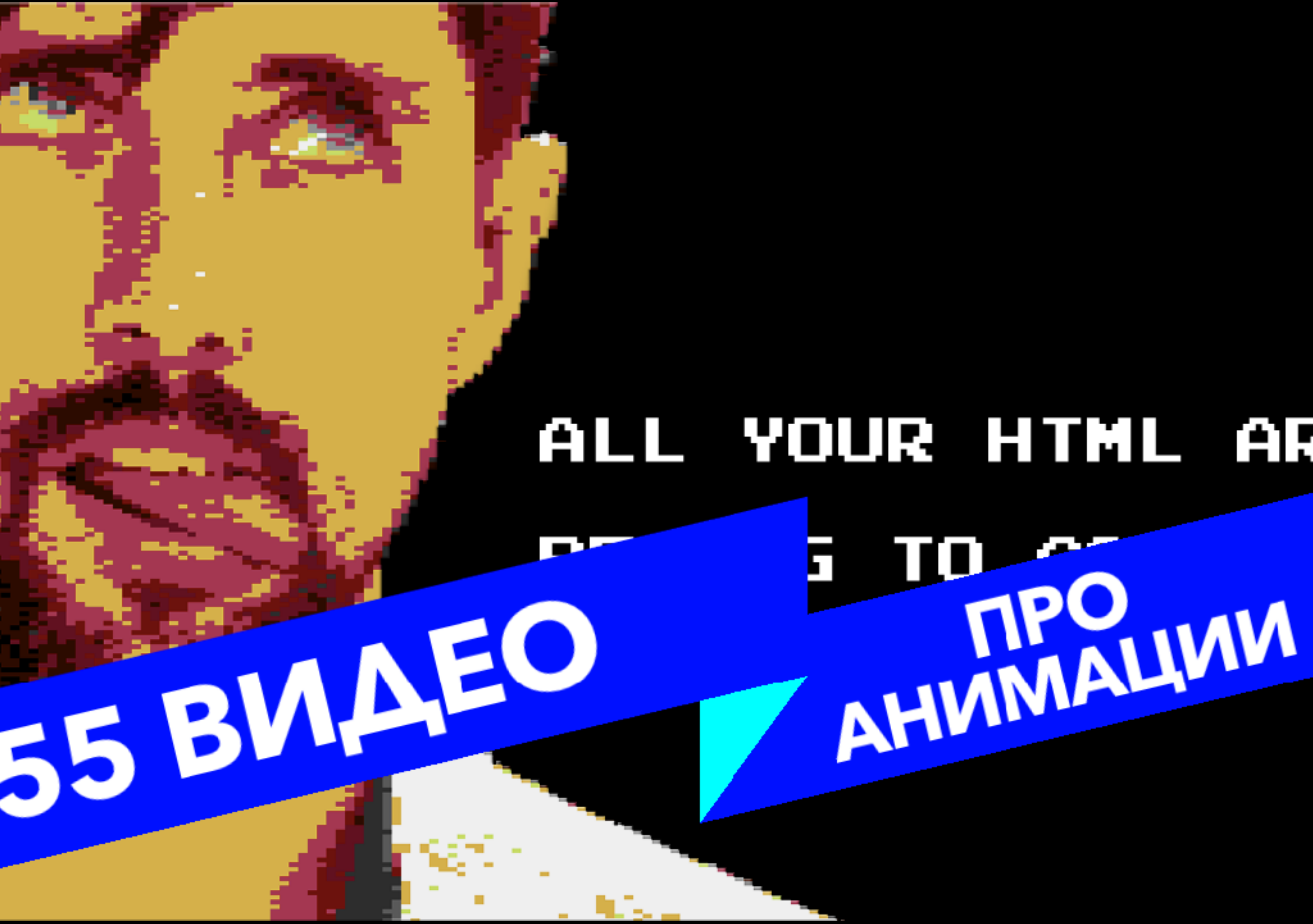
Z

Одна простая анимация

Юра

Верстка



55 ВИДЕО

ПРО АНИМАЦИИ

- Занимаюсь версткой
- Не профи в WebGL
- Просто нравится учиться
- Люблю арбузы и бачату

255
W
285
300
NW
330
350
N
15
30
NE
60
75
E
29 ALIVE
AUTO
13
122
hrc-23 -4 [illegible] 1-0258921

“А можешь сделать паутинку, как будто

за запотевшим стеклом?”

–Дизайнер

marketing and fundraising provider

We have first-hand expertise gained since the crypto industry inception, significant trust within the community, and profound understanding of both the crypto market and its actors.

— Это сложно?

Не знаю как рисовать линию

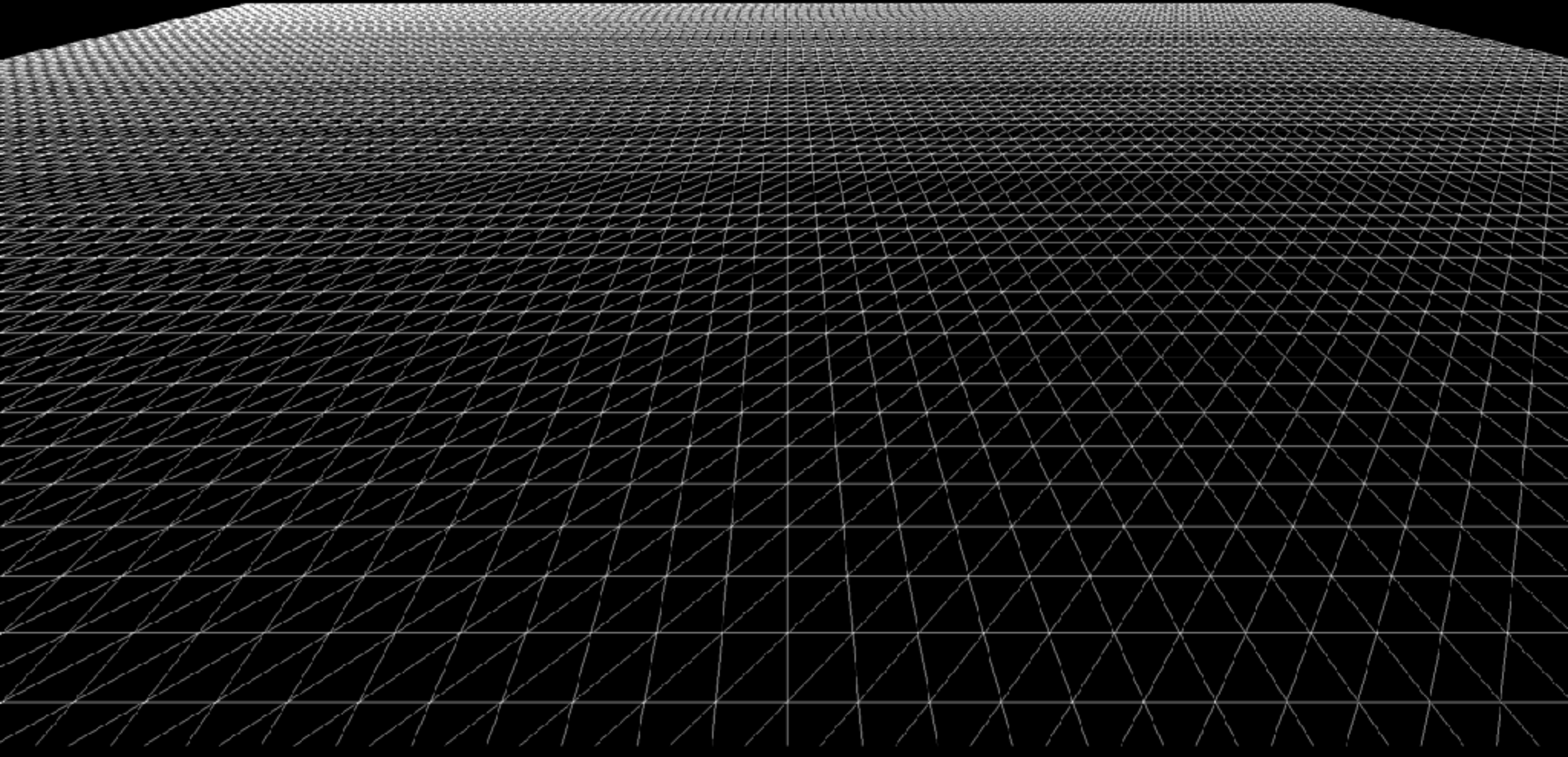
Не знаю как рисовать такой грид

Как, вообще сделать, чтобы оно не тупило?

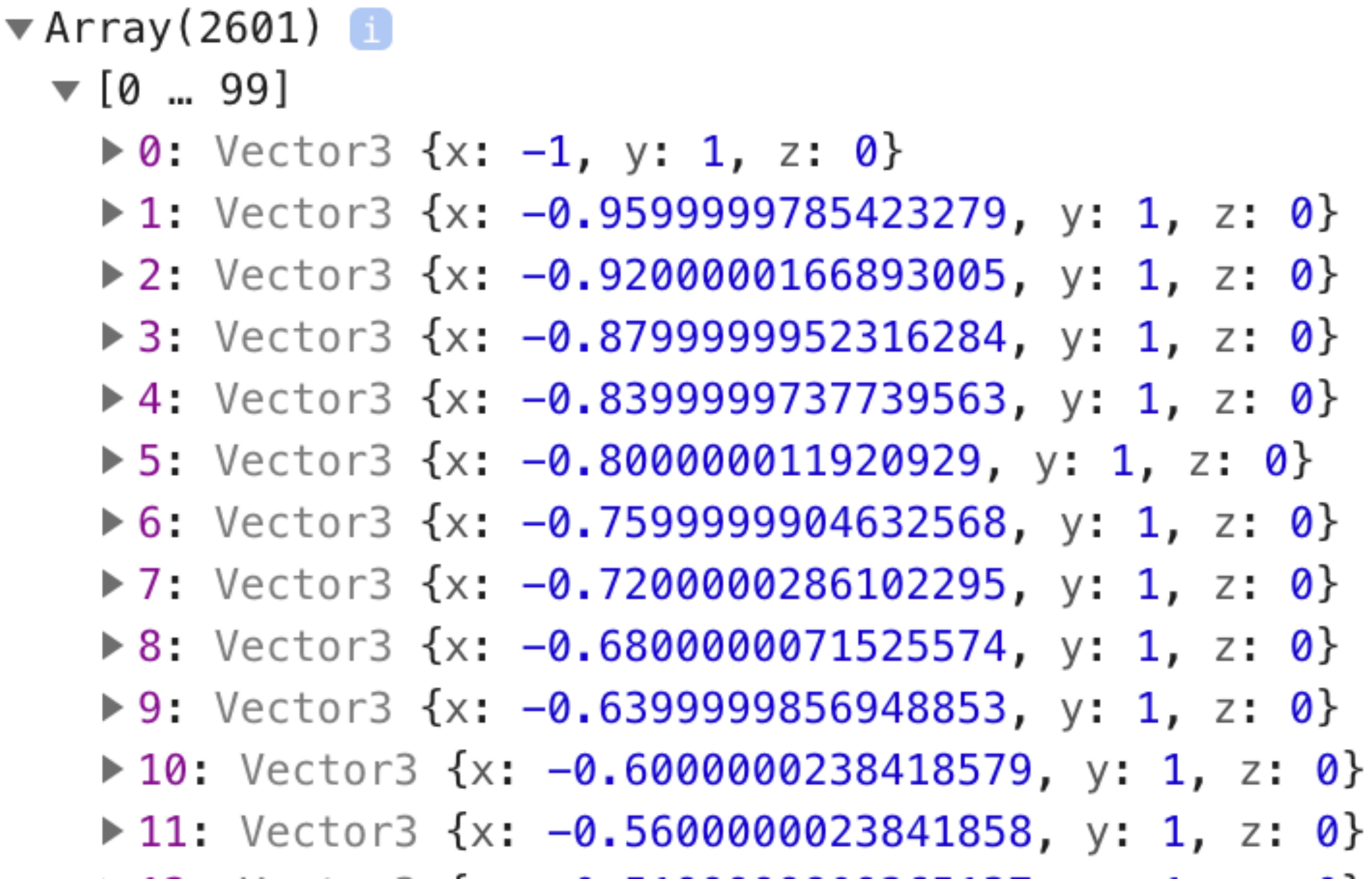
— Та, несложно, сделаем.

THREE.JS

PlaneGeometry

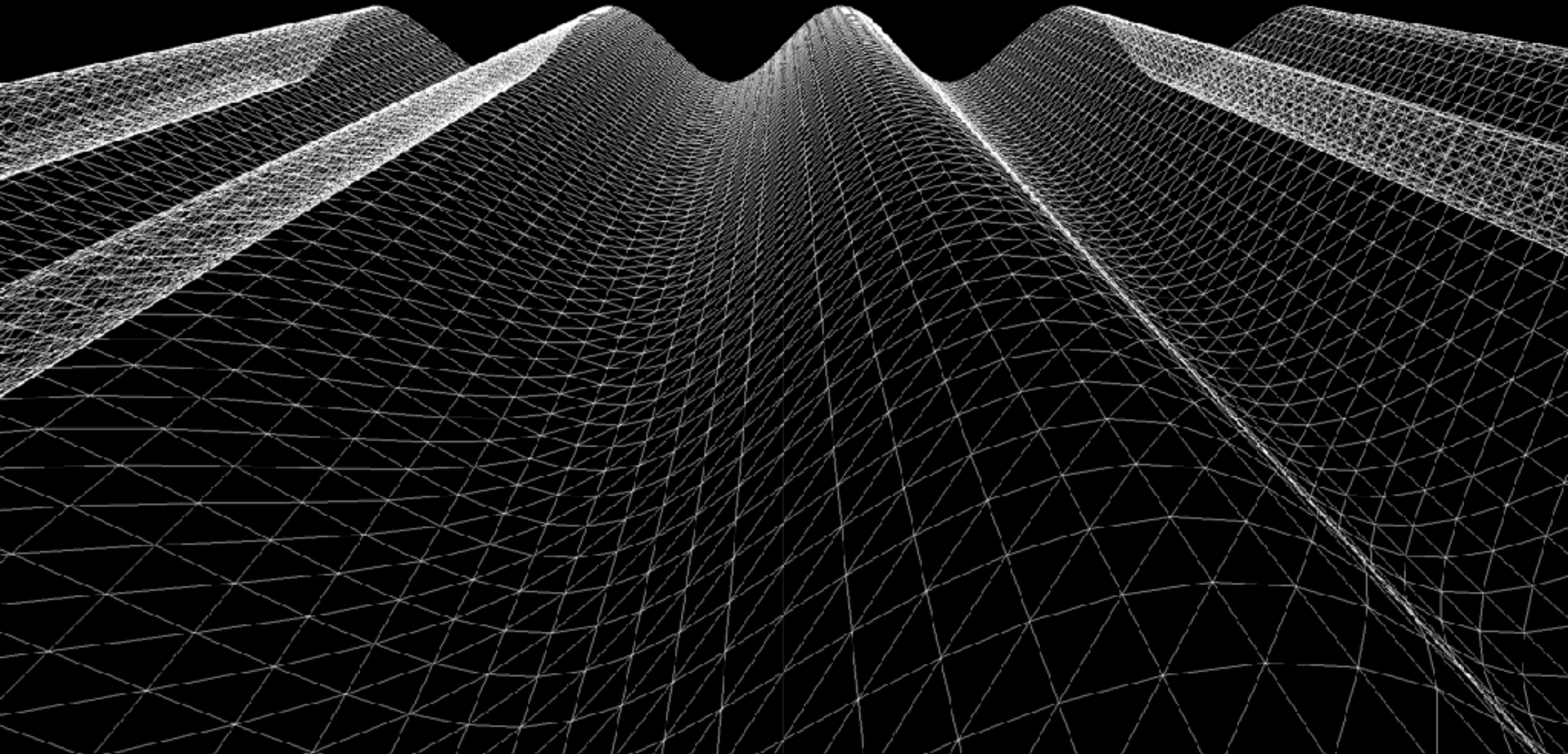


geometry.vertices

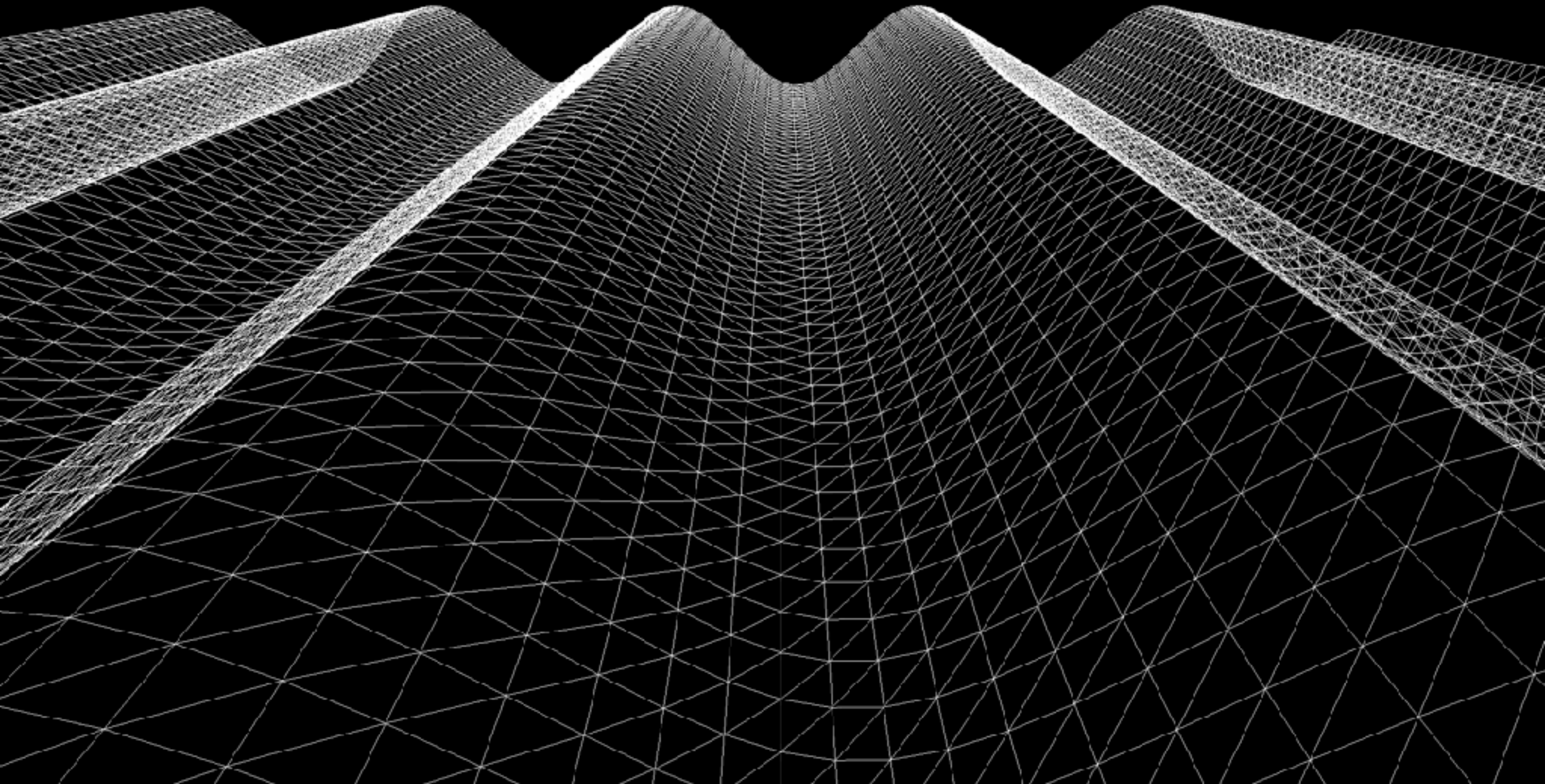


```
▼ Array(2601) i
  ▼ [0 … 99]
    ▶ 0: Vector3 {x: -1, y: 1, z: 0}
    ▶ 1: Vector3 {x: -0.9599999785423279, y: 1, z: 0}
    ▶ 2: Vector3 {x: -0.9200000166893005, y: 1, z: 0}
    ▶ 3: Vector3 {x: -0.8799999952316284, y: 1, z: 0}
    ▶ 4: Vector3 {x: -0.8399999737739563, y: 1, z: 0}
    ▶ 5: Vector3 {x: -0.800000011920929, y: 1, z: 0}
    ▶ 6: Vector3 {x: -0.7599999904632568, y: 1, z: 0}
    ▶ 7: Vector3 {x: -0.7200000286102295, y: 1, z: 0}
    ▶ 8: Vector3 {x: -0.6800000071525574, y: 1, z: 0}
    ▶ 9: Vector3 {x: -0.6399999856948853, y: 1, z: 0}
    ▶ 10: Vector3 {x: -0.6000000238418579, y: 1, z: 0}
    ▶ 11: Vector3 {x: -0.5600000023841858, y: 1, z: 0}
```

