

CS160: Section 3

Intro to Android Wear

Sept 11, 2015

Agenda

- Affordances (8m)
- Wear design guidelines (10m)
- Activities, Services, Threads (12m)
- Setting up the wear emulator (17m)

Admin

- **Programming 1 due at midnight!**
- Get your phones by 9/14 - 5.0 Lollipop
- Reading Response 3: Due Thurs 9/17
- Group petition: Due Fri 9/18

Affordances

- “Perceived and actual properties of the thing” - Don Norman
- How you use an object
 - Ex: handles afford pulling, glass affords breaking, balls afford bouncing
- Signifiers: physical object itself
 - Ex: the flat bottom of the chair is signifier which affords sitting

Look around you: affordances

Android Wear

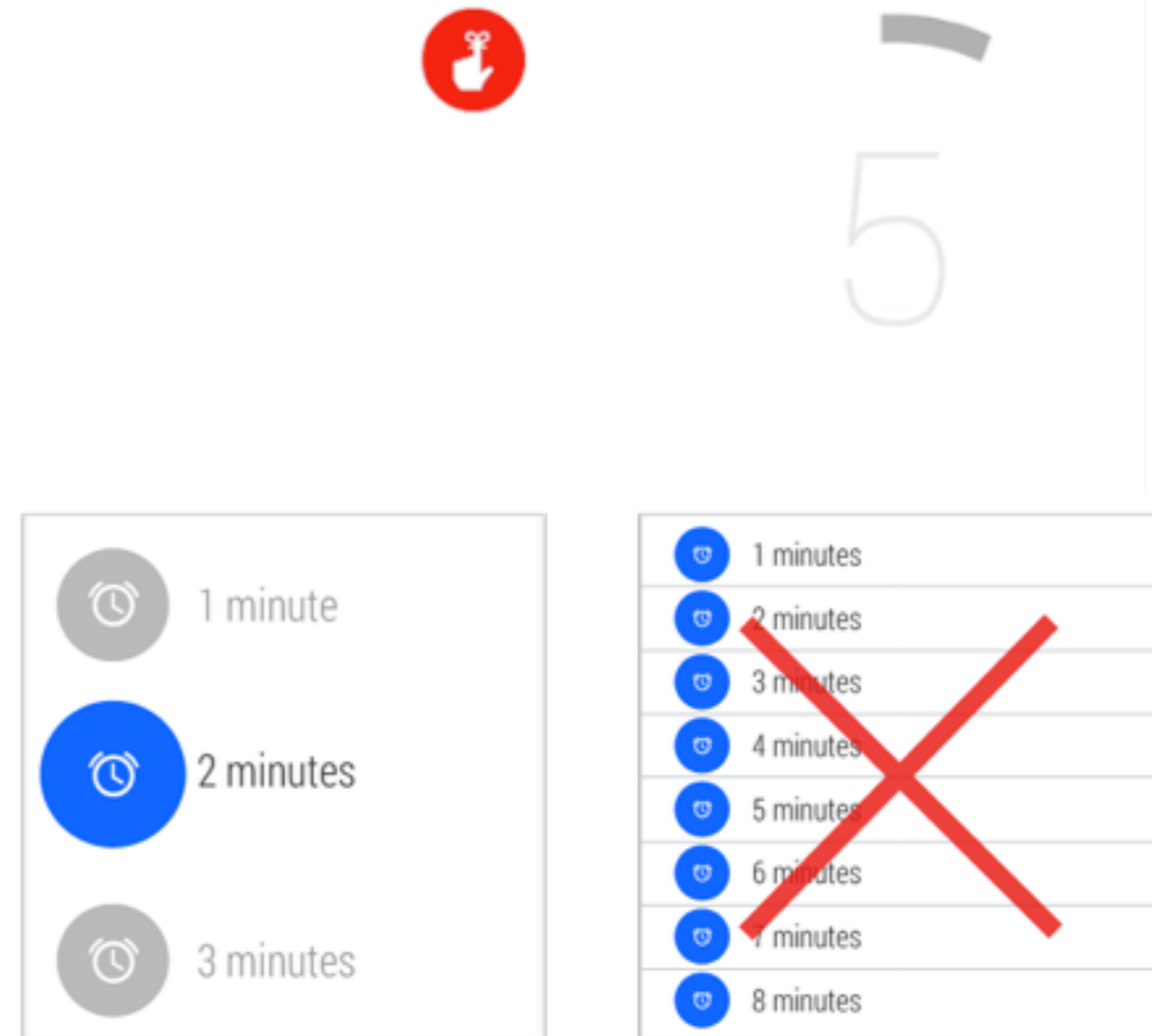
Information that moves with you.

Small, powerful devices, worn on the body. Useful information when you need it most. Intelligent answers to spoken questions. Tools to help reach fitness goals. Your key to a multiscreen world.



Design Principles for Wear

- Focus on not stopping the user and all else will follow
 - 5 second interactions
- Design for big gestures
 - No more than 3 items

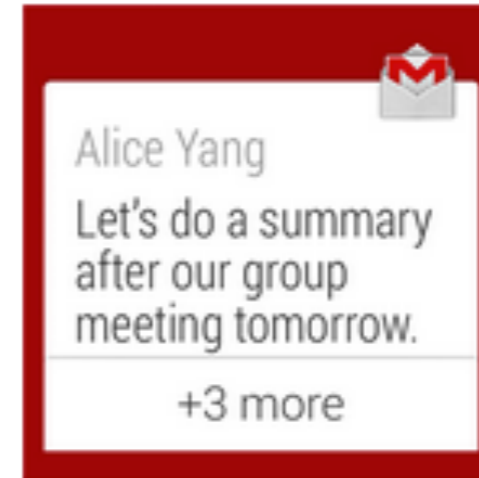
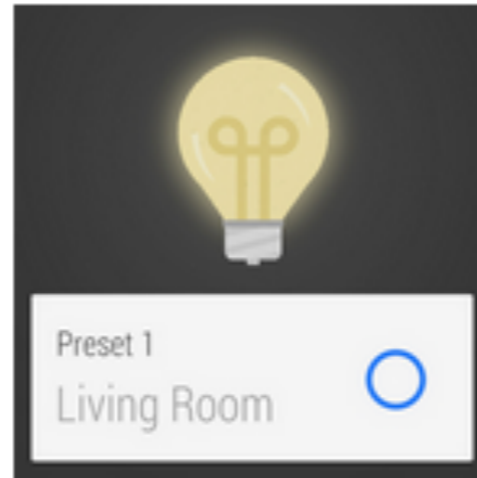


