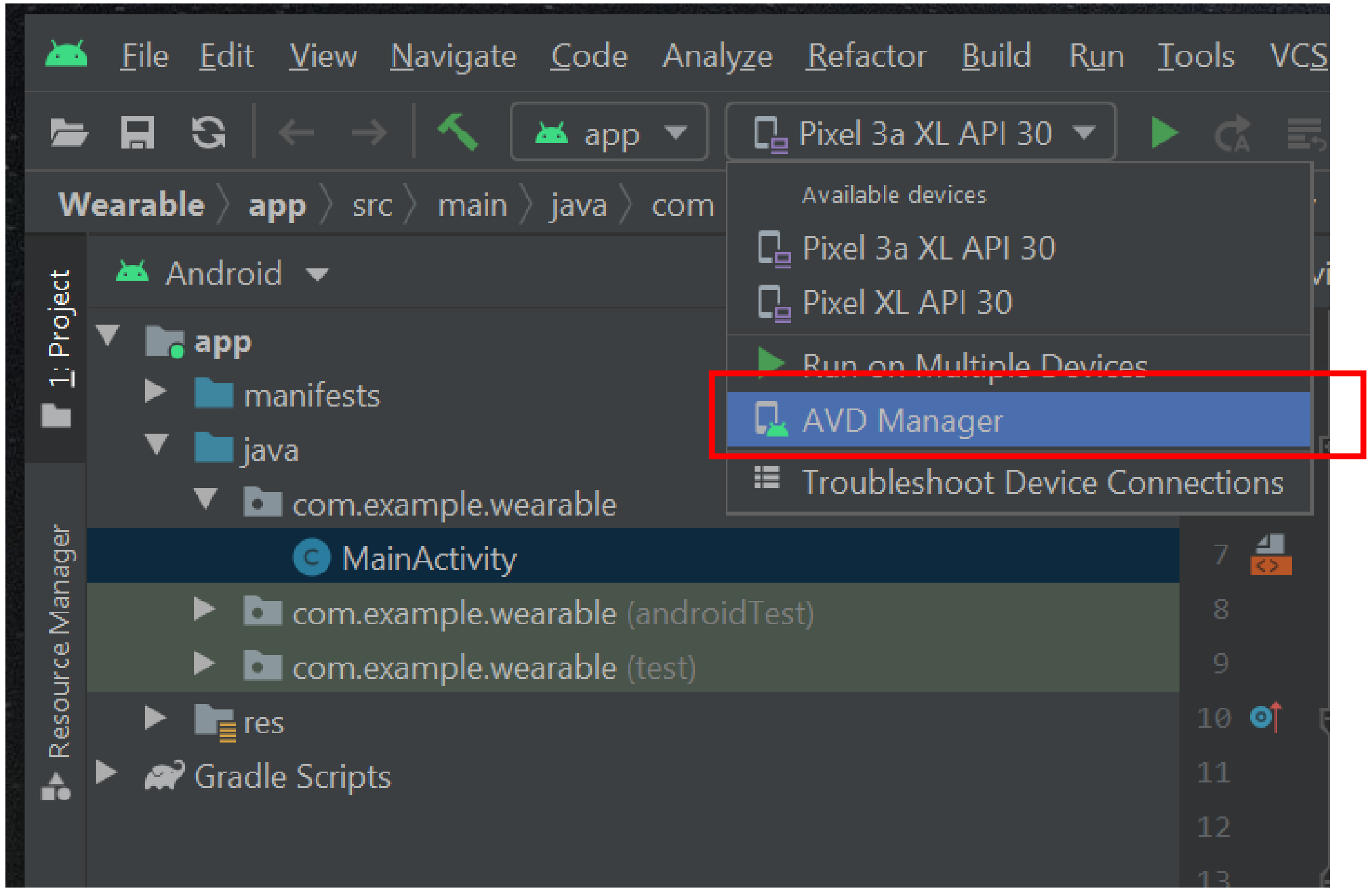
