OpenXR 1.0!

Brent E. Insko, PhD
Lead XR Architect at Intel & OpenXR Working Group Chair

SIGGRAPH, July 2019
OpenXR 1.0 Release is Here!

Khronos Releases OpenXR 1.0 Specification Establishing a Foundation for the AR and VR Ecosystem

Final specifications and shipping implementations freely available today. Growing adoption from XR industry and expanding ecosystem support.

July 29, 2019 – 6:00 AM PT – SIGGRAPH, Los Angeles – Today, The Khronos® Group, an open consortium of leading hardware and software companies creating advanced acceleration standards, announces the ratification and public release of the OpenXR™ 1.0 specification together with publicly available implementations and substantial ecosystem momentum. OpenXR is a unifying, royalty-free, open standard that provides high-performance, cross-platform access to virtual reality (VR) and augmented reality (AR)—collectively known as XR—platforms and devices. The new specification can be found on the Khronos website and via GitHub.

“The working group is excited to launch the 1.0 version of the OpenXR specification, and the feedback from the community on the provisional specification released in March has been invaluable to getting us to this significant milestone,” said Brent Insko, OpenXR working group chair and lead XR architect at Intel. “Our work continues as we now finalize a comprehensive test suite, integrate key engine support, and plan the next set of features to evolve a truly vibrant, cross-platform standard for XR platforms and devices. Now is the time for software developers to start putting OpenXR to work.”

After gathering feedback from the XR community during the public review of the provisional specification, improvements were made to the OpenXR input subsystem, game engine editor support, and loader. With this 1.0 release, the working group will evolve the standard while maintaining full backwards compatibility from this point onward, giving software developers and hardware vendors a solid foundation upon which to deliver incredible, portable, user experiences.
Agenda

- What is OpenXR?
- A Brief History of the Standard
- What are the Problems we are trying to Solve
- OpenXR Timeline of Development
- Brief Overview
- Demos
- What’s Next?
- Recap
A Brief Aside…

- Talk assume that you know:
  - A little bit about VR and AR
  - A little bit about programming and very basic real-time rendering
  - Nothing about the specification process
  - Nothing about any of the other Khronos specifications
What is OpenXR?

OpenXR is a royalty-free, open standard that provides high-performance access to Augmented Reality (AR) and Virtual Reality (VR)—collectively known as XR—platforms and devices.
A Brief History of OpenXR

• Among the first VR hardware available 2016

• Need applications...
  - Each platform provided an SDK to interface with the hardware
  - Each was different from the other
XR Ecosystem Fragmentation

- Increased development time and therefore cost.

- Increased validation overhead and therefore cost.

- Time and resources spent developing one title, impacts developers’ ability to create more titles.
## Major XR Runtimes

<table>
<thead>
<tr>
<th></th>
<th>Virtual Reality</th>
<th>Augmented Reality</th>
<th>Console VR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC</td>
<td>AIO</td>
<td>Mobile</td>
</tr>
<tr>
<td>Oculus Rift</td>
<td>SteamVR</td>
<td>Mixed Reality</td>
<td>Oculus Go</td>
</tr>
<tr>
<td>Facebook</td>
<td>Valve</td>
<td>Microsoft</td>
<td>Facebook</td>
</tr>
<tr>
<td>OS support</td>
<td>Windows</td>
<td>Windows</td>
<td>Android</td>
</tr>
</tbody>
</table>

© Khronos® Group Inc. 2019 - Page 9
OpenXR

- Recognizing the problem, several companies got together in late 2016 / early 2017 and formed the OpenXR working group in Khronos.
OpenXR - Solving XR Fragmentation
OpenXR - Solving XR Fragmentation

Before OpenXR
XR Market Fragmentation

After OpenXR
Wide interoperability of XR apps and devices
VR Software Stack (Example)

VR APIs
- Application interface
- Driver(HW) interface

VR Application
Game Engine
Server
Compositor
VR Runtime

Vulkan | Direct3D | OpenGL (+ other system APIs)

