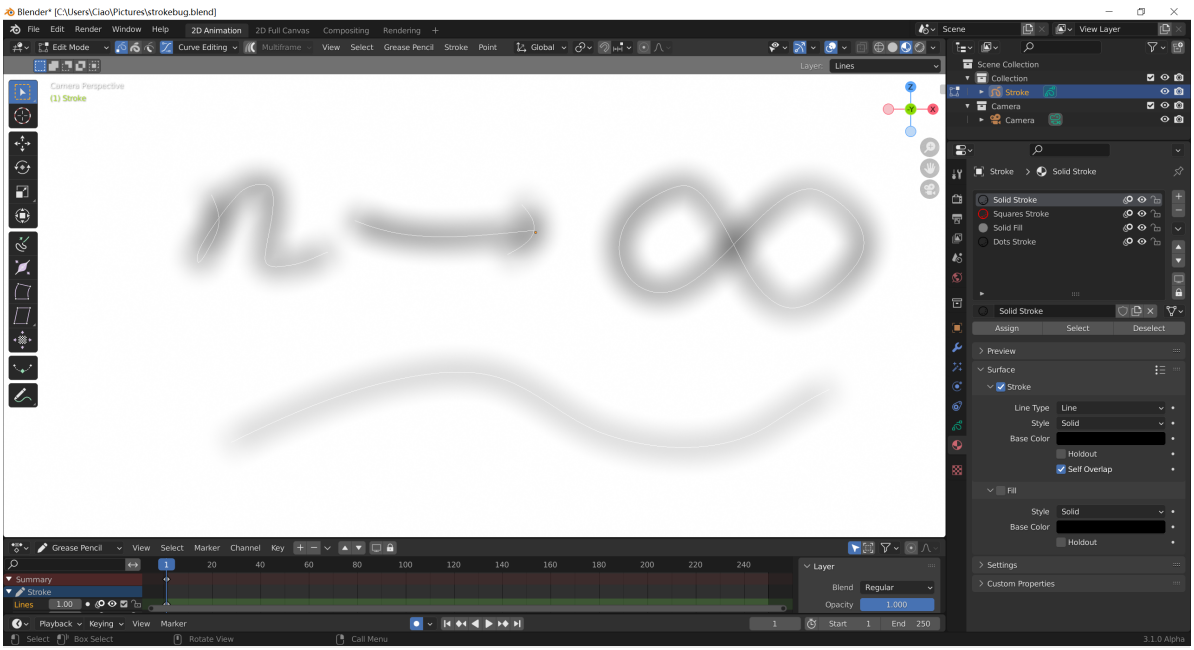
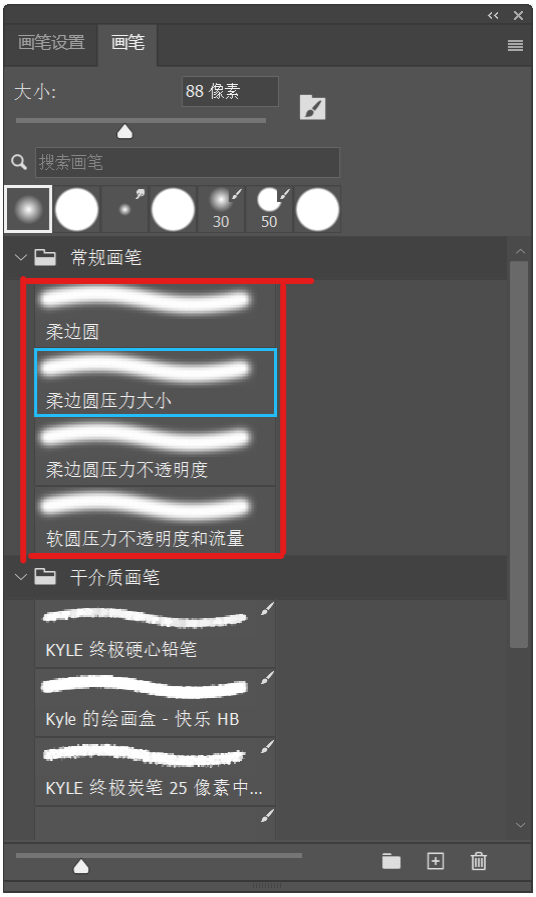
Continuous Airbrush Stroke Rendering