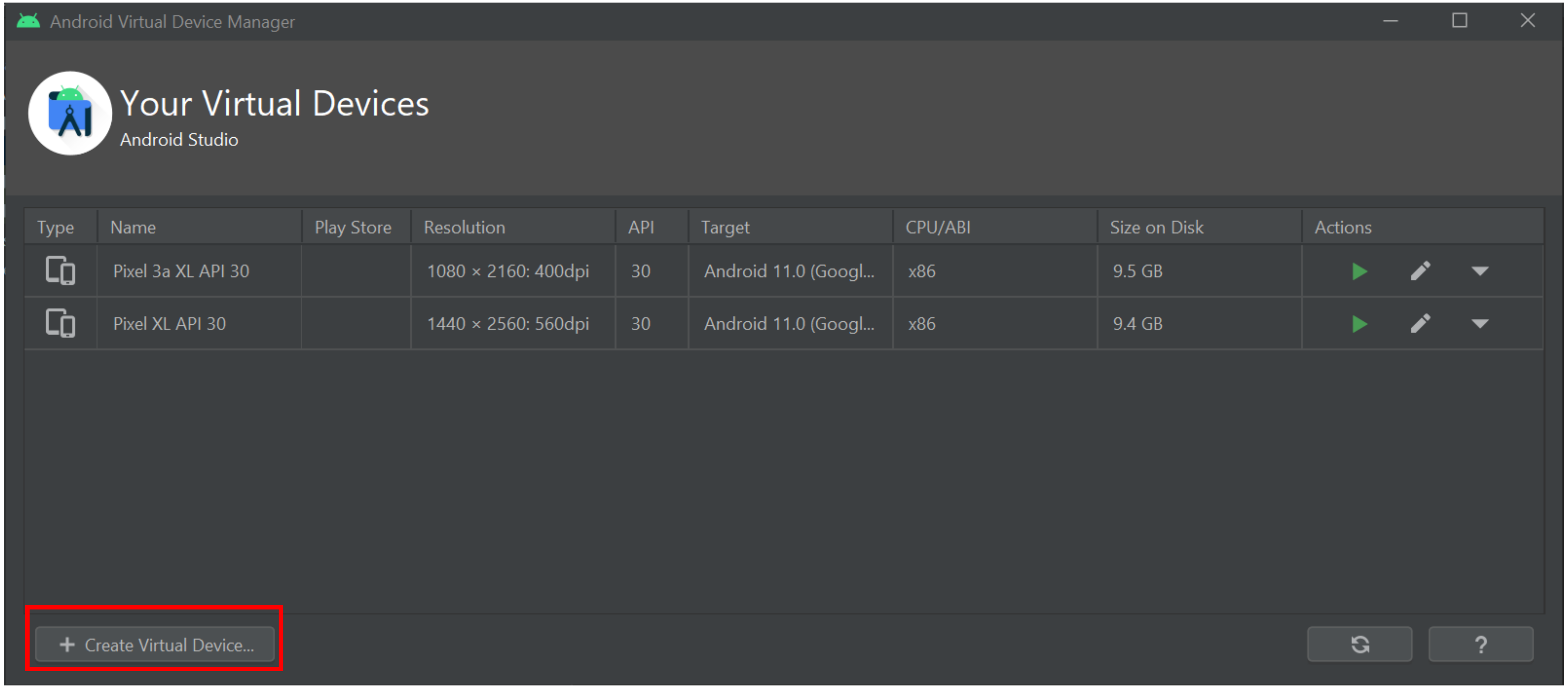
CSE 162 Mobile Computing

