Advanced Android Audio Techniques

Dave Sparks
20-May-2010

Google™ 10 I|O

View live notes and ask questions about
this session on Google Wave:

http://bit.ly/aDtLRp

Agenda

- Native audio signal processing
- Tips on power and resource usage
- What's new in Froyo?
- Roadmap
- Q&A

Native Audio Signal Processing

AudioTrack Overview

- Raw PCM audio API
- Streaming or static buffers
- Can set callbacks to refill buffer
- Retrieve play position
- Useful for games or streaming audio

AudioTrack Sample Code

```
package com.example.audiotracksample;

public class AudioTrackSample implements Runnable {

   static private final int mBufferSize = 8000;
   private AudioTrack mTrack;
   private short mBuffer[];
   private short mSample;

   class AudioTrackSample {
      mBuffer = new short[mBufferSize];
      mTrack = new AudioTrack(STREAM_MUSIC, 44100, CHANNEL_OUT_MONO,
         ENCODING_PCM_16BIT, mBufferSize * 2, MODE_STREAM);
      mSample = 0;
   }

   public void run() {
      mTrack.play();
      while(1) {
         // fill the buffer
         generateTone(mBuffer, mBufferSize);
         mTrack.write(mBuffer, 0, mBufferSize);
      }
   }

   ...
```

```
// continuation of class AudioTrackSample

   ...

   public void generateTone(short [] data, int size) {
      for (int i = 0; i < size; i++) {
         pData[i] = mSample;
         mSample += 600; // ~400 Hz sawtooth
      }
   }
}
```

Using Native Code

- Requires the Android NDK
- Build a Java application
- Create a native library
- Add “native” methods to Java class
- Load the native library
- Call native methods from Java

AudioTrack Native Code

```
package com.example.jnisample;

public class JNISample implements Runnable {

  static private final int mBufferSize = 8000;
  private AudioTrack mTrack;
  private short mBuffer[] = new short[mBufferSize];
  private int mSample;

  class JNISample {
      mBuffer = new short[bufferSize];
      mTrack = new AudioTrack(STREAM_MUSIC, 44100, CHANNEL_OUT_MONO,
        ENCODING_PCM_16BIT, mBufferSize * 2, MODE_STREAM);
      mSample = 0;
  }

  public void run() {
    mTrack.play();
    while(1) {
      // fill the buffer
      generateTone(mBuffer, mBufferSize);
      mTrack.write(mBuffer, 0, mBufferSize);
    }
  }

   ...
```

```
// continuation of class JNISample
   ...

   static {
      System.loadLibrary("generate_tone");
   }

   public native int generateTone(short [] data, int size);
}
```

```
jint Java_com_example_jnisample_JNISample_generateTone(
    JNIEnv *env, jobject thiz, jshortArray data, jint size)
{
    if (size <= 0) {
        return ERR_NO_DATA;
    }

    // convert java short array to C pointer, pinning the array in memory
    pData = (short*) env->GetPrimitiveArrayCritical(data, NULL);
    if (!pData) {
        return ERR_NO_DATA;
    }

    // fill buffer with 16-bit PCM audio
    short sample = 0;
    for (int i = 0; i < size; i++) {
        pData[i] = sample;
        sample += 600; // ~400 Hz tone
    }

    // unpin the array
    env->ReleasePrimitiveArrayCritical(data, pData, 0);

    return SUCCESS;
}
```

Google™ 10 I/O

Accessing Objects from Native Code

Google™ 10 I/O

```
static jobject mSampleField = 0;

jint Java_com_example_jnisample_JNISample_generateTone(
JNIEnv *env, jobject thiz, jshortArray data, jint size)
{
   if (mSampleField == 0) {
      jclass clazz = env->FindClass("com.example.jnisample.JNISample");
      if (clazz != 0) {
         mSampleField = env->getFieldID(clazz, "mSample", "S");
      }
      if (mSampleField == 0) {
         return FATAL_ERROR; // fatal error;
      }
   }
   // ... code to pin array and return pointer goes here

   short sample = env->GetShortField(thiz, mSampleField);
   for (int i = 0; i < size; i++) {
      pData[i] = sample;
      sample += 600; // ~400 Hz tone
   }
   env->SetShortField(thiz, mSampleField, sample);

   // ... code to unpin array goes here
   return SUCCESS;
}
```

Tips on Power and Resource Usage

- Make sure you call release() on media objects
- Powers down unneeded hardware
- Allows other apps to use h/w resources
- Call release() from onPause(), not just onDestroy()

What’s new in Froyo?

- Audio focus and transport API’s
- Soundpool improvements
- Audio routing improvements for earpiece and BT SCO
- Camera improvements

Audio Focus/Transport API's

Audio Focus and Transport API's

- Need for applications and system services to cooperate
- Application developers remain in control
- Apps and services can register to receive events
- Apps and services can request audio focus
- Focus request can be transient or permanent
- Requests may also include a "ducking" hint
- When done, an app abandons audio focus
- Transport controls work in the same way
- Helper classes will be available for backwards compatibility

Audio Focus Sample Code

```
public void onCreate() {
   super.onCreate();

   mAudioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
   setVolumeControlStream(AudioManager.STREAM_MUSIC);
   ...
}

public void play() {
   mAudioManager.requestAudioFocus(mListener, AudioManager.STREAM_MUSIC,
      AudioManager.AUDIOFOCUS_GAIN);
   start();
   mPlaying = true;
}

// called when stream stops for any reason
private void onStopped() {
   mPlaying = false;
   if (!mRestart) {
      mAudioManager.abandonAudioFocus(mListener);
   }
}

public void onDestroy() {
   mAudioManager.abandonAudioFocus(mListener);
   ...
}
```

```
private onAudioFocusChangeListener mListener =
   new OnAudioFocusChangeListener() {

   public void onAudioFocusChange(int focusChange) {

      switch (focusChange) {
         case AudioManager.AUDIOFOCUS_LOSS:
         mRestart = false;
         if (mPlaying) {
               pause();
            }
            break;
         case AudioManager.AUDIOFOCUS_TRANSIENT:
         case AudioManager.AUDIOFOCUS_TRANSIENT_CAN_DUCK:
            if (mPlaying) {
               mRestart = true;
               pause();
            }
            break;
         case AudioManager.AUDIOFOCUS_GAIN:
            if (!mPlaying && mRestart) {
               mRestart = false;
               resume();
            }
            break;
      }
   }
}
```

```
public void onAudioFocusChange(int focusChange) {
   switch (focusChange) {

      // handle permanent and transient loss as before
         ...

      // special case for ducking
      case AudioManager.AUDIOFOCUS_TRANSIENT_CAN_DUCK:
         if (mPlaying) {
            mDucking = true;
            mOldVolume = mVolume;
            setVolume(mVolume * 0.125);
         }
         break;
      case AudioManager.AUDIOFOCUS_GAIN:
      if (!mPlaying && mRestart) {
            mRestart = false;
            resume();
         } else if (mDucking) {
            mDucking = false;
            setVolume(mOldVolume);
         }
         break;
      }
   }
}
```

SoundPool Improvements

- New callback when sound is loaded
- autoPause() - pauses all active tracks
- autoResume() - resumes all previously active tracks

Audio Routing Improvements

- STREAM_VOICE_CALL routed to earpiece by default
- Call AudioManager.setSpeakerOn() to route to speaker
- Call AudioManager.startBluetoothSco() to route to BT SCO
- Use for scenarios where user wants privacy (e.g. voice mail)

Camera Improvements

- New preview API avoids GC's
- Compress YUV to JPEG
- "No thumbnail" mode for JPEG encoder
- FOV and focal length parameters
- Exposure control settings
- Portrait mode support

Camera Preview Code

```
// camera preview snippet

private Camera mCamera;
private Object[] mBuffers;

public void startPreview() {
   // initialize camera, etc.
   ...

   // initialize callback function and allocate buffers
   Size size = mCamera.getParameters().getPreviewSize();
   mBuffer[0] = new byte[size.width * size.height * 3 / 2 + 1];
   mBuffer[1] = new byte[size.width * size.height * 3 / 2 + 1];
   mCamera.addCallbackBuffer(mBuffer[0]);
   mCamera.addCallbackBuffer(mBuffer[1]);
   mCamera.setPreviewCallbackWithBuffer(new PreviewCallback());
   mCamera.startPreview();
   ...
}

// callback
private final class PreviewCallback() implements PreviewCallback {
   public void onPreviewFrame(byte[] data, Camera camera) {
      processData(data);
      mCamera.addCallbackBuffer(data);
   }
}
```

Roadmap

Roadmap

- OpenSL ES native API
- OpenAL support
- Audio effects processing framework
- Expose low-level media API's
- WebM stream support (VP8 + Vorbis)
- FLAC decoder
- AAC-LC encoder
- AMR-WB encoder

Q & A

View live notes and ask questions about
this session on Google Wave:

http://bit.ly/aDtLRp