How to Draw a Line

Simon Ever-Hale
Oberlin College
simoneverhale@gmail.com
simoneverhale.com

Abstract

When an artist draws a line on paper, the line is a function of the mental and physical state of the artist. I seek to bypass the second parameter, and compute the line produced by an arbitrary mental state. To do this, I have created an online algorithm that simulates the drawing of a line given the concentration and relaxation of the artist.

Introduction

The equation of a line is simple and well-defined. What is actually produced when a human tries to draw that line is not pinned down so easily. This has been emphasized in several works, including Sol LeWitt's 1972 *Wall Drawing #123* [1] and John Franzen's 2016 series, *Each Line One Breath* [2]. LeWitt's piece was done by dozens of different artists, taking turns drawing lines next to the previous in an attempt to copy it as closely as possible. Over many lines, the little inconsistencies in their drawing became amplified and the line morphed as a collective product of all the artists participating. Franzen's concept is similar; he draws a line and then attempts to draw a new line next to it that is identical to the previous. However, Franzen's work is done by one artist—himself—and is created entirely in one sitting in a period of intense focus. The goal is to produce a piece that uniquely captures the mindset of the artist at the time it was created. As a result, there is a great deal of variation between drawings in the series.

I wanted to create a way for any person to have the experience of creating something that takes so much time and effort and directly reflects their mental state. I have implemented this concept by simulating the same process used in LeWitt and Franzen's works: a person drawing a line using only the previous line as guidance.

Algorithm

Suppose you are drawing a line, trying to copy the line next to it, and you are at some position p. To find the next position that the pen will move to, you must take a few factors into account. These include the angle of the previous line, the desired distance away from the previous line, the current angle and the desired speed. Additionally, as a substitute for shakiness in drawing, I've introduced a third parameter, which is a small random angle added to the current angle.

Formally, let $L = \{l_0, \ldots, l_n\}$ with each $l_i \in \mathbb{R}^2$ represent the previous line as a discrete series of points. The pen is currently at position $p \in \mathbb{R}^2$ moving at an angle α with speed s (call the velocity vector v), and the point on the previous line which it will pass next is l_{next}. The steps for computing the next position of the pen are as follows:

1. Add noise. Set $\alpha = \alpha + U$ where U is a uniform random variable in the range $[-c, c]$ and c is a predefined constant.
2. Correct for previous angle. Compute the angle β of the previous line at the current position using a weighted average of the vectors between consecutive points from $l_{i_{\text{min}}}$ to $l_{i_{\text{max}}}$ so that angles closer to

l_{next} have a higher weight. So the weight of the vector between l_i and l_{i+1} is $\frac{1}{|i-\text{next}|+.25}$. Create a vector with angle β and magnitude equal to s, interpolate with v by some amount t_2 and normalize to magnitude s to get the new velocity.

3. Correct for distance from previous line. Compute the target point $p_{\text{target}} = l_{\text{next}} + \langle d, 0 \rangle$ where d is the target horizontal distance from the previous line. Linearly interpolate between the velocity and the vector pointing from p to p_{target} by some amount t_3, then normalize to s to get new velocity v.
4. Add the velocity to the current position to get the next position, p_{next} and determine the new l_{next}, the next point on the previous line such that l_{next} lies ahead of p_{next}.
5. Draw a line from p to p_{next}.
6. Although p_{next} is the point that was just drawn, it's not necessarily the point the artist would copy when drawing the next line. In the case of p_{next} being behind the previous line, the artist would follow whichever was further along the x-axis. So we must compute the point p_{yprev} on the previous line with the same y-value, and add whichever of p_{yprev} and p_{next} has a larger x-value to the next line, L_{next}.
7. If the current line is complete, set $p = L_{\text{next}} + \langle d, 0 \rangle$, v to the velocity of L_{next} at index 0, $L = L_{\text{next}}$, and $L_{\text{next}} = \{\}$.

Parameters

Many of the constants referenced in the algorithm above can be linked to the concentration and relaxation of the artist. Intuitively, higher concentration should result in a line that more closely represent the previous, as when an artist is concentrating harder they tend to move slower and correct for errors more quickly. Higher meditation should result in a line that is smoother, as when an artist is more calm they tend to have less shakiness in their lines. To produce this effect, I've established the following relationships:

- The range of the random angle, c, decreases as meditation increases.
- The maximum index used to calculate the slope of the line, i_{max}, decreases as concentration increases. This simulates the "tunnel vision" that occurs when concentrating hard, and results in the line being more sensitive to smaller changes in the previous line.
- The interpolation constants in steps 2 and 3, t_2 and t_3, increase as concentration increases, resulting in a quicker response to changes in the previous line.
- The speed s decreases as concentration increases. This simulates moving slower and more carefully, with more time to correct for mistakes.

Examples

I purchased an $80 NeuroSky EEG headset that measures eletrical brain activity, and wrote a sketch using Processing that kept track of two of the data streams output by the headset: "attention" and "meditation". Since I had a measurement of my own concentration and relaxation in real-time, I was able to create a visualization of my mental state over the span of about 50 seconds. Even with an extremely slow refresh rate (1 second), I was still able to create images that made sense in how they reflected what I was doing at the time.

