Virtualizing Physical Space

Sebastian Lennard Marwecki

Hasso Plattner Institute, Department of
Digital Engineering, University of Potsdam

Dissertation submitted in partial fulfillment
of the requirements for the degree of
– Dr. rer. nat –
Computer Science, Human-Computer Interaction

January 2021

Advisor:
Prof. Dr. Patrick Baudisch (Hasso Plattner Institute)

Reviewers:
Prof. Dr. Harald Reiterer (University of Konstanz)
Prof. Dr. Florian 'Floyd' Mueller (Monash University, Melbourne)

Members of the committee:
Prof. Dr. Andreas Polze (Hasso Plattner Institute, head of committee)
Prof. Dr. Robert Hirschfeld (Hasso Plattner Institute)
Prof. Dr. Felix Naumann (Hasso Plattner Institute)

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-52033
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-520332

"The ideal space must contain elements of magic,
serenity, sorcery and mystery."
— Luis Barragan

"Berlin-Charlottenburg, 1 room, 38m², 1089€/month"
— Immobilienscout.com

Abstract

The true cost for virtual reality is not the hardware, but the physical space it requires, as a one-to-one mapping of physical space to virtual space allows for the most immersive way of navigating in virtual reality. Such "real-walking" requires physical space to be of the same size and the same shape of the virtual world represented. This generally prevents real-walking applications from running on any space that they were not designed for.

To reduce virtual reality's demand for physical space, creators of such applications let users navigate virtual space by means of a treadmill, altered mappings of physical to virtual space, hand-held controllers, or gesture-based techniques. While all of these solutions succeed at reducing virtual reality's demand for physical space, none of them reach the same level of immersion that real-walking provides.

Our approach is to *virtualize* physical space: instead of accessing physical space directly, we allow applications to express their need for space in an abstract way, which our software systems then map to the physical space available. We allow real-walking applications to run in spaces of different size, different shape, and in spaces containing different physical objects. We also allow users immersed in different virtual environments to share the same space.

Our systems achieve this by using a tracking volume-independent representation of real-walking experiences — a graph structure that expresses the spatial and logical relationships between virtual locations, virtual elements contained within those locations, and user interactions with those elements. When run in a specific physical space, this graph representation is used to define a custom mapping of the elements of the virtual reality application and the physical space by parsing the graph using a constraint solver. To re-use space, our system splits virtual scenes and overlap virtual geometry. The system derives this split by means of hierarchically clustering of our virtual objects as nodes of our bi-partite directed graph that represents the logical ordering of events of the experience. We let applications express their demands for physical space and use pre-emptive scheduling between applications to have them share space. We present several application examples enabled by our system. They all enable real-walking, despite being mapped to physical spaces of different size and shape, containing different physical objects or other users.

We see substantial real-world impact in our systems. Today's commercial virtual reality applications are generally designing to be navigated using less immersive solutions, as this allows them to be operated on any tracking volume. While this is a commercial necessity for the developers, it misses out on the higher immersion offered by real-walking. We let developers overcome this hurdle by allowing experiences to bring real-walking to any tracking volume, thus potentially bringing real-walking to consumers.

ZUSAMMENFASSUNG

Die eigentlichen Kosten für Virtual Reality Anwendungen entstehen nicht primär durch die erforderliche Hardware, sondern durch die Nutzung von physischem Raum, da die eins-zu-eins Abbildung von physischem auf virtuellem Raum die immersivste Art von Navigation ermöglicht. Dieses als „Real-Walking“ bezeichnete Erlebnis erfordert hinsichtlich Größe und Form eine Entsprechung von physischem Raum und virtueller Welt. Resultierend daraus können Real-Walking-Anwendungen nicht an Orten angewandt werden, für die sie nicht entwickelt wurden.

Um den Bedarf an physischem Raum zu reduzieren, lassen Entwickler von Virtual Reality-Anwendungen ihre Nutzer auf verschiedene Arten navigieren, etwa mit Hilfe eines Laufbandes, verfälschten Abbildungen von physischem zu virtuellem Raum, Handheld-Controllern oder gestenbasierten Techniken. All diese Lösungen reduzieren zwar den Bedarf an physischem Raum, erreichen jedoch nicht denselben Grad an Immersion, den Real-Walking bietet.

Unser Ansatz zielt darauf, physischen Raum zu *virtualisieren*: Anstatt auf den physischen Raum direkt zuzugreifen, lassen wir Anwendungen ihren Raumbedarf auf abstrakte Weise formulieren, den unsere Softwaresysteme anschließend auf den verfügbaren physischen

Raum abbilden. Dadurch ermöglichen wir Real-Walking-Anwendungen Räume mit unterschiedlichen Größen und Formen und Räume, die unterschiedliche physische Objekte enthalten, zu nutzen. Wir ermöglichen auch die zeitgleiche Nutzung desselben Raums durch mehrere Nutzer verschiedener Real-Walking-Anwendungen.

Unsere Systeme erreichen dieses Resultat durch eine Repräsentation von Real-Walking-Erfahrungen, die unabhängig sind vom gegebenen Trackingvolumen – eine Graphenstruktur, die die räumlichen und logischen Beziehungen zwischen virtuellen Orten, den virtuellen Elementen innerhalb dieser Orte, und Benutzerinteraktionen mit diesen Elementen, ausdrückt. Bei der Instanziierung der Anwendung in einem bestimmten physischen Raum wird diese Graphenstruktur und ein Constraint Solver verwendet, um eine individuelle Abbildung der virtuellen Elemente auf den physischen Raum zu erreichen. Zur mehrmaligen Verwendung des Raumes teilt unser System virtuelle Szenen und überlagert virtuelle Geometrie. Das System leitet diese Aufteilung anhand eines hierarchischen Clusterings unserer virtuellen Objekte ab, die als Knoten unseres bi-partiten, gerichteten Graphen die logische Reihenfolge aller Ereignisse repräsentieren. Wir verwenden präemptives Scheduling zwischen den Anwendungen für die zeitgleiche Nutzung von physischem Raum. Wir stellen mehrere Anwendungsbeispiele vor, die Real-Walking ermöglichen – in physischen Räumen mit unterschiedlicher Größe und Form, die verschiedene physische Objekte oder weitere Nutzer enthalten.

Wir sehen in unseren Systemen substantielles Potential. Heutige Virtual Reality-Anwendungen sind bisher zwar so konzipiert, dass sie auf einem beliebigen Trackingvolumen betrieben werden können, aber aus kommerzieller Notwendigkeit kein Real-Walking beinhalten. Damit entgeht Entwicklern die Gelegenheit eine höhere Immersion herzustellen. Indem wir es ermöglichen, Real-Walking auf jedes Trackingvolumen zu bringen, geben wir Entwicklern die Möglichkeit Real-Walking zu ihren Nutzern zu bringen.

DECLARATION OF AUTHENTICITY

I declare that all material presented is my own work, or collaborations in which I was always the scientific lead, or fully and specifically acknowledged wherever adapted from other sources. This dissertation has not been previously submitted, in part or whole, to any university or institution for any degree, diploma, or other qualification.

Some ideas and figures have been previously published. Specific chapters and sections that directly derived from these publications are listed in the following:

Ich erkläre, dass es sich bei allen vorgestellten Materialien um meine eigene Arbeit bzw. um Kollaborationen handelt, bei denen ich stets die wissenschaftliche Leitung innehatte, oder die vollständig und spezifisch anerkannt werden, wenn sie aus anderen Quellen adaptiert wurden. Diese Dissertation wurde bisher weder ganz noch teilweise an irgendeiner Universität oder Institution für irgendeinen Abschluss, ein Diplom oder eine andere Qualifikation eingereicht.

Einige Ideen und Bilder sind bereits veröffentlicht. Hier gelistet sind die Kapitel und Sektionen, die direkt aus diesen Veröffentlichungen abgeleitet sind:

- Chapter 3 was published and presented as: Sebastian Marwecki, Maximilian Brehm, Lukas Wagner, Lung-Pan Cheng, Florian 'Floyd' Mueller, and Patrick Baudisch. 2018. VirtualSpace - Overloading Physical Space with Multiple Virtual Reality Users. In *Proc. CHI '18*, Paper 241.

- Chapter 4 was published and presented as: Sebastian Marwecki, Andrew D. Wilson, Eyal Ofek, Mar Gonzalez Franco, and Christian Holz. 2019. Mise-Unseen: Using Eye Tracking to Hide Virtual Reality Scene Changes in Plain Sight. In *Proc. UIST '19*, pg. 777–789.

- Chapter 5 was published and presented as: Sebastian Marwecki and Patrick Baudisch. 2018. Scenograph: Fitting Real-Walking VR Experiences into Various Tracking Volumes. In *Proc. UIST '18*, pg. 511–520.

Sebastian Marwecki

Potsdam, January 15th 2021

ACKNOWLEDGEMENTS

More than work, a thesis requires the opportunity and privilege to apply it. I wish to thank all the people who gave me this opportunity, the ones who advised me professionally, and the ones who supported me personally. Here I present my quixotic attempt at expressing my gratitude to them to do their effort some justice.

I owe this thesis to numerous people at HPI. Foremost, I want to thank my supervisor Professor Patrick Baudisch. Thank you for your many lessons, lessons in research vision, in teaching, lessons in distilling and conveying structured thought, and in critical thinking. True, also lessons in resilience, but more importantly the insight that research is only truly outstanding if focused. I admire and I have learned from your dedication to computer science and research. Thank you for your continuous belief in the importance of this work.

I thank the whole human-computer interaction lab at HPI. To the ones who arrived with me, Robert Kovacs and Thijs Roumen: thank you for the many times we could support each other as colleagues. To the ones who came after, Jotaro Shigeyama, Shohei Katakura, Abdullah Muhammad: thank you for learning from one another. To the ones who came before, Lung-Pan Cheng, Oliver Schneider, Pedro Lopes, Stefanie Müller, Alexandra Ion: the same. And thank you for proving that one's

effort will eventually pay off. It was a pleasure to work with and among all of you.

The students at HPI rightfully rank amongst the best and it is humbling to work with them. Thank you Maximilian Brehm for your persistency and your ideas which shaped VirtualSpace. It shows. Thank you Hendrik Bomhardt, Georg Tennigkeit, Jannis Bolik, Lukas Wagner, Lisa Ihde, and especially Lukas Fritzsche, who I could create all the improbable prototypes with, some of which found their way into this thesis. Thank you Moritz Loos, and Klara Seitz, Jan-Tobias Matysik, Alexander Popiak, and Max Jendruk for your work on Stuff Haptics – this project rocks because of you.

The research school on *service oriented systems engineering* has provided me with the opportunity to spend the last years on this topic. They guided me with an abundance of ideas and references. I thank all its members and especially its Professors Andreas Polze, Felix Naumann and Robert Hirschfeld, as well as Sabine Wagner for all their effort and enabling me throughout. I thank the helpdesk, the staff at HPI for all the small and large things we do not notice often enough.

Thanks to all the people who helped shaping my mind the right way along the way. Thanks to Professor Harald Reiterer, thanks to Professor Florian 'Floyd' Mueller, thanks to all my old lab mates everywhere, thanks to Professor Narcís Parés at Universitat Pompeu Fabra, thanks to the folks at Microsoft Research: Andy, Eyal, Christian and Mar. Thanks for all the exemplary portrayals of what it is a researcher should do.

Working in research entails a certain probability for setbacks. I want to thank my family for their relentless support, thanks to my mum, to my brothers Daniel and Simon, and my granny, my favorite proof reader. Thanks to my Lindy Hop crew, my friends, and all people who enabled this unknowingly. And thank you Natalie, for everything.

Lastly, I also thank you, dear reader, for your interest. This work is dedicated to the spaces of yours it may occupy – in your shelves, clouds, calendars, and in your heads.

TABLE OF CONTENTS

Abstract ..... **i**

Zusammenfassung ..... **iii**

1 Introduction **13**
1.1 Virtualizing physical space, an example 17
1.2 Virtualization in the history of computing 18
1.3 Contribution 20
1.4 Structure of this dissertation 21

2 Related work **23**
2.1 Real-walking and space compression techniques 23
2.2 Passive haptics in mixed reality 25
2.3 Procedurally generated mixed reality 27

3 Virtualizing physical space for concurrent use **33**
3.1 VirtualSpace: Overloading space with multiple users 33
3.2 VirtualSpace algorithm 39
3.3 Example applications 43
3.4 Design considerations for applications 45
3.5 Implementation and hardware 47

3.6 User study 47
3.7 Conclusion on concurrent use of virtualized space 53

4 Virtualizing props for concurrent use 54
4.1 Synchronized use of physical props for haptic feedback 55
4.2 Multi-purposing of props through covert scene changes 56
4.3 Conclusion on concurrent use of virtualized props 60

5 Virtualizing the extent and shape of physical space 61
5.1 Scenograph: 1:1 experiences for any physical space 61
5.2 Our tracking volume-independence data structure 63
5.3 Scenograph algorithm 66
Step 1: Take in given physical space 67
Step 2: Pack virtual objects into given space 68
Step 3: Re-use space by splitting spatial nodes 68
Step 4: Instantiate the experience 71
5.4 Implementation and hardware 71
5.5 User study 72
5.6 Conclusion on space-independent experiences 77

6 Virtualizing sets of props within physical space 79
6.1 Stuff-Haptics: 1:1 experiences for any set of props 79
6.2 Our space- and prop-independent data structure 82
6.3 Stuff-Haptics extended algorithm 85
Step 1: Take in geometry of physical props 86
Step 2: Map virtual objects onto geometry 87
Step 3: Handle insufficient props and space 90
Step 4: Instantiate the experience 95
6.4 Implementation 96
6.5 Technical evaluation, assessing parametric designs 96
6.6 User studies 98
6.7 Conclusion on space- and prop-independence 101

7 Conclusion and Discussion 103
7.1 Summary 103
7.2 Benefits of virtualizing physical space 104
7.3 Open challenges 108
7.4 Final remarks 113

8 References 114

1 Introduction

The true cost for virtual reality is not the hardware it requires, it is physical space.

The most immersive way of navigating through virtual reality is to allow users to freely walk in the physical world. This one-to-one mapping of the user's physical motion to the motion in the virtual world is known as *real-walking* [95]. Real-walking leads to the highest levels of immersion when compared to simulated walking using treadmills [20] or to virtual locomotion techniques such as *teleportation* [11] or *walking-in-place* [93].

One would expect real-walking to become the dominant navigation technique, but this is far from true. Virtual reality arcades that employ real-walking, such as *The Void* [101] or *Dreamscape* [25], among others, have not caught on the same way consumer-grade virtual reality hardware has, such as *Oculus* [72] or *Vive* [100]. These tracking systems are capable of enabling real-walking in a room-sized tracking volume. Instead of requiring users to go to the setup, these systems allow users to bring virtual reality experiences into their preferred environments, specifically their living rooms. However, when looking at experiences available for room-scale virtual reality, surprisingly few use real-walking. Most experiences still employ simulated walking or virtual locomotion techniques, despite the reduced experience. How can this be?

The reason is that while creators tailor real-walking experiences to fit onto tracking spaces of specific size and shape, only a minority of consumers have that exact tracking space available.

For creators the typical workflow demands real-walking experiences to be designed with specific physical tracking volumes in mind. The workflow is to initially determine the size and shape of the available tracking volume, such as the creator's research lab or some standardized installation, and then design the virtual world for that volume accordingly. Because of this, virtual reality experiences tend to be specific to the size and shape of the tracking volume they were designed for.

For most consumers real-walking is being made impractical due to its high demand of physical space. For most consumers the rent for example of 4m x 4m surpasses the cost of a virtual reality headset and tracking system in a matter of a few months. If any, consumers' available physical space varies widely. Only 0.3% of Steam users can dedicate this amount of space to virtual reality use [87]. The problem is further aggravated as systems such as Steam require dedicated tracking spaces, which are rectangularly shaped and also free of any physical objects, while users' physical environments tend to be obstructed with physical objects, resulting in various shapes of free space.

As a consequence creators do not design for real-walking. Creators of virtual reality experiences at home are faced with a choice: (1) to artificially narrow down their market by designing for niche tracking volumes or (2) abandon real-walking – creators have picked the latter. Instead they let users navigate by teleporting around, sacrificing the added immersion real-walking offers, with games that employ real-walking being the exception [94] or still in development [92]. We find ourselves in a situation where users have paid for a virtual reality system capable of real-walking but have essentially no real-walking contents even if they have space available. Creators cannot afford to

create multiple instances of their content for various tracking volumes. The current approach of designing experiences that are tailored to users' individual tracking volumes becomes impossible and creators design for small spaces to reach a broader audience.

Researchers have proposed several techniques to reduce the space demands of real-walking. These techniques, however, still mandate specific tracking volumes. *Redirected walking* [74] folds long walking paths into a finite tracking volume and has sparked a field of research, but still requires very large installations to take effect (4m x 10m). *Impossible spaces* [89] folds virtual geometry and also considerably reduces space requirements. Neither approach ultimately addresses the problem that applications still assume a fixed tracking volume.

The most promising approach is to automatically reshape virtual environments to fit arbitrary room shapes. A substantial step in this direction is *Oasis* [85], which enables customization of virtual worlds to whatever physical space the user has available. Users scan their tracking volume with a depth camera, and from this data, Oasis creates a static virtual location that fits into the space. This makes it easier to fit real-walking content into arbitrary room shapes.

The reason that also this technique has not been adopted by creators lies in the workflow described earlier, in which creators co-design virtual experiences and the physical space they occupy. Simply put, creators still use space to express any virtual experience. This often entails a narration, or story. Procedural content for real-walking virtual reality as of now has little means of describing such experiences independent of space or offering fallbacks in cases where content for narration does not fit into the physical space available. While research has shown how to adapt virtual *environments*, it has yet to show how to create any adaptable *experience*.

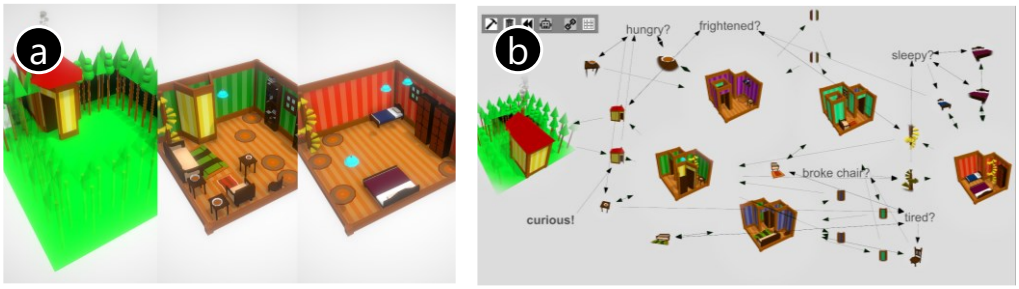

Figure 1: (a) Traditionally, designers of real-walking experiences have specific tracking volumes in mind. This rendition of the fairy tale 'Goldilocks and the three bears' consists of three 25m^2 locations filled with interactive assets ('bowls of porridge', 'bears' chairs', 'bears' beds'). Users change locations using corridors as portals [87]. Unfortunately, specifying the tracking volume prevents the experience from running on smaller tracking volumes. (b) Our software tool uses a tracking volume-independent representation of real-walking experiences to instantiate experiences for tracking volumes of different size and shape. Here we use our system to map 'Goldilocks' to an L-shaped 8m^2 space. While maintaining the narrative structure, it splits the three locations into six smaller ones, each fitting the new tracking volume.

This thesis addresses the question in how far real-walking experiences can be created independently of the physical space they occupy. We cannot change the invariant in our problem, the various sizes and shapes of tracking volumes available to consumers. However, we can address the second part of the problem, the creators' workflow of co-designing virtual experiences and the space they occupy. Creators should still be able to design any narration they want to convey. However, these experiences should neither consume exorbitant amounts of space, nor should they require specific shapes of space, or the absence or presence of specific physical objects in that space.

Our approach will be to demonstrate that space can be *virtualized*. Real-walking experiences that run on such virtualized physical space are independent of the size and shape of the physical space the user has available and any physical objects that the user contains within that space, and they allow for any expressive storyline that their creators envision. By making real-walking experiences independent of any particular tracking volume, virtualized physical space provides the missing component for making real-walking experiences available to consumers.

1.1 VIRTUALIZING PHYSICAL SPACE, AN EXAMPLE

Figure 1 shows an example of a real-walking experience that runs on virtualized physical space, 'Goldilocks and the three bears'. This experience is enabled by one of our software systems. We achieve two of our primary goals, firstly providing independence from the amount and shape of physical space, and secondly allowing for expressive storylines. However, it still assumes that the physical space is devoid of physical objects, or props for short.

Figure 2 shows an extension of this software system. We achieve our third goal and enhance the example of 'Goldilocks' to incorporate whatever props users have available within their tracking volumes.

Figure 2: (a) The user invokes our virtual reality experience, 'Goldilocks and the three bears', in their regular living environment, which is full of physical objects, like tables, chairs and countertops filled with appliances. (b) Our software tool translates this physical environment into our virtual 'Goldilocks' experience and provides real-walking by shaping its virtual environment to the physical space available, it provides passive haptics by automatically lining up relevant virtual objects with physical objects of matching haptic qualities, such as the 'bears' three bowls of porridge' with the physical water kettle, and 'papa bear's chair' with the physical table. (c) In another room, our software tool redesigns the virtual experience on the fly. (d) It now places the 'bowls of porridge' onto the physical window sill above the physical heater and the chair onto the physical living room table.

1.2 VIRTUALIZATION IN THE HISTORY OF COMPUTING

Our approach of making real-walking experiences independent from available physical space is inspired by this evolution of computing itself.

In the 1950s and 60s, the invention of operating systems made application programs independent of the physical hardware. Instead of requiring application programs to run on 'bare-metal' and accessing systems resources directly, operating systems allow applications to run on arbitrary computers and architectures by providing an abstraction of the underlying physical hardware, so that applications access their system's resources through an application programming interface.

For example, instead of accessing the systems physical memory directly, applications refer to addresses of virtual memory, which the operating system then maps to the system's actual physical memory using a memory management unit. This simple mechanism of *managing space* has allowed computer systems to deal more gracefully with limitations of physical memory space. For example, it allowed computer systems to load programs larger than the actual physical memory, or to load multiple programs into the same memory at once.

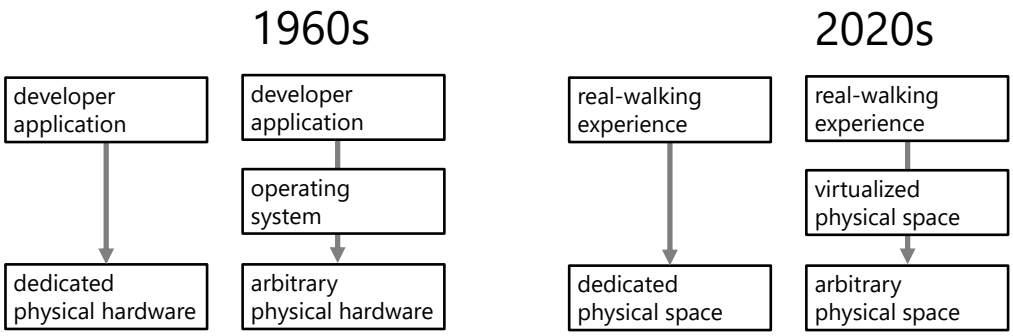

Figure 3: The invention of operating systems made application programs independent of the underlying physical hardware. This thesis makes real-walking applications independent from physical space.

We very much still live in the 1950s with regard to how today's virtual reality systems deal with space. These systems still access their primary resource, physical space, directly and with all the resulting limitations;

physical space can be used by a single user only, if an experience requires more space than available it simply fails, and so on. Every real-walking application today is basically an embedded system. The problem that application developers have faced before the onset of operating system translates directly to the problem that creators of real-walking applications face today. It now is the impracticability of developing for various amounts and shapes of physical space, and various sets of physical props. Real-walking applications still directly access their physical resources without any layer of abstraction.

This thesis takes paradigms of operating system paradigms that drove the evolution in computing and applies them to real-walking applications. We will introduce abstractions between real-walking experiences and physical space, the main resource for virtual reality. In our systems, real-walking experiences do not access space directly anymore, but instead access space through a range of application programming interfaces. This allows our software systems, which detail parts of a larger 'space operating system', to manage physical space.

Virtualization eased, but also changed software development. Consequently, developing for virtual instead of physical space is different. Creators of real-walking applications today use space to express their desired user experience, similar to programmers having used assembler or machine code. With our approach, creators will not program on 'bare-metal' anymore, instead they express the experience on a higher level that is independent of physical space, similar to the development of languages such as C or Fortran. By first principle analysis creators primarily want to convey a story to the users, a concatenation of events that entail user choices and stage-setting. Arguably space has been the most intuitive tool for this so far. However, a story can be expressed using different tools than space. We define a tracking volume-independent representation of real-walking experiences, which creators produce through an interface. Through this representation, real-walking experiences become modular and *re-useable*.

Our systems work with these independent representations of real-walking experiences to manage physical space. One of our systems is akin to a space memory management unit. Instead of managing independent processes on the same memory, it manages users immersed in different virtual environments and their access to the same physical space. Schedulers, namely fixed priority pre-emptive scheduling, manage these multiple applications over time. Our systems deal with differently sized and shaped physical space, as found in space fragmentation. We take in the idea of device drivers so that virtual objects can map to arbitrary physical props. Our systems compile real-walking experiences by arranging virtual objects for specific spaces, they link groups of compiled and assembled objects together and they load them onto physical space.

Altogether, virtual reality systems that inherit the benefits of operating systems can deal more gracefully with limitations of physical space. For example, experiences access virtual space to have more space than physically present, multiple experiences can share the same physical space, and most importantly, experiences can run on different sets of physical hardware.

1.3 Contribution

Our main contribution is providing an abstraction layer between real-walking experiences and the physical space they use, by drawing from the field of operating systems. At the core of this abstraction is a tracking volume-independent representation of virtual reality experiences that offer real-walking or passive haptics, as expressed by creators. This representation can be interpreted through our software systems to be instantiated in any tracking volume, specifically tracking volumes that can be shared with other people (VirtualSpace), that can have any size and shape (Scenograph), that contain limited sets of physical props (Mise Unseen) or arbitrary sets of physical props or room geometry (Stuff-Haptics).

Making real-walking experiences independent of the user's physical space will have substantial commercial impact for virtual reality systems, as the problem of requiring specific physical spaces has all but stopped the proliferation of real-walking due to developers' reluctance to require users to have access to that space. Future virtual reality applications will run on a wide range of installations and will soar past current limitations by accessing additional virtual space, sharing resources, and building on re-useable high level descriptions of real-walking content.

1.4 Structure of this dissertation

This dissertation is structured as follows:

In chapter 2 we introduce relevant research on related work, specifically the subjects of real-walking, passive haptics, and procedural content in mixed reality.

In chapter 3 we describe how to apply virtualization to achieve concurrent use of physical space. We present overloading, a technique that allows multiple users immersed in different real-walking experiences to share the same physical space at the same time. Our system divides the overall tracking space into computationally determined individual tiles and one to each application. Frequent *maneuvers* then allow applications to incentivize users to walk across the entire physical space, thereby progressing each application's narrative and to prevent users from noticing that they are confined to a tile. This strategy enables our system to achieve packings of high density, such as four users in $16m^2$.

In chapter 4 we describe how to achieve concurrent use not only with physical space, but with physical props as well. We show different techniques to efficiently use physical props into virtual reality experiences. The first technique allows multiple users immersed in different virtual worlds synchronously to use the same physical prop. The second technique maps multiple potential virtual objects to single

physical prop at runtime by utilizing the concept of inattention blindness together with eye-tracking data.

In chapter 5 we present how virtualization makes real-walking experiences independent of the extent and shape of physical space. We represent real-walking experiences with a complex storyline in a way that is agnostic to any tracking volume. This representation is not spatial until instantiated into tracking volumes that can have any size and shape. This allows creators of real-walking experiences to define storylines independent of space, and it allows users to instantiate the experience to whatever tracking volume they have available.

In chapter 6 we present how to apply virtualization to achieve independence not only with regard to space, but also to the physical props contained within, so that more spaces can be used by the same real-walking experience. We build on our existing systems so that storylines for real-walking applications can remain complex and they can still run in tracking volumes of arbitrary size and shape, while additionally they can now contain any set of physical props. These can be limited and uncurated set of props, specifically stuff already found in the home, such as hairdryers, half-used candles, etc. It allows for different sets of props as well as switching props at runtime. It specifically also allows creators to choose between different automated solutions for handling limited prop sets; re-using props or altering the storyline. Our system achieves this by combining a story's structure and its parametric models into a space- and prop-independent representation of a real-walking experience.

In chapter 7 we conclude this thesis by discussing the impact of virtualizing physical space on a broader level, sketching out future uses, and outlining key directions for future research.

2 Related work

This thesis builds upon work on multiple fields: real-walking and space compression techniques, passive haptics and opportunistic use of props, and procedurally generated content for mixed reality including its layouting algorithms and data structures for representing story narratives.

2.1 Real-walking and space compression techniques

While walking in virtual reality can be enabled by omni-directional treadmills [20, 82], it is largely simulated with virtual locomotion techniques such as *walking-in-place* [93] or *teleportation* [11]. However, researchers agree that real-walking, a continuous one-to-one mapping of physical to virtual locomotion, leads to the highest user satisfaction [95]. Real-walking has high space requirements, which researchers have tried to reduce with several techniques, such as *resetting* [103], which rotates or repositions users once they hit the tracking volume's borders. However, as these techniques perceptibly interrupt or alter the one-to-one mapping of physical to virtual movement, the immersive quality of real-walking is reduced. This is explained by the fact that users inherently prefer to rely on known real-world concepts when learning and applying any interaction technique, see *Blended Interaction* [47]. This very much includes walking for virtual locomotion.

