Manual: GeZA for Android

GeZA is an Android App to examine and browse 1 dimensional cellular automata. View the attractive images. GeZa is for free and without advertising. You'll get it from the Google Play Store.

Table of contents:

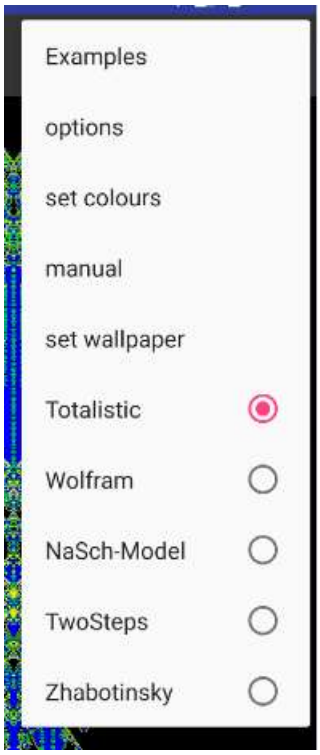
User Interface

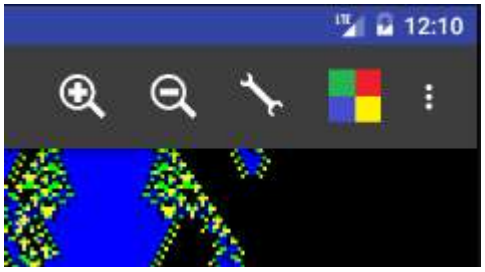
General

After starting GeZA a cellular automaton is be displayed. With wiping gestures (look table below) you can initiate additional functions.

wipe gesture	function
Short touch	start/stop animation
bottom up	Compute and show the next generations
top down	show the first generations, if it's already the first, then the initial start (only 1 cell of state 1) is used
right to left slowly	shows a new random, symmetric initial generation
right to left fast	shift display to the left (half display size)
left to right slowly	shows a new random, non symmetric initial generation
left to right fast	shift display to the right (half display size)

In addition to the wipe gestures there are more functions choosable in the drop down menu and icons in the title bar.





Depending of the physical display size, there are not displayed all icons in the title bar, then these functions are choosable via the drop down menu

menu item	Icon	function
zoom in		zoom in for more details.
zoom out		zoom out for better overview.
Examples		shows a list of all examples. Touching an icon starts the related automaton.
options		going to a dialog to change settings in current context.
set colors		changing colors for the different cell states.
manual	-	calling the manual
set wallpaper	-	setting the current image for wallpaper.
Totalistic	-	activates the totalistisc mode: more informations under fundamentals Totalistic.
Wolfram	-	activates the Wolfram mode: more information under Wolfram.
NaSch-Model	-	activates the NaSch-Model: more information under NaSch-Model.

Rule editor

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

You see the rule string on the top. Touching on this string sets the cursor. The number in red among them shows the current cursor/string position. This is useful, cause the character at this position dictates the value of the cell in the next generation, if the sum of the neighbour cells values is equal to the position number.
Touch the button "use it" and provides the new rule to the option dialog and goes back.

controling:

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.

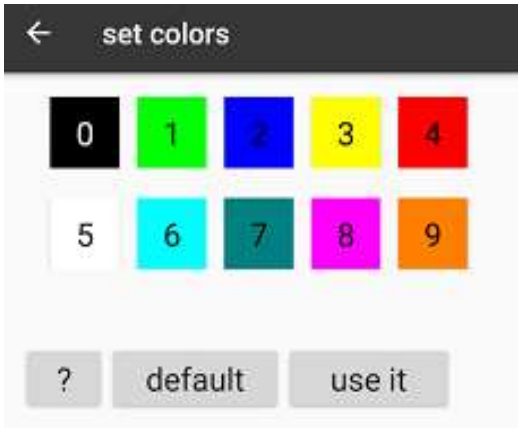
function of the rule characters:

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.

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

Set colours

This shows the current selected colours for the different states (0-9) of the cells.
You see 10, 2 or 7 cells depending of the context (totalistic, Wolfram, NaSch).



