Multimedia

Programming the Android Platform

Multimedia

- Android provides built-in encoding/decoding for a variety of common media types
- Allows you to play & record audio, still images & video

Some Multimedia Classes

- AudioManager
- SoundPool
- RingtoneManager & Ringtone
- MediaPlayer
- MediaRecorder
- Camera

AudioManager

- Manages volume and ringer mode control
- Loads & plays system sound effects
  - e.g., key clicking
- Acquire AudioManager instance via
  - Context.getSystemService(Context.AUDIO_SERVICE)

SoundPool

- Represents a collection of audio samples (streams)
- Can mix and play multiple simultaneously

AudioManager/SoundPool

```
public class AudioVideoAudioManagerActivity extends Activity {
  private float volume = 0;
  private SoundPool soundPool;
  private int soundId;
  public void onCreate(Bundle savedInstanceState) {
    ...
    final AudioManager audioManager = (AudioManager)
                        getSystemService(AUDIO_SERVICE);
    audioManager.loadSoundEffects();
    final TextView tv ...
    volume = audioManager
            .getStreamVolume(AudioManager.STREAM_MUSIC);
    tv.setText(String.valueOf(volume))
    ...
```

AudioManager/SoundPool (cont.)

```
…
final Button upButton = …
 upButton.setOnClickListener(new OnClickListener() {
   public void onClick(View v) {
   …
   audioManager.playSoundEffect(AudioManager.FX_KEY_CLICK);
 }
});
final Button playButton =  …
  playButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
      …
      soundPool.play(soundId, volume, volume, 1, 0, 1.0f);
    }
});
```

AudioManager/SoundPool (cont.)

```
 soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
 soundPool.setOnLoadCompleteListener( new OnLoadCompleteListener() {
  public void onLoadComplete(SoundPool soundPool,int sampleId,int status) {
     playButton.setEnabled(true);
  }
});
 soundId = soundPool.load(this, R.raw.sound, 1);
}
  protected void onPause() {
    if (null != soundPool) {
      soundPool.unload(soundId);
      soundPool.release();
      soundPool = null;
    }
    super.onPause();
  }
```

RingtoneManager

- RingtoneManager provides access to audio clips used for ringtones, notifications, alarms, etc.
- Manages multiple providers for audio clips
  - getCursor () returns a Cursor for accessing available ringtones

RingtoneManager (cont.)

```
public class AudioVideoRingtoneManagerActivity extends Activity {
 public void onCreate(Bundle savedInstanceState) {
  …
 final Button ringtoneButton = …
 ringtoneButton.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
   Ringtone r = RingtoneManager.getRingtone
                (AudioVideoRingtoneManagerActivity.this,
                Settings.System.DEFAULT_RINGTONE_URI);
   if (null != r) r.play();
  }
 });
 …
```

RingtoneManager (cont.)

- Similar code for notification & alarm ringtones
  - Settings.System.DEFAULT_NOTIFICATION_URI
  - Settings.System.DEFAULT_ALARM_ALERT_URI

MediaPlayer

- Controls playback of audio/video files & streams
- Allows greater control over stream playback
  - start(), stop, pause(), seekTo()

MediaPlayer

- Operation based on a state machine
  - See documentation
- Some key steps
  - setDataSource()
  - prepare()
  - start()
  - pause(), seekTo()
  - stop()
  - release()

VideoView

- View for displaying video files
- Can load video from multiple sources
- Provides various display options & convenience functions

VideoView Example

```
public class AudioVideoVideoPlayActivity extends Activity {
…
 public void onCreate(Bundle savedInstanceState) {
  …
  videoButton.setOnClickListener(new OnClickListener() {
   public void onClick(View v) {
    videoView.setMediaController(
        new MediaController(AudioVideoVideoPlayActivity.this));
    videoView.setVideoURI(Uri.parse(/* video URI */));
    videoView.start();
   }
  });
 …
```

