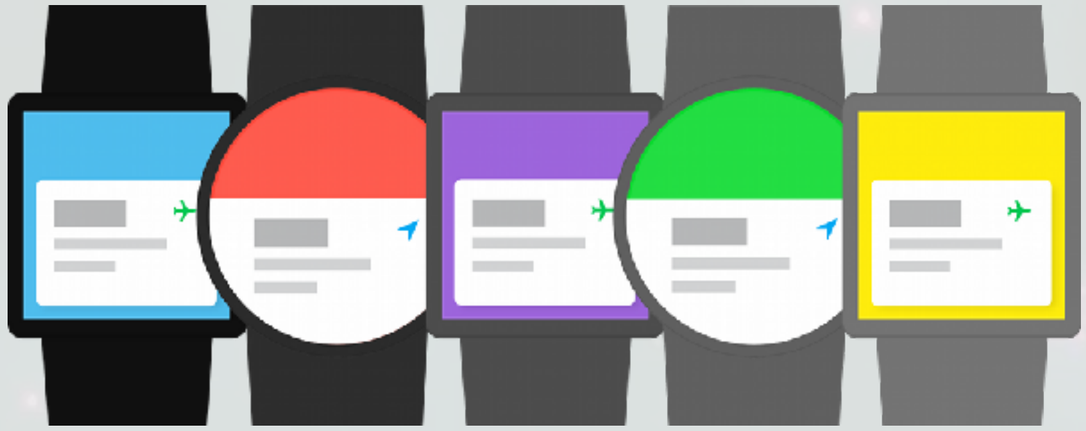
SCIENCES SORBONNE UNIVERSITÉ

Wearables

Renault@lrde.epita.fr

Wearable



- Current trends for Application
- provide quick access to your app
- Easily customizable
- Objectives: ensure that the user will not miss any notifications from its favorite app

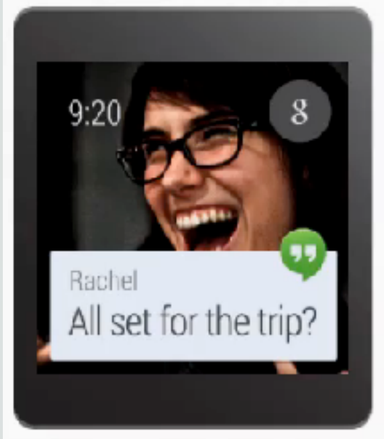
Many kind of wearable exists

- Activity trackers
- Watches
- Glasses
- ...

ContextStream and CueCard

ContextStream

- Smarter Notifications than in the phone
- Notifications are displayed vertically
- On a card, a right swipe displays more informations



CueCard

- open when saying "ok google" or when tapping the home screen
- Is triggered through voice intents

How to build a wearable activity?

Similar to a *classical* application

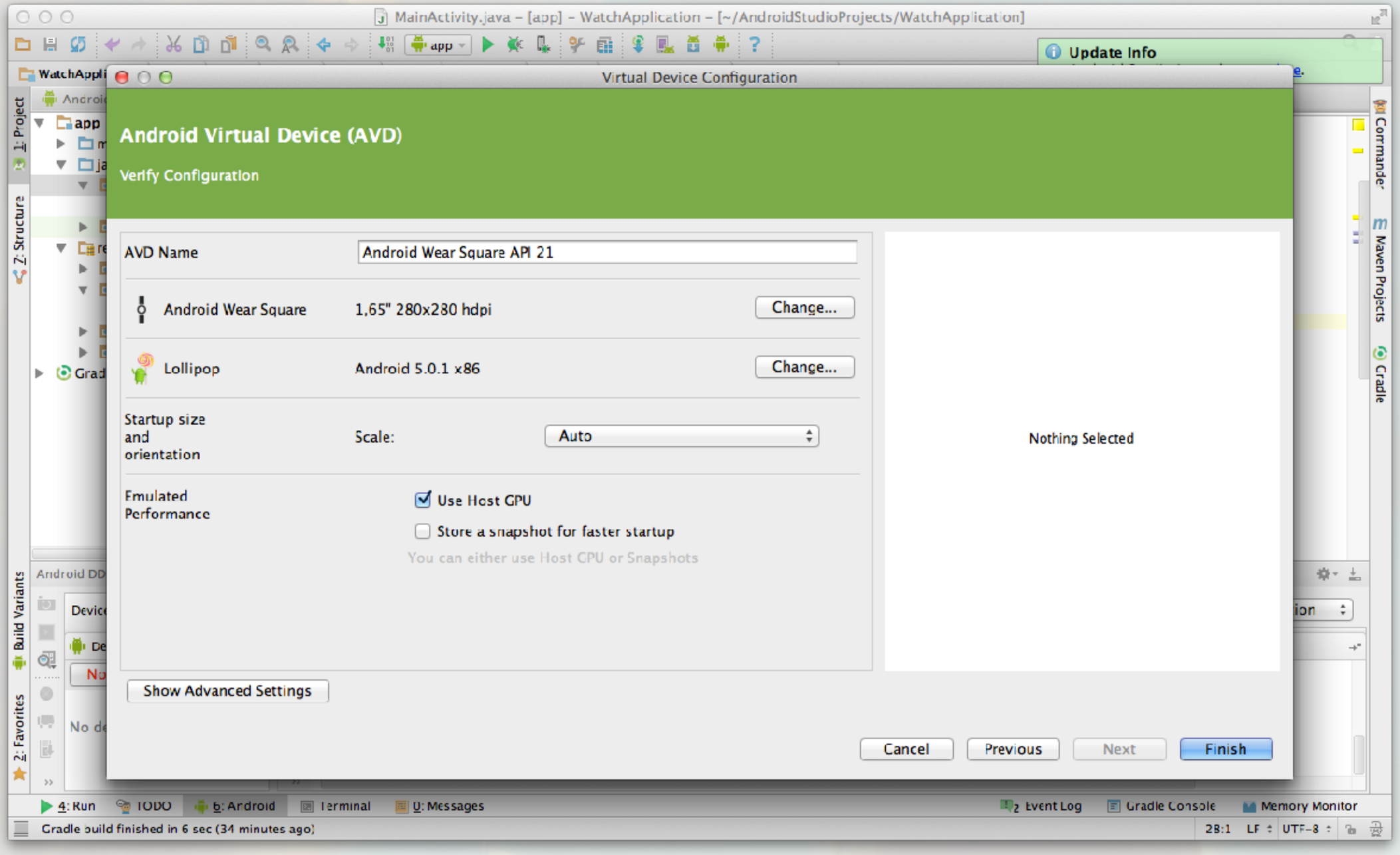
Things to know about:

- After some time the operating system is sleeping
  - **when the user will go back, its application will no longer be the active one**
  - **Home screen will be displayed**
- Small screen size !
  - **Not easy to manipulate a complex UI**
- Some frameworks are not supported
- Apps are not downloaded on the device
  - **The mother app is running on the phone**
  - **The wearable app is displaying on the wearable**

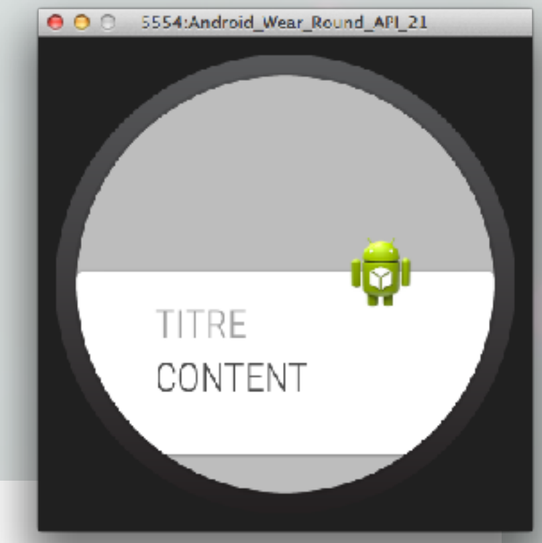
Building a test device

Building a virtual wearable is easy

- AVD manager provides a lot of predefined skills



Building a notification



Notifications are displayed after a top-down move

- An action can be defined when tapping a notification

```
Builder notificationBuilder = new Builder(this)
        .setSmallIcon(R.drawable.ic_launcher)
        .setContentTitle("TITRE")
        .setContentText("CONTENT");

 // Get an instance of NotificationManager Service
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);

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

Using voice to launch Applications (1/2)

Fundamental mechanisms for wearable

- Running predefined apps:
  - **Take a note, Set an alarm**
- Running third party apps:
  - **Reaction to the "Start" command**
  - **The application is then launched**

Running predefined apps

```
<activity android:name="MyNoteActivity">
    <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category
    android:name="com.google.android.voicesearch.SELF_NOTE" />
</intent-filter>
</activity>
```

Using voice to launch Applications (2/2)

Running third party apps

- Same mechanism
- Only specify the name that must be speak

```
<application>
 <activity android:name="StartRunActivity"
           android:label="My Running App">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
          android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
 </activity>
</application>
```

Voice interaction (1/2)

The app may interact with voice

- Build an intent to capture user voice
- Result is provided as a list of string

Start listening

```
private static final int SPEECH_REQUEST_CODE = 0;

// Create an intent that can start the Speech Recognizer
private void displaySpeechRecognizer() {
    Intent intent = new
    Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    // Start the activity, the intent will be
    // populated with the speech text
    startActivityForResult(intent, SPEECH_REQUEST_CODE);
}
```

Voice interaction (2/2)

Getting results

```
// This callback is invoked when the Speech Recognizer
// returns. This is where you process the intent and extract
// the speech text from the intent.
@Override
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent data) {
    if (    requestCode == SPEECH_REQUEST_CODE
         && resultCode == RESULT_OK) {
         List<String> results = data.getStringArrayListExtra(
                 RecognizerIntent.EXTRA_RESULTS);
         String spokenText = results.get(0);
         // Do something with spokenText
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Summary

Developing a wearable activity

- is similar to developing a classical one
- Android studio also provides builtin tools
- can't be done without a phone-side application

Some specificities must be considered

- ContextStream