Jeremy Paul
Michael Chen

Project Overview

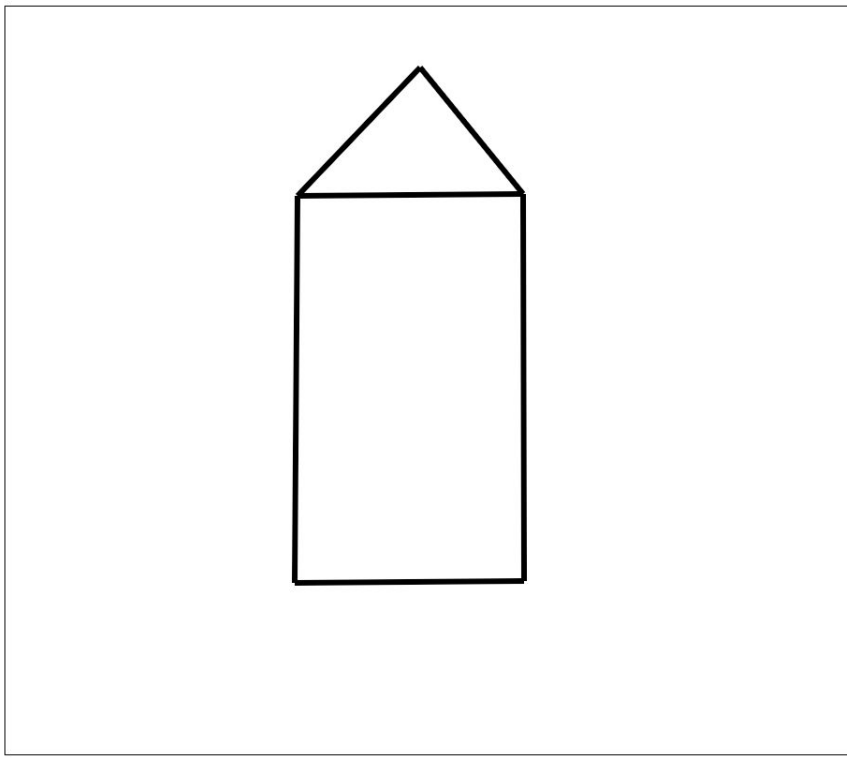
Our project aims to create a tool that is used to assist drawing through software. Generally, this will be used for design purposes as well as for casual interests from users. We have called this task “shape beautification” as we attempt to make the freehand drawings of our users look cleaner and more presentable. However, we do not wish to overwrite what the user’s original intentions and end up causing more frustration than assistance. Ideally, this would become a tool that aids the user in creating simple, yet clean looking designs and diagrams that can be manipulated for their purposes. To create our project we have used D3.js (https://www.npmjs.com/package/d3v4) , HTML, CSS, and a simple Python server.

We have created a number of interesting functionality for users to work with. First, we implemented an algorithm that registers a path that is drawn on the canvas and determines if it should be overwritten to a circle or a line. For the former, an algorithm is in place that takes in a path and calculates the average ‘X’ and ‘Y’ coordinates or the center of the circle. Then, it calculates the average distance from that center point to all other points in the path. If all the points occur within a given bound, then it assumes the intention is to draw a circle and overwrites the path. However, if this is not the case, it is passed to another function to see if the path is a line. The algorithm breaks the path into three segments and then calculates their individual slopes. If these slopes are within a certain range of each other, it is assumed that the path drawn is intended to be a line. If this is the case,the path is removed from the canvas and replaced with a clean line.

Additionally, we recognize that often times users are not drawing individual lines, but connecting them into shapes. As such, when we have drawn shapes using the methods

described above, we check to see if these shapes are intended to be connected. Everytime we draw a new line, it gets appended to an array of shapes. Then, we loop through the array and check the endpoints of existing lines to see if they lie within a given bounds of the end points of the new line we are drawing. This proximity check is calculated by taking the square root of the sum of the deltas of the x and y coordinates. If the lines touch at their ends within a certain proximity, we assume that the user likely wants to connect those lines and will group them into a singular object. This object is its own entity and will move together. Additionally, this functionality works with circles as well. To implement this, we took the shortest path from the center of a circle to the edge of a line. Then, we figured out the angle of that line relative to the circle using cos and sin and then checked the proximity of the point along the edge with the line, similar to line-to-line comparisons.

Figure 1. Lines connected together to form a house. This house moves as one object and were connected on-the-fly by our proximity algorithm.



This handles the majority of the drawing tools that we have implemented, but we also recognize that after the initial drawing, users may want to modify what they have made. In these cases, we support a number of other options to aid them in correcting mistakes or tweaking their designs. For starters, we have allowed these shapes to be selected so that they can drag them around the screen as needed to create more space or get them closer to other designs. As mentioned before, when lines are connected into their groupings they are treated as single entities, and are similarly moved as a single entity.

Additionally, there will be times when we have grouped elements that the user does not actually want to be grouped. In order to revert this, we support a tool that allows elements to be deselected from their groupings. By double clicking on an element, such as a line or circle, it will detach from the rest of the object and move independently of the others. The intent here is to assist users and give them as much control as possible. Inherently in software tools that aim to do things for users, there will be times that the software does things the user does not actually want. The idea is to minimize these occurrences, but we cannot eliminate them and therefore allow the users to handle those rarer contexts on their own to mitigate any frustrations that might arise.

Finally, by clicking on an item and pressing “delete”, we allow users to remove any shape from the canvas. This is important in case users have a series of mistakes and needs to clear the canvas.

In our initial proposal, we listed that we would have functionality for rotations and reflections. We believed that adding more novel functionality, such as the selection, disconnection, and deletion, would serve better. These new functionalities allow users to have more control over how objects are created and deleted. We wanted to add some more error recovery to the system since that was one of the main issues that current models had with

ambiguity. It seemed to be higher priority than adding some functionality that might not be used quite as often.

With all the above taken into consideration, we are quite proud of the tool and have had some fun developing, testing, and playing around with it.