Cellular Automata

Formal Languages and Abstract Machines

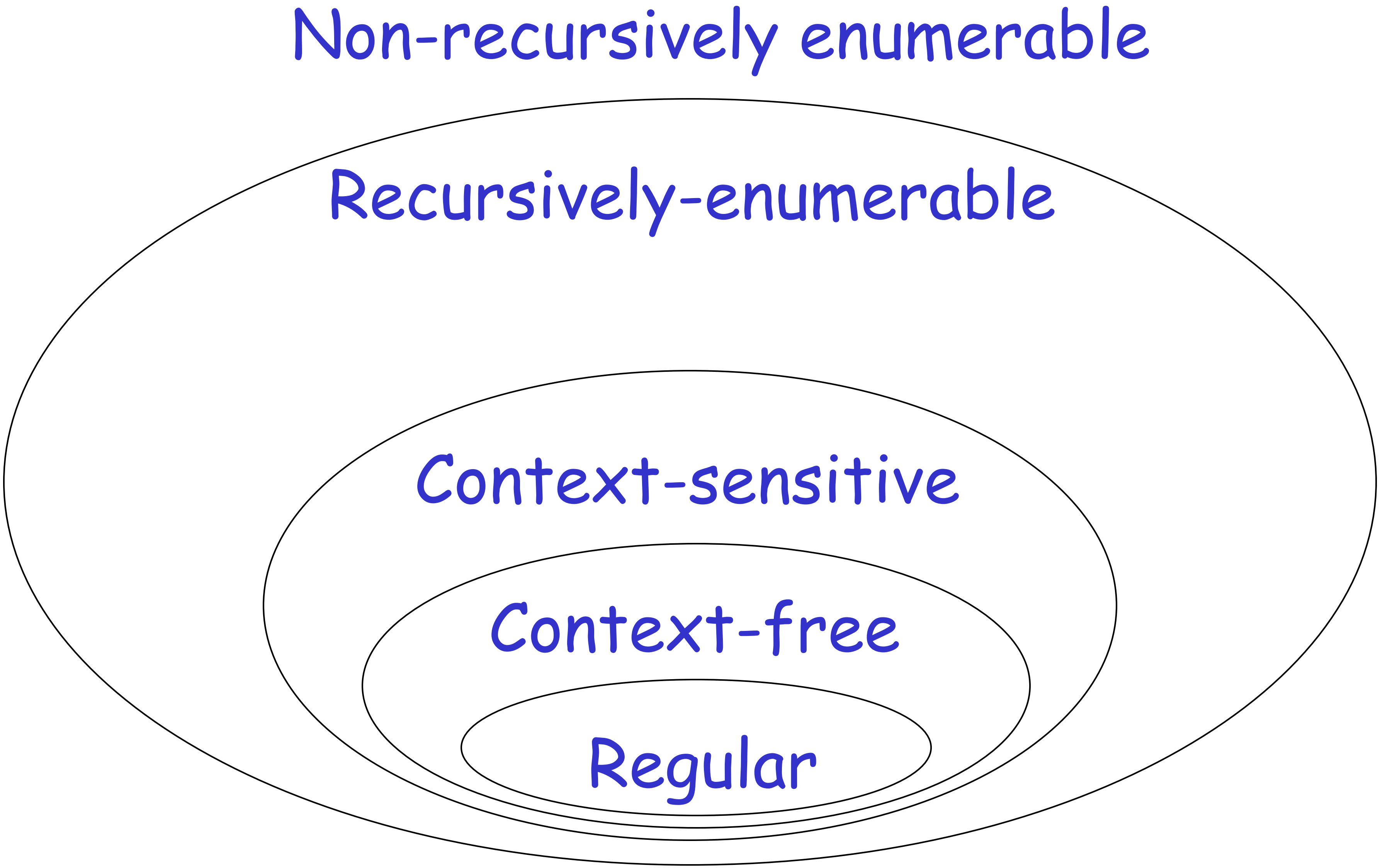
Week 13

Baris E. Suzek, PhD

Outline

- Review of last week
- Cellular Automata

The Chomsky Hierarchy



Efficiency of a Computation

Time Complexity: The number of steps during a computation

Space Complexity: Space used during a computation

$$DTIME(n)$$

$$\{a^n b^n : n \geq 0\}$$

$$\{ww\}$$

In a similar way we define the class

$$DTIME(T(n))$$

for any time function: $T(n)$

Examples: $DTIME(n^2), DTIME(n^3), \ldots$

The class P

$$P = \cup DTIME(n^k) \qquad \text{for all } k$$

- Polynomial time
- All tractable problems(can be solved effectively)

P

CYK-algorithm

$\{a^n b^n\}$

$\cdots$ $\{ww\}$

Exponential time algorithms: $DTIME(2^n)$

Represent intractable algorithms:

Some problem instances may take centuries to solve

Non-Determinism

Language class: $NTIME(n)$

$NTIME(n)$

L_1 L_2 L_3

A Non-Deterministic Machine accepts each string of length n in time $O(n)$ (testing solution is $O(n)$)

Example: $L = \{ww\}$

Non-Deterministic Algorithm
to accept a string ww :

- Use a two-tape Turing machine

- Guess the middle of the string
  and copy w on the second tape

- Compare the two tapes

$$L = \{ww\}$$

Time needed:

- Use a two-tape Turing machine

- Guess the middle of the string and copy w on the second tape — $O(|w|)$

- Compare the two tapes — $O(|w|)$

Total time: $O(|w|)$

$$NTIME(n)$$

$$L = \{ww\}$$

In a similar way we define the class

$$NTIME(T(n))$$

for any time function: $T(n)$

Examples: $NTIME(n^2), NTIME(n^3), \ldots$

So if problems are

P

- They can be solved in polynomial time.

NP

- You can quickly (in polynomial time) test whether a solution is correct
  - without worrying about how hard it might be to find the solution).
- They are still relatively easy: if only we could guess the right solution, we could then quickly test it.
  - e.g. RSA(key,text) and the "known plaintext attack"

Outline

- Review of last week
- Cellular Automata

What is a Cellular Automata?

- n-dimensional homogeneous and cellular space; consisting of cells of equal size
  - Cells in one of a discrete number of states;
  - Cells change state as the result of a transition rule;
- Transition rule is defined in terms of the states of cells that are part of a neighbourhood;
- Time progresses in discrete steps.
  - All cells change state simultaneously.

Concept introduced by Von Neumann, Ulam and Burk in late 1940-ies and 1950-ies; (Self-reproducible mechanical automata)

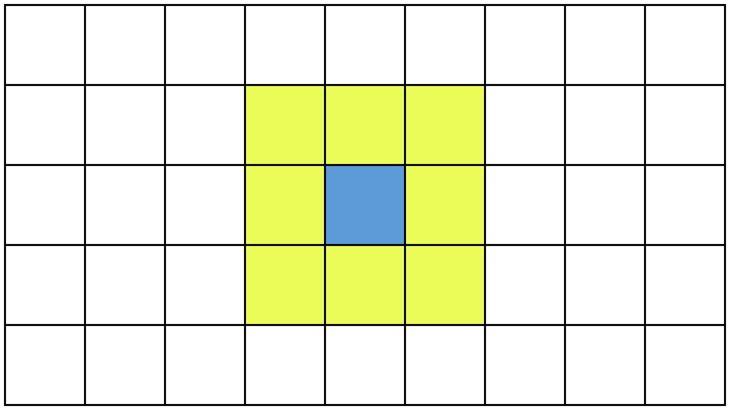
How does Cellular Automata Work?

- When the time comes for the cells to change state, each cell looks around and gathers information on its neighbors' states.
  - Exactly which cells are considered "neighbors" is also something that depends on the particular CA.
- Based on
  - its own state
  - its neighbors' states
  - the rules of the CA

  the cell decides what its new state should be.

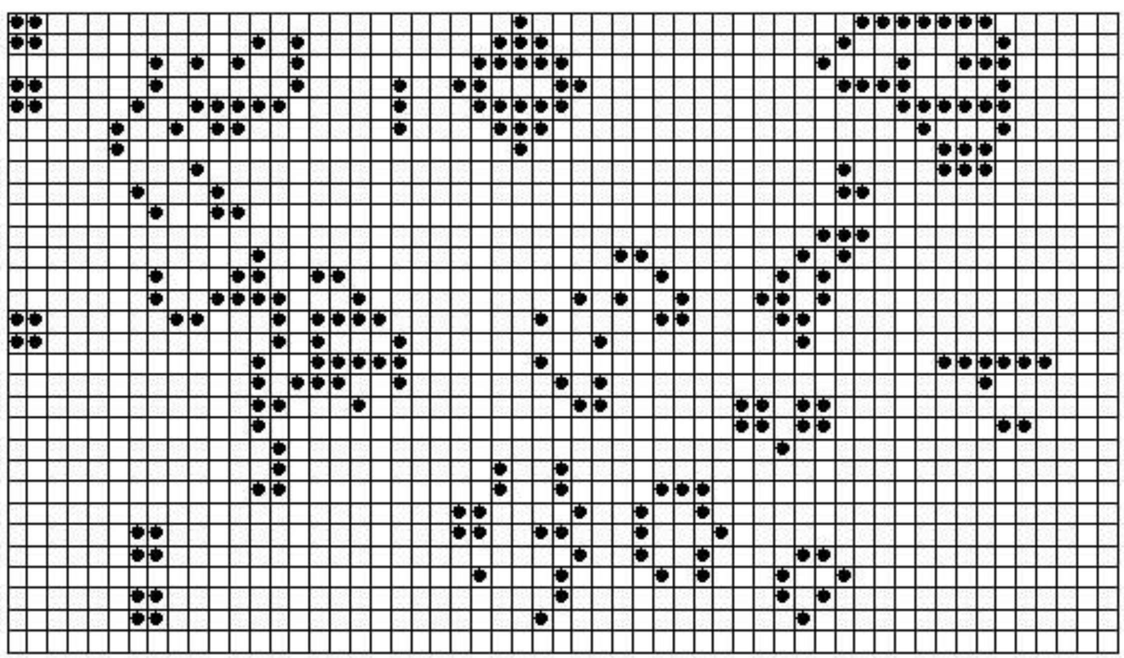
2-Dimensional CA

- Consists of an infinite (or finite) grid of cells, each in one of a finite number of states. Time is discrete and the state of a cell at time t is a function of the states of its neighbors at time t-1.



