ECE 1778 – Creative Applications for Mobile Devices

September 2016

Programming Assignment P1, for Programmers

Introducing Yourself & Development Environment & Touch Buttons

PART I

A key part of this course is to form an inter-disciplinary group to make a new and exciting application. To 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, with the following items:
    - A list of your degree(s), and where you received them. Be sure to include the field(s) you were studying.
    - A list of the computer programming courses that you have taken.
    - A list (name and one line description) of the programming *projects* you have undertaken, together with the size (in number of lines of code) and the computer language used.
    - A list of any companies that you have worked for as a programmer, if any, with a brief description of your responsibilities.

2. Create a video of yourself 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 no more than 2 minutes long.** Upload this video to YouTube by going to the site https://www.youtube.com/upload and set this as an 'unlisted' type video. (This means only people with the link can see it). Once you've done this, be sure to copy the YouTube link that is provided, for use in posting your video below.

To post these for viewing by the class, go to the **Piazza** course website (instructions for how to enroll into the Piazza are given in an announcement on the Blackboard Portal announcement/email). Click on the **programmer_intros** link shown at the top left of the site. Click the 'new post' button, and insert a new 'question.' You should place both your written text (from (1) above) and your video (from (2) above) into this post. To insert a video click 'insert' -> insert video, and in the *insert media* popup, use the 'general' method, and past your YouTube link into the *source* field.

Part I is due on Thursday September 22nd at 6pm. Sooner is better, though! A penalty will be levied if late. See below for Part II.

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 Mac computer, all of which are supported by 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:

https://developer.android.com/studio/index.html

Follow the instructions to download and install the **Android Studio**, 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: https://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: https://developer.android.com/studio/workflow.html

Learn Basic Environment

Obtain a copy of **The Busy Coder's Guide to Android Development, version 7.6.** First obtain a free coupon code for the book from Braiden Brousseau (braiden.brousseau@utoronto.ca), one of the TAs for the course. [To use your coupon code, create an account on the Warescription site (https://wares.commonsware.com). Then, click "Subscribe!" in the nav bar at the top of the page, paste in the coupon code and your name in the "Apply a Coupon" form, then submit the form. Your book editions should be ready in 20-30 seconds if nobody else's books are being generated right then, in the "Your Book" portion of the site.]

Look through pages 1 through 156 of the Busy Coder's Book, doing the small coding exercises given there. There are parts you can safely ignore, mostly associated with the older Eclipse environment. Also, some of this repeats what you learned from the above websites, but it could make for a handy reference.

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 Android Studio 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.

For iOS (iPhone) Programmers

Go to the website https://developer.apple.com click on the member center and register, and then download the Xcode 7.3.1 development kit from the mac App store. 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

For iOS-based devices (iPhone, iPad) you have two choices, depending on which language you wish to use – Swift or Objective-C. At this point, I would advise Swift, as it is the focus of Apple and has been for a few years.

1. The newer **Swift 2** language, The newer **Swift 2** language, the **Beginning iPhone Development with Swift 2** by Mark, Topley, Nutting Olsson and LaMarche. It can be found here: http://www.apress.com/9781484217535. This is $USD32 as of this writing (e-book price is $16USD).

2. The older **Objective-C** language - use the book **Beginning iOS 7 Development** by Jack Nutting, Fredrik Olsson, David Mark and Jeff LaMarche , is good, and I think still applies to the current iOS9. It can be found here: http://www.apress.com/9781430260226. It can be purchased electronically, right off the Apress website, and is $USD 34.99 as of this writing.

The two books have similar chapter structure; for this assignment read and do the exercises in Chapters 1, 2 and 3.

<u>Assignment</u>

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

1. A text field that initially contains the word ‘No Clicks Yet’
2. A button labeled ‘**Register a Click’**.
3. A button labeled ‘**Toggle Image’**.
4. A menu item.

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

- When ‘**Register a Click’** is pressed, the text field should have its contents changed to ‘Clicked 1 times.’ Subsequent presses of the button should increment beyond the number 1.
- When the ‘**Toggle Image’** button is pressed, a picture of a dog should appear. The next time it is pressed, the picture should disappear, and then appear on alternate presses.
- Also, make a menu item that will act as a reset. When this button is selected, the text field should return to ‘No Clicks Yet’, the count variable should be reset to 0, and the picture of the dog if visible should become invisible.

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 Thursday September 22nd at 6pm, as described above.

For Part II: Thursday September 29th, 6pm. Marked out of 10, 0.5 marks off every hour late.
What to submit:

> Android: a zip file containing both a final .apk of your assignment and your complete project, runnable in android studio.
>
> iOS: a zip file of complete project, runnable on Xcode 7.3.

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 **Programmer Assignments** link on the left side of the blackboard portal for this course, and click on Assignment P1. You should be able to upload your submission file there.

Braiden Brousseau’s Guide To Quality Apps

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 waiting for large files to be read or written to storage. For example if you need to load a very high resolution image consider loading a thumbnail version first while waiting for the full resolution version. When saving a large piece of data consider doing so asynchronously if the required save time noticeable impedes use of the app.

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.

List and other scrollable elements should always scroll smoothly.

In the context of the assignments in this course your app should never ‘feel” slow or laggy. Most of the time your apps will feel fast and smooth without any intentional consideration while programming them. When they don’t however it is usually a sign something is implemented in a considerably suboptimal way and you are encouraged to research and evaluate other techniques and implement something slightly more complex.

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, colors etc. to convey to this type of information.

Fill Space Appropriately

UI elements should have 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 font size 8sp in the upper right hand corner while making a 'share my time to twitter' button 3/4 of the screen is probably bad design.

Explore the apps you use everyday for inspiration on this type of design. Ultimately we are not looking for programmers to be amazing designers but we want you to learn and demonstrate your ability to finely control UI elements. Becoming confortable with this during the assignments will pay significant dividends when the project works starts as you have short development cycles in between presentations of your progress.