SIGGRAPH ASIA 東京 2024 TOKYO 3–6 DEC

KHRONOS® GROUP

OpenXR™

OpenXR BOF

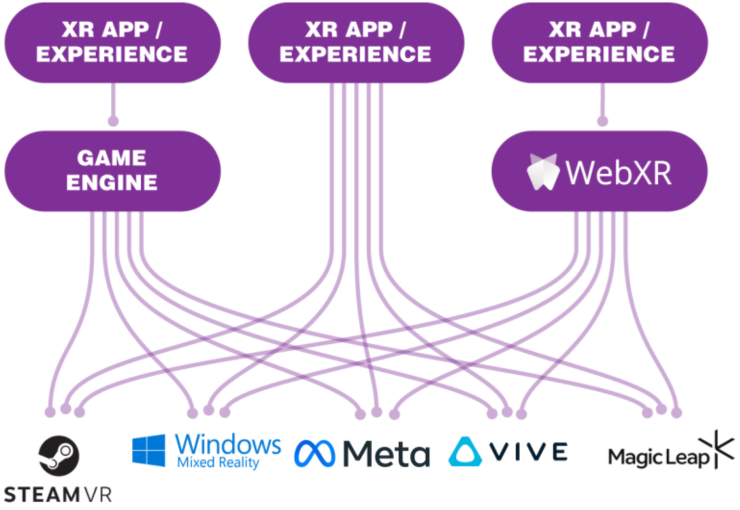
Empowering Cross-Platform Immersive Experiences

Neil Trevett, NVIDIA and Khronos
Jian Zhang, PICO

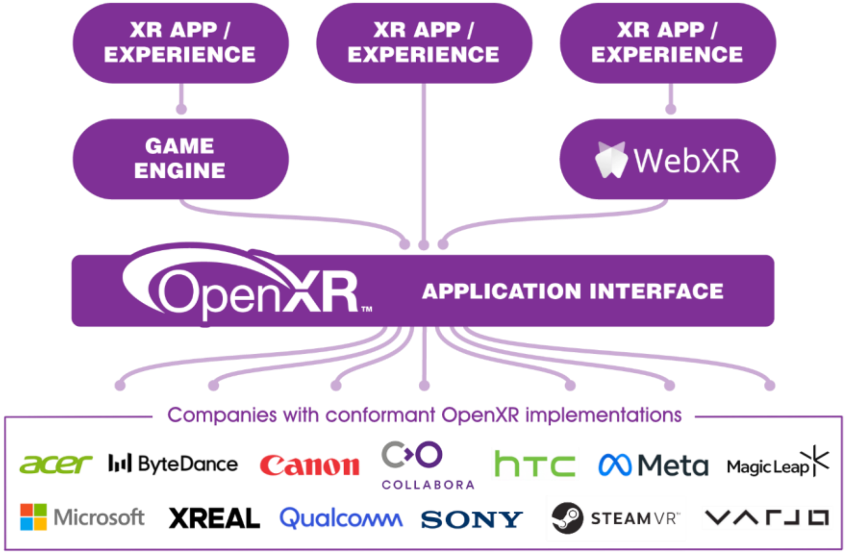
Speakers

Session Title	Speaker	Length
Introduction to OpenXR	Neil Trevett, Khronos	5 minutes
Updates on OpenXR 1.1 and Roadmap	Jian Zhang, PICO	5 minutes
Multi-Application Support & Rendering architectures in XR	Jian Zhang, PICO, Praveen Babu J D, PICO	20 minutes
SecureMR: security and privacy for camera access in XR applications	Jimmy Alamparambil, PICO Jane Tian, PICO	20 minutes
Audience Q&A	All	10 minutes

OpenXR Cross-Platform Portability



Before OpenXR: Applications and engines needed separate proprietary code for each device on the market.



OpenXR provides a single cross-platform, high-performance API between applications and all conformant devices.

Applications and engines can portably access any OpenXR-conformant hardware

Conformant OpenXR Devices

Microsoft	Meta	htc
HoloLens and Mixed Reality Headsets. Hand and eye tracking extensions	Rift S, Quest 3, Quest 2 and Quest Pro Meta Deprecated own API for OpenXR	Vive Focus 3, Vive Cosmos, Vive XR Elite, Vive Wave Runtime
VALVE	VARJO	Canon
Valve Index Valve Deprecated OpenVR APIs for OpenXR	All Varjo Headsets are fully compliant XR-3, XR-4	MREAL X1
MagicLeap	XREAL	Snapdragon spaces
Magic Leap 2	XREAL Air 2, Air 2 Pro, Air 2 Ultra	Qualcomm Snapdragon Spaces XR Development Platform
acer	PICO	SONY
Spatial Labs Display Series	Neo 3, Pico 4, Pico 4 Ultra	Spatial Reality Displays

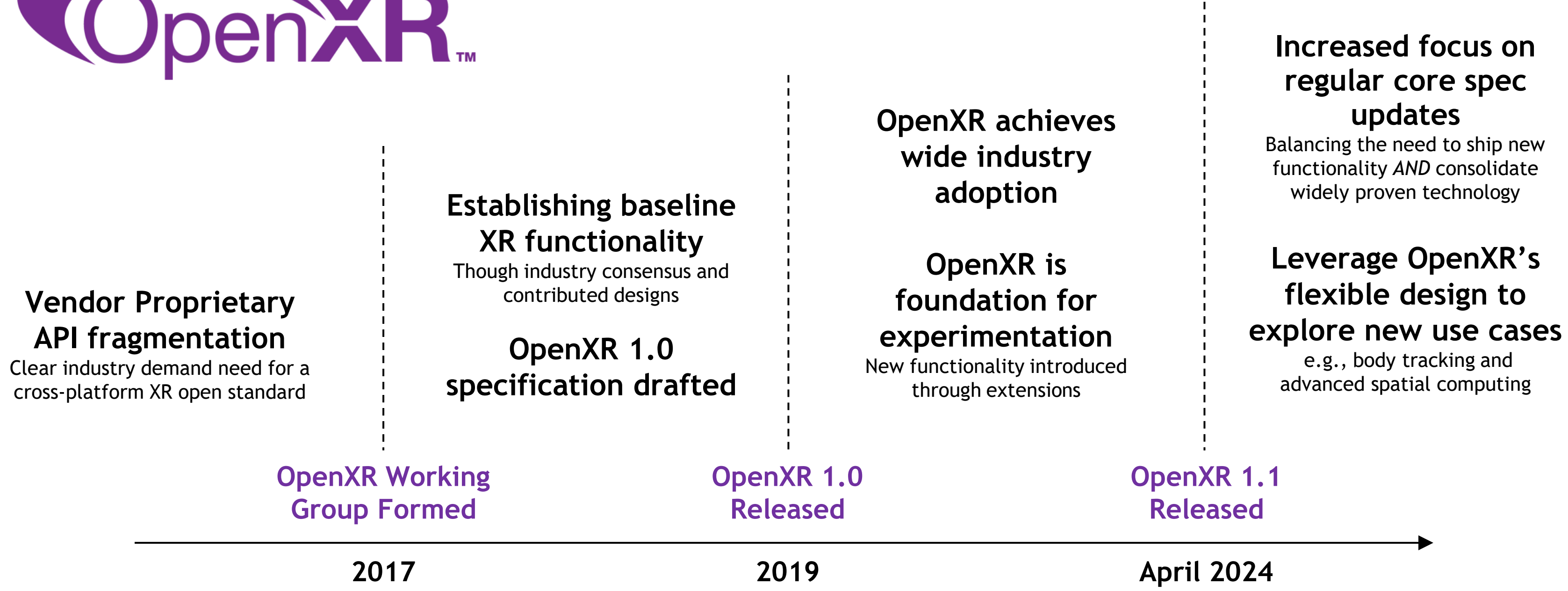
The OpenXR Story So Far…

Empowering Cross-platform Immersive Experiences

OpenXR 1.1

Consolidates multiple extensions to streamline application development and reduce fragmentation

Adds new functionality with spec improvements



Vendor Proprietary API fragmentation

Clear industry demand need for a cross-platform XR open standard

Establishing baseline XR functionality

Though industry consensus and contributed designs

OpenXR 1.0 specification drafted

OpenXR achieves wide industry adoption

OpenXR is foundation for experimentation

New functionality introduced through extensions

Increased focus on regular core spec updates

Balancing the need to ship new functionality *AND* consolidate widely proven technology

Leverage OpenXR's flexible design to explore new use cases

e.g., body tracking and advanced spatial computing

OpenXR Working Group Formed

2017

OpenXR 1.0 Released

2019

OpenXR 1.1 Released

April 2024

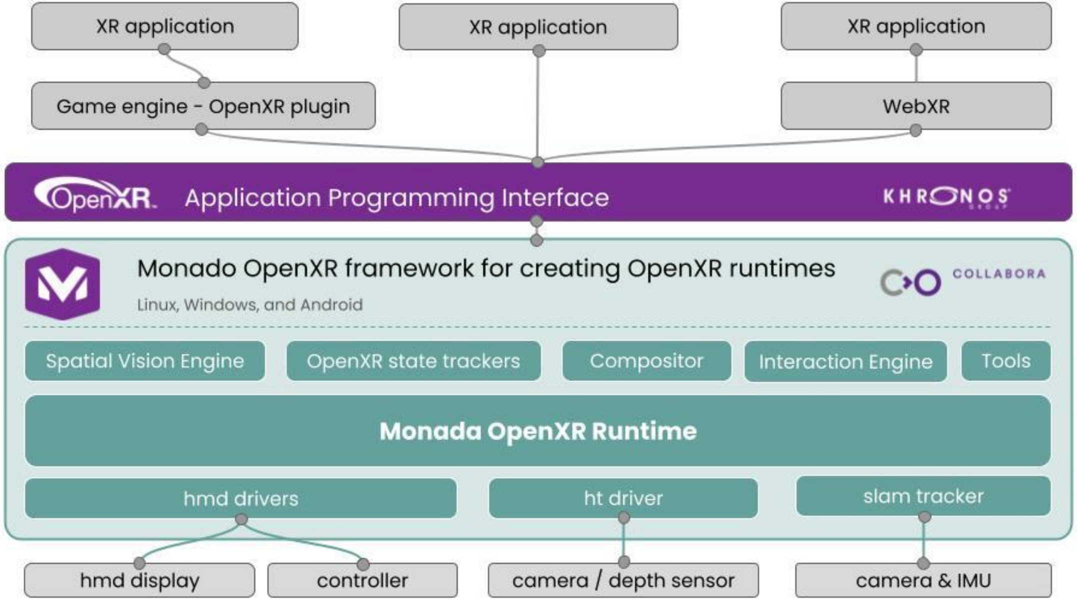
Engines, Browsers, and Libraries with OpenXR

UNREAL ENGINE	Unity	GODOT Game engine
Unreal has been providing support since 4.24. UE 5.0 supports OpenXR	Unity's OpenXR plugin available since 2020 LTS	Godot provides OpenXR support since March 2023 (Core 4.0 Alpha 4)
AUTODESK VRED Library	NVIDIA CLOUDXR™ NVIDIA OMNIVERSE™	
OpenXR supported since VRED 2023.4	NVIDIA Omniverse and CloudXR Platforms	WebXR in Chrome, Edge, and Firefox uses OpenXR as the default backend
COLLABORA MONADO	Meta	stereokit
Open-source OpenXR Implementation	A lightweight XR Meta XR Simulator to Speed Unity OpenXR Development	Open-source mixed reality library for building HoloLens and VR applications

Monado from Collabora

OPEN-SOURCE

- Open source OpenXR Runtime and Framework
- Modular Framework to simplify XR runtime development



