Lab 2

Sensor and Camera Programming

Lecture: Feb 2 / Feb 4
Demo: Feb 9 / Feb 11
Hard Deadline: Feb 16 / Feb 18

Today’s Schedule

- Lab 2 lecture (in zoom main session)

- Lab 2 implementation / discussion (in zoom main session)
- Lab 1 demo / Lab 2 question with TA (in zoom breakout room)

Lab 2a

Requirement

Lab 2a: sensor
I will show you:
- Display accelerometer sensor

You will do:
- Display gravity sensor

(If have more time)
- Display both on one screen

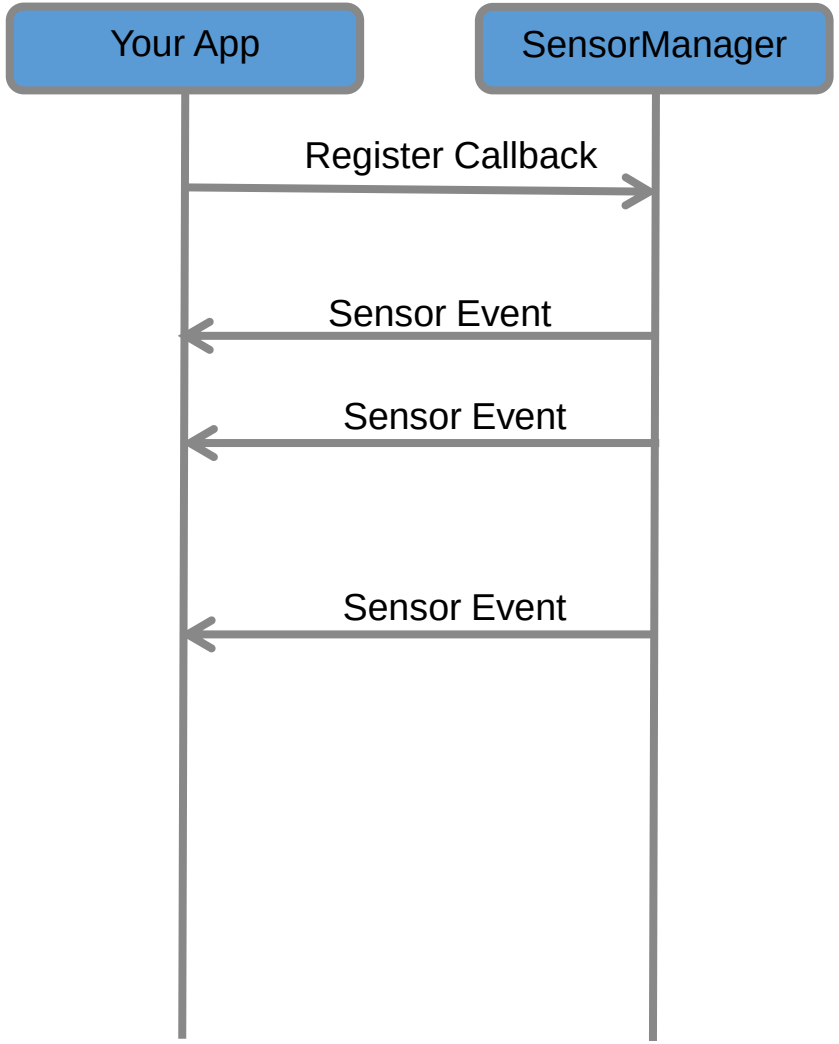
Demo

Sensor Programming

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

SensorEventListener

- Android’s sensors are controlled by external services and only send events when they choose to. An app must register a callback to be notified of a sensor event
- Each sensor has a related XXXXListener interface that your callback must implement e.g. LocationListener, SensorEventListener
- In order for an object to receive updates from a sensor, it must implement the SensorEventListener interface




```
public class MainActivity extends
AppCompatActivity implements
SensorEventListener {
```

