hand coding round robin

part I ~ lecture

**1. COMPUTATION AS METAPHOR**

In leading up to earlier versions of this workshop, I had been thinking a lot about computation as metaphor. I think a lot of the things I was originally thinking about were kind of imploded by our past two weeks together so I am going to try and incorporate some of the things we have learned from each other and the teachers from this session. I thought a lot about computation as metaphor because throughout my education as a programmer I had learned that all computational systems are always and already built on top of metaphor in that nearly everything I did was somehow removed from what my computer was actually doing and so I was given sets of metaphors to remember and understand how to do those things. Computers require that programming, as we know it today, be a set of extremely abstract processes. So, when I’m coding something, I’m often coding on top of someone else’s code, in someone else’s software, on top of the software that makes up my operating system which is nearly permanently baked into the proprietary silicon chip of my Apple computer. Even the Terminal is a piece of software, a relic of teletypes and later screen-based computing that was text only, no folder icons or double clicks.

**2. MARK II**

Computers at their core consist of a complex series of sets of on and off switches, ones and zeroes. Everest Pipkin walked us through this idea, how a letter is actually a series of bits, and images are a series of complex letters and numbers, which are also just bits. This means that everything we do on our computers today needs to abstract away those complex patterns of numbers in order for us to use them in the way that we do. These abstractions allow us to do things like writing text in a file, without thinking about *how* or even *if* the computer will be able to do this. It is important to note that since the early days of computing computer time has always been more important than human time. When programmers were writing code for computers the size of rooms the time it took to run the program was really expensive, so people were always writing code that was most efficiently read by the computer itself not by the people who were making the programs. Hence the situation we’re in now where code is written in often illegible jargon and minified into unreadable blobs of text.

**3. DESK**

Back to the idea of computation as metaphor - The most obvious metaphor might be the desktop metaphor. The idea of a file inside a folder on a desktop...and there's a trashcan. I’m always wondering, when you put something in the trash, does it ever really go away? The office metaphor might be one which we may never get away from, at least as long as computers are as personal as they are now. But what is the file really? And what is the folder? Melanie unpacked some of these questions for us in their folder poetry class. All files are a series of numbers and the organization and re-making of those files can be a poetic act. A computer scientist once told me that folders aren't actually anything. They don't take up \*any\* space on your computer. Picking up on Kameelah Janan Rasheed’s concern with analogies, what if we didn’t think of computer desktops like the desks we sit at to use them and instead we thought of computer desktops as what they were, screens emitting light, and what happens then when we take the screen on its own terms? How did it come to be? If not for the baked in interface of files and folders what might we put on a screen emitting light?

**4. ILLUSION OF POWER**

This is all just to interrogate the fact that there has been a significant shift from physical programming, the act of physically adjusting switches or weaving code into core rope memory to abstracted programming, building software on top of software. Wendy Chun has a really amazing essay on the implications of this shift which I’ve included in the are.na channel. Since folders and files, which if we looked closely at them don’t resemble anything close to a piece of paper inside another folded piece of paper. I’m left wondering if computation is understood through metaphors does the way that we use our computers also require some kind of illusion of control over our computers.

**5. GRAMMERS**

I think having an illusion of control leads the people programming computers down a potentially dangerous path. Programmers, through the creation of complex processes and algorithmic systems go through so many hurdles trying to get the computer to do what they want it to do. When the computer finally obeys their commands, they are granted with this feeling of immense gratification and power. I’ve felt this feeling of gratification as a programmer after hours and hours of trying to debug something I wrote. It’s a cyclical command and response from you to your computer. My provocation here is not to call for an undoing of this cycle of command and response (although I’m interested in that) but to become aware of what is happening in between. So instead of humans controlling computers and computers controlling humans I am interested in how computers can be between, never over or under, the people that use them.

**6. VACUUM**

If the process of programming a smart device, for instance, is rooted in notions of command and control and illusion and power, and if all computation is metaphor, does this make for dangerous results? Does it perhaps mean that programmers get so caught up in the power dynamic between themselves and their computer that they lose sight of how what they're doing lives in the real world alongside real people?

**7. WRITING EMAILS**

This is a video of my grandma writing an email. It makes me wonder what it would be like if programmers had to press the keys of their keyboards so gently that it required them to move extremely slow and therefore with care? what would it mean if programmers had to manually move the switches to on and off positions in order to make a letter appear in an code they were writing? Or what would it be like if we could hear all of the computation happening inside our laptops as we typed out an email? Thinking about listening as a form of in-betweeness and like Mehrnaz mentioned in Ruha’s class listening as a form a resistance.

**8. CAMERONS WORLD OF GEOCITIES**

And so, all of these questions have led me to care deeply about hand coding. Hand coding is the process of coding that bridges the gap between you as the programmer and your computer as the programmed. In other words, it is a way to code which is not dependent on someone else’s software or framework or library, instead it is the process of writing in the inherent language of the browser. So I kind of lied here because your browser is software but hand coding in the software of the browser, like making folder poetry in the terminal, is closer to the metal of your computer then doing something like making a Wordpress website which is software on a big server sitting on top of your browser depending on fragile code written by tons and tons of people. Not that this is bad or good but if we are interested in questioning relationships between intimacy, transparency, surveillance, encoding and decoding, file systems and proprietaryness then doing something like hand coding might be a step towards new questions about these relationships. So your browser is software which translates HTML, CSS and JavaScript into web pages. Your browser is the performer of the program you are writing. The code you will write in this workshop will be very particular and you will write out each character slowly and by hand, there will very little copying or pasting. You will also do this on each other computers. The intention here is to perhaps erode this feeling of ultimate control that you have over your computer's system and to move carefully around someone else's.

Ok so I am in the camp of learning by doing so instead of me going on about what HTML and CSS and Javascript are we are just going to write code in these languages and then immeadiately see what they are doing in the browser. Of course if you have more specific questions about what HTML is, how CSS works, why use Javascript you can ask!