# Simulation of High-Visual Quality Scenes in Low-Cost Virtual Reality

Matías N. Selzer, M. Luján Ganuza, Dana K. Urribarri, Martín L. Larrea, and Silvia M. Castro

VyGLab Research Laboratory, UNS-CICPBA, Bahía Blanca, 8000, Argentina.
Institute of Computer Sc. and Eng. (CONICET-UNS), Bahía Blanca, Argentina.
{matias.selzer,mlg,dku,mll,smc}@cs.uns.edu.ar
http://vyglab.cs.uns.edu.ar

**Abstract.** With the increasing popularity of virtual reality, many video games and virtual experiences with high-visual quality have been developed recently. Virtual reality with a high-quality representation of scenes is still an experience linked to high-cost devices. There are currently low-cost virtual reality solutions by using mobile devices, but in those cases, the visual quality of the presented virtual environments must be simplified for running on mobile devices with limited hardware characteristics. In this work, we present a novel Image-Based Rendering technique for low-cost virtual reality. We have conducted a performance evaluation of three mobile devices with different hardware characteristics. Results show that our technique represents high-visual quality virtual environments with considerably better performance compared to traditional rendering solutions.

**Keywords:** Virtual Reality, Low-Cost VR, Navigation, Image-Based Rendering

## 1 Introduction

Virtual Reality (VR) is a technology already established in many different markets and platforms around the globe. One of the main factors in the rise of VR was the possibility of experiencing VR with low-cost mobile devices. This low-cost VR, when properly implemented, can not only enable people to experience this technology by using their mobile phones but also brings a new world of possibilities to those users who cannot afford a high-end VR system.

The games and experiences designed for this type of VR are usually graphically very simple, and rendering complex 3D scenes in these devices is a challenging problem. This is partly due to mobile device limitations in hardware and graphical processing (compared to modern gaming computers).

In this paper, we introduce a novel technique to display high-visual quality 3D scenes in low-cost VR devices. More specifically, the work presented here yields the following benefits and contributions:

- Our technique simulates a 3D environment by using a spherical panorama 3D texture matrix and dynamic image warping.
- We designed a novel image warping technique that enables a smooth transition between image points.
- Our technique was compared to traditional rendering of the same scene, obtaining better performance for multiple hardware configurations.
- Our technique allows a fully 3D VR navigation.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. Section 3 presents the details of the proposed technique and the proposed algorithm description. The experimental setup is described in Section 4. The experiment results are shown in Section 5 and discussed in Section 6. Finally, we draw our conclusions and point out the future work in Section 7.

## 2 Related Work

In recent years there has been tremendous growth in the number and variety of GPU-intensive mobile applications, enabling users to interact and navigate in high-visual quality virtual environments. To get a suitable system for mobile phones, many approaches vary both in the degree of realism of the virtual environment and the navigation technique.

The most popular render techniques for this purpose are those belonging to traditional geometry rendering and those oriented to image-based rendering (IBR). In the first case, the most viable alternatives to having high-visual quality virtual worlds are those that integrate progressive meshes, caching and restricted objects resolution techniques, thus allowing to overcome, to different degrees, the limitations of mobile devices in regards to processing, memory and power consumption [1–5]. In those cases, rendering high-visual quality 3D scenes on limited mobile devices obtained a very low performance.

Currently, Image-Based Rendering (IBR) is emerging as the most promising approach since the rendering process requires less computational resources than traditional geometry rendering. These techniques emerged in the late 1990s to overcome the severe limitations that exist for the representation of photo-realistic 3D scenes in real-time [6–8].

While more recent methods, running on PC, can synthesize new views based on captured panoramas from both video and images, the new generated views do not allow free navigation. Furthermore, they are based on the capture of a few panoramas and the synthesis of the intermediates is done by computationally intensive warping and interpolation algorithms [9, 10]. More recent approaches based on image interpolation for free navigation were proposed in the literature [11–15] but they only run on PC.

