$ AUTOMATA

DISTRIBUTION MODE	About the shareware distribution mode of AUTOMATA for Windows (concerns only the limited version)
REGISTERED VERSION	What is the interest of the registered version AUTOMATA for Windows ? (concerns only the limited version)
INTEREST AND APPLICATIONS	What can you do with the AUTOMATA software ?
MOUSE USE	How to use the mouse with the AUTOMATA application?
KEYBOARD SHORTCUTS	What are the keyboard shortcuts ?
COMMANDS	What are the menu commands and functions of AUTOMATA ?
DISCLAIMER OF WARRANTY	What are the conditions of warranty?
SUPPORT	Which support is provided?
DLL	How to use AUTODLL.DLL?
CONFIGURATION	Which configuration is requested?
INSTALLATION	How to install the software?
FILES	Which files are included in the package?

MAIN_INDEX
$ MAIN MENU

$ + K **COMMANDS**

The AUTOMATA application contains the features listed below:

FILE
New
Save
About
Shareware info
Quit

EDIT
Copy informations window
Copy bitmap window
Copy mosaïc window
Copy array window

INITIALIZATION
Simple seed
Standard
Standard-2
Null
Random
Special patterns
Array fill

OPTIONS
Step
Array
Zoom
Mosaïc
Automata
Random
Colors
Transition rules
Neighborhood parameters
Preferences
Cumulative automata
Statistics

PROCESS
Next
Level
Until stable
Continuous
Continuous until

WINDOW
Informations
Zoom

---

m_commands
$ COMMANDS
+ pro:0010
K commands

Array window
Mosaïc
Multi-bitmap

SAMPLES
499-8 (LifeGame)
387-8-0-4-0-0-0-0-112
337-0-128-4-2-0-0-40
289-6-4-0-128-16-4-0-10
289-72-0-128-16-4-2

HELP
Index
Keyboard
Shareware
Commands

$ + K DISTRIBUTION MODE

This section only concerns the limited shareware version:

The AUTOMATA 3.4 application is a shareware program, and is provided at no charge to the user for evaluation. Feel free to share it with the others, but do not give it away altered or as part of another system.

If you find this program useful, and continue to use the AUTOMATA for Windows software, after a reasonable trial period, you must make a registration payment of 39$ (200 F). You will receive the extended version AUTOMATA for Windows 4.0.

You can register directly form Paul FRANCESCHI or form Public (software) Library with your MC, Visa, AmEx, or Discover card by calling 800-242-4PsL (from overseas: 713-524-6394) or by FAX to 713-524-6398 or by CompuServe to 71355,470, and you will receive your license. Please specify item number **#11005**. These numbers are for ordering only. Paul FRANCESCHI can NOT be reached at those numbers. To contact Paul FRANCESCHI for information about dealer pricing, volume discounts, site licensing, the status of shipment of the product, the latest version number or for technical information, or to discuss returns, write

Paul FRANCESCHI
Résidence la Pietrina
Avenue de la Grande Armée
20000 Ajaccio
FRANCE

The registration will license one copy for use on any one computer at any one time. Commercial users of the AUTOMATA for Windows software must register and pay for their copies within 30 days of first use. Site-License arrangements may be made by contacting the author.

Commercial users of AUTOMATA must register and pay their copies of AUTOMATA within 30 days of first use or their license is withdrawn.

---

m_distribution
$DISTRIBUTION
+ pro:0020
K distribution

$ + K **REGISTERED VERSION**

This section only concerns the limited shareware version:

The advantages of registering your version, are the following:

- you receive the **advanced version AUTOMATA for Windows 4.0**, including new functionalities and options:

- **Initialization/Array fill...** new function
- **Options/Cumulative automata...** functionalities
- available default fractal **Options/Automata/Sierpinski automaton**
- **Process/Continuous until...** option
- **Option/Neighborhood parameters**... allowing creation and use of thousands of new automata

- you are advised of the availibility of a **new version** of the Automata for Windows software

-you can directly **contact** the author; any comments or suggestions are welcome

- you can freely use, as a registered user, the AUTOMATA application **DLL**, named AUTODLL.DLL, in your commercial applications

---

m_register
$ REGISTERED VERSION
+ pro:0022
K register

$ + K APPLICATIONS

There are two main and distinct utilisations of the AUTOMATA software: the first is for scientific use, to study and analyse the cellular automata; the second is to create special bitmap graphics, with specific automata. AUTOMATA is both a scientific and a graphical software.