VideoView (cont.)

```
  protected void onPause() {
    if (videoView != null && videoView.isPlaying()) {
      videoView.stopPlayback();
      videoView = null;
    }
    super.onPause();
  }
...
```

MediaRecorder

- Used to record audio and video
- Operation based on a state machine
  - See documentation
- Some key steps
  - setAudioSource()/setVideoSource()
  - setOutputFormat(), …
  - prepare(), start()
  - stop(), release()

MediaRecorder (cont.)

```
public class AudioRecordingActivity extends Activity {
  …
  private MediaRecorder mRecorder = null;
  …
  private void startRecording() {
     mRecorder = new MediaRecorder();
    mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mRecorder.setOutputFormat (
                                      MediaRecorder.OutputFormat.THREE_GPP);
    mRecorder.setOutputFile(mFileName);
    mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    try { mRecorder.prepare(); } catch (IOException e) {}
    mRecorder.start();
}
…
```

MediaRecorder (cont.)

```
  …
private void stopRecording() {
  mRecorder.stop();
  mRecorder.release();
  mRecorder = null;
}
```

Camera

- Used to
  - Manage image capture settings
  - Start/stop preview
  - Take pictures
  - etc.
- Client for the Camera service, which manages the actual camera hardware

Camera Permissions

- <uses-permission android:name="*android.permission.CAMERA*" />
- <uses-feature android:name="*android.hardware.camera*" />
- <uses-feature android:name=
  "*android.hardware.camera.autofocus*" />

Camera Usage

- Get Camera instance
- Set Camera parameters as necessary
- Setup preview display
- Start the preview
- Take a picture & process image data
- Release the Camera when not in use

Camera (cont.)

```
public class AudioVideoCameraActivity extends Activity {
  …
  public void onCreate(Bundle savedInstanceState) {
    …
    getWindow().setFormat(PixelFormat.TRANSLUCENT);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags  (
                        WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);
    setContentView(R.layout.main);
    SurfaceView mSurfaceView = …
    SurfaceHolder mSurfaceHolder = mSurfaceView.getHolder();
    mSurfaceHolder.addCallback(mSurfaceHelper);
    mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
```

Camera (cont.)

```
protected void onPause() {
  super.onPause();
  mCamera.release();
  mCamera = null;
}
protected void onResume() {
  super.onResume();
  mCamera = Camera.open();
}
```

Camera (cont.)

```
public void surfaceCreated(SurfaceHolder holder) {
  mFrame = (LinearLayout) findViewById(R.id.frame);
  mFrame.setOnTouchListener(mTouchHelper);
}

public void surfaceChanged(SurfaceHolder holder, int format,
                                                  int width, int height) {
  …
  Camera.Parameters p = mCamera.getParameters();
  p.setPreviewSize(width, height);
  mCamera.setParameters(p);
  mCamera.setPreviewDisplay(holder);  // try-catch block elided
  mCamera.startPreview();
  mPreviewRunning = true;
}
```

Camera (cont.)

```
public void surfaceDestroyed(SurfaceHolder holder) {
  mPreviewRunning = false;
  if (null != mCamera) {
    mCamera.stopPreview();
    mCamera.release();
  }
 }
...
```

Camera (cont.)

```
Camera.ShutterCallback mShutterCallback = new
   Camera.ShutterCallback() {
  public void onShutter() {
   // do something
  }
};

Camera.PictureCallback mPictureCallback = new
   Camera.PictureCallback() {
  public void onPictureTaken(byte[] data, Camera camera) {
   // do something
  }
};
```

Camera (cont.)

```
  View.OnTouchListener mTouchHelper =
                                                  new View.OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
      mCamera.takePicture(mShutterCallback, null, mPictureCallback);
      return true;
    }
  };
}
```

Source Code Examples

- AudioVideoAudioManager
- AudioVideoAudioRecord&Play
- AudioVideoCamera
- AudioVideoRingtoneManager
- AudioVideoVideoPlay