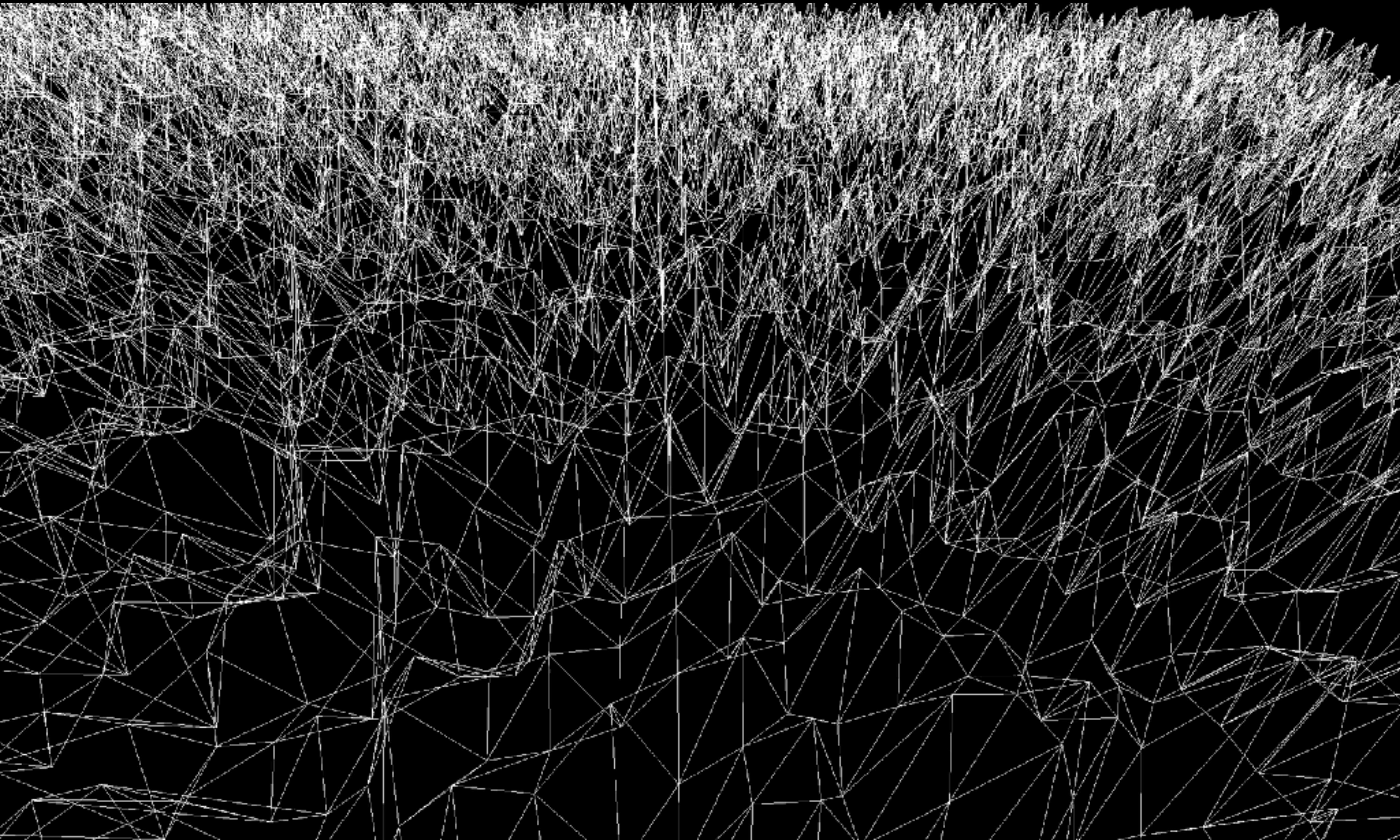
$$z = \sin(x)$$

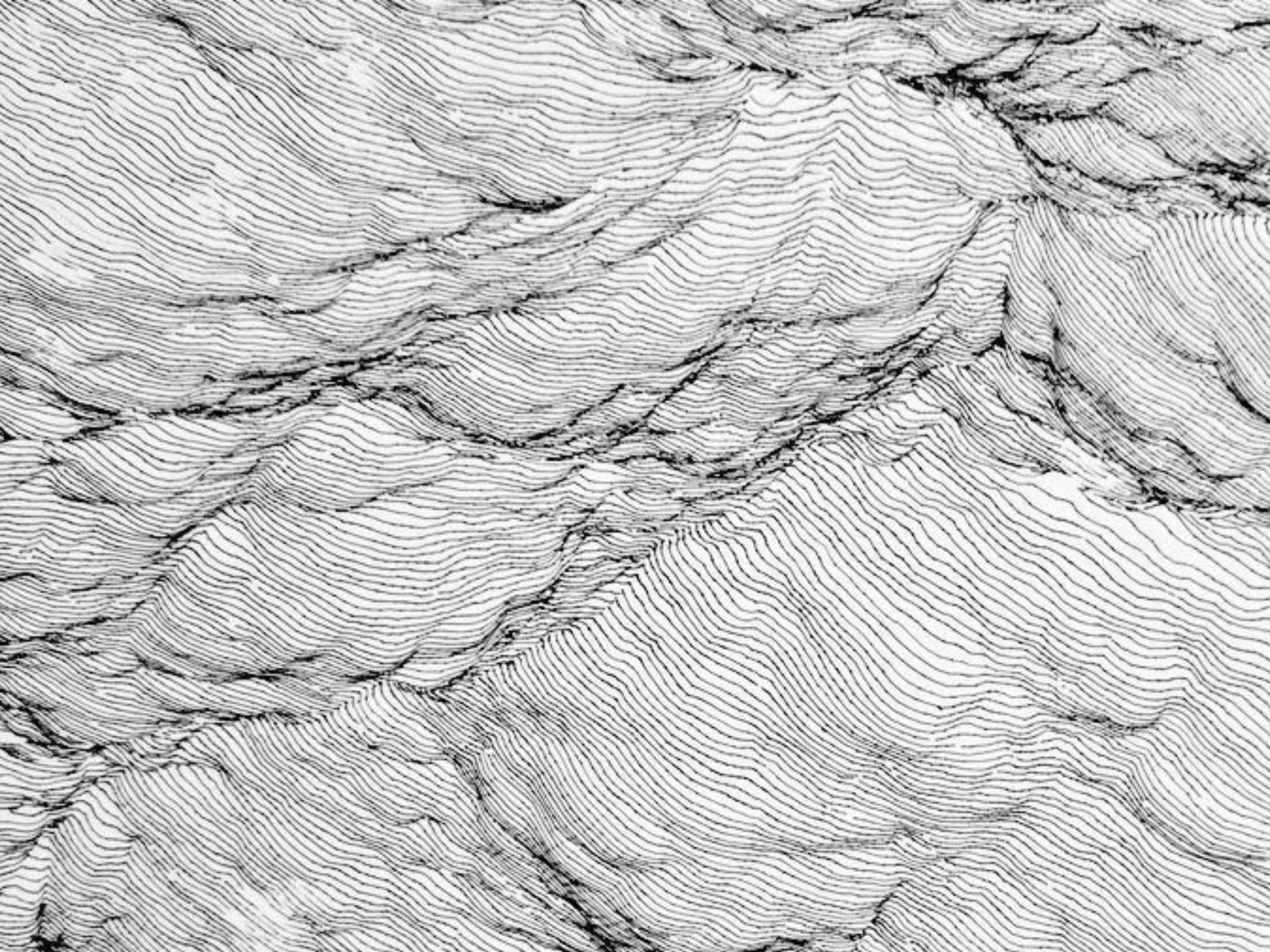


$$z = \sin(x + time)$$

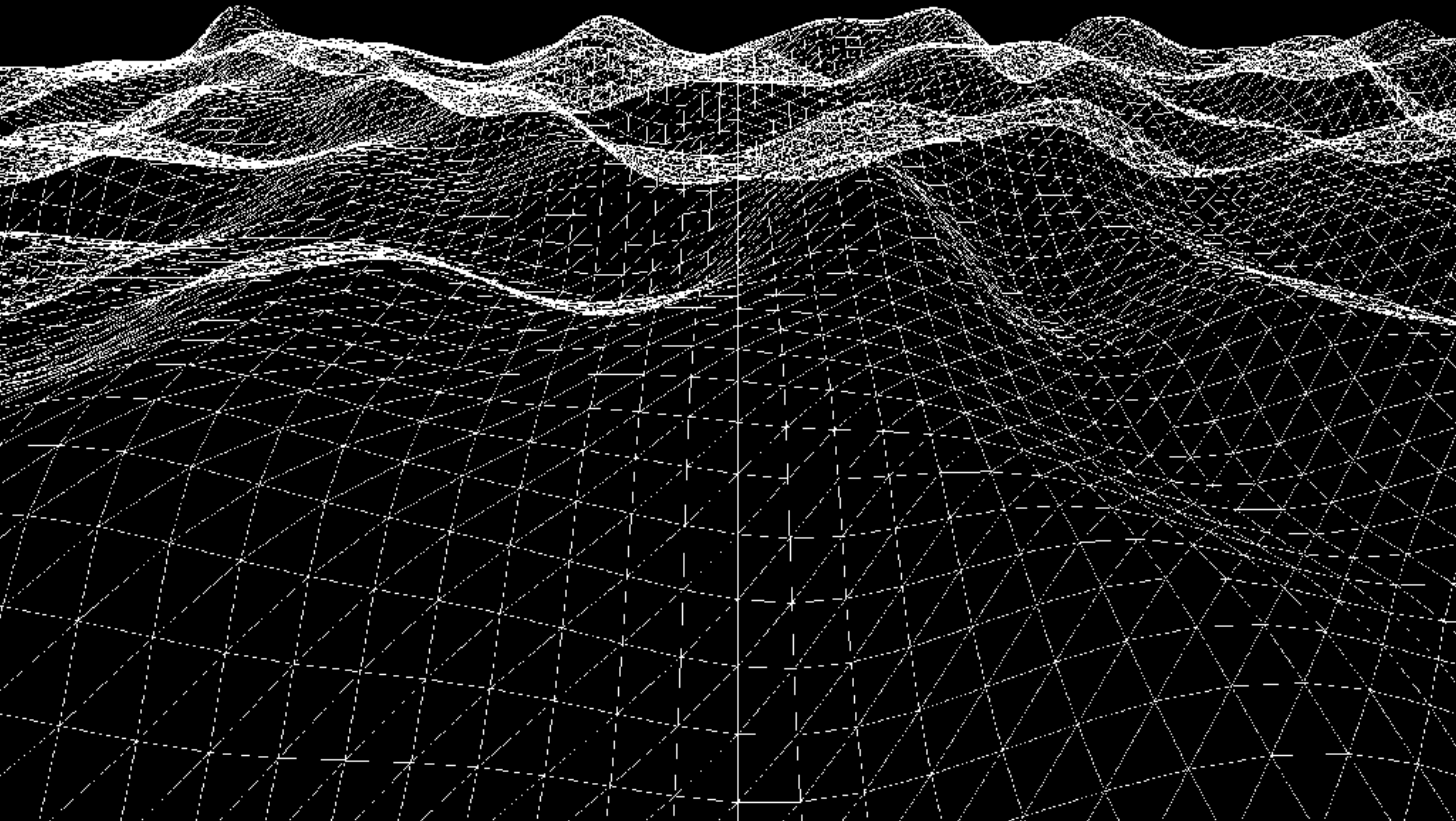


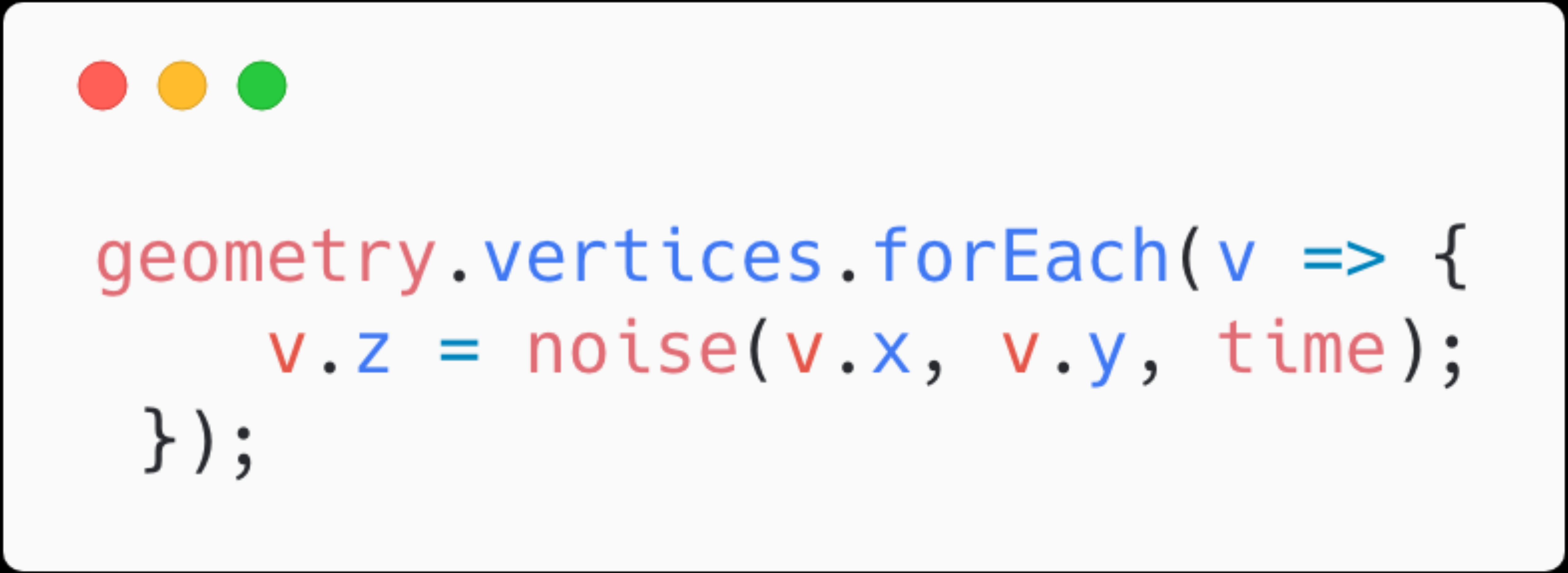
z = Math.random()





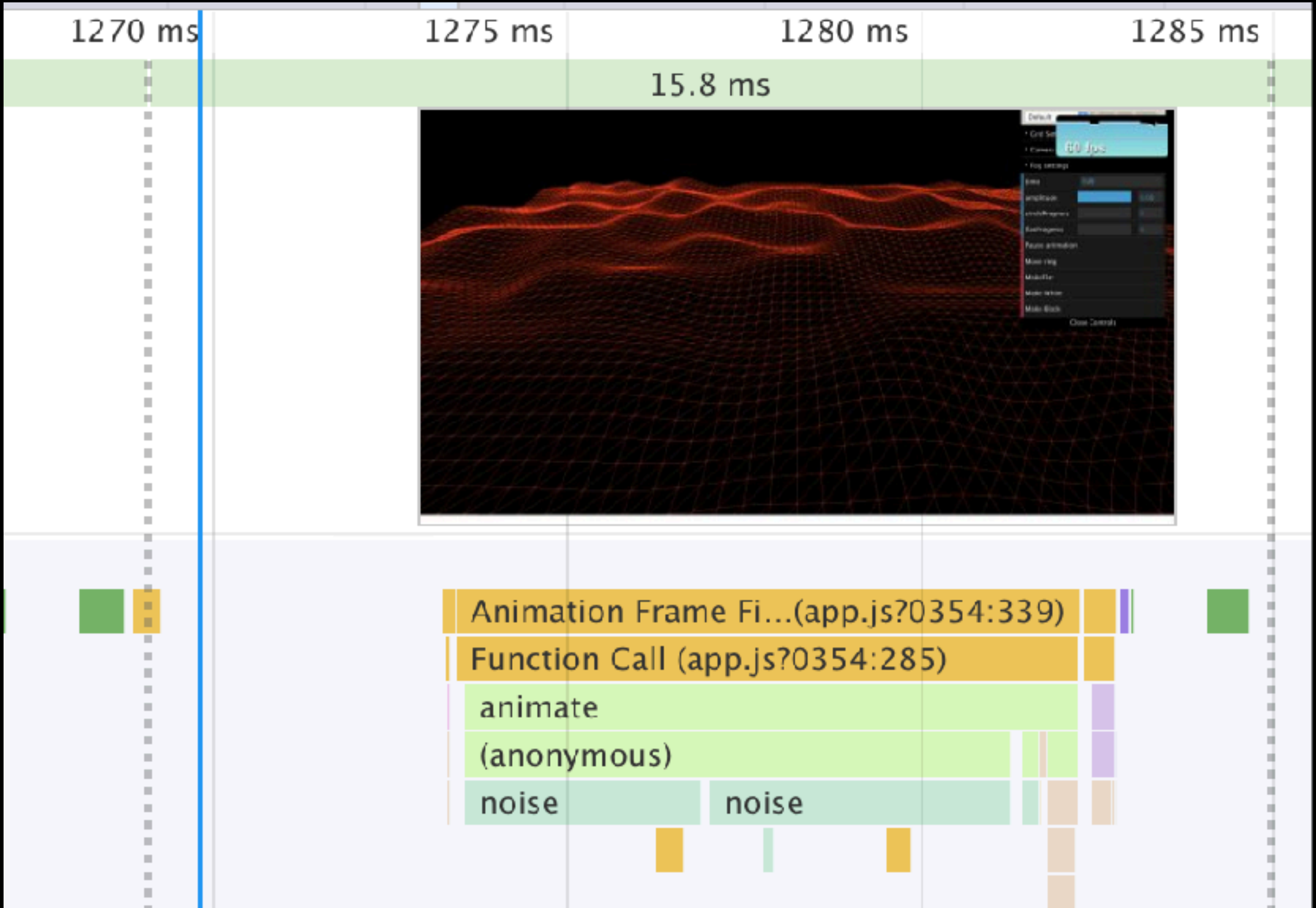
$z = noise(x,y)$





```
geometry.vertices.forEach(v => {
    v.z = noise(v.x, v.y, time);
 });
```

