Developing Media Applications on Android

Shawn Van Every - 3/7/2011 - AnDevCon

Utilizing the built-in Camera via an Intent

- Easiest method to allow image capture
- Familiar and capable UI
- Least flexible

Utilizing the built-in Camera via an Intent

```
Intent i = new Intent
(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(i, CAMERA_RESULT);

protected void onActivityResult(
  int requestCode,
  int resultCode, Intent intent) {
      ...
      Get Bundle extras = intent.getExtras();
      Bitmap bmp = (Bitmap) extras.get("data");
      ....
```

Capturing Full Size Images

```
File imageFile = new File(
  Environment.getExternalStorageDirectory().getAbsolutePath
  () + "/myfavoritepicture.jpg"
);
Uri imageFileUri = Uri.fromFile(imageFile);

Intent i = new Intent
(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT,
imageFileUri);
startActivityForResult(i, CAMERA_RESULT);
```

Displaying Large Images

```
BitmapFactory.Options bmpFactoryOptions = new
BitmapFactory.Options();
bmpFactoryOptions.inSampleSize = 8;
Bitmap bmp = BitmapFactory.decodeFile(imageFilePath,
bmpFactoryOptions);
```

Sizing based on screen

```
BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
bmpFactoryOptions.inJustDecodeBounds = true;
Bitmap bmp = BitmapFactory.decodeFile(imageFilePath, bmpFactoryOptions);
int heightRatio = (int)Math.ceil(bmpFactoryOptions.outHeight/(float)
currentDisplay.getHeight());
int widthRatio = (int)Math.ceil(bmpFactoryOptions.outWidth/(float)
currentDisplay.getWidth());
if (heightRatio > 1 && widthRatio > 1) {
   if (heightRatio > widthRatio) {
      bmpFactoryOptions.inSampleSize = heightRatio;
   } else {
      bmpFactoryOptions.inSampleSize = widthRatio;
   }
}
bmpFactoryOptions.inJustDecodeBounds = false;
bmp = BitmapFactory.decodeFile(imageFilePath, bmpFactoryOptions);
```

Image and Metadata Storage - MediaStore

```
ContentValues contentValues = new ContentValues(3);

contentValues.put(Media.DISPLAY_NAME, "This is a test title");
contentValues.put(Media.DESCRIPTION, "This is a test description");
contentValues.put(Media.MIME_TYPE, "image/jpeg");

Uri imageFileUri = getContentResolver().insert(
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
contentValues);
```

Retrieving Image

```
Bitmap bmp = BitmapFactory.decodeStream(
getContentResolver().openInputStream(imageFileUri), null,
bmpFactoryOptions);
```

Adding Metadata

```
ContentValues contentValues = new ContentValues(2);
contentValues.put(Media.DISPLAY_NAME, "This is a test title");
contentValues.put(Media.DESCRIPTION, "This is a test description");
getContentResolver().update(imageFileUri,contentValues,null,null);
```

Building a Custom Camera Application

- Tricky to get right
- Flexible UI
- http://developer.android.com/reference/android/hardware/Camera.html

Permissions

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

Camera Preview Surface

```
<SurfaceView android:id="@+id/CameraView"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
</SurfaceView>
```

SurfaceHolder.Callback

```
public class SnapShot extends Activity implements SurfaceHolder.Callback {
   SurfaceView cameraView;
   SurfaceHolder surfaceHolder;

public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   cameraView = (SurfaceView) this.findViewById(R.id.CameraView);
   surfaceHolder = cameraView.getHolder();
   surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
   surfaceHolder.addCallback(this);
}
public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {}
public void surfaceCreated(SurfaceHolder holder) {}
public void surfaceDestroyed(SurfaceHolder holder) {}
```

Implementing the Camera

```
Camera camera;
public void surfaceCreated(SurfaceHolder holder) {
   camera = Camera.open();
   try {
      camera.setPreviewDisplay(holder);
   } catch (IOException exception) {
      camera.release();
   }
   camera.setDisplayOrientation(90);
   camera.startPreview();
}

public void surfaceDestroyed(SurfaceHolder holder) {
   camera.stopPreview(); camera.release();
}
```

Capturing and Saving

```
public class SnapShot extends Activity implements SurfaceHolder.Callback,
Camera.PictureCallback {
...
  public void onPictureTaken(byte[] data, Camera camera) {
    Uri imageFileUri = getContentResolver().insert
    (Media.EXTERNAL_CONTENT_URI, new ContentValues());
    try {
      OutputStream imageFileOS = getContentResolver
      ().openOutputStream(imageFileUri);
      imageFileOS.write(data);
      imageFileOS.flush();
      imageFileOS.close();
    } catch (FileNotFoundException e) {
    } catch (IOException e) {}
  }
```

Capturing and Saving Cont.

```
public void onCreate(Bundle savedInstanceState) {
...
  cameraView.setFocusable(true);
  cameraView.setFocusableInTouchMode(true);
  cameraView.setClickable(true);
  cameraView.setOnClickListener(this);
}

public void onClick(View v) {
  camera.takePicture(null, null, null, this);
}
```