When considered as a **scientific software**, the AUTOMATA application can be used to study the automata properties. You can select different automata, among the 8^9 possibilities, and test them. You can then use the **Stability limit** option, to search the attractors of the current automaton. You can use the **Array visualisation mode**, or the bitmap graphical mode (in **Options/Preferences**) , depending on your study purposes.

If you choose to use the AUTOMATA application as a **graphical software**, you may select the bitmap visualisation mode, in the **Options/Preferences**. You can select pre-defined automata int the **Options/Automata**, whose graphical properties are already remarkable, or define your own automata, using the **Options/Parameters**, which allows you to modify the parameters of the current automaton.

You may also change the colors corresponding to each state (form 0 to 7); you will the use the **Options/Colors** option in the main menu, and access to the default colors modification. Thi possibiliy is very important for graphical creation, and you may define your own colors, corresponding to your particular scopes. An other available and interesting mode, is to modify the colors, several times, when using a specified automaton. Colors can be changed at any time, according to the effects desired.

When you are satisfied of your bitmap, you can transfer and export toward another graphical software, such as PAINTBRUSH, COREL DRAW, etc. or a word processor such as WRITE, WINWORD, etc. The bitmaps created withe the AUTOMATA application are in WINDOW 3.x bitmap format (.BMP), and are compatible with other softwares. Simply copy your bitmap in the clipboard, using the **Edit/Copy bitmap** option, and transfer the bitmap in the other application, with the **Edit/Paste** or equivalent command of the destination software.

---

m_applications
$ APPLICATIONS
+ pro:0024
K applications

$ + K **DISCLAIMER OF WARRANTY**

THE AUTOMATA APPLICATION AND ITS DOCUMENTATION ARE SOLD "AS IS", AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETER EXPRESSED OR IMPLIED, BECAUSE OF THE VARIOUS HARDWARE AND SOFTWARE ENVIRONMENTS INTO WHICH THIS PROGRAM MAY BE PUT, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. ANY LIABILITY OF THE SELLER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF PURCHASE PRICE.

THE AUTHOR ASSUMES NO LIABILITY FOR DAMAGES, DIRECT OR CONSEQUENTIAL, WHICH MAY RESULT FROM THE USE OF AUTOMATA.

---

m_warranty
$ WARRANTY
+ pro:0026
K warranty

$ + K **SUPPORT**

A mail suport is provided during a period of 3 months, from the date of registration.

Elsewhere, your suggestions, remarks and propositions are greatly encouraged and accepted. Any suggestion concerning new options or fonctionnalities are welcome. Your feedback, if your are a professionnal user, is important, too.

---

m_support
$ SUPPORT
+ pro:0028
K support

$ + K **DLL**

The AUTOMATA package contains a file named AUTODLL.DLL; this is a DLL, to use with your WINDOWS applications. The commercial use of the DLL is free, if you are a registerd user.

You can use the following classical C code sequence, in your WINDOWS program, to access to the bitmaps included in the AUTODLL.DLL of the AUTOMATA application:

```
/* Load the AUTODLL.DLL DLL */
hLibrary= LoadLibrary( "AUTODLL.DLL");
(...)
/* Load the nBitmap bitmap of the DLL */
hBitmap= LoadBitmap( hLibrary, MAKEINTRESOURCE( nBitmap) );
```

---

m_dll
$ DLL
+ pro:0030
K dll

$ + K **CONFIGURATION**

The system requirements for the AUTOMATA for Windows application, are the following,:

- an IBM-PC compatible computer: at least 80386
- WINDOWS 3.1 or further
- 2 megas RAM
- a mouse or tablet suported by Microsoft Windows 3.1 or further

- a color monitor with a VGA card is not necessary, but highly recommended
- a color printer may be useful, for professional use of the AUTOMATA Software

---

m_configuration
$ CONFIGURATION
+ pro:0032
K configuration

$ + K INSTALLATION

Microsoft Windows 3.1 or further must be installed and working, before you install the AUTOMATA application.

