A Vision for OpenCL

Neil Trevett
Vice President Mobile Ecosystem at NVIDIA
President of Khronos and Chair of the OpenCL Working Group
A Vision for OpenCL

- Where are we?
- How did we get here?
- Where might we be going next?

“Know from whence you came. If you know whence you came, there are absolutely no limitations to where you can go.”
— James Baldwin

“The best way to predict the future is to invent it.”
— Alan Kay

OpenCL has definitely broken out of being ‘yet another competing standard for parallel programming.’ How will it continue to add value to the industry?

— [https://xkcd.com/927/](https://xkcd.com/927/)
OpenCL Roadmap

- What markets has OpenCL been aimed at?
- What problems is OpenCL solving?
- How will OpenCL need to adapt in the future?
OpenCL Desktop Usage

- Broad commercial uptake of OpenCL
  - Mainly imaging, video and vision processing

- “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

http://streamcomputing.eu/blog/2013-12-28/professional-consumer-media-software-opencl/
OpenCL Implementations

Desktop

1.0 | May09
1.0 | Aug09
1.0 | May10
1.0 | May09
1.1 | Jun10
1.1 | Aug10
1.1 | Feb11
1.1 | Mar11
1.2 | May12
1.2 | Dec12
1.2 | Jun12
1.2 | Sep13
1.2 | Apr14

Mobile

1.0 | Jan10
1.0 | Feb11
1.1 | Nov12
1.1 | May13
1.1 | Apr12

FPGA

1.0 | Aug09

OpenCL 1.0
OpenCL 1.1
OpenCL 1.2
OpenCL 2.0

Speciation
Speciation
Speciation
Speciation

Dec08
Jun10
Nov11
Nov13

2008
2010
2011
2013

© Copyright Khronos Group 2014 - Page 6
Khronos Foundational APIs

A successful standard *enables* and *encourages* innovation in implementation *and* usage

**Implementer Innovation**

**Developer Innovation**

OpenCL

Market Momentum...
Applications, libraries and frameworks that find OpenCL acceleration can deliver a better end-user experience

Deliver the lowest level abstraction possible API that still provides portability - this is functionality needed on every platform

Market Momentum...
Many devices competing on performance and power to tap into the value of OpenCL content

Applications, libraries and frameworks that find OpenCL acceleration can deliver a better end-user experience

A successful standard *enables* and *encourages* innovation in implementation *and* usage
OpenCL as Parallel Language Backend

- **WebCL**: JavaScript binding for initiation of OpenCL C kernels
- **Halide**: Language for image processing and computational photography
- **C++ AMP**: MulticoreWare open source project on Bitbucket
- **aparapi**: Embedded array language for Haskell
- **Java language extensions for parallelism**: River Trail Language extensions to JavaScript
- **Compiler directives for Fortran, C and C++**: OpenACC
- **Python wrapper around OpenCL**: PyOpenCL
- **High level language for GPU programming**: Harlan

