

Manual: GeZA-2D for Android

GeZA-2D is a Tool for exploring 2-dimensional cellular Automata and displays them in attractive graphics.

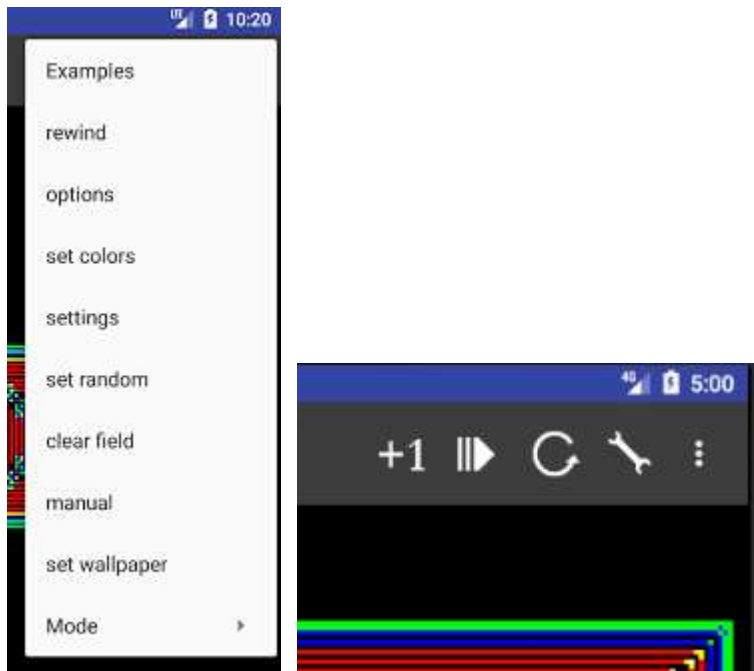
Table of contents

User interface.....	2
General.....	2
Rule editor.....	4
Rule converter.....	5
Set colours.....	6
Settings.....	7
save file.....	8
load file.....	9
create gif.....	10
Fundamentals.....	11
Standard.....	11
Hexagonal.....	13
WaTor (Water Torus).....	13
Brian's Brain.....	14
HPP Gas.....	14
Margolus (block cell automaton).....	15
Zhabotinsky.....	16
Options.....	17
Standard.....	17
Hexagonal.....	18
WaTor.....	19
Brian's Brain.....	19
Margolus.....	19
Zhabotinsky.....	21
Credits.....	22
Links.....	23

User interface

General

On starting GeZA-2D the standard automaton is running or the last state is displayed. You can draw on the main window, which means giving cells a new value. More functions of this app you'll find in the drop down menu and in the title bar.

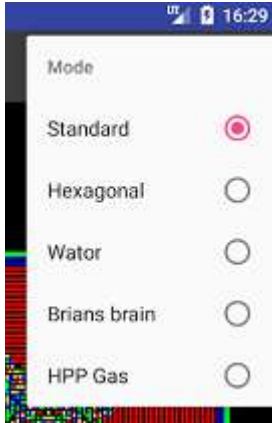


Depending on the physical resolution not all of the 4 icons in the title bar are displayed, these function are choosable via drop down menu then.

menu item	icon	function
1 step	+1	computes just one next generation.
start/stop	▶	starts resp. stopps animation.
rewind	↺	rewind to the starting cell states (running animation will be stopped).
options	🔧	opens a dialog for settings depending on the context.
set colours	-	opens a dialog for changing colours.
settings	-	opens a dialog for changing global settings
set random	-	sets the field to a random initial state
clear field	-	sets all cells to state 0.

manual	-	opens this manual
set wallpaper	-	the current image may be set as wallpaper
Mode	-	opens submenu for selecting supported basic type of automata

The sub menu "Mode":



Here you may set the basis mode of the cellular automaton.

The field in this app is a closed one, which means the left border goes to the right and the bottom goes to the top border and vice versa.

This is called in mathematics a toroid. You can imagine it as a bike tube.

Rule editor

Touching on the rule string in the option dialog, this rule editor will be opened.



On the top you see the rule string. Touching on this string sets the position of the cursor. The number in red among them shows the current cursor/string position. This is useful, because the character at this position dictates the value of the cell in the next generation, if the sum of value of the neighbor cells is equal to the position number.

Touch the button "use it" and provides the new rule to the option dialog and returns.

controlling:

button	function
<---	cursor just 1 position to the left.
--->	cursor just 1 position to the right.
Hilfe	opens this help side.
< del	deletes 1 character left from the cursor.
del >	deletes 1 character right from the cursor.


function of the rule characters:

character	function
=	the cell keeps the state
+	increments state of the cell by 1, 9 goes to 0.
-	decrements state of the cell by 1, 0 stays 0.
!	0 goes to 1, otherwise the cell goes to 0.
?	the cell gets a random state (0..9).
>	the cell gets the highest state from the neighbours.
<	the cell gets the lowest state from the neighbours.

Remark: There is a problem with rules longer than display width. I'm working on it.

Rule converter

Sometimes you find in literature or in programs (mostly in "Game of Life" themes) rules in notation like 23/3 or S23/B3 or similar. S means stable and B Birth. This says for the Moore-

neighbourhood , that for the next generation the cell with 2 living (state = 1) neighbours is stable and with 3 living neighbours the cell is still living or born. This is the rule of the "Game of Life".

But GeZA2D is using a new Kind of ruling convention. For converting rules from the old notation into the new GeZA like, this little tool helps you.

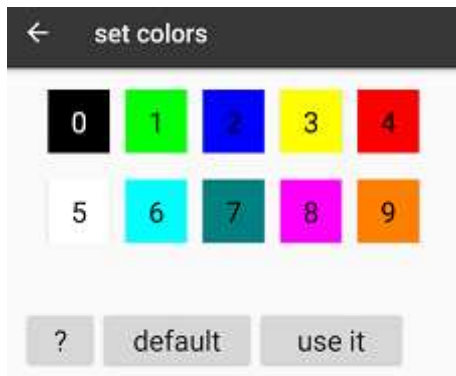


Click on the buttons (0..8) right from the S (stable) or B (Birth) to activate or deactivate them. After "old Style" you'll find the rule in conventional notation and beneath "GeZA rule" the notation using this app.

Clicking on the button "use it" will go back to the option dialog with the new defined rule. If the current rule is not valid for 'old style'-notation, the 'game of life' rule is preset!

Set colours

This shows the current selected colours for the different states (0-9) of the cells.



Clicking on such coloured cell will open a colour picking window, which assigns the new colour to the selected cell state.

To activate the new colours, click button "use this". The program will then show the main view screen with new colours.

For returning to the default colour palette, click the button "default".

Settings

You do here global settings, which are independent from the selected mode.



parameter	function
size of cell	size of the cell on display. value is between 2 and 7. 2 means a cell gets 2x2 pixel, 7 means 25x25 pixel. The smaller one cell, the larger the amount of cells to compute. This may slow down the programm dramatically.
Speed	Speed of computing generations.. value is between 1 (very slow) and 8 (top speed) .
max. random	for setting cell state by random, the cell gets a value between 1 and max. random. 1 up to 9 are valid.
edit colour	By drawing on the screen the new cells/pixel gets this value, respectively the colour matching this value. 0 up to 9 are valid.

save file

In this item you may write your current CA into a file (GeZA type and/or picture(PNG))



directory:
/storage/emulated/0/GeZA2D

file (png):
geza_2021-04-05_140939.png

file (GeZA-2D):
geza_2021-04-05_140939.gz2

help save

On top is the directory shown, which is the destination of file operation here. Changing the directory isn't possible yet.

You may save two types of files:

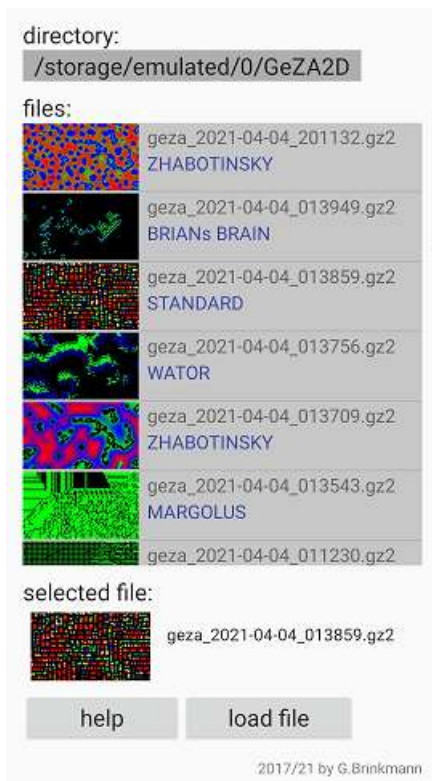
- 1) a graphic of the current screen by type of PNG.
- 2) the current CA with all parameters and settings into a file with extension "gz2". This file is readable by the GeZA2D app only, but of course by other android devices also.

Activated checkboxes says which types of files will be written.

The filenames are the combination of "geza_", Timestamp and suffix.

load file

In this you open files, which you have written with GeZA before.



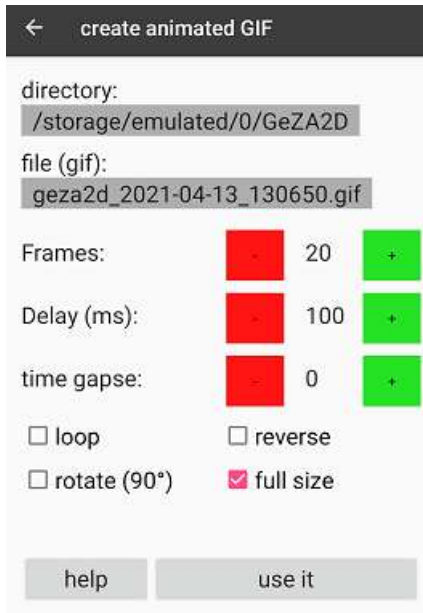
The app searches in a predefined directory for "GeZA-2D files". This directory ("GeZA2D" on the internal memory) is not changeable yet. The complete name of this directory is shown in the first line.

Below you see the list of the files found (snapshot plus name). Clicking on one of this entries will show you the related informations under "selected files" again. Clicking the button "load file" the related automat will be startet.

Double clicking on an entry will start the automat immediately.

create gif

In this item you may create an animated gif from the current CA!



The description of parameter and options:

parameter/ option	description	possible values
frames	count of pics in animation	2 - 200
delay	delay between 2 images in ms	10 - 2000
time gapse	after every picture this amount of pictures are skipped.	0 -20
loop	after the standard sequence the same sequence is added, but reverse in time	true/false
reverse	the animation reverse in time	true/false
rotate (90°)	rotates the animation 90° to the left.	true/false
full size	full screen size or half of height and width.	true/false

Below some examples as result of setting parameters and options:



full_size



small



reverse



rotate 90°



loop



delay =300

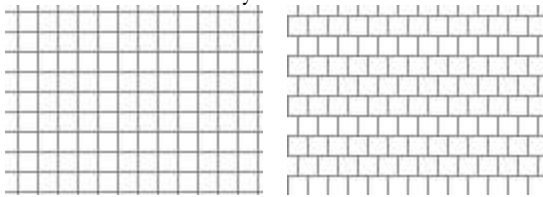
Fundamentals

Standard

A cellular automaton consists by a lot of cells, every one have a defined, mostly discrete state (starting configuration). A rule, which considered the state of the cell itself and the states of an amount of neighbour cells determined the state in the next generation. The cells may be elements of a line (1-dimensional), a plane surface (2-dimensional) or of higher dimensions too..

One of the best known cellular automaton is the "Game of Life", discovered by John Conway in 1970. The cells are positioned on a field (2-dimensional).

GeZa-2d treats only 2-dimensional cellular automata. The cells are positioned on a grid regular.



The left picture shows a chessboard arrangement and is used here for mode standard, WaTor, brians brain and HPP.

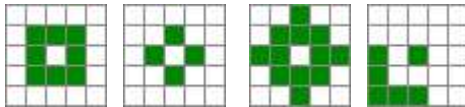
The right picture shows a shifted grid and is logical the same as a honeycomb. It is the base of the mode hexagonal.

There are much more tilings thinkable. In the following description only the standard mode is treated.

The single cell on the field has a discrete state, in this app a value between 0 and 9.

Cells with these different values are displayed with different colours (0 = black, 1 = green, 2 = blue, etc.) You may change this colour scheme..

Furthermore there have to be defined a neighbourhood:



On the left picture you see, that the cell gets 8 neighbours (green). This is known under the name Moore neighbourhood and is used by the mentioned Game of Life.

The second is called von Neumann neighbourhood. The cell itself can be part of the neighbourhood.

In this app you can define the neighbourhood freely.

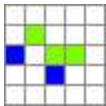
At last we have to define a transformation regulation (rule), which determines the changing of the cell values by transition to the next generation.

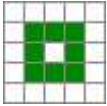
This app is using in standard and hexagonal mode socalled "totalistical" rules. This means all values of the neighbours are added to a sum. This sum and the rule table determine the new value of the cell.

Let's do an example:

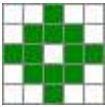
sum values of neighbors	0	1	2	3	4	5	6	7	8	9	...
new value	0	2	1	4	3	+	=	0	0	0	...

This rule written in a short form : 02143+=

starting state:  (0=white, 1=green, 2= blue)

neighbourhood: 

The neighbourhood of the middle green cell consisting of 2 green, 1 blue and 5 white cells. The sum of the neighbour cells ($5*0 + 2*1 + 1*2$) is 4. In the rule table you find for the sum of 4 the value 3. This value 3 is therefore the value of the cell in the next generation.

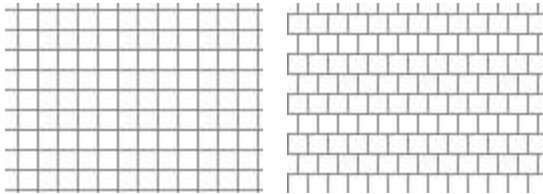
 the neighbourhood, so gets the same cell 8 white, 2 green and 2 blue neighbour cells, the sum is therefore 6. In the rule table we find for sum 6 the equals sign =, that means the objected cell will not change the value in the next generation and stays green.

In this table you see which functional characters exist except the digits.:

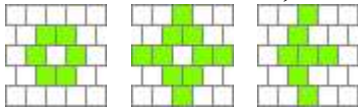
character	function
=	The cell keeps the value.
+	Increments the value of the cell by 1, 9 goes to zero.
-	Decrements the value of the cell by 1, 0 stays 0..
!	0 goes to 1, otherwise the value of the cell gets 0.
?	The cell gets a random value
>	The cell gets the largest value of a neighbour.
<	The cell gets the smallest value of a neighbour.

Hexagonal

The only difference to the standard mode is, that the field is chequered but have an offset structure (logical like a honeycomb).



This results to the fact, that the neighbourhood is different too:



Everything else is just like the [standard mode](#).

WaTor (Water Torus)

WaTor is Water Torus. Water represents here the sea and torus is a closed area like a bike tube for instance.

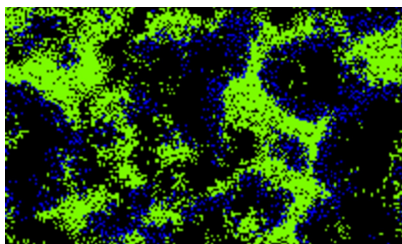
In the sea WaTor are living fish and sharks. The fish are multiplying on certain rules, the sharks eating fish for living and multiplying.

Low shark population leads to increasing fish population strongly. Sharks will multiply itself and destroying their live base, the fish population.

The low population of fish leads to hungry and dying sharks. And so on.

WaTor shows you very nice the development of a predator-prey-relation. Also known as the

WaTor is Water Torus. Water represents here the sea and torus is a closed area like a bike [Lotka-Volterra-equations](#).



For more information see the recommended [Wikipedia article](#).

Brian's Brain

This special automaton was discovered by Brian Silverman.
 For more information see the recommended [Wikipedia article](#).

HPP Gas

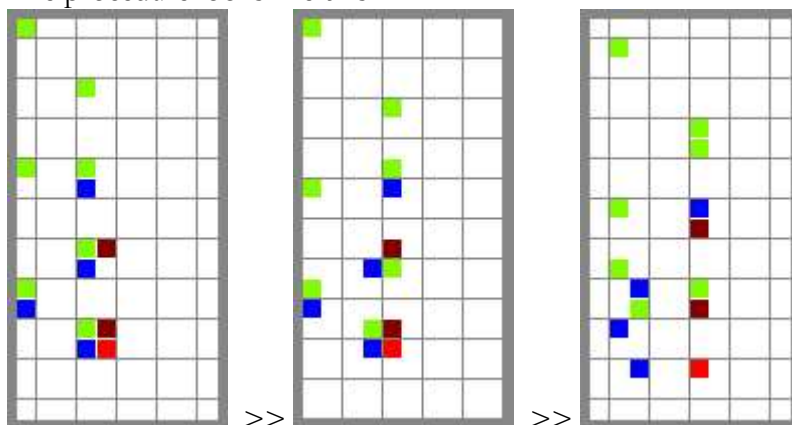
This automaton does not use the known neighbourhood, but brackets 4 cells together (near the border 1 and 2 cells too).

