6. Specialized Instrument and Devices

Objectives

- Explain basic requirements of a camera app
- Identify and explain other hardware sensors
- Develop app that could capture image and video
- Implement app that can play audio and video files

Camera - Considerations

- Consider the following:
  - Camera requirement: Is it a must or a need?
  - Quick picture or customized camera: Taking a quick picture or developing a new way to use camera?
  - Storage: private or public? Internal or external?

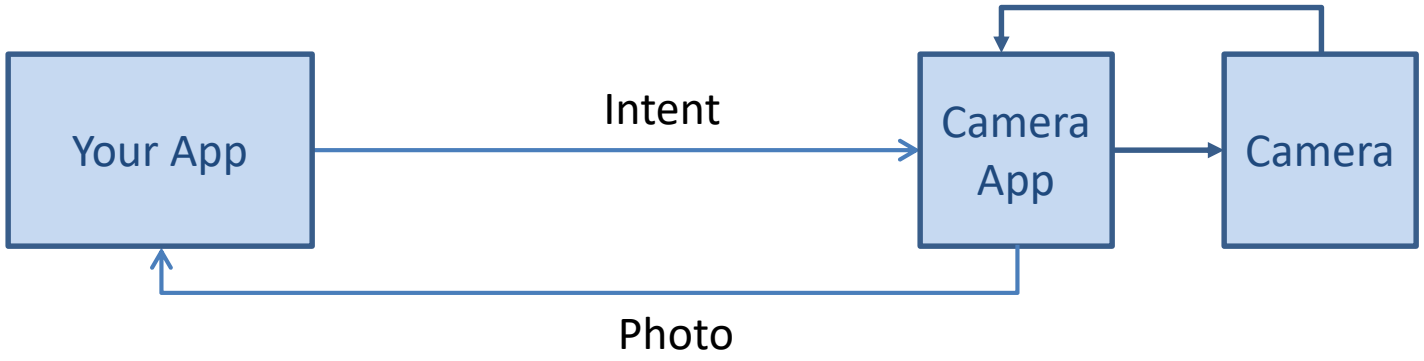
https://developer.android.com/training/camera

