Standardizing all the Realities: A Look at OpenXR

Robert Menzel
December 2018

A Note on What We'll Cover

3D Development (Today)

Pick an existing engine (e.g. Unity, Unreal)

or

Write your own engine (e.g. using Vulkan)

Pick a GPU

3D Development (Today)

Pick an existing engine (e.g. Unity, Unreal)

or

Write your own engine (e.g. using Vulkan)

Pick a GPU

VR Development (Ideal)

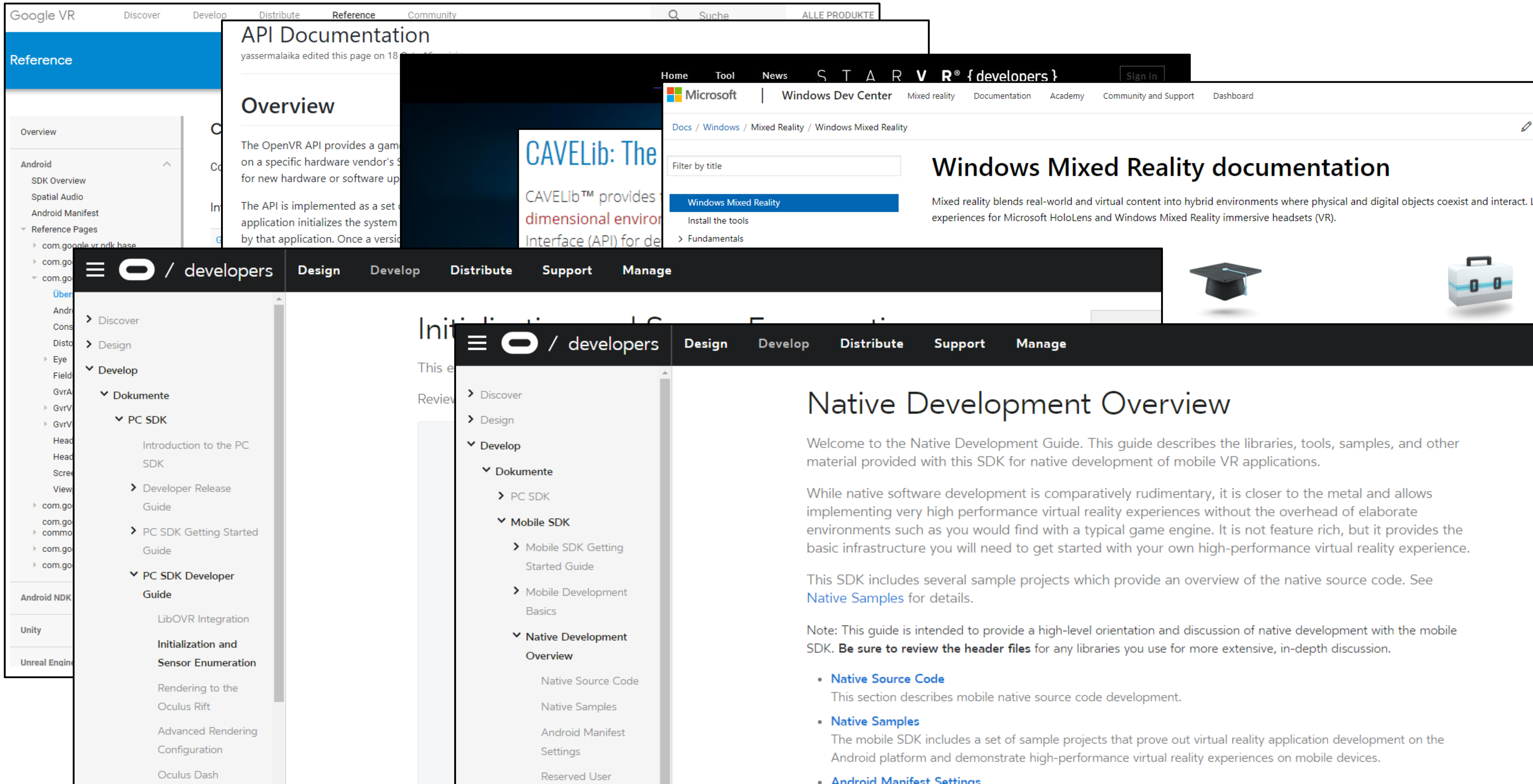
Pick an existing engine (e.g. Unity, Unreal)

or

Write your own engine

Pick a HMD

VR Development: Reality today



Google VR Discover Develop Distribute Reference Community Suche ALLE PRODUKTE

Reference

Overview

Android
SDK Overview
Spatial Audio
Android Manifest
Reference Pages
com.google.vr.ndk.base
com.go
com.go
Über
Andr
Cons
Disto
Eye
Field
GvrA
GvrV
GvrV
Head
Head
Scree
View
com.go
com.go
commo
com.go
com.go

Android NDK

Unity

Unreal Engin

C

Co

In

API Documentation

yassermalaika edited this page on 18

Overview

The OpenVR API provides a gam
on a specific hardware vendor's S
for new hardware or software up

The API is implemented as a set
application initializes the system
by that application. Once a versi

Home Tool News STARVR® { developers } Sign in

CAVELib: The

CAVELib™ provides
dimensional enviror
Interface (API) for de

Microsoft | Windows Dev Center Mixed reality Documentation Academy Community and Support Dashboard

Docs / Windows / Mixed Reality / Windows Mixed Reality

Filter by title

Windows Mixed Reality
Install the tools
Fundamentals

Windows Mixed Reality documentation

Mixed reality blends real-world and virtual content into hybrid environments where physical and digital objects coexist and interact.
experiences for Microsoft HoloLens and Windows Mixed Reality immersive headsets (VR).