Design Principles for Wear

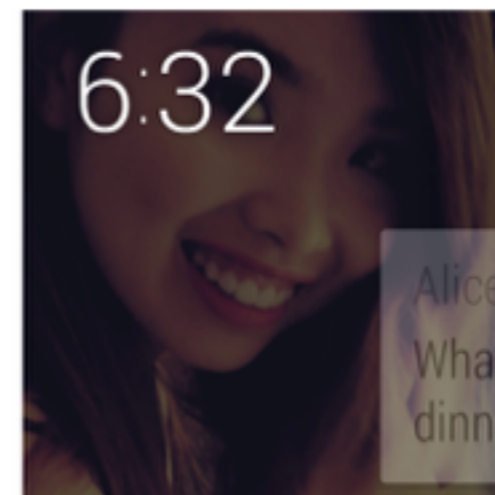
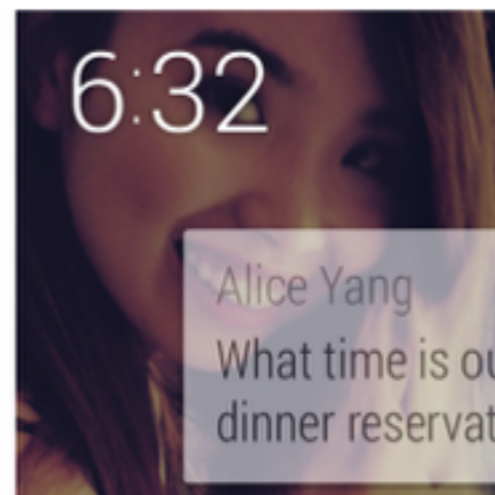
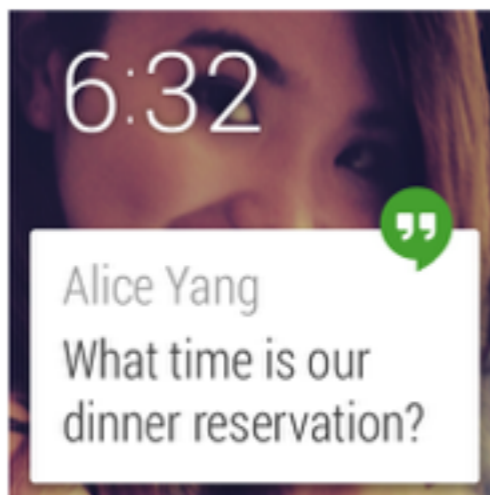
- Do one thing, really fast
- Design for the corner of the eye
- Don't be a constant shoulder tapper