The following approaches reduce space requirement for real-walking, either by altering the one-to-one mapping, by altering the virtual environment, or by sharing physical space.

Altering the mapping of physical to virtual motion

There exists a multitude of virtual locomotion techniques [56], all of which in one way or another alter the motion mapping.

For example, *Seven league boots* [45] virtually scale the user's movements to cover a larger virtual area while walking.

Razzaque et al. [74] found that *redirected walking*, which folds long walking paths into a finite tracking space, can lower the amount of space required for walking in virtual reality (Figure 3). This breaks the one-to-one mapping only unperceivably. Redirected walking benefits from the a priori knowledge of the virtual environment, onto which the walking paths can be mapped, as a vast body of research shows [12, 22, 41, 90]. Even with the constraint of a priori setting of walking paths, this approach is sub-perceptible only for large tracking volumes (e.g., 4m x 10m [74]). For smaller spaces, complementary fallback techniques such as *resetting* [103] are needed, which disrupt the walking experience.

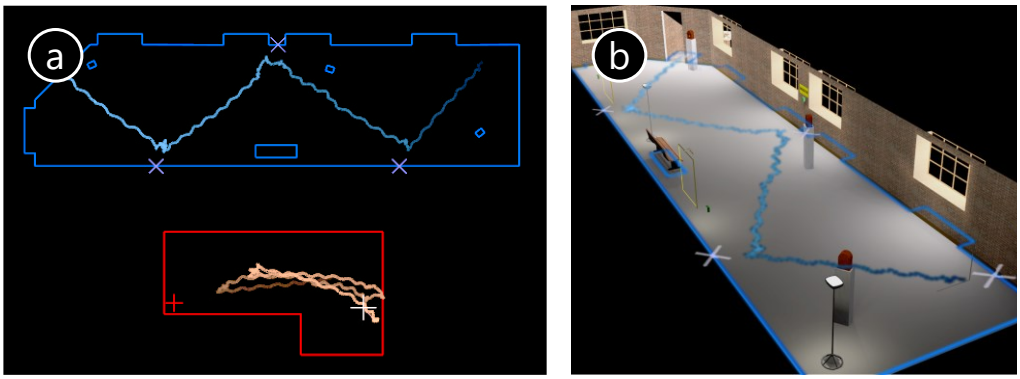

Figure 3: Techniques like *redirected walking* [72] compress space and reduce space demand. (a) This technique folds virtual walking paths (blue) into finite tracking volumes (red), so that (b) virtual environments can appear larger than the tracking volume the user has available.

Altering the virtual environment

In *impossible spaces*, Suma et al. [89] lets virtual rooms unnoticeably overlap to compress space (Figure 4). As *impossible spaces* uses change blindness [88], it requires large amounts of space to be sub-perceptible (9m x 9m [88]). However, even when the overlap is noticeable this technique still enables enjoyable experiences, like the commercially available game *Unseen Diplomacy*, a game with relative success [94] that uses only a fairly small space (4m x 3m). *Impossible spaces* can be combined with *redirected walking* [53].

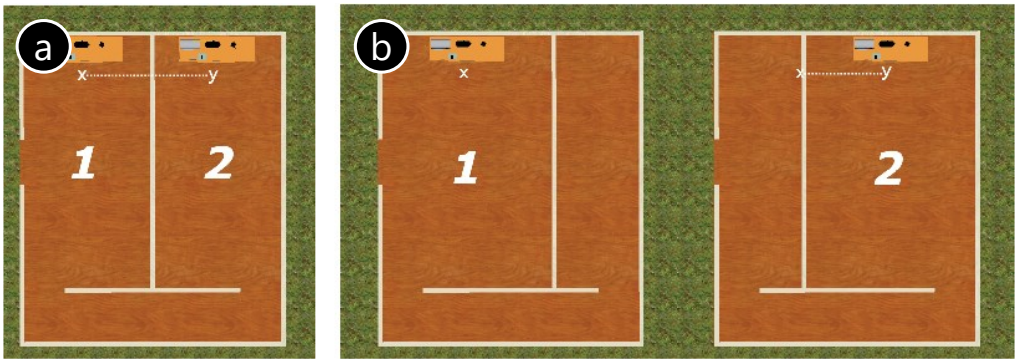

Figure 4: *Impossible spaces* [89] compresses (a) this space by (b) letting room 1 and 2 overlap. The different rooms are loaded into the environment at runtime, while the user is in the corridor below.

Sharing physical spaces

Real-walking naturally extends to multiple users. Sra et al. [86] developed virtual reality applications that allow sharing the same physical space between multiple users in the same virtual environment, albeit at a virtual distance. Redirected walking has been shown to also apply to multiple users [5, 43], but still has a relatively high risk of collisions.

2.2 Passive haptics in mixed reality

Passive haptics [40] increases immersion in virtual reality by making all virtual objects in a virtual scene tangible through co-located physical props. Hinckley et al. pioneered the concept by using generally passive

and low-fi props. Passive haptics can enhance the sense of presence (Figure 5). In a study by Hoffman [42], participants in virtual environment could guess an object's properties, such as the weight of a teapot more accurately if it had been given a physical representation. In a study by Insko [44], participants immersed in a virtual environment crossed a virtual pit by balancing on a ledge. Behavioral presence, heart rate, and skin conductivity were affected more if the ledge was created using a physical wooden plank.

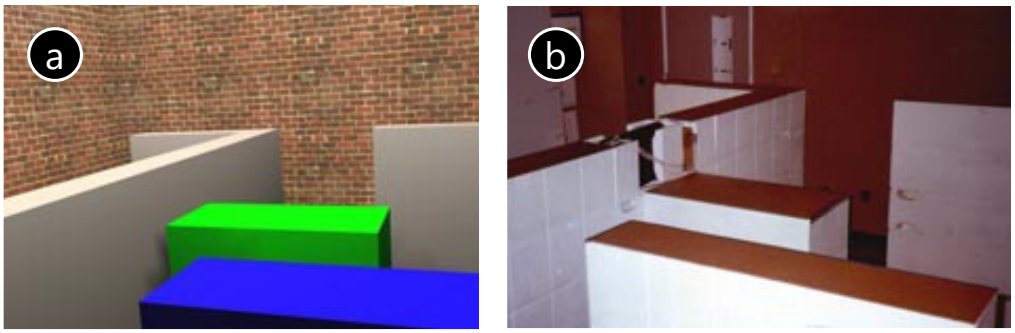

Figure 5: *Passive haptics* [40] matches (a) a virtual environment with (b) matching physical counterparts (physical props).

Passive haptics is used in augmented reality as well. Low et al. project different virtual worlds into the same physical room using Styrofoam walls [58]. VHMR allows to both see and touch virtual objects [96].

Passive haptics requires physical objects that match virtual geometries to align prior to interaction. This limits their use to pre-constructed places such as labs or virtual reality arcades, such as *The Void* [101] and others [25, [106]]. Passive haptics experiences have not caught on for home use, as such experiences will not be run often enough to justify the expense, time and storage effort for new passive haptics props for every new experience.

Re-configurable and re-usable props

Props can be re-configurable or assembled from scratch. *VirtualBricks* [4] and *TanGi* [28] are sets of small props that form versatile larger props. This configuration takes time and does not scale well. *TurkDeck* [15]

allows for large-scale passive haptics, but requires human actuators as a trade-off.

Props can be re-used. *Haptic Retargeting* [6] matches multiple small objects onto only one prop by altering the mapping of physical to virtual body motion. *AutoTurk* [13] uses the *impossible spaces* approach [87] to re-use props.

Opportunistic use of props

Instead of requiring specific physical props, researches have opportunistically paired virtual objects with arbitrary objects or even household items. *Opportunistic controls* [37], for example, connect the displayed environment to physical surfaces that are re-purposed to provide haptic responses to specific widget-like digital controls, such as sliders and buttons. Using a physical tabletop as an interface, *ZuneBuggy* [67] was an early example of how physical properties can be integrated into virtuality.

Combining opportunistic use with passive haptics, *substitutional reality* [79] investigated the limits of mismatches between physical and virtual objects (Figure 6a+b). One way to tackle the resulting mismatch of virtual-to-physical object pairing is by redirecting users' movement with *retargeting touches* [107], quite similar to redirected walking, a concept that has been re-used [17, 49]. Another way to tackle the mismatch is to adapt the virtual environment. *Annexing Reality* [39] defines its virtual content parametrically, analyzes the environment for small suitable props, and opportunistically assigns objects as passive proxies by evaluating its shape.

2.3 Procedurally generated mixed reality

An alternative solution to reducing space demand by breaking the mapping of physical to virtual motion, is to alter or altogether generate the virtual world. Procedural generation is used in games [38]. Mourato et al. [66], for example, created variations of 2D games (platformers)

based on the analysis of original game segments. Procedural content can be used in mixed reality as well. For example, Vasylevska et al.'s *flexible spaces* [97, 98] automates *impossible spaces* [89].

An important step in that vein has been taken in *Oasis* [85]. The system builds a full model of a physical environment to then procedurally generate a virtual environment the user can walk through (Figure 6c+d). This concept has been reused, for example in *Reality Skins* [78] and others [80], which carries a virtual scene (underground tunnels, spaceships) across large physical environments preserving the multiple constraints of both physical and virtual objects. *DreamWalker* [105] applies the idea of procedural worlds to outdoor environments, *VRoamer* [16] to rogue-like games spanning over large indoor environments. This widens the possibilities for mixed reality setups.

Scanning physical environments

Procedural generated content requires a scan of the user's physical surroundings. Scanning physical environments is a well-researched area. *Oasis* [85] uses a *Project Tango* to obtain a point cloud. Many other devices can provide similar functionality, for example household robots [23]. Scanning can be improved with 3D semantic parsing [3] to derive individual objects (as point clouds or meshes) or by using shape retrieval [54] to derive the correct models and, possibly, also material properties. Another option is to obtain this information through smart objects [26].

Spatial layouting for procedural mixed reality

Procedural content requires automating spatial layouting. This is achieved using a constraint solver and parametric descriptions of virtual content.

Manual methods exist, such as in *ARchitect* [59], which requires its users to (once) layout virtual content within a physical setting. Related

work also suggests a half-automated generative design process (e.g., *DreamRooms* [102], or *Project Mars* [61]), blending skillsets of human designers and computers. These automations use constraint-based algorithms [27, 62]. *Annexing Reality* [39] uses a Hungarian algorithm for the best global mapping. For larger immobile props the user touches, *Flare* [33] analyzes a given captured 3D geometry for generating levels for racing games using a random walk algorithm. More broadly, there also exists automated spatial layouting of avatars for cinematography [60], an idea that found its way to consumer applications [29].

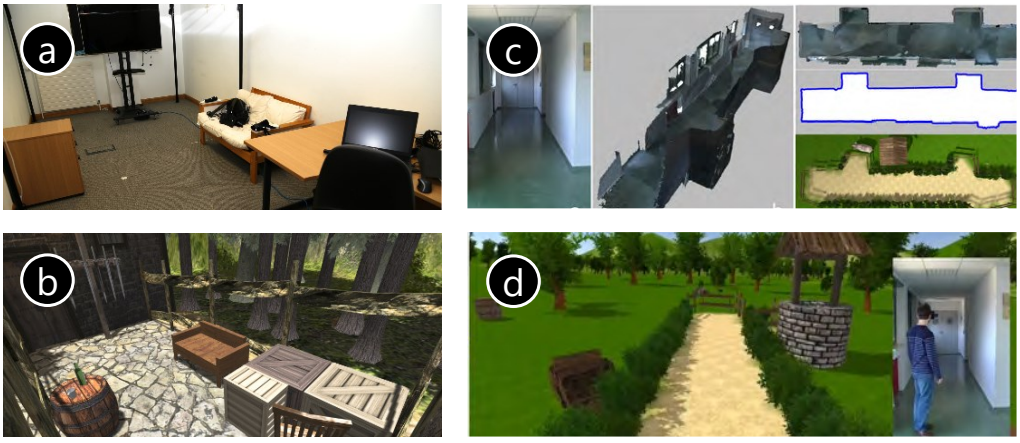

Figure 6: (a) *Substitutional reality* [79] transforms this physical living room into (b) this virtual environment, opportunistically using objects for passive haptics. (c) *Oasis* [85] automates this approach, by scanning this physical corridor to then generate (d) this virtual environment. This approach does not yet allow for complex storytelling within those generated environments.

Story representations in procedural mixed reality

Such generated virtual 'siblings' have also not yet caught on for home use. One reason is that they do not maintain a storyline across physical environments. Any story arguably mandates certain virtual objects to be present. If one of these story elements fails to map onto a limited prop set, then this story simply cannot be conveyed to the user at this location.

Spatial layouting is part of any story and any storytelling medium. In movies it is referred to as 'staging' or 'blocking', in games as 'environmental storytelling' [81].

While spatial layouting can be automated during procedural generation of content for these media, a challenge remaining is the "quantification of [user] experience," according to Yannakakis et al. [104]. Such "experience-driven" procedural content has not yet been extended to real-walking.

Next to spatial layouting creators of real-walking content want to convey an 'experience' to their users, a 'narration' or a 'story'. We here define these expressions as a structure describing all possible orderings of events and user interactions that can unfold in a virtual experience.

Research has already found data structures for stories. Graph-based story representations are used in a variety of contexts, for example in location-based narratives [63], where geolocations are matched onto a given story graph to convey an experience to the user, or interactive narratives, such as in *Storyspace* [10]. However, in current content editors for commercially available products today, such as *Source 2* for the virtual reality game *Half Life Alyx* [83] or *REDkit* for *The Witcher 3* [75], storytelling is still very much intertwined with spatial layouting, see Figure 7. As described before in chapter 1, experiences are designed with a fixed tracking volume in mind.

Figure 7: In content editors, such as *Source 2* [83], creators design experiences within a space that is fixed, by applying scripts onto objects and characters in a manually designed virtual environment. The workflow is (a) define the required space (red line), (b) place virtual objects onto that space, and (c) define a succession of events involving those objects.

This interleaving of storytelling and spatial layouting of present-day content editors is a problem for real-walking applications which play in user spaces that vary. A content editor for real-walking has two constraints instead of one. First, we need to maintain a given story structure, for which the solution normally are content editors that combine manual spatial layouting and experience design. Second, we need to account for limitations in tracking volume, for which the solution is procedural content generation for automating the spatial layouting. Procedural content solves the first constraint of limitations in tracking volume by quickly generating variance in the virtual scenery, but it is usually not used for story focused experiences.

A solution to this problem solves both of these constraints.

In a solution to this problem a user should be able to experience a story in any way or ordering possible, while being in different physical environments. Approaches such as *Oasis* solve spatial layouting through procedural generation, but without enabling such elaborate storytelling. If a story mandates a certain virtual object to be present, but it fails to map onto the limited prop set, then the story simply cannot be conveyed to the user at this location.

In a solution to this problem a creator should be able to design any story, while still using procedural generation for spatial layouting. A solution would be scalable, meaning it would work for any experience-driven application with real-walking. It is possible to create a single story that carries over different physical environments, such as in *Reality Skins* [69]; however, they are not scalable in that sense.

Maintaining user interactions has been solved in other areas. *Supple*++ [32] is a tool for automatic UI generation. It maintains ordering of all possible interaction sequences, here called "traces", similar to our objective of maintaining a story. The tools' objective is to incorporate 2D space constraints while preserving functionality. This directly translates to our constraints, tracking volume limitations and preservation of the

narrative and virtual experience. Similar to our virtual object placement, it is "impossible to compute the cost of a layout without knowing all the interface elements comprising that layout." This then requires "modifications to the optimization algorithm." As in Supple and similar GUI generating systems [8, 71] we use a high-level structure to maintain these "traces" [32] and let the automated spatial layouting be influenced by this structure.

3 VIRTUALIZING PHYSICAL SPACE FOR CONCURRENT USE

In this chapter we apply virtualization of physical space to require less space per user. We present *overloading*, a technique that allows concurrent use of space, so that multiple users immersed in *different* real-walking experiences can share the *same* physical space at the *same* time without being aware of each other. We describe and evaluate *VirtualSpace*, our software system implementing this technique.

3.1 VIRTUALSPACE: OVERLOADING SPACE WITH MULTIPLE USERS

VirtualSpace is designed to give each user the illusion of being in possession of the entire physical space. As illustrated in Figure 8, VirtualSpace accomplishes this by containing each user in a subset of the physical space at all times, which we call *tiles*; app-invoked *maneuvers* then shuffle tiles and users across the entire physical space. This allows apps to move their users to where their narrative requires them to be while hiding from users that they are confined to a tile. We show how this enables VirtualSpace to pack four users into 16m^2. In our study, presented in chapter 4, we found that VirtualSpace allowed participants to use more space and to feel less confined than in a control condition with static, pre-allocated space.

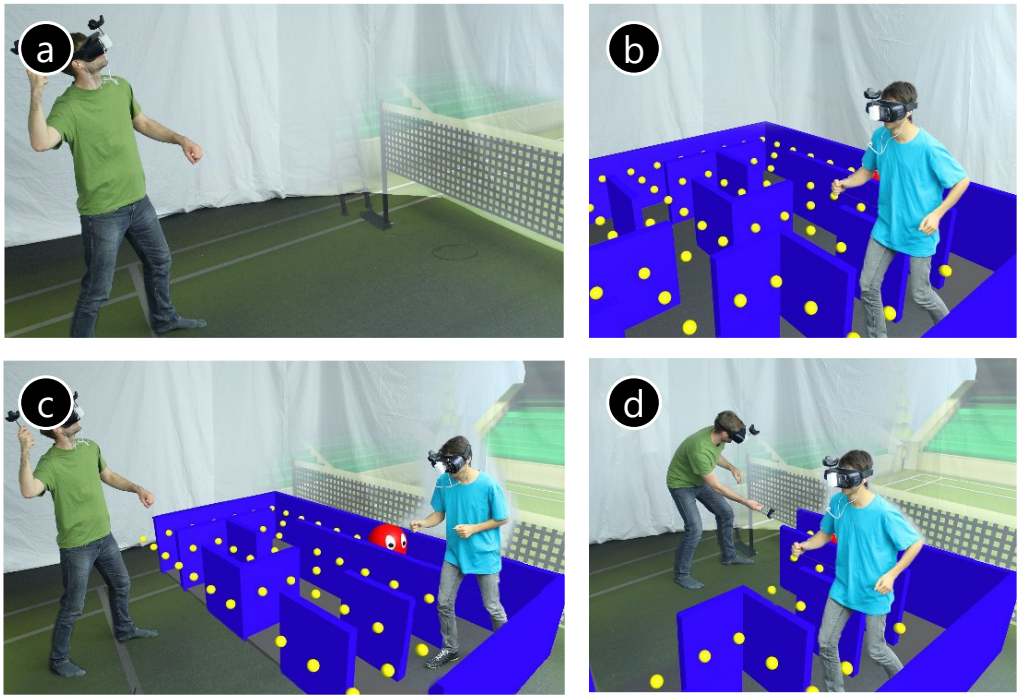

Figure 8: (a) This user is playing a badminton app. His side of the court fills the entire 4x4m tracking volume. (b) This other user is playing a Pac-Man game mapped to *the same* tracking volume. (c) VirtualSpace allows both users to share the same tracking space, without being aware of the other user. To keep users from running into each other, VirtualSpace limits each app to non-overlapping tiles at any given time. Client apps handle this in a way transparent to their users. The badminton app, for example, always makes the user's virtual opponent return the ball to locations inside the tile currently assigned to the app. (d) By reassigning tiles frequently, VirtualSpace moves users across the entire space, thereby seemingly allowing for unrestricted walking.

In the example in Figure 8, two users share the same 4x4m physical space, while being tracked using a virtual reality tracking system (Vive [100]). Both users are in their own, separate virtual environments. (a) The green user is immersed in a badminton app, while (b) the blue user experiences a Pac-Man game.

The key point is that *both* apps are mapped to the *entire* physical space, meaning that VirtualSpace allows each app to be designed under the assumption that the user has physical access to the entire 4x4m space. And that is true, albeit not necessarily at every particular moment, as VirtualSpace limits each app to a different non-overlapping tile of space. Client apps handle this in a way transparent to their user. The

badminton app, for example, makes the user's virtual opponent return the ball always to locations inside the tile currently assigned to this app.

VIRTUALSPACE MANAGES PHYSICAL SPACE USING MANEUVERS

To allow users to still complete their narrative and to prevent users from noticing that the system is confining them, VirtualSpace employs what we call *maneuvers*.

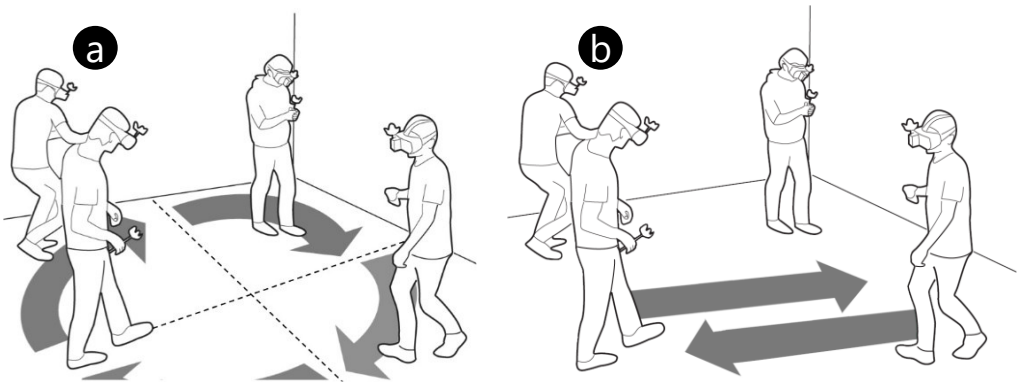

Figure 9: Users are confined into *tiles* that VirtualSpace changes using *maneuvers*. (a) The *rotation* maneuver allows apps to move their user to the adjacent tile. This way users do not feel confined. (b) Similarly, the *switch maneuver* allows two apps to switch tiles.

Figure 9a shows an example, here with four users in the same 4x4m tracking space, which is the configuration we used in our user study. When the user plays Pac-Man, the app needs to progress towards the area with the remaining pellets, but the segmentation into tiles prevents this progression. The player's app thus requests access to the desired tile, here the tile that is adjacent in clockwise order. As shown in Figure 9, VirtualSpace can provide the Pac-Man app with this access by rotating the entire field of all four users in clockwise direction. We call this the *(clockwise) rotation maneuver*.

Every maneuver comes with a certain start-up delay that allows all apps to get their users ready and every maneuver takes place at a certain movement speed. In this example, the apps may agree on a three-second delay and a one second transition speed.

As shown in Figure 10a, the Pac-Man app prepares for the maneuver by offering a virtual reward (the red cherry), yet it prevents the user from getting there by blocking the path using a ghost. Meanwhile the badminton app plays a slow ball to get the player synchronized with the timing of the upcoming maneuver.

Now the maneuver starts and every app moves their user to the new target position, here the next tile clockwise. The badminton app serves the user a stop ball that brings the player forward towards the net. As shown in Figure 10b, the Pac-Man app sends a ghost that chases the player down the corridor.

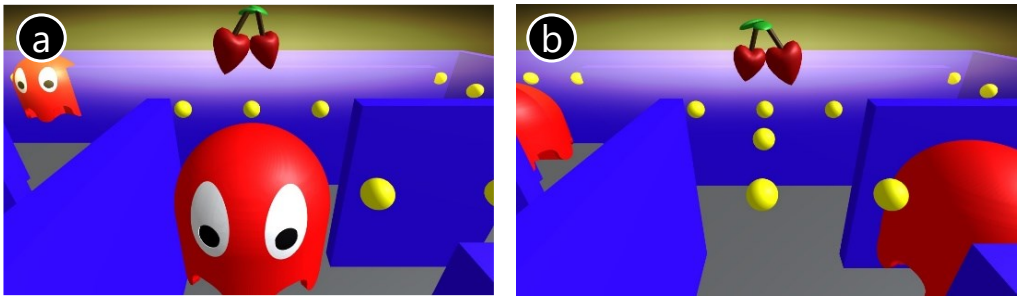

Figure 10: (a) The Pac-Man app guides users to follow the clockwise rotation maneuver by offering a virtual reward, here a cherry. (b) It prevents users' movement by placing ghosts in their way.

VirtualSpace allows for changing the rotation order and for rotations involving a subset of users. For two users we call this a *switch* maneuver. This enables applications to move their users freely around the tracking space while others can stay at specific preferred areas.

Focus maneuvers allow for fast movements

Rotation maneuvers are sufficient in that it never takes more than two revolutions to send a user to any tile. Rotations take time though, which is not always compatible with game action sequences that require fast, large, and erratic movements of a user. To enable such movement sequences, VirtualSpace offers the *focus* maneuver. This maneuver allows a single app to temporarily take over most of the tracking space. Figure 11 shows an example when the badminton app uses a Focus

maneuver to allow its user to hit the birdie in the center of the court in a quick succession of ball exchanges.

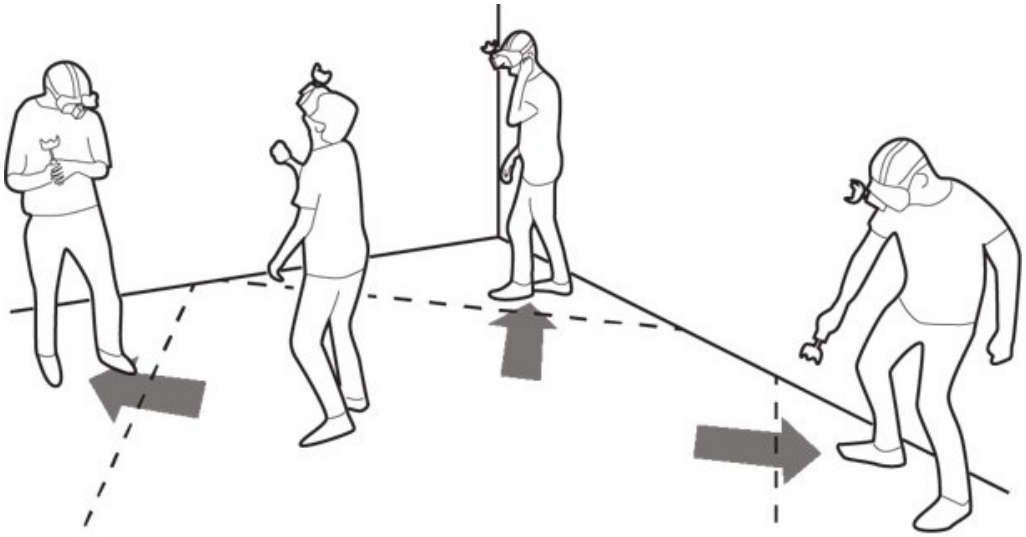

Figure 11: The *focus maneuver* temporarily provides the badminton app with control over most of the physical space. The other apps go into a *defocus state*.

While the one app has the focus, all other apps contain their users in a small amount of space at the rim of the tracking volume by providing them with a stationary task, e.g., by trapping the user in a corner (Pac-Man), or multiple moles appearing in the same bushes (Whac-A-Mole).

Focus maneuvers obviously take place at the expense of all other apps, so that focus maneuvers are only possible when all other apps agree to it. We added a possibility for economic models to attach to this process, so that apps can attach an added value (credits, money) to their maneuver request.

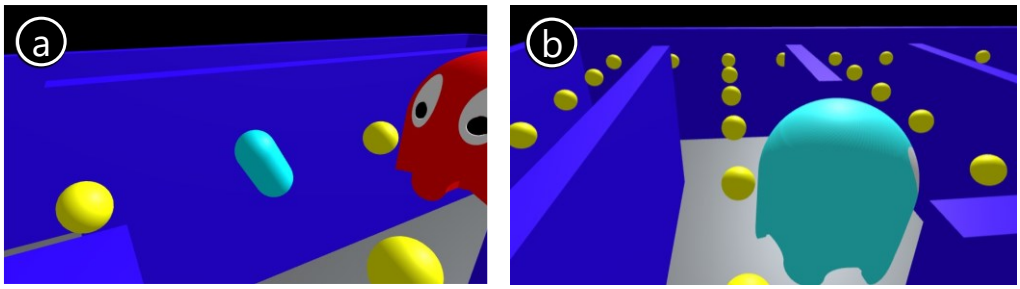

Figure 12: VirtualSpace allows for *in-app purchases*. (a) Here the Pac-Man user can collect a blue pill which lets VirtualSpace value that app higher. That allows for a focus maneuver, which can be used to (b) chase the ghosts.

VirtualSpace allows users to join or leave in real-time

VirtualSpace is designed to run continuously with users joining or leaving at any time, any reasonable number of users and any combination of apps. Additional users can join as long as there is enough free space and leave at any time, thereby freeing up space. VirtualSpace continues to offer the same maneuvers irrespective of the number of users, while the tile size remains the same. If free space is available, apps can move on to empty areas without triggering a multi-user maneuver. As shown in Figure 13b, an app can request a maneuver, e.g. a rotation, without having another user move.

