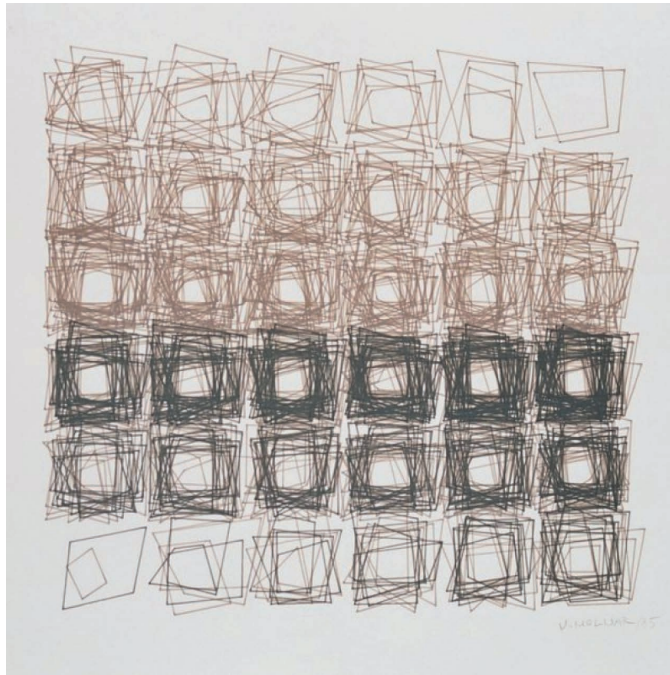
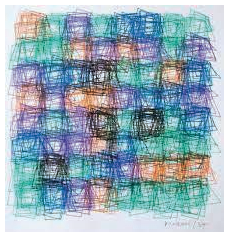
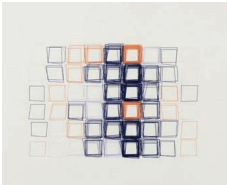
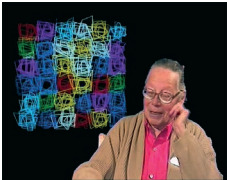


# Recoding project algorithmic aesthetics

Tristan Debeaune & Wendy Gervais

## Chosen artwork



Véra Molnar, *Structure de quadrilatère*, 1985

Ink on paper  
30 cm x 30 cm  
Algorithm on computer + plotter

Vera Molnar is a pioneer in computer art, her work is experimental, she tries and makes programs in order to experiment randomness with shapes - oftenly squares.

## Analysis

The version we chose :

- **squared light-grey canvas**
- **black and brown squared lined pattern**
- **6 per 6 grid**

Each cell :

- scribbled with **quadrangles**
- all cells are different, **randomly generated**
- can overlap its neighbours

Number and colours of quadrangles are **variable** :

- **3 first rows** : brown, the density of lines grows row by row, right to left, from top right (2 quadrangles) to bottom left (about thirty).

- **3 last rows** : superposition of half brown and half black lines. The repartition of density is mirrored (the density increases row by row from bottom left to top right).

**We have been very interested in the «ordered» chaos created by this grid, and its imbrication of alternative squares ; we are now trying to recreate it with an algorithm.**

