



K H R O N O S
GROUP

OpenXR State of the Union

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

GDC, March 2019

OpenXR Provisional 0.90 Release is Here!



Khronos Releases OpenXR 0.90 Provisional Specification for High-performance Access to AR and VR Platforms and Devices

March 18, 2019 – 6:00 AM PT – Game Developer Conference, San Francisco – 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™ 0.90 provisional specification. OpenXR is a unifying, royalty-free, open standard that provides high-performance access to augmented reality (AR) and virtual reality (VR)—collectively known as XR—platforms and devices. The new specification can be found on the [Khronos website](#) and is released in provisional form to enable developers and implementers to provide feedback at the [OpenXR forum](#).

The OpenXR 0.90 provisional release specifies a cross-platform Application Programming Interface (API) enabling XR hardware platform vendors to expose the functionality of their runtime systems. By accessing a common set of objects and functions corresponding to application lifecycle, rendering, tracking, frame timing, and input, which are frustratingly different across existing vendor-specific APIs, software developers can run their applications across multiple XR systems with minimal porting effort—significantly reducing industry fragmentation.

The Khronos OpenXR working group was formed in early 2017 with the support and participation of leading XR companies. Throughout the development of the specification,

Press announcement



OpenXR™

UNIFYING REALITY

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.

OpenXR 0.90 is Here!

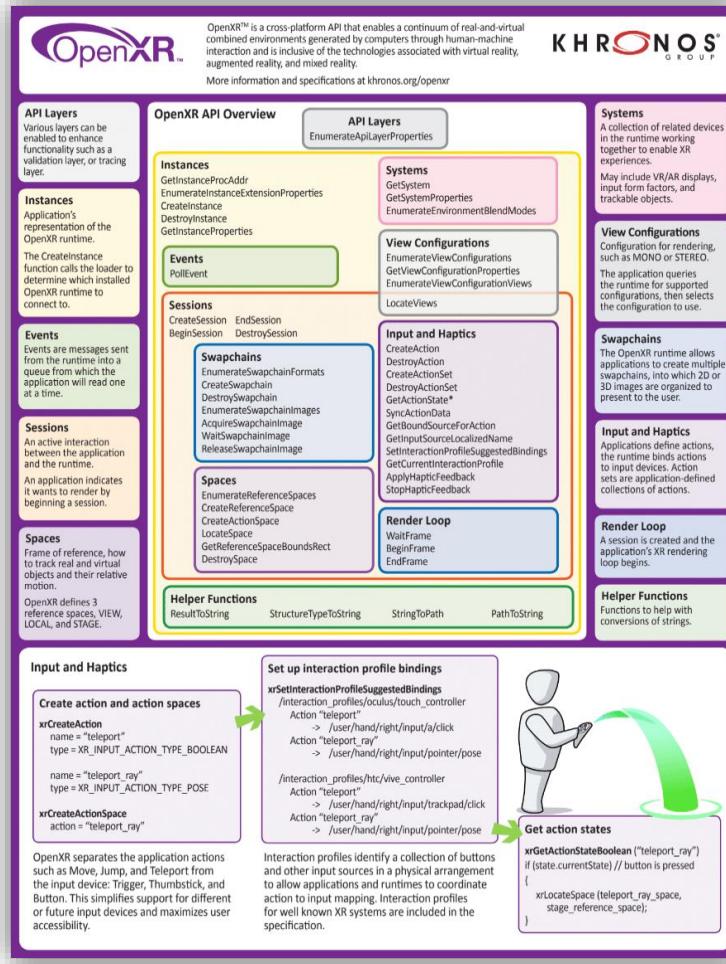
The OpenXR 0.90 provisional specification was released on March 18th 2019

[Press Release](#) [Provisional Specification](#) [Sample Code](#) [Launch Slides](#)

[Reference Pages](#) [Overview Guide](#) [Merchandise](#)

Updated landing page

OpenXR Overview Guide



For video viewers download from: <https://www.khronos.org/openxr> “Overview Guide”

Agenda

- What is OpenXR?
- A Brief History of the Standard
- What are the Problems we are trying to Solve
- OpenXR Timeline of Development
- Technical Walkthrough
- Provisional Release
- What's Next?
- Recap
- Call To Action
- Questions

A Brief Aside...

- **Note: This is a provisional release**
 - Things are likely to change between now and final release of the 1.0 specification
- **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

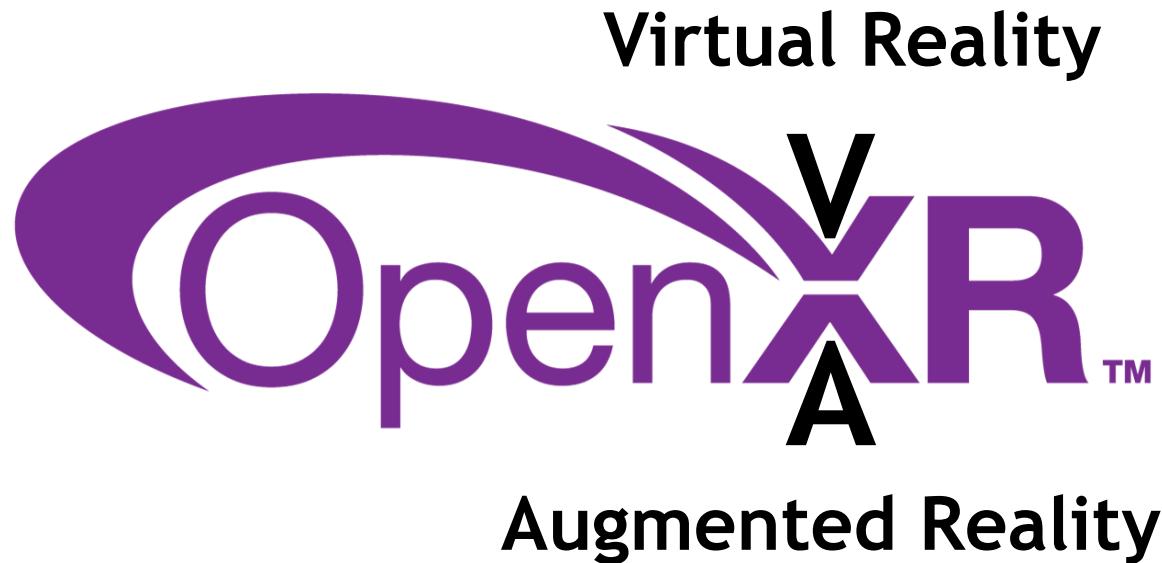
- Increased development time and therefore cost.
- Increased validation overhead and therefore cost.
- Time and resources spent developing one title, impacts developers' ability to create more titles.

Major XR Runtimes

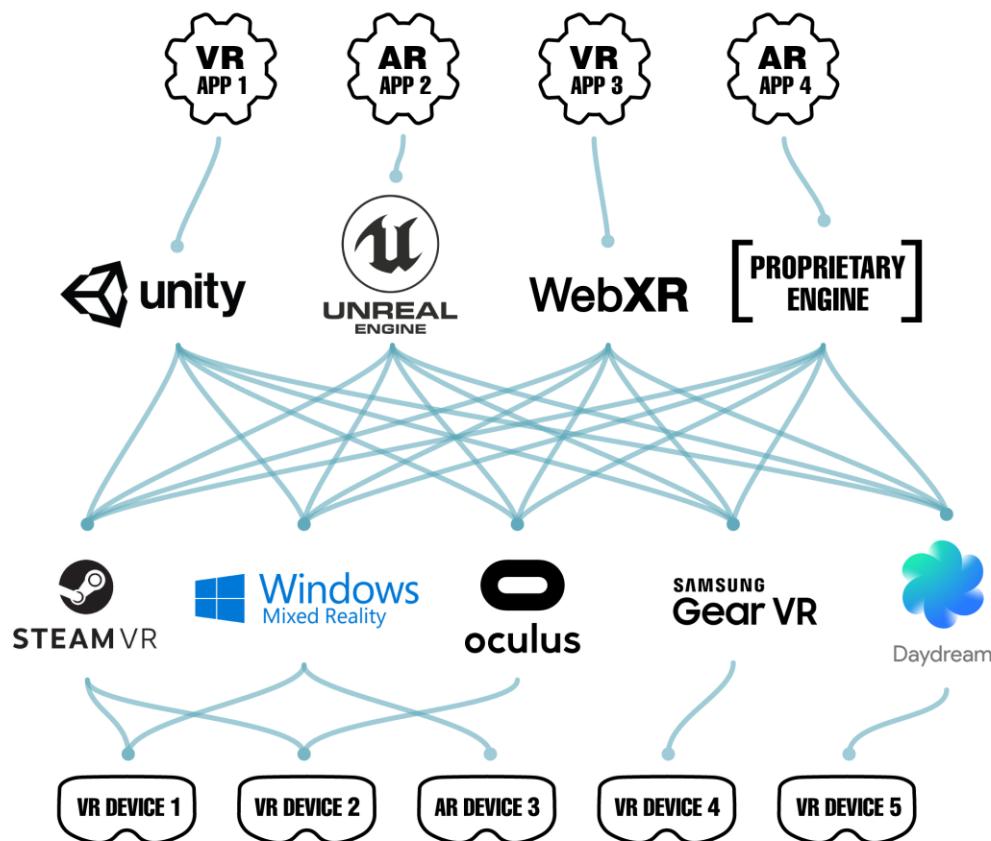
	Virtual Reality						Augmented Reality				Console VR
	PC			AIO	Mobile		AIO		Mobile		
Company	Oculus Rift	SteamVR	Mixed Reality	Oculus Go	Daydream	GearVR	HoloLens	ML1	ARKit	ARCore	PSVR
OS support											

