SOC Programming Tutorial
Hot Chips 2012
Neil Trevett
Khosronos President
Welcome!

• An exploration of SOC capabilities from the programmer’s perspective
  - How is mobile silicon interfacing to mobile apps?

• Overview of acceleration APIs on today’s mobile OS
  - And how they can be used to optimize performance and power

• Focus on mobile innovation hotspots
  - Vision and gesture processing, Augmented Reality, Sensor Fusion, Computational Photography, 3D Graphics

• Highlight silicon-level opportunities and challenges still to be solved
  - While exploring the state of the art in mobile programming

SOC = ‘System On Chip’
Minus memory and some peripherals
# Speakers

<table>
<thead>
<tr>
<th>Session</th>
<th>Speaker</th>
<th>Company</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connecting Mobile SOC Hardware to Apps</td>
<td>Neil Trevett</td>
<td>Khronos</td>
<td>Khronos President and NVIDIA VP of Mobile Content</td>
</tr>
<tr>
<td>Break</td>
<td></td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>Camera and Video</td>
<td>Sean Mao</td>
<td>ArcSoft</td>
<td>VP Marketing, Advanced Imaging Technologies</td>
</tr>
<tr>
<td>Vision and Gesture Processing</td>
<td>Itay Katz</td>
<td>Eyesight</td>
<td>Co-Founder &amp; CTO</td>
</tr>
<tr>
<td>Augmented Reality</td>
<td>Ben Blachnitzky</td>
<td>Metaio</td>
<td>Director of R&amp;D</td>
</tr>
<tr>
<td>Sensor Fusion</td>
<td>Jim Steele</td>
<td>Sensor Platforms</td>
<td>VP Engineering</td>
</tr>
<tr>
<td>3D Gaming</td>
<td>Daniel Wexler</td>
<td>the11ers</td>
<td>CXO</td>
</tr>
<tr>
<td>Panel Session</td>
<td></td>
<td>All Speakers</td>
<td></td>
</tr>
</tbody>
</table>
Khronos Connects Software to Silicon

- Khronos APIs define processor acceleration capabilities
  - Graphics, video, audio, compute, vision and sensor processing

APIs developed today define the functionality of platforms and devices tomorrow
APIs BY the Industry FOR the Industry

• **Khronos defines APIs at the software silicon interface**
  - Low-level "Foundation" functionality needed on every platform

• **Khronos standards have strong industry momentum**
  - 100s of man years invested by industry experts
  - 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**
  - Standards are cooperative – one company, one vote
  - Proven legal and IP framework for industry cooperation
  - Khronos membership fees to cover expenses

• **Khronos standards are FREE to use**
  - Members agree to not request royalties
Over 100 members – any company worldwide is welcome to join

Board of Promoters
New API technology first evolves on high-end platforms.

Mobile is the new platform for apps innovation. Mobile APIs unlock hardware and conserve battery life.

Apps embrace mobility’s unique strengths and need complex, interoperating APIs with rich sensory inputs e.g. Augmented Reality.

Diverse platforms – mobile, TV, embedded – means HTML5 will become increasingly important as a universal app platform.
A New Era in Computing

1990’s

PC

2000’s

Internet

2010’s

Mobile Computing
20 Years Faster to 100M Per Year

Cumulative Shipments

- iOS & Android
- MacOS & Windows

Units in Millions

Year 1  Year 2  Year 3  Year 4  Year 5  Year 6  Year 7  Year 8  Year 9

Source: Gartner, Apple, NVIDIA
The Largest Device Market Ever

- IDC - 1.8 billion mobile phones will ship in 2012
  - By the end of 2016, 2.3 billion mobile phones will ship per year

Smart phones account for approximately half of total phone market
ARM is Licensable and Pervasive

Source: ARM, Mercury Research
Mobile Performance Increases

CPU/GPU AGGREGATE PERFORMANCE

Core i5

2011

2012

2013

2014

TEGRA 2

Tegra 3 (KAL-EL)

WAYNE

LOGAN

STARK

TEGRA 2

CPU

GPU

TEGRA 3

Tegra 3 (KAL-EL)

WAYNE

LOGAN

STARK

75x

50x

10x

5x

Core i5

Core 2 Duo

© Copyright Khronos Group 2012 | Page 13
Power is the New Design Limit

- The Process Fairy keeps bringing more transistors
  - Transistors are getting cheaper
- The End of Voltage Scaling
  - The Process Fairy isn’t helping as much on power as in the past

