PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 07/08

A Multi-Agent-based 3D Visualization of Stem Cell Behavior

Geisa Martins Faustino
Maíra Atanásio de Cerqueira Gatti
Carlos José Pereira de Lucena
Marcelo Gattass

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 07/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena February, 2008

A Multi-Agent-based 3D Visualization of Stem Cell Behavior

Geisa Martins Faustino, Maíra Atanásio de Cerqueira Gatti, Carlos José Pereira de Lucena and Marcelo Gattass

{gfaustino, mgatti, lucena}@inf.puc-rio.br, mgattass@tecgraf.puc-rio.br

Abstract. This paper presents some results achieved in order to build a multi-agent based 3D visualization for the stem cell behavior simulation. The main challenge of this work was to develop the visualization of the behavior of stem cells, such as proliferation and differentiation as self-organizing agents, considering that it is an autonomous, adaptive, complex system. We designed the stem cell spatial self-organization that made it possible to visualize the emergent phenomenon and the entity agent. It also allowed visualization of the interaction between the agent and the environment.

Keywords: Multi-agent systems, 3D visualization, stem cells.

Resumo. Este artigo apresenta alguns resultados obtidos na construção de um visualizador 3D baseado em multiagentes para a simulação do comportamento das células tronco. Considerando que este é um sistema autônomo, adaptativo e complexo, o principal desafio deste trabalho foi desenvolver a visualização do comportamento das células tronco, tal como proliferação e diferenciação como agentes auto-organizáveis. Nós projetamos a auto-organização espacial das células tronco, o que tornou possível a visualização do comportamento emergente e da entidade agente. Isto também possibilitou visualizar as interações entre agente e ambiente.

Palavras-chave: Sistemas multiagentes, visualização 3D, células tronco.

In charge of publicationss:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introduction

Stem cells have the ability to divide itself for indefinite periods – often throughout the life of the organism. Under the right conditions, stem cells can give rise to the many different kinds of cells that composes the organism through a process called differentiation. Thus, stem cells have been used in the treatment of a wide variety of diseases, such as: leukemia, Parkinson's disease, etc.

In recent years, stem cells have been cultivated in laboratories by a self-induced process that stimulates the differentiation into a specific mature cell. It is still hard to predict the stem cell's behavior in the presence of some substances. Therefore, many stem cells can be wasted on the presence of wrong substances. This fact makes the stem cell culture even more expensive. The prediction of the stem cell's behavior in the presence of these substances by computer simulation not only reduces costs but also makes it easier to test different combinations of substances. A stem cell behavior simulator that provides visualization of cell structures in several levels of details is a powerful tool for understanding and accelerating the stem cell therapy process. Moreover, if the model is sufficiently detailed and accurate, then it serves as a reference for interpreting experimental results and is a powerful means for suggesting new hypotheses.

Stem cell behavior can be characterized as a self-organizing complex system with emergent behavior generation. In the context of cell biology, self-organization can be defined as the capacity of a macromolecular complex or organelle to determine its own structure, based on the functional interactions of its components. In a self-organizing system, the interactions of its molecular parts determine its architectural and functional features [10]. Therefore, Multi-Agent Systems (MAS) fit nicely into this problem and it is has become possible to understand how stem cells organize themselves, and to deal with their emergent global behavior [9] and [5].

This paper describes the results achieved towards a multi-agent-based 3D visualization system for stem cell behavior simulation, using the previous 2D visualization system [9] as its basis. The main contribution of this paper is the reformulation of the spatial self-organizing of the stem cells and its extension to the 3D space. Our second contribution is the incorporation of the MASON toolkit [13] into the system, has several advantages: a) development of a new architecture that provides independence between the model and visualization components; b) improvement on the visualization component that makes it possible to visualize thousands of cells; c) and improvements to the user interface to which several tools were added, such as arcball and zoom. Both contributions resulted in, to the authors' knowledge, the first multi-agent-based 3D visualization system of stem cell's behavior.

This paper is organized as follows: in section 2 we highlight the related works; section 3 presents the problem; section 4 shows the proposed solution; section 5 describes the advances in the visualization component; and finally, section 6 concludes this paper and presents some future works.