Lab 4 Android Wear Programming

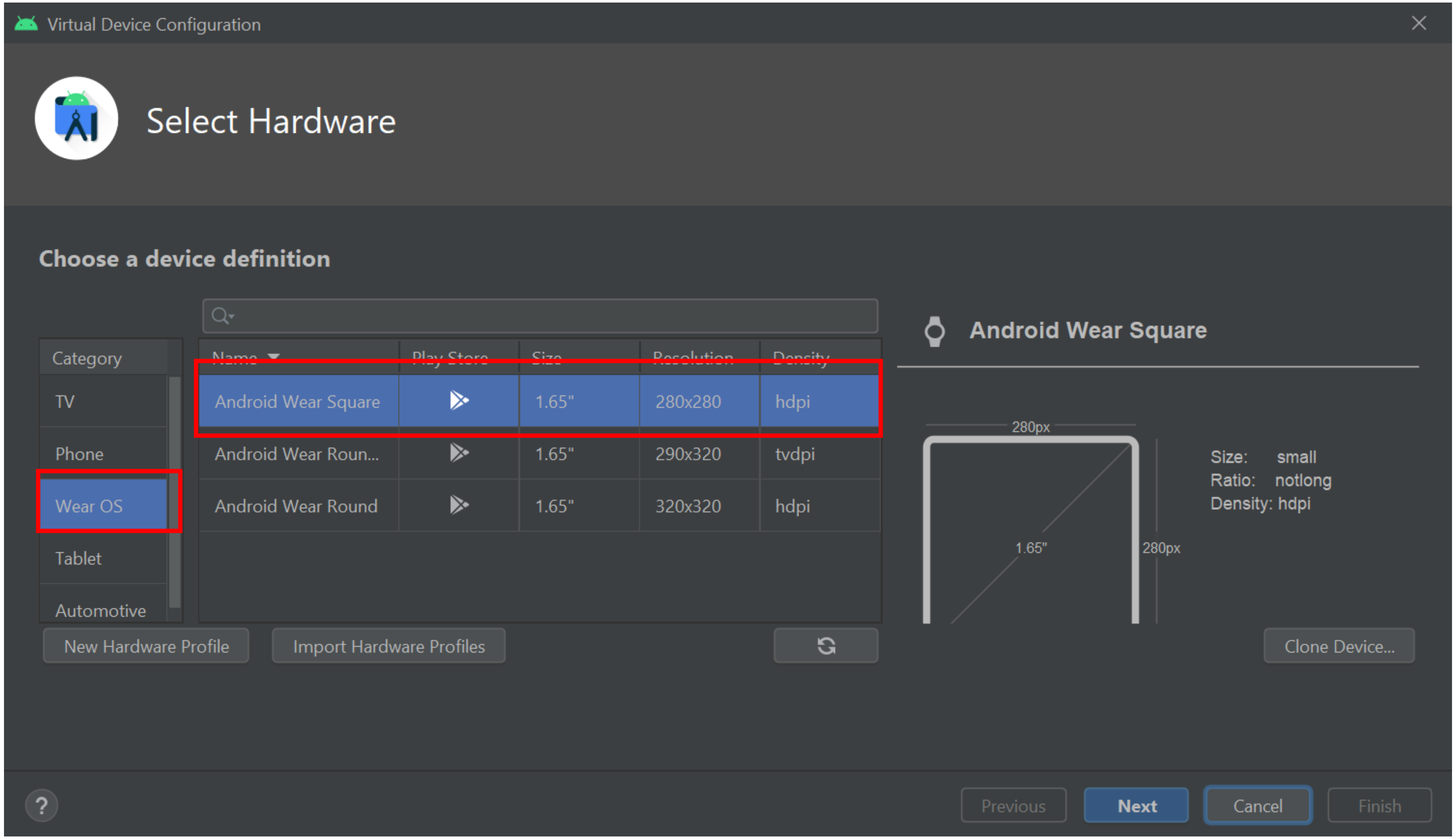
Hua Huang
Department of Computer Science and Engineering
University of California, Merced

Lecture: Mar 2/Mar 4
Demo: Mar 9/Mar 11
Deadline: Mar 16/Mar 18


File
Edit
View
Navigate
Code
Analyze
Refactor
Build
Run
Tools
VCS
app
Pixel 3a XL API 30
Wearable
app
src
main
java
com
Android
1: Project
Resource Manager
app
manifests
java
com.example.wearable
MainActivity
com.example.wearable (androidTest)
com.example.wearable (test)
res
Gradle Scripts
Available devices
Pixel 3a XL API 30
Pixel XL API 30
Run on Multiple Devices
AVD Manager
Troubleshoot Device Connections



Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 3a XL API 30		1080 × 2160: 400dpi	30	Android 11.0 (Googl...	x86	9.5 GB	
	Pixel XL API 30		1440 × 2560: 560dpi	30	Android 11.0 (Googl...	x86	9.4 GB	