Clicking on such coloured cell will open a colour picking, which assign the new colour to the selected cell state.
To activate the new colours, click button "use this". The program will than show the main view screen with new colours.
For returning to the default colour scheme, click the button "default".

save file

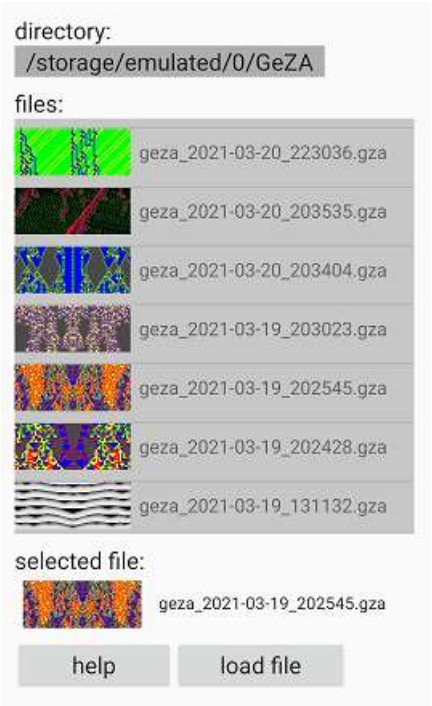
In this item you may write your current CA into a file (GeZA type and/or picture(PNG))

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 "gza". This file is readable by the GeZA app only, but of cause by other android devices also.

Activated checkboxes says which types of files will be written.
The filnames are the combination of "geza_", Timestamp and suffix.

read file

In this you open files, wich are writen by GeZA before.



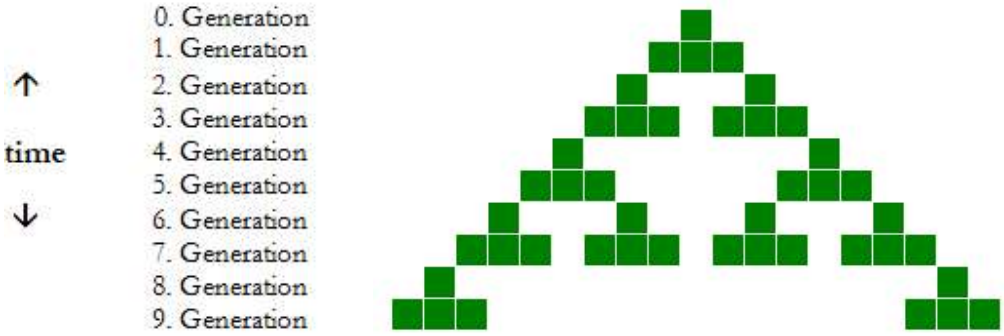
The app searches in a predefined directory for "GeZA files". This directory ("GeZA" on the internal memory) is not changeable yet. The complete name of this directory is shown in the first line.
Below you see the list of the files found (snapshot plus name). Clicking on one of this entries will show 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.

Fundamentals

Totalistic

A cellular Automaton (CA) is an amount of cells, each have a defined, almost discrete state (initial state). A rule, which considers the cell state and the state of defined neighbour cells, determines the cell state of the next generation. The cells may arranged in a line (1-dimensional), in a plain (2-dimensional) or higher dimensions.
One of the most famous CA is the Game of Life by John Conway from the 1970ies. It's 2-dimensional.

But with GeZA you can explore CA only in one dimension, just one line. The second dimension will show here the temporal lapse. This means that the ongoing generations are arranged from the top to the bottom. For example here the rule 0100, resp. 01 (missing digits interprets GeZA as 0).



Initial state (= 0. generation) is one cell with state 1 (green), all other 0 (white).

Rule 01 (3 Neighbours) means the following instruction for computing the next generation:
In this case a cell may have only the state 0 (white) or 1 (green)

sum from the 3 neighbor cells (left, middle, right)	0	1	2	3
state of the (middle) cell in the next generation	0	1	0	0

This rule is left/right symmetric. The CAs which rules depends only on the sum of the neighbours are called **"totalistic"**.

The digit in the nth pos give the state for the new generation, if the sum of the neighbour states is n-1. For example: sum is 3, the state in the next generation is given by the 4th position of the rule. Doesn't exist this position, it will become 0. This notation is different from Wolfram.

7 additional Characters geting the following special functions:

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

Wolfram

In this simple Wolfram cellular automaton, the cells gets values of 0 or 1.
For transformation to the next generation is interesting the state from the left neighbour, the cell itself and the right neighbour. That gives 2^3 = 8 possible states of neighbourhood. Every one of this 8 states you can dedicate the state 0 or 1. This will be the state for the regarded cell in the next generation. The 8 digits of 0 or 1 represents a binary number. This gives gives a distinct transformation rule.

neighbours	111	110	101	100	011	010	001	000
state of new gen.	0	0	0	1	1	1	1	0

This example is the rule 00011110 binary resp. decimal 30. Because of the importance of this rule, GeZA starts with this one in the Wolfram mode.
Another important rule is 110 (decimal). It is proven that rule 110 is an universal Turing machine
With this App you can explore the rules from 0 up to 255.

