De transnationale Universiteit Limburg is een uniek samenwerkingsverband van twee universiteiten in twee landen: de Universiteit Hasselt en Maastricht University.

UHASSELT
KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculteit Wetenschappen

School voor Informatietechnologie

master in de informatica

Masterthesis

Infinite spaces : procedural generation of virtual environments with self-overlapping geometry for infinite walking

Hannes Keunen
Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
Prof. dr. Philippe BEKAERT

COPROMOTOR :
Prof. dr. Fabian DI FIORE

BEGELEIDER :
dr. Jeroen PUT
De heer Bram VAN DEURZEN

2020
2021

Faculteit Wetenschappen

School voor Informatietechnologie

master in de informatica

Masterthesis

Infinite spaces : procedural generation of virtual environments with self-overlapping geometry for infinite walking

Hannes Keunen

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :

Prof. dr. Philippe BEKAERT

COPROMOTOR :

Prof. dr. Fabian DI FIORE

BEGELEIDER :

dr. Jeroen PUT

De heer Bram VAN DEURZEN

Infinite Spaces

Procedural generation of Virtual Environments with Self-overlapping Geometry for Infinite Walking

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Computer Science

by

Hannes Keunen

Supervisors:
prof. dr. Philippe Bekaert (promotor)
prof. dr. Fabian di Fiore (co-promotor)
dr. Jeroen Put (Begeleider)

Hasselt University
Faculty of Sciences
June 2021

Acknowledgements

Of course this thesis does not stand alone. I would like to thank all the people who have given me help and support while writing my thesis and developing *Infinite Spaces*.

First of all, I would like to thank my tutor, Dr. Jeroen Put for his guidance throughout the whole process. You helped me find the right direction, see the forest through the trees again when it was needed, and guided me in completing my thesis.

Along with Jeroen, I want to thank Bram van Deurzen for helping me with the user study. Your advice and feedback were invaluable to make this happen, because I had no prior experience with doing user studies.

I would also like to mention my promotor, Professor Philippe Bekaert, though he was unable to assist me due to health issues. I also want to thank Professor Fabian di Fiore for his enthousiasm when I first mentioned this as a potential subject for my thesis, because otherwise I would probably not even have started with this subject.

Finally, I want to thank my friends from my student fanfare, Ambifaarke. You gave me the distraction that I needed to keep motivated during the coronavirus pandemic, first with our online game nights, and later the rehearsals and spontaneous campfire nights.

Abstract

A major challenge in virtual reality applications, is to accommodate real walking in large virtual environments when physical space is limited. A great deal of research has already been focused on developing methods to redirect users to stay within the bounds of a limited tracking area. However, this has mostly been used in hand-crafted virtual environments. This thesis aims to develop a system for procedurally generating an infinite virtual world where users can walk around indefinitely, without stepping out of the tracking area, and without exposing the redirection techniques that are used. The resulting technique and application are called *Infinite spaces.*

To test that *Infinite Spaces* actually meets those requirements, a small user study was conducted. Test participants were asked to perform three test tasks, to evaluate the performance of the generated environments in terms of subjective sense of presence, orientation, and perception of the redirection techniques that were used. Unfortunately, due to the current coronavirus pandemic, not many people were allowed to participate in the study so the results were not as accurate as they might have been otherwise.

The pilot study did suggest that procedural generation did not have a negative impact on orientation in the virtual environment, and the environments performed quite well in terms of subjective presence, even though they are physically impossible. Participants did suspect some of the redirection techniques that were used, but none of them found this troublesome to the overall experience.

This suggests that *Infinite Spaces* is indeed a good system for infinite walking in a limited amount of physical space. Because of the limited user study, further research is required to draw definitive conclusions. Also, there are still many ways in which *Infinite Spaces* can be extended to create interesting-looking virtual worlds that are suitable for real applications.

Samenvatting

Virtual reality headsets zijn de laatste jaren wijd commercieel beschikbaar, en daarmee wordt het gebruik daarvan ook steeds populairder in virtual reality games en andere toepassingen. Maar een van de grootste uitdagingen bij het maken van zulke applicaties, is nog steeds om een geschikte manier te vinden om rond te wandelen in een virtuele omgeving. Daar bestaan verschillende opties voor, maar echt wandelen blijft de meest natuurlijke methode. Dat is natuurlijk een probleem als de virtuele wereld groter is dan de fysieke ruimte die beschikbaar is voor de gebruiker. In dat geval zouden er technieken moeten worden toegepast waarmee de gebruiker subtiel wordt omgeleid om binnen een bepaald gebied te blijven.

Een tweede uitdaging bij het maken van virtual reality-toepassingen, is simpelweg de hoeveelheid werk die nodig is om een virtuele omgeving met de hand op te bouwen. Dit is eigenlijk niet specifiek een uitdaging voor virtual reality, maar voor alle soorten toepassingen. In veel gevallen is dat natuurlijk onvermijdelijk. Denk maar een een game zoals *Super Mario Galaxy*[1], waar elk level een eigen structuur, doel, en zelfs eigen regels kan hebben. Bij dat soort toepassingen is het opbouwen van een virtuele wereld juist een onmisbaar deel van het ontwikkelingsproces.

In andere toepassingen is het juist wel mogelijk om dat handwerk, of toch een groot deel ervan, te vervangen door algoritmes die automatisch inhoud genereren. Zulke methodes kunnen worden gebruikt door een ontwikkelaar of artiest om bepaalde inhoud te genereren, die uiteindelijk wordt meegeleverd met de applicatie. Maar dat kan ook worden gebruikt door de applicatie zelf, wanneer die wordt uitgevoerd door de gebruiker. Een goed voorbeeld daarvan is het spel *Minecraft*[2], waar een een oneindige, automatisch gegenereerde wereld juist een onmisbaar deel uitmaakt van het spel.

Het automatisch genereren van oneindige werelden is een interessant idee, maar hoe kan dat worden toegepast in virtual reality-toepassingen? Het lijkt onmogelijk om oneindig rond te wandelen binnen een beperkte fysieke ruimte. Of zijn er manieren om toch de illusie van een oneindige virtuele wereld op te wekken?

Zoals eerder al kort aangehaald, bestaan er inderdaad technieken die ervoor zorgen dat de gebruiker onopgemerkt wordt omgeleid, om zo de illusie te creëren van een veel grotere virtuele omgeving dan de eigelijke fysieke ruimte. In de wetenschappelijke literatuur wordt verwezen naar die technieken met de term *redirected walking*.

De focus bij redirected walking ligt vaak bij technieken die toestaan om rond te wan-

[1]`https://en.wikipedia.org/wiki/Super_Mario_Galaxy`
[2]`https://www.minecraft.net/`

delen in omgevingen uit de echte wereld, of die tenminste mogelijk zouden zijn in de echte wereld. Een voorbeeld daarvan zou een virtuele rondleiding van een gebouw kunnen zijn. In zulke toepassingen worden meestal de bewegingen van de gebruiker gemanipuleerd, bijvoorbeeld door constant de omgeving lichtjes rond hen te laten draaien, waardoor ze in een cirkel worden geleid terwijl het lijkt alsof ze gewoon in een rechte lijn wandelen.

In andere toepassingen is het echter niet altijd nodig om omgevingen te modelleren die de regels van de echte wereld volgen. In zulke gevallen kan de structuur van de wereld zelf worden gemanipuleerd om de illusie van een veel grotere ruimte te creëren. Dat wordt gewoonlijk gedaan door subtiele veranderingen te introduceren terwijl de gebruiker afgeleid is, of door verschillende gebieden in de virtuele omgeving dezelfde fysieke ruimte te laten innemen. Zulke onmogelijkheden kunnen in veel gevallen worden geïntroduceerd zonder dat de gebruiker het meteen opmerkt, of tenminste op zo'n manier dat de illusie van een samenhangende ruimte niet wordt verbroken.

Het doel van deze thesis is dus om een methode te vinden waarmee een oneindig grote virtuele omgeving kan worden gegenereerd terwijl de gebruiker rondwandelt binnen een beperkte fysieke ruimte. Hierbij mogen de gebruikte technieken voor het omleiden van gebruikers niet duidelijk merkbaar zijn voor de gebruiker, of tenminste niet op een manier die als storend wordt ervaren. De techniek die wordt gepresenteerd, en de bijhorende applicatie, zullen *Infinite Spaces* worden genoemd.

Het doel van *Infinite Spaces* is niet om omgevingen uit de echte wereld na te maken, maar eerder om omgevingen te genereren die er op het eerste zicht realistisch uit zien, maar niet noodzakelijk dezelfde regels volgen. Dit soort omgevingen is bruikbaar voor applicaties waar de structuur van de wereld zelf niet belangrijk is, maar de inhoud wel. Enkele voorbeelden zijn een shooter game, of een virtuele trainingsomgeving voor soldaten of revalidatiepatiënten.

Om *Infinite Spaces* te ontwikkelen, werd verder gewerkt op een bestaande techniek voor redirected walking, die *flexible spaces* wordt genoemd. In *flexible spaces* wordt een virtuele wereld opgebouwd uit kleine kamers die onderling verbonden zijn met gangen. Wanneer de gebruiker van de ene kamer naar de andere wil gaan, wordt de volgende kamer telkens op een willekeurige positie in de omgeving geplaatst. Vervolgens wordt er automatisch een gang gegenereerd om die kamer te kunnen bereiken. Wanneer de gebruiker dan de volgende kamer binnen gaat, worden de vorige kamer en de gang onmiddellijk verwijderd. Door de willekeurige plaatsing van de kamers kan het zijn dat kamers en gangen met elkaar overlappen, maar door de complexiteit van de gangen wordt dat doorgaans niet opgemerkt door gebruikers.

In *flexible spaces* worden virtuele omgevingen gedefinieërd door middel van een vaste connectiviteitsgraaf. *Infinite Spaces* werkt daarop verder door ook die graaf automatisch te genereren terwijl de gebruiker rondwandelt in de virtuele omgeving. Op die manier wordt er effectief een oneindige virtuele wereld opgebouwd.

De werelden die worden gegenereerd door *Infinite Spaces*, zijn vervolgens getest in een kleinschalige pilot study. Omwille van de huidige Coronapandemie, was een volledige user study jammer genoeg niet mogelijk. De omgevingen zijn getest op vlak van orientatiegevoel van gebruikers, en detectie van overlappingen en veranderingen in de structuur van de werelden. Er werd ook kort gekeken naar de gebruikte aanwijzingen voor orientatie, en subjectief gevoel van aanwezigheid in de virtuele omgeving.

De tests werden uitgevoerd in het demolokaal van het Expertisecentrum voor Digitale Media (EDM) in Diepenbeek, met een HTC Vive headset en één Valve Index controller om te interageren met elementen in de omgeving. De ruimte in het lokaal was beperkt tot een speelveld van 4×4 meter. Aangezien de test binnen plaats vond, moesten deelnemers een mondmasker dragen, en werden er ramen open gezet om de ruimte te verluchten. De headset, de controller en de laptop om de vragenlijst in te vullen, werden tussen de tests door ontsmet om besmetting te vermijden.

Ondanks het kleine aantal testpersonen, heeft de pilot study toch voor een aantal interessante inzichten gezorgd. Het eerste doel van de studie was om erachter te komen of het automatisch genereren van de omgeving al dan niet een negatieve invloed heeft op het orientatiegevoel van gebruikers. Uit de resultaten leek het erop dat dat niet het geval is, maar een studie met meer deelnemers is nodig om sluitende conclusies te kunnen trekken.

Door de beperkte hoeveelheid fysieke ruimte die beschikbaar was voor de tests, hadden alle testpersonen het vermoeden dat sommige ruimtes in de omgeving met elkaar moesten overlappen. Een van de deelnemers zei ook dat hij eerder al had gelezen over het gebruik van overlappende ruimtes voor redirected walking. Toch werden overlappingen nooit als storend ervaren. Achteraf werd ook telkens gevraagd om een plattegrond te tekenen van de omgeving. Daaruit bleek dat, afgezien van de persoon die hier al voorkennis over had, gebruikers zwaar onderschatten in welke mate ruimtes met elkaar overlappen.

