

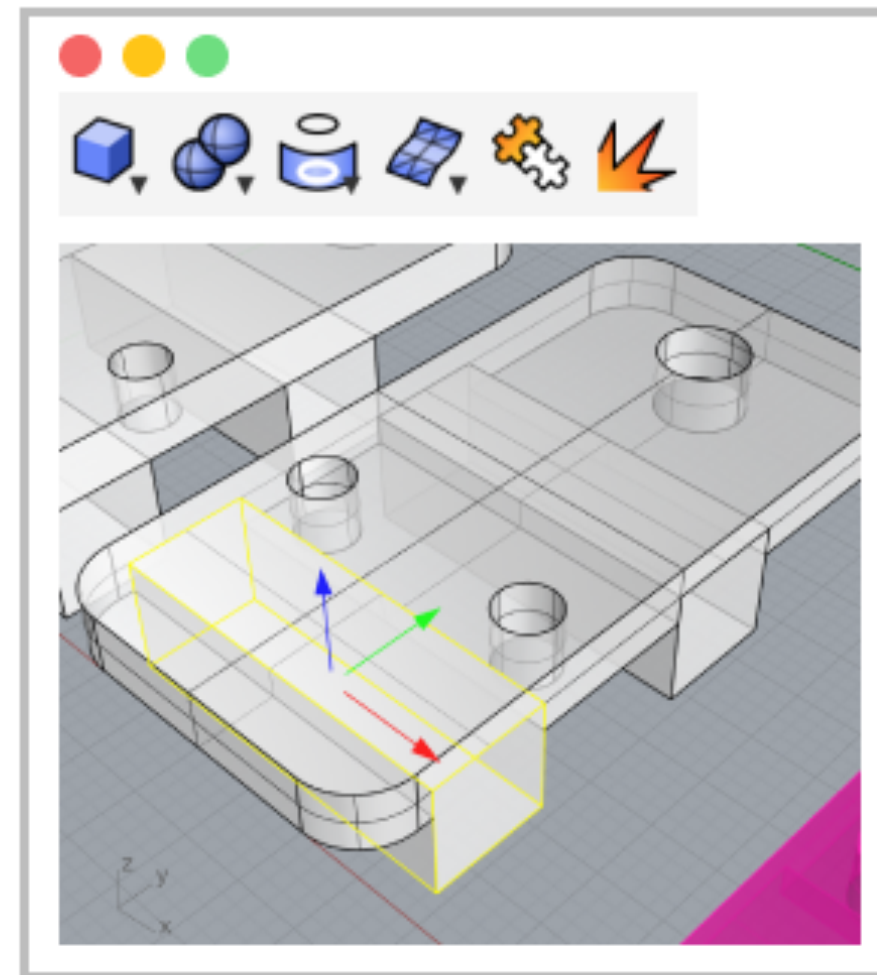
Towards Fabrication-as-Programming

SCF 2022

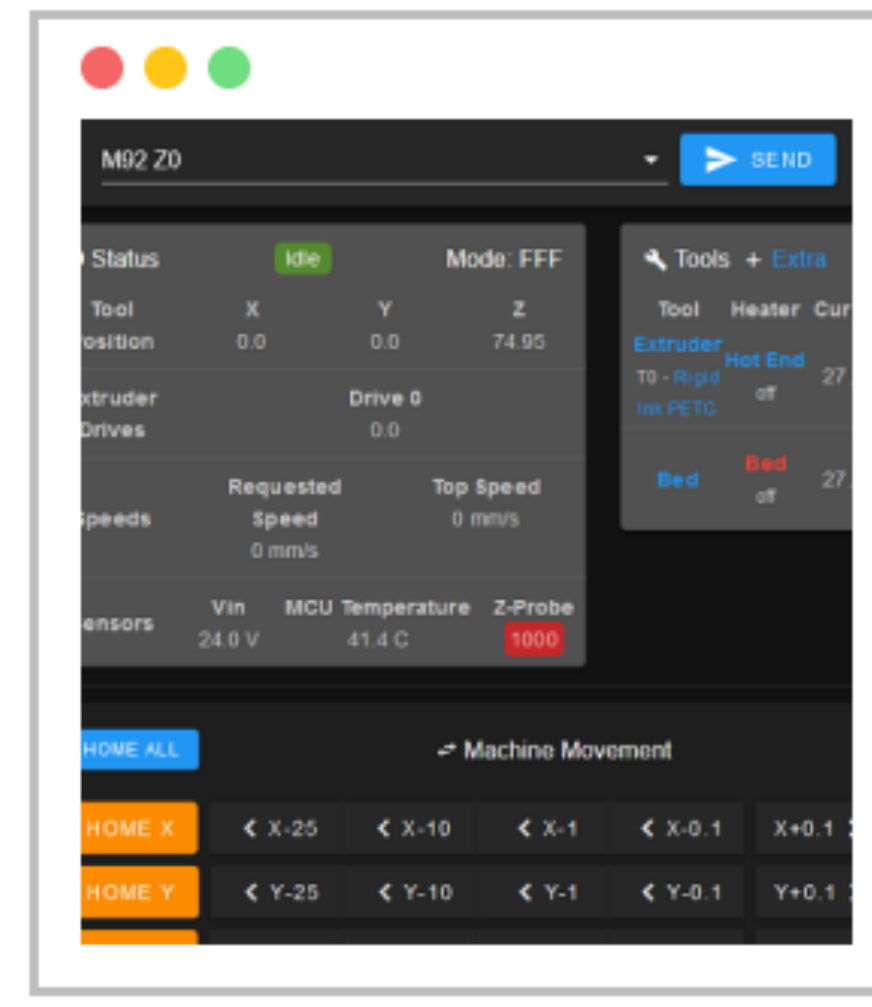
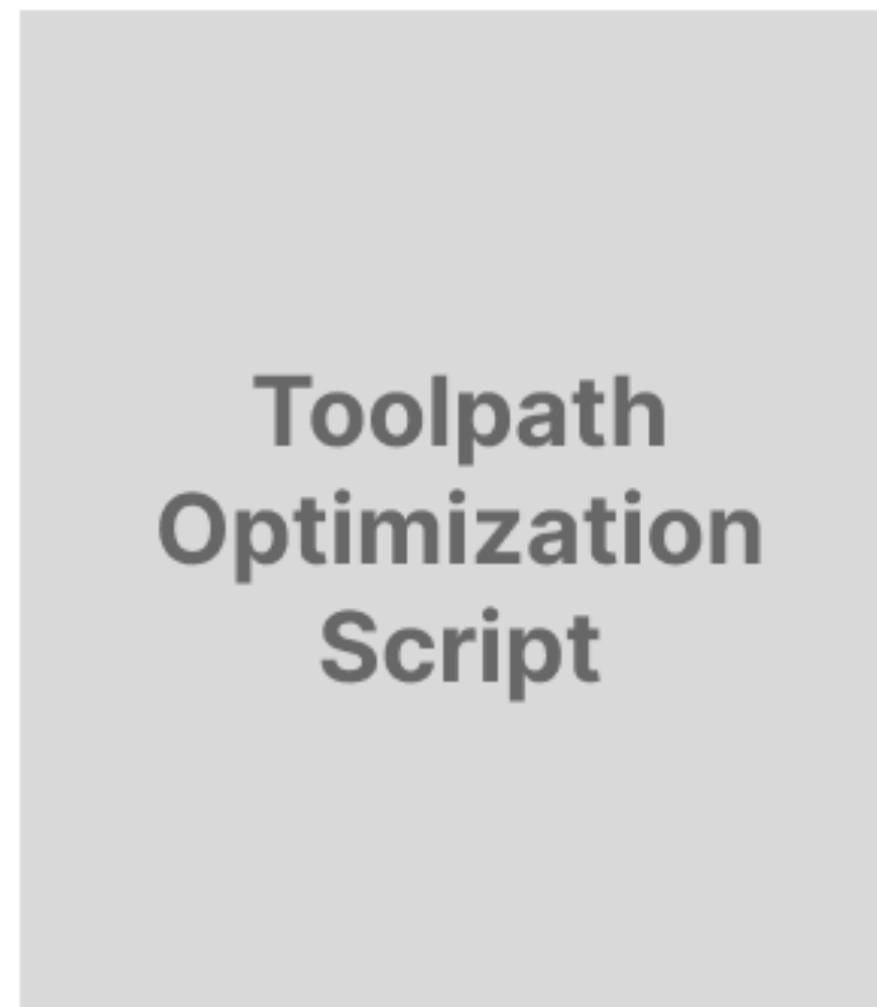
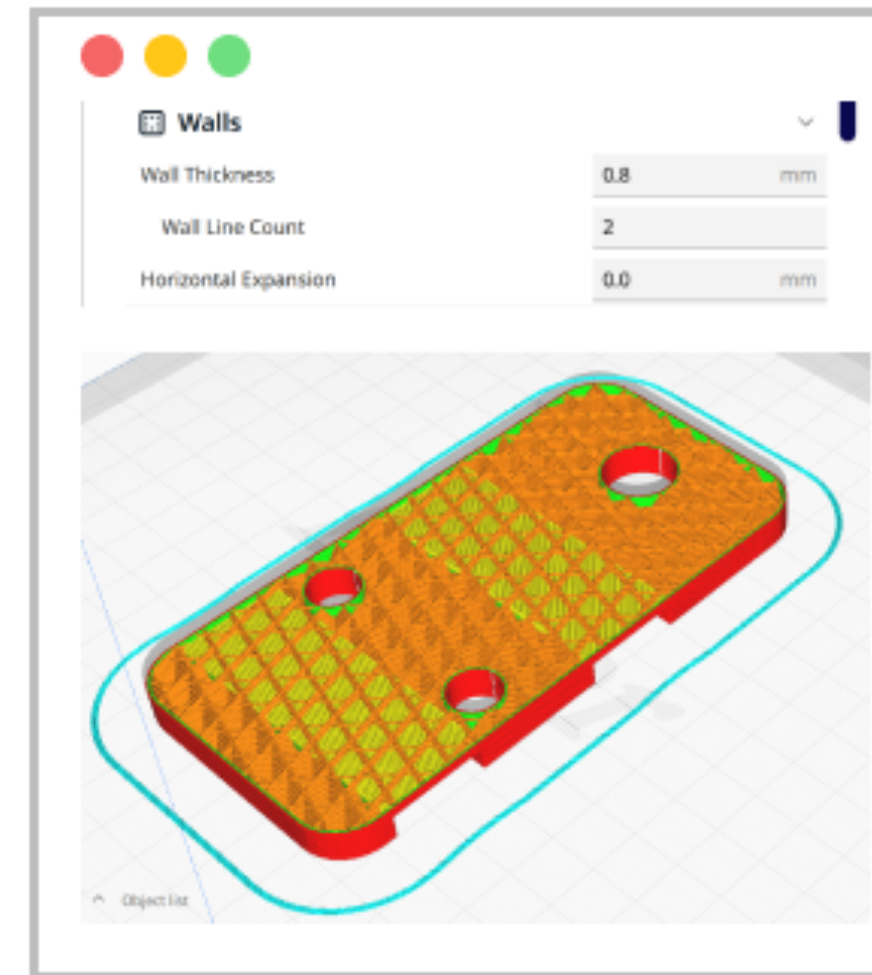
Jasper Tran O’Leary, Eunice Jun, Nadya Peek
University of Washington



CAD



CAM

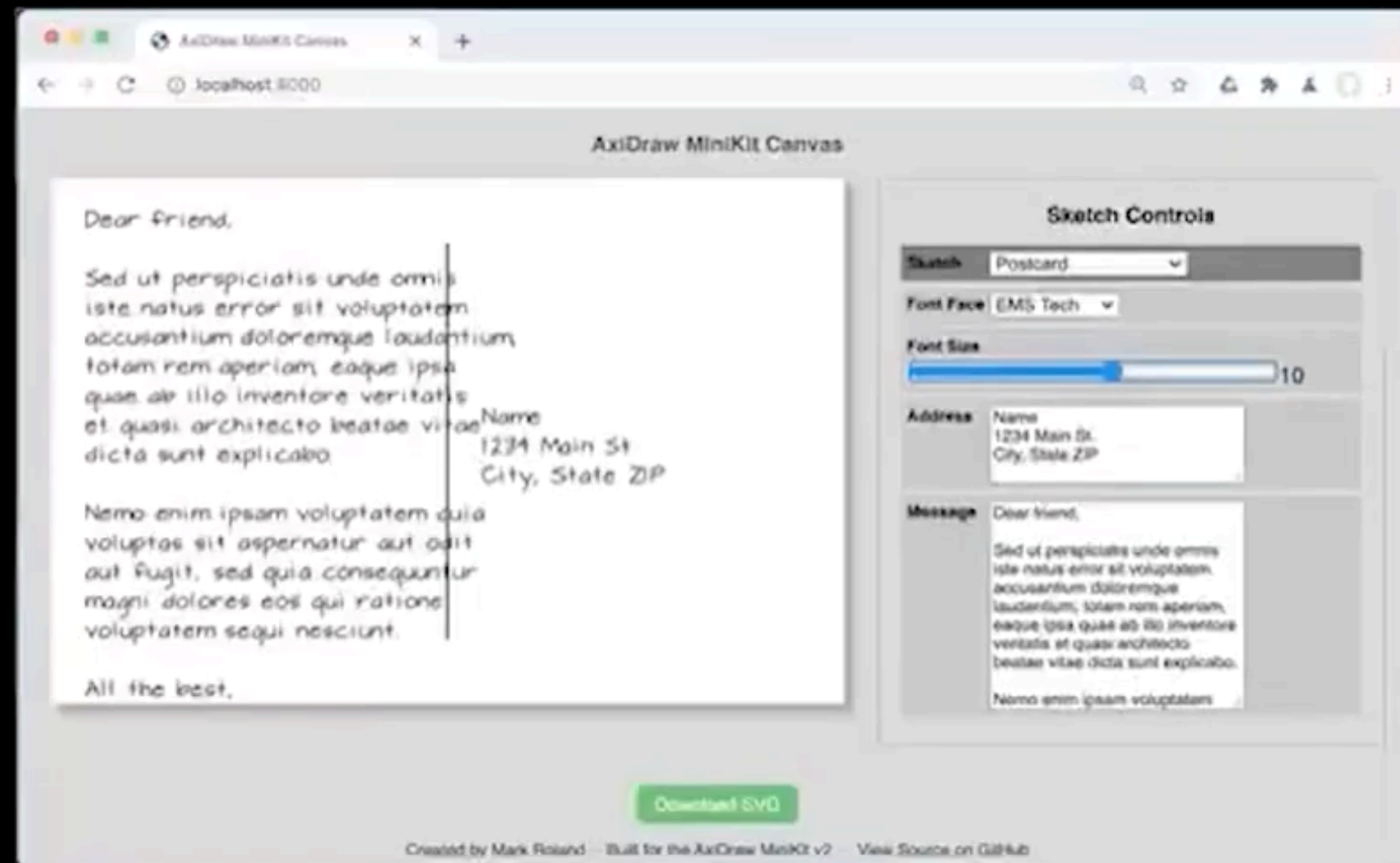


Post-Processing (sometimes)

Machine Control

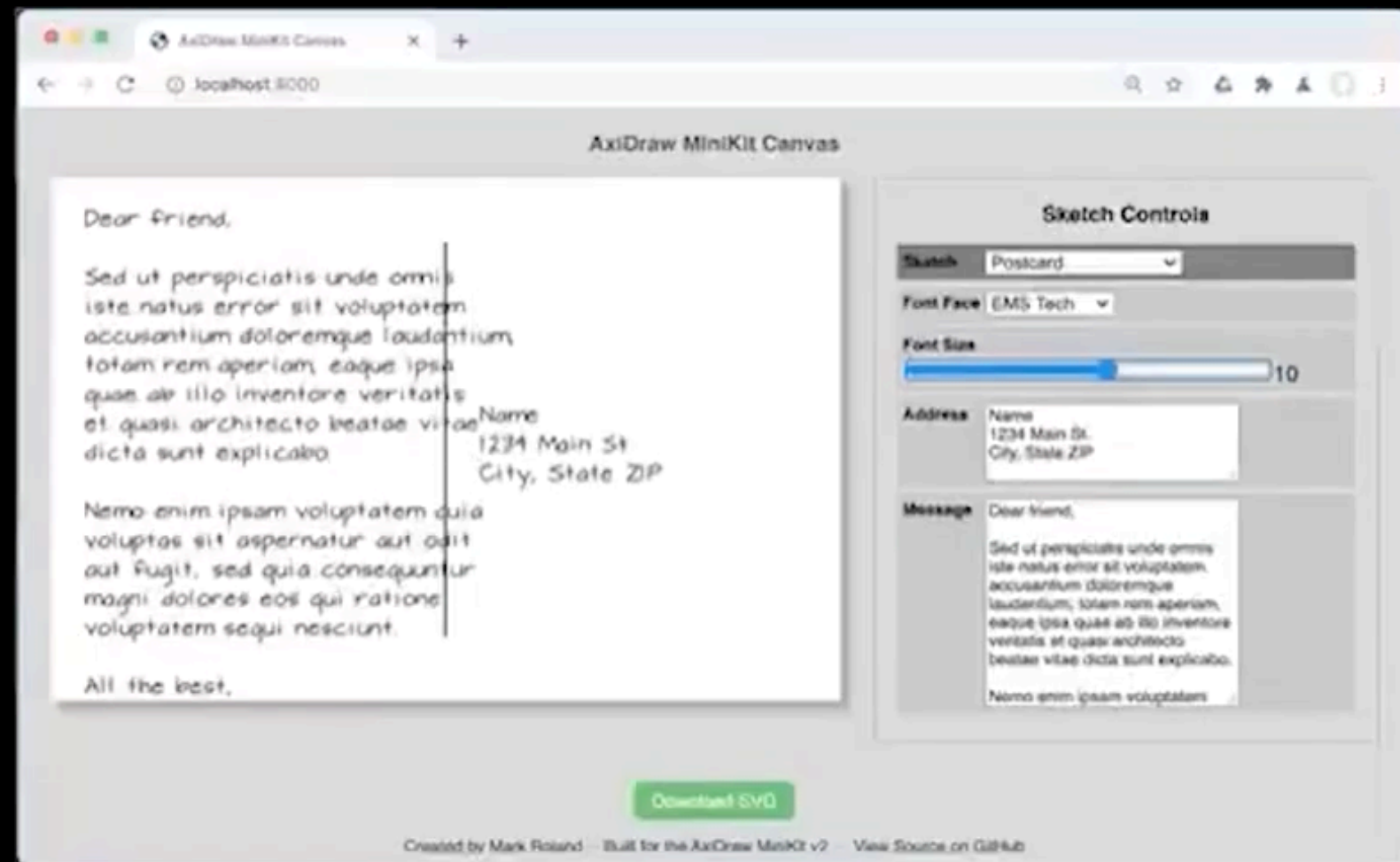
Exploratory digital fabrication

Exploratory digital fabrication



postcard creator, @rolandojones

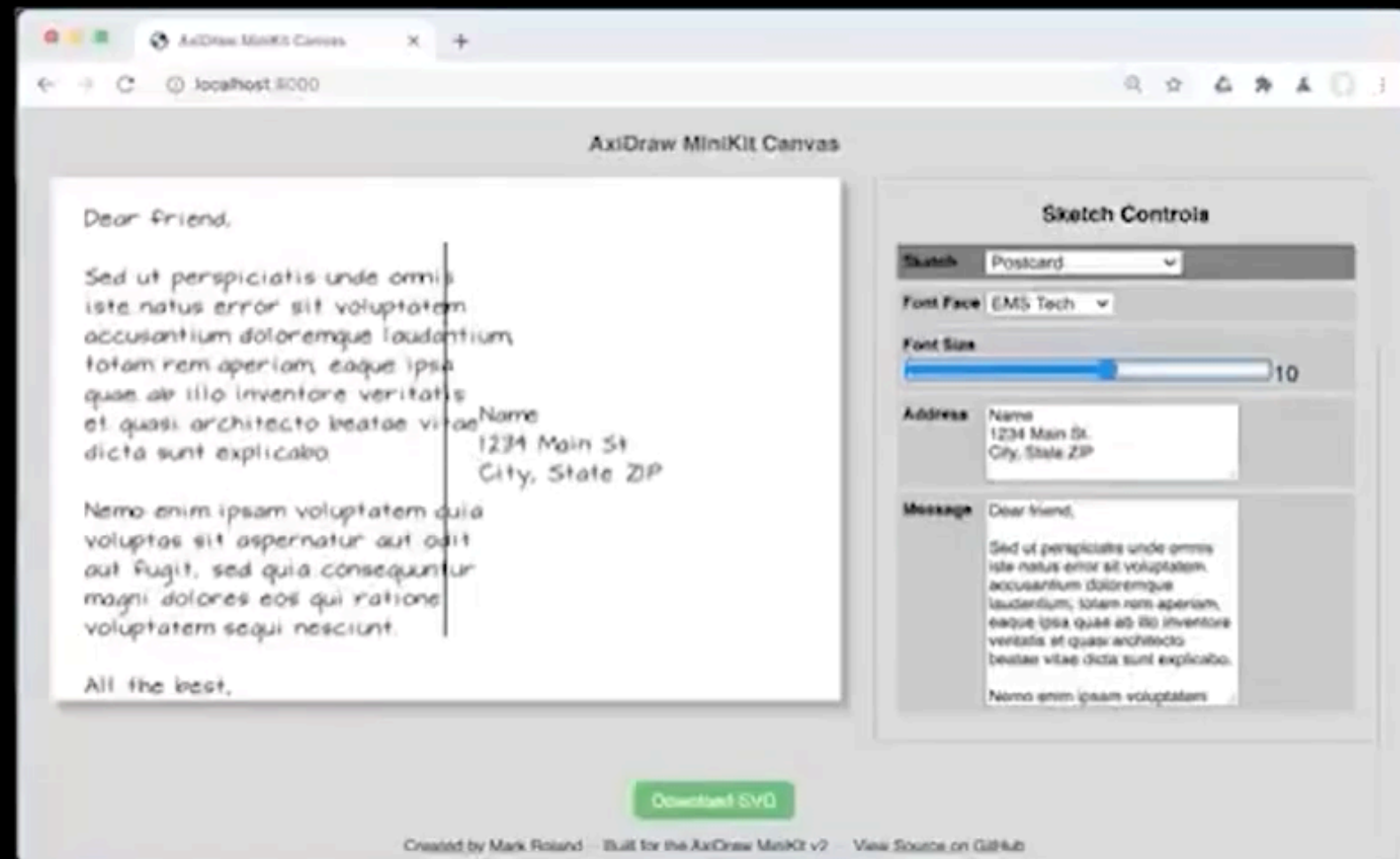
Exploratory digital fabrication



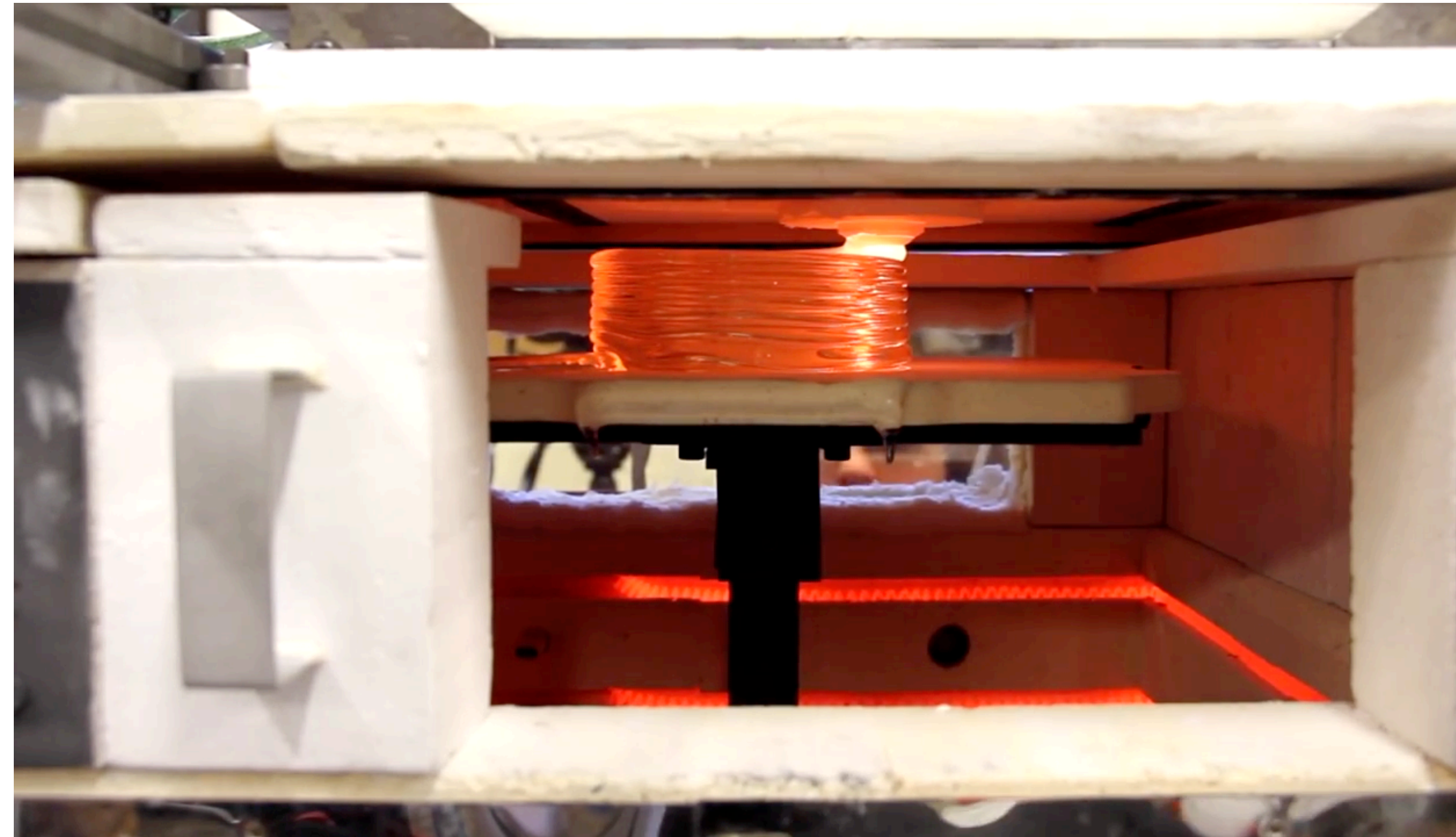
postcard creator, @rolandojones

Twigg-Smith et al. Tools, Tricks, and Hacks. Investigating Novel Digital Fabrication Workflows on #PlotterTwitter. 2021.

Exploratory digital fabrication

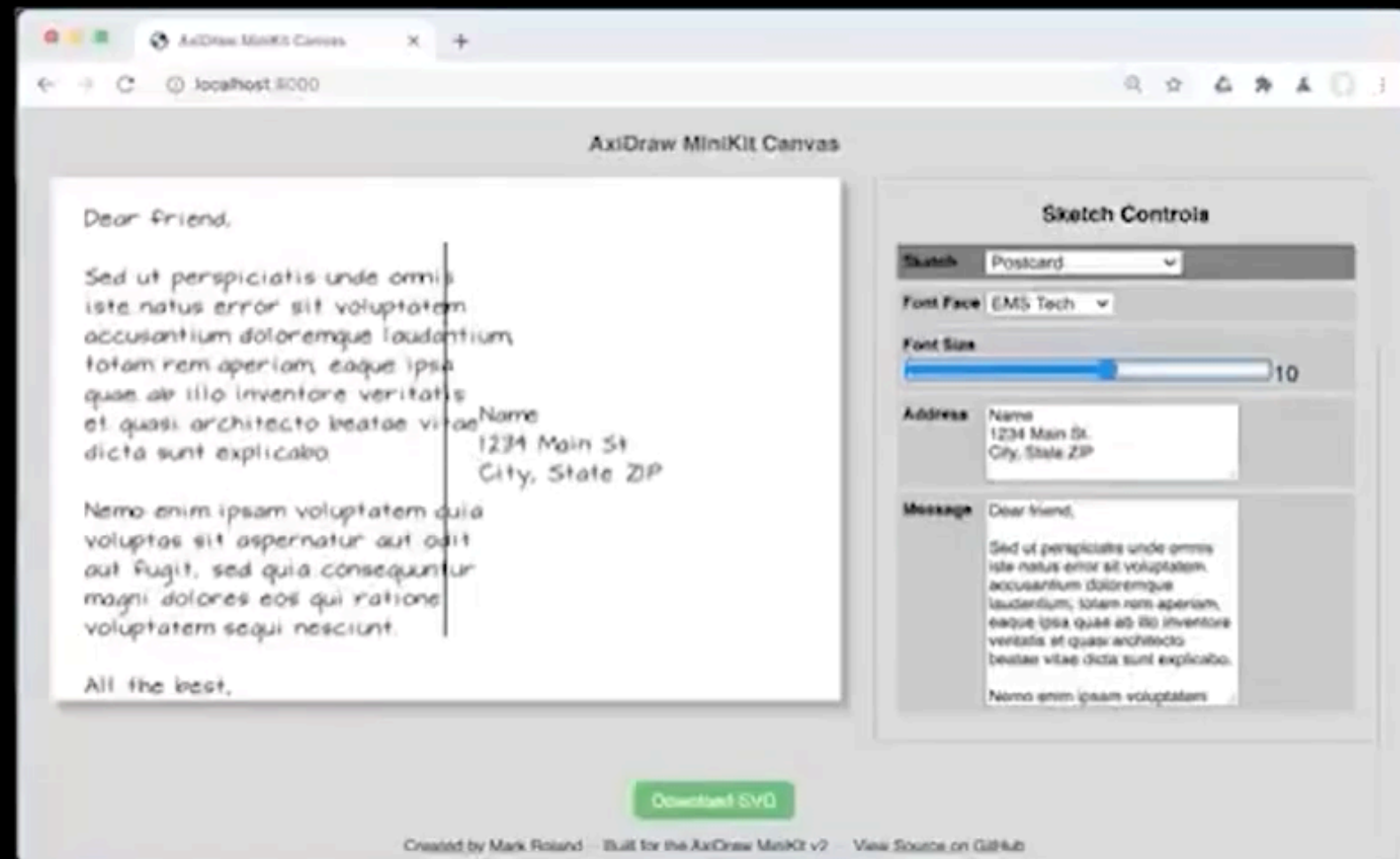


postcard creator, @rolandojones



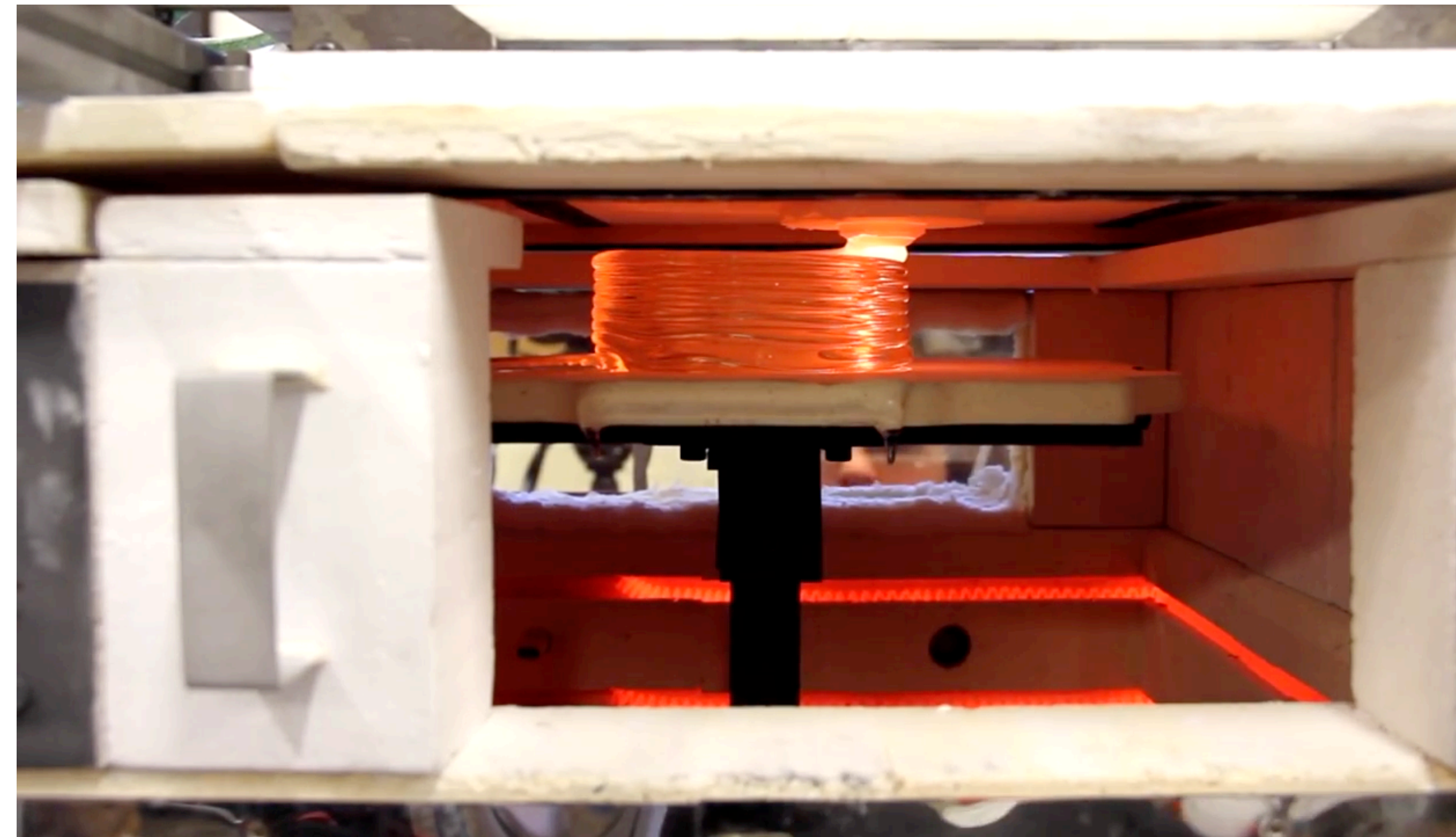
Twigg-Smith et al. Tools, Tricks, and Hacks. Investigating Novel Digital Fabrication Workflows on #PlotterTwitter. 2021.

