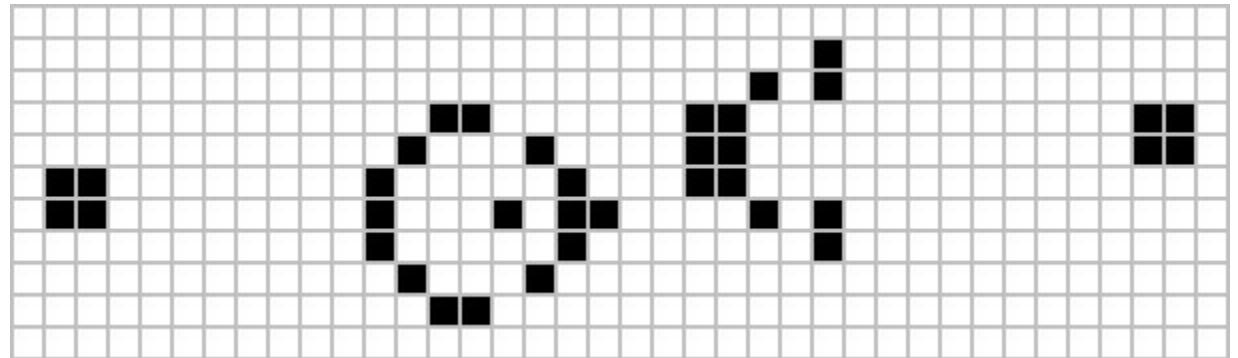


Introduction

- Cellular automata represent a concept which has been discovered under various names and with different purposes in a multitude of fields, such as pure mathematics or electrical engineering.
- Stephen Wolfram, author of the study "A New Kind of Science", believes that cellular automata are proof of a fundamental principle in nature, which states that "very simple programs produce great complexity".

CA are mathematical models for discrete dynamical systems, of simple construction but complex and varied behaviour.

A cellular automaton consists of a uniform lattice of sites, usually infinite in extent, with a finite set of possible values.



Main points

- Cellular automata
- Game of Life
- Reversible CA
- Applications

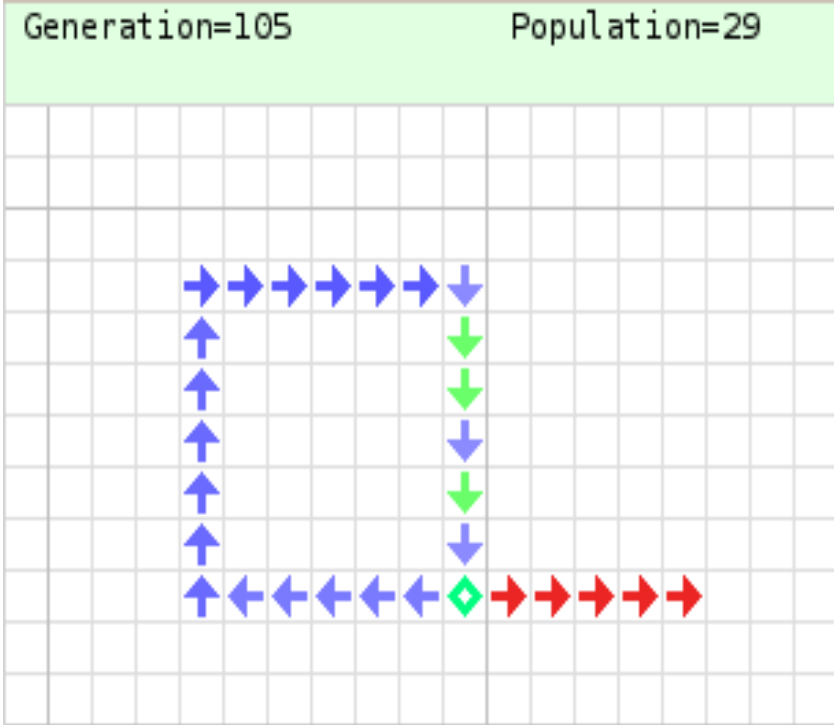
History

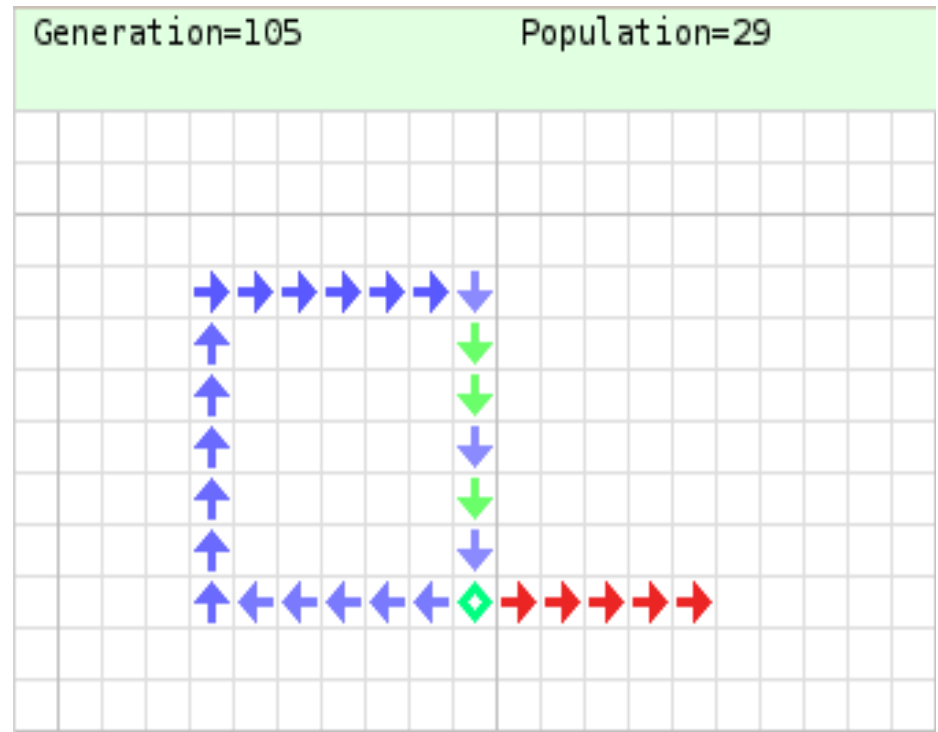
1940s - 1950s

- Stanislaw Ulam was among the first to consider a similar system to a cellular automaton while studying the growth of crystals
- John von Neumann (right) wanted to be able to build self-replicating machines and realizes the potential of an abstract approach, such as cellular automata systems



1950s - 1960s and von Neumann's CA

- 1952 - 1953 - von Neumann develops the first widely acknowledged cellular automaton
 - 2D cellular automaton
 - each cell has 29 possible states
 - is a self replicating machine
 - helps understand self replication
- 

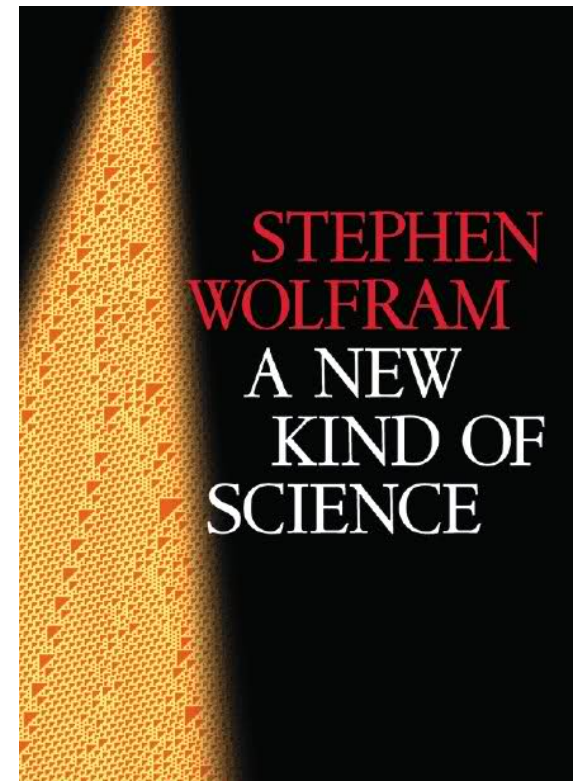


Further research and progress

- cellular automata's mathematical properties are being investigated
- some, for instance Edward Fredkin, consider the idea that cellular automata systems can simulate the laws of physics, not only real-world derived phenomena
- new rules and models are being built, constantly enriching the collection of known automata
- with its appearance, Conway's Game of Life introduces cellular automata to a wide public

