AccompliceVR: Lending Assistance to Immersed Users by Adding a Generic Collaborative Layer

Anthony Steed*
Department of Computer Science, University College London

Abstract

The current model of development for virtual reality applications is that a single application is responsible for construction of the main immersive experience. If the application is collaborative, that application must implement the required network functionality. We present VRAccomplice an overlay application that adds a collaboration layer to applications running on SteamVR. Using the Ubiq software, we can add avatars controlled by remote users as an overlay into running application. Remote users can see video of the local user. We demonstrate this with some simple examples of a remote user helping a local user in two popular SteamVR games.

Index Terms: Human-centered computing—Interaction Paradigms—Virtual reality; Computer graphics—Graphics systems and interfaces—Virtual reality

1 Introduction

Currently, most virtual reality applications are monolithic: a single application, coded in a development environment such as Unity or Unreal, is responsible for the behaviour of the virtual environment. In particular, if an application is collaborative, the application must itself implement the necessary networking functionality. Given that users are immersed in the system, it can even be hard for co-located persons to aid the immersed user. This might be contrasted with either gaming platforms such as PSNet or XBox Live where party formation and group chat persists outside any particular application, or conferencing systems such as Zoom where applications can be recruited and shared as video.

In this poster we introduce AccompliceVR which is a collaborative layer that integrates with existing VR applications running on the SteamVR platform. As shown in Fig. 1, a remote user (*Co-Pilot* following the naming adopted by XBox for its assistive controls[1], see also [5]), controls an avatar inside an application that is run by a local user (*Pilot*). In this case the two applications are Job Simulator[2] and Half Life Alyx[3]. Neither of these games is multi-user. The Co-Pilot can see what the immersed user is doing and can position themselves and gesture around the environment. We implemented this system using the Ubiq social virtual reality software [1]. Any number of immersed users can start AccompliceVR. Each immersed user creates an independent room on a Ubiq server. Remote users can then browse through the room list and join one of the immersed users.

2 Background

Collaboration support has been one of the key applications of VR from its inception (see short history of such systems in [4]). Today, many dedicated social VR platforms have been built; Schulz lists over 160 applications at the time of writing [3]. In addition there are 100s of games that support collaboration. What distinguishes a platform from a game, is, in general, that the platforms support extensibility in some form, e.g. by uploading and sharing objects, creating avatars or creating and animating worlds or rooms.

These platforms and games all implement their networking independently use one or more common APIs for networking. Unreal has built-in networking which is optional for developers to use. Unity has a variety of options, including the external Photon system[4]. Different applications take very different strategies for implementing their own networking (see [4]). While there have been various calls for standard inter-networking between different applications, there are no modern standards for this (though see the DIS standard developed in the 1980s to communicate between simulator systems). AccompliceVR works at a different level to game networking in that it works as an overlay application: networking, audio and avatars are provided by the Ubiq system [1].

Steam is a digital distribution platform developed by Valve Corporation. As a user, one installs Steam on one or more personal devices. Steam then manages software downloads and installations

Figure 1: Demonstrations showing AccompliceVR in action. Top: a remote user (Co-Pilot) represented as a Ubiq avatar (the blue-bodied avatar) within the game Job Simulator running on the local user's (Pilot's) machine. Bottom: the same Co-Pilot now joining the Pilot within the game Half Life Alyx.

[1] https://support.xbox.com/en-US/help/account-profile/accessibility/copilot
[2] Owlchemy Games, https://jobsimulatorgame.com/
[3] Valve Corporation, https://www.half-life.com/en/alyx/
[4] Exit Games, https://www.photonengine.com//Realtime

*e-mail: A.Steed@ucl.ac.uk

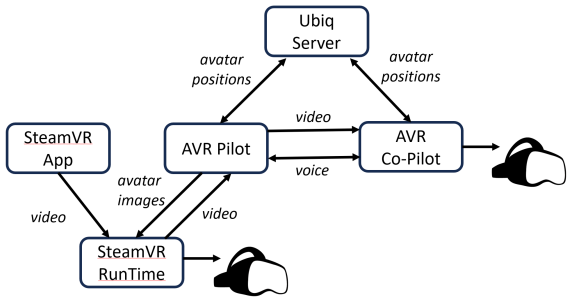

Figure 2: AccompliceVR system showing interactions between the processes. SteamVR App corresponds to an application that typically runs on SteamVR. The AVR Pilot receives video from the SteamVR runtime and relays it to the AVR Co-Pilot. It also renders dynamic avatar images that are overlaid on the SteamVR App by the SteamVR runtime.

on each device. Steam is one way of managing VR applications on a Windows PC. SteamVR, an extension of Steam, supports the interfacing of applications to VR devices[5]. In particular, one typically launches SteamVR to manage the device(s) connected to the device. SteamVR creates a default environment for the user when no other application is running. When a VR application is launched by Steam, it then creates an immersive display using SteamVR services. For the Windows case, an application developer uses a SDK such as OpenXR to create an executable that can then can use SteamVR services at run-time.