OpenXR: Transforming the Future of Cross-Platform Augmented and Virtual Reality

Jian Zhang

Head of XR Foundation Engineering, PICO

OpenXR 1.1 Key Extensions Promoted to Core

- **Local Floor Reference Space**
  - Gravity-aligned world-locked origin for standing-scale content
  - Estimated floor height built in
  - Recenter to current user position at the press of a button without a calibration procedure
- **Grip Surface**
  - Reliable anchors visual content relative to the user's physical hand
  - Can be tracked directly or inferred from a physical controller’s position and orientation
- **Stereo with Foveated Rendering for XR headsets**
  - Runtimes MAY optionally expose eye-tracked or fixed foveated rendering
  - Portable across multiple graphics rendering APIs
  - Applications renders quad views (two high-res insets)
- **Additional enhancements**
  - Interaction Profile improvements
  - Spec language cleanup and clarifications
  - 13 new interaction profiles added to the core spec

OpenXR Releases in 2024

1.1.38 (Jun.2024)	1.1.40 (Aug.2024)	1.1.41 (Sep 2024)	1.1.43 (Nov 2024)
XR_EXT_composition_layer_inverted_alpha Maintenance updates	XR_KHR_metal_enable Maintenance updates	XR_HTC_body_tracking: XR_ML_spatial_anchors XR_ML_spatial_anchors_storage XR_ML_system_notifications XR_ML_world_mesh_detection XR_ML_view_configuration_depth_range_change Maintenance updates	XR_ML_facial_expression XR_META_passthrough_layer_resumed_event XR_META_colocation_discovery XR_META_spatial_entity_sharing XR_META_spatial_entity_group_sharing Maintenance updates

Coming Soon…

- **Extending hand tracking**
  - To include full body tracking
- **Expanded haptics support**
  - Support immersive experiences through PCM, vibrotractiles, and transients
- **Controller render models (glTF)**
  - Showing and animating a model of the user’s actual controller in a future-proof way

OpenXR and Spatial Entities

- **Enhanced handling of spatial entities for advanced spatial computing applications**
  - Standardized methods to interact with the user’s environment
- **Multiple spatial entity types**
  - Planes
  - Objects
  - World Meshes
  - Spatial Anchors
  - Marker Tracking (ArUco, AprilTag, QR code)
- **With BROAD development support from all the major players**
  - Expecting wide portability

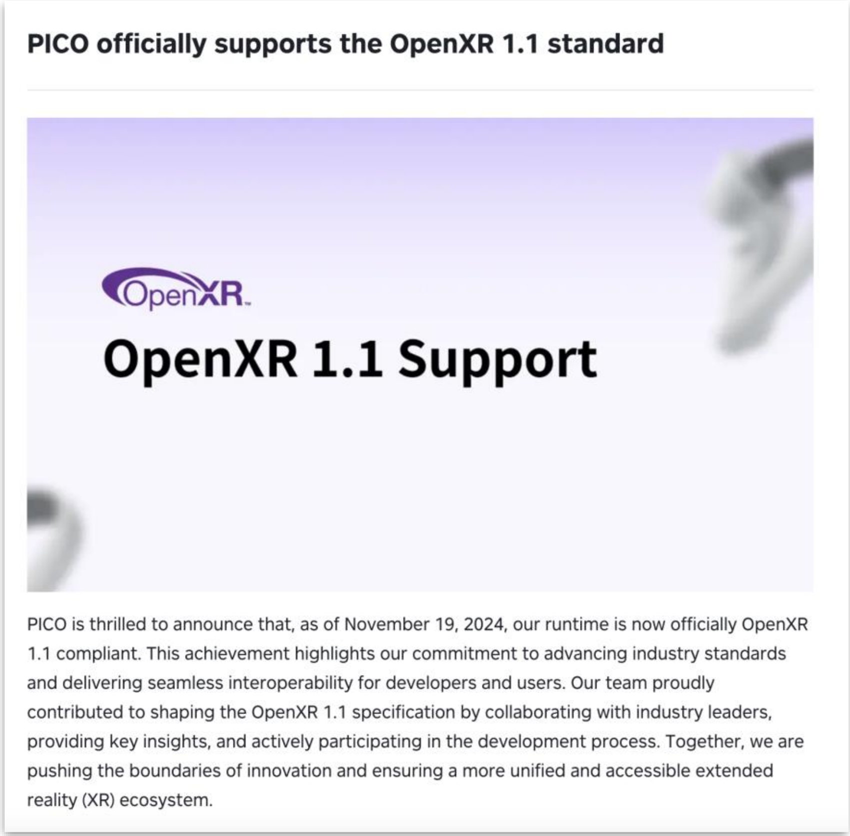
Toward the Next-Gen Open Standard of Spatial Computing

Enabling Multi-Application Support with new rendering architecture in XR

Jian Zhang
Head of XR Foundation Engineering, PICO

Praveen Babu J D
Tech Lead, PICO

PICO 4 Ultra + OpenXR 1.1



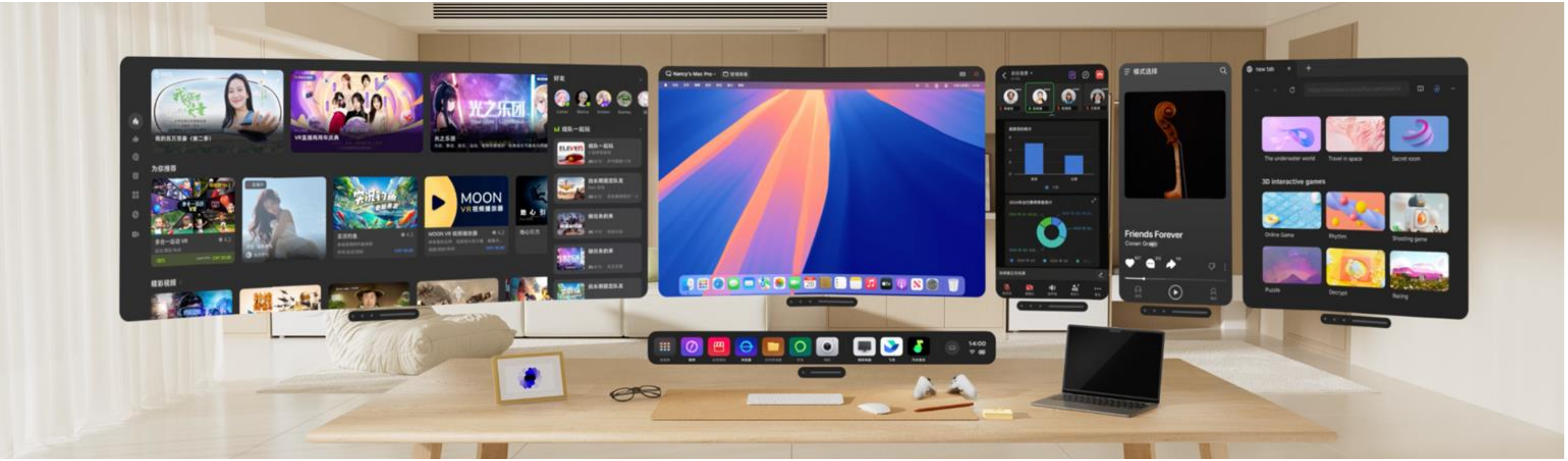
PICO officially supports the OpenXR 1.1 standard

PICO is thrilled to announce that, as of November 19, 2024, our runtime is now officially OpenXR 1.1 compliant. This achievement highlights our commitment to advancing industry standards and delivering seamless interoperability for developers and users. Our team proudly contributed to shaping the OpenXR 1.1 specification by collaborating with industry leaders, providing key insights, and actively participating in the development process. Together, we are pushing the boundaries of innovation and ensuring a more unified and accessible extended reality (XR) ecosystem.

Current Status - Immersive App

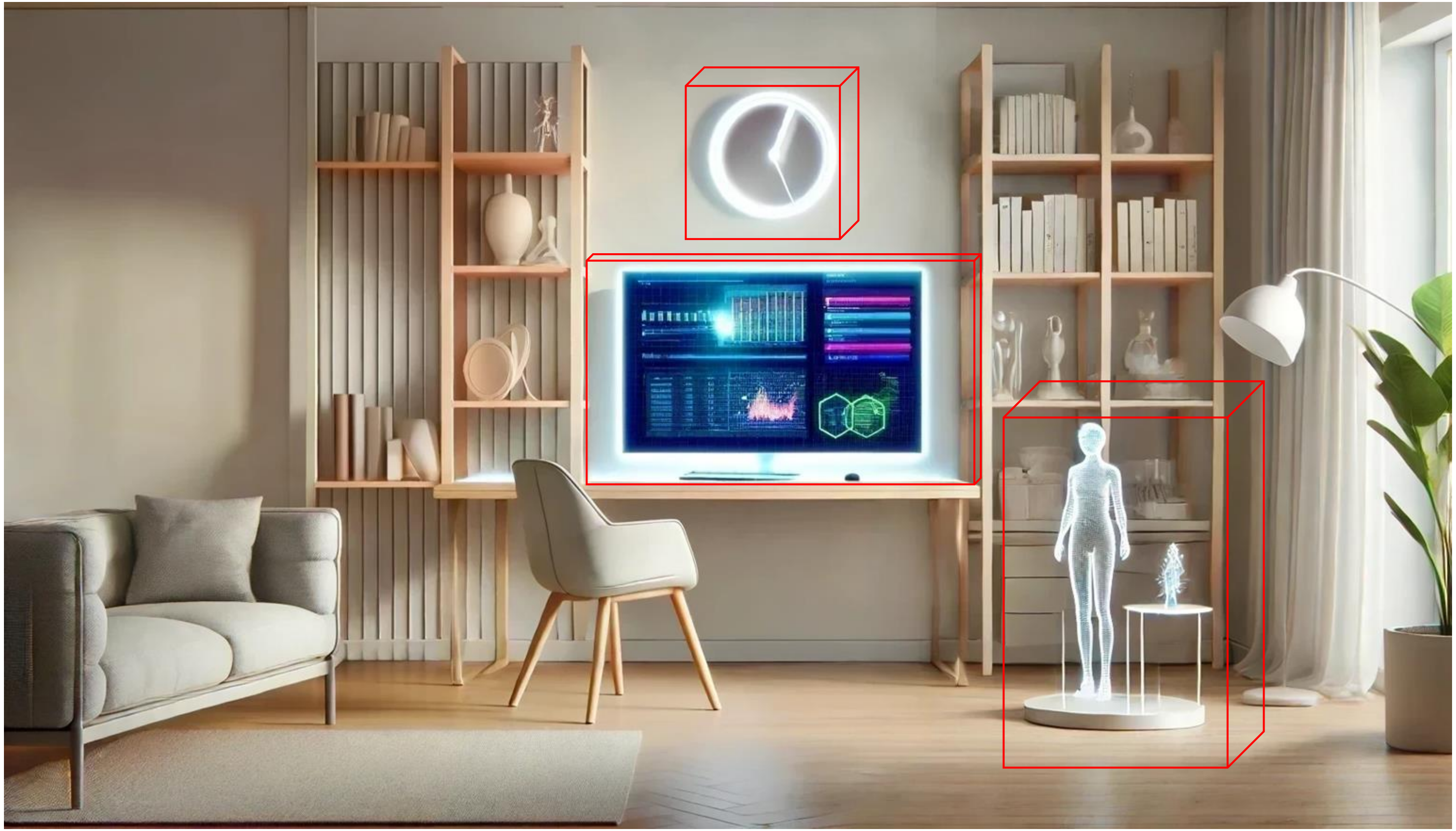
Courtesy to Vertical Robot: Red Matter 2