Uit de studie bleek ook dat veranderingen in de omgeving door de meeste gebruikers niet werden opgemerkt. Duidelijke veranderingen vinden plaats wanneer een gebruiker een kamer binnengaat, en vervolgens terug gaat via dezelfde gang. Normaal wordt de gang dan opnieuw gegenereerd, waardoor die een andere vorm kan hebben. Eén deelnemer heeft dat wel meerdere keren opgemerkt. Een goede verbetering is dus om altijd de structuur van de laatste gang die de gebruiker gevolgd heeft, te onthouden.

Als laatste werd ook kort gekeken naar aanwijzingen voor oriëntatie. Tijdens de pilot study stond in elke kamer van de virtuele omgeving een object in een bepaalde kleur. Elke deur kreeg de kleur van de kamer waar die naartoe leidde. Op die manier konden gebruikers zich oriënteren in de omgeving. Uit de studie bleek dat die kleurcodering voor de meeste gebruikers voldoende was om hun weg te vinden. Maar voor een echte applicatie is het waarschijnlijk niet echt wenselijk om felgekleurde objecten doorheen de omgeving te plaatsen. Verder onderzoek zou kunnen uitwijzen wat nog meer goede aanwijzingen zouden kunnen zijn voor orientatie in dit soor virtuele omgevingen.

Hoewel *Infinite Spaces* op zich al een vrij flexibel en krachtig systeem is, zijn er nog veel mogelijke verbeteringen en uitbreidingen mogelijk. In de huidige implementatie ziet elke kamer er vrij eenvoudig uit, en zijn alle kamers ook ongeveer hetzelfde. Een eerste mogelijke uitbreiding zou zijn om meer variatie toe te laten in de grootte, vorm en inkleding van de kamers. Dat zou kunnen worden geïmplementeerd door aan elke kamer een specifiek type te geven, zoals "slaapkamer" of "keuken", en op basis daarvan de inkleding te bepalen. Afhankelijk van de toepassing zou op die manier ook functionaliteit kunnen worden toegevoegd aan kamers. Dit zou waarschijnlijk ook handig zijn als aanwijzing voor orientatie.

Een volgende mogelijke verbetering zit in de manier waarop de connectiviteitsgraaf van de virtuele omgeving wordt gegenereerd. Op dit moment heeft die graaf een boomstructuur, wat wil zeggen dat er geen cycli mogelijk zijn. In de meeste toepassingen zou het waarschijnlijk beter zijn om wel cycli toe te laten. Dat kan nog verder worden verbeterd door meer geavanceerde methoden te gebruiken voor het genereren van grafen, zoals *graph grammars* of *L-systemen.* Daarmee zouden bijvoorbeeld regels kunnen worden gedefinieerd die bepalen welke kamers met elkaar verbonden kunnen zijn en welke niet. Een keuken zou bijvoorbeeld niet verbonden kunnen worden met een slaapkamer, maar wel met een vooraadkamer of een eetzaal.

Tot nu toe is *Infinite Spaces* beperkt tot binnenomgevingen, maar met slim gebruik van textures zou het misschien ook mogelijk zijn om buitenomgevingen te simuleren. Een andere mogelijkheid zou zijn om hoogteverschillen te simuleren, bijvoorbeeld door middel van een haptische simulatie van een lift.

Tenslotte zijn er nog enkele mogelijkheden voor verder onderzoek. De pilot study werd uitgevoerd op een redelijk kleine ruimte van 4×4 meter, wat ervoor zorgde dat alle kamers met elkaar moesten overlappen. Een grotere ruimte zou kunnen zorgen voor meer variatie in de layout van de omgeving, en zou er dus voor kunnen zorgen dat onmogelijkheden minder snel worden opgemerkt. Anderzijds is 4×4 meter nog steeds een vrij grote ruimte voor een gemiddelde eindgebruiker. Het zou dus ook interessant zijn om te onderzoeken wat de minimale grootte is om *Infinite Spaces* goed te laten werken. In kleinere ruimtes zou het misschien ook interessant zijn om *Infinite Spaces* te combineren met andere technieken voor redirected walking, waarbij ook de bewegingen van de gebruiker worden gemanipuleerd.

Over het algemeen is dus gebleken dat *Infinite Spaces* een zeer krachtig en flexibel systeem is, dat in veel verschillende toepassingen gebruikt zou kunnen worden. Bij gebrek aan een volledige user study, kunnen er nog geen sluitende conclusies worden getrokken, maar de pilot study heeft al enkele veelbelovende inzichten kunnen geven. Verder onderzoek zou daar nog meer duidelijkheid in kunnen brengen, en kunnen aantonen hoe *Infinite Spaces* nog verder kan worden uitgebreid en verbeterd.

Contents

1 Introduction **13**
1.1 Background: Virtual Reality . . . 13
1.1.1 Creating Virtual Environments . . . 13
1.1.2 Infinite Worlds in Virtual Reality . . . 14
1.2 Goal . . . 14
1.2.1 Applications . . . 14
1.2.2 Requirements . . . 15
1.3 Outline . . . 15

2 Related work **17**
2.1 Redirected Walking . . . 17
2.1.1 Manipulation of self-motion . . . 17
2.1.2 Manipulation of virtual space . . . 18
2.2 Procedural Content Generation . . . 20
2.3 Rendering Impossible Spaces . . . 22

3 Infinite Spaces **25**
3.1 Environment Generation . . . 25
3.1.1 Generating Rooms and Corridors . . . 25
3.1.2 Removing Rooms and Corridors . . . 29
3.2 Framework . . . 29
3.2.1 Rendering Portals . . . 29
3.2.2 Object Picking . . . 29
3.2.3 OpenVR . . . 31
3.3 Implementation . . . 31
3.3.1 Placement in virtual space . . . 31
3.3.2 Minimap . . . 33

4 Evaluation **35**
4.1 Evaluation Criteria . . . 35
4.1.1 Orientation . . . 35
4.1.2 Change detection . . . 35
4.1.3 Overlap detection . . . 36
4.2 Test setup . . . 36

4.2.1 Experimental Design . . . 36
4.2.2 Participants . . . 37
4.2.3 Apparatus . . . 37
4.2.4 Procedure . . . 37
4.3 Preliminary Results . . . 38
4.3.1 Orientation . . . 38
4.3.2 Change and Overlap Detection . . . 39
4.3.3 Subjective Presence . . . 41

5 Conclusion **43**
5.1 Extending *Infinite Spaces* . . . 43
5.1.1 Room Types . . . 44
5.1.2 Graph Generation . . . 44
5.1.3 Outdoor Environments . . . 44
5.1.4 Elevation . . . 45
5.1.5 Combining with Other Techniques . . . 45
5.2 Further Testing . . . 45
5.2.1 Tracking Area Size . . . 45
5.2.2 Orientation Cues . . . 45

A Questionnaire **47**
A.1 SUS Presence Questionnaire . . . 47
A.2 Change and overlap detection . . . 48
A.3 Orientation . . . 48

Chapter 1

Introduction

1.1 Background: Virtual Reality

With Head-Mounted displays (HMDs) being commercially available for everyone, their use is becoming increasingly popular in Virtual Reality games and other applications. However, one of the most fundamental challenges in developing Virtual Reality applications remains choosing the right method for users to walk around in a virtual environment. Different alternatives for walking in a virtual environment exist, such as using a joystick, point-and-click methods for teleporting, and real walking. Real walking seems like the preferred method when exploring a large, immersive virtual environment. But unfortunately, this is not always a viable solution due to space constraints on the user's end. If a virtual environment is larger than the actual space available to the user, it would be impossible to explore the whole environment without being redirected in some way.

1.1.1 Creating Virtual Environments

Before addressing how that first challenge can be overcome, let us first talk about a second challenge in the creation of virtual environments. This second challenge is the sheer amount of manual labor required to build large, hand-crafted environments. In applications where the structure of the environment is essential, this is often unavoidable. This does not only apply to environments for virtual reality applications, but actually to all types of virtual worlds, including video games. One could think of a video game like *Super Mario Galaxy*[1], where each level has its own distinct structure, goal, and even specific rules. In those types of applications, the creation of virtual worlds is a core part of the development process, and cannot be avoided.

In other applications, however, it may be possible to skip all or most of the manual labor by using algorithms to generate the environment, or parts of it. So-called procedural content generation methods may by used by artists to help them create assets that are later shipped by the game or application, but procedural content generation may also be used by the game itself at run-time. A good example is the game *Minecraft*[2], where an infinite, procedurally generated world is an essential gameplay concept.

[1]`https://en.wikipedia.org/wiki/Super_Mario_Galaxy`

[2]`https://www.minecraft.net/`

1.1.2 Infinite Worlds in Virtual Reality

Procedural generation of truly infinite virtual worlds like *Minecraft* does, is an exciting idea, but how would this apply to virtual reality? It would seem impossible to let players roam around freely in an infinite, procedurally generated virtual world, while they stay within the constrained space of their own living room. Or are there ways to simulate this as realistically as possible?

In fact, there exist a number of techniques to redirect users in some way, to make sure they never step out of the tracking area, while still having the illusion of a seamless virtual world that is potentially much larger than the tracking area. Those techniques are usually referred to as *redirected walking (RDW).*

Redirected walking is often focused on allowing users to walk around naturally in large environments from the real world. An example might be a virtual tour of a real building. In those particular applications, the player's motions are usually manipulated, for example to direct them in a circle by slightly, but unnoticeably, rotating the environment around them.

However, in other applications it may not be necessary to model environments from the real world. In those cases, the structure of the virtual environment itself may be manipulated, instead of the player's motions. This is often achieved by introducing subtle changes to the structure of the environment, or by letting different parts of the environment occupy the same space. With some care, such impossibilities may be introduced without immediately being exposed to the user, or at least in such a way that it does not completely break the immersion of the virtual environment.

1.2 Goal

The goal of this thesis is essentially to find a method to generate an infinitely large virtual environment while the user is walking around, all within a restricted amount of physical space. The user should be able to walk around indefinitely in this environment with an Head-Mounted Display, without stepping outside the tracking area. This should be possible without directly exposing the techniques used, or at least not in a way that is troublesome to the user, creating the illusion of a truly infinite virtual environment. The technique and corresponding application presented in this thesis, will be referred to as *Infinite Spaces.*

1.2.1 Applications

The intended purpose of *Infinite Spaces* is not to recreate environments from the real world, but to generate environments that at first glance look like they could be built in the real world, but do not necessarily follow the same rules. These kinds of environments may be useful for applications where the structure of the environment is not important, but the content is. Example applications could include a VR shooter game, or a virtual training environment for soldiers or rehabilitation patients.

1.2.2 Requirements

Environments generated by *Infinite Spaces* should conform to some requirements in terms of realism and usability. A user study was conducted to compare different techniques, and to compare *Infinite Spaces* with an existing technique.

1. **Orientation**
   Users should be able to orientate themselves in the environment without too much effort. Techniques to facilitate orientation should be implemented when needed.

2. **Realism**
   Users should not be able to notice the redirection techniques used in the environment. This includes changes to its structure and parts that overlap with each other.

1.3 Outline

The following steps were taken in order to achieve the goals described above:

1. First, a study of existing literature on redirected walking, procedural content generation, and rendering worlds with impossible geometry was performed. This is discussed in Chapter 2.

2. Based on some of the related work, an application was built that matches the requirements described in the previous section. The implementation of the application, and the framework that was used to build it, are described in Chapter 3.

3. A user study was performed to compare different techniques against the requirements described in the previous section, and to compare *infinite spaces* with an existing technique. The exact goals, setup and results of this user study are described in Chapter 4.

4. Finally, based on the results from the user study, some potential options for future extensions to *Infinite Spaces* and further research on infinite virtual worlds are discussed in Chapter 5.