by ShenCiao demo version

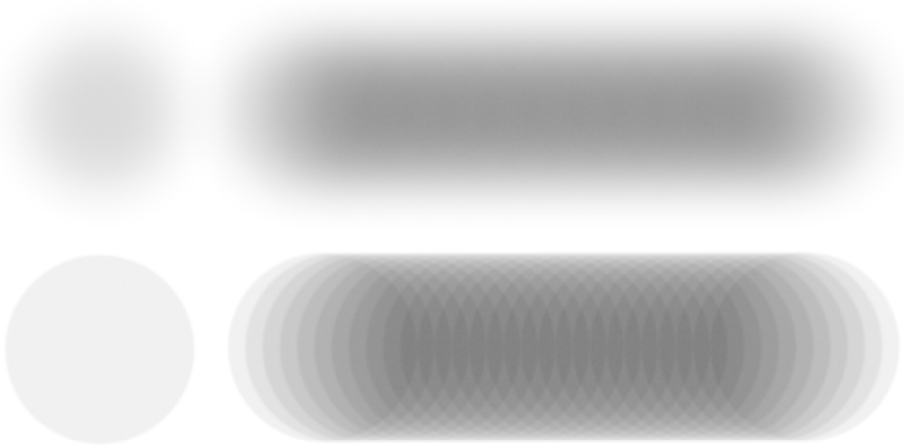


What is a digital airbrush?



There is an alpha value gradient from the middle to the edge of the airbrush stroke. Artist usually uses the brush to draw lights and shadows in digital paint software.

How does traditional (discrete) airbrush stroke form?



Artist chooses a single dot with nonzero alpha value. When moving the cursor on the canvas with the mouse click, the software puts dots evenly along the stroke and blends those dots normally.

The RGB is invariant for pixels inside dots but the alpha value variant with distance to the center. I denote it as $\alpha_{dot}(r)$. The alpha value always falls to zero at the rim of the dots. Those who fall slowly and smoothly is soft airbrush (upper one). Those who fall precipitously at the edge is hard airbrush (lower one).

The defect and solution

In paint software with "vector" canvas (e.g. Adobe Illustrator, Inkscape). We always make sure that every single stroke is infinitely zoomable. But the traditional method would get the perceivable discontinuity after zooming in. The artifact will look like the hard airbrush above.

If we can make the distance between dots infinite small, the greyscale of dots infinite small, and put infinite numbers of dots along the stroke, the discontinuity will be eliminated. So we need calculus to formalize the problem (I wish the process of mathematical proof wouldn't be in a mess since I graduate from psychology).

How discrete airbrush strokes formed

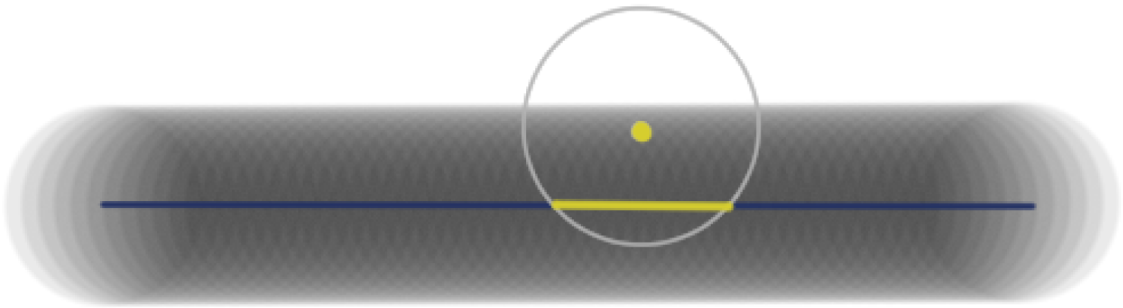
Blend function

If we have two pixels with the same RGB and different in alpha. They overlap each other and form a single pixel. What is the color of the newly formed pixel? (RGBA are normalized to 0 and 1) Based on the normal blend function $C = C_{dst} * (1 - \alpha_{src}) + C_{src} * \alpha_{src}$, we can easily prove that RGB stays the same and $\alpha = 1 - (1 - \alpha_{src}) * (1 - \alpha_{dst})$.

If there are n dots, it comes to:

$\alpha = 1 - \prod_{i=1}^{n}(1 - \alpha_i)$

Formation of a straight stroke



As I mentioned above, dots are placed along the middle of a stroke (blue line). For a randomly selected pixel (yellow) inside stroke, draw a circle with the radius same as dots. The intersection between the circle and middle of the stroke is the yellow line segment. Obviously, only dots placed inside the line segment can affect the color of this yellow pixel.

