Interactive Augmented Virtuality
Project Report

Marlene Mayr

February 8, 2021

Abstract

In this project, the integrated cameras of modern virtual reality headsets are used to create windows into the real world. This integration enables common interactions with everyday objects while being present in a virtual environment. For example, it is possible to draw on a whiteboard or optimize an extended reality application in real-time on the computer.

Contents

Chapter 1

Introduction

As extended reality (XR) is becoming increasingly popular among consumers, several variations of the technology are emerging in research. Some of them are entirely new concepts, while many others have been around for some decades. These former findings are reused in many applications due to evolved hardware.

Augmented reality (AR) displays digital content on top of the real world. This way, users can still interact with their real surroundings and familiar objects that are not part of the application. On the other hand, virtual reality (VR) captures digital content in a virtual surrounding. With this technology, users are usually more immersed, and there is more potential to use the virtual space. Augmented virtuality (AV) can be used to combine both worlds. This project uses a superimposed camera feed to provide windows into the real world. The goal is to interact with real-world objects through this method. For this purpose, the HTC Vive Pro is used. This head-mounted-display (HMD) accesses the tracking cameras for pass-through AR.

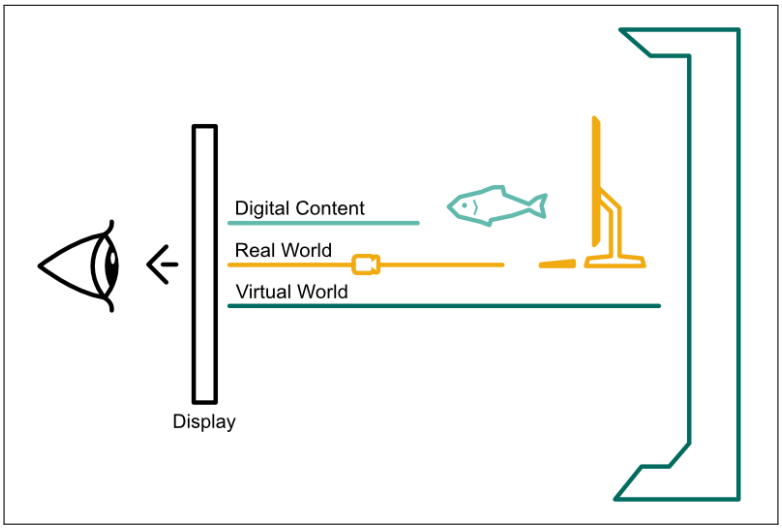

Figure 1.1: Augmented virtuality combines the real and the virtual world. This illustration depicts what is rendered to the display.

Chapter 2

Augmented Virtuality

Augmented virtuality is the core of the project. Its technical setup connects the headset, the real and the virtual world.

2.1 Technology Stack

All the tools needed for augmented virtuality exist in theory but become incompatible and slow the more specific the application is. The project is built with Unity as its base. Several plugins and SDKs have been added throughout the process.

Steam VR provides a framework for the whole tracking system of the HMD. On execution, Unity connects the rendered output to the Steam VR application, which also returns information about the tracking to Unity via an additional plugin.

SRWorks is an SDK and a Runtime provided by HTV Vive to access the device's cameras. The SRWorks Runtime integrates with Steam VR, extracts the cameras' feeds and passes them into Unity to the respective SRWorks plugin.

Open Broadcaster Software (OBS) captures the computer screen and connects it to Unity as a virtual camera. This input is mapped to a texture in the running AV application.

2.2 Render Pipeline

The core structure in Unity and its use in rendering is illustrated in figure 2.1. The Steam VR Unity plugin retrieves the headsets transform and adjusts the position of the virtual camera. This camera is similar to a regular VR setup in Unity but does not render directly to the HMD. Instead, its output is rendered onto a render texture. This Unity-specific texture can be set as any camera's target, is usable in materials and in this case serves as an additional render pass. Simultaneously, SRWorks captures the camera content and forwards it to the respective SRWorks plugin in Unity. There follows an algorithm to undistort the image because the cameras are wide-angled for better tracking. The undistorted image is then also added to the render texture. This render texture is placed in front of another virtual camera in Unity. This is the camera which finally renders to the device.

