# LifeMaker 2.1

## I. About LifeMaker

LifeMaker is a cellular automata program for color Macs. Cellular automata (CA) can be thought of as mini-universes. The universe starts at some initial state and evolves according to a simple local rule. Conway's Game of Life is by far the best known CA. LifeMaker includes Conway's Life along with 32 other rules. Unlike most Life programs LifeMaker uses color, allows you to change the size of the "universe", and lets you create your own rules. LifeMaker requires 32 Bit QuickDraw 1.1 or above.

LifeMaker is free, but cannot be sold for profit without my prior consent. If you have any comments, suggestions, or bug reports please send them to:

CompuServe: >INTERNET:jesjones@halcyon.com
Internet: jesjones@halcyon.com

## II. Basics of CA

A cellular automata consists of a bunch of cells and a rule determining the evolution of the system. The cells typically exist in one or two dimensions (but see [4] for a fascinating 3D version of Life). Each cell has an associated state. The number of possible states is finite and usually quite small. In LifeMaker the cells are arranged in two dimensions and there are a maximum of four states with each state having its own color.

The rule uses the state of a cell and its neighbors to find the cells new state. The rule is applied to each cell simultaneously so that the cells change all at once. The neighborhood of a cell can be any finite collection of nearby cells. For 2D CA the most command neighborhoods are the Moore (the eight immediately adjacent cells) and the Von Neumann (the four cells to the north, south, east and west).

To see how the rules work examine figure 1. It shows a pattern evolving according to the Life rule. In this rule cells remain alive if they have two or three living neighbors. Cells are born if they have three living neighbors. Life is an example of a *totalistic* rule: a cells new state depends on the sum of its neighbors states.

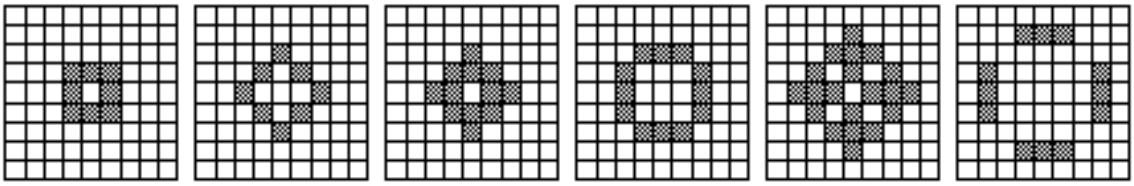

**Figure 1:** Conway's Life

A rather different neighborhood is used by partitioning cellular automata. Here the cells are divided into a finite number of disjoint uniformly arranged blocks. The rule looks at a block and updates the whole block at once. Since the blocks do not overlap information does not move across blocks. To get around this the partition changes from one step to the next.

The simplest partitioning CA uses blocks of size 2x2. The block moves down and to the right with the next generation. It then moves back (see figure 2). This partitioning scheme is called the Margolus neighborhood and is useful for modeling physical systems.

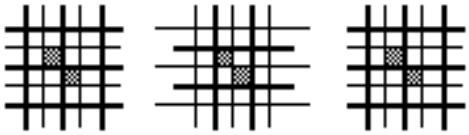

**Figure 2:** The Margolus neighborhood

## III. Uses of CA

Cellular automata are useful because they directly model systems with a large number of elements that behave in a non-linear fashion. These systems must, in general, be solved numerically. For example fluid flow is governed by a non-linear differential equation called the Navier-Stokes equation. It turns out that a simple CA using a hexagonal lattice obeys the Navier-Stokes equation exactly. Obstacles can be added by merely drawing them on the screen. Different amounts of friction can be programmed in by altering the texture of the obstacles. See [7] for a discussion of the advantages of CA and [13] for a description of an experiment using a Connection Machine.

Cellular automata have several properties that make them useful for modeling physical systems. To begin with neighborhoods are finite so action at a distance is impossible. In other words CA are *local*. CA are also *uniform* : they obey the same rule everywhere. Finally CA rules can be designed so that they are *reversible* [15]: every pattern has a unique predecessor so that the CA can be run backwards in time. These three properties describe matter at a fundamental level. Using them we can model a multitude of diverse phenomena, e.g. annealing, diffusion, fluid dynamics, optics, Ising systems, and the Zhabotinsky reaction. See Wolfram [16] for examples.

Cellular automata are also interesting from a Computer Science perspective. Von Neumann developed CA in the forties to illustrate how self-reproduction can take place within an extremely simple framework. Later it was shown that it is possible to construct CA that are *computation universal* . In other words a CA can, in theory, do anything a digital computer can do. In fact Conway's Life has been shown to be computation universal [6]. An even more interesting computation universal CA is Fredkin's [5, 8, 15] "billiard-ball" computer. The novelty here is that the computer is reversible.

Cellular automata concepts have also been used to design parallel computers. CA are of course inherently parallel so that speed of operation is directly proportional to the number of processors. CA are also local so that speed of light constraints are not an issue. The machine can therefore be both large and fast. This type of parallel computer is called a *systolic array* [3, 9].

## IV. Writing Rules

A cells state changes based on its current state and the state of its neighbors. To speed things up LifeMaker uses a look up table to store the new states. The indices into the table are the bit patterns that correspond to the states of the cell and its neighbors. So the first entry in the table is the state a cell will change to if it and all its neighbors have state zero. The table sizes for the VonNeumann and Margolus neighborhoods are 1024 bytes. The table size for the Moore neighborhood is a whopping 262,144 bytes.

When you select the “Run” command LifeMaker uses the current rule to build the look up table. It does this by compiling the rule into machine language and calling the result once for every entry in the table. Moore rules are by far the slowest: they average 4-6 seconds on my SE/30. (If you’re is interested in the code generated by the compiler type Command-Option-D before the “Run” command).

In appearance LifeMaker rules bear a strong resemblance to Modula-2 (or, to a somewhat lesser extent, Pascal). There are significant differences however. The differences can be summarized as follows:

1) There are only three statement types: assignment, IF, and the Return statement.

2) INTEGER is the only variable type. There are three types of operators: arithmetic (+, -, *, /) comparison (=, <>, <, > , <=, >=) and bitwise (AND, OR, XOR). An expression that evaluates to a non-zero result is considered to be TRUE. Zero is FALSE.

3) Comparison operators evaluate to 1 if they are TRUE and 0 if they are FALSE (e.g. 5 > 3 = 1). The comparisons used are the unsigned comparisons so that -1 < 10 will evaluate to zero (the less then op treats negative numbers as very large positive numbers).

4) Individual bits can be accessed using brackets. For example to determine if a number is odd we can look at the low bit: isOdd := numb[0].

5) Constants may be written in hexadecimal if preceded by a dollar sign (e.g. $F = 15). Constants preceded by a percent sign are assumed to be in binary.

To get a feel for how rules are written consider the GreenBerg rule:

```
RULE GreenBerg (center, north, south, west, east)
 VAR new
BEGIN
  IF center THEN
    new := 0
  ELSE
    new := north[0] OR south[0] OR east[0] OR west[0]
```

```
  END
  IF center[0] THEN new := new + 2 END    (* sets high bit *)
  RETURN new
END
```

The first line of the rule contains the rules name and an optional list of the *implicit variables*. These correspond to the cells in the neighborhood. Their values range from 0 to 3. The next line declares the user variables. These can be used for any purpose and should be positive. The following lines are the body of the rule. The cells neighbors are used to calculate the new state for the center cell. The third line from the bottom is used to provide a little more pizzazz to the display. It sets the high bit of the new state if the center cell was non-zero.

The Margolus neigborhood includes one more variable Rand. By using Rand you can create *probalistic* rules. Rand randomly returns either zero or one. The number returned is the same for each cell in the block. As an example the following rule will send a particle (State 1 say) on a random walk:

```
RULE RandomWalk (cell, opp, cw, ccw)
  VAR new
BEGIN
  IF rand THEN              (* Returns 0 or 1 *)
       new := cw            (* Each cell in the block is rotated cw *)
  ELSE
       new := ccw           (* or ccw *)
  END
  RETURN new
END
```

