ECE 1778 - Creativity and Programming for Mobile Devices
January 2013

Programming Assignment P1, for Programmers

Introducing Yourself + Development Environment & Simple Widgets

PART I

A key part of this course is to form an inter-disciplinary group to do a new and exciting application. To both help form those groups, and to make sure you have sufficient background as a programmer, please create the following:

1. A text description of your background, as follows:
    - A list of your degree(s), where you received them. Be sure to include the field you were studying.
    - A list of the computer programming courses that you have taken.
    - A list of the programming projects you have undertaken, together with the size (in number of lines of code) and the computer language used.
    - Give the list of any companies that you have worked for as a programmer, if any.
2. Create a video of yourself (on YouTube, as an 'unlisted' video) describing your most significant programming project done in the past – its goal, your role in its creation, and how well it worked. **The video must be less than 2 minutes long.**

Go to the **Pepper** course website (you'll see instructions on how to acquire a Pepper account on the Blackboard Portal announcement) and click to the 'Home' page. There, select the 'folder' labeled **Introductions**, and then **Programmer Introductions** and create a **New Note** with *your name as the title*. In the text of the Note, put your text from (1) above. Embed at the end the unlisted video you created in (2). To learn how to embed the video into your note, read the **'How to upload your YouTube introduction in Pepper'** note that the TA, Alexandra Makos, has put there.

This is due on Tuesday January 14th at 6pm. Sooner is better, though! A penalty will be levied if late.

PART II

The goal of this part of the assignment is to set up the Development environment that you will use throughout this course, and make a basic 'Hello World' program and run it. As we are trying to move quickly on the basics in the course, you will also learn about how to layout simple buttons on the phone and how write code that reacts to them.

For Android Programmers

In this course you must have access to a Windows, Linux or Macintosh computer, all of which are supported in the Android environment. You will have to download and install several packages on your computer to begin learning development. Go to this web page to begin:

http://developer.android.com/sdk/index.html

Follow the instructions to download and install the **ADT Bundle**, which contains everything you need.

To ensure that you've got the basic setup working, do the "Building Your First App" tutorial described in the tutorials section of the Android Developers website: http://developer.android.com/training/basics/firstapp/index.html. The key thing is to learn how to start a project and make the Android Phone emulator work on your development computer. It will also be helpful to read this section of the Android developer's website: http://developer.android.com/tools/workflow/index.html

The installation and tutorial make use of the Eclipse development environment (which is used in many other software development contexts) together with the Android Development Tools (ADT), which is specific to the Android operating system.

Learn Basic Environment

Read pages 1 through 123 of the Murphy Book, **The Busy Coder's Guide to Android Development, version 5.4**, doing the small coding exercises given there. This textbook can be downloaded from http://commonsware.com/Android/, a license key can be obtained by emailing Braiden, the technical TA for the course.

You can simply read through the tutorials if you wish. Tutorial #1 gives a second set of instructions on how to install the development kit, in a somewhat different order, **which you won't need if you've followed the instructions above.** If you have trouble with the instructions above, you could try these instead. Tutorial #2 is a lot like the first app tutorial that you did above.

This will expose you to the basic development environment, as well as the structure of an Android Project's files, and the Eclipse environment. The later pages move on to describe *activities* (which are the pages that an app user sees), and how to lay these out. This includes user interfaces such as buttons and text fields, and how to display images.

You can download all of the examples in the **The Busy Coder's Guide to Android Development** book from the website https://github.com/commonsguy/cw-omnibus. To download all of the examples in a zip or tar file, click on the download 'zip' button on the left upper part of the page.

If you want to run these examples within the Eclipse environment, start Eclipse, see the instructions on page 35 of the text. (Basically choose File->New->Android Project. Put a related name in the 'Project Name' box, and click on the 'Create project from existing source' radio button. Using the browse button next to the 'Location', select the directory that contains the complete project – i.e. the one that contains AndroidManifest.xml, assets, bin, gen, libs, res and src folders/directories. Click 'finish' and the code will be imported into Eclipse as usual.)