2 Related Works

There are two ways for visualizing multi-agent systems: the agents interaction protocol and the agent entity. In the former, we visualize a sequence of messages between agents

and the constraints on the content of those messages. On the other hand, the latter method visualizes the entity agent and its iteration with the environment. Most software programs, such as JADE platform [6] and Zeus toolkit [2], provide graphical tools that allow the visualization of the messages exchanged between agents.

The toolkits MASON [13], Repast [11] and Swarm [8] provide the visualization of the entity agent and its interaction with the environment.

The Repast [11] is a free and open-source agent-based modeling and simulation toolkit. Three Repast platforms are currently available: Repast for Java (Repast J), Repast for the Microsoft .NET framework (Repast .NET), and Repast for Python Scripting (Repast Python). Each of these platforms contains the same core features.

The Swarm [8] is a library of object-oriented classes that implements the Swarm conceptual framework for agent-based models and provides many tools for implementing, observing, and conducting experiments on ABMs. It was written in Objective-C and an object-oriented extension of the C programming language.

The MASON [13] is multi-agent simulation library core developed in Java. It provides both a model library and an optional suite of visualization tools in 2D and 3D. We chose the MASON toolkit because, when the project started, it was the only one to offer 3D visualization feature, to our knowledge.

Vizzari et al. [7] developed a framework supporting the development of MAS-based simulations based on the Multilayered Multi-Agent Situated System model provided with a 3D visualization. We did not adopt this framework because it offers many features that are not used in our system. It would add a non-desired complexity to our system. Mason is simpler and provides all the features required.

Gatti et al. [9] developed a simplified stem cell conceptual based-agent model and a stem cell behavior simulator based on a multi-agent system. It was possible visualize the emergent behavior by means of a 2D visualization component. This prototype also provided a 2D cell visualization in a higher level of detail (macro level) by clicking over the cell. This system did not use any framework, middleware or agent-base platform. The proposed solution overloaded the CPU even with small numbers of entities. The visualization froze up many times as a result.

Lales et al. [3] proposed a model metabolism pathways with a multi-agent systems (MAS) in the context of the mitochondrial metabolism. They also presented a first approach to a 3D molecular designing and the application to the particular case of the two mitochondrial membranes. This approach is closer to ours through the fact that each entity is represented by a reactive agent situated in a 3D space and the reactions are scheduled by time steps. However, we could not find the visualization tool used for this simulation. It was not cited in the paper and we did not find it in the literature at all.

Bandini et al. [12] described how the Situated Cellular Agents (SCA) model was applied to model the Immune System (IS). A prototype for a multiŰagent system was designed and developed using Repast.

3 Problem Description

In this section we show the stem cell differentiation and division process. Both are important to the visualization. Some features and entities relevant for our system also are presented.

An important entity with an active influence in the stem cell differentiation process is the Niche. The Niche is a microenvironment in which stem cells are found, which interacts with them to regulate their fate and contains proteins that constitute the extra cellular environment. It saves stem cells from depletion, while protecting the host from over-exuberant stem-cell proliferation. The word "niche" can be in reference to the *in vivo* or *in vitro* stem cell microenvironment.

In developmental biology, cellular differentiation is the process by which a less specialized cell becomes a more specialized cell type. The stem cells can be specialized in a several kinds of cells, such as heart cells, skin cells, or nerve cells. Figure 1 shows some of the kinds of cells that a stem cell can differentiate itself into. We are interested in the case of a stem cell that differentiates in to a mature neuron.