There are two methods to install the AUTOMATA for Windows software:

Method 1:

Place your AUTOMATA application disk in drive A or B; select the concerned drive A or B using DOS command A: or B:, and type:

INSTALL and then press **<*Return*>** . Then wait for complete installation.

The executable file is the file named AUTO.EXE. Type WIN AUTO, or double-click on the AUTOMATA application icon.

Method 2:

Copy all files provided with the AUTOMATA application in a directory of your hard drive.

For example, you can create a directory named AUTOMAT and copy all the files in it, using the following command:

COPY *.* C:\AUTOMAT

---

m_installation
$ INSTALLATION
+ pro:0034
K installation

$ + K **FILES**

The AUTOMATA application includes the following files:

- AUTO.EXE: the executable file
- AUTO. HLP: the help file
- AUTO.DLL: the Dynamic Link Library of the application
- AUTO.INI: the initialisation file, containing the application parameters

---

m_files
$ FILES
+ pro:0036
K files

$ + K **MOUSE**

When double-clicking on one element, it will automatically increment its state: 0-state cell will become 1-state cell. Another double-click on the same cell will change it to state 2, etc. When the state 7 is reached, a double-click on the concerned cell will select state 0.

Double-clicking outside the limits of the array, will cause an iteration. This action is equivalent to **Process/Next** menu command.

---

m_mouse
$ MOUSE
+ pro:0038
K Element, array

$ + K **KEYBOARD SHORTCUTS**

Several keyboard shortcuts are available, in order to optimize the options access for the experimented user:

Ctrl-N	**Process/Next**
Ctrl-L	**Process/Level**
Ctrl-U	**Process/Until stable**
Ctrl-Q	**File/Quit**
Ctrl-C	**Initialization/Special patterns/Cross**
Ctrl-P	**Initialization/Special patterns/Point**
Ctrl-S	**Initialization/Special patterns/Square**
Ctrl-R	**Initialization/Special patterns/Rhomb**
Ctrl-H	**Initialization/Special patterns/Hexagon**
Ctrl-O	**Initialization/Special patterns/Octogon**

---

m_keyboard
$ KEYBOARD
+ pro:0039
K keyboard

$ + K **NEW**

Clears the whole array elements. Cells are initialized with the standard initial configuration of the Life Game. All options and preferences previously defined are cancelled. The options, preferences, and parameters of the currently saved configuration (included in AUTO.INI file) are active.

---

m_new
$ NEW OPTION
+ pro:0040
K initialization, array;elements

$ + K **SAVE**

This options saves the configuration file, including the options and preferences declared.

At next use of the software, the options and preferences previously saved, will be automatically loaded.

A file called **AUTO.INI** is automatically written in \WINDOWS repertory, including all selected preferences and parameters.

---

m_save
$ SAVE OPTION
+ pro:0050
K options, preferences

$ + **ABOUT**

Shows Copyright informations about the AUTOMATA application. The version number of the software also appears on this box.

---

m_about
$ ABOUT OPTION
+ pro:0060

$ + **SHAREWARE INFO**

This section only concerns the limited shareware version:

This option shows a dialog box, with informations about AUTOMATA for Windows, and its shareware distribution mode. The main modalities of registration are described. See **Distribution mode** section in this Help file, for more informations.

---

m_shareware
$ Shareware
+ pro:0062

$ + **QUIT**

Quits the AUTOMATA application, and returns to WINDOWS (TM) interface.

---

m_quit
$ QUIT OPTION
+ pro:0070

$ + K **COPY INFORMATIONS**

Copy the main informations about the current automaton, and its state. All informations displayed in the Informations window (automaton number, current level, etc.), are copied in the clipboard. This option is useful to note all interesting informations about a specific automaton, in order to re-use it. All the informations copied, can be transfered to another WINDOWS application, such as a word processor.

---

m_copyinfo
$ COPY INFORMATIONS
+ pro:0080
K copy, informations

$ + K **COPY BITMAP**

Copy the current bitmap in the clipboard, in order to transfer it to another WINDOWS application, such as a graphical software, or a word processor. The current bitmap is exported, via the clipboard to other softwares. This option is fundamental for professional use of the AUTOMATA software. Sophisticated, esthetical bitmaps can be created with specific automata, and exported to other applications.