OpenXR

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

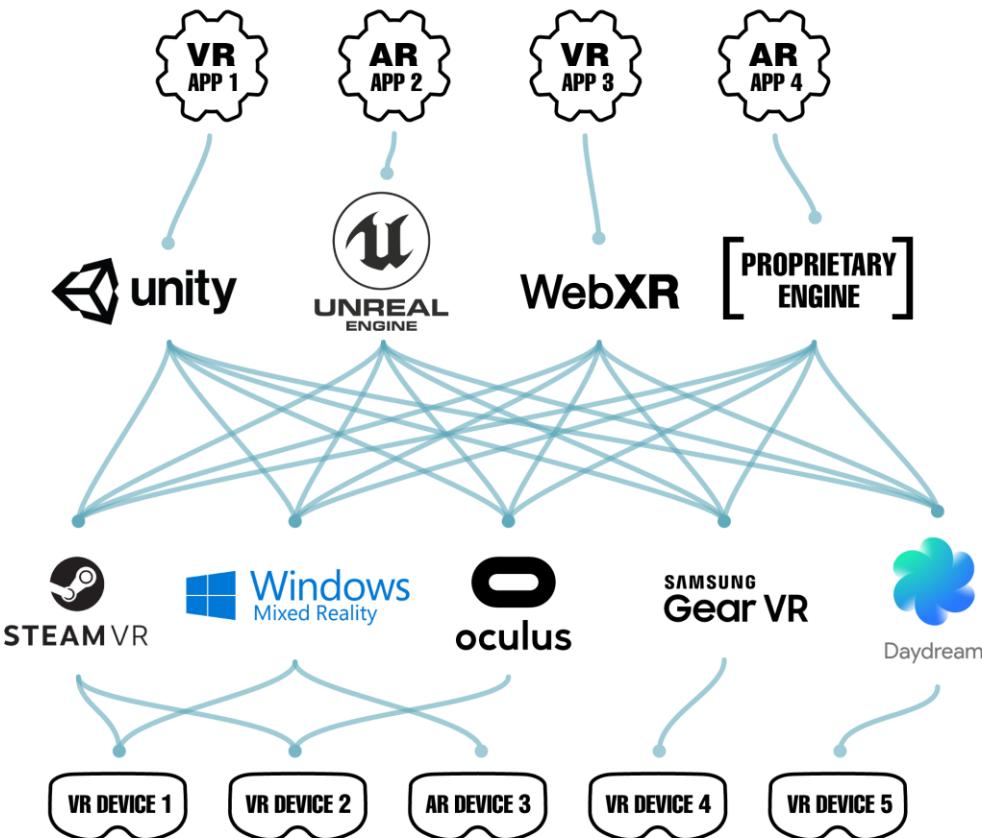


OpenXR - Solving XR Fragmentation

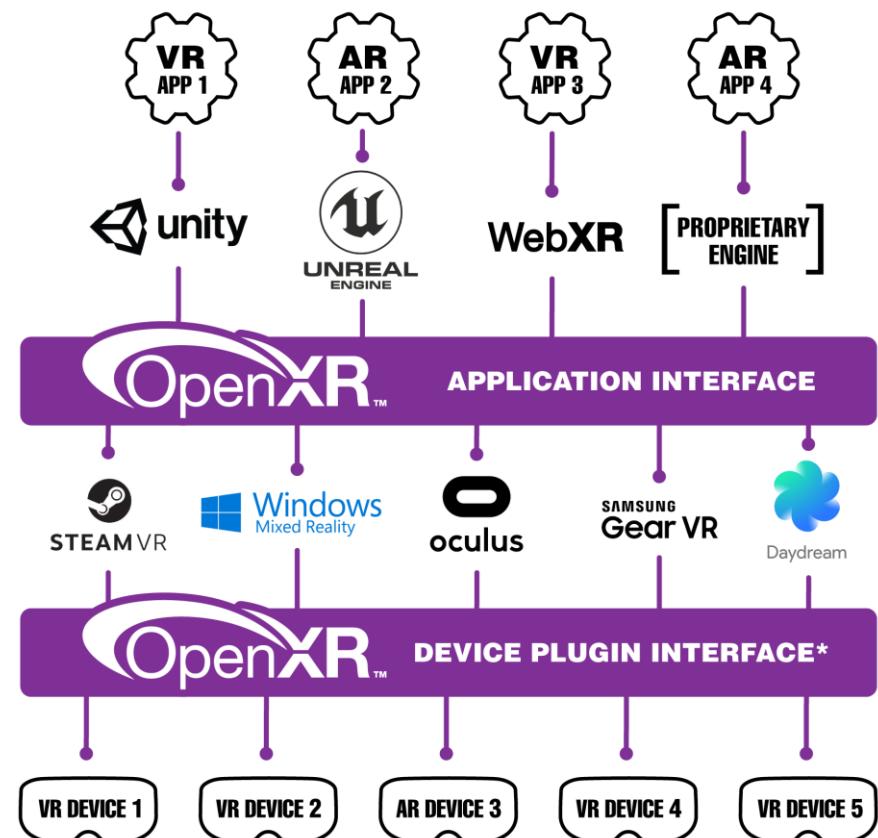


Before OpenXR
XR Market Fragmentation

OpenXR - Solving XR Fragmentation

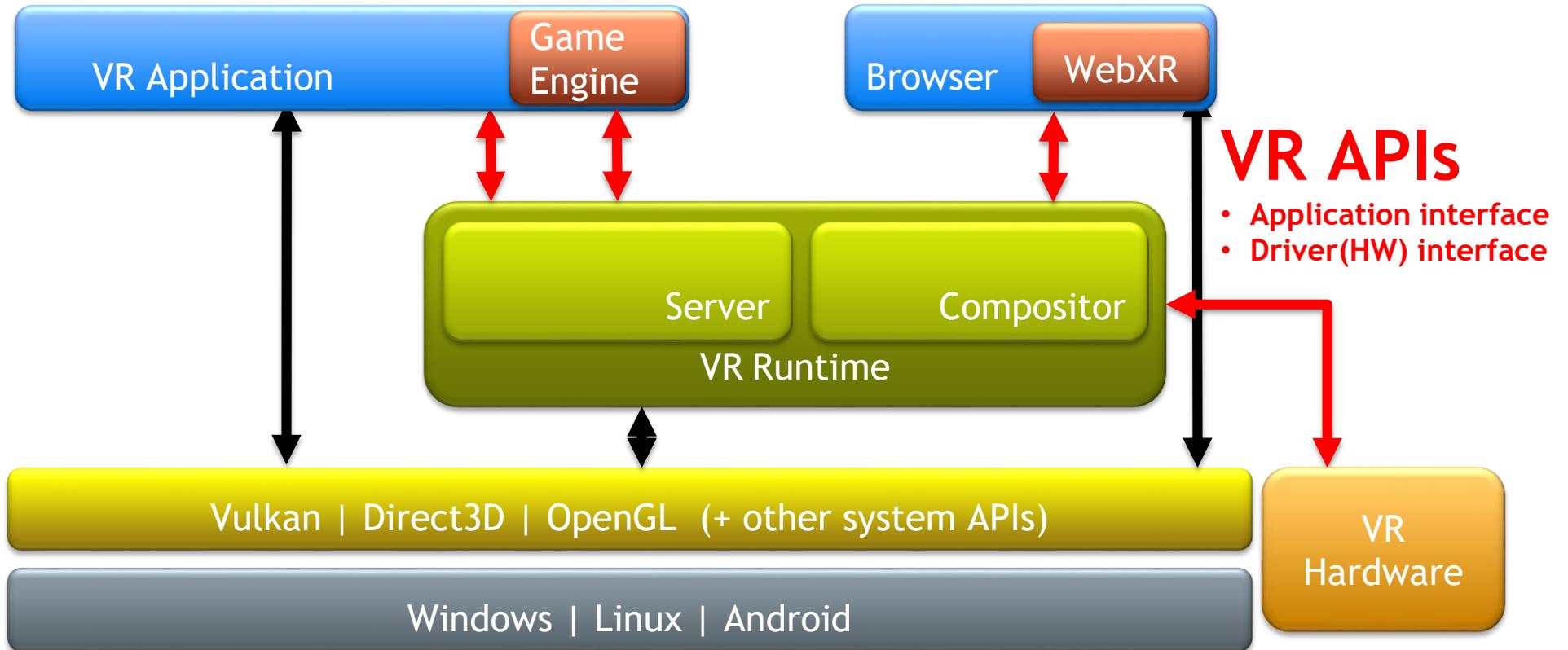


Before OpenXR
XR Market Fragmentation



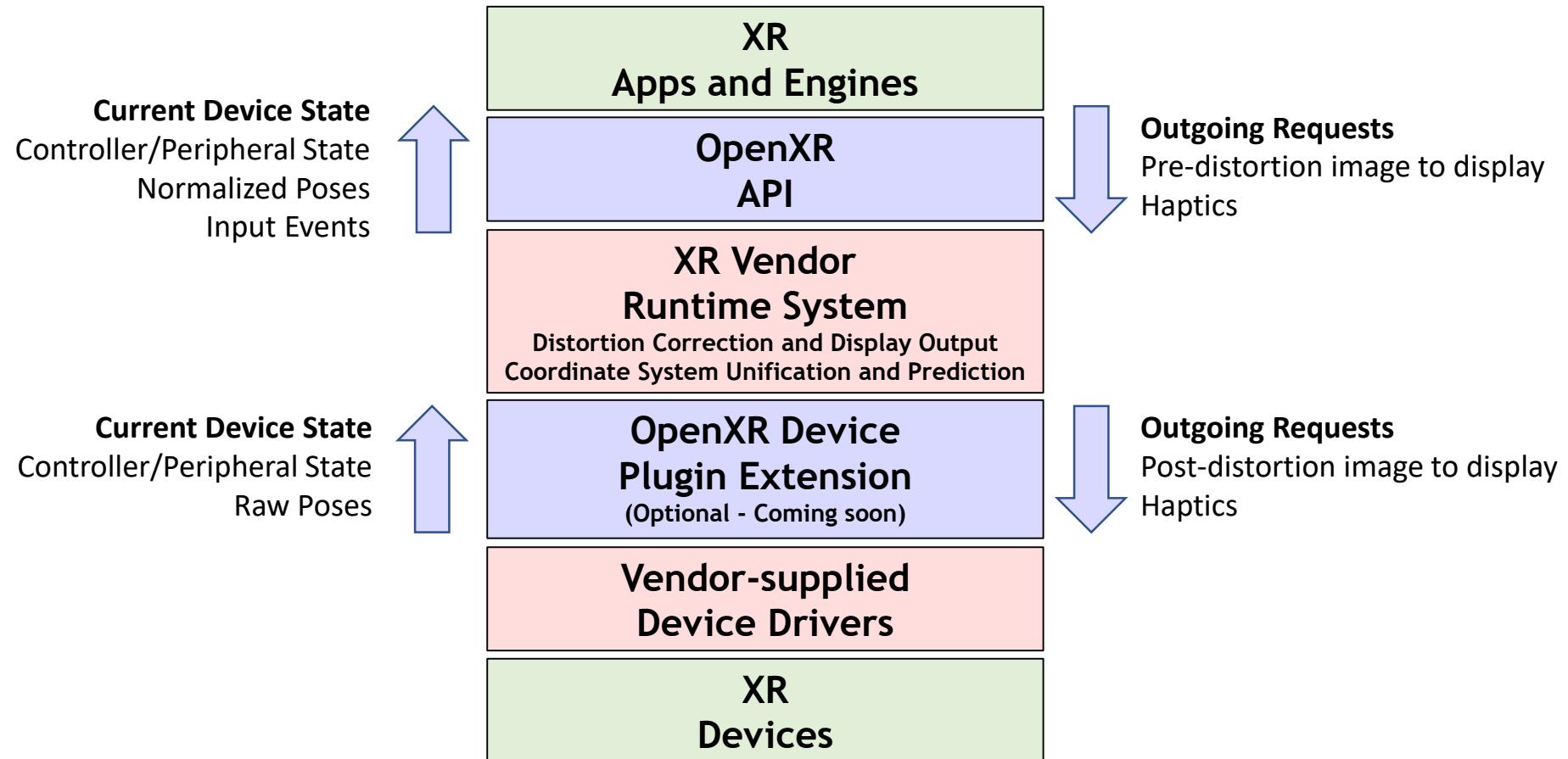
After OpenXR
Wide interoperability of XR apps and devices

VR Software Stack (Example)

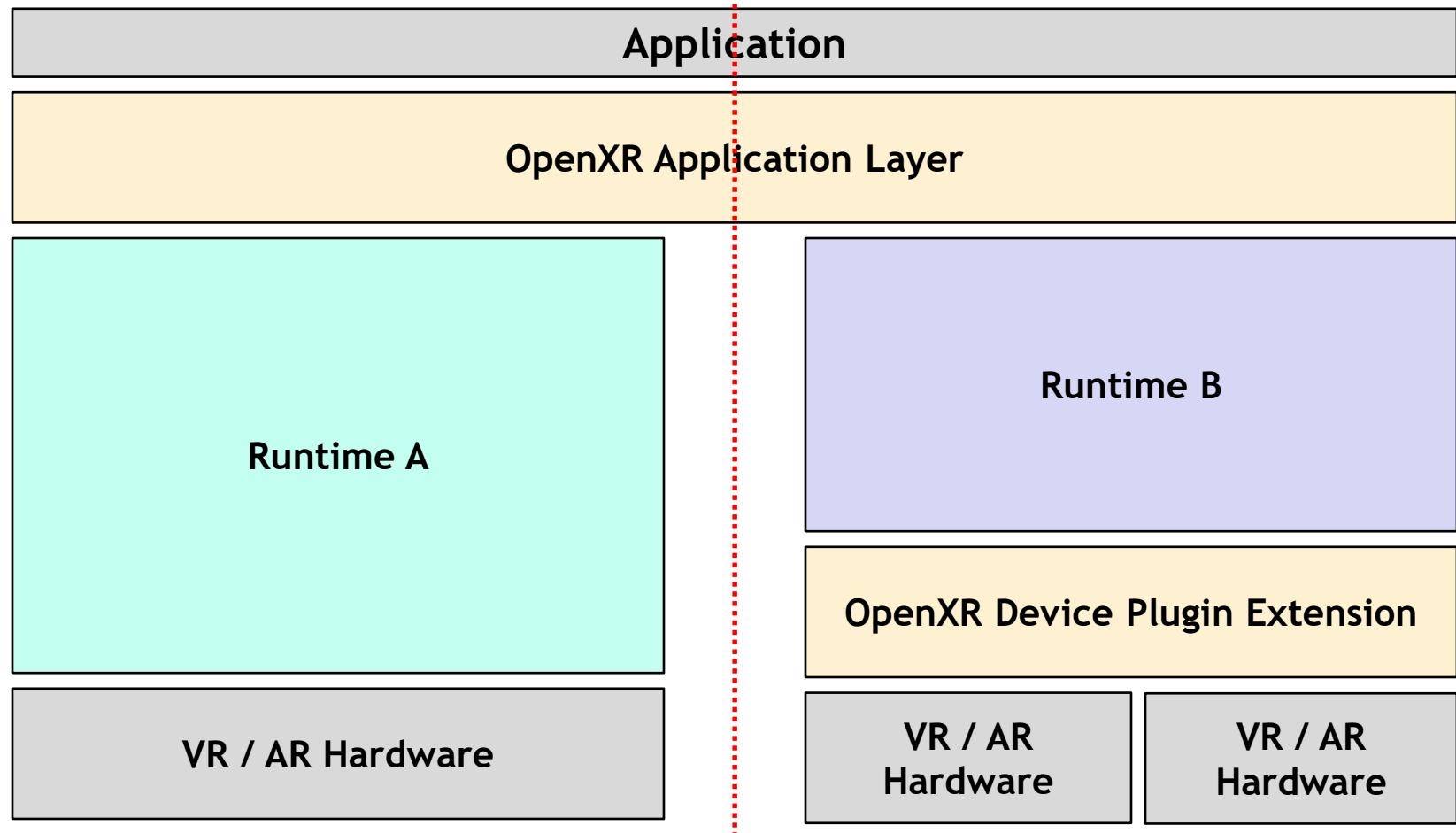


OpenXR Architecture

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



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

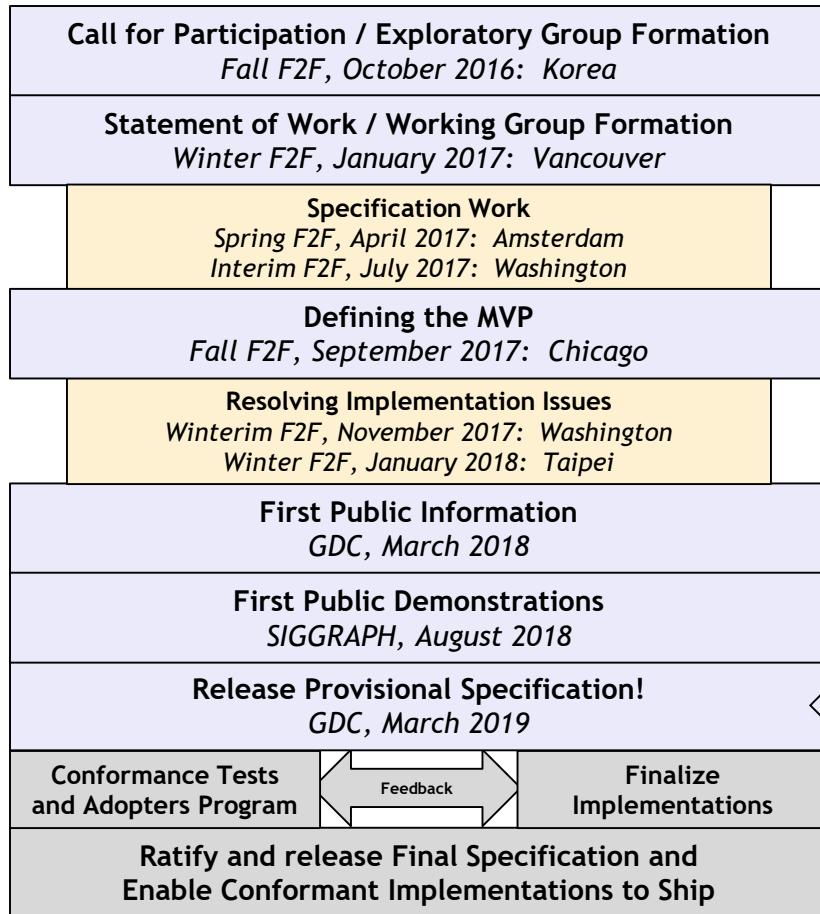
While trying to predict the future details of XR would be foolhardy, OpenXR uses forward-looking API design techniques to enable engineers to easily harness new and emerging technologies

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?

