Introduction to OSXLCAU21 System

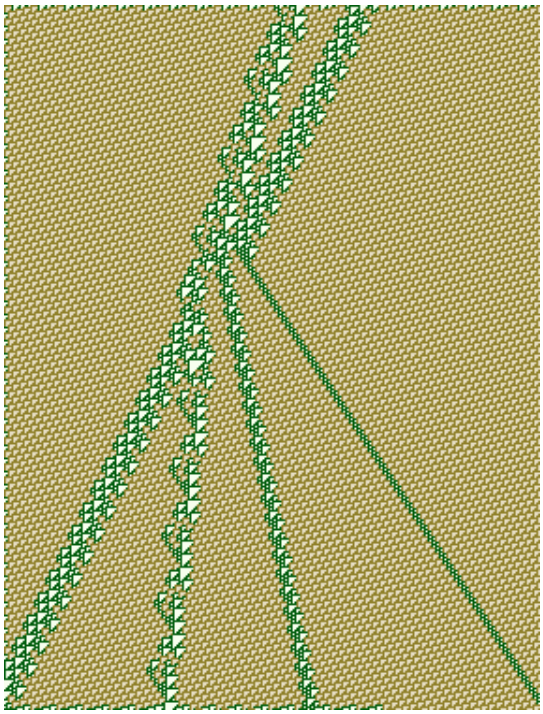
Genaro Juárez Martínez
genarojm@correo.unam.mx

Rule 110 Winter WorkShop
Bielefeld, Germany

March 2004

Abstract

This paper describes each one of the parts of OSXLCAU21 system, the main feature of this application is the panel of gliders which allows to reproduce any binary collisions among them. The system is implemented with the idea of providing an useful and easy enviroment to the user for studying and realizing experimentation in the analysis of Rule 110.



1 OSXLCAU21 System

The OSXLCAU21 system is developed in order to satisfy the necessity of a detailed study in the evolution space of Rule 110, this system is coded in C-Objetive and at the moment it is available for OPENSTEP, Mac OS X and Windows operating systems (the version for Windows needs the installation of the WebObjects system for Windows).

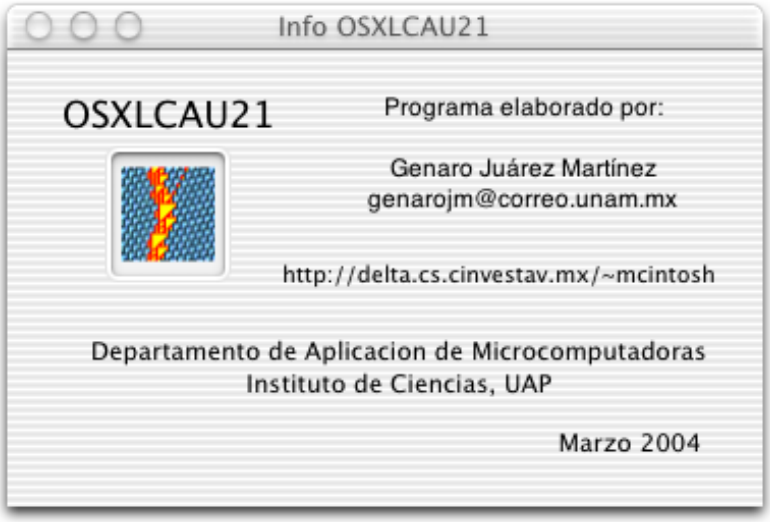


Figure 1: OSXLCAU21 System

The use of the OSXLCAU21 system is not complicated, the interface is dominated by buttons and windows. The system is able to work with all the evolution rules of order $(2,1)$. Nevertheless the handling of gliders is only for Rule 110; the system is still in developing process but it suitably works for illustrating the use of the phases and reproduce several complex behaviors.

Next we describe the facilities offered by the system and the operations that can be realized with them. The three versions of the system use the same original environment of OPENSTEP. The system and its source code are of public domain and can be freely downloaded from [3].

2 *Main panel OSXLCAU21* window

The *Main panel OSXLCAU21* window is the one which controls most of the actions of the system and provides useful information. The NSTextField object *evolution rule* allows to edit directly the evolution rule that we desire to visualize; the rule is introduced in decimal notation, taking values from 0 up to 255. The NSSlider object *density* allows to define the density of the configuration initial, a density close to zero means that state 0 will occupy most of the cells in the

configuration initial, in the opposite case state 1 dominates. This density can be directly introduced in its NSTextField whether an exact density is desired.

For instance, Rule 110 finds its statistical stability approximately in 0.57, hence this value can be directly introduced in the initial density.

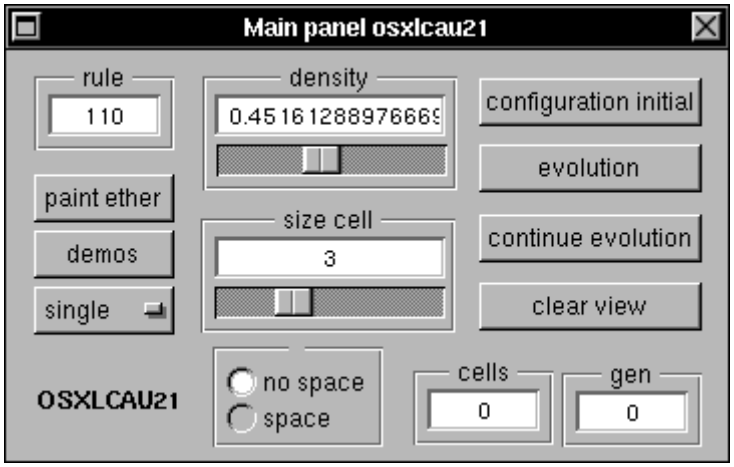


Figure 2: Panel main

The object *size cell* allows to define different sizes in which the cells can be seen in the evolution space, the rank of the different sizes goes from 1 up to 10. This feature is useful when we desire a better visualization of both a particular collision or a given structure.

The object *cells* shows the number of cells in the configuration initial and the object *gen* presents the number of calculated generations, these values are updated whenever the user defines a new size in the window presenting the evolution.

The NSButton object *configuration initial* assigns a random initial configuration according to the density and the colors defined at this time. The button *evolution* calculates and shows the evolution from the currently defined initial configuration. The evolution is drawn in each line.

The button *continue evolution* calculates and shows the continuation of the previous evolution from the last generation of the first evolution. Each time we continue with the evolution, the number of generations is also updated in the NSTextField *gen*. The button *clear view* cleans the evolution space and initializes all the variables of the system. The button *paint ether* colors the T_3 mosaic in other colors defined by the user, this feature is useful to get a better visualization of single gliders or complex collisions between several gliders against the periodic background.

The NSMatrix objects *space* and *no-space* establish a small division between each one of the cells, this is useful when the user wants a detailed view of the number of cells involved in a given sequence or region.

3 *Evolution* window

The evolution space is controlled by a NSView object assigned to the *Evolution* window, the window is resizable over all the screen. One current and important problem in the system is that the initial configuration is limited by the resolution of the screen.

Another relevant detail is that the system does not determine the size of the configuration introduced by the user, producing undesired structures in the limits of the configuration. In order to avoid these undesired structures in the evolution, one alternative is to fit manually the width of the *Evolution* window so that the regular expression defining *ether* finishes precisely in the last cell.

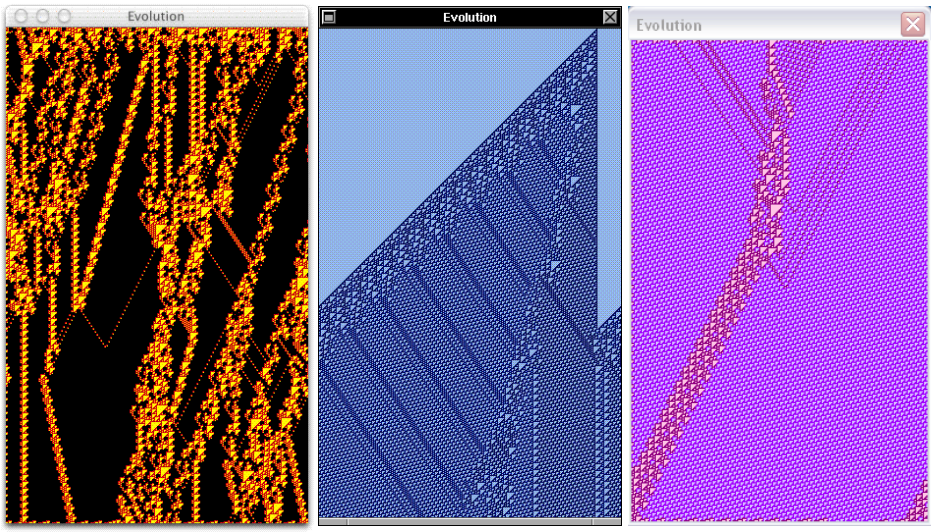


Figure 3: Evolution space

Figure 3 shows three evolutions in Rule 110, the first illustrates the evolution from a random configuration, in this example we have applied different colors to *ether* in order to get a better identification of the gliders or simply to obtain an attractive figure.

