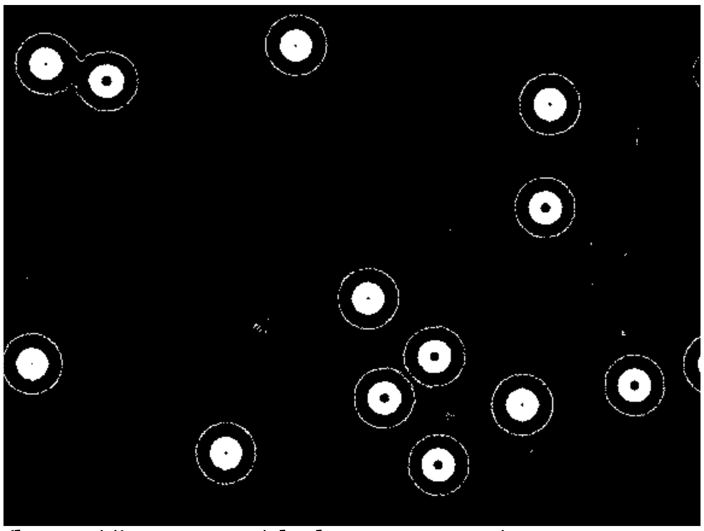
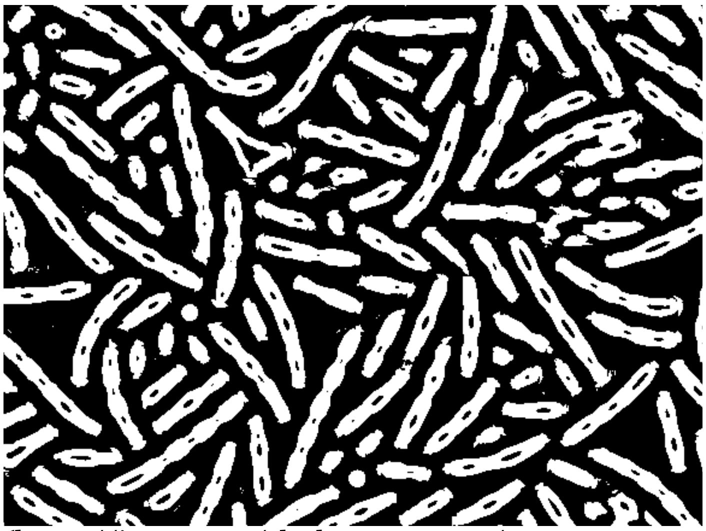
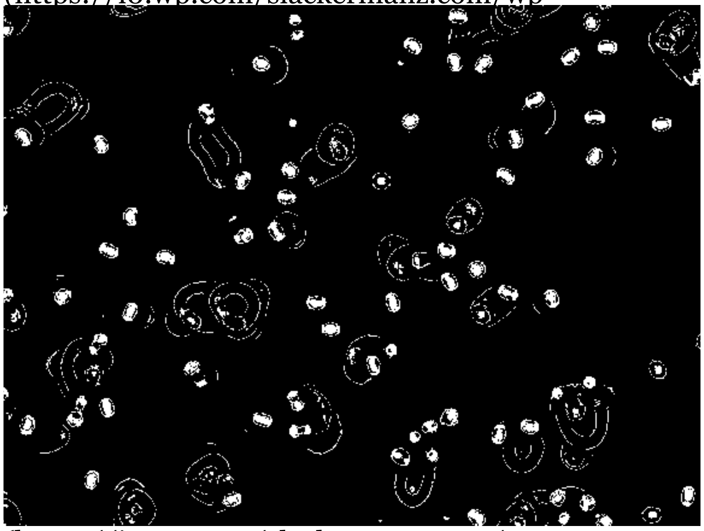
Understanding Multiple Neighborhood Cellular Automata

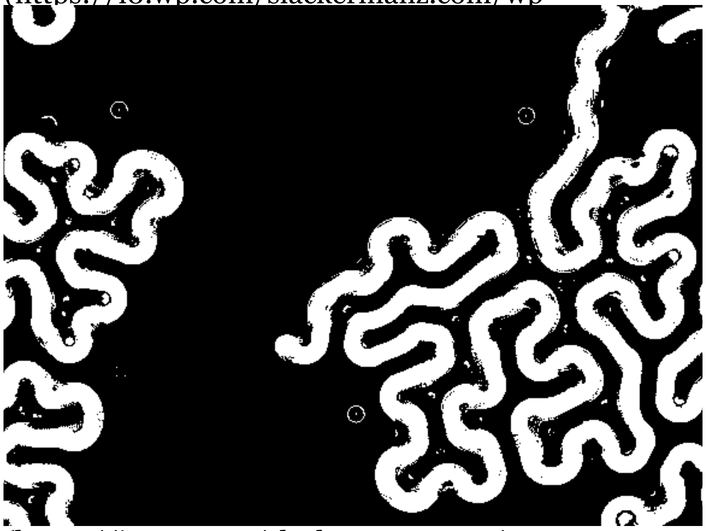
Posted on: May 23, 2021 Last updated on: June 11, 2021 Written by: Slackermanz Tagged as: Agent Based Artificial Life Biological Cellular Automata Complexity Conways Game of Life Emergence MNCA Model Multiple Neighborhood Reaction Diffusion Slackermanz Smoothlife Softology