For iOS (iPhone) Programmers

Go to the website https://developer.apple.com/devcenter/ios/index.action and register, and download the Xcode 5 development kit. To be enabled to download code into an actual device, you will either have to pay the $99 annual fee, or go to the following site that explains how to acquire the UofT site license for these tools:

http://mobile.utoronto.ca/build/ios

Read and do the exercises in Chapters 1, 2, and 3 of Beginning iOS 6 Development, by David Mark, Jack Nutting, Jeff LaMarche and Fredrik Olsson, which provide a good introduction to the iOS basics necessary to do this assignment.

Assignment

Write a mobile application that presents the users with four fields:

1. A text field that initially contains the word 'Things haven't started yet'
2. A button labeled **Change**.
3. A button labeled **Picture**.
4. The standard menu.

The program should respond to the pressing of the buttons in the following way:

- When **Change** is pressed, the text field should have its contents changed to 'The Change Button has been pressed 1 times.' Subsequent presses of the button should increment beyond the number 1.
- When the **Picture** button is pressed, a picture of a dog should appear below the buttons. The next time it is pressed, the picture should disappear, and then appear on alternate presses.
- Also, make use of the default menu that is supplied in Android, and change the 'settings' button to become a 'reset' button. When this button is selected, the text field should return to be 'Things haven't started yet' and the count should be reset to 0.

You should only need an emulator to do this assignment, not an actual phone.

With all assignments, including this one, we'd like you to produce applications that work well and robustly, as good training for the app that you'll make in the project. As such, we require that you follow **Braiden Brousseau's guide to Quality Apps**, which is appended below.

To Hand In

For Part I: Due January 14th at 6pm, as described above.

For Part II: January 21st, 6pm. Marked out of 10, 0.5 marks off every hour late.
What to submit:

Android: a zip file containing your complete project, runable from Eclipse.
iPhone: a zip file of complete project, runnable on Xcode 5.

Submit your zip file through the Blackboard Portal for this course. Be sure to submit it to the assignment ‘P1’ To do this, go to the **Assignment** section of Blackboard, and click on Assignment P1. You should see, under ‘2 Assignment Materials’ a place where you can attach your file.

<u>Braiden Brousseau's Guide To Quality Apps</u>

The purpose of this guide is to ensure that the software you create in this course – both the project and the assignments, meet a certain standard of quality and robustness. The assignments will include grades allocated towards these guidelines, as motivation to make sure your code works well. This will stand you in good stead as you build the large app that is your project. These guidelines come from our previous year's experience with marking assignments and projects. These guidelines are written for Android programmers, but similar concepts apply for iOS.

Don't let the User crash the App

The user shouldn't be able to crash an app by pressing touch objects in the "wrong" order or by spam clicking something. If an activity requires button 1 to be pressed before button 2 (for example to select what data file should be used before processing), then pressing button 2 first should not crash the app. Nothing should happen, or better yet the user could be notified that they must select a data file first by pressing button 1.

Don't make the user wait

Avoid unnecessary slowdowns caused by overuse of new activities and fragments where not appropriate.

Avoid excessively reading and writing of large files to disk. For example if you want to make a picture move from left side of the screen to the right, move the location of the ImageView, don't destroy the ImageView and make a new one at a new location reloading a 2-3MB photo.

Make use of time while a user is idling. In an app that shows a user the top stories of the day from a website, load the data for story 2 while the user is reading story 1. It is very likely the next action from the user will be to click for the next story. This is better experience than having to wait for a download every time.

Use UI Elements Appropriately

Although you can programmatically set the text in an "EditText" box, if the app has no intention of the user entering text here use a "TextView" or another element that the user cannot change.

Content-Independent Interface

The UI should behave and look similar regardless of what data is loaded or entered. For example, loading a picture of different sizes should not cause buttons in the app to physically move around to different places on the screen. Loading and displaying a large text file shouldn't push UI elements off of the bottom of the screen, or somewhere else unreachable. If the UI changes based on what content is loaded it should be intentional

Appropriately Sized and Labeled Touch Targets.

Avoid making UI elements (that the user must touch) very small and close to other elements they might touch by accident. It should be clear in an app what can be touched and ideally what action will be taken. One can use text labels, icons, pictures, colours etc. to convey to this type of information.

Fill Space Appropriately

UI elements should have a fairly commonsense space occupancy. For example, suppose an activities' main purpose is to measure how long it takes to run a certain distance and show you your current time along the way. Making the running time a font 8 textview in the upper right hand corner while making a “share my time to twitter” button 3/4 of the screen is probably bad design.