# CS160: Section 4
# Heuristic Evaluations & Wear Sensors

Sept 18, 2015

# Android Wear: In summary

- Download the Wear Emulator
- Get familiar with the Wear Emulator
- Get Genymotion for Android 5.1.0
- Install Google Apps and Wear APKs to the Genymotion Emulator
- Start Wear app and connect to Wear emulator
- Open up a gateway via the command line

Create New Project

# Target Android Devices

## Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet

Minimum SDK: API 21: Android 5.0 (Lollipop)

Lower API levels target more devices, but have fewer features available. By targeting API 21 and later, your app will run on approximately **9.7%** of the devices that are active on the Google Play Store.

Help me choose

☑ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

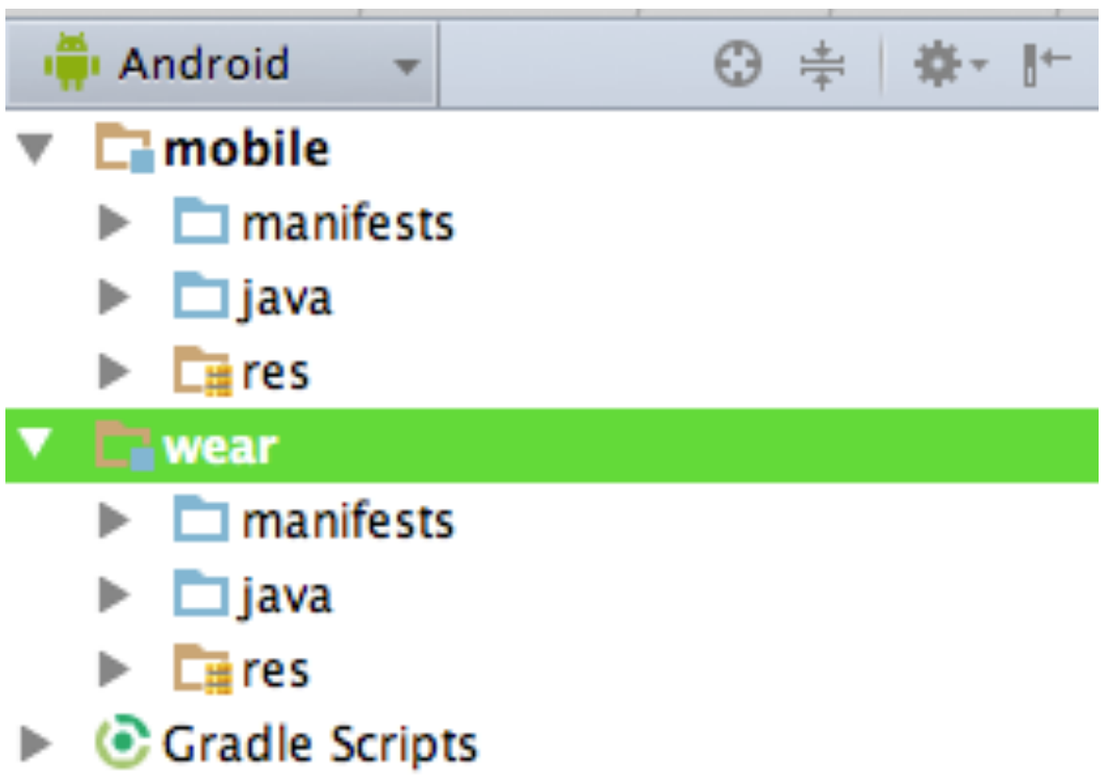
☐ Android Auto

☐ Glass

Minimum SDK: MNC: Android M (Preview)

Cancel | Previous | Next | Finish

# Your watch has its own folders




- Don’t mix up mobile and wear folders!
- Now have 2 separate .APKs as well

# Wear notifications

- Create an intent for the notification

```
int notificationId = 001;

// Build intent for notification content
Intent convertIntent = new Intent(this,
ConversionActivity.class);
viewIntent.putExtra(“Pig”, animal_name);
PendingIntent convertPendingIntent =
        PendingIntent.getActivity(this, 0,
convertIntent, 0);
```

the new activity/ service to launch (if any)

request code

flags

https://developer.android.com/training/wearables/notifications/creating.html

# Wear notifications

- Create the notification itself

```
NotificationCompat.Builder notificationBuilder =
    new NotificationCompat.Builder(MainActivity.this)
      .setSmallIcon(R.drawable.ic_launcher)
       .setContentTitle("Title")
.setContentText("Android
      Wear Notification");
```