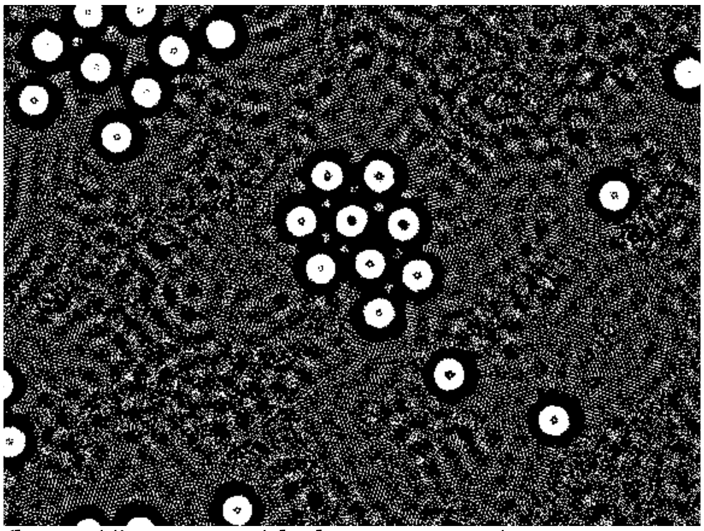
Multiple Neighborhood Cellular Automata (MNCA) are an extension of traditional cellular automata like *Conway's Game of Life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)*, developed in 2014 while I was experimenting with neighborhood configurations.

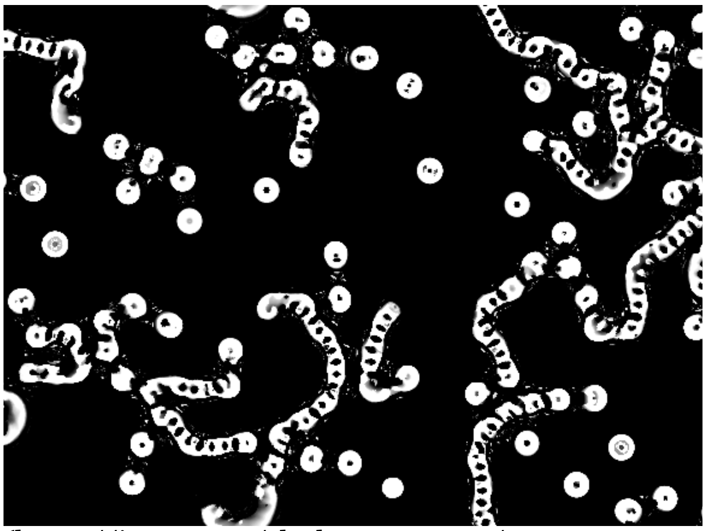

(https://i0.wp.com/slackermanz.com/wp-


(https://i0.wp.com/slackermanz.com/wp-

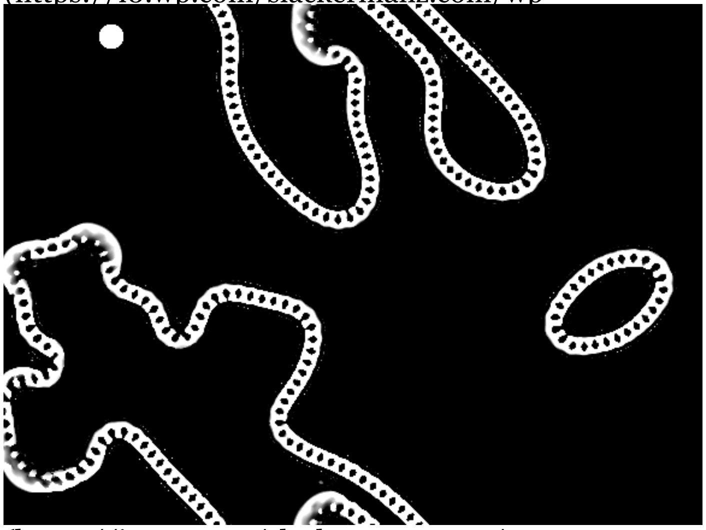

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000077_SCR_1090047_1620604848_1051.PAM.png?ssl=1)

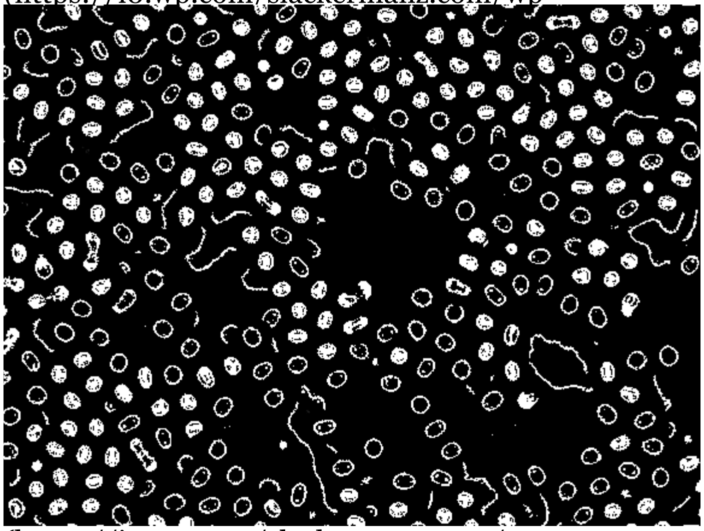

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000012_SCR_221432_1620632532_278.PAM.png?ssl=1)

