Mobile OS Fragmentation

- Every handset is unique from the programmers perspective
  - Differences in OS functions, Java implementations and media functionality

Severe platform fragmentation

ISVs need to port to and support 100s (even 1000s) of source variants of each title

We need cross-platform standard APIs to de-fragment access to multimedia acceleration if the market is to be fully realized
Streaming Media Portability

• Media infrastructure portability is a multi-level industry problem
  - Media infrastructure is time-consuming and expensive to develop, integrate and program

- Application programmers need a cross-platform portable API for controlling high-level media operations
- System integrators need cross-vendor standard for media component integration with sophisticated data routing and robust synchronization
- Software component & silicon vendors need a reliable way to accelerate diverse codecs on diverse silicon

Media Infrastructure Stack
Hardware and software to deliver rich media processing solutions

Media Components
Inputs, video codecs, image and sound libraries, outputs etc.

Media Graphs
Media components connected to process media in real-time

Media Applications
Use media networks to deliver a compelling user experience

Media applications are not portable as proprietary hardware-centric libraries are needed to access media acceleration

Difficult to flexibly integrate diverse, multi-vendor media components into complete media processing solutions

Software components are not portable across processors - exacerbated by proliferation of media standards and increasing silicon complexity
Streaming Media in the Khronos Family

Applications or middleware libraries (JSR 184 engines, Flash players, media players etc.)

3D
OpenGL ES

Vector 2D
OpenVG

EGL
Graphics surface management for mixed mode 2D/3D/video rendering

Platform Media Frameworks

OpenMAX AL
Playback and recording interfaces

SOUND
Enhanced Audio

Accelerated media primitives for codec development

Component interfaces for codec integration

Image Libraries, Video Codecs, Sound Libraries

OpenMAX DL

Media Engines – CPUs, DSP, Hardware Accelerators etc.
OpenMAX – Streaming Media

- **Application Layer**
  - Media Application Portability
  - Applications programmed using cross-vendor interfaces

- **Integration Layer**
  - Media Graph Portability
  - Integrate media networks using standard interconnect protocols

- **Development Layer**
  - Media Component Portability
  - Develop portable media components using low-level media APIs

More media applications available on more platforms for more end-user value!

Portable and powerful media processing graphs can flexibly leverage available platform media components

Component vendors can ship more advanced functionality across more processors. Media silicon vendors reduce costs and time to market

OpenMAX layers can be implemented together or independently from the other layers to provide complete streaming media infrastructure portability
Development Layer
OpenMAX DL – Component Portability

- **OpenMAX DL is a library of key static primitive functions**
  - Designed to cover 80% of the processing required in a multimedia codec
- **Abstracts the ISA from the multimedia codec**
  - Enables faster codec development time and faster porting of existing codecs
- **Enables third party codec vendors to sell processor-agnostic codecs**
  - Multi-core architectures (i.e. ARM + DSP) gain greater code reuse between cores

- An increasing number of multimedia API codecs for video, audio, graphics and images
- Silicon vendors supply optimized OpenMAX DL library for rapid porting of codecs across multiple accelerators
- A wide range of media acceleration silicon using many diverse architectures
OpenMAX DL Domains

- **Video Domain**
  - MPEG-4 SP/H.263 BL (encode and decode)
  - H.264 (encode and decode)

- **Image Codec Domain**
  - JPEG (encode and decode)

- **Image Processing Domain**
  - Color space conversion
  - Pixel packing/unpacking
  - De-blocking / de-ringing
  - Rotation, scaling, compositing, etc.

- **Multimedia Audio Domain**
  - MP3
  - AAC

- **Signal Processing Domain**
  - FIR
  - IIR
  - FFT
  - Dot Product
Integration Layer
OpenMAX IL Media Acceleration

OpenMAX IL enables diverse high-level media frameworks and applications to portably tap into silicon media acceleration.
OpenMAX IL – Component Networks

- Enables arbitrary multimedia pipelines by plugging blocks together
  - Componentized architecture abstracts multimedia functionality block interfaces
- Wide variety of building blocks for imaging, video and audio functions
  - Encode, decode, apply an effect, capture, render, split, mix, etc
- Enables blocks from different sources to work together
  - Blocks can be implemented in software or hardware
OpenMAX IL “Component”

