )]
10 public class GrabbablePhotonOwnership : MonoBehaviourPun
11 {
12     private XRGrabInteractable interactable = null;
13     private void Awake()
14     {
15         interactable = GetComponent<XRGrabInteractable>();
16     }
17     private void OnEnable()
18     {
```

```
19          interactable.onSelectEnter.AddListener(
                RequestOwnership);
20      }
21      private void RequestOwnership(XRBaseInteractor interactor
            )
22      {
23          if( this.photonView.Owner == PhotonNetwork.
                LocalPlayer )
24          {
25              Debug.Log( "Not requesting ownership. Already
                    mine." );
26              return;
27          }
28
29          this.photonView.RequestOwnership();
30          Debug.Log( "Requesting ownership" );
31      }
32
33  }
```

D.11 Material Change Handler

Simple script that creates abstracted functions for material change which can then be hooked up to Hover/Select/Activate events from XR Controllers in the Unity interface.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
```

```
4
5  public class GrabbableColorChanger : MonoBehaviour
6  {
7      Material redMat;
8      Material blueMat;
9      Material greenMat;
10     Material clearMat;
11     Material defaultMat;
12
13     void Start()
14     {
15         defaultMat = GetComponent<MeshRenderer>().material;
16         redMat = Resources.Load("Materials/Select-Red",
               typeof(Material)) as Material;
17         blueMat = Resources.Load("Materials/Select-Blue",
               typeof(Material)) as Material;
18         greenMat = Resources.Load("Materials/Select-Green",
               typeof(Material)) as Material;
19         clearMat = Resources.Load("Materials/Select-Clear",
               typeof(Material)) as Material;
20     }
21
22     public void SetMaterialDefault()
23     {
24         GetComponent<MeshRenderer>().material = defaultMat;
25     }
26     public void SetColorRed()
27     {
28         GetComponent<MeshRenderer>().material = redMat;
29     }
```

```
30
31      public void SetColorBlue()
32      {
33          GetComponent<MeshRenderer>().material = blueMat;
34      }
35
36      public void SetColorGreen()
37      {
38          GetComponent<MeshRenderer>().material = greenMat;
39      }
40
41      public void SetColorClear()
42      {
43          GetComponent<MeshRenderer>().material = clearMat;
44      }
45  }
```