Maze Generation, Cell Automata

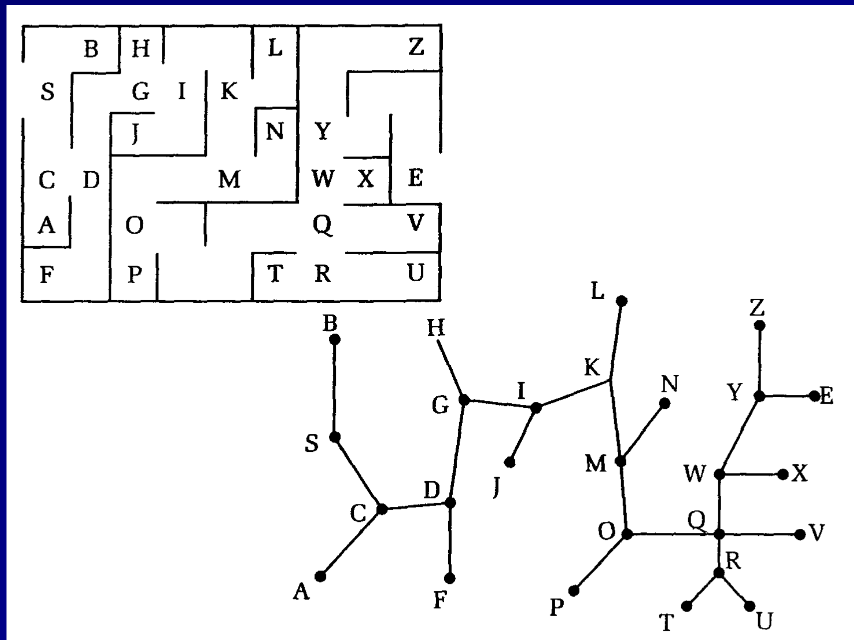
Ivana Kolingerová

Contents:

1. Maze generation
2. Cell automata

1. Maze generation

- Rectangular maze of RxC cells, random, but a path must exist
- Representation: a matrix or a graph (node ~ a cell)
- Cycles - possible but rare



- Representation in a program: e.g., to remember for each cell, whether its north and east wall is full

var north_wall, east_wall: **array**[0..num_rows,0..num_cols]
of boolean;

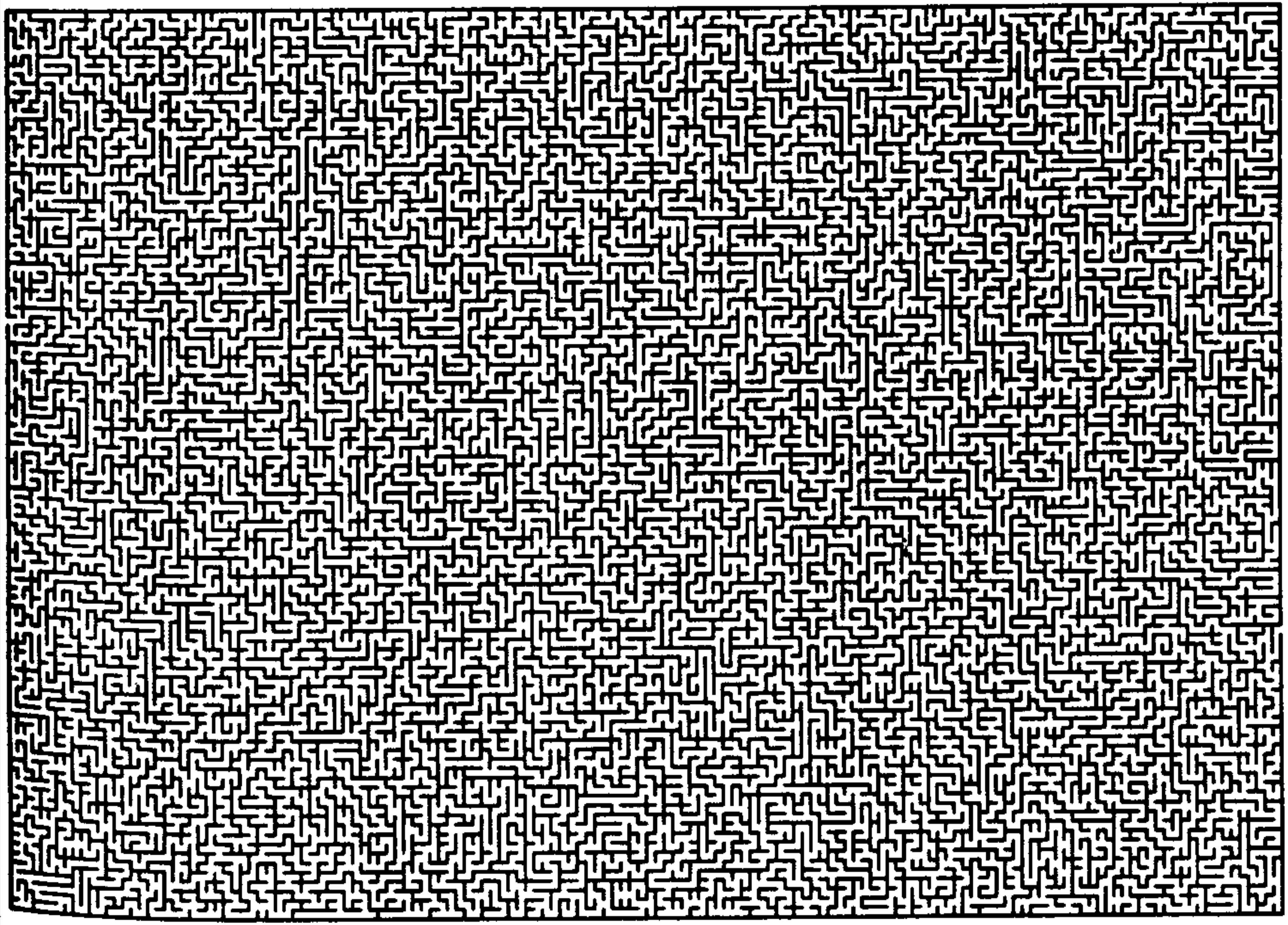
=> north_wall[i,j]=true <=> the cell i,j has the north wall "without a gate"

The row num_rows: a phantom row of cells below the maze, info about the lower edge of the maze

Analogously for the column num_cols – the right edge of the maze

Maze generation

- To start with all walls full, so there are no gates in the maze
- To move into a random cell, the task is to continue into the neighbouring cells:
  - Check 4 nb. cells, whether they have all walls intact. If not, the cell has been visited and is part of some path
  - Choose randomly one of non-visited nb. cells and remove the wall to it
  - Place the other non-visited
- nb. cells into a stack/queue
- The same is done after transfer into the newly joined cell.
- If all four nb. cells have been visited, a cell to continue is chosen from the stack/queue.
- An empty stack/queue means all cells have been visited



In the end, choose the star and target cells S, T
(usually on the edge of the maze)

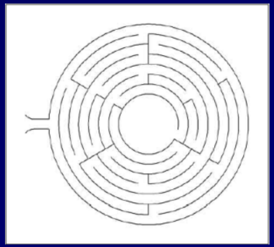
Warning: the new cell is suitable to connect
with already visited neighbor to avoid mutually isolated paths