In the Good Old Days
Leakage was not important, and voltage scaled with feature size

\[
\begin{align*}
  L' &= L/2 \\
  V' &= V/2 \\
  E' &= CV^2 = E/8 \\
  f' &= 2f \\
  D' &= 1/L^2 = 4D \\
  P' &= P
\end{align*}
\]

Halve L and get 4x the transistors and 8x the capability for the same power

The New Reality
Leakage has limited threshold voltage, largely ending voltage scaling

\[
\begin{align*}
  L' &= L/2 \\
  V' &= \sim V \\
  E' &= CV^2 = E/2 \\
  f' &= \sim 2f \\
  D' &= 1/L^2 = 4D \\
  P' &= 4P
\end{align*}
\]

Halve L and get 4x the transistors and 8x the capability for 4x the power!!
Mobile Thermal Design Point

- **4-5” Screen takes 250-500mW**
- **7” Screen takes 1W**
- **10” Screen takes 1-2W**
  - Resolution makes a difference!
  - The iPad3 screen takes up to 8W

Max system power before thermal failure:
- **2-4W**
- **4-7W**
- **6-10W**
- **30-90W**

Even as battery technology improves - these thermal limits remain.
Apps and Power

- Much more expensive to MOVE data than COMPUTE data

- Process improvements WIDEN the gap
  - 10nm process will increase ratio another 4X

- Energy efficiency must be key metric during silicon AND app design
  - Awareness of where data lives, where computation happens, how is it scheduled
Energy Optimization Opportunities

• Dark Silicon
  - Lots of space for transistors – just can’t turn them all on at same time
  - Multiple specialized hardware units that are only turned on when needed
  - Increase locality and parallelism of computation to save power compared to programmable processors

• Dynamic and feedback-driven software power optimization
  - Instrumentation for energy-aware compilers and profilers
  - Most compilers just look at one thread, take a more global view
  - Power optimizing compiler back-end / installers

• Smart, holistic use of sensors and peripherals
  - Wireless modems and networks
  - Motion sensors, cameras, networking, GPS
Camera Sensor Processing

• **CPU**
  - Single processor or Neon SIMD
  - Makes heavy use of general memory
  - Non-optimal performance and power

• **GPU**
  - Many way parallelism
  - Efficient image caching into general memory
  - Programmable and flexible
  - Still significant use of cache/memory

• **Camera ISP = Image Signal Processor**
  - Scan-line-based
  - Data flows through compact hardware pipe
  - No global memory used to minimize power
  - Little or no programmability
Typical Camera ISP

- ~760 math Ops
- ~42K vals = 670Kb
- 300MHz → ~250Gops

- Computational photography apps beginning to mix non-programmable ISP processing with more flexible GPU or CPU processing
- ISP pipelines could provide tap/insertion points to/from CPU/GPU at critical pipeline points
Programmers View of Typical SOC c. 2012

Unified memory is critical departure from PC architecture

CPU Complex
1-5 Cortex A9/A15 Cores
L1 Cache

L2 Cache

Automatic scaling of frequency/voltage and # cores to meet current software load

HD Audio Engine / IO

Peripheral Busses

1-4 Display Controllers

Sensor Array

Unified Memory
Controller for 0.5-2GB DDR3L/LP-DDR3

2D/3D GPU Complex

Graphics Cache

Common virtual address space for all units - but typically not coherent

Significant programmable acceleration. OpenGL ES today. OpenCL soon. 10-50+ cores

Image Signal Processor

Video Encoder Decoder

Vision Processor (Future)

1-3 Cameras

Software configurability

Typically NO or very limited programmable functionality

Programmable? Or dedicated hardware for power efficiency?