To represent high-visual quality 3D scenes with mobile devices, IBR is more suitable because it uses images as input and the rendering cost does not depend on the scene complexity, but on the final image resolution. Different approaches have been proposed that provide a variety of image types and qualities [6, 16]. More recently approaches, based on high-quality synthetic scenes generated in the server, have been presented in [17, 18]. However, in all these cases, a significant latency is observed.

## 3 Our Proposal

In this article, we present a novel Image-Based Rendering technique based on spherical panoramic images to simulate high-visual quality 3D environments for low-cost VR devices. An overview of the technique's main components and their interaction is depicted in Figure 1. This process is continuously repeating as the user is moving, creating the sensation of a smooth transition through the virtual environment. All these components are detailed next.

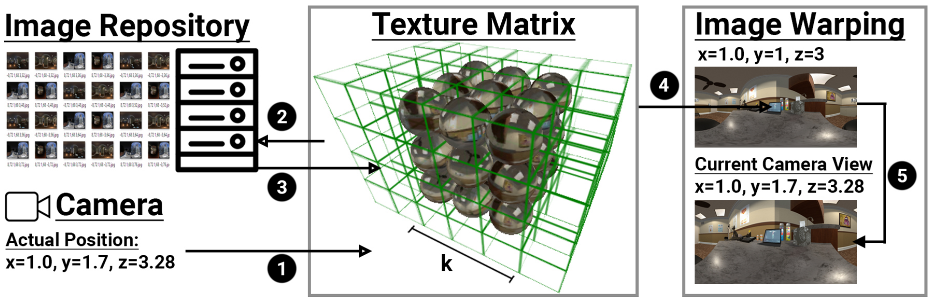


**Fig. 1.** Technique overview. (1) The camera sends its current position to the Texture Matrix component. (2) A set of images is requested to the Image Repository. (3) The Image Repository sends the requested images. (4) The nearest image that corresponds to the camera current position is sent to the Image Warping component. (5) The Image Warping component calculates and generates the image that corresponds to the camera exact current position.

### 3.1 Requirements

Our technique requires a set of 360° panoramic images captured from the original 3D environment to be simulated. The environment has to be sampled by capturing 360° panoramic images, each one from a different and specific position. If the environment has a dimension of $dim_x$ in the x-axis, $dim_y$ in the y-axis, and $dim_z$ in the z-axis, and all images are separated by a distance $\Delta$, then all the images can be represented as an abstract 3D matrix with dimension $N \times M \times R$ as shown in Figure 3 (right), where $N = dim_x/\Delta$, $M = dim_y/\Delta$, and $R = dim_z/\Delta$. These 360° panoramic images are then stored in the Image Repository, which provides the images to the Texture Matrix component of our technique.

### 3.2 Local Image Management - Texture Matrix

The Texture Matrix component is in charge of loading, storing, and managing the 360° panoramic images recovered from the Image Repository. The number of cells in the Texture Matrix depends on the size of the original 3D environment and the distance $\Delta$ between samples. The Texture Matrix dimension is $N \times M \times R$, where $N$, $M$, and $R$ are specified in Section 3.1. Therefore, this matrix is eventually capable of storing every sample of the scene.

In our approach, we provide efficient management of images. At any time, the Texture Matrix maintains a small number of images loaded in memory that correspond to the samples captured in a delimited neighborhood regarding the current position of the camera. As the camera moves, the Texture Matrix is updated. Finally, the matrix is regularly cleaned by removing those images captured in positions located at a certain distance of $k$ from the current position of the camera.

