Standards for Interoperable Open AR
Augmented World Expo, June 2015

Neil Trevett | Khronos President
NVIDIA Vice President Mobile Ecosystem
Khronos Connects Software to Silicon

Open Consortium creating ROYALTY-FREE, OPEN STANDARD APIs for hardware acceleration

Defining the roadmap for low-level silicon interfaces needed on every platform

Graphics, compute and vision processing

Rigorous specifications AND conformance tests for cross-vendor portability

Acceleration APIs BY the Industry FOR the Industry

Well over a BILLION people use Khronos APIs Every Day...
Khronos APIs for Augmented Reality

OpenVX™
Vision Processing for tracking, odometry and scene analysis

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

OpenGL
3D Graphics for Display of augmentations and visualizations

WebGL

Vulkan

Image: ScreenMedia
Mobile Graphics: The Story in Standards

1.0 1.1 2.0 2011 3.0 3.1

- Fixed function Pipeline
- Programmable Shaders
- 32-bit integers and floats
- NPOT, 3D/depth textures
- Texture arrays
- Multiple Render Targets
- Compute Shaders
- On GPU Work Dispatch

---

WebGL 2.0 - Open Review
- 8->32 textures
- High precision fragment shaders
- Occlusion queries
- Geometry instancing
- Multiple render targets etc.

http://www.khronos.org/registry/webgl/specs/latest/2.0/

Spec at GDC March 2014
Standard in Android L

WebGL 1.0

WebGL 2.0
Under Development
WebGL Ecosystem

Content downloaded from the Web
JavaScript, HTML, CSS, ...

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

Content

JavaScript Middleware

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

WebGL

CSS

JavaScript

HTML5

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

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

Reliable WebGL relies on work by both GPU and Browser Vendors
->
Khronos has the right membership to enable that cooperation

© Copyright Khronos Group 2015 - Page 5
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
Next Generation GPU APIs

Ground-up API designs for driving high-efficiency graphics and compute on GPUs

One OS

One GPU on One OS

All Modern Platforms and GPUs

An explicit API that is also cross-platform needs careful design balance

Unified memory and tiler architectures are first-class citizens

Vulkan-supported platforms will include Linux, Windows XP to Windows 10, Mobile
Vulkan - Portable, Nexgen GPU Access

- Ground-up design - open standard for high-efficiency GPU graphics and compute
  - Simpler drivers for low-overhead efficiency and cross vendor consistency
  - Unified API for mobile, desktop, console and embedded platforms
  - Layered architecture so validation and debug layers unloaded when not needed

- Multi-threading friendly - create graphics, compute and DMA command buffers
  - General model - could be extended to heterogeneous processing in the future
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
  - Will work on any GPU hardware that supports OpenGL ES 3.1/OpenGL 4.X and up
  - Can ship on any OS - including Windows XP/7/8

Working Group Participants
3D Needs a Transmission Format!

- Need to bridge the gap between tools and today’s GL based apps
  - Reduce duplicated effort in content pipelines
  - Common publishing format for content tools and services

- Browsers support loading of standard formats for many media types
  - With compression - for fast network transmission
  - With efficient client decode - especially important on mobile devices

- Why is 3D the last data type with an agreed transmission format?
  - More degrees of freedom
  - Many different use cases and ways that 3D data may want to be transmitted

<table>
<thead>
<tr>
<th>Audio</th>
<th>Video</th>
<th>Images</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP3</td>
<td>H.264</td>
<td>JPEG</td>
<td>?</td>
</tr>
<tr>
<td>Napster</td>
<td>YouTube</td>
<td>Facebook</td>
<td>!</td>
</tr>
</tbody>
</table>

An effective and widely adopted codec ignites previously unimagined opportunities for a media type
glTF = “JPEG for 3D”

- ‘GL Transmission Format’
  - Runtime asset format for WebGL, OpenGL ES, and OpenGL applications

- Compact representation for download efficiency
  - Binary mesh and animation data
  - Extension capability for future formats with compression and streaming

- Loads quickly into memory
  - JSON for scene structure and other high-level constructs
  - GL native data types require no additional parsing

- Full-featured
  - 3D constructs (hierarchy, cameras, lights, common materials, animation)
  - Full support for shaders and arbitrary materials

- Runtime Neutral
  - Can be created and used by any tool, app or runtime
glTF Internals

• JSON to describe node hierarchy
  - Platform neutral, run-time neutral, many processing libraries available
  - Node hierarchy refers to geometry, textures, materials, animations...

• Texture Blob
  - Can use existing standard image compression formats
  - ATSC tuned for textures and can be use directly by GPUs

• Shaders Blob
  - Use GLSL shaders today - compression is not normally an issue
  - Vendor portable PBR (physically-based rendering) materials is interesting topic

• Vertices Blob
  - Default is native GL typed array format
  - 3D asset streaming and compression is next optimization to explore - including MPEG compression
Vision Pipeline Challenges and Opportunities

Growing Camera Diversity
Flexible sensor and camera control to GENERATE an image stream

Diverse Vision Processors
Use efficient acceleration to PROCESS the image stream

Sensor Proliferation
Combine vision output with other sensor data on device
OpenVX - Vision Acceleration

- Royalty-free open standard API
  - Reliably accelerated by hardware vendors
  - Tightly defined conformance tests

- Targeted at low-power, real-time applications
  - Mobile and embedded platforms

- Portability across diverse heterogeneous processors
  - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs ...

- Doesn’t require high-power CPU/GPU Complex
  - Low-power host can setup and manage frame-rate vision processing pipeline

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

• 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
OpenVX Status

- Finalized OpenVX 1.0 specification released October 2014
  - OpenVX 1.0.1 spec maintenance update released June 2015
  - 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

- Full conformance test suite and Adopters Program available
  - Test suite exercises graph framework and functionality of each OpenVX 1.0 nod
### OpenVX and OpenCV are Complementary

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

- Lower-level compute APIs can be used to implement OpenVX nodes
  - Depending on the available processors
- Coding in OpenCL can provide portability across heterogeneous processors
  - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs ...

Implementers may choose to use OpenCL or Compute Shaders to implement OpenVX nodes on programmable processors ...

And then use OpenVX to enable a developer to easily connect those nodes into a graph

<table>
<thead>
<tr>
<th>Application Software</th>
<th>OpenVX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engines/frameworks</td>
<td>C/C++</td>
</tr>
<tr>
<td>Powerful, flexible</td>
<td></td>
</tr>
<tr>
<td>low-level APIs/languages</td>
<td>OpenCL, OpenGL ES</td>
</tr>
<tr>
<td>Processor Hardware</td>
<td>Dedicated Vision Hardware</td>
</tr>
</tbody>
</table>
OpenCL

- Heterogeneous parallel programming of diverse compute resources
  - Targeting supercomputers -> mobile devices -> embedded systems
- One code tree can be executed on CPUs, GPUs, DSPs, FPGA and hardware
  - Distribute work across all available processors in a system
- Can represent function of hardware ‘Custom Devices’ as built-in kernels
  - Control hardware from OpenCL run-time: e.g. video encode/decode, Camera ISP
- Robust framework for coding complete applications
  - One source tree with CPU and multiple accelerated paths
OpenCL - How it Works

- OpenCL = Two APIs and Two Kernel languages
  - C Platform Layer API to query, select and initialize compute devices
  - C Runtime API to build and execute kernels across multiple devices
  - OpenCL C and OpenCL C++ kernel languages

OpenCL C++ Kernel Language

- Static subset of C++14
- Classes, lambda functions, templates, operator overloading..
- Fast and elegant sharable code - reusable device libraries and containers with PERFORMANCE PORTABILITY
- Template meta-programming for highly adaptive software
  - Lambdas used for nested/dynamic parallelism

OpenCL 2.1 Provisional - March 2015
OpenCL C++ kernel language
SPIR-V in Core - no C++ front-end in driver
Runs on any OpenCL 2.0-capable hardware
OpenCL Implementations

Vendor timelines are first implementation of each spec generation.

- AMD: 1.0 May 2009
- IBM: 1.0 Aug 2009
- Intel: 1.0 May 2010
- NVIDIA: 1.0 May 2009
- ARM: 1.0 Feb 2011
- Imagination: 1.0 Feb 2011
- Qualcomm: 1.0 Jan 2010
- Vivante: 1.0 May 2010
- MediaTek: 1.0 Jan 2010
- STI: 1.0 Jan 2010
- Samsung: 1.0 Dec 2010
- Texas Instruments: 1.0 Dec 2010
- Altera: 1.0 May 2010
- Xilinx: 1.0 Jul 2013
- AMD: 1.1 Jul 2011
- IBM: 1.1 Aug 2010
- Intel: 1.1 Mar 2011
- NVIDIA: 1.1 Jun 2010
- ARM: 1.1 Mar 2012
- Imagination: 1.1 May 2010
- Qualcomm: 1.1 Feb 2011
- Vivante: 1.1 Mar 2011
- MediaTek: 1.1 Feb 2011
- STI: 1.1 Apr 2011
- Samsung: 1.1 Aug 2012
- Texas Instruments: 1.1 Nov 2012
- Altera: 1.1 May 2013
- Xilinx: 1.1 Apr 2012
- AMD: 1.2 Jun 2012
- IBM: 1.2 May 2012
- Intel: 1.2 Jun 2012
- NVIDIA: 1.2 May 2012
- ARM: 1.2 May 2015
- Imagination: 1.2 Sep 2013
- Qualcomm: 1.2 Apr 2014
- Vivante: 1.2 Dec 2014
- MediaTek: 1.2 Sep 2014
- STI: 1.2 May 2015
- Samsung: 1.2 Sep 2014
- Texas Instruments: 1.2 Dec 2014
- Altera: 1.2 May 2015
- Xilinx: 1.2 Dec 2014