OpenCL provides vendor optimized, cross-platform, cross-vendor access to heterogeneous compute resources.
<table>
<thead>
<tr>
<th>Library Name</th>
<th>Overview</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerate</td>
<td>accelerate: An embedded language for accelerated array processing</td>
<td><a href="http://hackage.haskell.org/package/accelerate">http://hackage.haskell.org/package/accelerate</a></td>
</tr>
<tr>
<td>amgCL</td>
<td>Simple and generic algebraic multigrid framework</td>
<td><a href="https://github.com/ddemidov/amgcl">https://github.com/ddemidov/amgcl</a></td>
</tr>
<tr>
<td>Aparapi</td>
<td>API for data parallel Java. Allows suitable code to be executed on GPU via OpenCL.</td>
<td><a href="https://code.google.com/p/aparapi/">https://code.google.com/p/aparapi/</a></td>
</tr>
<tr>
<td>ArrayFire</td>
<td>Array-based function library</td>
<td><a href="https://www.accelereyes.com/products/arrayfire">https://www.accelereyes.com/products/arrayfire</a></td>
</tr>
<tr>
<td>Bolt</td>
<td>Bolt C++ Template Library</td>
<td><a href="https://github.com/hsa-libraries/Bolt/releases/tag/v1.1GA">https://github.com/hsa-libraries/Bolt/releases/tag/v1.1GA</a></td>
</tr>
<tr>
<td>Boost.Compute</td>
<td>Boost.Compute is a GPU/parallel-computing library for C++ based on OpenCL</td>
<td><a href="https://github.com/kylelutz/compute">https://github.com/kylelutz/compute</a></td>
</tr>
<tr>
<td>Bullet Physics</td>
<td>Bullet Physics: OpenCL accelerated Rigid Body Pipeline</td>
<td><a href="http://bulletphysics.org/wordpress/?p=381">http://bulletphysics.org/wordpress/?p=381</a></td>
</tr>
<tr>
<td>C++ AMP</td>
<td>CLANG/LLVM based C++AMP 1.2 standard and transforms it into OpenCL-C</td>
<td><a href="https://bitbucket.org/multicoreware/cppamp-driver-ng/wiki/Home">https://bitbucket.org/multicoreware/cppamp-driver-ng/wiki/Home</a></td>
</tr>
<tr>
<td>BLAS</td>
<td>BLAS implementation</td>
<td><a href="https://github.com/ctMathLibraries/cblas">https://github.com/ctMathLibraries/cblas</a></td>
</tr>
<tr>
<td>clFFT</td>
<td>OpenCL FFT Library</td>
<td><a href="https://github.com/ctMathLibraries/clFFT">https://github.com/ctMathLibraries/clFFT</a></td>
</tr>
<tr>
<td>cIMAGMA</td>
<td>cIMAGMA 1.1 is an OpenCL port of MAGMA</td>
<td><a href="http://id.cs.utk.edu/magma/software/view.html?id=190">http://id.cs.utk.edu/magma/software/view.html?id=190</a></td>
</tr>
<tr>
<td>clpp</td>
<td>OpenCL Parallel Primitives Library</td>
<td><a href="https://code.google.com/p/clpp/">https://code.google.com/p/clpp/</a></td>
</tr>
<tr>
<td>clSpMV</td>
<td>Sparse Matrix Solver</td>
<td><a href="http://www.eecs.berkeley.edu/~subrian/clSpMV.html">http://www.eecs.berkeley.edu/~subrian/clSpMV.html</a></td>
</tr>
<tr>
<td>Clyther</td>
<td>Python just-in-time specialization engine for OpenCL</td>
<td><a href="http://sross.sross.github.io/Clyther/">http://sross.sross.github.io/Clyther/</a></td>
</tr>
<tr>
<td>Codeplay Math Lib</td>
<td>OpenCL 1.2 Math library</td>
<td><a href="https://www.codeplay.com/products/math/">https://www.codeplay.com/products/math/</a></td>
</tr>
<tr>
<td>Concord</td>