Virtual Device Configuration

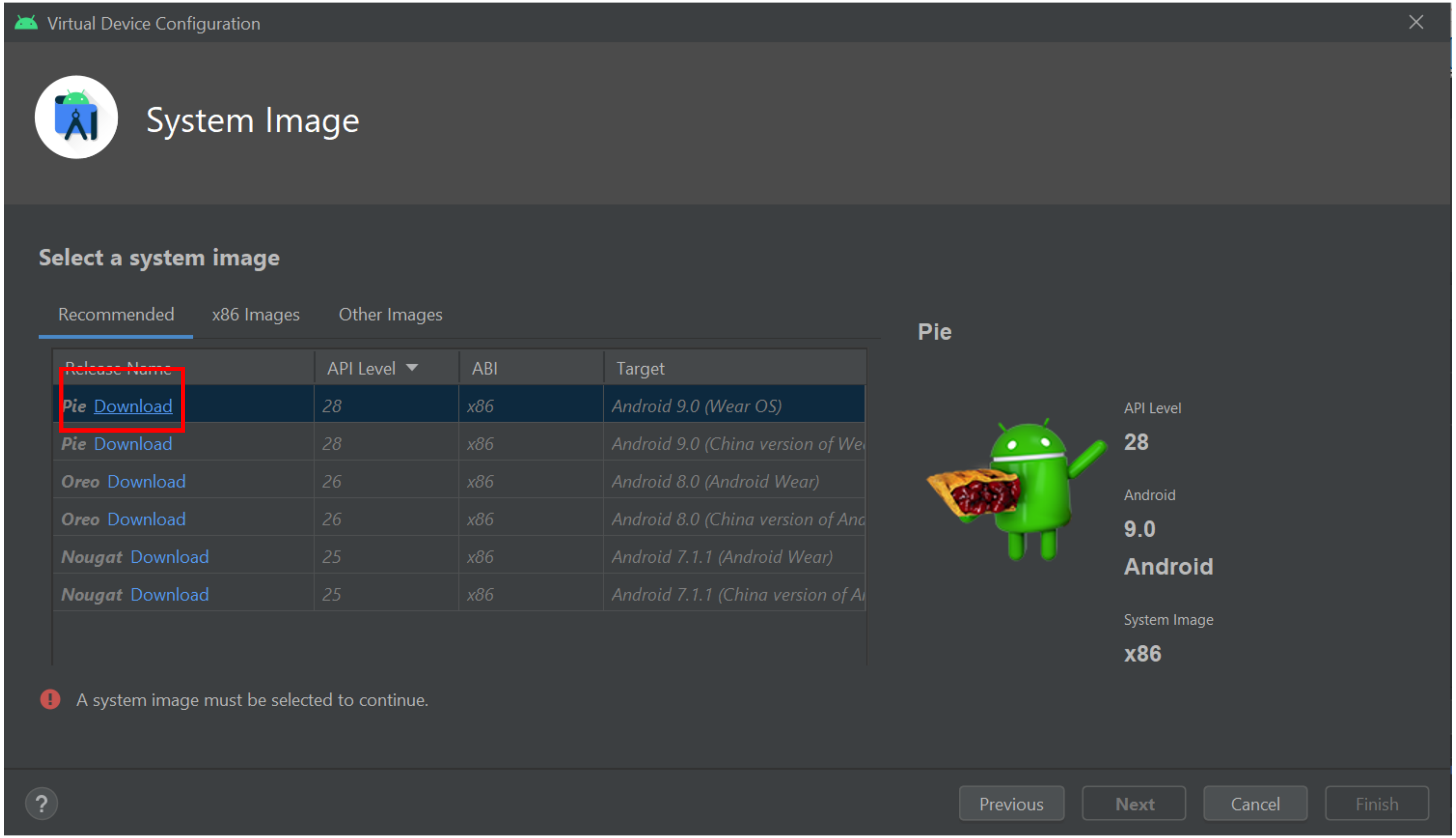
Select Hardware

Choose a device definition

Name	Play Store	Size	Resolution	Density
Android Wear Square	▶	1.65"	280x280	hdpi
Android Wear Roun...	▶	1.65"	290x320	tvdpi
Android Wear Round	▶	1.65"	320x320	hdpi

Android Wear Square

Size: small
Ratio: notlong
Density: hdpi


Virtual Device Configuration

System Image

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
Pie Download	*28*	*x86*	*Android 9.0 (Wear OS)*
Pie Download	*28*	*x86*	*Android 9.0 (China version of We*
Oreo Download	*26*	*x86*	*Android 8.0 (Android Wear)*
Oreo Download	*26*	*x86*	*Android 8.0 (China version of And*
Nougat Download	*25*	*x86*	*Android 7.1.1 (Android Wear)*
Nougat Download	*25*	*x86*	*Android 7.1.1 (China version of A*

Pie

API Level
28

Android
9.0
Android

System Image
x86

A system image must be selected to continue.

Our goal: Idleness monitor

- Build an app that can alert the user if they spends too much time without movements.
- Understand how to use wearable sensors, vibration notifications, and countdown timer.