Windows | Linux | Android

VR Hardware

Browser
WebXR
OpenXR Architecture

OpenXR does not replace XR Runtime Systems!
It enables any XR Runtime to expose CROSS-VENDOR APIs to access their functionality

- **XR Apps and Engines**
- **OpenXR API**
- **XR Vendor Runtime System**
  - Distortion Correction and Display Output
  - Coordinate System Unification and Prediction
- **OpenXR Device Plugin Extension**
  (Optional - Coming soon)
- **Vendor-supplied Device Drivers**
- **XR Devices**

Current Device State
Controller/Peripheral State
Normalized Poses
Input Events

Outgoing Requests
Pre-distortion image to display Haptics

Outgoing Requests
Post-distortion image to display Haptics

- **Current Device State**
  - Controller/Peripheral State
  - Raw Poses

© Khronos® Group Inc. 2019 - Page 14
The Structure
OpenXR Philosophies

1. Enable both VR and AR applications
   The OpenXR standard will unify common VR and AR functionality to streamline software and hardware development for a wide variety of products and platforms.

2. Be future-proof
   While OpenXR 1.0 is focused on enabling the current state-of-the-art, the standard is built around a flexible architecture and extensibility to support rapid innovation in the software and hardware spaces for years to come.

3. Do not try to predict the future of XR technology
   While trying to predict the future details of XR would be foolhardy, OpenXR uses forward-looking API design techniques to enable engineers to easily harness new and emerging technologies.

4. Unify performance-critical concepts in XR application development
   Developers can optimize to a single, predictable, universal target rather than add application complexity to handle a variety of target platforms.
Where are we on the timeline?

Call for Participation / Exploratory Group Formation
Fall F2F, October 2016: Korea

Statement of Work / Working Group Formation
Winter F2F, January 2017: Vancouver

Defining the MVP
Fall F2F, September 2017: Chicago

First Public Information
GDC, March 2018

First Public Demonstrations
SIGGRAPH, August 2018

Release Provisional Specification!
GDC, March 2019

Resolving Issues / Incorporating feedback
Spring F2F, April 2019: Singapore
Summer F2F, July 2019: Bellevue

Release 1.0 Specification!
SIGGRAPH, July 2019

Conformance Tests and Adopters Program

Feedback

Finalize Implementations

Enable Conformant Implementations to Ship

OpenXR 1.0 Specification Released
The OpenXR Loader

- A separate component for supporting multiple runtimes on a single system
- Similar mechanism to other Khronos APIs
- Loader determines the runtime to use for the requesting application
- Complexity can vary from “just pick one” to more intelligent decisions based on user factors, hardware, running apps, etc
- Not all systems will have a loader
Initial steps: API Layers & Extensions

- API Layers
  - `xrDoSomething()`
  - `Layer1::xrDoSomething()`
  - `LayerN::xrDoSomething()`
  - `MyRuntime::xrDoSomething()`

- Extensions
  - `xrEnumerateInstanceExtensionProperties()`
The Instance

**XrInstance:**
- The XrInstance is basically the application’s representation of the OpenXR runtime
- Can create multiple XrInstances, if supported by the runtime
- `xrCreateInstance` specifies app info, layers, and extensions

```
Application
```

```
xrCreateInstance
```

```
Loader
```

```
OpenXR Runtime
```

The System

**XrSystem:**
- OpenXR groups physical devices into logical systems of related devices
- `XrSystem` represents a grouping of devices that the application chooses to use (e.g. HMD and controllers)
- `XrSystems` may have display, input, tracking, etc.