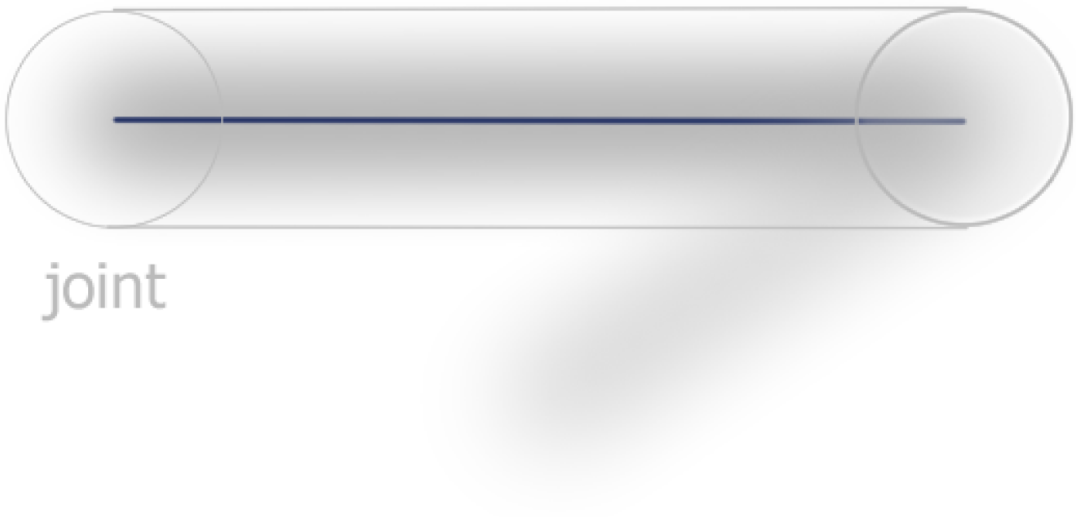
If randomly selected pixels are in the same distance with the blue line (ignore pixel around endpoint temporarily), the length of the yellow line stays invariant and we get exactly the same colors. So we get a "falloff" visual effect, the color only variant with the distance to the middle of a stroke.

Pixels inside dots are the same in RGB but different in alpha. According to the blend function, only the alpha channel is variant inside the stroke. So we can define a falloff function to represent the alpha value of the current point. I denote the **falloff function** as $\alpha_{fo}(d)$.

If we know about the distance between the dots and the falloff function of a single dot $\alpha_{dot}(r)$, we can calculate every single pixel's color with the n dots blend function. But when n comes to infinity, it's a little bit different.

Theory about continuous airbrush stroke

Capsule method



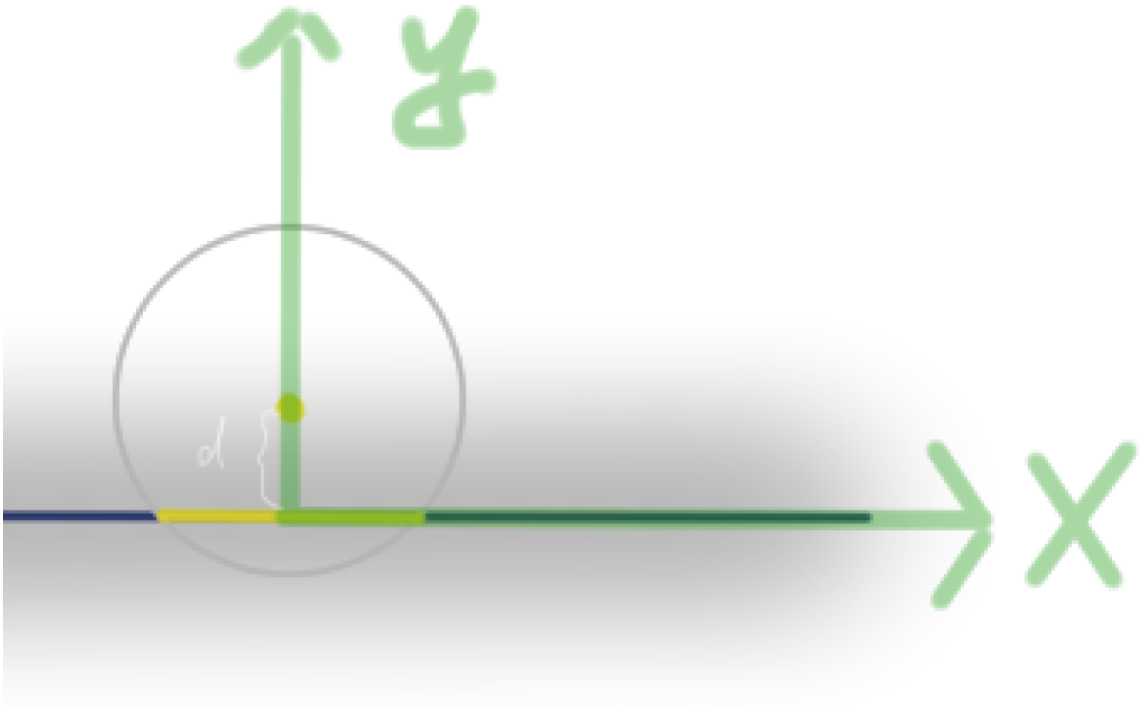
For each segment of a stroke, we render a "capsule" along the stroke. The diameter of the circles that center at the endpoint is the thickness of the stroke. I name the circle "joint". We get the final stroke by blending those capsules. We need to calculate the alpha value of the pixels inside each individual capsules.