Current Status - Multi 2D Application



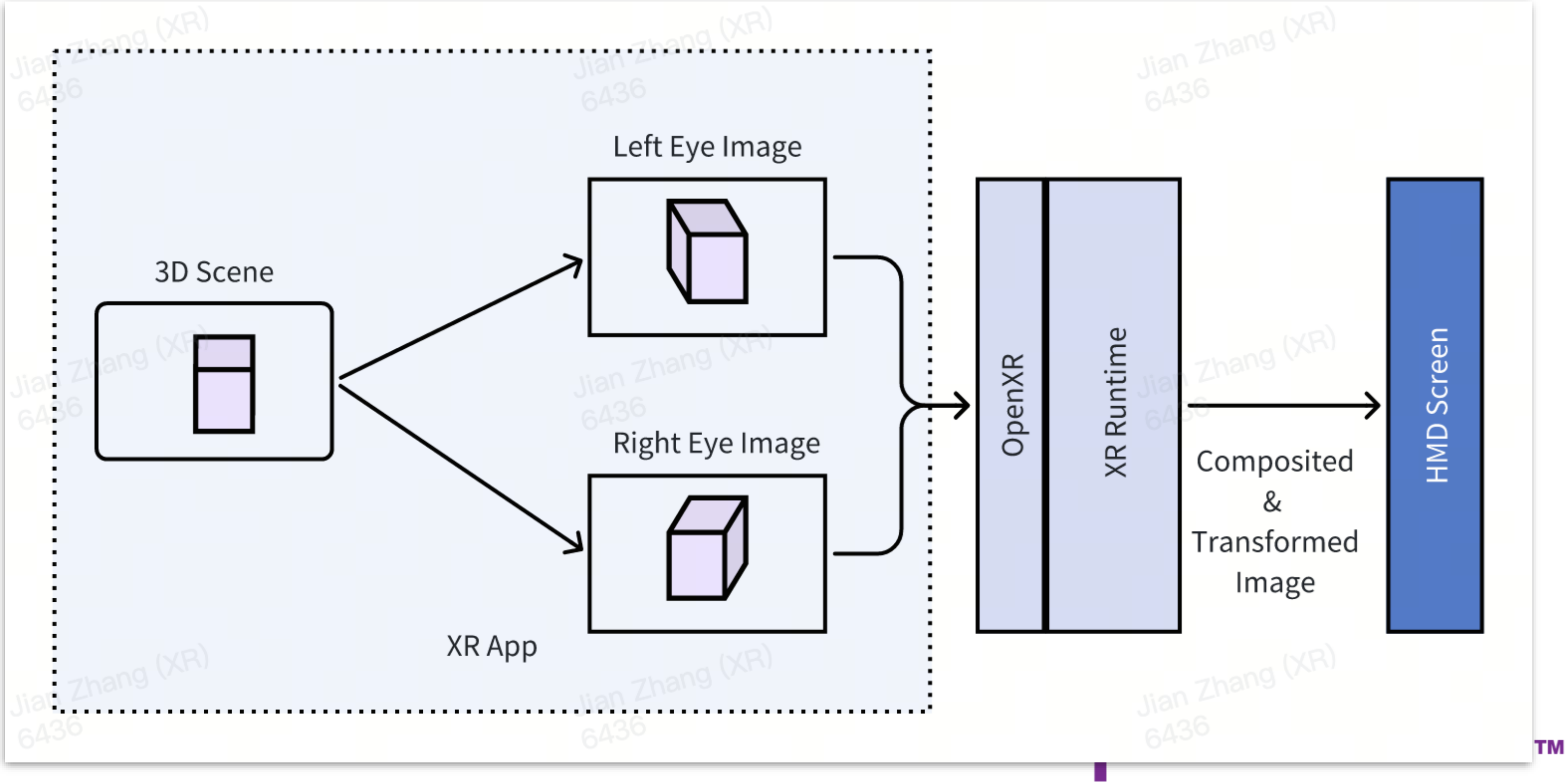
PICO OS - Multiple 2D Application supporting

Next Step



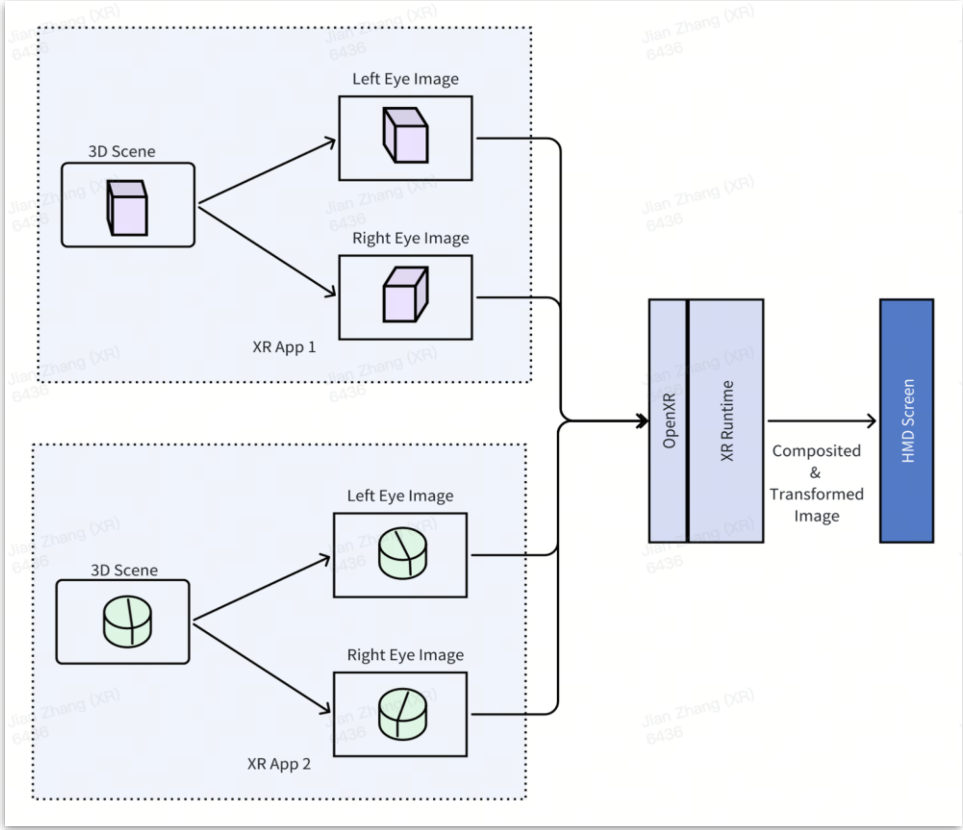
Different form of XR Apps running simultaneously

Single Application: Self-Rendering



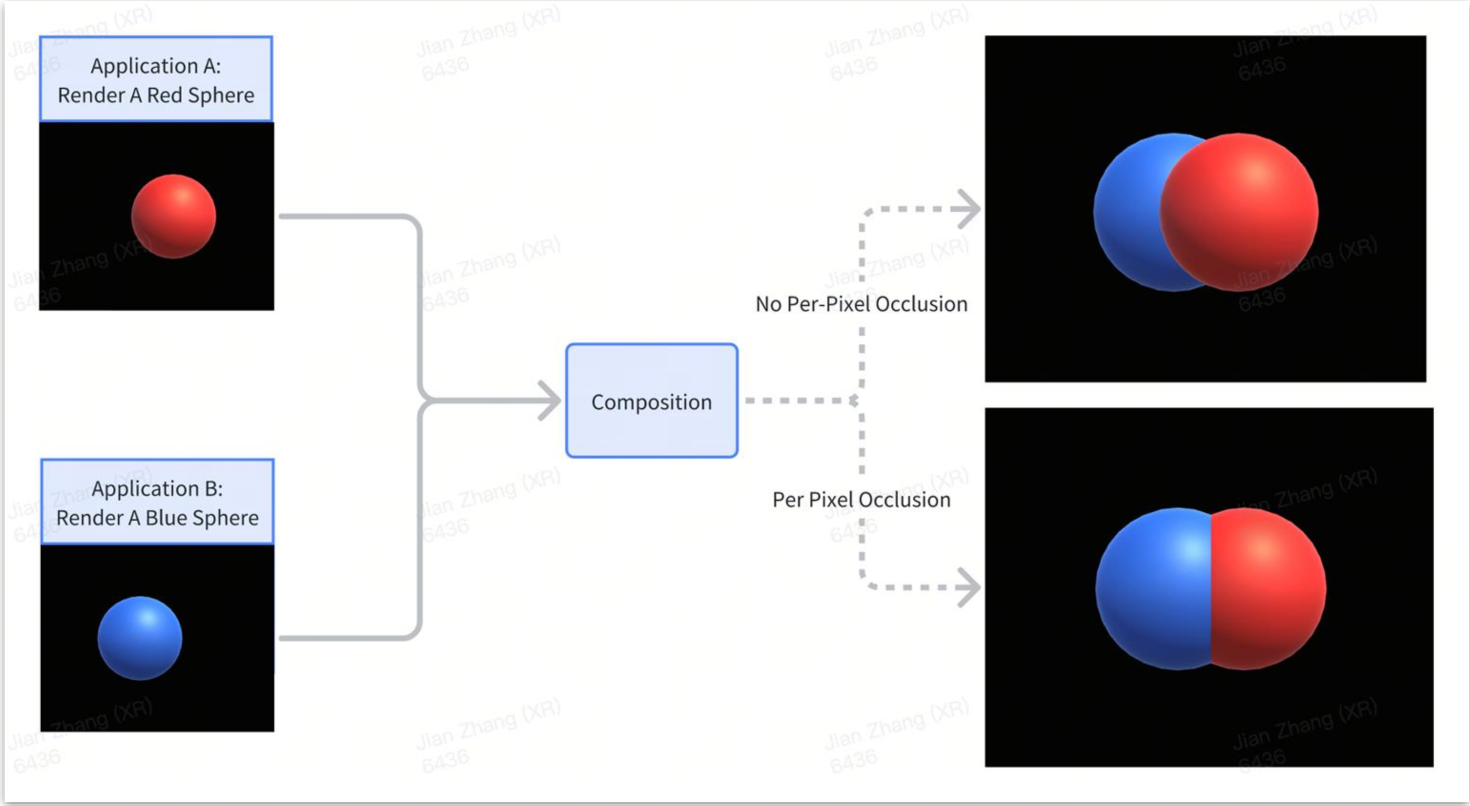
From Stereoscopic rendering to human 3D Perception