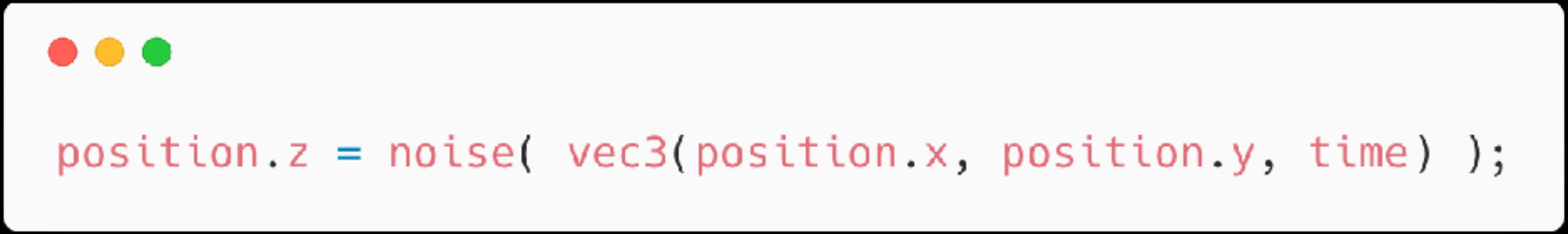
CPU


1270 ms
1275 ms
1280 ms
1285 ms
15.8 ms
60 fps
Animation Frame Fi…(app.js?0354:339)
Function Call (app.js?0354:285)
animate
(anonymous)
noise
noise

THREE.JS !== GPU

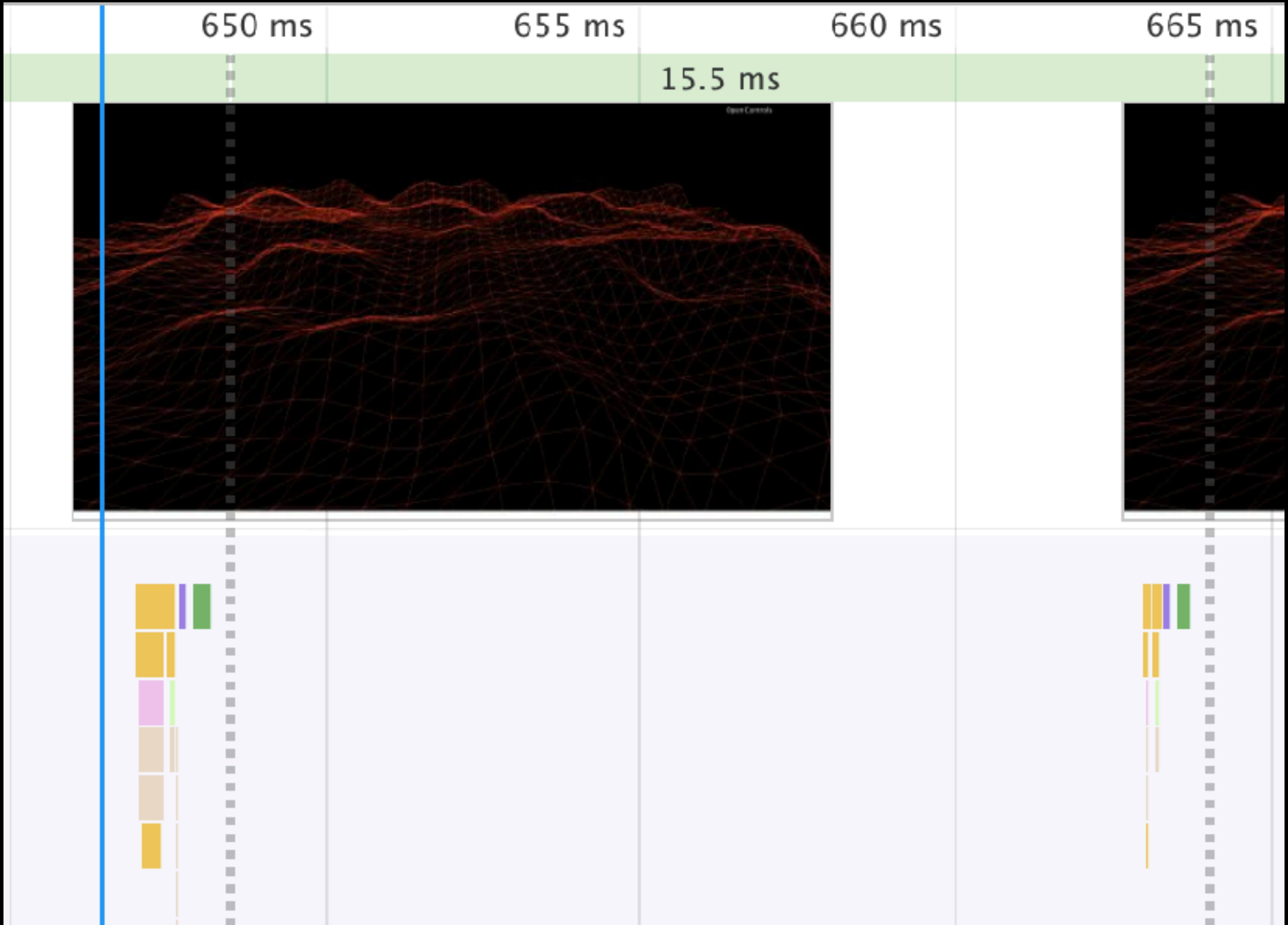
vertex shader

No loop.

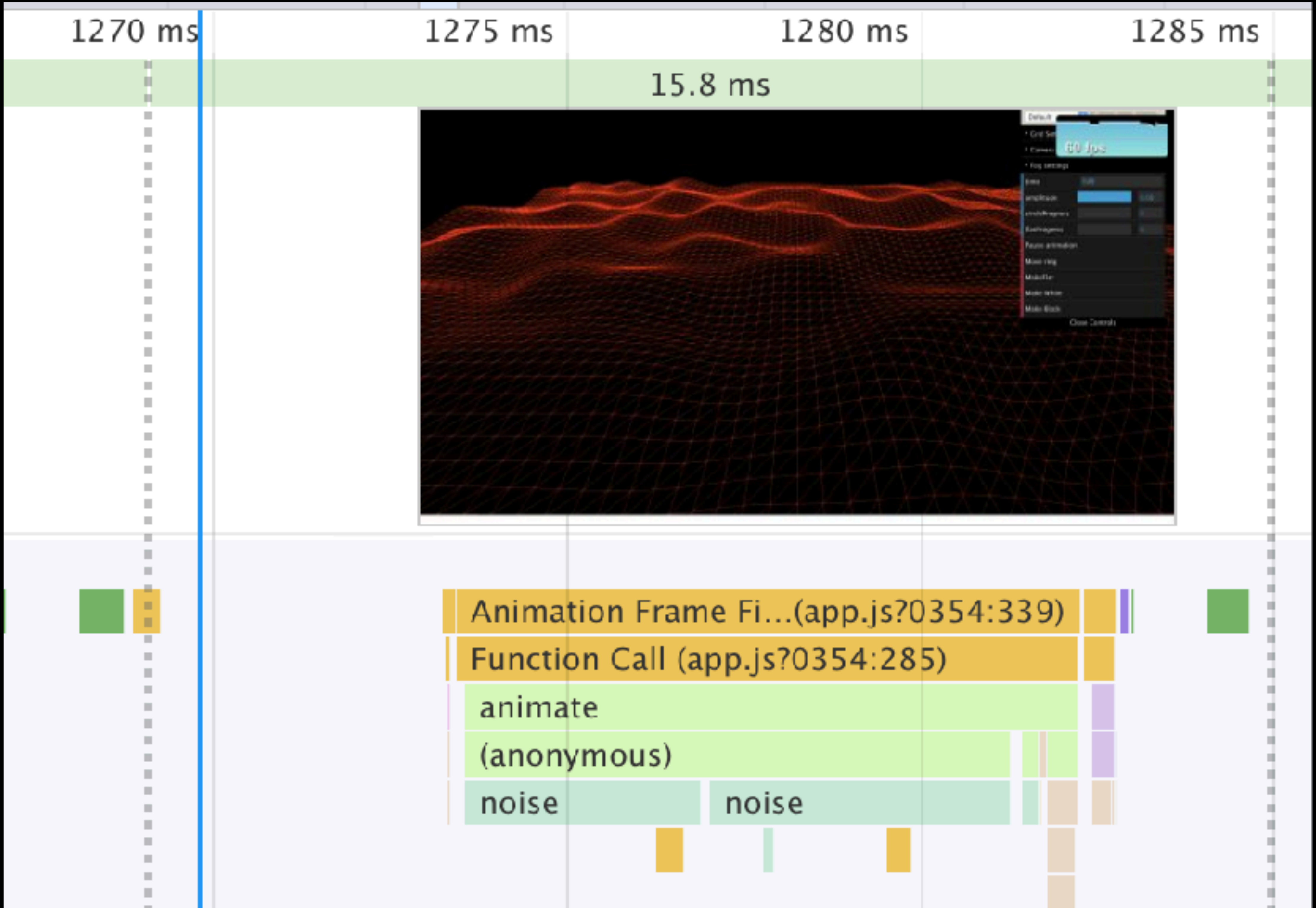


```
position.z = noise( vec3(position.x, position.y, time) );
```

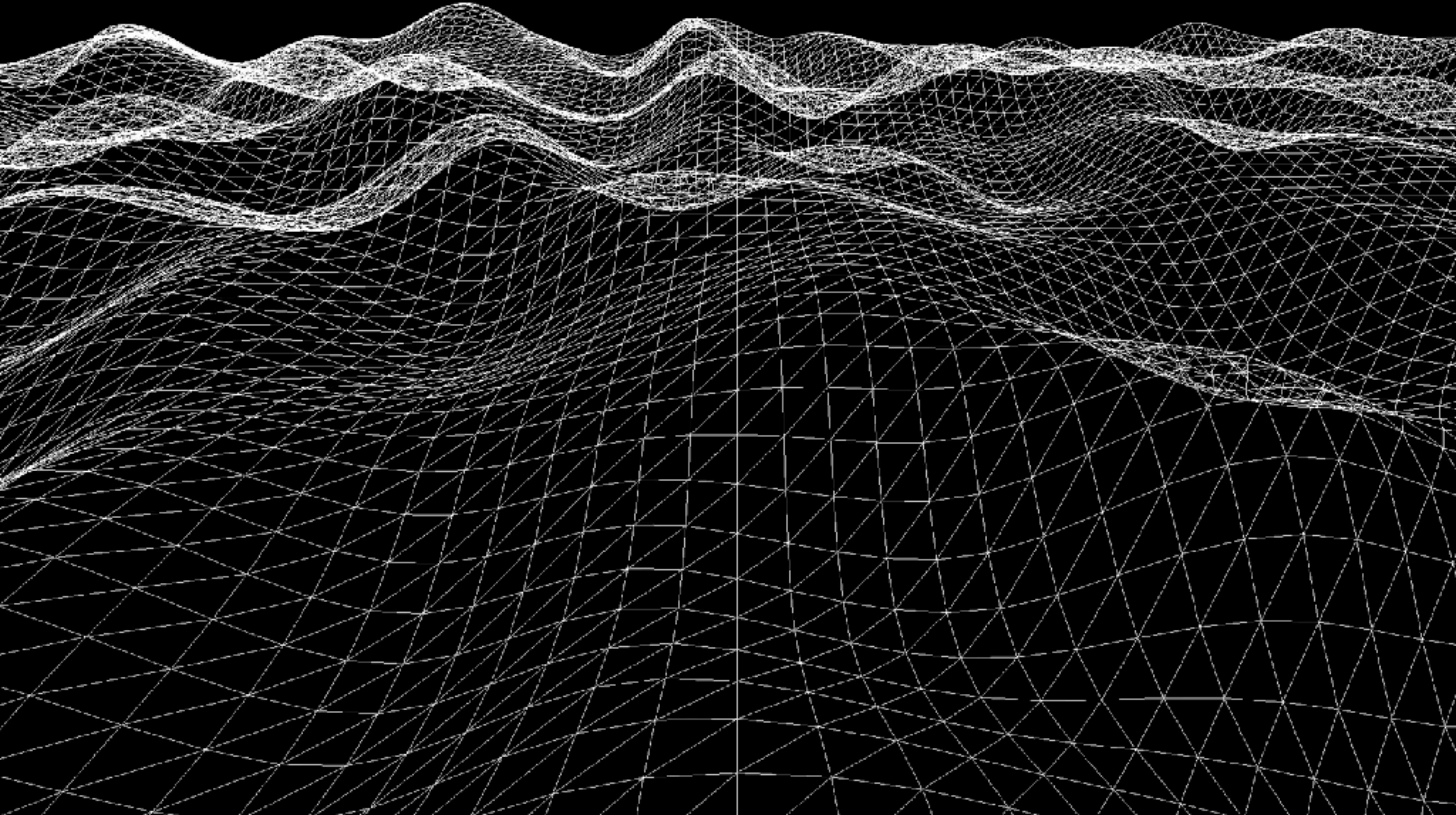
GLSL

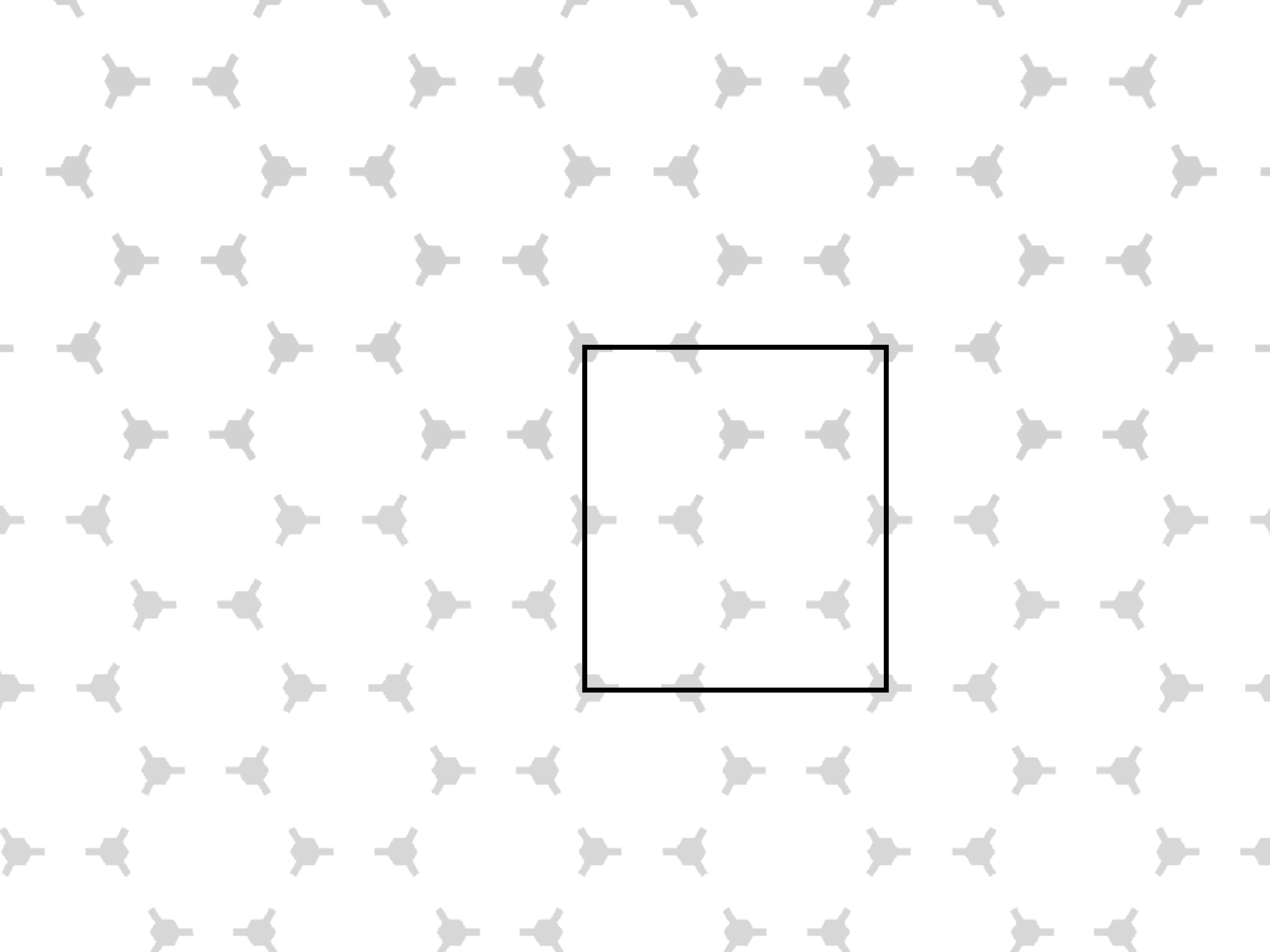


CPU

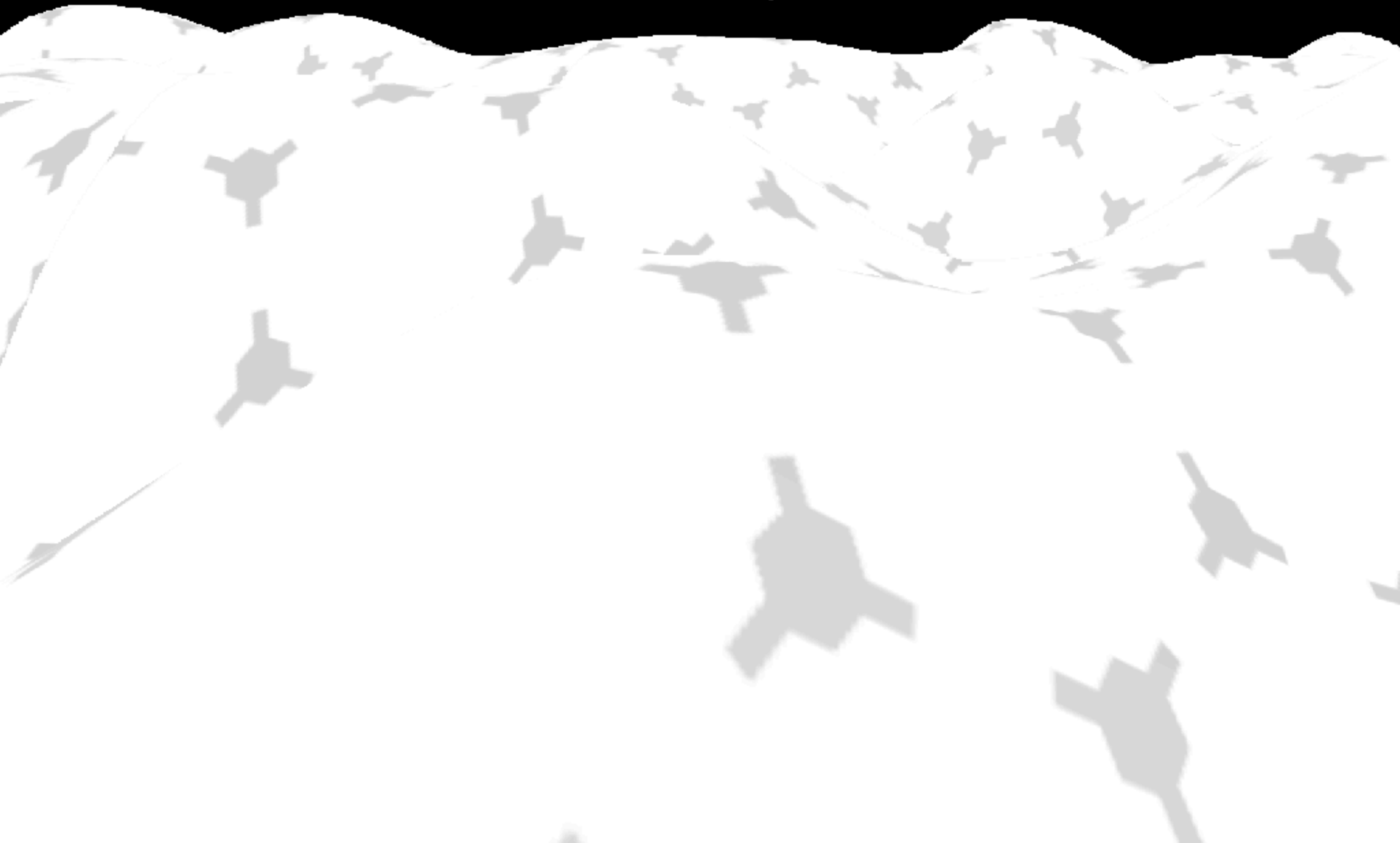

1270 ms
1275 ms
1280 ms
1285 ms
15.8 ms
Animation Frame Fi…(app.js?0354:339)
Function Call (app.js?0354:285)
animate
(anonymous)
noise
noise

