A platform for developing and fine tuning adaptive 3D navigation techniques for the immersive web

AHMED KAMAL, Universitat Politècnica de Catalunya, Spain
CARLOS ANDUJAR, Universitat Politècnica de Catalunya, Spain

Navigating through a virtual environment is one of the major user tasks in the 3D web. Although hundreds of interaction techniques have been proposed to navigate through 3D scenes in desktop, mobile and VR headset systems, 3D navigation still poses a high entry barrier for many potential users. In this paper we discuss the design and implementation of a test platform to facilitate the creation and fine-tuning of interaction techniques for 3D navigation. We support the most common navigation metaphors (walking, flying, teleportation). The key idea is to let developers specify, at runtime, the exact mapping between user actions and virtual camera changes, for any of the supported metaphors. We demonstrate through many examples how this method can be used to adapt the navigation techniques to various people including persons with no previous 3D navigation skills, elderly people, and people with disabilities.

CCS Concepts: • **Computing methodologies** → *Virtual reality*; • **Human-centered computing** → **Virtual reality**; **Interaction techniques**.

Additional Key Words and Phrases: virtual reality, WebXR, navigation, 3D interaction techniques, immersive web

1 INTRODUCTION

In the last few years we have witnessed a substantial evolution of the different technologies that enable people to experience 3D virtual environments. Thanks to the advances in laser scanning and photogrammetry techniques, the 3D digitization of objects, buildings and sites has become a common practice in many fields including cultural heritage [Gomes et al. 2014; Pintore et al. 2020]. The Web is arguably the ideal platform for the dissemination of these 3D models. Since it is open and based on well-defined standards, anyone can publish content which can be accessed by anyone [Maclntyre and Smith 2018]. The web is by far the most widely available platform, and web browsers exist for desktop computers, tablets, smartphones and even AR/VR headsets. Furthermore, distributing content updates is easier for the Web [Engberg et al. 2018] and thus the Web is an excellent medium for dynamic 3D content. An example of web platform for interactive presentation of 3D models is SketchFab.

On the other hand, advances in display and sensor technologies have enabled the development of affordable AR/VR headsets [Al Zayer et al. 2020], including shell (phone-based) headsets, and standalone devices such as the Oculus Quest. Most of these VR devices are supported by the WebXR Device API, which provides a platform-independent interface to access VR and AR devices [Maclntyre and Smith 2018].

Navigation is one of the major user tasks in 3D virtual environments, and interaction techniques for 3D navigation have received substantial attention [Al Zayer et al. 2020]. A recent study has identified and characterized over one hundred navigation techniques for VR [Di Luca et al. 2021]. The reason for such an increasing variety of navigation techniques is twofold. On the one hand, AR/VR software and hardware technologies are in constant evolution, opening new opportunities for 3D interaction. On the other hand, current interaction techniques for 3D navigation have some limitations. For example, many works have pointed out the inherent trade-off between simulation sickness and accessibility [Di Luca et al. 2021]. As a result, no universal technique for 3D navigation exists that works well in all situations. The suitability of a certain interaction technique depends on many factors including type of application, virtual environment, target platform, and user profile.

The final publication is available at ACM via http://dx.doi.org/10.1145/3485444.3487643

We believe that there is an increasing need for more exploratory studies of VR navigation techniques, as well as for mechanisms to facilitate the adaptation of existing techniques to those individuals for which VR navigation might imply a high entry barrier, such as people with disabilities or no prior 3D/VR experience.