<td>C++ Heterogenous Programming Framework ( Support OpenCL 1.2 ) TBB like</td>
<td><a href="https://github.com/IntelLabs/iHRC/">https://github.com/IntelLabs/iHRC/</a></td>
</tr>
<tr>
<td>COPRTHR</td>
<td>CO-Processor THRReads (COPRTHR) SDK</td>
<td><a href="http://www.browndeerotechnology.com/coprthr.htm">http://www.browndeerotechnology.com/coprthr.htm</a></td>
</tr>
<tr>
<td>DL-Data Layout</td>
<td>DL Enables Optimized Data Layout Across Heterogeneous Processors</td>
<td><a href="http://www.multicorewareinc.com/dl.html">http://www.multicorewareinc.com/dl.html</a></td>
</tr>
<tr>
<td>ForOpenCL</td>
<td>Fortran to OpenCL tool</td>
<td><a href="http://sourceforge.net/projects/fortran-parser/files/ForOpenCL/">http://sourceforge.net/projects/fortran-parser/files/ForOpenCL/</a></td>
</tr>
<tr>
<td>fortranCL</td>
<td>FortranCL is an OpenCL interface for Fortran 90.</td>
<td><a href="https://code.google.com/p/fortrancl/">https://code.google.com/p/fortrancl/</a></td>
</tr>
<tr>
<td>FSCl Compiler</td>
<td>FSharp to OpenCL Compiler</td>
<td><a href="https://github.com/GabrieleCocco/FSClCompiler">https://github.com/GabrieleCocco/FSClCompiler</a></td>
</tr>
<tr>
<td>GALAS</td>
<td>GPU Automatically Tuned Linear Algebra Software ( Project looks stalled)</td>
<td><a href="https://github.com/cjang/GALAS">https://github.com/cjang/GALAS</a></td>
</tr>
<tr>
<td>GMAC</td>
<td>Global Memory for Accelerators</td>
<td><a href="http://www.multicorewareinc.com/gmac.html">http://www.multicorewareinc.com/gmac.html</a></td>
</tr>
<tr>
<td>GPUlib</td>
<td>Iterative sparse solvers</td>
<td><a href="http://www.txcorp.com/">http://www.txcorp.com/</a></td>
</tr>
<tr>
<td>gpumatrices</td>
<td>A matrix and array library on GPU with interface compatible with Eigen.</td>
<td><a href="https://github.com/rudaoshi/gpumatrices">https://github.com/rudaoshi/gpumatrices</a></td>
</tr>
<tr>
<td>GPUVerify</td>
<td>GPUVerify is a tool for formal analysis of GPU kernels written in OpenCL</td>
<td><a href="http://multicore.doc.ic.ac.uk/tools/GPUVerify/">http://multicore.doc.ic.ac.uk/tools/GPUVerify/</a></td>
</tr>
<tr>
<td>Halide</td>
<td>Halide Programming language for high-performance image processing</td>
<td>halide-lang.org</td>
</tr>
<tr>
<td>Harlan</td>
<td>Harlan: A Scheme-Based GPU Programming Language</td>
<td><a href="https://github.com/eholk/harlan">https://github.com/eholk/harlan</a></td>
</tr>
<tr>
<td>HOOpenCL</td>
<td>Haskell OpenCL Wrapper API</td>
<td><a href="https://github.com/bgaster/hopenocl">https://github.com/bgaster/hopenocl</a></td>
</tr>
<tr>
<td>libCL</td>
<td>C++ Generic parallel algorithms library</td>
<td><a href="http://www.libcl.org/">http://www.libcl.org/</a></td>
</tr>
<tr>
<td>MUMPS</td>
<td>Direct Sparse solver</td>
<td><a href="http://graal.ens-lyon.fr/MUMPS/">http://graal.ens-lyon.fr/MUMPS/</a></td>
</tr>
<tr>
<td>Octave</td>
<td>Octave acceleration via OpenCL</td>
<td><a href="http://indico.cern.ch/event/93877/session/13/contribution/89/material/slides/0.pdf">http://indico.cern.ch/event/93877/session/13/contribution/89/material/slides/0.pdf</a></td>
</tr>
</tbody>
</table>

