Jules Fouchy

Recoding Frieder Nake

Matrizenmultiplikation, by Frieder Nake, 1967
Nr.1, Series 2.5-1 and Nr.2, Series 2.5-5

http://spalterdigital.com/artworks/nr-1-series-2-5-1/
http://spalterdigital.com/artworks/nr-2-series-2-5-5/

In this series, Frieder Nake is tiling his squared canvas with small squares who have a randomly chosen colour.

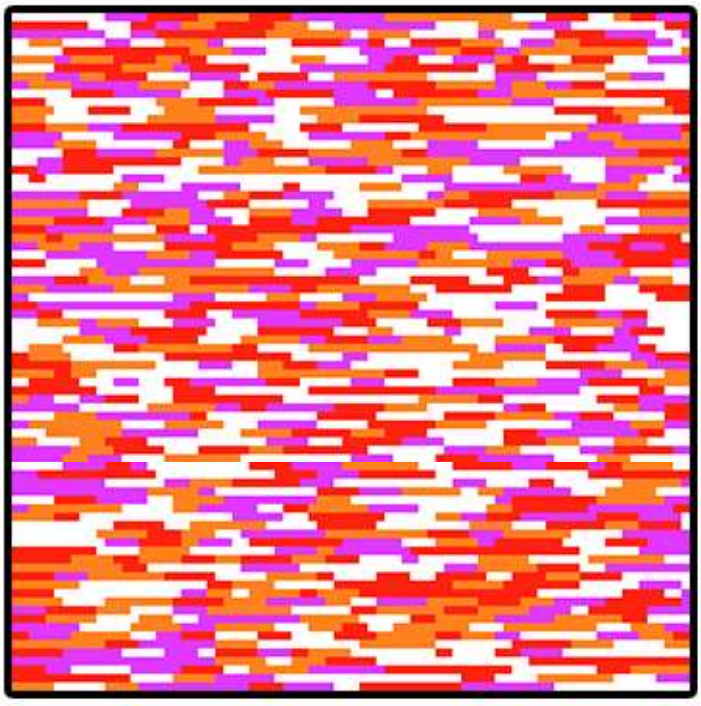
The randomness is not uniform though, and makes so that we see shapes appear : lines and “corners” in the two examples I have picked.

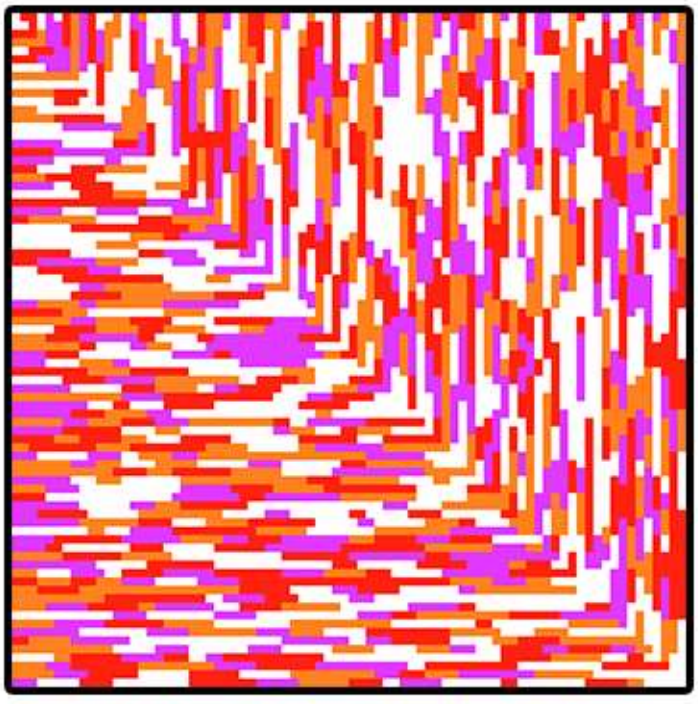
To reproduce this effect I have decided to have a marker (let’s call it an Ant) walking through the canvas and laying squares as it goes.

