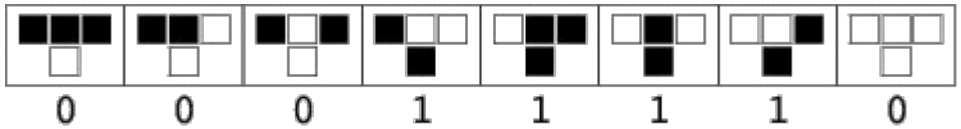
1D Cellular Automata

Neil Babson

Two state, neighborhood three CAs

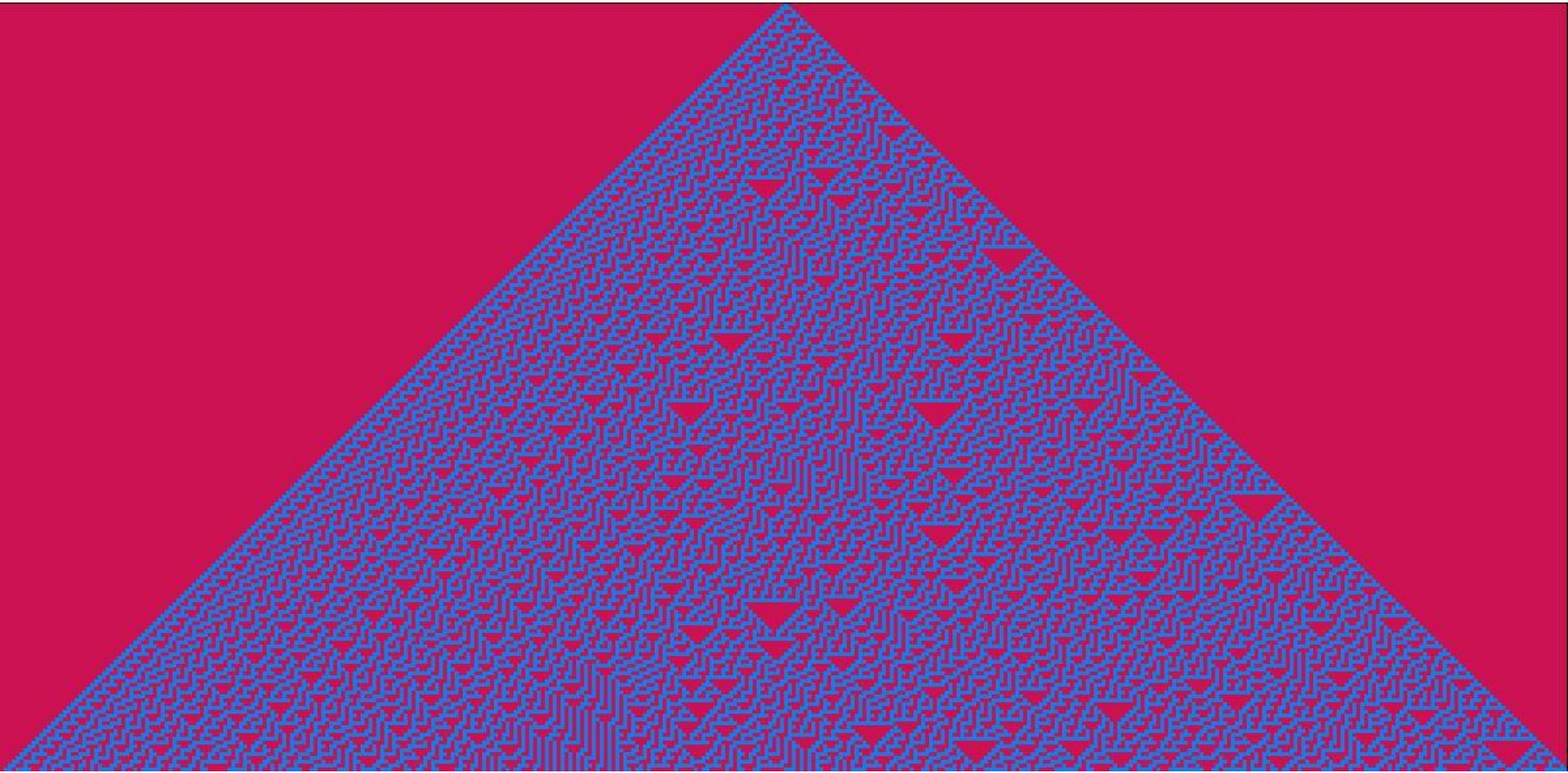
Each cell of a two state CA can take the value 0 or 1, and the cell's state in the next generation depends only on its current state and that of its two immediate neighbors. Therefore the CA's behavior can be completely described by a lookup table encoded as an eight digit binary number.



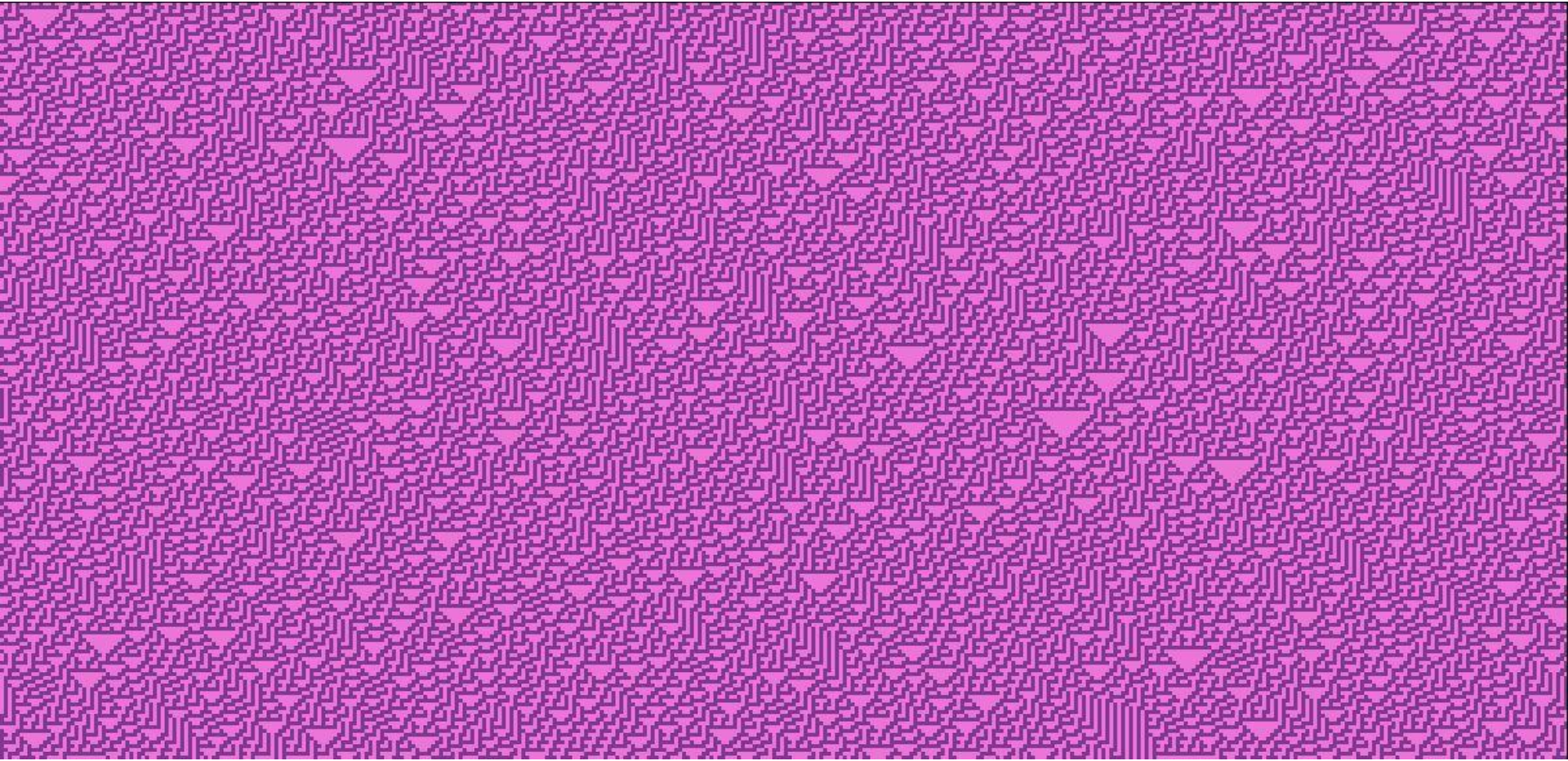
Here, for example, is rule 30 (image taken from mathworld.wolfram.com/ElementaryCellularAutomaton.html 12/1/2015).

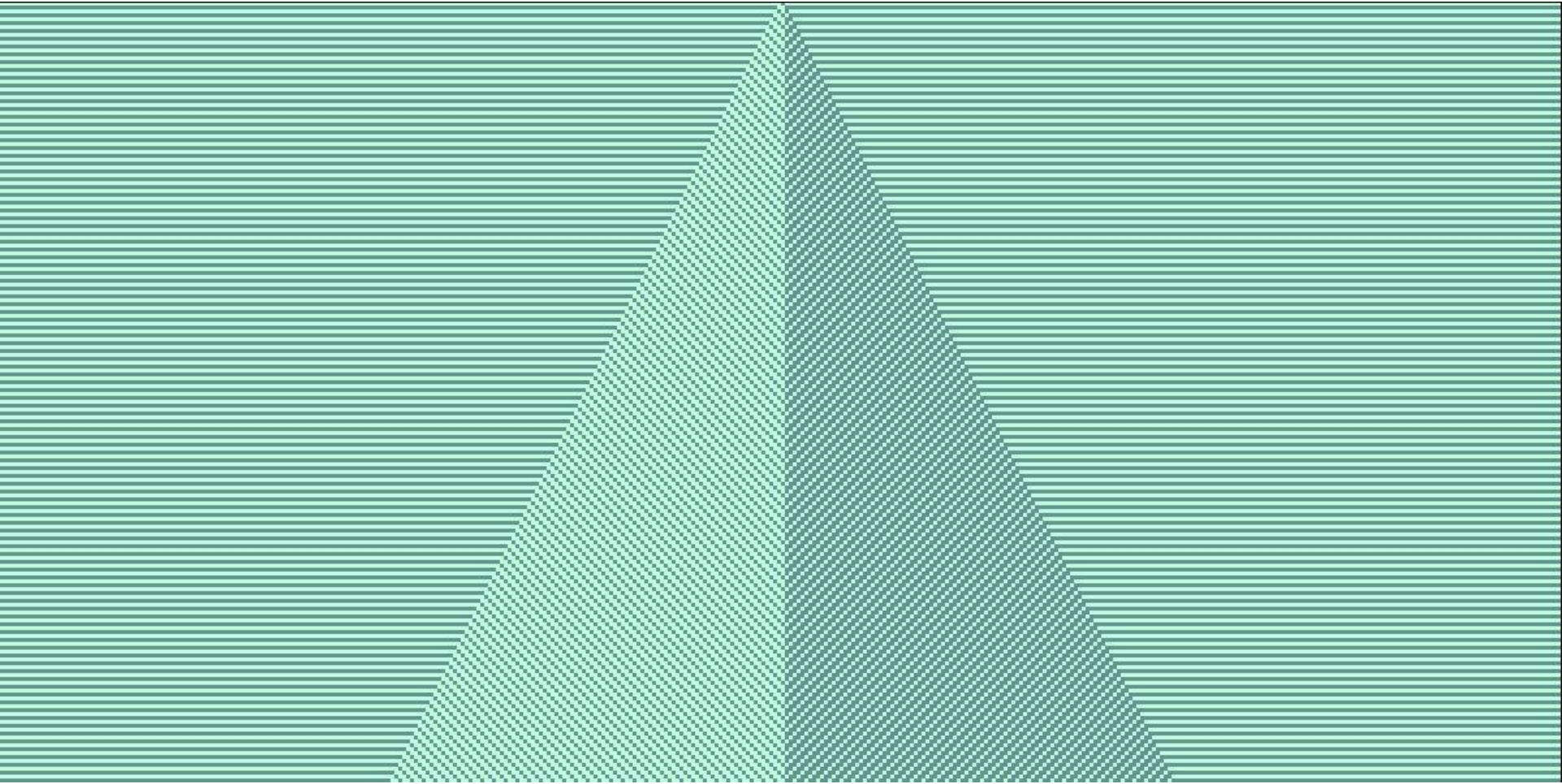
There are a total of 2^2^3 = 256 of these elementary CAs.