UI Patterns for Wear

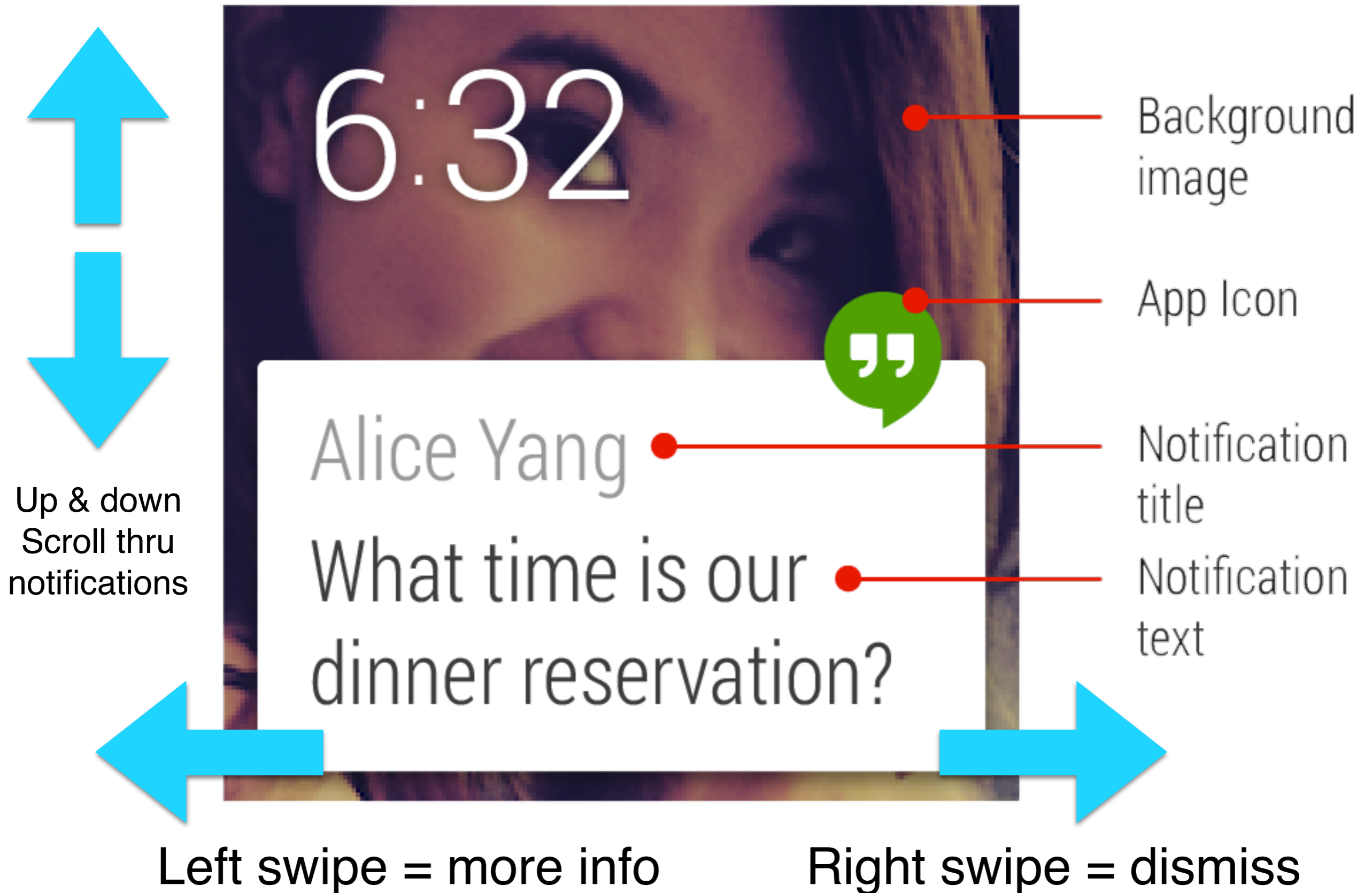
- Cards



- Notifications

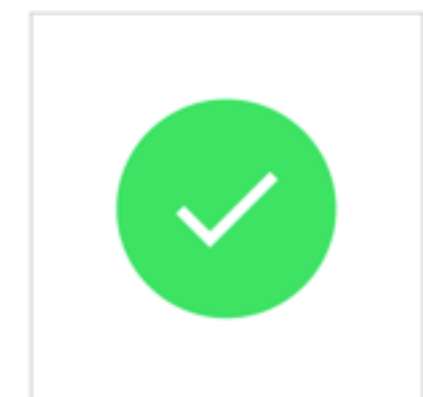
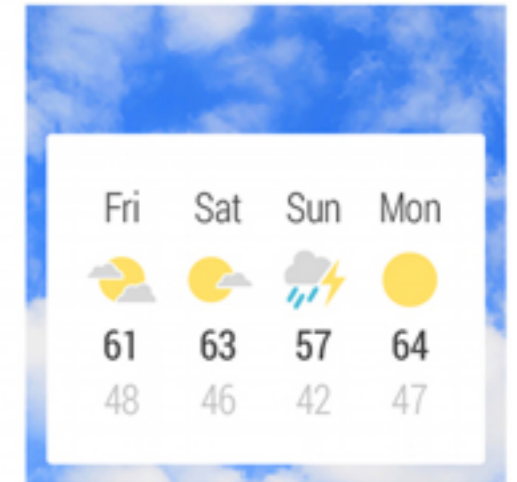
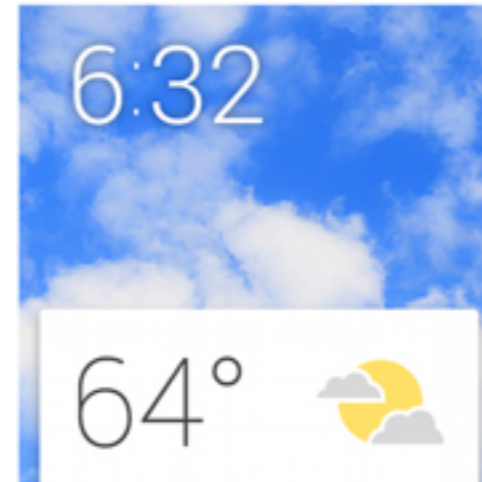


The Notification



More UI Guidelines

- Separate info into chunks
- Use clear, bold typography
- Keep notifications to a minimum
- Use consistent branding and color
- Omit needless text: Less is more
- User feedback: confirmation animations



You're shopping for groceries at Berkeley Bowl. We're in the future with great GPS technologies that know where grocery items are in the store. Storyboard a user flow for a watch only app to make grocery shopping fast and convenient.

Activities and Services

Organizing your App's Code

Review Related Terms

What does it mean for a process to run in the **foreground**?

When it runs in the **background**?

Brainstorm examples of each with a partner.

Take 2 minutes for this.

Let's Review Your Answers

:30

15

Defining *Activity*

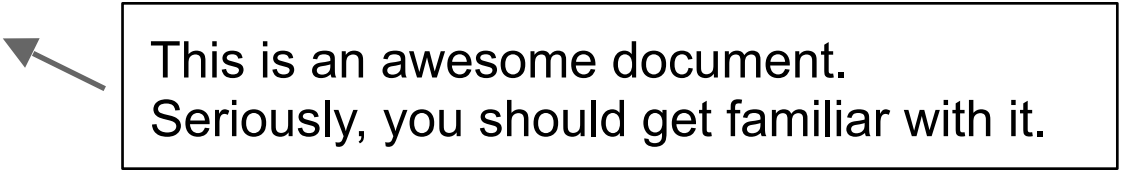
- An *Activity* is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.
- Each activity is given a window in which to draw its user interface.

From *Android Developers Guide*

Defining *Service*

- A *Service* is an application component that can perform long-running operations in the background and does not provide a user interface
- A component can bind to a service to interact with it and even perform interprocess communication (IPC)

From *Android Developers Guide*



This is an awesome document.
Seriously, you should get familiar with it.

Choosing Activities vs. Services

- When programming Android, most of the Java files you make will be either an activity or a service
- Make an activity if:
 - The user needs to see it
- Make a service if:
 - it's mechanical work unrelated to a View *or*
 - other applications have want to access it *or*
 - it will block the UI thread from running

Your Turn

Do I implement an activity or a service when I want to...

1. Make a form for entering medical information and submitting to a web service
2. Perform some numeric computation and save the results to SQL
3. Decrypt a message and launch a notification when finished
4. Implement a media player to show movies
5. Poll temperature and wake up the device when the temperature drops below 32

Pair up, justify your reasoning, and discuss for *3 minutes*.

Your Turn

Do I implement an activity or a service when I want to...

1. Make a form for entering medical information and submitting to a web service
2. Perform some numeric computation and save the results to SQL
3. Decrypt a message and launch a notification when finished
4. Implement a media player to show movies
5. Poll temperature and wake up the device when the temperature drops below 32

Let's Review

:40

How is this Related to Foreground and Background?

Let's get 2 people's thoughts on this.

How is this Related to Foreground and Background?

In general terms, activities are what is in the “foreground” for both users and the program execution -- the user can see it, and it runs on the main UI thread.

Services are more like “background” work, and can run on threads that are not the main UI thread.

Implementing a Service

```
public class MyService extends Service {  
  
    @Override  
    public int onStartCommand(Intent intent, int flags,  
        int startId) {  
        // Kick off new work to do  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        // Return a binder to this service  
    }  
  
}
```

Implementing a Service

```
public class MyService extends Service {
```

```
    @Override
```

All Services extend the base class “Service”

```
    public int onStartCommand(Intent intent, int flags,
        int startId) {
        // Kick off new work to do
    }
```

```
    @Override
```

```
    public IBinder onBind(Intent intent) {
        // Return a binder to this service
    }
```

```
}
```


Implementing a Service

```
public class MyService extends Service {  
  
    @Override  
    public int onStartCommand(Intent intent, int flags,  
        int startId) {  
        // Kick off new work to do  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        // Return a binder to this service  
    }  
  
}
```

onStartCommand gets called when you start a service with the `startService()` method of an activity or service

Implementing a Service

```
public class MyService extends Service {  
  
    @Override  
    public int onStartCommand(Intent intent, int flags,  
        int startId) {  
        // Kick off new work to do  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        // Return a binder to this service  
    }  
}
```

You can also 'bind' to a service from another application. Basically, this method let's us call methods on this service from *other applications*.

26

But let's ignore it for now.

You need to override it, but you can just return *null* for the time being.

Implementing a Service

```
public class MyService extends Service {  
  
    @Override  
    public int onStartCommand(Intent intent, int flags,  
        int startId) {  
        // Kick off new work to do  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        return null; // :D  
    }  
}
```

You can also 'bind' to a service from another application. Basically, this method let's us call methods on this service from *other applications*.

27

But let's ignore it for now.

You need to override it, but you can just return *null* for the time being.

Let's Write a Service Together

A service that adds together 2 numbers and outputs them to the log using the *Log* utility.

Preparing and Running a Service

AndroidManifest.xml

```
<application ... >  
  <!-- ... -->  
  <service android:name=".MyService" />  
  <!-- ... -->  
</application>
```

You need to do add the service to the manifest before it can be called from anywhere else.