My first way of picking the colour at random has been to give the ant an initial colour, as well as a small probability of changing colour each time it draws a square. This way the ant draws a series of squares with the same colour before switching to another colour.

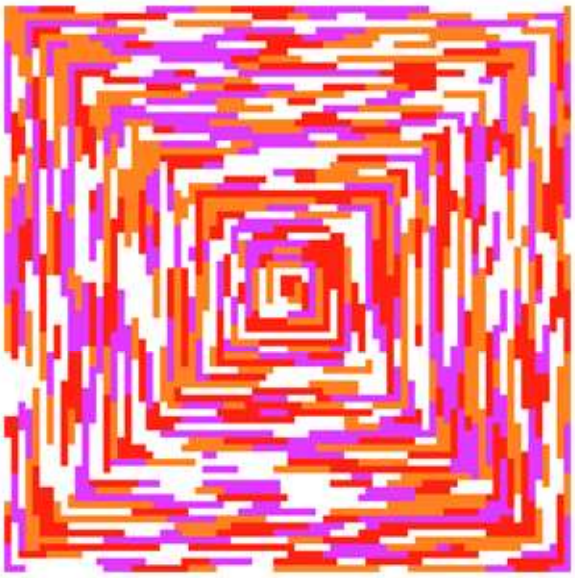
Now to have any shape I want appear I only have to tell the ant in which order it should go through the whole canvas.

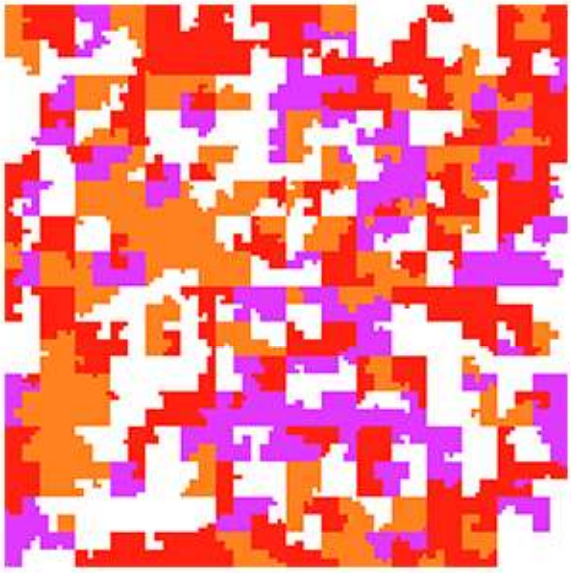
So I started with “line after line” and “corner after corner” and the results follow :



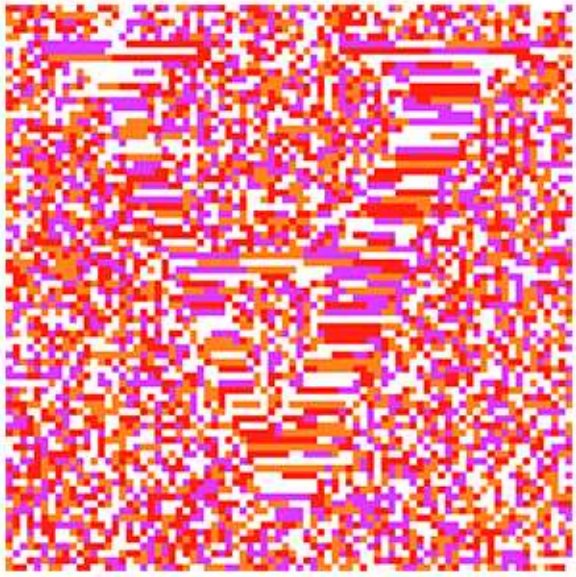


The first other trajectory I have tried was the spiraling one.
And then I played with the Hilbert Curve. To put it simple, it is a very windy trajectory that remains in a given squared region as long as possible. A very good explanation of the Hilbert Curve can be found here : https://youtu.be/3s7h2MHQtxc
I also made a real-time-drawing version of the Hilbert Curve, which helps a lot in understanding how it is drawn (check out the code and play the animation !)

