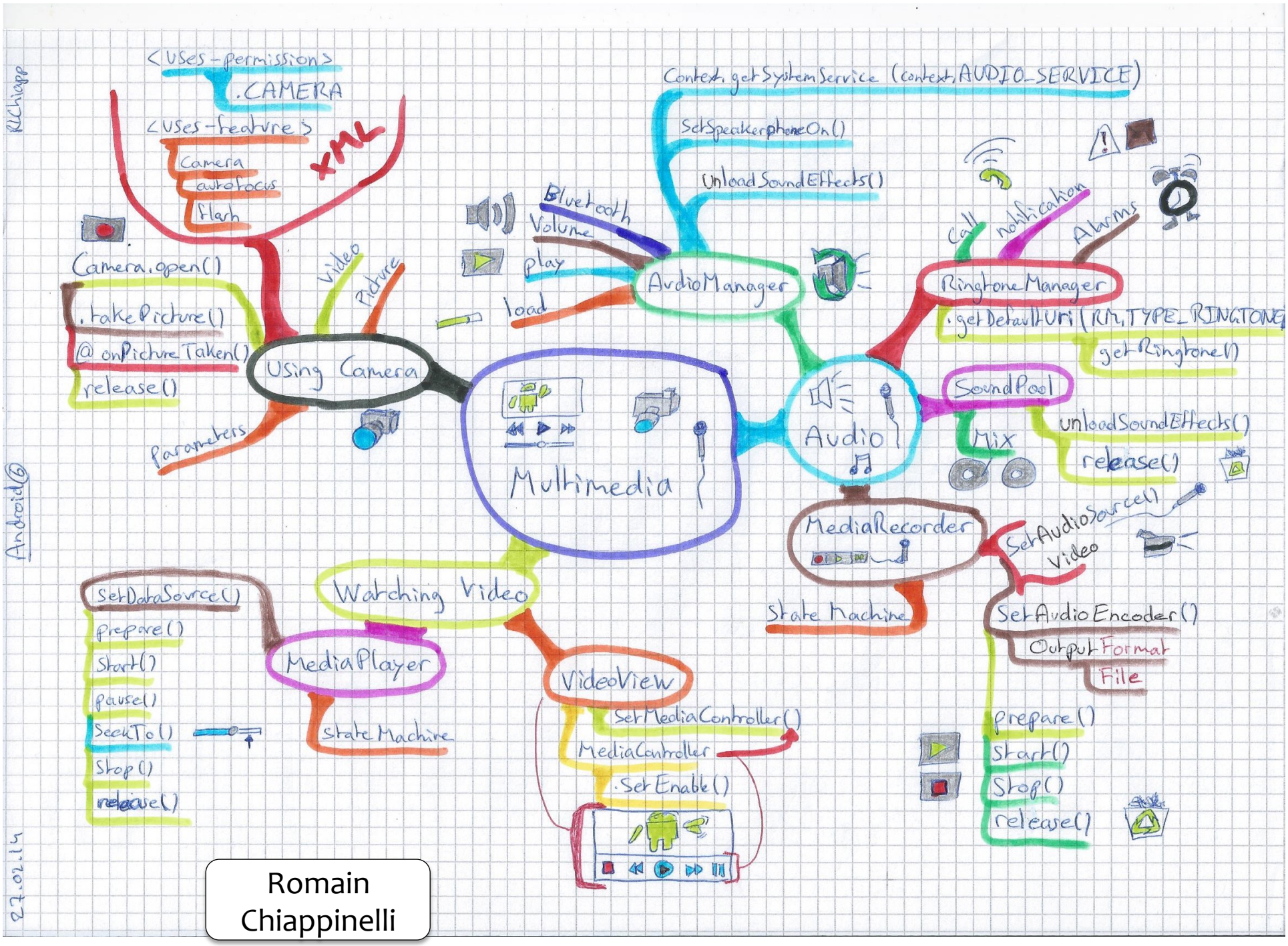

RLChiapp
Android©
27.02.14
<Uses-permission>
.CAMERA
<Uses-feature>
Camera
autofocus
Flash
XML
Camera.open()
.takePicture()
@onPictureTaken()
release()
Parameters
Video
Picture
Using Camera
Bluetooth
Volume
play
load
AudioManager
Context.getSystemService(Context.AUDIO_SERVICE)
SetSpeakerphoneOn()
UnloadSoundEffects()
Multimedia
Audio
Call
notification
Alarms
RingtoneManager
.getDefaultUri(RM.TYPE_RINGTONE)
getRingtone()
SoundPool
unloadSoundEffects()
release()
Mix
MediaRecorder
SetAudioSource()
Video
State Machine
SetAudioEncoder()
OutputFormat
File
prepare()
start()
stop()
release()
Watching Video
SetDataSource()
prepare()
Start()
pause()
SeekTo()
Stop()
release()
MediaPlayer
State Machine
VideoView
SetMediaController()
MediaController