/ developers Design Develop Distribute Support Manage

Discover
Design
Develop
Dokumente
PC SDK
Introduction to the PC SDK
Developer Release Guide
PC SDK Getting Started Guide
PC SDK Developer Guide
LibOVR Integration
Initialization and Sensor Enumeration
Rendering to the Oculus Rift
Advanced Rendering Configuration
Oculus Dash

Init

This e

Reviev

/ developers Design Develop Distribute Support Manage

Discover
Design
Develop
Dokumente
PC SDK
Mobile SDK
Mobile SDK Getting Started Guide
Mobile Development Basics
Native Development Overview
Native Source Code
Native Samples
Android Manifest Settings
Reserved User

Native Development Overview

Welcome to the Native Development Guide. This guide describes the libraries, tools, samples, and other material provided with this SDK for native development of mobile VR applications.

While native software development is comparatively rudimentary, it is closer to the metal and allows implementing very high performance virtual reality experiences without the overhead of elaborate environments such as you would find with a typical game engine. It is not feature rich, but it provides the basic infrastructure you will need to get started with your own high-performance virtual reality experience.

This SDK includes several sample projects which provide an overview of the native source code. See Native Samples for details.

Note: This guide is intended to provide a high-level orientation and discussion of native development with the mobile SDK. **Be sure to review the header files** for any libraries you use for more extensive, in-depth discussion.

- Native Source Code
  This section describes mobile native source code development.
- Native Samples
  The mobile SDK includes a set of sample projects that prove out virtual reality application development on the Android platform and demonstrate high-performance virtual reality experiences on mobile devices.
- Android Manifest Settings

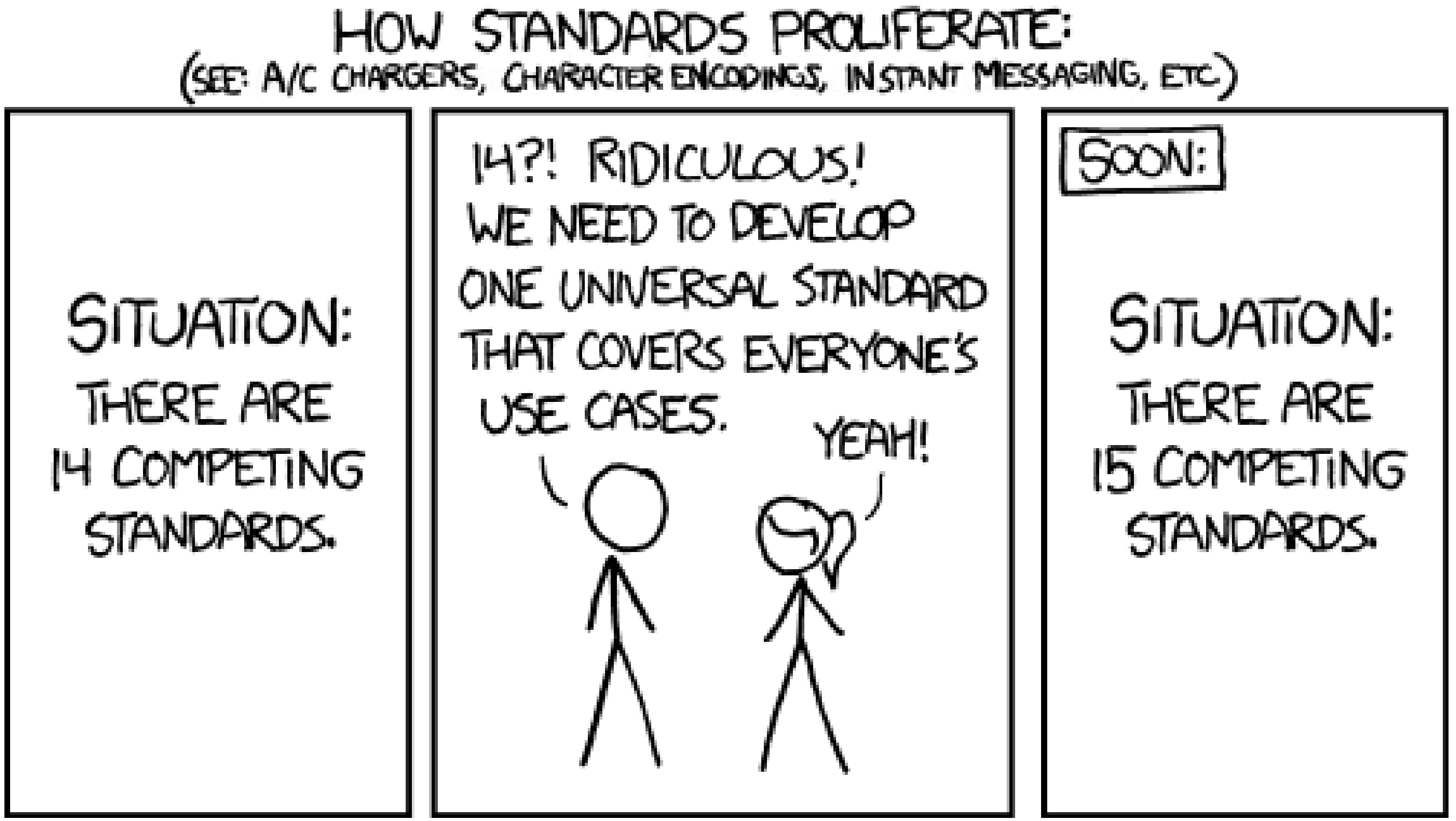
VR Development: Reality today

- VR API choice limits the hardware your application will support
- Supporting multiple APIs is possible, but more expensive

