A Multi-User and Multi-Purpose CA Simulator

Hayk Nahapetyan, Jean-Pierre Jessel, Suren Poghosyan, Yuri Shoukourian

A Multi-User and Multi-Purpose CA Simulator

Hayk E. Nahapetyan
IIAP
NAS of RA
Yerevan, Armenia
hayknahapetyan@yahoo.com

Jean-Pierre Jessel
IRIT
University of Toulouse
Toulouse, France
jessel@irit.fr

Suren S. Poghosyan
IIAP
NAS of RA
Yerevan, Armenia
psuren55@yandex.ru

Yuri H. Shoukourian
IIAP
NAS of RA
Yerevan, Armenia
shouk@sci.am

***Abstract*—In this paper, a software package for cellular automata simulation, 2D/3D visualization with a shared work support system was introduced, that was designed considering the needs of researchers in both local and virtual laboratories. As an example of cellular automata, abelian sandpile model has been chosen. An appropriate software package has been developed using Microsoft .Net and C# enabling users to work at the same time on the same models independent of the geographical location of users within the public network.**

***Keywords*— CA, ASM, .Net, C#, multi-user, simulation, visualization**

I. Introduction

Cellular automata (CA) are discrete models studied in computability theory, mathematics, physics, complexity theory, theoretical biology and microstructure modeling. The concept of self-organized criticality was first introduced by Bak, Tang and Wiesenfeld in 1987 [1], and gave rise to growing interest in the study of self-organizing systems. Bak et al. argued that in many natural phenomena, the dissipative dynamics of the system is such that it drives the system to a critical state, thereby leading to ubiquitous power law behaviors. This mechanism has been invoked to understand the power law distributions observed in turbulent fluids, earthquakes, distribution of visible matter in the universe, solar flares and surface roughening of growing interfaces. The Sandpile models, being a class of cellular automata, are among the simplest theoretical models which exhibit self-organized criticality. A special subclass of interest consists of so called Abelian sandpile models (ASM). The Abelian property means that the final stable state of the CA is independent of the order in which the updates of cells are carried out. This property plays a key role during the numerical, as well as analytical studies of the ASM [2]– [4].

Many scientists and students previously presented research works on various types of CA, included ASM, and provided the relevant modeling and simulation [5]. Undoubtedly, it is vital for researchers to have tools for simulating the models under consideration. For this purpose, software solutions with appropriate functionality are required to visualize the models, also to perform simultaneous changes with provision of getting and viewing the results. Besides, the solution should provide logging the changes made during the model exploration; introduce required attributes for accounting each change, as well as memorize the model current state for further investigation. Researchers in virtual laboratories are in need of sharing and processing the same models at the same time independent of the team members geographical locations. There are a number of software solutions developed to meet the researchers needs.

For this purpose, NetLogo [6] can be selected as an appropriate development environment supporting multi-agent programmable modeling with provision of simulation and visualization of discrete models and cellular automata. NetLogo has its own programming language based on Lisp which allows users to create and develop their own models. Besides, Net Logo provides the researchers to work on the same model at the same time within local network (See Fig. 1).

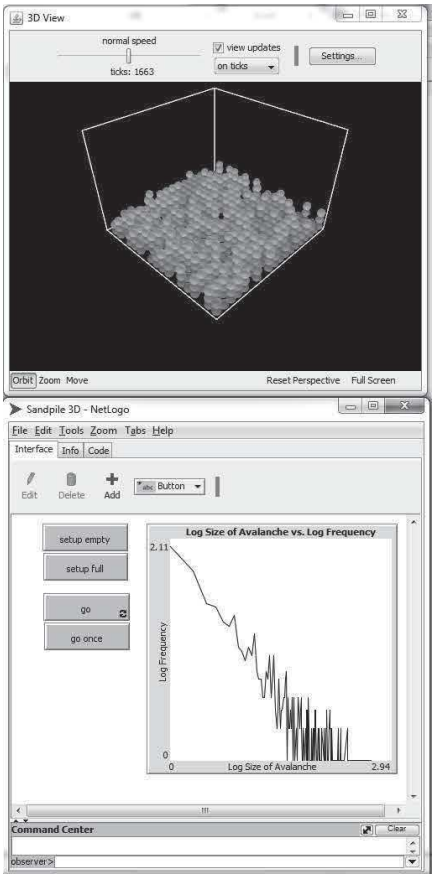

Fig. 1. Sandpile 3D simulation and visualization from NetLogo library.

