CS378 - Mobile Computing

Audio
Android Audio

• Use the MediaPlayer class

• Common Audio Formats supported:
  — MP3, MIDI (.mid and others), Vorbis (.ogg), WAVE (.wav) and others

• Sources of audio
  — local resources (part of app)
  — internal URIs (Content Provider for other audio available)
  — External URLs (streaming)
MediaPlayer

- Playback control of MediaPlayer managed as a state machine
  - Idle
  - Initialized
  - Preparing
  - Prepared
  - Started
  - Paused

- Playback Complete
- Stopped
- End
- Invalid state transitions result in errors
MediaPlayer State Diagram

- Single arrows are synchronous transitions
- Double arrows are asynchronous transitions
Simple Sound Demo App

• audio files local to app placed in res/raw

• CAUTION
  – large sound files difficult to install on emulator:
  – [http://tinyurl.com/3pwljlfj](http://tinyurl.com/3pwljlfj)
  – better success with dev phones / actual devices
Playing Local Audio

• To play audio local to the app
• use the MediaPlayer.create convenience method
  — when complete MediaPlayer in the `prepared` state
• start MediaPlayer
• approach:
  — build listeners for each button to call the playSound method with appropriate song id when clicked
Simple Approach

```java
private void buildListners() {
    int[] ids = {R.id.gong, R.id.ava, R.id.fax, 
                R.id.folk, R.id.rise, R.id.rain};
    int[] songs = {R.raw.gong, R.raw.ava_maria, 
                   R.raw.fax, R.raw.music, 
                   R.raw.rise, R.raw.rain};

    for(int i = 0; i < ids.length; i++) {
        final Button button = (Button) findViewById(ids[i]);
        final int SONG_ID = songs[i];
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                playSound(SONG_ID);
            }
        });
    }
}
```
playSound method

```java
private void playSound(int songID) {
    MediaPlayer mediaPlayer = MediaPlayer.create(this, songID);
    mediaPlayer.start();
    // no need to call prepare(); create() does that for you
}
```

- okay for *short* sounds
- downsides:
  - plays to completion
  - multiple sounds play at same time (desirable in some cases)
  - audio continues to play when app paused
Changing Behavior

• Add instance variable for MediaPlayer
• If playing stop and release before creating new Player

```java
private void playSound(int songID) {
  if(player == null || !player.isPlaying()) {
    Log.d(TAG, "player null or not playing " +
           "- creating new player");
    player = MediaPlayer.create(this, songID);
  }
  if(player.isPlaying()) {
    Log.d(TAG, "player playing - " +
           "stopping and releasing");
    player.stop();
    player.release();
    player = MediaPlayer.create(this, songID);
  }
  player.start();
}
```
Cleaning Up

• Current version does not end well
• Audio continues to play if back button pressed and even if home button pressed!
• Activity Life Cycle
• on pause we should stop MediaPlayer and release
stopPlayer method

- Connect app stop button to stopPlayer
  - could use XML onClick and add View parameter or set up listener ourselves

```java
// set up the stop button
Button stop = (Button) findViewById(R.id.stop);
stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        stopPlayer(v);
    }
});

private void stopPlayer() {
    if (player != null) {
        player.stop();
        player.release();
        player = null;
    }
}
```
onPause

• `onPause()` should call the `stopPlayer` method
• what happens if activity resumed?

```java
@Override
protected void onPause() {
    super.onPause();
    // stop the music!!
    stopPlayer();
}
```
Saving State

- Resume music where we left off if paused or activity destroyed due to orientation change

```java
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    stopPlayer();
}

@Override
protected void onPause() {
    super.onPause();
    stopPlayer();
}
```
Saving MediaPlayer State

- Not a lot of data so used the SharedPreferences

```java
private void stopPlayer() {
    if(player != null) {
        if(player.isPlaying()) {
            SharedPreferences mPrefs
                = getSharedPreferences("sound_demo", MODE_PRIVATE);
            SharedPreferences.Editor ed = mPrefs.edit();
            ed.putInt("songID", currentSongID);
            ed.putInt("audioLocation", player.getCurrentPosition());
            ed.commit();
        }
    player.stop();
    player.release();
    player = null;
}
```
Restarting Audio

• In onCreate check if audio was interrupted recreate player with same id and move to correct position

• Can write data to shared preferences or bundle (onSaveInstanceState) and pull out in onCreate

• Possible fix for orientation changes
  — in app manifest file under activity field
    android:configChanges="orientation"
    - But now we are responsible for orientation changes
Playing Audio from Phone