.SetEnable()
Romain
Chiappinelli

Multimedia

Today's Topics

Multimedia Support Classes

Playing Audio

Watching Video

Recording Audio

Using the Camera

Multimedia

Android provides support for encoding and decoding a variety of common media formats

Allows you to play & record audio, still images & video

Some Multimedia Classes

AudioManager & SoundPool 

RingtoneManager & Ringtone

MediaPlayer 

MediaRecorder

Camera

AudioManager

Manages volume, system sound effects, and ringer mode control

Acquire AudioManager instance via

```
Context.
    getSystemService(Context.AUDIO_SERVICE)
```

AudioManager

Load & play sound effects

Manage volume

Manage peripherals

SoundPool

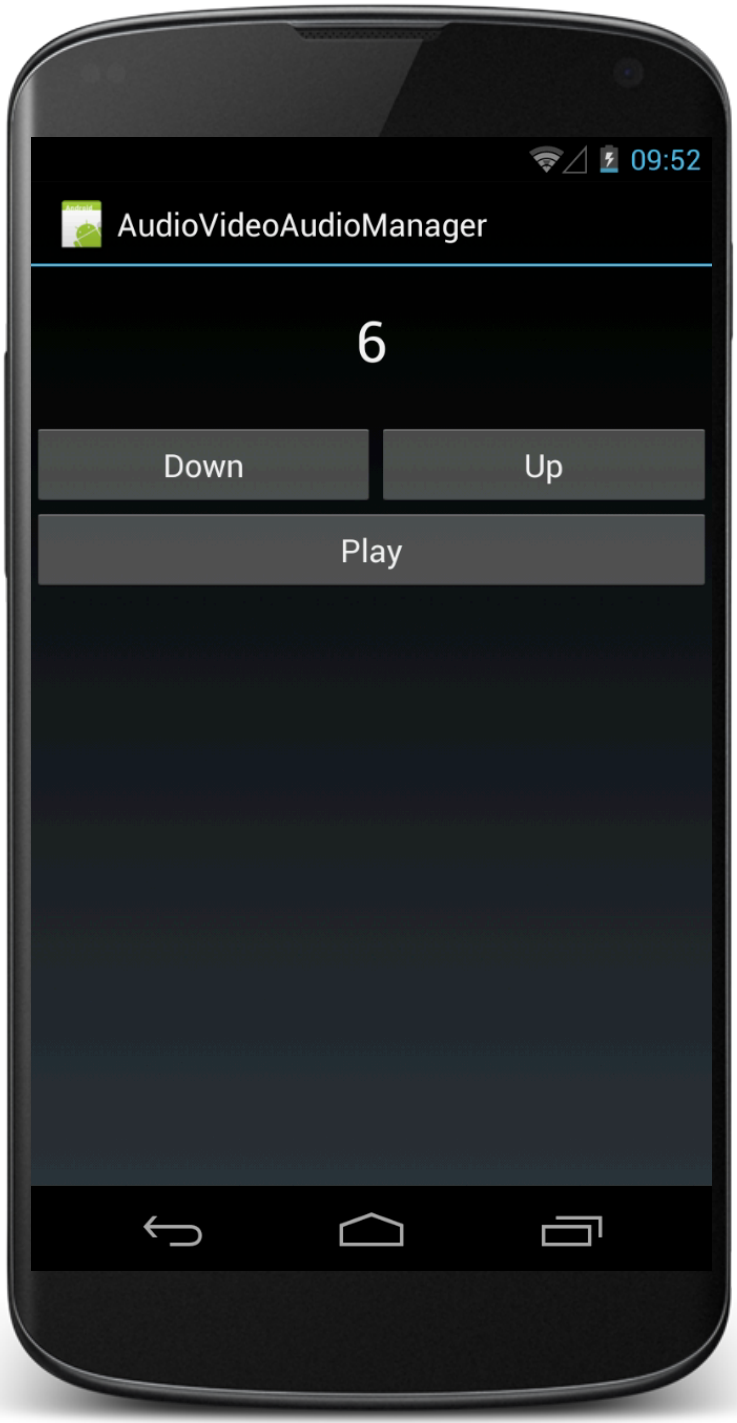
Represents a collection of audio samples (streams)