- `xrGetSystem` also returns the type of form factor to be used
Form Factors

- Currently two `XrFormFactors` in the specification:
  - `XR_FORM_FACTOR_HANDHELD_DISPLAY`
  - `XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY`

<table>
<thead>
<tr>
<th>Camera Passthrough AR</th>
<th>Stereoscopic VR / AR</th>
<th>Projection CAVE-like</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Camera Passthrough AR" /></td>
<td><img src="image2.png" alt="Stereoscopic VR / AR" /></td>
<td><img src="image3.png" alt="Projection CAVE-like" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>View Type</th>
<th><code>XrFormFactor</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>One View</td>
<td><code>XR_FORM_FACTOR_HANDHELD_DISPLAY</code></td>
</tr>
<tr>
<td>Two View</td>
<td><code>XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY</code></td>
</tr>
<tr>
<td>Twelve Views</td>
<td>(future support)</td>
</tr>
</tbody>
</table>

Photo Credit: Dave Pape
View Configurations

- Currently two `XrViewConfiguration` types in the specification:
  - `XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO`
  - `XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO`  

<table>
<thead>
<tr>
<th>Camera Passthrough AR</th>
<th>Stereoscopic VR / AR</th>
<th>Projection CAVE-like</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Camera Passthrough AR" /></td>
<td><img src="image" alt="Stereoscopic VR / AR" /></td>
<td><img src="image" alt="Projection CAVE-like" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>One View</th>
<th>Two View (one per eye)</th>
<th>Twelve Views (six per eye)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XR_FORM_FACTOR_HANDHELD_DISPLAY</td>
<td>XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY</td>
<td>(future support)</td>
</tr>
<tr>
<td>XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO</td>
<td>XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO</td>
<td>(future support)</td>
</tr>
</tbody>
</table>

Photo Credit: Dave Pape
The Session

Session:
- A session is how an application indicates it wants to render and output VR / AR frames
- An app tells the runtime it wants to enter an interactive state by beginning a session with `xrBeginSession` and ending it with `xrEndSession`
- No session, no frames.

![Diagram](attachment:image.png)
Session Lifecycle

- xrCreateInstance
  - xrGetSystem
  - xrCreateSession

- IDLE
  - user: quit
  - runtime: session is ready
  - xrEndSession

- READY
  - xrBeginSession

- SYNCHRONIZED

- STOPPING
  - runtime: stop session

- VISIBLE

- FOCUSED

- EXITING
  - runtime: losing system or device

- LOSS_PENDING
  - any
  - optional: poll xrGetSystem
  - xrCreateSession again

- xrDestroySession

- xrDestroyInstance
Events

Events are messages sent from the runtime to the application. They’re put into a queue by the runtime, and read from that queue by the application using `xrPollEvent`

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XrEventDataEventsLost</td>
<td>event queue has overflowed and some events were lost</td>
</tr>
<tr>
<td>XrEventDataInstanceLossPending</td>
<td>application is about to lose the instance</td>
</tr>
<tr>
<td>XrEventDataInteractionProfileChanged</td>
<td>active input form factor for one or more top level user paths has changed</td>
</tr>
<tr>
<td>XrEventDataReferenceSpaceChangePending</td>
<td>runtime will begin operating with updated space bounds</td>
</tr>
<tr>
<td>XrEventDataSessionStateChanged</td>
<td>application has changed lifecycle state</td>
</tr>
</tbody>
</table>
Input and Haptics
When user clicks button “a” it results in the user teleporting.
Input and Haptics

Input in OpenXR goes through a layer of abstraction built around Input Actions (XrActions). These allow application developers to define input based on resulting action (e.g. “Grab”, “Jump,” “Teleport”) rather than explicitly binding controls.

While the application can suggest recommended bindings, it is ultimately up to the runtime to bind input sources to actions as it sees fit (application’s recommendation, user settings in the runtime’s UI, etc).
Input and Haptics - Interaction Profiles

- Collections of input and output sources on physical devices
- Runtimes can support multiple interaction profiles

ControllerCorp’s Fancy_Controller:
- /user/hand/left
- /user/hand/right
- /input/a/click
- /input/b/click
- /input/c/click
- /input/d/click
- /input/trigger/click
- /input/trigger/touch
- /input/trigger/value
- /output/haptic
Interaction profiles for many current products are predefined in the OpenXR specification including:

- Google Daydream* controller
- HTC Vive and Vive Pro* controllers
- Microsoft* Mixed reality motion controllers
- Microsoft* Xbox controller
- Oculus Go* controller
- Oculus Touch* controllers
- Valve Index* controllers
Input and Haptics - Runtime Binding Decision - Why?

