



# **Khronos Graphics, Compute and Vision APIs**

## **Including Vulkan**

### **Next Generation GPU Acceleration**

Neil Trevett | Khronos President  
NVIDIA Vice President Mobile Ecosystem  
SIGGRAPH Asia | November 2015

# Khronos Connects Software to Silicon

Industry Consortium creating **OPEN STANDARD APIs** for hardware acceleration  
Any company is welcome - one company one vote

ROYALTY-FREE specifications  
State-of-the art IP framework protects  
members AND the standards

Software



Conformance Tests and Adopters  
Programs for specification integrity  
and cross-vendor portability

Silicon

Low-level silicon APIs  
needed on almost every platform:  
graphics, parallel compute,  
rich media, vision, sensor  
and camera processing

International, non-profit organization  
Membership and Adopters fees cover  
operating and engineering expenses

Strong industry momentum

100s of man years invested by industry experts

**Well over a *BILLION* people use Khronos APIs *Every Day...***



ERICSSON



harmonic



### BOARD OF PROMOTERS



Over 100 members worldwide  
any company is welcome to join





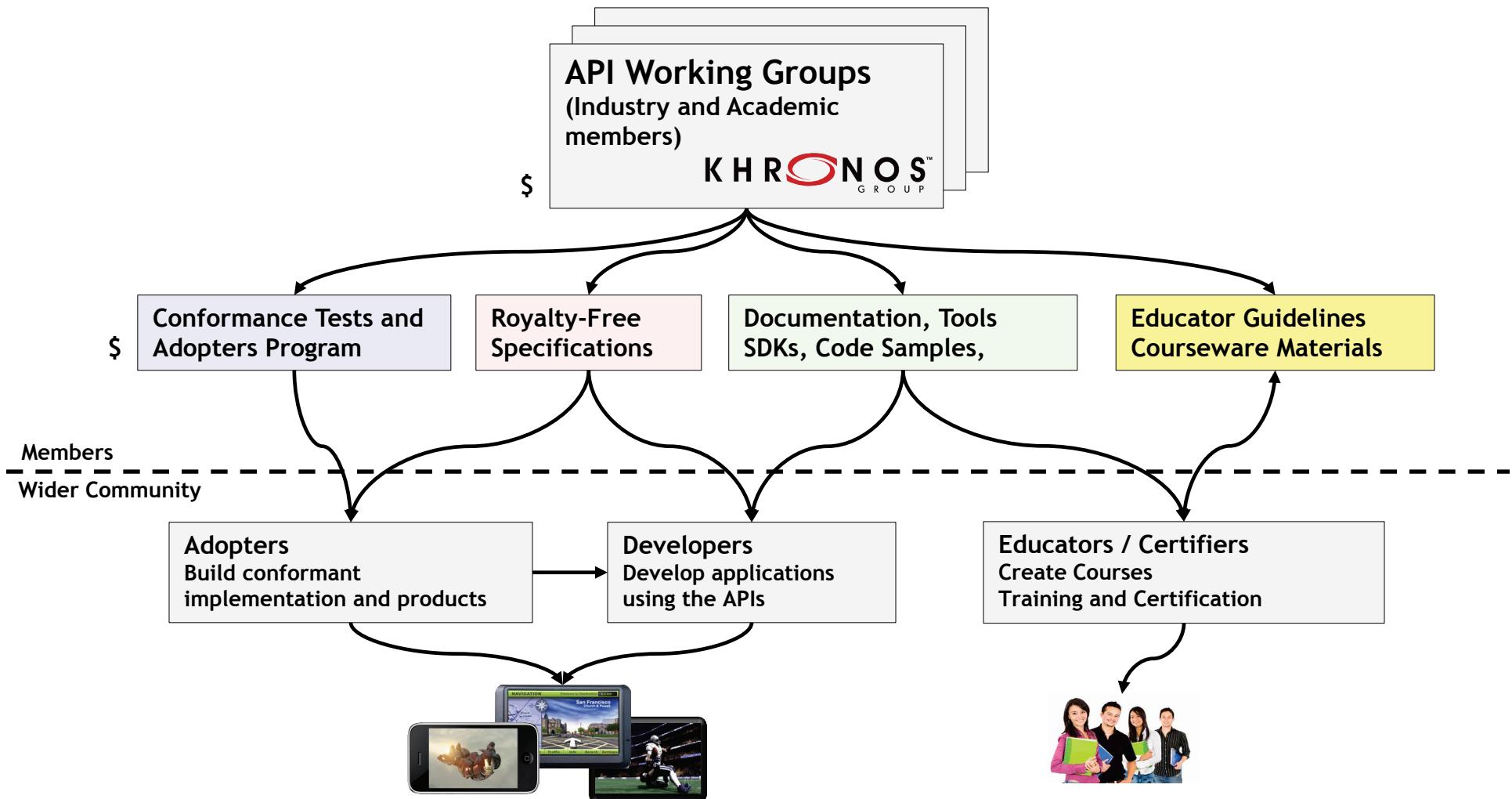
KHRONOS  
GROUP

# OPENROAD

HOW KHRONOS OPEN STANDARDS  
**ACCELERATE YOUR WORLD**

<http://accelerateyourworld.org/>

# Khronos Cooperative Framework



# Khronos Standards for Advanced Processing



Low-power vision processing  
for tracking, odometry and  
scene analysis



3D File formats for  
AUTHORING and TRANSMISSION  
of 3D runtime assets



Image: ScreenMedia



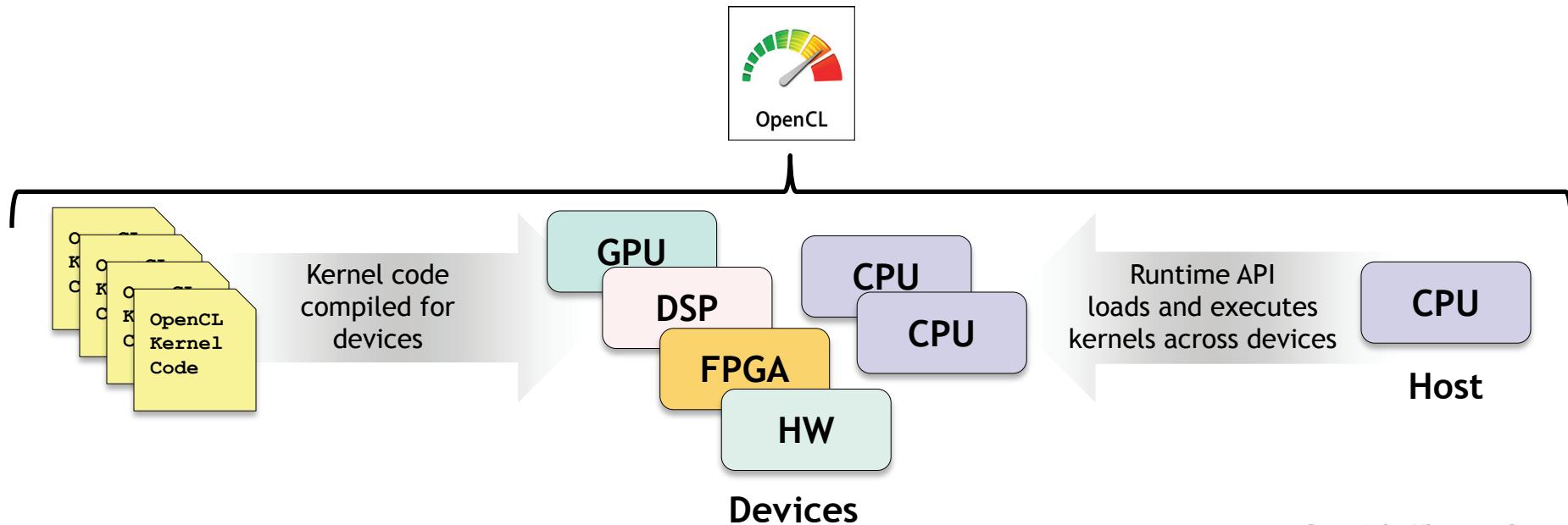
3D Graphics for  
Portable display of augmentations  
and visualizations on every platform



Heterogeneous Processing Acceleration  
e.g. Neural Net Processing for scene understanding

# OpenCL - Portable Heterogeneous Computing

- OpenCL = Two APIs and Two Kernel languages
  - C Platform Layer API to query, select and initialize compute devices
  - OpenCL C and (soon) OpenCL C++ kernel languages to write parallel code
  - C Runtime API to build and execute kernels across multiple devices
- One code tree can be executed on CPUs, GPUs, DSPs, FPGAs and hardware
  - Dynamically balance work across available processors



# OpenCL Momentum

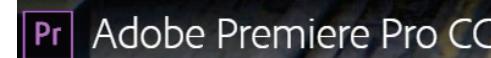
- Broad commercial uptake of OpenCL
  - Imaging, video, vision, simulation
  - Adobe, Apple, SONY, Corel, ArcSoft
  - Dassault, Houdini, Mathematica, MAYA...
- “OpenCL” on Sourceforge, Github, Google Code, Bitbucket finds over 2,000 projects
  - OpenCL implementations - Beignet, pocl
  - VLC, X264, FFmpeg, Handbrake
  - GIMP, ImageMagick, IrfanView
  - Hadoop, Memcached
  - WinZip, Crypto++ Etc. Etc.
- Desktop benchmarks use OpenCL
  - PCMark 8 - video chat and edit
  - Basemark CL, CompuBench Desktop

<https://www.khronos.org/opencl/resources/opencl-applications-using-opencl>

**WOLFRAM**



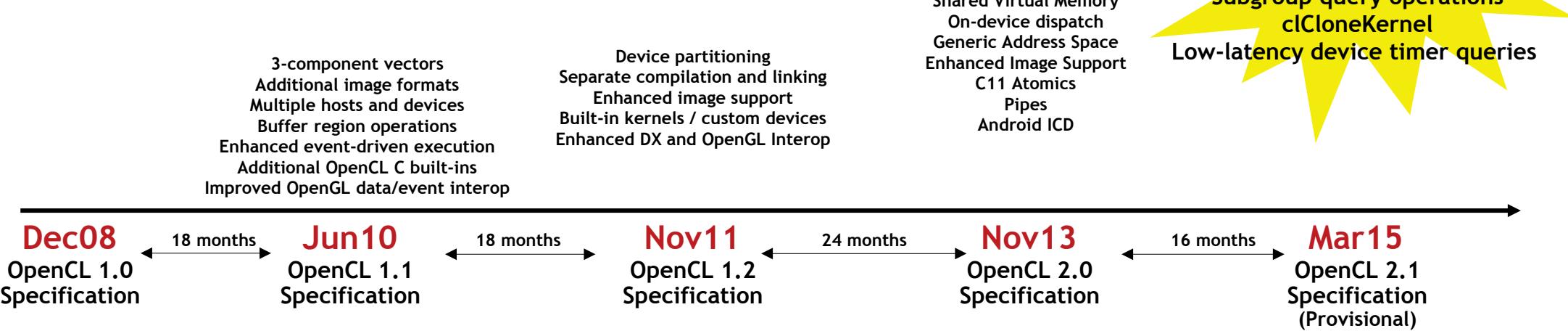
**CyberLink**



AUTODESK® **MAYA®**

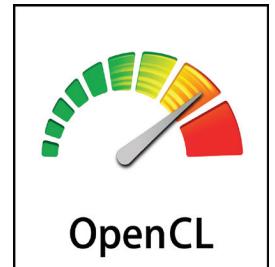
# OpenCL 2.1 Provisional - March 2015

- Support for the new Khronos SPIR-V intermediate language in core
  - E.g. SPIR-V used to ingest from C++ front-end - no C++ compiler in driver
  - OpenCL C ingestion still supported to preserve kernel code investment
- New OpenCL C++ kernel language based on a subset of C++14
  - Significantly enhanced programmer productivity and code performance
- Runs on any OpenCL 2.0-capable hardware
  - Only driver update required



# OpenCL 2.1 API Enhancements

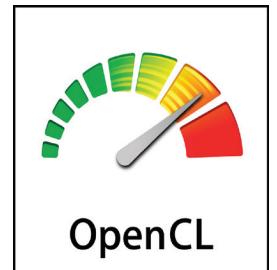
- **clCreateProgramWithIL**
  - SPIR-V support built-in to the runtime
- **Subgroup query operations**
  - Subgroups expose hardware threading in the core feature set
- **clCloneKernel enables copying of kernel objects and state**
  - Safe implementation of copy constructors in wrapper classes
- **Low-latency device timer queries**
  - Support alignment of profiling between device and host code
- **Priority and throttle hint extensions for queues**
  - Specify execution priority on a per-queue basis
- **Zero-size enqueue**
  - Zero-sized dispatches are valid from the host



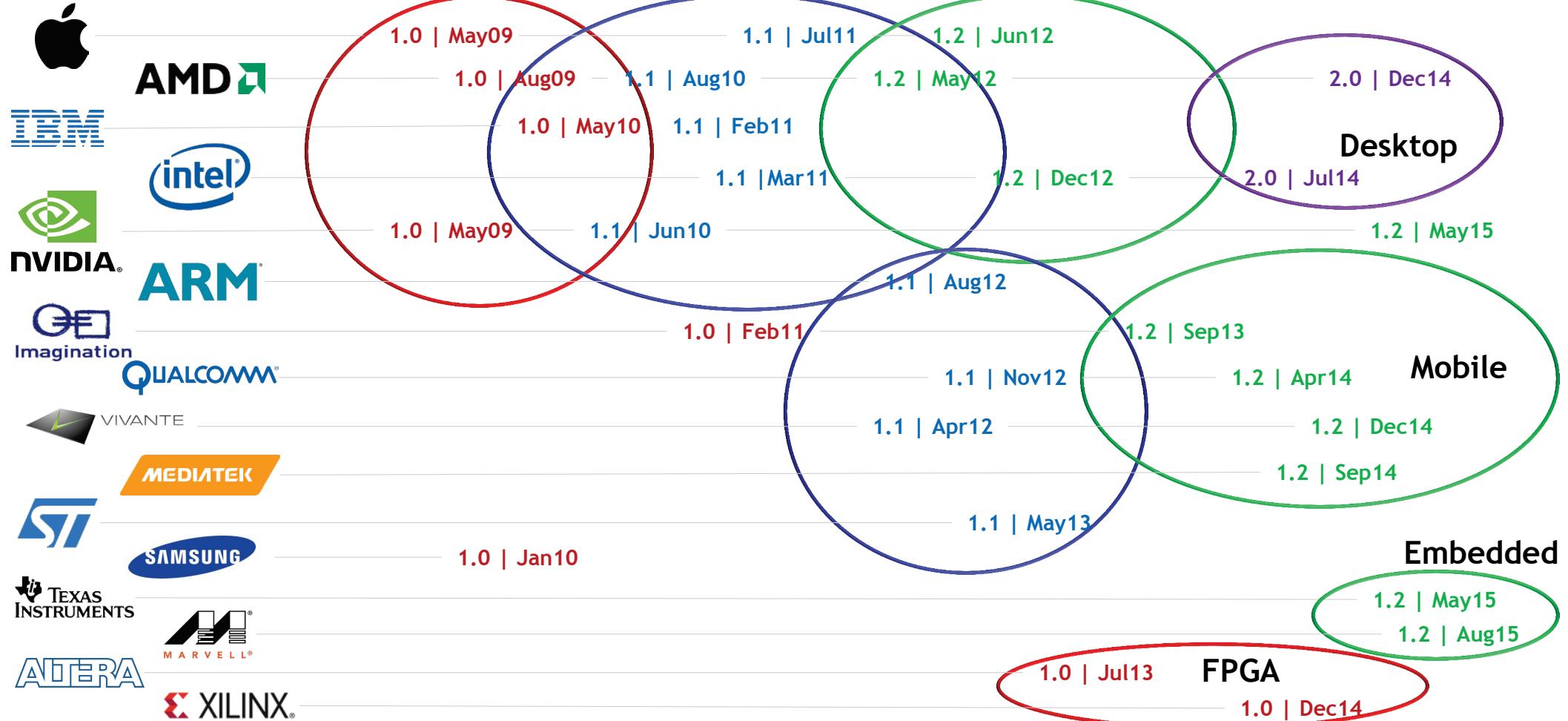
# OpenCL C++

- The OpenCL C++ kernel language is a static subset of C++14
  - Frees developers from low-level coding details without sacrificing performance
- C++14 features removed from OpenCL C++ for parallel programming
  - Exceptions, Allocate/Release memory, Virtual functions and abstract classes Function pointers, Recursion and goto
- Classes, lambda functions, templates, operator overloading etc..
  - Fast and elegant sharable code - reusable device libraries and containers
  - Templates enable meta-programming for highly adaptive software
  - Lambdas used to implement nested/dynamic parallelism
- C++11-based standard library optimized for data-parallel programming
  - Atomics, meta-programming & type traits, math functions...
  - Plus new library features: Work-item & Work-group functions, Dynamic parallelism, Image & Pipe functions...