Key work: A New Kind of Science

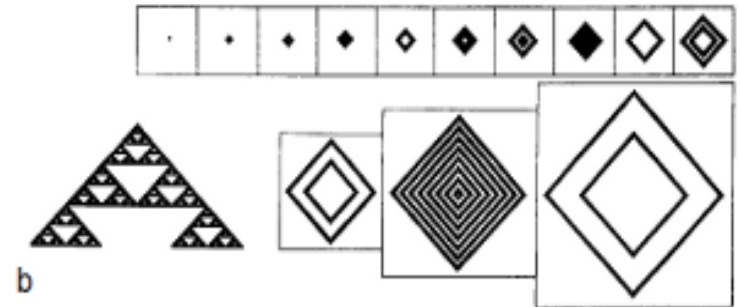
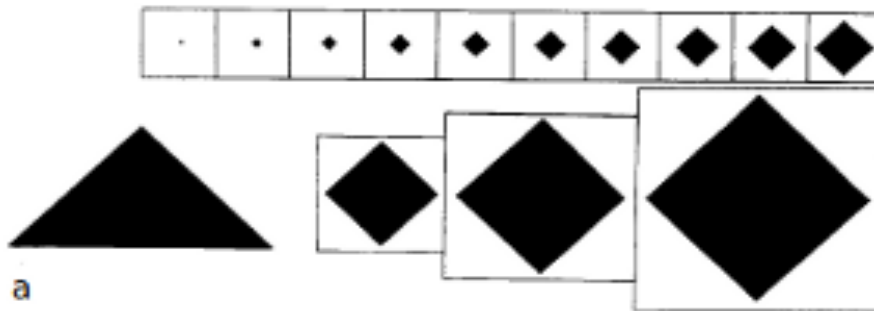
- published by Stephen Wolfram in 2002
- represents an elaborate study of computer systems such as cellular automata
- explores a multitude of new rules and configurations and is one of the most valuable resources on this field



Evolution from simple seeds

The evolution of cellular automata follows a variety of rules:

- uniform growing patterns



- boundary patterns
- non-homogeneous patterns, with complex appearance



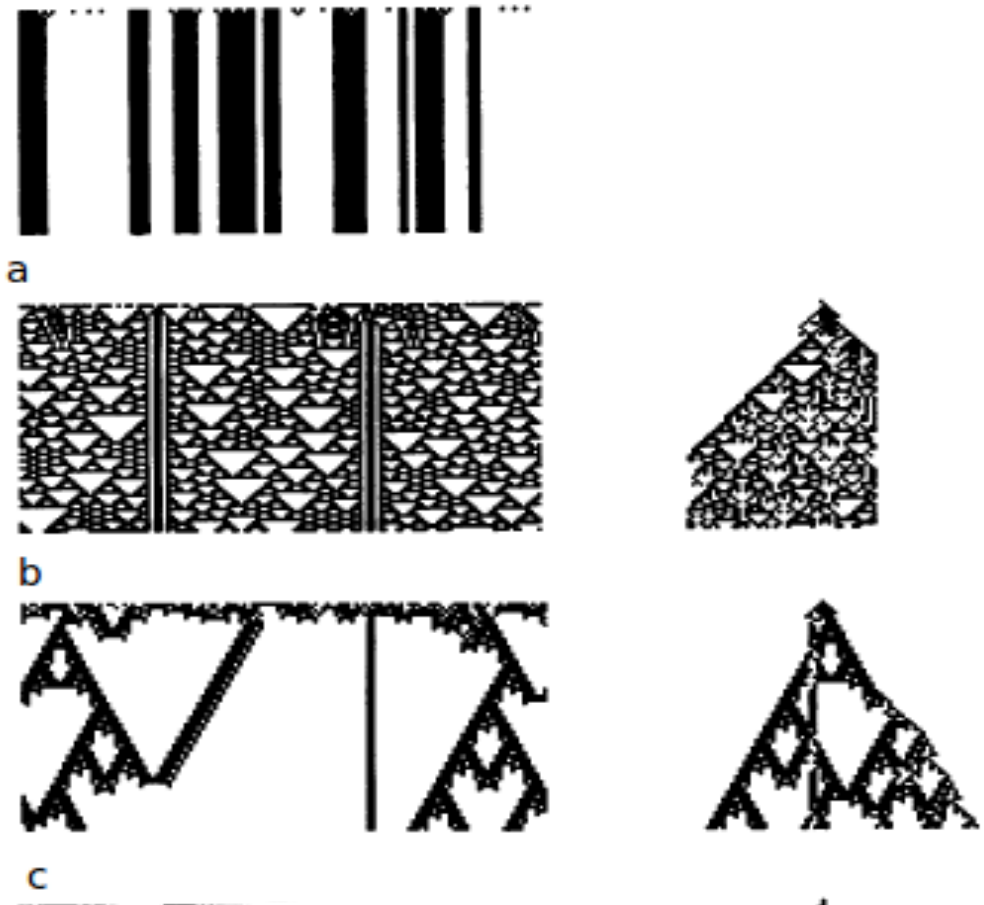
Four classes of behaviour for one-dimensional CA have been identified:

1. evolve to homogeneous final states
2. yield separated periodic structures
3. display aperiodic patterns with chaotic behaviour
4. generate localized and propagating structures

Evolution from disordered initial states

The evolution rules are like a "filter", preserving only some features of the initial states.

It is unstable in many cases, as small changes in the initial state can result in increasing changes of pattern over a period of time.



Phases in the evolution of cellular automata have as parameters spatially periodic patterns.

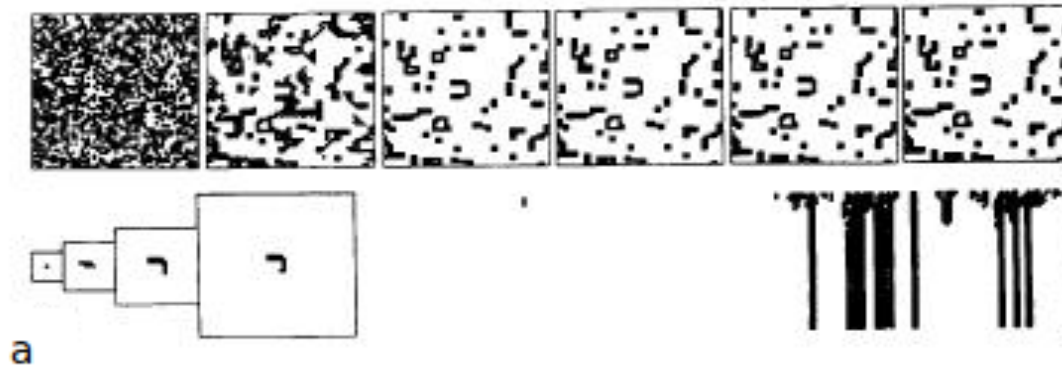
Two-dimensional cellular automata may contain

- point "defects" - localized regions in the domains
- line "defects" - walls that separate the domains

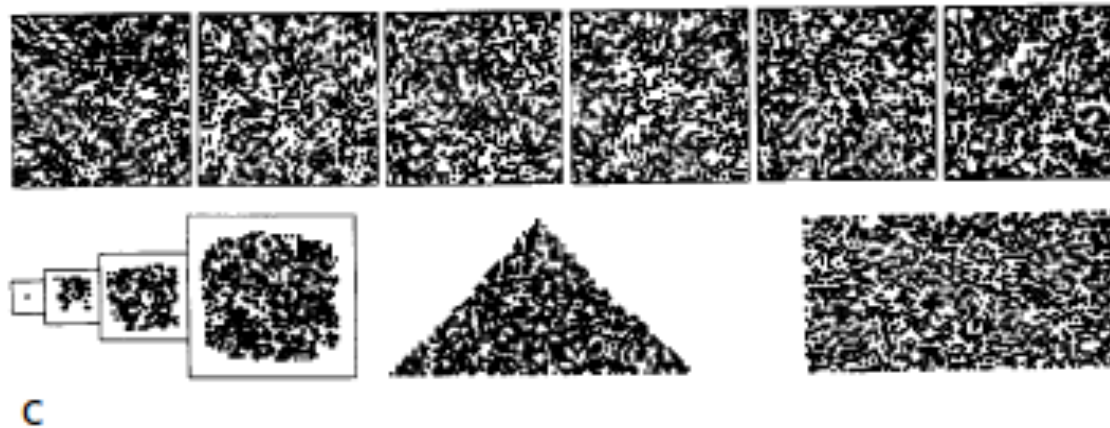


Cellular Automata Types

For class-2 cellular automata, the changes in the initial pattern make a difference only on a finite region and interval of time.

