



PEDRO ANJOS FERREIRA

B.Sc. in Computer Science and Engineering

ADAPTATIVE PROCEDURAL GENERATION OF VIRTUAL REALITY ENVIRONMENTS FOR NAVIGATION IN LIMITED PHYSICAL SPACES

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2024



ADAPTATIVE PROCEDURAL GENERATION OF VIRTUAL REALITY ENVIRONMENTS FOR NAVIGATION IN LIMITED PHYSICAL SPACES

PEDRO ANJOS FERREIRA

B.Sc. in Computer Science and Engineering

Adviser: Rui Pedro da Silva Nóbrega

Assistant Professor, NOVA School of Science and Technology

Co-adviser: Ana Rita Oliva Rebelo

Researcher, NOVA School of Science and Technology

Examination Committee

Chair: Pedro Abílio Duarte de Medeiros

Associate Professor, NOVA School of Science and Technology

Rapporteur: Alfredo Manuel dos Santos Ferreira Júnior

Associate Professor, Instituto Superior Técnico da Universidade de Lisboa

Adviser: Rui Pedro da Silva Nóbrega

Assistant Professor, NOVA School of Science and Technology

Adaptative Procedural Generation of Virtual Reality Environments for Navigation in Limited Physical Spaces

Copyright © Pedro Anjos Ferreira, NOVA School of Science and Technology, NOVA University Lisbon.

The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

ACKNOWLEDGEMENTS

To begin, I would like to thank my supervisor, Rui Nóbrega, for guiding me on the path of virtual reality and providing me with the tools and advice necessary to achieve my goals. I would also like to thank NOVA LINCS for providing the resources to develop my work.

Next, I would like to wholeheartedly thank my co-supervisor, Ana Rita Rebelo, for being a mentor, counselor, friend, and, most of all, for your patience. You were always ready to listen to my ambitious ideas and help whenever possible. You consistently believed in me and fought to ensure I never lacked anything. Without you, this final stretch of my five-year journey would not have been the same. From the bottom of my heart, thank you.

To my friends over these five years, thank you. Without you, the late nights studying, gaming, or celebrating would not have been as fun as they were. I know that wherever life takes me, I can always count on your support.

Last but not least, I want to thank my parents, brother, and the rest of my family with whom I share my life. You have always been, and will remain, the pillars of my life. Wherever I go, I will carry your love and care with me. Thank you!

”

*“For many events, roughly 80% of the effects come
from 20% of the causes.”*

— **Vilfredo Pareto**

ABSTRACT

This dissertation aims to enable walking in Virtual Reality (VR) within limited physical spaces by adapting procedurally generating Virtual Environments (VE) that align with the physical play area and user constraints or preferences.

A key challenge in creating expansive virtual worlds is the user's spatial limitations, which can undermine immersion and presence – essential components of a compelling VR experience. To overcome this, this dissertation researches techniques for adapting existing virtual layouts or generating new ones, optimizing them to fit the available physical space. This is achieved through techniques such as spatial manipulation, impossible spaces, portals, corridors, and redirected walking to facilitate seamless navigation.

The researched tool also offers customization options, allowing users to modify parameters such as room connection techniques and the number of rooms generated. These parameters tailor the VE to specific scenarios and user needs. A notable feature is the degree of realism parameter, which includes sub-parameters such as the *realistic factor*, *layout preservation*, and *layout size*. These factors determine how the VE is adjusted, ranging from highly realistic layouts resembling real-world spaces to more unrealistic settings where spatial manipulations are more prevalent.

Evaluations were conducted to test the tool's effectiveness in generating VEs that fit the physical space. Early interviews with Human-Computer Interaction (HCI) experts provided feedback on how to refine the tool's development. A user study positively assessed walking experiences in the generated environments and revealed positive results on motion sickness, immersion, presence, and overall comfort. Load tests were used to discover the limits and evaluate the tool's ability to generate multiple layouts with varying parameters.

Keywords: Virtual Reality, Walking, Locomotion, Redirected Walking, Impossible Spaces, Procedural Generation,

RESUMO

Esta dissertação tem como objetivo possibilitar a caminhada em Realidade Virtual (RV) dentro de espaços físicos limitados, adaptando a geração procedimental de Ambientes Virtuais (AV) que se alinham com a área física disponível e as restrições ou preferências do utilizador.

Um dos principais desafios na criação de mundos virtuais expansivos é a limitação espacial do utilizador, que pode comprometer a imersão e a presença — componentes essenciais para uma experiência de RV envolvente. Para superar este desafio, esta dissertação investiga técnicas para adaptar layouts virtuais existentes ou gerar novos, otimizando-os de forma a adequar-se ao espaço físico disponível. Isto é alcançado através de técnicas como manipulação espacial, espaços impossíveis, portais, corredores e caminhada redirecionada, permitindo uma navegação fluida.

A ferramenta investigada também oferece opções de personalização, permitindo que os utilizadores modifiquem parâmetros como técnicas de conexão de salas e o número de salas geradas. Estes parâmetros ajustam o AV a cenários específicos e às necessidades dos utilizadores. Uma funcionalidade notável é o parâmetro grau de realismo, que inclui subparâmetros como o *fator de realismo*, *preservação do layout* e *dimensão do layout*. Estes fatores determinam como o AV é ajustado, desde layouts altamente realistas que se assemelham a espaços do mundo real até cenários mais irreais, onde as manipulações espaciais são mais prevalentes.

Foram realizadas avaliações para testar a eficácia da ferramenta na geração de AVs que se adequam ao espaço físico. Entrevistas iniciais com especialistas em Interação Humano-Computador (IHC) forneceram feedback sobre como refinar o desenvolvimento da ferramenta. Um estudo com utilizadores avaliou positivamente as experiências de caminhada nos ambientes gerados e revelou resultados positivos em relação a enjoo, imersão, presença e conforto geral. Testes de carga foram realizados para descobrir os limites e avaliar a capacidade da ferramenta de gerar múltiplos layouts com parâmetros variados.

Palavras-chave: Realidade Virtual, Caminhada, Locomoção, Caminhada Redirecionada,

Espaços Impossíveis, Geração Procedimental,

CONTENTS

List of Figures	x
List of Tables	xii
Acronyms	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives and Research Questions	2
1.3 Solution Overview	3
1.4 Contributions	4
1.5 Document Structure	5
2 State of Art Review	6
2.1 Navigation in Virtual Reality	6
2.1.1 Locomotion Techniques in Desktop Applications	8
2.1.2 Locomotion Techniques in VR	9
2.1.3 Redirected Walking	10
2.1.4 Impossible and Flexible Spaces	15
2.2 Procedural Content Generation	19
2.2.1 Generating Connected Spaces	20
2.2.2 Adaptative Algorithm	25
2.2.3 Applications in VR	27
2.3 Summary	30
3 Analysis and System Architecture	31
3.1 Requirements and Methodology	31
3.2 Architecture	33
3.3 Virtual Layout Generation	35
3.3.1 Simple Room Displacement	35

3.3.2	Layout Room Connection	36
3.3.3	Room Stride Calculation	37
3.4	VR Room Connection	38
3.4.1	Portal	39
3.4.2	Corridor	40
3.4.3	Central Hub	46
3.4.4	Scoring System	48
3.5	Adaptive and Additional Features	51
3.5.1	Adaptiveness in Tool	51
3.5.2	Virtual Layout Import and Export	52
3.5.3	VR Guardian Boundaries	53
3.6	Summary	53
4	Implementation	55
4.1	Virtual Layout Generation	55
4.1.1	Virtual Layout Creation	56
4.1.2	Room Creation	60
4.1.3	Virtual Layout Persistence	62
4.2	Room Connection Techniques	64
4.2.1	Scoring System	64
4.2.2	Portals	65
4.2.3	Corridors	67
4.2.4	Central Hubs	70
4.3	Adaptive Systems	71
4.3.1	Tool Menu	72
4.3.2	Virtual Layout Adaptability	72
4.3.3	Guardian Boundaries	73
4.3.4	Additional Algorithms	74
4.4	Summary	75
5	Evaluation	76
5.1	Interviews and Feedback	76
5.1.1	Protocol	77
5.1.2	Results	77
5.1.3	Discussion	79
5.2	User Study	80
5.2.1	Protocol	80
5.2.2	Data Collection	81
5.2.3	Results	82
5.2.4	Presence	82
5.2.5	Cybersickness	84

5.2.6	Spatial Awareness and Orientation	84
5.2.7	User Experience and Naturalness	85
5.2.8	Completion Time	85
5.2.9	Discussion	86
5.3	Load Test	88
5.3.1	Protocol	88
5.3.2	Results	89
5.3.3	Discussion	91
5.4	Final Discussion	92
5.5	Summary	93
6	Conclusion	95
6.1	Future Work	96
	Bibliography	98
	Appendices	
A	<i>User Study Questions</i>	106

LIST OF FIGURES

1.1	Solution Overview Image	4
2.1	VR in Museums	7
2.2	Interconnected VR	8
2.3	Categories Locomotion Techniques	10
2.4	Four types of gains used to manipulate user senses	11
2.5	Taxonomy Redirected Walking	12
2.6	Example Player in VR	14
2.7	Combining Locomotion Techniques	14
2.8	Example of Impossible Spaces	16
2.9	Portal Game	16
2.10	Impossible Space Experiment	17
2.11	Example Flexible Spaces	18
2.12	PCGExample	20
2.13	Different PCG texture generation	20
2.14	PDGExample2	21
2.15	Different PCG algorithms groups	23
2.16	Example Cellular Automata	23
2.17	Example Voronoi Tessellation	24
2.18	Overview Adaptivity architectural principles	25
2.19	Graphic of State of Flow	26
2.20	Left For Dead 2 Game	27
2.21	Different states of AI Director	27
2.22	PCG VR from 3D reconstructed space	28
2.23	VRoamer Illustration	29
2.24	Example Redirected Spaces	29
3.1	System Diagram of tool	33
3.2	Propose Solution Diagram	34
3.3	Simple Room Displacement Diagram	36

3.4	Layout Room Connection Diagram	37
3.5	Portal Guideline	39
3.6	Corridor Guideline	41
3.7	Seamless Stride Transition in Corridor	44
3.8	Stride change within the corridor.	46
3.9	Central Hub Guideline	47
3.10	Tool Pipeline	52
3.11	VR Guardian Example	53
4.1	Layout Generation	57
4.2	Layout Overview Examples	58
4.3	Wall Door Cut	58
4.4	Wall Door Cut Mesh	59
4.5	Room Factory	61
4.6	Room Creation Examples	61
4.7	Exporting Virtual Layout	62
4.8	Importing Virtual Layout	63
4.9	External Virtual Layout Example	64
4.10	Layout User Perspective Examples	66
4.11	Portals Example	67
4.12	Corridor Node	68
4.13	Corridor Example	69
4.14	CH Examples	71
4.15	Elevator prefab	71
4.16	Menu Example	72
4.17	Starting Room Example	73
5.1	User Studies Task	81
5.2	IPQ results	83
5.3	SSQ results	84
5.4	Virtual Layout Drawings	85
5.5	UX results	86
5.6	Time results	87
5.7	Load Tests Results	90

LIST OF TABLES

2.1	Overview of the different techniques used in the research papers.	30
3.1	Scoring system degree of realism parameters weight	50
5.1	Overview of the feedback collected during the interviews with HCI researchers	78
5.2	User Distribution per technique	83
5.3	IPQ descriptive statistics	83
5.4	UX descriptive statistics	86
5.5	Computers Components	89
5.6	Mean score of number of rooms in each computer	91

ACRONYMS

AR	Augmented Reality (<i>p. 7</i>)
CH	Central Hub (<i>pp. 31, 34, 46–49, 55, 70, 75</i>)
DT	Delaunay Triangulation (<i>pp. 34–37, 56</i>)
IVR	Interconnected Virtual Reality (<i>pp. 7, 8</i>)
MST	Minimum Spanning Tree (<i>pp. 34, 36, 37, 56</i>)
PCG	Procedural Content Generation (<i>pp. 1–3, 5, 6, 19–31, 33, 35, 55, 95–97</i>)
PCG-G	Procedural Content Generation for Games (<i>pp. 19, 21</i>)
PDG	Procedural Dungeon Generation (<i>pp. 6, 20–22</i>)
SRD	Simple Room Displacement (<i>pp. 35, 36, 41, 51, 55, 60, 61</i>)
SS	Scoring System (<i>pp. 47–52, 55, 57, 64–68, 70, 72, 73</i>)
VE	Virtual Environment (<i>pp. 1–6, 8, 10–12, 15–18, 26–28, 30–33, 38–40, 44, 46, 48, 51–53, 70, 82, 84, 85, 87, 93, 94, 97</i>)
VR	Virtual Reality (<i>pp. 1–10, 13, 15, 17–19, 27–31, 33, 38, 51–53, 75, 86, 93–95</i>)

INTRODUCTION

Over the past few years, [Virtual Reality \(VR\)](#) has rapidly evolved, capturing widespread interest by allowing users to immerse themselves entirely in [Virtual Environment \(VE\)](#) from the convenience of their own homes. As an example, users can explore a virtual representation of an apartment they want to buy [13] or explore a virtual museum with the pieces of art they want to see without having to travel to a different country [52]. Designers can benefit from it by using [VR](#) to prototype new design ideas, saving a lot of resources, and users can benefit from more immersive experiences.

Another noteworthy area of focus is [Procedural Content Generation \(PCG\)](#) and its adaptability to produce content with minimal designer intervention while ensuring coherence and alignment with the designer's specifications. [PCG](#) has many uses, and in conjunction with [VR](#), can be used to generate different scenarios for a user to explore in [VR](#), such as virtual apartments or random emergency scenarios [49] for firefighters to train and be better prepared.

1.1 Motivation

This dissertation aims to tackle one of the big problems that [VR](#) users face, the fact that although [VR](#) allows the creation of endless virtual worlds, many users are constrained by the size of their available physical space — often as small as 2.4 x 2.2 meters [34, 2] — such as a room or office. Since the virtual world can be much larger than the real world, using teleport — a locomotion technique where users employ a handheld controller to point to the desired location for instantaneous transportation — has been the preferred choice. The teleportation technique mitigates the challenge of limited physical space, as users do not need to move physically during exploration. However, teleporting is a technique that is very different from walking, which can compromise the immersion of the [VR](#) experience [50]. Walking is established as a more immersive technique [35]. However, its usage is less common because it mimics human walking in the real world, thus requiring a lot more physical space that ordinary users do not have available.

Given this challenge, the main focus of this dissertation was to research and design a

tool that can procedurally generate large virtual worlds that can be explored using walking while being limited by the physical space. Manual creation of such environments is time-consuming and challenging because the virtual space needs to fit inside the small physical space and still guarantee seamless navigation. PCG was used to facilitate the creation of these large virtual worlds, benefiting from its potential to generate VE based on known initial characteristics (e.g., the size of the playing area and number of rooms). Consequently, the emphasis could shift towards optimizing the user's exploration experience and which techniques could be used to enable walking as a locomotion technique. For instance, techniques such as Redirected Walking [38] allow the redirection of users away from the boundaries of the physical space. Others, such as Impossible Spaces [57], harness non-Euclidean geometry properties to create spaces that defy reality, allowing for self-overlapping layouts to exist and better use the available space.

1.2 Objectives and Research Questions

Based on this dissertation's motivation and goals, these were the main objectives proposed:

- **Research VR walking techniques:** Research techniques that can be used to enable walking in VR within the limited physical space.
- **Procedural generation and adjustment of virtual layouts:** Develop a system to procedurally generate or receive VE, dividing them into smaller, manageable spaces (i.e., rooms) and making seamless adjustments to fit these rooms in small physical spaces in VR.
- **Enable navigation and customization of virtual layouts in small spaces:** Enable seamless navigation of adjusted VE in small physical spaces using research techniques while providing automatic layout adjustments and user customization options to enhance the navigation and exploration experience.
- **Conduct user studies and analyze the acquired results:** Conduct a user study to evaluate the tool's effectiveness and navigation experience in VR of the adjusted VE fitted into the small physical space.

Three research questions were established based on these objectives.

- **Research Question 1** - How can a tool seamlessly integrate diverse walking techniques in small spaces?

The first research question explores how various techniques, such as Impossible Spaces and portals, can be combined to enable walking and achieve seamless locomotion in the generated VR spaces.

- **Research Question 2** - What are the physical and virtual boundaries beyond which the user experience may be compromised?

This question explores the limits of enabling walking in small physical spaces, aiming to provide valuable insights for future projects. It explicitly considers how large the **VE** can be relative to the physical space available and determines the minimum physical area required to generate valid layouts. Additionally, it evaluates the extent to which each technique can support comfortable navigation within these constraints.

- **Research Question 3** - How can **PCG** enable users to customize a large adaptive virtual world according to their preferences and explore it in **VR**?

This final question explores how **PCG** can be customized through parameter adjustments, enabling the creation of virtual worlds that align with user preferences while remaining explorable using walking in small physical spaces. It also considers how allowing users to modify these parameters in real time (e.g., replacing a portal with a corridor) makes the virtual layout more dynamic. Finally, it questions how the system can automatically and gradually adapt the virtual layout to match the user's desired customizations.

1.3 Solution Overview

A tool was developed to address the established objectives and research questions. This tool and its various systems will be detailed throughout this document, but a brief overview is presented here. As a use case, a virtual apartment tour (see [Figure 1.1](#)) was used, where users could navigate in **VR** within a small 2.5m x 2.5m physical space. Alternatively, the tool can leverage **PCG** to generate the **VE** for users to explore. To enable navigation in **VR** within such a small physical area, the tool takes the virtual apartment and divides it into smaller spaces (i.e., rooms). These rooms are then adjusted to fit the physical space, allowing users to explore them in **VR**. Once adjusted, the tool applies three techniques studied and implemented to facilitate room transitions without requiring users to leave the physical space. These techniques are:

- **Portals:** Positioned anywhere in a room, portals allow users to preview the connected room and offer an instantaneous transition. Marked with a red door frame.
- **Corridors:** Maintain the original room door positions, guiding users smoothly through a corridor from one room to another. Marked with a blue door frame.
- **Central Hubs:** External rooms not part of the original layout, central hubs connect multiple rooms within the layout. Doors leading to central hubs have green door frames.



Figure 1.1: The virtual apartment is fitted into a small physical space using three room connection techniques — Portals, Corridors, and Central Hubs — to enable walking.

After the techniques are applied, the virtual layout is ready for exploration. Within the virtual layout, users can navigate through all the virtual rooms, modify the techniques used to connect rooms, generate entirely new virtual layouts for exploration, and export or import different virtual layouts as needed.

A user study was conducted to evaluate the tool’s capability of adapting virtual environments for walking in small spaces using VR. This study aimed to gather data on the tool’s effectiveness, assess the user navigation experience, and provide insights to answer the research questions.

1.4 Contributions

This thesis contributes to a solution that enables walking in VR and in small physical spaces through procedural generation, various room connection techniques, and adaptive algorithms. The proposed solution will include the following:

- **Generation of virtual environments:** The tool adjusts a given virtual layout or generates a new one to be explored in a physical and limited play area. The tool does not generate one if a virtual layout is provided as a parameter.
- **Enabling walking using different room connection techniques:** The tool, based on the parameters provided by the user, such as the realistic factor or the preservation of the original layout, among others, fits the virtual layout into the physical space using different techniques that enable walking. For this, the tool arranges the virtual rooms and, depending on the parameters, selects portals, corridors, or central hubs to connect them.
- **Adaptive environment:** After the VE is generated and fitted, the user can start exploring it in VR, and the adaptiveness of the environment starts. These adaptive algorithms allow the user to change the tool’s parameters and change the virtual world in real-time. They also make automatically and gradual changes to the VE to avoid disruptions to the user navigation.

- **Fit into physical space:** The tool, by adjusting user strides in the [VE](#), can also modify virtual layouts to accommodate various physical play area sizes, allowing users to walk naturally within those layouts despite spatial limitations.

Publication A Late-Breaking Work [49] has been published and presented at the CHI 2024 (A*) conference, presenting the proposed solution, its initial development steps, and Human-Computer Interaction (HCI) experts validation.

Submissions A full paper has been submitted to IEEE VR (A*), presenting the proposed room connection techniques and the results of the user studies.

1.5 Document Structure

In [Chapter 2](#) it will be presented a literature review covering the field of [VR](#), various locomotion techniques, and methods for overcoming the challenges of walking in [VR](#), such as Redirected Walking, Impossible Spaces, and Flexible Spaces. It also explores the application of [PCG](#) and its algorithms, including adaptive algorithms and their role in [VR](#) environments. [Chapter 3](#) presents an in-depth analysis and outlines the system architecture. This includes the process of generating virtual layouts, the theory behind each [VR](#) room connection technique, adaptiveness in the tool, and additional features. [Chapter 4](#) focuses on the tool's implementation, detailing how each concept was realized, along with the challenges faced during development. All tests and studies conducted on the tool are outlined in [Chapter 5](#), while [Chapter 6](#) offers conclusions and discusses potential future developments for the tool.

STATE OF ART REVIEW

This chapter aims to provide an overview of the two main areas of study related to **Virtual Reality (VR)** and **Procedural Content Generation (PCG)**. The topic of **VR** covers concepts regarding locomotion techniques, including Redirected Walking and Impossible and Flexible Spaces. Regarding **PCG**, this concept will be studied, along with its sub-field, **Procedural Dungeon Generation (PDG)**, and with special relevance in **PDG**, the applications of **PCG** in **VR** and adaptive algorithms.

By analyzing these areas, it is possible to can grasp the current relevant works that can be used as the basis for the study to create large virtual worlds that can fully be explored through walking and with limited physical space available.

First, the current state of **VR** technology will be analyzed. Then, all current locomotion techniques related to desktop applications and **VR** will be discussed, along with their differences and applicability. To close the **VR** section, two crucial areas within the study of **VR** locomotion techniques will be analysed: Redirected Walking and Impossible or Flexible Spaces.

The next and final section of this chapter is about **PCG**. In this section, what is **PCG** and how it works is the first subject to be studied. Then, **PDG**, a sub-field of **PCG**, will be presented because the virtual layout to be created by the tool will use concepts and algorithms of this sub-field, so its study is important. To complement **PCG**, adaptive algorithms will be analysed, including how they can elevate **PCG**. Finally, some applications of **PCG** in **VR** environments will be studied.

2.1 Navigation in Virtual Reality

Virtual Reality (VR) is a simulated 3D environment that enables users to experience, explore and interact with a virtual environment in a way that approximates reality [13]. **VR** has the capability of creating a sense of immersion [72] in users, where they feel deeply engaged and absorbed in the **Virtual Environment (VE)** to the point where the boundary between the real world and the virtual world becomes blurred. The advances in **VR** technology in the last decade have provided the impetus for applying **VR** to different

engineering applications such as product design, modelling, shop floor controls, process simulation, manufacturing planning, training, testing and verification and much more [39].

In museums [52], for example, and as seen in Figure 2.1¹, VR can be used to create new opportunities for users to explore museums, such as reconstructing historical environments and creating interactive, engaging, and immersive experiences, among others. In the healthcare sector, VR can also provide a more effective rehabilitation intervention [58]. In the manufacturing industry, having access to a real-time model of a prototype in VR can help engineers anticipate possible flaws in the product. This facilitates solving problems before starting to manufacture the product and, in turn, saves time and money. It can also help present to the clients what the product will look like, and because it is in VR, the engineers can modify and make adjustments at any time.

While VR offers immersive experiences, it also presents limitations and challenges. A primary concern is the limited available physical space of users. Additionally, users can have cybersickness [46], a phenomenon akin to motion sickness, which can manifest during or after virtual immersion.

Despite these limitations and challenges, there has been a noticeable growth in the use and interaction with VR and Augmented Reality (AR), as highlighted by Cipresso et al. [13]. This trend, along with integrating both concepts, has captured the attention of investors and the general public. For instance, modern VR headsets can also function as AR headsets, enhancing user's everyday activities and productivity. This is the aim of VR systems: to develop new VR technologies into Interconnected Virtual Reality (IVR). Bastug et al. [4] envisioned a three-stage evolution for VR systems, depicted in Figure 2.2. Starting with current VR systems, it transitions into IVR and ultimately culminates in the realization of the ideal VR system. In their investigation, the researchers [4] delve into the potential integration of 5G technology with VR, exploring the prospect of linking VR systems to cloud services. This integration aims to leverage the benefits of cloud services, paving

¹VR- Where technology meets art <https://www.augray.com/blog/virtual-reality-where-technology-meets-art/>, Last Access: 2024



Figure 2.1: With VR, a user can explore a museum while at home and personalize this experience.

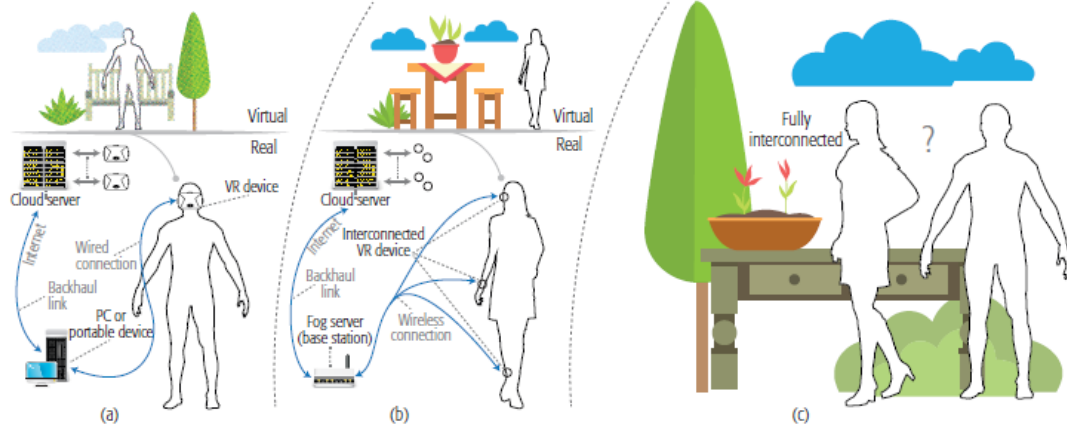


Figure 2.2: An illustration of evolution stages of VR: a) current VR systems; b) interconnected; c) ideal (fully interconnected) systems [4].

the way for wireless **IVR** experiences where multiple **VR** systems are interconnected, enabling collaborative and shared virtual environments. The study not only explores the possibilities but also confronts the challenges and obstacles inherent in adopting this cutting-edge technology.

While **IVR** is still an idea and has to overcome many challenges, current **VR** and **IVR** share something unique to **VE** technologies: users can be fully immersed when exploring these **VE**. Currently, the best form to achieve this immersion is through walking, and Lutfallah [37] states that real walking in **VE** can significantly increase immersion. However, achieving walking in large **VE** can be challenging because users usually have little physical space to play **VR** [50]. Still, there are a lot of techniques to make it possible for users to explore large virtual worlds, be it through walking or other methods.

2.1.1 Locomotion Techniques in Desktop Applications

Locomotion techniques, whether in desktop applications, **VR**, or even real life, allow users to navigate through their environments, explore the world around them, and interact with them. As Lochner and Gain [35] state, locomotion techniques in **VR** are how users traverse a **VE** and are considered an integral and indispensable part of user interaction. This is why locomotion techniques [12] are considered an essential form of human-computer interaction.

The way the users navigate will affect how they perceive the world. For example, walking and running are two different locomotion techniques from real life that permit humans to navigate their surroundings, and the two have different functions. One is to reach a location in the world faster, at the cost of getting tired more quickly, and the other is to be immersed in the world, having more time to appreciate what surrounds them, at the expense of being slower.

Different locomotion techniques provide users with different ways of navigating and experiencing their environment. Understanding how to choose these techniques is essential, as they shape the user's experience of the product, application, and environment

In desktop applications, the more common locomotion techniques are:

- **Mouse Pointer:** The mouse pointer is the primary input device for desktop applications, and with it, users can move the cursor to interact with icons, buttons, menus, and other on-screen elements. There is also the possibility of using hand gestures [65] as a virtual mouse, but this technology is still an ongoing research.
- **Keyboard:** In most video games, the keyboard is the primary locomotion technique, enabling the user to explore the virtual world. Another great utility of the keyboard is its shortcuts. These are beneficial because they allow users to interact quickly with the software without relying on the mouse. However, the downside is that in each application, there are a lot of shortcuts that the user has to remember. Recent alternatives in research include virtual keyboards and finger recognition [73].
- **Gamepad/Joystick:** Gamepads and joysticks provide users with alternative and tactile means of navigating and interacting with on-screen elements. These devices are particularly advantageous for gaming and other interactive applications.
- **Menus/Toolbars:** Menus provide a hierarchical structure for organizing application features and options. They also help the user access these features. These menus often include drop-down menus and toolbar menus for easy access to various functions.

These are some of the more common locomotion techniques in desktop applications. In VR, different methods have to be applied to enable immersive navigation while using the headset display.

2.1.2 Locomotion Techniques in VR

As is the case with desktop applications, VR also has distinct locomotion techniques to be used, and these even have different categories [35, 38], detailed in Figure 2.3. Some of the more popular techniques are:

- **Touchpad/Joystick:** In the steering-based category, touchpads and joystick are controller-based techniques in which users direct their movement based on input from a 2D-axial control. This technique is commonly used in scenarios with limited physical playing space, as it does not require the user to move. However, the disparity between visual and motor stimuli may lead to motion sickness [35].
- **Arm Swing:** Belongs in the walking-based category, is a technique where users simulate real-life walking by activating both controller's grips and swinging their

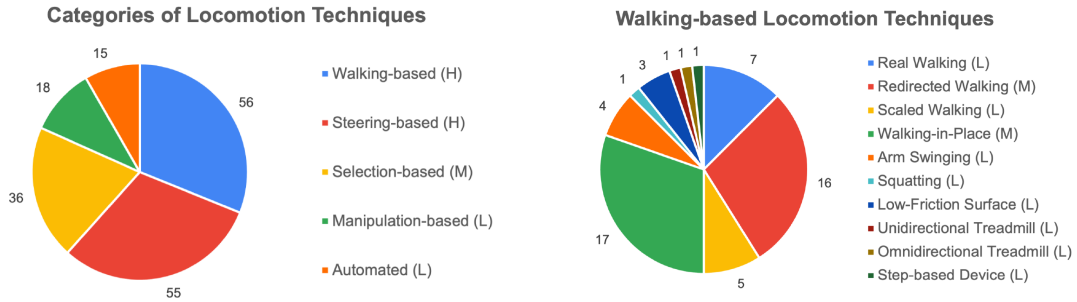


Figure 2.3: Different categories of locomotion techniques on the left, each with various techniques based on the category. On the right are the different walking-based techniques and if they have been highly (H), moderately (M), or less (L) explored [38].

arms back and forth, subsequently moving in the direction they are facing. This technique has been used in video games, such as Thief Simulator VR [38] and Sprint Vector.

- **Teleport:** Belongs in the selection-based category. It is one of the most explored techniques in this category. This technique is a teleportation-based technique where users point a ray, originating from their dominant hand, towards a position in the virtual world and instantaneously teleport there once the controller's trigger is pulled. The teleport is commonly used to traverse large virtual spaces and is suitable to be used in small physical play areas since it does not require the user to move [48].
- **Walking:** Belongs in the walking-based category, walking is a room scale-based technique whereby a user's physical motion is reflected in the VE, and it is proven to be the most immersive technique [35]. This technique is effective in terms of immersion; however, it requires a large physical area for free walking.

Each of these techniques has its benefits and limitations. The touchpad, arm swing, and teleportation enable navigation in the virtual world, even when physical space is limited. However, all of them represent forms of locomotion vastly different from human walking. In contrast, walking offers a more immersive form of locomotion but requires a spacious physical area, which VR users rarely have available. Because of this limitation, walking [35] remains the most challenging locomotion technique to implement and, as such, is one of the less applied locomotion techniques, as seen in Figure 2.3. One of the ways to circumvent this limitation of walking is the concept of Redirected Walking, which will be discussed next.

2.1.3 Redirected Walking

Redirected Walking [38] is a technique that circumvents the limitations of walking by redirecting users away from the boundaries of the tracking space, and it is one of the first techniques to address the limitations of walking in VE. Razzaque et al. [26] state that

Redirected Walking works by interactively and gradually rotating the virtual scene about the user, such that the user is made to continually walk in the **VE** while avoiding colliding with the limits of the physical space. With this technique, it is possible to make users believe they are walking a straight line while, in reality, they are not, as seen in [Figure 2.4](#). This is possible because the technique exploits the limitation of the human perceptual mechanism for sensing position, orientation and movement.

The use of Redirected Walking and, in turn, redirection techniques [\[56\]](#) is a promising solution that allows considerably large **VE** to be explored using natural body motions within a confined physical space. Suma et al. [\[56\]](#) present a novel taxonomy that separates redirection techniques according to their geometric flexibility versus the likelihood that users will notice them. In [Figure 2.5](#) it is possible to see how the redirected techniques are separated and for each cell a distinct implementation strategy.

Repositioning Techniques [\[10, 56\]](#) manipulate the correspondence between points in the real and virtual worlds to compress an ample virtual space.

- **Subtle Continuous Repositioning:** A continuous repositioning [\[43\]](#) can be applied subtly by applying translation gains to the user's physical locomotion, effectively scaling walking motions to cover greater distances in **VE**. If the gains applied are not small enough to avoid detection, this technique is no longer subtle, and the user starts to notice these changes.
- **Overt Continuous Repositioning:** A simple repositioning can be achieved by continuously translating the **VE** about the user's position, allowing the user to walk to areas not previously accessible within the limited physical space. If the translation to the **VE** is done unexpectedly, it may be disorientating and make the **VE** appear unstable.

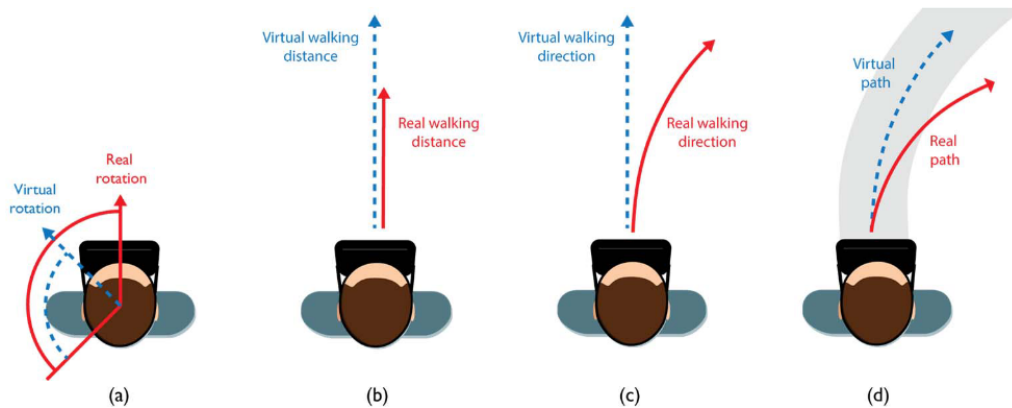


Figure 2.4: Four types of gains [\[43\]](#) used to manipulate user perception of **VE**. (a) rotation gains (user stationary), (b) translation gains, (c) curvature gains, and (d) bending gains.

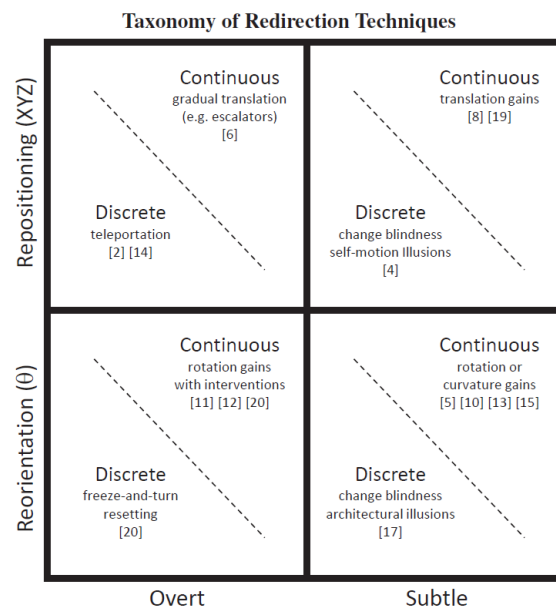


Figure 2.5: Taxonomy of redirection techniques [56] for supporting walking through immersive VE. The vertical axis distinguishes how the technique is applied in the environment. The horizontal axis provides a ranking in terms of noticeability to the user. The division in cells represents distinct implementation strategies for each type of technique.

- **Subtle Discrete Repositioning:** This technique uses visual optic flow effects to mask abrupt translations in the environment, and these small discrete updates can be repeated periodically as the user walks, allowing distances to be scaled similarly to continuous techniques.
- **Overt Discrete Repositioning:** This technique teleports users to new locations in the VE. However, this technique can be disorienting if the user does not expect the virtual position to be manipulated. To mitigate this problem, Bruder et al. [8] have leveraged the concept of portals from popular science fiction to provide an environmental grounding for teleportation.

Reorientation Techniques [10, 56] attempt to rotate the user’s heading away from the boundaries of the physical workspace.

- **Subtle Continuous Reorientation:** This technique introduces an imperceptible gain to the user's head rotation to guide the user away from the boundaries of the physical space. It can also, as an alternative, dynamically apply a continuous rotation as the user travels forward, resulting in a curvature of the walking path. As in Subtle Continuous Repositioning, users will start to notice if these gains exceed a threshold.
- **Overt Continuous Reorientation:** Resetting is a method of reorienting the user that requires intervention when the user reaches the boundaries of the physical space.

During the intervention, the user is instructed to turn around. During which a rotation gain is applied without the user noticing. This reset will break the user immersion so visual distractions can be used to elicit head turns and apply the rotations, mitigating the break in the user immersion.

- **Subtle Discrete Reorientation:** This technique instantaneously changes locations or orientation of architecture features in a virtual scene at runtime. This technique is based on **change blindness** [56, 64] paragraph, which is a phenomenon that can be observed when users fail to notice alterations to a visual scene that occur outside their visual field. Change blindness techniques are primarily limited to interior environments with manipulated doorways, limiting their applicability.
- **Overt Discrete Reorientation:** Resetting can also be applied discretely. In their implementation of freeze-and-turn resetting, Williams et al. [68] freeze motion tracking and instruct the user to rotate away from the boundaries of the physical space. After the user completes the rotation, the virtual view is unfrozen and motion tracking resumes. While this discontinuity will be obvious to the user, resetting remains useful as an emergency "failsafe" technique.

As was already stated, walking as a locomotion technique is a field that still has many challenges in allowing fully comfortable natural navigation. New research and techniques are making the choice of using walking as a locomotion technique more viable. But is it possible to combine this **Redirected Walking** techniques to achieve seamless locomotion? That is what Paulmann et al. [44] study tries to answer. Their goal was to create a locomotion concept for a VR adventure game named The Amusement² that had a seamless virtual experience and could make natural movements happen in a play space of only 2x2 meters, as it can be seen in Figure 2.6. This was only possible by combining the various natural locomotion techniques. From their evaluation, there were three categories that techniques could belong to: **redirection techniques**, which include Impossible Spaces, a concept that will be discussed further in the chapter, **passive movement techniques**, and **indirect movement techniques**.

Paulmann et al. [44] aimed to maximize presence and accessibility, so they needed to be creative when implementing these various techniques. As seen in Figure 2.7, it is possible to mask the use of more overt techniques with the context of the VR world, allowing for the user's immersion to be kept without sacrificing the use of techniques. An example from the Figure 2.7 is the use of ladders, which fall under the **Indirect Movement Techniques**. The world is translated around the ladder when the player climbs it. Another example is, in the game, there were times when the path forward was blocked by planks that the users had to remove and throw into a garbage bin. In this obstacle, every time the user rotated his head, a slight rotation gain was used, making it so that when the users were done

²The Amusement Game Website <https://curvaturegames.com/projects/the-amusement/>, Last Access: 2024



Figure 2.6: A screenshot of a user playing The Amusement game. On the left is the user view in the VE; on the right is the physical space available for the user to move [44].

with the task, they had turned 180 degrees, allowing them to continue without hitting a border of the physical space. This is an excellent example of using the **Overt Continuous Reorientation Technique**. Lastly, an example of **Passive Movement Techniques** is using moving platforms to rotate the user passively or reorient him to where physical space is available to continue moving.

By combining these various natural techniques and blending them or hiding them in the context of the game, Paulmann et al. [44] created this game that the user can fully explore and be immersed in without breaking the user's immersion in a physically limited space.

In short, redirection and reorientation techniques are a great way to overcome the limits of the physical space, but they are not the only ones capable of overcoming them. There are other techniques that, instead of manipulating the user perception of the virtual space, manipulate the virtual space in itself, which allows it to achieve the same objective without being an intrusive user redirection method. Vasylevska and Kaufmann [63] present in their paper an overview of existing approaches and techniques that enlarge a walkable virtual space, including this spatial manipulation technique, namely Impossible Spaces and Flexible Spaces.



Figure 2.7: A screenshot of a user playing The Amusement game presenting scenarios that use different locomotion techniques. The image on the left is a good example of the Passive Movement Technique(Platform), and the other images are various scenarios in which Indirect Movement was used [44].

2.1.4 Impossible and Flexible Spaces

Up to this point, it was discussed different locomotion techniques in [VE](#), such as teleporting, joysticks, and arm swings. Although these techniques [50] are practical for small/restricted physical spaces, they significantly deviate from natural human locomotion, disrupting immersion and presence in [VE](#), unlike walking, which closely mirrors real-world locomotion [56].

The biggest challenge with applying walking in [VE](#) is the requirement for a large physical area. To overcome this requirement, Redirected Walking techniques were introduced to present possible solutions. Although effective in preserving the user's feeling of moving naturally through [VE](#) while simultaneously keeping the user within the boundaries of the physical space, they are still somewhat limited in their practical applicability [57].

Paulmann et al. [44] show that combining these different concepts and techniques makes it possible to achieve seamless navigation in a [VE](#) while still maintaining the user within the confinements of the physical space. However, their work uses a method that augments the combined utility of the redirection techniques called **Impossible Spaces**.

Impossible Spaces [57] are [VR](#) environments that contain geometry that violates the rules of Euclidean space and, therefore, cannot exist in reality. By taking advantage of **Non-Euclidean Geometry**, Impossible Spaces [50] leverage the fact that real-world physical laws do not bind [VR](#). This allows for strategies to maximize available physical spaces, including self-overlapping architectural layouts.

Impossible Spaces represent an architectural illusion that maximizes the virtual area that users can explore through natural locomotions [16, 49]. Most importantly, the user's perception of the space remains consistent when exposed to these illusions. This makes it possible to reorient users away from the boundaries of their physical workspace, achieving a seamless [VR](#) experience.

As stated before, Impossible Spaces are less intrusive because they do not manipulate the user's perception of the [VE](#); they manipulate the [VE](#) itself. They also complement Redirected Walking techniques well. A simple example of an impossible [VE](#) can be seen in [Figure 2.8](#) where the space into adjacent rooms is overlapped to give the impression the rooms are much larger than they are.

Lochner and Gain [35] state that several games, such as *Spellbound Spire*, *Tea for God*, *traVRsal* and *Unseen Diplomacy*, have been developed using walking in Impossible Spaces as the dominant means of locomotion. Although these titles achieve the goal of self-overlapping architectures, they differ in their application; some are used with transition areas and others with portals.

Transitioning areas [64] are intermediate spaces, such as curved corridors, that allow invisible changes to be made to the user. These transitioning areas exploit a phenomenon already mentioned, **change blindness** [56, 64].

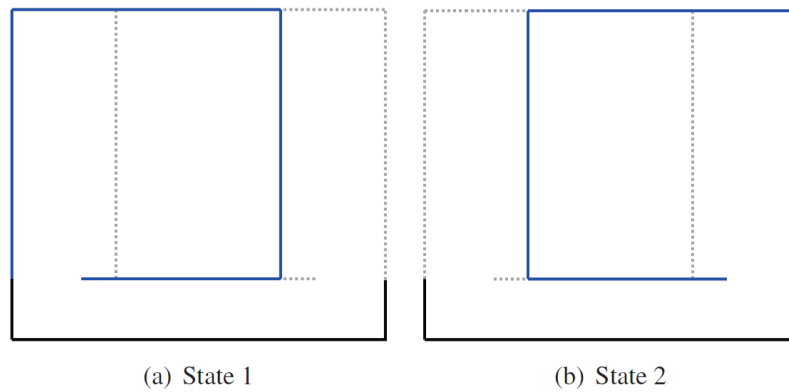


Figure 2.8: An example of an impossible virtual environment [57] consisting of a building with two adjacent rooms that spatially overlap. To ensure that the user's view is visually consistent, the environment shifts between two states, with only a single room visible at any one time.

Portals [35] are virtual "windows", usually doors, that teleport users to a different location when stepped through. These take advantage of **Overt Discrete Repositioning Technique** and mitigate this technique's disorientation problem because the user expects a relocation in the environment when entering the portal. Portals are a concept in 3D video games well established and showed an initial surge in popularity in 2007 after the release of the game Portal³, in Figure 2.9 it is possible to see how portals work in the game.

An example of the applicability of portals in Impossible Spaces can be seen in the experiment conducted by Rebelo and Nóbrega [50]. In this study, they compared exploring a VE using **Teleportation** or **Walking using overlapping spaces and stride adjustment**. In their walking approach, they used Impossible Spaces connected by portals, as seen in Figure 2.10. The experiment involved the users exploring a house and completing various tasks. Half of the users used teleportation as locomotion, and others used walking

³Portal Game on Steam <https://store.steampowered.com/app/400/Portal/>, Last Access: 2024



Figure 2.9: Example of how a portal works in the game Portal, where the blue ellipse connects to the orange ellipse, allowing the player to be teleported from one ellipse to the other when entered.

as locomotion. The authors concluded that both techniques were equally satisfactory regarding task completion time, sense of presence, cybersickness symptoms, user experience and naturalness, spatial awareness and orientation. However, the walking technique stood out as significantly more natural and intuitive. The participant's preferences further supported the effectiveness of the walking technique, with 70% preferring walking over teleportation. This was due to users commenting that walking emphasized a heightened sense of immersion, facilitating memorization of the VE layout and object position.

This research and many more, such as Lochner and Gain [35], Suma et al. [57], and Fisher et al. [16], strongly advocate adopting walking as the preferred locomotion technique in VR. Furthermore, by employing and combining Impossible Spaces and other redirection techniques, a seamless walk in VR can be achieved. This achievement is reinforced by several video games, including The Amusement [44], Tea For God⁴, among others.

Flexible Spaces [64] are a redirection technique that allows walking in VR without replicating real-world layouts, providing an expansive experience and exploration. Flexible Spaces overcome these limitations, generalize overlapping spaces, and address change blindness [56, 64] using procedural layout generation. Furthermore, Vasylevska et al. [64] define a Flexible Space as a dynamic self-overlapping layout that executes automatic relocation and restructuring of environment parts to fit the physically limited space.

⁴Tea For God <https://void-room.itch.io/tea-for-god> <https://void-room.itch.io/tea-for-god>, Last Access 2024

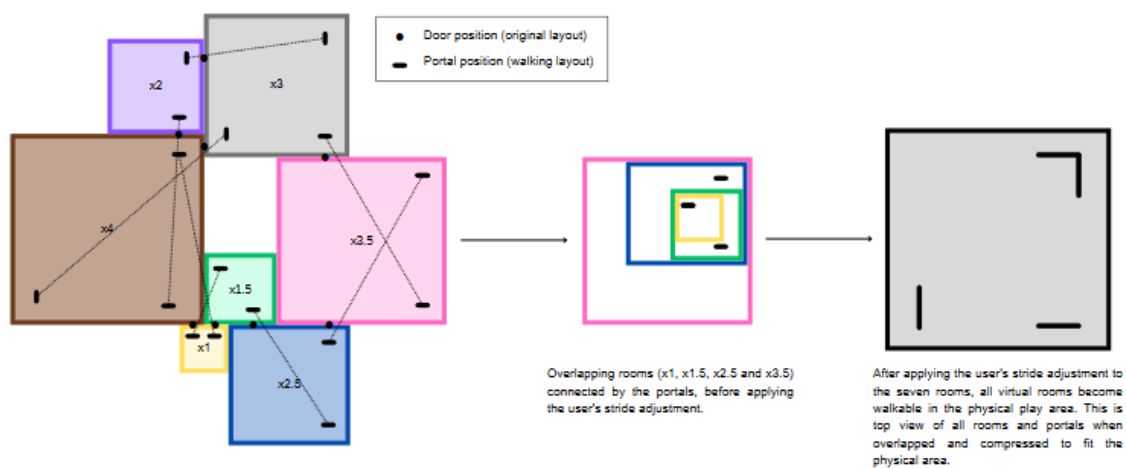


Figure 2.10: Rebello and Nóbrega [50] VE illustration. The left image represents all the possible rooms that the user can explore (the yellow room has the same size as the physical available space), the location of the doors (original layout), and the portal positions (walking layout). In the middle image, the rooms overlap. It is possible to see where the portals will be. In the right image, the user's stride adjustments are applied to all the rooms, making them all walkable.

The use of procedural layout generation, a topic which will be explored in more detail in the **Procedural Content Generation** section 2.2, allows a Flexible Space to be dynamic and change the layout of the **VE** following the state of the environment and the user. Still, this algorithm can take a bit of time to compute and to compensate for this, Flexible Spaces take advantage of the change blindness [56, 64] techniques.

In Figure 2.11, it is possible to see Vasylevska et al. [64] proof-of-concept example on Flexible Spaces in a large-scale environment. The figure shows two rooms, where the user navigates from one room to another following one route and eventually comes back with a different path. It is possible to see that overlapping spaces were sometimes used and that the **VR** environment context limits this technique. Flexible Spaces work best when the **VE** emulates, for example, a museum with many rooms and corridors connecting them. This is not to say that Flexible Spaces cannot be used in outdoor settings, but the context in which they are applied matters and influences how the technique is applied.

Flexible Spaces are another way to maintain seamless navigation through **VR**. Importantly, as highlighted by Vasylevska et al. [63] in their research, Flexible Spaces represent a crucial advancement in making spatial manipulation techniques for real walking in **VR** more practical and applicable across diverse application domains. Although Flexible Spaces are a promising solution, more research is needed in order to adapt the solution to all types of spaces, including small ones.

It is up to the developers to choose which techniques better suit their vision for the **VR** environment. Do they want endless exploration without the user worrying about the layout, combine Flexible Spaces with Impossible Spaces, and some repositioning techniques. Or maybe they want to have a **VE** where the user can explore a specific place and be immersed, then combine the various Redirected Walking techniques and Impossible Spaces. There are a lot of possibilities, all depending on the user's needs, the system requirements, the context of the scenario, and the developer's vision.

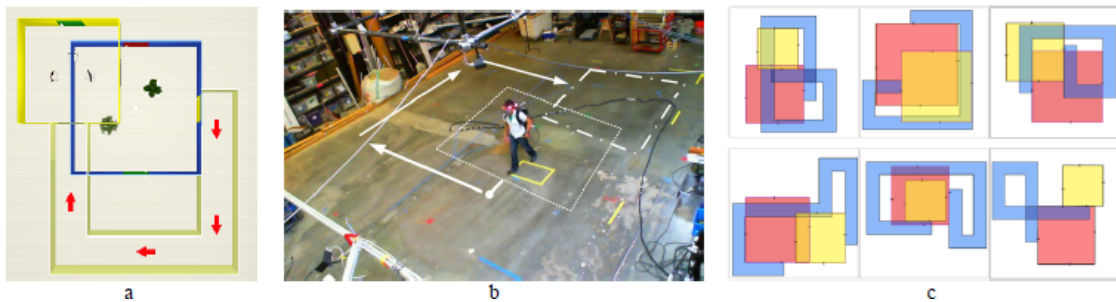


Figure 2.11: The Flexible Spaces [64] algorithm generates the layout dynamically but maintains connections. a) A simple layout in Flexible Spaces. The corridor between the blue and yellow rooms is rerouted whenever the user leaves. b) User navigates in **VE** by walking through the corridors from one room to another while staying within tracked space. c) Possible layouts generated with the algorithm of Flexible Spaces.

2.2 Procedural Content Generation

Creating large virtual worlds that can be explored in VR while in a restricted space is challenging. Even more so if these worlds are to be explored using walking as a locomotion technique. When done manually, these virtual worlds can be time-consuming to create because of their size and the implementation of techniques to allow walking to be used. The benefits of using **Procedural Content Generation** to enhance and accelerate this process arise. Flexible Spaces are one example of a technique that has shown these benefits. Hence, understanding PCG is important to the work of this dissertation.

Procedural Content Generation (PCG) [31, 47] is creating data algorithmically by computers instead of manually. PCG [21] can help designers create content and enhance the creativity of individual creators with little intervention from the designer. For this reason, PCG [15] is widely used in many fields of computing, such as textures, models, simulations, movies, video games and others.

In video games, Togelius et al. [60] state that the use of PCG has been a part of publicising games since the early eighties. It has helped creators provide variety to the games, reducing the development time and cost and, most importantly, augmenting human creativity and enabling adaptivity in games. Gellel and Sweetser [22] further reinforce the benefits of PCG by stating that the use of PCG methods in video games is easy to use, consistent and a reliable way of creating infinite fresh and varied content that can be replayable and enjoyable.

There are a lot of definitions of what is PCG, Freiknecht and Effelsberg [20] in their research study and present some of them. They also came up with their own unified definition of PCG by stating that PCG is the automatic creation of digital assets for games, simulations or movies based on predefined algorithms and patterns requiring minimal user input. In Figure 2.12, it is possible to see two results of PCG applications in games.

With such a wide range of applications, there was a need to differentiate the fields of use of PCG. For this reason, Hendrikx et al. [23] introduced the abbreviation **Procedural Content Generation for Games (PCG-G)** to separate PCG for games and other fields. For the scope of this thesis, PCG-G is the most relevant, but having an understanding of how PCG works is essential.

The main way PCG generates content is by employing algorithms, which can vary massively from one another. Some are better for generating natural terrain, others are for generating dungeons or labyrinths, and many more. Although there are a lot of algorithms [27], all of them share a key property: the developer is able to describe what he wants to generate. It can be, for instance, related to geometry, texture, effects, etc. Not only that, but the developer can also give instructions on how the algorithm should generate the content. These instructions can then be called to generate instances of the asset with varying characteristics by parametrising the description. In Figure 2.13 is an example of different PCG algorithms generating different textures.

In more concrete terms, every PCG needs input parameters (the descriptions) that are



Figure 2.12: On the left is the game Minecraft and its procedurally generated terrain, on the right is the world generation of the game No Man's Sky.

defined and adjusted. These parameters specify the algorithm's behaviour and dictate what and how the content is generated. **PCG** also needs constraints [53] imposed in the generation process to satisfy the explicit designer's intent and ensure that the content created remains plausible and consistent.

More often than not, simply employing an algorithm or technique is not enough to realize the designer's vision, and this, according to Togelius et al. [60], is the real challenge of **PCG**. This challenge is the creation of content generators that can generate purposeful, coherent, original and creative content. There are many ways of approaching this challenge, and the most common one is combining different algorithms and techniques to create more sophisticated and complex **PCG** algorithms.

2.2.1 Generating Connected Spaces

Procedural Dungeon Generation (PDG), as the name implies, is the creation of dungeons using **PCG** techniques. A dungeon [66] is a maze-like environment that is mostly composed of three game features: rewards, challenges and puzzles. A dungeon is usually a set of interconnected rooms that have a starting room and an ending room. These rooms can vary in shape and size, and there is usually more than one way of traversing a dungeon.

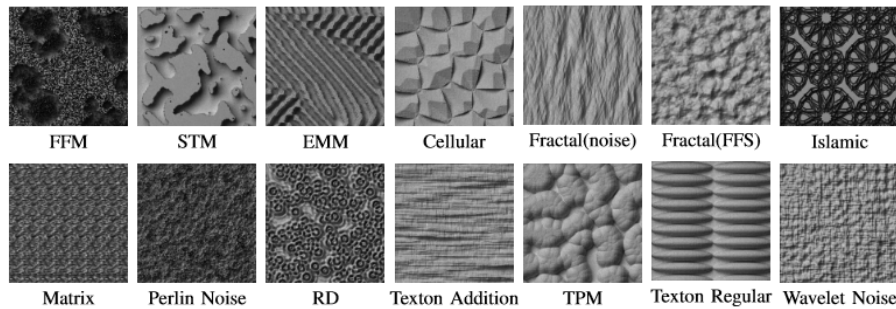


Figure 2.13: Cai et al. [9] sample textures that use different **PCG** algorithms to generate them. This image shows the versatility of **PCG** algorithms and how they differ from one another.

In Figure 2.14 are two examples of dungeons that were procedurally generated. Linden et al. [33] state that dungeons are unique due to the close bond between designing gameplay and game space and that dungeons incentivise exploration while maintaining strict control over gameplay experience, progression and pacing.

Given these unique features of dungeons, PDG helps in the creation of levels and maps for game content, and these elements are equally essential as the rules of the game and are the main reason that incentivises users to interact with the environment [71]. The essence of PCG-G, as facilitated by PDG [21], lies in its capability to generate various levels and maps within a consistent game mechanic or world vision. This ability to diversify the game content and experience is fundamental to altering gameplay and enhancing the player's experience. This core objective represents what PCG-G genuinely strives to accomplish, and it is the driving force behind the widespread adoption of PCG in diverse applications over the years, ranging from games such as Rogue and Minecraft to various non-gaming domains.

As stated before, PCG is a vast concept with many fields; there are fields only for simulations, textures, games and others, and inside each field, there are even more sub-fields that compose the whole. In PCG-G, for example, there are fields for terrain modelling, animations, game levels, world, and many more. Within this wide range of options, it is important that the field chosen contributes and has significant value to what is being created. When it was stated that only PCG-G was going to be studied, it was with these values in mind, and now, narrowing even more the scope and talking about PDG is still with these values in mind.

Having control over how the world is generated and how it is meant to be explored is key to the work of this thesis. PDG is the most relevant field in relation to this goal, so it is important to present some of the techniques and algorithms that are used to generate these worlds.

As mentioned earlier, a dungeon typically comprises interconnected rooms and hallways, forming the fundamental topology and geometry. While the context of a dungeon

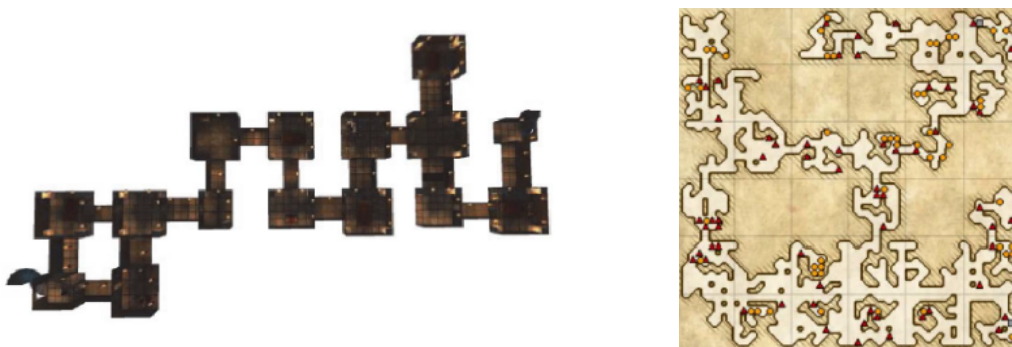


Figure 2.14: On the left is the example of Linden, Lopes, and Bidarra [33] dungeon generation layout, on the right, is Liapis [30] example.

may vary, such as being a labyrinth, cave, etc., these elements constitute the basic structure. PDG refers to the generation of this topology, and a typical dungeon generation method consists of three elements [33]:

- Representation model – an abstract and simplified representation of a dungeon and a simple overview of the final structure.
- A method for constructing the representation model.
- A method for creating the actual geometry of the dungeon based on the representation model.

While not all methods need to have these three elements, it is important that the dungeon-generated geometry and content follow gameplay requirements. This is because, according to Linden et al. [32], gameplay naturally determines which unique content is required, whereas content can be ambiguous regarding which gameplay it sustains.

Enhancing content to align more effectively with gameplay necessitates increased control over procedural methods [33]. Control is a pivotal aspect of any method, encompassing the options provided to a designer to guide the level-generation process toward the desired outcome intentionally. It ensures that generated content remains consistent while retaining the desired properties and variability. Insufficient parameters or a lack of understanding of their functions typically result in limited control over a PCG algorithm, leading to undesired outcomes.

Viana and Santos [66] state that there are two groups to which PCG algorithms can belong, these being constructive or search-based algorithms. Togelius et al. [61] further expands these groups, as illustrated in Figure 2.15, by the addition of Generate and Test algorithms.

- **Constructive algorithms** generate the same content type while being quite different from one another. Most are based on random positioning and/or algorithms created for other purposes, such as cellular automata and generative grammars.
- **Generate and Test algorithms** (G&T) are similar to constructive algorithms. Still, while constructive generates the content once and has to ensure the content generated is good enough, G&T incorporate both a generate and test mechanism that generates the content and then tests the content according to some criteria. If the test fails, all or some of the content generated is discarded and regenerated.
- **Search-based algorithms**, equal to G&T algorithms, always test the generated content to ensure consistency. They do this by keeping track of the generated population of content, evaluating each one according to some quality function instead of accepting or rejecting the content as in G&T, and then choosing the one with the highest score. Most Search-based algorithms are based on Genetic Algorithms.

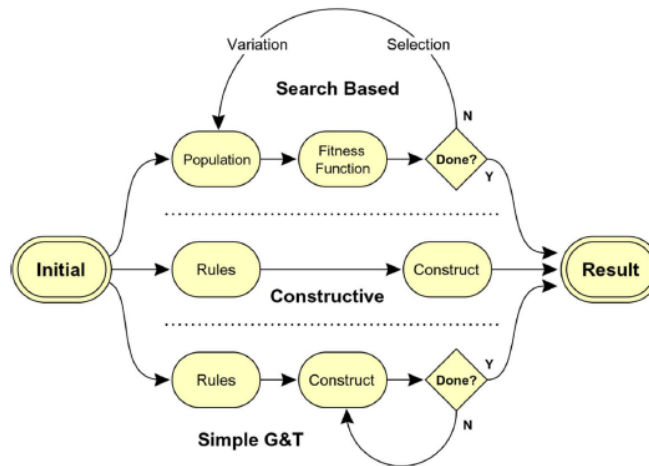


Figure 2.15: Three approaches, according with Togelius et al. [61], to PCG: constructive, simple generate and test and search based.

Cellular Automata [33] is a technique that is a self-organizing structure that consists of a grid of cells in any finite number of dimensions, usually 2D. Each cell refers to a set of cells that comprise its neighbourhood and an initial state. This state is then recursively calculated again in the next generations, with a set of rules being applied in each generation. After multiple generations, a pattern starts to emerge in the grid, which depends greatly on the rules and cell states. These states [14] can denote a solid tile or an empty tile, for example, with the values of 1 or 0 to represent them.

This technique has various applications in PCG and is quite popular among game developers that use PCG. In Figure 2.16, the algorithm was used to generate a "cave-level"-like structure. Another example is in Temuçin et al. [59] Procedural Generation of Organic Cities, where cellular automata generated clusters of areas, creating an organic city layout with fractal properties. Adam et al. [1] also show a different use in Maze Level Generation, where cellular automata and genetic algorithms were used to create playable mazes.

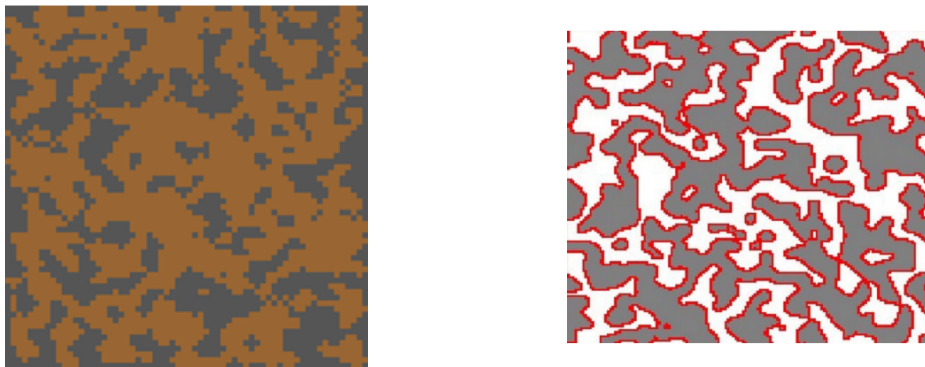


Figure 2.16: Example of Cellular Automata output representing caves layouts.

Voronoi Tessellation [15] or Voronoi Diagrams is a mathematical technique that represents the division of 2D space into regions. These regions (Voronoi cells) are based on the distance to points in a specific subset of the plane. In Figure 2.17, it is possible to see an example of the output of the algorithm.

A **Delaunay triangulation [55]** for a set of points in a plane is a triangulation such that no point is inside the circumcircle of any triangle. The centers of the circumcircles of the triangles in the Delaunay triangulation correspond to the vertex of each Voronoi cell of the Voronoi diagram [51]. This means that the points connected in the triangulation have their Voronoi cell adjoining, making the Delaunay triangulation the connectivity graph of the Voronoi cells.

Voronoi Tessellation is an effective tool for creating natural-looking plane divisions. For that reason, this technique is commonly used in areas such as PCG, where it is used in terrain generation to create, for example, mountainous landscapes. It can also be used to generate biomes of a map or even be used to generate dungeons⁵ and structures in a game world.

Genetic Algorithms [66] are metaheuristic algorithms based on the neo-Darwinian theory of evolution and are a sub-class of evolutionary algorithms. Although this is the case, Togelius et al. [62] decided to refer to solutions that use evolutionary algorithms as “search-based” due to it being a more value-neutral term and allowing all forms of metaheuristics and stochastic search/optimization algorithms. These algorithms, as stated in the search-based algorithms, generate content and then evaluate it based on criteria or parameters. Based on this evaluation, the algorithm generates new candidates by relying on biologically inspired operators such as mutation, crossover and selection to produce new content with higher value than the old one. In Pech et al. [45] study, they use cellular automata rules to generate the maze-like dungeon levels. Then, a genetic algorithm evolved the cellular automata rules to satisfy some level’s constraint.

⁵Procedural Map Generation with Voronoi Tessellation Github <https://github.com/mjvar/procedural-map-gen><https://github.com/mjvar/procedural-map-gen>, Last Access 2023

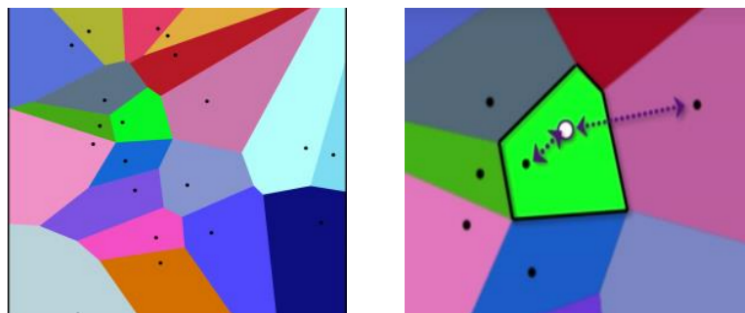


Figure 2.17: Example of the division Voronoi Tessellation does to a plane [55]. Every point inside the same region is closest to their seed (black dots) and no other.

2.2.2 Adaptative Algorithm

One of the problems with PCCG and content generated by computers is that the content tends to be static and rigid. This, according to Lopes and Bidarra [36], has a prescribed nature that can lead to predictable and impersonal experiences while alienating unconventional users. If the objective is to generate large virtual worlds, if it starts to be repetitive, the users will get bored and see the experience and exploration as more of a task, diminishing the enjoyment in the virtual world. To avoid this scenario, Huber et al. [24] states that to keep users motivated over long periods, games often employ Dynamic Difficulty Adjustments (DDA) to adapt the game's challenge, in this case, the virtual world challenge, to the user's capabilities. This DDA is what turns a static and rigid virtual world into a dynamic and adaptative virtual world, and it is going to be this adaptability that is going to make the virtual world more challenging and appealing [36]. In Figure 2.18 is an Architectural principle of game adaptivity presented by Lopes and Bidarra.

To achieve this adaptability in the virtual world, it is necessary that along the PCCG to generate the content of the virtual world, there needs to exist an adaptative AI⁶ system that can rearrange and adjust the world following the user interactions with the world. But how can the virtual world adjust to each unique user that explores the virtual world? Bakkes et al. [3] propose an approach to adaptative game AI that aims to adapt the game circumstances rapidly and reliably. With this approach, the system can adjust the virtual world on the fly to the user's current state. All the domain knowledge required to make this adaptation to the world is gathered by the system AI and exploited immediately. This adaptive AI system is what introduces uncertainty into the virtual world [74], and it

⁶In this case, Artificial intelligence does not refer to machine learning algorithms, but an intelligent system that will have criteria to adapt the virtual world accordingly

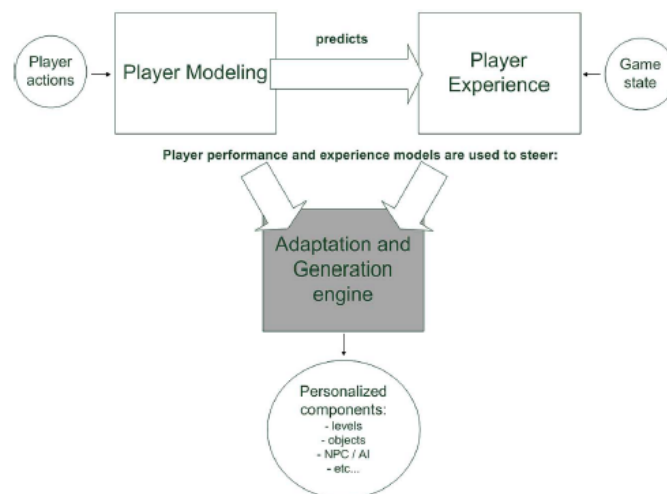


Figure 2.18: Architectural principles of game adaptivity according to Lopes and Bidarra [36].

is this unpredictability that will deliver enjoyment and excitement to users during their exploration of the VE.

If done well, this adaptability in the virtual world will provide what is called a **State of Flow** in the user. A State of Flow [25] describes a state of optimal experience whereby an individual is completely immersed in an activity. At that moment, everything seems to come together for the user, or in other words, it is when the virtual world can balance the challenges it gives to the users according to their skills and experience [24]. Figure 2.19 demonstrates well how the state of flow is achieved. The game Left for Dead⁷ and its AI Director is an excellent example of an adaptive algorithm that implements all these ideas and concepts and achieves a state of flow in the users.

Figure 2.20 provides a screenshot of the game's atmosphere. Designed to encourage replayability, the game uses PCG. To ensure that players can revisit the same sections without losing interest, PCG generates unpredictable experiences. In the game, PCG dynamically positions enemies and items, preventing the game from becoming a static memorization challenge [6]. This aspect is particularly crucial for cooperative gameplay, where expecting players to memorize static locations can diminish the overall experience. The game's AI system [18], known as the AI Director, plays a pivotal role in managing pacing and enhancing replayability. By monitoring the player's state, the AI Director dynamically adjusts the number of enemies in the game to match a dramatic intensity scale. This dynamic control creates peaks and valleys of intensity, crafting a dramatic experience on a moment-to-moment basis throughout the game. As shown in Figure 2.21, the AI Director in Left 4 Dead utilizes the player's state [6] to adjust the enemy population dynamically, employing four distinct states. For enemy placement, the AI Director divides the level into "areas". Enemies are strategically spawned in areas relative to the players, maintaining an optimal distance. The AI Director can discern areas in front of or behind

⁷Left For Dead Website <https://store.steampowered.com/app/500/Left4Dead/>
<https://store.steampowered.com/app/500/Left4Dead/>, Last Access 2024

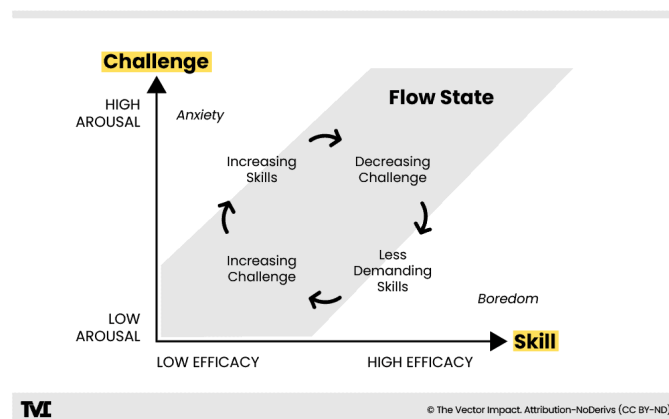


Figure 2.19: A graphic demonstrating the state of flow.



Figure 2.20: Example of the game Left for Dead 2

the survivors, enabling the creation of encounters such as ambushes or blockades. Left 4 Dead's pacing incorporates alternating periods of quiet tension and unpredictable, intense combat. This element of unpredictability enhances the game's replayability, ensuring players derive maximum satisfaction from the gaming experience, free from monotony and overwhelm [74].

In summary, incorporating an adaptive algorithm for virtual world generation enables the creation of dynamic and unpredictable experiences. This approach will also allow users to make real-time changes to the virtual world, enhancing adaptability and elevating the overall user experience, fostering a desire for exploration within these VE.

2.2.3 Applications in VR

PCG excels in crafting large, rule-driven, and randomly generated worlds, providing significant time-saving benefits for developers. Its utility also extends to VR with distinctive advantages. Exploring the integration of concepts such as Impossible Spaces, Flexible Spaces, and user perception manipulation with PCG becomes essential for generating VE tailored for exploration within constrained physical spaces.

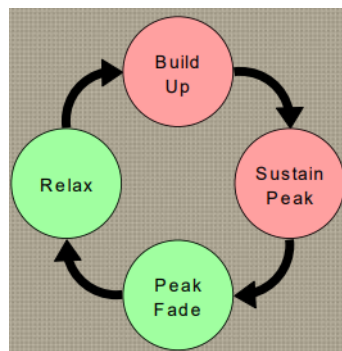


Figure 2.21: The game AI is constantly swapping between states depending on the status of the players, this way making the experience unique to each play session and making the world adapt to the players, giving more challenges when they are relaxed, and after hard challenges giving the players time to relax

Two notable studies that utilize **PCG** in conjunction with **VR** stand out, each showcasing unique applications of the technique. The first study conducted by Sra et al. [54] introduces a novel system that automatically crafts an immersive and interactive virtual environment by employing the real world as a template. This system, shown in Figure 2.22, allows users to pre-map their surroundings in 3D and, employing **PCG** techniques, generates diverse **VR** environments for exploration. Users can experience these environments through walking as a means of locomotion and have the option to interact with real-world objects, enabling the creation of personalized **VR** experiences. Although the system efficiently utilizes available space by scanning and mapping the environment before creation, it requires substantial pre-work before users can immerse themselves in the **VE**. Moreover, it does not address limitations faced by users without access to extensive physical areas or the necessary equipment for real-world mapping, restricting their exploration and interaction within the system.

The second study, conducted by Cheng et al. [11], presents VRoamer. This system, shown in Figure 2.23, allows users to traverse unexplored physical spaces, where VRoamer dynamically generates a virtual space on the go. Unlike the previous study, which requires pre-mapping of the 3D real space before exploring the **VE**, VRoamer identifies walkable areas and detects obstacles in real-time. It then creates virtual rooms if their size aligns with physically walkable areas or generates virtual corridors and doors leading to undiscovered physical spaces.

VRoamer shares the same problems as the first system; the user still needs to have a large physical area and extra equipment necessary for the system to explore. In both examples, the systems analyze the surroundings of the user and, with that data, create the **VR** environment using **PCG**. But what if the **PCG** was the one used to generate the layout of the virtual world, and the only information available to the algorithm was the limits of the physical area? This way, the users would not need any additional equipment or to have a large physical area to have the opportunity to explore a large **VE**.

A recently released game, Tea for God ⁸ is able to do just that by cleverly combining

⁸Tea For God Website <https://void-room.itch.io/tea-for-god> <https://void-room.itch.io/tea-for-god>, Last Access 2024

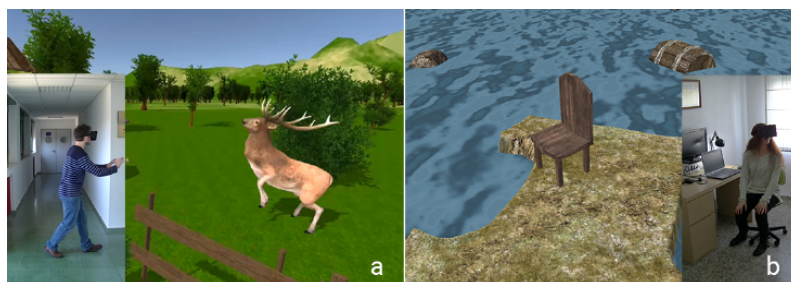


Figure 2.22: Illustration of Sra et al. [54] work showing the physical space and the virtual space reconstructed from the 3D mapping of the physical work.

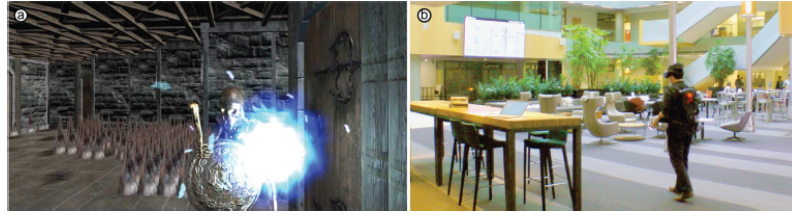


Figure 2.23: Illustration of Cheng et al. [11] VRoamer where it shows on the right the real world and on the left what the user is seeing.

the use of Impossible Spaces, corridors and PCG to deliver a unique experience to the players, where they are able to play the game, even when the available physical space is limited.

There is also research that can augment the use of PCG algorithms in VR. This is the case of Wang et al. [67] that proposed two **automatic virtual portal placement** methods for efficient VR navigation and also a real-time portal orientation determination algorithm to naturally guide users to the centre of the physical space. Another interesting concept is **Redirected Spaces**, introduced by Langbehn et al. [29], and belonging to the **Flexible Space** field, focus on using curved corridors that benefit from **Redirected Walking** techniques, as seen in Figure 2.24.

In summary, achieving seamless locomotion in VR through walking is challenging and suggests a combination of the diverse talked about walking techniques, along with the integration of PCG. This synergy is what will enable the generation of large virtual worlds that can adapt to limited physical spaces.

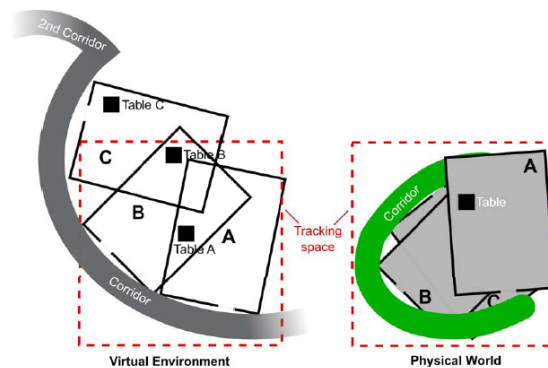


Figure 2.24: An illustration of Langbehn, Lubos, and Steinicke [29] combination of Redirected Walking and Impossible Spaces. The VE comprises a curved corridor and rooms A, B, and C. When applying a redirection gain at the corridor, the VE is compressed even more, and the whole setup fits into the tracking space.

2.3 Summary

This chapter explores the diverse applications of [VR](#), its navigation techniques, and its capability of immersing users in virtual worlds beyond physical limitations. It highlights the importance of locomotion in VR and distinguishes it from traditional desktop applications that rely on keyboards and mouse. In [VR](#), techniques such as teleportation, arm swing, controllers, and walking enable users to move through [VE](#). Walking, which mimics real-world movement, enhances presence and immersion, but its adoption is limited by the user's available physical space. The chapter examines how locomotion techniques, from Redirected Walking to virtual space manipulation, have evolved to address these constraints.

The second section introduces [PCG](#), a key concept for creating expansive, dynamic virtual worlds. [PCG](#) allows developers to generate varied layouts by adjusting parameters, supporting techniques such as Redirected Walking and Impossible Spaces. More details on its use for virtual layout generation will be discussed in [Section 4.1](#).

The thesis focuses on using portals, corridors, and other techniques to connect layouts while preserving limited physical space. The virtual world's context, provided by the developer, influences user perception. For example, exploring a museum versus an apartment offers distinct emotions based on how the rooms are connected through portals, corridors, or other techniques, as discussed in [Section 3.4](#).

Adaptive algorithms are also introduced to regulate user interaction, ensuring a seamless experience by adjusting the virtual world to the user's state in both the virtual and physical spaces. These algorithms will be further explored in [Subsection 3.5.1](#). An overview of the techniques from the research papers covered in this chapter is provided in [Table 2.1](#). The techniques used in the proposed tool outlined in [Chapter 3](#) and implemented in [Chapter 4](#), correspond to those in the table's fourth row.

Table 2.1: Overview of the different techniques used in the research papers.

	Techniques	Walking Techniques	World Generation	Reference
1	Redirected Walking	Impossible Spaces; Rotation Gains	Manual	[44]
2	Redirected Walking	Impossible Spaces	Manual	[50, 56, 57, 35, 16, 67]
3	Redirected Walking	Impossible Spaces; Flexible Spaces	PCG Algorithm	[63, 64]
4	Redirected Walking	Impossible Spaces; Redirected Spaces	Manual	[29]
5	3D Mapping	Walking	Genetic Algorithm	[54]
6	3D Mapping	Walking	PCG Algorithm	VRoamer [11]
7	Redirected Walking	Impossible Spaces	PCG Algorithm	Tea For God

ANALYSIS AND SYSTEM ARCHITECTURE

This chapter establishes the groundwork for addressing the Research Questions outlined in [Section 1.2](#) by detailing the proposed tool architecture. The tool enables users to navigate physically constrained spaces in [Virtual Reality \(VR\)](#) using walking as the locomotion method. the tool must adapt virtual rooms to the available physical space to allow for exploration and seamlessly connect these rooms, ensuring smooth transitions.

The tool’s requirements are outlined in [Section 3.1](#), and an overview of its capabilities and architecture is provided in [Section 3.2](#). [Section 3.3](#) discusses how [Procedural Content Generation \(PCG\)](#) generates virtual layouts with multiple rooms and adapts them for exploration in [Virtual Reality \(VR\)](#). The theory behind the implemented room connection techniques — Portals, Corridors, and [Central Hub \(CH\)](#) — is covered in [Section 3.4](#), along with how a scoring system manages these techniques. Finally, [Section 3.5](#) introduces key concepts related to the tool’s architecture, focusing on how it can import/export layouts and how adaptive algorithms enhance the tool’s flexibility.

3.1 Requirements and Methodology

With the analysis of related work on the topics of [VR](#) and [PCG](#), and keeping the thesis objectives and research questions defined in [section 1.2](#) in mind, an early conceptualization of the tool was developed and will be presented in this chapter.

In this phase, the overall design and architecture of the tool’s systems and capabilities were outlined, with the primary goal of adapting [Virtual Environment \(VE\)](#) for small physical areas. A virtual apartment visit was developed using an early manual implementation of portals, corridors, and a central hub for this initial proposed solution. This solution is also built on previous work developed inside the research affiliation group, incorporating a VR setup for exploring a virtual apartment through walking, portals to connect rooms, and stride adjustments. Initial interviews with Human-Computer Interaction (HCI) experts were conducted to gather feedback and insights that helped strengthen and refine the proposed solution. This early work led to the submission and publication of a Late-Breaking Work to the CHI 2024 (A*) [49], which presented the proposed solution,

including the interviews to validate it.

Following the tool's conceptualization and the outlining of tasks to achieve it, an iterative development approach was adopted. The process began with the development of virtual layout generation algorithms, designed to divide VE into smaller spaces and make them explorable in VR, even within small physical areas. Next, three room connection techniques were conceptualized – Portals, Corridors, and Central Hubs – incorporating lessons from the initial work, feedback from the expert interviews, and research into state of the art. After conceptualization, the techniques were implemented and tested. Through an iterative process, each technique was refined to achieve seamless walking in confined physical spaces in VR, aiming to answer the research questions.

A user study was conducted to validate the tool and assess the effectiveness of the implemented techniques, a user study was conducted. This study evaluated how well the tool met the research objectives and provided feedback on user experience. This feedback not only helped refine the tool's implementation but also contributed to the development of adaptive systems that enable dynamic, real-time customization of VE based on user preferences and automatically adjustments to the VE to enhance navigation.

Finally, a load test was conducted to assess the tool's capability to generate virtual layouts and identify its limitations.

User Requirements Throughout development, and with the research questions in mind, three types of users were identified:

- **World Builder Users:** These users have access to a more advanced version of the tool that allows for greater control over the creation and customization of virtual layouts. They must have a strong understanding of the tool to fully utilize its features.
- **VR Users:** These users can explore virtual layouts generated by the tool and customize them at runtime through a customization menu. Basic knowledge of navigating the menu is required.
- **End Users:** These users simply explore and navigate pre-generated virtual layouts without access to the customization options.

System Requirements The primary requirements for the tool include a VR headset, such as the HTC Vive Cosmos, and a computer capable of running VR applications. World Builder users should have access to a high-performance computer to support the more complex version of the tool. Section 5.3 will detail more about the necessary computer components to run the tool. Another essential requirement is physical space for movement during the VR experience. The recommended dimensions for this space are approximately 2.4 x 2.2 meters [34, 2].

3.2 Architecture

An overview of the tool's architecture is presented in [Figure 3.1](#), highlighting its three core components: input, processing, and output. The tool can take various inputs, such as input parameters that guide how the virtual layout should be generated or provided layouts that need to be fitted to a physical space. These inputs pass through a user interface that connects with the Unity application, where the tool's core is implemented. The Unity application relies on libraries such as OpenVR to enable VR support and integration. Within Unity, the tool utilizes the layout fitting component, which employs PCG algorithms to generate virtual layouts. The tool also integrates three room connection techniques — Portals, Corridors, and Central Hubs — to create virtual layouts that can be navigated using walking within small physical spaces in VR. During runtime, the tool's adaptive systems allow for real-time modifications to the virtual layout based on the user's needs and preferences. Once the virtual layout is created, users can explore the virtual world using a VR headset, and they have the option to request modifications via the adaptive systems while immersed in the VE. Some potential applications of this tool include creating virtual museum experiences, virtual apartment tours, or exporting layouts for use as input in other scenarios.

A deeper understanding of how the tool adapts virtual layouts to constrained physical spaces is illustrated in [Figure 3.2](#), which outlines the process. In step (A), the user sets the parameters. If a virtual layout is provided, step (B) — generating the layout — is skipped. Otherwise, step (B) generates a virtual layout and rooms. The tool operates by using a PCG algorithm to create rooms of varying sizes and formats. Key rooms are

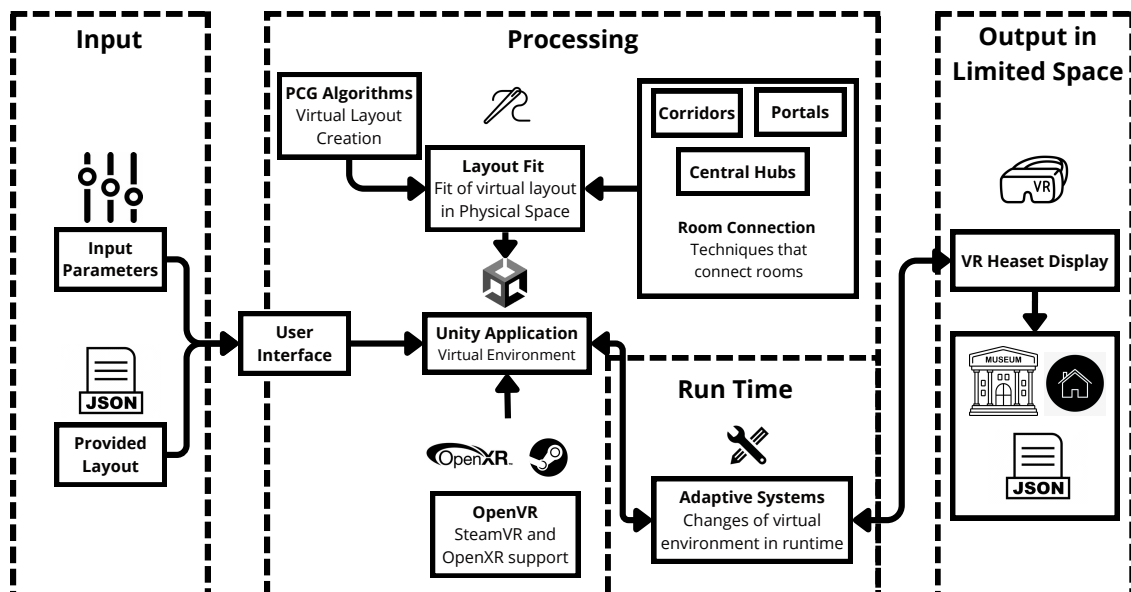


Figure 3.1: General Architecture of tool.

selected, and their midpoints (represented by circles in the figure) are calculated. These midpoints are then used in a [Delaunay Triangulation \(DT\)](#)¹ process (illustrated by lines) to generate a triangular graph, where vertices represent rooms and edges define the distance between adjacent rooms. A graph is constructed to store the layout.

The tool then generates a [Minimum Spanning Tree \(MST\)](#)² (shown by full lines) to ensure at least one path connects all rooms, allowing users to navigate from one end to another. At this stage, the rooms are connected to form the layout, as seen in step (C). However, this layout is not yet optimized for exploration in a confined physical space, requiring further adaptation.

In step (D), the tool calculates the ideal stride for each room, determining the stride increment needed to traverse the space. After calculating all strides, the tool selects a starting room and identifies adjacent rooms, adjusting overlaps and increasing stride in larger virtual spaces. The tool assigns scores to different techniques (e.g., Portals, Corridors, CH) based on input parameters, as detailed in [Subsection 3.4.4](#). Once a technique is chosen, the tool ensures it is properly integrated to guarantee safe and seamless movement between rooms. Each technique presents unique challenges, and the tool must meet their specific requirements, as detailed in [Section 3.4](#). The process repeats

¹Delaunay Triangulation [55, 42] creates a triangulation of points in a plane, ensuring no point lies within the circumcircle of any triangle.

²The minimal spanning tree [5] seeks the minimum-weight tree that connects all vertices of a weighted graph, minimizing the sum of edge weights.

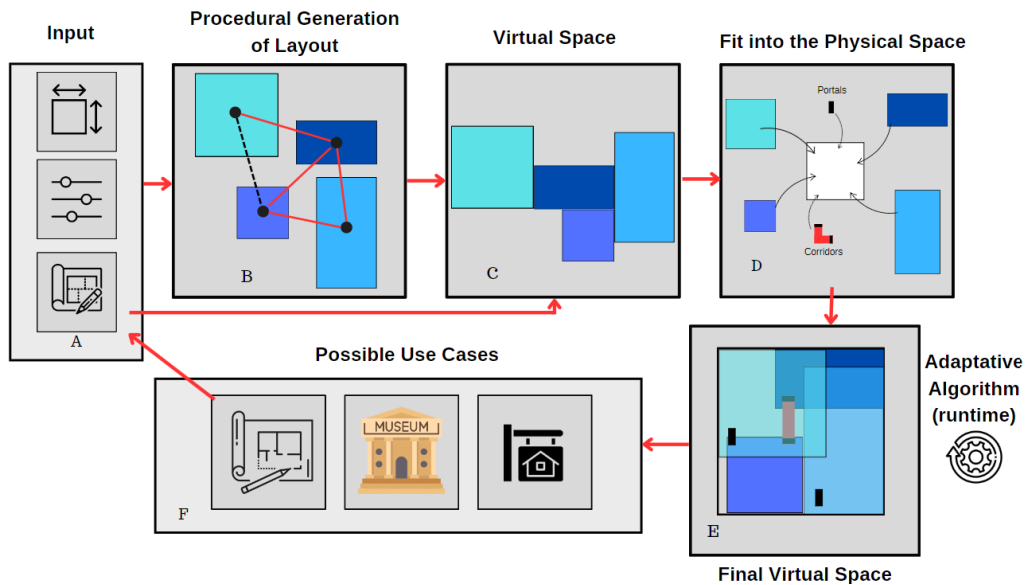


Figure 3.2: Illustration of the main components of the tool architecture. Step A - Input parameters. If a virtual layout is provided, then step B is skipped. Step B - Generation of virtual layout and room connections. Step C - Virtual space without adjustments. Step D - Fit virtual space into physical space using techniques dictated by parameters. Step E - Final virtual space with necessary adjustments. The adaptive algorithm starts to adapt space if necessary. Step F - Possible use cases for the tool.

for all remaining rooms. The result, as seen in step (E), is a connected layout adapted to the physical constraints of the user's space. In this scenario, portals are predominantly used to connect the rooms.

After the virtual space is finalized, users can begin exploration, with the adaptive system ensuring the environment adjusts in real-time to user preferences or requirements. For example, if the user prefers corridors to portals, the system modifies the layout accordingly. The adaptive systems can also generate new layouts, import existing ones, or export the current configuration without restarting the application.

Step (F) demonstrates applications such as exporting the layout for use as a parameter and exploring virtual museums or virtual apartments.

3.3 Virtual Layout Generation

This section provides an in-depth explanation of the **PCG** component of the virtual layout. It begins with **Subsection 3.3.1**, which details how the **Simple Room Displacement (SRD)** algorithm functions. This algorithm is responsible for generating the rooms that will form the virtual layout. Following this, **Subsection 3.3.2** describes how **DT** and graphs are used to connect the rooms created by the **SRD** algorithm. Lastly, **Subsection 3.3.3** presents the method for calculating the ideal user stride based on the dimensions of the physical area, ensuring that users can fully explore the virtual room without encountering walls.

3.3.1 Simple Room Displacement

A **Simple Room Displacement (SRD)** algorithm generates a specified number of rooms based on initial parameters. To place each room within the layout, the algorithm identifies a random point (RP) on a circumference using the following equations:

$$Angle = (2\pi * a), a \in [0, 1] \quad (3.1)$$

$$RP_x(r) = \cos(angle) * dist * r, dist \in [0, 1] \quad (3.2)$$

$$RP_y(r) = \sin(angle) * dist * r, dist \in [0, 1] \quad (3.3)$$

Here, r is the desired radius of the circle, and a and $dist$ are random values between 0 and 1. Note that increasing the radius results in greater spacing between rooms.

Figure 3.3 illustrates how the **SRD** algorithm creates the rooms for the virtual layout. In step (A), a representation of the circumference used by **Equation 3.1** is shown to generate random points. The circumference is centered at the coordinate (0,0) for convenience. The point (RP) indicates a possible coordinate for a room, and r is the radius used in the **Equation 3.1**. Once a point is determined, step (B) uses this point to generate a room with its center at these coordinates. The process of room generation will be discussed in greater detail in **Subsection 4.1.2**. Finally, in step (C), the algorithm repeats steps (A)

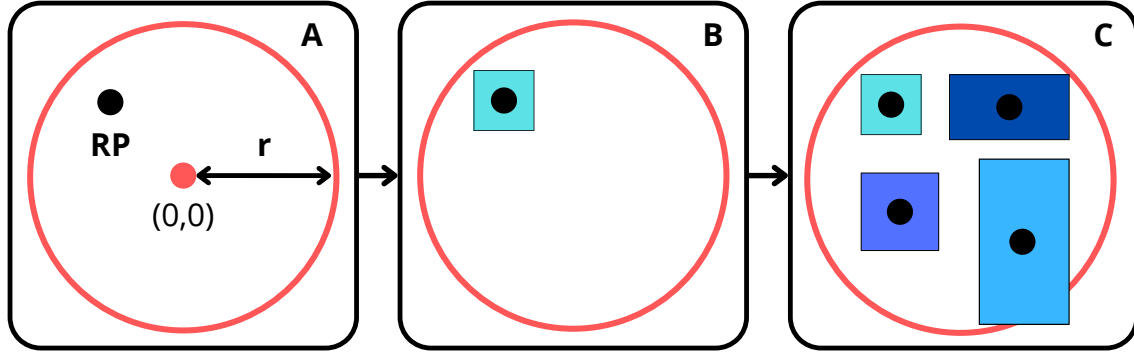


Figure 3.3: In step A, the algorithm creates a circumference of radius r and finds a random point. Step B creates a room at that point. Step C shows the iteration process until all rooms are generated.

and (B) for a number of iterations equal to the specified number of rooms. Algorithm 1 provides further details on how step (C) is executed.

3.3.2 Layout Room Connection

After the rooms are created, the next step is establishing connections between them to define the virtual layout. Two algorithms, DT and the MST, are essential for creating these connections.

Delaunay Triangulation [55, 42] involves creating a triangulation from a set of points in a plane, ensuring that no point lies inside the circumcircle of any triangle. This is crucial in fields such as terrain modeling, as it allows for flexible resolution adjustment based on application needs.

Minimum Spanning Tree [5] focuses on creating a tree with the least total weight that connects all vertices in a weighted, undirected, and connected graph.

In Figure 3.4, the subsequent steps after the SRD creates the rooms are illustrated. Step (A) shows all the rooms created and their respective midpoints. These midpoints are relevant because they are input into the DT algorithm. The DT algorithm returns a list of triangles that connect only adjacent rooms. An example of this output can be seen in step

Algorithm 1 How the Simple Room Displacement generates a desired number of rooms

Require: $numberOfRooms \geq 3$ ▷ Delaunay Triangulation needs at least 3 points(rooms)
 $r \leftarrow radius$ ▷ Desired circumference radius
for $i = 0$ to $numberOfRooms$ **do**
 $coordinates \leftarrow (RP_x(r), RP_y(r))$ ▷ The circumference angle is the same
 $CreateRoom(coordinates)$
end for

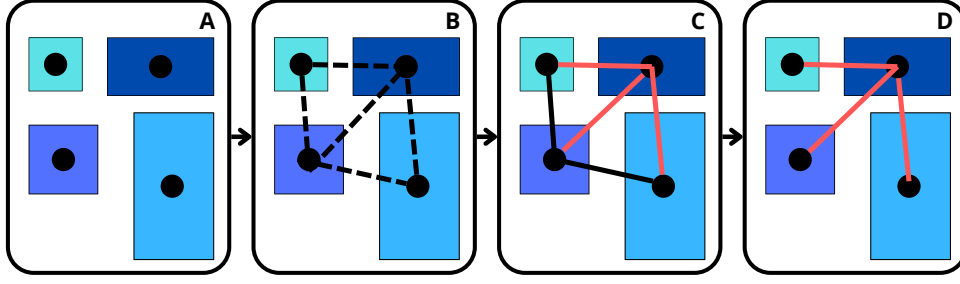


Figure 3.4: This figure illustrates the process of connecting rooms to form a virtual layout. Step A shows the rooms and their midpoints. In Step B, the Delaunay Triangulation algorithm connects adjacent rooms. Step C highlights the connections the Minimum Spanning Tree algorithm selects, shown in red. Finally, Step D removes unselected edges, leaving the final connections that define the layout.

(B), where each room's midpoint is connected only to adjacent room midpoints, forming triangles between the midpoints. With the DT algorithm's output, a graph can be created where each midpoint is a node representing that room's midpoint, and each connection is an edge linking two nodes (rooms).

Once the graph is created, step (C) involves running the MST algorithm on the graph to find the minimum path that connects all the rooms. This path will be used as the definitive connection for the virtual layout. The MST algorithm creates more manageable and realistic layouts. Consider the layout of a house, the main room often connects to all other rooms, and each room typically connects back to the main room. This common layout pattern is what the MST algorithm emulates. Another reason for using the MST is that in more complex layout (e.g., more than 10 rooms), the DT algorithm would return all possible connections between rooms, resulting in a chaotic layout for users to explore. By employing the MST algorithm, the connections are significantly reduced to the minimum needed to allow a simpler exploration of all rooms in the layout.