Multi Application: Self-Rendering



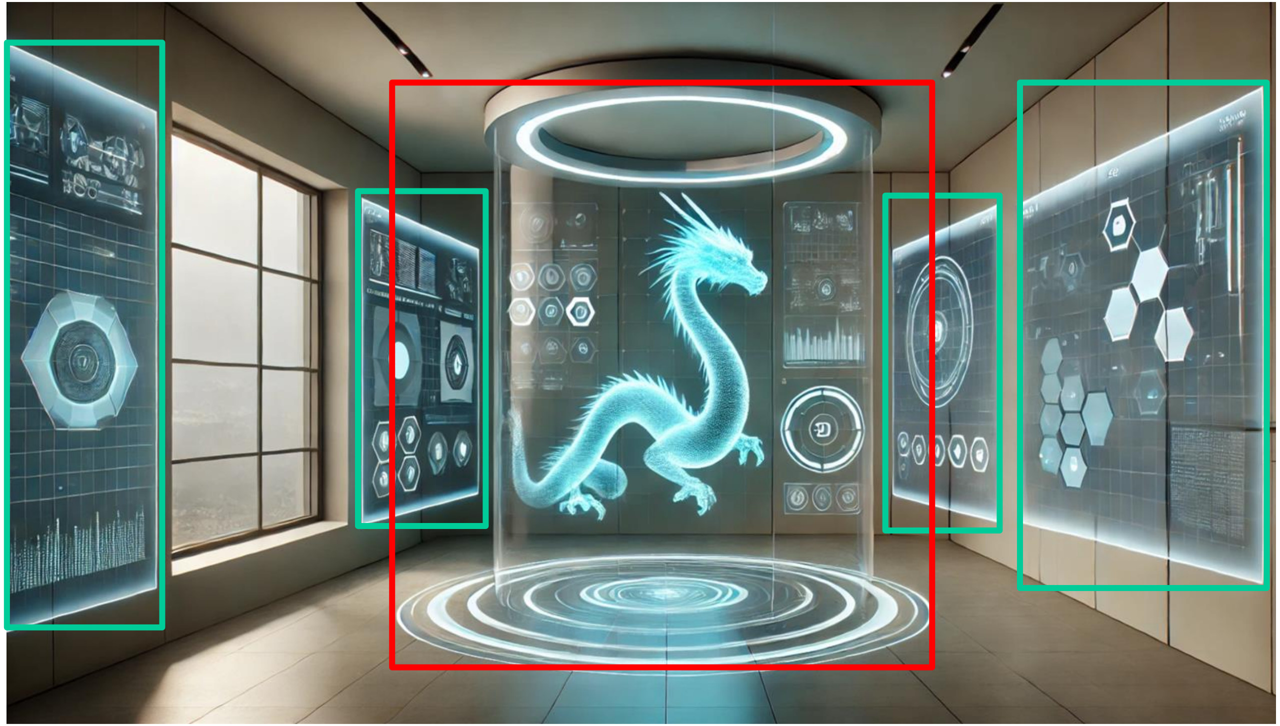
Self-Rendering + Multi-App

Challenges-Shared Space 3D Composition



Shared space occlusions | Composition freedom

Challenges-Resolution Management



Different XR App has different resolution

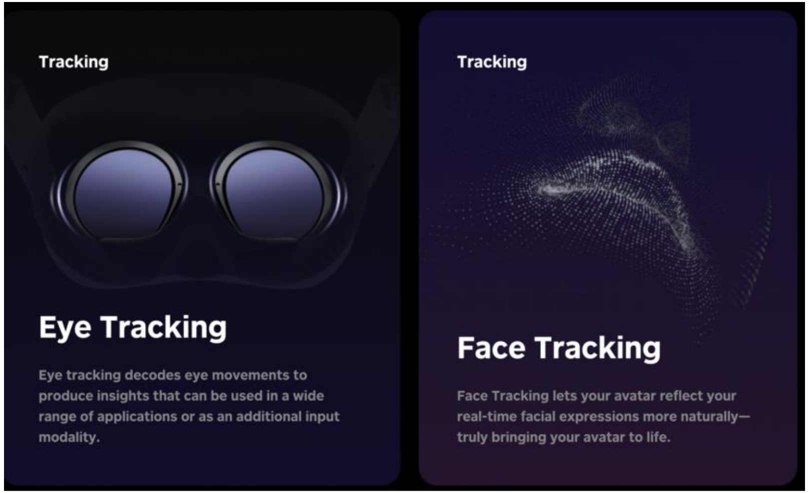
Challenges-Scalability

Many many XR App can co-existing

4k * 4k * 2 (eyes) * 3 (triple buffer swapchain) * 4 (pixel byte depth) = 384M

Challenges-Privacy

Foveated Rendering



Tracking

Eye Tracking

Eye tracking decodes eye movements to produce insights that can be used in a wide range of applications or as an additional input modality.

Tracking

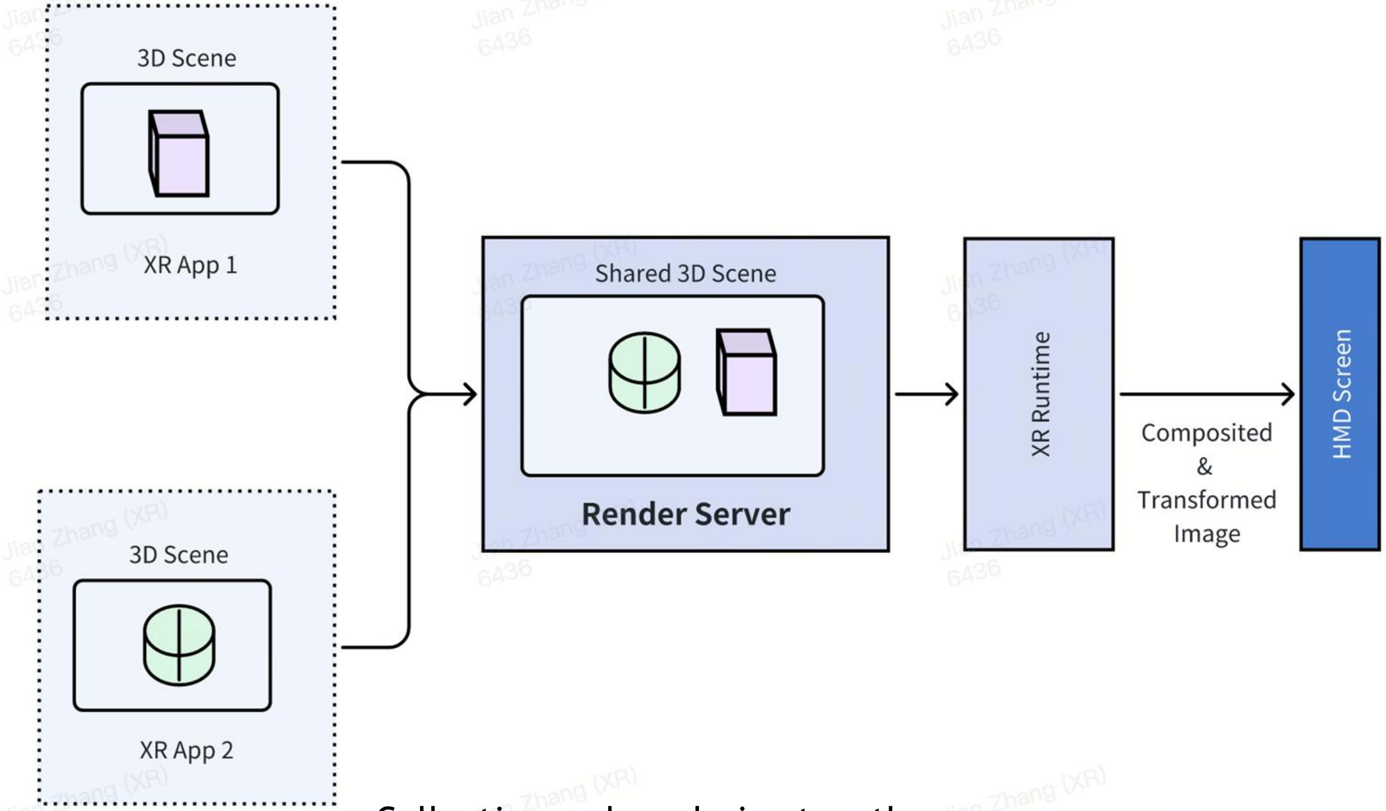
Face Tracking

Face Tracking lets your avatar reflect your real-time facial expressions more naturally—truly bringing your avatar to life.

Eye/Face Tracking

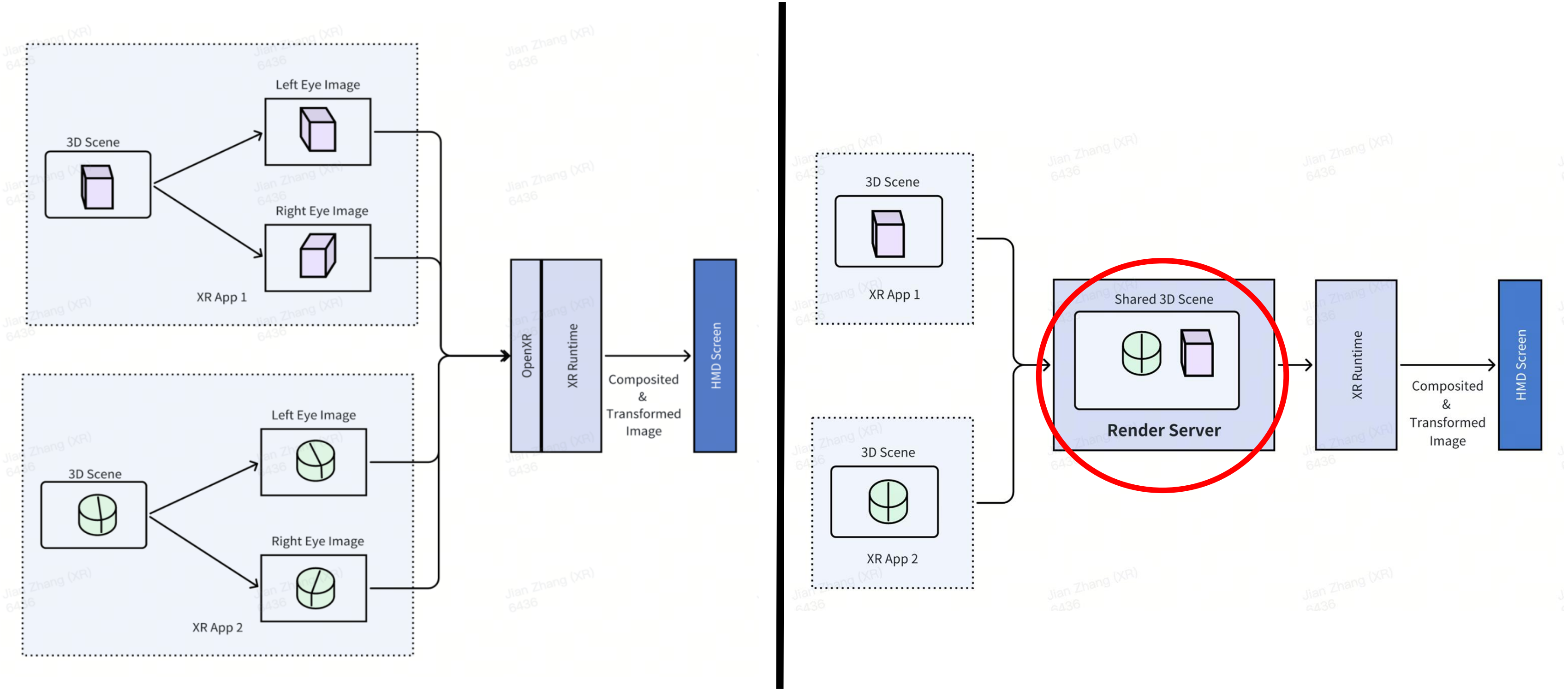
Application need sensitive data for functionalities in Self-Rendering Model

Alternative Solution: Unified Rendering



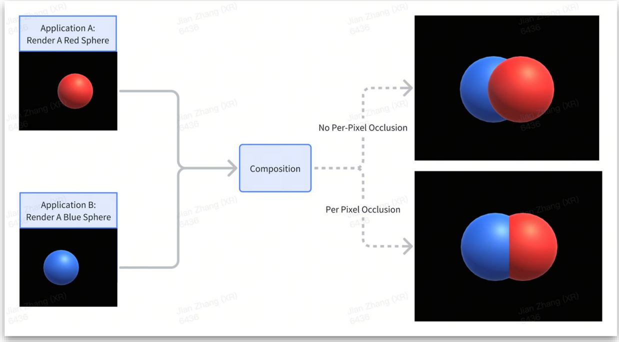
Collecting and rendering together

Self Rendering vs Unified Rendering



Role of Render Server | Not a new thing

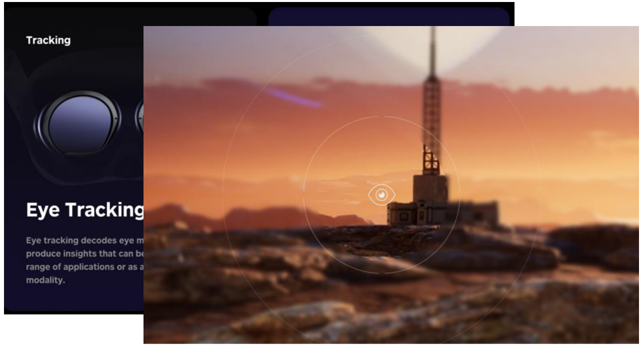
Addressing issues



High freedom Shared Space 3D High Freedom

Automatic Resolution Management

Low overhead -High Scalability



Tracking

Eye Tracking

Eye tracking decodes eye m
produce insights that can be
range of applications or as a
modality.