Highly adaptive parallel software that delivers tuned performance across diverse platforms



# OpenCL Implementations



Vendor timelines are  
first implementation of  
each spec generation

Dec08  
OpenCL 1.0  
Specification

Jun10  
OpenCL 1.1  
Specification

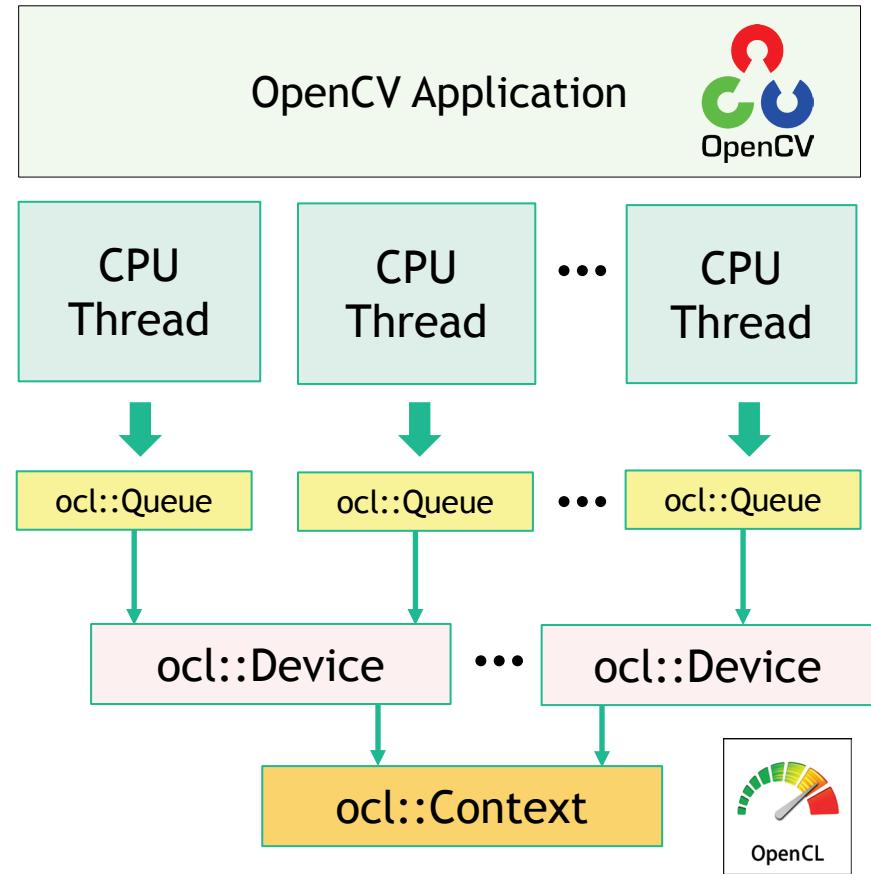
Nov11  
OpenCL 1.2  
Specification

Nov13  
OpenCL 2.0  
Specification

Mar15  
OpenCL 2.1  
Specification

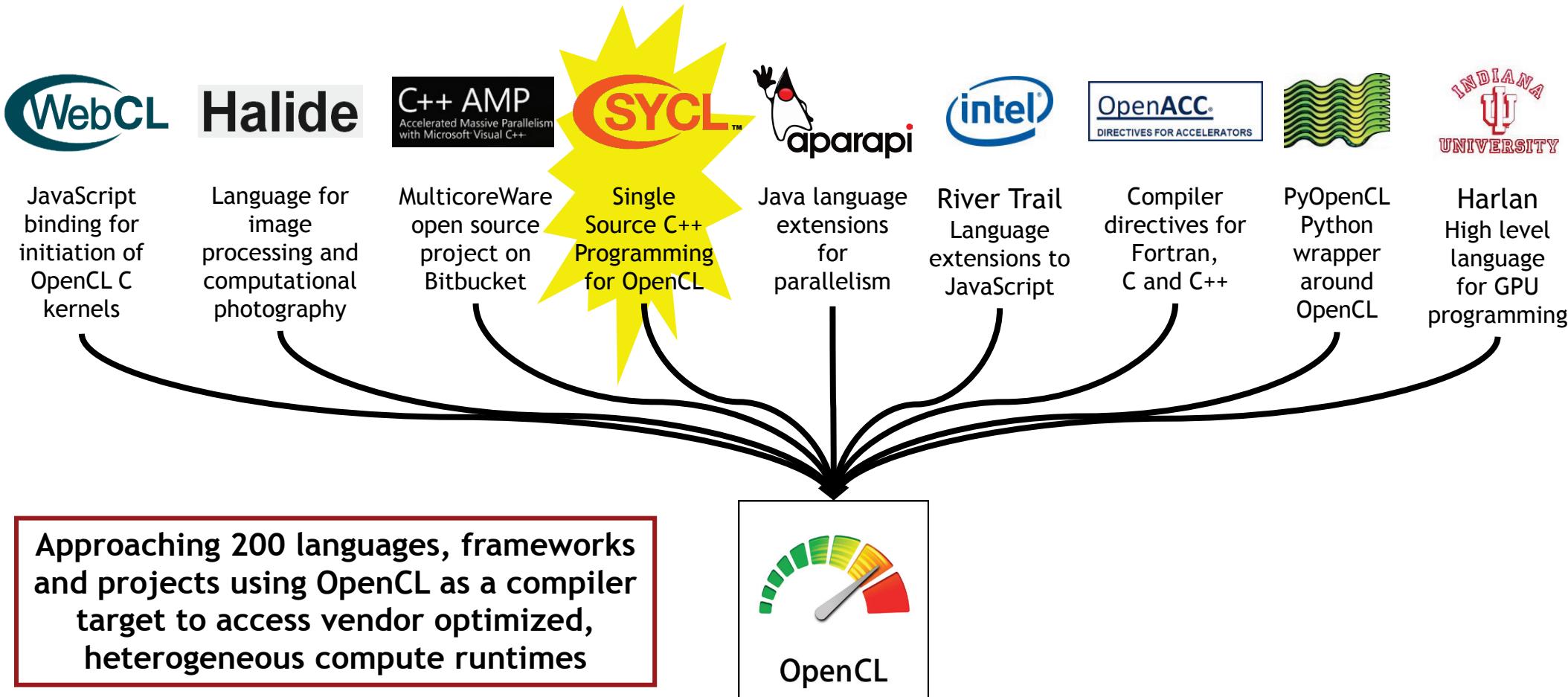
# OpenCL used to Accelerate OpenCV

- Extensive and widely used open source vision library - 1,000s of functions
  - Released under a free-use BSD license
  - Written in optimized C/C++
- C++, C, Python and Java interfaces
  - Windows, Linux, Mac OS
  - iOS and Android
- Increasingly taking advantage of heterogeneous processing using OpenCL
  - OpenCV 3.0 Transparent API - single API entry for each function/algorithm
  - Dynamically loads OpenCL runtime if available; otherwise falls back to CPU code
  - Runtime Dispatching - no recompilation!



- One queue and one OpenCL device per CPU thread
  - Different CPU threads can share a device
    - but use different queues
- OpenCL kernels are executed asynchronously

# OpenCL as Parallel Language Backend

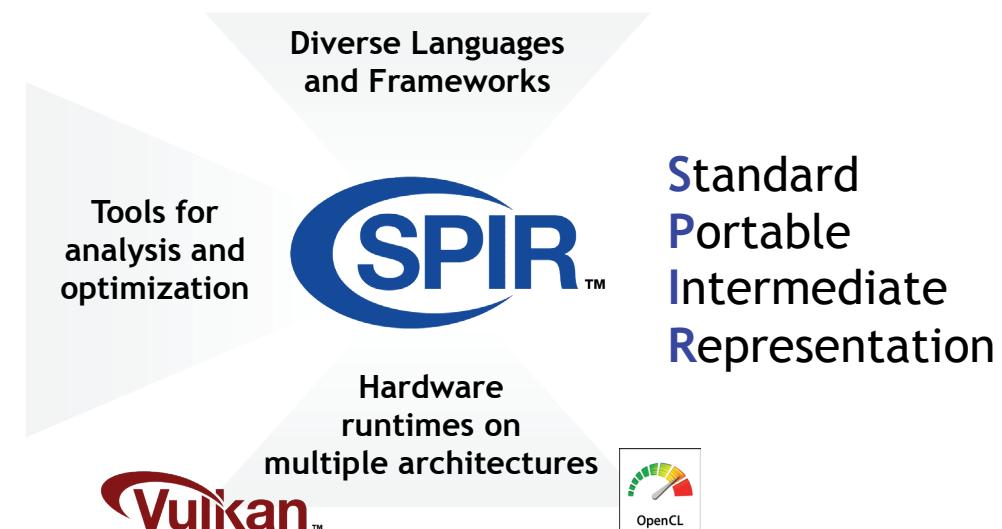


# SPIR-V Transforms the Language Ecosystem

- First multi-API, intermediate language for parallel compute and graphics
  - Native representation for Vulkan shader and OpenCL kernel source languages
  - <https://www.khronos.org/registry/spir-v/papers/WhitePaper.pdf>
- Cross vendor intermediate representation
  - Language front-ends can easily access multiple hardware run-times
  - Acceleration hardware can leverage multiple language front-ends
  - Encourages tools for program analysis and optimization in SPIR form

## Multiple Developer Advantages

Same front-end compiler for multiple platforms  
Reduces runtime kernel compilation time  
Don't have to ship shader/kernel source code  
Drivers are simpler and more reliable

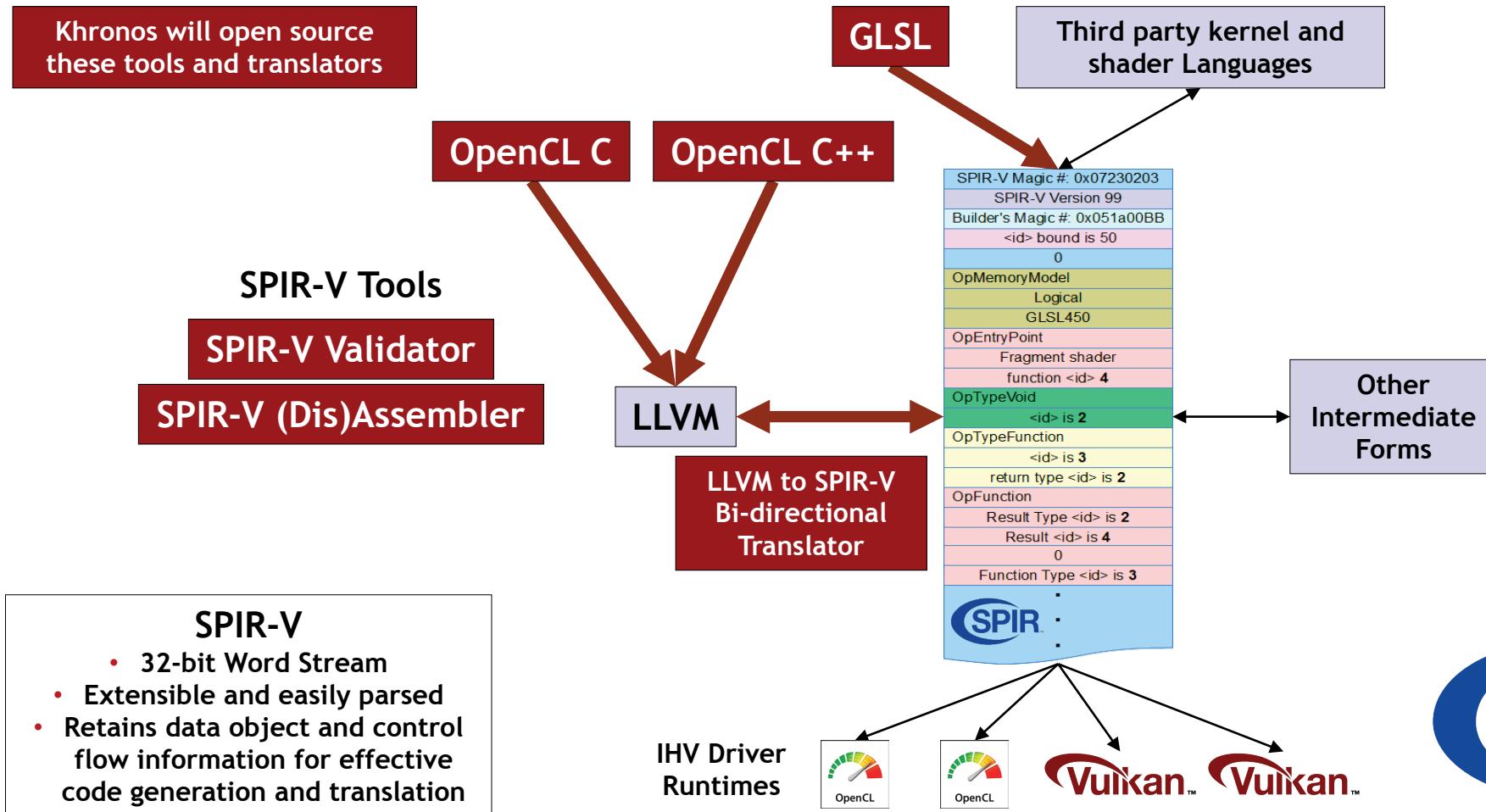


# Evolution of SPIR Family

- SPIR-V is first fully specified Khronos-defined SPIR standard
  - Does not use LLVM to isolate from LLVM roadmap changes
  - Includes full flow control, graphics and parallel constructs beyond LLVM
  - Khronos will open source SPIR-V <-> LLVM conversion tools

SPIR™	SPIR 1.2	SPIR 2.0	SPIR-V 1.0
LLVM Interaction	Uses LLVM 3.2	Uses LLVM 3.4	100% Khronos defined Round-trip lossless conversion
Compute Constructs	Metadata/Intrinsics	Metadata/Intrinsics	Native
Graphics Constructs	No	No	Native
Supported Language Feature Set	OpenCL C 1.2	OpenCL C 1.2 OpenCL C 2.0	OpenCL C 1.2 / 2.0 OpenCL C++ GLSL
OpenCL Ingestion	OpenCL 1.2 Extension	OpenCL 2.0 Extension	OpenCL 2.1 Core
Vulkan Ingestion	-	-	Vulkan Core

# Driving the SPIR-V Open Source Ecosystem



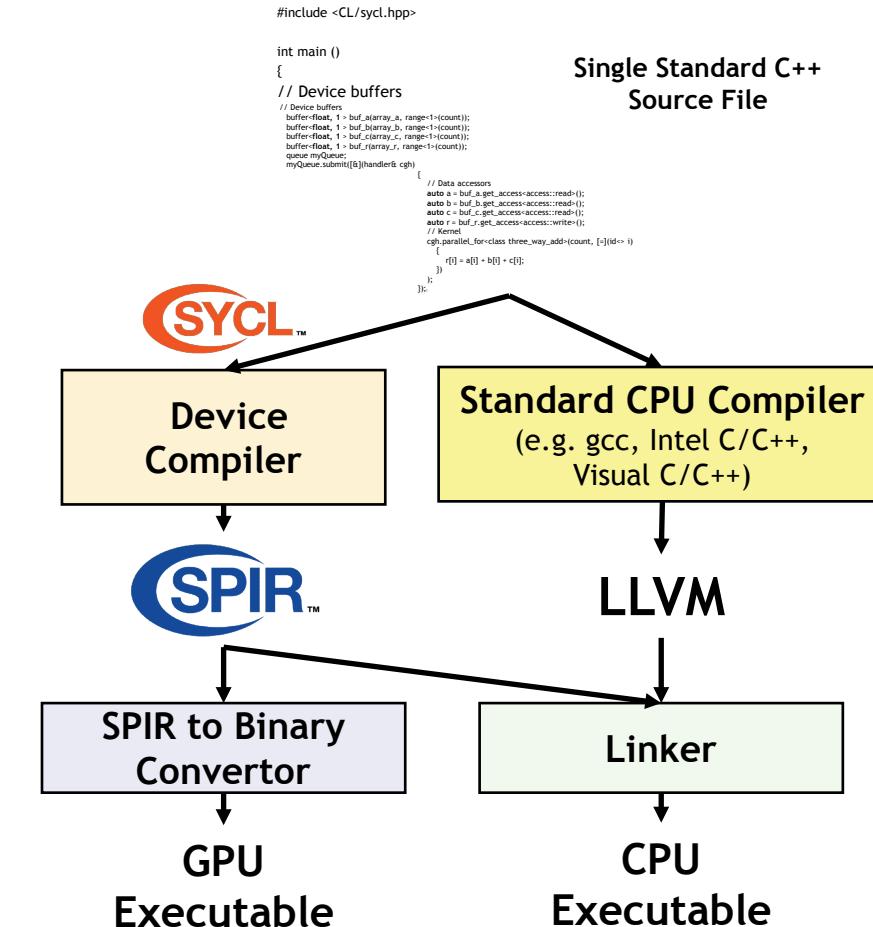
# SPIR-V Open Source Community Activity

- Python byte code to SPIR-V Convertor
  - Write shaders or kernels in Python, Encode and decode SPIR-V in Python
  - Dis(Assembler) with high level human readable assembler syntax
- .NET IL to SPIR-V Convertor
  - Write and debug shaders or kernels using C# , SPIR-V interpreter
- Shade SPIR-V virtual machine
  - Test and debug SPIR-V binaries for binary correctness in human readable format
- Otherside SPIR-V virtual machine
  - Academic software rasterizer project to produce C code from SPIR-V
- Rust (Dis)Assembler
  - Encode and decode SPIR-V binaries in Rust
- Go (Dis)Assembler
  - Encode and decode SPIR-V in Go, SPIR-V represented in Go data structures
- Haskell EDSL
  - SPIR-V like language embedded in Haskell with significantly relaxed layout constraints
- Lisp SPIR-V Specification
  - Lisp readable SPIR-V specification
- JSON SPIR-V specification
  - Conversion of HTML SPIR-V specification to JSON format
- This is just the start....



# SYCL - Single Source Heterogeneous C++

- Pronounced ‘sickle’
  - To go with ‘spear’ (SPIR)
- C++11 code for multiple OpenCL devices
  - Construct complex reusable algorithm templates using OpenCL for acceleration
- C++ templates contain host & device code
  - e.g. `parallel_sort<MyType>(myData);`
- Cross-toolchain as well as cross-platform
  - No language extensions - so standard C++ compilers can process SYCL source
- Device compilers enable SYCL on devices
  - Can have multiple device compilers linking into final executable



# SYCL Status and Benefits

- SYCL 1.2 Final spec released
  - At IWOCL in May 2014
- Multiple implementations
  - Including open source triSYCL from AMD
  - <https://github.com/amd/triSYCL>
- Developers can move quickly into writing SYCL code
  - Provides methods for dealing with targets that do not have OpenCL(yet!)
- A fallback CPU implementation is debuggable!
  - Using normal C++ debuggers
  - Profiling tools also work on CPU device
- Huge bonus for productivity and adoption
  - Cost of entry to use SYCL very low



*SYCL is a practical, open, royalty-free standard to deliver high performance software on today's highly-parallel systems*

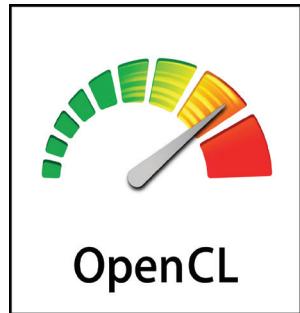
# OpenCL Ecosystem

## Implementers

Desktop/Mobile/FPGA



Single Source C++ Programming



Core API and Language Specs



Portable Kernel Intermediate Language

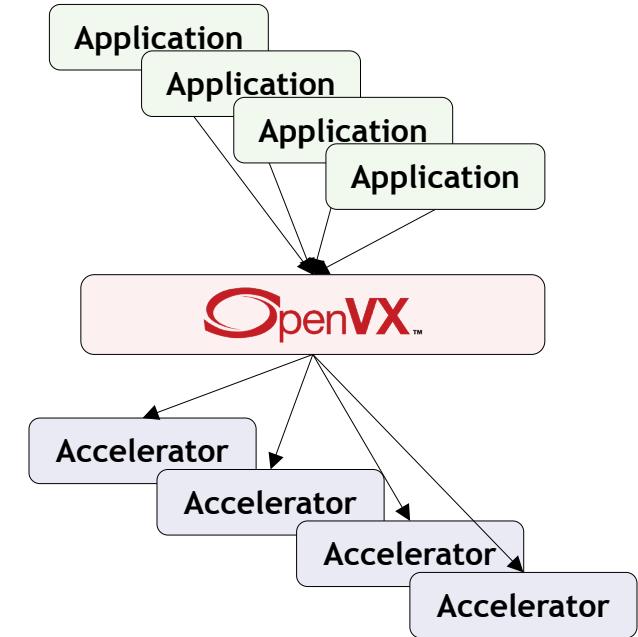
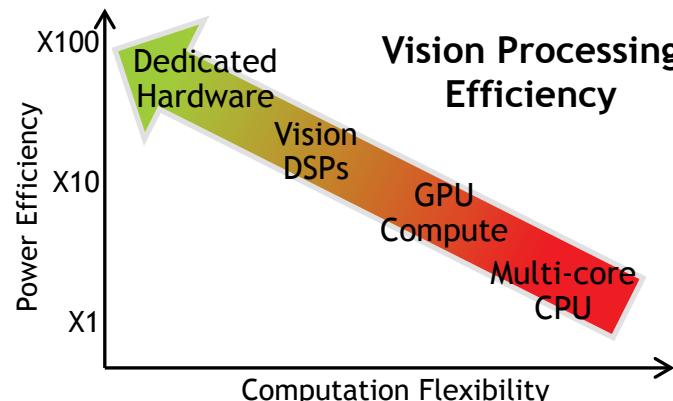
## Working Group Members

Apps/Tools/Tests/Courseware



# OpenVX - Vision Acceleration

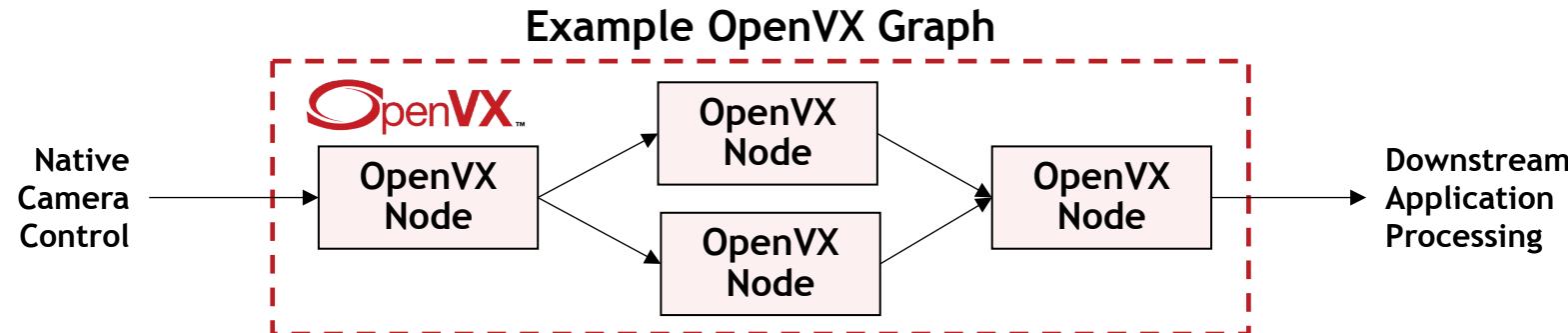
- Targeted at low-power, real-time applications
  - Mobile and embedded platforms
- Portability across diverse heterogeneous processors
  - Multi-core CPUs, GPUs, DSPs and DSP arrays
  - ISPs, Dedicated hardware...
  - Lower precision requirements than OpenCL
- Doesn't require high-power CPU/GPU Complex
  - Low-power host can setup and manage frame-rate vision processing graph



OpenVX extends easily re-usable vision acceleration to very low power domains

# OpenVX Graphs - The Key to Efficiency

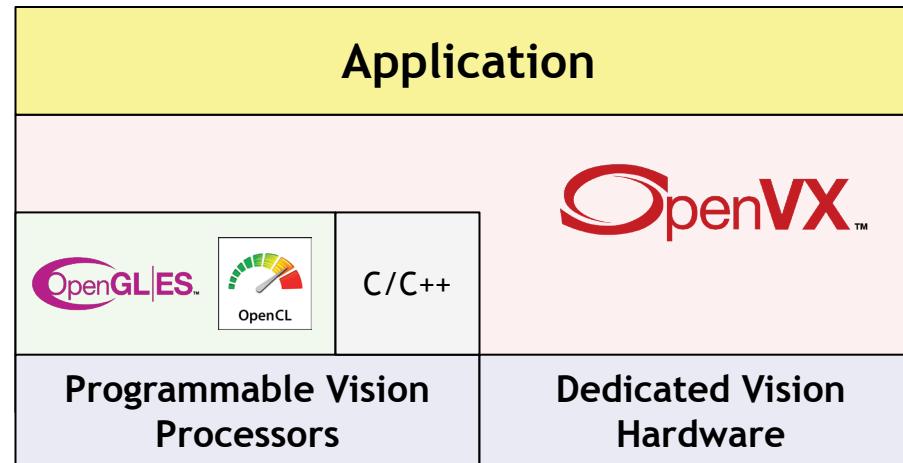
- OpenVX developers express a graph of image operations ('Nodes')
  - Nodes can be on any hardware or processor coded in any language
  - E.g. on GPU nodes may implemented in OpenCL
- Graph enables implementations to optimize for power and performance
  - E.g. Nodes may be fused by the implementation to eliminate memory transfers
  - E.g. Processing can be tiled to keep data entirely in local memory/cache
- Minimizes host interaction during frame-rate graph execution
  - Host processor can setup graph which can then execute almost autonomously



# Layered Vision Processing Ecosystem

- Lower-level compute APIs *can be used to implement* OpenVX nodes
  - Depending on the available processors
  - E.g. use OpenCL or OpenGL Compute Shaders
- Developers can then *use* OpenVX to easily connect those nodes into a graph
  - With portable performance
- High-level graph abstraction gives implementation flexibility
  - And significant optimization opportunities

Application Software  
—  
OpenVX Runtime  
—  
Powerful, flexible  
low-level APIs / languages  
—  
Processor Hardware



# OpenVX and OpenCV are Complementary

	 OpenCV	 OpenVX™
<b>Implementation</b>	Community driven open source library	Open standard API designed to be implemented by hardware vendors
<b>Conformance</b>	Extensive OpenCV Test Suite but no formal Adopters program	Implementations must pass defined conformance test suite to use trademark
<b>Consistency</b>	Available functions can vary depending on implementation / platform	All core functions must be available in all conformant implementations
<b>Scope</b>	Very wide 1000s of imaging and vision functions Multiple camera APIs/interfaces	Tight focus on core hardware accelerated functions for mobile vision - but extensible Uses external/native camera API
<b>Embedded Deployment</b>	Re-usable code	Callable library
<b>Efficiency</b>	Memory-based architecture Each operation reads and writes to memory	Graph-based execution Optimizable computation and data transfer
<b>Typical Use Case</b>	Rapid experimentation and prototyping - especially on desktop	Production development & deployment on wide range of mobile and embedded devices

# OpenVX Status

- Finalized OpenVX 1.0 specification released October 2014
  - OpenVX 1.0.1 spec maintenance update released June 2015 [www.khronos.org/openvx](http://www.khronos.org/openvx)
- Khronos open source sample implementation of OpenVX 1.0 released
  - [https://www.khronos.org/registry/vx/sample/openvx\\_sample\\_20141217.tar.gz](https://www.khronos.org/registry/vx/sample/openvx_sample_20141217.tar.gz)
- Full conformance test suite and Adopters Program available
  - Test suite exercises graph framework and functionality of each OpenVX 1.0 node
- Commercial conformant products
  - Intel, Imagination, NVIDIA, Synopsis, Vivante and many more coming...



# Access to 3D on Over 2 BILLION Devices



300M Desktops / year  
Windows, Mac, Linux



Worldwide Device Shipments by Segment (Thousands of Units)				
Device Type	2012	2013	2014	2015
PC (Desk-Based and Notebook)	341,273	299,342	277,939	268,491
Tablet (Ultramobile)	119,529	179,531	263,450	324,565
Mobile Phone	1,746,177	1,804,334	1,893,425	1,964,788
Other Ultramobiles (Hybrid and Clamshell)	9,344	17,195	39,636	63,835
<b>Total</b>	<b>2,216,322</b>	<b>2,300,402</b>	<b>2,474,451</b>	<b>2,621,678</b>

Source: Gartner (December 2013)

Source: Gartner (December 2013)



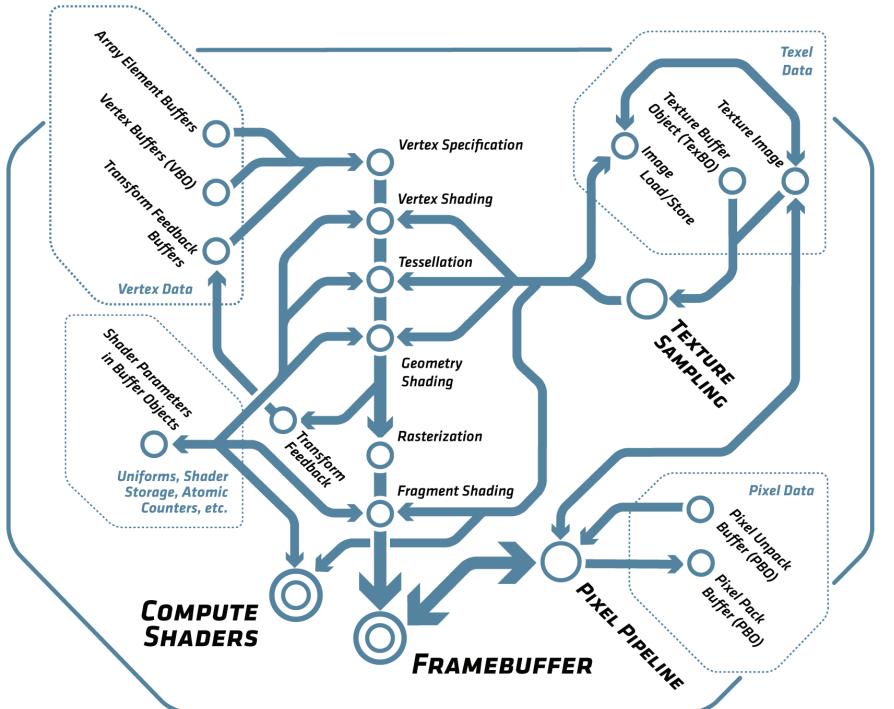
1.7B Mobiles / year



1B Browsers / year



# Continuing OpenGL Innovation



OpenGL 2.0

OpenGL 2.1

OpenGL 3.0

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

DirectX  
9.0c

DirectX  
10.0

DirectX  
10.1

DirectX  
11

DirectX  
11.1

DirectX  
11.2

Bringing state-of-the-art  
functionality to cross-  
platform graphics

Direct State Access (DSA)  
Flush Control  
Robustness  
DX11 emulation  
OpenGL ES 3.1  
compatibility

OpenGL 4.5

OpenGL 4.4

OpenGL 4.3

OpenGL 4.2

OpenGL 4.1

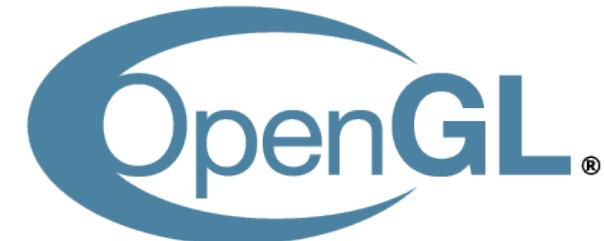
OpenGL 3.3/4.0

OpenGL 3.2

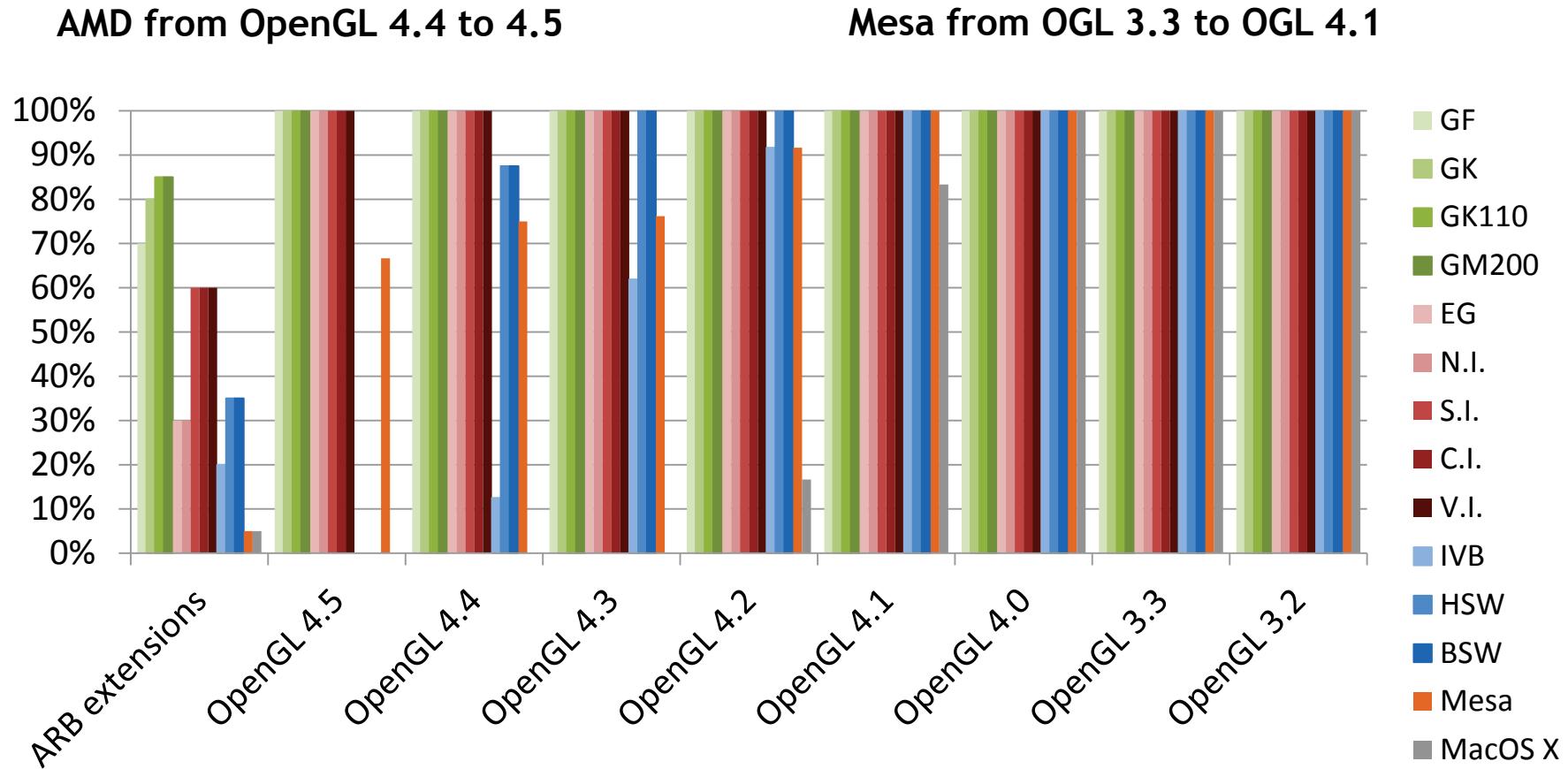
OpenGL 3.1

# 13 New OpenGL Extensions at SIGGRAPH 2015

- Expose cutting-edge desktop graphics
  - Paving the way for new core OpenGL when functionality pervasively available
- Streamlined sparse textures
  - Manage multisample sparse textures and uncommitted/unpopulated areas
- Enhanced shaders
  - Interlocks for multi-pass algorithms, 64-bit integer handling
  - Early fragment testing, enhanced atomic counters, 64-bit timing counter
- Multi-thread shader compilation
  - Faster application loading
- Modifiable locations of multi-samples within a pixel
  - Increase antialiasing quality
- OpenGL ES 3.2 compatibility
  - Use OpenGL to develop OpenGL ES 3.2 applications



# OpenGL Driver Support since 2014



Intel added OGL 4.3 support for IVB  
Significant progress on OGL 4.4

NVIDIA added support for 13 ARB extensions at launch  
Supported OpenGL 4.5 since SIGGRAPH 2014

# OpenGL ES Roadmap

Fixed function  
Pipeline



Programmable  
Vertex and  
fragment shaders



32-bit integers and floats  
NPOT, 3D/depth textures  
Texture arrays  
Multiple Render Targets



Compute Shaders



Tessellation and geometry shaders  
ASTC Texture Compression  
Floating point render targets  
Debug and robustness for security



Epic's Rivalry demo using full Unreal Engine 4  
<https://www.youtube.com/watch?v=jRr-G95GdaM>