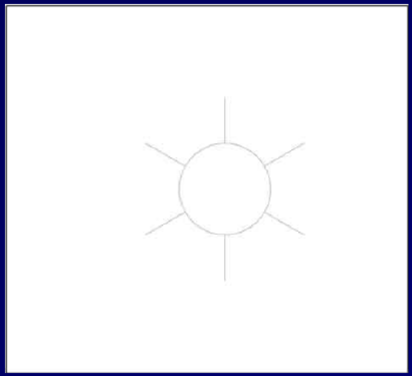
Variants – connect always x sometimes (then check the
existence of the path between S and T)
- sometimes do “some more connections" (~1-20 x)

Maze path finding
- backtracking – movement in a random direction with return
and marking the incorrect path
- instead of random direction we can go
“right/left all the time", then with S and T inside
we may loop

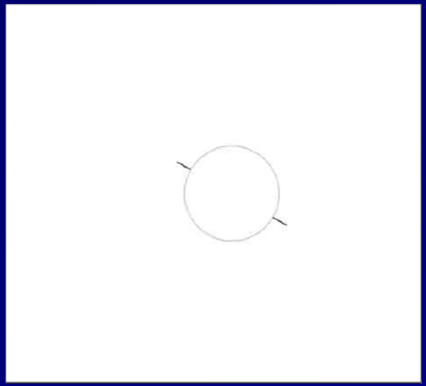
KPG1-7

Central maze



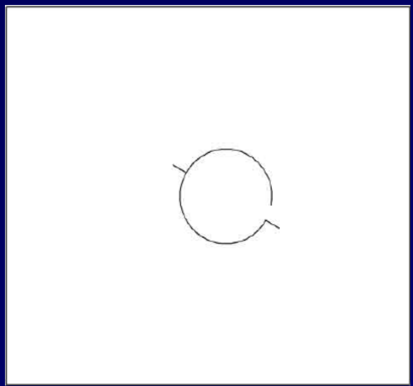


1. Choose a max. number of radial walls N per 1 layer, N≥2 (here N=6)

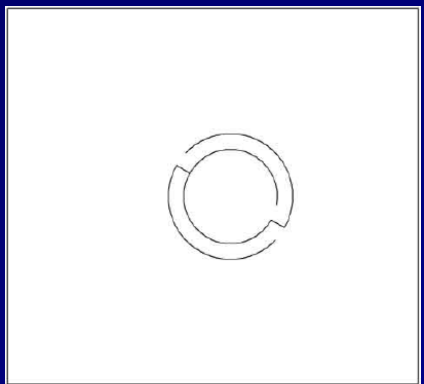


2. Choose randomly which of N radial walls will be in the first layer (at least 1) and draw them

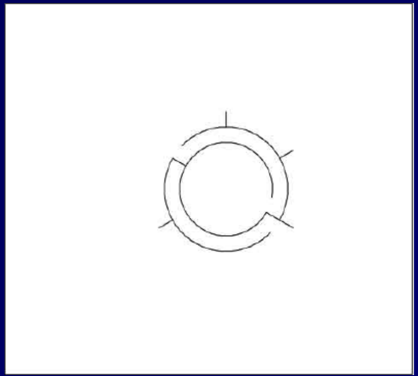
KPG1-8



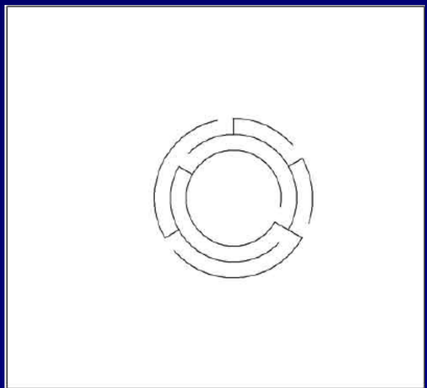
3. Choose randomly one of radial walls, draw a CW-oriented circle from the inner edge of the radial wall, end a bit earlier



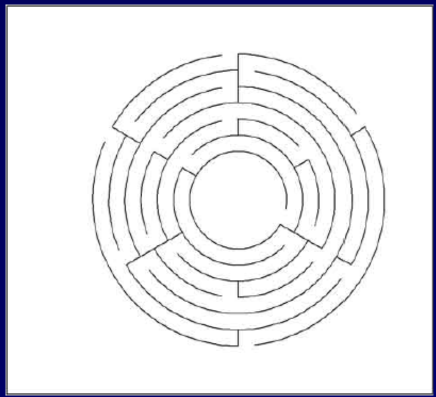
4. For all radial walls: draw a CCW-oriented arc in the direction to the following wall, end a bit earlier



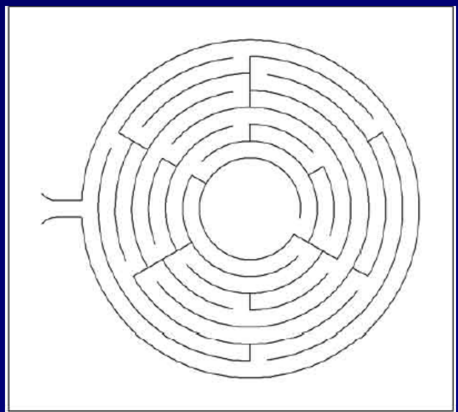
5. Choose randomly, which of N radial walls will be present in the next layer (at least 1), and draw them