- Runtime ultimately decides the bindings
  - “dev teams are ephermal, games last forever”
  - More likely the runtime is updated than individual games

- Reasons for selecting different bindings:
  - 1. this runtime does not have ControllerCorp’s fancy_controller currently attached, but it knows how to map the inputs and outputs to the controllers that *are* attached
  - 2. Some runtimes can support user mapping of inputs such that the controls per game can be customized by the user, such as swapping trigger and button ‘a’, this enables customization without the original application knowing about it
  - 3. Some future controller is developed but the application is not updated for it, a new interaction profile can help map the actions to the new inputs
  - 4. Accessibility devices can be used and input mapped appropriately
Input and Haptics

```
/user/hand/left/input/a/click
(/interaction_profile/ControllerCorp/fancy_controller/input/a/click)
```

**OpenXR Runtime**

```
.../input/trigger/click          |   Explode
.../input/a/click               |   Telecom
```

**XrAction: “Teleport”**
Input and Haptics

Haptics build upon the same XrAction system, and have their own Action Type: XR_HAPTIC_VIBRATION. Just like other XrActions, they can be used with XrActionSets, but unlike inputs, they are activated with xrApplyHapticFeedback.

Currently, only XrHapticVibration is supported:

- Start Time
- Duration (s)
- Frequency (Hz)
- Amplitude (0.0 - 1.0)

xrStopHapticFeedback can also be called to immediately end haptic feedback.

We expect that many more haptic types will be added through extensions as the technology develops.
That’s it for Core API coverage today
Extensions

Core Standard
Core concepts that are fundamental to the specification for all use cases
Examples: Instance management, tracking, frame timing

KHR Extensions
Functionality that a large class of runtimes will likely implement
Examples: Platform support, Graphic API Extensions

EXT Extensions
Functionality that a few runtimes might implement
Examples: Performance Settings, Thermals, Debug Utils

Vendor Extensions
Functionality that is limited to a specific vendor
Examples: Device specific functionality
Current KHR Extensions

- **Platform-specific support:**
  - KHR_android_create_instance
  - KHR_android_surface_swapchain
  - KHR_android_thread_settings

- **Graphics API support:**
  - KHR_D3D11_enable
  - KHR_D3D12_enable
  - KHR_opengl_enable
  - KHR_opengl_es_enable
  - KHR_vulkan_enable
  - KHR_vulkan_swapchain_format_list

- **Support for specific XR layer types:**
  - KHR_composition_layer_cube
  - KHR_composition_layer_cylinder
  - KHR_composition_layer_depth
  - KHR_composition_layer_equirect

- **Performance improvement by masking non-visible portions of the display:**
  - KHR_visibility_mask

- **Time Conversion functions:**
  - KHR_convert_timespec_time
  - KHR_win32_convert_performance_counter_time
What hasn’t made it in?
What hasn’t made it in?

• Top priority: solve application fragmentation
Device Plugin Extension

MyDevice1 (HMD)
/devices/MyVendor/MyDevice
Display
Trackability
Audio Out
...

xrDevicePluginSetDeviceStatusKHR()
There are a list of things to consider for 1.1 and for additional extensions

- Many of the items on the list are obvious next steps in the progress of AR and VR development

- Won’t list them here 😊

- But feel free to send us your list of requests in via the feedback channels we’ll provide in a sec
OpenXR in Action... Time for Demos
OpenXR 1.0 Release is Here!

Oculus, Hololens and Vive headsets will soon be able to share apps.

The OpenXR standard could lead to loads of cross-platform AR and VR apps.