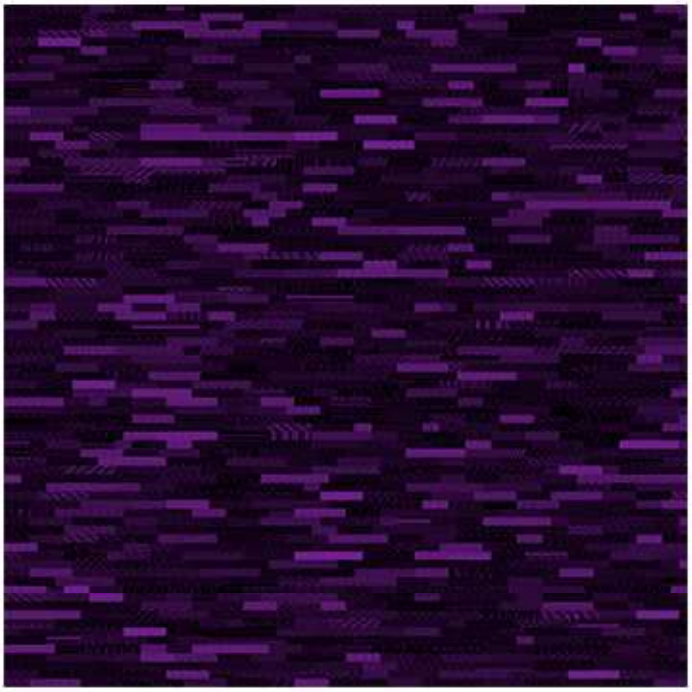




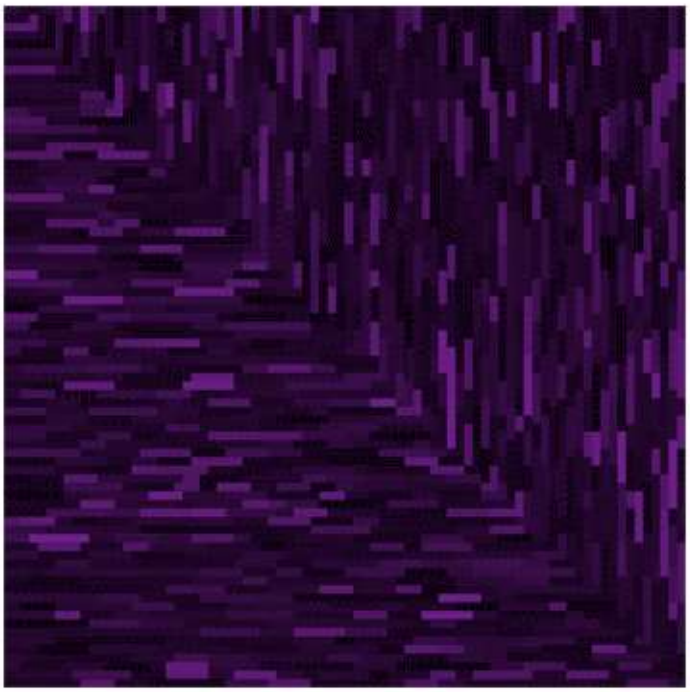
Finally I added the possibility to give a model image, and the ant would draw completely random squares outside the model. This way you can (can you?) see a shape appear on top of all the random noise.