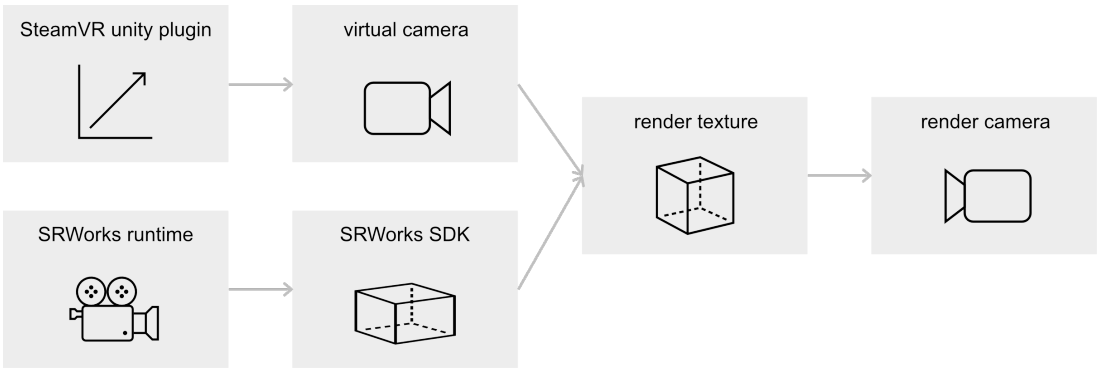

Figure 2.1: The virtual and the real cameras are aligned with each other. Their content is combined to render the AV output to the displays.

Since there are two cameras and two displays involved, this process happens twice. There is one rendering setup for the left and one for the right eye, which results in multiple virtual cameras involved in the process. Unfortunately, the cameras' feeds' can not be easily combined into one image to reduce processing time.

2.3 Stencil Objects

When it comes to extracting parts of the cameras' feeds, there are multiple ways to create a compound environment. These include various computer vision algorithms for analyzing the video. However, this project does not use any of these because they can be very complicated and resource-intensive, especially on a stereo camera feed. Since a camera image is only two-dimensional and the Vive Pro does not come with depth cameras, this also means that any depth information would be lost.

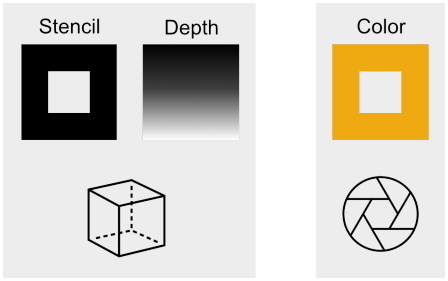

Figure 2.2: This figure illustrates the use of buffers. The reference object provides placement and depth information while the camera image is drawn only on relevant fragments. The result is a masked camera feed where the window is placed.

As an alternative, a virtual object is aligned with the real world, which then masks the cameras' videos. For each window, a reference object is placed in the virtual world. This object defines the location and scaling of the camera feeds' pixels to remain. The objects' shader writes to the stencil buffer, an extra buffer provided by Unity. Details can be found in the documentation in [2]. Any shader can compare any desired value

against the stencil buffer to determine if a fragment will be dropped from rendering.

The crucial part of this project's shader code is:

```
ColorMask 0
ZWrite On
Stencil{
  Ref 1
  Comp always
  Pass replace
}
```

This functionality is also illustrated in figure 2.2. The reference object writes the value 1 to the stencil buffer, and the SRWorks camera image only adds color where the stencil buffer equals 1. Additionally, the reference object writes to the depth buffer. Having a three-dimensional shape with depth information is a significant advantage of the implemented method. Otherwise, the camera feed would either appear always on top or always behind all the virtual content, depending on the camera setup. The virtual object also contains a collider so that virtual objects on top also stay on top in the compound world.