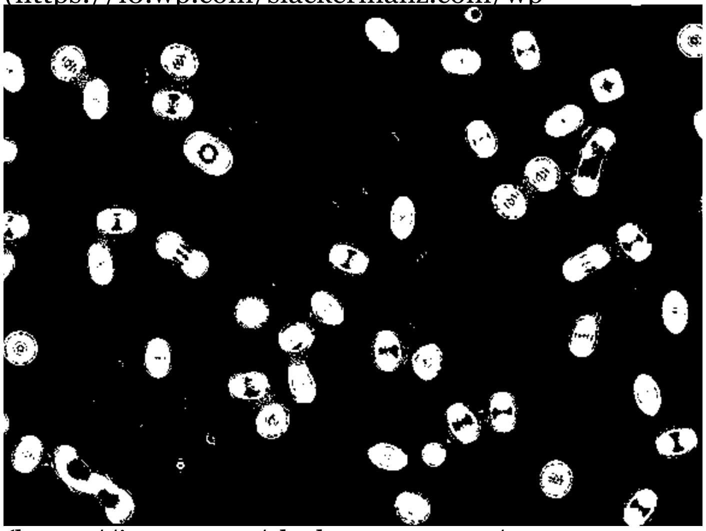

(https://i0.wp.com/slackermanz.com/wp-

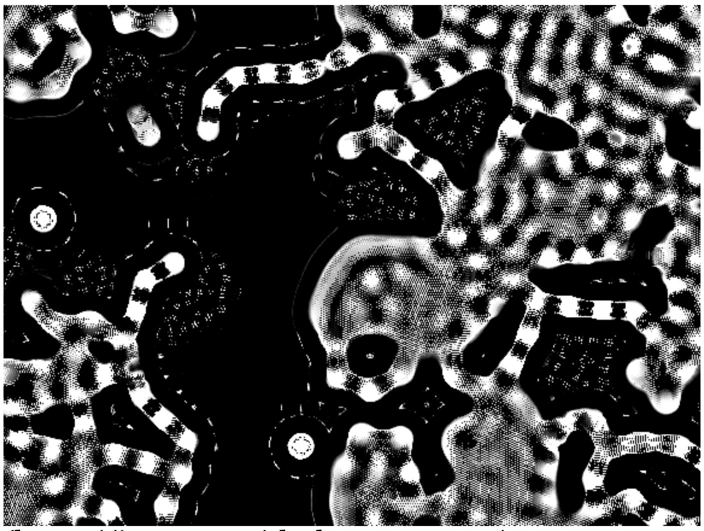

(https://i0.wp.com/slackermanz.com/wp-

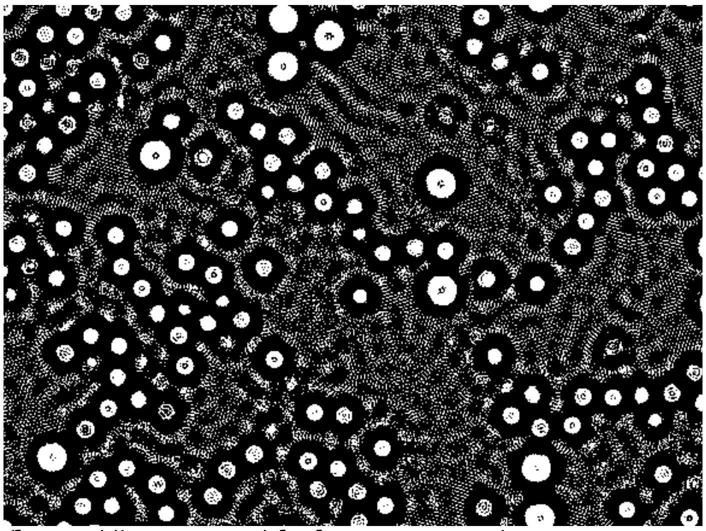
(https://i0.wp.com/slackermanz.com/wp-

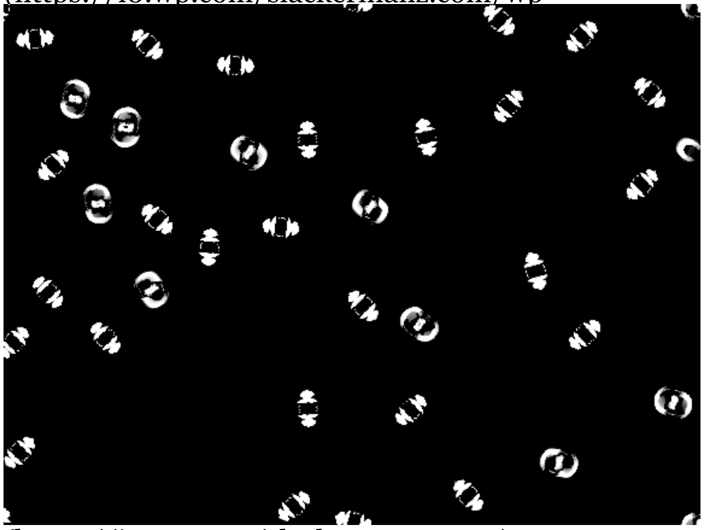

(https://i0.wp.com/slackermanz.com/wp-

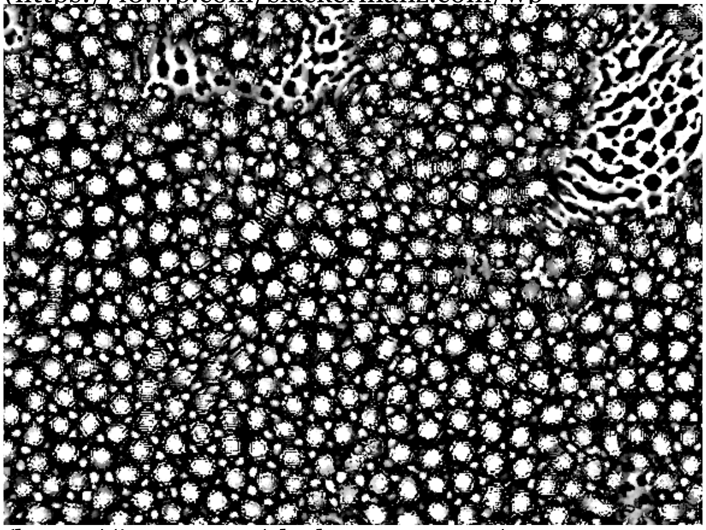

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000018_SCR_499576_1620617474_453.PAM.png?ssl=1)