The second figure illustrates the evolution of a cell with state 1, this example is interesting because if the space is sufficiently large, it is possible to see how the produced margin also generates gliders gun, and in four thousand generations the evolution becomes periodic.

The third figure in a special case illustrating the construction of $\bar{B}^2$ glider. Two *ether* configurations are assigned using the panel of gliders, then an A glider in phase 2 is specified. The location of gliders and phases in the panel of gliders is by means of coordinates, first the user looks for the row of the A glider, then the user takes the column corresponding with phase 2 and this selection assigns the desired sequence, this process is used to yield the whole expression. In this

example it is necessary to extract sequences which are not specified in the panel of gliders, for instance the group of A and B gliders must be assigned from the chain console. The reason of this problem is because the regular expressions are coded for a single glider and not for groups or extensions of them. In this case we have another problem due to the unlimited number of times in which they can be grouped or extended.

4 *Matthew Cook Gliders* window

The main feature of this program is the panel of phases for each glider in Rule 110, this panel corresponds with the classification established by Matthew Cook. If we want to introduce a particular phase, we just need to click in the button of the desired phase and the adequate fragment of configuration is immediately assigned and drawn in the *Evolution* window. This panel contains all the phases f_i_1 for every glider including the glider gun.

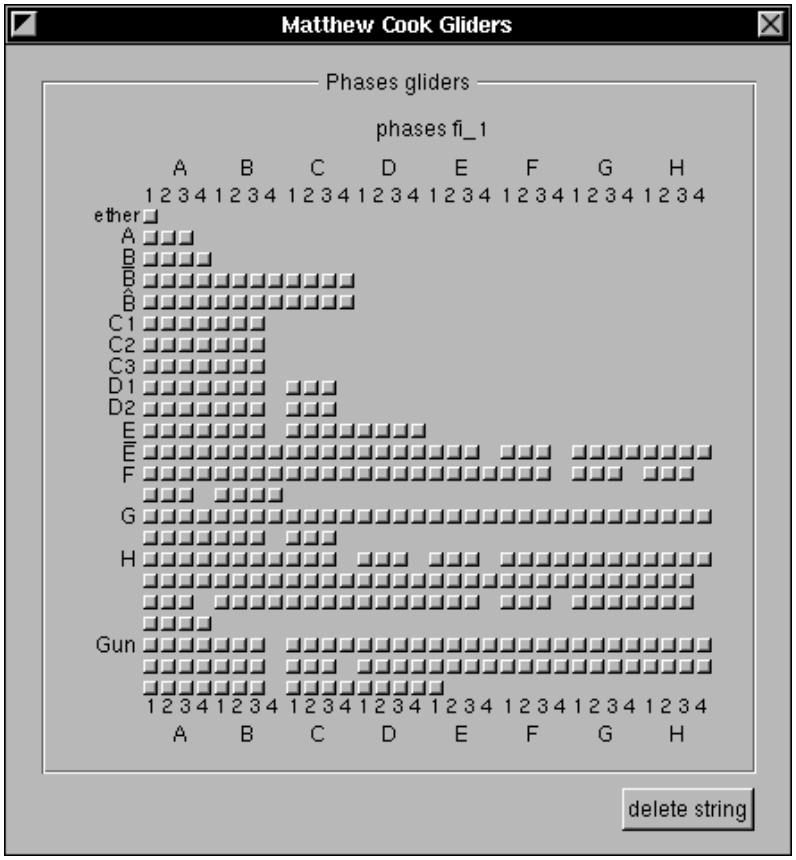


Figure 4: Panel of glider phases

Sometimes we can make a mistake at the moment of introducing the phase, in order to solve this problem we use a *delete string* button which deletes the last introduced chain in the configuration. One problem here is that the system does not save the constructed configuration.

5 *Color Panel* window

The colors for the states of the cellular automaton may be controlled with the *Color Panel* window, when we desire to paint ether using different colors we can define them for each state of the T_3 mosaic. The color wells invoke automatically to the *Colors* panel when we click them, this feature is a unique advantage of the NeXTSTEP system.