Driver  
Update

Silicon  
Update

Silicon  
Update

Driver  
Update

Silicon  
Update

2003 1.0

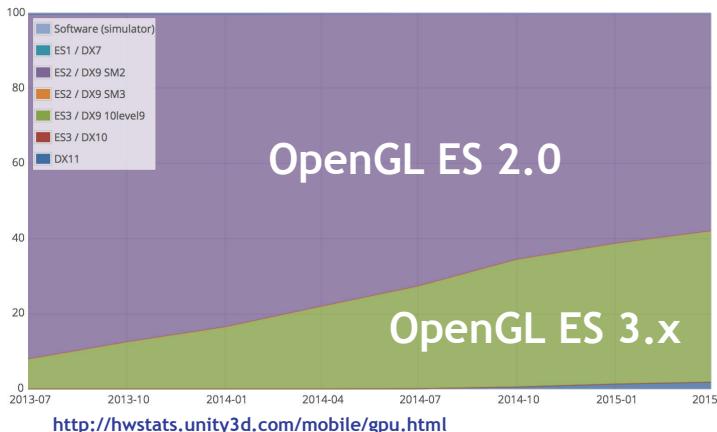
2004 1.1

2007 2.0

2012 3.0

2014 3.1

2015 3.2



The most widely deployed 3D graphics API in history  
Industry will ship >1.7 billion OpenGL ES-enabled devices in 2015

# Next Generation GPU APIs



Only Windows 10



Only Apple



Cross Platform

Any GPU with OpenGL ES 3.1/OpenGL 4.X and up  
Any OS



Vulkan Target Availability



SteamOS



ubuntu



redhat



Vulkan Committed Platform Adoption

# Ground-up Explicit API Redesign

	
Originally architected for graphics workstations with direct renderers and split memory	Matches architecture of modern platforms including mobile platforms with unified memory, tiled rendering
Driver does lots of work: state validation, dependency tracking, error checking. Limits and randomizes performance	Explicit API – the application has direct, predictable control over the operation of the GPU
Threading model doesn't enable generation of graphics commands in parallel to command execution	Multi-core friendly with multiple command buffers that can be created in parallel
Syntax evolved over twenty years – complex API choices can obscure optimal performance path	Removing legacy requirements simplifies API design, reduces specification size and enables clear usage guidance
Shader language compiler built into driver. Only GLSL supported. Have to ship shader source	SPIR-V as compiler target simplifies driver and enables front-end language flexibility and reliability
Despite conformance testing developers must often handle implementation variability between vendors	Simpler API, common language front-ends, more rigorous testing increase cross vendor functional/performance portability

# Vulkan Explicit GPU Control

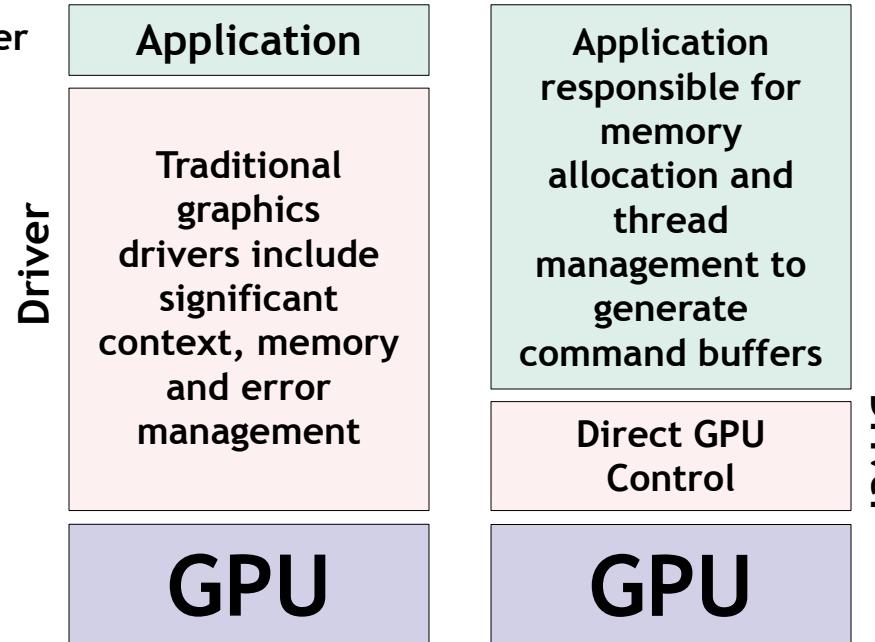


Complex drivers lead to driver overhead and cross vendor unpredictability

Error management is always active

Driver processes full shading language source

Separate APIs for desktop and mobile markets



Simpler drivers for low-overhead efficiency and cross vendor consistency

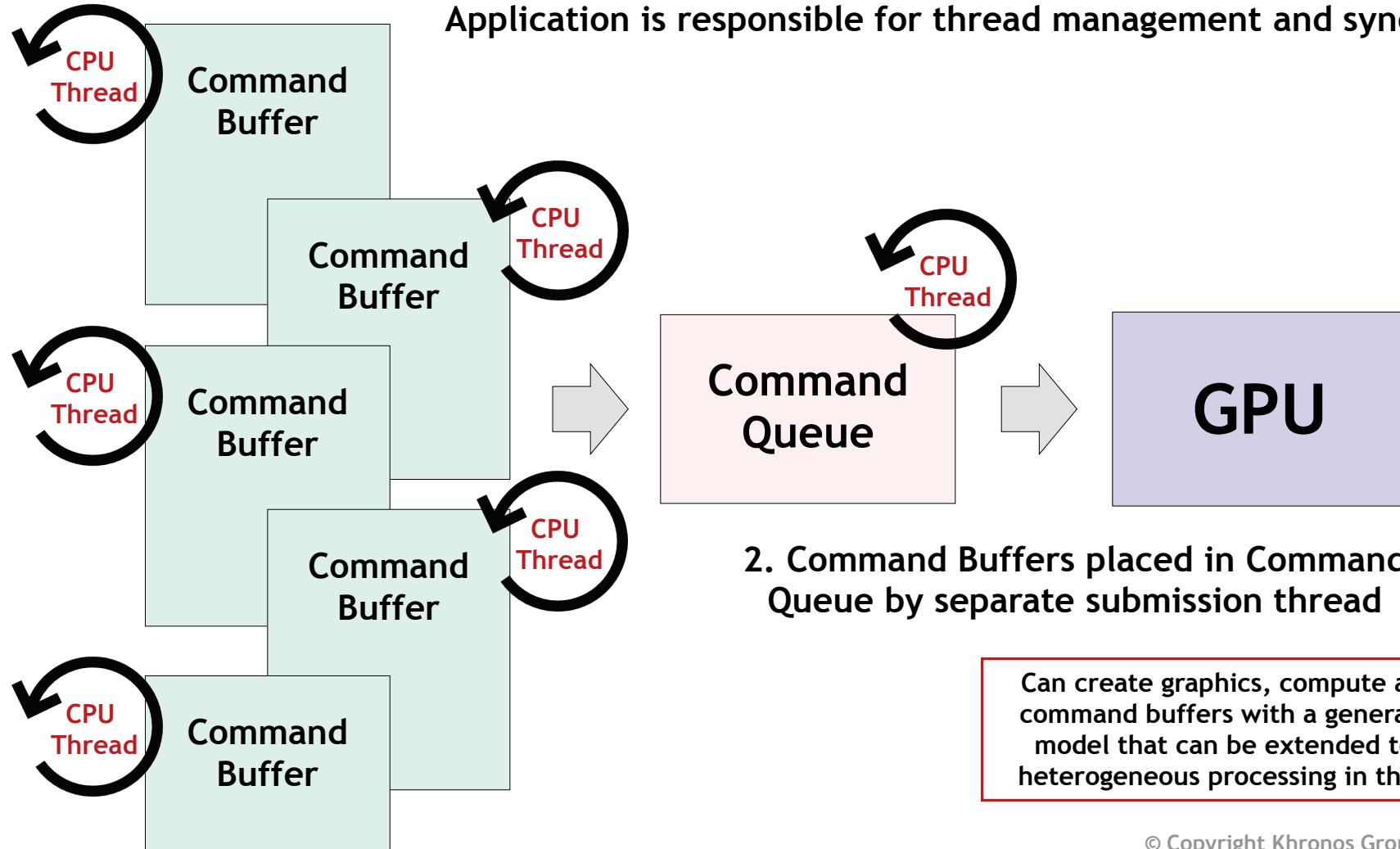
Layered architecture so validation and debug layers can be unloaded when not needed

Run-time only has to ingest SPIR-V intermediate language

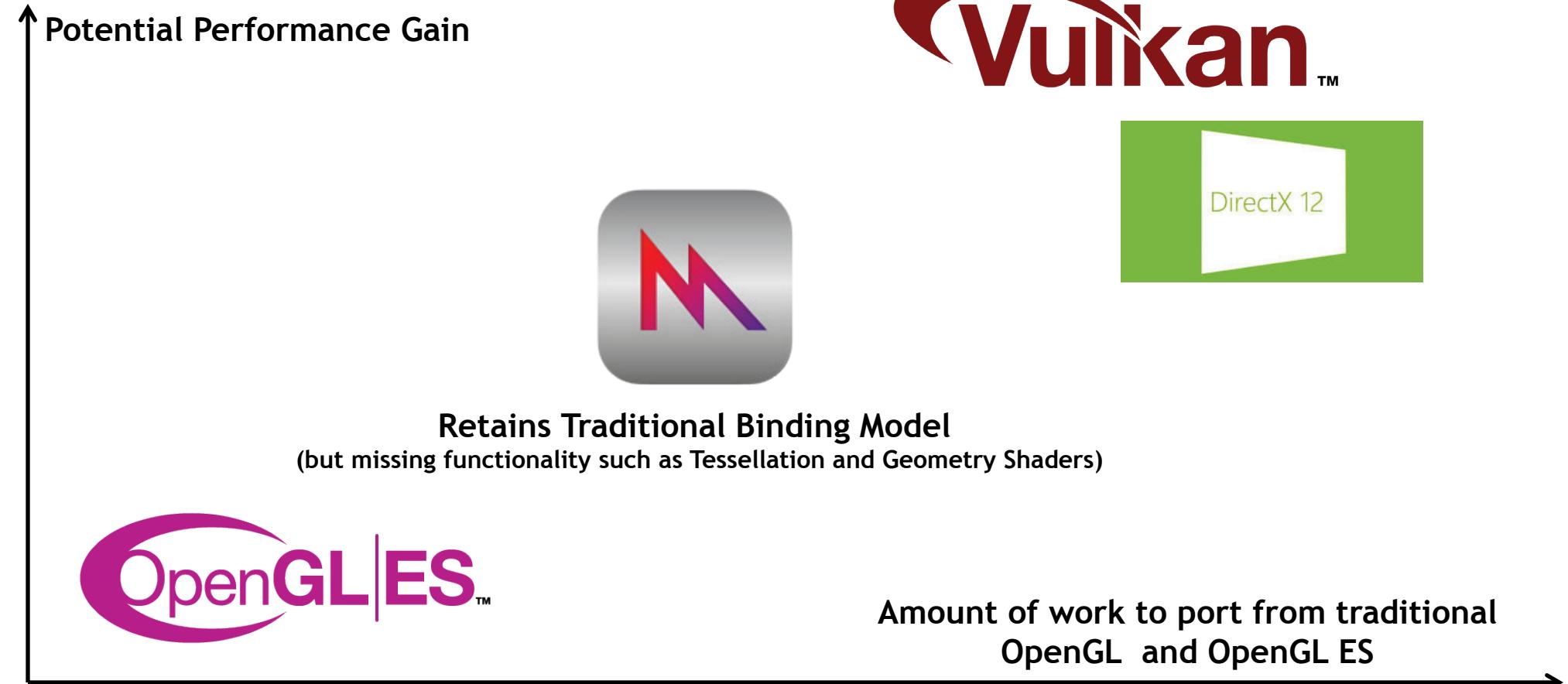
Unified API for mobile, desktop, console and embedded platforms

Vulkan delivers the maximized performance and cross platform portability needed by sophisticated engines, middleware and apps

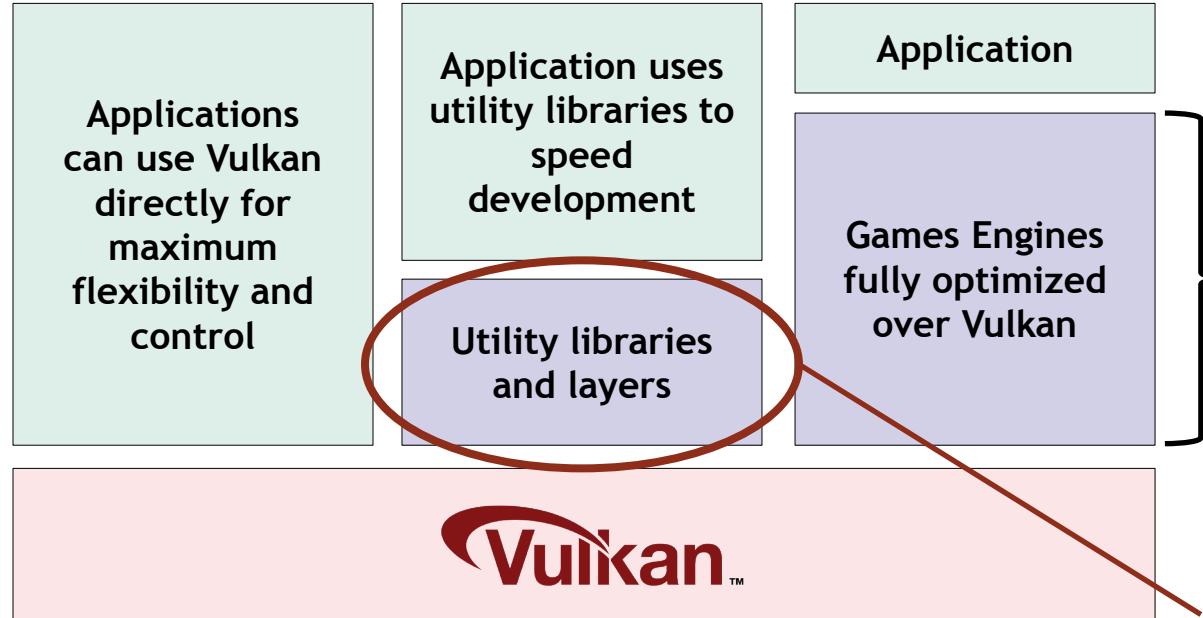
# Vulkan Multi-threading Efficiency



# No Compromise



# The Power of a Three Layer Ecosystem



Developers can choose at which level to use the Vulkan Ecosystem

The industry's leading games and engine vendors are participating in the Vulkan working group



## Rich Area for Innovation

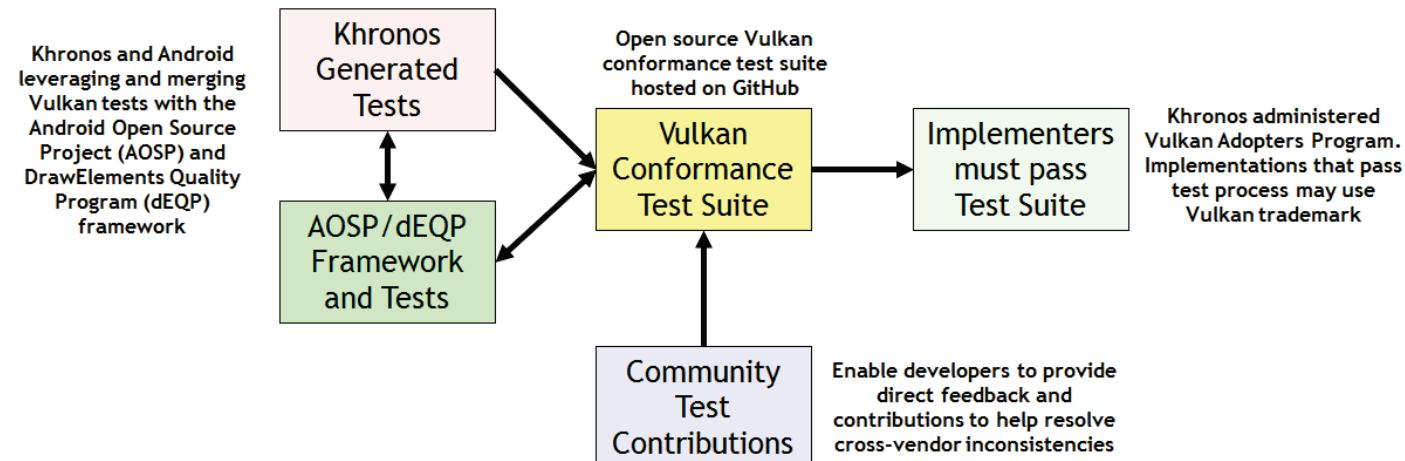
- Many utilities and layers will be in open source
- Layers to ease transition from OpenGL
- Domain specific flexibility

The same ecosystem dynamic as WebGL

A widely pervasive, powerful, flexible foundation layer enables diverse middleware tools and libraries

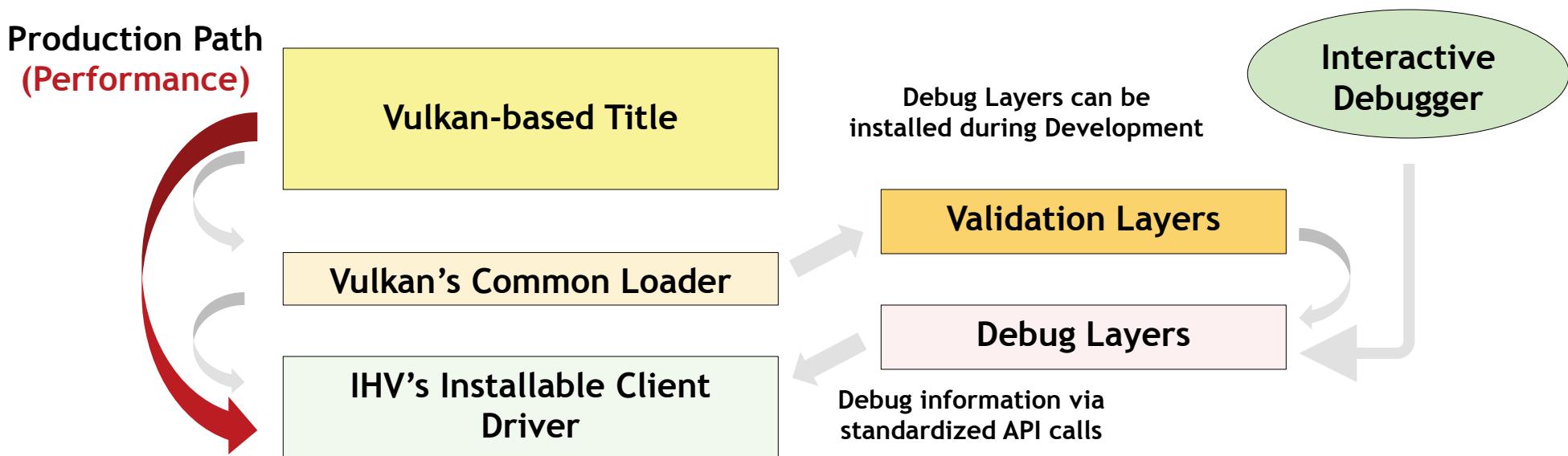
# Developing Ecosystem and Spec in Parallel