In this paper we present a method to facilitate the design, testing and fine-tuning of interaction techniques following the major navigation metaphors for VR (walking, flying and teleportation). At the core of these interaction techniques is the mapping of user input onto linear/angular speed/acceleration of the virtual camera, although other components also play an important role for the user experience, e.g. the addition of techniques for alleviating motion sickness [Fernandes and Feiner 2016]. Our main goal is to describe our experiences with a web platform (using JavaScript, WebXR and ThreeJS) for defining and evaluating different variants of major navigation metaphors. In particular, we designed the platform to support:

- Easy experimentation with interaction techniques for navigation, where developers can easily create and test new mappings through user-defined custom functions.
- Easy customization to particular 3D scenarios.
- Inclusive access to VR. Many of the navigation techniques available in games and other applications assume certain user capabilities that are not universally present. Using appropriate custom functions facilitates lowering the VR entry barrier for them.
- Adaptation to people with no experience on 3D navigation (e.g. art historians willing to explore a virtual museum).

2 PREVIOUS WORK

2.1 VR Navigation

Virtual reality navigation is an essential task in virtual environments. Navigation techniques impact essential aspects such as user performance, physical effort, comfort, and motion sickness. A proper design of the navigation technique could be the difference between enjoying the sense of presence in another world, and suffering discomfort, disorientation, and nausea [Al Zayer et al. 2020].

Real walking is arguably the most natural way to travel in VR. Many current VR headsets support room-size tracking and therefore real walking is an excellent option for small-scale navigation. However, a natural walking interface is not sufficient. VR navigation techniques must allow users to navigate virtual environments that extend far beyond the size of the physical space, to cover large virtual distances with small physical effort, and to inspect virtual environments at different heights and perspectives [Al Zayer et al. 2020].

Virtual locomotion techniques can be classified onto walking-based, steering-based, selection-based and manipulation-based techniques [Al Zayer et al. 2020; LaViola Jr et al. 2017]. *Walking-based* techniques include real-walking as well other techniques involving repetitive motor actions such as head-bobbing and arm swinging [Al Zayer et al. 2020]. Depending on the resemblance to the human gait cycle, walking-based techniques are classified as full gait, partial gait, and gait negation techniques [Al Zayer et al. 2020; LaViola Jr et al. 2017]. Redirection techniques [Razzaque et al. 2002; Steinicke et al. 2008; Suma et al. 2012] attempt to keep the user within the limits of the physical space while being able to travel in a larger virtual environment. Partial gait techniques, e.g. walking-in-place (WIP) have attracted much attention since its origins [Slater et al. 1995]. Gait negation techniques aim to provide the full gait cycle while keeping the user stationary, at the expense of dedicated mechanical devices such as treadmills [Al Zayer et al. 2020]. *Steering-based* techniques provide a continuous control of direction [LaViola Jr et al. 2017]. The virtual direction can be controlled using the user's body (e.g. gaze, hand, body leaning, and torso orientation) or through physical steering props [LaViola Jr et al. 2017]. *Selection-based* techniques greatly simplify navigation by letting users focus on where to get to instead of how to get to a specific location [LaViola Jr et al. 2017]. A widely known selection-based technique is teleportation [Schroeder et al. 2001], which moves the virtual camera to the chosen destination either instantly [Bozgeyikli et al. 2016] or gradually [Bhandari et al. 2018; Mackinlay et al. 1990]. Instant teleportation is very popular in the VR industry as it helps reducing the incidence of VR sickness. Despite a number of interaction metaphors and variants have been proposed in the literature, researchers agree that there is not a unique "best" technique for all scenarios, users and tasks.

2.2 Motion sickness

A major issue with VR headsets is that users often experience adverse symptoms such as nausea, dizziness, and eye strain [LaViola Jr et al. 2017; Lim et al. 2021]. These symptoms are often referred to as motion sickness or simulator sickness. A recent study with high-end VR headsets has shown that they still cause a greater level of sickness compared to desktop displays [Yildirim 2019]. A number of theories have been proposed to explain simulator sickness [LaViola Jr et al. 2017]. Two well-known theories are the cue-conflict theory [Kolasinski 1995] and the postural stability theory [Keshavarz et al. 2015]. Common methods for alleviating motion sickness are FOV reduction, brightness reduction, and independent visual background [Fernandes and Feiner 2016; Lim et al. 2021]. As stated above, some techniques such as instant teleportation lack optical flow and thus are unlikely to trigger motion sickness.