Rule 30



Rule 30





Neighborhood five CAs

The rule for a CA with a neighborhood of five has 2^5 = 32 bits.

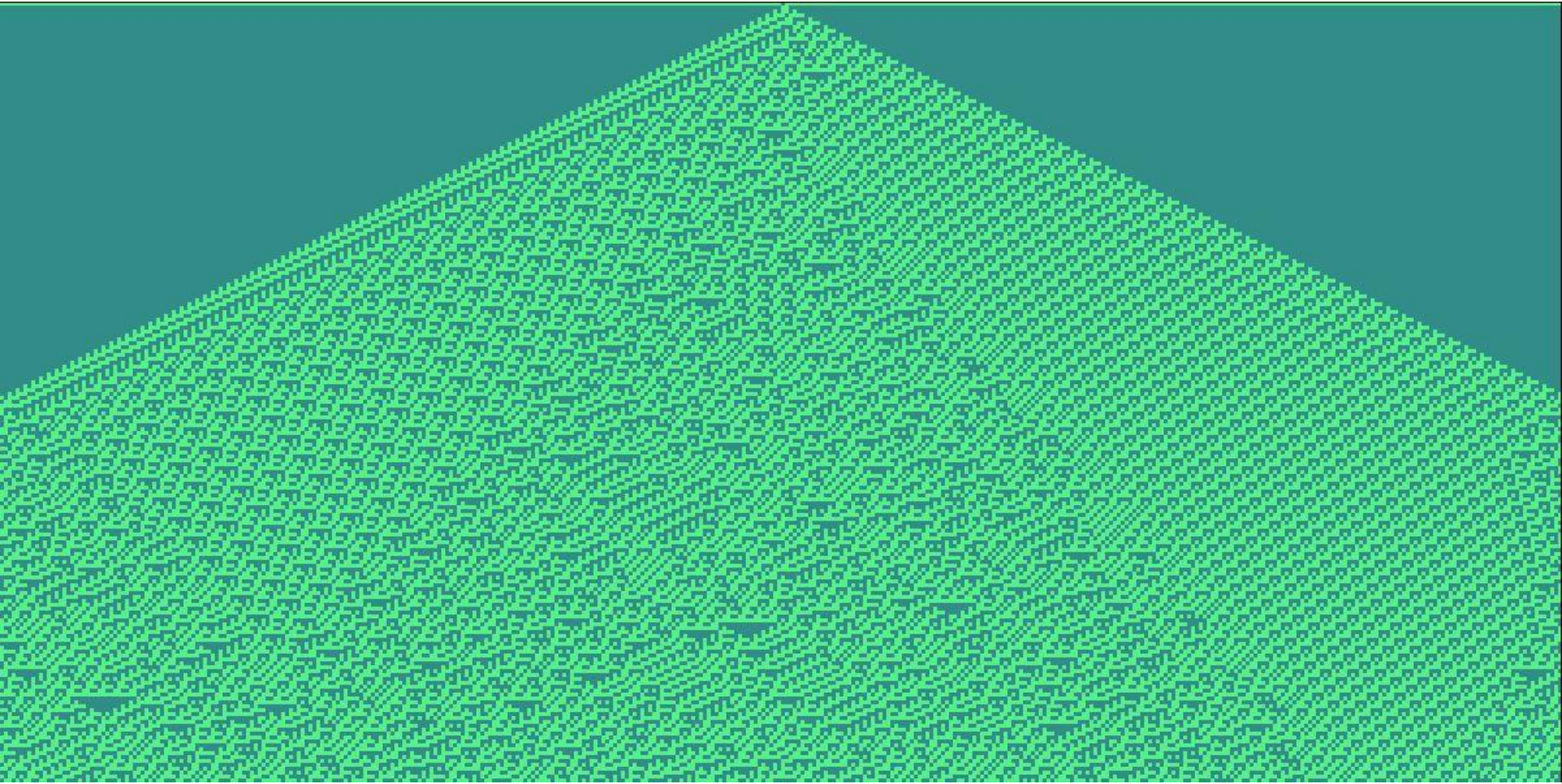
The number of possible such rules is 2^2^5 = 4,294,967,296.

Randomly generated rules can sample this huge space of possible CAs.

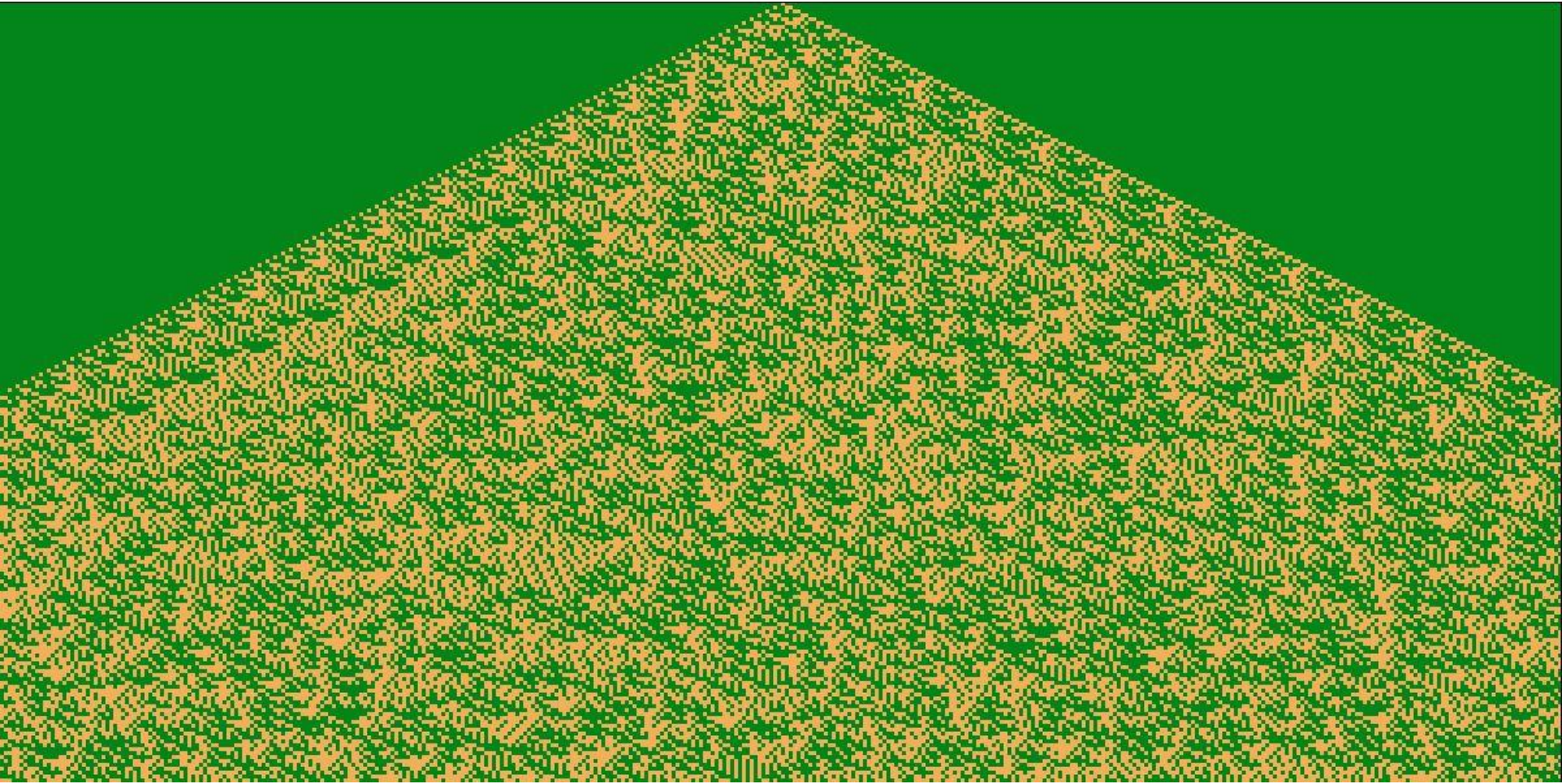
Two image files are created for each rule, one with a randomly populated first generation, and one in which all the cells of the first generation are dead (0) with the exception of the middle cell, which is alive (1).

All of the CAs run for 200 generations and wrap around at the edges.

