Merging Cellular Automata for Simulating Surface Effects

Stéphane Gobron, Denis Finck, Philippe Even, Bertrand Kerautret

Merging Cellular Automata for Simulating Surface Effects

Stéphane Gobron[1], Denis Finck[1], Philippe Even[2], and Bertrand Kerautret[2]

[1] Université Henri Poincaré, IUT de St Dié
{Stephane.Gobron,Denis.Finck}@iutsd.uhp-nancy.fr
www.iutsd.uhp-nancy.fr/isn/StGo
[2] LORIA/ADAGIo – Université Henri Poincaré, IUT de St Dié,
11, rue de l'Université, F-88100 Saint-Dié-des-Vosges, France
{Philippe.Even,Bertrand.Kerautret}@loria.fr

Abstract. This paper describes a model of three-dimensional cellular automata allowing to simulate different phenomena in the fields of computer graphics and image processing, and to combine them together in order to produce complex effects such as automatic multitexturing, surface imperfections, or biological retina multi-layer cellular behaviours. Our cellular automaton model is defined as a network of connected cells arranged in a natural and dynamic way, which affords multi-behavior capabilities. Based on cheap and widespread computing systems, real-time performance can be reached for simulations involving up to a hundred thousand cells. Our approach efficiency is illustrated through a set of CA related to computer graphics –*e.g.* erosion, sedimentation, or vegetal growing processes– and image analysis –*e.g.* pipeline retina simulation.

Key words: cellular automata, geometric modelling, image processing, environmental and biological systems, surface effects, fluid simulation

1 Introduction

This paper presents a model for simultaneous and real-time simulations of surface effects. It refers to the field of Cellular Automata (CA) intended for 3D geometric modeling and image processing and applied to various purposes such as texture synthesis, surface imperfections or simulations of natural phenomena, or retina structure and behaviour simulations as well. We believe that CA are an acceptable solution to deal with the inherent complexity to that domain.

The complex behavior and the diversity that can be observed at a macroscopic level is essentially due to the fact that there are numerous particles with continual interactions. Of course, it is not possible to study all the phenomena coming back at the atomic level. We must consider the right level of abstraction.

Although computer *reality* consists of electrons moving through electronic components, their behaviour can be described in terms of electronics, then logic, bits and instructions, algorithms and data structure, languages, software engineering, etc. So when working at a given level, the upper level description can

lead to unify phenomena that were previously considered as individual cases. Therefore, a level likely to unify the study of surface effects is to consider the surface as a set of small pieces of materials interacting locally –taking into account its thickness, density, color, or elasticity.

Literature about computer graphical CA has become more and more active, especially since Graphics Processing Unit (GPU) became popular [4, 20] and since a fundamental book that covers all practical aspects of CA was published by Stephen Wolfram [22]. We mention hereafter some key steps of the CA evolution in this domain. Published in the mid-80s, Thalmann[17] is one of the first computer graphics references directly dealing with the CA issue. In the late 80s and early 90s, CA became quite popular in various fields of computer graphics studies [13, 14, 16, 7, 15]. At that time, a work on tumor growth simulation [5] initiated the use of CA for graphical interpretation in biology. In 1991, both Turk [19] and Witkin *et al.* [21] presented outstanding approaches for texturing; the main idea was to make a texture projection of a 2D reaction-diffusion CA function. In the late 90s, an approach of restricted hyper-texture CA [8] led to a new model, which was called *3DSCA* -for 3D Surface CA-. A series of surface simulations could be generated with this model, unfortunately with many restrictions. In 2003, Tran *et al.* [18] proposed a CA implemented on GPU, and Harris *et al.* [12, 11] presented even more complex simulations of natural behaviors (boiling fluids) using GPU programming.

Recently, very interesting results were obtained in generating realistic textures with surface imperfections including corrosion, weathered stone, impacts, scratches or even lichen growth. But most of those simulations were designed on a single purpose with a specific and restricted model. Actually most of the modeled effects require special *ad hoc* data structures. Our present goal is to improve the former *3DSCA* model in order to design a flexible approach, open to any type of multi-behavior surface simulation. This model basically relies on a dynamic data structure combined with CA rules which do not depend on the number of neighbours.

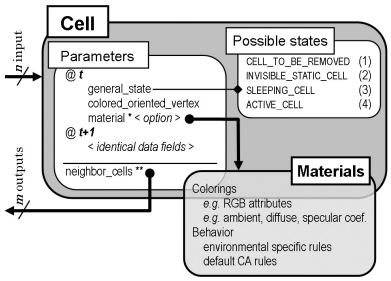
Section 2 describes the general architecture of our model: the generation of initial cells defining the notion of grid type, the way those cells are stored into a non-trivial geometric data structure, and a method for building a regular web of connected cells. Rules of applied cellular automata, design considerations, and corresponding results are detailed section 3. Finally, section 4 concludes this paper and puts forward some of the related future work emphasizing CA computations based on Graphics Processing Unit (GPU).

2 Merging cellular automata

In our approach a surface is a set of cells that are locally interacting together and with the environment of the object they are part of. We show that their limited but non null capabilities can be simulated using the discrete nature of cellular automata. A set of simple rules can produce a wide range of 3D surface simulations. So we define a geometric and dynamic cellular network that has the

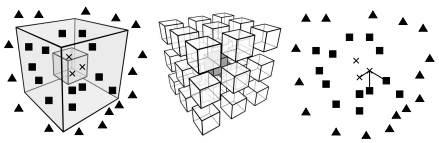
following advantages: no restriction is made on the input polygon type, any number of vertices, concavities, or holes being accepted; multiple input objects and natural linking between cells of different 3D sources are automatically managed; any type of grids -random, square, triangular or hexagonal- are accepted; cell movements are left possible in all directions; and the cellular structure has no limited number of neighborhood cells or communication types. Our model allows any of the classical natural phenomena to be simulated –*e.g.*patina/corrosion– and demonstrates the ability to easily code many types of formal CA, such as the famous life game CA [17]. To achieve this aim, we merge and synchronize different cellular automata constructed on the same principle. Specific geometrical and CG information on this model is presented in [10].

From polygons to initial cells The following figure illustrates the cell structure of our new model. This structure is composed of two main areas: two data fields (t) and (t+1) which respectively allow to transmit and to receive information separately, and the four possible cellular states. Note that this kind of meta-state has the advantage of increasing CA computation as only non-sleeping cells can interact with their neighborhood. Both a random selection of cells position or a pre-selected order can be used to distribute cells over the polygon structure. Harmonious pattern dispositions are restricted to three cases: triangular, square and hexagonal grid. Furthemore, in our model, polygons can be of any type, the cell repartition remains the same, and input polygons and grids are only used for initially positioning the cells.



Dynamic space boxes and cellular network A new structure called *Dynamic Space Boxes* (DSB) was introduced in order to efficiently determine the neighbor of any cell. This structure relies on a space partition into boxes so that only cells inside their own boxes and the other 26-direct neighbor boxes have to be compared.

The following figure illustrates the 3D relationship between cells and englobing boxes: (a) tested cells are represented by crosses, possible neighbors by squares, and cells that should not be tested by triangles; (b) tested cells are inside the gray box surrounded by the 26 direct neighbor boxes; (c) segments denote successful connections between cell elements and the central tested box. In association with DSB, a special data structure was used to access and store 3D cells. This structure is based on dynamic double-linked-list space tree.



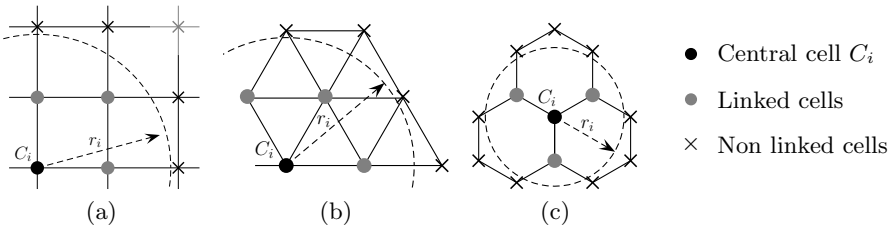


Fig. 1. Radius values r_i of the neighborhood sphere depending on the grid type.

The first step is to make the algorithm go through the space-structure and connect every cell C_i to its surroundings making a web of connected cells. For this purpose, we check cells on the 27 boxes (i.e. 26 surrounding + current) that are elements of a sphere of center C_i and radius r_i. The value r_i is set so that only direct neighbor cells are connected. For the square grid (of size c_s), r_i was set to $2.c_s - \varepsilon$ and for the triangular and hexagonal grids we use $r_i = \sqrt{3}.c_s - \varepsilon$. Figure 1 gives a graphical interpretation for each different type of grid.

Harmonizing the cellular network Once a neighbor list has been defined for each cell, the second step is to apply an attraction/repulsion function to the entire system. This function has to repulse cells that are very close or too far from each other and attract the ones that are nearby. After system convergence, the last step consists in keeping only the best links –once again– depending on the type of grid, respectively four or eight for square grids, six for triangular grids, and three for hexagonal grids.

However, this current model presents some limitations. First the harmonization is not trivial and could be improved. For instance, the pre-computation time is very long. Another drawback is that even if a cell knows its neighbor number, the current data structure does not provide the relative positions between neighbors. This property is essential for oriented-rule CA [22], and that is why it is appropriate to call this model *surface cellular network* instead of *surface cellular automaton*. Fortunately, many interesting results can be obtained without harmonizing the connected web or having cellular orientation map. In fact, self-identity and deduced-from-all CA can already be tested; this is what is proposed in the following sections.

3 Current CA model and experimental fields of study

For a given cell, the number of neighbors is not necessarily constant. Our cellular automaton model thus requires the use of symmetrical rules. Therefore only restricted directional data transfer CA are presented in this section. Nevertheless, the following subsections shows the capabilities and especially the strong potential variety of application field that our model is able to produce.

Results presented in this paper were generated using an AMD Athlon$^{TM}64X2$ Dual Core 4400+ CPU $2.21GHz$ with $2Go$ of RAM and a NVidia^{TM}GeForce

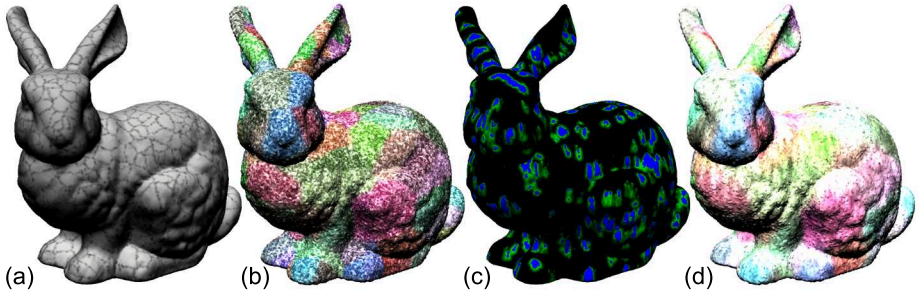


Fig. 2. Multi surface effects on the *Stanford bunny*: (a) fractures, (b) Voronoï diagram and hyper texture, (c) watermap, (d) solvent simulation.

7800 graphical card. For the implementation we used *C++* language on MS-Visual Studio^{TM}2005 and MS-Windows^{TM}2000 as operating system. Real-time was reached for geometric models composed of less than a hundred thousand cells.

