

A Method for the Use of Patterns Drawn from Cellular Automata as Generative Source for Discrete Virtual Environments

Gottin, V. M. Schardong, G.G. Felipetto, C.M. Pozzer, C.T.
Universidade Federal de Santa Maria, LaCA – UFSM, Brazil



Figure 1. A graphical representation of the method proposed. The patterns in a cellular automaton (a) are translated through a set of rules (b) into generative data structures, here depicted graphically (c) containing information for a rendering process to generate an environment (d).

Abstract

This paper proposes a method for creating discrete virtual environments using recognition of patterns from Cellular Automata. The application of rules over the recognized patterns generates the data structures describing the virtual environment. The steps for implementation are described and a case study is presented. Results are analyzed and future works for harnessing and developing the method are proposed.

Keywords: Cellular Automata, environment generation, discrete environments;

Author's contact: {vmgottin, schardon, felipeto, pozzer} @ inf.ufsm.br

1. Introduction

Cellular Automata (CA) have been widely explored due to their intrinsic characteristic of simulating complex behaviors parting from sets of simple rules [Wolfram 2002]. This paper proposes a method for the creation of environments through a composition of discrete components derived from patterns in a cellular automaton.

An application of the proposed method consists of the translation of recognized patterns in a cellular automaton's configuration into environment components aggregated into a single environment. As an application based on CA, it might benefit from some characteristics that enable performance gain.

Additionally, this paper explores the possibility of creating environments capable of 'evolution' following the logical rules of the cellular automaton it is based on. Hence, this paper brings two original contributions: a new procedural generation method for creation of content for tile-based games and a new proposal for simulation of topographic evolution.

2. Related Work

CA have been widely studied and much remains to be explored. The interpretation of patterns in CA is a field

of study all by itself, being the subject of many studies [Chou 1996]. Use of CA for systems simulation is also the subject of many researches [Lima 2005], for physical systems containing many discrete elements with local interactions are often conveniently modeled as CA [Wolfram 1983].

The proposed method aims to generate discrete environments through the creation of aggregated discrete environment components, thus fitting into that description. Not much has been specifically done about the creation of virtual environments utilizing CA, save for very specific components for non-discrete environments, like the approach presented by Judice et al. [2008].

3. Cellular Automata

CA are models of complex systems in which an infinite lattice of finite state machines updates itself in parallel according to an isotropic local rule [Culik et al. 1990]. Development of methods, techniques and algorithms relating CA to other areas [Wolfram 2002] are a result of the interest and possibilities over the characteristics of CA: complex behavior through the application of simple rules [von Neumann 1966]; and the possibility of viewing the evolution of a system like a cellular automaton as a computation. Sipper [1999] gives various examples of applications using CA that show the three characteristics of this paradigm: simplicity (each cell can do very little work separately, leaving a very narrow margin for error), vast parallelism (vast numbers of cells operating in parallel) and locality (connections between cells are local).

Thus, it is proposed that the parallelism and the possibility of performance enhancement make applications based on CA a viable alternative to pre or randomly generated environments. Nevertheless, specific statistical research over the proposed method is needed.

4. The Harp Method

The objectives and application of *Harp* significantly differ from other techniques of virtual environment generation [Mandelbrot 1982; Perlin 1985] that are somewhat focused on real-time processing of content. While these techniques aim to render real-like terrain, *Harp* focuses on creating data structures over which an environment may be rendered. The rendering step is also not a part of *Harp*. In the case study, Civilization IV's engine was used for the rendering of a discrete environment.

The choice over a particular cellular automaton for a fine-tuned implementation of *Harp* must take into consideration the dimension, rules, cellular states and

structure permanence of the automaton, as in the classes defined by Wolfram [2002]. Parameters for this choice would require further research, not in the scope of this proposal and, as such, throughout the article all examples – including the case study - of the application of *Harp* will be presented utilizing a *Conway's Life* [Gardner 1970] implementation, one of the most extensively explored CA¹.

The first step of *Harp* depends entirely on the chosen cellular automaton. The step is very simple, consisting on capturing the configuration of this automaton and outputting it for Step Two: recognition of patterns in a given generation of the chosen cellular automaton. The implementation of the pattern recognition algorithm used in the case study is formalized as follows: given a space D of any dimension and a pattern with a set of live cells $L\{x_1, x_2, \dots, x_p, \dots, x_n\}$, with n live cells and x_i being the position in D of the i -th live cell in the set; then the algorithm will recognize this pattern in D if:

- A set of live cells L' exist in D and;
- For every live cell i' in L' with a position $x_{i'}$ in D there is a live cell i in L with a position x_i equal to $x_{i'}$;
- The pattern configured by L presents the same immediate neighborhood of dead cells as L' .

In the case of the possibilities explored above, the pattern recognition algorithm returns as a result a string of patterns with some information attached: its type, its position X (i.e., the set of positions x of all live cells and immediate neighborhood dead cells) on D and, through simple interpretation of its type, the number of live cells n composing it. This will, for the rest of this paper, be referenced as the *generative information* of a pattern.