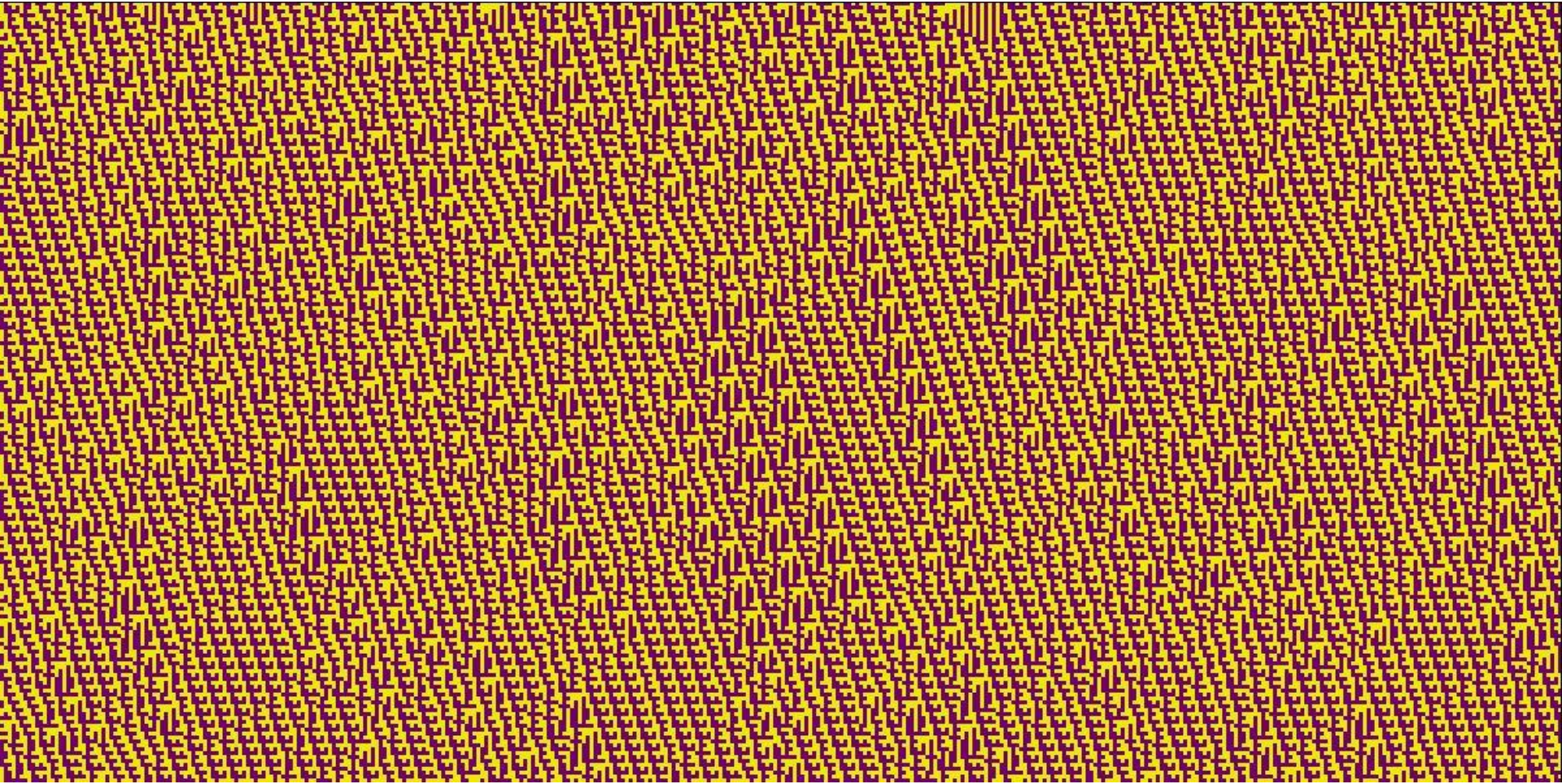
Rule 2,955,355,279



Rule 3,483,459,914



Rule 2,973,282,783



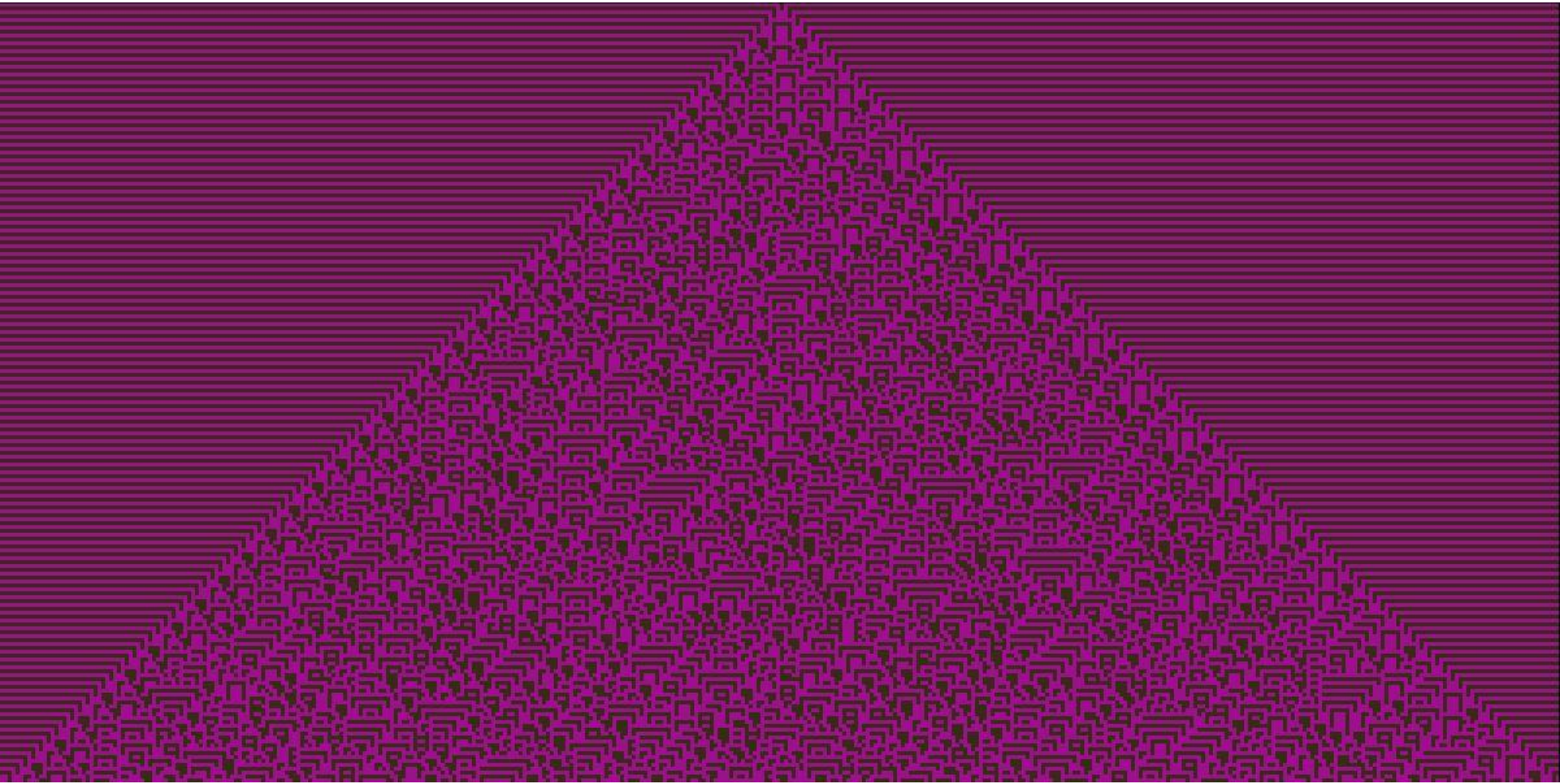
Neighborhood seven CAs

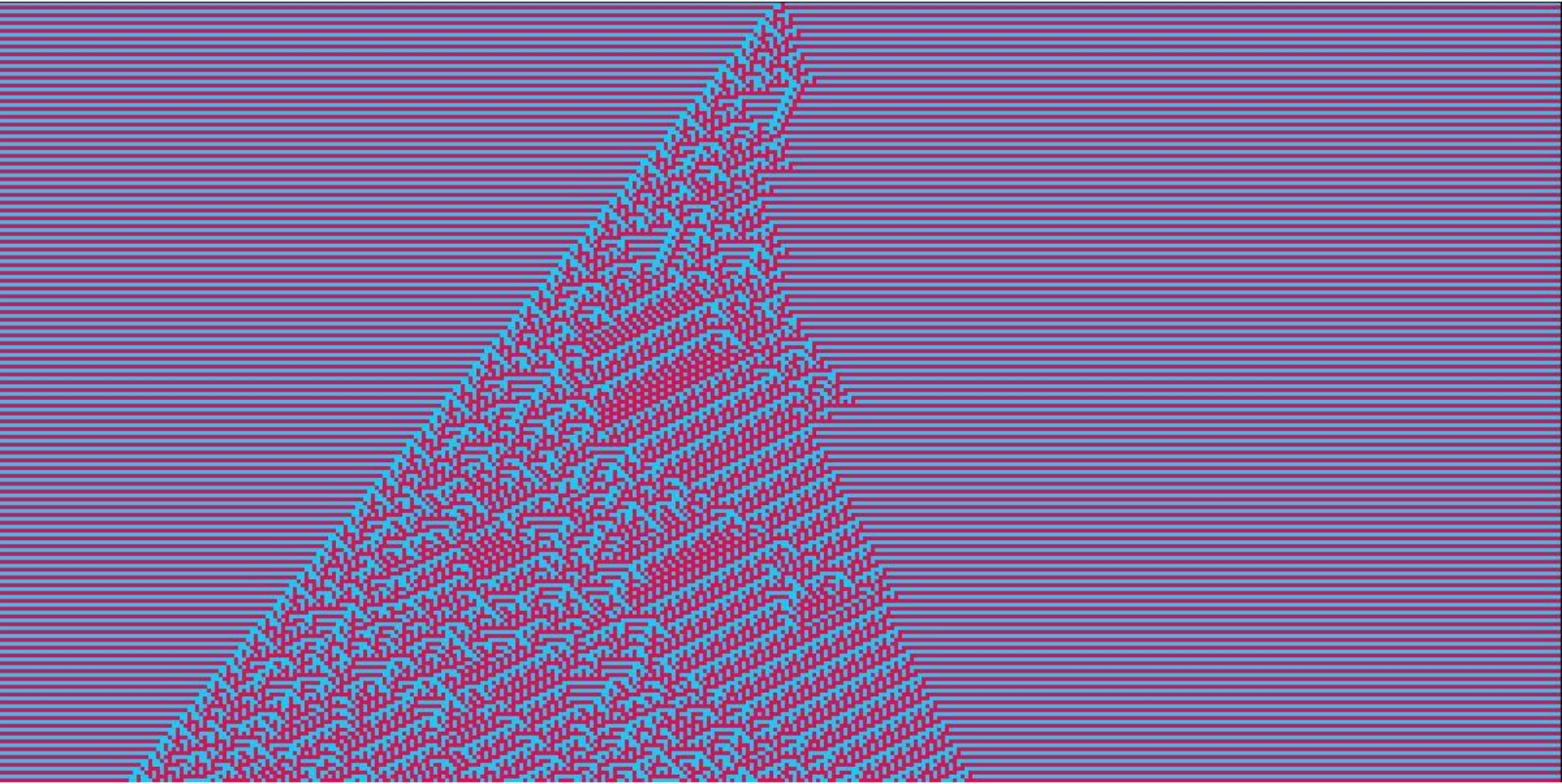
A neighborhood seven rule has a length of 2^7 = 128.