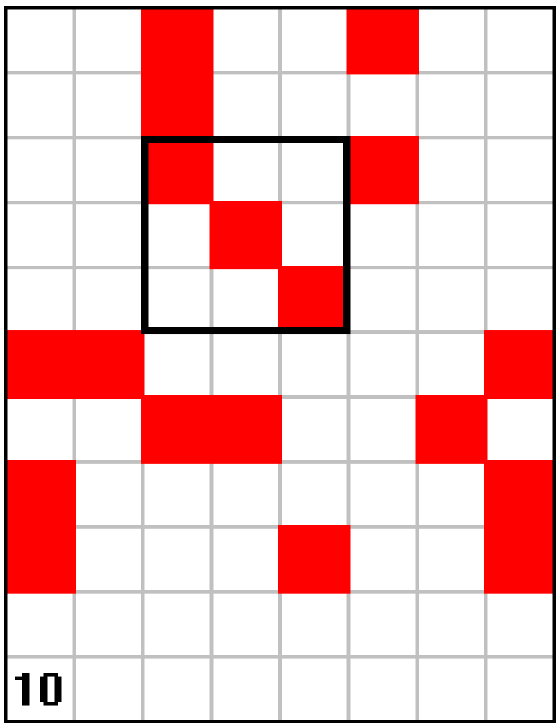
CA Example: Conway's Game of Life

The universe of the Game of Life is an infinite two-dimensional grid of cells, each of which is either alive or dead.

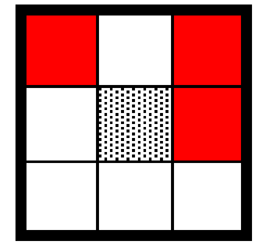


CA Example: Conway's Game of Life – Grid, Neighbourhood

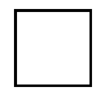
2-D ***cellular space***
consisting of identical cells

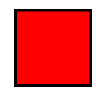


neighbourhood
(Moore)



cells are in 1 of 2 ***states:***

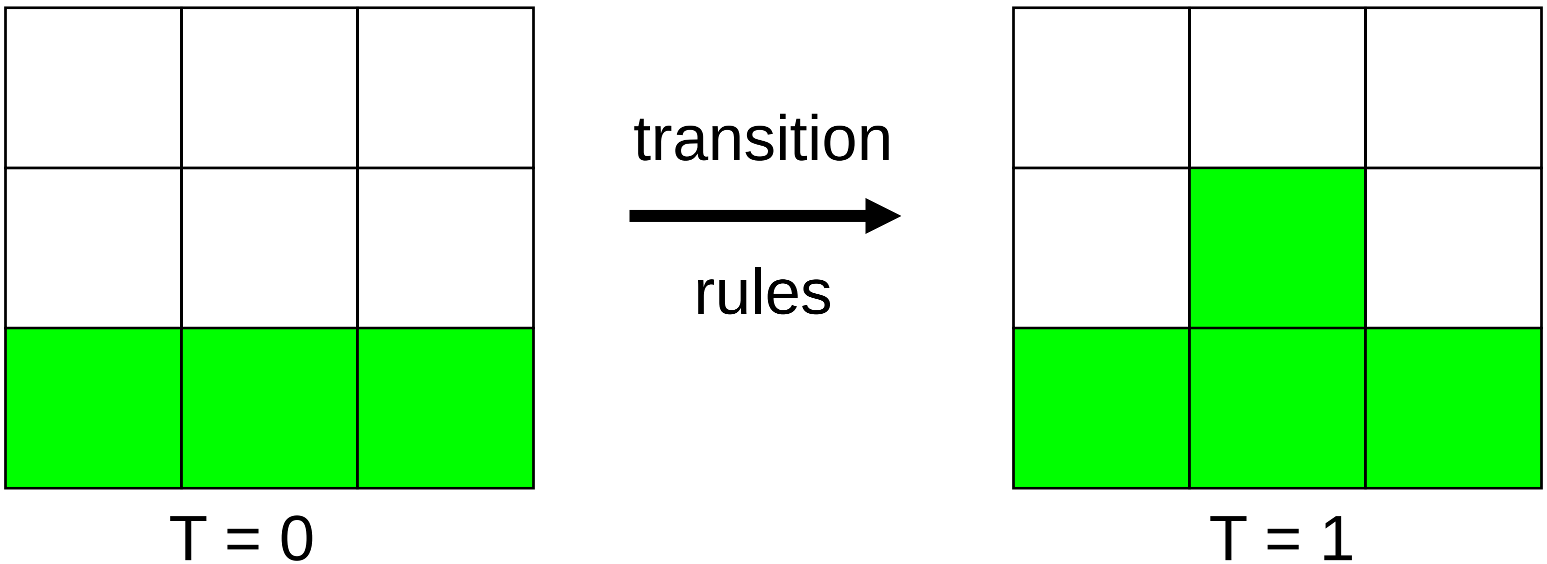
dead, 

or alive 

state changes due to ***transition rules:***

- live cell stays alive if 2 or 3 of its neighbours are alive, otherwise it dies.
- dead cell will come to life if it has 3 live neighbours.

CA Example: Conway's Game of Life - Transition



CA Example: Conway’s Game of Life - Rules

Cell

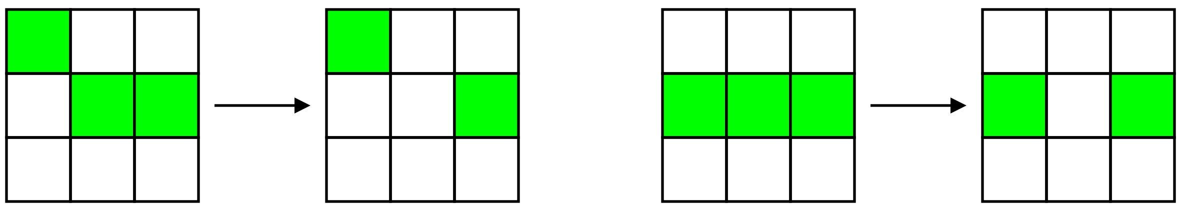
- dies if number of alive neighbour cells =< 2
  (*loneliness*)

- dies if number of alive neighbour cells >= 5
  (*overcrowding*)

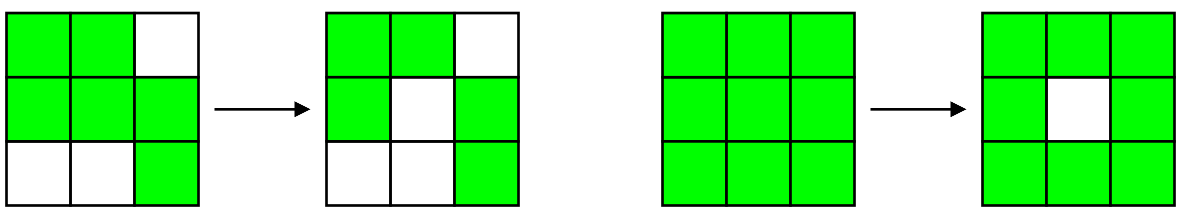
- lives if number of alive neighbour cells = 3
  (*procreation*)

CA Example: Conway's Game of Life – Rules Applied

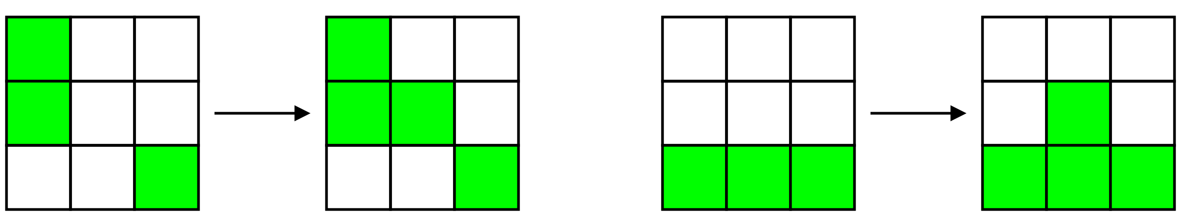
- **loneliness** (dies if #alive =< 2)



- **overcrowding** (dies if #alive >= 5)



- **procreation** (lives if #alive = 3)



CA Example: Conway’s Game of Life – Initial Pattern, Generations

- The **initial pattern** constitutes the first generation of the system.
- The **second generation** is created by applying the above rules simultaneously to every cell in the first generation -- births and deaths happen simultaneously.
- The rules continue to be applied repeatedly to create **further generations**.

Where can Cellular Automata be used?

- Can be used to model complex systems if you can
  - divide problem space into cells
  - each cell can be in one of several finite states
  - cells are affected by neighbors according to rules
  - all cells are affected simultaneously in a generation
  - rules are reapplied over many generations

What are Applications of Cellular Automata?

- Simulation of Biological Processes
- Simulation of Cancer Cells Growth
- Predator – Prey Models
- Art
- Simulation of Forest Fires
- Simulations of Social Movement
- …many more..

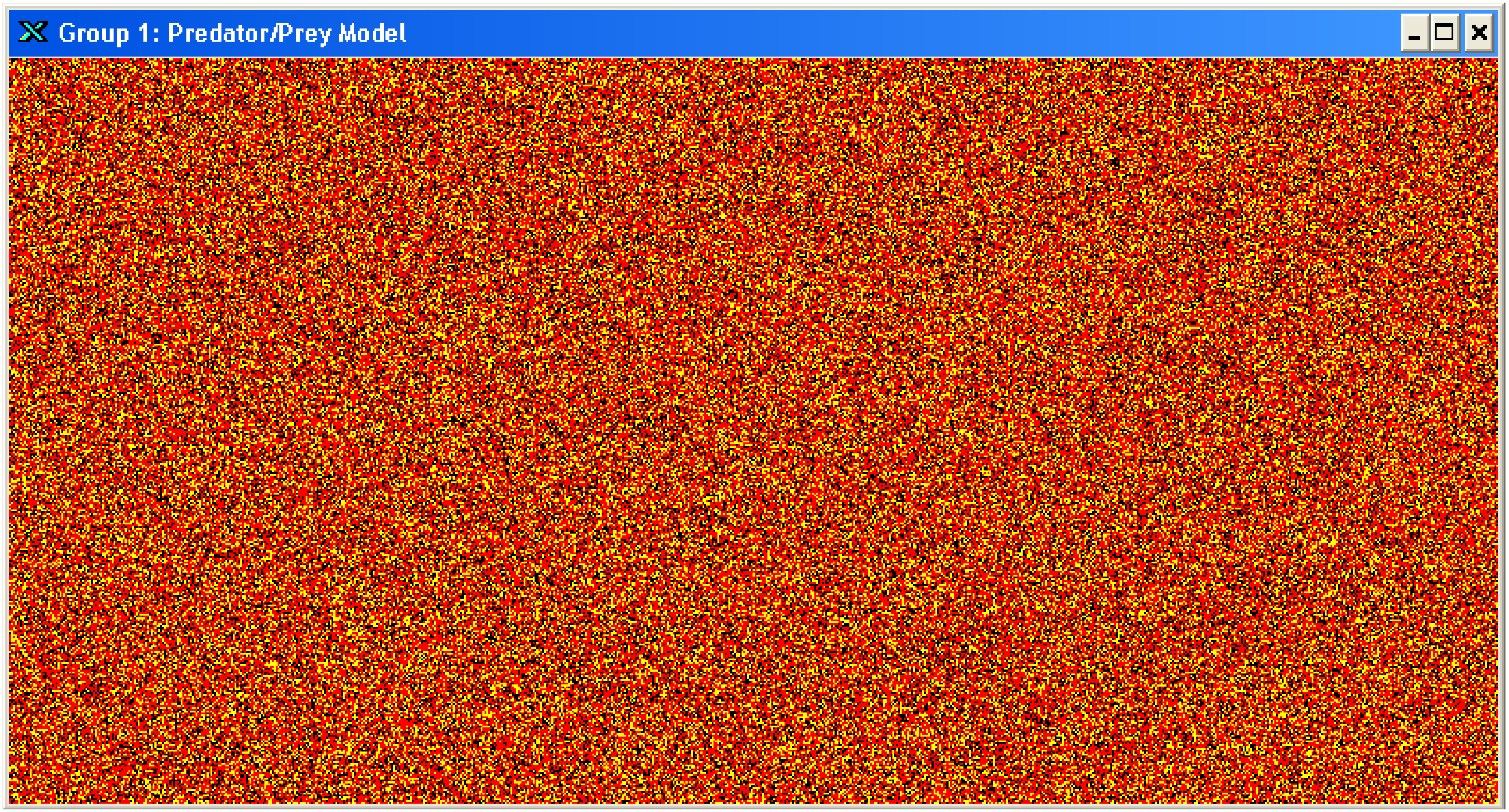
Another CA Example: Sharks and Fish – Initial Pattern

- Model predator/prey relationship by CA
- Define set of rules
- Begins with a randomly distributed population of fish, sharks, and empty cells in a 1000x2000 cell grid (2 million cells)
- Initially,
  - 50% of the cells are occupied by fish
  - 25% are occupied by sharks
  - 25% are empty

Based on the work of Bill Madden, Nancy Ricca and Jonathan Rizzo (Montclair State University)

Here's the number 2 million

- Fish: red; sharks: yellow; empty: black



Another CA Example: Sharks and Fish – Rules Applied

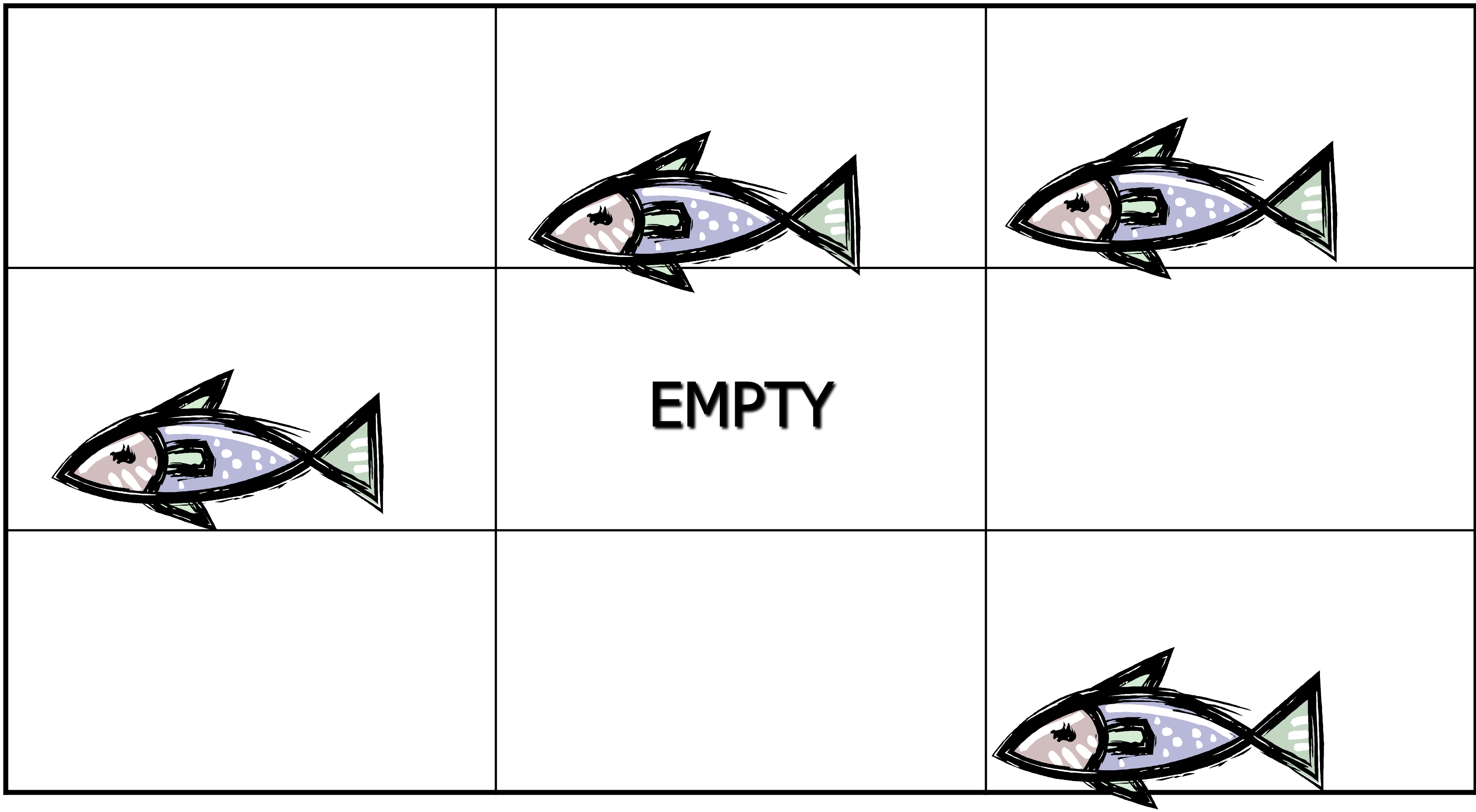
Breeding rule:

If the current cell is empty and if all below apply:

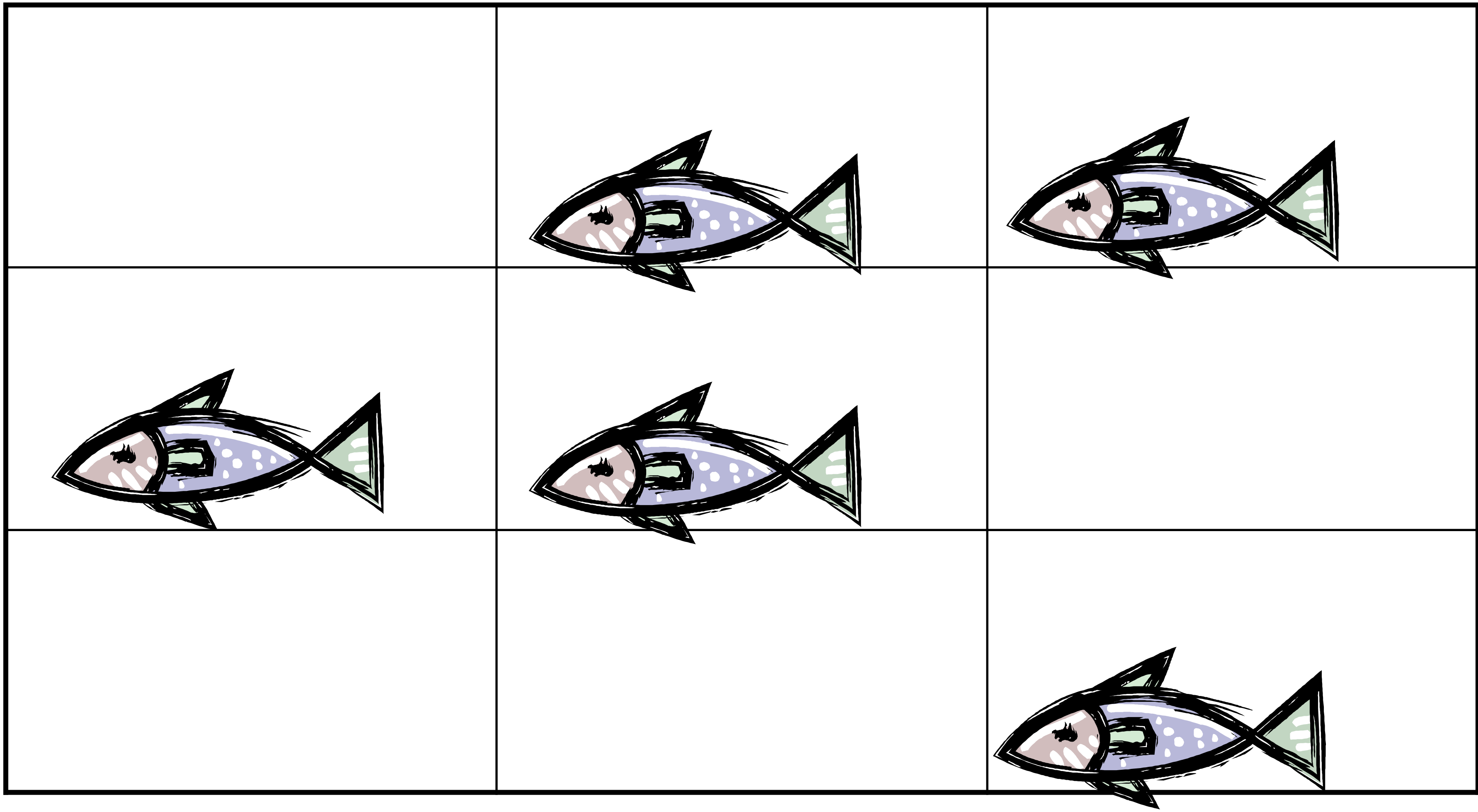
- there are >= 4 neighbors of one species,
- >= 3 of them are of breeding age, [Fish breeding age >= 2,Shark breeding age >=3]
- there are <4 of the other species:

then create a species of that type [Fish or Shark]

Breeding Rule: Before



Breeding Rule: After



Another CA Example: Sharks and Fish – Rules Applied

Fish rules:

If the current cell contains a fish:

- Fish live for 10 generations
- If >=5 neighbors are sharks, fish dies (shark food)
- If all 8 neighbors are fish, fish dies (overpopulation)
- If a fish does not die, increment age

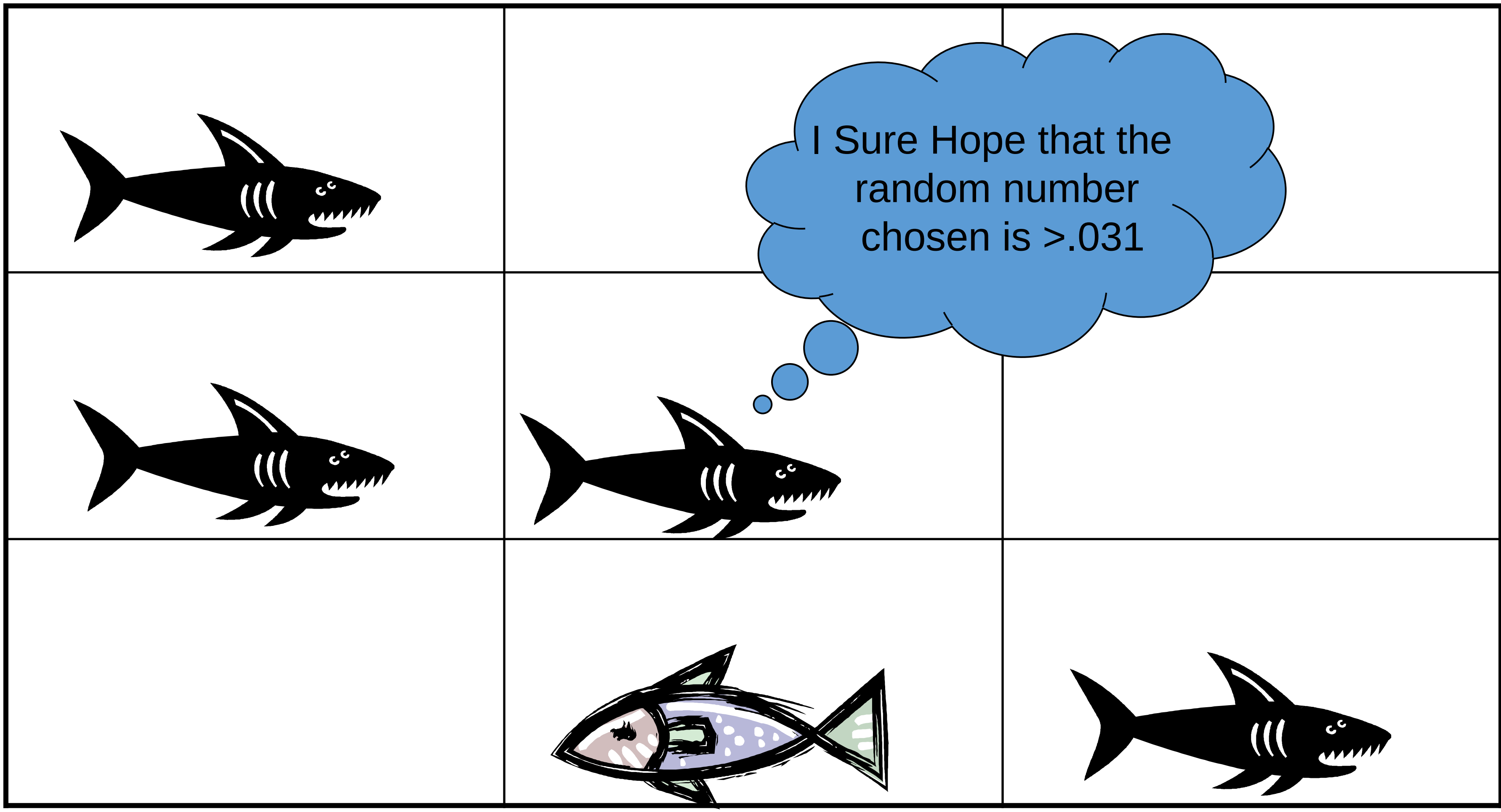
Another CA Example: Sharks and Fish – Rules Applied

Shark rules:

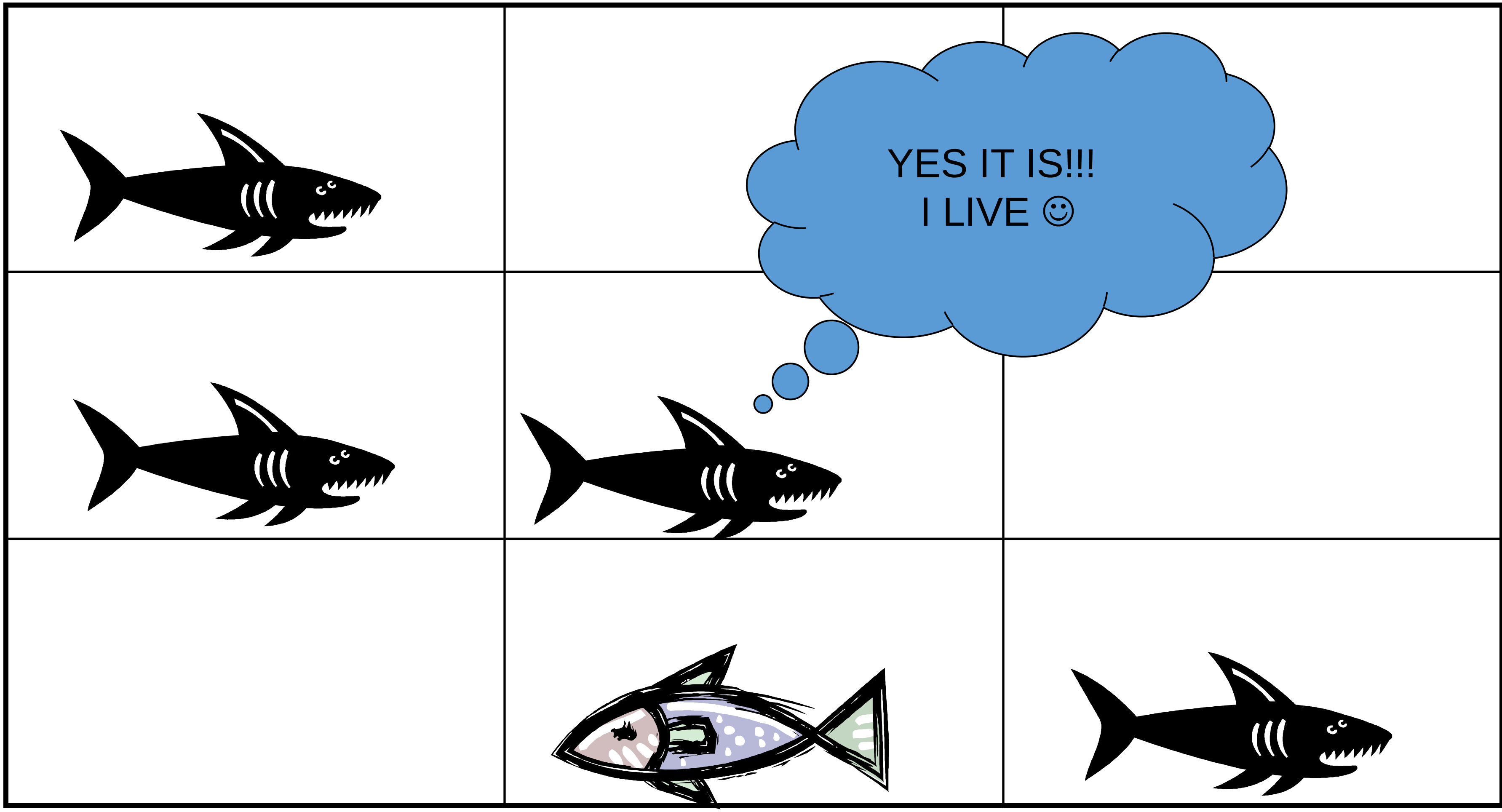
If the current cell contains a shark:

- Sharks live for 20 generations
- If >=6 neighbors are sharks and fish neighbors =0, the shark dies (starvation)
- A shark has a 1/32 (.031) chance of dying due to random causes
- If a shark does not die, increment age