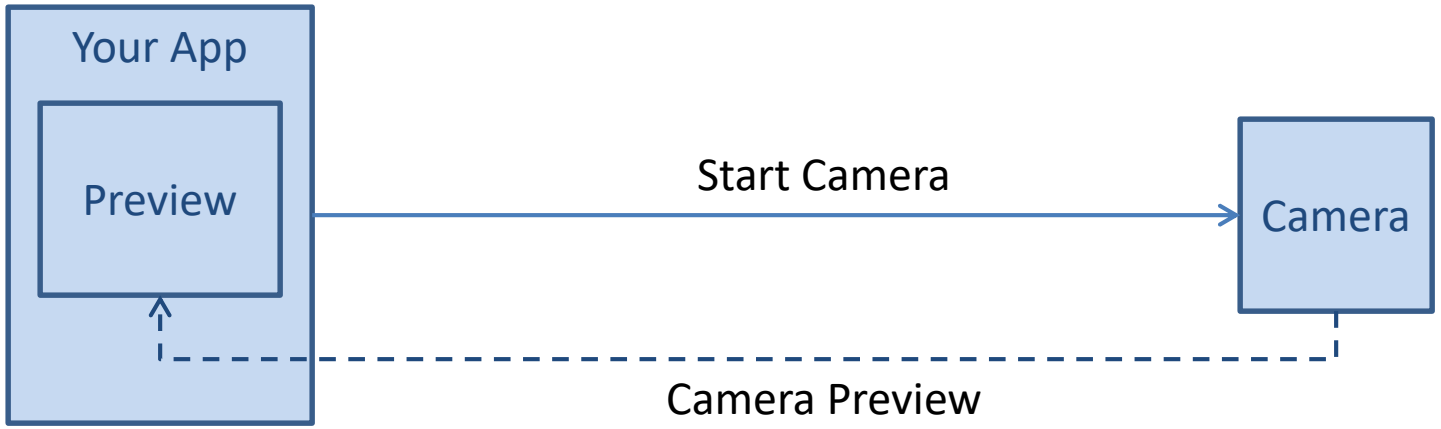
Methods

1. Use existing camera app
   - Minimum code
   - Use Intent

2. Build your own camera function
   - Write your own code
   - Use CameraX

The Basics


Your App
Intent
Camera App
Camera
Photo


Your App
Preview
Start Camera
Camera
Camera Preview

Manifest Declarations

1. Camera Permission
   - If using Intent: No need permission
   - If using Camera:

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

2. Camera Features

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

   - Prevents your app from being installed to devices that do not include a camera

Manifest Declarations

3. Storage Permission

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

4. Audio Recording Permission

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

5. Location Permission

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

Using Camera Intent

- A quick way to enable taking pictures or videos in your app with very minimum coding efforts
- Process:
  1. Create an Intent using MediaStore.ACTION_IMAGE_CAPTURE or MediaStore.ACTION_VIDEO_CAPTURE
  2. Execute the Intent using startActivity()
  3. Set up an onActivityResult() to receive Intent result

Take a photo

Kotlin

```kotlin
val REQUEST_IMAGE_CAPTURE = 1

private fun dispatchTakePictureIntent() {
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
        takePictureIntent.resolveActivity(packageManager)?.also {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
        }
    }
}
```

Java

```java
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 thumbnail

Kotlin

```kotlin
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        val imageBitmap = data.extras.get("data") as Bitmap
        imageView.setImageBitmap(imageBitmap)
    }
}
```

Java

```java
@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");
        imageView.setImageBitmap(imageBitmap);
    }
}
```

Save the full-size photo

- If you give the camera app a file to save into, it saves a full-size photo
- Photos are saved in the public external storage provided by the getExternalStoragePublicDirectory()
- Use the DIRECTORY_PICTURES argument

Save the full-size photo

- Create authorities string and store it in a dedicated resource file

- E.g. res/xml/file_paths.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>
```

Save the full-size photo

- Configure `FileProvider` in the manifest file

```
<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>
```

Save the full-size photo

Kotlin

```kotlin
var currentPhotoPath: String

@Throws(IOException::class)
private fun createImageFile(): File {
    // Create an image file name
    val timeStamp: String = SimpleDateFormat("yyyyMMdd_HHmmss").format(Date())
    val storageDir: File = getExternalFilesDir(Environment.DIRECTORY_PICTURES)
    return File.createTempFile(
            "JPEG_${timeStamp}_", /* prefix */
            ".jpg", /* suffix */
            storageDir /* directory */
    ).apply {
        // Save a file: path for use with ACTION_VIEW intents
        currentPhotoPath = absolutePath
    }
}
```

Save the full-size photo

Java

```java
String currentPhotoPath;

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
    currentPhotoPath = image.getAbsolutePath();
    return image;
}
```

Save the full-size photo

Kotlin

```kotlin
val REQUEST_TAKE_PHOTO = 1

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

Save the full-size photo

Java

```
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);
        }
    }
}
```

Decode a scaled image

- App may running our of memory after displaying too many images

- Images should be scaled to match the size of the destination view

Decode a scaled image

Kotlin

```kotlin
private fun setPic() {
    // Get the dimensions of the View
    val targetW: Int = imageView.width
    val targetH: Int = imageView.height

    val bmOptions = BitmapFactory.Options().apply {
        // Get the dimensions of the bitmap
        inJustDecodeBounds = true

        val photoW: Int = outWidth
        val photoH: Int = outHeight

        // Determine how much to scale down the image
        val scaleFactor: Int = Math.min(photoW / targetW, photoH / targetH)

