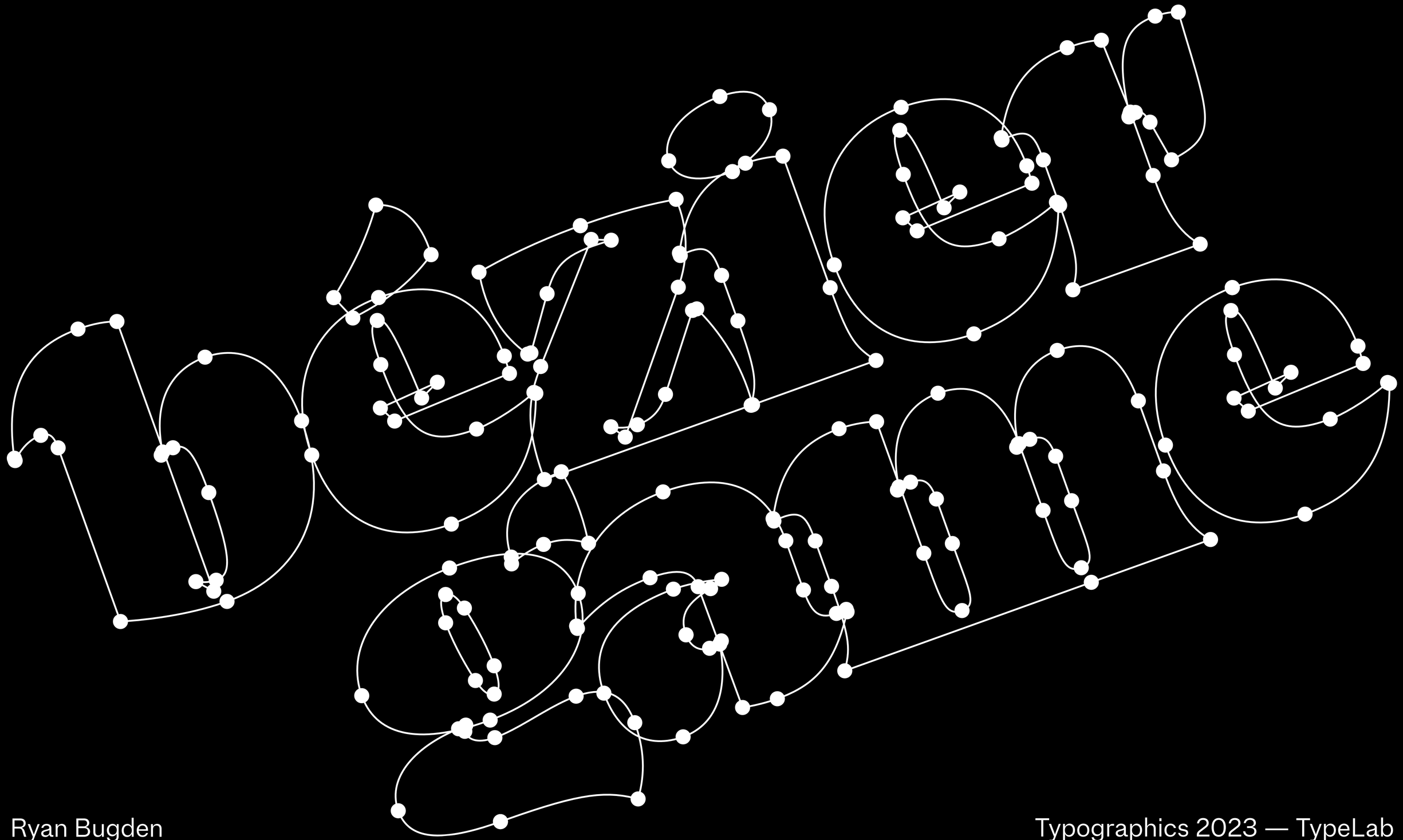


Ryan Bugden

Typographics 2023 — TypeLab

TypeLab 2017
TypeLab 2018
KABK Open Day 2019
TypeLab 2023

Quick background
Math demo
Tips
Game

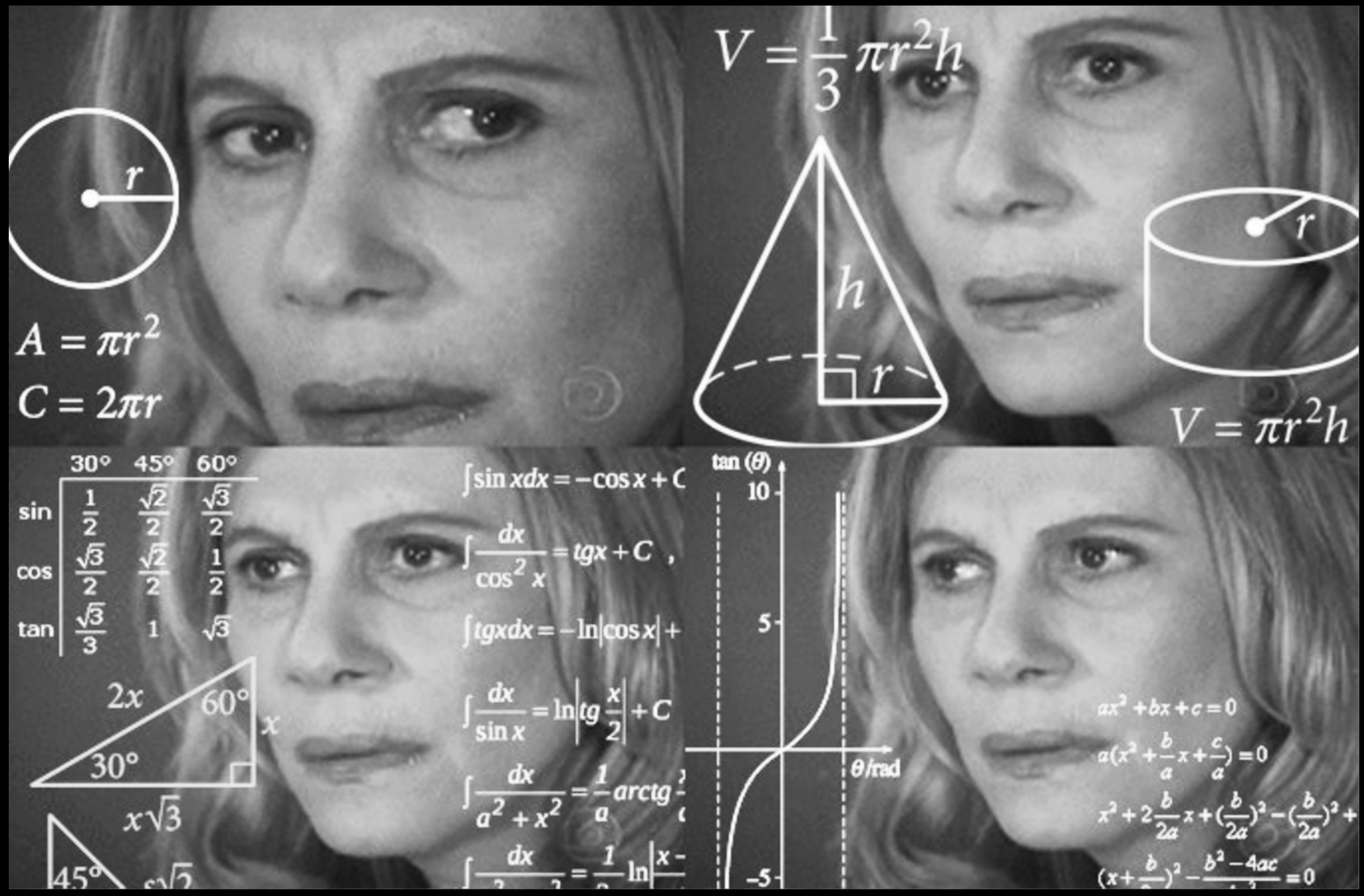
Quick background
Math demo
Tips
Game

Quick background

Math demo

Tips

Game


r
$A = \pi r^2$
$C = 2\pi r$
$V = \frac{1}{3}\pi r^2 h$
h
r
r
$V = \pi r^2 h$
30° 45° 60°
sin $\frac{1}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{\sqrt{3}}{2}$
cos $\frac{\sqrt{3}}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{1}{2}$
tan $\frac{\sqrt{3}}{3}$ 1 $\sqrt{3}$
2x
60°
x
30°
$x\sqrt{3}$
45°
$\int \sin x dx = -\cos x + C$
$\int \frac{dx}{\cos^2 x} = tgx + C$,
$\int tgx dx = -\ln|\cos x| +$
$\int \frac{dx}{\sin x} = \ln\left|tg\frac{x}{2}\right| + C$
$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} arctg$
tan (θ)
10
5
-5
θ/rad
$ax^2 + bx + c = 0$
$a(x^2 + \frac{b}{a}x + \frac{c}{a}) = 0$
$x^2 + 2\frac{b}{2a}x + (\frac{b}{2a})^2 - (\frac{b}{2a})^2 +$
$(x + \frac{b}{2a})^2 - \frac{b^2 - 4ac}{4a^2} = 0$

DrawBot

DrawBot Documentation

DrawBot is a powerful, free application for MacOSX that invites you to write simple Python scripts to generate two-dimensional graphics. The builtin graphics primitives support rectangles, ovals, (bezier) paths, polygons, text objects and transparency.

Education

Quick background

Math demo

Tips

Game

Now we’re staying
in cubic land.

Look out for this:

IMPORTANT TIP FOR THE GAME!

1

Use as many points as you need.
No more, no less.

IMPORTANT TIP FOR THE GAME!

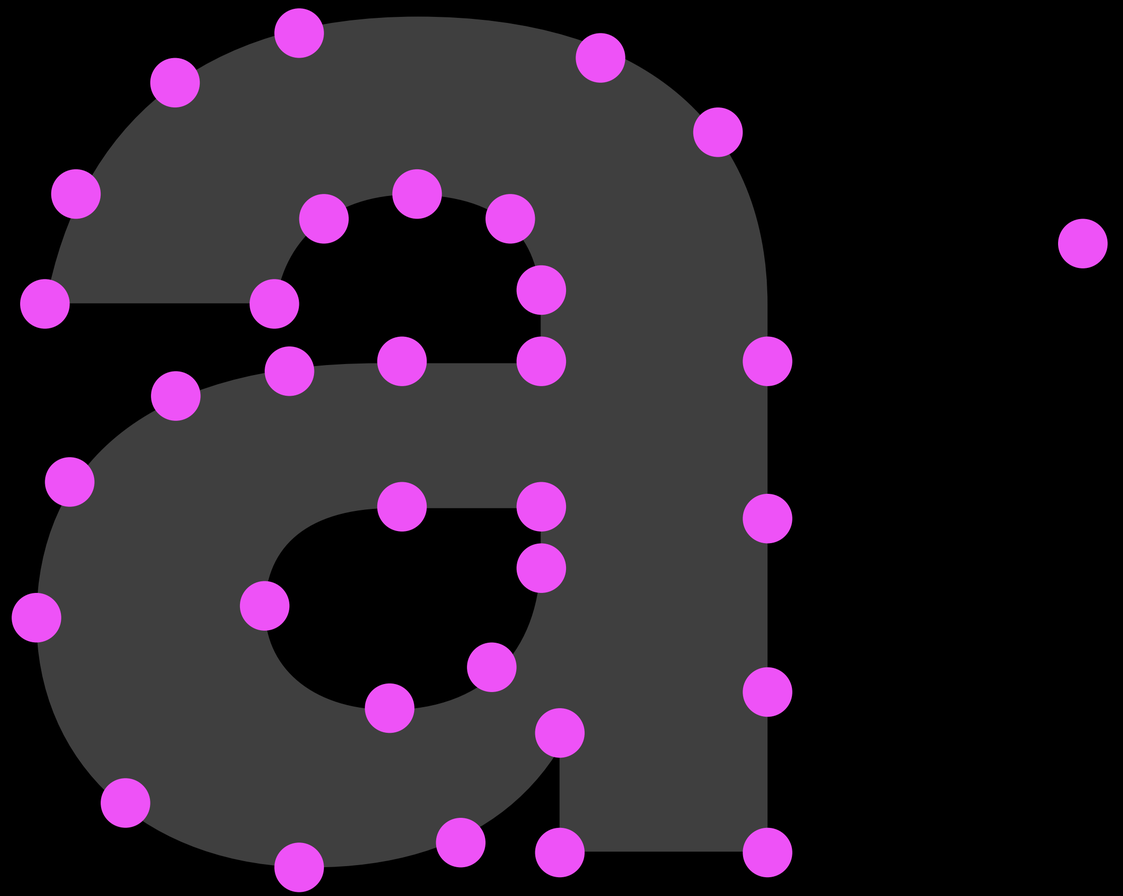
1

Use as many points as you need.
No more, no less.

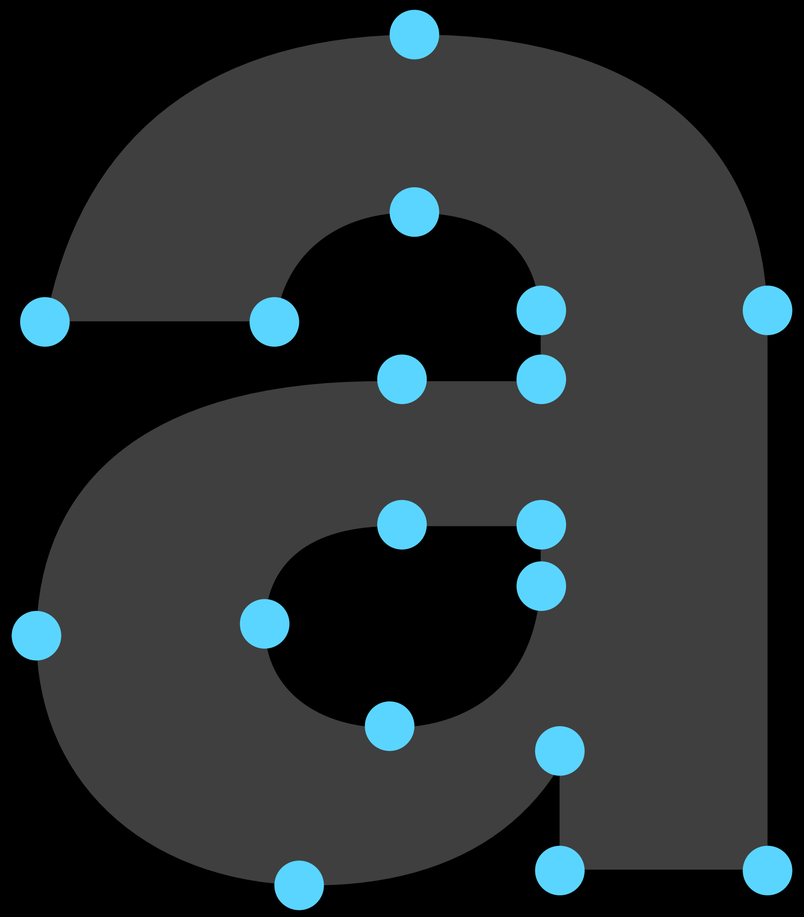
K.I.S.S.
~~(Keep It Simple, Stupid)~~
(Keep It Simple, Smartperson)

IMPORTANT TIP FOR THE GAME!

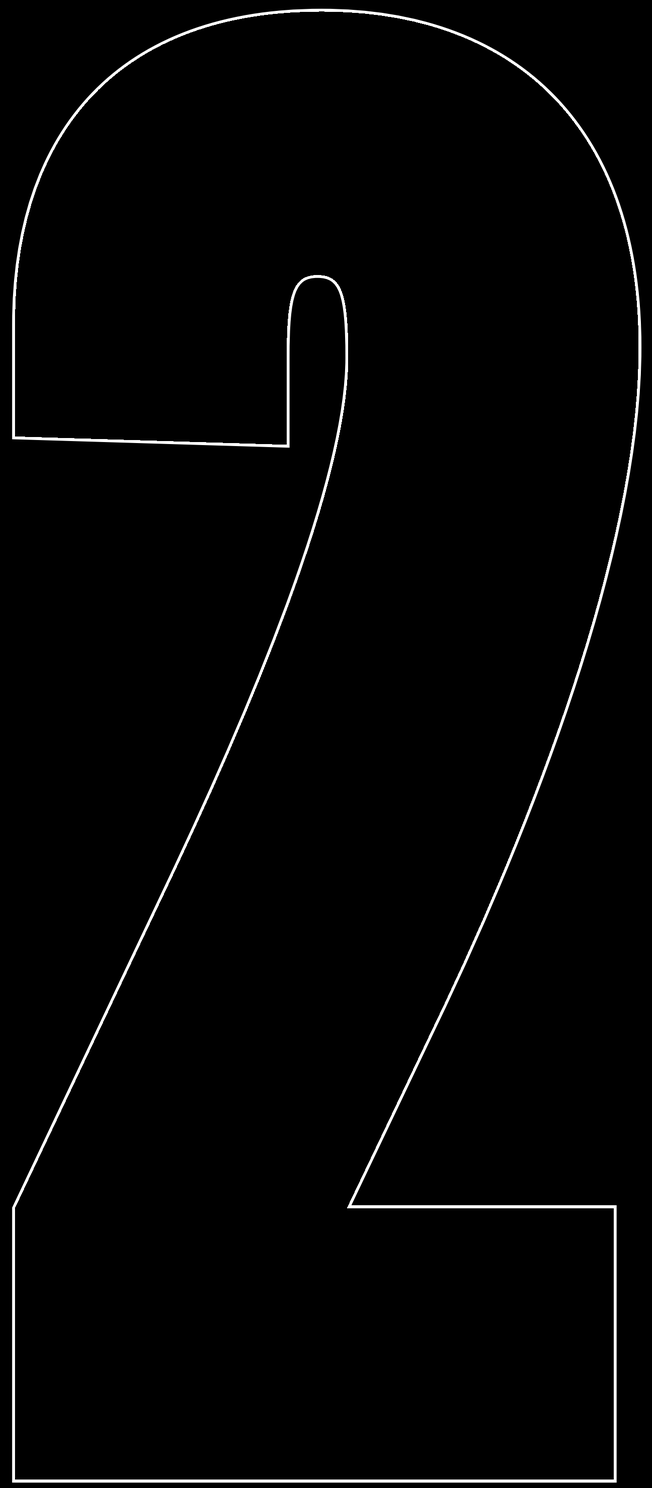
Not good



Good



“But how do I know where to put them?”



Most of your points should be on extremes*

*The top-, bottom-, left-, and right-most points on a curve.

The rest go on corners or any “meaningful” change in curve inflection, (i.e. anywhere else where necessary to help describe the curve you want).

IMPORTANT TIP FOR THE GAME!