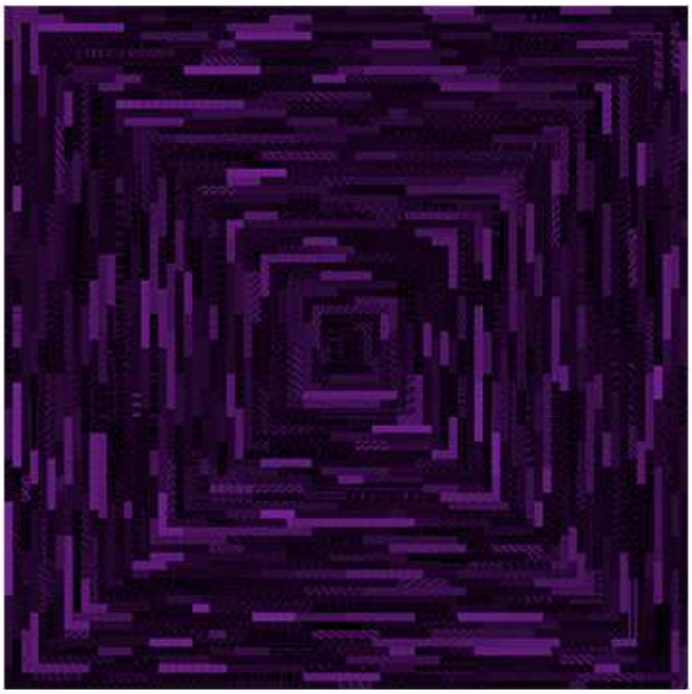


After that, I had to prepare the pen plotter session. Because the plotter cannot draw filled squares, I replaced them with hatched squares. Also, since it limits the number of colours, I moved on to one background colour (the paper colour) et one drawing colour, and went for a combination that felt good to me :

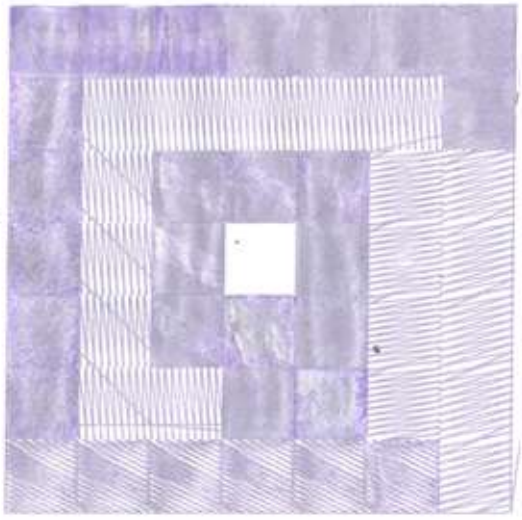


The change in colour has been replaced by a change in orientation and space between the hatches, making the squares more or less black / violet.



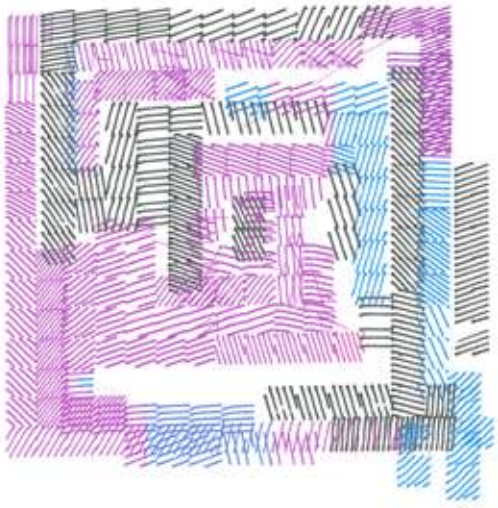


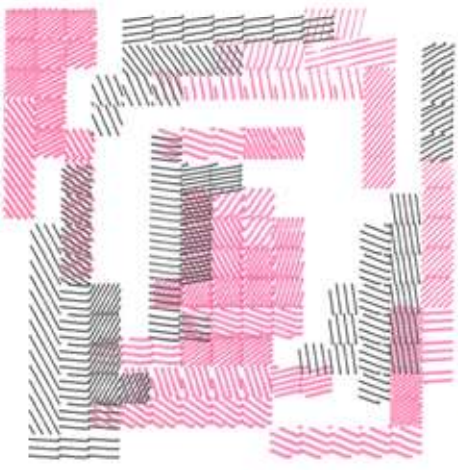
And here is what I came back with from my first pen plotter session. It worked OK except that my pen was too heavy and the machine couldn't pull it up, which caused unintended lines between the squares.

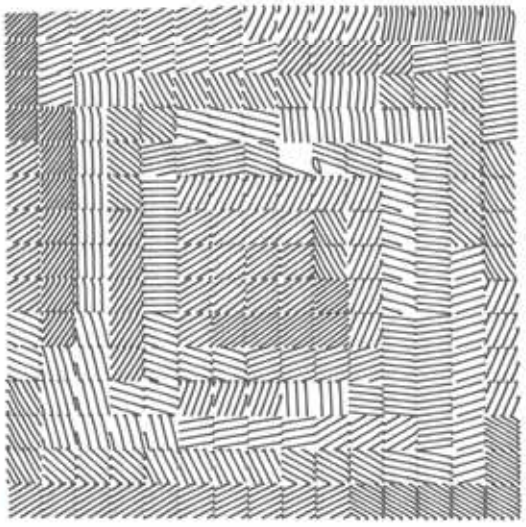


After that though, I still wanted to do polychromatic drawings. The problem was that changing the pen each time the ant actually changes colour would be way too long (and source of many inaccuracies in the drawing). So what I first tried was to create multiple SVGs, one for each colour : therefore I would only need to change the pen between two SVGs. But it turned out that placing the different SVGs at the exact same location on the software window each time was difficult, and therefore the colours did overlap :

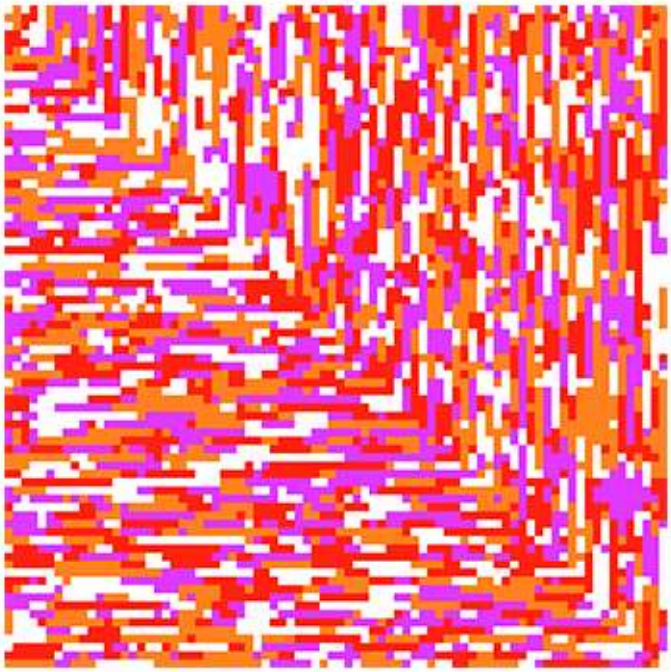
So I came up with a second solution which was to keep all the colours in one SVG file, but to first draw all the squares of a given colour before moving on to the next colour. In order to do so, instead of drawing the squares instantly as the ant was moving along, I saved them in different ArrayLists, one for each colour, and at the end I would draw the ArrayLists one after the other. This way I obtained slightly more occurate results :

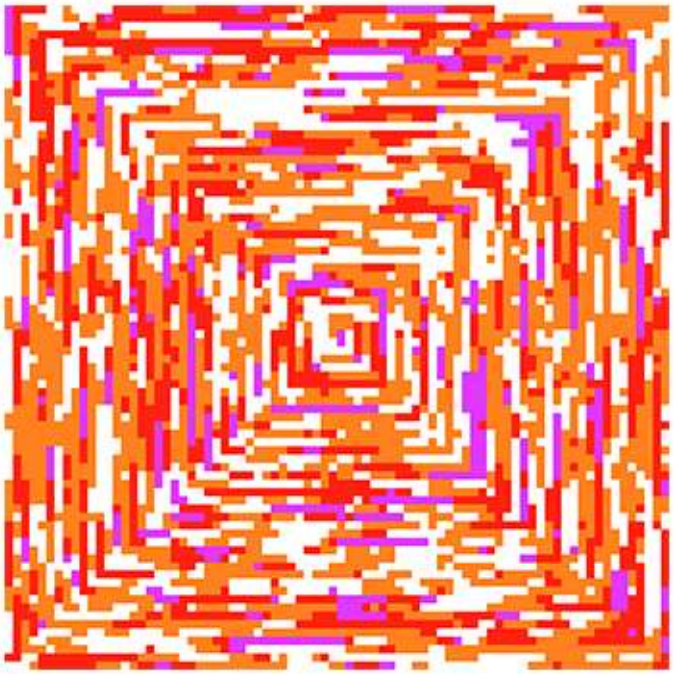




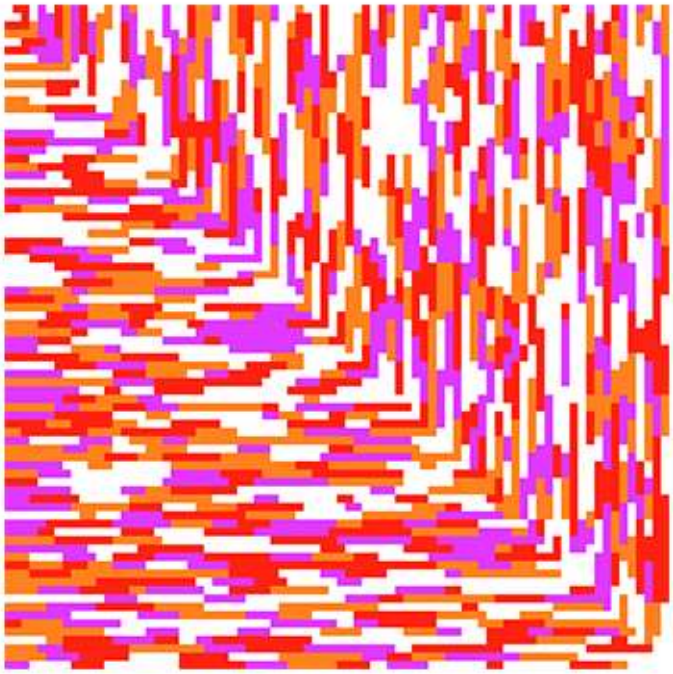


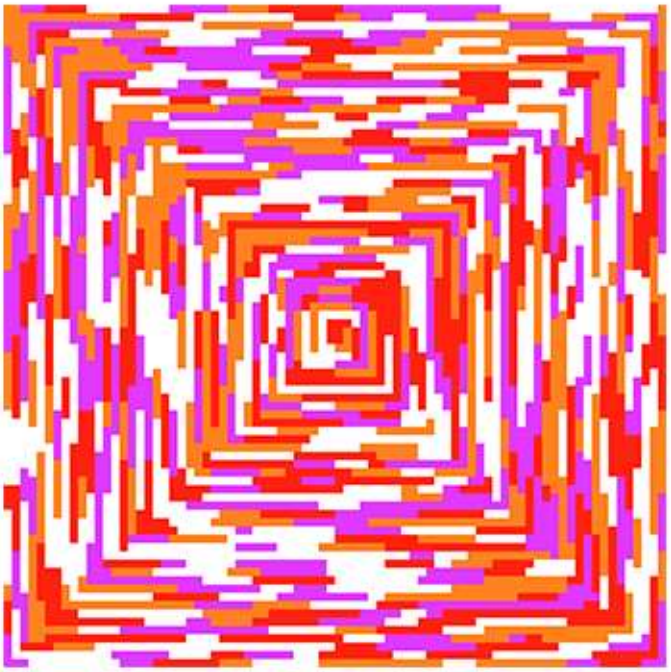
I have also tried another way of generating randomness, using Markov chains. So I wrote a program to generate a random Markov matrix with high diagonal values (i.e. high probability of keeping the same colour from one square to the next).
Markov Chains introduce assymetry between the different colours and each drawing has it's own repartition of colours. For example on the right you can see that violet had fewer chances of appearing than orange.