Окей, получилось





Текстура



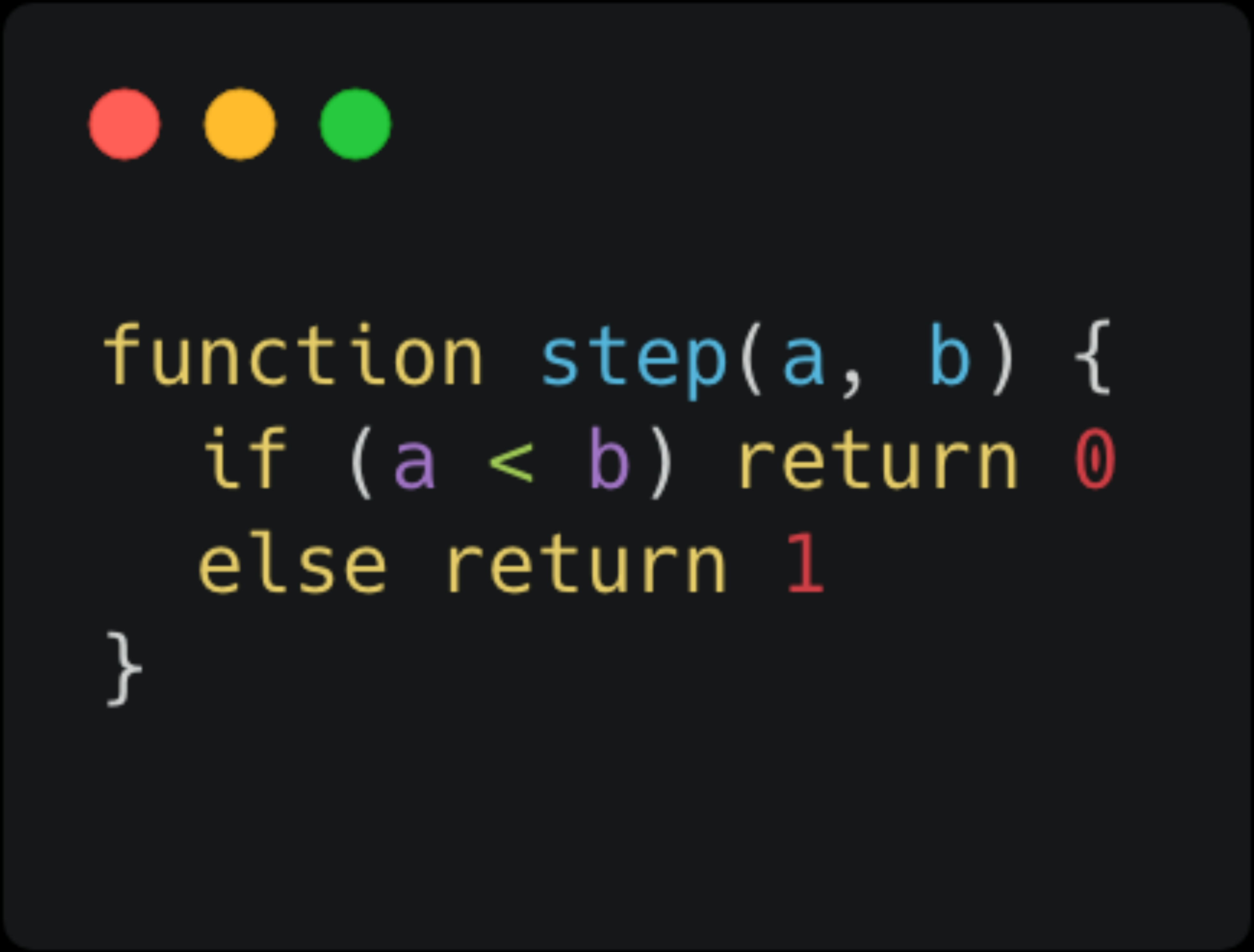
😭😭😭

“На GLSL можно сделать четенько”

–Кто-то в интернетах

fragment shader

step(a,b)

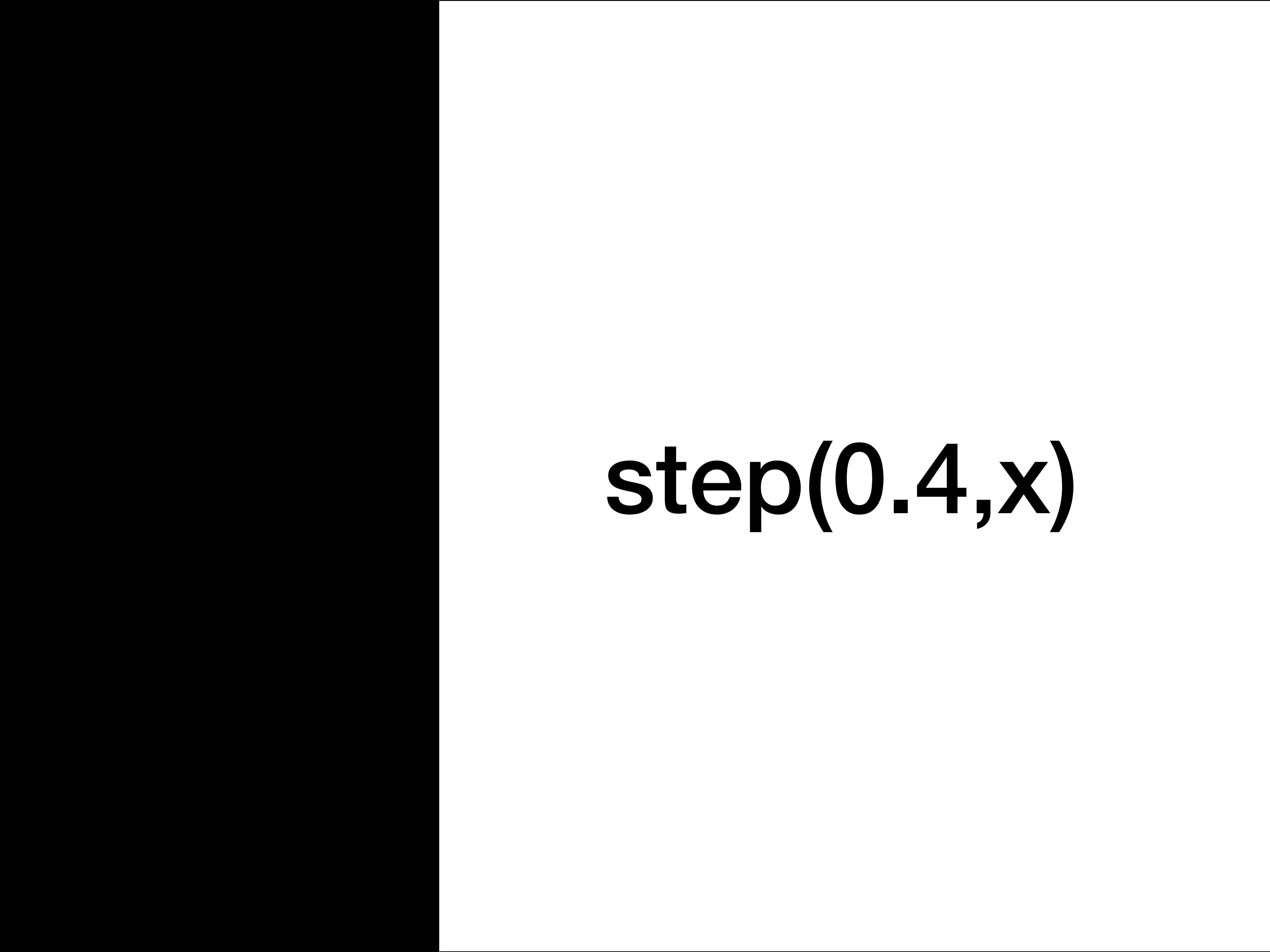


```
function step(a, b) {
  if (a < b) return 0
  else return 1
}
```

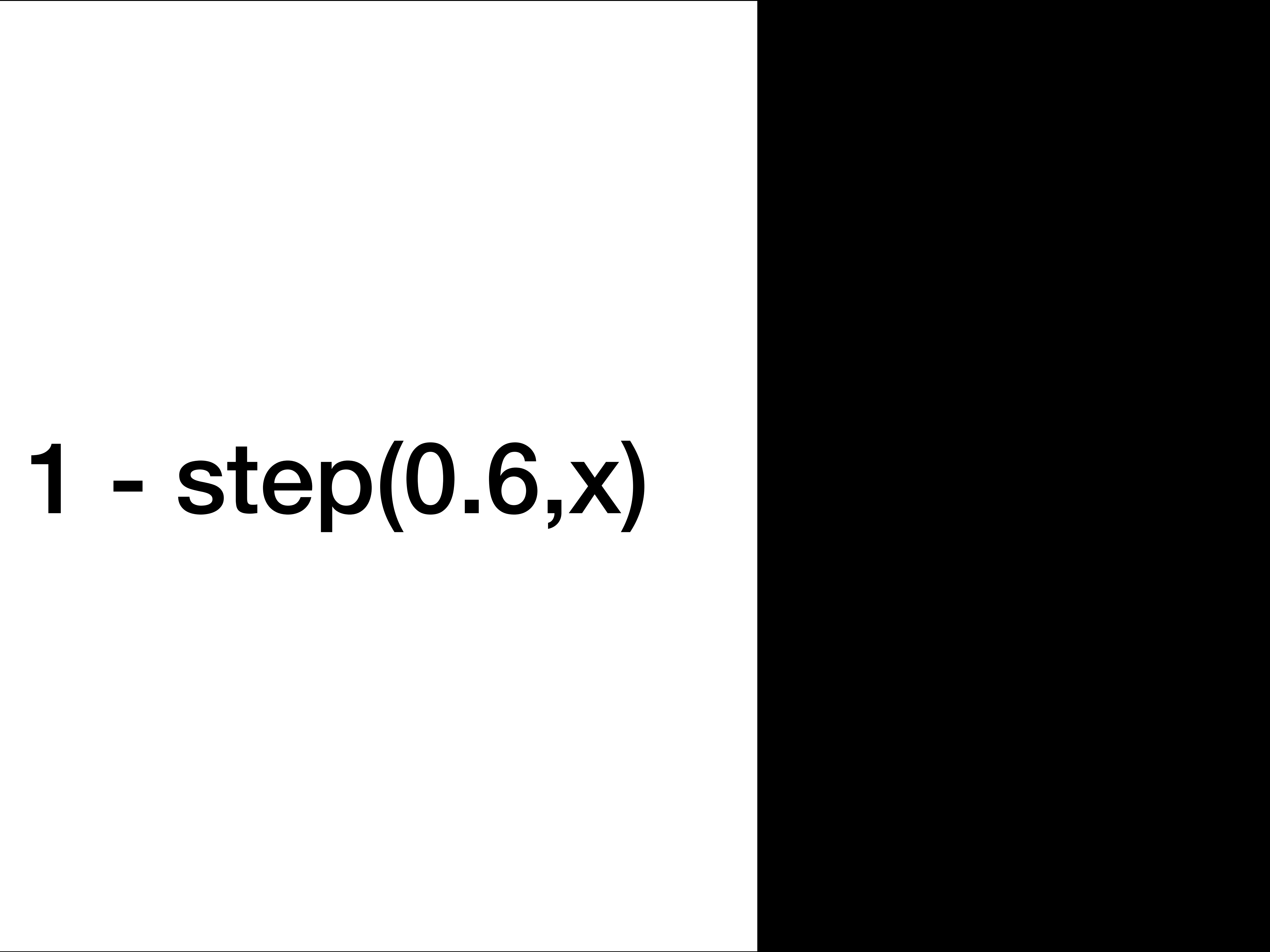
(0,0) **(1,0)**

step(a,b)

(0,1) **(1,1)**



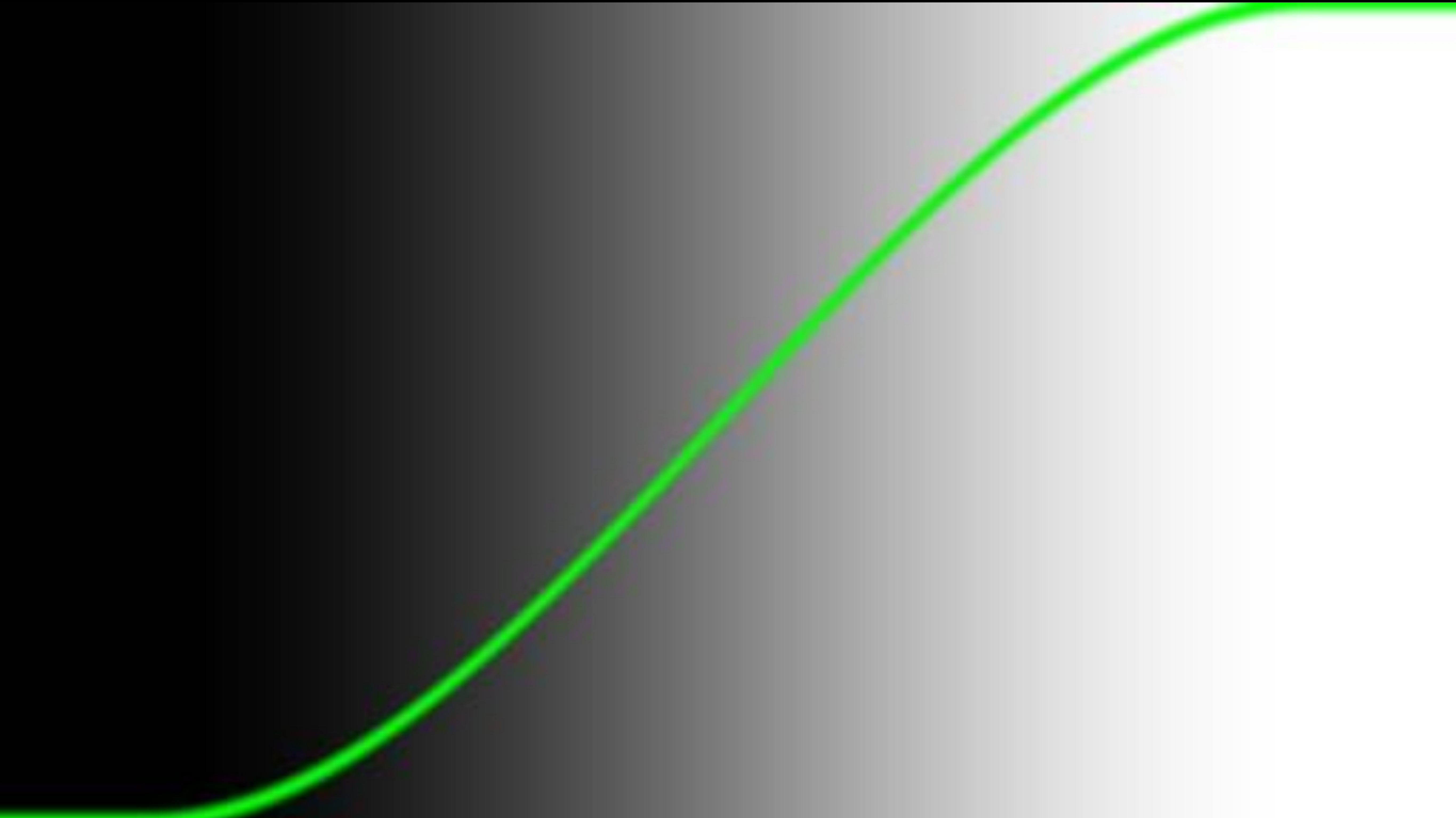
step(0.4,x)