Can mix and play multiple simultaneously

AudioVideoAudioManager

Presents two buttons that adjust the volume up or down

Presents a play button that, when pressed, plays a bubble popping sound at the current volume level


09:52
AudioVideoAudioManager
6
Down
Up
Play

AudioVideoAudioManager

```
// Get reference to the AudioManager
mAudioManager = (AudioManager) getSystemService(AUDIO_SERVICE);

// Display current volume level in TextView
final TextView tv = (TextView) findViewById(R.id.textView1);
tv.setText(String.valueOf(mVolume));

// Set up Button to increase the volume
final Button upButton = (Button) findViewById(R.id.button2);
upButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        // Play key click sound
        mAudioManager.playSoundEffect(AudioManager.FX_KEY_CLICK);

        if (mVolume < mVolumeMax) {
            mVolume += 2;
            tv.setText(String.valueOf(mVolume));
        }
    }
});
```

AudioVideoAudioManager

```
// Create a SoundPool
mSoundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);

// Load bubble popping sound into the SoundPool
mSoundId = mSoundPool.load(this, R.raw.slow_whoop_bubble_pop, 1);

// Set an OnLoadCompleteListener on the SoundPool
mSoundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {

    @Override
    public void onLoadComplete(SoundPool soundPool, int sampleId,
            int status) {

        // If sound loading was successful enable the play Button
        if (0 == status) {
            playButton.setEnabled(true);
        } else {
            Log.i(TAG, "Unable to load sound");
            finish();
        }
    }
});
```

AudioVideoAudioManager

```
// Play the sound using a SoundPool
playButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mCanPlayAudio)
            mSoundPool.play(mSoundId, (float) mVolume / mVolumeMax,
                    (float) mVolume / mVolumeMax, 1, 0, 1.0f);
    }

});
```

AudioVideoAudioManager

```java
// Get ready to play sound effects
@Override
protected void onResume() {
    super.onResume();

    mAudioManager.setSpeakerphoneOn(true);
    mAudioManager.loadSoundEffects();

}

// Release resources & clean up
@Override
protected void onPause() {

    if (null != mSoundPool) {
        mSoundPool.unload(mSoundId);
        mSoundPool.release();
        mSoundPool = null;
    }

    mAudioManager.setSpeakerphoneOn(false);
    mAudioManager.unloadSoundEffects();

    super.onPause();
}
```