Idleness monitor: basics (10pts)

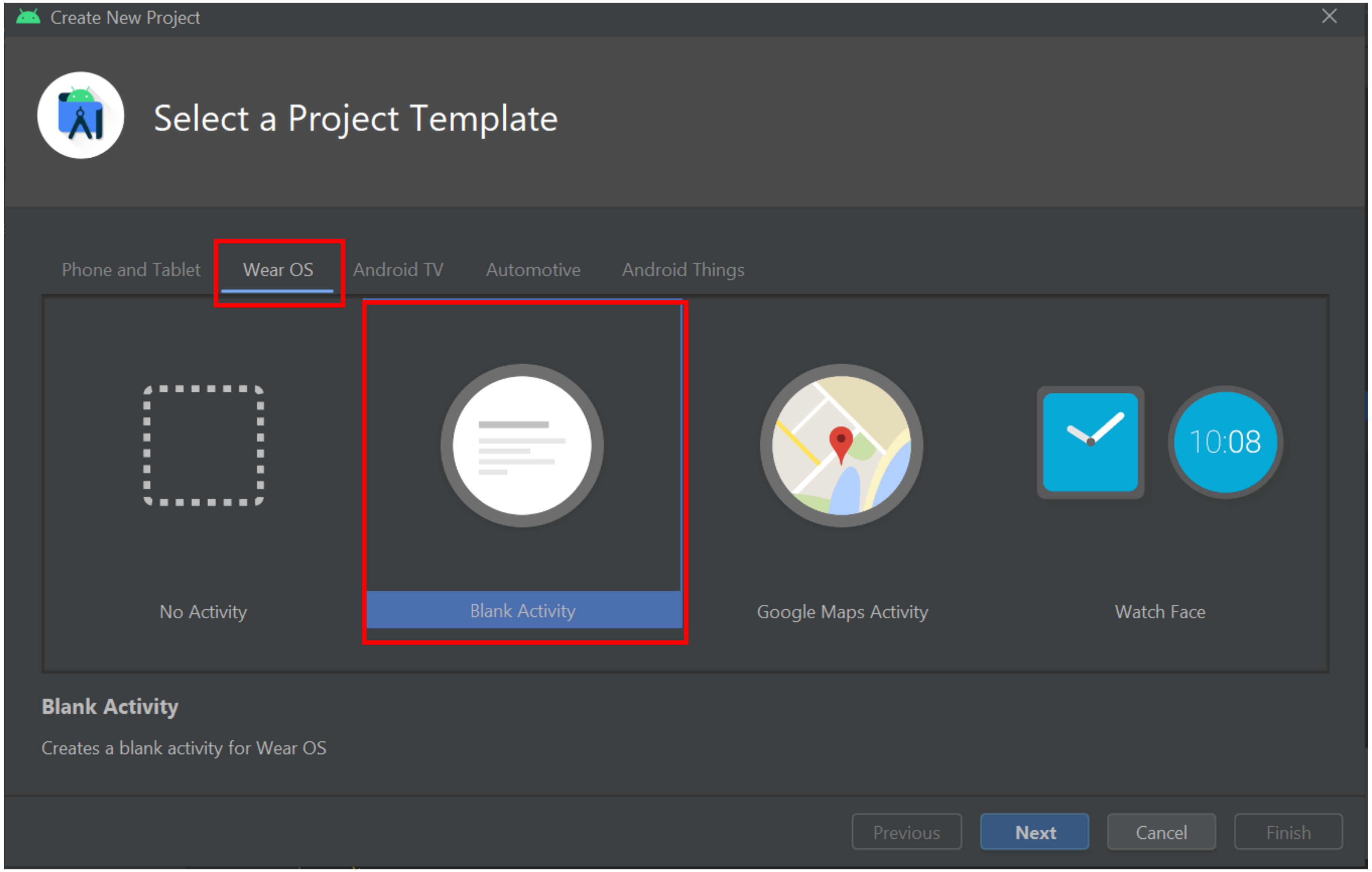
- The user clicks a button, then the watch monitors movements.
- Once a large movement is detected, a timer begins.
- After the timer finishes, vibrate and notify the user

Idleness monitor: extra credits (2pts)

- The user clicks a button, then the watch monitors movements.
- Once a large movement is detected, a timer begins.
- During the timer, everytime the user moves, then the timer is reset.
- After the timer finishes, vibrate and notify the user

- Tip for extra credits: cancel the countdown timer and then restart.

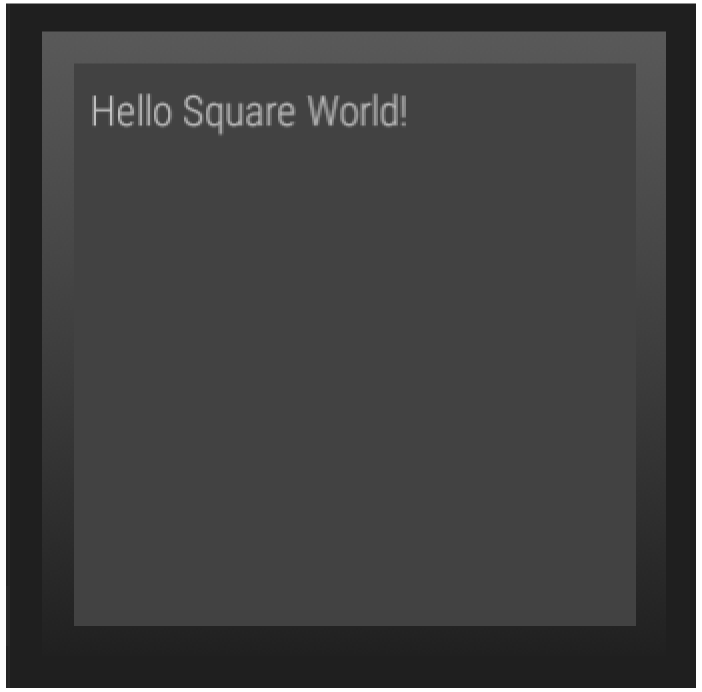
Create the project



Blank Activity

Creates a blank activity for Wear OS

Create the project


Hello Square World!

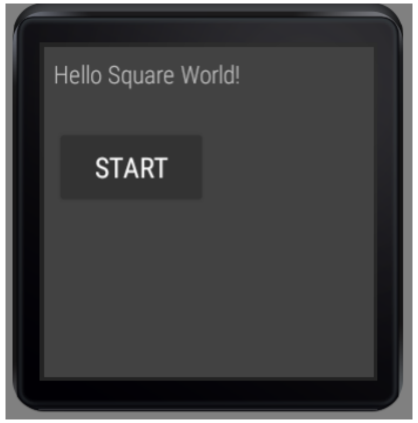
SensorEventListener

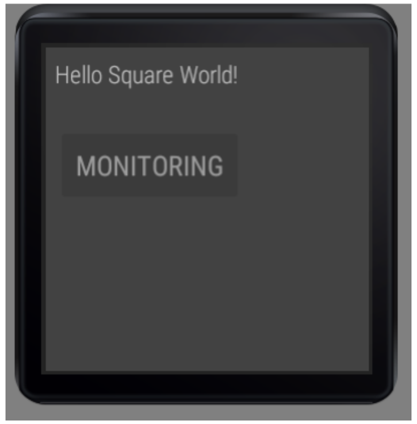
```
public class MainActivity extends WearableActivity implements SensorEventListener {

    private TextView mTextView;
    private TextView mCountDown;
    private Button mButton;

    private SensorManager mSensorManager;
    private Sensor mSensor;
```

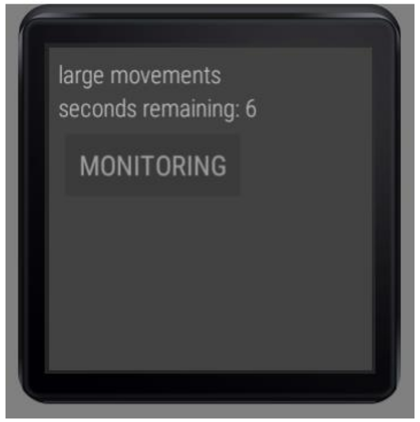
Layout and String

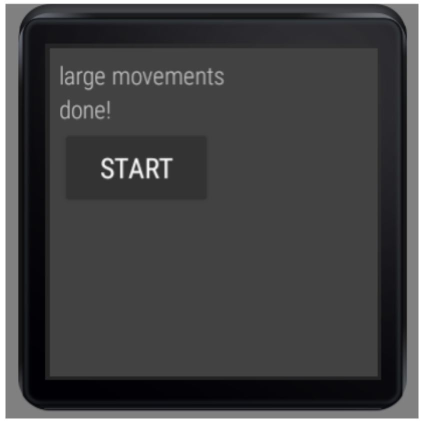
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
      android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world" />

    <TextView
      android:id="@+id/count"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/count" />


    <Button
      android:id="@+id/button"
      android:onClick="start_countdown"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/button"/>

