

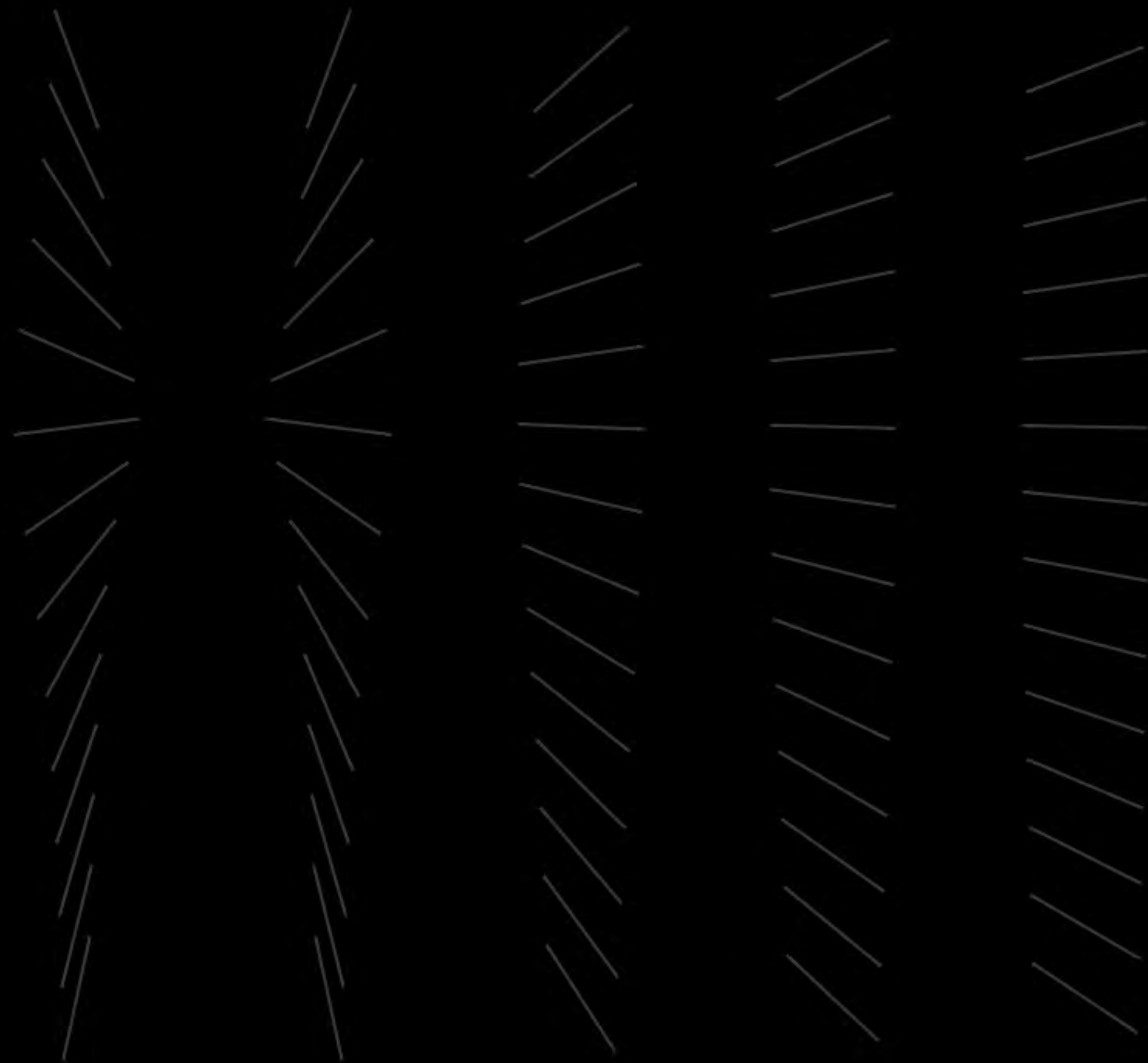
The background is a solid black field. It is populated with numerous short, thin white line segments. These segments are arranged in a somewhat regular grid-like pattern, with three columns of three lines each on the left side, and two columns of five lines each on the right side. The lines are slightly angled, giving a sense of depth or perspective.

creative coding & generative art

with Matt DesLauriers

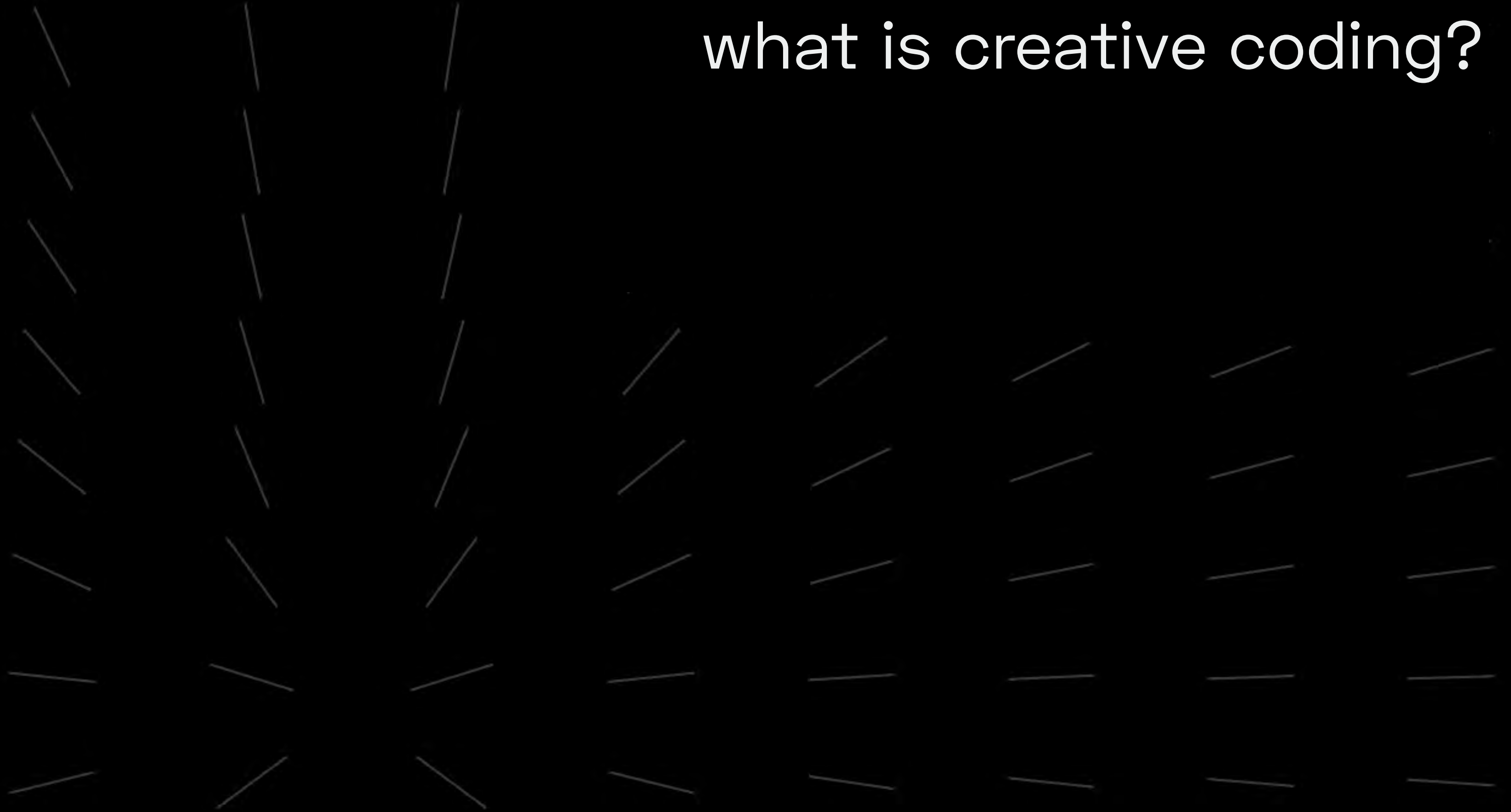
about the
workshop





definitions

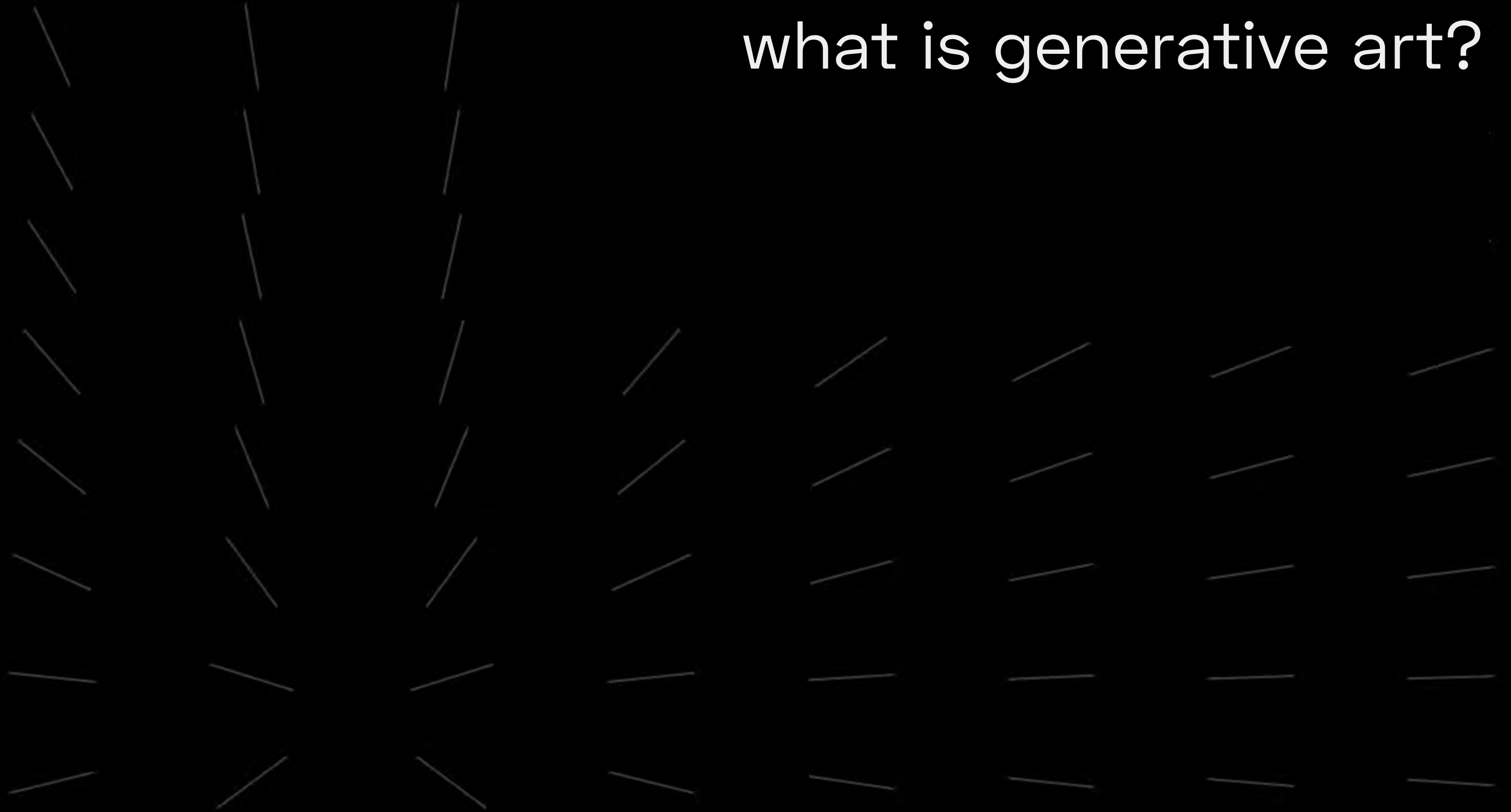
what is creative coding?



what is creative coding?

using code to create something
expressive, rather than functional

what is generative art?

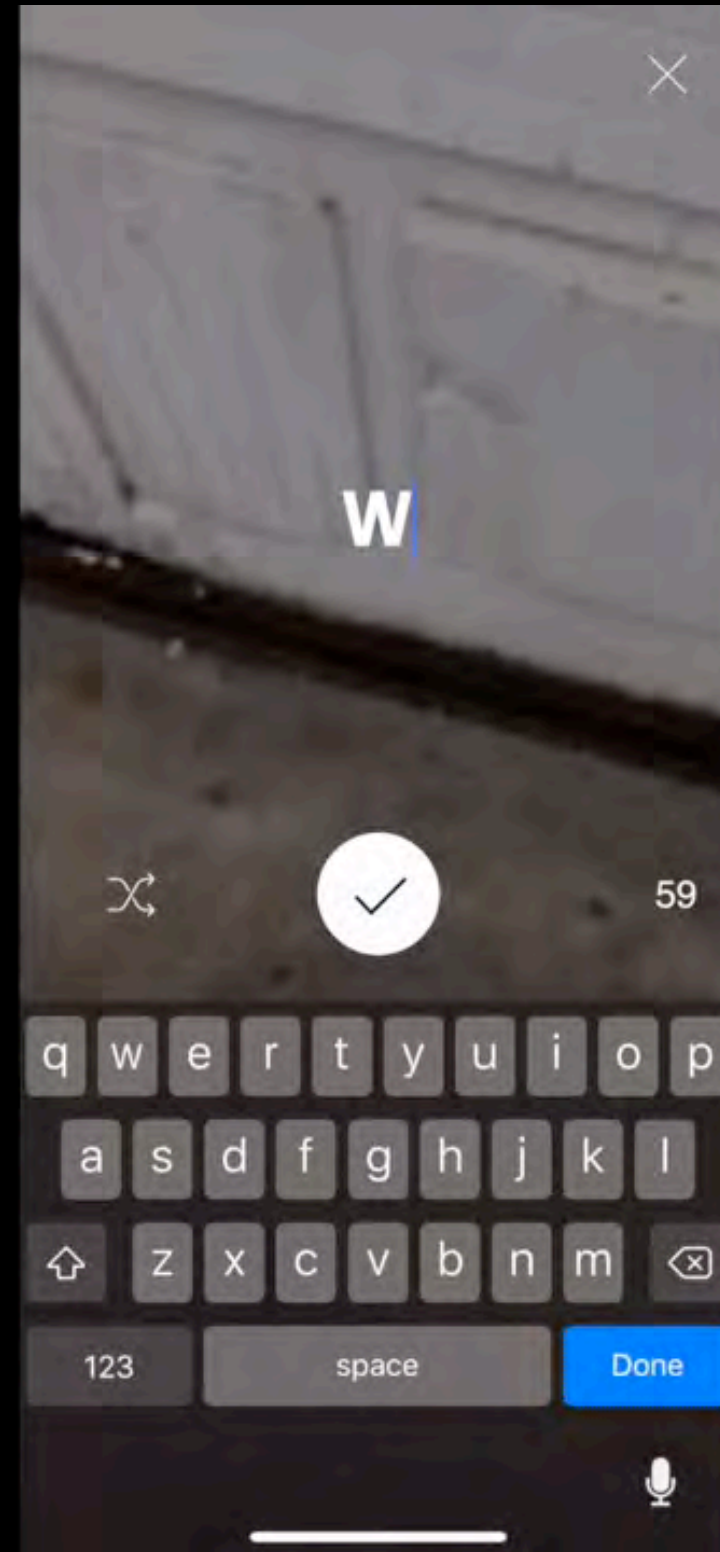


what is generative art?

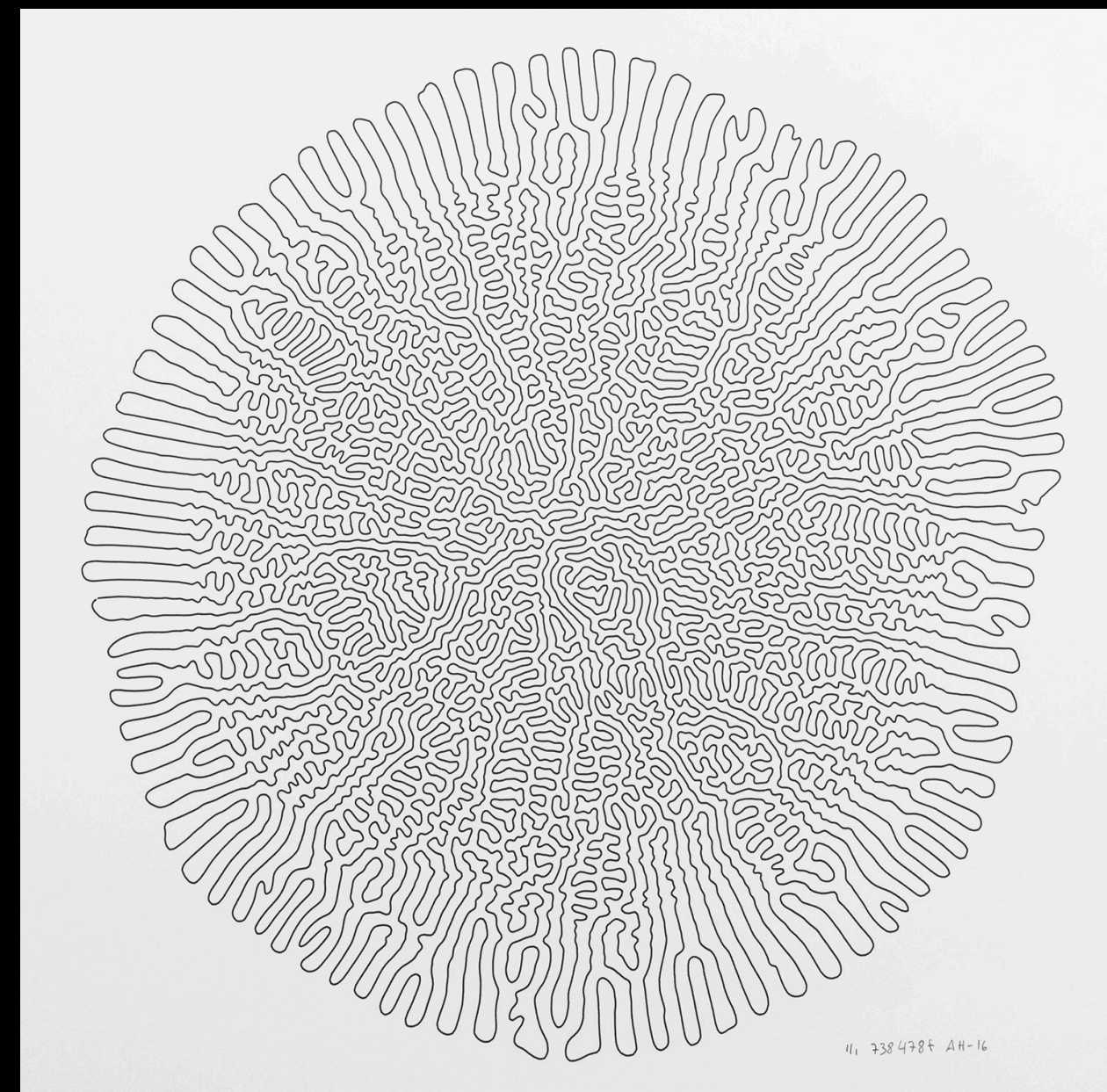
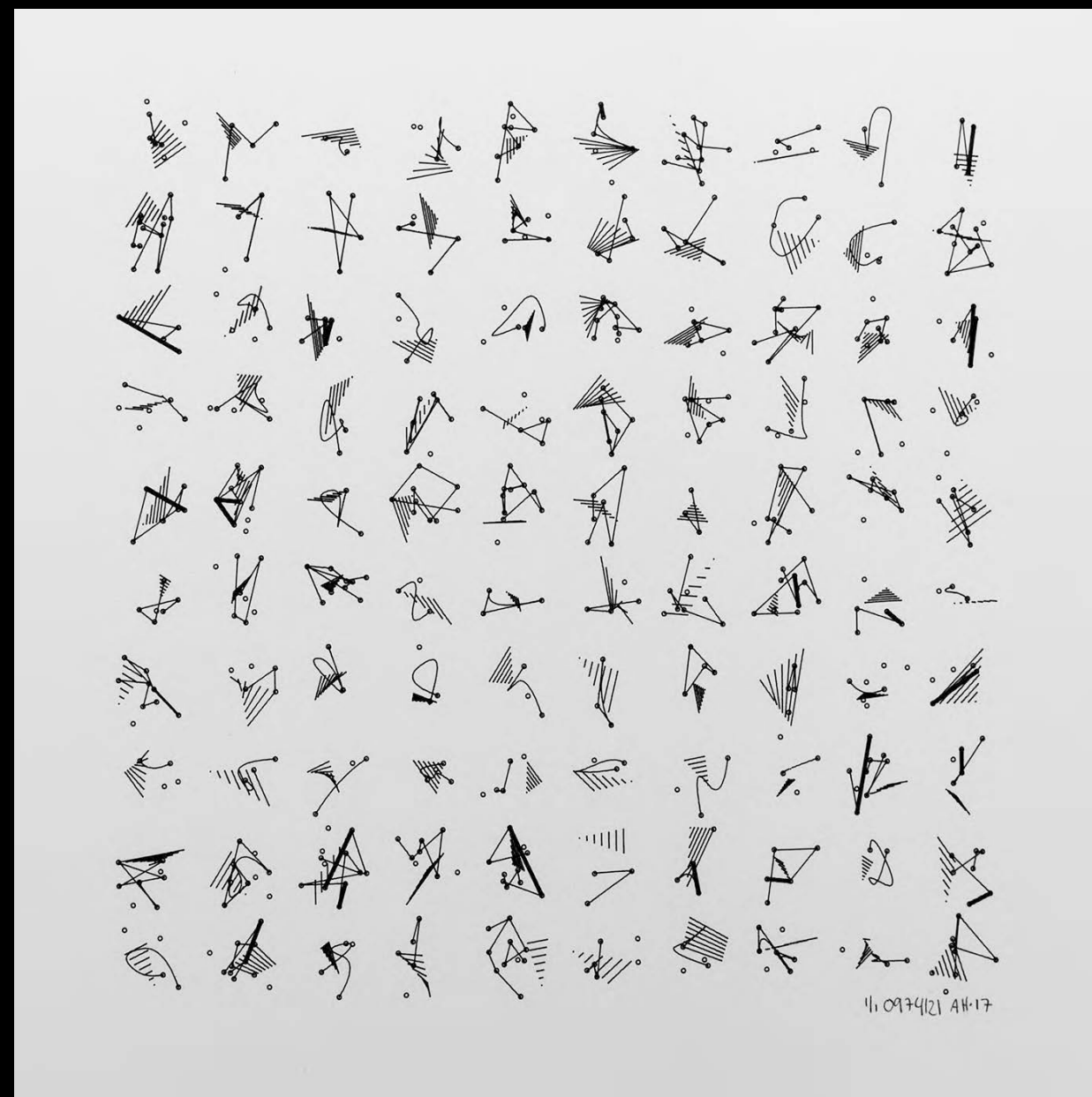
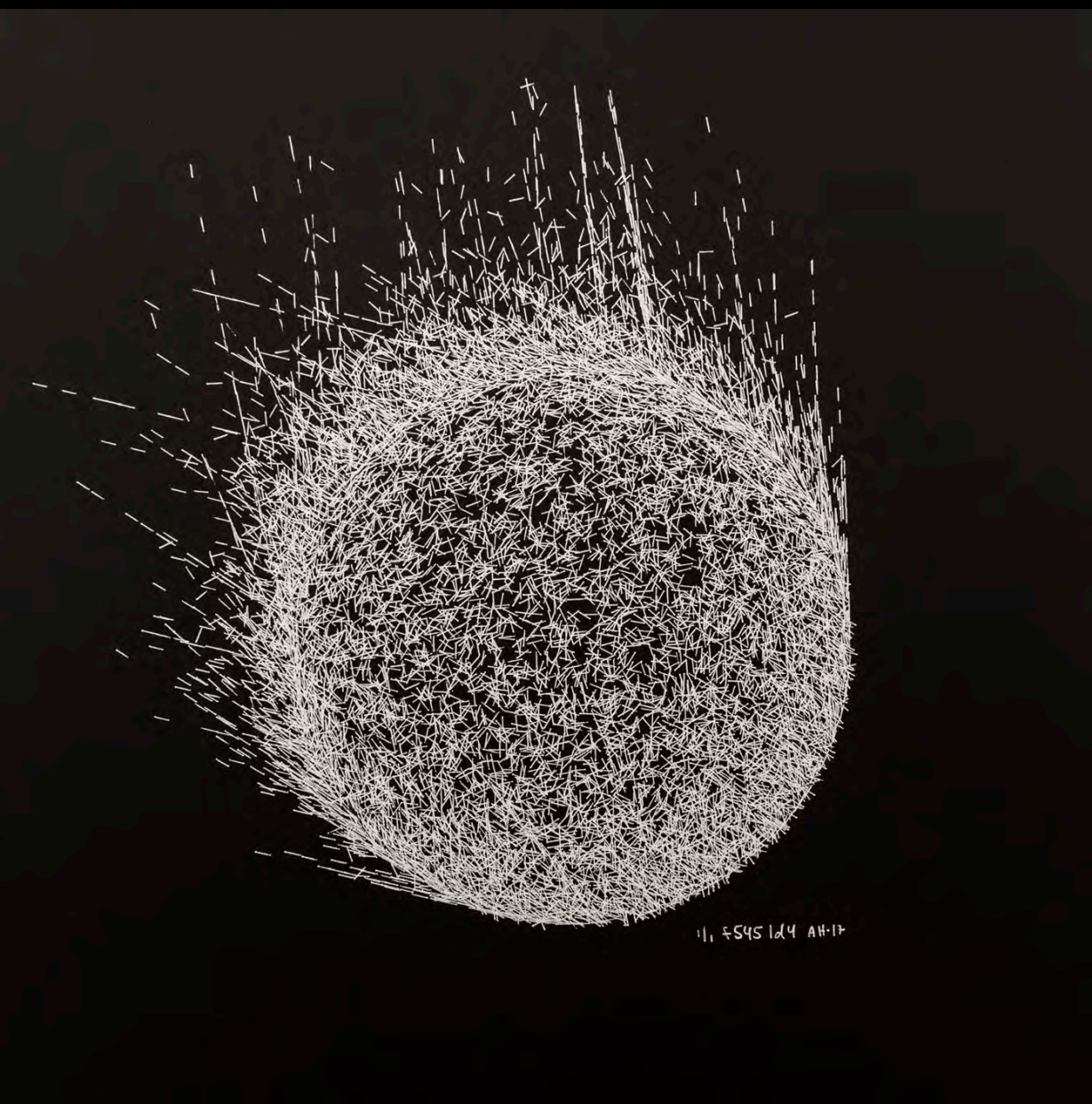
artworks created in part by an
autonomous or rule-based system