Ringtone and RingtoneManager

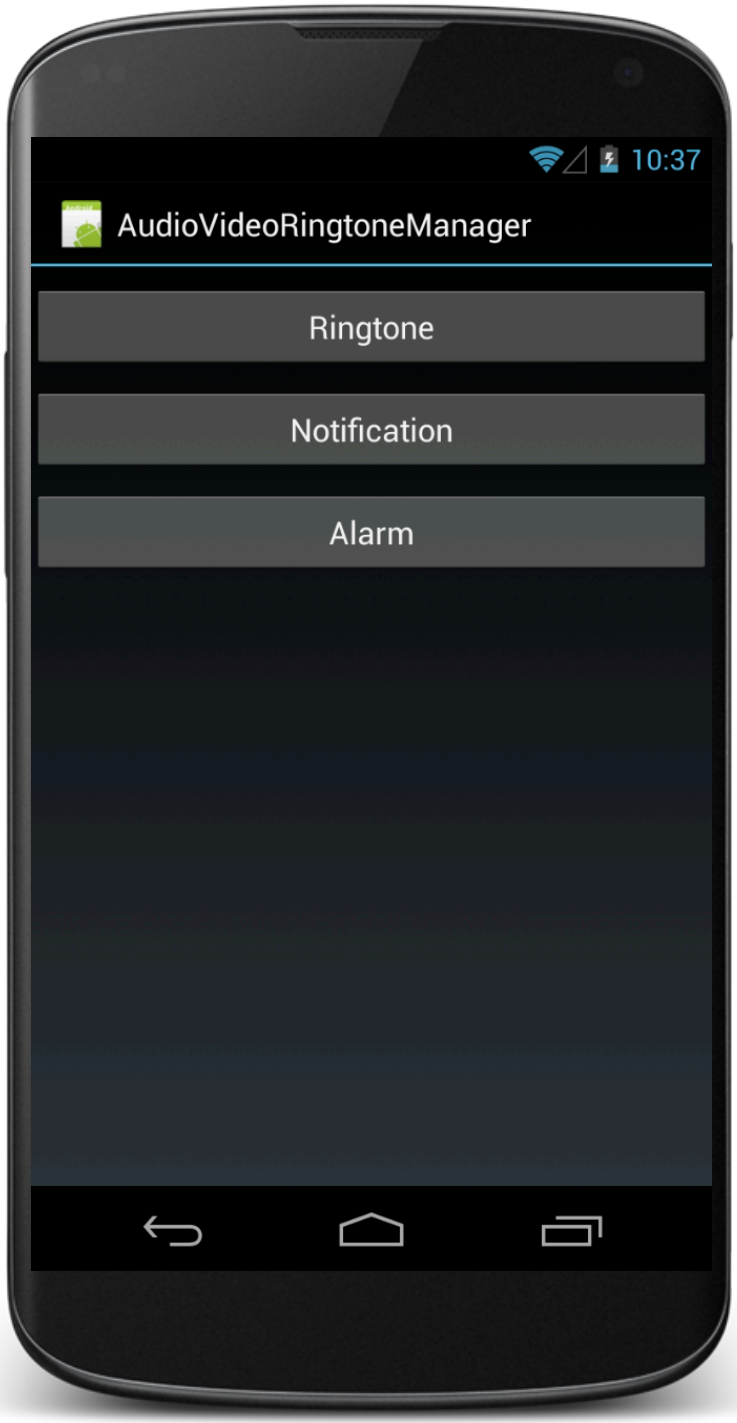
RingtoneManager provides access to audio clips used for incoming phone calls, notifications, alarms, etc.

Allows applications to get and set ringtones and to play and stop playing them

AudioVideoRingtoneManager

Application presents three buttons labeled ringtone, notification and alarm

Pressing one of these buttons causes the associated default ringtone to play


10:37
AudioVideoRingtoneManager
Ringtone
Notification
Alarm

AudioVideoRingtoneManager

```
// Get the default Phone Ringer RingTone and play it.
final Button ringtoneButton = (Button) findViewById(R.id.button1);
ringtoneButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        Uri ringtoneUri = RingtoneManager
                .getDefaultUri(RingtoneManager.TYPE_RINGTONE);

        playRingtone(RingtoneManager.getRingtone(
                getApplicationContext(), ringtoneUri));
    }
});
```

MediaPlayer

Controls playback of audio and video streams and files

Allows applications to control playback

Operates according to a complex state machine

See:
http://developer.android.com/
reference/android/media/
MediaPlayer.html

Some MediaPlayer Methods

setDataSource()

prepare()

start()

pause()

seekTo()

stop()

release()

VideoView

SurfaceView for displaying video files

Can load video from multiple sources

Provides various display options & convenience functions

AudioVideoVideoPlay

Application plays a movie in a VideoView


11:09
AudioVideoVideoPlay

AudioVideoVideoPlay

```
mVideoView = (VideoView) findViewById(R.id.videoViewer);

// Add a Media controller to allow forward/reverse/pause/resume

final MediaController mMediaController = new MediaController(
        AudioVideoVideoPlayActivity.this, true);

mMediaController.setEnabled(false);

mVideoView.setMediaController(mMediaController);

mVideoView
        .setVideoURI(Uri
                .parse("android.resource://course.examples.AudioVideo.VideoPlay/raw/moon"));

// Add an OnPreparedListener to enable the MediaController once the video is ready
mVideoView.setOnPreparedListener(new OnPreparedListener() {

    @Override
    public void onPrepared(MediaPlayer mp) {
        mMediaController.setEnabled(true);
    }
});
```

AudioVideoVideoPlay

```java
// Clean up and release resources
@Override
protected void onPause() {

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

MediaRecorder

Used to record audio and video

Operates in accordance to a state machine

See:
http://developer.android.com/

reference/android/media/
MediaRecorder.html

Some MediaRecorder Methods


setAudioSource()

setVideoSource()


setOutputFormat()

prepare()

start()

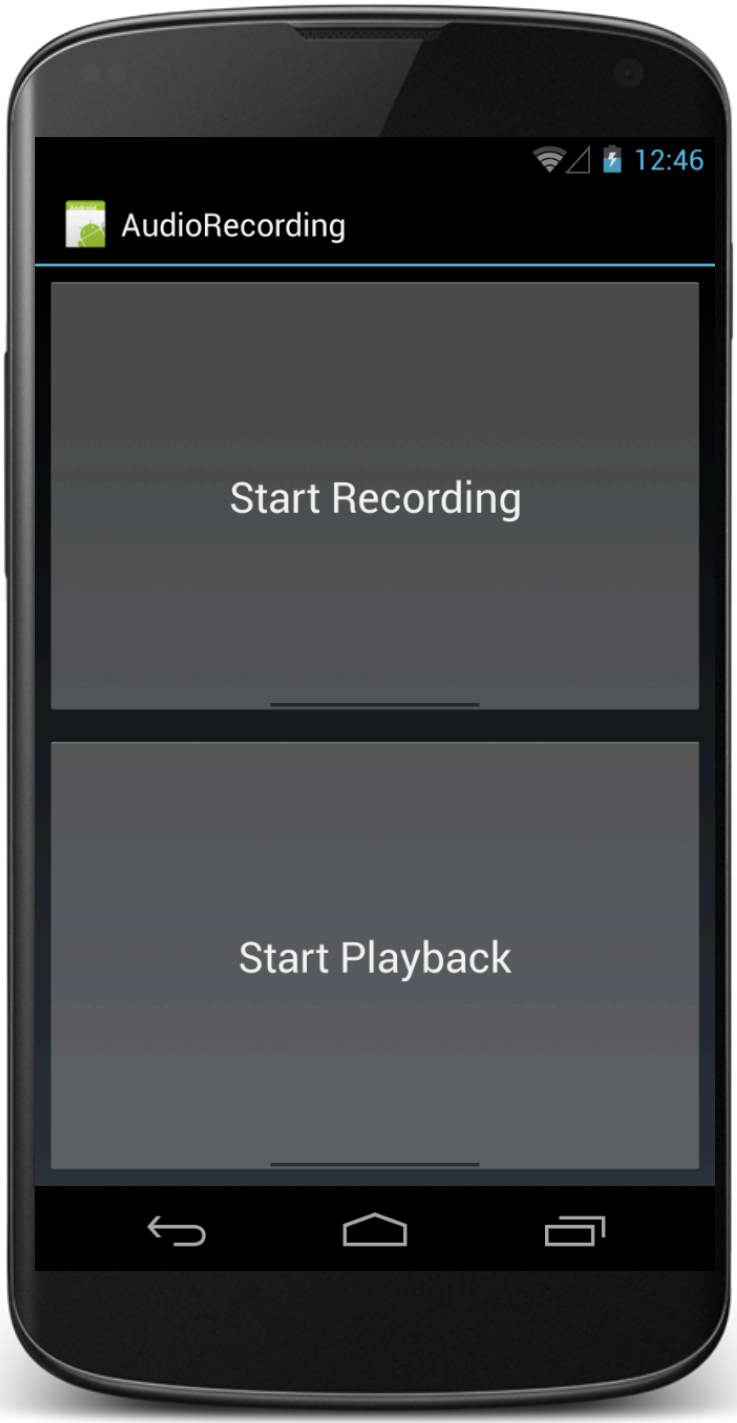
stop()

release()

AudioVideoAudioRecording

Can record audio from the user

Can play back recorded audio


12:46
AudioRecording
Start Recording
Start Playback

AudioVideoAudioRecording