Privacy Protected

Comparison

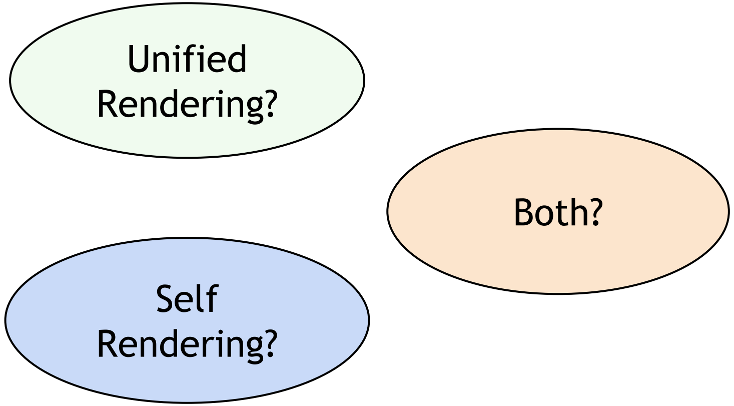
	Unified Rendering	Self Rendering
Privacy	Keeps the privacy data in the system instead exposing them into the application For example: **Avatar** : Hyper realistic avatar will become mainstream in XR, they are highly personal **EyeTracking:** Eye tracking data can potentially reveal a lot of sensitive information about the person	Requires sharing sensitive data with applications to ensure the functionality of features. For example, Foveated Rendering requires information on people's gaze movement Malicious apps can store and upload that data to the internet, and it is extremely difficult to detect this
Low overhead / Scalability	Render buffer and render pass are shared for all applications	The app requires its dedicated render buffer and render pass, even though the app only needs to render very simple things like a single quad. This will bring a constant overhead to each app
Shared space 3D composition	All the applications will be rendered in the same space, composition is natural and consistent	Each application will be rendered on their own. The system can only composite them as 2D images
Shared space features	Easier to enforce shared lighting, physics, etc.	Difficult to achieve

What's the catch?

	Unified Rendering	Self Rendering
Developer Tool Ecosystem	Requires new or updated tools	Already have an established toolset
Developer freedom	Developers use system-provided APIs; less developer flexibility	Full freedom to create custom rendering techniques

Open Standard

- **No standard for Unified Rendering YET**
  - Fragmentations
- **Start the discussion early!**



Unified Rendering?

Both?

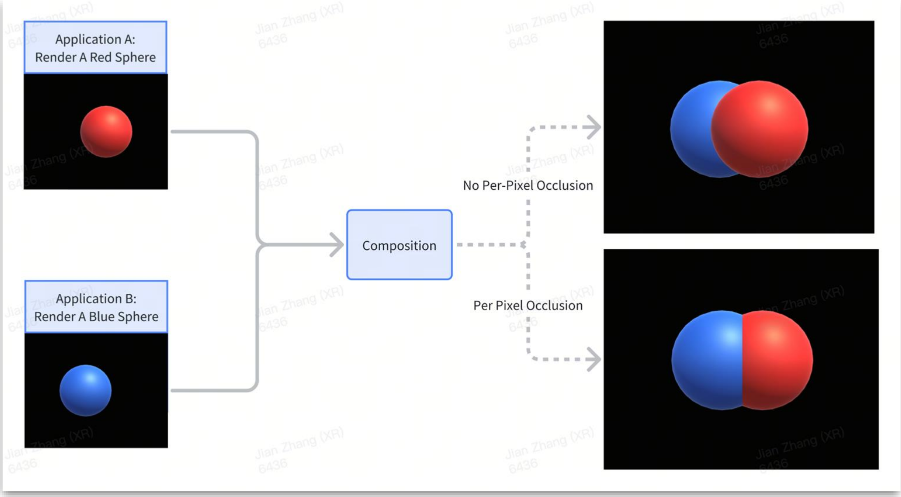
Self Rendering?

Challenges & Next Steps

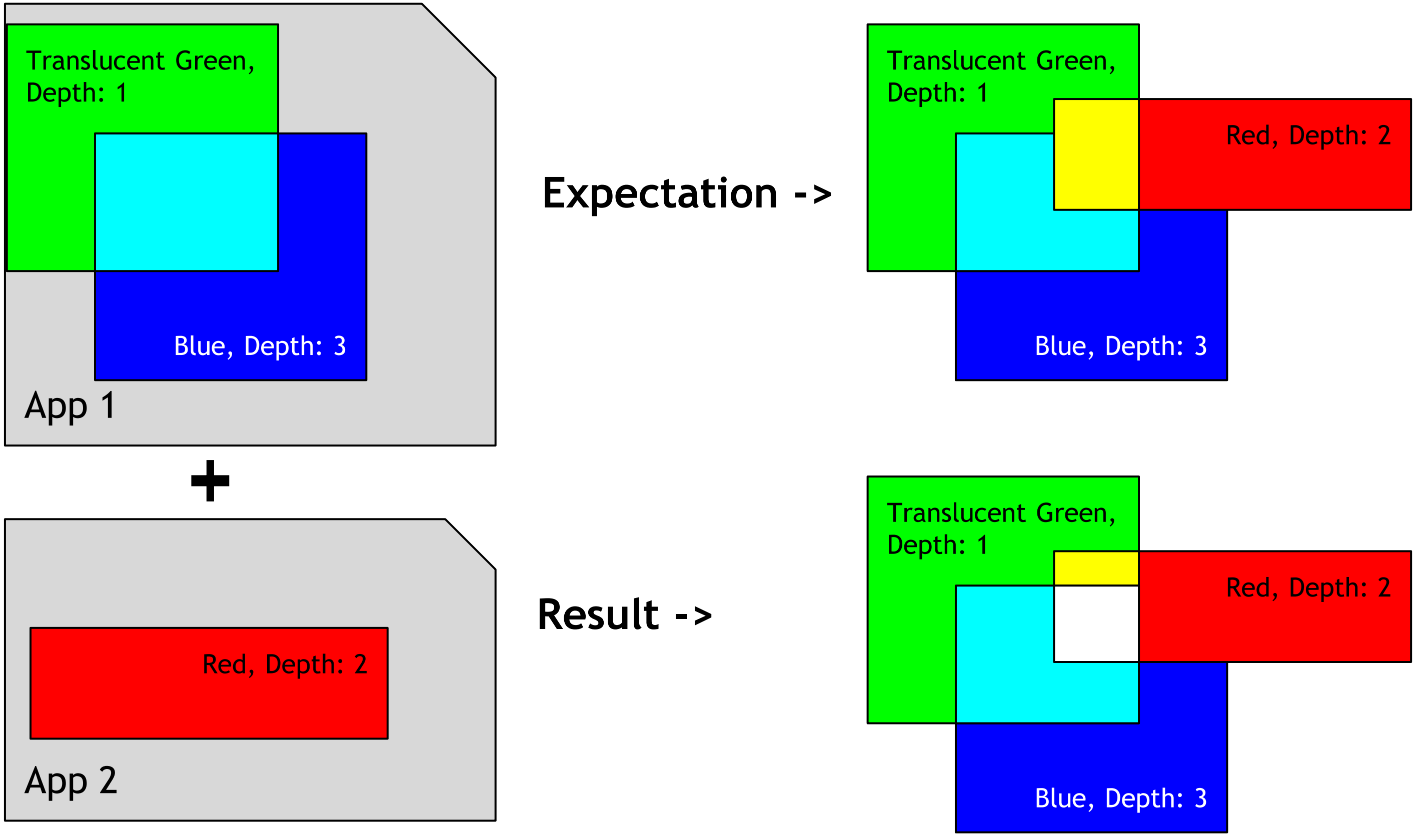
- **Technical challenges**
  - Self Rendering: Apps render using their preferred technology
  - Unified Rendering: Apps render using a centralized Renderer
  - Hybrid: Support all pathways options
- **Standardization challenges**
  - Developer Adoption
  - Vendor Alignment

Self rendering challenges

- **Compute & Optimization Issues**
- **Missing Information**
  - 2x2D surfaces as a baseline
  - Per-pixel occlusion
  - Per-pixel depth information is: Optional
  - Are we losing anything functional ?
  - Multiple fragments per pixel e.g.,: Translucent fragments
  - Warping: re-project an existing 2D image with missing pixels
- **Optimization vs Standardization**


Application A: Render A Red Sphere
Application B: Render A Blue Sphere
Composition
No Per-Pixel Occlusion
Per Pixel Occlusion

Self rendering: Even with depth buffer we will hit limitations



Translucent Green, Depth: 1

Blue, Depth: 3

App 1

+

Red, Depth: 2

App 2

Expectation ->

Translucent Green, Depth: 1

Red, Depth: 2

Blue, Depth: 3

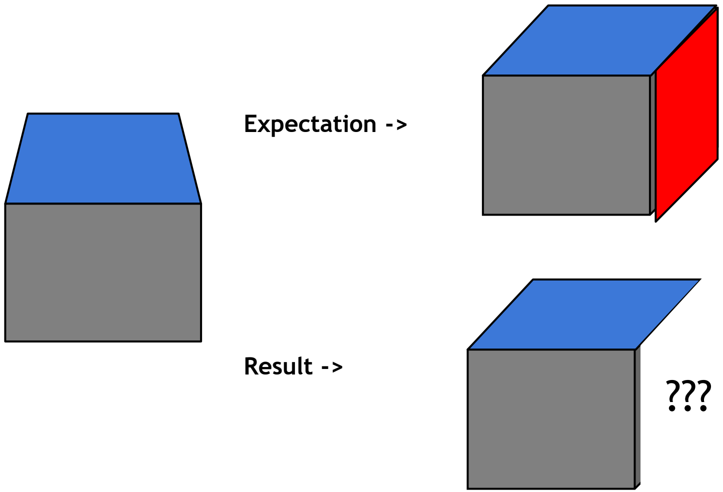
Result ->

Translucent Green, Depth: 1

Red, Depth: 2

Blue, Depth: 3

Reprojection: Missing information reduces accuracy reprojection

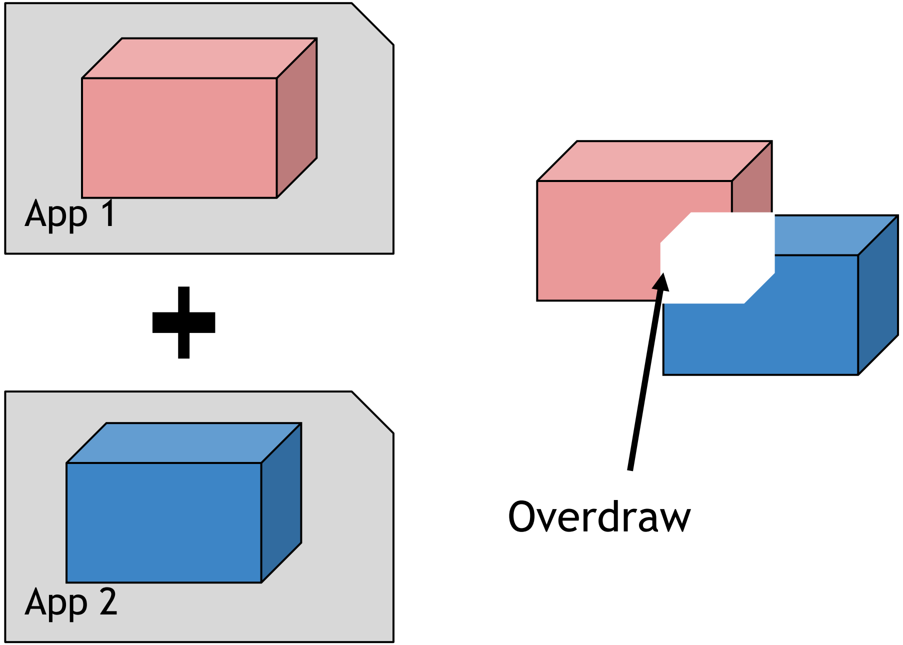


Complications:

Late-Latching is trickier due to composition needing to sample from various apps running at different cadences.

No standard way to provide additional meta-data.

