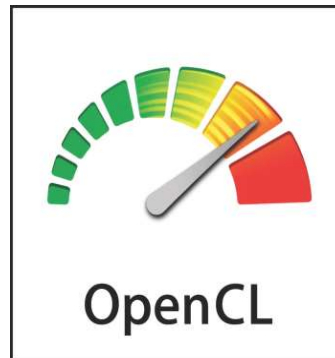


# KHRONOS™ GROUP



**Ben Ashbaugh**  
Intel



**Adam Lake**  
Intel

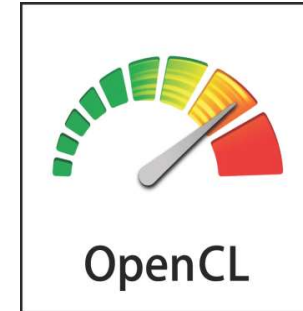


**Maria Rovatsou**  
Codeplay

IWOCL, May 2015

# Agenda

- OpenCL 2.1 Design Methodology
- C++ Kernel Language Overview
- SPIR-V Overview
- OpenCL 2.1 API Enhancements
- SYCL for OpenCL
- Panel Discussion



Core API and Language Specs



Portable Kernel Intermediate Language



Single Source C++ Programming

# Khronos Connects Software to Silicon

Open Consortium creating OPEN STANDARD APIs for hardware acceleration  
Any company is welcome - many international members - one company one vote

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

Software

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



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

Silicon

API Specifications AND Conformance  
Tests for cross-vendor portability

Strong industry momentum

100s of man years invested by industry experts

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



BOARD OF PROMOTERS



Over 100 members worldwide  
any company is welcome to join

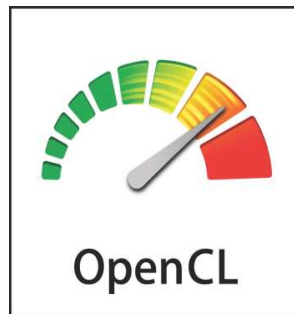


# OpenCL Ecosystem

## Implementers Desktop/Mobile/FPGA



Single Source C++ Programming



Core API and Language Specs



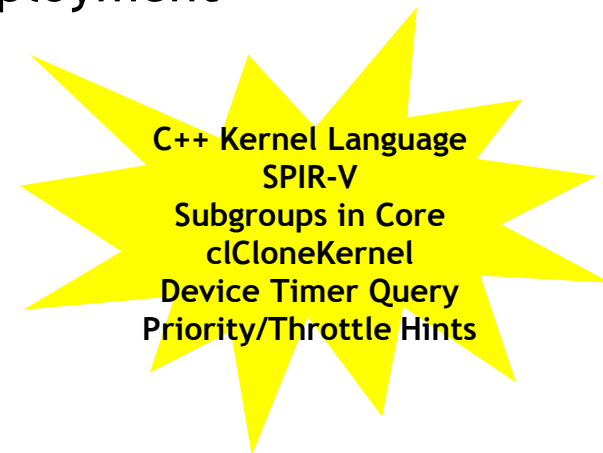
Portable Kernel Intermediate Language

## Working Group Members Apps/Tools/Tests/Courseware



# OpenCL 2.1 Provisional Released March 2015!

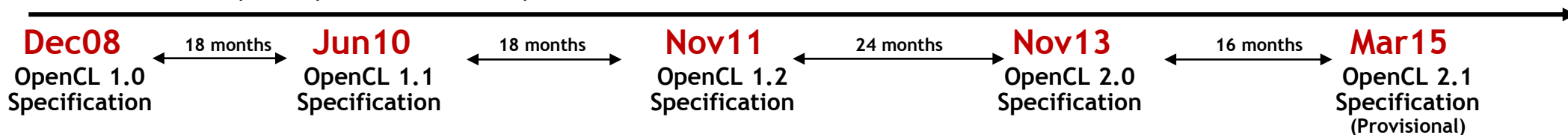
- **New OpenCL C++ Kernel Language**
  - Significantly enhanced programmer productivity and code performance
  - Still supporting OpenCL C to preserve kernel code investment
- **Support for the New Khronos SPIR-V Intermediate Language**
  - Improves portability and simplifies C++ Kernel Language deployment
- **Runs on any OpenCL 2.0-capable hardware**
  - Only driver update required