        // Decode the image file into a Bitmap sized to fill the View
        inJustDecodeBounds = false
        inSampleSize = scaleFactor
        inPurgeable = true
    }
    BitmapFactory.decodeFile(currentPhotoPath, bmOptions)?.also { bitmap ->
        imageView.setImageBitmap(bitmap)
    }
}
```

Decode a scaled image

Java

```java
private void setPic() {
    // Get the dimensions of the View
    int targetW = imageView.getWidth();
    int targetH = imageView.getHeight();

    // Get the dimensions of the bitmap
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;

    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    // Determine how much to scale down the image
    int scaleFactor = Math.min(photoW/targetW, photoH/targetH);

    // Decode the image file into a Bitmap sized to fill the View
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Bitmap bitmap = BitmapFactory.decodeFile(currentPhotoPath, bmOptions);
    imageView.setImageBitmap(bitmap);
}
```

Questions?

1. “You must obtain a permission to enable camera feature in your app.” Comment on this statement
2. How to prevent your app from being installed on devices that do not have a camera?
3. What is the quick way to enable taking pictures or videos in your app without a lot of extra code?

Audio Streams

- Android maintains a separate audio stream for:
  - playing music,
  - alarms,
  - notifications,
  - the incoming call ringer,
  - system sounds,
  - in-call volume, and
  - etc

Audio Stream

- Most of these streams are restricted to system events
- We focus on playing audio using the STREAM_MUSIC stream
- Use the STREAM_MUSIC stream for: background music or sound effects

Audio Playing

- Classes to play sound and video in the Android framework:
  - MediaPlayer: plays sound and video.
  - AudioManager: manages audio sources and audio output on a device.

MediaPlayer

- Plays local and external (streaming) files
- Supports any media codec that is provided by the Android platform and those that are device-specific
- Recommendation: use core media formats

MediaPlayer

- Permission declaration
  - If you are using MediaPlayer to stream network-based content

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

  - If your player application needs to keep the screen from dimming or the processor from sleeping

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

Core Media Format

- Audio: .3gp .mp3 .mp4 .mid .wav .ogg
- Picture: .jpg .gif .png .bmp
- Video: .3gp .mp4
- Full list: http://developer.android.com/guide/appendix/media-formats.html

MediaPlayer

Kotlin

```kotlin
//Playing local file
var mediaPlayer: MediaPlayer? = MediaPlayer.create(context, R.raw.sound_file_1)
mediaPlayer?.start()


val url = "http://........" // your URL here
val mediaPlayer: MediaPlayer? = MediaPlayer().apply {
    setAudioStreamType(AudioManager.STREAM_MUSIC)
    setDataSource(url)
    prepare() // might take long! (for buffering, etc)
    start()
}
```

Java

```java
//Playing local file
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start();


String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Releasing the MediaPlayer

- A MediaPlayer can consume valuable system resources
- Always call `release()` to make sure any system resources allocated to it are properly released

Kotlin

```
mediaPlayer?.release()
mediaPlayer = null
```

Java

```
mediaPlayer.release();
mediaPlayer = null;
```

Questions?

1. “You are free to use any media codec in your Android app”. Comment on this statement.

Sensors Overview

- Three categories

Motion	Environmental	Position
		

https://developer.android.com/guide/topics/sensors/sensors_overview

Motion sensors

- Measure acceleration forces and rotational forces along three axes
  - Accelerometers
  - gravity sensors
  - gyroscopes, and
  - rotational vector sensors

https://developer.android.com/guide/topics/sensors/sensors_motion

Environmental sensors

- Measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity
  - barometers
  - photometers, and
  - thermometers

https://developer.android.com/guide/topics/sensors/sensors_environment

Position sensors

- Measure the physical position of a device
  - orientation sensors
  - magnetometers

https://developer.android.com/guide/topics/sensors/sensors_position

Questions?

Mobile devices are equipped with sensors and hardware components. Discuss how mobile app could utilized these sensors or hardware components to provide a unique experience to users. Use a suitable example to explain your answer. (5 marks)