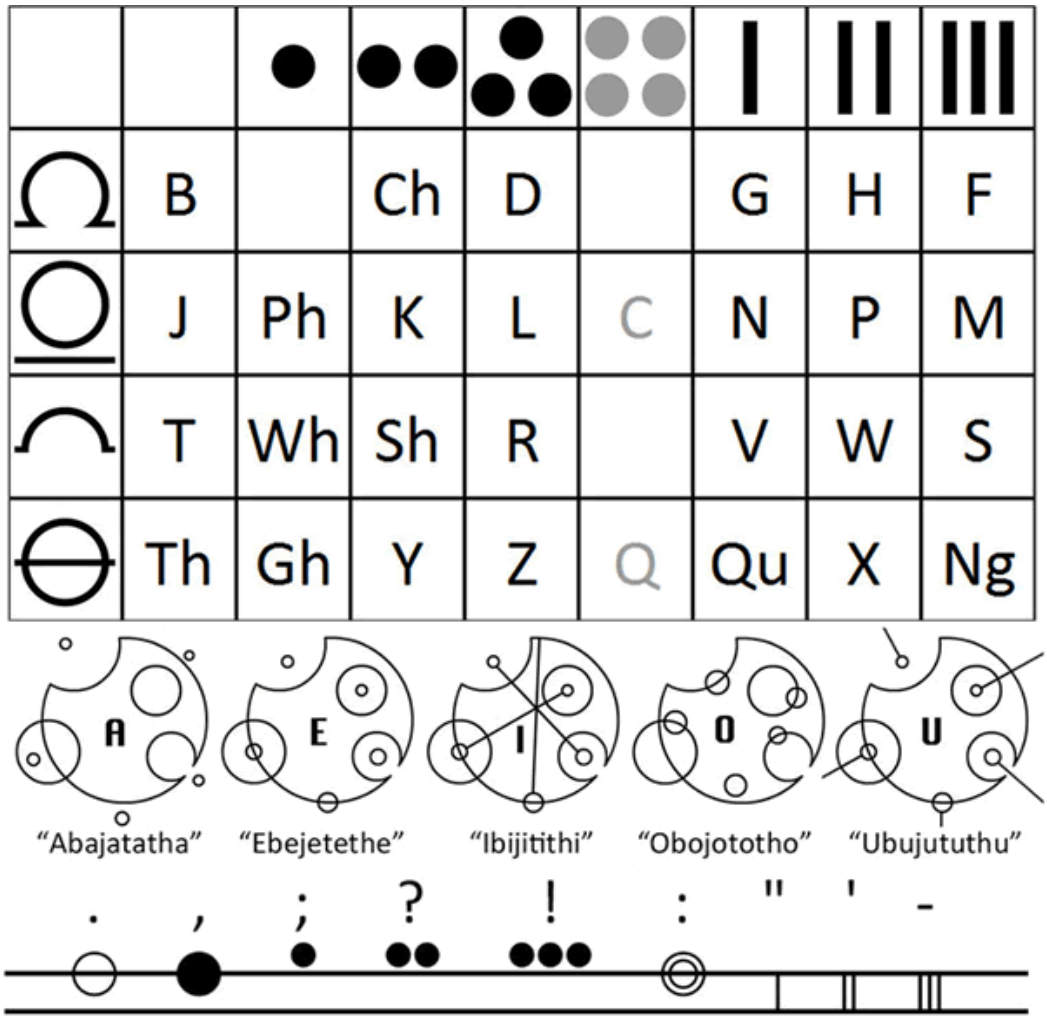
Circular Gallifreyan is a sci-fi alphabet inspired by Doctor Who that was created in 2011. As the name implies, the core structure is based off of circles, with different traits of the circles differentiating letters.

Alphabet Explanation:

Consonants in Circular Gallifreyan have one of four shapes, as shown in the figure below. They also can have up to three dots or three lines, giving a total of 28 different possible consonants, 27 of which are used. There is a small amount of variation between sources on what consonants exist because it is not a standardized language. The figure shows the variation I based my tool off of. I did not include the "c" and standalone "q" as they are not the way I am familiar with writing the alphabet, I don't think they even existed 10 years ago when I first discovered it. Double consonants ("ll", "dd", "bb", etc.) can be written either separately or as two concentric circles.