Chapter 2

Related work

This chapter discusses some of the existing work this thesis builds upon. Section 2.1 discusses existing techniques for redirected walking. In Section 2.2, some existing techniques for procedural content generation are discussed. Finally, Section 2.3 discusses the use of virtual portals for rendering scenes, and how this can be used to build impossible environments.

2.1 Redirected Walking

Several different methods for walking in virtual environments exit. Possible techniques include real walking, flying, walking-in-place, etc. It has been shown that real walking is the preferred method in terms of subjective presence [27], orientation, and reducing motion sickness [4]. A collection of techniques has been proposed to accomodate real walking in large virtual environments, when physical space is limited. These techniques are collectively referred to as *redirected walking* (RDW) [19]. A taxonomy of the techniques discussed in this section is shown in Table 2.1.

Methods for RDW can be roughly separated into two categories: approaches that manipulate the user's motion in physical space, and approaches that manipulate the structure of the virtual environment itself. We may also differentiate between subtle techniques, which are usually not noticed by most users, and overt techniques, which are usually detected by the user. The use of overt techniques may be necessary when the user reaches the limits of the physical tracking space and needs to be redirected. Williams et al. [31] have proposed three such overt techniques: *freeze-backup*, *freeze-turn*, and *2:1 turn*. While those techniques obviously break the immersion of the virtual environment, they may be necessary to avoid accidents when the user steps out of the tracking area.

2.1.1 Manipulation of self-motion

In the first class of techniques, the mapping between the user's rotation and movement is manipulated. The original idea was to rotate the scene around the user so that they stay within the tracking area. The original paper on Redirected Walking described a simple method where users were directed towards some objects in the scene. Each time they reached an object and turned around to face the next object, the scene was unnoticeably

Redirected Walking	
Manipulation of self-motion	Rotational Gains [19] Translational Gains [9] Curvature Gains [13] Bending Gains
Manipulation of virtual space	Change Blindness Redirection [24] Impossible Spaces [25] Flexible Spaces [28]
Overt techniques	freeze-backup, freeze-turn, 2:1 turn [31]

Table 2.1: Taxonomy of redirected walking techniques

rotated to amplify the user's rotation and allow them to reach the next object while staying within the tracking area. This is shown in Figure 2.1.

Another simple example is the so-called *seven-league-boots* technique [9]. This technique scales the user's steps, increasing the size of the virtual environment relative to the size of the tracking space.

When *curvature gains* [13] are employed, the user has the illusion of walking in a straight path while actually walking along a curved path in physical space. At a speed of 0.75m/s, users are able to walk along a circle of approximately 10m without noticing. This detection treshold increases at higher walking speeds.

Figure 2.2 summarizes the different types of gains that can be used to manipulate the player's motion. Other reorientation methods include subtly rotating the scene while the user's attention is captured by some distractor [7], or even during eye blinks [2].

2.1.2 Manipulation of virtual space

The second category consists of approaches that manipulate the structure of the environment itself. E. Suma et al. proposed *impossible spaces* [25], an approach which uses self-overlapping architecture. This allows rooms in an indoor virtual environment to occupy the same physical space, and therefore compresses large indoor environments in a much smaller physical space. Figure 2.3 illustrates the "expanding room" technique demonstrated in [25], where rooms are expanded to maximally fill the available physical space with 50% overlap. Further research has shown that the shape and length of the corridor connecting two overlapping rooms, influences the user's ability to actually detect that the rooms overlap [29].

In the paper on *impossible spaces*, the authors also refer to earlier work where users are transported from one location to another within the VE through virtual portals [3, 22].

A second approach is to exploit *change blindness* [20], or the user's inability to notice subtle changes to the environment, when distracted by a simple task. E. Suma et al. [24] used this to change the location of doors and corridors in a VE, creating the illusion that users were in a large office building, while they were actually walking in circles in a small tracking space of 4.3m × 4.3m. This technique is illustrated further in Figure 2.4. A

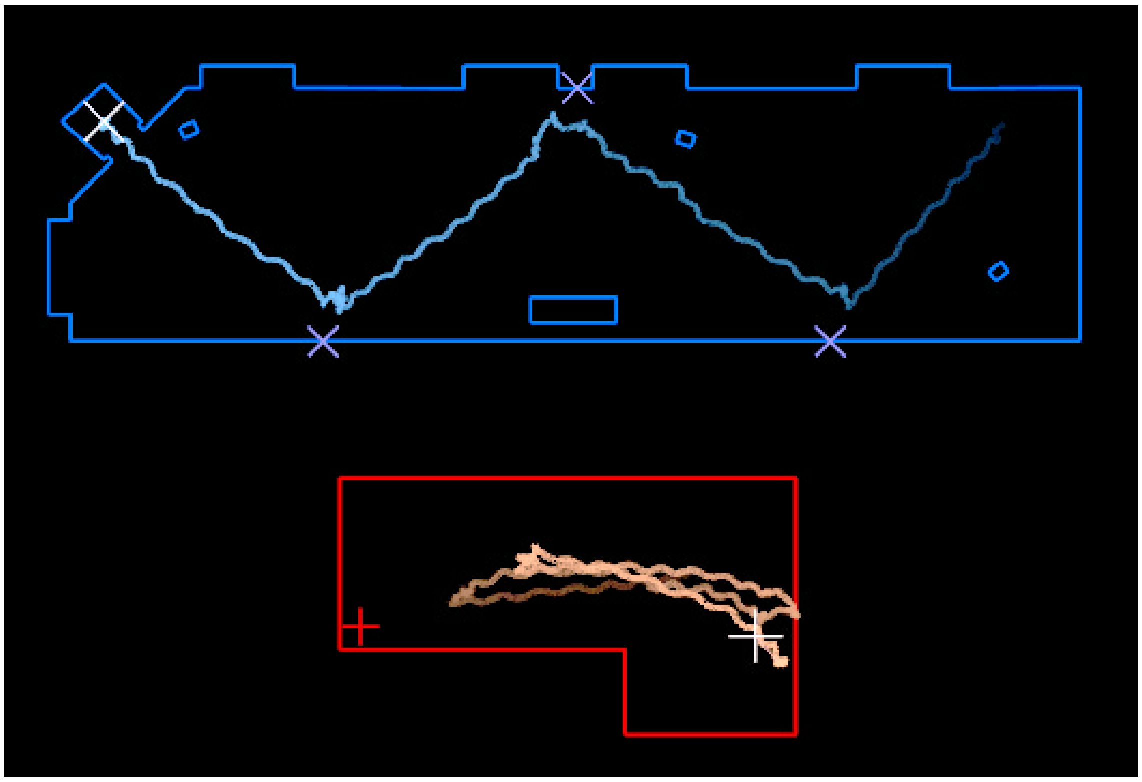


Figure 2.1: Paths taken by users in the original RDW paper. The top (blue) shows the view of the virtual environment. The bottom (red) shows their actual path in the tracking space. The blue crosses in the top image indicate locations of objects that users are directed towards. Each time a user reaches such an object and takes a 90 degree turn towards the next object, the scene rotated 90 degrees in the same direction as well, so that the user actually has to take a 180 degree turn in order to face the next object in the virtual scene. This way the user actually walks back and forth in the tracking area, as can be seen in the red image. Image from [19].

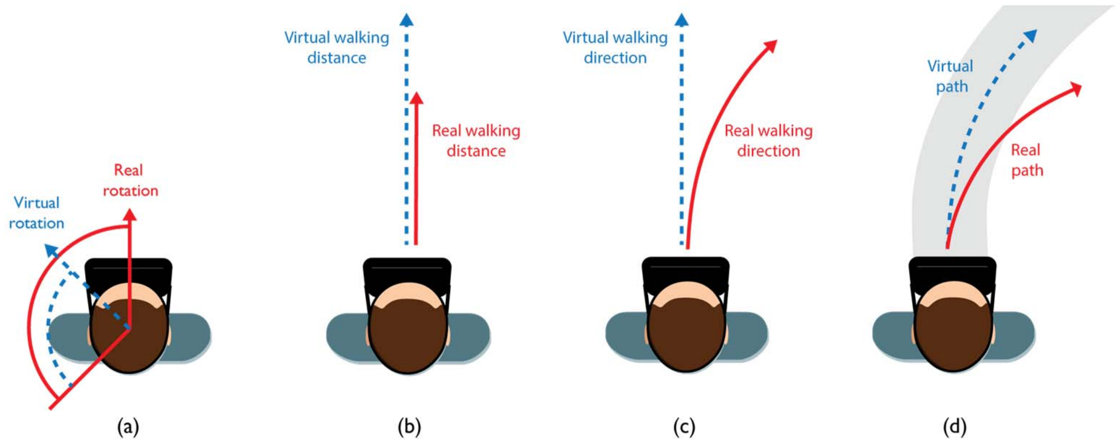


Figure 2.2: The different types of gains to manipulate the player's motion. (a) Rotational gains, as described in the original paper on Redirected Walking. (b) Translational gains, analogous to the *seven-league-boots* technique. (c) Curvature gains, where the player is redirected in a circle while it appears as if they are walking straight. (d) Bending gains, where curvatures in the player's virtual path are amplified. Images from [14].

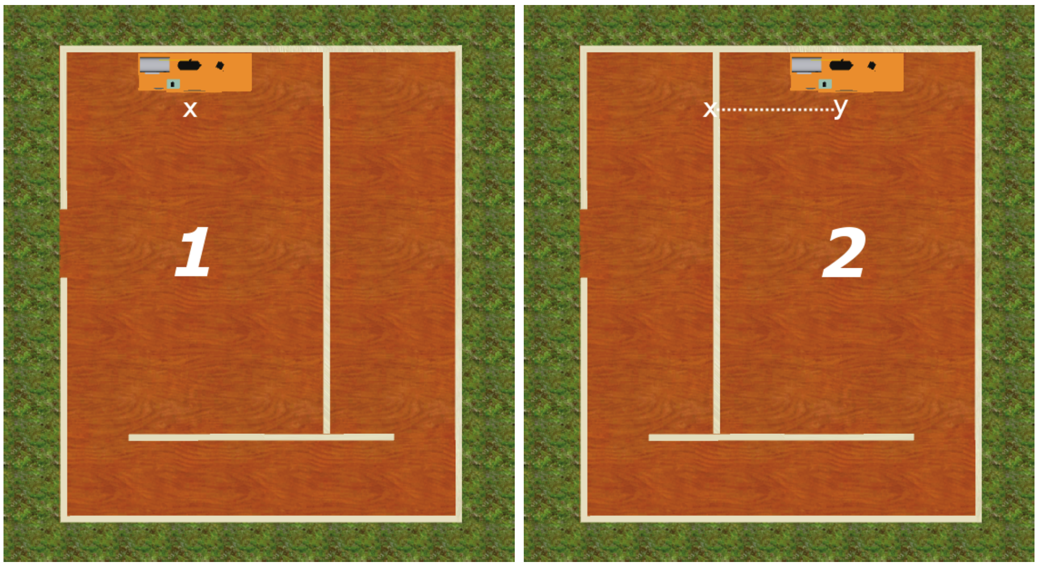


Figure 2.3: The "expanding room" approach used in *Impossible spaces* [25]. As the user walks through the corridor from room 1 to room 2, room 2 is expanded to overlap with room 1. Images from [25].

user study has shown that most users did indeed not notice the changes introduced in the virtual environment. An interesting detail is that even though changes were mostly unnoticed, most users did have the sense that they were walking in circles.

Flexible spaces [28] combines the approaches from *impossible spaces* and *change blindness redirection*, by combining the use of overlapping rooms with procedurally generated corridors which dynamically change location and shape based on the user's current position within the tracking area. The virtual environment in *flexible spaces* consists of small rooms which are interconnected by corridors. Whenever the user wants to go from one room to the next, the next room is placed at a completely random position in the tracking space. This means that rooms may even fully overlap. Next, a corridor is randomly generated to connect both rooms. The corridor is immediately deleted after the user enters the next room. Because the layout of the environment is constantly changing, *flexible spaces* avoids a buildup of knowledge about its structure. Therefore, users have to rely on other clues for orientation. This is done by color-coding rooms and the doors that lead to them.