The image features a black background with a vector field represented by numerous thin, light gray lines. These lines are distributed across the frame, with a higher density on the left side and a more sparse distribution on the right. The lines vary in orientation, generally pointing from left to right, but with some local variations in direction, particularly in the upper and lower regions. In the center of the image, the text "in the wild" is written in a clean, white, sans-serif font.

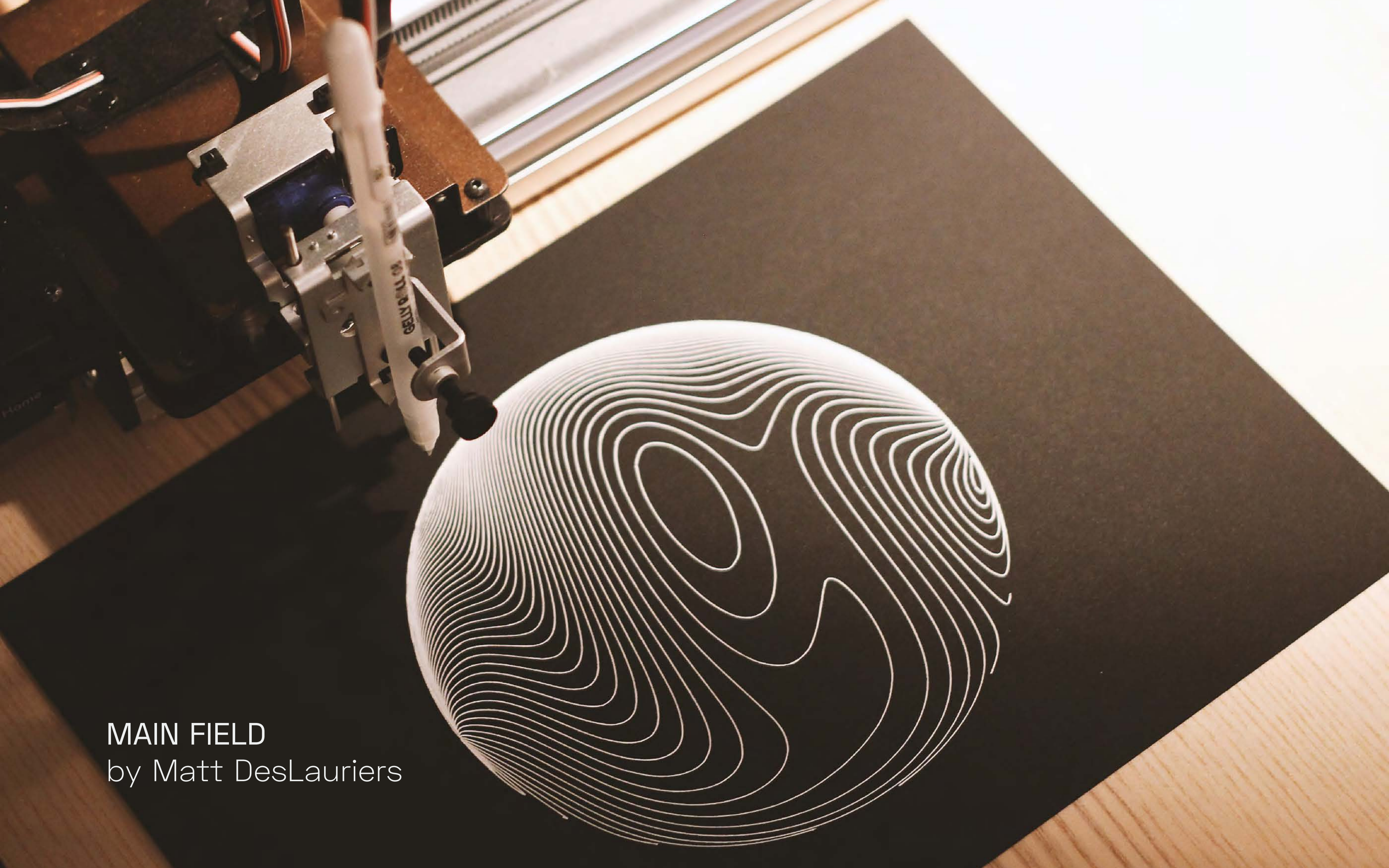
in the wild



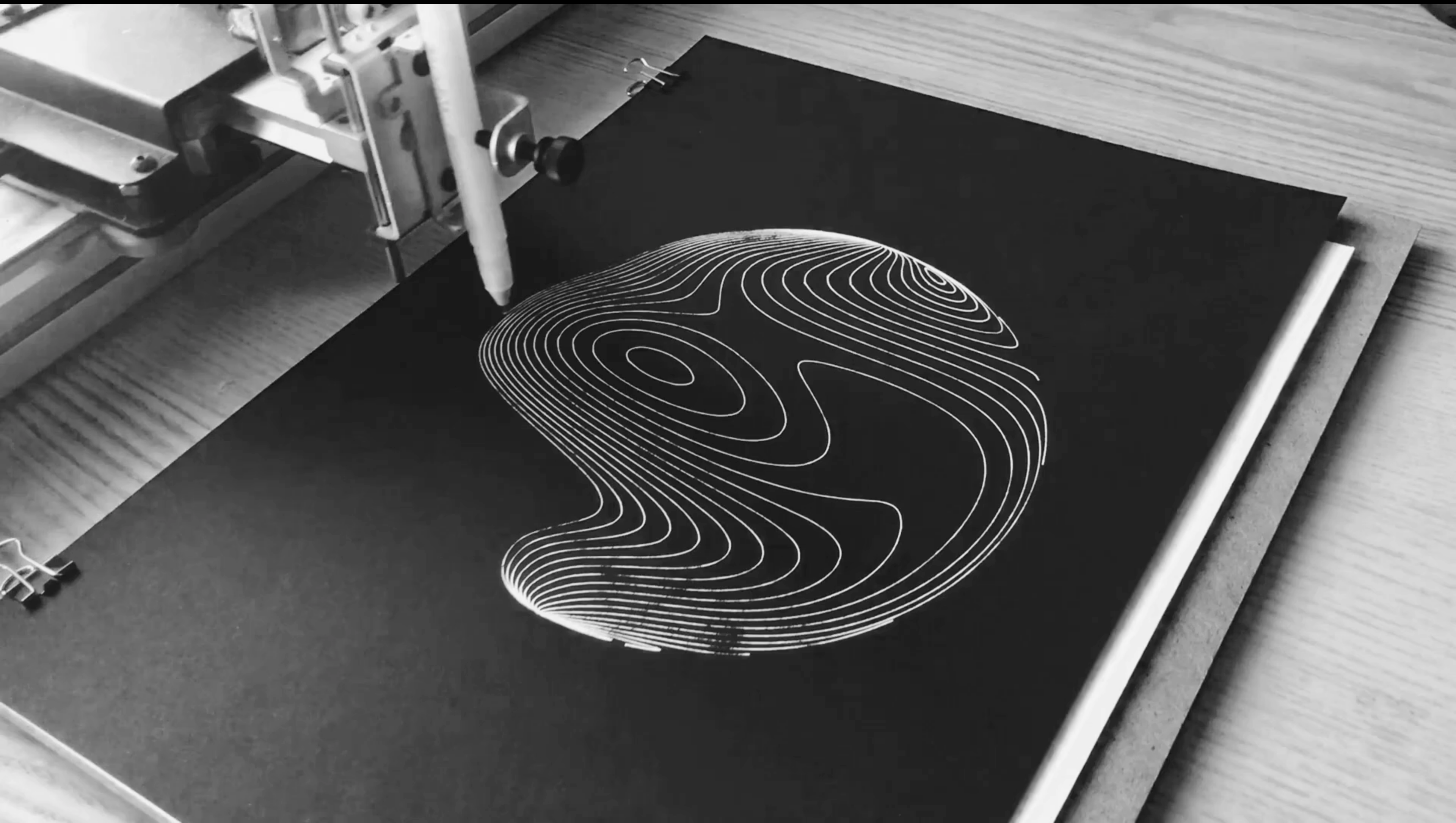
Weird Type
by Zach Lieberman



@inconvergent
Anders Hoff



MAIN FIELD
by Matt DesLauriers





SUN
by Philip Schütte x Random Studio



Paper Planes
by Active Theory




Generative Puzzles, Housewares and other Products
Nervous System



let's make art!