# Wear notifications

- Actually send the notification

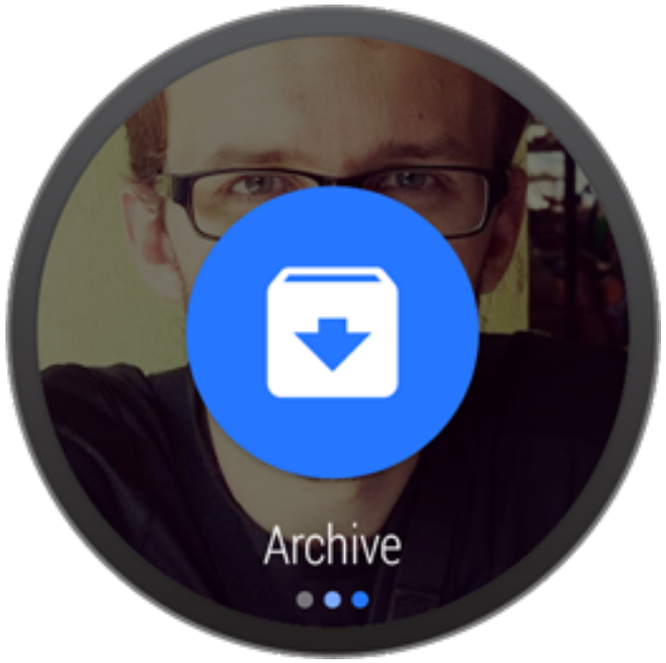
```
// Get an instance of the NotificationManager service
NotificationManagerCompat notificationManager =
    NotificationManagerCompat.from(this);

// Build the notification and issue with manager.
notificationManager.notify(notificationId,
                           notificationBuilder.build());
```

.notify is when the notification
actually pops up on the screen!

https://developer.android.com/training/wearables/notifications/creating.html

# Wear notifications




- Add extra “action buttons” by passing a PendingIntent into addAction()

```
PendingIntent archivePendingIntent =
PendingIntent.getActivity(this, 0, archiveIntent, 0);
```

```
NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_event)
        .setContentTitle(eventTitle)
        .setContentText(eventLocation)
        .setContentIntent(viewPendingIntent)
        .addAction(R.drawable.archive_button,
                   getString(R.string.archive),
                   archivePendingIntent);
```

https://developer.android.com/training/wearables/notifications/creating.html

# Moto 360 Sensors

- Pedometer
- Accelerometer, gyroscope
- Optical heart rate monitor
- Ambient light sensor
- Dual microphone

Take 1 minute and brainstorm with a partner the most wacky app that uses one of these sensors!

# Sensors

- Each sensor needs a manager

```
private SensorManager mSensorManager;
private Sensor mSensor;
   ...
mSensorManager = (SensorManager)
        getSystemService(
              Context.SENSOR_SERVICE);
mSensor =
        mSensorManager.getDefaultSensor
            (Sensor.TYPE_ACCELEROMETER);
```

| Sensor |
| --- |
| TYPE_ACCELEROMETER |
| TYPE_AMBIENT_TEMPERATURE |
| TYPE_GRAVITY |
| TYPE_GYROSCOPE |
| TYPE_LIGHT |
| TYPE_LINEAR_ACCELERATION |
| TYPE_MAGNETIC_FIELD |
| TYPE_ORIENTATION |
| TYPE_PRESSURE |
| TYPE_PROXIMITY |
| TYPE_RELATIVE_HUMIDITY |
| TYPE_ROTATION_VECTOR |
| TYPE_TEMPERATURE |

http://developer.android.com/guide/topics/sensors/sensors_overview.html

# Sensor Registration

- Have your class “implement SensorEventListener”

```
public class SensorActivity extends Activity
implements SensorEventListener {
  @Override
   protected void onResume() {
     super.onResume();
     mSensorManager.registerListener(this,
   mSensor, SensorManager.SENSOR_DELAY_NORMAL);
  }
}
```

Always listen in for
sensor values changing

http://developer.android.com/guide/topics/sensors/sensors_overview.html

# Sensor Registration

- Do something when the values change

```
@Override
public final void
onSensorChanged(SensorEvent event)
{
    // Use values from event.values array
 }
```

http://developer.android.com/guide/topics/sensors/sensors_overview.html

# Emulating Sensors

- Do it from the command line

```
1.telnet localhost <watch port>
  (usually 5556, adb devices to check)
2.sensor set acceleration <number>
```

That’s it! Update acceleration values with “sensor set acceleration <number>” as many times as you like.

http://developer.android.com/guide/topics/sensors/sensors_overview.html