Vowels are written as smaller circles on their own or connected to the preceding consonant.

Lines coming from both consonants and vowels should be connected to other lines where possible.



(https://omniglot.com/conscripts/shermansgallifreyan.htm)

I hope this short explanation has made clear how much room for freedom and creativity exists in writing with this alphabet. This is something I wanted to actualize in a tool to help write the alphabet.

The Project:

I initially envisioned an editor that would give the user a base for the word they want to write and allow them to customize the word to they way they like. Specifically, it would have made the outlines of the consonants and vowels and put the dots inside of consonants that contain dots, but then allow the user the freedom to draw the lines as they wish. It would also have given the user the option for double letters to either be concentric or not and to either attach vowels to the preceding consonant (if any) or leave them freestanding. This, specifically the location of the lines, is the problem I have with a tool already existing online (https://adrian17.github.io/Gallifreyan/). I think this is a great tool, but I personally dislike the way that it draws the semicircle-shaped consonants (usually not a full half-circle, especially for longer words) and it never draws the lines in the exact way that I would have drawn them. For that reason, if I am working on an piece of art containing Circular Gallifreyan writing I can use the tool as a spellchecker, but it isn't of any use to me to do that actual writing of the word I desire. I wanted to create a tool that would be able to perform this function for me.

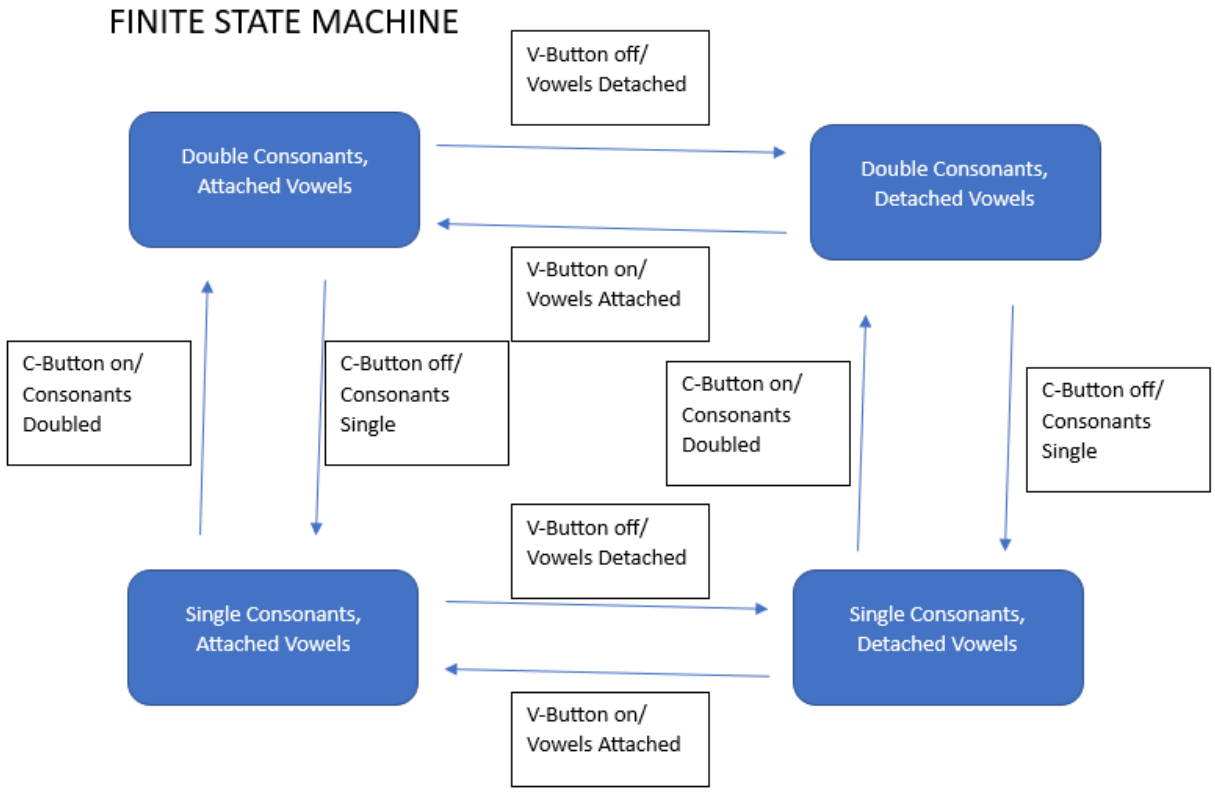
Creating NLRs:

I started out by creating NLRs for the project. This helps me to organize my thoughts about what I want and acts as a checklist for functionality. Were it not for my list of NLRs, I would have forgotten to implement the error message functionality. I also created a list of the rules of writing Circular Gallifreyan that I wanted the tool to follow. As I completed aspects of the project, I was able to mark them as completed, and when I finished I highlighted them all in green for those that were completed, yellow for those that were partially completed, and red for those that we not completed. All of my NLRs are listed out in the attached document entitled "Artifacts.pdf".

Finite State Machine:

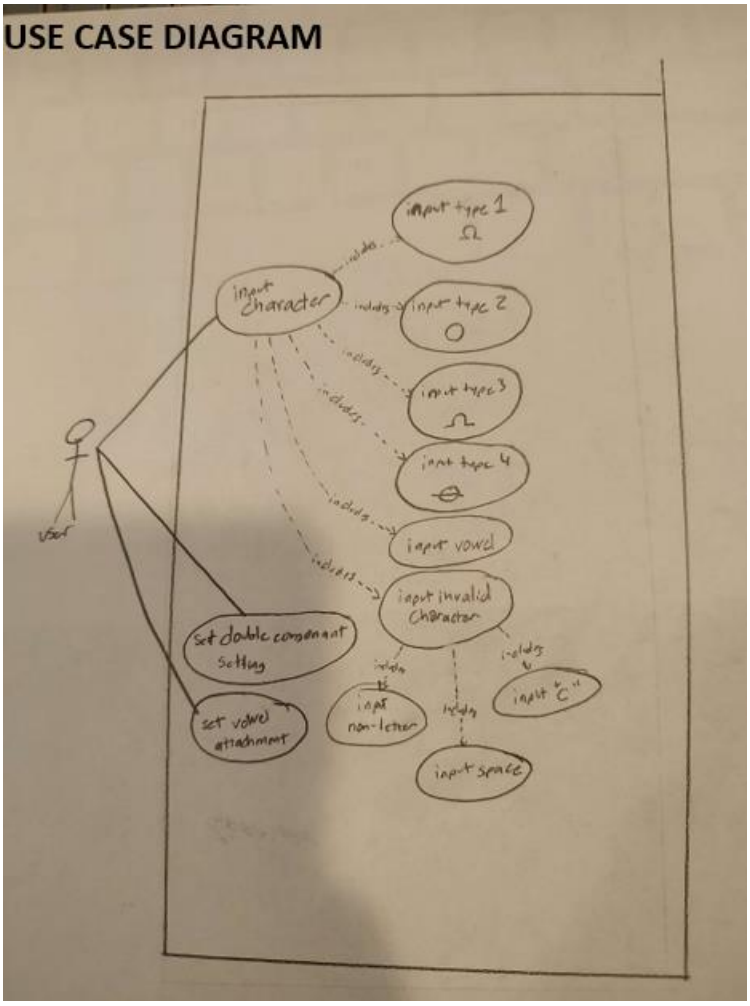
On the main screen of the application underneath the text input, there are two settings that can be switched on or off by the user. The first is the option for double consonants to be written as concentric circles and is defaulted to off, and the second is for vowels to be attached to the preceding consonant and is defaulted to on. This situation with two independently operating "switches" reminded me of the car headlight problem in class, so I created a finite state diagram to show how the application's

functionality should change depending on the “position” or the switches.