Exploratory digital fabrication



postcard creator, @rolandojones

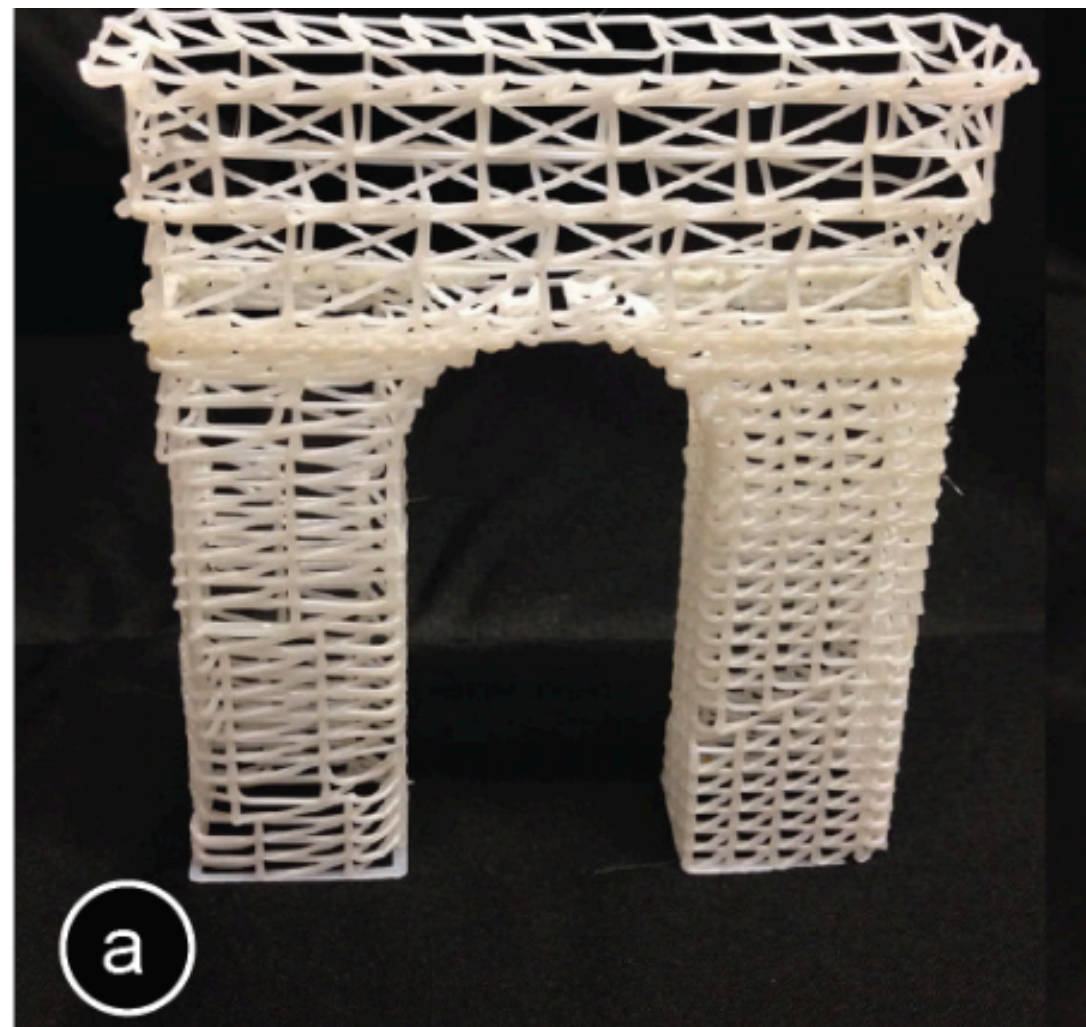
Twigg-Smith et al. Tools, Tricks, and Hacks. Investigating Novel Digital Fabrication Workflows on #PlotterTwitter. 2021.



Klein et al. Additive Manufacturing of Optically Transparent Glass. 2015.

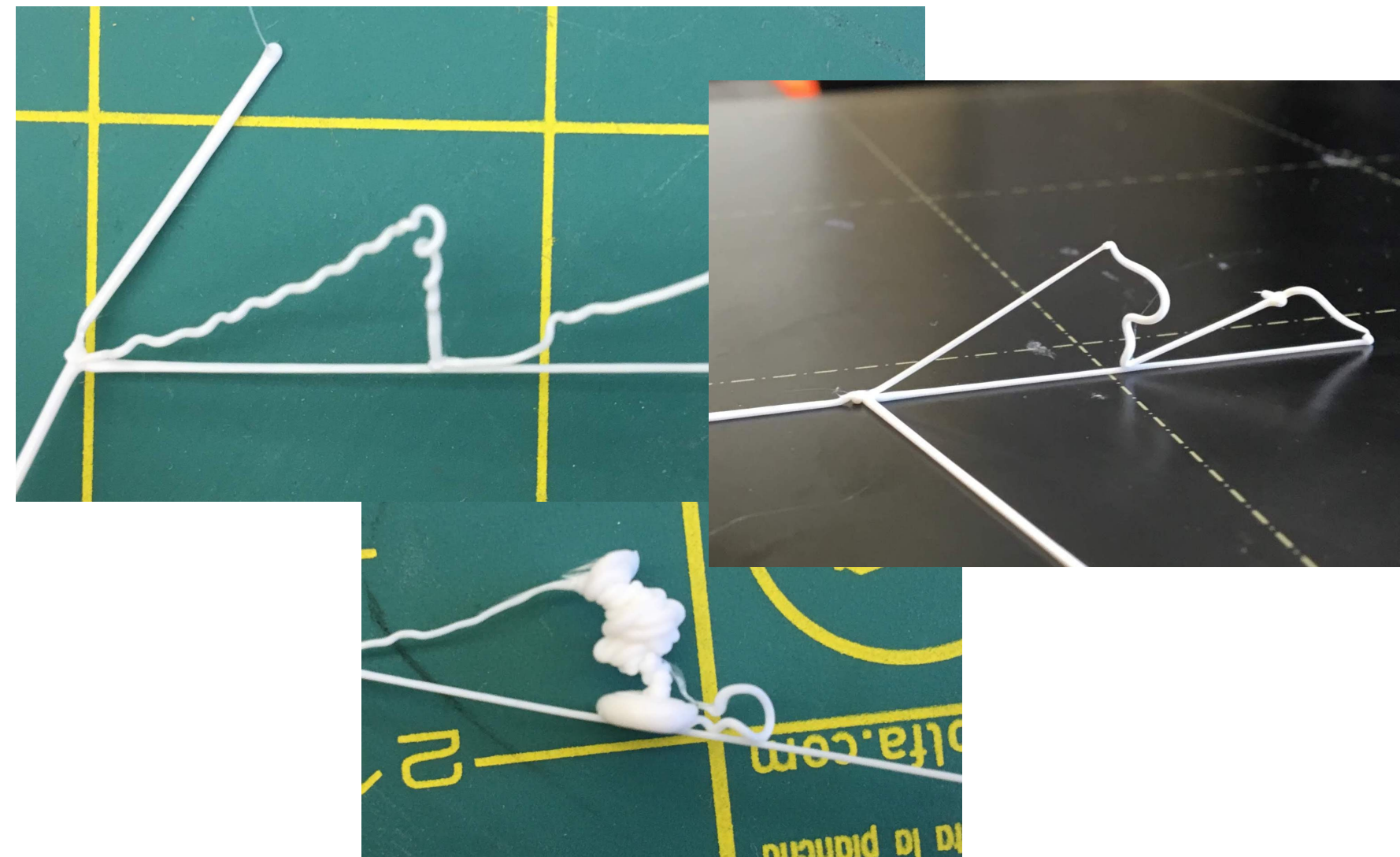
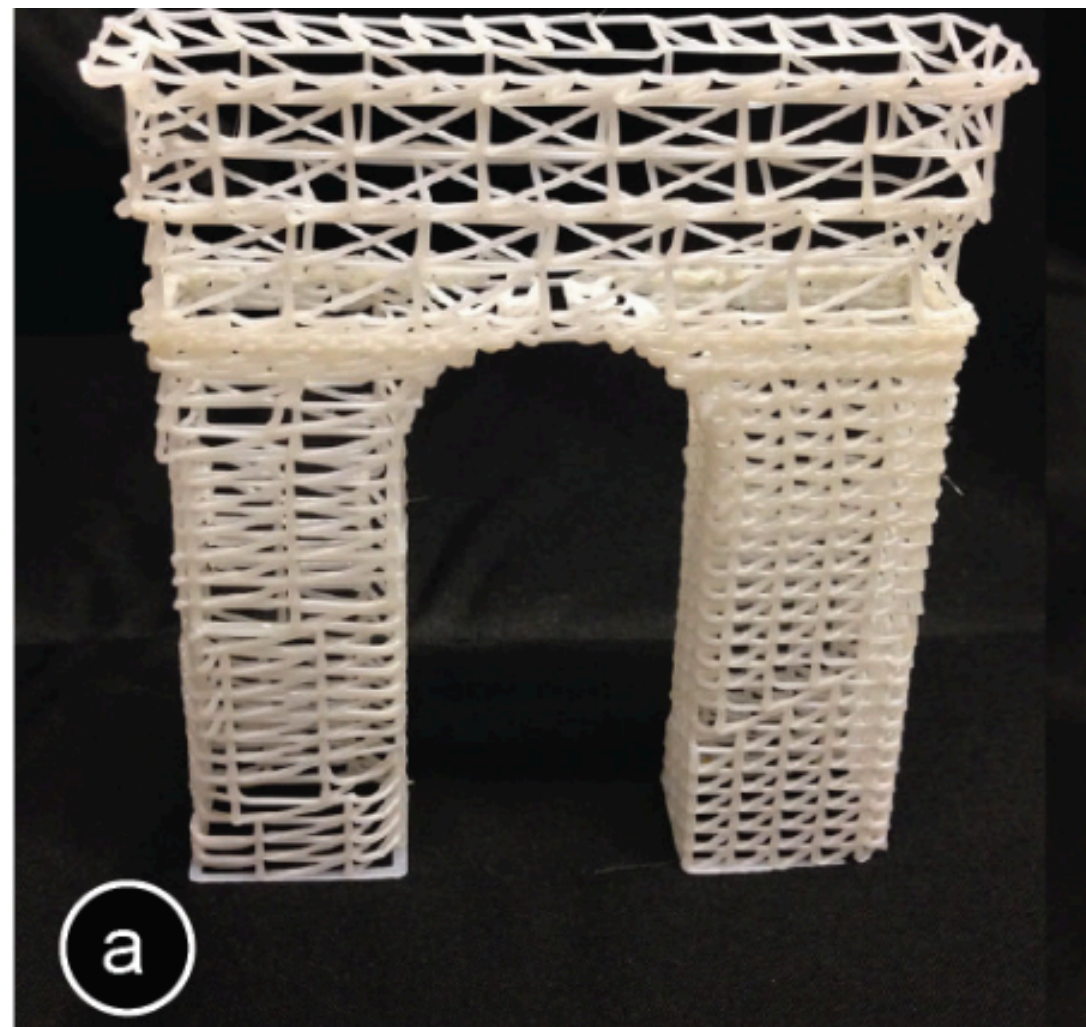
Programming for Exploratory Fab is Painful with Current Tools

Programming for Exploratory Fab is Painful with Current Tools



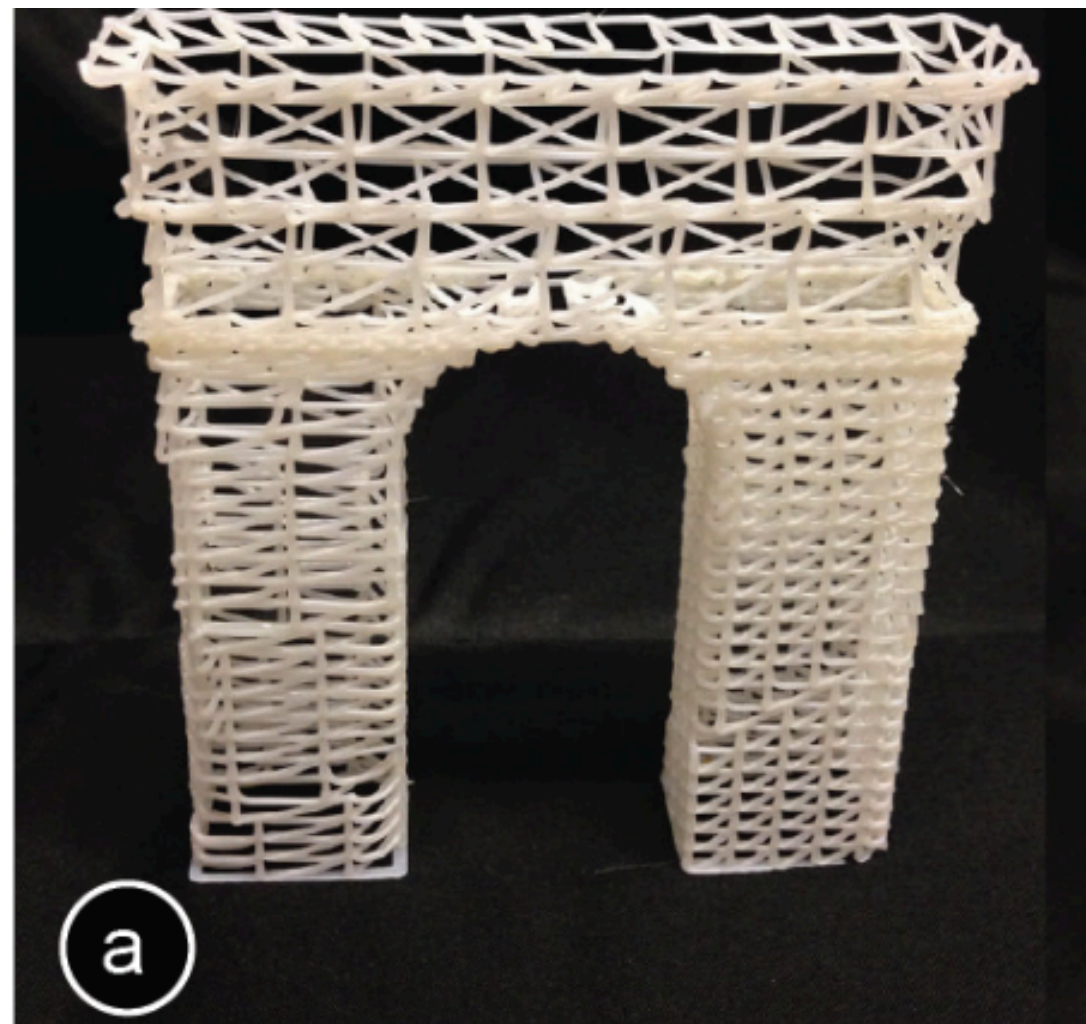
Mueller et al. Wireprint. 2014.

Programming for Exploratory Fab is Painful with Current Tools

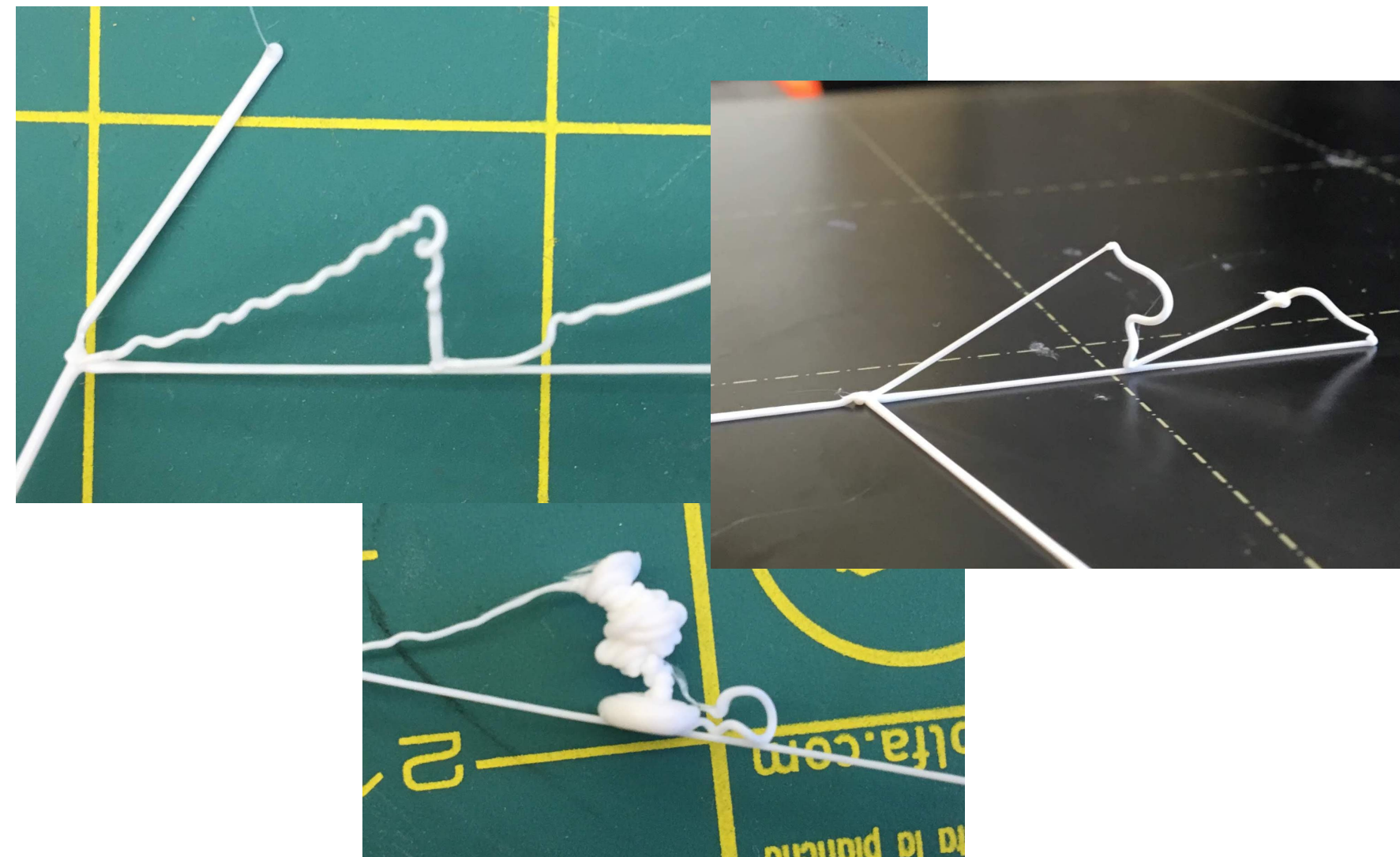


Mueller et al. Wireprint. 2014.

Programming for Exploratory Fab is Painful with Current Tools



Mueller et al. Wireprint. 2014.



Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Single fin v1

```
G1 X100 Y100 Z0 F2000  
G1 X130 Y100 Z0 E4 F500 ; base  
G1 Z20 E4.00 F50 ; vertical side  
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000  
G1 X120 Y100 Z0 E2 F500 ; base  
G1 Z10 E2.00 F50 ; vertical side  
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000  
G1 X120 Y100 Z0.2 E2.5 F500 ; base  
G1 Z10 E2.00 F50 ; vertical side  
G1 E0.2 F500 ; preemptive extrude before hypotenuse  
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

*“While both GUIs and languages ... [make] it easy to say common things, **a language empowers users to say uncommon things too.**”*

—Chasins et al. “PL and HCI: better together.” Comm. ACM ’21.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical side
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Functions?

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical side
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Functions?

Single fin v1

Visualization?

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; ba
G1 Z20 E4.00 F50 ; vertical side
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Functions?

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Visualization?

Testing?

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Functions?

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Visualization?

Testing?

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

Previewing in-situ?

This is doing well, but try to make hypotenuse thicker without buckling

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Functions?

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Visualization?

Testing?

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

Previewing in-situ?

This is doing well, but try to

*Interoperating with
outside libraries?*

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical side
G1 E0.2 F500 ; preemptive extrude before hypotenuse
G1 X100 Y100 Z0.2 E2. F50 ; hypotenuse
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; base
G1 Z20 E4.00 F50 ; vertical
G1 X100 Y100 Z0 E4 F50 ; hypotenuse
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ; base
G1 Z10 E2.00 F50 ; vertical
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try to

```
G1 X100 Y100 Z0.2 F2000
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vertical
G1 E0.2 F500 ; preemptive
G1 X100 Y100 Z0.2 E2. F500
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

Functions?

Visualization?

Testing?

Previewing in-situ?

*Interoperating with
outside libraries?*

*Adapting to different
machines?*

We need more programming power than raw G-Code

Max length vertical with rate

```
G1 Z20 E4.00 F50
```

Single fin v1

```
G1 X100 Y100 Z0 F2000
G1 X130 Y100 Z0 E4 F500 ; ba
G1 Z20 E4.00 F50 ; vertical
G1 X100 Y100 Z0 E4 F50 ; hyp
```

Too big, try smaller fin

```
G1 X100 Y100 Z0 F2000
G1 X120 Y100 Z0 E2 F500 ;
G1 Z10 E2.00 F50 ; vertic
G1 X100 Y100 Z0 E0.5 F50 ; hypotenuse
```

This is doing well, but try t

```
G1 X100 Y100 Z0.2 F20
G1 X120 Y100 Z0.2 E2.5 F500 ; base
G1 Z10 E2.00 F50 ; vert
G1 E0.2 F500 ; preemptiv
G1 X100 Y100 Z0.2 E2. F
```

Above worked okay at 205. Need to think about how to get hypotenuse less wiggly.

Functions?

Visualization?

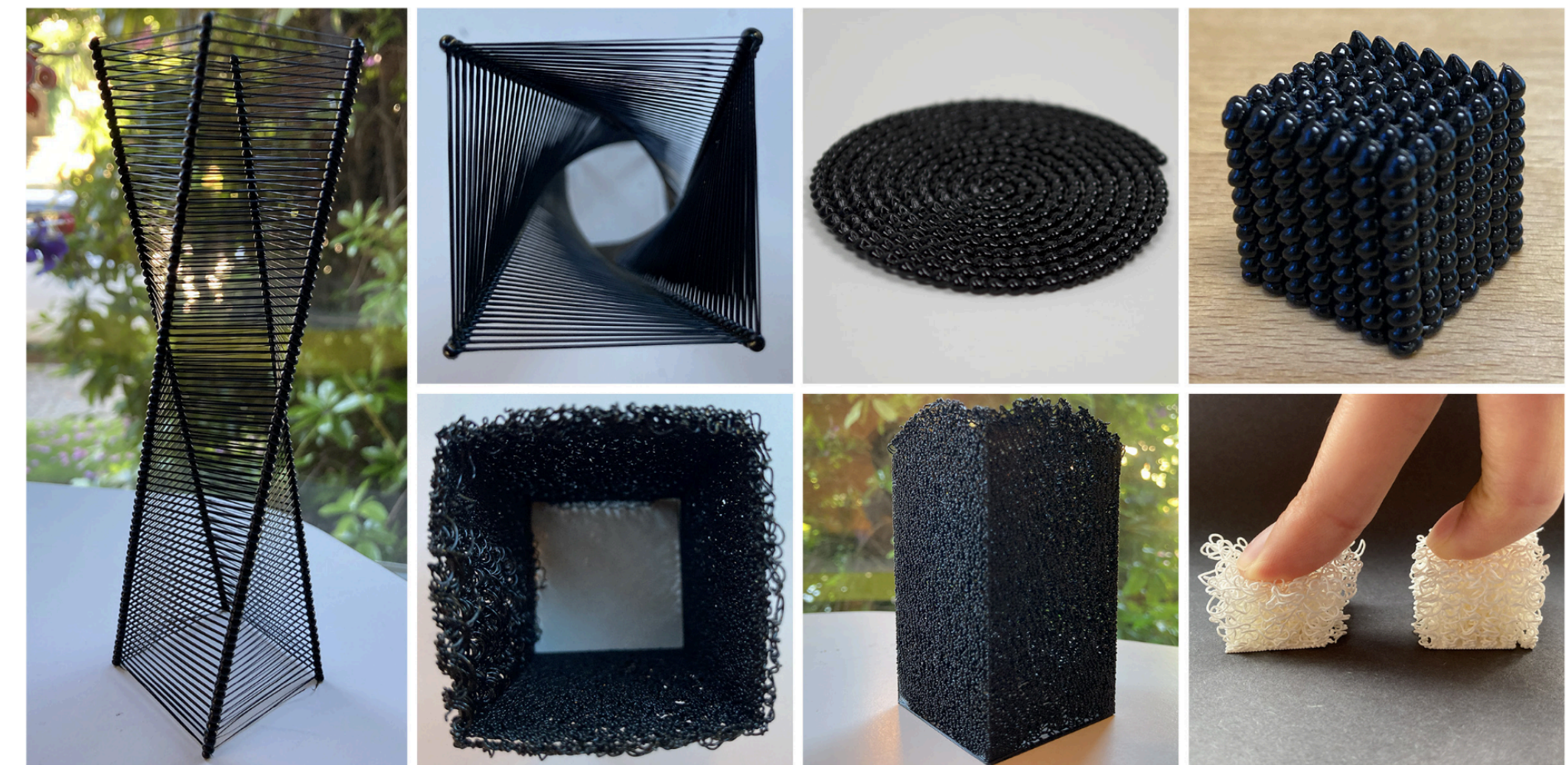
Testing?

Previewing in-situ?

*Interoperating with
outside libraries?*

*Adapting to different
machines?*

```
fab.moveExtrude(
  fab.maxX / 2 + x,
  fab.maxY / 2 + y,
  z
);
```



Subbaraman and Peek. p5.fab. 2022.

Verso is a tool for *fabrication-as-programming*.

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

Finish

3 let tabletop = await \$tabletopCalibrator(machine);

Tabletop Calibrator<axidraw>

open

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

4 let geometry = await \$geometryGallery(tabletop);

Geometry Gallery

close

streamlines.svg



5 geometry = geometry.translate(mm(105), mm(75));

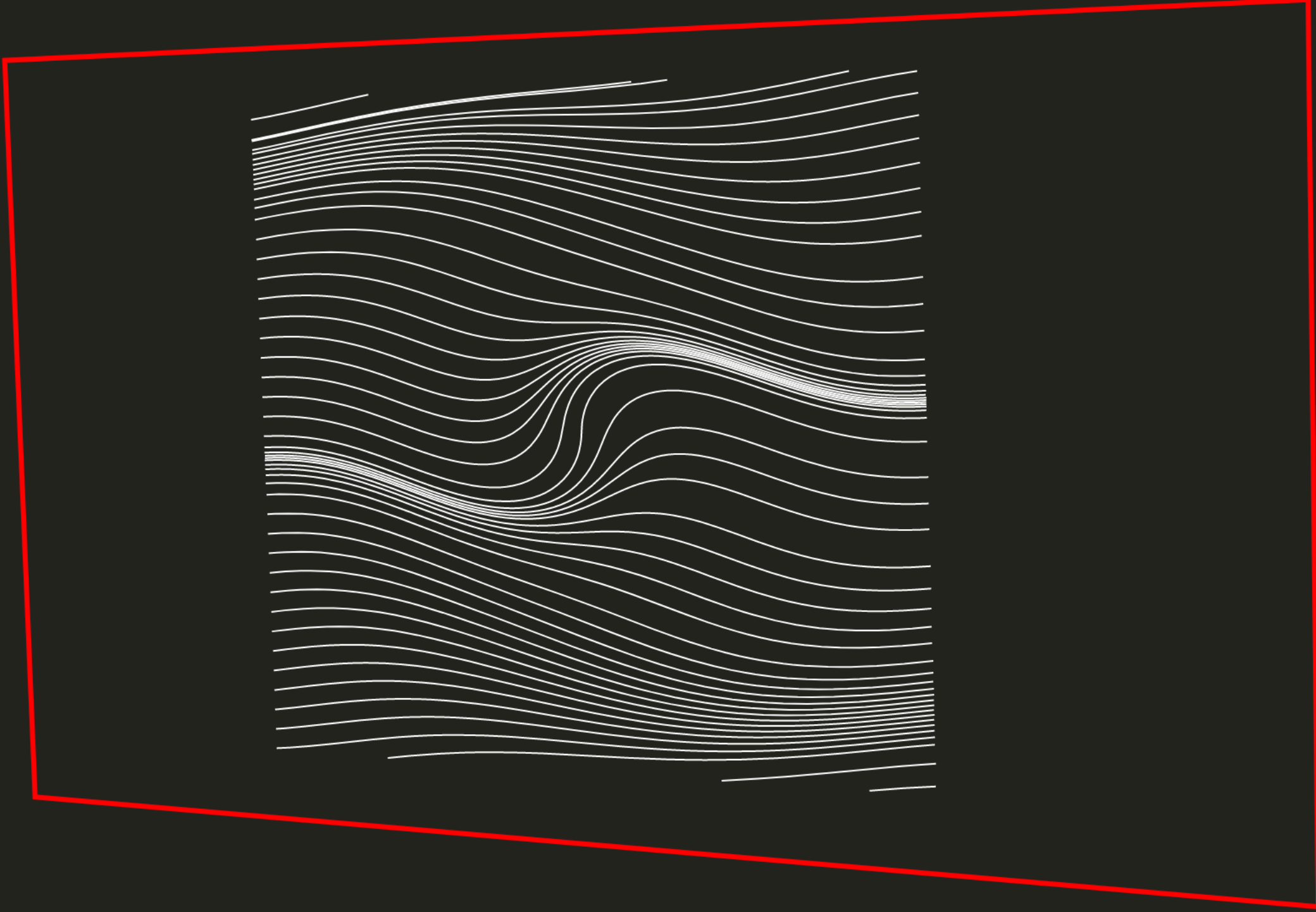
6 let toolpath = await \$axidrawDriver(machine, geometry);

Axidraw Driver

close

Axidraw EBB

71 SM,35,147,133
72 SM,62,255,246
73 SM,49,193,202
74 SM,30,100,135



(280, 180)

Verso is a tool for *fabrication-as-programming*.

Code Editor

The screenshot displays the Verso Code Editor interface, which is split into two main panels. The left panel is a code editor with a yellow background, showing a sequence of commands for a fabrication process. The right panel is a dark gray drawing canvas with a red border, displaying a series of white, wavy lines representing a streamlines SVG file. A red coordinate label **(280, 180)** is positioned at the bottom left of the canvas.

Code Editor Content:

```
2. Disengage the motors and push the pen as close as possible to the control board.
```

Disengage

Finish

```
3 let tabletop = await $tabletopCalibrator(machine);
```

Tabletop Calibrator<axidraw> **open**

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

```
4 let geometry = await $geometryGallery(tabletop);
```

Geometry Gallery **close**

streamlines.svg

The Geometry Gallery shows a grid of 12 small thumbnail images. The first row contains six thumbnails: a solid yellow square, a yellow square with the text "Bricscat", a yellow square with the text "VOC", a yellow square with the text "DR", a yellow square with a grid pattern, and a yellow square with a grid pattern. The second row contains six thumbnails: a yellow square with a wavy line, a yellow square with the text "NA", a yellow square with the text "PTC", a yellow square with a wavy line, a yellow square with a wavy line, and a yellow square with a wavy line. The third row contains three thumbnails: a yellow square with a wavy line, a yellow square with a wavy line, and a yellow square with a wavy line.

```
5 geometry = geometry.translate(mm(105), mm(75));
```

```
6 let toolpath = await $axidrawDriver(machine, geometry);
```

Axidraw Driver **close**

Axidraw EBB

```
71 SM,35,147,133
```

```
72 SM,62,255,246
```

```
73 SM,49,193,202
```

```
74 SM,30,100,135
```


Verso is a tool for *fabrication-as-programming*.

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

Finish

3 let tabletop = await \$tabletopCalibrator(machine);

Tabletop Calibrator<axidraw>

open

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

4 let geometry = await \$geometryGallery(tabletop);

Geometry Gallery

close

streamlines.svg



5 geometry = geometry.translate(mm(105), mm(75));

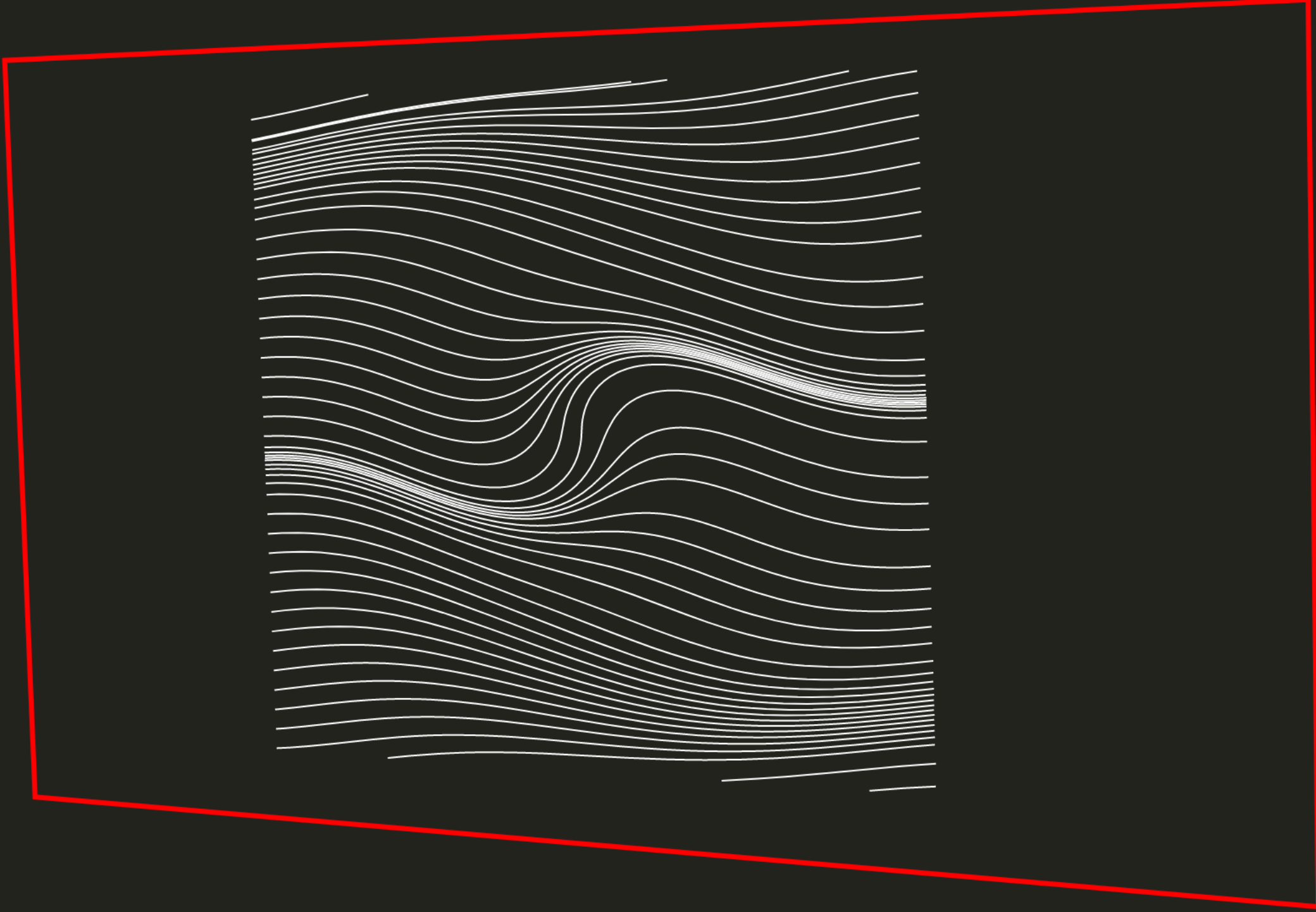
6 let toolpath = await \$axidrawDriver(machine, geometry);

Axidraw Driver

close

Axidraw EBB

71 SM,35,147,133
72 SM,62,255,246
73 SM,49,193,202
74 SM,30,100,135



(280, 180)

Verso is a tool for *fabrication-as-programming*.