- Open sourcing Vulkan test suite to enable developer feedback and contributions
- Khronos supplied loader and layered tools architecture
- Open source layered tools - Valve/LunarG are the first
- Flexible Windows System Integration - working with platform vendors
- Example code, documentation and course notes
- SPIR-V for language innovation



# Vulkan Tools Architecture

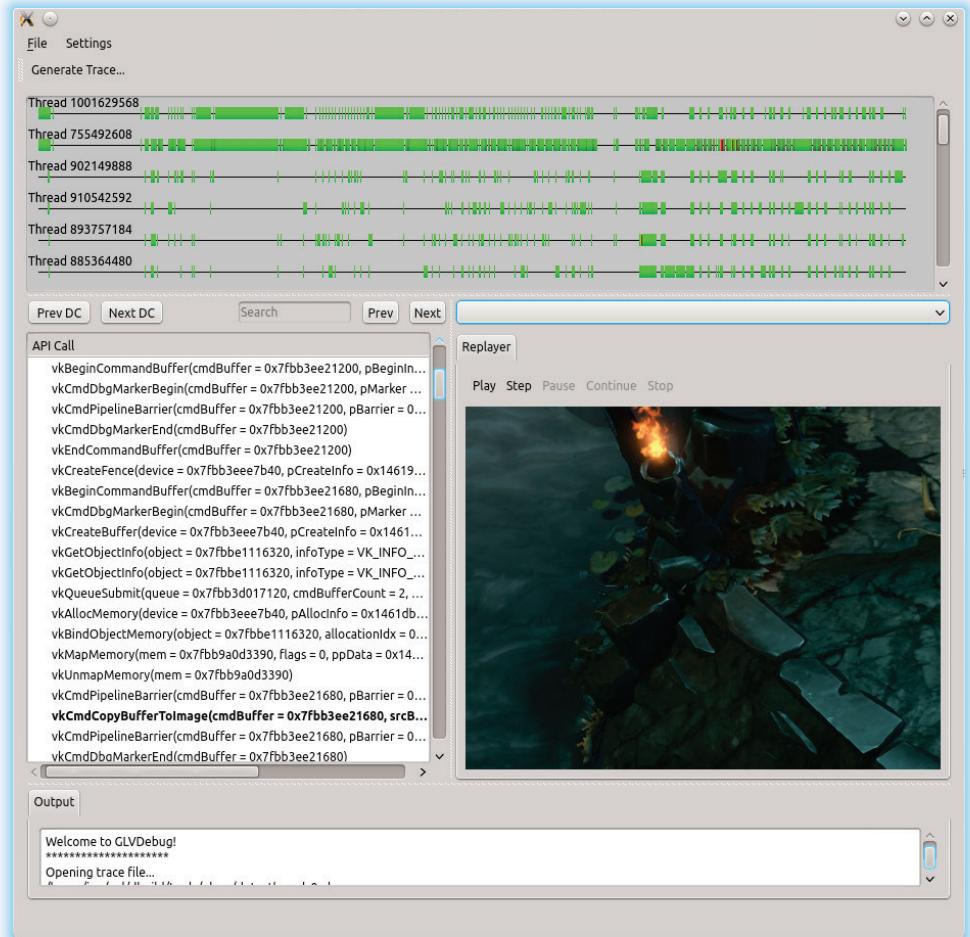
- Layered design for cross-vendor tools innovation and flexibility
  - IHVs plug into a common, extensible architecture for code validation, debugging and profiling during development without impacting production performance
- Khronos Open Source Loader enables use of tools layers during debug
  - Finds and load drivers, dispatches API calls to correct driver and layers



# Vulkan Tools Ecosystem

- Extensible modular architecture  
encourages many fine-grained layers - new  
layers can be added easily
- Khronos encouraging open community of  
tools e.g. shader debugging
- Valve, LunarG, Codeplay and others are  
already driving the development of open  
source Vulkan tools
- Customized interactive debugging and  
validation layers will be available together  
with first drivers

Prototype Vulkan Debugger from Valve and LunarG  
[LunarG.com/Vulkan](http://LunarG.com/Vulkan)



# LunarG Open Source SDK Layers

- Will be released in open source in parallel with Vulkan specification

Layer Name	Description
APIDump	Print API calls and their parameters and values
DrawState	Validate the descriptor set, pipeline state and dynamic state
Image	Validate texture formats and render target formats
MemTracker	Track & validate GPU memory, its binding to objects & command buffers
ObjectTracker	Track all Vulkan objects and flag invalid objects and object memory leaks
ParamChecker	Validate API parameter values
ShaderTracker	Validate the interfaces between SPIR-V modules and the graphics pipeline
Threading	Check validity of multi-threaded API usage

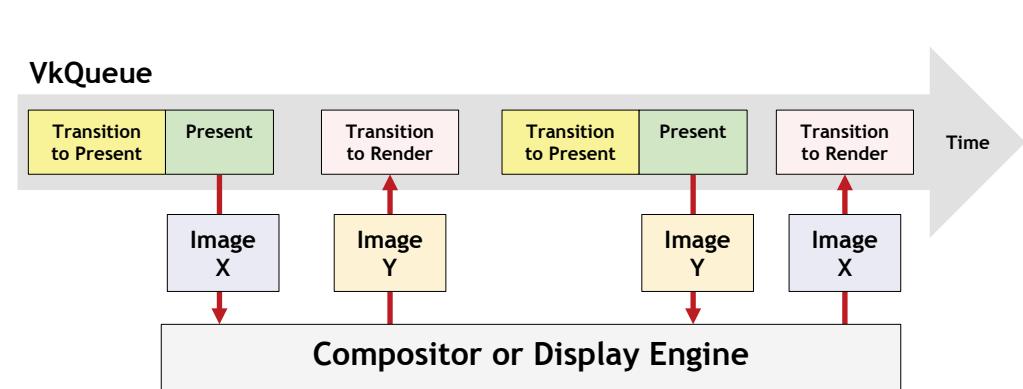
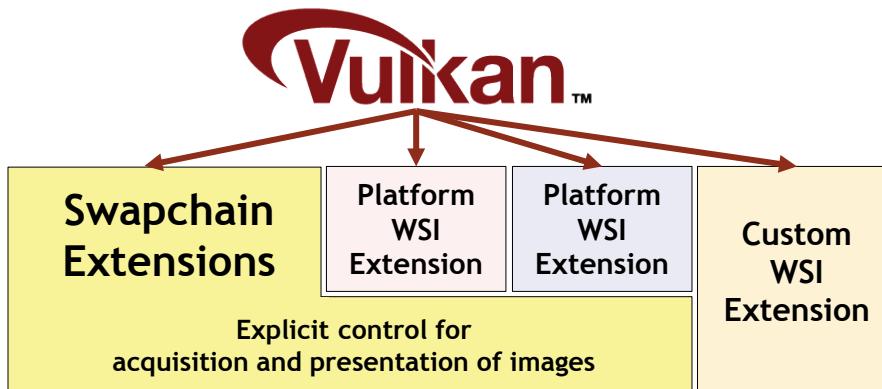
# Vulkan Feature Sets

- Vulkan supports hardware with a wide range of hardware capabilities
  - Mobile OpenGL ES 3.1 up to desktop OpenGL 4.5 and beyond
- One unified API framework for desktop, mobile, console, and embedded
  - No "Vulkan ES" or "Vulkan Desktop"
- Vulkan precisely defines a set of "fine-grained features"
  - Features are specifically enabled at device creation time (similar to extensions)
  - Consistently available on that platform
- Platform owners define a Feature Set for their platform
  - Vulkan provides the mechanism but does not mandate policy
  - Khronos will define Feature Sets for platforms where owner is not engaged



# Vulkan Window System Integration (WSI)

- Explicit control for acquisition and presentation of images
  - Designed to fit the Vulkan API and today's compositing window systems
  - Cleanly separates device creation from window system
- Platform provides an array of persistent presentable images = Vulkan Swapchain
  - Device exposes which queues support presentation
  - Application explicitly controls which image to render and present
- Standardized extensions - unified API for multiple window systems
  - Works across Android, Mir, Windows (Vista and up), Wayland and X (with DRI3)
  - Platforms can extend functionality, define custom WSI stack, or have no display at all



# Vulkan - Enhancing App Portability

SPIR-V intermediate language improves shader program portability and reduces driver complexity

Streamlined API is easier to implement and test

Cross-vendor App Portability



Open source conformance test source components for community engagement

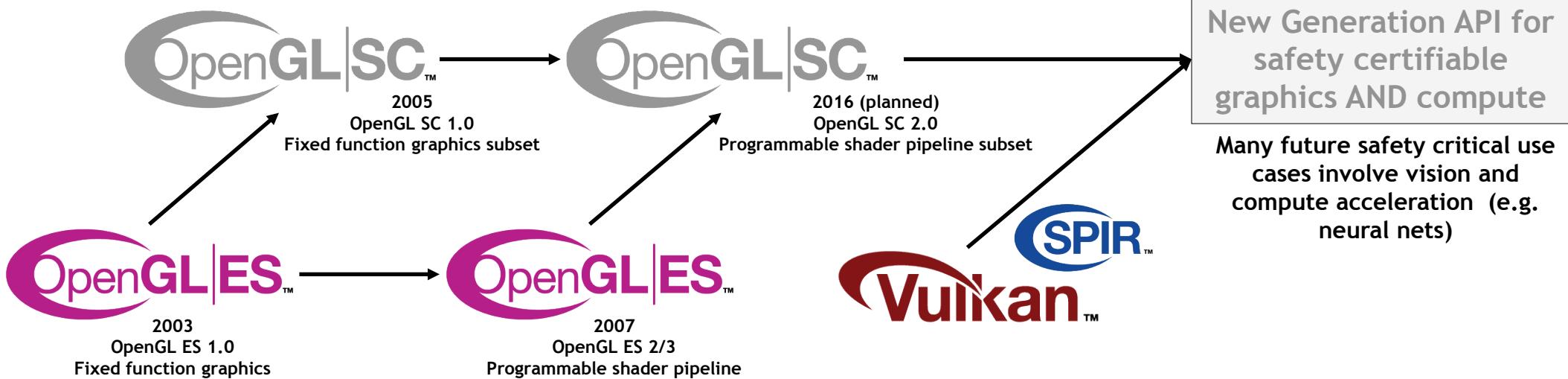
# Vulkan Status

- Rapid progress since project start in June 2014
  - Significant proposals and IP contributions received from members
- Participants come from all segments of the graphics industry
  - Including an unprecedented level of participation from game engine ISVs
- Initial specs and implementations expected this year
  - Driver upgrades will enable Vulkan on lots of current hardware



Working Group Participants

# Safety Critical Working Group



# WebGL Ecosystem

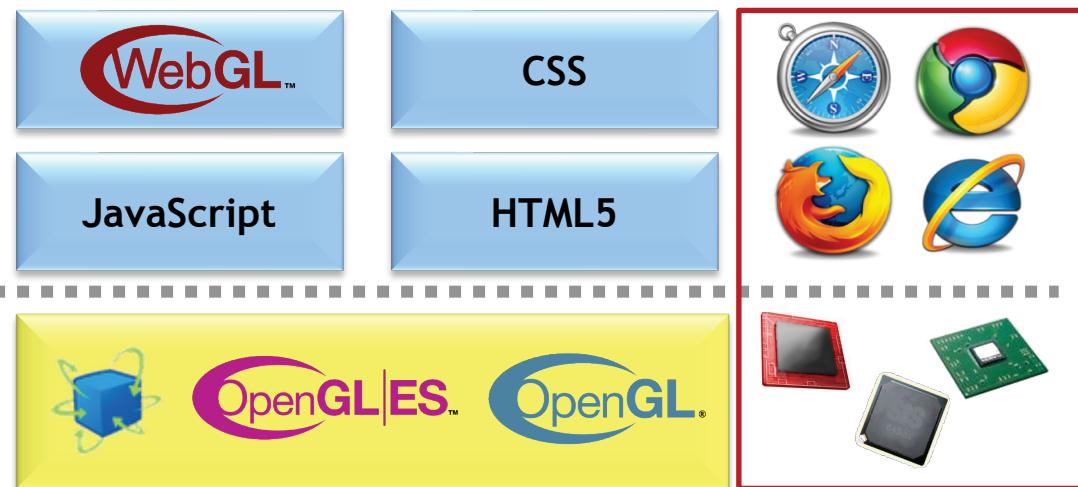
Content downloaded from the Web

Middleware provides accessibility  
for non-expert programmers  
E.g. three.js library



Low-level APIs provide  
a powerful foundation  
for a rich JavaScript  
middleware ecosystem

Browser provides WebGL  
3D engine alongside other HTML5  
technologies - no plug-in required



Reliable WebGL  
relies on work by  
both GPU and  
Browser Vendors

->

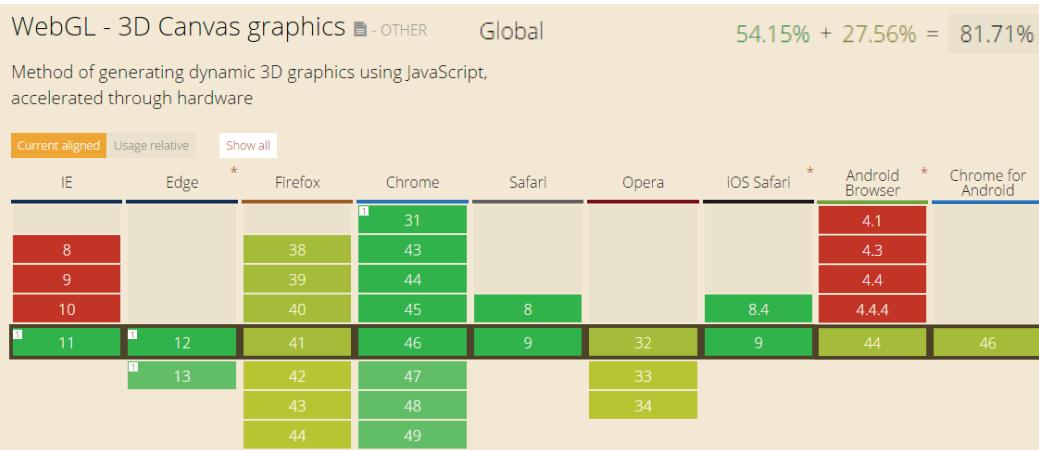
Khronos has the  
right membership  
to enable that  
cooperation

OS Provided Drivers  
WebGL uses OpenGL ES 2.0 or  
Angle for OpenGL ES 2.0 over DX9

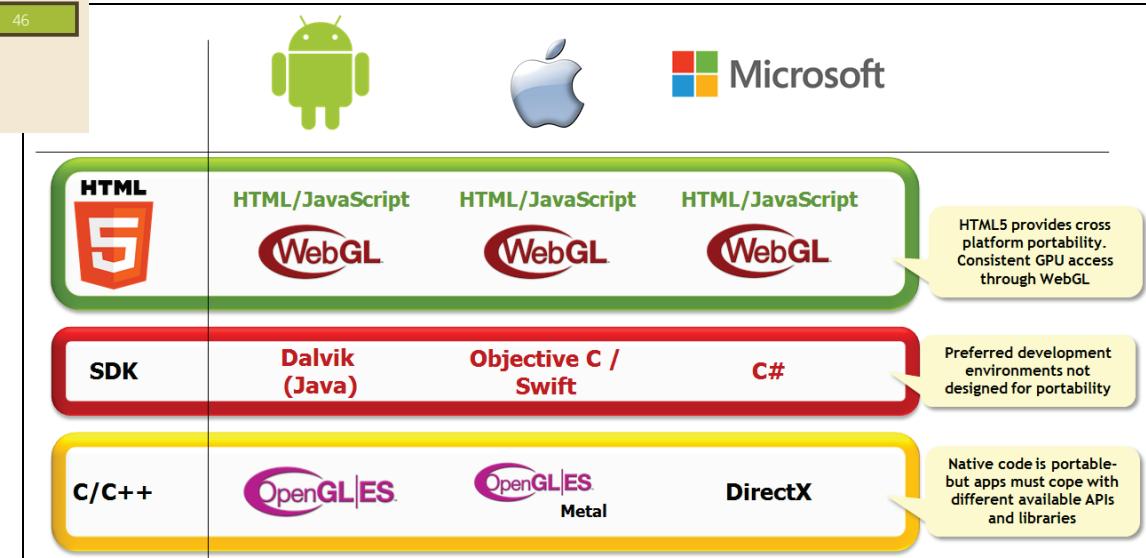


# Pervasive WebGL

- WebGL on EVERY major desktop and mobile browser
- Portable (NO source change) 3D applications are possible for the first time

