Khronos Open API Standards
The Foundation for Mobile Innovation
Neil Trevett, President, Khronos Group
Why Do We Need Standards?

- Defines interoperability interfaces so compelling user experiences can be created cheaply to build a mass market
  - Don’t slow growth with fragmentation that adds no value
- E.g. Wireless and IO standards
  - GSM/EDGE, UMTS/HSPA, LTE, IEEE 802.11, Bluetooth, USB …

Standards drive mobile market growth by expanding device capabilities
Khronos Connects Software to Silicon

ROYALTY-FREE, OPEN STANDARD APIs for advanced hardware acceleration

Low level silicon to software interfaces needed on every platform

Graphics, video, audio, compute, visual and sensor processing

Defines the forward looking roadmap for the silicon community

Shipping on billions of devices across multiple operating systems

Rigorous conformance tests for cross-vendor consistency

Khronos is OPEN for any company to join and participate

Acceleration APIs BY the Industry FOR the Industry
Khronos API Standards Evolution

**New API technology first evolves on high-end platforms**

**MOBILE**
- Mobile is the new platform for apps innovation. Mobile APIs unlock hardware and conserve battery life
- Apps need interoperating APIs with rich sensory inputs for advanced use cases such as Augmented Reality

**DESKTOP**
- New API technology first evolves on high-end platforms

**WEB**
- Diverse platforms - mobile, TV, embedded - means HTML5 will become increasingly important as a universal app platform

**INTEROP, VISION AND SENSORS**

- **OpenGL**, **OpenMAX**, **OpenCL**
OpenCL - Heterogeneous Computing

- A low-level, cross-platform, cross-vendor standard
  - For harnessing all system compute resources
- C Platform Layer API
  - Query, select and initialize compute devices
- Kernel Language Specification
  - Subset of ISO C99 with language extensions
  - Well-defined numerical accuracy - IEEE 754 rounding with specified max error
  - Rich built-in functions: cross, dot, sin, pow, log ...
- C Runtime API
  - Runtime or build-time compilation of kernels
  - Execute compute kernels across multiple devices
OpenGL 3D API Family Tree

**Mobile 3D**
- OpenGL ES 1.0
- OpenGL ES 1.1
- OpenGL ES 2.0
- OpenGL ES 3.0

**Desktop 3D**
- OpenGL 1.5
- OpenGL 2.0
- OpenGL 2.1
- OpenGL 3.0
- OpenGL 3.1
- OpenGL 3.2
- OpenGL 3.3
- OpenGL 3.4
- OpenGL 4.0
- OpenGL 4.1
- OpenGL 4.2
- OpenGL 4.3

- Fixed function 3D Pipeline
- Programmable vertex and fragment shaders
- ES3 is backward compatible so new features can be added incrementally
- WebGL 1.0
- WebGL-Next
- ES-Next
- GL-Next

OpenGL 4.3 is a superset of DX11
OpenGL 4.3 Compute Shaders

- Execute algorithmically general-purpose GLSL shaders
  - Can operate on uniforms, images and textures
- Process graphics data in the context of the graphics pipeline
  - Easier than interoperating with a compute API if processing ‘close to the pixel’
- Standard part of all OpenGL 4.3 implementations
  - Matches DX11 DirectCompute functionality

Examples:
- Physics
- AI Simulation
- Ray Tracing
- Imaging
- Global Illumination
### Texture Compression is Key

- Texture compression saves precious resources
  - Network bandwidth, device memory space AND device memory bandwidth
- Developers need the same texture compression EVERYWHERE
  - Otherwise portable apps - such as WebGL need multiple copies of same texture

<table>
<thead>
<tr>
<th>Texture Format</th>
<th>Platform</th>
<th>Mandated in</th>
<th>Royalty-Free</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXTC/S3TC</td>
<td>Windows</td>
<td></td>
<td></td>
<td>NOT Royalty-free. Platform Fragmentation</td>
</tr>
<tr>
<td>PVRTC</td>
<td>iOS</td>
<td></td>
<td></td>
<td>Royalty-free BUT only optional in ES. Only 4bpp</td>
</tr>
<tr>
<td>ETC1</td>
<td>Android Froyo (400M devices)</td>
<td>Mandated in Android Froyo (400M devices)</td>
<td>Royalty-free Backward compatible with ETC1 ETC2: 4bpp</td>
<td>3 channel EAC: 4 (8) bpp</td>
</tr>
<tr>
<td>ETC2/EAC</td>
<td>OpenGL ES 3.0 and OpenGL 4.3 extensions -&gt; Core once proven</td>
<td>Royalty-free</td>
<td>Best quality. Independent control of bit-rate and # channels 1 to 4 channel 1-8bpp in fine steps</td>
<td></td>
</tr>
</tbody>
</table>