 </LinearLayout>
```

```
<string name="hello_world">Hello Square World!</string>
  <string name="count"> </string>
  <string name="button">start</string>
```


Hello Square World!

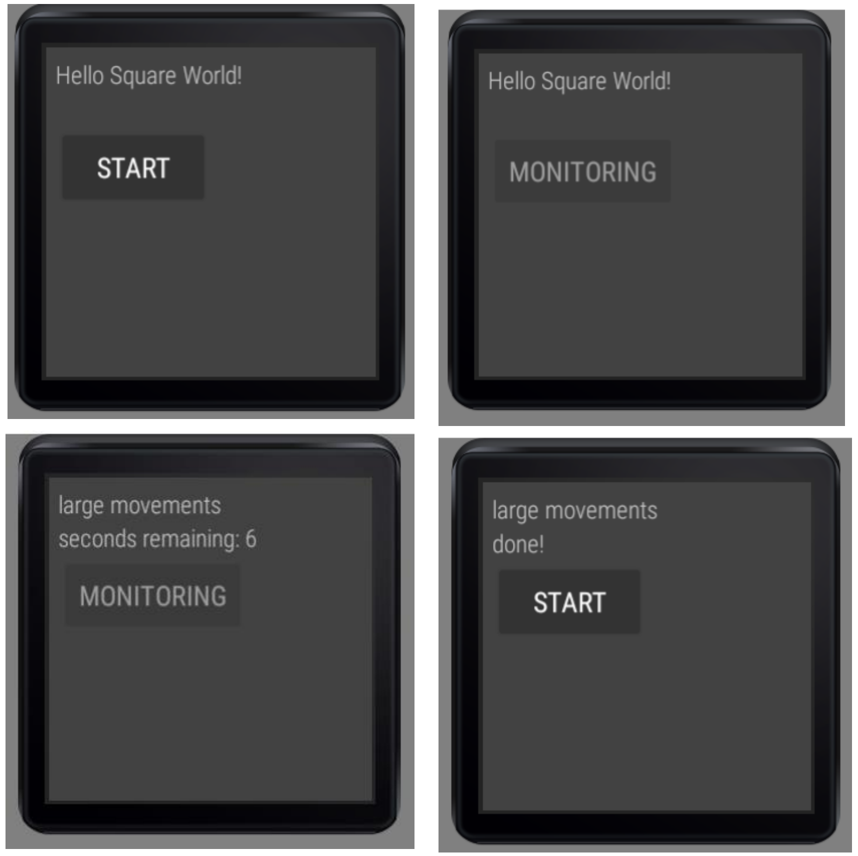

Hello Square World!


large movements
seconds remaining: 6


large movements
done!

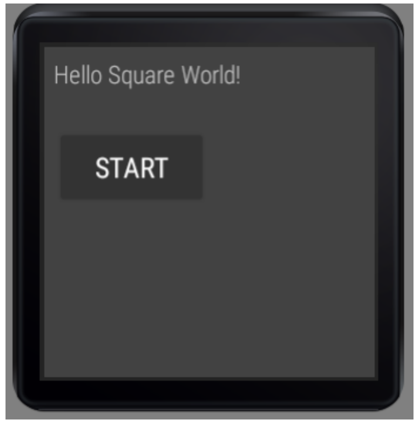
onCreate()

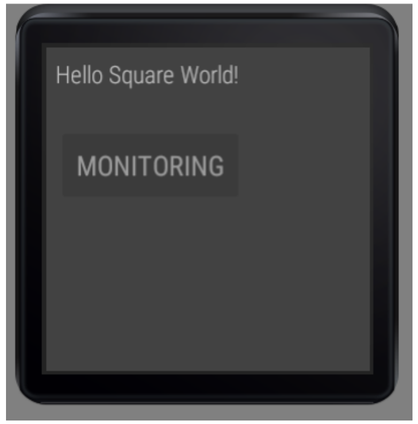
```
 //obtain the views, and initiate the sensors
@Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     mTextView = (TextView) findViewById(R.id.text);
     mCountDown = (TextView) findViewById(R.id.count);
     mButton = (Button) findViewById(R.id.button);