## V. Rule Syntax

The syntax for rules is defined below in Backus-Naur form. If you’re not familiar with BNF notation it should become clear on examination.

```
Rule := Header Vars? 'BEGIN' stateList 'END'

Header := 'RULE' name Implicit?

Implicit := '(' VarList ')'

Vars := 'VAR' VarList

VarList := name (',' name)*

stateList := statement*

statement := (assignment | If | Return)

assignment := name ':=' expr

Return := 'RETURN' expr
```

```
If := 'IF' expr 'THEN'
                    stateList
              ('ELSIF' expr 'THEN'
                    stateList)*
              ('ELSE'
                    stateList)?
              'END'

expr := subExpr ((bitOp | compareOp) subExpr)*

subExpr := term (addOp term)*

term := factor (multOp factor)*

factor := variable | const | ’RAND’ | '(' expr ')'

variable := name ('[' const ']')?

multOp := '*' | '/'

addOp := '+' | '-'

bitOp := 'AND' | 'OR' | 'XOR'

compareOp := '=' | '<>' | '<' | '>' | '<=' | '>='

name := letter (letter | digit)*

const := base? digit digit*

base = '$' | '%'


Where
    |  means OR
    ?  means optional
    *  means repeat zero or more times
    '' is used for string literals
    () is used for grouping
```

## VI. The Digital Rule

One of the best rules in LifeMaker models digital logic. The rule is derived from Toffoli [15] and uses the Margolus neighborhood. State zero can be considered as an insulator. State one represents wire. State two is used to construct gates. And state three represents a signal on a wire.

Signals follow wires: a signal in a block containing two or three wires will travel to the other wires (see figure 3). A block containing four pieces of wire is special: signals will travel diagonally so signals can cross without interfering with each other.

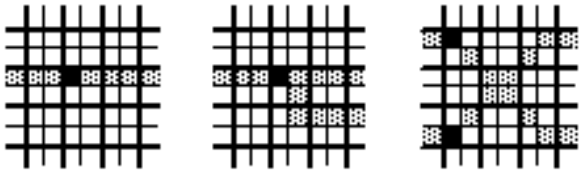

**Figure 3:** Signal propagation, fanout, and cross-over

A logic NOT is constructed using a cell of state two sitting next to a wire. Signals will be complemented as they pass through the block. An AND gate is represented by three pieces of wire and a cell of state two in a block (see figure 4). OR gates can be constructed using three NAND gates. As the circuitry gets more complex timing becomes more important and some wires may have to be lengthened or shortened to synchronize signals.

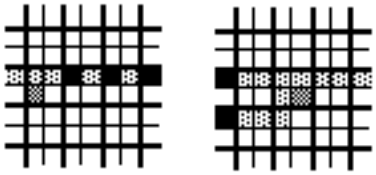

**Figure 4:** A NOT gate and an AND gate.

## VII. Using LifeMaker

The main window shows all the cells in the automaton. The cells are arranged in a torus (or doughnut) to avoid edge problems. Initially the array of cells is set to 320x200. The user can zoom in or out, change the window size, scroll about, change the colors, and change the number of cells.

The rule window is used to write the rules defining how the cells evolve. All the basic editing operations are supported along with Undo. Triple clicking selects entire lines. Changes are automatically saved.

The states window displays the colors used for the four states. To change a color simply double click on it. The color with a white box around it is used while drawing. To select a new color click on it. Holding the command key down and pressing the numbers 0 through 3 will also select a new color.

The tools window works just like in a paint program. There are a few differences however: instead of toggling cells the pencil sets cells to the selected state and the eraser sets cells to state zero. All drawing operations can also be undone. As a shortcut holding the command key down temporarily selects the selection tool, the option key selects the magnifying glass, the shift key selects the eraser, and the control key selects the pencil.

Advanced users can use ResEdit to customize various features of LifeMaker. For example to change the default state colors edit the ‘clut’ resource. To change the default size of the main window edit the ‘WIND’ resource named “main”. When “Create Rule” is selected the rule window is set to a default skeleton rule. The skeleton rules can be changed be editing the appropriate ‘TEXT’ resource.

## VIII. Menu Commands

# Apple menu

**About LifeMaker**
The about box has the version number and my address.

**DA's**
The usual list of desk accessories.

# File menu

**New**
Clears the cells to state zero and sets the window name to “Untitled”.

**Open**
Loads the cell states, rule, window positions, etc.

**Save**
Saves all the data defining the current model.

**Save As**
Allows you to save the current model under a different name. It will also allow you to save a selection in color (PICT2 format) or in black and white (using patterns).

**Revert File**
Reverts the model to the last save.

**Print**
If the rule window is in front the rule is printed using draft quality. Otherwise the whole screen is dumped to the printer (which won’t work without special software on a color Mac). This has been disabled since it's never worked real well.