AccompliceVR works by using SteamVR run-time functionality for overlaying content over running applications. It has some similarities, to the DreamStream system [7]. The types of modification AccompliceVR does are supported by the SteamVR run-time, whereas DreamStream uses code injection techniques. On the flipside, AccompliceVR is unable to access a depth buffer, so cannot reconstruct a 3D environment. This makes ApplicanceVR in part similar to spectator systems such as TransceiVR [6], but AccompliceVR allows multiple remote users to interact with the local user.

3 System

The architecture of the system is shown in Fig. 2. The networking of player positions, audio and video is done with Ubiq [1]. We run a Ubiq server (a Node process) that acts as a point of rendezvous. Each Pilot creates their own *Room* on the server. New Co-Pilots can connect to the Ubiq server and will see a list of Rooms to which they can connect. Multiple Co-Pilots can connect to the same Room. Once a Co-Pilot joins, Ubiq sets up the voice and video communication between them as peers. The server only relays positions of Co-Pilot(s) and Pilot to each other.

For the Pilot we used built-in functionality of SteamVR via the OpenVR XR plugin (v1.1.4) to Unity (v2021.3.16f). SteamVR allows client applications to overlay textures into the running application. This is typically used to create user-interfaces, but it was used by the Vermillion software to create a painting application within other applications[6]. The Pilot software is thus a standalone application that runs at the same time as the main VR application. Note that SteamVR allows multiple overlay applications to run at the same time as a normal application, whereas only one normal application can run at once and thus controls the main view on the screen(s). The Pilot creates, using the `OpenVR.Overlay` class, one overlay texture per Co-Pilot. The avatar for that Co-Pilot is rendered into the overlay texture. The texture is positioned on a billboard that is moved to the position where it intersects the correct view volume for where that avatar would be relative to the user. Conversely, the Pilot process uses the `OpenVR.Compositor` class to extract images of the screen (via `GetMirrorTextureD3D11` function) and streams these to the Co-Pilot. For the audio on the Pilot, audio mixing happens naturally at the operating system level, and we use the built-in Ubiq 3D audio positioning.

The Co-Pilot is very simple. We show the streamed video relative to an avatar of the Pilot. The Co-Pilot can be an immersed user or they can use a desktop or other style of client supported by Ubiq.

[5]Valve Corporation, `https://store.steampowered.com/steamvr`

[6]Mountainborn Studios OÜ, `https://vermillion-vr.com/`

4 Discussion & Conclusions

We described AccompliceVR, a generic collaboration layer that allows a remote user to interact with an immersed user of applications running on SteamVR. We demonstrated this running in two common VR games, but it works in all of the games that we have tried so far. Multiple Co-Pilots can connect. The main disadvantages compared to a system such as DreamStream [7] is that the remote view is simple video and the immersed user only sees an overlay of an avatar, not a depth-mixed avatar. However, the main advantages of our system are that it is relatively lightweight and works with most SteamVR applications as it uses built-in functionality of the SteamVR run-time.

Our tool's first substantive use will be as an accessibility tool to help on-board naïve users to VR experiences. We can have the demonstrator of the system help the user outside the system, and once the user has donned the equipment, the demonstrator can assist them inside the VR system by pointing out how different parts of the user interface work. To further assist the Co-Pilot, we will also investigate reconstruction of scenes from video [2]. We expect that many other uses will be found as a generic assistance and observation tool.

Acknowledgments

The author wishes to thank Sebastian Friston and Ben Congdon, the main developers of Ubiq. This work was partly funded by the UK's EPSRC project Graphics Pipelines for Next Generation Mixed Reality Systems (EP/T01346X/1)

References

[1] S. J. Friston, B. J. Congdon, D. Swapp, L. Izzouzi, K. Brandstätter, D. Archer, O. Olkkonen, F. J. Thiel, and A. Steed. Ubiq: A system to build flexible social virtual reality experiences. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–11, 2021.

[2] G. Pintore, C. Mura, F. Ganovelli, L. Fuentes-Perez, R. Pajarola, and E. Gobbetti. State-of-the-art in automatic 3d reconstruction of structured indoor environments. In *Computer Graphics Forum*, vol. 39, pp. 667–699. Wiley Online Library, 2020.

[3] R. Schulz. Comprehensive List of Social VR Platforms and Virtual Worlds, 2023. `https://ryanschultz.com/list-of-social-vr-virtual-worlds`.

[4] A. Steed and M. F. D. d. Oliveira. *Networked Graphics - Building Networked Games and Virtual Environments*. Academic Press, 2009.

[5] F. J. Thiel and A. Steed. " lend me a hand"–extending the reach of seated vr players in unmodified games through remote co-piloting. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 214–219. IEEE, 2021.

[6] B. Thoravi Kumaravel, C. Nguyen, S. DiVerdi, and B. Hartmann. TransceiVR: Bridging Asymmetrical Communication Between VR Users and External Collaborators. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pp. 182–195. ACM, Virtual Event USA, Oct. 2020. doi: 10.1145/3379337.3415827

[7] B. Thoravi Kumaravel and A. D. Wilson. Dreamstream: Immersive and interactive spectating in vr. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2022.