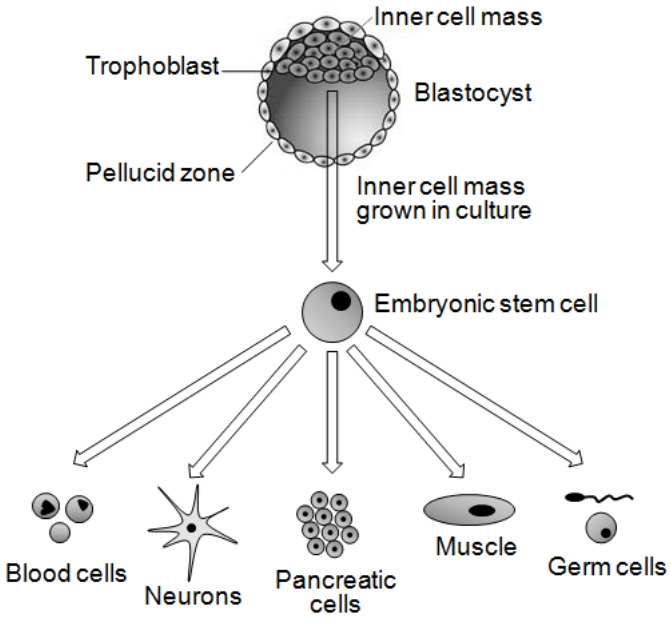


Figura 1: Stem cell differentiation: note the several kinds of cells that the stem cell can be differentiated into.

We based our model layer on the cell life cycle and stem cell process conceptual model designed with stem cell researchers. A new stem cell conceptual base-agent model (more refined than the previous one) and a multi-agent-based framework for support applications that simulate the stem cells behavior were developed. The conceptual model and the framework details are beyond the scope of this paper, which is to show in detail the framework's visualization component and its advances in relation to the previous version [9].

At the agent-based stem cell computational model, there are four kinds of cells: **multipotent cells** are cells with a hight power of differentiation, that can give rise to several other cell types; **precursor cells** are cells there are able to self-differentiate into a specific kind of cell, for instance, a blood cell; **progenitor cells** are stem cells that have developed to the stage where they are committed to forming a particular kind of new specific cell; **differentiated cells** are specialized cells with no power of differentiation.

In the case that the differentiation process generates a neuron we have: a stem cell like a multi-potent cell; a precursor neuron like a precursor cell; a progenitor neuron like a progenitor cell and a neuron like a differentiated cell – for instance, a neuroblast and the neuron itself, respectively.

In order to ensure self-renewal and differentiation, the stem cells undergo two types of cell division: **symmetric division**: giving rise to two identical daughter cells, both endowed with stem cell properties;and **asymmetric division**: produces only one stem cell and a progenitor or precursor cell with limited self-renewal potential until the mature cell generation with no differentiation potentiality. Figure 2 shows the stem cell division process until the mature cell generation with no differentiation potentiality and for the case when a neuron is generated.

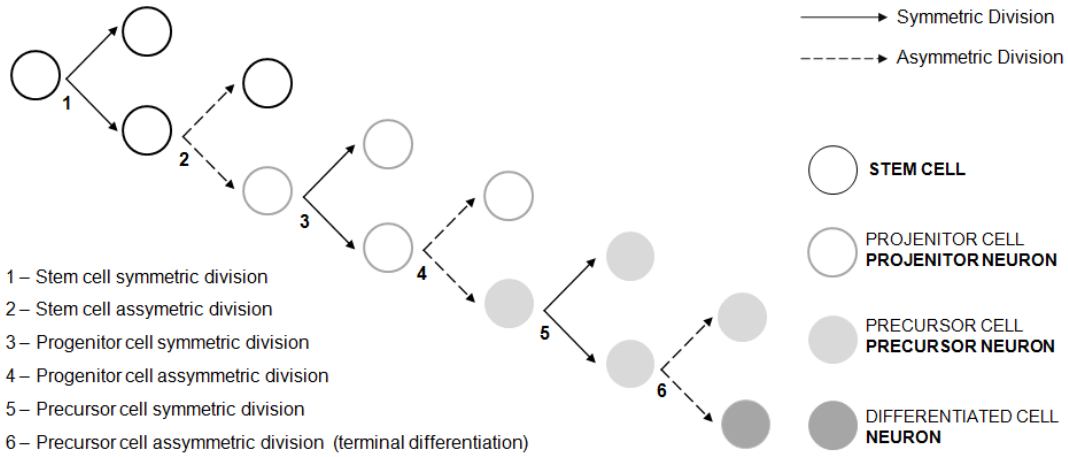


Figura 2: Stem cell process: on right, a legend showing the cells present in the process until the mature cell and (in bold) the cells present until the neuron generation; and on left, all division steps until the mature cell generation.