**Quit**
Exits the program.

## Edit menu

**Undo**
Allows you to undo or redo any Edit or Transform command. It will also undo dragging and drawing.

**Cut**
**Copy**
**Paste**
**Clear**
The standard clipboard commands.

**Select All**
Selects all the cells.

## Transform menu

**Rotate Left**
**Rotate Right**
Rotates the selection by 90 degrees.

**Flip Horz**
**Flip Vert**
Flips the selection.

**Random**
Replaces the current selection by a random pattern.

**Density**
A sub-menu that controls how many cells are set to a random value. The default value for density is 100% (i.e. every pixel is randomized).

**Randomize**
A sub-menu that controls which bit is randomized. The default is both bits.

**Fill**
Allows the user to fill the selection with any state.

**Change**
Lets the user change any state to another. For example in the digital rule signals can be removed by changing state three to state one.

**Invert**
Inverts the selection.

## View menu

**Show/Hide/Select Time**
**Show/Hide/Select Tools**
**Show/Hide/Select States**
**Show/Hide/Select Rule**
If the window is the frontmost window its menu command will let you hide the window. If its visible but not in front the command changes to select so you can bring it to the front. If the window is hidden the command changes to show so you can make it visible.

**Zoom In**
**Zoom Out**
These commands let you zoom in on the cell array. The maximum zoom size is 8x8. As a shortcut you can press the 1, 2, 4, and 8 keys to immediately zoom in or out to that size. The window will be centered to the mouse position.

**Arrange**
This command rearranges the windows so that the main window is as large as possible and the other windows are all visible.

## Control menu

**Run**
Starts the CA.

**Run Sparse**
This works only in the Moore neighborhood. It allows CA with a small number of active rows to be run quickly. Unlike the other commands Run Sparse doesn’t wrap around. This way you can run Life quickly and not have to worry about gliders coming back and interfering with the other cells.

**Run Until**
This command allows you to specify a generation count at which LifeMaker will stop.

**Stop**
Suspends the CA.

**Speed**
This is a sub-menu that controls how fast new generations are displayed. Single step pauses each generation until the space bar is hit.

**Start Recording**
Records the cellular automata as a QuickTime movie. Recording will stop when the cellular automata is stopped or Stop Recording is selected.

**Stop Recording**
Stops QuickTime recording.

**Reset Time**
Clears the generation count to zero.

## Space menu

**Von Neumann**
**Moore**
**Margolus**
These are sub-menus that hold all of LifeMaker's rules. You can select a rule, delete a rule, or create a new rule.

**Number of Cells**
This allows you to change the number of cells.

**Color**
This is another way to change the colors of all the states. You can also use this to restore the colors to the default values.

**Show/Hide Blocks**

This is used to outline the blocks in the Margolus neighborhood. For obvious reasons it works only when the cells are zoomed to 4x4 or 8x8. It’s useful for understanding how rules work in the Margolus space (e.g. Digital).

## Patterns menu

This menu changes whenever a new rule is selected. It shows all the patterns defined for the current rule. Patterns may be added or deleted.

## IX. Known Bugs

LifeMaker doesn’t work with version 1.0 of 32 Bit QuickDraw. Redo Erase also occasionally misses a few cells.

## X. Changes

Version 2.1

• Fixed a silly bug that caused crashes on some machines when LifeMaker started up.
• Fixed a bug that caused Undo Eraser to crash occasionally.
• Flushes the data cache after compiling a rule so LifeMaker works reliably on '040 machines.
• Added support for recording the CA as a QuickTime movie.
• The main window can no longer be hidden or closed.
• Added balloon help to all of the menus and most of the dialogs.
• If the main monitor is black and white or gray-scale the windows will appear on a color monitor if one is available.
• The Arrange command now really does make the main window as large as possible.
• The 1, 2, 4, and 8 keys zoom the main window in and out. The window will be centered on the mouse's position.
• The selection can now be dragged while the selection tool is active.
• Single-step now works with either space-bar or command-space-bar.
• Disabled the Print command.
• Added HighLife rule.

Version 2.0

• Added the Rand keyword to Margolus rules.
• Added the very neat probalistic rules Brownian and Dendrite.
• Density sub-menu now includes an “Other” item so the user can specify exactly how many cells he wants to randomize.
• Fixed a bug that messed up wrap-around in the Margolus space.

Version 2.0b3

• The rule compiler has been added so that completely general rules can be added.