- If audio is on device / system, but not local to app use a URI
- Obtain URIs of Music via a Content resolver
- Example of simply listing URIs to the logcat
private void showContent() {
    ContentResolver contentResolver = getContentResolver();
    Cursor cursor = contentResolver.query(uri, null, null, null, null);
    if (cursor == null) {
        Log.d(TAG, "cursor == null, query failed");
    } else if (!cursor.moveToFirst()) {
        Log.d(TAG, "no media on the device");
    } else {
        int titleColumn
            = cursor.getColumnIndex(android.provider.MediaStore.Audio.Media.TITLE);
        int idColumn
            = cursor.getColumnIndex(android.provider.MediaStore.Audio.Media._ID);
        do {
            long thisId = cursor.getLong(idColumn);
            String thisTitle = cursor.getString(titleColumn);
            Log.d(TAG, "Found media: thisID: " + thisId + ", thisTitle: " + thisTitle);
        } while (cursor.moveToNext());
    }
}
MediaPlayer and System Audio

- After URI retrieved can play audio with MediaPlayer
- this approach requires calling prepare yourself
  - no convenience method
Playing Audio Via Local URI

- id obtained via approach from showContent method

```java
private void playRandomSong() {
    stopPlayer();

    // get id of random song
    long id = showContent();

    Uri contentUri = ContentUris.withAppendedId(
        android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, id);

    player = new MediaPlayer();
    player.setAudioStreamType(AudioManager.STREAM_MUSIC);

    try {
        player.setDataSource(this, contentUri);
        player.prepare();
        player.start();
    } catch (Exception e) {
        // handle exception
    }
}
```
Other Audio

- Other audio for ringtones, notifications, and alarms can be accessed via a RingtoneManager
- Obtain URIs and play with media player
- from DDMS:

![Filesystem with audio subdirectories](image)
private void showRingtones() {
    RingtoneManager rm = new RingtoneManager(this);
    rm.setType(RingtoneManager.TYPE_ALL);
    Cursor cursor = rm.getCursor();
    if (cursor == null) {
        Log.d(TAG, "cursor == null, query failed");
    } else if (!cursor.moveToFirst()) {
        Log.d(TAG, "no ringtones on the device");
    } else {
        int count = cursor.getCount();
        Log.d(TAG, "count of ringtones: \" + count);
        for(int i = 0; i < count; i++) {
            Ringtone r = rm.getRingtone(i);
            Log.d(TAG, "ringtone num: \" + i + " name: \" + r.getTitle(this));
        }
    }
}
Playing Other Audio

• Once the URI is obtained, playing other audio is same as playing song

```java
int count = cursor.getCount();
Log.d(TAG, "count of ringtones: " + count);
for(int i = 0; i < count; i++) {
    Ringtone r = rm.getRingtone(i);
    Log.d(TAG, "ringtone num: " + i
         + " name: " + r.getTitle(this));
}
```
Playing Audio from Remote URL

- Straightforward given the URL

```java
private void playFromURL() {
    String url = "http://www.pacdv.com/sounds/" + "machine_sound_effects/chain-saw-2.mp3";
    stopPlayer();
    if(player == null)
        player = new MediaPlayer();
    player.setAudioStreamType(AudioManager.STREAM_MUSIC);
    try {
        player.setDataSource(url);
        player.prepare(); // might take long! (for buffering, etc)
        player.start();
    } catch (IOException e){
        // error handling...
    }
}
Completion of Audio

• If action required when audio done playing implement the MediaPlayer.onCompletionListener interface

```java
MediaPlayer.OnCompletionListener done
    = new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            // take necessary action on completion
        }
    };
```

• could make activity the listener
Looping

• to loop sound (play over and over) simply set the isLooping method of the MediaPlayer to true
SoundPool

• Another Android class

```java
public SoundPool (int maxStreams, int streamType, int srcQuality)
```

Constructor. Constructs a SoundPool object with the following characteristics:

**Parameters**

- `maxStreams` the maximum number of simultaneous streams for this SoundPool object
- `streamType` the audio stream type as described in AudioManager. For example, game applications will normally use `STREAM_MUSIC`.
- `srcQuality` the sample-rate converter quality. Currently has no effect. Use 0 for the default.
Using SoundPool

• Great for applications with a number of short sound samples

• `maxStreams` parameter sets maximum number of sounds that can be played at once via this SoundPool

• If `max` is exceeded stream with lowest priority stopped
  — and then by age (oldest) with lowest priority
SoundPool play

```java
public final int play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soundID</td>
<td>a soundID returned by the load() function</td>
</tr>
<tr>
<td>leftVolume</td>
<td>left volume value (range = 0.0 to 1.0)</td>
</tr>
<tr>
<td>rightVolume</td>
<td>right volume value (range = 0.0 to 1.0)</td>
</tr>
<tr>
<td>priority</td>
<td>stream priority (0 = lowest priority)</td>
</tr>
<tr>
<td>loop</td>
<td>loop mode (0 = no loop, -1 = loop forever)</td>
</tr>
<tr>
<td>rate</td>
<td>playback rate (1.0 = normal playback, range 0.5 to 2.0)</td>
</tr>
</tbody>
</table>
Using SoundPool

• Looping of sounds:
  – 0 no looping
  – -1 loop forever
  – >0, play that many times

• frequency (speed) can be changed
  – range from 0.5 to 2.0
  – 0.5 twice as long to play
  – 2.0 half as long to play