10 Comments 166 Shares

OpenXR - Solving XR Fragmentation

An effort to standardize certain aspects of VR and AR applications gains wide industry support today with the release of version 1.0 of the OpenXR specification.

https://www.engadget.com/2019/07/30/openxr-1-launch/
https://uploadvr.com/openxr-specification-standard/
https://pcper.com/2019/07/the-khronos-group-ratifies-and-releases-openxr-1-0/
“OpenXR 1.0 release is a huge milestone and AMD is proud to have been a member in its creation. The expanding XR industry and ecosystem continues to be a key focus for AMD and we are excited by the potential for market growth that OpenXR 1.0 enables. As always, AMD is a proponent of open industry standards,” said Daryl Sartain, Director of XR at AMD.

“Arm is focused on developing technology innovations that power the next generation of untethered, standalone AR/VR devices. The release of the OpenXR 1.0 specification will further enable us to break barriers for cross-platform XR applications, while bringing the performance and efficiency required to support these complex, immersive use cases,” said Roger Barker, director of IP solutions, Immersive Experience Group, Arm.

“As part of our unwavering commitment to open source and open standards, Collabora is proud to be part of bringing OpenXR 1.0 to life. We are pioneering the Monado open source runtime for OpenXR to ensure the future of XR is truly open and accessible to all hardware vendors. As the OpenXR specification editor, I am grateful for the diligent efforts of the working group, as well as the community feedback that shaped this release,” said Ryan Pavlik, OpenXR Specification Editor, XR Principal Software Engineer at Collabora.

“OpenXR 1.0 steers us toward an alignment of many crucial emerging interface platforms. CTRL-labs is excited to contribute to this important step forward and to give developers the tools they need to explore neural interfaces.” said Attila Maczak, Software Architect at CTRL-labs.

“We're thrilled to support the OpenXR 1.0 release, along with all of the Khronos Group members who have worked tirelessly to create the standard. Unreal Engine led the way with support for the OpenXR 0.9 provisional specification, and we're excited to move the 1.0 revision forward in collaboration with our hardware partners releasing at the same time. Epic believes that open standards are essential to driving technology and bridging the gaps between digital ecosystems,” said Jules Blok, Epic Games.

“Facebook and Oculus continue to believe in the value the OpenXR standard delivers to users and developers. We plan to provide runtime support for apps built on OpenXR 1.0 on the Rift and Quest platforms,” said Nate Mitchell, Oculus Co-founder and head of VR product, Facebook.

“We're excited about OpenXR and believe this is a significant step towards a more open ecosystem.” said Vinay Narayan, vice president, platform strategy, HTC. “Bringing the community together to help define standards and best practices, allows all of us to move forward, together.”

“The mobile era of computing was defined and ultimately constrained by closed ecosystems. With mixed reality, the next wave of computing must be and will be open,” said Don Box, Technical Fellow at Microsoft. “Today, Microsoft is proud to release the first OpenXR 1.0 runtime that supports mixed reality, for all Windows Mixed Reality and HoloLens 2 users. We are excited to now work with the OpenXR community to design the key extensions that will bring mixed reality to life, with full support by end of year for HoloLens 2 hand tracking, eye tracking, spatial mapping and spatial anchors.”

“Congratulations to the OpenXR team on this important 1.0 release. The flexible, extensible design of OpenXR 1.0 will support innovative, next-generation graphics technologies that accelerate XR applications and even enable new XR use-cases,” said David Weinstein, Director of Virtual Reality at NVIDIA.

“OpenXR provides a solid foundation for developers to more easily support a broader range of platforms and devices. 1.0 is an exciting milestone and it is just the beginning. This release will open new doors for developers to experiment and extend the capabilities, pushing the VR and AR industry into the future.” said Jared Cheshier, CTO at Pluto VR.

