# DreamStream: Immersive and Interactive Spectating in VR

Balasaravanan Thoravi Kumaravel
Microsoft Research
Redmond, WA, USA
UC Berkeley
Berkeley, CA, USA
bala@eecs.berkeley.edu

Andrew D. Wilson
Microsoft Research
Redmond, WA, USA
awilson@microsoft.com

## ABSTRACT

Today spectating and streaming virtual reality (VR) activities typically involves spectators viewing a 2D stream of the VR user's view. Streaming 2D videos of the game play is popular and well-supported by platforms such as Twitch. However, the generic streaming of full 3D representations is less explored. Thus, while the VR player's experience may be fully immersive, spectators are limited to 2D videos. This asymmetry lessens the overall experience for spectators, who themselves may be eager to spectate in VR. DreamStream puts viewers in the virtual environment of the VR application, allowing them to look "over the shoulder" of the VR player. Spectators can view streamed VR content immersively in 3D, independently explore the VR scene beyond what the VR player sees and ultimately cohabit the virtual environment alongside the VR player. For the VR player, DreamStream provides a spatial awareness of all their spectators. DreamStream retrofits and works with existing VR applications. We discuss the design and implementation of DreamStream, and carry out three qualitative informal evaluations. These evaluations shed light on the strengths and weakness of using DreamStream for the purpose of interactive spectating. Our participants found that DreamStream's VR viewer interface offered increased immersion, and made it easier to communicate and interact with the VR player.

## CCS CONCEPTS

• **Human-centered computing → Virtual reality**; **Collaborative interaction**.

## KEYWORDS

Virtual Reality, Streaming, Collaborative Interactions

**ACM Reference Format:**
Balasaravanan Thoravi Kumaravel and Andrew D. Wilson. 2022. DreamStream: Immersive and Interactive Spectating in VR. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3491102.3517508

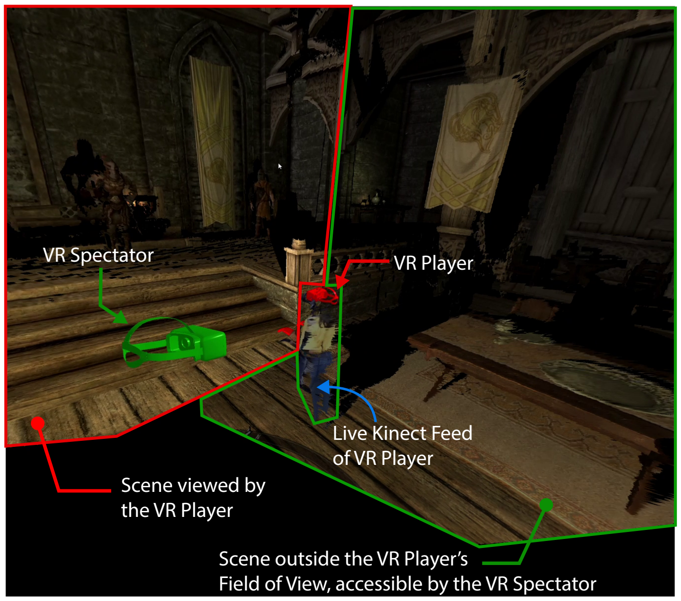
VR Spectator
VR Player
Live Kinect Feed of VR Player
Scene viewed by the VR Player
Scene outside the VR Player's Field of View, accessible by the VR Spectator

**Figure 1: DreamStream allows spectators to view live 3D reconstructions of a VR game either through VR or through a 2D Desktop interface. A VR spectator (represented by a green avatar) watches the live stream of a VR player (represented by a red avatar) who is playing Skyrim. Besides seeing what the VR player sees, the spectator can also see the ambient parts of the scene from a perspective of their choice. DreamStream also composites a live 3D reconstruction of the player.**

## 1 INTRODUCTION

Virtual Reality (VR) is increasingly popular, with applications in gaming, collaboration, design and media consumption. As VR use increases, there is a growing population of VR users that share their VR activity to multiple spectators through live media streams. This is popularly done for purposes such as entertainment, instruction and guidance. Such spectating practises have the potential to influence the overall VR user experience, as they have done for gaming before VR [17, 20]. New social dynamics emerge out of such spectating practises, at different levels, including one-one, small groups, and, crowds at scale. Prior work [5, 17, 21, 23] has studied the varying motives of the spectators, as well as the roles they take in these different settings. The role of a spectator can range from someone who purely observes the activity, to someone who actively engages with other spectators and the VR player itself.

Present day systems that enable live streaming of VR activity stream live 2D video feeds of the VR player's activity. This has the advantage that there are codecs, platforms and user bases firmly established by today's livestreaming practices. However, streaming

2D video of VR content introduces an asymmetry between the VR player and spectators: the VR player has access to an immersive and stereo view, while spectators view a 2D video on a conventional display. Prior work [26, 27] has discussed issues in using such 2D videos to spectate and interact with a VR player. Spectating VR content via a 2D display is perhaps expected given that few people today own sophisticated VR hardware. However, as VR becomes more accessible and more mainstream, we expect there will be more demand to spectate VR in VR. For someone with a VR headset, the vanilla 2D spectating experience of a VR activity may be disappointing and even puzzling.

Some VR applications include support for multiple users. In these, the spectators would be located within the virtual environment of the VR activity, and would have the ability to freely look around them. They may even be able to move independently in the virtual space of the VR player. These experiences are usually multi-player VR games, and live VR social spaces. In these, the spectator is required to download an entire application onto their system. This contains all the 3D assets of the virtual environment and spectator interactions. Thus, during run-time, spectators and the VR player, cohabit a shared virtual space, and such a system facilitates a range of interactions between them. This is similar to how multi-player games work today. A drawback of this approach is that it requires every spectator to own a full copy of the player's VR application. This can be problematic, as the game may be large, require time to install and may demand specific hardware. "Drop-in" access that is typical of live streaming platforms is difficult or impossible. Furthermore, the VR application must be developed to support such multi-spectator interactions.

In this paper, we discuss design considerations that are relevant when developing systems for spectating a VR activity. The proposed DreamStream system allows sharing VR activity carried out in existing commercial VR applications today. It retrofits these applications and follows a live *streaming approach*, in which the virtual assets are present only at the VR player's computer. DreamStream streams custom video textures (*2.5D frames*) that comprise color and depth components in real-time. At the spectator end, DreamStream uses these custom textures to create a live 3D reconstruction of the virtual scene. Spectators can view and interact with the streamed content either immersively using a VR HMD or through an interactive 2D UI. In addition to spectating only what the VR player sees, a spectator can also explore the scene independently and view regions outside the VR player's field of view. DreamStream's approach decouples the spectators' view from that of the VR player.

To visualize presence of one another in the space, spectators as well as the VR player are rendered as virtual avatars in the scene. Optionally, DreamStream can render a live 3D capture of the VR player in addition to the avatar. Such a digital representation of users allow spectators and the VR player to cohabit and feel co-present in the 3D virtual environment of the application. Thus DreamStream facilitates a user experience and interface style similar to one found in a multiplayer application, but uses the scalable technological pipeline of the *streaming approach*, allowing for "drop-in" access.

A key feature of DreamStream, is that it uses interventions made at the at the *VR platform level* [27], enabling it to work with existing commercial VR applications, without source code modification. We tested this approach with seven popular VR applications - Skyrim, TiltBrush, Blocks, TrainerVR, BeatSaber, Waltz of the Wizard, Fallout 4 (Fig. 2).

We conducted three informal qualitative evaluations: (1) A qualitative evaluation of DreamStream with four professional VR streamers (experts), (2) An initial user evaluation with eight participants playing the role of spectator, comparing DreamStream with a standard VR mirror and (3) A final user evaluation with twelve participants in which DreamStream was compared with TransceiVR [27], a relevant system from prior work. Study results highlight how these interfaces impact a spectator's ability to experience, understand, interact with and enjoy a VR stream. Together, they show that DreamStream's VR viewer shows promise in enabling better VR spectating experiences.

Our main contribution is the DreamStream system that carries out interventions at the *VR platform level*, uses 3D capture systems, leverages depth textures to stream 3D views at scale via H.264. DreamStream implements immersive and 2D interfaces, allowing spectators of a VR activity to cohabit the space of the VR player, offering interactions similar to a multi-user system. A secondary contribution of the work is a discussion of design considerations for building systems that support interactive spectating of VR activities.

## 2 RELATED WORK

DreamStream builds closely on prior work in collaboration in VR, General-purpose application-agnostic media sharing techniques and livestreaming VR content.

### 2.1 Collaboration in VR

Research on collaboration in digital 3D virtual environments goes back many years [7]. One of the earliest proposals of a multi-user VR system is DIVE [4], in which users are part of a shared 3D virtual environment, and share the screens of their 2D desktop applications. *Populated Web* [3] proposes a multi-user 3D web browsing experience in which web pages are laid out in a shared 3D virtual space. Using a 2D desktop interface or VR headset, users can spatially navigate web content, and are themselves represented as avatars, interacting with each other through text, voice, and video. Though the work focuses on web browsing, the system's core interactions are strikingly similar to those of today's social 3D virtual environments such as Horizon Workrooms[1], Mozilla Hubs[2] and VRChat[3]. Subsequent works highlight the essential factors to facilitate collaboration in virtual environments. These include enabling co-presence in virtual environments, allowing independent exploration in 3D space and access to relevant 3D information across a variety of interfaces [1, 2, 4, 12, 25, 28].

Three-dimensional environments supposedly allow users to more easily perform tasks which are inherently 3D in nature, such as 3D modelling and design, planning surgeries, performing architectural layouts, and watching immersive media content such as movies and games. Developing immersive systems requires an understanding of the unique goals and challenges posed by the specific

[1]Workrooms: VR for business meetings, https://www.oculus.com/workrooms/
[2]Hubs by Mozilla, https://hubs.mozilla.com/
[3]VRChat, https://hello.vrchat.com/

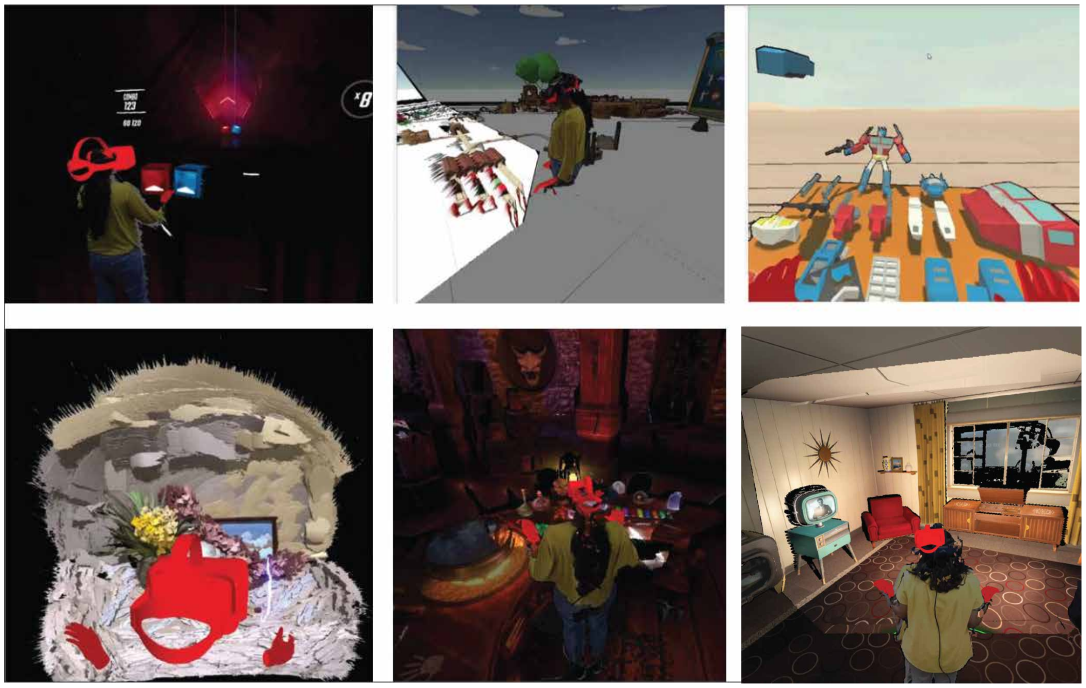

**Figure 2: A glimpse of other applications that have been tested with DreamStream - (Top) - Beatsaber, TrainerVR, Blocks; (Bottom) - Tiltbrush, Waltz of the Wizard, Fallout 4**

task and its target users. The recent emergence of consumer-grade VR devices is allowing researchers to better understand and design collaborative VR systems for these tasks. Xia et. al [29] notes challenges in collaborative scene editing through VR interface. Their SpaceTime system uses interaction techniques to allow for parallel manipulation and conflict resolution of 3D designs. Nguyen et. al's CollaVR [18] system allows for efficient synchronous collaborative review of 360 degree media content.