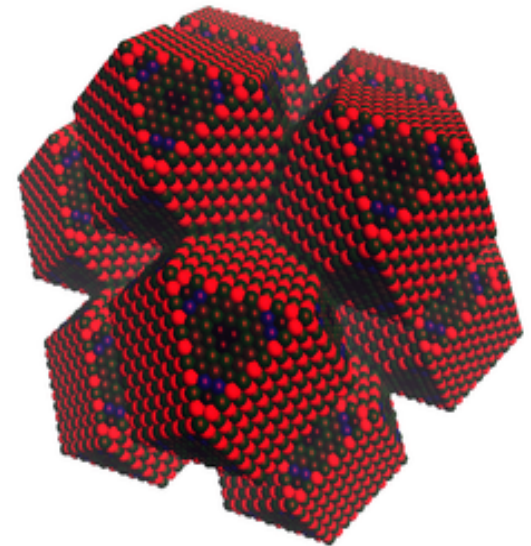
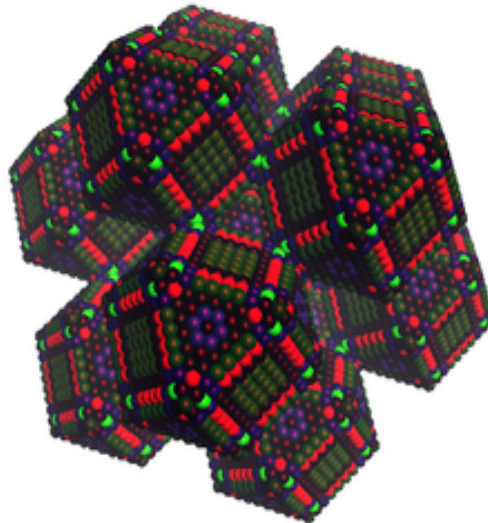
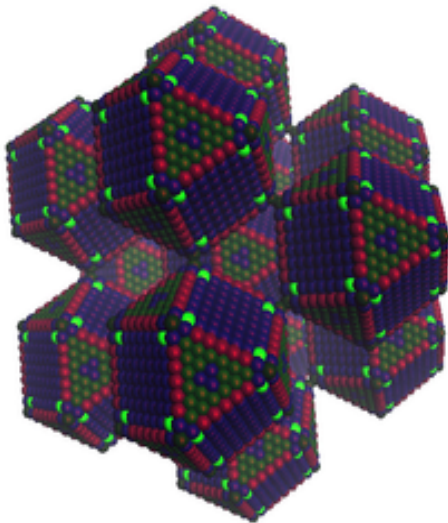


For class-3, the information tends to propagate forever, so the patterns grow without bound.



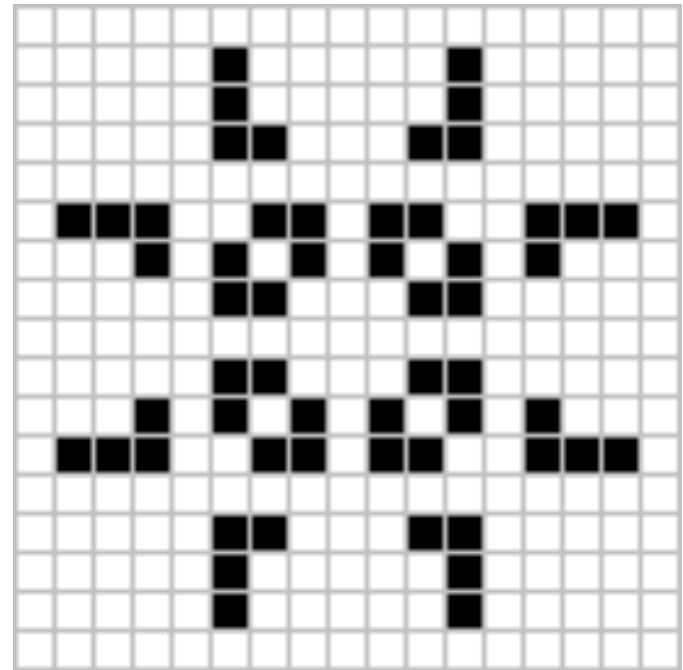
The expansion occurs at the same speed in all directions, thus resulting in an asymptotically circular difference pattern.

Totalistic Cellular Automata - the state of each cell depends on the sum of the values of the sites in the neighbourhood.



Game of Life

- When did it appear?
- Who created it?
- What is it?



- **When did it appear?**

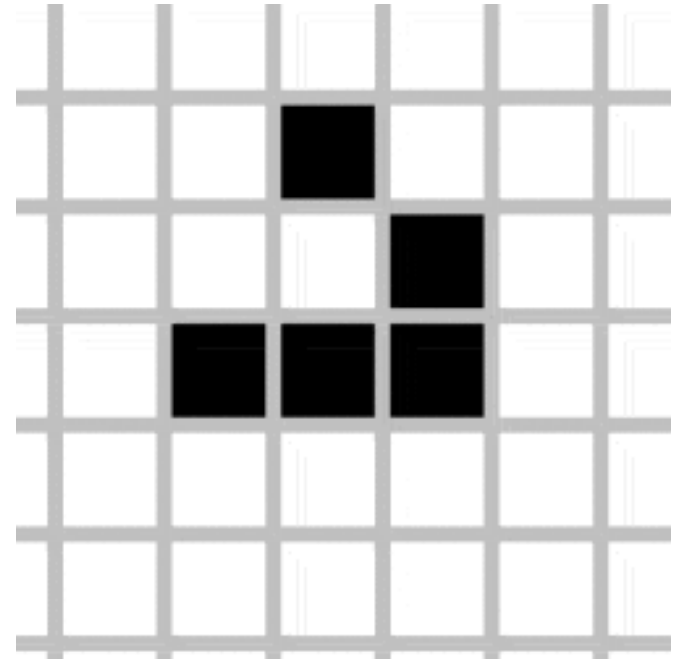
- experimenting with various rules began around 1968
- first revealed in the October 1970 issue of the Scientific American magazine

- **Who created it?**

- John Horton Conway, British mathematician at Cambridge University

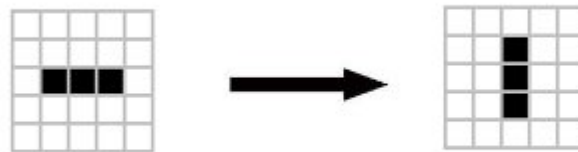
- **What is it?**

- a two-dimensional cellular automaton
- the cells have two possible states - alive or dead - and 8 neighbours
- has simple rules for births and deaths
- a zero-player a game

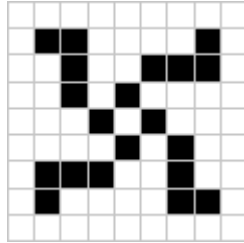
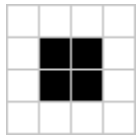


The Rules of Life

- Births : when a dead cell has exactly 3 live neighbours
- Deaths : when a live cell has less than two or more than three live neighbours
- Survival : when a live cell has 2 or 3 live neighbours

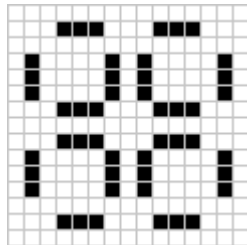
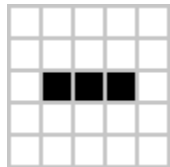


Many interesting patterns appear (or are made to appear) in Game of Life:



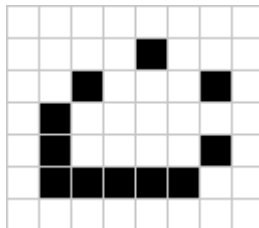
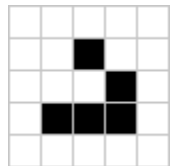
Still lives

Patterns that do not change



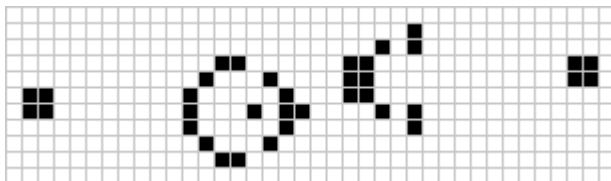
Oscillators

Move between several states



Spaceships

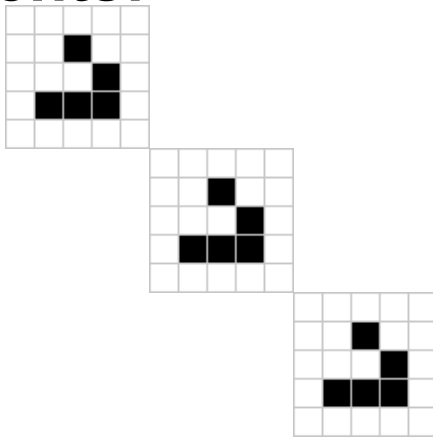
Travel along the Universe



Constructs

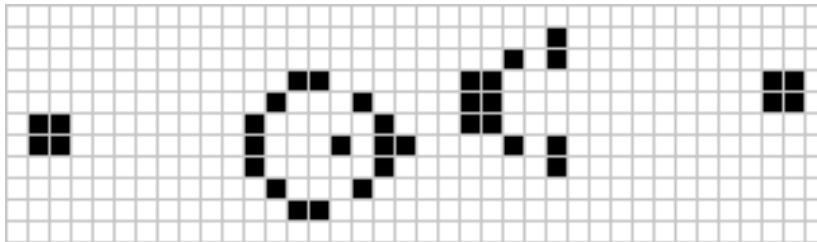
Created by humans, various roles

Logic gates can be created from three simple ingredients:



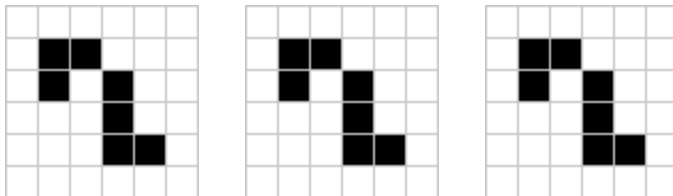
Gliders

They form the “bit stream”



Glider Guns

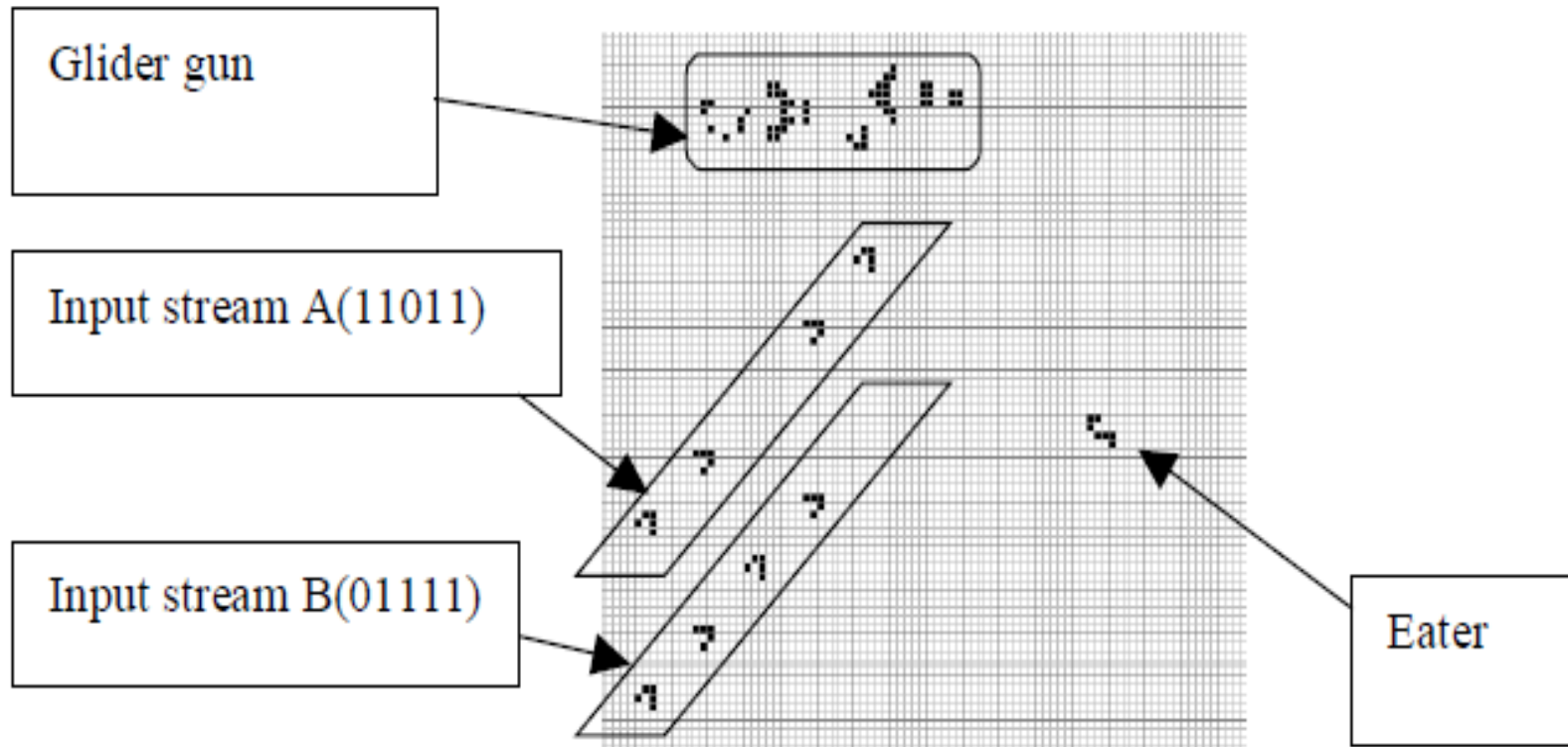
They create new streams to interact with the input



