Extirswipe:

* Start Date:

October 2015

* Completion Date:

January 2016

* Languages & Technologies Used:

Swift, iOS

* Inspiration:

I was simply messing around with Xcode to see what I could do with it. One thing led to another, and a few months later, I had a game. (And learned how to code in Swift along the way).

* Description:

Extirswipe is a speed based matching game where you have a limited amount of time to match objects of certain categories (shapes, colors, or numbers).

* Development & Challenges:

Developing Extirswipe was particularly challenging because Swift was the first Language I had taken upon myself to learn that had significantly different syntax than Java. Instead of taking an online course in Swift or going through a tutorial, I decided to tackle the learning head on by programming what I could and researching whatever I didn’t know the syntax for. I feel like this was a much more rewarding experience than if I had just taken an online Swift course instead.

Shortcomings:

As an app, this had many shortcomings. First of all, since I was using this as more of a way to teach myself a new programming language rather than an actual app, I wasn’t following any particular design plan. This lead to an array of problems such as the game being too difficult to intuitively understand how to play, and for the game itself to lack elements that would make it interesting enough for a user to keep coming back to play more.

* External links:

Github:

BUG:

* Start Date:

August 2016

* Completion Date: B.U.G. is never complete >:)
* Languages & Technologies Used:

ATmega32u4 (Adafruit 32u4 Bluefruit), Bluetooth LE, Motors & Drivers, Ultrasonic (obstacle avoidance), camera, piezo element, C++/Arduino

* Inspiration:

Long long ago (summer of 2015), I was on the 3rd floor of an electronics mall in Taipei, Taiwan when I walk into a somewhat sketchy looking shop. To my delight, they were selling a whole bunch of cool stuff (Arduinos, Raspberry Pis, robots, etc.). After looking around a bit my eyes settled upon a plastic chassis (I had attempted hacking a really old motorized toy I had with my Arduino before with limited success). I bought the chassis with the intent of getting my Arduino to control it someday.

Toward the end of summer of 2016 (about a year later), I was bored and decided to build a very small box like robot using the chassis that I had bought. Instead of using the full chassis, I decided just to glue the gearbox motors together with a piece a cardboard resulting in a much smaller size.

I would often bring this contraption to school with me and a friend started calling it my little bug. Hence, as a joke, I named it B.U.G., the Basically Useless Gearbox.

* Description:

The Basically Useless Gearbox, or B.U.G. for short, is a robot that can dance, wander around aimlessly whilst avoiding obstacles, take pictures, play music, or follow your ever command via the Adafruit app.

* Development & Challenges:

One of the main challenges with B.U.G. is its size. I can attach so many things to it before it becomes impractical. This made powering BUG a challenge. At first, I used 4 triple As and tried a few different placements and battery packs, after a time it became clear 4 triple As and their 1000 mah capacity would give my 500ma power hungry robot very little battery life. I eventually switched to 4 double As and I plan to eventually switch to rechargeable li-ion.

Shortcomings:

Not too many shortcomings in terms of living up to its name I’m afraid. However, if I were to choose something, I think that B.U.G.’s size is both a redeeming and restricting feature. It limits what I can add on to it while also challenging me to see what I can do on such a small platform.

External links:

Github:

Seat a Student:

* Start Date:

Winter 2015

* Completion Date:

Summer 2016

* Languages & Technologies Used:

Java, Machine Learning/Genetic Algorithm, Graphs (data structure)

* Description:

“Seat a Student”, uses student preferences and classroom layout to generate a seating chart.

* Inspiration:

One day I watched my high school computer science teacher as she slogged through completing seating charts for each of her 5 classes. She would always pass around half sheets asking students about their seating preferences (who they did/didn’t want to sit next to, preference to sit closer to windows or front of the room, etc.). It seemed like such a repetitive and arduous task and it made me wonder if there was a way to get a computer to do it…

The idea had slipped out of mind for a few months until I learned about a data structure called a graph. The graph instantly reminded me of the seat a student problem: each student could be a node while each listed preference for another student would be an edge connecting the two students! (I named it “seat a student” to parody the intro programming class’s assignment called “pick a student”) I got to work coding and white boarding, and I soon realized that although I had a powerful way of modeling the problem. I had yet to find a decent way to solve it. Once again the project faded back into the depths of my mind.

That summer I participated in an internship at a startup company called “The Wireless Registry”, and they wanted me to look into methods to optimize their mobile API. One day while looking for machine learning algorithms and libraries I could implement, I came across evolutionary algorithms. I realized this was the last piece of the seat a student problem! I came up with a way to represent possible seating arrangements in chromosomal form and ran these in the algorithm to optimize the seating charts.