2.3 Inclusive VR

VR applications are often not designed to be inclusive to all users. Besides motion sickness issues, most VR applications require users to perform some physical movements which could be an important barrier. People with limited mobility might not be able to experience VR without assistance. Some recent works address the problem of inclusive VR design, as well as adapting VR applications for people with disabilities, using either physical hardware-based solutions or conceptual tools [Dombrowski et al. 2019]. Mott et al. [Mott et al. 2019] analyze alternative sensors to get user inputs from users with movement limitations, including motion, eye gaze, and audio sensors. Brain computer interfaces [Coogan and He 2018] are becoming a promising tool to provide VR control for people with serious motor function problems. Despite these encouraging works, many users are being left out from the vast majority of VR experiences in the immersive web [Dombrowski et al. 2019].

2.4 Evaluation testbeds for VR locomotion

A large number of systematic reviews and evaluation testbeds have been proposed for VR locomotion [Al Zayer et al. 2020; Boletsis 2017; Cardoso and Perrotta 2019; Nilsson et al. 2018]. A recent example can be found in [Cannavò et al. 2020], where the authors propose a methodology for evaluating navigation techniques. Different metrics concerning users' performance and experience are discussed. These metrics can be assessed during the execution of a number of navigation tasks in a set of representative scenarios. The proposed methodology is accompanied by a scoring system to provide a ranking of navigation techniques. Instead, in this paper we focus exclusively on how to facilitate the design and fine tuning of navigation techniques with immediate iteration cycles.

3 OUR APPROACH

3.1 Premises

We propose a tool to facilitate the design, testing and fine tuning of navigation techniques. The method is based on the following premises: (a) adapting the navigation technique to a specific scenario can improve the user experience; for example, a narrow cave and a train station might require completely different navigation techniques; (b) adapting the technique to the intended primary task (search for a target, explore a virtual scene, or maneuver obstacles [LaViola Jr et al. 2017]) can also improve usability; (c) adapting the technique to specific user profiles (e.g. art historians) or functional abilities can make the application more inclusive; (d) fine tuning the different parameters and design options that define a navigation technique can make a large difference; very often the devil is in the details, and a poor choice for some parameter (e.g. input sensitivity) can make a navigation technique usable in some specific scenarios but unusable for the rest; (e) all the adaptations above are practical if they can be implemented and tested in an easy way, with immediate iteration cycles, complete control of the mappings, and arbitrary runtime modifications.

3.2 Three-level adaptation of navigation techniques

We distinguish three different abstraction levels at which designers can customize a navigation technique:

- Parameter-level adaptation: the designer chooses a specific navigation technique (e.g. continuous steering) and simply modifies predefined parameters, such as the speed (e.g. 2 m/s). The designer can also specify non-numerical choices, e.g. which button must be used to navigate, but available choices are predefined.
- User-defined adaptation: the designer chooses one of the predefined metaphors (e.g. teleportation), and provides the source code defining the mapping between user input and viewpoint changes. Therefore designers are completely free to specify the mapping, with the only limits imposed by the selected metaphor.
- User-defined metaphor: at this level, designers would be able to implement completely new locomotion metaphors (e.g. mimicking swimming), beyond those already supported by the framework.

Our current prototype supports the first two levels of adaptation. Since the first level requires no programming skills, modifications can be done by the developer while testing a specific scenario, as well as by the end user at runtime. Unfortunately, this level is insufficient e.g. to get complete control on the mapping between user actions and viewpoint changes. We thus also support the second level through custom functions.

3.3 Supported navigation metaphors

Our current prototype supports three major navigation metaphors: free teleportation (any target), teleportation points (predefined targets) and flying/walking (steering-based navigation). Most commercial games for VR headsets provide navigation techniques in one of these three categories. We now describe the basic behavior of these metaphors; related parameters and options for customizing these techniques are described in Section 3.5.