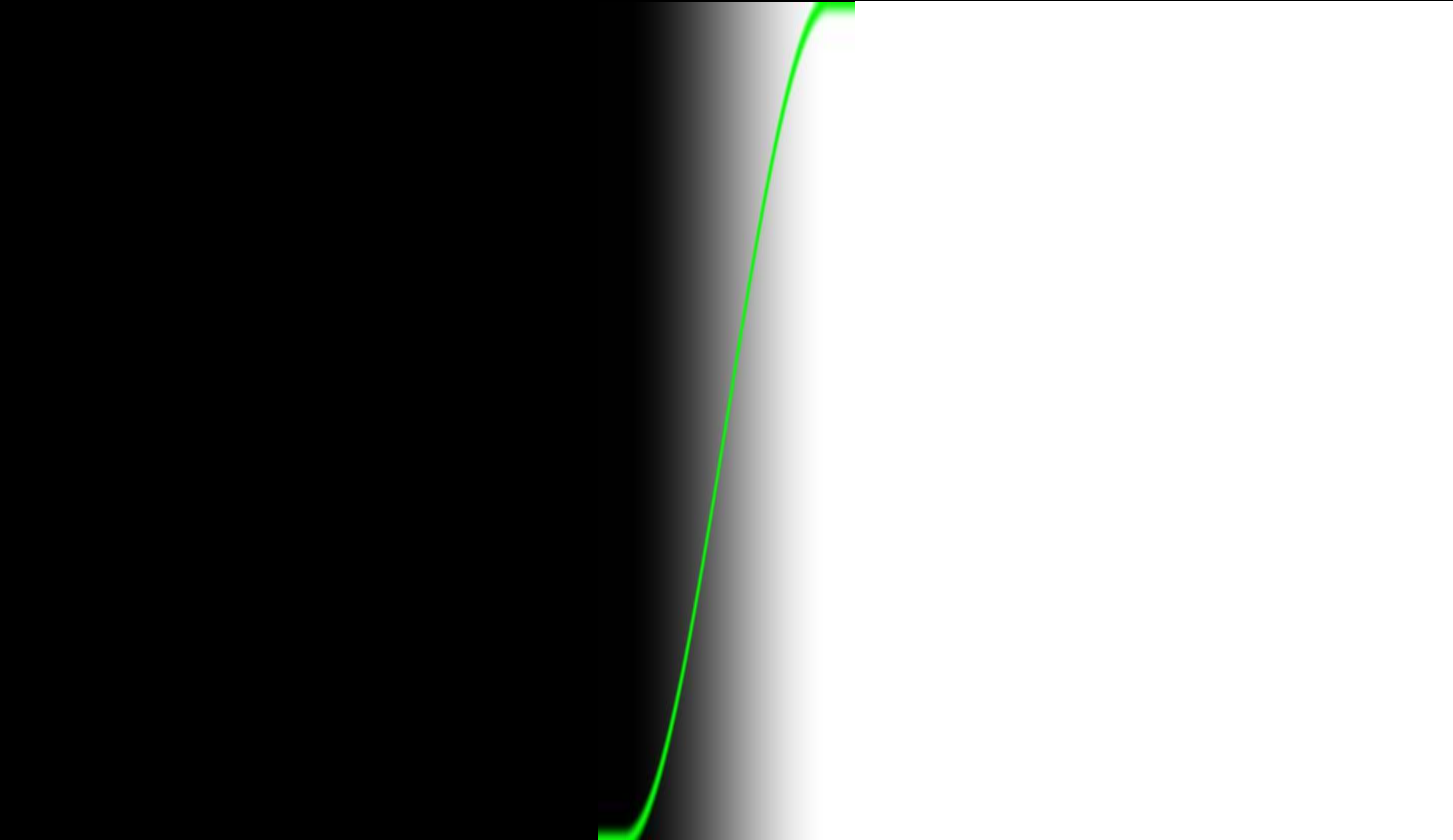
1 - step(0.6,x)

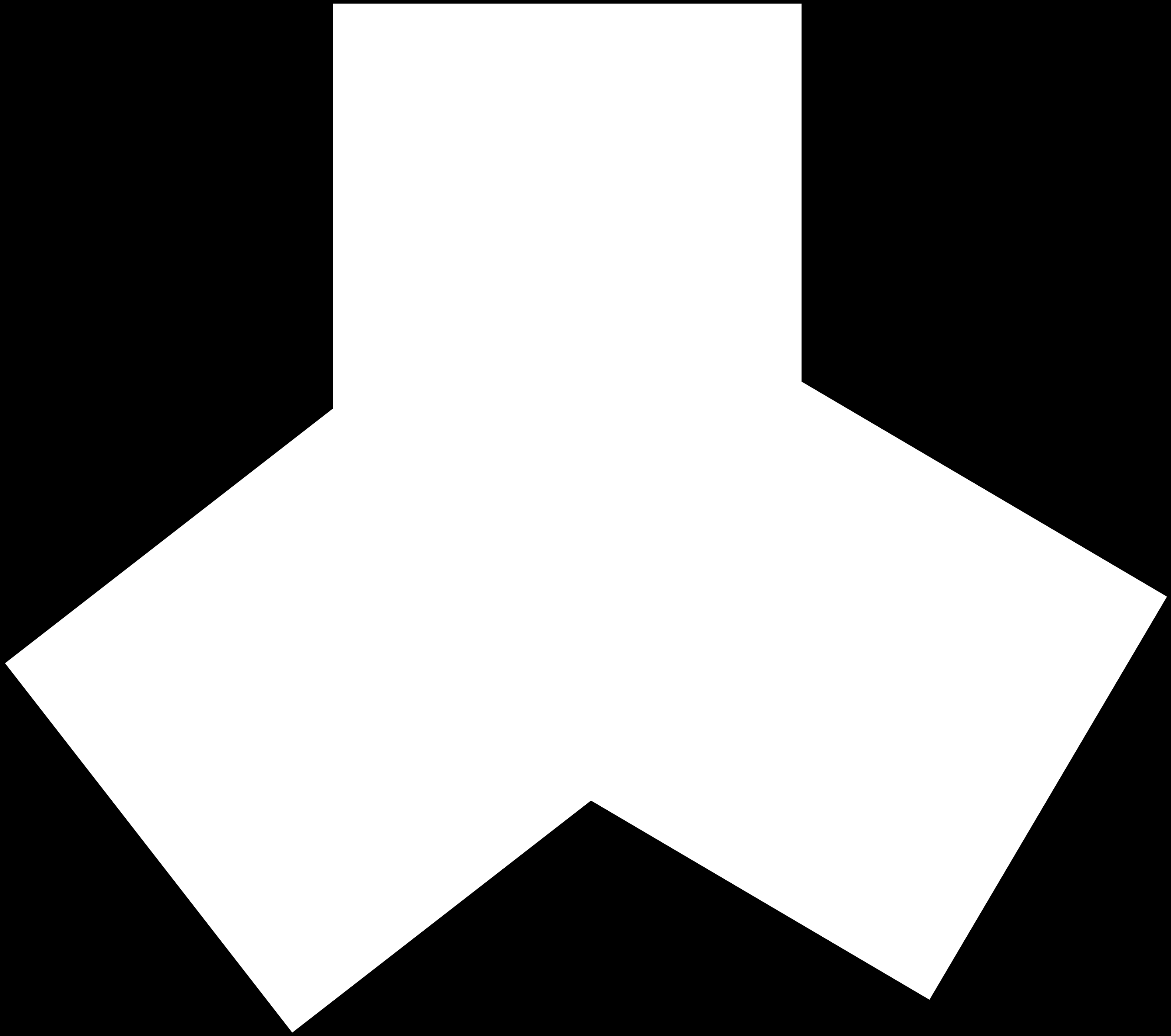
😭😭😭😭😭

*smooth*step

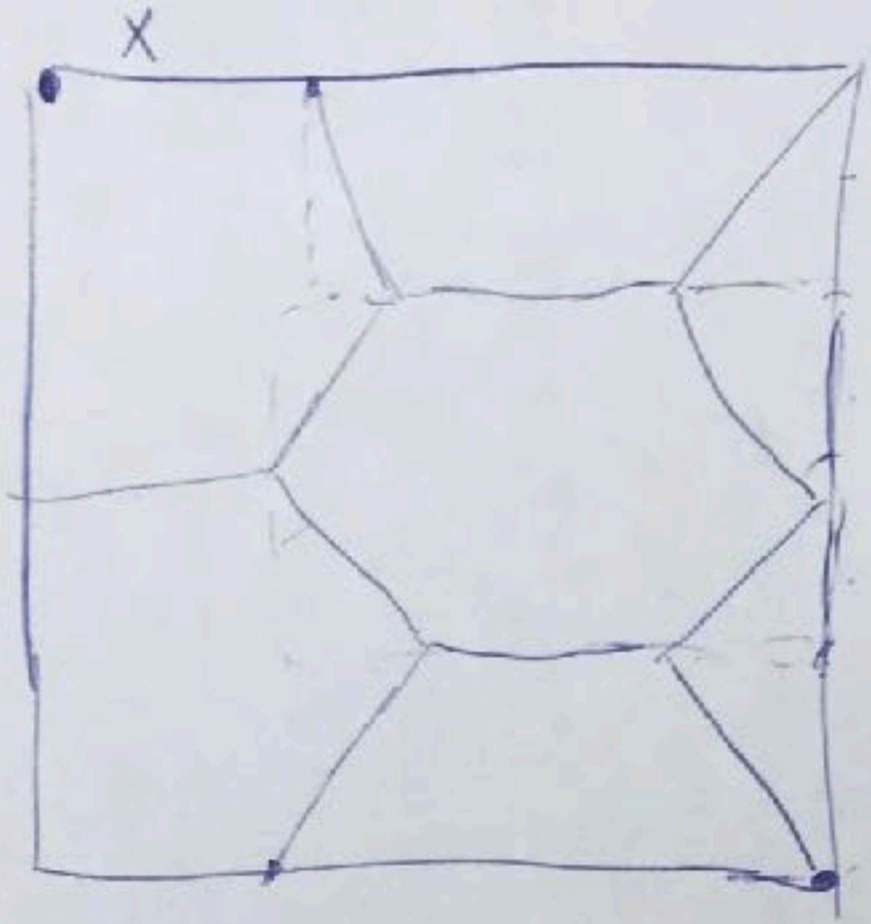


*smooth*step





$3x$



$$\frac{2\sqrt{3}}{3} \qquad \frac{2}{\sqrt{3}}$$

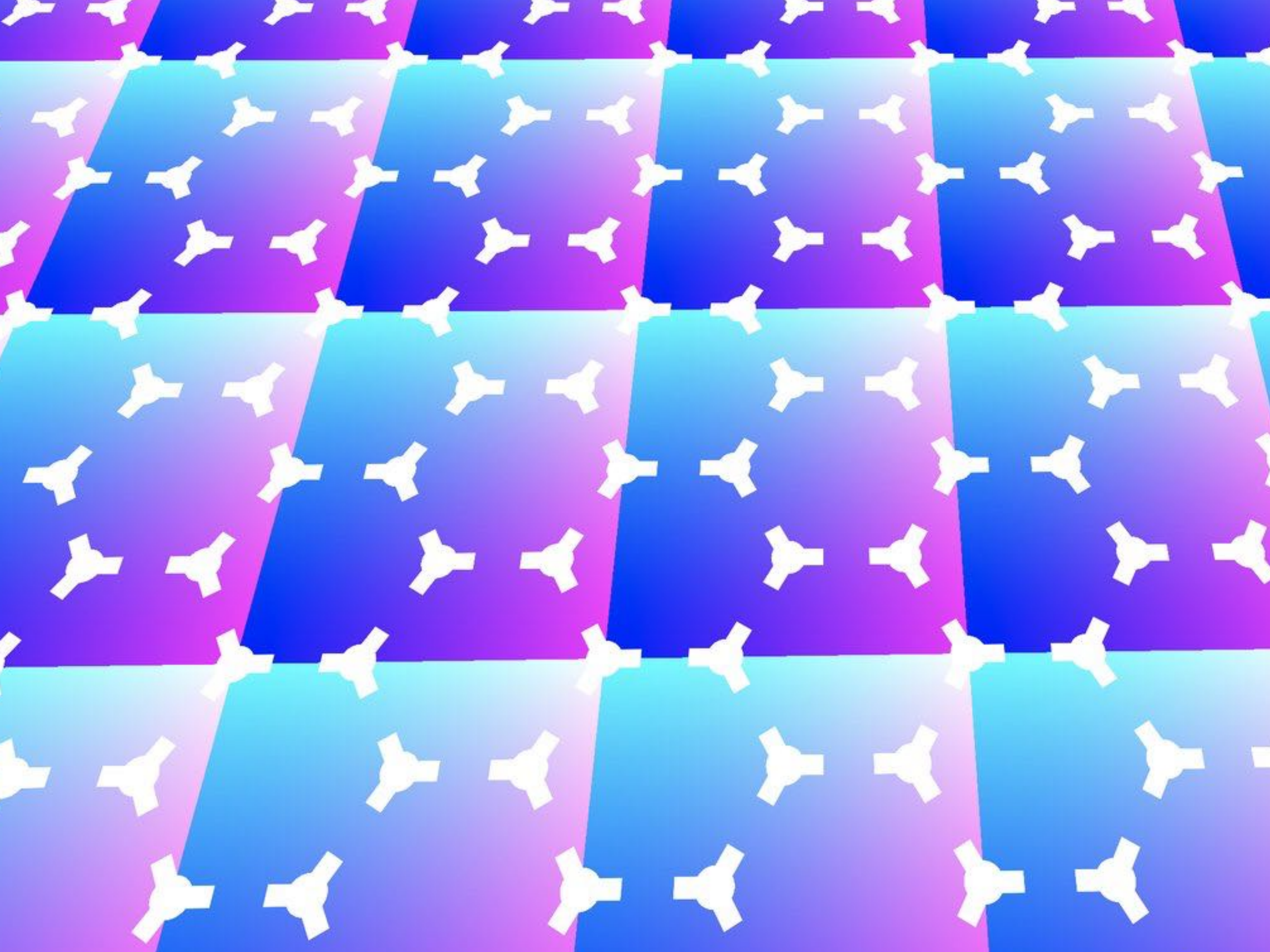
Как нарисовать сову

1.

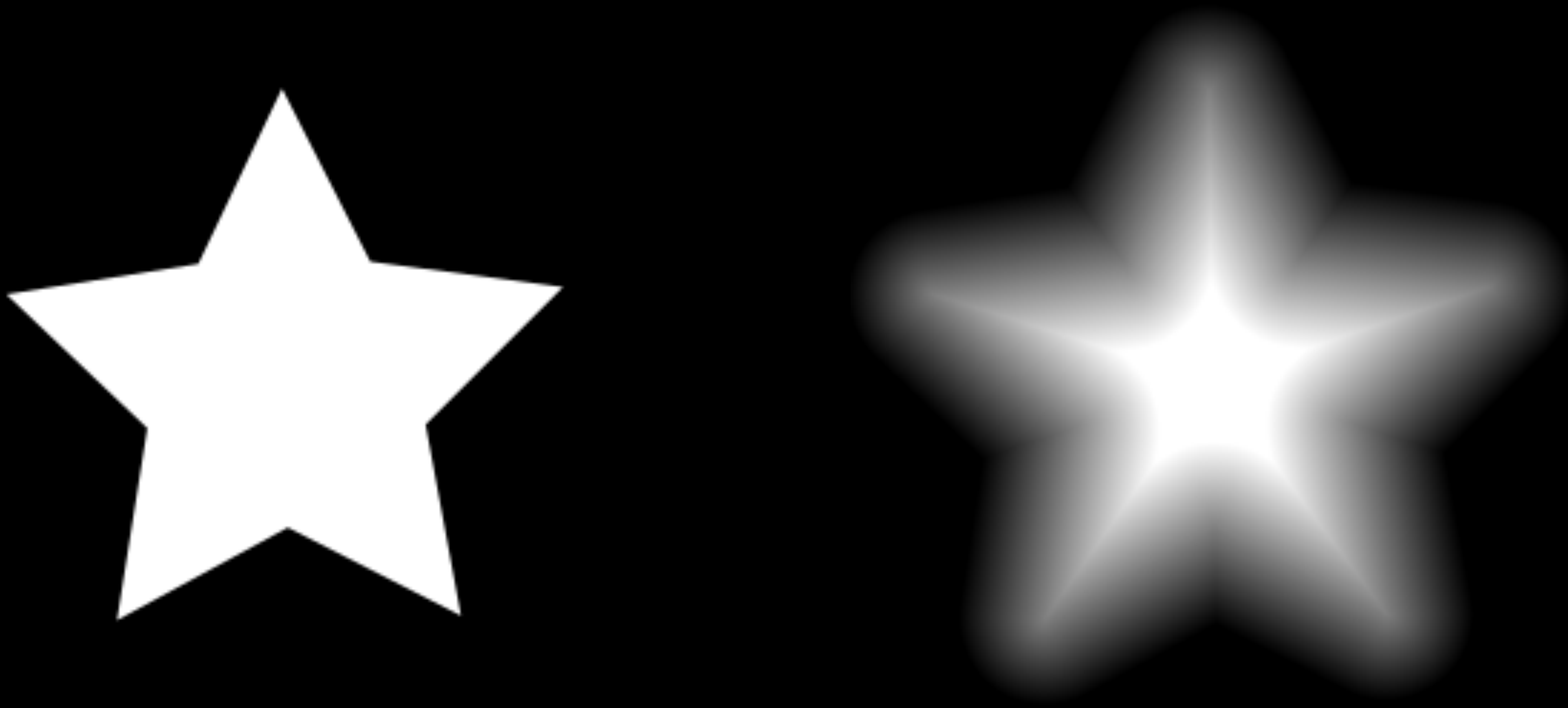
2.