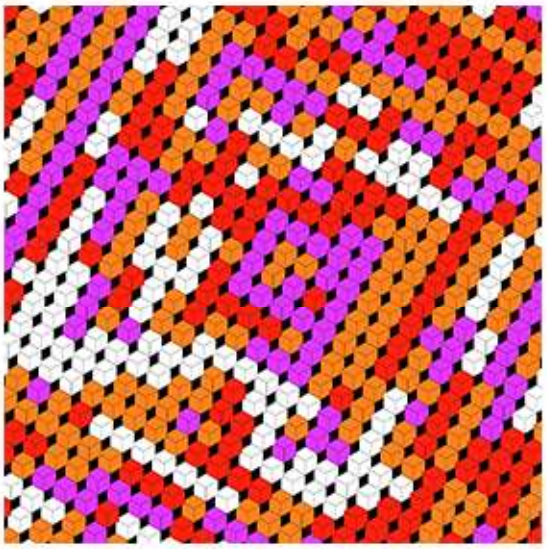
These are to be compared with my first drawings :



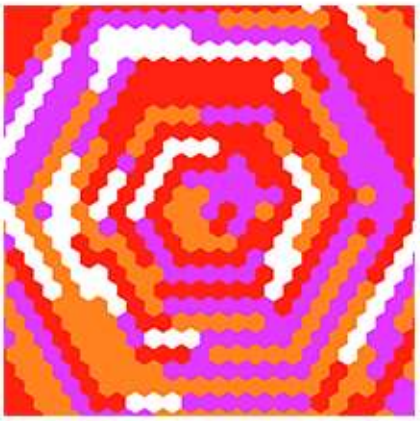


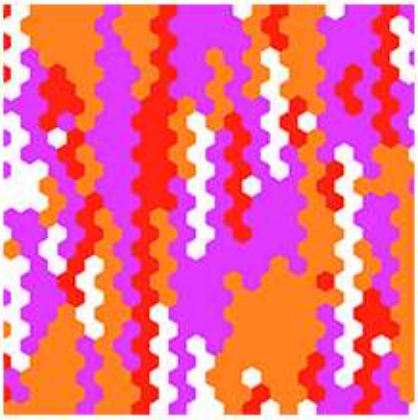
Next up I decided to switch the squared tiling for a hexagonal one, for the sake of diversity. The randomness is still made through Markov chains.

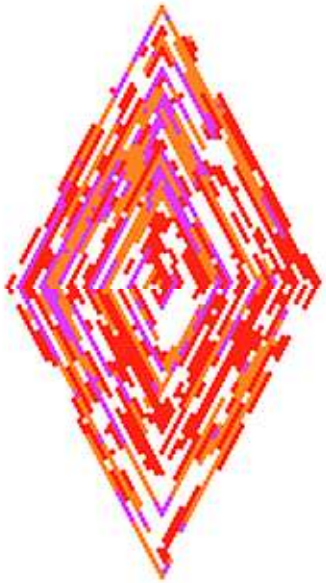
In the process of changing my code, I first miscalculated the position of the hexagons, but it was cool looking nonetheless :

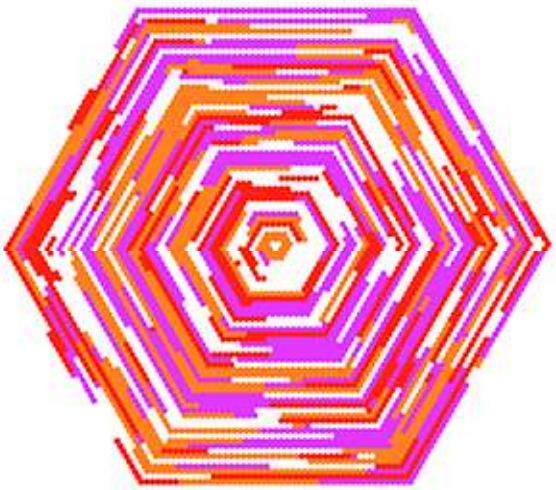


Once the hexagonal tiling was working properly, I first made a squared spiral, and then a hexagonal one, which makes more sense. I also added a linear version lately.