Eaters

They destroy gliders

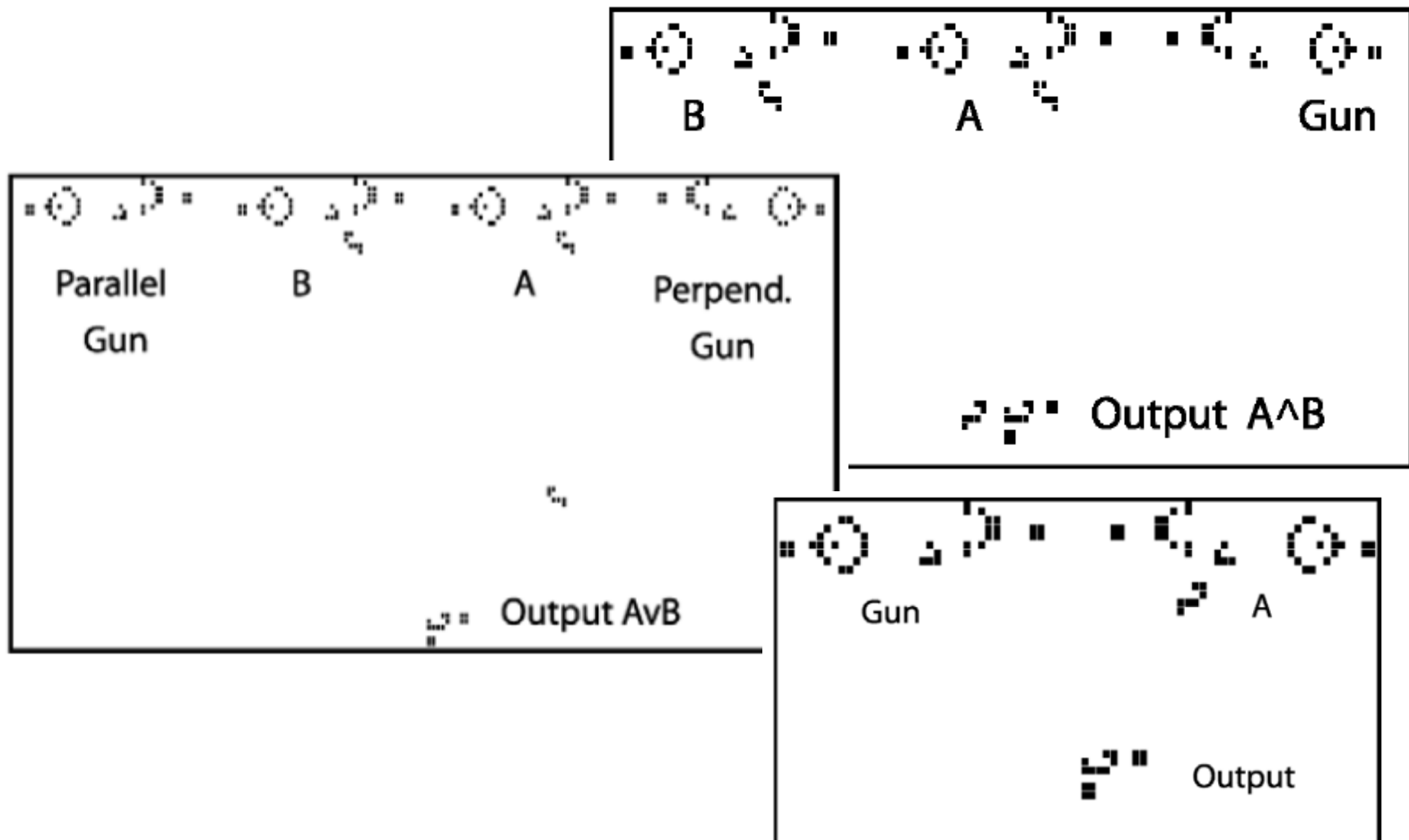
An AND gate example



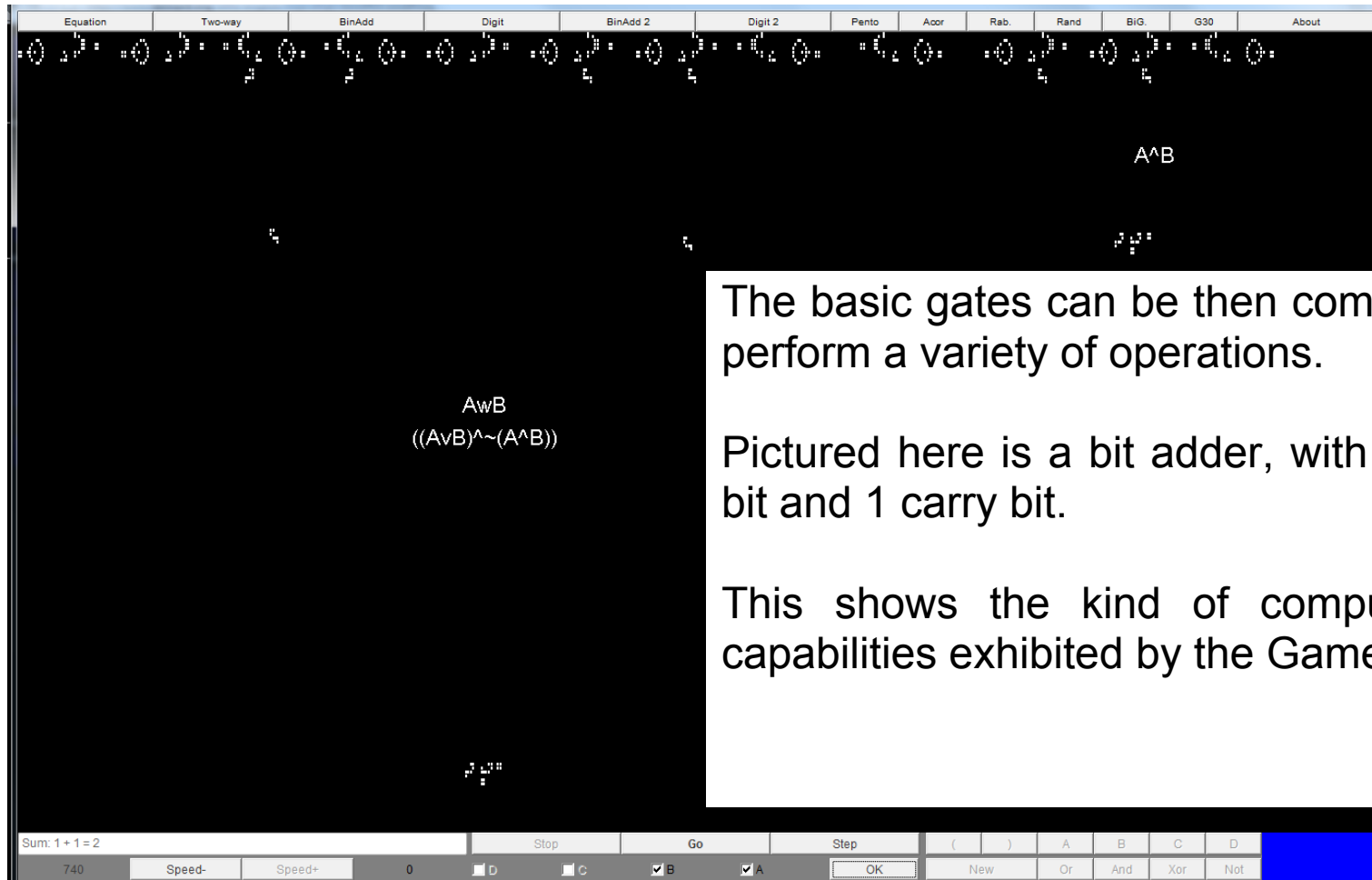
AND gate credit : Sapin, Bailleux and Chabrier

Rennard's LogiCell

A complete system for Boolean Operations in Life



Rennard's LogiCell



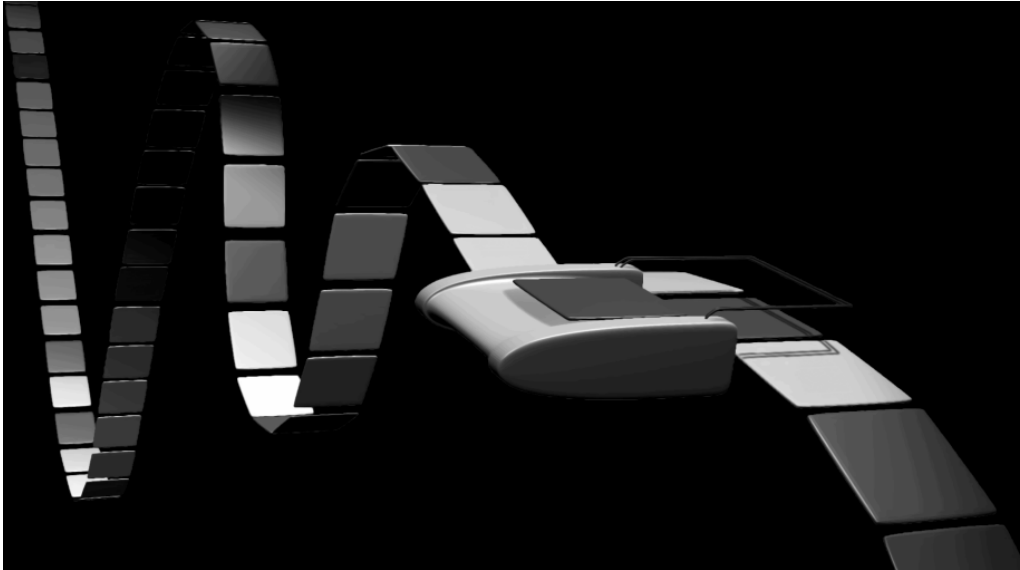
The basic gates can be then combined to perform a variety of operations.

Pictured here is a bit adder, with 1 result bit and 1 carry bit.

This shows the kind of computational capabilities exhibited by the Game of Life

Screenshot of LogiCell performing bit addition

A Turing Machine?



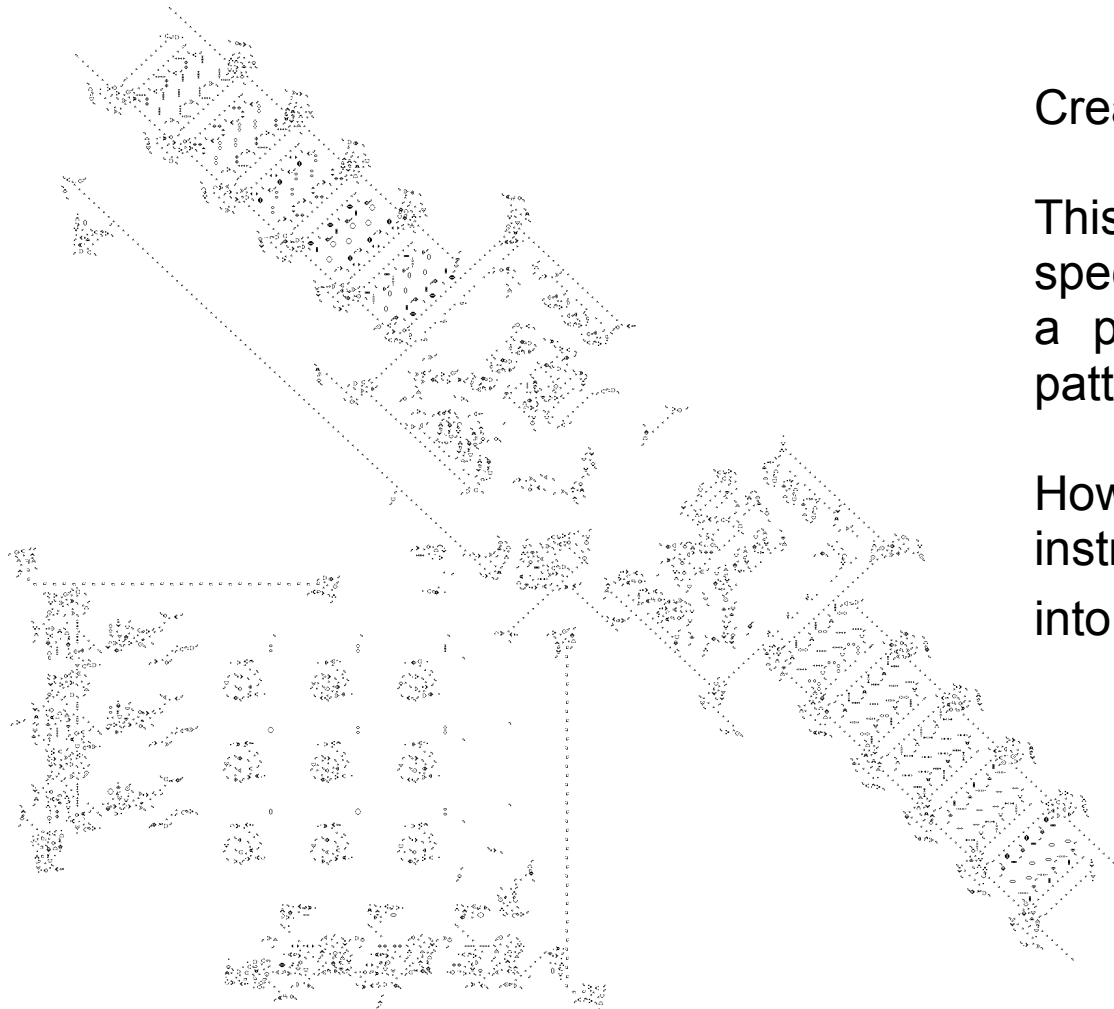
A simple conceptual device that can perform any possible computation