Lastly, in step (D), all edges not included in the path chosen by the MST algorithm are removed, leaving only the selected edges.

3.3.3 Room Stride Calculation

With the virtual layout created, having all the rooms and their respective connections, the last step before applying room connection techniques to enable natural navigation in VR is to calculate the ideal stride for each room. This ideal stride is crucial because it adjusts the user's stride when the user enters a room larger than the physical layout to allow them to explore the entirety of the room.

To calculate this stride (S), the following formula can be used:

$$S_{room} = \frac{\max(width, height)}{\min(PA_{width}, PA_{height})} \quad (3.4)$$

where *width* and *height* are the room dimensions where the stride is being calculated. PA_{width} and PA_{height} represent the dimensions of the physical area, and their relevance is better explained in [Subsection 3.5.3](#), where it will be explained how VR headsets handle and provide these dimensions. The logic behind the calculation is to determine the largest dimension of the virtual room and divide it by the smallest dimension of the physical area. This approach ensures that the user's stride allows navigation throughout the entire room while remaining within the shortest dimension of the physical space. The formula is based on the Max-Min Theorem [41, 40], commonly used in optimization to balance competing constraints by maximizing the minimum gain or minimizing the maximum loss. In the context of VR navigation, the theorem ensures users can explore VE fully without exceeding the physical boundaries. The Max-Min Theorem has broad applications across scientific fields, including game theory and economics, where it analyzes optimal strategies under constraints. In this context, it optimizes the stride value to enhance users' ability to navigate virtual spaces while avoiding physical walls.

Once all room strides are calculated, the virtual layout is complete.

3.4 VR Room Connection

With the virtual layout defined and all its rooms created, the next challenge is to connect the rooms in a way that allows users to explore the entire layout without leaving the physical area. To address this challenge, three techniques were developed for connecting rooms: **Portals**, **Corridors**, and **Central Hubs**. This section will describe how each technique is applied to ensure automatic placement and seamless connections between rooms. This section also covers how a scoring system manages these techniques.

The choice of these connection techniques was carefully considered based on their relevance in existing literature. All of these techniques maximize the use of physical space by creating overlapping areas in the virtual environment. Each technique offers distinct advantages and addresses specific challenges in connecting rooms:

- **Portals** were chosen because they are widely recognized in the literature on Impossible Spaces. They provide a flexible and efficient way to connect rooms without requiring additional virtual elements, as they can be placed anywhere in the virtual layout and are not confined to walls. Portals offer users a clear preview of the connected rooms, enhancing spatial awareness. However, they have limitations, such as requiring precise placement to avoid conflicts with other portals or objects in the environment.
- **Corridors** were selected based on the concept of transitional areas, commonly used in games, architecture, and real-world layouts to connect rooms. Corridors preserve the natural flow of room connections and serve as transition zones, allowing smoother navigation between spaces. This makes them more familiar, as they appear as natural doorways within the walls of connected spaces. However, corridors require

additional virtual elements (i.e., adding walls to build the corridor). They also do not provide users with a preview of the next room, which may require more mental effort for users to create a mental map of the [VE](#). Corridors may be impractical in very small play areas, such as 1 x 1 meter, as the space may not accommodate both the corridor and comfortable navigation.

- **Central hubs** were introduced to connect more than two rooms. This technique simplifies navigation, as a hub can link multiple rooms or even other hubs, making it useful for large, interconnected spaces (e.g., museums). By acting as a centralized space, it reduces the mental effort required by users, allowing them to easily return to the hub to choose their next room. However, it adds non-original elements (i.e., new rooms acting as hubs) to the [VE](#) that alter the original layout.

Each technique utilizes the graph of the virtual layout, specifically its edges, to determine which rooms to connect. Throughout this section, figures will illustrate each technique, showing the results of the rooms fitted within the physical area. The room dimensions are based on the applied stride. In practice, if the rooms were shown at their original size, they would be much larger than the physical area. Therefore, in the figures, the room sizes are adjusted for better visualization.

3.4.1 Portal

Starting with portals, [Figure 3.5](#) illustrates the necessary steps for automatically placing portals to connect two rooms. The process begins by identifying the two rooms and determining their midpoints, as shown in step (A). These midpoints can be extracted from the graph's edges and nodes, which store the necessary information about the rooms.

In step (B), the rooms are overlapped using their midpoints and dimensions, resulting in an intersection that defines a common area between them. This intersection ensures that portals placed within this area are positioned correctly in both rooms. This step is crucial because two rooms may have different shapes, and certain areas in one room might not correspond to valid areas in the other room. By identifying a common intersection, it is possible to ensure that any portal placement is valid in both rooms.

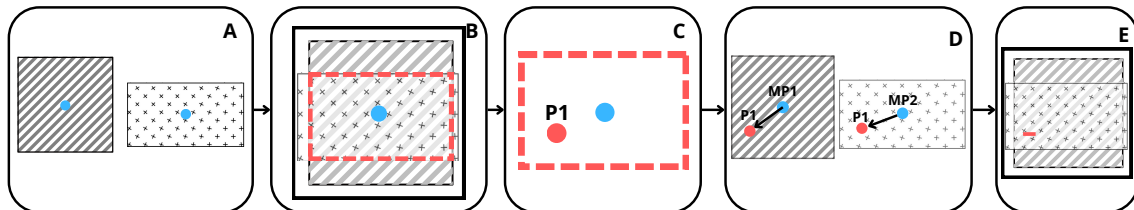


Figure 3.5: Overview of steps to create a portal connecting two rooms in a virtual layout.

Following this, in step (C), a random coordinate within the intersected area is calculated to place the portals. Proper alignment of the portals in both rooms is essential to prevent offsets in the virtual layout when traversing between them. Misalignment can occur if portals are not positioned identically in both rooms or if one portal is larger than the other, leading to a gradual shift in the virtual layout. Over time, this shift could prevent users from fully exploring the entire layout.

After determining the portal positions, the algorithm validates them to ensure they do not interfere with existing portals or objects in the layout. This position validation is crucial because, as more rooms are connected, a room can have more than two portals within it. To prevent conflicts with existing portals, the positions of the portals in the rooms are stored, which helps place new portals in valid positions that do not overlap with existing ones. If an invalid position is detected, a new random position is generated and verified. This process repeats until a valid position is found. In step (D), the final portal position (PP) within each room is calculated using the following formula:

$$PP = MP_{room} + P1, \quad (3.5)$$

where MP is the midpoint of the room and $P1$ is the portal position. The result is the new position within the desired room. Finally, step (E) displays the final result, showing the two rooms fitted into the physical space with the portal connecting them.

3.4.2 Corridor

Before developing the algorithm to generate corridors, it is essential to address three key considerations related to the challenge of accommodating corridors in a limited physical play area. First, the virtual corridor must be wide enough to ensure comfortable passage and prevent feelings of claustrophobia while also being compact enough to integrate seamlessly into the VE. The recommended minimum width is 50 centimeters to allow users to navigate them comfortably. Second, the corridor design must ensure that entrances guide users outward from a room and exits lead them inward, maintaining logical flow and smooth transitions within the VE while preventing users from walking into walls or outside the designated boundaries. Third, each virtual layout room must have an exit point (i.e., door) connecting to the other room. Every pair of connected rooms has a corresponding pair of entry and exit points positioned opposite each other in the VE. The corridor technique must preserve these entry/exit points. While it may seem straightforward to design a corridor that immediately turns into another room, this would violate the established rule of opposite entry and exit doors in the rooms.

These considerations were taken into account to implement the algorithm for automatically generating corridors linking two rooms in a VE. Figure 3.6 illustrates the algorithm's steps to generate a corridor connecting two rooms. The first step (A) involves selecting two rooms and their respective door positions. As with portals, the rooms and their

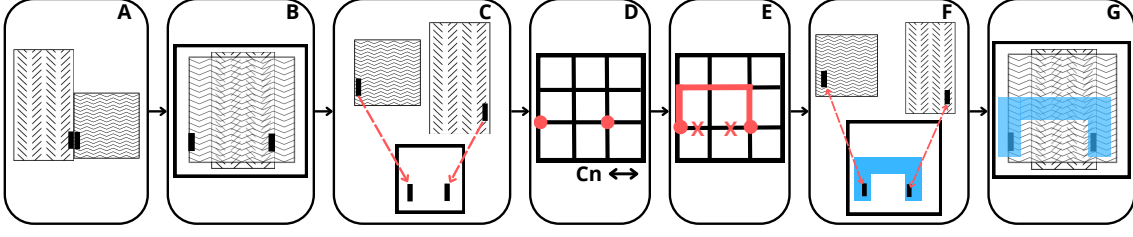


Figure 3.6: Overview of steps to create a corridor connecting two rooms in a virtual layout.

connections are extracted from the virtual layout graph. The door position in the rooms is handled by the [SRD](#) algorithm and will be covered in more detail in [Subsection 4.2.3](#). This step illustrates how, in a real layout, the rooms would be connected. In step (B), the rooms are overlapped by aligning their middlepoints, which helps determine the position of doors in the physical area. This overlap also ensures that each room is situated at a safe distance from any physical barriers. This step illustrates how the rooms and their doors are fitted inside the physical area to create the virtual layout. Next, in step (C), the door positions are mapped to their corresponding local door position (LDP) in the physical area using the following formula,

$$LDP = \frac{(P_{door} - MP_{room})}{S_{room}}, \quad (3.6)$$

where P_{door} is the door position in the layout, S_{room} is the room scale and MP_{room} is the room midpoint. These positions serve as the starting and ending points for creating the corridor.

Moving to step (D), the physical area is divided into a grid, with each subdivision matching the corridor node's size (a corridor is composed of smaller corridor nodes). This grid facilitates the creation of a graph, where each node represents a potential corridor node position and its edges a potential direction the corridor can go. Although the door positions are added to this graph, they may not always align perfectly with a node position. Thus, the door position needs to be adjusted (ADP) to match the nearest node position using the formula,

$$ADP_{Cn} = \frac{\lceil P_{door} * \frac{1}{Cn} \rceil}{\frac{1}{Cn}} \quad (3.7)$$

where Cn is the corridor node size and P_{door} is the local door position. The numerator is rounded to the nearest integer value before dividing, returning a discrepancy α between the original door position and node positions. This value α is recorded and used in Step (F).

Step (E) involves adjusting the graph to prevent undesirable corridor generation by removing edges where corridors either start inside or end outside the room. Next, a

pathfinding algorithm, such as A*, is used to find a path between the start and end nodes. A* [17] is a widely used pathfinding algorithm due to its simplicity, efficiency, and adaptability. While A* is employed here, other algorithms can be adapted to control factors such as the number of corridor turns or other criteria.

Once the path is established, step **(F)** instantiates corridor nodes along this route. After all corridor nodes are created, the earlier recorded offset α is added to the door position, ensuring precise door placement. The corridor is then connected to the room doors. Finally, step **(G)** displays the virtual space, showing the rooms and connecting corridor.

One additional step is required. If a room is larger than the tracking area, the user's stride is proportionally increased to ensure full navigation. Rooms may have different user strides, so the corridor must seamlessly adjust the user's stride while within it. Achieving this transition is complex, requiring precise calculations to maintain the correct entry and exit positions within the physical space. The user's stride in the corridor must also be adjusted according to the available physical area. This step is described in the next section.

3.4.2.1 Seamless Stride Transition in Corridors

Achieving a seamless stride transition within a corridor is challenging due to the constraints of the physical space. The goal is to adjust the user's stride in such a way that they do not notice the change while moving through the corridor. This adjustment must consider the following requirements:

- **Stride Adjustments:** Adjusting the user's stride impacts how quickly they traverse the corridor. If the stride is increased too much, the user may reach the next room at an incorrect physical location, disrupting the virtual layout and causing misalignment. On the other hand, decreasing the stride too much can cause the user to move too slowly and potentially reach a physical wall before exiting the corridor. Proper calibration is essential to ensure smooth transitions between rooms without exceeding the physical boundaries.
- **Physical Area Consideration:** The size of the physical area directly affects the virtual layout. A reduced physical space can make virtual rooms appear larger, requiring the user's stride to increase accordingly. However, this can cause the user to traverse the area too quickly. Corridors must ensure precise entry and exit points, adjusting the user's stride relative to the physical space to maintain smooth transitions between rooms without rapid traversal.
- **Seamless Transition:** The user's stride must transition smoothly between rooms, ensuring an imperceptible change as they navigate the corridor. This requires continuous adjustment throughout the corridor to maintain a seamless stride shift between different room sizes.

Figure 3.7 illustrates the requirements and constraints corridors must follow to ensure seamless transitions between rooms.

Step (A) shows a corridor connecting two rooms within a limited physical space. While the corridor and rooms are larger than the physical area, their dimensions are adjusted to fit within it. Each room has a door acting as the corridor's entry or exit point. If the user does not enter or exit exactly through these doors, the virtual layout can misalign, causing offsets. This step highlights the challenge of connecting rooms of different sizes. If the user moves too quickly through the corridor, the door positions within the physical space may shift, leading to misalignment. In smaller physical spaces, stride adjustments are critical to maintaining alignment and preventing such offsets.

Step (B) illustrates how strides, calculated using Equation 3.4, are based on the physical area dimensions. If the physical area is reduced, the room strides increase to compensate.

Step (C) outlines the solution for achieving seamless transitions between rooms. A corridor consists of multiple nodes connected sequentially, and to ensure smooth transitions, each node is assigned a specific stride. The first and last nodes are adjusted to match the strides of the connecting rooms. These strides are then fine-tuned to fit the physical area dimensions, as explained further in this section with Equation 3.8. The middle node is assigned the ideal stride for the corridor, allowing the user to traverse its longest section within the physical space. To ensure a smooth transition, if the starting node's stride is less than the ideal corridor stride, strides are gradually increased from the start to the midpoint of the corridor. If the starting node's stride exceeds the ideal stride, strides gradually decrease. This same process applies from the midpoint to the end of the corridor. This method ensures a gradual, seamless transition between room strides within the corridor.

Ideal Corridor Stride is a key concept. The corridor is treated as a room, meaning it has an ideal user stride, just as rooms do. This stride represents the minimum increase necessary for the user to traverse the corridor effectively within the physical space. To calculate the ideal corridor stride, Equation 3.4 is applied to the corridor's dimensions. This stride is an anchor for connecting the adjusted strides of the two rooms. Since the corridor is built with nodes, as illustrated in Figure 3.7 step (C), the middle node is set with the ideal stride. This ensures that when the user reaches the midpoint — typically the longest traversal — they will have the optimal stride, ensuring a smoother, more consistent experience.

Adjusted Room Strides refer to the strides at the start and end nodes of the corridor to match the rooms they connect but adjusted per the physical area dimensions. If the physical space is small while room proportions remain the same, strides increase to compensate. When entering a corridor that starts with the same stride as the room can cause the user to move too quickly through the corridor, leading to significant offsets in the virtual layout. To prevent this, the user's stride upon entering the corridor —

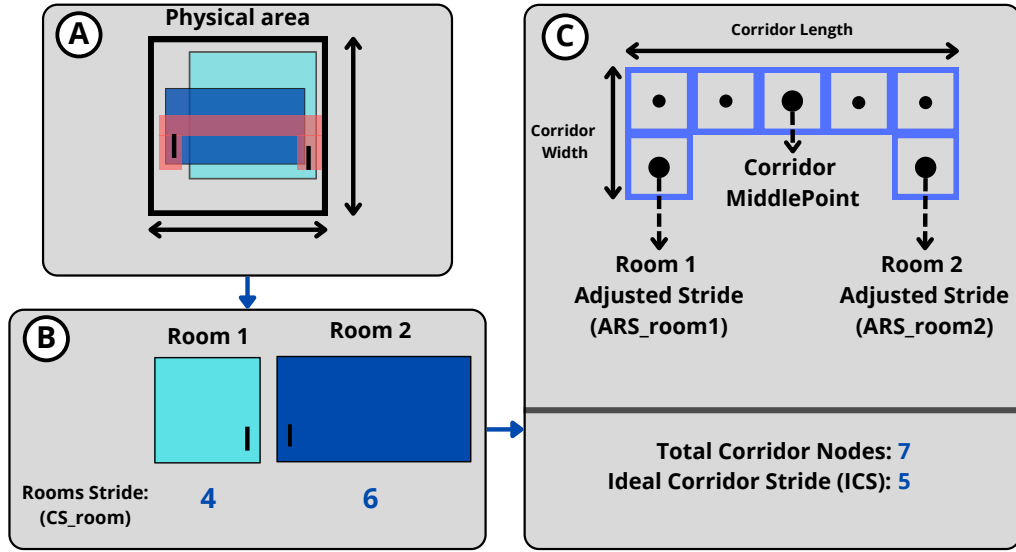


Figure 3.7: Seamless stride transition in a corridor. Step A represents the physical area dimensions with the virtual layout fitted inside. Step B shows the strides of the rooms to related to the physical area dimensions. Step C shows how the physical area dimensions affect the corridor length and ideal stride. It also shows how each corridor node has a stride in order to gradually increase or decrease the stride alongside the corridor.

called the Adjusted Room Stride (ARS) — is adjusted according to the corridor’s size and the physical area. This ensures seamless transitions between rooms and prevents misalignments in the VE. The ARS is calculated using the following formula:

$$ARS = \begin{cases} 1 - \frac{S_{room}}{TN-PA}, S_{room} > S_{corridor} \\ 1 + \frac{S_{room}}{TN-PA}, S_{room} \leq S_{corridor} \end{cases} \quad (3.8)$$

where S_{room} is the stride of the room being considered, TN is the total nodes in the corridor, PA is the smallest dimension of the physical area (e.g. if the physical area is 2.5m x 2m, PA is going to be 2m) and $S_{corridor}$ is the ideal stride of the corridor, which Equation 3.4 can calculate using the corridor dimensions as variables. This formula compares the room stride with the ideal corridor stride. If the room stride exceeds the ideal, the ARS must be reduced. This reduction is proportional to the total number of nodes in the corridor and the physical area size. In smaller physical spaces (e.g., 1.5m x 1.5m), where room strides are higher, the reduction is more significant to ensure a seamless transition. Conversely, if the room stride is smaller than the ideal corridor stride, the ARS must be increased, also in proportion to the physical area and total corridor nodes. These adjustments smooth the transition between the room and corridor, preventing abrupt stride changes that could disrupt the virtual layout or user experience.

This adjustment is vital for ensuring the user enters and exits the corridor with an optimal stride, allowing seamless exploration of the virtual layout. It also makes the corridor adaptable to different dimensions of the physical area. As shown in Figure 3.7,

changes in physical area can present challenges. In larger areas, room strides are smaller, and the corridor can be longer, requiring only slight stride adjustments for smooth transitions. However, in smaller spaces (1.5m x 1.5m), room strides increase significantly due to limited space, causing the user to traverse the physical area much faster. If this higher stride were maintained in the corridor, the user could reach the end prematurely, creating offsets in the layout. Therefore, a more substantial stride reduction is necessary to preserve the corridor's start and end positions. While this reduction may cause a more noticeable transition, it is essential to maintain the alignment of the virtual layout.

After introducing the concepts of **Adjusted Room Strides** and the **Ideal Corridor Stride** and explaining how to calculate them, the next step is to detail how to connect these values smoothly. To achieve a seamless transition between each *ARS* and the corridor stride, **Linear Interpolation** (*Lerp*) can be used.

Linear interpolation [19] is a method used to estimate an unknown value that lies between two known values. This approach assumes that the change between these two known values is linear. In this context, the stride at a given corridor node, positioned between the start/end node and the middle corridor node, can be calculated using the following formula:

$$Lerp = start + percentage * (end - start) \quad (3.9)$$

where *start* is the starting value to interpolate, *end* is the final value to interpolate, and *percentage* is how close the result value is going to be to the *start* or *end*. A lower percentage value means the result value is close to the *start* while a higher value is closer to the *end* value.

With Lerp established, to calculate the corridor stride (CCS) the following formula:

$$CCS = \begin{cases} Lerp(ARS(S_{room1}, TN, PA, S_{corridor}), S_{corridor}, \frac{CN}{TN}), & CN < \frac{TN}{2} \\ Lerp(S_{corridor}, ARS(S_{room2}, TN, PA, S_{corridor}), \frac{CN}{TN}), & CN \geq \frac{TN}{2} \end{cases} \quad (3.10)$$

where *CN* is the current corridor node being calculated, *TN* is the total nodes in the corridor, *S_{corridor}*, *S_{room1}*, *S_{room2}* are the respective stride of the corridor, the first room and the second room to be connected. The CCS formula is applied to each node within the corridor. It determines the appropriate stride at each corridor node by checking its proximity to the connected rooms or the corridor midpoint. If the current node is closer to the first room, it applies Lerp between the **adjusted first room stride** and the **ideal corridor stride**. Conversely, if the node is closer to the second room, it applies Lerp between the **ideal corridor stride** and the **adjusted second room stride**. This approach ensures a linear adjustment of the stride from the start of the corridor to the ideal stride value and, similarly, from the ideal stride to the end corridor stride. This method results in four possible scenarios, all of which are illustrated in [Figure 3.8](#). The scenarios include cases where both rooms have an adjusted room stride (ARS) higher than the ideal corridor

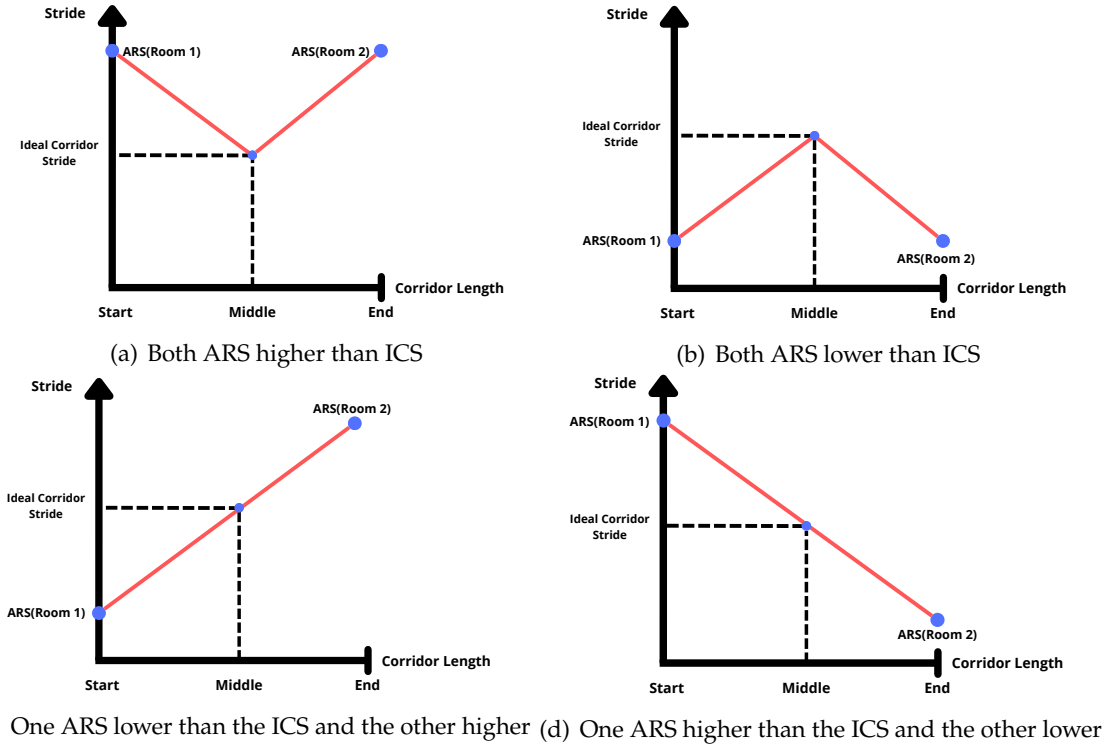


Figure 3.8: There are four different scenarios of the room's strides compared to the corridor's ideal stride. The plots demonstrate the stride changes alongside the corridor.

stride, where both have a smaller stride than the ideal or where one room has a higher and the other a smaller stride than the ideal. The important note is that while the corridor can adapt to the dimensions of the physical area, there are limits to its precision and adaptability. When the physical area is too small, it becomes increasingly difficult to prevent offsets in the virtual layout due to these physical space constraints. These limitations will be explored more deeply in [Subsection 4.2.3](#).

To summarize, [Subsection 3.4.2](#) encapsulates the theory and logic behind designing corridors that seamlessly connect two virtual rooms, ensuring walking experiences in VR while navigating within constrained physical spaces. These corridors are engineered to provide smooth transitions between varying room strides and adapt dynamically to the dimensions of the physical area, thereby enhancing user immersion and comfort.

3.4.3 Central Hub

The [CH](#) builds on the portal method by adding a fixed-position room, the central hub, into the layout. Unlike randomly placed portals, the [CH](#) places the central hub at predefined portal locations, connecting it to various rooms and acting as a central linking point. While this additional room may alter the user's perception of the original layout, it greatly enhances navigation within the [VE](#), as detailed in [Section 5.2](#).

Figure 3.9 illustrates the CH placement. In step (A), an existing virtual layout with connected rooms is analyzed. Rooms to be added to the CH are determined by the Scoring System (SS) described in Section 3.4.4. The SS evaluates each room connection, and when the CH technique is selected, both rooms connected by that link are added to the CH selection. Alternatively, in step (A), room selection could be based on layout regions, where a region contains multiple rooms. Using this approach, each CH is assigned rooms from the same region, creating a more organized structure rather than random room assignment.

After evaluating all connections, the SS initiates the creation of the CH, as detailed in Algorithm 2. The algorithm iterates through the *CHroomSelected* list, adding rooms to a temporary list. When this list contains eight rooms (the maximum capacity of a CH to avoid overcrowding), the CH algorithm creates the hub with those rooms. The list is then cleared, and the process repeats until all rooms are added to the CH.

Step (B) illustrates the creation of the CH and its subsequent impact on the layout. The algorithm extends each previous connection between rooms to the newly created CH. The CH is generated using a pre-fabricated room that contains placeholders for portals—either eight portals (two doors per wall) or four portals (one door per wall). The algorithm then iterates through each selected room provided by the SS, assigning a portal placeholder for each. This process continues systematically until all rooms have been associated with their respective portals.

To accurately determine the position of the portal placeholder of the CH within a

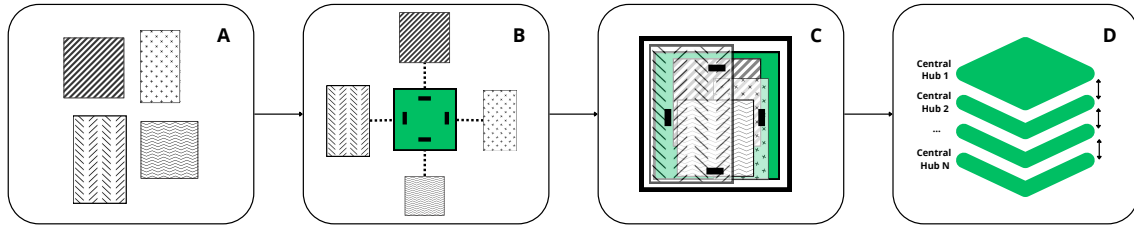


Figure 3.9: Overview of steps to create a central hub in a virtual layout.

Algorithm 2 How Scoring System setups the creation of Central Hubs

Require: *CHroomSelected.Count* ≥ 1

temp $\leftarrow$ List < Room >

max $\leftarrow$ *CHMaxCapacity*

▸ How many rooms the CH connects

for *i* = 0 to *CHroomSelected.Count* **do**

room $\leftarrow$ *CHroomSelected[i]*

temp.Add(room)

if *temp.Count* = *max* **then**

 CreateCH(*temp*)

temp.Clear()

end if

end for

CreateCH(*temp*)

CreateCHConnections

▸ Remaining Rooms

▸ Connect the Central Hubs

designated room (PP_{room}), the following formula is applied:

$$PP_{room} = MP_{room} + \frac{(PP_{CH} - MP_{CH})}{S_{CH}} * S_{room} \quad (3.11)$$

where MP_{room} , MP_{CH} are selected room and the **Central Hub** middlepoints, PP_{CH} is the position of the portal placeholder and S_{room} , S_{CH} are the respective strides of the room and the **CH**. This formula is adapted from [Equation 3.5](#), but in this case, the portal position is only common to one room (the central hub), requiring translation to the selected room. Since both rooms may have different strides, an additional step is to align the portal position consistently across both rooms. To achieve this, the portal position is divided by the central hub's stride, localizing the position within the physical area. Then, this localized position is multiplied by the selected room's stride, and the result is added to the room's midpoint. This ensures proper alignment of the portal position within the selected room, considering its specific stride.

In cases where more than one **CH** exists, the method for connecting multiple hubs is employed. This is accomplished through an elevator that transports the user between hubs. Step (C) illustrates a layout with a single **CH** at the center of the physical area, with three portals connecting to respective rooms. When multiple hubs are present in the same **VE**, they must be interconnected via an elevator system, as depicted in step (D), allowing users to choose their destination hub.

Central Hub Connections: When placing a **CH** portal, there are instances where an invalid position occurs. This typically happens when the **CH** and the selected room have different shapes, causing the portal to be improperly placed within the room (e.g., positioned on top of furniture). In such cases, an alternative portal position from the current **CH** is selected. If no valid positions are available, these unused portal positions and the unconnected rooms are stored for later processing. Once all necessary **CH** portals have been placed and connected, the final step is to link the **CH** together. The means to connect **CH** will be more detailed in [Subsection 4.2.4](#). During the process of connecting **CH**, the unconnected rooms are revisited. As each **CH** is connected, each unused portal of the **CH** is checked to determine if any are suitable for linking the remaining unconnected rooms. If, after this process, there are still rooms that require connection, new **CH** are created specifically to connect these last rooms.

3.4.4 Scoring System

The **Scoring System (SS)** is a system designed to manage which room connection technique — Portals, Corridors, or **CH** — will be applied to each room connection within the virtual layout. The **SS** takes various input parameters, detailed further in this document, that can significantly influence how the virtual layout is perceived and navigated. By offering fine-tuned control over the selection of room connection techniques, the **SS** ensures that

the virtual layout is optimized for alignment with the physical space. The **SS** operates based on a primary parameter called the **Degree of Realism**, which is divided into three sub-parameters. Each can be adjusted according to user preferences to determine how the different techniques are prioritized. The sub-parameters are as follows:

- **Realistic factor:** This parameter dictates the desired level of realism for the virtual layout. A more realistic layout (e.g., a building) prioritizes techniques that connect rooms similarly to the real world, such as corridors. Conversely, less realistic scenarios (e.g., games) allow for the use of techniques that defy reality, such as portals that connect two spaces that might exist far apart.
- **Layout preservation:** This parameter refers to whether the user wants to maintain the spatial perception of the original environment layout. Techniques such as corridors and portals can preserve the original room connections or door positions, thereby retaining most of the original layout. Alternatively, if preserving the layout is not a priority, techniques such as introducing a **CH** can be used. In this case, adding an extra room and removing direct connections between rooms can significantly alter the spatial perception of the original layout.
- **Layout Size:** This parameter relates to the number of rooms in the layout. A layout with fewer rooms can be easily accommodated using portals and corridors. However, as the number of rooms increases, the complexity of connections also increases. The **CH** technique is more suitable for larger layouts, providing an easier way to navigate by connecting the rooms through an interconnecting space.

The **SS** calculates different weights for each technique based on these three parameters. **Table 3.1** shows each parameter and the corresponding weights for each technique. There is also an option for custom parameters, where each technique weight can be manually changed to enforce a desired set of techniques. Once the parameters are set, the **SS** calculates the score of each technique based on the desired parameter's weight. This technique score (TS) is calculated using the following formula,

$$TS = W_l(T) + W_r(T) + \frac{1}{2}W_n(T) \quad (3.12)$$

where T is a given technique, $T \in \{P, C, CH\}$ and each weight is extracted from **Table 3.1**.

Subsequently, based on the technique scores, it is possible to calculate the probability (PT) of each technique being selected to connect two different rooms using the following formula,

$$PT = \frac{TS(T)}{TS(P) + TS(C) + TS(CH)} \quad (3.13)$$

Table 3.1: The weight (W) of a technique (T) is calculated based on the three parameters of the Scoring System (SS). The table describes the meaning of the range of weights for each parameter.

	Layout Preservation $W_l(T)$ Weight [-1,1]		Realistic Factor $W_r(T)$ Weight [-1,1]				Layout Size $W_n(T, n)$ Weight [-1,1]	
	Original	Altered	Unrealistic	Semi-U.	Semi-R.	Realistic	Big	Small
Portal	1	-1	1	0.5	-0.5	-1	-1	1
Corridor	1	-1	-1	-0.5	0.5	1	-1	1
C.Hub	-1	1	1	0.5	-0.5	-1	1	-1

where T is a given technique, $T \in \{P, C, CH\}$. This formula calculates the probability of a given technique by dividing its score by the total value of all technique scores.

Using these probabilities, the SS iterates through each room connection and selects the technique to be employed, as outlined in Algorithm 3. Techniques with higher weights are more likely to be chosen, while techniques with lower weights have a lower probability of selection. Thus, the SS prioritizes the connection techniques deemed most appropriate based on the parameter weights specified by the user.

For this thesis, the SS is configured to choose among three techniques. However, the SS is designed to be extensible, allowing for future integration of new room connection techniques. This system also monitors the techniques currently placed in the layout, and in Chapter 4, it will be explained how the SS can be used to dynamically adapt the virtual layout in real-time, modifying its room connection techniques by adjusting input parameters while inside the virtual layout.

Algorithm 3 How Scoring System selects room connection techniques

Require: *PortalProb, CorridorProb, CHProb*

graph $\leftarrow$ RoomConnectionGraph

▷ Virtual Layout connections

CHroomSelected $\leftarrow$ List < Room >

for *edge* in *graph.edges()* **do**

room1 $\leftarrow$ *edge.from*

room2 $\leftarrow$ *edge.to*

random $\leftarrow$ RandomNumber

▷ Random number between 0 and 1

if *random* > *PortalProb* **then**

ApplyPortalTechnique(*room1*, *room2*)

else if *random* > *PortalProb* + *CorridorProb* **then**

door1 $\leftarrow$ *from.GetDoor()*

door2 $\leftarrow$ *to.GetDoor()*

LocalStartDoor1Pos $\leftarrow$ LDP(*door1*, *room1.GetStride()*, *room1.position()*)

LocalStartDoor2Pos $\leftarrow$ LDP(*door2*, *room2.GetStride()*, *room2.position()*)

StartPos $\leftarrow$ ADP(*LocalStartDoor1Pos*)

EndPos $\leftarrow$ ADP(*LocalStartDoor2Pos*)

ApplyCorridorTechnique(*StartPos*, *EndPos*, *room1*, *room2*)

else

CHroomSelected.add(*Room1*)

CHroomSelected.add(*Room2*)

end if

end for

ApplyCHTechnique(*CHroomSelected*)

▷ For more detail, see Subsection 3.4.3

3.5 Adaptive and Additional Features

This section will address supplementary concepts that enhance understanding of the previously discussed topics. These concepts are crucial for complementing the existing framework. First, the concept of adaptiveness, its application within the tool, and its potential to enhance established methods will be explored in [Subsection 3.5.1](#). Next, the process of importing and exporting virtual layouts will be detailed in [Subsection 3.5.2](#). Finally, the role of VR headsets in defining the physical boundaries of available space and how these dimensions can be extracted will be discussed in [Subsection 3.5.3](#).

3.5.1 Adaptiveness in Tool

The concept of adaptiveness in this thesis refers to the tool's ability to modify virtual layouts and room connection techniques in real-time to fit the physical space, dynamically adjusting elements (rooms, Portals, Corridors, etc.) based on user needs or preferences. This allows the tool to accommodate any physical space, whether large or small, by adjusting the [VE](#) accordingly.

Adaptiveness as a core feature enhances the tool's flexibility and versatility. For instance, the [SS](#) allows users to modify input parameters while navigating the virtual layout. This feature can seamlessly reassess and update room connections based on new parameters without disrupting the user's experience.

To maintain immersion, when changes affect the whole virtual layout, the tool guides the user to a designated room outside the main layout while generating new configurations. This approach grants users greater control over the environment, enabling them to customize it according to their preferences and explore various parameter sets.

Additionally, adaptiveness includes systems that operate in the background to enhance the user experience without direct intervention. For example, if navigation offsets occur, the tool automatically detects and corrects them, ensuring a seamless experience with minimal user awareness.

Briefly, adaptiveness enables the tool to adjust [VE](#) to match physical spaces, either autonomously or based on user input, enhancing natural navigation in [VR](#). This ensures the tool remains responsive and dynamic during the whole experience.

[Figure 3.10](#) illustrates the tool's pipeline and its key components. The process begins with input parameters that guide the procedural generation of virtual layouts. The [SRD](#) algorithm generates a layout that meets these criteria, calculating room strides. The [SS](#) then selects appropriate room connection techniques based on the input parameters, resulting in a final layout ready for exploration in VR, where walking is enabled within limited physical space. While the user navigates, adaptive algorithms continuously run in the background, making real-time adjustments, correcting offsets, or modifying layout aspects to better suit user needs. These systems can also accept new parameters to change the [VE](#) as required. Finally, the tool provides users the flexibility to export or load layouts

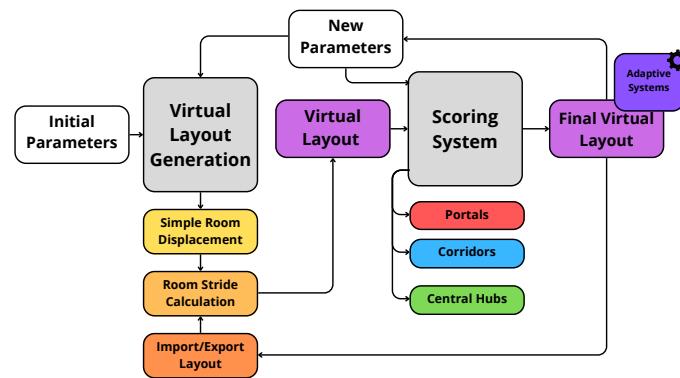


Figure 3.10: Demonstration of the tool pipeline, showcasing all the necessary steps to create a virtual layout. It also shows how the adaptive systems pass new parameters to generate new virtual layouts at runtime.

during runtime, complete with the desired room connection techniques. This adaptability ensures the [VE](#) remains optimized for both physical space and user preferences, providing a seamless and immersive [VR](#) experience.

3.5.2 Virtual Layout Import and Export

The tool accepts an externally provided layout to override its procedural generation when generating a virtual layout ([Figure 3.2](#)). Additionally, the tool must be capable of exporting created layouts for later use. The layouts the tool can import may either be those generated by the tool itself or manually created by a developer. The specifics of how a virtual layout can be manually created for use within the tool will be further detailed in [Subsection 4.1.3](#).

This subsection aims to explain how the tool’s virtual layouts can be exported and imported, utilizing the concept of JavaScript Object Notation (JSON) [7] – a lightweight data-interchange format based on the data types of the JavaScript programming language. JSON documents are among the most popular formats for exchanging data on the web due to their simplicity and readability, making them easily understandable by both humans and machines.

JSON documents can store the virtual layout graph, encapsulating each room’s properties (such as middle point, size, and connections) and their connections. Room connection techniques – such as Portals, Corridors, or Central Hubs – are not stored because they can easily be recreated thanks to the [Scoring System](#). When importing a JSON document, the tool can parse the document to recreate the virtual layout graph, allowing the rest of the tool’s functionalities to operate as usual.

Using JSON files, virtual layouts can be easily serialized, preserving critical information. This document can then be shared with any tool instance, where it can be interpreted and the layout generated accordingly. Moreover, developers can create their own versions of a virtual layout and then extract the necessary information into a JSON document, which can be provided to the tool to enable natural navigation in VR and limited physical spaces.

3.5.3 VR Guardian Boundaries

VR headsets [70] have a built-in boundary system, often called "guardians."³ which establish predefined boundaries around users to prevent collisions with the surrounding environment. The system alerts users when they approach or cross these boundaries, reducing the risk of physical accidents. As shown in Figure 3.11, the guardian creates a boundary within the physical space and notifies users when they near these limits. Given the immersive nature of VR, users can easily lose awareness of their physical surroundings, increasing the risk of collisions. The guardian system mitigates this by maintaining spatial awareness, even during deep engagement with the VE.

Beyond safety, the guardian system is essential for adapting VE to fit within the user's physical space. VR headsets equipped with guardians store data about the dimensions of the physical area, which can be extracted and used to design virtual layouts suited to the user's environment. However, the immediate relevance lies in how these dimensions are crucial for calculating room or corridor strides, particularly in implementing corridors within the virtual layout. By leveraging this data, the tool ensures that VE are precisely tailored to fit any user's available space, enabling seamless exploration regardless of the area size.

The guardian system plays a key role in user safety during virtual exploration. Since the tool promotes walking as the primary mode of locomotion, users are more likely to approach the boundaries of their physical space. Guardian notifications help prevent collisions, especially when the virtual layout uses the full extent of the available space.

3.6 Summary

This chapter presented the tool requirements and how the tool can create or receive a virtual layout adaptable to small physical spaces, allowing for natural exploration in VR. The discussion also covered how the tool can seamlessly adapt the virtual layout in real

³VR Guardian <https://www.uploadvr.com/oculus-quest-guardian-size-increasing/>, Last Access: 2024

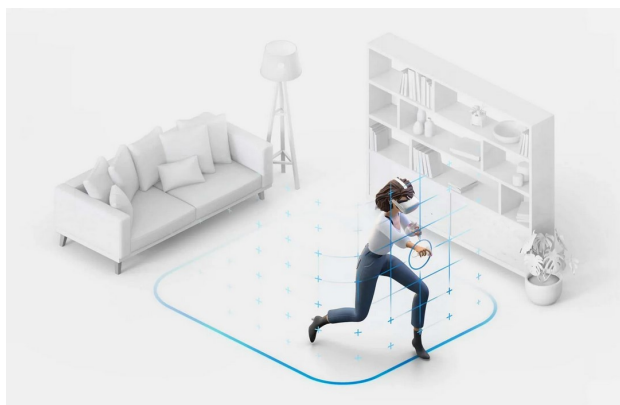


Figure 3.11: Example of how the guardian boundaries work in VR headsets.

time according to user preferences and adjust to various physical spaces. Three techniques – Portals, Corridors, and Central Hubs –, along with their underlying theory and challenges, were introduced to facilitate navigation and connect virtual layout rooms for walking in limited physical areas. Additional concepts were also discussed, including how the tool can export and import layouts and the role of the VR Guardian in adapting virtual layouts to physical spaces.

The next chapter will delve into the tool's implementation, detailing how the concepts discussed in this chapter were crucial in its development.

IMPLEMENTATION

This chapter focuses on the tool's implementation, detailing how the concepts from [Chapter 3](#) were realized in Unity Game Engine¹ and the challenges faced during development. The tool was built using Unity, a cross-platform game engine developed by Unity Technologies, and tested with the HTC VIVE Cosmos VR headset. The Unity asset packages were utilized: *OpenXR Plugin*, *MockHMD XR Plugin*, *XR Interaction Toolkit*, and *XR Plugin Management*. These packages provide essential VR scripts that streamline VR hardware integration, input handling, interaction systems, and device emulation. Using these packages, a player prefab was created in Unity to act as the anchor between the VR headset and Unity actions. This prefab handles rendering and transmitting the virtual scene to the headset. It includes utilities for tracking the player's head position, a virtual hand for interacting with the environment, and the *XR Rig* script from the *XR Interaction Toolkit*, which manages the VR headset's connection to Unity.

The chapter starts with [Section 4.1](#), which discusses the implementation of the *virtual layout generation* scripts, highlighting the [Simple Room Displacement \(SRD\)](#) algorithm, room creation process and scripts for exporting and importing virtual layouts. Next, [Section 4.2](#) covers the development of room connection techniques — Portals, Corridors, and [Central Hub \(CH\)](#) — and the integration of these techniques by the [Scoring System \(SS\)](#) to connect rooms, enabling navigation in limited physical spaces. Finally, [Subsection 2.2.2](#) explains the adaptive systems that support runtime adjustments to the virtual layout and also details features such as extracting the VR headset Guardian boundaries and the tool menu.

4.1 Virtual Layout Generation

This section details the core systems essential for creating virtual layouts, including the [Procedural Content Generation \(PCG\)](#) scripts responsible for generating the virtual layout, discussed in [Subsection 4.1.1](#). Next, the room creation scripts, which play a role in shaping the individual rooms within the layout, are explored in [Subsection 4.1.2](#). Finally,

¹Unity Game Engine Website <https://unity.com/>, Last Access: 2024

the scripts that ensure the virtual layout's persistence, allowing the saving and loading of layouts, are covered in [Subsection 4.1.3](#).

4.1.1 Virtual Layout Creation

The first step to create a virtual layout is to define key parameters that dictate its generation. These parameters include:

- **Radius:** Defines the radius within which rooms are placed. A larger radius results in more dispersed room placement (developer parameter).
- **Number of Rooms:** Specifies the total number of rooms in the layout.
- **Max Room Scale:** Sets the maximum room size, with larger rooms requiring larger strides.
- **Min Room Scale:** Defines the minimum room size, with smaller rooms having strides closer to the physical area's dimensions.
- **Import Layout:** Specifies whether a pre-existing layout is used.

Once these parameters are set, the script generates the desired number of rooms. The specifics of room creation will be detailed in [Subsection 4.1.2](#). The Unity physics engine is accelerated to speed up room placement and avoid collisions. If two rooms intersect, they are pushed apart, and accelerating the physics engine ensures quicker placement.

After room creation, their midpoints are passed to the [Delaunay Triangulation \(DT\)](#) script², which processes the midpoints into triangles to form a graph representing the layout. This graph includes edges and nodes, with nodes containing room details such as position, scale, rotation, and type (square or rectangle).

Next, the [Minimum Spanning Tree \(MST\)](#) script is applied to the graph, selecting the shortest edges to connect all rooms. Starting from a random node, the algorithm selects the minimum distance edge and adds the corresponding node to the graph, repeating the process until all nodes are visited. As shown in [Figure 4.1](#), an example with five rooms demonstrates how the [DT](#) output (green lines) and the [MST](#) output (blue lines) form connections between rooms without overlap.

The next step involves creating doors to represent room connections. Using Unity's raycasting feature, the process includes:

- **Iterating Through Graph Edges:** Each edge is processed.
- **Raycasting Between Room Midpoints:** A raycast is shot between connected rooms.

²Credits to Oskar Sigvardsson <https://github.com/OskarSigvardsson/unity-delaunay.git>, Last access 2024

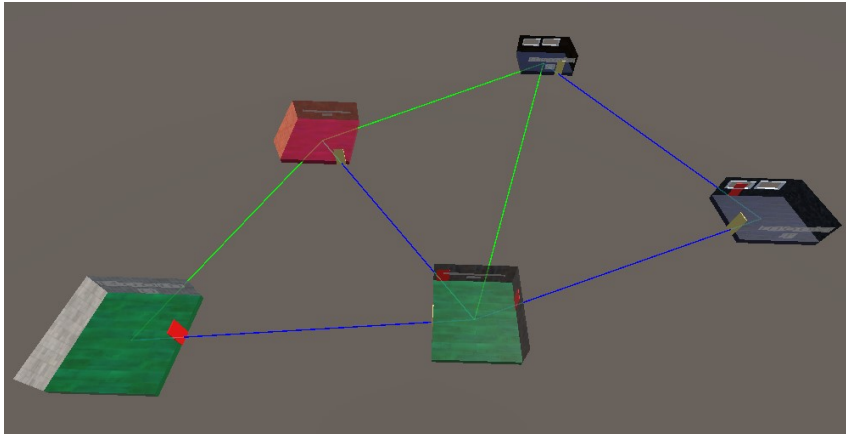


Figure 4.1: Example of the generation of the initial room. The lines represent all possible connections of the rooms, and the blue lines represent the definitive connection. The rooms have yellow and red rectangles that represent the doors that connect the virtual rooms.

- **Detecting Collider Hits:** The raycast detects the room's collider, recording the hit position.
- **Instantiating Doors:** Doors are placed at hit points to represent room connections.
- **Mesh Cutting:** The *Cut Mesh* script is used to cut the room wall, forming the door shape.

This process is repeated for all edges, ensuring each room connection has a corresponding door.

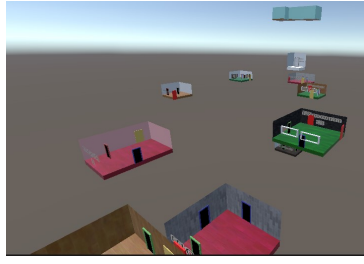
After this, room collisions are deactivated to prevent player interaction, and the stride for each room is calculated based on its floor scale and the dimensions of the physical area, implementing the formula 3.4.

Rooms are also added to the *Portal Position Verifier*, which checks the validity of portal placement. This script will be further detailed in Subsection 4.2.2. Room strides are added to a script that manages the user's stride based on colliders tied to room floors. As users enter a collider, their stride adjusts accordingly.

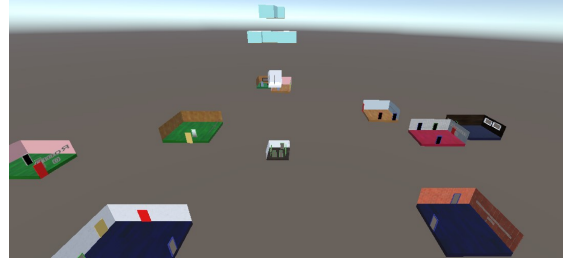
Lastly, a room outside the main layout, matching the physical space's dimensions, serves as the starting point for the user. A portal connects this room to the first layout room, managed by the virtual layout generation scripts since it is excluded from the main graph.

With the layout complete, the **Scoring System (SS)** can be initiated. An overview of the final layout, including implemented room connection techniques, is shown in Figure 4.2.

Cut Mesh Script ensures that room walls are properly cut to accommodate doors at runtime, as shown in Figure 4.3. In Unity, walls are made of standard planes, and the script precisely cuts these planes to fit the door shapes. This process is crucial because



(a) Layout Overview Example 1



(b) Layout Overview Example 2

Figure 4.2: Layout Overview Examples.

the corridor technique replaces original doors with corridor entrances. Without this cut, users would be unable to see the corridor and would need to pass through a wall, disrupting immersion. The script automatically modifies the wall mesh to prevent this issue. The script runs after the tool starts, as door positions are layout-dependent and vary dynamically.

As illustrated in [Figure 4.4](#), the process starts by retrieving the wall closest to the door's raycast hit point (Step **A**). In Step **B**, the script calculates the door dimensions based on the hit point, which varies depending on the door prefab. Step **C** shows the wall being divided into three sections: the left, right, and top of the door. Since the door connects to the bottom edge of the wall, no bottom section is needed. To create these sections, the script calculates the $zPos$ of the door extremities using the following formula:

$$zPos = \begin{cases} doorPosition_z + \frac{DoorWidth}{2}, & \text{if LeftWall} \\ doorPosition_z - \frac{DoorWidth}{2}, & \text{if RightWall} \end{cases} \quad (4.1)$$

where $doorPosition_z$ is the z position of the raycast hit in the wall, indicated in step **(A)** by the blue circle, $DoorWidth$ is the width of the door prefab. One important note is that all calculations are done or applied to the z -axis because, in the Unity scene, that is the axis that moves alongside the wall. After finding the $zPos$, the next step is to find the middlepoint of the left and right walls cuts (WC) using the following formula:

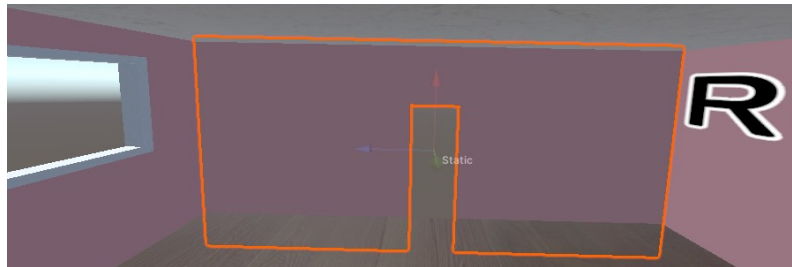


Figure 4.3: Example of a door cut in a room wall.

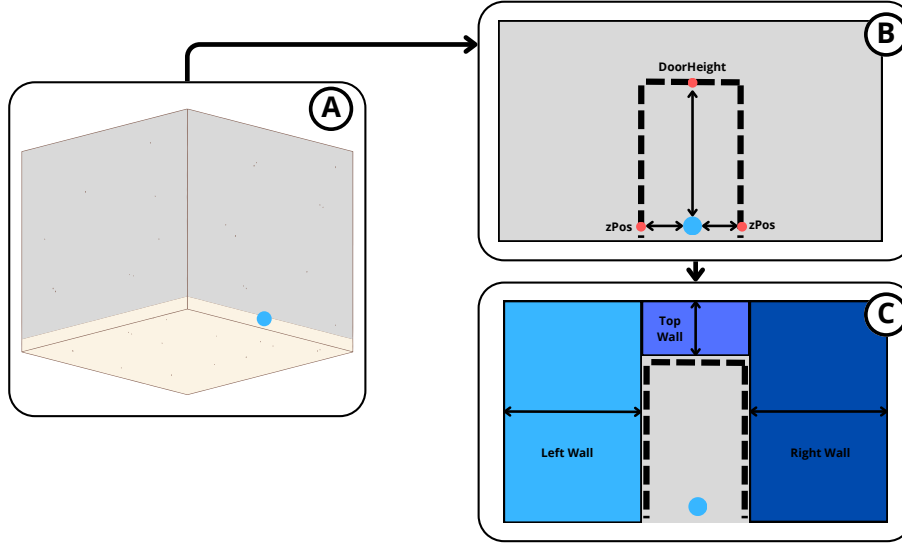


Figure 4.4: Demonstration of how a door cut is made onto the room wall mesh. Step A shows the raycast hitpoint in the room. Step B shows the wall closest to the hitpoint and the door prefab dimensions on the wall. The $zPos$ represents the points of the door edges. Step C shows the three cuts done to the wall. These cuts meshes are then combine to create the new wall.

$$WC_{MP} = \begin{cases} \frac{(zPos + \frac{PlaneSize}{2})}{2}, & ifLeftWall \\ \frac{(zPos - \frac{PlaneSize}{2})}{2}, & ifRightWall \end{cases} \quad (4.2)$$

where $PlaneSize$ is the size of the unity plane object, which is ten times bigger in scale compared to the other unity objects. With the middlepoint found for both left and right wall cuts, the last step is to calculate their respective width, which can be found using this formula:

$$WC_{width} = \begin{cases} |zPos + \frac{PlaneSize}{2}|, & ifLeftWall \\ |zPos - \frac{PlaneSize}{2}|, & ifRightWall \end{cases} \quad (4.3)$$

For the top wall cut, although the logic behind it is the same, the formulas differ. While there still is the need to find the edge position of the door frame ($zPos$) and determine the middle point and width of the top wall cut, the height also needs to be found. The top wall cut height is the difference between the door height and the original wall height. To find the necessary measures for the top wall cut (TWC), the following formulas can be used:

$$Width = DoorWidth \quad (4.4)$$

$$Height = \frac{(PlaneSize - DoorHeight)}{2} \quad (4.5)$$

$$TWC_{MP} = DoorHeight + height \quad (4.6)$$

With all the necessary measures to create the wall cuts, the position and scale of each wall cut can be defined as follows:

- **Left Wall:** $Position = (0, doorPosition_y, WC_{MP})$, $Scale = (1, 1, \frac{WC_{width}}{PlaneSize})$.
- **Right Wall:** $Position = (0, doorPosition_y, WC_{MP})$, $Scale = (1, 1, \frac{WC_{width}}{PlaneSize})$.
- **Top Wall:** $Position = (\frac{TW_{C_{MP}}}{2}, 0, doorPosition_z)$, $Scale = (\frac{Height}{PlaneSize}, 1, \frac{Width}{PlaneSize})$.

The three wall cuts can be found and created using these formulas and their result, as seen in step (C).

Each time a door is created, the closest section of the room wall is divided into three parts, and this process occurs immediately after the door is placed. In cases where a new door is added to a wall that already has a door, the closest wall section to the new door will be one of the pieces from the previously cut wall. This allows multiple doors to be accurately placed on the same wall, with the cuts precisely reflecting the positions of all door frames. Once all doors have been created and their corresponding wall pieces generated, the *Cut Mesh* script processes each main room wall. The wall segments are passed to a method within the script, which combines these segments into a single mesh. This method ensures the new mesh retains the same properties as the original room wall. To achieve this, the main wall's position, rotation, and scale are normalized, with the original values stored to guarantee precise calculations and prevent any unintended transformations. After the wall segments are combined into the main wall, the segments are deleted, and the original values are restored to the main wall, resulting in the final configuration shown in [Figure 4.3](#).

4.1.2 Room Creation

The rooms are a central component of the virtual layout, and their creation process is detailed here. From the outset, a factory design pattern³ was chosen to manage room creation. This pattern allows each room type to be handled by its own factory, ensuring controlled creation and customization. Currently, the tool supports two room types: square and rectangular, each available with or without windows. This setup allows developers to instantiate room templates and modify them as needed while also providing flexibility for future expansions, such as new room types or updates to existing ones.

The process works as follows: once the [Simple Room Displacement \(SRD\)](#) script selects a random midpoint for room placement, this point, along with a randomly chosen room type and the defined maximum and minimum room scales, is sent to the *Room Factory* script. The factory, shown in [Figure 4.5](#), initializes sub-factories for each room type. Depending on the selected room, the corresponding sub-factory handles the creation, choosing from prefabs specific to that room type.

³Factory Design Pattern <https://refactoring.guru/design-patterns/factory-method>
<https://refactoring.guru/design-patterns/factory-method>, Last Access 2024

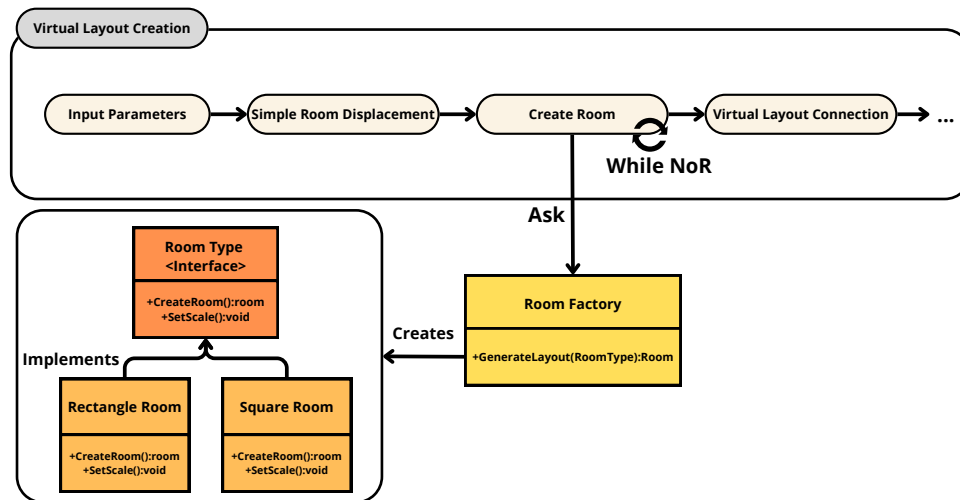


Figure 4.5: Room factory class structure and methods. This example shows the process of when a new room needs to be created.

After selecting a prefab, the room's scale is randomly determined within the provided range. While interior decoration is outside the scope of this thesis, the design allows for easy integration of decorations in the future, supported by sub-factories that can be expanded with additional scripts for more complex room configurations and interiors.

Following instantiation, the main factory assigns a unique name to each room and updates the wall materials from a predefined list. In Figure 4.6 are illustrated some examples of rooms created by the factory. Although the main factory manages general decoration, specific material control can be delegated to sub-factories if needed. The room is then returned to the [SRD](#) algorithm.

A key feature of this implementation is the use of *scriptable objects* to manage room prefab variations. These objects store the room type and its associated prefab, allowing the sub-factories to search the Unity Resources folder — a folder for essential assets that persist in build versions of Unity— to locate the specific *scriptable objects* matching their room type.

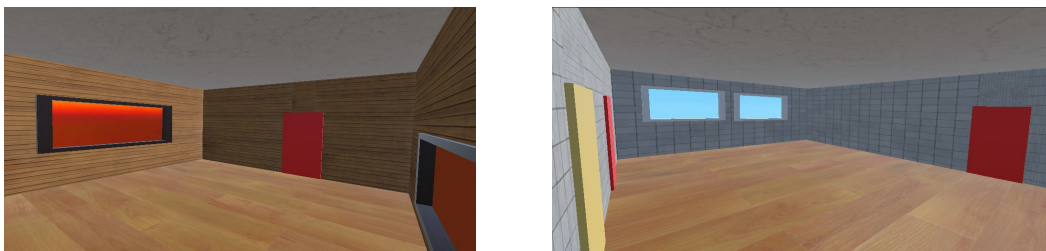


Figure 4.6: Examples of rooms created by the room factory script.

4.1.3 Virtual Layout Persistence

As mentioned in [Subsection 3.5.2](#), JSON is an ideal format to simplify importing and exporting content. To facilitate this, a *Layout Persistence* system was developed that allows virtual layouts to be saved and loaded using JSON files. [Figure 4.7](#) illustrates the process of exporting a virtual layout. The layout is primarily represented as a graph containing rooms and their connections, which can be easily serialized into JSON. Other relevant data may also need to be saved within the file.

To manage the necessary data, a data class named *Layout Data* was created to store all essential information related to the virtual layout. This class allows any script that needs to save data to the JSON file to easily create an instance of *Layout Data* and populate it with the necessary details. The instance is then passed to a script called *Persistence Manager*, which serializes the data into a JSON file and saves it in a local folder that Unity uses to store data.

Exporting a generated virtual layout starts with pressing a button in the Unity editor — *Export Layout* — or in the menu — detailed in [Subsection 4.3.1](#). This triggers the creation of a *Layout Data* instance containing the current layout information, as shown in step (1). In step (2), the *Virtual Layout Creation* scripts generate a *Layout Data* instance that includes the virtual layout graph. This data is passed to the *Layout Persistence Manager*, as depicted in step (3). Finally, in step (4), the *Persistence Manager* serializes the data into a JSON file and saves it in a local folder, which Unity uses to store important tool-related information.

The system also supports importing layouts, as shown in [Figure 4.8](#). When the tool starts, the *Persistence Manager* loads all JSON files related to virtual layouts from the local Unity folder, as indicated in step (1). The data from these files is deserialized into a list of *Layout Data* classes, as shown in step (2). This allows scripts to access stored information

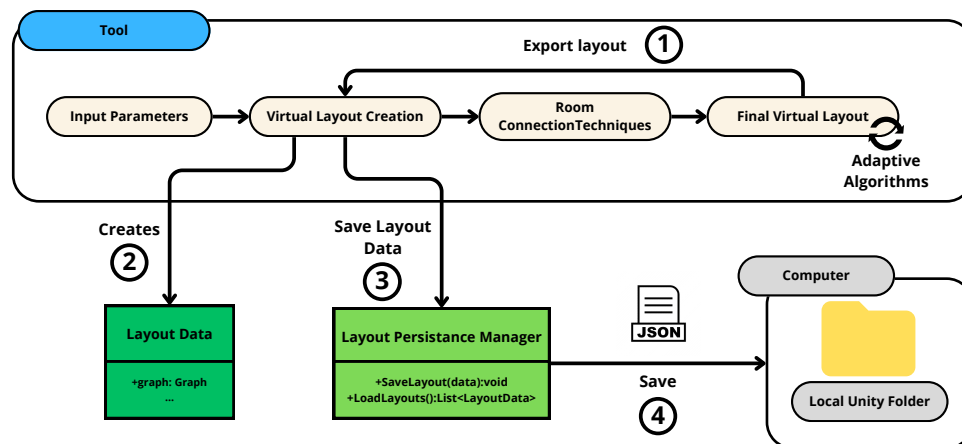


Figure 4.7: Exporting virtual layout process. First the export command is called when the layout is finished. Then the *Virtual Layout Creation* scripts create a new *Layout Data* to store the layout graph. The scripts then pass this *Layout Data* to the *Layout Persistence Manager*. This will serialize the data into a json file and store it in the computer.

and restore previously saved layouts. Step (3) demonstrates how the *Persistence Manager* makes this list available as an input parameter, enabling developers to import existing layouts by selecting from the list of loaded *Layout Data*. This list is also displayed in the tool menu for users to choose. The *Virtual Layout Creation* scripts then read the graph stored in the *Layout Data* class and instantiate rooms with their corresponding connections and information based on the imported graph.

External Virtual layouts: While this framework works well for layouts created by the tool, it also needs to accommodate layouts created externally by developers. In this regard, developers must consider several aspects to ensure the validity of the virtual layout for use with the tool:

- Prefabs used by the room factory should serve as room templates. These prefabs contain the necessary scripts and information required by the room factory. Not using these prefab templates can lead to unexpected issues within the tool, so it is recommended that they be utilized to minimize potential problems.
- The virtual layout's graph is crucial for the tool's operation, and it must be emulated when manually creating a layout. A script called *Extract Current Scene* can be used for this purpose, allowing developers to specify which rooms are nodes of the graph and how they connect to other rooms (the edges of the graph).
- Manually created rooms must be turned into prefabs and stored in Unity's Resources folder to enable their instantiation in build versions of Unity.
- Manual placement and naming of virtual layout doors need to follow specific rules. Every door created by the tool follows a naming convention: the name of each door

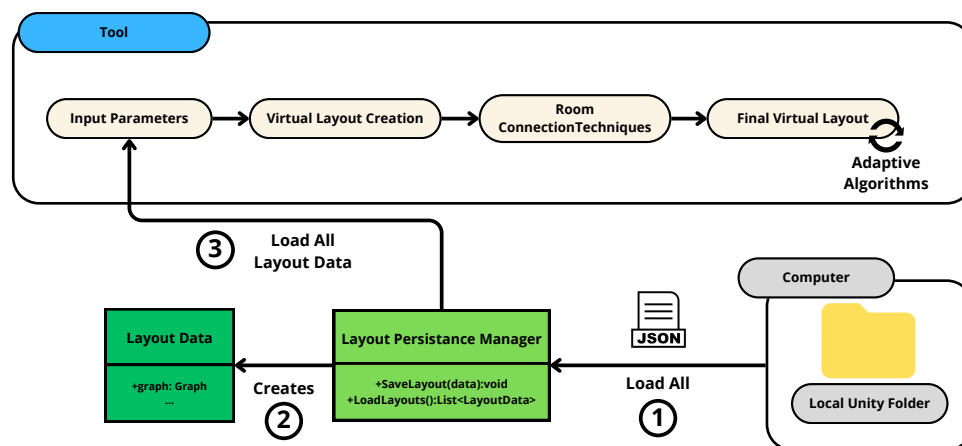


Figure 4.8: Importing virtual layout process. First when the tool is started, the *Layout Persistence Manager* loads all JSON files related to virtual layouts from the computer. Then for each JSON, it deserializes it into a *Layout Data*. Lastly it provides the list of *Layout Data*s to the input parameters to be selected.

is "Door," followed by the name of the room in which it is located, then "\$\$" (a delimiter used by the tool to split strings), and finally the name of the room to which it connects. This step is crucial because the corridor technique needs to identify and connect to the doors of the rooms. The technique will not function correctly if the door names do not follow these rules.