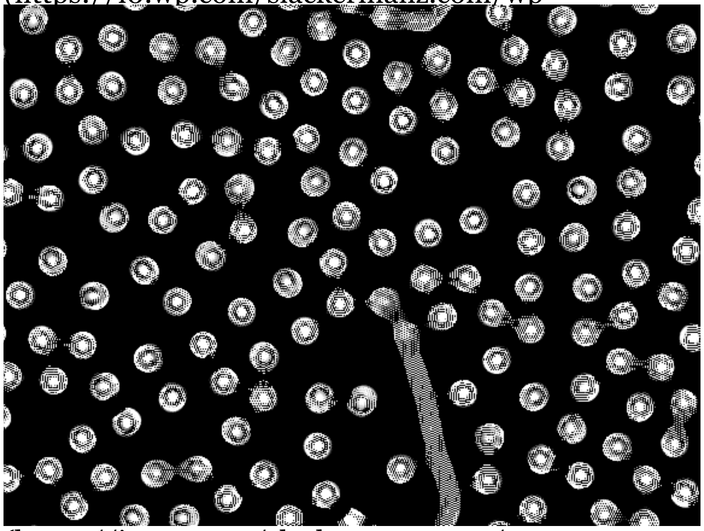

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000043_SCR_251126_1620698883_106.PAM.png?ssl=1)

