Designing communication add-ons for VR applications using VR-embedded Widgets

Cuong Nguyen
Adobe Research
San Francisco, CA, USA
cunguyen@adobe.com

Stephen DiVerdi
Adobe Research
San Francisco, CA, USA
diverdi@adobe.com

ABSTRACT

A software application rendered on a physical screen can naturally support social activities. Multiple users can watch the screen together, exchange ideas, or provide guidance. These interactions, however, are not easily supported in applications rendered in an enclosed Virtual Reality system. Information about how a user is using the application in VR such as the screen output and the controller interactions are not accessible to other users. Worse, users outside of VR also cannot easily interact with the user in VR. These barriers prevent fluid collaborative interactions around a VR application. This position paper discusses our ongoing effort in making VR applications more social using *VR-embedded widget*. These widgets are interactive interface elements that can be rendered on top of an existing VR application. They allow a VR application to interface with other applications without modifying the source code. We will motivate our initial effort in using VR-embedded widget in developing a tutorial system for VR painting, and illustrate several potential ideas and opportunities to apply VR-embedded widgets.

KEYWORDS

virtual reality, collaboration

CHI'19 Extended Abstracts, May 4-9, 2019, Glasgow, Scotland UK

Figure 1: Many social and collaborative activities can happen naturally around a software application without explicit support from the developer. How can we enable this type of collaboration in current VR applications, which are often rendered in en enclosed and isolated HMD? [Creative Commons], via Knight Foundation on Flickr. (http://flic.kr/p/avxjtF**).**

INTRODUCTION

Many communicative, collaborative, and social interactions can happen around a software application without explicit support from the developer. Consider a creative director discussing a new Photoshop sketch with her designer. They both look at a same design, rendered on the interface canvas on the computer screen. The designer can point on the canvas and brief the director on what's new. The director can sometimes take over the computer input to highlight issues or make small changes.

This communal experience of using a software application is mostly missing from Virtual Reality (VR). VR head-mounted display (HMD) technologies have matured enough for VR software development to grow rapidly, with applications in games, art, design, and healthcare. However, an application rendered in an enclosed VR HMD puts the user in isolation. Both the application output and the VR controller input are often accessible exclusively to the VR user. It is difficult for a user in VR and another user on any other platforms to exchange information, discuss, or interact with the application together. Simple gestures such as telling a friend in VR how to trigger a teleport move using use the VR controller often result in awkward and inefficient hand-holding moments.

Contemporary approaches to make VR applications less isolated often rely on built-in components that are tied into a specific application such as the Studio Share feature in Oculus Medium [1]. Other solutions explore instrumented hardware to capture and project VR content to outside users [4]. Some researchers have also begun to propose new design concepts that can better support collaborative tasks [7, 10], which requires rebuilding existing applications from the ground up.

How can we support social activities for VR applications without modifying their source code? We propose that one way to achieve this is to support communication add-ons in VR. We envision these add-ons to be usable and accessible in a VR application without interrupting the application process. In this way, in addition to using the VR application, a user could also access communication features such as chat box, video stream, and annotations to communicate with other user without stopping her current work in the HMD.

Designing VR communication add-ons raises three challenging questions. First, what would be the user mental model when multitasking between a VR application and a communication add-on? Second, how do we make sure the add-on is truly usable and accessible when it runs in parallel with a VR application that can completely immerse the user view and interactions. Third, what are the potential social and collaborative applications that these add-ons can support?

In order to explore these questions, we have developed a set of interface building blocks called *VR-embedded widgets*. Briefly, these widgets are interactive interface elements that can be rendered on top of an existing VR application, allowing us to experiment and build communication add-ons without modifying an existing VR application. In the next sections, we will explain VR-embedded widgets in more details and discuss potential social applications.