These are the main aspects to consider when manually creating virtual layouts for the tool. Once the layout is created and the necessary values are filled in the *Extract Current Scene* script, the *Layout Data* class can be serialized and saved as a JSON file. The graph created in this way includes a special value indicating that it was created externally to the tool. This value is essential because when importing external virtual layouts, the scripts must load all the prefabs stored in the Resources folder that are required by the external virtual layout and then instantiate them. [Figure 4.9](#) shows an example of an external layout created to be used by the tool.

4.2 Room Connection Techniques

This section will detail how each room connection technique – Portals, Corridors, Central Hubs – was implemented. Alongside these techniques, the [Scoring System \(SS\)](#) implementation will also be explained.

4.2.1 Scoring System

The [SS](#) utilizes *scriptable objects* to store its main input parameter values and associated settings, as detailed in [Table 3.1](#). This approach simplifies both the creation of new parameters and the adjustment of existing ones. To effectively manage these parameters, the [SS](#) includes a variable for each *scriptable object*, initialized when the tool starts. Each

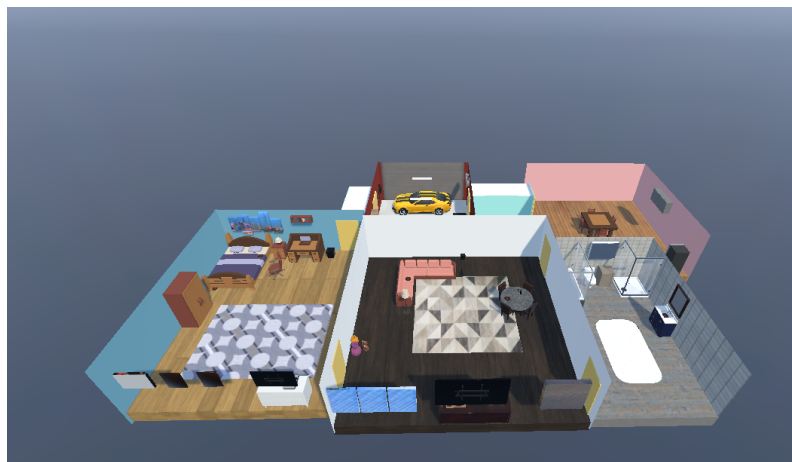


Figure 4.9: External virtual layout example.

scriptable object corresponds to the values in a specific column of Table 3.1. For each sub-parameter under the main categories—*realistic factor*, *layout size*, and *layout preservation*—a scriptable object was created to hold the values for each room connection technique.

The SS begins by receiving the graph of the virtual layout and employs a basic implementation of the *State*⁴ design pattern. It operates through three main stages:

- **Start State:** In this state, the technique scores obtained through the *scriptable objects* are first refined and adjusted before calculating each technique's probability. Each technique score is the sum of all the main SS parameters and is calculated using Equation 3.12. These scores can be negative, so after calculating each technique score, a shift is applied to ensure all scores are positive. This shift involves finding the smallest technique score and, if it is negative, adding its absolute value to all the technique scores. Once all the technique scores are calculated, Equation 3.13 determines each technique's probability. With the probabilities calculated, the next state can commence. It is important to note that with custom parameters, the user chooses each technique score, directly influencing each technique's probability.
- **Calculate Room Connection State:** With the probabilities determined, this state selects a technique for each edge connecting two rooms. The graph edges are iterated over, generating a random value for each iteration. This random value is then compared against the technique probabilities to find the closest match, which is then chosen to connect the two rooms associated with that edge. Once a technique is selected, the SS invokes the specific method, preparing the necessary information for the script responsible for implementing the technique. The details of what each technique implementation requires will be discussed in their respective subsections. After all edges are processed, the final state of the SS is triggered.
- **End State:** The final state of the SS manages the invocation of the adaptive algorithms, signaling them to begin. These algorithms will be further detailed in Subsection 4.3.2. When the SS finishes, the virtual layout is ready to be navigated, and Figure 4.10 shows some examples of the user perspective while exploring the finalized virtual layout with the techniques implemented.

4.2.2 Portals

Two important points must be addressed before discussing the implementation of the *automatic portal placement* script. First, the portals used to connect rooms in this thesis were adapted from *Portals VR* by Daniel Lochner⁵. Lochner's work significantly accelerated the

⁴State Design Pattern, <https://refactoring.guru/design-patterns/state><https://refactoring.guru/design-patterns/state>, Last Access 2024

⁵Portals VR Github, <https://github.com/daniellochner/portals-vr><https://github.com/daniellochner/portals-vr>, Last Access 2024

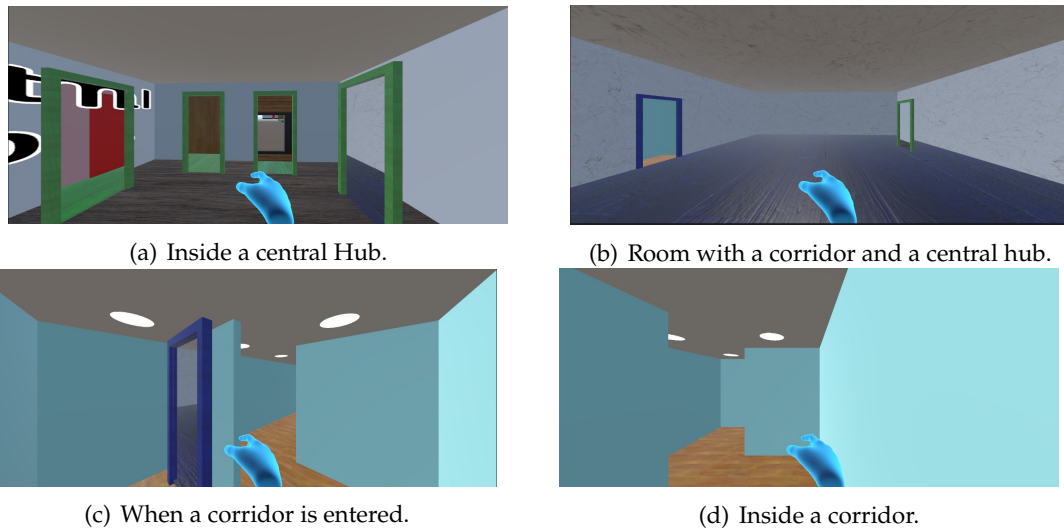


Figure 4.10: Examples of the user Perspective when exploring a virtual layout.

tool's progress by providing functional portals that required only minor adjustments to integrate with the tool's scripts. Second, all room connection techniques in the tool utilize portals, though each technique implements them differently. For instance, the portal technique directly links two rooms via a portal, while the corridor technique uses portals to connect rooms to corridor entrances. This use of portals allows rooms to be separated in the editor view—facilitating debugging—while appearing connected and overlapped in the game view, creating the effect of Impossible Spaces.

With these points in mind, the script for *automatically placing portals* between two rooms works as follows: First, the script receives two room game objects from the [SS](#). Using these game objects, the script extracts the dimensions of both rooms and calculates their intersection area. To define this intersection area, the script first determines the borders of each room along the x and z axes, from which the intersection borders are calculated. These borders are slightly shrunk to ensure the portals are not placed too close to the intersection walls. Once the intersection border is defined, a random point within this area is selected for portal placement. Before creating the portals, this position is passed to a *portal position verifier* script, which checks the validity of the chosen location. After verification, the portal prefab is instantiated at the selected position, and the portals are placed inside the rooms according to the method outlined in [Equation 3.5](#). [Figure 4.11](#) illustrates some examples of portal placement between rooms.

Portal Position Verifier is a script responsible for managing the placement of portals within the rooms of the virtual layout, ensuring that new portals do not overlap with existing ones. The script uses a dictionary where each room acts as a key, and the value is a list of portal information—a dedicated data class that stores the portal's position and the technique to which it belongs.

When a new portal is about to be placed, the proposed position and the room are

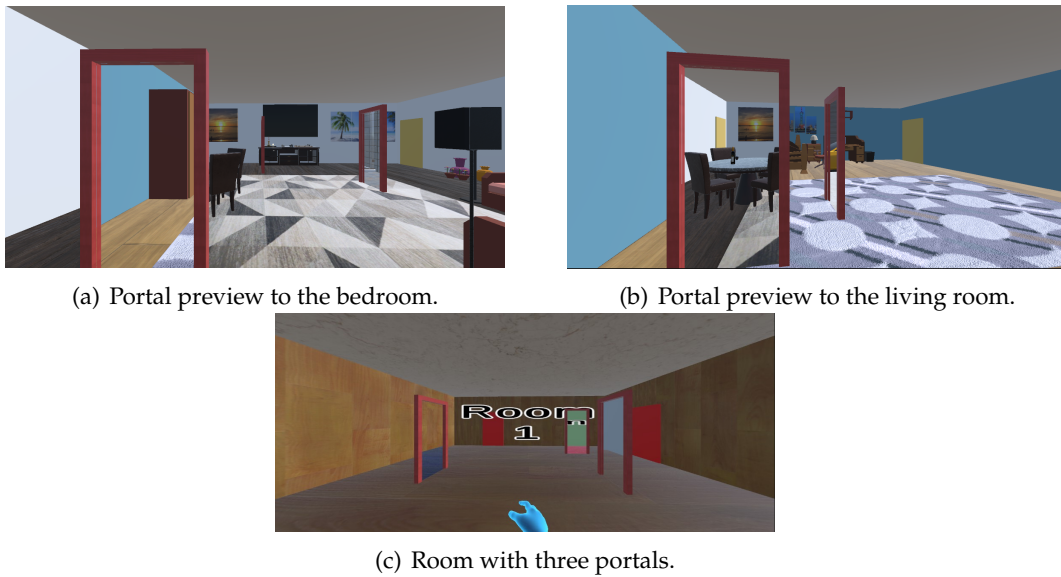


Figure 4.11: Examples of the user perspective when exploring a virtual layout only connected by portals.

passed to the *Portal Position Verifier* script. The script then iterates through the list of established portal positions in the corresponding room to check if the new position is too close to any existing portals. This verification process occurs simultaneously for the portal positions in both rooms involved. The position is considered valid if no existing portal position overlaps with the new one. Conversely, if the new position overlaps with any portal in either room, the position is deemed invalid, and a new one must be determined. Once a valid position is found, the dictionary is updated with the new portal's position and associated rooms.

4.2.3 Corridors

To implement the *Automatic Corridor Placement* script, a corridor node prefab was first created, as shown in Figure 4.12. This prefab represents a corridor node and is designed to be connected together, allowing the corridor to take on various shapes by selectively deactivating the inner walls. The size of the corridor nodes can be adjusted by scaling them up or down to fit the specific requirements of the layout. For this tool, a scale of (1.5, 1.5, 1.5) was chosen, providing sufficient space for navigation without causing a claustrophobic effect while still being compact enough to fit within the limited physical area available for corridor placement.

The node prefab includes portal placeholders on each wall to facilitate placing entrance and exit portals within the corridor. Additionally, the floor of the corridor node contains a script that detects when a player enters, adjusting the player's stride to the appropriate value for that particular node.

The *Automatic Corridor Placement* script is responsible for creating corridors that connect two rooms within the constrained physical space. This process begins once the SS provides



Figure 4.12: Example of a corridor node prefab.

critical data, such as the start and end positions of the corridor and the rooms that need to be connected. Algorithm 3 outlines how the [SS](#) calculates and passes this information to the *Automatic Corridor Placement* script. Additional details, such as the difference between the original door positions and their corresponding start/end points within the corridor, as well as the orientation of the doors, are provided to guide corridor generation.

The process begins by generating a graph representing the physical space, divided into a grid based on the size of corridor nodes. This graph ensures the corridor fits within the space, with each node representing a potential corridor position. Invalid nodes are removed to avoid incorrect placement, such as starting or ending a corridor in the wrong location. The *A* pathfinding* algorithm then determines the shortest path from the start to end node. Corridor node prefabs are instantiated along this path, with each node's stride calculated and assigned to adjust the user's movement. Interior walls are deactivated as needed to keep the corridor unobstructed.

Special attention is given to the start and end nodes, where portals must be placed. The portals are positioned in the corresponding placeholders in the corridor node prefab, with their orientation aligned to match the inverse direction of the room doors. For example, if a room door is on the left wall, the corridor portal will be placed on the right wall to ensure a proper connection. The offset provided by the [SS](#) is applied to ensure accurate alignment with the room door in physical space. These portal positions are then registered with the *Portal Position Verifier* script to prevent portal placements too close to the corridor portals.

The instantiation of corridor nodes continues until the entire path is built. Once all nodes are in place and properly connected, the corridor is ready for use. Several examples of corridors generated by this script are shown in [Figure 4.13](#), showcasing the flexibility and functionality of the *Automatic Corridor Placement* system.

A* Algorithm: To implement the *A** algorithm, the website [A*AlgorithmExample](#) was used as a reference. The first step involved modifying the existing *graph* script, specifically

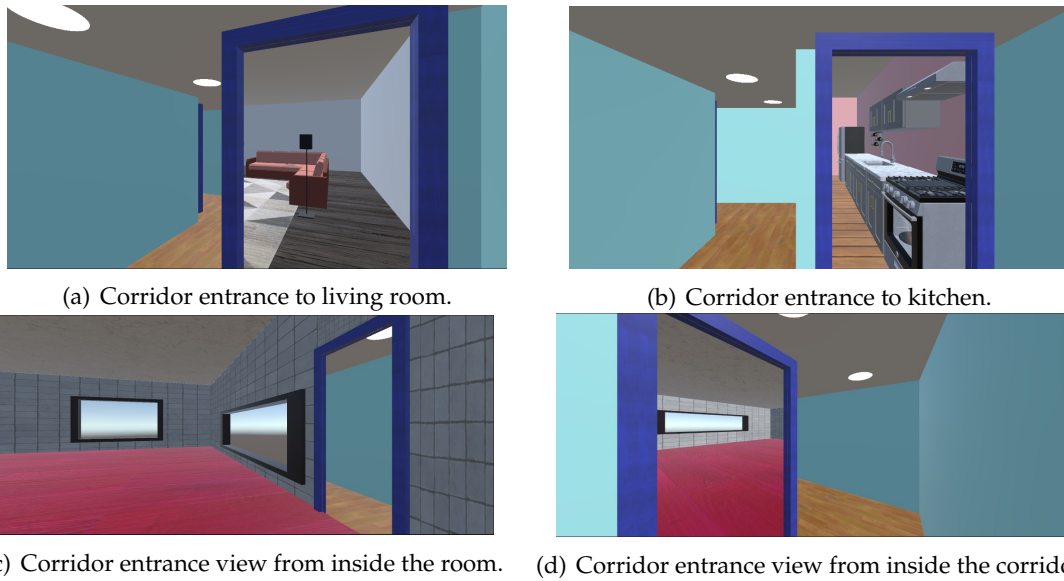


Figure 4.13: Examples of the user perspective when exploring a virtual layout only connected by corridors.

the *node* class, to store additional values necessary for the A^* algorithm: parent node, G , H , and F values, which are defined as follows:

- **Parent Node:** The parent node is the node that comes before the current node in the path. It is used to find the final path.
- **G :** The cumulative path length from the start node to the current node.
- **H :** The straight-line distance from the current node to the end node.
- **F :** The estimated total distance or cost, calculated as $F = G + H$.

With each node now equipped to store these values, the implementation of the algorithm can proceed. Two lists are required for nodes to visit, ordered by their respective F values, and another for already visited nodes.

The algorithm checks the node with the lowest F value from the to-visit list. If this node is the end node, the final path is traced back from it, which will be detailed later. If not, the node is removed from the to-visit list and added to the visited list.

Next, the neighbors (connected nodes) of the selected node are examined. For each neighbor:

- If the neighbor is in the visited list, it is skipped.
- Otherwise, the neighbor's G score is calculated. If the neighbor is not already in the to-visit list, it is added, and its G , H , and F values are set. Additionally, the current node is stored as its parent node.

- If the neighbor is already in the to-visit list, but the newly calculated G score is lower than its previous G score, the neighbor's G value and parent node are updated to reflect the better path.

This process repeats, selecting nodes from the to-visit list based on their F values until the end node is reached. Once the end node is reached, the final path is constructed by tracing back through the parent nodes, starting from the end node and continuing until the start node is reached. As each node is traversed, it is added to the final path list, ultimately yielding the complete path from start to end.

4.2.4 Central Hubs

To implement the *Automatic Central Hub Placement* script, two prefabs were created: one with four portal placements (Figure 4.14 (d)) and another with eight (Figure 4.14 (b)). These prefabs serve as **Central Hub (CH)**, allowing multiple rooms within the virtual layout to be accessed through them. In layouts with a large number of rooms, multiple central hubs may be instantiated. In such cases, an elevator (Figure 4.14 (a,c)) is positioned in the middle of the central hub, providing a way to move between different hubs.

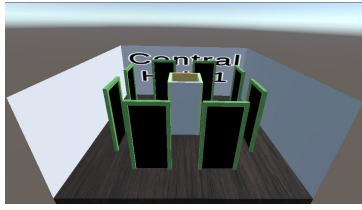
By utilizing these prefabs, the *Automatic Central Hub Placement* script can focus on populating each prefab with portals that connect to various rooms in the virtual layout.

With the prefabs created, the script now requires information about the rooms that need to be connected. This information, provided by the **SS** and detailed in [Subsection 3.4.3](#), includes a list of rooms to be connected to a **CH**. After receiving the rooms, the first step is determining the type of **CH** is required and whether it needs four or eight connections. Based on this decision, the corresponding **CH** prefab is instantiated along with its associated stride, using [Equation 3.4](#). The instantiated **CH** is then added to a list of currently instantiated hubs, which will be relevant later.

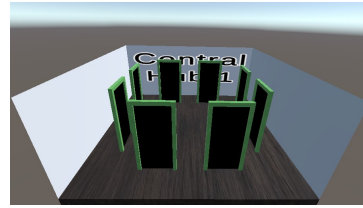
Next, the script connects the rooms by selecting available portal positions on the hub and checking their validity. If a position is invalid, another is chosen. If no valid positions remain, the room is added to a list of unconnected rooms. When a valid position is found, portals are instantiated in both the hub and the room, with their locations set using [Equation 3.11](#). This process continues until all rooms are connected. Unused portal positions are saved for future connections.

Once all hubs are connected to their rooms, the **SS** interconnects the hubs. The elevator prefab, illustrated in [Figure 4.15](#), is activated during this phase. The elevator, equipped with buttons for ascending and descending, teleports users between hub levels. As users anticipate movement while inside the elevator, they do not perceive the **VE** change, reducing cybersickness.

Finally, the script revisits unused portal positions and attempts to connect any unlinked rooms. This process continues until all hubs are interconnected and unused portals are filled. If any rooms remain unconnected, an extra hub is instantiated to link them, ensuring



(a) Big central hub with eight connections and the elevator.



(b) Big central hub with eight connections.



(c) Small central hub with four connections and the elevator.



(d) Small central hub with four connections.

Figure 4.14: Example of the different central hubs prefabs, with and without the elevator.

complete connectivity to all selected rooms. Each portal placement is also recorded in the *portal position verifier* script.

4.3 Adaptive Systems

The final aspect of the implementation involves the *adaptive systems*, which are a set of scripts, each designed to enhance interactivity and improve the user experience during virtual layout navigation. These systems automatically and gradually adjust virtual layouts to facilitate seamless user navigation and prevent disruptions that could break immersion.

The *adaptive systems* also include a menu, as detailed in [Subsection 4.3.1](#), allowing users to edit and adapt the virtual layout in real time. Through this menu, users can modify room connection techniques, tweak various parameters, or even generate entirely new

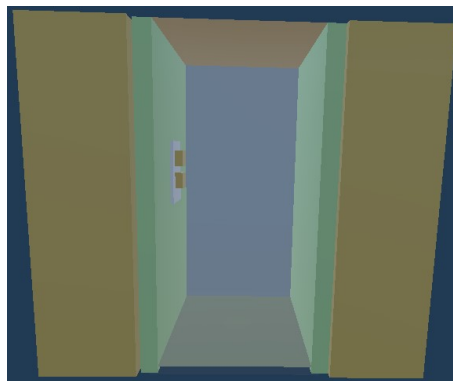


Figure 4.15: Elevator prefab demonstration.

layouts while the application is running, as explained in [Subsection 4.3.2](#).

Additionally, the *adaptive systems* are responsible for extracting the VR headset's guardian boundaries and providing this information to other scripts to ensure the virtual layout generation aligns with the available physical space. This process is discussed further in [Subsection 4.3.3](#). Finally, this section outlines additional algorithms that monitor user actions and trigger adaptive functionalities during runtime, with detailed implementation covered in [Subsection 4.3.4](#).

4.3.1 Tool Menu

To implement the menu, a virtual hand mapped to the VR headset controls was created for user interaction. The *XR Interaction Toolkit* package provided key scripts such as *XR Controller*, *XR Ray Interactor*, and *XR Line Visual*, enabling the creation of this virtual hand. Unity's new input system was used to map controller buttons to actions within the scripts.

The menu UI, shown in [Figure 4.16](#), offers three main features: editing *SS* parameters to modify room connection techniques, altering the virtual layout (e.g., number of rooms, sizes, and connections), and recentering the layout.

The menu also needs to display the current layout configurations. For example, the menu will reflect the current room connection techniques or layout settings, allowing users to make necessary changes.

4.3.2 Virtual Layout Adaptability

Starting with the adaptability of the virtual layout, the menu offers two options for modification: adjusting *SS* parameters, which affect room connection techniques, and directly changing the virtual layout parameters.

Edit Room Connections: In this menu, users can modify *SS* parameters, including *Layout Preservation*, *Layout Size* (tied to the number of rooms), and *Realistic Factor*. Users may also enable custom parameters and manually set probabilities for each technique. Upon confirmation, the menu sends the data to the *Adapt Scoring System* script, which



Figure 4.16: Menu pages.

updates the [SS](#) parameters and restarts the system to create the new room connection techniques.

To ensure the [SS](#) is adaptive, several changes were implemented. A dictionary tracks each room's portal game object (portal, corridor, or central hub). The dictionary is cleared when new techniques are requested, and all portal game objects are deleted from the layout. Key variables are reset, and scores are recalculated based on new parameters. If custom parameters are provided, the new probabilities do not need to be calculated. This setup allows runtime changes to room connections with no usage limit.

Edit Layout: This menu allows users to adjust the number of rooms, their size range, and [SS](#) parameters. Users can also import or export layouts. Imported layouts are recreated with the new size and [SS](#) parameters unless created manually (external virtual layouts), in which case only the [SS](#) is applied.

Upon confirmation, the menu signals the *Tracker* script, which will be detailed later in [Subsection 4.3.4](#), to guide users to a separate starting room, seen in [Figure 4.17](#). This serves as a waiting area while the virtual layout is recreated, preventing the user from witnessing the layout's destruction and preserving immersion. Once the user is in the waiting room, the *Virtual Layout Creation* script destroys the previous layout, resets variables, and either recreates the imported layout or generates a new one based on the new user's inputs. Afterward, the [SS](#) restarts to create room connection techniques for the new layout, and a portal appears in the waiting room for the user to enter and explore the updated virtual environment.

4.3.3 Guardian Boundaries

As previously mentioned, *Guardian Boundaries* is a feature present in every VR headset, serving as a failsafe to prevent users from leaving the designated play area. This feature is extremely useful because it stores the dimensions of the physical area, which are necessary for adapting the tool to any physical space. To leverage this, a script called *Guardian Boundaries* was created to retrieve these dimensions from the headset and store them



Figure 4.17: Example of user perspective inside the starting room. This rooms does not belong to the virtual layout.

for use by other scripts, such as the *Virtual Layout Creation* scripts and room connection techniques.

To obtain information about the guardian dimensions, a list of *XRInputSubsystem*—a script from the *Unity Engine XR* package—was created and initialized. The *SubSystem Manager* instances were then passed to this list of *XRInputSubsystem*. The *SubSystem Manager* is a script that manages subsystems providing additional functionality through plugins. One such functionality of the *XRInputSubsystem* is the ability to retrieve the boundary points of the guardian. These boundary points can then be applied to the starting/waiting room to ensure they match the scale and size of the physical area while also storing its dimensions. Any script needing access to the guardian boundaries can then request this information from the *Guardian Boundaries* script, which returns the scale of the starting/waiting room.

4.3.4 Additional Algorithms

The final feature of the *adaptive systems* is the tracker algorithms, consisting of scripts that work in the background while the user explores the virtual layout. These scripts provide essential information to other components and enhance navigation within the virtual layout. Two types of tracker algorithms are implemented: the *tracker script* and the *offset tracker script*. The *Tracker script* tracks the user's position within the virtual layout and guides the user to the waiting room to initiate the recreation of a new layout. The *offset tracker script* recenters the entire virtual layout discreetly, ensuring the user remains unaware of the adjustment.

Tracker Script: The first script in the *trackers* algorithms is the *Tracker* script, whose purpose is to track the user's current position within the virtual layout. It achieves this by storing the room information when the user enters a room's collider, and their stride is adjusted, signaling the *tracker* script to record the current room. Knowing the user's exact location provides significant advantages, particularly for other *adaptive systems* such as *edit layout*, *recenter*, and the *offset tracker*. The *tracker* script is also crucial when recreating entirely new virtual layouts during runtime. When the user submits new parameters via the menu to create a new layout, the *tracker* script places a portal leading to the waiting room in the current room where the user is located. Once the user enters the waiting room, the *tracker* script activates the *virtual layout creation* scripts, initiating the process of generating a new layout with the updated parameters. Although simple, the *Tracker* script is vital in helping other scripts adapt the virtual layout based on the user's current location.

Offset Tracker Script: This script was developed to automatically recenter the virtual layout gradually and unnoticed, preserving user immersion. The script works by detecting when the user crosses the center of the physical play area. The *detect player in area center*

script then signals the *offset tracker*, providing the user's current room and its center coordinates. Since the physical play area representation in Unity moves with the player, its center varies depending on the room. So, each room's physical center must be tracked individually for automatic recentering across all rooms.

The *offset tracker* script maintains a dictionary linking each room to its physical area center in the virtual layout. As the user crosses the center of a room, the *detect player in area center* script updates this dictionary with the current room and its center coordinates. Over time, the dictionary expands as the user explores more rooms. When the user enters a new room (signaled by the *tracker* script), the *offset tracker* recenters all rooms except the one the user is in, using the same logic described in [Subsection 4.3.3](#). Although this method is not flawless, as it depends on the user crossing each room's center, it effectively reduces layout offsets without requiring manual intervention.

4.4 Summary

This chapter detailed the implementation of the primary systems that compose the tool, building on the concepts defined in [Chapter 3](#). The chapter began with creating virtual layouts, including how they can be saved and imported for future use. The chapter then explained the implementation of each room connection technique – Portals, Corridors, and Central Hubs – and the scoring system, emphasizing using prefabs such as portals, corridor nodes, and [CH](#) rooms to streamline these processes. The application of formulas from the previous chapter in developing automated scripts for connecting rooms was also demonstrated.

The chapter further explored the *adaptive systems* that enhance the tool's flexibility and user experience. This included discussing the menu features, challenges faced during implementation, and how modifications to existing scripts enable real-time changes to the virtual layout. The extraction and use of *Guardian Boundaries* and methods for recentering the virtual layout were also covered, along with the tracker algorithm's purpose and implementation.

The next chapter introduces all the studies made on the tool to test and validate its implementation and whether it can enable walking in [VR](#) and in small physical spaces.

EVALUATION

This chapter aims to address further the research questions proposed in [Section 1.2](#) and validate the tool developed. To achieve this, three types of evaluations were conducted. The first evaluation involved interviews with HCI experts to assess the efficacy and feasibility of the tool's architecture before its development and to gather early feedback for improvement. These interviews are detailed in [Section 5.1](#).

Next, a user study was conducted to evaluate the tool's implementation and the three main room connection techniques: Portals, Corridors, and Central Hubs. These tests were crucial for obtaining feedback on the tool's functionality, validating its effectiveness in creating virtual layouts that can be navigated in small physical areas, and addressing the proposed research questions. The details of this user study is presented in [Section 5.2](#).

Finally, a load test was conducted to assess the limits of the tool's procedural generation systems. The methodology and results of this test is discussed in [Section 5.3](#).

Each section begins by explaining the testing protocol, including how each task was tested and the specific questions posed to participants. Following the protocol explanation, the analysis of the collected data is described, along with a discussion of the results for each task. Each section concludes with a broader discussion and analysis of the findings. The chapter concludes with a comprehensive discussion in [Section 5.4](#), which integrates the results of all the tests conducted.

5.1 Interviews and Feedback

After researching the state of the art relevant to this thesis, an early conceptualization of the tool and its features was developed. To validate this initial design, ten interviews were conducted with experts from various fields of HCI. The goal was to assess the efficacy and feasibility of the proposed tool architecture and gather feedback for improvements. This process enriched the early design by incorporating diverse perspectives, ensuring a more robust solution capable of accommodating a wide range of scenarios and addressing different user needs.

5.1.1 Protocol

The interviews were designed as semi-structured sessions to allow participants the freedom to direct the conversation according to their expertise and insights. However, to ensure a well-rounded discussion, specific topics were consistently addressed in each interview:

1. **Application scenarios:** Participants were encouraged to share their perspectives on potential real-world applications where our proposed solution could be effectively implemented.
2. **Usability challenges:** Discussions centered around identifying usability challenges or limitations faced by users, including exploring potential discomforts, restrictions in movement, and challenges in interaction.
3. **Enhancing user experience:** Participants were prompted to provide feedback to enhance the overall user experience within constrained VR settings. This encompassed suggestions for improving navigation, mitigating motion sickness, and optimizing user comfort.
4. **Refinement suggestions:** Participants were encouraged to offer suggestions for refining the tool and overcoming limitations identified during the discussions.

The interviews began with a short 3-minute pitch, briefing participants about the problem statement and our proposed solution. The intention was to avoid biasing the participants by delving into extensive details that might influence their initial perceptions. Following this, initial feedback was collected to gauge participants' immediate thoughts and observations. Subsequently, participants engaged in a hands-on experience with the developed system. They were presented with a virtual apartment connected with one of the three techniques – Portals, Corridors, or Central Hubs – to enable exploration in a 2.5m x 2.5m physical space. The goal of the use case was to simply explore each room. After the hands-on experience, participants were encouraged to share detailed feedback, insights, and observations based on their interaction with the system and the scenarios presented. This post-experience feedback aimed to capture their perceptions after engaging with the solution firsthand. Individual interviews, lasting approximately 30 to 45 minutes, were conducted in person to ensure that all participants had the opportunity to experience the VR demo firsthand.

5.1.2 Results

The insights gathered from the interviews are summarized in Table 5.1. Participants identified diverse application scenarios showcasing the tool's versatility, such as virtual tours, museums, galleries, and real estate visits, all accessible by simply walking through

Table 5.1: Overview of the feedback collected during the interviews with HCI researchers

Application Scenarios		
Virtual Experiences: - Virtual tours - Virtual museums and galleries - Exploration of environments - Real estate visualization	Interactive and Collaborative Environments: - E-commerce platforms - Collaborative spaces for remote teamwork - Gaming - Virtual Escape Rooms - Chat Rooms	Simulation and Training: - Construction testing (e.g. amusement parks) - Training for emergency scenarios
Usability Challenges		
Spatial Configuration and Boundaries: - Determine minimum viable physical area - Irregularly shaped physical spaces - Uneven terrain (e.g. sloping terrain) - Portal positioning - Abrupt transition between rooms - Fixed play area obstacles (e.g., a pillar) - Preserving distance perception	User Accessibility and Comfort: - Accessibility for people with limited mobility - Addressing claustrophobia in confined spaces - Addressing simulator sickness for sensitive users	Environmental and External Influences: - Play area changes during experiment - Multiple users in the same play area - Space limitations affecting activity types - Consistency in environmental sounds
Enhancing User Experience and Refinement Suggestions		
Portal Design and Functionality: - Context-orientated portal aesthetics - Rotating portals based on user orientation - Show both original doors and added portals	Navigation and Transition Enhancements: - Room transition with gradual stride adjustment - Corridors for smoother room transitions - Real-time virtual stride adjustments - Adjust space before or during the experience - Provide maps for user orientation	User Experience and Adjustments: - FOV adjustment based on walking speed - Alerts for changes in the physical playing area - Include demos for new users - Customizable soundscapes - Dynamic environmental lighting - Moving objects within VR (e.g., move a chair) - Algorithmic suggestions for input parameters - Space generation and adjustment in real time - Consideration of proxemic relationships in VR

one's living room. This extends to interactive and collaborative environments, enhancing e-commerce platforms, remote collaboration, and simulation training for emergency scenarios. For instance, schools and companies could use the tool for immersive emergency response training, while firefighters could train in randomly generated virtual environments.