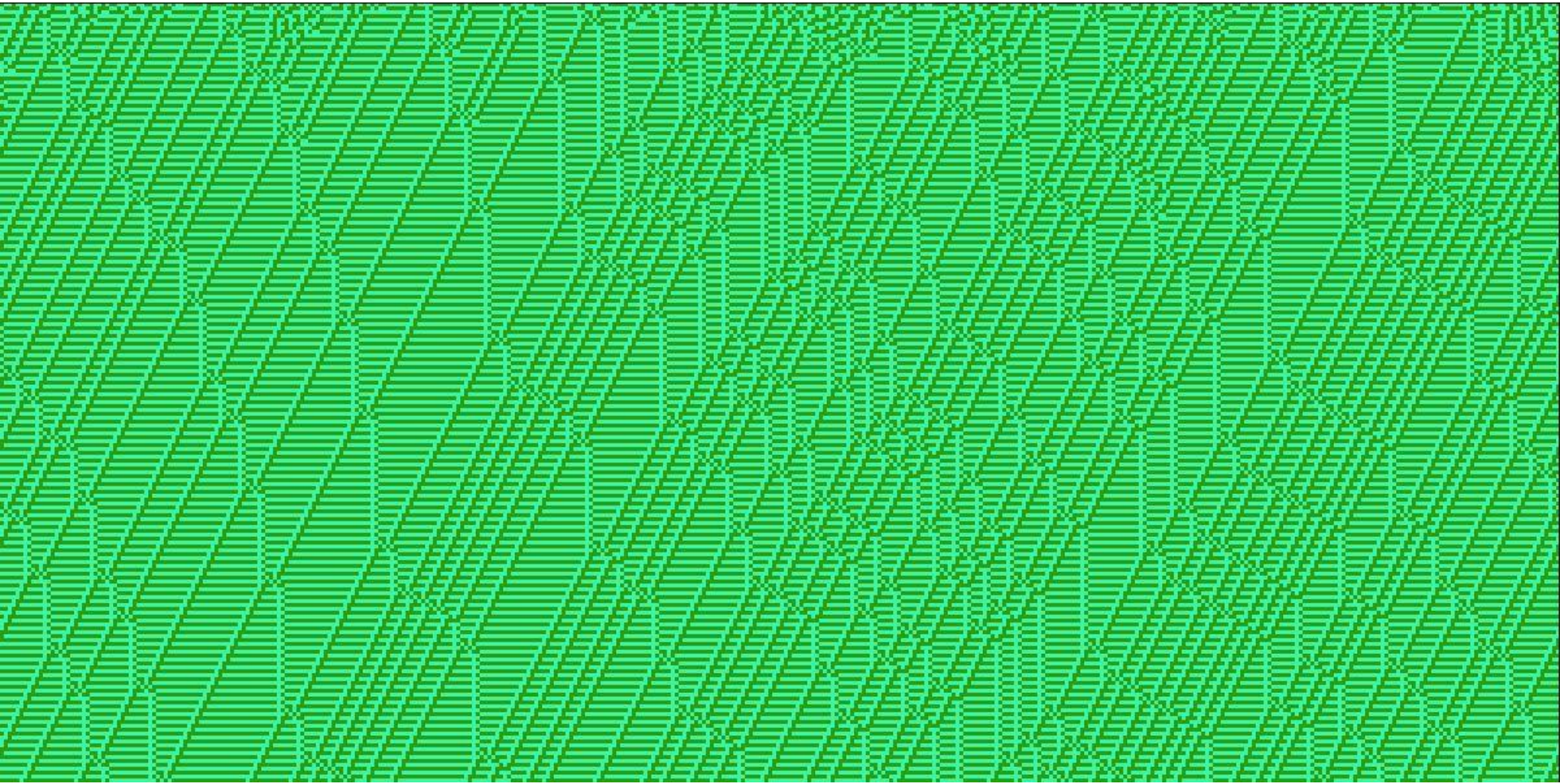
The number of two state neighborhood seven CAs is 2^2^7 ≃ 3.4028e+38.

maxBound::Int ≃ 9.223e+18

The program doesn’t accurately record the rule number for these CAs (because of overflow), but a separate file is made recording each rule as a list in case one wanted to recreate a particular example.





