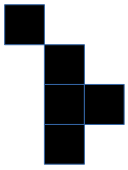
# Self-Organizing Fractal Noise

by Sven Nilsen, 2020

*In this paper I describe how I came up with the idea of self-organizing fractal noise.*

It started by me watching a Star Trek Voyager episode about an alien robot being fixed by chief engineer B'Elanna Torres. The episode started by filming through the eyes of the robot. At the bottom of the picture, there was a cool pattern of dynamic boxes, kind of like the following:



The pattern kept changing dynamically, but only one cell at a time.

I thought it was cool and got curious about the feeling you get when a single cell is flipped at a time. The more I thought about it, the more curious I got until I started coding. I started thinking about how to program rules such that a new filled cell would only occur when surrounded by filled cells. First, I played with the condition that some filled cell exists among the four neighbor cells:

```
map[ip][j] || map[i][jp] || map[im][j] || map[i][jm]
```

I tried changing the value probabilistically based on this condition. For example, I made it more likely to fill the mutated cell than emptying it, in order to control the ratio of filled versus empty cells.

Somehow, I ended up with this, where `map[ip][j]` is a shorthand for `if map[ip][j] {1} else {0}`:

```
count := map[ip][j] + map[i][jp] + map[im][j] + map[i][jm]

map[i][j] = random() > count / 4
```

I noticed that this resulted in local regions of stable chess patterns. At first, when you see the formula, it looks as if it is asymmetric. This is because `count` depends only on the neighboring cells.

Despite the simple formula, it secretly hides a symmetric pattern. If all the surrounding cells are filled, then the mutated cell will be empty. However, when the surrounding cells are empty, there is a 100% chance that a random number in the unit interval is greater than zero. So, this causes the stable chess patterns.

Although mutation is probabilistic, the algorithm converges toward two possible solutions for each cell where the local region around the cell is stable. Mathematically, this phenomena is very interesting. In spaces where you ignore coordinates, there is no way to tell "where" a cell is. Coordinates are used to map from outside space to the inside. However, when you introduce this algorithm with an extra layer of information, containing a bit for each cell, it is possible to, sort of, tell whether a cell has an "even" or "odd" coordinate. It is kind of a "dynamic coordinate system" that has some numerical properties.