Shark Random Death: Before



Shark Random Death: After



Another CA Example: Sharks and Fish – Rules Programming Logic

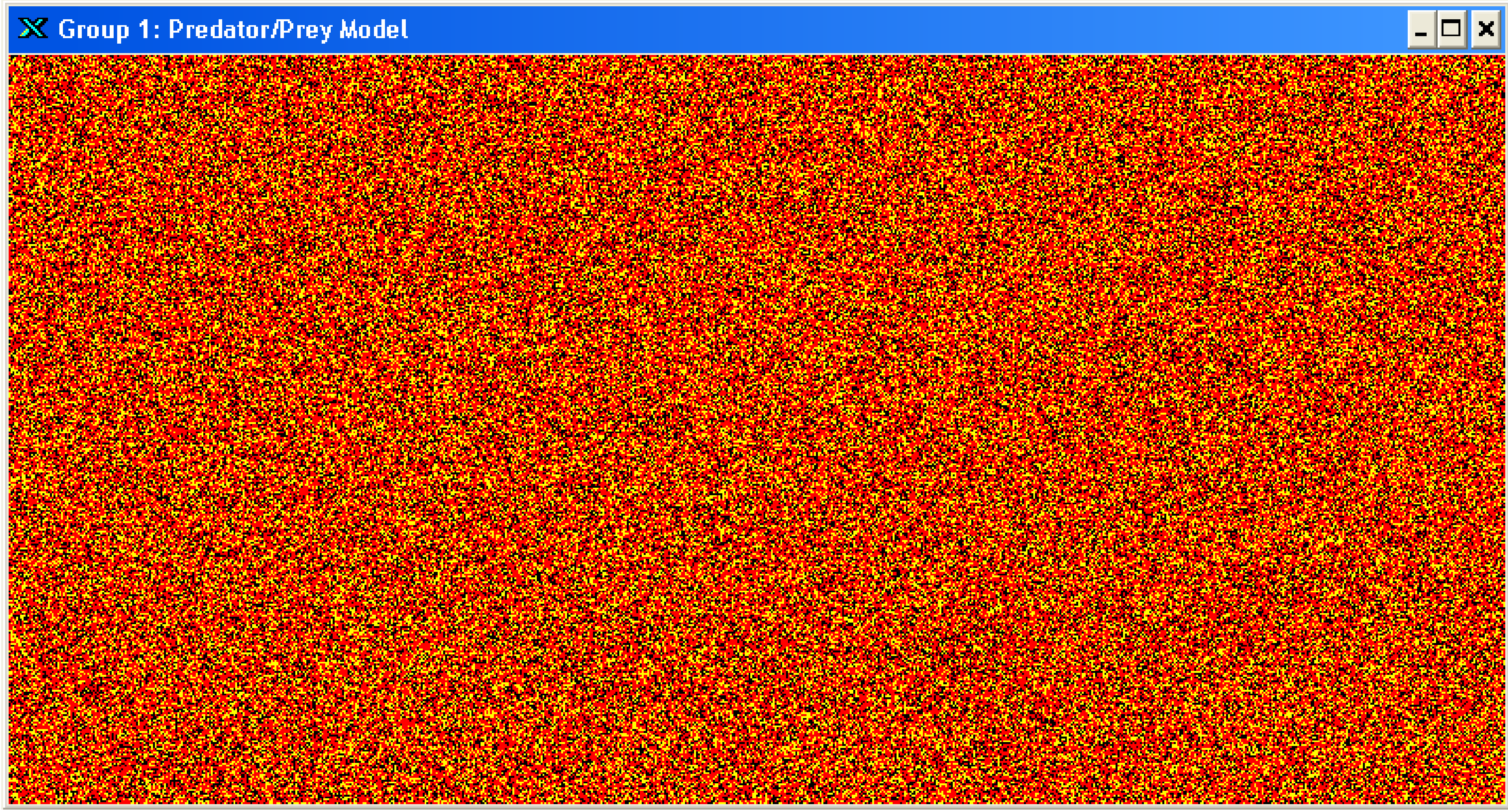
- At any one (x, y) position, value is:
  - Positive integer (fish present)
  - Negative integer (shark present)
  - Zero (empty cell)
  - Absolute value of cell is age

```
// If the current cell is empty, check to
// see if a shark or a fish is born
if (*current == 0)
{
  if ((fish >= 4) && (fish > sharks) && (fishbreed > 2))
    (*uu)(i,j) = 1;
  else
  {
    if ((sharks >= 4) && (sharks > fish) && (sharkbreed > 2))
      (*uu)(i,j) = -1;
    else
      (*uu)(i,j) = (*current);
  }
  continue;
}
```

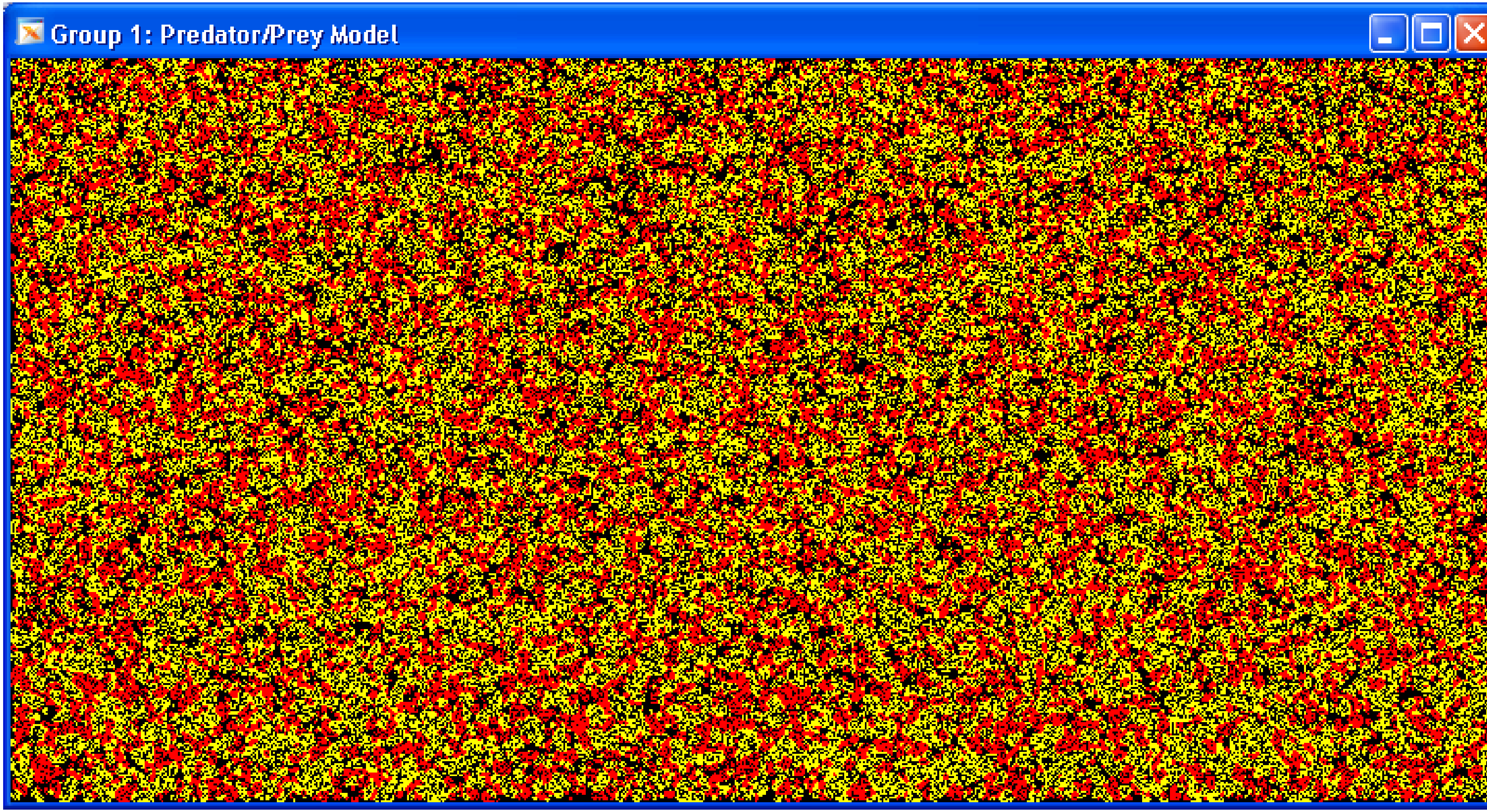
Another CA Example: Sharks and Fish – Illustration

- Behavior over a span of 10,000+ generations (about 25 minutes on a cluster of 20 processors)
    - Fish = 1 (red pixel)
    - Sharks = –1 (yellow pixel)