A pilot study [30] on *flexible spaces* has shown that this is indeed a good method to create immersive virtual environments for applications where the content of the environment is more important than its structure. The authors mentioned a virtual museum as a possible example application.

2.2 Procedural Content Generation

Procedural content generation (PCG) is the use of algorithms to automatically generate content as opposed to manually creating it, and has been used in games since the early

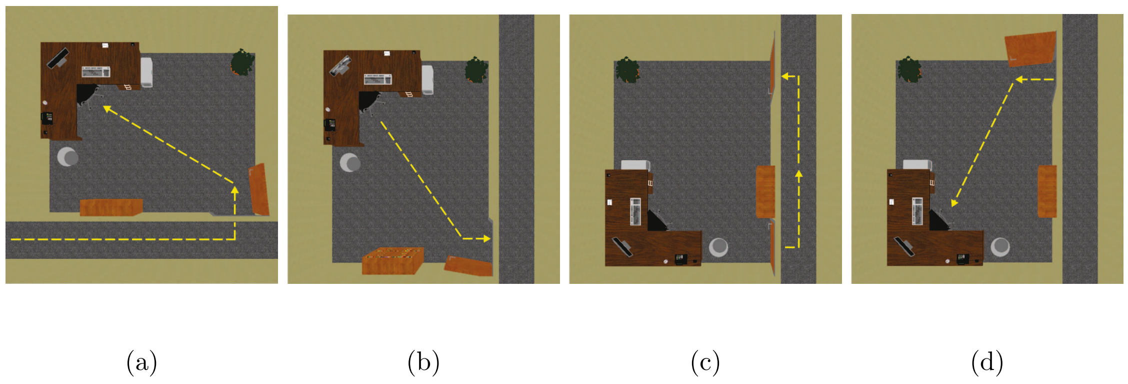


Figure 2.4: Change blindness redirection [25]. (a) The users enters the first room. (b) While the user is distracted by a simple task, the room changes shape and the door is moved to a different location. (c) The user exits the room through the new door. (d) The user enters the next room, which occupies the same physical space as the first room. Images from [24].

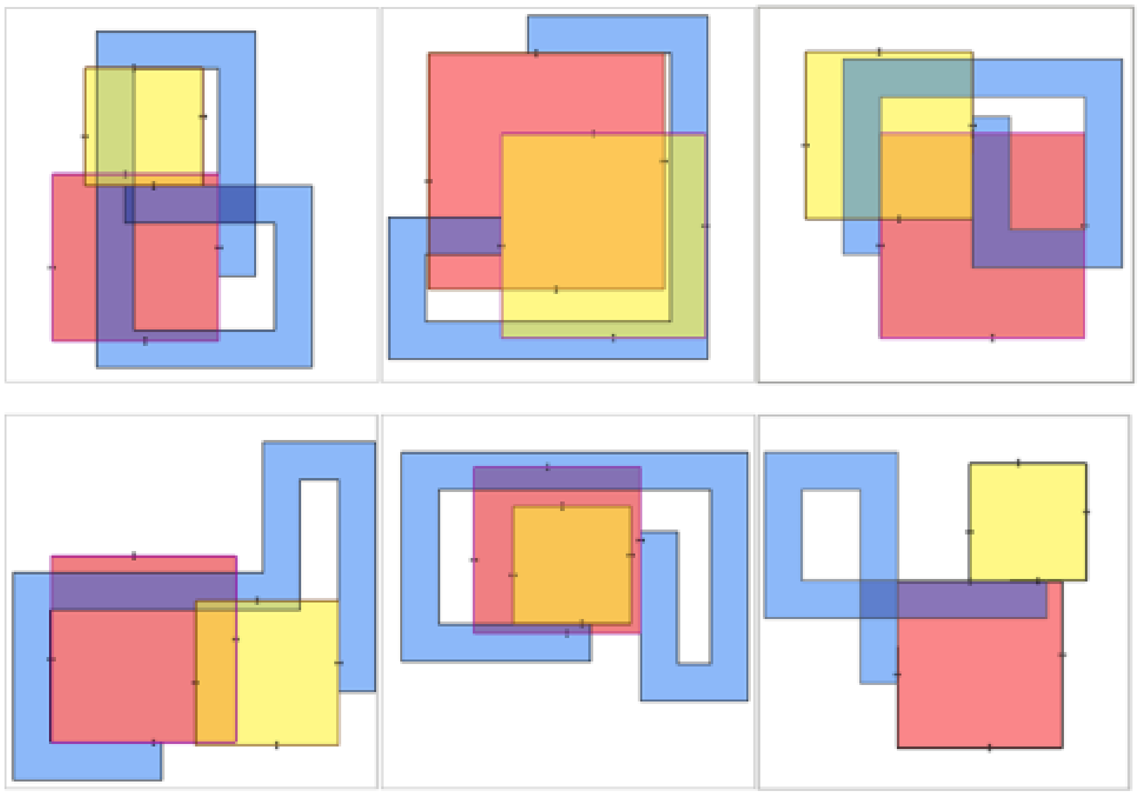


Figure 2.5: Examples of layouts generated by the *flexible spaces* algorithm. Image from [28].

eighties. Early games using PCG include *Rogue*[1] (1980) and *Elite*[2] (1984). Some modern-day titles are *Minecraft*[3] and *No Mans's Sky*[4]. PCG can be used dynamically at run-time, where content is generated as the user requires it (such as world generation in *Minecraft*), or by artists during development to aid the creation of regular content to be shipped by the game. PCG methods have also been employed in architectural applications, where buildings [11, 12], indoor environments [5, 8] and even cities [12, 15] can be procedurally generated.

Traditional PCG methods include, but are not limited to, pseudo-random number generators (PRNGs), noise algorithms [16, 17], or generative grammars like shape grammars [23], graph grammars [6] and L-systems [18]. Generative grammars are often used in architectural applications. An example of such a generative grammar is *CGA Shape* [12], a shape grammar which can efficiently generate massive cities with high level of detail. L-systems, which are usually used for modelling plants, have also been used for modelling buildings [15].

A system that generates rooms of a single-story residential house, within a given floor-plan, is given in [11]. This is achieved by using a graph grammar to generate a graph where each node represents a room. When generating the room graph, each room is given a specific type, like *foyer* or *bathroom*. After generating the room graph, each room is given an exact location and size within the house.

2.3 Rendering Impossible Spaces

One way to render impossible virtual environments such as those described in *impossible spaces* [25], is through the use of virtual portals. The use of virtual portals in computer graphics was originally intended as a visibility detection method for rendering indoor environments [1, 10, 26]. In this method, an indoor environments consists of a set of cells, connected by portals. Visible cells are then rendered in a hierarchical order, starting from the cell the user is currently in. If a cell contains a portal to a different cell, the content of that cell is rendered as well, but is clipped by the portal. This is done recursively until no more portals are visible. This method makes sure that only potentially visible objects are rendered instead of the whole scene, and could dramatically improve rendering performance in large environments.

The original use of portals in computer graphics can easily be adapted to turn a portal into a mirror, or even to make it 'look at' a completely different location in the environment. Games like *Portal*[5], *Antichamber*[6] and *Manifold Garden*[7] use this technique to allow players to teleport through the environment, and to create impossible, Escher-esque environments.

[1] `https://en.wikipedia.org/wiki/Rogue_(video_game)`
[2] `https://en.wikipedia.org/wiki/Elite_(video_game)`
[3] `https://www.minecraft.net/`
[4] `https://www.nomanssky.com/`
[5] `https://store.steampowered.com/app/400/Portal/`
[6] `https://store.steampowered.com/app/219890/Antichamber/`
[7] `https://manifold.garden/`

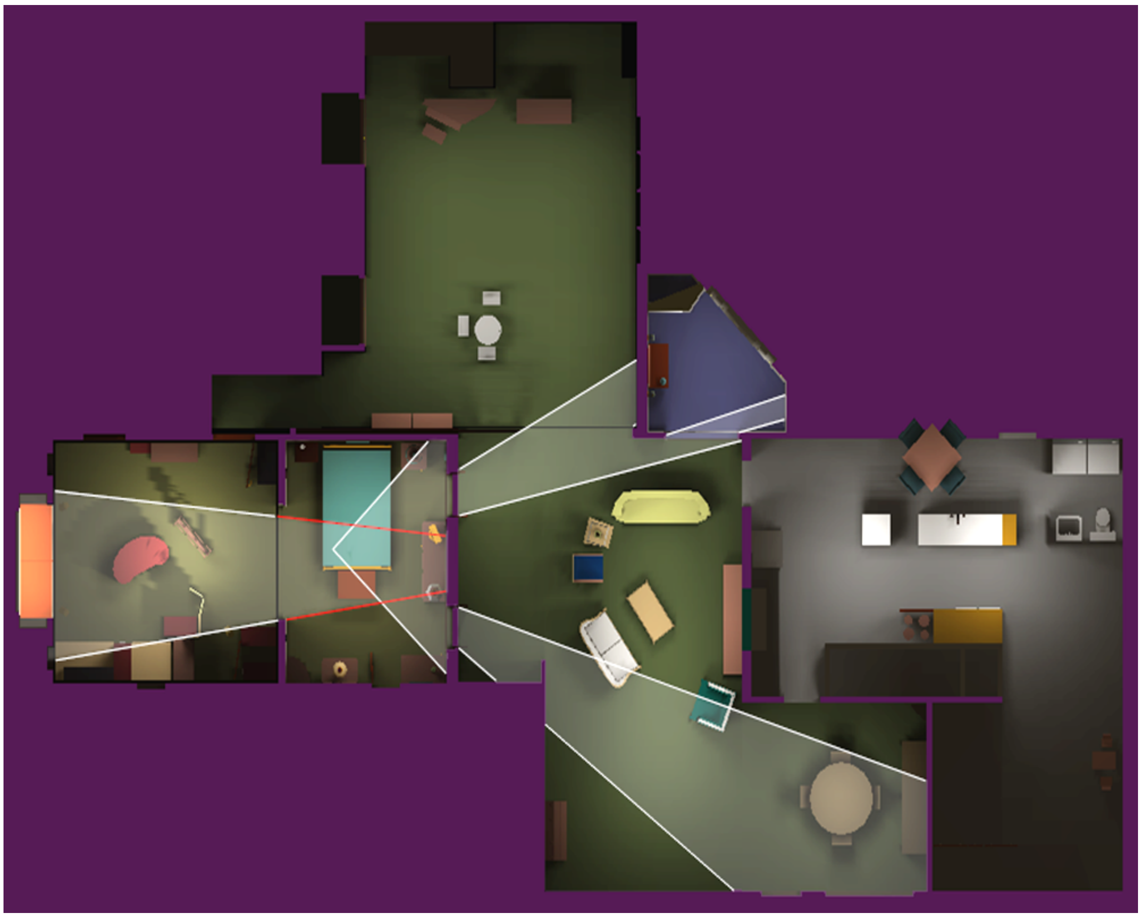


Figure 2.6: The cells-and-portals method for rendering indoor environments. The player is located in the second room from the bottom-left, where the white lines converge. The white lines indicate which parts of the environment are visible and which parts are not. The red lines indicate a part of the environment that is visible through a reflection in a mirror. Only cells (rooms) that are partly visible, are rendered. Image from [10].

Chapter 3

Infinite Spaces

The implementation of *Infinite Spaces* is largely based on the *flexible spaces* [28] paper. The main difference is that new rooms are generated on the fly as the user is walking, instead of using a fixed set of rooms. The generated environment will have a tree structure, where from each room, the user can either go back to the previous room, or continue to a room that has not been visited before. New rooms can be generated indefinitely. As in flexible spaces, each room is square, and two rooms are always connected through a randomly generated corridor.