More information you'll find in the related Wikipedia entry.

NaSch-model

"NaSch" means the Nagel-Schreckenberg-model. It's a traffic simulation published by the 2 physicists Kai Nagel and Michael Schreckenberg in 1992.
It gaves the first explanation for "traffic jam out of nowhere".

The cells represents in the NaSch-model a street of equal sections. This cells are empty or contains a car with a defined, discrete speed. The speeds are one of six different values (from 0 upto 135 in steps of 27 km/h). The initial state is given by random. You can define the density of cars under menu item "options"(15% is standard).
A car accelerates, if it has enough empty cells in front and maximum speed isn't reached. A car slowed, if it has less cells in front. Additional there is an unmotivated slowing by a part of the cars. The amount of unmotivating slowing cars is defined by the "dawdle factor", which you find in options too.

state	representing
0	cell of street without car
1	car speed 0 km/h
2	car speed 27 km/h
3	car speed 54 km/h
4	car speed 81 km/h
5	car speed 108 km/h
6	car speed 135 km/h

Driving direction is left to right, the timing sequence is top-down. You can see: traffic jam moving to the left, though the cars moving to the right.

More information you'll find in the related Wikipedia entry.

TwoSteps

This type of cellular automata is similar to the "totalistic", but "TwoSteps" use different rules/neighborhoods alternatingly.
This means: One rule for odd generation and the other rule for even generation.
(In case of equality of rules and neighborhood this automaton is identical to the "totalistic".)

Zhabotinsky

The name of this CA is based on the Belousov-Zhabotinsky-reaction, a well known chemical oszillator.

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).

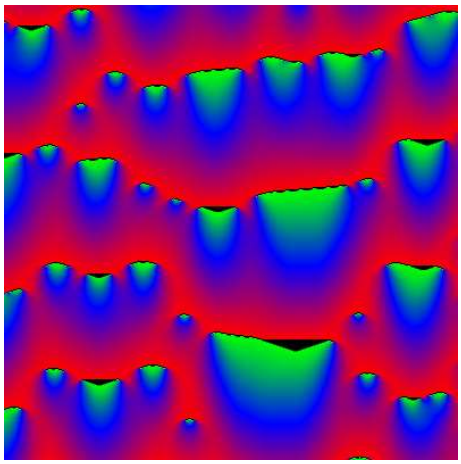
A cell of state 0 (healthy) will get the state [K/k1] + [I/k2] in the next generation.
K=sum of ill neighbors, I=sum of infected neighbors,
k1 = illness threshold (1-5), k2 = infect threshold,
the square brackets rounding off to integer.

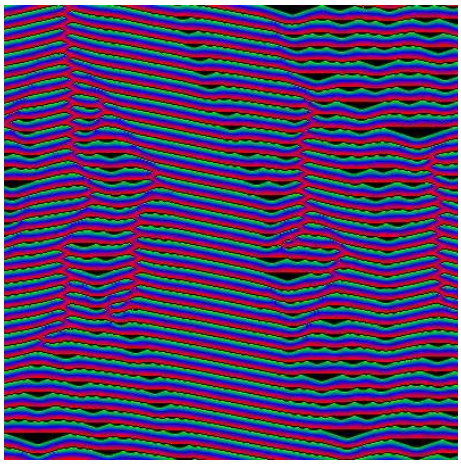
Infected cells (state 1-255) go to [S/A] + g, but max 256 in the next generation.
S = sum of the values from all neighbors and the cell itself. A is the number of these cells.
[S/A] is just the mean value.
g = constant (1-120) allways added, it is a kind of stimulation.

An ill cell (state 256) will be healthy (state 0) in the new generation.
This automaton is neighborhood with r=3. 