Desktop:
- AMD: 2.0 Dec 2014
- IBM: 2.0 Jul 2014
- Intel: 2.0 May 2015
- ARM: 2.0 May 2015
- Imagination: 2.0 May 2015
- Qualcomm: 2.0 May 2015
- Vivante: 2.0 May 2015
- MediaTek: 2.0 May 2015
- STI: 2.0 May 2015
- Samsung: 2.0 May 2015
- Texas Instruments: 2.0 May 2015
- Altera: 2.0 May 2015
- Xilinx: 2.0 May 2015

Mobile:
- AMD: 1.1 Aug 2010
- IBM: 1.1 Jul 2011
- Intel: 1.1 May 2012
- ARM: 1.1 Nov 2012
- Imagination: 1.1 May 2013
- Qualcomm: 1.1 Apr 2012
- Vivante: 1.1 Mar 2011
- MediaTek: 1.1 May 2013
- STI: 1.1 May 2010
- Samsung: 1.1 Nov 2011
- Texas Instruments: 1.1 May 2010
- Altera: 1.1 Jul 2013
- Xilinx: 1.1 Dec 2013

Embedded:
- AMD: 1.2 Sep 2013
- IBM: 1.2 Apr 2014
- Intel: 1.2 Dec 2014
- ARM: 1.2 Sep 2014
- Imagination: 1.2 May 2015
- Qualcomm: 1.2 Sep 2014
- Vivante: 1.2 May 2015
- MediaTek: 1.2 May 2015
- STI: 1.2 May 2015
- Samsung: 1.2 May 2015
- Texas Instruments: 1.2 May 2015
- Altera: 1.2 May 2015
- Xilinx: 1.2 May 2015

Vendor timelines are first implementation of each spec generation.
Embedded Needs Driving OpenCL Roadmap

- OpenCL roadmap discussions focused on mobile and embedded markets
  - Very different needs and constraints to HPC/desktop

Different Bottlenecks to HPC
- Mobile GPU have less processing throughput than desktop
- GPU often busy with rendering hi-res screen
- Mobile memory bandwidth is often perf bottleneck

Always-on vision / sensor processing key use case
- Must run at very low power levels
- Thermal conditions often limit performance

Vision Applications Must Not Drop Frames
- Real-time / guaranteed QOS

App performance portability is much more critical
- More diverse architectures in mobile than HPC
  - 1000s of different devices

Many diverse hardware processing blocks
- DSP, ISP, dedicated hardware

Dynamic load-balancing
- Instrumentation/control to route work to different compute resources depending on system loading

Optimize Energy Efficiency
- Reduce precision - even integer-only profiles?
- Autonomous frame processing - no host involvement?

Extended Execution Model
- Pre-emption, yielding

Higher-level portability frameworks and engines
- Built over powerful low-level APIs

Expand support for Custom Devices
- Extended pipes and graphs in OpenCL run-time?
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

• 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

SPIR-V is a significant convergence point in the language ecosystem for graphics and parallel computation
SPIR-V at the Center of Language Ecosystem

- OpenCL C
- OpenCL C++
- GLSL
- Vulkan Driver X
- Vulkan Driver Y
- Other Languages

Other Intermediate Forms

LLVM

OpenCL Driver A

OpenCL Driver B

New kernel and shader Languages

* Khronos considering developing open source implementations of these translators

SPIR-V
- 32-bit Word Stream
- Extensible and easily parsed
- Retains data object and control flow information for effective code generation and translation

* Khronos considering developing open source implementations of these translators

New kernel and shader Languages

Other Languages

SPIR-V

• 32-bit Word Stream
• Extensible and easily parsed
• Retains data object and control flow information for effective code generation and translation

* Khronos considering developing open source implementations of these translators

Other Languages
Summary

- Khronos is creating cutting-edge royalty-free open standards
  - For graphics, compute and vision processing

- Khronos standards are key to many emerging markets such as advanced gaming, Augmented Reality 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
  - neiltrevett@nvidia.com