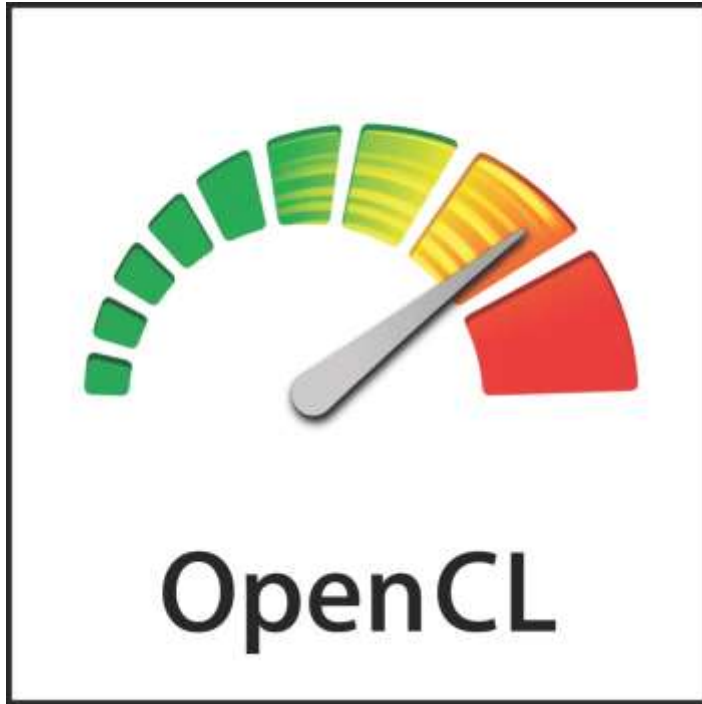


3-component Vectors  
Additional Image Formats  
Multiple Hosts and Devices  
Buffer Region Operations  
Enhanced Event-Driven Execution  
Additional OpenCL C Built-ins  
Improved OpenGL Data/Event Interop

Device Partitioning  
Separate Compilation and Linking  
Enhanced Image Support  
Built-in Kernels / Custom Devices  
Enhanced DX and OpenGL Interop

Shared Virtual Memory  
Device Enqueue  
Generic Address Space  
Enhanced Image Support  
C11 Atomics  
Pipes  
Android ICD





**K H R O N O S**<sup>™</sup>  
G R O U P

# OpenCL 2.1 C++ Kernel Language

# OpenCL C++ Kernel Language Overview

- **A “Static Subset” of C++14**
  - Frees developers from low-level coding details without sacrificing performance
- **In: Classes, templates, function and operator overloading, more...**
  - Reusable device libraries and containers - fast and elegant sharable code
  - Templates enables meta-programming for highly adaptive software
- **In: Upgraded Standard Library**
  - Leverages C++ standard library features
  - Examples: atomics, images, device queues, math functions
- **Out: Virtual Functions, Exceptions, Type Identification, C++ Standard Library...**



# Example: A Simple OpenCL C++ Kernel

```
#include <opencl_stdlib>
using namespace cl;

template<typename T>
void add_vectors(const T* srcA, const T* srcB, T* dst)
{
    size_t id = get_global_id(0);
    dst[id] = srcA[id] + srcB[id];
}

kernel void
add_vectors_float(const float* srcA, const float* srcB, float* dst)
{
    add_vectors(srcA, srcB, dst);
}

kernel void
add_vectors_float4(const float4* srcA, const float4* srcB, float4* dst)
{
    add_vectors(srcA, srcB, dst);
}
```

New! Kernel Language Functions  
Organized into Header Files

New! Kernel Language Functions  
in the `cl` Namespace

New! Full support for Templates

# OpenCL C++ Address Spaces

- OpenCL C has *global*, *local*, *constant* and *private* address space type qualifiers
- OpenCL C++ 2.1 does not need address space qualifiers
  - Pointers refer to allocations in the generic address space
- For local memory allocations, use the following types:
  - *local\_ptr<typename T>*
  - *local\_array<typename T, size\_t N>*
  - *local<T>*
- For constant memory allocations, use the following types:
  - *constant\_ptr<typename T>*
  - *constant\_array<typename T, size\_t N>*
  - *constant<T>*

# OpenCL C++ Device-Side Enqueue Syntax

- **Kernels can independently launch work on the device**
  - without host interaction
  - control execution order with event dependencies (user events or markers)
- **Kernels can enqueue:**
  - a kernel function or
  - code represented as a kernel lambda function
- **A kernel lambda function is described as:**
  - [ capture-list ] ( params ) **kernel** { body }