# Pseudo-code

1. Take a brown pen.

2. Start on top-right corner. Draw two deformed superposed squares, their shapes are random.

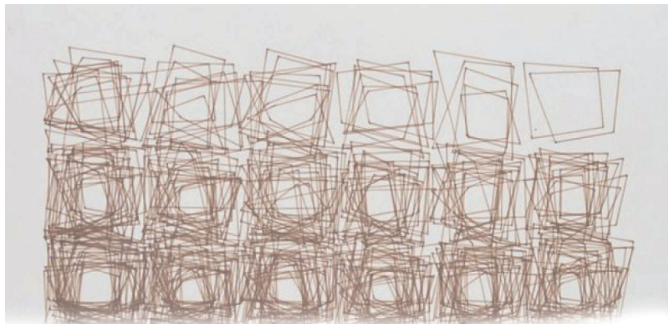


3. On the left of this first shapes, start again drawing 3 more random quadrangles over the same position.

4. Repeat on the row until having 6 squares increasing the amount of lines each time.



5. Repeat this 2 more times, starting from the right each row.



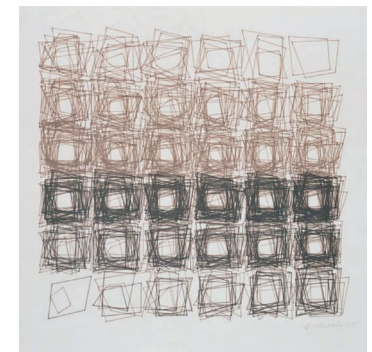
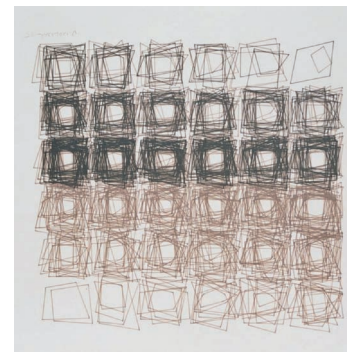
6. Rotate the canvas upside down.



7. Take also a black pen.

8. Repeat the 5 first steps again. But, for each shape, begin by drawing the first half of the lines brown, then the second half black.

9. Rotate the canvas upside down again.



# Pseudo-algorithm

	<pre><b>function</b> setup()   create a 700*700 canvas   DrawMesh(6, 6)</pre>
<b>step 2</b>	<pre><b>function</b> DrawSquare(quad_amount, column, line)   push()     translate by 60*column on x axis, 60*line on y axis, 0 on z axis     for k from 0 to quad_amount, with a step of 1       dispersion &lt;- []       add 8 random relative integers between -10 and 10 to the dispersion array       draw a quadrangle centered on the square generated by the 4 vertices (40,40),       (80,40), (80,80), (40,80) using the 8 integers of dispersion added to the 8 coordinates     end for    pop()</pre>
<b>steps 1 to 5</b>	<pre><b>function</b> DrawMesh(width, height)   for j from 0 to height/2, with a step of 1     for i from width to 0, with a step of 1       ijquad_amount &lt;- 2*(width-i-1 + width*j + 1)       draw in brown       DrawSquare(ijquad_amount, i, j)     end for   end for</pre>
<b>steps 6 to 8</b>	<pre>  for j from 0 to height/2, with a step of 1     for i from width to 0, with a step of 1       ijquad_amount &lt;- width-i-1 + width*j + 1       drawn in brown       DrawSquare(ijquad_amount, width-1-i, height-1-j)       drawn in black       DrawSquare(ijquad_amount, width-1-i, height-1-j)     end for   end for</pre>

# p5 version

## Code

```
1 // Recoding project 2022
2 // T. DEBEAUNE et W. GERVAIS
3 // Vera Molnar - Square Structures
4
5 // Static Version (V2)
6
7 function setup() {
8   let img = createCanvas(700, 700);
9   background(220);
10  scale(1.5,1.5,0);
11  DrawMesh (6,6);
12
13  //saveCanvas('img', 'png');
14 }
15
16
17 // Draw n superposed quadrangles (based on 40*40 squares)
18 // Position i-column, j-row in the Mesh
19
20 function DrawSquare(n, i, j) {
21   push();
22   noFill();
23   translate(60*i, 60*j, 0); // put it on i-column, j-row, with a 20px "distance" between 2 quads
24
25   for (let k=0; k<n; k++) { // n times
26     let dispersion = [];
27
28     for (let d=0; d<8; d++) { // generate the dispersion array, filled with 8 random relative integers
29       which will be the dispersions around the 8 coordinates
30       let disp = int (random(-12, 15));
31       dispersion.push(disp);
32     }
33
34     // draw the quadrangle :
35     quad(40+dispersion[0],40+dispersion[1],80+dispersion[2],40+dispersion[3],80+dispersion[4],80+dispersion[5],
36     40+dispersion[6],80+dispersion[7]);
37
38   }
39   pop();
40 }
41
```