**Courtesy: AMD**
Libraries and Languages using OpenCL #2

- **Open Fortran Parser**: ANTLR-based parsing tools that support the Fortran 2008 standard

- **OpenACC to OpenCL Compiler**: Rose based OpenACC to OpenCL Compiler
  - [https://github.com/tristanvdb/OpenACC-to-OpenCL-Compiler](https://github.com/tristanvdb/OpenACC-to-OpenCL-Compiler)

- **OpenCL.jl**: Julia OpenCL 1.2 bindings
  - [https://github.com/jakebolewski/OpenCL.jl](https://github.com/jakebolewski/OpenCL.jl)

- **OpenCLIPP**: OpenCL Integrated Performance Primitives - A library of optimized OpenCL image processing functions
  - [https://github.com/CRVI/OpenCLIPP](https://github.com/CRVI/OpenCLIPP)

- **OpenCLLink**: Mathematica to use the OpenCL parallel computing language

- **OpenCV-CL**: OpenCL accelerated OpenCV

- **OpenHMPP**: Directive-based OpenACC and OpenHMPP Source to OpenCL compiler

- **Pardiso**: Direct Sparse solver
  - [http://www.pardiso-project.org/](http://www.pardiso-project.org/)

- **PETSc**: Portable, Extensible Toolkit for Scientific Computation
  - [https://github.com/geomcomp/pencil](https://github.com/geomcomp/pencil)

- **PyOpenCL**: OpenCL parallel computation API from Python
  - [http://mathema.tician.de/software/pyopencl/](http://mathema.tician.de/software/pyopencl/)

- **QT with OpenCL**: Using OpenCL with QT

- **RaijinCL**: library for matrix operations for OpenCL
  - [http://www.rajincl.org/](http://www.rajincl.org/)

- **Riventrail**: JavaScript which supports Data Parallelism via OpenCL
  - [https://github.com/rivertrail/riventrail/wiki](https://github.com/rivertrail/riventrail/wiki)

- **RNG**: Random number generation for parallel computations
  - [http://www.iro.umontreal.ca/~lecuyer/index.html](http://www.iro.umontreal.ca/~lecuyer/index.html)

- **ROpenCL**: Parallel Computing for R Using OpenCL

- **Rust-OpenCL**: OpenCL bindings for Rust
  - [https://github.com/roccui/Rust-OpenCL](https://github.com/roccui/Rust-OpenCL)

- **SkelCL**: SkelCL is a library providing high-level abstractions for alleviated programming of modern parallel heterogeneous platforms
  - [https://github.com/skelcl/skelcl](https://github.com/skelcl/skelcl)

- **SnuCL**: SnuCL naturally extends the original OpenCL semantics to the heterogeneous cluster
  - [http://snucn.nchu.edu.tw/](http://snucn.nchu.edu.tw/)

- **SpeedIT 2.4**: OpenCL based OpenFoam acceleration library

- **streamscan**: StreamScan: Fast Scan Algorithms for GPUs without Global Barrier Synchronization-
  - [https://code.google.com/p/streamscan/](https://code.google.com/p/streamscan/)

- **SuperLU**: Direct Sparse solver

- **TM-Task Management**: Heterogeneous Task Scheduling and Management
  - [http://www.multicorewareinc.com/tm.html](http://www.multicorewareinc.com/tm.html)

- **Trilinos**: Building blocks for the development of scientific applications; constructing and using sparse and dense matrices

- **VexCL**: VexCL is a C++ vector expression template library for OpenCL/CUDA
  - [http://ddemidov.github.io/vexcl](http://ddemidov.github.io/vexcl)

- **ViennaCL**: open-source linear algebra library for computations on many-core architectures (GPUs, MIC) and multi-core CPUs
  - [http://viennACL.sourceforge.net/](http://viennACL.sourceforge.net/)

- **VirtualCL**: VirtualCL (VCL) cluster platform is a wrapper for OpenCL™
  - [http://www.mosix.cs.huji.ac.il/~txt_vcl.html](http://www.mosix.cs.huji.ac.il/~txt_vcl.html)

- **VOBLA**: Vehicle for Optimized Basic Linear Algebra - Optimized Basic Linear Algebra DSL
  - [https://github.com/carpert项目/vobla](https://github.com/carpert项目/vobla)

- **VOCCL**: Virtualized OpenCL environment

- **VSI/Pro™**: VSI/Pro implementation in OpenCL

- **WAMS**: Algebraic Multigrid Solver using state-of-the-art wavelet preconditioners- solver for sparse linear equations

**Courtesy: AMD**
Widening OpenCL Ecosystem

- OpenCL C Kernel Source
- SPIR Generator (e.g. patched Clang)
- OpenCL C Runtime
  - OpenCL run-time can consume SPIR

SPIR
- Standard Portable Intermediate Representation
- CLOSE COOPERATION WITH LLVM COMMUNITY
  - SPIR 1.2 Released January 2014 (uses LLVM 3.2)

Alternative Language for Kernels

Alternative Language for Kernels

Alternative Language for Kernels

Apps and Frameworks

SYCL
- Programming abstraction that combines portability and efficiency of OpenCL with ease of use and flexibility of C++
- SYCL 1.0 Provisional Released March 2014

https://github.com/KhronosGroup/SPIR

SPIR is easier compiler target than C

Device X
Device Y
Device Z
The Future is Mobile

- Mobile SOCs now beginning to need more than just ‘GPU Compute’
  - Multi-core CPUs, GPUs, DSPs, ISPs, specialized hardware blocks

- OpenCL can provide a single programming framework for all processors on a SOC
  - OpenCL 1.2 Built-in Kernels for custom HW
  - How should OpenCL embrace DSPs?
  - Appropriate precision/resource demands?

- What are the key mobile applications that will drive use of heterogeneous SOCs?
  - What will it mean for compute APIs?
Mobile Visual Computing

- Compute acceleration is most useful when a LOT of data needs to be processed
- On Mobile - where will you get a LOT of data? The camera!

- Computational Photography and Videography
- Face, Body and Gesture Tracking
- 3D Scene/Object Reconstruction
- Augmented Reality
Hyper Realistic AR Using Visual Compute

Augmented Reality before compute acceleration

Augmentations are not convincingly integrated into the scene in terms of positioning or lighting

Courtesy Metaio
http://www.youtube.com/watch?v=xw3M-TN0o44&feature=related

After - Real-time demo on CUDA laptop

Virtual glass

Significant ray-casting and light field reconstruction processing enables augmentations to appear realistically in the scene

High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence
P. Kán, H. Kaufmann, Institute of Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria
https://www.youtube.com/watch?v=i2MEwVZzDaA
APIs for Mobile Compute

**GPU Compute Shaders (OpenGL 4.4 and OpenGL ES 3.1)**
- Pervasively available on almost any mobile device or OS
- Easy integration into graphics apps - no API interop needed
- Program in GLSL not C
- Limited to acceleration on a single GPU

**C/C++ Language Integrated GPU Compute**
- Easy programmability and low level access to GPU: Unified Memory, Virtual Addressing,
  Mature and optimized tools and performance
- Extensive compute and imaging libraries available (NPP, cuFFT, cuBLAS, cuda-gdb, nvprof etc.)
- NVIDIA only, GPU only

**General Purpose Heterogeneous Programming Framework**
- Flexible, low-level access to any devices with OpenCL compiler
- Open standard for any device or OS - being used as backend by many languages and frameworks
- Single programming and run-time framework for CPUs, GPUs, DSPs, hardware
- Needs full compiler stack and IEEE precision

**Easy, High-level Compute Offload from Java**
- C99 based kernel language for simple offload from Java apps to CPU and GPU
- JIT Compilation provide host and device portability
- Android only
- Limited control over acceleration configuration
RenderScript and OpenCL

- RenderScript and OpenCL do not directly compete
  - RS addressing very different needs to OpenCL - at a different level in the stack

- RenderScript designed for 99% of Android developers - using Java
  - Code critical sections as native C - automatic offload to CPU/GPU
  - Programmer Simplicity and Portability across 1,000’s Android handsets
  - Future - Dynamic load balancing through integration with Android instrumentation and power management systems

- BUT - other types of developer need OpenCL-class control in native code
  - Middleware engines: Unity, Epic Unreal, metaio AR, Bullet Physics ...
  - Leading edge apps: real-time video/vision/camera
  - OEM functionality: e.g. camera pipeline
  - These are the developers/apps/engines that hardware vendors want for differentiation

OpenCL on Android can enable specialized access to native acceleration and be an effective platform for RenderScript innovation
Vision Processing Power Efficiency

- GPUs are more power efficient than CPUs at vision acceleration
  - When exploiting data parallelism can be x10 as efficient - and getting better!
- SOCs have space for transistors - but can’t turn on at same time!
  - Would exceed Thermal Design Point of mobile devices
- Dedicated units can further increase locality and parallelism for efficiency
  - Dark Silicon - specialized hardware - only turned on when needed
- Ultra-low power vision scanners will become essential for wearables
  - Sensor and visual awareness to trigger full vision acceleration subsystems

Enabling new mobile vision-based experiences requires pushing computation onto GPUs and dedicated hardware
OpenVX - Efficient Vision Acceleration

- **Khronos open-standard - out-of-the-Box vision framework**
  - Focus on low-power, real-time mobile and embedded vision acceleration

- **Performance portability across diverse processor architectures**
  - ISPs, Dedicated vision blocks, DSPs and DSP arrays, GPUs, Multi-core CPUs

- **Suited for low-power, always-on acceleration**
  - Can run on dedicated hardware - no compiler, CPUs or GPUs required

- **Complementary to OpenCV**
  - Which is great for prototyping

Directed graphs of vision operators provide opportunity for optimizing performance and power

Each Node can be implemented in software or accelerated hardware
Nodes may be fused to eliminate memory transfers
Processing can be tiled to keep data entirely in local memory/cache

Example OpenVX Graph
OpenVX and OpenCL are Complementary

<table>
<thead>
<tr>
<th>Use Case</th>
<th>General Heterogeneous programming</th>
<th>Domain targeted Vision processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of Use</td>
<td>General-purpose math libraries with no built-in vision functions</td>
<td>Fully implemented vision operators and framework ‘out of the box’</td>
</tr>
<tr>
<td>Architecture</td>
<td>Language-based - needs online compilation</td>
<td>Library-based - no online compiler required</td>
</tr>
<tr>
<td>Target Hardware</td>
<td>‘Exposed’ architected memory model - can impact performance portability</td>
<td>Abstracted node and memory model - diverse implementations can be optimized for power and performance</td>
</tr>
<tr>
<td>Precision</td>
<td>Full IEEE floating point mandated</td>
<td>Minimal floating point requirements - optimized for vision operators</td>
</tr>
</tbody>
</table>

It is possible to use OpenCL to build OpenVX Nodes on programmable devices
BUT - do we need definition of an efficient, vision-capable OpenCL Device?
Precisely defined precision for image and vision operations?
Need for Camera Control API

• We have choice of APIs for image and vision image processing
  - BUT no open standard API for camera control to FEED these APIs!

• Need advanced control of ISP and camera subsystem
  - Generate sophisticated image stream for advanced imaging & vision apps

• Khronos Camera Control API in development!
  - Advanced, high-frequency burst control of camera and sensor operation

• EGLStreams provides efficient streaming of images between APIs
  - OpenCL needs efficient handling of EGLStreams

Khronos Camera Control API
Defines control of Sensor, Color Filter Array
  Lens, Flash, Focus, Aperture

Auto Exposure (AE)
Auto White Balance (AWB)
Auto Focus (AF)

Image Signal Processor (ISP)

EGLStreams

OpenCL needs enqueing mechanisms to process and image stream without Host intervention?

Image/Vision Applications

© Copyright Khronos Group 2014 - Page 20
Mixamo - Avatar Videoconferencing

- Real time facial animation capture on mobile - ported directly from PC
- Animate an avatar while conferencing
- Full GPU acceleration of vision processing using OpenCL
Embedded Compute

• Embedded computing will be everywhere
  - Automotive - ADAS
  - Sensor processing - IOT
  - Etc. etc...

• Android/Linux increasingly popular embedded OS
  - Already have OpenCL support

• Many embedded processors will be tiny
  - But will need heterogeneous processing
  - But won’t need IEEE floating point

• Security will be key for many embedded apps
  - Life Critical apps

• Software Certification
  - Safety Critical subset - like OpenGL SC?

Will OpenCL reach down into these new resource constrained markets?
Web Acceleration APIs

- Khronos and W3C liaison for Web APIs
  - Leverage proven native APIs
  - Fast API development/deployment
  - Designed by hardware community
  - Familiar foundation reduces developer learning curve

WebVX?
Vision Processing
WebStream?
Sensor Fusion
WebCAM(!)
Camera control and video processing

JavaScript
Native

Native APIs shipping or Khronos working group
JavaScript API shipping, acceleration being developed or work underway
Possible future JavaScript APIs or acceleration

Mobile Developer Mindshare, Q3 2013
% of developers using each platform (n = 5,270)

Sources: Developer Economics Q3 2013 - State of the Developer Nation
www.developer economics.com/q313 | Licensed under Creative Commons Attribution 3.0 License
WebCL: Heterogeneous Computing for the Web

- WebCL 1.0 specification officially finalized at GDC March 2014
  - [https://www.khronos.org/webcl](https://www.khronos.org/webcl)
- WebCL defines JavaScript binding to the OpenCL APIs
  - Enables initiation of Kernels written in OpenCL C within the browser
- Typical Use Cases
  - 3D asset codecs, video codecs and processing, imaging and vision processing
  - Physics for WebGL games, Online data visualization, Augmented Reality
WebGL/WebCL Ecosystem

Content downloaded from the Web

Middleware can make WebGL and WebCL accessible to non-expert programmers
E.g. three.js library: [http://threejs.org/](http://threejs.org/) used by majority of WebGL content

Browser provides WebGL and WebCL
Alongside other HTML5 technologies
No plug-in required

OS Provided Drivers
WebGL uses OpenGL ES 2.0 or Angle for OpenGL ES 2.0 over DX9
WebCL uses OpenCL 1.X

Low-level APIs provide a powerful foundation for a rich JavaScript middleware ecosystem
Open Source Implementations and Resources

- WebCL Conformance Framework and Test Suite (contributed by Samsung)
  - [https://github.com/KhronosGroup/WebCL-conformance/](https://github.com/KhronosGroup/WebCL-conformance/)

- Nokia - Firefox build with integrated WebCL
  - Firefox extension, open sourced May 2011 (Mozilla Public License 2.0)
  - [https://github.com/toaarnio/webcl-firefox](https://github.com/toaarnio/webcl-firefox)

- Samsung - uses WebKit, open sourced June 2011 (BSD)
  - [https://github.com/SRA-SiliconValley/webkit-webcl](https://github.com/SRA-SiliconValley/webkit-webcl)

- Motorola Mobility - uses Node.js, open sourced April 2012 (BSD)
  - [https://github.com/Motorola-Mobility/node-webcl](https://github.com/Motorola-Mobility/node-webcl)

- AMD - uses Chromium (open source)
  - [https://github.com/amd/Chromium-WebCL](https://github.com/amd/Chromium-WebCL)

Based on Iñigo Quilez, Shader Toy

Based on Apple QJulia

Based on Iñigo Quilez, Shader Toy

http://fract.ured.me/
WebCL - Parallel Computation for the Web

http://www.youtube.com/user/SamsungSISA#p/a/u/1/9Ttux1A-Nuc
WebCL - Designed-in Architectural Security

- Leverages OpenCL 1.2 robustness/security extensions
  - Context Termination: to prevent DoS from long running kernels
  - Memory Initialization: no leakage from out of bounds memory access

- API and Language Restrictions
  - Not supported: structures as Kernel arguments, Kernel names > 256 characters, mapping of CL memory objects into host memory, program binaries, some OpenCL API calls and built-ins

- WebCL Kernel Validator  
  [https://github.com/KhronosGroup/webcl-validator](https://github.com/KhronosGroup/webcl-validator)
  - Open source - provided as a “library API” for easy integration into browsers
  - Parses and validates kernel code against specification
  - Initializes local/private memory if underlying OpenCL implementation does not
  - Tracks memory allocations and traces valid ranges for reads and writes
  - Run time checks to make all memory accesses safe
Wrap Up - Broadening OpenCL’s Impact

SPIR and SYCL
Evolve C++ support into core
Converge SPIR and LLVM

Foundation API
Evolve memory and Execution Models for all markets

Lots of OpenCL Devices
Drive availability of latest API version.
Minimize extensions and optional features

Lots of apps, languages and frameworks
Better debug, profiling and trans-API Interop

WebCL
Needs pervasive OpenCL including Android

Heterogeneous Mobile SOCS
Drive mobile platform adoption
Fully Embrace DSPs and Dedicated Hardware

Vision Processing
Vision friendly precision and stream handling

Embedded
Precision and resource handling for small processors

Security
Memory and execution security mechanisms

Safety Critical
Functionality subset for certification

Now is a great time for input and feedback
We do read the forums!
Or join Khronos and have a voice at the working group!