Laval Virtual: OpenXR Master Class

Ryan A. Pavlik, Ph.D.
Principal Software Engineer, Collabora, Ltd.
OpenXR Working Group Spec. Editor
April 2020

Agenda

- About Me
- Introduction to OpenXR
- OpenXR in context
- Dive into OpenXR app structure/API usage
- Time permitting: Question and Answer

About Me: Ryan Pavlik

- **Open-source VR software developer since 2009**
- **OpenXR working group**
  - participant since the first official meeting in January 2017
  - elected specification editor in April 2019
- **Principal Software Engineer at Collabora**
  - Focusing on XR
  - Leading our OpenXR contributions
  - Developer on Monado

COLLABORA

KHRONOS GROUP

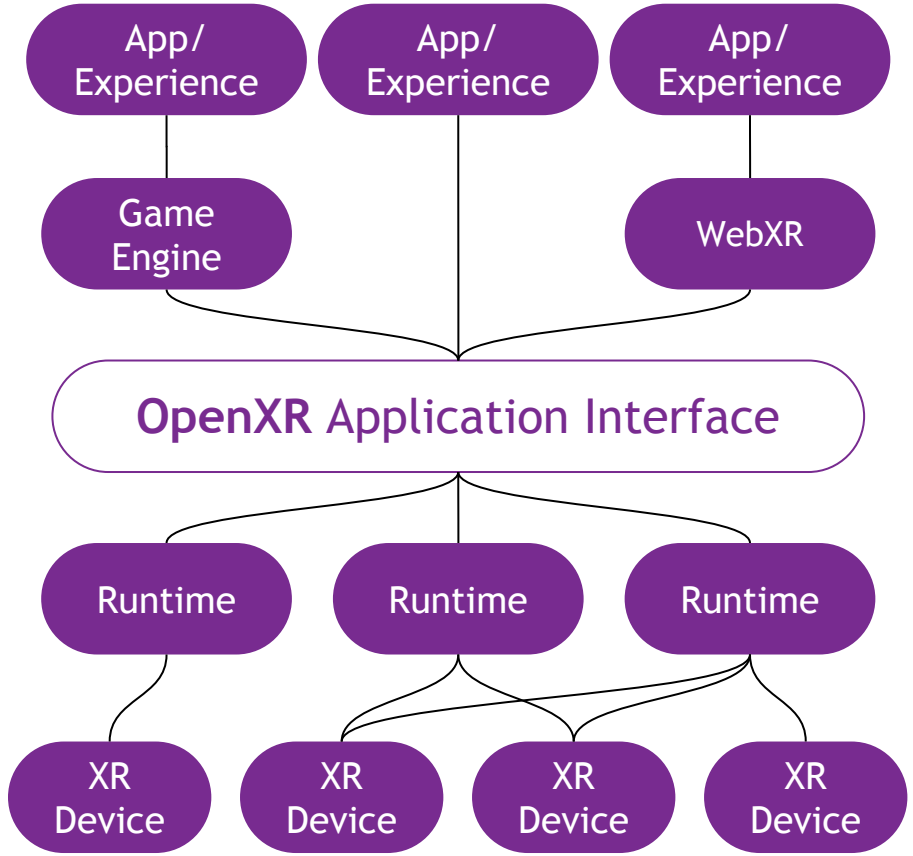
About OpenXR™

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.

- **OpenXR 0.90 provisional release at GDC 2019**
  - Show progress publicly and seek feedback
- **OpenXR 1.0 released at SIGGRAPH 2019.**
  - Defines an interface between an engine/application and a runtime, and required behavior of a runtime.
  - Runtime conformance tests and adopter program forthcoming.

Status

- **OpenXR 1.0 includes the "Application Interface" as stable, with a compatibility promise**
  - 1.0.x patch releases fix spec bugs and add extensions
  - Typically on Fridays
  - If there are spec or extension changes in Khronos private GitLab on a given week, I will typically do a release
- **The "Device Layer" as mentioned in various marketing material is still WIP**
  - No public ETA: WG focusing on conformance testing and uptake



OpenXR in Context

- **A Khronos API**
  - Developed by a non-profit industry consortium
  - Extensions a key part of the design
  - Conforming runtimes receive patent protection and trademark license per the IP framework
  - Royalty-free
- **Conventions and spec style of OpenXR strongly influenced by Vulkan**
  - Mostly-shared spec toolchain
  - Similar development and release practices
  - Support for API layers

OpenXR vs Vulkan

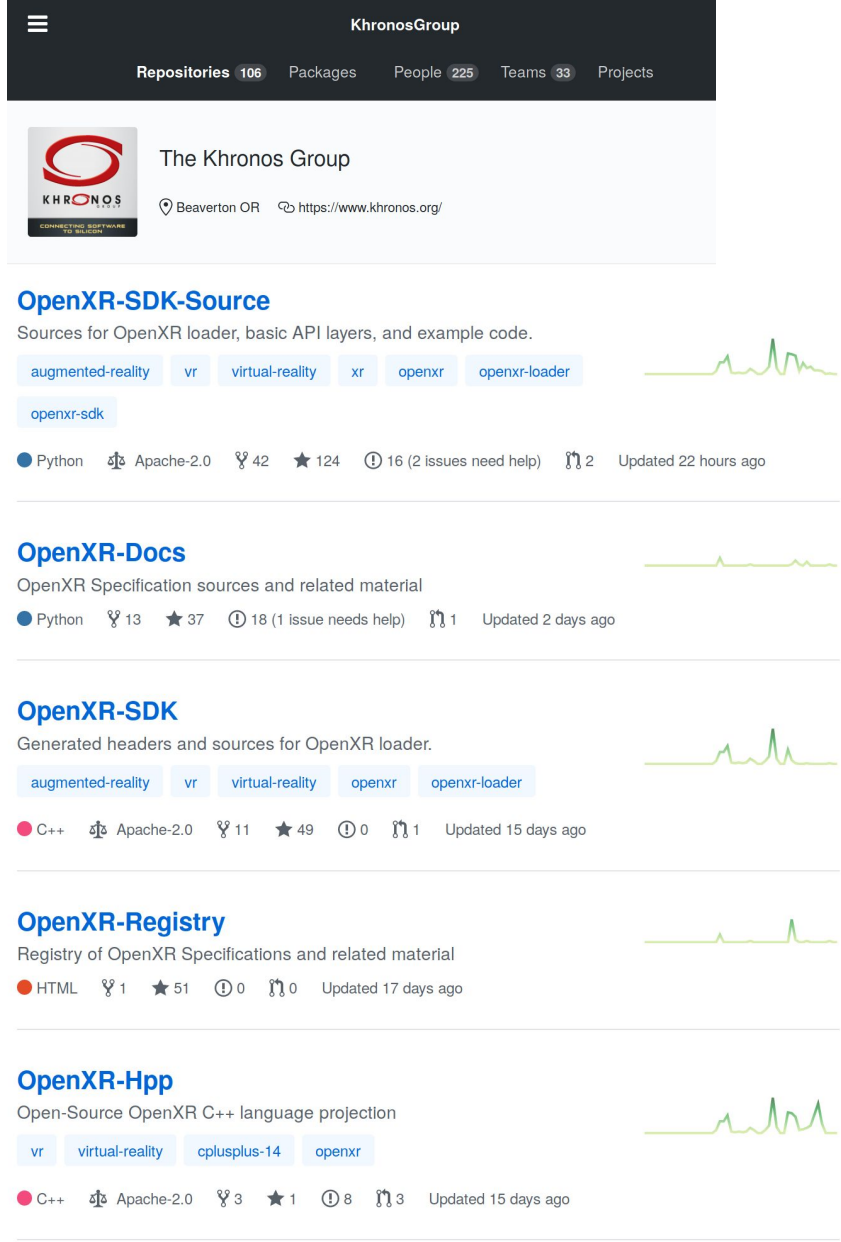
- **OpenXR is rendering-API-neutral**
  - Graphics API support in extensions designed to work similarly
- **OpenXR is a "lower-frequency" API than Vulkan, influencing design choices**
  - Runtimes must detect nearly all invalid usage and return an error code
  - Minimal risk of "Undefined Behavior" aside from that inherent in a C API
    - Can be reduced further by a language projection/wrapper
- **OpenXR is a much smaller spec**
- **Lessons learned**
  - Loader ships with apps, not system/driver, on Windows
    - Still system-wide on Linux, tentatively

Runtime Availability

- **Microsoft: OpenXR runtime for Windows MR and HoloLens 2 available** aka.ms/openxr
  - Also works with the device simulator - no hardware needed
- **Oculus**
  - https://developer.oculus.com/blog/prototype-openxr-for-oculus/
  - Desktop (Rift): runtime in Public Test Channel, SDK to be officially rolled out soon
  - Mobile (Quest): OpenXR Mobile SDK 1.0.6 available
- **Linux (multi-device)**
  - Monado - open-source project founded by Collabora monado.dev monado.freedesktop.org
- **Note that all are technically previews because conformance tests not yet complete and available.**

OpenXR and FOSS

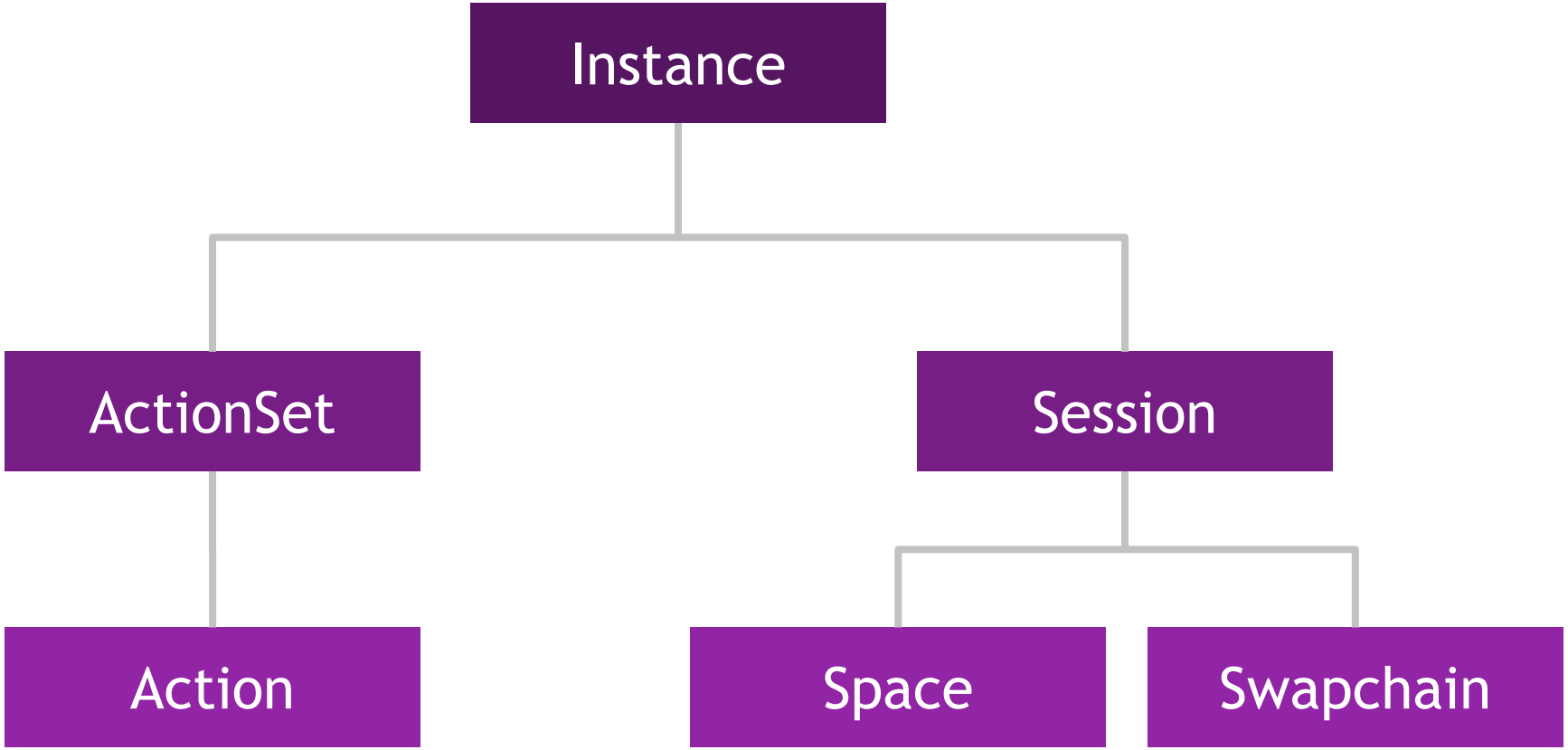
- **Most WG-originated OpenXR software and tooling is open source**
  - Licensed under Apache 2.0
  - Maintained publicly together with the community.
  - WIP: Adding GPL/LGPL 2.x compatibility exception like Vulkan's
- **Growing community developing FOSS tools, libraries, etc.**
- **Open-source runtime: Monado**
  - Development led by Collabora
- **Community has already provided valuable feedback on the spec and SDK**



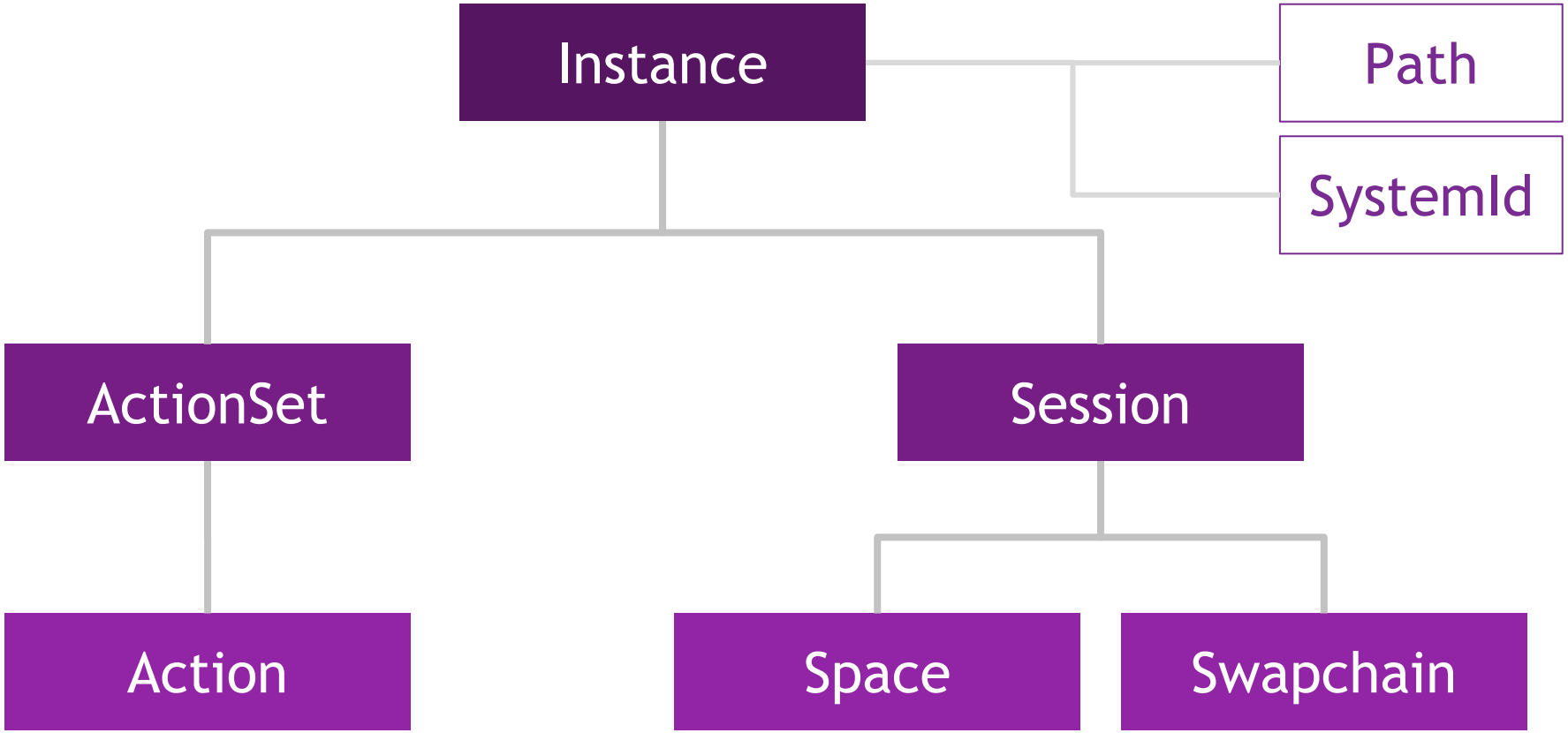
Structure of an OpenXR App

- **Get started**
  - Instance
- **Find out where/how to run**
  - SystemId atom
  - ViewConfigurationType enum
- **Set up your interaction/input handles**
  - Create Action Sets, Actions
  - Suggest bindings
- **Prepare your immersive experience**
  - Create Session
  - Attach action sets
  - Create Reference and Action Spaces
  - Create Swapchain
- **Participate in the frame loop and handle input**
  - Poll for events too

OpenXR Handle Types



Atoms



Creating an Instance

- **Choose which extensions you want**
  - Need at least one graphics binding extension
  - Can identify available extensions: xrEnumerateInstanceExtensionProperties
- **Choose which API layers you want, if any**
  - Optional
  - xrEnumerateApiLayerProperties
- **Set up application info**
  - So runtimes can identify your app
- **xrCreateInstance**

System and Views

- xrGetSystem
  - with your desired form factor: HMD or handheld
  - may be temporarily unavailable
- **View configuration**
  - Mono, Stereo, …
  - xrEnumerateViewConfigurations if you support more than one
  - xrGetViewConfigurationProperties
  - xrEnumerateViewConfigurationViews
    - mono has one view
    - stereo has two views

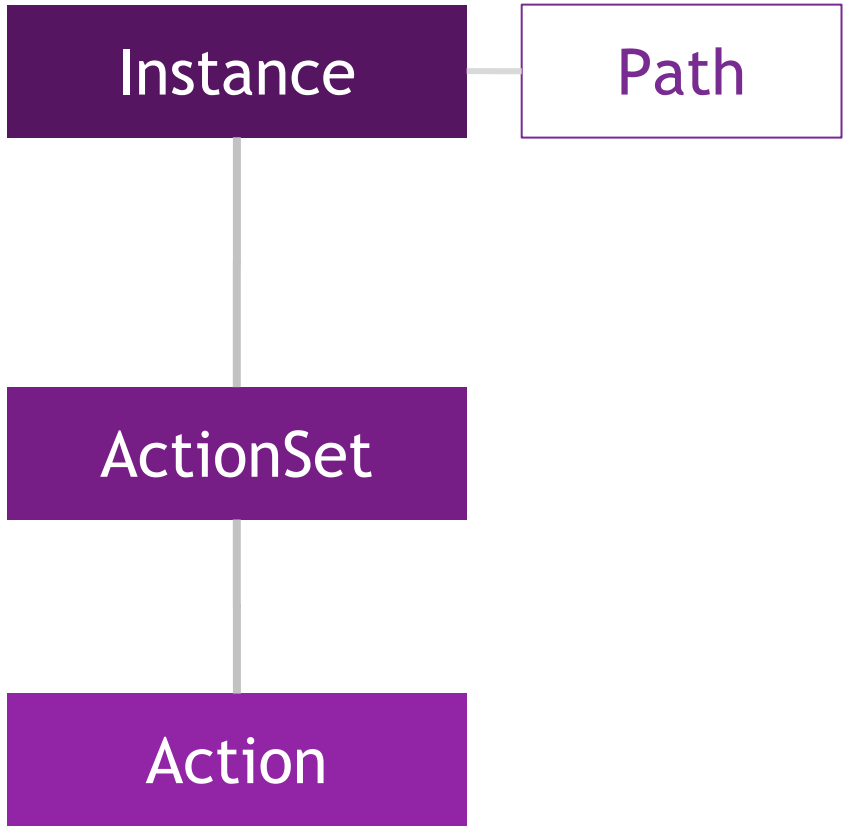
extended in vendor extension XR_VARJO_quad_views,
multi-vendor extension XR_EXT_view_configuration_depth_range

Creating your Actions

- ActionSet: **a group of related actions for a context, environment, etc.**
  - e.g. “menu”, “game”, “driving”, etc
  - xrCreateActionSet
- Action: **A semantic (meaningful) bit of interaction**
  - Types: Boolean (button), Float (analog), Vec2, Pose (tracked object), Haptic
  - e.g. “grab_object”, “teleport”, “hand_pose”
  - xrCreateAction

Suggested Bindings and Interaction Profiles

- **How you customize for hardware you've tested, without excluding the rest**
- **For each controller type you've tested ("interaction profile") call xrSuggestInteractionProfileBindings once**
  - With as many or few action-binding pairs as you like - OK if not all actions have a suggested binding
  - Can suggest multiple bindings per action in a call: e.g. both left and right hands can "`grab_object`"
  - Binding is an XrPath atom representing a path string like `/user/hand/right/input/select/click`



interaction profiles added by vendor extension XR_MSFT_hand_interaction, multi-vendor extension XR_EXT_eye_gaze_interaction

Sample of Actions

- These are the actions from “hello_xr” - see `OpenXrProgram::InitializeActions`
- All in one action set, “`gameplay`”, due to simplicity of the app
- All are specified for both left and right hand as “subaction paths” because we might want to know which hand did an action
  - which hand grabbed object, etc.

actionName	localizedActionName	actionType	subaction path
`grab_object`	Grab Object	Float Input	/user/hand/left /user/hand/right
`hand_pose`	Hand Pose	Pose Input	/user/hand/left /user/hand/right
`quit_session`	Quit Session	Boolean Input	/user/hand/left /user/hand/right
`vibrate_hand`	Vibrate Hand	Vibration Output	/user/hand/left /user/hand/right

xrSuggestInteractionProfileBindings 1

- **Standard defines “khr/simple_controller” as a minimal subset profile**
- **Note here that `grab_object` is float, but suggested to bind to “select/click” (boolean)**
  - Runtime will automatically convert boolean to a 1 or 0.

actionName	actionType	subaction path	/interaction_profiles/khr/simple_controller
`grab_object`	Float Input	/user/hand/left	/user/hand/left/input/select/click
		/user/hand/right	/user/hand/right/input/select/click
`hand_pose`	Pose Input	/user/hand/left	/user/hand/left/input/grip/pose
		/user/hand/right	/user/hand/right/input/grip/pose
`quit_session`	Boolean Input	/user/hand/left	/user/hand/left/input/menu/click
		/user/hand/right	/user/hand/right/input/menu/click
`vibrate_hand`	Vibration Output	/user/hand/left	/user/hand/left/output/haptic
		/user/hand/right	/user/hand/right/output/haptic

xrSuggestInteractionProfileBindings 2

- HTC Vive controller
- The `grab_object` action is here suggested for the “squeeze/click” input
  - still boolean like simple controller, but squeeze instead of select

actionName	actionType	subaction path	/interaction_profiles/htc/vive_controller
`grab_object`	Float Input	/user/hand/left	/user/hand/left/input/squeeze/click
		/user/hand/right	/user/hand/right/input/squeeze/click
`hand_pose`	Pose Input	/user/hand/left	/user/hand/left/input/grip/pose
		/user/hand/right	/user/hand/right/input/grip/pose
`quit_session`	Boolean Input	/user/hand/left	/user/hand/left/input/menu/click
		/user/hand/right	/user/hand/right/input/menu/click
`vibrate_hand`	Vibration Output	/user/hand/left	/user/hand/left/output/haptic
		/user/hand/right	/user/hand/right/output/haptic

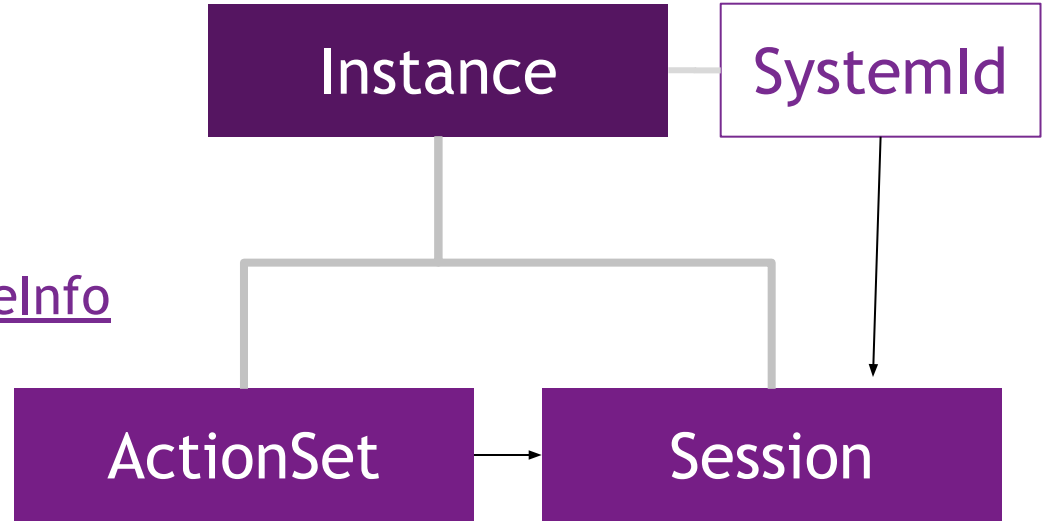
xrSuggestInteractionProfileBindings 3

- Oculus Touch controller
- Has a float input suitable for `grab_object` action - called “squeeze/value”
- Only left controller has a menu button, so not suggesting a binding for `quit_session` on the right hand.

actionName	actionType	subaction path	/interaction_profiles/oculus/touch_controller
`grab_object`	Float Input	/user/hand/left	/user/hand/left/input/squeeze/value
		/user/hand/right	/user/hand/right/input/squeeze/value
`hand_pose`	Pose Input	/user/hand/left	/user/hand/left/input/grip/pose
		/user/hand/right	/user/hand/right/input/grip/pose
`quit_session`	Boolean Input	/user/hand/left	/user/hand/left/input/menu/click
		/user/hand/right	
`vibrate_hand`	Vibration Output	/user/hand/left	/user/hand/left/output/haptic
		/user/hand/right	/user/hand/right/output/haptic

Creating your Session

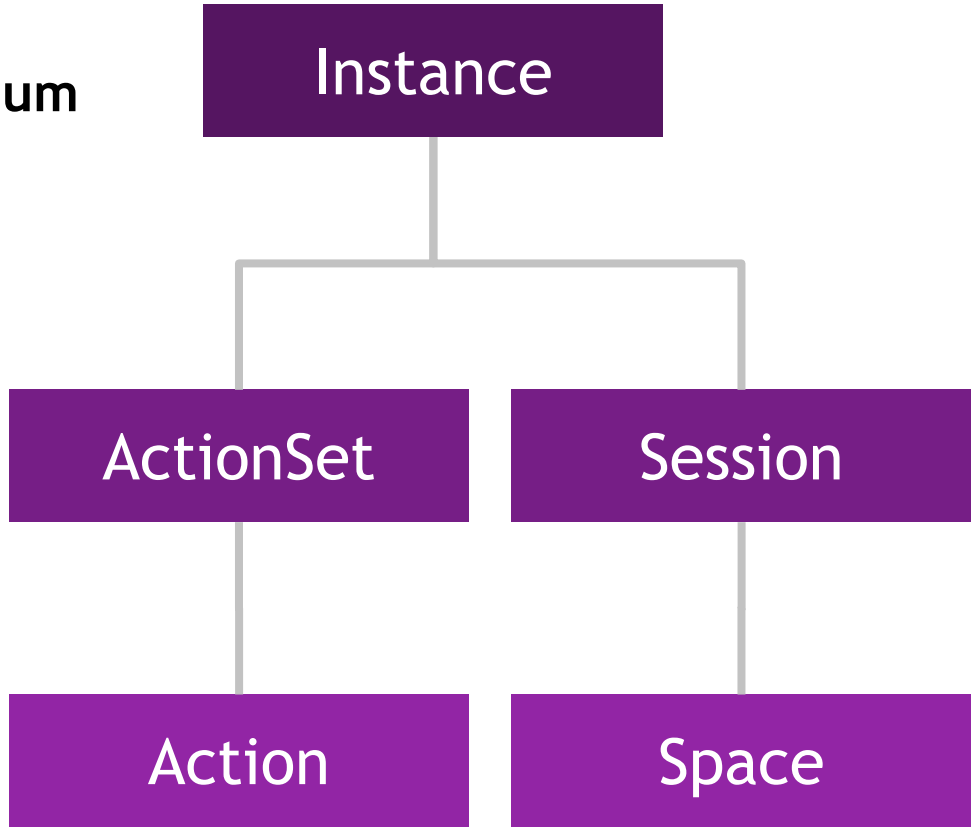
- **Graphics binding**
  - Do your graphics binding's "GetGraphicsRequirements" call
  - Create your graphics binding struct
  - Chain it via next on XrSessionCreateInfo
- xrCreateSession
  - Requires a SystemId
- **Attach your action sets to the session**
  - xrAttachSessionActionSets
  - Associates them with the session
  - Makes them immutable
  - Editor authors: tear down session, actions, action sets to modify them
- **Why is action setup done all up front?**



extended in vendor extension XR_MND_headless

Create Spaces

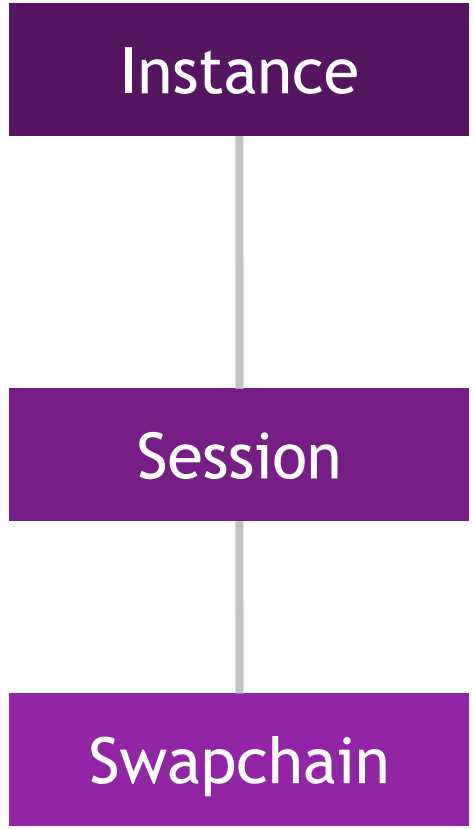
- **Multiple ways to get XrSpace handles**
- **Reference space: from Session and enum**
  - local space
  - view space
  - stage space
  - xrCreateReferenceSpace
- **Action space: from Session and pose Action**
  - xrCreateActionSpace
- **For both reference and action spaces**
  - Session is the parent handle
  - Can specify an additional, fixed transform at handle creation time
  - xrLocateSpace



extended in vendor extensions XR_MSFT_spatial_anchor, XR_MSFT_unbounded_reference_space

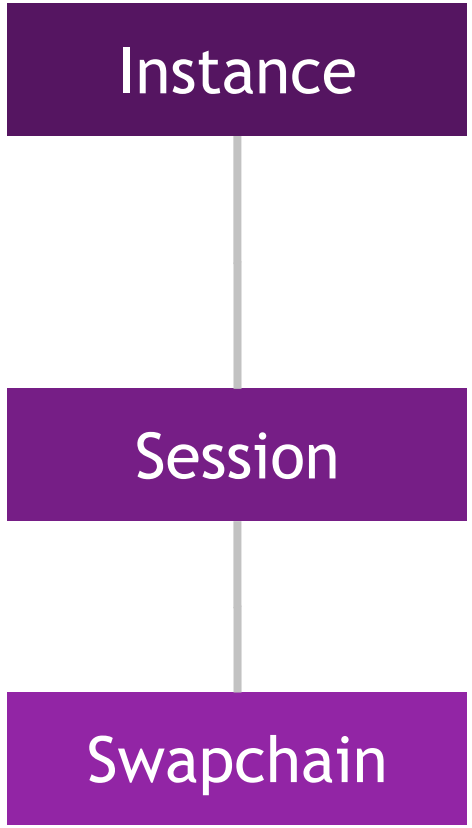
Create your Swapchain

- **Get graphics API-specific formats via xrEnumerateSwapchainFormats**
- **xrCreateSwapchain**
- **Get access to graphics API-specific handles/references to the swapchain images**
  - xrEnumerateSwapchainImages
  - Pass array of *extension-defined* structures
  - Save this information to use every frame



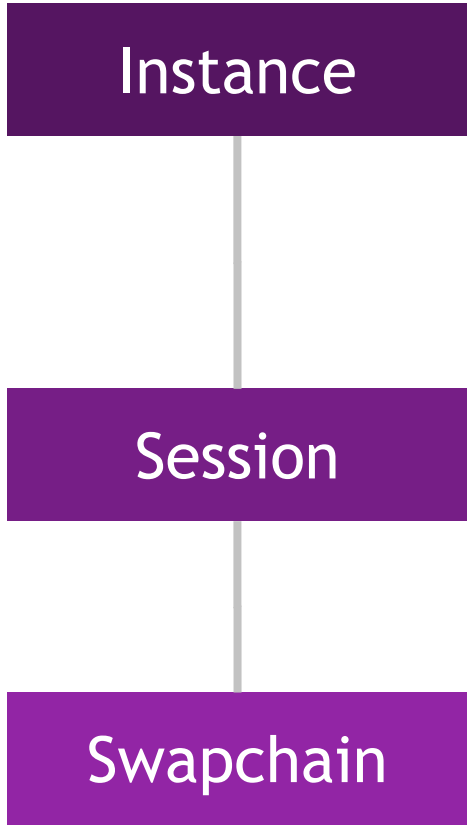
Frame loop

- **Frame functions**
  - xrWaitFrame to block until head-pose-dependent sim and rendering
  - xrBeginFrame to mark start of render
  - xrEndFrame to submit the image
  - xrBeginFrame/xrEndFrame calls must be ordered “as if” single-threaded
  - Populate XrFrameEndInfo::displayTime using output of xrWaitFrame



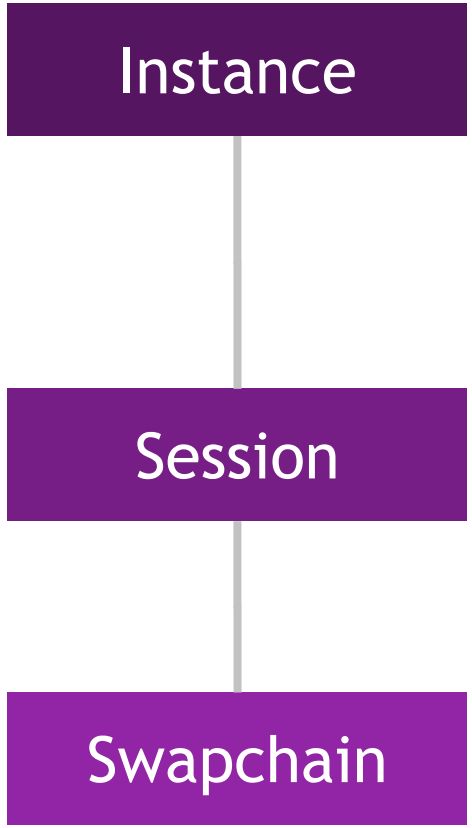
Pipelined rendering

- **Frame function synchronization**
  - At most one simultaneous xrWaitFrame call at a time
  - Each xrWaitFrame must eventually be matched with a unique xrBeginFrame
  - Any xrWaitFrame call must block until the previous frame's xrBeginFrame



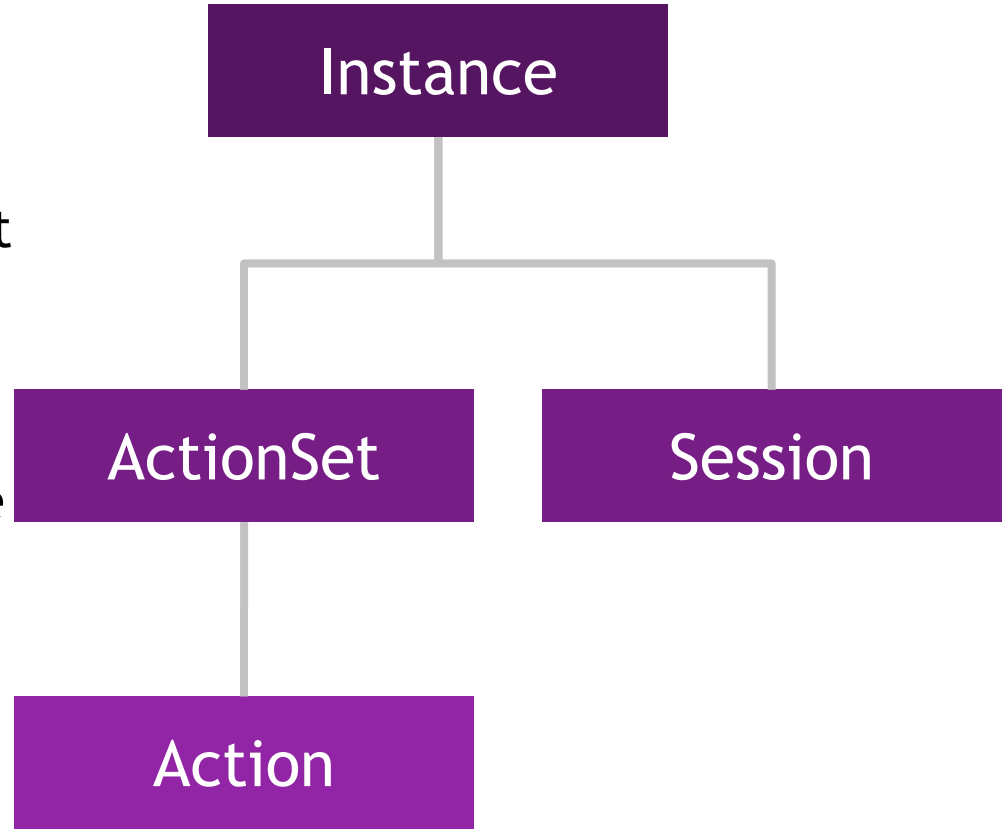
Swapchain and view management

- **Swapchain management**
  - xrAcquireSwapchainImage to get index
    - To look up/create your command buffers
  - xrWaitSwapchainImage before writing
    - Typically immediately after acquire
    - Do not submit command buffers until this returns
  - xrReleaseSwapchainImage before xrEndFrame: implicitly uses most recently released image
- xrLocateViews



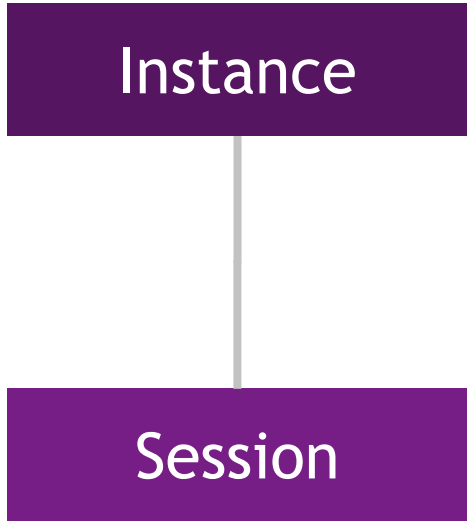
Getting input

- xrSyncActions
  - Specify which ActionSets should be active at this time
  - This is the only time non-pose input data updates
- **Get the data**
  - All ActionSets attached but not specified in xrSyncActions will have their actions return “not active”
  - Actions might not get data if your session is not focused, for privacy/security
  - xrGetActionState* calls
- **Poses: use xrLocateSpace**



Events

- **xrPollEvents**
  - Requires an Instance
  - Many events only happen during a Session
- **Describes changes to**
  - Active interaction profile
  - Continuity of reference spaces/tracking
  - Session state
- **Provide an XrEventDataBuffer for the runtime to populate with an event of some other type**



Wrap-up

- **Outline**
  - About Me
  - Introduction to OpenXR
  - OpenXR in context
  - OpenXR app structure/API usage
  - Time permitting: Question/Answer
- **Resources**
  - Landing page with news: khronos.org/openxr
  - API registry (links to the spec, ref pages, all the repos, etc) khronos.org/registry/openxr

Thank you!

- **Community**
  - Source, issue trackers, etc github.com/KhronosGroup?q=openxr
  - Chat khr.io/slack
  - Forum community.khronos.org/c/openxr
- **Open-Source Runtime for Linux: Monado**
  - Community project founded by Collabora, not a Khronos/OpenXR WG project
  - Repos, including additional (cross-platform) OpenXR-related projects gitlab.freedesktop.org/monado

FOSS XR - OpenXR Overview

Ryan A. Pavlik, Ph.D.
Principal Software Engineer, Collabora, Ltd.
OpenXR Working Group Spec. Editor
October 2019