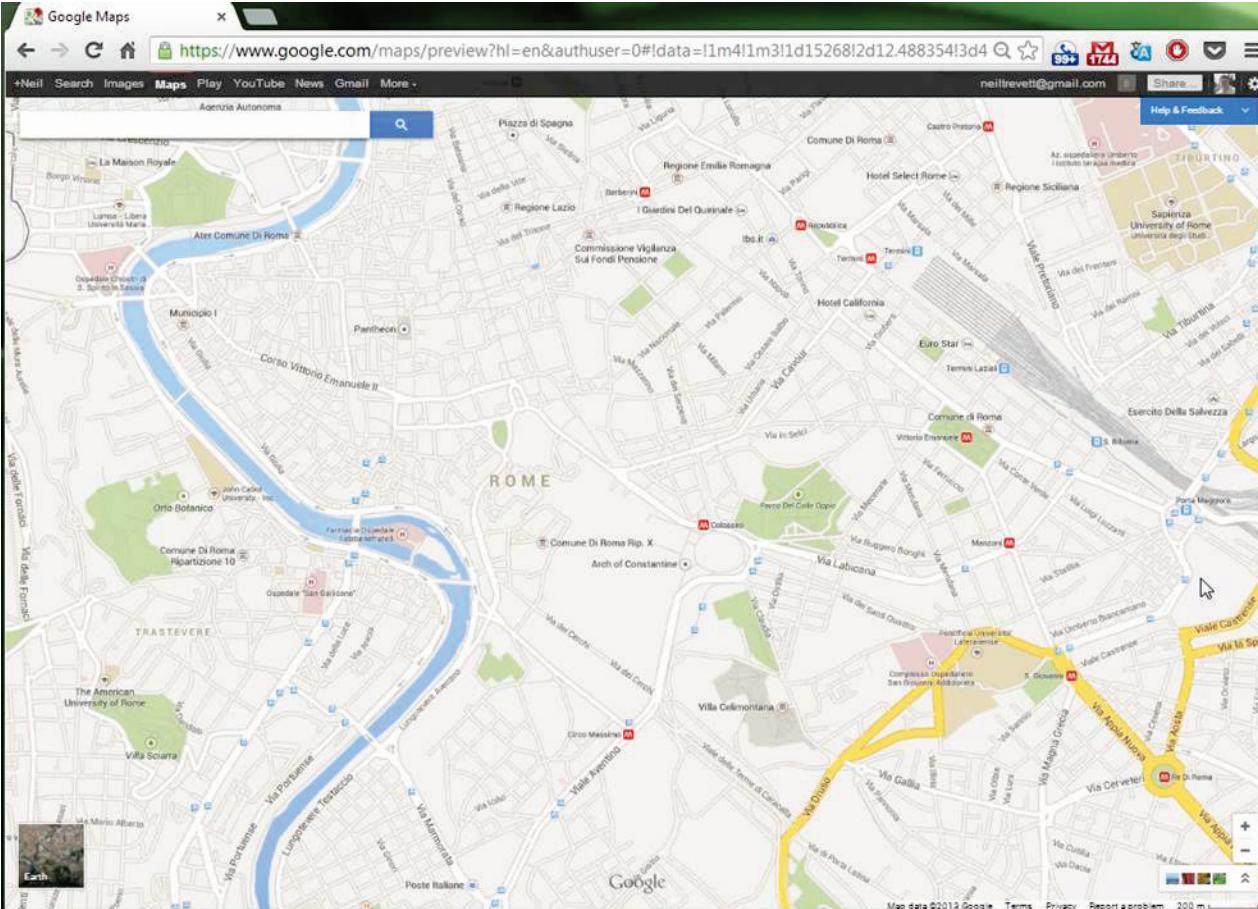


<http://caniuse.com/#feat=webgl>



# Google Maps

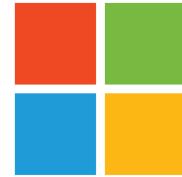
- All rendering (2D and 3D) in Google Maps uses WebGL



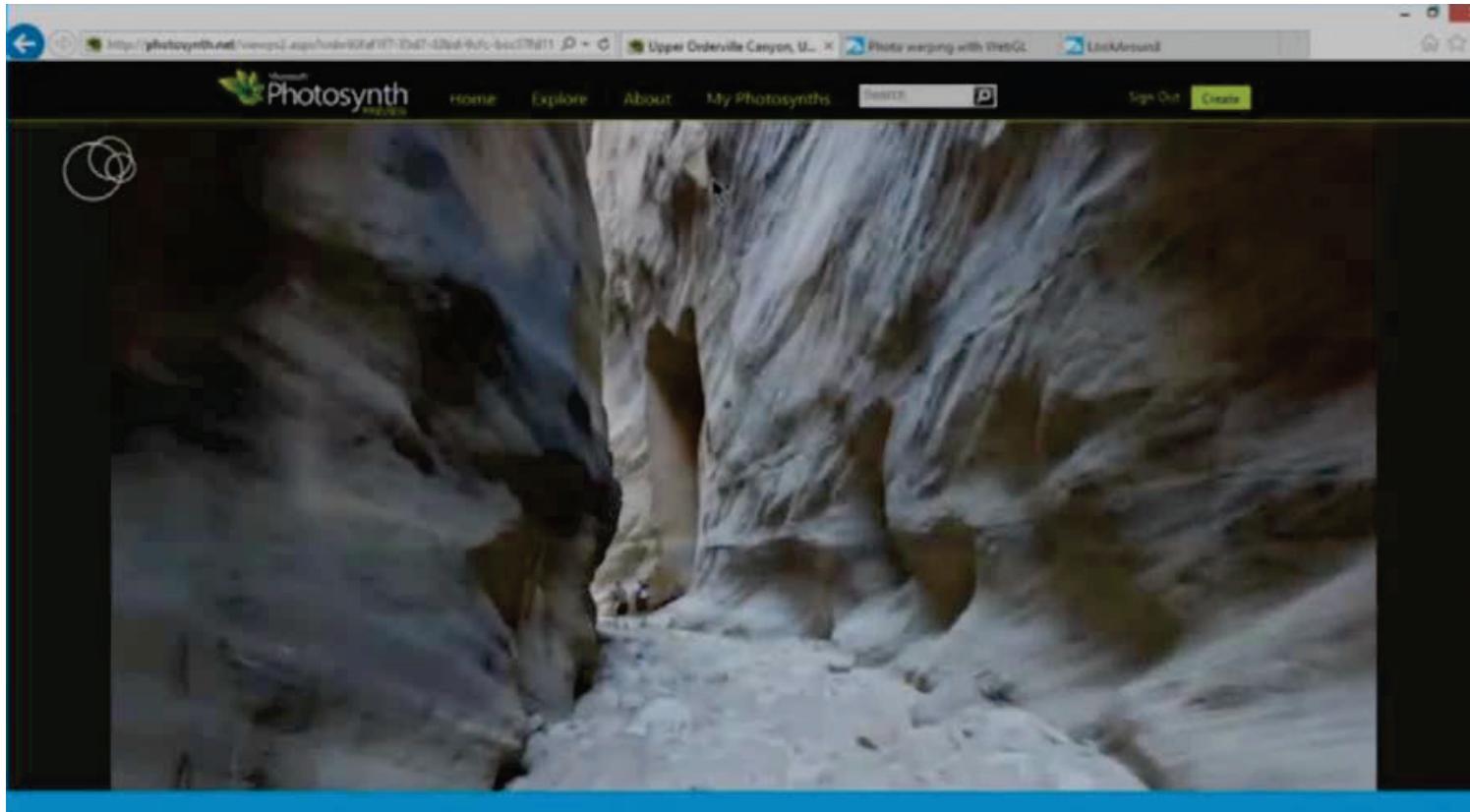
Google™

# Microsoft PhotoSynth2

- Demonstrated at Build 2013

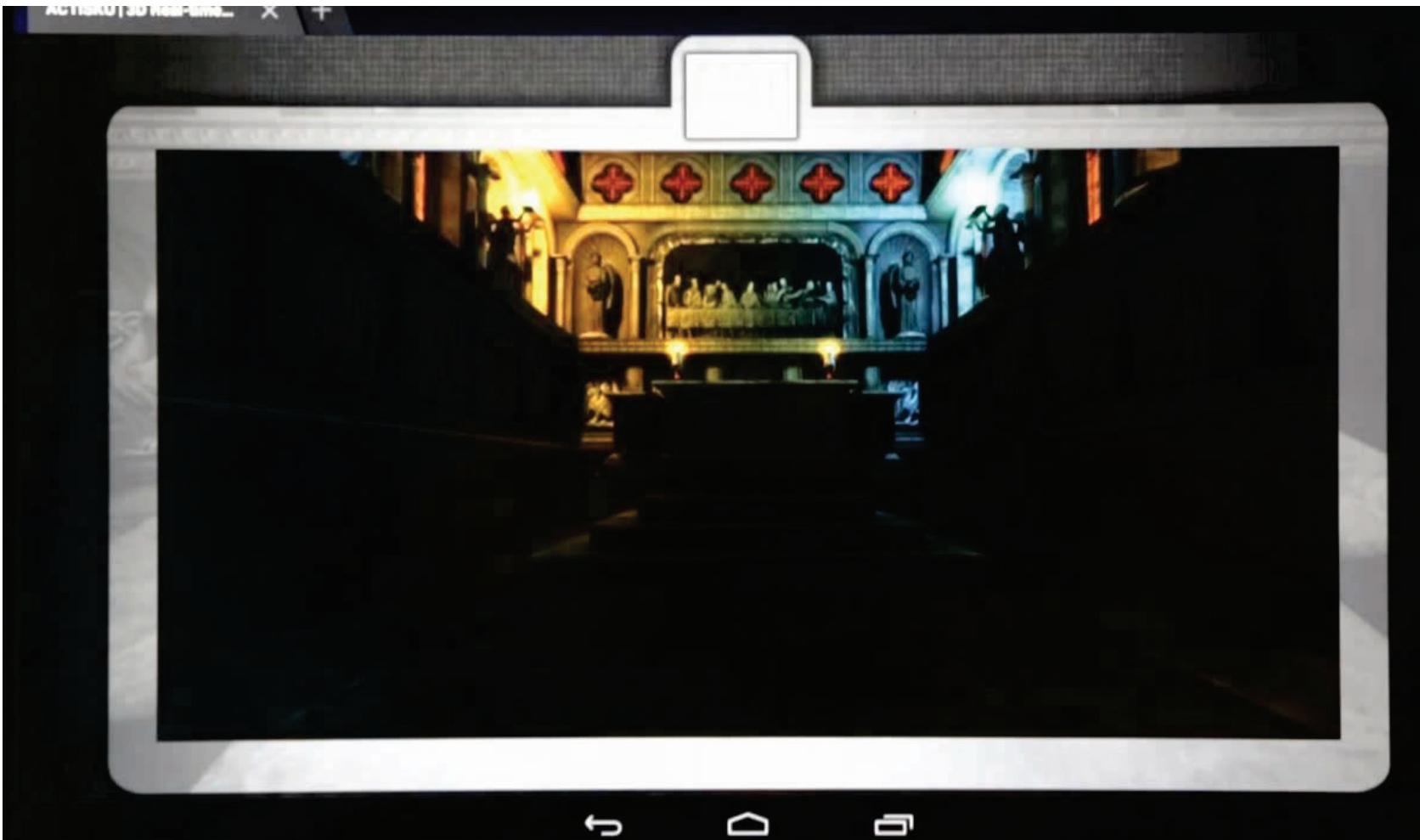


# Microsoft



<http://channel9.msdn.com/Events/Build/2013/4-072>

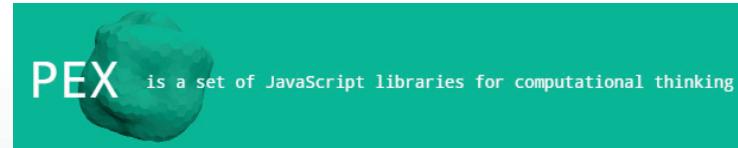
# WebGL on NVIDIA Shield Tablet



# WebGL Tools and Engines

PLAYCANVAS

xeoEngine



blend4web



sceneJS  
3D Engine for the Web



CESIUM



unity

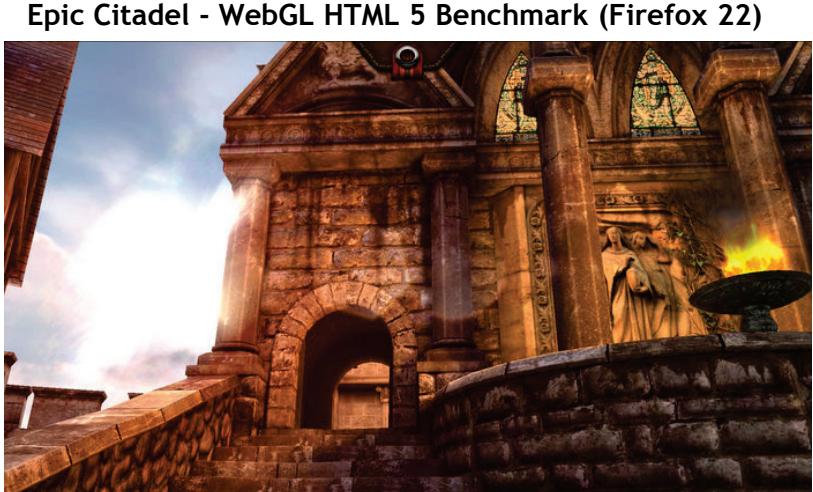


babylon  
js

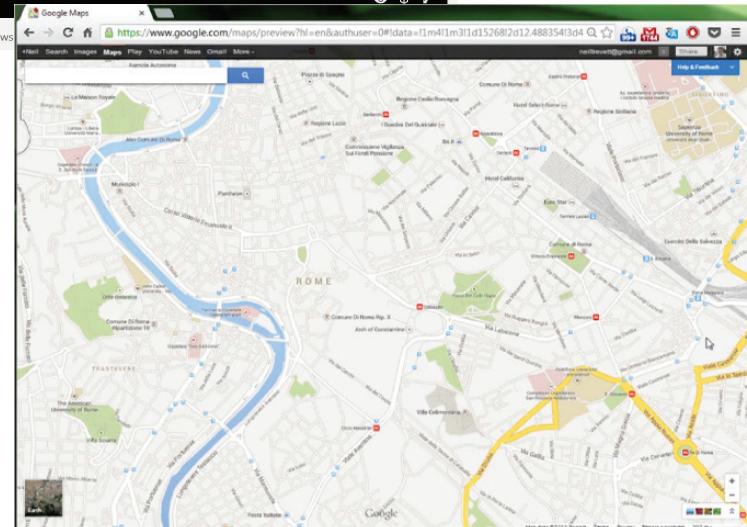
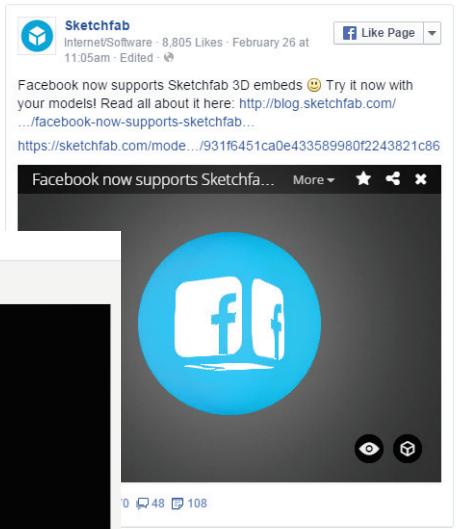
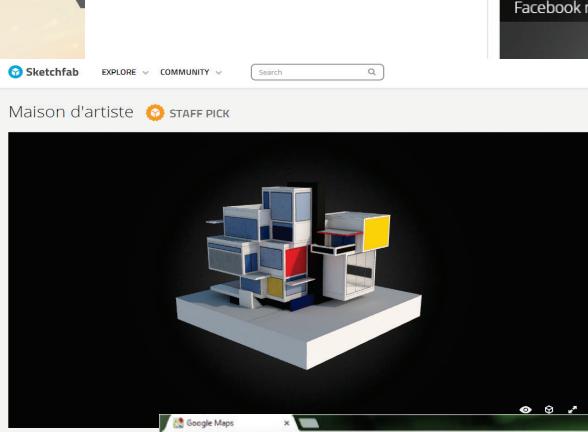
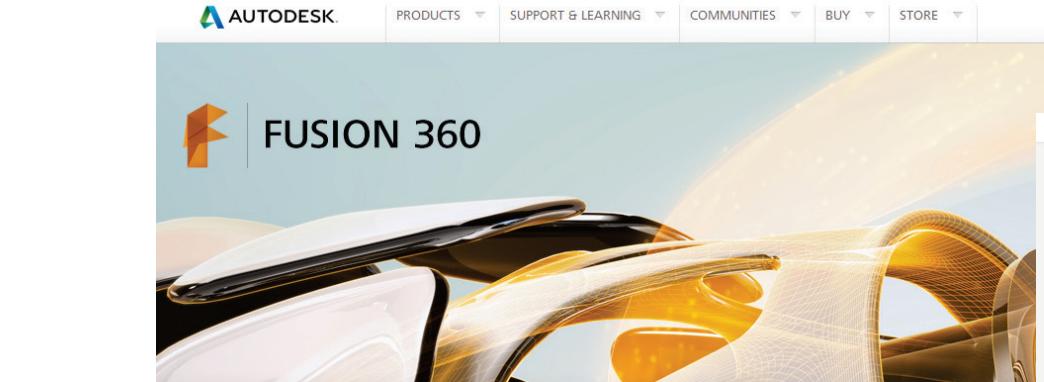
three.js



# WebGL Applications



<https://www.youtube.com/watch?v=l9KRBuVBjVo>

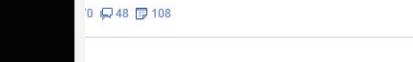
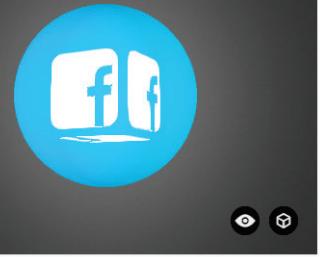
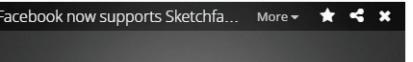


Facebook now supports Sketchfab 3D embeds!