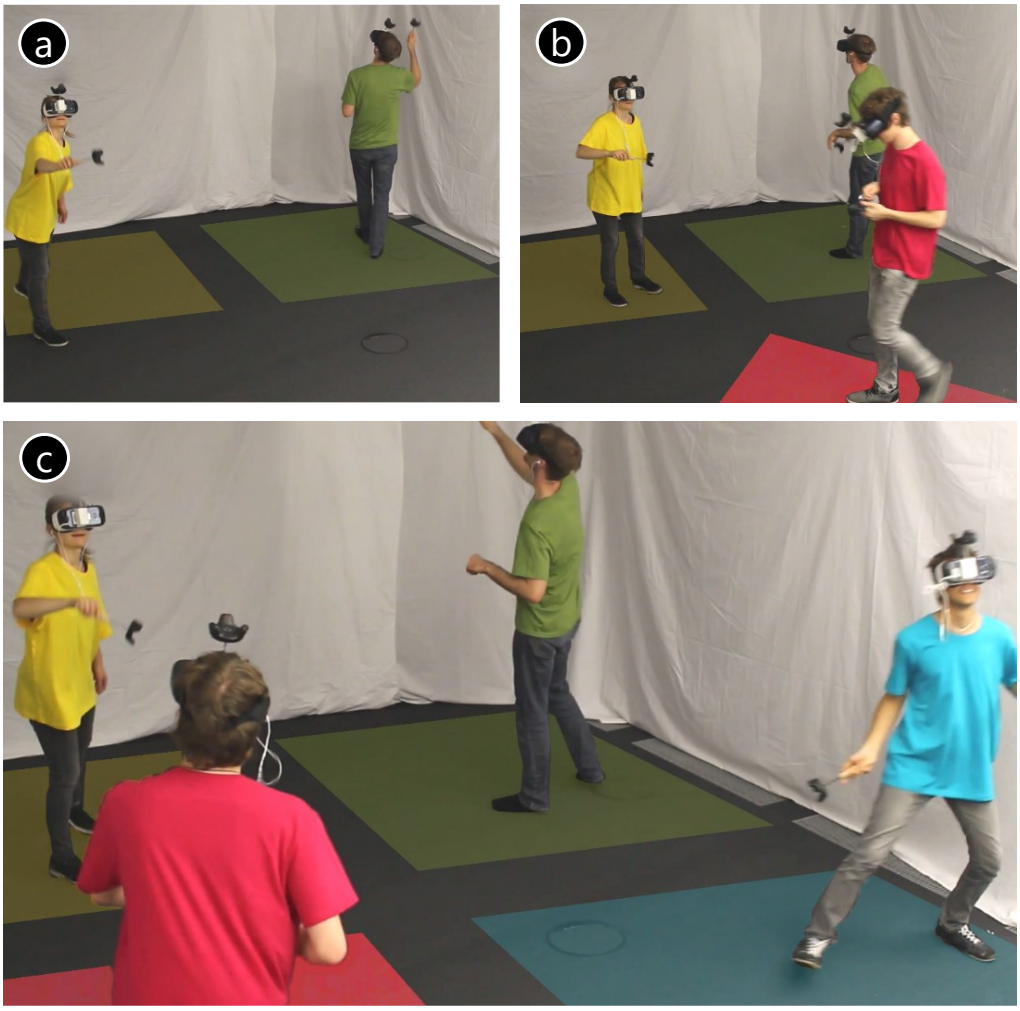

Figure 13: Internal assignment of tiles, displayed as an overlay of real-time data. (a) VirtualSpace handles configurations of fewer than four users. (b) If free space is available, apps can perform different maneuvers at the same time, until (c) all space is allocated, then maneuvers need syncing.

Strategic versus tactical approach

One of the main insights we generated during development is that we found a top-down *strategic* approach of space management to work much better than the bottom-up *tactical* approach. In the tactical approach collisions are detected by extrapolating movement and then notifying apps to avoid collisions. The strategic approach of managing movement, while somewhat constricting freedom of real-walking applications, circumvents collisions by design and allows for much tighter packing densities. Simulations of bottom-up approaches that require high precision analytical planning with virtual humans (such as *ORCA* [7]) perform significantly worse. By relying on the described tile-based approach we further avoid deadlocks, where multiple users are crunched in one corner.

Fallback visualizations

The assumption is that the apps succeed at confining their users to their tiles and at guiding their users to the agreed-upon target tiles. As with any real-walking system this may fail, e.g., when users simply ignore the system and walk into other users' tiles. VirtualSpace handles this situation like most real-walking systems (such as Vive [100], or [72]) by (1) rendering a cage around the currently available walking area, before the user might exit and if the user does, (2) draws the outline of the users, so they can see each other and stop.

3.2 VirtualSpace algorithm

VirtualSpace runs with any app implementing its API. Apps can work in any combination, with multiples of a single app or all different apps. The API is outlined in the following:

VirtualSpace API

classes:

Tile{Vector2[] area}

Frame{Tile tile, float time}

Maneuver{Frame[] frames}

Tick{float duration, float variance}

Valuation{float weight, float preparationTime, float executionTime}

functions :

Vector2[] ProvideProbabilityDistribution ()

Tick[] ProvideTicks ()

void AdaptToTicks(float[] ticks)

Valuation[] Evaluate (Maneuver[] maneuvers)

void Execute(**Maneuver maneuver**)

VirtualSpace's API acts in the following five stages:

Initial positioning

When an app registers with VirtualSpace, the system arranges its place in the tracking volume, coordinating it with the other apps that are already running. Figure 14 shows an example. Here, the badminton app predicts an uneven spatial probability distribution for its user, as users tend to go back to the baseline whenever they can. Let us assume now that we have for example a second badminton user (playing against a separate AI). If two badminton apps were placed in identical orientation, it would result in frequent collisions. VirtualSpace avoids this by requesting a sample of each app's spatial probability distribution. Then, the system tries out all possible positions and orientations of the apps to minimize overlap. In this example, the system decided to rotate the second badminton app by 180 degrees.

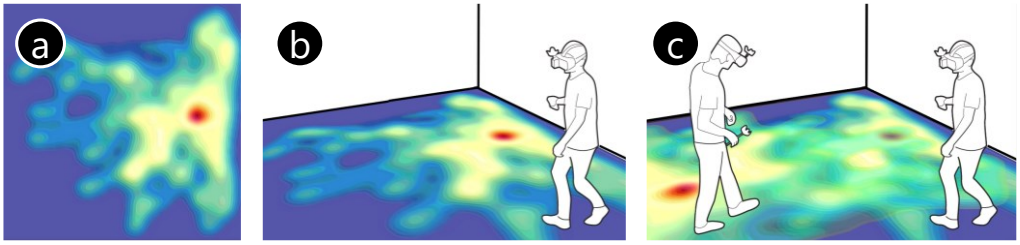

Figure 14: Spatial probability distribution. (a) Spatial probability distribution based on three minutes of gameplay of the badminton app (axis lengths: 4m). (b) VirtualSpace places the first badminton user without rotational offset. (c) The second badminton user is placed at a 180-degree angle, minimizing the overlap in their probability distributions.

Ranking and selecting maneuvers

To help VirtualSpace perform maneuvers fitting all apps, each app informs VirtualSpace about the usefulness for each potential maneuver. The system deduces the potential maneuvers using a simple state machine (e.g., after the *focus* maneuver the system is in the 'focus' state, no *rotation* maneuver is possible, but a *defocus* or *switch* maneuver). Apps then valuate potential maneuvers. The badminton app, for example, generally values those maneuvers highly that allow bringing the user back to baseline, to previously unused areas, or larger areas in case of a fast past rally, as seen in Figure 15b. Pac-Man's erratic movements or badminton's reach typically require more space, leading to higher valuations of larger areas, or areas where there are more uneaten pellets (Figure 15f). Additionally, apps provide the system with information on how fast they can comply with the suggested maneuver. They provide the preparation time (delay until the maneuver starts, in which incentive is placed) and the execution time (time for moving tiles, in which users follow incentives). This information can also be provided in linear dependencies (e.g., preparation and execution time add to a specific value). For example, Pac-Man gives the sum of preparation time and maneuver duration by computing the average player speed. VirtualSpace processes this information using a linear solver [35].

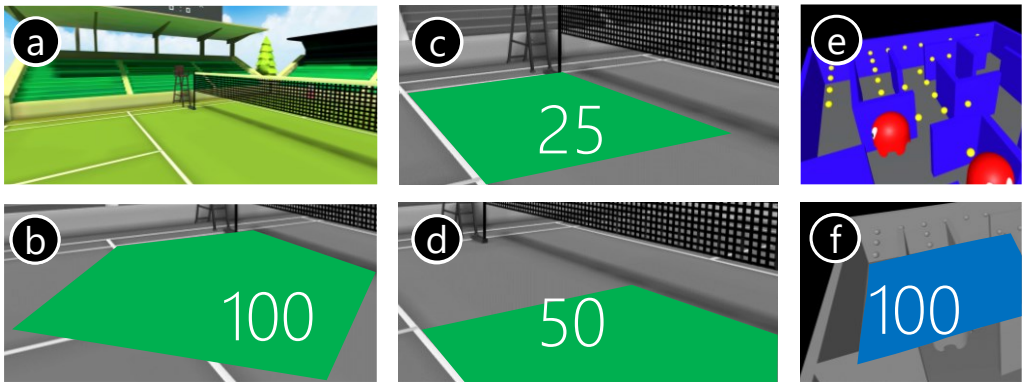

Figure 15: (a) The badminton app wants to play a fast paced rally. (b) It values more space more highly, as given through the focus maneuver. (c) Staying in front of the net has a lower value and (d) switching places has a higher value. (e) For Pac-Man, when only a few pellets are left, those areas are valued higher (f).

The system now decides which maneuver to invoke. It adds up the utility for each maneuver reported by each app and picks the maneuver that maximizes utility across apps. A *focus* maneuver for the first app, for example, implies three corresponding *defocus* maneuvers for the other users, which might not have the same utility as a *rotation*. The system also tries to ensure that apps receive similar utility over time and tries to avoid maneuvers subject to timing mismatches between apps.

Scheduling maneuvers and synchronizing applications

The scheduling we utilize is fixed-priority pre-emptive scheduling, a common scheduling system for real-time applications. Other schedulers, such as round-robin, would have resulted in lower utility for the apps, as apps cannot lead their users where they need them to be, or, such as shortest-remaining-time, cannot be used, as unpredictable variations in apps task times need to be taken into account.

When the system decides on the maneuver to perform, it also needs to determine the time until the maneuver can start, i.e. when the applications are ready. In badminton for example, this should be within the narrow time frame when the enemy AI can hit the birdie and the app can show the trajectory to the impact point. To ensure this, the system synchronizes the apps by using what we call *ticks*, events in which apps have the possibility to influence their user's movement. In the example

of badminton, a tick is the moment the enemy AI hits. The moment that the system starts a maneuver should be when application ticks are in sync. The applications constantly provide information on their ticks using linear conditions, in badminton for example ticks occur once every full exchange. Each tick is given an allowed variance (which for our apps increases over time) together with a preference (e.g., badminton prefers quick exchanges but can let the AI play differently to synchronize). Ticks are synchronized by the system sending back master ticks, which the apps adapt to. The system can then compute the delay time, whose requirements were sent with the maneuver evaluations (see above). When a synced tick is then the same as the delay time, VirtualSpace can start the maneuver as we now ensured that every app can direct its users at that time.

Apps follow selected and scheduled maneuvers

The system informs apps about the upcoming maneuvers. It does so by sending a sequence of what we call *frames*, information about assigned areas at given times, from which apps can derive the need to place incentives. The system computes assigned areas, the tiles, as Voronoi tessellations to keep them convex, this enables apps to easily compute which paths their user can walk on. Apps respond to maneuvers by placing rewards and obstacles inside their virtual environment, thus guiding the user to the next tile.

The whole process described in this section is executed multiple times in parallel, to ensure quick interaction rates. While apps synchronize for the next maneuver, they already evaluate the next couple of maneuvers. This queue length is determined by the apps themselves.

3.3 Example applications

We have implemented four apps to run with VirtualSpace. Here we show how the API is implemented.

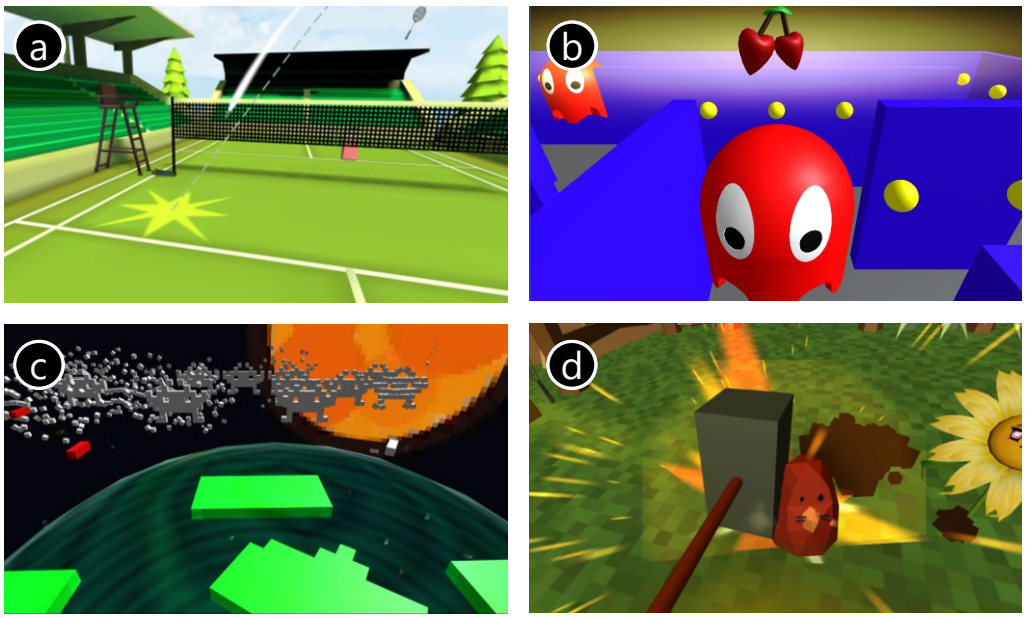

Figure 16: We have implemented four apps to run with VirtualSpace. (a) Badminton, (b) Pac-Man, (c) Space Invaders, (d) Whac-A-Mole. All apps that implement the API of VirtualSpaces work together in any combination.

Badminton

A user plays badminton against an AI. The user sees the birdie's trajectory as soon as the AI hits, and an estimated trajectory before that. We display the trajectory at all times to avoid the natural tendency of users to walk towards the court's center. We made the virtual field slightly larger than the tracking area as users would optimize their movement and not walk onto the side of their field. When valuating maneuvers, badminton valuates larger areas stronger for quick ball exchanges that do not need to sync, also areas where the user has not yet hit that often, and areas with higher overlap to the court's service line. The minimum preparation time is the maximum time of a ball-exchange (dynamically computed), the minimum duration of the maneuver is based on the velocity of the birdie.

Pac-Man

As in the traditional Pac-Man, the user's goal is to collect all yellow pellets in a labyrinth. Ghosts position themselves so that the user does not pass into another user's area. Cherries provide additional points and

thus draw users towards the next tile. They focus the user's attention using a distinct sound and a halo effect. When valuating maneuvers, Pac-Man valuates higher areas with more yellow pellets. The blue pill from Pac-Man serves as an in-app purchase (Figure 12); when collecting it, for a short duration the app valuations are generally higher than the ones of other applications.

Space Invaders

As in the original arcade game, the user controls a spaceship to shoot enemy ships. Enemy ships are placed on all four sides of the environment. An area with protective blocks incentivizes the player to be in a certain area. Space Invaders valuates maneuvers that require the user to walk more. Preparation time uses a fixed delay and the duration is capped by the player's maximum walking speed.

Whac-A-Mole

The user is placed within a fenced area and is given a hammer to hit moles that spawn from the ground. Negative incentives, like flowers, are used to counteract the urge of users to walk towards the center. Audio cues given by the moles taunt the user to look towards them, in case the user faces the wrong way. Whac-A-Mole valuates areas higher where fewer moles have been hit. Preparation time and duration are similar to Space Invaders.

3.4 Design considerations for applications

For creators of VirtualSpace applications, the following guidelines should be taken into account.

Planning and reacting

The design of apps is driven by two constraints: planning and reacting. VirtualSpace benefits from apps planning ahead of time and being able to react quickly. For a set of apps to run within the system, it must be assured that the apps react to the maneuvers that result from their own

planning. Thus, the reaction time of the slowest application must be smaller to the planning time of the app least capable of planning. As an example, in our badminton app the worst possible time to perform a maneuver is right after the AI hits, as the app needs another full exchange (roughly two seconds), to influence the user again, so the reaction time is two seconds. It provides its valuations two exchanges ahead, taking roughly four seconds. If the badminton app was not able to do that, the birdie would not arrive where the app would want it to be. In Pac-Man, ghosts would need to travel too fast, etc.

Incentive considerations

Applications should always be able to place incentives inside the users' field of view, additionally to using audio cues. Also, both negative and positive incentives should be used – we found positive incentives (e.g., the birdie or moles) to work better for maneuvers, while negative incentives, such as the Pac-Man ghost, work better for keeping users within their area if no maneuver is active. If applications do not provide negative incentives, fallback routines from VirtualSpace happen more often, which lowers immersion.

Cognitive processing delay

Every app affords a different cognitive load to the user, so users' reaction time heavily depends on cognitive processing of the given incentives. For example, space invaders' protective obstacles start to move 400ms before the maneuver starts, giving the user a time to process and react. The other apps, which we also intentionally based on existing games to lower reaction times, are quite similar. In Pac-Man, the cherry, as a strong positive incentive, pops up seconds before the maneuver starts, but only when it does, do the ghosts give way for the user to collect it. Pac-Man needs to allow for additional cognitive processing, as it includes more numerous game elements. We found iterative testing

crucial for deducing the correct lead for placing incentives in the virtual environment, as various factors contribute to human reaction time [50].

3.5 Implementation and hardware

The applications were developed in Unity3D. The backend uses C# with the libraries Clipper [18], Protobuf [73] and Gurobi [35]. To allow researchers to replicate our work, we provided the full source code of VirtualSpace and the apps in C#/Unity3D [99]. The space is tracked using the Vive Lighthouse system with eight trackers [100]. The tracking information is forwarded via UDP to head-mounted displays (four GearVRs with Samsung S6 running Android) as shown in Figure 17.

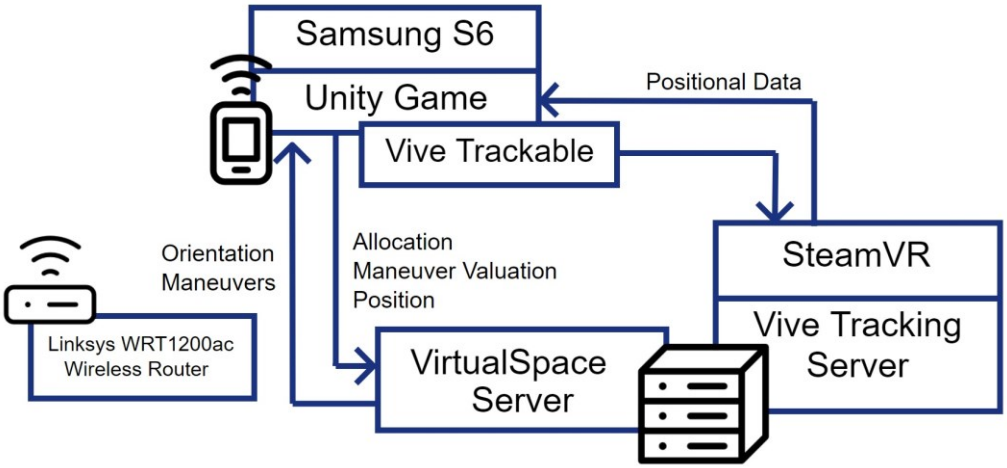

Figure 17: VirtualSpace's setup.

3.6 User study

To better understand the resulting experience of allowing VirtualSpace to manage actual virtual reality apps, we conducted a user study in which we compared VirtualSpace against the most commonly used approach, i.e., static pre-allocation of space ([3] uses a similar baseline). Our main hypothesis was that VirtualSpace provides more space coverage per user. Additionally, we assumed that VirtualSpace improves the experience, measured in ratings of confinement, enjoyment and presence.

Interface conditions

There were two space allocation conditions. In the *VirtualSpace* condition, we tested our system with the described maneuvers. In the *pre-allocated* condition each participant was confined to a static tile (no maneuvers).

We did not use *whole space allocation* as an additional baseline condition, in which participants take turns in using the whole space, as this assumes unlimited space for each app, which outperforms any technique.

In both conditions, participants walked in a 4x4m space, with a safety boundary of 60 cm between allocated areas. We deemed this boundary length sufficient after initial pilot studies.

Task and procedure

Participants were split into groups of four with each given one of the four games described above. Prior to the tasks, participants had one minute of training, in which they were playing their game alone in the tracking volume. Each group had two sessions, one for each space allocation condition. The order of conditions was counterbalanced. During the first session, participants played their app for five minutes, while the first space allocation strategy was applied. Participants then filled in a questionnaire about their experience containing two questions: "How much did you enjoy the experience?" and "How confined did you feel?" (Likert scale, 1-7) and the realism subscale of the presence questionnaire [104]. During the second session, participants played the same app again for five minutes, while the other space allocation strategy was applied (within-subject design) and then again filled in another questionnaire. Each participant thus played their app twice, five minutes in each session, only using one app.

Participants

We recruited 16 participants from our organization (9 female, 7 male, age 28.8 ± 3.1 years), forming four groups. Seven participants had prior

experience with virtual reality (one had experienced real-walking). The remaining nine participants had never tried virtual reality before.

Results

As shown in Figure 18, users felt more confined in the pre-allocation condition, while VirtualSpace provided a greater sense of freedom ($p < .05$, $t(15) = -1.79$, one-sided). However, we could not observe a statistically significant difference in enjoyment ($p = .12$, $t(15) = 1.20$, one-sided).

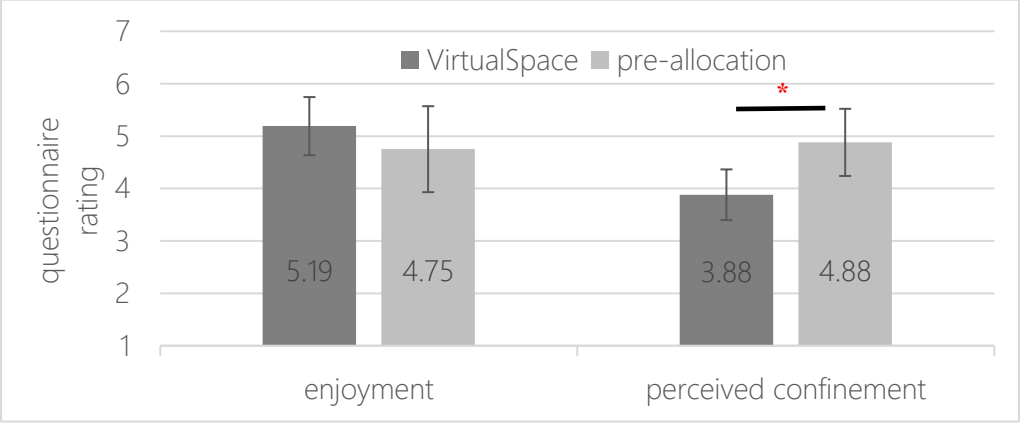

Figure 18: Participants felt more confined in the static pre-allocation condition, while VirtualSpace provides a greater sense of freedom.

Space coverage was measured using the position of the tracked head mount throughout one session given a 30cm radius. Accounting all users, in the VirtualSpace condition 52 rotation maneuvers were conducted, 10 focus maneuvers, and 3 switch maneuvers. This led to significantly larger space coverage of 15.51 m^2 ± 2.01 per user in the VirtualSpace condition when compared to the space coverage of 3.46 m^2 ± 1.02 per user in the control condition ($p < .001$, $t(15) = 21.12$, one-sided). Figure 19 depicts these results.

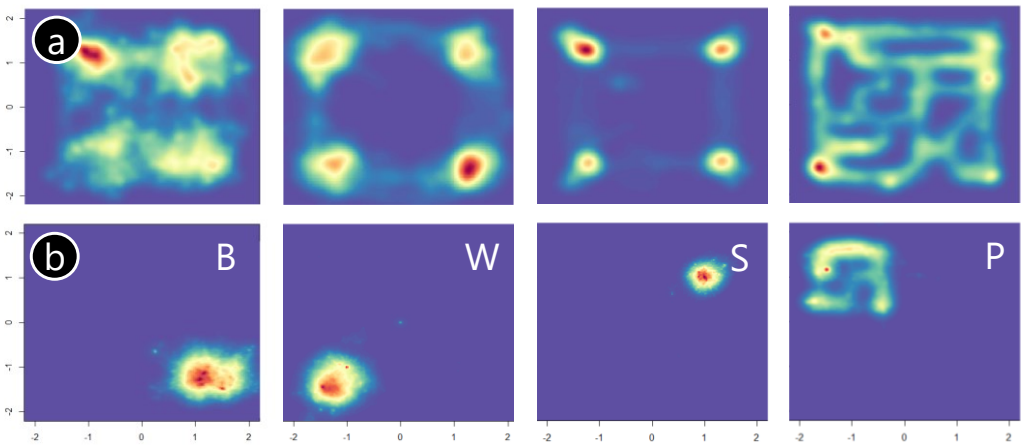

Figure 19: The four apps' space coverage for all participants (apps by initials). (a) Participants covered more space using VirtualSpace than when using (b) a static pre-allocation (axis lengths: 4m).

For our analysis, we define a collision as a mutually unintended contact of two participants. We observed 7 collisions in total. Two collisions occurred in the first group, none in the second, two in the third and three in the fourth. We recorded differences between apps; the badminton app was involved in six collisions while the remaining apps were only involved in three collisions or less. All collisions occurred after a participant remained standing still when their tile moved; participants were absorbed in their experience, but did not follow the given incentive or the fallback visualization. Collisions occurred mostly during the first minutes of gameplay, and further inspection revealed training effects; participants were less likely to breach (measured as the time ratio of being outside one's assigned tile, see Figure 20) if they had already experienced the app in the control condition ($p < .05$, $t(14) = 2.28$, one-sided).

Despite participants occasionally experiencing the fallback mechanisms to keep them within their designated area, results did not show a significant difference in the realism subscale of the presence questionnaire [104] between conditions ($p = .38$, $t(15) = 0.90$, two-sided).

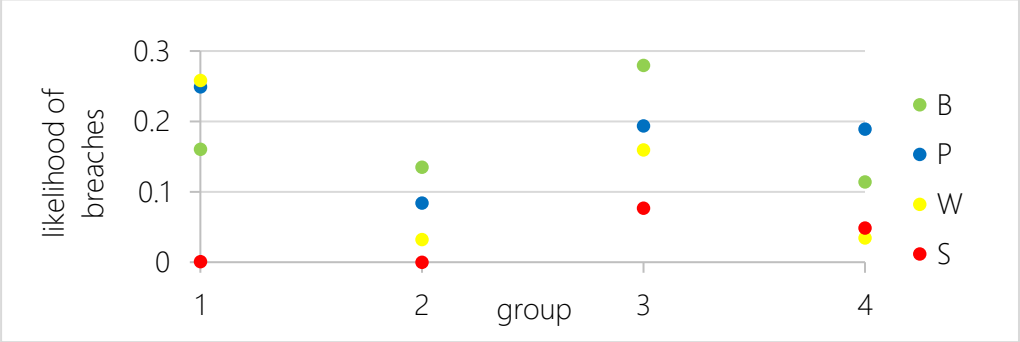

Figure 20: Participants compliance varied. Those who experienced the control condition first (group 2+4) were less likely to breach through their assigned areas, suggesting a training effect. Also, certain apps seem to keep their users within their tiles more effectively (apps by initials).

Qualitative feedback

Participants felt they had more space and made comments such as: "I just enjoyed having more space", "In the beginning [pre-allocation] it was not only more boring, but I felt way more cramped".

Participants remarked that they trusted the system when apps kept participants within their assigned areas. However, when users did not recognize the incentive and reacted too late or apps needed to rely on the fallback to keep participants in place, participants' trust decreased: "The trust into the system was relatively high, that you do not walk into one another", "It then comes at quite a shock when they [fallbacks] pop up", "I actually felt safe – until that [collision] happened", "I felt I reacted too slowly".

Participants commented on the other users' participation. "Actually, I liked knowing that there were other people", "I would not play this alone, so it was actually even fun to maybe hit somebody".

Discussion

Our main finding is that VirtualSpace outperforms static pre-allocation of space, as users feel less confined and can cover more space. This is true even for high user densities such as for our four users within 16m^2. This density is an improvement over related work on space reduction techniques for real-walking in virtual reality, such as *redirected*

walking [74] or dynamic layout generation for overlapping virtual spaces [98], in which individual space requirements are of a far greater magnitude.

We believe our overloading technique can be used not only as an alternative, but also as a complement to space compression techniques. Researchers [3, 43] have already shown that redirected walking can apply to multiple users, while again using relatively large areas, which VirtualSpace can help to utilize more effectively. VirtualSpace on the other hand could use redirection techniques to orient users, instead of relying on in-game mechanics.

Our current system can be extended to more space, more users, and more types of applications. The shown apps were isolating experiences, as is typical for virtual reality. VirtualSpace can naturally be extended to multi-user applications, mobile scenarios, and spaces of different shape or size. Since primarily rotation maneuvers have been carried out, a more even coverage of space still seems possible. Increasing the size of the tracking space while maintaining user density could also improve ratings, as the effect of real-walking might prove higher in larger areas on enjoyment and perceived confinement. Even though apps can move users anywhere within the tracking space, getting there may be subject to a delay. Along the same lines, apps have to be able to generally comply when another app requests a maneuver. We found this to be more acceptable for apps with short interaction cycles, such as casual games or sports games rather than story-driven games, such as adventure games. We imagine, although not tested yet, that VirtualSpace can run only one story-driven app with strong constraints together with any number of casual games with softer constraints.

For our study we used some pre-existing metrics (collisions [5], presence [104]) and some that we specifically developed (perceived confinement, breach ratio, space coverage). To make their results comparable, we propose that future work also use these metrics.

3.7 Conclusion on concurrent use of virtualized space

VirtualSpace is a novel software system that makes real-walking more space-efficient by having users immersed in different virtual reality experiences share the same physical space. VirtualSpace achieves this by containing each app in a smaller tile. Frequent *maneuvers* allow apps to incentivize their users to walk across the entire physical space, thereby allowing each app to progress its narrative and to prevent users from noticing that they are confined to a tile. This strategy enables VirtualSpace to achieve packings of the unprecedented density of four users in 16m², as we demonstrated in our user study.

This technique of *overloading* physical space will help real-walking applications expand to different scenarios such as mobile virtual reality, where space needs to be shared with other people, or application domains requiring space with high interaction rates, such as e-sports, as demonstrated with our demo apps.

We have argued that VirtualSpace can be expanded to more users and other tracking volumes. However, applications using overloading are still designed with a fixed tracking volume in mind, in our case 16m². As argued in the introduction, this makes overloading unsuitable for a lot of setups in the home; either applications do not make use of possible surplus space the user has available, or it does not even run at all for tracking volumes that do not fit the required 16m². Another limitation is that VirtualSpace primarily focuses on arcade-style applications without much focus on story, and applications still assume a space devoid of physical objects. We also assume spaces to be devoid of any physical objects. In the following chapters we show how to remedy these shortcomings.

4 VIRTUALIZING PROPS FOR CONCURRENT USE

We have presented how real-walking experiences can be instantiated in tracking volumes that are shared across users. In this chapter we show how the concept of concurrent use cannot only be applied to physical space, but to physical *objects* as well, or *props*.

Physical props have benefit of potentially providing passive haptics to their users. Passive haptics [40] increases immersion in virtual reality by making all virtual objects in the scene tangible through co-located physical props. Passive haptics props are typically used in virtual reality arcades (e.g., The Void [101] and others [25]) and have not caught on for home use, as such experiences will not be run often enough to justify the expense, time and storage effort for new passive haptics props for every new experience.

We want to reduce this cost by requiring fewer physical props per virtual object. We show two techniques. The first is a synchronized use of props across users. The second is multi-purposing a single prop, requiring only one physical prop for multiple virtual objects in a single-user environment.

4.1 SYNCHRONIZED USE OF PHYSICAL PROPS FOR HAPTIC FEEDBACK*

The term *human actuation* [12] describes the idea of providing force feedback to users through *people*, here called human actuators. *Mutual* human actuation [14] is a variation of the idea of human actuation and works without dedicated human actuators. The key idea is to have pairs of users immersed in different virtual worlds that provide human actuation to *each other* at the same time through the use of shared props.

This is similar to *VirtualSpace*. We find that the idea of *overloading* applies not only to physical space, but to physical props as well; we allow multiple users immersed in *different* experiences to share the *same* physical prop at the *same* time without being aware of each other.

Five different types of props enable different types of force-feedback. The first, continuous force, we show in Figure 21.

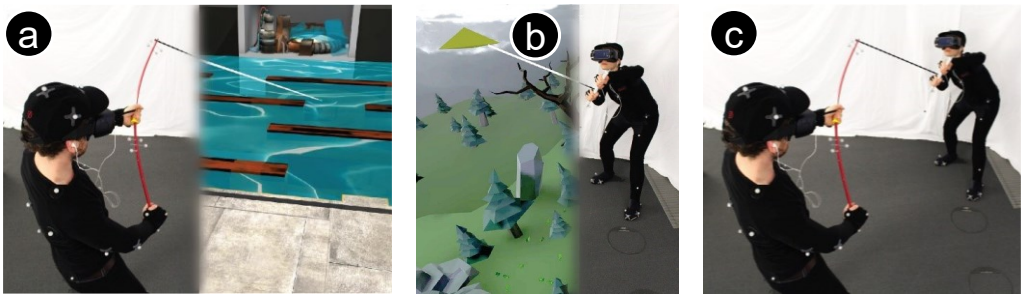

Figure 21: Continuous force using a rod with string and handle. (a) The user, alone in his virtual world, is trying to pull a huge creature out of the water. He feels how the creature is struggling and pulling on his fishing rod. (b) At the same time, this other user, also alone in her virtual world, is struggling to control her kite during a heavy storm, which is whipping her kite through the air. (c) While users' experiences of force might suggest the presence of a force feedback machine, they instead experience force feedback using a shared prop that transmit forces between them.

---

*This section (text and images) is derived from: Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. 2017. Mutual Human Actuation. In Proc. UIST '17, pg. 797–805. It was already presented in Lung-Pan Cheng. 2018. Human Actuation. We shortly summarize this work here, due to co-authorship and overlap with the topic of managing physical resources.

Figure 22 shows enabling passive haptics through rearranging props. The same prop is used multiple times.

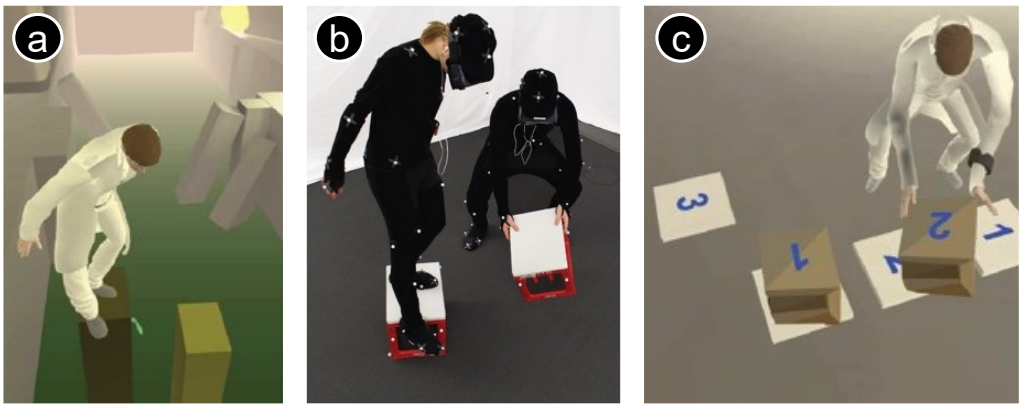

Figure 22: Rearranging physical props, here boxes from plastic and foam. (a) One user is waiting for a series of pillars to rise in order to allow her to cross the pit ahead of her. As she lowers her right foot to probe the space below, she can feel the void. Once she sees that the pillar has fully risen, she can step on it. (b) In the meanwhile, the other user is solving a puzzle that requires him to place numbered boxes on matching tiles. In reality, one user places boxes for the other.

There are three other types of force feedback (see [14]). These are: moving (using a chair), impact (using a foam stick), and contactless sensations (using a foam board).

4.2 Multi-purposing of props through covert scene changes

Another technique for having multiple virtual objects share one physical prop is through unnoticeable manipulation of the virtual scene. We implemented a software tool called *Mise-Unseen* that applies such unnoticeable runtime changes to any virtual environment by means of eye tracking. Mise-Unseen thereby makes efficient use of physical objects by mapping multiple virtual objects to them.

This technique is similar to *impossible spaces* [88], which achieves re-use of physical space by re-routing the user to new rooms, so that changes to the virtual environment happen outside the user's view. Mise-Unseen applies these changes also *inside* the user's field of view without them noticing.

Figure 23 shows and describes how Mise-Unseen re-uses a prop for multiple virtual medieval tools and weapons inside this forge environment.

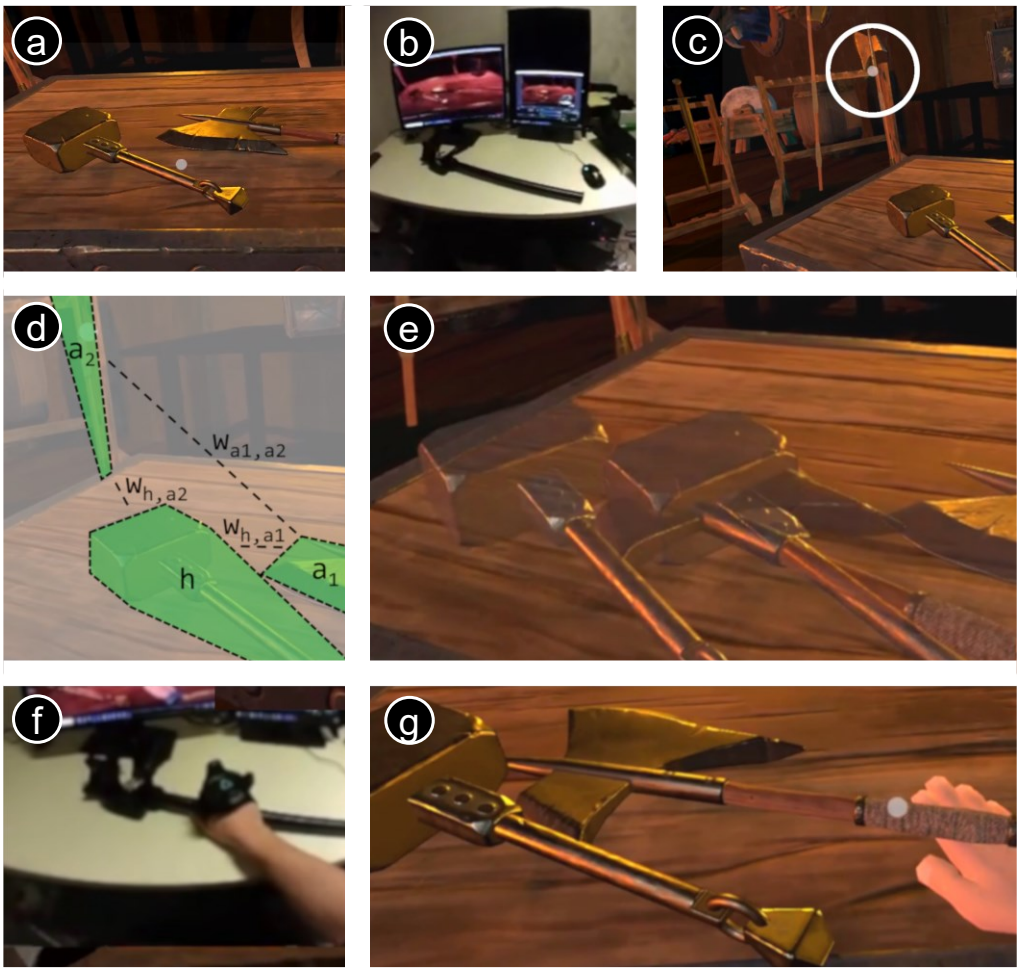

Figure 23: Passive haptics enabled by eye-tracking. (a) In this forge environment, the user may pick up any of the two weapons on the table. (b) However, only one physical prop is available. The weapons are already clustered around the prop, but neither one is mapped onto it. (c) The user looks at an axe a_2 on the wall informing Mise-Unseen that the user probably more interested in axes than hammers. (d) The spatial memory model is a weighted graph that represents all objects w, a_1, a_2 as nodes. As the user looks at weapons, the weights change to represent internalized (here allocentric) distances between objects. (e) Mise-Unseen now shifts the axe onto the physical prop. (f) The user picks up the axe virtually and (g) physically.

Creators who want to tell a story with a limited physical prop set would require such a software tool. Of course, creators could try to manipulate the virtual scene outside the users view, but the user may freely look

around the scene and therefore might observe a change while it is happening. As the field of view of headsets will become even larger in the future, these changes will become even more noticeable. Today's commercial virtual reality headsets have a field of view of up to 120° horizontal and we expect them to eventually reach the maximum of human vision (~180° horizontal). Contrast this to traditional cinematography, where the creator (the director) fully controls the viewpoint and camera focal length. There are other problems, for example the user could remember the scene being different or even anticipate a change.

The whole purpose of our software tool is to prevent the user from noticing exactly those changes, to not break the user's perceived consistency of the scenery. Our tool covertly injects these changes to the displayed content inside the user's field of view. Mise-Unseen leverages gaze tracking to create models of user attention, intention, and spatial memory to determine if and when to inject a change and combines these models with a variety of masking techniques (e.g., change saliency, distractors, etc.). These changes are completely authored and can be used by any application.

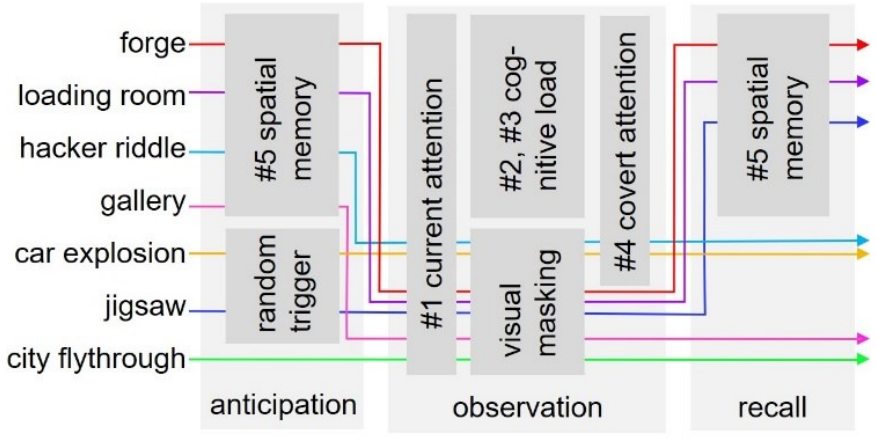

Figure 24: Mise-Unseen prevents *anticipation*, *observation*, and *recall* of a scene change using five attention models together with visual masking techniques. Our different example applications make use of a different sub-set of these five models.

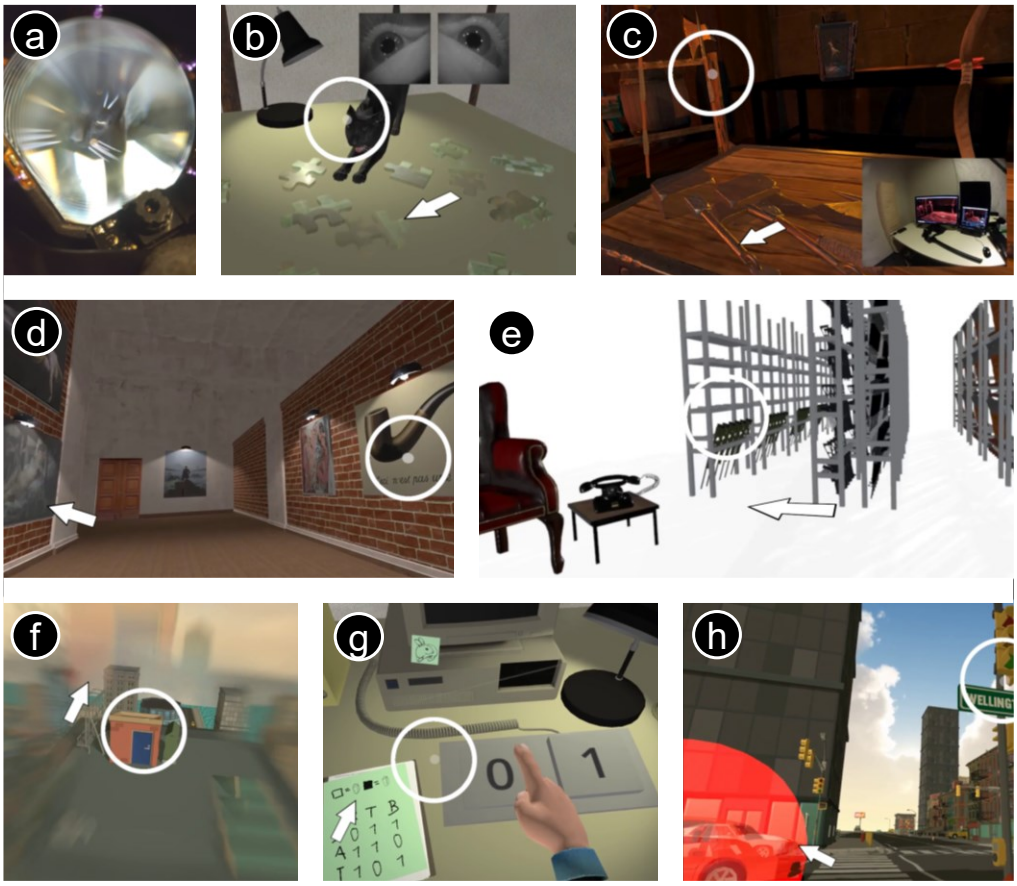

Figure 25: (a) Mise-Unseen uses eye-tracking in virtual reality headsets to hide otherwise obvious changes that occur inside the user's field of view. (b) Mise-Unseen injects those changes (arrow) while the user is focusing elsewhere (circle): For example, we move the jigsaw pieces using a cross-fade to help the user. (c) We match the virtual axe's position to the haptic prop following the detected user interest. (d) We swap the gallery painting to adapt to the user's detected interest in modern art. (e) We shift storage racks while walking to adapt to a lack of physical space. (f) We can reduce motion sickness during teleportation by blending static images outside the fovea. (g) We can update hints to prevent the user from solving this riddle without deducing its solution. (h) We hide the low fidelity of this 'explosion'.

Figure 24 shows the tool's internal layering of these models, to prevent *observation* of the change when it is happening, *recall* after it has happened and *anticipation* when it is about to happen.

Figure 25 shows the system's other uses through example applications. Figure 25e sketches out how to re-use space using eye-tracking. In the storage room environment the user is walking towards two lines of storage racks. Mise-Unseen shifts the rack that the user appears more interested in, so that the user can walk through. Instead of

matching weapons onto a prop, one of two walking paths are matched onto space.

4.3 Conclusion on concurrent use of virtualized props

We have shown two techniques that map physical props onto multiple virtual objects. The first is overloading, but for physical props, and the second is unnoticeably injecting runtime changes to virtual environments. These techniques illustrate how physical resources such as props can be used concurrently.

However, our virtualization of physical props is only partial. The two techniques can be embedded inside production pipelines for real-walking experiences, but they still require authoring with somewhat of an a-priori knowledge of which props to use, similar to VirtualSpace which requires knowledge of the tracking volume's size and shape. This contrasts with the idea of full virtualization. Similar to related work, such as *annexing reality* [39], we gain some level of abstraction by adapting virtual objects to the shape of given physical props. Props cannot be fully arbitrary and creators must still require props that fall into the scope of these techniques.

We made concurrent use of physical props and physical space possible. In the next chapter we show how to achieve an *independence* between complex real-walking experiences and any given physical space.

5 VIRTUALIZING THE EXTENT AND SHAPE OF PHYSICAL SPACE

When developing a real-walking virtual reality experience, creators generally design virtual locations to fit a specific tracking volume. Unfortunately, this prevents the resulting experience from running on a smaller or differently shaped tracking volume.

In this chapter we address this issue. In accordance to our analogy to virtualization and operating systems we provide an abstraction layer between any virtual reality experience and the physical space it occupies. This requires a tracking volume-independent representation of real-walking experiences. This is the core of *Scenograph*, the software system we describe and evaluate in the following.

5.1 SCENOGRAPH: 1:1 EXPERIENCES FOR ANY PHYSICAL SPACE

The core of Scenograph is a tracking volume-independent representation of real-walking experiences. Instead of designing for a tracking volume of specific size and shape, Scenograph lets designers specify an experience independent of the tracking volume. Scenograph instantiates the experience to a tracking volume of arbitrary size and shape. Based on the representation of the experience, Scenograph splits virtual locations into smaller ones while maintaining any narrative

structure (Figure 1, page 16). During instantiation Scenograph applies space compression techniques like *impossible spaces* [88]. In our user study, participants' ratings of realism decreased significantly when we used existing techniques to map a 25m² experience to 9m² and an L-shaped 8m² tracking volume. In contrast, ratings did not differ when we used Scenograph to instantiate the experience.

The interface to Scenograph is an editor in which application designers define the unfolding of their real-walking experience, which we can define as the sum of all possible interaction sequences with virtual objects and their arrangement in virtual locations, or *scenes* to be more general.

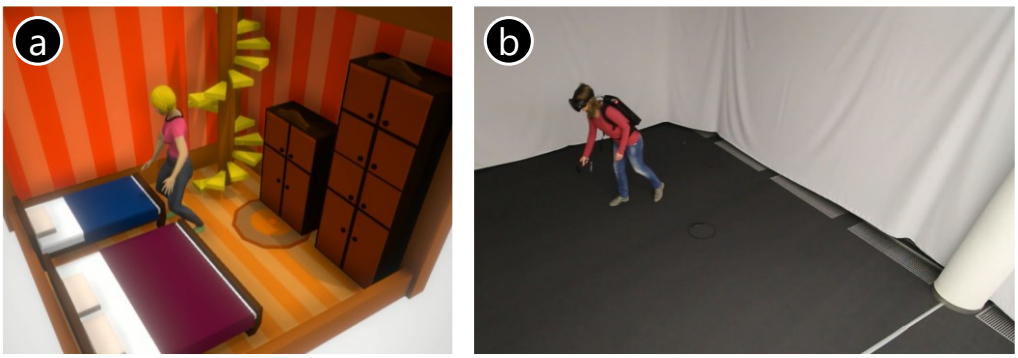

Figure 26: (a) Our adaptation of the fairy tale 'Goldilocks and the Three Bears', in which Goldilocks maliciously enters the home of the three bears, eats their porridge, sits on their chairs and sleeps in their beds. (b) The user in our tracking space of 5m x 5m.

Example application: Goldilocks and the three bears

We demonstrate this process through an example application based on the 19th-century fairytale 'Goldilocks and the three bears' ([84], see Figure 26 for our design). Naturally, Scenograph allows for the design of any application that can be procedurally generated, 'Goldilocks' is a good example, as the narrative unfolds within one connected environment, a 'dark forest', the 'three bears' home' within that forest, and an 'upstairs bedroom' of that home (see Figure 1).

5.2 Our tracking volume-independence data structure

Internally, Scenograph represents the experience as a bipartite graph, specifically, a petri-net. As such it has *transitions* and *nodes*. Nodes are either *spatial* or *logical*. The *spatial* nodes are the scenes of the experience, in our case of Goldilocks the three scenes 'dark forest', 'three bears' home', and 'upstairs bedroom'. The *logical* nodes are the states that enable story progression when the user interacts with certain objects, for example after eating 'little bear's porridge', the user changes into the 'tired' state, so that they can now interact with the 'chairs'. These 'chairs' and 'bowls of porridge' are *transitions* which connect the two kinds of nodes, the virtual objects in the scenes. Transitions pass tokens between input and output nodes. In Figure 27 we see that the 'porridge' is a transition that takes tokens away from the 'hungry' state. Transitions that pass tokens between spatial nodes are corridors that, similar to portals in [88], enable unperceivably switching between scenes. The first 'door' for example, passes tokens from the 'forest' (spatial) and 'curious' (logical) nodes to the 'home' (spatial) and 'hungry' (logical) node.

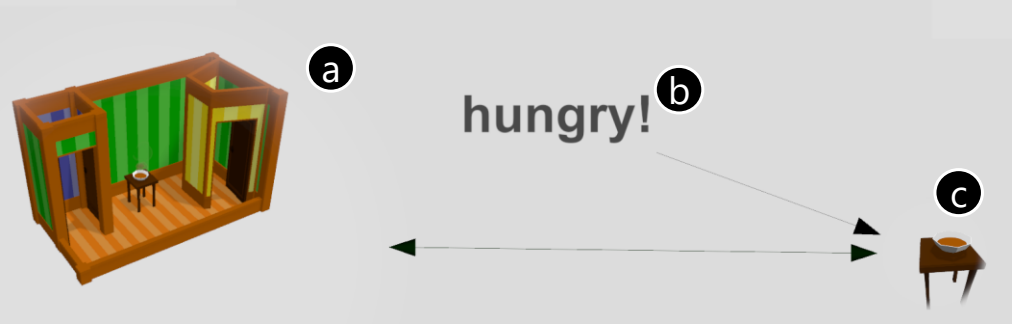

Figure 27: Scenograph encodes all possible interaction sequences in a petri-net. (a) Spatial nodes define the virtual scenes, such as this 'three bears' home'. (b) Logical nodes express states, such as Goldilocks being 'hungry'. (c) Transitions, here a 'porridge', let users switch states and scenes, they populate the virtual scenes as objects.

A petri-net is suitable for encoding any direction a narrative may take (see Figure 28). This data structure can be expressed in any environment the application is developed in, here we used Unity3D/C# (see implementation section).

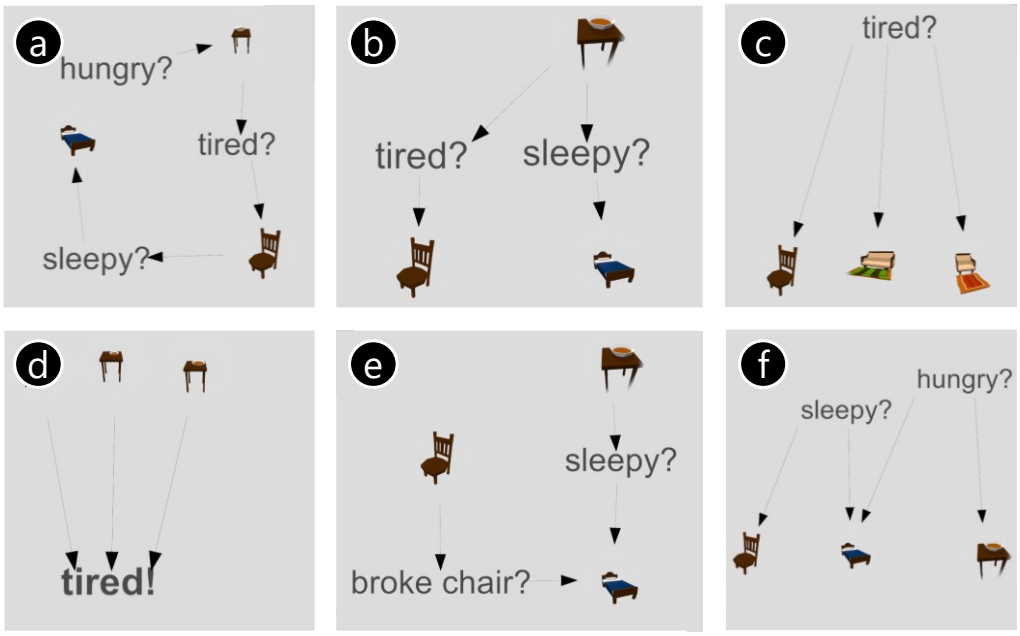

Figure 28: Examples of different narrative arrangements. (a) Sequence: the 'porridge', 'chair' and 'bed' are accessed in a set order. (b) Concurrency: eating the 'porridge' allows using the 'chair' and the 'bed'. (c) Conflict: the user needs to decide between one of the 'chairs'. (d) Merge: only eating all 'bowls of porridge' makes Goldilocks 'tired'. (e) Synchronization: sitting on the 'chair' and eating the 'porridge' is necessary before sleeping in the 'bed'. (f) Confusion: a mixture of conflict and merge, using the 'bed' denies eating the 'porridge' or sitting in the 'chair'.

Scenograph adapts the scenes to the available space by splitting the nodes into multiple instances – this is the core value of the system. As shown before in Figure 1 (page 16)Figure 29, limiting the tracking volume from 25m² to 8m² results in splitting the 'home' node into four nodes. Figure 29 shows how limiting it to the L-shaped 12m² results in a split into two nodes, as that space does not fit all three 'bowls of porridge' and three 'chairs'. Scenograph re-links the transitions to maintain narrative structure. Scenograph takes the petri-net and the available tracking volume as input and transforms them into the new layout.

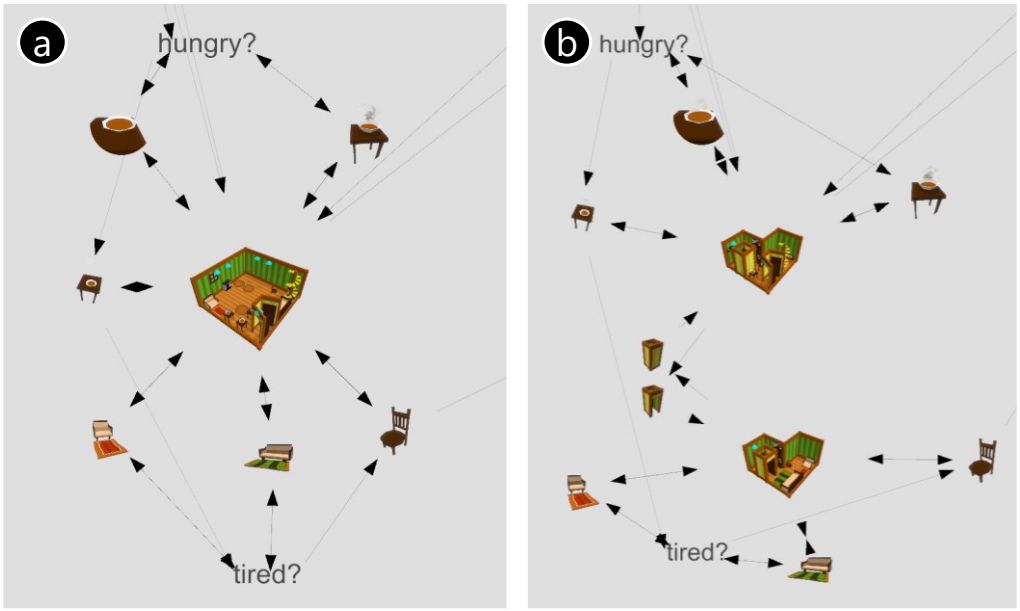

Figure 29: Scenograph splits the 'home' node into two as the designed for 25m^2 get reduced to an L-shaped 12m^2. (a) This node has six transitions (three bowls of porridge followed by three chairs). (b) The bowls of porridge are placed in the first (upper) node, the chairs in the second, as they are interacted with *after* the bowls of porridge.

The end-user has no knowledge of Scenograph's data structure. The system merely requires a specification of the user's tracking volume in the form of a polygon. This specification can be provided by a range of tracking technologies. The designer provides the volume-independent representation of the experience.

Authoring tracking volume-independent experiences

To create an experience, the designer must define all possible interaction sequences, i.e., the connections between spatial nodes, logical nodes, and transitions. This specification follows a bipartite graph structure, as nodes and transitions can only connect to each other, not to themselves. After creating the necessary virtual objects of the experience (3D models, animations, etc.), the designer connects transitions to nodes and vice versa by clicking on the corresponding objects in the editor's graph representation (Figure 30).

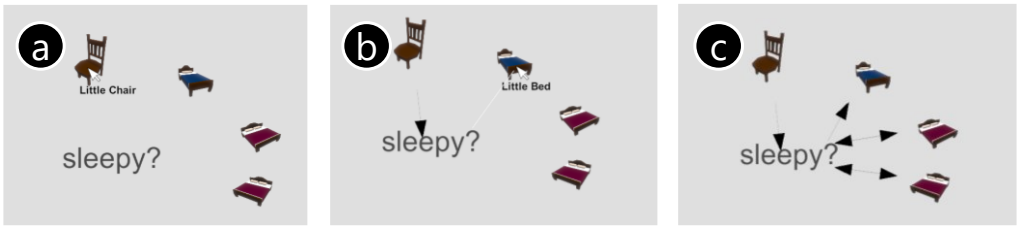

Figure 30: The designer authors the Goldilocks experience by connecting nodes and transitions. (a) For the user to interact with the 'beds' after 'little bear's chair', the designer connects the 'chair transition to a state ('sleepy'), (b) and that state to the 'beds'. (c) The final narrative structure.

Scenograph can also attribute multiple transitions to the same virtual object. In Figure 30, for example, one side of the 3D model of the big bed is considered 'mama bear's bed', the other side 'papa bear's bed'. A staircase can be used for 'going upwards' and 'going downwards', etc.

The classic Goldilocks fairy tale is sequential (like most stories). Goldilocks eats the porridge, sits on the chairs, then lies on the beds. She always starts with the item of papa bear, then mama bear and finally the small bear. In our rendition of Goldilocks, we instead chose to allow for user decisions: any porridge is edible until small bear's porridge is eaten, sitting on any chair is possible until the user sits on small bear's chair, etc. A sequentially told story in Scenograph would provide an easy to solve problem (cut off the story when we run out of space, then split the location node). Scenograph usefulness increases with the complexity of the narrative, for example, when user decisions are involved, since the problem of where to split the nodes is then non-trivial. On the other end of the spectrum, if there is no logical connection at all (all objects can be interacted with anytime without consequences or story progression), then the decision where to split the node would be arbitrary.

5.3 Scenograph algorithm

Scenograph runs with any app that provides the graph structure described above. The following five steps describe Scenograph's algorithm in detail.

Step 1: TAKE IN GIVEN PHYSICAL SPACE

Scenograph requires a specification of the given tracking volume in the form of a polygon, as well as the resolution into which the space gets virtualized, which is provided by the application. In our lab setup we have 5m x 5m available, and our example applications is designed for a resolution of 1m x 1m so that Scenograph tessellates the space into 5 x 5 tiles (Figure 31).

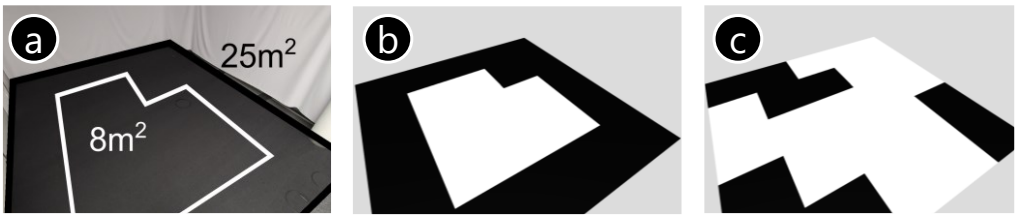

Figure 31: (a) Our tracking volume of 5m x 5m and a sub-space of an L-shaped 8m^2. (b) Scenograph tesselates the area using 'Goldilocks' resolution of 1m x 1m to either 25 or 8 tiles, the configuration we used for our user study. (c) Many other sizes and shapes are possible.

The generated chunks of tiles are allocated to the system for the generation of the virtual scenes. Different physical setups will thus result in different scenes (Figure 32). Note that each experience requires a minimum size based on its largest virtual object, e.g., for Goldilocks this is the sofa with 2m x 2m. Scenograph allows to switch setups at runtime, for example if space gets occupied or freed up suddenly. In this case Scenograph re-instantiates the experience, however, it maintains the current logical and spatial nodes (e.g., 'frightened' and 'upstairs bedroom').

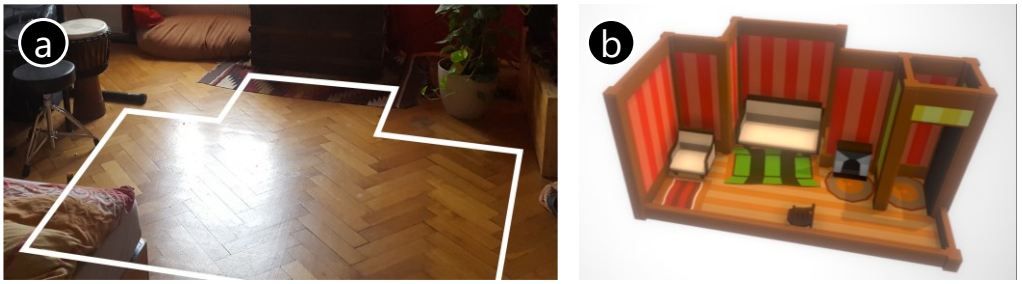

Figure 32: Scenograph takes various tracking volumes as input, such as (a) this living room. (b) Scenograph then instantiates the experience.

Step 2: Pack virtual objects into given space

Scenograph requires its applications to declare the space requirements for the virtual objects the transitions are paired with. Space requirements entail the object's length and width as hard constraints, and placement preferences (close to a wall, middle of the room, etc.) as soft constraints with a cost function.

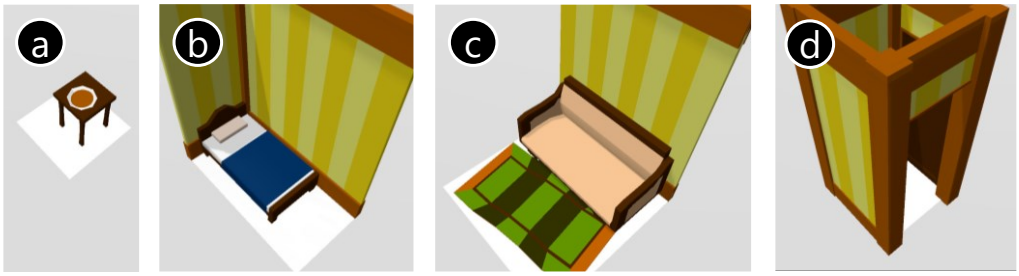

Figure 33: (a) A 'porridge' is a virtual object that requires 1m^2, (b) 'papa chair' needs 4m^2 and a wall, (c) 'little bed' 1m x 2m and a corner, (d) a generic corridor-portal connecting scenes takes 1m^2 and a corner.

Scenograph now needs to determine if all nodes can support their transitions, i.e., if the virtual objects' can be packed together onto the space the scene is given. The system offers different packing algorithms, (*best-fit*, using simulated annealing), or random placement (*first-fit*, using random placement). Given a smaller or differently shaped tracking volume, the packing algorithm might not find a solution and the node is then not able to support its transitions. In Figure 29 we cannot pack three 'bowls of porridge' and three 'chairs' into L-shaped 12m^2, thus Scenograph splits the 'home' node into two.

Step 3: Re-use space by splitting spatial nodes

To determine the number of splits per node and the distribution of transition onto split nodes, Scenograph uses divisive hierarchical clustering. The distance computation between transition pairs, required for our hierarchical clustering, uses a simple semi-decision technique (distance of 1 if two transitions can be interacted with in any order, 2 if one transition needs to be interacted with after the other, 3 for neither).

Scenograph now needs to evaluate which clustering to take for each node.

Scenograph cannot linearly iterate through each node separately to find the right clustering, as some virtual objects need to be instantiated on the same tiles in more than one scene (transitions with different spatial nodes as input and output, such as a door). This means that Scenograph needs to consider all possible clusterings for all nodes in parallel, making the packing problem 3-dimensional (width, depth, occurrence in nodes).

Scenograph iterates through all possible clustering in an informed manner. The number of clusters and therefore the potential connected scenes to consider is exponential in the number of virtual objects. Our hierarchical clustering does not reduce this amount, it merely sorts all potential clusterings based on the conceptual distance between transitions. The number of virtual objects and thus of potential clusters is different for each scene. For example, our three nodes have one transition ('forest' has a door leading to 'home'), eight transitions ('home' has 3 'bowls of porridge', 3 'chairs', 1 'door', 1 'stairway') and three transitions ('upstairs bedroom' has 2 'beds', 1 'stairway'), leaving $2^0 + 2^7 + 2^2 = 512$ possibilities. Each possibility corresponds to a certain clustering depth per node, which we represent using a mixed radix numeric system (e.g., $1_1 2_6 1_3$ corresponds to splitting the second node in two). We iterate through this numeric system linearly first based on the checksum of this clusterings number (to reduce the number of nodes) and then on its order within the hierarchical clustering (maximizing proximity of virtual objects that are also conceptually close).

After finding the right split of nodes, additional transitions might need to be generated to connect them (Figure 34). This problem has multiple solutions (e.g., sequence, loop, full connection, etc.). However, as a tradeoff, added transitions lead to a linear decrease of available space per cluster (to a maximum of number of clusters – 1 for full connection). This makes this decision a design problem and

consequently the application designer defines how scenes should be connected, our example application uses a sequence.

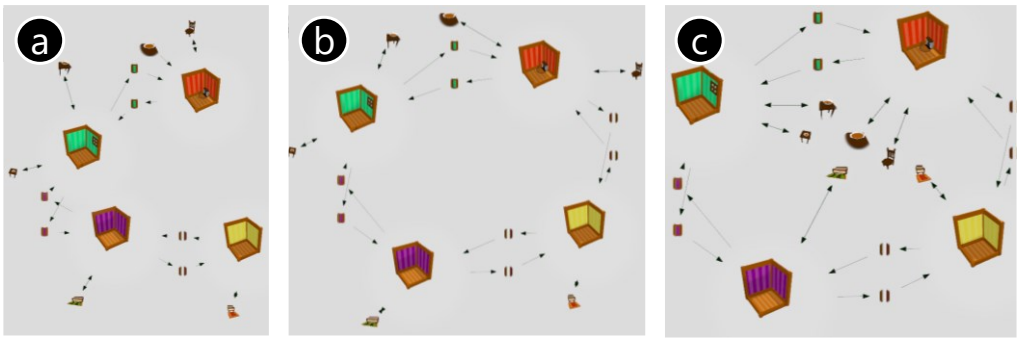

Figure 34: These four 4m^2 nodes can be linked in (a) a sequence, (b) a loop, (c) completely.

Algorithm 1: Re-use of space

Input: P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules), M – mapping of P onto given physical space

Output: scene splits based on P and M

$N \leftarrow GetSpatialNodes(P)$
$r \leftarrow GetResolution()$
$s \leftarrow GetAvailableSpace(r)$
for each n **in** N **do**
    $V \leftarrow P.GetVirtualObjectsOfNode(n)$
    $d \leftarrow P.GetDistanceMatrix(V)$
    for each v_a, v_b **in** V **with** $a \neq b$ **do** *
        $R_a \leftarrow v_a.GetPlacementRules()$
        for each r in R_a **do**
            if $v_b = r.RelativeVirtualObject()$ **then**
                $d(a,b) \leftarrow 0$
            end if
        end for
    end for
    $g \leftarrow GenerateDendrogram(d,V)$
    $C \leftarrow GetPossibleClustersSorted(g)$
    while $M.VirtualObjectsOverlap(V,P)$ **do** *
        $c \leftarrow C.Next()$

```
        N_c ← n.SplitNode (c)
        for each n_i in N_c do
            V_i ← P.GetVirtualObjectsOfNode(n_i)
            if M.VirtualObjectsOverlap(V_i, P) then
                P.ReplaceNode(n, N_c)
            end if
        end for
    end while
end for
for each n in GetSpatialNodes(P) do
    GenerateScene(n)
end for
```

Step 4: INSTANTIATE THE EXPERIENCE

The virtual scenes are now generated. Each virtual object is placed onto the space the packing algorithms allocated for it. Afterwards, the rest of the scene is generated. While an application may create the visuals for each scene itself, Scenograph offers some default procedural generation algorithms to create the scene automatically. The application just provides the decorative objects, walls, floors, etc., together with placement constraints (e.g., a wall element with a window cannot be used next to occupied tiles). Scenograph loads and unloads scenes dynamically depending on the users' interaction with the virtual objects or where they walk. Scenograph uses corridors similar to *impossible spaces* to connect scenes [88], which serve as portals or locks. The L-shaped 12m² space in Figure 29 can thus be used twice to fit the 'bowls of porridge' as well as the 'chairs'. Less overlap makes the technique less perceptible. However, since we focus on small spaces, Scenograph here fully overlaps the scenes.

5.4 IMPLEMENTATION AND HARDWARE

The system was implemented in C#, the example application and editor interface in Unity3D. We used ALGLIB [2] for clustering. Participants

wore a backpack PC for tether-less virtual reality. For tracking we used the VIVE system [100]. To allow researchers to replicate our work, we provided the full source code online [77].

5.5 User study

Scenograph can fit experiences into tracking volumes with different physical sizes and shapes. To validate this, we let Scenograph instantiate our 'Goldilocks' experience for different tracking volumes. We compared these instantiations to two commonly used locomotion techniques for small tracking volumes (see Figure 35): *motion scaling*, the altered mapping of users' physical to virtual motion (similar to *Seven-League-Boots* [45]), and *teleportation*, a form of artificial locomotion [11]. We hypothesized that Scenograph would receive higher ratings of realism and enjoyment.

Interface conditions

We compared six conditions.

In *large-square*, we contained every scene of the narrative in a single volume by allocating a total of 5m x 5m to the application. No split of any location nodes and no motion scaling needed to be applied. This condition emulates the best possible situation as every scene can incorporate all virtual objects.

In *small-square-Scenograph* we allocated 3m x 3m, emulating a smaller tracking volume where splitting the experience is necessary as not all virtual objects fit into the space.

In *small-shape-Scenograph* we allocated the small space in the form of a L-shape ($8m^2$), emulating variable shape of users' tracking volume.

In *small-square-scaled* we rendered the same $25m^2$ virtual world as in *large-square* but applied a motion mapping of 1:5/3 to all translations of the participant so that only 3m x 3m are needed.

In *small-shape*-scaled we also rendered the 25m² virtual world, but for 2m x 2m tracking volume, the smallest quadratic shape fitting into the L-shape of *small-shape-Scenograph*, for a 1:5/2 motion mapping.

In *small-teleport*, we rendered the same 25m² virtual world as in *large-square* but enabled a teleport mechanism: users could point to any virtual position and teleport themselves to it. The limited tracking volume, here our *small* 3m x 3m, was visualized like in most real-walking systems (such as Vive [100], or Oculus [72]) by rendering *chaperone bounds*.

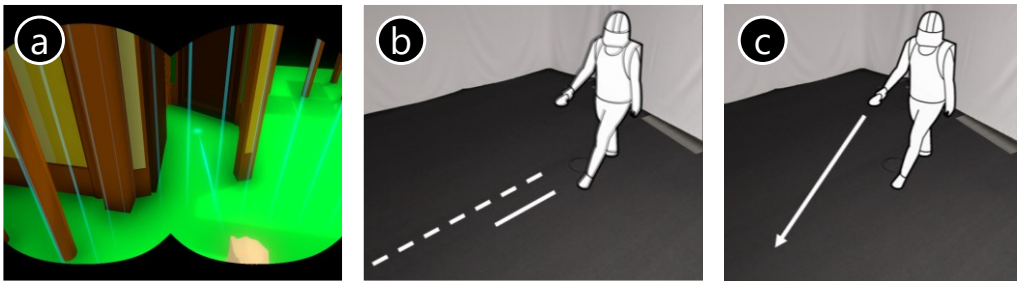

Figure 35: (a) Our first control condition implements a *teleport* functionality and displays chaperone bounds to keep users from leaving the tracking volume. (b) The participant travels by clicking or by walking. (c) Our second control condition, *scaled motion*, changes the mapping of physical motion (solid line) to virtual motion (dashed line).

Apparatus

The study took place in our 5m x 5m empty lab space (Figure 31). The experience we used was the 'Goldilocks' application described earlier. Participants could interact with the virtual objects of the application (*transitions*, see system section) with their hand controller.

Task and procedure

We gave participants a summary of the story and let them experience the virtual reality application before testing, so it was familiar prior to all conditions. Each participant then traversed once through each condition for a total of six sessions. The order of the conditions was randomized for each user. Each session took about two minutes to complete. After each session, participants filled in a questionnaire.

Measurements

The questionnaire contained two statements to complete on a 1-7 Likert scale: "I enjoyed the experience (not at all-very much)" and "Moving around felt (artificial-natural)" for our measures *enjoyment* and *realism*.

Participants

We recruited twelve participants from our organization (six female, six male, mean age 23.6 sd 3.6 years). Seven of the participants had previous experience with virtual reality, two of whom with real-walking. The remaining five participants had never tried virtual reality before.

Hypotheses

Our hypotheses compared Scenograph to both control conditions: For *small* spaces, we hypothesized the *Scenograph* conditions would be perceived as more realistic and enjoyable than the *scaled* conditions for both *square* (**H1)** and *shape* **(H2**). The *Scenograph* conditions would be perceived as more realistic and enjoyable than the *teleport* condition (**H3, H4**). The *small-square-Scenograph* condition would be perceived as less realistic and enjoyable than the *large-square* condition (**H5**).

Results

This section reports Bonferroni-corrected *p*-values. T-tests (applied as suggested by [70]) are one-sided.

Figure 36 shows our main finding. As hypothesized, the *Scenograph* condition outperformed the *scaled* conditions in both space setups *square* ($p < .05$, $t(11) = 3.22$), supporting **H1**, and *shape* ($p < .001$, $t(11) = 9.88$), supporting **H2**. *Scenograph* conditions performed better than the *teleport* control (both $p < .01$, $t(11) = 6.23$), supporting **H3** and **H4**. Surprisingly, no difference was found between *large-square* and *small-square-Scenograph* ($p = .73$, $t(11) = 1.27$), so **H5** cannot be supported in terms of realism (underlining the system's effectiveness).

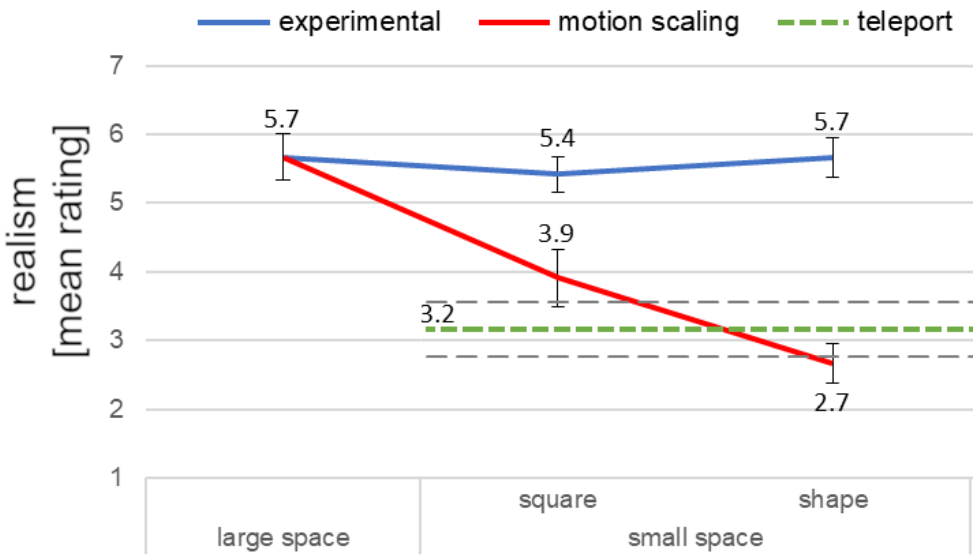

Figure 36: Participants rated the *Scenograph* conditions more realistic than both control conditions *scaled motion* and *teleport* (bars show std. error).

Enjoyment was generally high, the *large-square* space condition unsurprisingly scored the highest (mean 5.8, sd 0.9) together with *small-teleport* (5.8 sd 1.1). The *Scenograph* conditions performed almost equally well for *small-square* (5.6 sd 0.9), and *small-shape* (5.6 sd 1.1). The *scaled* conditions scored lower for both *small-square* (4.8 sd 1.5) and *small-shape* (4.3 sd 1.5), so that the difference in mean scores was significant for *small-shape* ($p < .05$, $t(11) = 3.61$), supporting **H2**. We did not find support for the remaining hypotheses in terms of enjoyment, notably again **H5** ($p = .80$, $t(11) = 1.2$).

To make our results comparable for future studies, we want to report on the distance walked by the users. On average, users walked 37.0m in the *large-square* condition, 28.0m in *small-square-Scenograph*, 29.0m in *small-shape-Scenograph*, 22.3m in *small-square-scaled*, 14.2m in *small-shape-scaled*, 13.4m in *small-teleport*.

Qualitative feedback

Participants found walking experiences generated by Scenograph to be most "realistic" (P3, P7, P8) as movement felt most "natural" (P3, P7).

One participant stated that changing rooms was sometimes "irritating" (P2), another found the experience got "harder" (P6).

Participants found teleportation to be "very artificial, but cool" (P6, P12). Others stated that the chaperone bounds broke immersion (P8, P9). One participant argued for its naturalness due to its "familiarity" (P1).

Motion scaling was perceived as "non-intuitive" (P5, P6, P7) and "uncomfortable" (P2, P4, P12). Participants liked the experience better if it was "less fast" (P10). Some participants enjoyed it as it was "like in a movie, but not natural" (P9), though "it became more natural over time" (P8).

Discussion

Our main finding is that Scenograph can create real-walking experiences in virtual reality for tracking volumes of any size and shape. Scenograph's experiences were rated more realistic than two commonly used locomotion techniques, namely *instant teleportation* [11] and scaling the mapping of physical to virtual motion (e.g., [45]). This leads us to conclude that Scenograph degrades more gracefully with limitations of tracking volume with qualitative feedback supporting these findings.

Space compression impacts realism differently than enjoyment. Both teleportation and motion scaling still provided enjoyable experiences to the participants. Also, the size of the tracking volume did not measurably impact enjoyment (comparing $9m^2$ to $25m^2$). The number of participants might have played into not discovering the hypothesized differences. It seems, however, that even when space compression is high and becomes more perceivable to the user (teleport, stronger motion scaling, higher overlap of impossible spaces) it affects realism first rather than enjoyment. Only when the compression becomes very high (motion scaling for L-shaped $8m^2$) does it also impact enjoyment. Based on our results, we can thus only make claims about Scenograph's impact on realism.

The study design has some limitations. No set of control conditions exists that can sufficiently represent the broad spectrum of available locomotion techniques. We chose our control conditions as mere representatives of those existing techniques; motion scaling as one instance of locomotion techniques which distort the mapping of physical to virtual motion (instead of, e.g., *redirected walking* [74]) and teleportation as one instance of virtual locomotion techniques (instead of, e.g., *walking-in-place* [93]). It can be argued that virtual locomotion always comes at the cost of perceived realism and that the motion mapping can be optimized, as for example done in *seven-league-boots* [45]. A great way of optimizing is by knowing about the intended destination of travel, which make these techniques particularly suitable for sequential narratives (*seven-league-boots* scales motion and *redirected walking* alters the yaw rotation towards the direction of travel). For that reason, we imagine that these or similar techniques can play into future iterations of the system. As of now, our study showed that Scenograph is a working system that supports end-users in any tracking volume and compares well to the chosen control conditions.

5.6 Conclusion on space-independent experiences

We have presented Scenograph, a novel software system that supports the design of real-walking experiences, which can adapt to any tracking volume's size and shape. For creators, who generally design experiences to fit a specific tracking volume, this allows them to define experiences with complex storylines independent of the users physical space, and it allows users to run these experiences in whatever space they have available. The core of Scenograph is a tracking volume-independent representation of real-walking experiences. This representation is not spatial until instantiated. Scenograph instantiates the experience after splitting the locations into smaller ones while maintaining narrative structure. In our user study, participants' ratings of realism decreased significantly when existing techniques were used to map a 25m^2

experience to 9m² and an L-shaped 8m² tracking volume. In contrast, ratings did not differ when Scenograph was used to instantiate the experience.

We have the idea of virtualizing physical space further and, apart from sharing resources, we now enable space-independent real-walking in complex experiences. Scenograph thus provides another crucial component for making real-walking experiences available to consumers.

There is a final limitation. We assume spaces devoid of any physical objects which arguably are found inside most users' tracking volumes. In the next chapter, we show how virtualization of sets of such physical objects can be achieved.

6 Virtualizing sets of props within physical space

We now have gathered almost all attributes for a full virtualization of physical space. We allow for concurrent use of tracking volumes, concurrent use of one prop, and we allow complex experiences to run in tracking volumes of arbitrary size and shape. We now also want experiences to run in physical spaces that contain arbitrary *sets* of physical *props*, so that experiences can truly run anywhere.

This chapter directly builds on the last as it extends our abstraction layer between virtual reality experience and the arbitrary physical spaces they can occupy, so that these spaces can contain any set of physical props. This is the goal of *Stuff-Haptics*, the software system that extends *Scenograph* and which we describe and evaluate in the following.

6.1 Stuff-Haptics: 1:1 experiences for any set of props

We present Stuff-Haptics, a software tool that provides consistent experiences across limited, uncurated sets of physical props. Unlike traditional passive haptics experiences, Stuff-Haptics does not require the set of physical props to be premeditated, and allows experiences to run anywhere, specifically in users' homes with *stuff* already found in the home, such as hairdryers, half-used candles, etc. Stuff-Haptics

accomplishes this with a prop-independent representation of the storyline,which consists of the narrative structure of events and the spatial layouting rules for mapping of virtual content onto the available physical prop set. Stuff-Haptics lays out the virtual objects onto an annotated scan of the stuff in users' homes. Given limited sets of props, Stuff-Haptics offers different solutions to best preserve the experience, either through re-use of props or by pruning the storyline to still fit the most important story elements. In our user studies, Stuff-Haptics both successfully ran the same experience for different users' prop sets and ran different passive haptics experiences on the same prop set.

Example applications

We continue to demonstrate our approach using 'Goldilocks and the three bears' as our main example, as shown previously in Figure 2 on page 17, and which we now detail in Figure 37.

For creators, Stuff-Haptics allows them to define the spatial layouting and narrative structure of a story. For example, the relative placement rules of virtual objects onto props ('papa bear's chair' is bigger than 'mama bear's chair') and logical progression (after sitting in the 'small bear's chair', Goldilocks can go to the 'upstairs bedroom'). It specifically also allows creators to choose between different automated solutions for handling limited prop sets; re-using props or altering the storyline. For users, Stuff-Haptics allows for different sets of props as well as switching props at runtime.

To show that Stuff-Haptics extends to other experiences as well, we further add two more example experiences (Figure 38). Our 'The Golden Key' includes a warming fire place and some stairs to walk on to reach a key to then progress into the next level. Our 'Mona Lisa Heist' experience includes touching a showcase and afterwards some pictures on the wall.

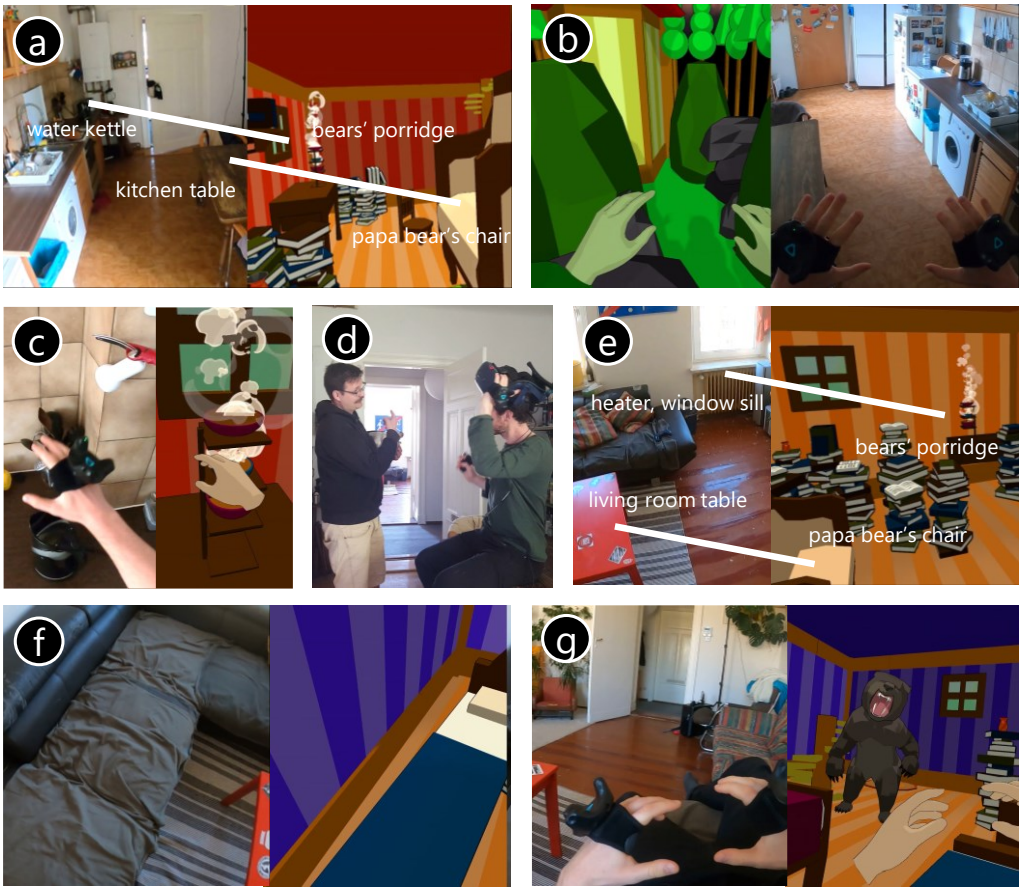

Figure 37: (a) The user invokes a virtual reality experience, 'Goldilocks and the three bears' in their regular living environment. Our software tool Stuff-Haptics provides passive haptics to the experience by automatically lining up relevant virtual objects with physical props of matching haptic qualities, such as the 'bears' three bowls of porridge' with the physical water kettle, and 'papa bear's chair' with the physical table. (b) When the user is walking through their kitchen they are 'Goldilocks', a little girl that finds herself lost in the woods, and stumbles across the home of the three bears. (c) In the three bears' home, Goldilocks sees the bears' three bowls of porridge and finds one to be 'too hot', another 'too cold' and the third 'just right' – the user feels the heat of the porridge as they hold one hand over the kitchen's water kettle. Stuff-Haptics places the 'too hot' porridge closer to the physical water heater to be physically hotter than the 'just right' porridge. Goldilocks, then tired, tries to sit down on the different chairs. The first one feels 'too big' – the user sits down on the kitchen table. (d) Back in reality, when a flatmate needs to use the kitchen, our user moves on to another room. (e) Stuff-Haptics redesigns the virtual experience on the fly. It now places the chair onto the physical living room table. The story continues, another chair is 'still too big' – Stuff-Haptics placed the 'too big' chair onto the physical table instead of the 'too small' one. After sitting on the chairs, Goldilocks is tired. (f) Goldilocks enters the upstairs bedroom, finds the first bed 'too hard', another 'too soft' and the third 'just right' – Stuff-Haptics placed the 'just right' bed onto this physical sofa and the 'too hard' again on the table, so that the user lies down on the sofa. (g) As the user lies down, Goldilocks falls asleep, and when the bears return home, Goldilocks wakes up and runs away.

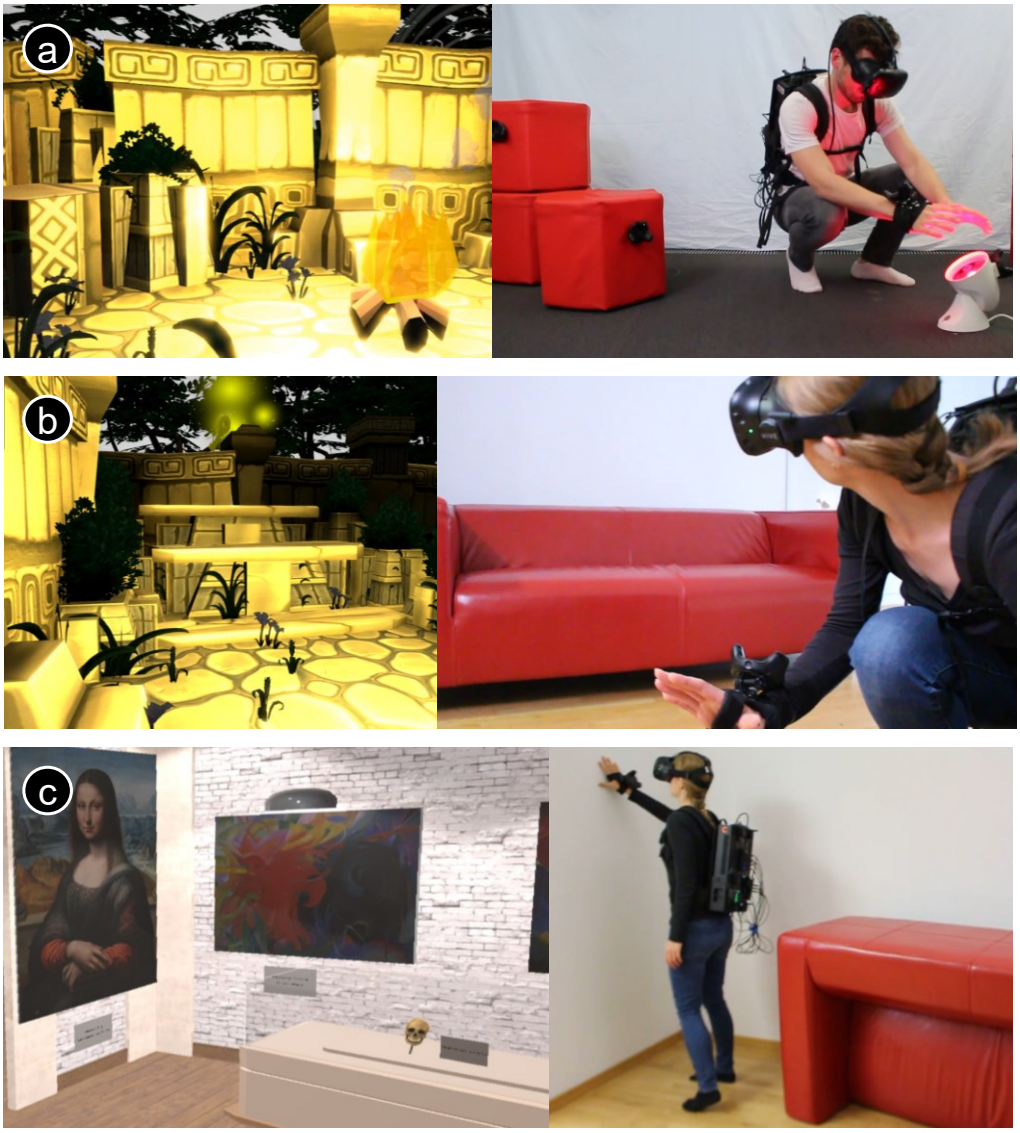

Figure 38: Other Stuff-Haptics experiences. (a) Here some virtual stairs and a fire map to these physical chairs and heat-lamp. (b) For another user the chairs are mapped onto the physical sofa. (c) This art gallery maps onto the physical walls.

6.2 Our space- and prop-independent data structure

Stuff-Haptics combines a story's structure and its parametric models into a space- and prop-independent representation of a real-walking and passive haptics experience.

The placement constraints and parametric models inform the layouting process. The storyline informs the handling of insufficient props, such as *splitting* of environments for re-use, similar to re-use of space, as previously detailed in chapter 5, step 3 ("Re-use space by

splitting spatial nodes"). We show how those placement constraints and storyline now inform each other for their specific tasks.

The procedural generation of a virtual environment requires a valuation of possible mappings from virtual content onto the scan of the physical environment. This requires a description of the range of matching props (sets) for each virtual object (set). We follow a similar approach as in previous work [32, 39] and use parametric model descriptions together with a constraint solver to place virtual content. We added material tags (such as 'surface elasticity', or 'heat emission') for a better haptic experience.

AUTHORING PROP- AND SPACE INDEPENDENT EXPERIENCES

Figure 39 shows the 'bears' beds', one example of such a parametrically defined virtual object set that can be authored.

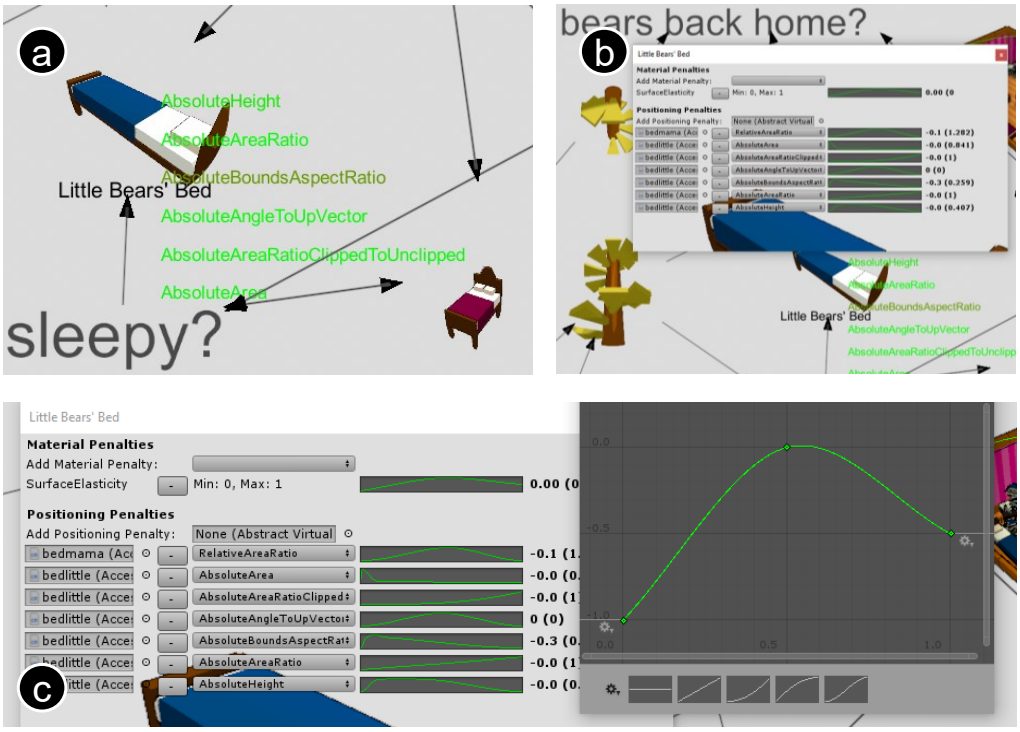

Figure 39: We extend our authoring pipeline with defining procedural content. (a) By hovering over 'little bear's bed' we preview its parametric description. (b) This popup window details all parameters and current fitness values for 'little bear's bed'. (c) We change the valuation for 'surface elasticity' to get a somewhat soft 'bed'.

In the example of 'little bear's bed', different props can be considered somewhat functional substitutes such as tables, sofas, etc. The fitness value assigned to each prop depends on the *material properties* of the physical counterpart (does the prop have a certain 'surface elasticity'), its *absolute positioning properties* (does the surface it is mapped to have accessible height), and its *relative positioning properties* (is it close or far from the other bed counterparts). Figure 39c shows that 'little bear's bed' requires a 'surface elasticity' of ideally 0.5 as a material property of fitting geometry (here we define surface elasticity within a range of 0 to 1, each surface in the room scan is annotated with such a value). 'Little bear's bed' further requires a surface of certain height (~0.5m), size (~1m^2), size ratio and horizontal alignment. Ideally it also differs in size from the other beds. Here we describe the additional constraint that it needs to have a relative size of ~0.5 compared to 'mama bear's bed' (which again defines its own constraints, and so on). Using these rules Stuff-Haptics can match the bed for example to a couch in our user's living room, or a chair in the kitchen. If a fit seems bad, parameters can be further tweaked. The Goldilocks experience is based on a total of 91 of such parametric rules formulated by the experience designer.

Other objects work similarly. For the user to feel its warmth, 'mama bear's porridge' uses the kitchen's water kettle, or the window sill above the heater in the living room. For the user to sit down, 'papa bear's chair' uses the kitchen table or the table in the living room. 'Papa bear's chair' needs to be bigger than the others, 'little bear's bed' needs to be softer, 'mama bear's porridge' needs to be colder, etc. Stuff-Haptics offers a total of 28 different placement rules (easily extendable in the source code). The 13 virtual objects in Goldilocks have a total of 91 rules, with the 'porridge' defining the least amount with 4 and 'mama bear's bed' defining the most with 9. These rules can be applied to other experiences. The virtual staircase in 'The Golden Key', for example, consists of a maximum of five steps, each of which encompasses a set of soft

constraints. The first step requires a horizontal surface of certain height, aspect ratio and material properties. Consecutive steps additionally require its predecessor to be placed within a certain horizontal and vertical distance and are not instantiated when those requirements are not met, so that the mapping algorithm might find a match in a sofa, but not in a shelf.

Some objects are virtual only, without physical counterpart, such as the bears themselves, as Goldilocks does not touch or interact with them anyway, or 'little bear's chair', which breaks under Goldilocks weight and might be better suited for complementing haptic rendering devices. Those objects still define their own constraints ('little bear' is placed on the floor next to its 'bed', 'parent bears' are downstairs next to the 'bowls of porridge' and 'chairs'). Some virtual objects are not strictly relevant but support others, such as the table placed underneath the porridge.

6.3 Stuff-Haptics extended algorithm

Stuff-Haptics runs with any app that provides both Scenograph's graph structure and the parametric design of virtual content as described above. Together they form the tracking volume-independent representation of real-walking experiences. The following four steps provide detailed description how Stuff-Haptics extends Scenograph's algorithm (please compare with the four steps of Scenograph's algorithm in chapter 5).

Stuff-Haptics takes in (1) the geometry of the physical props and the representation of a story (as a petri-net) and its content (as parametric placement rules). It then (2) maps that content onto the physical geometry. In the next (3) step, it handles the case of ill-fitting or limited physical prop set by splitting the virtual environments, or pruning the storyline to still instantiate parts of the story. Finally (4), Stuff-Haptics adapts and/or generates the virtual models and environment(s).

The main contributions of this chapter are in step 2, altering the mapping function with relevant information from the story, and in

step 3, handling the case of insufficient props. The main insight is that we can merge the two approaches, parametric models for spatial layouting and story descriptions for re-use of resources. We do not provide contributions in any other step, but will also go into detail about them, pointing to related work if necessary, to give the reader an overview on the general process.

Step 1: Take in geometry of physical props

The procedural generation of a virtual environment requires a scan of the physical environment containing the shape of the tracking volume and the shape of the physical props. Next to the shape of the props, their material properties such as heat emission or elasticity are useful, as they can be used later in step 2 to better match the virtual affordances. For example, the 'bears' porridge' fits better onto hot surfaces, 'papa bear's bed' fits better onto a hard surface.

Scanning physical environments is a well-researched area, as described in the related work section, chapter 2 ("scanning physical environments"). We thus extrapolate one step further and assume tracking is not an issue. We use *Vive trackables* [100] and pre-modelled meshes as a placeholder for this technology (Figure 40). We leave any discussion of the scanning process to the body of existing research.

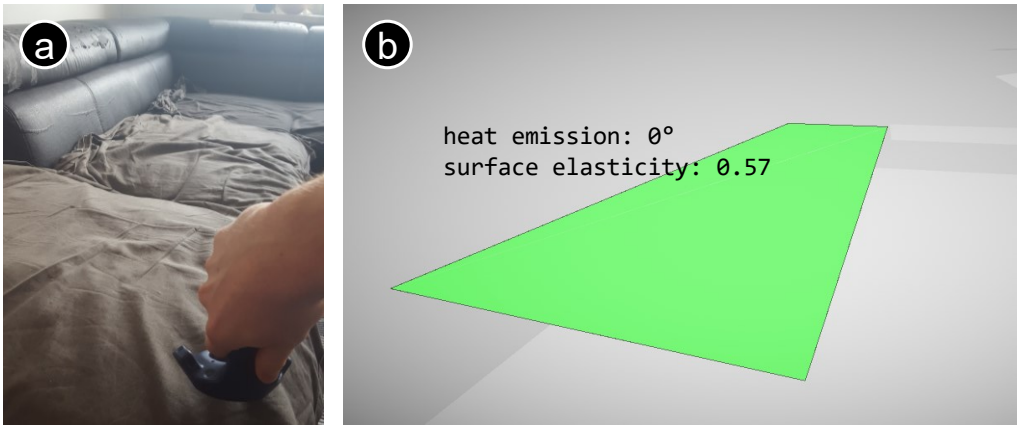

Figure 40: (a) We manually draw out the surfaces of the room. (b) We then annotate material properties, such as heat emission or elasticity.

Step 2: Map virtual objects onto geometry

Stuff-Haptics now automatically matches the virtual objects onto the physical geometry at hand.

In the following we show an internal view of Stuff-Haptics as it layouts the different elements (bowls of porridge, chairs, beds) of our example application onto the two different example environments; the user's kitchen (Figure 41) and the living room (Figure 42).

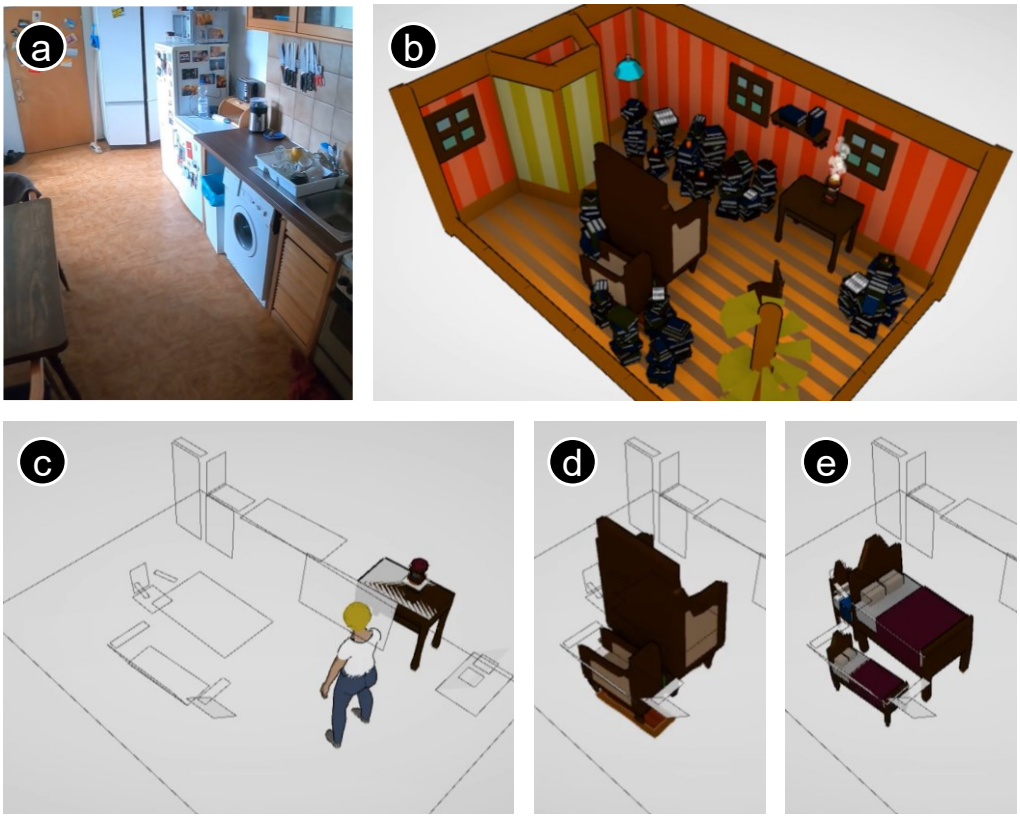

Figure 41: (a) Stuff-Haptics takes in the user's kitchen to generate the different story locations, such as (b) the 'bear's home'. (c) The annotated scan of the kitchen. Stuff-Haptics starts out by mapping the bowls of porridge on the countertop and water kettle. (d) The 'bears' chairs' cannot share the same prop, but (e) the 'bears' beds' in the 'bedroom' can re-use the table

Our implementation uses a constraint solver and parametric descriptions of virtual content, as suggested by the related work described above, but integrates the story structure in the process.

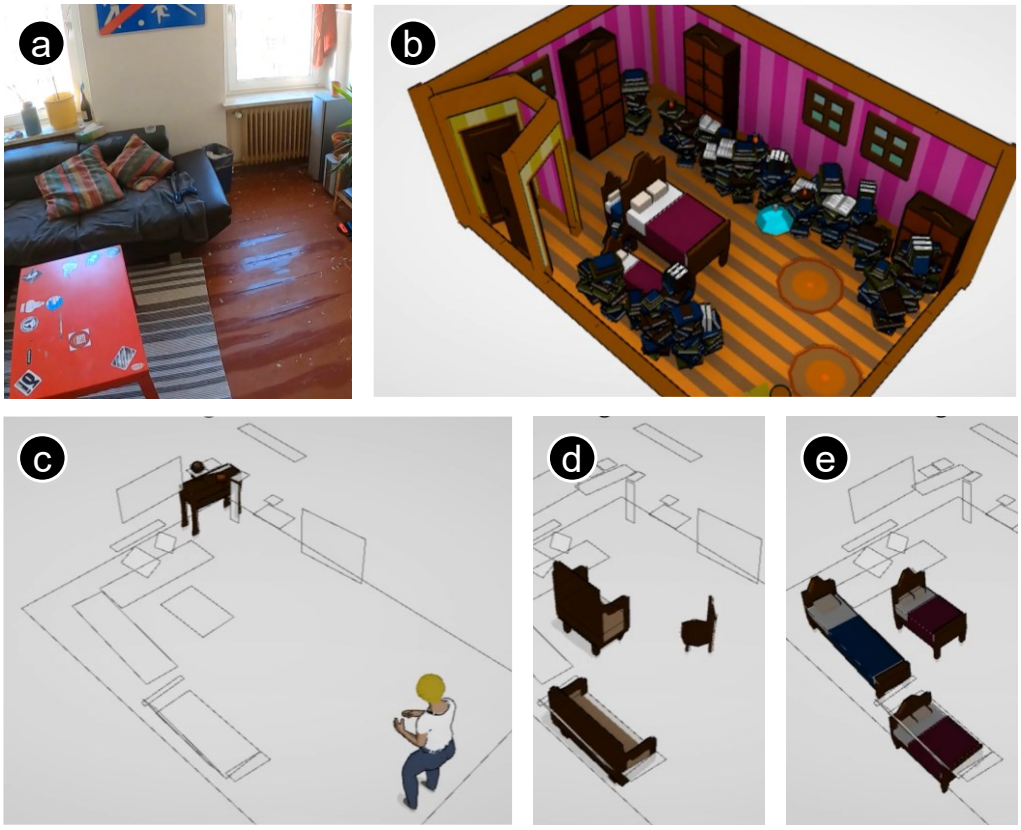

Figure 42: (a) For the living room, Stuff-Haptics re-generates the story locations, such as (b) the 'bear's bedroom' and re-layouts the elements, (c) the 'bowls of porridge', (d) 'chairs', and (e) 'beds'.

Our algorithm takes in an annotated scan of the physical environment and simplifies it by extracting only relevant (big enough) surfaces. In a pre-processing step for the mapping, for every virtual object, our algorithm first dismisses any surface which would not provide a fit. For example, the 'bears' beds' could not be placed on a vertical surface, such as a wall.

In each processing step Stuff-Haptics calculates a new fit for each virtual object using a stochastic method, the covariance matrix adaptation evolution strategy (CMA-ES [36]). Note that each virtual object set, such as the 'bear's porridge', is really comprised of a set of objects, such as three 'bowls of porridge' and a 'table'. The optimization runs separately for every virtual object set, but it runs simultaneously across sets to account for interdependencies between object sets. For

example ‘mama bear’ stands next to the ‘table’, and ‘papa bear’s bed’ must be larger than ‘mama bear’s bed’.

Algorithm 2: Map virtual objects to physical surfaces

Input: S – scan of the physical environment (as annotated surfaces), P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules)

Output: mapping M from P onto S

$V \leftarrow P.GetVirtualObjects()$
for each $v_i$in V **do**
    $M(v_i) \leftarrow RandomSolution(v_i, S)$
    $o_i \leftarrow CreateOptimizer(v_i, S)$
end for
$A \leftarrow AmountNodesChanged(P, V)$
for each o_i **where** $o_i.StillOptimizing()$ **do**
    $s_i \leftarrow o_i.SampleNewSolution(S)$
    $f \leftarrow Fitness(v_i, s_i) =>$
        $L \leftarrow \{\}$
        $R_i \leftarrow v_i.GetPlacementRules()$
        for each r in R_i **do**
            $eval \leftarrow r.Evaluate(s_i)$
            $v_r \leftarrow r.GetRelativeVirtualObject()$
            $relevance \leftarrow A(v_r)/\ Max(A)$
            $L.add(eval * relevance)$
        end for
        $Fitness(v_i, s_i) \leftarrow LogSum(L)$
        if $f > Fitness(v_i, M(v_i))$ **then**
            $M(v_i) \leftarrow s_i$
        end if
end for
for each v_i in V **do**
    if $Fitness(v_i, M(v_i)) < v_i.Thres(M(v_i))$ **then**
        $M(v_i) \leftarrow \{\}$
    end if
end for

Our algorithm uses the graph-based story structure to weight virtual objects more heavily in the global optimization, which are more relevant for story progression. Virtual objects are ranked by the amount of logical states they affect. For example, it is more relevant to find a good fit for 'little bear's bed' than for the other beds. The reason Stuff-Haptics finds this out is that the bed changes the 'sleepy' and 'bears back home' states. After lying down in the 'little' bed, Goldilocks falls asleep and finds herself waking up after the bears have returned home.

Step 3: Handle insufficient props and space

In practice, a lack of matching physical props prevents mixed reality experiences from running everywhere. If it is essential for the user to feel 'warm porridge' or sit on 'bears' chairs' and insufficient props are available, the story cannot take place. We here propose some solutions based on Stuff-Haptics prop-independent representation of the experience.

Solution 1: re-use props and space

We can apply our idea from before, splitting spatial nodes to re-use space, onto the re-use of props. Figure 43a-c shows how 'Goldilocks' requires three virtual beds, but it may be that fewer physical props exist that fit the parametric design, say just one sofa. That prop then can be assigned to all beds. This contradicts some placement rules, such as 'papa bear's bed' should be the hardest, but it maps all beds onto the environment. Stuff-Haptics automatically splits the 'bedroom' into three connecting ones by modifying the *Scenograph* algorithm from chapter 5 (page 68) as follows:

Algorithm 3: Re-use props and space

Input: P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules), M – mapping of P onto current physical environment

Output: scene splits based on P and M

$N \leftarrow GetSpatialNodes(P)$
for each n **in** N **do**
    $V \leftarrow P.GetVirtualObjectsOfNode(n)$
    $d \leftarrow P.GetDistanceMatrix(V)$
    for each v_a, v_b **in** V **with** $a \neq b$ **do** *
        $R_a \leftarrow v_a.GetPlacementRules()$
        for each r in R_a **do**
            if $v_b = r.RelativeVirtualObject()$ **then**
                $d(a,b) \leftarrow 0$
            end if
        end for
    end for
    $g \leftarrow GenerateDendrogram(d, V)$
    $C \leftarrow GetPossibleClustersSorted(g)$
    while $M.VirtualObjectsOverlap(V, P)$ **do** *
        $c \leftarrow C.Next()$
        $N_c \leftarrow n.SplitNode\ (c)$
        for each n_i in N_c **do**
            $V_i \leftarrow P.GetVirtualObjectsOfNode(n_i)$
            if $M.VirtualObjectsOverlap(V_i, P)$ **then**
                $P.ReplaceNode(n, N_c)$
            end if
        end for
    end while
end for
for each n in $GetSpatialNodes(P)$ **do**
    $GenerateScene(n)$
end for

* These are the main modifications to the original algorithm

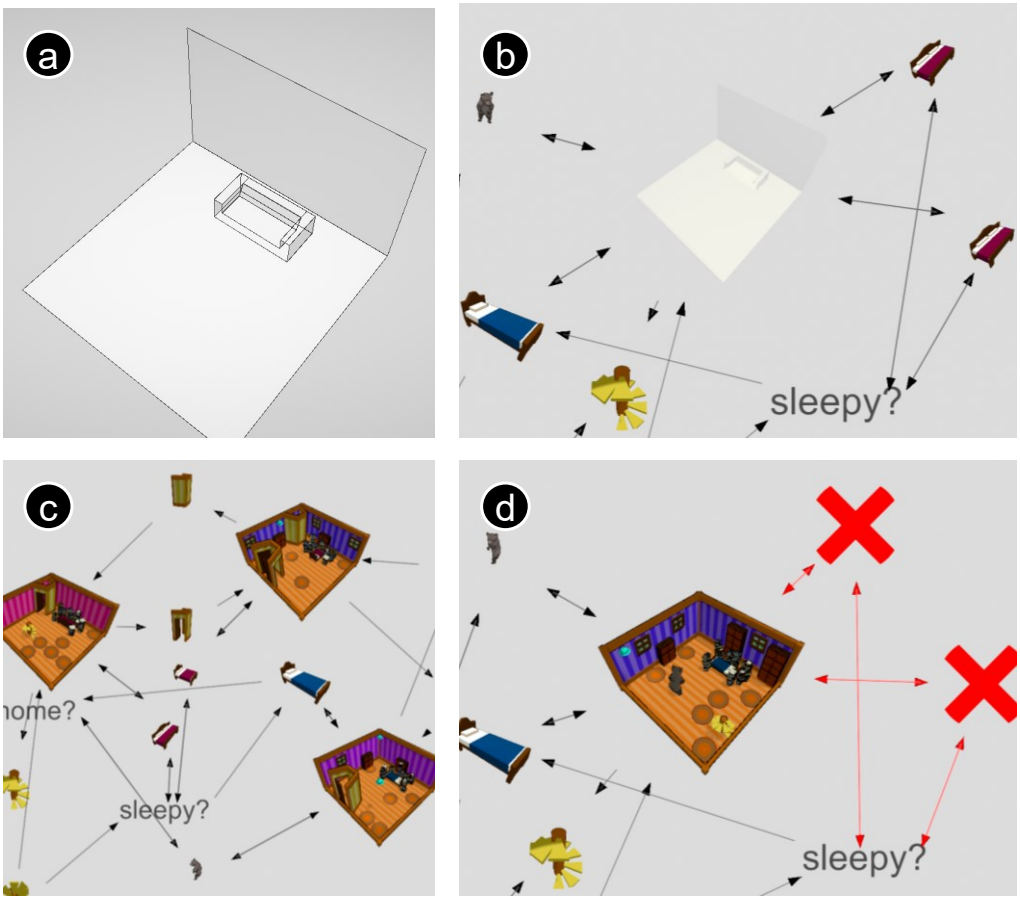

Figure 43: (a) This room contains just one sofa. (b) The 'upstairs bedroom' contains three 'bears' beds', not all of which will fit onto the sofa. (c) Solution 1: Stuff-Haptics splits the 'bedroom' into three connected locations and re-uses the sofa for all three 'beds'. (d) Solution 2: Stuff-Haptics prunes 'Mama bear's bed' and 'Papa bear's bed' out of the narrative so that our Goldilocks experience only contains the most important 'little bear's bed' for the user to lie down on.

Solution 2: delete virtual objects and prune the storyline

Creators might not want to split the virtual environment. Stuff-Haptics offers to automatically remove less relevant story objects to free up props and space for more relevant objects. This relevance is measured by the impact of the deletion on the story graph, similar to when we can only find a bad fit for a virtual object.

Figure 43d shows an example. Using 'little bear's bed' enables the state 'bears back home' and is more important than 'mama bear's bed' or 'papa bear's bed'. All beds find their best match in the sofa. Instead

of splitting the 'upstairs bedroom' into three, only 'little bear's bed' appears in the story.

Whenever Stuff-Haptics disables an object it avoids logical deadlocks by backtracking. In our example, using the 'stairs' to flee from the 'bedroom' requires the 'bears back home' state to be active, which happens by using 'little bear's bed'. Using the stairs is now also possible even when the bed cannot be mapped. By simply walking into the bedroom, the bears return home and the experience can continue.

Algorithm 4: Prune storyline

Input: P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules), M – mapping of P onto current physical environment

Output: pruned transitions in P

$N \leftarrow GetSpatialNodes(P)$
for each n **in** N **do**
    $V \leftarrow P.GetVirtualObjectsOfNode(n)$
    $A \leftarrow AmountNodesChanged(P, V)$
    while $M.VirtualObjectsOverlap(V, P)$ **do**
        $(v_a, v_b) \leftarrow M.GetFirstOverlap(V, P)$
        if $A(v_a) < A(v_b)$
            $P.RemoveTransition(v_a, n)$
        else
            $P.RemoveTransition(v_b, n)$
        end if
        $V \leftarrow P.GetVirtualObjectsOfNode(n)$
    end while
$GenerateScene(n)$
end

While this solution allows the experience to still run, some basic shaped props should exist to be able to instantiate enough relevant parts of the experience. For Goldilocks this is a horizontal surface of certain size for

the 'beds' and 'chairs' and a heat source for the 'porridge'. All the objects in the narrative structure have an importance for the story, which, if removed, decrease the overall fitness value. This overall value is also reduced by the valuations of each mapping. The software tool compares this total fitness value to a threshold set by the application to decide whether or not to run that application in the given physical environment.

Solution 3: efficient use

In chapter 4 we have already proposed several techniques to map one prop to multiple potential candidates. The case here matches the requirements for injecting unnoticeable runtime changes.

In a mixed approach those techniques can be applied at runtime in Step 2, the layouting process (page 87). In this case the virtual objects can be clustered around their physical counterparts, e.g. the virtual bowls of porridge are placed around the heat source.

Related work on this topic suggests to alter the one-to-one mapping of physical to virtual body motion, e.g. *Haptic Retargeting* [17], to match multiple objects onto only one prop. As discussed before, breaking this mapping leads to a potential decrease of immersion.

However, sometimes re-use can be achieved through design. By stacking the three bowls of porridge we distance them further from the heat source, which takes care of the requirement of three different temperatures. If more heat sources had been available, the bowls of porridge would have been able to use those as well.

Switching props – use story progression

Props might become unavailable at runtime, or new props come in, or users need to switch environments altogether. In these cases Stuff-Haptics recomputes the layout. The current story progression informs all relevant algorithms, i.e. tokens distribution in the petri-net. Figure 42c shows an example of this, where the user already interacted with the 'porridge'. The porridge now has a low importance in the layouting

process and could be mapped almost anywhere, as users are unlikely to feel the porridge again. Here Stuff-Haptics can make due with the somewhat less fitting solution of the window sill above the heater.

Step 4: Instantiate the experience

As the final step, Stuff-Haptics generates virtual scenery around the mapped virtual objects.

Abundance of props – obfuscate props

There are nearly always props that an experience has no use for. To prevent users from running into them, the application provides virtual objects for each scene that Stuff-Haptics automatically places to obfuscate the prop (Figure 44).

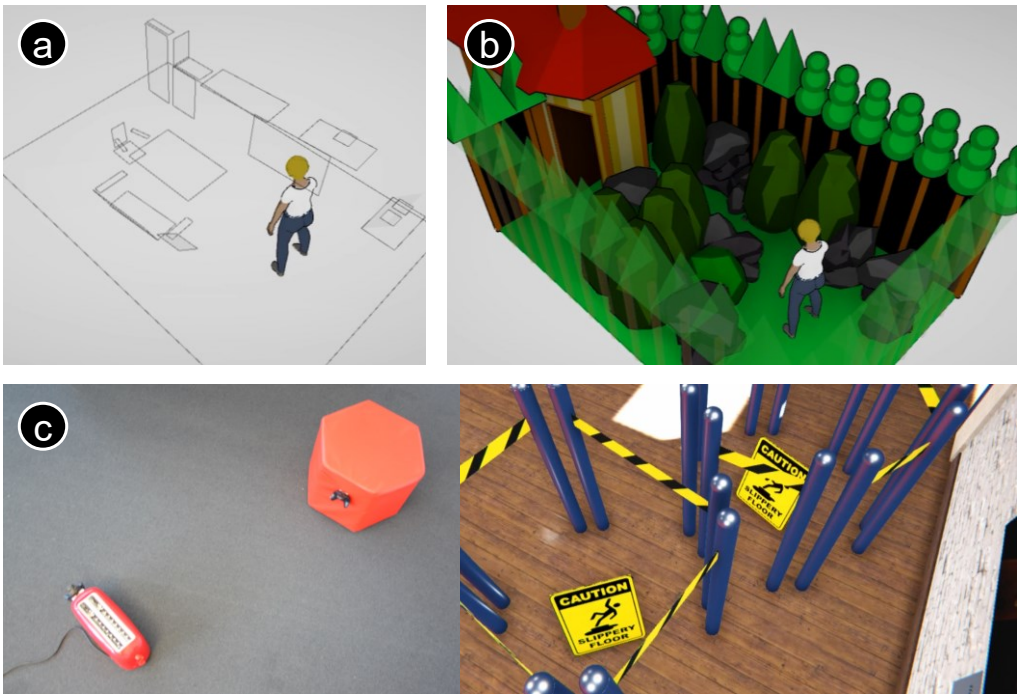

Figure 44: (a) The scan of the user's kitchen as annotated surfaces of props. (b) Goldilocks' 'dark forest' has no use for these props so it obfuscates them with bushes and rocks. (c) To cover these props our 'Mona Lisa Heist' experience uses warning tape and signs.

Different shapes of props – procedural modeling

Stuff-Haptics uses two approaches to build virtual models. In a top-down approach, we morph existing virtual content into the desired

shape using free-form-deformation. We found the top-down approach quite easy to model, but virtual models will not fit onto a lot of props. In a bottom-up approach, we assemble the virtual model from scratch. Figure 45 shows an example of a virtual staircase, designed with the bottom-up approach, that fits both our physical chairs and our physical sofa. The bottom-up approach, while harder to model for, affords a broader range of physical counterparts and is known as procedural modeling (e.g., [62]), here applied to mixed reality and passive haptics.

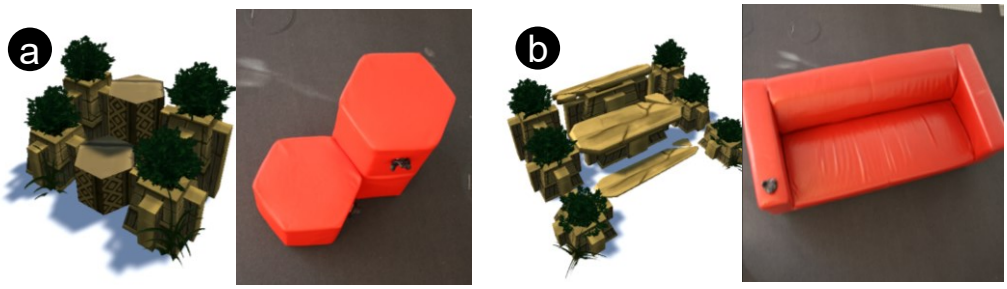

Figure 45: These virtual chairs fit on both (a) the chair and (b) the sofa.

Dangerous props – prevent touch via hardware

While Stuff-Haptics discourages users from touching certain objects, for example through obfuscation, users wearing head-mounted displays still tend to become unaware of their physical surroundings. Some users might still reach into the virtual fire, whereas they would never reach inside, for example, a toaster that emits that heat. Similar to VirtualSpace we can only provide fallback visualizations when users decide to ignore audiovisual guidance we provide through the experience.

6.4 Implementation

Stuff-Haptics is implemented in *C#* as a plug-in for *Unity3D*. We integrated the open source genetic algorithm CMA-ES [19] based on [36]. For assessing parametric design we use a 3D model database [46].

6.5 Technical evaluation, assessing parametric designs

Creators might not be able to intuit the correct parameters for spatial placement. We implemented an approach based on example models to

allow for quick evaluation of the design, inspired by related work such as *Project MARS* [61] and others [31]. We included a subset of 206 3D models from Lim et al.'s library of IKEA furniture [46, 54], removing redundant models, and imported them into Stuff-Haptics with simplified 3D meshes. By passing a virtual object through this database its parametric design can be accessed by the amount of furniture fitting to the virtual object. Figure 46 shows some examples. 78 pieces of furniture achieved the same or a better value for the *showcase* shown below, 120 pieces achieved the value of the *staircase*, only 4 achieved the value of the *pictures*, which may suggest further work on the design.

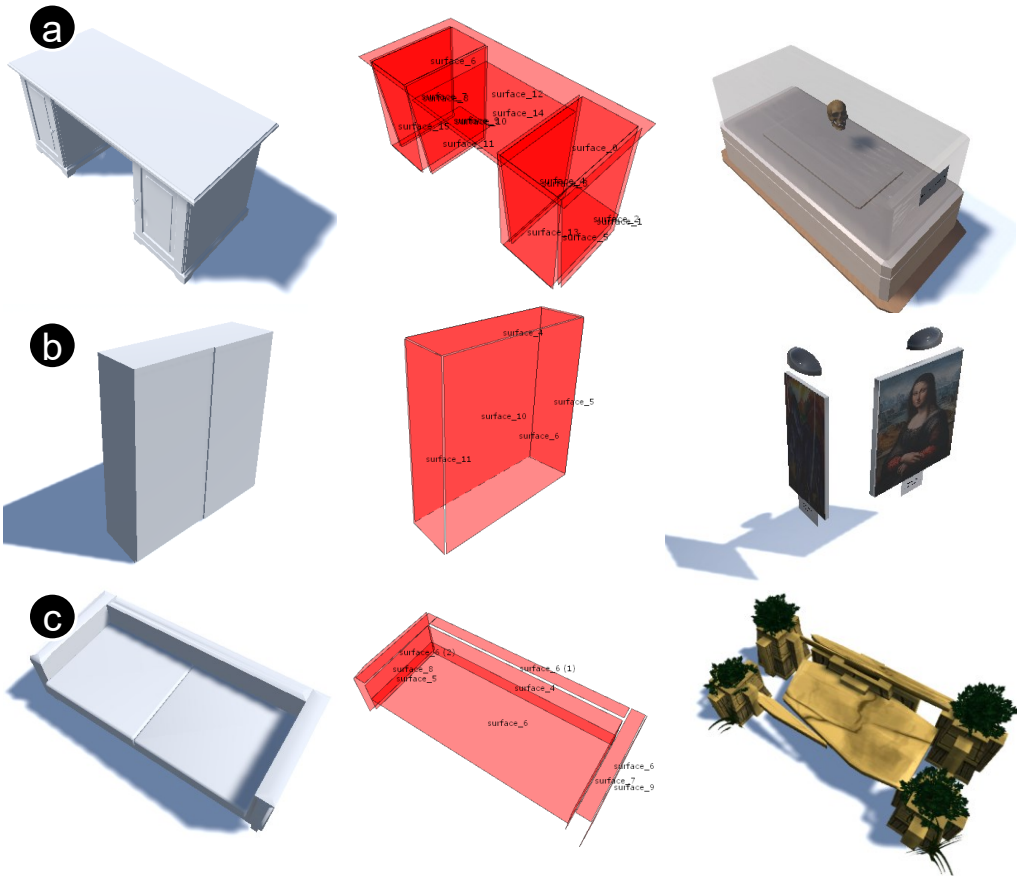

Figure 46: (a) The desk matches the virtual showcase. (b) The wardrobe matches the virtual pictures. (c) The sofa matches the virtual staircase.

6.6 User studies

We conducted two user studies to verify that Stuff-Haptics algorithm works.

User study 1: evaluating different environments

In the first study we tested Stuff-Haptics as a whole and ran 'Goldilocks' for five users in different rooms (kitchen, living room, bedrooms, storage room). We collected the qualitative feedback of participants to evaluate Stuff-Haptic's ability to effectively instantiate Goldilocks for different users' homes.

Apparatus

We tested our tool as described in the previous chapters to map 'Goldilocks' onto users' different physical props (couches, beds, tables, countertops). We used *solution 2: prune the storyline* for the case of limited props. We used a tether-less virtual reality setup (*VIVE* [100]).

Task and procedure

Participants experienced the whole 'Goldilocks' story and rated it afterwards.

Measurements

Participants rated the 'Goldilocks' experience on a 1-7 Likert scale (not at all – very much). *Fit*: "The experience fits well into my home/room." *Realism*: "The experience felt realistic." *Enjoyment*: "The overall experience is enjoyable."

Participants

We recruited five participants (3 male, 2 female, mean age 32.0 ± 2.3 sd). Four of the participants had previous experience with virtual reality headsets, one of them with passive haptics.

Results

Participants enjoyed Goldilocks (mean rating 6.4/7 ± 0.9 sd). They found the experience only somewhat realistic (mean rating 3.8/7 ± 0.8 sd), but found it to fit well into their home environment (mean rating 5.4/7 ± 0.9 sd). The experience successfully ran in all setups using different rooms and different props, such as tables, countertops, sofas, windowsills, or beds. Heat sources were an exposed lamp, a tea kettle, a water kettle, a candle in a jar. Some objects failed to map: the 'parent bears' failed three times, the 'bowls of porridge' once did not have a heat source.

Qualitative Feedback

For some objects, participants were positively surprised by the added haptics ("the bed felt stunning, it was really soft", 2 similar). Participants stated that Goldilocks "works very well" (2 similar). However, they found visuals to not always match to the haptics ("I could reach through the chair a bit", 2 similar) and realism to be "influenced by the optic and lack of sound" (1 similar). Some objects were not occluded ("something was in the way I could not see", 1 similar) and that occlusion objects were not diverse enough ("surprisingly many books", 1 similar).

User study 2: evaluating different experiences

We conducted an earlier user study in a controlled lab environment to show that Stuff-Haptics works for a broad range of virtual content, i.e., also for our experiences 'The Golden Key' and the 'Mona Lisa Heist'.

Interface conditions

We compared two conditions.

In *Stuff-Haptics*, we instantiated the virtual objects of our experiences 'The Golden Key' (staircase, fire, wind) and the 'Mona Lisa Heist' (pictures, showcases) onto a fixed prop set (sofa, walls, heat lamp, fan).

In *baseline*, we preserved the form of the generated virtual objects, but did not assign physical props to them.

Apparatus

We conducted the study in our 25m^2 lab using a tether-less virtual reality setup (*VIVE* [100]).

Task and procedure

Participants went through both experiences for each condition and rated each virtual object afterwards resulting in a total of 10 ratings per participant (2 conditions, 2 and respectively 3 virtual objects). The order of experiences and conditions was randomized.

Measurements

Participants rated virtual objects on a 1-7 Likert scale (not at all – very much). Realism: "This object felt realistic." Enjoyment: "This object is enjoyable."

Participants

We recruited 14 participants from our organization (4 female, 9 male, 1 not stated, mean age 24.4 ± 3.2 sd). 8 of the participants had previous experience with virtual reality headsets, 3 of them also with passive haptics.

Hypotheses

We hypothesized that *Stuff-Haptics* would provide higher ratings than the baseline condition for each virtual object for both realism (**H1**) and enjoyment (**H2**).

Results

S*tuff-Haptics* provided a higher mean rating than our *baseline* for both *realism* ($p < .001$, $t(13) = 6.94$, mean ratings 5.0/7 ± 2.2 sd vs. 3.2/7 ± 3.7 sd), as well as *enjoyment* ($p < .001$, $t(13) = 4.88$, mean ratings 5.3/7 ± 1.8 sd vs.

3.7/7 ± 3.8 sd) supporting both **H1** and **H2**. We found no differences between virtual objects of the two experiences.

Qualitative feedback

Participants found objects to mostly correlate with their expectation and were positively surprised when encountering haptic effects ("Hey! this feels really warm", "I can even sit here").

Discussion

As our main finding, we conclude that Stuff-Haptics is able to generate different virtual reality experiences with passive haptics in limited, uncurated physical environments, specifically users' homes. Our first study indicates that the same passive haptics experience ('Goldilocks') runs for different users' prop sets. Our second study shows that Stuff-Haptics can run different passive haptics experiences.

The studies have some major limitations. Our first study was conducted with only a handful of participants and shows preliminary results at best, while our second study, while showing that we could successfully use Stuff-Haptics to instantiate our experiences, mainly measured the effect of passive haptics.

We can thus only state that Stuff-Haptics can successfully instantiate experiences and can map multiple experiences to the same environment and the same experience to multiple environments. We cannot make claims on the external validity, i.e., the scaling to a broad bandwidth of applications.

6.7 Conclusion on space- and prop-independence

We presented Stuff-Haptics, a software tool that maintains a consistent storyline across different tracking volumes containing uncurated and limited sets of physical props. At its core, Stuff-Haptics uses a space- and prop-independent representation of a real-walking and passive haptic experience to generate its interconnected virtual world, re-using space and props. This representation combines the graph-based narrative

structure from Scenograph and the parametric design that describes virtual objects' spatial layout as well as material properties. Stuff-Haptics uses this representation to lay out the virtual objects onto an annotated scan of the available prop set. To account for limitations in the available prop set, Stuff-Haptics re-uses props or prunes the narrative structure where needed. Our technical evaluation shows that Stuff-Haptics design can be assessed. In our user studies, Stuff-Haptics both successfully ran the same experience for different users' prop sets and ran different passive haptics experiences on the same prop set. Stuff-Haptics may thus help creators to design passive haptics experiences independently from users' prop sets and provides a necessary step to bringing procedurally generated passive haptics experiences closer to users' homes.

With this project we seem to have now covered all relevant aspects for a full virtualization of physical space, so that real-walking experiences can take place anywhere. In the last chapter we will discuss the benefits and limitations of our work, and outline approaches for consecutive future work.

7 Conclusion and Discussion

In this chapter, we bring this work to its conclusion. We first summarize the work presented so far. We then zoom out to discuss the benefits of virtualization of physical space, checking whether our results hold up against our analogy of virtualization in computing history. We will close this thesis by focusing on open questions and challenges.

7.1 Summary

The main cost for real-walking in virtual reality is not its hardware but the physical space it demands. In this thesis we interpret physical space as a resource that we can virtualize and thus manage.

We propose the concurrent use of physical space to reduce the space demand per user. Applications with different users immersed in different experiences need to adhere to the API of our software system VirtualSpace to synchronously share the same space (chapter 3). For the concurrent use of physical *props*, applications need to follow the API of Mise-Unseen, a software tool that dynamically adapts virtual environments based on runtime demands, such as providing passive haptics to a certain virtual object (chapter 4). This concurrent use makes the use of resources more efficient, similar to a memory manager in an operating system where multiple programs have access to the same physical memory.

To render applications not only more cost-effective, but independent of specific physical resources, we have proposed a tracking-volume independent representation of the real-walking experience. The first part of the representation is a petri-net for representing the connectedness of those virtual objects through logical progression in an experience (chapter 5). Our software Scenograph uses hierarchical clustering on the petri-net to split virtual environments into multiple interconnected instances to then generate the virtual experience. The second part of this representation is the parametric description of virtual objects the experience contains (chapter 6). Our software Stuff-Haptics, which extends Scenograph, uses a constraint solver for spatial layouting of those virtual objects onto given geometry. The combination of automatic layouting and splitting of virtual scenery based on this representation allows our software to run complex experiences in various tracking volumes, like an operating system that is able to run programs on various machines.

7.2 Benefits of virtualizing physical space

For users virtualization of physical space means that, for any real-walking experience, their physical space can be shared with other people immersed in different virtual realities (VirtualSpace), their physical props can be multi-purposed (Mise Unseen), their physical space can have any size and shape (Scenograph) and contain arbitrary sets of physical props (Stuff-Haptics).

For creators virtualization of physical space means that they can express experiences with the same complexity as before, while not designing them with a fixed tracking volume in mind.

This work extends the field of procedural content for mixed reality. Procedural content for home and other un-curated environments has been proposed in *substitutional reality* [79] or *Oasis* [85]. Algorithms for spatial layouting have been proposed in *annexing reality* [39] or *Flare* [33]. We similarly believe that the only way real-walking can be achieved is

by generating the virtual world onto the physical one, as otherwise we need to break the immersive one-to-one motion mapping.

Making real-walking experiences independent of the user's physical space will have substantial commercial impact for virtual reality systems. Users have the benefit of using all their space, creators do not need to require users to have certain space and can tell more complex stories. Future virtual reality applications will run on a wide range of installations and will soar past current limitations by accessing additional virtual space, sharing resources, and building on re-useable high level descriptions of real-walking content.

We project virtualization of physical space to impact people's lives in the following ways:

Applications will be entertaining *and* productive

We believe that applications do not need to be productive to hold value. That being said, mixed reality applications of this nature can put user effort to a productive use, for example by applying it to training scenarios, or educational purposes.

We have developed a mixed reality application that holds another value, by having users perform physically demanding tasks as a byproduct of game play. In *Tower Pretense*, users virtually play a "tower defense"-style game while as a side product they physically carry moving boxes up a staircase, as shown in Figure 47.

Tower Pretense contains a strategic element in that players choose the order in which to carry up batteries. Rather than shooting monsters, players set up stationary weapons that shoot monsters. Each wave of monsters requires a different set of turrets to be activated, since each turret takes down only one type of monster. Players may choose to play tactically by getting a battery from the turret just below, deactivating it, and activating the next turret that eliminates the current wave of monsters, or strategically by planning ahead and positioning surplus batteries where they are likely to be needed next. Carrying n boxes up

m flights of stairs allows for $(mn)! * \prod_{k=0}^{m-1} (k!/(n+k)!)$ [34] strategies; for 8 boxes and 4 flights of stairs this allows for 1.5 trillion different strategies.

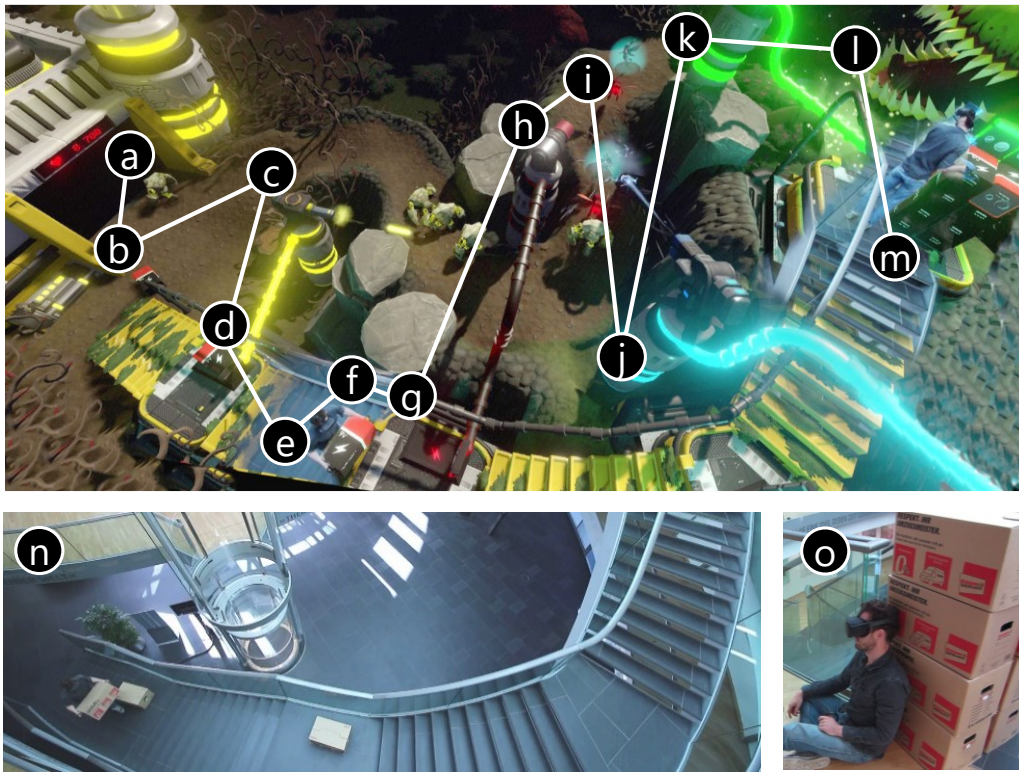

Figure 47: (a) Monsters attack (b) the player's home base and need to be stopped. (c) The turrets along their path can stop the monsters, as long as (d) they have batteries to power them. (e) The player thus (f) grabs batteries and (g) places them into the next turret's battery compartment. (h) The corresponding turret (here the red turret) is activated and (i) starts killing the corresponding red monsters. (j) The player activates more and more powerful turrets located further and further up the stairs. (k) Once the most powerful (green) turret is activated, it takes down the green boss monster. (m) This requires *all* batteries though, so that players ultimately find themselves carrying their batteries all the way up to the top in order to win the game. (n) Switching back to the physical world, we see that each battery was represented by a moving box as a passive haptics prop, which the player carried to the top of the staircase. (o) The moment players win the game, they have thus also moved their entire household.

Ahn's *Games with a Purpose* [1] already established the idea of gamers providing labor for a third party. We investigate how to apply this general idea of games with a purpose to mixed reality applications. Unlike gamification [65] or serious games [64], in which users find themselves scored and incentivized as in a game, users are actually

playing a game, often unaware of its purpose, such as generate training data for machine learning, as in *Foldit* [30]. Mixed reality games have a long tradition of leveraging elements of the physical surrounding to enhance users' virtual experiences, as summarized in the related work (chapter 2, page 27), or to enhance users themselves, for example through exergames [68]. We suggest that by virtualizing physical space the opposite direction is possible as well, leveraging elements of the *virtual* surrounding to enhance users' *physical* world.

Users will run experiences on public physical spaces

In the future mixed reality gaming will take place in public and other shared spaces [16] or even city-wide applications [105] – walking simulators which put storytelling front and center, like *Dear Esther* [21], are a current trend in the gaming industry. This trend might benefit from virtualizing physical space.

If applications will be put to a productive use, as stated above, and be able to run in public spaces, such applications will then also pursue objectives of public interest, such as cleaning up public parks.

Real-walking experiences will be created on the fly

Our work adds to an overarching trend currently happening in content development, the ever-shortening life-cycle and production time of virtual content. Development tools have been democratized, as seen in commodified game editors such as the Unity and Unreal Engines, but also in applications that are itself ecosystems for other applications, such as *Dreams* [24] or *Little Big Planet* [55]. Applications offer their content as a service, and let their users both consume and develop that content themselves through rapid application development and micropayments.

Especially mixed reality applications will benefit from this trend, as they require more design effort. Experiences will adapt or be generated onto given geometry more quickly. Either content creation is so quick it can be crowdsourced, generated just-in-time by users, or data-driven

AI's will outrun humans in the creation. Creators of small development teams are incentivized to join this trend, as discoverability in the market today is an issue which localized experiences to certain public places might help mitigate.

7.3 Open challenges

Further advances in procedurally generated mixed reality experiences are needed to get the attention of a majority of creators.

Designing with *all* tracking-volumes in mind, or: Finding a high-level language for real-walking experiences

As mentioned throughout this work, experiences are usually designed with a fixed tracking-volume in mind. Our solution is arguably a bit more difficult to design for, as it requires creators to design for *all* tracking-volumes in mind. To help creators intuit the correct parametric design, future research should try to support them in areas of procedural modeling, such as modeling by example [31].

The best way of designing with all tracking-volumes in mind is by having a design language that is non-spatial. This would be a more abstract, high-level language, quite akin to abstractions that have been presented to programmers after the development of concurrent programming. Atomic constructs would tackle the mismatch between concepts in programmer's head, and those expressed in code.

These abstractions could be supported by advances in automated storytelling. Automatic storytelling is a field of research, for example by creating dialogue [48], and has found its way into public awareness through art pieces [91], but cannot compete with human creativity, due to the admittedly complex design space. We project that these efforts will find prosperous use in story-driven real-walking applications that have a more limited design spaces, due to natural limitations of the physical space it occurs in. The physical setting itself will inform the story that is told, while the story informs the procedural content that is

generated onto the physical setting. We see these kind of convergences of opposing goals elsewhere, for example in SLAM, a technique to localize a user's position and orientation *and* its spatial frame of reference. Runtime changes through eye-tracking may also have implications for storytelling as advances in narrative are now possibly contingent on where one looks.

After finding higher levels of abstraction, generating a story in the future will be as simple as writing code like this:

Outlook: A high-level language for real-walking experiences

```
var story = new Story();
story.underlying_theme = "sense of completion";
story.people = {"mentor", "student", "team"}
story.setting = "institute";
story.progression = "classic three act structure";
var scan = GetSemanticScan();
story.GenerateWorld(scan);
story.Start() ;
```

AVOIDING OVER-CONSTRAINED REAL-WALKING EXPERIENCES

Our systems cannot improve all real-walking scenarios. Our systems use the degrees of freedom that an experience provides. For example, it may not matter to the Goldilocks designer where the 'chairs' are in relation to the 'bowls of porridge', and Scenograph can place them in different scenes. However, if the designer specifies that each chair must be next to its porridge, then the system cannot optimize – the problem is over-constrained. This holds true for any experience with strong spatial constraints, say in a soccer simulation where the goal must be in a fixed relation to the other goal or to the corner flags. In VirtualSpace, if multiple applications must have a certain spot in the tracking volume, there is no solution. Our systems buy their advantages through a certain

flexibility on the side of their applications. Over-constrained problems are more of a natural limitation than an open challenge, however, future work could address how engaging real-walking experiences with a solvable amount of constraints can be formulated.

Ensuring user safety

We acknowledge that our systems might pose a certain physical risk. In the user study of VirtualSpace a total of seven collisions occurred. Related work [5, 9] achieved a reduction of collisions of over 58% in a simulated framework with much lower user density – although we have not conducted a collision baseline due to safety risks, we assume this compares well, but agree that no collisions are more desirable. In Stuff-Haptics' user study, participants criticized situations in which they did not trust the system's placement of virtual objects due to tracking irregularities and mapping that places virtual objects more than 2-3cm off their designated geometry.

We can argue that a certain risk may contribute to positive experiences for some participants, as "too close a distance" between users can be an intriguing game element [69]. However, as a general rule a lack of trust impacts enjoyment.

Having users with different mindsets or player types [7] might pose an additional challenge, as explorative and/or social types might deliberately walk into unintended areas, which happened during all of our studies.

Another challenge is to ensure that users not only follow, but notice and comprehend a given visual incentive. Research on user visualizations [52, 76] in shared space addresses this issue only in part.

Reducing setup cost

While Stuff-Haptics enables the use of various props, certainly not all passive haptics environments will provide enough props for every single experience. Of course, this is the exact problem Stuff-Haptics

addresses by providing different means (re-using props, altering storyline) to gracefully degrade the experience and preserve designers' intent as closely as possible despite physical shortcomings. However, some props might need to be turned on by the user, such as the water kettle, while other props might stand in unfavorable positions. Figure 38c (page 82) shows one example where the turned sofa provides a better fit for the virtual showcase. One direction of future work could support users in spending less time setting up props. Another approach is to rely on smart objects [26]. Another interesting research direction would be to integrate this setup into the same or a different application and make it a byproduct of the activity itself.

Semantic scanning and obtaining material properties

Our systems require a matching virtual twin for each prop. As Stuff-Haptics considers the shape of props as well as their material properties, this virtual twin ideally also describes material properties (such as heat emission, softness, etc.). Participants' individual remarks on material mismatches underline the necessity of fitting virtual objects correctly onto those material properties.

Research has suggested approaches for obtaining room scans, e.g. using point clouds [85], which can be enhanced with shape retrieval [54] or semantic parsing [3] to obtain material properties, or using smart objects so that props provide the information themselves [26]. Future work could bind more complex machinery into the haptic experience, such as elevators or moving stairways. Similar to drivers that describe hardware functionality, a networked or on-prop solution will let users re-purpose those props at home or in public spaces.

Merging procedural mixed reality with other locomotion techniques and devices

We think our solution takes pre-existing ones a step further. Virtual locomotion techniques, such as redirected walking [74] still require

specific amounts of space to work. Locomotion devices, such as treadmills, small [20] or room-scale [82], require costly hardware. Procedural mixed reality, especially in public spaces, is the only way one can experience limitless real-walking and passive haptics in a cost-efficient manner.

One challenge for procedural content is dealing with virtual objects that are supposed to be somewhat dynamic. Static objects, such as the 'little bear's bed' we can generate onto static geometry. Dynamic objects are more difficult – 'little bear's chair' should break, and 'little bear's porridge' should be eaten, according to the story. Rendering haptic effects using real-world objects will not always generate a solution.

To avoid this problem, procedural content can be used in conjunction with synthetic solutions. Experiences then use procedural content to generate the virtual world and its static objects, and use haptic rendering devices to support dynamic objects and haptic effects, especially devices that potentially can be used in mobile contexts as mixed reality will take in public spaces. Examples of these devices are based, for example, on electric muscle stimulation [57], which derive energy for actuation from the user's body, or on servo-motors [51], that can be word on the body. On the other hand, these haptic rendering devices fail reliably when rendering grounded, static objects. The reason is that a machine that actuates needs to be on par with what it simulates. A wall is never truly rigid [57] if the device is not grounded, so that opportunistic use of physical props seems to point into the more feasible direction. Again, both approaches have their unique sets of upsides, so that we project a hybrid model to emerge – synthetic generation of haptic effects through hardware, opportunistic use of physical space and static props through virtualization of physical space.

Other challenges

Solutions to all aforementioned challenges are needed to avoid another virtual reality winter, as happened after its invention in the 60's by Ivan

Sutherland. Further advances include developments in display technologies to include a versatile optical focus of lenses and waveguides to make rendering visual effects on the body more perceptible and thus haptic effects more meaningful. Research also needs to address the design challenge of shared literacy of mixed reality applications, so that users can intuit which objects to touch. Extending the concept of virtualization to remote collaboration scenarios is a challenge to enable 'co-walking' over a distance.

7.4 Final remarks

We set out to solve real-walking for virtual reality by pushing the idea of procedural content for mixed reality in uncurated and arbitrary physical spaces. We achieved this by applying one of computer science's oldest tools, virtualization, to physical space. This enabled us to run complex real-walking experiences anywhere.

We hope to advance the way we tell stories to each other. Any medium tends to govern the story told with it; books tell stories through writing, graphic novels tell stories visually, movies add motion, games include users' choices and abilities. Once a medium reaches the ability to tell stories, one of our primal instincts, we use it. By enabling storytelling in procedurally generated mixed reality we hope to have laid some groundwork for mixed reality to reach users' homes.

8 References

[1] Luis von Ahn. 2006. Games with a Purpose. Computer 39, 6: 92–94. https://doi.org/10.1109/MC.2006.196

[2] Alglib, cross-platform numerical analysis and data processing library. Retrieved January 15, 2021 from http://www.alglib.net/

[3] Iro Armeni, Ozan Sener, Amir Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). 1534-1543. 10.1109/CVPR.2016.170.

[4] Jatin Arora, Aryan Saini, Nirmita Mehra, Varnit Jain, Shwetank Shrey, and Aman Parnami. 2019. VirtualBricks: Exploring a Scalable, Modular Toolkit for Enabling Physical Manipulation in VR. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Paper 56, 1–12. https://doi.org/10.1145/3290605.3300286

[5] Mahdi Azmandian, Timofey Grechkin, and Evan Suma Rosenberg. An evaluation of strategies for two-user redirected walking in shared physical spaces. In *Virtual Reality (VR)*, Los Angeles, CA, pp. 91-98. https://doi.org/10.1109/VR.2017.7892235

[6] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. 2016. Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 1968-1979. https://doi.org /10.1145/2858036.2858226

[7] Richard Bartle. 1996. Hearts, Clubs, Diamonds, Spades: Players who suit MUDs. *Journal of MUD research*. 1(1), 19-58.

[8] Blaine A. Bell and Steven K. Feiner. 2000. Dynamic space management for user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (UIST '00), 239–248. https://doi.org/10.1145/354401.354790

[9] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In Robotics Research - The 14th International Symposium ISRR, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, May 2011, pp. 3-19. https://doi.org/10.1007/978-3-642-19457-3_1

[10] Mark Bernstein. 2002. Storyspace 1. In Proceedings of the thirteenth ACM conference on Hypertext and hypermedia (HYPERTEXT '02). Association for Computing Machinery, New York, NY, USA, 172–181. https://doi.org/10.1145/513338.513383

[11] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. 2016. Point & Teleport Locomotion Technique for Virtual Reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (CHI PLAY '16), 205-216. https://doi.org/10.1145/2967934.2968105

[12] Haiwei Chen, Samantha Chen, and Evan Suma Rosenberg. 2018. Redirected Walking Strategies in Irregularly Shaped and Dynamic Physical Environments. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces* (VR '18). *Workshop on Everyday Virtual Reality*.

[13] Lung-Pan Cheng, Li Chang, Sebastian Marwecki, and Patrick Baudisch. 2018. iTurk: Turning Passive Haptics into Active Haptics by Making Users Reconfigure Props in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, Paper 89, 10 pages. https://doi.org/10.1145/3173574.3173663

[14] Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. 2017. Mutual Human Actuation. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17). Association for Computing Machinery, New York, NY, USA, 797–805. https://doi.org/10.1145/3126594.3126667

[15] Lung-Pan Cheng, Thijs Roumen, Hannes Rantzsch, Sven Köhler, Patrick Schmidt, Robert Kovacs, Johannes Jasper, Jonas Kemper, and Patrick Baudisch. 2015. TurkDeck: Physical Virtual Reality Based on People. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15). Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/2807442.2807463

[16] Lung Pan Cheng, Eyal Ofek, Christian. Holz and Andrew. D. Wilson. 2019. VRoamer: Generating On-The-Fly VR Experiences While Walking inside Large, Unknown Real-World Building Environments. In *IEEE Conference on Virtual Reality and 3D User Interfaces* (VR), Osaka, Japan, 2019, pp. 359-366.

[17] Lung-Pan Cheng, Eyal Ofek, Christian Holz, Hrvoje Benko, and Andrew D. Wilson. 2017. Sparse Haptic Proxy: Touch Feedback in Virtual Environments Using a General Passive Prop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 3718-3728. https://doi.org/10.1145/3025453.3025753

[18] Clipper, Open source freeware library. Retrieved January 15, 2021 from http://angusj.com/delphi/clipper.php

[19] CMA Evolution Strategy. Retrieved January 15, 2021 from http://cma.gforge.inria.fr/

[20] Rudolph P. Darken, William R. Cockayne, and David Carmein. 1997. The omni-directional treadmill: a locomotion device for virtual worlds. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (UIST '97), 213-221. http://doi.org/10.1145/263407.263550

[21] Dear Esther. Retrieved January 15, 2021 https://www.thechineseroom.co.uk/games/dear-esther

[22] Zhi-Chao Dong, Xiao-Ming Fu, Chi Zhang, Kang Wu, and Ligang Liu. 2017. Smooth assembled mappings for large-scale real walking. *ACM Trans. Graph.* 36, 6, Article 211 (November 2017), 13 pages. https://doi.org/10.1145/3130800.3130893

[23] Doomba, from Rich Whitehouse. Retrieved January 15, 2021 from http://www.richwhitehouse.com/index.php?postid=72

[24] Dreams. Retrieved January 15, 2021 from https://www.mediamolecule.com/ games/dreams

[25] Dreamscape. Retrieved January 15, 2021 from http://www.dreamscapeimmersive.com/index.html

[26] Benjamin Eckstein, Eva Krapp, Anne Elsässer, and Birgit Lugrin. 2019. Smart Substitutional Reality: Integrating the Smart Home into Virtual Reality. *Entertainment Computing*. 100306. 10.1016/j.entcom.2019.100306.

[27] Barrett Ens, Eyal Ofek, Neil Bruce, and Pourang Irani. 2015. Spatial Constancy of Surface-Embedded Layouts across Multiple Environments. In Proceedings of the 3rd ACM Symposium on Spatial User Interaction (SUI '15). Association for Computing Machinery, New York, NY, USA, 65–68. https://doi.org/10.1145/2788940.2788954

[28] Martin Feick, Scott Bateman, Anthony Tang, André Miede, and Nicolai Marquardt. 2020. TanGi: Tangible Proxies for Embodied Object Exploration and Manipulation in Virtual Reality. arXiv preprint arXiv:2001.03021

[29] Fragments, AR Game. Retrieved January 15, 2021, from https://www. asobostudio. com/games/fragments

[30] Foldit | Solve Puzzles for Science. Retrieved January 15, 2021 from https://fold.it/portal/

[31] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by example. In ACM SIGGRAPH 2004 Papers (SIGGRAPH '04). Association for Computing Machinery, New York, NY, USA, 652–663. https://doi.org/10.1145/1186562.1015775

[32] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2007. Automatically generating user interfaces adapted to users' motor and vision capabilities. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (UIST '07). ACM, New York, NY, USA, 231-240. https://doi.org/10.1145/1294211.1294253

[33] Ran Gal, Lior Shapira, Eyal Ofek and Pushmeet Kohli. 2014. FLARE: Fast layout for augmented reality ap-plications. In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality* (ISMAR '14) 207--212. http://doi.org/10.1109/ISMAR.2014.6948429

[34] K. Gorska and K. A. Penson. 2013. Multidimensional Catalan and related numbers as Hausdorff moments. Retrieved January 15, 2021 from http://arxiv.org/abs/1304.6008

[35] Gurobi, Linear Solver, Academic License. Retrieved January 15, 2021 from http://www.gurobi.com

[36] Nikolaus Hansen. 2006. The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larrañga, I. Inza and E. Bengoetxea (eds.).

Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75-102, Springer.

[37] Steven J. Henderson and Steven Feiner. 2008. Opportunistic controls: leveraging natural affordances as tangible user interfaces for augmented reality. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (VRST '08). ACM, New York, NY, USA, 211-218. http://dx.doi.org/10.1145/1450579.1450625

[38] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1, Article 1 (February 2013), 22 pages. http://dx.doi.org/10.1145/2422956.2422957

[39] Anuruddha Hettiarachchi and Daniel Wigdor. 2016. Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 1957-1967. http://dx.doi.org/10.1145/2858036.2858134

[40] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. Passive real-world interface props for neurosurgical visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '94), Beth Adelson, Susan Dumais, and Judith Olson (Eds.). ACM, New York, NY, USA, 452-458. http://dx.doi.org/10.1145/191666.191821

[41] Christian Hirt, Markus Zank, and Andreas Kunz. 2018. Preliminary Environment Mapping for Redirected Walking. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces* (VR '18). IEEE. https://doi.org/10.3929/ethz-b-000253659

[42] H. G. Hoffman. 1998. Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments. In Proceedings of the Virtual Reality Annual International Symposium (VRAIS '98). IEEE Computer Society, USA, 59.

[43] Jeannette E. Holm. 2012. Collision prediction and prevention in a simultaneous multi-user immersive virtual environment. PhD diss., Miami University. Retrieved January 15, 2021 from https://etd.ohiolink.edu/

[44] Brent Edward Insko. 2001. Passive Haptics Significantly Enhances Virtual Environments. Ph.D. Dissertation. The University of North Carolina at Chapel Hill. Advisor(s) Frederick P. Brooks, Jr.. AAI3007820.

[45] Victoria Interrante, Brian Ries and Lee Anderson. 2007. Seven League Boots: A New Metaphor for Augmented Locomotion through Moderately Large Scale Immersive Virtual Environments. In *Symposium on 3D User Interfaces*, Charlotte, NC, 2007, pp. https://doi.org/10.1109/3DUI.2007.340791

[46] IKEA database. Retrieved January 15, 2021 from http://ikea.csail.mit.edu/

[47] Hans-Christian Jetter, Harald Reiterer, and Florian Geyer. 2014. Blended Interaction: understanding natural human---computer interaction in post-WIMP interactive spaces. *Personal Ubiquitous Comput.* 18, 5 (June 2014), 1139–1158. https://doi.org/10.1007/s00779-013-0725-4

[48] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. arXiv preprint arXiv:1606.01541 (2016).

[49] Luv Kohli. 2013. *Redirected Touching*. Ph.D. Dissertation. University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. Advisor(s) Frederick P. Brooks, Jr.. AAI3562754.

[50] Robert J. Kosinksi. 2013. *A Literature Review on Reaction Time.* Clemson University.

[51] Robert Kovacs, Eyal Ofek, Mar Gonzalez Franco, Alexa Fay Siu, Sebastian Marwecki, Christian Holz, and Mike Sinclair. 2020. Haptic PIVOT: On-Demand Handhelds in VR. In *Proceedings of the 33rd Annual*

ACM Symposium on User Interface Software and Technology (*UIST '20*). Association for Computing Machinery, New York, NY, USA, 1046–1059. DOI:https://doi.org/10.1145/3379337.3415854

[52] Jérémy Lacoche, Nico Pallamin, Thomas Boggini, and Jérôme Royan. 2017. Collaborators awareness for user cohabitation in co-located collaborative virtual environments. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology* (VRST '17). ACM, New York, NY, USA, Article 15, 9 pages. https://doi.org/10.1145/3139131.3139142

[53] Eike Langbehn, Paul Lubos, and Frank Steinicke. 2018. Redirected Spaces: Going Beyond Borders. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces* (VR '18). Demo.

[54] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. 2013. Parsing IKEA Objects: Fine Pose Estimation. In *Proceedings of the 2013 IEEE International Conference on Computer Vision* (ICCV '13). IEEE Computer Society, Washington, DC, USA, 2992-2999. http://dx.doi.org/10.1109/ICCV.2013.372

[55] Little Big Planet. Retrieved January 15, 2021 https://www.mediamolecule.com/games/littlebigplanet

[56] Locomotion Vault. Retrieved January 15, 2021 from https://locomotionvault. github.io/

[57] Pedro Lopes, Sijing You, Lung-Pan Cheng, Sebastian Marwecki, and Patrick Baudisch. 2017. Providing Haptics to Walls & Heavy Objects in Virtual Reality by Means of Electrical Muscle Stimulation. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1471–1482. DOI:https://doi.org/10.1145/3025453.3025600

[58] Kok-Lim Low, Greg Welch, Anselmo Lastra, and Henry Fuchs. 2001. Life-sized projector-based dioramas. In *Proceedings of the ACM*

symposium on Virtual reality software and technology (VRST '01). ACM, New York, NY, USA, 93-101. http://dx.doi.org/10.1145/505008.505026

[59] Chuan-en Lin, Ta Ying Cheng, Xiaojuan Ma. 2020. Architect: Building interactive virtual experiences from physical affordances by bringing human-in-the-loop. In *Conference on Human Factors in Computing Systems* (CHI). New York, NY, USA: ACM. doi:10.1145/2839462.283948

[60] Amaury Louarn, Marc Christie, and Fabrice Lamarche. 2018. Automated staging for virtual cinematography. In Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games (MIG '18). Association for Computing Machinery, New York, NY, USA, Article 4, 1–10. https://doi.org/10.1145/3274247. 3274500

[61] Project Mars, Unity Labs. Retrieved January 15, 2021 from https://unity.com/ unity/features/mars

[62] Paul Merrell, and Dinesh Manocha. 2010. Model Synthesis: A General Procedural Modeling Algorithm. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 715-728, June 2011. doi: 10.1109/TVCG.2010.112

[63] David E. Millard, Charlie Hargood, Michael O. Jewell, and Mark J. Weal. 2013. Canyons, deltas and plains: towards a unified sculptural model of location-based hypertext. In Proceedings of the 24th ACM Conference on Hypertext and Social Media (HT '13). Association for Computing Machinery, New York, NY, USA, 109–118. https://doi.org/10.1145/ 2481492.2481504

[64] Ho Ming Lau, Johannes H. Smit, Theresa M. Fleming, and Heleen Riper. 2017. Serious Games for Mental Health: Are They Accessible, Feasible, and Effective? A Systematic Review and Meta-analysis. Frontiers in Psychiatry 7: 209. https://doi.org/10.3389/fpsyt.2016.00209

[65] Alberto Mora, Daniel Riera, Carina Gonzalez, and Joan Arnedo-Moreno. 2015. A Literature Review of Gamification Design Frameworks. In 2015 7th International Conference on Games and Virtual Worlds for Serious

Applications (VS-Games), 1–8. https://doi.org/10.1109/VS-GAMES.2015.7295760

[66] Fausto Mourato, Fernando Birra, and Manuel Próspero Dos Santos. 2013. The Challenge of Automatic Level Generation for Platform Videogames Based on Stories and Quests. In *Proceedings of the 10th International Conference on Advances in Computer Entertainment - Volume 8253* (ACE 2013), Dennis Reidsma, Haruhiro Katayose, and Anton Nijholt (Eds.), Vol. 8253. Springer-Verlag New York, Inc., New York, NY, USA, 332-343. http://dx.doi.org/10.1007/978-3-319-03161-3_24

[67] ZuneBuggy MSR TechFest Demo 2007 by Andrew D. Wilson. Retrieved January 15, 2021 from https://channel9.msdn.com/Blogs/Rory/Microsoft-Research-TechFest-XNA-a-depth-sensing-camera-an-LCD-projector-and-some-genius

[68] Florian Mueller, Rohit Ashok Khot, Kathrin Gerling, and Regan Mandryk. 2016. Exertion Games. *Foundations and Trends® in Human–Computer Interaction* 10, 1: 1–86. https://doi.org/10.1561/1100000041

[69] Florian Mueller, Sophie Stellmach, Saul Greenberg, Andreas Dippon, Susanne Boll, Jayden Garner, Rohit Khot, Amani Naseem, and David Altimira. 2014. Proxemics play: understanding proxemics for designing digital play experiences. In Proceedings of the 2014 conference on Designing interactive systems (DIS '14), 533-542. https://doi.org/10.1145/2598510.2598532

[70] G. Norman: Likert Scales, level of measurements, and the “laws” of statistics. In *Advances in Health Sciences Education* 15, 5, 625-632, 2010. https://dx.doi.org/10.1007/s10459-010-9222-y

[71] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. 2002. Generating remote control interfaces for complex appliances. In *Proceedings of the 15th annual ACM symposium on User interface software*

and technology (UIST '02). ACM, New York, NY, USA, 161-170. DOI: https://doi.org/10.1145/571985.572008

[72] Oculus System. Retrieved January 15, 2021 from https://www.oculus.com/

[73] Protocol Buffer, Contract Based Serializer. Retrieved January 15, 2021 from https://github.com/mgravell/protobuf-net

[74] Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton. 2001. Redirected walking. In *Proceedings of EUROGRAPHICS* 9, 105-106.

[75] REDkit. Retrieved January 15, 2021 from https://redkitwiki.cdprojektred.com/Welcome+to+the+REDkit+Wiki

[76] Anthony Scavarelli and Robert J. Teather. 2017. VR Collide! Comparing Collision-Avoidance Methods Between Co-located Virtual Reality Users. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '17), 2915-2921. https://doi.org/10.1145/3027063.3053180

[77] Scenograph, Online Repository. https://github.com/sebastianmarwecki/Scenograph

[78] Lior Shapira and Daniel Freedman. 2016. Reality Skins: Creating Immersive and Tactile Virtual Environments. In *IEEE International Symposium on Mixed and Augmented Reality* (ISMAR), Merida, 2016, pp. 115-124.

[79] Adalberto L. Simeone, Eduardo Velloso, and Hans Gellersen. 2015. Substitutional Reality: Using the Physical Environment to Design Virtual Reality Experiences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3307-3316. http://dx.doi.org/10.1145/2702123.2702389

[80] Keng Hua Sing and Wei Xie. 2016. Garden: A Mixed Reality Experience Combining Virtual Reality and 3D Reconstruction. In *Proceedings of the*

2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16), 180-183. https://doi.org/10.1145/2851581.2890370

[81] Harvey Smith, Matthias Worch. What Happened Here? Environmental Storytelling, Online Lecture. GDC Vault. Retrieved January 15, 2021, from https://www.gdcvault.com/play/1012647/ What-Happened-Here-Environmental

[82] J. L. Souman, P. Robuffo Giordano, M. Schwaiger, I. Frissen, T. Thümmel, H. Ulbrich, A. De Luca, H. H. Bülthoff, and M. O. Ernst. 2008. CyberWalk: Enabling unconstrained omnidirectional walking through virtual environments. *ACM Trans. Appl. Percept.* 8, 4, Article 25 (November 2011), 22 pages. https://doi.org/10.1145/2043603. 2043607

[83] Source 2. Retrieved January 15, 2021 from https://developer. valvesoftware.com/wiki/Source_2

[84] Robert Southey. 1839. The Story of The Three Bears. In *Fairy Tales and Other Traditional Stories*, Lit2Go Edition. Accessed April 1, 2018 from http://etc.usf.edu/lit2go/68/fairy-tales-and-other-traditional-stories/5105/the-three-bears/.

[85] Misha Sra, Sergio Garrido-Jurado, Chris Schmandt, and Pattie Maes. 2016. Procedurally generated virtual reality from 3D reconstructed physical space. In *Proceedings of the 22nd ACM Conference on Virtal Reality Software and Technology* (VRST '16), 191-200. https://doi.org/10.1145/ 2993369.2993372

[86] Misha Sra. 2016. Asymmetric Design Approach and Collision Avoidance Techniques For Room-scale Multiplayer Virtual Reality. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16 Adjunct), 29–32. https://doi.org/10.1145/ 2984751.2984788

[87] Steam VR blog post from 10 Nov, 2017. Retrieved January 15, 2021 from https://steamcommunity.com/app/358720/discussions/0/3505325361035 14259/?ctp=3

[88] Evan. A. Suma, Seth Clark, David Krum, Samantha Finkelstein, Mark Bolas and Zachary Warte. 2011. Leveraging change blindness for redirection in virtual environments. In *2011 IEEE Virtual Reality Conference*, Singapore, 2011, pp. 159-166. https://doi:10.1109/VR.2011.5759455

[89] Evan A. Suma, Zachary Lipps, Samantha Finkelstein, David M. Krum, and Mark Bolas. 2012. Impossible Spaces: Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture. *IEEE Transactions on Visualization and Computer Graphics* 18, 4 (April 2012), 555-564. http://dx.doi.org/10.1109/TVCG.2012.47

[90] Qi Sun, Li-Yi Wei, and Arie Kaufman. 2016. Mapping virtual and physical reality. *ACM Trans. Graph.* 35, 4, Article 64 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925883

[91] Sunspring. Retrieved January 15, 2021 from https://www.thereforefilms.com/sunspring.html

[92] Tea For God. Retrieved January 15, 2021 from https://void-room.itch.io/tea-for-god

[93] James N. Templeman, Patricia S. Denbrook and Linda E. Sibert. 1999. Virtual Locomotion: Walking in Place through Virtual Environments. In *Presence* 8(6), 598-617. https://doi.org/10.1162/105474699566512

[94] Unseen Diplomacy VR real-walking game. Retrieved January 15, 2021 from http://store.steampowered.com/app/429830/Unseen_Diplomacy/

[95] Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P. Brooks, Jr. 1999. Walking > walking-in-place > flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '99), 359-364. https://doi.org/10.1145/311535.311589

[96] James Vallino and Christopher Brown. 1999. Haptics in augmented reality. In *Proceedings IEEE International Conference on Multimedia*

Computing and Systems (IEEE Comput. Soc), 195–200. http://doi.org/10.1109/MMCS.1999.779146

[97] Khrystyna Vasylevska, Hannes Kaufmann, Mark Bolas and Evan A. Suma. 2013. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *2013 IEEE Symposium on 3D User Interfaces* (3DUI'13), 39–42. https://doi.org/10.1109/3DUI.2013.6550194

[98] Khrystyna Vasylevska and Hannes Kaufmann. 2017. Compressing VR: Fitting Large Virtual Environments within Limited Physical Space. In *IEEE Computer Graphics and Applications*, vol. 37, no. 5, pp. 85-91, 2017. https://doi: 10.1109/MCG.2017.3621226

[99] VirtualSpace, Online Repository. https://github.com/HPI-VirtualSpace

[100] Vive and Vive Trackers. Retrieved January 15, 2021 from https:// www.vive.com/us/vive-tracker/

[101] The Void. Retrieved January 15, 2021 from https://www.thevoid.com/

[102] Ariel Weingarten, Ben Lafreniere, George Fitzmaurice, and Tovi Grossman. 2019. DreamRooms: Prototyping Rooms in Collaboration with a Generative Process. In *Proceedings of the 45th Graphics Interface Conference on Proceedings of Graphics Interface 2019* (GI'19). Canadian Human-Computer Communications Society, Waterloo, CAN, Article 19, 1–9. https://doi.org/10.20380/GI2019.19

[103] Betsy Williams, Gayathri Narasimham, Bjoern Rump, Timothy P. McNamara, Thomas H. Carr, John Rieser, and Bobby Bodenheimer. 2007. Exploring large virtual environments with an HMD when physical space is limited. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization* (APGV '07), 41-48. https://doi.org/ 10.1145/1272582.1272590

[104] Bob G. Witmer and Michael J. Singer. 1998. Measuring Presence in Virtual Environments: A Presence Questionnaire. Presence:

Teleoperators and Virtual Environments 7(3), 225-240. http://doi.org/10.1162/105474698565686

[105] Jackie (Junrui) Yang, Christian Holz, Eyal Ofek, and Andrew D. Wilson. 2019. DreamWalker: Substituting Real-World Walking Experiences with a Virtual Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (UIST '19). Association for Computing Machinery, New York, NY, USA, 1093–1107. https://doi.org/10.1145/3332165. 3347875

[106] Zero Latency. Retrieved January 15, 2021 from https://zerolatencyvr.com/

[107] Yiwei Zhao and Sean Follmer. 2018. A Functional Optimization Based Approach for Continuous 3D Retargeted Touch of Arbitrary, Complex Boundaries in Haptic Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, Paper 544, 12 pages. https://doi.org/10.1145/ 3173574.3174118