Modules

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

Finish

```
3 let tabletop = await $tabletopCalibrator(machine);
```

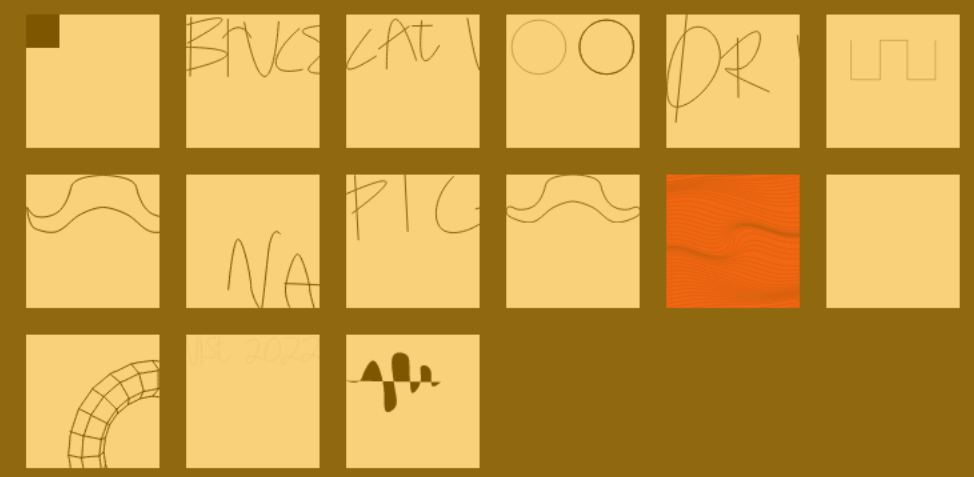
Tabletop Calibrator<axidraw> open

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

```
4 let geometry = await $geometryGallery(tabletop);
```

Geometry Gallery close

streamlines.svg

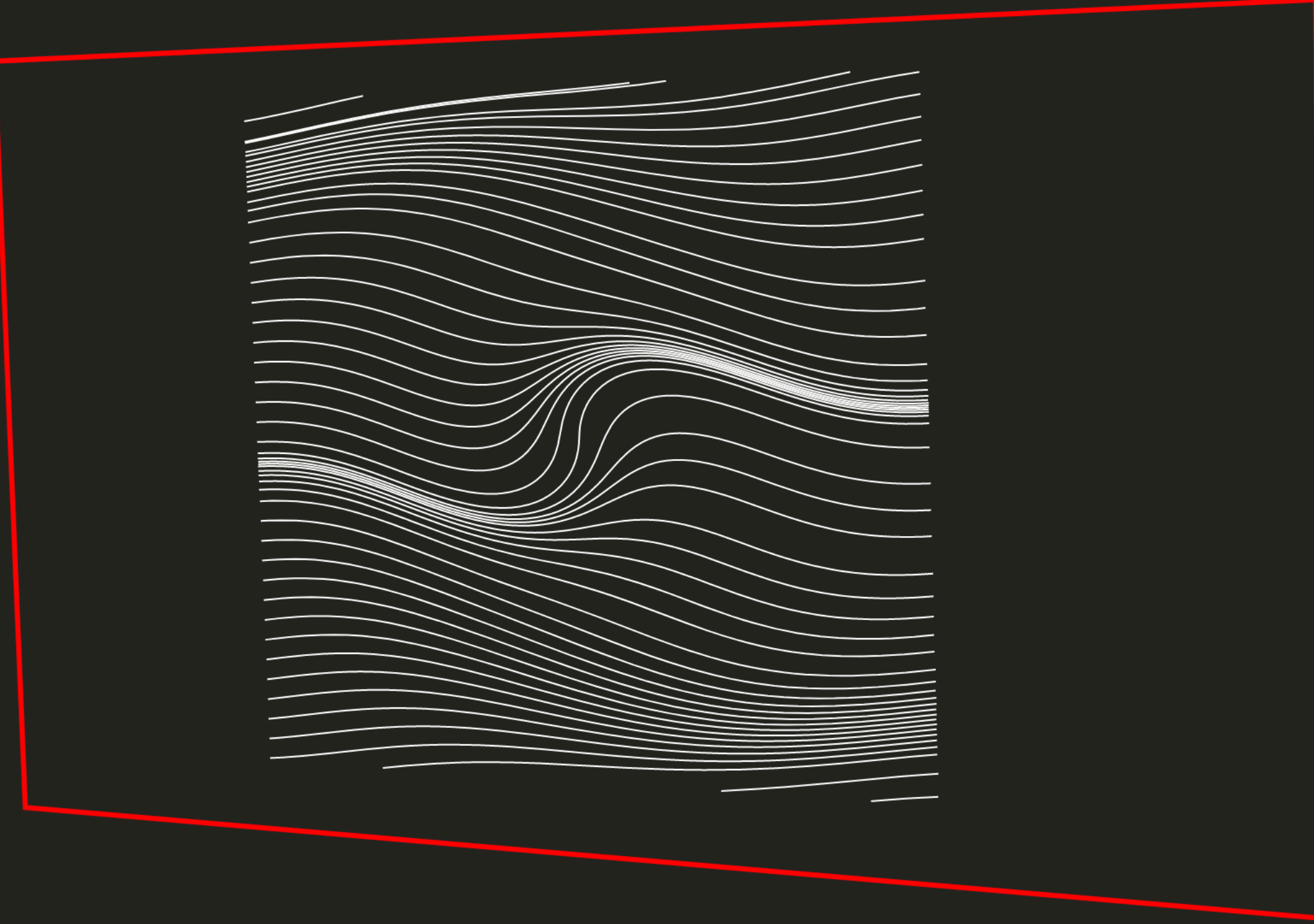


```
5 geometry = geometry.translate(mm(105), mm(75));  
6 let toolpath = await $axidrawDriver(machine, geometry);
```

Axidraw Driver close

Axidraw EBB

```
71 SM,35,147,133  
72 SM,62,255,246  
73 SM,49,193,202  
74 SM,30,100,135
```



(280, 180)

Verso is a tool for *fabrication-as-programming*.

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

Finish

3 let tabletop = await \$tabletopCalibrator(machine);

Tabletop Calibrator<axidraw>

open

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

4 let geometry = await \$geometryGallery(tabletop);

Geometry Gallery

close

streamlines.svg



5 geometry = geometry.translate(mm(105), mm(75));

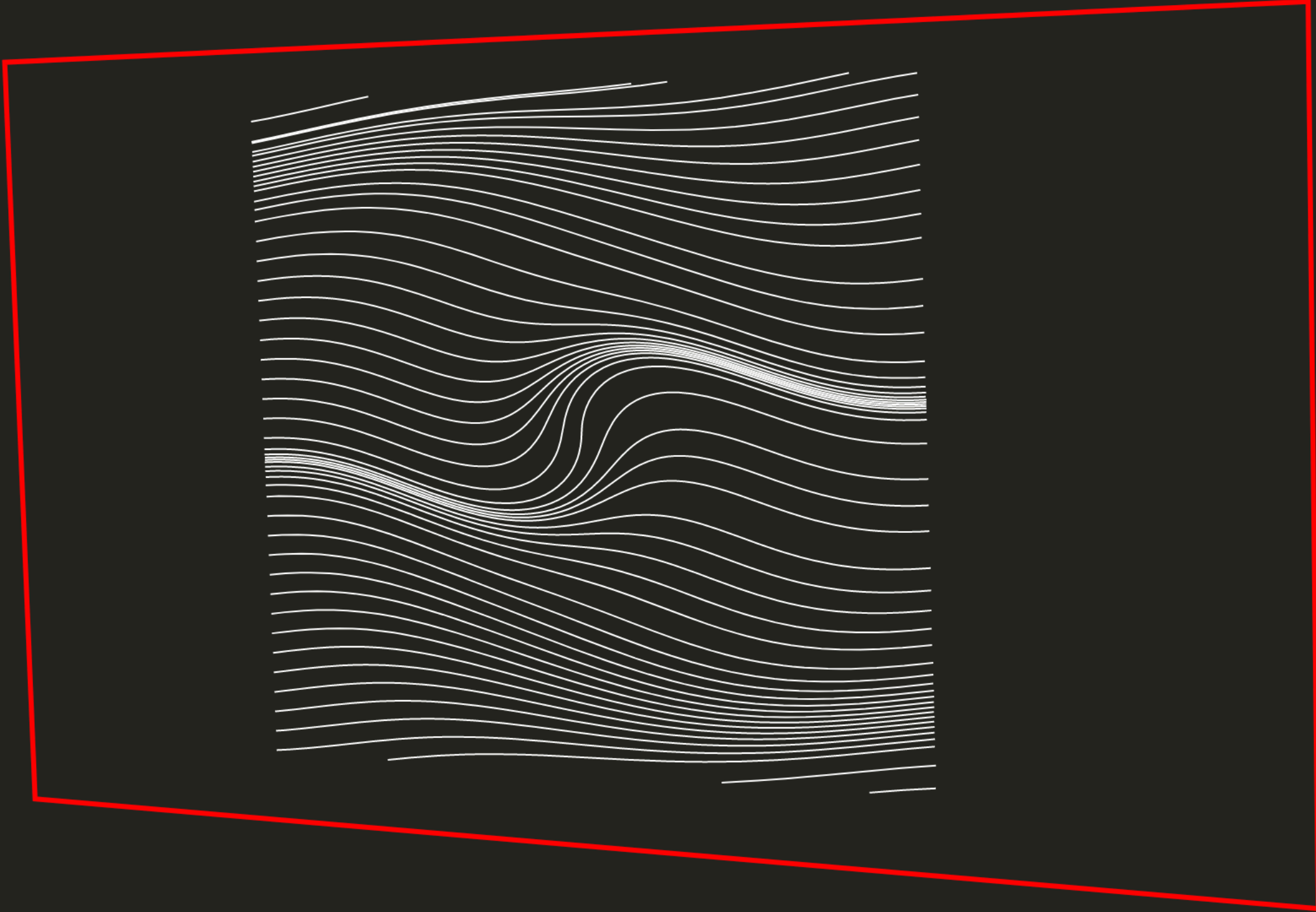
6 let toolpath = await \$axidrawDriver(machine, geometry);

Axidraw Driver

close

Axidraw EBB

71 SM,35,147,133
72 SM,62,255,246
73 SM,49,193,202
74 SM,30,100,135



(280, 180)

Verso is a tool for *fabrication-as-programming*.

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

Finish

3 let tabletop = await \$tabletopCalibrator(machine);

Tabletop Calibrator<axidraw>

open

Tabletop(WorkEnvelope(homography:
[0.5793794235,0.0283735682,52.3439633712,-
0.0410112791,0.6531995084,68.265749768,-
0.0001792844,0.0000253614,1]))

4 let geometry = await \$geometryGallery(tabletop);

Geometry Gallery

close

streamlines.svg



5 geometry = geometry.translate(mm(105), mm(75));

6 let toolpath = await \$axidrawDriver(machine, geometry);

Axidraw Driver

close

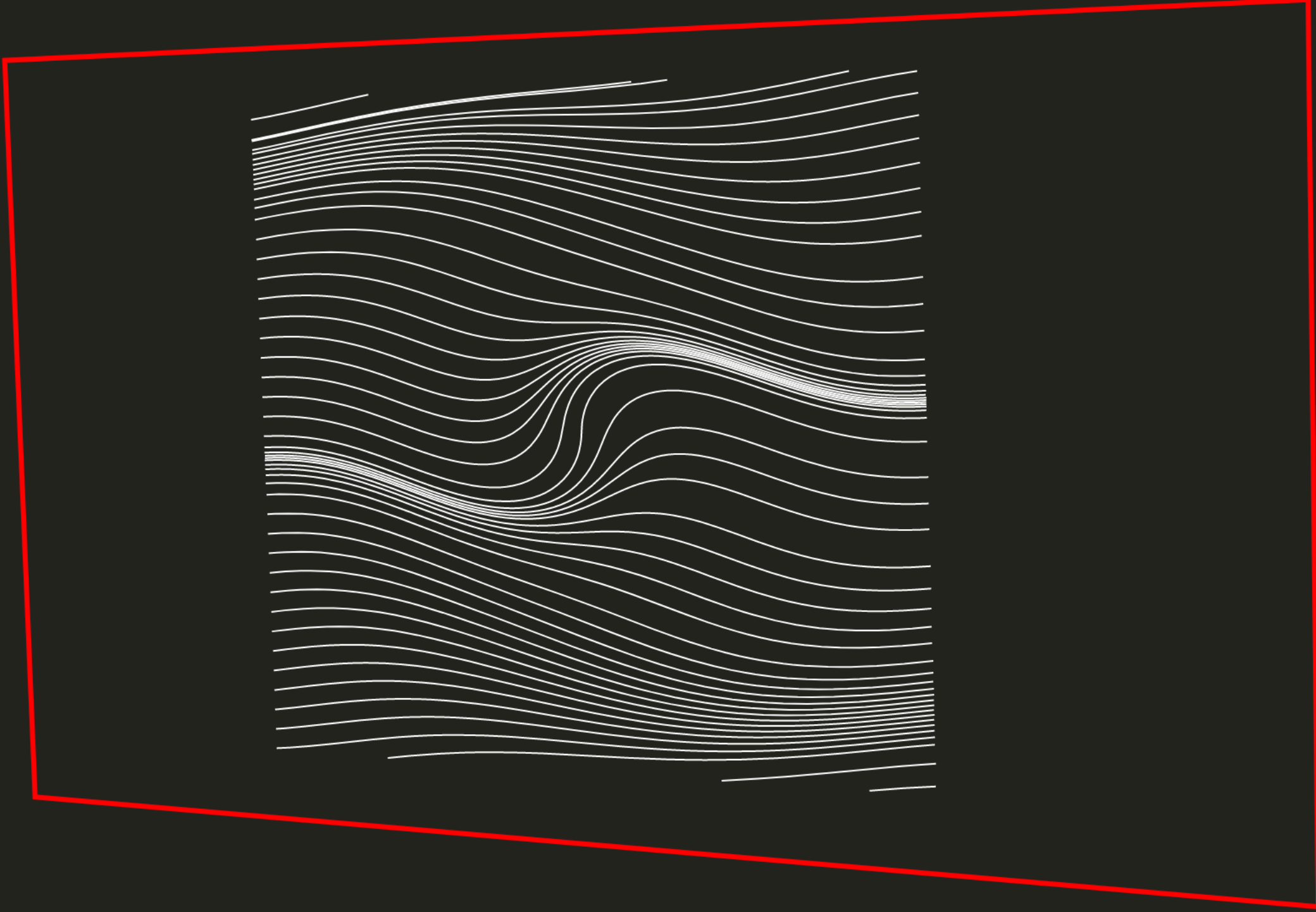
Axidraw EBB

71 SM,35,147,133

72 SM,62,255,246

73 SM,49,193,202

74 SM,30,100,135



(280, 180)

Visualization to be Projected

Fabrication-as-programming brings to digital fabrication:

Fabrication-as-programming brings to digital fabrication:

```
let q1_max = 36. in
```

```
let grades =  
  $dataframe
```

= q1_max +. 24. +. \$fslider 0. 40.

	"A1"	"A2"	"A3"	"Midterm"	"Final"
"Andrew"	80.	92.	83.5	95.	88.
"Cyrus"	61.	64.	98.	70.	85.
"David"	75.	81.	73.	82.	79.

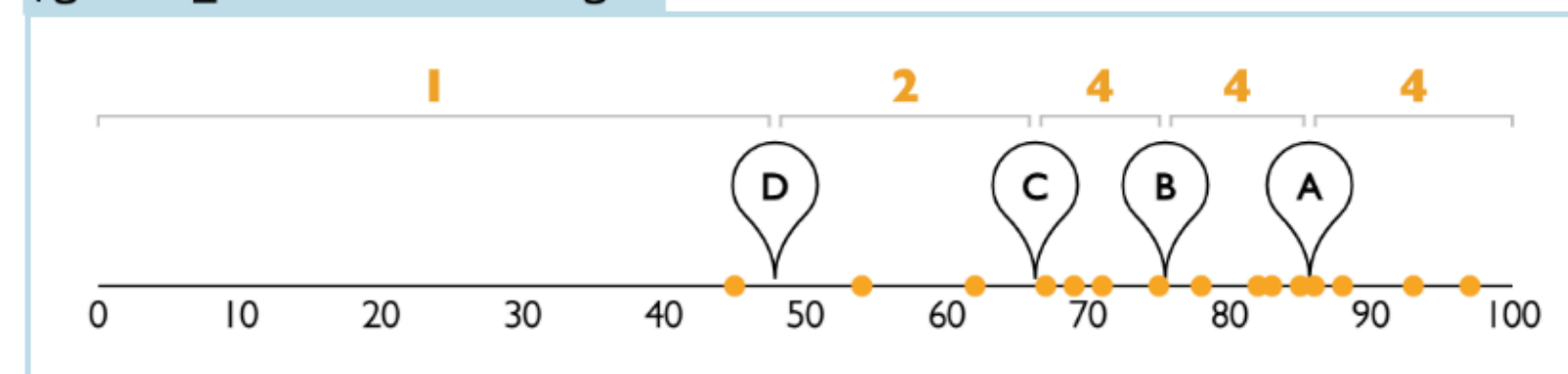
+

```
in
```

```
let averages = compute_weighted_averages grades weights in
```

```
let cutoffs =
```

```
  $grade_cutoffs averages
```



```
in
```

```
format_for_university (assign_grades averages cutoffs)
```

Filling Typed Holes with Live GUIs. Omar et al. 2019.

Multimodal Programming

Fabrication-as-programming brings to digital fabrication:

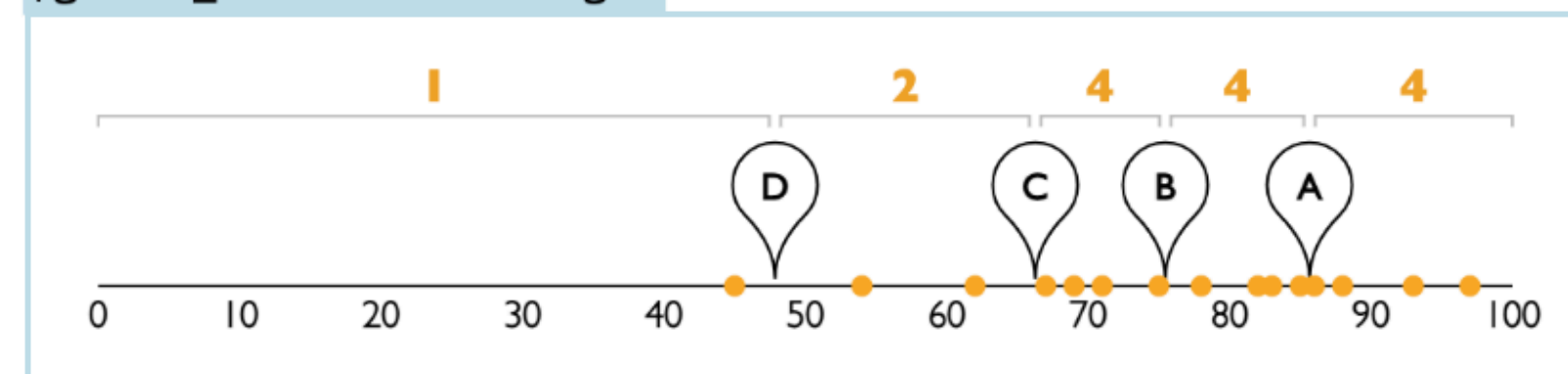
```
let q1_max = 36. in
let grades =
  $dataframe
```

`= q1_max +. 24. +. $fslider 0. 40.`

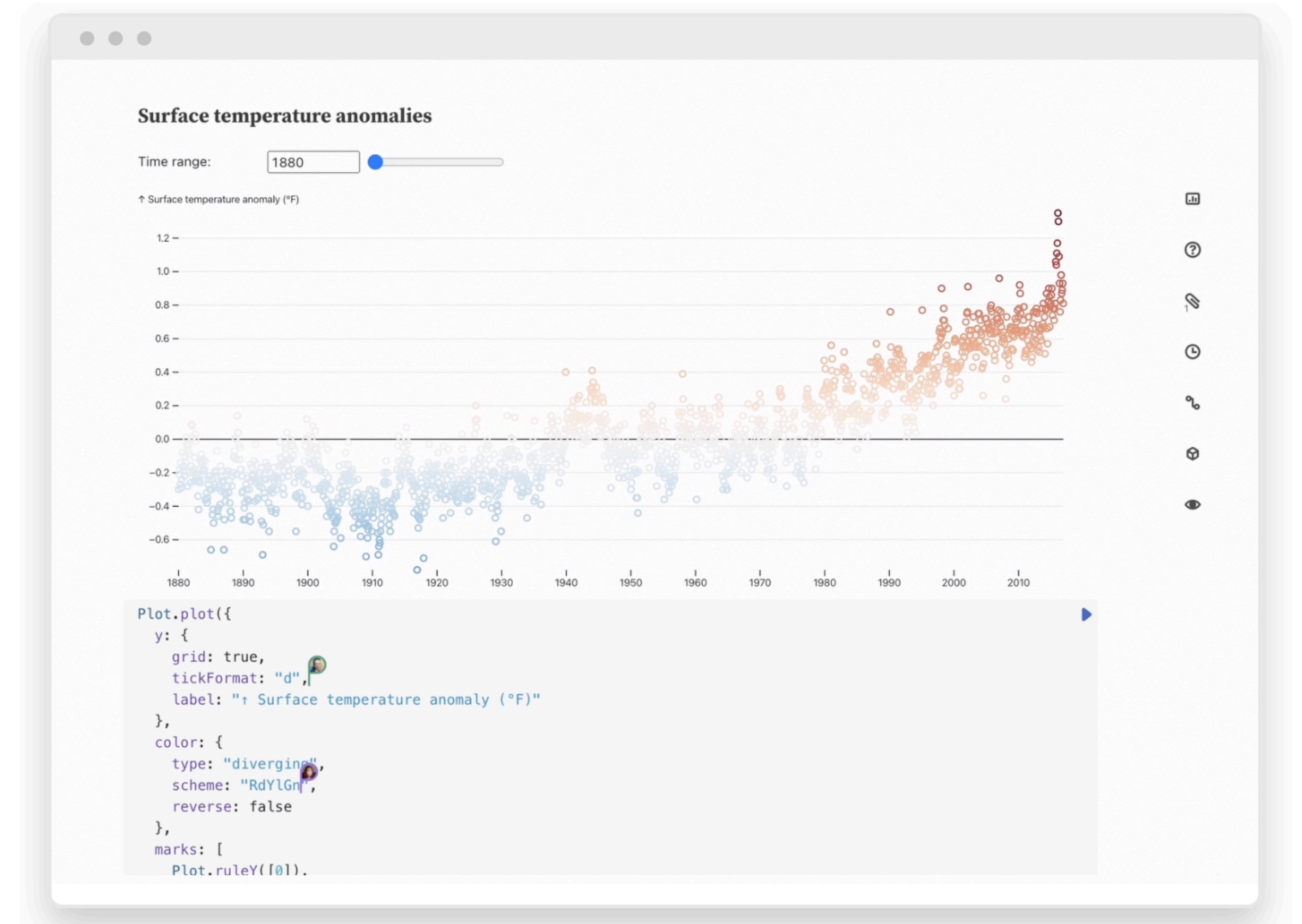
	"A1"	"A2"	"A3"	"Midterm"	"Final"
"Andrew"	80.	92.	83.5	95.	88.
"Cyrus"	61.	64.	98.	70.	85.
"David"	75.	81.	73.	82.	79.

+

```
in
let averages = compute_weighted_averages grades weights in
let cutoffs =
  $grade_cutoffs averages
```



```
in
format_for_university (assign_grades averages cutoffs)
```



Filling Typed Holes with Live GUIs. Omar et al. 2019.

D3.js and Observable Notebook.

Multimodal Programming

Data visualization

Multimodal Programming

Modules: inline GUIs that accept and return values.

Multimodal Programming

Modules: inline GUIs that accept and return values.

```
$machineInitializer(myAxidraw)
```


Multimodal Programming

Modules: inline GUIs that accept and return values.

`$machineInitializer(myAxidraw)`

MachineInitializer<axidraw>

close

Machine(initialized: true)

Machine Initialized? Yes

1. Connect to the machine.

/dev/tty.F900S-COM3

/dev/tty.Bluetooth-Incoming-Port

/dev/tty.BIGJAMBOXbyJawbone-SPPD

Connect

2. Disengage the motors and push the pen as close as possible to the control board.

Disengage

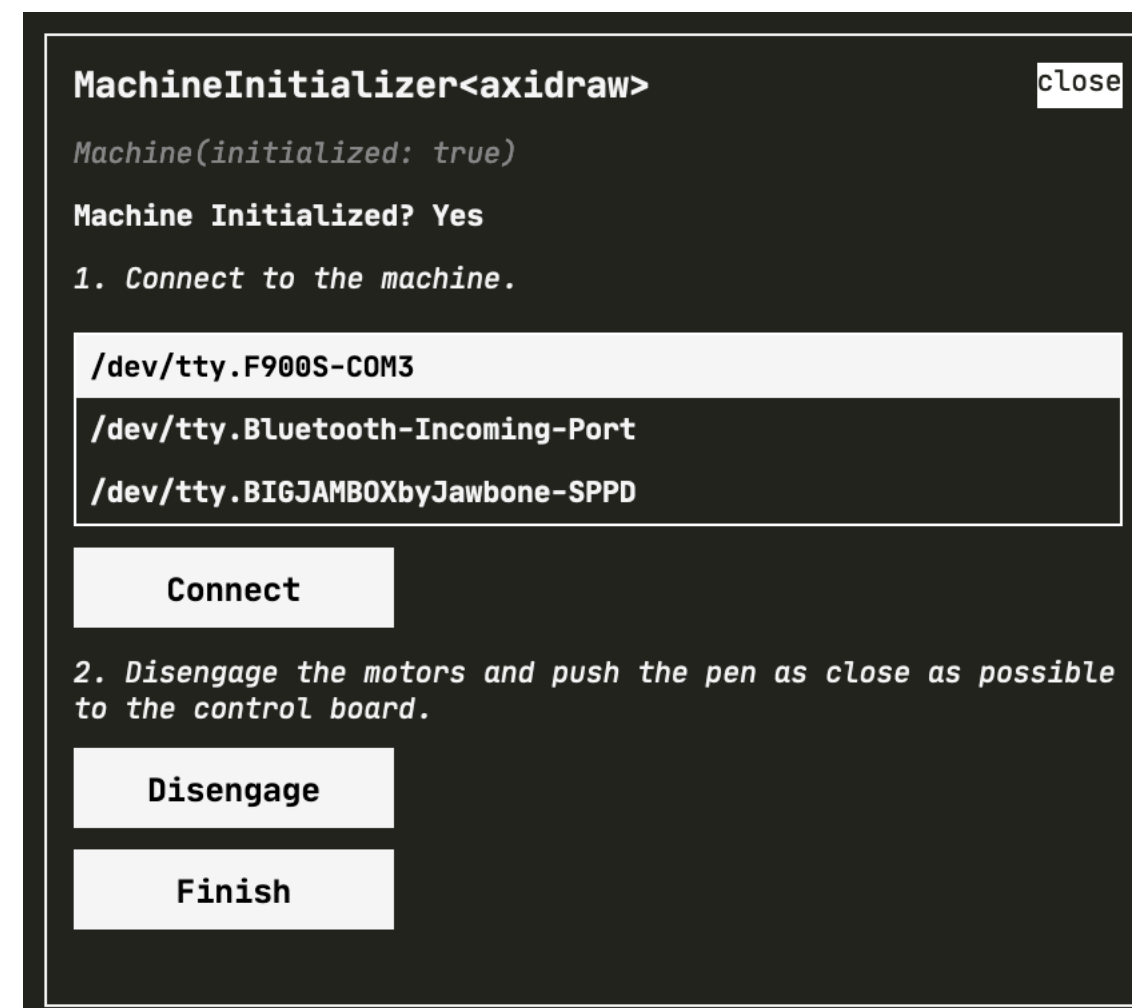
Finish

Multimodal Programming

Modules: inline GUIs that accept and return values.

```
Machine {  
  type: "axidraw",  
  initialized: false  
}
```

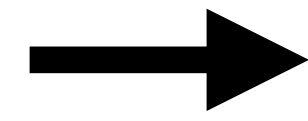
→ \$machineInitializer(myAxidraw)



Multimodal Programming

Modules: inline GUIs that accept and return values.

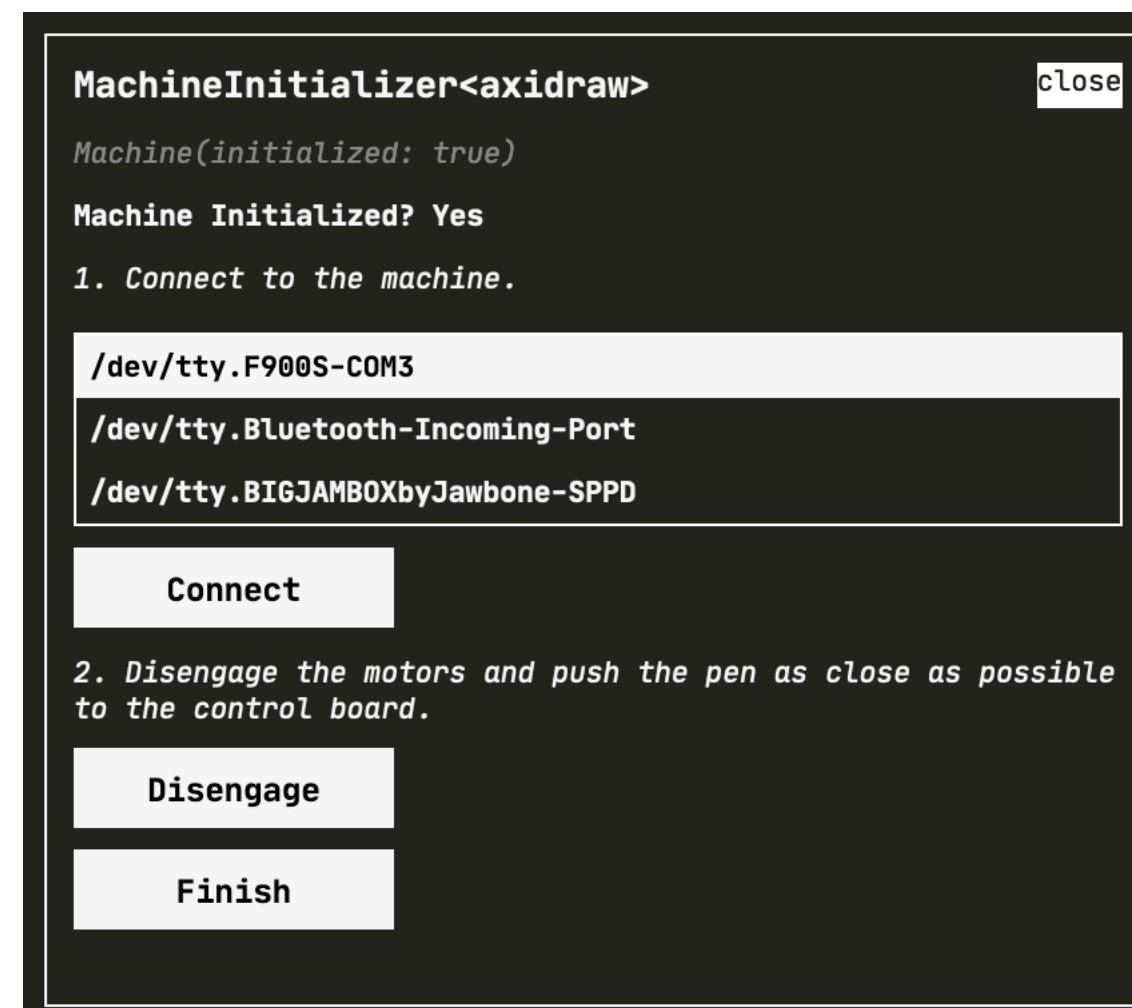
```
Machine {  
  type: "axidraw",  
  initialized: false  
}
```



`$machineInitializer(myAxidraw)`



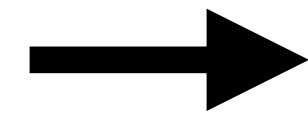
```
Machine {  
  type: "axidraw",  
  initialized: true  
}
```



Multimodal Programming

Modules: inline GUIs that accept and return values.

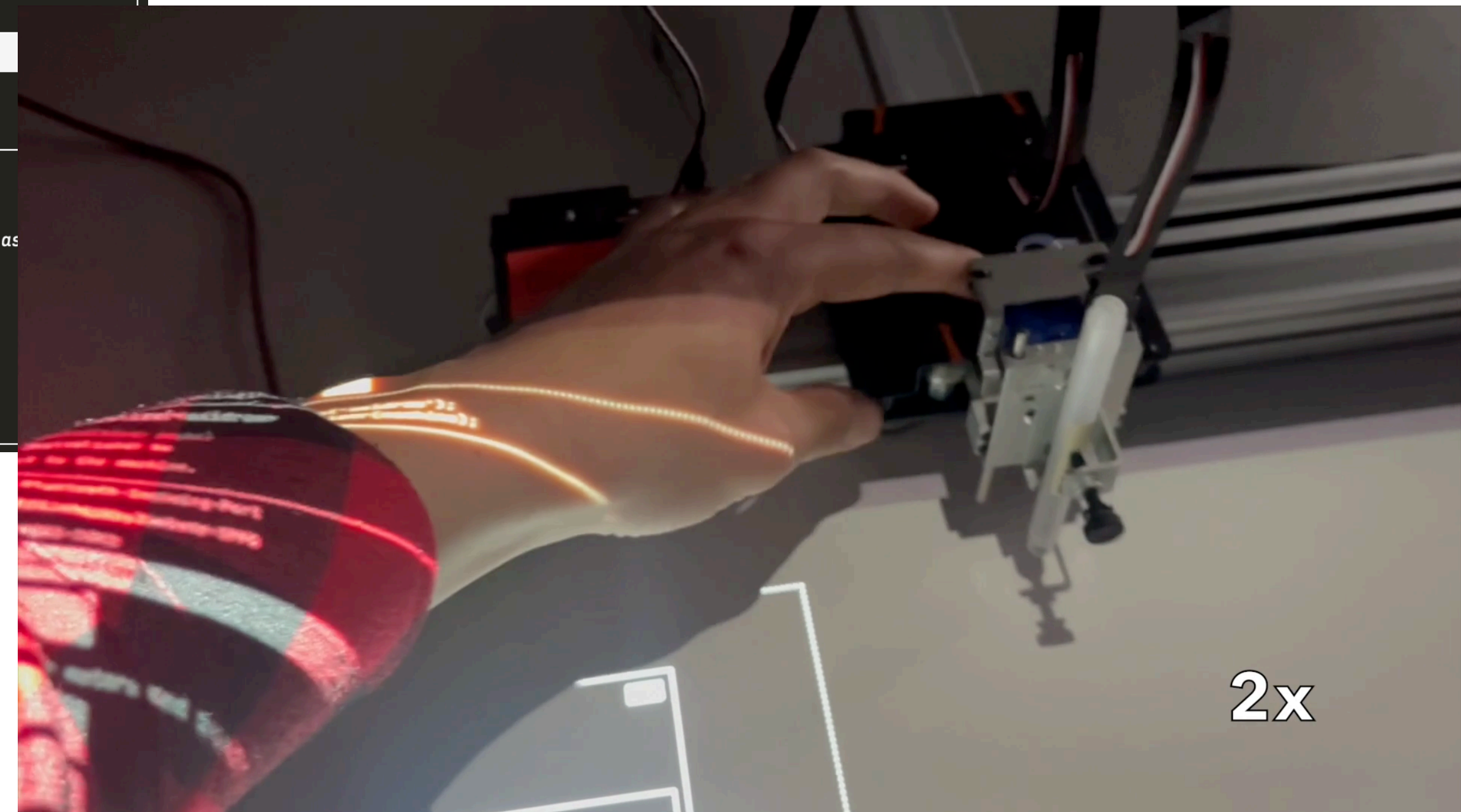
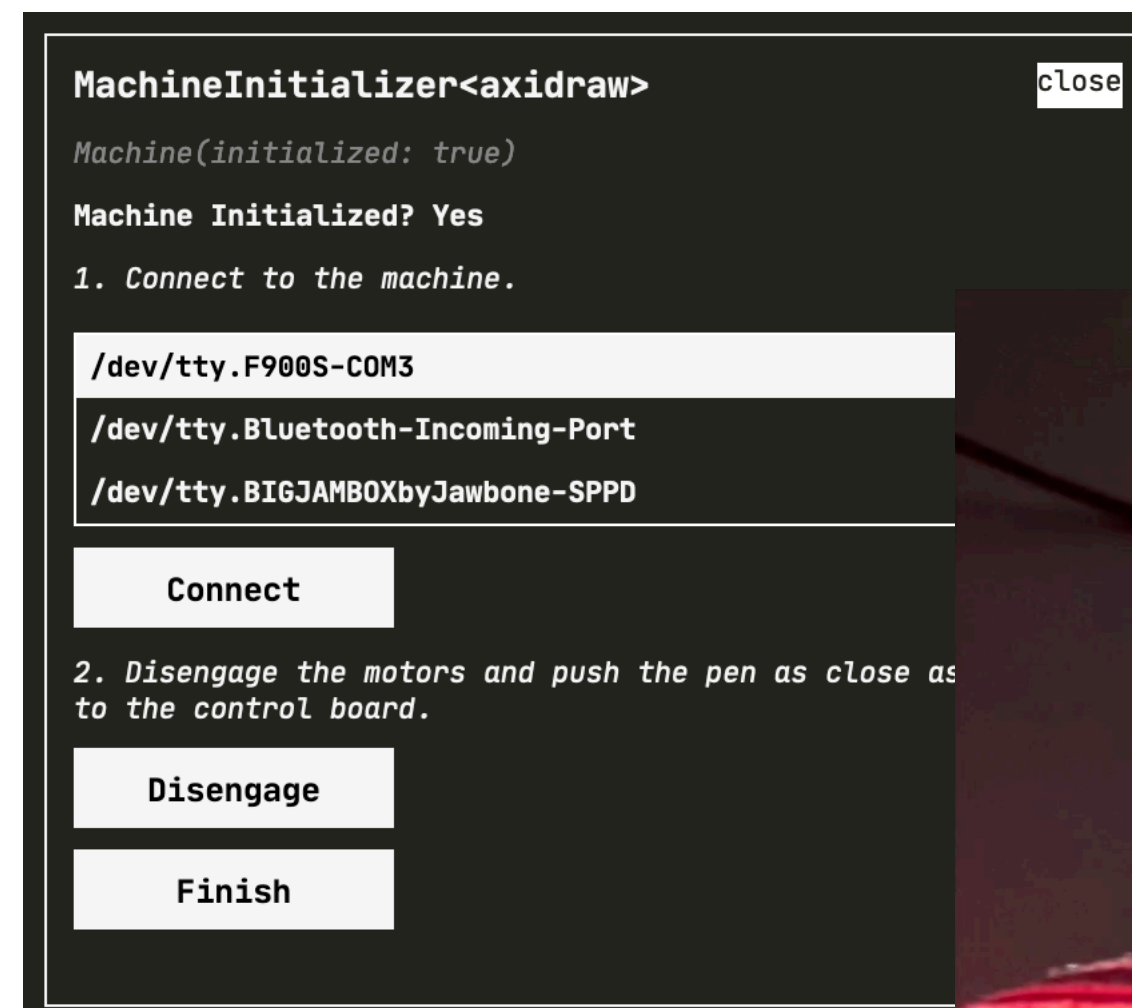
```
Machine {  
  type: "axidraw",  
  initialized: false  
}
```



```
$machineInitializer(myAxidraw)
```



```
Machine {  
  type: "axidraw",  
  initialized: true  
}
```



2x

Interoperate with existing tools

Axidraw Driver

close

Axidraw EBB

1 EM,1,1

2 SM,26,24,71

3 SM,27,49,141

4 SM,26,73,213

5 SM,26,98,283

6 SM,27,122,354

7 SM,304,1692,4908

8 SM,27,122,354

9 SM,26,98,283

10 SM,27,73,213

11 SM,26,49,141

12 SM,26,24,71

13 SP,0,200

14 SM,34,91,-90

15 SM,34,115,-113

Interoperate with existing tools

Axidraw Driver

close

Axidraw EBB

1 EM,1,1

2 SM,26,24,71

3 SM,27,49,141

4 SM,26,73,213

5 SM,26,98,283

6 SM,27,122,354

7 SM,304,1692,4908

8 SM,27,122,354

9 SM,26,98,283

10 SM,27,73,213

11 SM,26,49,141

12 SM,26,24,71

13 SP,0,200

14 SM,34,91,-90

15 SM,34,115,-113

Selectively deploy parts of a toolpath

Dispatcher

close

Machine(initialized: maybe)

Machine status: free

; Enter G-code to adjust the tool prior to dispatch.