1. Рисуем кружочки

2. Рисуем остаток совы



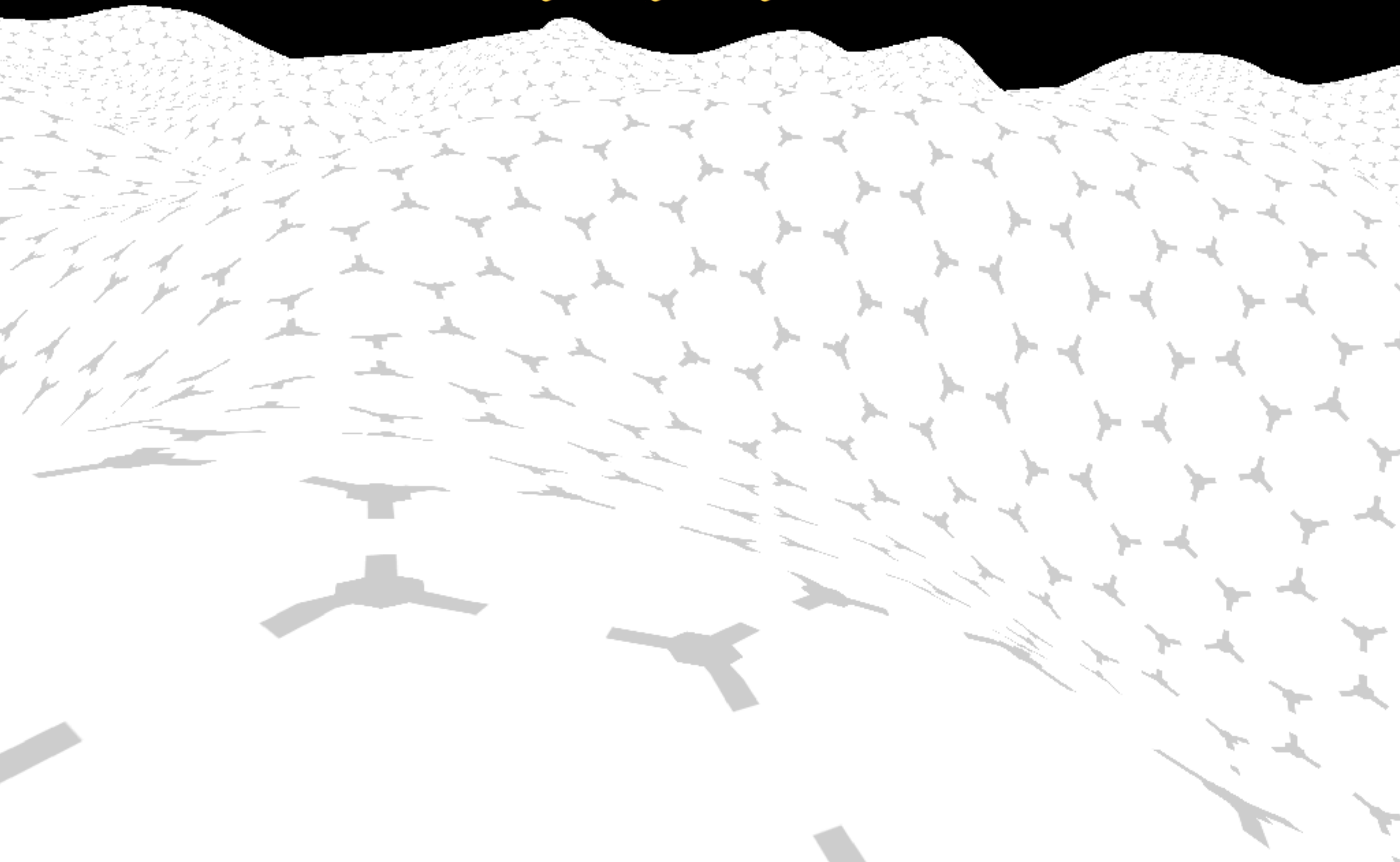
SDF



j] - G T 0 5 p f i e
) O H U 2 ? q 1 l u
% X N Z 6 L k ! s n
(Q M Y B 8 b h ^ z
} S # A R P 9 d c +
{ C g \ V D E y . : w
< ; >

128x128px 🟦

🎉🎉🎉



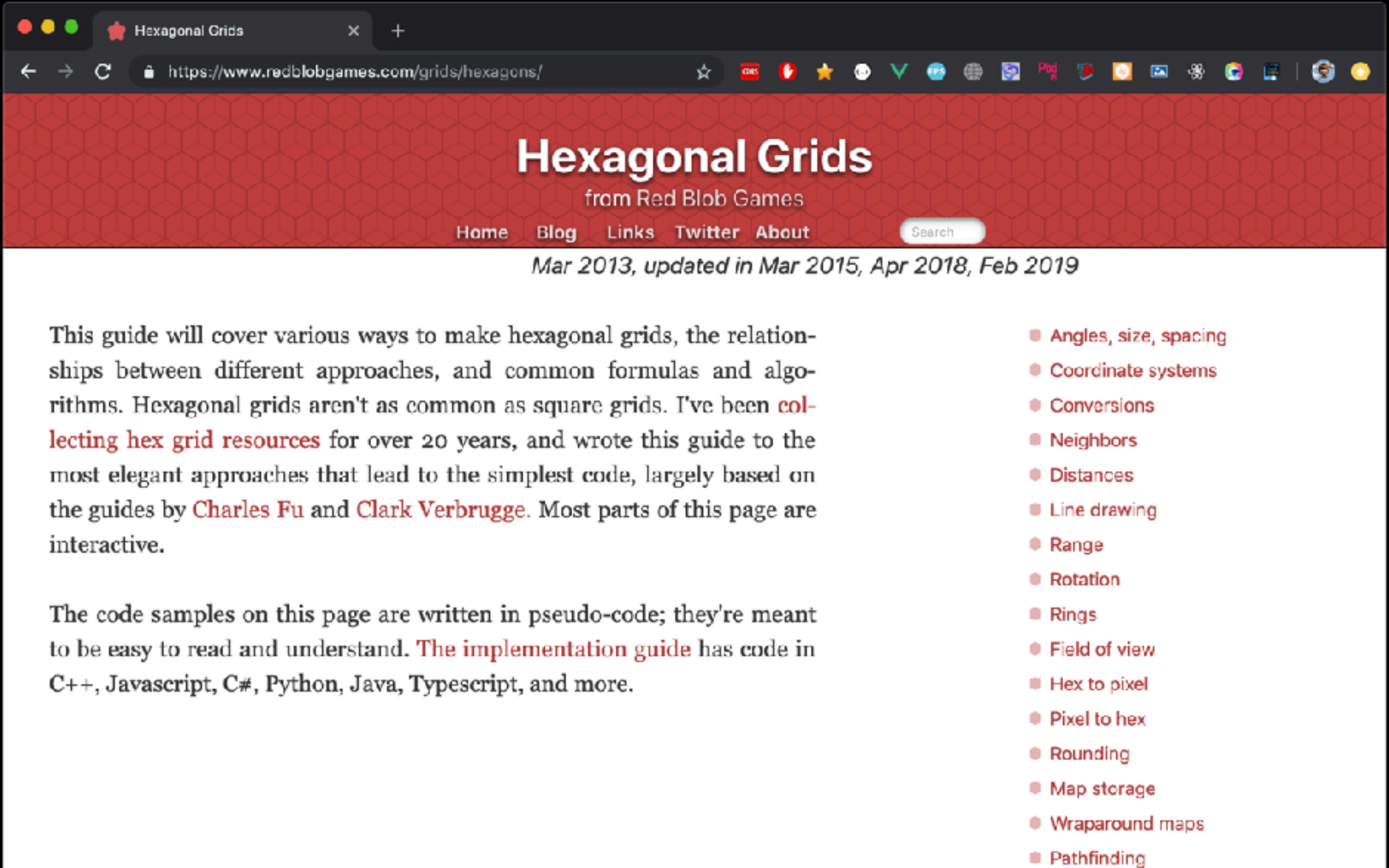
*“А еще, пусть он двигает мышью, и путь новый прокладывается.
А соты подсвечиваются.”*

– ❤

marketing and fundraising provider

We have first-hand expertise gained since the crypto industry inception, significant trust within the community, and profound understanding of both the crypto market and its actors.

Статья с материалами за 20 лет!



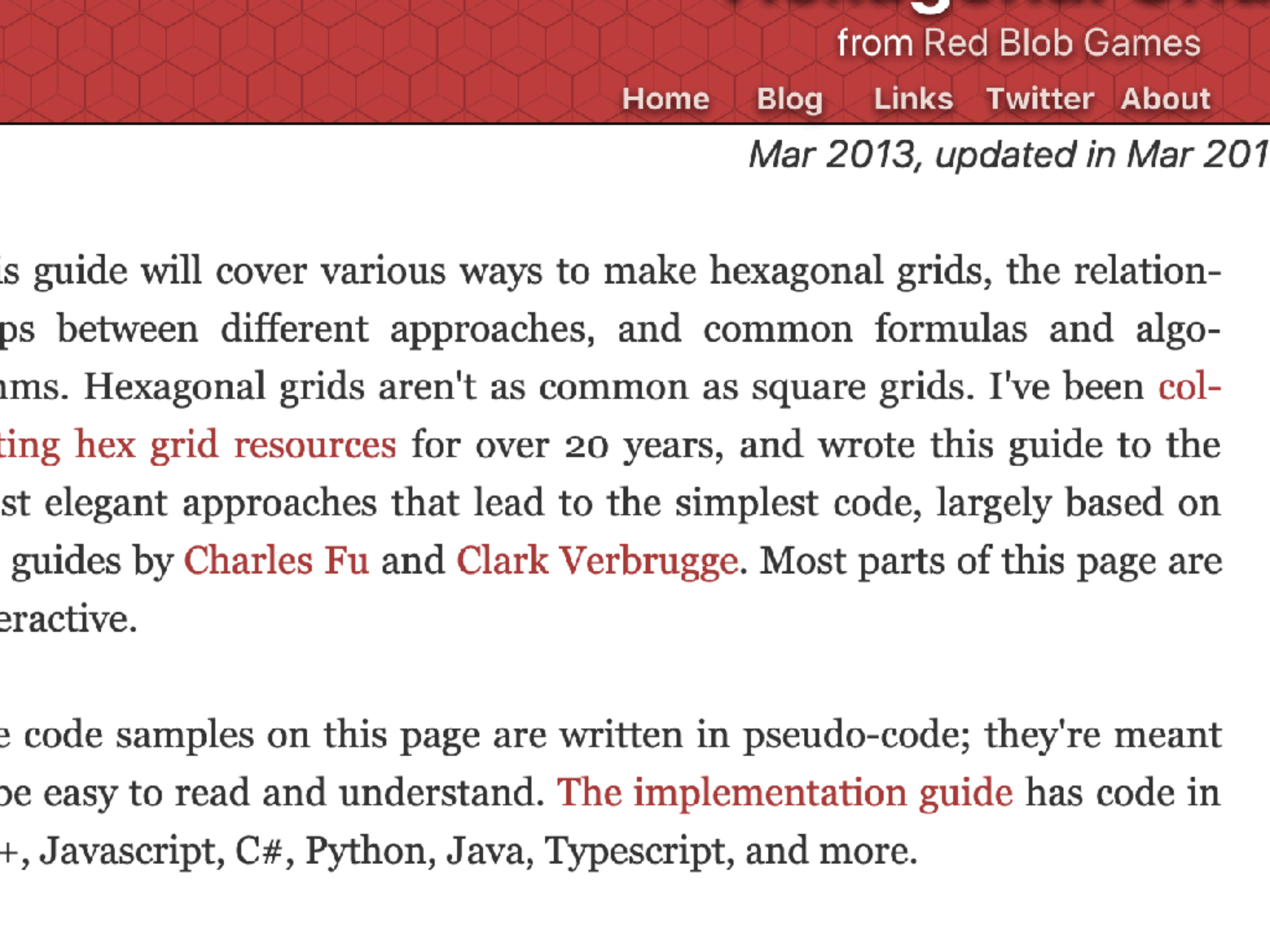
Hexagonal Grids

from Red Blob Games

Mar 2013, updated in Mar 2015, Apr 2018, Feb 2019

This guide will cover various ways to make hexagonal grids, the relationships between different approaches, and common formulas and algorithms. Hexagonal grids aren't as common as square grids. I've been collecting hex grid resources for over 20 years, and wrote this guide to the most elegant approaches that lead to the simplest code, largely based on the guides by Charles Fu and Clark Verbrugge. Most parts of this page are interactive.

The code samples on this page are written in pseudo-code; they're meant to be easy to read and understand. The implementation guide has code in C++, Javascript, C#, Python, Java, Typescript, and more.



from Red Blob Games

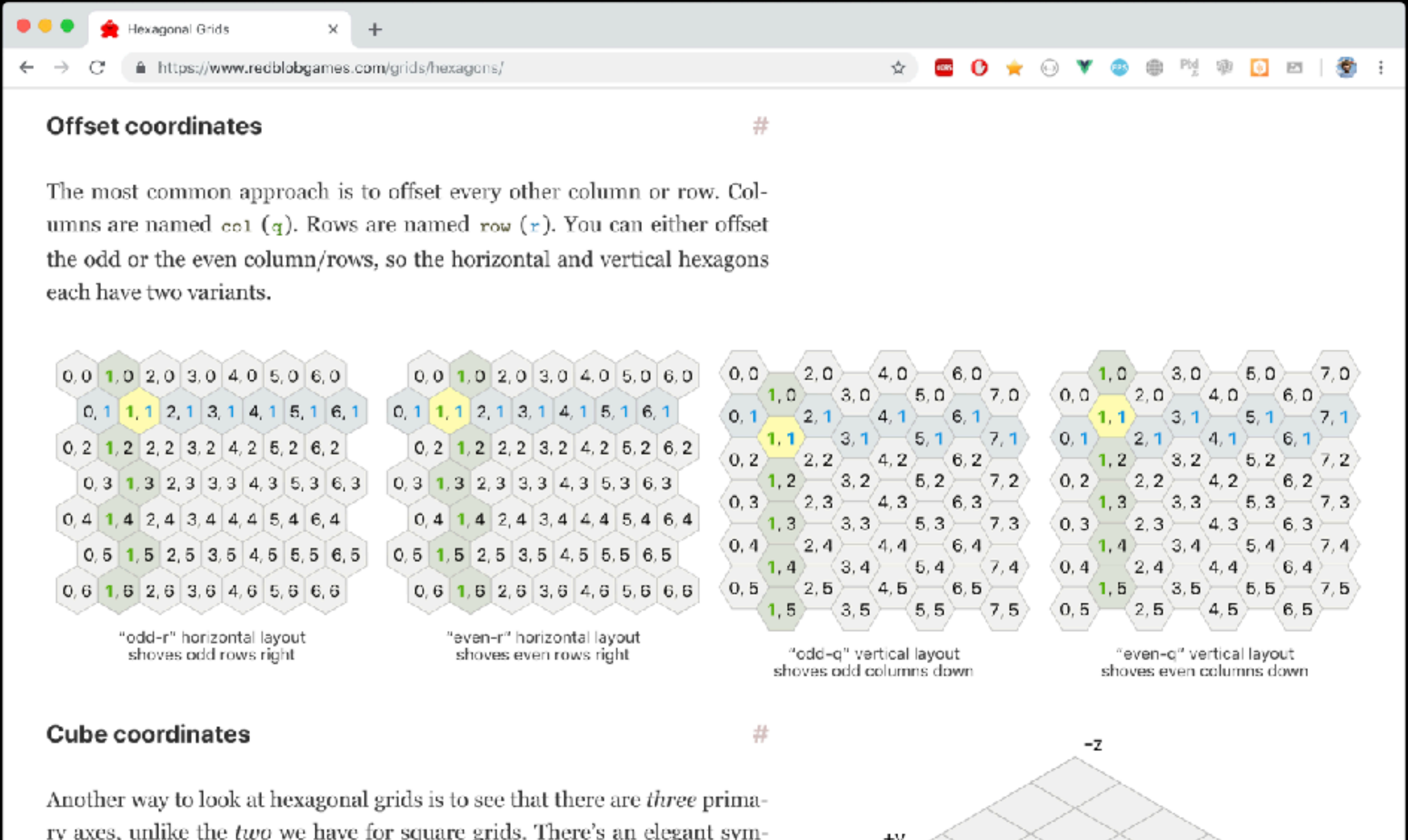
Home Blog Links Twitter About

Mar 2013, updated in Mar 201

is guide will cover various ways to make hexagonal grids, the relation-
ps between different approaches, and common formulas and algo-
nms. Hexagonal grids aren't as common as square grids. I've been col-
ting hex grid resources for over 20 years, and wrote this guide to the
st elegant approaches that lead to the simplest code, largely based on
guides by Charles Fu and Clark Verbrugge. Most parts of this page are
eractive.

e code samples on this page are written in pseudo-code; they're meant
be easy to read and understand. The implementation guide has code in
+, Javascript, C#, Python, Java, Typescript, and more.

over various ways to make
different approaches, and co
nal grids aren't as common as
resources for over 20 years,
proaches that lead to the sim
arles Fu and Clark Verbrugge



Offset coordinates

The most common approach is to offset every other column or row. Columns are named `col` (`q`). Rows are named `row` (`r`). You can either offset the odd or the even column/rows, so the horizontal and vertical hexagons each have two variants.

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5
0, 6	1, 6	2, 6	3, 6	4, 6	5, 6	6, 6

"odd-r" horizontal layout
shoves odd rows right

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5
0, 6	1, 6	2, 6	3, 6	4, 6	5, 6	6, 6

"even-r" horizontal layout
shoves even rows right

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0	7, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2	7, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3	7, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4	7, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5	7, 5

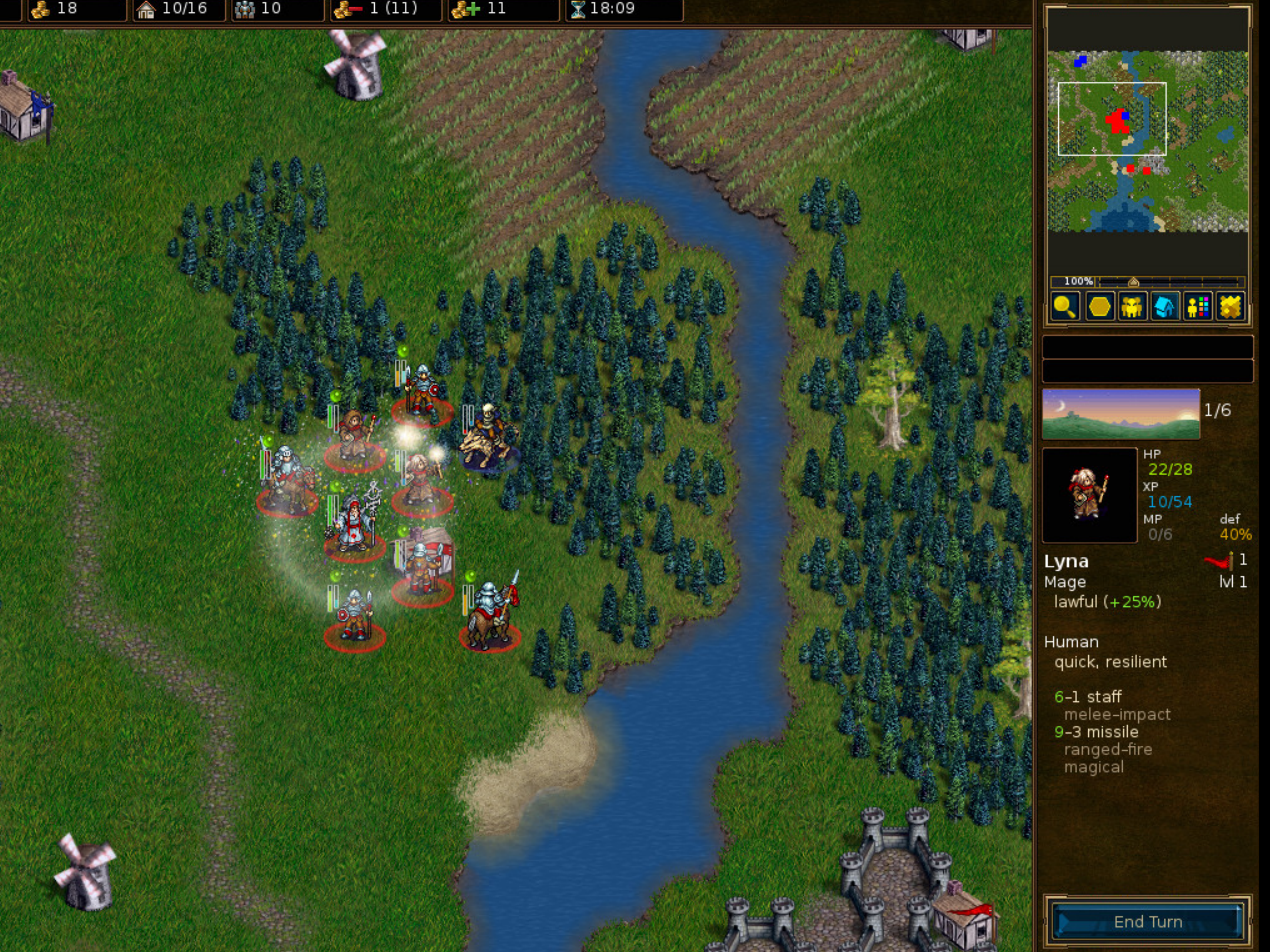
"odd-q" vertical layout
shoves odd columns down

0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0	7, 0
0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1
0, 2	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2	7, 2
0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3	7, 3
0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	6, 4	7, 4
0, 5	1, 5	2, 5	3, 5	4, 5	5, 5	6, 5	7, 5

"even-q" vertical layout
shoves even columns down

Cube coordinates

Another way to look at hexagonal grids is to see that there are *three* primary axes, unlike the *two* we have for square grids. There's an elegant sym-