6. Draw CW-oriented arcs from the outer ends of radial walls … etc.



7. Repeat the process of adding CW and CCW arcs until the size of maze corresponds to your idea

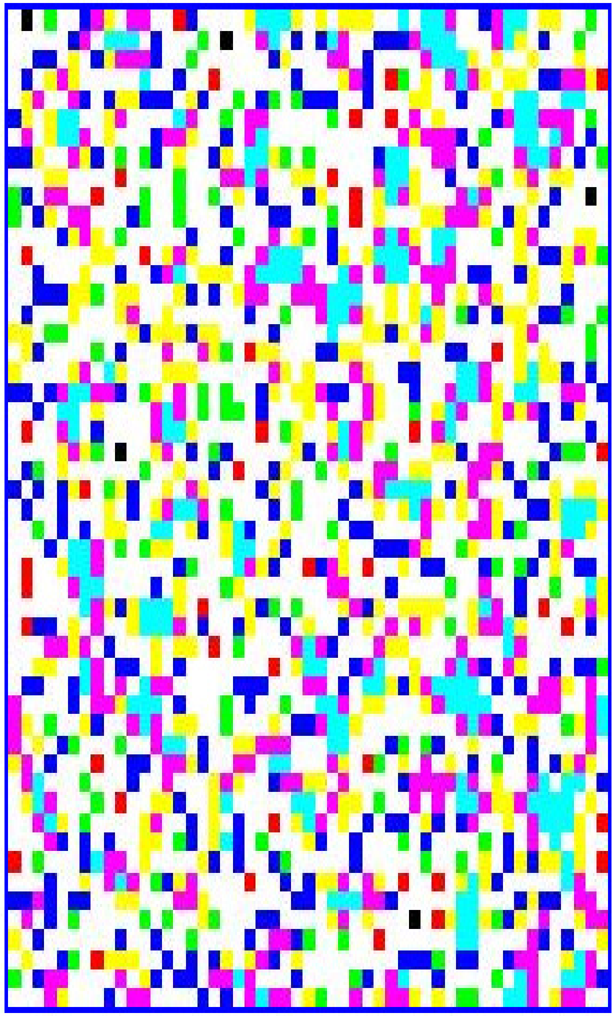


8. Add one more outer circular layer with an arbitrarily placed entrance

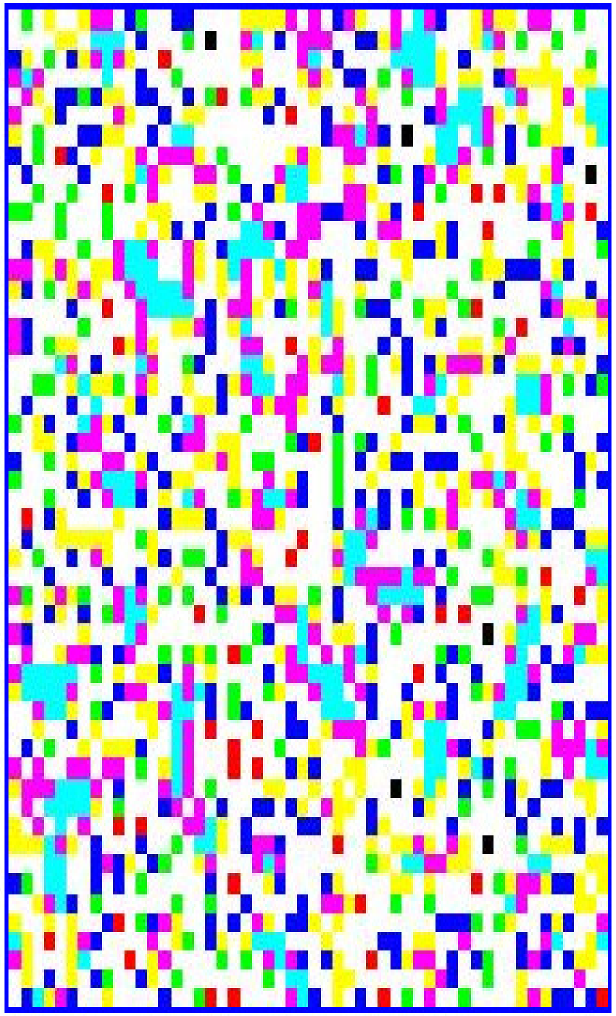
2. Cell automata

- Simulation of real life
- John Conway (1970)
- Game of Life – 2D binary cell automaton

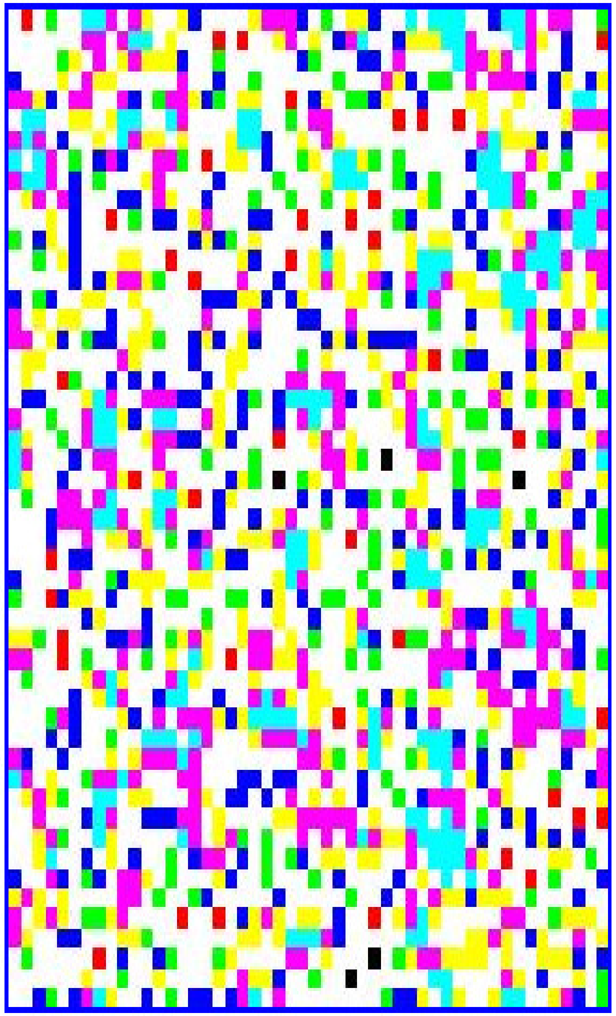
- Each generation – an infinite matrix of cells where each cell is alive or dead
- A cell in time t -> the cell in time t+1, changed or not
- according to the status of the cell and its 8-neighbourhood



Random configuration I; 400 step animation



Random configuration II; 600 step animation

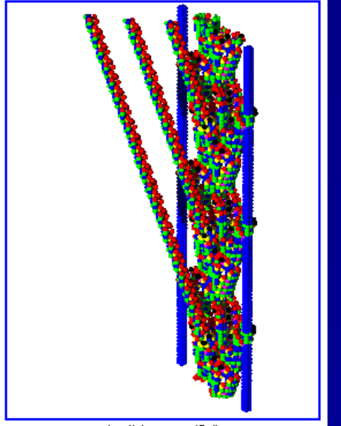


Random configuration III; 1000 step animation

Typical rules:

- The cell alive in time t is alive in time t+1 <=> it has two or living neighbours in time t
- The cell dead in time t is alive in time t+1 <=> it has three living neighbours in time t
- Other cells are dying or stay dead

Preview – by animation or by placing consequent generations in columns in a 3D picture



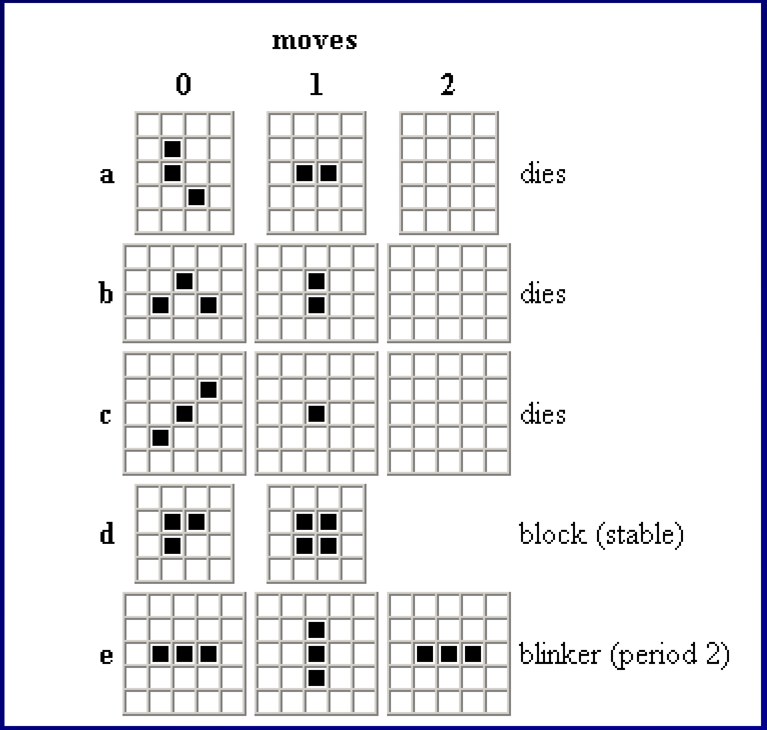
A glider gun (3d)