A number of prior works in collaboration focus on asymmetric interaction scenarios, where only some users use VR interfaces. FaceDisplay [11], ShareVR [10] and CoVR [15] allow external spectators to independently view and sometimes interact with the virtual environment through projection or touch displays. These works provide valuable insight and novel interaction patterns for developing VR experiences where multi-user interactions are a core aspect of the experience. These features must be built into the system during the development cycle of the target VR application, much as how multi-player games are developed today. In contrast to such *multiplayer approaches*, the focus of DreamStream is to bring such interactions to applications that were designed to be single-user. It accomplishes this by retrofitting these applications at the VR platform level [27].

## 2.2 General purpose media sharing

Today collaboration in video conferencing often employs general purpose techniques to share 2D application windows or entire desktops. These sharing features are most often implemented using video streaming codecs, so that other users on the call need not download or install the shared application. Capturing and transmitting the 3D geometry and texture information of virtual spaces is more difficult and less standardized. Niederauer et. al [19] proposes a non-invasive technique to access and reconstruct the 3D geometry of a scene rendered in OpenGL. This is an inspiring idea, but their technique does not work well with today's VR applications. Firstly, it assumes that geometry culling is turned off, which allows it to capture the entire architectural model of the surrounding scene. However, VR applications today cull parts of the scene outside the player's field of view. Secondly, it makes assumptions on how geometry information is stored, and requires user inputs to properly generate the scene geometry. Thirdly, it assumes that the architecture of the scene remains mostly static and does not change often. Finally, its implementation is specific to OpenGL, whereas most VR applications today rely on the DirectX framework. In contrast, DreamStream does not fail entirely even if culling is performed. This is because DreamStream does not capture the entire scene at a single moment but continuously captures the parts of the scene in view. For the same reason, DreamStream does not make any assumption on the static nature of the scene nor on the structure of the geometry buffer.

Other prior works exists which propose application-agnostic techniques to share VR activity to others: TutoriVR [26] allows users to record and playback tutorials using enhanced capturing and playback techniques. RealityCheck [13] and TransceiVR [27] are perhaps the most closely related works to ours. RealityCheck allows co-located viewers to spectate the VR experience mapped onto a physical environment. It also composites a Kinect capture of the surrounding physical world into the VR scene. To do this, RealityCheck intercepts the graphical rendering pipeline of the VR application. We build directly on this approach, modifying it to suit

DreamStream . TransceiVR allows a viewer to interact with a VR player through a live 2D VR feed, fostering collaboration between the two. The work suggests key design goals to enhance communication between the player and a viewer: *providing a stable VR view, supporting independent exploration by viewers, augmenting conversation with spatial referencing* and *supporting multi-modal context sharing*. The *TransceiVR* system supports these design goals on a 2D interface. This can be sufficient when the goal of the interaction is a focused collaboration between a single VR user and a single non-VR user. However, it does not scale beyond two users and does not enhance the spectating experience, co-presence and interactivity between the VR player and possibly multiple spectators. DreamStream inherently supports these design goals by providing an immersive interface, along with a live reconstruction of the virtual scene that the VR user and spectators may occupy together. We discuss the advantages of our approach in the context of the final user study, where we compared DreamStream with TransceiVR. Unlike TransceiVR and TutoriVR, DreamStream supports dynamic and fast-paced virtual environments and activities and allows the spectator to have a secondary view of the scene independent of the VR player's view.

## 2.3 Livestreaming VR content

Spectating live video game streams is an increasingly popular online activity. Typically, a live screen capture of the host player's game play is encoded and streamed at scale to multiple spectators. Through real time chat and other mechanisms, spectators can influence the way players go about their games [20, 21], and host players must consider how their in-game activities impacts the spectator experience. This is difficult because spectators themselves may have different goals. Cheung and Huang [5] identify nine different spectator personas involving a varying degree of interaction with the host player. We may consider spectating as a form of collaboration, where the goal is entertainment for all, and as such can face many of the same challenges. Providing a good spectating experience in VR is even more challenging for a number of reasons. Foremost is the fact that while the player operates in an immersive environment, spectators view a 2D video stream. The 2D video is limited in conveying the scale and the richness of the 3D experience of the player. Secondly, the choice of viewport of the video can dramatically impact VR streams. Often VR streamers share their first person view (FPV), but this can be quite shaky and difficult to watch. With a third person view (TPV), spectators may miss some critical parts of the action. This problem is related to the challenges in watching FPV videos of users doing physical tasks [9, 22]. JackIn [16] stitches frames from the FPV video stream and maps them to spatially registered frames immersively surrounding a spectator, thereby decoupling the head motions of the spectator from that of the camera. In contrast to approaches that mitigate the effects of FPV videos, an online survey by Emmerich et. al [6] suggests that most viewers prefer the first-person version when spectating VR game play as 2D video feeds.

Today, Liv[5] and MixCast[6] are popular general purpose solutions for streaming VR activity. They offer stabilized first person views, increased field of view, and third person views (TPVs). Liv and MixCast also allow compositing 2D video or an abstract avatar of the player placed between the set background and foreground layers of a VR feed. Output of these systems is ultimately a 2D video, in which the viewer experience is very different from that of the player. These systems do not allow for spatial interaction between the player and viewers, nor do viewers have independent control of their viewport. Some of these features are built into certain games to facilitate enhanced spectator experience e.g. JobSimulator[4]. To increase spectator immersion in VR, vReal[7] allowed spectators to be part of the virtual space of the game. All these tools require developer support. From our interviews with expert streamers, we learned that while vReal was thought to be promising, it had failed to gain traction with developers and was shut down.

Conventional 2D *streaming approaches* can work with any VR application. However, supporting interactions typical of multi-user VR applications will likely require deep integration with custom SDKs during development, particularly as they manipulate elements of the application's 3D scene. In contrast, DreamStream performs interventions at the *SteamVR platform level* [27] by intercepting the graphic rendering pipeline itself. Thus, it works with many of today's VR applications, and enables similar multi-user immersive interactions while also leveraging the scalable content distribution pipelines used by traditional video *streaming approaches*.

## 3 DESIGNING SYSTEMS FOR SPECTATING VR ACTIVITIES

Multi-user apps and games can be networked and shared in a variety of ways. Typically there is a player who performs some activity, observed by spectators. Today, there are a few broad approaches in which the player's activity could be shared in VR:

- *Multiplayer approach*: The player and the spectators each have a local copy of the application assets. For spectators, the player experience is recreated by transmitting data such as player and object poses and events. Most multiplayer games today use this approach.
- *Streaming approach*: A video stream of the player's VR activity is streamed to spectators typically via an online service such as Twitch[8], MixCast[6] or Liv[5]. The video stream may be generated from the rendered first-person view of the player or a separate third person camera.
- *Cloud approach*: Games may now be hosted in the cloud, and therefore may support spectators in the same manner as locally hosted multiplayer games, but with the advantage that all assets are stored on the cloud and rendered frames are streamed to each player and spectator.

Given these broad approaches to sharing activities in VR, the following design considerations come to light:

[4]Job Simulator, https://www.jobsimulatorgame.com/
[5]Liv, https://www.liv.tv
[6]MixCast, https://www.mixcast.me/
[7]vReal, https://vreal.net/
[8]Twitch, https://twitch.tv/

[9]Stadia, https://stadia.google.com/
[10]Amazon Luna, https://www.amazon.com/luna/landing-page
[11]Xbox cloud gaming,www.xbox.com/en-US/xbox-game-pass/cloud-gaming

## 3.1 Where is the spectator view rendered?

*3.1.1 Multiplayer approach - Rendering at everyone's PC.* The *multiplayer approach* recreates the player's experience for spectators using the same graphical assets found on the player's system. Local rendering (on the spectator's system) has the advantage that it can proceed asynchronously with the player and therefore can meet VR hardware's high frame-rate, low latency requirements. It can also easily allow each spectator their own view into the scene. However, this approach can be very demanding for the spectator, especially for occasional use: they must have a copy of the application installed on their system, and their system must be powerful enough to run the full application in VR. Finally, the application must be designed and developed to support multi-user features.

*3.1.2 Streaming Approach - Rendering at the player's PC.* The *streaming approach* on other hand, is a popular activity sharing approach today. A 2D video feed of the VR user's activity is encoded and streamed to multiple spectators simultaneously using desktop sharing techniques and standardized video codecs. The spectator is typically not in an immersive environment and need not have the VR application installed. This is possible because the infrastructure facilitating video streaming is widely available today, across every device, and scales for a large number of users. Additionally, the screen-capturing ability is supported at a platform-level [27] today for most computing systems. This means that, in contrast to other approaches, any existing VR application can be streamed as a 2D video. Transmitting geometry and texture information for local rendering is technically possible but today there is no generic means to do so, and while there is mature support for video compression, there are comparatively few good options for compressing and transmitting 3D geometry data.

*3.1.3 Cloud approach - Rendering at cloud PC.* With the *cloud approach*, the application is stored and run on a remote server. Therefore, the spectator as well as the player need not have the application installed in their local computers and is therefore closer to supporting the "drop-in" access of streaming. For multiplayer games, every user can independently choose and control their view of the player user's scene. The computer in the cloud is ideally powerful enough to simultaneously render, encode and transmit the different video streams corresponding to viewports of different spectators. However, such a multi-spectator support needs to be built into the application during development and does not work for every application. This is currently deployed for non-VR games by services such as Google Stadia[9], Amazon Luna[10], and Xbox Cloud Gaming[11]. An important consideration is the latency between a user's input action (often made using a controller) and the updated video stream that results form the input action; i.e., the time between a spectator or player invoking a move action, and the updated stream corresponding to the move action. Latency to cloud systems is typically high (on the order of 100ms), and reducing this is still an active area of research.

## 3.2 How is the spectator view rendered?

Multiplayer games synchronize game state between players and spectators and render the scene with full access to geometry and textures. These natively support giving the player and spectators their own view. Meanwhile, streaming approaches benefit from the generic nature of 2D video capture, but lack support for multiple views. The player's first person view can be difficult to watch due to camera movement, especially in VR. This is a major drawback of the *streaming approach* for VR today: spectators give up the control of their view into the scene. This supports only a single video stream, which is encoded and transmitted to all spectators. Hence they all take the same perspective of the VR scene. Applications that use the cloud approach need to be developed specifically to support either form of rendering.

The VR player performs activities in an immersive 3D environment. However, spectators' view can be rendered in different ways. They might share the same immersive environment, but they may also view 2D, stereo or 360 degree video. Each of these modalities preserve detail at different levels. 2D video is the most common, but degrades both the sense of 3D depth as well as the spectator's overall spatial and directional awareness [27]. Stereo video enhances the perceived 3D depth and is used today in 3D TVs and 3D cinema. This requires some kind of hardware to allow for providing different video streams to both eyes of the user. Today, both 2D and stereo feeds of any VR applications can be captured at a platform level [27]. Theoretically, it is feasible to have the spectators observe the activity in VR via a stereo video feed which embeds left and right eye views. However, when viewed in VR, such stereo feeds can cause motion sickness for spectators when there is significant camera motion. 360 degree video enhances spatial awareness of a scene by allowing the user to change their viewpoint, and are becoming increasingly common in online video sharing platforms [8]. Stereo 360 degree video is also an increasingly popular option in VR but provides limited means of changing the viewer's position. Additionally, access to such 360 video feeds is not present in existing applications and needs to be specifically supported by developers. Finally, a complete 3D environment is rendered locally just like a game, and ideally retains the perceived depth through stereo rendering. It allows spectators to look around the VR player's environment, as well as to move within it.