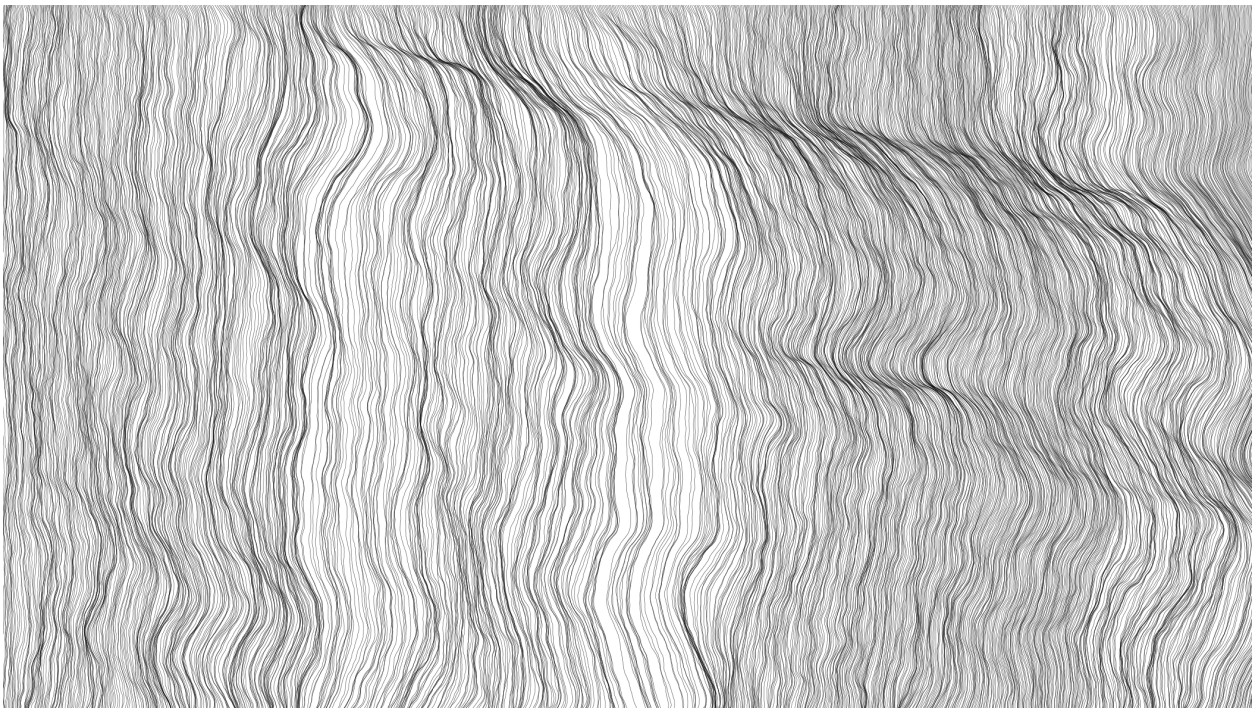

Figure 1 : *Playing a mindless iPhone game.*

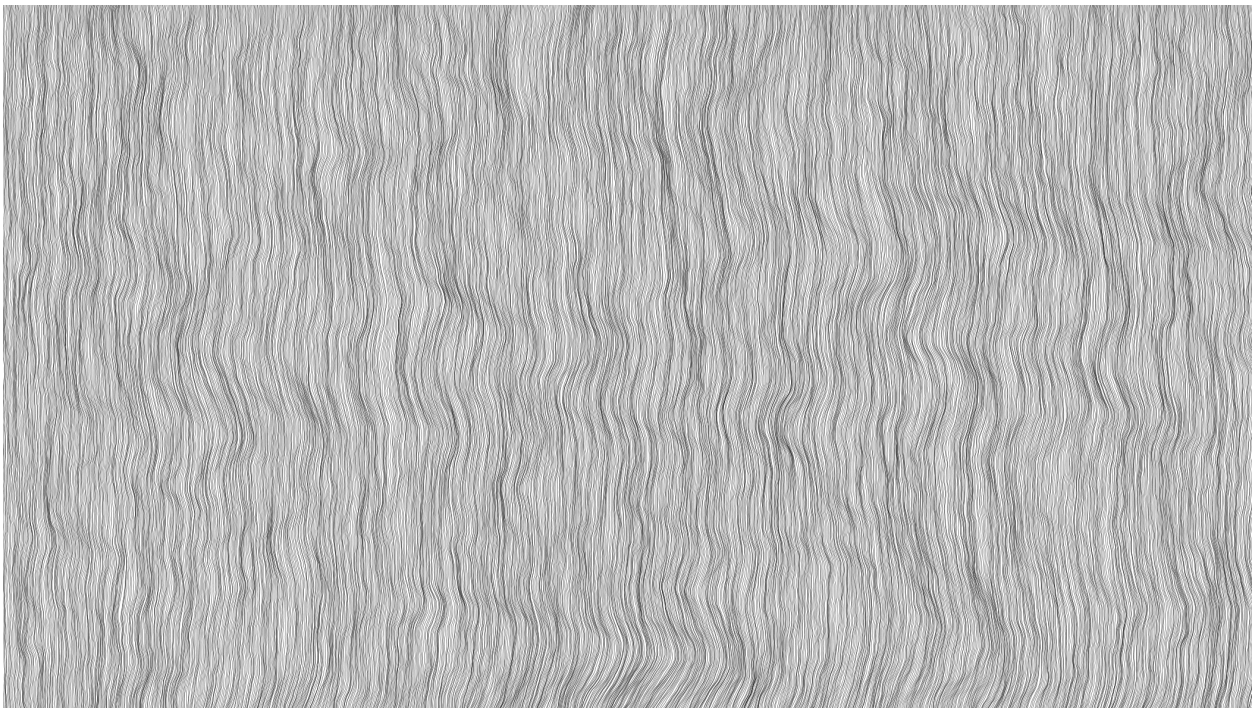

Figure 2 : *Drawing detailed shading by hand*

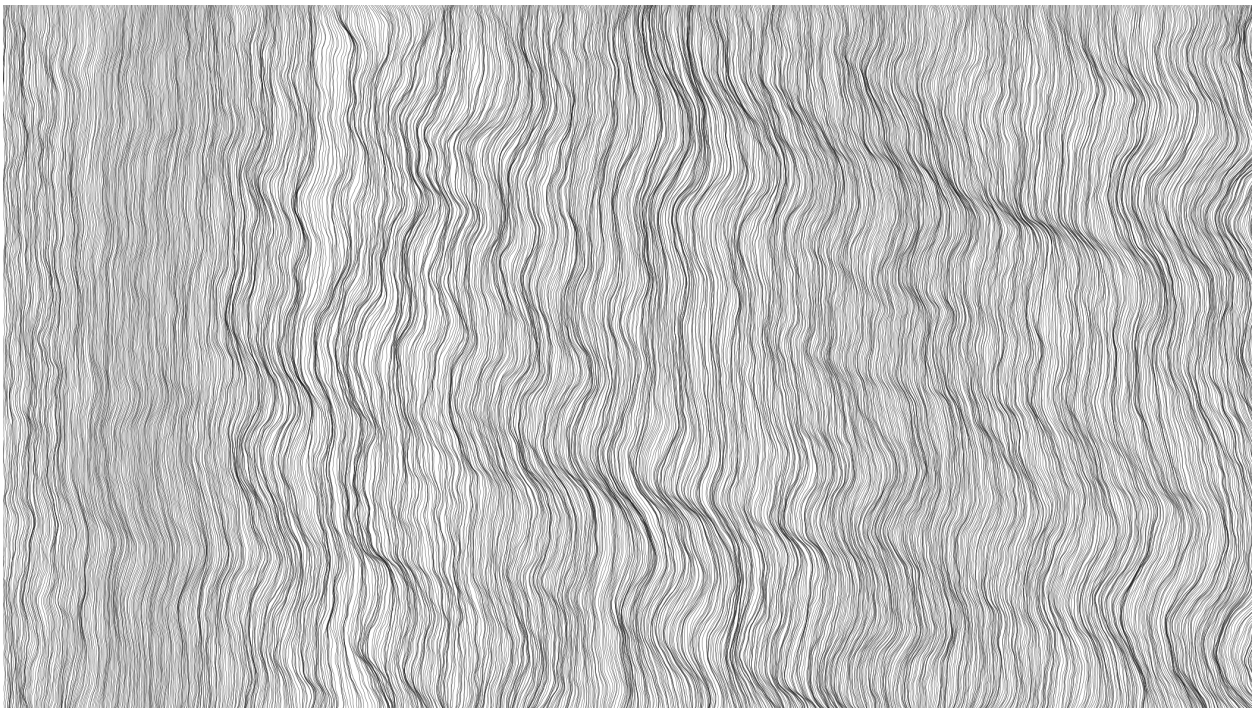

Figure 3 : *Attemping to take a baseline by sitting, moving around a little, and drinking water.*

Notice that in Figure 1 and Figure 3, the drawing feels like it's slowly sliding downwards from left to right, but in Figure 2 the patterns are moving horizontally. Due to the lower concentration in Figure 1 and Figure 3, the reactions to changes in the line are slower. I chose to draw lines from top to bottom, so the bumps become lower and lower on the page over time. This phenomenon shows up when doing this type of drawing by hand as well.

Future Work

Since this algorithm produces an approximation of the line that would be drawn by a person at any given time, the most exciting application would be in creating artistic tools for people with disabilities. Physical ability and even typical mental ability are both optional and unnecessary for this algorithm to work. Any mental state is a valid one and will still produce output.

References

[1] S. LeWitt, *Wall Drawing #123: Copied Lines*, 1972, graphite on wall.

[2] J. Franzen, *Each Line One Breath*, `http://www.johnfranzen.com/each-line-one-breath.html` (as of Feb. 27, 2017).