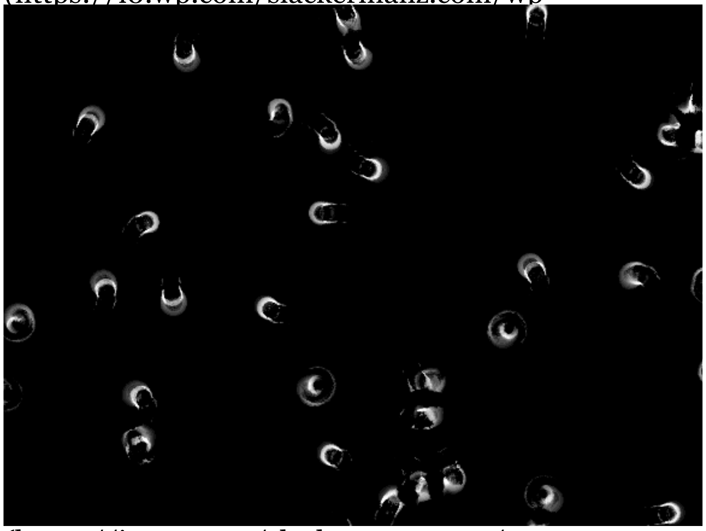

(https://i0.wp.com/slackermanz.com/wp-

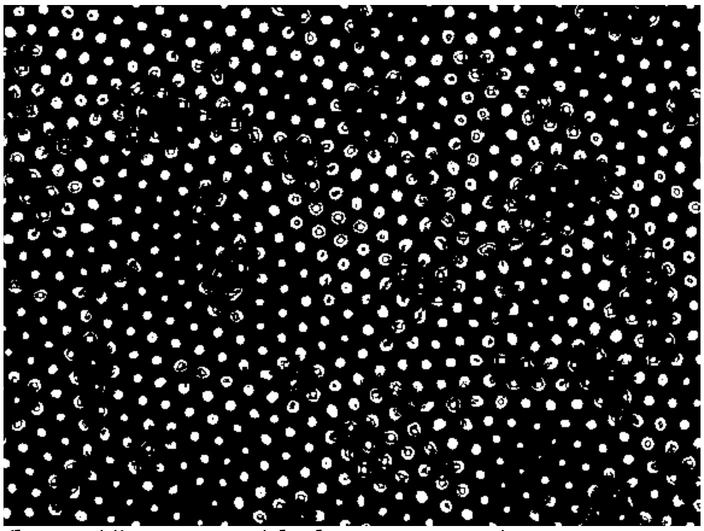

(https://i0.wp.com/slackermanz.com/wp-

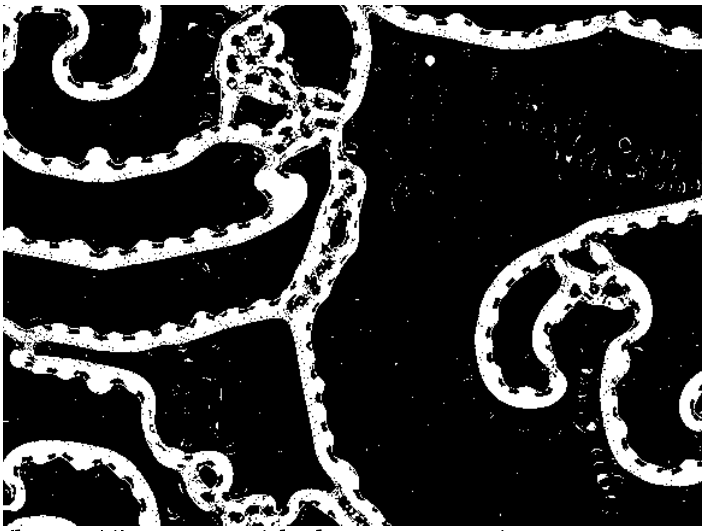

(https://i0.wp.com/slackermanz.com/wp-

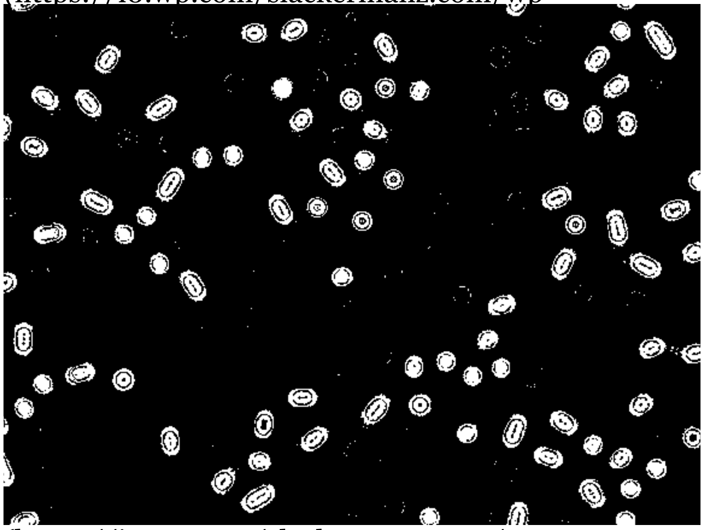

(https://i0.wp.com/slackermanz.com/wp-

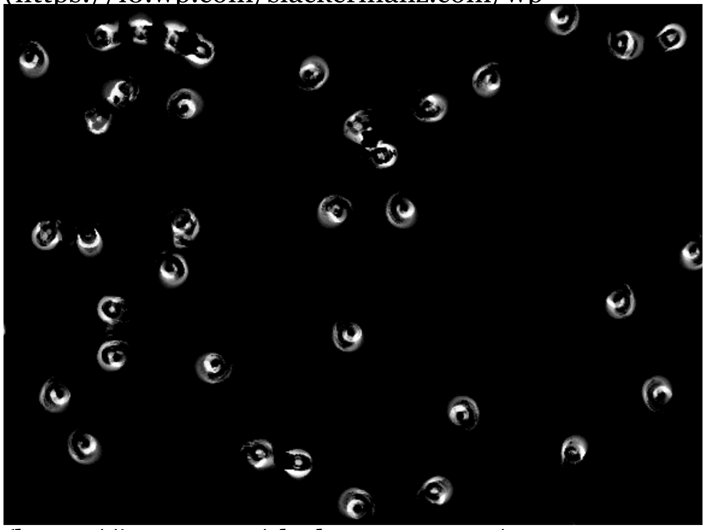

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/CMNCA/feh_738970_000011_SCR_507711_1620710969_185.PAM.png?ssl=1)

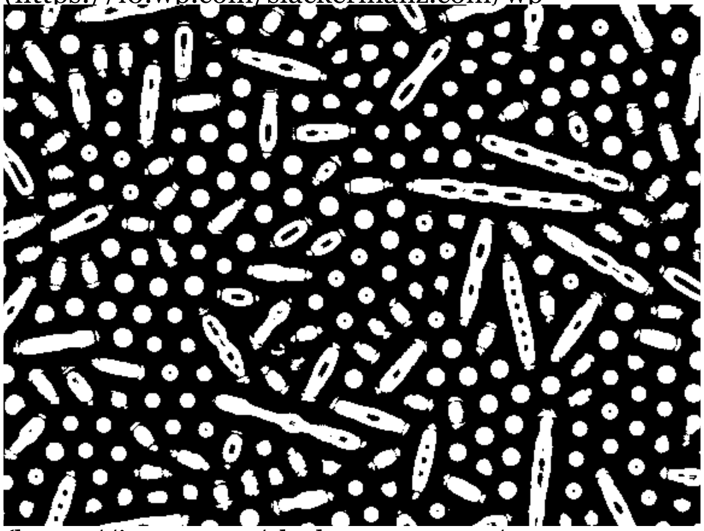

(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/CMNCA/feh_738970_000022_SCR_239130_1620708697_235.PAM.png?ssl=1)

