Reality Stack I/O: A Versatile and Modular Framework for Simplifying and Unifying XR Applications and Research

Anonymous Author(s)

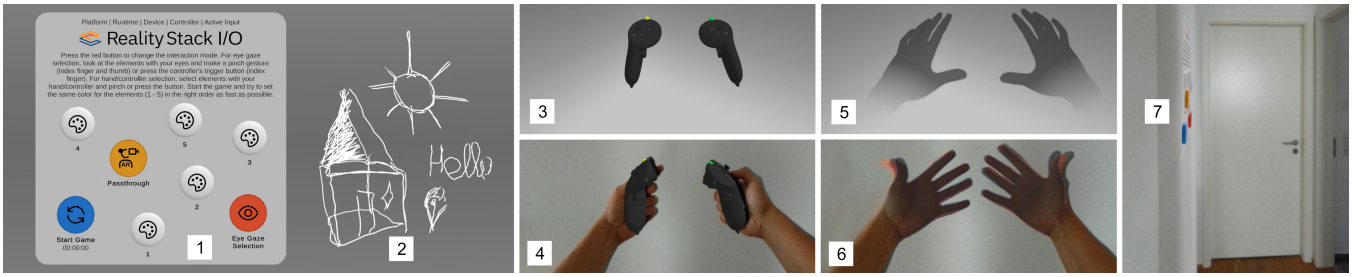

Figure 1: The Reality Stack I/O example includes a 2D user interface (1), a 3D sketching tool (2), and a virtual-to-physical surface alignment technique (3ViSuAl [3]). In this scenario, the user uses a Meta Quest Pro device and interacts with controllers (3, 4), hand tracking (5, 6), and/or eye tracking. By pressing the passthrough UI button (1), the device switches between VR (3, 5) and VST AR (4, 6). The 3D sketching tool allows for simple drawings (2), and 3ViSuAl makes it easy to align virtual surfaces with flat physical surfaces. Here, the user aligned the 2D user interface with a wall, enabling direct touch interaction (7).

ABSTRACT

This paper introduces Reality Stack I/O (RSIO), a versatile and modular framework designed to facilitate the development of extended reality (XR) applications. Researchers and developers in XR often encounter time-consuming challenges due to the variety of XR devices available, leading to delays and increased complexity. RSIO provides the essential features to simplify and unify XR applications across various XR devices. It also enhances cross-device and cross-platform compatibility, expedites integration, and allows developers to focus more on building XR experiences rather than on device integration. We offer a public reference implementation.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; **Mixed / augmented reality**.

KEYWORDS

VR, AR, XR, XR, CR, Framework, Interaction

ACM Reference Format:

Anonymous Author(s). 2023. Reality Stack I/O: A Versatile and Modular Framework for Simplifying and Unifying XR Applications and Research. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology Adjunct (ISMAR '23).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

ISMAR '23, October 16-20 2023, Sydney, Australia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Spatial computing is an important step towards seamlessly integrating physical and virtual environments for extended reality (XR) workspaces. It leverages technologies such as augmented reality (AR), virtual reality (VR), and mixed reality (MR) that enable users to interact with digital and real content in an intuitive, spatial way. In particular, cross-reality (CR) or transitional interfaces (TI) support simultaneous use and transition between different points on the reality-virtuality continuum (RVC) [5, 7]. CR/TI are a promising approach for understanding and exploring spatial data and related information, but still remain underexplored in HCI research [2, 6].

For example, in the field of CR/XR research, Schröder et al. [6] presented analytical lenses for understanding dyadic collaboration in transitional interfaces, Jetter et al. [1] explored VR as a design tool for sketching and simulating spatially-aware interactive spaces, and Kern et al. [4] investigated controller-based virtual tap- and swipe keyboards in VR and video see-through (VST) AR.

While the design of these applications already requires extensive knowledge, another major challenge becomes apparent during development and research: the variety of XR devices.

XR devices offer a wide range of capabilities that are continually expanded. For example, early devices (e.g., Oculus Rift S) only supports head and controller tracking. In comparison, latest devices (e.g., Meta Quest Pro or Pico 4 Enterprise) additionally provide sensors for MR, hand, eye, and face tracking. Therefore, researchers and developers spent a significant amount of time in enabling cross-device and cross-platform compatibility, rather than building XR experiences for visualization, interaction, design, or collaboration.

Frameworks such as Mixed Reality Toolkit (MRTK)[1] or XR Interaction Toolkit (XRI)[2] have emerged as popular solutions for developing applications across realities, devices, and platforms. With a focus on the OpenXR standard, these frameworks provide

[1] https://learn.microsoft.com/windows/mixed-reality/mrtk-unity/
[2] https://docs.unity3d.com/Manual/com.unity.xr.interaction.toolkit/

flexible input systems and basic components for mixed reality interactions and interfaces. However, XR device manufacturers often make their latest features available only through their native XR plugins, rather than directly integrating the OpenXR standard by default. As a result, only a limited number of devices are supported, and incorporating new features can be highly time-consuming or even impossible.

In contrast, frameworks like the Virtual Reality Toolkit (VRTK)[3] and UltimateXR[4] simplify development by using native XR plugins instead of relying solely on the OpenXR standard. While these frameworks provide support for a wide range of consumer XR devices, they focus primarily on interaction features for VR rather than CR/TI. Consequently, they offer limited support for technological advances such as cross/mixed reality, eye and face tracking, or spatial awareness.

With Reality Stack I/O (RSIO), we aim to bridge the gap between feature-rich interaction frameworks and most advanced XR device features. RSIO builds on both the OpenXR standard and, if the required functionality is not yet available, on manufacturers' native XR plugins. Our approach enhances cross-device and cross-platform compatibility, rapid integration, and allows developers to focus on building XR experiences rather than device integration. From a research perspective, RSIO supports replicability of previous experiments and enables direct comparisons between different XR devices. We offer a reference implementation with examples.

2 DESIGN REQUIREMENTS

We designed RSIO to address the following challenges of current XR development and research: For extensibility, RSIO should have a flexible architecture to allow for easy and timely extensions. In terms of broad compatibility, RSIO should work with various XR devices, established interaction frameworks, and custom solutions for social XR, as well as sketching and text input. Additionally, it should be able to function as a standalone framework. For usability, RSIO should be user-friendly and provide examples and documentation.

3 IMPLEMENTATION

For our reference implementation, we use Unity 2021.3 LTS. Unity is a widely used and versatile game engine that provides extensive support for XR development, making it an ideal platform for our purposes. Using a version with long-term support (LTS) ensures stability and consistency. Figure 2 shows the architecture of RSIO. RSIO provides developers with access to lightweight interfaces that abstract the OpenXR standard and, if features are not yet available, utilize manufacturers' native XR plugins. We provide essential features, including head tracking, eye tracking, hand tracking, controller tracking, hand gestures, and input capabilities. Considering the growing interest in and accessibility of mixed reality devices, we also include passthrough support. The system also provides developers with features for aggregated, unified data access. For example, the generic input feature unifies user input from controller buttons, hand gestures, and traditional keyboard and mouse.

Fine-grained recording of both raw and unified data is an essential feature for XR research. Therefore, we offer a flexible data recording interface. The CSV-based recording capability empowers researchers to conduct in-depth analyses of their user studies, and also enables playback of previously performed user sessions. Each feature (e.g., eye or hand tracking) implements the recording interface and is responsible for providing its CSV header and data.

XR devices usually offer similar tracking sources with a head-mounted display (HMD) and two hand controllers. However, controllers can greatly vary in shape and grasping possibilities [3]. Therefore, our system provides tracking anchors that are by default attached to fingertips and controller models. The idea of tracking anchors is based on previous research [3] regarding unified reference points for 2D interactions (in 3D), and handwriting and sketching with arbitrary XR controllers.

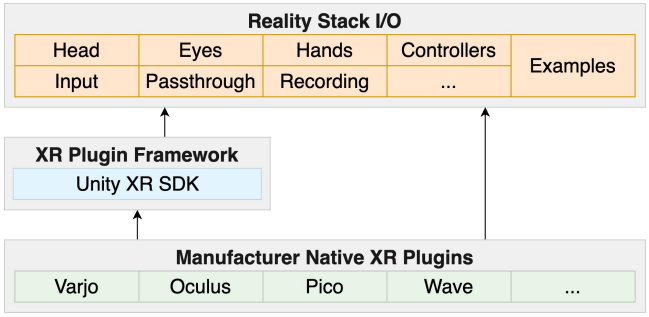

Figure 2: The architecture of Reality Stack I/O. We use Unity XR SDKs and, when features are not yet supported, the manufacturers' native XR plugins (e.g., passthrough capabilities).

4 EXAMPLES

We support researchers and developers by providing various examples of how to use RSIO. In addition to fundamental XR device integration, we also provide examples for spatial interaction and spatial input. These include distance-based UI interaction with controllers, hand gestures, and eye-gaze pointing approaches, as well as direct touch-based solutions (Figure 1). We have integrated RSIO with Ubiq [8], a public social XR platform, as well as the Off-The-Shelf Stylus framework [3] for sketching and handwriting. We also try to continually ensure cross-device and cross-platform compatibility with Microsoft Windows and Android-based XR devices.

5 CONCLUSION

Reality Stack I/O is a versatile and modular solution for XR applications and research. Our approach enhances cross-device and cross-platform compatibility, expedites integration, and allows developers to focus more on building XR experiences rather than on device integration. We offer a public reference implementation accompanied by various examples (https://anonymized/realitystack-io).

REFERENCES

[1] Hans-Christian Jetter, Roman Rädle, Tiare Feuchtner, Christoph Anthes, Judith Friedl, and Clemens Nylandsted Klokmose. 2020. "In VR, everything is possible!": Sketching and Simulating Spatially-Aware Interactive Spaces in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu, HI, USA, 1–16. https://doi.org/10.1145/3313831.3376652

[3] https://www.vrtk.io/
[4] https://www.ultimatexr.io/

[2] Hans-Christian Jetter, Jan-Henrik Schröder, Jan Gugenheimer, Mark Billinghurst, Christoph Anthes, Mohamed Khamis, and Tiare Feuchtner. 2021. Transitional Interfaces in Mixed and Cross-Reality: A new frontier?. In *Interactive Surfaces and Spaces*. ACM, Lodz Poland, 46–49. https://doi.org/10.1145/3447932.3487940

[3] Florian Kern, Peter Kullmann, Elisabeth Ganal, Kristof Korwisi, René Stingl, Florian Niebling, and Marc Erich Latoschik. 2021. Off-The-Shelf Stylus: Using XR Devices for Handwriting and Sketching on Physically Aligned Virtual Surfaces. *Frontiers in Virtual Reality* 2 (June 2021), 684498. https://doi.org/10.3389/frvir.2021.684498

[4] Florian Kern, Florian Niebling, and Marc Erich Latoschik. 2023. Text Input for Non-Stationary XR Workspaces: Investigating Tap and Word-Gesture Keyboards in Virtual and Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (May 2023), 2658–2669. https://doi.org/10.1109/TVCG.2023.3247098 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[5] Paul Milgram and Fumio Kishino. 1994. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information and Systems* 77 (1994), 1321–1329.

[6] Jan-Henrik Schröder, Daniel Schacht, Niklas Peper, Anita Marie Hamurculu, and Hans-Christian Jetter. 2023. Collaborating Across Realities: Analytical Lenses for Understanding Dyadic Collaboration in Transitional Interfaces. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg Germany, 1–16. https://doi.org/10.1145/3544548.3580879

[7] Richard Skarbez, Missie Smith, and Mary C. Whitton. 2021. Revisiting Milgram and Kishino's Reality-Virtuality Continuum. *Frontiers in Virtual Reality* 2 (March 2021), 647997. https://doi.org/10.3389/frvir.2021.647997

[8] Anthony Steed, Lisa Izzouzi, Klara Brandstätter, Sebastian Friston, Ben Congdon, Otto Olkkonen, Daniele Giunchi, Nels Numan, and David Swapp. 2022. Ubiq-exp: A toolkit to build and run remote and distributed mixed reality experiments. *Frontiers in Virtual Reality* 3 (Oct. 2022), 912078. https://doi.org/10.3389/frvir.2022.912078