Depending on the hardware available to the spectator, each of these mediums can be viewed through a VR headset or a 2D desktop UI. Additionally, multiple spectators can use different modalities simultaneously. It is then important for the system to gracefully degrade the experience tailored to the capabilities of the user's hardware, while also maximizing the information perceivable by them. DreamStream allows spectating using both 2D desktop UI as well as through a VR headset, through a medium that represents a part of the complete 3D environment of the VR player.

## 3.3 What interactivity is offered to the spectator?

Interaction among players in multiplayer games is limited only by what the developer has anticipated and built into the game. With streaming, interaction between the player and spectators is limited to that which the streaming platform supports. Today, that is typically text chat, rendered as an overlay or in a web page.

Systems for spectating VR activities, can operate at different layers of the technology stack [27]. With source code/API access, developers can implement features that can directly access and modify scene elements for providing an enhanced spectating experience. This includes addition of new spectator cameras, rendering spectator avatars into the scene, adding interfaces for the player to interact with spectators, as well as for spectators to manipulate scene elements. Without such source code/API access, modification of the VR rendering may be limited to adding 2D overlays.

Hence, prior work [13, 26, 27] has proposed developing systems at the VR platform level, working with existing VR applications and without source code/API access. The ability to retrofit existing applications is an important consideration for our system design. Specifically, we introduce software interventions that modify the OpenVR library allowing changing camera positions, capturing depth information and rendering custom objects into a VR scene. We use this approach to enable an enhanced, interactive spectating experience for existing VR applications.

A key aspect of any activity between player and spectator is the feeling that the spectator is together with the player, a sense of co-presence. With video-based sharing and collaboration, co-presence is enabled at scale through techniques such as text chat, viewer counts, emoji reactions, etc. For a smaller group of users, audio chat, and screen control sharing and annotations also becomes feasible. With 360 degree videos, techniques that convey the viewport of different users have been used in prior research systems [18].

In the case of complete 3D environments, spectators can adopt different perspectives in the scene and can perceive different parts of the action. To facilitate co-presence in such scenarios, existing systems typically adopt an avatar-based representation of each user in the space. These avatars can be abstract or a real-life replica of the users, and can vary in degrees of freedom depending on the fidelity of pose that is tracked for each user. These are commonly seen in today's social VR applications such as Mozilla Hubs, and Horizon Workrooms.

In DreamStream, the spatial presence of a spectator is communicated to the VR player, as well as other spectators through low-fidelity abstract avatars. However, the VR player's spatial presence and actions are communicated by compositing a live color and depth reconstruction of them into the VR scene. The data for this is acquired from Kinect cameras present in the VR player's space. The depth images received from depth cameras such as the Kinect are naturally suited to be combined with the scene depth images and in fact use similar rendering techniques; this can be contrasted with approaches that use a conventional 2D camera and a fixed overall viewpoint, such as Liv.

### 3.4 DreamStream's 2.5D video streaming and 3D reconstruction approach

We introduce a new approach that adopts many of the features and technology of streaming, but also allows the spectators and the player to have experiences and interactions that are typical of the multiplayer approach. The main idea is to embed the depth buffer used in rendering in the streamed video (Figure 4-d,i). The depth buffer or "z-buffer"[12] is commonly generated in today's 3D graphics rendering pipelines. It captures much of the 3D information of the scene from the player's viewpoint. For each pixel in an image, the corresponding pixel in a z-buffer contains information of the "depth into the scene". However, because it does not give the depth to surfaces behind visible objects, it is not a complete representation of the 3D scene. It also does not contain information on any screen-space effects that maybe rendered in the scene. We apply appropriate transformations to these depth buffers to reduce noise during the encoding and decoding process, and embed them alongside color (RGB) video frames. We call these depth-embedded video frames as *2.5D video frames*. Examples of these depth buffers and 2.5D video frames are shown in Figure 4.

To transmit this 2.5D representation, we leverage standard H.264 encoding, streaming and decoding pipelines. This allows a partial 3D reconstruction of the VR scene that is seen by a player. Therefore, unlike in the *streaming approach*, every spectator can now take different and independent perspectives from which they view this 3D reconstruction. We also propose the use of a secondary 2.5D stream that fills the spectators' view with the 3D information regarding the surrounding ambient environment of the VR scene, that may not be viewed by the VR user. This is integrated with the 3D reconstruction of what the VR user sees, and provides added context. Spectators can view and interact with these either using a VR headset or a 2D desktop user interface. The frame rate at which the 3D reconstruction is updated matches that achieved by existing *streaming approaches*. However, in our approach, the user does not suffer from motion sickness issues, because the final rendering now occurs locally at frame rates sufficient for VR (e.g., 90Hz) and because the spectator has control over their view. Moreover, we demonstrate that existing VR applications can be retrofit to use this technique. A drawback of this approach is that, depending upon the quality of encoding and decoding, the 3D reconstruction can exhibit certain artifacts, particularly around discontinuities in depth. In section 4.1.2, we discuss our approaches to mitigate these through appropriate handling of the depth buffer.

In summary, depth buffers have a number of advantages as a representation of scene geometry:

(1) They are ubiquitous in current graphics pipelines
(2) They capture much of the 3D information of a scene from a given perspective
(3) Unlike vertex buffers (meshes) they are more suited to conventional image compression and streaming approaches

Our current prototype gains access to the depth buffer by intercepting calls made at the VR platform level (i.e., OpenVR). We note that VR platforms could enable a wide variety of novel compositing techniques and mixed-reality scenarios [13] if the depth buffer and related information were exposed more directly. Because the depth buffer is so commonly used, it may be as generically useful for streaming, just as 32 bit color video is today.

## 4 DREAMSTREAM SYSTEM

The DreamStream system consists of three parts that operate coherently at different locations: DreamStream-Host, DreamStream-Client and DreamStream-Relay. The DreamStream-Host program runs on the VR player's computer. The DreamStream-Client runs on a spectator's computer, handles the rendering and provides the interface for the spectators who can either use it through a VR headset, or through a regular 2D UI. Finally, a DreamStream-Relay program hosted in the cloud acts as a relay between the

[12]Z-buffering, https://en.wikipedia.org/wiki/Z-buffering

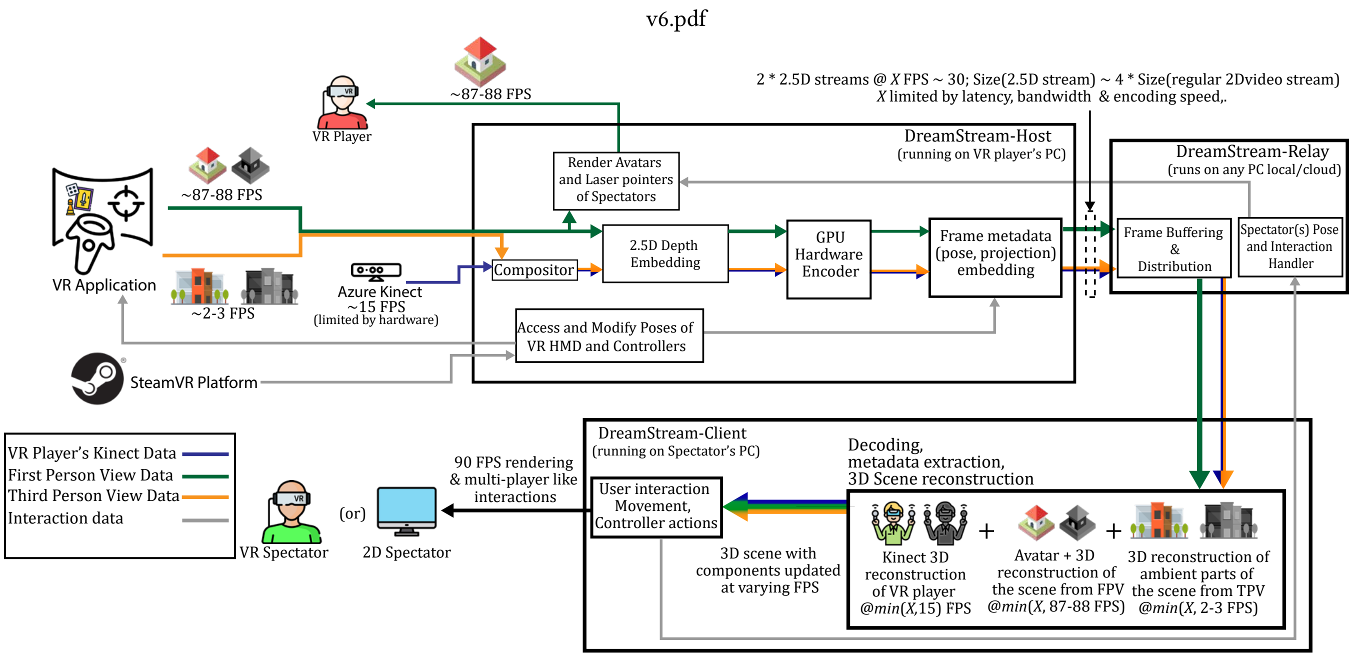
v6.pdf
~87-88 FPS
VR Player
2 * 2.5D streams @ X FPS ~ 30; Size(2.5D stream) ~ 4 * Size(regular 2Dvideo stream)
X limited by latency, bandwidth & encoding speed,.
DreamStream-Host
(running on VR player's PC)
DreamStream-Relay
(runs on any PC local/cloud)
~87-88 FPS
Render Avatars and Laser pointers of Spectators
2.5D Depth Embedding
GPU Hardware Encoder
Frame metadata (pose, projection) embedding
Frame Buffering & Distribution
Spectator(s) Pose and Interaction Handler
VR Application
Compositor
Azure Kinect ~15 FPS (limited by hardware)
~2-3 FPS
Access and Modify Poses of VR HMD and Controllers
SteamVR Platform
DreamStream-Client
(running on Spectator's PC)
Decoding, metadata extraction, 3D Scene reconstruction
VR Player's Kinect Data
First Person View Data
Third Person View Data
Interaction data
90 FPS rendering & multi-player like interactions
User interaction Movement, Controller actions
VR Spectator
(or)
2D Spectator
3D scene with components updated at varying FPS
Kinect 3D reconstruction of VR player @min(X,15) FPS
Avatar + 3D reconstruction of the scene from FPV @min(X, 87-88 FPS)
3D reconstruction of ambient parts of the scene from TPV @min(X, 2-3 FPS)

**Figure 3: In DreamStream, the 3D reconstruction of the VR scene seen by the spectators have three main graphical components, each of which may operate different frame rates: (1) FPV data of the VR player, (2) TPV data of ambient parts of the VR scene. TPV is controllable by the spectator and (3) Live Kinect data of the physical body of the VR player. These reconstructions corresponding to these data are rendered in a spatially coherent manner with its different parts being updated at different rates. The spectator however, has a full 90 FPS interaction over viewing and navigating these parts.**

DreamStream-Host and the multiple DreamStream-Client applications. The DreamStream-Host streams the player's VR environment data to the client applications, and receives from them the information of the different spectators.

## 4.1 DreamStream-Host: VR player side of DreamStream

The player side of DreamStream has three main functions: (1) intercept the video and depth textures from the VR application, (2) transform and stream them as 2.5D frames to the DreamStream-Relay, and (3) handle the rendering of spectator-related information into the player's VR headset.

DreamStream targets applications built to work with SteamVR. These applications use the OpenVR library to interface with VR hardware. The library is implemented by the openvr_api.dll file. We use similar C++ vTable injection techniques proposed in RealityCheck [13], through a custom DLL file that intercepts and operates within the functions calls made by the VR application.