onCreate

The non-media (e.g. not camera) sensors are managed by a variety of XXXXManager classes:
-LocationManager (GPS)
-SensorManager (accelerometer, gyro, proximity, light, temp)

```
//mSensorManager: one interface for multiple types of sensors, registering is to obtain a reference to the relevant manager
//mSensor: registration for one type of sensor, a reference to the specific sensor you are interested in updates from
```

```
public class MainActivity extends AppCompatActivity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor mSensor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
    }
```

onSensorChanged / onAccuracyChanged

```
// Called when a registered sensor changes value
   @Override
   public void onSensorChanged(SensorEvent event){
      EditText field = (EditText)findViewById(R.id.editText3);
      field.setText(event.values[0] + " / " + event.values[1] + " / " + event.values[2]);
   }

// Called when a registered sensor's accuracy changes
   @Override
   public void onAccuracyChanged(Sensor mSensor, int value) {

   }
```

```
public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Do something here if sensor accuracy changes.
        // You must implement this callback in your code.
        if (sensor == mValuen) {
            switch (accuracy) {
                case 0:
                    System.out.println("Unreliable");
                    con=0;
                    break;
                case 1:
                    System.out.println("Low Accuracy");
                    con=0;
                    break;
```

onResume / onPause

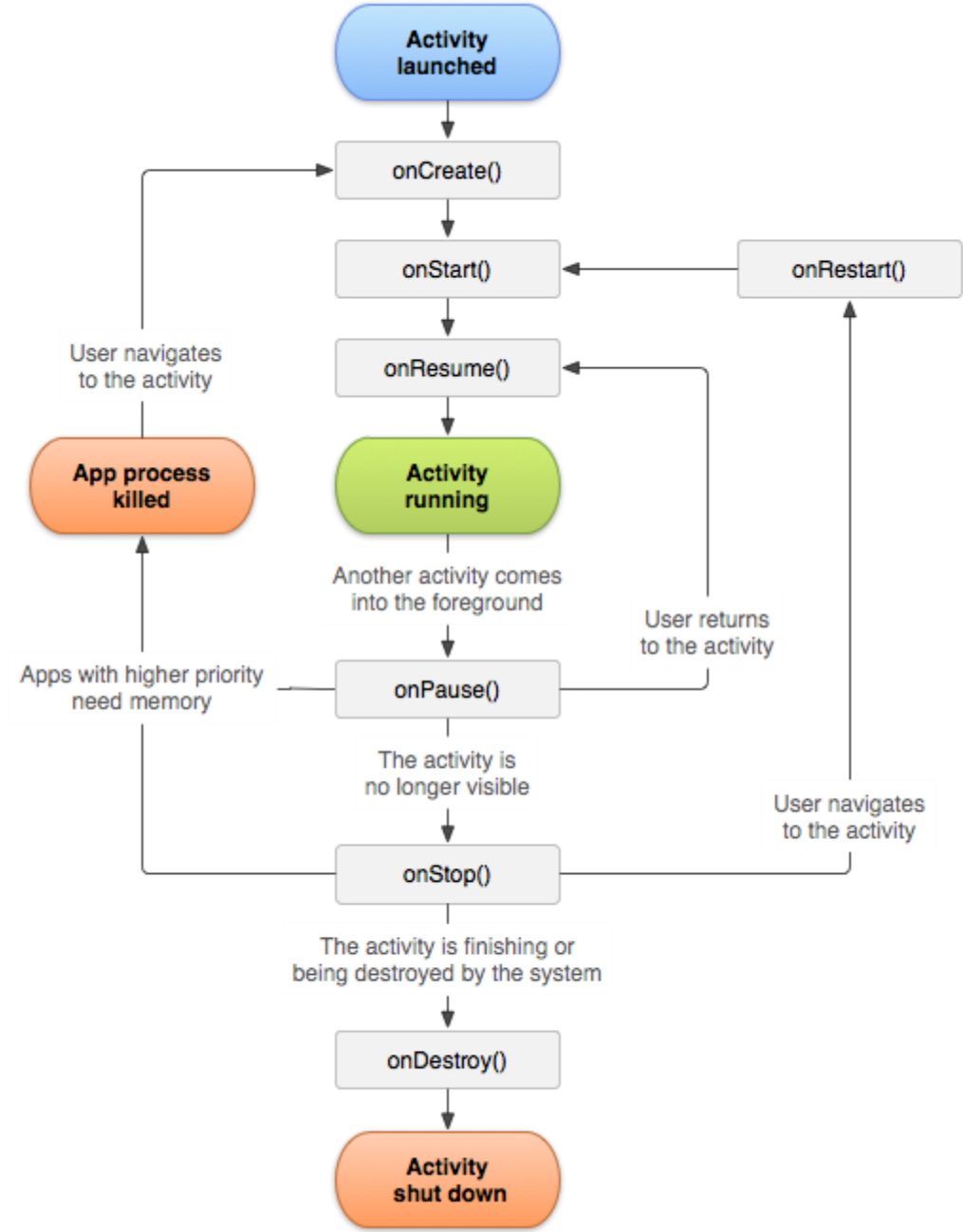
```
//The arguments passed into the registerListener method determine the sensor that
you are connected to and the rate at which it will send you updates

    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mSensor, SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }



}
```


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textMultiLine"
        android:id="@+id/editText3"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Linear Acceleration"
        android:id="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="10dp" />
</RelativeLayout>
```

Hint for multiple sensors

```
public class MyActivity … {

        private class AccelListener implements SensorEventListener {
                public void onSensorChanged(SensorEvent sensorEvent) {
                        …
                }
                public void onAccuracyChanged(Sensor arg0, int arg1) {}
        }

        private class LightListener implements SensorEventListener {
                public void onSensorChanged(SensorEvent sensorEvent) {
                        …
                }
                public void onAccuracyChanged(Sensor arg0, int arg1) {}
        }

        private SensorEventListener accelListener_ = new AccelListener();
        private SensorEventListener lightListener_ = new LightListener();

        …
        public void onResume(){
           …
         sensorManager_.registerListener(accelListener, accelerometer,
                                         SensorManager.SENSOR_DELAY_GAME);
         sensorManager_.registerListener(lightListener, lightsensor,
                                         SensorManager.SENSOR_DELAY_NORMAL);
        }
        public void onPause(){
           sensorManager_.unregisterListener(accelListener_);
           sensorManager_.unregisterListener(lightListener_);
        }
```

Lab 2b

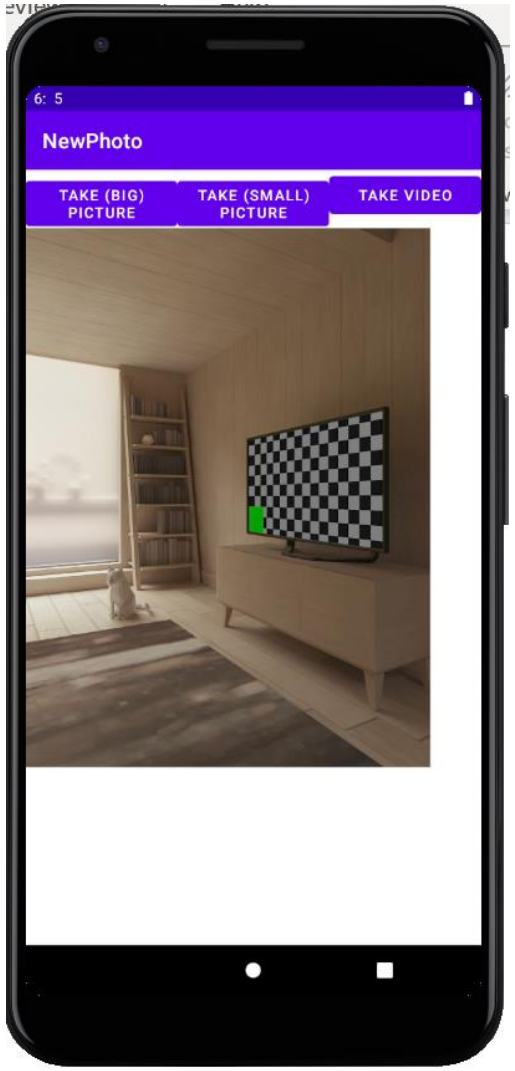
Requirement

Lab 2b: photo
I will show you:
- Take big picture and save it

You will do:
- Take small picture

(If have more time)
- Take video

Demo



Manifest

- Create a project called photo
- put a <user-feature> tag to advertise that your application depends on having a camera
- Enable the image capture action
- put a <uses-permission> tag to enable write permission to the external storage

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.photo">

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

   <queries>
      <intent>
         <action android:name="android.media.action.IMAGE_CAPTURE" />
      </intent>
   </queries>

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

Manifest

- Configure the FileProvider: add a provider to your application
- Make sure that the authorities string matches the argument to getUriForFile in the app

```
<application
        …
        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="com.example.photo.provider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths" />
        </provider>

        ….
</application>
```

Create the resource file

- The provider expects paths to be configured in res/xml/ **provider_paths**.xml
- Create a xml directory under res, create a **provider_paths**.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <external-path name="external_files" path="."/>
</paths>
```

Layout / String

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <LinearLayout
        android:orientation="horizontal"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        >
        <Button android:text="@string/btnIntend" android:id="@+id/btnIntend"
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="1" />
        <Button android:text="@string/btnIntendS" android:id="@+id/btnIntendS"
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="1" />
        <Button android:text="@string/btnIntendV" android:id="@+id/btnIntendV"
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="1" />
    </LinearLayout>
    <ImageView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:visibility="visible"
        android:id="@+id/imageView1" />
    <VideoView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:visibility="invisible"
        android:id="@+id/videoView1" />
</LinearLayout>
```

```
<resources>
    <string name="app_name">Photo</string>
    <string name="btnIntend">Take (big) Picture</string>
    <string name="btnIntendS">Take (small)
Picture</string>
    <string name="btnIntendV">Take Video</string>

</resources>
```

Main Code

```
public class MainActivity extends AppCompatActivity {

    private static final int ACTION_TAKE_PHOTO_B = 1;
    private static final int ACTION_TAKE_PHOTO_S = 2;
    private static final int ACTION_TAKE_VIDEO = 3;

    private static final String BITMAP_STORAGE_KEY = "viewbitmap";
    private static final String IMAGEVIEW_VISIBILITY_STORAGE_KEY = "imageviewvisibility";
    private static final String VIDEO_STORAGE_KEY = "viewvideo";
    private static final String VIDEOVIEW_VISIBILITY_STORAGE_KEY = "videoviewvisibility";

    private String mCurrentPhotoPath;
    private ImageView mImageView;
    private Bitmap mImageBitmap;
    private VideoView mVideoView;
    private Uri mVideoUri;
```

```
 /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mImageView = (ImageView) findViewById(R.id.imageView1);
        mVideoView = (VideoView) findViewById(R.id.videoView1);
        mImageBitmap = null;
        mVideoUri = null;

        Button picBtn = (Button) findViewById(R.id.btnIntend);
        picBtn.setOnClickListener(mTakePicOnClickListener);
}
 Button.OnClickListener mTakePicOnClickListener =
            new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                    dispatchTakePictureBIntent();
                }
            };
```

The Android way of delegating actions to other applications is to invoke an Intent that describes what you want done. Three components: the Intent itself, a call to start the external Activity, some code to handle the image data when focus returns to your activity.