Facebook now supports Sketchfab 3D embeds 😊 Try it now with your models! Read all about it here: <http://blog.sketchfab.com/facebook-now-supports-sketchfab...>

<https://sketchfab.com/models/931f6451ca0e433589980f2243821c86>



# WebGL 2.0

Fixed function  
Pipeline



Programmable  
Vertex and  
fragment shaders



32-bit integers and floats  
NPOT, 3D/depth textures  
Texture arrays  
Multiple Render Targets



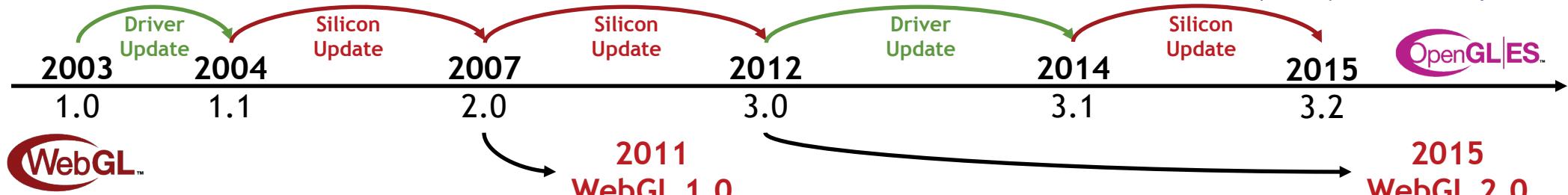
Compute Shaders



Tessellation and geometry shaders  
ASTC Texture Compression  
Floating point render targets  
Debug and robustness for security



Epic's Rivalry demo using full Unreal Engine 4  
<https://www.youtube.com/watch?v=jRr-G95GdaM>



## WebGL 2.0

Enhanced visual quality, performance, GPU acceleration

Instancing | Multiple render targets | Uniform buffers  
Transform feedback | Multi-sampled Renderbuffers | 3D textures  
NPOT textures | More texture formats | Occlusion queries  
Vertex array objects | Sampler objects | Sync objects  
Fragment depth | Primitive restart | ...

# WebGL 2.0 Status

- Draft spec is available
  - <https://www.khronos.org/registry/webgl/specs/latest/2.0/>
- Prototype implementations in Chrome and Firefox
  - [https://www.khronos.org/webgl/wiki/Getting\\_a\\_WebGL\\_Implementation](https://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation)
- Chromium aiming to pass all known conformance tests by the end of the year
- Many WebGL 2.0 features are available today as extensions
  - ANGLE\_instanced\_arrays, OES\_vertex\_array\_object, WEBGL\_draw\_buffers
- Formal release soon
  - Watch this space!

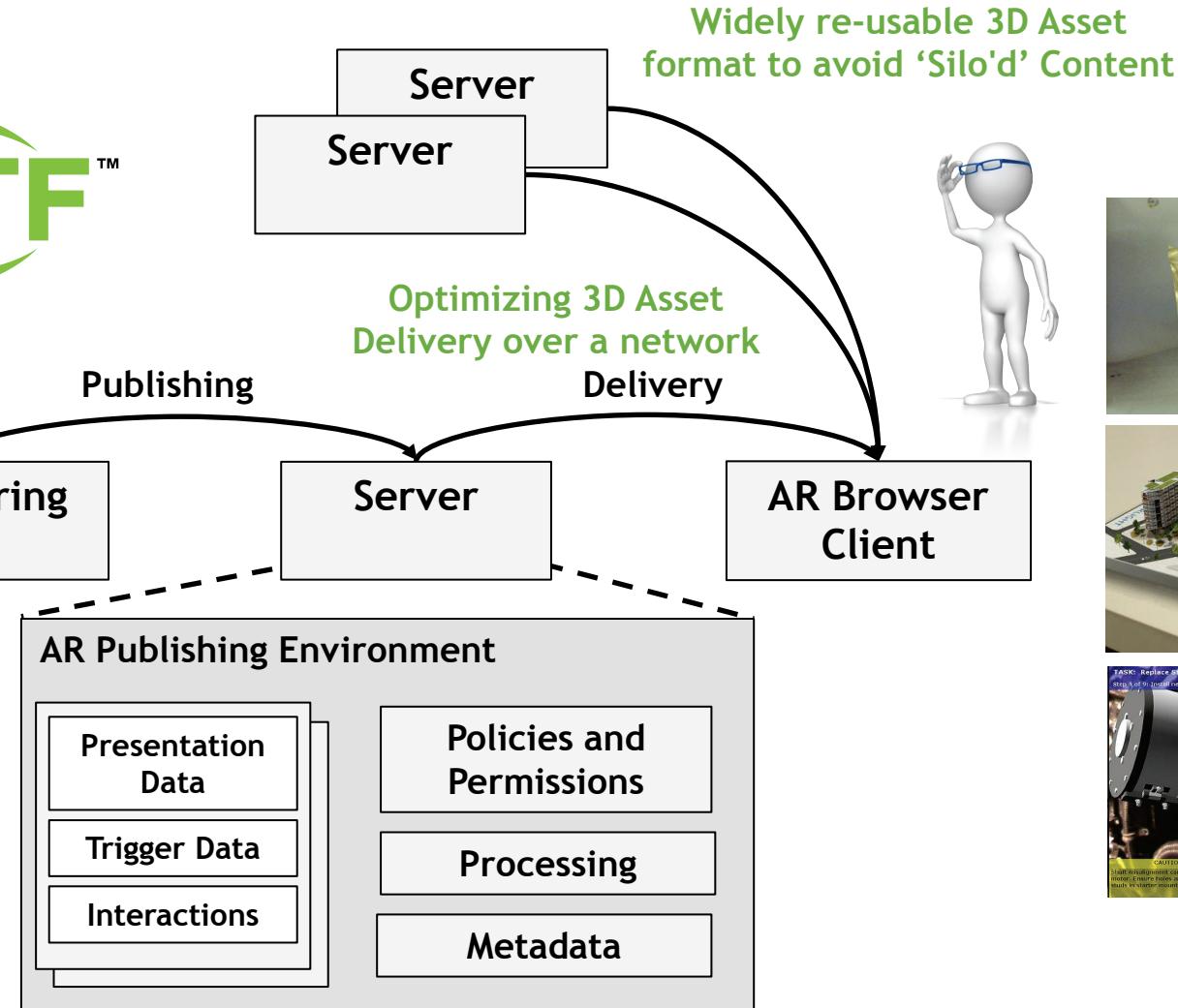


# AR Publishing Environment

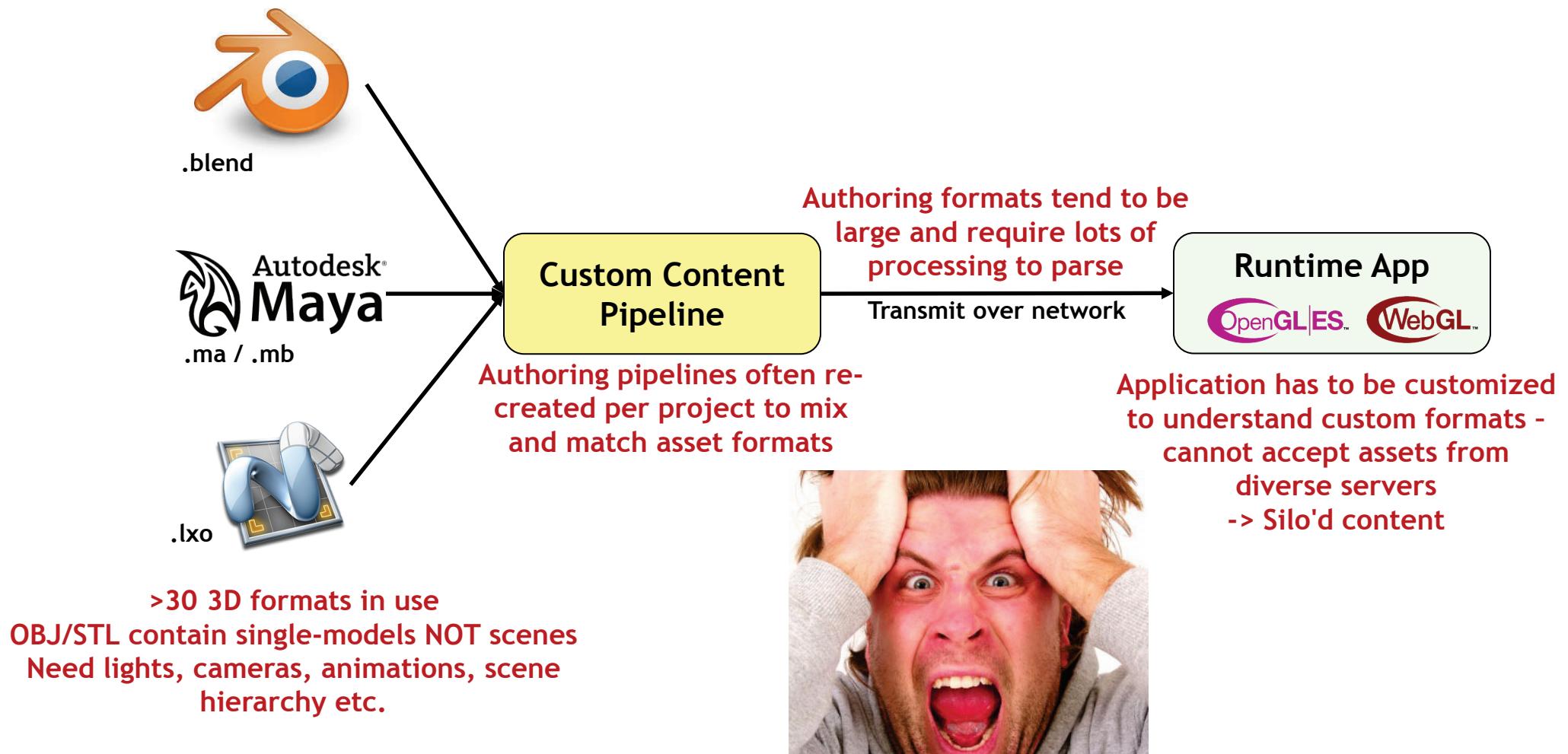


Physical World

Easing the authoring  
of 3D assets



# 3D Model Creation and Deployment - Today



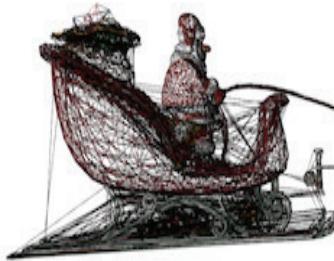
# 3D Needs a Transmission Format!

- Efficient run-time transmission of 3D assets becoming essential
  - Connected applications need access to increasingly large asset databases
- Bridge the gap between tools and ‘GL’ based apps
  - Reduce duplicated effort in content pipelines
  - Enable richer 3D representation - OBJ, STL etc. too limited
  - Provide common publishing format for content tools and services

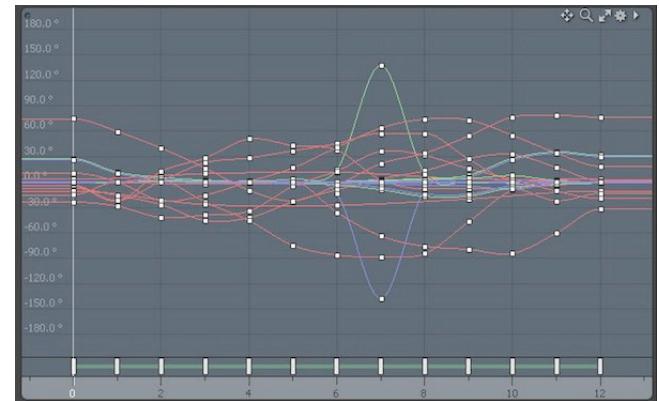
Audio	Video	Images	3D
MP3	H.264	JPEG	?
 napster.	 YouTube™	 facebook.	!

A widely adopted format ignites previously unimagined opportunities for a media type

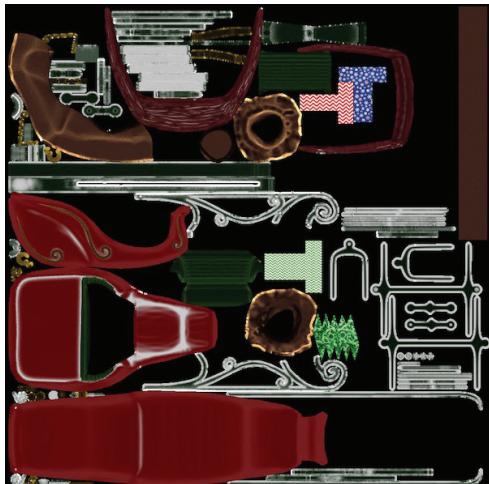
# What's in a 3D Asset or Model?



Scene hierarchy and geometry



Animations and skins

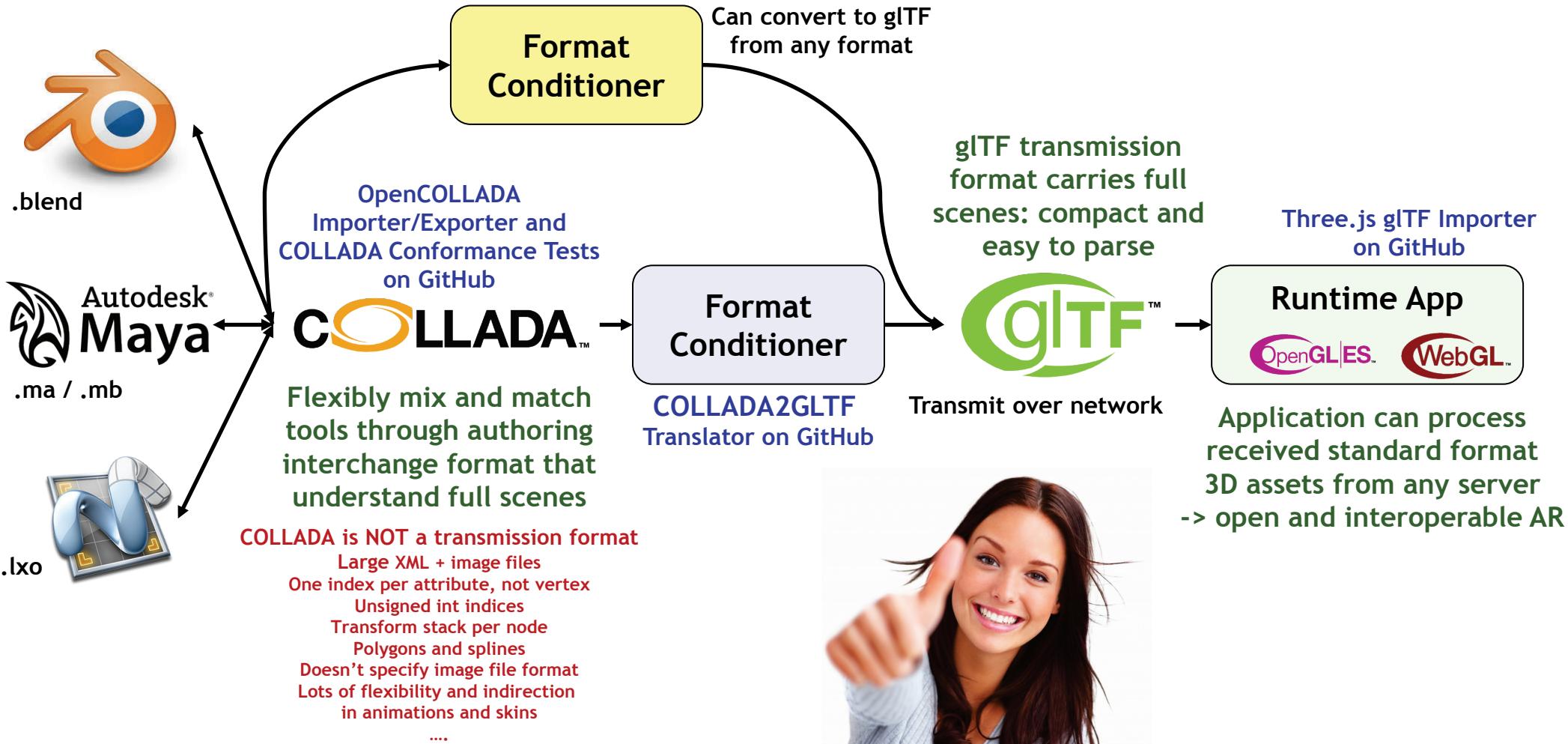


Materials and textures



Final Asset in Scene

# 3D Model Creation and Deployment Standards!



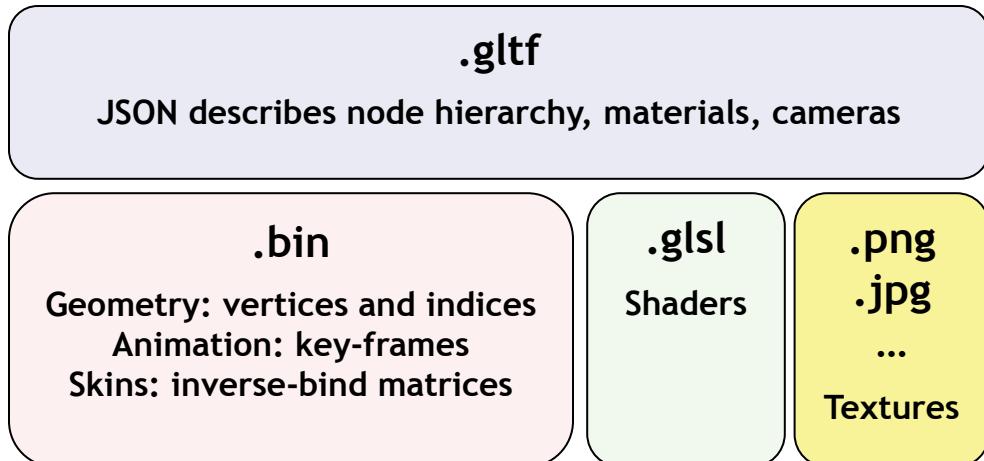
# glTF = “JPEG for 3D”

- ‘GL Transmission Format’
  - 3D asset runtime format for any application
  - Optimized for WebGL, OpenGL ES, and OpenGL apps
- Compact representation for download efficiency
  - Binary mesh and animation data
- Loads quickly into memory
  - GL native data types require no additional parsing
- Full-featured scenes
  - 3D constructs (node hierarchy, materials, animation, cameras, lights)
- Runtime Neutral
  - Can be created and used by any tool, app, or runtime
- Flexible Extensibility
  - E.g. payloads with compression and streaming