• A component is a building block encapsulating one function

• Plumbing
  - Each component port is the entry point or exit point for a stream of data
  - Ports are connected between components

• Knobs
  - A component parameter is a value that is set prior to component execution
  - A component config is a value that may be set during component execution
  - Examples: rate, volume, resolution, scaling, bit rate, error correction, brightness, etc

• Control – a standard interface common to all components to control:
  - The connection of ports and how data flows in and out of ports
  - Query/set configs and parameters
  - State management
OpenMAX IL Example Graph

- Standardized component interfaces enable flexible media graphs
- Includes multi-stream synchronization
- Allows for custom plug-ins

Example: MPEG-4 video synchronized with AAC audio decode
OpenMAX IL 1.1

• **Problem:** OpenMAX IL 1.0 only defines generic component semantics
  - How can system integrators create portable solutions if implementations don’t expose the same functionality

• **Solution:** OpenMAX IL 1.1 defines a set of standard components
  - To make most streaming applications easier to construct and more portable

• **Standardized interfaces and controls**
  - Readers/writers: 3gp, asf, image, video, audio
  - Audio decoders/encoders: AAC, AMR, MP3, WMA, Real Video
  - Audio post-processor: stereo widening, equalizer, reverb
  - Video decoders/encoders: MPEG4, H.264, etc
  - Image decoders/encoders: JPEG
  - Input devices: camera, audio input
  - Output devices: audio renderer, video renderer
  - Synchronization: clock component, video scheduler
OpenMAX AL

Application Layer
OpenMAX AL – Component Control

- **OpenMAX IL is powerful but complex**
  - More power than most application developers require

- **Most application developers just want to playback and record media**
  - Specifying where the content comes from
  - Specifying where the content should be rendered to
  - Manipulate a few playback controls
  - Have simple configurability

- **That’s what OpenMAX AL provides…**
  - A simple high-level multimedia API for playback and recording use cases
OpenMAX AL – Object Oriented Media

- **OpenMAX AL - simplified object-oriented streaming media**
  - Built to enable common use cases – but also extensible
  - Can be implemented over OpenMAX IL

- **OpenMAX AL Media Objects enable PLAY and RECORD of media**
  - Media Objects input and output to Devices
  - Perform some operation on an input and emit the result as output
  - Can handle audio, images, video with synchronized audio

- **Objects have control interfaces**
  - Play, Seek, Rate, Audio, Display Region, Metadata Extraction
  - Record, Camera, Video Encoder, Audio Encoder, Metadata Insertion, Radio, MIDI
Other OpenMAX AL Features

• Extensive camera controls
  - Flash modes, Metering modes
  - White balance controls, Focusing controls
  - Exposure compensation, ISO Sensitivity
  - Shutter speed & Aperture
  - Zoom (digital and optical)

• Analog radio controls
  - Tuning, RDS

• Audio routing
  - Application-selectable audio inputs and outputs, based on location, connectivity, etc.
  - I/O device capability querying

• Metadata extraction and insertion
  - Search/extract and insert/overwrite metadata in a variety of file formats
OpenMAX AL Profiles and Extensions

• Two profiles:
  - Media Player – media playback-only devices
  - Media Player/Recorder – full-featured media devices

• Optional features and extensions
  - Some features optional in all profiles – hardware or use case constraints
  - APIs are consistent when hardware is available
  - E.g. Vibra, LED, Analog Radio and MIDI are optional features of OpenMAX AL

• Vendor-specific extensions can be integrated into future API core specs
Embedded Audio
OpenSL ES – Object Oriented Audio

- OpenSL ES has an object-oriented programming model
  - Simplifies common use cases – but also extensible

- Engine Objects are central to any OpenSL ES session
  - Objects created using methods on the Engine Object interfaces

- OpenSL ES Objects enable PLAY and RECORD of audio
  - Perform some operation on an input and emit the result as output
  - Can handle almost any audio use case

- Objects have control interfaces
  - Play, Seek, Rate, Audio, Metadata Extraction
  - Record, Audio Encoder, MIDI
OpenSL ES
Designed with audio application developers in mind

• Cross-platform portability
  - No need to rewrite an application for every platform

• Audio Playback and Recording
  - Playback of everything from music files to MIDI ring-tones to UI sounds

• Full range of effects and controls – including advanced 3D effects such as Doppler and virtualization
  - Experience rich, enhanced sound from locations other than the handset, even moving, for the ultimate gaming experience

• Advanced MIDI
  - Use the output of the MIDI engine as a 3D sound source, making the ring tone appear coming from another direction than the music

• 3D Audio makes OpenSL ES the natural audio companion to OpenGL ES for gaming
  - Better gaming experience; 3D-audio for conferencing calls; more vivid music experience

• Designed for implementation by either a hardware or software solution
  - Unlike any other advanced audio API
Features Overview

• Playback of audio files
  - Playback PCM and encoded content
  - Good for sound effects; device UI sounds

• SP-MIDI, Mobile DLS, Mobile XMF
  - For interactive music and ring-tones

• Effects & Controls
  - Music and media player effects
  - Advanced environmental effects for gaming

• 3D Audio
  - Enhanced gaming experience
OpenSL ES Audio Playback Example

- Create Engine object
  - To drive this session

- Create Audio Output Mix object
  - Method on Engine interface
  - Mix object drives audio output devices

- Create Media Player object
  - Method on Engine interface
  - Input is URI pointing to a local mp3 file
  - Output drives audio output mix

- Register event callback
  - Method on Media Player interface

- Set PlayState to Playing
  - Method on Media Player interface

- Wait for end of file event
  - Via registered callback
Example: Playback of two audio files

URI
Memory
Buffer Queue

URI
Memory
Buffer Queue

Audio Player
SLPlayItf

Output Mix
SLVolumeltf

Audio Player
SLPlayItf
SL3DLocationItf
SLVolumeltf
OpenSL ES Profiles

Game-centric mobile devices
Advanced MIDI functionality, sophisticated audio capabilities such as 3D audio, audio effects, ability to handle buffers of audio, etc.

Music-centric mobile devices
High quality audio, ability to support multiple music audio codecs, audio streaming support

Basic mobile phones
Ring tone and alert tone playback (basic MIDI functionality), basic audio playback and record functionality, simple 2D audio games
Combine the Profiles – To Best Suit Any Product

- **Phone + Music** = Phone combined with portable music player
  - Advanced stereo ring tones
  - Digital music playback

- **Game + Music** = Advanced music player / Musical games
  - High quality music playback
  - 3D audio games with advanced audio effects

- **Phone + Game** = Gaming phone
  - 3D ring signals
  - Games utilizing advanced audio techniques

- **Phone + Game + Music** = Ultimate Audio Rendering Device
  - Multiple audio sources perceived
  - Encapsulates the entire functionality of OpenSL ES
OpenSL ES and OpenMAX AL

(Enhanced audio API)

(Enhanced audio API)

(Multimedia API)

3D Audio

Audio Effects

Audio Recording

Basic MIDI

Buffer queues

Audio Playback

Video playback

Video recording

Radio and RDS

Camera

Image capture & display

Working groups collaborate to define the common API functionality
OpenSL ES and OpenMAX AL

- **Independent**
  - No dependency between the APIs – either one can exist by itself
  - A device may support any combination of the APIs that most suits the device:
    - OpenMAX AL only (Media Player/Recorder profile)
    - OpenSL ES only (Phone, Game, Music profiles)
    - OpenMAX AL + OpenSL ES (Media Player/Recorder + Music)
    - OpenMAX AL + OpenSL ES (Media Player/Recorder + Game)
    - ....

- **Compatible**
  - Working groups collaborated to make sure the APIs work together well

- **Consistent**
  - Identical API architecture
  - Identical APIs for same functionality

- **Distinct**
  - OpenMAX AL represents basic multimedia functionality (audio, video and image)
  - OpenSL ES represents advanced audio-only functionality
OpenSL ES and OpenMAX IL – Example Use Case

URI DataSource → Audio Player → Output Mix

File Reader → Audio Decoder → Audio Effect → Audio Mixer → Audio Sink

Clock
Khronos Audio and Streaming Media

- Khronos provides a full multimedia framework solution through OpenMAX and OpenSL ES
  - Components and applications are easily ported between platforms
- The complete package provides innumerable market opportunities for silicon, OS vendors, IP vendors and application developers
  - With both hardware and software solutions possible
- Meets the consumer demand for rich multimedia
  - Everything from interactive games to streaming content
Thank You