*4.1.1 Intercepting RGB and depth information.* Similar to RealityCheck, calls to the functions `IVRCompositor::Submit`, `ID3D11DeviceContext::OMSetRenderTargets` and `ID3D11DeviceContext::ClearDepthStencil` are intercepted to access the rendered video frames and depth buffers sent by the VR application to the VR player's headset. However, at the time of this work, multiple VR applications appear to use additional rendering pipelines that do not follow either of the three rendering pipelines outlined by RealityCheck; e.g., the depth buffer for both eyes may be populated and processed only after the call to `Submit` when called for the right eye. In such cases, we use additional per-title heuristics to obtain the textures.

*4.1.2 Primary 2.5D video frames.* Primary 2.5D video frames are used to reconstruct and render a high frame rate 3D feed of the VR user's First Person View (FPV). To do this, The depth buffer and the video frame for both eyes must be prepared for streaming. Today, H.264 is the most common video encoding scheme. While this can be used to encode the color video frames, it generates noisy artifacts when directly applied to the depth buffer. This is due to two underlying factors. Firstly, H.264 algorithms are designed to preserve the visual appearance of video, and introduces compression artifacts when applied to depth buffers. To address this, Sonoda and Grunnet-Jepsen proposed a technique that is based on an inverse-colorization technique which maps depth values to the hue space [24]. We apply this operation to the depth buffers, using custom compute shaders, prior to embedding them alongside the video stream.

A second issue is that the depth buffer used by VR applications typically have 10 bit channel values, whereas H.264 commonly works with RGB textures with 8 bits per channel. Discretizing 10 bit to 8 bit values leads to visible discontinuities during 3D reconstruction. To address this, we split the depth buffer values into three regimes: Background, Far-Foreground, and Near-Foreground, using predefined distance-based thresholds tuned for specific ranges of interest. We arrived at these threshold values through empirical testing on different VR applications. These can be changed at run-time if required. We drop all depth buffer information that are closer than the threshold for Near-Foreground and further than the threshold for Background. Typically these influence the least significant bits (LSBs) and most significant bits (MSBs) of the depth buffer value. The range of depth buffer values is thus reduced so that it may be encoded. The Far-Foreground and Near-Foreground ranges then focus on different exclusive (non-overlapping) 8 bit ranges of the depth buffer. We then apply the inverse-colorization technique to these, and stitch them together with the RGB texture of the video feed to obtain a 2.5D video frame. The stitched figure is shown in Figure 4.d.

These operations are applied to VR user's FPV video feed to generate the Primary 2.5D video frames. The rate at which these frames are encoded and streamed are maximized and is limited by the latency, bandwidth and the GPU's encoding speed. In our setup, we achieve about 30 FPS. This is on par with widely used video streams today.

*4.1.3 Secondary 2.5D video frames.* Besides reconstructing the VR scene from FPV of the VR player, DreamStream reconstructs parts of the VR scene and the VR player themselves from a Third Person View (TPV). It does this by creating Secondary 2.5D frames that composite the *ambient* parts of the VR scene at a low frame rate, and a 3D Kinect feed of the VR player at a higher frame rate.

DreamStream captures the *ambient* parts of the VR scene surrounding the VR user, from a Third Person View (TPV). Spectators can control the TPV for which they need the *ambient* parts of the scene to be rendered. These *ambient* parts are rendered for a single frame by momentarily overriding the VR player's HMD pose reported by the VR platform to the VR application. We only alter its pose to capture from the chosen TPV but otherwise do not modify the rendering process. This TPV frame is hidden from the VR user by momentarily presenting the previous frame that was rendered using their actual HMD position in the VR world. Since the TPV frames are captured at the cost of FPV frame rate of the VR player, the *ambient* capture rate is limited to 2-3 FPS. Due to this, the effective frame rate of the VR user and the primary stream's frame rate drops slightly (up to 87-88 FPS). From our experience using the system, this drop in frame rate is often unnoticed by the VR player and spectators.

In addition to capturing ambient parts of the VR scene, we also capture live 3D kinect-based depth capture of the VR player as per their corresponding location in the VR space (see Figure 6(i)). We calibrate and acquire the kinect depth capture through a modified version of the RoomAlive Toolkit [14]. The Kinect feed is also captured from the same TPV chosen by the spectator.

Both the captured ambient parts of the VR scene from the chosen TPV and the 3D Kinect feed from the same TPV are then composited together to obtain a single RGB frame and depth buffer. This composited capture is then used to generate the secondary 2.5D frame using the same procedures used for generating the Primary 2.5D frames. This can be seen in Figure 1.

Taken together, the primary and secondary 2.5D streams contain high frame rate 3D data of VR player's main focus of interaction in the physical and virtual environments. The low frame rate 3D data of VR user's ambient environment provides added context. Qualitatively, the idea is that spectators have an "over the shoulder" view of the VR player from within the scene of the VR application. The Field of View (FoV) of the secondary 2.5D frame is the same as that of the primary 2.5D frame which is determined by the VR headset and the application. However, together the primary and secondary 2.5D frames increase the effective FoV perceived by the spectators when compared to using only the primary stream.

*4.1.4 Encoding textures.* Once the primary and the secondary 2.5D video frames are generated, the next task is to encode them. DreamStream encodes these streams in parallel threads and leverages separate hardware video encoders that are available on a GPU. For our setup, we use the NVIDIA GTX 1080 and its NVENC API that supports concurrent hardware encoding of up to two video textures with a maximum pixel resolution of $4096 \times 4096$. Once the video frames are encoded, we append a binary payload containing the view and projection matrices associated with that frame. These are required for successfully reconstructing the frame as a 3D mesh at the correct pose.

*4.1.5 Rendering into the VR scene.* Besides transmitting the streams, DreamStream also renders the avatars of spectators as well as their interactions into the VR scene. To do this, we leverage both the color texture and the depth buffer. These avatars denote the spatial location of each spectator. For the VR spectators, their avatar contains their head and hand poses depicted by headset and hand models. These poses are obtained using the spectators' VR HMD and hand controller poses. For spectators using the 2D UI, their avatar is represented using only a headset model that denotes the pose from which they view the VR scene. All spectators also have access to a laser pointer, with which they can point to different parts of the VR scene. This can be seen in Figure 5. The laser pointer can be triggered either by pressing a button on the VR controller (for VR spectator), or using right click on the mouse or pressing a specific key on the keyboard (for 2D UI spectator).

## 4.2 DreamStream-Relay

The relay acts as an intermediary between the DreamStream-Host and all the DreamStream-Client programs. It can achieve NAT traversal as well as efficient distribution to multiple spectators. Upon connecting to the relay server, a DreamStream-Client is assigned a unique client ID and is initialized with the stream's information required for 3D reconstruction, such as texture size and the VR camera's calibration parameters. A buffer for storage and distribution of encoded frames is allocated for each spectator client. This buffering mechanism ensures that spectators are not affected by intermittent network issues that maybe faced by any arbitrary spectator. It also keeps track of the slowest client, and uses that to dynamically change the encoding rate at run-time.

Besides distributing frames, the relay server exchanges and synchronizes information about spectator's viewport poses, hand controllers poses, and corresponding actions to the VR player and other spectators. This is sent through a communication channel separate from the ones uses for the primary and secondary frames.

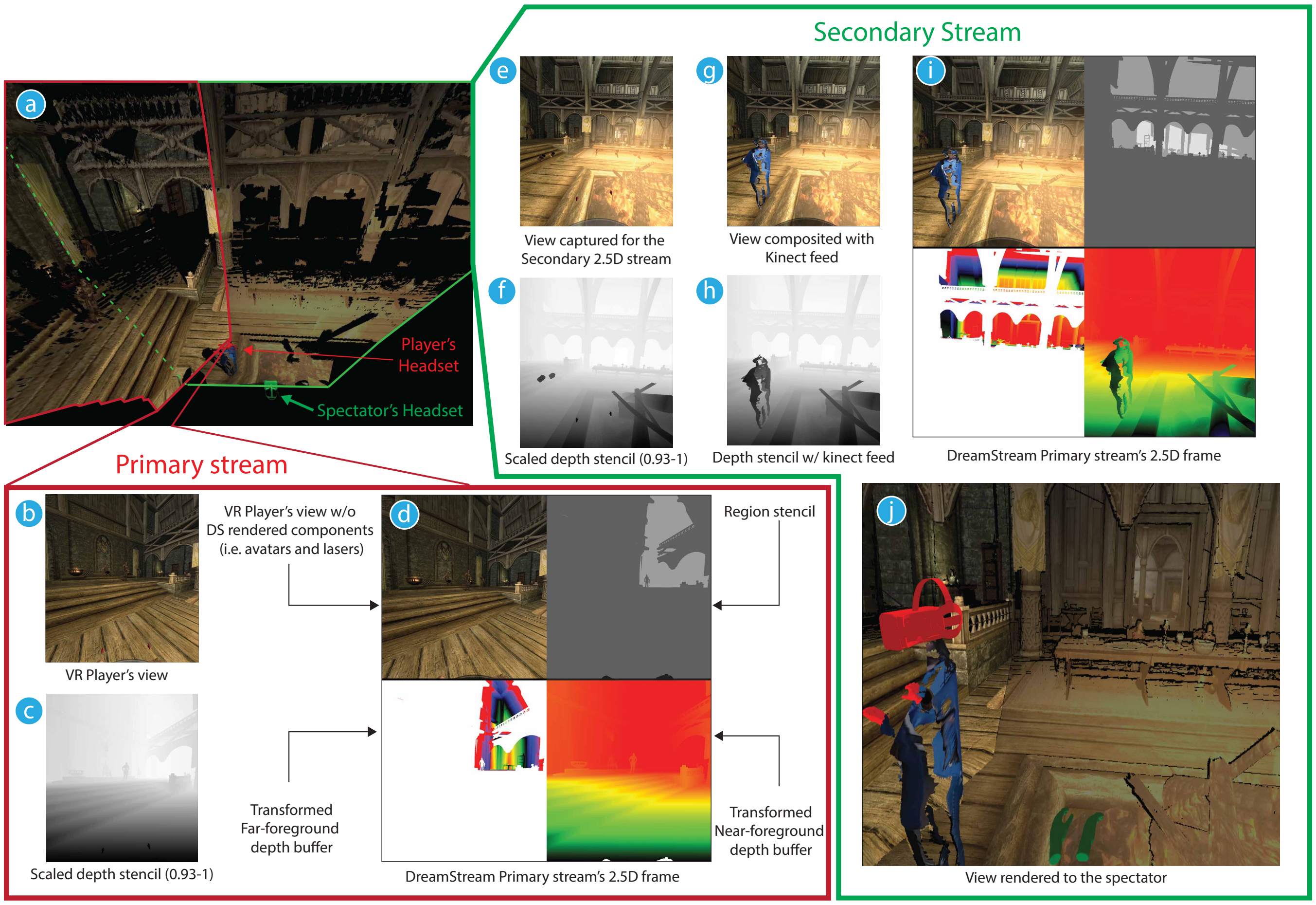
Secondary Stream
a
Player's Headset
Spectator's Headset
e
View captured for the Secondary 2.5D stream
g
View composited with Kinect feed
i
f
Scaled depth stencil (0.93-1)
h
Depth stencil w/ kinect feed
DreamStream Primary stream's 2.5D frame
Primary stream
b
VR Player's view
VR Player's view w/o DS rendered components (i.e. avatars and lasers)
d
Region stencil
j
c
Scaled depth stencil (0.93-1)
Transformed Far-foreground depth buffer
DreamStream Primary stream's 2.5D frame
Transformed Near-foreground depth buffer
View rendered to the spectator

**Figure 4: (a) 3D reconstruction of Primary and Secondary 2.5D frames. The head poses of the VR player and the spectator are represented as red and green avatars. (b) First-person video feed that VR player sees. (c) Depth stencil that has been re-scaled to the 0.93-1 range. This is where most scene data exists in the current scene. (d) A frame of DreamStream's primary 2.5D stream. Top-Left has the RGB feed from the user; Top-Right has the Stencil indicating the region the pixel belongs to (i.e near/far-foreground, too close, too far/Background); Bottom-Left contains the Depth information of the far-foreground range; Bottom-Right contains the Depth information of the near fore-ground range. (e,f) RGB feed and depth stencil of the view seen from the overridden position corresponding to the spectator's view. (g,h) Kinect feed being composited into the RGB and depth textures. (i) A frame of DreamStream's secondary 2.5D stream. This is similar to the primary stream. (j) View seen by the VR spectator based after reconstruction at DreamStream-Client**

## 4.3 DreamStream-Client

Every spectator runs the DreamStream-Client program, which is unaware of the inner details of the VR application. When connected, it obtains metadata from the DreamStream-Relay and initializes itself. It then uses the primary and the secondary streams to reconstruct the VR scene in 3D as well as composite the live Kinect feed of the VR user. The spectators have the ability to change the viewport used for generating the ambient parts of the VR scene, or disable it entirely.