3.3.1 Free-target teleportation. The user points at any target location (constrained to intersect geometry labeled as *floor*) and the application automatically moves the viewpoint to the target location [Bhandari et al. 2018; Bozgeyikli et al. 2016; Mackinlay et al. 1990; Schroeder et al. 2001]. See Figure 1.

3.3.2 Teleportation points. This metaphor is similar to free-target teleportation, but target locations are limited to a predefined collection of targets (Figure 1-right).

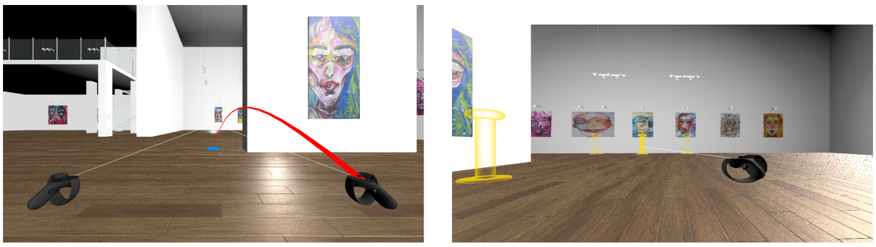
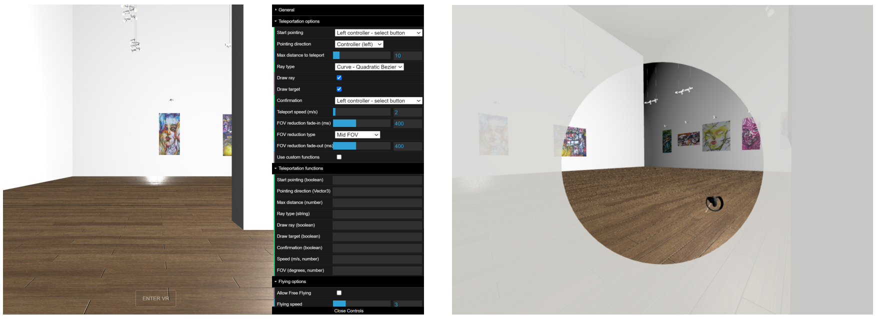


Fig. 1. Left: Teleportation example. Once the target location is confirmed, the viewpoint will move to it. Right: Teleportation example with predefined targets (shown here as yellow columns). Art Gallery model by 3DTwins licensed under CC-BY-4.0.

3.3.3 Walking/flying. This metaphor allows users to walk (grounded navigation) or fly through the virtual environment, using a steering-based technique [Christou and Aristidou 2017; LaViola Jr et al. 2017].

3.4 GUIs for technique adaptation

A key issue for fast experimentation is to let users (and interaction designers) modify the behavior of the base techniques through a GUI. Our prototype includes two different GUIs. The *2D GUI* is available anytime the application is not in VR mode. Since the components of the 2D GUI are part of the Document Object Model (DOM), they support advanced CSS style options and can be interacted with traditional interaction devices (e.g. mouse and keyboard on a desktop device). Conversely, the *VR GUI* is available only when the browser is in VR mode. In this case, the GUI components are actually 3D geometry that must be rendered together with the 3D scene. This limits a bit the type of supported components, as well as how they can be interacted with (e.g. by pointing using the VR controllers).



▸ General

▾ Teleportation options

Start pointing	Left controller - select button
Pointing direction	Controller (left)
Max distance to teleport	10
Ray type	Curve - Quadratic Bezier
Draw ray	✓
Draw target	✓
Confirmation	Left controller - select button
Teleport speed (m/s)	2
FOV reduction fade-in (ms)	400
FOV reduction type	Mid FOV
FOV reduction fade-out (ms)	400
Use custom functions	☐

▾ Teleportation functions

Start pointing (boolean)	
Pointing direction (Vector3)	
Max distance (number)	
Ray type (string)	
Draw ray (boolean)	
Draw target (boolean)	
Confirmation (boolean)	
Speed (m/s, number)	
FOV (degrees, number)	

▾ Flying options

Allow Free Flying	☐
Flying speed	3

Close Controls

Fig. 2. The panel on the margin shows the 2D GUI for configuring teleportation techniques. Right: FOV reduction example. Art Gallery model by 3DTwins licensed under CC-BY-4.0.