Unified memory is critical departure from PC architecture

Unified memory is critical departure from PC architecture

Software configurability

HD Audio Engine / IO

Peripheral Busses

1-4 Display Controllers

Sensor Array

Unified Memory
Controller for 0.5-2GB DDR3L/LP-DDR3

2D/3D GPU Complex

Graphics Cache

Common virtual address space for all units - but typically not coherent

Significant programmable acceleration. OpenGL ES today. OpenCL soon. 10-50+ cores

Image Signal Processor

Video Encoder Decoder

Vision Processor (Future)

1-3 Cameras

Software configurability

Typically NO or very limited programmable functionality

Programmable? Or dedicated hardware for power efficiency?
HSA Feature Roadmap

Heterogeneous System Architecture Foundation: AMD, ARM, Imagination, TI, MediaTek

Physical Integration
- Integrate CPU & GPU in silicon
- Unified Memory Controller
- Common Manufacturing Technology

Optimized Platforms
- GPU Compute C++ support
- User mode scheduling
- Bi-Directional Power Mgmt between CPU and GPU

Architectural Integration
- Unified Address Space for CPU and GPU
- GPU uses pageable system memory via CPU pointers
- Fully coherent memory between CPU & GPU

System Integration
- GPU compute context switch
- GPU graphics pre-emption
- Quality of Service

Optimized Platforms

Current Innovation
Mobile Innovation Hot Spots

- New platform capabilities being driven by SILICON and APIs

Console-Class 3D
Performance, Quality, Controllers and TV connectivity

Vision - Camera as sensor
Computational Photography
Gesture Processing
Augmented Reality

Sensor Fusion
Devices become ‘magically’ context aware – location, usage, position

OpenCL

Media and Image Streaming
Heterogeneous Parallel Processing

WebGL
Web Apps that can be discovered on the Net and run on any platform

HTML5
OpenMAX - Media Acceleration

- Family of royalty-free, cross-platform open API standards for video, image stream and camera processing

```
Applications

OpenMAX|AL
"Application Layer"
Native Multimedia Framework

OpenMAX|IL
"Integration Layer"
Media component interfaces

Audio Codecs
e.g. MP3

Video Codecs
e.g. 264

Image Codecs
e.g. JPEG
```

Portable access to native multimedia hardware acceleration

Portable and flexible access to media components (codecs)

Sophisticated apps that provide their own media framework processing can interface to codecs directly

OpenMAX does not define CODECs - but provides codec interfaces
OpenMAX AL Streaming Media Framework

- Enables key video, image stream and camera use cases
  - Enables optimal hardware acceleration with app portability

- Create Media Objects to play and process images and video with AV sync
  - Connect to variety of input and output objects to PLAY and RECORD media

- Full range of video effects and controls
  - Including playback rate, post processing, and image manipulation

OpenMAX AL includes sophisticated camera controls

Efficient data routing for CPU processing

EGLStreams extension enables efficient transfer of image stream to GPU texture memory
Android Application Development Options

Java apps provide easy access to the Android framework and portability across CPU architectures.

Dalvik SDK
- Dalvik (Java)
- WebView Engine

Android NDK
- JNI (Java Native Interface)
- IO and system resources
- C/C++

Android NDK used for performance critical portions of applications such as 3D games, or to access advanced functionality such as extended camera controls.

```
-.apk
Installable Applications
```

```
Browser
```

```
Applications run in system browser
```

Apps as web pages - searchable and instantly updated.

Soon!

Native libraries and APIs

© Copyright Khronos Group 2012 | Page 25
Accelerating Streaming Media on Android

**DRM Movies**
- Inject encrypted elementary streams into decrypt/decode/render
- Dynamic format changes
- Support for Widevine

**HD Video Teleconferencing**
- Inject video into decode/render
- Extract video from capture/render
- Extended controls (fine-grained codec query/config, force key frames)

**Video Editing**
- Video decoding to texture
- Texture to video encoding

**Augmented Reality**
- High frame-rate, low latency camera capture to app and GPU texture
- Advanced camera control over format, ROI

**Computational Photography**
- Fast, low latency camera capture to app and GPU texture
- Advanced camera control over format, bracketed burst mode with sequenced key/value pairs

- No dynamic format change.
- Unencrypted streams only.

- No extraction - so no receiver support, no extended video controls

- Video to texture via SurfaceTexture. No texture to video encoding

- 30 FPS frame-rate but 2-3 frames latency. YUV camera data to texture via SurfaceTexture. No advanced camera controls

- Proprietary extensions for advanced camera access
Accelerating Streaming Media on Android

OpenMAX AL subset as shipped by Google does not completely fulfill any use cases.
Full OpenMAX AL with extensions fulfills all use cases for native code.
Google incrementally extending Java capabilities – but not yet meeting full requirements.
OpenCL – Heterogeneous Computing

- Native framework for programming diverse parallel computing resources
  - CPU, GPU, DSP – as well as hardware blocks(!)

- Powerful, low-level flexibility
  - Foundational access to compute resources for higher-level engines, frameworks and languages

- Embedded profile
  - No need for a separate “ES” spec
  - Reduces precision requirements

A cross-platform, cross-vendor standard for harnessing all the compute resources in an SOC

One code tree can be executed on CPUs or GPUs
OpenCL Overview

• **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 set of built-in functions: cross, dot, sin, cos, pow, log ...

• **C Runtime API**
  - Runtime or build-time compilation of kernels
  - Execute compute kernels across multiple devices

• **Memory management is explicit**
  - Application must move data from host → global → local and back
  - Implementations can optimize data movement in unified memory systems
OpenCL: Execution Model

- **Kernel**
  - Basic unit of executable code ~ C function
  - Data-parallel or task-parallel

- **Program**
  - Collection of kernels and functions
    ~ dynamic library with run-time linking

- **Command Queue**
  - Applications queue kernels & data transfers
  - Performed in-order or out-of-order

- **Work-item**
  - An execution of a kernel by a processing element
    ~ thread

- **Work-group**
  - A collection of related work-items that execute on a single compute unit ~ core
OpenCL Built-in Kernels

- Used to control non-OpenCL C-capable resources on an SOC – ‘Custom Devices’
  - E.g. Video encode/decode, Camera ISP ...

- Represent functions of Custom Devices as an OpenCL kernel
  - Can enqueue Built-in Kernels to Custom Devices alongside standard OpenCL kernels

- OpenCL run-time a powerful coordinating framework for ALL SOC resources
  - Programmable and custom devices controlled by one run-time

Built-in kernels enable control of specialized processors and hardware from OpenCL run-time
OpenCL Milestones

• Six months from proposal to released OpenCL 1.0 specification
  - Due to a strong initial proposal and a shared commercial incentive

• Multiple conformant implementations shipping on desktop
  - For CPUs and GPUs on multiple OS

• 18 month cadence between releases
  - Backwards compatibility protects software investment
Adobe at SIGGRAPH 2012

Adobe ♥ OpenCL

- Compute API supported across vendors
- Programming model familiar to C programmers
- Demonstrated performance
- Same compute kernels on CPU and GPU!

- Adobe is now active member of OpenCL working group
  - Contributing Adobe’s experience and minds to continue OpenCL evolution
OpenCL Roadmap

OpenCL-HLM (High Level Model)
Exploring high-level programming model, unifying host and device execution environments through language syntax for increased usability and broader optimization opportunities

Discussions include ways to optimize use of unified and shared virtual memory systems

Long-term Core Roadmap
Exploring enhanced memory and execution model flexibility to catalyze and expose emerging hardware capabilities

OpenCL-SPIR (Standard Parallel Intermediate Representation)
Exploring LLVM-based, low-level Intermediate Representation for code obfuscation/security and to provide target back-end for alternative high-level languages
OpenCL as Parallel Compute Foundation

- **OpenCL**: Base technology
- **HLM**: C++ syntax/compiler extensions
- **C++ AMP**: C++ syntax/compiler extensions
- **Aparapi**: Java language extensions for parallelism
- **RenderScript**: C99 kernels for Dalvik with JIT compilation for device portability
- **River Trail**: Language extensions to JavaScript
- **WebCL**: JavaScript binding to OpenCL for initiation of OpenCL C kernels

CUDA or DirectCompute may also be used as compiler targets - but OpenCL provides cross-platform, cross-vendor coverage
Many advanced photo apps today run on a single CPU
- Suboptimal performance and power

OpenCL is a platform to harness CPUs/GPUs for advanced imaging
- Even if code is ‘branchy’

“The tablet … has new multimedia capabilities, including a computational camera, which lets devs tap directly into its computational capability through new application programming interfaces such as OpenCL. That access enables next-generation use cases such as light-field cameras for mobile devices.”