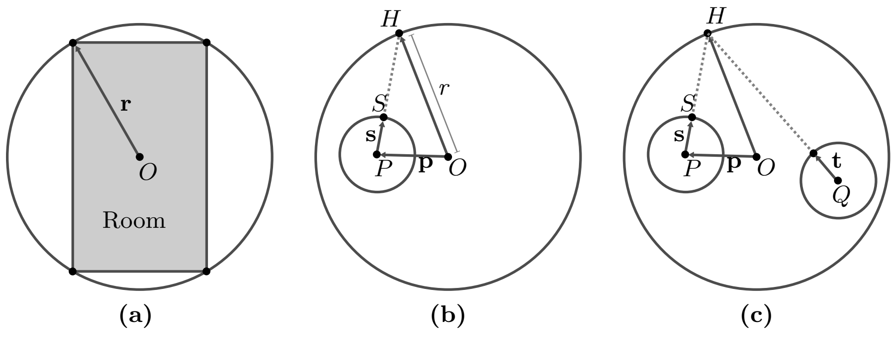


**Fig. 2.** Cross-section of a room centered at $O$ and two unitary spheres centered at $P$ and $Q$. (a) Cross-section of the sphere with center $O$ and radius $r$ containing a room. (b) Point $H$ over the room corresponding to the projection $S$. The unitary sphere is centered at $P$, where the equirectangular panoramic image $\mathcal{I}_P$ was taken. (c) Projection of point $H$ over the unitary sphere centered at $Q$, where no image was taken.

### 3.3 Image Warping

In order to avoid the user noticing jumps between the samples, we implemented a function to smoothen that transition. Let us assume that two contiguous spherical panoramic images $\mathcal{I}_A$ and $\mathcal{I}_B$ were taken at points $A$ and $B$, then if the user stands at one of those points the system shows the corresponding image. However, while the user is moving from point $A$ to $B$ the system should show an intermediate image that smoothly switches from $\mathcal{I}_A$ to $\mathcal{I}_B$.

We consider that the spherical panoramic images correspond to an equirectangular projection of a circular room that fully contains the original one (see Figure 2a). Let $O$ be the center of the circle of radius $r$ that contains the room, $Q$ be the point where the user stands and $P$ be the point where the nearest panoramic image $\mathcal{I}_P$ was taken. The strategy is to warp $\mathcal{I}_P$ to match the panoramic image that should have been taken from the point of view of $Q$. If $(u, v) \in [0, 1]^2$ is a texture coordinate over the panoramic image $\mathcal{I}_P$, we invert the projection to obtain the point $S$ over a unitary sphere centered at $P$ that projects onto $(u, v)$ (see Figure 2b). Since the panoramic image is the result of a equirectangular projection, the vector $\mathbf{s}$ between $P$ and $S$ is: $\mathbf{s} = (x, y, z) = (\sin(\phi)\cos(\theta), \sin(\theta)\sin(\phi), \cos(\phi))$, where $\theta = 2u\pi$ and $\phi = v\pi$. Now, let $H$ be the point over the room of radius $r$ that projects on $S$. Then, $H = k\mathbf{s} + \mathbf{p}$ where $\mathbf{p} = Q - O$, $|k\mathbf{s} + \mathbf{p}| = r$ and $k = \frac{-(\mathbf{p}\cdot\mathbf{s})+\sqrt{(\mathbf{p}\cdot\mathbf{s})^2-|\mathbf{s}|^2(|\mathbf{p}|^2+r^2)}}{|\mathbf{s}|^2}$.

To warp the image to match the point of view of $Q$, we find the projection $\mathbf{t}$ of $H$ over the unitary sphere centered at $Q$ (see Figure 2c), $\mathbf{t} = \frac{H-Q}{|H-Q|}$ Finally, the 2D projection $(u, v)$ of $\mathbf{t} = (x, y, z)$ over the equirectangular texture is $(u, v) = (\frac{\theta}{2\pi}, \frac{\phi}{\pi})$ where $\theta = \text{atan2}(y, x)$ and $\phi = \text{atan2}(\sqrt{x^2 + y^2}, z)$.