Self rendering: Optimization issues, hard to prevent overdraw without sharing info

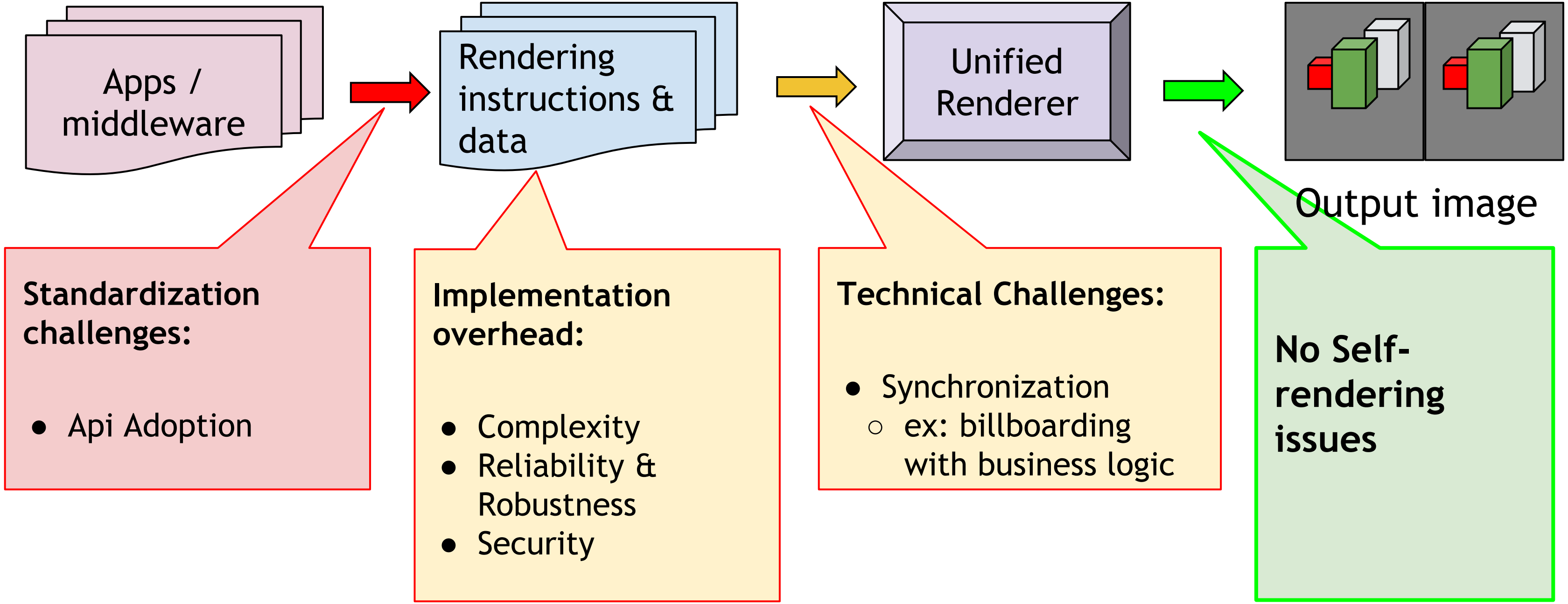


Complications:

How to prevent overdraw without each application not knowing about the other application ?

Unified Rendering Challenges

- Solves several challenges of Self Rendering, However:



Apps / middleware

Rendering instructions & data

Unified Renderer

Output image

Standardization challenges:

- Api Adoption

Implementation overhead:

- Complexity
- Reliability & Robustness
- Security

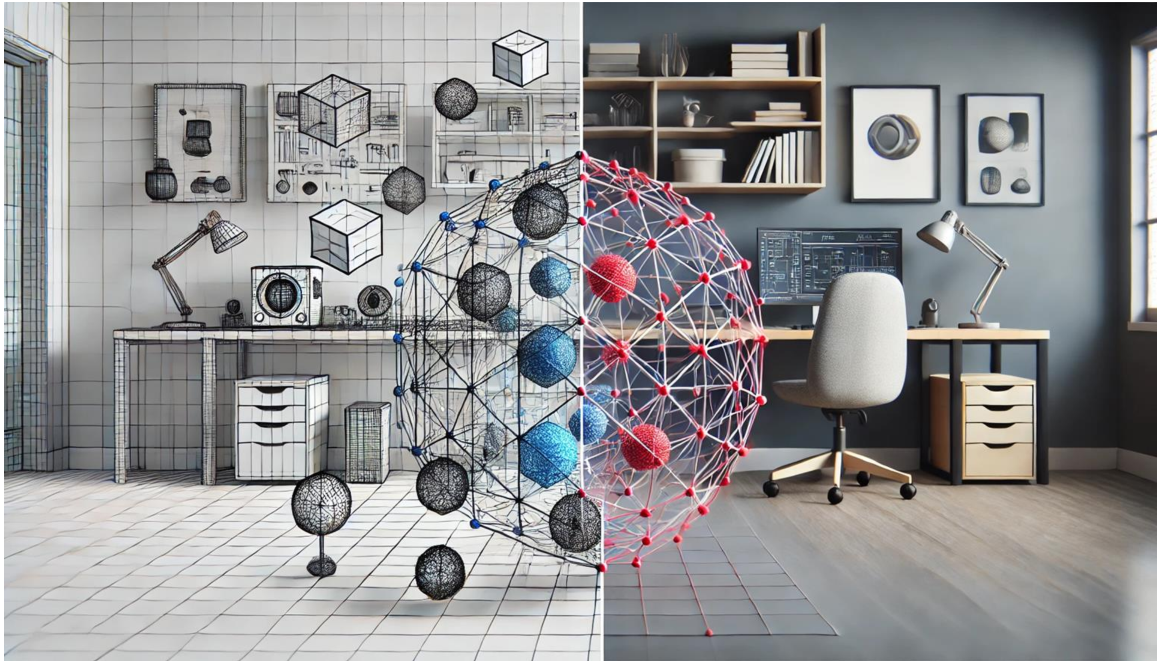
Technical Challenges:

- Synchronization
  - ex: billboarding with business logic

No Self-rendering issues

Hybrid: Support both

- **Best of both worlds**
- **Challenges of both**
- **Additional Challenges**
  - Higher complexity
  - Fragmentation
  - Optimization
  - Quality / Tuning
  - Anti-pattern ?



Standardization challenges

- Diverse Ecosystems
- Vendor Adoption
- Rapid Evolution
- Economic Factors
- Alignment on existing formats
  - Ex: GLTF vs USDZ Vs Other?
  - Ex: Image, Particles etc..
- Backward Compatibility
- Security and Requirement variations
- Vehicle for Standardization
  - OpenXR

Standardization

- **Why Standards are even more important for XR Multi-app**
  - Multi-App support implementations potentially require **intrusive** modifications to the current app development flow
  - **UX**/User interaction could easily **deviate** between platforms which will affect standardization and user expectations
  - Multi-app style of development could become the **defacto** mode of app development for XR devices and we need to avoid fragmentation
  - Make it easy to **transit** to the new **paradigm**

Conclusion

- **The good news**
  - We have solutions for technical challenges
- **Our biggest challenge**
  - Migration for existing applications
  - Standardization
- **Potential solution - OpenXR!**
  - Pico is making these proposals to Khronos
  - If adopted will be available to all OpenXR platforms and applications

SecureMR: Custom features for your app that conserves privacy

Jimmy Alamparambil
Tech Lead, PICO

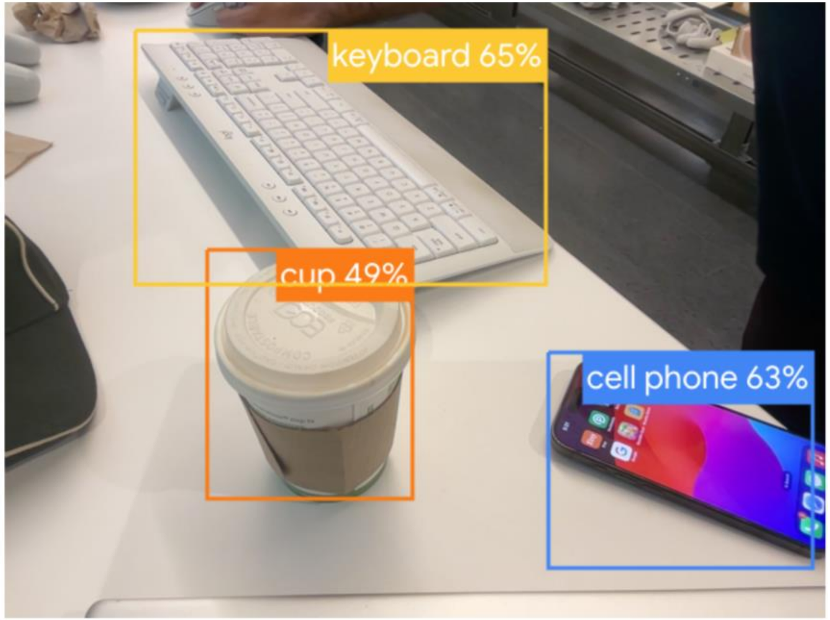
Jane Tian
Product Manager, PICO

Privacy First for XR hardware

- Current climate of ubiquitous surveillance
- People are protective about their private information
- XR headsets do not provide camera images directly to developers

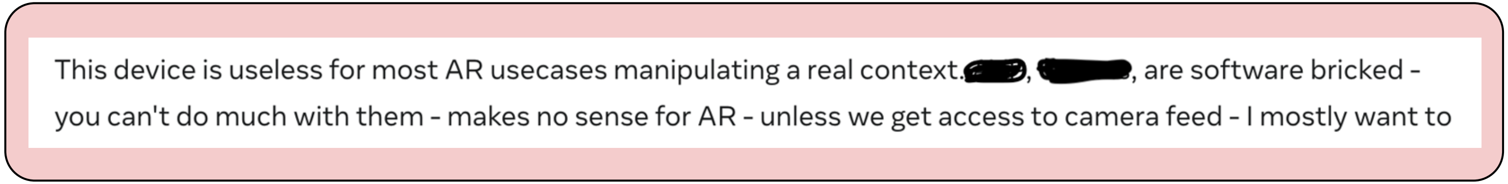
Feature explosion

- Rise of ML and AI has brought into focus some interesting MR features using scene understanding and perception
- These features use ML models and require images of your surroundings as input, and the more real-time they are, the better

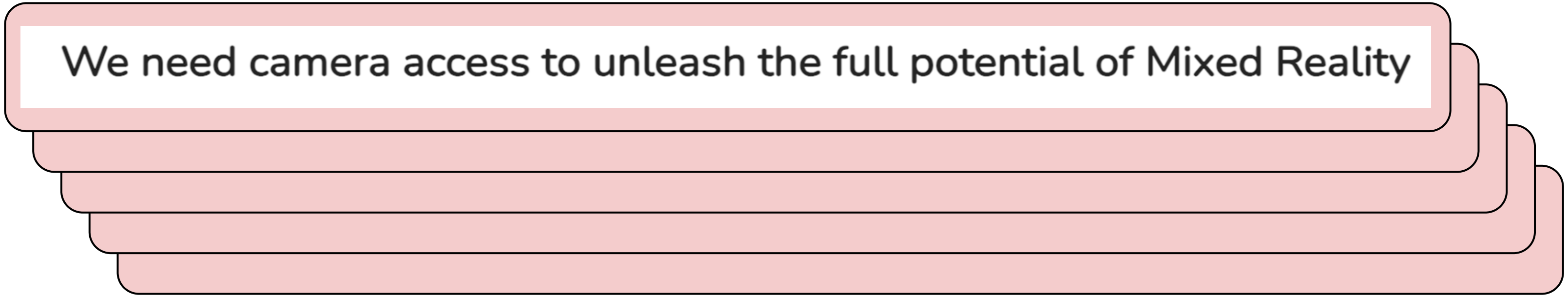


Demand for custom features

- Developers are clamoring for access to camera images so that they can provide some of these interesting features
- So we have a dilemma on our hands - how do we solve this?