Consists of:

- A **tape** divided into cells, containing symbols from a finite alphabet
- A **head** which can read and erase symbols
- A finite **table** of instructions (transition function)
- A **state register** which holds the current state

Building one would surely prove the computational power of cellular automata!

A Game of Life Turing Machine!



Created by Paul Rendell, 2000

This 36549 cell pattern is a specific Turing Machine, running a program which duplicates a pattern of 1s.

However, by extending the instruction tape, it can be turned into a Universal Turing Machine.

A Universal Game of Life Turing Machine!

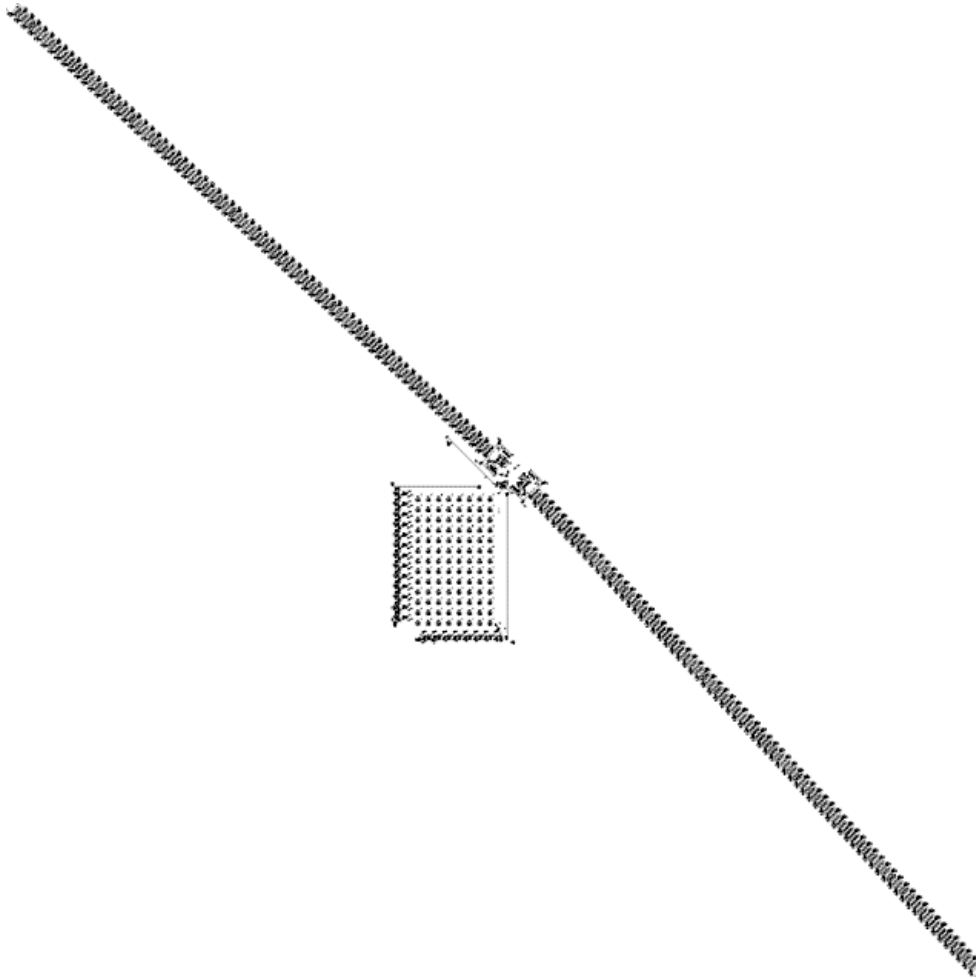
Created by Paul Rendell, 2010

The Universal Turing Machine can simulate all other Turing Machines.

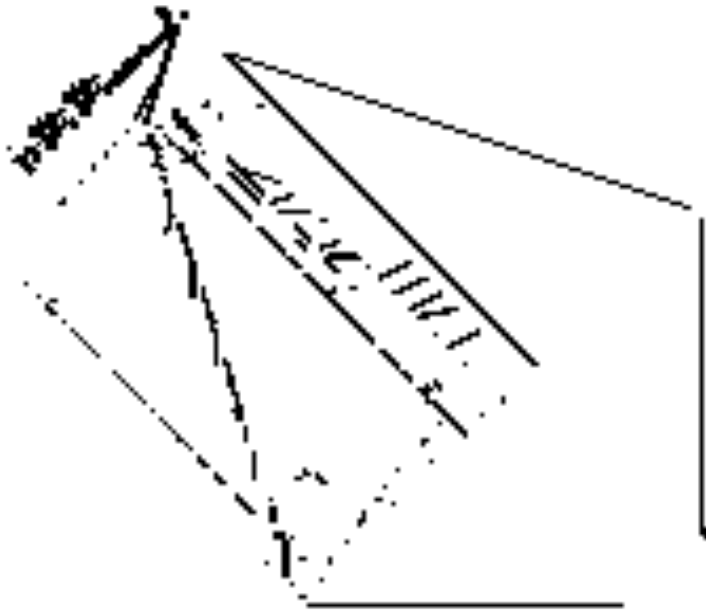
This is achieved using:

- 252192 cells
- 13 states
- 8 tape symbols

One cycle takes 79×240 generations.



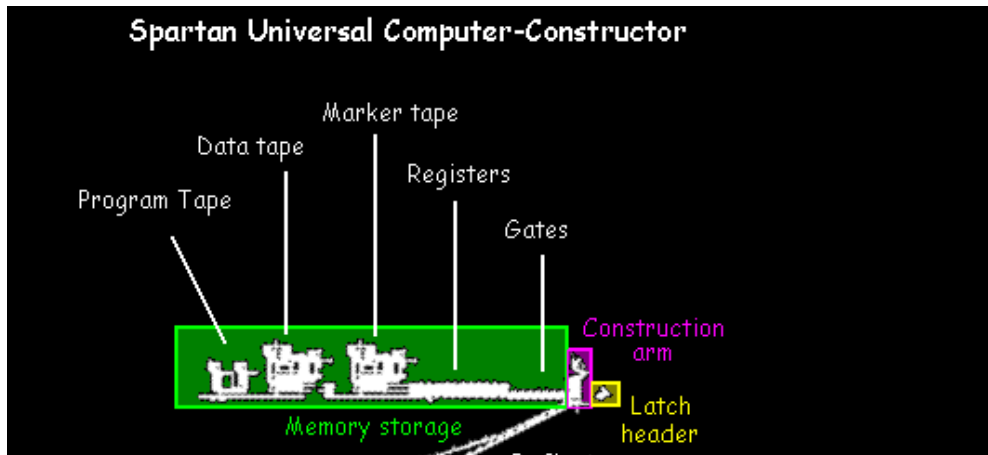
The Spartan Universal Computer-Constructor



Created by Adam P. Goucher, this computer is composed of “Spartan” still lives (under 7 cells each).