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

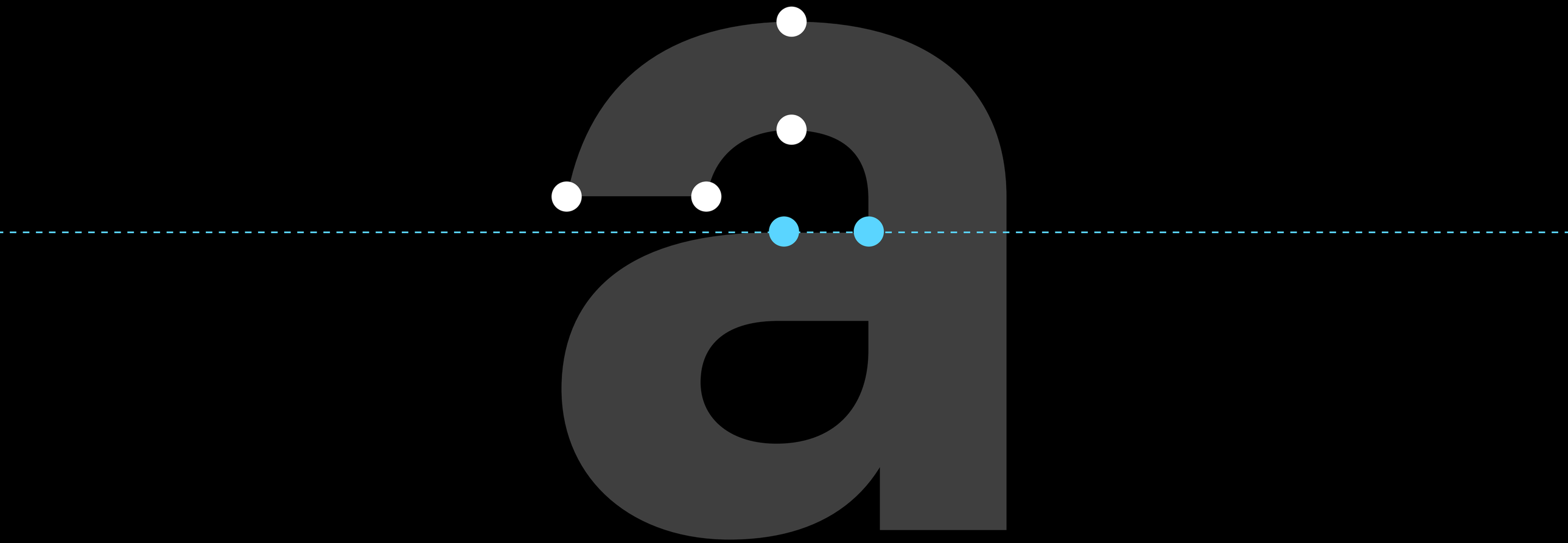
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

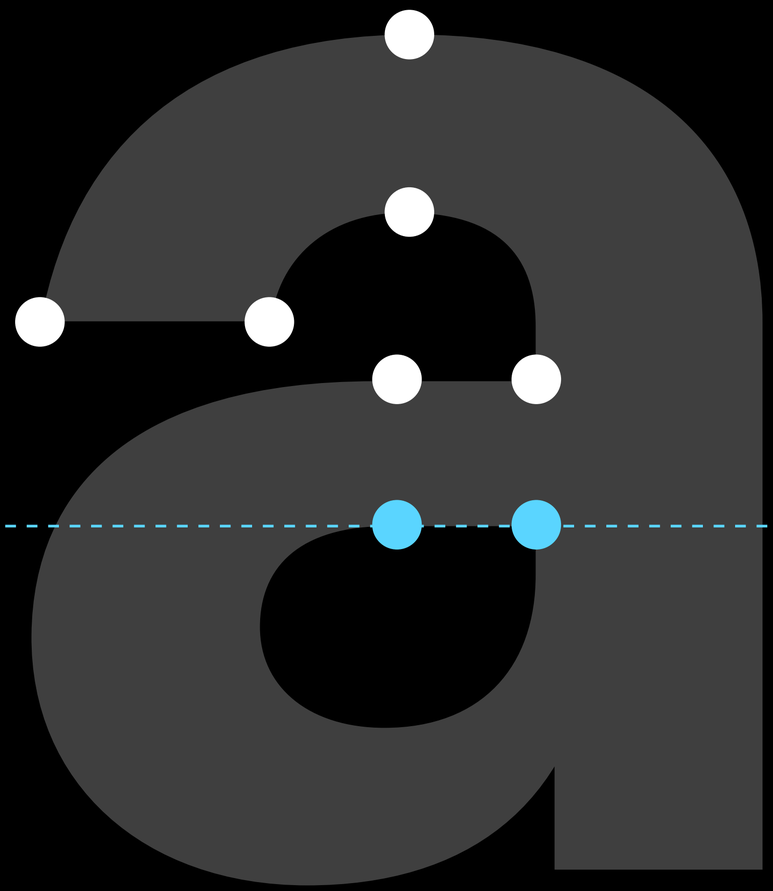
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



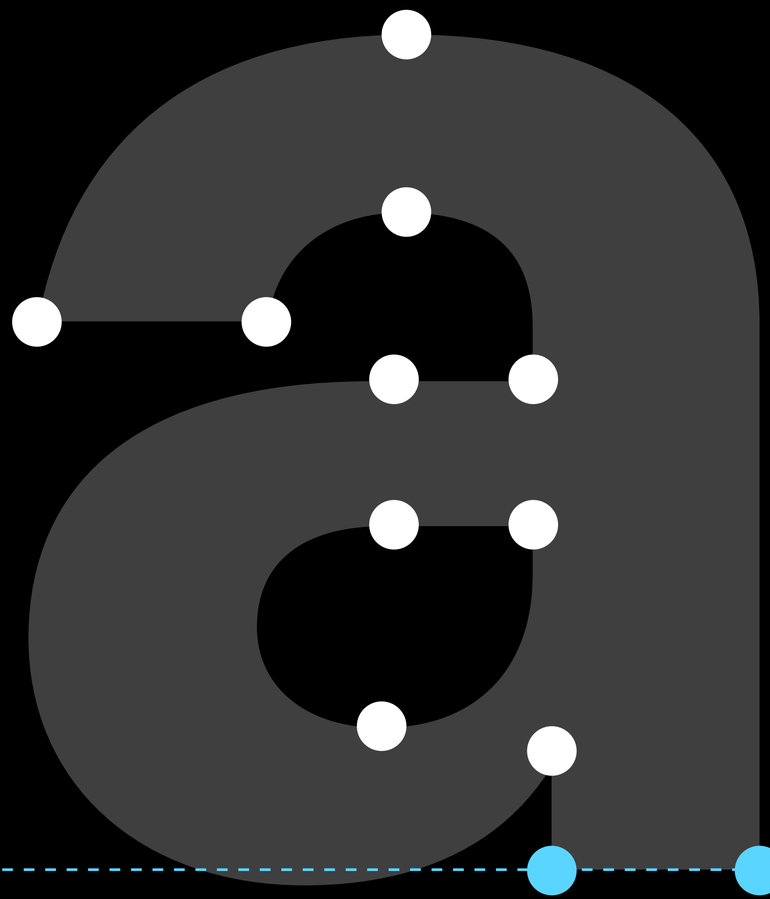
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



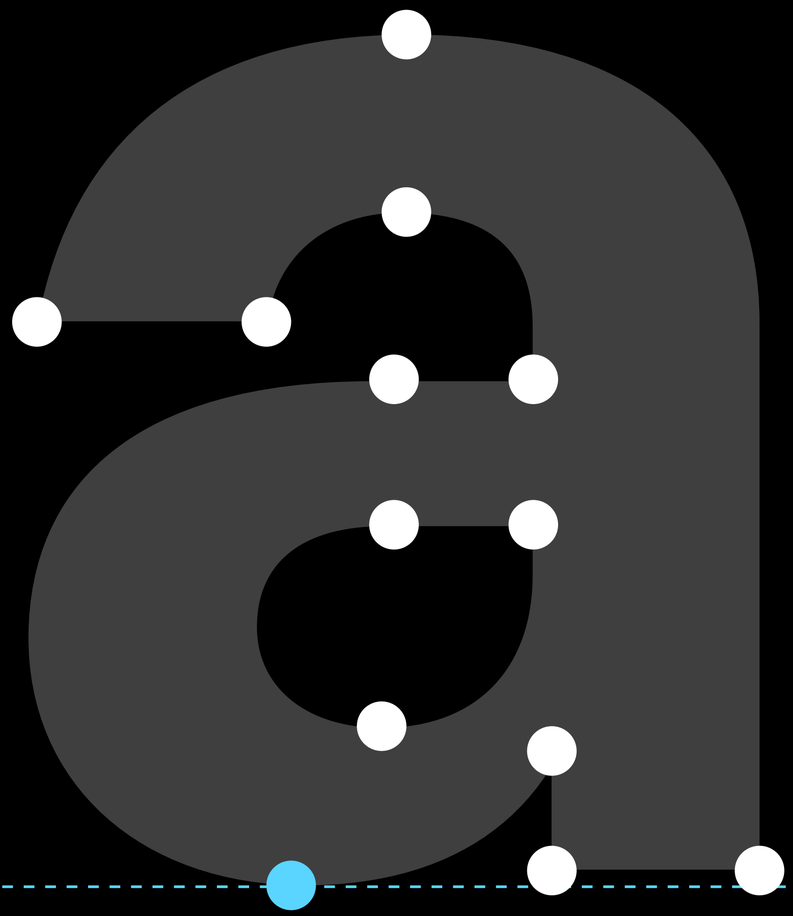
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

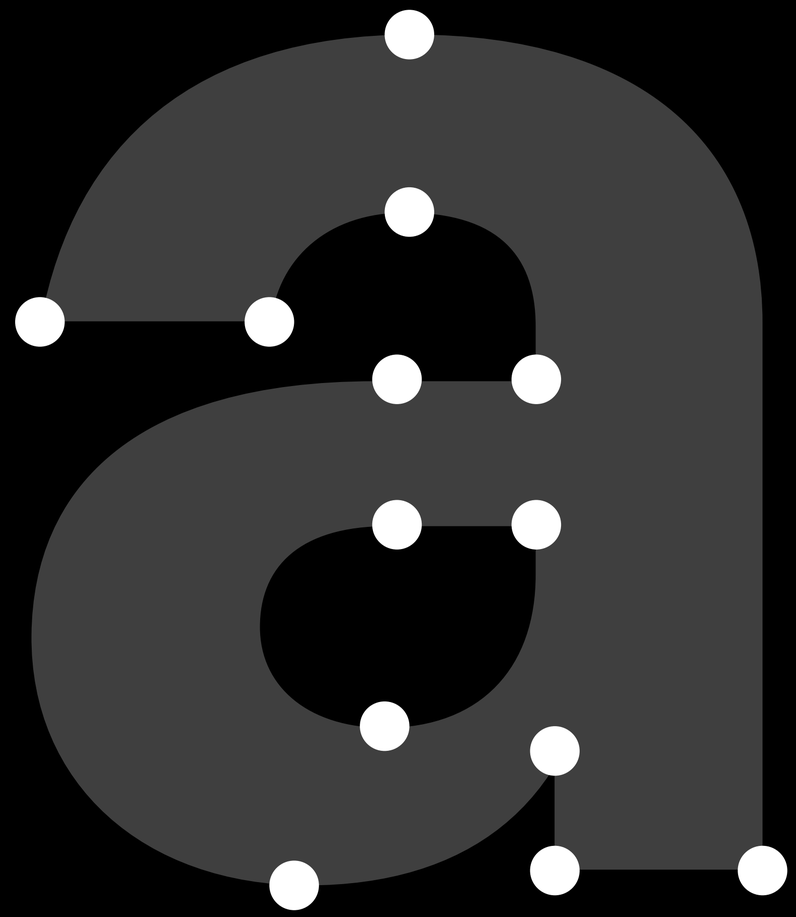
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



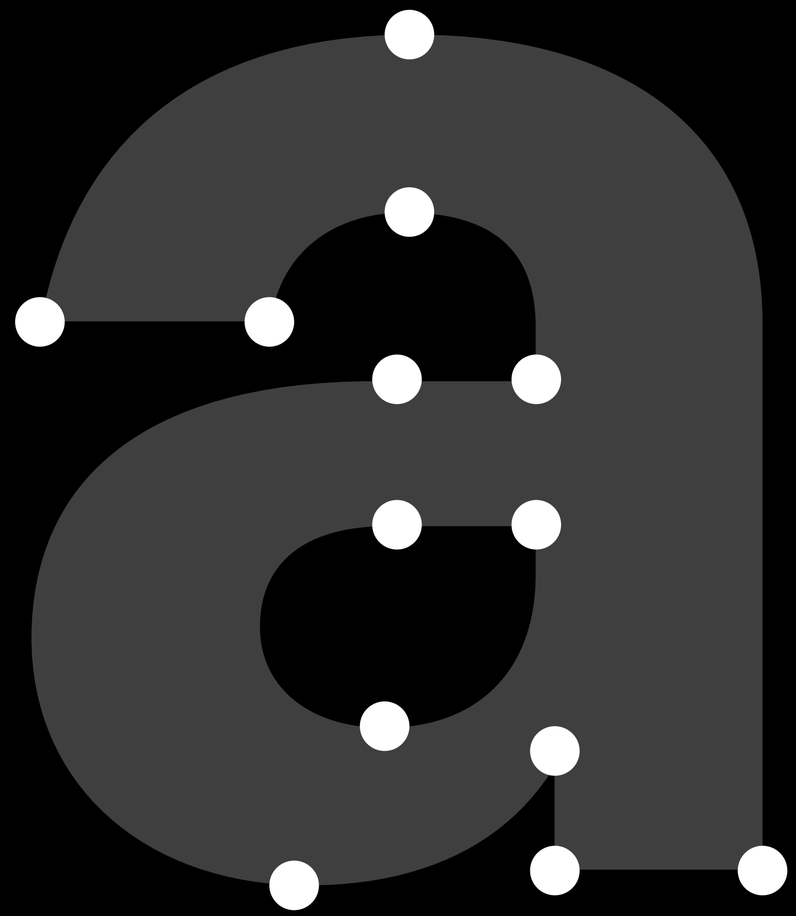
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



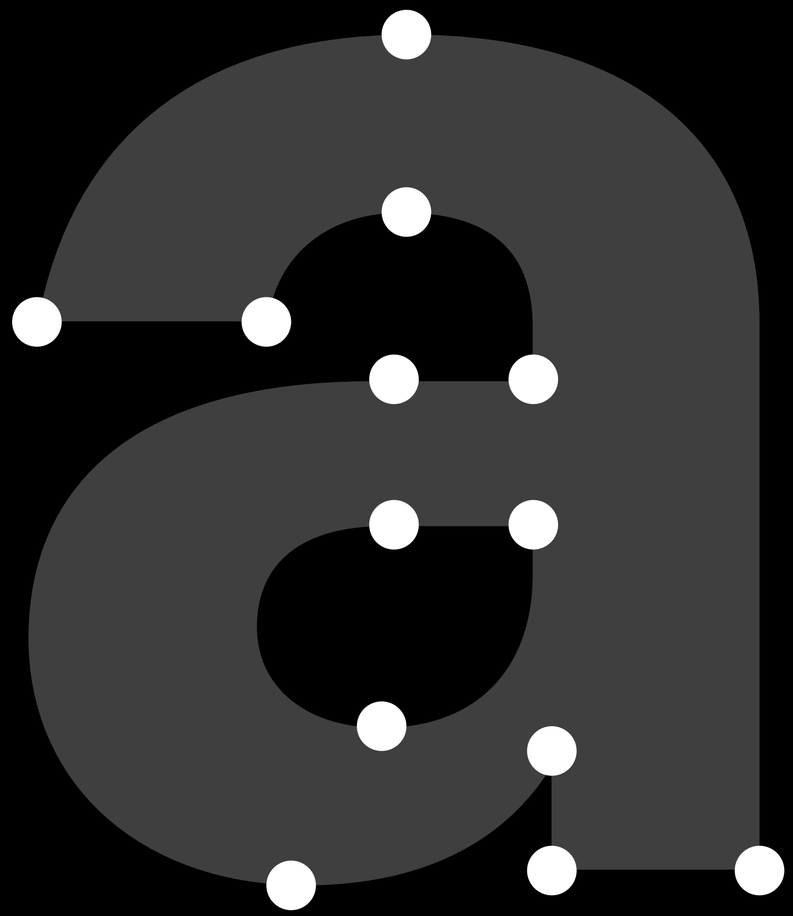
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



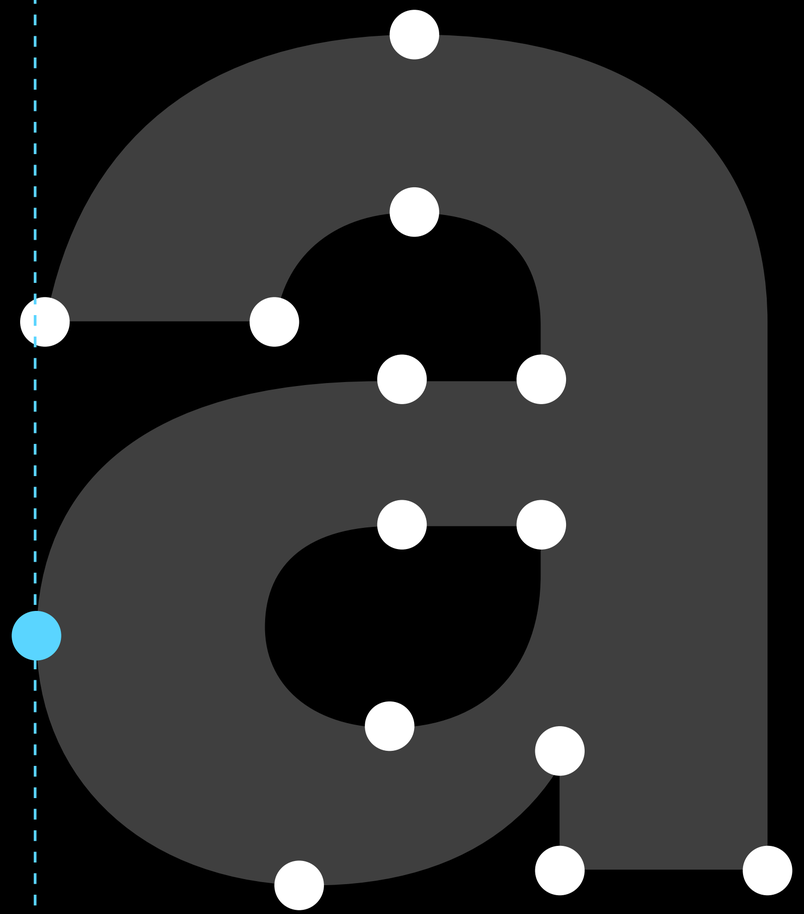
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



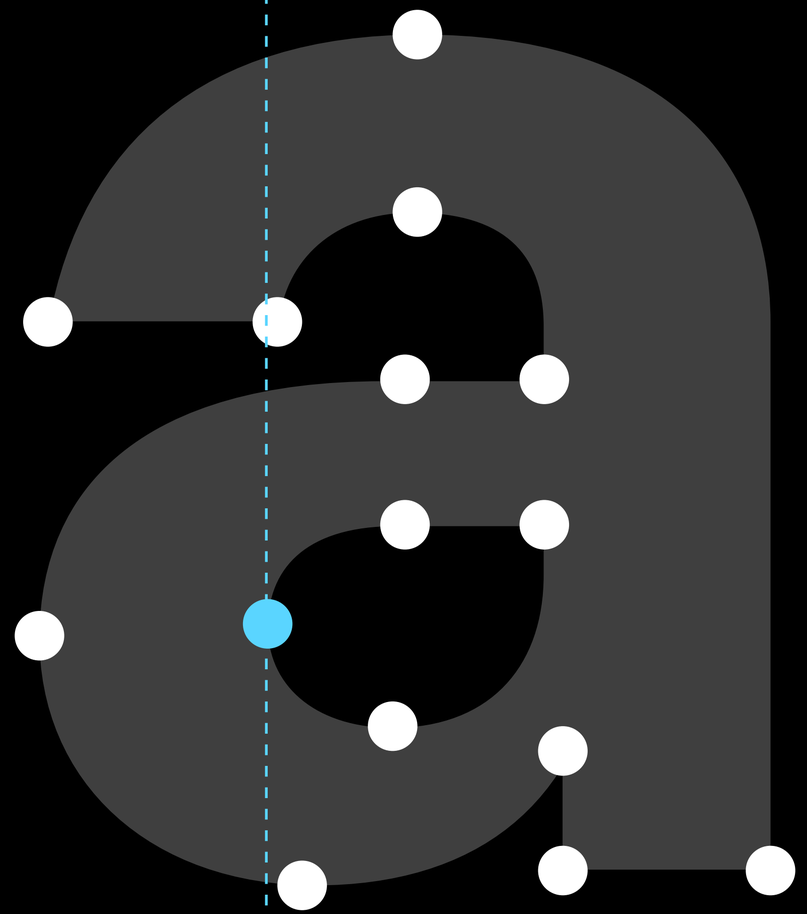
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



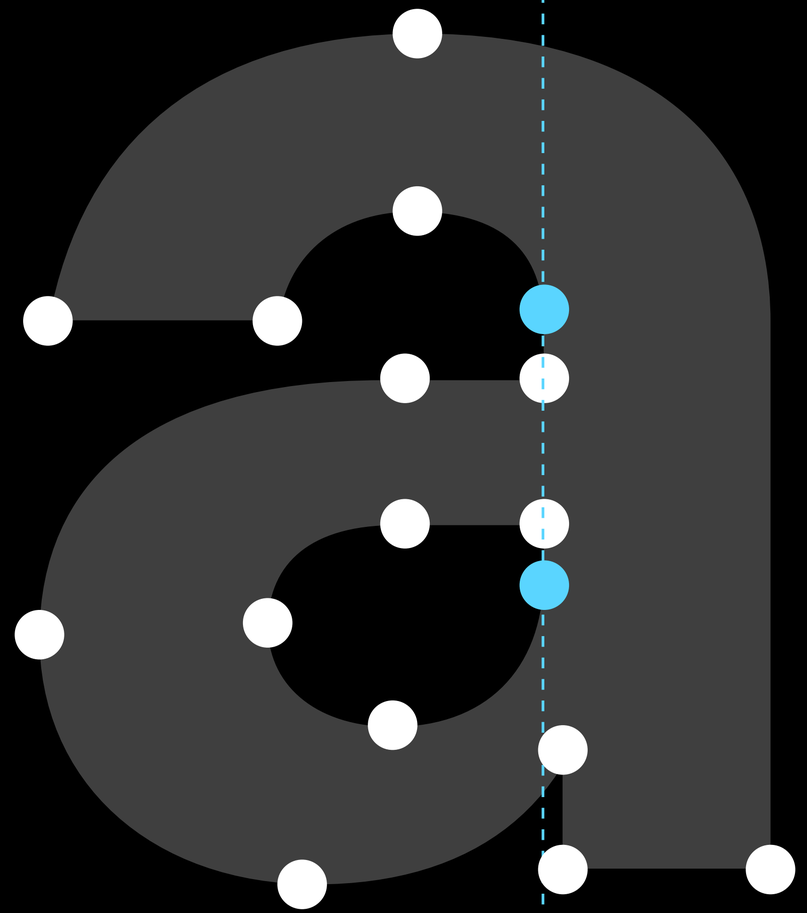
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



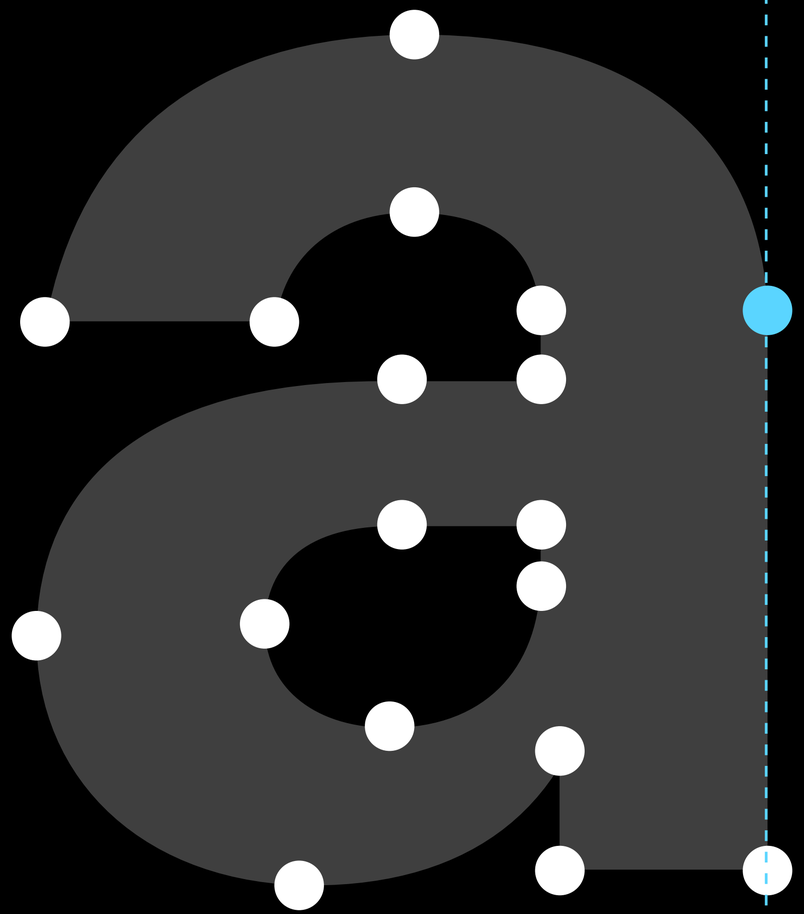
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



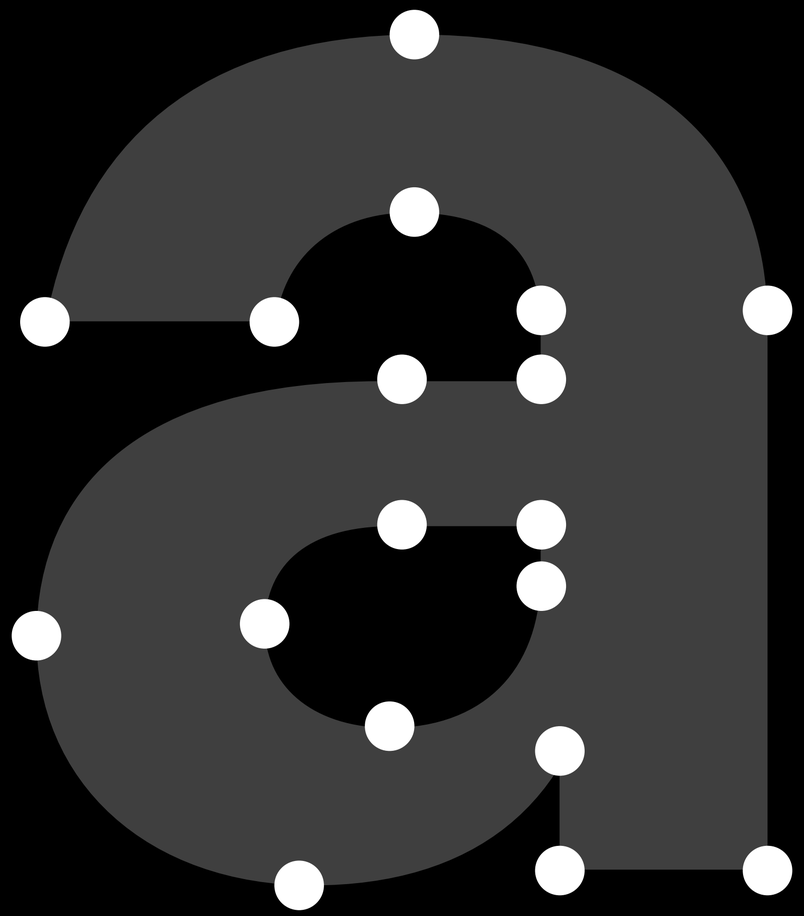
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



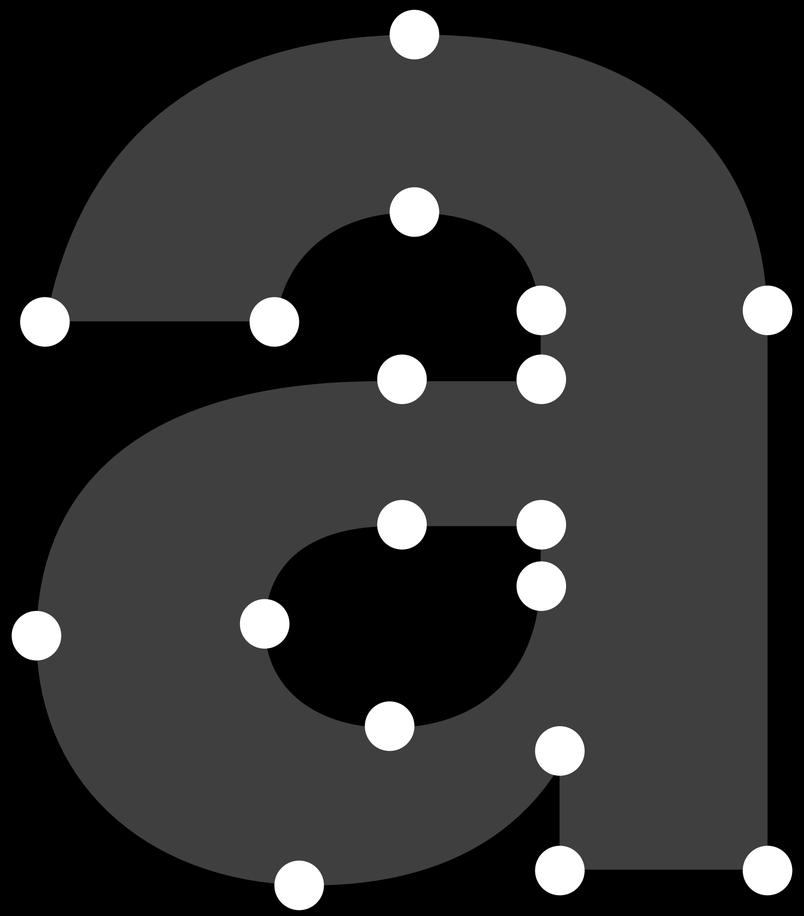
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



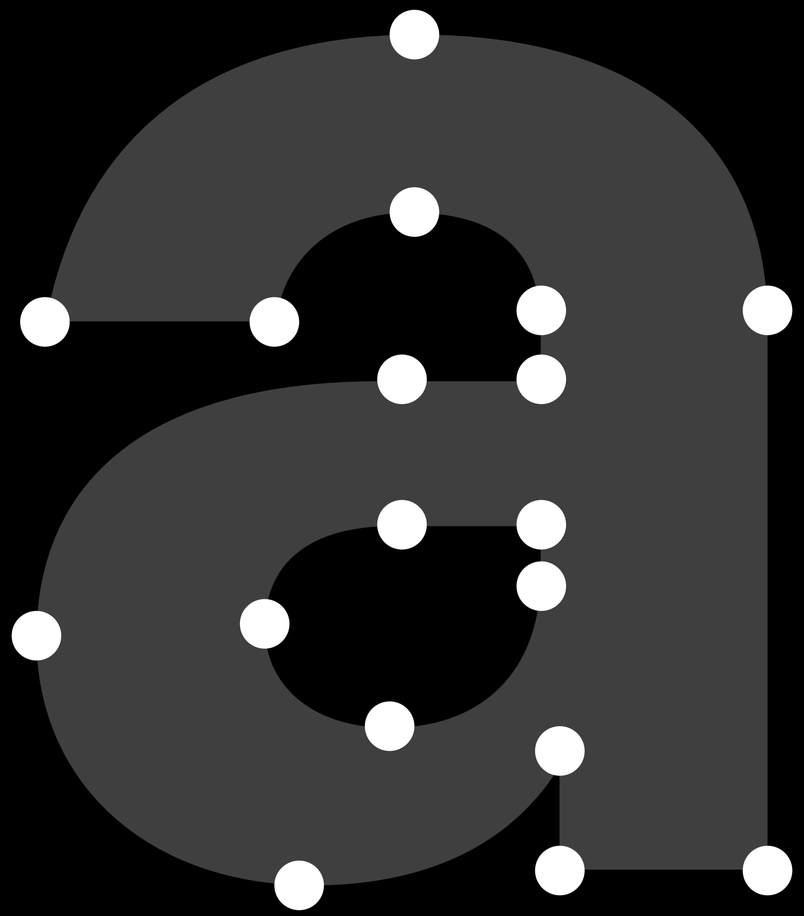
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



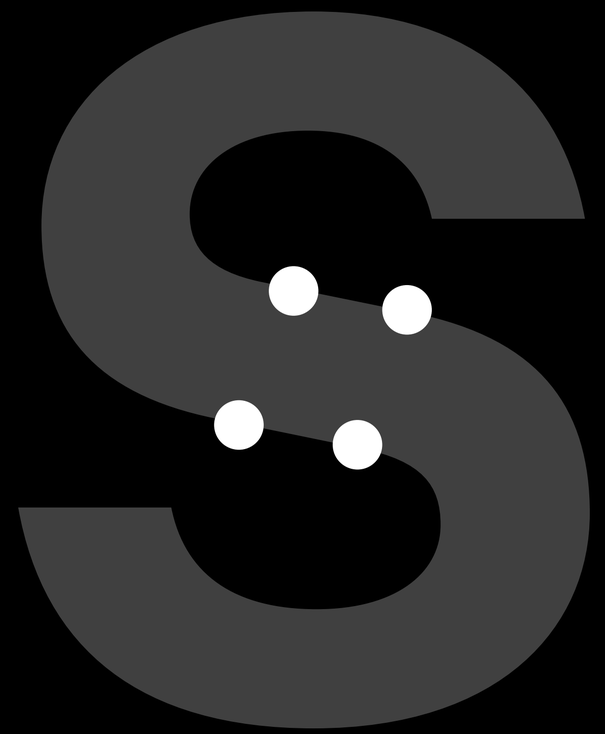
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.



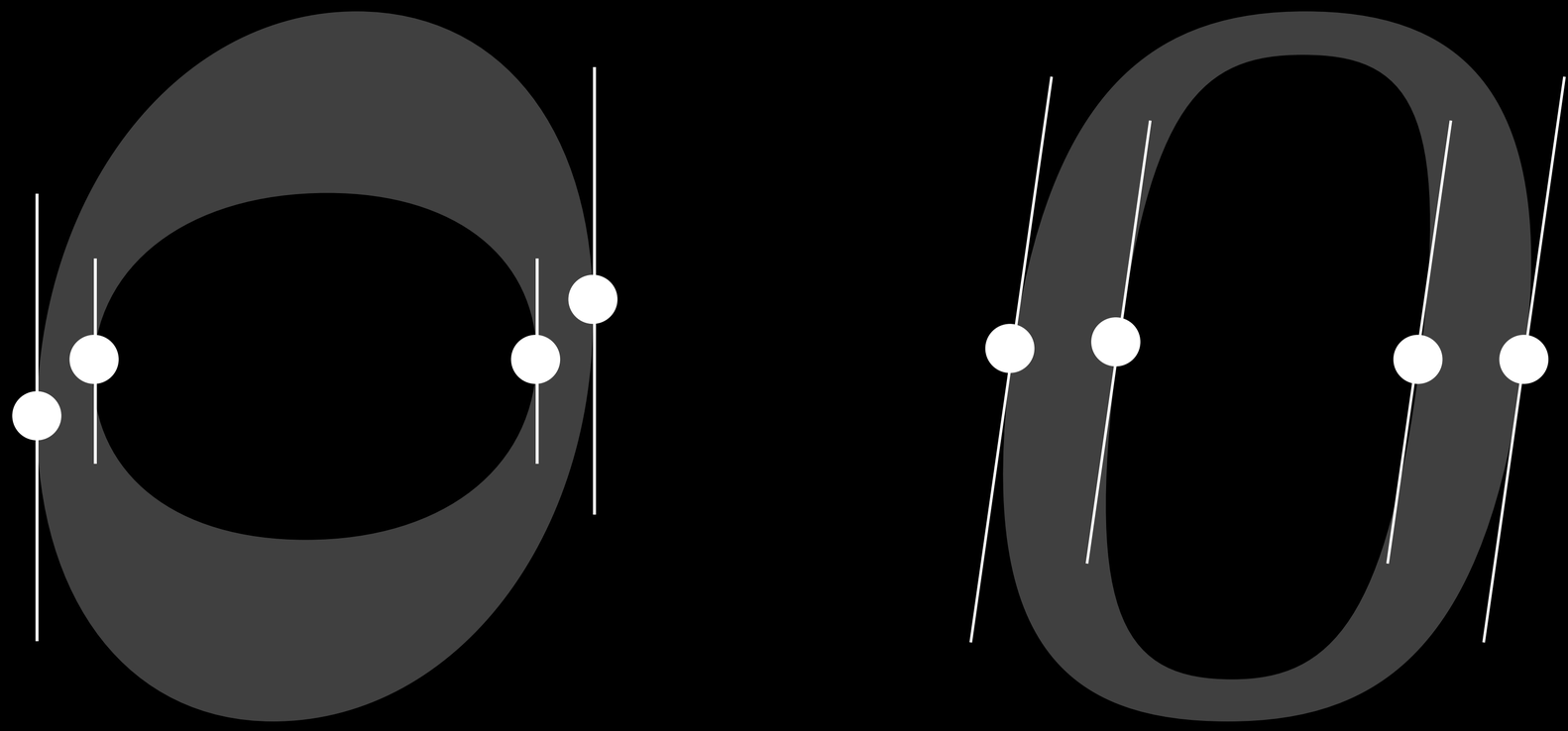
Imagine invisible rulers coming from the top or sides.
Whenever they hit something, drop a point there.

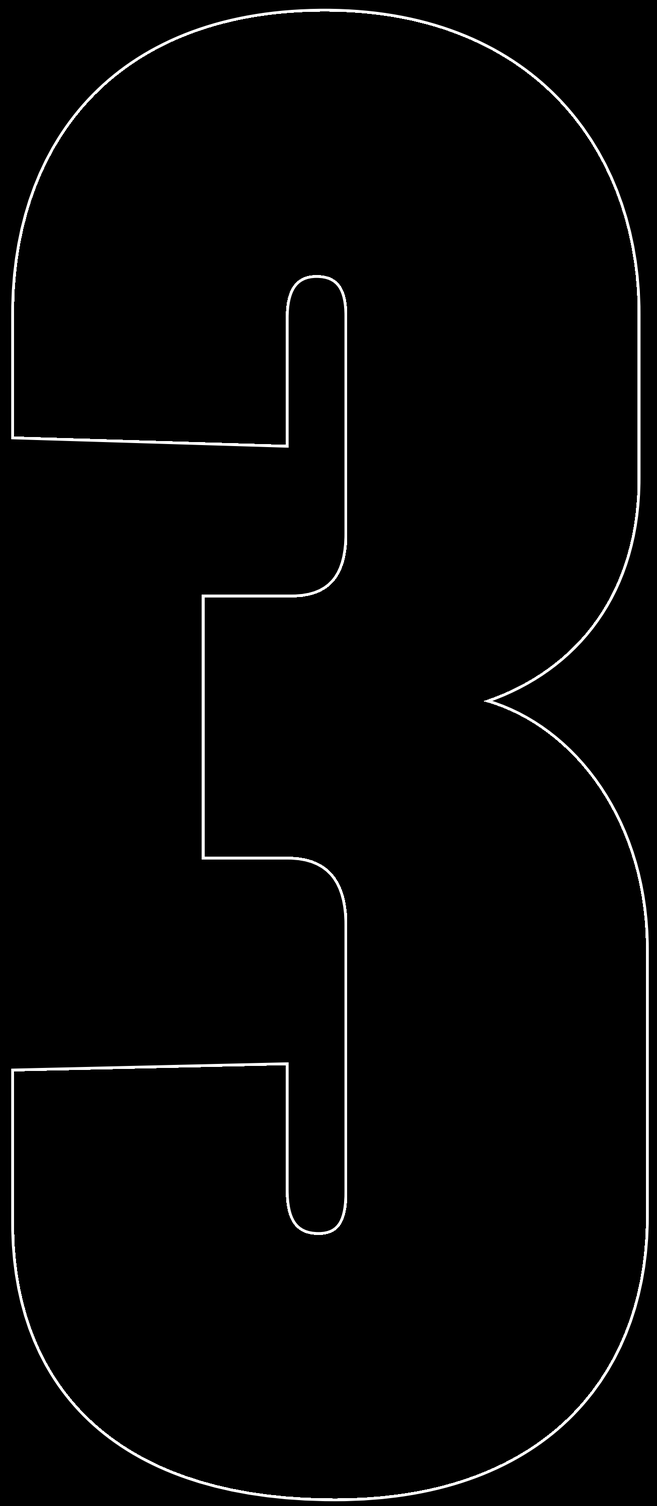


It’s never quite
that simple, is it?



It’s never quite that simple, is it?

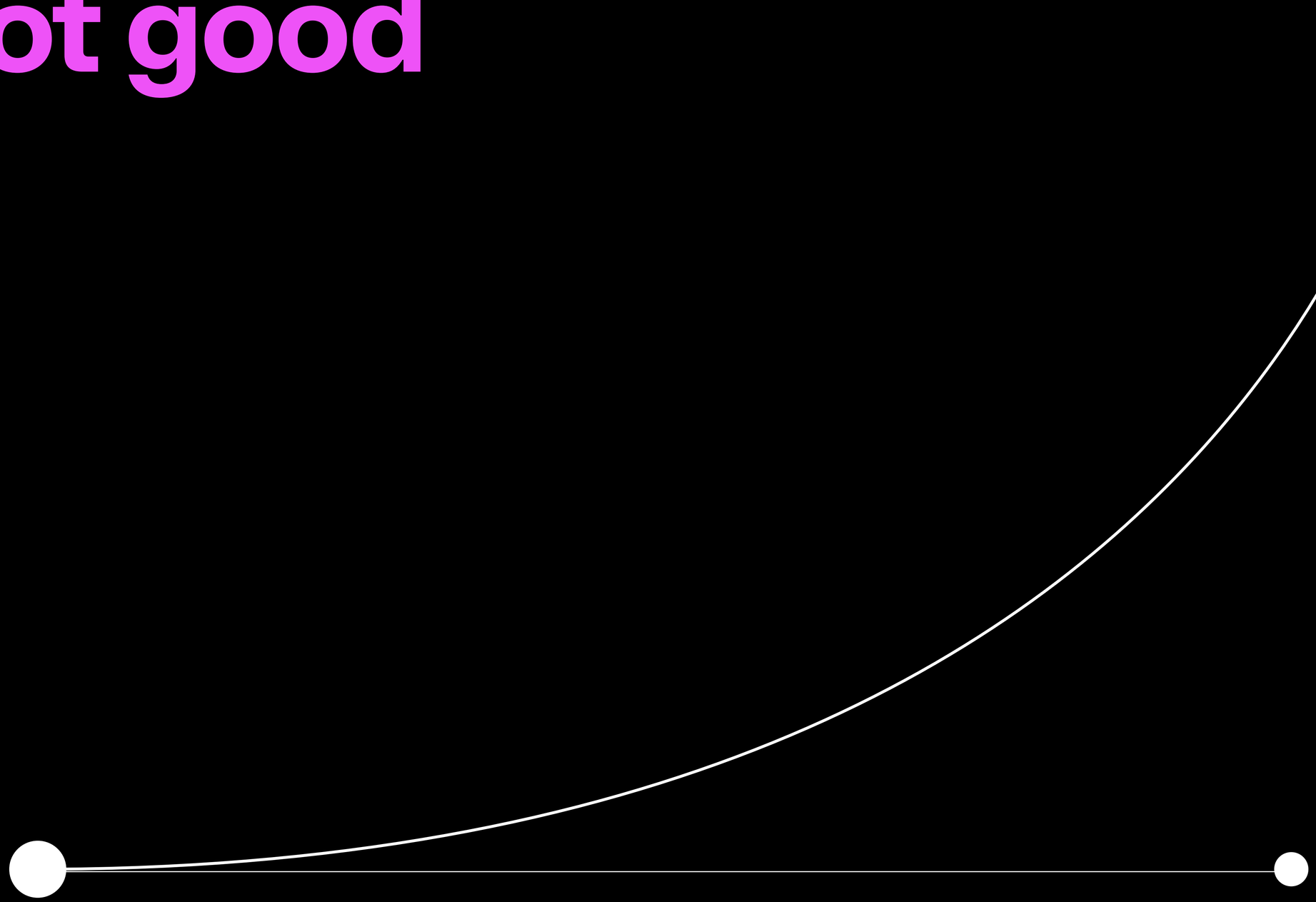




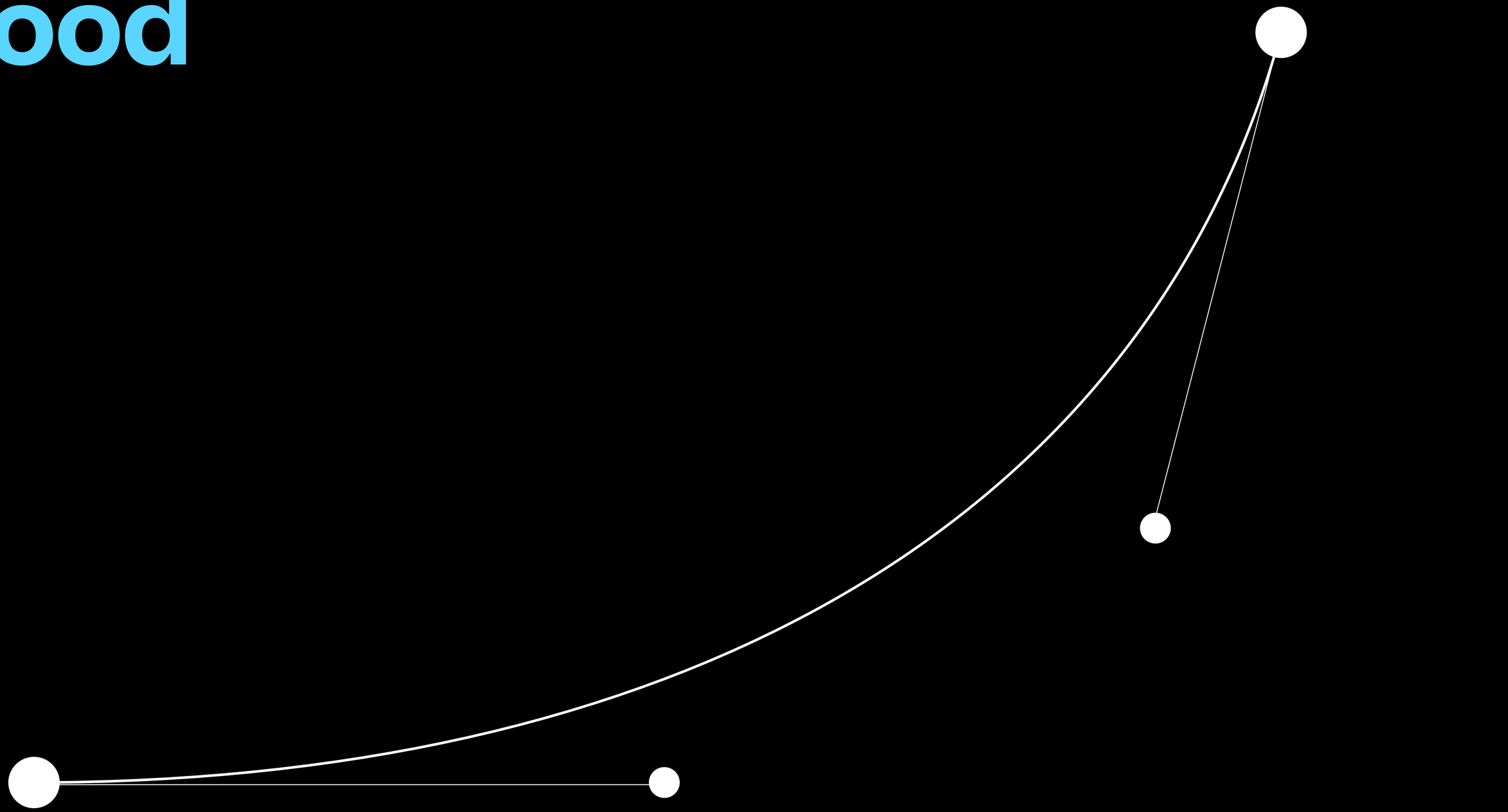
Always use two handles on each curve segment.

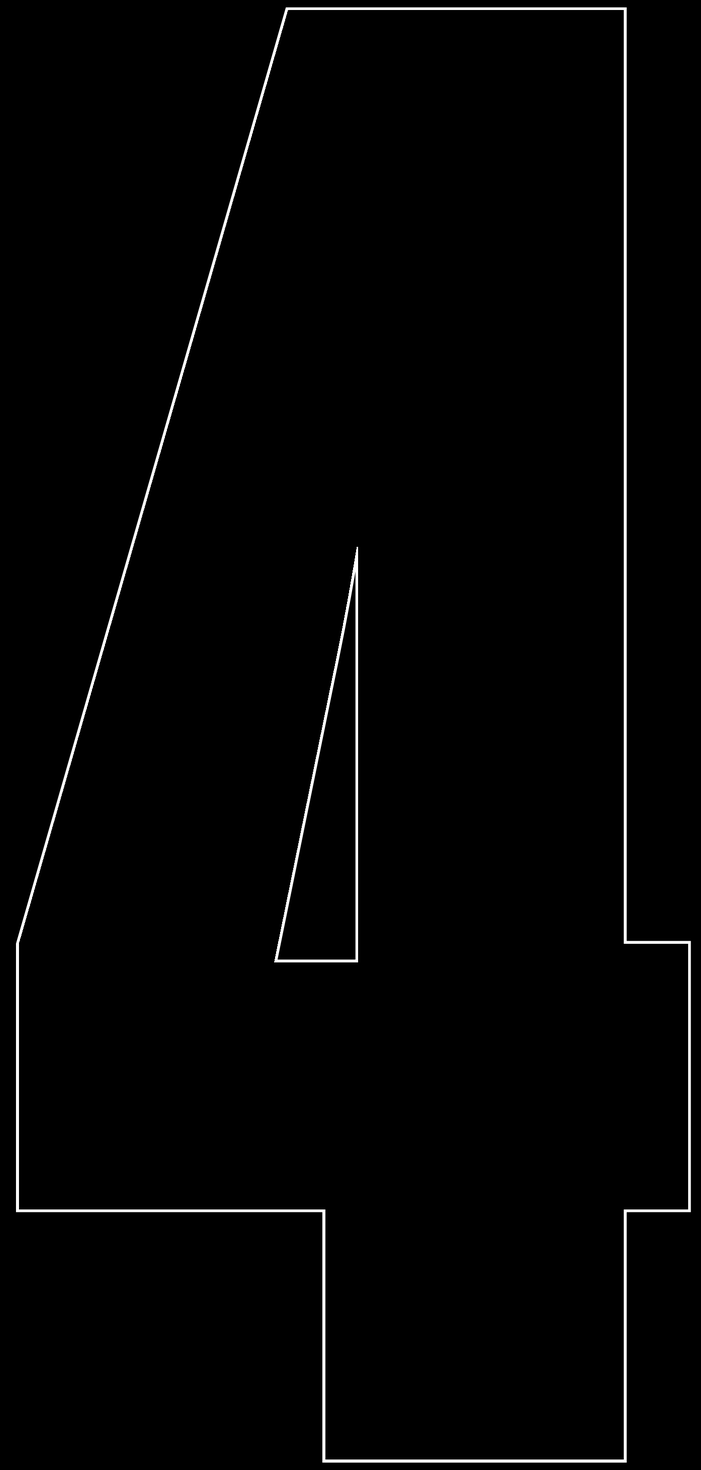
This gives you more control of your curve and helps you to avoid kinks.

Not good



Good

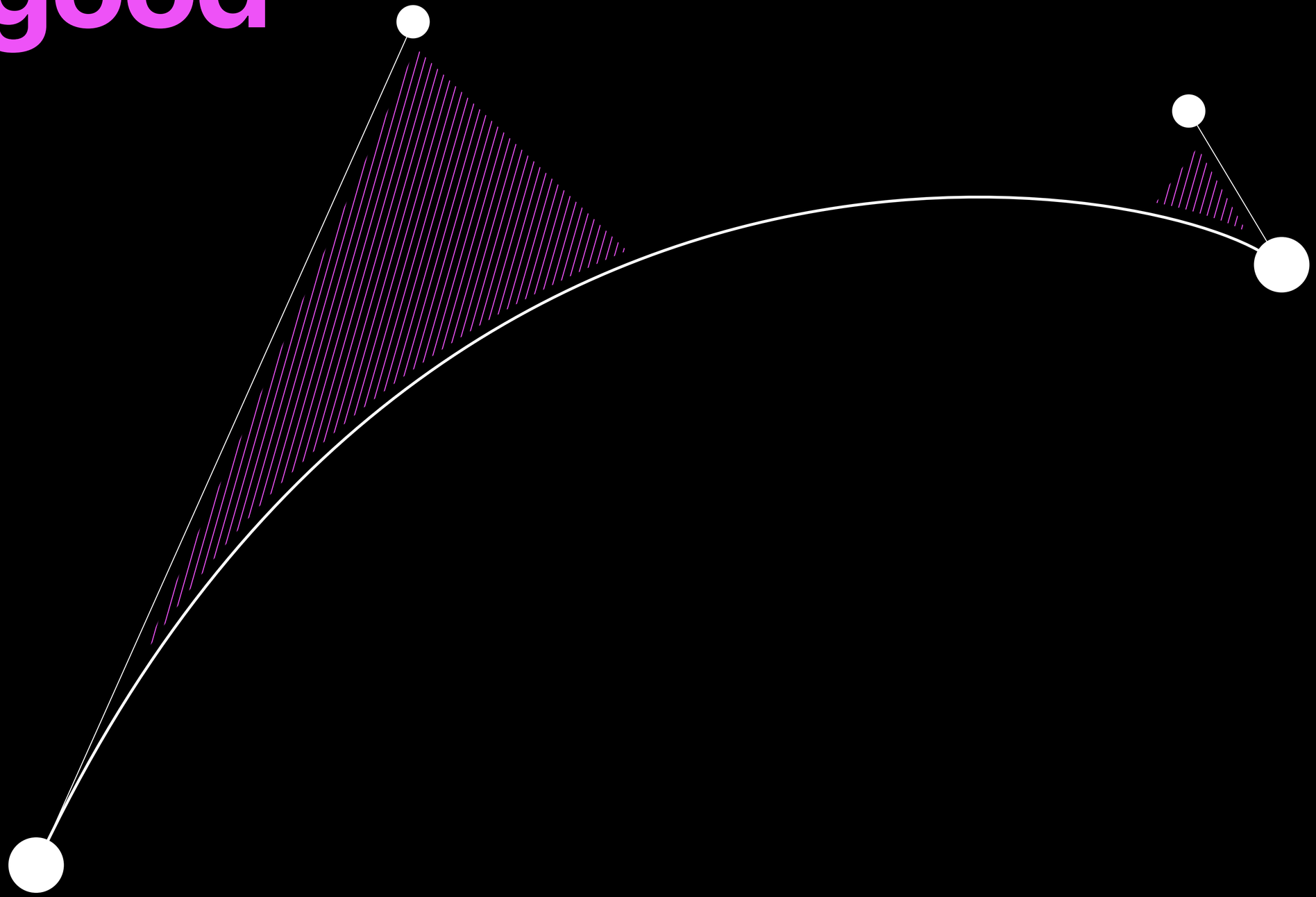




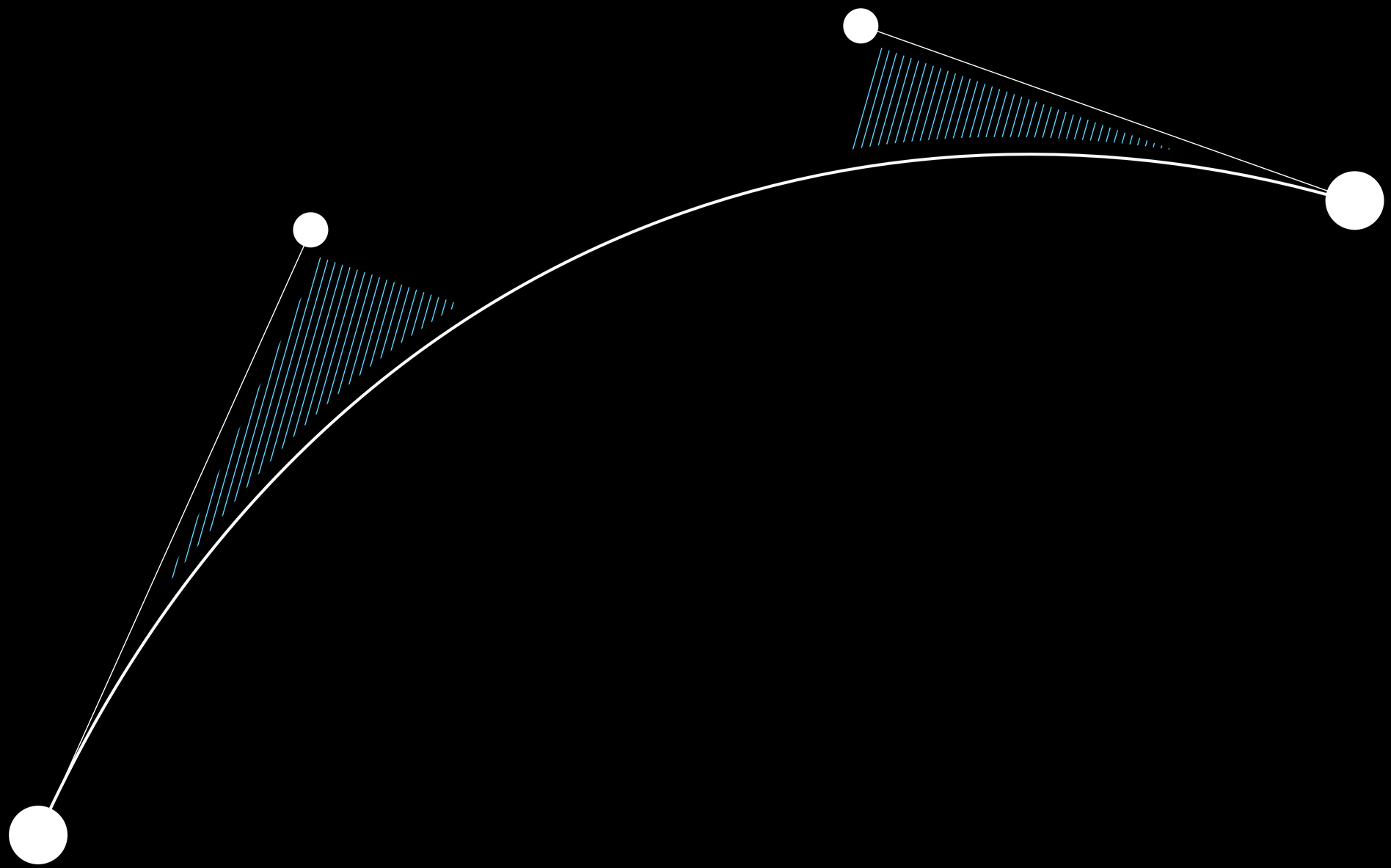
The handles should help each other out.