Cells can be also coloured according to the number of their living neighbours or their age

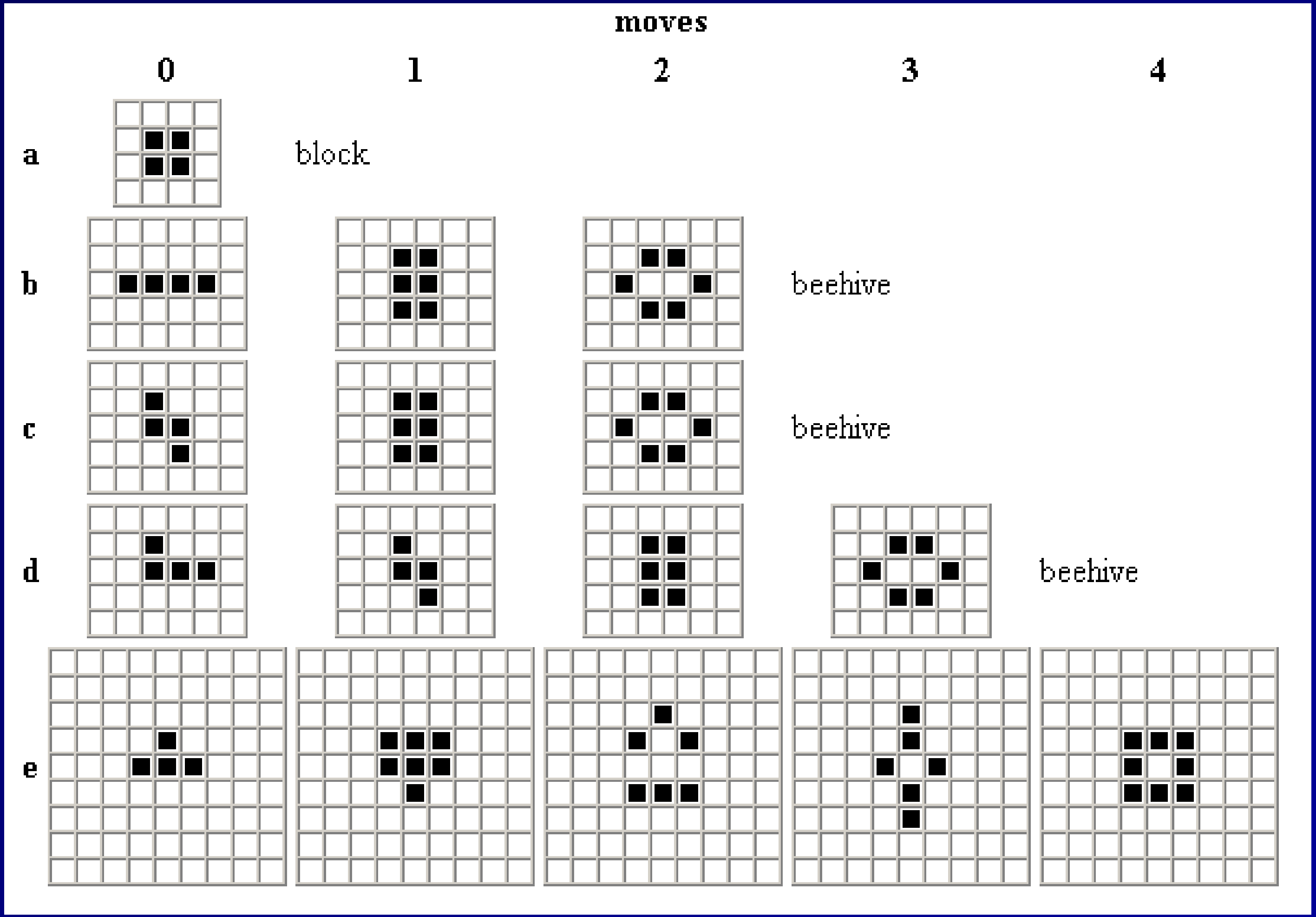
All births and deaths have to be solved simultaneously – input and output generations/matrices must be separated!

Implementation trick: to use matrices with an extra row and column on each size to simplify the test in the first and last rows and columns, respectively

Basic configurations:



Basic configurations:



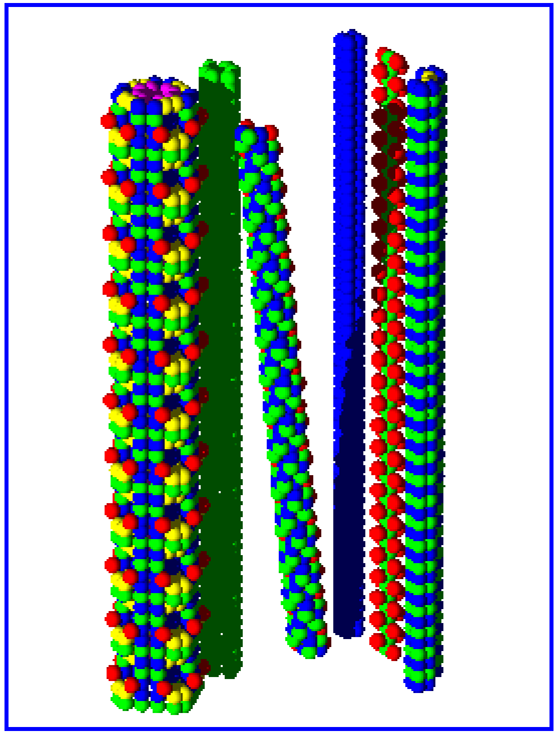
Basic configurations:

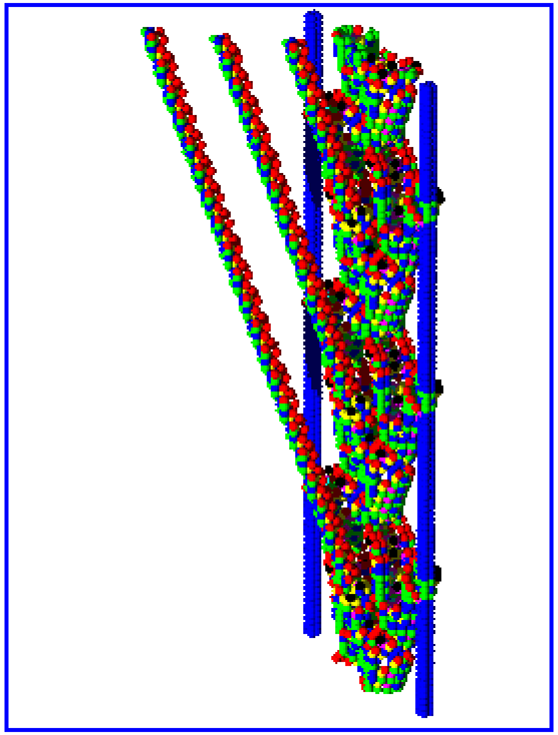
Table 1. The long-term behavior of the Game of Life when the initial configuration is a filled square with sides of length n.