Section 3.1 gives a high-level explanation of how rooms and corridors are generated. Section 3.2 proceeds by explaining which framework was used and how it works, how this could be used to build impossible worlds like those in *Infinite Spaces*, and how it was extended to suit all the needs of *Infinite Spaces*. Section 3.3 then explains how this was applied to actually implement the application.

3.1 Environment Generation

When the application is started, the environment consists of only a single square room, with a random amount of doors. Each of those doors will open up to a corridor leading to a different room once the user clicks on it. Walking through the environment by opening doors and entering new rooms, will gradually build up an environment graph. Since doors can only either lead to the previous room, or to a new room, this graph actually has a tree structure.

3.1.1 Generating Rooms and Corridors

Whenever the user clicks on a closed door, a new corridor to the next room is generated. If the next room has not been visited before, its node in the environment tree must be generated first. When generating a new node, the application chooses at least one and up to three sides where the room is connected to subsequent rooms. The next room is then assigned a position in physical space. As in *flexible spaces*, the position of each room is chosen randomly, so rooms may even fully overlap. Finally, a corridor is generated between the current room and the next. This is also very similar to the algorithm from *flexible spaces.* The complete process of opening a door and placing a room and a corridor is as follows:

1. The user clicks on a closed door in the current room.
2. If the door's target room has not been visited before, its node is generated first.
   (a) At least one, and up to three connections to subsequent rooms are chosen in the new room.
3. The target room is placed at a random position in the environment. The only constraint is that the space between the target room and the border of the tracking area is equal to or greater than the width of a corridor, so it is possible to place a door on any side of the room.
4. The position of the opened door is taken as a starting point S of the corridor.
5. An intermediary point I is selected randomly. I may not be inside or behind the current room or inside the target room, and the horizontal and vertical distance from S must be equal to or greater than the corridor width. If no suitable position for I is found, the algorithm returns to step 3.
6. An additional point $a0$ is picked to connect the points S and I. The only constraint is that $a0$ does not lie inside or behind the first room, because otherwise it can only be reached by taking a 180 degree turn in S.
7. The end point E of the corridor is selected as the door in the target room that is closest to I. E must also be reachable from I without taking a 180 degree turn, and without intersecting with the second room, and the horizontal and vertical distance between I and E must be equal to or greater than the corridor width. If no suitable choice for E is found, the algorithm returns to step 3.
8. A second additional point $a1$ is picked to connect the points I and E. Because of the requirements for E, there is always a suitable position for $a1$.
9. The corridor mesh is generated and placed in the environment.

The algorithm is illustrated in more detail in Figure 3.1. Figure 3.2 shows an example where a valid I point was picked in step 5, but no valid E point could be found in step 7. When this happens, the algorithm simply restarts from step 3, and picks a new position for the room.

It should be clear from Figure 3.1 that there are at least some instances where the algorithm can find a suitable corridor layout. But are there cases where it is impossible to find a valid layout? From experience, it seemed like this never caused any problems, even with a rather small tracking area. The user study was conducted with a tracking area of 4×4 meters and no problems were encountered where no corridor could be generated. Figure 3.3 illustrates why, for suitable room and tracking area sizes, a valid corridor layout can always be found. This also shows how a larger tracking area can allow more flexibility in corridor layouts.

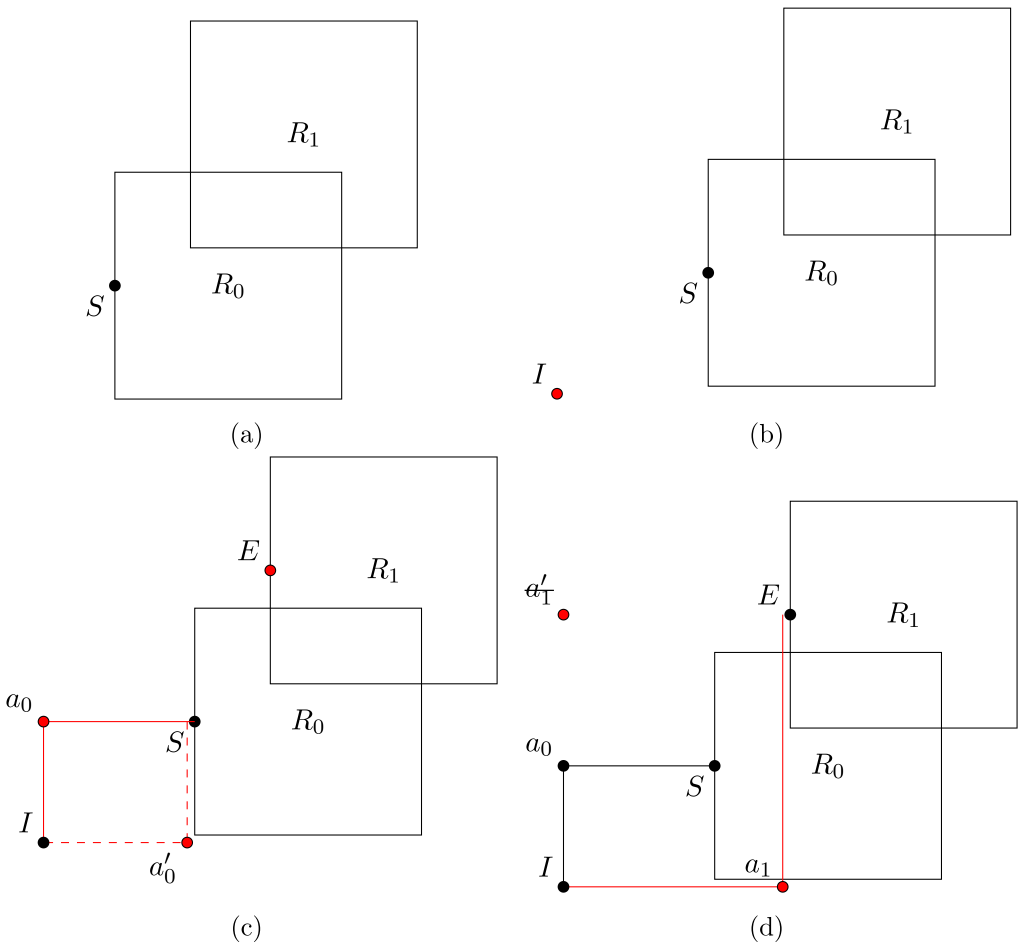


Figure 3.1: The algorithm for generating corridors. (a) From room R_0, the user has selected a door leading to room R_1. R_1 is placed at a random location inside the tracking area. The corridor's start point is located at S. (b) An intermediate point I is selected randomly, but outside both R_0 and R_1. (c) A first a point is selected between S and I. Both a_0 and a'_0 are valid points, but a_0 is selected randomly out of the two. E is selected as the end point of the corridor because it is the closest potential end point to I. (d) a second a point is selected between I and E. Since a'_1 cannot be reached directly from I without doing a 180 degree turn, a_1 is the only valid option.

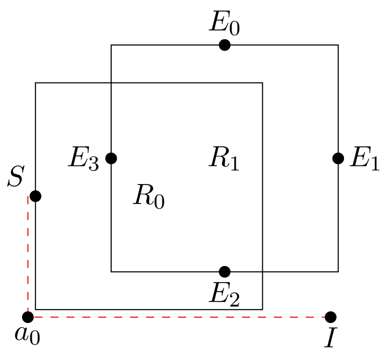


Figure 3.2: An instance where no valid E point could be found. Possible E points are evaluated in a counterclockwise order starting from E_0. E_0 is invalid because a corridor from I to E_0 would intersect with R_1. E_1 and E_2 are invalid because they are too close to I along the z and x axes respectively. E_3 is invalid because it can only be reached from I by taking a 180 degree turn.

.

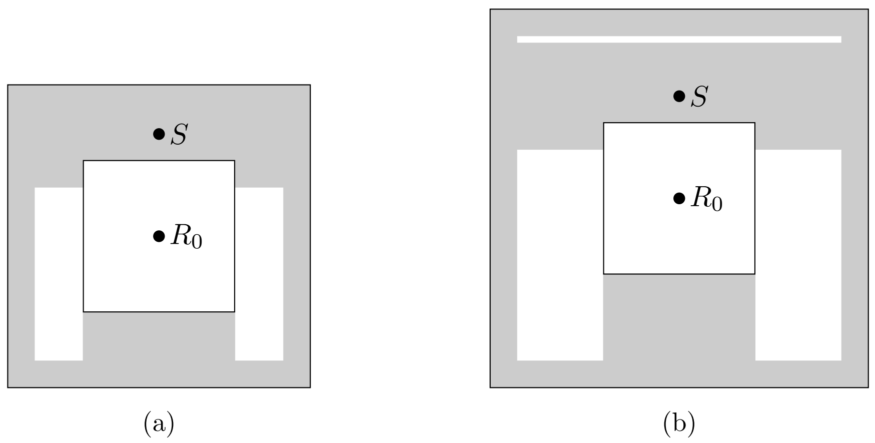


Figure 3.3: Two examples of simple environments with the positions where an intermediate point can *not* be placed marked in gray. (a) uses a tracking area of 4×4 meters, and (b) a tracking area of 5×5 meters. Both use a room size of 2×2 meters and a minimal corridor length of 0.7 meters. This does not take the placement of the second room into account, but intuitively, because a room can never cover all available space for I, it should be clear that there is always some valid position for I.

3.1.2 Removing Rooms and Corridors

In *flexible spaces*, corridors are immediately removed after a user enters a new room. So even when the user immediately wants to returns to the same room, the corridor is completely regenerated. This avoids a buildup of knowledge about the structure of the environment, because otherwise overlaps may be detected more easily. But, as was already pointed out by the authors of *flexible spaces*, a disadvantage is that those changes to the corridor layout can easily be detected by users. To avoid this, they suggested that it may be better to keep the last corridor, and only delete it when the user opens a new one.

Both strategies are actually implemented in *infinite spaces*. As will be explained in Chapter 4, one of the goals of the user study was to find out if keeping the last corridor does indeed reduce the risk of users noticing the change in the corridor's layout.

3.2 Framework

The application is implemented using a framework that is publicly available online[1]. The framework can be used to build a virtual world that consists of objects and portals. Portals are used as described in section 2.3, to teleport the player to different places in the virtual world. With a bit of thought, this can be used to build an environment where different parts can overlap with each other.

The idea is that the virtual environment is subdivided in separate sections that do not overlap with themselves, but can overlap with each other. Each section is then placed at different positions in the framework's virtual coordinate system so that they do not interfere with each other, and are connected through virtual portals. This is demonstrated in Figure 3.4.

3.2.1 Rendering Portals

The main purpose of the framework is to render a world with virtual portals. When rendering a portal, the part of the scene that is visible through the portal, is rendered to an off-screen texture first. To do this, the camera is first transformed by the matrix associated with the portal, and then the view frustrum is clipped to make sure that only the parts of the scene that are visible through the portal, are rendered. The rendered texture is then mapped on a rectangle at the position of the portal in the scene. This is done recursively up to some predefined maximum recursion depth.