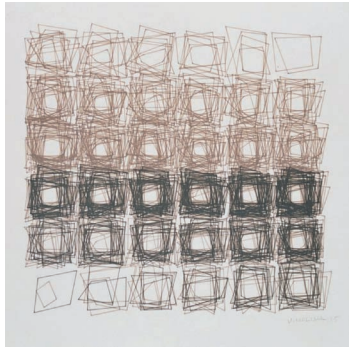
```
42 // Draw the i*j mesh of n superposed quadrangles
43 function DrawMesh(w,h) {
44   let quad_amount;
45   strokeWeight(0.7);
46
47   for (let j=0; j<h/2; j++) {
48     for (let i=w-1; i>=0; i--) {
49       ijquad_amount=2*((w-i-1+w*j)+1); // increasing from right to left (-i), keep the number increasing
50       from row to row (w*j), as in the original artwork (*2)
51
52       // 3 first lines (brown)
53       stroke("#8f746d");
54       DrawSquare(ijquad_amount,i,j);
55     }
56
57     for (let j=0; j<h/2; j++) {
58       for (let i=w-1; i>=0; i--) {
59         ijquad_amount=(w-i-1+w*j)+1; // split in 2 the previous amount (/2) to draw half brown and half
60         black
61
62         // 3 last lines (brown)
63         stroke("#8f746d");
64         DrawSquare(ijquad_amount, w-1-i, h-1-j); // positions = start bottom right
65
66         // 3 last lines (black)
67         stroke("#515757");
68         DrawSquare(ijquad_amount, w-1-i, h-1-j);
69       }
70     }
71   }
72 }
```

p5 - <https://editor.p5js.org/kau.grv/sketches/OC2bP--qq>

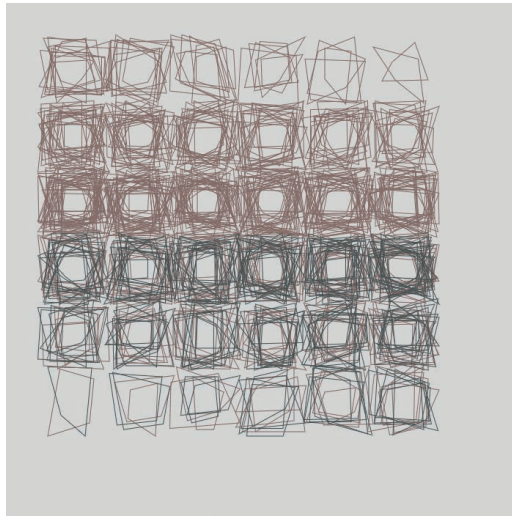
gitlab (STATIC) - <https://gitlab.com/debeaunetristan/projet-recoding>