This device is useless for most AR usecases manipulating a real context.[redacted], [redacted], are software bricked - you can't do much with them - makes no sense for AR - unless we get access to camera feed - I mostly want to



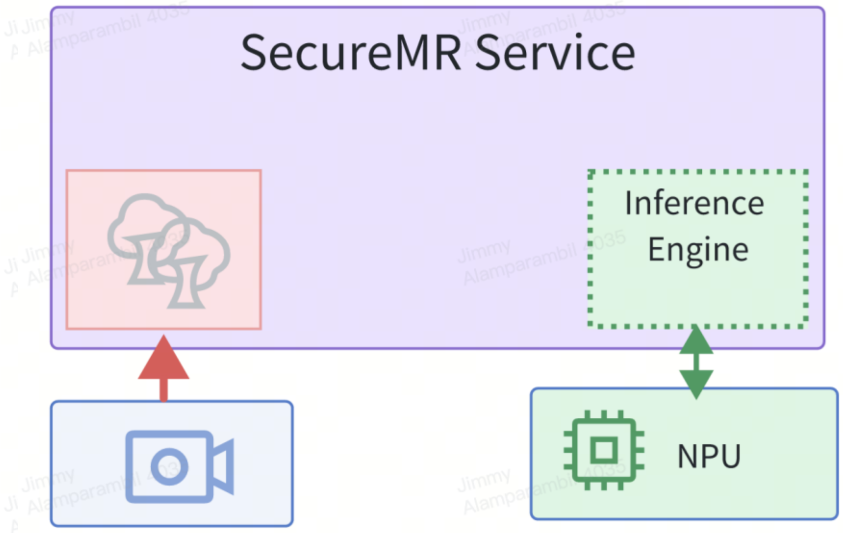
We need camera access to unleash the full potential of Mixed Reality

Our solution

- How do we allow feature creation while conserving privacy?
- We want to enable XR app developers to create custom features without access to camera images directly
- This was how SecureMR was born…

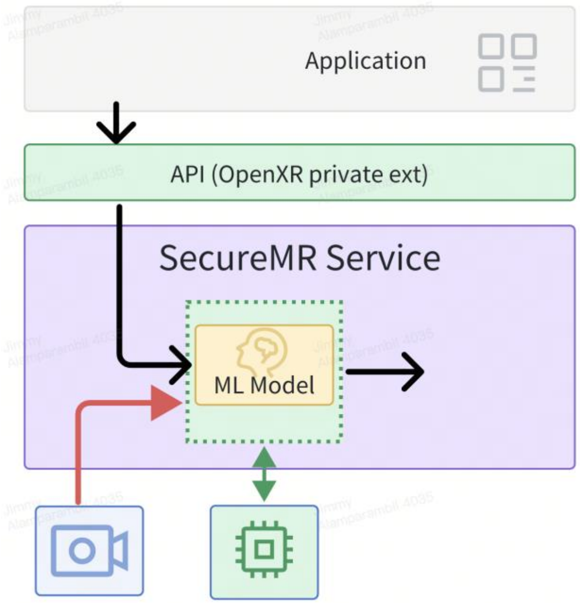
SecureMR

- The framework runs in a secure, privileged process environment as part of the OS
- It has access to the camera image data provided by the XR headset
- It also contains a hardware accelerated inference engine that can run ML models performantly



API

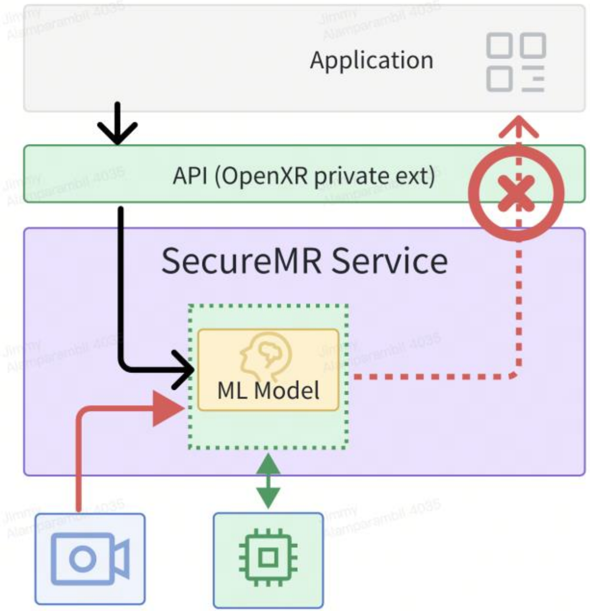
- Your application can use the API to pass down the ML model that implements your feature, as well as other inputs to the model
- The ML model runs in the Inference Engine and produces some outputs



One way data flow

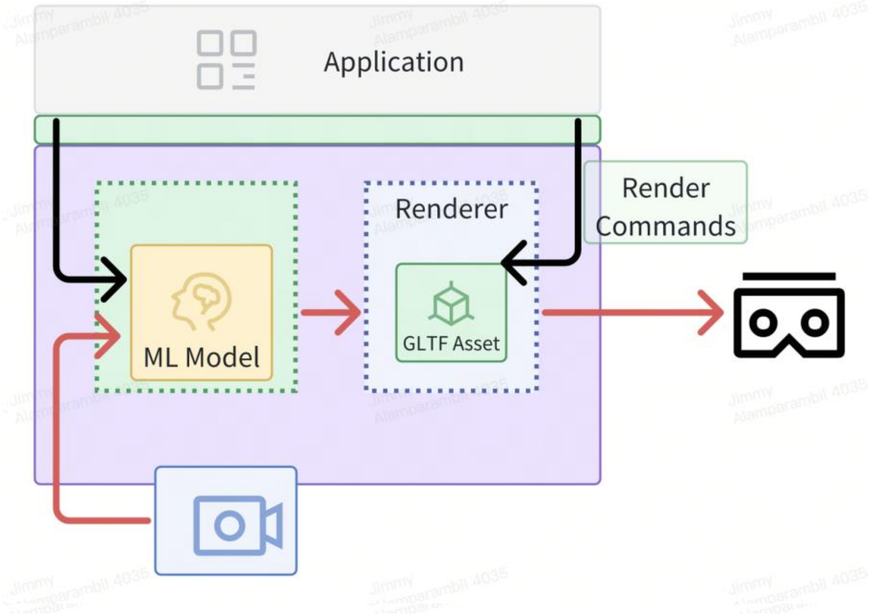
- **We cannot feed this output back to the application, as it could contain sensitive data!**

So how do we use it?



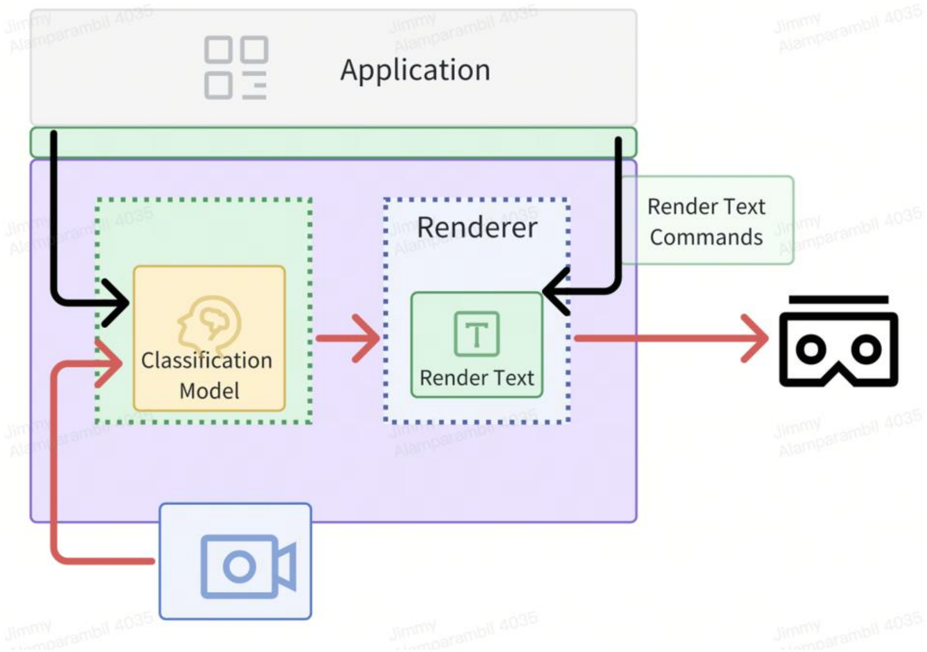
Renderer

- Instead you pass the output (after processing) to a renderer that renders something based on that output
- Application can pass down "render glTF" or other render commands down to the framework



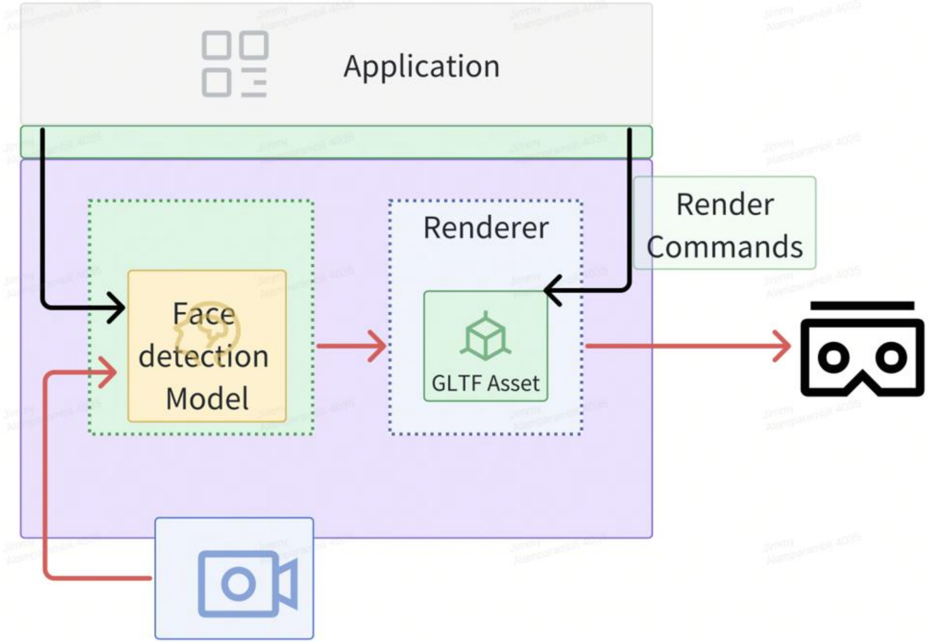
Example - Classification

- Pass down a classification model and render text commands
- Framework will print labels of objects that it recognizes in the surroundings

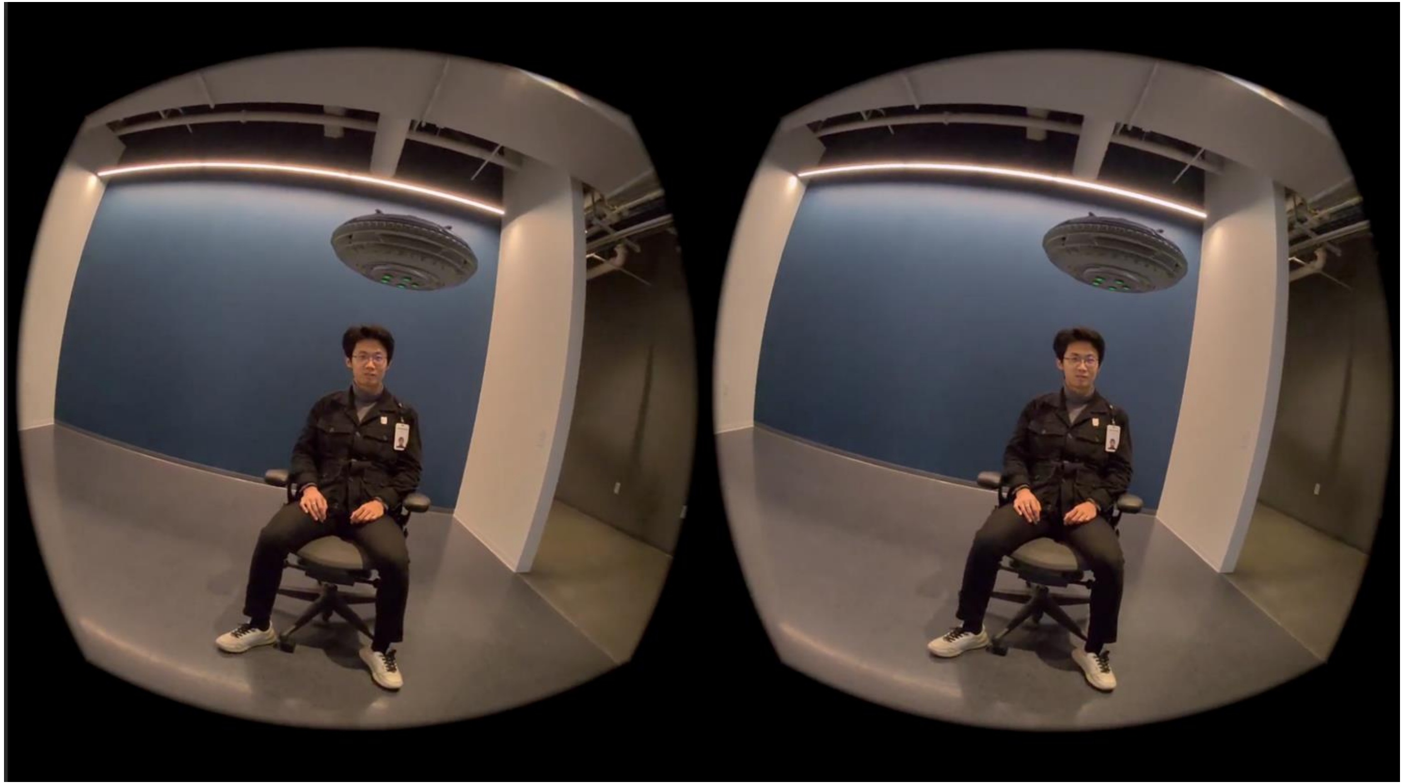


Example - Face position

- Pass down a face detection model, glTF of a ufo and render glTF command
- Framework will render ufo above the face it recognizes in the surroundings