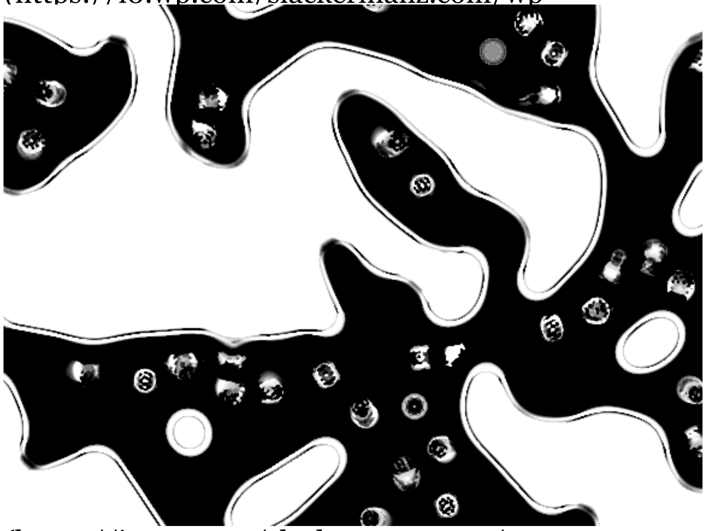

(https://i0.wp.com/slackermanz.com/wp-

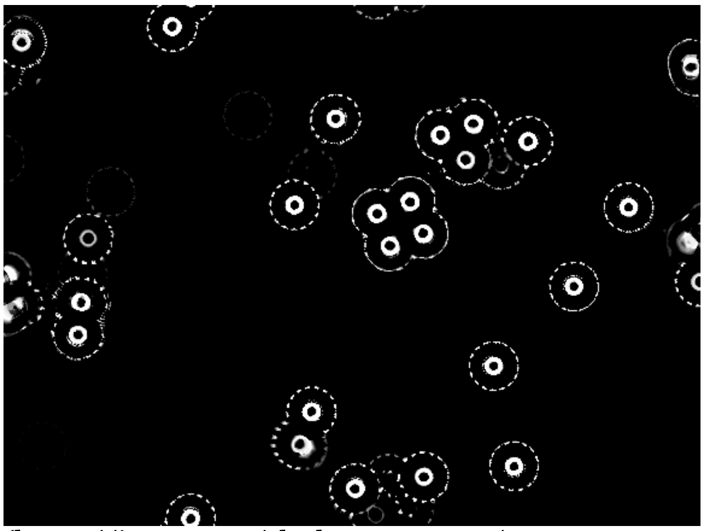

(https://i0.wp.com/slackermanz.com/wp-

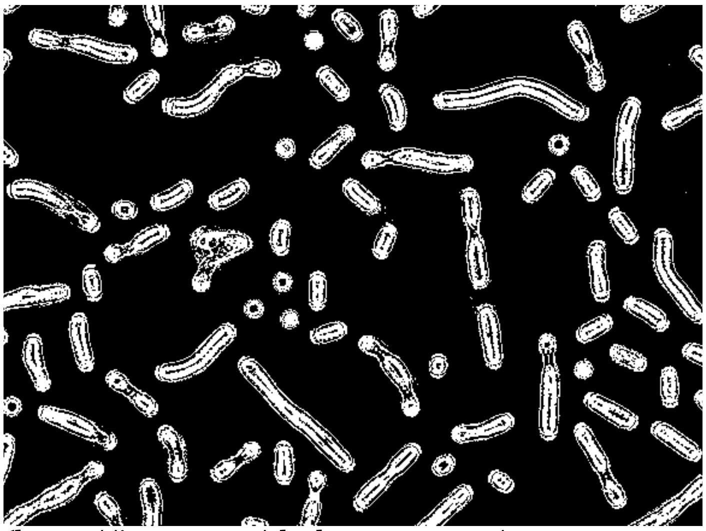

(https://i0.wp.com/slackermanz.com/wp-


(https://i0.wp.com/slackermanz.com/wp-


(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000073_SCR_1925897_1620617474_1893.PAM.png?ssl=1)


(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/CMNCA/feh_738970_000075_SCR_68103_1620708697_73.PAM.png?ssl=1)


(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/CMNCA/feh_738970_000091_SCR_868450_1620710969_511.PAM.png?ssl=1)


(https://i0.wp.com/slackermanz.com/wp-content/uploads/IMG/Plane/2MNCA/feh_738970_000079_SCR_795646_1620698883_647.PAM.png?ssl=1)

MNCA produce complex emergent patterns, often featuring robust local structures similar to solitons. The unique properties of these structures offer a vast increase in the diversity of resulting phenomena in comparison to single-neighborhood cellular automata.

Furthermore, the method behind MNCA can be directly applied to continuous-space models, producing similar results:

Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/CMNCA/SCALE/VKAutomata1524_SCALE.mp4)

- **Shadertoy Implementation: https://www.shadertoy.com/view/7ts3D7 (https://www.shadertoy.com/view/7ts3D7)**

For the sake of demonstration simplicity, the MNCA referenced further in this post will be two-state 'discrete' models.

Neighborhoods and Update Functions

Traditional cellular automata like *Conway's Game of Life* define a single group of local neighbors for each pixel, and sum the values at those locations. This 'neighborhood' is used to decide how the update functions will determine that pixel's value for the next time-step.

Unlike these traditional methods, MNCA use two or more neighborhoods of distinct composition, with each neighborhood assigned its own update functions. These updates are assessed in the order they are executed, with later updates potentially overwriting those that took place before them.

Observations of Emergent Structures

MNCA models often form structures that exhibit individualized, unit-like local identities, comparable to solitons. These structures are significantly different to those found in other cellular automata, exhibiting incredible resilience as they interact with their environment in a robust and non-destructive manner.

There are a wide variety of emergent phenomena seen in MNCA that are not common in simpler models. Some observations include:

- Formation of robust solitons with individualized local identities

- Non-destructive interactions between emergent structures
- Reaction to attractive and repulsive forces
- Formation of stable bonds and second-order emergent structures
- Collective movements / flocking
- Self-duplication in a variety of forms
- Metamorphosis of structure layouts causing different behaviors
- Compressive / elastic interactions
- Formation of crystalline lattices
- Pseudo-conservation of various attributes and quantities

Example Videos

- **[21 Videos] Two-State (Discrete) MNCA (https://slackermanz.com/2-state-discrete-mnca-example-videos/)**
- **[16 Videos] Continuous-State MNCA** (https://slackermanz.com/continuous-mnca-example-videos/)

Implementing Multiple Neighborhood Cellular Automata

The implementation details of MNCA are nearly identical to those of traditional cellular automata. However, they incur a significant additional computational cost related to the summation of large quantities of nearby values.

For this reason, MNCA are best simulated with the assistance of GPU-acceleration, although this is not strictly required.

Many cellular automata share a set of fundamental components that define a medium in which they can be simulated:

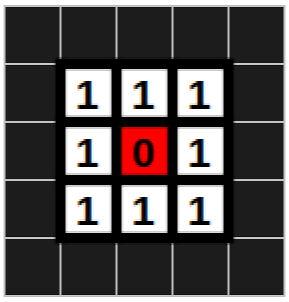
- A n-dimensional '**world**', represented by a pair of arrays or buffers
- A '**time-step**' to progress time, facilitated by a loop or recursive function
- A '**neighborhood**' for local perception, defined as a set of relative pixel coordinates
- A set of '**rules**' or update functions that determine the value of each pixel at a given time-step

By correctly setting up the medium defined by these parts, we can assign neighborhoods and update functions to generate many different types of patterns.

