Android Camera

Alexander Nelson

October 6, 2017

University of Arkansas - Department of Computer Science and Computer Engineering

Why use the camera?

Why not?

Translate

Call

Learn

Capture

How to use the Android Camera

Two ways to include the camera sensor in your application:

1. Wrap camera app with intent
2. Use the Camera/Camera2 API

Using a Camera Intent

Why use the Camera Intent?

Intent – Simply wrap the existing application

- No need to reinvent the wheel
- Has all the expected user functionality

How to use the Intent

Steps to use the camera

1. Request use of the camera
2. Wrap the camera with an intent
3. Get the image from the returned intent
    - Can get thumbnail from returned Intent object
    - Can put a URI extra for the camera to place the full image
4. (optional) Make the photo available to other applications

Request Permission

Declare in manifest:

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera"
                  android:required="true" />
    ...
</manifest>
```

Wrap the Camera App with Intent

Use MediaStore.ACTION_IMAGE_CAPTURE as class

```
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

Get the Image Thumbnail

The thumbnail is returned as part of the Result Intent

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

Save the Full Resolution Image to a File

Request write permission if SDK version <18

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
                     android:maxSdkVersion="18" />
    ...
</manifest>
```

Save the Full Resolution Image to a File

Create a file where the image will be saved

```
String mCurrentPhotoPath;

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
        imageFileName,  /* prefix */
        ".jpg",         /* suffix */
        storageDir      /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}
```

Save the Full Resolution Image to a File

Pass the file location as a URI as an extra in the Intent Object

```
static final int REQUEST_TAKE_PHOTO = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            // Error occurred while creating the File
            ...
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
            Uri photoURI = FileProvider.getUriForFile(this,
                                                      "com.example.android.fileprovider",
                                                      photoFile);
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
            startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
        }
    }
}
```

Particulars

FileProvider.getUriForFile() – Will not allow passing of a file:// URI after API 24

- Requires use of a defined FileProvider in the Application

```
<application>
    ...
    <provider
         android:name="android.support.v4.content.FileProvider"
         android:authorities="com.example.android.fileprovider"
         android:exported="false"
         android:grantUriPermissions="true">
         <meta-data
             android:name="android.support.FILE_PROVIDER_PATHS"
             android:resource="@xml/file_paths"></meta-data>
     </provider>
     ...
</application>
```

Particulars

Create an XML Resource Directory with a file_paths.xml file

Place the directory where photos are to be saved