3.2.2 Object Picking

In *infinite spaces*, players should be able to click on doors to open them. Unfortunately, the original framework did not have a feature where players could click on objects. So this had to be implemented manually with OpenGL. To achieve this, each object was given a unique identifier. Each frame, the identifier of each visible object is rendered to an off-screen texture at the on-screen coordinates where it should be visible. When the

[1]https://github.com/HackerPoet/NonEuclidean

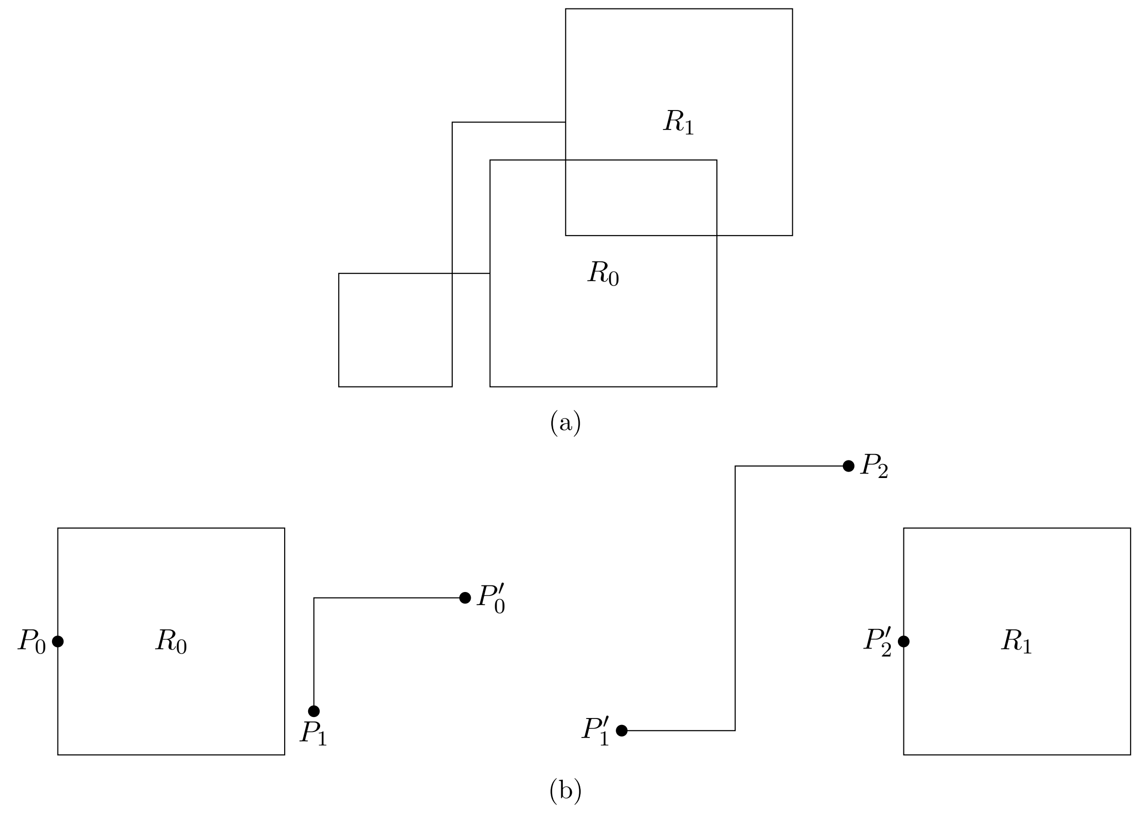


Figure 3.4: (a) The floorplan of a simple virtual environment consisting of two overlapping rooms R_0 and R_1, and a corridor between them. Notice that the corridor also overlaps with itself. (b) To construct the VE from (a), it is decomposed into four separate sections. Each room is its own section, and the corridor is also split in two section because it overlaps with itself. Each section of the VE is placed at a different position in the virtual world, and the player is teleported between them through portals. This creates the illusion of a seamless virtual world with areas that overlap with each other.

user presses the left mouse button (or uses the trigger on their controller when using an HMD), the framework reads the pixel at the center of this identifier texture. If a valid object identifier is read, the framework assumes that this is the object that the player has clicked on.

3.2.3 OpenVR

The original framework was intended for desktop applications, and not for virtual reality. Because it was directly implemented using C++ and OpenGL, support for OpenVR had to be implemented directly with C++ and OpenGL as well. This is where it became clear that it would probably have been a better choice to use a game engine like Unity, because it turned to be quite a lot of work to get OpenVR to work with the framework.

3.3 Implementation

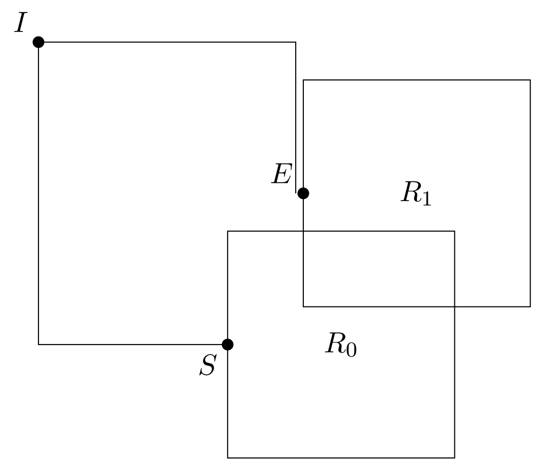
Section 3.1 described how rooms and corridors are layed out in physical space. The final piece of the puzzle is to find a way to give every object a position in the virtual coordinate system, so that everything can be rendered without objects interfering with each other.

3.3.1 Placement in virtual space

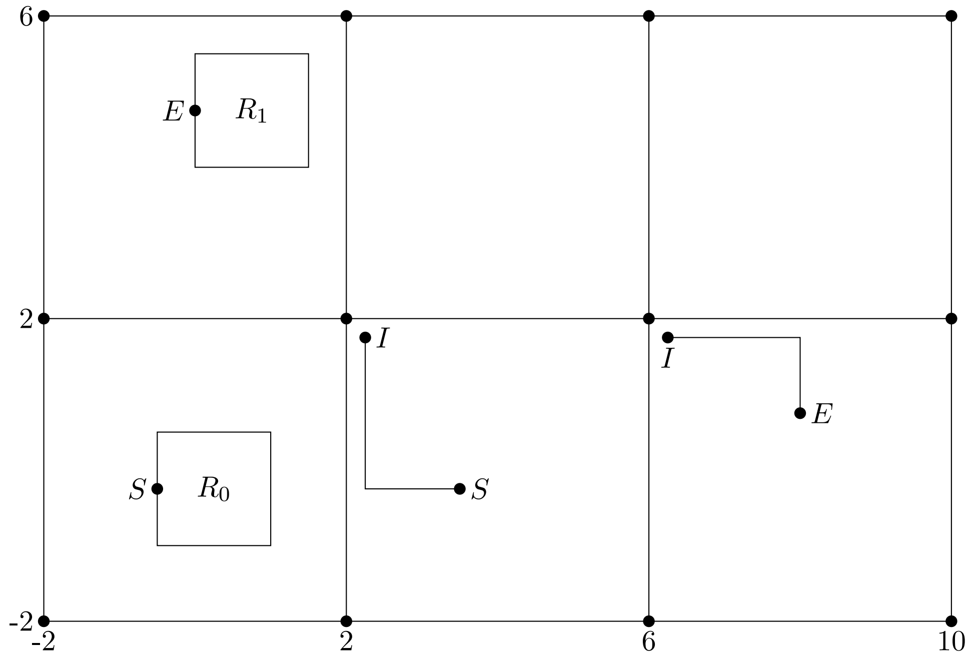
First, the virtual environment is decomposed in different sections, as was explained in Figure 3.4. Next, each section had to be assigned some position in virtual space. This is done as demonstrated in Figure 3.5. The virtual coordinate system is subdivided in cells of the same size as the physical tracking area. Each section of the virtual environment is placed in its own cell. This makes sure that overlapping objects never interfere with each other.

Internally, rooms and corridors are kept in simple arrays. When placing a room or a corridor, its cell is picked based on its index in the array. Rooms are always placed in cells along the $x = 0$ axis. So, the first room is placed in the first cell along the $x = 0$ axis, the second room in the second cell, and so on. The same happens with the first and second part of each. For example, with a tracking area of 4×4 meters, the first part of each corridor is placed in cells along the $x = 4$ axis, and the second part along the $x = 8$ axis. This also has the advantage that the position of the player in virtual space can at any time be reduced to the coordinates of the cell they are currently in, and thus to the room the player is currently in.

Say for example that the player is at the virtual coordinates $(1.5, 9)$, in a tracking area of 4×4 meters. If we divide those coordinates by the room size, and then round up or down to the nearest integer, we obtain the cell coordinates $(0, 2)$. Because the x-coordinate is 0, and rooms are placed along the $x = 0$ axis, we now know that the player must be in a room. Because the y-coordinate of the cell is 2, the player must be in the second room. This was a good way to determine the index of the current room after the player was teleported through a portal.



(a)



(b)

Figure 3.5: Placement of the different sections of a virtual environment in the virtual coordinate system. (b) shows how the environment from (a) would be layed out by the application. The virtual coordinate system is first subdivided in cells of the size of the physical tracking space, which is 4×4 meters in this case. Each section of the virtual environment is placed in its own cell, at the position it would have in physical space, but relative to the center of the cell. Rooms are placed along the $x = 0$ axis, the first part of each corridor in the second cell along the x axis, and the second part of each corridor in the third cell along the x axis.

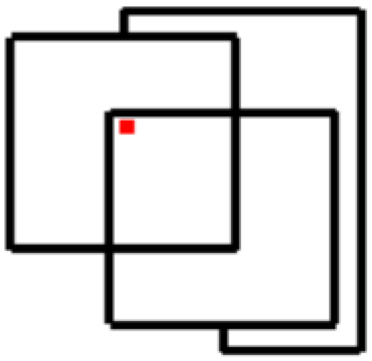


Figure 3.6: The minimap, showing two overlapping rooms connected by a corridor. The red point indicates the current position of the player.

3.3.2 Minimap

Before testing with an head-mounted display, there was no way of knowing that the player did actually stay within some limited amount of physical space. Therefore a small map of the environment was displayed in the top-right corner of the screen. This map simply shows the player's position, and all rooms and corridors that are currently active inside the environment. While this was not part of the final application, it was still a great tool for debugging the generation and placement algorithms from Sections 3.1 and 3.3. Figure 3.6 shows an example of what this looks like in the application.

Chapter 4

Evaluation

A user study was performed to evaluate the performance of *infinite spaces* in terms of orientation, change detection and overlap detection.

Section 4.1 describes those three criteria in more detail, and how they could be evaluated. Section 4.2 describes how a set of test cases was derived from those criteria, and how the actual tests were performed. Section 4.3 discusses the results of the user study.

4.1 Evaluation Criteria

Based on the goals and requirements mentioned in Section 1.2, three evaluation criteria were selected.

4.1.1 Orientation

The first criterium is the ability of users to orientate themselves in the environment. Specifically, the goal was to compare *Infinite Spaces* to *flexible spaces*, to see if and how random generation impacts the ability of users to orientate themselves in the environment. This was done by letting users walk through two variants of the virtual environment: one where the environment graph is generated randomly, and a second one where the environment graph is hard-coded. So this second test case is actually the same as *flexible spaces*. Afterwards, users were asked to rate how easy or hard it was for them to find their way in the environment. As an objective measure for orientation, the time each user needed to perform a simple retrieval task was recorded as well.

4.1.2 Change detection

A second goal was to see if the method for generating corridors from *flexible spaces* could be improved by always keeping the last corridor, and thereby reducing the chance that changes in the environment are detected by users. To test this, users were again asked to walk through two different variants of the environment. In the first variant, corridors were removed immediately after the user entered a room. In the second one, the structure of the last corridor was kept until the user opened a new corridor. The hypothesis was that, if the user decided to go back via the same corridor immediately after entering a room,

A	B	C
C	A	B
B	C	A

Figure 4.1: Latin square design for three test cases, as used in the user study.

the chance of the user noticing this change would be lower. In the *flexible spaces* paper, the authors already mentioned that this could be a possible improvement, hence why it seemed to be a good test case. To test this, users were asked about a number of things they may or may not have noticed during their time in the virtual environment.

4.1.3 Overlap detection

The third and final objective was to examine whether or not users are able to notice overlapping parts in the environment. Each time users walked through the environment, they were asked to draw a sketch of its layout. They were also asked some questions about the arrangement of the room, and whether or not the virtual environment would be possibe to build in the real world.

4.2 Test setup

Before starting the actual test, users were first asked to sign a consent form. Then, they had the chance to perform a small training task to become familiar with the structure of the environment, and the task they had to perform in the actual test cases.