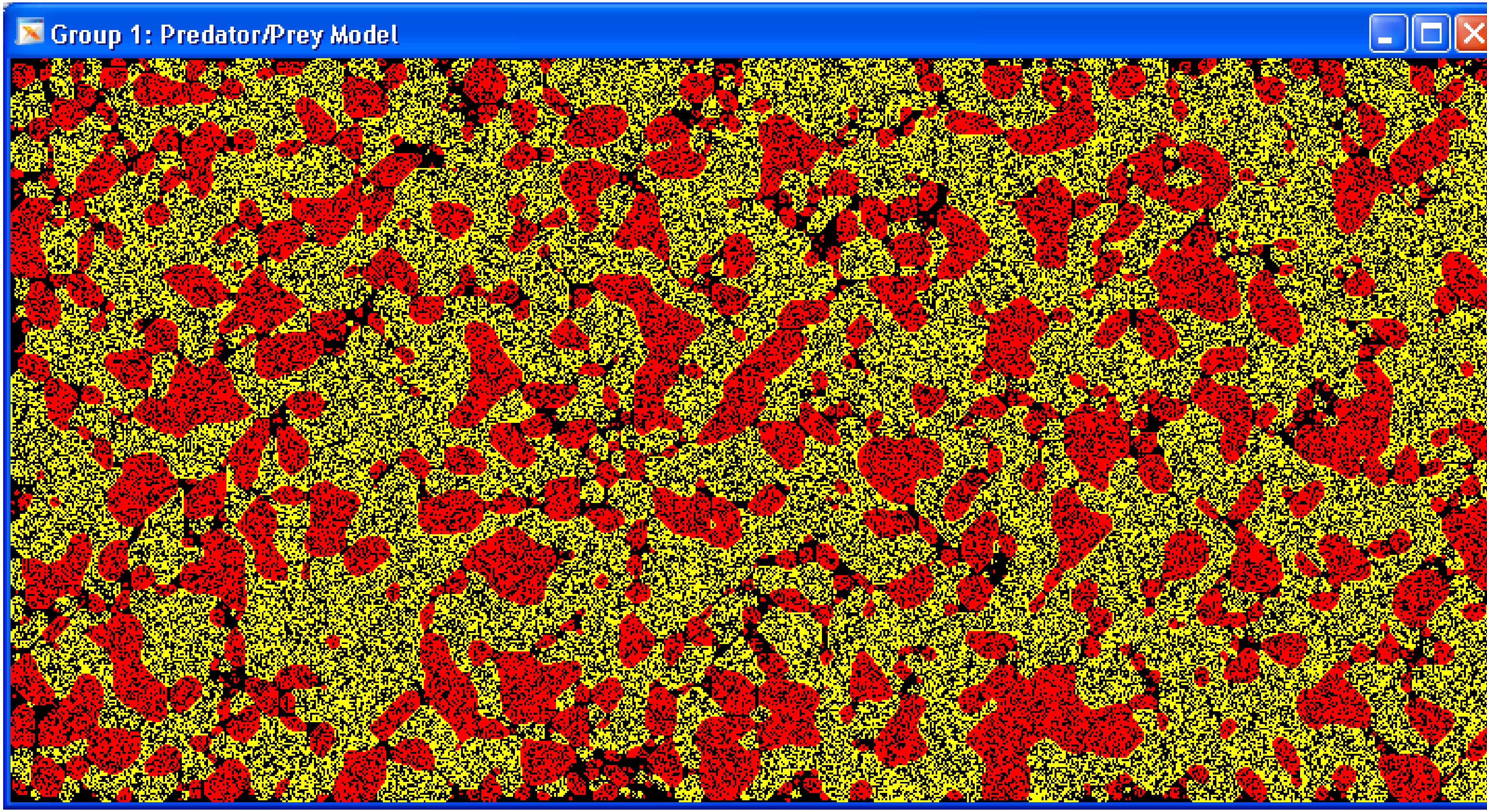
Generation: 0



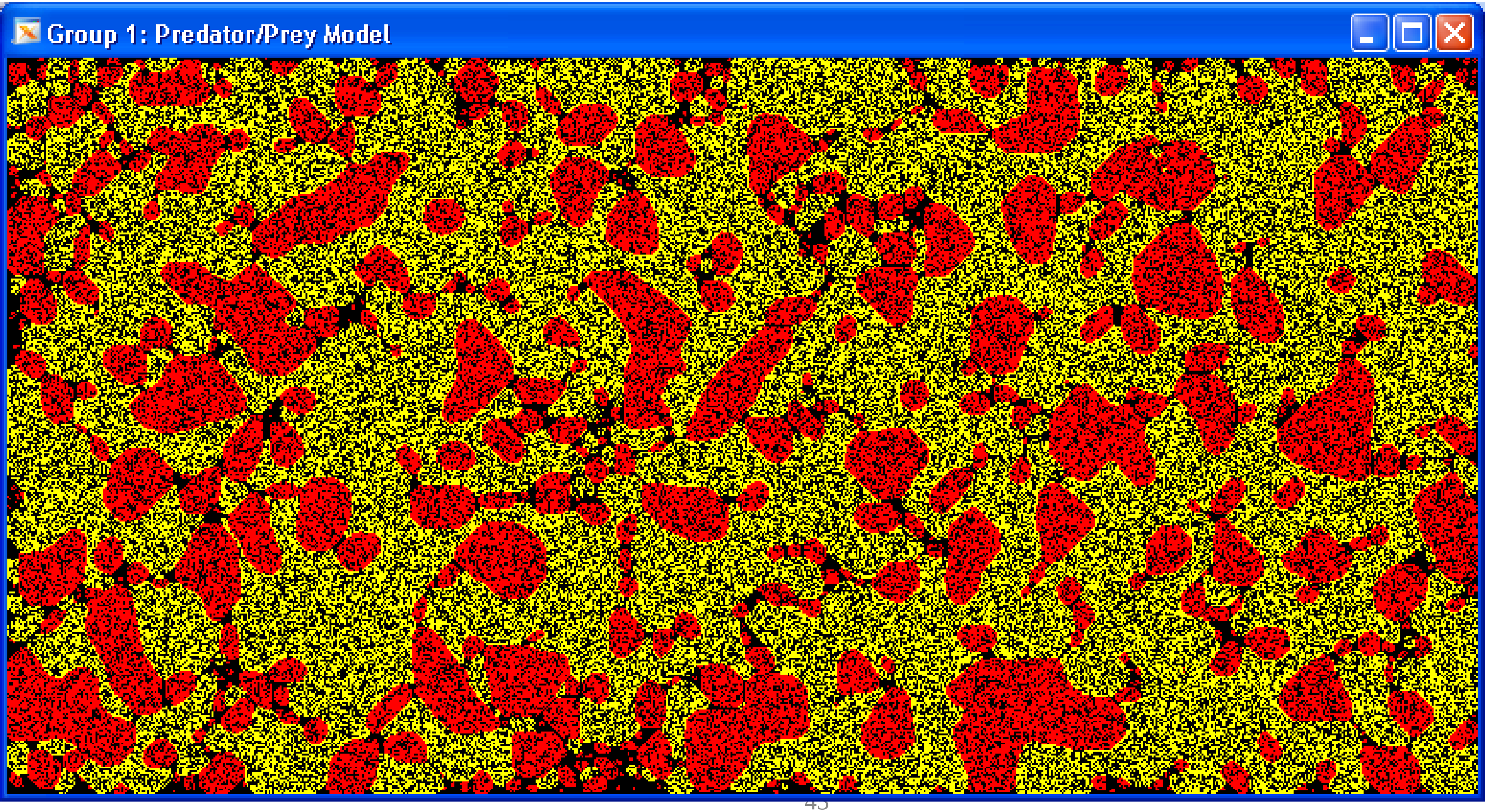
Generation: 100



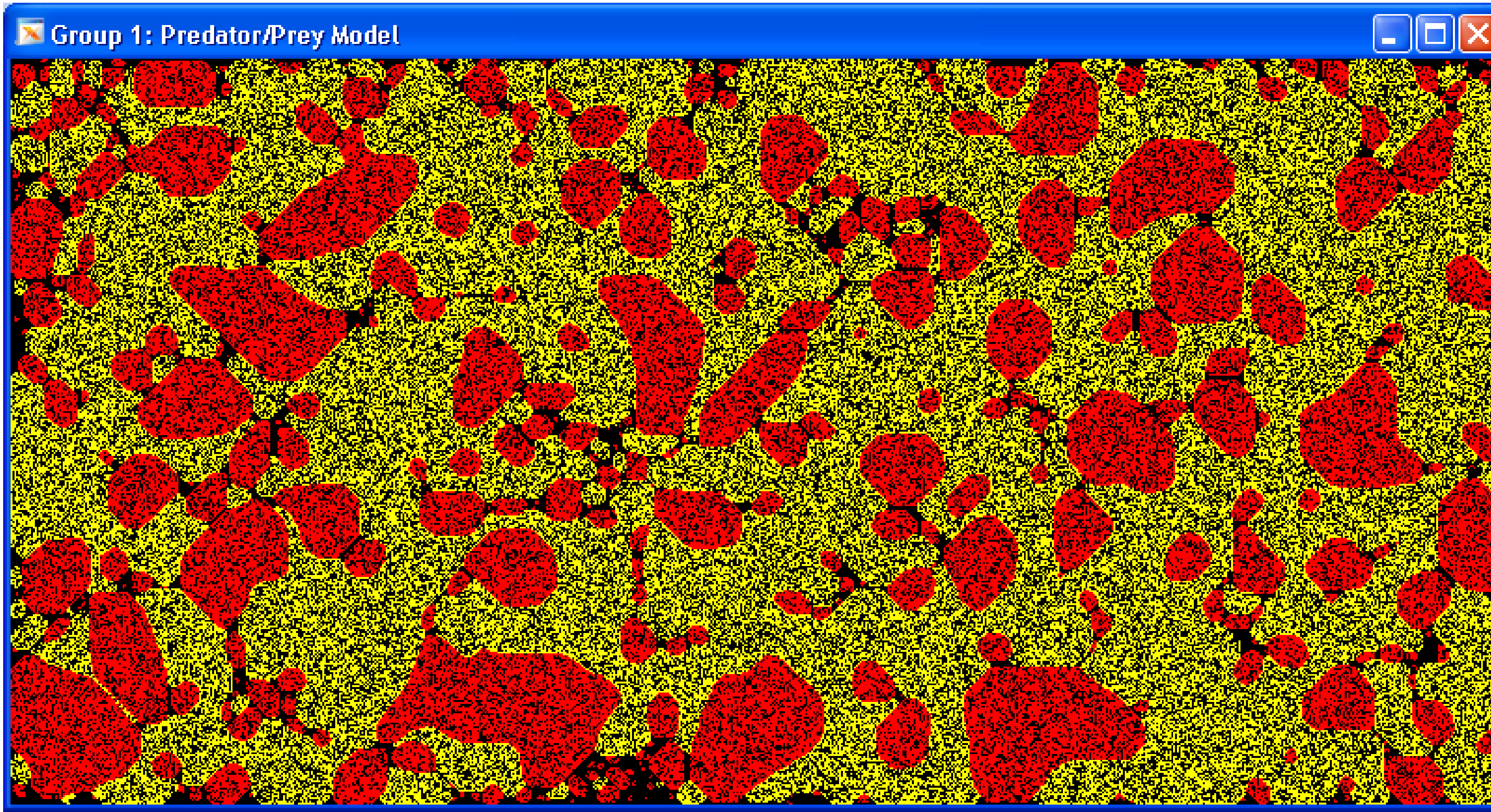
Generation: 500



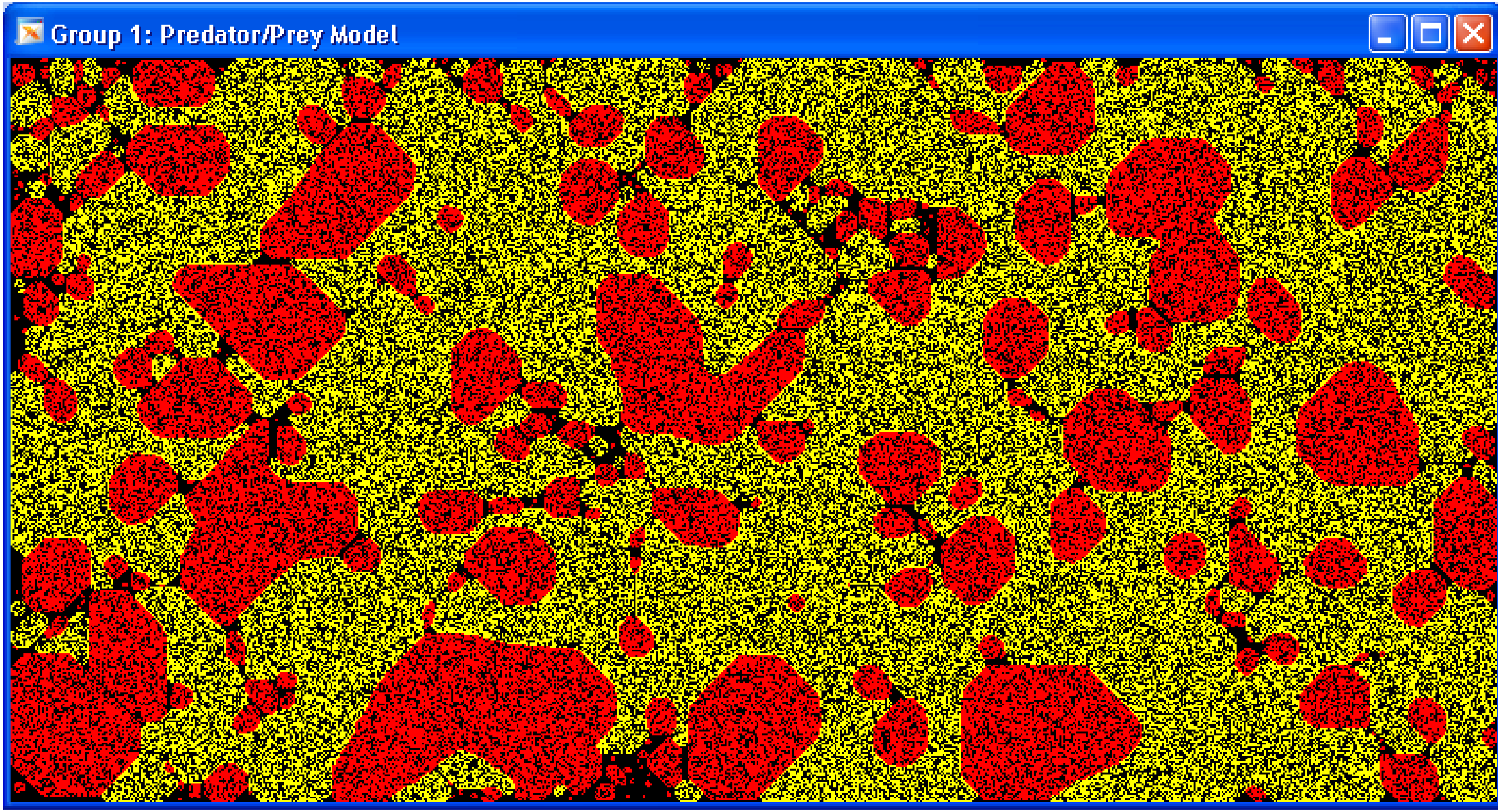
Generation: 1000



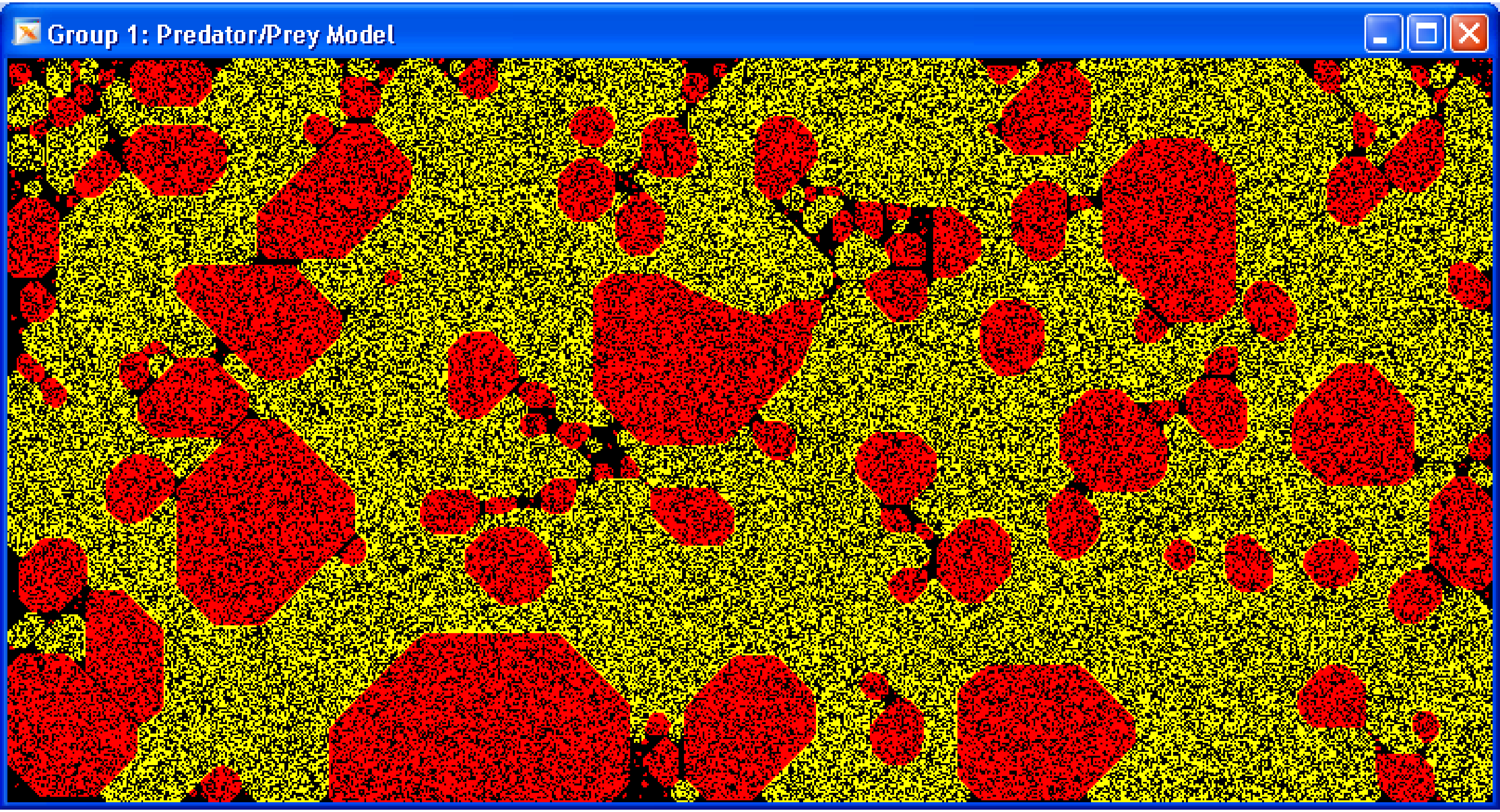
Generation: 2000



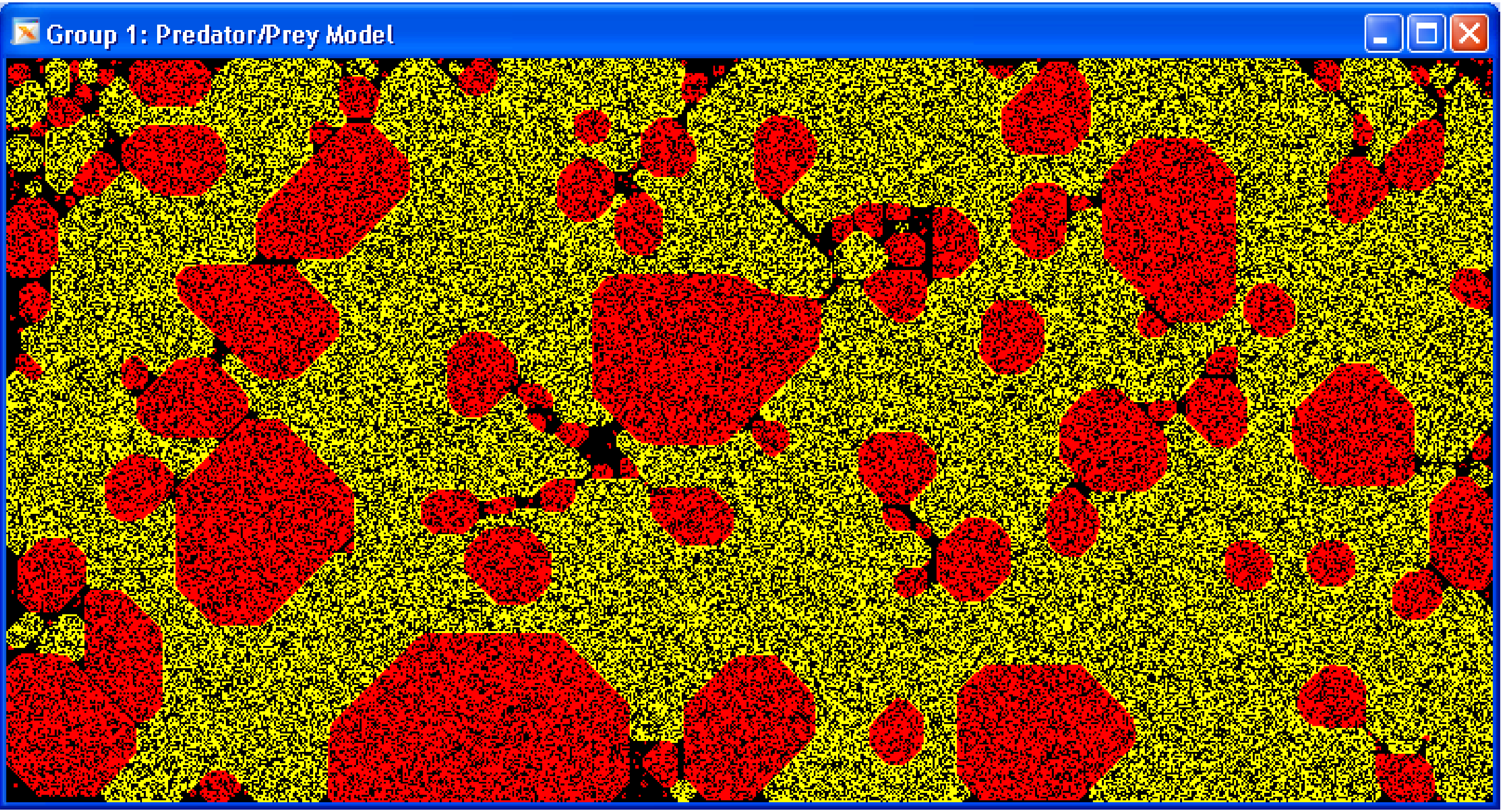
Generation: 4000



Generation: 8000

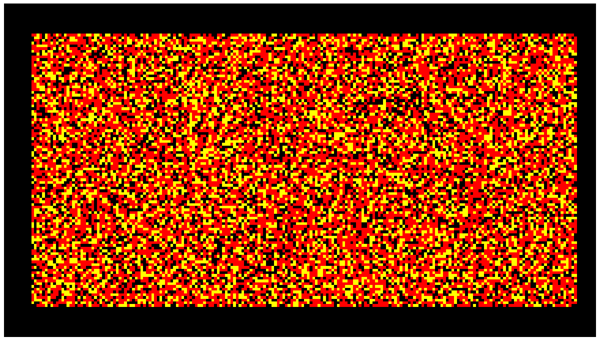