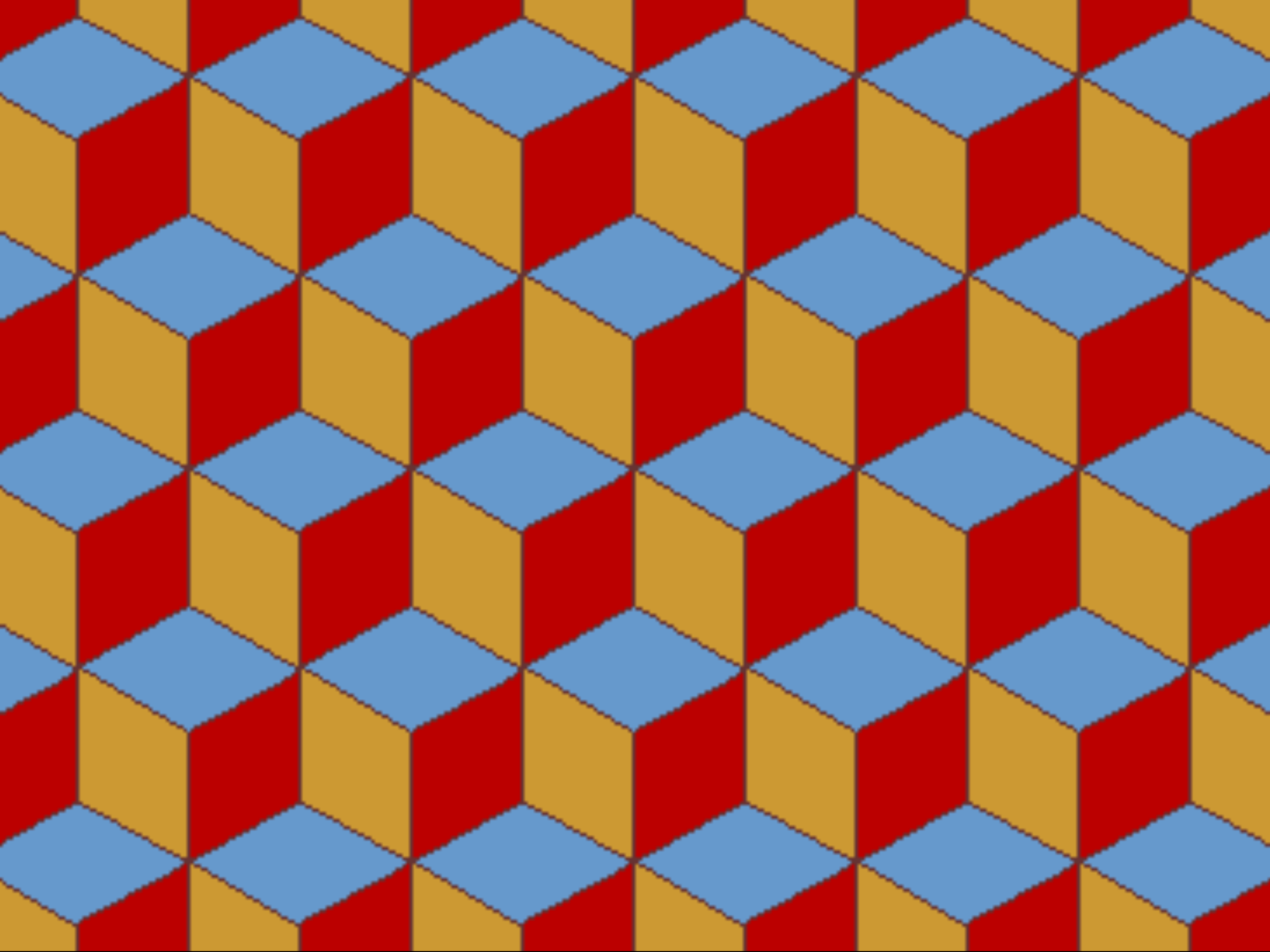


HP 22/28
XP 10/54
MP 0/6
def 40%

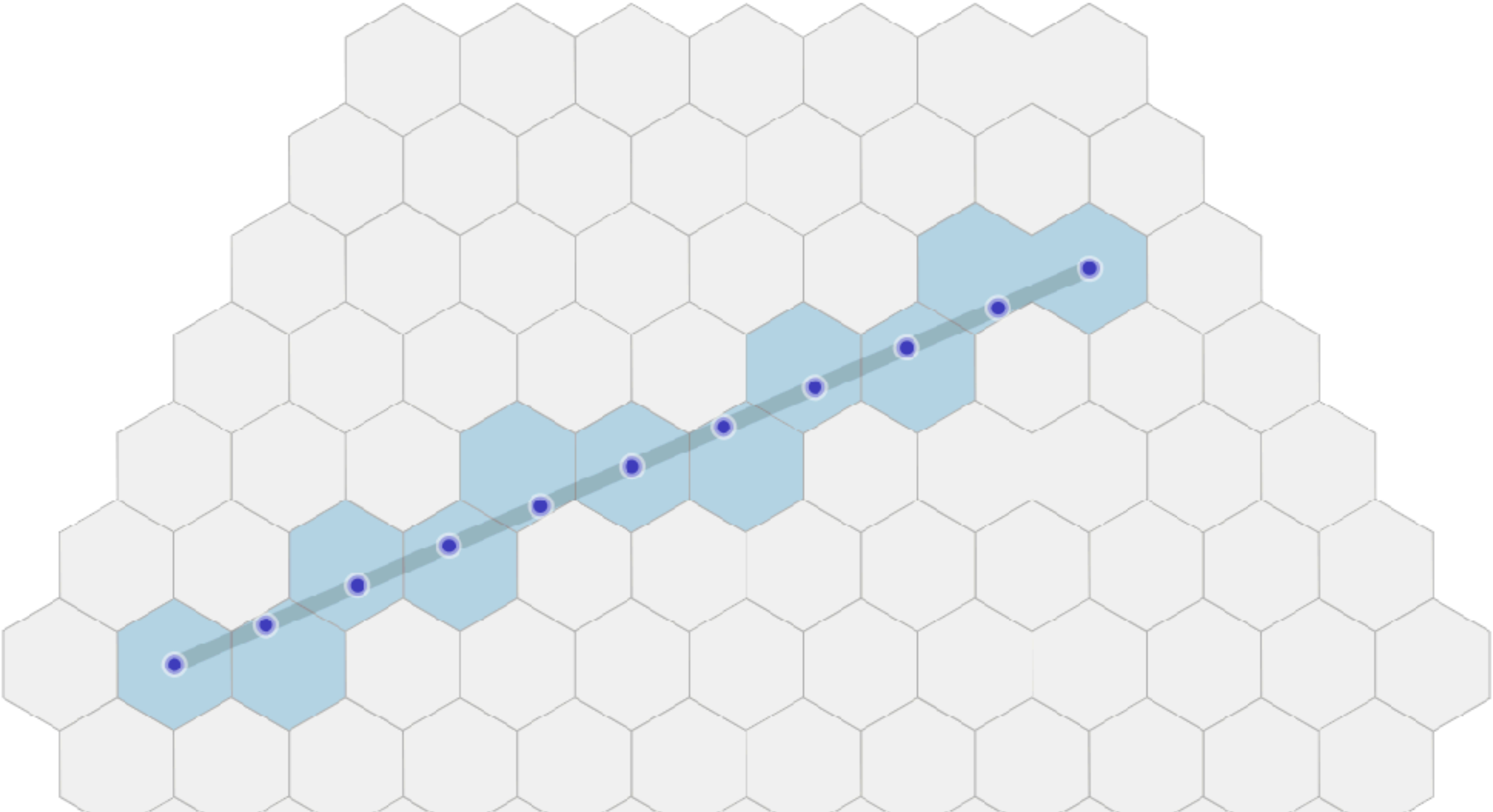
Lyna 1
Mage lvl 1
lawful (+25%)

Human
quick, resilient

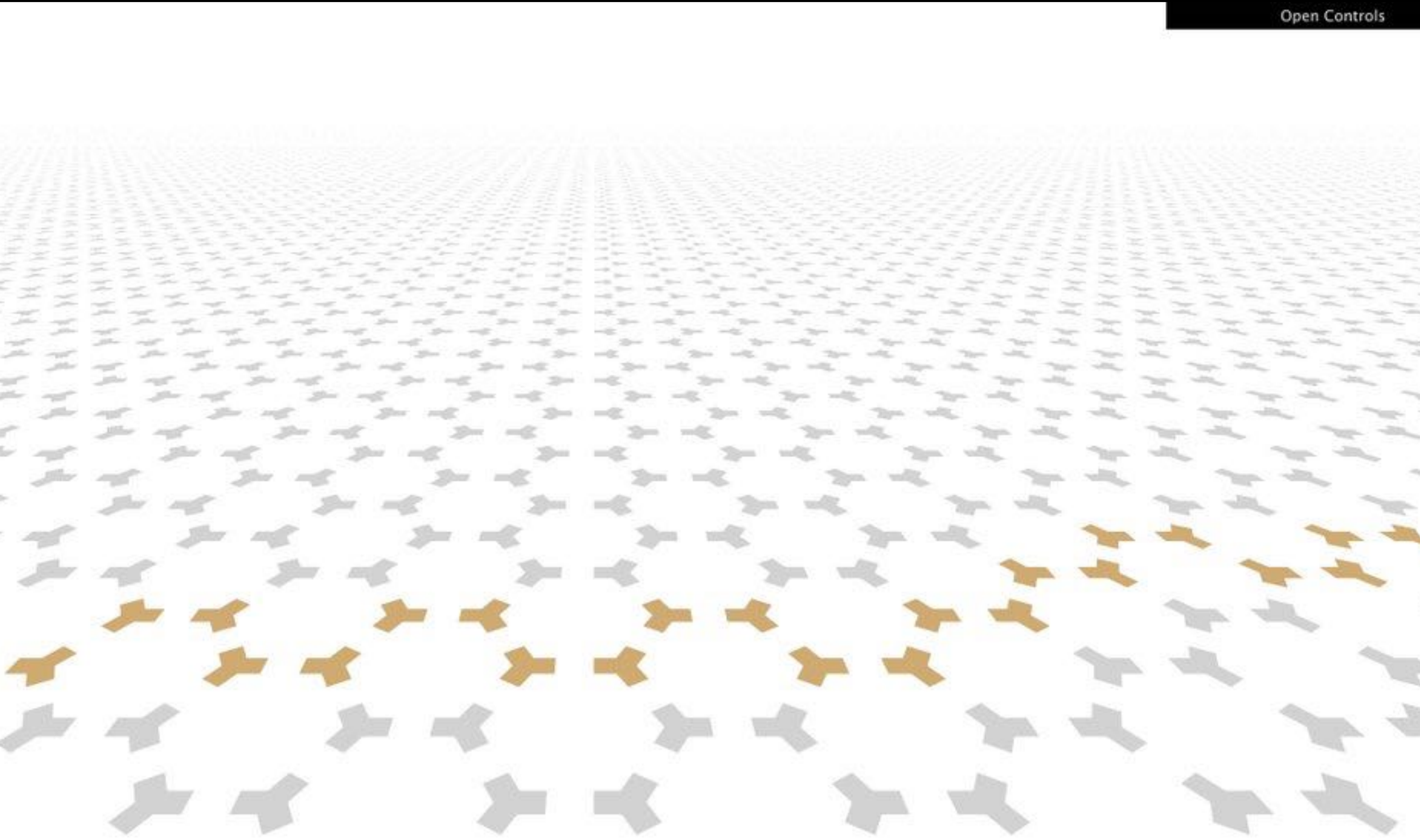
6–1 staff
melee–impact
9–3 missile
ranged–fire
magical



Как делают pathfinding?



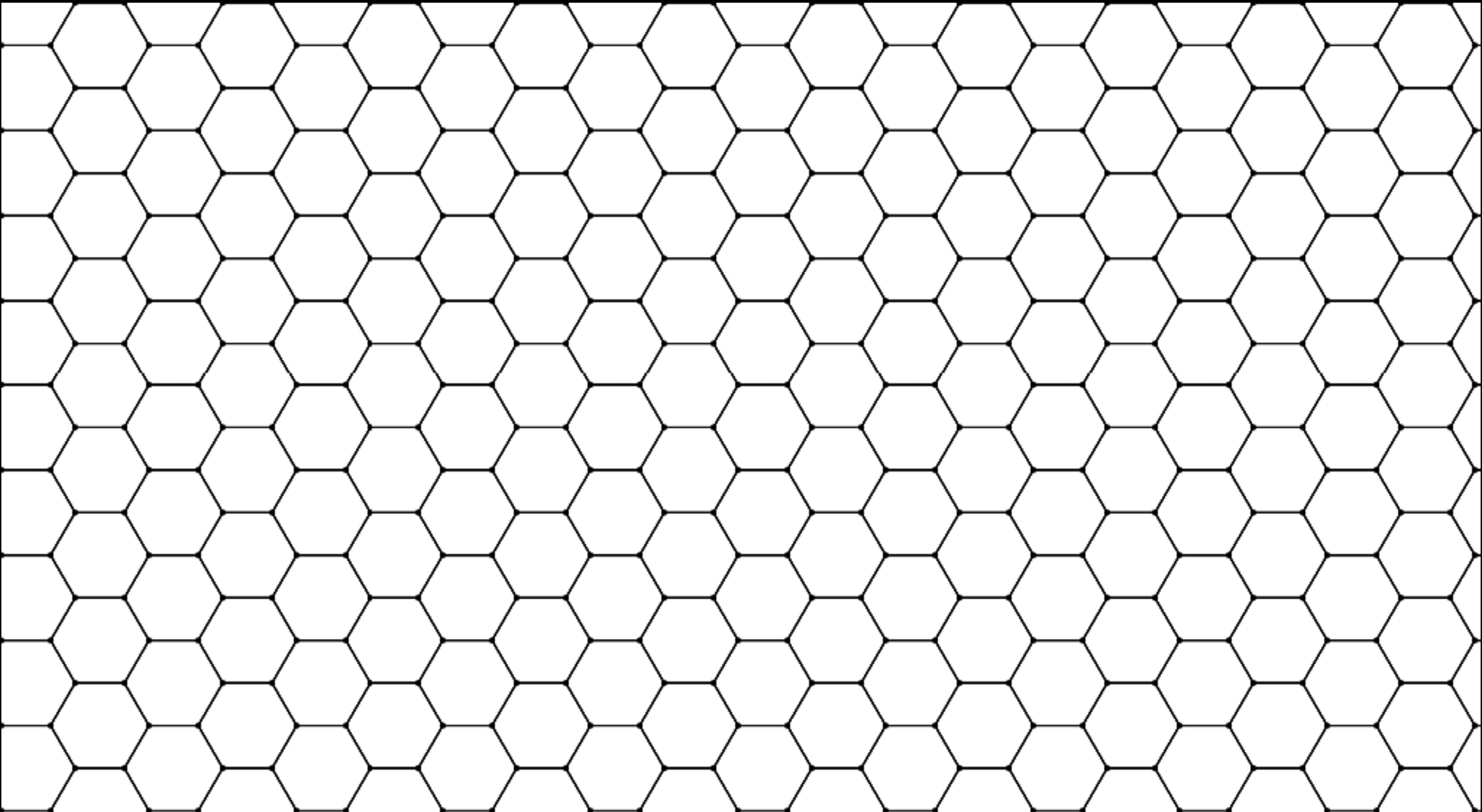
Не то.