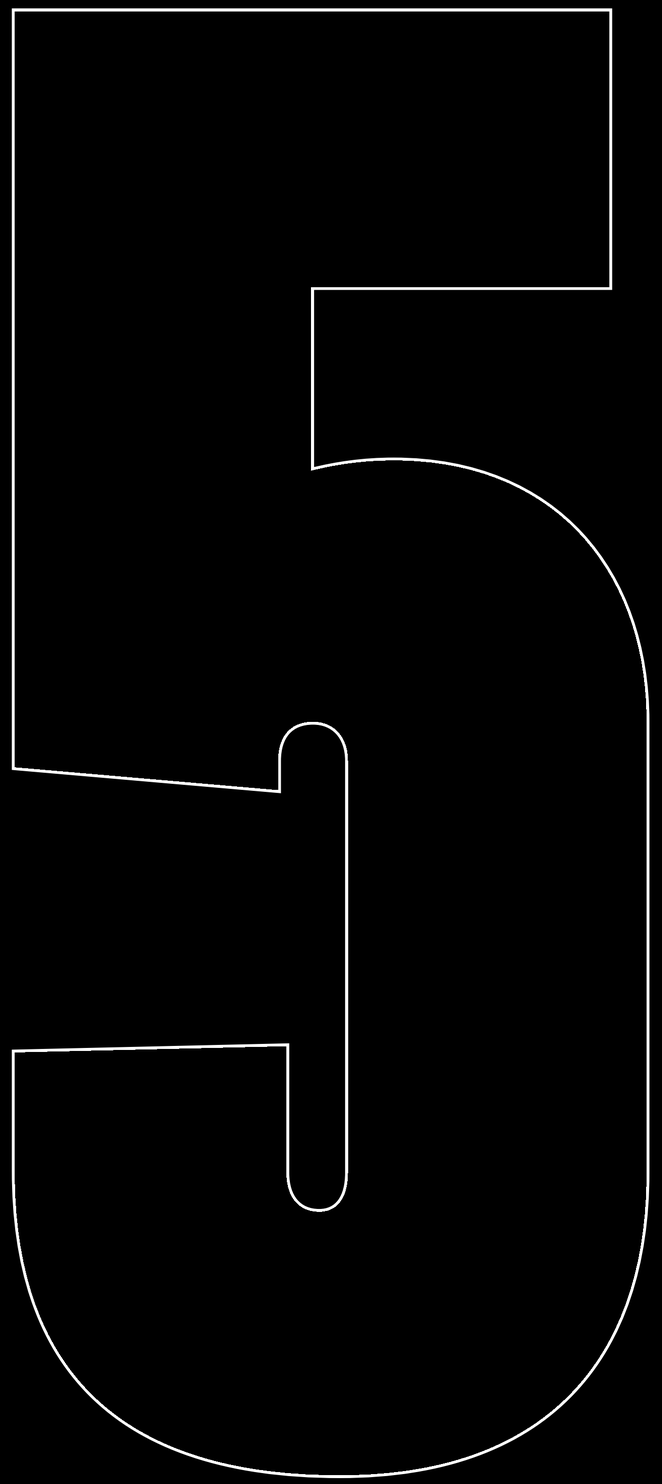
Don’t make one of the handles do all the work. Each handle should be collaborating with the other one to make one simple gesture.

Not good



Good



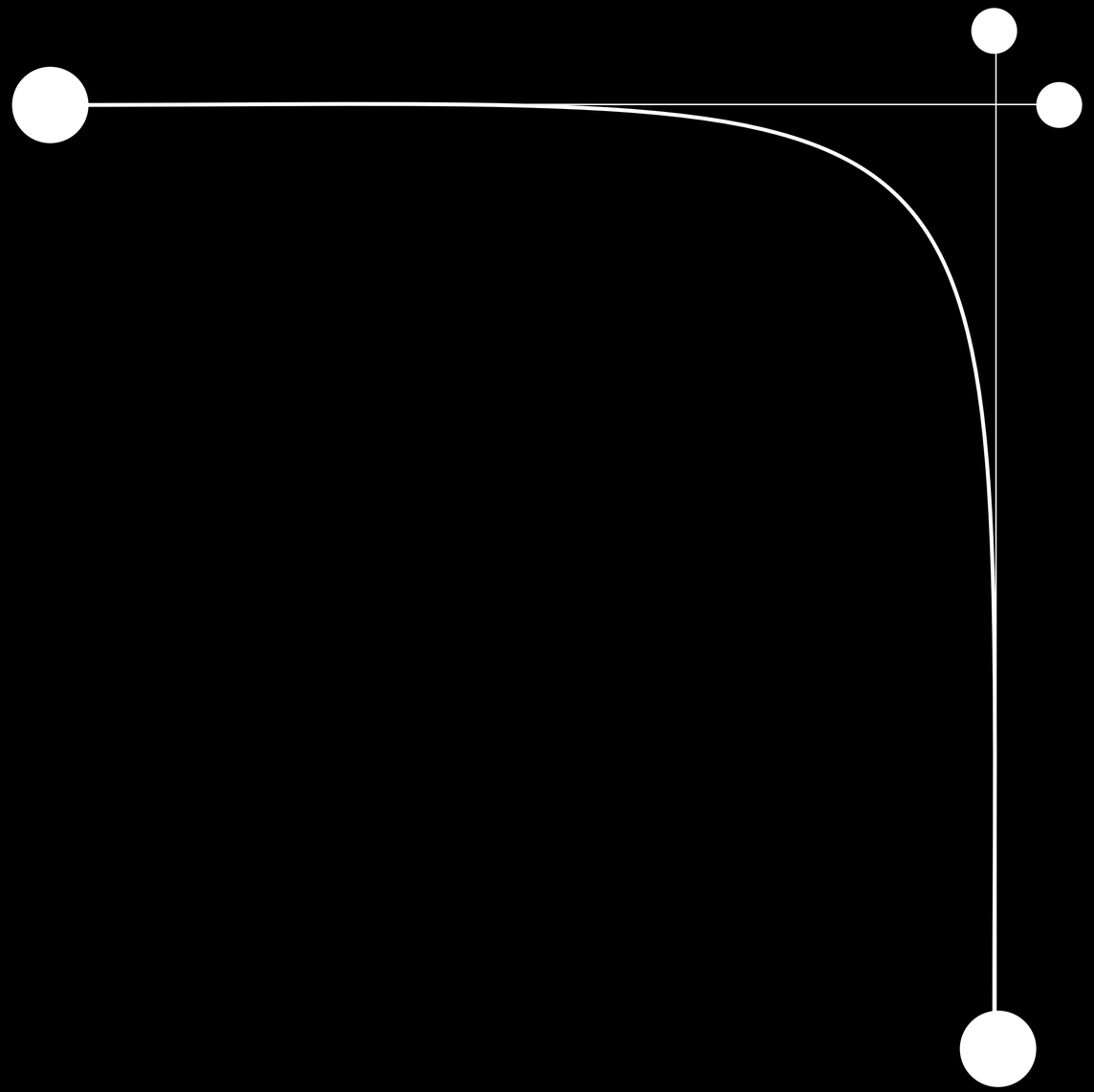


A handle should not cross into the other's axis.

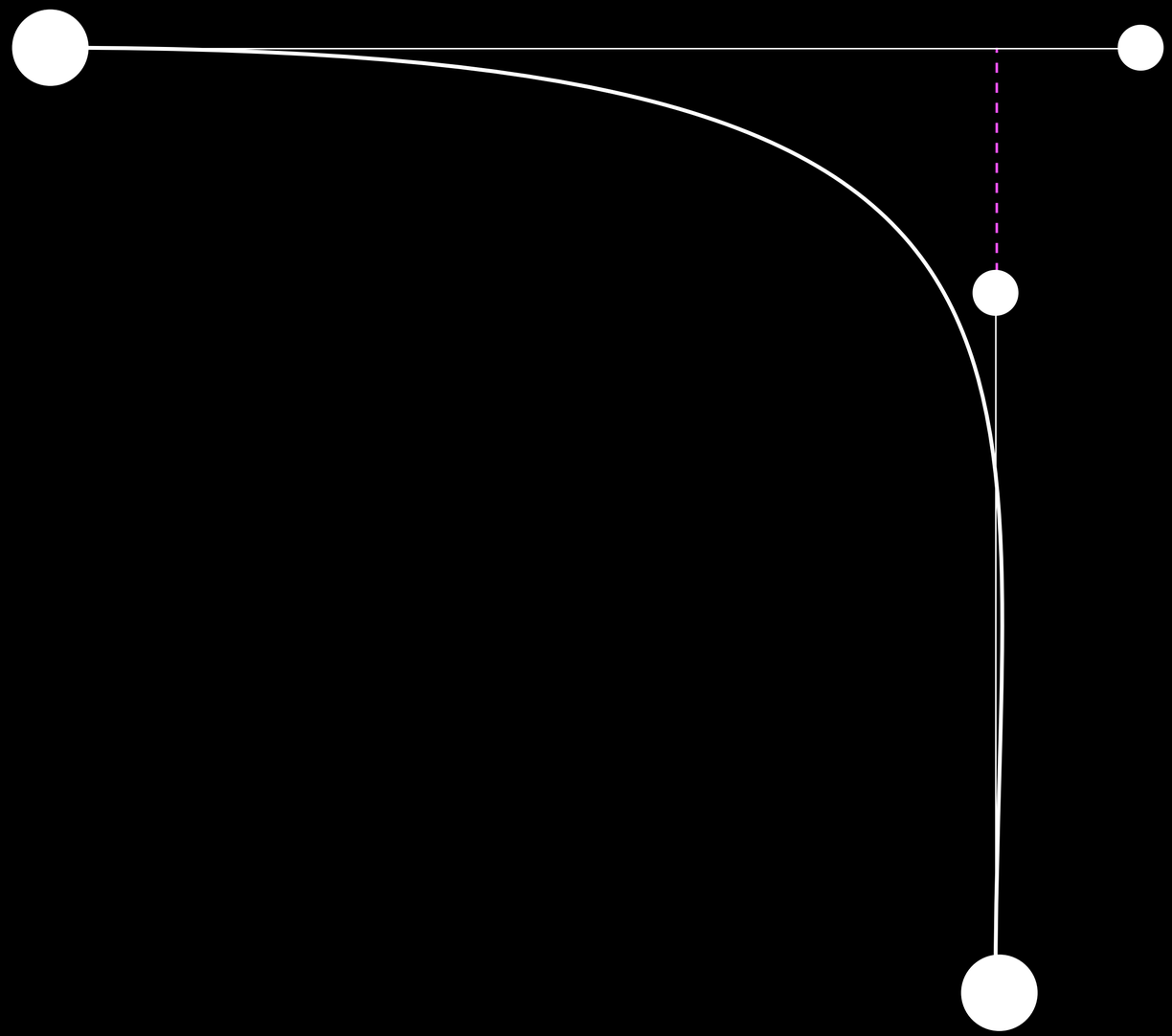
Maybe do this instead:

- move the on-curve point
- add points
- approach your drawing a different way altogether