During the user study, participants were asked to perform three different test cases, each corresponding to one or several of the evaluation criteria listed in the previous section. After each test case, they were asked to perform some evaluation tasks to evaluate those criteria. The order in which participants performed the different test cases, was balanced by means of a *latin square* design, as shown in Figure 4.1. This design is used to reduce order-effects, where learning from one test case can have an influence on the user's performance on the next one.

4.2.1 Experimental Design

The criteria described in the previous section were reduced to three test cases:

- TC1: An environment with a fixed connectivity graph as apposed to generating it on the fly.
- TC2: An environment with an automatically generated connectivity graph, where corridors are removed immediately after entering a room.
- TC3: An environment with an automatically generated connectivity graph, where the last corridor is always kept alive until the user opens a new one, to reduce the chance that changes in the corridor layout are noticed by users.

Like in *flexible spaces*, each room was color-coded, along with each door that led to it. Participants were specifically asked to use the color-coding as a cue for orientation. This was done so they would pay less attention to the structure of the environment, and thus hopefully be less likely to notice overlaps and changes in the environment.

The time required to perform each test case was also recorded. This was not only an objective measure for the participant's sense of orientation, but was also used as an incentive for users to perform the task as quickly as possible. So this was also a way to distract users from the structure of the environment, along with the color-coding of the rooms and doors.

4.2.2 Participants

The user study took place at the Expertise centre for Digital Media (EDM), in Diepenbeek. Unfortunately due to the Coronavirus pandemic, no external participants were allowed at the EDM. This meant that only students, researchers and EDM staff were allowed to participate in the study. This was actually a major drawback because many of the people who work at the EDM already had a lot of experience with virtual reality. One of the participants even mentioned that he had already read about the use of overlapping spaces before. A second drawback was simply the fact that not many people were available to participate in the study. Three people were able to participate; two males and one female.

4.2.3 Apparatus

The study was performed with an HTC Vive headset and one Valve Index controller for selecting doors and target objects. The size of the tracking area was 4×4 meters, which is actually much smaller than the one used for *flexible spaces*, but still large enough to generate interesting-looking environments. In this case, the size of the tracking area was constrained by the size of the demo room, and the length of the cable of the HMD.

Due to the current Coronavirus pandemic, some additional safety measures had to be taken into account. Participants obviously had to wear a face mask during the test. Between each test, the HMD, the controller and the laptop for filling in the questionnaire were properly sanitized. Also, the HMD was equipped with a disposable cover which could be replaced between each test, to facilitate sanitization. Because the user study took place in May, it was warm enough outside to open some windows to ventilate the room.

4.2.4 Procedure

For each test case, users were asked to perform a simple retrieval task. In some rooms a colored object was placed as a "target" object, which users could "pick up" by looking at it and then pressing the trigger on the controller of the HMD. In each task, users were asked to find two of those objects, and then to immediately return to the room they initially started in via the same way.

After each test case, participants were first asked to draw a sketch of the structure of the environment. The goal was to see if, and to what extent, overlaps were detected

	TC1	TC2	TC3
How difficult did you find it to orientate yourself while searching for the objects?	2.75 ± 1.3	2.5 ± 1.1	3.0 ± 1.6
How difficult did you find it to find your way back to the starting room?	2.75 ± 1.5	2.75 ± 1.5	2.75 ± 1.8

Table 4.1: Average results on the questions about orientation.

during their time in the environment. A set of colored pencils was availabe to also indicate the color of each room.

Next, they were asked to fill in the questionnaire given in appendix A. First, the *SUS presence questionnaire* [21] was used to rate their subjective sense of presence in the virtual environment. Next, they were asked to rate a number of statements about things that may or may not have happened during their time in the virtual environment, from 0 (="Did not notice or did not happen") to 5 (="Very obvious"). This list included questions about changes and other impossibilities in the environment, along with some decoy statements, to distract the participants from the actual goal of the study. Finally, the questionnaire included some statements about their sense of orientation, and the usefulness of the color-coding used in the virtual environment, which they had to rate from 1 (="Totally disagree) to 5 (="Totally agree").

4.3 Preliminary Results

Because of the small number of participants, no final conclusions could be drawn from the test study. Therefore, this is regarded more like a pilot study rather than an actual user study. But still, some interesting insights were provided.

4.3.1 Orientation

The first test criterium was to see whether procedural generation affects the ability of users to orient themselves in the virtual environment. From the pilot study, it seemed that participants did not notice any difference between test cases TC1 and TC2. This suggests that procedural generation has no effect on orientation. Table 4.1 shows the average responses to the questions about orientation for test cases TC1 and TC2. A repeated measures ANOVA test shows that there is no significant difference between the two ($p = 0.24$ for the first question and $p = 1.0$ for the second question), but some further testing with more participants is required to draw a definitive conclusion.

One of the questions stated that "It felt as if I was being turned around all the time". This does actually constantly happen intrinsically because the environment has to be generated in such a way that the user always stays inside the tracking area. One of the participants actually mentioned that this had to be the case, but that he just could not figure out when or how it happened. The answers to this question actually varied a lot, with an average score of 2.3/5 and a standard deviation of 1.2. The reason for this response is probably that this redirection happens so subtly that it is not immediately noticeable, but it just has to happen because of the small tracking area. Multiple participants also said that they did not really pay attention to the structure of the corridors. They were really just walking from room to room, and just followed the corridors because they had to, but the shape did not really matter. A reason for this may be the fact that the structure of the environment is not used as an explicit cue for orientation, in favor of the color-coding.

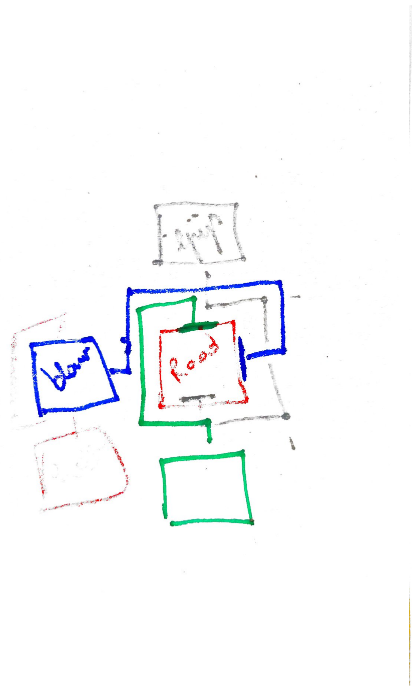
The questionnaire also included some questions about color-coding of the rooms and the doors as a cue for orientation. From the questionnaire, it seemed that this was indeed a good orientation cue. The question whether the color-coding of the doors alone was useful for orientation was answered with an average score of 4.4/5. The question whether it was also sufficient, without the coloring of the rooms, was answered much lower, with an average score of 3.4. But then the last question, which stated that the color-coding of the rooms, alongside that of the doors, was necessary as an orientation cue, had an average score of 4.1. This shows that the combination of the two was a good choice of an orientation cue for these test cases. Further research may be done to find other, maybe more subtle orientation cues for these kinds of virtual environments. One of the participants did mention that during one of the test tasks, he had only looked at the color-coding on the doors, and actually thought that the color-coding of the rooms themselves was removed.

4.3.2 Change and Overlap Detection

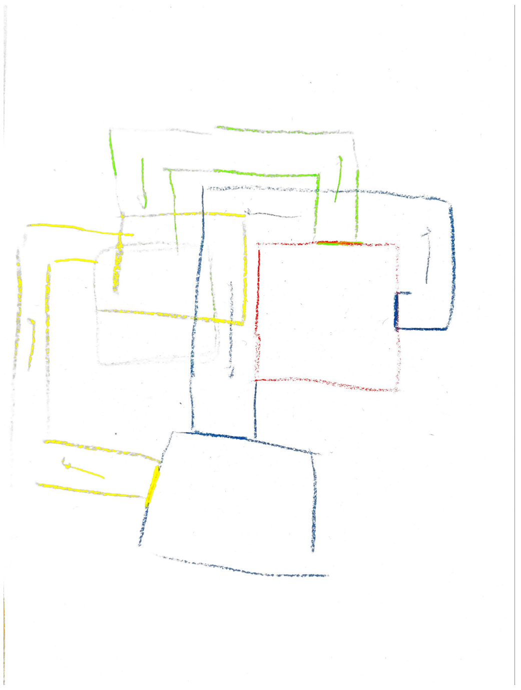
Because of the small tracking area, all participants suspected that there had to be some amount of overlap in the virtual environments. But the question about overlap perception ("Some rooms seemed to be closer to each other than should be possible.") was still answered with a rather low average confidence of 1.6/5 ($\sigma = 1.3$). Only one participant answered this with a higher confidence, but he also mentioned that he had read about techniques with overlapping spaces before.

This can also be seen in the floor plans each participant drew of the environment. Only the person that already knew about the use of overlapping spaces, drew a somewhat realistic map of the environment. This is shown in Figure 4.2c Interestingly enough, he still thought that the layout of the environment was fixed and never suspected that any changes had happened during the test. The participant that drew the floor plan in Figure 4.2a did suspect that some rooms were located in more or less the same place. However, her floor plan does not have any overlaps. Figure 4.2b does show some amount of overlap, but still underestimated a lot. This again might be because, even though users know that there has to be some amount of overlap, it is still perceived as a natural-looking environment as long as users do not pay to much attention to its structure.

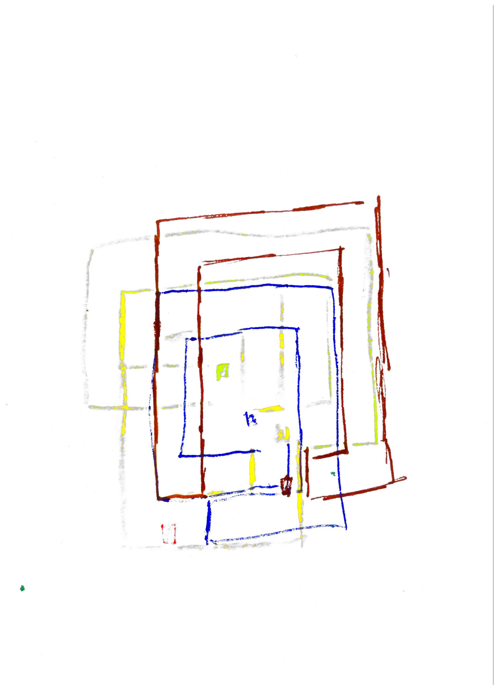
In the question about changes in the shape of the environment ("I have seen something in the virtual world change shape."), only one of the participants seemed to have noticed changes in the layout of a corridor with high confidence. All others answered this question with a score of 0(="Did not notice or did not happen"). This might suggest that most users actually do not notice any changes, even in test cases TC1 and TC2, where corridors were immediately removed when the user entered a room. On the other hand, when a



(a)



(b)



(c)

Figure 4.2: Some floor plans drawn by three different test participants. (a) has absolutely no overlap, (b) does have some overlap, but it is underestimated a lot. (c) is the closest to something that could actually be generated by *Infinite Spaces*

	Q1	Q2	Q3	Q4	Q5	Q6
$\overline{x}$	5.1	4.8	5.6	5.2	4.6	5.2
s	0.9	1.2	0.8	0.8	1.0	1.3

Table 4.2: SUS presence scores. Questions are listed in A.1

player does notice the change, it is usually with a high confidence. Further testing is required to see if this is actually the case.

4.3.3 Subjective Presence

Table 4.2 shows the average scores on the SUS presence questionnaire. A question that was particularly interesting, is Q5, where users were asked if the virtual environment is in any way similar to other places in the real world. That question was answered with an average score of 4.6. This is interesting because the worlds generated by *Infinite Spaces* are impossible, so they are definitely unlike the real world. This again suggests that even though participants knew that they were being redirected, and that rooms had to overlap with each other, this was not troublesome to them.