VR Development: Reality today

- VR API choice limits the hardware your application will support
- Supporting multiple APIs is possible, but more expensive

- Fragmentation is bad for hardware vendors as well!


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Randall Munroe, XKCD, https://xkcd.com/927/

KHRONOS® GROUP

OpenGL®	WebGL™	Vulkan®	glTF™
Since 1992	Since 2011	Since 2016	Since 2015
Latest release: 2017	Latest release: 2017	Latest release: 2018	Latest release: 2017
3D API for PCs	3D API for Web	3D API for PC, Console, Mobile	3D file format
Windows, Linux, MacOS X, FreeBSD, ...	Chrome, Firefox, Safari, Opera, IE, ...	Windows, Linux, Nintendo Switch, Android...	Blender, 3ds Max, Maya, Paint 3D, PowerPoint, ...

OpenXR Contributors

AMD antilatency arm COLLABORA EPIC GAMES Google HTC

HUAWEI Imagination intel LG logitech LUNAR G magic leap

MEDIATEK Microsoft moz://a NOKIA NVIDIA. oculus PICO

pluto QUALCOMM RAZER SAMSUNG Sony Interactive Entertainment STARBREEZE tobii

unity VALVE VARJO VeriSilicon 兆芯 zSpace

KHRONOS GROUP

A Brief History of the Standard

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

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

Specification Work
Spring F2F, April 2017: Amsterdam
Interim F2F, July 2017: Seattle
Fall F2F, September 2017: Chicago
Winterim F2F, November 2017: Seattle
Winter F2F, January 2018: Taipei

First Public Information! -- *GDC, March 2018: San Francisco*

Specification Work
Spring F2F, April 2018: Montreal
Fall F2F, September 2018: Budapest

Updates & First Demonstration! -- *SIGGRAPH, August 2018*

Today $\longrightarrow$

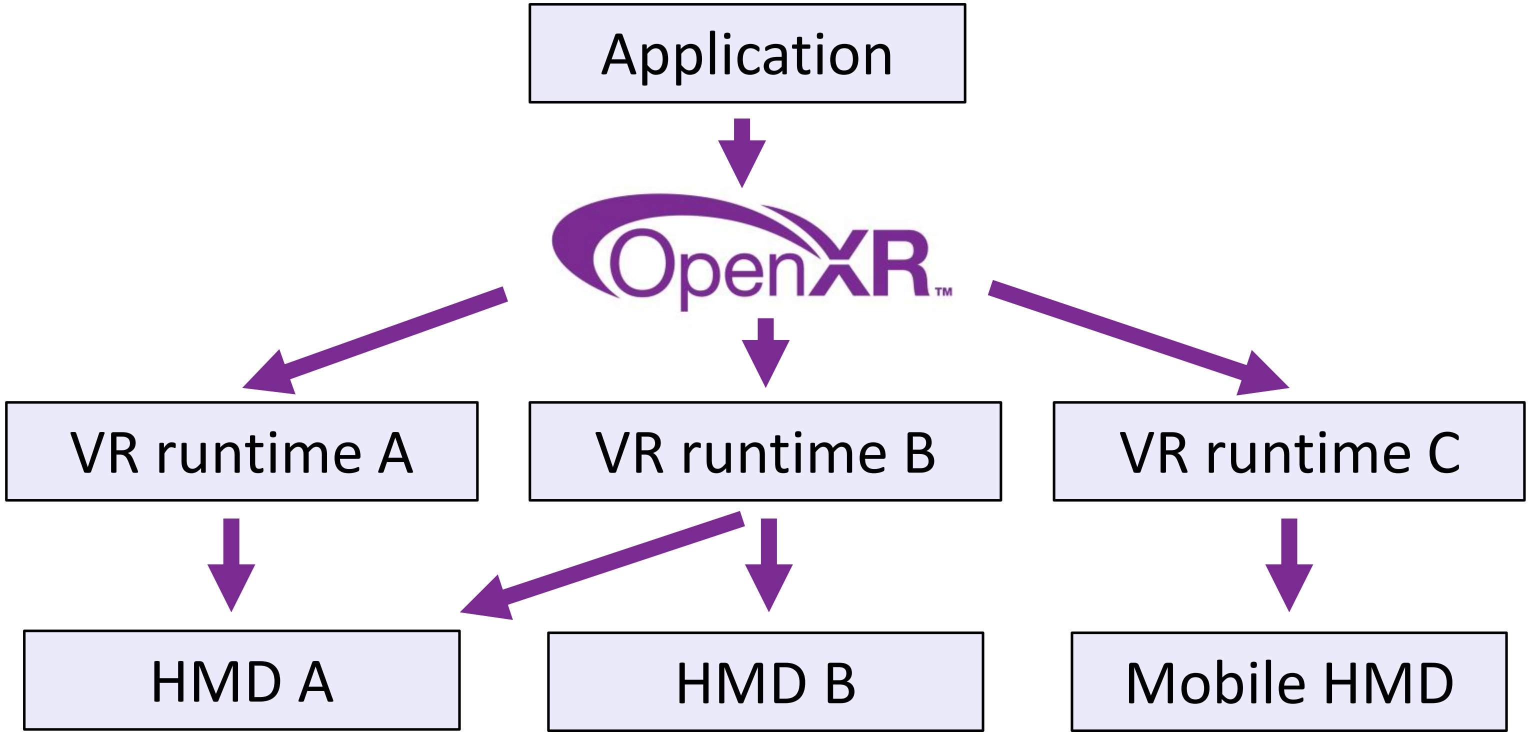
Specification Work
Winterim F2F, December 2018: Seattle (right now)
Winter F2F, January 2019: San Diego