* Development & Challenges:

Several times during this project, I ran into road blocks that hindered me from proceeding simply because I didn’t know enough about what I was doing to know where to even look for solutions. I overcame this just by keeping the idea/project in the back of my mind and trying to learn as much as I could. I have begun to realize that sometimes I have to give myself some time so my knowledge has time to catch up with my ideas.

Shortcomings:

The UI/UX. The whole program was text based and I never really got around to building a GUI and that made it pretty difficult to use.

* External links:

Github: https://github.com/JThr3e/SeatAStudent

Music USB:

• Start Date:

January 2017

• Completion Date:

May 2017

• Languages & Technologies Used:

ATTINY85, Arduino/C, Piezo elements, USB

• Description:

The Music USB is a small USB flash drive looking device that, when plugged in, will play a tune.

Inspiration:

I was thinking of some ways to maybe pull off a funny senior prank. I decided that since the computers at my school had a free USB port on the side facing away from the monitor, this would be a perfect place to hide and/or power a pranking device. To me, the idea of the media centers computers sputtering out video game theme songs whilst frantic librarians and students rushed to find the source seemed quite hilarious.

• Development & Challenges:

Unfortunately after completing my design I realized I could barely hear the music the piezo was playing! No one would even notice it if they weren’t in a completely quiet room.

A few months later, I was on adafruit.com when I read about ‘differential drive’ for piezo elements. This method used two I/O pins to swat the voltage levels back and forth between high and low that allowed the Music USB to be twice as loud. Alas, school was already too close to ending to have enough time to pull off the prank.

Shortcomings:

I never got to use it to prank anyone D: … perhaps I shall use it sometime in the future ;)

• External links:

Github:

EQ Boom Box:

• Start Date:

May 2017

• Completion Date:

July 2017

• Languages & Technologies Used:

Arduino/C, Analog Passive Filters (RC), Amplifiers, LED matrix, Serial,

• Description:

The EQ Boom Box has an AUX cable that can be plugged in to your phone. As the EQ Boom Box plays music, the LED display will show a 3 band EQ of the audio that is playing.

Inspiration:

One night I decided to go dumpster diving. My neighborhood occasionally has public dumpsters where people can throw away things they don’t need anymore. So, I decided to see if there was anything that might be of use to me. Among my spoils were many different types of cables, chargers, wires, a set of nice speakers (in working condition), and … an iPhone?! Anyway, I decided I would hook the speakers up to a cheap amplifier I got off amazon so they could play music from my phone. This worked, and was cool and everything, but I thought about ways to make it cooler. I decided I should add an LED board EQ display.

• Development & Challenges:

To make an LED display of the EQ, I had to find some way to process the audio signal to get the amplitudes of certain frequency bands. Among my options were FIR filters, Fast Fourier transform, and Analog circuit filters. I decided to go with the analog circuit filters because they do not take up any additional processing power, and they are fairly simple to understand.

Using a set of three of these filters (a high pass, band pass, and low pass) I create 3 different frequency bands. After the filtering, an Arduino Nano samples the signals from each filter and processes a sliding window of samples to create a smooth visual for the LED matrix. Finally, the Arduino Nano sends what lights to turn on over Serial to the Adafruit metro mini that controls the LED matrix.

Shortcomings:

Although I designed and tested all the circuitry before soldering it in to perf board, I did not design the entire project before creating it. This combined with the fact I didn’t know how to use power tools very well led to the frame of the boom box to be of poor build quality.

• External links:

Github:

Computational Complexity in a Nutshell:

• Start Date:

April 2017

• Completion Date:

May 2017

• Languages & Technologies Used:

Paint, Paint.net, Windows Live Movie Maker, YouTube, Python (yes, I actually did use this)

• Description:

Computational Complexity in a Nutshell is a video that gives a brief (and somewhat humorous) overview of computational complexity and related topics (P vs NP, ML & AI, Big O, etc).

Inspiration:

Our final project for Math (senior year of high school) was to make/give a presentation on a Math topic unrelated to calculus. This seemed kind of boring to me… so I settled on attempting to make a humorous video explaining computational complexity instead!

• Development & Challenges:

Halfway through drawing up the frames for my video I realized I was using the wrong aspect ratio!! However, instead of going back and editing 50 or so frames by hand, I decided to write a Python script to fix each of the images for me.

Shortcomings:

Of course, my drawings in the video were quite awful, and sometimes the timing of certain parts were a little off.

• External links:

Github: