Multimedia Stack Overview

Erik Noreke
Chair of OpenSL ES Working Group
Chair of OpenMAX AL Working Group
Peruser of OpenMAX IL
Today’s Consumer Requirements

- **Rich media applications and UI**
  - Consumer decisions are driven by the multimedia experience
  - Drives smart phone market

- **Enhanced user experience**
  - Games
  - Theater-like audio for video playback
  - Streaming media services
  - Mobile TV

- **Availability of third party applications**
  - Consumers expected to purchase $35 billion of mobile applications by 2014
Application Developer Requirements

- **Use case driven design**
  - For ease of use

- **Comprehensive feature set**
  - Including advanced audio and streaming media

- **Consistent APIs**
  - Same APIs for both hardware and software implementations

- **Full, predictable functionality**
  - No need to guess what's available or how it is going to work

- **Portable**
  - No need to port application for different platforms
System Integration Requirements

- **Function driven design**  
  - For ease of integration

- **Comprehensive feature set**  
  - Including advanced audio and streaming media

- **Consistent APIs**  
  - Same APIs regardless if using a software implementation or hardware acceleration

- **Full, predictable functionality**  
  - No need to guess about available functionality or how it is going to work

- **Ease of Integration**  
  - Expected set of components reduces integration time and costs
Multimedia Stack - Overview

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- HW Integration / Implementation Layer
- System Hardware

Use cases:
- Use case driven design

Ease of use:
- Creates abstraction layer for application developer

Abstraction level:
- Function driven design with access regardless of implementation

Ease of integration:
- Implementation of Multimedia functions in either software or hardware

Choice of HW / SW

HW Acceleration
Multimedia Stack - Domains

Applications

Application Access

Multimedia Control Layer

Multimedia System Access

HW Integration / Implementation Layer

System Hardware

Use cases

Ease of use

Abstraction level

Ease of integration

Choice of HW / SW

HW Acceleration

Application developers

Operating Systems

Device Manufacturers

Application developers

Operating Systems

Device Manufacturers
Multimedia Stack – Khronos APIs

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- HW Integration / Implementation Layer
- System Hardware

Use cases
Ease of use
Abstraction level
Ease of integration
Choice of HW/SW
HW Acceleration
Why Create Khronos Multimedia APIs?

Khronos APIs
• Conformance Tests
• Portable
• IP Protection
• Well defined layers of implementation
• Predictable set of functionality

Open Source Framework
• Free to change
• Every implementation unique
• GPL license
• Crosses implementation layers
• Functionality dependent on implementation
Streams – The Core of Multimedia

• Streams are content delivery channels
  - Video, Audio, Text data flow
  - Additional metadata

• Controlling multimedia is about manipulating the streams
  - Play rate, effects, source and destinations

• The streams are sent through various components
  - Readers, parsers and decoders (codecs)
  - Effects, schedulers and renderers
  - Splitters and mixers

• Combined with HW clock synch and system controls this forms the core of the multimedia stack
OpenMAX IL – Integration Layer

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- Integration / Implementation Layer
- System Hardware
- Use cases
- Ease of use
- Abstraction level
- Ease of integration
- Choice of HW/SW
- HW Acceleration
OpenMAX IL – Managing Streams

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

Portable & Re-usable streaming media building blocks
OpenMAX IL Component

- Component abstracts a single actor –
  - Decoder, effect or renderer
- Standardized ports allows chaining of arbitrary components
  - Including port tunneling for compatible components
- Commands are sent to the component
- Events are received from the component
- Ports have calls and callbacks
OpenMAX IL Example Graph

- Standardized component interfaces enable flexible media graphs
- Components are chained into pipelines as needed for each stream
- Includes multi-stream synchronization

Example: MPEG-4 video synchronized with AAC audio decode
OpenSL ES and OpenMAX AL – Application Access

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- Integration / Implementation Layer
- System Hardware

Use cases
Ease of use
Abstraction level
Ease of integration
Choice of HW/SW
HW Acceleration
OpenSL ES & OpenMAX AL – Controlling Streams

- Hardware and low level access 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
- OpenMAX AL and OpenSL ES provide...
  - Simple high-level multimedia APIs for playback and recording use cases
- ...With Cross-platform portability
  - No need to rewrite an application for every platform
OpenMAX AL
Focused on Streaming Media – *Not Media Streams*

- **Media playback and recording**
  - Playback and recording of everything from video files to radio to camera input

- **Full range of video effects and controls** – including playback rate, post processing, and image manipulation
  - Apply advanced effects to captured images and video before sending to friends

- **Complete metadata support**
  - Read, Edit, and Set the metadata for most types of content
OpenSL ES – Powerful 3D Audio

• 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 multimedia experience

• Full 3D audio functionality enhances any gaming experience
  - Perfect companion to OpenGL ES

• Designed to take full advantage of system capabilities
OpenSL ES and OpenMAX AL

- Working groups collaborated to define the common API functionality
- Independent, Compatible, Consistent and Distinct
The OpenSL ES / OpenMAX AL Object

- Object represents a logical grouping of functionality
  - Acts on a stream or set of streams
  - Player, recorder, mixer

- Object has a source and a destination
  - Acts on a stream or set of streams as they flow through

- Objects have control interfaces
  - Grouped methods and callbacks
  - Play, Record, Seek
OpenMAX AL & OpenSL ES – Play Audio
Shared Use Case
OpenMAX AL & OpenSL ES – Play Video

Shared Use Case
Multimedia Stack – Khronos APIs

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- HW Integration / Implementation Layer
- System Hardware

Use cases
Ease of use
Abstraction level
Ease of integration
Choice of HW/SW
HW Acceleration
Stream Control vs. Stream Management
Audio playback

OR

URI Data Source -> Player -> Output Mix

File Reader -> Audio Decoder -> Audio Effect -> Audio Mixer -> Audio Sink

Clock
Multimedia Stack – Khronos APIs

- Applications
- Application Access
- Multimedia Control Layer
- Multimedia System Access
- HW Integration / Implementation Layer
- System Hardware

- Use cases
- Ease of use
- Abstraction level
- Ease of integration
- Choice of HW/SW
- HW Acceleration
Example: 3D Playback of two audio files
Example: Video Camera
Complete Multimedia Use-Case

Much more flexibility than just “show a video”
Multimedia – Managing Streams

- **OpenMAX IL manages the pipeline**
  - Controls where and how the stream flows

- **OpenMAX AL and OpenSL ES manage the flow**
  - Controls when and how fast the stream flows

- **Together they separate the domains of the multimedia stack**
  - A distinct integration layer
  - Use case driven application access layer

- **To form the basis for a complete multimedia stack**
  - Without limiting implementation or innovation
Suggested Student Assignment: OpenMAX IL NULL Component

- Create a NULL OpenMAX IL Component
  - First with one input, one output port (1 stream)
  - Then with two input, two output ports (2 streams)

- Illustrates
  - Buffer handling and Latency
  - Stream synchronization and control
  - Command queue and event handling
  - Importance of industry standards conformance
  - Introduction to stream management

Commands

Events

Input ports
  • Callbacks

OMX IL NULL Component

Output ports
  • Calls
  • Callbacks
Thank You