Unicode Compositions

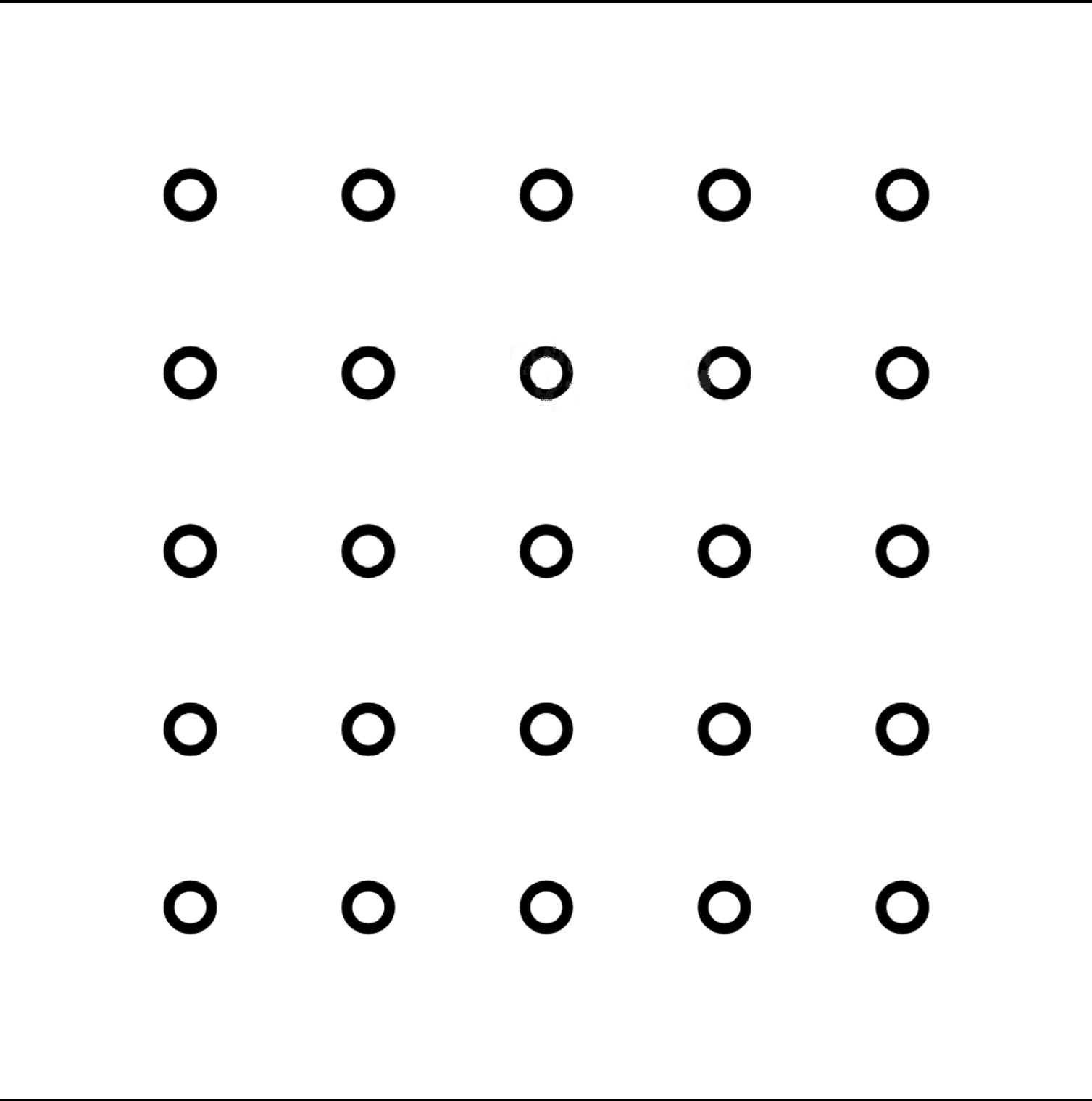


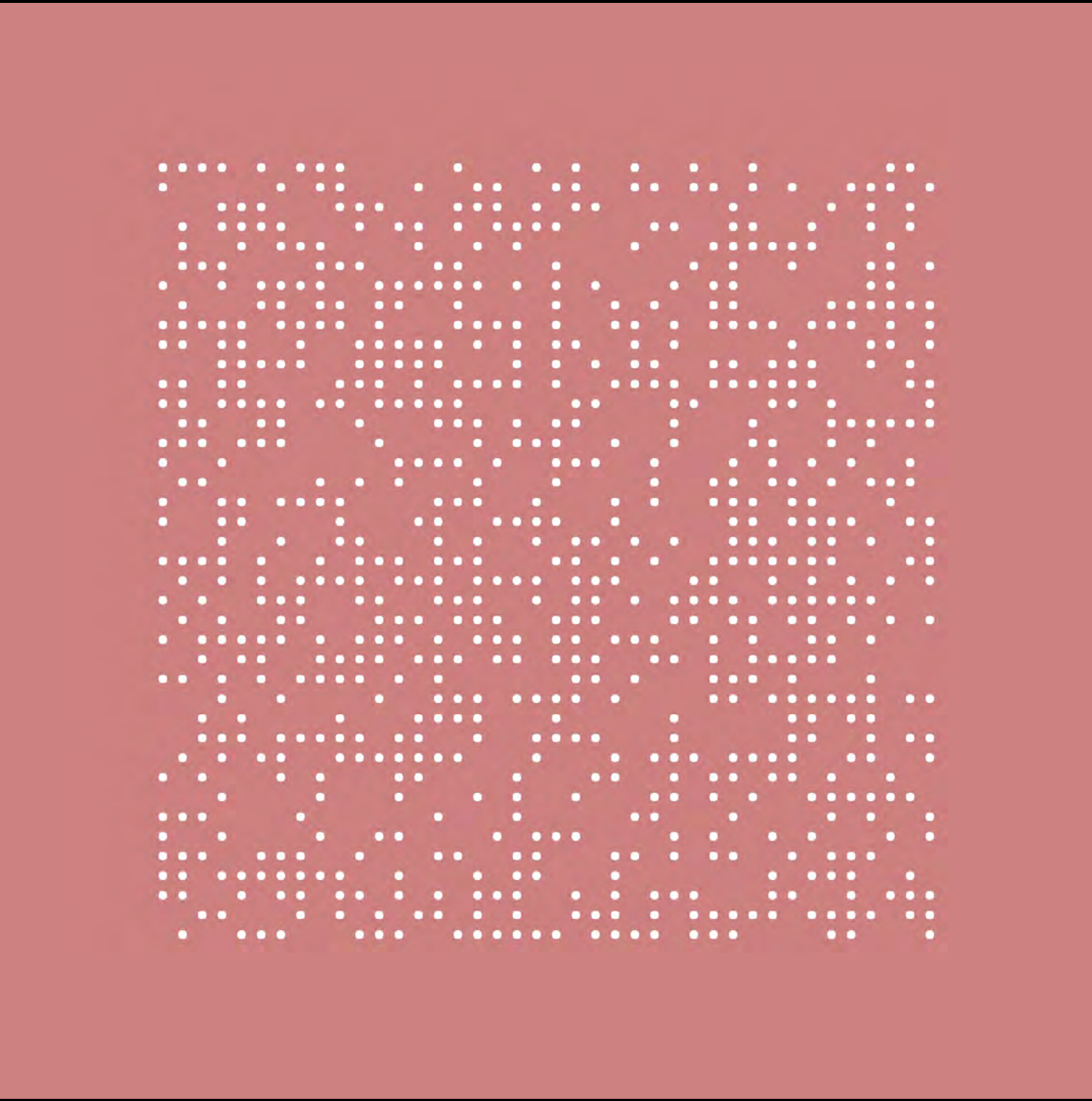
tools & setup

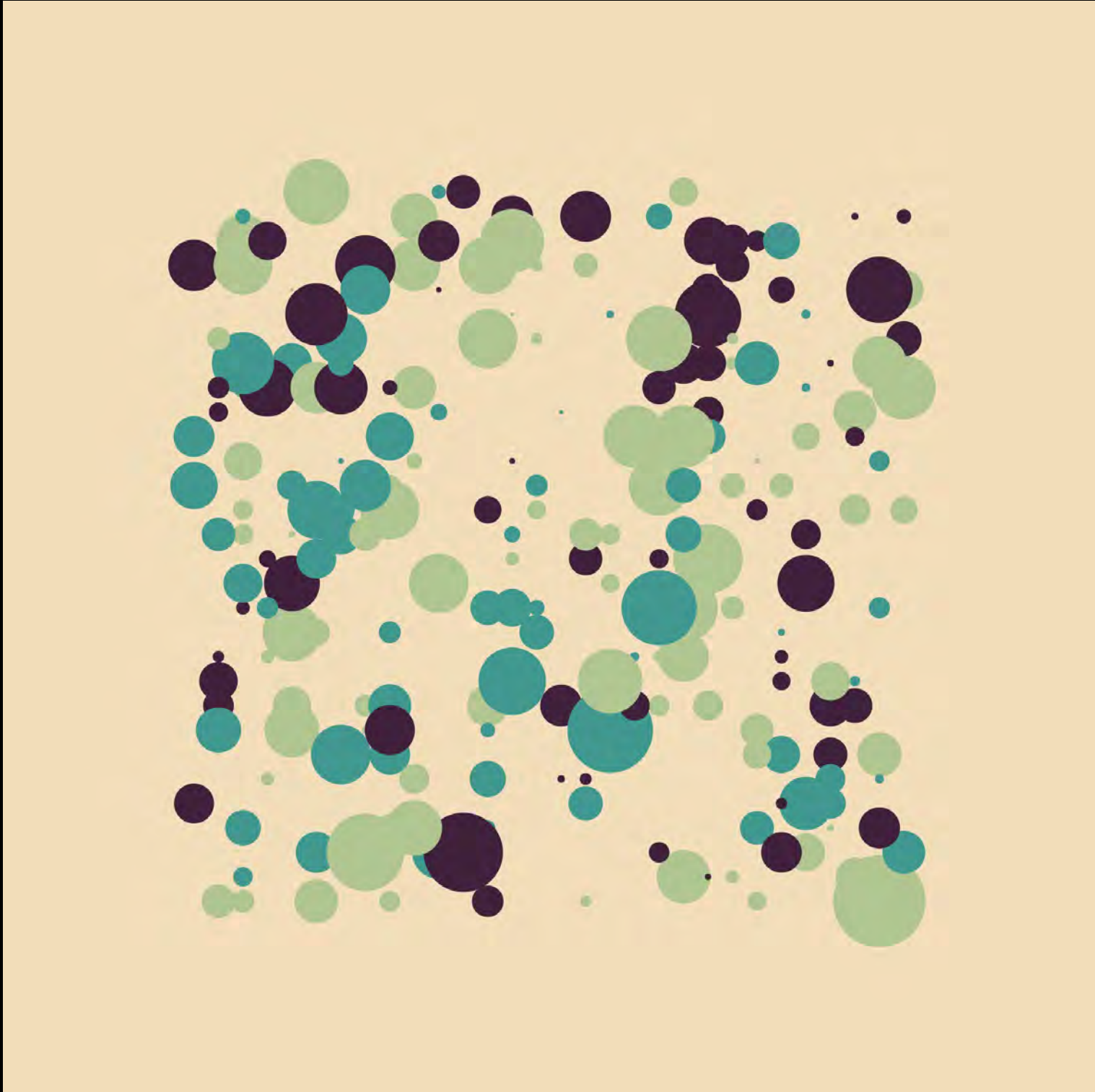
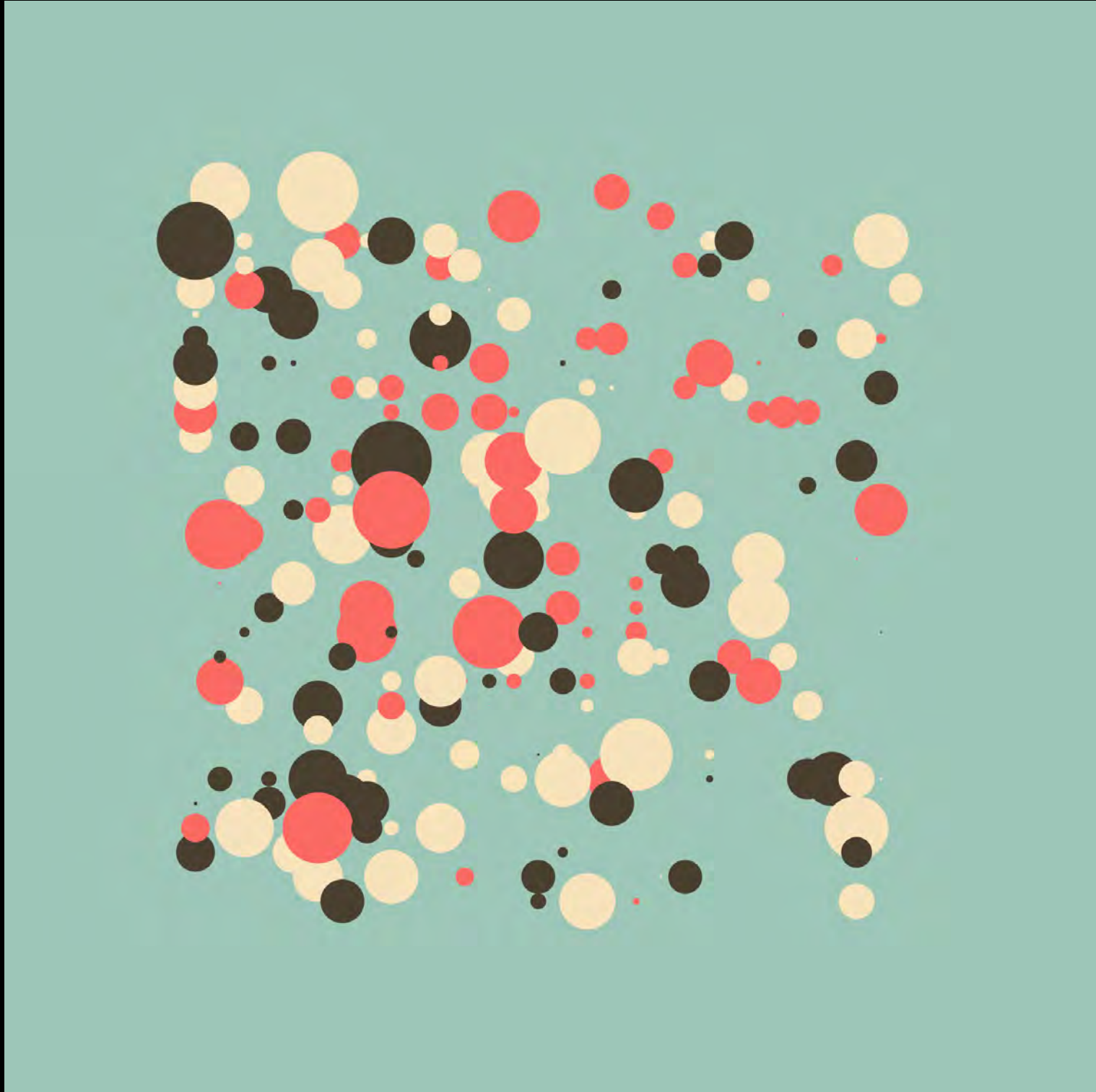


<https://github.com/mattdesl/workshop-generative-art>

the grid







B A ↓ → A A ← → ←
↓ → B ← ↑ ↓ B
↓ ↑ A A ↑ ↓ → A B ← ↑
↑ ↓ ← ← → ↑ ↓ ↑ A ↓ B
A ↑ ← B → → → ↓
↓ ← A ↓ ↑ ← ← ← ↑ ← ↓ →
↓ B ← → ↑ → A B A → B A B
↑ ← ← A ↑ ↑ ↑ ↑ A A ↓ →
↑ ↑ ↑ B ← ← B ← →
↓ A → → → ← ↓ B A → ↑ → ↑ ←
↓ → ↓ A ↓ ↑ → ↑
← ← A ↓ B B B ← A B
↑ → → A → ← → ↑ → → B
↑ → ↑ ↑ ↑ ↓ → ←
← B A ↓ A ↑ ↑ ↓ ↑ B B ↓
B A A ↑ ← →
A B A ↑ ← ↓ ↑ ↓ A B ← ↑ ↑ →
↓ A → B A B ← ↑ → → →
↓ A B ↑ ↑ ← ← → A → ↓
↑ + ↓ A → ← ← B ← B ↓ ↓ ↑ ← A

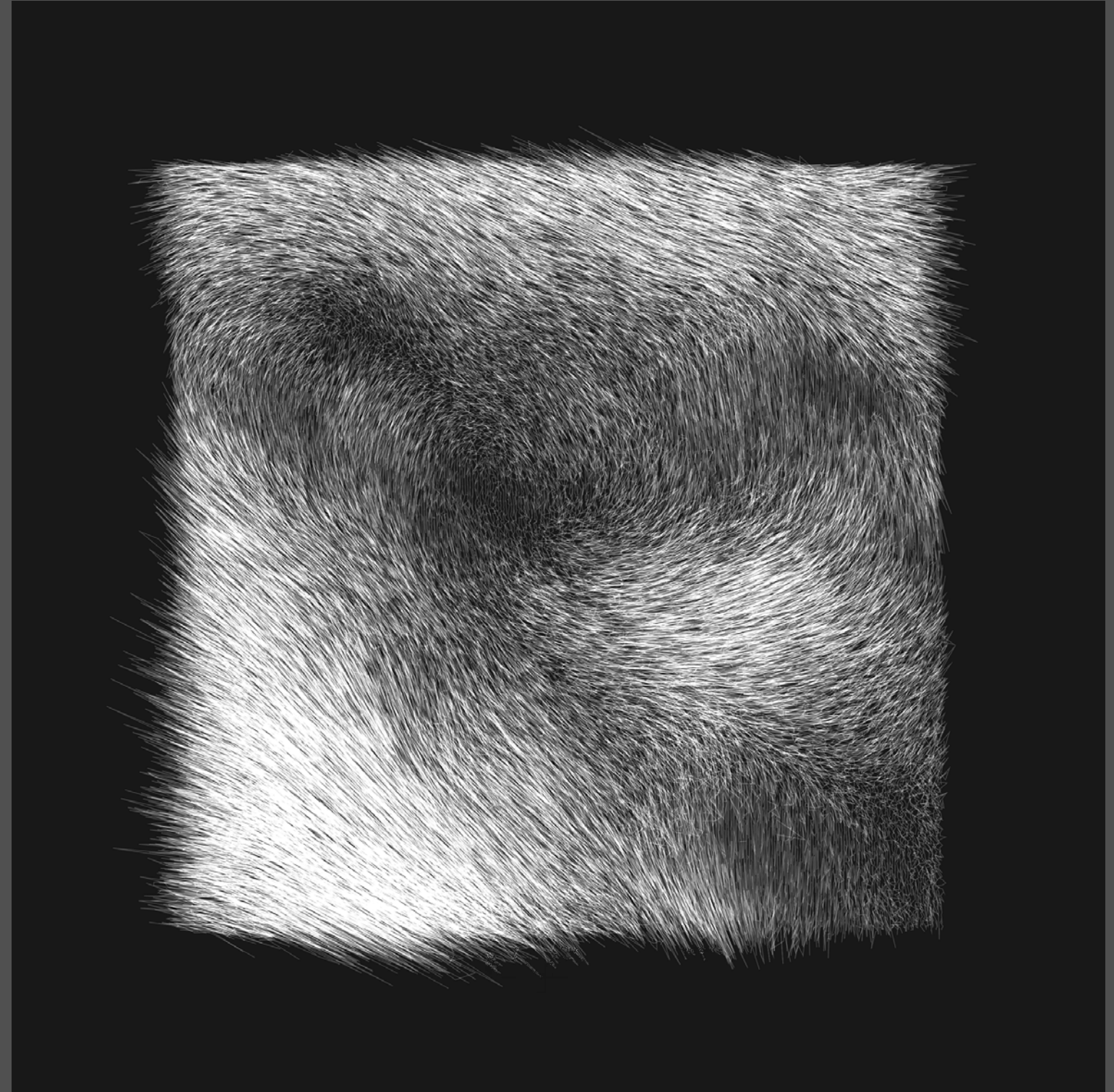
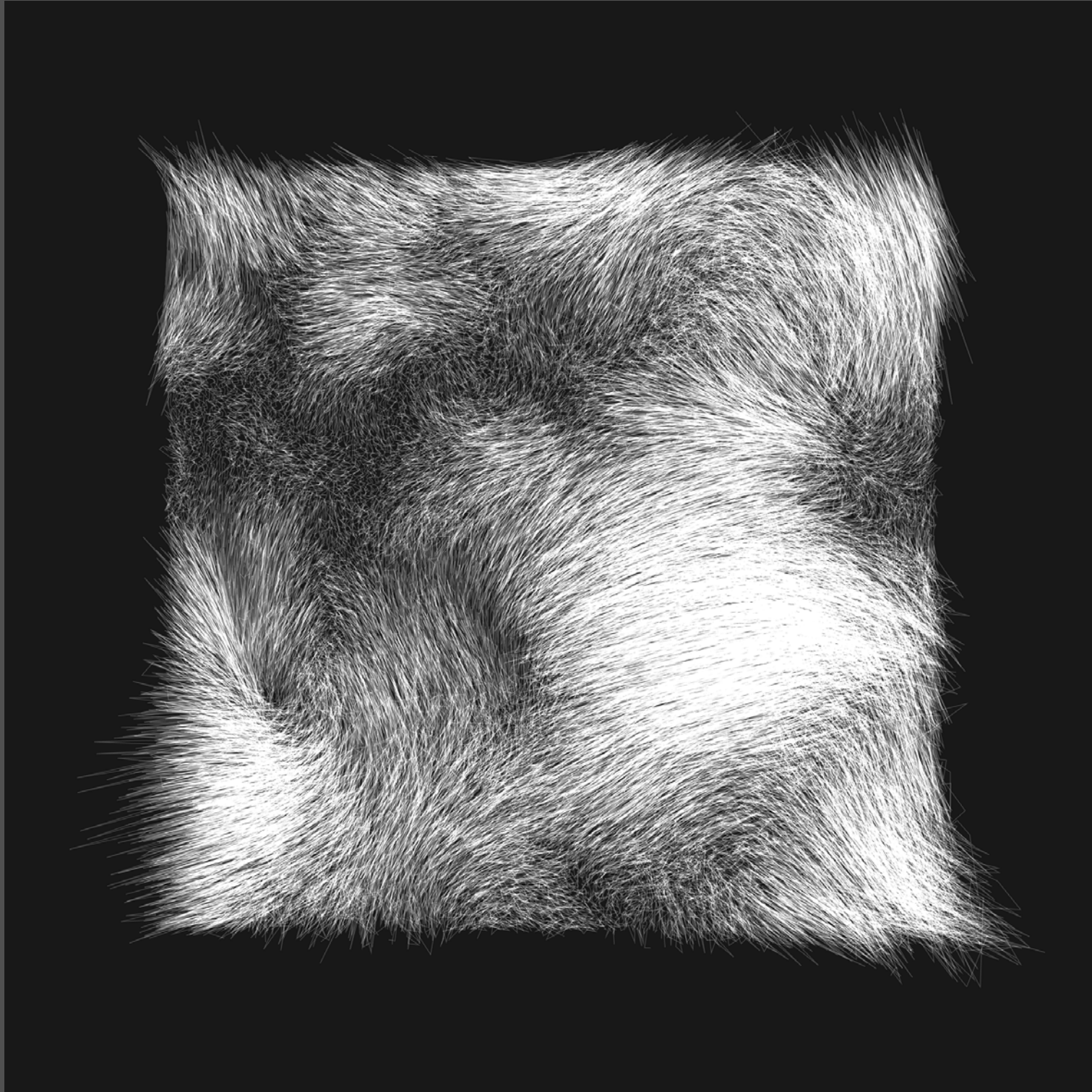
→ ↑ → → → A
→ → A ← ← → B ↓ → A A A
↓ B → ↑ ↓ ↑ → ↑ B B
← ← A ↑ B B B ← A ← B
B → ← → ← ↑ ↓ → B
A ← A ↓ ↑
← → ↓ ← B → A ↓ A ←
→ B ↑ B → → → ↑ →
A ← ← B → B ← ← ← B
A A ← ↓ B → B A
↓ ↓ ↑ B B B ↓ ↑ ← ← ↑ A ↓
↑ ↓ ← → ← A ← B ← A
↑ B ↓ → ↓ → ↓ ← ↑
↓ ↑ A → ← A ← ↑
B ← → ↓ A A ↑ → →
← ↓ B ↑ ↓ ← ↓ ← ←
B ↓ ← ↑ → ↑ ← ← A
↓ ↓ → A ↑ ← ← ↑ A
↑ ↑ B ← ← → ↓ →