However, these applications present significant usability challenges, as one participant noted: "It is necessary to establish limits on the physical size and the step adjustment. How small can the physical area be and still remain usable?". Key issues include accommodating various physical spaces, maintaining accurate distance perception in virtual environments, and strategically placing portals to avoid physical obstacles. Accessibility and comfort are also critical; the system must cater to users with limited mobility, address claustrophobia in confined spaces, and minimize simulator sickness. Additionally, factors such as changes in the play area or multiple users sharing a space could lead to unexpected objects entering the area or collisions.

Suggestions for improvement include contextually appropriate portal designs, such as traditional doors for real estate or futuristic portals for gaming. Portals that rotate and adapt to user orientation could enhance navigation in complex layouts. Gradual stride adjustments and virtual corridors could improve transitions in environments with varying room sizes, such as virtual real estate tours. One participant emphasized the need to "adjust the step size in real-time, consciously modify settings, and allow the system to provide suggestions for changing settings.". Real-time adjustments, such as door repositioning and modifying the field of view based on walking speed, would ensure the virtual environment remains tailored to user needs, enhancing both safety and immersion. Personalized soundscapes and dynamic ambient lighting could further deepen immersion, reflecting real-world conditions and smoothing transitions between virtual and physical spaces.

5.1.3 Discussion

The interviews underscore the tool's potential to significantly enhance user engagement across various fields, such as virtual tours, e-commerce, training, museums, and remote collaboration. By enabling users to walk naturally within a small 2.5m x 2.5m play area, the tool offers a more intuitive and immersive experience compared to traditional controller-based systems. The presented use case demonstrates how the tool can condense a virtual environment of approximately 100 m² into a walkable space within this limited area.

The feedback and results from early testing were instrumental in shaping the tool's development. They not only highlighted usability challenges and safety concerns — particularly in mapping complex environments and ensuring safety in shared or confined spaces — but also informed subsequent implementation decisions. As a result, the tool has been refined to better address these issues, enhancing its adaptability and user experience.

This focus on integrating user feedback aligns with the existing literature on VR's transformative impact but advances the field by emphasizing the adaptability of virtual environments and natural navigation in small spaces. This is crucial for designing home-use experiences, making VR more accessible and demonstrating its potential to provide immersive experiences in everyday settings. Practical applications include improving training simulations in emergency services and enhancing customer engagement in retail and real estate, especially for remote interactions.

The tool's implementation, guided by early feedback, positions it as a solution for creating immersive and adaptable VR experiences, with the potential to significantly enhance both daily life and professional activities.

5.2 User Study

To evaluate the tool implementation and user navigation using the three connection techniques — Portals, Corridors, and a Central Hub — user studies were conducted involving 33 participants to answer the first research questions:

- *RQ1: How can a tool seamlessly integrate diverse walking techniques in small spaces?*

To address this research question, the study assessed the sense of presence, cybersickness, spatial awareness, naturalness, and overall user experience while navigating the virtual environment generated by the tool.

5.2.1 Protocol

The participants in this user study were volunteers recruited from diverse backgrounds with varying levels of expertise in VR. To ensure a fair evaluation of the three connection techniques, the participants were divided into three groups, each testing one technique. This division was opted because testing all three techniques with each participant would have made the experience long and tiring. The participants were grouped to ensure homogeneity in terms of their experience with VR.

Experimental Setup The experiment occurred in a standard laboratory environment with an available tracking area of approximately 2.5 m x 2.5 m. The computer used was equipped with an NVIDIA GeForce RTX 4080 with 16 GB of dedicated memory. Participants used a VR headset (HTC VIVE Cosmos) and one controller to perform the task. This setup provided all participants with a controlled and consistent environment, ensuring standardized conditions for comparing the portal, corridor, and central hub techniques.

Task Participants searched for sixteen music vinyl LP record discs distributed across five rooms within the virtual apartment shown in Figure 4.9. Each room, except the first (living

room), contained three discs; the first had four. Object placements and task difficulty were consistent across all scenarios, ensuring uniformity. The task aimed for simplicity and intuitiveness, allowing participants to focus on navigating the virtual environment rather than managing additional cognitive challenges. Participants navigated physically through the apartment, guided by a visual cue on their virtual hand indicating the number of discs found (Figure 5.1). Upon locating a disc, participants interacted by touching the disc with their virtual hand to collect it. They then returned to the first room to place the disc into a music player, initiating playback throughout the virtual apartment. Participants continued until all sixteen discs were found, concluding the task.

5.2.2 Data Collection

Data collection for the study involved a combination of quantitative and qualitative measures. Completion time was automatically recorded by the application and extracted for analysis. The participant's subjective feedback were collected through post-task questionnaires (see Appendix A) gathering the following insights:

Presence: Participants answered the Igroup Presence Questionnaire (IPQ) [69], which evaluated their subjective feelings about being present in the virtual apartment.

Cybersickness: Participants answered the Simulator Sickness Questionnaire (SSQ) [28] before and after the experiment to address any discomfort symptoms caused by the navigation.

Spatial Awareness and Orientation: Participants were asked about their spatial awareness, including their ability to navigate easily through the virtual apartment, their understanding of room connections and layout, and their memory of specific objects or landmarks within the virtual apartment. In particular, after the experience, the participants were asked to draw the apartment layout they had visited.

User Experience and Naturalness: Participants were asked to rate their immersion, comfort level, and perception of naturalness while navigating through the virtual apartment.

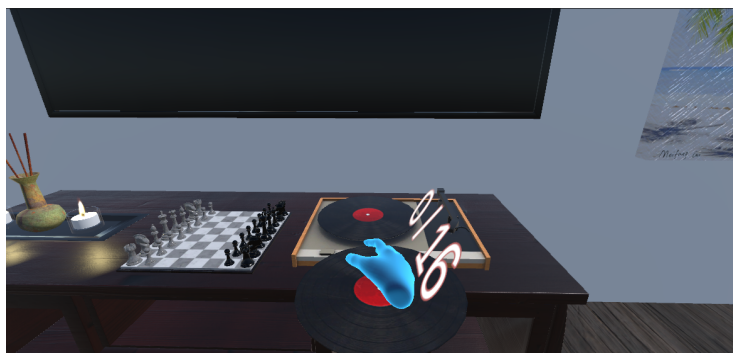


Figure 5.1: The participant's task consisted of finding and placing sixteen music discs in the music player.

Experiment Procedure All participants performed the experiment under the same external conditions to ensure consistency and maintain uniformity across participants. The session began with the participants reading and signing the informed consent form, indicating voluntary participation, anonymization of data, and the possibility of giving up at any time. Then, participants filled out the SSQ questionnaire to assess symptoms of motion sickness before starting the experiment. A brief introductory was provided, explaining that participants would navigate through a virtual apartment to locate a set of music discs. The participants were also asked to pay close attention to the virtual apartment layout. No further information was given so as not to bias users. The participants were randomly divided into three groups, each group had 11 participants and performed one of three scenarios: Portals, Corridors, or Central Hub. Each participant began by executing their assigned scenario and accomplishing the given task. After completing the assigned scenario, they completed post-session questionnaires, offering subjective feedback about their experiences. At the end, participants were given time to express any additional comments they considered relevant. All participants were rewarded with a chocolate to thank them for taking part in the user study.

5.2.3 Results

The results of the user study to assess user navigation into the three connecting room techniques — Portals, Corridors, and Central Hubs — will now be presented.

To select the most appropriate statistical test to analyze the results, the distribution of the data was first assessed using the Shapiro-Wilk test for each group of questionnaires. The One-Way ANOVA test would be used if the data met the normality assumption. Otherwise, the Kruskal-Wallis H test was chosen to analyze the results, given its appropriateness for comparing three independent groups on the same dependent variable.

Demographics A total of 33 participants took part in the study (9 females, 24 males). The participant's ages ranged from 19 to 28 ($M = 23.30$, $SD = 2.59$). The majority of participants were students from the computer science department of NOVA FCT. In terms of education, 12 participants held a Master's degree, 13 participants held a Bachelor's degree, and 8 participants had a high school diploma or equivalent. Regarding their experience with VR applications, 7 participants reported having never experienced VR before, 18 participants had only experienced VR a few times (at least once), 6 participants reported using VR applications from time to time, and 2 participants reported using VR regularly on a monthly, weekly, or daily basis. The participants distributions is displayed in [Table 5.2](#)

5.2.4 Presence

[Figure 5.2](#) visually compares the scores obtained from the IPQ questionnaire, which assesses participant's sense of presence in the [VE](#). The figure allows for a comparison

Table 5.2: User Distribution per technique

	Female	Male	Experienced User	Some Experience	No Experience
Portals	3	8	54%	28%	18%
Corridors	3	8	37%	27%	36%
Central Hub	3	8	36%	45%	19%

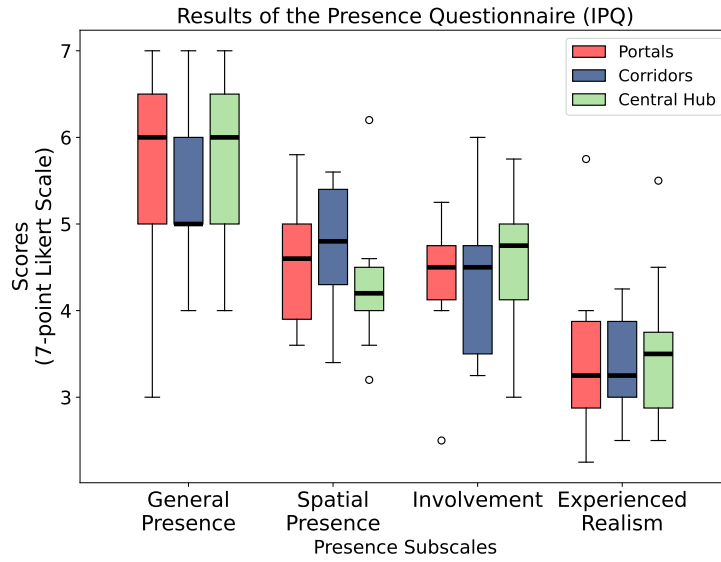


Figure 5.2: IPQ sub-scales results for each technique.

of the overall presence reported by participants across the three techniques, providing insights into different dimensions of presence: general presence (GP), spatial presence (SP), involvement (I), and experienced realism (ER). Table 5.3 details the mean and standard deviation results. Notably, scores obtained in these subscales were consistently close across the three techniques. The Kruskal-Wallis H test analysis revealed no significant difference between them in terms of general presence ($\chi^2(2) = 0.431, p = 0.806$), spatial presence ($\chi^2(2) = 2.743, p = 0.254$), involvement ($\chi^2(2) = 0.855, p = 0.652$) and experienced realism ($\chi^2(2) = 0.163, p = 0.922$).

Table 5.3: Descriptive statistics for the IPQ subscales across the three connection techniques.

Subscale	Portals (M $\pm$ SD)	Corridors (M $\pm$ SD)	Central Hub (M $\pm$ SD)
General Presence	5.55 $\pm$ 1.29	5.45 $\pm$ 0.82	5.73 $\pm$ 1.01
Spatial Presence	4.55 $\pm$ 1.84	4.78 $\pm$ 0.96	4.31 $\pm$ 2.00
Involvement	4.41 $\pm$ 1.67	4.36 $\pm$ 1.67	4.59 $\pm$ 1.69
Experienced Realism	3.43 $\pm$ 1.68	3.41 $\pm$ 1.59	3.55 $\pm$ 1.69

5.2.5 Cybersickness

The three connection techniques resulted in low levels of simulator sickness symptoms before and after the navigation tasks. The subscores acquired from the questionnaire, including nausea, oculomotor, and disorientation, were further analyzed. A Wilcoxon signed-rank test was conducted to compare each technique's pre-session and post-session simulator sickness questionnaire scores. Figure 5.3 compare the results of each symptom assessed by the SSQ questionnaire before and after the experiment. For the group that tested the corridors technique, the analysis revealed a significant difference between the pre-session and post-session disorientation scores ($Z = -2.226, p = 0.026$). Similarly, for the portals group, there was a significant difference between the pre-session and post-session nausea scores ($Z = -2.157, p = 0.031$). Apart from these two cases, no significant differences were found in any other subscore. Post-analysis found that these differences were attributable to two participants who were outliers in the sample.

5.2.6 Spatial Awareness and Orientation

Participant's spatial awareness and orientation were assessed using three questions: Q1) "I was able to easily navigate through the virtual apartment", Q2) "I had a clear sense of the room connections and layout", and Q3) "I remember the locations of specific objects or landmarks within the virtual apartment". Each question was rated on a 7-point Likert scale (1 = Strongly Disagree, 7 = Strongly Agree). The Kruskal-Wallis H test showed no significant difference between the three techniques in Q1 ($\chi^2(2) = 1.133, p = 0.568$), Q2 ($\chi^2(2) = 2.930, p = 0.231$), and Q3 ($\chi^2(2) = 1.859, p = 0.395$).

After the virtual experience, each participant drew what they perceived to be the layout of the VE. Each drawing was manually analyzed to evaluate the accuracy of the

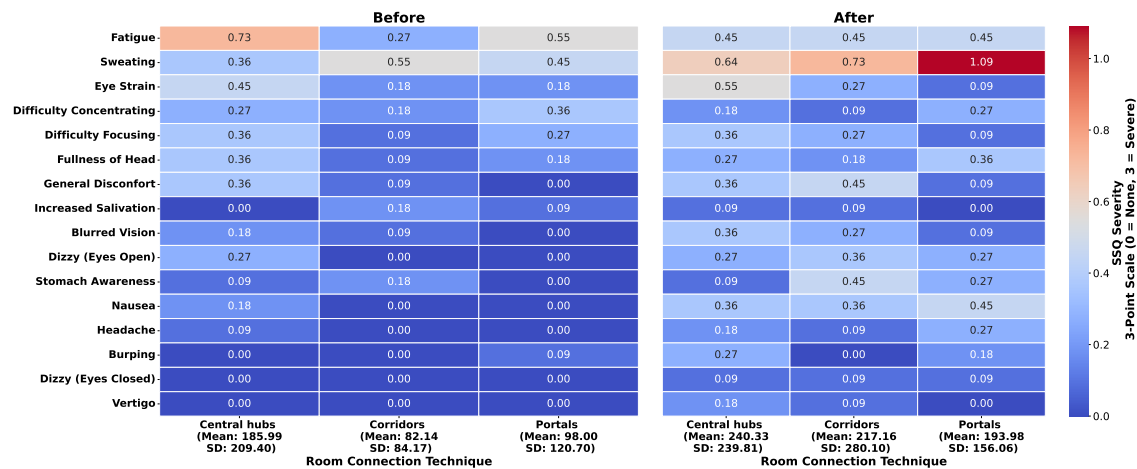


Figure 5.3: Heat map showing the SSQ results for each technique before and after the experiment.

depicted room connections. The VE consisted of five connections. Each correct connection scored 1 point, while each incorrect or missing connection incurred a deduction of 1 point. Figure 5.4 illustrates two drawings representing participant's perceptions of the virtual apartment layout. The right drawing is from a central hub scenario, and the left one is from a corridor scenario. Statistical analysis using the Kruskal-Wallis H test indicated no significant difference in layout scores between the portals group ($M = 4.182$, $SD = 1.079$), corridors group ($M = 3.818$, $SD = 1.779$), and central hub group ($M = 4.455$, $SD = 1.293$), $\chi^2(2) = 3.383$, $p = 0.429$.

5.2.7 User Experience and Naturalness

Participants rated four statements on a 7-point Likert scale (1 = Strongly Disagree, 7 = Strongly Agree) to assess user experience and naturalness: Q4) "I felt immersed in the virtual apartment", Q5) "I felt comfortable while navigating through the virtual apartment", Q6) "I found navigating through the apartment natural", and Q7) "I found the transition between rooms natural". The scores for questions Q4, Q5, Q6, and Q7 were similar across the three techniques, as shown in Table 5.4 and Figure 5.5. No significant difference was detected for Q4 ($\chi^2(2) = 1.158$, $p = 0.560$), Q5 ($\chi^2(2) = 1.814$, $p = 0.404$), Q6 ($\chi^2(2) = 2.053$, $p = 0.358$), and Q7 ($\chi^2(2) = 2.286$, $p = 0.319$).

5.2.8 Completion Time

Completion time refers to the duration participants took to perform the task. Figure 5.6 displays a violin plot illustrating the time results. There was a statistically significant difference in completion time between groups as determined by one-way ANOVA ($F(2, 30) = 18.547$, $p < .001$). A Tukey post hoc test revealed that completion time was significantly lower in the portals group ($09:12 \pm 01:38$ min) compared to corridors ($13:21 \pm 02:16$ min), $p < .001$. Similarly, completion time was significantly lower in the central hub group ($09:16 \pm 01:28$ min) compared to corridors ($13:21 \pm 02:16$ min), $p < .001$.

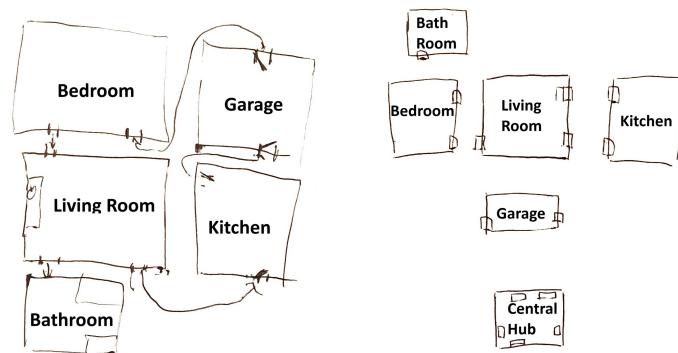


Figure 5.4: Examples of drawings made by participants depicting the layout of the VE. The labels were written in a computer font afterwards for better readability. On the left is a corridor example, and on the right is a central hub example.

Table 5.4: Descriptive statistics for the user experience subscales across the three connection techniques.

Questions		Portals	Corridors	Central Hub
Q4 - Immersion	Med.	6.00	6.00	6.00
	Q1	6.00	5.00	5.00
	Q3	7.00	6.00	6.00
Q5 - Comfort	Med.	6.00	5.00	5.00
	Q1	5.50	4.00	5.00
	Q3	6.00	6.00	6.00
Q6 - Natural Navigation	Med.	6.00	4.00	4.00
	Q1	4.00	3.50	3.50
	Q3	6.50	5.50	5.50
Q7 - Natural Transitions	Med.	5.00	4.00	5.00
	Q1	4.00	3.00	3.00
	Q3	6.50	4.00	5.00

5.2.9 Discussion

The primary objective of this study was to evaluate the effectiveness of three different room connection techniques — Portals, Corridors, and a Central Hub — in enhancing the user experience within a constrained physical play area for VR. The user study allowed the gathering of insights into the usability and comfort of these techniques, particularly in terms of presence, cybersickness, spatial awareness, orientation, and overall user experience.

Presence: The IPQ questionnaire indicated that all three techniques performed similarly across the dimensions of spatial presence, involvement, and experienced realism. This similarity suggests that each technique is equally capable of maintaining a strong sense of presence within the virtual environment. The lack of significant differences in presence scores implies that the choice of connection technique may not critically impact the user’s feeling of being “there” in the VR space. This is a positive outcome, as it provides

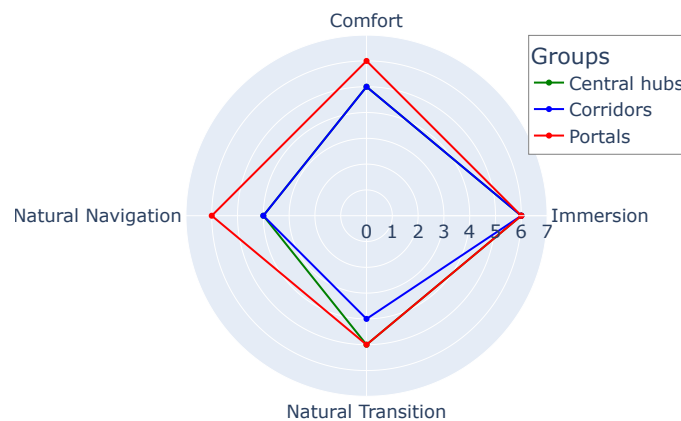


Figure 5.5: Radial graph comparing the results of the user experience questions for the three techniques.

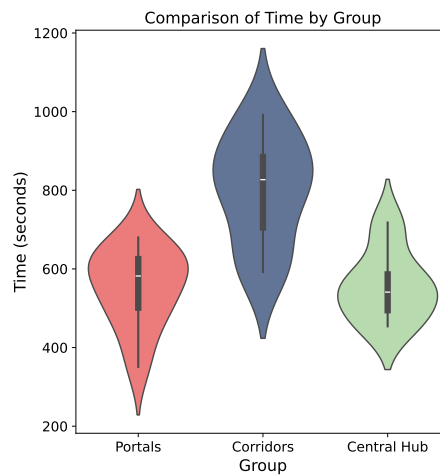


Figure 5.6: Violin plot comparing the completion times of the task for the three techniques.

flexibility in choosing a connection method based on other factors, such as user preference or specific application requirements, without compromising the immersive experience.

Cybersickness: The study revealed low levels of simulator sickness symptoms across all techniques, which is encouraging. However, specific subscores for disorientation and nausea indicated significant differences pre and post-session for the corridors and portals groups, respectively. This suggests that while overall cybersickness was minimal, particular aspects of these techniques might still induce discomfort in some users. Notably, these differences were traced back to outliers, indicating that individual susceptibility plays a role. Future iterations of the tool should consider customizable settings or gradual exposure to mitigate such effects.

Spatial Awareness and Orientation: User feedback and layout drawing tasks revealed no significant differences in spatial awareness and orientation between the techniques. Participants reported similar ease in navigation, understanding room connections, and remembering object locations. This suggests that portals, corridors, and central hubs all provide a coherent spatial experience and support clear mental maps of the virtual environment. The accuracy of layout drawings further supports this, showing similar correctness regardless of the technique used. However, visual observations during the experiment noted that participants in the central hub scenario drew the [VE](#) layout quicker than those in the other two scenarios. In contrast, participants in the corridors scenario took more time and appeared to exert additional effort when drawing the layout.

User Experience and Naturalness: User experience ratings showed no significant differences across the techniques, indicating that all three methods provided a comparable level of immersion, comfort, and naturalness in navigation and room transitions. This uniformity in user experience is particularly noteworthy as it suggests that users can seamlessly adapt to any of the three connection strategies without perceiving a drop in the quality of their interaction with the virtual environment. This adaptability is advantageous for developers and users alike, allowing for the choice of technique based on practical

considerations and preferences without sacrificing user satisfaction.

Completion Time: There was a significant difference in task completion times, with the corridor group performing slower than the portal and central hub groups. Based on the participant's feedback and observations, two main factors were found for the increased time required for navigating corridors. First, corridors add additional paths to the original layout, requiring users to walk more. Second, since participants were exploring the virtual apartment for the first time, they were familiarizing themselves with the space. With the portals and central hub techniques, participants had a preview of the rooms they could enter, as they could see them through the door. However, with the corridors technique, participants only saw the destination room as they reached the end of the corridor. This lack of preview often led participants to walk in and out of the corridor to check which room was on the other side, contributing to longer navigation times. For future improvements, users suggested including hints about the room the corridor leads to, such as labels with the room name or symbols.

5.3 Load Test

A load test was conducted to evaluate the tool's performance in generating virtual layouts of varying sizes and using different room connection techniques on two computers with distinct hardware configurations, with the aim of providing usage guidelines based on the results.

5.3.1 Protocol

This study evaluated the tool's performance behaviour in generating virtual layouts with varying numbers of rooms and measured the time required to complete each layout using one of three room connection techniques — Portals, Corridors, or Central Hubs. To understand the tool's performance comprehensively across different environments, the tests were conducted on two computers with different hardware configurations: one high-end and one low-end (see [Table 5.5](#)). Each virtual layout was generated using only one technique at a time, ensuring a fair assessment of the performance cost associated with each room connection method.

Experimental Setup The experiment occurred in a standard laboratory environment with the two available computers. These two computers – one of them is a high-end computer while the other is a low-end computer – were used to conduct the tests. The computers have the following components:

This setup provided a controlled and consistent environment for the load tests, ensuring standardized conditions for comparing the tool's room generation using only portals, corridors, or central hubs.

Table 5.5: Computers Components

	High-end PC	Low-end PC
Operative System	Windows 11 Pro	Windows 10 Pro
Processor	13th Gen Intel(R) Core(TM) i7-13700KF 3.40 GHz (16 Total Cores)	Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz 4.20 GHz (4 Total Cores)
RAM	32.0 GB - DIMM 3200 MT/s	16.0 GB - DIMM 3200 MT/s
Storage	SSD 1TB - Sequential Read: 4500 (MB/s) - SequentialWrite: 3600 (MB/s)	SSD 500MB - Sequential Read: 3500(MB/s) - Sequential Write: 3000(MB/s)
Graphics Card	NVIDIA GeForce RTX 4080 - 16 GB Dedicated Memory / 16 GB Shared Memory	NVIDIA GeForce GTX 1070 - 8 GB Dedicated Memory / 8 GB Shared Memory

Task The tool was tested on each computer to create virtual layouts with room counts of **10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, and 200** rooms. Each room connection technique — Portals, Corridors, and Central Hubs — was tested in isolation for each set of rooms. For example, the tool was tested to generate ten rooms using only portals, corridors, and central hubs. Room increments of 10 were tested up to 100 rooms to closely observe the impact of adding 10 rooms. Beyond 100 rooms, increments of 50 were used to place a greater load on the tool and test its limits.

Data Collection Data collection for the test involved quantitative measures, including the type of computer used (high-end or low-end), the number of rooms in the virtual layout, the room connection technique employed, and the time required to generate the virtual layout. Each configuration was tested five times, and the mean value was calculated. If a technique could run the configurations at least once, the rest were recorded as the maximum allowable time, assumed to be 10 minutes or more. If the tool could not complete the generation of the virtual layout for any technique, the data was recorded as missing values.

Experiment Procedure All tests on each computer were conducted under consistent external conditions to ensure uniformity across both machines. The testing procedure began by launching the tool in Unity, entering the desired number of rooms, and setting custom parameters to enforce the use of a single-room connection technique. Once the tool was initiated with the specified configuration, a timer tracked the time required for the tool to generate the virtual layout, complete it, and make it ready for exploration. This process was repeated five times for each room configuration and each of the three connection techniques.

5.3.2 Results

This section presents the results of the load tests, which assessed the tool's performance behavior in generating virtual layouts of varying sizes using different room connection techniques on two computers: a high-end and a low-end setup.

The data distribution was first assessed using the Shapiro-Wilk test to determine the appropriate statistical analysis. As the data did not meet the normality assumption, the non-parametric Kruskal-Wallis H test was applied, which is suitable for comparing three independent groups on the same dependent variable.

Computers: Figure 5.7 visually represents the time taken to generate virtual layouts with varying room counts and different connection techniques across both computers. The comparison illustrates the overall task completion time for the high-end versus low-end computer and the performance cost associated with each room connection technique. The Kruskal-Wallis H test revealed a statistically significant difference in completion times between the two computers, $\chi^2(2) = 13.288$, $p < 0.001$, with a mean rank time completion score of 25.64 seconds for the high-end computer and 42.93 seconds for the low-end computer.

Number of Rooms: As shown in Figure 5.7, increasing the number of rooms significantly impacted layout generation times on both computers. Larger room sets consistently led to longer generation times, indicating a direct correlation between room count and performance. For room sets larger than 100, the low-end computer exceeded the 10-minute generation limit, resulting in incomplete data for these cases. A Kruskal-Wallis H test was performed to compare the time completion across different room set sizes on both computers. On the high-end computer, the test revealed no statistically significant difference in time completion between the different room set sizes, $\chi^2(2) = 18.077$, $p = 0.080$. Similarly, on the low-end computer, the Kruskal-Wallis H test also indicated no statistically significant difference in time completion between the different room set sizes, $\chi^2(2) = 12.495$, $p = 0.187$. The means rank of time completion for each set of number of rooms in each computer is shown in Table 5.6.

Techniques: The completion times for each room connection technique were measured across all room sets on both computers to assess whether there were notable differences in virtual layout generation times. As illustrated in Figure 5.7, the corridor technique

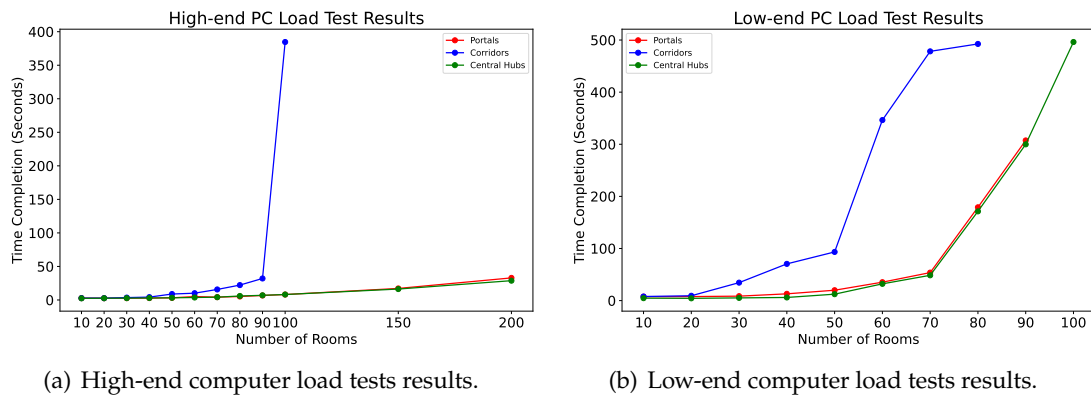


Figure 5.7: Load Tests Results. For each configuration, the tool was run five times, and the mean time to generate the virtual layout was calculated.

Table 5.6: Mean scores of time completion in each set of number of rooms in the high-end computer and low-end computer.

NoR	High-end PC (seconds)	Low-end PC (seconds)
10	7.17	7.67
20	5.00	8.33
30	10.17	11.33
40	14.00	14.00
50	18.00	16.67
60	21.33	20.33
70	21.33	22.33
80	25.00	25.33
90	27.00	17.67
100	29.00	11.33
150	20.83	–
200	23.17	–

consistently required more time to complete than the other two techniques on both systems. However, the Kruskal-Wallis H test indicated no statistically significant difference in completion times between the techniques. For the high-end computer, the test yielded $\chi^2(2) = 0.350$, $p = 0.840$, with mean time completion scores of 18.13 seconds for portals, 19.92 seconds for corridors, and 17.46 seconds for central hubs. On the low-end computer, the test results were $\chi^2(2) = 0.437$, $p = 0.804$, with mean time completion scores of 14.70 seconds for portals, 17.00 seconds for corridors, and 14.80 seconds for central hubs.

5.3.3 Discussion

The load tests were conducted to analyze the tool's procedural generation performance under different conditions, including variations in hardware, the number of rooms generated, and the room connection techniques used. The primary goal was to observe the tool's behavior and provide usage guidelines.

Computers: Performance varied significantly between the two computers. As expected, the high-end system handled the generation of larger virtual layouts much more efficiently than the low-end system. For instance, the high-end computer successfully generated up to 100 rooms across all techniques, while the low-end system struggled with larger configurations. This suggests that users with less powerful systems may face limitations when generating complex layouts, and using simple room sets is recommended.

Number of Rooms: The time required to generate layouts increased as the number of rooms grew, which is consistent across both systems. This is a predictable outcome, as larger room sets naturally involve more processing. While the tool is capable of generating substantial layouts, users aiming for faster generation times may opt for smaller room sets, especially when using lower-end hardware. Based on the load test results, it is recommended to generate a maximum of **90** rooms on high-end computers and **50** rooms on low-end computers.

Techniques: Among the different room connection techniques, the corridor method consistently took slightly longer to generate layouts compared to the portal and central hub techniques, likely due to the corridor technique requiring the instantiation of more objects. However, the difference in completion time was not statistically significant, and the tool performed reliably across all techniques. Notably, the central hub and portals techniques stood out as the only techniques capable of generating **200** rooms on the high-end computer within a reasonable time and up to **90** rooms on the low-end computer. In contrast, the corridor technique struggled with higher room counts and failed more tests. This suggests that the central hub and portal technique is a more suitable option for adjudicating or generating more complex layouts (e.g., museums, virtual apartments) when a larger number of rooms is required.

These tests provide insight into the tool’s performance across different setups and help guide users in optimizing the tool based on their system capabilities and layout requirements.

5.4 Final Discussion

The tests conducted on the tool offered valuable insights into its functionality, potential applications, and the validation of its implementation. They also assessed the tool performance behavior when generating large virtual layouts with diverse configurations.

The interviews detailed in [Section 5.1](#) were crucial in gathering early feedback from HCI experts. This feedback confirmed that the tool was progressing in the right direction and enabled adjustments to the work plan before implementation. It also revealed several potential use cases, including virtual apartment tours, virtual museums, e-commerce platforms, and simulation scenarios. This input helped shape a relevant use case for the user study, reinforcing the tool’s purpose and versatility. The feedback from these interviews also influenced key features, such as the gradual stride adjustments during room transitions and the use of the corridor technique for smooth navigation between rooms.

User studies were designed to evaluate the tool’s room connection techniques – Portals, Corridors, and Central Hubs –, focusing on their feasibility and implementation. The primary objective was to determine whether a virtual layout — specifically, a virtual apartment — could be navigated using walking within a small 2.5m x 2.5m physical space, addressing **RQ1**. Results showed that all implemented techniques effectively connected the apartment’s rooms, allowing users to explore without exceeding physical boundaries.