The following table shows the transition rules. Per rotation you get all possibilities.

before	after

In the next generation the bracket together is shifted by 1 row and 1 column.

The procedure looks like this:



This process is called Margolus neighbourhood.

More information you'll find in the related Wikipedia article [lattice gas automaton](#).

Margolus (block cell automaton)

For more information see the recommended Wikipedia-article and sites from MirekWojtowicz resp. Tim Tyler:

https://en.wikipedia.org/wiki/Block_cellular_automaton

http://www.mirekw.com/ca/rullex_marg.html

<http://cell-auto.com/neighbourhood/margolus/index.html>

In this mode a cell takes only the state 0 or 1. The area is a closed toroid and has even number of rows and columns.

Zhabotinsky

The name of this CA is based on the [Belousov-Zhabotinsky-reaction](#), a well known chemical oscillator.

To simulate this chemical reaction i used the "Misch-Masch-Maschine" by Martin Gerhardt and Heike Schuster:

cells may have states of 0 (healthy), 1 to 255 (infected) and 256 (ill).

A cell of state 0 (healthy) will get the state $\lfloor K/k1 \rfloor + \lfloor I/k2 \rfloor$ in the next generation.

K =sum of ill neighbors, I =sum of infected neighbors,
 $k1$ = illness threshold (1-5), $k2$ = infect threshold,
the square brackets rounding off to integer.

Infected cells (state 1-255) go to $\lfloor S/A \rfloor + g$, but max 256 in the next generation.

S = sum of the values from all neighbors and the cell itself. A is the number of these cells. $\lfloor S/A \rfloor$ is just the mean value.

g = constant (1-120) allways added, it is a kind of stimulation.

An ill cell (state 256) will be healthy (state 0) in the new generation.



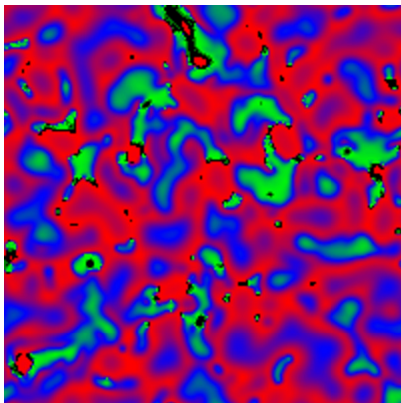
This automaton is using Moore neighborhood ($r=1$) and running on a closed Area (Toroid).

This process may be described in this words:

- healthy cells will be infected by infected and ill neighbors.
- infected cells are powered by a mechanism of stimulation and diffusion until the cell is ill.
- "ill" cells will be healthy immediately and the process starts again.

There are generated a lot of pattern, depending of the parameter $k1$, $k2$, and g , but of the starting states also.

Below you see 2 typical examples:



$k1=3, k2=2, g=5$



$k1=3, k2=2, g=64$

Options

Standard

You may set the determining properties of standard cellular Automata in this dialog. On the one hand the transformation rule and on the other hand the neighbourhood.



Clicking on the rule string opens the rule editor.

You may toggle the Button 0 to 24 between grey and green. The green one are defining the current neighbourhood.

The button "rule converter" opens a dialog, where it is possible to convert classical rules into GeZA style rules.

Hexagonal

You may set the determining properties of hexagonal cellular Automata in this dialog. On the one hand the transformation rule and on the other hand the neighbourhood.

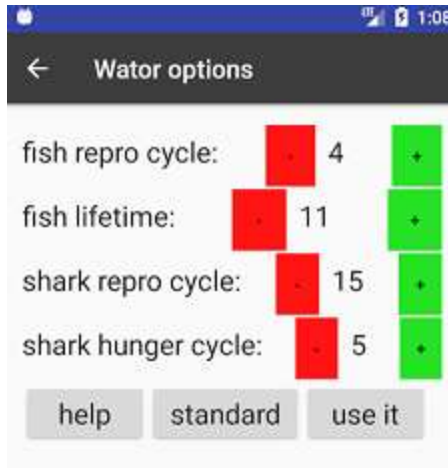


Clicking on the rule string opens the rule editor.

You may toggle the Button 0 to 18 between grey and green. The green one are defining the current neighbourhood.

WaTor

In this mode we have 4 parameters you may change.



parameter	meaning
fish repro cycle	number of generations after a fish reproduce itself.
fish lifetime	a fish dies after this number of generations.
shark repro cycle	number of generations after a shark reproduce itself.
shark hunger cycle	doesn't eat a shark as long as this number of generations, he dies.