Provisional Release

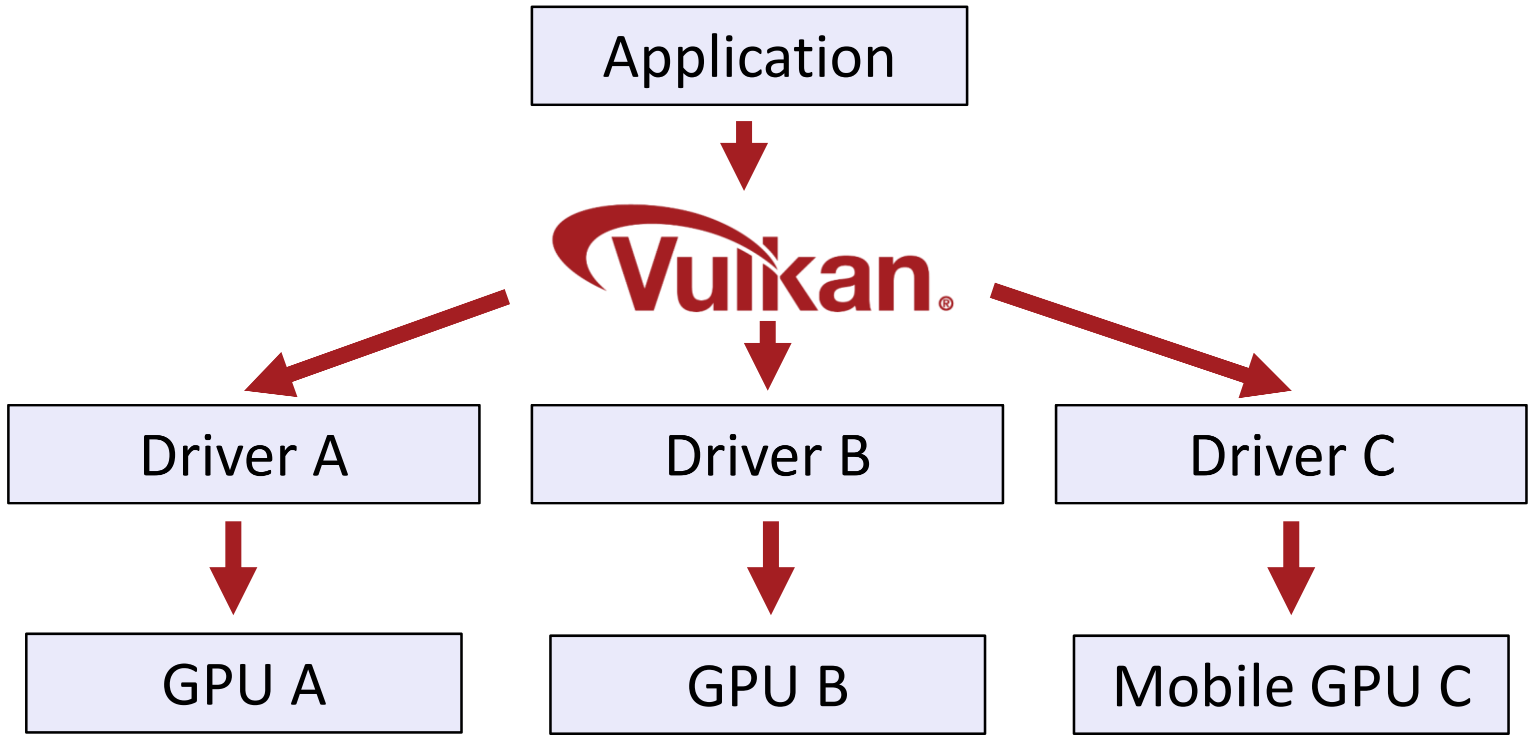
Testing and Implementation

Ratification and Release

OpenXR: Vision for Version 1.0

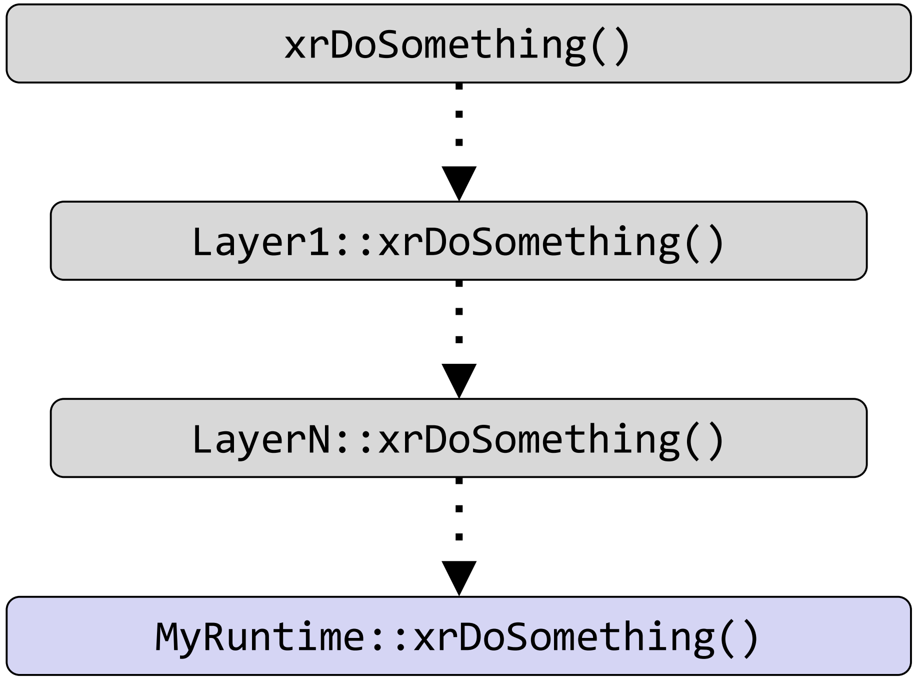


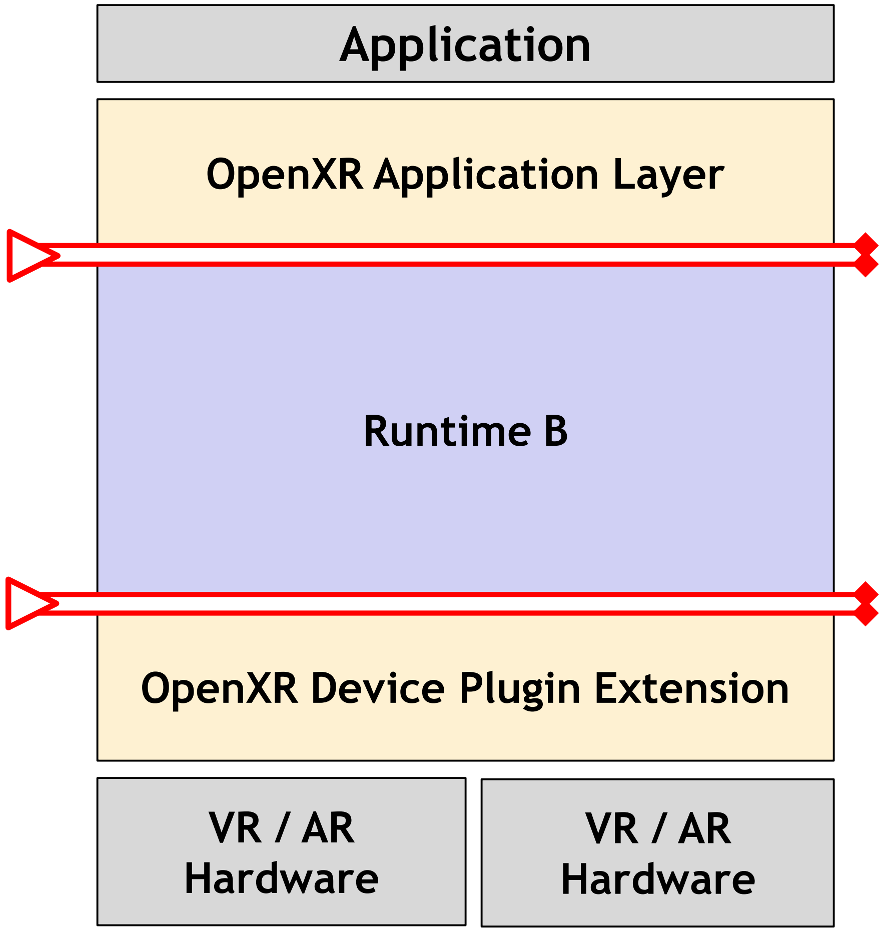
Vulkan: Same Concept, one API for all Devices



Architecture Overview

Layered API


xrDoSomething()
Layer1::xrDoSomething()
LayerN::xrDoSomething()
MyRuntime::xrDoSomething()


Application
OpenXR Application Layer
Runtime B
OpenXR Device Plugin Extension
VR / AR Hardware
VR / AR Hardware

Core and Extensions

Core Standard	Core concepts that are fundamental to the specification for all use cases *Examples: Tracking, Controller input, Presenting Renderings*
KHR Extensions	Functionality that a large classes of runtimes will likely implement *Examples: Platform support, Device Plugin, Tracking Bounds*
EXT Extensions	Functionality that a few runtimes might implement *Examples: Performance Settings, Debug Utils*
Vendor Extensions	Functionality that is limited to a specific vendor *Examples: Device specific functionality*