The reconstructed 3D scene can be viewed by a spectator either through a VR headset, or through an interactive 2D UI (Figure 6). In the 2D UI, DreamStream offers four viewing modes: (1) Free Camera View allows for free and unconstrained movement of a spectating camera in the 3D space; (2) Orbit View also allows for free movement, but always orients towards the VR user. The controls are mapped in such a manner as to orbit the user. (3) Over the shoulder (Follow) view positions the spectator camera behind the VR user. The horizontal distance, height and the pitch of the camera can be varied in this mode; and finally the (4) First-person view replicates what the VR user sees. In addition to regular camera controls, the 2D UI offers an option for the view to be stabilized when in First Person View or Over the Shoulder View (Figure 6.f). The is performed by smoothing camera motion with an empirically tuned exponential filter. In all viewing modes, the ambient parts of the scene are captured from the point of view of the 2D viewer.

For viewing with the VR headset, the user can use their hand controllers to navigate the scene. By default the view for rendering ambient parts of the VR scene is set such that the entire body of the

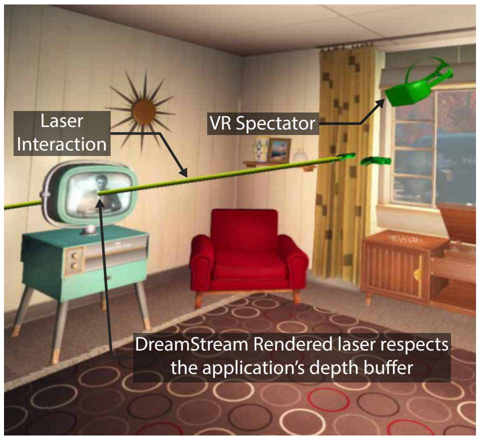
Laser Interaction
VR Spectator
DreamStream Rendered laser respects the application's depth buffer

**Figure 5: The VR player's view represents a VR spectator with a green avatar. The spectator is using a DreamStream's laser pointer to point to a TV present in a room in Fallout 4 VR. Note that the laser respects the depth buffer of the scene and can actually pass through objects in it as if it was rendered by the game itself.**

VR player is seen. Viewers can manually override this, by holding a specific button on the controller and use it point at part of the scene where they want to see, and the ambient parts in that region are rendered. The interaction is analogous to how one would use a flash light to illuminate unseen dark regions.

As in the player's view, a spectator can see avatars of other spectators in the scene. When viewing through a 2D UI, the viewport of the 2D viewer is used as a proxy for the spectator's spatial location in the scene. When viewing through a VR viewer, the HMD and hand controller poses are used to render the avatar. As mentioned earlier, all spectators have access to a laser pointer that can be toggled on and off through pressing a key on the VR controller or the keyboard or the mouse.

## 5 EVALUATION

To better understand the opportunities and shortcomings of the DreamStream system, we conducted three sets of informal evaluations: an expert evaluation with four experts, an initial evaluation with eight users and a revised evaluation with twelve users. We first describe the methodology and measurements for these. We found common themes from our expert study as well as from both user studies. We discuss these together in a common subsection.

### 5.1 Expert Evaluation

We recruited four professional VR steamers as experts (E1-E4) in our study; They have 12K, 120K, 380K and 3.3K subscribers on their streaming channel, and each has been streaming VR content for the last 6-7 years. We omit other information about our experts to ensure their anonymity. We interacted remotely with each streamer for about 70 minutes, and as a token of appreciation, we gave them a gift certificate of $60. At the start of the study, we conducted a semi-structured interview in which we elicited details on the state of the art of VR streaming; its goals, challenges and the workarounds that they employ to solve those. This served two purposes: (1) it helped us better understand these and (2) helped the streamers collect their thoughts on these so that they can evaluate our tool. After demoing DreamStream through Zoom, we asked them if it might solve any of the issues that they had mentioned earlier, and whether they could anticipate any issues in using the system.

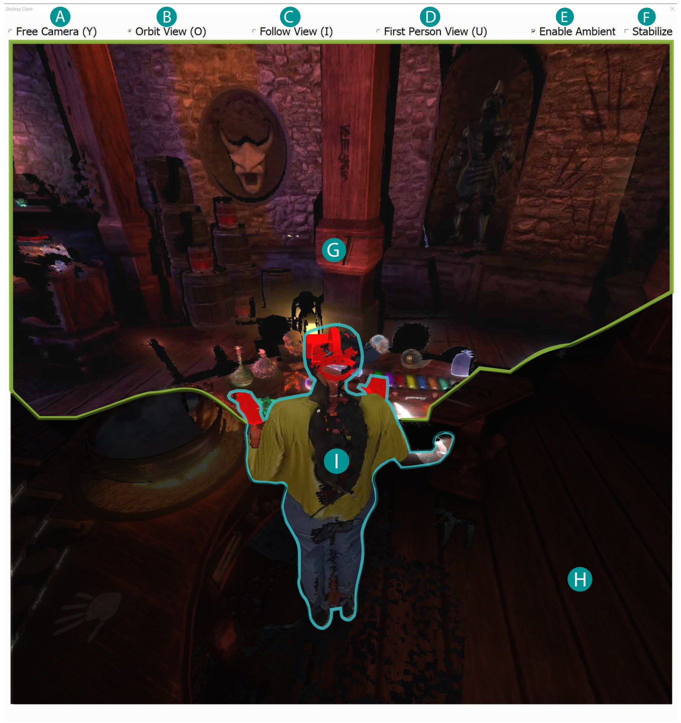

**Figure 6: DreamStream's interactive 2D viewer; (a)-(d) allow for switching to different viewing modes. (e) toggles the ambient reconstructions and (f) toggles the stabilization in follow and first-person view. Region (g) is the reconstruction from the primary 2.5D video frame. These are parts that are seen by the player. Region (h) is the reconstruction from the secondary 2.5D frame. These are rendered dimmer in color and are parts not seen by the player. This region is also updated at a low frame rate. Region (i) is a live 3D reconstruction of the player using the secondary 2.5D frames, but is updated at a high frame rate. (j) shows the red avatars of the headset and hands of the Player.**

### 5.2 User Evaluation

*5.2.1 Initial User Evaluation.* We conducted an initial evaluation with eight participants (I1-I8) (5 Male, 3 Female, Age range 20-28). All of them had prior experience with VR and games, and used VR and played games at least a couple of times every month. In this study, we compared DreamStream with the Standard VR mirror. Each study lasted 90 minutes, and the participants were compensated with a $40 gift card for their time. As per our campus regulations, to minimize Covid-related risks, an author of the paper took the role of a VR player and played Skyrim in VR, and a participant spectated and interacted with them. Participants were told that their goal was to have fun, occasionally interact with the player and aid them in making key decisions in the game. Beyond spectating, spectator interactions included identifying objects, navigating terrains as well as helping to choose other game player actions. Participants used DreamStream-Client through a laptop placed 10ft away from the player. The DreamStream-Host and the DreamStream-Client interfaces through the local network. The author's audio was heard directly by the participants, whereas the game audio was channeled in stereo through Zoom. They spectated through three different interfaces: (1) Standard video mirror, (2) DreamStream's Interactive 2D interface (DS-I2D), and (3) DreamStream's VR interface (DS-VR). Each participant spent roughly 12-15 minutes with each of the interfaces (within-subjects), and the order of the interfaces was counterbalanced.

*5.2.2 Final User Evaluation.* Our Final user evaluation consisted of 12 participants (P1-P12) (7 Male, 5 Female, Age range 23-31). All of them had used VR for at least a couple of times in the last 6 months. They play games at least a couple of times on a monthly basis. In this evaluation, we compared DreamStream with TransceiVR [27], which shares similar functionality for its overall goal of enabling asymmetric collaboration in VR. In line with the original implementation, TransceiVR was deployed and presented through an iPad-based touchscreen interface with a stylus. The evaluation was conducted in a similar manner as our initial evaluation.

*5.2.3 Measures.* For our final evaluation, our questionnaire consisted of 5-point Likert scale questions rating the following elements: (1) Easy to communicate to the VR player, (2) Easy to point/refer to objects to the VR player, (3) Easy to direct the VR player, (4) Easy to understand the VR user and their actions, (5) Easy to understand the VR scene, (6) Easy to explore independently, (7) Spectating was enjoyable, (8) They felt immersed in the game and (9) They felt along side the VR player. A Likert Rating of 1 corresponds to 'strongly disagree' and 5 to 'strongly agree'. Intermediate values were labelled accordingly. In our initial evaluation, we used similar but slightly different Likert scale questions (reported in the supplementary material), in which we asked participants to directly compare pairs of interfaces. However, in our final evaluation, we asked participants to rate one interface at a time, since this is more amenable to standard analysis techniques. They also completed the NASA-TLX instrument that measured the user's perceived workload. Using 5-point Likert scale questionnaires, we asked the participants on the role of stabilization, the utility of ambient reconstructions and Player's 3D body reconstruction on their spectating experience. At the end, we conducted a semi-structured interview with participants, that aimed at understanding how the interfaces compared with each other. We also welcomed any open-ended feedback on the interfaces and their study experience. In this paper, we only report the Likert scale responses and the NASA-TLX responses of the final user evaluation. Responses collected in the initial evaluation can be found in supplementary material.

*5.2.4 Final User Evaluation Results.* In this subsection, we report the statistically significant results from our final user evaluation.

*Analysis Methodology*: Since Likert-scale ratings are ordinal values, we first performed a Friedman test to determine if the interface condition had an overall effect on the measured rating. If an overall effect was found, we performed a post-hoc pairwise exact

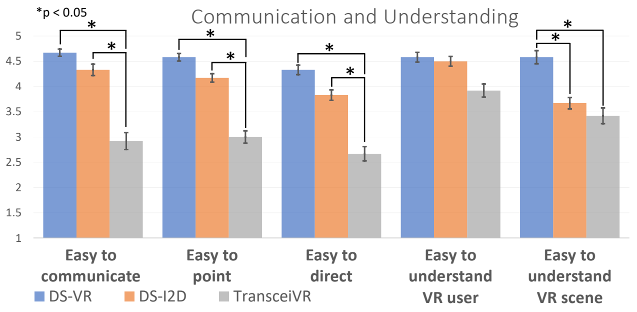

**Figure 7: 5-point Likert scale responses of participant self rated ease of (1) Communication with VR player, (2) Pointing to objects in the scene, (3) Directing the VR player, (4) Understanding VR player's actions and (5) Understanding the VR scene. The error bars denote Standard Error ($\sigma/\sqrt{n}$). A Likert Rating of 1 corresponds to 'strongly disagree' and 5 to 'strongly agree'. All intermediate ratings were labelled accordingly. All significance values were calculated at p < 0.05 after Bonferroni correction.**

Wilcoxon-Pratt signed-rank test between conditions to see if there was a significant difference. For the NASA-TLX scores, we first carried out a Repeated measures ANOVA to check for the overall effect of the interface. We then performed post-hoc pair-wise t-tests to check for individual differences. All p-values used for testing were adjusted with Bonferroni correction.