Send Snippet

Toolpath 0

Dispatch

Pause

Interoperate with existing tools

Axidraw Driver

close

Axidraw EBB

1 EM,1,1

2 SM,26,24,71

3 SM,27,49,141

4 SM,26,73,213

5 SM,26,98,283

6 SM,27,122,354

7 SM,304,1692,4908

8 SM,27,122,354

9 SM,26,98,283

10 SM,27,73,213

11 SM,26,49,141

12 SM,26,24,71

13 SP,0,200

14 SM,34,91,-90

15 SM,34,115,-113

Selectively deploy parts of a toolpath

Dispatcher

close

Machine(initialized: maybe)

Machine status: free

; Enter G-code to adjust the tool prior to dispatch.

Send Snippet

Toolpath 0

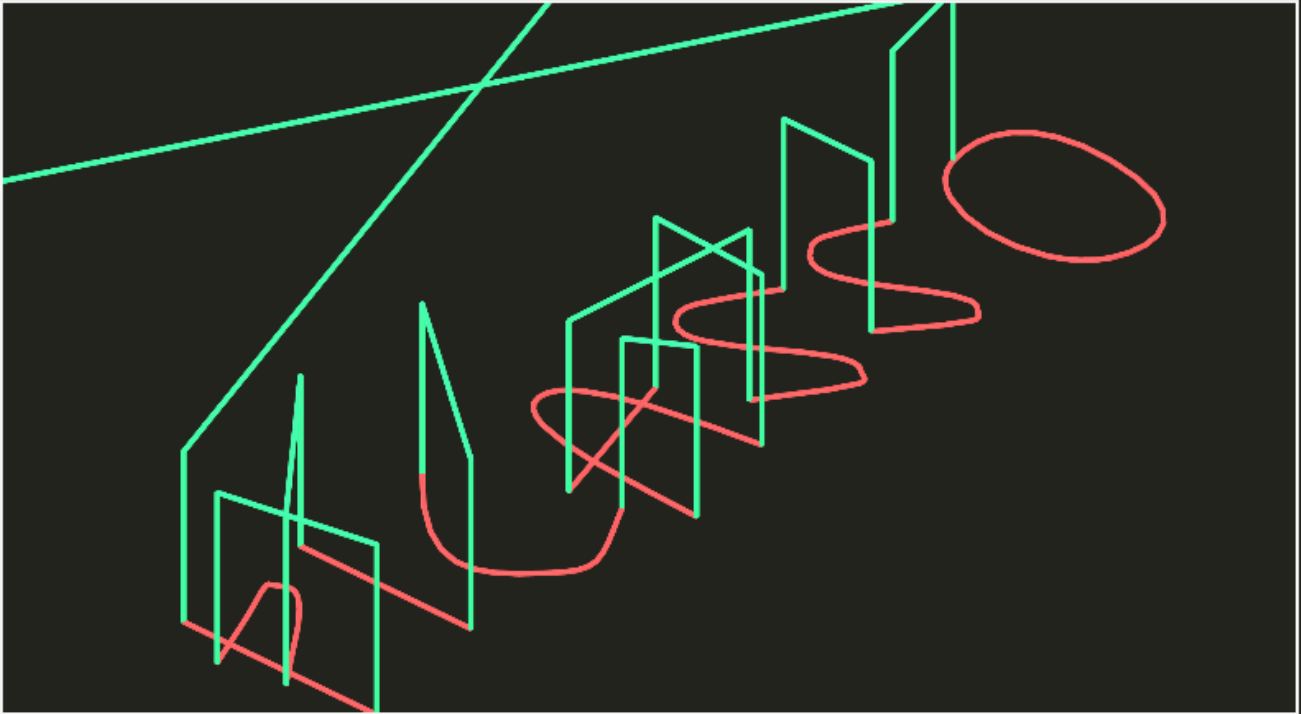
DispatchPause

Visualize machine behavior

Toolpath Visualizer

close

TSS(Colored Travel vs Draw)



Toolpath Stylesheets

Basic LinesAll movement lines.

Colored Travel vs DrawTravel and plot lines encoded by color.

Ordering (G-code)Maps toolpath order to colors of the rainbow as shown in the legend.

OrderingMaps toolpath order to colors of the rainbow as shown in the legend.

Heat MapHighlights areas with many close movements.

Sharp AngleHighlights print moves with sharp angles.

DirectionsShows directions of print movements.

Data Visualization

Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.

Data Visualization

Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.

A black rectangular box containing the text 'PLCASSO' written in a white, stylized, handwritten font. The letters are slanted and have a slightly irregular, hand-drawn appearance.

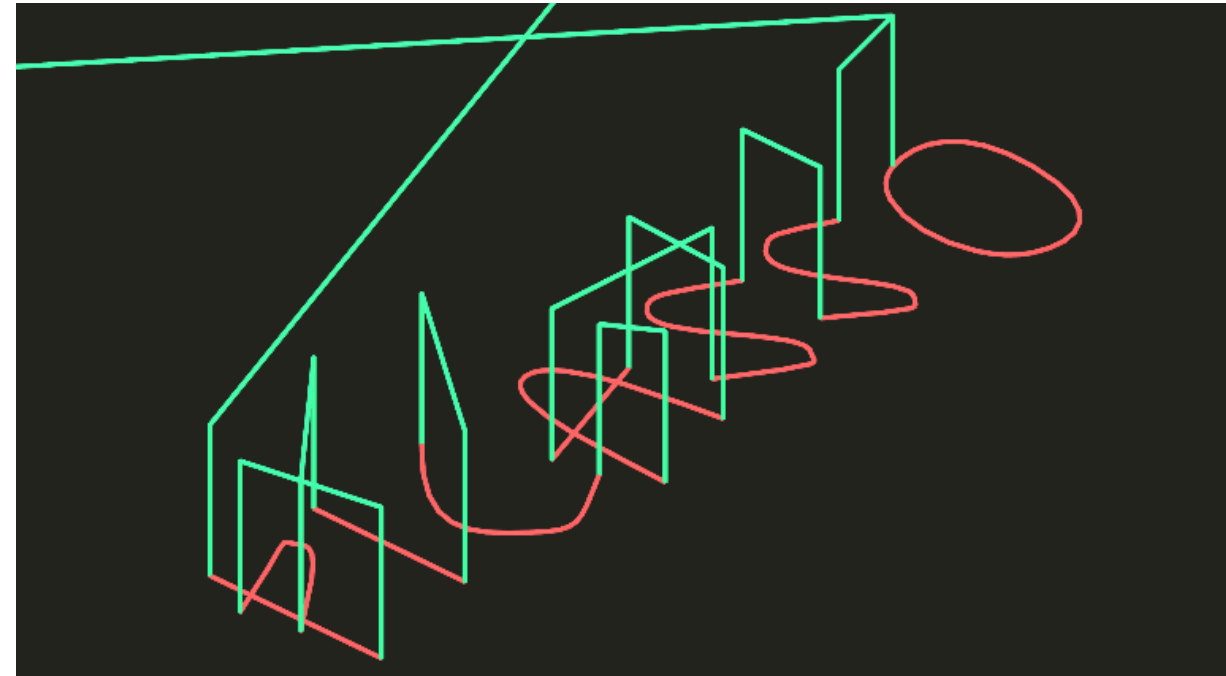
Pen markings

Data Visualization

Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



Pen markings



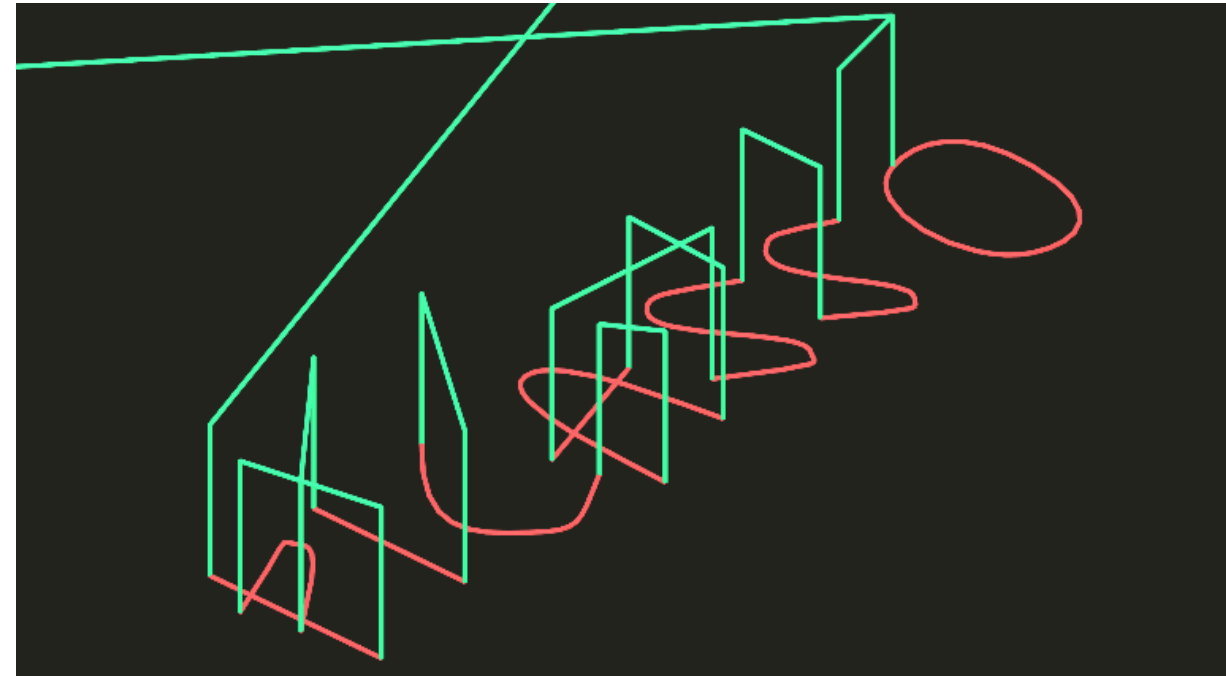
Markings vs. travel

Data Visualization

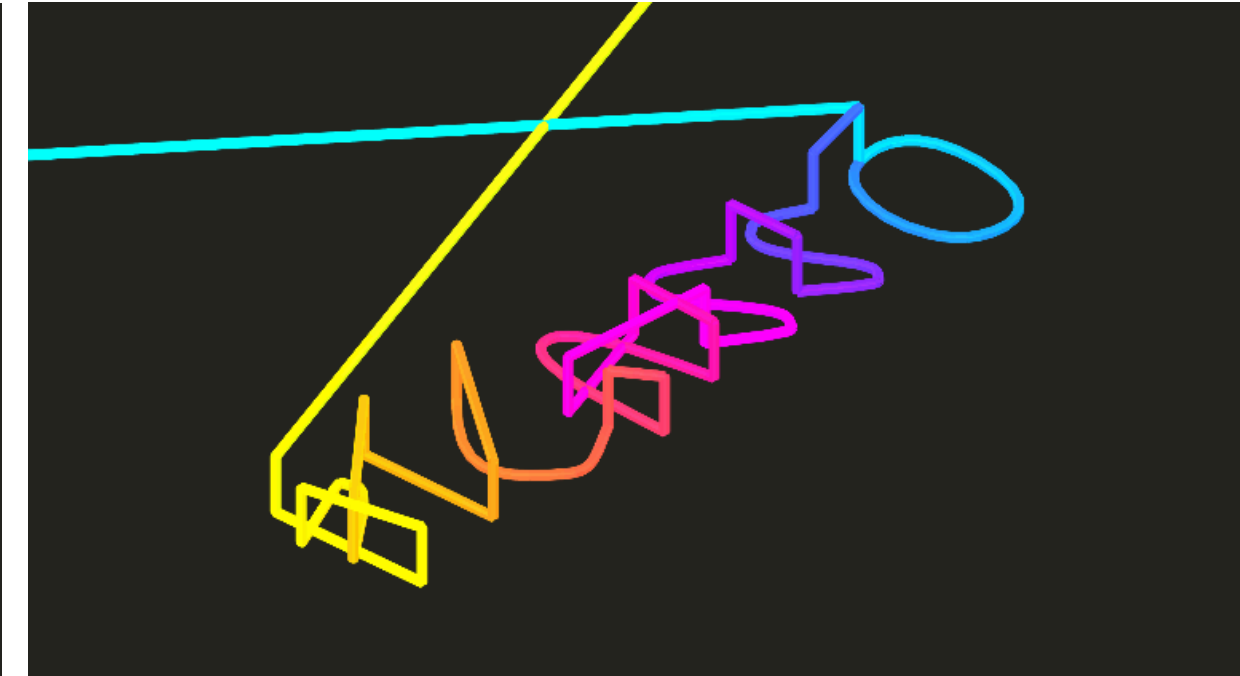
Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



Pen markings



Markings vs. travel



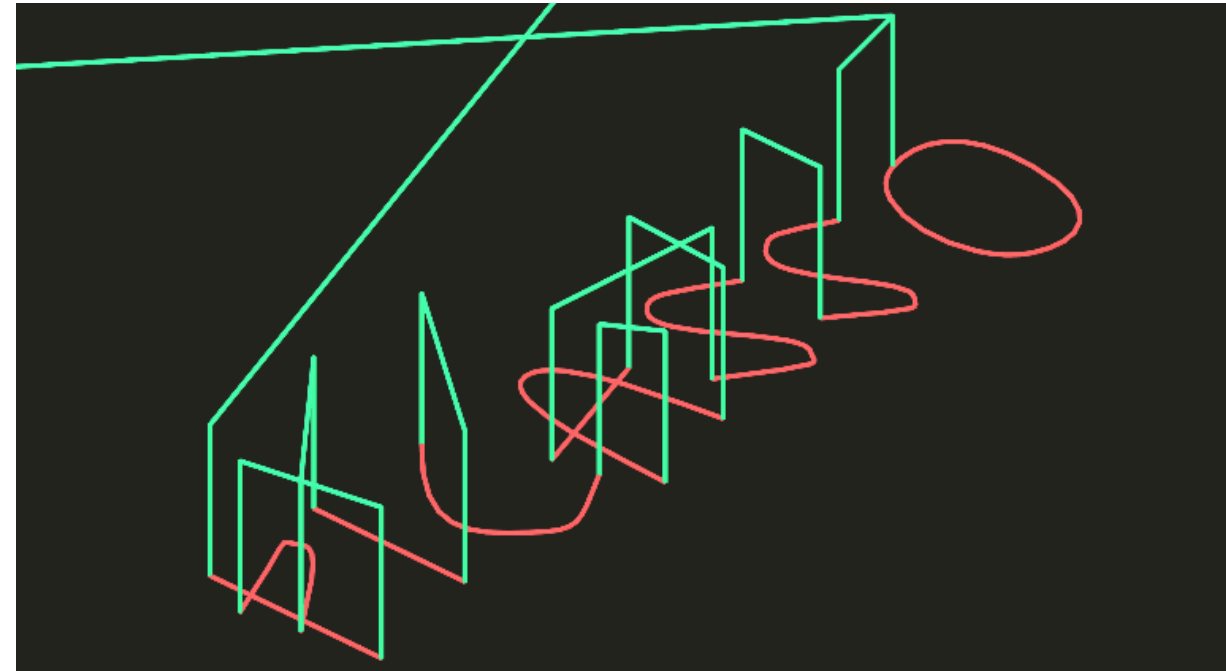
Move order

Data Visualization

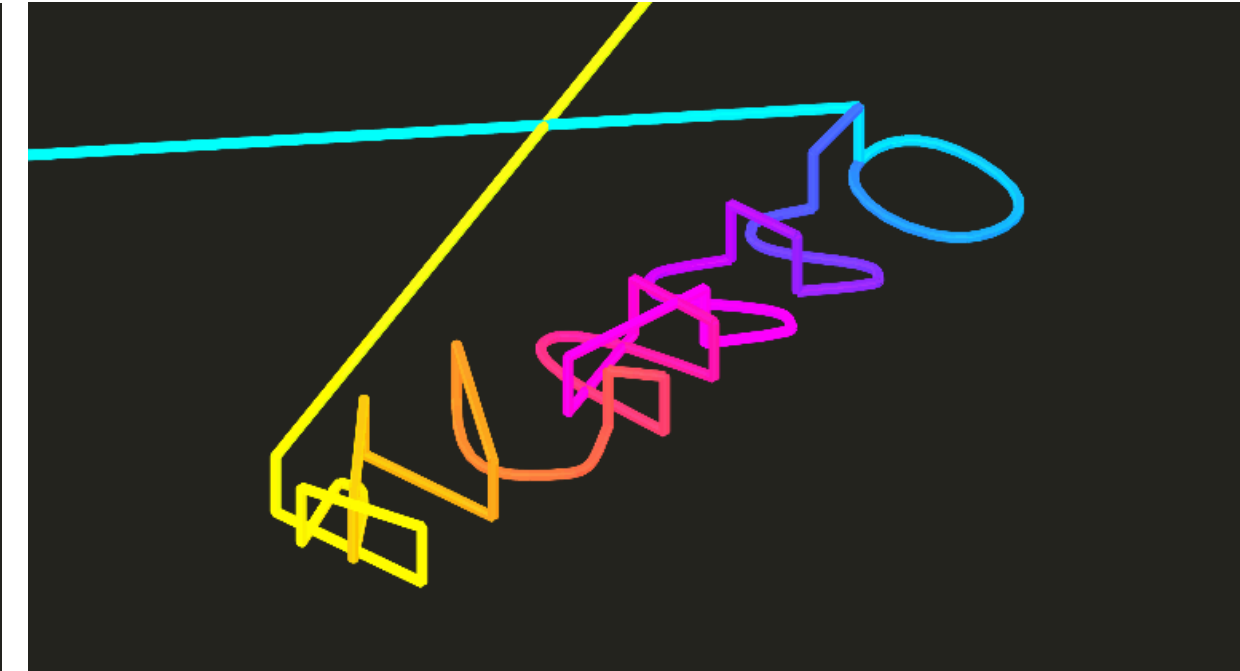
Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



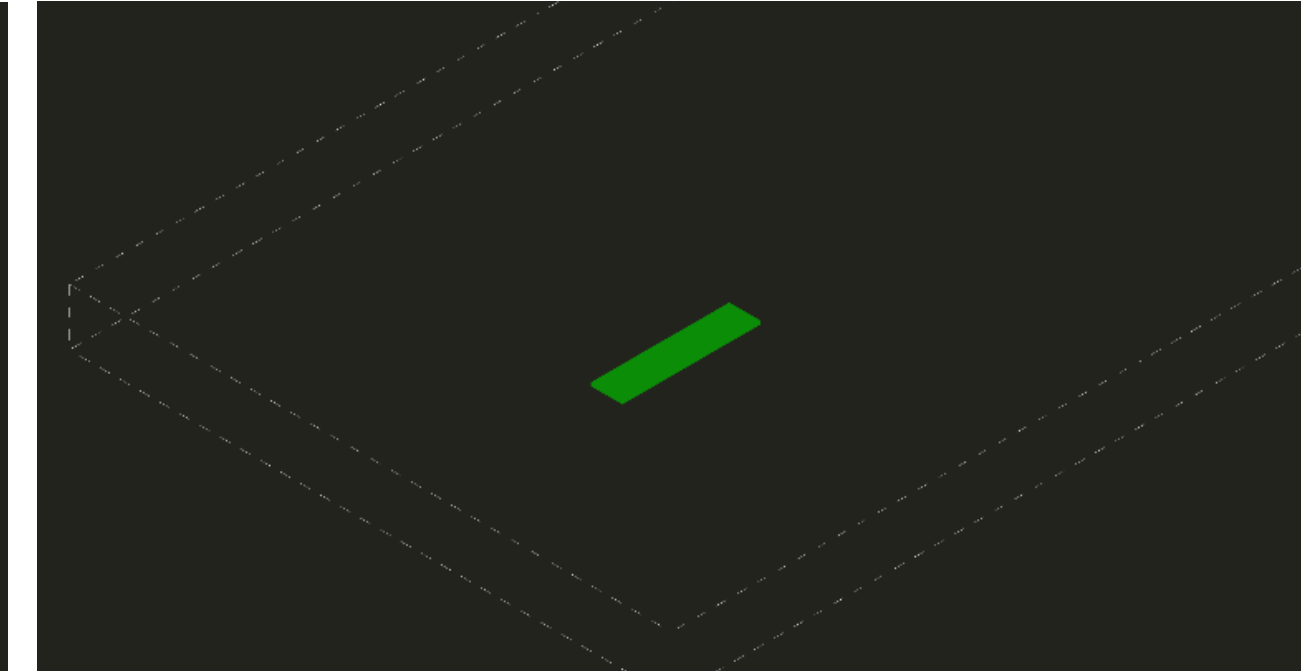
Pen markings



Markings vs. travel



Move order



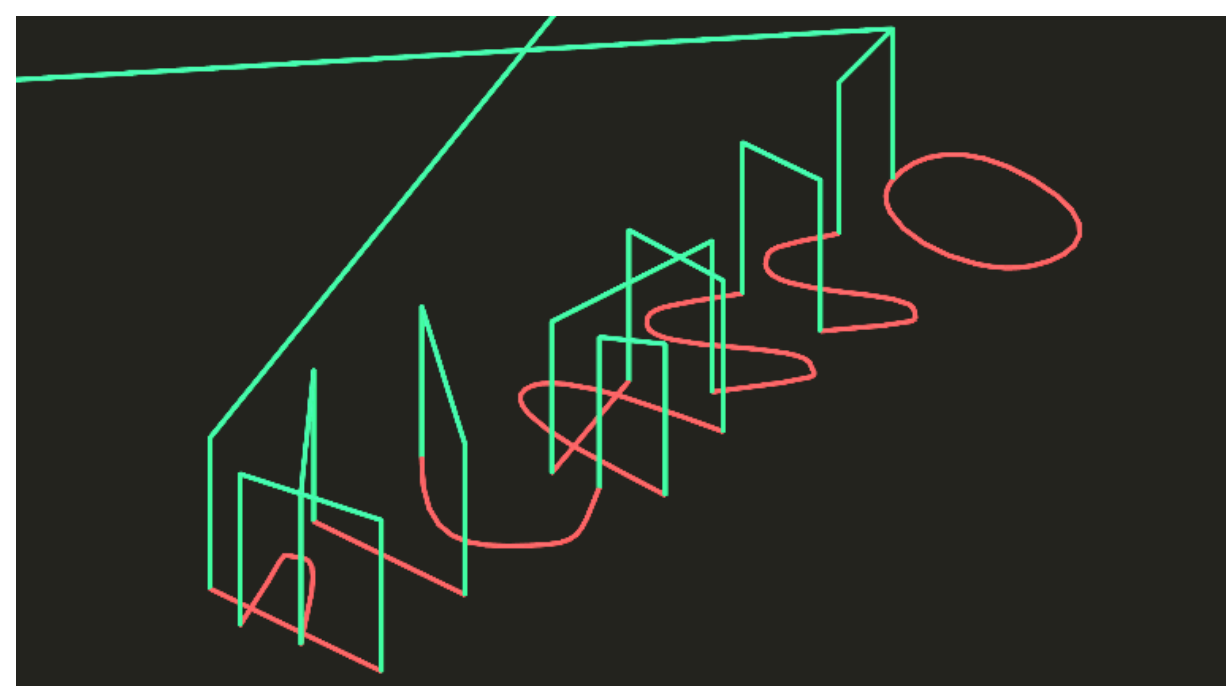
Scale w.r.t. work envelope

Data Visualization

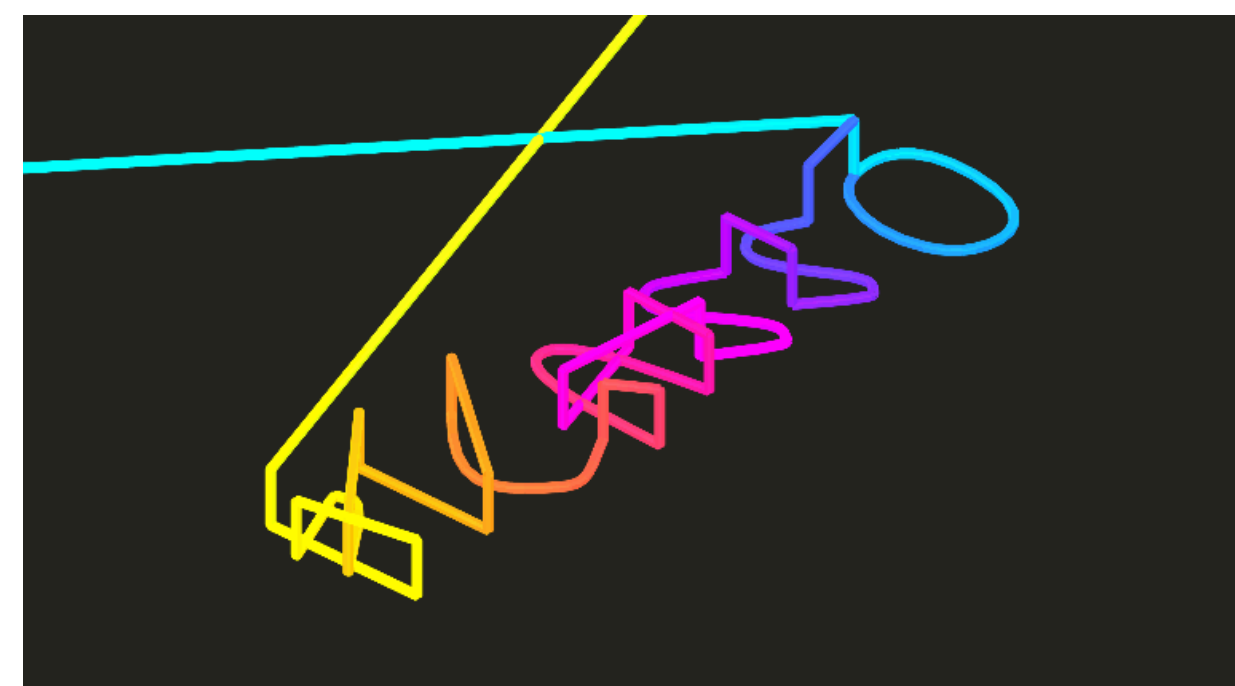
Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



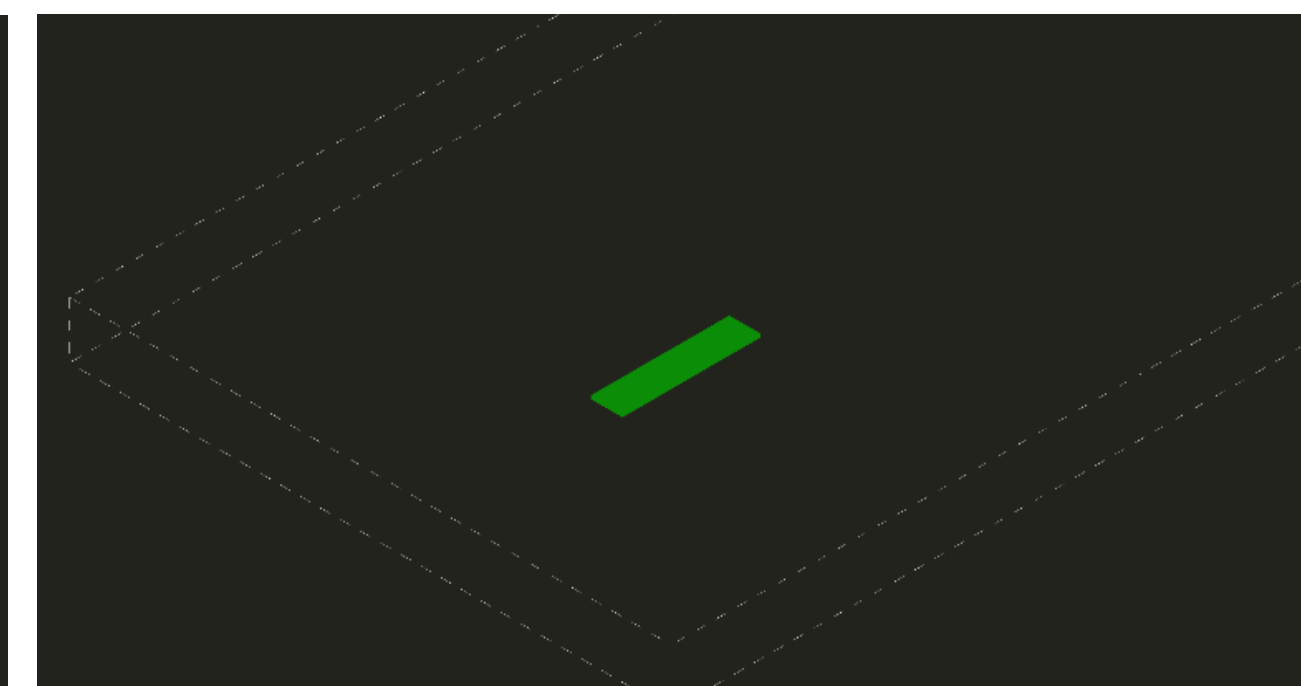
Pen markings



Markings vs. travel



Move order



Scale w.r.t. work envelope



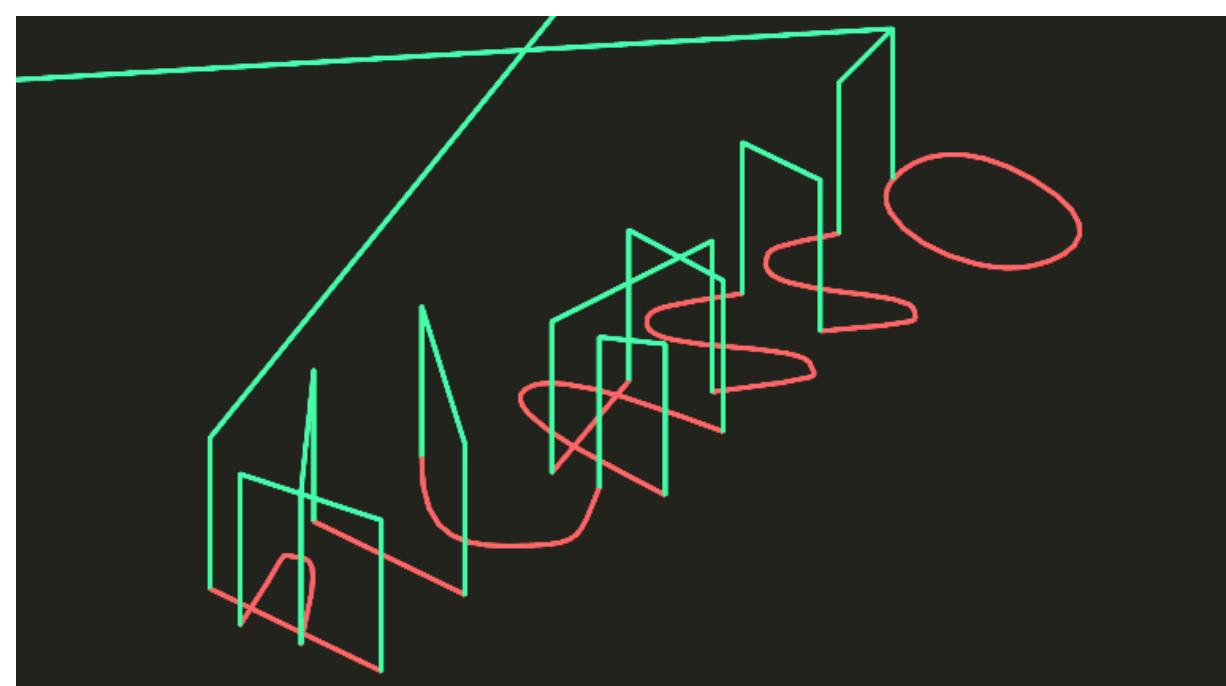
Vinyl cutter sharp angles

Data Visualization

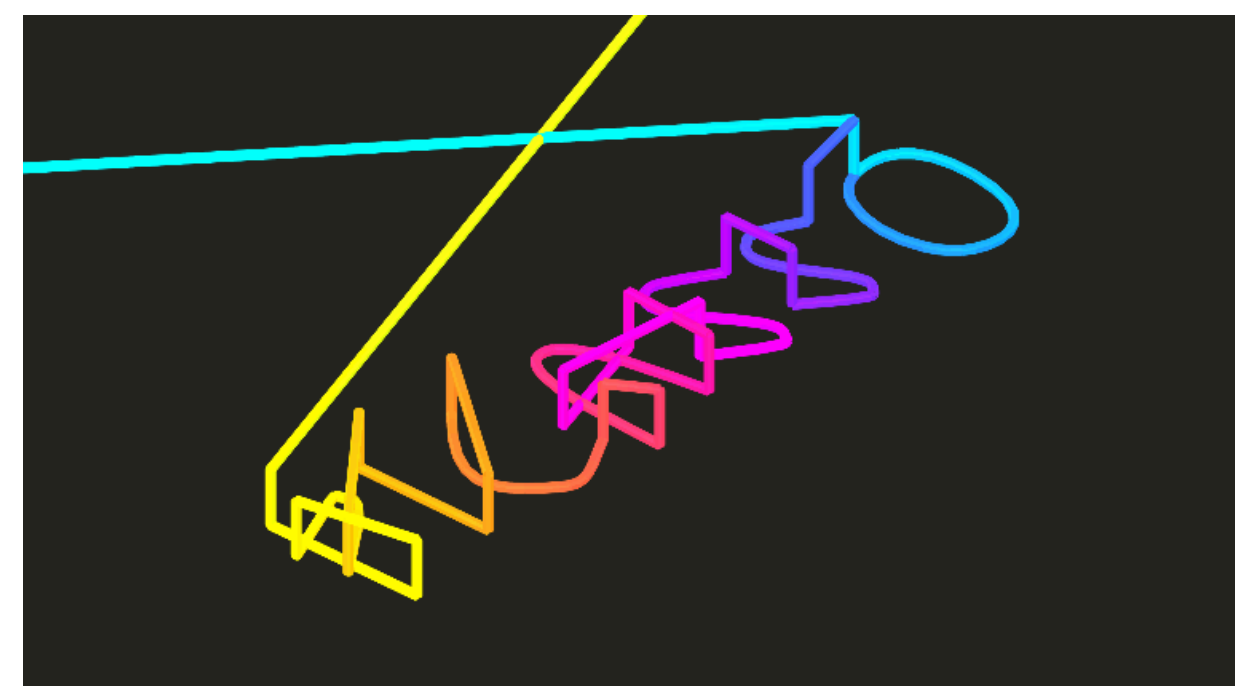
Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