The third and fourth steps in *Harp* are tightly connected. Their objective is to apply a set of transformation rules over a set of generative information, derived from the pattern recognition in the previous step, to obtain environment components.

In the formalization of Step Three, a single rule is described as R part of a set S of rules, and each input item p is part of the set including all input P . Every item in P has a type p_j , with j being an element descriptive of its nature. As seen in Figure 2, each rule R in S is an effect f (Figure 2.a) over a single type of input item j – in this case, the type is correspondent to a pattern type, a block (Figure 2.b). Here, the algorithm for environment components creation will compare each rule in S with each item in P (Figure 2.c) and, whenever the type j of f is identical to p_j , the rule (the effect) will be applied over the item p (Figure 2.e expresses the output of this step).

Formally, a rule in a transformation method may be expressed as the mathematical model of strings and sequences of symbols as used in the theory of finite-

state automata [Roche and Schabes 1997]. In this formalization, the input information of *Harp*, a sequence of patterns, forms a string. Each symbol composing a string, in this case, is one input item of Step Three – a pattern and the information it contains, as seen in the previous section. The alphabet over which such strings can be built is, in this case, derived from the patterns recognized by the method in Step Two.

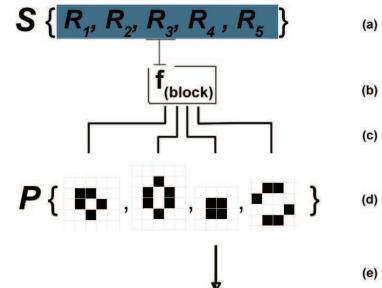


Figure 2. Each rule corresponds to a single effect over a single type of item. Each rule (i.e., each effect) is compared to each input item (here depicted as the graphical patterns in P) and if their types are a match, then the effect f is applied in Step Four.

Three conclusions may be drawn from this formalization. First we additionally define, now within the formal model proposed, that every rule R in S must have an effect f related to every input item in the *alphabet*, and not in the input set P – i.e., one effect f related to every symbol in the alphabet. Second, the formal notation allows the creation of a simple standard for the notation of these rules – now considered symbols in an alphabet or characters in a string – which will be addressed later. Finally, as a last consequence of this formalization, it may be possible to implement operations like union, intersection, subtraction and inversion on a string that is a set of rules.

Step Four comprises of the creation of the data structures that will be rendered into a virtual environment for any kind of visual output. The output in the example would be a data structure with enough information to describe a grid with determined colors in determined spots. This information will be denominated *generative data*, in order to differentiate it from the *generative information*.

Step Four depends in great measure of the desired characteristics of the environment: its dimension, the level of detail, the rate of changes necessary for the environments simulation and anything else that the designer(s) consider necessary. This will be henceforth referred as the *design criteria*. First, though, an example of the output of Step Three, input to Step Four, is shown in Figure 3.

It is noteworthy that according to the rules of transformation applied in Step Three, effects are applied over the generative information, and that may not include its position in the dimension of the cellular automaton. In the example, the number of live cells in

¹ [online] Conway's Game of Life at the Open Directory Project: http://www.dmoz.org/Computers/Artificial_Life/Cellular_Automata/Conway%27s_Game_of_Life/

the pattern is used to determine the position of the components created.

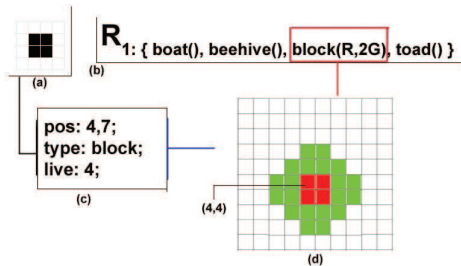


Figure 3 In this example, the pattern in the cellular automaton (a) translated through a rule (b) that changes its information (c) as in the previous example. The position of the component in the virtual environment generated is given by the coordinate of (n,n) , with n being the number of live cells in the pattern. The position of the pattern in the automaton is unused information.

In conclusion, Step Four will output a generative data structure that contains information for a rendering technique to act. This rendering technique is not part of *Harp*, and in the following case study, a commercial game's scenario editor was utilized.

5. Case Study

In this section, a case study of *Harp*'s implementation is exposed. This study's design criteria is to test the application of *Harp* exposing in detail the results of each step of the method, meanwhile generating an environment with some coherence. The Civilization IV game engine was responsible for rendering the information in visual output. The approach used for this was to save the generative data outputted from *Harp* into a file readable by the engine. This was done through a C++ implementation intertwined with the *Life* implementation mentioned earlier.

Some adjustments to meet design criteria were experienced as part of the study. First, the set of rules was extended so that each pattern recognizes nearby patterns and include that information as part of its generative information. That way, not all generative data descendant from blocks in the automaton is the same. Without this, all landmasses generated from the same pattern would be identical. An example of this is shown in Figure 4.



Figure 4. An application of *Harp* where the generative information is based solely on the position of the pattern in the automaton generates several identical islands (a). When each pattern is affected by nearby patterns, more detailed and differentiated landmasses are generated, such as the one depicted in (b).

Second, a set of more composite rules - in which terrain features like hills were generated by the overlapping of patterns - was then developed to

conform the output of *Harp* to the design criteria. An example is shown in Figure 5.