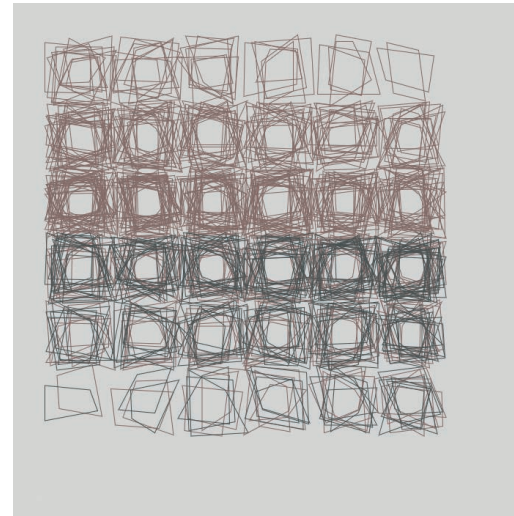
# Outputs



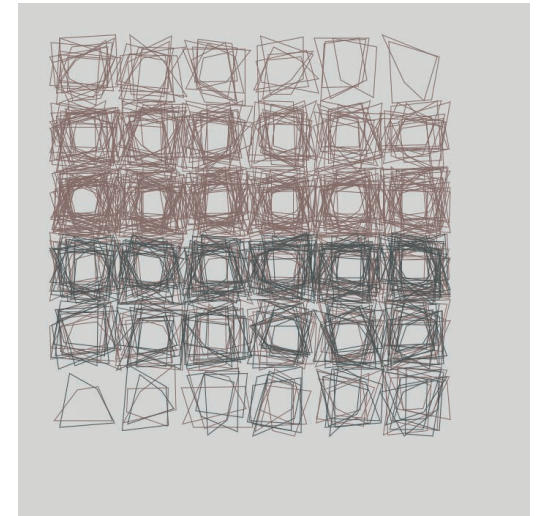
Original artwork



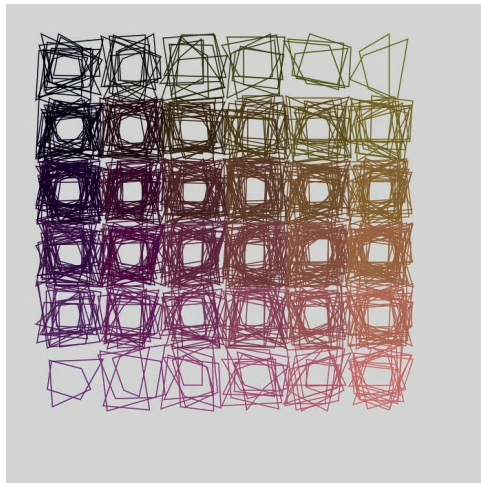
#1



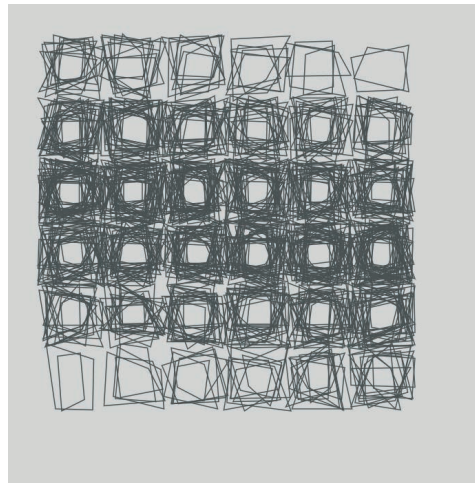
#2



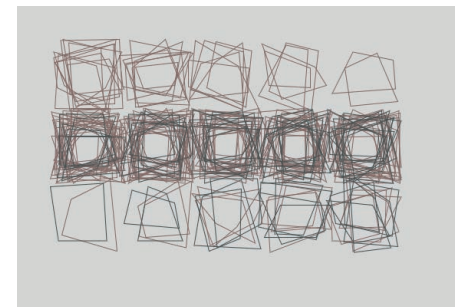
#3



with a color gradient



all black + 1px stroke weight

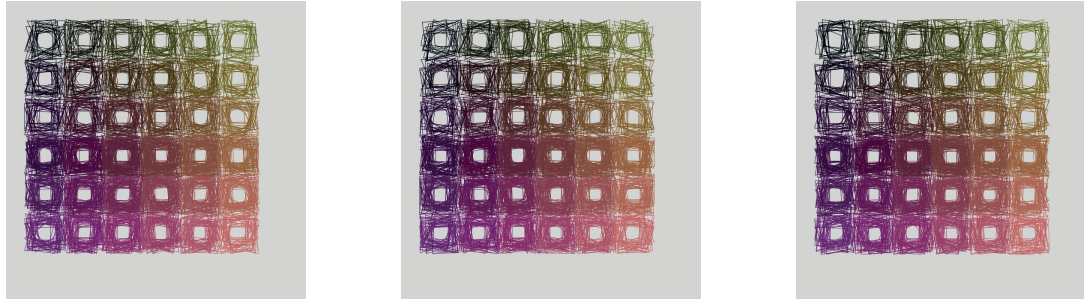


5\*3

# Variations (v5)

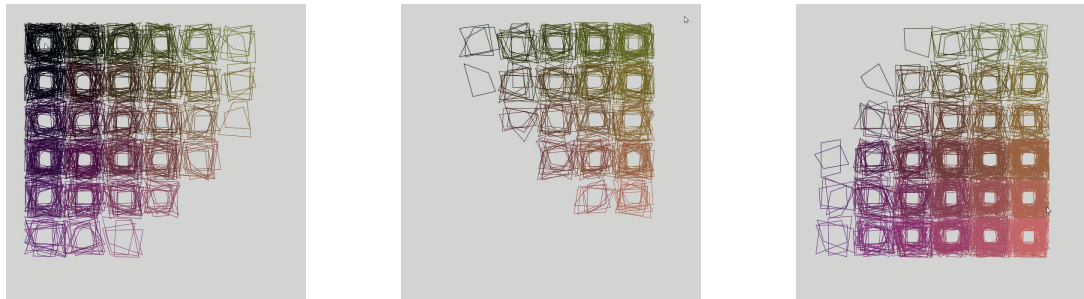
We implemented 3 major interactive extensions to our algorithm (**please watch attached video!**)

## 1. Animation (shapes are shaking with frame rate of 10)



```
function draw() {  
  frameRate(10);  
  background(220);  
  scale(1.5,1.5,0);  
  DrawMesh (6,6);  
}
```

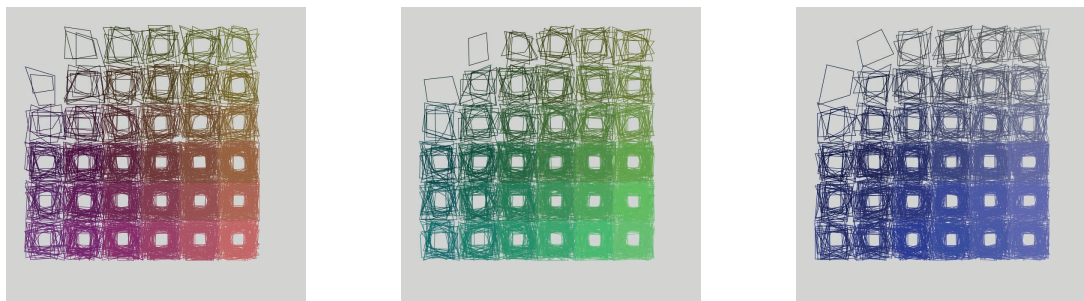
## 2. Mouse tracking (the density of line follows the cursor)



```
function DistIJ(i, j) {  
  let X = i*90 + 90;  
  let Y = j*90 + 90;  
  return sqrt((X-mouseX)*(X-mouseX) + (Y-mouseY)*(Y-mouseY));  
}
```

```
ijquad_amount=50-DistIJ(i,j)/10;
```

## 3. Mouse click interaction (the color changes when the mouse is clicked)



```
function mouseClicked() {  
  s++;  
  if (s === 4) {  
    s = 1;  
  }  
}
```

```
if (s==1) {  
  stroke(20*(i+j),20*i,20*j);  
}  
if (s==2) {  
  stroke(20*i,20*(i+j),20*j);  
}  
if (s==3) {  
  stroke(20*i,20*i,20*(i+j));  
}
```

# Variations (v5)

p5 - <https://editor.p5js.org/kau.grv/sketches/cGmNMxEaF>

gitlab (DYNAMIC/v5) - <https://gitlab.com/debeaunetristan/projet-recoding>

# Annex

To look like the original, we used a 0.7 stroke weight, #8f746d and #515757 as brown and black and made a 6\*6 grid, but we could change all those parameters in our fonctions to have the liberty of create other grids!

In the first version we used rotate functions to do the two parts, but we eventually managed to respect the pseudo-code better by really drawing each shape in the good order. We also increased the range of the random numbers so as the shapes can overlap.

All-in-all, we discovered the quad() function, we learnt how to implement a grid with «for» loops using their positions in an «aesthecial» way.