ATLS 4120/5120: Mobile Application Development
Week 9: Android Intro

Android Development

Developer web site http://developer.android.com
Develop | Android Studio (or Get Android Studio link)
http://developer.android.com/studio/index.html (scroll down for system requirements)

- Android development can be done on Windows, Mac OSX, or Linux systems
- Android Studio
  - Integrated Development Environment (IDE)
  - Android Software Developer's Kit (SDK)
  - Emulator to run, test, and debug your apps on different virtual devices
  - initially released in 2013, previously the SDK was bundled with Eclipse
  - Can also use the SDK with the command line or another SDK
- Java programming language
  - Although Android uses Java it does not run .class or .jar files, it uses its own format for compiled code
- eXtensible Markup Language (XML)

As of Android Studio 2.2, the IDE comes bundled with a custom OpenJDK build which contains a bunch of additional fixes to make the IDE work better (such as improved font rendering).

Use the SDK Manager to download additional tools and components

Older versions needed the Java Development Kit (JDK) version 7 http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html Do NOT install version 8.
On the Mac you needed the Java Runtime Environment (JRE) version 6. This is NOT the latest. Either install it at this point or wait to see if Android Studio tells you to install it https://support.apple.com/kb/DL1572?locale=en_US

Android Releases

(slide)

- Android updates take about 3-6 months for a new version is available on a carrier
- Google -> phone manufacturers (OEMs)
- OEMS -> carriers
- When creating a new project you need to decide what Android versions to support
- latest stats on the Android Developer site dashboard http://developer.android.com/about/dashboards

Android Devices

Android runs on devices that are all different http://developer.android.com/about/dashboards (scroll down)

- Screen sizes
- Processor speed
- Screen density
- The number of simultaneous touches the touch screen can register
- The quantity and positioning of front and back cameras
- Bluetooth

The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen.
At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use.
The conversion of dp units to screen pixels is simple: px = dp * (dpi / 160).
For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels.

Screen densities https://developer.android.com/guide/practices/screens_support.html

Android Terminology

- An activity is a single, defined thing a user can do
  - Usually associated with one screen
  - Written in Java
  - Android Studio has activity templates
- A layout describes the appearance of the screen
  - Design view
  - Written in XML
- The emulator lets you define Android Virtual Devices(AVD)
  - A device configuration that models a specific device
- Simulators imitate the software environment of the device but not the hardware.
  - They have access to all the host(Mac) hardware's resources
- Emulators imitate the software AND hardware environments of the actual device
  - A flight simulator simulates a flight. If it was a flight emulator then you'd actually be transported to the destination.

Android Setup

The first time you launch Android Studio it will take you to a setup wizard.
Configure | SDK Manager to install other needed SDK components.
OR
Tools | Android | SDK Manager to install other needed SDK components.
SDK Platforms

- Android 7.0 (API 24) (show package details)
- SDK platform
- ARM EABI v7a System Image (for the emulator)

Tools:

- Android SDK Tools
- Android SDK Platform
- Android SDK Build tools (for the latest version)
- Documentation

Support Repositories (do these later)

- Android Support Repository
- Android Support Library
- Google USB Driver (windows only)
- Intel x86 Emulator Accelerator (HAXM Installer)

Chose top license, click Accept. Install.

Android Studio
Let's create our first Android app and make sure our environment is all set up.
From the Welcome screen chose Start a new Android Studio Project (File | New | New Project)

Application Name: Hello Android
Company name: a qualifier that will be appended to the package name
Package name: the fully qualified name for the project
C++ support NOT needed
Project location: the directory for your project /Users/aileen/Documents/AndroidProjects/HelloAndroid
Form factors: Phone and Tablet (leave others unchecked)
Minimum SDK: API 16(Jelly Bean)
On the next screen we can just take the defaults for now.
Add an Activity to Mobile: Empty Activity
Activity Name: MainActivity (we can leave this)
Generate Layout File should be checked
Layout name: activity_main
Backwards Compatibility should be checked

Android Studio Tour
The left most pane is the Project Tool Window which provides a hierarchical overview of the project file structure.
The default setting is the Android view which is what I usually leave it in but there are other choices for viewing your project.
Also notice the tabs on the left which will change what is shown in in the left pane.

The middle pane is the editor window which shows the file you're editing. You'll see different editors here depending on what type of file you're editing. If you have more than one file open you will see tabs right above the editor.
Above that is the navigation bar which provides another way to navigate your project files.

When you're editing a layout's xml file the preview window shows on the right side. You can toggle this using the tabs on the far right or hide it.

The bottom window shows the status. There are different tabs at the bottom to choose what you want to see and you can also hide it.