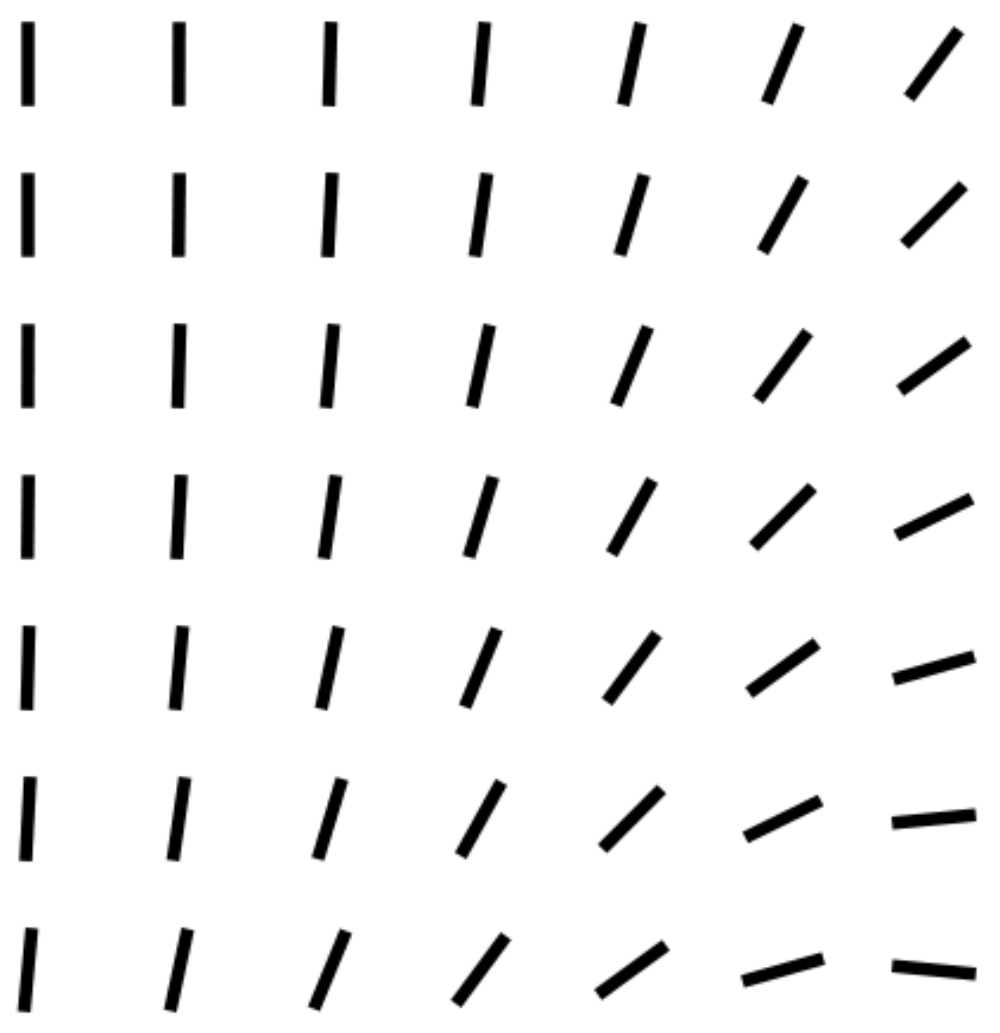
Unicode Compositions

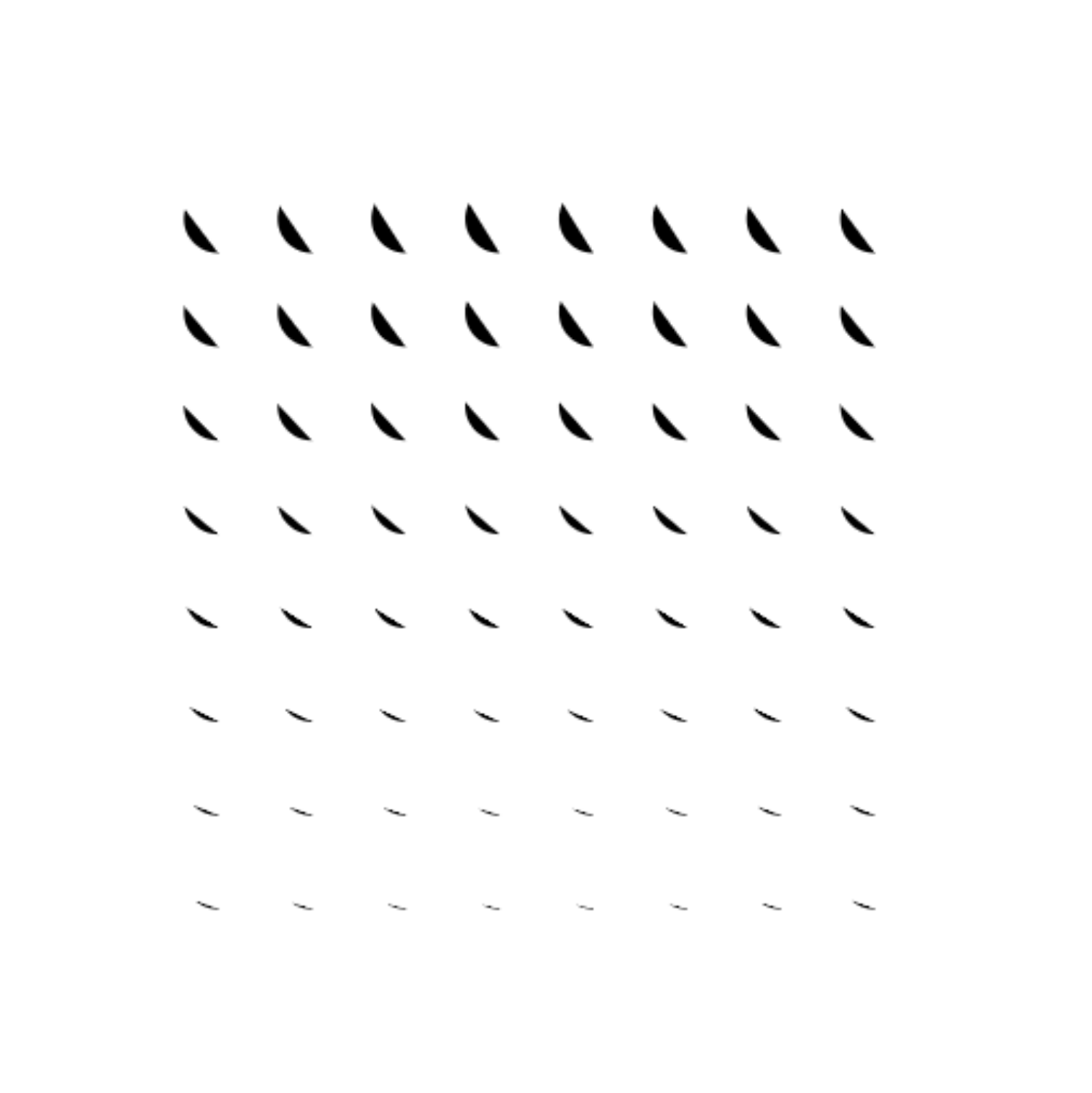


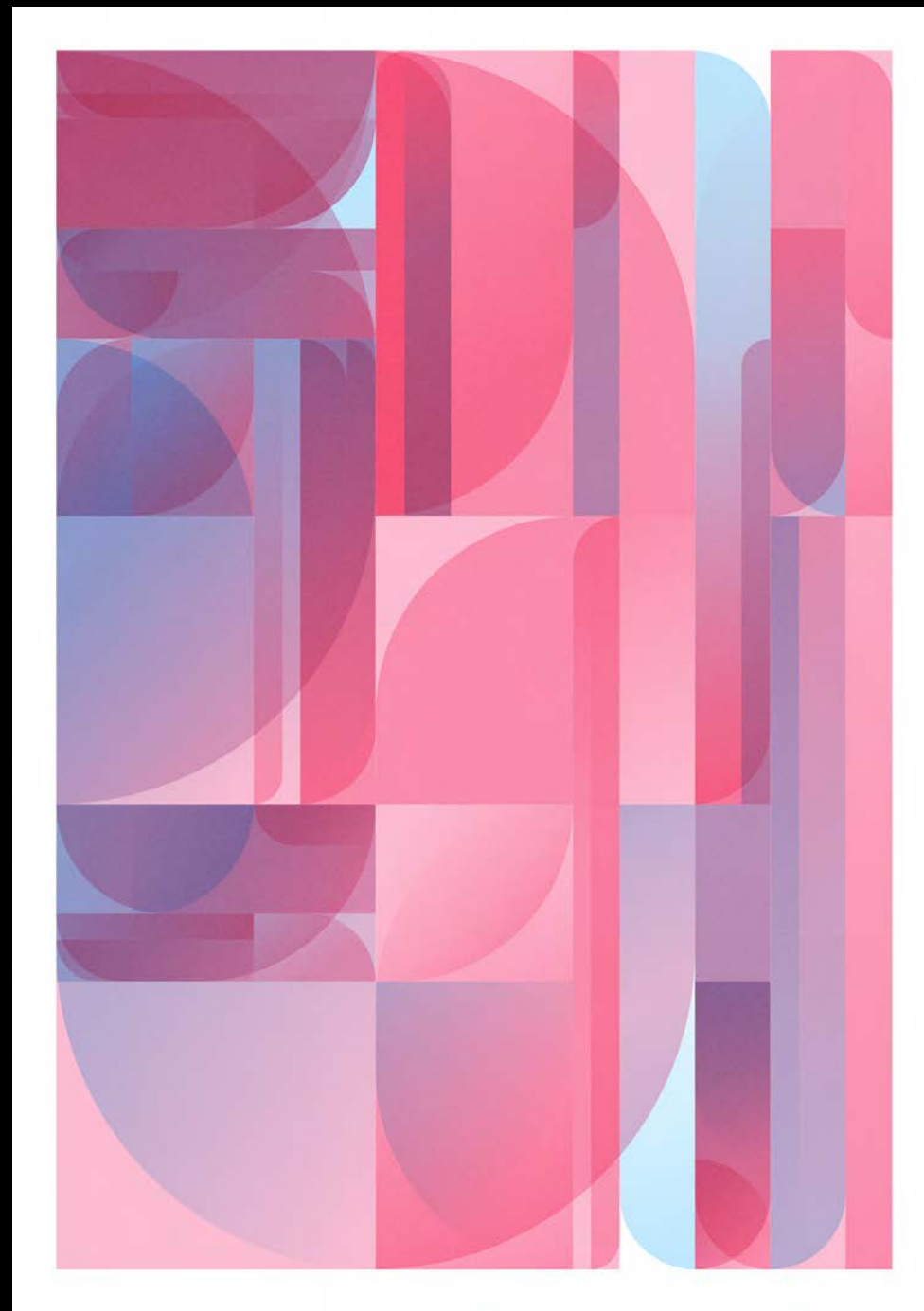
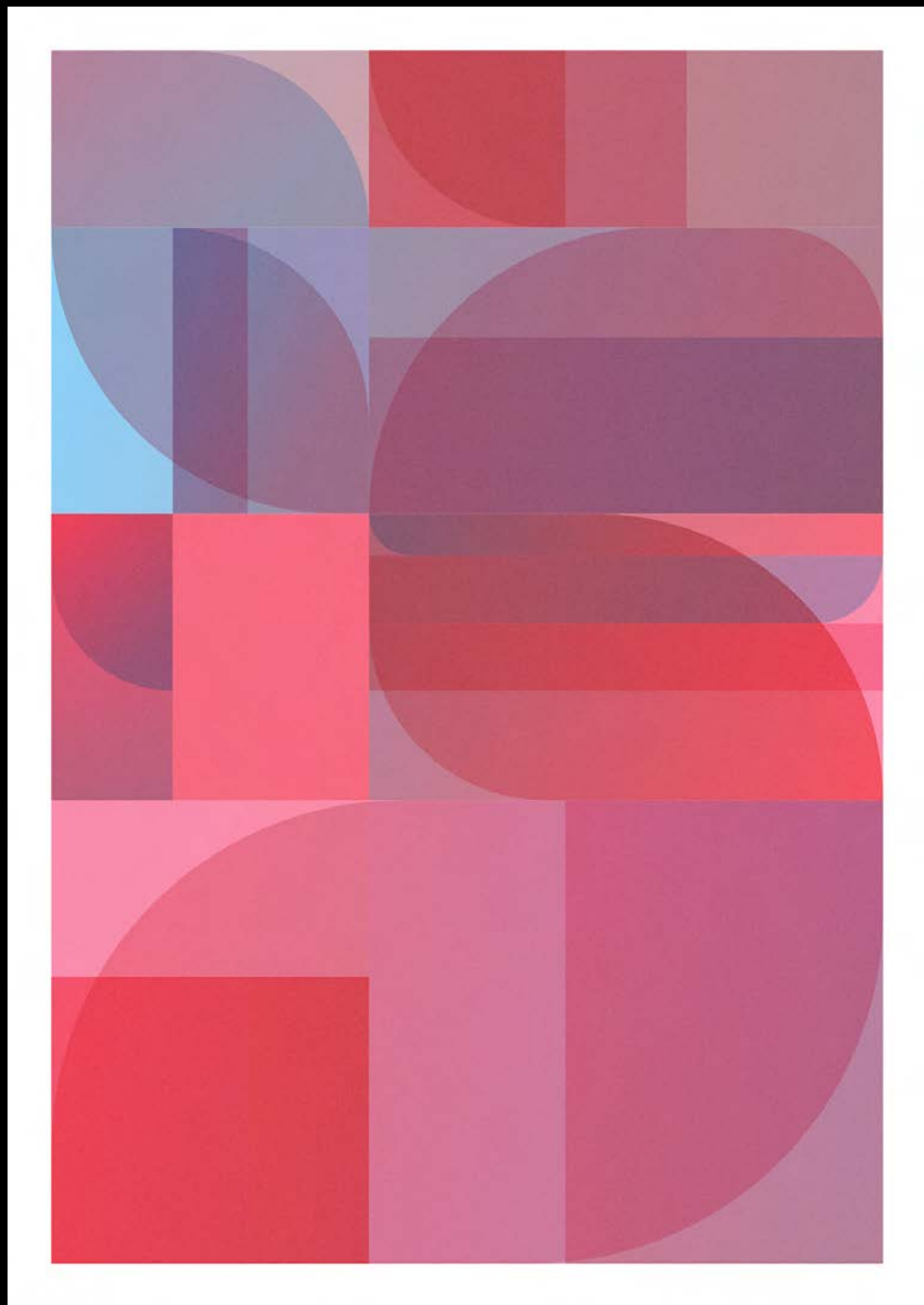
Unicode Compositions



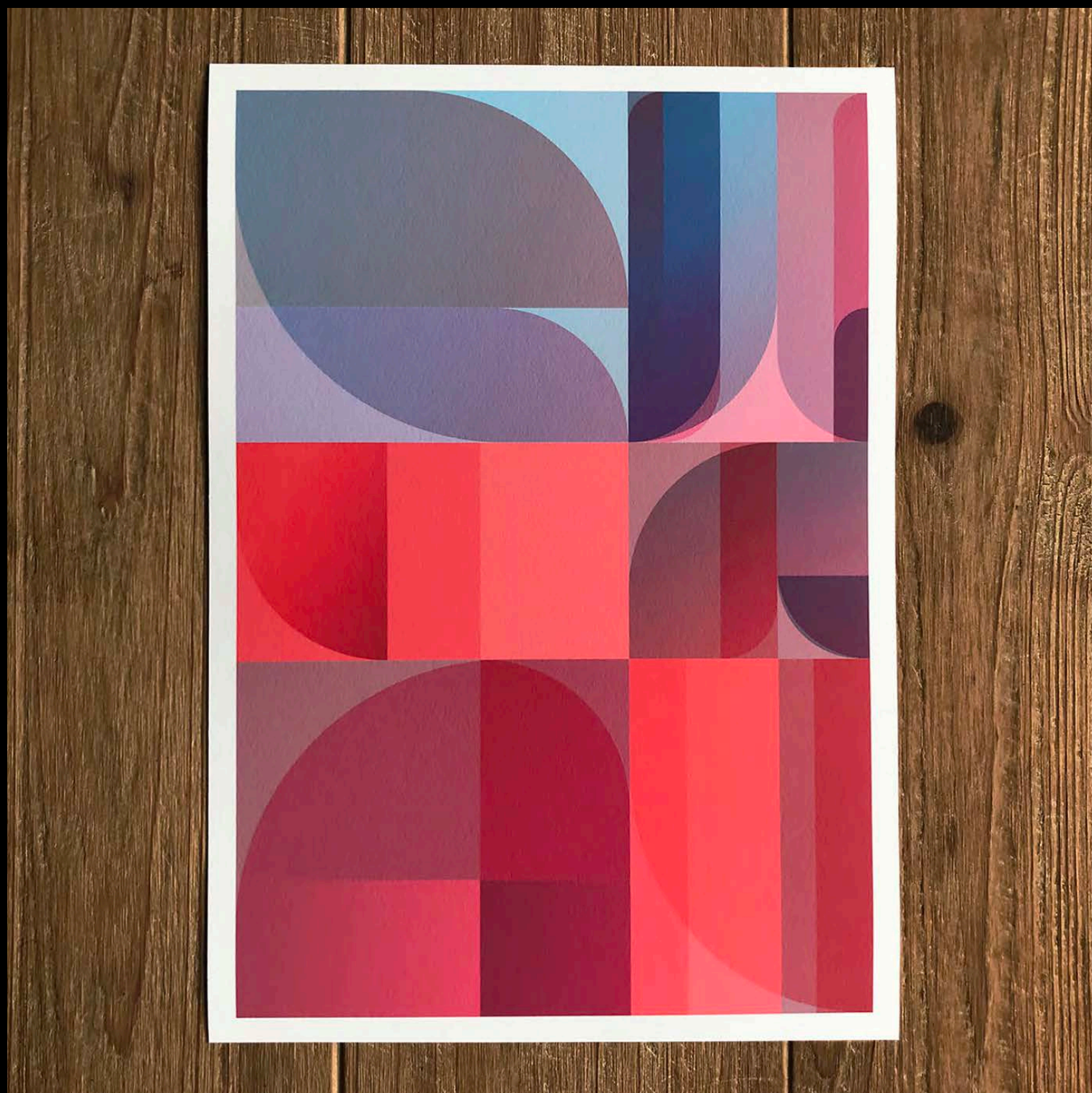
Computational Fur Rendering







TAU (2018)



TAU (2018)

A vector field visualization on a black background. The field consists of numerous small, light gray line segments representing vectors. These vectors are arranged in a grid-like pattern, with their orientation varying across the space. In the center of the image, the text "time to code !" is displayed in a white, sans-serif font.

time to code !



[take a breath]





noise functions



```
v = noise2D(x, y)
v = noise3D(x, y, z)
v = noise4D(x, y, z, w)
```






```
// value is in -1...1 range
const v = noise2D(x, y);

// map to 0..1 range
const n = v * 0.5 + 0.5;

// turn into a percentage
const L = Math.floor(n * 100);

// get color value
const hsl = `hsl(0, 0%, ${L}%)`;
```





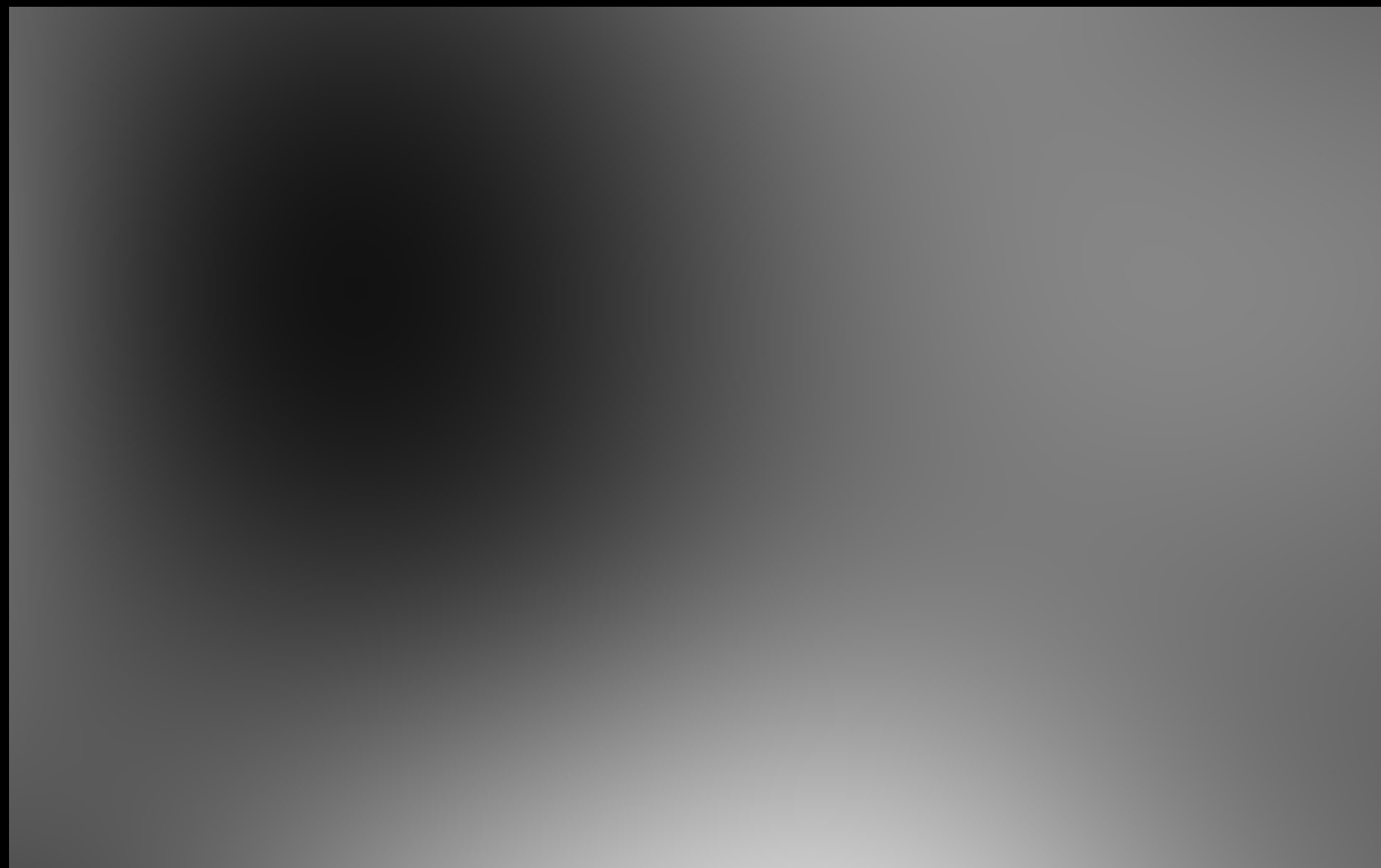

```
// frequency of the noise signal
```

```
const frequency = 5.0;
```

```
const v = noise2D(x * frequency, y * frequency);
```




/ frequency = 5.0 \



/ frequency = 0.5 \



theory



Sol LeWitt (1928 – 2007)



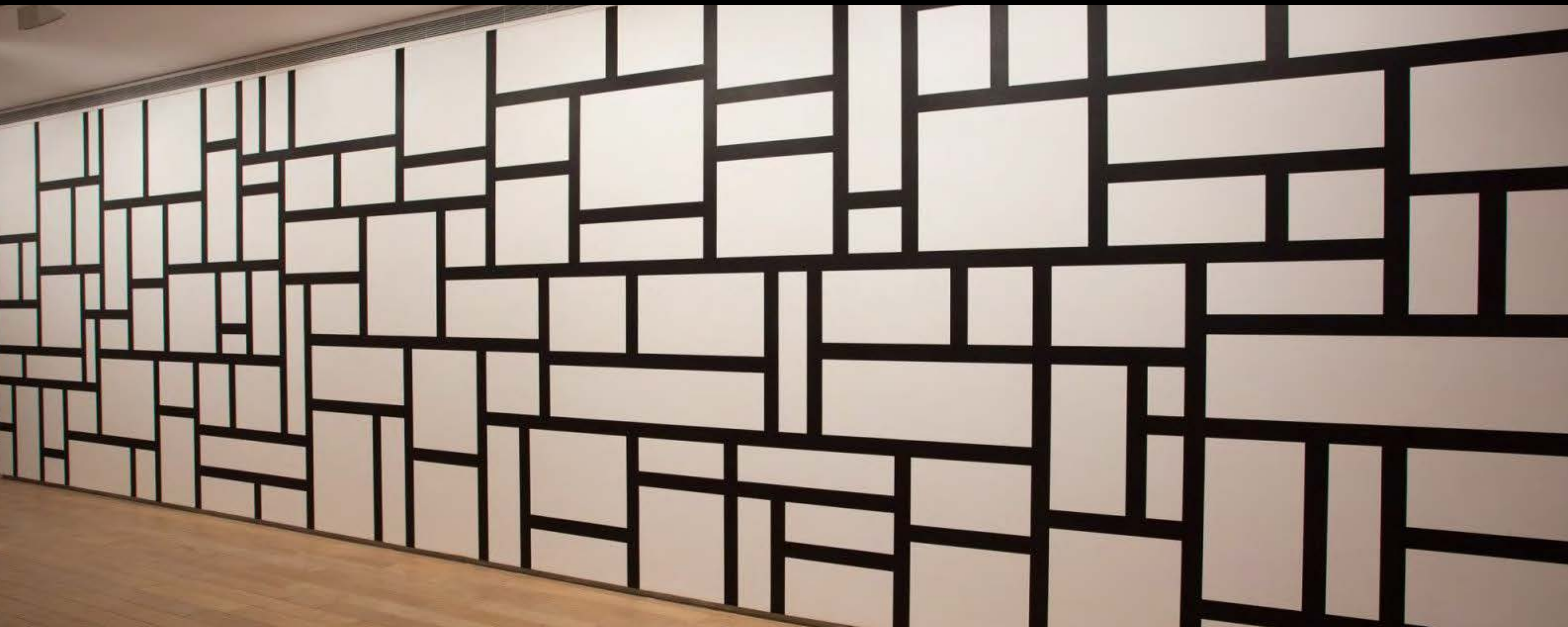
Wall Drawing #1131



Wall Drawing #260



Wall Drawing #692



Wall Drawing #614

LeWitt created the following instructions for this work:

A six-inch (15 cm) grid covering the walls. Lines from corners, sides, and center of the walls to random points on the grid.