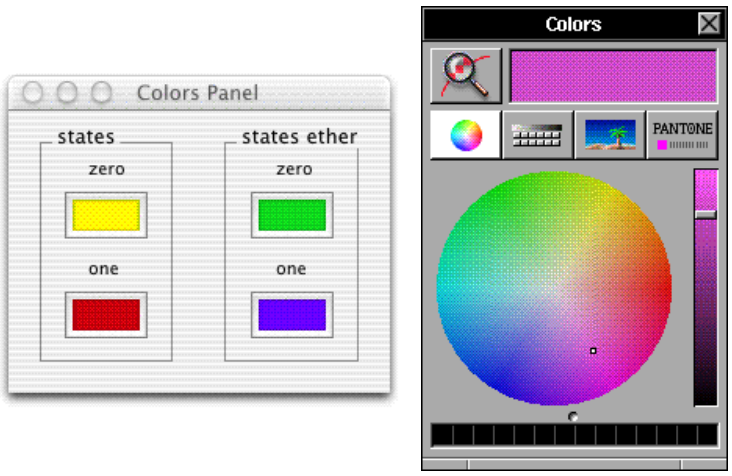


Figure 5: Panel's of colors

There is a wide range of color combinations offered by this panel, including both the PANTONE and PANTONE Process classification in the case of NeXTSTEP and OPENSTEP systems, which in addition have all their outputs in PostScript.

6 *Console Strings* window

At the time of experimenting the construction of certain configurations, we have to define chains which require precise manipulation, for example groups of gliders which cannot be constructed neither by means of phases or extensions nor from the configuration initial.

In order to solve this problem there is a small console which is able to receive a particular chain and assign it to the initial configuration clicking the corresponding button.

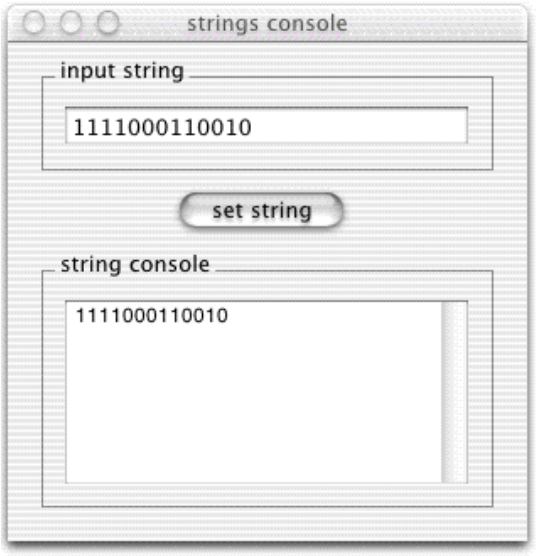


Figure 6: Panel of strings

Figure 6 shows the chain console, it has a simple form but it is very useful as well. The desired chain is introduced in the object *input string*, this one can support up to 1500 elements. Once we have typed the chain, the user must press the button *set string*, then the chain is presented in the lower part of the console to verify if it has been correctly interpreted and the chain is immediately drawn in the evolution space. If the introduced chain is not the desired one for some reason, the user can return to the *Matthew Cook Gliders* panel and delete the chain with the button *delete string*.

7 Conclusions

In conclusion the OSXLCAU21 system has yet several limitations such as:

- Controlling the evolution and stopping it at any desired moment.
- Creating a BrowserView which allows to manipulate evolutions of larger dimensions.
- Editing directly some wished configuration in the evolution space or if the space is full, then we can obtain an automatic update of the evolution with regard of the new introduced cell [1].
- Creating zoom-in and zoom-out features of visualization.
- Introducing matrix tools for making analysis of graph theory and probabilities [2].

Acknowledgements

In special to professor Harold V. McIntosh, and Department of Microcomputers in UAP. To Martin Schneider, Konrad Diwold and to Rule 110 Winter WorkShop by the hospitality and support. To Juan Carlos Seck Tuoh Mora by his aid and comments. This work was partially supported by CONACyT with registry number 139509.

References

[1] Applet elaborated by Matthew Cook updating the evolution space in all directions when a cell is introduced in Rule 110, http://www.paradise.caltech.edu/~cook/Workshop/Java/Sponge/index.html, December 2002.

[2] Collection of programs *NXLCAU* elaborated by Harold V. McIntosh for one-dimensional cellular automata of different order, plataforms NeXTSTEP/OPENSTEP and MS-Dos. Aplication and code sourse are of public domain available in: http://delta.cs.cinvestav.mx/~mcintosh/oldweb/software.html, 1995.

[3] Program of public domain *OSXLCAU21* available for OPENSTEP, Mac OS X and Windows operating systems. Application and code source available in: http://delta.cs.cinvestav.mx/~mcintosh/comun/ s2001/s2001.html, August 2001; and http://www.rule110.org/downloads/.