Alpha density $f(r)$

We have to define *alpha density*. If you've ever learned about probability density, the idea is exactly the same. The alpha value of a pixel is no more determined by several dots, but by line segments. The $\alpha_{segment}(r)$ is equal to **alpha density function** $f(r)$ times the length of the segment as the length of the segment approaches infinitely small (such a crappy definition, please somebody help me on this).

Calculate $\alpha_{fo}(d)$ of continuous stroke

Let's set up a coordinate system first:



.

We place the x-axis along the middle of the stroke and the y-axis pass through the current pixel and let the radius of the circle (half thickness of the stroke) equal 1. We cut the yellow line into n segments, blend them together, and get the alpha value of the current pixel. (Ignore all the math notations if they blow up your mind at first glance).

$$\alpha_{fo}(d) = 1 - \prod_{i=1}^{n}(1 - \frac{2\sqrt{1-d^2}}{n}f(r_i))$$

As n approach infinity $\lim_{n\to\infty}$

$\prod_{i=1}^{n}(1 - \frac{2\sqrt{1-d^2}}{n}f(r_i)) = e^{g(d)}$ (It's an advanced version of $(1+\frac{x}{n})^n = e^x$)

within which $g(d) = -\sum_{i=1}^{n}\frac{2\sqrt{1-d^2}}{n}f(r_i)$

And according to the definition of definite integrals and symmetry of yellow segment.

$$-g(d) = \int_0^{\sqrt{1-d^2}} f(r)\mathrm{d}x + \int_{-\sqrt{1-d^2}}^0 f(r)\mathrm{d}x = 2\int_0^{\sqrt{1-d^2}} f(r)\mathrm{d}x$$

within which $r = \sqrt{x^2+d^2}$. And eventually:

$$\alpha_{fo}(d) = 1 - \exp\left(-2\int_0^{\sqrt{1-d^2}} f(r)\mathrm{d}x\right)$$

For the pixels inside joints, we only need to change the range of integral of $g(d)$.

That's it! Given an alpha density function $f(r)$, we could find out any pixel's alpha value inside the capsule. By connecting capsules consecutively and blending them together to form a polyline stroke, we can approximate any kind of curve.

But practically I never render a stroke derived from alpha density function. Here is the reason.

Why not derive $\alpha_{fo}(d)$ from alpha density

inefficient

I've derived the $\alpha_{fo}(d)$ with a quite simple alpha density function $f(r) = 1 - r$, which means alpha density falls to zero from one linearly. Here is the result without normalization:

$$1 - \exp\left(-2\sqrt{-d^2+1} + \sqrt{-d^2+1} + d^2 \ln\left(\frac{\left|\sqrt{-d^2+1}+1\right|}{|d|}\right)\right)$$