Creating a Double Exposure Camera App

- Using a custom camera
- Draw a Bitmap on a Bitmap

Update onPictureTaken

```
File imageFile1;
File imageFile2;
int currentPicture = 1;

public void onPictureTaken(byte[] data, Camera camera) {
	try {
		File imageFile = File.createTempFile("doubleexposure", ".jpg");
		FileOutputStream imageFileOS = new FileOutputStream(imageFile);
		imageFileOS.write(data);
		imageFileOS.flush();
		imageFileOS.close();

	if (currentPicture == 1) {
		imageFile1 = imageFile;
		currentPicture++;
		camera.startPreview();
	} else if (currentPicture == 2) {
		imageFile2 = imageFile;
		cameraView.setVisibility(View.INVISIBLE);
		 ...
```

Add an ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <SurfaceView android:id="@+id/CameraView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    </SurfaceView>
    <ImageView android:id="@+id/DoubleExposureView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    </ImageView>
</FrameLayout>
```

Create the Bitmaps

```
BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
bmpFactoryOptions.inJustDecodeBounds = true;
Bitmap bmp = BitmapFactory.decodeFile(imageFile1.getPath(), bmpFactoryOptions);

int heightRatio = (int)Math.ceil(bmpFactoryOptions.outHeight/(float)getWindowManager
().getDefaultDisplay().getHeight());
int widthRatio = (int)Math.ceil(bmpFactoryOptions.outWidth/(float)getWindowManager
().getDefaultDisplay().getWidth());

if (heightRatio > 1 && widthRatio > 1) {
   if (heightRatio > widthRatio) {
      bmpFactoryOptions.inSampleSize = heightRatio;
   } else {
      bmpFactoryOptions.inSampleSize = widthRatio;
   }
}
bmpFactoryOptions.inJustDecodeBounds = false;

Bitmap picture1bmp = BitmapFactory.decodeFile(imageFile1.getPath(), bmpFactoryOptions);
Bitmap picture2bmp = BitmapFactory.decodeFile(imageFile2.getPath(), bmpFactoryOptions);
```

Create the Bitmap for Drawing

```
Bitmap drawingBmp = Bitmap.createBitmap(picture1bmp.getWidth(),
picture1bmp.getHeight(), picture1bmp.getConfig());
Canvas canvas = new Canvas(drawingBmp);
Paint paint = new Paint();
```

Draw the first Bitmap

```
canvas.drawBitmap(picture1bmp, 0, 0, paint);
```

Set the Paint and Draw the second Bitmap

```
paint.setXfermode(new PorterDuffXfermode
(android.graphics.PorterDuff.Mode.MULTIPLY));
canvas.drawBitmap(picture2bmp, 0, 0, paint);
```

Set and Display the ImageView

```
ImageView doubleExposureView = (ImageView) this.findViewById
(R.id.DoubleExposureView);
doubleExposureView.setImageBitmap(drawingBmp);
```

Other Paint Xfermodes

- LIGHTEN: Takes the lightest pixel of the two images from each position and shows that.
- DARKEN: Takes the darkest pixel from the two images in each position and shows that.
- MULTIPLY: Multiplies the two pixels from each position, divides by 255, and uses that value to create a new pixel for display. Result Color = Top Color x Bottom Color / 255

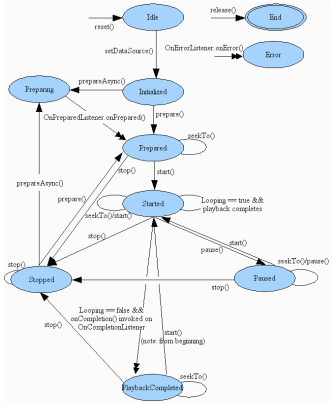
Audio Playback

Utilizing the built-in Audio Player via an Intent

```
Intent intent = new Intent
(android.content.Intent.ACTION_VIEW);
intent.setDataAndType(audioFileUri, "audio/mp3");
startActivity(intent);
```

Creating a Custom Audio-Playing Application

MediaPlayer States



MediaPlayer State Diagram from Android Developer site.

MediaPlayer Audio Example

Playing file from "res/raw"

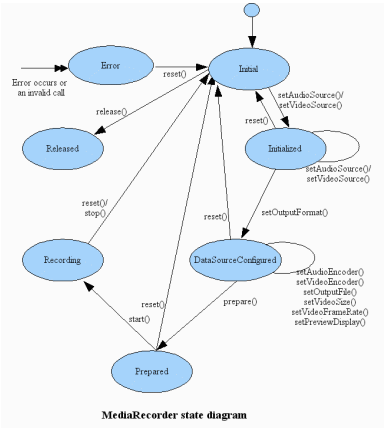
```
mediaPlayer = MediaPlayer.create(this, R.raw.goodmorningandroid);
mediaPlayer.start();
...
mediaPlayer.stop();
mediaPlayer.release();
```

OnCompletionListener

```
public class CustomAudioPlayer extends Activity implements
OnCompletionListener {
...
        mediaPlayer = MediaPlayer.create(this, R.raw.goodmorningandroid);
        mediaPlayer.setOnCompletionListener(this);
        mediaPlayer.start();
...
    public void onCompletion(MediaPlayer mp) {
        mediaPlayer.start();
    }
```

Audio Capture

MediaRecorder States



MediaRecorder State Diagram from Android Developer site.

Initialize and Set Source

```
MediaRecorder recorder = new MediaRecorder();
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

Set Output Format

```
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
```

Output Formats

- MPEG_4: This specifies that the file written will be an MPEG-4 file. It may contain both audio and video.
- RAW_AMR: This represents a raw file without any type of container. This should be used only when capturing audio when the audio encoder is AMR_NB.
- THREE_GPP: This specifies that the file written will be a 3GPP file (extension .3gp). It may contain both audio and video tracks.