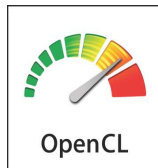


**K H R O N O S**<sup>™</sup>  
G R O U P

**OpenCL 2.1**  
**SPIR-V**

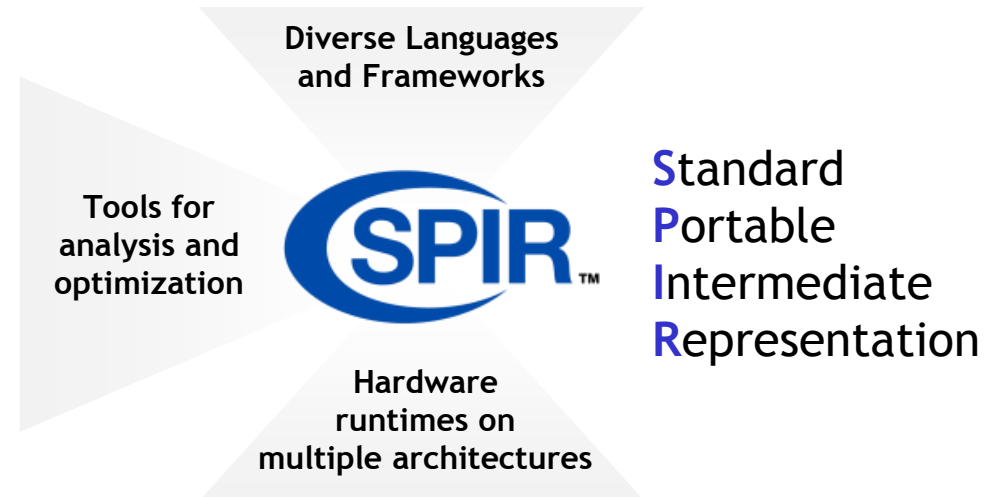
# What is SPIR-V?

- **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
- **SPIR-V - first multi-API, intermediate language for parallel compute and graphics**
  - Native representation for Vulkan shader and OpenCL kernel source languages