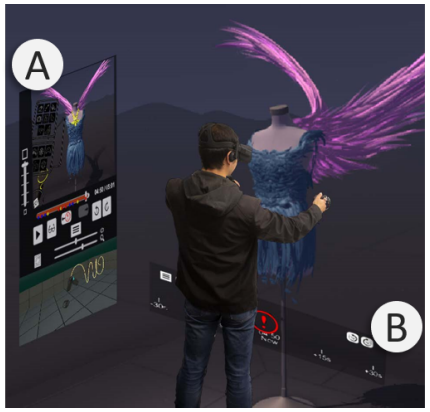

Figure 2: We have explored using VR-embedded widgets to design a video tutorial system for VR painting [6]. Here, a user is creating a 3D painting in a VR application Tilt Brush (https://www.tiltbrush.com/**). We rendered several video player widgets (A, B) directly into this application to help him easily watch and learn tutorial content.**

VR-EMBEDDED WIDGETS

At its core, VR-embedded widgets are similar to traditional graphical user interface (GUI) widgets. They are both interface elements that display information and can interact with the user [9]. However, because our widgets are designed to be used in VR and in parallel with a VR application, certain aspects of VR-embedded widgets are fundamentally different than widgets designed for a 2D screen, which we will outline below.

Appearance and placement

On a desktop computer, an application is often shown as a window. In VR, an application is a world—one that surrounds and immerses the user. Thus, the interfaces of the communication add-on need to be rendered as a part of the world. To do this, we leverage the OpenVR API[2] to access VR transformation matrices of the current running application. These matrices include the HMD and the left and right hand controllers. Using this information, we can render the UI elements into the back buffer of the current application and make it appear as a part of the scene. The UI can be anchored to the 3D world as a 3D panel (world-fixed, Figure 2A), to the user's HMD as a HUD display (view-fixed, Figure 2B), or to the user's controller as a hand-held UI (hand-fixed).

A trade-off of this approach is that the UI elements will always overlay the graphics of the running application. In a stereoscopic-rendered VR environment, it can cause depth conflict problems—visual discomforts that occur when the widgets overlay contents that are close to the viewer. One way to alleviate conflicts is to acquire the depth buffer exposed by the VR SDK and adjust the rendering of the widget based on solutions proposed by Nguyen et al. [8].

Interaction

The OpenVR API also exposes interaction data of the left and right hand controller. These data include translation, rotation, and button presses. By listening to these interaction events, we can program VR-embedded widgets to respond to user input in the HMD. An important point to note is that, unlike in commercial overlay systems such as Dash [3], the user does not have to pause the current running VR application to interact with the widget. As a result, tasks that require frequent switch between the VR application and the widget can be carried out more seamlessly.

DESIGNING COMMUNICATION ADD-ONS USING VR-EMBEDDED WIDGETS

By having widgets that integrate directly into the user's HMD, we can extend a VR application's communication capability without modifying its source code. The extent to what these applications can and cannot do depend on the level of integration of the widget. Currently, our widgets can access publicly available data from OpenVR such as the application display view, the rendering matrices,

and the controller interaction. In future, we also expect the view associated depth map data. With these rudimentary data, below, we will describe our first exploration with VR-embedded widget and illustrate a range of social applications. Higher-level data such as tool uses, scene map, or enemy health can be made available from the application developer and further enrich the design possibilities.

Tutorial systems

In TutoriVR [6], we leveraged VR-embedded widgets to design a video tutorial system for 3D design tasks in VR. 3D painting, sketching, and sculpting applications in VR have recently received significant interest from the creative community. Currently, many users seek community-posted videos on YouTube to explore and learn new creative skills of VR design. However, users still do not have a convenient way to browse, watch, and learn these videos directly in the HMD. They either have to pause the VR application to switch to another video player software, or need to take off the HMD to watch the video.

TutoriVR addressed this problem by showing a tutorial video player in the user's VR design application using VR-embedded widget. Figure 2 illustrates our system running on top of Tilt Brush. In addition to the video player, we were able to experiment with a number of new widgets to enhance the user learning experience. For example, the Perspective Thumbnail (Figure 2A) widget allows us to better visualize the 3D strokes of the tutorial author in the video. The Awareness widget (Figure 2B) is a view-fixed panel that helps a user to keep track of the video tutorial progress even when she is focusing on the painting and not the video.