- 11 memory registers
- Three tapes: program, data marker
- A construction arm capable of synthesizing any pattern

The Spartan Universal Computer-Constructor



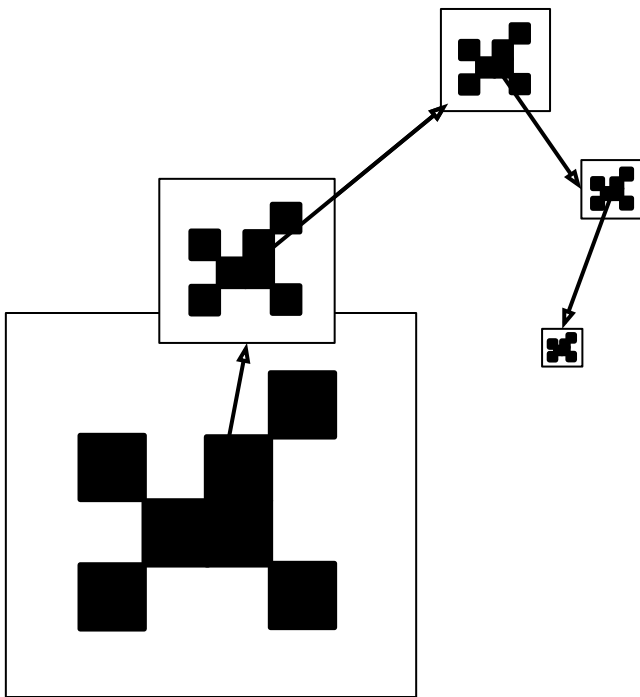
Detail of memory and construction arm

This pattern uses binary storage and is capable of a many operations, as well as creating patterns from the instructions given.

Theoretically, it could create a copy of itself. It would take 10^8 generations!

The pattern total size is 481672 cells.

A Life Cell inside Life



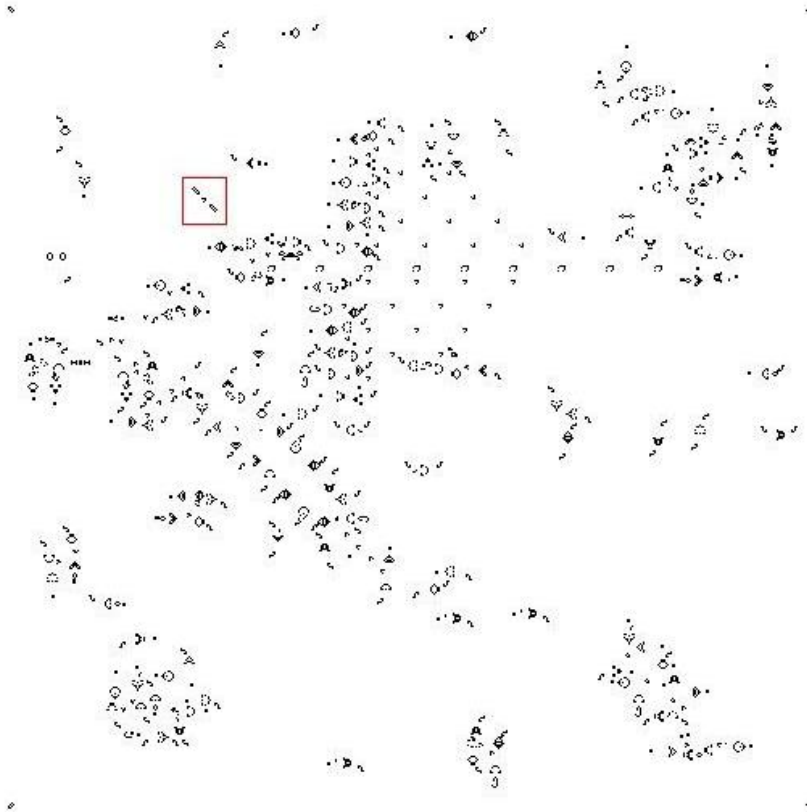
"a pattern with two states, which is determined by its previous state and the previous state of its neighbors, using exactly the rules used to compute it; that is, it simulates its own universe."

Alan Hensel's lexicon

It is possible to build one, with two notable examples:

- David Bell's Cell – 1996
- OTCA metapixel – 2006

David Bell's Cell

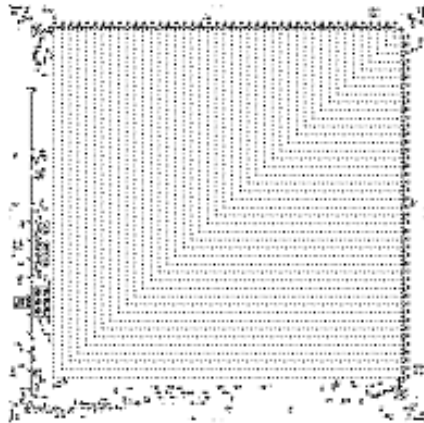


5760 generations make up a cycle - one generation in the universe simulated by this cell.

After each cycle, the cell shoots a glider if it is still is or has become alive.

While it functions correctly, this design fails to visually simulate a Game of Life universe.

Outer Totalistic Cellular Automata Meta-Pixel - Brice Due

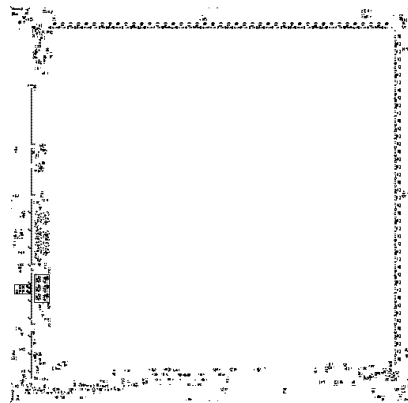


ON

The OCTA Meta-Pixel has clearly distinguishable ON and OFF states.

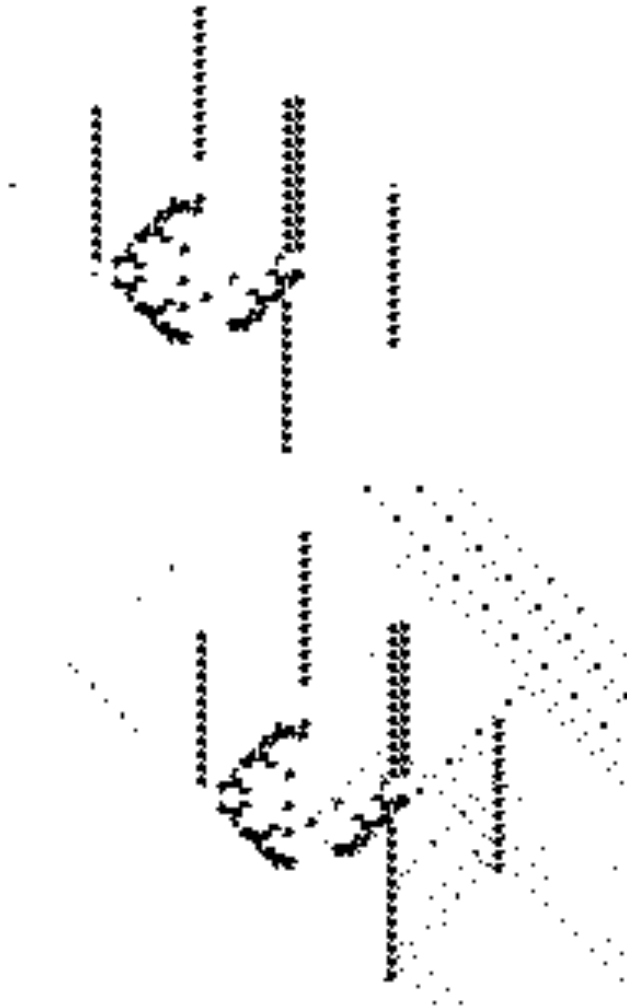
It can not only simulate Game of Life, but also any other similar 2-D cellular automaton .

OFF



It has 64691 cells and one cycle takes 35328 generations.