**Deployment**

- **2008-2010**
- **2012-2013**
- **2014-**
ASTC - Universal Texture Standard

- Adaptive Scalable Texture Compression (ASTC)
  - Quality significantly exceeds S3TC or PVRTC at same bit rate
- Industry-leading orthogonal compression rate and format flexibility
  - 1 to 4 color components: R / RG / RGB / RGBA
  - Choice of bit rate: from 8bpp to <1bpp in fine steps
- ASTC is royalty-free and so is available to be universally adopted
  - Shipping as OpenGL/OpenGL ES extension today for industry feedback

![Original 24bpp vs. ASTC Compression 8bpp, 3.56bpp, and 2bpp images]
Native APIs for Augmented Reality

- Proprietary Vendors APIs
- Positional Sensors
- Positional and GPS Sensor Data
- Synchronization and Sensor fusion
- Camera
- Advanced Camera Control and Stream generation
- EGLStream Image streams to GPU and CPU
- Computer Vision/Tracking & Computational Photography
- Tracked features
- Position and Tracking Semantics
- Application on CPUs and GPUs
- Audio Rendering
- 3D Rendering and Video Composition on GPU
- Dataflow and synchronization
- OpenSL ES
- OpenVX
- OpenGL ES

© 2012 NVIDIA - Page 10
OpenVX

- **Vision Hardware Acceleration Layer**
  - Enables hardware vendors to implement accelerated imaging and vision algorithms
  - For use by high-level libraries or apps

- **Focus on enabling real-time vision**
  - On mobile and embedded systems

- **Diversity of efficient implementations**
  - From programmable processors, through GPUs to dedicated hardware pipelines

**Dedicated hardware can help make vision processing performant and low-power enough for pervasive ‘always-on’ use**
OpenVX Execution Flow

- OpenVX Graph for efficient execution
  - Each Node can be implemented in software or accelerated hardware
- EGL provides data and event interop - with streaming
  - BUT use of other Khronos APIs are not mandated
- VXU Utility Library provides efficient access to single nodes
  - Open source implementation

![Diagram of OpenVX execution flow](image-url)
OpenVX Participants and Timeline

- Aiming for provisional specification in 2H 2013
- Itseez is working group chair
Market Demand for Sensor Fusion API

Innovative use of growing sensor diversity

Synchronized use of multiple interoperating sensors in one app

PORTABLE apps need to be isolated from sensor and OS details

Application developers do not wish to be Sensor Fusion experts

StreamInput
A High-level Sensor Fusion API

Do NOT force the application developer to access individual sensors (unlike almost all other sensor APIs)

High-level API enables sensor vendors to drive and deliver competitive sensor fusion innovation
StreamInput - Portable Access to Sensor Fusion

Apps request semantic sensor information
StreamInput defines possible requests, e.g.
“Provide Skeleton Position” “Am I in an elevator?”

Advanced Sensors Everywhere
RGB and depth cameras, multi-axis motion/position, touch and gestures, microphones, wireless controllers, haptics keyboards, mice, track pads

Apps Need Sophisticated Access to Sensor Data
Without coding to specific sensor hardware

Processing graph provides sensor data stream
Utilizes optimized, smart, sensor middleware
Apps can gain ‘magical’ situational awareness
Leveraging Proven Native APIs into HTML5

- Leverage native API investments into the Web
  - Faster API development and deployment
  - Familiar foundation reduces developer learning curve
- Khronos and W3C exploring liaison
  - Multiple potential joint projects

Native APIs shipping or working group underway

JavaScript API shipping or working group underway

Possible future JavaScript APIs
WebGL - 3D Browser Visualization

- JavaScript Binding to OpenGL ES 2.0
  - 3D rendering into the Canvas
- Shipping on desktop browsers last year
  - Mobile browsers this year
- Enables the browser to access the full power the GPU
WebCL - Parallel Computing for the Web

- JavaScript bindings to OpenCL APIs
  - Enables initiation of Kernels written in OpenCL C within the browser

http://www.youtube.com/user/SamsungSISA#p/a/u/1/9Ttux1A-Nuc
Busting Some Standardization Myths

- “Standards are slow to develop”
  - Time to productive multi-vendor ecosystem is the key rather than minimizing time to a proprietary specification
  - Cooperative refinement can be highly effective - OpenCL 1.0 took just 6 months - intensive cooperation

- “If I participate in standards I ‘lose’ my IP”
  - Khronos IP Framework fully protects Members IP and the specification - Members agree not to assert claims against other Members for essential IP in conformant imple

- “Using a Standard means that I can’t differentiate”
  - Well designed standards enable strong implementation diversity

- “Standards are boring”
  - An effective standard is industry coming together to solve real issues
In Summary

- APIs are key to enable compelling applications on advanced hardware - APIs developed on high-end hardware are now enabling mobile devices
- APIs no longer exist alone - they interoperate and provide input AND output processing to form a complete platform for advanced content
- Significant cooperation underway between native and Web APIs to bring advanced visual computing to HTML5

- Khronos is driving open standards for hardware acceleration
  Participate, change the industry AND get the inside edge for your products!