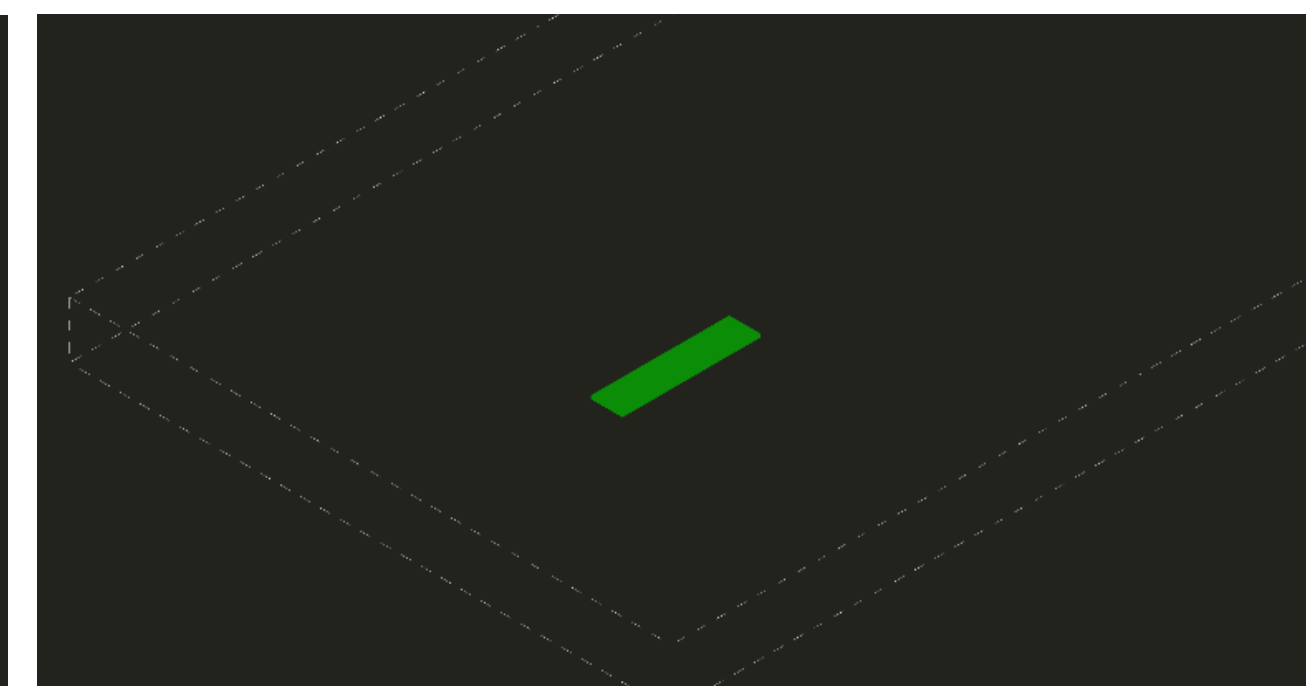
Pen markings



Markings vs. travel



Move order



Scale w.r.t. work envelope



Vinyl cutter sharp angles



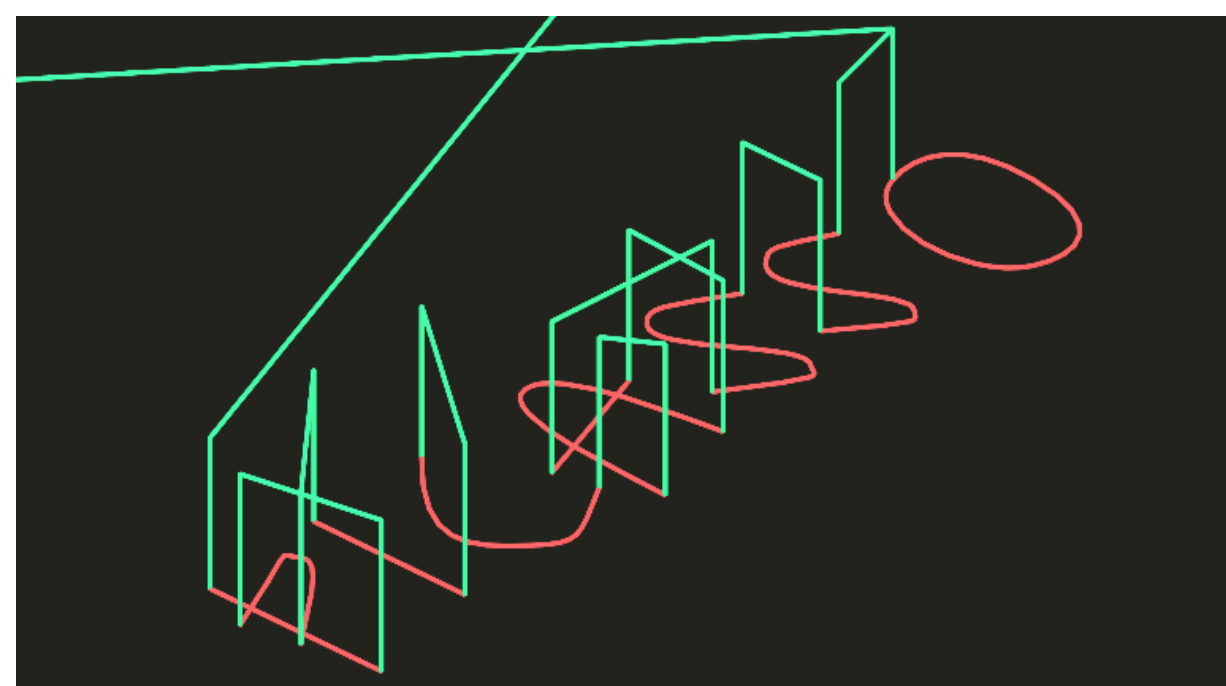
Laser cut geometries

Data Visualization

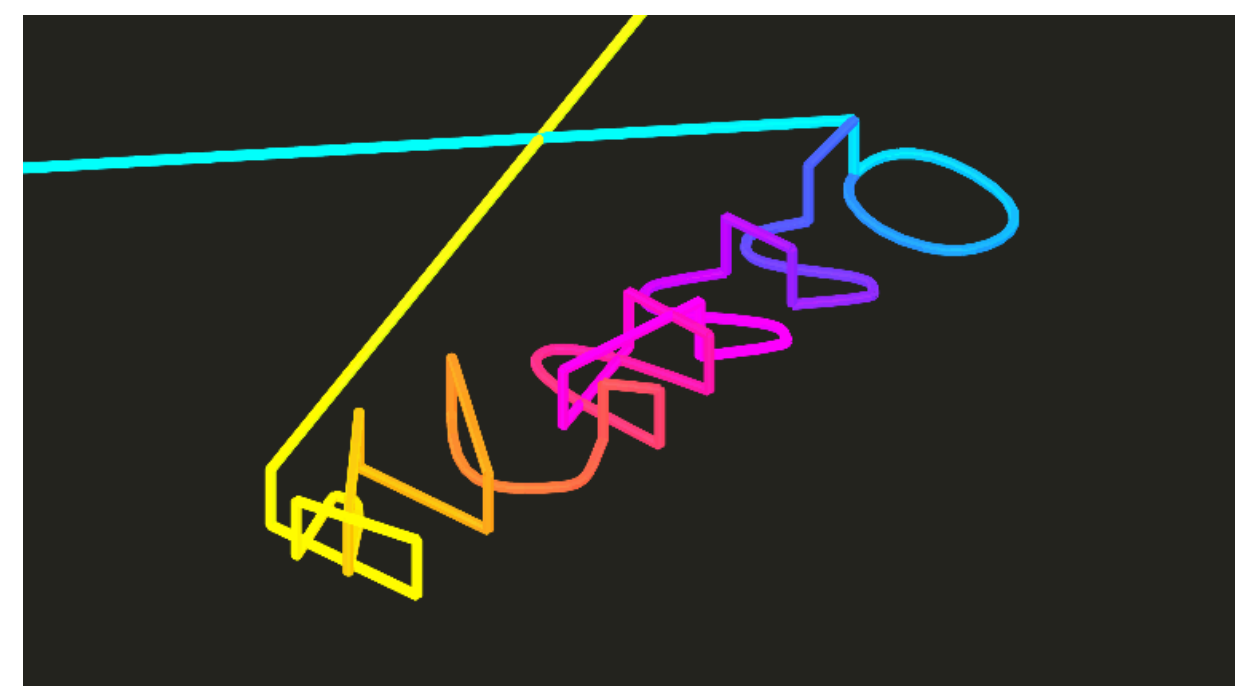
Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



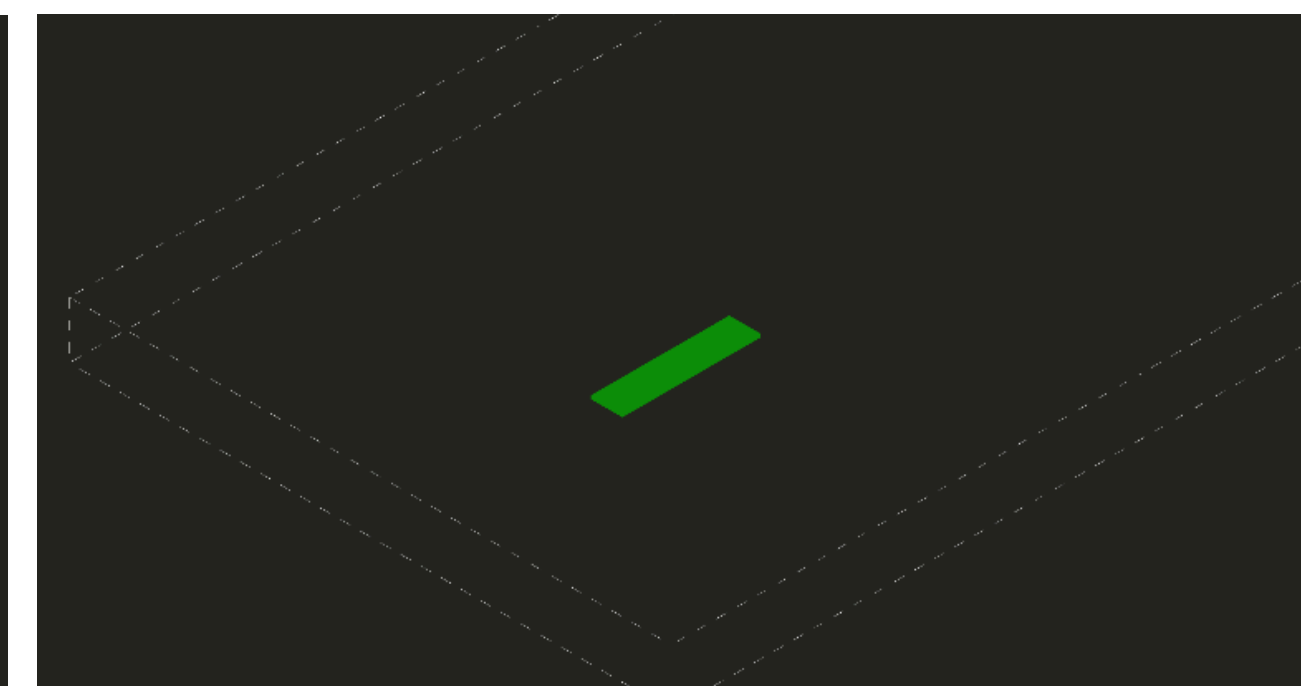
Pen markings



Markings vs. travel



Move order



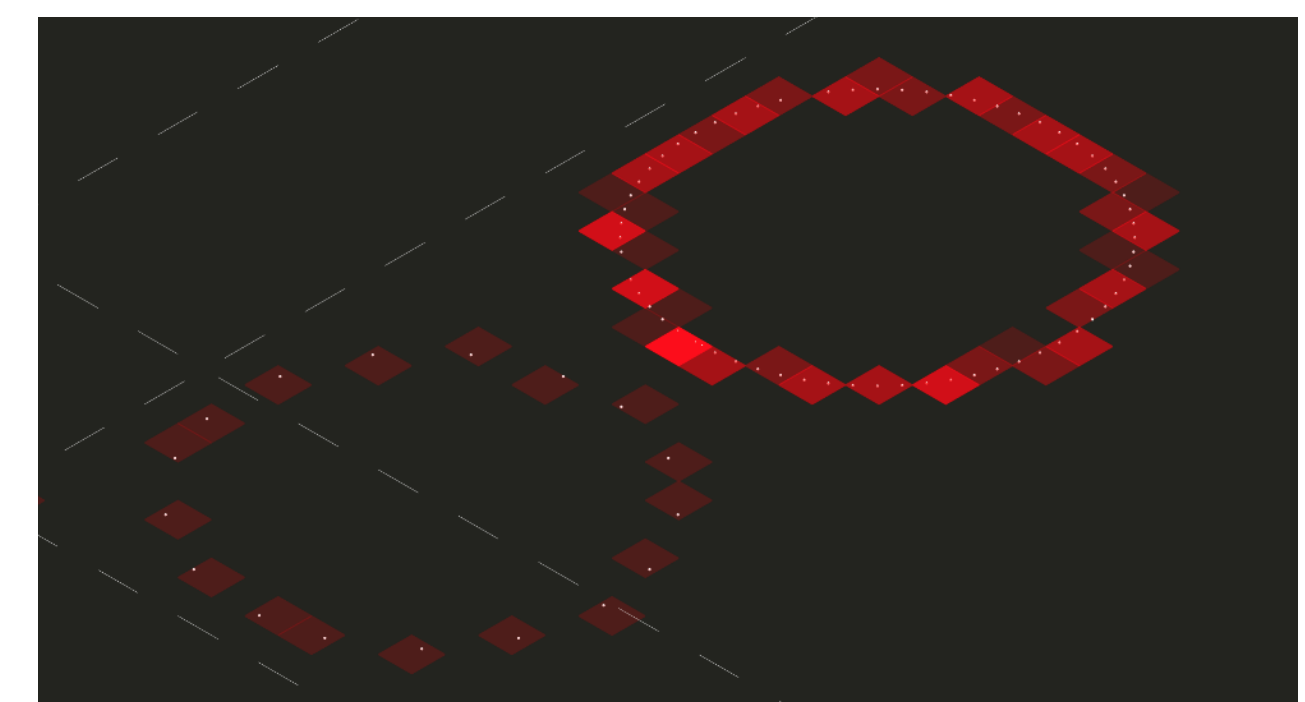
Scale w.r.t. work envelope



Vinyl cutter sharp angles



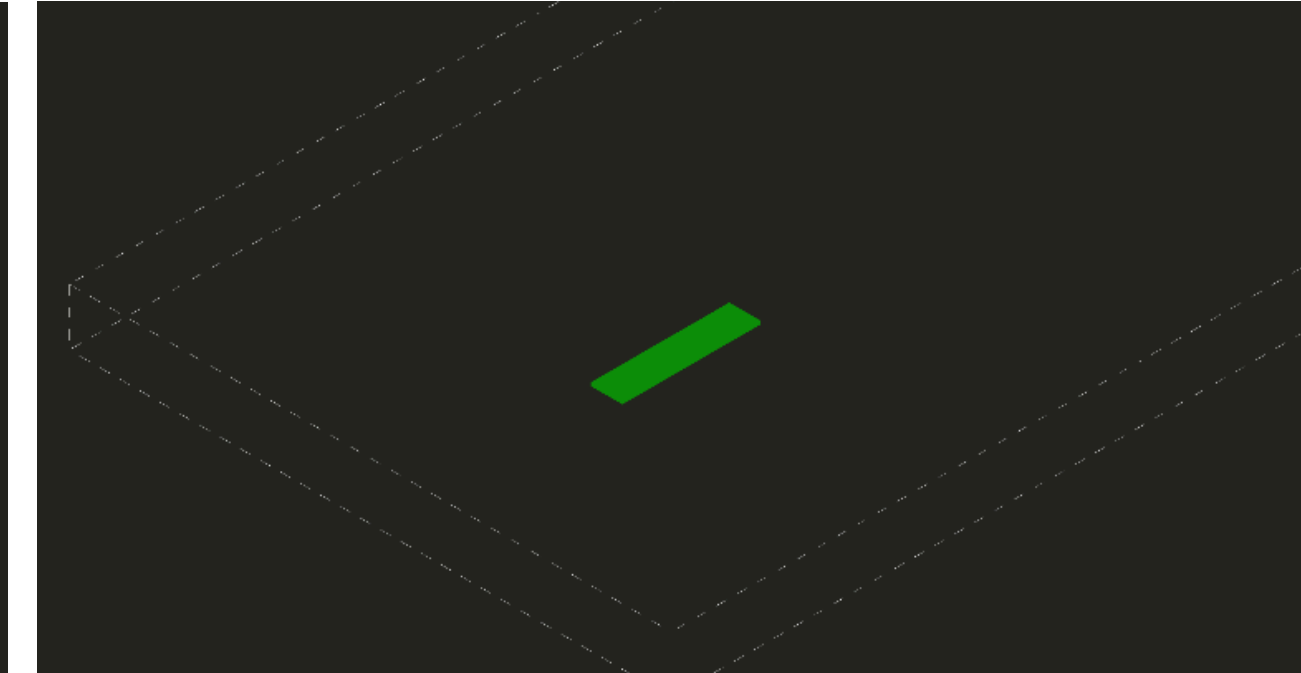
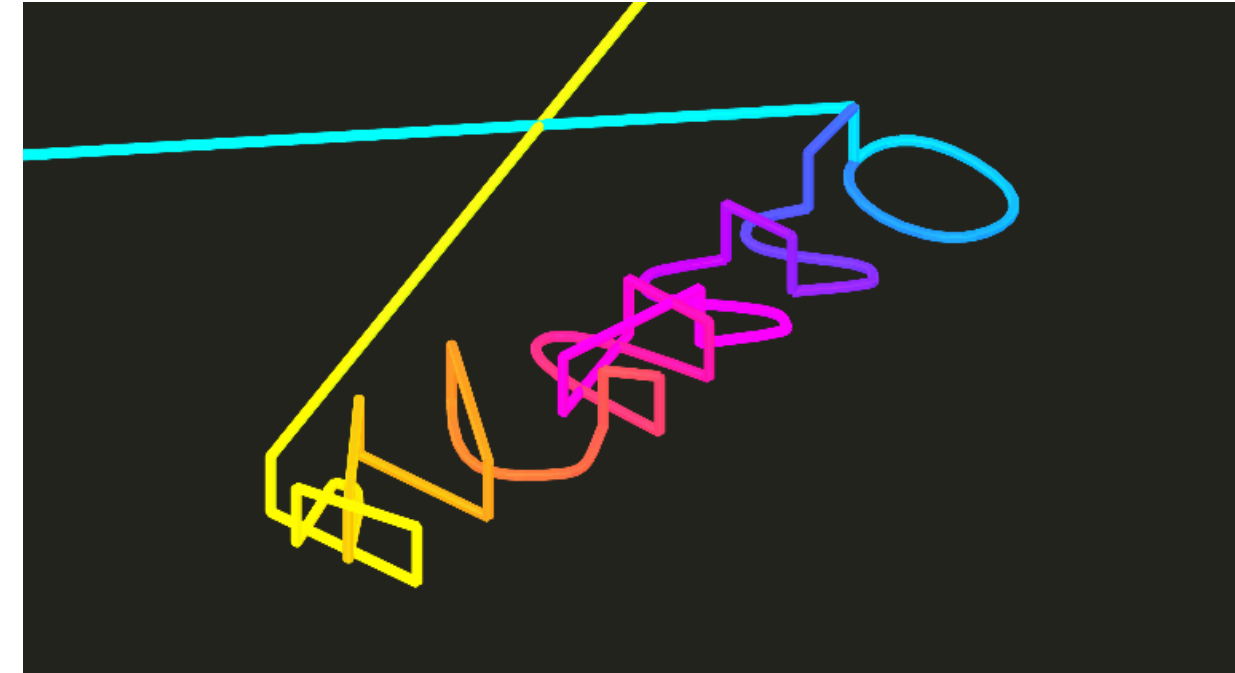
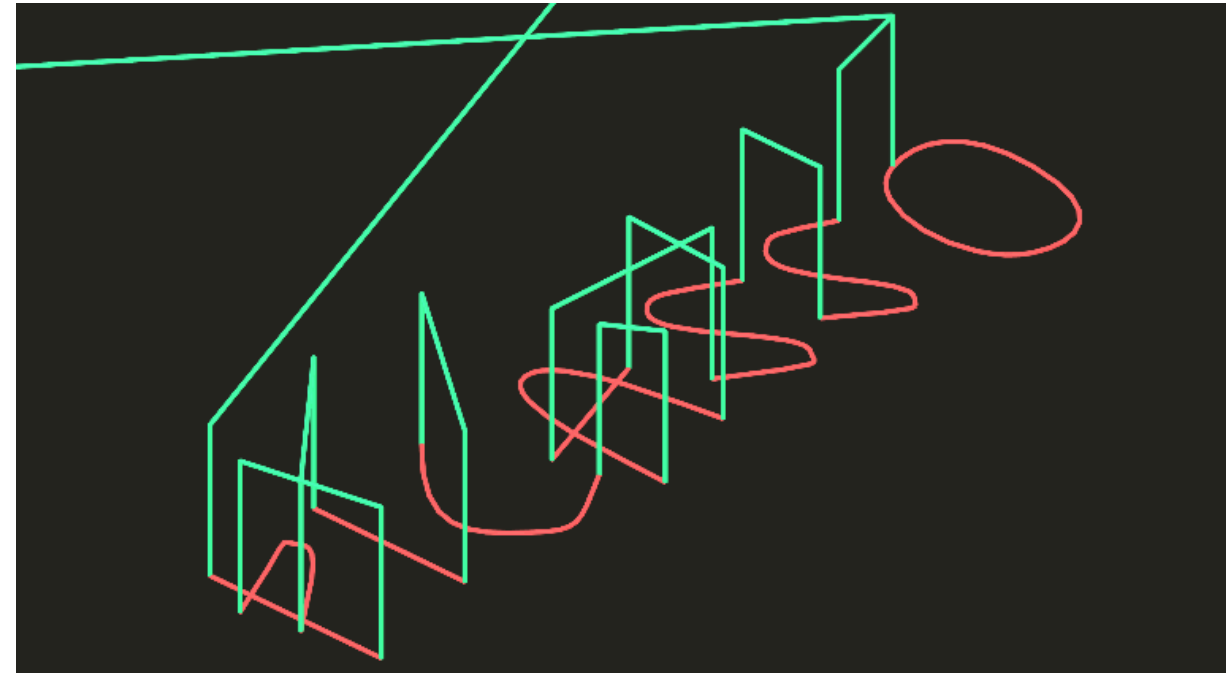
Laser cut geometries



Laser heat map

Data Visualization

Toolpath stylesheets (TSS): generate task-specific views of a given toolpath.



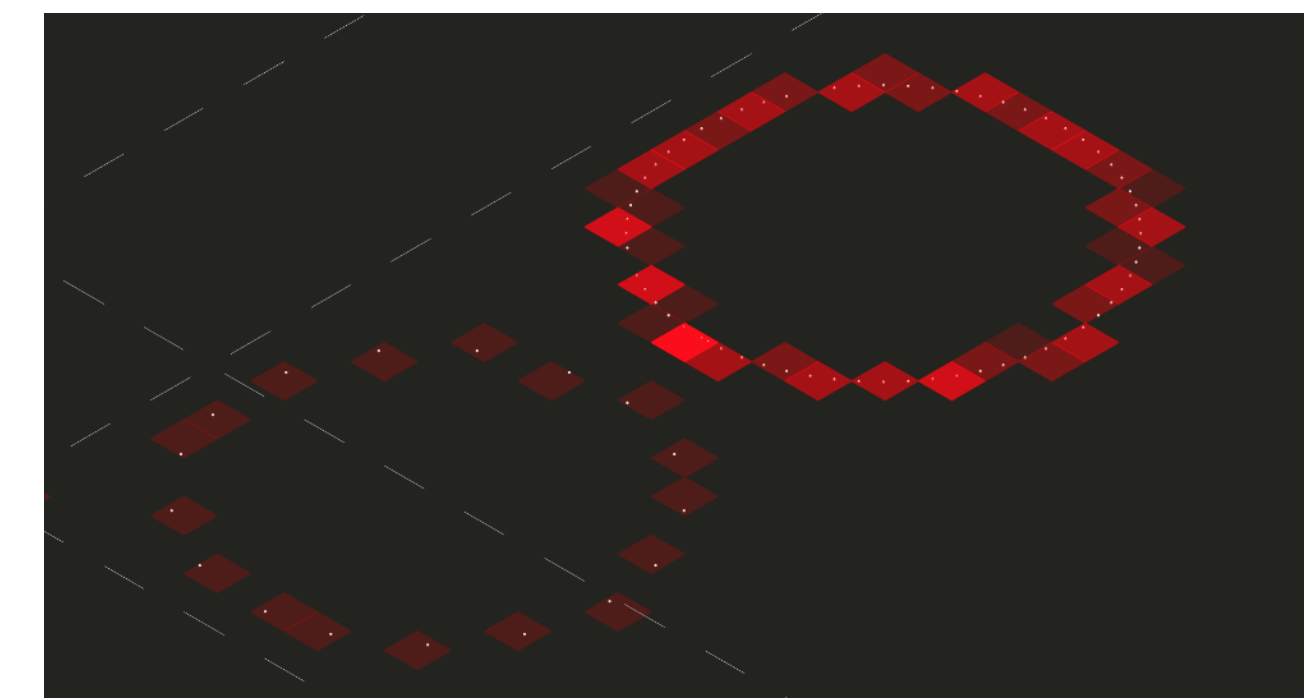
Pen marking



Vinyl

Order

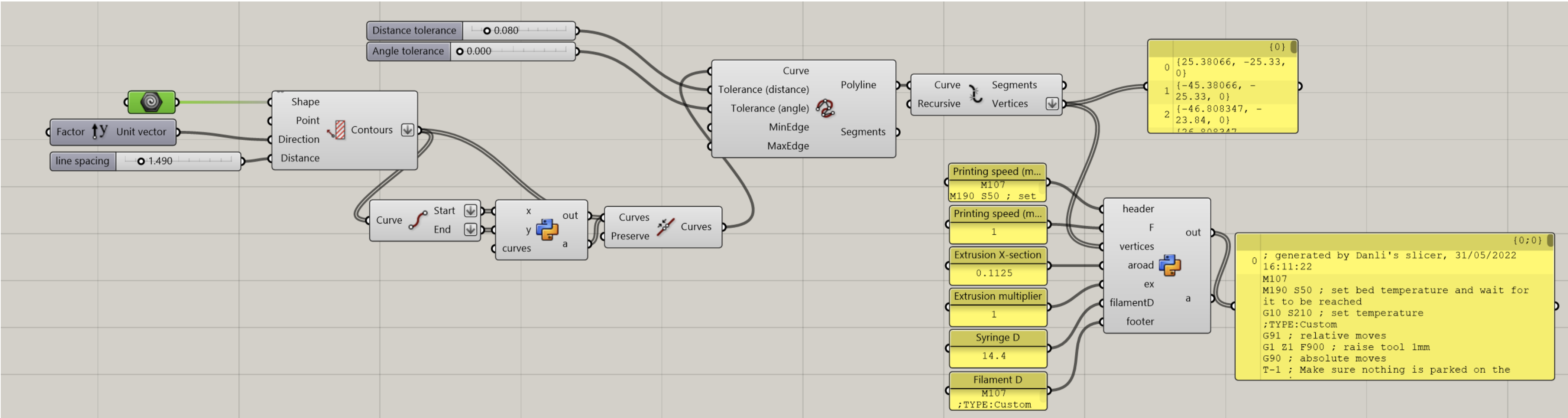
Scale w.r.t. work envelope



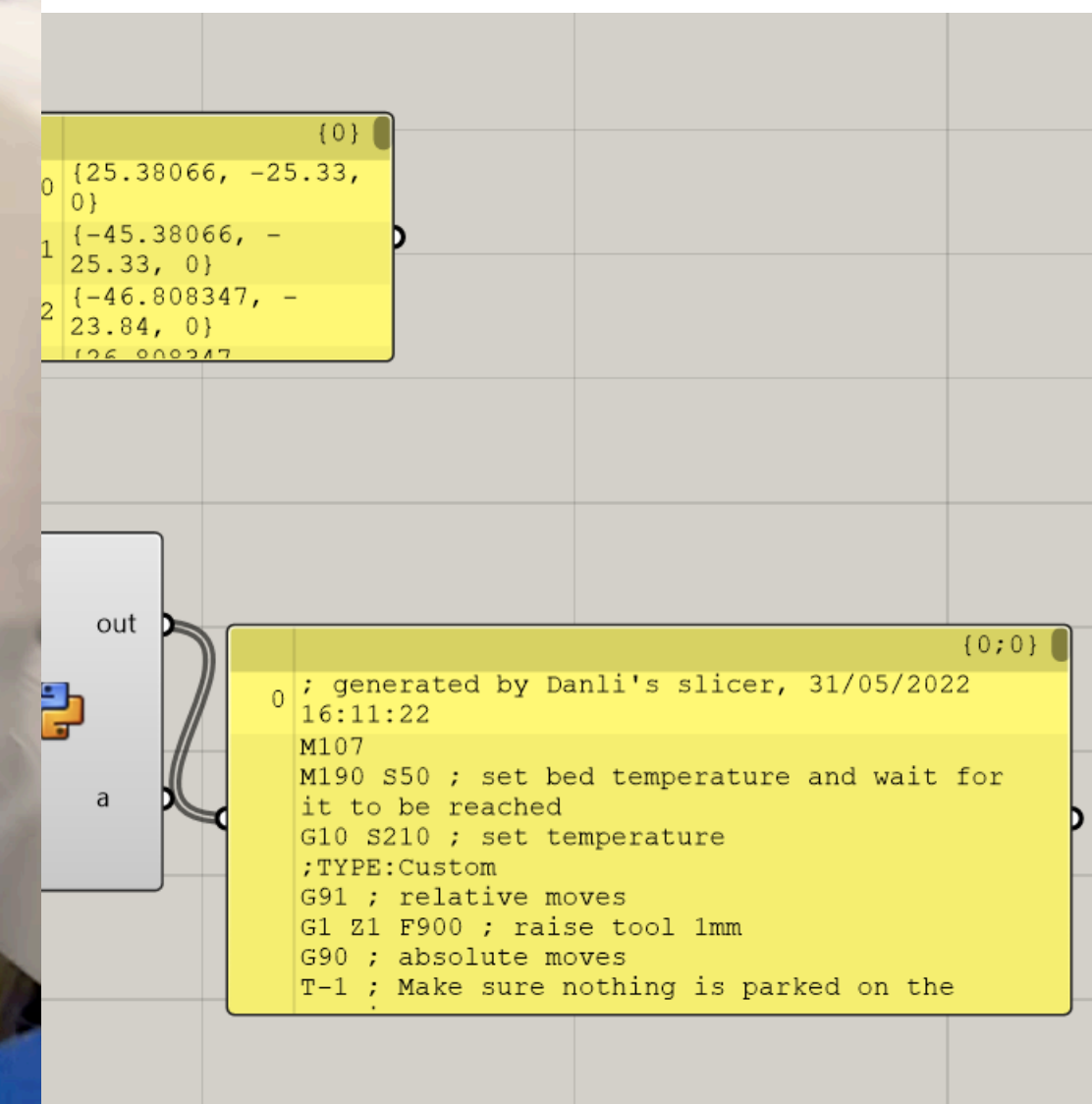
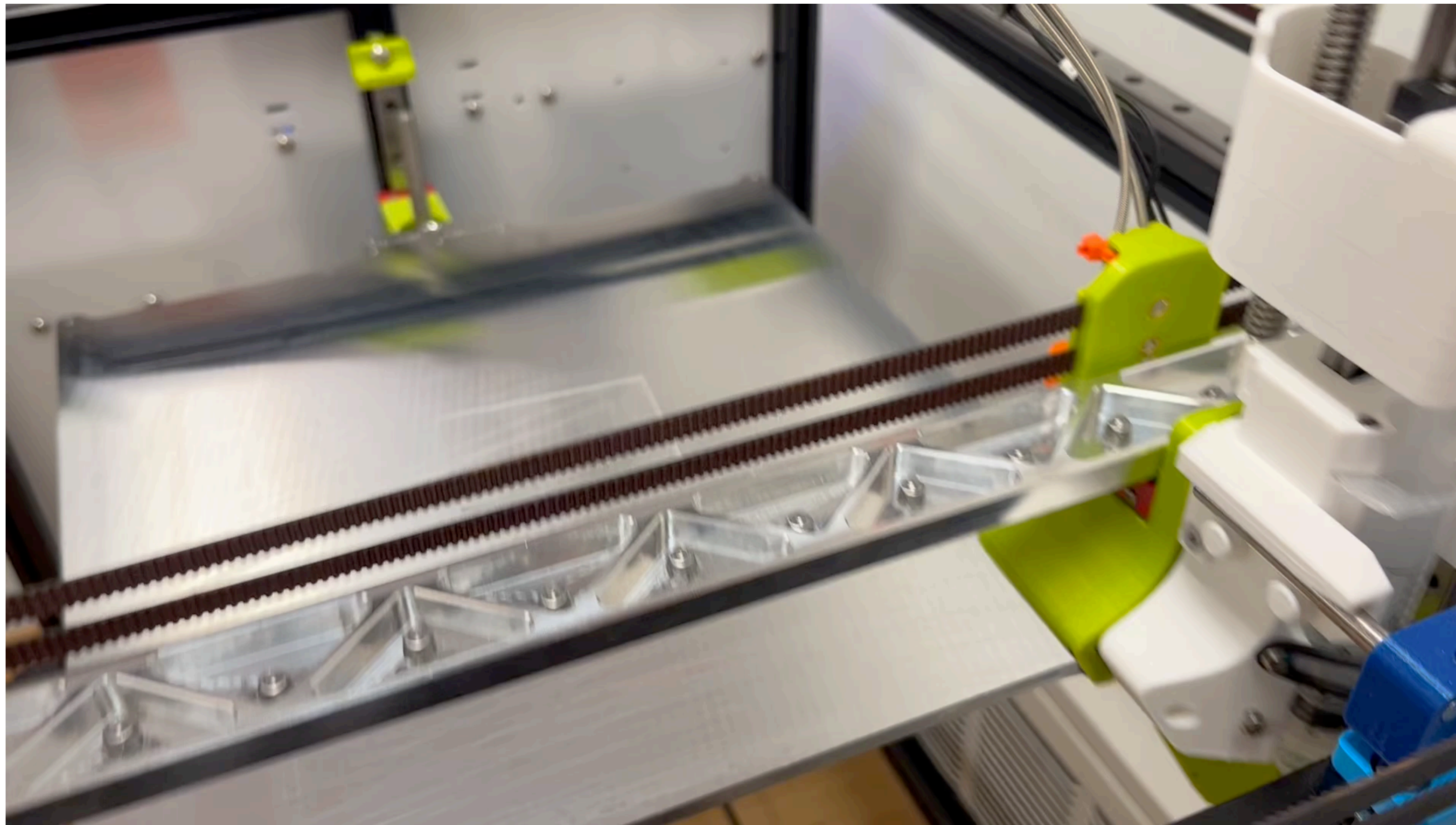
Laser heat map

Knowing exactly where the toolpath will go in physical space is crucial in many EDF applications.