If you use the AUTOMATA software as an **icon drawer**, you may choose a 32x32 array, and transfer the resulting bitmap with the **Edit/Copy bitmap** option, to another application.

# m_copybitmap
$ copy BITMAP
+ pro:0082
K copy, bitmap

$ + K **COPY MOSAIC**

Using this option, you copy the bitmap resulting of the **Mosaïc window** option in the clipboard. The background window is created by the juxtaposition of the current bitmap, representing the current state of the automaton. The background window can be compared to the **Mosaïc** option of the WINDOWS interface.

---

m_copybackground
$ COPY BACKGROUND
+ pro:0084
K copy, background

$ + K **COPY ARRAY IMAGE**

Copy the contents of the current array, when the Array visualisation mode (**Options/Preferences**) is selected. It is not possible with the Bitmap visualisation mode. The array can be considered as a particular zooming mode, for graphical work.

Copy array image may be useful when AUTOMATA is used as a scientific software, to study and analyse AUTOMATA.

---

m_copyarray
$ COPY ARRAY IMAGE
+ pro:0086
K copy, array, image

$ + K **SIMPLE SEED**

Initializes the array with a central simple site, of state 1.
For example, with a 1-dimensional array:

---

m_simpleseed
$ SIMPLE SEED
+ pro:0087
K seed

$ + K **STANDARD**

Initializes the array with the **Life Game** standard beginning: four elements are present.

If some preferences or parameters have been defined, using the **Preferences** or **Parameters** options, they are still active.

---

m_standardinit
$ STANDARD INITIALIZATION
+ pro:0088
K array

$ + K **STANDARD-2**

Initializes the array with the following pattern, which creates very long and interesting sequence, with the **Life Game**:

---

m_standard2init
$ STANDARD-2 INITIALIZATION
+ pro:0089
K Life Game

$ + K **NULL**

Initializes the array with no elements. All cells are set to 0-state. This option may be used to configure the initial array, as desired. You can add cells where you want, by double-clicking at a cell position, and then use **Process** options(**Next**, **Level**, **Until stable**).

If some preferences or parameters have been defined, they are still active.

The following 4x4 array, is initialized with the **NULL** command:

---

m_nullinit
$ NULL INITIALIZATION
+ pro:0090
K array, elements

$ + K **RANDOM**

Initializes the array with random elements. Using this option, initialization is always different.
If some preferences or parameters have been defined, they are still active.
The count of elements randomly determined for each specific state are defined with the **Options/Random** option.
The following array results from the **random** option application:

---

m_randominit
$ RANDOM INITIALIZATION
+ pro:0100
K array, elements

$ + K **SPECIAL PATTERNS**

Initial figures are very important, when you use a specific automaton; the result mainly depends on the initial configuration. The following special patterns, for initialisation of a selected automaton are available:

- cross

- point (central point)

- screw (7x7)

- screw (8x8)

- square

- line (3 aligned and central points)

- rhomb

- hexagon

- square

- octogon

All the initial figures central and constitued of state-1 (red, by default) points.

---

m_patternsinit
$ SPECIAL PATTERNS INITIALIZATION
+ pro:0102
K figure

$ + K **ARRAY FILL**

Fills the current array with a central rectangle (1-state cells) whose coordinates are parametrable.

For example, the following array results from the use of the **Array Fill** option, 7x3 parameters whose width and height are respectively 7 and 3:

The use of the **Array Fill** option with 5x5 parameters produces:

---

m_arrayfill
$ ARRAY FILL INITIALIZATION
+ pro:0104
K array

$ + K **STEP**

Defines the step selected, for each iteration.

At loading time, this value is set to 1 (single step).

A message error is emitted if the step, added to current iteration level, is greater than the **Stability limit** (500 by default).

---

m_step
$ STEP
+ pro:0120
K iteration

$ + **ARRAY**

Defines the array dimensions. Horizontal and vertical values are declared.

New width and height of cells, are automatically computed and displayed, if the **Dynamic cell** option has been selected (See **Options/Preferences**).

An error message is emitted, if **Horizontal maximum**, or **Vertical maximum**, exceed certain limits: respectively 100 and 100.