Portals offer flexibility and efficiency in connecting rooms without the need for additional virtual elements, as they can be placed anywhere in the virtual space, not just on walls. Portals provide users with a clear preview of the connected rooms, enhancing spatial awareness. They also received the highest ratings in terms of user experience and

naturalness. This versatility makes them suitable for various scenarios, though they are particularly well-suited for sci-fi settings, where portals are more naturally integrated.

Corridors maintain the natural flow of room connections, serving as familiar transition zones between spaces. Appearing as doorways within walls, they offer smoother navigation and are ideal for scenarios requiring realism and immersion, such as virtual apartment tours or emergency training simulations.

Central hubs simplify navigation by linking multiple rooms or other hubs, making them effective for large, interconnected environments. Acting as a central space, they reduce users' mental load, allowing easy access to other rooms. Virtual museums are a good use case for central hubs, as they enable users to explore without worrying about navigation, enhancing the overall experience.

Through the study, **RQ1** was fully addressed, confirming that the tool could integrate multiple walking techniques — Portals, Corridors, and Central Hubs — enabling navigation in limited physical spaces and in [VR](#). The study also demonstrated that the tool successfully generated virtual layouts that could be comfortably explored within at least a 2.5m x 2.5m space.

For **RQ2**, preliminary evaluations during implementation indicated that virtual layouts could be generated in even smaller areas, such as 1.5m x 1.5m. However, while techniques such as Portals and Central Hubs allowed for feasible navigation in smaller spaces, using Corridors proved more challenging. This suggests a comfortable minimum physical space is currently around 2.5m x 2.5m since a simple corridor (at least 2 nodes) would take around 1 m and require at least 50 cm for the user to walk and do the turns comfortably. With regard to computational limits, load tests suggest that the tool can accommodate **90** rooms in working time on a high-performance computer and **50** rooms on a medium-performance computer.

Regarding **RQ3**, the *adaptive systems* and menu provide users with options to personalize their experience. By allowing users to choose and mix room connection techniques or even modify the entire virtual layout, these systems offer a high level of customization within the [VE](#).

Overall, these tests validated the tool's current implementation and provided directions for future development.

5.5 Summary

This chapter presented the comprehensive testing conducted throughout this dissertation and the analysis and insights derived from these evaluations. The tests validated various aspects of the tool's implementation, except for the adaptive systems, which were developed last and, therefore, not included in the user study.

Interviews were conducted early in the tool's development to better assess, guide the implementation, and validate its solution. These interviews were instrumental in

gathering early feedback and focusing on the key features that the tool should have or could be enhanced.

A user study was then performed, primarily addressing **RQ1**, to test the tool's capability of connecting a **VE** using three room connection techniques to enable walking in small physical spaces and in **VR**.

A load test was conducted on the tool's procedural generation system to evaluate its ability to generate multiple virtual layouts of varying sizes and with different room connection techniques and gather insights to answer **RQ2**.

Lastly, tests on the *adaptive systems*, mentioned in the final discussion, provided valuable insights to address **RQ3**.

The final chapter will conclude by summarizing the tool's development and implementation, offering final insights into its achievements, reflecting on the research questions, and suggesting potential directions for future work.

CONCLUSION

Virtual Reality (VR) immerses users in virtual worlds, enabling them to experience environments beyond reality from home. Navigation methods vary, with teleportation being the most common due to its disregard for physical space, though it lacks immersion. Walking offers a more immersive experience but is limited by the need for large physical spaces, often beyond user's reach.

This thesis introduces a tool that generates virtual layouts adapted to the user's physical environment, enabling walking in **VR** within limited spaces. The tool utilizes **Procedural Content Generation (PCG)** algorithms to tailor layouts to specific configurations. Guided by the research questions in **Section 1.2**, it addresses the challenges of facilitating walking in confined physical spaces. Detailed in **Chapter 4**, the tool connects multiple virtual layouts through three room connection techniques – Portals, Corridors and Central Hubs – that facilitate walking and adapt to user needs in real-time. The primary challenge was implementing these room connection techniques to ensure navigation without breaching physical boundaries. The tests on the adaptive system, detailed in **section 4.3**, revealed a positive user experience by turning static layouts into dynamic environments, accommodating changes during exploration. The adaptive systems also make the tool be able to adapt to physical spaces as small as 1.5m x 1.5m, albeit with limitations on technique use. Tests validated the tool's effectiveness and addressed the research questions. These included expert interviews, a user study in a 2.5m x 2.5m space, and a load test to evaluate the tool's limits. Feedback revealed positive results on the tool's versatility, usability, performance, comfort, navigation and areas for improvement.

Regarding the proposed research questions, these were:

- **Research Question 1** - How can a tool seamlessly integrate diverse walking techniques in small spaces?
- **Research Question 2** - What are the physical and virtual boundaries beyond which the user experience may be compromised?
- **Research Question 3** - How can **PCG** enable users to customize a large adaptive virtual world according to their preferences and explore it in **VR**?

RQ1 explores whether a tool could integrate multiple techniques for seamless navigation in small spaces. The tool achieves this with Portals, Corridors, and Central Hubs, enabling navigation in virtual layouts within small physical areas. Portals offer a flexible and straightforward solution, corridors provide a seamless and natural transition between spaces, and central hubs simplify navigation. Each technique offers a different way to experience navigating the virtual layouts, with unique benefits and limitations that should be considered when applying to a specific application scenario.

RQ2 addressed the limits of optimal physical and virtual boundaries, with the user study confirming that a 2.5m x 2.5m space is comfortable for natural navigation. The tool can also adapt virtual layouts to spaces as small as 1.5m x 1.5m but with certain limitations. The most significant challenge is that long corridors have a tendency to accumulate small deviations in the virtual layout. Nevertheless, each corridor requires at least 50 cm for comfortable navigation and must also allow for a seamless transition for the user between rooms. In very small physical spaces, there is not enough room to create a corridor that both ensures a smooth transition and fits within the physical constraints. As a result, the implementation becomes compromised, leading to layout offsets and disruptions in user navigation. The load test also indicates that the tool can efficiently generate up to 90 rooms on a high-performance computer and 50 rooms on a medium-performance computer within an acceptable working time. Beyond those values, the generation is more performance-intensive and depends on the computer used.

RQ3 explored how **PCG** allows for customization of virtual layouts. The tool parameters and adaptive systems enable significant customization and on-the-fly modification. The user inside the tool's virtual layout can change what room connection techniques he wants present in the virtual layout, mix them, or even change the whole virtual layout with new parameters. This gives the user greater control over what the tool generates and makes it more customizable to their preferences. Moreover, the tool employs systems that automatically adapt the virtual layout to prevent navigation disruptions and to adapt the virtual layout to match the user's desired customizations.

In conclusion, the tool effectively generates and adapts virtual layouts, incorporating room connection techniques that allow users to explore environments by walking in small physical spaces.

6.1 Future Work

Further work on this topic could benefit areas such as virtual museums or apartment tours, enabling users to explore these environments from home. Collaborative platforms or gaming could also benefit by allowing users to interact within shared virtual spaces. Additionally, the tool could be applied to training for emergency scenarios, such as firefighter drills within tool-generated layouts. Possible directions include more navigation techniques, adaptive navigation systems, interior generation, collaborative environments, and implementation enhancements.

New Walking and Navigation Techniques This could involve creating new room connection techniques or introducing techniques that improve navigation within individual rooms, not just across the entire virtual layout. For example, the tool could include the ability to subdivide larger rooms into smaller sections, allowing users to explore these areas without needing to extend their stride artificially.

Adaptive Navigation Systems The adaptive systems can be further developed to enable more dynamic, real-time adjustments that enhance customization based on individual user preferences. Future work could focus on minimizing the need for user intervention, allowing the system to autonomously adapt the environment based on inferred user behavior and preferences. This might include dynamically altering room layouts or adjusting navigation techniques in response to user movement patterns, leading to a more fluid and tailored virtual experience. Additionally, improving the system's ability to efficiently manage limited physical space could help optimize room configurations for better navigation.

Interior Generation A critical area for future development is the customization and generation of furniture within rooms. Currently, the tool generates empty rooms, which limits its applicability and user experience. Expanding the tool [PCG](#) features to include furniture generation would broaden its use cases and create more immersive and functional [VE](#).

Collaborative Environments Additionally, the tool could be expanded to support multi-user environments, where two or more people can explore the same virtual layout simultaneously and interact with each other. This feature could be particularly compelling if users can do this from either the same physical space or separate physical locations, adding a social dimension to the virtual experience.

Tool Enhancements Lastly is enhancing the tool's performance and stability to ensure smoother operation, especially when generating complex virtual layouts. Additionally, several smaller yet impactful features could make the tool more accessible and robust. For instance, implementing indicators on portals to show which doors they correspond to, providing access to a minimap for easier navigation, and offering more editing tools within the virtual layouts could significantly enhance user experience.

BIBLIOGRAPHY

- [1] C. Adams and S. Louis. “Procedural maze level generation with evolutionary cellular automata”. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017, pp. 1–8. DOI: [10.1109/SSCI.2017.8285213](https://doi.org/10.1109/SSCI.2017.8285213) (cit. on p. 23).
- [2] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. “Physical Space Requirements for Redirected Walking: How Size and Shape Affect Performance”. In: *ICAT-EGVE 2015 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*. Ed. by M. Imura, P. Figueroa, and B. Mohler. The Eurographics Association, 2015. ISBN: 978-3-905674-84-2. DOI: [10.2312/egve.20151315](https://doi.org/10.2312/egve.20151315) (cit. on pp. 1, 32).
- [3] S. Bakkes, P. Spronck, and J. van den Herik. “Rapid and Reliable Adaptation of Video Game AI”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 1.2 (2009), pp. 93–104. DOI: [10.1109/TCIAIG.2009.2029084](https://doi.org/10.1109/TCIAIG.2009.2029084) (cit. on p. 25).
- [4] E. Bastug, M. Bennis, M. Medard, and M. Debbah. “Toward Interconnected Virtual Reality: Opportunities, Challenges, and Enablers”. In: *IEEE Communications Magazine* 55.6 (2017), pp. 110–117. DOI: [10.1109/MCOM.2017.1601089](https://doi.org/10.1109/MCOM.2017.1601089) (cit. on pp. 7, 8).
- [5] P. Biswas, M. Goel, H. Negi, and M. Datta. “An Efficient Greedy Minimum Spanning Tree Algorithm Based on Vertex Associative Cycle Detection Method”. In: *Procedia Computer Science* 92 (2016). 2nd International Conference on Intelligent Computing, Communication Convergence, ICC3 2016, 24-25 January 2016, Bhubaneswar, Odisha, India, pp. 513–519. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2016.07.376> (cit. on pp. 34, 36).
- [6] M. Booth. “The AI Systems of Left 4 Dead”. In: 2014 (cit. on p. 26).
- [7] P. Bourhis, J. L. Reutter, and D. Vrgoč. “JSON: Data model and query languages”. In: *Information Systems* 89 (2020), p. 101478. ISSN: 0306-4379. DOI: <https://doi.org/10.1016/j.is.2019.101478>. URL: <https://www.sciencedirect.com/science/article/pii/S0306437919305307> (cit. on p. 52).

-
- [8] G. Bruder, F. Steinicke, and K. Hinrichs. "Arch-Explore: A natural user interface for immersive architectural walkthroughs". In: *2009 IEEE Symposium on 3D User Interfaces* (2009-03), pp. 75–82. DOI: [10.1109/3DUI.2009.4811208](https://doi.org/10.1109/3DUI.2009.4811208) (cit. on p. 12).
 - [9] X. Cai, J. Liu, L. Qi, J. Dong, Y. Gao, and H. Yu. "Toward a psychophysical-based procedural texture generation system for interactive design". In: *2015 8th International Conference on Human System Interaction (HSI)*. 2015, pp. 373–378. DOI: [10.1109/HSI.2015.7170696](https://doi.org/10.1109/HSI.2015.7170696) (cit. on p. 20).
 - [10] J. C. Cardoso and A. Perrotta. "A survey of real locomotion techniques for immersive virtual reality applications on head-mounted displays". In: *Computers Graphics* 85 (2019), pp. 55–73. ISSN: 0097-8493. DOI: <https://doi.org/10.1016/j.cag.2019.09.005> (cit. on pp. 11, 12).
 - [11] L.-P. Cheng, E. Ofek, C. Holz, and A. D. Wilson. "VRoamer: Generating On-The-Fly VR Experiences While Walking inside Large, Unknown Real-World Building Environments". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 359–366. DOI: [10.1109/VR.2019.8798074](https://doi.org/10.1109/VR.2019.8798074) (cit. on pp. 28–30).
 - [12] H. Cherni, N. Métayer, and N. Souliman. "Literature review of locomotion techniques in virtual reality". In: *International Journal of Virtual Reality* 20 (2020-03), pp. 1–20. DOI: [10.20870/IJVR.2020.20.1.3183](https://doi.org/10.20870/IJVR.2020.20.1.3183) (cit. on p. 8).
 - [13] P. Cipresso, I. A. C. Giglioli, M. A. Raya, and G. Riva. "The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature". In: *Frontiers in Psychology* 9 (2018). ISSN: 1664-1078. DOI: [10.3389/fpsyg.2018.02086](https://doi.org/10.3389/fpsyg.2018.02086) (cit. on pp. 1, 6, 7).
 - [14] M. Cook, S. Colton, J. Gow, and G. Smith. "General Analytical Techniques For Parameter-Based Procedural Content Generators". In: *2019 IEEE Conference on Games (CoG)*. 2019, pp. 1–8. DOI: [10.1109/CIG.2019.8848024](https://doi.org/10.1109/CIG.2019.8848024) (cit. on p. 23).
 - [15] K. S. Emmanuel, C. Mathuram, A. R. Priyadarshi, R. A. George, and J. Anitha. "A Beginners Guide to Procedural Terrain Modelling Techniques". In: *2019 2nd International Conference on Signal Processing and Communication (ICSPC)*. 2019, pp. 212–217. DOI: [10.1109/ICSPC46172.2019.8976682](https://doi.org/10.1109/ICSPC46172.2019.8976682) (cit. on pp. 19, 24).
 - [16] J. A. Fisher, A. Garg, K. P. Singh, and W. Wang. "Designing intentional impossible spaces in virtual reality narratives: A case study". In: *2017 IEEE Virtual Reality (VR)*. 2017, pp. 379–380. DOI: [10.1109/VR.2017.7892335](https://doi.org/10.1109/VR.2017.7892335) (cit. on pp. 15, 17, 30).
 - [17] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan. "A Systematic Literature Review of A* Pathfinding". In: *Procedia Computer Science* 179 (2021). 5th International Conference on Computer Science and Computational Intelligence 2020, pp. 507–514. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2021.01.034>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050921000399> (cit. on p. 42).

- [18] T. Fort. "Controlling Randomness: Using Procedural Generation to Influence Player Uncertainty in Video Games". In: 2015 (cit. on p. 26).
- [19] I. Fortran, W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. "Numerical Recipes". In: *Cambridge, UK, Cambridge University Press* (1992-01) (cit. on p. 45).
- [20] J. Freiknecht and W. Effelsberg. "A Survey on the Procedural Generation of Virtual Worlds". In: *Multimodal Technologies and Interaction* 1.4 (2017). ISSN: 2414-4088. DOI: [10.3390/mti1040027](https://doi.org/10.3390/mti1040027) (cit. on p. 19).
- [21] T. Gao, J. Zhang, and Q. Mi. "Procedural Generation of Game Levels and Maps: A Review". In: *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. 2022, pp. 050–055. DOI: [10.1109/ICAIIIC54071.2022.9722624](https://doi.org/10.1109/ICAIIIC54071.2022.9722624) (cit. on pp. 19, 21).
- [22] A. Gellel and P. Sweetser. "A Hybrid Approach to Procedural Generation of Rogue-like Video Game Levels". In: 2020-09, pp. 1–10. DOI: [10.1145/3402942.3402945](https://doi.org/10.1145/3402942.3402945) (cit. on p. 19).
- [23] M. Hendrikx, S. Meijer, J. Velden, and A. Iosup. "Procedural Content Generation for Games: A Survey". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 9 (2013-02). DOI: [10.1145/2422956.2422957](https://doi.org/10.1145/2422956.2422957) (cit. on p. 19).
- [24] T. Huber, S. Mertes, S. Thull, S. Flutura, and E. Andre. "Dynamic Difficulty Adjustment in Virtual Reality Exergames through Experience-driven Procedural Content Generation". In: 2021-08 (cit. on pp. 25, 26).
- [25] A. Isham and T. Jackson. "Finding flow: exploring the potential for sustainable fulfilment". In: *The Lancet Planetary Health* 6 (2022-01), e66–e74. DOI: [10.1016/S2542-5196\(21\)00286-2](https://doi.org/10.1016/S2542-5196(21)00286-2) (cit. on p. 26).
- [26] E. Jonathan, C. Roberts, S. Presentation, S. Razzaque, Z. Kohn, and M. Whitton. "Redirected Walking". In: *Proceedings of Eurographics* (2001-08) (cit. on p. 10).
- [27] G. Kelly and H. McCabe. "A Survey of Procedural Techniques for City Generation". In: *The ITB Journal* 7 (2006), p. 5 (cit. on p. 19).
- [28] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal. "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness". In: *The International Journal of Aviation Psychology* 3.3 (1993), pp. 203–220 (cit. on p. 81).
- [29] E. Langbehn, P. Lubos, and F. Steinicke. "Redirected Spaces: Going Beyond Borders". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2018, pp. 767–768. DOI: [10.1109/VR.2018.8446167](https://doi.org/10.1109/VR.2018.8446167) (cit. on pp. 29, 30).
- [30] A. Liapis. "Multi-segment evolution of dungeon game levels". In: 2017-07, pp. 203–210. DOI: [10.1145/3071178.3071180](https://doi.org/10.1145/3071178.3071180) (cit. on p. 21).
- [31] R. Linden, R. Lopes, and R. Bidarra. "Designing procedurally generated levels". In: 2013-10, pp. 41–47 (cit. on p. 19).

-
- [32] R. Linden, R. Lopes, and R. Bidarra. “Designing procedurally generated levels”. In: 2013-10, pp. 41–47 (cit. on p. 22).
 - [33] R. van der Linden, R. Lopes, and R. Bidarra. “Procedural Generation of Dungeons”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 6.1 (2014), pp. 78–89. DOI: [10.1109/TCIAIG.2013.2290371](https://doi.org/10.1109/TCIAIG.2013.2290371) (cit. on pp. 21–23).
 - [34] J. Liu, H. Parekh, M. Al-Zayer, and E. Folmer. “Increasing Walking in VR using Redirected Teleportation”. In: *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. UIST ’18. New York, NY, USA: Association for Computing Machinery, 2018, 521–529. ISBN: 9781450359481. DOI: [10.1145/3242587.3242601](https://doi.org/10.1145/3242587.3242601). URL: <https://doi.org/10.1145/3242587.3242601> (cit. on pp. 1, 32).
 - [35] D. C. Lochner and J. E. Gain. “VR Natural Walking in Impossible Spaces”. In: MIG ’21. Virtual Event, Switzerland: Association for Computing Machinery, 2021. ISBN: 9781450391313. DOI: [10.1145/3487983.3488305](https://doi.org/10.1145/3487983.3488305) (cit. on pp. 1, 8–10, 15–17, 30).
 - [36] R. Lopes and R. Bidarra. “Adaptivity Challenges in Games and Simulations: A Survey”. In: *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (2011-07), pp. 85–99. DOI: [10.1109/TCIAIG.2011.2152841](https://doi.org/10.1109/TCIAIG.2011.2152841) (cit. on p. 25).
 - [37] M. Lutfallah. “[DC] Maximizing Natural Walking in Virtual Environments”. In: *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2023, pp. 989–990. DOI: [10.1109/VRW58643.2023.00337](https://doi.org/10.1109/VRW58643.2023.00337) (cit. on p. 8).
 - [38] E. S. Martinez, A. S. Wu, and R. P. McMahan. “Research Trends in Virtual Reality Locomotion Techniques”. In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2022, pp. 270–280. DOI: [10.1109/VR51125.2022.00046](https://doi.org/10.1109/VR51125.2022.00046) (cit. on pp. 2, 9, 10).
 - [39] T. Mujber, T. Szecsi, and M. Hashmi. “Virtual reality applications in manufacturing process simulation”. In: *Journal of Materials Processing Technology* 155-156 (2004). Proceedings of the International Conference on Advances in Materials and Processing Technologies: Part 2, pp. 1834–1838. ISSN: 0924-0136. DOI: <https://doi.org/10.1016/j.jmatprotec.2004.04.401> (cit. on p. 7).
 - [40] J. F. Nash. “Equilibrium points in n -person games”. In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. DOI: [10.1073/pnas.36.1.48](https://doi.org/10.1073/pnas.36.1.48). eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.36.1.48>. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.36.1.48> (cit. on p. 38).
 - [41] J. von Neumann. “1. On the Theory of Games of Strategy”. In: *Contributions to the Theory of Games, Volume IV*. Ed. by A. W. Tucker and R. D. Luce. Princeton: Princeton University Press, 1959, pp. 13–42. ISBN: 9781400882168. DOI: [doi:10.1515/9781400882168-003](https://doi.org/10.1515/9781400882168-003). URL: <https://doi.org/10.1515/9781400882168-003> (cit. on p. 38).

- [42] C. M. Nguyen and P. J. Rhodes. “Delaunay triangulation of large-scale datasets using two-level parallelism”. In: *Parallel Computing* 98 (2020), p. 102672. ISSN: 0167-8191. DOI: <https://doi.org/10.1016/j.parco.2020.102672>. URL: <https://www.sciencedirect.com/science/article/pii/S0167819120300624> (cit. on pp. 34, 36).
- [43] N. C. Nilsson, T. Peck, G. Bruder, E. Hodgson, S. Serafin, M. Whitton, F. Steinicke, and E. S. Rosenberg. “15 Years of Research on Redirected Walking in Immersive Virtual Environments”. In: *IEEE Computer Graphics and Applications* 38.2 (2018), pp. 44–56. DOI: [10.1109/MCG.2018.111125628](https://doi.org/10.1109/MCG.2018.111125628) (cit. on p. 11).
- [44] H. Paulmann, T. Mayer, M. Barnes, D. Briddigkeit, F. Steinicke, and E. Langbehn. “Combining Natural Techniques to Achieve Seamless Locomotion in Consumer VR Spaces”. In: *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2021, pp. 383–384. DOI: [10.1109/VRW52623.2021.00079](https://doi.org/10.1109/VRW52623.2021.00079) (cit. on pp. 13–15, 17, 30).
- [45] A. Pech, P. Hingston, M. Masek, and C. P. Lam. “Evolving Cellular Automata for Maze Generation”. In: *Artificial Life and Computational Intelligence*. Ed. by S. K. Chalup, A. D. Blair, and M. Randall. Cham: Springer International Publishing, 2015, pp. 112–124. ISBN: 978-3-319-14803-8 (cit. on p. 24).
- [46] J. Plouzeau, J.-R. Chardonnet, and F. Merienne. “Using Cybersickness Indicators to Adapt Navigation in Virtual Reality: A Pre-Study”. In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2018, pp. 661–662. DOI: [10.1109/VR.2018.8446192](https://doi.org/10.1109/VR.2018.8446192) (cit. on p. 7).
- [47] R. G. de Pontes and H. M. Gomes. “Evolutionary Procedural Content Generation for an Endless Platform Game”. In: *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2020, pp. 80–89. DOI: [10.1109/SBGames51465.2020.00021](https://doi.org/10.1109/SBGames51465.2020.00021) (cit. on p. 19).
- [48] A. Prithul, I. B. Adhanom, and E. Folmer. “Teleportation in Virtual Reality; A Mini-Review”. In: *Frontiers in Virtual Reality* 2 (2021). ISSN: 2673-4192. DOI: [10.3389/frvir.2021.730792](https://doi.org/10.3389/frvir.2021.730792) (cit. on p. 10).
- [49] A. R. Rebelo, P. A. Ferreira, and R. Nóbrega. “Adaptive Virtual Environments in Small Physical Spaces: Navigation Design for Customized User Experiences in Virtual Reality”. In: *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI EA ’24. New York, NY, USA: Association for Computing Machinery, 2024. ISBN: 9798400703317. DOI: [10.1145/3613905.3650942](https://doi.org/10.1145/3613905.3650942). URL: <https://doi.org/10.1145/3613905.3650942> (cit. on pp. 1, 5, 15, 31).
- [50] A. R. Rebelo and R. Nóbrega. *Non-Euclidean Navigation: Enabling Virtual Environment Walking in Small Physical Spaces*. Tech. rep. NOVA School of Science and Technology, Department of Informatics, 2023-09. URL: <https://www.researchgate.net/publication/371125628>.

- [net/publication/387213888_Non-Euclidean_Navigation_Enabling_Virtual_Environment_Walking_in_Small_Physical_Spaces](#) (cit. on pp. 1, 8, 15–17, 30).
- [51] A. Santamaría-Ibirika, X. Cantero, S. Huerta, I. Santos, and P. G. Bringas. “Procedural Playable Cave Systems Based on Voronoi Diagram and Delaunay Triangulation”. In: *2014 International Conference on Cyberworlds*. 2014, pp. 15–22. DOI: [10.1109/CW.2014.11](#) (cit. on p. 24).
- [52] M. Shehade and T. Stylianou-Lambert. “Virtual Reality in Museums: Exploring the Experiences of Museum Professionals”. In: *Applied Sciences* 10.11 (2020). ISSN: 2076-3417. DOI: [10.3390/app10114031](#) (cit. on pp. 1, 7).
- [53] R. Smelik, K. Galka, K. J. de Kraker, F. Kuijper, and R. Bidarra. “Semantic constraints for procedural generation of virtual worlds”. In: 2011-06. DOI: [10.1145/2000919.2000928](#) (cit. on p. 20).
- [54] M. Sra, S. Garrido-Jurado, C. Schmandt, and P. Maes. “Procedurally Generated Virtual Reality from 3D Reconstructed Physical Space”. In: *VRST '16*. Munich, Germany: Association for Computing Machinery, 2016, 191–200. ISBN: 9781450344913. DOI: [10.1145/2993369.2993372](#) (cit. on pp. 28, 30).
- [55] T. su, W. Wang, H. Liu, L. Zhendong, X. Li, Z. Jia, L. Zhou, Z. Song, M. Ding, and A. Cui. “An adaptive and rapid 3D Delaunay triangulation for randomly distributed point cloud data”. In: *The Visual Computer* 38 (2022-01), pp. 1–25. DOI: [10.1007/s00371-020-02011-3](#) (cit. on pp. 24, 34, 36).
- [56] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. “A taxonomy for deploying redirection techniques in immersive virtual environments”. In: *2012 IEEE Virtual Reality Workshops (VRW)*. 2012, pp. 43–46. DOI: [10.1109/VR.2012.6180877](#) (cit. on pp. 11–13, 15, 17, 18, 30).
- [57] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas. “Impossible Spaces: Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.4 (2012), pp. 555–564. DOI: [10.1109/TVCG.2012.47](#) (cit. on pp. 2, 15–17, 30).
- [58] H. Sveistrup. “Motor rehabilitation using virtual reality”. In: *Journal of neuroengineering and rehabilitation* 1 (2005-01), p. 10. DOI: [10.1186/1743-0003-1-10](#) (cit. on p. 7).
- [59] M. B. Temuçin, I. Kocabas, and K. Oguz. “Using Cellular Automata as a Basis for Procedural Generation of Organic Cities”. In: *European Journal of Engineering Research and Science* (2020) (cit. on p. 23).
- [60] J. Togelius, A. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. Stanley. “Procedural content generation: goals, challenges and actionable steps”. In: *Dagstuhl Follow-Ups* 6 (2013-01), pp. 61–75 (cit. on pp. 19, 20).

- [61] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. "Search-Based Procedural Content Generation: A Taxonomy and Survey". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186. DOI: [10.1109/TCIAIG.2011.2148116](#) (cit. on pp. 22, 23).
- [62] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. "Search-Based Procedural Content Generation: A Taxonomy and Survey". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186. DOI: [10.1109/TCIAIG.2011.2148116](#) (cit. on p. 24).
- [63] K. Vasylevska and H. Kaufmann. "Compressing VR: Fitting Large Virtual Environments within Limited Physical Space". In: *IEEE Computer Graphics and Applications* 37.5 (2017), pp. 85–91. DOI: [10.1109/MCG.2017.3621226](#) (cit. on pp. 14, 18, 30).
- [64] K. Vasylevska, H. Kaufmann, M. Bolas, and E. A. Suma. "Flexible spaces: Dynamic layout generation for infinite walking in virtual environments". In: *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. 2013, pp. 39–42. DOI: [10.1109/3DUI.2013.6550194](#) (cit. on pp. 13, 15, 17, 18, 30).
- [65] P. Verma, S. Mahanta, S. N, and A. B. "Virtual Mouse and Keyboard for Computer Interaction by Hand Gestures Using Machine Learning". In: *International Journal of Current Science Research and Review* 06 (2023-08). DOI: [10.47191/ijcsrr/V6-i8-13](#) (cit. on p. 9).
- [66] B. M. F. Viana and S. R. dos Santos. "A Survey of Procedural Dungeon Generation". In: *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2019, pp. 29–38. DOI: [10.1109/SBGames.2019.00015](#) (cit. on pp. 20, 22, 24).
- [67] L. Wang, Y. Liu, X. Liu, and J. Wu. "Automatic Virtual Portals Placement for Efficient VR Navigation". In: *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 2022, pp. 628–629. DOI: [10.1109/VRW55335.2022.00165](#) (cit. on pp. 29, 30).
- [68] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. "Exploring Large Virtual Environments with an HMD When Physical Space is Limited". In: New York, NY, USA: Association for Computing Machinery, 2007. ISBN: 9781595936707. DOI: [10.1145/1272582.1272590](#) (cit. on p. 13).
- [69] B. G. Witmer and M. J. Singer. "Measuring presence in virtual environments: A presence questionnaire". In: *Presence: Teleoperators & Virtual Environments* 7.3 (1998), pp. 225–240 (cit. on p. 81).
- [70] S. Wu, J. Li, M. Sousa, and T. Grossman. "Investigating Guardian Awareness Techniques to Promote Safety in Virtual Reality". In: *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 2023, pp. 631–640. DOI: [10.1109/VR55154.2023.00078](#) (cit. on p. 53).

- [71] G. N. Yannakakis and J. Togelius. *Artificial Intelligence and Games*. <https://gameaibook.org>. Springer, 2018 (cit. on p. 21).
- [72] L. Yuan. “Regarding the Immersion of VR, What do We Need to do to Make it Better?” In: *Highlights in Science, Engineering and Technology* 39 (2023), 121–130. DOI: [10.54097/hset.v39i.6511](https://doi.org/10.54097/hset.v39i.6511) (cit. on p. 6).
- [73] Y. Zhang, W. Yan, and A. Narayanan. “A virtual keyboard implementation based on finger recognition”. In: *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. 2017, pp. 1–6. DOI: [10.1109/IVCNZ.2017.8402452](https://doi.org/10.1109/IVCNZ.2017.8402452) (cit. on p. 9).
- [74] Y. Zhang, G. Zhang, and X. Huang. “A Survey of Procedural Content Generation for Games”. In: *2022 International Conference on Culture-Oriented Science and Technology (CoST)*. 2022, pp. 186–190. DOI: [10.1109/CoST57098.2022.00046](https://doi.org/10.1109/CoST57098.2022.00046) (cit. on pp. 25, 27).

USER STUDY QUESTIONS

SSQ Questions (None, Slight, Moderate, Severe)

- General Discomfort
- Fatigue
- Headache
- Eye Strain
- Difficulty Focusing
- Increased Salivation
- Sweating
- Nausea
- Difficulty Concentrating
- Fullness of Head
- Blurred Vision
- Dizzy (eyes open)
- Dizzy (eyes closed)
- Vertigo
- Stomach Awareness
- Burping

IPQ Questions (7-point Likert Scale)

1. In the computer generated world I had a sense of "being there".

-
2. Somehow I felt that the virtual world surrounded me.
 3. I felt like I was just perceiving pictures.
 4. I did not feel present in the virtual space.
 5. I had a sense of acting in the virtual space, rather than operating something from outside.
 6. I felt present in the virtual space.
 7. How aware were you of the real world surrounding while navigating in the virtual world? (i.e. sounds, room temperature, other people, etc.)?
 8. I was not aware of my real environment.
 9. I still paid attention to the real environment.
 10. I was completely captivated by the virtual world.
 11. How real did the virtual world seem to you?
 12. How much did your experience in the virtual environment seem consistent with your real world experience ?
 13. How real did the virtual world seem to you?
 14. The virtual world seemed more realistic than the real world.

User Experience and Naturalness (7-point Likert Scale)

- I felt immersed in the virtual apartment.
- I felt comfortable while navigating through the virtual apartment.
- I found natural navigating through the apartment.
- I found natural the transition between rooms.

Spatial Awareness and Orientation (7-point Likert Scale)

- I was able to easily navigate through the virtual apartment.
- I had a clear sense of the room connections and layout.
- I remember the locations of specific objects or landmarks within the virtual apartment?

User Preference (Long text response)

APPENDIX A. *USER STUDY QUESTIONS*

- What did you like or dislike about walking through the virtual world?
- Do you have any suggestions for improving the navigation experience in the virtual apartment?
- Were there any aspects that felt unnatural or disorienting to you?
- Any further comments?





2024

Adaptive Procedural Generation of Virtual Reality Environments for Navigation in Limited Physical Spaces

Pedro Ferreira

NOVA

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY