Prop-oriented world rotation: enabling passive haptic feedback by aligning real and virtual objects in virtual reality

Steven G. Wheeler[1] · **Simon Hoermann**[2] · **Robert W. Lindeman**[1] · **George Ghinea**[3] · **Alexandra Covaci**[4]

Abstract

Passive haptics have long been used to enhance the user's experience in virtual reality (VR). However, creating props to be used in a virtual environment can be a complicated and lengthy process. Current research looks to create passive haptic props based on the layout of, or objects in, the user's real environment. However, we identify three key limitations of current research. Firstly, procedural generation introduces many unknown variables into the design process, which complicates applying such techniques to scenarios requiring knowledge of the virtual environment's layout ahead of time. Furthermore, such techniques limit the size and dimensions of the virtual space to that of the real space. Lastly, current research necessitates pre-scanning or real-time scanning of the user's real environment, often requiring specialist equipment and expertise, thus limiting its generalisability. This research proposes *Prop Oriented World Rotation*, a technique that attempts to answer the aforementioned limitations and simplify the process of adding haptic feedback to VR applications. We implemented this technique in a demonstration game and give an overview of the steps taken to apply the technique in a real context. We analysed the demonstration system's performance and conducted an initial user evaluation in three different physical environments. While our stress test of the system's performance highlights the necessity for certain optimisations in complex environments, our initial user feedback suggests that users experienced a stronger sense of presence and feelings of safety in our passive haptics-enhanced environment. Hence, we conclude that our proposal has the potential to enhance experiences in VR with haptic feedback.

Keywords Virtual reality · Passive haptics · Presence · Human-computer interaction

1 Introduction

Since its inception, virtual reality (VR) research has mainly focused on the visual and aural senses. To enhance the sense of presence in virtual environments (VEs), researchers have since investigated the addition of touch [1–3]. There are two primary approaches for enabling

---

✉ Steven G. Wheeler
 steven.wheeler@pg.canterbury.ac.nz

Extended author information available on the last page of the article

haptic feedback in a VE – the use of either active or passive interfaces. Active haptics techniques simulate the sensation of touch with purpose-built haptic feedback devices under computer control [4], while passive haptics uses physical props representing virtual objects to provide feedback through their intrinsic shapes or textures [5]. A key advantage of active haptics techniques is their versatility – a single device can simulate haptic feedback for many different objects. However, in addition to being expensive and computationally demanding [2], these devices introduce an extra layer of complexity; many are cumbersome to take on and off or require additional time to learn how to operate them correctly [4].

In contrast, with passive haptics, virtual objects can be represented by objects in the user's surroundings, thus allowing for more natural and intuitive interactions [2]. Furthermore, unlike haptic feedback devices, these props are economical, potentially allowing passive haptics to find consumer-level adoption. However, the main disadvantage of passive haptics is that a single purpose-built object cannot easily represent a wide range of virtual objects of varying form factors. In addition, creating physical props to be used in VEs can be time-consuming and technical. Due to this, recent research has focused on procedurally generating VEs using the user's surroundings as a template [6–8]. Research into procedurally generating props based on the user's environment is highly promising as no bespoke props need to be made ahead of time, and such techniques received positive feedback from study participants related to increased feelings of safety, realism, and presence [7, 9].

However, while promising, we identify three primary limitations when using such techniques. Firstly, procedural generation of the environment introduces many unknown variables, which can complicate designing experiences in such environments [10]. When generating virtual worlds based on the user's surroundings, the designers of the VE will not know the room's layout or the number of potential props. Therefore, the design of the experience must be flexible enough to cater to all room layouts and potential numbers of props. Due to this uncertainty, in-depth, story-driven experiences are harder to implement when using procedural generation [10], with such techniques lending themselves more to random, labyrinth-like experiences [11].

Secondly, generating a virtual world based on the layout of the user's surroundings limits the traversable virtual space to the dimensions of the user's environment, which further limits the applicability of such techniques. Simeone et al. [6] identified this coupling of virtual and real spaces as a limitation. In their recommendations for future research, the authors suggested turning the user 180 degrees when they reach a transition point in the environment (such as a door), which would then warp the user to a different virtual space. Sun et al. [8] and Sra et al. [7] recommended a teleportation system that allows the user to explore large virtual spaces without a large physical play area. However, each of these techniques requires real-time or pre-scanning of the real environment (RE), which necessitates that the potential end-users have specialised equipment for scanning.

Due to these limitations, we believe a research gap has yet to be adequately filled for a system which provides passive haptics that is both easy to use for the end-user and to implement by VR developers. To fulfil both aspects, we consider the system must be flexible enough to fit in various applications and not be bound to the dimensions of the user's real surroundings. Furthermore, it should not require special equipment or complex configuration. Lastly, it should be viable for such a system to be incorporated into an existing VR application rather than being specifically designed for it.

In this paper, we present our proof-of-concept, *Prop-Oriented World Rotation* (*POWR*), an easy-to-configure, deterministic yet flexible alternative to previously proposed techniques of creating passive haptic props based on the user's environment. Our proposed technique avoids procedural generation while reinforcing virtual objects with real objects from the

user's environment. Furthermore, the flexibility of our approach allows any virtual object in the scene to be reinforced by a physical object, which, in turn, does not limit the play space of the user's environment to the dimensions of their real surroundings. To demonstrate this technique, we devised a simple system of importing real-world objects into the virtual coordinate space and created a demonstration game. We detail the steps taken to incorporate *POWR* into the demonstration game, evaluate its performance and present preliminary user studies with six participants. We show that, although not without its own limitations, *POWR* is a solid proof-of-concept that enhances VR experiences with passive haptic feedback without procedural generation or limiting the user's virtual play space.

In summary, the key contributions of this paper are:

- *POWR* - a technique of aligning virtual and real-world objects by rotating and translating the virtual environment to turn them into passive haptic props.
- An implementation of *POWR* in a VR demonstration game, and outlining the steps taken to incorporate the technique into a VR application.
- A performance evaluation of *POWR* where we look into the technical implications of adding *POWR* to VR experiences.
- A preliminary evaluation of user experience while interacting with our VR shooter game with and without haptic feedback via *POWR*.

2 Related work

In the work of Simeone et al. [6] and Valentini et al. [12], the authors generate VEs that map onto the user's real-life surroundings, thus enabling participants to touch virtual objects physically. Both Simeone et al. [6] and Valentini et al. [12] focus on generating virtual worlds based on real-world items, where each item is substituted with an object specifically relevant to the environment's theme. For example, in a medieval fantasy setting, a bed from the RE could become a boulder or a treasure chest in the VE.

Simeone et al. [6] use *OptiTrack Flex 3 cameras* to track the absolute position of all tracked objects in the virtual environment. The room was filled with typical furniture in a house, and the authors measured the extents of each piece of furniture. The furniture measurements were then used to create 3D models that fit within the measured volumes to substitute real-world objects. The authors pre-made two environments for participants to interact with.

Valentini et al. [12] pre-scan the real environment with a *Microsoft Kinect* and *Skanect* software. The authors then segment the scan through voxelisation, which produces a series of cubes that approximate the geometry of the original detailed 3D scan. This simplified geometry is substituted with virtual environmental props, such as rocks. The authors do not propose object detection; rather, all the generated cubes based on the real environment are treated equally and replaced with the same type of prop but scaled to fit within the target cube's volume. The authors' proposed technique allows the user to touch and interact with the virtual world physically reinforced by the real environment.

Sra et al. [7] primarily focus on generating virtual worlds based on the floor layout of the user's room but also include real-world objects that a 3D model of the real-world counterpart virtually represents. In the work of Sra et al. [7], no substitution is made; a virtual chair represents a real chair. Sra et al. [7] use a *Tango* device, which features a motion and depth-sensing camera, to scan the real environment. The point cloud of the real-world scan is then analysed, and a walkable area is estimated. Once analysed, a virtual world is generated based on the walkable area delimited by virtual objects typically understood as impassable

boundaries in day-to-day life, such as fences or bushes. Real objects are detected and classified during the scanning phase with the *Tango* and are later tracked in real-time so the user can actively interact with the object.

Cheng et al. [13] present *iTurk*: a system that allows the user to reuse a small selection of props in various scenarios. *iTurk* does not create virtual objects from the user's environment but rather has a small set of configurable props, which are actively tracked using an attached *HTC* device, in which the user performs tasks. The demonstration application displays the prop as a different virtual item to the user and once the user completes the task the scene and the item change appearance. Each task the users perform results in the props being left in the correct starting position for the following scene.

The authors demonstrate their system using two generic props placed in the RE: a pendulum hanging from the ceiling and a foldable board (a group of three boards attached together with hinges). In the demo, the user must complete certain tasks which require interacting with the props. For example, in one scenario the foldable board is positioned vertically and represents a fuse box; the user needs to open the "door" of the fuse box (i.e. they open one of the hinged boards) and flip a switch to turn off the "light" (the pendulum in the RE). On completion, they continue to the next scenario: a nuclear reactor. In this new scene, what was previously the fuse box (the foldable board left in an open position) is now a safety railing; the light is now a floating ball of plasma.

The authors measured enjoyment and realism levels in a within-subjects study involving 12 participants who tried a demonstration scene with and without haptic feedback. In the study, participants reported a higher level of enjoyment and experiencing a higher level of realism in the condition using *iTurk* compared to the non-haptic scenario.

Kohli et al. [14] present a system that looks to combine both the concept of *redirected walking* [15] – imperceptibly rotating the VE while the user walks in the RE – with passive haptics. In their example, the real environment contains only a single pedestal that will be used, unknowingly to the user, to provide haptic feedback for five virtual pedestals. To change which virtual pedestal is paired with the real one, the user is guided into walking to a particular point in the scene and rotating their head. To facilitate this, the users are given a particular task to perform with the help of a virtual character whose purpose is twofold: guiding the user to stand in a position equidistant from the real and virtual objects so that the rotation will align the new pairing correctly, and tracing a path that manipulates the user into moving their head back and forth. While the user is distracted performing the task, the world is rotated incrementally. A pilot study was conducted using their system, and the authors found that, while the system worked, for more complex scenarios the requirement to have the user act in a particular way (i.e., walk in a certain direction or look at a specific point) would be difficult, if not impossible, for more complex environments.

Azmandian et al. [16] present *haptic retargeting*, a system that, imperceptibly to the user, uses one prop to represent multiple virtual objects. Azmandian et al. [16] use an overhead camera that tracks the user's hand and incrementally warps the position/orientation of the user's virtual body, the VE itself, or both. All three techniques were tested with 20 participants which resulted in the user, with varying levels of perceptibility, placing their hand on the same prop in the RE while believing they have touched a different object in the VE. The idea of warping the world to align with a prop shares many similarities with the work discussed above of Kohli et al. [14]. However, instead of rotating the world about the user, Azmandian et al. [16] rotate the world with the prop as the central pivot point. As the prop is static, its position is reliable and can be used to rotate the world consistently. This method removes the variance in rotations caused by the user not precisely standing in the ideal pivot point, as exhibited in the system of Kohli et al. [14].
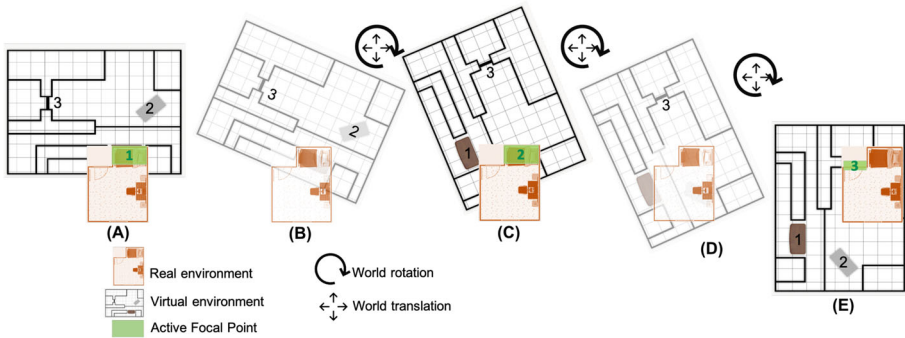
Fig. 1 Green: the currently aligned real-world object ('focal point'). Red: Fig. 2a overlaid on Fig. 2b. Orange: the user. NB: only the virtual environment moves, never the real environment represented in the red rectangle

3 Prop oriented world rotation

Figure 1 shows "Prop Oriented World Rotation", our proposed technique, which aligns real and virtual objects by translating and rotating the virtual world. With *POWR*, we adopt a similar approach to Simeone et al. [6], Sra et al. [7] and Valentini et al. [12], by using real-world objects as templates for virtual objects. However, instead of generating the entire virtual world based on the user's real-world surroundings, we propose using a small collection of key objects, known as "focal points" (FP), from the user's surroundings that the user predefines. Likewise, we propose that the application developer defines designated virtual objects in their environment to be used as these physically reinforced focal points.

Our proposed technique works by making the current focal point the root of the scene graph and moving all child objects relevant to the focal point's position and rotation. To demonstrate translating and rotating the world based on a new FP, observe Fig. 2a: a bird's-eye view of the room and objects defined by the user. Figure 2b and c show an example virtual scene and highlight the virtual objects designated by the developer to be reinforced by real objects. Figure 1 demonstrates two transitions between three FPs. In the first diagram ('A') of Fig. 1, the scene has already moved and aligned the first real object with the focal point '1'. Diagrams 'B' and 'C' then demonstrate the virtual world translating and rotating to align the second FP to its associated real object (in this example, both FPs share the same real object, labelled *cuboid* in Fig. 2a). The scene is translated again to the third FP, demonstrated by diagrams C, D and E in Fig. 1; in this example, the real object changes from the item labelled 'cuboid' in Fig. 2a to the item labelled 'door'.

While only one virtual object can be reinforced at any given time, we consider this concession worthwhile due to the flexibility this approach gives the developer. With our approach, when designing VR applications, the developers only need to consider the virtual objects they have designated to be reinforced by a physical object. As such, the number of unknown variables is vastly reduced, unlike the work of Valentini et al. [12] or Sra et al. [7] where the entire environment is generated and unknown to the developer when designing the experience. Due to this, we believe our approach applies to a wider range of VR applications and does not require the VR experience to be specifically designed with passive haptics in mind.

Furthermore, unlike all previously mentioned related work, which relies on active tracking placed on props [13, 17], real-time tracking via cameras [8, 16], or pre-scanning environments to generate procedural layouts [6, 7, 12] for virtual worlds, our approach only requires very simple coordinate data. As no sophisticated equipment is necessary to scan or process the
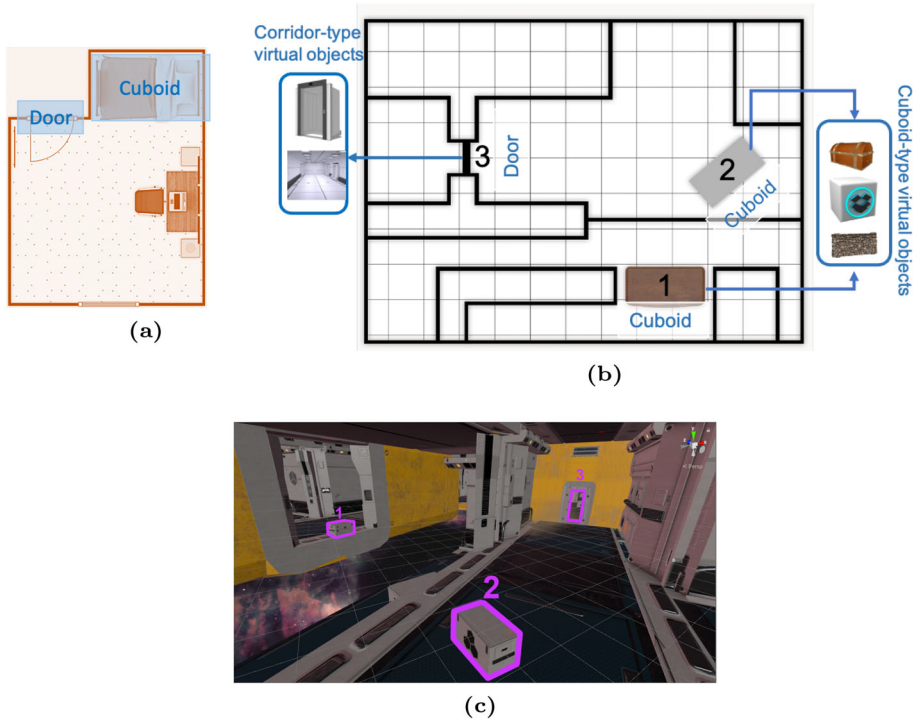
Fig. 2 (a) A birds-eye-view of an example real-world environment with real-world objects labelled as primitive shape types. (b) The layout of an example VE built to demonstrate *POWR*. The developer-designated virtual objects to be reinforced by real-world objects are numbered. (c) A screenshot that illustrates the VE described by the layout in Fig. 2b. The virtual objects that can be configured to provide haptic feedback are highlighted in pink and numbered

environment for our system to work, we consider the barrier of entry to be considerably lower when configuring or incorporating passive haptics in a VR application.

POWR has a similar objective to *iTurk* of re-purposing a small selection of props to be used in multiple distinct situations. However, unlike *iTurk*, *POWR* does not require the user to continuously configure the physical props in a specific way for the system to function correctly. Instead, there is no configuration from the end user after they have defined the real objects they wish to use as focal points. However, this comes at the disadvantage that the real objects that reinforce the virtual objects in *POWR* must remain stationary, while *iTurk* allows for more interactivity at the cost of more user involvement to ensure the system works correctly.

Likewise, *POWR* is a similar approach to the work of Kohli et al. [14] in that the player only interacts with a single real object at a time, and the VE is then repositioned to align a new virtual object to the same real object. However, Kohli et al. [14] align the virtual and real objects using rotation only, without translation, which requires the user to stand at a specific point in the environment for the rotation to align the objects correctly. Due to needing the user to stand and behave in a certain way, the real/virtual object pairing is prone to desynchronising due to slight errors in the rotation, which the user then has to reset manually when the margin of error is too large. However, in our system, the user and the virtual coordinates of the real-world objects remain stationary. *POWR* translates and rotates the virtual world itself to

align the static user-defined virtual coordinates of the physical objects with the virtual objects designated by the developer to be reinforced physically. Initial prototypes experimented with translating and rotating all user-defined virtual coordinates relative to the player's position as they moved through the scene. However, we encountered that this introduced a margin of error similar to what Kohli et al. [14] described in their study. As such, similar to Azmandian et al. [16]'s approach mentioned previously, we decided to keep the virtual coordinates and the user static and instead transform the world to align the real/virtual object pairing. To the user, translating and rotating the virtual world around them gives the same effect as moving through the scene as normal but ensures the object pairings do not desynchronise.

While Kohli et al. [14] and Azmandian et al. [16] aim to subtly change the alignment of the virtual and real objects in a way that is imperceptible to the user, *POWR* as a technique does not detail a method of masking the re-alignment or redirecting the user to the new object. *POWR* refers specifically to the rotation and translation of the environment to align user-defined real objects and developer-defined virtual objects, and we consider masking the re-alignment as a context-dependent implementation detail. For example, in our implementation of *POWR* detailed in the next section, we mask the realignment by using the momentary fading of the user's view during teleportation as an opportunity to re-orientate the world.

In summary, our approach's main benefit over related work is its simplicity and flexibility. Our approach requires the developer to consider only the designated objects when designing their application instead of designing experiences around a purely procedural world. Similarly, such an approach does not bind the virtual world's dimensions to that of the real world, allowing the user to teleport around the environment while still physically interacting with virtual objects. However, the main limitations of our approach are there may only be one physically reinforced object at any given time, and the real objects must remain static. Furthermore, while its approach is less complicated as the user does not need to behave in a certain way, *POWR* is perceptible to the user, unlike Kohli et al. [14] or Azmandian et al. [16]. Lastly, translating and rotating the virtual environment about the user could potentially be inefficient and affect performance. As such, while we believe our approach is easy to incorporate, it still leaves some implementation details to the developer.

The following section will outline our implementation and the steps to incorporate *POWR*. Furthermore, we will analyse the performance implications of incorporating *POWR* into a VR application by testing its effects on our demonstration game.

4 Implementation

At its core, our implementation of *POWR* consists of a series of *Unity* scripts that developers can add to their *Unity* VR applications to enable haptic feedback. However, it is important to note that, although our implementation was created in *Unity*, the concepts and techniques outlined in this paper can be applied to any game engine. In this section, we will discuss the demonstration game that we created for this research, how *POWR* was incorporated into the project and what steps end-users and developers would need to take to incorporate *POWR* into their projects.

4.1 Mapping real & virtual objects

Our system of implementing *POWR* requires the developer to define a set of virtual objects (by applying and configuring the relevant scripts via the *Unity* editor) that will be used

to represent the real-world objects selected by the user. Following this, to enable users to experience the sensation of touch while in VR, our system then: 1) scales the virtual objects to the dimensions of the real-world objects defined by users; 2) translates and rotates the VE so that the position of the virtual object matches one of the real objects (*POWR)*. In this section, we explain the key steps the developer and end-user need to carry out to use our system.

4.1.1 Designating virtual objects

For a developer to incorporate *POWR* into their application, they must designate the objects in their scene they wish to be reinforced by a physical object by attaching the relevant scripts to the object in *Unity*. Each designated virtual object must be assigned a primitive shape that best represents each chosen virtual object. These primitive shapes define the collision volume of the virtual object. Currently, we only use two generic primitive shapes – *cuboids* and *doors*. A *cuboid* represents virtual objects that can be contained within a cube collision volume that the user can touch and interact with. However, a *door* primitive is essentially an inverted *cuboid* – a gap surrounded by a solid object; the user can touch the sides of the gap (e.g., the door frame) and pass through the centre of the object (e.g., the doorway). The advantage of these two primitive shapes is that just two user-defined vertices can represent them, and either primitive shape can be used to approximate a wide range of real or virtual objects. A cuboid could be a bed or chest of draws in real life but be represented by a treasure chest or crate in the virtual environment. A door in the virtual environment could represent a door in the real environment, but, for example, a slim corridor or an enclosed mountain passage could also be appropriate. In future iterations of our system, more shapes could be added to give developers a wider selection from which to choose. These shapes could be more complex, such as a sofa or table, but would require more user-defined data and be less generic in its application. As our implementation is a proof of concept, we only focus on these two primitive shapes in this research.

4.1.2 Defining real objects

The user defines the bounds of the real-world objects for our system to use in a separate calibration application. The data the user defines in this environment is saved and can be loaded by VR applications that require it. To define an object, the user first selects a primitive shape type (as previously mentioned, currently, there are only two options: *cuboid* and *door*) that would best represent the real-world object they want to define. The user then defines the real-world object by measuring its dimensions, achieved by placing the controller on the bottom-left corner of the front of the object and pressing a button on the controller, which stores the corner's position; this process is then repeated for the top-right corner. Currently, in our calibration application, the user is expected to temporarily lift the VR headset to see the real object when defining its corners.

In Fig. 3b, these user-defined vertices are represented as blue dots. As mentioned in the last subsection, by defining the vertices of opposing corners of the real-world object, we have the necessary information to interpolate the other vertices of the cuboid. Expressed in pseudocode, imagining that we have already defined the bottom left and top right corners of the front face of the object, the remaining two vertices of the front face would be defined as
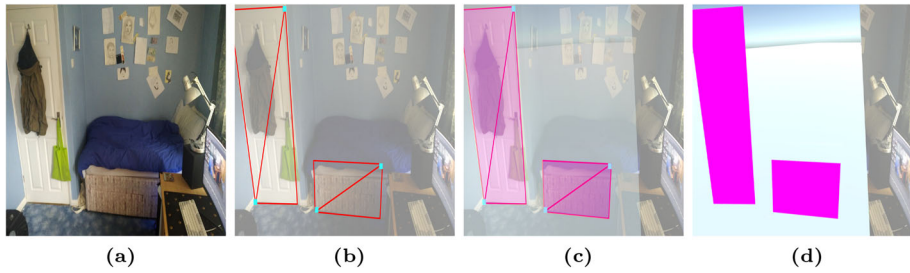
Fig. 3 a) The player's room (the RE). b) In blue: the points defined by the player. In red: the outline of the planes that will be generated from the points in blue. c) The player's room is overlaid by the *calibration room*. d) The *calibration room* with the generated planes representing the door (primitive shape type: *door*) and the bed (primitive shape type: *cuboid*). NB: The door in the player's bedroom should be open when playing the game

follows:

$$\text{topLeft}.x = \text{bottomLeft}.x$$
$$\text{topLeft}.y = \text{topRight}.y$$
$$\text{bottomRight}.x = \text{bottomLeft}.x$$
$$\text{bottomRight}.y = \text{topRight}.y$$

In our implementation, we only require the user to define the front face of the object. We do not capture any information regarding the depth of the object. However, to define an object with depth information, we still would only require two vertices: the user would define the bottom left vertex of the front face of the object and the top left vertex of the back face, and the remaining vertices can be interpolated in the same manner as the front face.

Once these two vertices are defined, our system then creates a simple virtual representation of the real-world object (a plane which represents the front face of the object the user measured), which then appears in front of the user (see Fig. 3d). It is important to note that, if more convenient, the user need not define the entirety of the real-world object and can instead use only a portion of it; for example, in Fig. 3b, only half of the bed is used.

In our implementation, we do not have any additional safety measures to prevent collisions with objects in the play area not defined by the user to be used as a 'focal point' in the application. Similar to the user's responsibility when defining the bounds of their play space in standard VR systems such as *Windows Mixed Reality* or *SteamVR*, the user is responsible for defining objects with a clear path between each other. For example, unobstructed real objects that are exclusively on the fringes of the boundaries of the cleared play space. Furthermore, the user must define objects appropriately to avoid collisions when using only part of an object, such as only half of the bed, as in Fig. 3b.

4.1.3 Scaling the virtual objects

At runtime, each of the designated virtual objects defined as described in Section 4.1.1 are scaled to match the size of the real-world objects selected by users in Section 4.1.2. For example, a virtual chest in the VE (of type *cuboid*) could be scaled to match the dimensions of a real-world bed (Fig. 3b) chosen by the user in the calibration application. Each virtual object is only scaled to a real object of the same primitive shape type. For example, a chest with type *cuboid* will never be scaled to a real object of type *door*. If multiple real objects

have been defined with the same primitive shape type, we arbitrarily assign one to the virtual object. In the future, this could be improved upon by assigning a real object of the most appropriate size or location for the virtual object.

4.2 Demonstration game

In this section, we explain the concept behind our demonstration game, why it was chosen, and outline other details that were needed to successfully implement *POWR* into our application.

4.2.1 Concept

To demonstrate *POWR*, we developed a shooter experience inspired by the popular video game *Time Crisis*[1] and similar to the VR game *Arizona Sunshine*.[2] In *Time Crisis*, the player uses objects in the environment as cover, which they duck behind for protection, and the player is stationary when shooting the enemies. Movement around the scene is prescribed, where a short, non-interactive cinematic transition is played between each predefined point of cover. We based our demonstration game on this type of experience due to the added benefit haptic feedback would give to ducking behind cover. Unlike *Time Crisis*, a non-VR experience, if users in VR are asked to duck, they must physically perform the action. Therefore, being able to touch the virtual object and rest or lean upon it physically provides a utilitarian benefit as well as potentially increasing levels of immersion. Furthermore, we believe that the cinematic transitions between each preset point of cover found in *Time Crisis*, or a similar transition mechanic, would provide an ideal opportunity to mask the rotation and translation of the virtual world (explained subsequently). Our demonstration game's objective is to progress through the level and defeat all the enemies the player encounters, and lasts approximately ten minutes; the player is unable to lose in the game.

4.2.2 Moving around the virtual environment

Our demonstration game has various points of cover throughout the virtual environment, which are physically reinforced by both *cuboid* or *door* primitive shapes. Once the player defeats the enemies near this point of cover, a teleportation point (a beacon of light with an arrow indicator that the player can interact with) appears by the next point of cover. Once ready to continue, the player points and selects the teleportation point, their view is temporarily faded to black, and *POWR* is used to shift the level to align a new virtual object (a 'focal point') with a real object.

However, a consequence of this design choice is that if the player is not standing in front of the next real object that will be used to reinforce a virtual object, the player must be first guided to the correct real-world location before applying *POWR*. To illustrate this, observe the transition shown by Fig. 1C, D and E, the VE aligns FP 3 with its associated real-world object (the player's door in the RE). If the player continues to stand in the same position that they occupied in Fig. 1C (in which they were standing in front of their bed in the RE), then, once the world shifts to align the new virtual object (FP 3), the player would be incorrectly positioned in the VE; for example, they would be facing a wall or possibly be placed within the virtual terrain itself. To avoid this, once the player initiates the teleportation – if they

(a) The player initiating the teleport.

(b) The player being guided where to stand.

(c) The completed teleport, with the player now standing in front of the door in the RE.
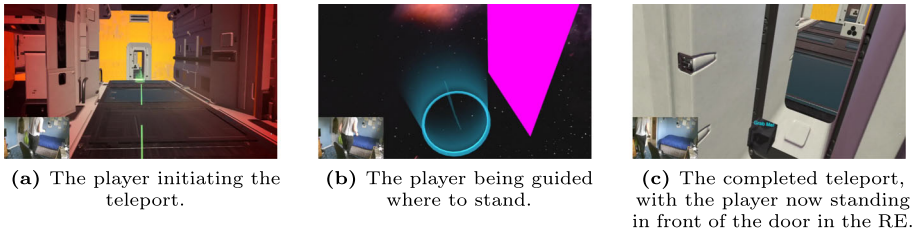
Fig. 4 The player re-positioning themselves when teleporting

are not already standing in the correct position in the RE – the virtual world fades out and shows only two things: all the real-world objects the user has previously defined and a marker indicating where they should stand next (see Fig. 4b). Once the player has physically moved to stand on the marker (thereby now standing in the correct RE position), the virtual world then completes the teleportation as usual (Fig. 4c).

5 Performance profiling

POWR transposes the environment to align a virtual object onto its real-world counterpart. However, due to the continuous repositioning of the VE, several common performance optimisations are difficult or impossible to implement in an application that uses *POWR*. For example, optimisations that require pre-calculations based on static geometry, such as *Light Baking* (pre-calculating the light sources of an environment) are not easily possible due to the constant movement of the VE. Furthermore, *Occlusion Culling* (the culling of objects that are obscured by another or outside of the camera's frustum) needs to be calculated entirely in real-time (i.e., without any prior calculation), which could potentially impact performance. Therefore, in this section, we profile the performance of our demonstration game with and without the optimisations that cannot be enabled when using *POWR*. Furthermore, due to the majority of POWR's added computation occurring while the user is teleporting, we will also profile the performance of the teleportation script versus the standard *SteamVR* teleportation script.

5.1 Procedure

As the demo itself is graphically basic and potentially not demonstrative of the complexity of a commercial product, a stress test was devised to test the performance of *POWR* under more computationally demanding conditions. The stress test was achieved by placing a 3D model in the scene consisting of 162 million vertices with many additional light sources to add to the computational workload. The demo scene without this 3D model contained 1.6 million vertices. While not a substitute for testing *POWR* with a real project of significant complexity, this model adds to the scene's computational demands, and we consider that it will yield more interesting insights into the performance ramifications of applying *POWR*.

The demonstration environment was tested incrementally, starting with a 'benchmark' with all optimisations enabled and without the 'stress test' 3D model. Following this, the 3D model was included in the scene, and the environment was tested with all optimisations

enabled. For each subsequent test, an optimisation was disabled until none were being used. By gradually removing optimisations and testing separately, we can better observe the impact of each optimisation technique. The removal order of the optimisations was based on the hypothesised impact of each technique (from most to least) based on past experience and *Unity* guidelines [18].

Each test was conducted for exactly three minutes to stabilise the average frame rate and minimise any external factors affecting the results (such as a task running in the background). The player's camera was stationary and looking at the same point on a wall for consistency between tests. Apart from the benchmark where the model was absent, the 'stress test' 3D model was placed behind the aforementioned wall, allowing the object to be culled via *Occlusion Culling*. All tests were run using single-pass rendering per Unity's recommended guidelines for optimising VR applications [18].

The teleportation scripts were profiled by inspecting their execution time via the Unity profiler. Both scripts were tested in identical environments with no optimisations enabled that were incompatible with *POWR*. The 'stress test' 3D object was not included in the scene.

Equipment Used All tests were performed on a system with 32GB of RAM (3GHz), an *AMD Ryzen 5 3600* CPU and a *GeForce RTX 2070 Super* GPU. The headset used in the tests was the *Valve Index* running at a refresh rate of 144Hz.

Minimum Baseline Performance *Flicker* caused by low refresh rates and, by extension, low frame rates, can be a significant contributing factor of *VR sickness* [19]. Most common VR headsets today focus on providing high refresh rates, with the lowest being 80Hz and the highest 144Hz [20]. Therefore, to provide a comfortable experience to all users and to be in line with the refresh rates of most headsets, the tests without the optimisations that are not possible to use when using *POWR* must achieve a frame rate of at least 80 FPS to be deemed acceptable.

5.2 Performance results

The 'stress test' 3D model had a marginal impact between the benchmark and the scene with all optimisations enabled (see Fig. 5). However, when disabling *Occlusion Culling*, the average FPS in the scene dropped by 82% compared to the benchmark. With *Light Baking* disabled, the average FPS dropped by 67% from the previous test and is 88% lower than the benchmark. Tests without any optimisations were 87% lower than the benchmark. Furthermore, tests with *Static Batching* and *Light Baking* showed minimal CPU usage (<0.1ms) required to calculate the global illumination of the scene compared to the test with *Static*
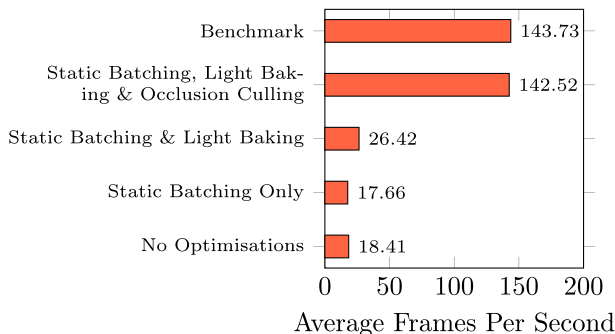

Fig. 5 Chart of average FPS of all tests

Batching enabled, but *Light Baking* disabled, where CPU usage ranged from 0.1ms to 0.25ms. Tests showed that teleportation with *POWR* is slower than regular SteamVR teleportation, with the CPU taking 6.51ms to execute POWR's teleportation script compared to 0.02ms. Executing the *POWR* script causes a momentary drop in FPS from 144 to 130.

6 User study

To validate our system and understand how it influences the user experience, we conducted a user study in which participants experienced our demo with and without *POWR* passive haptics.

6.1 Procedure

6.1.1 Interfaces

There were two interface conditions. In the *POWR* condition, users could select two real-world objects in their surroundings for in-game interaction. In the *control* condition, users experienced the game without haptic feedback.

6.1.2 Participants

In total, six participants took part in the user testing. Of the six, three had more than two years of experience with VR and three had minimal to no previous VR experience. The mean age of the participants was 45, and four identified as male and two as female.

6.1.3 Protocol

Due to COVID-19 restrictions, the testing took place in three rooms with different layouts (Fig. 6) – In Room 1 (3 participants: R1P1, R1P2, R1P3), the objects defined by users were positioned opposite each other; to transition between the two objects players only needed to turn and move a short distance. In Room 2 (1 participant: R2P1), the objects were adjacent to one another thus players only needed to step to one side when transitioning. In Room 3 (two participants: R3P1, R3P2), one object was placed in the centre of the room, which obstructed the player's path when transitioning to the second object.

Participants in Rooms 1 and 2 used an *Oculus Rift S* connected to a computer with an *AMD Ryzen 5 3600* CPU and a *GeForce RTX 2070 Super* GPU. Participants in Room 3 used an *Oculus Rift CV1* attached to a computer with an *Intel i7-7700k* CPU with a *GeForce GTX 1080* GPU. Both PCs had comparable performance for running the demonstration games and both *Oculus* headsets shared very similar resolutions and refresh rates. The most notable difference between the two setups was that the *Oculus Rift CV1* uses an external tracking solution while the *Oculus Rift S* does not.

The testing followed a within-subjects design with each participant testing both conditions, (*POWR* and *control*), in a randomised order: R1P1 (Control then Experimental), R1P2 (Experimental then Control), R1P3 (Experimental then Control), R2P1 (Control then Experimental), R3P1 (Control then Experimental), R3P2 (Experimental then Control). After completing each experimental condition, participants completed a questionnaire consisting of the validated and widely used Igroup Presence Questionnaire (IPQ) [21] and a customised
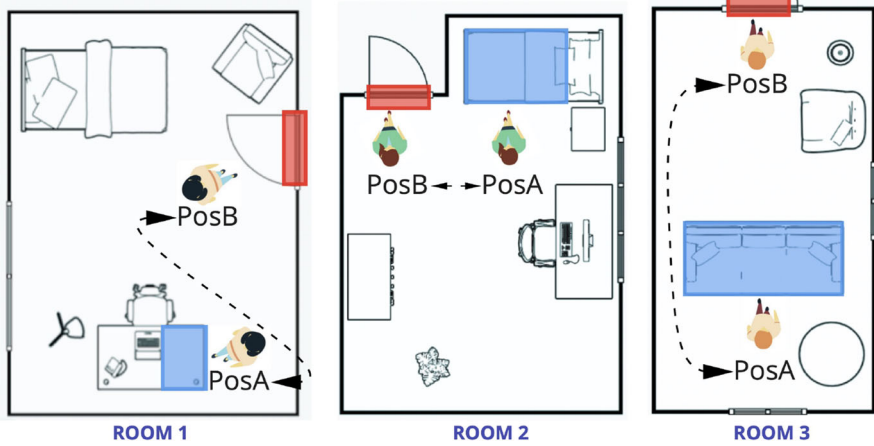
Fig. 6 Layouts of the three rooms used in the *POWR* user study

questionnaire. The IPQ consists of thirteen 7-point Likert scale questions varying from -3 = fully disagree/not at all, to +3 = fully agree/very much. The questionnaire evaluates three main aspects related to the sense of presence: 1) Spatial Presence - the sense of being 'physically there' in the VE. 2) Involvement - the attention paid to the VE and the involvement experienced. 3) Realism - the subjective experience of realism in the VE. An additional question was used to assess the general feeling of "being there" in the VE. The customized questionnaire – which consisted of three customised 5-point Likert scale questions – was used to give a rough indication of how safe and confident the users felt using the *POWR* condition compared to the control condition which used SteamVR's default – *Chaperone* (a 3D blue boundary marker that appears when users are too close to the edge of the play-area). All users had the option to give additional written feedback at the end of each questionnaire, but this was not compulsory.

6.2 Study results

Participants reported an enhanced feeling of presence in the *POWR* condition: *"I felt fully immersed in the virtual world."*, *"I felt I was actually in the VR world. Total immersion."*, *"Game was fun, felt pretty immersed in the virtual world."*. This is supported by the IPQ results (Fig. 7) where the *POWR* condition had a higher median score than the control for the generic feeling of "being there" (G), Spatial Presence (SP) and Realism (REAL) and similar for the attention paid to the VE (Involvement, INV).

Regarding the feeling of safety experienced by participants in both conditions, all but one user (R3P1) answered that they felt "very safe" in the *POWR* condition. For the same question, the control demo received lower scores with a median of 3, with only two participants awarding a maximum score for their feelings of safety (R3P1, R3P2). This sentiment was reflected in the user feedback in which participants expressed a *"feeling of trepidation"* (R1P2, R1P3), with some noting they felt *"afraid to move"* despite the protection of the default *Chaperone* system. During the experiment in Room 3, there were some minor tech-
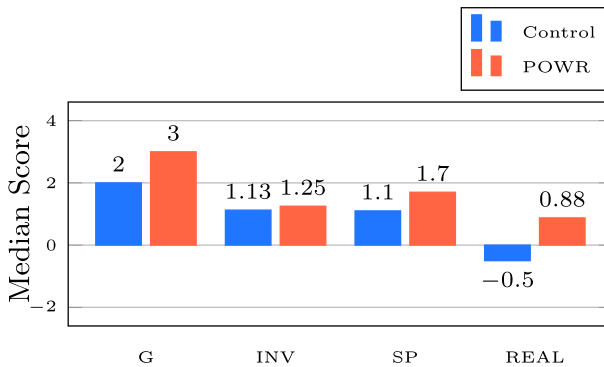
Fig. 7 Results of the IPQ Questionnaire

nical issues: the door was out of tracking range with the affected participant (R3P1) marking the safety of the *POWR* lower than the other participants.

For the demo using *POWR*, responses to how the player felt when they had to reposition themselves in the RE when moving between Focal Points (FPs) received a median score of 3/5, with two participants from Room 1 rating their feeling of safety maximum points. One participant from Room 3 (R3P2) stated that their confidence was affected due to the layout of their room (Fig. 6) where they had defined their cuboid object in such a way that they had to step around it to transition to the door object. Likewise, the single participant of Room 2 – a small, cramped room with both FPs next to each other – also gave a poor rating (2/5) for confidence in moving despite having to move very little in the RE. However, all three participants of Room 1 – the largest layout with a clear, unobstructed path between each FP – awarded the demo using *POWR* maximum score.

7 Discussion

Removing the optimisation techniques incompatible with POWR had a severe to moderate impact on the performance of the demonstration scene. The stress test especially showed the importance of *Occlusion Culling* in the scene, with its omission leading to the most significant drop in FPS. *Light Baking* was also shown to be significant, but less so than *Occlusion Culling*. In environments using more complicated lighting, such as objects with specular textures reflecting large volumes of light, the importance of *Light Baking* could be more significant than the drop in performance observed in the stress test.

The difference in performance between the two teleportation scripts was noticeable, with *POWR* performing worse than the standard teleportation script. However, the drop to 130 FPS is small (around a 10% drop), and far above our baseline of 80 FPS. Furthermore, as both versions of the teleportation temporarily fade the user's vision to black while teleported, any stutter that occurs when rotating the world using *POWR* is obscured from the user. Due to this, we tentatively conclude that the practical significance of the performance difference observed with the teleportation script is low. Therefore, rotating the world around the user while teleporting is not a bottleneck. Rather, the optimisations that must be disabled to use the technique are where performance suffers. However, while the stress test was designed to tax the system to observe the importance of each optimisation, the benchmark itself, without any optimisations, ran at the maximum FPS the headset allowed. Therefore, disabling

these optimisations in scenes with moderate computational demands may be a viable option. However, the performance profiling results demonstrate that, for more demanding scenes, a custom culling or *Light Baking* process would be necessary that does not rely on *Unity*'s built-in solution.

Overall feedback on the *POWR* demo was positive, with many participants commenting they felt safer and more engaged when experiencing passive-haptic feedback than in the *control* condition. User observations also supported this while interacting in both conditions: participants tended to be more vocal, animated and confident in their movements in the *POWR* demo. Our results are in line with the evaluation of systems that provide passive-haptic feedback through scanning and voxelisation of the scene [12], procedural generation [7] or via specific physical props [13]. The values obtained for the IPQ are consistent with the ones of Valentini et al. [12], who tested the effect of contextualised addition of virtual objects at the position of real-world objects following a similar procedure - six participants, two conditions, presence and safety evaluation.

Many users noted that, although they were aware of the RE, they believed that being conscious of where they were helped them participate more enthusiastically. For example, as the participant knew they were beside their door in the RE, they could confidently thrust out their arm through the doorway, knowing they would not hit something unexpectedly. This is reinforced by the safety questionnaire, which showed the majority of users felt much safer using *POWR* than when simply using the *Chaperone* boundary. In particular, participants noted they felt *"reassured"* and *"anchored"* by the real object as opposed to a *"feeling of being lost"* in a purely virtual space. This is particularly interesting as being aware of the RE traditionally is seen as a negative immersion-breaker in VR [21–23].

Accordingly, we consider that, when measuring *POWR*'s effect (or the effect of similar systems [6, 7, 12]) on Involvement or Spatial Presence, a greater distinction needs to be made between simply being aware of the RE – in *POWR*'s case, an object in the RE – and being negatively distracted or interrupted by it (such as tripping over a cable or losing tracking). As such, due to how these particular sub-scales are measured, it is difficult to look at the results of the IPQ and give a conclusive analysis of *POWR*'s effect on presence in these areas.

7.1 Limitations & future research

Although our implementation worked well in some technical VR setups, it is less suitable for others. For example, external trackers that have a limited range, as noted with the participants in Room 3 using the *Oculus Rift CV1*, can make setting up the play area inconvenient and, at times, impossible. Due to its infinite range, inside-out tracking avoids this issue, but we encountered separate issues with this form of tracking in which the orientation of the play area was changed periodically, which de-synchronised the real and virtual objects. However, although it does highlight that *POWR* is currently better suited to those who have certain headsets or tracking solutions, this issue of tracking range is more a restriction of certain VR setups as opposed to a limitation of *POWR* itself.

The room layout has also been shown to affect user experience: if the player does not have a clear path between them and their next FP, it puts them in an unsafe and immersion-breaking situation. However, in its current state, our method of physically relocating the user when teleporting, as shown in Fig. 4, is less than optimal. Asking the user to consider the real world while immersed in a virtual environment causes a 'break in presence', negatively impacting immersion [24].

Our study used a custom-built video game designed with suitability to be used with *POWR* in mind. Future research applying *POWR* to different forms of video games that were not designed with *POWR* in mind would be necessary to assess the usefulness of such a technique. While initially conceptualised with video games in mind, future research investigating the potential applications of *POWR* or similar techniques in different types of VR experiences outside the realm of video games would be beneficial. For example, using *POWR* to enhance architectural visualisations, or museums or exhibitions, with passive haptic feedback would be an interesting avenue of future research. In addition, research into combining *POWR* with different forms of locomotion, such as redirected walking [15], or iterating upon our proposed teleportation system would be necessary to investigate the breadth of applications that *POWR* could potentially be applied to.

POWR is dependent on whether the user has suitable objects in their room for the game to use. For example, if a player chooses a bed that is relatively low to the ground to represent a *cuboid* primitive shape, they would have to crouch a lot lower and have a less optimal experience than someone using a taller object. This was evident in the user testing, where certain users needed to add objects to their bed or couch to provide a better experience in the game. However, the solution to this issue can be as simple as adding cushions to raise the height of an object. On the other hand, the room layout is less easily changed and is a much larger issue.

Performance was tested using only a single system with high-end, modern components; on older or different hardware, the impact of using *POWR* may be more significant. Testing both demos on various hardware and scenes with different properties would be necessary for a more conclusive analysis.

As a consequence of limitations within our choice of game engine – Unity – it was not possible to enable *Occlusion Culling* in the demo that used *POWR*. Therefore, an investigation into the possibility of writing a custom solution to achieve the same result, instead of relying on the built-in *Occlusion Culling* feature that *Unity* already provides, could be of great value, especially as the need to disable these optimisations within *Unity* is the primary drawback when using *POWR* in a project.

Due to COVID-19, only six participants could be involved in user testing. Naturally, this heavily affects the validity of the results and conducting research with a larger sample of participants is recommended.

8 Conclusion

POWR enables the user to use objects from their surroundings to provide haptic feedback in the VE while allowing them to traverse large VEs without being restricted by the physical play area. User feedback was highly encouraging, and the demo was successfully played in three distinct rooms of different sizes and layouts, demonstrating its robustness and versatility. This was achieved without procedural elements, allowing more story-driven experiences to leverage our system. *POWR*'s performance in the stress test was less than ideal due to the lack of existing suitable optimisations. However, with additional time, *POWR* could incorporate its own custom optimisation methods, such as providing *Occlusion Culling*, to improve performance further.

POWR is a solid *proof of concept* which can be applied to VR applications, allowing them to take advantage of passive haptic feedback. Any issue arising from testing *POWR* has a solution; there are no fundamental problems that cannot be resolved. As such, we consider

POWR to have made a valid contribution that expands upon previous research and provides a solid foundation to build in the future.

Declarations

References

1. Hinckley K, Pausch R, Goble JC, Kassell NF (1994) Passive real-world interface props for neurosurgical visualization. In Proceedings of the SIGCHI conference on human factors in computing systems, Boston, MA, USA. New York, NY, USA, pp 452–458. ACM. Backup Publisher ACM
2. Hoffman HG (1998) Physically touching virtual objects using tactile augmentation enhances the realism of virtual environments. In Proceedings IEEE 1998 virtual reality annual international symposium (Cat. No. 98CB36180), Atlanta, GA, USA. New York, NY, USA, pp 59–63. IEEE. Backup Publisher IEEE
3. Insko BE (2001) Passive Haptics Significantly Enhances Virtual Environments. PhD Thesis, The University of North Carolina at Chapel Hill, Chapel Hill, North Carolina
4. Choi I, Ofek E, Benko H, Sinclair M, Holz C (2018) Demonstration of CLAW: A Multifunctional Handheld VR Haptic Controller. In Extended abstracts of the 2018 CHI conference on human factors in computing systems, Montreal, QC, Canada. New York, NY, USA, pp 1–4. ACM. ISBN 978-1-4503-5621-3. https://doi.org/10.1145/3170427.3186505. https://dl.acm.org/doi/10.1145/3170427.3186505
5. Lindeman RW, Sibert JL, Hahn JK (1999) Hand-held windows: towards effective 2D interaction in immersive virtual environments. In Proceedings IEEE Virtual Reality, Houston, TX, USA. New York, NY, USA, pp 205–212. IEEE Comput Soc ISBN 978-0-7695-0093-5. https://doi.org/10.1109/VR.1999.756952. http://ieeexplore.ieee.org/document/756952/
6. Simeone AL, Velloso E, Gellersen H (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, Seoul, Republic of Korea. New York, NY, USA, pp 3307–3316.

ACM. ISBN 978-1-4503-3145-6. https://doi.org/10.1145/2702123.2702389. https://dl.acm.org/doi/10.1145/2702123.2702389

7. Sra M, Garrido-Jurado S, Maes P (2018) Oasis: Procedurally Generated Social Virtual Spaces from 3D Scanned Real Spaces. IEEE Trans Vis Comput Graph 24(12):3174–3187. ISSN 1077-7626, 1941-0506, 2160-9306. https://doi.org/10.1109/TVCG.2017.2762691. https://ieeexplore.ieee.org/document/8067498/. Number 12

8. Sun Q, Wei L-Y, Kaufman A (2016) Mapping virtual and physical reality. ACM Trans Graph 35(4):1–12. ISSN 0730–0301:1557–7368. https://doi.org/10.1145/2897824.2925883. https://dl.acm.org/doi/10.1145/2897824.2925883. Number 4

9. Sra M, Schmandt C (2016) Bringing real objects, spaces, actions, and interactions into social VR. In 2016 IEEE Third VR International workshop on collaborative virtual environments (3DCVE). Greenville, SC, USA, pp 16–17. IEEE. ISBN 978-1-5090-2138-3. https://doi.org/10.1109/3DCVE.2016.7563561. http://ieeexplore.ieee.org/document/7563561/

10. Short T, Adams T (2017) Procedural generation in game design. CRC Press, Boca Raton, FL, USA, 1 edition. ISBN 1-4987-9920-5

11. van der Linden R, Lopes R, Bidarra R (2014) Procedural Generation of Dungeons. IEEE Trans Comput Intell AI Games 6(1):78–89. ISSN 1943-068X, 1943-0698. https://doi.org/10.1109/TCIAIG.2013.2290371. http://ieeexplore.ieee.org/document/6661386/. Number 1

12. Valentini I, Ballestin G, Bassano C, Solari F, Chessa M (2020) Improving Obstacle Awareness to Enhance Interaction in Virtual Reality. In 2020 IEEE Conference on virtual reality and 3d user interfaces (VR). Atlanta, GA, USA, pp 44–52 IEEE. ISBN 978-1-72815-608-8. https://doi.org/10.1109/VR46266.2020.00022. https://ieeexplore.ieee.org/document/9089649/

13. Cheng L-P, Chang L, Marwecki S, Baudisch P (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, Montreal, QC, Canada, New York, NY, USA, pp 1–10. ACM. ISBN 978-1-4503-5620-6. https://doi.org/10.1145/3173574.3173663. https://dl.acm.org/doi/10.1145/3173574.3173663

14. Kohli L, Burns E, Miller D, Fuchs H (2005) Combining passive haptics with redirected walking. In Proceedings of the 2005 international conference on Augmented tele-existence - ICAT '05, Christchurch, New Zealand. New York, NY, USA, p 253. ACM. ISBN 978-0-473-10657-7. https://doi.org/10.1145/1152399.1152451. http://portal.acm.org/citation.cfm?doid=1152399.1152451

15. Razzaque S (2005) Redirected walking. The University of North Carolina at Chapel Hill. ISBN 0-542-34053-4

16. Azmandian M, Hancock M, Benko H, Ofek E, Wilson AD (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, San Jose, CA, USA. New York, NY, USA, pp 1968–1979. ACM. ISBN 978-1-4503-3362-7. https://doi.org/10.1145/2858036.2858226. https://dl.acm.org/doi/10.1145/2858036.2858226

17. Cheng L-P, Roumen T, Rantzsch H, Köhler S, Schmidt P, Kovacs R, Jasper J, Kemper J, Baudisch P (2015) TurkDeck: Physical Virtual Reality Based on People. In Proceedings of the 28th annual ACM symposium on user interface software & technology, Charlotte NC USA. New York, NY, USA, pp 417–426. ACM. ISBN 978-1-4503-3779-3. https://doi.org/10.1145/2807442.2807463. https://dl.acm.org/doi/10.1145/2807442.2807463

18. Unity Technologies (2020) Optimizing your VR/AR experiences. https://learn.unity.com/tutorial/optimizing-your-vr-ar-experiences#5e60ff9cedbc2a002071e0f5

19. LaViola JJ (2000) A discussion of cybersickness in virtual environments. ACM SIGCHI Bulletin 32(1):47–56. ISSN 0736-6906. https://doi.org/10.1145/333329.333344. https://dl.acm.org/doi/10.1145/333329.333344. Number 1

20. Valve Software (2020) Steam Hardware Survey, July 2020. manual, Valve Software. https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam. tex.organization: Valve Software

21. igroup (2020) Presence Questionnaire 1 | igroup.org – project consortium. http://www.igroup.org/projects/ipq/

22. Witmer BG, Singer MJ (1998) Measuring Presence in Virtual Environments: A Presence Questionnaire. Presence: Teleoperators and Virtual Environments 7(3):225–240. ISSN 1054-7460. https://doi.org/10.1162/105474698565686. https://www.mitpressjournals.org/doi/abs/10.1162/105474698565686. Number 3

23. Slater M (1999) Measuring Presence: A Response to the Witmer and Singer Presence Questionnaire. Presence: Teleoperators and Virtual Environments 8(5):560–565. ISSN 1054-7460. https://doi.org/10.1162/105474699566477. https://direct.mit.edu/pvar/article/8/5/560-565/18256

24. Slater M, Brogni A, Steed A (2023) Physiological responses to breaks in presence: A pilot study. In Proceedings for presence 2003: The 6th annual international workshop on presence. Aalborg , Denmark, vol 157, Aalborg , Denmark, October 2003. International Society for Presence Research

Authors and Affiliations

Steven G. Wheeler[1] (ID) **· Simon Hoermann**[2] (ID) **· Robert W. Lindeman**[1] (ID) **· George Ghinea**[3] (ID) **· Alexandra Covaci**[4] (ID)

Simon Hoermann
simon.hoermann@canterbury.ac.nz

Robert W. Lindeman
rob.lindeman@canterbury.ac.nz

George Ghinea
george.ghinea@brunel.ac.uk

Alexandra Covaci
a.covaci@kent.ac.uk

[1] HIT Lab NZ, University of Canterbury, Kirkwood Avenue, Christchurch 8041, Canterbury, New Zealand

[2] School of Product Design, University of Canterbury, Kirkwood Avenue, Christchurch 8041, Canterbury, New Zealand

[3] Department of Computer Science, Brunel University, Kingston Lane, London 10587, London, UK

[4] School of Engineering, University of Kent, Giles Lane, Canterbury 610101, Kent, UK