Note that to obtain a 3D visualization, Wolfram Mathemat-

ica or MatLab can be used, meanwhile they do not support a shared work. Studies were conducted on the problem of information sharing, like the one introduced in [7] which presents a research on distribution and stream of large-scale 3D data in an efficient way. There are also studies regarding different implementations of collaborative virtual environments, as given in [8], where the importance of awareness and communication in collaborative virtual environments are evaluated. Along with the research done before, still there is a need for tools to merge the studies conducted, also to support modeling; simulation; 2D/3D visualization; access and availability aiming at provision of working on the same models within global networks (collaborative work).

II. Distributed simulation context (and state of the art)

As an example of cellular automata ASM has been chosen. CA simulator (See Fig. 2) supports 2D/3D visualization along with model rotation and zooms in/out possibilities. For creating a new model, the user selects ”File” from the top panel, chooses the ”New Sandpile Model” option, and then inserts a size for the parameter n. Changes, such as adding grains, are made by selecting ”Edit”, and then ”Add Grain” on the top panel (See Fig. 3). It is possible to add as many grains as it is required to a node with the given coordinates or with a layer selected.

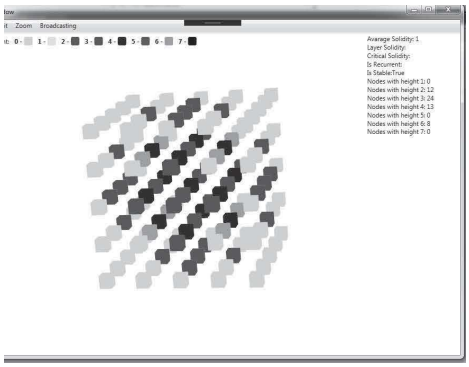

Fig. 2. CA simulator environment on the example of ASM.

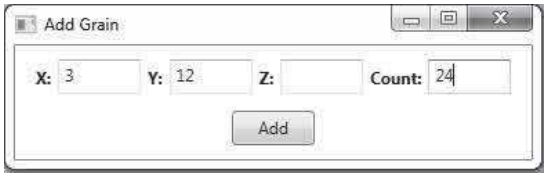

Fig. 3. Dialog for adding grain.

Microsoft .Net implements a strategy for web services to connect information; people; systems, and devices through software, thus making easier sharing and using the information between multiple websites; programs, and computers. Also, it is developed to establish client-server-client connections and to implement working on shared models. To proceed with sharing the model, user starts broadcasting from the top panel ”Broadcasting”, and then selects ”Start Broadcasting” by ordering the name of the channel. Meanwhile,the other users open the channels list from the top panel ”Broadcasting”; select the ”Connect to Chanel (See Fig. 4), and then choose the desired channel from the list. A prominent advantage of the software is that it provides simulation of all changes made during the model exploration, even in case of users lateness. Besides, accounting of attributes is implemented for each change in state, such as: average/layer/critical solidities; the model stability/non stability; belonging to recurrent states, count of nodes of the same height, etc (See Fig. 5).

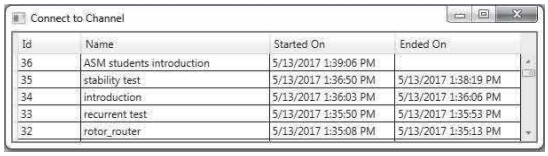

Fig. 4. Dialog of channels’ list.

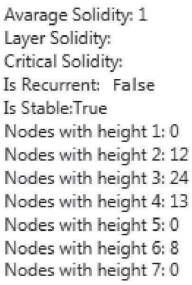

Fig. 5. Attributes.

III. Sandpile model

Consider an undirected graph $G = (V, E)$ described with the set of vertices $V = \{v_1, v_2, \ldots, v_N\}$ and the set of edges E. Each vertex $v_i \in V$ is assigned a variable h_i which takes integer values and represents the height of the sand at that vertex. h_i^{max} denotes the maximal allowed height for the vertex v_i in the graph G. For a d-dimensional lattice we take $h_i^{max} = 2d + 1$. C_T denotes the set of heights h_i which determines the configuration of the system at a given discrete time T. A configuration is called stable, if all heights satisfy $h_i < h_i^{max}$. The vertex v_i is called closed, if $h_i^{max} = deg(v_i)$, where $deg(v_i)$ indicates degree of v_i. The dynamics of the system is defined by the following rules. Consider a stable configuration C_T at a given time T. We add a grain of sand at a random vertex $v_i \in V$ by setting h_i to $h_i + 1$ (we assume that the vertex is chosen randomly with a uniform distribution on the set V). This new configuration, if stable, defines C_{T+1}. If $h_i \geq h_i^{max}$, then the v_i becomes unstable and topples losing h_i^{max} grains of sand, while all neighbors of v_i receive one grain. Note that if the vertex is open, then

the system loses grains. During the toppling of the closed vertices, the number of grains is conserved. Note also that toppling of a vertex may cause some of its neighboring vertices to become unstable. In this case those vertices also topple according to the same toppling rule. Once all unstable vertices are toppled, a new stable configuration C_{T+1} is obtained. If the finite connected graph G has at least one open vertex, then all vertices become stable after finite number of topplings. Moreover, the new stable configuration is independent of the toppling order. Therefore, the dynamics is well defined. Let $\hat{a}_i$ be an operator, which acts on sandpile configurations and adds a grain at vertex i. It can be easily be shown that $\hat{a}_i\hat{a}_j = \hat{a}_j\hat{a}_i$. This is the reason why the sandpile model is called Abelian.

IV. Application

As already mentioned, CA simulator was developed using .Net and C#. To facilitate the collaborative work in the global network, Microsoft Azure has been used. From the developers view point, the CA simulator may be divided into three modules: ”Visualization”; ”Local Simulation” and ”Service-client Architecture”. In order to visualize the model zooms in/out and provide rotation, **.Net**’s native libraries have been used (See Fig. 6).

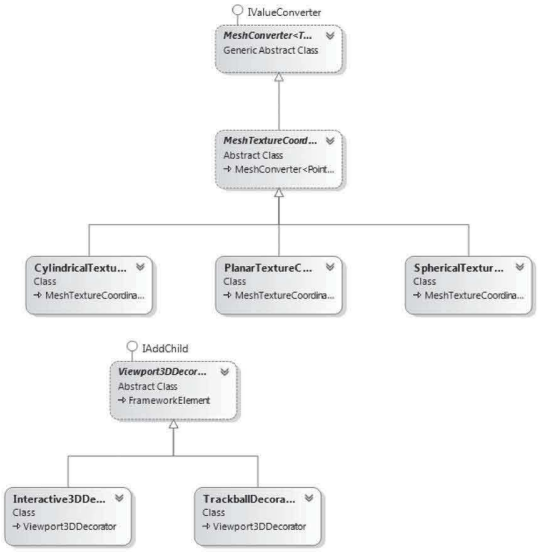

Fig. 6. Visualization class diagram.

Within the “Local Simulation” module, a **GuiHelper** class has been designed to provide the models creation; saving and loading; grains adding and toppling, as well as attributes counting, as follows:

```
public static class GuiHelper {
    event EventHandler GrainAdded;
    Viewport3D _mainViewPort; int _size;
    List<InteractiveSphere>
    _points; List<InteractiveSphere> Points

    //Initialize 3D view
    public static void Init(Viewport3D vp);

    //Creates model with given sizes
    public static void CreateModel(Size size);

    //Draws Sandpile model
    public static void DrawSandpileModel();

    //Add grain on Sandpile model
    public static void AddGrain(Position pos);

    //Adds grain from visual aspects
    private static void addGrainOnVertex
        (InteractiveSphere point);

    //Returns color regarded to grains count
    private static Brush GetColorByWeight
        (int weight);

#region File/String IO

    //Save model in file
    public static void WriteToFile() {}

    //Load model from file
    public static void LoadFromFile(){}

#endregion

}
```

Within the ”Service-client Architecture” module, we have a **BroadcastingHelper** class which includes essential functions to enable the broadcaster-subscriber connection, and a **SeService** class which implements the **ISeService** interface. The logic behind is to provide for a broadcaster to subscribe itself the same channel in order to get changes from other users. The channel keeps the whole information about changes made by all subscribers. Meanwhile ,when a new user starts to listen to that channel, he/she not only gets up-to-date knowledge of the model, but also he/she gets provided with all the changes made since the moment of broadcasting. For the channels’ database, SQLite has been chosen.

```
public static class BroadcastingHelper
{
    public static long SelfChannelId;
    public static long SubscribedChannelId;
    private static long LastActionId;
    private static Timer timer;
    private static ActionModel locker;
    public static EventHandler<> ChannelClosed;

    //Starts to listen to the given channel
    public static void ListenChannel
        (ChannelModel channel);

    //Disconnects from channel if it's closed
    static void timer_Elapsed
        (object sender, ElapsedEventArgs e)

    //Ends broadcasting
    public static void EndBroadcasting();

    //Disconnects from channel
    public static void DisconnectFromChannel();


    public interface ISeService
    {
        [OperationContract]
        long StartBroadcasting(string name);

        [OperationContract]
        void EndBroadcasting(long id);

        [OperationContract]
        void AddAction(long channelId,
```

```
        ActionType type, string data);

    [OperationContract]
    ActionModel GetNextAction(long channelId,
        long lastActionId);

    [OperationContract]
    List<ChannelModel> GetActiveChannels();
}
```