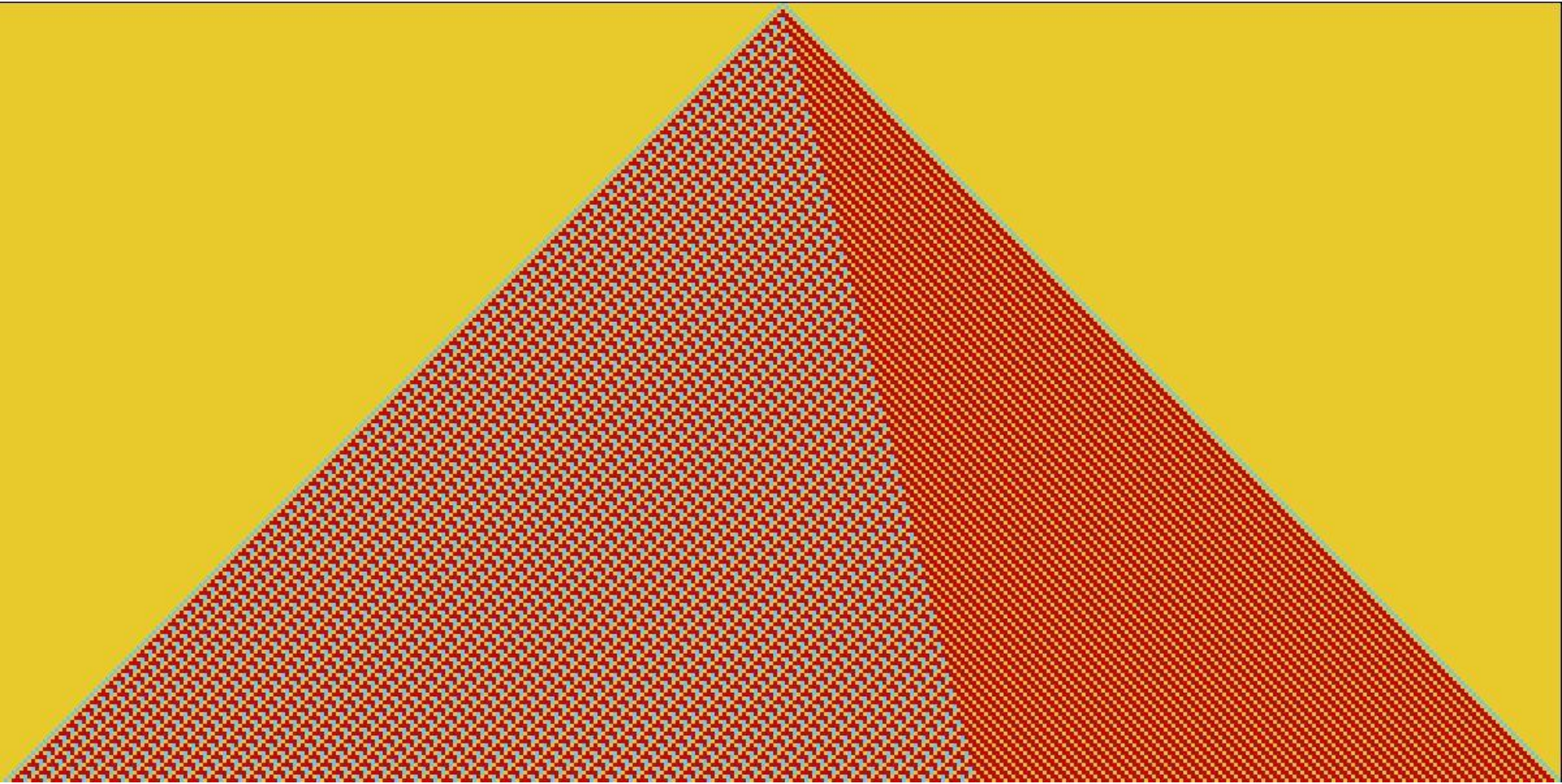


Three state neighborhood three CAs

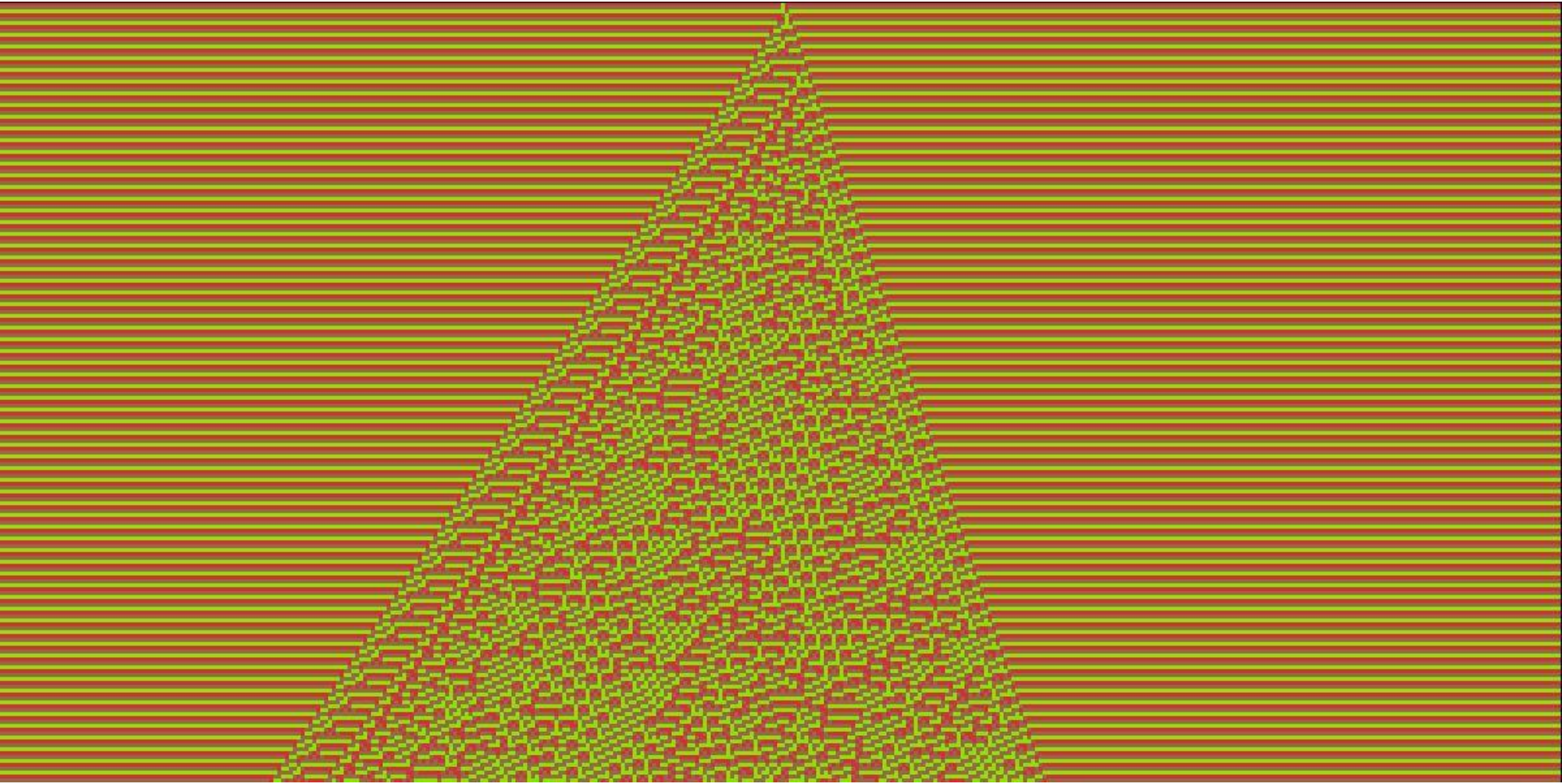
The rule for a three state CA of neighborhood three is a list of 3^3 =27 0s, 1s, and 2s.

There are 3^3^3 = 7,625,597,484,987 such CAs.