## 4 Experimental Setup

To study the benefits of our technique compared to a traditional 3D rendering approach, a performance experiment was conducted. Since performance depends on the used hardware, we designed an experiment to test our technique with three different hardware configurations. Here, the same scenario is presented in two different treatment conditions: a traditional rendering approach and by using our technique. This section details the experiment designed to evaluate our technique.

### 4.1 Virtual Environment

We used "Doctor's Office" scenario[19] as the 3D virtual scenario for the test. This scenario is designed to run in Unity3d[20] in which we developed our technique. It consists of a representation of a doctor's office with 2.1 million triangles, 2.3 million vertices, and it is rendered with high-level graphics techniques.

### 4.2 Images Dataset

The image dataset was created by using Unity360ScreenshotCapture[21]. This tool generates a 360° panoramic image based on the position of the virtual camera. The whole dataset is created automatically by iterating along the desired volume. Figure 3 shows an example of the sampling process.

### 4.3 Hardware Configurations

The proposed technique is specifically designed for low-cost VR, i.e., by using mobile phones. Since we want to show that our technique works for both low-end and high-end mobile phones, the current study evaluates and compares

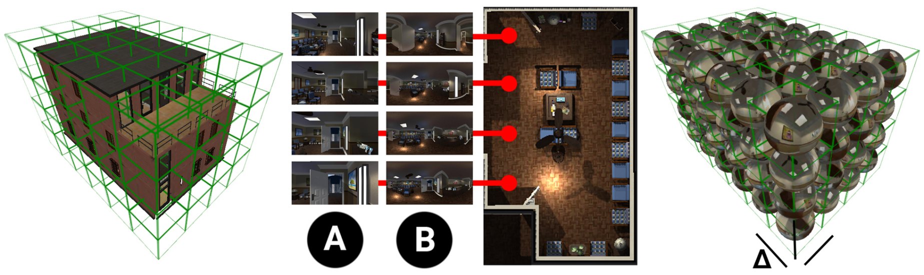


**Fig. 3.** Image capture example. For each specified camera position, the application captures a 360° panoramic screenshot. (Left) The whole virtual environment is sampled. (Center) Column A shows the camera current view. Column B shows the 360° picture captured at that position. (Right) All the environment is sampled and an image corresponding to each cell of an abstract 3D matrix is generated.

the performance of our technique in the three different hardware configurations presented in Table 1.

In the rest of the paper, we use the letter H (high-end) to refer to the Le Eco Max 2 mobile phone, the letter M (mid-end) to refer to the Motorola Moto G mobile phone, and the letter L (low-end) to refer to the Google Nexus 7 tablet.

**Table 1.** Hardware specification of the three mobile devices used in the experiment.

| **Name** | Le Eco Max 2 | Motorola Moto G 2 | Asus Google Nexus 7 |
|---|---|---|---|
| **OS** | Android 6.0.1 / API-23 | Android 6.0 / API-23 | Android 6.0.1 / API.23 |
| **Screen** | 2560x1440@59Hz | 1280x720@60Hz | 1280x800@60Hz |
| **Graphics API** | OpenGL ES 3.1 | OpenGL ES 3.0 | OpenGL ES 2.0 |
| **GPU** | Adreno (TM) 530 | Adreno (TM) 305 | NVIDIA Tegra 3 |
| **VRAM** | 2048 MB | 256 MB | 256 MB |
| **Max texture size** | 16384px | 4096px | 2048px |
| **Shader level** | 50 | 35 | 30 |
| **CPU** | Quad-core 2.1 GHz Snapdragon-820 | Quad-core 1.2 GHz Cortex-A7 | Quad-core 1.2 GHz Cortex-A9 |
| **RAM** | 5774 MB | 890 MB | 971 MB |

## 4.4 Variables

In this study, we consider one independent variable, which is the technique used to render the virtual environment. This variable takes two values and the two treatment conditions of this study are explained as follows:

- Treatment Condition 1 (TC1) - Traditional Rendering: The virtual scenario is exported and rendered with a traditional graphic configuration.
- Treatment Condition 2 (TC2) - Our Technique: The application only contains the required elements for our technique.