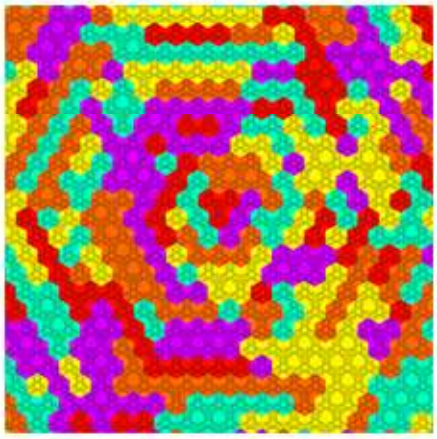


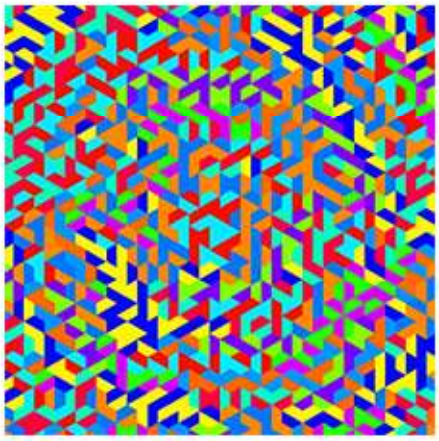




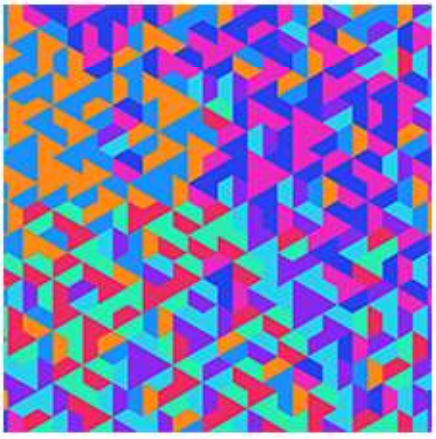
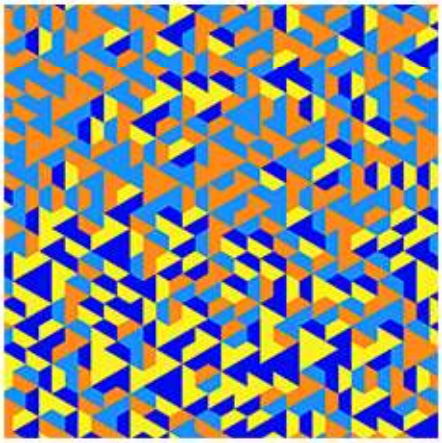


Finally I wanted to give these a nicer look, so I have tried several things : first I added a motive inside the hexagons, then I gave them a gradient of brightness to simulate some 3D, and finally I cut the hexagons in two and coloured each half differently, which allows interesting patterns to emerge.





Then I changed the randomness once again, making it depend on the position inside the canvas.
To do so I pick a random value following a gaussian distribution, then offset it by the angle of the current position, and finally convert it into an integer (modulo the number of colours). This way colour n°0 is the most likely to appear in the first quadrant, colour n°1 in the second quadrant, and so on, as you can see in the images below.

And the last version I added was to put multiple ants at the same time on the canvas, each with a given colour, and allow them to walk completely randomly.

I also made a user interface for the last one, allowing you to play with all the parameters of the drawing and generate very diverse instances.