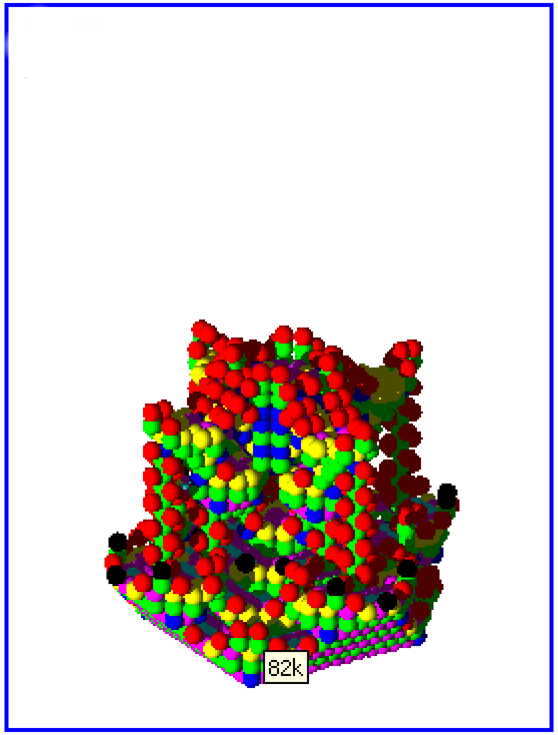
$n=1$	dies immediately
$n=2$	is stable from $t=0$ (it is a block)
$n=3$	dies at $t=9$
$n=4$	dies at $t=4$
$n=5$	terminates in the honey farm at $t=11$
$n=6$	forms a pond at $t=5$
$n=7$	makes a traffic light at $t=11$
$n=8$	dies at $t=6$
$n=9$	turns into 4 traffic lights and 1 tub at $t=17$
$n=10$	forms 1 pond at $t=17$
$n=11$	becomes a traffic light at $t=33$
$n=12$	fades at $t=9$
$n=13$	makes a tub at $t=18$
$n=14$	turns into a pond at $t=9$
$n=15$	dies at $t=22$
$n=16$	forms 4 ponds and 4 beehives at $t=11$
$n=17$	makes 1 tub, 8 blocks, and 12 blinkers at $t=33$
$n=18$	turns into 8 blocks, and 1 pond at $t=17$
$n=19$	makes 5 traffic lights at $t=21$
$n=20$	dies at $t=12$

KPG1-19

Some 3D pictures:


Some periodic patterns and a glider (3d)

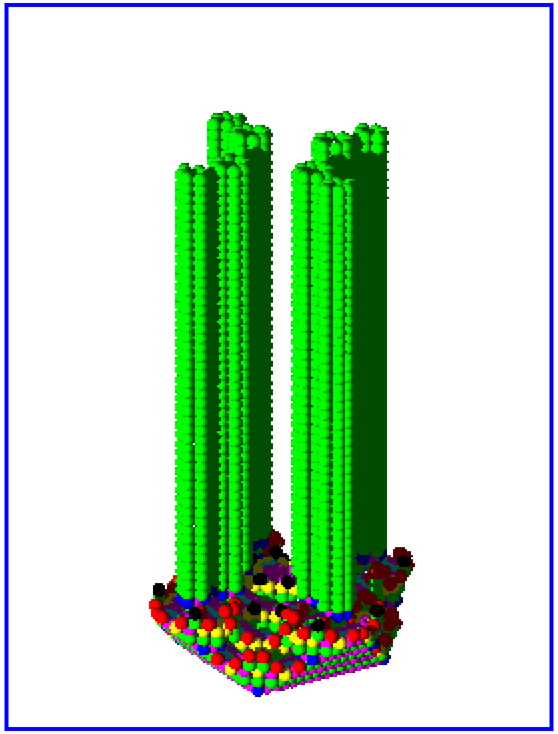

A glider gun (3d)



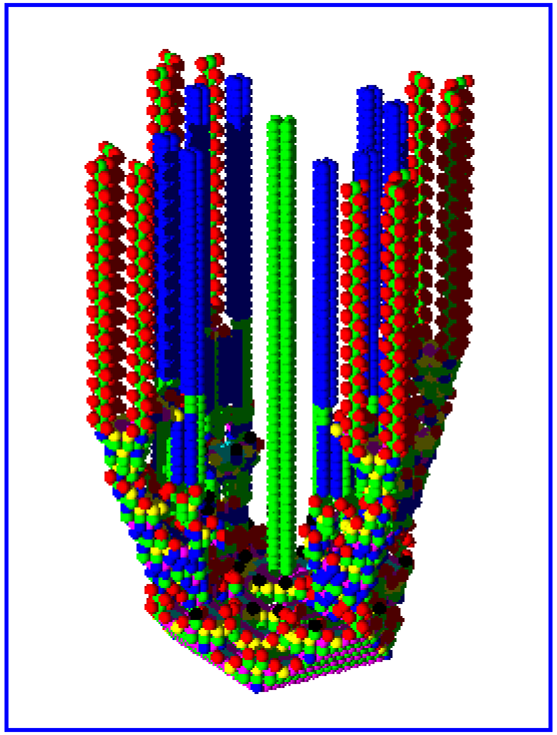
A 15 by 15 square (3d)

KPG1-20

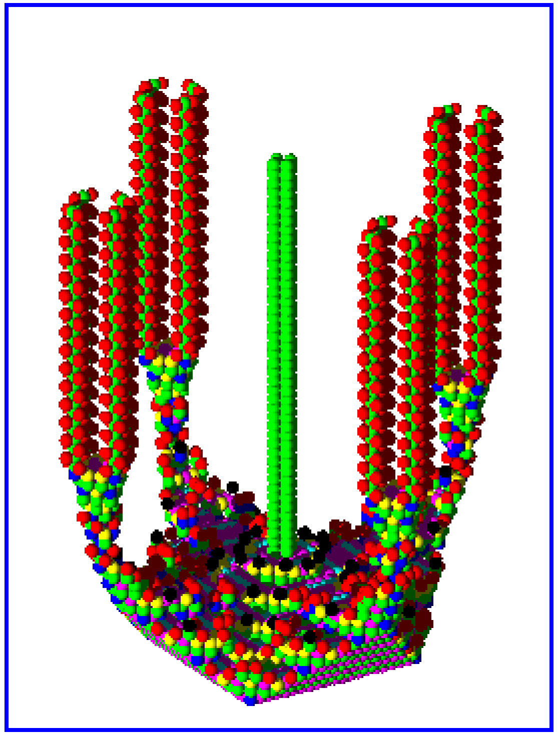
Some 3D pictures:



A 16 by 16 square (3d)

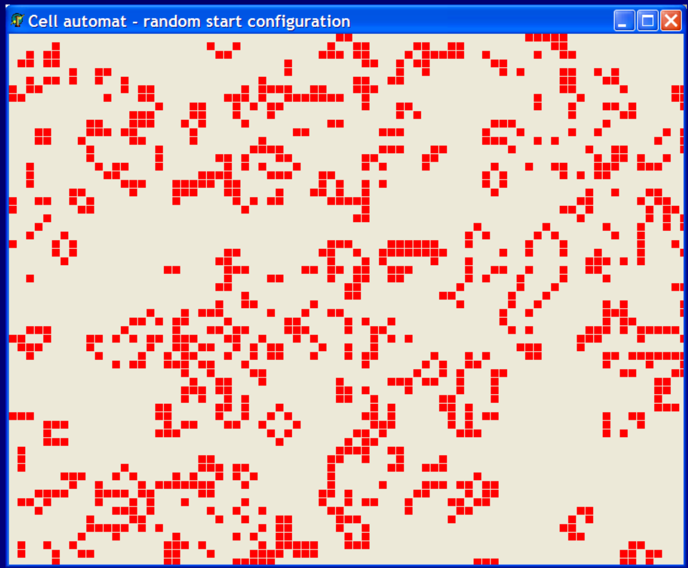


A 17 by 17 square (3d)



A 21 by 21 square (3d)

It is also interesting to start with a random initial configuration

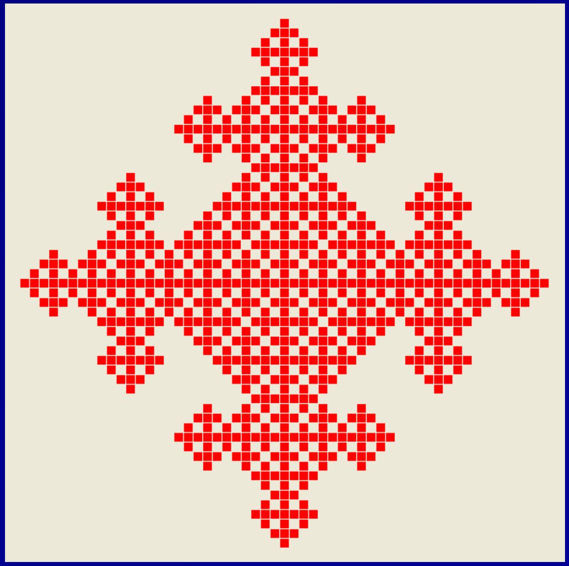


Modifications:

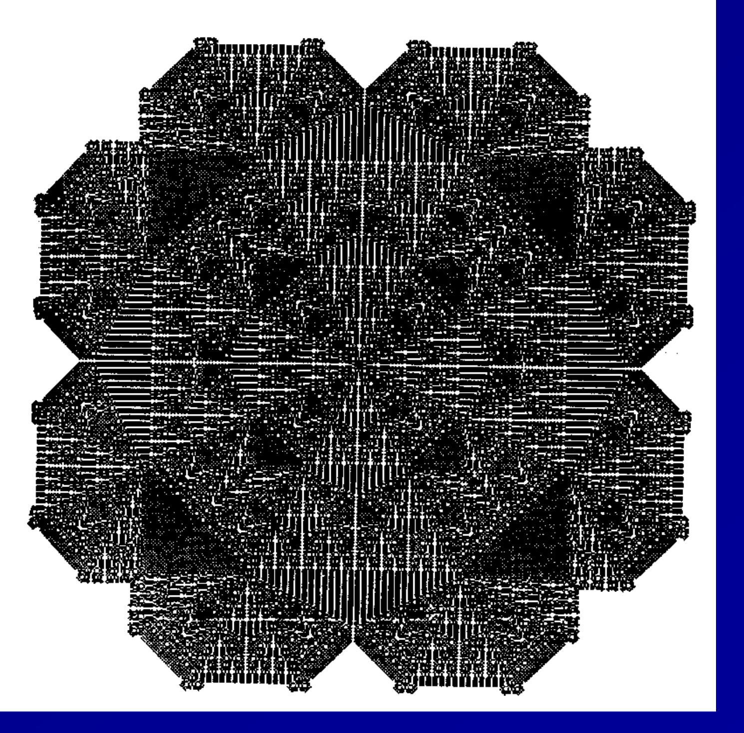
- Behaviour based on one defect and a rules modification

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Type 1: 1 appears if the cell has one orthogonal neighbor of value 1, no death



Type 2: 1 appears, if the cell neighbours orthogonally with one neighbor of value 1 for even t or with neighbor of value 1 orthogonally and vertically for odd t, no death

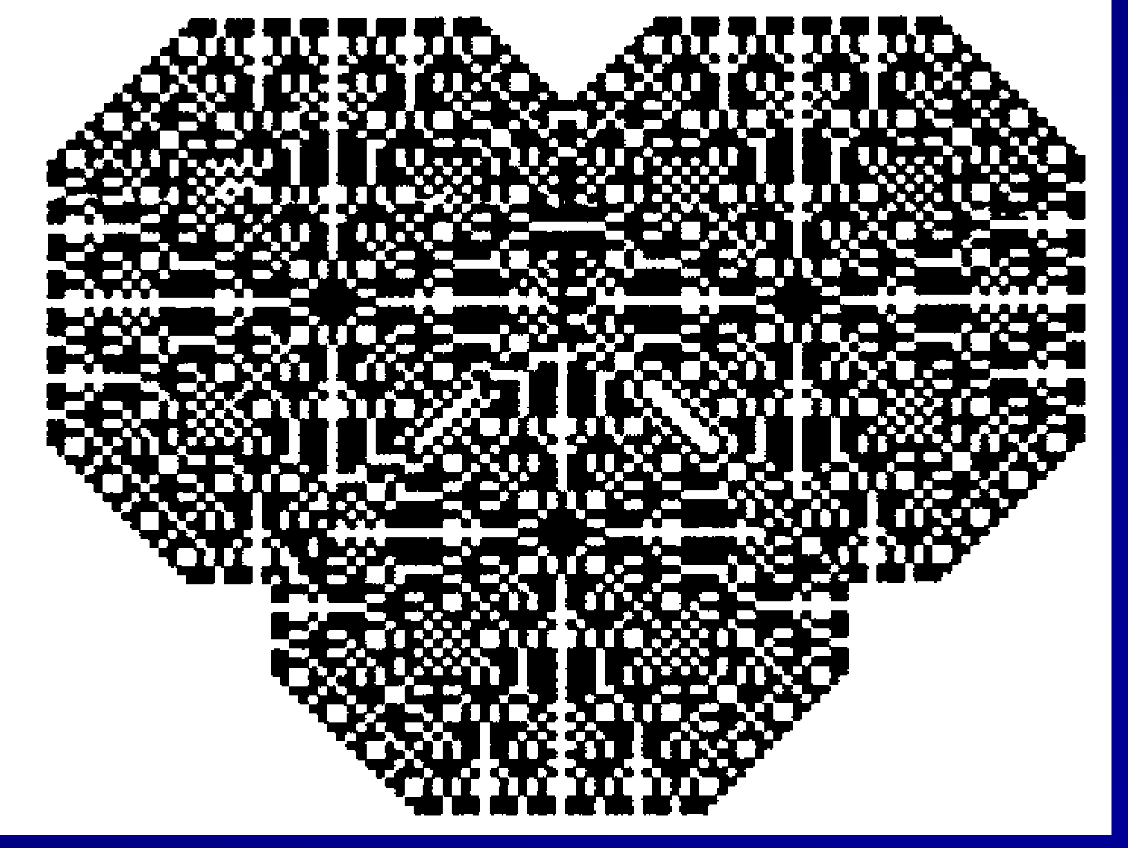


Type 2

Type 3: more defects, they grow, then compete (+cell death)

Type 4: bigger neighbourhood

.... etc.