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

     // Enables Always-on
     setAmbientEnabled();
  }
```



Functions of the button

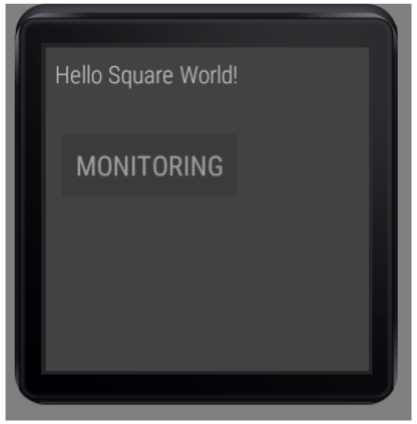
```
// change the texts, make it unclickable, and begin sensor data monitoring

public void start_countdown(View view){

      Log.d("TAG","Entered function");
      mButton.setText("Monitoring");
      mButton.setEnabled(false);
      mSensorManager.registerListener(this, mSensor, 20);

  }
```

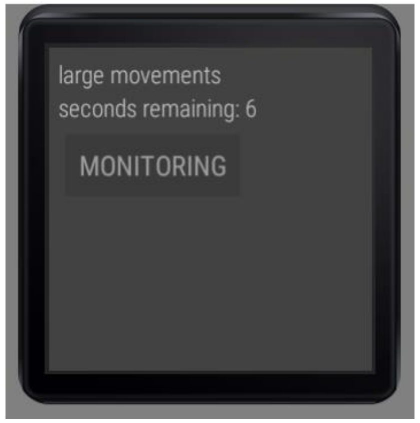




onSensorChanged()

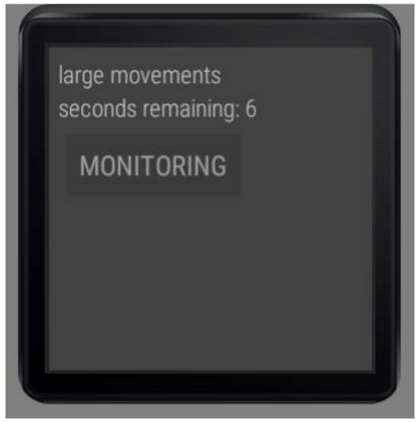
```
@Override
  public void onSensorChanged(SensorEvent event) {
     float maxValue=1;
     //When sensor data changes, check if the motion is greater than a threshold
     if(Math.abs(event.values[0]) + Math.abs( event.values[1]) +  Math.abs( event.values[2]) > maxValue) {
        maxValue = event.values[0] + event.values[1] + event.values[2];

        // display the texts
        mTextView.setText("large movements");

        //obtain the permission of using vibration
        Vibrator vibrator = (Vibrator) getSystemService(VIBRATOR_SERVICE);
        long[] vibrationPattern = {0, 500, 50, 300};
        //-1 - don't repeat
        final int indexInPatternToRepeat = -1;
        // vibrates the watch
        vibrator.vibrate(vibrationPattern, indexInPatternToRepeat);

        mSensorManager.unregisterListener(this);
```

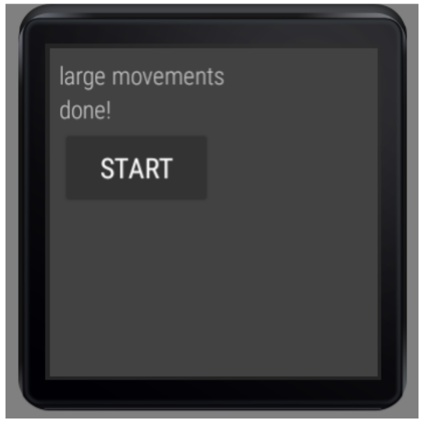

Hello Square World!


large movements
seconds remaining: 6

CountDownTimer

```
new CountDownTimer(10000, 1000) { //10 seconds in total, update the display every second
        public void onTick(long millisUntilFinished) {
          Log.d("TAG","TICK");
          mCountDown.setText("seconds remaining: " + millisUntilFinished / 1000);
        }

        //update the texts and enable the button again after the time is finished.
        public void onFinish() {
          mCountDown.setText("done!");
          mButton.setText("Start");
          mButton.setEnabled(true);

          Vibrator vibrator = (Vibrator) getSystemService(VIBRATOR_SERVICE);
          long[] vibrationPattern = {0, 500, 50, 300};
          //-1 - don't repeat
          final int indexInPatternToRepeat = -1;
          vibrator.vibrate(vibrationPattern, indexInPatternToRepeat);
        }
      }.start();
```


large movements
seconds remaining: 6


large movements
done!

onResume() / onPause()

```
@Override
  protected void onResume() {
     super.onResume();
     mSensorManager.registerListener(this, mSensor, 2000);

  }

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

@Override
  public void onAccuracyChanged(Sensor sensor, int accuracy) {

  }
```