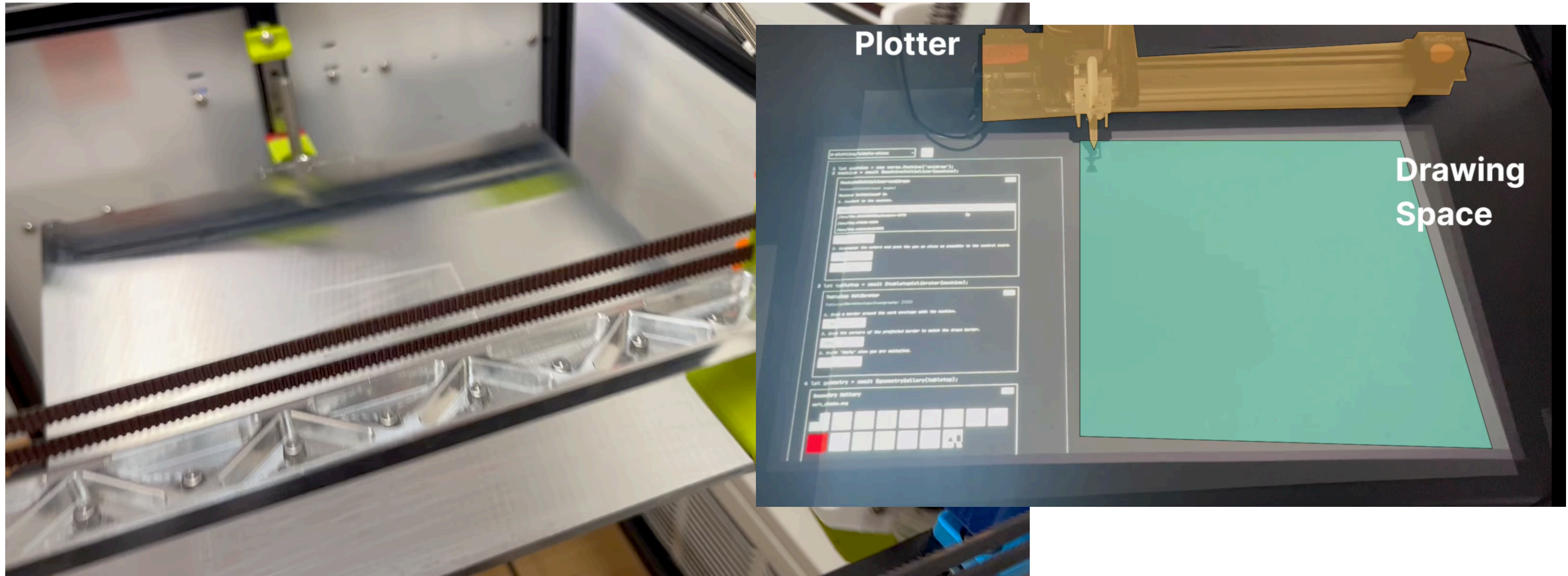
Knowing exactly where the toolpath will go in physical space is crucial in many EDF applications.



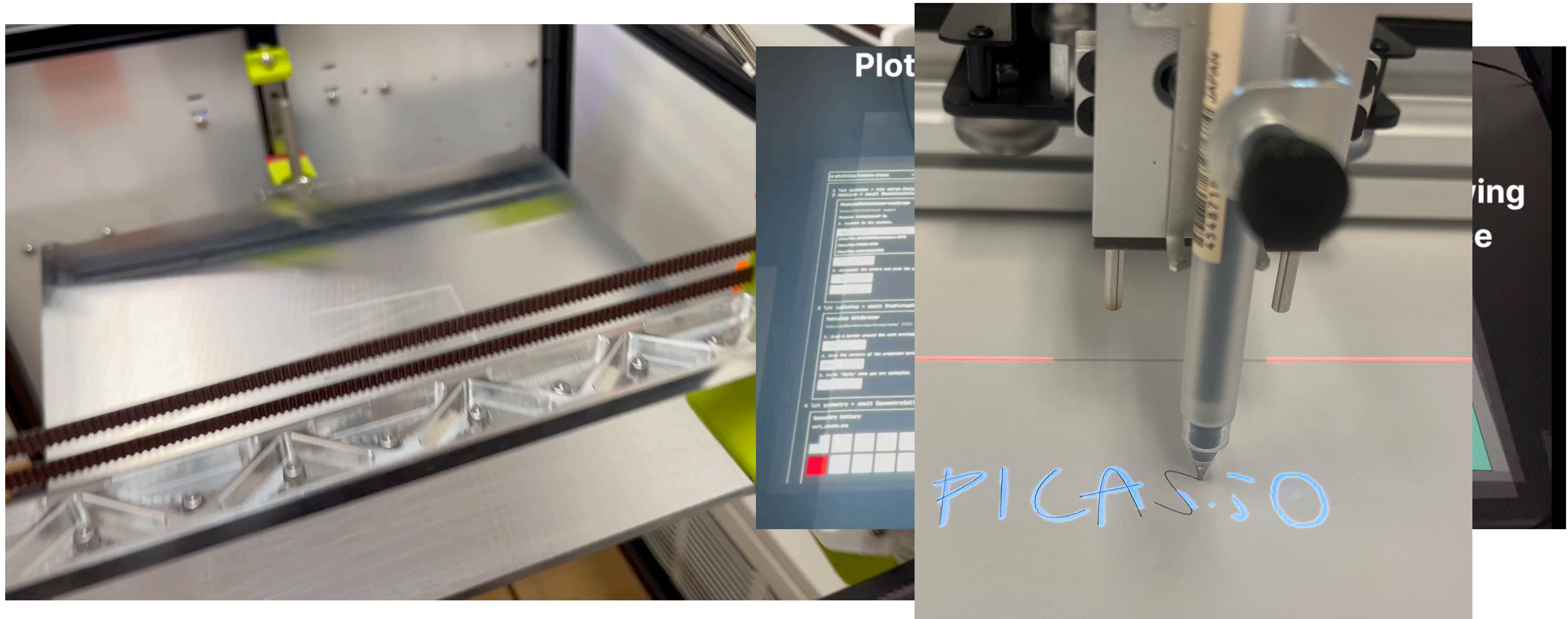
Knowing exactly where the toolpath will go in physical space is crucial in many EDF applications.



Knowing exactly where the toolpath will go in physical space is crucial in many EDF applications.



Knowing exactly where the toolpath will go in physical space is crucial in many EDF applications.



DIANA

a-plotting/simple-place

save

```
1 let machine = new verso.Machine('axidraw');  
2 machine = await $machineInitializer(machine);
```

MachineInitializer<axidraw>
Machine(initialized: maybe)

```
3 let tabletop = await $tabletopCalibrator(machine);
```

Tabletop Calibrator

Tabletop(WorkEnvelope(homography: [?]))

1. Draw a border around the work envelope with the machine.

Draw border

2. Drag the corners of the projected border to match the drawn border.

Unlock Corners

3. Press 'Apply' when you are satisfied.

Apply

```
4 let geometry = await $geometryGallery(tabletop);
```

3x

DIANA

a-plotting/simple-place

save

```
1 let machine = new verso.Machine('axidraw');  
2 machine = await $machineInitializer(machine);
```

MachineInitializer<axidraw>
Machine(initialized: maybe)

```
3 let tabletop = await $tabletopCalibrator(machine);
```

Tabletop Calibrator

Tabletop(WorkEnvelope(homography: [?]))

1. Draw a border around the work envelope with the machine.

Draw border

2. Drag the corners of the projected border to match the drawn border.

Unlock Corners

3. Press 'Apply' when you are satisfied.

Apply

```
4 let geometry = await $geometryGallery(tabletop);
```

3x

9 await \$dispatcher(machine, [toolpath]),

Dispatcher

Machine(initialized: maybe)

Machine status: free

; Enter G-code to adjust the tool prior to dispatch.

Send Snippet

Toolpath 0

Dispatch

Pause

9 await \$dispatcher(machine, [toolpath]),

Dispatcher

Machine(initialized: maybe)

Machine status: free

; Enter G-code to adjust the tool prior to dispatch.

Send Snippet

Toolpath 0

Dispatch

Pause

Next steps for fabrication-as-programming

Next steps for fabrication-as-programming

More visualizations with toolpath stylesheets

Next steps for fabrication-as-programming

More visualizations with toolpath stylesheets

More modules for emerging human-machine interactions

Next steps for fabrication-as-programming

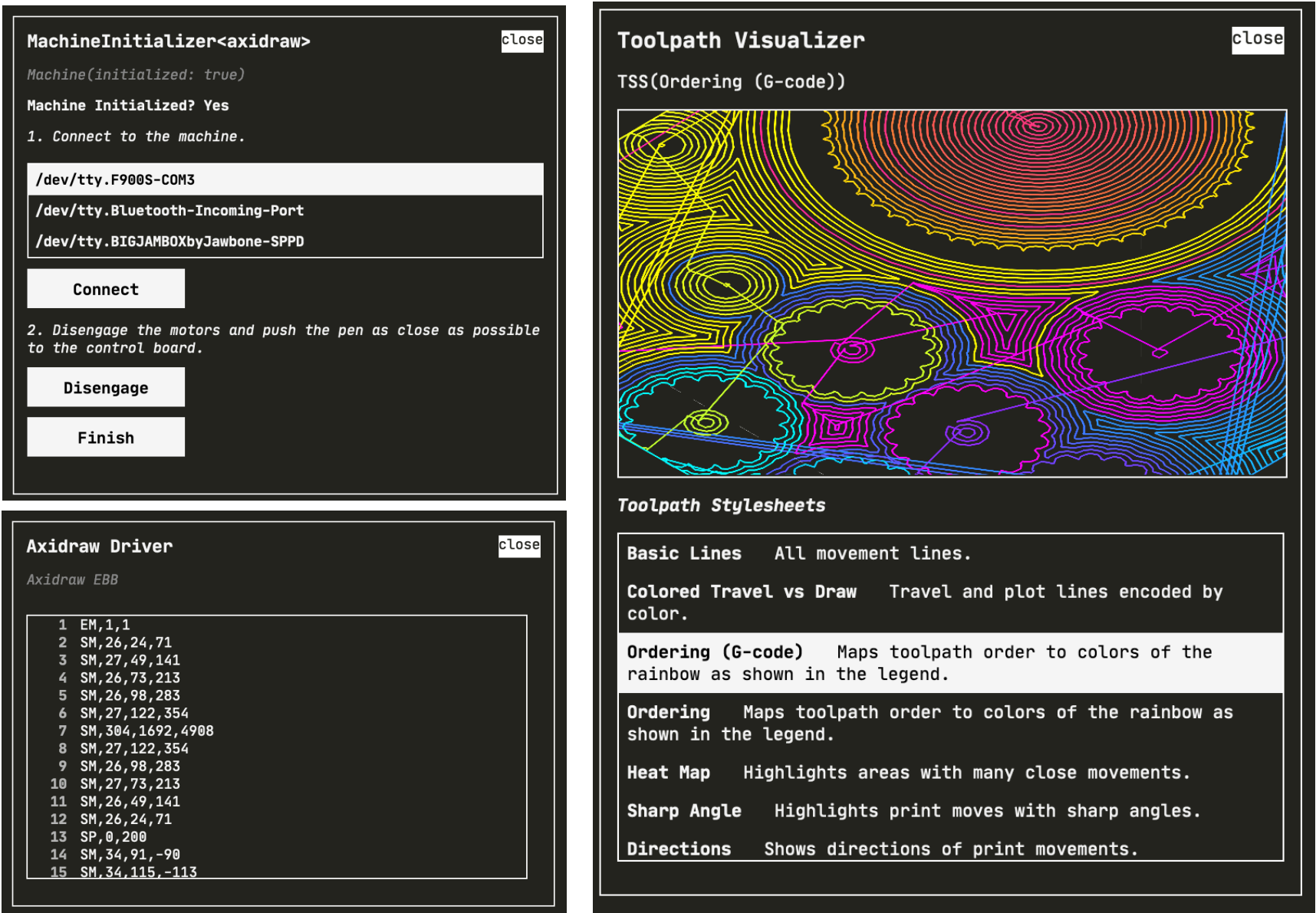
More visualizations with toolpath stylesheets

More modules for emerging human-machine interactions

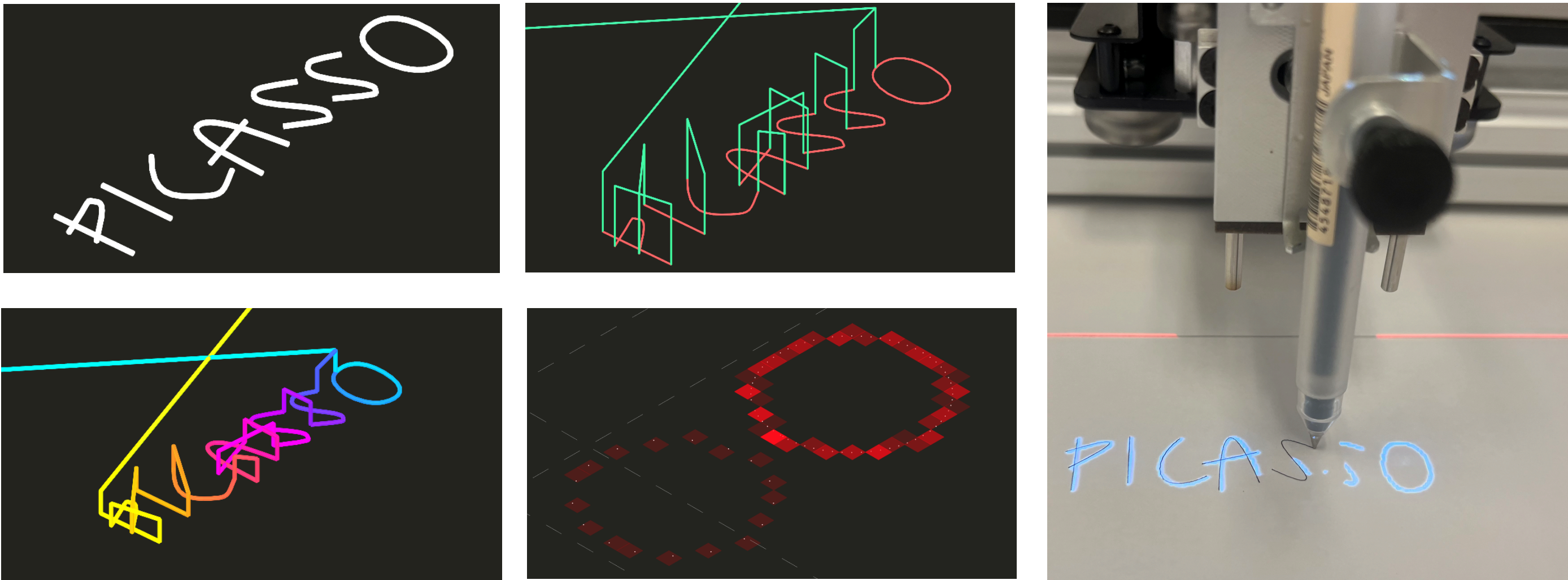
More sharing of exploratory workflows

Verso: Towards Fabrication-as-Programming

Multimodal Programming



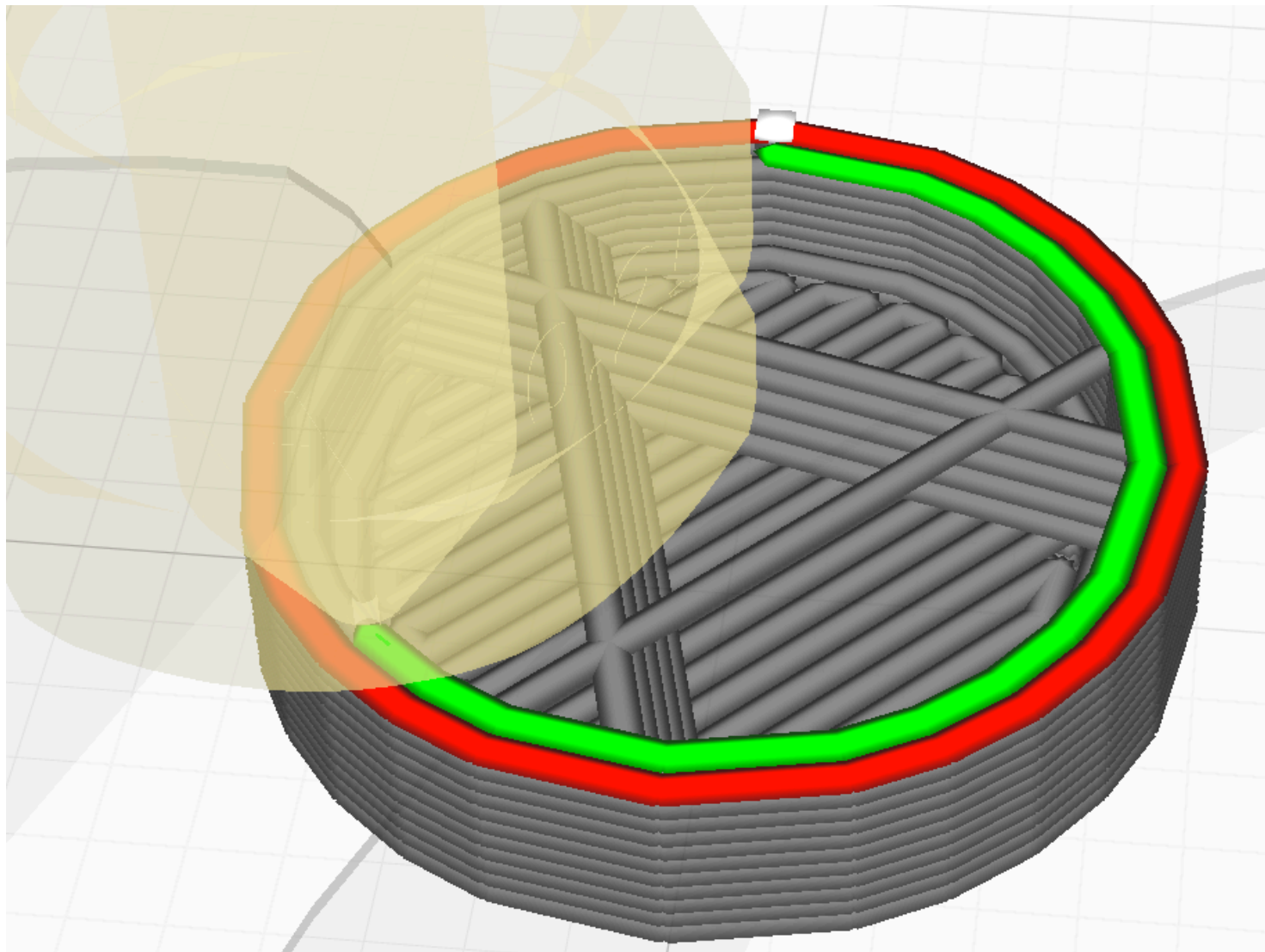
Toolpath Stylesheets



Jasper Tran O’Leary, Eunice Jun, Nadya Peek
University of Washington

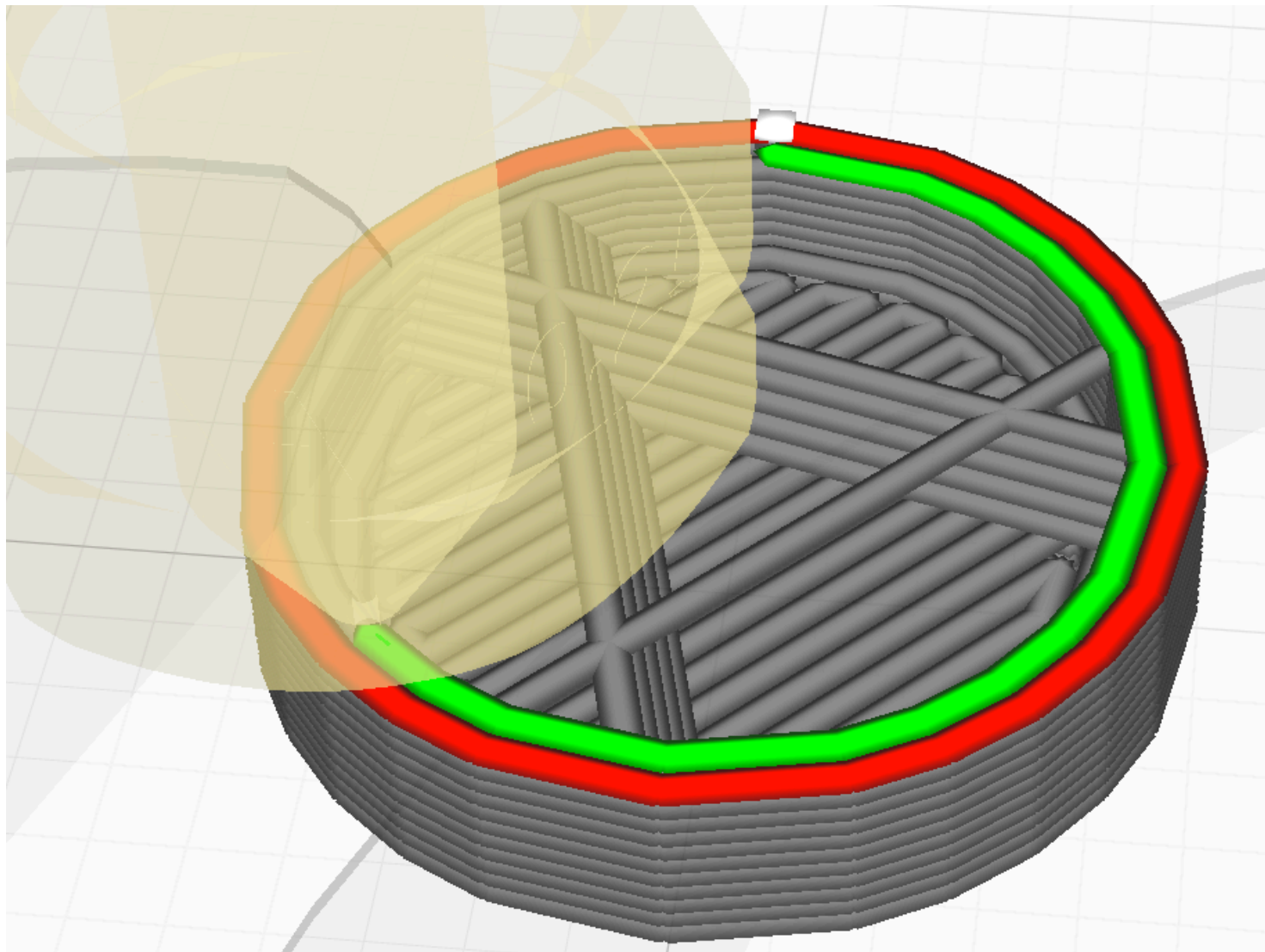


“Can’t Cura/Fusion360/etc... already visualize toolpaths?”



Yes, and they typically provide great common-case visualization.

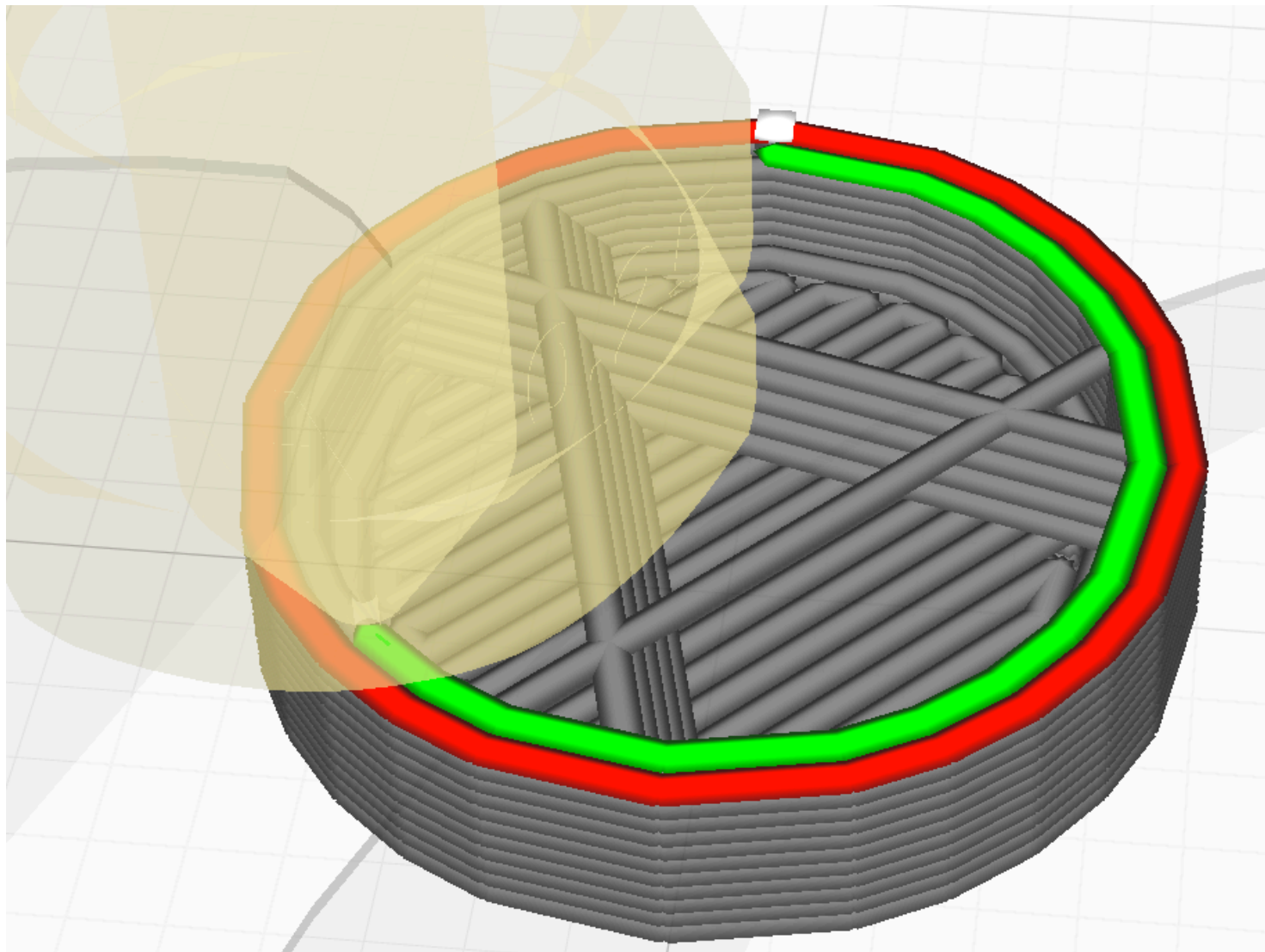
“Can’t Cura/Fusion360/etc... already visualize toolpaths?”



Yes, and they typically provide great common-case visualization.

However we’re focused on uncommon tasks which may require specialized visualizations.

“Can’t Cura/Fusion360/etc... already visualize toolpaths?”



Yes, and they typically provide great common-case visualization.

However we’re focused on uncommon tasks which may require specialized visualizations.

TSS are extensible.

A TSS is an interpreter that provides a ***visual semantics*** for a given set of instructions.

G55

M8

G0 X73.332 Y62.712

G0 Z15.

G0 Z5.

G0 Z-3.498

G1 Z-4.498 F400.

G1 X73.331 Y62.704 Z-4.618

G1 X73.325 Y62.683 Z-4.737

G1 X73.316 Y62.649 Z-4.852

G1 X73.304 Y62.601 Z-4.962

G1 X73.288 Y62.54 Z-5.066

G1 X73.144 Y61.975 Z-5.469

G1 X73.114 Y61.86 Z-5.49

G1 X73.084 Y61.743 Z-5.498

G3 X72.693 Y59.021 R12.765

G1 Y58.871 F800.

Algorithm 1 Ordering TSS (G-Code Instruction Set)

Require: *instructions*

points $\leftarrow$ []

for *instruction* in *instructions* **do**

opcode $\leftarrow$ *parseOpcode(instruction)*

match *opcode* **do**

case "G0" or "G1"

X, Y, Z, F, E $\leftarrow$ *parseArgs(instruction)*

push(points, Vector3(X, Y, Z))

G55

M8

G0 X73.332 Y62.712

G0 Z15.

G0 Z5.

G0 Z-3.498

G1 Z-4.498 F400.

G1 X73.331 Y62.704 Z-4.618

G1 X73.325 Y62.683 Z-4.737

G1 X73.316 Y62.649 Z-4.852

G1 X73.304 Y62.601 Z-4.962

G1 X73.288 Y62.54 Z-5.066

G1 X73.144 Y61.975 Z-5.469

G1 X73.114 Y61.86 Z-5.49

G1 X73.084 Y61.743 Z-5.498

G3 X72.693 Y59.021 R12.765

G1 Y58.871 F800.

Algorithm 1 Ordering TSS (G-Code Instruction Set)

Require: *instructions*

points $\leftarrow$ []

for *instruction* in *instructions* **do**

opcode $\leftarrow$ *parseOpcode(instruction)*

match *opcode* **do**

case "G0" or "G1"

X, Y, Z, F, E $\leftarrow$ *parseArgs(instruction)*

push(points, Vector3(X, Y, Z))

Algorithm 1 Ordering TSS (G-Code Instruction Set)

Require: *instructions*

points $\leftarrow$ []

for *instruction* in *instructions* **do**

opcode $\leftarrow$ *parseOpcode(instruction)*

match *opcode* **do**

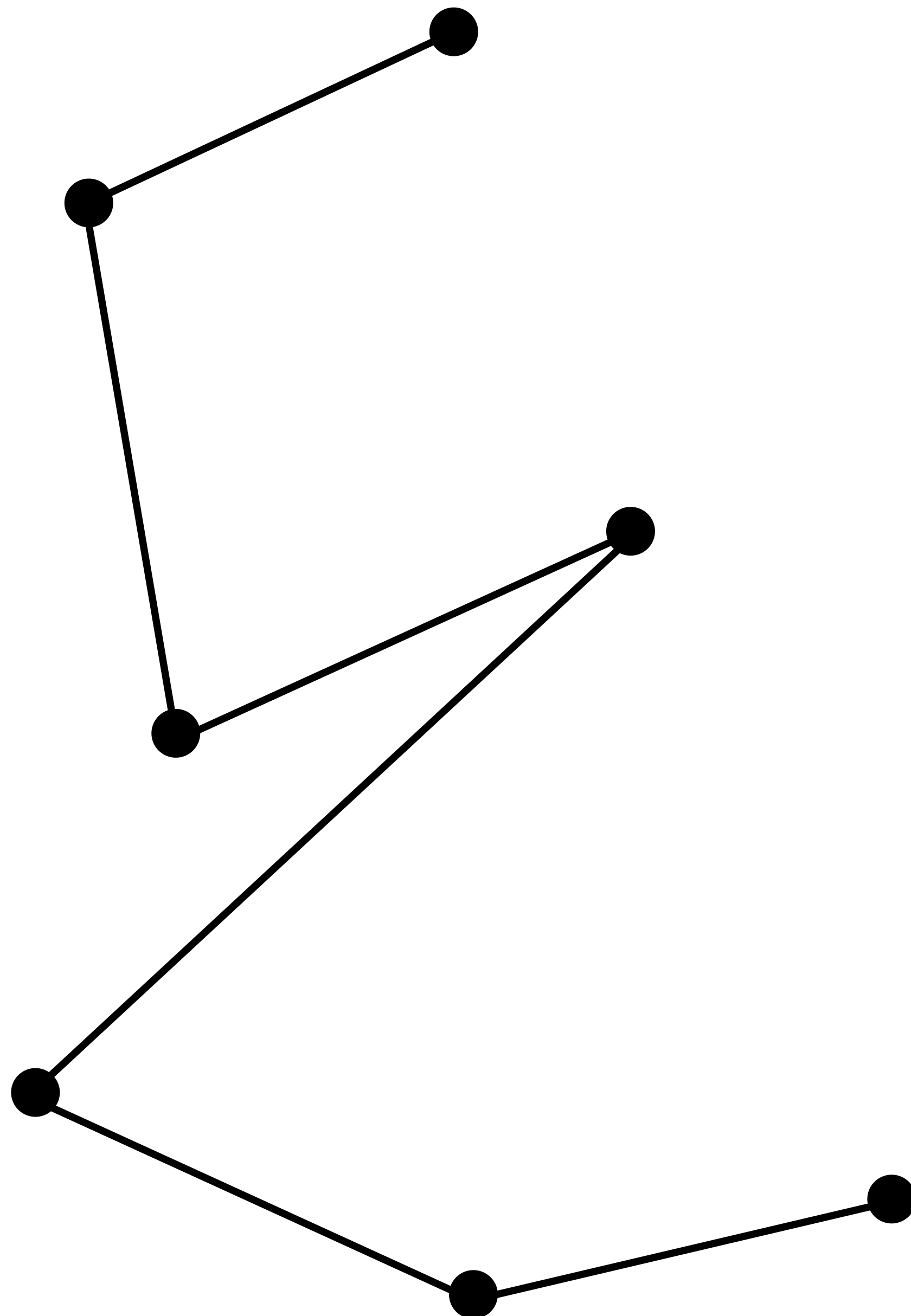
case "G0" or "G1"

X, Y, Z, F, E $\leftarrow$ *parseArgs(instruction)*

push(points, Vector3(X, Y, Z))

end for

curve $\leftarrow$ *interpolate(points)*



Algorithm 1 Ordering TSS (G-Code Instruction Set)