```xml
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="my_images" path="Android/data/com.example.package.name/files/Pictures" />
</paths>
```

This location corresponds to the path for getExternalFilesDir()

Particulars

If you don't want to save them to a private directory getExternalStoragePublicDirectory()

- Pass DIRECTORY_PICTURES as an argument

Camera2 API

Why use the Camera2 API?

Reasons to use the Camera2 API:

- Draw on the image
- Automatically capture certain images
- Feed raw frame data as a sensor point

In short, if you need access to each frame

How to get access to the camera sensor

CameraManager – System service manager for detecting, characterizing, and connecting to Cameras

CameraManager camManager = Context.getSystemService(CameraManager.class);

How to get access to the camera sensor

CameraManager.getCameraIdList() – Returns a list of all connected camera devices

- Identified by a String called cameraId
- cameraId can be used to query, check if available, and open the camera

How to get access to the camera sensor

CameraManager.getCameraCharacteristics(cameraId) – Returns a CameraCharacteristics object which describes an attached camera

CameraCharacteristics – Dictionary-like object with key-value pairs describing the camera

Camera Characteristics

Useful Characteristics fields:

- LENS_FACING
  - LENS_FACING_FRONT
  - LENS_FACING_BACK
  - LENS_FACING_EXTERNAL
- SCALER_STREAM_CONFIGURATION_MAP – Describes the streaming ability of the camera
- SENSOR_ORIENTATION – Clockwise angle which the output image must be rotated to be upright on the device
- FLASH_INFO_AVAILABLE – Boolean whether flash available or not

...And many more

How to get access to the camera sensor

To capture, a CameraCaptureSessions is required

- Requires set of output surfaces

createCaptureSession(List,CameraCaptureSession.StateCallback,Handler)

- List<Surface>– Set of output surfaces
- StateCallback – The callback which will occur after the configuration finishes
- Handler – Where the callback should be invoked (i.e. Which thread)

What is an appropriate output surface?

Capture session requires access to print to an output surface
Appropriate surfaces:

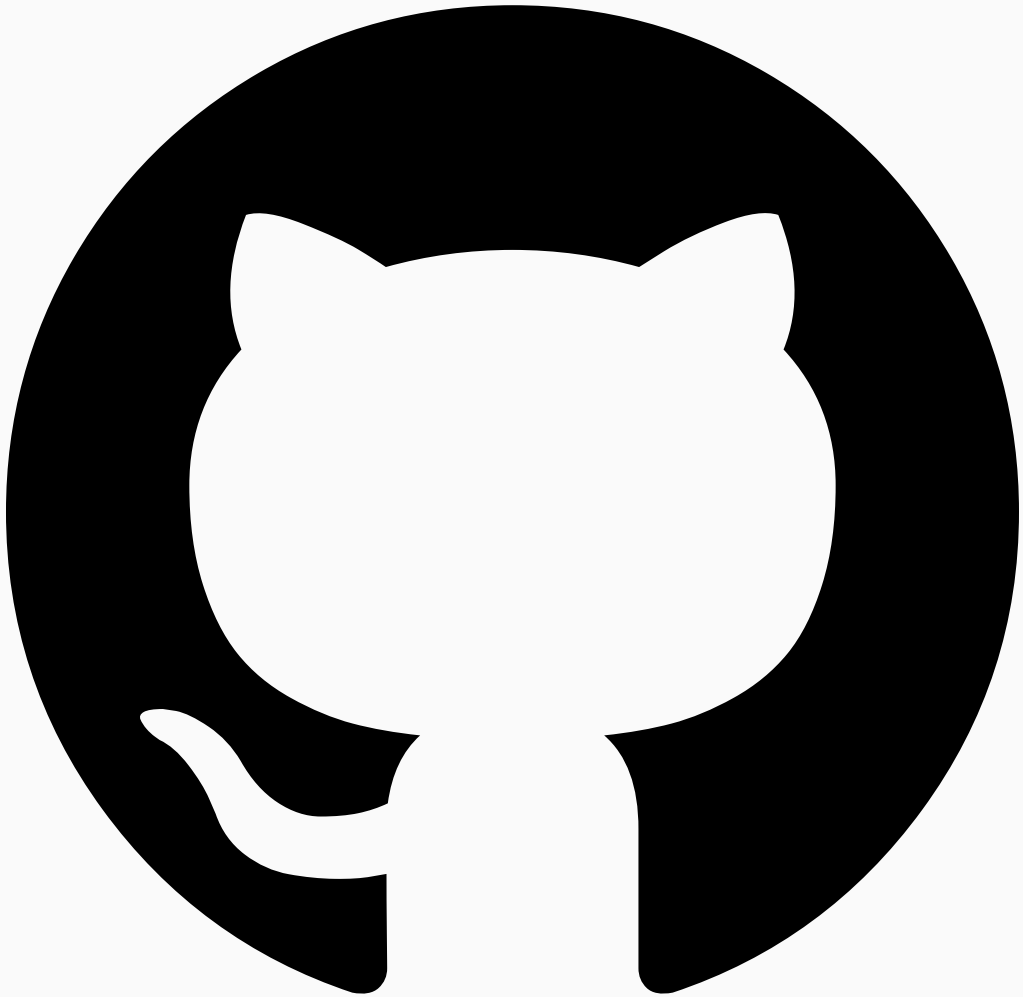
- SurfaceView – Drawable View
- TextureView – Hardware accelerated View
- OpenGL Texture – SurfaceTexture
- Recording – MediaCodec with createInputSurface()
- Recording – MediaRecorder with getSurface()

Create a CaptureRequest

Once the capture session has been created, the application should construct a CaptureRequest

- Defines the capture parameters to get a single image
- Lists which output surfaces should be used as target for the capture
- Can be used for one-shot capture or repeating capture

Code Example



Google Samples Camera2 API