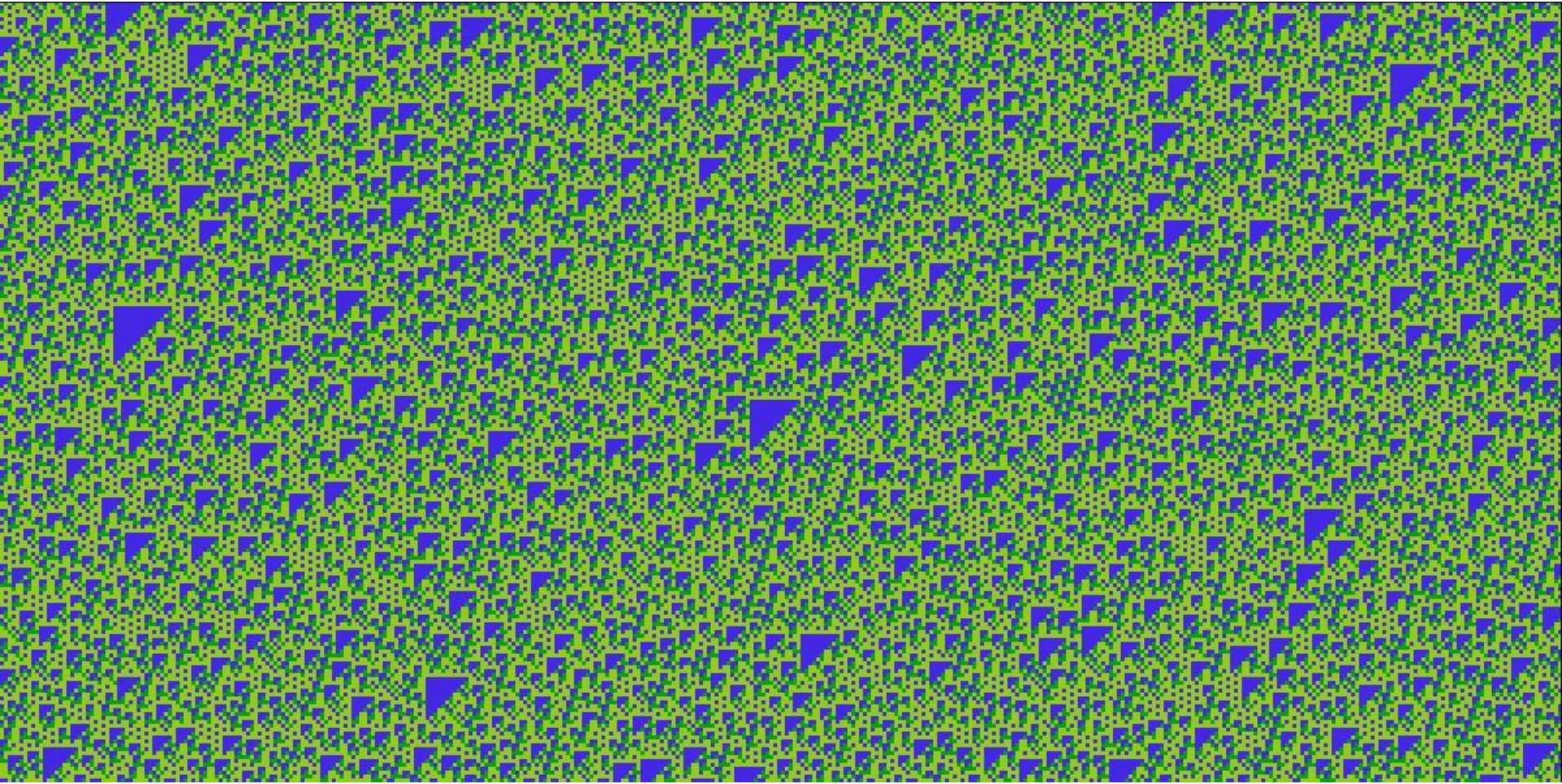
Rule 5,031,131,187,684



Rule 3,849,333,813,953



Rule 5,441,981,128,445

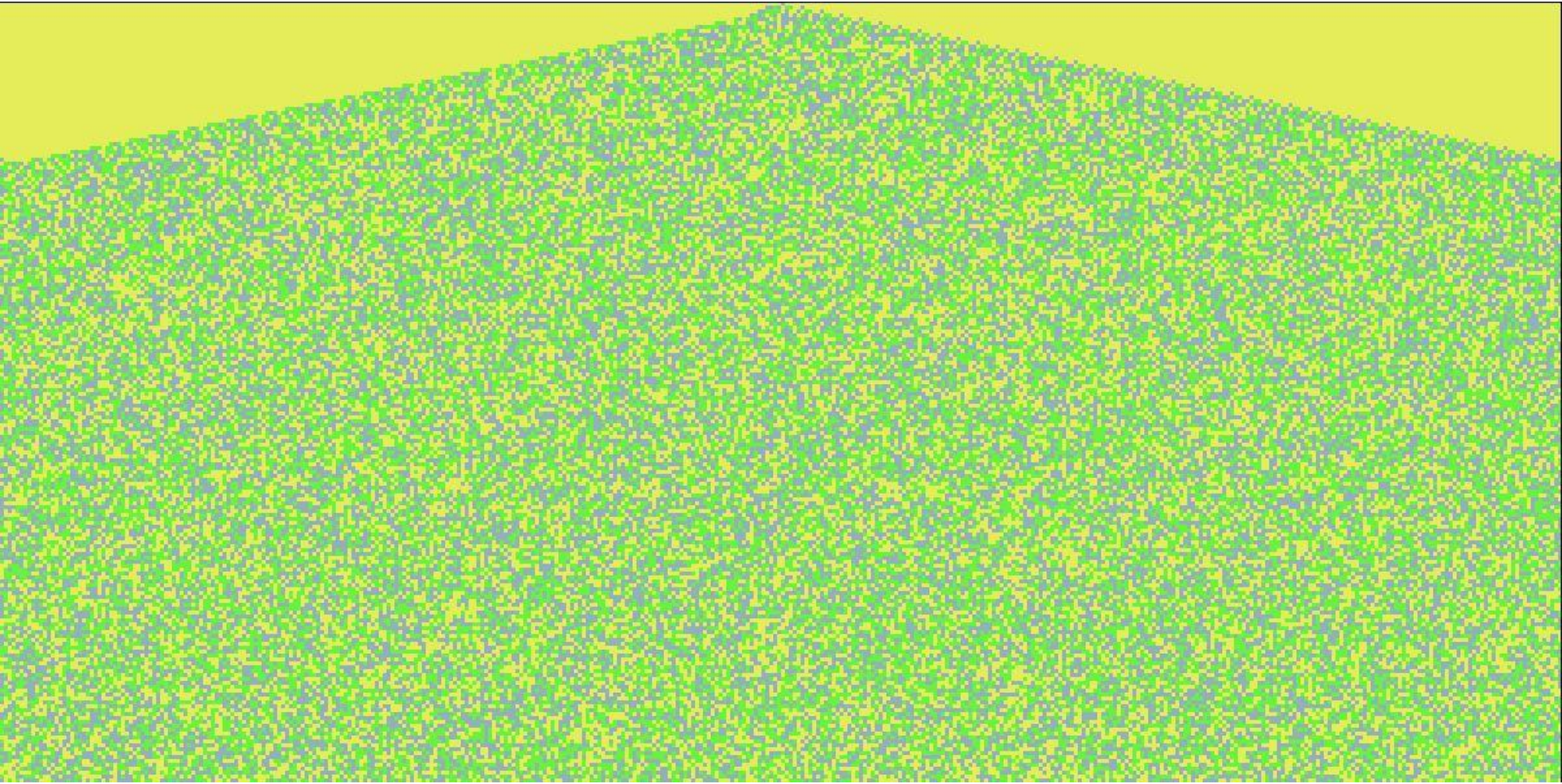


Arbitrary state and neighborhood size

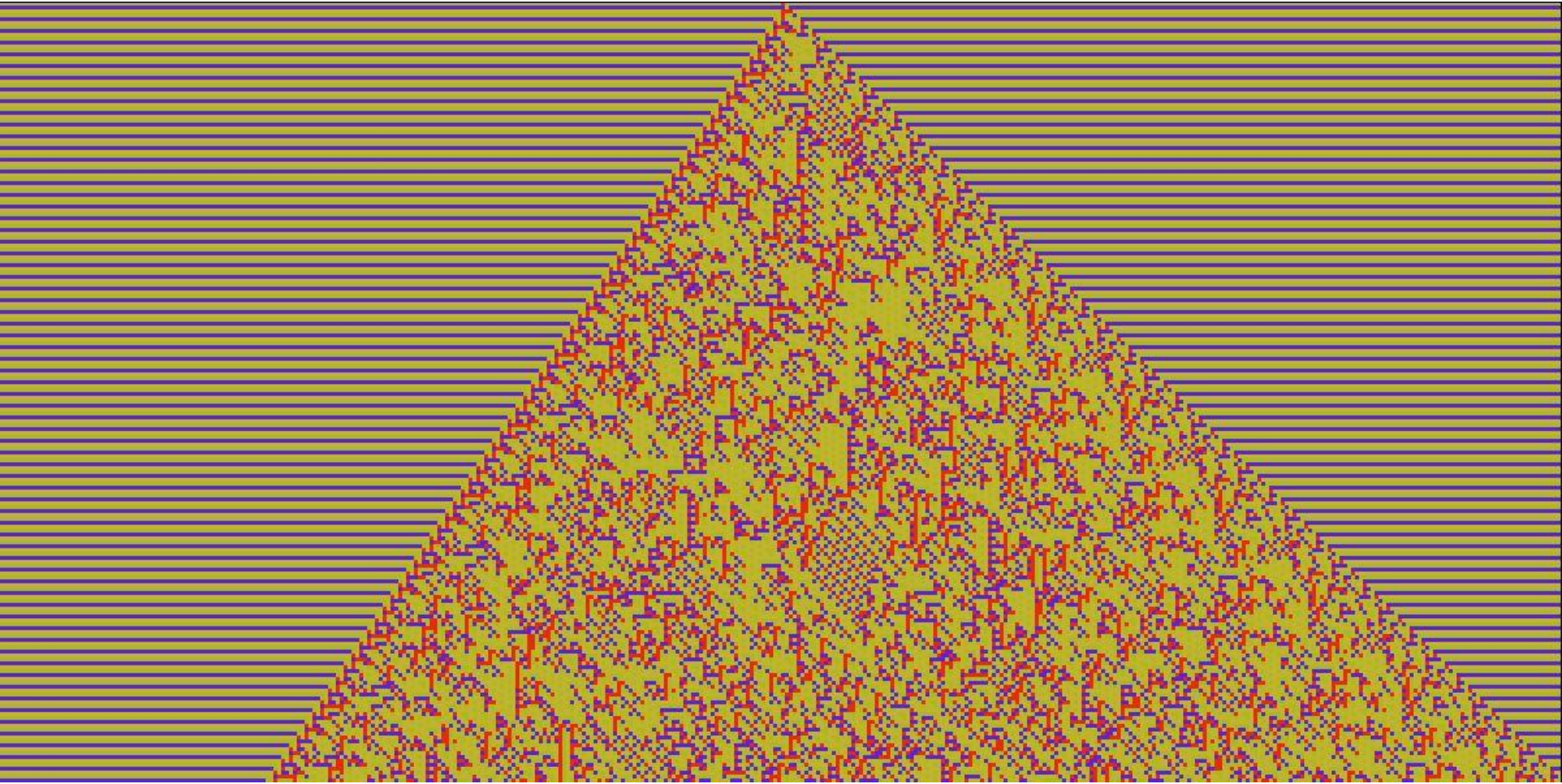
The program allows the user to specify any number of states and neighborhood size.

As states and/or neighborhood size increase the complexity of the rules causes the images to trend towards colorful static.

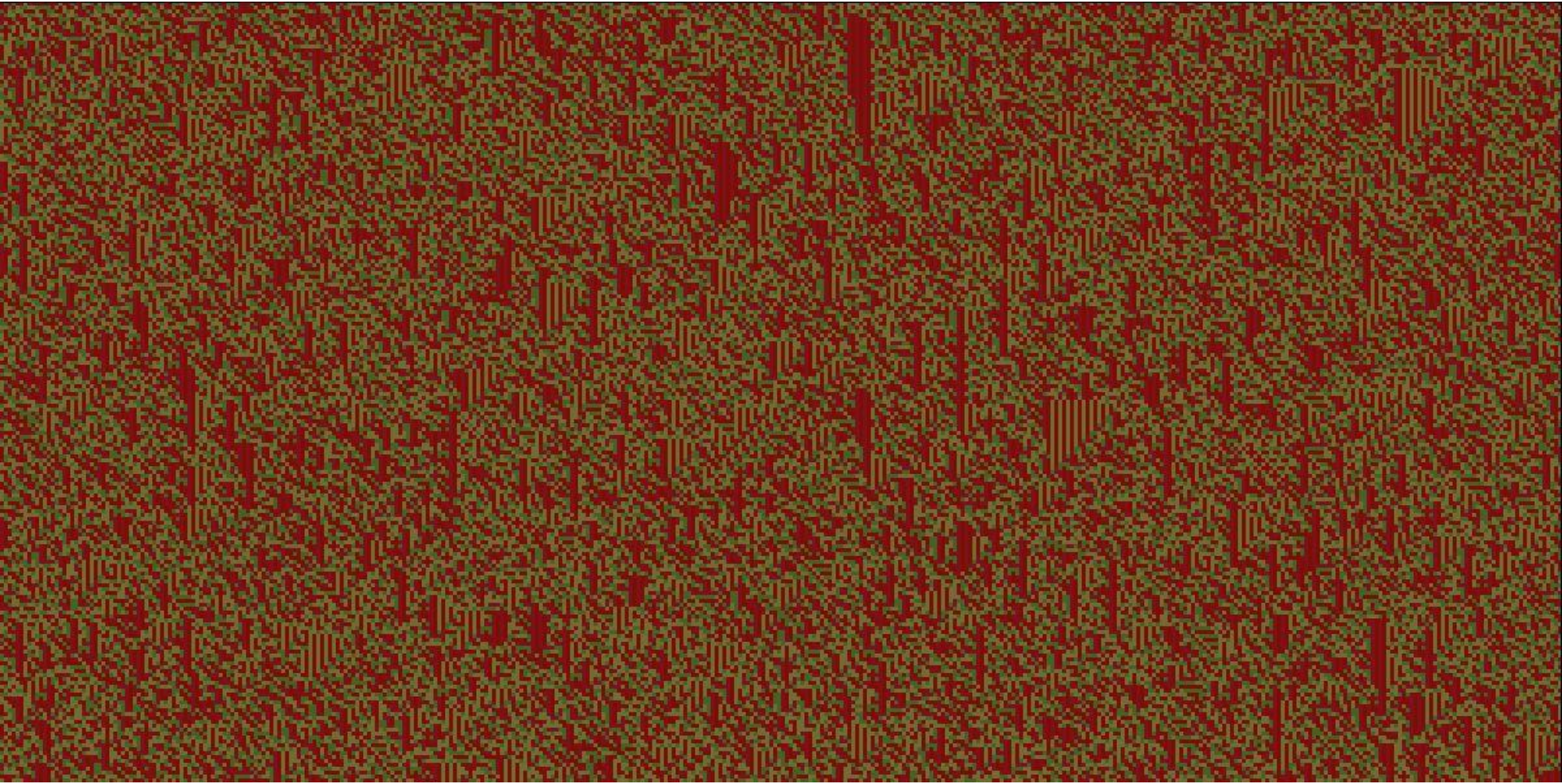
Three states, neighborhood 11 (Rule has length of 177,147)