Now let's take a closer look at what was created in our project. In the Android view you see 3 main areas of your app

1. Mainfests: Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file describes the fundamental characteristics of the app and defines each of its components.
2. Java: These are your Java files.
    a. MainActivity.java defines a class for us called MainActivity that extends AppCompatActivity. AppCompatActivity is a class in the Android SDK that is a subclass of Android's Activity class. Extend in Java creates a subclass from a superclass.
    b. Context sensitive Java and Android documentation can be accessed by clicking on the declaration and pressing the Ctrl-Q (PC), Ctrl-J (Mac).
3. Res: these are resource files that include
    a. Drawable – images for the project

b. Layout – the layout files for the user interface in xml
    i. Activity_main.xml defines the RelativeLayout properties and the one TextView that was created for us.
c. Mipmap – includes files such as launcher images
d. Values – resource files that include values such as the values for strings, styles, colors, etc

So the logic is in Java and everything else is a resource.
If you switch to the Project view you'll see the actual file structure of the project. There are a lot of other files, but we're not going to worry about most of them.
I do want to point out just one file because you'll see it referred to.
App/build/generated/source/r/debug/com.example.aileen.helloandroid/R.java is automatically generated for every Android project and stands for resources. This is a utility class that provides references to the resources in the project.

Also, you're going to see the term Gradle often. Gradle is the build tool in Android Studio. It compiles and builds the app package file known as an APK file.

Android Virtual Device (AVD)
Tools | Android | AVD Manager
Create a virtual device
Category: Phone
Nexus 4 has the size: normal, density: xhdpi
System Image: MNC API 24 x86 Android 7.0 (if you don't see this one click Show downloadable system images)
AVD Name: Nexus 4 API 24
Leave the rest as is and Finish
Now you should see your AVD as a virtual device

Run the App
Close the AVD Manager and let's run the default app in the emulator.
Click the Play button to run your app.
Chose Launch Emulator and chose the AVD you just set up.
You might have to wait a bit for the Emulator to start and launch your app.
The first time an emulator starts it can take a few minutes.
You can leave the emulator running so it will be faster.
You might need to unlock your device – just swipe the padlock icon upwards and you should see the sample app.
Most newer Android devices have 3 virtual buttons.
Left button (back arrow) goes back to the last activity
Middle button (home) takes you to the home screen
Right button show a list running apps (like a double tap of the home button on iOS)
The favorites tray is at the bottom with the all apps button in the center.

Layout
Open the activity_main.xml file.
This is opened in the Design editor.
Two tabs at the bottom let you switch between text and design mode.
To the right of the Palette you can choose between Design view and/or blueprint view.
Moving across to the right is a button that let's you change the orientation.

Then there's a drop down that let's you chose what device you preview, the API, app theme, language, and layout variants.
Take a look in the palette to see all the layouts and widgets available for use.
A text view has already been added for us. Look at the XML for TextView.
In design mode move it to the center. Now look at the XML and see what was added.
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"

Notice the textview has **android:text="Hello World!"**
It's not good practice to hard code string values in your xml file, you should use string resources.
strings.xml in the values folder stores all the string resources.
Go into strings.xml and add this resource with some initial value so you can see it worked.
<**string name="text_message"**>Hello Android <**string**>
Back in activity_main.xml change the textview to **android:text="@string/text_message"**
The '@' sign means it's defined as a resource.
Now go into Design mode you will see our textview has the new string value.
You can click anywhere on **@string/hello_world** and click cmd-B (cntrl B on a pc) to automatically open the strings.xml file where the values for the string resources are stored.

Let's also make the text larger by either scrolling in the Properties pane to textSize or adding in the xml
android:textSize="48sp"
We will always use the "sp" unit for font sizes as it allows the font size to scale for the user based on their settings. (scale-independent pixel)
sp has scalable ratio: sp = px * ratio * scale. Where ratio never changes, but scale is user configurable.
This scale can be used by people who need larger font sizes, for example, to use device more comfortably.
Now let's change the background color through the xml.
In the <RelativeLayout tag before the close tag > add **android:background="#ff6232"**
Notice that a small colored square appears in the gutter to the left. Click on it and it brings up a color wheel.
Open up the preview and you'll see the result.

Code completion:
The editor in Android Studio has code completion just like in Xcode. As you type you will get selections. To accept the top most suggestion, press the Enter or Tab key on the keyboard. To select a different suggestion, use the arrow keys to move up and down the list, once again using the Enter or Tab key to select the highlighted item.
In the XML file because everything starts with "android" it gets old typing that. You can just start typing what comes after the : and the code suggestions will still come up. So instead of typing **android:background** just start typing **background** and you'll see **android:background** as a choice.
To see other suggestions, click on a word and then use cntrl-space and the editor will show you a list of alternative suggestions.

Can you guess how you would change the font color for the textview?
In <TextView> add **android:textColor="#FFFFFF0D"**