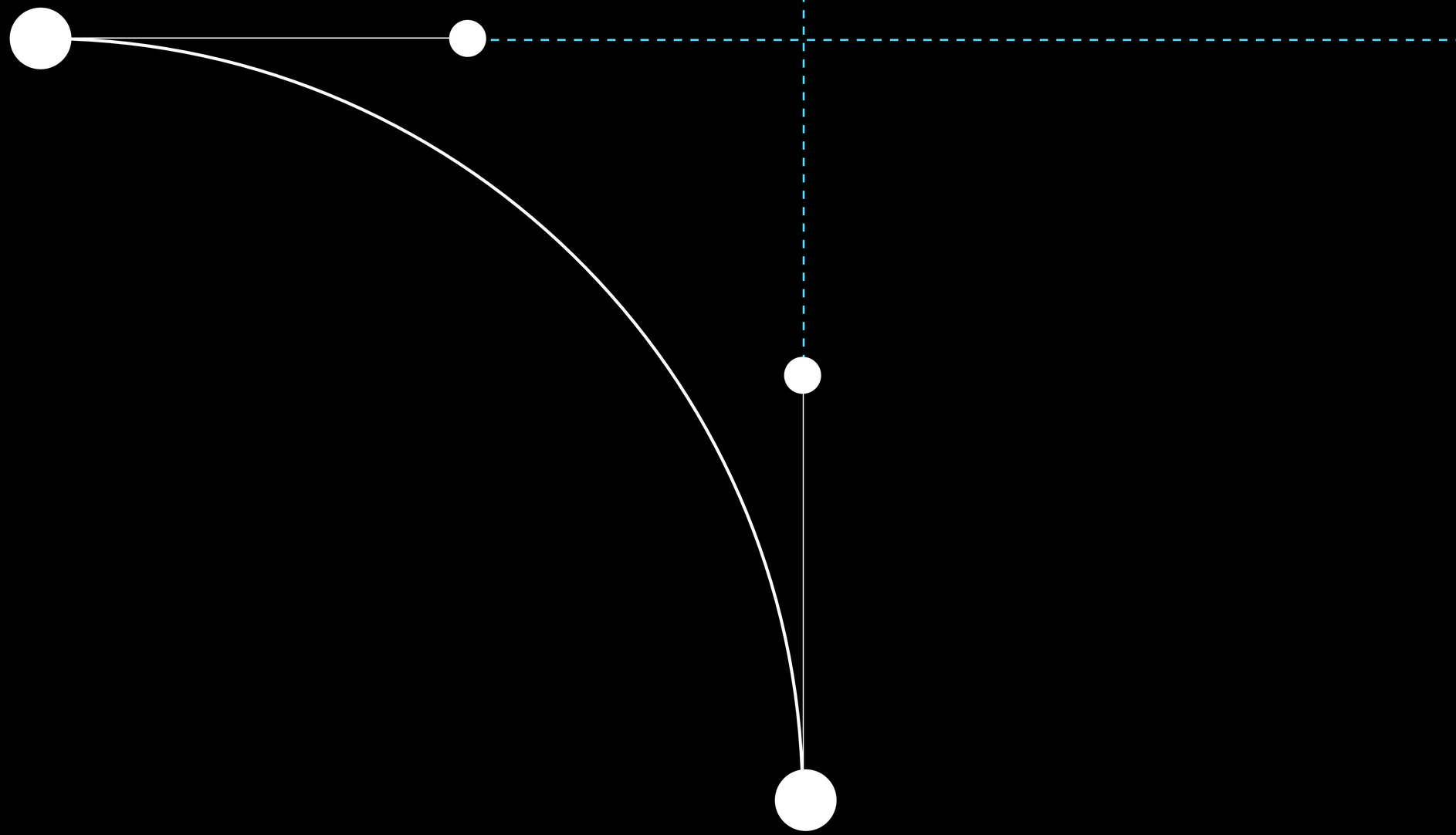
Not good



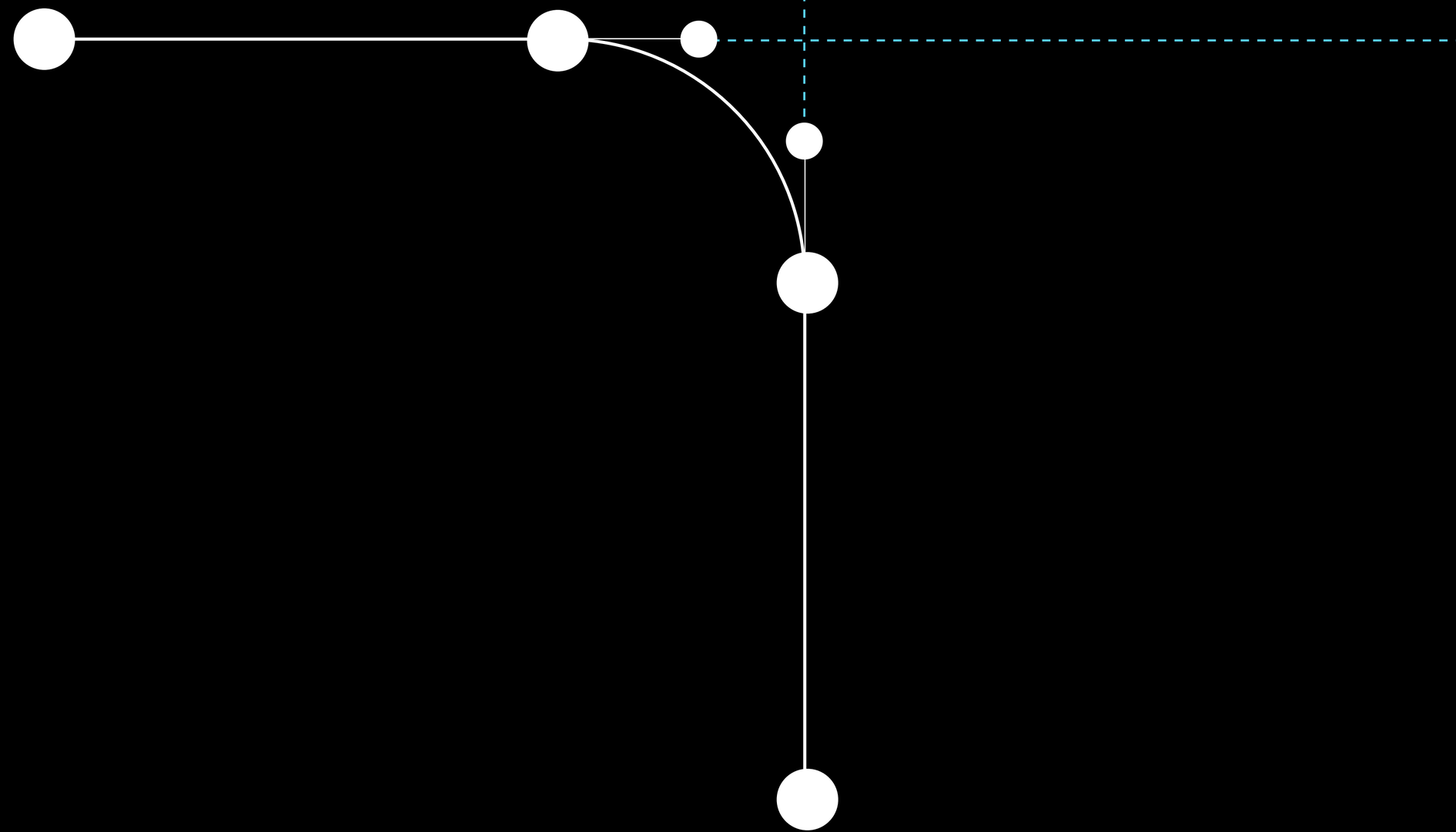
Not good

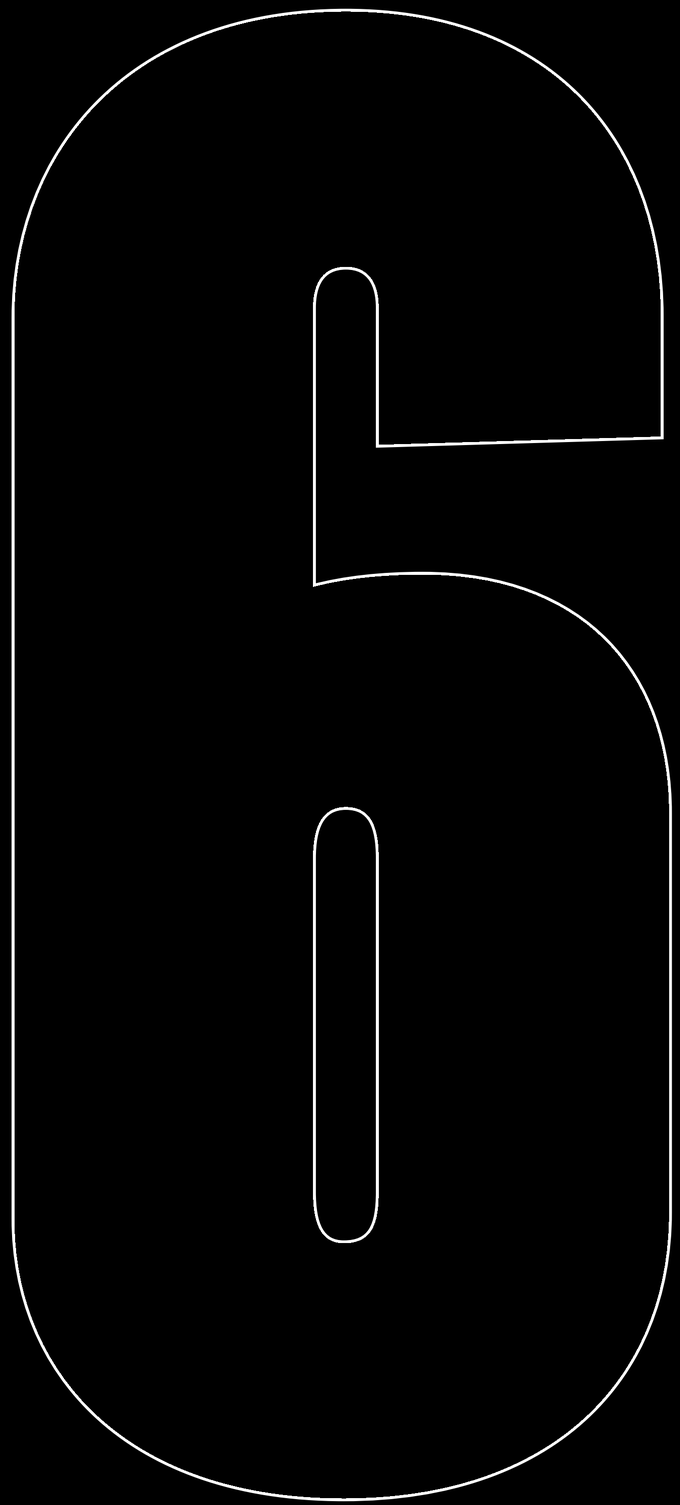


Good



Good





Overlaps are your friends.

Overlap paths or even whole shapes!

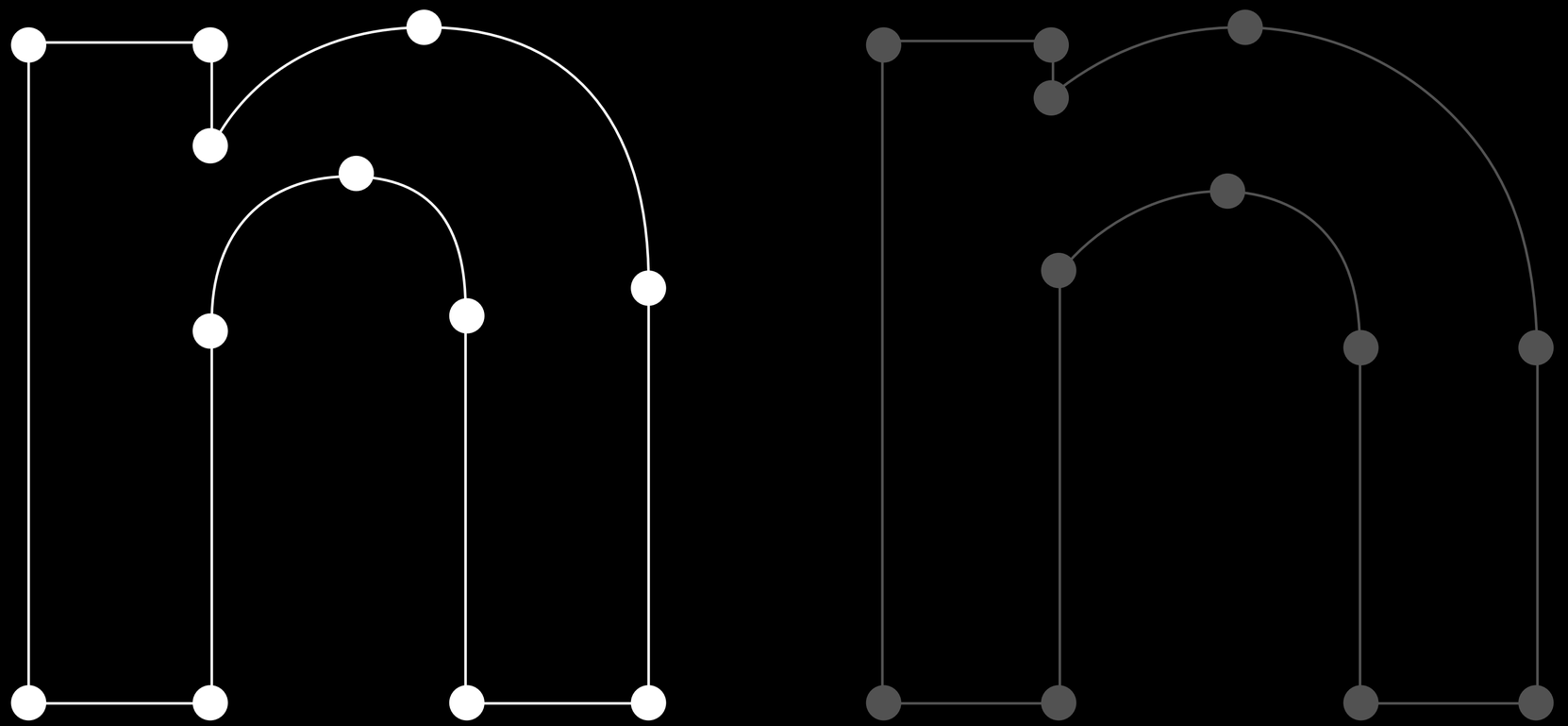
They’re:

- Easier to edit
- Better for interpolation (in most cases)

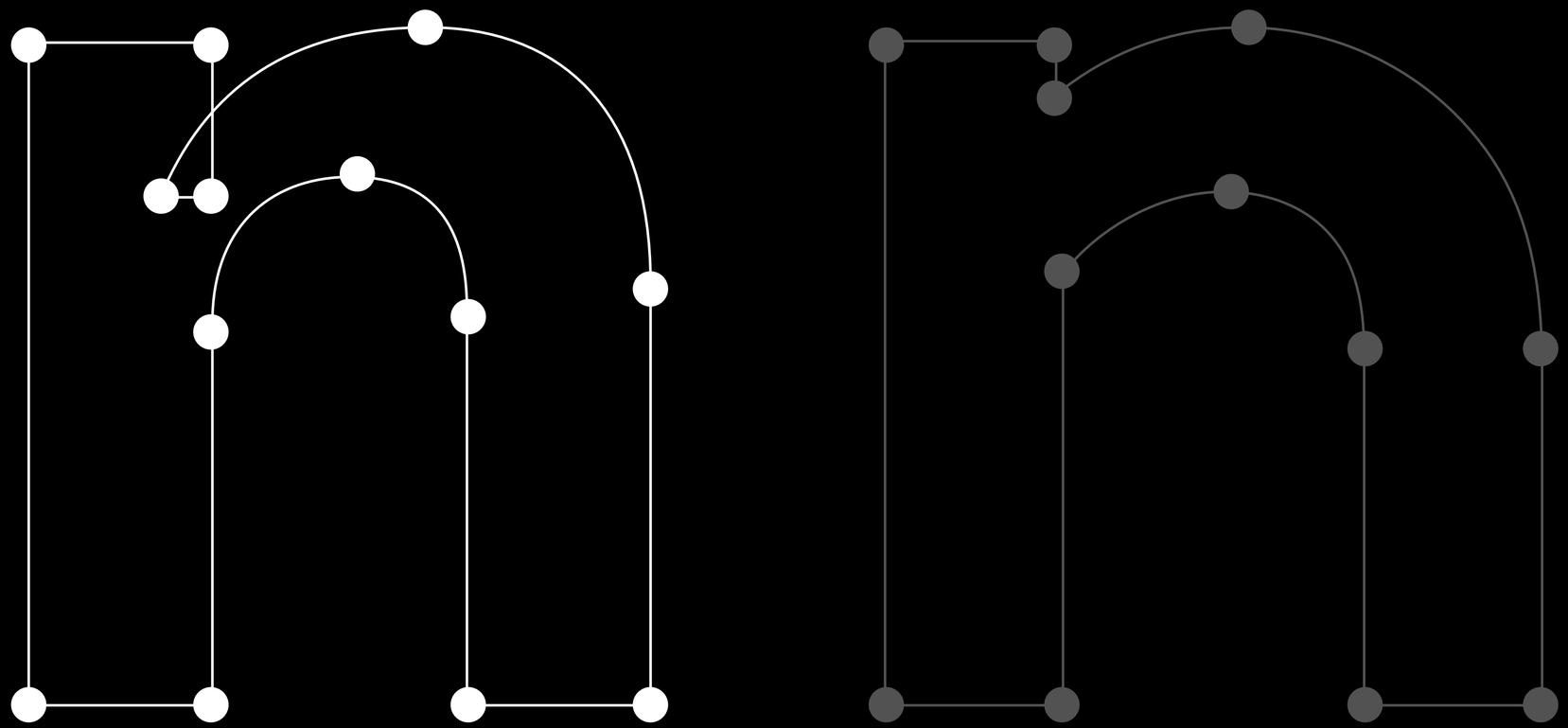
Do what makes sense for your shape.

nn

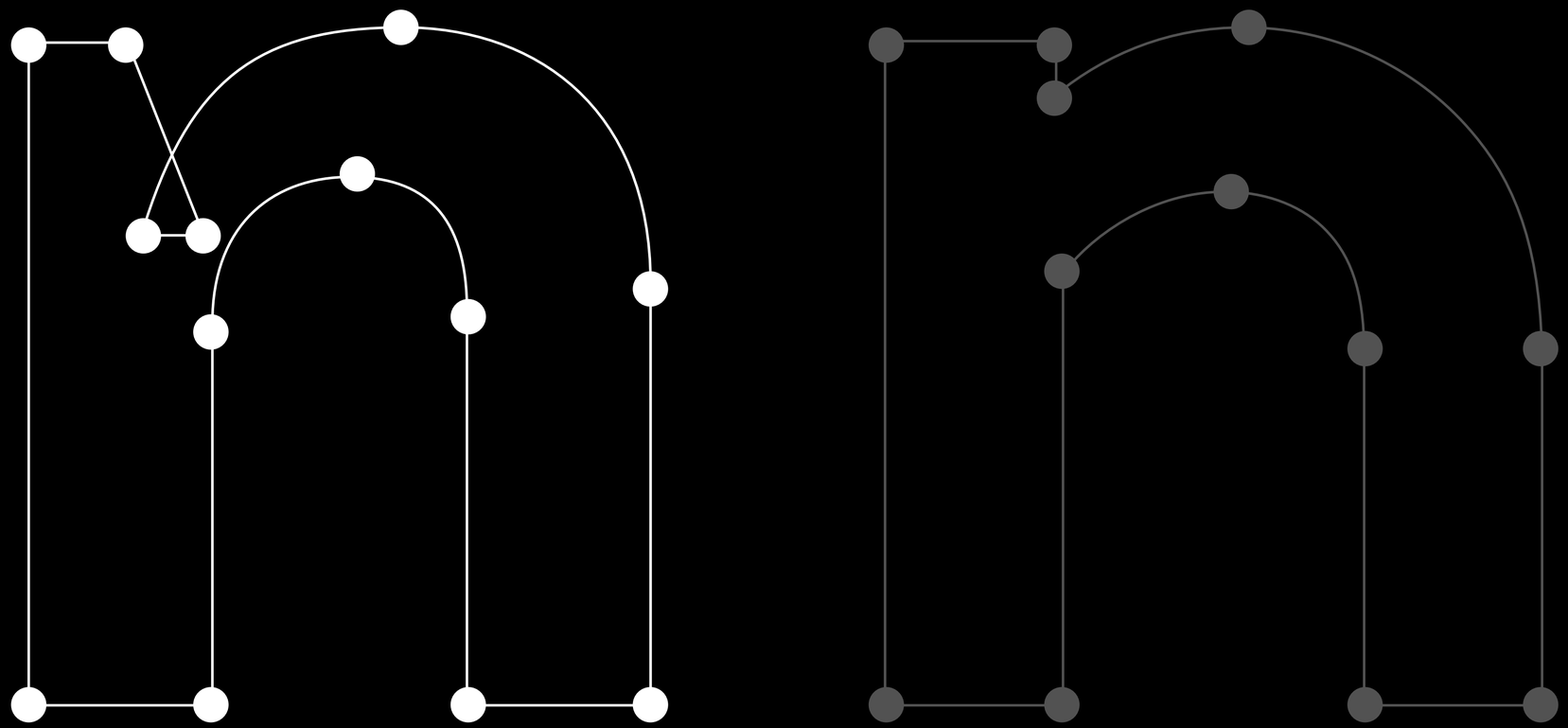
Do what makes sense for your shape.



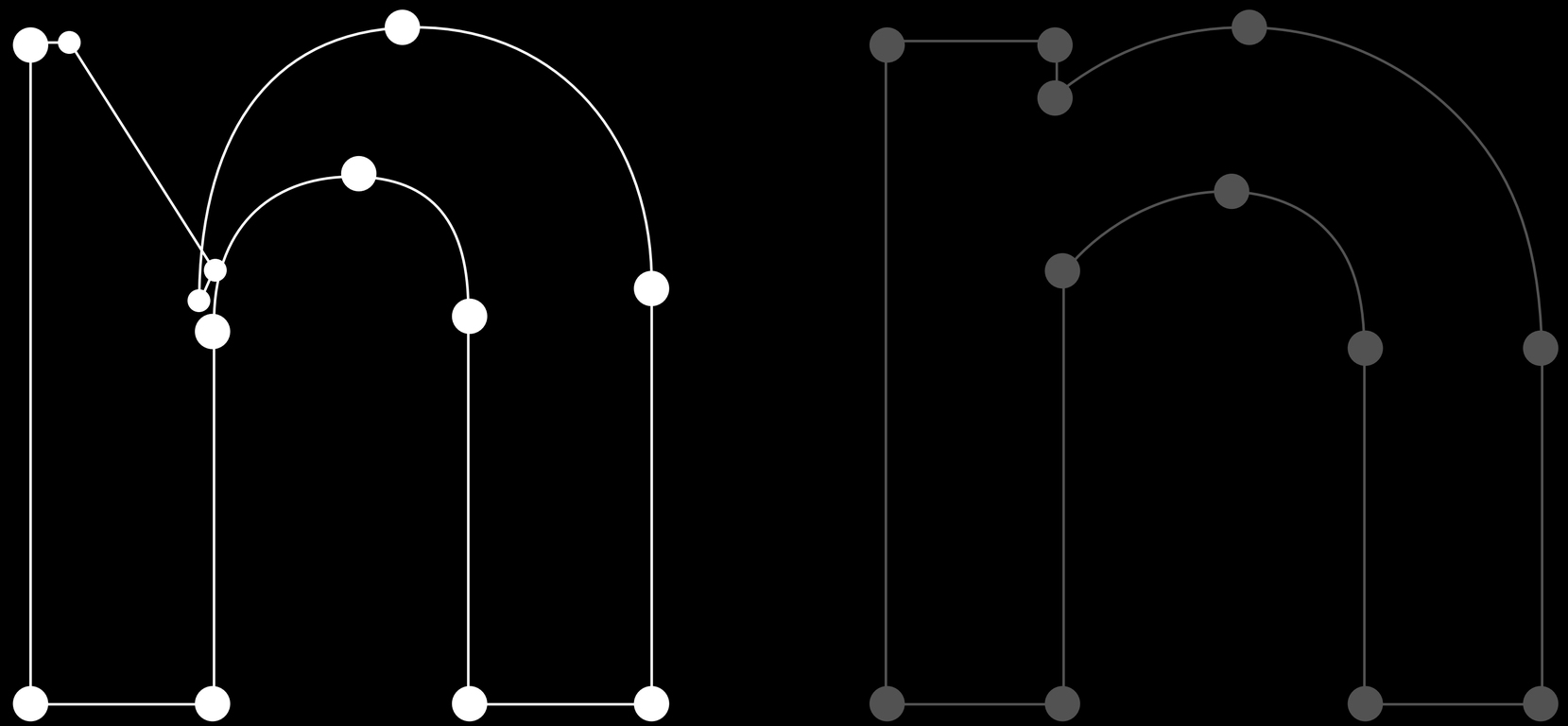
Do what makes sense for your shape.