We are interested in a 3D visualization of the differentiation and division processes for the neuron generation in a higher level of detail. The visualization in a macro level is fundamental for analyzing the differentiation process. In the next section, we present the approach adopted for the visualization process.

4 The Proposed Approach

In this section, we first present how MASON's visualization layer is instantiated; next we show how the entities are represented; and finally we detail how the spatial self-organizing of the entities are modeled.

We used the MASON toolkit as the basis for developing the model and visualization component. It provides a 3D visualization of the entity agent and the emergent behavior. Both layers are independent; therefore it is possible to run the simulation without visualizing it.

In order to instance the MASON's visualization layer, there are some classes that must be specialized: *SimState* and *GUIState*. *SimState* class represents the simulation and the *GUIState* class encapsulates it. These two classes are responsible for linking the model layer

with the visualization layer. Besides that, the *GUIState* class associates *portrayals* with *displays*. *Displays* are responsible for the graphic control components of the simulation and *Portrayals* define how to draw and graphically manipulate instances and attributes in the physical simulation space.

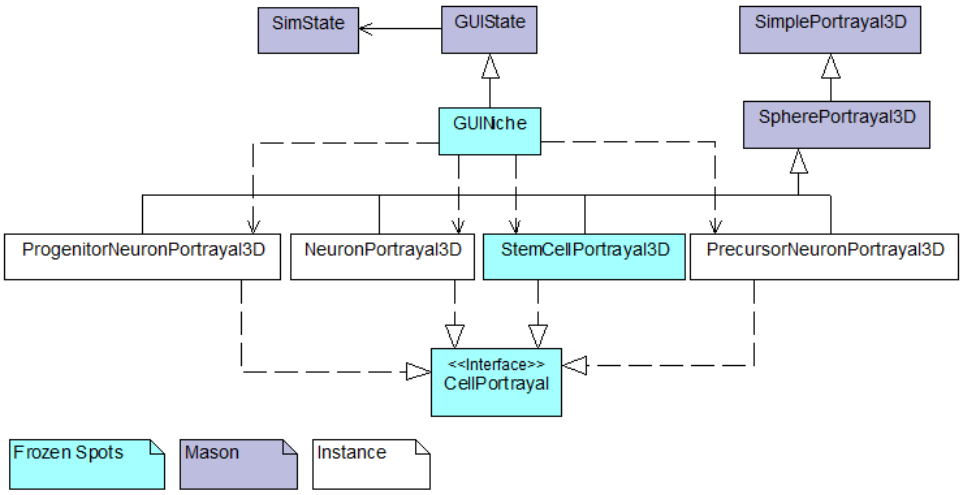


Figura 3: Class diagram of the visualization component.

Figure 3 shows the class diagram of the visualization component of our system. There is one portrayal for each kind of cell (agent) of the conceptual model (ProgenitorNeuron-Portrayal3D, PrecursorNeuronPortrayal3D, etc). This organization enables us to change the cell design (e.g. sphere, mesh, etc.) without making changes in other parts of the system.