1st wall: Red lines from the midpoints of four sides;

2nd wall: Blue lines from four corners;

3rd wall: Yellow lines from the center;

4th wall: Red lines from the midpoints of four sides, blue lines from four corners;

5th wall: Red lines from the midpoints of four sides, yellow lines from the center;

6th wall: Blue lines from four corners, yellow lines from the center;

7th wall: Red lines from the midpoints of four sides, blue lines from four corners, yellow lines from the center.

Each wall has an equal number of lines. (The number of lines and their length are determined by the draftsman.)



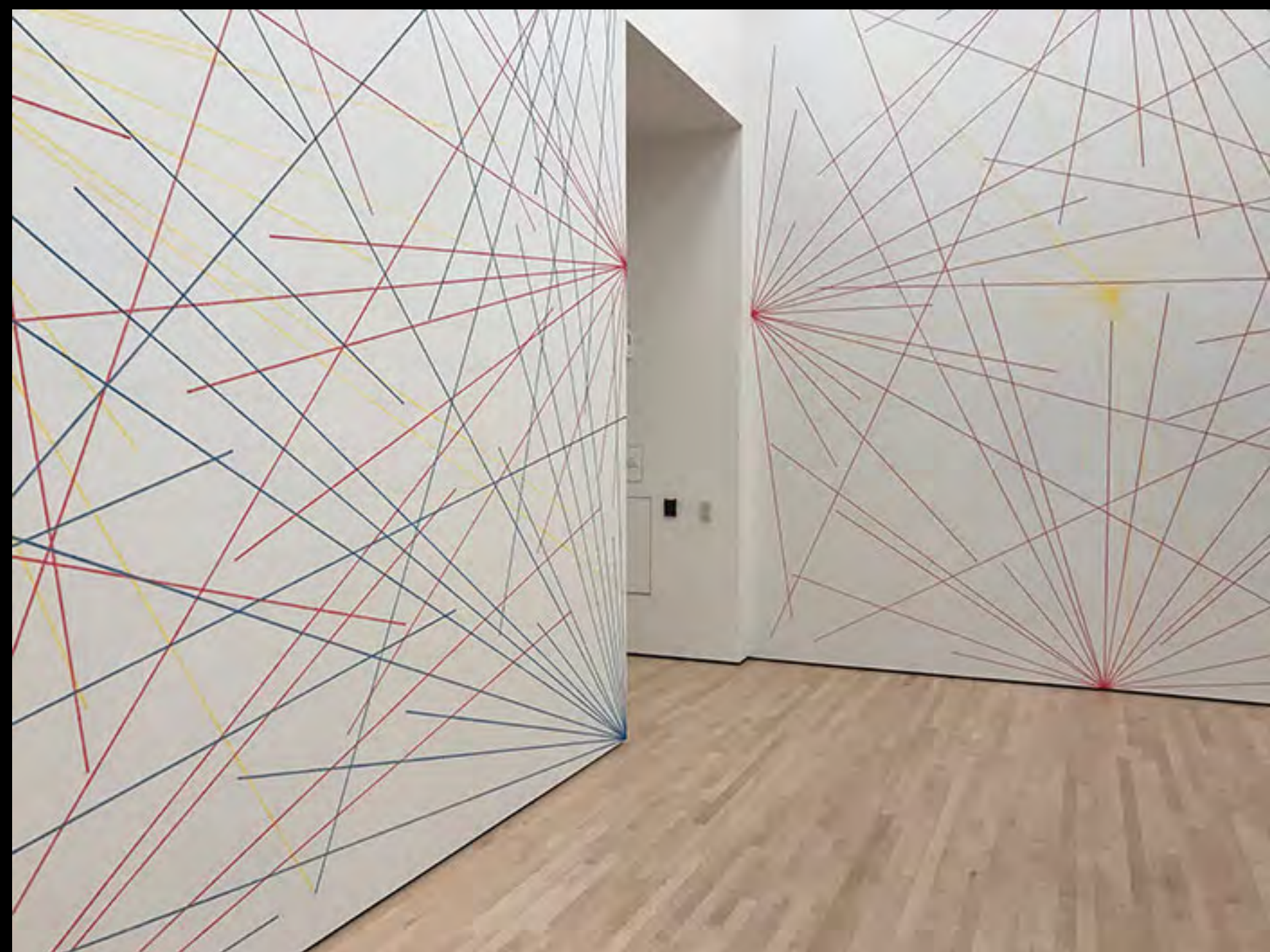
Artist Roland Lück on making
LeWitt's Wall Drawings

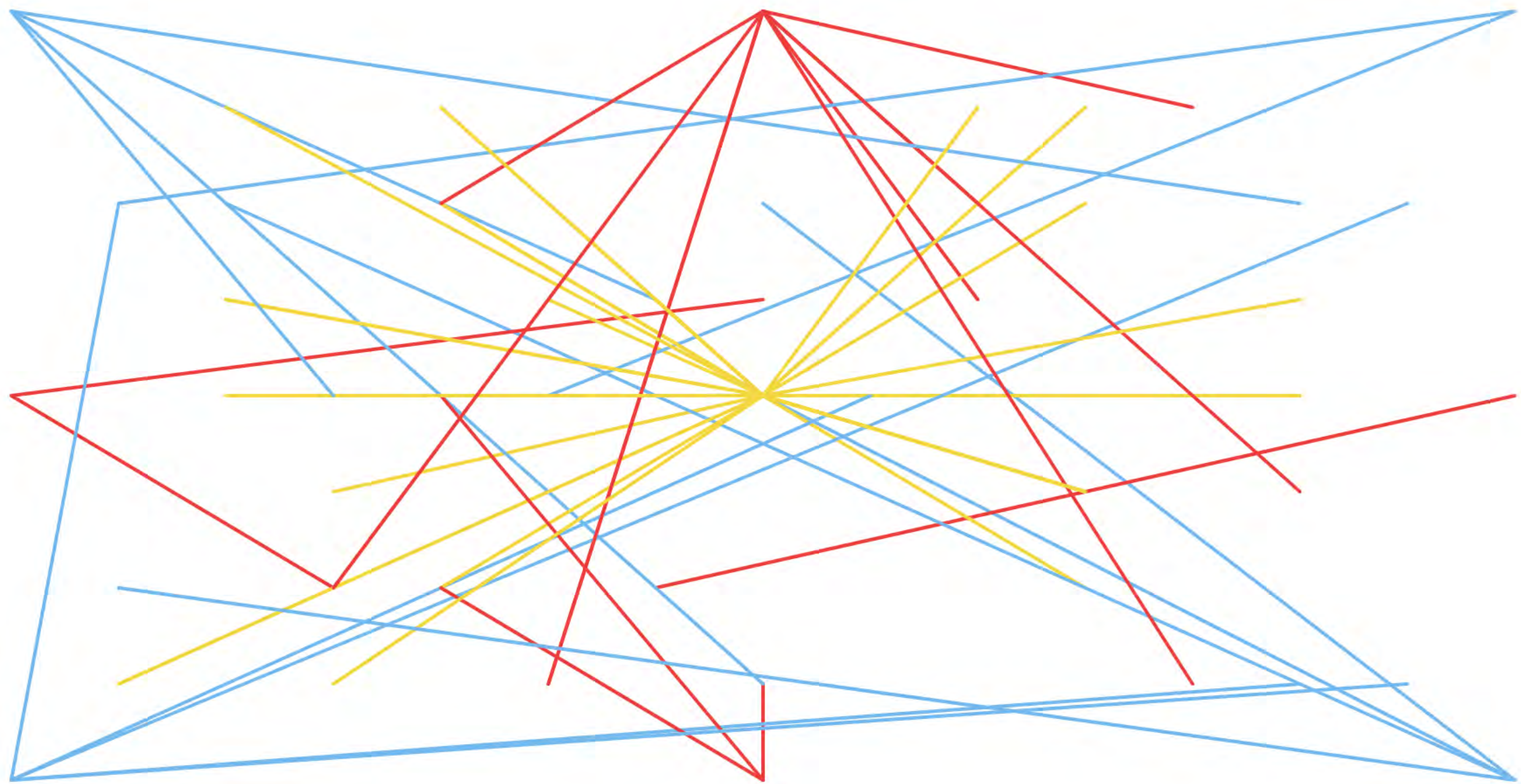
Wall Drawing #273

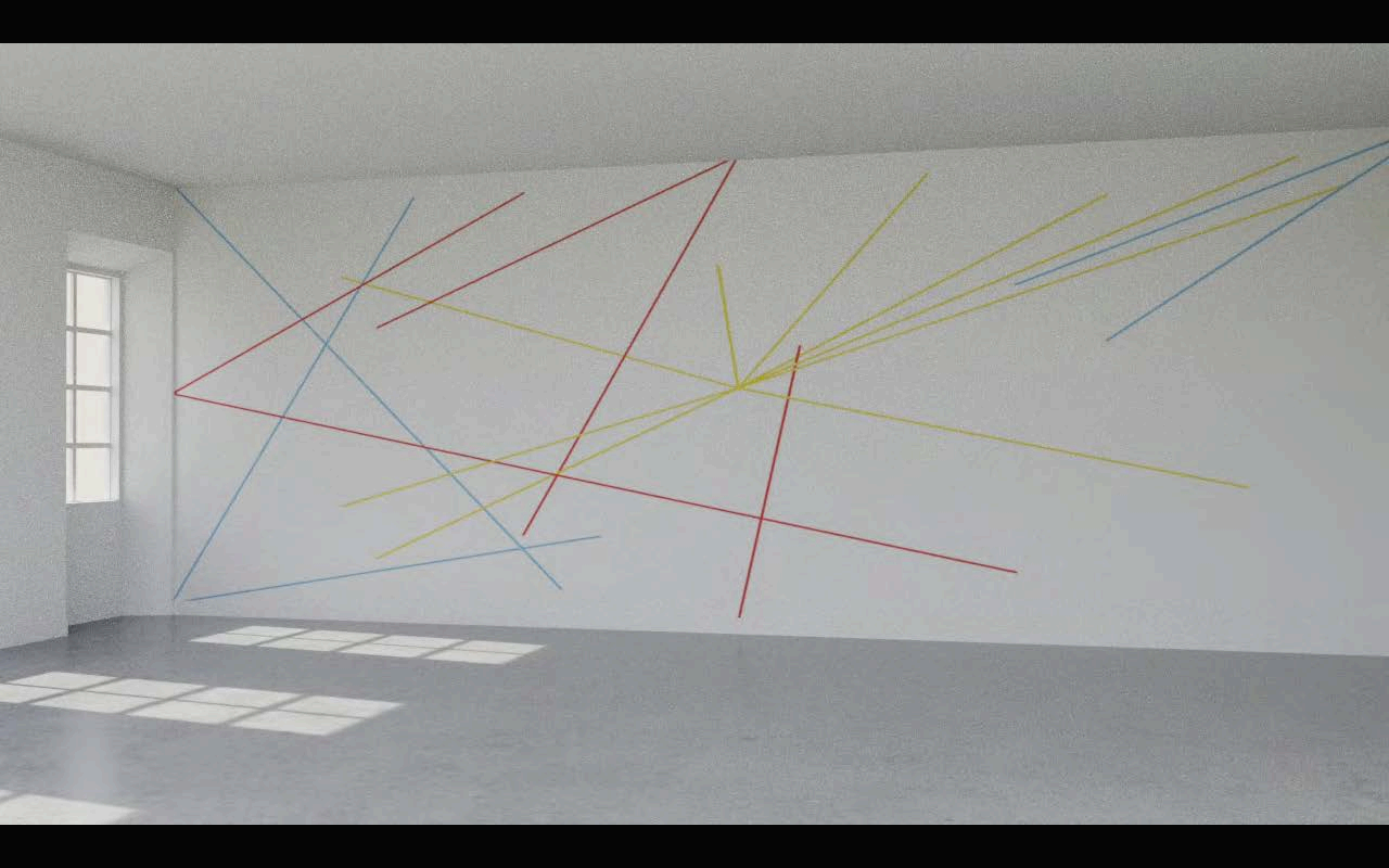
A six-inch (15 cm) grid covering the walls. Lines from corners, sides, and centre of the walls to random points on the grid.

...

7th wall: Red lines from the midpoints of four sides, blue lines from four corners, yellow lines from the center.



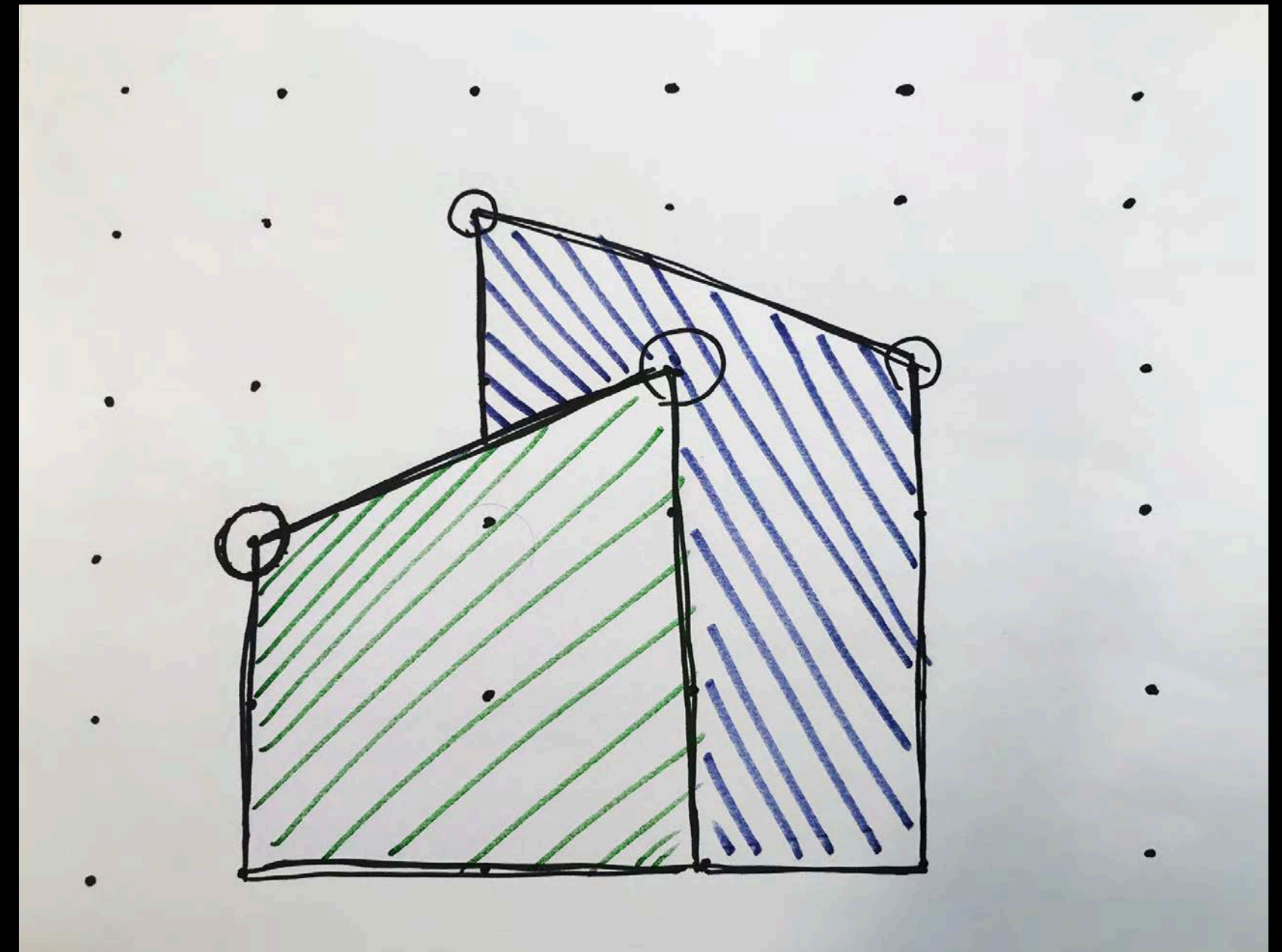




<https://solvingsol.com/>

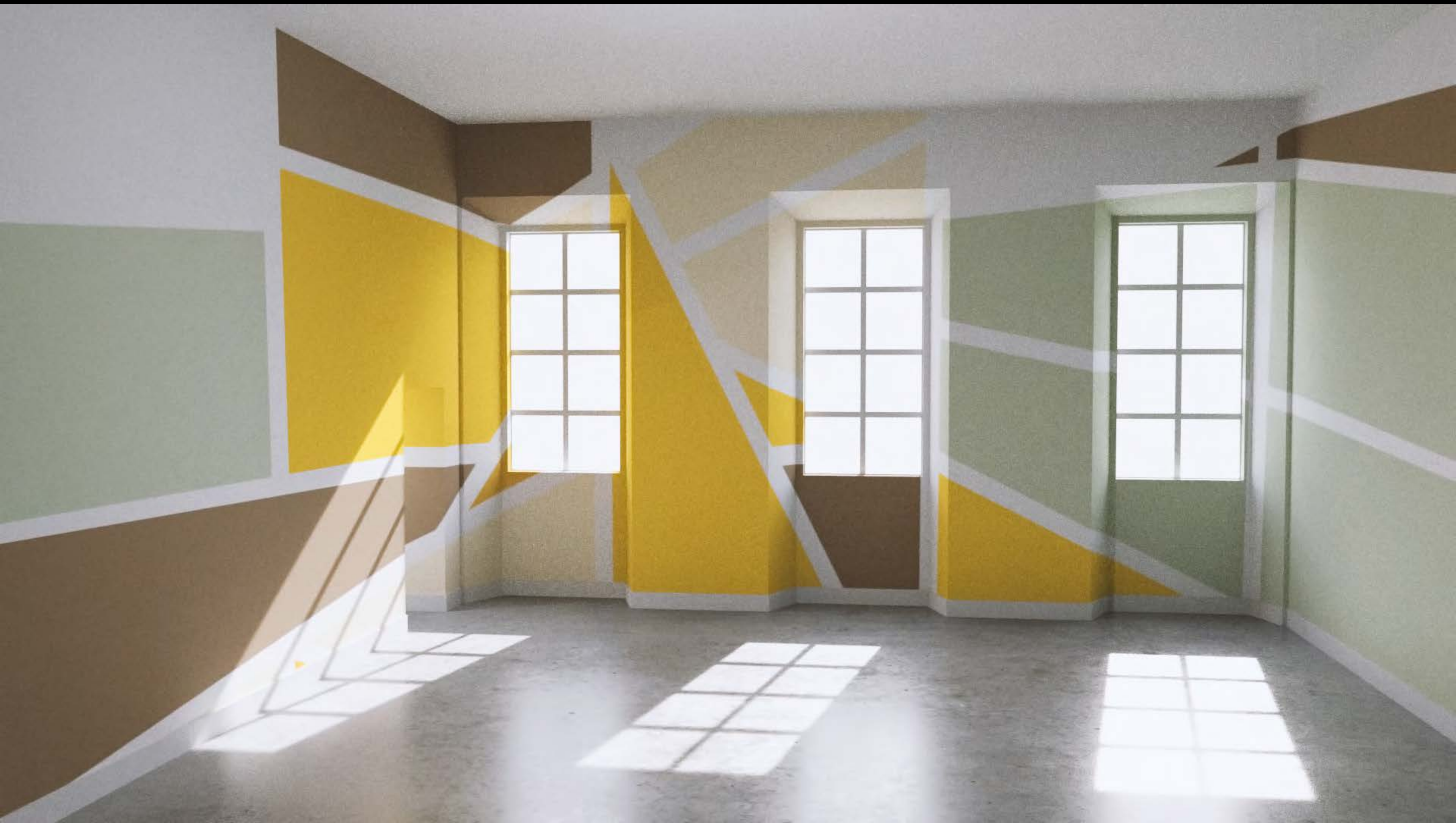
Generative Wall Drawing #2

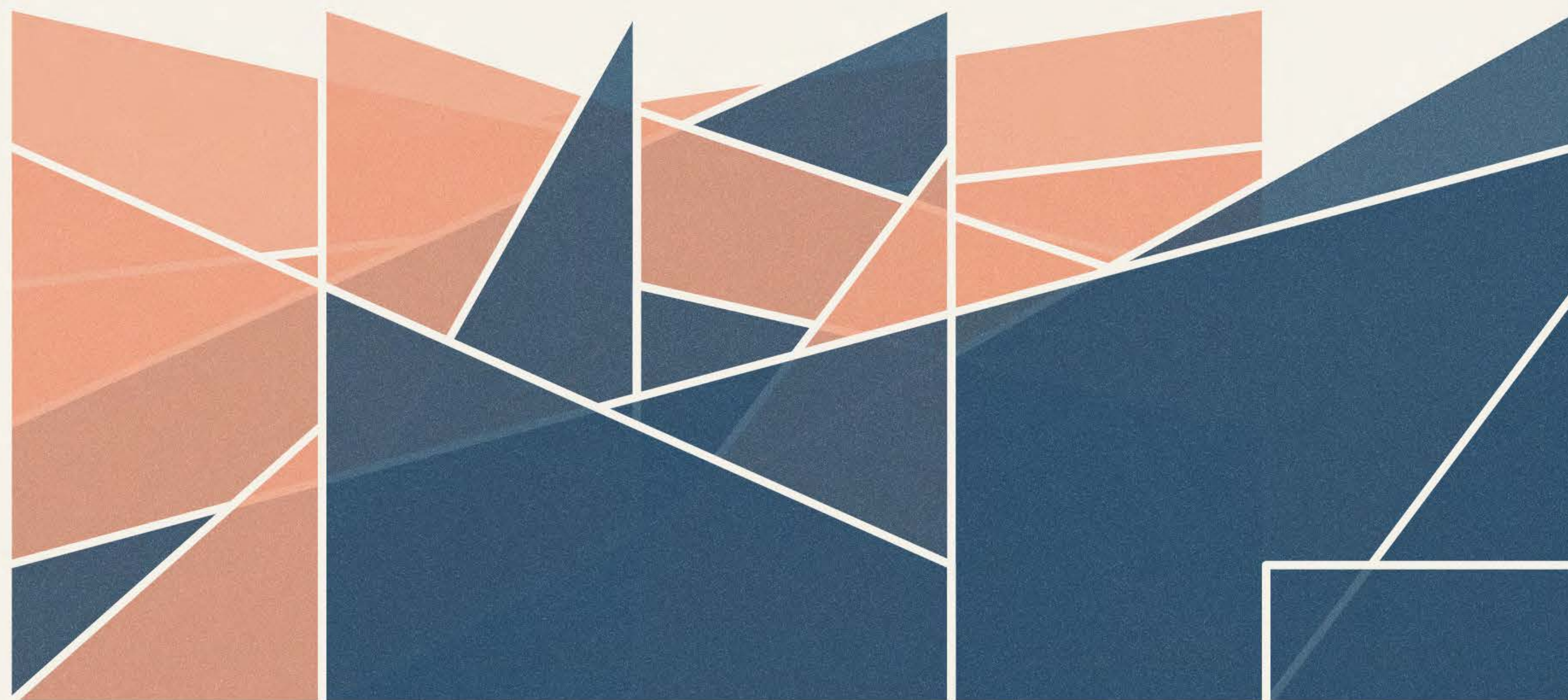
- Using a 6x6 grid of evenly spaced points
- Connect two random points on the grid; forming a trapezoid with two parallel sides extending down
- Fill the trapezoid with a colour, then stroke with the background colour
- Find another two random points and repeat; continuing until all grid points are exhausted
- Layer the shapes by the average Y position of their two grid points