Typ 3e (3 defects)

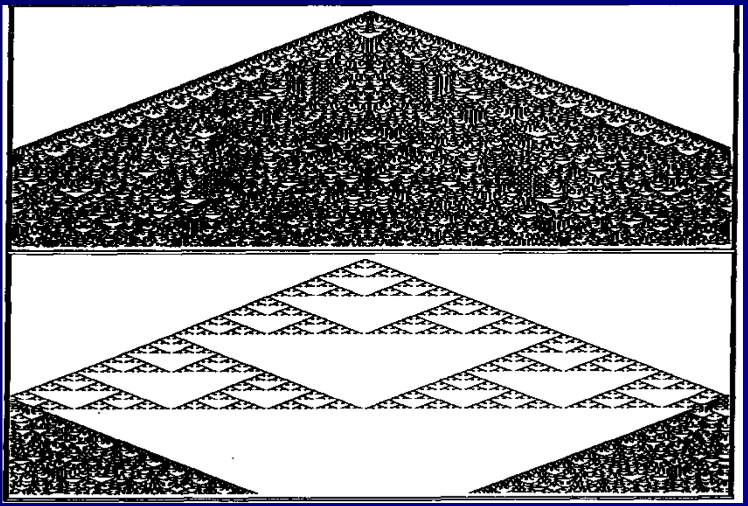
One-dimensional cell automata

- The cell and 2 neighbours on each side
- A rule is written as: (d0,d1,d2,d3,d4;a0,a1,a2,a3,a4)

  0 in di – if the central cell is dead
  and i cells from 4 neighbours are alive,
  the central cell will die

  1 in di – if the central cell is dead
  and i cells from 4 neighbours are alive,
  the central cell will be reborn

  ai – the same for the central cell alive

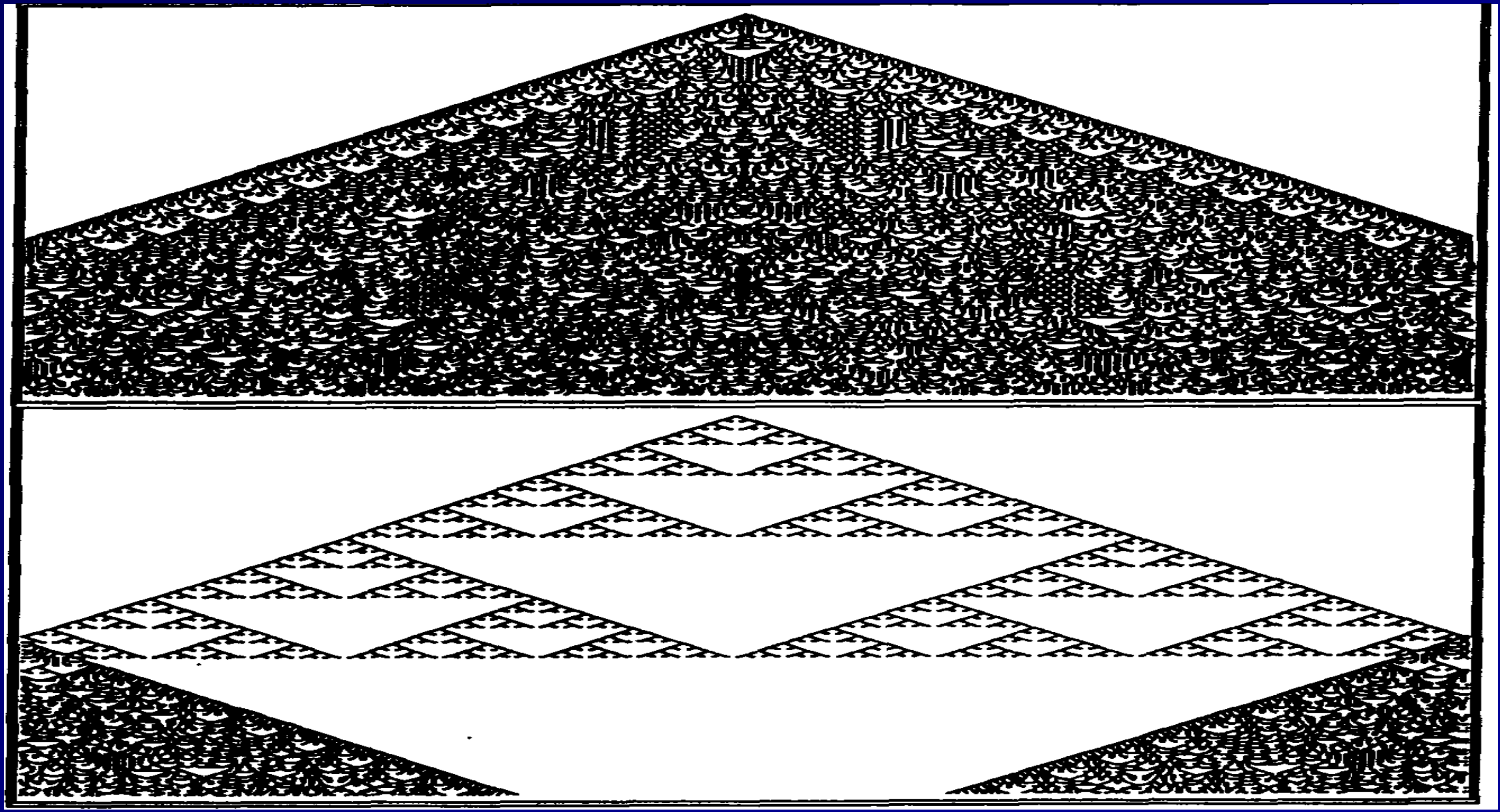


Ex.:(0,1,1,0,0,;0,0,1,0,0) – an alive cell in the next generation, if the central cell is dead
and (1-2) neighbours are alive or
the central cell is alive
and 2 neighbours are alive