---

m_array
$ ARRAY OPTION
+ pro:0130

$ + K **ZOOM**

This option allows you to examine the bitmap pattern created by the current automaton, with a scale zoomed factor of 2, 3, 4,or 5. By default, the zoom uses a x 2 factor.

---

m_zoom
$ ZOOM OPTION
+ pro:0132
K zoom

$ + K **MOSAIC**

This option displays a new window, composed of juxtapositions of the current bitmap, considered as a repetitive pattern.

It is useful to visualize and see the effect produced by the repetition of the current bitmap.

The following images results of the use of this option:

---

m_background
$ BACKGROUND OPTION
+ pro:0134
K background

$ + K **MULTI-BITMAP**

This option displays a new window, including the following states of the current automaton. The successive steps are displayed.

The following are examples of the resulting window, when this option is selected:

Here are the 27 first iterations of the 385-32-8-128-2-16-0-64 automaton, with a 32x32 array, intialized with a 8x8 screw:

---

m_multibitmapwindow
$ MULTI-BITMAP WINDOW OPTION
+ pro:0135
K multi-bitmap

$ + K **AUTOMATA**

The following automata are available:

- **Life Game**: 499-8

- 387-8-0-4-0-0-0-112

- 387-8-0-0-16-0-0-100

- 289-64-0-128-16-4-0-10

- 289-72-0-128-16-4-2

- 385-32-8-128-2-16-0-64

- 297-64-16-132-0-0-0-10

- 337-0-128-4-2-0-0-40

- **Sierpinski**: 32-4 (only available on extended 4.0 version)

These automata are remarkable, and you may test their astonishing properties.But this list is non-exhaustive, and I recommend you to explore and experiment other automata.

---

m_automata
$ AUTOMATA OPTION
+ pro:0136
K automata

$ + K **RANDOM**

This option is necessary to adjust the random options parameters. A dialog box appears, allowing you to select the amount of elements initially affected to a selected automaton. For each state, the count of elements can be adjusted.

---

m_random
$ RANDOM OPTION
+ pro:0138
K array, elements

$ + K **COLORS**

Using this option, a dialog box appears, allowing you to select a specific color, for each state. The default colors can be modified, to create special effects, and specific graphical patterns.

---

m_colors
$ COLORS OPTION
+ pro:0140
K colors

$ + **TRANSITION RULES**

Defines the **transition rules** for the AUTOMATA application.

Please carefuly read this section, which is very important, for an efficient use of the AUTOMATA application.

Modifying parameters, and testing AUTOMATA is highly recommended; many kinds of AUTOMATA are astonishing, and their properties are very different.

The standard parameters are initialized with the automaton used to create the Life Game (John CONWAY).

For 16-state and 9-neighbours automata, that are implemented, it is necessary to have an efficient **numbering method** for precise identification and individualisation of automata, in order to recognize without ambiguity whatever automaton has been selected. An amount of 8^9 different automata is **theoretically** available, corresponding to 8 states and 9 possibilities of transition for each state, according to the total count of neighbours for each cell: this count can be equal to 0, 1, 2, ..., 7, or 8, corresponding to 9 different possibilities.

In a perspective of **ascending comptability**, whose specifications are increase of available states and colors, with the same numbering method, we assign a number to every set of transition rules determining every state change: this allows the complete identification of a specific automaton with 8 numbers. If new states and colors, or additionnal neighbours, are subsequently added, it will not be necessary to modify the numbering system. Accordingly, we can consider our rules array of 8 states and 9 neighbours, as the top left part of an illimited and extensible array, consisting of m columns (neighbours) and n rows (states).

Consequently, the **Life Game automaton** corrisponds in this numbering system, to the automaton referenced 499-8-0-0-0-0-0-0, that is to say the **automaton 499-8**, because the 6 following states or colors are set to null and not used. We also remark another advantage of this numbering system: it is not necessary to specify the maximum count of availablecolors and states.