*Communication:* Friedman test revealed a significant effect of interface condition on the participant's ease of communication to the VR user ($\chi^2(2) = 19.5$, p<0.05). Post-hoc pairwise Wilcoxon signed-rank tests showed statistically significant differences between DS-I2D and TransceiVR (p<0.05, with a large effect size - $r = 0.61$) as well as between DS-VR and TransceiVR (p<0.05, with a large effect size - $r = 0.62$). Aggregate statistics are : $\mu_{TransceiVR} = 2.92$, $\sigma_{TransceiVR} = 1.16$, $\mu_{DS-I2D} = 4.33$, $\sigma_{DS-I2D} = 0.78$, $\mu_{DS-VR} = 4.67$, $\sigma_{DS-VR} = 0.49$.

Friedman test also revealed a significant effect of interface on participant's ease of referring to objects in the VR scene ($\chi^2(2) = 18.57$, p<0.05) as well as ease of directing the VR player($\chi^2(2) = 14.6$, p<0.05). The pairwise Wilcoxon tests for the ease of referring to objects in the VR scene found statistically significant difference between DS-I2D and TransceiVR (p<0.05, $r = 0.52$) as well as between DS-VR and TransceiVR (p<0.05, $r = 0.58$). Similarly Wilcoxon tests for ease of directing the VR player also showed significant differences between DS-I2D and TransceiVR (p<0.05, $r = 0.52$) as well as DS-VR and TransceiVR (p<0.05, $r = 0.58$).

These results indicate that our participants found it easier to communicate, point and direct with both DS-2D and DS-VR compared to TransceiVR.

*Spectator's understanding of VR scene and player*: Friedman test revealed a significant effect of interface condition on the spectator's ease of understanding the VR user's actions ($\chi^2(2) = 6.28$, p<0.05) as well as their ease of understanding the VR scene. Posthoc Wilcoxon tests did not yield any statistically significant differences for the ease of understanding the VR user's actions.

On other hand, Wilcoxon tests did yield a statistically significant difference for ease of understanding the VR scene between

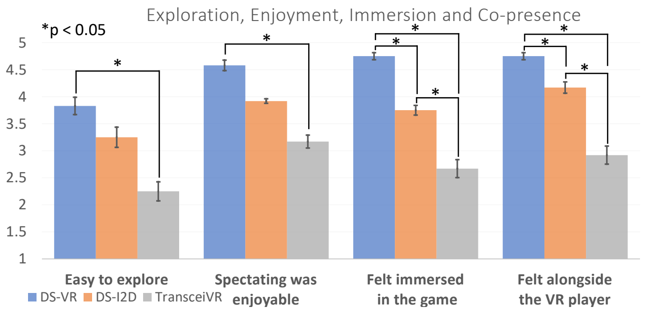

**Figure 8: 5-point Likert scale responses of participant rating of (1) Ease of independent exploration of the scene, (2) Enjoyable spectating experience, (3) Feeling immersed in the game with VR player and (4) Feeling present alongside the VR player in the game. The error bars denote Standard Error ($\sigma/\sqrt{n}$). A Likert Rating of 1 corresponds to 'strongly disagree' and 5 to 'strongly agree'. All intermediate ratings were labelled accordingly. All the significance values were calculated at p < 0.05 after Bonferroni correction.**

DS-VR and DS-I2D (p<0.05, $r = 0.63$) as well as between DS-VR and TransceiVR (p<0.05, $r = 0.56$). Aggregate statistics are: $\mu_{TransceiVR} = 3.42$, $\sigma_{TransceiVR} = 1.08$, $\mu_{DS-I2D} = 3.67$, $\sigma_{DS-I2D} = 0.78$, $\mu_{DS-VR} = 4.58$, $\sigma_{DS-VR} = 0.9$.

This implies that the choice of interface did not make a difference in easing our participants' understanding of the VR user's actions. However, they found it easier to understand the VR scene with DS-VR compared to both DS-I2D and TransceiVR.

*Scene exploration*: Friedman test revealed a significant effect of interface condition on the spectator's ease of exploring the VR scene ($\chi^2(2) = 10.55$, p<0.05). The pairwise Wilcoxon tests showed a statistically significant difference between DS-VR and TransceiVR (p<0.05, $r = 0.54$). Aggregate statistics are: $\mu_{TransceiVR} = 2.25$, $\sigma_{TransceiVR} = 1.21$, $\mu_{DS-VR} = 3.83$, $\sigma_{DS-VR} = 1.11$, $\mu_{DS-I2D} = 3.25$, $\sigma_{DSI2D} = 1.29$. This implies that our participants found it much easier to independently explore the VR scene using DS-VR compared to TransceiVR.

*Enjoyment*: Friedman test revealed a significant effect of interface condition on spectator's agreement that the spectating experience was enjoyable ($\chi^2(2) = 15.6$, p<0.05). The pairwise Wilcoxon tests showed a statistically significant difference between DS-VR and TransceiVR (p<0.05, $r = 0.61$). Aggregate statistics are: $\mu_{TransceiVR} = 3.17$, $\sigma_{TransceiVR} = 0.84$, $\mu_{DS-VR} = 4.58$, $\sigma_{DS-VR} = 0.67$. This implies that our participants found the spectating experience more enjoyable using DS-VR compared to TransceiVR.

*Immersion and Co-presence*: Friedman test revealed a significant effect of interface condition for spectator's agreement that they felt immersed inside of the game during the spectating experience ($\chi^2(2) = 21.54$, p<0.05) as well as for their agreement that they felt alongside the VR player ($\chi^2(2) = 20.15$, p<0.05). For both ratings, pairwise Wilcoxon tests found statistically significant differences between all pairs.

For Immersion, difference between DS-I2D and DS-VR was significant at p<0.05, $r = 0.62$; between DS-I2D and TransceiVR at p<0.05, $r = 0.59$; and between DS-VR and TransceiVR at p<0.05, $r = 0.63$.

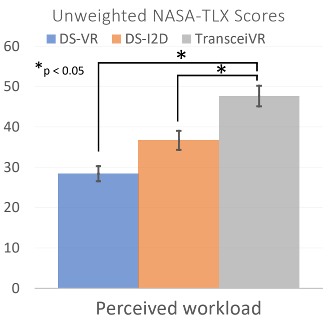

**Figure 9: NASA-TLX scores that measured participants' perceived workload while using each of the interfaces. The error bars denote Standard Error ($\sigma/\sqrt{n}$). Note: A Higher perceived workload for an interface means that the participants found it more difficult to use. The maximum score possible is 100.**

Aggregate statistics are: $\mu_{TransceiVR} = 2.67$, $\sigma_{TransceiVR} = 1.15$, $\mu_{DS-I2D} = 3.75$, $\sigma_{DS-I2D} = 0.62$, $\mu_{DS-VR} = 4.75$, $\sigma_{DS-VR} = 0.45$.

For agreement to feeling alongside the VR player, difference between DS-I2D and DS-VR was significant at p<0.05, $r = 0.54$; between DS-I2D and TransceiVR at p<0.05, $r = 0.59$; and between DS-VR and TransceiVR at p<0.05, $r = 0.64$. Aggregate statistics are: $\mu_{TransceiVR} = 2.92$, $\sigma_{TransceiVR} = 1.16$, $\mu_{DS-I2D} = 4.17$, $\sigma_{DS-I2D} = 0.72$, $\mu_{DS-VR} = 4.75$, $\sigma_{DS-VR} = 0.45$.

These results imply that our participants felt that they were most immersed in the game and alongside the player when they used DS-VR, followed by DS-I2D and then TransceiVR.

*NASA-TLX*: We used the unweighted version of the NASA-TLX scores. A Higher perceived workload score for an interface means that the participants found it more difficult to use. Mauchly's test did not show a violation of sphericity against the interface conditions (W(2) = 0.65, p = 0.12). With one-way repeated-measure ANOVA, we found a significant effect of interface on the NASA-TLX scores (F(2,22)=11.51, p<0.05, $\eta^2_{partial} = 0.51$). Pairwise t-test results found a significant difference between DS-I2D and TransceiVR ($t(11) = 4.19$, p<0.05, Cohen's d = 1.21) as well as between DS-VR and TransceiVR ($t(11) = 4.056$, p<0.05, Cohen's d = 1.17). Aggregate statistics are: $\mu_{TransceiVR} = 47.64$, $\sigma_{TransceiVR} = 17.68$, $\mu_{DS-I2D} = 36.67$, $\sigma_{DS-I2D} = 16.36$, $\mu_{DS-VR} = 28.4$, $\sigma_{DS-VR} = 12.97$. This indicated that our participants perceived a higher workload to interact when they used TransceiVR compared to both DS-VR and DS-I2D.

## 5.3 Discussion of User and Expert Evaluation

In this subsection, we qualitatively discuss and compare the different interfaces.

*5.3.1 Overall interface preference.* In the initial user study, five of eight participants rated the VR interface of DreamStream (DS-VR) as their most preferred viewing mode. Two preferred Follow view of the Interactive 2D viewer (DS-I2D), and one preferred the standard VR mirror. In our final user evaluation, nine of twelve participants rated the VR interface of DreamStream (DS-VR) as their first preference, one each preferred first-person view and follow view of the interactive 2D viewer (DS-I2D) and one preferred TransceiVR. In further subsections we break down the factors that provide insights to this preference.

*5.3.2 Understanding the VR scene and actions of the VR player.* In our final study, we found no statistically significant difference among the three interfaces in the spectator's ease of understanding the actions of the player. However, we did find statistically significant differences in the spectator's ease of understanding the VR scene when using DS-VR compared to when using DS-I2D and TransceiVR. We observed no significant difference between DS-I2D and TransceiVR.

TransceiVR uses *angle Frames* to allow a spectator to view parts of the VR scene other than that of the VR user's current view. These are static frames captured when the VR player last looked along certain directions. While these can be useful for collaboration in relatively static scenes, they are less so in the more dynamic environments of games: terrain and other elements frequently change, and players often teleport and change orientation through in-game controls. Both these interactions tended to break the *angle frames* interaction in TransceiVR. For example, the VR player can be looking straight and exiting a cave into an open terrain. However, since the VR player had not turned around in this process, *angle frames* would still display the outdated frames of the cave. Another issue is that the TransceiVR system relies on changes in the pose of the headset in order to update the *angle frames*. Therefore, any change in orientation or position caused by using the controller (such as teleportation) will not be tracked by TransceiVR, and a previously captured frame's view direction may now be unrelated to that of the current view. Both of these interactions are common in VR games and interfere with the spectator's understanding of the scene. P3: "*I would have loved the TransceiVR interface if the images below [Angle frames] had updated in real-time. But they were delayed. So I could not use them appropriately and ultimately did not use it.*" P11: "*With TransceiVR, the latency of interactions were too high with moving objects. When we were searching for chickens, for example, the [angle frames] on the bottom were not updating fast enough.*" P6: "*With TransceiVR there's a lot of frustration when the frames [angle frames] are not updating. It happens when the player has not looked in a direction for a while. It's a few more layers away from the player.*"

With DS-I2D, ambient reconstructions gave spectators a comparatively more live view of the player's surroundings. But there was still no significant difference in scene understanding between TransceiVR and DS-I2D. While both DS-VR and DS-I2D rendered the same information, participants felt DS-VR to be easier than DS-I2D for understanding the VR scene. They attributed this to an increased immersion through which they could better understand the depth and scale of the scene as a whole, as well as to the ease of interaction and independent exploration. P4: "*In [DS-VR], I felt that I was much more on the same plane as the [VR Player]. I had the same ability as the [VR Player], so I could easily relate to the way [VR Player] was looking around in the world. So instead of having this omniscient view [DS-I2D], I felt more like another person in the scene. I was able to perceive the same things in the same way as the [VR Player] did. I could get a more sense of depth, and I was able to better perceive my relative distance to the [VR Player].*" This was also noted by one of our experts, E3: "*I would say that [DS-VR] fixes the fact that normally viewers can't see the 3D effects in the depth. Like, which is the big thing. It never comes across properly on a 2d screen. [DS-VR] means they're in VR with the [VR Player]. They're going to get the 3d depth of that, and completely understand the world they are in. It also means that, they can help the [VR Player]. They can guide them. So it actually adds more interaction than a standard stream, which are just text chats.*"

*5.3.3 Communicating with the VR player.* In our final study, participants found it easier to communicate with the VR player with both DS-I2D and the DS-VR, when compared to TransceiVR. Spectators similarly found it easier to point scene elements and direct the VR player's attention. There were no statistically significant differences between DS-I2D and DS-VR. We observed similar responses in our initial study, in which participants found it easier to communicate, point and direct using DS-VR and DS-I2D compared to VR mirror. While the results of the initial study were not surprising, we found the results of the final study to be surprising. TransceiVR is primarily designed to facilitate communication with the VR user, so we expected that annotation and shared screen features would enable levels of communication as good as both DreamStream interfaces.

From the study, we found that the lack of effectiveness of TransceiVR is due to two key factors. First is the difficulty in understanding the VR scene, as described above. The lack of regularly updated views in the more dynamic scenes and interactions of games makes it more difficult for spectators to quickly and easily parse the VR scene. This impacts their ability to easily communicate with the VR player regarding the VR scene.

Second, TransceiVR spectators can communicate with the VR player by creating annotations on the live feed. However, these annotations do not track the changes in player orientation. As a result, annotations quickly move out of position as the player teleports, translates or rotates in the game. There is no mechanism to correct the positions of these annotations after placement and so they must be cleared and re-drawn if they are to persist. Furthermore, the live freed is frozen during the time that the annotation is begun, resuming after a 500ms time out. Alternatively, spectators can share annotated frames as separate screenshots to be placed in the scene, but this is somewhat more cumbersome.

As in TransceiVR, DreamStream has access to only the system-level VR poses of the player's HMD and their controllers. DreamStream's VR and I2D interfaces' dynamic laser pointer feature offers some functionality related to TransceiVR's static annotations. Laser pointers are rendered into the live feed of the scene, and spectators can quickly adapt their pointing in response to changes in the scene. Furthermore, laser pointer interaction is simpler, easier to trigger (right-click on DS-I2D, and trigger on DS-VR), and less time consuming, and is therefore more suited to fast-paced VR experiences that have dynamic elements and interactions. Experts as well as users from both our studies noted this difference. I5: "*In VR, the laser pointers are so nice because it's easier and faster for me to use it 360 degrees and then just tell the [VR Player] to look for my laser pointers.*" E2: "*The laser pointers in [DreamStream] could really help with new kinds of engagement. I mean, this is so much more engaging than just the chat.*" P11:"*In the VR experience, I think a laser pointer was more accurate in terms of having feedback and it felt more like I could directly communicate with the [VR Player] about the scale of* *their surroundings*." P5: "*The pointing in TransceiVR freezes the view and I do not see the live scene anymore. If the player then moves, the annotation does not follow them nicely. Whereas in the DreamStream there is a real-time pointer which is [more apt and helpful].*"

In summary, while TransceiVR allows for conveying more complex information, it requires more effort and time, and fails for dynamic scenes. However, DreamStream's laser pointer allows for conveying simple information with less effort and works for dynamic scenes.

*5.3.4 Immersion, Enjoyment, Co-presence and Independent exploration.* In our final study, we see statistically significant differences across all interfaces in spectator ratings of immersion in the game as well as their rating of feeling alongside the VR player. They felt most immersed and felt most alongside the VR player in DS-VR, followed by DS-I2D, with TransceiVR the least. We observed a similar trend with our initial study participants. We believe this is because the DS-VR interface allowed participants to have a more symmetrical spectating experience compared to other interfaces. DS-I2D allowed for more control over their viewport compared to TransceiVR. Immersion is also influenced by the participant's ability to perceive and understand the VR scene, as well as to communicate with the VR player, as discussed above. While DS-I2D and DS-VR offered similar information, participants noted two key differences between them that made the former more immersive.

Firstly, with DS-VR, participants could better understand the depth and scale of things in the environment, as well as the ambient surrounding in which the VR player operated. P4's comment in section 5.3.2 also reflects this. I1 also says "*[DS-VR] helped me feel more immersed, I think, because I felt like I was fully in the environment. I didn't feel like just an external spectator.*".

A second key difference was that participants felt controls in VR to be easier, more natural and intuitive. P3: "*[DS-VR] was much easier to use, and I could feel I was with the [VR Player] more than the iPad [TransceiVR] or Desktop [DS-I2D] interfaces. When I want to move or rotate my view quickly with iPad or Desktop, it's pretty hard. When controlling using the mouse or keyboard, its like different from the real world. In real world, when I look around, I just turn like this. I was able to do that in VR. But with the Desktop and the iPad interface, I had to click or do some action.*" This was also seen with ease of independent exploration. We found a statistically significant difference only between DS-VR and TransceiVR. Both DS-VR and DS-I2D allows for free exploration of the VR scene, and hence we did not expect a difference between them. However, for the same reasons mentioned in section 5.3.2, we expected both DS-VR and DS-I2D to outperform TransceiVR. This was not the case. From participant comments we understand that while DS-I2D allowed for free exploration of the VR scene, there was a cost involved when they had to switch back and forth between what the player is seeing and their own exploration. They noted that this cost in DS-I2D is much higher compared to DS-VR. In DS-I2D this meant changing viewing modes or using mouse/keyboard to manually change viewports every time. P8: "*In VR [DS-VR] I only had to look away to detach from the [VR Player's] view. When I wanted to see the [VR Player's] view, I could easily turn back. It was really easy for me to switch back and forth between independently exploring and looking at what the [VR Player] did.*"

Better perception, ease of control and an overall more symmetrical experience likely leads to more immersion in DS-VR. Participants felt more like they were with the player, alongside them. I7:"*VR appealed to me the most, because it felt like we [VR Player and I7] were together doing something. We were experiencing the same thing*". E3 summarized this as "*The idea that [in DS-VR], they can see what I'm seeing and they can move independently around the scene, maybe play the game with me, can actually see things I missed out, and point to it. These are just groundbreaking.*"

In terms of enjoyment, the only significant difference was between DS-VR and TransceiVR, where participants found DS-VR more enjoyable. Participants rated DS-I2D somewhere between these two interfaces but the pairwise comparison with either was not significant. Through our interviews, we attribute this to the fact that the VR interface offered increased immersion as well as more intuitive usage.

*5.3.5 Perceived Workload.* All experts mentioned the issue of shaky video feeds, and lower field of view (FOV) of the VR scene as the two biggest issues that spectators face with VR streams. Experts noted that they currently resort to withholding information from the spectators for the same reason - E1: "*One of the unwritten rules of streaming - Don't talk about something [spectators] can't or didn't see. They are already shackled by the fact that they're looking through a flat display with a reduced FOV*".

DreamStream addresses this issue by decoupling the VR player's head motion from that of the spectator. This makes spectator feeds easier to watch. Accordingly, we observed a significant difference between the perceived workload as measured by NASA-TLX between TransceiVR and the DreamStream's interface. We also saw a significant difference between perceived workload of DS-VR and DS-I2D. We believe that this is due to the relatively larger number of keyboard controls and viewing modes in the latter. A few participants recommended that a joystick controller be used as input instead of a keyboard and mouse for DS-I2D. In summary, DS-VR was better for viewing and interaction. DS-I2D was also good for viewing but made interaction more difficult. TransceiVR was difficult to view and also made interaction more difficult.

*5.3.6 Reconstructions of Ambient parts of the scene.* All experts agreed that reconstructions of the ambient parts of the scene, add value to the spectating experience. E1: "*I think the value of this [Ambient reconstructions] for something like Half Life Alyx or Skyrim really stands out because it gives the spectator the peripheral fill-in information that, even if less accurate, even if less frequently refreshed, is very valuable. For example, they can see if there's a Wolf running up behind me, even if it was like low frame rate.*" User study participants had somewhat more varied opinions on its value. In the 5-point Likert-Scale questionnaires, Eight of twelve participants agreed or strongly agreed that ambient information made it easier to understand the scene. Three were neutral and one disagreed. Participants who agreed felt that ambient information helped because they did not have to rely on the player's view. I3: "*[Ambient reconstructions] made it more comfortable to see the full scene without, a black void. So it was more enjoyable that way as well. It also gave me a little more awareness of where I was and where I was facing*".

But they also noted some limitations. First was its lower frame rate. P11 noted that they turned off ambient temporarily when the scene had too many fast moving objects. P11: "*I enjoyed it more with the ambient reconstructions. I tried to keep it on most of the time, mainly because not only was it enjoyable, but it helped orient me a little bit more. In action scenes I had to turn it off so that I could focus on that person that the [VR Player] was trying to hit, but then I'd turn it back on if I wanted to just check up the surroundings and navigate through to the top of that mountain. Ambient also helped me forget about how enclosed the field of view was. So it was definitely helpful for just making the field view less limiting for my experience.*"

The second issue we observed was that Skyrim's rendering occasionally will omit objects that are not in the player's field of view. This will cause far away terrain, for example, to pop out of view as the player moves towards it. In our user evaluations, some participants noticed this effect.

Aside from these issues, P1 noted that they would not use ambient reconstruction because "*The fun part about spectating games is the shared experience. So I don't find a lot of value in looking at other parts of the scene, I guess. So just like being with the [VR Player] and seeing what they see, I think is the most fun part of the shared gaming experience.*"

*5.3.7 Player live 3D reconstruction.* Participants' opinions on the value of the player's live 3D reconstruction were varied. In the 5-point Likert-Scale questionnaires, seven out of twelve participants agreed or strongly agreed that the VR player's 3D reconstruction made the experience enjoyable, while three were neutral and one disagreed. Eight felt that it made it easy to understand the VR user's actions while three disagreed (one remained neutral). Some participants felt that the rendering of the player added social value to the experience, but not much value in understanding their actions. They felt that it sometimes gave an idea of what the VR player can physically do, such as whether an object was within reach. P6: "*It definitely helps to see see how a friend [VR Player] is behaving in real life when they are playing the game. With productivity of interactions, it only marginally helps. I definitely have a better picture of how and what the player is doing. Whether he is capable of reaching certain points. I couldn't figure out the emotions because their facial expressions were covered. It helps understand the body gestures, but still I am missing the face.*" While some participants such as P4 felt it added to the co-presence of the experience, they also noted that the reconstruction occludes their view at times. P4: "*It helped that I could feel that I was with the [VR Player], the feeling that we [VR Player and P4] are together was really enhanced with that. But whenever the [VR Player] is occluding some of my views, it was not nice.*" They also noted that this can potentially break the immersion of the game. P1: "*Having the [VR Player] sitting there kind of took me out of the world of the game. Like, I know, the [VR Player] is running around murdering chickens and stuff. But then the [VR Player] is just like sitting in their chair that doesn't really connect.*"

*5.3.8 Preference within spectating modes in 2D displays.* Within the different modes of spectating in DS-I2D, participants had widely varying preferences, and described the trade-offs between them. We counted the number of users who ranked each of the viewing modes as their first or second preference interface. Highest was Follow view with six users, followed by free view with three users. Follow view was popular because it allowed participants to sit back and passively observe the action. Follow view also provided a greater field of view, with minimal controls of height, distance, pitch and yaw of the camera. Often these were set once, switching to other modes if they wished to explore the scene. I4: "*I stuck to the follow and free view the most. In that, I can not only see the [VR Player], but I also see the things around them that they aren't seeing. That gives me a much better perspective and allows me to direct them better...I would switch to the free view was primarily to explore and interact*" P6: "*The Follow view synchronizes my view with the players view. That's very nice, and eases burden from using arrow keys. Follow view also gives more information than first person view. First-person view always restricts me with the players view*" The camera stabilization deployed in DreamStream received mixed feedback. While some users liked that it smoothed any jarring motion, they also complained that many times it broke the authenticity of a fast-paced actions by the VR player.