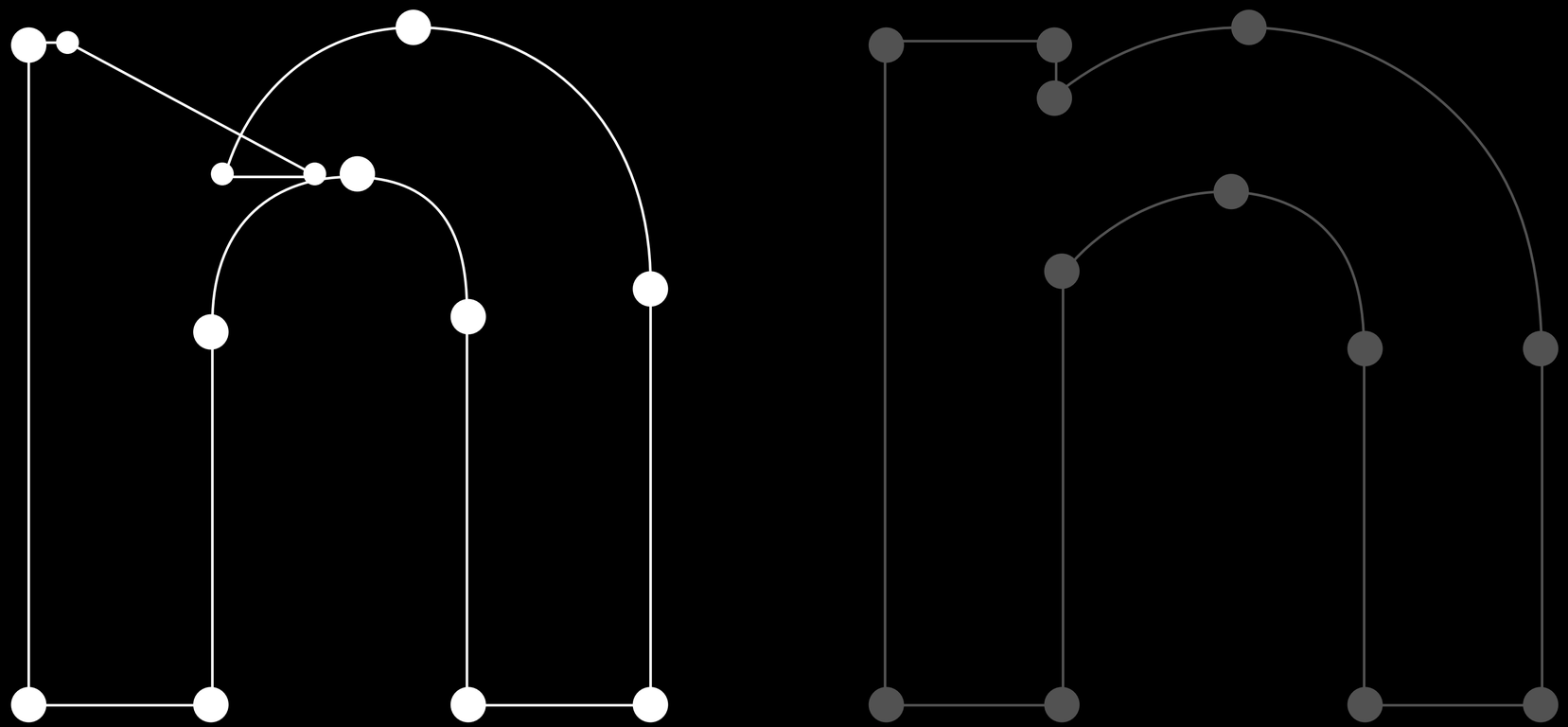
Do what makes sense for your shape.



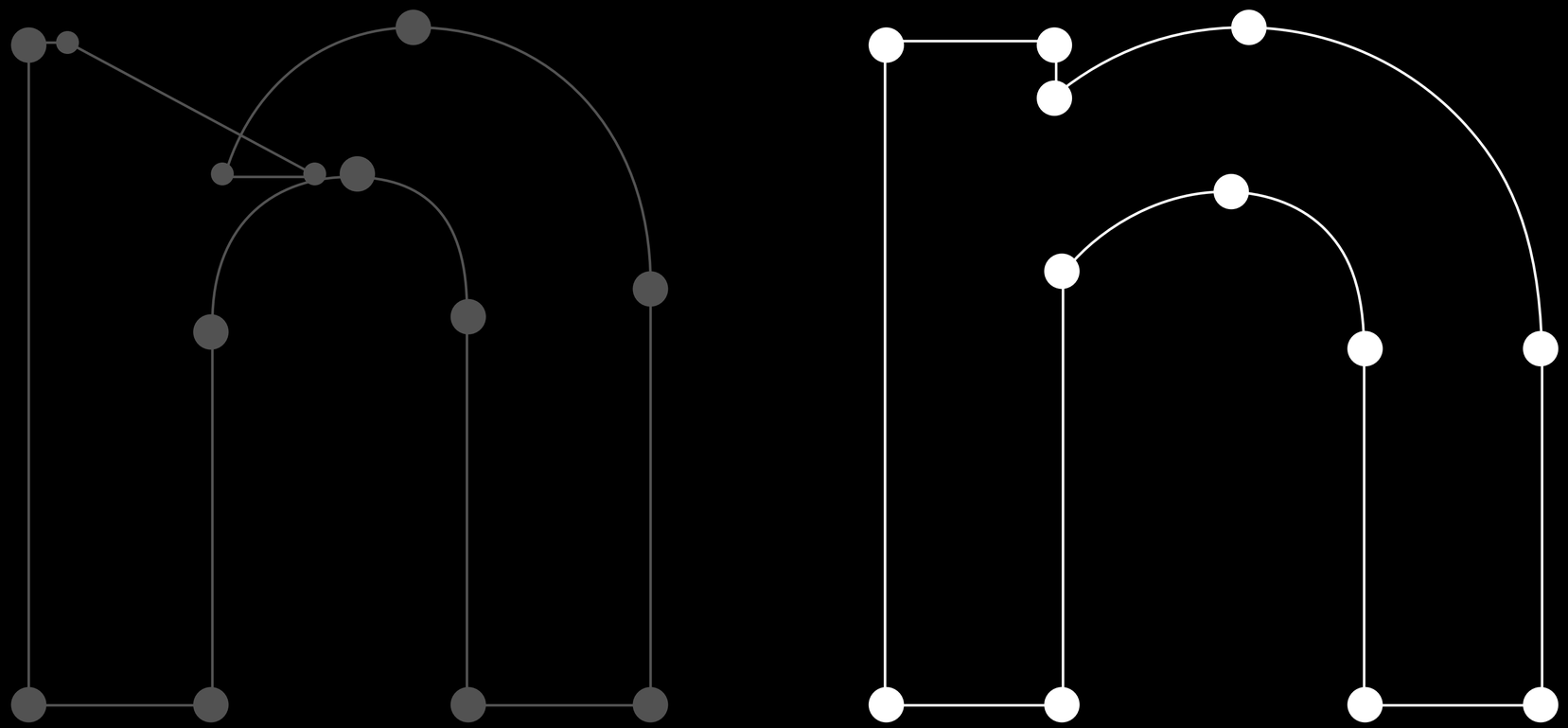
Do what makes sense for your shape.



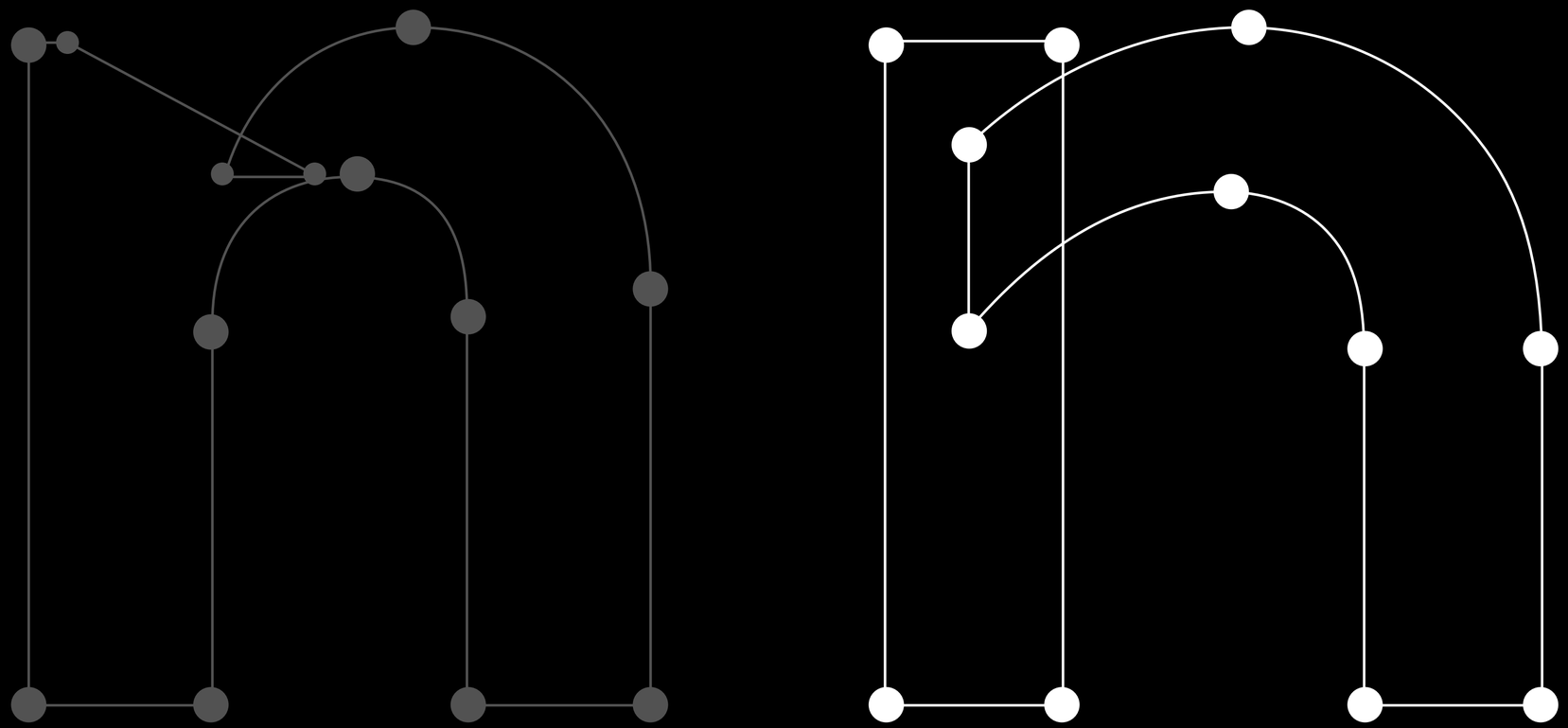
Do what makes sense for your shape.



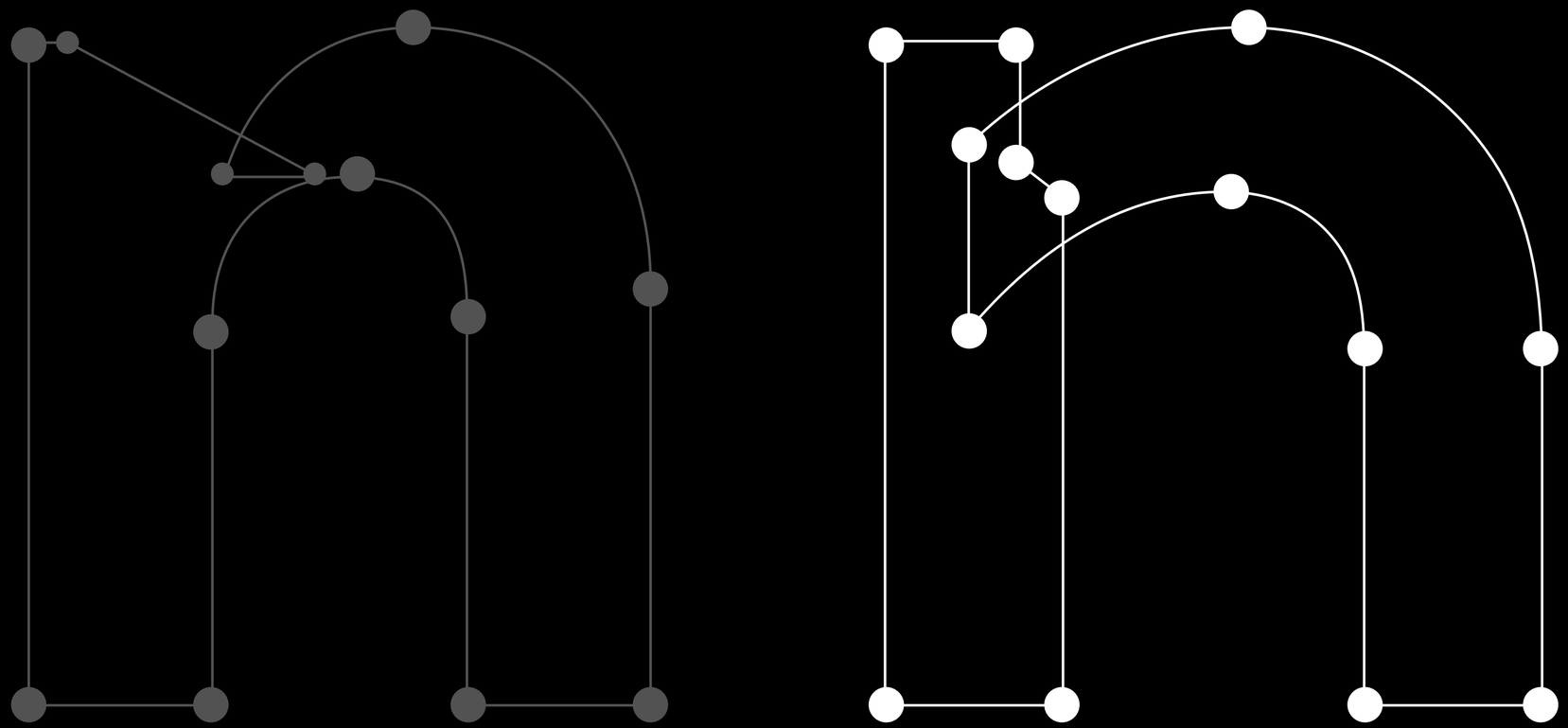
Do what makes sense for your shape.



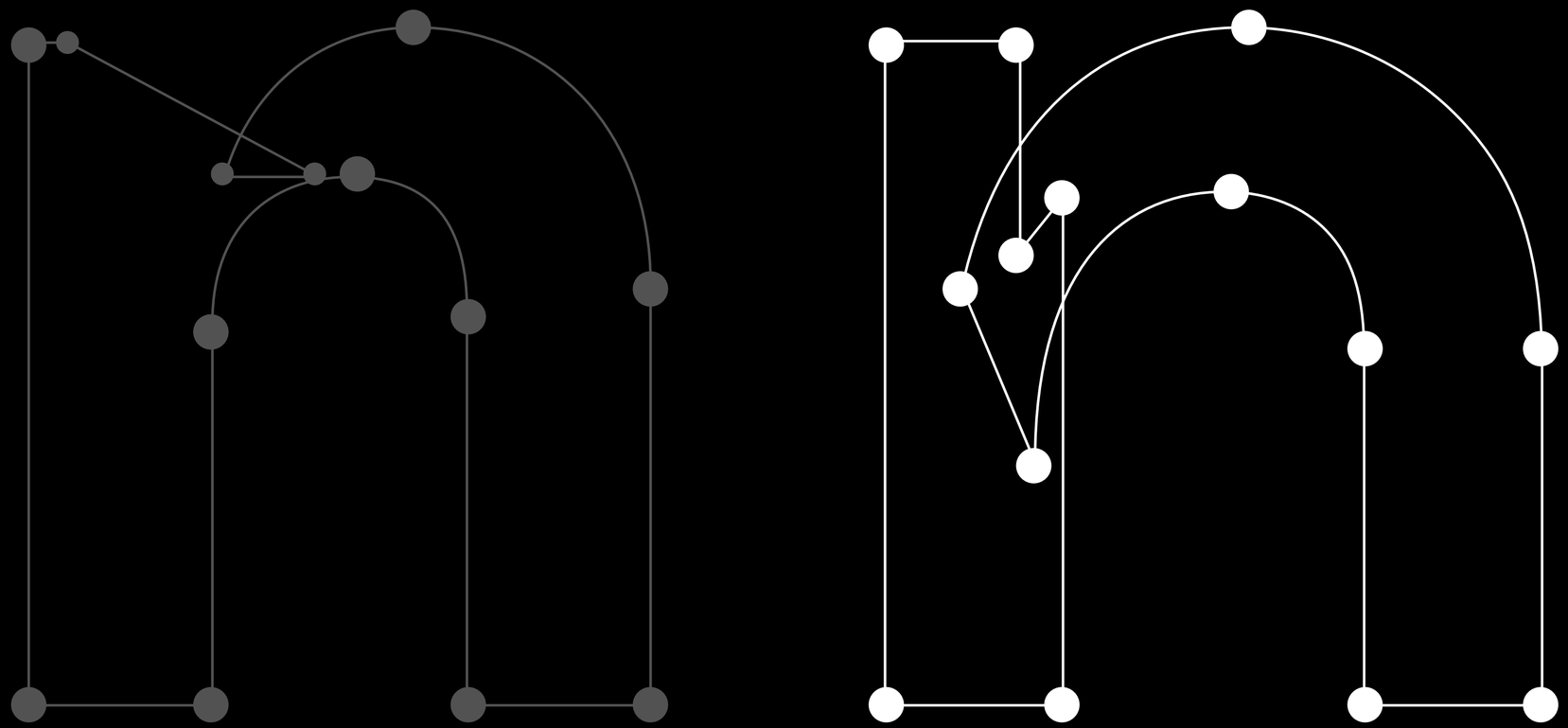
Do what makes sense for your shape.