Chapter 5

Conclusion

This thesis described *Infinite Spaces*, a system for real walking in an infinite, procedurally generated virtual environment, while the user stays within the bounds of a tracking area. This was based on an existing technique, called *flexible spaces*, first introduced in [28]. The main contribution of this thesis was to extend *flexible spaces* to generate a truly infinite virtual environment.

Because of the Coronavirus pandemic, it was impossible to perform a full user study to test *Infinite Spaces* and to compare it to *flexible spaces*, so only a small pilot study could be done. The three major test criteria were orientation, detection of changes in corridor layout, and detection of overlapping parts of the environment. Subjective presence and the use of orientation cues were investigated shortly as well.

From the pilot study, it appeared that procedural generation probably has little to no impact on the sense of orientation. Overlaps were almost always suspected, but none of the participants seemed to find it disturbing. Changes in corridor layout were noticed in some cases, which suggests that it would indeed be a good idea to keep track of the last corridor the user has opened, or perhaps even keep a buffer of n corridors alive. The color-coding of the rooms and the doors, which were used as cues for orientation, did indeed seem to be sufficient, but some further research may still be done to find other, more suitable cues for these kinds of virtual environments.

In the following sections, some potential future work is explored. Section 5.1 lists some ways in which *Infinite Spaces* may be extended and improved to suit to different use cases. Section 5.2 discusses some possibilities for further research about virtual environments like those in *Infinite Spaces*.

5.1 Extending *Infinite Spaces*

Though *Infinite Spaces* is already a quite powerful and flexible system, there may still be many ways to extend it and make it even more flexible and adapt it to more use cases. This section explores some potential improvements and extensions.

5.1.1 Room Types

Currently, *Infinite space* only supports simple environments consisting of square rooms, without too much decorations in them. There is a system to assign a type to each room, but currently this is only used to pick the texture for the room, and to determine whether it should contain a target object or not. This may easily be extended to control many more parameters, like size and shape of the room, objects inside the room, and so on. In some applications, like games, this may even add functionality to rooms. For example, think of a roguelike dungeon game with treasure rooms, enemy rooms and a boss room. More variation between different types of rooms may also facilitate orientation in the virtual environment, and distract users even more from the impossibilities in the structure of the environment.

One of the test participants also mentioned that he found it hard to recall each individual room as a distinct space, because all rooms looked more or less the same. This may suggest that distinct room types may also have an influence on the subjective sense of presence.

5.1.2 Graph Generation

Another potential improvement, related to the room types discussed above, lies in the fact that currently, all virtual environments generated by *Infinite Spaces* have a tree structure. this means that from any room in the environment, one can only go back to its parent room, or go further to a room that was not reachable from any other previous room. It was implemented this way because it seemed easier at the time. But for most applications, it may be better to also allow loops in the environment graph.

To improve this even more, environment graph generation may be controlled by some procedural graph generation method discussed in Section 2.2. For example, a generative grammar may be used to control which types of rooms can be connected to each other and which can not, similar to how one would not expect a kitchen to be connected to a bedroom, but rather to a dining room or a pantry. Graph generation methods may also generate complete substructures of many different rooms at once instead of just one room at a time. Something like this could result in more coherent environments being generated.

5.1.3 Outdoor Environments

Although *Infinite spaces* seems to be limited to indoor environments, there may still be ways to simulate outdoor environments as well. For example, cube maps may be used to make rooms look like larger open spaces. Of course there must be some way to limit the movement of the user to stay within the actual rooms and corridors. Also, enough occluding elements must be kept in place to not expose the techniques used by *Infinite Spaces*. A simple example might be a hedge maze where intersections and larger open spaces are represented by rooms. The top of the maze might be open to the sky, which is rendered by using a cube map around the player.

5.1.4 Elevation

The idea of adding elevation to virtual environments was also shortly mentioned in [30]. The authors suggested that this might be implemented by replacing a corridor by something like a haptic elevator simulation. This would be especially useful if the tracking area is very small, because adding multiple levels can allow for a denser environment layout.

5.1.5 Combining with Other Techniques

As mentioned several times, *Infinite Spaces* is based on an existing technique, called *flexible spaces*. In [30], the authors of *flexible spaces* mentioned that this may be combined with techniques for redirected walking that manipulate the player's motion. For example, curved corridors may be generated with splines or bezier curves, and then bending gains may be added as the user is walking through such corridors.

5.2 Further Testing

Apart from the fact that there were not enough test participants, some additional testing may be useful for *Infinite Spaces*. For example, as was already mentioned in the previous section, more variation and customization between different types of rooms may also facilitate orientation in the virtual environment, and distract users even more from the impossibilities in the structure of the environment. So naturally, this would be a good test case as well.

5.2.1 Tracking Area Size

Another drawback of the user study was the limited size of the tracking area. The demo room at the EDM allowed for a tracking area of only approximately 4×4 meters, while the initial tests for *flexible spaces* were done in a much larger tracking area [28]. In fact, all test participants noticed that rooms in *Infinite Spaces* overlapped with each other, even though none of them found it disturbing. However, earlier studies [29] have shown that overlaps are detected less easily with longer, more complex corridors. Therefore, a larger tracking area may have the same effect because corridors can then also be longer and more complex. An interesting question would thus be to what extent the size of the virtual rooms and the physical tracking area affects the perception of overlap and changes in a virtual environment.

On the other hand, although still not very large, typical end users do not have a free space of 4×4 meters in their living room. So another interesting research question might be how small a tracking area can be for *Infinite Spaces* to work well.

5.2.2 Orientation Cues

In the current implementation of *Infinite Spaces* and during the user study, users oriented themselves by looking at the colors of the rooms and the doors leading to them. This is necessary because the structure of the environment itself cannot be used for orientation, since it is constantly changing. From the user study, it seemed that this color-coding was sufficient as a cue for orientation. However, it is hard to find enough distinct colors for each room, especially in an infinite environment, and it is probably not always desirable to place bright-colored objects around a virtual environment in a real-world application. An example of a different orientation cue may be to display room numbers on each door. Or maybe with more distinct room types, the layout of each room may even be a cue in itself, so it may not even be necessary to add additional cues. Though orientation cues are beyond the scope of this thesis, they were explored shortly in the user study, and may be an interesting subject for future research on this kind of virtual environments.

Appendix A

Questionnaire

During the user study, participants were asked to answer the following questions after each test case.

A.1 SUS Presence Questionnaire

The SUS (Slater-Usoh-Steed) questionnaire [21] is used to let users rate their sense of subjective presence in a virtual environment.

1. Please rate your sense of being in the virtual environment, on the following scale from 1 to 7, where 7 represents your normal experience of being in a place.
   I had a sense of "being there" in the virtual environment:
   (1) Not at all. (7) Very much.

2. To what extent were there times during the experience when the virtual environment was the reality for you?
   There were times during the experience when the virtual environment was the reality for me...
   (1) At no time. (7) Almost all the time.

3. When you think back about your experience, do you think of the virtual environment more as images that you saw, or more as somewhere that you visited? The virtual environment seems to me to be more like...
   (1) Images that I saw. (7) Somewhere that I visited.

4. During the time of the experience, which was strongest on the whole, your sense of being in the virtual environment, or of being elsewhere?
   I had a stronger sense of...
   (1) Being elsewhere. (7) Being in the virtual environment.

5. Consider your memory of being in the virtual environment. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today? By "structure of the memory", consider things like the extent to which you have a visual memory of the virtual environment, whether that memory

is in color, the extent to which the memory seems vivid or realistic, its size, location in your imagination, the extent to which it is panoramic in your imagination, and other such structural elements.
I think of the virtual environment as a place in a way similar to other places that I've been today...
(1) Not at all. (7) Very much so.

6. During the time of the experience, did you often think to yourself that you were actually in the virtual environment?
During the experience I often thought that I was really standing in the virtual environment...
(1) Not very often. (7) Very much so.

A.2 Change and overlap detection

Participants were asked to rate a number of statements about things they may have noticed in the environment, from 0 (=”did not notice or did not happen”) to 5 (”very obvious”). Some of these quetions are actually decoy questions to distract the participants from the actual goal of the study. Those are marked in cursive.

1. I have seen something in the virtual world change shape.
2. *I have seen something in the virtual world change size.*
3. *I have seen something in the virtual world move.*
4. It felt as if I was being turned around all the time.
5. *It felt as if the virtual world was turning around me.*
6. *Some rooms seemed to be larger than should be possible.*
7. Some rooms seemed to be closer to each other than should be possible.
8. *Some corridors seemed to be longer than should be possible.*

A.3 Orientation

To examine their sense of orientation, participants were first asked to rate they thought they were able to orient themselves in the environment, on a scale from 1 (=”very easy”) to 5 (=”very hard”). Again, decoy questions are marked in cursive

1. How difficult did you find it to orient yourself while searching for the objects?
2. How difficult did you find it to find your way back to the starting room?
3. *How difficult did you find it to find enough objects?*

Finally, users were asked to rate a number of statements about the usefulness of the color-coding in the environment, on a scale from 1 (=”totally disagree”) to 5(”=totally agree”)

1. The color-coding of the doors was useful to find my way in the environment.
2. The color-coding of the doors was sufficient to find my way in the environment.
3. The color-coding of the pillars, along with the doors, was useful to find my way in the environment.
4. The color-coding of the pillars, along with the doors, was sufficient to find my way in the environment.

Bibliography

[1] J. M. Airey. Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations. 1990.

[2] B. Bolte et al. Subliminal reorientation and repositioning in immersive virtual environments using saccadic suppression. 2015.

[3] G. Bruder et al. Arch-explore: A natural user interface for immersive architectural walkthroughs. 2009.

[4] S. Chance et al. Locomotion mode affects the updating of objects encountered during travel: The contribution of vestibular and proprioceptive inputs to path integration of painting and sculpture. 1998.

[5] A. Dahl and L. Rinde. Procedural generation of indoor environments. 2008.

[6] H. Ehrig et al. Introduction to graph grammars with applications to semantic networks. 1992.

[7] T. Grechkin. Towards context-sensitive reorientation for real walking in virtual reality. 2015.

[8] E. Hahn et al. Persistent realtime building interior generation. 2006.

[9] V. Interrante et al. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. 2007.

[10] D. Luebke and C. Georges. Portals and mirrors: Simple, fast evaluation of potentially visible sets. 1995.

[11] J. Martin. Algorithmic beauty of buildings: Methods for procedural building generation. 2005.

[12] P. Müller et al. Procedural modeling of buildings. 2006.

[13] C. T. Neth et al. Velocity-dependent dynamic curvature gain for redirected walking. 2012.

[14] N. Nilsson et al. 15 years of research on redirected walking in immersive virtual environments. 2018.

[15] Y. I. H. Parish and P. Müller. Procedural modeling of cities. 2001.

[16] K. Perlin. An image synthesizer. 1985.

[17] K. Perlin. Noise hardware. 2001.

[18] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. 1991.

[19] S. Razzaque. Redirected walking. 2001.

[20] D. Simons and D. Levin. Change blindness. 1997.

[21] M. Slater et al. Using presence questionnaires in reality. 2000.

[22] F. Steinicke et al. Does a gradual transition to the virtual world increase presence? 2009.

[23] G. Stiny et al. Shape grammars and the generative specification of painting and sculpture. 1971.

[24] E. A. Suma et al. Leveraging change blindness for redirection in virtual environments. 2011.

[25] E. A. Suma et al. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. 2012.

[26] S. J. Teller. Visibility computation in densely occluded polyhedral environments. 1990.

[27] M. Usoh et al. Walking > walking-in-place > flying, in virtual environments. 1999.

[28] K. Vasylevska and H. Kaufmann. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. 2013.

[29] K. Vasylevska and H. Kaufmann. Influence of path complexity on spatial overlap perception in virtual environments. 2015.

[30] K. Vasylevska and H. Kaufmann. Compressing vr: Fitting large virtual environments within limited physical space. 2017.

[31] B. Williams et al. Exploring large virtual environments with an hmd when physical space is limited. 2007.