Remark: a greater fish repro cycle than the fish lifetime is nonsense, because the fish dies before reproduction.

Brian's Brain

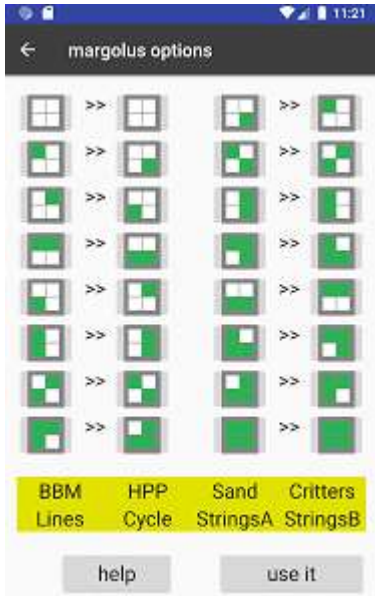
There is only one parameter for changing.



"Brian Max" is the value, which gets a cell having 2 neighbours with value "Brain Max". 2 is default.

Margolus

Here you may define the rules for the Margolus automaton.
 In this type a block of 4 cells will be taken and the rules sets the states of this cells in the next generation.
 One block may have 16 different states. In this window you'll see the current rule for these 16 states.

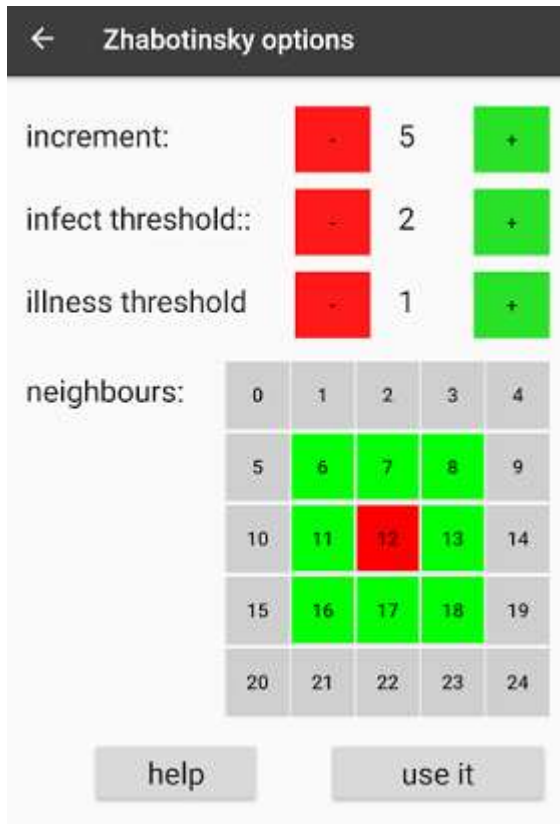


Touching one of this assignments opens a new window.



Choose here the new state for the selected block and you go back.
 There are 8 buttons also for setting a predefined rule.

Zhabotinsky



"increment" is the constant (g) which is added to infected cells in every generation.

"infect threshold" is the minimum of infected neighbors to infect a cell.

"illness threshold" is the minimum of ill neighbors to infect a cell.

The value of "increment" is between 1 and 120..

Values of "infect threshold" and "illness threshold" are between 1 and 5.

Here the neighborhood may be defined also. Green means the related cell will be used by the algorithm.

For more Information about the algorithm Click at ["fundamentals Zhabotinsky"](#).

Credits

GeZA-2D - App für Android



entwickelt 2016/17 von G. Brinkmann

Webside: www.zellauto.de

E-Mail: geza2d@zellauto.de

Acknowledgement:



Without supporting, understanding, pushing by my wife Brigitte, GeZA-2D was not thinkable.

IHDL

WIKIPEDIA

Thanx to the great Wikipedia project!

Literature:

Gerd Brinkmann

Das Software-Experiment
PC Schneider International 2/1987 S.108ff.

Martin Gerhard/Heike Schuster

Das digitale Universum
Vieweg 1995

Stephen Wolfram

A New Kind of Science
Wolfram Media 2002

Jörg R. Weimar

Simulation with Cellular Automata
Logos Verlag 1997/2003
<http://www.jweimar.de/ZAscriptmml/gliederung.html>

Daniel Scholz

Pixelspiele
Springer Spektrum 2014

Links

www.cell-auto.com

www.mirekw.com

www.fourmilab.ch/cellab