Performance Optimisations for Rendering Portals in Virtual Reality

Milan van Zanten
University of Basel
milan.vanzanten@unibas.ch

November 1, 2022

1 Introduction

There are several uses for portals in computer graphics including determining the visibility of parts of a 3D scene[5], dividing a scene up into separate areas, rendering a mirror surface, or as traversable portals that can be seen and moved through. These uses can be broadly categorised as either an optimisation technique or a rendering trick. The former two of the mentioned applications are used to determine geometry that can be ignored when rendering a scene and speed up the process. Mirror surfaces and traversable portals though are effects purposefully implemented in an application to benefit the experience.

There are already many implementations of traversable portals in media like video games or architectural visualisations[2]. In this paper we will focus on an application of traversable portals concerning the space limitations in a virtual reality (VR) experience.

One of the main challenges when implementing VR applications is immersion, since errors in tracking and latency are noticed particularly strong[1]. In an effort to maximise immersion, most of VR has moved from seated experiences with movement limited to three degrees of freedom (just rotation) to "room-scale" tracking. Here, in addition to the rotation of the VR headset, the user's position is tracked either via external devices with fixed positions or by cameras that analyse the surroundings and use computer vision algorithms to determine the position. With the added positional tracking, the six degrees of freedom allow the user to move around the room freely. Thus, the only limitation now is the available space. To move around virtual worlds larger than the available space, several different methods have been developed[3]. Examples include head-directed locomotion, point & teleport and more. Indisputably though, the technique that preserves immersion the most is actual walking inside the real space.

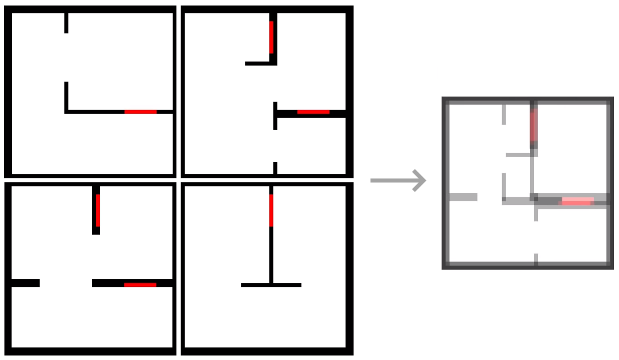
Figure 1: Four rooms with portals (in red) can all be accessed without leaving the smaller real space[4].

A recent method to circumvent the space limitations of walking inside a real space is the concept of impossible spaces. Overlapping rooms are connected through portals into a single space many times larger than the initial rooms themselves. If such an arrangement is made while factoring in the real available space, the whole virtual space can be accessed by passing through the portals. An example layout can be seen in Figure 1.

To allow for such impossible spaces to exist, the aforementioned portals are necessary. When viewed, they show what the user would be seeing through the portal in the other room, and if a user crosses the plane of a portal, they are transported to the connecting one. In virtual reality, implementing such a portal system poses some extra challenges.

Each portal requires rendering an additional viewpoint. When rendering the portals in VR, where each eye is rendered by its own camera, there are now two additional viewpoints to render from. In the naive case where each viewpoint is rendered the same, we produce quadruple the amount of work compared to a basic non-VR scene without portals. We will present two optimisations that can reduce the rendering time and analyse their impact. The first optimisation — using the stencil buffer to only render what is seen through the portal — is concerned with improving the rendering time of portals in general. In contrast, the second optimisation improves the overall performance of rendering VR by not pushing the whole scene to the GPU twice and rendering a texture for each eye, but rather rendering both eyes onto a single texture.

The way a VR scene is rendered also raises the question of how to handle the teleportation of the user. For example, consider the centre point between the eyes that could be used to decide when the portal plane has been crossed. If the user view the portal from an angle, they could clip through the portal with one eye when they enter it2. This could be solved by transporting each eye separately whenever it passes through the portal, but that idea conflicts with one of the performance optimisations we will discuss in the first part.

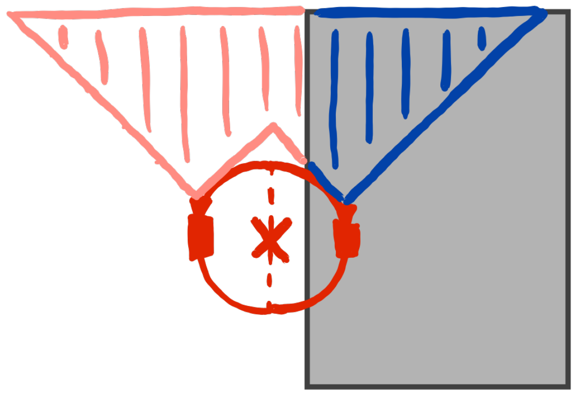

Figure 2: What should the right-eye camera render if it is inside the portal wall (in grey), but the centre of the head (in red) has not crossed the portal plane? If nothing is done, the blue part of the user's field of view would not render the next room, but whatever is inside or behind the portal wall.

References

[1] Michael Abrash. Why virtual reality is hard (and where it might be going). *Game Developers Conference 2013*, 2013.

[2] Daniel G Aliaga and Anselmo A Lastra. Architectural walkthroughs using portal textures. In *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, pages 355–362. IEEE, 1997.

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

[4] Daniel Lochner. Vr natural walking in impossible spaces. *Motion, Interaction and Games (MIG '21)*, 2021.

[5] Nick Lowe and Amitava Datta. A new technique for rendering complex portals. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):81–90, 2005.