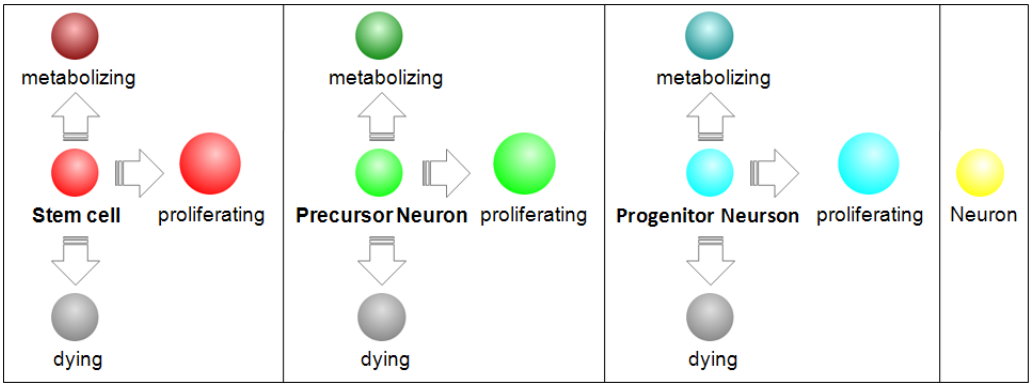


Figura 4: Cell representation: we only visualize the metabolizing, proliferating and dying states.

The *GUINiche* class extends the *GUIState* class and acts as a container of the 3D models, providing the graphical representations of the virtual environment. In our system, the virtual environment is the niche, which is represented by a 3D grid discretized in uniform dimensions of grid cells. Each grid cell can contain one cell at most. The *SimplePortrayl3D* class is responsible for portraying the objects within the 3D grid. MASON provides several

classes that specialize *SimplePortrayl3D* class (e.g. *SpherePortrayal3D, CubePortrayal3D*, *ConePortrayal3D*, etc.), allowing the drawing of simple primitives in a simple manner. The *CellPortrayal* interface guarantees the link of the model and visualization components by forcing the portrayals to define which class from the model component they are responsible for representing graphically (e.g. *StemCellPortrayal3D* is represents the *StemCell* class).

The cell entities are represented by sphere primitives. Each kind of cell in the conceptual model is represented by different colors. Each state of the cell life cycle is represented by different tones of the same color and some of them by different sizes. Figure 4 shows how we represented the cells. Note that not all states of the cell life cycle are visualized, nevertheless they are represented in the conceptual model. Also note that the proliferation state is represented by larger sphere. This is a more faithful representation of the entities.

The strategy to simulate the spatial self-organizing developed in the previous system was reformulated and extended to a 3D space. The real life, spatial self-organization of stem cells is roughly explained as follows. The stem cells divide itself into two news cells, due to the mitoses division process (Figure 5). Both cells assume new positions: one assumes the parent position and the other an adjacency position. Before the division process, the cell grows up, pushing its neighbors away and occupying the required space for the new cell.

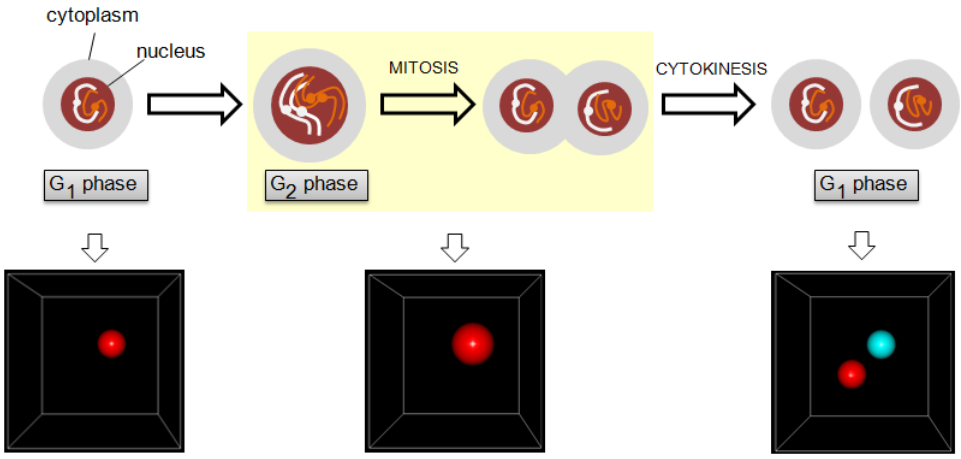


Figura 5: Division process: on the left, the figures represent the division steps in real life; on the right the figures are their representation in the system.

In order to model this spatial self-organization we must find the best direction in which the new cell is going to push into its neighborhoods. The best direction criterion is to determine out of all possible directions the one that minimizes the cell effort. The cell effort is minimize by choosing the closest empty cell grid in its neighborhoods. If there is more than one possible direction, we choose randomly among the possible directions.