Set Audio Encoder

```
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

Set Output

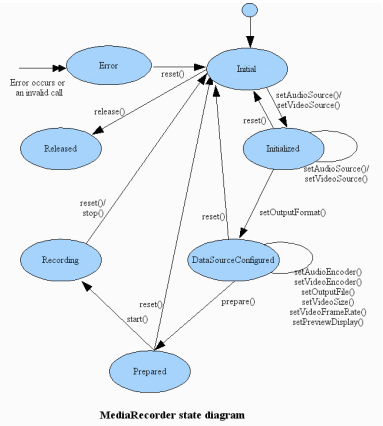
```
File path = new File(Environment.getExternalStorageDirectory
().getAbsolutePath() + "/Android/data/com.mobvcasting.ar/files/");
path.mkdirs();
audioFile = File.createTempFile("recording", ".3gp", path);
recorder.setOutputFile(audioFile.getAbsolutePath());
```

Prepare and Start Recording

```
recorder.prepare();
recorder.start();
```

Video Capture

MediaRecorder States



MediaRecorder State Diagram from Android Developer site.

MediaRecorder for Video

- setAudioSource, setVideoSource
- setOutputFormat
- setAudioEncoder, setVideoEncoder
- setAudioEncodingBitrate, setVideoEncodingBitRate
- setAudioSamplingRate, setAudioChannels
- setVideoFrameRate, setVideoSize
- setOutputFile

OR using CamcorderProfile

```
recorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);

CamcorderProfile cpHigh = CamcorderProfile
                .get(CamcorderProfile.QUALITY_HIGH);
recorder.setProfile(cpHigh);
recorder.setOutputFile("/sdcard/videocapture_example.mp4");
```

CamcorderProfile.QUALITY_HIGH

- Audio Bit Rate: 12,200 bits per second
- Audio Channels: 1
- Audio Codec: AMR-NB
- Audio Sample Rate: 8000 Hz
- Duration: 60 seconds
- File Format: MP4
- Video Bit Rate: 3,000,000 bits per second
- Video Codec: H.264
- Video Frame Width: 720 pixels
- Video Frame Height: 480 pixels
- Video Frame Rate: 24 frames per second

CamcorderProfile.QUALITY_LOW

- Audio Bit Rate: 12,200 bits per second
- Audio Channels: 1
- Audio Codec: AMR-NB
- Audio Sample Rate: 8000 Hz
- Duration: 30 seconds
- File Format: 3GPP
- Video Bit Rate: 256,000 bits per second
- Video Codec: 3
- Video Frame Width: 176 pixels
- Video Frame Height: 144 pixels
- Video Frame Rate: 15 frames per second

Custom Video Capture

Initialize MediaRecorder

```
MediaRecorder recorder;
boolean recording = false;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
     ...
    recorder = new MediaRecorder();

    recorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
    recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);

    CamcorderProfile cpHigh = CamcorderProfile
            .get(CamcorderProfile.QUALITY_HIGH);
    recorder.setProfile(cpHigh);
    recorder.setOutputFile("/sdcard/videocapture_example.mp4");
    recorder.setMaxDuration(50000); // 50 seconds
    recorder.setMaxFileSize(5000000); // Approximately 5 megabytes
```

SurfaceView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <SurfaceView android:id="@+id/CameraView"
      android:layout_width="640px"
      android:layout_height="480px">
      </SurfaceView>
</LinearLayout>
```

SurfaceHolder.Callback

```
public class VideoCapture extends Activity implements OnClickListener, SurfaceHolder.Callback {
      SurfaceHolder holder;

      public void onCreate(Bundle savedInstanceState) {
           ...
            SurfaceView cameraView = (SurfaceView) findViewById(R.id.CameraView);
            holder = cameraView.getHolder();
            holder.addCallback(this);
            holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
      }

       public void surfaceCreated(SurfaceHolder holder) {
           recorder.setPreviewDisplay(holder.getSurface());
           try {
               recorder.prepare();
           } catch (IllegalStateException e) {
           } catch (IOException e) {
           }
      }

      public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
      }

      public void surfaceDestroyed(SurfaceHolder holder) {
      }
```

Starting and Stopping Recording

```
public void onClick(View v) {
    if (recording) {
        recorder.stop();
        recorder.release();
        recording = false;
    } else {
        recording = true;
        recorder.start();
    }
}
```