```
// Start recording with MediaRecorder
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) {
        Log.e(TAG, "Couldn't prepare and start MediaRecorder");
    }

    mRecorder.start();
}

// Stop recording. Release resources
private void stopRecording() {

    if (null != mRecorder) {
        mRecorder.stop();
        mRecorder.release();
        mRecorder = null;
    }

}
```

AudioVideoAudioRecording

```
// Playback audio using MediaPlayer
private void startPlaying() {

    mPlayer = new MediaPlayer();
    try {
        mPlayer.setDataSource(mFileName);
        mPlayer.prepare();
        mPlayer.start();
    } catch (IOException e) {
        Log.e(TAG, "Couldn't prepare and start MediaPlayer");
    }

}

// Stop playback. Release resources
private void stopPlaying() {

    if (null != mPlayer) {
        if (mPlayer.isPlaying())
            mPlayer.stop();
        mPlayer.release();
        mPlayer = null;
    }

}
```

AudioVideoAudioRecording

```
// Listen for Audio Focus changes
OnAudioFocusChangeListener afChangeListener = new OnAudioFocusChangeListener() {

    @Override
    public void onAudioFocusChange(int focusChange) {

        if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
            mAudioManager.abandonAudioFocus(afChangeListener);

            // Stop playback, if necessary
            if (mPlayer.isPlaying())
                stopPlaying();
        }

    }

};
```

AudioVideoAudioRecording

```
// Release recording and playback resources, if necessary
@Override
public void onPause() {
    super.onPause();

    if (null != mRecorder) {
        mRecorder.release();
        mRecorder = null;
    }

    if (null != mPlayer) {
        mPlayer.release();
        mPlayer = null;
    }

}
```

Camera

Client for the Camera service, which manages the actual camera hardware

Manages image capture settings

Start/stops preview

Takes pictures

Camera Permissions

```
<uses-permission
        android:name="android.permission.CAMERA" />

<uses-feature
        android:name="android.hardware.camera" />

<uses-feature android:name=
        "android.hardware.camera.autofocus" />
```

Using the Camera

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

AudioVideoCamera

Takes still photos using the device's display as the camera's viewfinder

AudioVideoCamera

```
// Start the preview
private void startPreview() {
    if (null != mCamera) {
        try {
            mCamera.startPreview();
            mIsPreviewing = true;
        } catch (Exception e) {
            Log.e(TAG, "Failed to start preview");
        }
    }
}

// Shutdown preview
private void stopPreview() {
    if (null != mCamera && mIsPreviewing) {
        try {
            mCamera.stopPreview();
            mIsPreviewing = false;
        } catch (Exception e) {
            Log.e(TAG, "Failed to stop preview");
        }
    }
}

// Release camera so other applications can use it.
private void releaseCameraResources() {
    if (null != mCamera) {
        mCamera.release();
        mCamera = null;
    }
}
```

AudioVideoCamera

```java
@Override
protected void onResume() {
    super.onResume();

    if (null == mCamera) {
        try {

            // Returns first back-facing camera or null if no camera is
            // available.
            // May take a long time to complete
            // Consider moving this to an AsyncTask
            mCamera = Camera.open();

        } catch (RuntimeException e) {
            Log.e(TAG, "Failed to acquire camera");
        }

        // Ensure presence of camera or finish()
        if (null == mCamera)
            finish();
    }
}
```

AudioVideoCamera

```java
@Override
protected void onPause() {

    // Disable touches on mFrame
    mFrame.setEnabled(false);

    // Shutdown preview
    stopPreview();

    // Release camera resources
    releaseCameraResources();

    super.onPause();

}
```

Next Time

Sensors