3.5 Parameter-level adaptation

3.5.1 Free teleportation. Users and designers can modify the behavior of the base teleportation technique through the GUI. The options that can be adjusted were decided taking into account well-established analyses on the different subtasks involved in selection-based techniques [LaViola Jr et al. 2017]. Figure 2 shows our current prototype's 2D GUI for teleportation customization. We now briefly describe the different options. The first group is concerned with the target selection part:

- Start pointing: which user input initiates the selection (and causes the selection ray and/or selection target to be shown). Available choices include specific buttons of the controllers, as well as an "always active" option.
- Pointing direction: this option specifies how to compute the base pointing direction. Available choices include the controllers' forward direction, the user's head, and the user's torso.
- Max distance to teleport: we allow the user to limit the maximum distance for selectable target positions. This is specially suitable for curved rays (Figure 1).
- Ray type: whether to use a straight line segment or a curved selection ray. Curved rays provide a more equally-spaced mapping between ray orientations and target distances.
- Confirmation: which user input confirms the target selection.

The second group of options refer to the actual teleportation part, as well as FOV reduction to reduce motion sickness. See the accompanying video for a demonstration of specific options.

- Teleport speed (m/s): camera speed towards the target, or 0 for instant teleportation.
- FOV reduction fade-in: time (usually a few milliseconds) to gradually reduce the FOV.
- FOR reduction type: how much the normal FOV must be reduced during gradual teleportation.
- FOV reduction fade-out: time to gradually restore the FOV to its normal value.

3.5.2 Teleportation points. Similarly, our GUIs include options for teleportation with predefined targets. The major difference is that predefined targets are rendered as 3D proxies (e.g. yellow columns in Figure 1). Users must point at these proxies, so the target selection part is essentially a typical 3D object selection task. The options that can be configured are the same as above, but without the *max distance to teleport* and the *ray type* options.

3.5.3 Walking/flying. For the sake of brevity, we only describe walking/flying options related to viewpoint changes.

- Start trigger: which user input (e.g. a button press) initiates the movement.
- Stop trigger: which user input (e.g. a button release) terminates the movement.
- Motion direction: how to compute the forward direction. Available choices include the controllers' forward direction, the user's head, and the user's torso.
- Allow flying: this option determines whether the user is allowed to fly.
- Speed control: how does the user control the speed (constant speed, constant acceleration...).

3.6 Adaptation through custom functions

3.6.1 Motivation. Although the options above allow users to replicate many navigation techniques available for mainstream VR headsets, this adaptation is still limited to a collection of predefined choices. The most innovative part of this work is the idea of allowing users/experimenters to quickly define programmatically their own mappings between user actions (inputs) and navigation control (outputs). The user is free to provide Javascript (JS) code for a collection of metaphor-specific functions defining all aspects of the navigation technique. There is a default implementation for all these functions, so users can implement only a subset of them. These functions were inspired by Computer Graphics *shaders*: special pieces of code that run at specific stages of the graphics pipeline, and that can be customized to achieve different effects. In the same sense that shaders have a well defined task and execution framework, our custom functions also have a very clear purpose, for example, computing the camera speed depending on the application state and the user input.

One can argue that developers and UX designers can always write these functions just as part of their JS modules. However, our application allows these functions to be written directly on specific 2D GUI components. Furthermore, the functions can be edited at anytime and are evaluated at runtime. For example, we can navigate to a specific room, and test different mappings immediately without reloading the application. This method makes testing iteration cycles much shorter and thus allows for a more comprehensive testing of different mappings in varied situations.

3.6.2 Exposed variables. In order to facilitate writing these custom functions, we expose all relevant variables. In some cases, we provide simpler names to simplify the code. For example, instead of camera.quaternion.clone() the user can just write cameraQuaternion.

3.6.3 Teleportation. Here we briefly describe a subset of the custom functions experimenters might provide to define precisely the behavior of teleportation. We indicate the expected JS return type within parentheses. Unless otherwise stated, the functions below are invoked automatically every frame.

- *Start pointing (boolean)*: target indication will be initiated as soon as this function returns true.
- *Pointing direction (Vector3)*: function providing the 3D vector to be used as pointing direction.
- *Confirmation (boolean)*: teleportation target is confirmed as soon as this function returns true.
- *Speed (m/s, number)*: function providing the speed for gradual teleportation.

See the complete list (including FOV reduction behavior) in the 2D GUI components of Figure 2. Notice that custom functions roughly match the breakdown for the different configurable options. Alternative factorizations exist. For example, we could require the user to provide a single *target (Vector3)* function returning the selected target, instead of asking separately for the target selection subtasks. Our factorization though provides multiple advantages. On the one hand, it is close to that of the GUI-adjustable options, and thus easier to understand and remember. On the other hand, the proposed factorization facilitates code re-usability, and simplifies development since empty custom functions take the default behavior. For example, the experimenter might chose the provide a custom function just for the pointing direction while reusing the default behavior for the rest.

3.6.4 Walking/flying. Here we describe the most relevant user-defined functions for walking/flying. Each frame the camera is displaced along the forward direction. The distance is computed as the *speed* times the application-provided *dt* (delta time).

- *Start trigger (boolean)*: camera movement will start as soon as this function returns true.
- *Stop trigger (boolean)*: camera movement will stop as soon as this function returns true.
- *Motion direction (Vector3)*: function providing the 3D vector to be used as forward direction.
- *Speed (number)*: function providing the speed.

4 USE CASES

In this section we discuss several use cases that benefited from the short iteration cycles supported by our tool. In most cases the VR mode is offered in addition to conventional desktop-based or phone-based presentation of the 3D models.

4.1 A Medieval Church with mural paintings: *St. Quirze de Pedret*

The goal was to provide intuitive navigation for art historians to explore the mural paintings of *St. Quirze de Pedret* using a standalone VR headset (Figure 3). The designer used Google Chrome

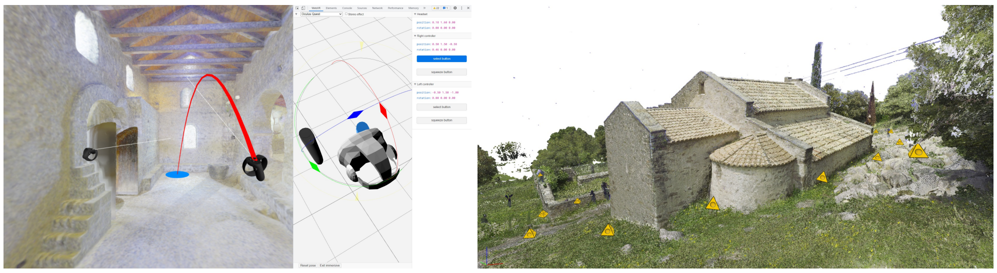


Fig. 3. St. Quirze de Pedret model. The image on the left also shows the WebXR API emulator.

running on a desktop PC for testing different navigation techniques, and an Oculus Quest for fine tuning. While using the desktop PC, we used the *WebXR API Emulator*, a browser extension that enables developers to test WebXR content without using a real XR device by emulating mainstream XR devices and their controllers (Figure 3). The tool allowed us to quickly discard continuous steering techniques due to motion sickness issues. We tested large FOV reduction but this prevented users from seeing the mural paintings above the line of sight (e.g. on the apses), so we found FOV reduction not appropriate during steering-based navigation. We thus opted to default to a free teleportation technique using the hand-held controllers for selecting the target position. Parameters such as maximum target distance and gradual teleportation speed were also fine tuned to the reduced size of the Church.

4.2 *St. Quirze de Pedret* surroundings and architecture

We wanted to enable art historians and the general public to explore the architecture of *St. Quirze de Pedret*, this time emphasizing on the outdoor model. This time free teleportation was tricky because the Church is on the edge of a long and jagged hill-mountain at the foot of the Pyrenees (Figure 3). On the other hand, the scan locations where readily available, so we decided to test and fine tune teleportation using as predefined targets the scan locations, which also guarantee high-quality viewpoints of the model.

4.3 Cultural heritage demos for lab visitors

We aim at providing immersive demonstrations of cultural heritage models for people visiting our VR lab. The priority was to reduce to a minimum the instructions for navigation. An example of adaptation facilitated by our custom functions was to let users use nearly any button of any controller to walk through the scene. A version for fair kiosks automatically detects when controllers are not being used to switch to a hands-free navigation that provides teleportation to targets chosen by gaze-based selection.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have discussed the benefits of a web-based tool for quickly designing, testing and comparing navigation techniques. Our hypothesis is that, if testing iteration cycles are very short, designers can fine tune the default parameters of the chosen navigation technique to the specificity of the 3D environment, the user task, the target users and other contextual issues. Our tool supports the design of inclusive VR experiences in the sense that fast testing allows designers to offer a catalog of preset navigation techniques to cover the largest possible audience, and because the VR GUI allows end users to further customize navigation options to suit their needs. As future work, our first priority is to evaluate the usability of our framework with designers outside our group.

We want to evaluate if the development of custom functions is fast and suitable for developers distinct from the authors. We also wish to add further high-level methods to facilitate writing such functions with a few lines of JS code.

ACKNOWLEDGMENTS

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness and FEDER under grant TIN2017-88515-C2-1-R, by EU Horizon 2020, JPICH Conservation, Protection and Use initiative (JPICH-0127) and the Spanish Agencia Estatal de Investigación, grant PCI2020-111979 *Enhancement of Heritage Experiences: the Middle Ages; Digital Layered Models of Architecture and Mural Paintings over Time (EHEM).* Art Gallery model by 3DTwins licensed under CC-BY-4.0.

REFERENCES

Majed Al Zayer, Paul MacNeilage, and Eelke Folmer. 2020. Virtual Locomotion: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 26, 6 (2020), 2315–2334. https://doi.org/10.1109/TVCG.2018.2887379

Jiwan Bhandari, Paul R MacNeilage, and Eelke Folmer. 2018. Teleportation without Spatial Disorientation Using Optical Flow Cues.. In *Graphics interface.* 162–167.

Costas Boletsis. 2017. The new era of virtual reality locomotion: A systematic literature review of techniques and a proposed typology. *Multimodal Technologies and Interaction* 1, 4 (2017), 24.

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.* 205–216.

Alberto Cannavò, Davide Calandra, F Gabriele Pratticò, Valentina Gatteschi, and Fabrizio Lamberti. 2020. An evaluation testbed for locomotion in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 27, 3 (2020), 1871–1889.

Jorge CS Cardoso and André Perrotta. 2019. A survey of real locomotion techniques for immersive virtual reality applications on head-mounted displays. *Computers & Graphics* 85 (2019), 55–73.

Chris G Christou and Poppy Aristidou. 2017. Steering versus teleport locomotion for head mounted displays. In *International conference on augmented reality, virtual reality and computer graphics.* Springer, 431–446.

Christopher G Coogan and Bin He. 2018. Brain-computer interface control in a virtual reality environment and applications for the internet of things. *IEEE Access* 6 (2018), 10840–10849.