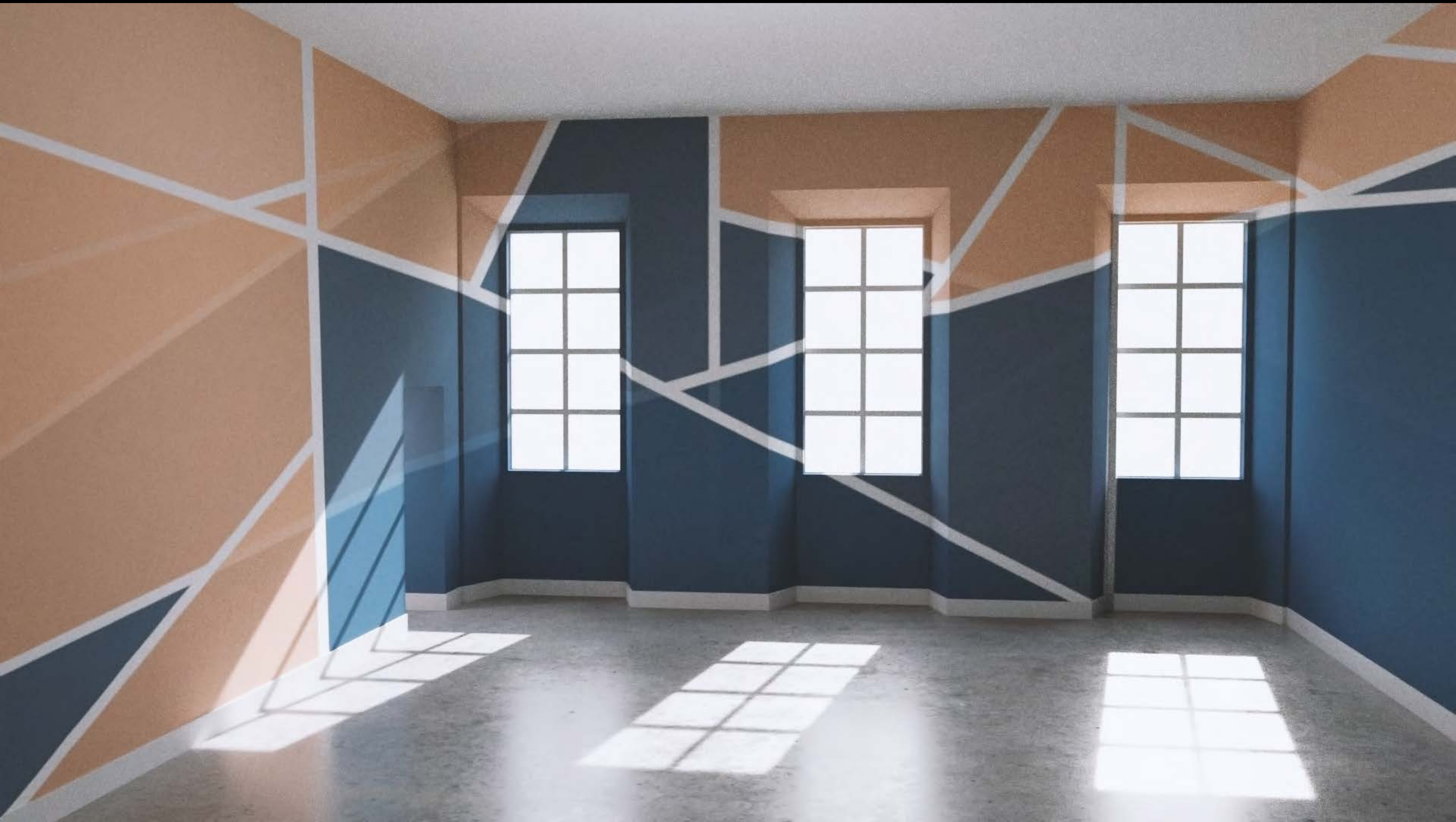


welcome back !



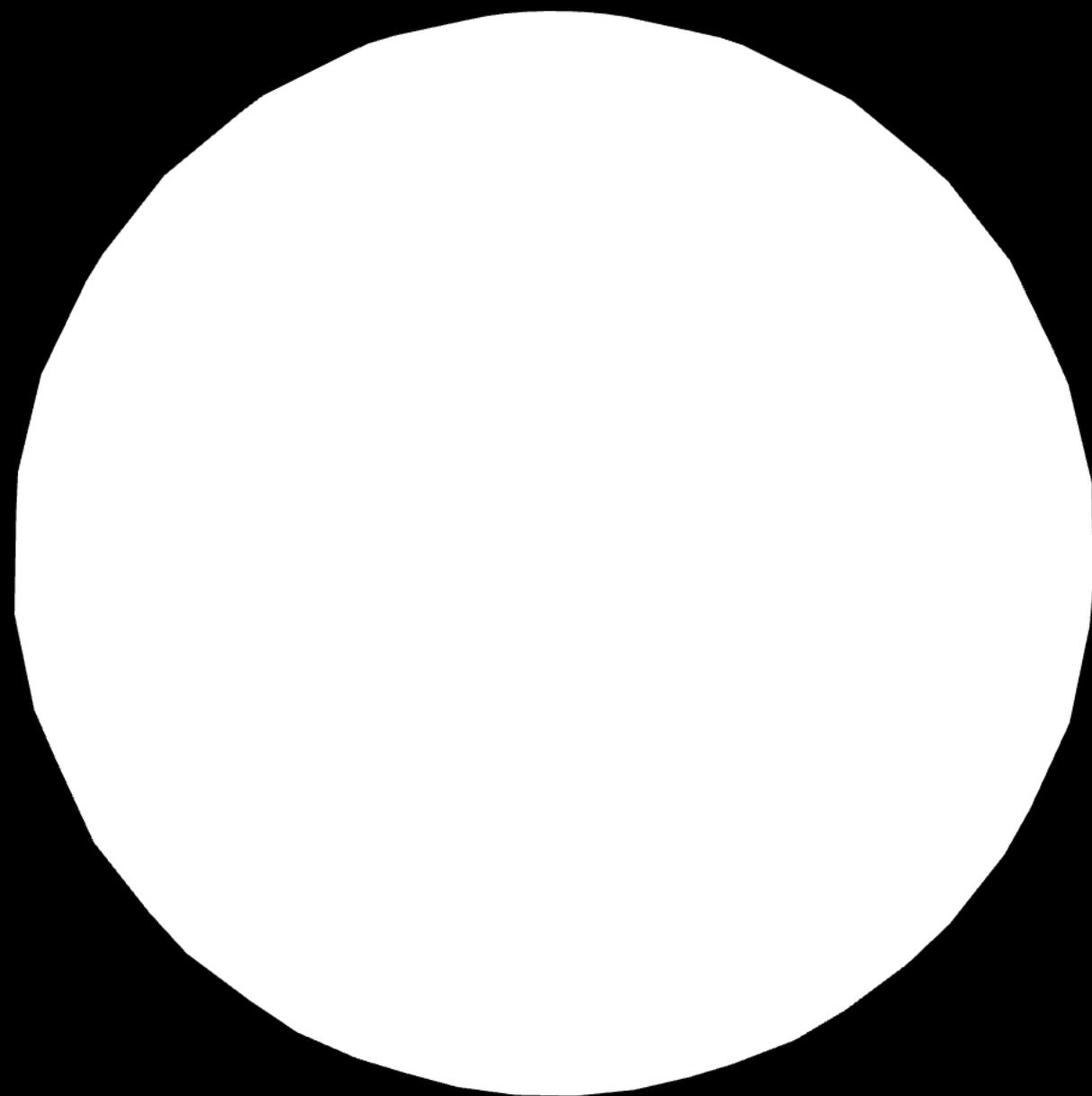






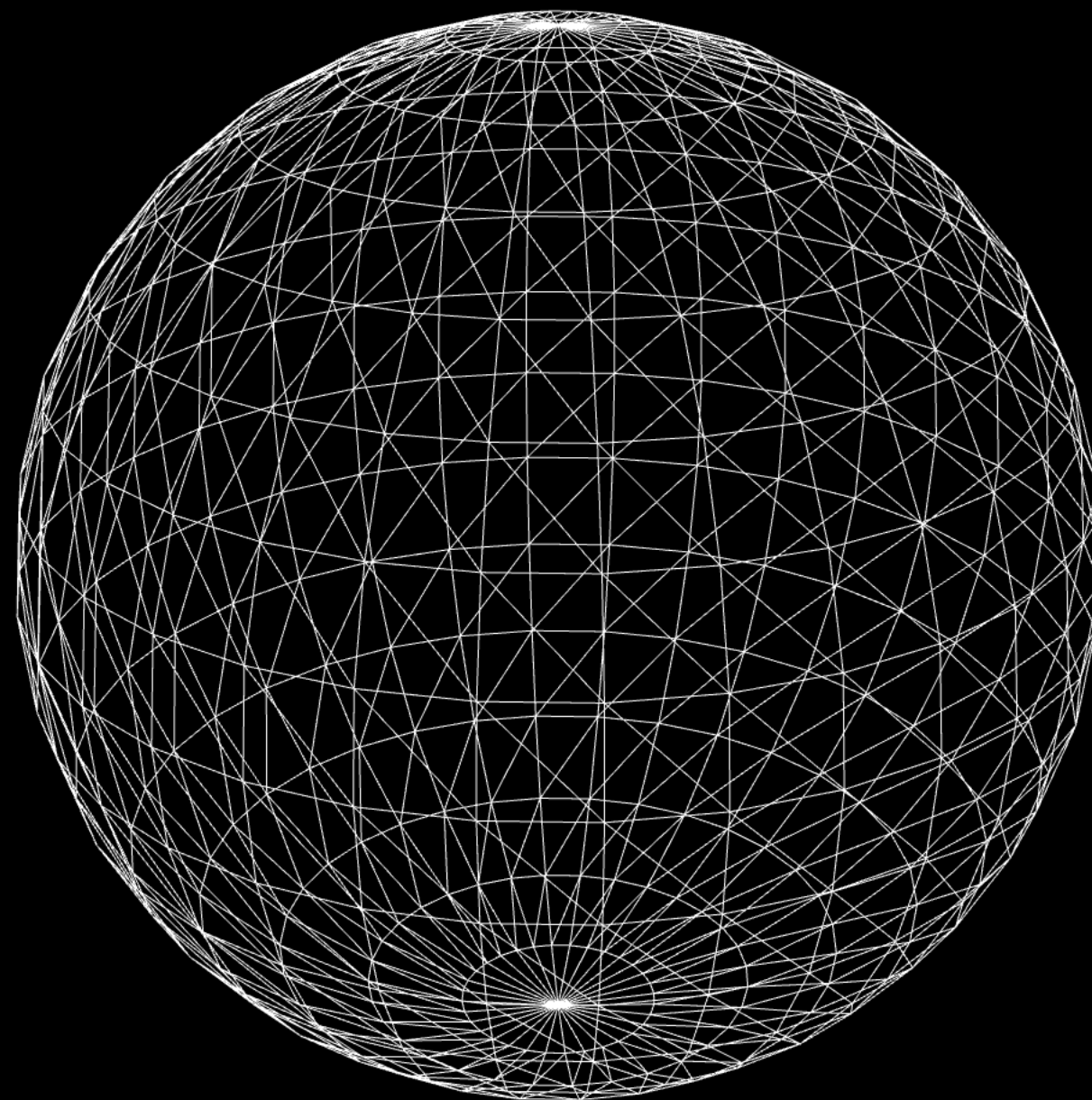
The image features a black background with a grid of small, white line segments. These segments are arranged in a pattern that suggests a vector field or a coordinate system. The segments are oriented in various directions, some pointing towards the center and others away from it, creating a sense of depth and movement. In the center of the image, the text "the third dimension" is written in a clean, white, sans-serif font.

the third dimension



material

MeshBasicMaterial



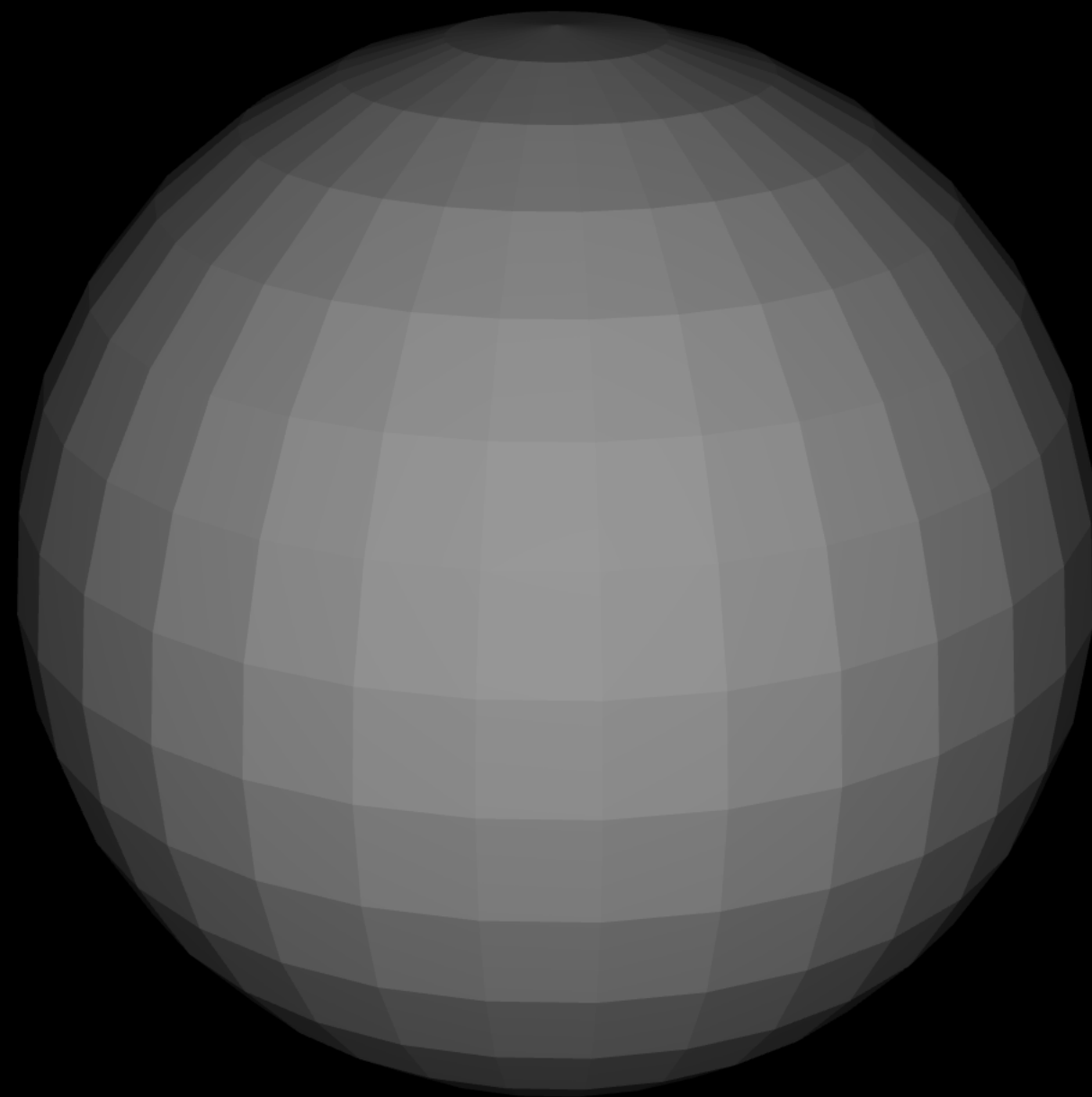
material

MeshBasicMaterial (wireframe)