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:

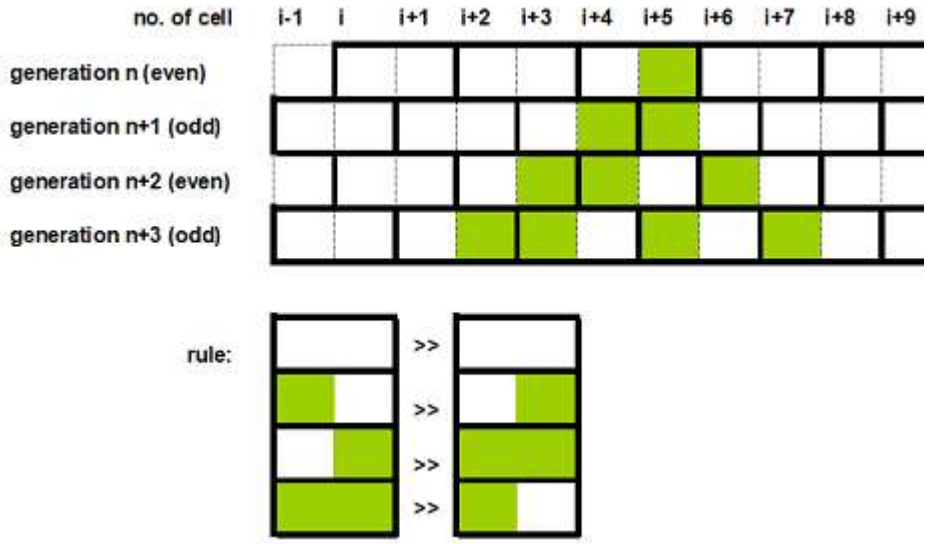

k1=1, k2=2, g=2


k1=1, k2=2, g=30

block cell automaton

In this type of cellular automaton are forming a constant amount of cells one block. A transformation rule decides, which states the associated cells take in the next generation. In the following generation the blocks are formed otherwise.

The type "block CA" in GeZA takes 2 cells for one block, in the next generation these blocks are shifted by one cell (see image below).



The mode "Margolus" in GeZA-2D is a block cell automaton also!

For more informations click the Wikipedia article block cell automaton:

Options

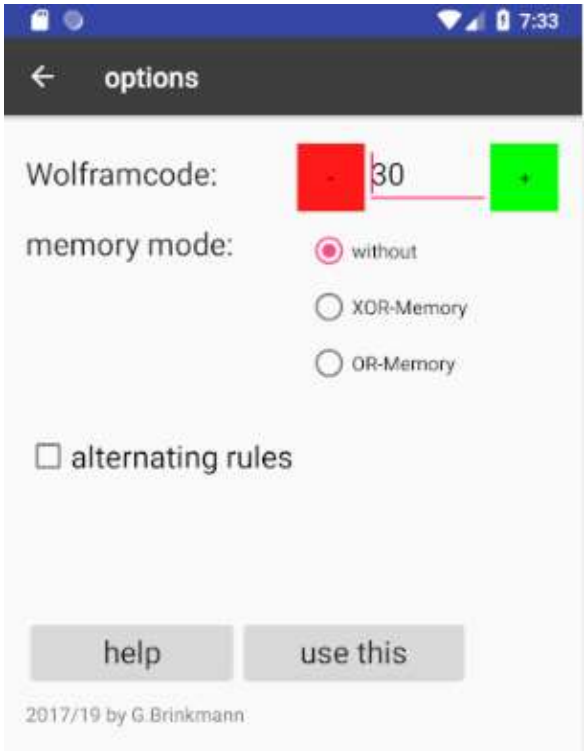
Totalistic

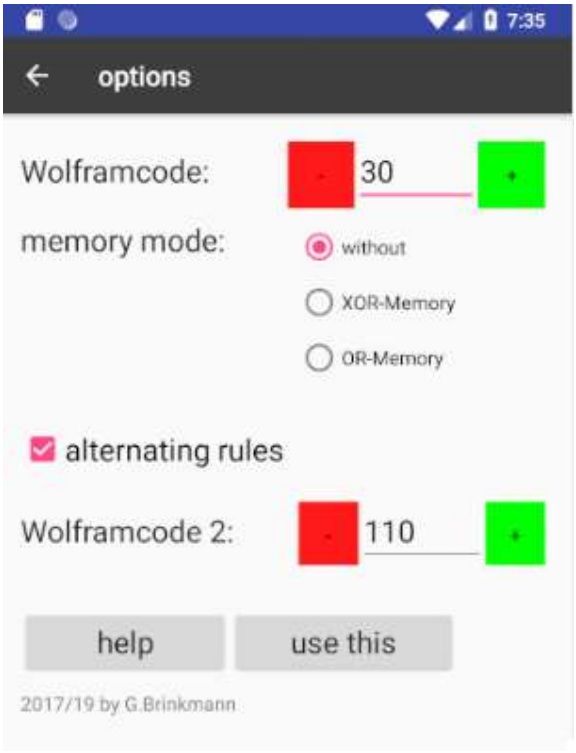
This dialog is for defining your own Automaton.
Code needs is a String of digits and special characters (+-=!?) This is the rule for transition to the next generation of a cell. More details you find in the manual in chapter 'fundamentals totalistic'!
'max. value of random' defines the largest random number for generating a randomized initial state. For question marks in the rule it's the largest random number too.