MainActivity.java

```
startService(new Intent(this, MyService.class));
```

Running in a new Thread

```
@Override
public int onStartCommand(Intent intent, int flags,
    int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                Log.i("tag", "Work");
            }
        }
    }).start();
    return START_STICKY;
}
```

Don't slow down the main thread. We have to explicitly create a new thread.

Running in a new Thread

```
@Override
public int onStartCommand(Intent intent, int flags,
    int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                Log.i("tag", "Work");
            }
        }
    }).start();
    return START_STICKY;
}
```

Don't slow down the main thread. We have to explicitly create a new thread.

A Runnable is an object that encapsulates a block of code to be run at a later time.

Running in a new Thread

```
@Override
public int onStartCommand(Intent intent, int flags,
    int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                Log.i("tag", "Work");
            }
        }
    }).start();
    return START_STICKY;
}
```

Don't slow down the main thread. We have to explicitly create a new thread.

A Runnable is an object that encapsulates a block of code to be run at a later time.

By making a Thread and calling its "start" method, we run this code in a new thread.

Running in a new Thread

```
@Override
public int onStartCommand(Intent intent, int flags,
    int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                Log.i("tag", "Work");
            }
        }
    }).start();
    return START_STICKY;
}
```

(This return code here means to restart the service when it fails.)

Don't slow down the main thread. We have to explicitly create a new thread.

A Runnable is an object that encapsulates a block of code to be run at a later time.

By making a Thread and calling its "start" method, we run this code in a new thread.

What Did We Have to Add to get a Service to Run in Its Own Thread?

There were 2 classes involved

Closing Remarks

Giving parameters to an Activity or Service

Know those `Intents` we keep making?

Look up `Intent.putExtra()`

Getting data back from activities

Look up `startActivityForResult()` and
`onActivityResult()`

Closing Remarks

Giving parameters to an Activity or Service

Know those `Intents` we keep making?

