ECE 1778 – Creative Applications for Mobile Devices
January 2015
Assignment A3, for Appers

Understanding Parts of the Canvas

One of the goals of this course is to give experience in reaching across disciplines in a meaningful way (on the way towards our goal of creating a novel application of a mobile device). In this assignment, you will be asked to engage in a learning process about the capabilities of the computers that are part of the mobile devices, with your programming partners, and then to continue your learning on your own.

We believe that there is much to be gained by reaching across disciplines, and learning some of the language and basic understanding in other disciplines. As an Apper, you will be teaching the programmers some of the key terms in your field - their meaning and the underlying concepts behind those terms. In this assignment, we are asking the programmers to teach you some of the key terms and understandings in the Computer Engineering and Science field.

1 Key Capabilities of Computation

The broad field of computer science and engineering can be broken down into many different sub-fields, at least one of which you're sure to be using in your project. Below is a list of different capabilities along with a small amount of context in which it is used. You are to choose **one** of these areas – one that you <u>are not</u> already familiar with, but your programming partners are - and to engage in a two-step learning process about it, aided by your programming partners.

1. **Fast Searching**. Computers are very good at organizing data in many ways, including in a database, but also in complex data structures, that make it easy to find something that you're looking for very quickly. A ground-breaking example of this is something called a Hash Table which is also called an *Index*. Google, for example, has 'indexed' the entire public Web so that it can be searched very quickly. Ask your programming partners to explain what kinds of problems a Hash table is used to solve, and how it cleverly does it better than the more obvious methods.

2. **Databases.** Similar, but not quite the same as fast searching, there is a large field of research and software that engages with a method of storing large amounts of related information, called a Database. Ask your programming partners to give you a coherent view of what a database is, and how it is used, and issues that occur in making use of it under different scenarios – such as large personnel databases, users of a website, the data that must exist in the Flurry database, and others that they know about.

3. **Digital Signal Processing.** Many of the things that our senses can perceive in the 'analog' world – sight, sound, touch – are converted into a series of discrete, digital numbers when represented inside a computer. The very broad field of digital signal processing has produced many techniques to enhance those sensory data – with the goal of trying to 'understand' it at some higher level, just like we do. Three examples of Digital Signal Processing are:
    - Filtering – isolating out different frequencies in a signal (which could be any one of a sound, picture or touch sensor data)
    - Speech Recognition – a major field itself, which converts the sound to known words, in text form.
    - Computer Vision – also a major field, which converts pixels into objects as well as other tasks such as tracking.

    Ask your programming partners to describe what and how one of these three tasks (filtering, speech recognition or computer vision) works – what is involved in the key steps of the task, and how well it currently works in the state of the art.

4. **Optimization.** Given a few choices, a human being is pretty good at selecting among those choices, which might be the best. Given millions, billions or trillions of possibilities, computers can do a great job of sorting through these and choose the best, whereas a human has little hope of success. There is a gigantic field of automatic 'optimization' for which many different methods have been invented in mathematics, computer science, and engineering. Optimization could perhaps be divided into two sub-categories:
    - **Combinatorial -** in which there are many different combinations of possible things to look at – for example, selecting the best route between two points on a map, in the presence of traffic data. Example techniques in this area include 'greedy' methods, 'hill-climbing' methods, dynamic programming, and the 'branch and bound' method.
    - **Continuous -** here the set of possible 'choices' is represented as a set of numbers, which can take on any value. This might be used to select locations of water-bomber planes in anticipation of the summer fire season for cross-Canada forest-fire fighters. Example techniques in this area include Linear Programming and Convex Optimization methods.

    Ask your programming partners to describe one application of these methods, and how they actually do the solution. Make sure to spend some time understanding the 'complexity' of the approach, and what that means.

5. **Internet Communication.** The modern world is now predicated on vast communication networks that connect our wired desktops and wireless mobile devices to everyone! Having some sense of the layers of communication, both physical and virtual, that connect us could well enhance some of the projects you will do. For example, you should learn about the Domain Name Server (DNS) that is part of the Internet address translation. Ask your programming partners to describe what happens from an entry on the Google web search page and where it goes, when you press enter. They will have to speculate some, but it would be

good if you had a clearer sense of all that is involved in that action. This should include some level of technical detail that includes notions of client/server, the nature and connectivity of the network (get the programmer to show you how many 'hops' and the average latency in the network it takes to go from your computer to Google's servers), but need not be concerned with actual search process itself (#1 above).

2 Learn from Your Partner

Choose **one** of the areas listed above (again: one that you are not already familiar with, but your programming partners are) and have your programming partners teach it to you. Ask the programmers to use one or two concrete examples of the concept that you have selected to help you understand it. Each of the concepts can be quite broad so if you select examples, this will help to focus your understanding. Take notes. Then, go away from them and write up a 2-page (500 words plus pictures; it *is really important to use pictures here*) description of the concepts involved. While you may connect with them and ask questions, do not have them edit your document in any way. Include a section in which you detail the things that you don't fully understand, or would like to understand better.

Submit this document online on Friday February 6th by 6pm, before moving on to the next step.

3 Take A Few Steps on Your Own

Using the Internet, pursue a deeper understanding of the topic have chosen. Look up the things that you didn't understand or wanted to know more about. Write a new document, also 500 words in length that conveys this deeper understanding. **In addition, write a single paragraph on your view of this learning process – how it went, how much you learned from Part 2 vs. Part 3, and what would have made it better.**

Due date:
For Part 2: Friday February 6th, at 6pm, 0.5 marks off every hour late.

For Part 3: Tuesday February 10th, at 6pm, 0.5 marks off every hour late.

Full assignment marked out of 10.

Submit your documents on the Blackboard, under the 'A3' Assignment, clearly delineating which is Part 2 and which is Part 3.