The niche is now extended to a 3D grid. Figure 6 shows the 2D case in which there are 8 possible adjacency directions. In the 3D space, there are 26 possible directions. It is more than three times the number of directions in the 2D space, which makes the problem computationally intensive. Note that the problem was already an exponential complexity time due to the cell division process that duplicates the number of agents at each iteration.

The spatial self-organizing strategy is described in detail in the algorithm 1. Through the cell's current position, represented by a triple (x, y, z), which is dividing itself, we find

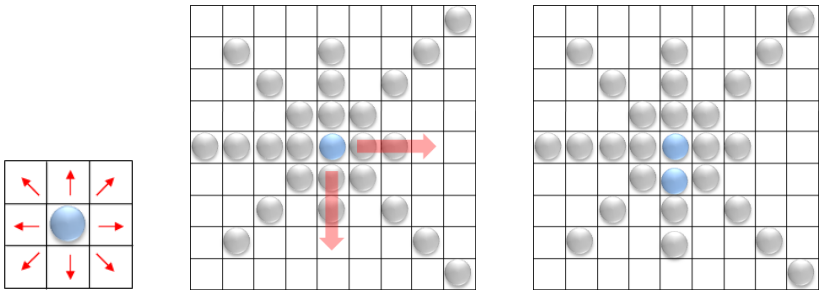


Figura 6: 2D spatial self-organizing simulation strategy: (a) it shows the 8 possible directions in a 2D space; (b) it shows an example with two possible directions in which the cell can move; (c) it shows the space reorganized after the south direction was selected.

which among 26 possible directions minimizes the cell effort. The 26 possible directions are represented by canonical base (e.g. the north direction is represented by $(1, 0, 0)$; the southwest direction is represented by $(1, -1, 0)$ and so on). Once the best direction is defined, the cells are shifted away by 1 unit in the defined direction. So, the new cell is inserted in the free adjacent cell grid.

Let d_k the number of possible direction in dimension k (e.g. when $k = 3$, the d_k is 26). The best case of the algorithm 1 occurs when the cell that is dividing itself is located in the center of the empty grid and it visits all empty adjacent cell grid, that is, it runs, in at least, $\Omega(d_k)$. The worst case occurs when the cell that is dividing itself is located in the center of the full grid and it visits the cells grids up to the limit of the grid in all d_k directions, that is, it runs ,at most $O(\max\{width, height, length\}$.

5 Advances on the Visualization Component

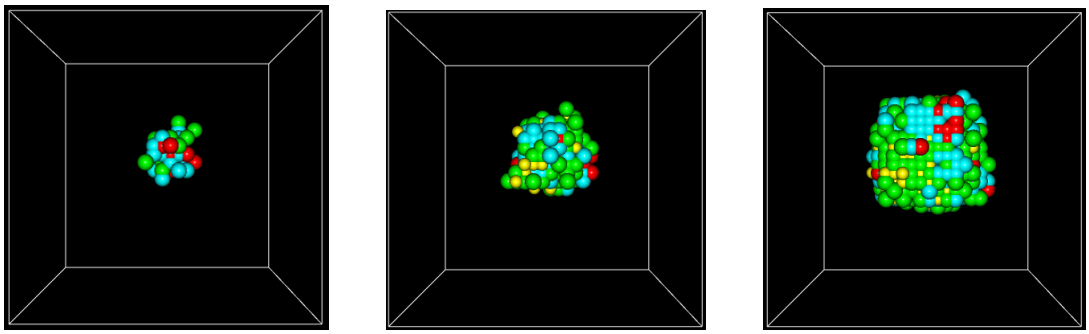
The model and visualization components architecture reorganization due to the incorporation of the MASON toolkit and the reformulation of the spatial self-organizing and its extension to a 3D space represent the major improvements of the visualization component. Nevertheless, it is worthwhile to describe other important improvements. In this section we present the visualization performance and the GUI interface improvements.

MASON toolkit [13] uses the Java3D graphic system [1] to render 3D models. One of the Java3D features is hardware-accelerated renderers. This feature combined with the improvements in the model and visualization components resulted in a significant improvement in visualization performance. The current visualization component can visualize up to thousands of cells, while the previous component could only dozens of them. Tests were carried out on a notebook on which the Windows Vista Ultimate operating system was installed; the notebook has an Intel Core Duo 2 processor with 2GHz clock and 2GB of RAM memory. We execute the simulation in a 3D grid with dimensions $50 \times 50 \times 50$ which, allows visualizations of up to $125,000$ cells. Figure 7 shows some screenshots made during a simulation.

The GUI interface has been improved by addition of: a console panel that contains useful settings for the simulation; an image and video capture; and 3D navigation tools.

Algorithm 1 3D Spacial Self-Organizing Algorithm

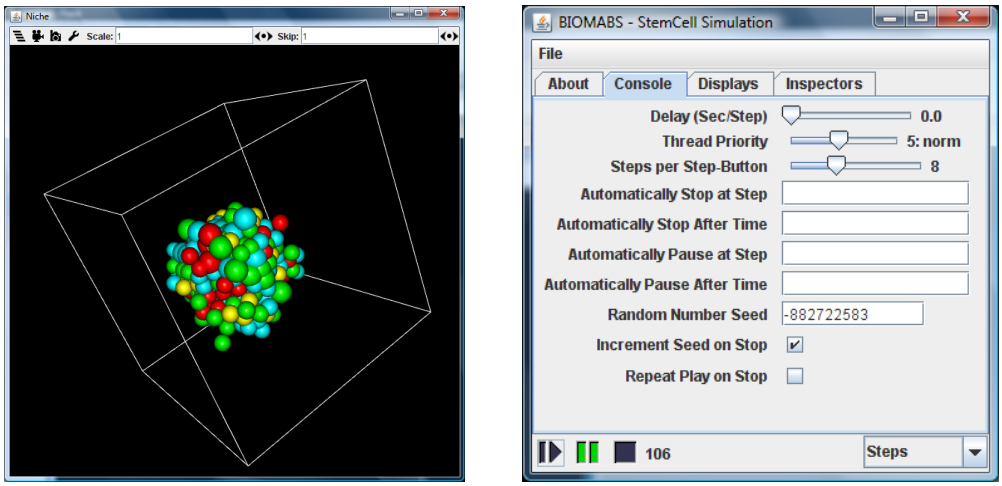
```
1: function FindBestDirection(grid[width][height][length], x̄, ȳ, z̄): direction
2:     closest ← max(width, height, length)
3:     possible_directions ← {}
4:     for i ← 1, 26 do
5:         distance ← 0
6:         x ← x̄, y ← ȳ, z ← z̄
7:         while x, y, z inside grid dimensions do
8:             if grid[x][y][z] is empty then
9:                 if distance < closest then
10:                    closest ← distance
11:                    possible_directions ← {i}
12:                else if distance = closest then
13:                    possible_directions ← possible_directions ∪ {i}
14:                end if
15:                break loop
16:            end if
17:            increment distance by 1
18:            adjust x, y, z values according to the direction i
19:        end while
20:    end for
21:    choose one direction randomly in possible_directions and return it
22: end function
```



(a) (b) (c)

Figura 7: Screenshots of a simulation: (a) at the initial; (b) the middle; (c) and at the end of simulation. Note the exponential growth of the number of cells.

The console panel provided three slides: the first allows the insertion of some delay time between each time step (in case the simulation is too fast); the second increases the priority of the simulation thread; and the third allows execution of any number of steps upon pressing the play button (when the simulation is paused). Image and movie capture tools are helpful to visualize the results of the simulation, without having to repeat the processing. 3D navigation tools (e.g. Arcball [14]) and zoom controls allows customized visualization of the simulation. Figure 8 shows a screenshot of the user interface of the current system.



(a) (b)

Figura 8: Screenshot of the user interface of the system: (a) it shows the display window and (b) it shows the console window.