Viewport Configurations

Camera Passthrough AR	Stereoscopic VR	Projection CAVE
		Photo Credit: Dave Pape
One Viewport	Two Viewports (one per eye)	Twelve Viewports (six per eye)

Applications can:

- Query the runtime for its supported Viewport Configurations
- Applications can then set the Viewport Configurations that they plan to use

Runtimes can:

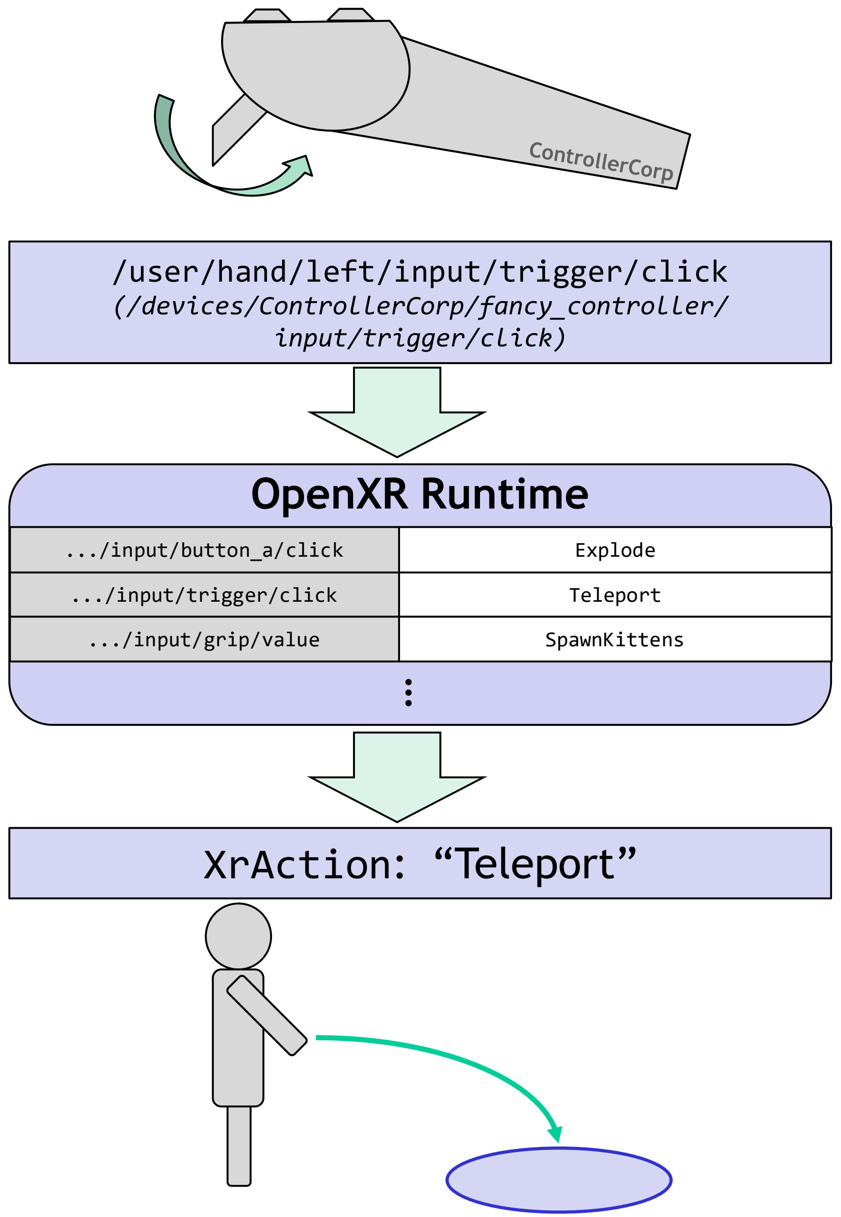
- Request the application change configuration, but app is not required to comply

Input and Haptics

Input in OpenXR goes through a layer of abstraction built around Input Actions

These allow application developers to define input based on resulting action (*e.g. "Move," "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, etc.).