For automata of this family, the **n numbers** are coding the **whole set of transition rules to state n**, from 0 to 2^9 - 1 (that is to say from 0 to 511); the first number is coding the commutation rule to state 0. For the Life Game automaton, 499 corrisponds to 2^0 + 2^1 +2^4 +2^5 +2^6 +2^7 +2^8 , because the transition to state 0 is made when the amount of life neighbours reaches 0, 1, 4, 5, 6, 7 ou 8. Similarly, 8, the second number identifying the automaton, is coding the transition rule to state 1, and corrisponds to 2^3, because the only case where a cell becomes life, is when the total of its life neighbours reaches 3.

---

m_parameters
$ PARAMETERS OPTION
+ pro:0145

$ + NEIGHBORHOOD PARAMETERS

Neighborhood is parametrable, in order to allow the implementation of many automata. You can also give a coefficient **b** to each of the 8 neighboring cells of a central site, and the central cell itself. So 9 total parameters of neighborhood are available.

For 1-dimensional automata, only 3 cells are eventually used: the central site **a(0)** itself, and its left **a(-1)** and right neighbors **a(1)**.

For 2-dimensional automata, only 9 cells are eventually used: the central site **a(0, 0)** itself, its left **a(-1, 0)** and right neighbors **a(1, 0)** , and the upper **a(-1, -1)**, **a(-1, 0)**, **a(-1, 1)** ,and lower sites **a(1, -1)**, **a(1, 0)**, **a(1, 1)**.

When the sum Σ **a(i, j)*b(i, j)** is computed, a modulo is applied, depending on the modulo option selected. The **b(i, j)** values are the coefficients for each of the 9 available neighbors.

With the parameter **Homogeneous neighbor mode** set to 1, the value **a(i, j)** is:

- set to 1 if the current state of the cell is diffenrent from 0
- set to 0 if the current site is a 0-state cell

If the **Homogeneous neighbor mode** is set to 0, the value **a(i, j)** is equal to the current value of the cellstate.

The calculus of the modulo applied to the cellular automaton may result from different techniques:

1) by default, this value is set to 10 (the maximum neighbors + 1)
2) this value can be user-defined
3) this value may be the maximum + 1 neighbors used in the **Transition rules** option
4) this value may be the maximum + 1 neighbors used in the **Neighborhood Parameters** option

A few neighborhood parameters are pre-defined:

- **1-dimensional**: 0-0-0-1-1-1-0-0-0 from top left to bottom right
- **2-dimensional**: 1-1-1-1-0-1-1-1-1 from top left to bottom right
- **5-neighbors**: 0-1-0-1-1-1-0-1-0 from top left to bottom right
- **9-dimensional**: 1-1-1-1-1-1-1-1-1 from top left to bottom right

The available automata are called *Totalistic automata*.

With the automaton, a 49x49 array, a single seed initialization, **Homogeneous neighbor mode** = 1, and a standard 2-dimensional neighborhood, the 32 first iterations are the following:

With the same automaton, but with a neighborhood parameters defined as (from left top to right bottom) 1-2-1-2-0-2-1-2-1 we obtain:

With the same automaton, but with a neighborhood parameters defined as (from left top to right bottom) 1-2-1-2-4-2-1-2-1 we obtain:

With the same automaton, but with a neighborhood parameters defined as (from left top to right bottom) 1-3-1-3-3-3-1-3-1 we obtain:

---

m_neighbourhood
$ NEIGHBOURHOOD OPTION
+ pro:0147

$ + **PREFERENCES**

Defines the user preferences:

DYNAMIC CELLS: this options, when selected, induces automatic computation of cell size, adapted to current window dimensions. At the beginning, this option is non-active: the cell height and witdth are set to a value of 20 pixels.

STATIC CELLS: the value indicated is used for width and height of the cells. This option is not active when the DYNAMIC CELLS options has been selected. If you use large arrays, we recommend that you choose small **static cells** value, such as 10.

SHOW GRID: when selected (default value), a grid is shown on the main window, separating each element.

ICON BAR: when selected (by default) the icon bar appears on the screen; if unselected, the icon bar is not present

ELLIPSE / RECTANGLE: determines the current shape of each site, in the array window.

ITERATION LIMIT: this value is initially set to 500; it is used by the **Process/Until Stable** option. If no stable state is found for the current automaton, iterations are continued until an amount of 500 iterations will be reached.

An error message is emitted if this value is less than the current **Level** value.