This section is organized as follows. Each subsection presents a different level of using our model: subsection 3.1 describes the cell states used to control the cellular network sequences; subsection 3.2 lists graphical applications mainly in the field of automatic texturing and simulations such as cracks and melting; subsection 3.3 illustrates how fluids can be simulated with watercolor or weathering effects such as erosion, sedimentation, or lichen propagation; subsection 3.4 illustrates how non-trivial image analysis processing can be simulated based on the retina pipeline non-linear architecture.

Most subsections present a table showing transition rules with their lexical, logical, and global behavior descriptions using a *C*-like pseudo-code for conditional gates. Symbol interpretations of equations are: C: current cell; g_s: general state of C; v, v_c, p: cell vertex (*i.e.* position and orientation), cell color, and cell potential; k: active neighbor number at C; k_p k_c: as k with true potential and non-null color attributes.

3.1 General states CA

In our approach, *general states* define the behavior and the existence of a cell: to be or not in terms of geometry and time sequence. The very simple set of rules presented in table 1 are essential for making this model simultaneously flexible, multi-task, and fast computing. The first and second rules maintain initialisation and synchronisation updatings. The third and fourth activate or deactivate a cell so that only useful region of the network is used. The last state transition rule randomly selects seed-cells that can be used for generating Voronoï diagram, fluid and pigment or sediment spring, high pressure region for fracture etc.

Behaviors	State transition rules: $\forall C$	Characteristics
First CA step	$C_{t+1} \leftarrow C_t$	Initialization
CA steps	$C_t \leftarrow C_{t+1}$	Synchronization: updating
Sleeping all	$gs_{t+1} \leftarrow sleepingCell$	Powerless
Active all	$gs_{t+1} \leftarrow activeCell$	Powerful
Random selection	$\forall$ seed type $if(rand_{selected}) \Rightarrow$ $state_{t+1} \leftarrow random_{value}$(seed type)	Seed values

Table 1. General state transition rules.

3.2 Automatic texturing CA

Automatic texturing owns to the field of computer graphics (CG) which is not the direct purpose of this paper, but allows to visualize global behaviour of CA. As only few CG plates can be presented in this paper, samples of resulting images or animations can be found at www.iutsd.uhp-nancy.fr/isn/StGo.

Behaviors	State transition rules: $\forall C$	Characteristics
Spreading color	$vc_{t+1} \leftarrow \frac{vc_t + \sum C_n vc_t}{k_c+1}$	Propagation
Melting-like	$v_{t+1} \leftarrow \frac{v_t + \sum C_n v_t}{k+1}$	Averaging vert.
Crack-like	$(\exists C_k \neq C)?\|v_c : \overline{v_c} \mid gs_{t+1} \leftarrow Sleeping_{cell}$	Derivative
Corrosion	$(p_t > 0)?\|p_{t+1} \leftarrow (p_t - \Delta_p)$	Destruction
	$\|(p_t \leq 0)?\|gs_{t+1} \leftarrow Sleeping_{cell}$	
	$else(k_p > \frac{k}{2})?\|p_{t+1} \leftarrow Max_p$	
Life game	$(k_p \neq 2)?\|(k_p = 3)?\|p_{t+1} : \overline{p_{t+1}}$	Reproduction
Maze-like	$(k_p \neq 2 \parallel 4)?\|(k_p = 3)?\|p_{t+1} : \overline{p_{t+1}}$	Construction
Regrouping	$(p_t)?\|((k_p < \frac{k}{2})?\|(p_{t+1}) : (\overline{p_{t+1}}))$	Digression-dif.

Table 2. Computer graphical automatic texturing CA transition rules.

The first two lines of table 2 propose the *spreading color* –see figure 2(b)– and *melting-like* simulations. These effects are very similar as they simply average the surrounding area respectively in terms of color and vertices (position + normal). However, we can observe that the first one surrounds the surface and the second one drastically changes the 3D topology. To generate a crack-like pattern –see figures 2(a) and 5– we first use a regular color propagation, which provides a Voronoï diagram [2]. We then determine the color derivative to define area edges with a single pass CA. And finally, we associate the resulting derivative potential to fracture the structure of the object. Corrosion as well as patina simulation can be interpreted as a kind or propagation substracting at each time step a potential of each corroded cells and spreading with a random factor proportional to the number of neighbor corroded cells. The last three lines of table 2 present well-known CA where basically, cell behavior depends on the equilibrium of surrounding –see figure 3. In the first case (*i.e. life game*), we seek a specific number (*2*) for stability, and its tangent (*3*) for sudden state change. Rules of the second case are almost identical to the game of life. Surprisingly, the result is completely different: after a few steps, the system converges to a

maze-like pattern with sometimes instable areas. Concerning the *"regrouping"* CA, it belongs to the family of activation/inhibition.

Fig. 3. Three famous CA: (a) life game; (b) maze-like; (c) reaction diffusion.

3.3 Fluid CA

In this subsection, we propose to show how action of fluids can be modeled using our cellular networks model. Of course, as we have a CA approach the fluid does not exist in terms of particle or flow fields; only CA states and CA transition rules remain to simulate fluid interaction on solid, *i.e.* erosion, evaporation, pigment or sediment concentration, and sedimentation. All fluid properties are demonstrated in the following paragraph as we propose an original and non-trivial simulation: *i.e.* watercolor simulation.

Simulations	**State transition rules:** $\forall C$	**Characteristics**
Watercolor	(see detailed formula in the main text)	Erosion, evaporation, and sedimentation
Moss propagation	Only if no-moss:	
	$(min_h < p_t < max_h)?Create(C_{moss})$	Spontaneous seed
	$(rand_{fct(mossAround)})?Create(C_{moss})$	Growth around
	Only if moss:	
	$(p_t < 1)?p_{t+1} \leftarrow (p_t + \delta_{age})$:	Aging
	$gs_{t+1} \leftarrow$cellToBeRemoved state	Dying

Table 3. Fluid CA transition rules.

Direct fluid effect, *e.g.* watercolor The emphasis is on real-time generation of the most salient features of watercolor on a 3D surface within a gravity field. Compared to other works [1] our simulation must be as simple as possible. So we only use two parameters: the potential corresponding to the quantity of water in a cell and the quantity of pigments. The effect we want to obtain consists of pigments spreading in every direction on the surface. The first hypothesis is that at every step, the potential of a cell becomes the average of its potential and the potentials of its neighbor cells. The second hypothesis is that gravity force is

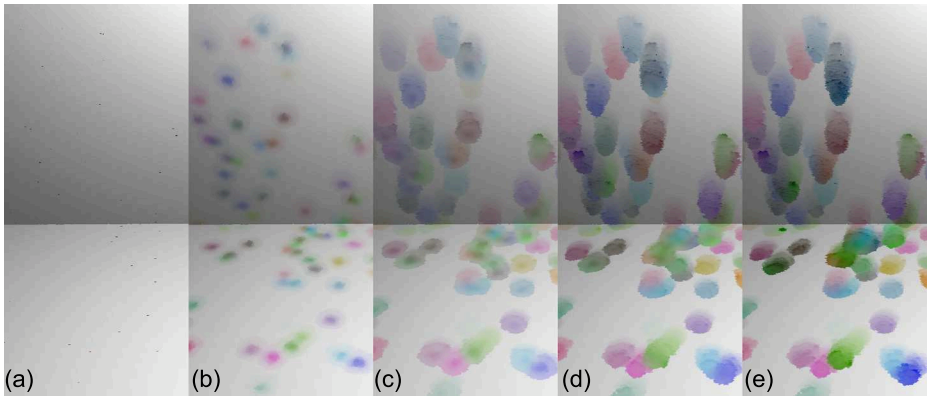


Fig. 4. Watercolor simulation: (a) seeds; (b) and (c) watercolor edge darkerning; (d) main spreading due to gravity; (e) drying process.

partially compensated by viscosity and surface tension. That is why the fluid is not only moving in the direction where the slope is the steepest but also diffusing in every direction proportionally to its slope and we assume that pigments are moving that way too. The potential of the current cell $C_{p_{t+1}}$ is:

$$p_{c,t+1} = \frac{p_{c,t} \sum_i p_{i,t} \left(1 + \frac{(h_i h_c)}{d(C_i, C_c)}\right)}{1+k} - \Delta p \tag{1}$$

where Ci denotes a neighbor cell, $(h_i h_c)/d(C_i, C_c)$ the slope and k the number of neighbors. This phenomena is illustrated in figure 4 and also in figures 2(c) (watermap) and (d) (solvent simulation).

Indirect fluid effects, *e.g.* vegetal growth In the real world weathering effect of fluid is not limited to erosion and sedimentation, it can also indirectly produce the growth of vegetal such as moss or lichen. To simulate such an effect we must introduce another parameter in the current model: the faculty of a cell to generate another cell. The figure 5 illustrates how powerful can be cellular automata. Based on a simple right-angled parallelepiped (six polygons) we first used a fracture CA simulation in order to generate surface irregularities over the 3D lattice. We then applied a waterflow CA simulation to determine where the humidity would be not too low and not too high to make moss seed appear. Finally we applied a self generated moss CA to make the moss propagate in a natural way over the lattice surface. Details of this model can be found in [6].

3.4 Retina simulation

The last field of study where we propose to apply our CA model is the architectural modeling of biological retina. Two aspects of the retina are taken

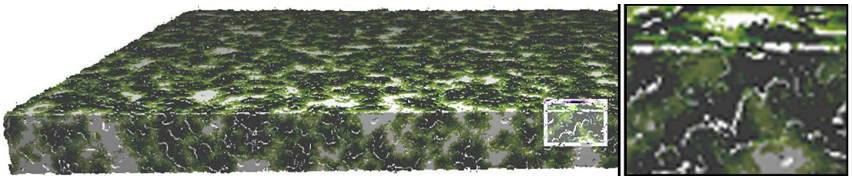


Fig. 5. Auto generated cellular moss covering a simple lattice.

into account: a simplified pipeline model of artificial retina based on cellular automata, and a 3D cellular network of the cone photoreceptors. Since the study is under way, details can be found in two publications: [3] for the topological 3D approach with simple cellular behavior, and [9] for a real-time pipeline based cellular automaton and GPU-based model. Figure 6 illustrates key steps of retina simulation using CA: (a) eye globe; (b) cellular topology around fovea area; (c) corresponding cellular network; (d) contour detection using CA; (e) interactive cellular automata.

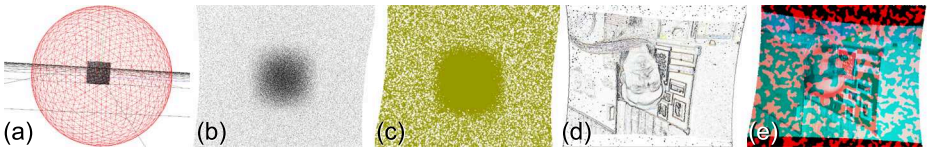


Fig. 6. Retina simulation using cellular network.

4 Conclusion and future work

We have presented a model of three-dimensional multi cellular automata allowing to simulate many phenomena in the fields of computer graphics and image processing. We detailed the process for constructing a web of connected cells. We also proposed a way to organise cellular network in any type of grid, and presented different types of CA network. Finally, to show this model capacities for different fields of study, we have applied a series of regular CA to simulate the following phenomena: spreading and diffusions CA, crack-pattern CA, environmental systems such as fluids and vegetation growth, and biological systems such as retina simulation.

From this basis, we are interested in investigating oriented-dependent CA models. Furthermore, we are also convinced that GPU programming has to play a major part in the application of CA, hence we are studying new approaches in order to take into account the inherent constraints to that type of programming.

References

1. C.J. Curtis, S.E. Anderson, J.E. Seims, K.W. Fleischery, and D.H. Salesin. Computer-generated watercolor. In *SIGGRAPH'97 Conf. Proc.*, volume 24, pages 225–232, 1997.
2. M. de Berg, O. Schwarzkopf, M. van Kerveld, and M. Overmars. *Computational Geometry: Algorithms and Applications, 2^{nd} ed.* Berlin: Springer, 1998.
3. F. Devillard, S. Gobron, F. Grandidier, and B. Heit. Implémentation par automates cellulaires d'une modélisation architecturale de rétine biologique. In *READ'05.* Institut National des Télécommunications, Evry, France, June 1-3 2005.
4. S. Druon, A. Crosnier, and L. Brigandat. Efficient cellular automaton for 2d / 3d free-form modeling. *Journal of WSCG*, 11(1, ISSN 1213-6972), 2003.
5. W. Duchting. Tumor growth simulation. *Computers and Graphics*, 14(3/4):505–508, 1990.
6. P. Even and S. Gobron. Interactive three-dimensional reconstruction and weathering simulations on buildings. In *CIPA 2005 XX^{th} International Symposium*, pages 796–801, Turin, Italie, 2005.
7. M. Gerhardt, H. Schuster, and J.J. Tyson. A cellular automaton model of excitable media including curvature and dispersion. *Science 247*, pages 1563–1566, 1990.
8. S. Gobron and N. Chiba. 3D surface cellular automata and their applications. *The Journal of Visualization and Computer Animation*, 10:143–158, 1999.
9. S. Gobron, F. Devillard, and B. Heit. Real-time contour restoration and segmentation using cellular automaton and GPU programming. in reviewing process, 2006.
10. S. Gobron and D. Finck. Generating surface textures based on cellular networks. In *The Geometric Modeling and Imaging international conference (GMAI06).* IEEE Computer Society, July 2006.
11. M. Harris. Implementation of a cml boiling simulation using graphics hardware. In *CS. Dept, UNCCH, Tech. Report*, volume 02-016, 2003.
12. M. Harris, G. Coombe, T. Scheueermann, and A. Lastra. Physically-based visual simulation on graphics hardware. *Graphics Hardware*, pages 1–10, 2002.
13. W. Li. Complex patterns generated by next nearest neighbors cellular automata. *Computers and Graphics*, 13(4):531–537, 1989.
14. R. Makkuni. Pixelated structures as a compositional medium. *The Visual Computer*, 2(4):243–254, 1986.
15. W.K. Mason. Art from cellular automata and symmetrized dot-patterns. *Computers and Graphics*, 16(4):439–442, 1992.
16. C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *ACM Computer Graphics Conf. Proc.*, volume 21:4, pages 25–33, July 1987.
17. D. Thalmann. A *lifegame* approach to surface modeling and rendering. *The Visual Computer*, 2:384–390, 1986.
18. J. Tran, D. Jordan, and D. Luebke. New challenges for cellular automata simulation on the GPU. *www.cs.virginia.edu*, 2003.
19. G. Turk. Generating texture for arbitrary surfaces using reaction-diffusion. In *SIGGRAPH'91 Conf. Proc.*, volume 25, pages 289–298, 1991.
20. L. Wang, X. Wang, X Tong, S. Lin, S. Hu, B. Guo, , and H-Y. Shum. View-dependent displacement mapping. In *SIGGRAPH'03 Conf. Proc.*, volume 22, pages 334–339, 2003.
21. A. Witkin and M. Kass. Reaction-diffusion textures. In *SIGGRAPH'91 Conf. Proc.*, pages 299–308, 1991.
22. S. Wolfram. *A new kind of science.* Wolfram Media Inc., 1st edition, 2002.