The system response time is the time between the user's actions and the perceived response [22]. Several studies suggest that this latency degrades the sensation of presence [23, 24], and it is suggested that the latency should not be greater than 100 ms [25]. However, more recent studies suggest that latency should be under 20 ms [26]. The frames-per-second (FPS) are a measure of how many frames the hardware can render in one second. Since the FPS represent a good measure of performance, this measure has been widely used to compare the performance between two or more different graphical techniques. For this reason, this study uses FPS as a measure of performance in both treatment conditions.

### 4.5 Procedure

Even though both treatment conditions allow the user to freely navigate the virtual environment, to compare the performance between both treatment conditions, a specific path through the virtual environment was defined. This path has a total distance of 30 meters and it is shown in Figure 4.

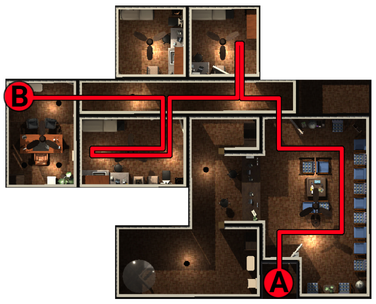


**Fig. 4.** The path followed by the virtual camera in the experiment. The camera travels from point A to point B in 300 seconds.

The duration time for the camera to get to point B is 5 minutes (or 300 seconds). At every 0.5 seconds, the current FPS value is recorded. Thus, for each test run, a total of 600 samples are generated. This test was performed by the three different hardware configurations in both treatment conditions. The results are presented in the next section.

## 5 Results

The results of the performance experiment among the treatment conditions are depicted in Figure 5. Figure 6 presents the average, lower, and higher FPS scores obtained for both treatment conditions for the three hardware configurations. Based on these values, the H configuration obtained a performance improvement of 514%; the M configuration a performance improvement of 682%; and the L configuration a performance improvement of 352%.

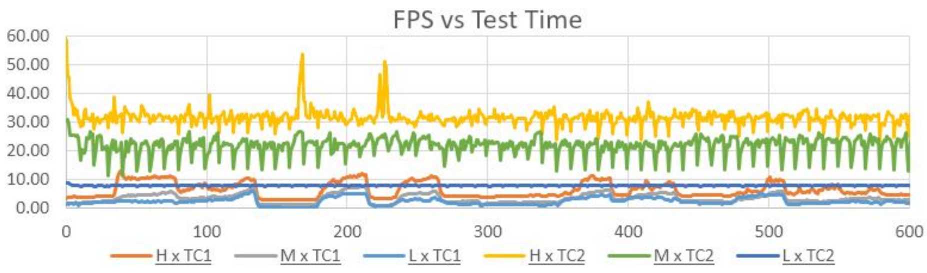
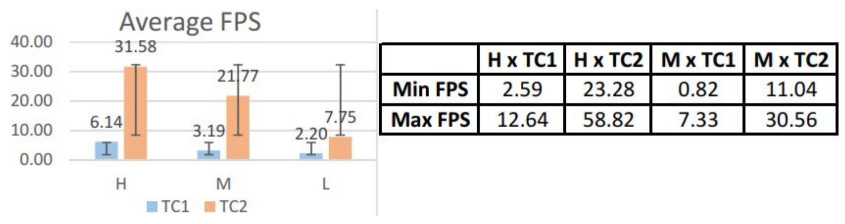


**Fig. 5.** Performance Results Overview. The plot's horizontal axis depicts the measured FPS during the 600 time samples of the total duration of the trial (300 seconds, one sample every 0.5 seconds) for each one of the hardware configurations. The histogram shows the average FPS of each trial.

To test the limits of the image warping effect beyond the 4 cm distance between samples, some informal user tests were also performed. Results showed that using a distance between samples up to 40 cm still proves an acceptable user experience. Higher values produced higher deformations in the images that induced cybersickness symptoms in the user. The implications of these results are discussed in the next section.



| | H x TC1 | H x TC2 | M x TC1 | M x TC2 |
|---|---|---|---|---|
| Min FPS | 2.59 | 23.28 | 0.82 | 11.04 |
| Max FPS | 12.64 | 58.82 | 7.33 | 30.56 |

**Fig. 6.** (Left) Average FPS scores. (Right) Lower and Higher FPS scores comparison grouped by hardware configuration.

## 6 Discussion

The results obtained in the performance test are remarkable. For each one of the hardware configurations, there is a clear difference in FPS measured during the test. Each one of the hardware configurations obtained a performance improvement of at least 352% with an average FPS increment of 516%. These results suggest that our technique outperforms the traditional rendering technique regardless of the used mobile phone. This performance improvement was expected since a traditional rendering approach invests a lot of resources on rendering vertices, polygons, meshes, materials, calculating dynamic illumination and shadows, among other things. In our technique, on the contrary, all such things are implicitly included in the images used to simulate the 3D environment.

Figure 6 shows the higher and lower FPS values obtained of each treatment condition and for each hardware configuration. For each one of the used mobile phones, the higher FPS value obtained while using traditional rendering was still lower than the lower value obtained by using our technique. If for example, we consider the middle hardware configuration used in this study, the average FPS value measured during the traditional rendering test was 3.19, which is equivalent to 1 image every 310 ms approximately. That is something unfeasible for VR in which the images presented to the user have to be displayed as fast as possible. On the contrary, by using our technique, the average measured FPS value was 21.77, which is equivalent to 1 image every 46 ms approximately. These results are more suitable for VR and satisfy the suggested values.

## 7 Conclusions and Future Work

This study presents a new Image-Based Rendering technique specially designed to enable the representation of high-quality virtual environments in low-cost VR devices. This work evaluated the performance of our technique compared to a traditional rendering approach. For this reason, image size optimizations were not investigated. Future work should, therefore, evaluate image optimizations for improving the performance.

One of the main benefits of our technique is that it does not depend on the visual quality or complexity of the virtual environment. Furthermore, this technique will also work with real-world pictures. If we are able to take 360° panoramic spherical pictures of the real world separated by a defined distance, this technique will allow virtual navigation through a total realistic environment. Future work should investigate the use of this kind of image dataset.

## References

1. Klaus Engel, Ove Sommer, and Thomas Ertl. A framework for interactive hardware accelerated remote 3d-visualization. In *Data visualization 2000*, pages 167–177. Springer, 2000.
2. JianPing Ma, Qiang Chen, Bo Chen, and Hong Wang. Mobile 3d graphics compression for progressive transmission over wireless network. In *11th IEEE Int. Conf. on CAD and Computer Graphics*, pages 357–362. IEEE, 2009.
3. Martin Isenburg and Peter Lindstrom. Streaming meshes. In *VIS 05. IEEE Visualization, 2005.*, pages 231–238. IEEE, 2005.
4. Zun Shen, Juan Liu, Yuhui Zheng, and Lu Cao. A low-cost mobile vr walkthrough system for displaying multimedia works based on unity3d. In *14th Int. Conf. on Computer Science & Education (ICCSE)*, pages 415–419. IEEE, 2019.
5. Ghada Fathy, Hanan Hassan, Walaa Sheta, and Reem Bahgat. Efficient framework for mobile walkthrough application. *Pervasive Mob. Comput.*, 18:40–54, 2015.
6. Shenchang Eric Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *Proc. of the 22nd annual Conf. on Comp. graphics and interactive techniques*, pages 29–38, 1995.