Conway's Game of Life

Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/CGOL_LTL_HROT/VKAutomata1531_SCALE.mp4)

A single-neighborhood cellular automata, *Conway's Game of Life* has the following neighborhood and update rules:



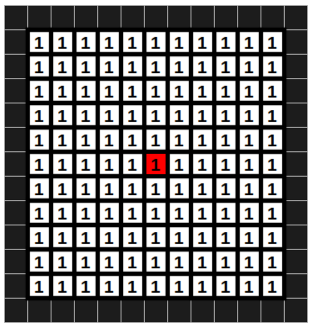
- Any cell with one or fewer live neighbors dies
- Any cell with three live neighbors becomes alive
- Any cell with four or more live neighbors dies
- All other cells survive, retaining the state they had in the previous time-step.

```
let OUTPUT_VALUE = REFERENCE_VALUE;

if( NEIGHBORHOOD_SUM >= 0.0
&&  NEIGHBORHOOD_SUM <= 1.0 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_SUM >= 3.0
&&  NEIGHBORHOOD_SUM <= 3.0 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_SUM >= 4.0
&&  NEIGHBORHOOD_SUM <= 8.0 ) { OUTPUT_VALUE = 0.0; }
```

Larger Than Life

Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/CGOL_LTL_HROT/VKAutomata1530_SCALE.mp4)

The family of patterns Larger-Than-Life use neighborhoods of greater size. The most well-known example is probably *'Bugs'*, and uses a radius-5 square neighborhood:



```
let OUTPUT_VALUE = REFERENCE_VALUE;

if( NEIGHBORHOOD_SUM >=   0.0
&&  NEIGHBORHOOD_SUM <=  33.0 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_SUM >=  34.0
&&  NEIGHBORHOOD_SUM <=  45.0 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_SUM >=  58.0
&&  NEIGHBORHOOD_SUM <= 121.0 ) { OUTPUT_VALUE = 0.0; }
```

Multiple Neighborhood Cellular Automata

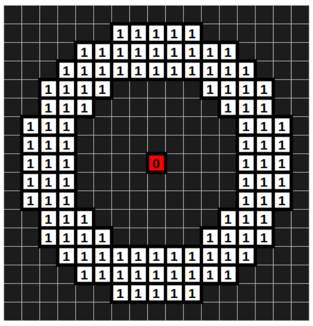
Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/2MNCA/SCALE/VKAutomata1533_SCALE.mp4)

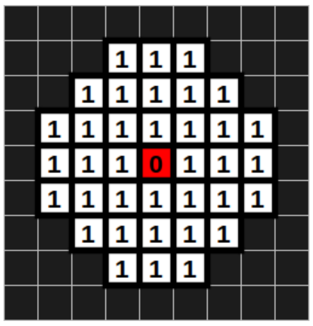
Note that for MNCA I've chosen to use the **Average (..._AVG)** instead of **Sum (..._SUM)** when referencing the Neighborhood. The NEIGHBORHOOD_AVG results are calculated by dividing each NEIGHBORHOOD_SUM value by the total number of neighbors in the neighborhood.

Another significant change is that the **NEIGHBORHOOD_AVG[n]** is an array of the four individual results, rather than a single value. That said, four individual variables is also a valid way to store these values.

A basic MNCA example:

Here is a simplistic discrete MNCA I manually constructed using two neighborhoods and six update functions:





```
let OUTPUT_VALUE = REFERENCE_VALUE;

if( NEIGHBORHOOD_AVG[0] >= 0.210
&&  NEIGHBORHOOD_AVG[0] <= 0.220 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.350
&&  NEIGHBORHOOD_AVG[0] <= 0.500 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.750
&&  NEIGHBORHOOD_AVG[0] <= 0.850 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[1] >= 0.100
&&  NEIGHBORHOOD_AVG[1] <= 0.280 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[1] >= 0.430
&&  NEIGHBORHOOD_AVG[1] <= 0.550 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.120
&&  NEIGHBORHOOD_AVG[0] <= 0.150 ) { OUTPUT_VALUE = 0.0; }
```

- **Shadertoy Implementation: https://www.shadertoy.com/view/7ll3R7 (https://www.shadertoy.com/view/7ll3R7)**

Using the same two neighborhoods with a different set of update functions results in a largely different kind of pattern:

Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/2MNCA/SCALE/VKAutomata1534_SCALE.mp4)

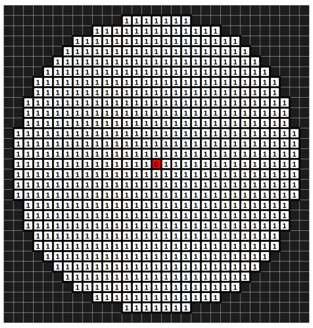
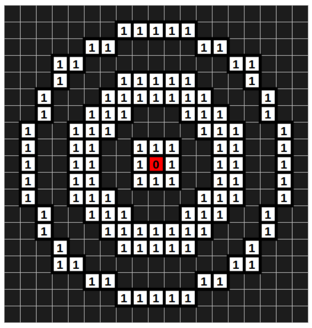
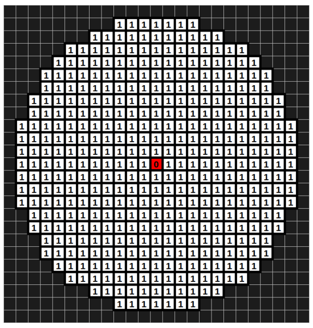
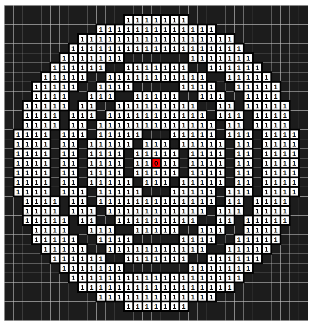
```
let OUTPUT_VALUE = REFERENCE_VALUE;

if( NEIGHBORHOOD_AVG[0] >= 0.185
&&  NEIGHBORHOOD_AVG[0] <= 0.200 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.343
&&  NEIGHBORHOOD_AVG[0] <= 0.580 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.750
&&  NEIGHBORHOOD_AVG[0] <= 0.850 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[1] >= 0.150
&&  NEIGHBORHOOD_AVG[1] <= 0.280 ) { OUTPUT_VALUE = 0.0; }
if( NEIGHBORHOOD_AVG[1] >= 0.445
&&  NEIGHBORHOOD_AVG[1] <= 0.680 ) { OUTPUT_VALUE = 1.0; }
if( NEIGHBORHOOD_AVG[0] >= 0.150
&&  NEIGHBORHOOD_AVG[0] <= 0.180 ) { OUTPUT_VALUE = 0.0; }
```