SPIR-V is supported in both Vulkan and OpenCL 2.1

**SPIR-V is a significant convergence point in the language ecosystem for graphics and parallel computation**

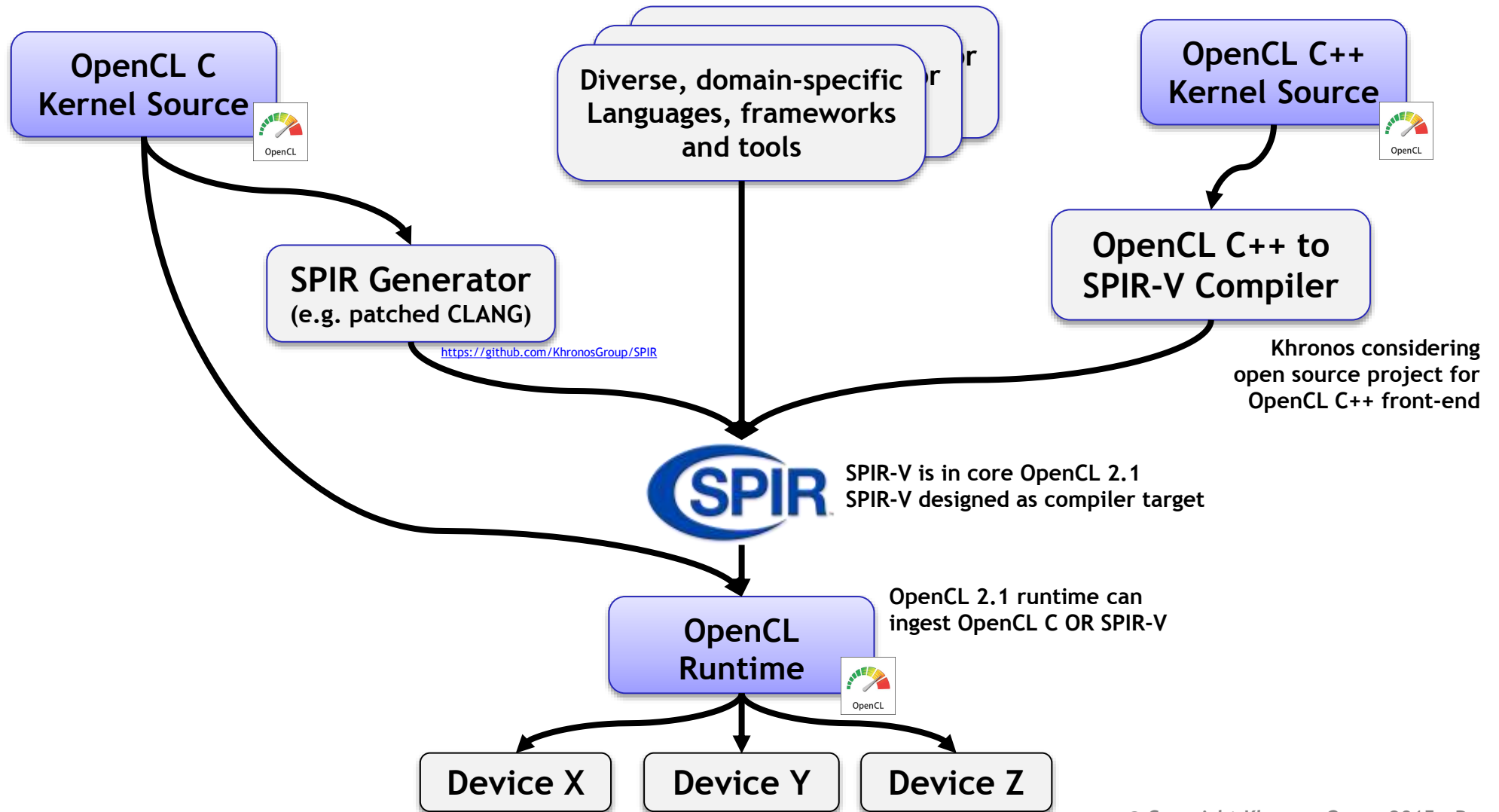


# Evolution of SPIR

- SPIR-V is the First Fully Specified Khronos-defined SPIR standard
  - Isolated from LLVM roadmap changes
  - Includes full flow control, graphics and parallel constructs beyond LLVM
  - Khronos considering open source SPIR-V <-> LLVM IR conversion tools

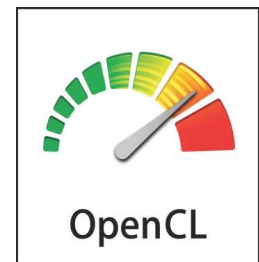
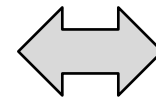
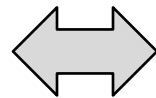
	<b>SPIR 1.2</b>	<b>SPIR 2.0</b> (Provisional)	<b>SPIR-V</b>
<i>LLVM Interaction</i>	<i>Uses LLVM 3.2 IR</i>	<i>Uses LLVM 3.4 IR</i>	<i>100% Khronos Defined</i>
<b>Compute Constructs</b>	<b>Metadata/Intrinsics</b>	<b>Metadata/Intrinsics</b>	<b>Native</b>
<i>Graphics Constructs</i>	<b>No</b>	<b>No</b>	<b>Native</b>
<b>Supported Language Feature Set</b>	<b>OpenCL C 1.2</b>	<b>OpenCL C 1.2 OpenCL C 2.0</b>	<b>OpenCL C 1.2 / 2.0 OpenCL C++ GLSL</b>
<i>OpenCL Consumption</i>	<i>OpenCL 1.2 Extension</i>	<i>OpenCL 2.0 Extension</i>	<i>OpenCL 2.1 CORE</i>
<b>Vulkan Consumption</b>	-	-	<b>Vulkan CORE</b>

