Applying Cell-DEVS in 3D Free-Form Shape Modeling

Pengfei Wu, Xiuping Wu, Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON, K1S 5B6, Canada
{pfwu, xpwu, Gabriel.Wainer}@sce.carleton.ca

Abstract. Modeling free-form shapes in 3D spaces based on strict physical laws require a considerable amount of computation time. Previous experiences with Cellular Automata demonstrated substantial improvements for these applications. Here, we describe an efficient approach applying the Cell-DEVS formalism to model the deformation of free-form shape objects in a 3D space. The objects are treated as virtual clay, and the values assigned to each cell in the model represent a certain amount of clay in the cell. The clay object is deformed in accordance with the physical conservation laws.

1 Introduction

3D modeling techniques have become increasingly attractive, ranging from CAD systems up to image morphing in computer graphics. The representation of solid 3D objects requires to use restrictive geometrical operations. Different studies on the simulation of such deformations for objects use strict physical models, such as finite element methods, methods based on elasticity theory, and applications of particle systems. All these methods and applications need considerable time for computing deformations according to the laws, and human interactions are not permitted especially for complex shapes.

Instead, imagining the 3D objects as clay that can be freely deformed, we can understand problems on 3D objects. Some of the undergoing efforts considering volume sculpting in a 3D virtual space use a discretization of the space in 2D or 3D cells. In [1], a solution based on Cellular Automata (CA) was presented. A 3D CA is used to simulate plastic deformations of clay, and each cell is allocated a finite state automaton, which is given the simple distribution rules of the virtual clay instead of complicated physical laws. Each automaton repeats state transitions according to the state of their neighbors. This approach tries to avoid the considerable computation time for plastic deformation and make the deformation process natural by showing the behavior of real clay. An extension of this work, presented in [2], includes new repartition algorithms.

We will show how to model a 3D free-form object using Cell-DEVS [3], and we will simulate deformation using the CD++ toolkit [4]. Cell-DEVS is a novel approach to represent models of real systems as cell spaces. Cell-DEVS uses the DEVS (Discrete Events systems Specifications) formalism [5] to define a cell space where each cell is defined as a DEVS model. This technique permits to build discrete-event cell spaces, and it improves their definition by making the timing specification more expressive. The Cell-DEVS presented here describes the behavior of each cell in a 3D free-form virtual clay model using the state transition rules presented in [1]. This model describes effectively the behavior of free-form object: compression (from outside) and deformation (from inside).

2 Background

Cell-DEVS [3] is a combination of CA and DEVS [5] that allows the implementation of cellular models with timing delays. A Cell-DEVS model is defined as a lattice of cells holding a state variable and a computing apparatus to update the cell state. This is done using the present cell state and a set of inputs coming from cells in the neighborhood. Cell-DEVS improves execution performance of cellular models by using a discrete-event approach. It also enhances the cell's timing definition by making it more expressive [6]. Each cell, defined as $TDC = < X, Y, S, N, delay, d, d_{INT}, d_{EXT}, t, l, D >$, uses N inputs to compute its next state. These inputs, which are received through the model's interface (***X***, ***Y***), activate the local computing function (τ). State (**s**) changes can be transmitted to other models, but only after the consumption of a delay (**d**). Two kinds of delays can be defined: *transport* delays model a variable commuting time, and *inertial* delays, which have preemptive semantics (scheduled events can be discarded). Once the cell behavior is defined, a coupled Cell-DEVS is created by putting together a number of cells interconnected by a neighborhood relationship.

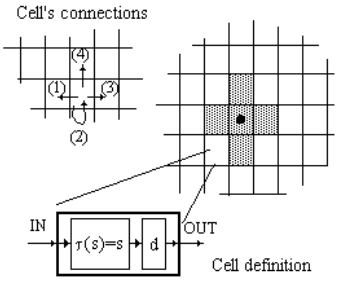


Fig. 1. Description of a Cell-DEVS model.

A coupled Cell-DEVS is composed of an array of atomic cells, with given size and dimensions, defined as $GCC = < Xlist, Ylist, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z >$. Each cell is connected to its neighborhood through standard DEVS input/output ports. Border

cells have a different behavior due to their particular locations. Finally, the model's external couplings can be defined in the Xlist and Ylist. Each cell in a Cell-DEVS is a DEVS atomic model, and the cell space is a DEVS coupled model. DEVS is an increasingly accepted modeling and simulation framework. DEVS is a sound formalism based on generic dynamic systems, including well defined coupling of components, and hierarchical modular construction. A DEVS model is described as a composite of submodels, each of them being behavioral (atomic) or structural (coupled). Each atomic model, defined by $AM = < X, Y, S, d_{ext}, d_{int}, l, ta >$, has an interface consisting of input (***X***) and output (***Y***) *ports* to communicate with other models. Every state (***S***) in the model is associated with a time advance (***ta***) function, which determines the duration of the state. Once this time is consumed, an internal transition is triggered. At that moment, the model generates results through the output ports by activating an output function (l). Then, an internal transition function (d_{int}) is fired, producing a state change. Input external events, which are received in the input ports, activate the external transition function (d_{ext}). Coupled models are defined as a set of basic components (atomic or coupled), which are interconnected through the model's interfaces. The model's coupling defines how to convert the outputs of a model into inputs for the others, and to inputs/outputs to the exterior of the model.

CD++ [4] is a modeling tool that was defined using DEVS and Cell-DEVS specifications. DEVS Atomic models can be incorporated onto a class hierarchy programmed in C++. Coupled models can be defined using a built-in specification language, in which the model is specified by including the size and dimension of the cell space, the shape of the neighborhood and borders. The cell's local computing function is defined using a set of rules with the form: POSTCONDITION DELAY { PRECONDITION }. These indicate that when the *PRECONDITION* is satisfied, the state of the cell will change to the designated *POSTCONDITION*, whose computed value will be transmitted to other components after consuming the *DELAY*.

3 A Virtual Clay Model

In virtual clay models, a 3D object deformation is considered as a physical process that equally distributes the virtual clay to the adjacent areas. A threshold is associated with the deformation of the object [1]: when the density is under the threshold, the object keeps its shape. If a portion receives an external force, its density changes; if the density is above the threshold, the object is deformed, and clay is transported from high-density to low-density portions. However, the total mass of the clay should be conserved. We model the object as a 3D Cell-DEVS. A positive value assigned to each cell represents the mass of clay in the piece. In [1], the authors define the model using a Margolus neighborhood, and the following rules for each block:

[Step A] For each cell *i* whose state is 1,

$dm_i = m_i * a;$

$m_i = m_i - dm_i;$

[Step B] For each cell *j* whose state is 0,

$$m_j = m_j + ((dm_1 + dm_2 + \ldots + dm_t) / n);$$

where a is a constant rate for distribution ($0 < a < 1$), t is the number of cells over threshold and n is the number of cells under threshold. Here, we denote the state of a cell as 1 if its virtual clay is over the threshold. Otherwise, the state is 0. The value dm_i represents the excessive clay in cell i, which will be distributed to the neighboring cells. From these two steps, we can see that the total mass of virtual clay within a block is conserved during the state transitions. Figure 3 illustrates the transition rules in 2D.

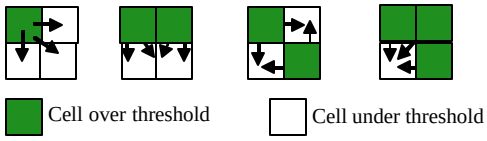


Fig. 2. 2D block patterns.

The deformation of a virtual clay object is based on a push operation. The clay is transported from a cell into the adjacent ones along the direction of pushing. The surface of a virtual clay object can be pushed at most one cell in depth per step.

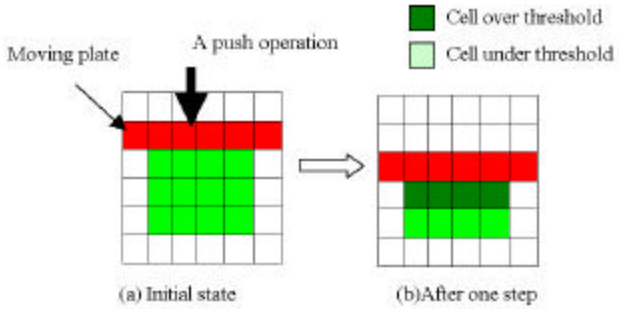


Fig. 3. Push operation by a moving plate.

Figure 3 shows an example of the push operation by a moving plate. This plate is modeled as a set of special cells to which virtual clay cannot be distributed. When the threshold is surpassed as the result of pushing, virtual clay is distributed according to the transition rules. The state transitions are repeated until there are no cells over threshold (a stable state). The number of steps to reach the stable state depends on the total mass of virtual clay, the threshold and the parameter a.

4 Modeling a Virtual Clay Object in Cell-DEVS

We used Cell-DEVS model to simulate the behavior of the deformation of a virtual clay object. Our focus is on a 3D free-form object, using the transition rules presented in

the previous section. Figure 4 illustrates the 3D Margolus neighborhood we used, in which the nearest 8 cells make one block. Similarly to the 2D neighborhood, each cell belongs to different blocks in odd and even steps.

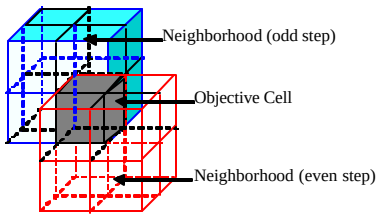


Fig. 4. 3D Margolus neighborhood

The values in each cell represent the density (mass) of that cell. A cell with a value of zero means this cell is out of the object, while a positive value means the cell is within the object. The final state contains the free-form object in stable state after deformation. The transition procedure is done in two stages as follows.

- Deformation: if there are cells with density over the threshold, the deformation transition rules are applied.
- Compression: we assume that there is a moving plate and the virtual clay next to the plate is transferred from a cell into the adjacent cell along the direction of pushing. In the model, this plate is handled as a dummy plate with each cell having a value of zero, staying right on top of the object.

During the transition procedure, each cell has its different neighbors at odd and even steps. The neighborhood of each cell is identified according to its location in the block. Meanwhile, each cell has different transition policies for the deformation and compression stages. Therefore, we used a four-dimensional Cell-DEVS model, in which each 3D hyperplane of (x, y, z) represents a set of state variables and the fourth dimension is a control variable, as follows.

- Hyperplane 0 (x, y, z, 0) represents the free form object.
- Hyperplane 1 (x, y, z, 1) defines the odd or even step so that each cell in Hyperplane 0 can identify its Margolus neighborhood.
- Hyperplane 2 (x, y, z, 2) is a control Hyperplane. Compression will be performed if (x, y, z, 2) = 1, and deformation if cell(x, y, z, 2) =0.

This Cell-DEVS coupled model is defined as follows:

$$Plastic = < Xlist, Ylist, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z > .$$

$\boldsymbol{X}$=$\boldsymbol{Y}$=$\boldsymbol{Xlist}$=$\boldsymbol{Ylist}$= Ø; **S** = {s| s >= 0, s∈ R }; $\boldsymbol{n}$=4; $\boldsymbol{t_1}$=10; $\boldsymbol{t_2}$=9; $\boldsymbol{t_1}$=12; $\boldsymbol{t_4}$=12;

N = {(-1,-1,-1,0),(-1,0,-1,0),(-1,1,-1,0),(0,-1,-1,0),(0,0,-1,0),(0,1,-1,0),(1,-1,-1,0), (1,0,-1,0),(1,1,-1,0),(-1,-1,0,0),(-1,0,0,0),(-1,1,0,0),(0,1,0,0),(0,0,0,0),(0,1,0,0), (1,-1,0,0),(1,0,0,0),(1,1,0,0),(-1,-1,1,0),(-1,0,1,0),(-1,1,1,0),(0,-1,1,0),(0,0,1,0), (0,1,1,0),(1,-1,1,0),(1,0,1,0),(1,1,1,0),(-1,-1,2,0),(-1,0,2,0),(-1,1,2,0),(0,-1,2,0), (0,0,2,0),(0,1,2,0),(1,-1,2,0),(1,0,2,0),(1,1,2,0),(0,0,0,1),(0,0,0,2)}

The definition of this Cell-DEVS coupled model using CD++ is illustrated in Figure 5 below.

```
[plastic]
dim : (10,9,12,3)     delay : transport
border : nowrapped
neighbors:(-1,-1,-1,0)(-1,0,-1,0)(-1,1,-1,0)
(0,-1,-1,0)(0,0,-1,0)(0,1,-1,0)(1,-1,-1,0)
(1,0,-1,0)(1,1,-1,0)(-1,-1,0,0)(-1,0,0,0)(-1,1,0,0)
localtransition : deformation-rule
[deformation-rule]
...
```

Fig. 5. Cell-DEVS coupled model specification in CD++

This Cell-DEVS model that has three hyperplanes of 10*9*12 each. The cell's behavior is defined by the localtransition clause, and it will be discussed in detail in the following sections. We will present the rules at deformation and compression stages and show how the transformation is controlled in Cell-DEVS formalism. The rules can be generalized as follows.

1. Perform deformation if cell (0, 0, 0, 2) = 0 and this cell is on Hyperplane 0.
2. Perform compression if cell (0,0,0,2) = 1 and this cell is on Hyperplane 0
3. Even/odd step alternates if cell (0, 0, 0,1) = 0 and this cell is on Hyperplane 1
4. Deformation/compression control alternates if this cell is on Hyperplane 2

4.1 Deformation Rules in the Cell-DEVS model

The deformation stage involves the activating different rules at odd and even steps, in which we decide the different neighbors that the objective cell receives clay from or distributes clay to. Only hyperplane 0 (x, y, z, 0) performs the deformation transition, while hyperplane 1 (x, y, z, 1) helps to judge whether the cell in Hyperplane 0 change in odd step or in even step and hyperplane 2 (x, y, z, 2) identify the deformation stage. In each step its neighborhood can be located in a 2x2x2 block. In Figure 6 we show the mechanism to select the Margolus block. The origin cell is colored in gray and its neighbors are defined according to the coordinates shown in the figure. We repeat the procedure for other cells in the same Margolus block to obtain the neighbors of each objective cell in the same hyperplane.

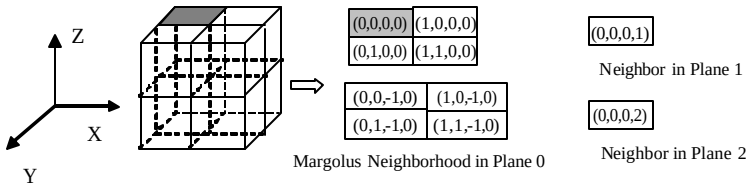


Fig. 6. A cell and its neighborhood definition at the deformation stage

The pair < (x,y,z,0), (x,y,z,1) > define the step (odd or even). The neighbor on the hyperplane 0 is described in the figure, and its value, together with the value of all the neighbors in the same Margolus block, can be used to decide which transition rule should be applied. The deformation rules can be generalized as follows (the detailed definition of these rules in CD++ can be found in [7].

- Cells **gain** clay from neighbors on hyperplane (x,y,z,0) if (0,0,0,2) = 0, (0,0,0,0) is below the threshold, and at least one neighbor is above the threshold.
- Cells **distribute** clay to neighbors on Hyperplane (x,y,z,0) if (0,0,0,2) = 0, (0,0,0,0) is above threshold and at least one neighbor is under threshold.

4.2 Compression Rules in the Cell-DEVS model

Similar to deformation, the compression only takes place on hyperplane 0. Hyperplane 2 controls when compression occurs: only when cell (x, y, z, 2) =1. During compression the clay in the cells right under the moving plate is transferred into the adjacent cells along the direction of pushing. The moving plate is represented by a set of cells with values of zero sitting on the top of the object. We assume the plate moves down along z-axis as shown in Figure 7. For each cell (x,y,z,0), if all neighboring cells (-1,-1,2,0), (-1,0,2,0), (-1,1,2,0), (0,-1,2,0), (0,0,2,0), (0,1,2,0), (1,-1,2,0), (1,0,2,0), (1,1,2,0) are zero and at least one of neighbor cells (-1,-1,1,0), (-1,0,1,0), (-1,1,1,0), (0,-1,1,0), (0,0,1,0) , (0,1,1,0), (1,-1,1,0) , (1,0,1,0), (1,1,1,0) is greater than zero, the cell should gain all clay in its neighbor (0,0,1,0). The neighbors of each cell are defined in Figure 7.

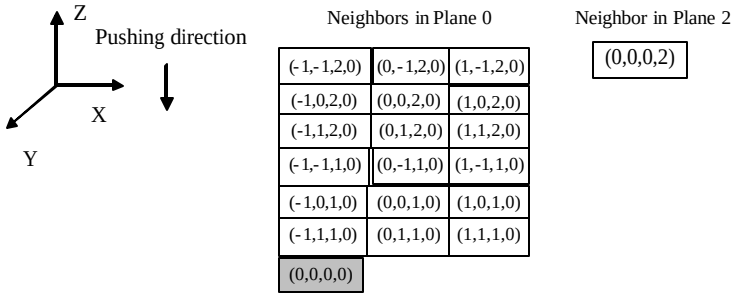


Fig. 7. A cell and its neighbor definition at the compression stage

The definition of these rules in CD++ is as follows:

```
[compression-rule]
%plate moving
rule : {(0,0,0,0)+(0,0,1,0)} 100 {(0,0,0,2)=1 and
```

```
cellpos(3)=0 and cellpos(0)>0 and cellpos(0)<9 and
cellpos(1)>0 and cellpos(1)<8 and cellpos(2) <10 and
((-1,-1,2,0)+(-1,0,2,0)+(-1,1,2,0)+(0,-1,2,0)+(0,0,2,0)
+(0,1,2,0)+(1,-1,2,0)+(1,0,2,0)+(1,1,2,0))=0 and
((-1,-1,1,0)+(-1,0,1,0)+(-1,1,1,0)+(0,-1,1,0)+(0,0,1,0)
+(0,1,1,0)+(1,-1,1,0) +(1,0,1,0)+(1,1,1,0))>0 }
%step 1: add the first row to the second row
rule : 0 100 {(0,0,0,2) = 1 and cellpos(3)=0 and
cellpos(0)>0 and cellpos(0)<9 and cellpos(1)>0 and
cellpos(1)<8 and cellpos(0) < 11 and ((-1,-1,1,0)+
(-1,0,1,0)+(-1,1,1,0)+(0,-1,1,0)+(0,0,1,0)+(0,1,1,0)+
(1,-1,1,0) +(1,0,1,0)+(1,1,1,0))=0 and ((-1,-1,0,0)+
(-1,0,0,0)+(-1,1,0,0)+(0,-1,0,0)+(0,0,0,0)+(0,1,0,0)+
(1,-1,0,0) +(1,0,0,0)+(1,1,0,0))>0}
%step2 :change the first row to 0, the plate has moved
one step further
```

Fig. 8. Cell-DEVS specification of the compression rules in CD++.

4.3 Control Hyperplane in Cell-DEVS model

Hyperplane 2 controls when the deformation or compression stages occur. The value of each cell on hyperplane 2 switches between 0 (deformation) and 1 (compression). We assume that the transport delay of performing a compression step is 3000 ms, thirty times longer than the delay of a deformation. The 3D free-form object will reach stable state within 3000 ms if deformation occurs.

The transition rule for the control hyperplane is as follows.

- S ←1 if cell (0,0,0,1)=0 and cell (0,0,0,0) =0 and cell itself is on hyperplane 1
- S ←0 if cell (0,0,0,1)=0 and cell (0,0,0,0) =1 and cell itself is on hyperplane 1
- S ← 1 if cell (0,0,0,0) =0 and cell itself is on hyperplane 2
- S ← 0 if cell (0,0,0,0) =1 and cell itself is on hyperplane 2

The specification of these rules in CD++ is illustrated in Figure 9 below.

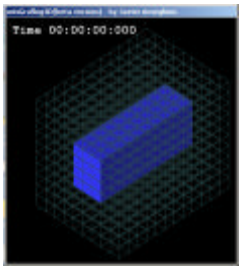
```
rule : 1 100 {(0,0,0,1)=0 and cellpos(3)=1 and
          (0,0,0,0)=0 }
rule : 0 100 {(0,0,0,1) = 0 and cellpos(3)=1 and
      (0,0,0,0)=1} %alternate Margolus neighborhood
rule : 1 3000 { cellpos(3)=2 and (0,0,0,0)=0 }
rule : 0 100 {cellpos(3)=2 and (0,0,0,0)=1}
%plate moving
```

Fig. 9. Cell-DEVS specification of control rules in CD++

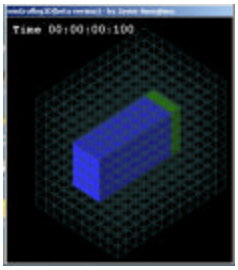
5 Experimental Results

We executed this Cell-DEVS model, and studied the deformation process of a clay object. We studied each cell at different time steps (compression or deformation), and the total mass of the object was examined. We found that the total mass (represented by cells in hyperplane 0) was conserved for every transition.

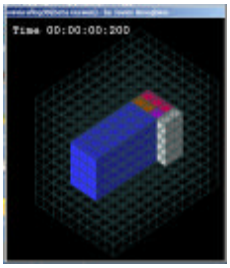
Some of the results obtained are presented in Figure 10. We show several steps during the transition process, which includes the initial state, first three compression steps and some related deformation steps. Figure (b) shows the immediate result after the first compression step. Figures (c) and (d) show the object deformation. In Figure (d), a stable state is reached. Figure (e) to (i) show the repartition of clay after the second and third compression steps.



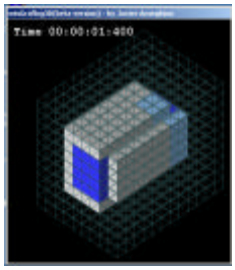
(a) Initial state



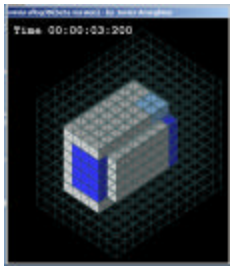
(b) Compression



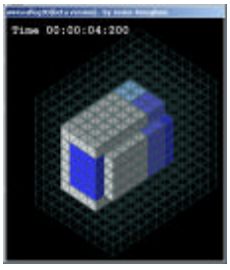
(c) Deformation



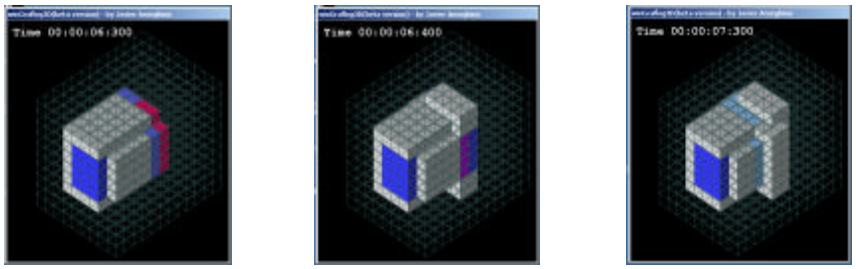
(d) Deformation, stable



(e) Second compression



(f) Deformation, stable



(g) Third compression (h) Deformation (i) Deformation, stable

Fig. 10. The deformation of the free-form object using Cell-DEVS

6 Conclusion

We have showed how to model the deformation of 3D free-form object using the Cell-DEVS formalism. The total mass of the free-form object is conserved during the stages of deformation. The complex behavior has been simulated using simple and easy-to-understand transition rules. These rules are defined using the specification language in CD++, which is itself is simple and easy to learn. A similar study has been carried out in [1] and [2]. However, in their work, a sophisticated simulation program was developed in order to simulate this complex behavior. The simulation program is application-specific, which can not be applied to general cases. By applying the Cell-DEVS formalism along with the CD++ toolkit support, more generality can be achieved. This allows us to model complex systems in a simple fashion. Likewise, changing parameters for studying simple conditions is straightforward. Cell-DEVS permits defining zones of cells with different behavior, which can be used to analyze objects composed of different materials. Likewise, Cell-DEVS models can be integrated with other DEVS and Cell-DEVS models, permitting defining multimodels interacting with each other, using different modeling techniques.

One of the key aspects to simulate the deformation is to understand the characteristics of the Margolus neighborhood. This has been studied and explored in the model specification. The results, which have been illustrated by a visualization tool have demonstrated the capability of Cell-DEVS modeling approach as well as the tool utilities. The behavior of the deformation is well simulated with a simple, easy-understood mechanism.

References

1. H. Arata, Y. Takai, N. K. Takai, T. Yamamoto: Free-form shape modeling by 3D cellular automata. Proceedings of the International Conference on Shape Modeling and Applications (1999)
2. S. Druon, A. Crosnier, L. Brigandat: Efficient Cellular Automata for 2D / 3D Free-Form Modeling. Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2003 (2003)
3. G. Wainer, N. Giambiasi: N-dimensional Cell-DEVS. Discrete Events Systems: Theory and Applications, Kluwer, Vol.12. No.1 (January 2002) 135-157
4. G. Wainer: CD++: a toolkit to define discrete-event models. Software, Practice and Experience. Wiley. Vol. 32. No.3. (November 2002) 1261-1306
5. B. Zeigler, T. Kim, H. Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press (2000)
6. G. Wainer, N. Giambiasi: Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. Simulation, Vol. 76. No. 1. (January 2001)
7. P. Wu, X. Wu.: A model of 3D deformation of free-form shapes in Cell-DEVS. Technical Report SCE-03-006. Dept. of Systems and Computer Engineering. Carleton University. (2003)