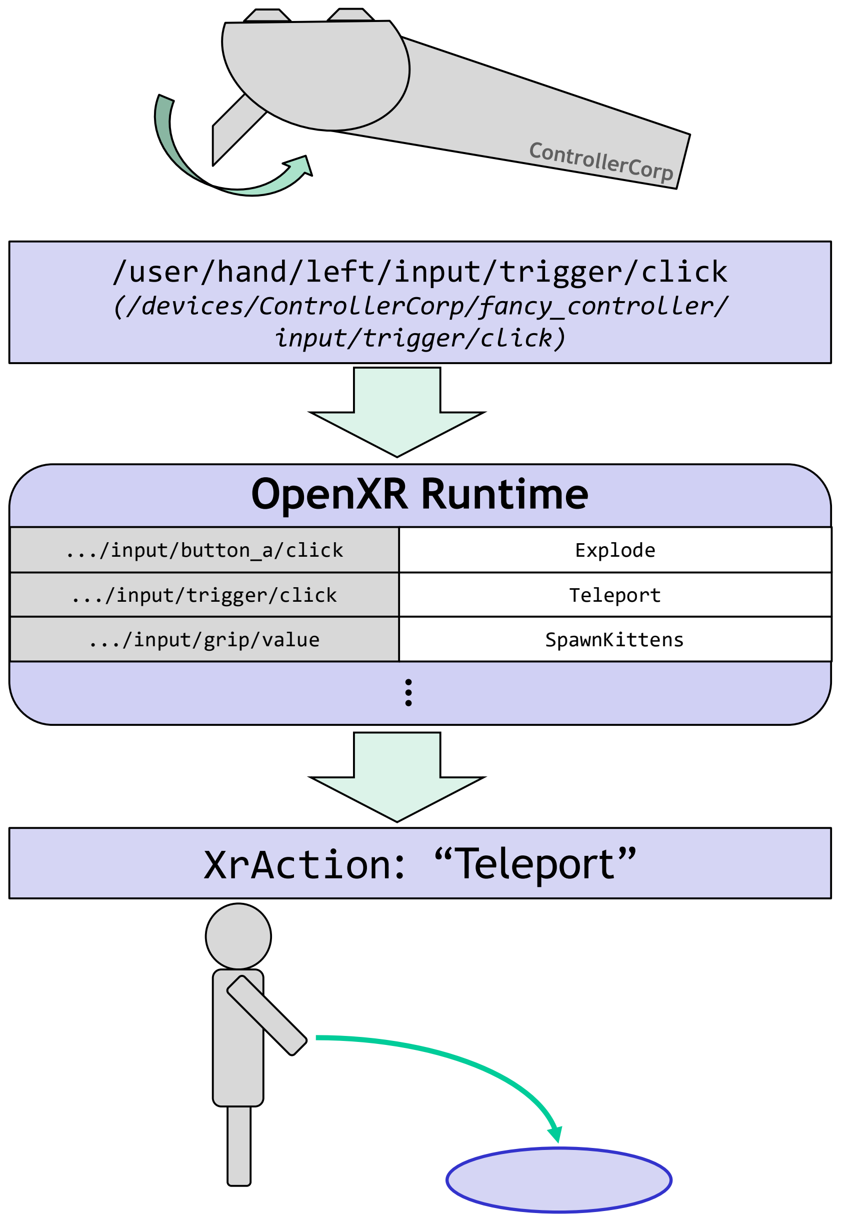

ControllerCorp
/user/hand/left/input/trigger/click
(/devices/ControllerCorp/fancy_controller/ input/trigger/click)
OpenXR Runtime
⋮
XrAction: "Teleport"

.../input/button_a/click	Explode
.../input/trigger/click	Teleport
.../input/grip/value	SpawnKittens

Input and Haptics

Forcing applications through this indirection has several advantages:

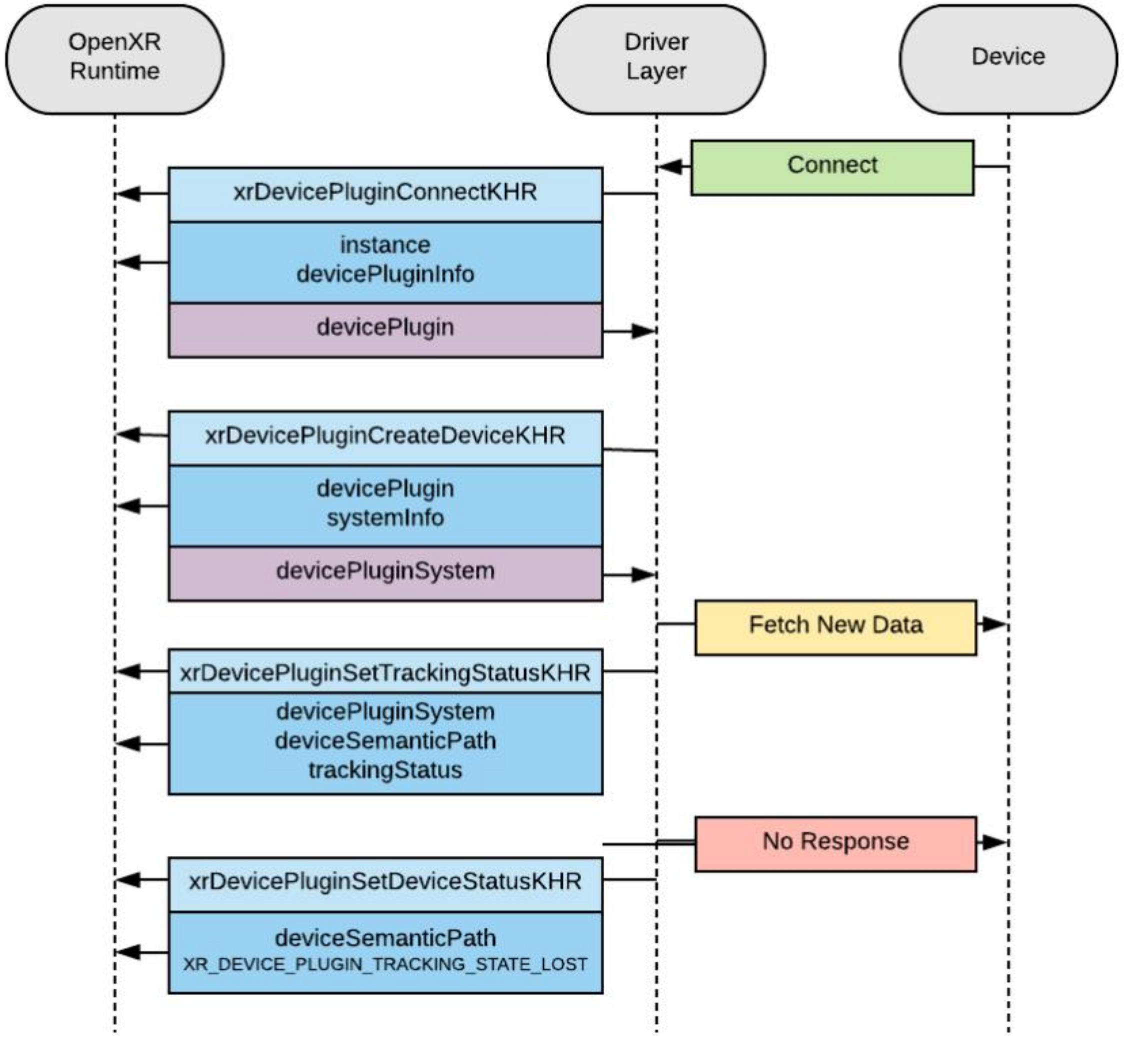
- Greater future-proofing as improvements to hardware and runtimes come out
- Allows for runtimes to “mix-and-match” multiple input sources
- Easy optional feature support (e.g. body tracking)
- Allows hardware manufacturers a pool of existing content to use with their new devices