Gemini – by Andrew J Wade, 2010



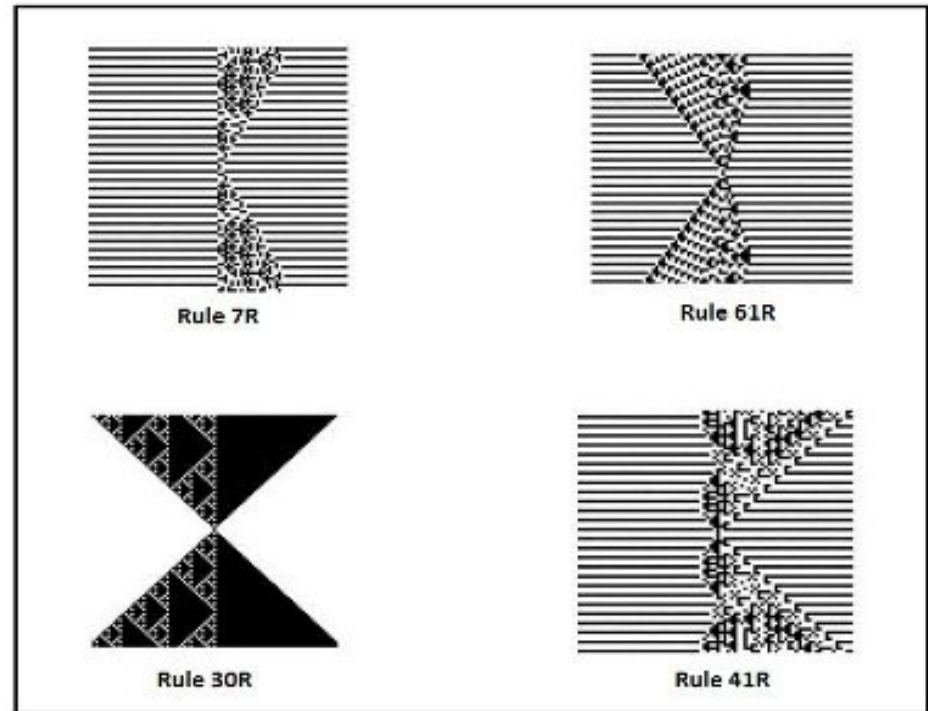
It is the first oblique spaceship – it travels in a direction that is neither orthogonal nor diagonal.

The way in which it achieves this that is truly innovative it moves by self replication.

An immensely long “instruction tape” of gliders carries all the information required to build the spaceship and the tape itself. At both ends of the tape, a spaceship is constructed while its old version is destroyed – thus the name Gemini.

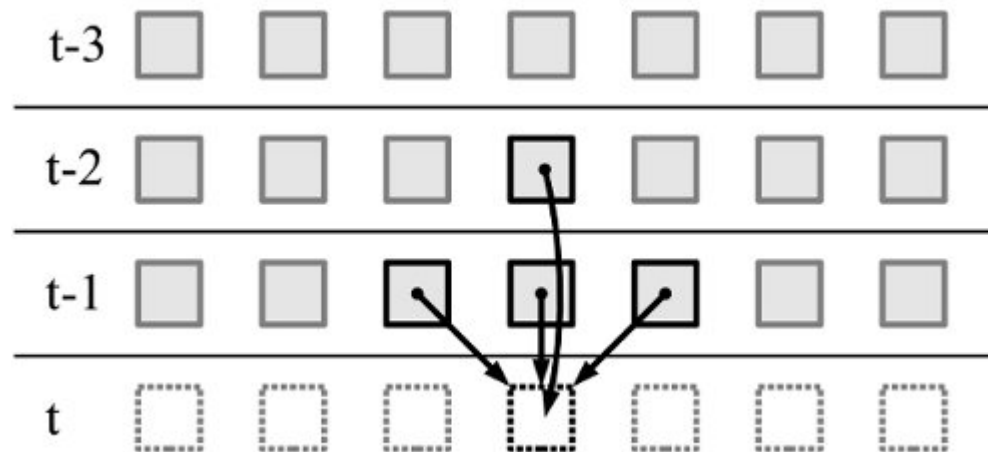
Reversible Cellular Automata

- are a natural model of Reversible Computing
- can simulate some natural processes
- reversibility is achievable, but not taken for granted: one most **program** it
- requires them to be deterministic



The second order technique

- a way to achieve reversibility in cellular automata
- idea: a cell's state at time t depends on the state of its neighbourhood at time $t-1$ and its own state at time $t-2$
- this is sufficient to obtain the 'reverse' transformations

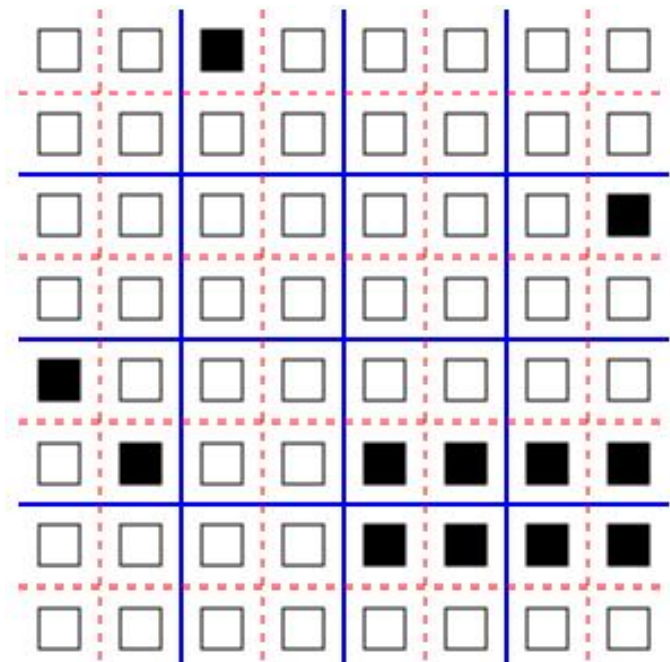


The partitioning technique

- a less mathematical approach than the second order technique
- uses a 'partitioning scheme' to divide cells into blocks, apply a rule to the whole block, and then re-partition cells into blocks differently
- a well known partitioning scheme is the **Margolus neighbourhood**

The Margolus neighbourhood

- partitions cells into 2x2 blocks
- two types of steps, where cells belong either to even aligned blocks (blue lines below) or to odd aligned blocks (red lines below)
- to obtain a reversible CA, the rule operating in a block must be invertible
- once we have it, we can apply the steps in reverse order to obtain the inverse transformation



Some applications of CA

- lattice gas automata
- Ising model (used to study the properties of magnetic systems)
- could be potentially used for encryption (especially Wolfram's Rule 30)
- can model fractal growth of organisms
- can generate multimedia content
- 1D CA can be used as random number generators

The making of

Add some lines of javascript a little html and some css for the flavor. Then stir well and taste. :)

Some statistics:

- javascript 1337 lines
- html 1706 lines
- css 3404 lines
- over 12,000 lines of open frameworks
- 0 cups of coffee

Game of life applet

- We used html5 canvas to draw the game. First paint an empty grid and catch any mouse event to place cells where needed.
- At every iteration traverse the grid and change cells bases on game of life rules Redraw!
- Add some extra functions to change speed, who doesn't like slow motion?

```
function nextStep() {  
  var change = [];  
  var algorithmTime, guiTime, liveCellNumber = 0;  
  
  algorithmTime = (new Date());  
  
  for(var i = 0; i <= N; ++i)  
  for(var j = 0; j <= M; ++j) {  
  
    var cnt = 0;  
    for(var k = 0; k < 8; ++k)  
      if( (i + xx[k] >= 0 && i + xx[k] <= N && j + yy[k] >= 0 &&  
        cnt = cnt + 1;  
  
    if(board[i][j] == true)  
      ++liveCellNumber;  
  
    if(board[i][j] == false && cnt == 3)  
      change.push( [i, j] );  
    else if(board[i][j] == true && cnt != 2 && cnt != 3)  
      change.push( [i, j] );  
  }  
  
  algorithmTime = (new Date()) - algorithmTime;  
  guiTime = (new Date());  
  
  //Update the grid  
  $.each(change, function(id) {  
    changeCellState(change[id][0], change[id][1]);  
    drawCell(change[id][0], change[id][1]);  
  });  
  
  guiTime = (new Date()) - guiTime;  
  ++generation;  
}
```

Timeline

- We parsed all the data stored in a json file. Then using a slightly modified version of the timeline js framework load and process it.
- The framework groups data based on year, duration and importance attributes. It constructs the applet for the user to browse all the data moving through time and space.
- Every item will display a box with its description and a picture when clicked.

Quiz

- We take all the content from a file and form a random list of questions.
- One each question we check if the answer is correct and adjust the score.
- We used jquery to update in real-time colors for a fancy display.
- If you miss the correct answer, detailed information is displayed in the left.

```
},  
{  
  "question": "In the Game of Life, when does a cell die out?",  
  "answers": [  
    "When it has greater than or equal to three live neighbours.",  
    "When it has less than two or more than three live neighbours.",  
    "When it has one live neighbour."  
  ],  
  "correct": "When it has less than two or more than three live neighbours.",  
  "info": "A cell with less than 2 live neighbours dies, as if by isolation ..."  
},  
}
```

Thank you for your attention!
Q & A