6 Conclusions and Future Works

Scientific Visualization is a helpful tool for understanding complex problems. Therefore, the visualization of cell structures in several levels of details and the emergent behavior is fundamental for analyzing the differentiation cell process.

This paper presented advances to the visualization component of the initial version system presented in [9]. We showed in detail the visualization layer of the multi-agent based framework for developing the stem cell behavior simulators. With the stem cell behavior simulator it is possible to visualize the emergent phenomenon that arises from the agents' interactions.

The visualization component advances are very promising. The collaborators observed in the visualization tool the first emergent phenomenon, which is similar to the emergent phenomenon in vitro: the differentiated cells are located in the colonyŠs extremity while the specialized and stem cells are located at the colonyŠs center.

The proposed visualization system also suggests further developments. One of them is make the cell representation more photo-realistic. A 3D cell visualization in more levels of details showing the interaction between the inner structures would make the system complete and realistic. Another future work is to improve interactivity with the user. As in the previous version, we would like to allow the user to visualize the cell state and its simulation individually. We also want to improve the interactivity by means of allowing the interaction of the specialists with the live execution besides the basic functionalities such as play, pause, stop and increase/decrease the speed, by means of putting some substances in the niche and observing the emergent behavior. It would allow the self-organization optimization and the proposal of new hypotheses. Even more: generation of reports about the information visualized during the simulation process in several levels of detail, which could increase the comprehension about the process. The conversion of the system to the C++ language using the OpenGL [4] would provide visualization in real time.

Acknowledgments

Thanks are due to Bruno T. Avila for his most valuable assistance and the team of collaborators in this project, entitled LANDIC, which is the Neurogenesis and Cell Differentiation Lab, including stem cell researcher Stevens Rehen. The team helped us in the conceptual model validation, with some discussions around the research hypothesis, and will continue to work with us in the future works. The authors are supported by CAPES and CNPq.

Referências

[1] Java 3D API. `http://java.sun.com/products/java-media/3D/`, last acessed in November 2007.

[2] Collis J. C., Ndumu D. T., Nwana H. S., and Lee L. C. The zeus agent building tool-kit. *BT Technol Journal*, 16(3), 1998.

[3] Lales C., Parisey N., Mazat J., and Beurton-Aimar M. Simulation of mitochondrial metabolism using multi-agents systems. In *AAMAS Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics*, pages 137 – 145, 2005.

[4] Shreiner D., Woo M., Neider J., and Davis T. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, Reading, Massachusetts, 1999.

[5] dŠInverno M. and Saunders R. *Agent-based Modelling of Stem Cell self-organisation in a Niche*, volume 3464 of *Lecture Notes in Computer Science*, pages 52–68. Springer Berlin / Heidelberg, 2005.

[6] Bellifemine F., Caire G., Poggi A., and Rimassa G. Jade a white paper. *EXP in search of innovation*, 3(3):06 – 19, 2003.

[7] Vizzari G., Pizzi G., and Silva F. S. C. da. A framework for execution and visualization of situated agents based virtual environments. In *Workshop dagli Oggetti agli Agenti*, pages 22 – 25, 2007.

[8] Swarm Development Group. `http://www.swarm.org/wiki/Main_Page`, last acessed in January 2008.

[9] Gatti M., Vasconcellos J. E., and Lucena C. J. P. An agent oriented software engineering approach for the adult stem-cell modeling, simulation and visualization. *In Third Workshop on Software Engineering for Agent-Oriented Systems – SEAS*, 2007.

[10] Tom Misteli. The concept of self-organization in cellular architecture. *The Journal of Cell Biology*, 155(2):181–185, 2001.

[11] M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos. The repast symphony runtime system. In *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, 2005.

[12] Bandini S., Manzoni S., and Vizzari G. Situated cellular agents and immune system modelling. In *AAMAS Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics*, pages 78 – 90, 2005.

[13] Luke S., Cioffi-Revilla C., Panait L., and Sullivan K. Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 SwarmFest Workshop.*, 2004.

[14] Ken Shoemake. *Arcball rotation control*, pages 175–192. Academic Press Professional, Inc., 1994.