```
private void dispatchTakePictureBIntent() {

    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "IMG_" + timeStamp + ".jpg";
    File albumF = getAlbumDir();
    File f = new File(albumF, imageFileName);
    mCurrentPhotoPath = f.getAbsolutePath();
    Uri contentUri = FileProvider.getUriForFile(
            this,
            "com.example.photo.provider",
            f);
    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, contentUri);

    startActivityForResult(takePictureIntent, ACTION_TAKE_PHOTO_B);
  }
```

private void dispatchTakePictureSIntent() { => only have the yellow part, do not have to save as file
private void dispatchTakeVideoIntent() {

```
 private File getAlbumDir() {
        File storageDir = null;
        if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {
            storageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
“CameraSample”);

            if (storageDir != null) {
                if (! storageDir.mkdirs()) {
                    if (! storageDir.exists()){
                        Log.d("CameraSample", "failed to create directory");
                        return null;
                    }
                }
            }
        } else {
            Log.v(getString(R.string.app_name), "External storage is not mounted READ/WRITE.");
        }

        return storageDir;
    }
```

The Android Camera application encodes the photo/video in the return Intent delivered to onActivityResult().

```
startActivityForResult(takePictureIntent, ACTION_TAKE_PHOTO_B);
```

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode) {
            case ACTION_TAKE_PHOTO_B: {
                if (resultCode == RESULT_OK) {
                    handleBigCameraPhoto();
                }
                break;
            }

              case ACTION_TAKE_PHOTO_S: {
              case ACTION_TAKE_VIDEO: {
              }
    }
```

```
//retrieves the image from path and displays it in an ImageView.
private void handleBigCameraPhoto() {
        if (mCurrentPhotoPath != null) {
            Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath);
            mImageView.setImageBitmap(bitmap);
            mVideoUri = null;
            mImageView.setVisibility(View.VISIBLE);
            mVideoView.setVisibility(View.INVISIBLE);
            mCurrentPhotoPath = null;
        }

    }
//The photo is encoded as a small Bitmap in the extras, under the key "data".
 private void handleSmallCameraPhoto(Intent intent) {
        Bundle extras = intent.getExtras();
        mImageBitmap = (Bitmap) extras.get("data");
        ....
    }
    private void handleCameraVideo(Intent intent) {
        mVideoUri = intent.getData();
        ....
        mVideoView.start(); => do not forget to start playing it
    }
```

```
 //to handle some display issue
@Override
   protected void onSaveInstanceState(Bundle outState) {
      outState.putParcelable(BITMAP_STORAGE_KEY, mImageBitmap);
      outState.putParcelable(VIDEO_STORAGE_KEY, mVideoUri);
      outState.putBoolean(IMAGEVIEW_VISIBILITY_STORAGE_KEY, (mImageBitmap != null) );
      outState.putBoolean(VIDEOVIEW_VISIBILITY_STORAGE_KEY, (mVideoUri != null) );
      super.onSaveInstanceState(outState);
   }

   @Override
   protected void onRestoreInstanceState(Bundle savedInstanceState) {
      super.onRestoreInstanceState(savedInstanceState);
      mImageBitmap = savedInstanceState.getParcelable(BITMAP_STORAGE_KEY);
      mVideoUri = savedInstanceState.getParcelable(VIDEO_STORAGE_KEY);
      mImageView.setImageBitmap(mImageBitmap);
      mImageView.setVisibility(
            savedInstanceState.getBoolean(IMAGEVIEW_VISIBILITY_STORAGE_KEY) ?
                  ImageView.VISIBLE : ImageView.INVISIBLE
      );
      mVideoView.setVideoURI(mVideoUri);
      mVideoView.setVisibility(
            savedInstanceState.getBoolean(VIDEOVIEW_VISIBILITY_STORAGE_KEY) ?
                  ImageView.VISIBLE : ImageView.INVISIBLE
      );
   }
```

Where is the gallery (of save big pic)?

On the emulator:
Files -> sdk_gphone -> picture -> CameraSample

On my phone:
Internal storage -> picture -> CameraSample

Run on your phone:
Settings -> Developer options -> Debugging -> USB debugging
If the app crashes -> App info -> Permissions -> Storage