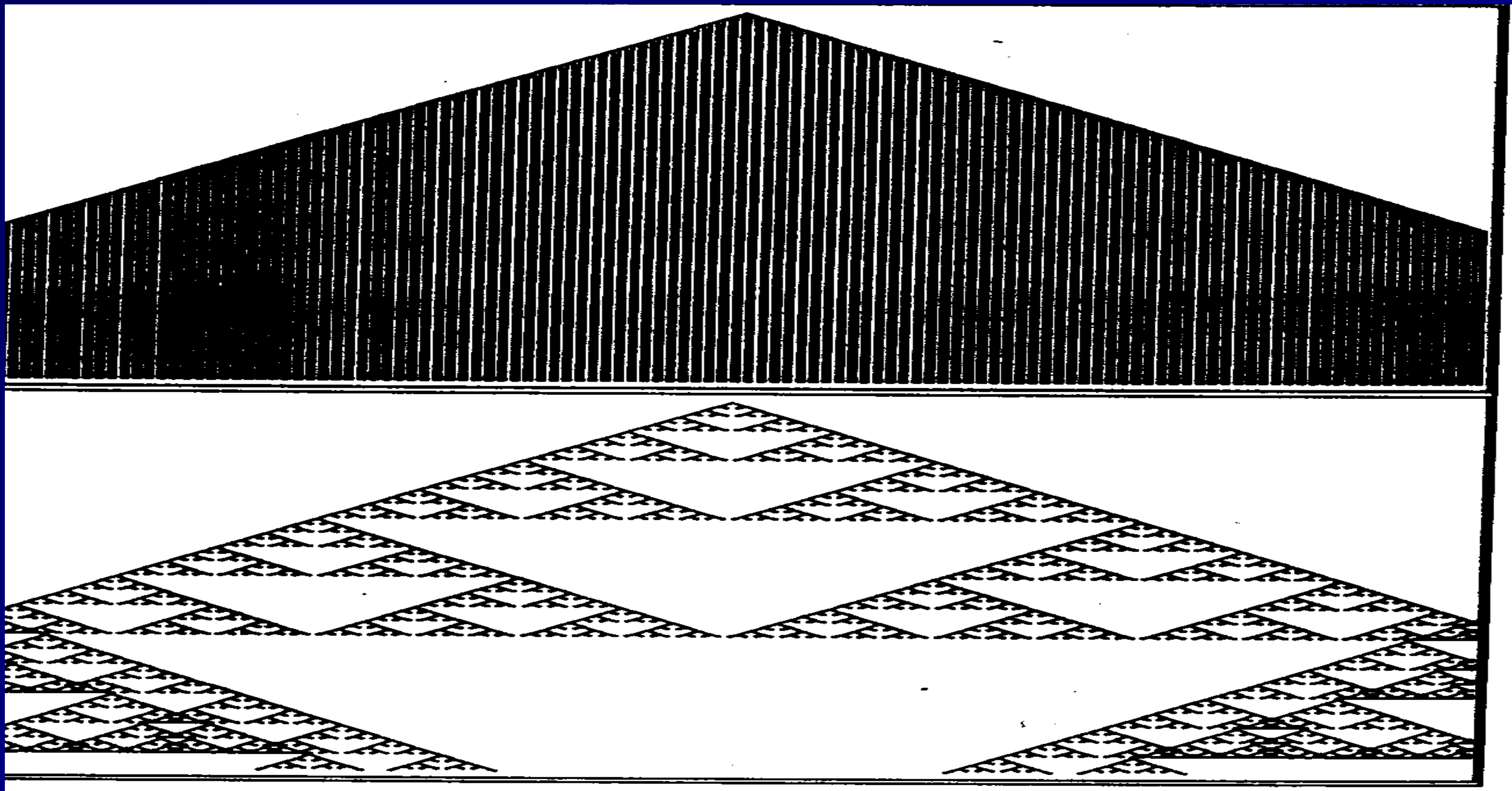
KPG1-26

Ex.: (0,1,1,0,0;0,0,1,0,0)

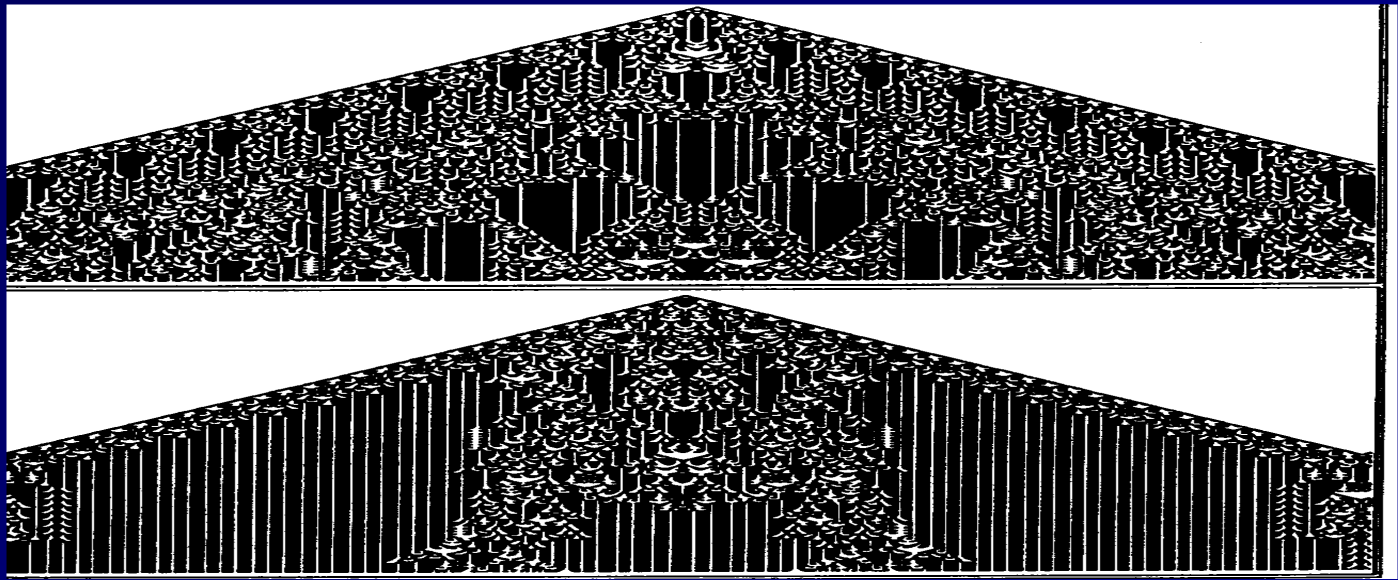
2 nb. Alive cells produce Sierpinski triangle,
from one cell “patterned triangle"
solved as "wrap-around"

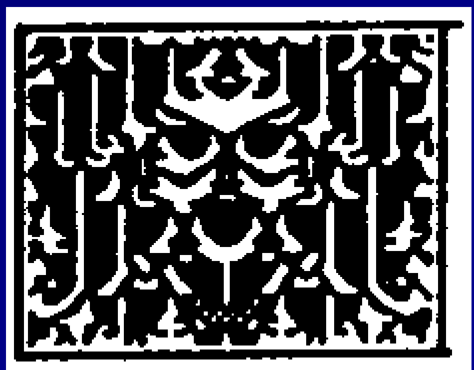


Ex.: (0,1,1,0,0;0,0,1,1,0) – from 1 cell strips, from 2 Seirpinski triangle



Ex.: (0,1,1,0,0;0,1,0,1,1) – from 3 neighbouring cells–
a “skull” appears





References:

Maze:

Francis S.Hill Jr.: Computer Graphics, Macmillan Publishing Company, New York, 1990

Cell automata:

- Martin Gardner: Mathematical Games. The Fantastic Combinations of J. Conway's New Solitaire Game "Life", Scientific American 223 (October 1970), pp.120-123
- John.E.Pulsifer, Clifford A. Reiter: One Tub, Eight Blocks, Twelve Blinkers and Other Views of Life, Computer s and Graphics, Vol. 20, No. 3, pp.457-462, 1996, Elsevier Science

- Elwyn R. Berlekamp,John H. Conway, Richard K. Guy: Winning Ways for Your Mathematical Plays, Academic Press, 1982

- Michael Frame: Sensitivity in Cellular Automata: Some Examples, Computers and Graphics, Vol.18, No. 5, pp.733-737, Elsevier Science

- C.A.Pickover: Mathematics and Beauty – VIII, Tesselation Automata Derived from a Single Defect, Computer Math Applic., Vol. 17, No. 1-3, pp.321-326, 1989, Pergamon Press