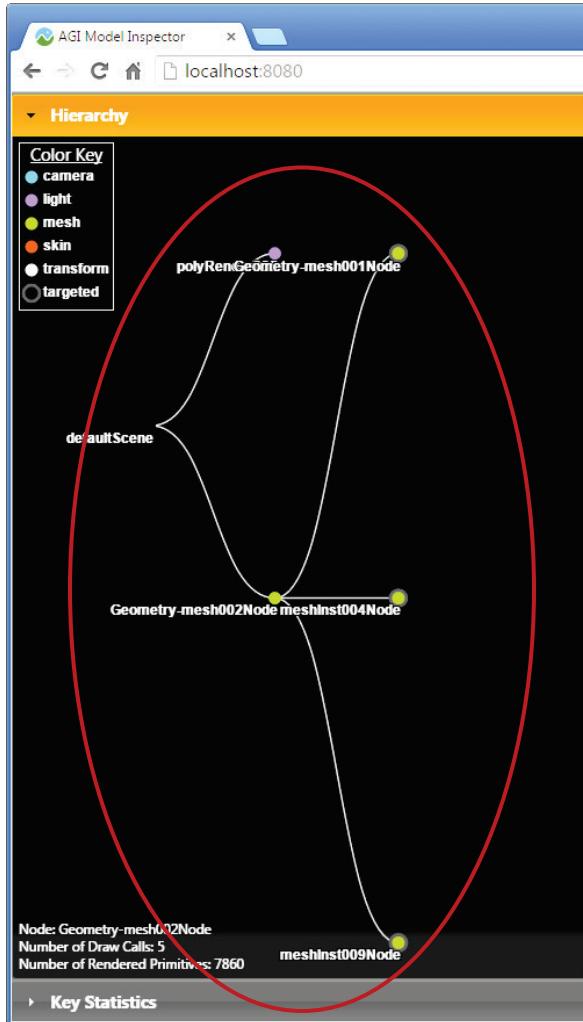
# glTF Internals

- JSON describes node hierarchy
  - Includes cameras
  - References geometry, animations, skins, shaders, textures
- Vertices
  - Uses native typed array format
  - Includes key-frame animations and skinning
- Shaders
  - With extensions for materials
- Textures
  - Use existing standard image compression formats e.g. JPEG
- Extras
  - For app-specific data (metadata)

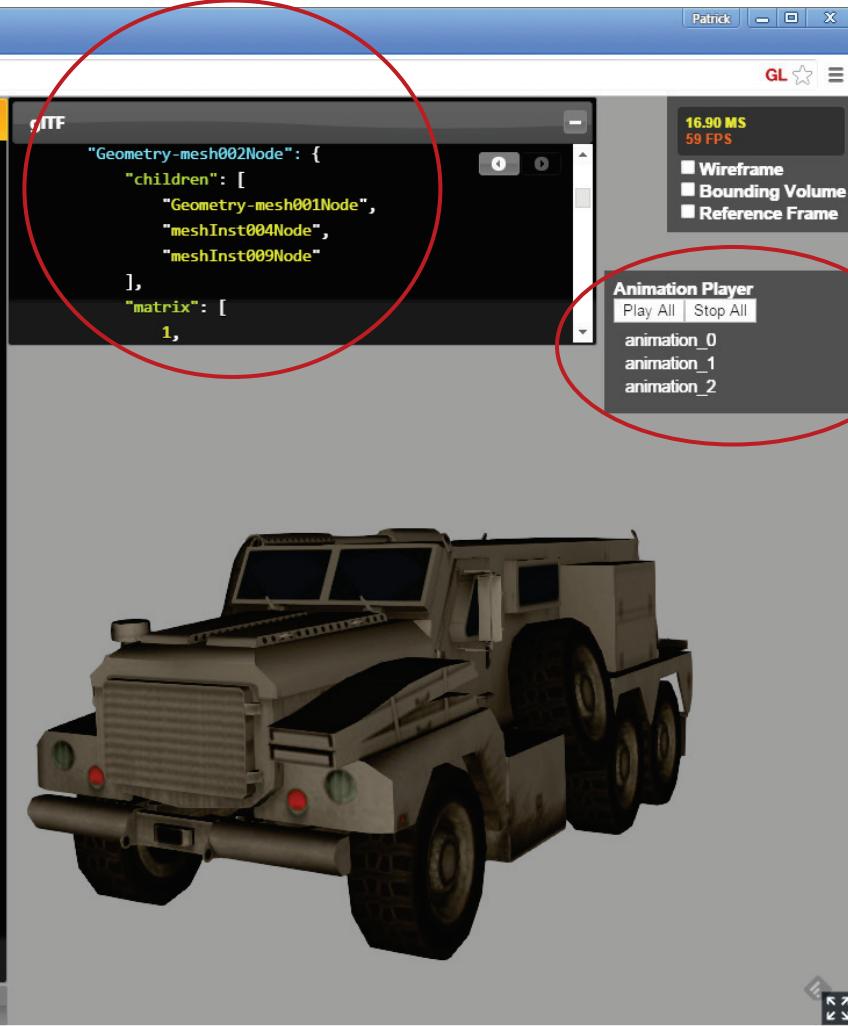


# glTF Example

Visualization  
of Node  
Hierarchy



JSON Node (the truck)  
with three children (sets of two wheels)



Three  
animations -  
one for each  
set of wheels

# glTF Project Status

- Open specification; Open process
  - Specification and multiple loaders and translators in open source
  - <https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md>
- glTF 1.0 spec finalized
  - Launched in October 2015!
- Extension mechanisms fully defined
  - Vendor, multi-vendor and official Khronos extensions (mirrors OpenGL)
  - Anyone can ship vendor extensions at any time - no permissions needed
  - First extensions included in launch
- More details
  - <https://www.khronos.org/gltf/>



# Launch Industry Support

**“It was obvious for the babylon.js team that glTF was a must have feature in order to integrate well within the 3D ecosystem.”**  
David Catuhe, principal program manager at Microsoft and author of babylon.js

**“glTF has some remarkable features that will make it simple for developers to include and run 3D digital assets in their web or mobile applications”**  
Cyrille Fauvel, senior ADN Sparks manager at Autodesk

**“Unlocking 3D content from proprietary desktop applications to the cloud creates massive new opportunities for collaboration. This future is so close we can feel it - the hardware is capable, the browsers are capable, now if only we could solve the content pipeline. Go glTF!”**  
Ross McKeegney , Platform @ Box

**“Defining a 3D graphics transmission model is challenging due to the extensive diversity of 3D graphics representations and use cases and the 3D ecosystem is being held back by a lack of a simple and universally efficient data representation. glTF has an important role by defining a foundation on which application specific compression and transmission components can be incrementally added. We are looking forward to glTF extensions to enable efficient MPEG compression technologies for 3D graphics to be widely deployed.”**  
Marius Preda of the MPEG Consortium

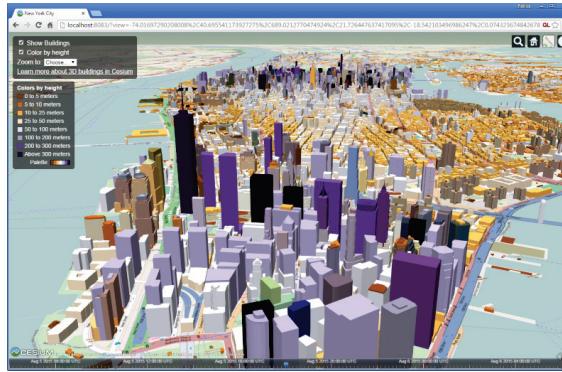
# glTF Adoption

## three.js Loader

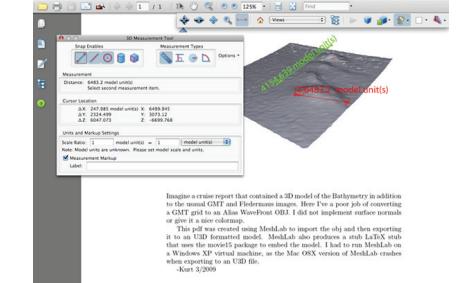
<https://github.com/mrdoob/three.js/>



It's the native format!  
<http://cesiumjs.org/>



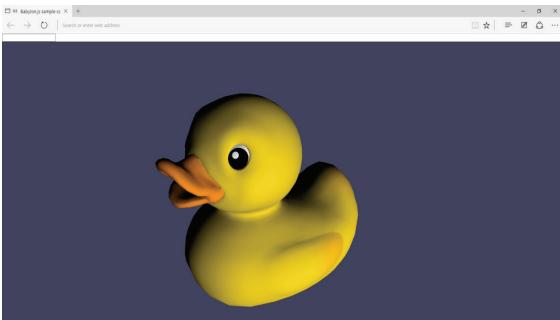
Native import and display of glTF models



KHRONOS™  
GROUP



**Babylon.js Loader** (in development)  
<http://www.babylonjs.com/>



## Pipeline Tools

### collada2gltf converter

<https://github.com/KhronosGroup/glTF>

**Online drag and drop COLLADA  
to glTF converter**

<http://cesiumjs.org/convertmodel.html>

**FBX to glTF Convertor  
(in development)**

Drag and drop convertor coming  
<http://gltf.autodesk.io/>



3D Advertising Solutions  
with native glTF import



# Initial glTF Extensions

- Any company can define glTF vendor extensions
  - Khronos manages extension name space
  - Popular extensions can be proposed to be adopted into standard extensions and then possibly into core
- **KHR\_binary\_gltf (Khronos extension)**
  - Enables a glTF file to use binary asset packages
- **EXT\_quantized\_attributes (vendor extension)**
  - Quantization-based attribute compression
  - Decompression in vertex shader
- **MPEG 3D mesh compression (in progress)**
  - MPEG-SC3DMC codec (Scalable Complexity 3D Mesh Compression)
  - Uses Open3DGC open source - C++ encoder/decoder + JavaScript decoder
  - 40-80% compression for many 3D assets
  - Extensions inserts decompression between file buffer and vertex data
  - Building support into the COLLADA2GLTF converter and Cesium loader

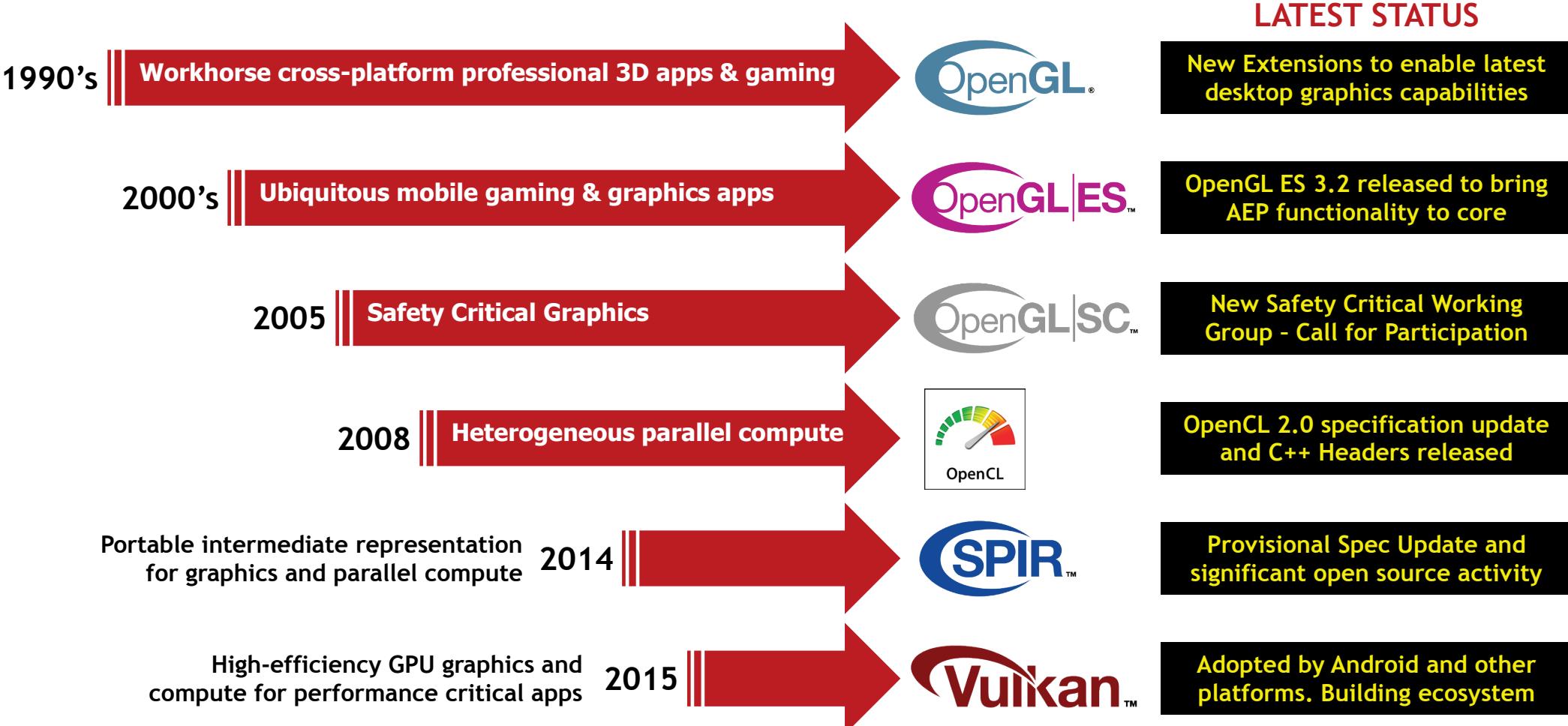


# Open3DGC glTF Extension Initial Results

Model	Vertices	Tris	Flat + Gzip	Open3DGC + Gzip	Compression Amount	JavaScript Execution Time
COLLADA Duck	2.1k	4.2k	54 KiB	14 KiB	-74%	24 ms
Stanford Bunny	2.5k	5.0k	105 KiB	56 KiB	-47%	30 ms
Stanford Dragon	435k	871k	7792 KiB	2141 KiB	-73%	630 ms
3D Tile	12.8k	6.5k	102 KiB	59 KiB	-42%	—
OpenStreetMap NYC	—	—	337 MiB	207 MiB	-39%	(Streamed)

Google Chrome 44.0, Windows 8.1, Intel i7-4980HQ @ 2.80GHz

# Khronos Open Standards for Graphics and Compute

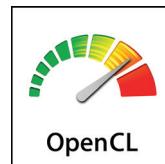


# Roadmap Possibilities

SPIR-V Ingestion for OpenGL and OpenGL ES for shading language flexibility



Thin and predictable graphics and compute for safety critical systems



1. C++ Shading Language
2. Single source C++ Programming from SYCL
3. OpenCL-class Heterogeneous Compute to Vulkan runtime

# Virtual Reality Will Influence Graphics APIs

- The ability to generate ‘Presence’ is becoming achievable at reasonable cost
  - Using visual input to generate subconscious belief in a virtual situation
  - PC-based AND mobile systems
- VR Requirements will affect how graphics APIs generate visual imagery
  - Control over generation of stereo pairs - slightly different view for each eye
  - Optical system geometric correction in rendering path
  - Reduce latency through elimination of in-driver buffering
  - Asynchronously warp framebuffer for instantaneous response to head movement

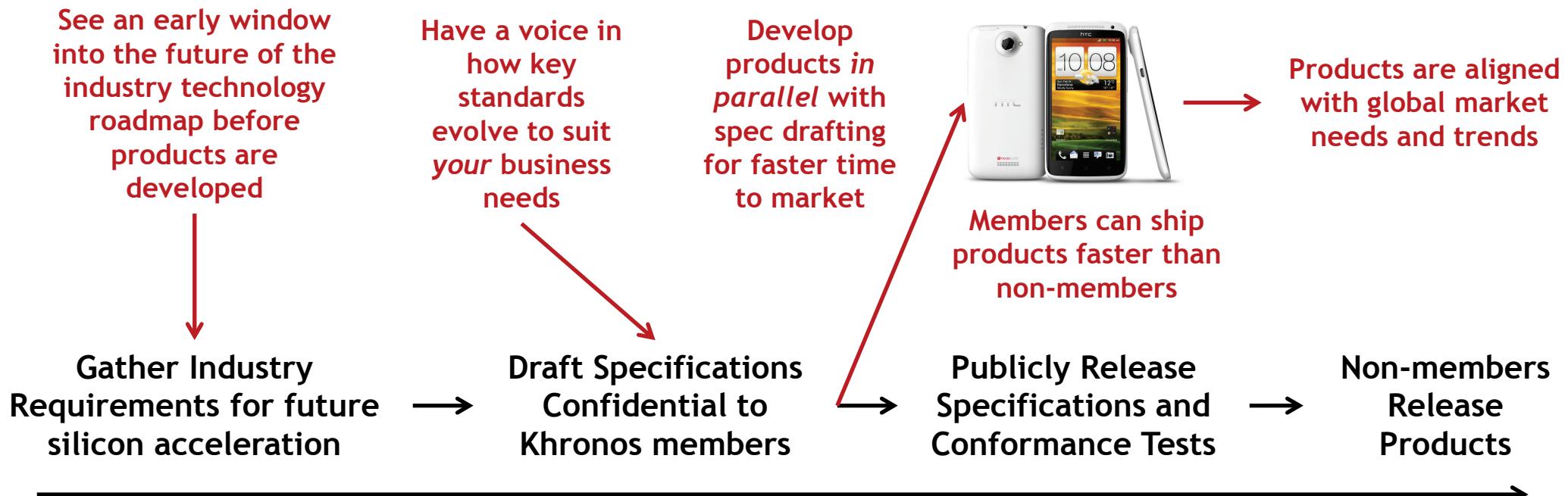


Many companies driving the VR revolution are participating in the Vulkan working group



Samsung Gear VR

# The Value of Khronos Participation



The Khronos standardization process is proven to RAPIDLY generate industry consensus on future hardware acceleration functionality to EFFICIENTLY create new market opportunities

# Summary

- Khronos is creating cutting-edge royalty-free open standards
  - For graphics and parallel computation
- Khronos standards are key to many emerging markets such as embedded vision, advanced gaming, Augmented and Virtual Reality
  - Advanced next generation capabilities for ALL platforms
- Any company is welcome to join Khronos influence the direction of these important international standards
  - \$15K annual membership fee for access to all Khronos API working groups
  - Well-defined IP framework protects your IP and conformant implementations
- More Information
  - [www.khronos.org](http://www.khronos.org)
  - [ntrevett@nvidia.com](mailto:ntrevett@nvidia.com)
  - [@neilt3d](https://twitter.com/neilt3d)