• When Run Until finishes SysBeep is called. If LifeMaker is running in the background the Notification Manager is used to alert the user.

• Gestalt is now used to get information about the users machine. In practice the only effect is that if 32 Bit QuickDraw 1.0 or earlier is being used LifeMaker will quit with an explanatory message (instead of crashing).

• The Change States command now uses a single dialog.

• The command key is now required.

• The ‘DFLT’ resource is no longer used. Instead change the ‘WIND’ resources and the ‘clut’ resource.

## XI. References


[1] Algorithm, PO Box 29237, Westmount Postal Outlet, 785 Wonderland Road, London, Ontario, Canada N6K, 1M6. This is a small magazine edited by Dewdney of Computer Recreations fame. It covers the same sort of thing as his column in Scientific American.

[2] CODD, E.F. Cellular Automata Academic Press. This book is a good introduction to CA from a computer science perspective.

[3] DEMONGEOT, J. and E. GOLES and M. TSHEUNTE editors. Dynamical Systems and Cellular Automata Academic Press. ISBN 0-12-209060-8. Lots of articles on applications including growth, complex systems, and systolic arrays.

[4] DEWDNEY, A.K. The Armchair Universe W.H. Freeman and Co. ISBN 0-7167-1938-X. This is a collection of Dewdney’s “Computer Recreations” articles in Scientific American. He covers the Mandelbrot Set, hypercubes, a program for creating caricatures of people called FaceBender, a 3D variant of Life, core wars, and more.

[5] FREDKIN, Edward and Tommaso TOFFOLI, “Conservative Logic”, Int. J. Theor. Phys. 21 (1982). Introduced the billiard-ball reversible CA computer.

[6] GARDNER, Martin. Wheels, Life, and other Mathematical Amusements W.H. Freeman and Co. ISBN 0-7167-1588-0. This is a collection of Gardner's "Mathematical Games" articles in Scientific American. He covers a wide variety of interesting mathematical puzzles and games including three chapters on Life.

[7] KADANOFF, Leo. "On two Levels", Physics Today 39:9 (September 1986) Discusses the theoretical and practical advantages to using CA instead of numerical methods for solving the Navier-Stokes equation.

[8] MARGOLUS, Norman. "Physics-like models of computation", Pysica 10D (1984). Discuses reversible compute-universal CA.

[9] MCCANNY, John and John MCWHIRTER and Earl SWARTZLANDER editors. Systolic Array Processors Simon & Schuster. ISBN 0-13-473422-X. This is a big book with a multitude of specialized systolic array processors.

[10] Media Magic POB 507 Nicasio, CA, 94946, Phone (415) 662-2426. I got one of their catalogs in the mail a while ago. They look like a good source for books and videos on chaos, fractals, cellular automata and the like. They also sell postcards, t-shirts, and a few posters.

[11] PICKOVER, Clifford. Computers, Pattern, Chaos and Beauty St. Martins Press. ISBN 0-312-04123-3. This is a really great book. Pickover presents a wide range of applications for computer graphics. He covers Fourier series, image processing, fractals, chaos, spirals, and lots more. There are gorgeous pictures on nearly every page and Pickover includes algorithms for almost every topic he discusses.

[12] RIETMAN, Edward. Exploring the Geometry of Nature Windcrest Books. ISBN 0-8306-9137-5. This is an excellent intermediate introduction to chaos, strange attractors, and fractals. It's aimed at readers with a background in calculus and he's included a bunch of Basic programs.

[13] SALEM, James and Stephen WOLFRAM. "Thermodynamics and Hydrodynamics of Cellular Automata", Theory and Applications of Cellular Automata World Scientific. Describes an experiment using a Connection Machine to model fluid flow.

[14] SOLOMAA, Arto. Computation and Automata Cambridge University Press. ISBN 0-521-30245-5. Covers Turing machines, the halting problem, computational complexity, and cryptography.

[15] TOFFOLI, Tommaso and Norman MARGOLUS. Cellular Automata Machines The MIT Press. ISBN 0-262-20060-0. This book was the inspiration for LifeMaker. It's a great introduction to using and constructing cellular automata. The authors present the rules for a lot of different CA. Unfortunately they used Forth to write the rules. (They do have a small appendix with a Forth tutorial).

[16] WOLFRAM, Stephen editor. Theory and Applications of Cellular Automata World Scientific. Excellent book with a number of interesting articles. Includes an extensive bibliography.