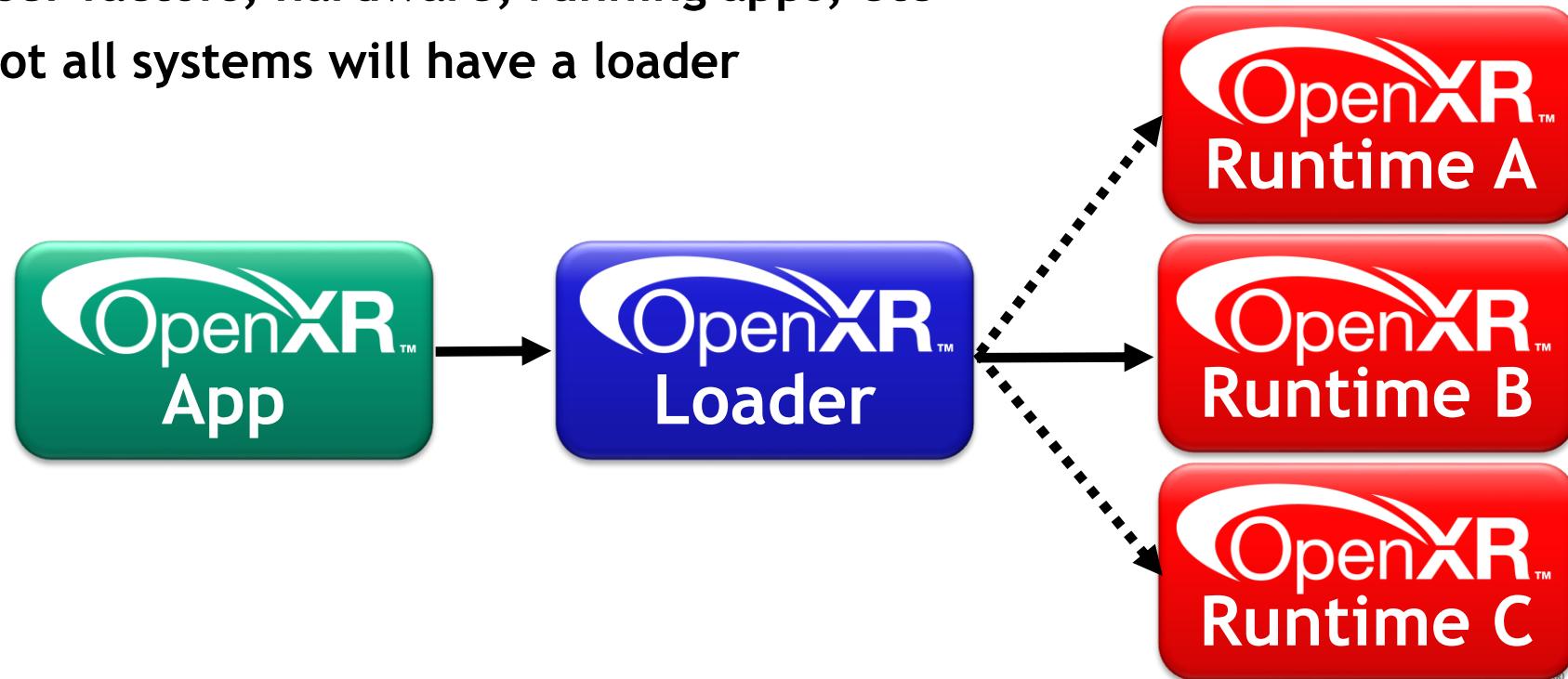


OpenXR 0.90 Provisional Specification Released
Enables industry review and feedback
First prototype implementations available

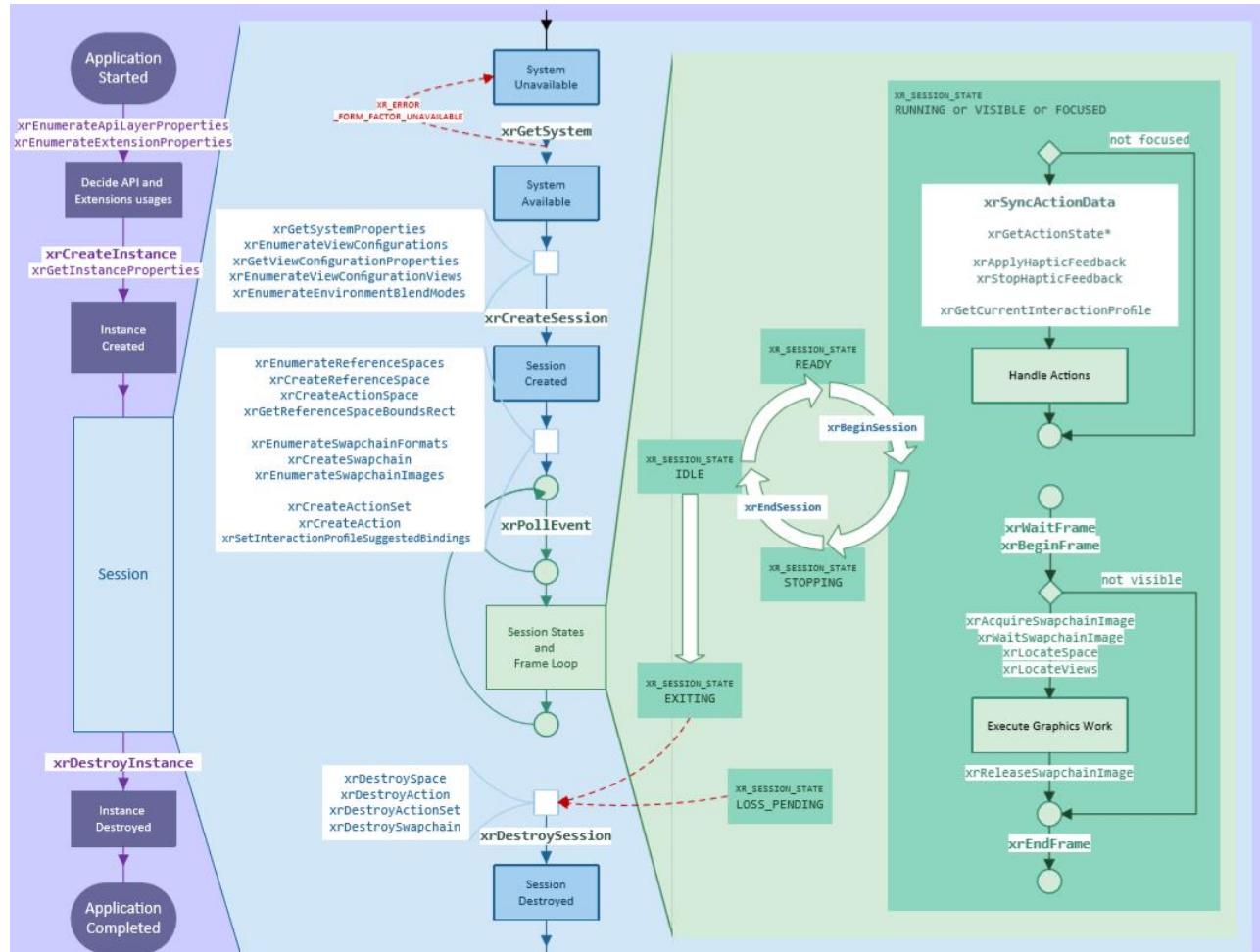
OpenXR Application Development Walkthrough

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



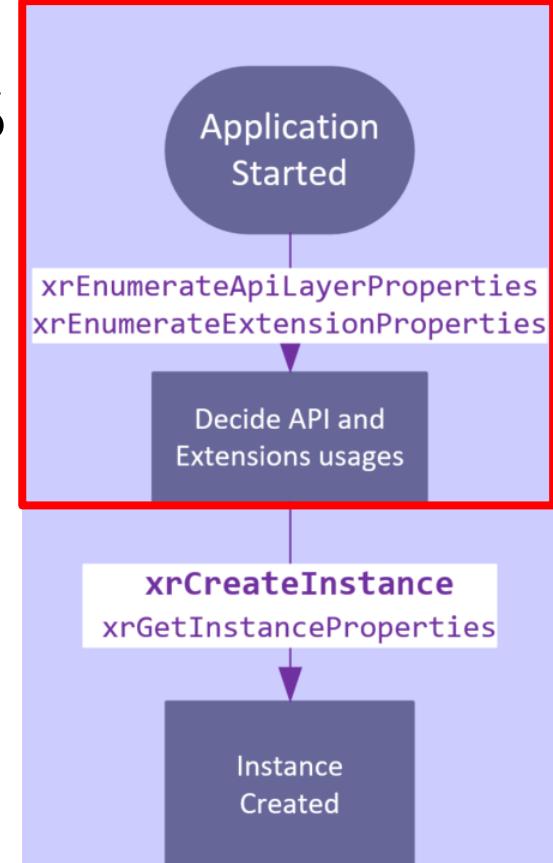
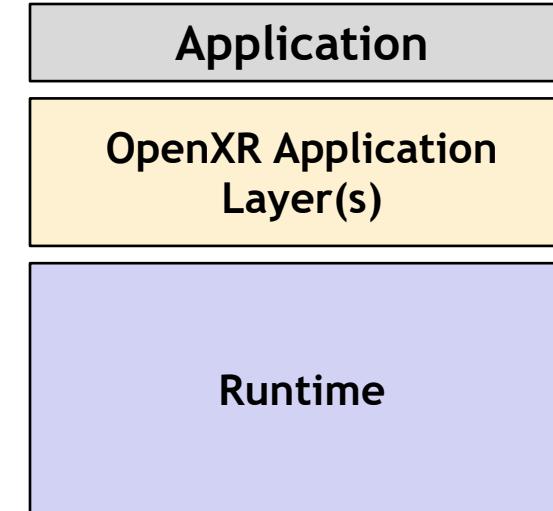
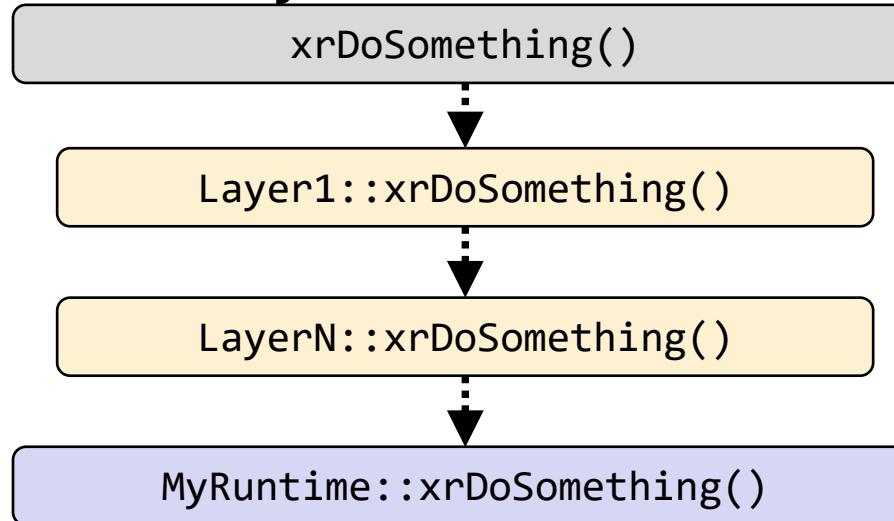
Time to use the Overview Guide (back)



Thanks to Yin Li, Microsoft

Initial steps: API Layers & Extensions

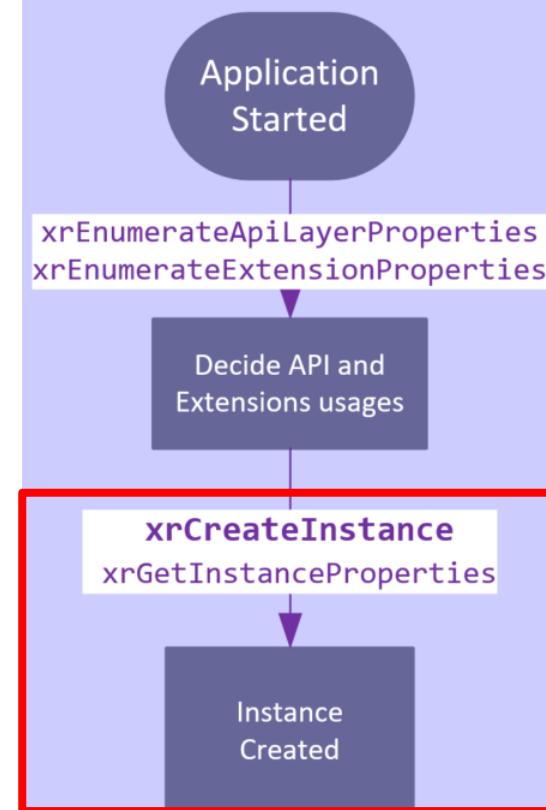
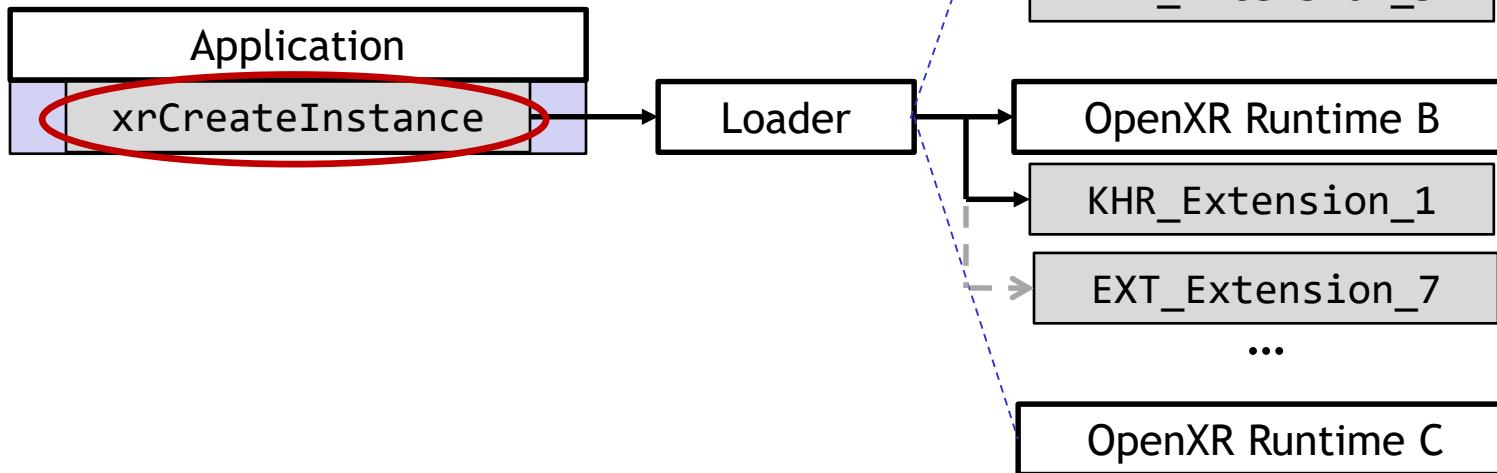
- **API Layers**



The Instance

XrInstance:

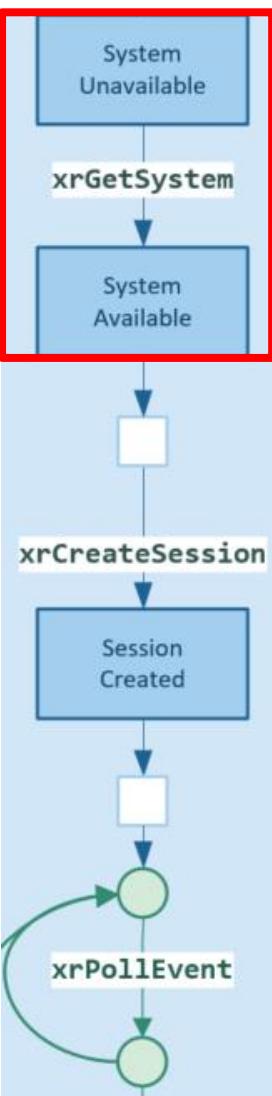
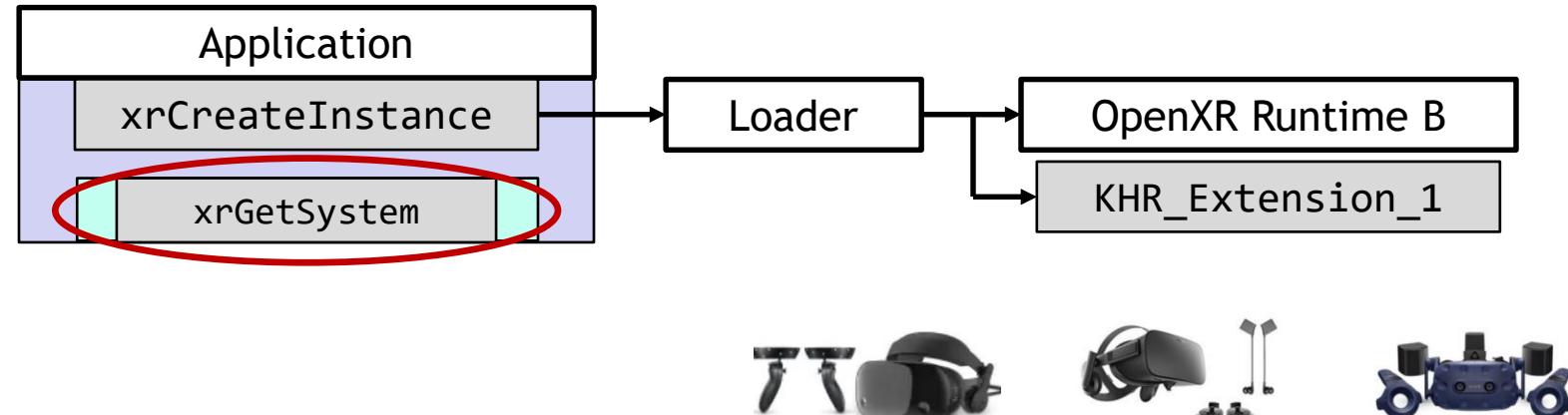
- The XrInstance is basically the application's representation of the OpenXR runtime
- Can create multiple XrInstances, if supported by the runtime
- xrCreateInstance specifies app info, layers, and extensions



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`

Camera Passthrough AR	Stereoscopic VR / AR	Projection CAVE-like
		 Photo Credit: Dave Pape
One View	Two View (one per eye)	Twelve Views (six per eye)
<code>XR_FORM_FACTOR_HANDHELD_DISPLAY</code>	<code>XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY</code>	(future support)

View Configurations

- Currently two `XrViewConfigurations` in the specification:
 - `XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO`
 - `XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO`

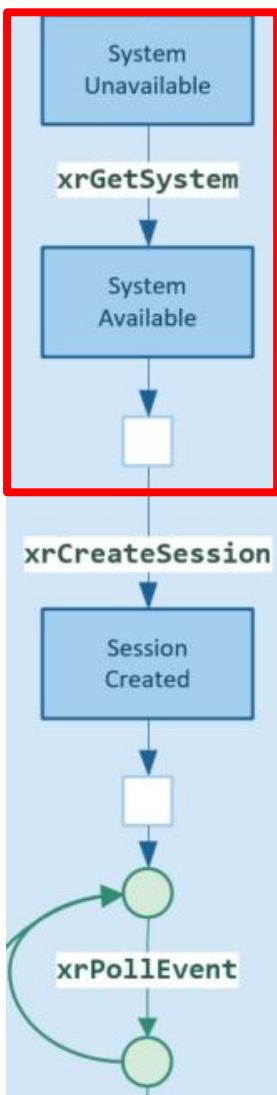


One View	Two View (one per eye)	Twelve Views (six per eye)
<code>XR_FORM_FACTOR_HANDHELD_DISPLAY</code>	<code>XR_FORM_FACTOR_HEAD_MOUNTED_DISPLAY</code>	(future support)
<code>XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO</code>	<code>XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO</code>	(future support)

View Configurations

Session:

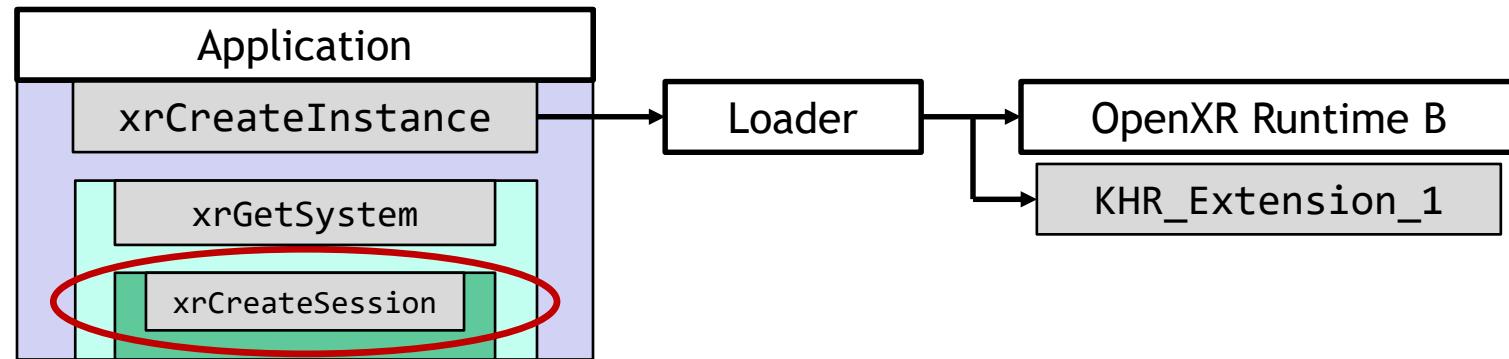
- After getting the system, must determine what view configuration(s) are supported
- `xrEnumerateViewConfigurations()`
- `xrGetViewConfigurationProperties()`
- **`xrEnumerateViewConfigurationViews()`**
 - Returns the recommended and max widths and heights of the views
- `xrEnumerateEnvironmentBlendModes()`



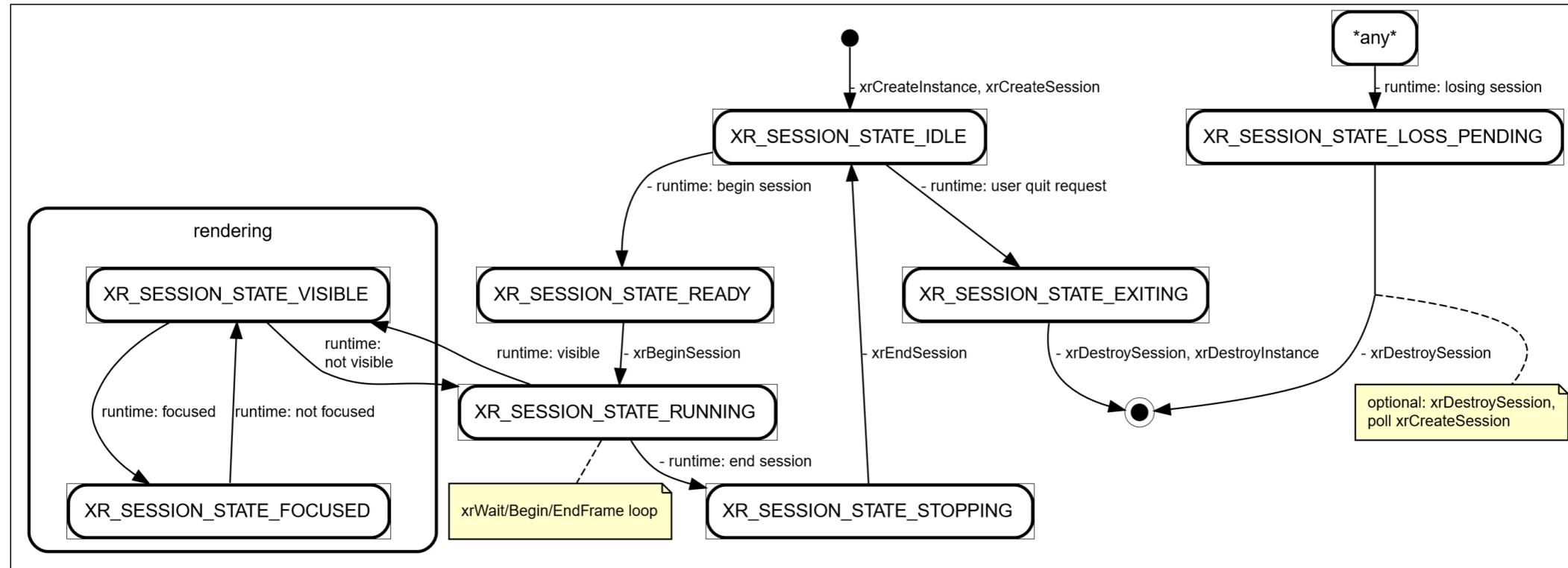
The Session

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.



Session Lifecycle

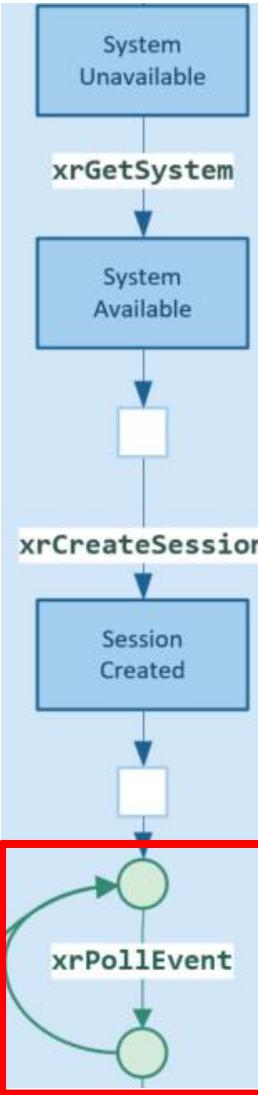


Spec page 118

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`

Event	Description
<code>XrEventDataEventsLost</code>	event queue has overflowed and some events were lost
<code>XrEventDataInstanceLossPending</code>	application is about to lose the instance
<code>XrEventDataInteractionProfileChanged</code>	active input form factor for one or more top level user paths has changed
<code>XrEventDataReferenceSpaceChangePending</code>	runtime will begin operating with updated space bounds
<code>XrEventDataSessionStateChanged</code>	application has changed lifecycle state



Back to the walkthrough

Frame Timing

Let's examine frame timing first in the simplest case of a single-threaded render loop

xrWaitFrame:

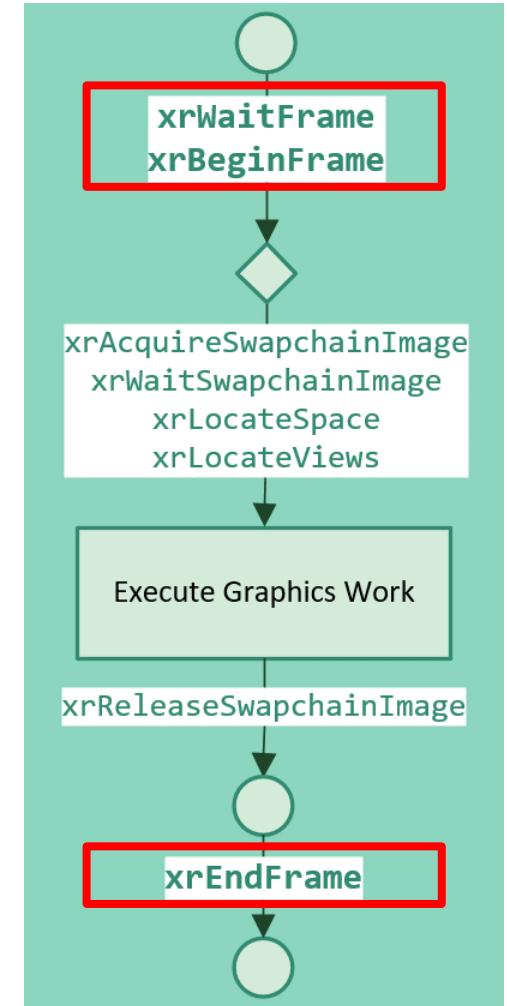
Called before we begin simulation of the next frame. This is responsible for throttling. Also returns when frame will be displayed.

xrBeginFrame:

Signals that we're ready to begin rendering pixels to the active image in our swap chain

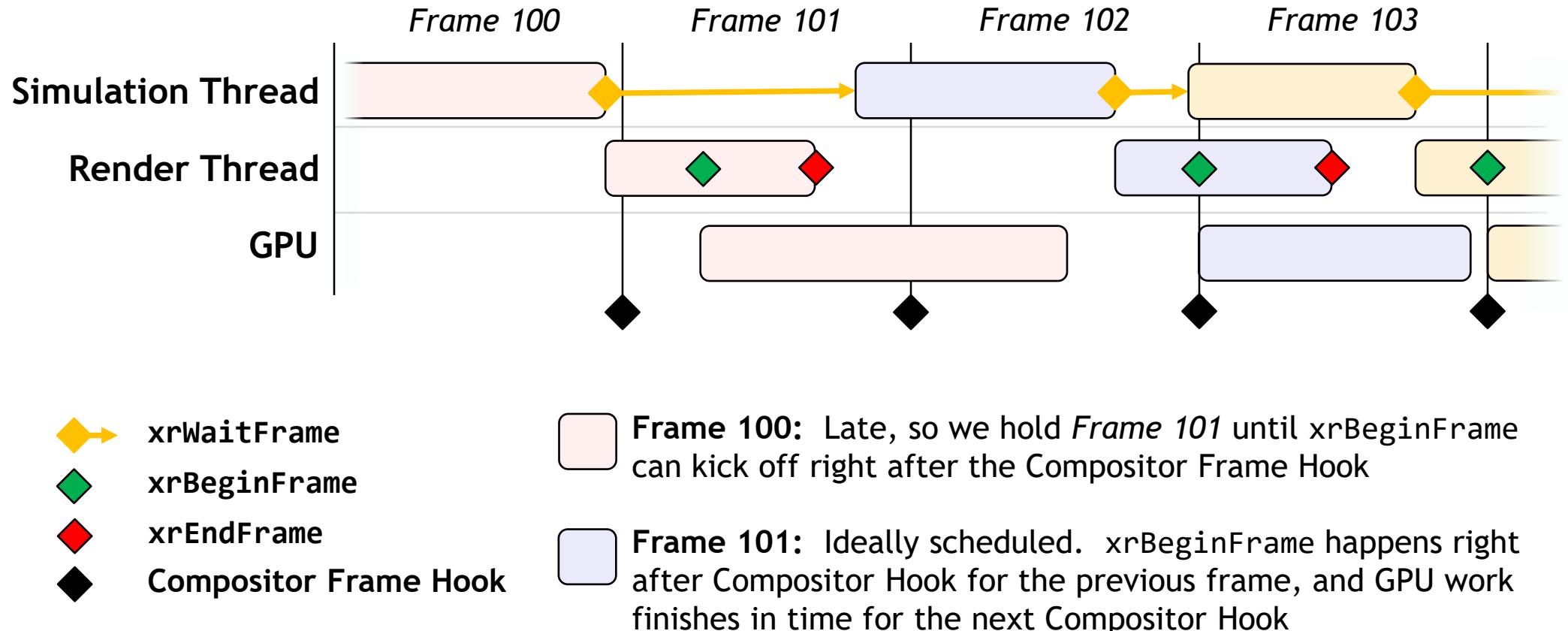
xrEndFrame:

We're finished rendering, and now are ready to hand off to the compositor for presentation. Takes the predicted display time, and layers to present