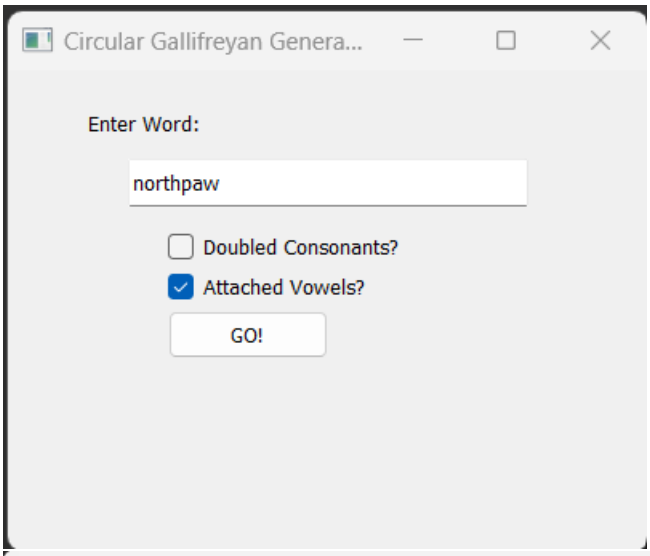


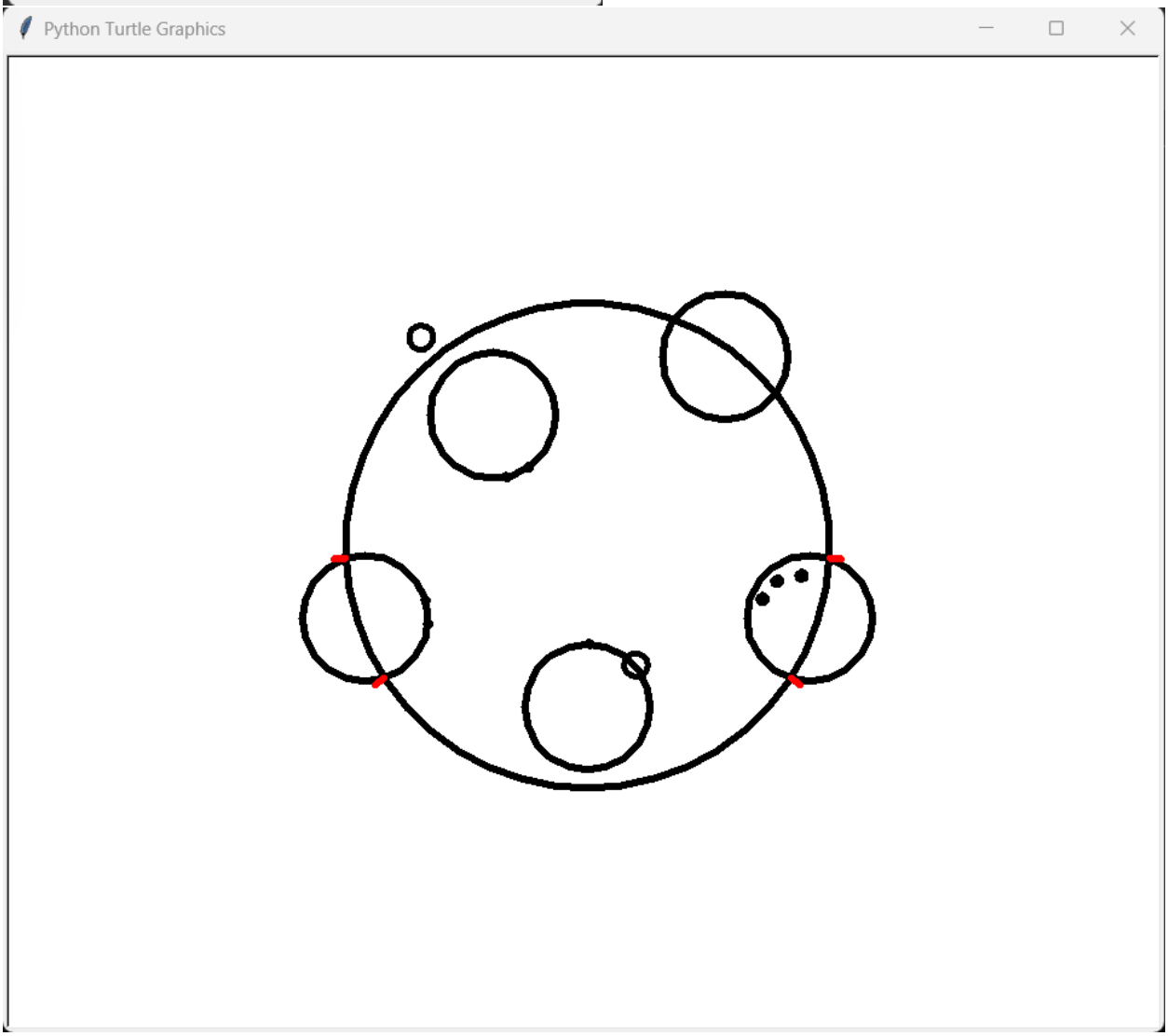
Use Case Diagram:

Initially I thought that a use case diagram would not be very useful to this project because there is only one actor performing one action (which I later realized was three due to the setting of the consonant doubling and vowel attachment options). However, I later realized that when I decompose the “input character” use case into separate use-cases for vowels, invalid inputs, and each shape of consonants, it was very useful for ensuring that I accounted for every type of input that could be received.



Final Product:


Circular Gallifreyan Genera...
Enter Word:
northpaw
Doubled Consonants?
Attached Vowels?
GO!


Python Turtle Graphics

I was unable to actualize my initial vision for the project, but I still like the tool that I created and will likely use it in the future. The failure of my project to meet all of the requirements fall into two categories of things that it does not do.

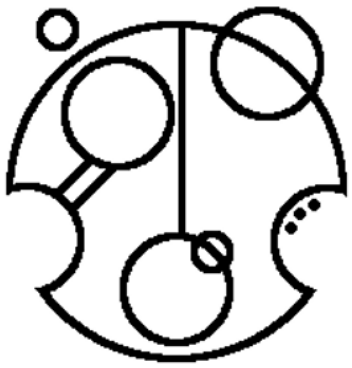
1: The program does not draw lines. I had wanted the program to supply the user with the points on the letters that need to be connected with lines and allow the user to draw their own lines in. Unfortunately, I was unable to implement this line-drawing function into the program. I did however put small markers on the letters needing lines so that the user could add the lines in themselves in another program.

2: The program does not properly draw type 1 (omega) or type 3 (half-circle) consonants. I opted to use the Turtle library for my program because I knew that I needed to draw a lot of circles and Turtle has a specific function for drawing circles. It worked fantastically for drawing the full-circles of type 2 and type 4 consonants, but I was unable to draw partial circles or erase the unneeded portions of the circles for types 1 and 3. In order to turn the output of this program into a usable word, I need to take a screenshot of it (I ran out of time to implement the ability to export the image as a .jpeg or .png) and import it into another program in to erase the unneeded parts. However, I did add little red lines to the letters to mark where the portion to be erased starts which makes the pieces that I erase by hand in another program more standardized.

Despite needing more editing in external programs, the output that this tool gives me is very useful. I tested how long it took me to write a word using this program as a starting point vs doing the entire thing manually, and my result was that writing the whole thing manually took me 17 minutes, but using the tool I was able to reduce that time to only 3 minutes. I'm sure I could further reduce that time with practice. Also, saying that it only took me 17 minutes to write the word manually is very generous, as I was able to re-use assets from previous words I had written. If I had truly done it from scratch, it could have easily taken me 25 minutes.

With Tool: ~3 minutes

Manually: ~17 minutes



There is not much difference in quality between the two versions of the word. When I first wrote them, I had a preference for the manual version, but the tool-assisted version is growing on me.