Massimiliano Di Luca, Hasti Seifi, Simon Egan, and Mar Gonzalez-Franco. 2021. *Locomotion Vault: The Extra Mile in Analyzing VR Locomotion Techniques.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445319

Matt Dombrowski, Peter A Smith, Albert Manero, and John Sparkman. 2019. Designing inclusive virtual reality experiences. In *International Conference on Human-Computer Interaction.* Springer, 33–43.

Moria Engberg, Jay David Bolter, and Blair Maclntyre. 2018. RealityMedia: an experimental digital book in WebXR. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct).* IEEE, 324–327.

Ajoy S Fernandes and Steven K Feiner. 2016. Combating VR sickness through subtle dynamic field-of-view modification. In *2016 IEEE symposium on 3D user interfaces (3DUI).* IEEE, 201–210.

Leonardo Gomes, Olga Regina Pereira Bellon, and Luciano Silva. 2014. 3D reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters* 50 (2014), 3–14.

Behrang Keshavarz, Bernhard E Riecke, Lawrence J Hettinger, and Jennifer L Campos. 2015. Vection and visually induced motion sickness: how are they related? *Frontiers in psychology* 6 (2015), 472.

Eugenia M Kolasinski. 1995. *Simulator sickness in virtual environments.* Vol. 1027. US Army Research Institute for the Behavioral and Social Sciences.

Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. 2017. *3D user interfaces: theory and practice.* Addison-Wesley Professional.

Kyungmin Lim, Jaesung Lee, Kwanghyun Won, Nupur Kala, and Tammy Lee. 2021. A novel method for VR sickness reduction based on dynamic field of view processing. *Virtual Reality* 25, 2 (2021), 331–340.

Jock D Mackinlay, Stuart K Card, and George G Robertson. 1990. Rapid controlled movement through a virtual 3D workspace. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques.* 171–176.

Blair Maclntyre and Trevor F Smith. 2018. Thoughts on the Future of WebXR and the Immersive Web. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct).* IEEE, 338–342.

Martez Mott, Edward Cutrell, Mar Gonzalez Franco, Christian Holz, Eyal Ofek, Richard Stoakley, and Meredith Ringel Morris. 2019. Accessible by design: An opportunity for virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct).* IEEE, 451–454.

Niels Christian Nilsson, Stefania Serafin, Frank Steinicke, and Rolf Nordahl. 2018. Natural walking in virtual reality: A review. *Computers in Entertainment (CIE)* 16, 2 (2018), 1–22.

Giovanni Pintore, Claudio Mura, Fabio Ganovelli, Lizeth Fuentes-Perez, Renato Pajarola, and Enrico Gobbetti. 2020. State-of-the-art in Automatic 3D Reconstruction of Structured Indoor Environments. *Computer Graphics Forum* 39, 2 (2020), 667–699.

Sharif Razzaque, David Swapp, Mel Slater, Mary C Whitton, and Anthony Steed. 2002. Redirected walking in place. In *EGVE*, Vol. 2. 123–130.

Ralph Schroeder, Avon Huxor, and Andy Smith. 2001. Activeworlds: geography and social interaction in virtual reality. *Futures* 33, 7 (2001), 569–587.

Mel Slater, Anthony Steed, and Martin Usoh. 1995. The virtual treadmill: A naturalistic metaphor for navigation in immersive virtual environments. In *Virtual environments' 95*. Springer, 135–148.

Frank Steinicke, Gerd Bruder, Luv Kohli, Jason Jerald, and Klaus Hinrichs. 2008. Taxonomy and implementation of redirection techniques for ubiquitous passive haptic feedback. In *2008 International Conference on Cyberworlds*. IEEE, 217–223.

Evan A Suma, Gerd Bruder, Frank Steinicke, David M Krum, and Mark Bolas. 2012. A taxonomy for deploying redirection techniques in immersive virtual environments. In *2012 IEEE Virtual Reality Workshops (VRW)*. IEEE, 43–46.

Caglar Yildirim. 2019. Don't make me sick: investigating the incidence of cybersickness in commercial virtual reality headsets. *Virtual Reality* (2019), 1–9.