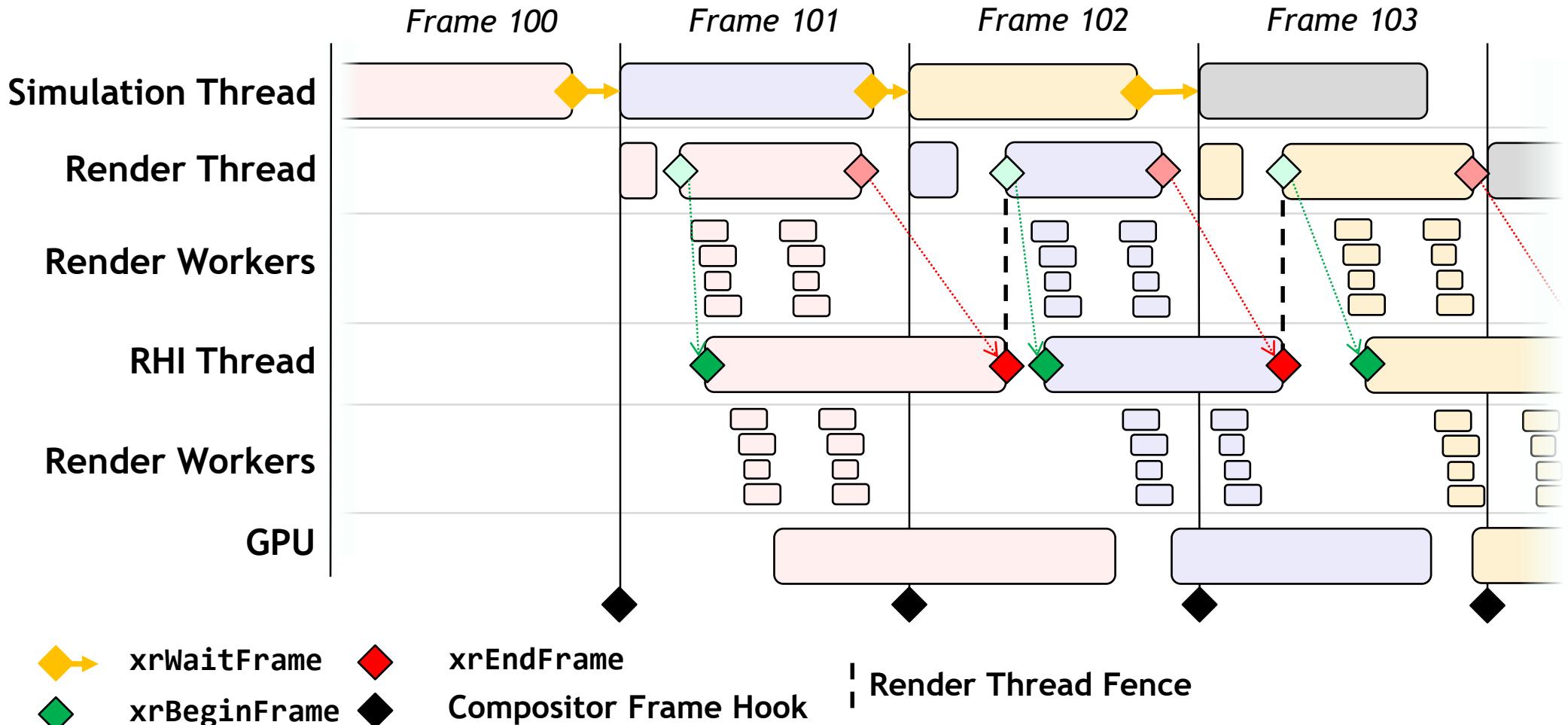
Frame Timing

Simple Multithreaded Example (DX11, OpenGL)

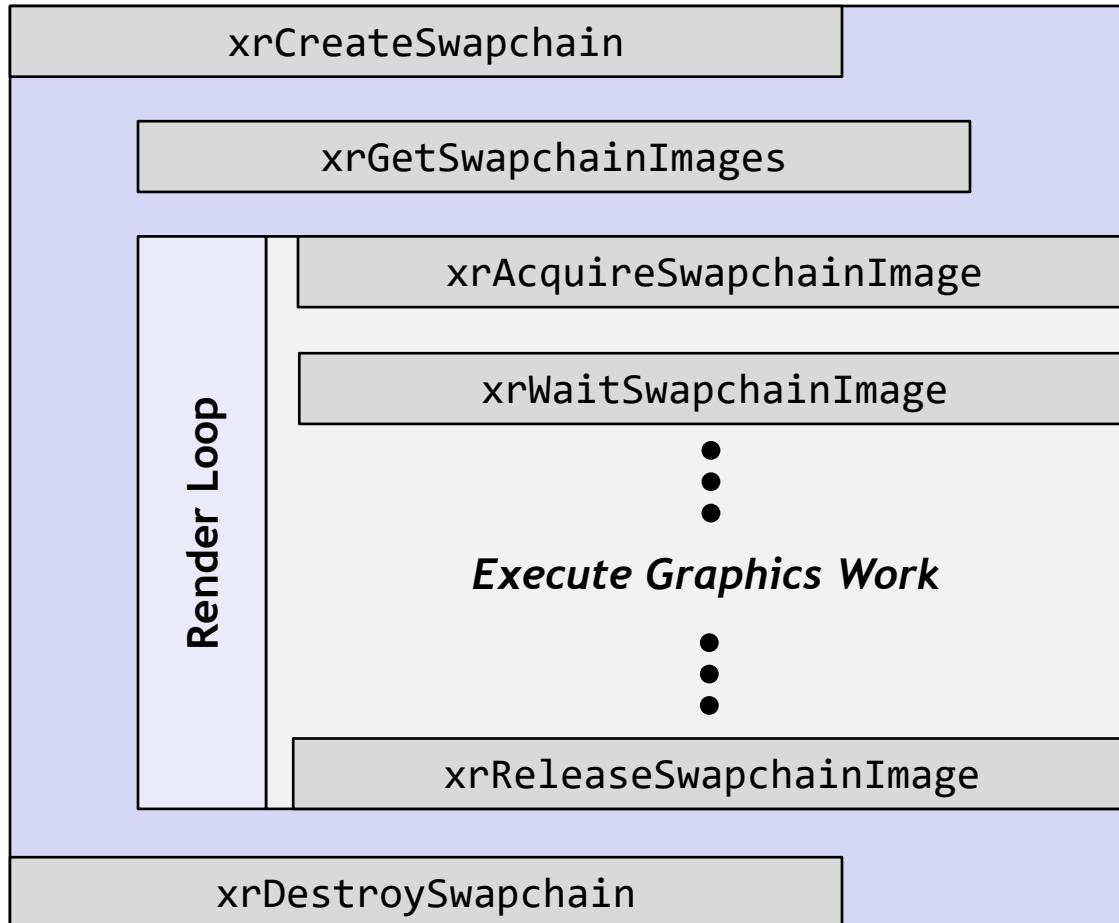


Frame Timing

Deeply Pipelined Multithreaded Example *(Unreal Engine 4 with Vulkan, DX12, Metal)*



Swap Chains and Rendering



XrSwapchains:

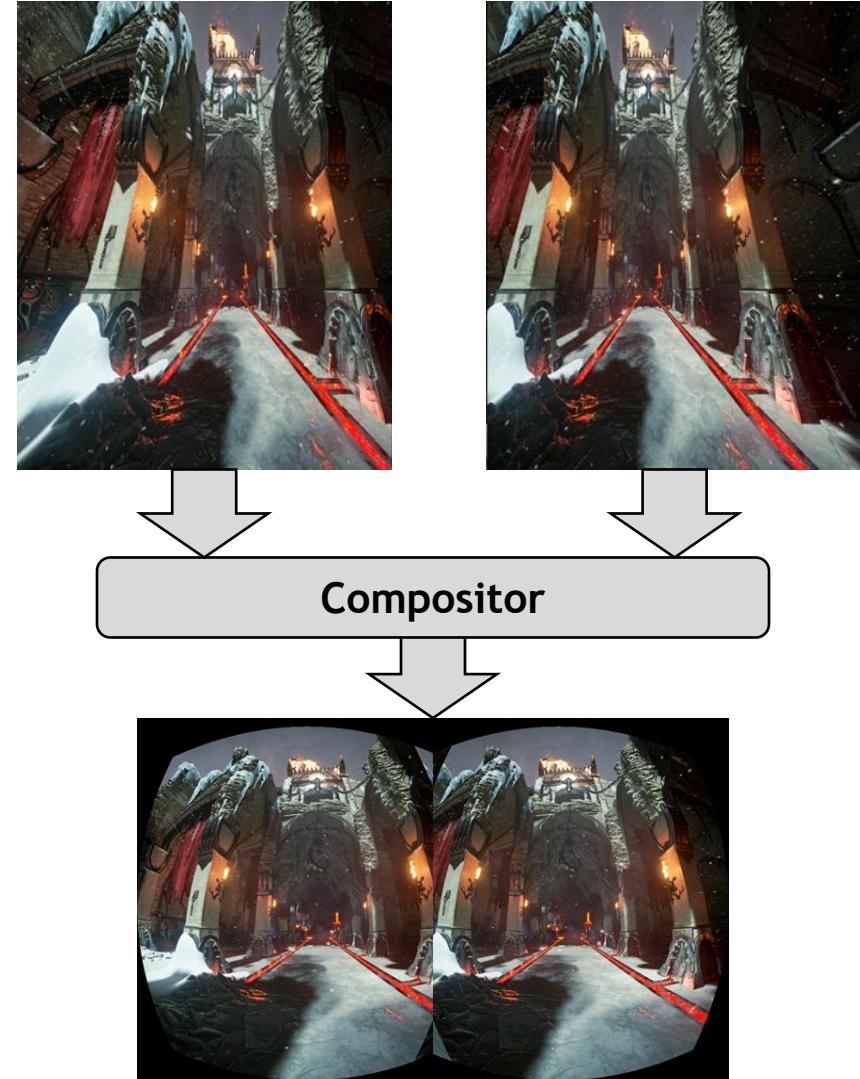
XrSwapchains are limited by the capabilities of the XrSystem that they are being created for, and can be customized on creation based on application needs

- Usage Flags
- Format
- Width
- Height
- Swap chain length

Compositor Layers

The Compositor is responsible for taking all the Layers, reprojecting and distorting them, and sending them to the display

- Layers are aggregated by the Compositor in `xrEndFrame` for display
- You can use multiple layers, up to the limit of the runtime

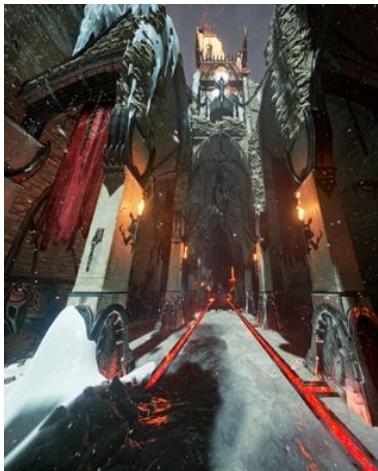


Compositor Layers

XR_TYPE_COMPOSITION_LAYER_PROJECTION:

Most common type of Layer. This is the classic “eye” layer, with each eye represented by a standard perspective projection matrix

XR_EYE_LEFT

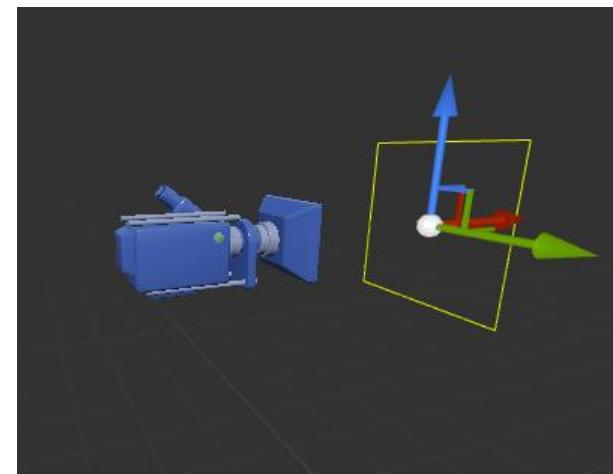


XR_EYE_RIGHT



XR_TYPE_COMPOSITION_LAYER_QUAD:

Quad layers are common for UI elements, or videos or images represented in the virtual world on a quad in virtual world space

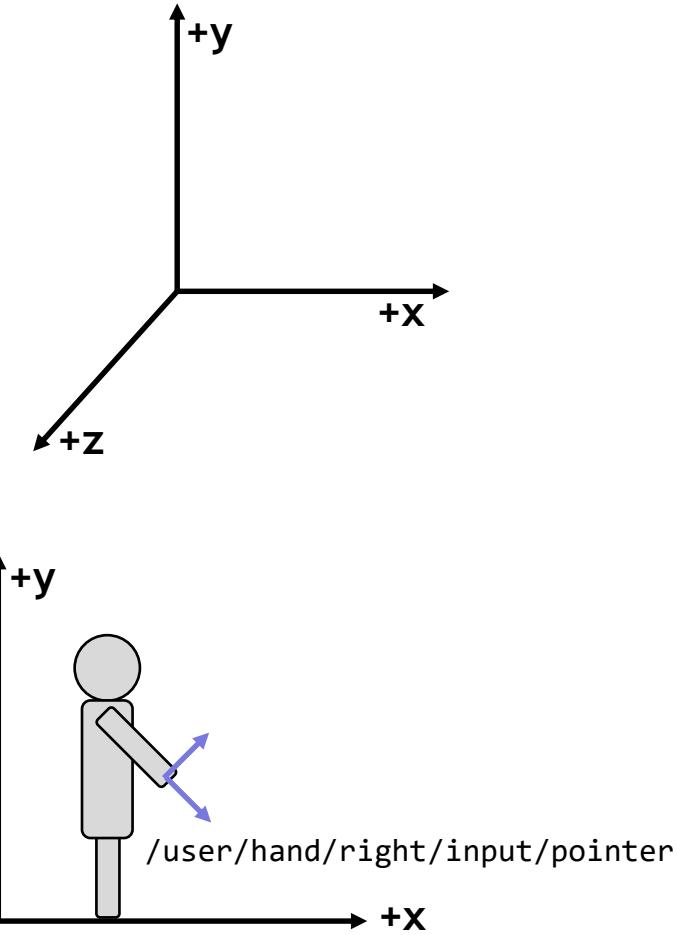


Spaces

XrSpace

XrSpace is one of the fundamental concepts used throughout the API to map virtual objects to real-world locations

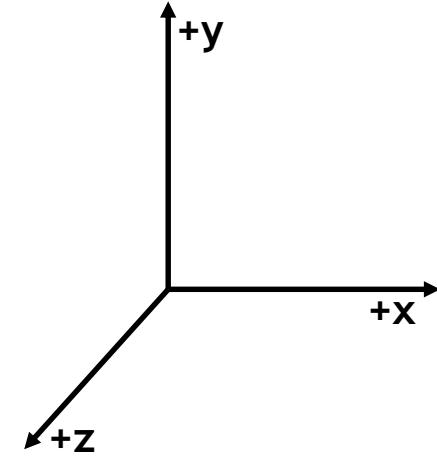
- XrSpaces define meaningful spaces within the environment, which can be related to one another, and used as a basis for functions that return spatial values
- The Runtime can hold any representation it wants internally, and can adjust them as new data is acquired
- When an app uses coordinates with OpenXR, it passes (or gets returned) the XrSpace the coordinates are in