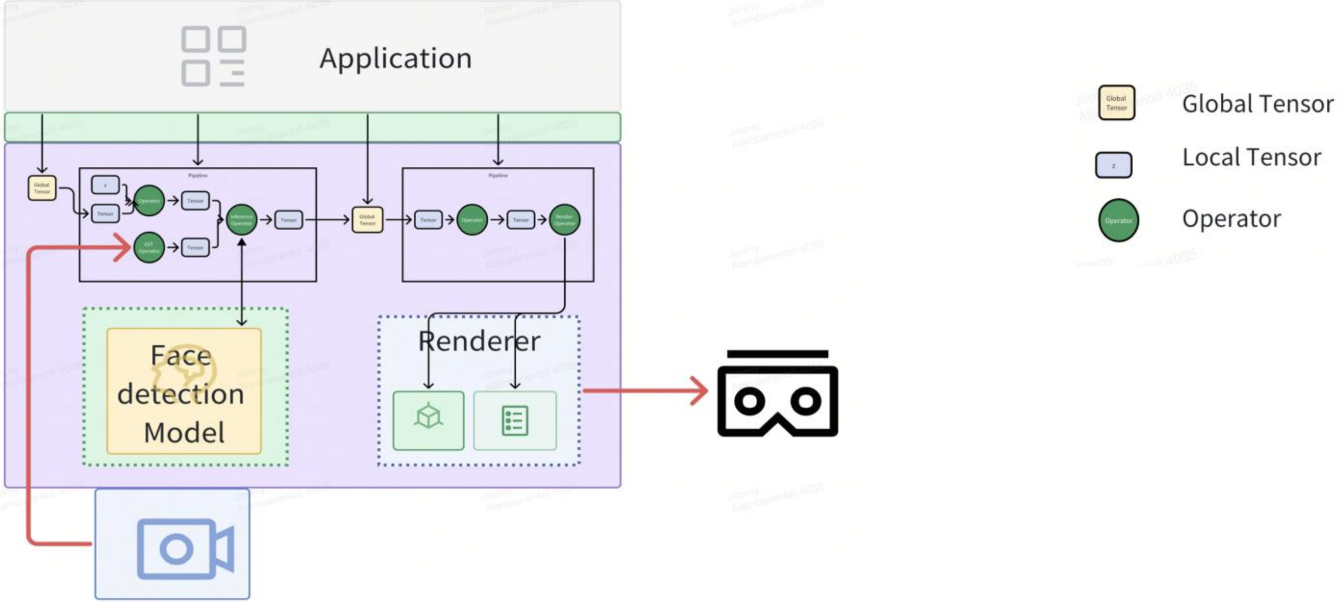


Example - Face position rendering



SecureMR Pipeline

- Implementing a feature or algorithm requires lots of data processing and logic in addition to the inference and rendering
- We provide a general purpose data processing pipeline using tensors (data) and operators (transformations) that developers can use
- Inference and rendering are also operators in the pipelines

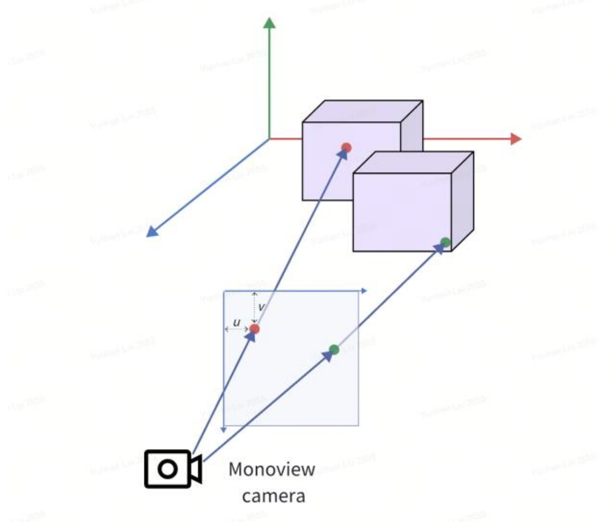


Tensors and Operators

- Tensors are containers of data that can abstract scalars, points, vectors, matrices, and images
- Operators carry out specific functionality needed by the algorithms
- Currently have included a small set of operators that are needed for most general use cases
- Looking for feedback on other operators that you might need for your algorithms

Stereoscopic View

- Another big difference between mobile devices and XR devices to consider
- Most vision based AI/ML models currently use a single RGB image as input, and output 2d coordinates for bounding boxes, vertices etc.
- For XR rendering to work accurately in stereoscopic views, we need 3d coordinates
- We have provided a UV_to_3D operator that does this, by aligning a depth buffer with the input RGB image we use



Developer Best Practices

Workflow

Step #1: Either train or find your intended ML model

Step #2: Use SecureMR tools to convert ML model into binary

Step #3: Create a pipeline using the API

Step #4: Add an inference operator with model binary

Step #5: Add pre processing and post processing operators

Step #6: Add a render operator, populated with a glTF asset

Step #7: Run the pipeline to render results of inference

To be released

- **Unity SDK**
  - C# API
  - Improved workflow
- **Native OpenXR extensions API**
  - Framework
  - Pipelines
  - Tensors
  - Operators
- **Tools**
  - Convert AI models into binary format to run in framework
  - Model evaluation
  - Android profiler
- **Samples**

Summary

- Privacy first XR framework lets you implement perception features on consumer headsets
- First of its kind, a blueprint for custom secure perception features as we move forward
- Infinitely customizable with off the shelf ML models or train your own!

Feature highlight

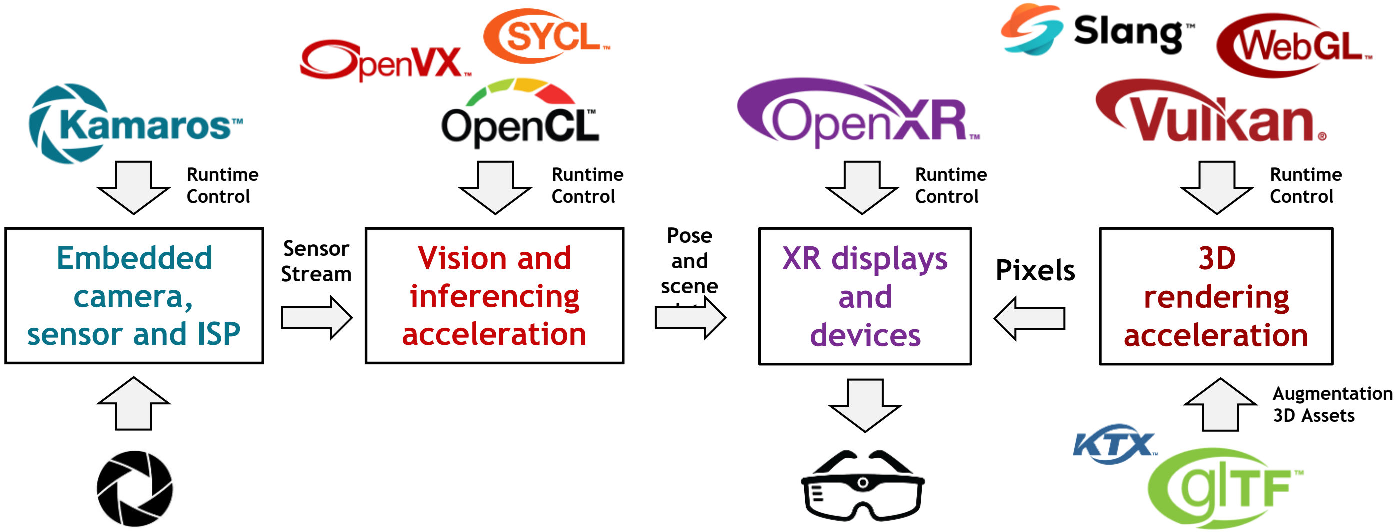
- Model inference is hardware accelerated
- Allows general purpose data processing and rendering to customize further
- Adapts existing ML model outputs for stereoscopic view on XR

PICO XR Developer Community Discord

Sign up for more information and early access!

Khronos Standards for Spatial Computing

- Ongoing discussions and proposals on how to evolve OpenXR to meet developer needs
  - Carefully considered additions can be widely adopted by the OpenXR ecosystem
- Opportunity to leverage other Khronos standards for camera control and inferencing
- Join Khronos to help inform and steer the evolution of open standards for XR!



OpenXR Development Resources & Tools

- OpenXR SDK
  - Headers, source code, and build scripts
  - https://github.com/KhronosGroup/OpenXR-SDK
- Reference Pages and Reference Guide
  - Developer documentation
- OpenXR Tutorial
  - For creating applications using Android, Linux, Windows
- Conformance Test Suite
  - For runtime developers to test, developed as open source
  - Part of the API Adopter Process to be an official OpenXR runtime requires passing these conformance tests
- Support & Community Forums
  - OpenXR on Discord
  - OpenXR Forum at Khronos
  - OpenXR Issue Tracker on GitHub
  - Developing OpenXR Resources? Let us know!

Beat Saber's PC implementation using OpenXR is portable to multiple devices

Get Involved!

Provide feedback and requirements on GitHub, Discord, or OpenXR Forums

Get questions answered and make suggestions for improvement!

Join Khronos and the OpenXR Working Group

https://www.khronos.org/openxr/

https://github.com/KhronosGroup/OpenXR-Docs

OpenXR Specification

OpenXR SDK GitHub

Khronos BOFs at SIGGRAPH Asia

Day	Time / Room	Session Title	Standards and Projects
Tuesday 3rd	1:00-2:00PM, G408	Khronos Fast Forward	Vulkan, OpenXR, Slang, ANARI, glTF
Wednesday 4th	1:00-2:00PM, G407	Slang Shading Language	Slang
Wednesday 4th	3:30-4:30PM, G407	Immersive Web with Khronos and W3C	WebGL, WebXR, WebGPU, three.js
Thursday 5th	2:15-3:15PM, G407	OpenXR Update and Roadmap	OpenXR
Thursday 5th	3:30-5:30PM, G407	Vulkan Update and Ecosystem	Vulkan, Vulkan SC, Slang
Friday 6th	11:00-12PM, G408	glTF 3D Transmission Format	glTF, VRM Avatar Format

All BOF slides and videos will be uploaded to the [Khronos SIGGRAPH event page]()

SIGGRAPH ASIA 東京
2024 TOKYO 3–6 DEC

Khronos BOFs

Khronos Information
www.khronos.org
memberservices@khronosgroup.org