- **Shadertoy Implementation: https://www.shadertoy.com/view/sll3R7 (https://www.shadertoy.com/view/sll3R7)**

A more complex MNCA example:

Download MP4 Video (https://slackermanz.com/wp-content/uploads/VID/2MNCA/SCALE/VKAutomata1529_SCALE.mp4)

This example of a Multiple Neighborhood Cellular Automaton was created with a series of random 'partial mutations' used along with 'checkpoints/saved configurations' to create the neighborhoods and update ranges. In effect, these represent a simple and interactive evolutionary algorithm. It can be described in the same format:

```
let OUTPUT_VALUE = REFERENCE_VALUE;

if( NEIGHBORHOOD_AVG[0] >= 0.262364076538086
&&  NEIGHBORHOOD_AVG[0] <= 0.902710297241211 ) { OUTPUT_VALUE = 0.0;
}
if( NEIGHBORHOOD_AVG[0] >= 0.876029204711914
&&  NEIGHBORHOOD_AVG[0] <= 0.764857985839844 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[0] >= 0.533621850585938
&&  NEIGHBORHOOD_AVG[0] <= 0.911603994750977 ) { OUTPUT_VALUE = 0.0;
}
if( NEIGHBORHOOD_AVG[0] >= 0.787092229614258
&&  NEIGHBORHOOD_AVG[0] <= 0.449131724243164 ) { OUTPUT_VALUE = 0.0;
}

if( NEIGHBORHOOD_AVG[1] >= 0.342407354125977
&&  NEIGHBORHOOD_AVG[1] <= 0.377982144165039 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[1] >= 0.453578572998047
&&  NEIGHBORHOOD_AVG[1] <= 0.057809033813477 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[1] >= 0.484706514282227
&&  NEIGHBORHOOD_AVG[1] <= 0.671474161987305 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[1] >= 0.057809033813477
&&  NEIGHBORHOOD_AVG[1] <= 0.11117121887207  ) { OUTPUT_VALUE = 0.0;
}

if( NEIGHBORHOOD_AVG[2] >= 0.342407354125977
&&  NEIGHBORHOOD_AVG[2] <= 0.382428992919922 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[2] >= 0.755964288330078
&&  NEIGHBORHOOD_AVG[2] <= 0.53806869934082  ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[2] >= 0.195661345214844
&&  NEIGHBORHOOD_AVG[2] <= 0.217895588989258 ) { OUTPUT_VALUE = 0.0;
}
if( NEIGHBORHOOD_AVG[2] >= 0.671474161987305
&&  NEIGHBORHOOD_AVG[2] <= 0.489153363037109 ) { OUTPUT_VALUE = 1.0;
}

if( NEIGHBORHOOD_AVG[3] >= 0.889369750976563
&&  NEIGHBORHOOD_AVG[3] <= 0.978306726074219 ) { OUTPUT_VALUE = 1.0;
}
if( NEIGHBORHOOD_AVG[3] >= 0.035574790039063
&&  NEIGHBORHOOD_AVG[3] <= 0.133405462646484 ) { OUTPUT_VALUE = 0.0;
}
if( NEIGHBORHOOD_AVG[3] >= 0.88492290222168
&&  NEIGHBORHOOD_AVG[3] <= 0.760411137084961 ) { OUTPUT_VALUE = 0.0;
}
if( NEIGHBORHOOD_AVG[3] >= 0.635899371948242
&&  NEIGHBORHOOD_AVG[3] <= 0.257917227783203 ) { OUTPUT_VALUE = 1.0;
}
```

- **Shadertoy Implementation: https://www.shadertoy.com/view/Nts3RM (https://www.shadertoy.com/view/Nts3RM)**

While these structures look very similar in notation, MNCA clearly produce emergent phenomena that are significantly different to the families of rules represented by the single-neighborhood examples.

Further Reading

Jason of ***Softology*** (@SoftologyComAu (https://twitter.com/SoftologyComAu), https://softologyblog.wordpress.com (https://softologyblog.wordpress.com)) has also done a series of blog posts about my work with MNCA and similar techniques, and has included thousands of my patterns in his legendary application, ***Visions of Chaos*** *(https://softology.com.au/voc.htm)*

- Multiple Neighborhoods Cellular Automata (https://softologyblog.wordpress.com/2018/03/09/multiple-neighborhoods-cellular-automata/)
- More explorations with Multiple Neighborhoods Cellular Automata (https://softologyblog.wordpress.com/2018/03/31/more-explorations-with-multiple-neighborhood-cellular-automata/)
- Even more explorations with Multiple Neighborhoods Cellular Automata (https://softologyblog.wordpress.com/2021/02/14/even-more-explorations-with-multiple-neighborhoods-cellular-automata/)