Reference & Action Spaces

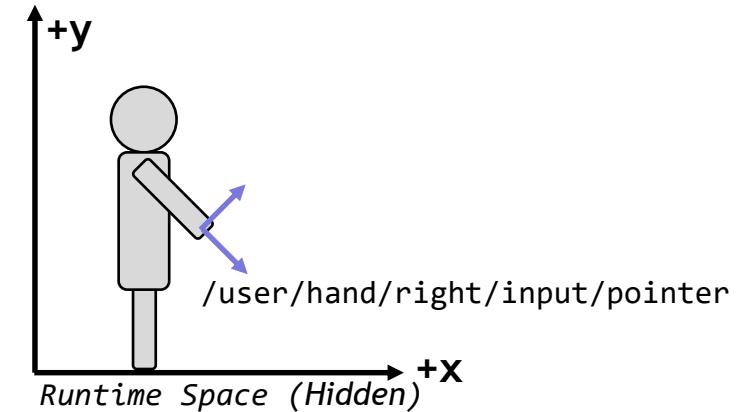
- Reference Spaces

- 3 reference spaces predefined by the API: VIEW, LOCAL, and STAGE
 - VIEW: Head-locked space, for view rays, reticle rendering, etc.
 - LOCAL: world-locked origin, gravity-aligned, +Y up, +X right, -Z forward (useful for seated experiences)
 - STAGE: runtime-defined rectangular space, origin at the floor in center of rectangle (“the playspace”)

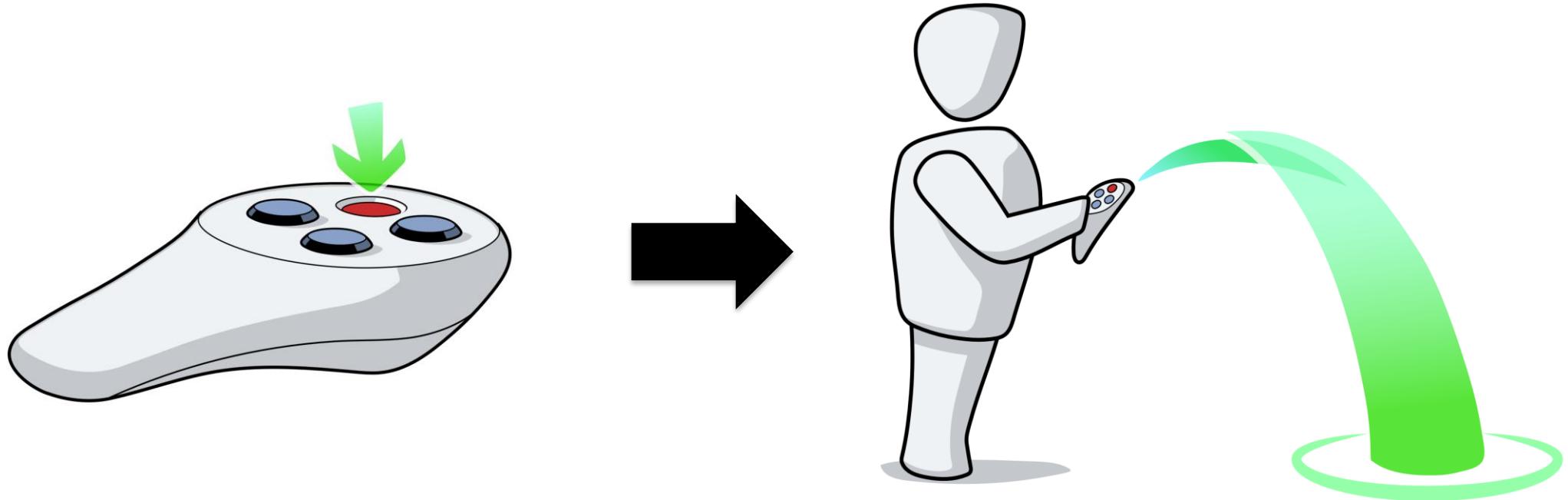


- Action Spaces

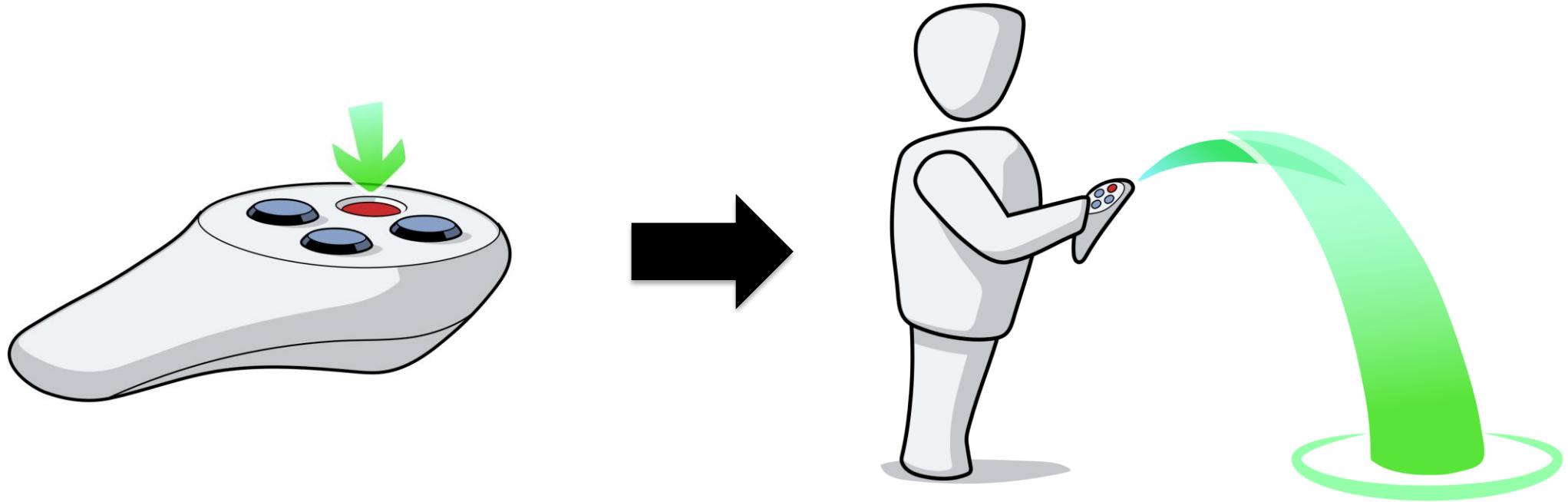
- Spaces created to track actions
 - for example tracking throwing motions



Input and Haptics



Input and Haptics

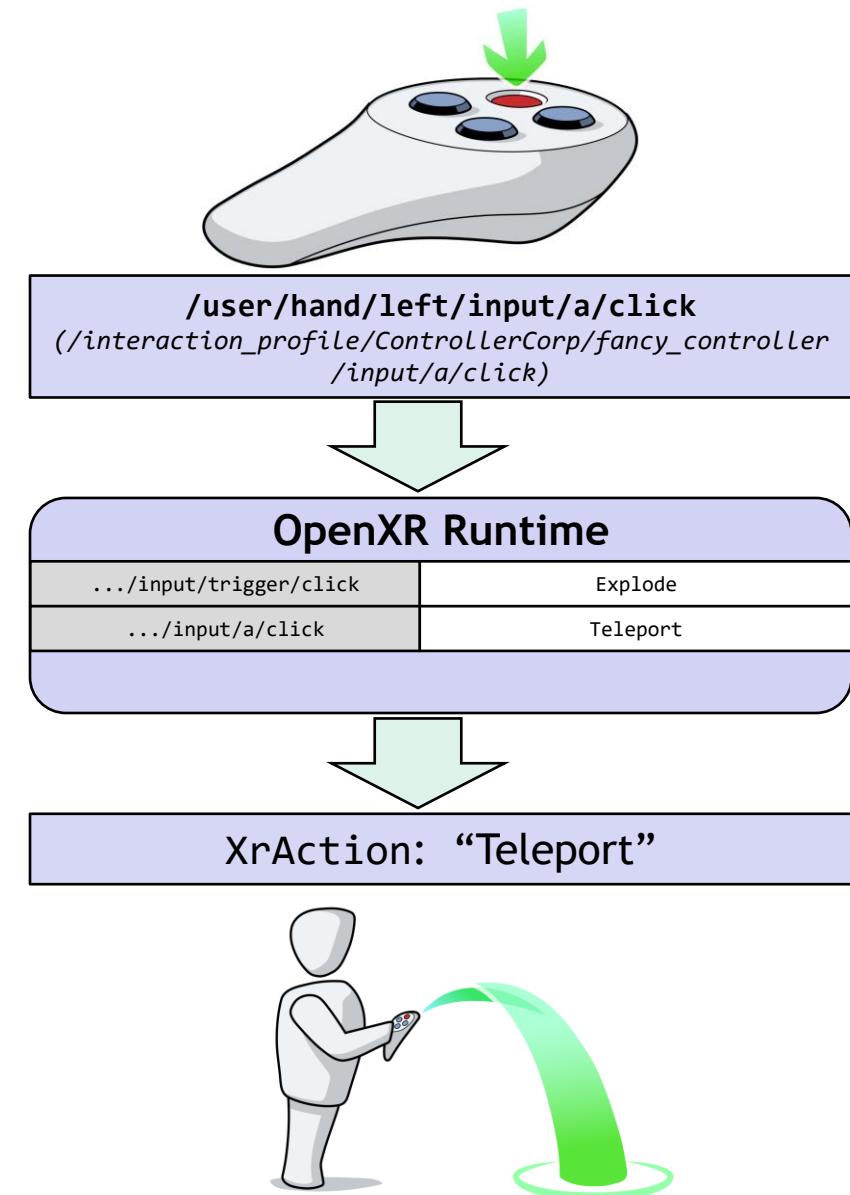


When user clicks button “a” it results in the user teleporting

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 - Simple Example

- Example Walkthrough:
 - Application has sets of inputs valid for different parts of the game, for example:
 - Game Actions:
 - Explode
 - Teleport
 - Menu Actions:
 - Select
 - Exit

Input and Haptics - Create Action Sets

Action Sets - collections of actions the application can quickly switch between during its lifetime

`xrCreateActionSet()`

Game Action Set

Menu Action Set

Input and Haptics - Create Actions

Actions are created and associated with an action set

`xrCreateAction()`

Game Action Set	Menu Action Set
Explode Action	Select Action
Teleport Action	Exit Action

In this example:

```
xrCreateAction( <Explode Action> , <Game Action Set>)
xrCreateAction(<Teleport Action> , <Game Action Set>)
xrCreateAction( <Select Action> , <Menu Action Set>)
xrCreateAction( <Exit Action> , <Menu Action Set>)
```

Input and Haptics - Bind Actions to Inputs

- Application suggests a set of bindings between actions and inputs
- **XrActionSuggestedBinding** structure

Action	Binding
Explode Action ←	?
Teleport Action ←	?

- How does the application know what bindings to use?

Enter Interaction Profiles...

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



example

Input and Haptics - Predefined Interaction Profiles

- 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 Knuckles* controllers

Input and Haptics - Bind Actions to Inputs

- Application suggests a set of bindings between actions and inputs
- **XrActionSuggestedBinding** structure

Action	Binding
Explode Action ←	?
Teleport Action ←	?

- How does the application know what bindings to use?

Input and Haptics - Bind Actions to Inputs

- Application suggests a set of bindings between actions and inputs
- **XrActionSuggestedBinding** structure

Action	Binding
Explode Action ←	?
Teleport Action ←	?

- How does the application know what bindings to use?
 - **The application is built around a particular interaction profile**

Input and Haptics - Bind Actions to Inputs

- Application suggests a set of bindings between actions and inputs
- `XrActionSuggestedBinding` structure

Action	Binding
Explode Action	Trigger Click
Teleport Action	Button A Click

- How does the application know what bindings to use?
 - The application chooses an interaction profile
 - In our example the application chooses the interaction profile of the fancy_controller

Input and Haptics - Suggested Bindings

- Application submits its suggested bindings along with the interaction profile it is using to the runtime
- `xrSetInteractionProfileSuggestedBinding()`

Action	Binding
Explode Action	Trigger Click
Teleport Action	Button A Click

Suggested Bindings

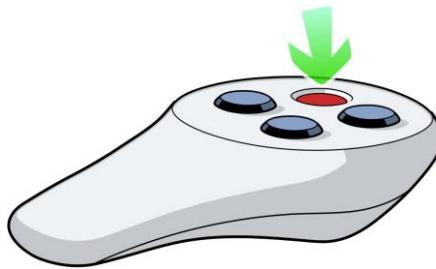
ControllerCorp
Fancy_Controller

Interaction Profile

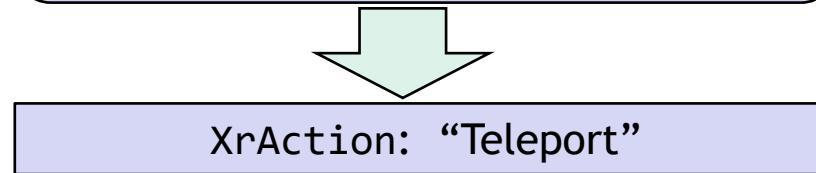
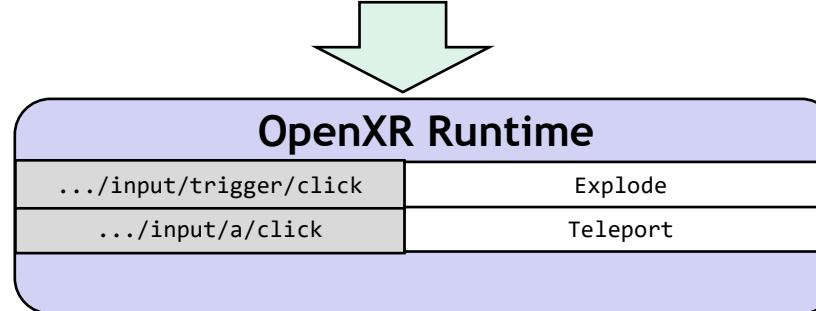
Input and Haptics - Runtime Binding Decision

- Runtime ultimately decides the bindings
 - “dev teams are ephemeral, 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)



Input and Haptics - Input Action Types

<code>XR_INPUT_ACTION_TYPE_BOOLEAN</code>	If path is a scalar value, a threshold must be applied. If not a value, needs to be bound to .../click
<code>XR_INPUT_ACTION_TYPE_VECTOR1F</code>	If path is a scalar value, then input is directly bound. If the bound value is boolean, the runtime must supply a 0.0 or 1.0 as the conversion
<code>XR_INPUT_ACTION_TYPE_VECTOR2F</code>	Path must refer to parent with child values .../x and .../y

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, Device Plugin, Headless, Tracking Bounds

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

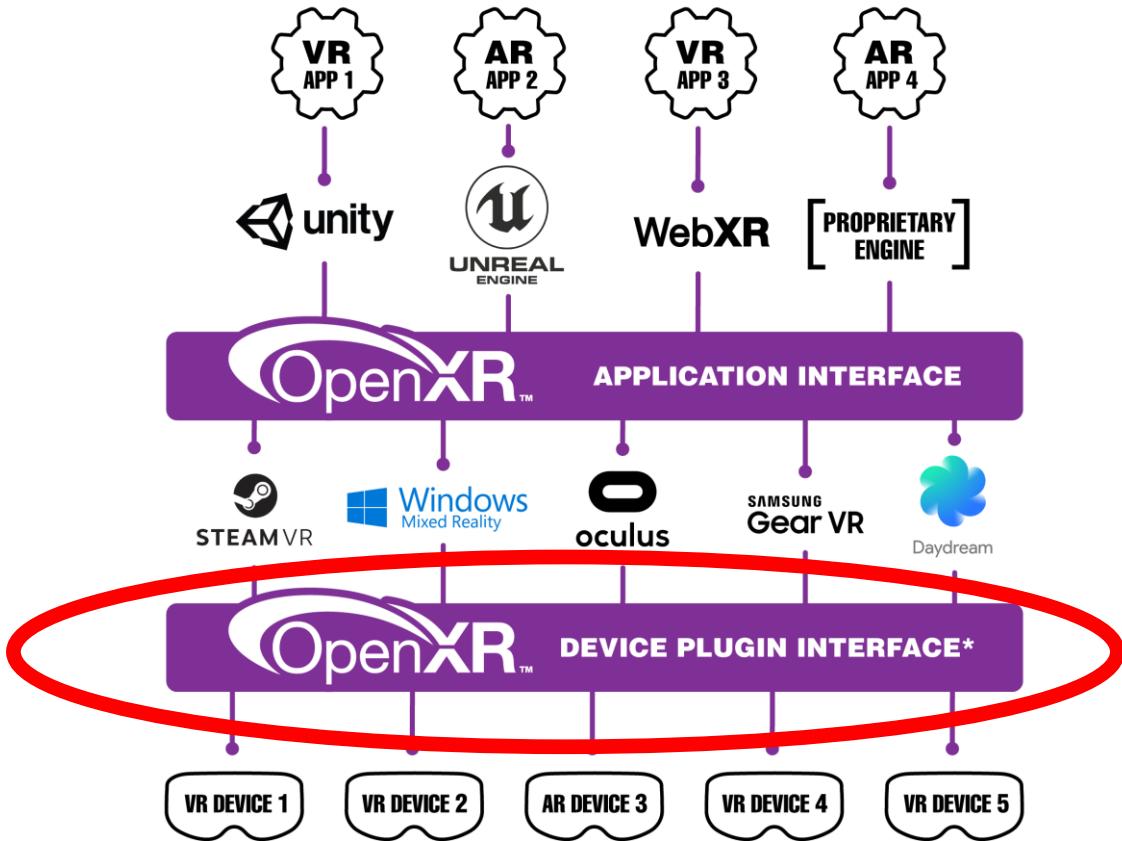
Current Provisional Extensions

- Platform-specific support:
 - KHR_android_create_instance
 - KHR_android_surface_swapchain
 - KHR_android_thread_settings
- Graphics API support:
 - KHR_D3D10_enable
 - 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_depth
 - KHR_composition_layer_equirect
- Performance improvement by masking non-visible portions of the display:
 - KHR_visibility_mask
- Non-display uses of OpenXR (for tracking or input-only use cases):
 - KHR_headless
- Time Conversion functions:
 - KHR_convert_timespec_time
 - KHR_win32_convert_performance_counter_time

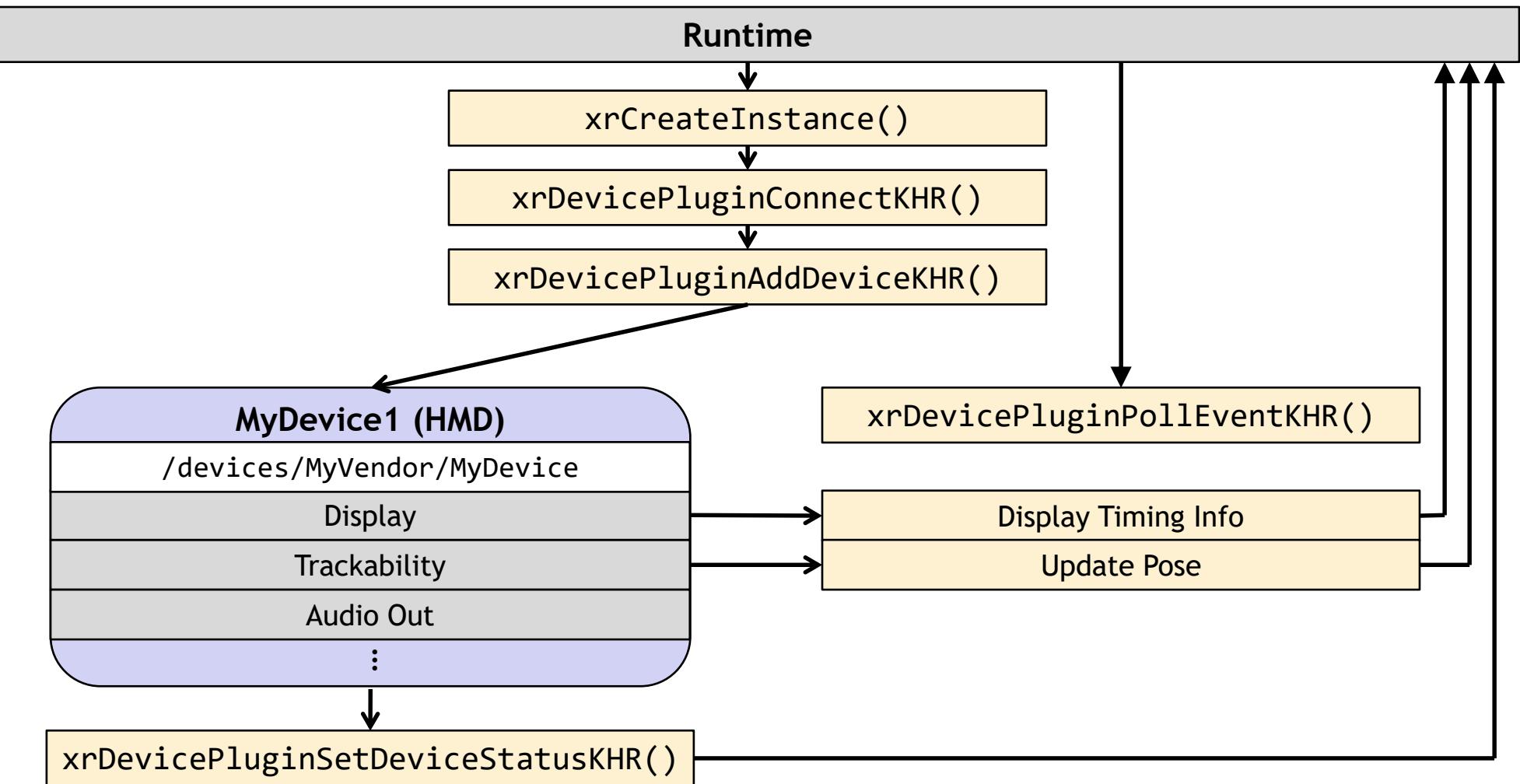
What hasn't made it in?

What hasn't made it in?

- Top priority: solve application fragmentation



Device Plugin Extension



There are a list of things to consider for after 1.0 or for 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 Provisional 0.90 Release is Here!

KHRONOS GROUP

UPLOAD NEWS REVIEWS GAMING

120.0k

GDC 2019: OpenXR Specification And API Released Publicly For AR And VR Devices

Khronos Group releases early OpenXR spec for AR and VR hardware standards

DEAN TAKAHASHI @DEANTAK

ROAD TO VR

Companies Pioneering the Future of VR

AMD antilliance

htc HUAWEI hp

magic leap MEDIATEK

Pico pluto

tobii unity VIVE

Above: Khronos Group has all the big names in VR.

NEWS

Khronos' OpenXR 0.90 Provisional Spec For VR & AR Devices Launched At GDC 2019

OpenXR aims to solve the issue of XR fragmentation.

By Peter Graham Last updated Mar 18, 2019 — 0 Comments

Game Developers Conference 2019 GDC 2019 OpenXR The Khronos Group

Ever since the Khronos Group announced back in 2016 plans to have an open standard for virtual reality (VR) and augmented reality (AR) devices called OpenXR, the initiative has garnered more and more support. With the first public demos showcased at SIGGRAPH last year, for the start of the Game Developer Conference (GDC) 2019 the group has publicly launched the OpenXR 0.90 specification.

OpenXR Standard Ratified, Microsoft, Oculus, & Others to Release First Implementations

By Ben Lang · Mar 18, 2019

Image courtesy Khronos Group

- <https://www.roadtovr.com/openxr-0-9-provisional-release-microsoft-oculus-collabora-implementations/>
- <https://uploadvr.com/openxr-provisional-release/>
- https://www.phoronix.com/scan.php?page=news_item&px=Khronos-OpenXR-0.90
- <https://venturebeat.com/2019/03/18/khronos-group-releases-early-openxr-spec-for-ar-and-vr-hardware-standards/>
- <https://www.vrfocus.com/2019/03/khronos-openxr-0-90-provisional-spec-for-vr-ar-devices-launched-at-gdc-2019/>
- <https://bit-tech.net/news/tech/software/khronos-group-launches-openxr-090/1/>

phoronix

ARTICLES & REVIEWS NEWS ARCHIVE FORUMS PREMIUM CATEGORIES

OpenXR 0.90 Released For AR/VR Standard - Monado Is An Open-Source Implementation

Written by Michael Larabel in Standards on 18 March 2019 at 09:00 AM EDT. 1 Comment

Last year we were expecting The Khronos Group to introduce OpenXR 1.0 for this standard to address fragmentation and provide interoperability in the VR space followed by AR. That debut last year didn't happen although they did show off the first demonstration at SIGGRAPH. This week though at GDC they are announcing the OpenXR 0.90 provisional specification release.

The OpenXR 0.90 provisional specification is now available today. Yes, v0.90 and not 1.0... This caught me by surprise too when being briefed last week. This provisional specification ended up being released before the official standard was due. They are hoping for the AR support

Memory Mobile Monitors Motherboards Networking PCs Peripherals | ▶

OpenXR

OpenXR Application Interface

Unity Lumberyard WebXR Proprietary Engine

Windows oculus Gear VR Daydream

OpenXR Device Plugin Interface

VR Device 1 VR Device 2 VR Device 3 VR Device 4 VR Device 5

In the OpenGL and Vulkan graphics application layer, the Khronos Group has announced the public release of OpenXR 0.90, its specification for high-performance augmented and virtual reality (AR and VR, combined as

known as the Khronos VR initiative, OpenXR was officially unveiled back in March 2017 with members including AMD, Intel, Arm, and Nvidia alongside game engine makers Epic Games and Unity. The freshly-founded working group aimed at solving the fragmentation problem in the virtual and augmented reality field, not simply by adding yet another competing standard to the pile but by promising to provide an open application interface which would allow any game engine to talk to any VR or AR platform and vice-versa.

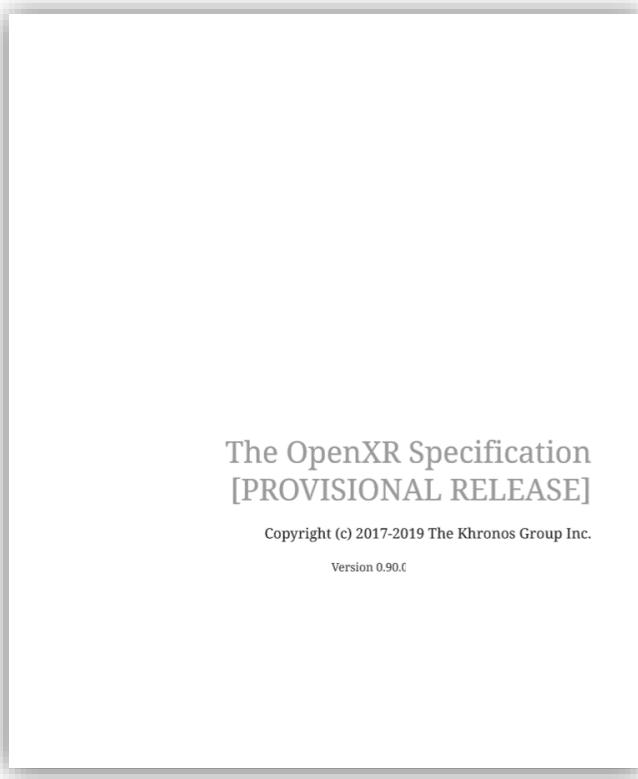
Today's release of the OpenXR 0.90 provisional specification, announced during the Game Developers Conference, is in line with recent changes to data protection legislation in the UK and Europe we would like to direct you to our updated Privacy Policy here.

What Resources Are Available?

What Resources Are Available?

- Fair warning:
 - This is a provisional release
 - This is our working group's first ever public release
 - Much of the effort and time leading to the provisional release was completing the release, the supporting software and infrastructure is lagging
 - Some things may be missing or incomplete
 - Some things may land later than Day 1
- Please be patient as we work through any launch issues, but do point out things that appear wrong or missing

What Resources Are Available?



200+ page specification

What Resources Are Available?

The OpenXR Specification [PROVISIONAL RELEASE]

Copyright (c) 2017-2019 The Khronos Group Inc.

Version 0.90.c

200+ page specification

xrCreateInstance(3) Manual Page

NAME

xrCreateInstance - Creates an OpenXR Instance

C Specification

The `xrCreateInstance` function is defined as:

```
XrResult xrCreateInstance(
    const XrInstanceCreateInfo* createInfo,
    XrInstance* instance);
```

Parameters

Parameter Descriptions

- `createInfo` points to an instance of `XrInstanceCreateInfo` controlling creation of the instance.
- `instance` points to an `XrInstance` handle in which the resulting instance is returned.

Description

`xrCreateInstance` creates the `XrInstance`, then enables and initializes global API layers and extensions requested by the application. If an extension is provided by an API layer, both the API layer and extension must be specified at `xrCreateInstance` time. If a specified API layer cannot be found, no `XrInstance` will be created and the function will return `XR_ERROR_API_LAYER_NOT_PRESENT`. Likewise, if a specified extension cannot be found the call will return `XR_ERROR_EXTENSION_NOT_PRESENT` and no `XrInstance` will be created. Additionally, some runtimes may limit the number of concurrent instances that may be in use. If the application attempts to create more instances than a runtime can simultaneously support, `xrCreateInstance` will return `XR_ERROR_LIMIT_REACHED`.

If the `XrInstanceCreateInfo` struct contains a platform-specific extension for a platform other than the target platform, `XR_ERROR_INITIALIZATION_FAILED` will be returned. The same is true if a mandatory platform-specific extension is defined for the target platform but no matching extension struct is provided in `XrInstanceCreateInfo`.

Valid Usage (Implicit)

- `createInfo` must be a pointer to a valid `XrInstanceCreateInfo` structure
- `instance` must be a pointer to an `XrInstance` handle

Reference Pages

What Resources Are Available?

The OpenXR Specification [PROVISIONAL RELEASE]

Copyright (c) 2017-2019 The Khronos Group Inc.

Version 0.90.c

200+ page specification

xrCreateInstance(3) Manual Page

NAME

xrCreateInstance - Creates an OpenXR Instance

C Specification

The `xrCreateInstance` function is defined as:

```
XrResult xrCreateInstance(
    const XrInstanceCreateInfo* createInfo,
    XrInstance* instance);
```

Parameters

Parameter Descriptions

- `createInfo` points to an instance of `XrInstanceCreateInfo` controlling creation of the instance.
- `instance` points to an `XrInstance` handle in which the resulting instance is returned.

Description

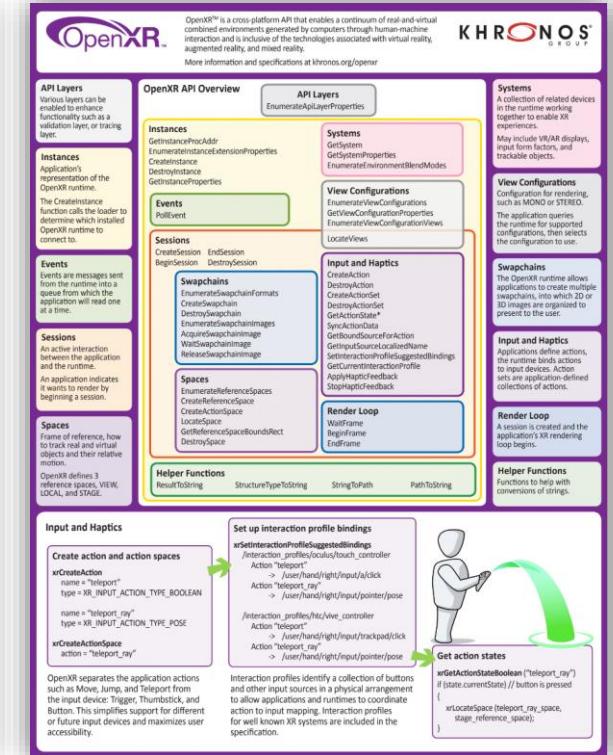
`xrCreateInstance` creates the `XrInstance`, then enables and initializes global API layers and extensions requested by the application. If an extension is provided by an API layer, both the API layer and extension must be specified at `xrCreateInstance` time. If a specified API layer cannot be found, no `XrInstance` will be created and the function will return `XR_ERROR_API_LAYER_NOT_PRESENT`. Likewise, if a specified extension cannot be found the call will return `XR_ERROR_EXTENSION_NOT_PRESENT` and no `XrInstance` will be created. Additionally, some runtimes may limit the number of concurrent instances that may be in use. If the application attempts to create more instances than a runtime can simultaneously support, `xrCreateInstance` will return `XR_ERROR_LIMITE_REACHED`.

If the `XrInstanceCreateInfo` struct contains a platform-specific extension for a platform other than the target platform, `XR_ERROR_INITIALIZATION_FAILED` will be returned. The same is true if a mandatory platform-specific extension is defined for the target platform but no matching extension struct is provided in `XrInstanceCreateInfo`.

Valid Usage (Implicit)

- `createInfo` must be a pointer to a valid `XrInstanceCreateInfo` structure
- `instance` must be a pointer to an `XrInstance` handle

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

OpenXR® API Documentation Project

[NOTE: This is the initial set up for the provisional release of the specification. Not all the files are populated yet, and expect significant changes as the spec moves towards 1.0.]

This repository contains sources for the formal documentation of the OpenXR API. This includes:

- the OpenXR API Specification
- OpenXR header files
- related tools and scripts.

The authoritative public repository is located at <https://github.com/KhronosGroup/OpenXR-Docs/>. It hosts public Issue tracker, and accepts patches (Pull Requests) from the general public.

Directory Structure

The directory structure is as follows:

README.adoc	This file
COPYING.md	Copyright and licensing information
CODE_OF_CONDUCT.md	Code of Conduct
specification/	Specification - files to generate the spec
include/openxr/	OpenXR headers, generated from the Registry

What Resources Are Available?

- <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 mostly found in the separate <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/>. Commits to the master branch of OpenXR-Registry will be reflected in the web view.

Interesting files in this repository include:

- index.php - toplevel index page for the web view. This relies on PHP include files found elsewhere on www.khronos.org and so is not very useful in isolation.
- specs/0.90/ - OpenXR 0.90 Provisional API specifications and reference pages and API reference card.

What Resources Are Available?

- <https://github.com/KhronosGroup/OpenXR-SDK>

- Contains the source for:
 - Loader
 - Some basic API layers
 - Test sample
- For the current best example code, see:
`src/tests/hello_xr`

OpenXR ® Software Development Kit (SDK) Project

[NOTE: This repository contains components that may eventually be assembled into an SDK, but are currently not being packaged into a distributable SDK.]

This repository contains source code and build scripts for implementations of the OpenXR loader, validation layers, and code samples.

The authoritative public repository is located at <https://github.com/KhronosGroup/OpenXR-SDK/>. It hosts public Issue tracker, and accepts patches (Pull Requests) from the general public.

Directory Structure

BUILDING.md	Instructions for building the projects
README.md	This file
COPYING.md	Copyright and licensing information
CODE_OF_CONDUCT.md	Code of Conduct
external/	External code for projects in the repo
include/	OpenXR platform include file
specification/	xr.xml file
src/	Source code for various projects
src/api_layer	Sample code for developing API layers
src/loader	OpenXR loader code
src/tests/	various test code (if looking for sample code start with hello_xr/)

Currently the best sample code is in src/tests/hello_xr/. More will be added in the future.

This structure is for the provisional specification. Things are incomplete at launch but will be added to going forward.

Additional Resources

- OpenXR Landing Page - Specification, Reference Pages, Sample Code, Overview
 - <https://www.khronos.org/openxr>
- OpenXR Forum and Slack Channel
 - Forum: <https://khr.io/openxrfeedback>
 - Discussion: <https://khr.io/slack>
- Detailed specification overview and SIGGRAPH session videos
 - <https://www.khronos.org/developers/library/2018-siggraph>
- Vendor prototype runtime implementations
 - Collabora: open source implementation
<http://monado.dev>
 - Microsoft: OpenXR runtime for Windows Mixed Reality headsets
<https://aka.ms/openxr>
- Khronos GDC Sessions - including OpenXR Presentation and demos
 - <https://www.khronos.org/events/2019-gdc>

Engine and Platform Support

Vinay Narayan, vice president, platform strategy, HTC

"HTC VIVE is committed to creating a viable ecosystem for the XR industry which is why we are proud to support OpenXR. Bringing the community together to help define standards and best practices, allows all of us to move forward, together."



Tim Sweeney, founder and CEO of Epic Games

"Epic believes that open standards like OpenXR are essential foundations for a vibrant, multi-platform VR and AR industry in the coming years. Epic plans to continue supporting OpenXR in Unreal Engine 4."



Nate Mitchell, Oculus Co-founder and head of VR product, Facebook

"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 later this year."



Alex Kipman, technical fellow, Microsoft

"Microsoft believes that for mixed reality to thrive, it must be open for everyone: open stores, open browsers and open developer platforms. We're dedicated to supporting the launch of OpenXR this year on Windows Mixed Reality and HoloLens 2. To help developers provide feedback, we're releasing today a developer preview of our OpenXR runtime with support for Windows Mixed Reality headsets."



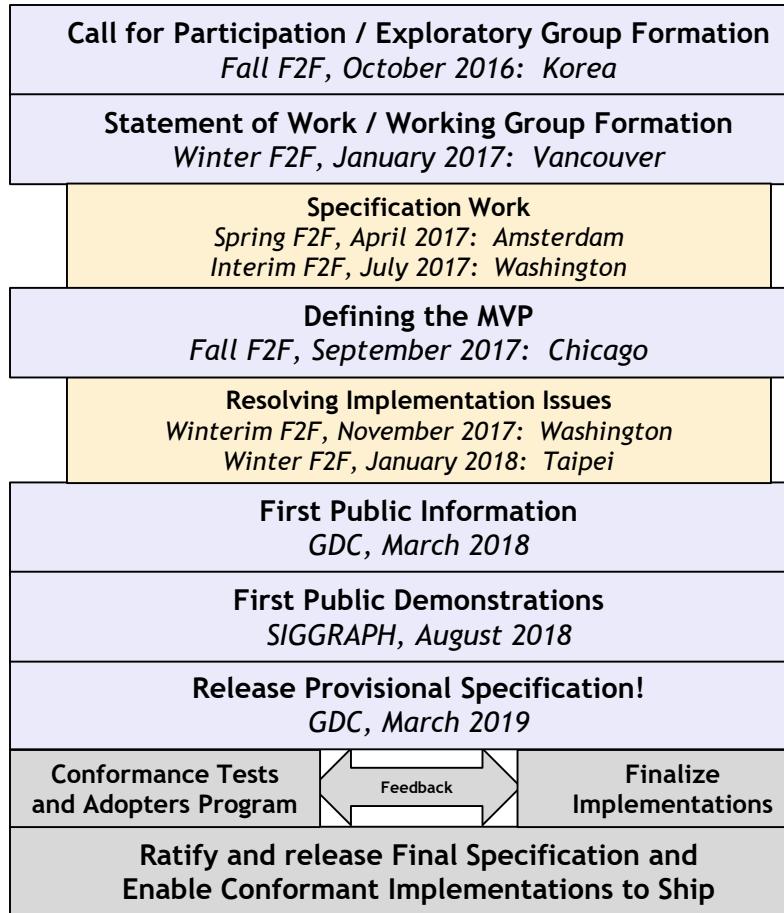
Philippe Kalaf, CEO, Collabora

Collabora is excited to announce Monado, an open source implementation of the newly released OpenXR spec. More than just a vendor SDK, Monado is an open source project and codebase to harness and focus wider community effort around XR technologies.

What's Next?

What's Next?

KHRONOS®



Drive towards 1.0 release!

What's Next?

- Continue Refining the Specification
 - Very unlikely any new functionality between now and 1.0
- Incorporate Community Feedback
 - Bug fixes, things missing, etc.
- Finalize Implementations
- Establish an Adopter's Program
- **Develop conformance tests!**
 - Major next hurdle
 - Conformance tests are critical to the health of an API (particularly a new one)
 - Potentially lots of corner cases requiring tests
 - Make sure not just the common paths work well
 - Make sure a particular vendor's implementation does not become the de facto standard
 - How to test AR and VR systems end to end ?
 - Can you test successfully without robotic arms and cameras?

Thanks!

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



Thanks to the Engineers!

Adam Gousetis, Google | Alex Turner, Microsoft | Andreas Loeve Selvik, Arm | Andres Rodriguez, Valve Software | Armelle Laine, Qualcomm Technologies, Inc | Blake Taylor, Magic Leap | Brad Grantham, Google | Brandon Jones, Google | Brent E. Insko, Intel | Brent Wilson, Microsoft | Bryce Hutchings, Microsoft | Cass Everitt, Facebook | Charles Egenbacher, Epic Games | Christoph Haag, Collabora | Craig Donner, Google | Dan Ginsburg, Valve Software | Dave Houlton, LunarG | Dave Shreiner, Unity Technologies | Denny Rönnsgren, Tobii | Dmitriy Vasilev, Samsung | Doug Twileager, ZSpace | Ed Hutchins, Facebook | Gloria Kennickell, Facebook | Gregory Greeby, AMD | Guodong Chen, Huawei | Jakob Bornecrantz, Collabora | Jared Cheshier, PlutoVR | Javier Martinez, Intel | Jeff Bellinghausen, Valve Software | Jiehua Guo, Huawei | Joe Ludwig, Valve Software | Johannes van Waveren, Facebook | Jon Leech, Khronos | Jonathan Wright, Facebook | Juan Wee, Samsung | Jules Blok, Epic Games | Karl Schultz, LunarG | Kaye Mason, Google | Krzysztof Kosiński, Google | Lachlan Ford, Microsoft | Lubosz Sarnecki, Collabora | Mark Young, LunarG | Martin Renschler, Qualcomm Technologies, Inc. | Matias Koskela, Tampere University of Technology | Matt Wash, Arm | Mattias Brand, Tobii | Mattias O. Karlsson, Tobii | Michael Gatson, Dell | Minmin Gong, Microsoft | Mitch Singer, AMD | Nell Waliczek, Microsoft | Nick Whiting, Epic Games | Nigel Williams, Sony | Paul Pedriana, Facebook | Peter Kuhn, Unity Technologies | Peter Peterson, HP Inc. | Pierre-Loup Griffais, Valve Software | Rajeev Gupta, Sony | Remi Arnaud, Starbreeze | Remy Zimmerman, Logitech | River Gillis, Google | Robert Memmott, Facebook | Robert Menzel, NVIDIA | Robert Simpson, Qualcomm Technologies, Inc. | Robin Bourianes, Starbreeze | Ryan Pavlik, Collabora | Ryan Vance, Epic Games | Sam Martin, Arm | Satish Salian, NVIDIA | Scott Flynn, Unity Technologies | Sophia Baldonado, PlutoVR | Sungye Kim, Intel | Tom Flynn, Samsung | Trevor F. Smith, Mozilla | Vivek Viswanathan, Dell | Yin Li, Microsoft | Yuval Boger, Sensics

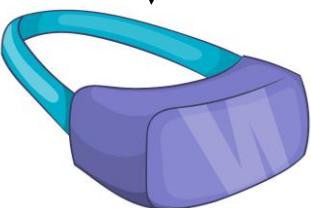
Recap

- What is OpenXR?
- A Brief History of the Standard
- What are the Problems we are trying to Solve
- OpenXR Timeline of Development
- Technical Walkthrough
- Provisional Release
- What's Next?

OpenXR Win-Win-Win

XR Vendors

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



XR End-Users

Can run the apps they want on their system
- reducing market confusion and increasing consumer confidence



XR ISVs

Can easily ship on more platforms for increased market reach

The OpenXR logo, featuring the word "Open" in a purple sans-serif font and "XR" in a larger, bold purple font with a trademark symbol.



Before we go... the Call for Action...

Before we go... the Call for Action... **Feedback!**

- Tell us:
 - What's wrong with the spec
 - What should be in the spec
 - What shouldn't be in the spec
 - How this won't work for your application/runtime/hardware/OS/...
 - How purple is a great color choice!

Before we go... the Call for Action... Feedback!

- Tell us:
 - What's wrong with the spec
 - What should be in the spec
 - What shouldn't be in the spec
 - How this won't work for your application/runtime/hardware/OS/...
 - How purple is a great color choice!
- Better yet, send us examples!**

Join Khronos!

- Get more involved
- Have direct impact on the direction of the API
- Be part of the effort to deliver OpenXR 1.0!



Thank You!

Questions...



Tweet #KhronosDevDay to win!