Require: *instructions*

points $\leftarrow$ []

for *instruction* in *instructions* **do**

$opcode \leftarrow \text{parseOpcode}(\text{instruction})$

match *opcode* **do**

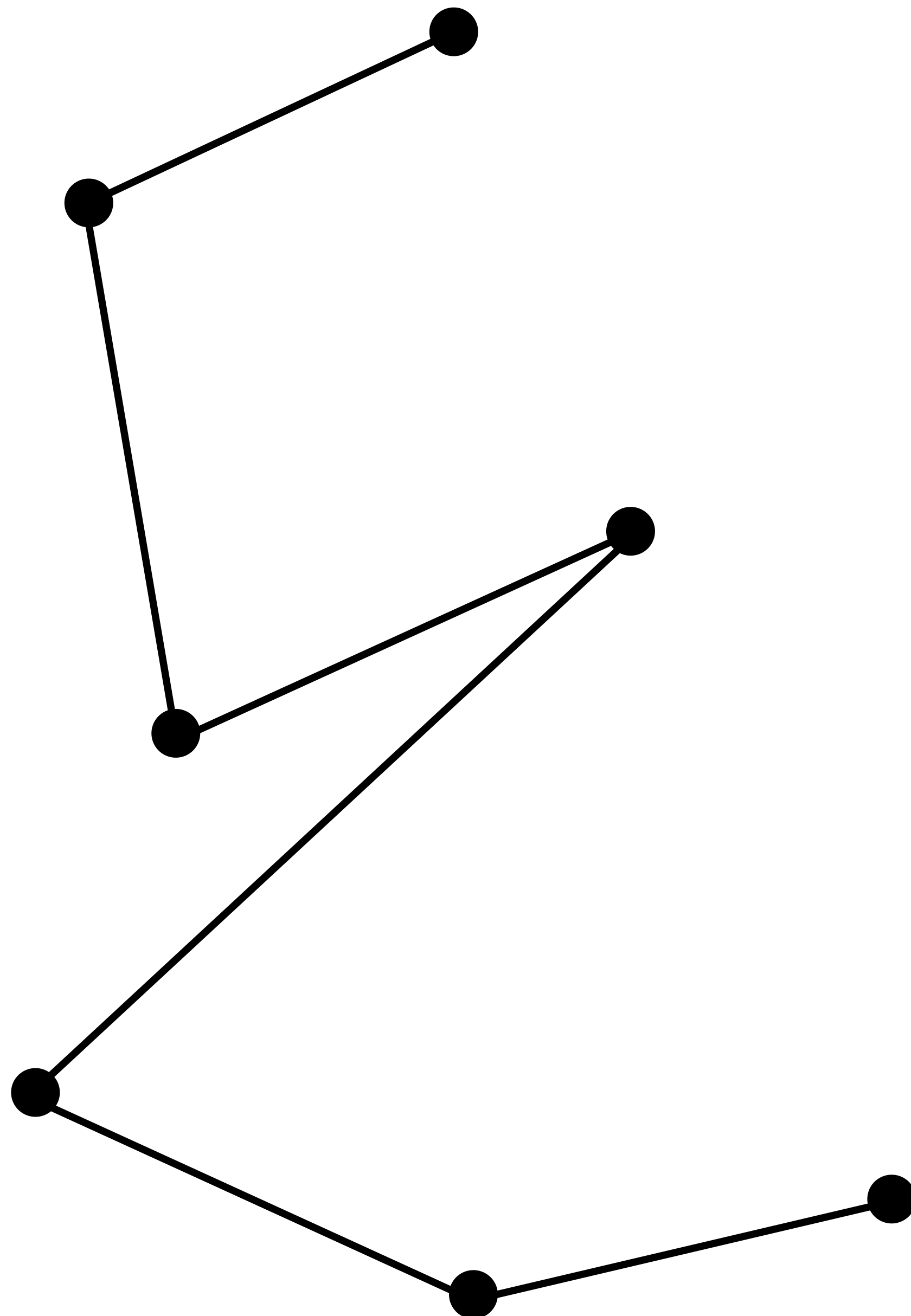
case "G0" or "G1"

$X, Y, Z, F, E \leftarrow \text{parseArgs}(\text{instruction})$

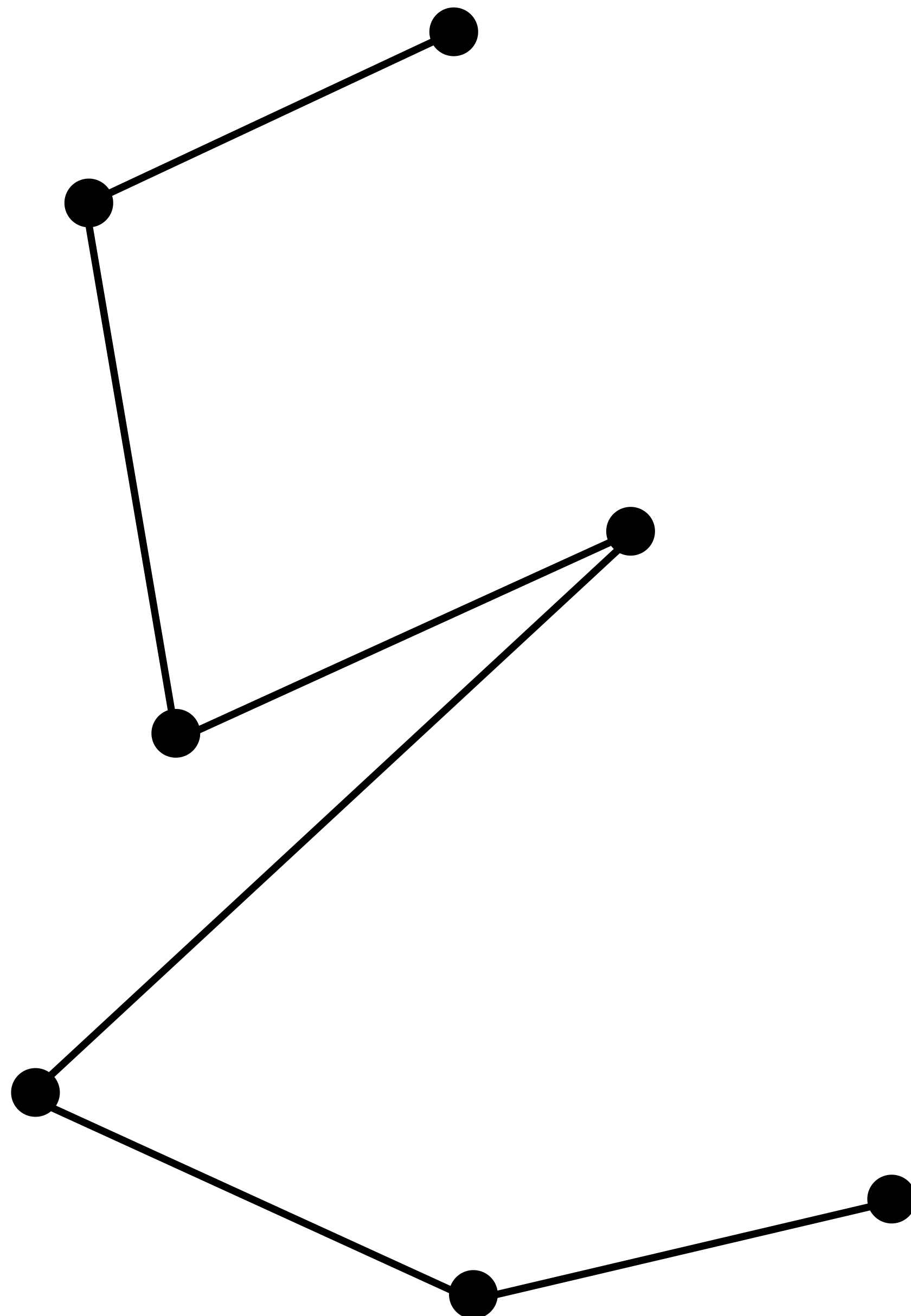
$\text{push}(\text{points}, \text{Vector3}(X, Y, Z))$

end for

curve $\leftarrow \text{interpolate}(\text{points})$



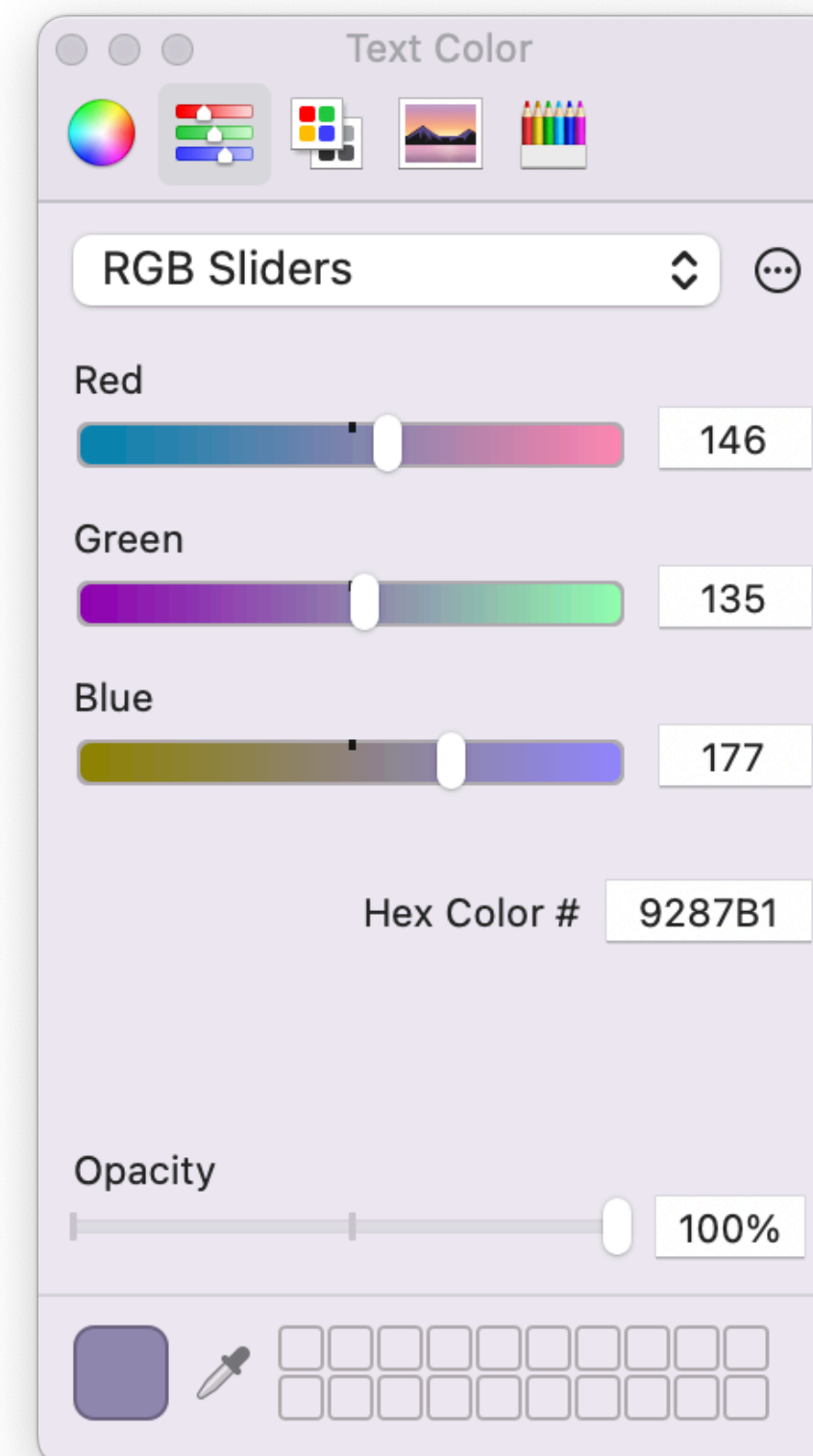
```
colors ← [ ]  
f ← ...      ▷ Choose a constant frequency for cycling colors.  
ϕr, ϕg, ϕb ← ...  ▷ Choose a constant phase offset per channel.  
for (_, index) in curve do  
    red ← sin(f × index + ϕr)  
    green ← sin(f × index + ϕg)  
    blue ← sin(f × index + ϕb)  
    push(colors, Color(red, blue, green))  
end for
```

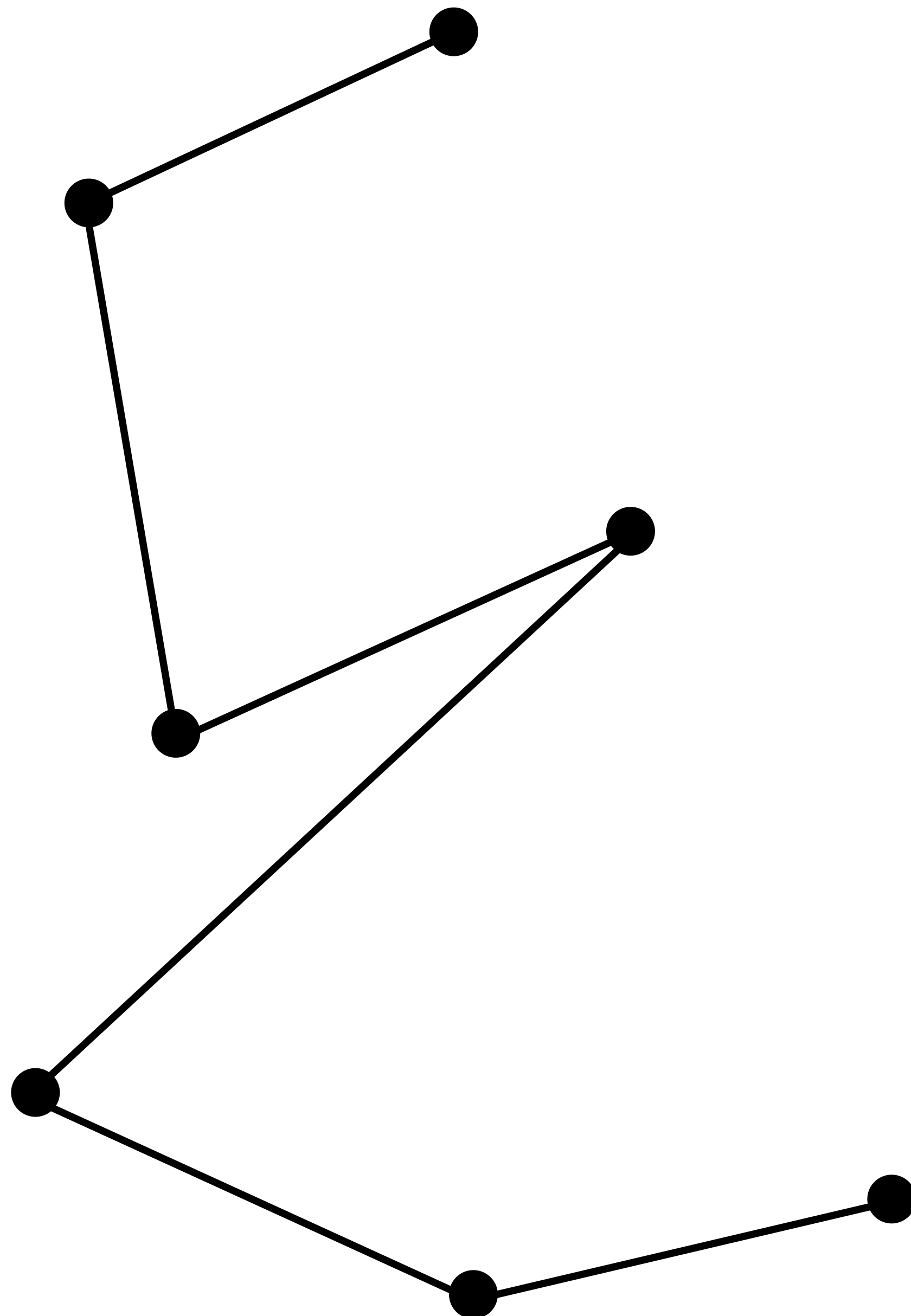


```

colors ← [ ]
f ← ...      ▷ Choose a constant frequency for cycling colors.
 $\phi_r, \phi_g, \phi_b \leftarrow \dots$   ▷ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ←  $\sin(f \times \text{index} + \phi_r)$ 
  green ←  $\sin(f \times \text{index} + \phi_g)$ 
  blue ←  $\sin(f \times \text{index} + \phi_b)$ 
  push(colors, Color(red, blue, green))
end for

```

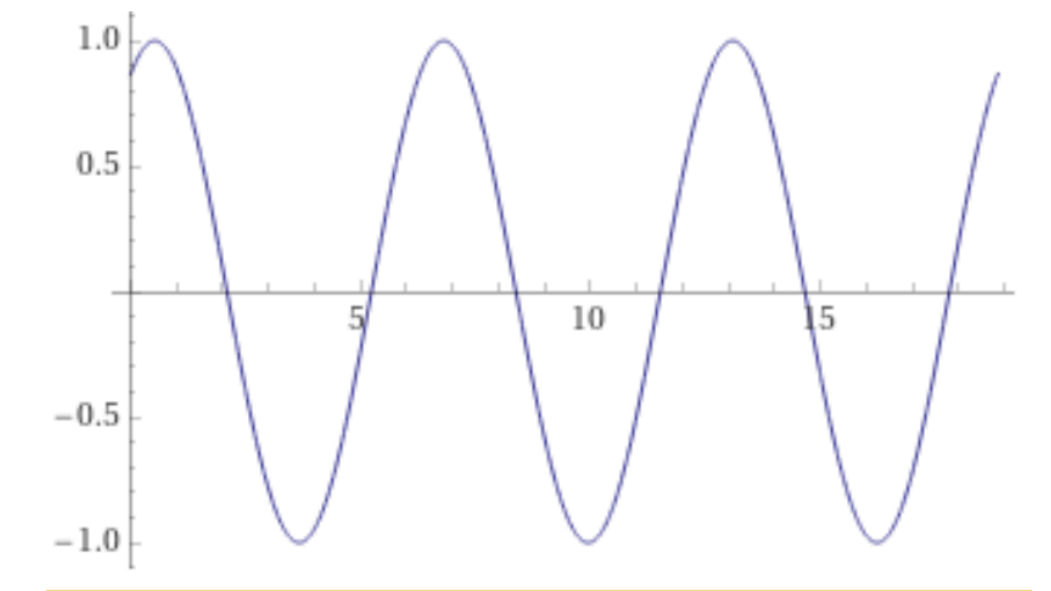
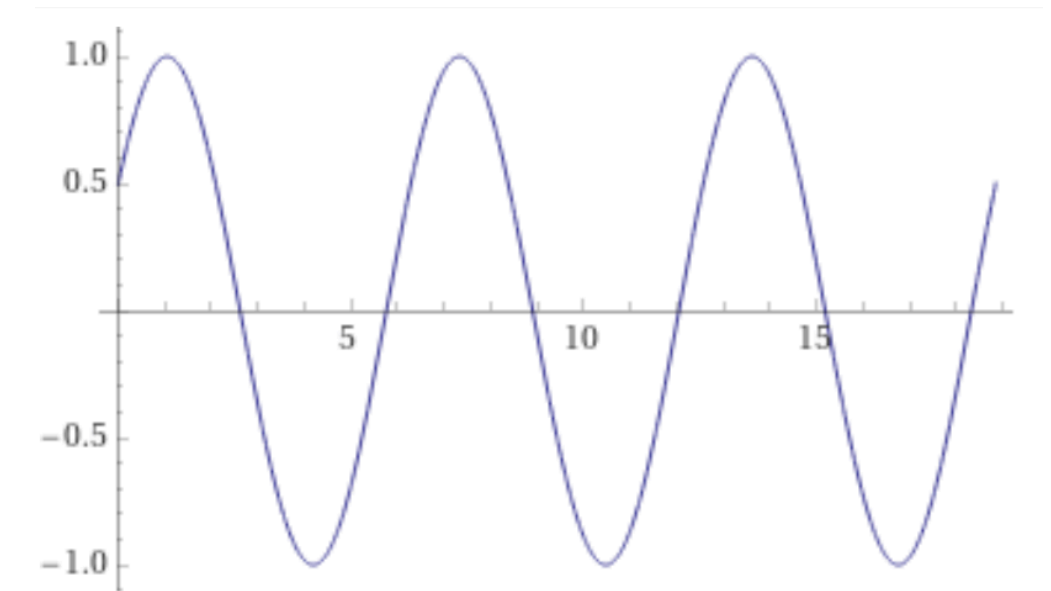
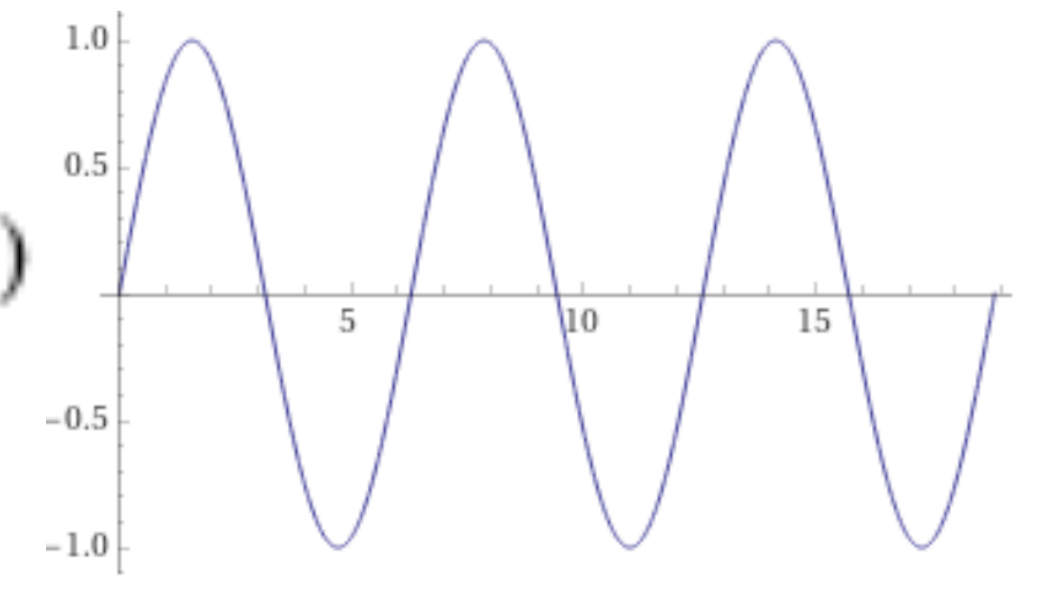
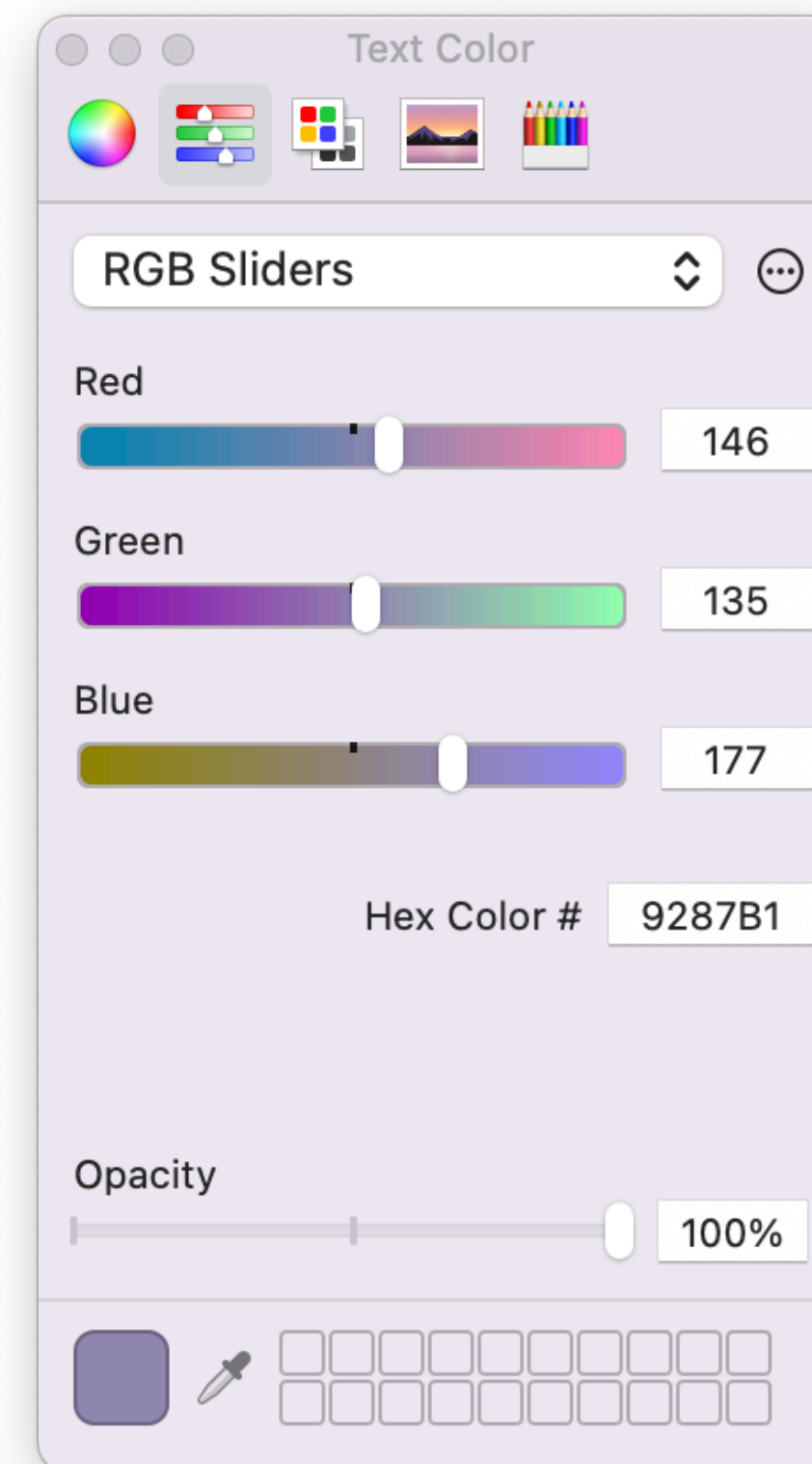


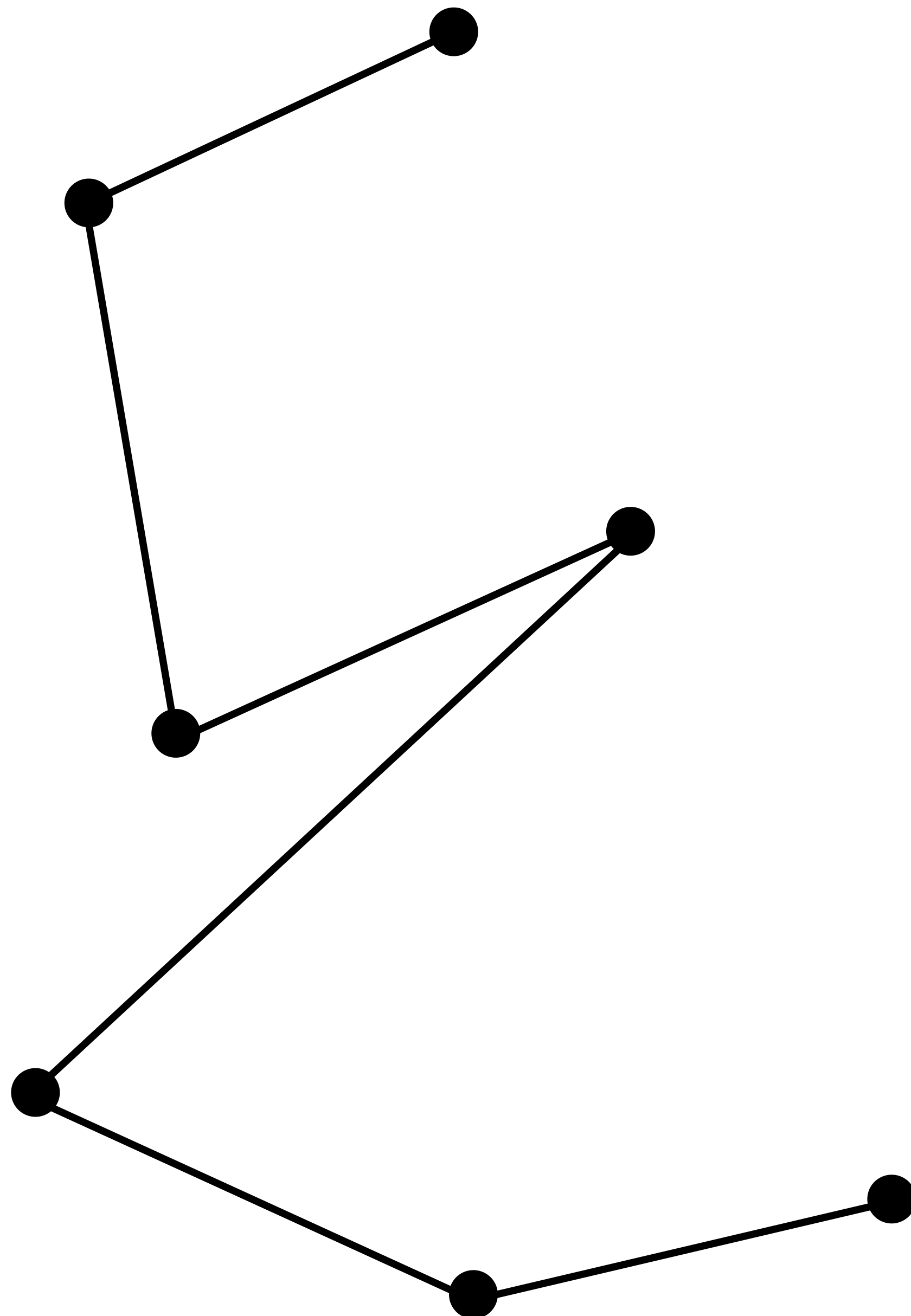


```

colors ← [ ]
f ← ...      ▶ Choose a constant frequency for cycling colors.
 $\phi_r, \phi_g, \phi_b \leftarrow \dots$  ▶ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ←  $\sin(f \times \text{index} + \phi_r)$ 
  green ←  $\sin(f \times \text{index} + \phi_g)$ 
  blue ←  $\sin(f \times \text{index} + \phi_b)$ 
  push(colors, Color(red, blue, green))
end for

```

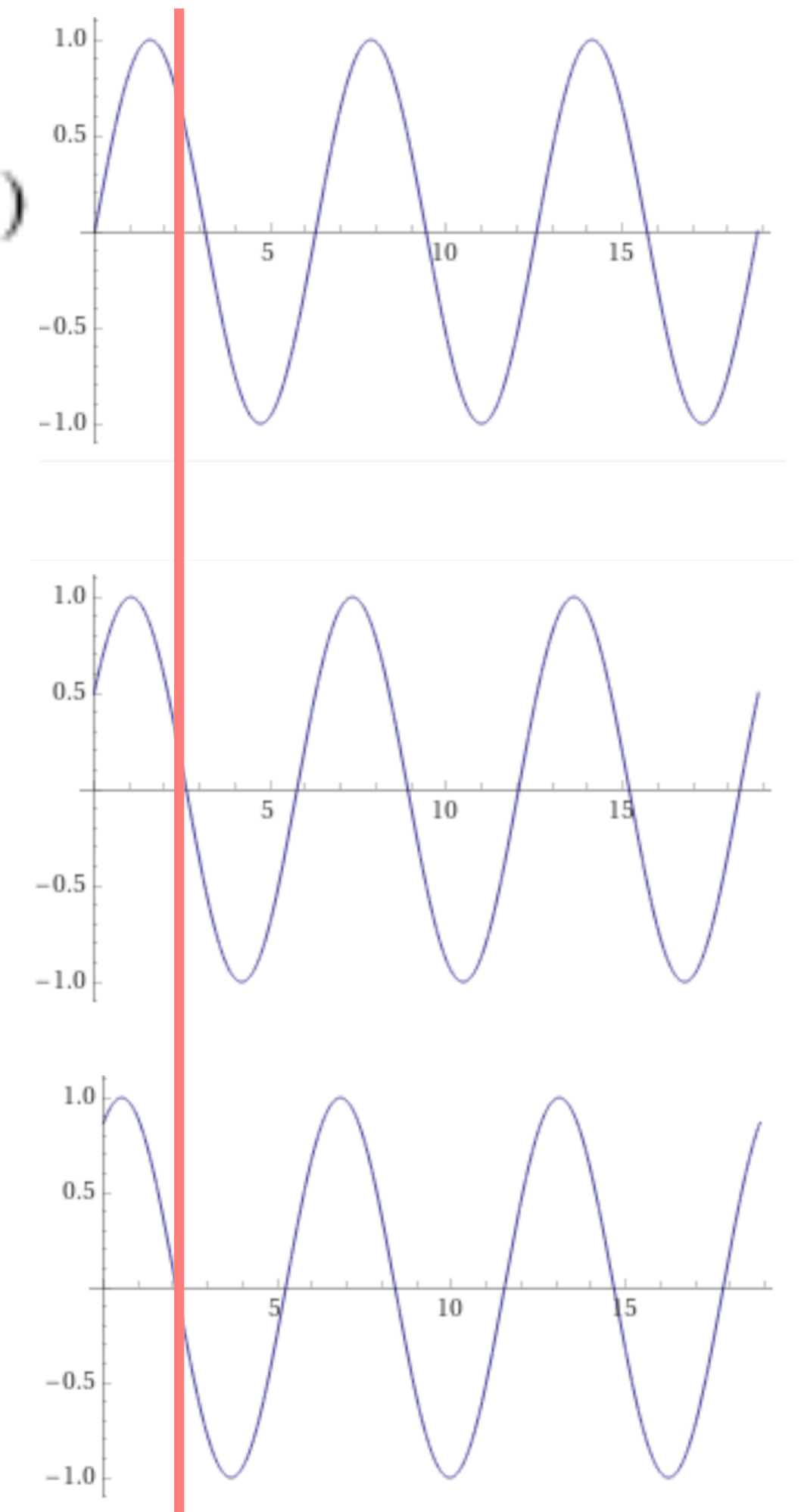
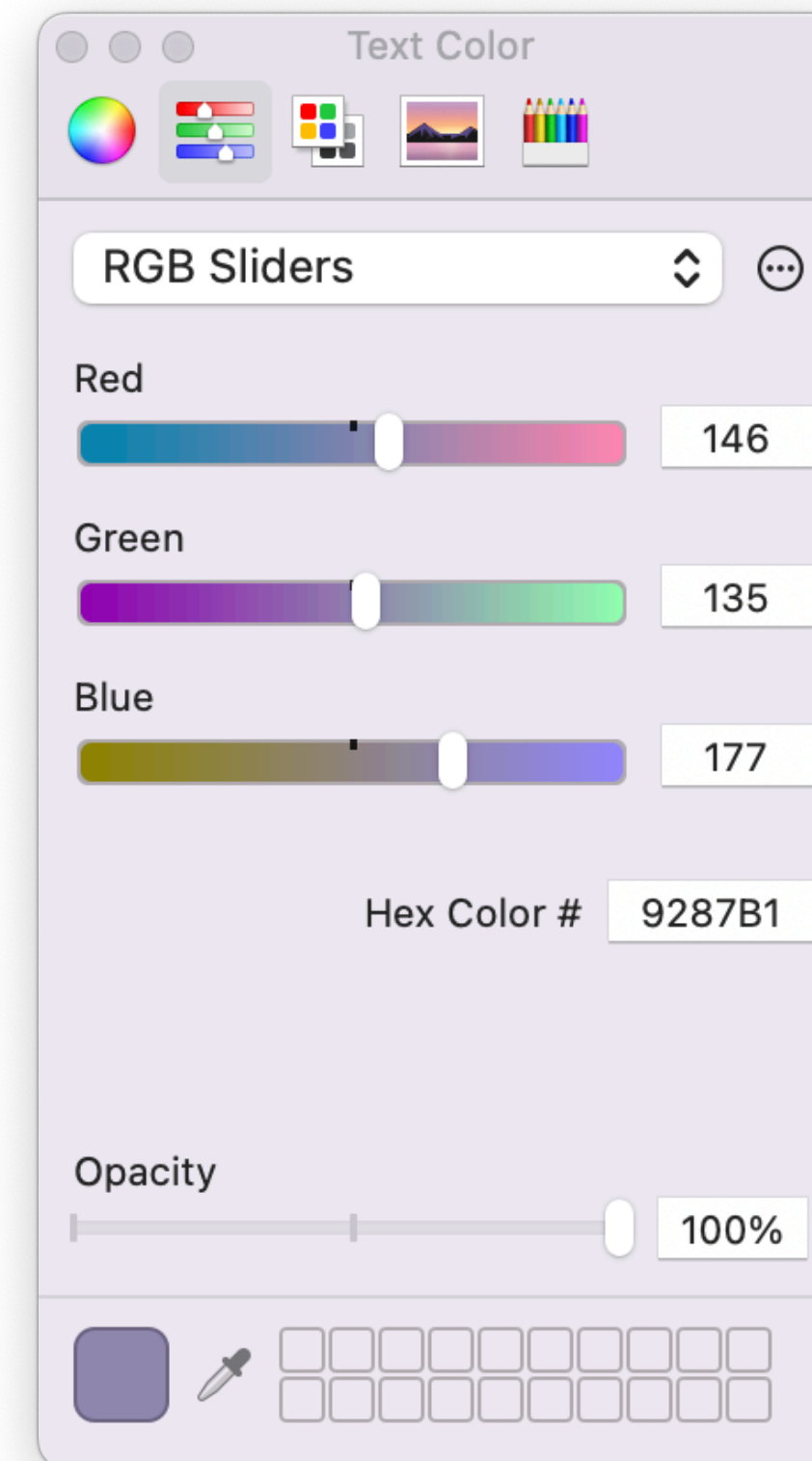