As already mentioned, CA simulator has been created on the example of ASM. There are two main ASM related functions: the **DrawSandpileModel()** which provides visualization of changes in already created model for ASM vision, and the **AddGrain(Position pos)** which supports changes performed by the user on an ASM model. It is quite easy to generate another CA model simply by manipulating the visualization and model modification functions. In order to make it a new CA available within a global network, a fewl functions of the **ISeService** interface should be adapted to the new CA model described within the **SeService** class. The sources of the CA simulator can be found in, Bitbucket under https://nhayk@bitbucket.org/nhayk/ca_simulator.git link.

V. Conclusion

In this paper, a software package, namely, "CA Simulator", for collaborative work implementation has been presented. The goal of the CA Simulator is to provide joint research of models under consideration. Features developed currently, are: simulation of ASM; visualization within 2D and 3D space; shared work on the same model at the same time within global networks; models' attributes counting. The concept of the multiuser simulator was introduced and implemented in a way to make the solution available and fitting to any other type of cellular automata. The solution presented is easily reproducible. Perspectives on the work will be outlined in the near future in order to make the simulator more user-friendly, as well as to increase its usability and scalability. Enhancements in visualization techniques will be implemented to make the simulator applicable for larger graphs.

VI. Acknowledgement

The authors are grateful to Dr. V. Poghosyan and Dr. Y. Alaverdyan for important discussions and critical remarks at all stages of the work. This work was supported by the State Committee of Science MES RA,in the frames of the research project No. 16YR-1B008 and Erasmus Project Armnie/ KA1 Mobilit internationale de crdits/Appel 2017 (Nr. 2017-1-FR01-KA107-036342).

References

[1] P. Bak, C. Tang, and K. Wiesenfeld,"Self-organized criticality: An explanation of the 1/f noise",*Phys. Rev. Lett.*, vol.59, no. 4, pp. 381384, 1987.

[2] V. S. Poghosyan, S. Y. Grigorev, V. B. Priezzhev, and P. Ruelle, , "Pair correlations in the sandpile model: A check of logarithmic conformal field theory", *Phys. Lett. B*, vol. 659, pp. 768--772, 2008.

[3] Su. S. Poghosyan, V. S. Poghosyan, V. B. Priezzhev, and P. Ruelle, "Numerical study of correspondence between the dissipative and fixed-energy Abelian sandpile models", *Phys.Rev. E*, **84, 066119**, 2011.

[4] V.S. Poghosyan, S. S. Poghosyan, and H. E. Nahapetyan, "The Investigation of Models of Self-Organized Systems by Parallel Programming Methods Based on the Example of an Abelian Sandpile Model", *Proc. CSIT Conference 2013*, Yerevan Armenia, Sept. 23-27, 2013, pp. 260-262.

[5] H. E. Nahapetyan, S. S. Poghosyan, V. S. Poghosyan, and Yu. H. Shoukourian, "The Parallel Simulation Method for d-dimensional Abelian Sandpile Automata", *Mathematical Problems of Computer Science*, vol. 46, pp. 117125, 2016.

[6] S. Tisue and U. Wilensky, "NetLogo: A simple environment for modeling complexity", *International Conference on Complex Systems*, Boston, May 1621, 2004.

[7] C. Desprat, J.-P. Jessel, and H. Luga, "3DEvent: a framework using event-sourcing approach for 3D web-based collaborative design in P2P."*International Conference on Web3D Technology (Web3D 2016)*, Anaheim, CA, ACM, july 2016, pp. 73-76.

[8] Thi Thuong Huyen Nguyen and Thierry Duval, "A Survey of Communication and Awareness in Collaborative Virtual Environments", *International Workshop on Collaborative Virtual Environments (3DCVE)*, Minneapolis, United States. IEEE, Mar 2014.