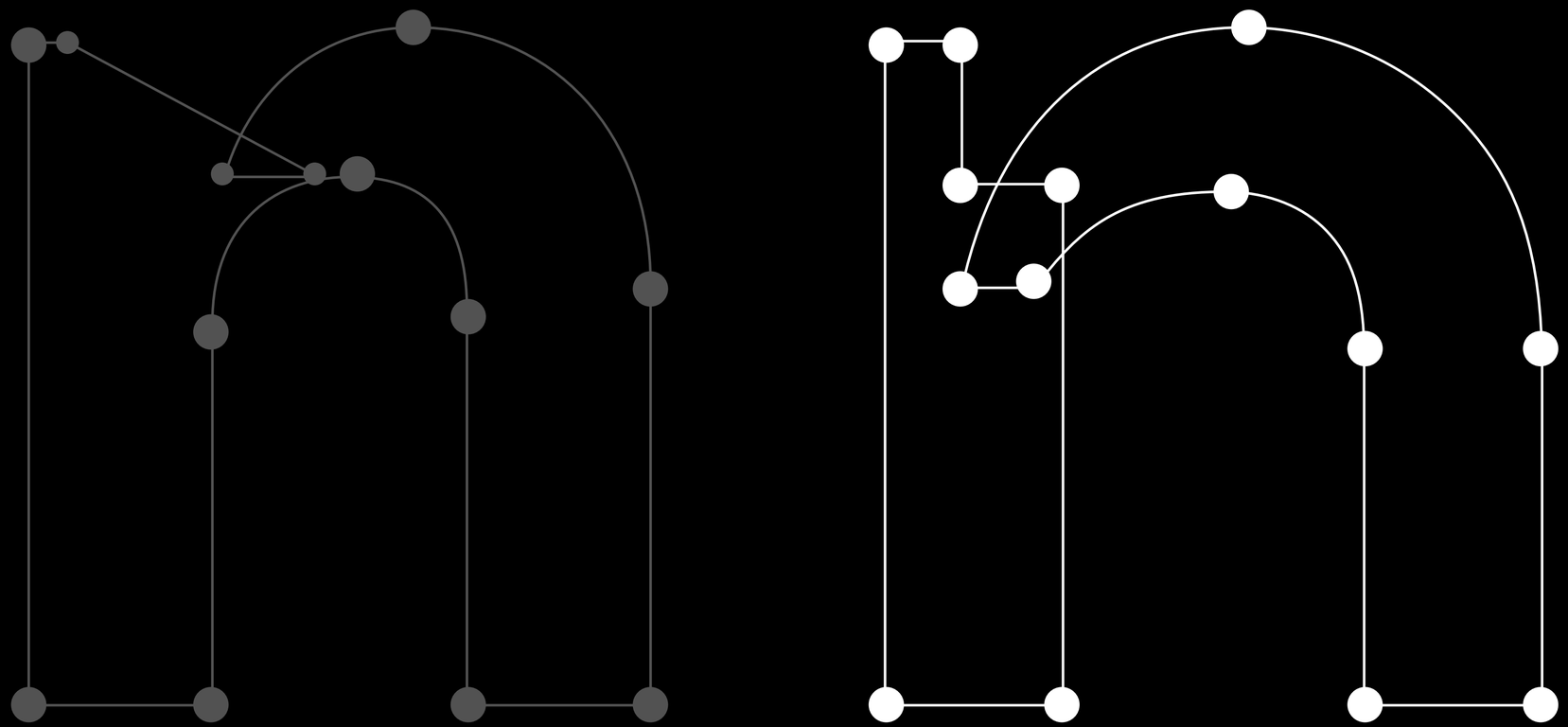
Do what makes sense for your shape.



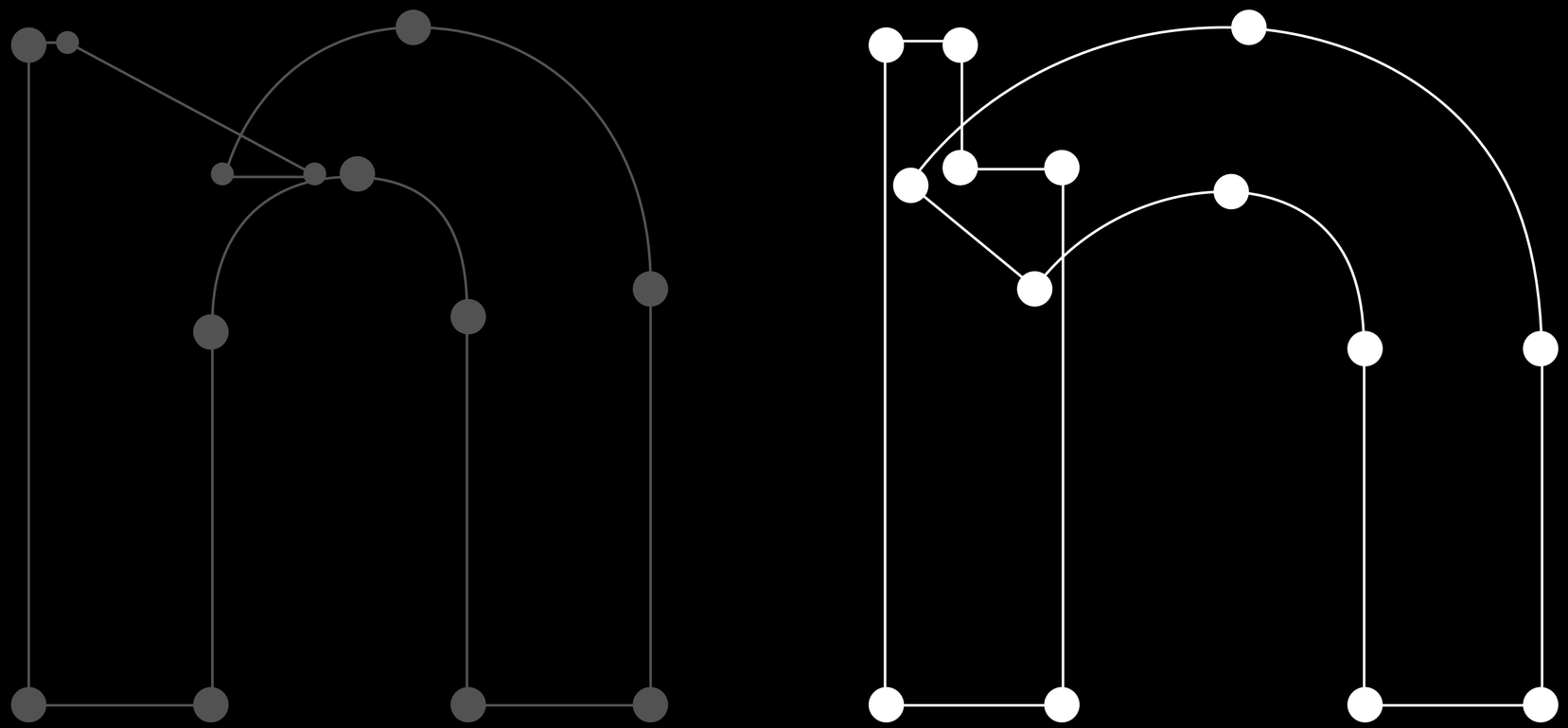
Do what makes sense for your shape.



Do what makes sense for your shape.



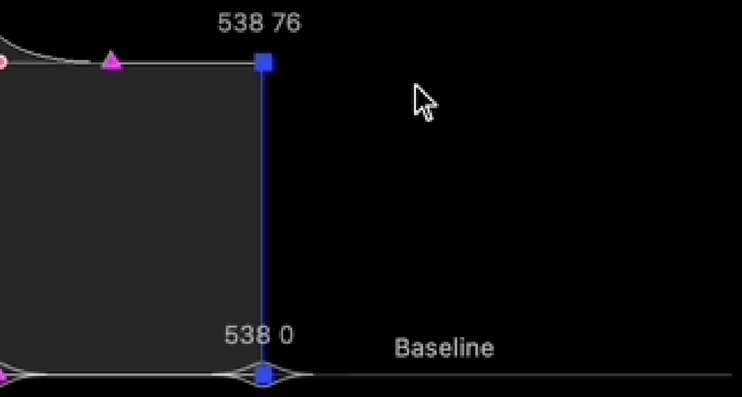
Do what makes sense for your shape.

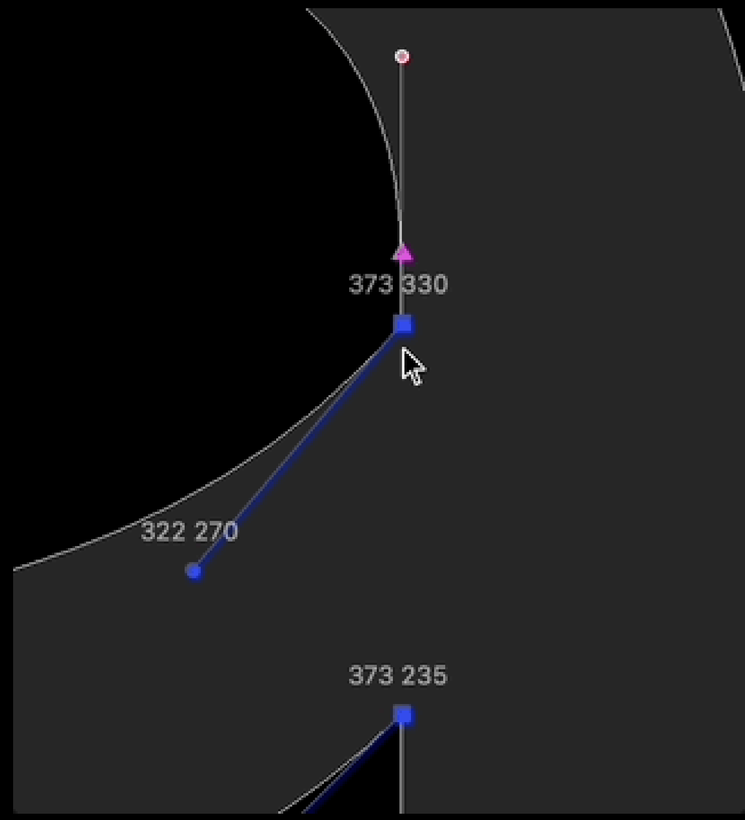


Erik van Blokland's
Drawing for Interpolation Cheatsheet

https://superpolator.com/drawing.html

Overlapper RoboFont Extension





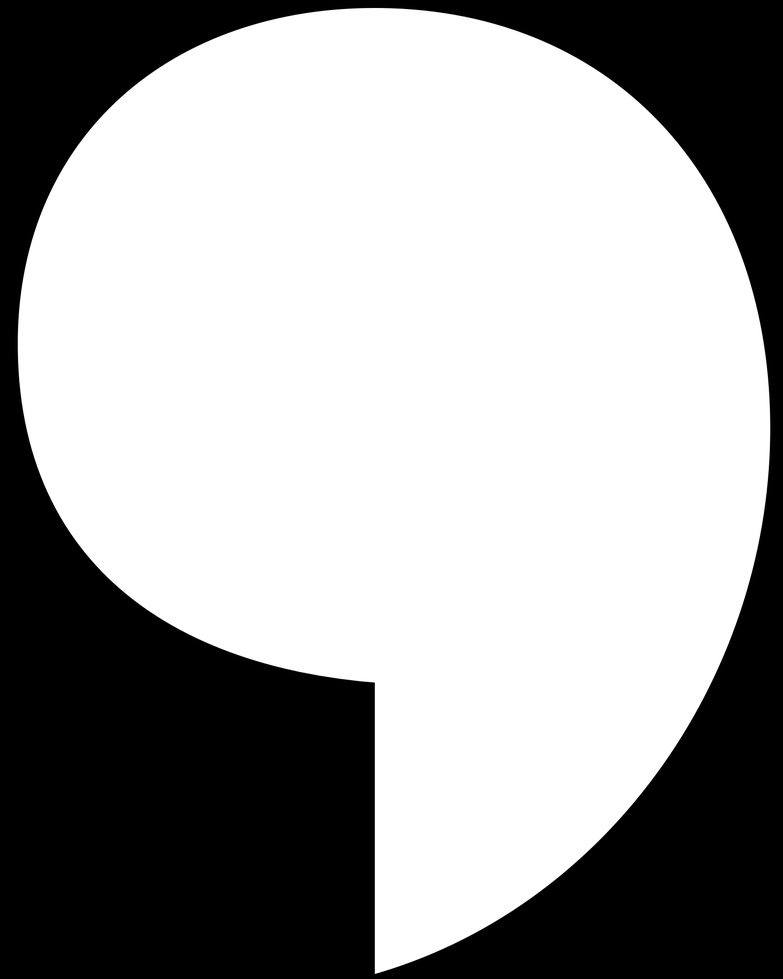
github.com/ryanbugden/Overlapper

Questions?

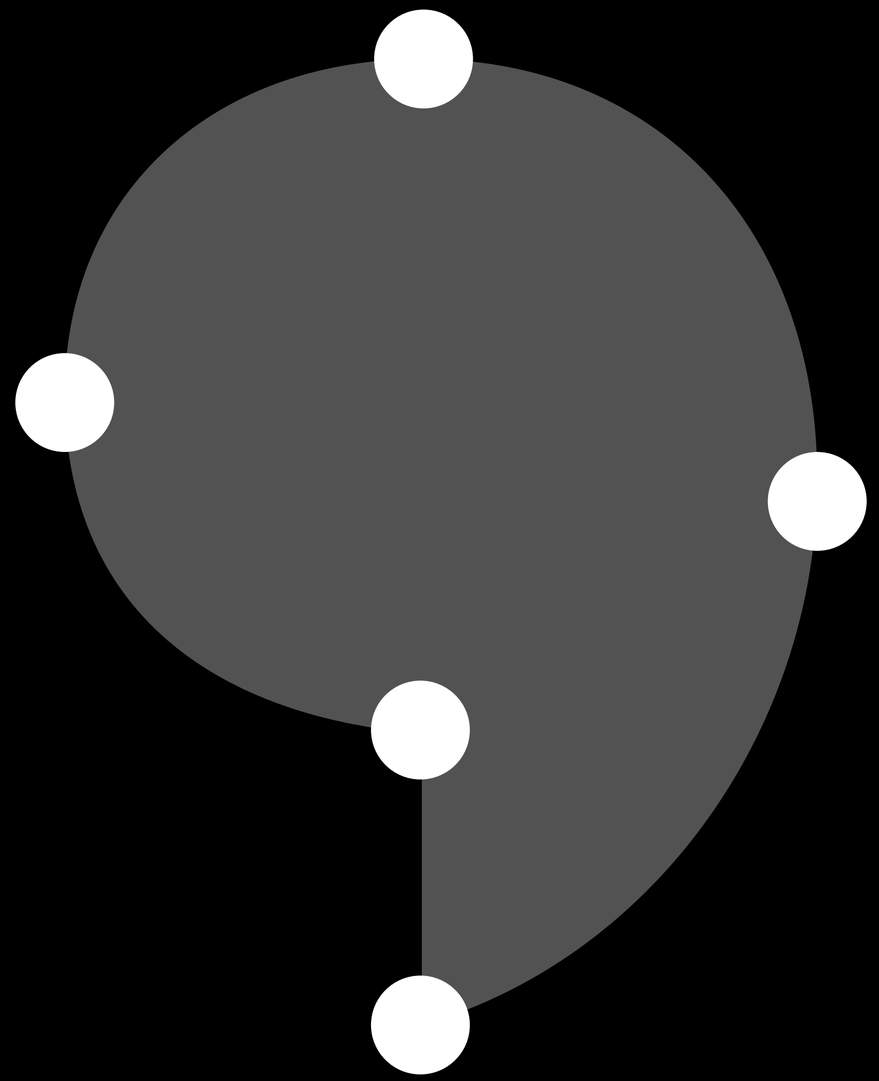
Quick background
Math demo
Tips
Game

Bézier Game is about point placement.

Demo:
Plot on-curve points



Demo:
Yell “Done”
Correct?
Win a point!



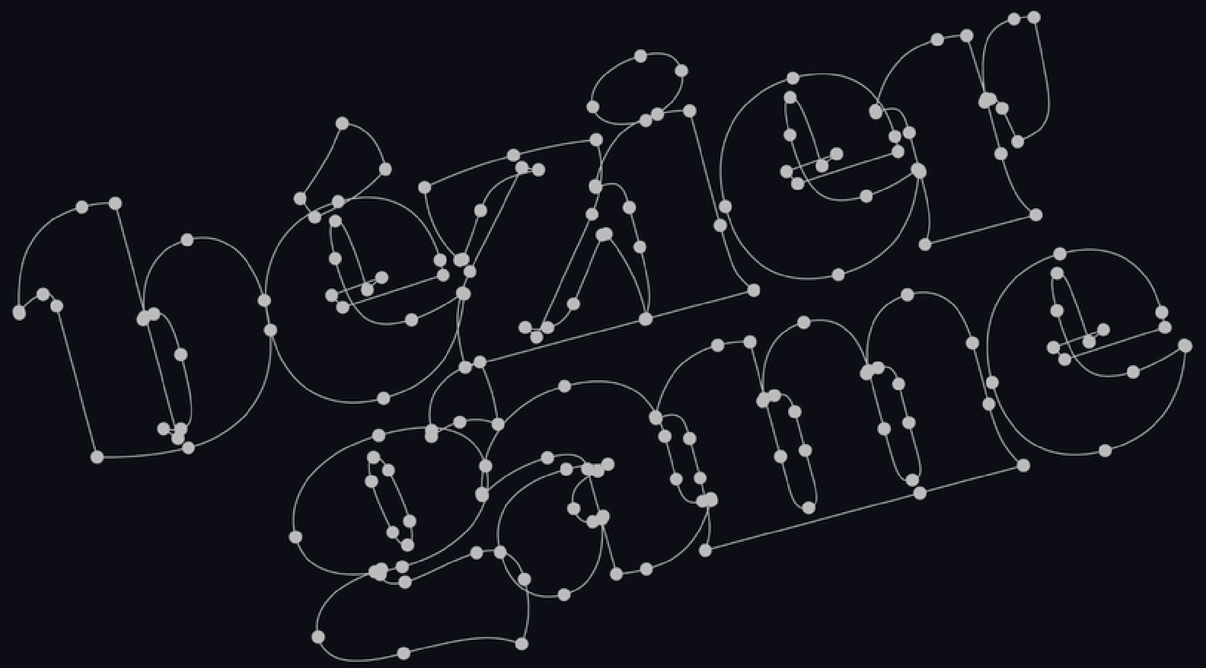
DrawBot now makes the game for us.

As of today, this is all open-source, and available for all educators to use.

README.md

The Bézier Game

This repository hosts the Bézier Game, and educational materials about curves. I'm putting this on GitHub so that educators can have access to this game and play with their students. If you'd still like to fly me to your city to conduct the game, feel free to contact me :)



How to set up your own game:

1. Save/clone this whole repository.
2. Open `generate_bezier_game.py` in DrawBot
3. Set up the title page; change `school_name` and `event_name`
4. Choose your 2 team names; have fun with it!

> *Optional: Change the list of glyphs that might show up in the game. The more obscure or complicated, the more difficult the drawing (usually).*

github.com/ryanbugden/Bezier-Game

The game generator, demos, resources, etc.

Let’s play!