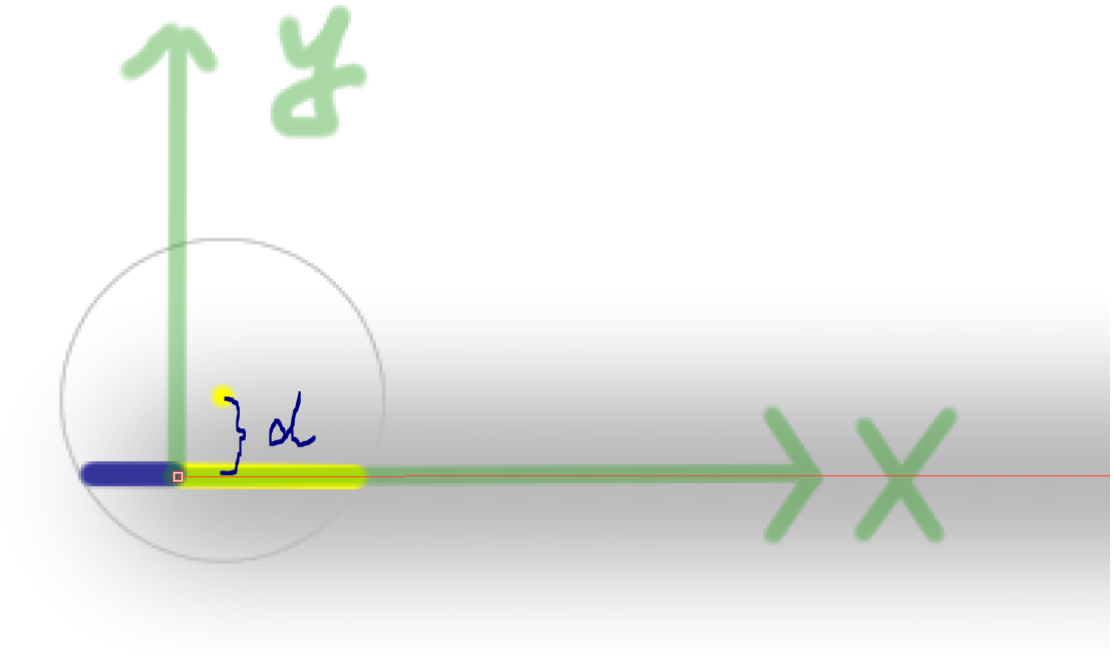
This is definitely a serious waste of GPU to compute.

uncontrollable

Sometimes artists want to control the $\alpha_{fo}(d)$ manually. But the visual effect could be quite different from the image of alpha density to the final strokes since calculus. We need a context that we know nothing about alpha density but only $\alpha_{fo}(d)$.

Joint pixel alpha value approximation

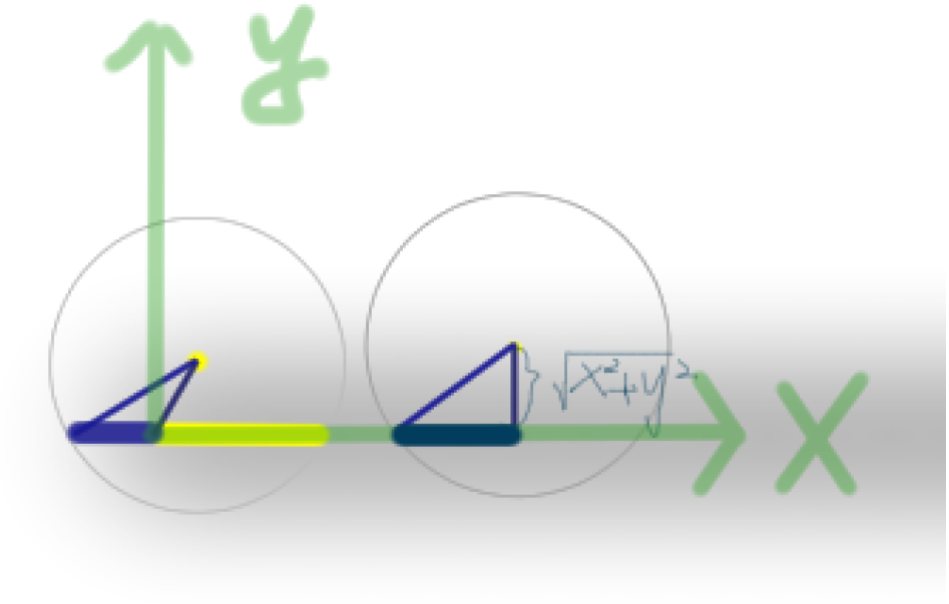
Now we let an artist define the falloff function $\alpha_{fo}(d)$. It's not hard to figure out that only pixels inside the joint do not comply with the falloff function. I did not find the method for us to calculate $f(r)$ (and maybe there are infinite possibilities for $f(r)$). We need a method to describe and approximate the alpha value.



I reset up the coordinate by placing origin at the endpoint. We select a random pixel (yellow) inside the joint and draw the unit circle. From a geometry point of view, the value of $\alpha_{fo}(d)$ is "blended" by alpha from the blue segment and the yellow segment. But for the pixel inside the joint, only the yellow segment actually contributes to the value. We need to "eliminate" the alpha from the blue segment. Then how to "eliminate" the blue α_{blue} in math? It's not hard to derive from the blend function or from the $g(d)$ that $\alpha_{yellow} = 1 - (1 - \alpha_{fo})/(1 - \alpha_{blue})$.