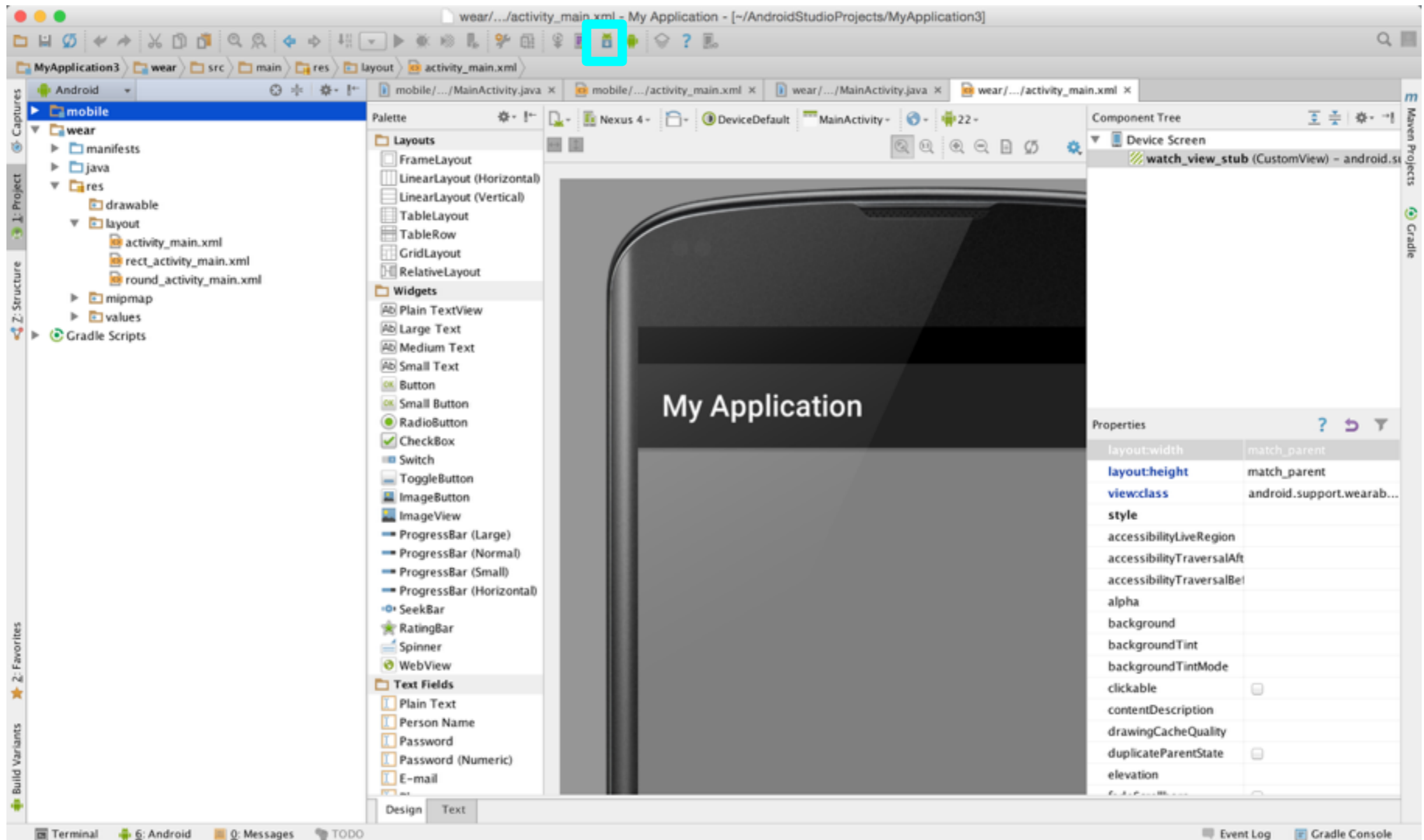
Look up `Intent.putExtra()`

Getting data back from activities

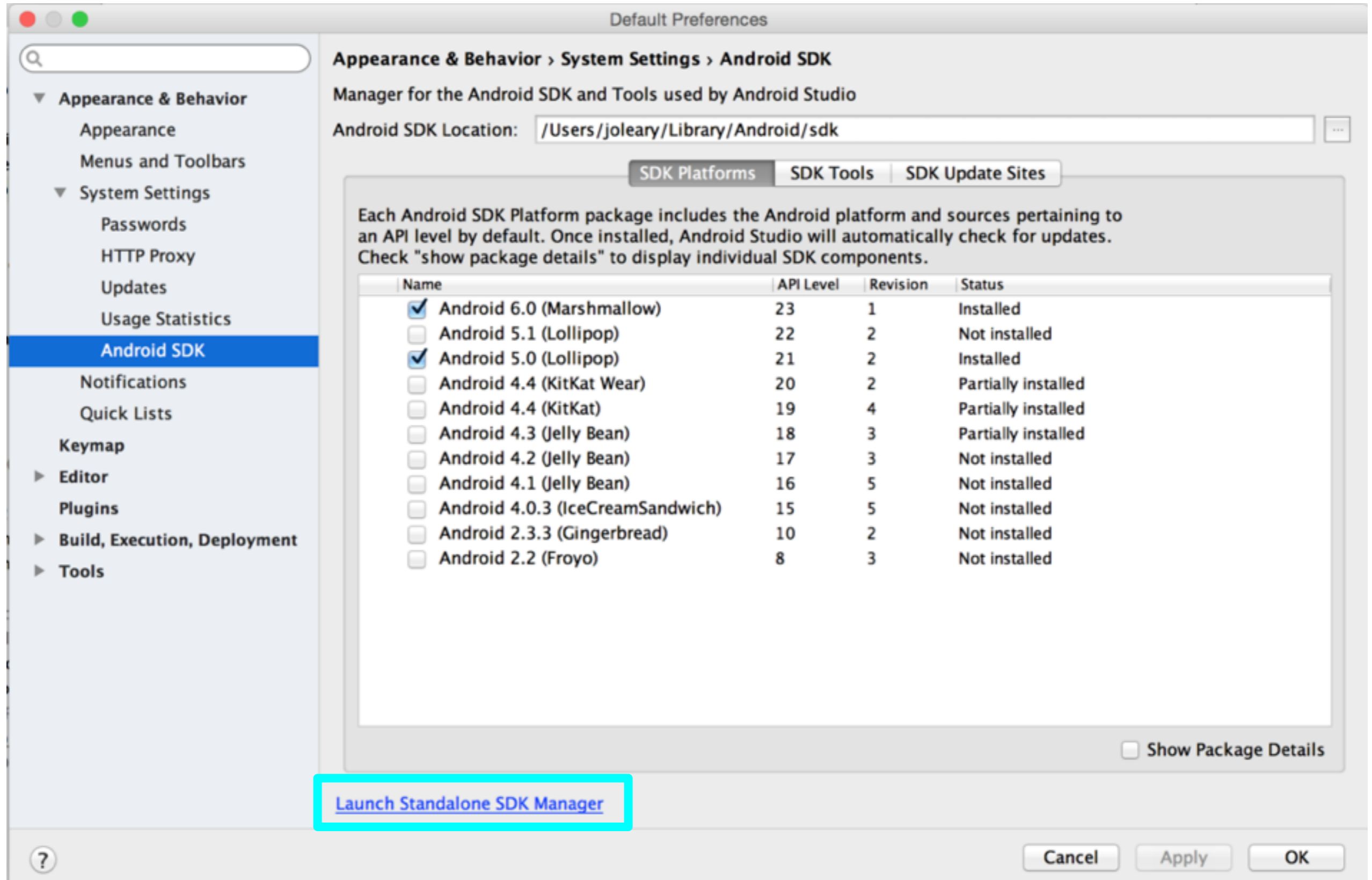
Look up `startActivityForResult()` and
`onActivityResult()`

Can a service listen for events?

Setting up the wear emulator



Setting up the wear emulator



















Setting up the wear emulator

Android SDK Manager

SDK Path: /Users/andrew/Library/Android/sdk

Packages

 Name	API	Rev.	Status
 Android 5.1.1 (API 22)			
<input type="checkbox"/>  Documentation for Android SDK	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  SDK Platform	22	2	<input checked="" type="checkbox"/> Installed
<input type="checkbox"/>  Samples for SDK	22	6	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android TV ARM EABI v7a System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android TV Intel x86 Atom System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android Wear ARM EABI v7a System Image	22	2	<input type="checkbox"/> Not installed
<input checked="" type="checkbox"/>  Android Wear Intel x86 Atom System Image	22	2	<input checked="" type="checkbox"/> Installed
<input type="checkbox"/>  ARM EABI v7a System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Intel x86 Atom_64 System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Intel x86 Atom System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google APIs	22	1	<input checked="" type="checkbox"/> Installed
<input type="checkbox"/>  Google APIs ARM EABI v7a System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google APIs Intel x86 Atom_64 System Image	22	1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Google APIs Intel x86 Atom System Image	22	1	<input checked="" type="checkbox"/> Installed

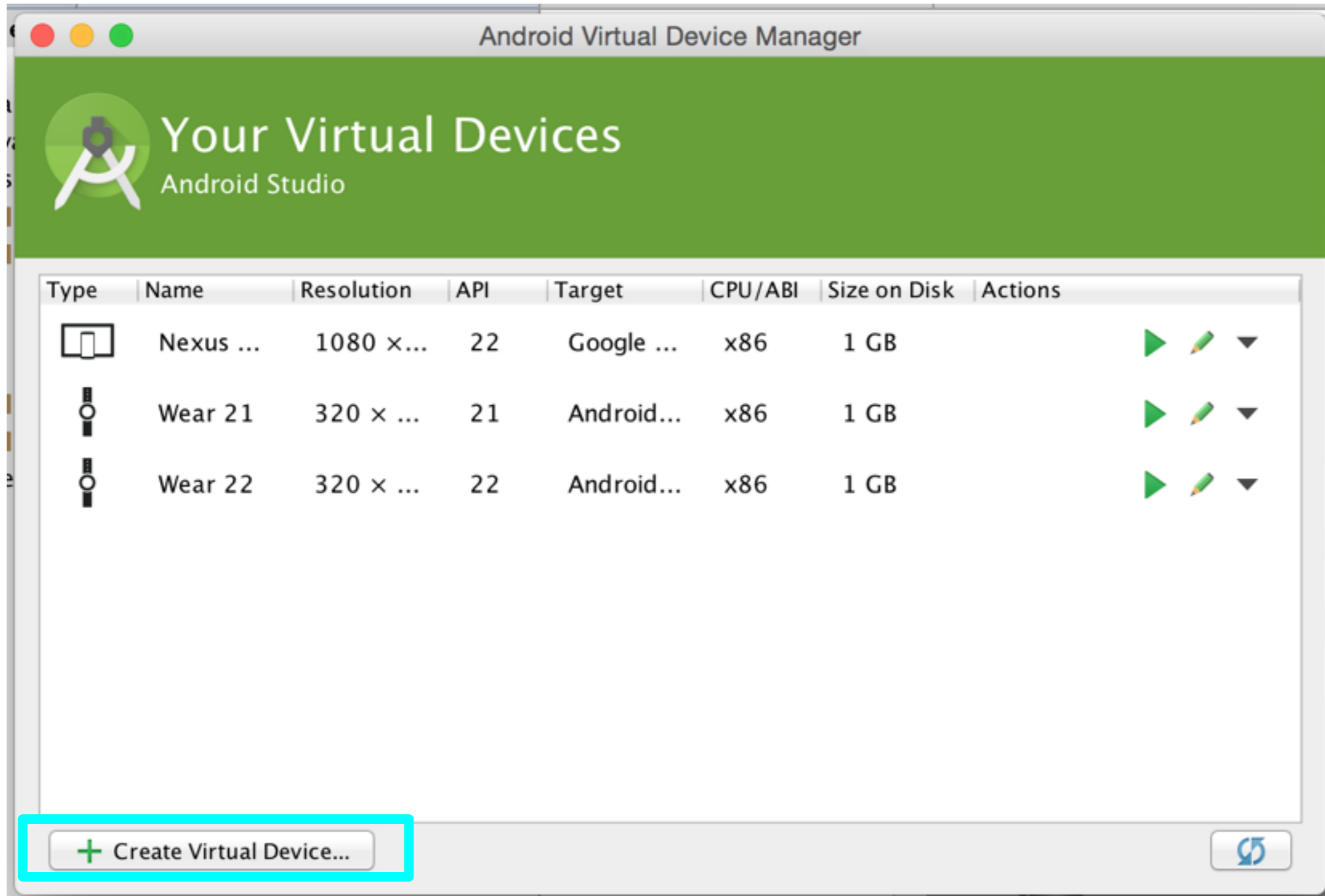
Show: Updates/New Installed [Select New or Updates](#)