Generation: 10000

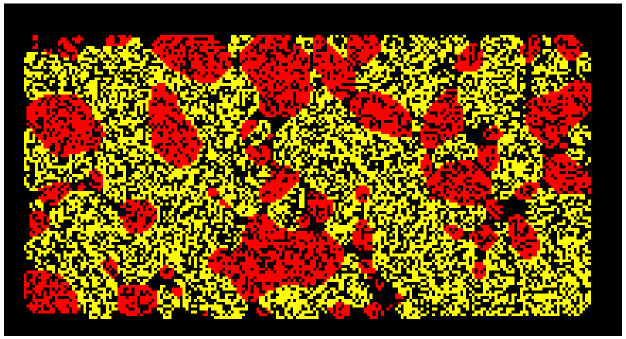


What if Initial Pattern Changes?

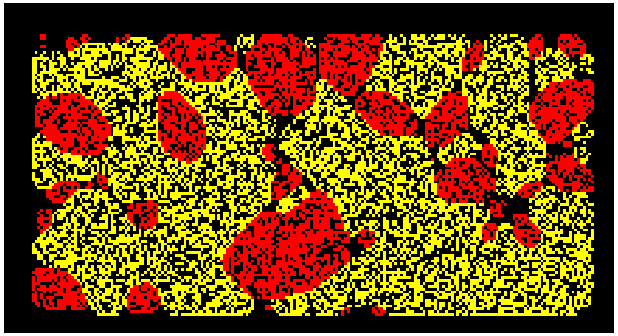
Generation 100



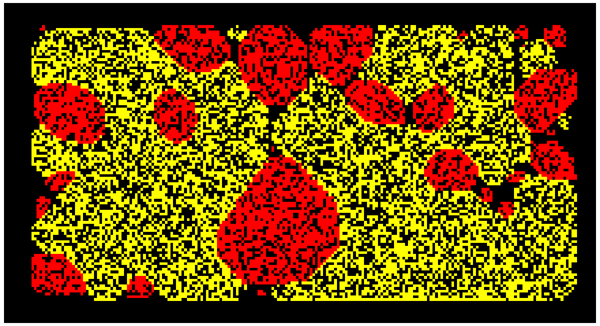
1000



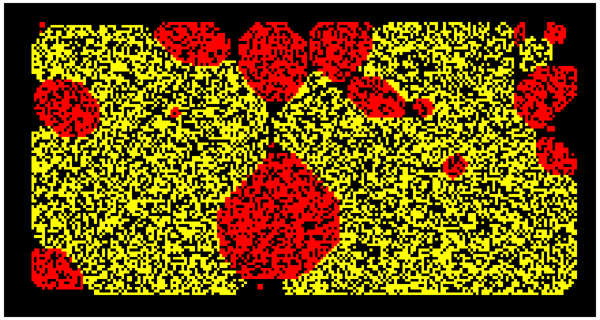
2000



4000

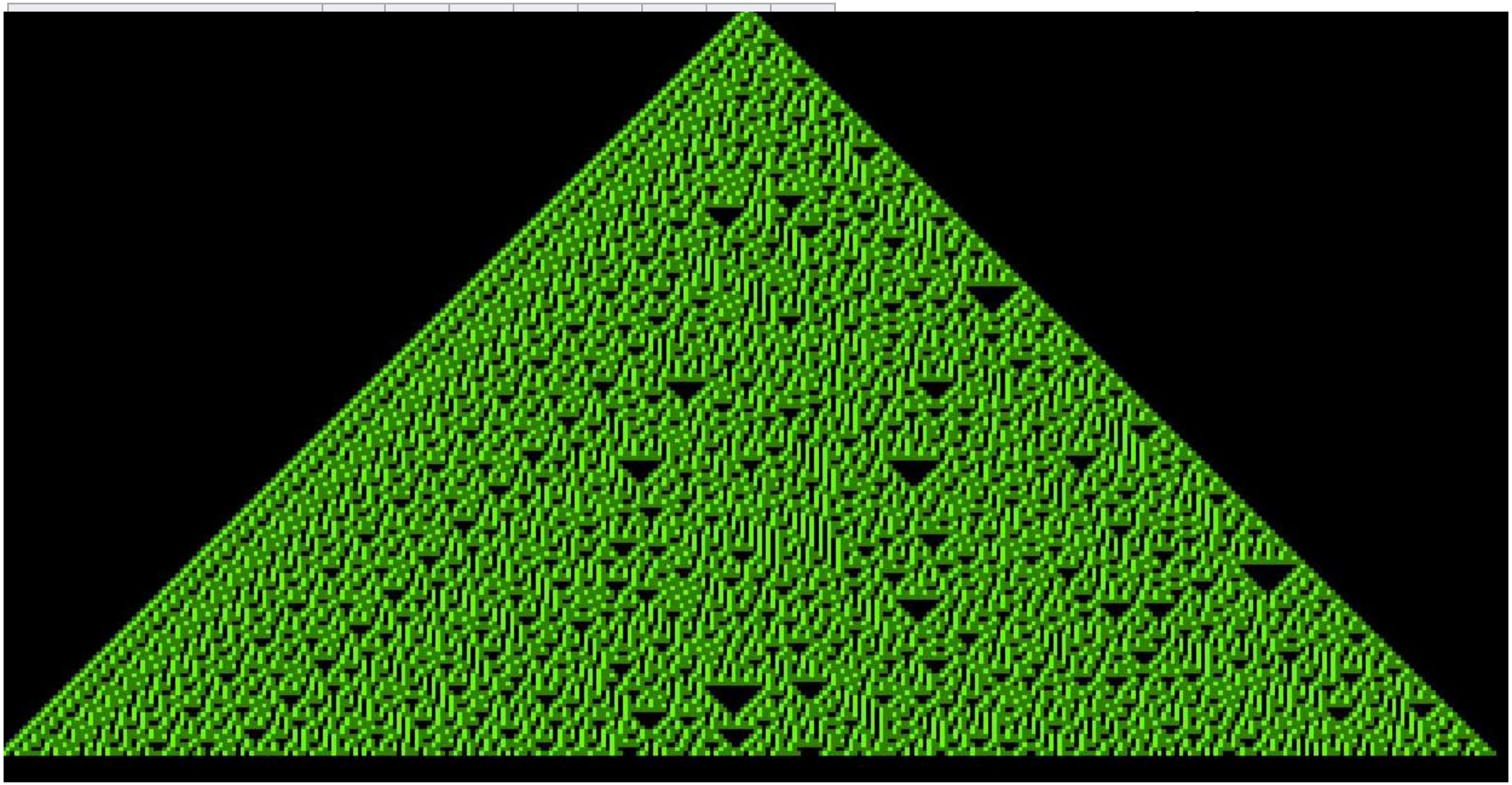


8000



- Random placement of very small populations can favor one species over another
- Fish favored: sharks die out
- Sharks favored: sharks predominate, but fish survive in stable small numbers

Another CA Example: Wolfram's Elementary 1D Automata



http://mathworld.wolfram.com/ElementaryCellularAutomaton.html

CA Software

- 3D:

https://cubes.charliedeck.com/

- 2D:

https://sourceforge.net/projects/golly/files/

- 1D:

https://www.wolframalpha.com/examples/science-and-technology/computational-sciences/cellular-automata/