ITERATION INCREMENT: this value is initially set to 250; it is used by the **Process/Until Stable** or **Process/Continuous** option. If the iteration limit is reached, a dialog box appears; il the user prefers to continue the iterations, the **iteration increment** is added to the initial **iteration limit**. For example, with the default parameters, the original iteration limit is 500. If this limit is reached, and the user wishes to continue, 250 is added, so the new value for the iteration limit is 750.

---

m_preferences
$ PREFERENCES OPTION
+ pro:0150

$ + CUMULATIVE AUTOMATA

Note: The introduction is the AUTOMATA sofware of this option is due to an idea of Alain BRUGUIERES (Paris VII University).

DISPLAY SUCCESSIVELY: to use specially with mono-dimensional automata; the different are displayed successively, in order to analyse the cumulative pattern resulting.

MULTI-BITMAP HORIZONTAL DISTANCE: number of pixels separating horizontally several bitmaps, when using the multi-bitmap window option. By default: 5.

The following image (20 first steps of the **385-32-8-128-2-16-0-64** automaton, with a 32x32 array, randomly initialized) results from the use of the **Copy multi-bitmap window** option. Bitmaps, by default, are horizontally and vertically separated by 5 pixels:

MULTI-BITMAP VERTICAL DISTANCE: number of pixels separating vertically several bitmaps, when using the multi-bitmap window option. By default: 5.

The following image results from the use of the **Copy multi-bitmap window** option, with Multi-bitmap horizontal and vertical distances set to 0 (automaton 387-8-0-4-0-0-0-12 , initialized with cross 3x3, and 4x64 array):

This parameter is interesting to study cumulative bitmaps, resulting of juxtaposition of several levels of an automaton.

A remarkable automaton -available in extended 4.0 version - , is **Sierpinski** (32-4); it produces auto-reproductive patterns, and the cumulative resulting bitmaps are fractals. For example, the following fractal pattern named Sierpinski's triangle is obtained with :

- **Sierpinski** automaton
- an **array** of 2x100
- **Preferences/ horizontal and vertical multi-bitmap distances** set to 0
- an initial figure: **point** 2x2
- 16 **iterations**

In array mode, we obtain the following 16 iterations:

Level 1

Level 2

Level 3

Level 4

Level 5

---

m_cumulative
$ CUMULATIVE AUTOMATA OPTION
+ pro:0153

Level 6

Level 7

Level 8

Level 9

Level 10

Level 11

Level 12

Level 13

Level 14

Level 15

Level 16

The minimal cellulara automaton to realize the Sierpinski' s triangle is the **32-4** automaton, whose transition rules are the following; but others automata - such as **297-64-16-132-0-0-0-10** - produce Sierpinski' s triangle

- commute to state 1 if the count of neighbours is equal to 2
- commute to state 0 if the count of neighbours is equal to 5

With the **32-4** automaton, you obtain the following triangle, in 32 iterations:

$ + **STATISTICS**

Display a few statistics concerning the current state of the automaton.

---

m_statistics
$ STATISTICS OPTION
+ pro:0155

$ + **NEXT**

Computes and displays next step of the current automaton.
An error message appears, if the level value of next iteration is greater than the **Iteration Limit**.

---

m_next
$ PROCESS NEXT
+ pro:0160

$ + K **LEVEL**

Using this option, you can select a specific level, for the current automaton. The new state of the automaton is showed. For complex, large arrays, and high levels, a delay may be necessary.

---

m_level
$ PROCESS LEVEL
+ pro:0165
K level

$ + **UNTIL STABLE**

Computes the iteration process, until a stable state of the current automaton is reached.

This option is specially useful to study the limits automata. Stable configurations are remarkable, and are often called attractors of the automaton.

This functionality is provided for scientific use and study with the AUTOMATA application.

---

m_untilstable
$ PROCESS UNTIL STABLE
+ pro:0170

$ + **CONTINUOUS**

Computes the iteration process, continuously. The iterative process is halted:
- if the user presses a key
- or if the iteration limit is reached

---

m_continuous
$ PROCESS CONTINUOUS
+ pro:0171

$ + **CONTINUOUS UNTIL**

Computes the iteration process, continuously, until the specified level il reached. The iterative process is halted:

- if the user presses a key
- or if the iteration limit is reached

---

m_continuousuntil
$ PROCESS CONTINUOUS UNTIL
+ pro:0172

$ + K **INFORMATIONS WINDOW**

A window describing several informations appears on the screen, when you select this option. The iteration current level, the amount of modified elements, the count of non-state-0 elements, are displayed on the screen, in an independent window.

Selecting the **Window/Informations** option activates the informations window.

The informations windows can be re-sized, and adapted to your specific needs.

An example of the informations displayed in this window, is shown below:

Level: 26
Count: 708
Changes: 0
Automaton: 387 8 0 0 16 0 0 100 0 0 0 0 0 0 0 0

---

m_informationswindow
$ INFORMATIONS WINDOW
+ pro:0173
K informations

$ + K **ARRAY WINDOW**

Selecting this option activates a resizable **Array window**, representing the whole automaton current array. This option is useful for scientific study of the automata cells evolution. When AUTOMATA is used as a graphical creation software, this option can also be considered as an alternative to the **Zoom** option.

---

m_arraywindow
$ ARRAY WINDOW
+ pro:0174
K array, window

$ + K **ZOOM WINDOW**

Selecting this option activates a resizable **Zoom window**. The **Zoom window** allows to increase the size of the bitmap graphical image of the current automaton, by a specific scale factor (2 by default).

---

m_zoomwindow
$ ZOOM WINDOW
+ pro:0175
K zoom, window

$ + K **MOSAIC WINDOW**

This option selects a resizable Background window, corresponding to the option specified with the **Options/Mosaïc** option.

---

m_backgroundwindow
$ BACKGROUND WINDOW
+ pro:0176
K background, window

$ + K **SAMPLES 499-8**

The 13 first iterations of the Life Game are showed. At level 12, an oscillation is encountered, and a repetitive sequence is done.

---

m_sample1
$ SAMPLES 1
+ pro:0180
K samples

$ + K **SAMPLES 387-8-0-4-0-0-0-0-112**

A few bitmaps, produced by the 387-8-0-4-0-0-0-0-112 automaton are showed

These bitmaps have been created with the Graphical visualisation mode (See **Options/Preferences**), and exported to PAINTBRUSH via the clipboard and the **Edit/Copy Bitmap** option.

---

m_sample2
$ SAMPLES 2
+ pro:0190
K samples

$ + K **SAMPLES 337-0-128-4-2-0-0-40**

A few bitmaps, produced by the 337-0-128-4-2-0-0-40 automaton are showed.

These bitmaps have been created with the Graphical visualisation mode (See **Options/Preferences**), and active Background window, and then exported to PAINTBRUSH via the clipboard and the **Edit/Copy Background** option.

Initial configuration used is **Screw**.

---

m_sample3
$ SAMPLES 3
+ pro:0200
K samples

$ + K **SAMPLES 289-6-4-0-128-16-4-0-10**

A few bitmaps, produced by the 289-6-4-0-1-128-16-4-0-10 automaton are showed.

These bitmaps have been created with the Graphical visualisation mode (See **Options/Preferences**), and exported to PAINTBRUSH via the clipboard and the **Edit/Copy Bitmap** option.

---

m_sample4
$ SAMPLES 4
+ pro:0210
K samples

$ + K SAMPLES 289-72-0-128-16-4-2

Not implemented.

---

m_sample5
$ SAMPLES 5
+ pro:0220
K samples

$ + K **HELP INDEX**

AUTOMATA is a self-documented software. This option load AUTOMATA Help file, named AUTO.HLP.This option displays the index of the help file.

---

m_helpindex
$ HELP INDEX
+ pro:0230
K help, index

$ + K **HELP KEYBOARD**

Displays the keyboard shortcuts of the AUTOMATA application.

---

m_helpkeyboard
$ HELP KEYBOARD
+ pro:0240
K help, keyboard

$ + K **HELP SHAREWARE**

Explains the concepts of shareware used by the AUTOMATA application.

---

m_helpshareware
$ HELP SHAREWARE
+ pro:0250
K help, shareware

$ + **HELP COMMANDS**

This option describes the commands of the AUTOMATA application main menu.

---

m_helpcommands
$ HELP COMMANDS
+ pro:0260