*5.3.9 Value of platform level support.* All experts expressed strong positive opinion about DreamStream's ability to operate at a platform level. They mentioned that, without this, it is hard to gain any traction for such a spectating experience. As an example, three of them cited *vReal*[7], a system providing some functionality similar to DreamStream, but which required developer support in each game. E1 mentions "*Even if one provided them[Developers] the perfect tool set and said, just write one line of code, one wouldn't even get to 80%, coverage of all the apps.*" This was also noted by participants who had played or watched similar games before. P6: "*Having spectators view it in VR, gives an entirely different experience. Some games are bound to be played as a single person. It's quite hard to allocate tasks [to spectators]. This interface [DS-VR] helps with that. It allows playing a single player game together [VR player and spectators]. It allows experiencing what a player experiences.*"

*5.3.10 Impact of noise and defects in scene reconstruction.* During the interviews, we asked all participants how much the noise and defects in scene reconstruction affected their experience. All except one participant felt that though noise and defects affected their experience, the increased immersion and interactivity made the experience better for them. Some users noted that artifacts were not uniform across distances and perspectives. P3 noted that, in DS-VR where they do not have an option to be in first person, noise seem more pronounced when they took perspectives far away from that of the player, but they also noted that the added immersion and interactivity is worth it. Other participants as well as experts we interviewed had similar opinions. E4: "*I do not think that the artifacts would detract a crazy amount from the viewer experience. The immersion that viewers gain from this would help so much more*".

## 6 LIMITATIONS

### 6.1 Artifacts in 3D rendering

Creating a 3D reconstruction from depth buffers has the advantage that it leads to a generic approach for streaming 3D data that leverages existing video codecs for compression and works across many if not most VR applications. Transmitting the depth buffer has the added advantage that it may be used to perform further rendering or compositing at either the player or spectator sites, such as rendering avatars.

[7]vReal, https://vreal.net/

The main limitation of this approach is that the depth buffer is only a complete 3D representation of the scene from the VR player's point of view. A spectator standing the side of the VR player may notice gaps in the geometry (see Figure 4a). These gaps will appear larger as the spectator moves further away from the VR player. The secondary, ambient reconstruction can effectively fill in these gaps but presently DreamStream is limited in how often this alternate view is rendered. We note that it may be possible to cache texture and geometry information to fill these gaps, using computer vision and video analysis techniques.

A second related problem is that when rendering 3D objects from a depth buffer, care must be taken around discontinuity in depth at the edges of objects. Spurious geometry that straddles the edge of an object and the background scene can lead to highly visible artifacts. This can be reduced by omitting triangles that exhibit an unreasonably large change in depth, but in practice eliminating all such artifacts without introducing new ones by setting a simple threshold is difficult. Again, this might be addressed by employing computer vision techniques for object segmentation.

Both of these problems might be addressed by analyzing scene geometry as it is drawn. However, this requires knowledge of how geometry is laid out in memory, and how the shaders of the application work. This will be difficult to do in general.

A final issue is that during rendering of the ambient view, the game may skip rendering of some objects. Such optimizations are common in 3D rendering, and DreamStream has no control over it.

### 6.2 Scaling to multiple users

While DreamStream supports multiple users, our user study focused only on a one-one interactions between a VR player and a spectator. This is a limitation of our user study. In practise, multiple spectators may use DreamStream to spectate the VR player. Each of them can either use a VR headset or a desktop interface, and would be represented in the scene through avatars. A further study is required to shed light on how DreamStream performs in these multi-user scenarios. During the feedback with experts, a challenge they noted is that unlike text chat, having hundreds of user avatars in a virtual space might not be feasible. It then becomes important for the player to either manually moderate or, for the system to automatically determine, which spectators the player can see and have beside them. It is also unclear how the interactions among spectators might play out in such a scenario. Interesting access control and social dynamics emerge, requiring careful design choices. For example, a few "super-spectators" may be admitted, whereas the rest can be part of general audience, and view it as they would in a 3D theatre.

## 7 CONCLUSION

In this paper, we identify key considerations for the design of VR spectator systems. We present DreamStream that operates at a platform level to achieve immersive and interactive spectating for VR experiences. We propose using the depth buffer to capture, transmit and reconstruct the geometry of the 3D scene. Using this approach, DreamStream allows viewers to spectate using both 2D desktop-based UI as well as immersively using a VR-based viewer. Through our user and expert studies, we found that the VR viewer is the preferred interface and can offer multiple advantages. We hope that our findings inspire future progress in developing VR spectating systems that offer increased immersion.

## ACKNOWLEDGMENTS

We thank Prof. Björn Hartmann for his valuable feedback during the different stages of the project. We thank Aiswarya Goutham Gouthaman for her help with pilot testing of DreamStream as well as with the production of video and image assets used in the work. Finally, we would like to thank the experts and study participants for their time.

## REFERENCES

[1] Steve Benford, John Bowers, Lennart E Fahlén, Chris Greenhalgh, and Dave Snowdon. 1997. Embodiments, avatars, clones and agents for multi-user, multi-sensory virtual worlds. *Multimedia Systems* 5, 2 (1997), 93–104.

[2] Steve Benford, John Bowers, Lennart E. Fahlen, John Mariani, and Tom Rodden. 1994. Supporting cooperative work in virtual environments. *Comput. J.* 37, 8 (1994), 653–668.

[3] Steve Benford, Ian Taylor, David Brailsford, Boriana Koleva, Mike Craven, Mike Fraser, Gail Reynard, and Chris Greenhalgh. 1999. Three Dimensional Visualization of the World Wide Web. *ACM Comput. Surv.* 31, 4es (dec 1999), 25–es. https://doi.org/10.1145/345966.346021

[4] Christer Carlsson and Olof Hagsand. 1993. DIVE A multi-user virtual reality system. In *Proceedings of IEEE virtual reality annual international symposium*. IEEE, 394–400.

[5] Gifford Cheung and Jeff Huang. 2011. Starcraft from the Stands: Understanding the Game Spectator. 763–772. https://doi.org/10.1145/1978942.1979053

[6] Katharina Emmerich, Andrey Krekhov, Sebastian Cmentowski, and Jens Krueger. 2021. *Streaming VR Games to the Broad Audience: A Comparison of the First-Person and Third-Person Perspectives.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445515

[7] Barrett Ens, Joel Lanir, Anthony Tang, Scott Bateman, Gun Lee, Thammathip Piumsomboon, and Mark Billinghurst. 2019. Revisiting collaboration through mixed reality: The evolution of groupware. *International Journal of Human-Computer Studies* 131 (2019), 81–98.

[8] Ching-Ling Fan, Wen-Chih Lo, Yu-Tung Pai, and Cheng-Hsin Hsu. 2019. A Survey on 360° Video Streaming: Acquisition, Transmission, and Display. *ACM Comput. Surv.* 52, 4, Article 71 (aug 2019), 36 pages. https://doi.org/10.1145/3329119

[9] Susan R. Fussell, Leslie D. Setlock, and Robert E. Kraut. 2003. Effects of Head-Mounted and Scene-Oriented Video Systems on Remote Collaboration on Physical Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. Association for Computing Machinery, New York, NY, USA, 513–520. https://doi.org/10.1145/642611.642701

[10] Jan Gugenheimer, Evgeny Stemasov, Julian Frommel, and Enrico Rukzio. 2017. ShareVR: Enabling Co-Located Experiences for Virtual Reality between HMD and Non-HMD Users. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4021–4033. https://doi.org/10.1145/3025453.3025683

[11] Jan Gugenheimer, Evgeny Stemasov, Harpreet Sareen, and Enrico Rukzio. 2018. *FaceDisplay: Towards Asymmetric Multi-User Interaction for Nomadic Virtual Reality.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173628

[12] Carl Gutwin and Saul Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3 (2002), 411–446.

[13] Jeremy Hartmann, Christian Holz, Eyal Ofek, and Andrew D. Wilson. 2019. *RealityCheck: Blending Virtual Environments with Situated Physical Reality.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300577

[14] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-Camera Units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 637–644. https://doi.org/10.1145/2642918.2647383

[15] Ikuo Kamei, Changyo Han, Takefumi Hiraki, Shogo Fukushima, and Takeshi Naemura. 2020. CoVR: Co-Located Virtual Reality Experience Sharing for Facilitating Joint Attention via Projected View of HMD Users. In *SIGGRAPH Asia 2020 Emerging Technologies* (Virtual Event, Republic of Korea) *(SA '20)*. Association for Computing Machinery, New York, NY, USA, Article 14, 2 pages. https://doi.org/10.1145/3415255.3422883

[16] Shunichi Kasahara and Jun Rekimoto. 2014. JackIn: Integrating First-Person View with out-of-Body Vision Generation for Human-Human Augmentation. In *Proceedings of the 5th Augmented Human International Conference* (Kobe, Japan) *(AH '14)*. Association for Computing Machinery, New York, NY, USA, Article 46,

8 pages. https://doi.org/10.1145/2582051.2582097

[17] Simone Kriglstein, Günter Wallner, Sven Charleer, Kathrin Gerling, Pejman Mirza-Babaei, Steven Schirra, and Manfred Tscheligi. 2020. Be Part Of It: Spectator Experience in Gaming and Esports. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3334480.3375153

[18] Cuong Nguyen, Stephen DiVerdi, Aaron Hertzmann, and Feng Liu. 2017. CollaVR: Collaborative In-Headset Review for VR Video. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 267–277. https://doi.org/10.1145/3126594.3126659

[19] Christopher Niederauer, Mike Houston, Maneesh Agrawala, and Greg Humphreys. 2003. Non-Invasive Interactive Visualization of Dynamic Architectural Environments. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics* (Monterey, California) *(I3D '03)*. Association for Computing Machinery, New York, NY, USA, 55–58. https://doi.org/10.1145/641480.641493

[20] Thomas Smith, Marianna Obrist, and Peter Wright. 2013. Live-Streaming Changes the (Video) Game. In *Proceedings of the 11th European Conference on Interactive TV and Video* (Como, Italy) *(EuroITV '13)*. Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/2465958.2465971

[21] Samantha Stahlke, James Robb, and Pejman Mirza-Babaei. 2018. The Fall of the Fourth Wall: Designing and Evaluating Interactive Spectator Experiences. *International Journal of Gaming and Computer-Mediated Simulations* 10 (07 2018), 42–62. https://doi.org/10.4018/IJGCMS.2018010103

[22] Hongling Sun, Yue Liu, Zhenliang Zhang, Xiaoxu Liu, and Yongtian Wang. 2018. Employing Different Viewpoints for Remote Guidance in a Collaborative Augmented Environment. In *Proceedings of the Sixth International Symposium of Chinese CHI* (Montreal, QC, Canada) *(ChineseCHI '18)*. Association for Computing Machinery, New York, NY, USA, 64–70. https://doi.org/10.1145/3202667.3202676

[23] Burak S. Tekin and Stuart Reeves. 2017. Ways of Spectating: Unravelling Spectator Participation in Kinect Play. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1558–1570. https://doi.org/10.1145/3025453.3025813

[24] Sonoda Tetsuri and Anders Grunnet-Jepsen. [n.d.]. Depth image compression by colorization for Intel® RealSense™ Depth Cameras. https://dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras

[25] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating Remote Instruction of Physical Tasks Using Bi-Directional Mixed-Reality Telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 161–174. https://doi.org/10.1145/3332165.3347872

[26] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Björn Hartmann. 2019. TutoriVR: A Video-Based Tutorial System for Design Applications in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300514

[27] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Bjoern Hartmann. 2020. TransceiVR: Bridging Asymmetrical Communication Between VR Users and External Collaborators. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 182–195. https://doi.org/10.1145/3379337.3415827

[28] Chiu-Hsuan Wang, Chia-En Tsai, Seraphina Yong, and Liwei Chan. 2020. *Slice of Light: Transparent and Integrative Transition Among Realities in a Multi-HMD-User Environment.* Association for Computing Machinery, New York, NY, USA, 805–817. https://doi.org/10.1145/3379337.3415868

[29] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Spacetime: Enabling Fluid Individual and Collaborative Editing in Virtual Reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 853–866. https://doi.org/10.1145/3242587.3242597