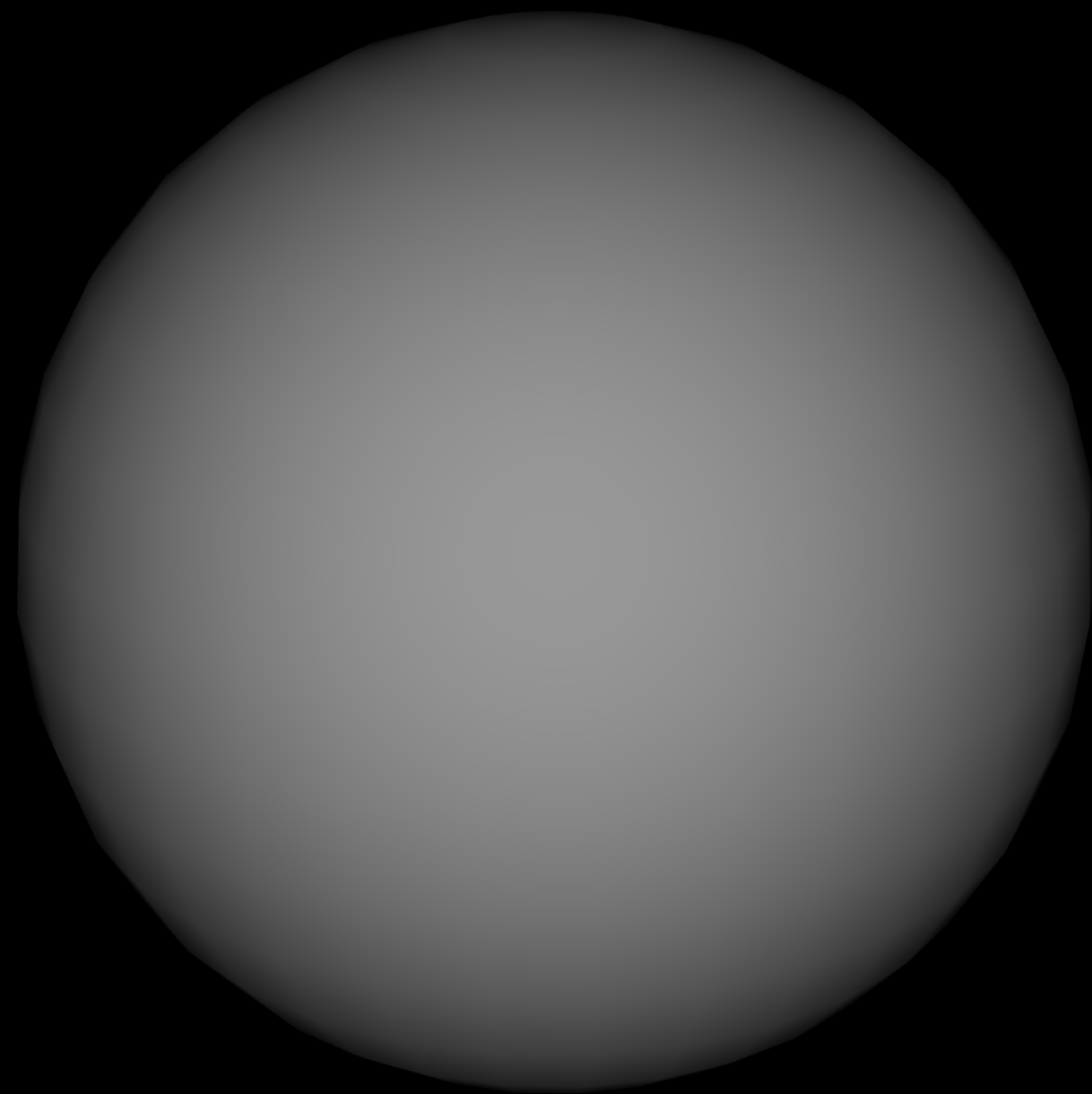
material

MeshNormalMaterial



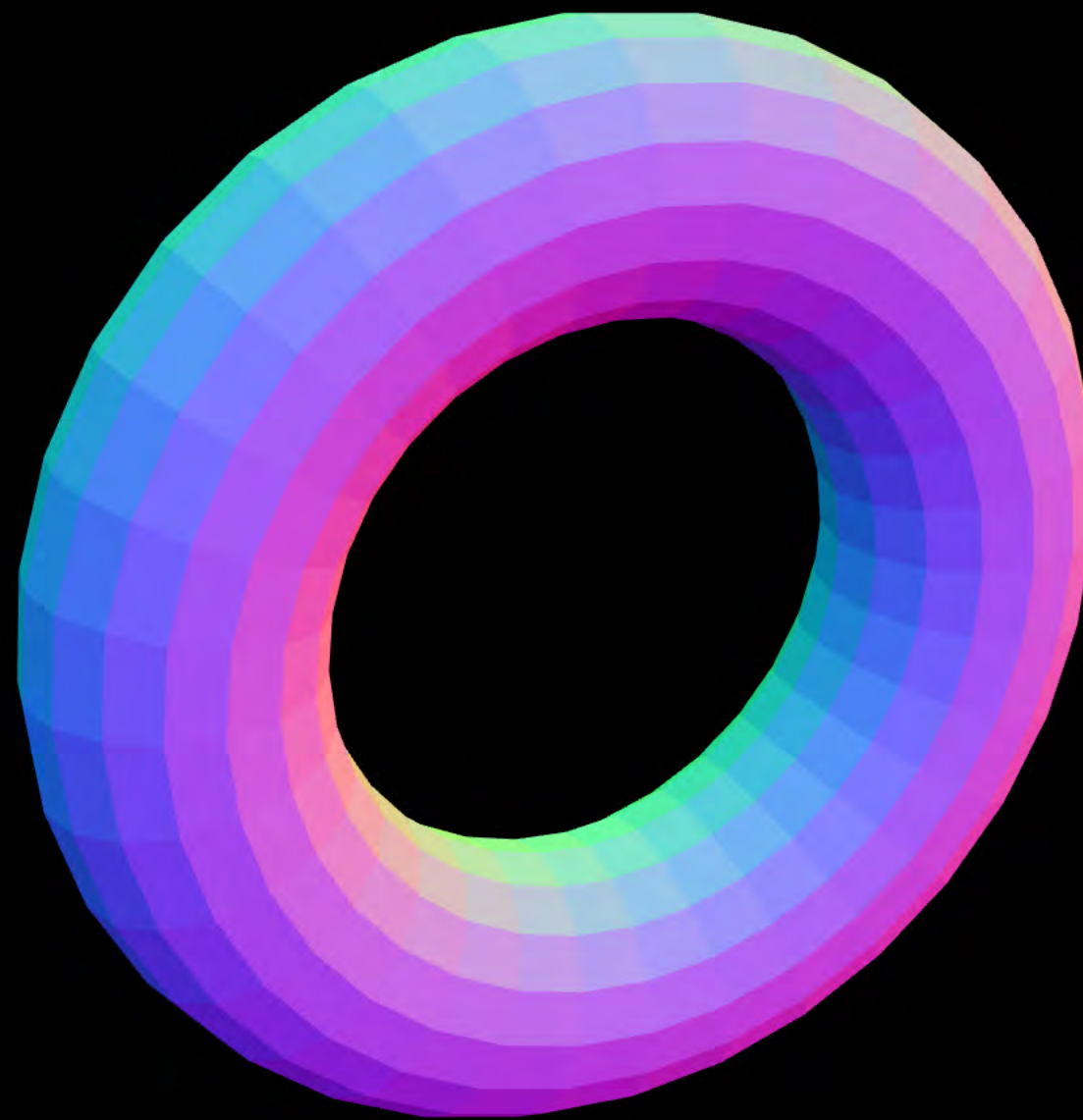
material

MeshStandardMaterial

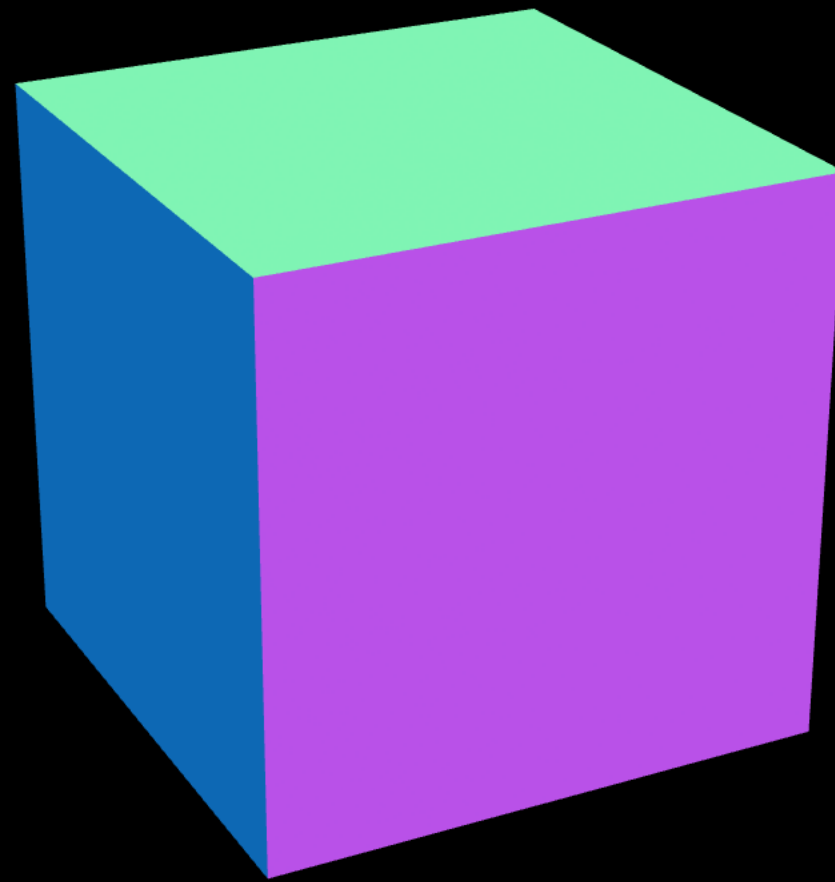


material

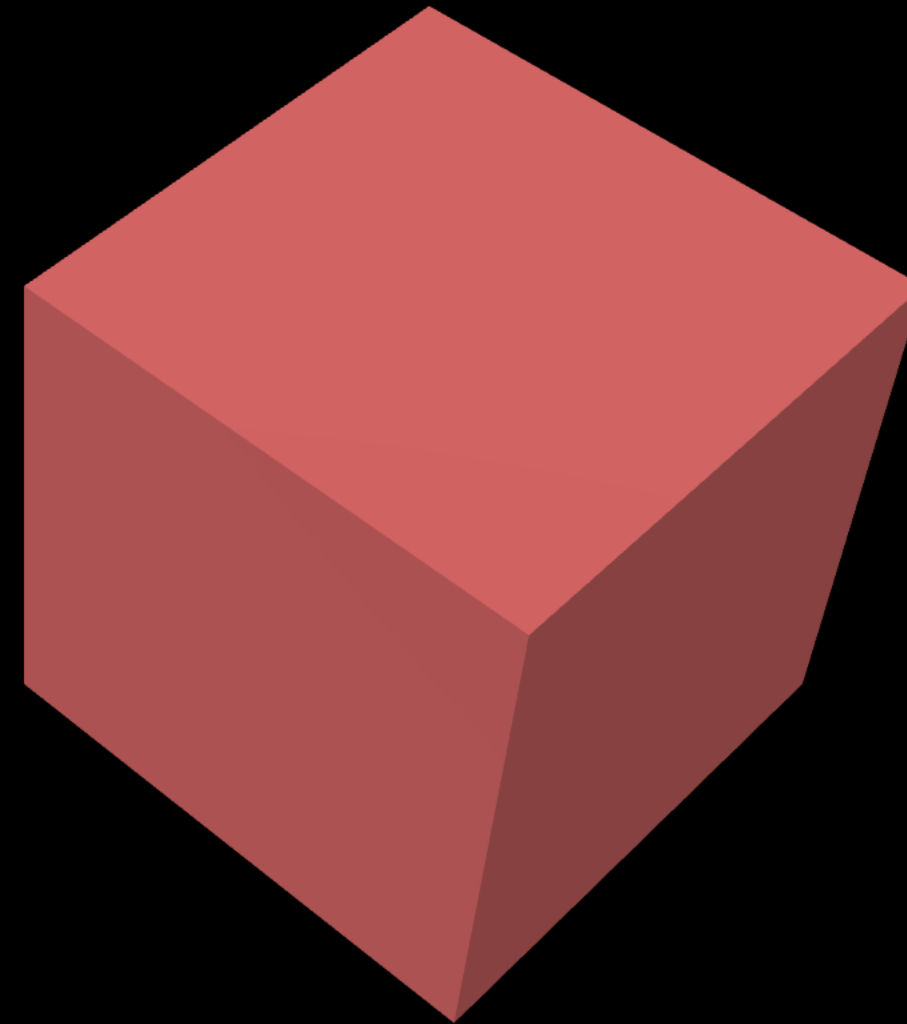
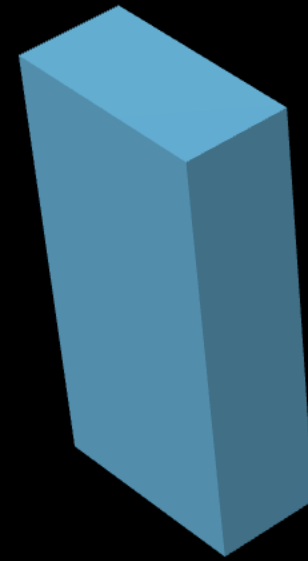
MeshStandardMaterial



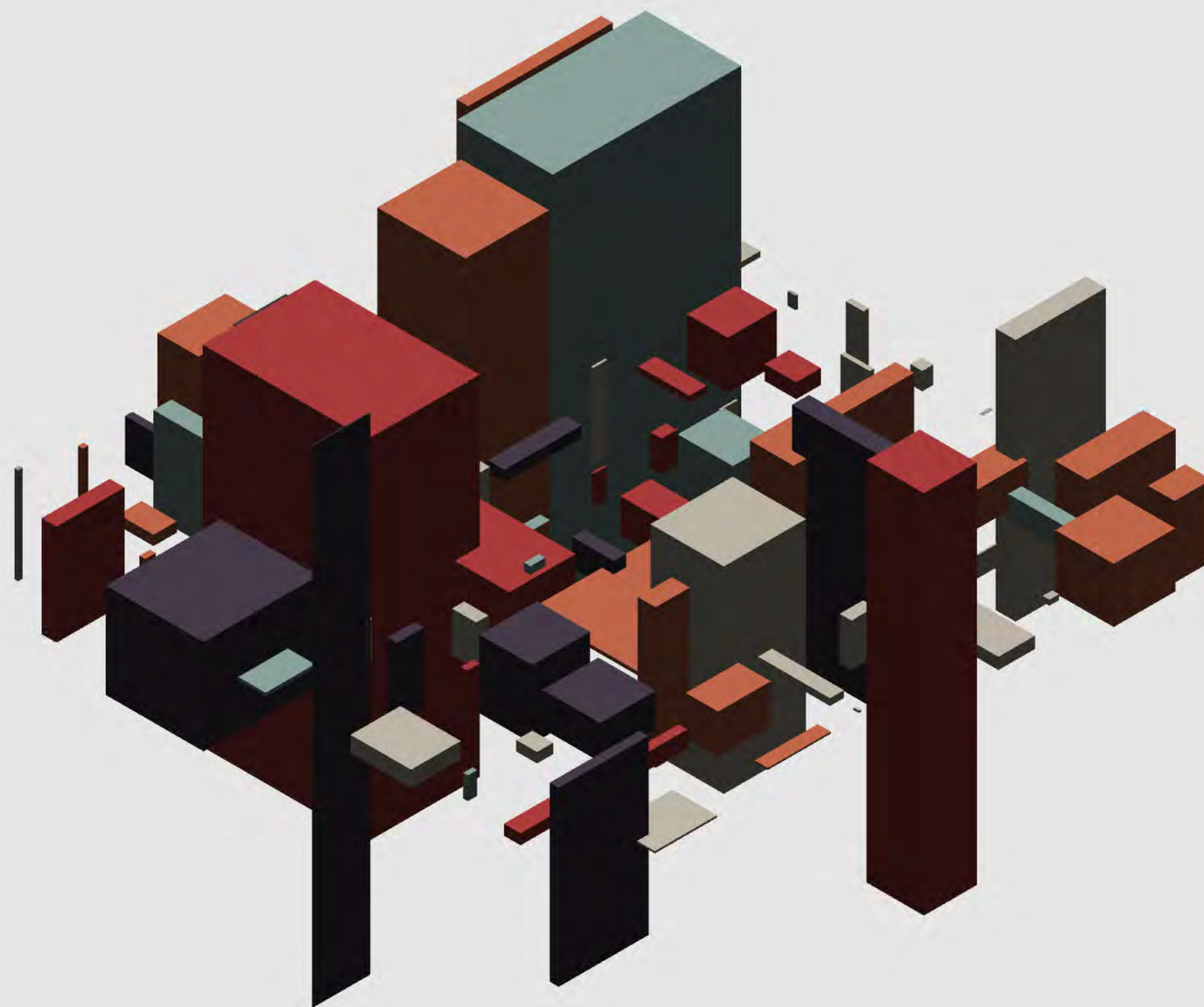
geometry
TorusGeometry



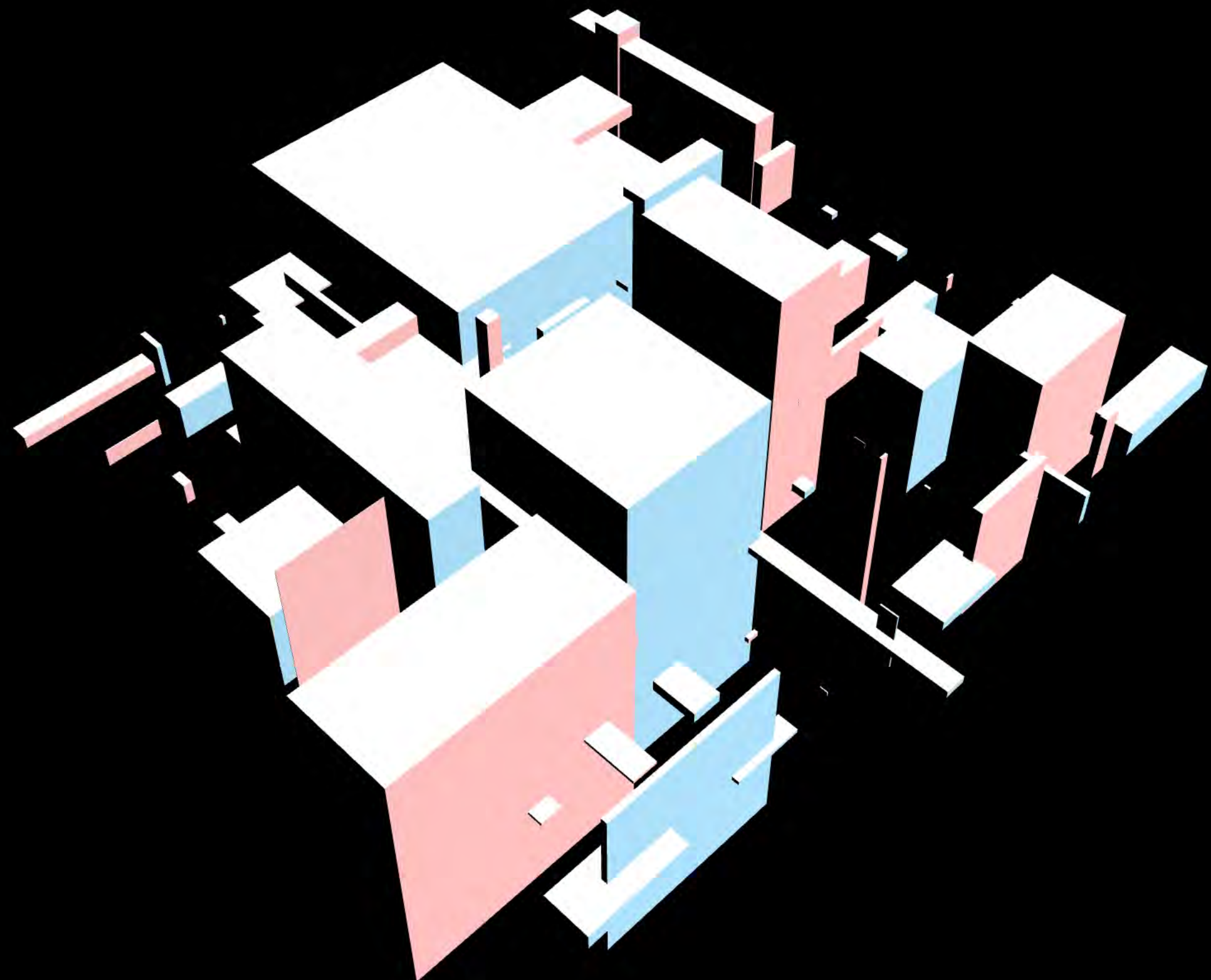
geometry
BoxGeometry



mesh

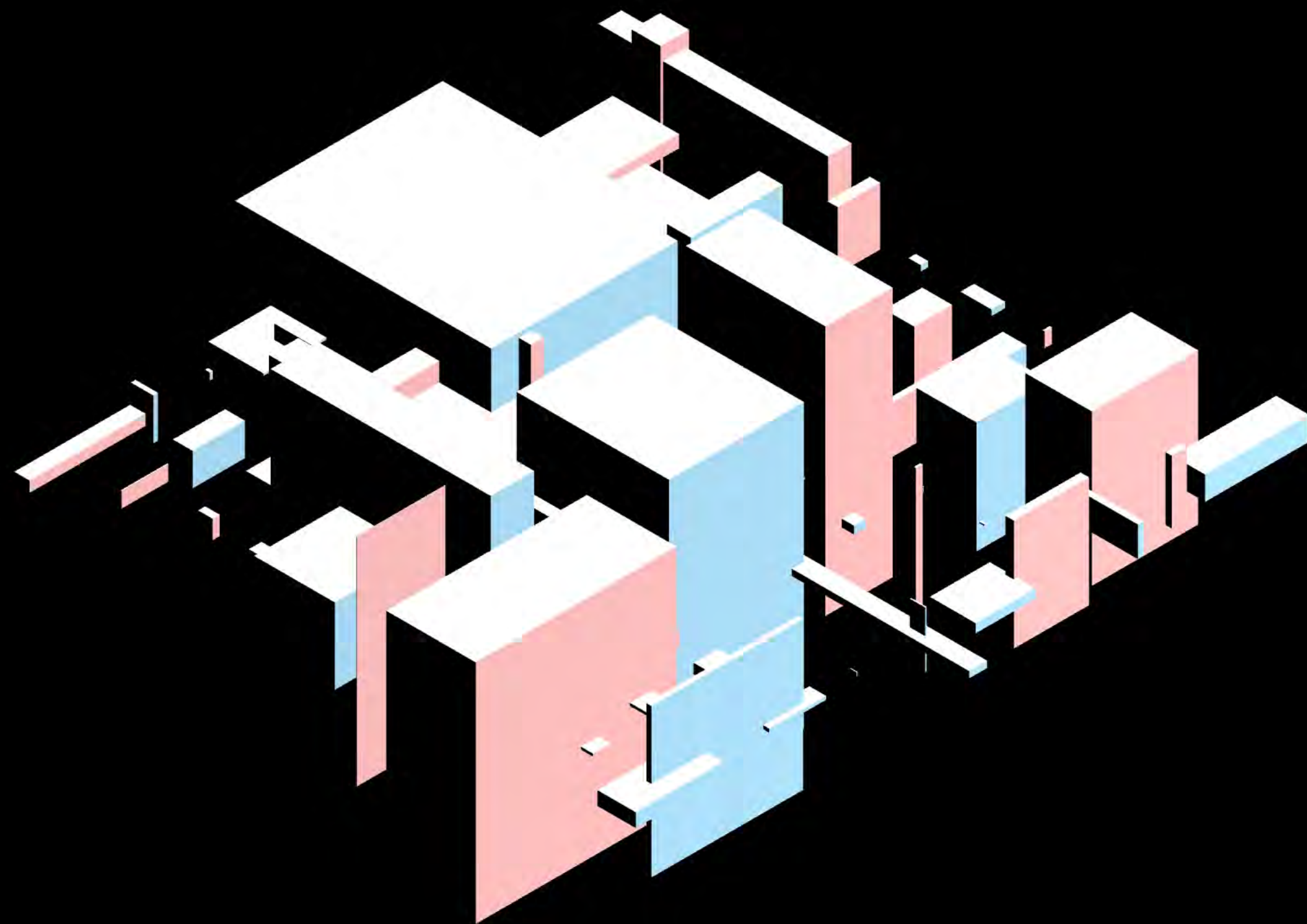


scene



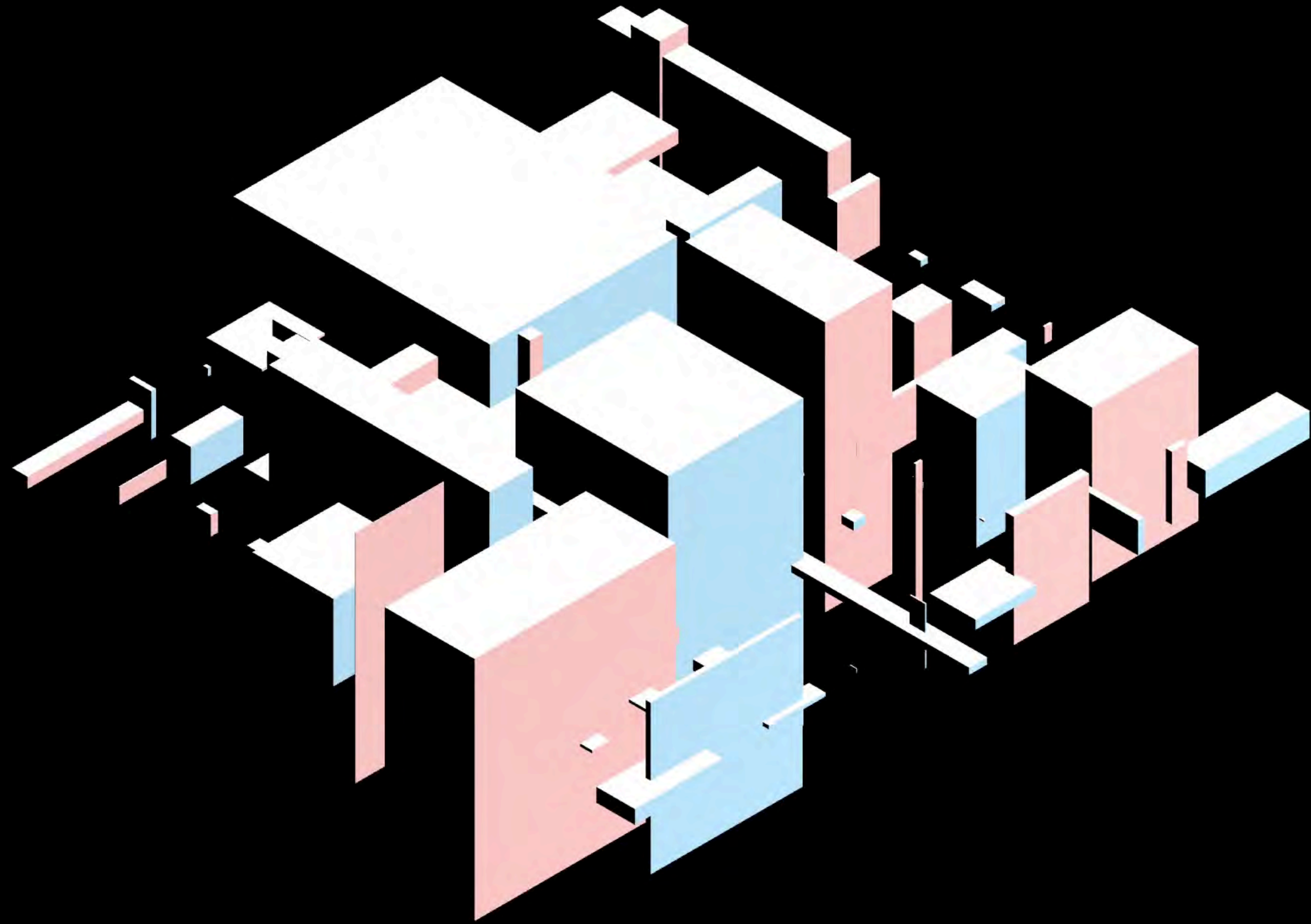
camera

PerspectiveCamera



camera

OrthographicCamera





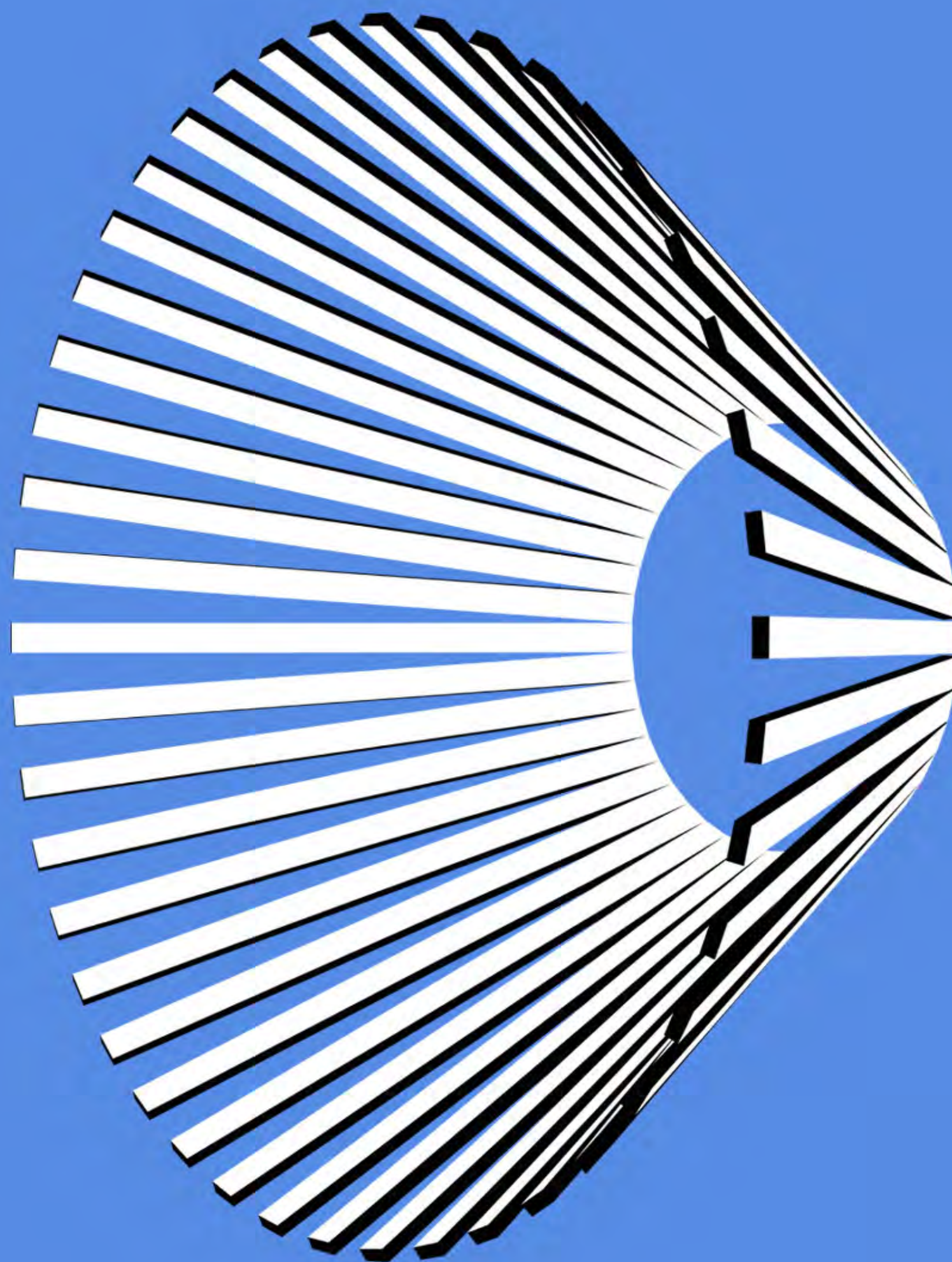




time to code !



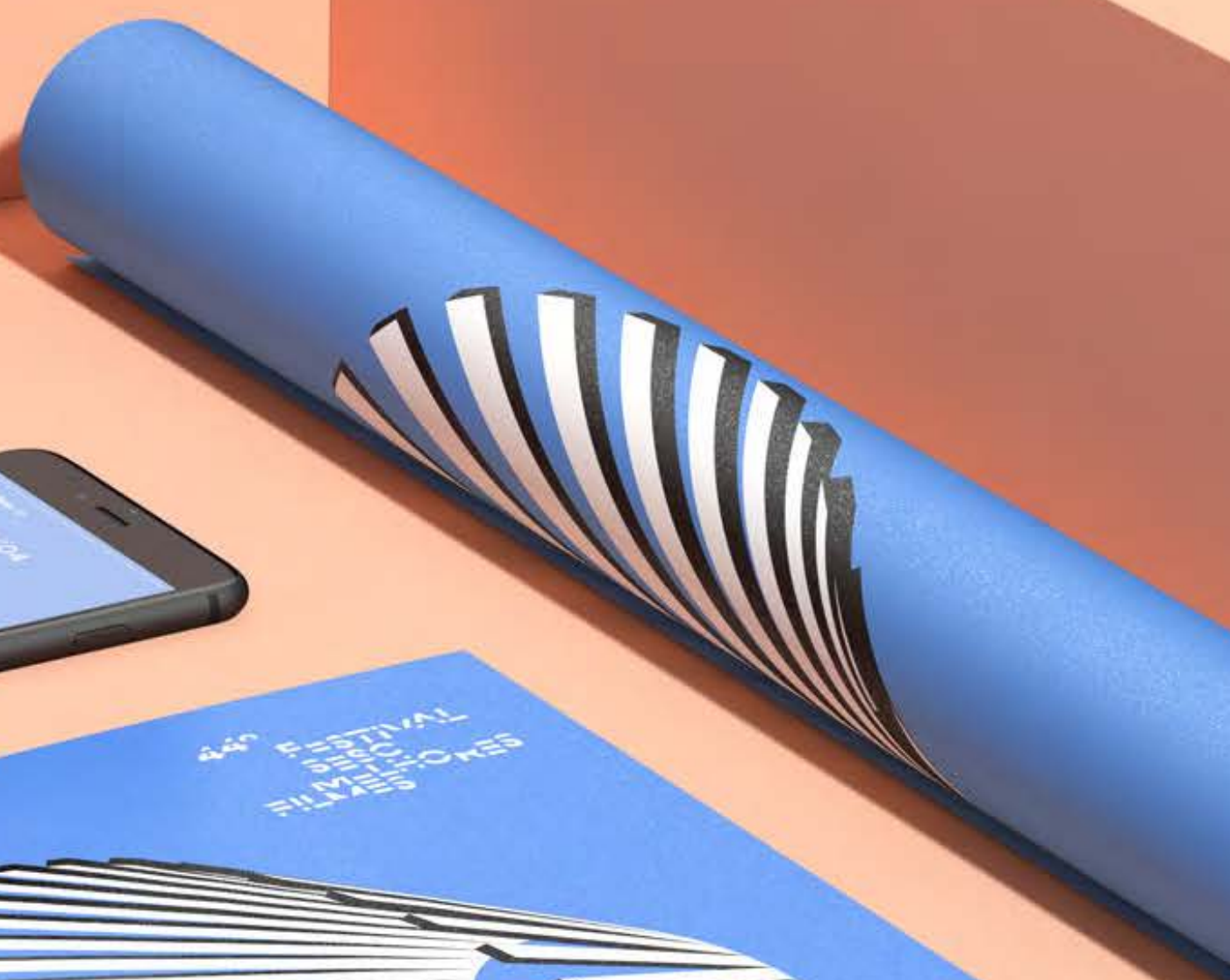
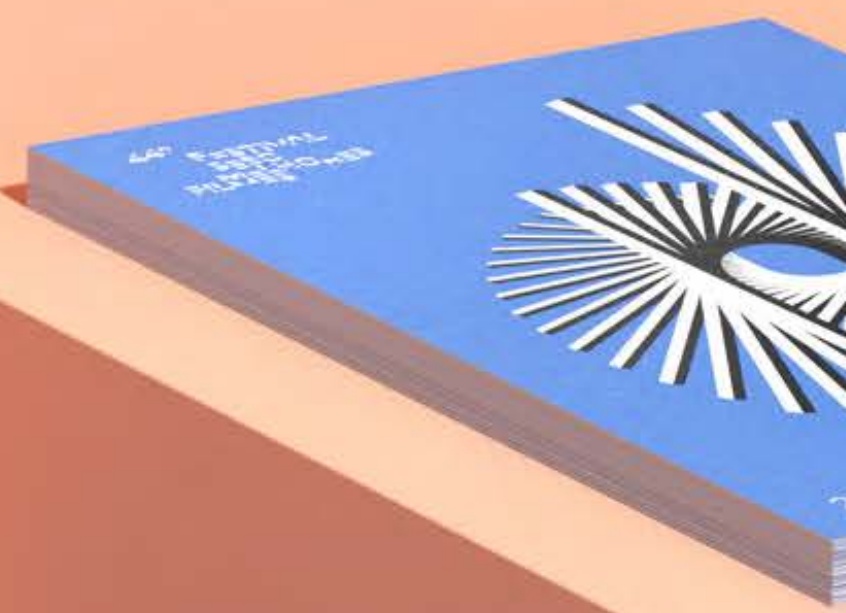
questions so far ?



44º Sesc Melhores Filmes



44º Sesc Melhores Filmes





let's make a GIF !

<https://gifttool.surge.sh/>

shaders

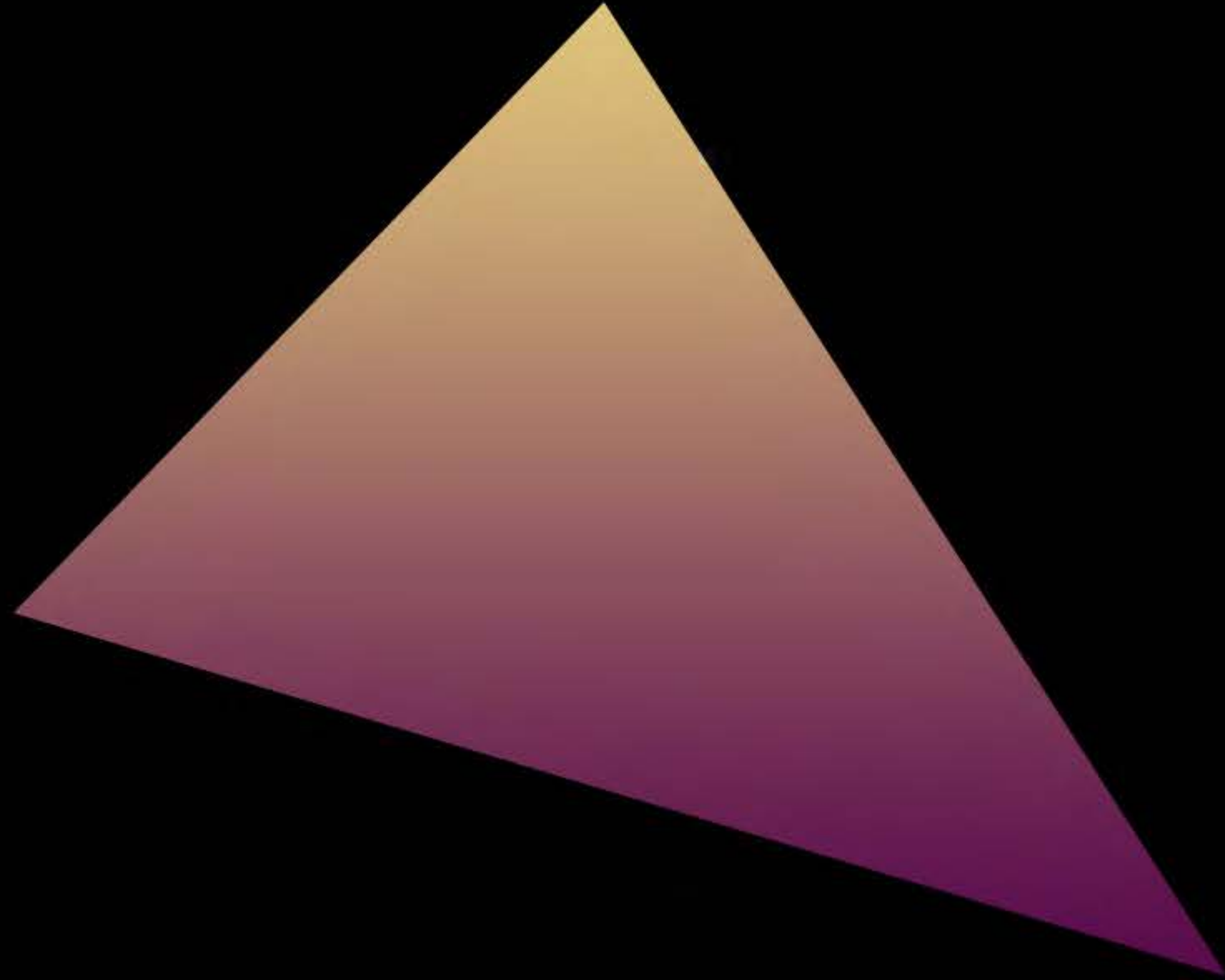


```
// Default floating point precision  
precision highp float;
```

```
// Inputs  
varying vec2 vUv;
```

```
// Variables from JavaScript  
uniform float time;
```

```
// Main function  
void main () {  
    // Output color  
    gl_FragColor = vec4(1.0);  
}
```







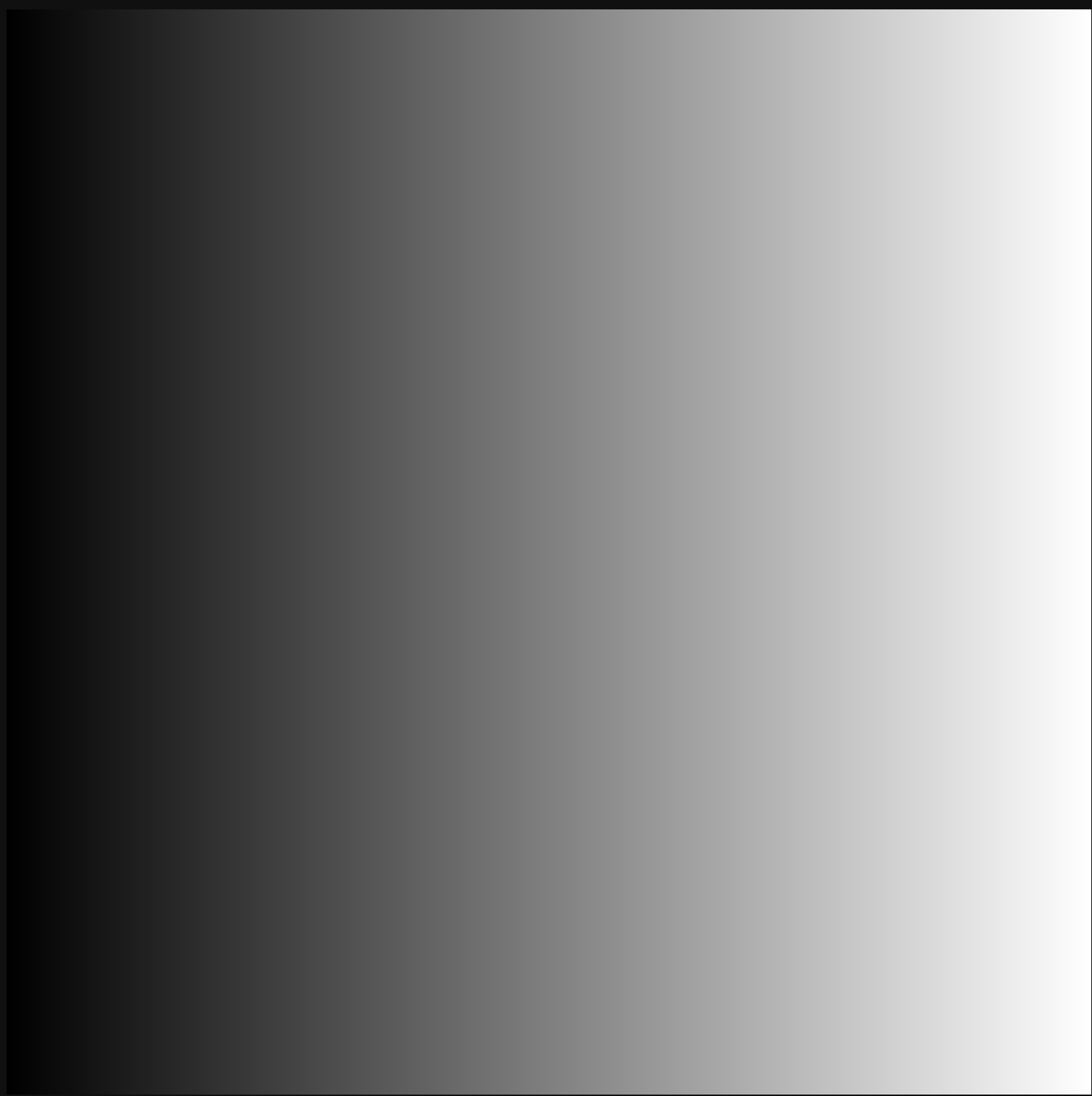


```
// Default floating point precision  
precision highp float;
```

```
// Inputs  
varying vec2 vUv;
```

```
// Variables from JavaScript  
uniform float time;
```

```
// Main function  
void main () {  
    // Output color  
    gl_FragColor = vec4(1.0);  
}
```





```
precision highp float;

varying vec2 vUv;

// Main function
void main () {
    // Create a RGB color
    vec3 color = vec3(vUv.x);

    // Create an opacity
    float alpha = 1.0;

    // Output color
    gl_FragColor = vec4(color, alpha);
}
```


0.0

U

1.0



```
gl_FragColor = vec4(vec3(vUv.x), 1.0);
```



```
gl_FragColor = vec4(vec3(vUv.y), 1.0);
```



Primary Branding by DIA Studio



Primary Branding by DIA Studio



time to code !