“It’s great to see this important milestone finally completed and we are excited by the promise OpenXR holds for lowering the barrier for creating XR applications across a wide array of devices. Now that the stable 1.0 release of OpenXR is out and we are beginning to see industry adoption, Tobii will work diligently to unlock support for eye tracking through OpenXR extensions for eye gaze interaction and foveated rendering.” said Denny Rönngren, architect at Tobii.

“Unity is committed to being an open and accessible platform and we remain supportive of open standards for XR applications and devices,” said Ralph Hauwert, vice president of platforms at Unity Technologies. "To that end, we're excited about OpenXR and believe this is a significant step towards a more open ecosystem.""OpenXR is an important milestone for VR. This API will allow games and other applications to work easily across a variety of hardware platforms without proprietary SDKs. Valve is happy to have worked closely with other VR industry leaders to create this open standard, and looks forward to supporting it in SteamVR,” said Joe Ludwig, programmer at Valve.

“Varjo is creating the world’s most groundbreaking VR/AR/XR hardware and software by merging the real and digital worlds seamlessly together in human-eye resolution. We’re excited for the release of the OpenXR 1.0 specification because it ensures the enterprise community has compatible, easy access to the best XR technology on the market today while removing barriers to future innovation,” said Rémi Arnaud, principal architect at Varjo.
What Resources Are Available?
What Resources Are Available?

• Fair warning:
  - This is our working group’s first ever 1.0 release
  - Much of the effort and time leading to the release was completing the release, the supporting software and infrastructure may be lagging
  - We think everything is complete, but let us know if we’ve missed something
What Resources Are Available?

200+ page specification
What Resources Are Available?

200+ page specification

Reference Pages

xrCreateInstance(3) Manual Page

NAME
xrCreateInstance - Creates an OpenXR instance

C Specification
The xrCreateInstance function is defined as:

void *xrCreateInstance(
    const struct xrInstanceCreateInfo *createInfo,
    void *instance,
    uint32_t contextFlags,
);

Parameters

Parameter Descriptions

- createInfo points to an instance of xrInstanceCreateInfo containing details of the instance.
- instance points to an instance handle in which the resulting instance is returned.

Description

xrCreateInstance creates the instance. If successful, the function returns the instance handle in the instance parameter. If the instance handle is not valid, then either the function returns NULL, or the call returns XR_ERROR_INVALID_NULL_HANDLE. The instance is also checkable if a mandatory platform-specific extension is defined and the platform is compatible with the extension. If the instance is not checkable, then the instance is created.

Valid Usage (Implicit)

- instance must be a pointer to a valid struct xrInstanceCreateInfo
- instance is an instance handle
What Resources Are Available?

- 200+ page specification
- Reference Pages
- Overview Guide
What Resources Are Available?

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

- Contains the source for generating the specification document and reference pages, scripts to be added soon

- Contains the openxr header files
What Resources Are Available?

- [https://github.com/KhronosGroup/OpenXR-Registry](https://github.com/KhronosGroup/OpenXR-Registry)

- Contains the specification, reference pages, and overview guide

---

**OpenXR-Registry**

The OpenXR-Registry repository contains the OpenXR™ API and Extension Registry, including generated specifications and reference pages, and reference cards. The sources for these documents are found in the separate [https://github.com/KhronosGroup/OpenXR-Docs](https://github.com/KhronosGroup/OpenXR-Docs) repository; this repository is used as a backing store for the web view of the registry at [https://www.khronos.org/registry/OpenXR/](https://www.khronos.org/registry/OpenXR/). Commits to the master branch of OpenXR-Registry will be reflected in the web view.

Interesting files in this repository include:

- `specs/1.0/` - OpenXR 1.0 API specifications and reference pages.
- `specs/0.90/` - OpenXR 0.90 Provisional API specifications and reference pages and API reference card.
- `index.php` - toplevel index page for the web view of [https://www.khronos.org/registry/OpenXR/](https://www.khronos.org/registry/OpenXR/). This relies on PHP include files found elsewhere on [www.khronos.org](http://www.khronos.org) and so is not very useful in isolation.
What Resources Are Available?