Obsolete [Deselect All](#)

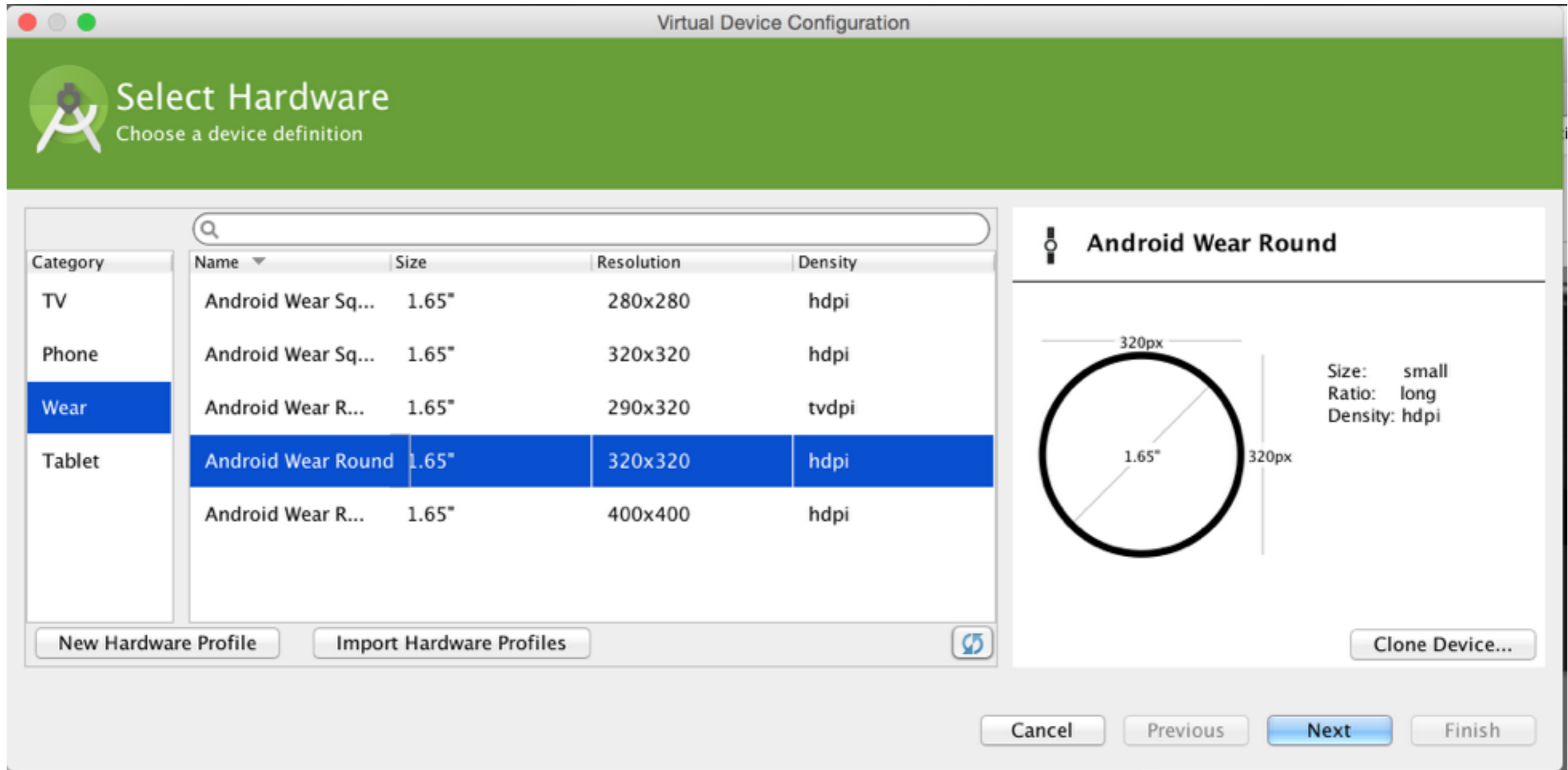
[Install packages...](#)

[Delete 1 package...](#)

Setting up the wear emulator




Setting up the wear emulator




Setting up the wear emulator


Virtual Device Configuration

 **System Image**
Select a system image

Release Name	API Level	ABI	Target
Lollipop	22	x86	Android 5.1.1 - a
Lollipop	22	armeabi-v7a	Android SDK Plat.
Lollipop	21	x86	Android 5.0.1 - a
Lollipop	21	armeabi-v7a	Android SDK Plat.
KitKat Wear	20	armeabi-v7a	Android SDK Plat.
KitKat Wear	20	x86	Android SDK Plat.

Show downloadable system images 

Lollipop



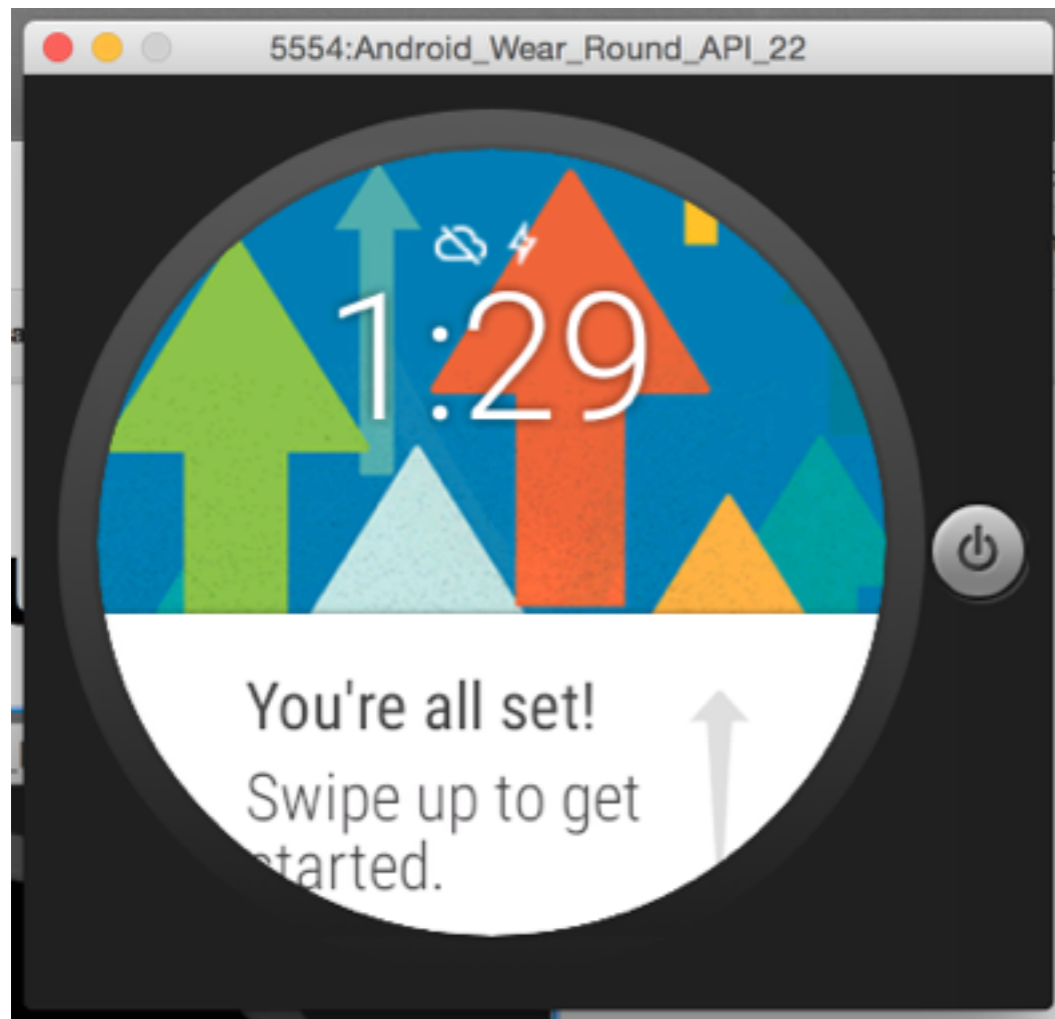
API Level
22

Android
5.1.1

Android
Android

System Image
x86

Cancel Previous **Next** Finish

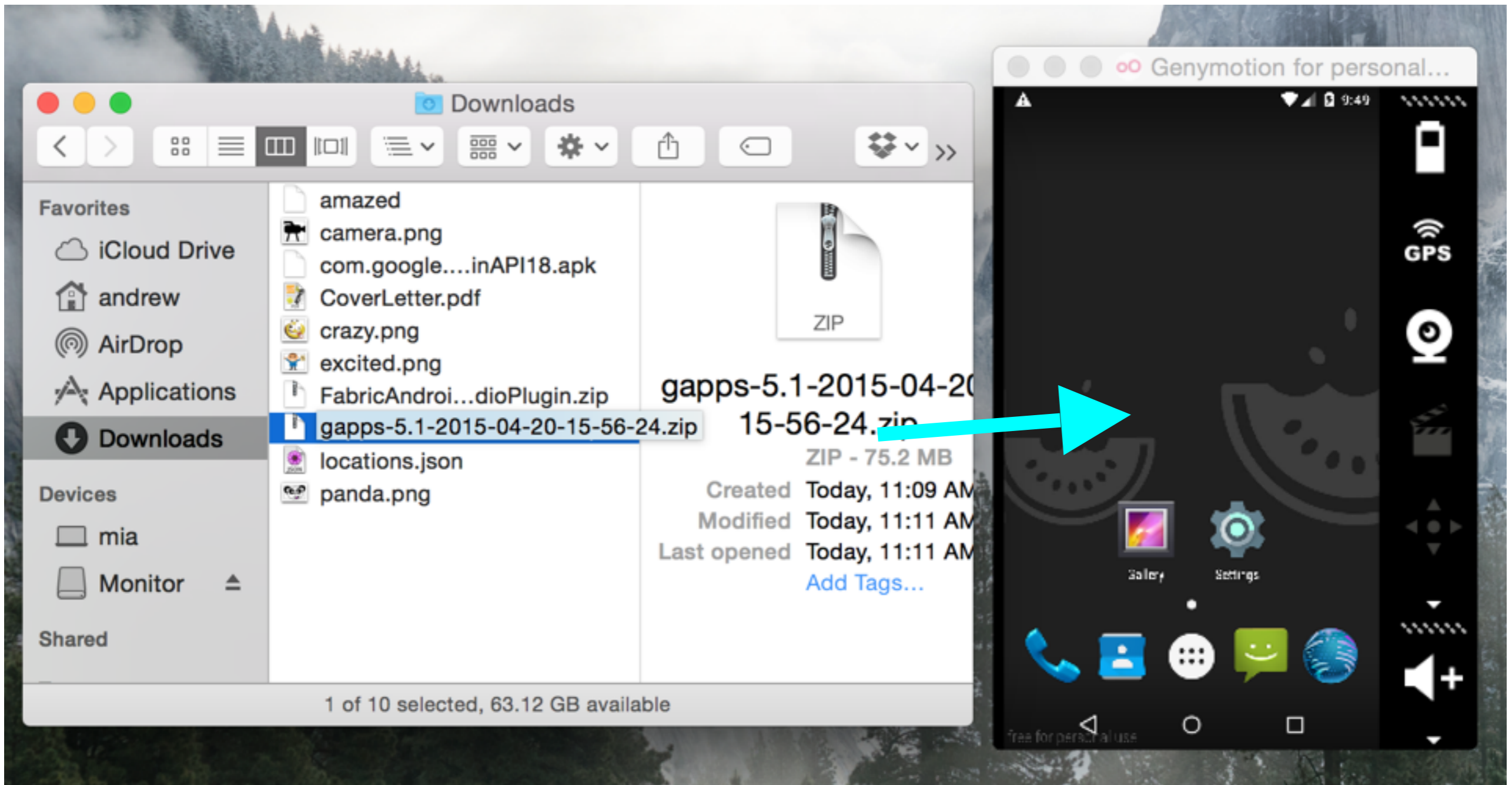


Follow the instructions to get to know the basic wear gestures:

- Swipes
- Cards
- Actions
- Dismissing Cards

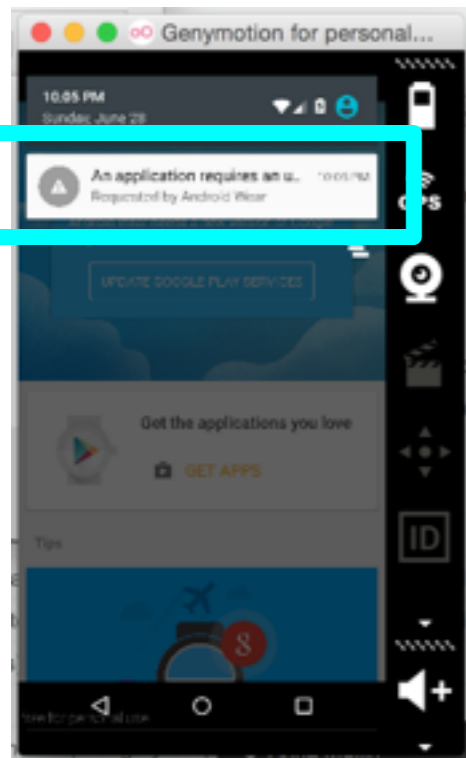
Genymotion: Installing Play Store & SDK

- Get dependencies for Wear and Play Store
 - [Google Apps APKs](#)
 - [Android Wear APK](#)
- Note: authenticate at own risk; if queasy, make new unconnected Google account for this assignment

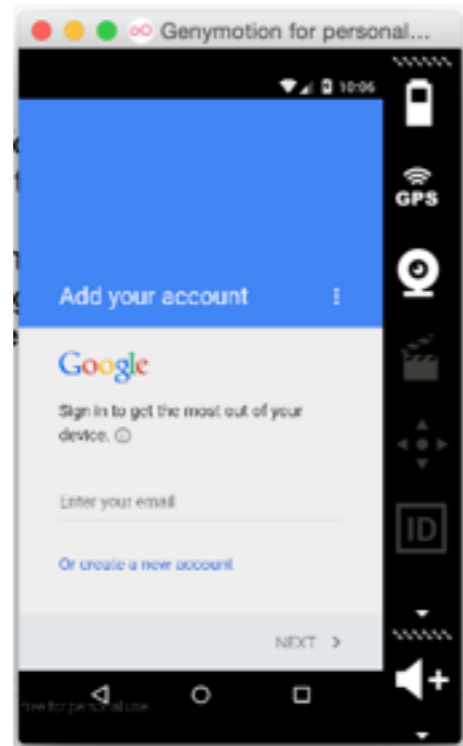


Drag and drop both the Google Apps zip and the Android Wear SDK into the Genymotion window to install. You might have to restart in between; restart the app after installing both.

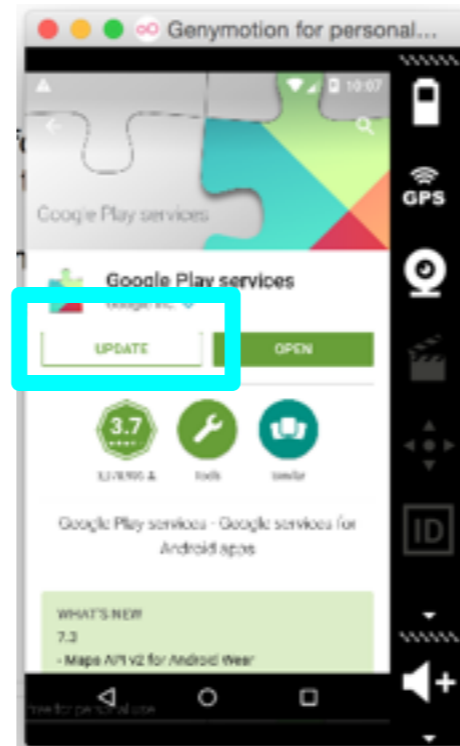
Pairing Genymotion & wear



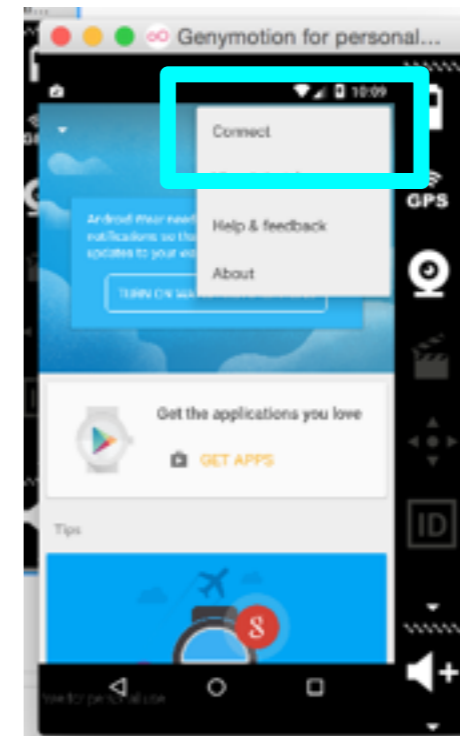
Click Wear Notification



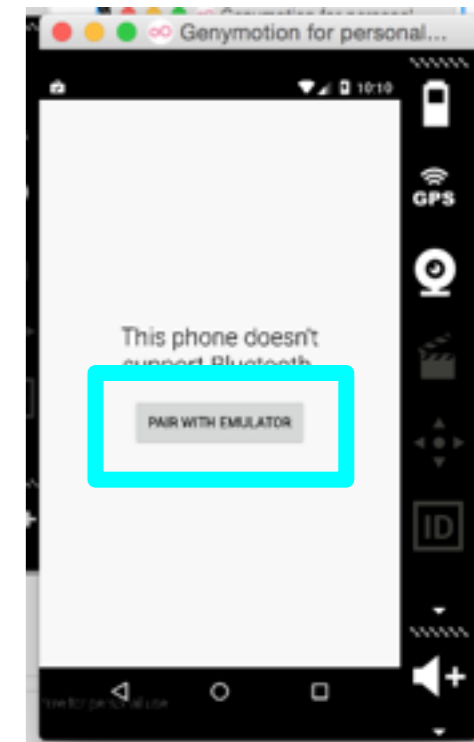
Sign In to Google



Update Play Services



Open Android Wear and Connect



Pair with Emulator

adb gateway

Type these two commands

```
C02MX11NFD58:~ andrew$ adb devices
List of devices attached
emulator-5554    device
192.168.57.102:5555  ————— device
C02MX11NFD58:~ andrew$ adb -s 192.168.57.102:5555 -d forward tcp:5601 tcp:5601
```

Do this every time your network restarts
(e.g., when you wake your computer up
from sleep)

- adb notes: if adb returns command not found, try adding its to your PATH (in ~/.bash_profile)
- if that doesn't work, you can cd to where adb was installed (most likely Library/Android/sdk/platform-tools) and run ./adb devices from there

~/

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