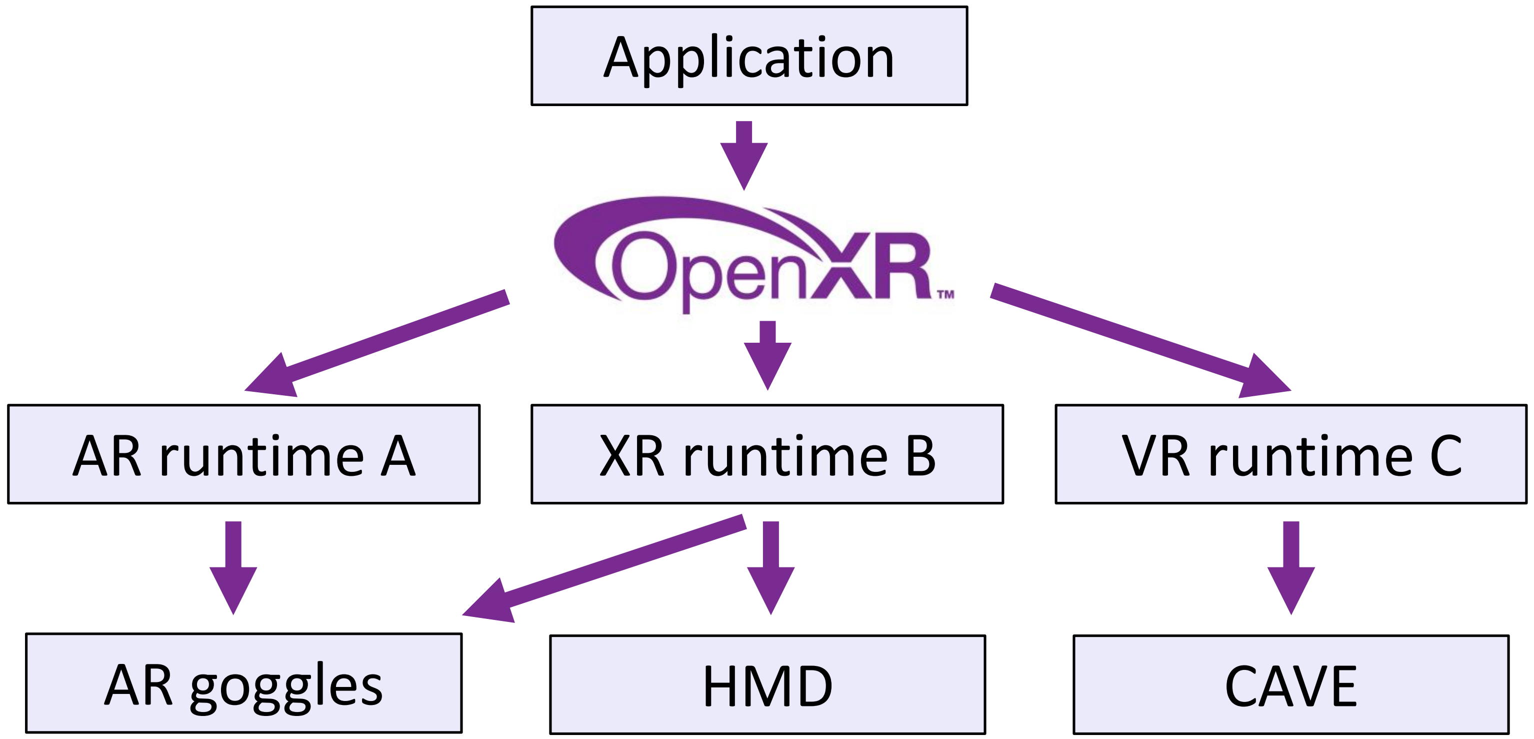
Where Do We Go From Here?

Device Plugin (post 1.0)

- Allows new devices to be used with existing Runtimes
  - New HMDs?
  - New Controllers?
  - New Haptic Devices?
- Optional feature
- After OpenXR 1.0



OpenXR: Long-term vision



Questions?