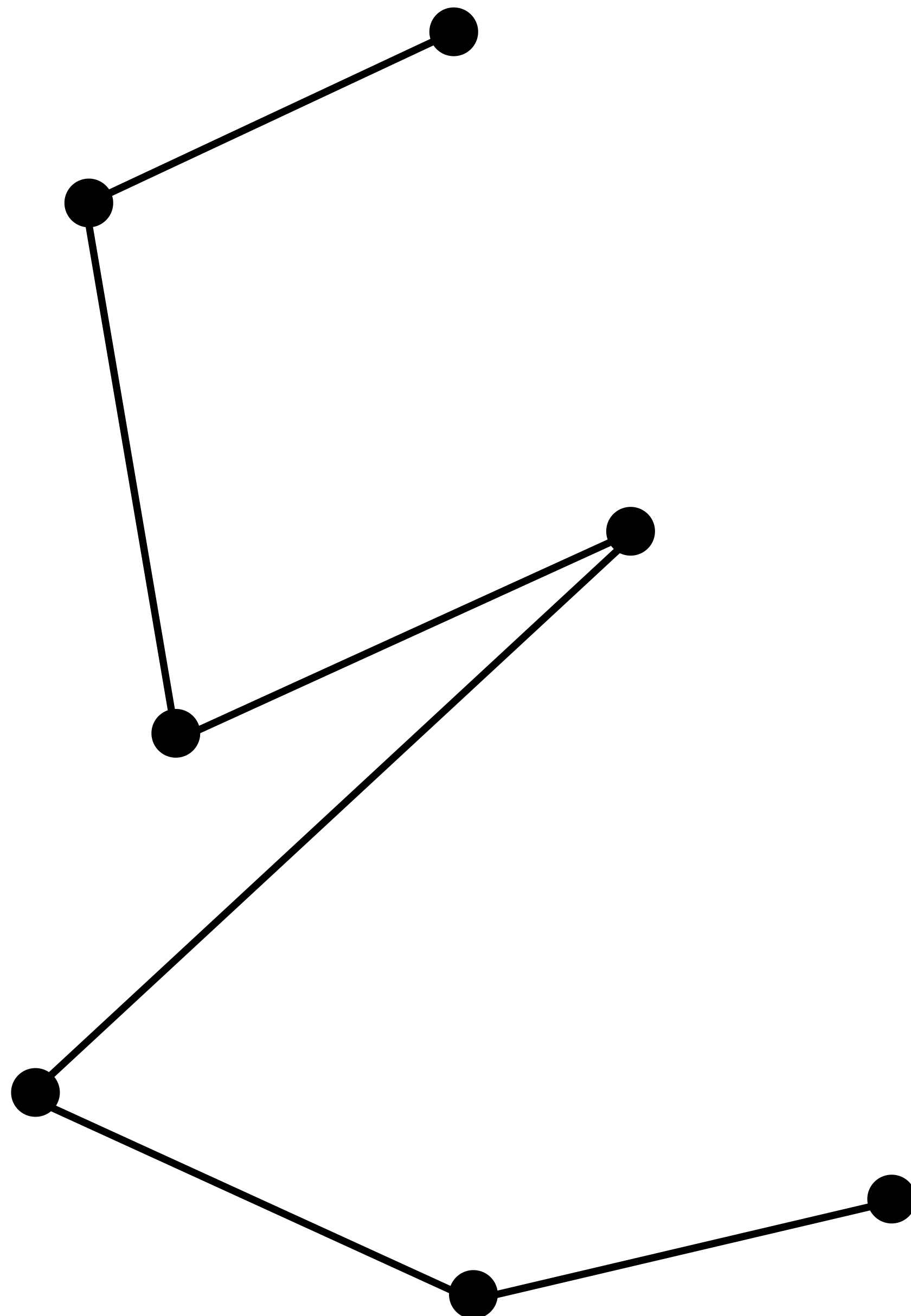


```

colors ← [ ]
f ← ...      ▶ Choose a constant frequency for cycling colors.
ϕr, ϕg, ϕb ← ...  ▶ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ← sin(f × index + ϕr)
  green ← sin(f × index + ϕg)
  blue ← sin(f × index + ϕb)
  push(colors, Color(red, blue, green))
end for

```

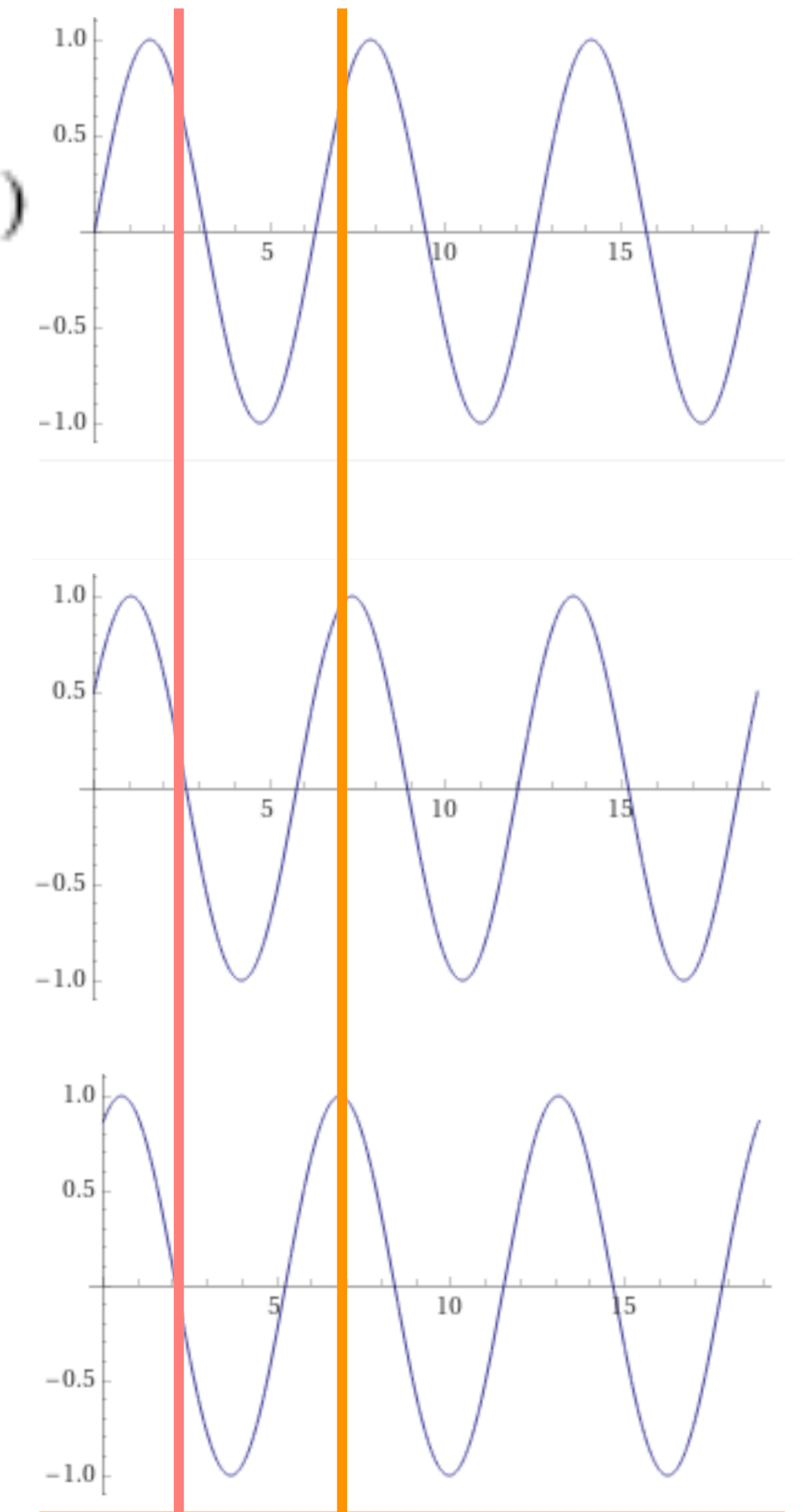
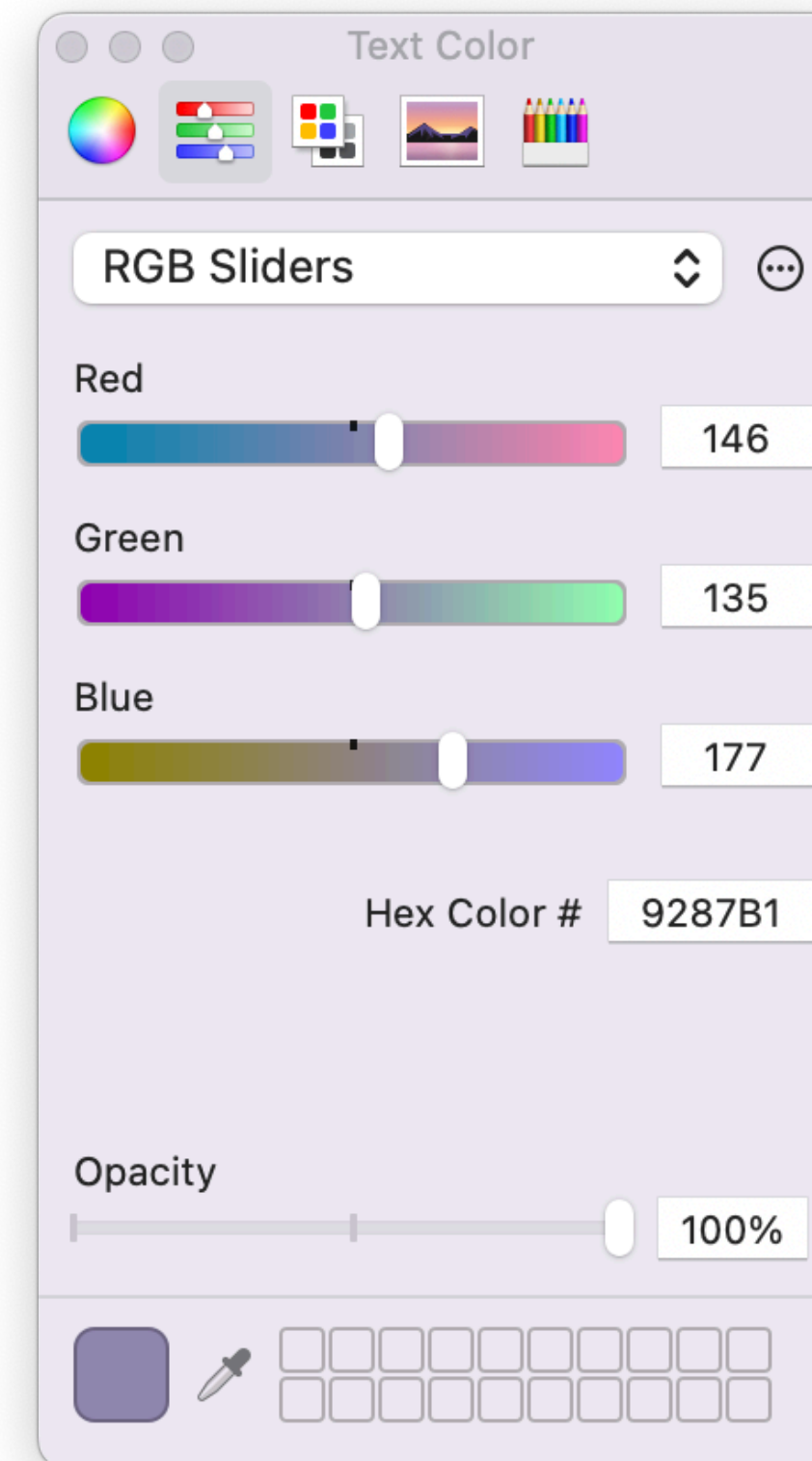


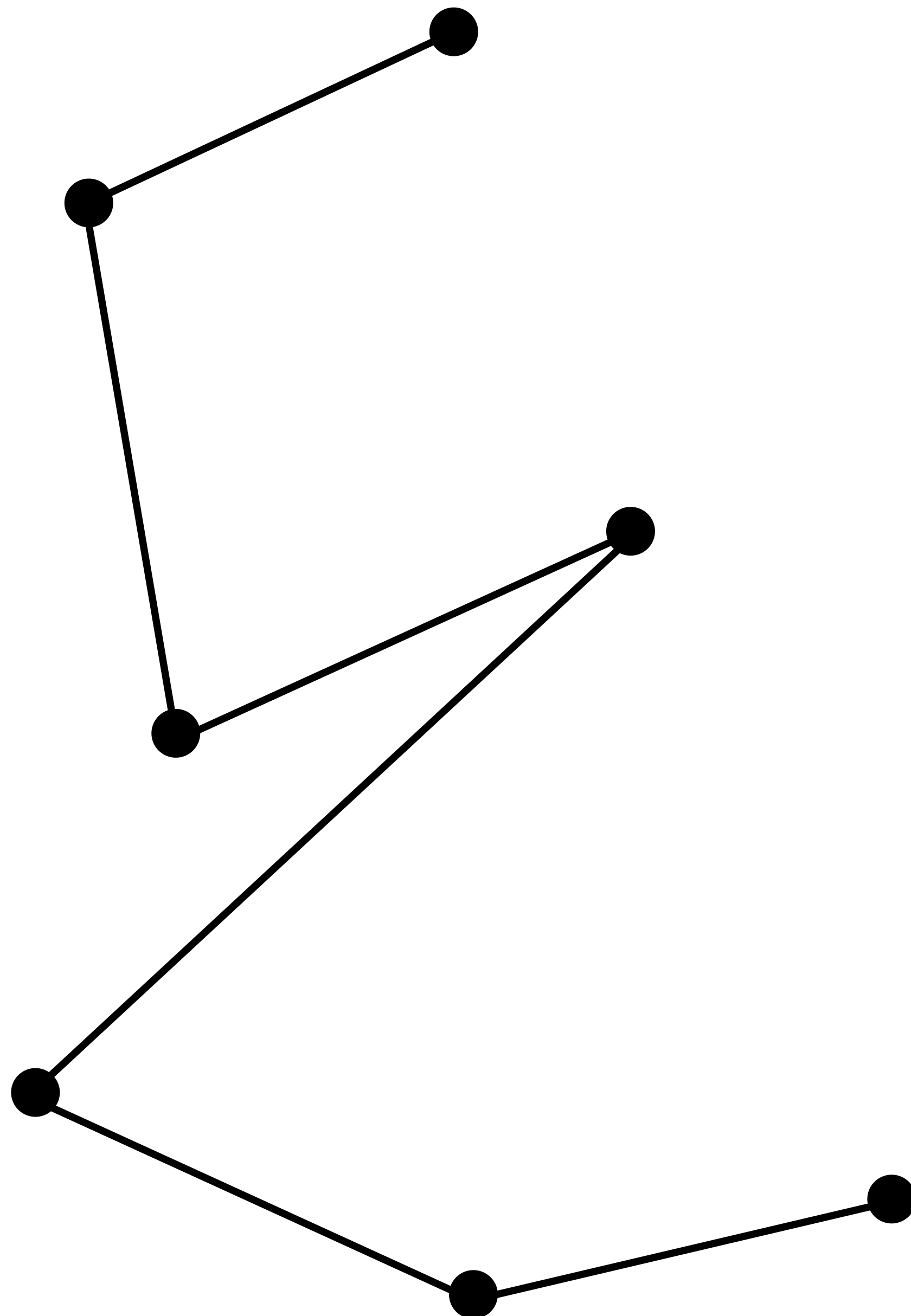


```

colors ← [ ]
f ← ...      ▶ Choose a constant frequency for cycling colors.
 $\phi_r, \phi_g, \phi_b \leftarrow \dots$  ▶ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ←  $\sin(f \times \text{index} + \phi_r)$ 
  green ←  $\sin(f \times \text{index} + \phi_g)$ 
  blue ←  $\sin(f \times \text{index} + \phi_b)$ 
  push(colors, Color(red, blue, green))
end for

```

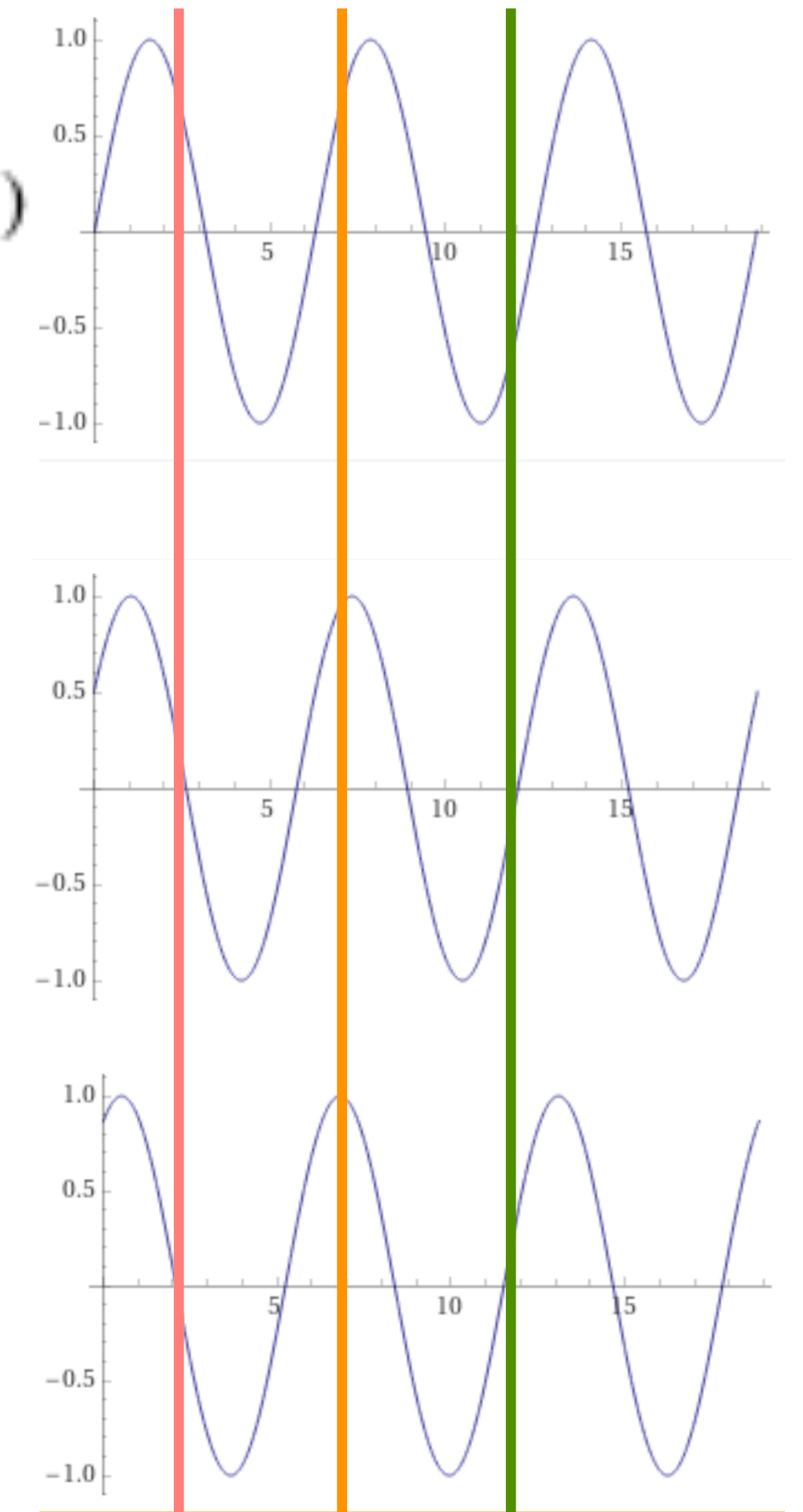
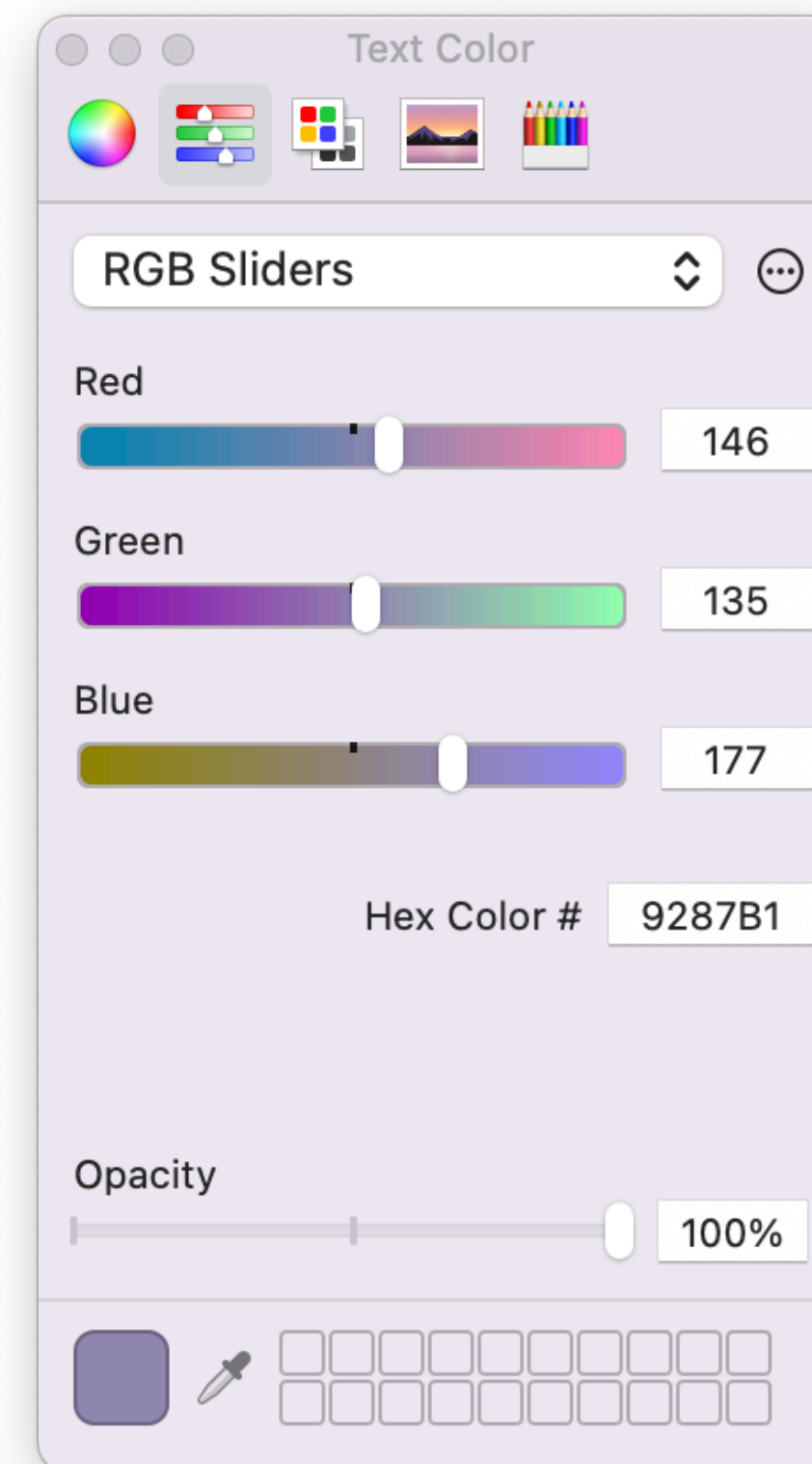


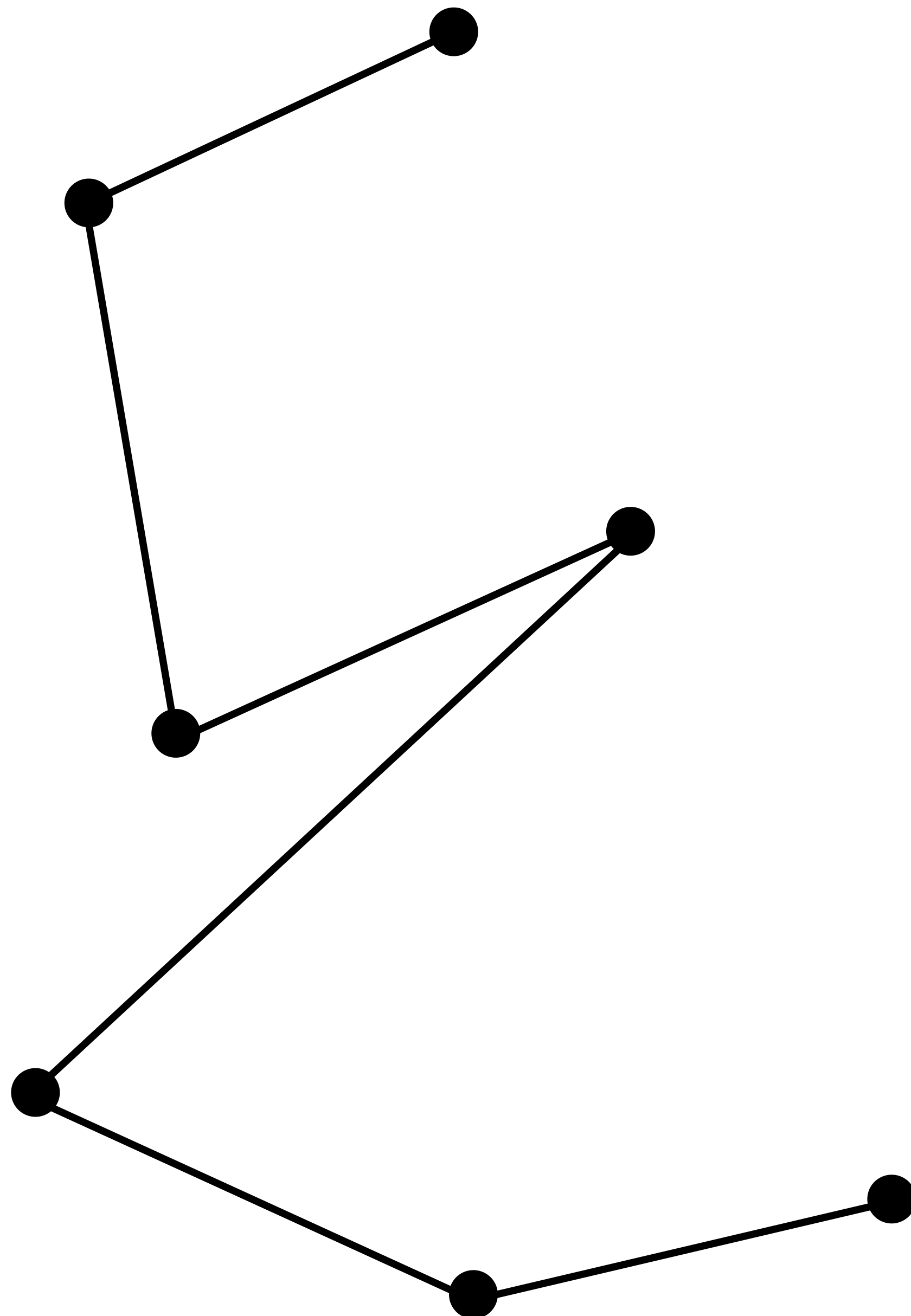


```

colors ← [ ]
f ← ...      ▶ Choose a constant frequency for cycling colors.
 $\phi_r, \phi_g, \phi_b \leftarrow \dots$  ▶ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ←  $\sin(f \times \text{index} + \phi_r)$ 
  green ←  $\sin(f \times \text{index} + \phi_g)$ 
  blue ←  $\sin(f \times \text{index} + \phi_b)$ 
  push(colors, Color(red, blue, green))
end for

```



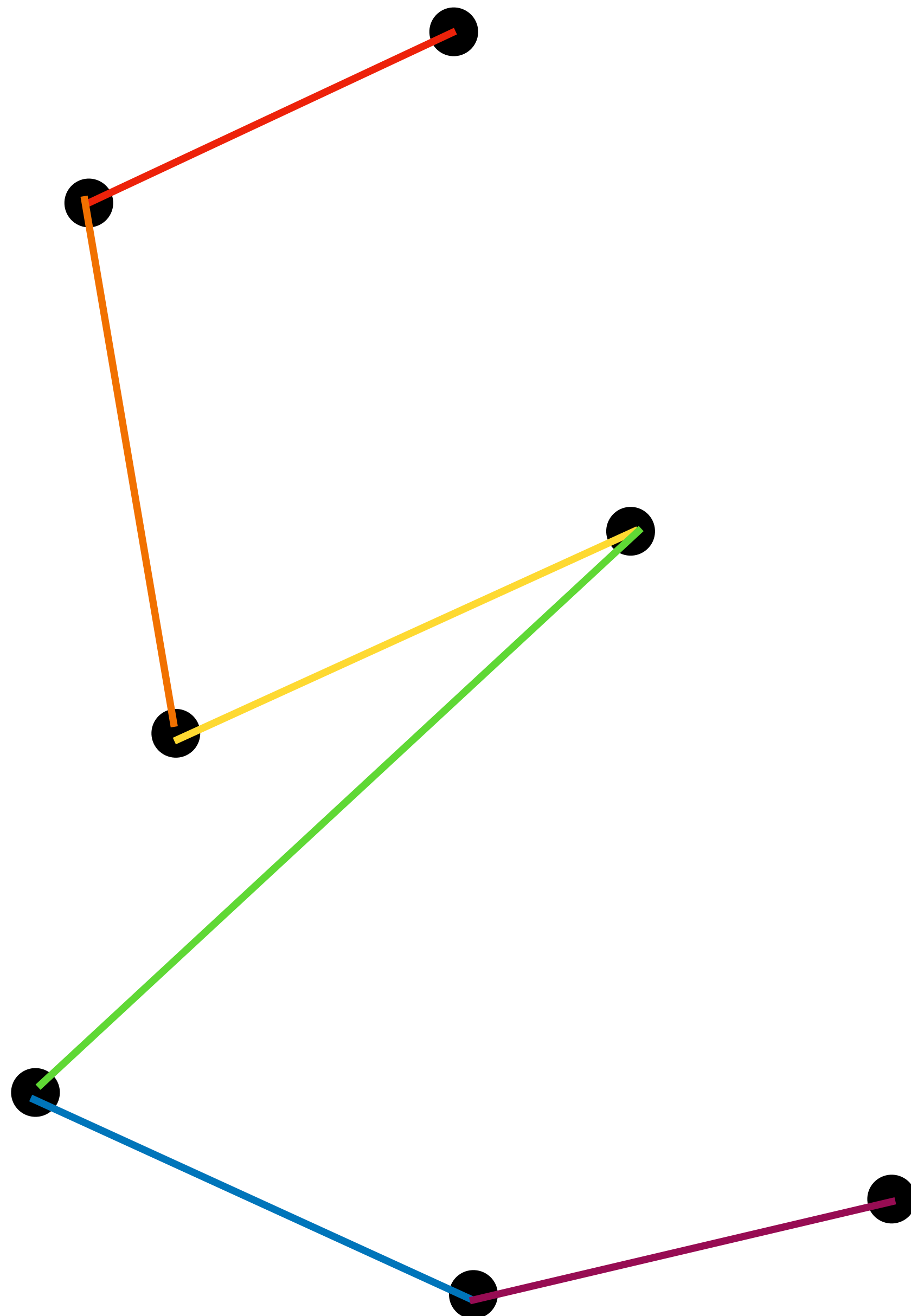


```

colors ← [ ]
f ← ...      ▷ Choose a constant frequency for cycling colors.
 $\phi_r, \phi_g, \phi_b \leftarrow \dots$   ▷ Choose a constant phase offset per channel.
for (_, index) in curve do
  red ←  $\sin(f \times \text{index} + \phi_r)$ 
  green ←  $\sin(f \times \text{index} + \phi_g)$ 
  blue ←  $\sin(f \times \text{index} + \phi_b)$ 
  push(colors, Color(red, blue, green))
end for

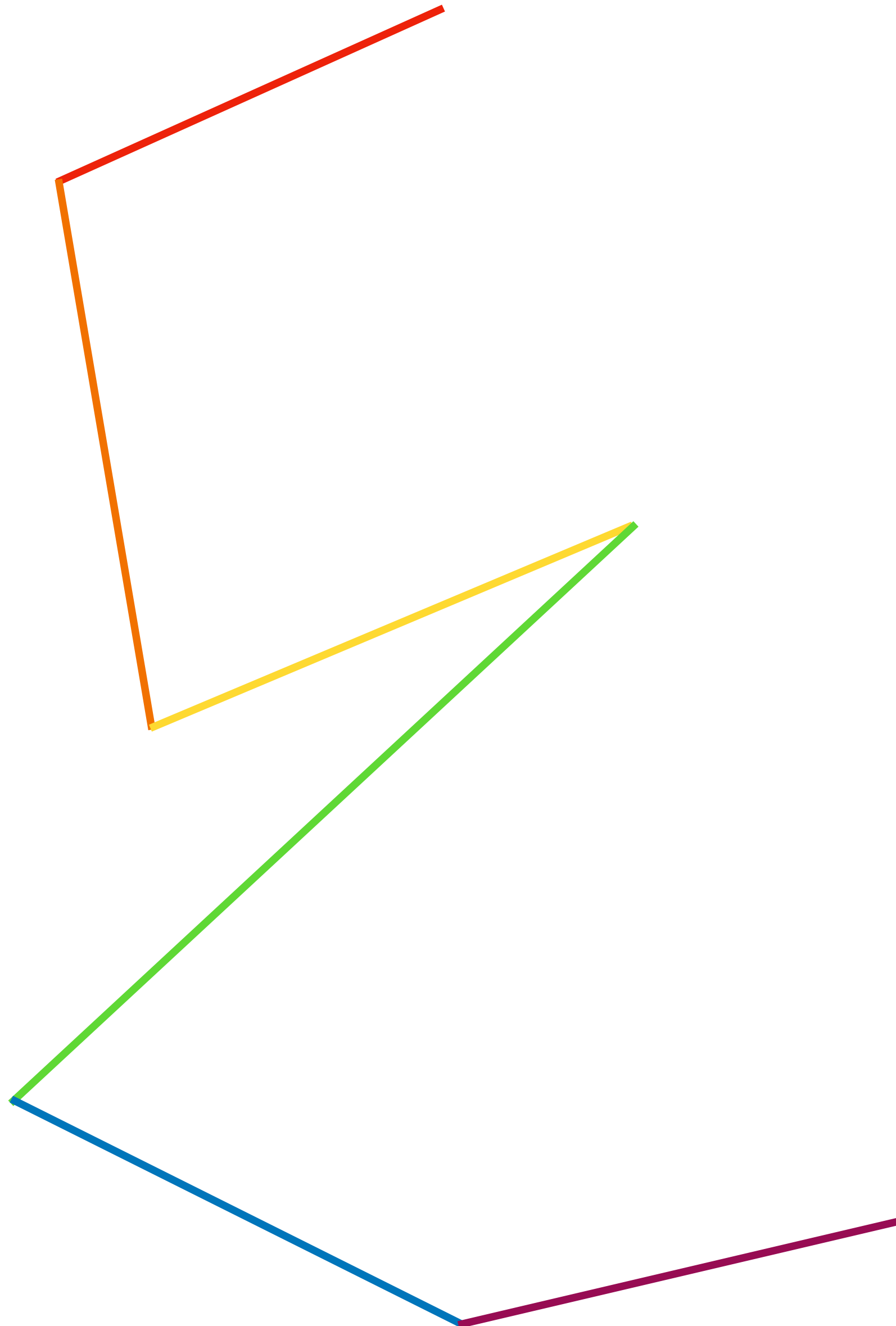
```





```
meshes  $\leftarrow$  [ ]  
for (segment, index) in curves do  
    color  $\leftarrow$  colors[index]  
    mesh  $\leftarrow$  Mesh(curve, color)  
    push(meshes, mesh)  
end for  
return meshes
```





```
meshes  $\leftarrow$  [ ]  
for (segment, index) in curves do  
    color  $\leftarrow$  colors[index]  
    mesh  $\leftarrow$  Mesh(curve, color)  
    push(meshes, mesh)  
end for  
return meshes
```