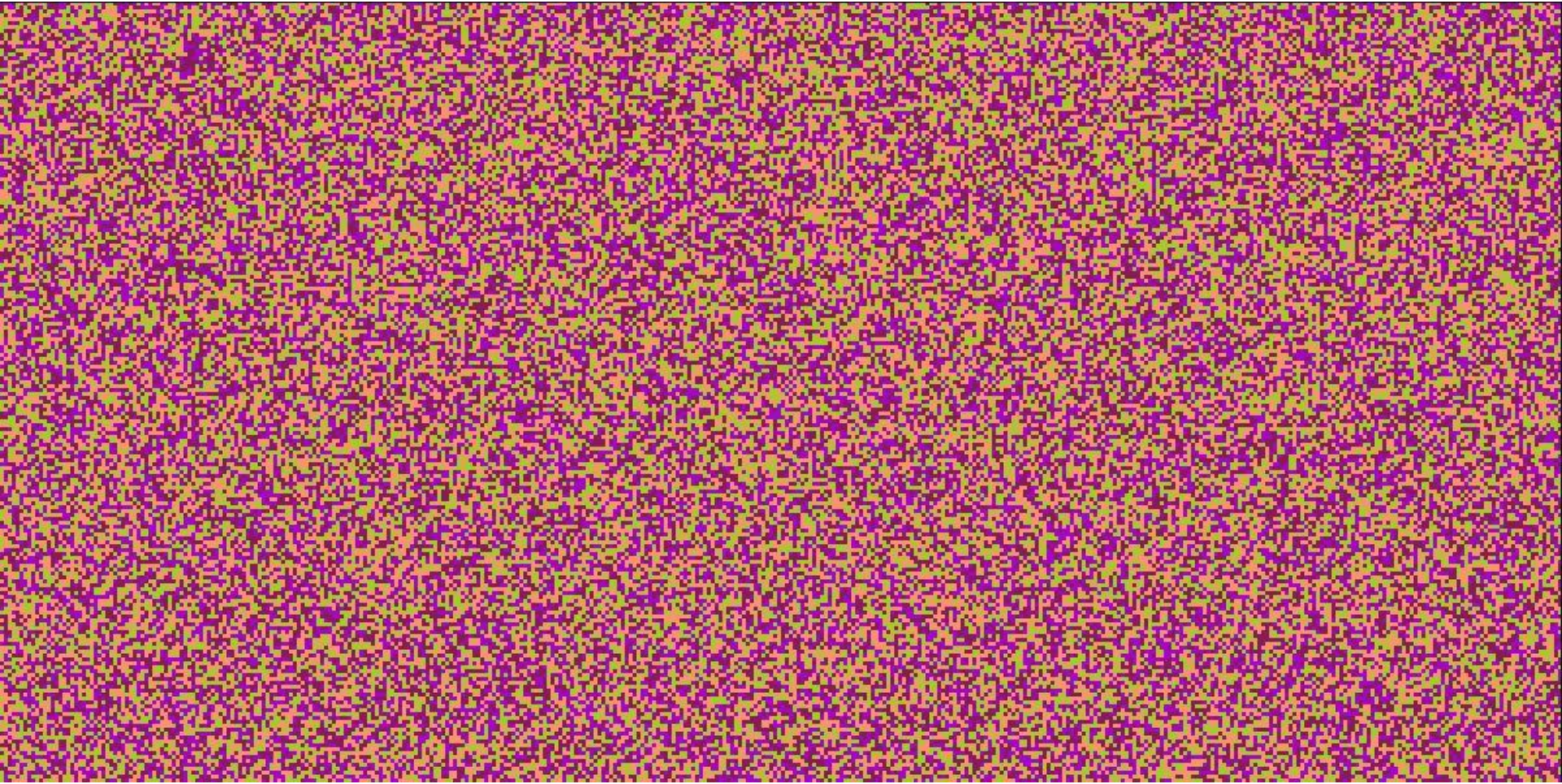
Four states, neighborhood 3



Four states, neighborhood three



Four states, neighborhood seven

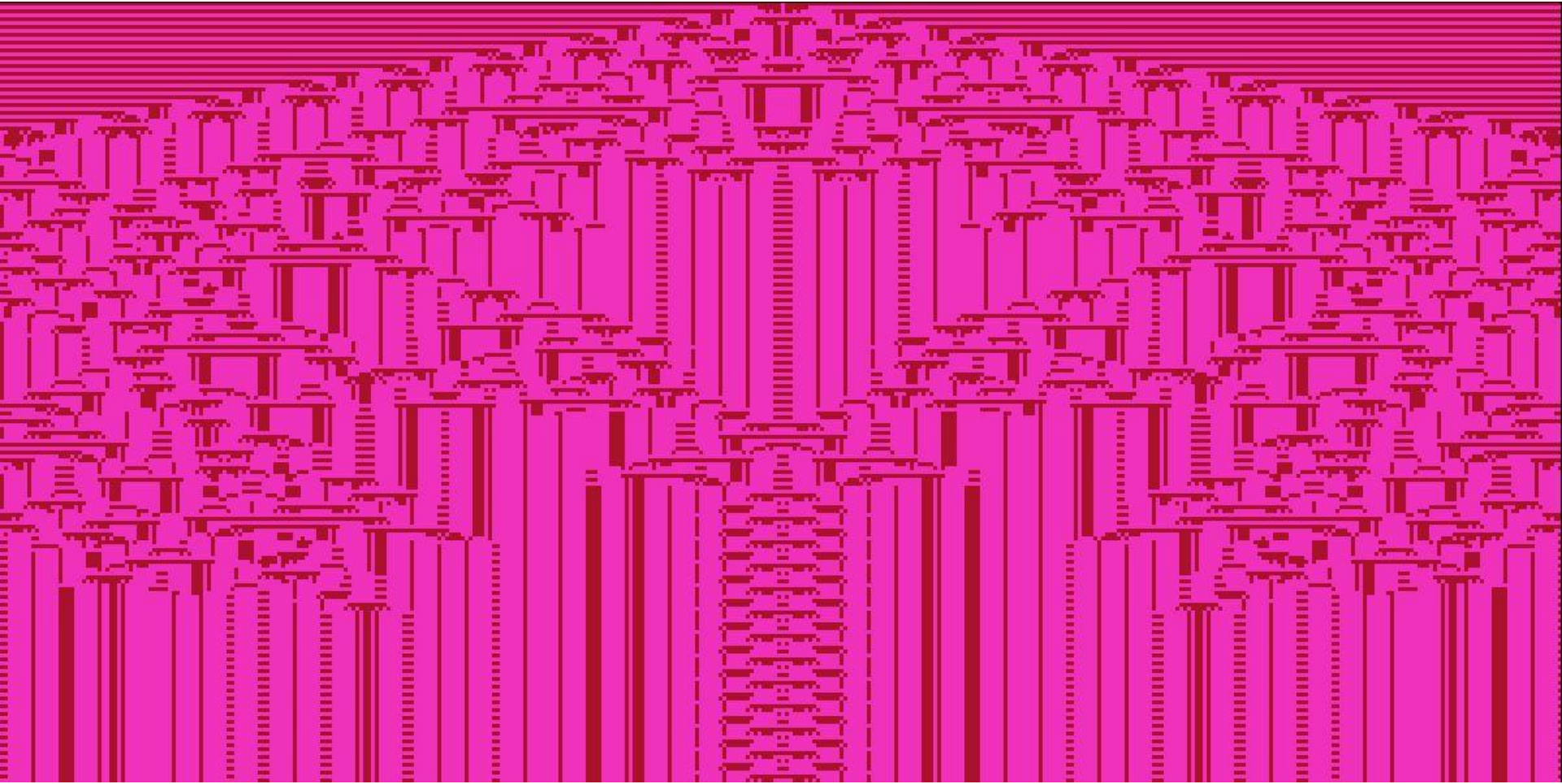


Population density rules

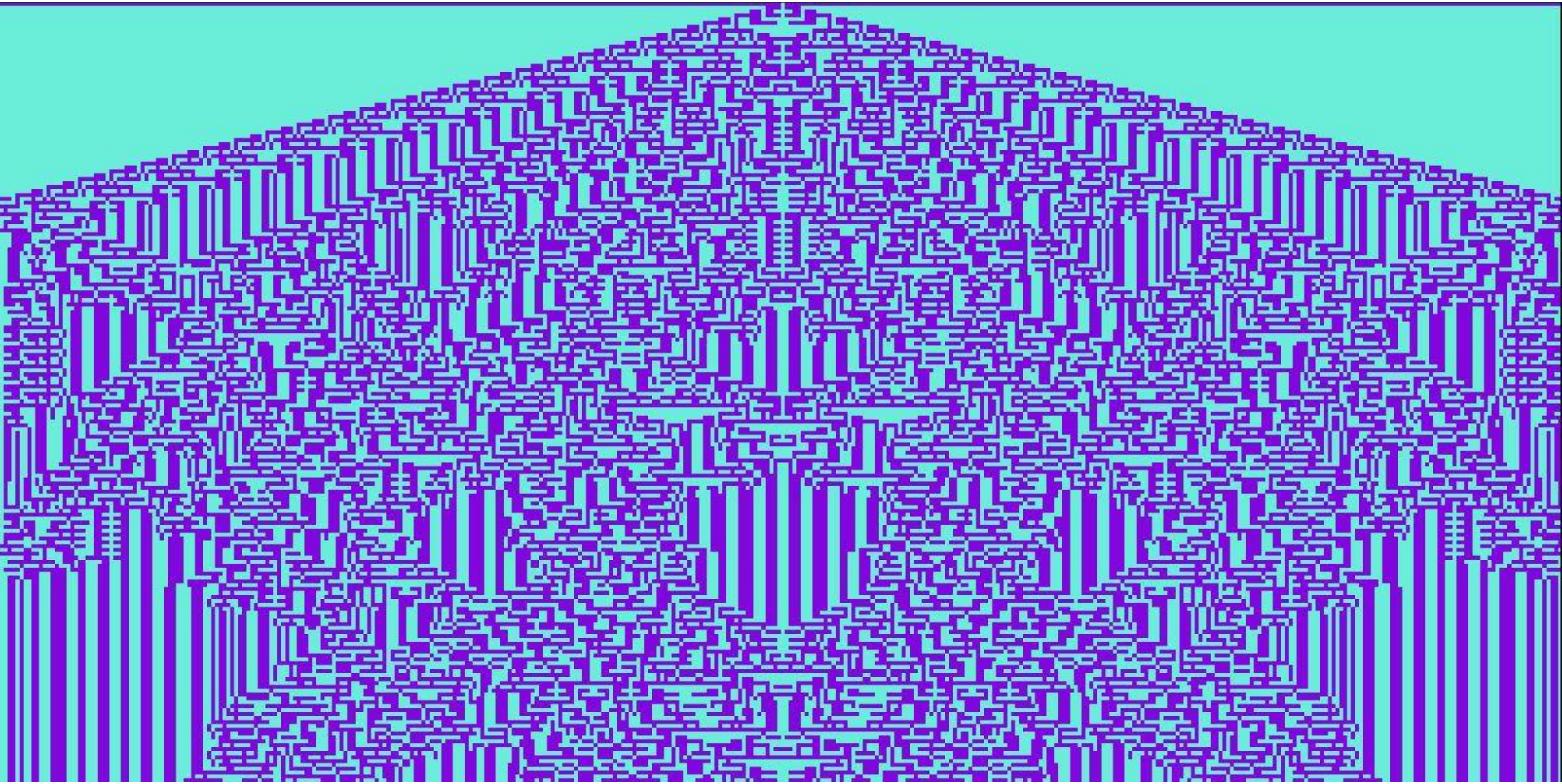
An alternate CA rule decides whether a cell lives in the next generation based on the number of live neighbors it had in the previous generation.

A rule consists of two binary lists: the ‘survive’ list and a ‘born’ list. A 1 at index n in the ‘survive’ list means that a living cell with n neighbors will survive to the next generation, and a 1 at index m in the ‘born’ list means that a dead cell with m neighbors will be born in the next generation.