- [https://github.com/KhronosGroup/OpenXR-SDK-Source](https://github.com/KhronosGroup/OpenXR-SDK-Source)

Contains the source for:
- Loader
- Some basic API layers
- Test sample

For the current best example code, see: src/tests/hello_xr
What Resources Are Available?

- [https://github.com/KhronosGroup/OpenXR-SDK](https://github.com/KhronosGroup/OpenXR-SDK)

- Contains Generated Files

- Use for building on Windows and Linux

- Embed this in your projects
Additional Resources

- **OpenXR Landing Page - Specification, Reference Pages, Sample Code, Overview**
  - [https://www.khronos.org/openxr](https://www.khronos.org/openxr)

- **OpenXR Forum and Slack Channel**
  - Forum: [https://khr.io/openxrfeedback](https://khr.io/openxrfeedback)
  - Discussion: [https://khr.io/slack](https://khr.io/slack)

- **Vendor preview implementations and integration**
  - Collabora: open source implementation
    [http://monado.dev](http://monado.dev)
  - Microsoft: OpenXR runtime for Windows Mixed Reality headsets
    [https://aka.ms/openxr](https://aka.ms/openxr)
  - Epic Games: Unreal Engine 4.23 Preview 4 with OpenXR 1.0 plugin
    [https://www.unrealengine.com](https://www.unrealengine.com)

- **Khronos SIGGRAPH Sessions - including OpenXR Presentation and demos**
  - [https://www.khronos.org/events/2019-siggraph](https://www.khronos.org/events/2019-siggraph)
What’s Next?
What’s Next?

Call for Participation / Exploratory Group Formation
Fall F2F, October 2016: Korea

Statement of Work / Working Group Formation
Winter F2F, January 2017: Vancouver

Defining the MVP
Fall F2F, September 2017: Chicago

First Public Information
GDC, March 2018

First Public Demonstrations
SIGGRAPH, August 2018

Release Provisional Specification!
GDC, March 2019

Resolving Issues / Incorporating feedback
Spring F2F, April 2019: Singapore
Summer F2F, July 2019: Bellevue

Release 1.0 Specification!
SIGGRAPH, July 2019

Conformance Tests and Adopters Program
Feedback
Finalize Implementations
Enable Conformant Implementations to Ship

Finish Conformance Suite and Release Conformant Implementations
Thanks!

- To these companies for enabling their engineers to dedicate time to OpenXR!
Thanks to the Engineers!

Recap

- What is OpenXR?
- A Brief History of the Standard
- What are the Problems we are trying to Solve
- OpenXR Timeline of Development
- Brief Overview
- Demos
- What’s Next?
- Recap
OpenXR Win-Win-Win

**XR Vendors**
Can bring more applications onto their platform by leveraging the OpenXR content ecosystem

**XR ISVs**
Can easily ship on more platforms for increased market reach

**XR End-Users**
Can run the apps they want on their system - reducing market confusion and increasing consumer confidence
Khronos APIs for XR

High-performance, low-latency 3D rendering and composition*
  Multiview
  Context priority
  Front buffer rendering
  Tiled rendering (beam racing)
  Variable rate rendering

Display, composition and optical correction parameters

Cross-platform access to XR HMDs and sensors
  XR application lifecycle
  Frame timing and display composition
  Sensor tracking and pose calculation
  Input device discovery and events
  Haptics Control

* OpenXR can be used with other 3D APIs such as Direct3D, OpenGL and OpenGL ES
Before we go...
Before we go... as always give us Feedback!

- Tell us:
  - What should be in the spec
  - What shouldn’t be in the spec
  - How things need to be added for your application/runtime/hardware/OS/...
Join Khronos!

- Get more involved
- Have direct impact on the direction of the API
- Be part of the effort to deliver OpenXR 1.1!
Quick reminder…

• Khronos networking reception tonight, here @ 5:30

• Demos will be shown
Thank You!