Figure 5. An example of terrain generated by the overlapping of generative data. Here, the hills were created as a result of the overlapping of block and blinker pattern's generative data.

Third, the use of multiple levels of *Harp* in the same environment was part of the experiment. Figure 6 shows a graphical representation of the concept, where a second cellular automaton (possibly with a new set of rules) has its patterns applied over a pre-existing environment - which was also generated with *Harp*. Figure 7 shows the results of utilizing a stance of *Life* to generate forests and jungle over the terrain depicted in Figure 5. This was accomplished by operating an intersection over the generative data: a forest was created in the environment where a landmass was created by the first stance *and* a forest was created by the second stance. This was possible to express in the chosen syntax because the sets of rules were formalized as strings.

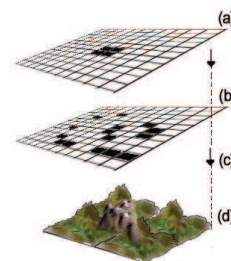


Figure 6. A representation of how a cellular automaton (a) outputs the data structures of its *Harp* application not for the rendering process, but for a second cellular automaton (b) that outputs for rendering (c) the information of the final terrain with multiple levels of detail (d).



Figure 7. In this example the landmasses are generated by one configuration of *Life* and the forests are generated by a second-level application of *Harp* with its own set of rules of transformation.

It was verified in these examples that yet more levels of application of *Harp* could be easily implemented, and that a concept of areas of interest might apply - i.e., the effects of a second-level automaton applying only over a desired part of the output of the first-level. A complete map with two second-level automata – one generating forests and the other, resources – acting over the landmasses produced by the first-level automaton is shown in Figure 8.



Figure 8. Screenshot of a map generated by the *Harp* method, using two second-level stances of *Harp* in addition to the first one that generates the landmasses.

A further objective of *Harp* is to develop the environment through the evolution (advancement of generation) of the CA over which it is based – basically, applying *Harp* iteratively. Since it is possible to predict the next state of the environment through the prediction of the automaton's next generation, it is possible to make this changes gradually appear in the environment, but this was not implemented and stands as the objective of current research. An example where the migration of forests in the scenario was created by subsequent generations of the same CA is shown in Figure 9.



Figure 9. Multiple levels of *Harp*. The first level creates the landmasses and a second level creates the forests through a *intersection* operation of the sets of rules.

6. Conclusions

In this paper two original contributions are presented: first, a new approach for the creation of content for tile-based games through the method for generation of environments called *Harp* - and, secondly, indication that this method might emulate scenario evolution in a credible manner if proper implementation needs are met.

The case study presented showed some characteristics of the method. The design criteria for this study were fully satisfied. Some proposed techniques for *Harp*'s implementation were successful as well: local influence of patterns over each other; fine adjustment of rules for better results; multiple levels of CA generating the same environment and, finally, evolution of the environment through changes in the automata.

Some ideas were not explored and remain as focus for future research. First and foremost, the effect of the chosen automata's characteristics over the final result of *Harp* and parameters for the choice of the CA for meeting the design criteria must be researched. As exposed throughout the paper, this choice affects the entire application, and therefore, should be a priority in future works.

Manipulation of rules through operations on sets of rules expressed as strings, like the *intersection* operation between two sets of rule exemplified in the case study, are also an interesting field for future work. Operations between rules of the same set have not been covered by this study and should be a natural extension of the method.

References

- CIVILIZATION IV, copyright 2005 Take-Two Interactive Software. <http://www.2kgames.com/civ4/>
- CHOU, H., 1996. *Self-Replicating Structures in a CA Space*. University of Maryland, USA.
- CULIK II, K., HURD, L. P. AND YU, S., 1990. Computation Theoretic Aspects of CA. In: *CA: Theory and Experiment*, H. Gutowitz, 1991.
- GARDNER, M. 1970. Mathematical Games – The Fantastic combinations of John Conway's new solitaire game "life".
- JUDICE, S. F., COUTINHO, B. B. S., GIRALDI, G. A., 2008. A CA Framework for Real Time Fluid Animation. National Laboratory for Scientific Computing.
- LIMA, E. B., 2007. *Modelos microscópicos para simulação do tráfego baseados em autômatos celulares*. Doctorate thesis, Universidade Federal Fluminense.
- MANDELBROT, B. B., 1982. *The Fractal Geometry of Nature*. W. H. Freeman and Company, New York, USA.
- PERLIN, K. An Image Synthesizer. Courant Institute of Mathematical Sciences, New York University.
- ROCHE, E., YVES, S. 1997. *Finite-State Language Processing*. MIT Press: Cambridge, Massachusetts, USA.
- SIPPER, M., 1999. The emergence of cellular computing. *IEEE computer*, v. 32, n. 7, p 18-26.
- VON NEUMANN, J. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966.
- WOLFRAM, S., 1983. *Statistical Mechanics of CA*. The Institute for Advanced Study, Princeton, USA.
- WOLFRAM, S., 2002. *A New Kind of Science*. Champaign: Wolfram Media.