'totalistic' means that the states/values of the neighbour cells will be summed up. This sum is relevant for the transition rule.
The 5 button after neighbours defines the neighbourhood: Green is neighbour, grey is not. '0' is the cell itself.
For exploring the new Automaton touch button 'use this'.

Wolfram

Here you'll define the number of the Wolfram automaton.
You may set one of three Memory-Types also.
For more information see the manual chapter 'fundamentals Wolfram'.





Activating the 'alternating rules' checkbox, the input of a second wolfram code appears:
The automaton will use now the both rules alternatingly, the selected memory mode is used allways

For exploring the new Automaton touch button 'use this'.For more information see the manual chapter 'fundamentals Wolfram'.

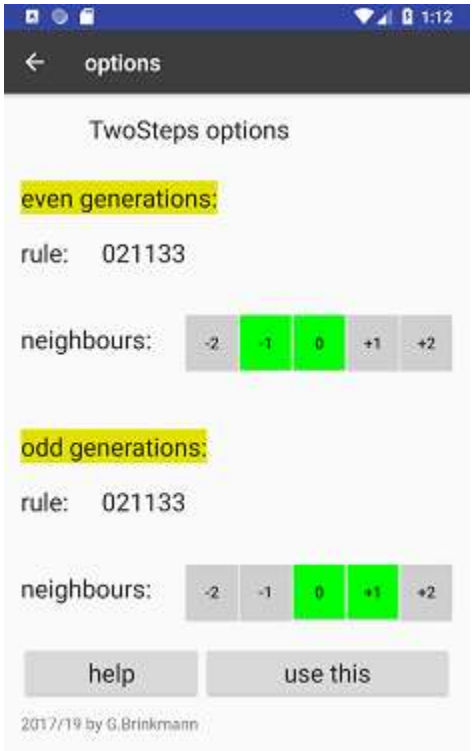
NaSch-Model

Here you can change the 2 parameters 'car density' and 'dawdle factor'.
Both may get a value from 0 up to 100.
More information you'll find in the manual chapter 'fundamentals NaSch model'.

For exploring the new automaton touch button 'use this'.

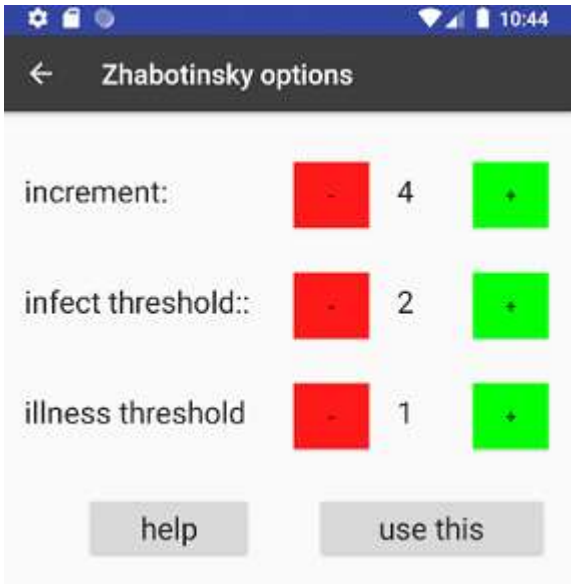
TwoSteps

"TwoSteps" works quite similar the "totalistic" type. But here you may define own rules for odd and even generations.
This dialog is for defining your own Automaton.
"rule"is a String of digits and special characters (+-=!?). This is the rule for transition to the next generation of a cell. More details you'll find in the manual in chapter 'fundamentals totalistic'!



'totalistic' means that the states/values of the neighbour cells will be summed up. This sum is relevant for the transition rule.
The 5 button after neighbours defines the neighbourhood: Green is neighbour, grey is not. '0' is the cell itself.
For exploring the new Automaton touch button 'use this'.

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.

Block CA

Here you may define the rules for the Margolus automaton.
In this type a block of 2 cells (for each cell are 4 states possible), 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 this new window.

Choose here the new state for the selected block and you go back.

Credits

GeZA - App for Android

created 2016/19 by G. Brinkmann

Webside: www.zellauto.de

E-Mail: geza2d@zellauto.de

Acknowledgements:

Without support, understanding, pushing by my wife Brigitte, GeZA 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