How to approximate $1-\alpha_{blue}$

I tried several ways and have found one of the best ways. $1-\alpha_{blue} \approx \sqrt{1-\alpha_{fo}(l)}$ within which $l=\sqrt{x_{yellow}^2+y_{yellow}^2}$ is the distance from the yellow point to the origin. I will derive it later on but let's try to understand it from a geometry point of view first.



The alpha value of dots that construct the segment only relates to the distance to the center of the dots. If we find a pixel that has the same starting point distance and ending point distance as yellow, it could be an approximation. As the diagram above shows, we are using the blue segment on the right to approximate the blue segment on the right.

We can evaluate the error with the function of α. Set the position of the yellow point (x_{real}, y_{real}), the approximate point (x_{appro}, y_{appro}). Change integral term from x into r. We can derive that:

$$1-\alpha_{real}=\exp\left(\int_l^1 \frac{r}{\sqrt{r^2-y_{real}^2}}f(r)\mathrm{d}r\right)$$

$$1-\alpha_{appro}=e^{\frac{g(l)}{2}}=\exp\left(\int_l^1 \frac{r}{\sqrt{r^2-y_{appro}^2}}f(r)\mathrm{d}r\right)=\exp\left(\int_l^1 \frac{r}{\sqrt{r^2-y_{real}^2-x_{real}^2}}f(r)\mathrm{d}r\right)$$

$$y_{appro}=l=\sqrt{x_{real}^2+y_{real}^2}$$

There is only one term x_{real}^2 different from the real one, which means that **larger x_{real}^2, worse approximation.** So in practice, I let $1-\alpha_{appro}=e^{g(l)*\frac{(1-|x_{real}|)}{2}}$ for a better approximation.

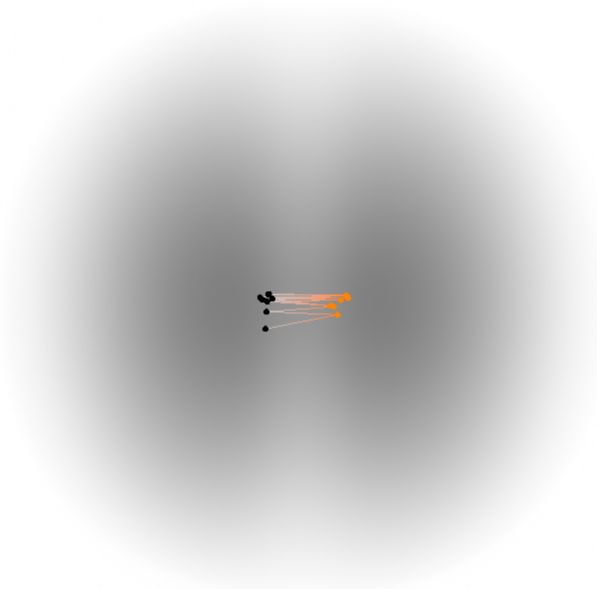
Practical method and issues

Eliminate exponential

We could totally eliminate exponential terms for better rendering performance. Just implement the function $\alpha_{yellow}=1-(1-\alpha_{fo})/(1-\alpha_{blue})$ in code.

Stroke with dense vertices

If a polyline is dense in vertices, the distance between two endpoints is small. A pixel could be inside both joints, we just need to implement $\alpha_{yellow} = 1 - (1 - \alpha_{fo})/(1 - \alpha_{blue\_left})/(1 - \alpha_{blue\_right})$. And it's quite robust for polyline with a high density of vertices. There does exist an ill case when placing vertices too close and stacked on each other by two groups. We get the shape from a top view of a brain. But it's really rare to happen.



Another alpha channel

If there are already existing alpha values and we need to apply our falloff alpha. We should time the existing alpha value into the falloff function before calculating the pixel alpha inside the joint. If not, you would get artifacts like this:

Maybe you have noticed the black dot sitting on the vertices. These are the artifacts caused by multiplying existing alpha values after calculating the pixel alpha inside the joint (and I spent three days debugging this).

Starting and ending point of strokes

The effect endpoint may be kind of wired at first glance. But it's not necessary to deal with since the artists tend to reduce the overall alpha value like the lower stroke. The effect is unnoticeable.