A Standard for Tomorrow’s Smartphone User

• Smartphones are rapidly approaching 25% of all handsets sold worldwide
  - Steadily increasing market share
  - Expected to exceed 50% by 2012

• Entertainment is a major selling feature of smartphones
  - Games, Multimedia

• Success in the smartphone market is dependent on user experience

• Audio & Multimedia performance is quickly becoming an important factor
  - Market needs differentiating factor
  - Form factor limits screen growth
  - Important for operator services
Smartphone Market Requires Standards

- The market focus is shifting from proprietary to standardized and open functionality
  - The consumer is becoming aware and asking for standardization:
    - Functionality, GUI, connectors and application suites
- Standards are becoming more easily identified and desired by consumers:
  - A certain set of features from applications
  - The same basic suite of applications regardless of manufacturer
- The underlying OS is becoming important to end-user sales
  - Android, Symbian, Linux, Windows
Addressing the Smartphone Consumer

• Consumers are becoming more aware of what is driving the functionality in their phones
  - Important part of product marketing

• Manufacturers are shifting to standardized solutions
  - Universal chargers, data cables and headphones
  - Wi-Fi, Bluetooth, email, touch screens, web access

• Availability of third party applications
  - Consumers spent ~$6.2 billion on mobile-device applications in 2010, up 62% from 2009
  - Application developers require standardized APIs for portability and increased sales potential

• OpenSL ES addresses the need for standardized access to audio functionality and user experience, regardless of operating system
OpenSL ES Adoption – Standardizing Mobile Audio

OpenSL ES now supported in Android 2.3 NDK

Platform Providers

Driving forces behind enabling high-performance audio in mobile devices

IP Vendors

Handset Manufacturers

First adopter of OpenSL ES

SRS

Offers a full Phone-Music-Game OpenSL ES solution

© Copyright Khronos Group, 2011 - Page 5
Major shift in the mobile audio market 2009-12:

- Proprietary APIs are being abandoned in favor of standards-based APIs
- OpenSL ES is expected to ship in over 50% of smartphones by the end of 2012

*Based on ABI & Noreke Research, Q2'10
OpenSL | ES – Powerful Mobile Audio

• Create theater-quality audio experience
  - In a mobile device!

• Profiles reduce application customization
  - Applications can query available profiles
  - Develop to a specific profile or profile combination

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

• Designed for implementation by either a hardware or software solution
  - Unlike any other advanced audio API
OpenSL ES 1.1 consists of:

- **The specification**
  - 58-page user guide providing a comprehensive overview of the API
  - 526-page API reference that details the objects and interfaces

- **Header files**
  - Available to the public for download

- **Conformance Tests**
  - Ensures application portability across implementations
  - Adopters that pass the conformance tests are able to use the OpenSL ES logo

- **Adopters Package**
  - Defines the process and requirements for a conformant implementation
Designed with Audio Application Developers in Mind

- **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

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

- **3D Audio makes OpenSL ES the natural choice for any audio application**
  - Better gaming experience; 3D audio for conferencing calls; more vivid music experience
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
• **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**
  - For application
– Playback of two audio files
What’s new in OpenSL|ES 1.1

- Buffer queues
- Content pipes
- Better control of 3D performance
- Explicit object ordering
- Dynamic sources and sinks
- Metadata support for streaming playback
- Multiple version support
- Extension configuration support
- And more...
The **OpenSL ES™** 1.1 Advantage

- **Object-based audio for architectural simplicity**
  - Reduces development time

- **Native access for speed**
  - Necessary for high-performance audio applications

- **Advanced functionality for full application control**
  - Everything from simple playback to 3D audio

- **Profiles reduce fragmentation**
  - Phone, Music, Game

- **Advanced simplicity**
  - Reduces development time
  - Facilitates porting

```c
/* Handle game events */
gameEvent = GAMEGetEvents();
switch( gameEvent )
{
    case EVENT_GUNSHOT:
        /* Fire gun shot */
        res = (*gunshot)->GetInterface(gunshot, SL_IID_PLAY);
        (*playItf)->SetPlayState(playItf, SL_PLAYSTATE_PLAYING);
        break;

    case EVENT_DEATH:
        /* Player has been shot, scream! */
        res = (*scream)->GetInterface(scream, SL_IID_PLAY);
        (*playItf)->SetPlayState(playItf, SL_PLAYSTATE_PLAYING);
        dead = !dead;
        break;

    case EVENT FOOTSTEP:
        /* Play footsteps */
        res = (*footstep)->GetInterface(footstep, SL_IID_PLAY);
        (*playItf)->SetPlayState(playItf, SL_PLAYSTATE_PLAYING);
        break;
}
```
• **Same API regardless of underlying solution**
  - hardware accelerated
  - software implementations

• **Simplifies porting between platforms**
  - Spend application development time on application development, not porting

• **Meets the demand for high-performance audio**
  - Opens up the market for advanced audio applications
  - Provides mobile consumers with a home-theater-like immersive audio experience

• **OpenSL ES does for audio what OpenGL ES does for graphics**
  - Brings life to your applications and devices!
Thank You