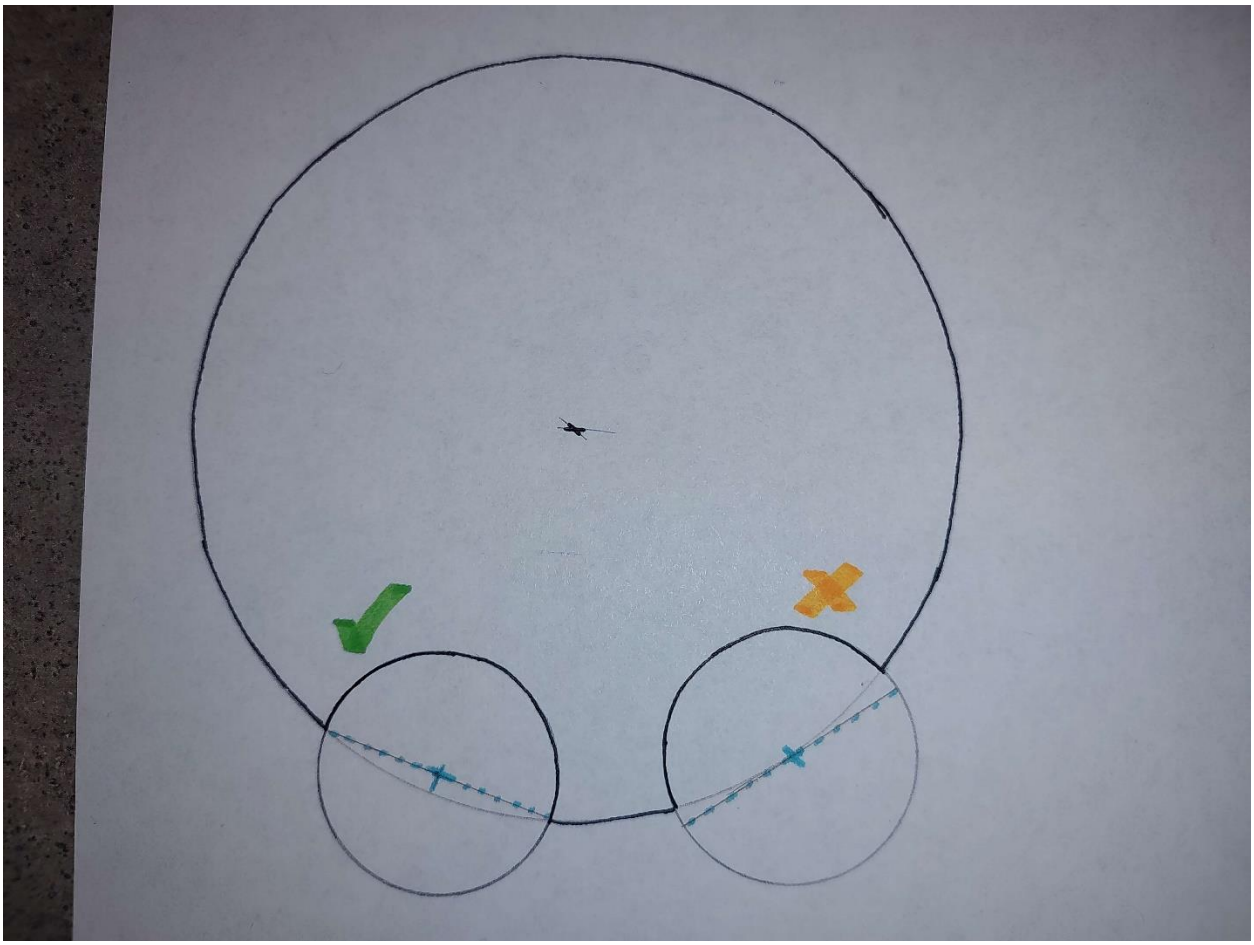
The future:

In the future, I would like to make two important changes to this project:

1. Refactoring. This project needs a lot of refactoring. Especially, there is a lot of re-used code in the drawing of the vowels and my lists of types of inputs (consonant types, diphthongs, vowels) would have been better made global so that I didn't have to re-declare them multiple times in different functions. I should have done more before turning it in, but I was too afraid of inadvertently breaking everything.
2. The positions of the vowels need to be fine-tuned. You cannot tell in the example word that I used, but my vowels are not quite centered where they are meant to be. I know exactly why this is and have a good idea of how to fix it, but simply ran out of time.
3. Re-write everything using something besides Turtle graphics because while Turtle is great for drawing circles, it does not easily draw the half-circles and omegas that I need

Additional Comments:

Something I am proud of doing well is the positioning of my type 3 (half-circle) and 3 (circle crossing the outer circle) consonants on the outer circle. If I simply used the outer radius of the large circle for the center of the consonant circle, the center of the circle would have been right on the outer circle, but would not have looked centered to me because I wanted the diameter of the smaller consonant circle to meet the outer word circle on both of its ends. In order to do this, I needed to reteach myself basic trigonometry in order to find the right value. These concepts were also useful in correctly placing the red dashes to mark where lines needed to be erased in another program.



Right: The easy way that I didn't want
Left: The more aesthetically pleasing way that required more math