HDR Imaging

Panorama Stitching

Flash / no-flash imaging
# OpenGL 20th Birthday - Then and Now

![OpenGL Logo](image_url)

**Launched OpenGL 4.3 at SIGGRAPH 2012**

## Ideas in Motion - SGI

- 1992 Reality Engine
- 8 Geometry Engines
- 4 Raster Manager boards

## 2012 Mobile

- NVIDIA Tegra 3
- Nexus 7 Android Tablet

## 2012 PC

- NVIDIA GeForce GTX 680
- Kepler GK104

<table>
<thead>
<tr>
<th></th>
<th>1992 Reality Engine</th>
<th>2012 Mobile</th>
<th>2012 PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangles / sec</td>
<td>1</td>
<td>103 (x103)</td>
<td>1800 (x1800)</td>
</tr>
<tr>
<td>Pixel Fragments / sec</td>
<td>240</td>
<td>1040 (x4.3)</td>
<td>14,400 (x60)</td>
</tr>
<tr>
<td>GigaFLOPS</td>
<td>0.64</td>
<td>15.6 (x25)</td>
<td>3090 (x4830)</td>
</tr>
</tbody>
</table>

### Performance Comparison

- **1.5KW**
- **<5W**

© Copyright Khronos Group 2012 | Page 37
OpenGL for Each Hardware Generation

- Fixed Function
- Vertex and Fragment Shaders
- Geometry Shaders
- Tessellation and Compute

Advanced 3D functionality available on PC and Foundation for the graphics stack on MAC and Linux
OpenGL 4.3 Compute Shaders

• Execute algorithmically general-purpose GLSL shaders
  - 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
OpenGL 4.3 with Compute Shaders

From Application

Element Array Buffer b
Draw Indirect Buffer b
Vertex Buffer Object b

Vertex Puller
Vertex Shader
Tessellation Control Shader
Tessellation Primitive Gen.
Tessellation Eval. Shader
Geometry Shader
Transform Feedback
Rasterization
Fragment Shader
Per-Fragment Operations
Framebuffer

From Application

Dispatch Indirect Buffer b
Image Load / Store t/b
Atomic Counter b
Shader Storage b
Texture Fetch t/b
Uniform Block b

From Application

Dispatch
Compute Shader

Legend

Fixed Function Stage
Programmable Stage
b – Buffer Binding
t – Texture Binding
Arrows indicate data flow

Pixel Unpack Buffer b
Texture Image t
Pixel Pack
Pixel Unpack Buffer b

© Copyright Khronos Group 2012 | Page 40
OpenCL and OpenGL Compute Shaders

- OpenGL compute shaders and OpenCL support different use cases
  - OpenCL provides a significantly more powerful and complete compute solution

1. Fine grain compute operations inside OpenGL
2. GLSL Shading Language
3. Execute on single GPU only

Developer driven decision

1. Full ANSI C programming of heterogeneous CPUs and GPUs
2. Utilize multiple processors
3. Coarse grain, buffer-level interop with OpenGL

Enhanced 3D Graphics apps “Shaders++”
Imaging Video Physics AI
Pure compute apps touching no pixels

Coming to mobile platforms soon!
OpenGL ES

• Streamlined subset of desktop OpenGL for embedded and mobile devices

- OpenGL ES 1.0 Spec Released
- OpenGL ES 1.1 Spec Released
- OpenGL ES 2.0 Spec Released
- OpenGL ES 2.0 Platforms Released
- OpenGL ES 2.1
- OpenGL ES 3.0 Spec Released
- OpenGL ES 3.0 Platforms Released
- OpenGL ES 3.0
- OpenGL 4.3
- OpenGL 4.3 is a superset of DX11

Fixed function 3D Pipeline
Programmable vertex and fragment shaders
ES3 is backward compatible with ES2 so new features can be added incrementally

GL4/DX11-class Capabilities come to ES ??

OpenGL 1.5 OpenGL 2.0 OpenGL 2.1

OpenGL ES 3.0 Highlights

• Better looking, faster performing games and apps – at lower power
  - Incorporates proven features from OpenGL 3.3 / 4.x
  - 32-bit integers and floats in shader programs
  - NPOT, 3D textures, depth textures, texture arrays
  - Multiple Render Targets for deferred rendering, Occlusion Queries
  - Instanced Rendering, Transform Feedback ...

• Make life better for the programmer
  - Tighter requirements for supported features to reduce implementation variability

• Backward compatible with OpenGL ES 2.0
  - OpenGL ES 2.0 apps continue to run unmodified

• Standardized Texture Compression
  - #1 developer request!
Texture Compression is Key

• Texture compression saves precious resources
  - Saves network bandwidth, device memory space AND 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>Quality</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXTC/S3TC</td>
<td>Windows</td>
<td></td>
<td></td>
<td>2008-2010</td>
</tr>
<tr>
<td>PVRTC</td>
<td>iOS</td>
<td>Android Froyo (400M devices)</td>
<td>NOT Royalty-free. Platform Fragmentation</td>
<td></td>
</tr>
<tr>
<td>ETC1</td>
<td></td>
<td>Mandated in</td>
<td>Royalty-free BUT only optional in ES. Only 4bpp</td>
<td>2012-2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No alpha support</td>
<td>ASTC</td>
</tr>
<tr>
<td>ETC2 / EAC</td>
<td>OpenGL ES 3.0 and OpenGL 4.3 Extension -&gt; Core once proven</td>
<td>MANDATED in</td>
<td>Royalty-free Best quality. Independent control of bit-rate and # channels 1 to 4 channel 1-8bpp in fine steps</td>
<td>2014-2013</td>
</tr>
<tr>
<td></td>
<td>OpenGL 4.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASTC</td>
<td>OpenGL ES 3.0 and OpenGL 4.3 Extension -&gt; Core once proven</td>
<td>Royalty-free</td>
<td></td>
<td>2014-2013</td>
</tr>
</tbody>
</table>

ETC2: 4bpp | 3 channel
EAC: 4 (8)bpp | 1(2) channel
COMBINE: RGBA 8bpp | 4 channel
Does not have 1-2 bit compression WITH ALPHA

NOT Royalty-free.
Platform Fragmentation

2014->
ASTC – Future 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 GL/ES extension today for industry feedback

Original 24bpp | ASTC Compression 3.56bpp | 2bpp
Kishonti GLBenchmark 3.0

Deferred rendering using multiple render targets (MRT) and depth textures

Occlusion query used to determine light visibility

Instanced drawing used for vehicles and particles

Kishonti “GLBenchmark 3.0” preliminary
OpenGL ES Deployment in Mobile

Use of 3D APIs in Mobile Devices
Source: Jon Peddie Research

OpenGL ES is the 3D API used in Android, iOS and almost every other mobile and embedded OS – other than Windows.

On PC – DirectX is used for most apps. On mobile the situation is reversed.
Visual-based Augmented Reality

Camera video stream sent to the compositor

Camera Tracking

Camera images used to track the camera’s location and orientation

3D Augmentation Rendering

3D augmentations composited with video stream

Other on-device Sensors
OpenCV

• **Computer vision open source project**
  - Excellent functionality - widely used in academia, fast prototyping, some products
  - Not an API definition and not managed by Khronos

• **Extensive functionality >1,000 functions**
  - Difficult for silicon vendors to provide complete acceleration

• **Traditionally runs on a single CPU**
  - Some partial acceleration projects underway: OpenCL, CUDA, Neon ...
  - E.g. MulticoreWare open source CPU/GPU enabled OpenCV over OpenCL
OpenVL

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

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

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

Dedicated hardware can help make vision processing performant and low-power enough for pervasive ‘always-on’ use
Possible Implementations of Vision Stack

- **StreamInput** combines camera vision with other sensors on device.
- **OpenCV** community can progressively port over OpenVL acceleration provided by silicon community.
- **OpenCL** can be used to implement OpenCV OR OpenVL with parallel execution on CPUs/GPUs.
- **OpenVL** does not need the camera, GUI or compute functionality in OpenCV – use API interop instead.
- **OpenVL** does not duplicate OpenCV functionality – JUST provides essential acceleration.
- **OpenVL** can be implemented using dedicated hardware.
- **Camera input** from OpenMAX AL or other camera subsystems.
- **Data and event interop** with OpenCL and OpenGL ES for display and compute processing.
Developer Requested Camera Extensions

• **Query camera information**
  - Focal length \((fx, fy)\), principal point \((cx, cy)\), skew \((s)\), image resolution \((h, w)\)
  - Spatial information of how cameras and sensors are placed on device
  - Calibration and lens distortion

• **FCAM+++ for extensive exposure parameters in single or burst mode**
  - Shutter, aperture, ISO, white balance, frame rate, focus modes, resolution
  - Preload parameters for each shot in a burst

• **ROI extraction**
  - From wide angle and fish-eye lenses

• **Data output format control**
  - Grayscale, RGB(A), YUV
  - Access to the raw data e.g. Bayer pattern
OpenSL ES – Advanced Audio

- **OpenSL ES does for audio what OpenGL ES does for graphics**
  - Advanced audio functionality from simple playback to full 3D positional audio

- **Object-based native audio API for simplicity and high performance**
  - Same object framework as OpenMAX AL
  - Reduces development time

- **Attractive alternative to open source frameworks**
  - Tightly defined specification with full conformance tests
  - Robust application portability across platforms and OS
EGLStream – Streaming Images

• **EGL** – originally embedded version of WGL
  - Abstraction layer to window systems and memory buffers

• **Role has expanded to provide API interop – data and events**
  - EGLImages – single buffers that can be passed between APIs
  - EGLStreams – provides stream of images – with AV sync
  - Cross process EGLStreams – Producer and Consumer can be in different processes for performance or security – e.g. browser compositing process

• **Android SurfaceTexture is a Java wrapper around EGLStreams**
  - Captures video decode or camera preview to OpenGL ES texture

---

OpenMAX AL Media Player is the EGLStream “Producer” and controls production of frames.

EGLStreams enables and hides details of video frame transport. Enables multiple buffering modes for different uses cases e.g.: FIFO and explicit latch/release

OpenGL ES GL_TEXTURE_EXTERNAL is the EGLStream “Consumer” and converts video format into RGB OpenGL ES texture
Portable Access to Sensor Fusion

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

Apps need sophisticated access to sensor data
Without coding to specific sensor hardware

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

Processing graph provides sensor data stream
Utilizes optimized, smart, sensor middleware
Apps can gain ‘magical’ situational awareness

Standardized node intercommunication
Universal timestamps
E.g. align samples from camera and other sensors
Example use of Khronos APIs in AR

Positional Sensors

- Positional and GPS Sensor Data
- Computer Vision and Tracking

OpenVL

Camera Processing

- Control Camera, Preprocess and generate video streams

Video stream to GPU

- Synchronization and sensor fusion

Application on CPUs

Audio Rendering

3D Rendering and Video Composition On GPU

Video TAP to Vision Subsystem

Position and Tracking Semantics
## OS API Adoption

<table>
<thead>
<tr>
<th>Platform</th>
<th>API</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>OpenGL ES 2.0</td>
<td>Shipping - Android 2.2</td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>OpenSL ES 1.0</td>
<td>(subset) Shipping – Android 2.3</td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>OpenMAX AL</td>
<td>1.0 (subset) Shipping - Android 4.0</td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>EGL</td>
<td>1.4 Shipping under SDK -&gt; NDK</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>Opera and Firefox WebGL now Chrome soon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MacOS</td>
<td>OpenGL 3.2</td>
<td>on MacOS</td>
<td></td>
</tr>
<tr>
<td>MacOS</td>
<td>OpenCL 1.1</td>
<td>on MacOS</td>
<td></td>
</tr>
<tr>
<td>iOS</td>
<td>OpenGL ES 2.0</td>
<td>on iOS</td>
<td></td>
</tr>
<tr>
<td>iOS</td>
<td>Can enable on MacOS Safari iOS5 enables WebGL for iAds</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Mobile Operating Systems

- **Microsoft Windows RT:**
  - Only Microsoft native APIs
  - HTML5 but not yet WebGL
Extended Native APIs on Android

- Native APIs can be shipped as NDK extensions before Google Adoption
  - Do not break/change existing Google APIs

- Khronos open standard APIs have strong momentum in silicon
  - Google has option to adopt into standard platform to eliminate fragmentation
  - Exposed directly – or wrapped in Java binding

- Extended APIs can be used by:
  - Bundled apps, Market apps with API selection
  - Multiple APKs behind single multi-APK SKU

Google has the choice to adopt, open, royalty-free standards and expose natively – or wrap in Java

Demonstrate the value of enabled use cases

Short-term differentiation opportunity for hardware vendors

E.g. video, camera, vision, sensor and compute APIs
Increasing diversity of devices creates a demand for a true cross OS programming platform

BUT need more than “more HTML”

Traditional Web-content

How can the Browser rapidly assimilate such diverse functionality?
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 creating close 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 on the Web – No Plug-in!