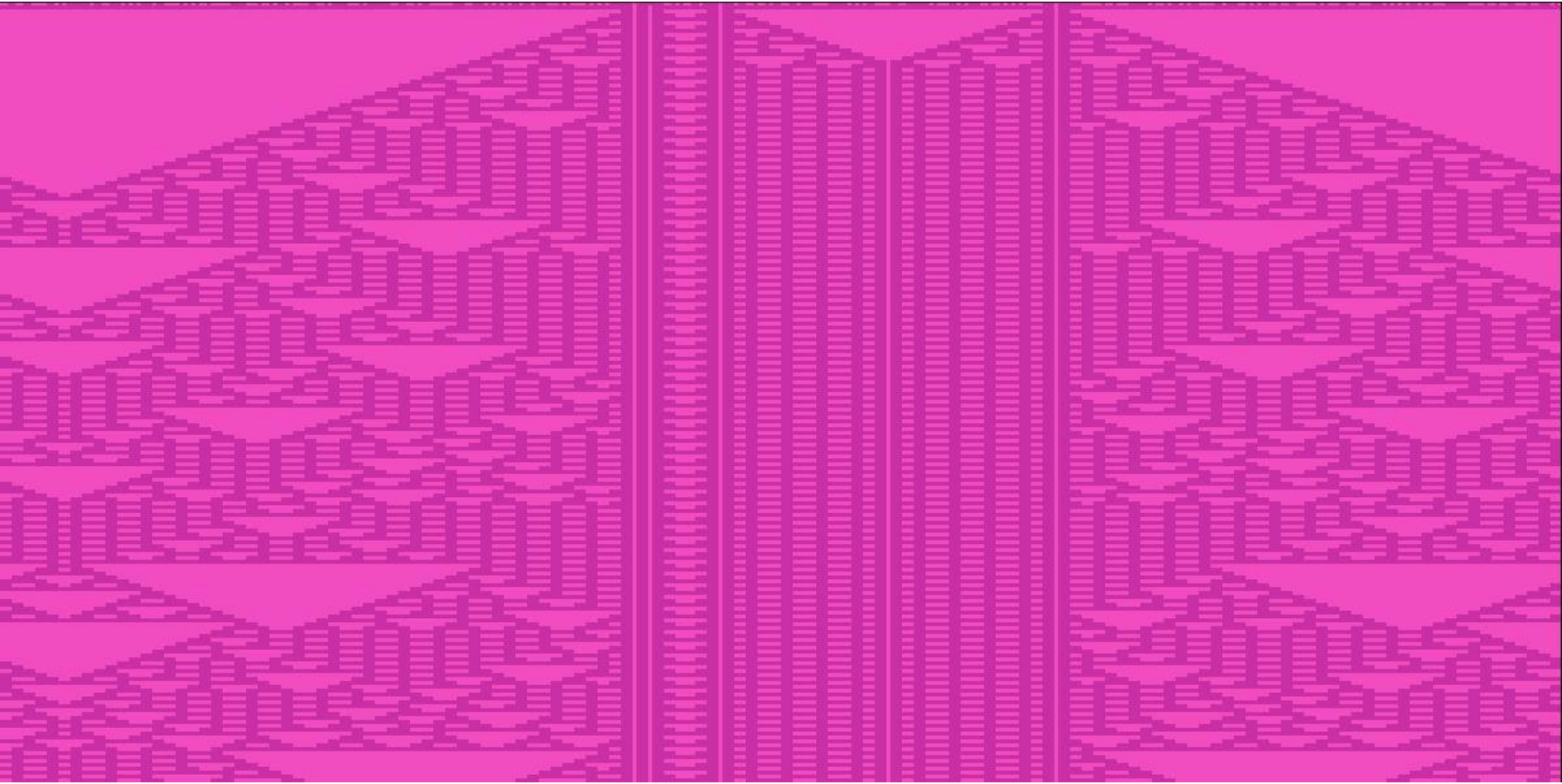
Neighborhood 13; Survive Rule 3416; Born Rule 4470



Neighborhood 9; Survive Rule 81; Born Rule 353



Neighborhood 7; Survive Rule 126; Born Rule 62



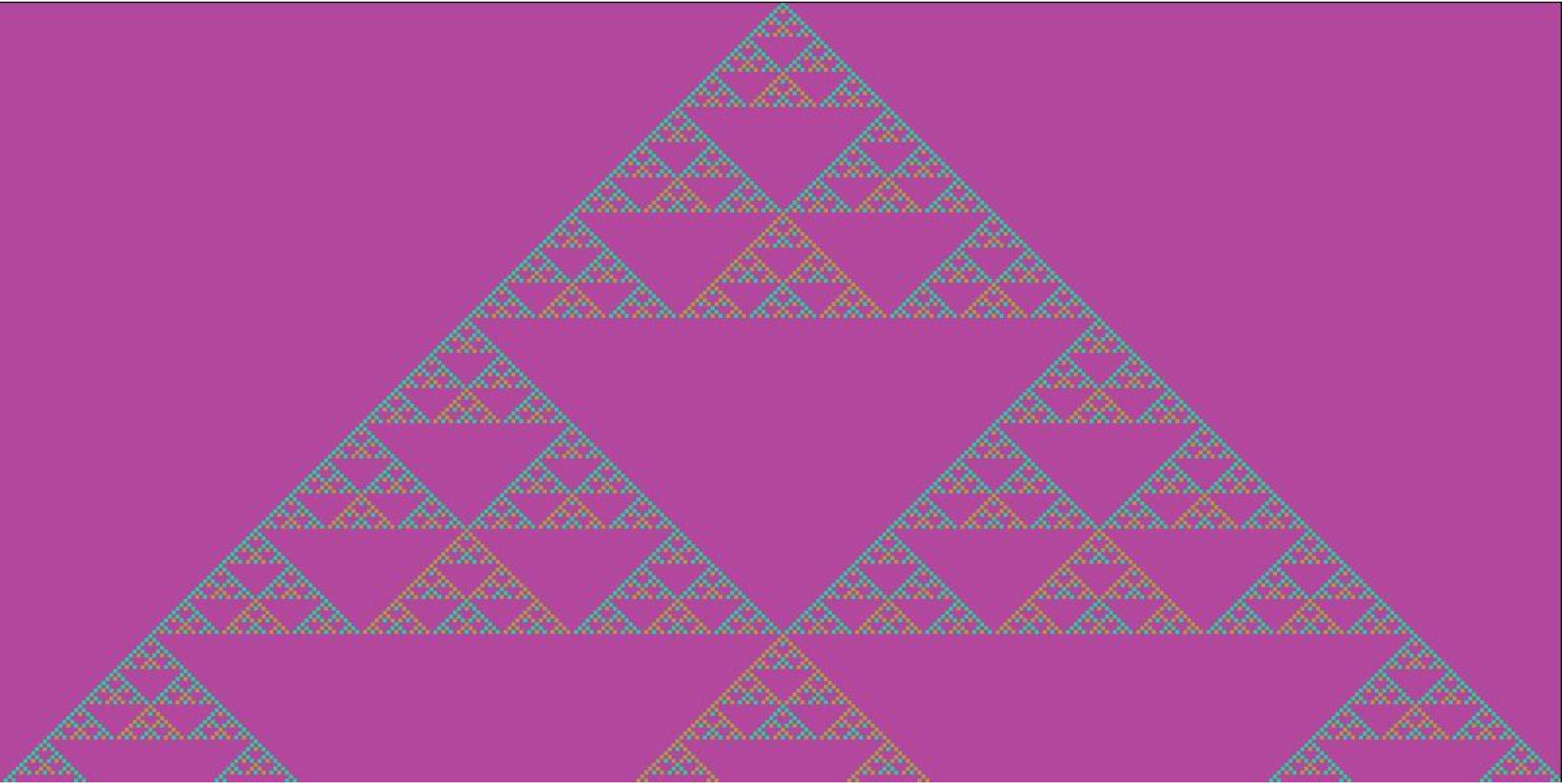
User designed CA rules

The program also allows the user to generate a CA of arbitrary neighborhood size and number of states, by specifying the rule number.

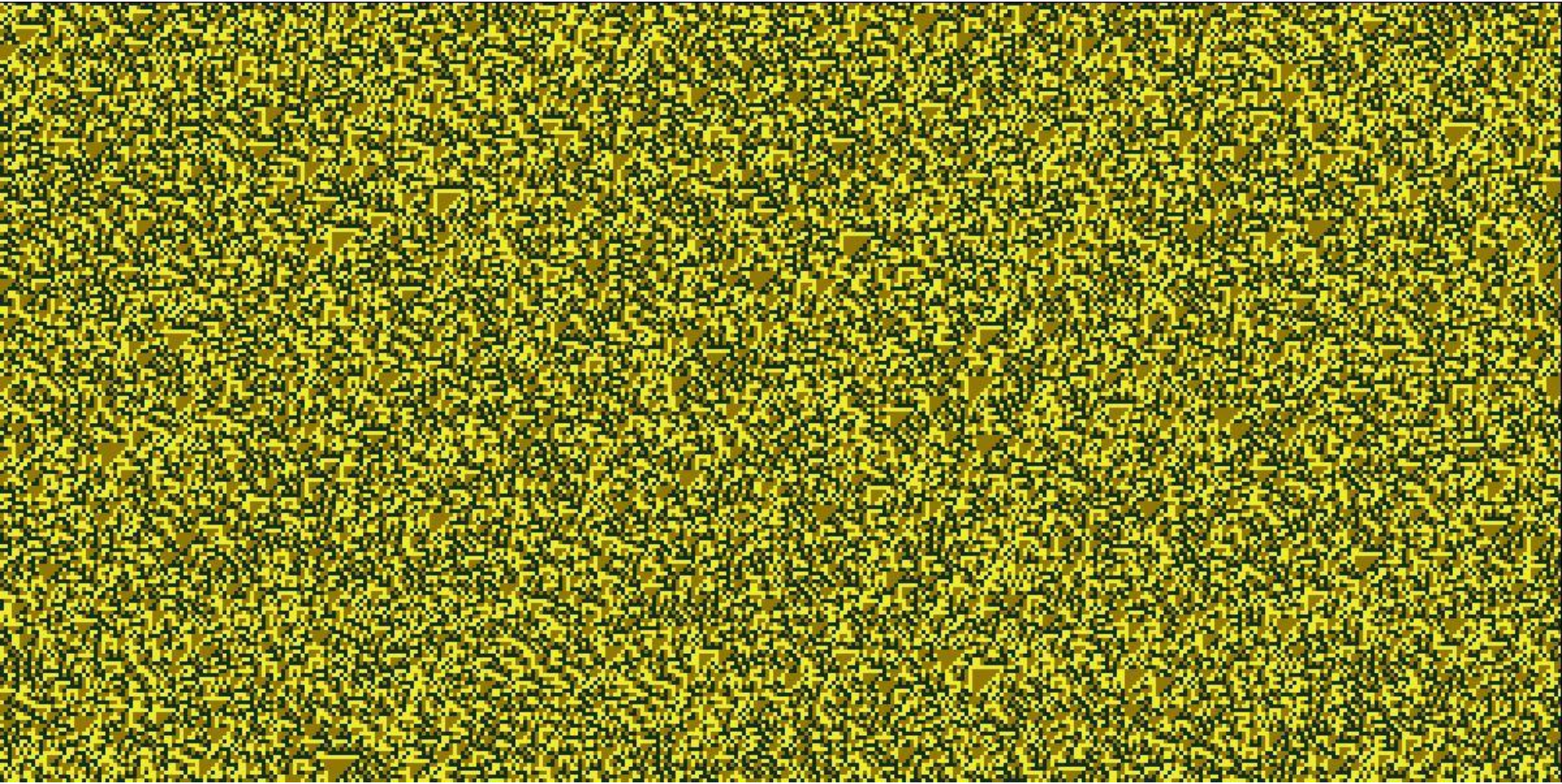
To find the rule number the user maps each input state to a result (e.g. 000 -> 1, 001 -> 2, 002 -> 1, 010 -> 0 ...) and then uses the function baseNtoDec to turn the resulting list into a decimal value, where the base N is equal to the number of states.

In practice it proves very difficult to design rules that are more interesting than random ones.

Three states, neighborhood three, rule 1,207,345,348,551



Three states, neighborhood 3, rule 1,616,986,408,089



Questions?