Other applications

Most VR systems can mirror the application output onto a display view, so a user outside of VR can get a sense of what the user inside in VR is doing. However, the outside user cannot easily communicate or interact with the VR user. By using VR-embedded widgets to exchange multimedia and interaction data between an external application and the VR application, we can support numerous social and collaborative activities.

(1) **Pointing**. A user outside of VR can touch a point on the VR user display view. The location of the point can be rendered on a view-fixed widget in the VR user's HMD. The shared pointer can help ground discussions and conversations around the VR application.
(2) **Screen sharing**. Similar to using Team Viewer (https://www.teamviewer.com/), two users can initiate a screen sharing session in VR. One application is that a person outside of VR can send graphical instructions into VR to help onboard a first time user. In another application, a VR artist can teach a group of VR users how to paint in VR. They can all share their screens and exchange feedback, re-creating the real world experience of taking an art class.

(3) **Social experiences**. In a VR multiplayer game, a group of friends can share their game screens. Being able to see what everyone else is seeing, they can potentially cooperate by sharing location and strategy with one another [5].
(4) **Creative collaboration**. Multiple VR artists could create a 3D asset together. Although VR-embedded widgets do not allow them to share the same canvas, they still could see each other's work and can coordinate tasks.
(5) **User study**. A researcher can inject multiple-choice questionnaires into a VR application to carry out user studies without having to instrument the application.
(6) **Collaborative UI prototyping**. A VR UI designer can deploy and test new design sketches in existing VR applications. For example, a new menu layout for Tilt Brush could be designed in Photoshop and then anchored on the user hand controller in VR. With this deployment, the designer can also collect feedback from the test user such as where she pointed, her head motion, and her verbal feedback.

CONCLUSION

This paper outlined our ongoing exploration into making VR applications more social. We proposed developing communication add-ons in VR to extend an existing VR application with communication features without modifying it. To illustrate this idea and motivate further research, we discussed an application in tutorial system and illustrated a number of new design ideas.

REFERENCES

[1] 2018. Introducing Studio Share: Sculpt with Friends in Oculus Medium. https://www.oculus.com/blog/introducing-studio-share-sculpt-with-friends-in-oculus-medium/
[2] 2019. OpenVR SDK. https://github.com/ValveSoftware/openvr
[3] Josh Constine. 2017. Oculus Dash replaces your computer monitor with VR. http://tcrn.ch/2g1XZSg
[4] 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 - CHI '17*. ACM Press, New York, New York, USA, 4021–4033. https://doi.org/10.1145/3025453.3025683
[5] Shunichi Kasahara, Mitsuhito Ando, Kiyoshi Suganuma, and Jun Rekimoto. 2016. Parallel Eyes: Exploring Human Capability and Behaviors with Paralleled First Person View Sharing. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), 1561–1572. https://doi.org/10.1145/2858036.2858495
[6] Balasaravanan 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 (CHI '19)*. ACM, New York, NY, USA. https://doi.org/3290605.3300514
[7] 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 - UIST '17*. ACM Press, New York, New York, USA, 267–277. https://doi.org/10.1145/3126594.3126659
[8] Cuong Nguyen, Stephen DiVerdi, Aaron Hertzmann, and Feng Liu. 2018. Depth Conflict Reduction for Stereo VR Video Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York,

NY, USA, Article 64, 9 pages. https://doi.org/10.1145/3173574.3173638

[9] Ralph R Swick and Mark S Ackerman. 1988. The X Toolkit: More Bricks for Building User-Interfaces or Widgets for Hire.. In *Usenix Winter*. Citeseer, 221–228.

[10] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Spacetime: Enabling Fluid Individual and Collaborative Editing in Virtual Reality. In *The 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*. ACM Press, New York, New York, USA, 853–866. https://doi.org/10.1145/3242587.3242597