• **WebGL defines JavaScript binding to OpenGL ES 2.0**
  - Leveraging HTML 5 and uses `<canvas>` element
  - Enables a 3D context for the canvas

• **WebGL 1.0 Released at GDC March 2011**
  - Mozilla, Apple, Google and Opera working closely with GPU vendors

• **Low-level foundational API for accessing the GPU in HTML5**
  - Flexibility and direct GPU access - support higher-level frameworks and middleware
WebGL Implementation Anatomy

Content downloaded from the Web. Middleware can make WebGL accessible to non-expert 3D programmers.

Browser provides WebGL functionality alongside other HTML5 specs - no plug-in required.

OS Provided Drivers. WebGL on Windows can use Google Angle to create conformant OpenGL ES 2.0 over DX9.
WebGL – Being Used by Millions Every Day
WebGL and Security

- **WebGL is Architecturally Secure**
  - NO known WebGL security issues
  - Impossible to access out-of-bounds or uninitialized memory
  - Use of cross-origin images are blocked without permission through CORS
  - Browsers maintaining black lists - used if unavoidable GPU driver bugs discovered

- **DoS attacks and GPU hardening**
  - Draw commands can run for a long time -> unresponsive system
    - Even without loops in shaders
  - WebGL working closely with GPU vendors to categorically fix this
  - Short term: mandate ARB_robustness and associated GPU watchdog timer
  - Longer term: GPUs need robust context switch and pre-emption
WebCL – Parallel Computing for the Web

• JavaScript bindings to OpenCL APIs
  - JavaScript initiates OpenCL C Kernels on heterogeneous multicore CPU/GPU

• Stays close to the OpenCL standard
  - Maximum flexibility to provide a foundation for higher-level middleware

• Minimal language modifications for 100% security and app portability
  - E.g. Mapping of CL memory objects into host memory space is not supported

• Compelling use cases
  - Physics engines for WebGL games, image and video editing in browser

• API definition underway – public draft just released
WebCL Demo

http://www.youtube.com/user/SamsungSISA#p/a/u/1/9Ttx1A-Nuc

WebCL for Hardware-Accelerated Web Applications

Advanced Browser Technology
Samsung R&D Center
San Jose, CA
Web Apps versus Native Apps

- Mobile Apps have functional and aesthetic appeal
  - Beautiful, responsive, focused
- HTML5 with GPU acceleration can provide the same level of “App Appeal”
  - Highly interactive, rich visual design
- Using HTML5 to create ‘Web Apps’ has many advantages
  - Web app is searchable and discoverable through the web
  - Portable to any browser enabled system
  - Same code can run as app or as web page
  - Not a closed app store – no app store ‘tax’

- How soon will we be able to write apps such as Augmented Reality in HTML5?
Expanding Platform Reach for Graphics and Computation

**Desktop**
- OpenGL
- Full Profile

**Mobile**
- OpenGL ES
- Full Profile and Embedded Profile

**Web**
- WebGL
- TypedList

Production Browsers Shipping with WebGL:
- Desktop - Chrome, Firefox, Opera, Safari
- Mobile - Opera and Firefox
- Apple iOS Safari uses WebGL for iAds

WebGL on majority of production desktops now.
WebCL pervasively available on mobile in next 12 months

WebCL will start deploying in next 12-18 months

OpenCL pervasively available on mobile in next 18-24 months
Cross-OS Portability

HTML5 provides cross-platform portability. GPU accessibility through WebGL available soon on ~90% mobile systems.

Preferred development environments not designed for portability.

Native code is portable—but apps must cope with different available APIs and libraries.
Summary

• Advances in SOC silicon processing and associated APIs are enabling significant new use cases
• Holistic cooperation between hardware and software needed to deliver increasing computational loads in a fixed power budget
• Architectural shifts, such as unified memory, are creating challenges and opportunities for applications and the APIs that enable them
• Mobile operating systems and HTML5 browsers both lag in exposing the latest SOC capabilities - creates functional differentiation opportunities
• Dynamic tension between platform vendors that want captive apps and developers that benefit from cross platform portability
• Cooperative API standards working hard to eliminate roadblocks to mobile industry growth
Thank You!

We will resume at 11:10AM