Chapter 3

Interaction

Various input devices can be used for digital and real surroundings simultaneously. The project is set up for the use with Vive controllers. Keyboard and mouse can be used on the computer screen but are not mapped in the virtual world. A regular xbox controller triggers the same events through Unity's input system. In this scenario, analog input such as a pen are involved as well. All these interaction modalities have to be mapped, so they make sense for potential users.

Since it can be cumbersome to switch the input device between real and virtual parts of the world, hand gestures have been added to interact with the virtual content. Two plugins are taken into consideration. The currently used Vive Hand Tracking SDK works well for the Vive Pro but will eventually be exchanged with the Windows Mixed Reality Toolkit (MRTK). This platform-independent toolkit might facilitate the change to a different headset if needed throughout the process. MRTK is also very feature-rich. It not only includes hand gestures but also a lot of UI presets, speech input and some handy development tools for logging and debugging.

Figure 3.1: This figure demonstrates some common spatial interaction possibilities provided by the Windows Mixed Reality Toolkit. Image source: [1].

Chapter 4

Implementation Details

Various implementational details improve the usability and visual coherence in AV.

The camera feeds can already be adjusted in brightness, contrast, saturation and white balance to better fit the virtual environment. To further enhance the visual perception, it is desired to blur the lines between the two worlds. One of the possibilities is to include three-dimensional frames around the reference objects, just like picture frames. Another option is to use similar overall colors and shapes in the virtual surrounding to match the real world. A different approach is to adjust the virtual cameras' resolution to fit the headset's cameras better. This might worsen the experience but enhance visual coherence and is worth a try.

Locomotion is another unsolved usability issue so far. In regular VR applications, the user is teleported through the world. However, the windows should stay in place and still approximately fit the virtual surroundings. Therefore, a movable virtual world alone will not solve the problem.

Chapter 5

Applications

Figure 5.1: In this demo, the object reference's shape is a cube which frames a physical desk. Additionally, a computer screen is streamed into the virtual environment. This application allows the user to develop and adjust a virtual reality application directly in Unity without removing the headset.

The system described in the previous chapters is the basis for interactive AV. Prototypical applications will demonstrate the benefits of this interaction method. There is a vast amount of possible use cases, some of which will be implemented. The project contains three demo applications so far. The windows into the real world are used to write onto a symbolic flipchart. The first demo includes a three-dimensional data visualization on Covid-19 cases, deaths and tests worldwide. It allows users to analyze the data and take notes on a whiteboard or a flipchart. The second one uses the three-dimensional world as a drawing reference. It shows a desk lamp that can be replicated on paper with a regular pen.

The third demo consists of a cube-shaped reference object, a computer screen feed,

and a virtual furnished apartment. Assuming that this environment is part of an ongoing VR project, the developers can change parameters, settings and code in Unity just as they are used to. However, with this method, all of this can happen within the VR headset for better perception of the environment.

This approach can be extended for business use cases. Especially during the last year, virtual collaboration and meeting tools flourished. Participants could bring along their real desk, a flipchart or a whiteboard or possibly even a sofa for a more comfortable experience.

Chapter 6

Summary

Augmented virtuality has become an appealing concept for interactive virtual environments thanks to improved VR headsets with pass-through AR support.

The concept of enabling interactions with the real world while using VR applications has been implemented prototypically in Unity with various plugins. These will be the basis for analyzing the benefits of this interaction method. The existing, upcoming and possible demo applications offer a lot of potential for expanding the project. They may be further categorized based on the amount of interaction with the real and the virtual surroundings.

References

[1] Yoon Park. *Introducing MRTK for Unity*. Version 2.5.4. May 2019. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/mrtk-getting-started (cit. on p. 7).

[2] Unity Technologies. *ShaderLab: Stencil*. Version 2019.4. Jan. 2021. URL: https://docs.unity3d.com/Manual/SL-Stencil.html (cit. on p. 5).