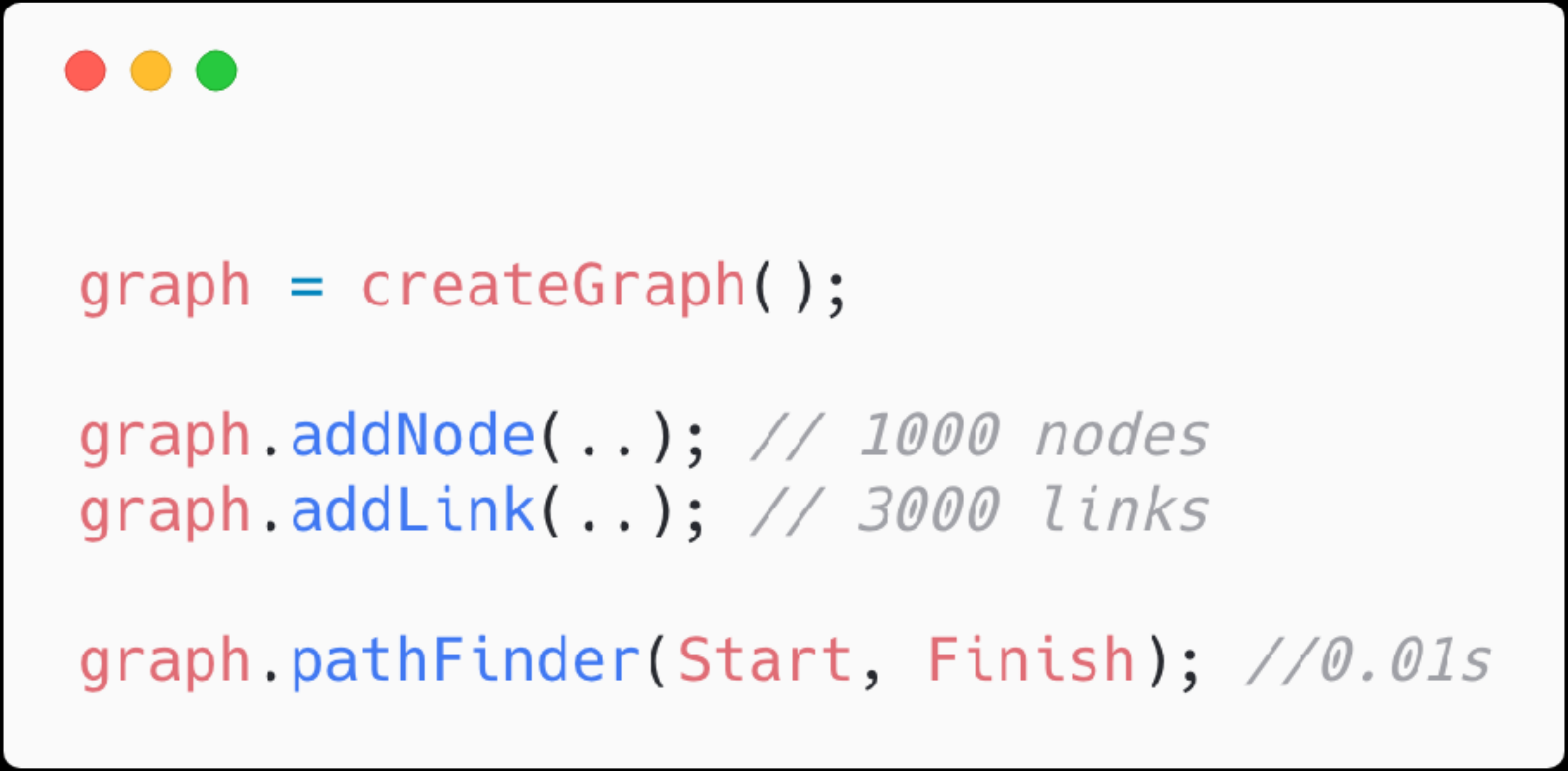


Canvas2D

Graph

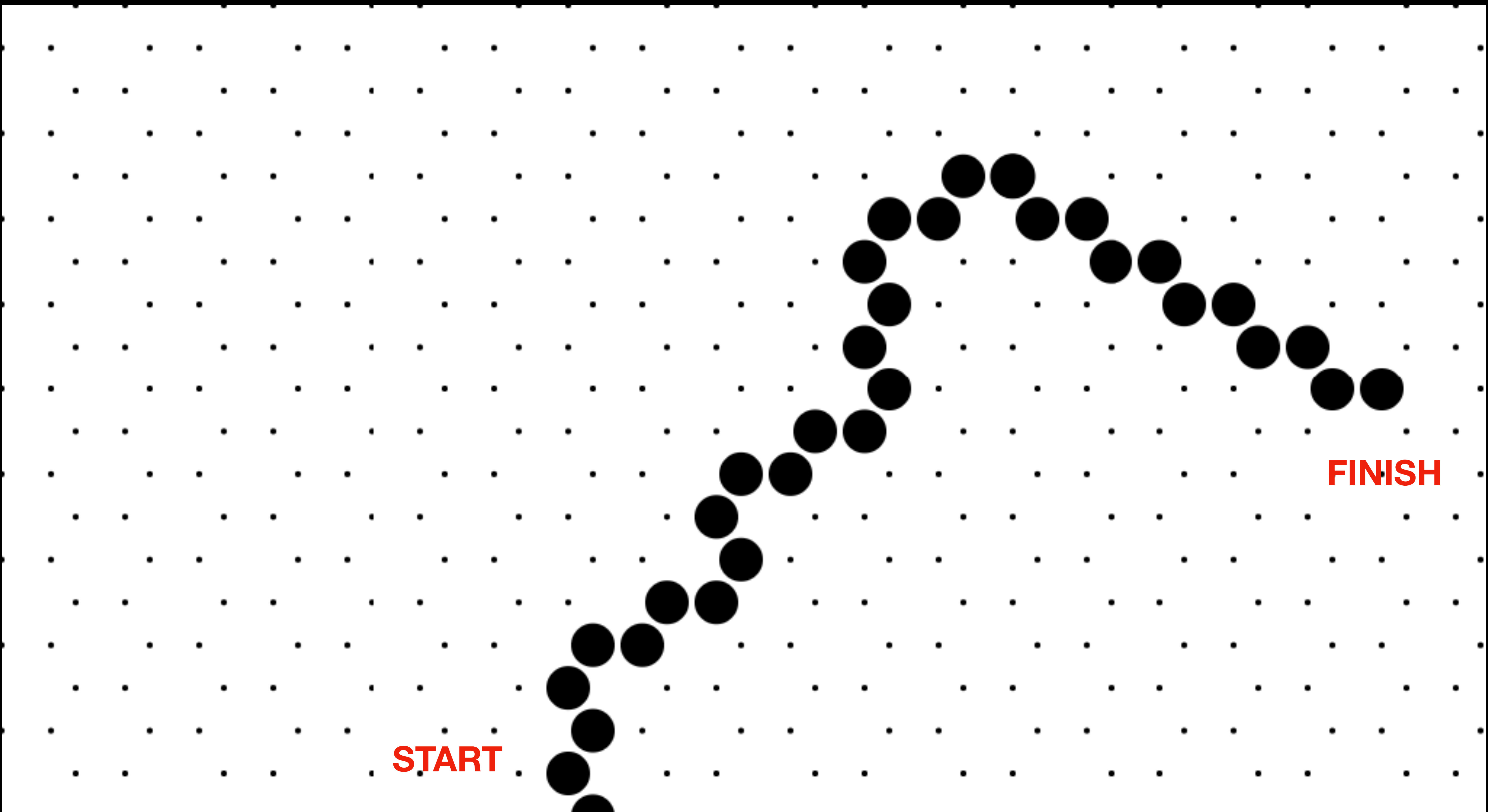
github.com/anvaka/ngraph

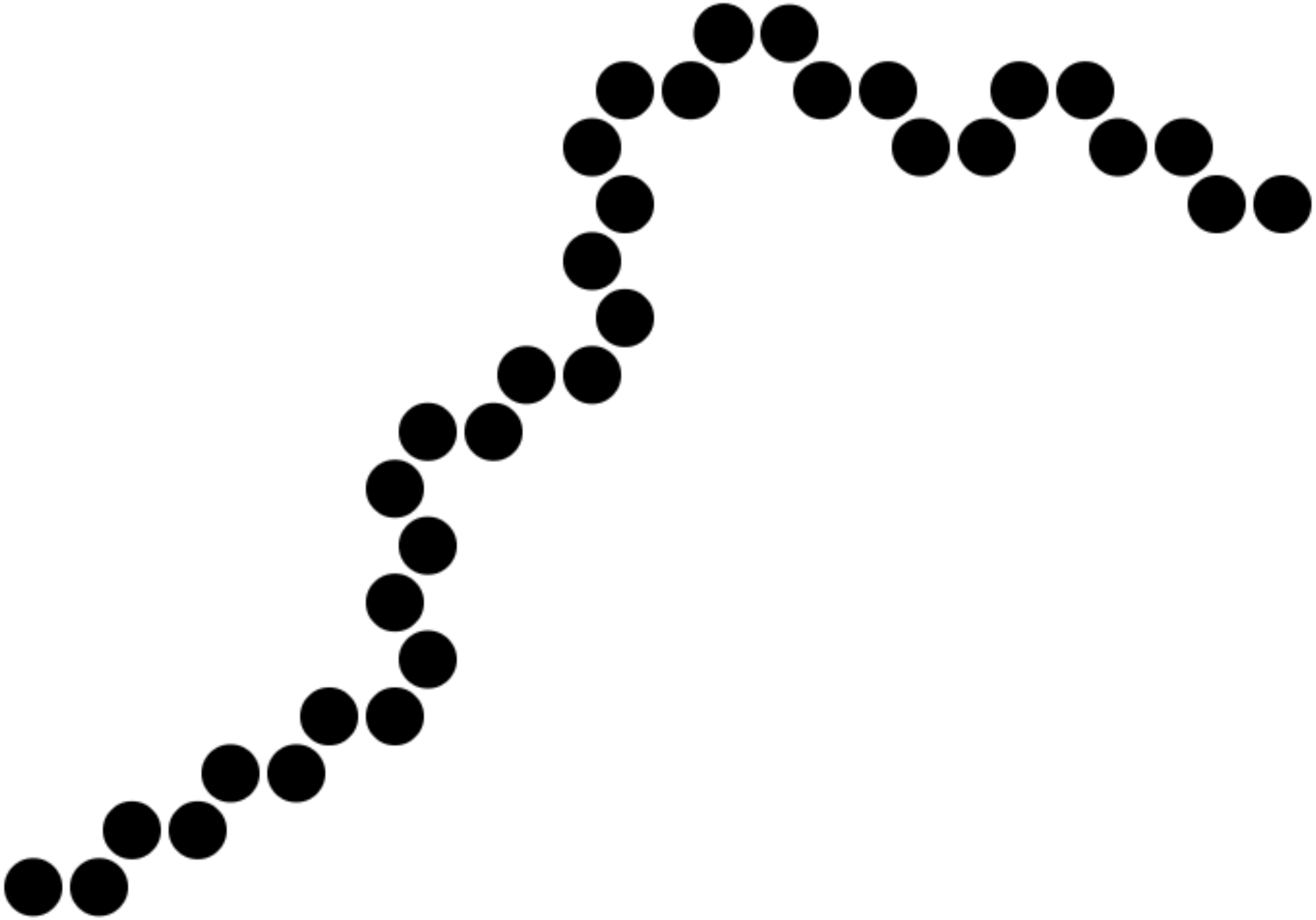


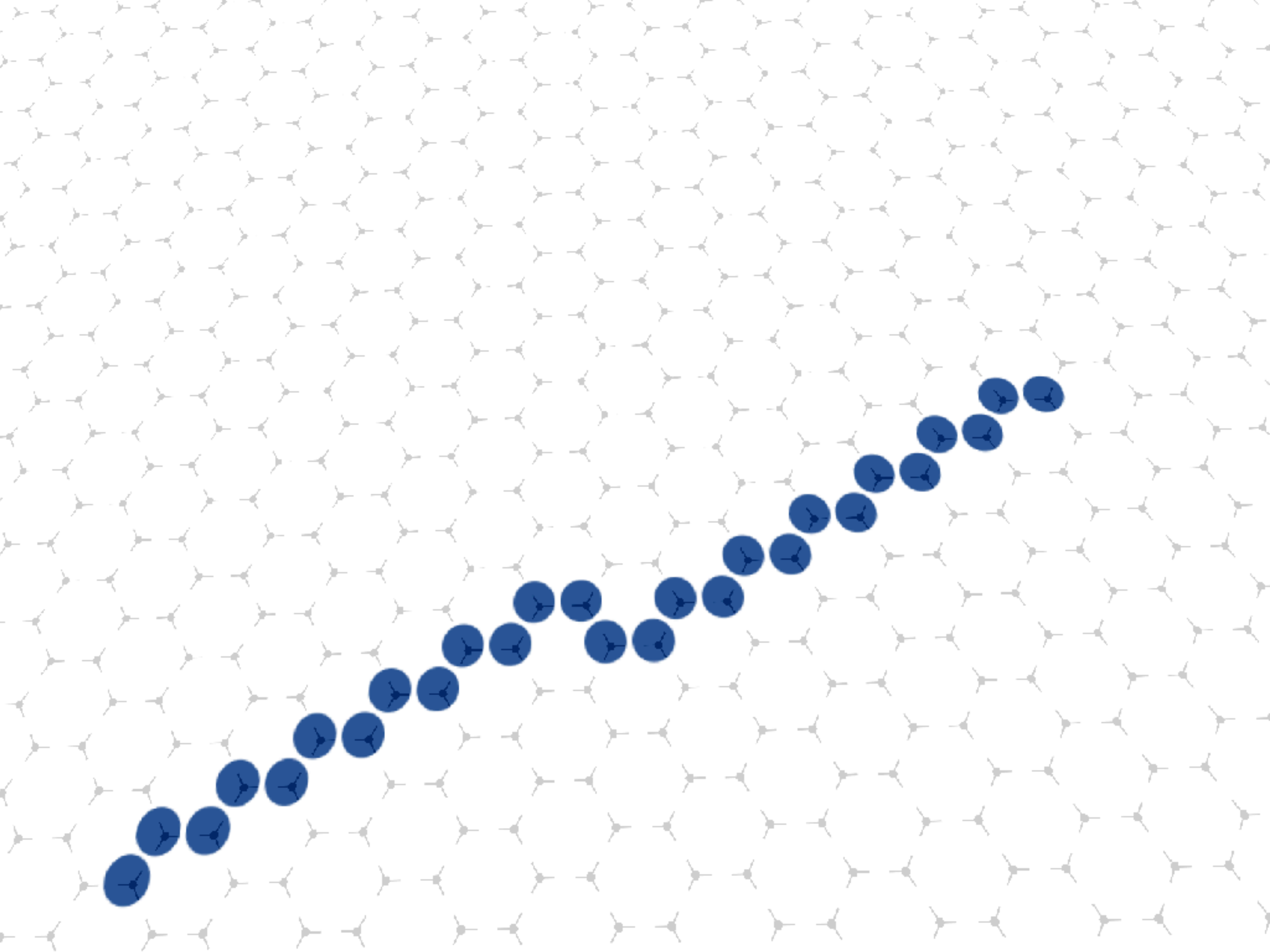


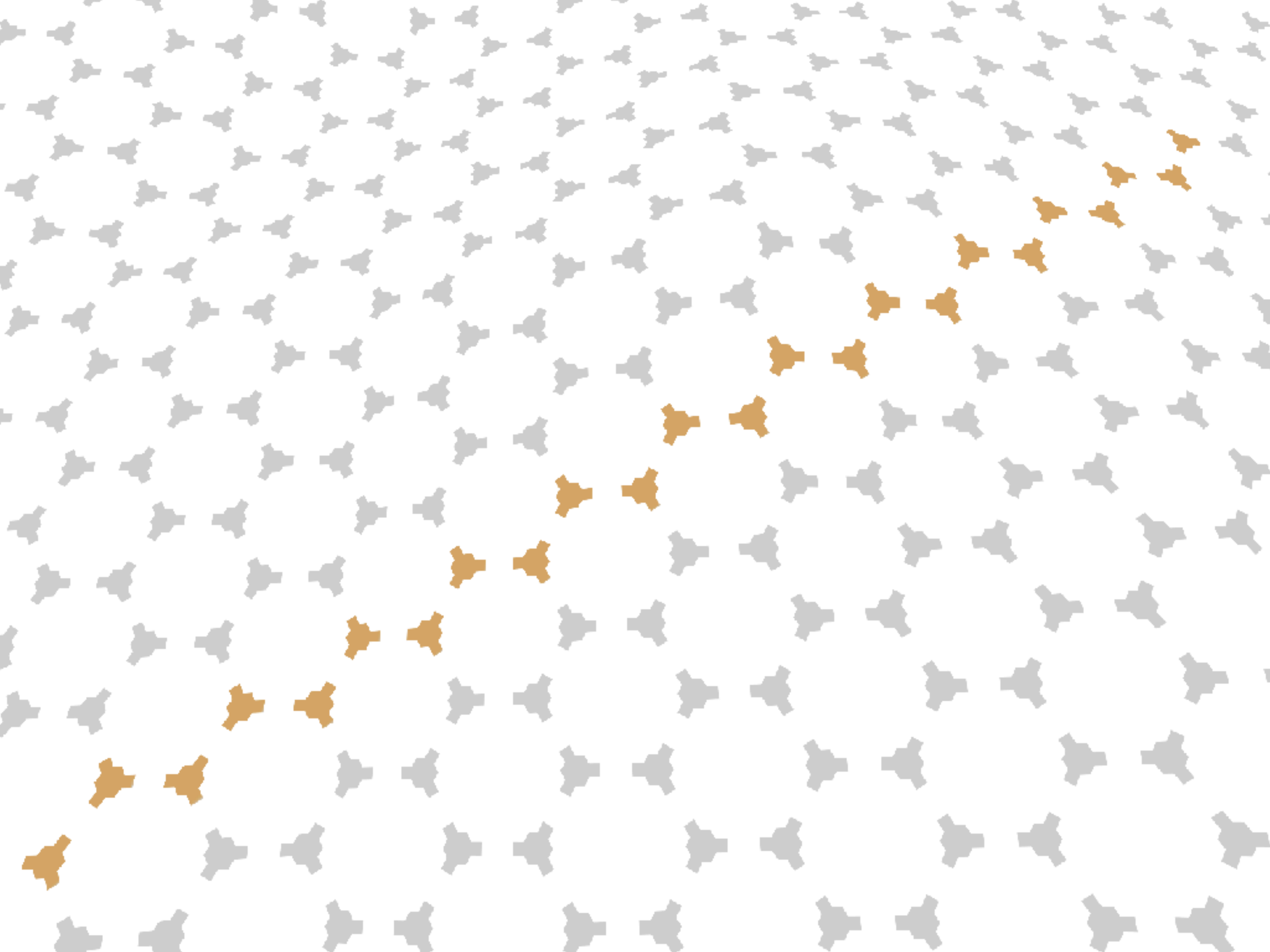
```
graph = createGraph();

graph.addNode(..); // 1000 nodes
graph.addLink(..); // 3000 links

graph.pathFinder(Start, Finish); //0.01s
```

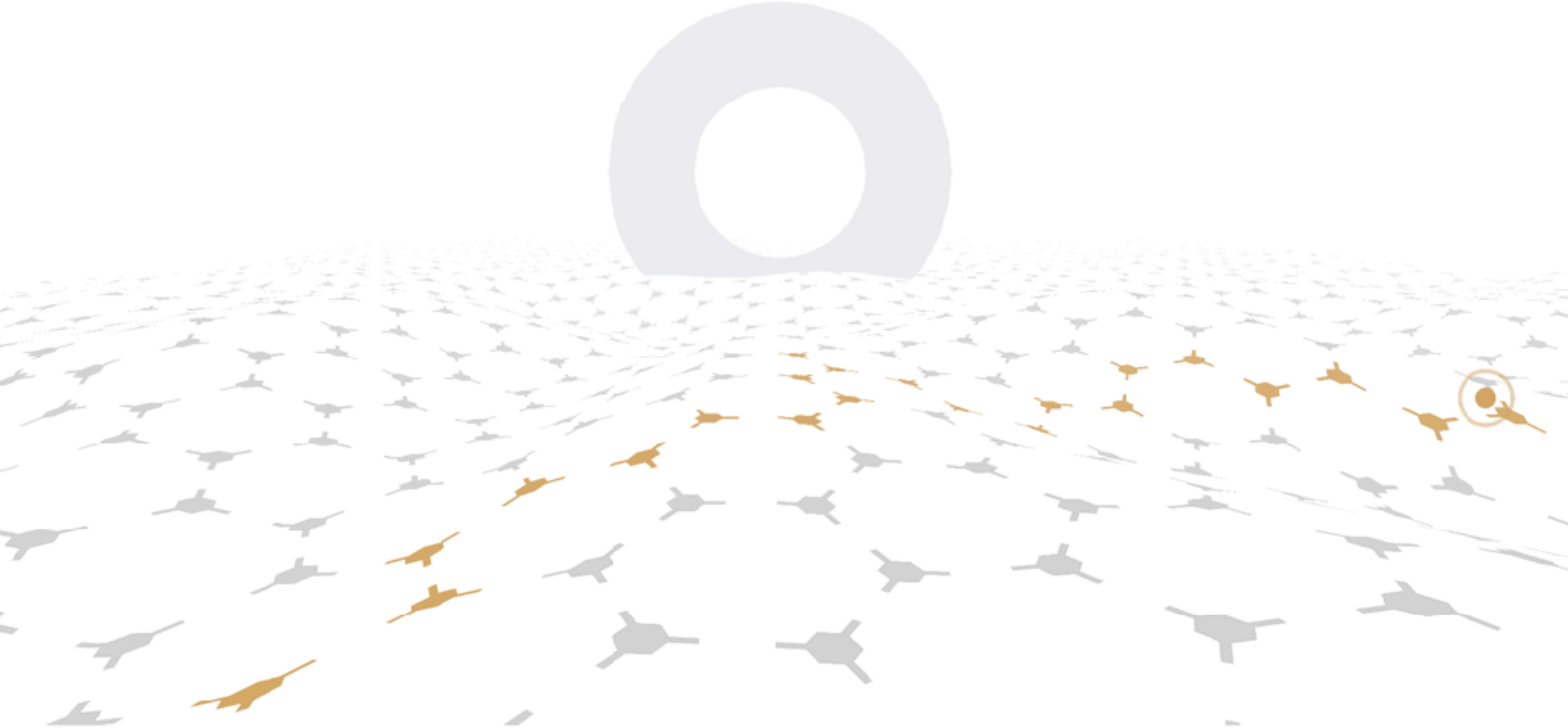
Yeah!











Ради чего?

Спасибо, клёво получилось

– Дизайнер

Овечка-Гальватрон!

Спасибо!

- http://riverco.de - работаю
- twitter.com/akella
- https://www.youtube.com/user/flintyara
- facebook.com/akella