# New OpenCL 2.1 Compiler Ecosystem

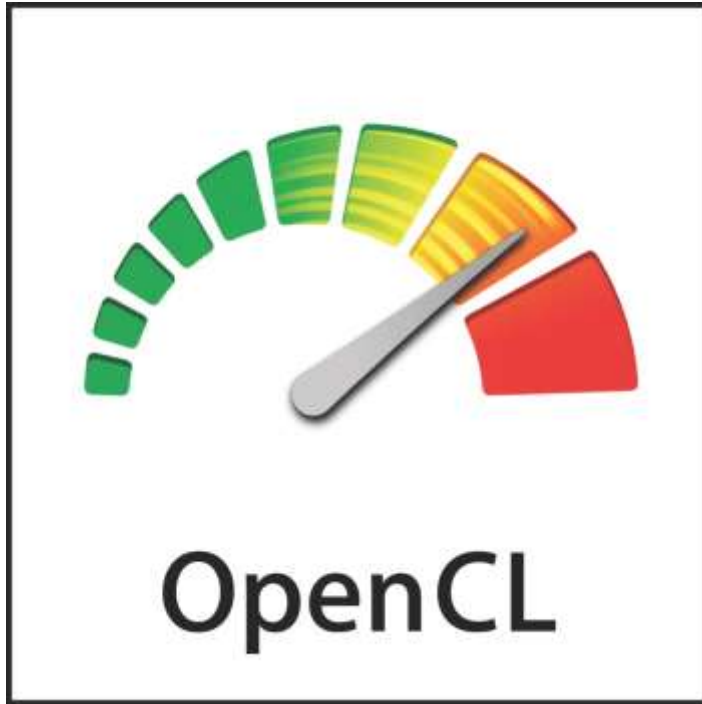


# SPIR-V Advantages for Developers

- **Eliminates a major source of cross-vendor portability**
  - Developers can use same front-end compiler across multiple platforms
- **Reduces runtime shader/kernel compilation time**
  - Driver only has to process SPIR-V, not full source language
- **Provides a measure of IP protection**
  - Don't have to ship shader/kernel source code
- **Drivers are simpler and more reliable**
  - No need to include front-end compilers
- **SPIR-V Whitepaper**
  - <https://www.khronos.org/registry/spir-v/papers/WhitePaper.pdf>







**K H R O N O S**<sup>™</sup>  
G R O U P

**OpenCL 2.1**  
**API**

# OpenCL 2.1 API Enhancements

- **clCreateProgramWithIL**
  - Clearly distinguish between SPIR-V and source/binary programs
- **clCloneKernel: deep copy of kernel, including arguments**
  - Safely pass kernels to threads or wrapper classes
- **cl\_khr\_subgroups: now a core feature**
  - Exposes hardware threads / warps / wavefronts and their cross-lane operations
- **Low-Latency Device Timer Query**
  - Synchronize host and device clock domains
- **Usability Enhancements**
  - Zero-sized dispatches are valid, support events and wait lists
  - NULL local work size supported with reqd\_work\_group\_size kernels

# OpenCL 2.1 API Extensions

- **Priority Hint**
  - Optionally, assign a “priority” to a command queue
  - Provides guidance when commands from two queues are ready to run
- **Throttle Hint**
  - Optionally, assign a “throttle level” to a command queue
  - Provides guidance to make appropriate power/performance tradeoffs



**K H R O N O S**<sup>TM</sup>  
G R O U P

# SYCL Update

# What is ?

- **SYCL**
  - Pronounced SICKLE
- **Royalty-free, cross platform, cross-toolchain C++ programming layer**
  - No language extensions, any standard C++ compilers can build SYCL source code, can have multiple device compilers linking into final executable
- **Full OpenCL feature set in a modern C++ single-source programming model**
- **A system that follows closely the developments in both C++ and OpenCL and enables projects that can serve as a dialog for both communities.**

# What does SYCL™ achieve?

- Single source C++11 programming model for OpenCL 1.2
- Ease of use
  - SYCL source compiled for host and device(s) (No language extensions, variety of environments and compilers for host and device)
  - Ease of integration with C++ libraries and applications optimized for other technologies
  - Development/Debugging on host
  - Programming interface based on abstraction of OpenCL components
- Provides the **full OpenCL feature set** and seamless integration with existing OpenCL code
- Enables the creation of **higher level programming models** and C++ templated libraries based on OpenCL

# Call to Action

- Khronos seeking feedback on OpenCL 2.1 and SPIR-V
  - Links provided on Khronos forums
  - [https://www.khronos.org/opencv/opencl\\_feedback\\_forum](https://www.khronos.org/opencv/opencl_feedback_forum)
  - [https://www.khronos.org/spir\\_v\\_feedback\\_forum](https://www.khronos.org/spir_v_feedback_forum)
- Or, give feedback to the panel *RIGHT NOW!*
- Reminder: Any company or organization is welcome to join Khronos for a voice and a vote in any of these standards
  - [www.khronos.org](http://www.khronos.org)