7. Yuval Noimark and Daniel Cohen-Or. Streaming scenes to mpeg-4 video-enabled devices. *IEEE Computer Graphics and Applications*, 23(1):58–64, 2003.
8. Jimmy Chim, Rynson WH Lau, Hong Va Leong, and Antonio Si. Cyberwalk: a web-based distributed virtual walkthrough environment. *IEEE T. Multimedia*, 5(4):503–515, 2003.
9. Adarsh Kowdle, Sudipta N Sinha, and Richard Szeliski. Multiple view object cosegmentation using appearance and stereo cues. In *European Conf. on Computer Vision*, pages 789–803. Springer, 2012.
10. Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013.
11. Christian Lipski, Christian Linz, Kai Berger, Anita Sellent, and Marcus Magnor. Virtual video camera: Image-based viewpoint navigation through space and time. In *Computer Graphics Forum*, volume 29, pages 2555–2568. Wiley Library, 2010.
12. Timo Stich, Christian Linz, Christian Wallraven, Douglas Cunningham, and Marcus Magnor. Perception-motivated interpolation of image sequences. *ACM Transactions on Applied Perception (TAP)*, 8(2):1–25, 2011.
13. Yanxiang Zhang and Ziqiang Zhu. Walk-able and stereo virtual tour based on spherical panorama matrix. In *Int. Conf. on Augmented Reality, Virtual Reality and Computer Graphics*, pages 50–58. Springer, 2017.
14. Feng Dai, Chen Zhu, Yike Ma, Juan Cao, Qiang Zhao, and Yongdong Zhang. Freely explore the scene with 360°field of view. In *IEEE Conf. on Virtual Reality and 3D User Interfaces (VR)*, pages 888–889. IEEE, 2019.
15. Chao Liu, Susumu Shibusawa, and Tatsuhiro Yonekura. A walkthrough system with improved map projection panoramas from omni directional images. In *7th Int. Conf. on Ubiquitous Intelligence & Computing and 7th Int. Conf. on Autonomic & Trusted Computing*, pages 45–51. IEEE, 2010.
16. Yu Lei, Zhongding Jiang, Deren Chen, and Hujun Bao. Image-based walkthrough over internet on mobile devices. In *Int. Conf. on Grid and Cooperative Computing*, pages 728–735. Springer, 2004.
17. Juergen Doellner, Benjamin Hagedorn, and Jan Klimke. Server-based rendering of large 3d scenes for mobile devices using g-buffer cube maps. In *Proc. of the 17th Int. Conf. on 3D Web Technology*, pages 97–100, 2012.
18. Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. Proxy-guided image-based rendering for mobile devices. In *Computer Graphics Forum*, volume 35, pages 353–362. Wiley Library, 2016.
19. Brick Project Studio. Doctorsoffice, 2020. http://sojaexiles.com/, online July 2020.
20. Unity. Unity3d, 2020. https://unity.com, online July 2020.
21. yasirkula. Unity3d, 2020. https://github.com/yasirkula/, online July 2020.
22. Jonathan Freeman, Jane Lessiter, and Wijnand IJsselsteijn. An introduction to presence: A sense of being there in a mediated environment. *The Psychologist*, 14:190–194, 2001.
23. Stephen R Ellis, Nancy S Dorighi, Brian M Menges, Bernard D Adelstein, and Richard H Jacoby. In search of equivalence classes in subjective scales of reality. *Advances in human factors/ergonomics*, pages 873–876, 1997.
24. Richard M Held and Nathaniel I Durlach. Telepresence. *Presence: Teleoperators & Virtual Environments*, 1(1):109–112, 1992.
25. Nathaniel I Durlach. Virtual reality-scicntific and techonorogical challenges. *National Academy Press*, pages 213–220, 1995.
26. Kjetil Raaen and Ivar Kjellmo. Measuring latency in virtual reality systems. In *Int. Conf. on Entertainment Computing*, pages 457–462. Springer, 2015.