Mathematical Problems of Computer Science 48, 5–13, 2017.

An Example of a Multiplayer Serious Game on 3D Sandpile Model

Hayk E. Nahapetyan

Institute for Informatics and Automation Problems of NAS RA
e-mail: hayknahapetyan@yahoo.com

Abstract

Purpose of this paper is to consider cellular automata models as a base of a type of serious games.As an example of cellular automata, the Abelian Sandpile model on 3D graphs has been chosen. Moreover, a networked multiplayer gaming environment (SandGame) has been developed using .Net and `C#` language.

Keywords: Serious games, CA, ASM, .Net, `C#`, Multi-user, Simulation, Visualization.

1. Introduction

A brief survey of serious games reveals that there seems to be as many definitions available as there are actors involved, but most agree on a core meaning that serious games (digital) are games used for purposes other than mere entertainment. Another question of interest concerns the claimed positive effects of such games, or of applications from related and sometimes overlapping areas, such as e-learning, edutainment, game-based learning, and digital game-based learning. Besides, that serious games allow users to experience situations impossible in real life because of cost, time, safety etc., they can have positive impacts on the players' development of a number of different skills. Even so, it is not the case that all games are good for all learning outcomes. Empirical studies on the use of serious games in science education here [1] are conducted.

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 [2], 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 [[3], [4], [5]].

In the CA simulator, [6] a new module has been added as an example of a serious game, named SandGame. The SandGame, developed on the basis of Abelian Sandpile model, has a carefully thought-out educational purpose and is not intended to be played primarily for amusement, whereas at the same time, it is entertaining. The idea behind the developed model relies on various theorems regarded to Sandpile model, which improved the concept of learning and increased attractiveness. Undoubtedly, it is vital for universities and schools to have learning mechanisms equipped with scientific games.

2. Concept of Serious Games

Today, the term "Serious Games"is becoming more and more popular. A Google-search on "serious games" renders about 39,900,000 hits [2017-10-11]. Nowadays, the term itself is established, but there is no current singleton definition of the concept. Serious games usually refer to games used for training, advertising, simulation, or education that are designed to run on personal computers, mobile phones or even on video game consoles. According to Corti (2006, p.1), game-based learning(serious games) is all about leveraging the power of computer games to captivate and engage end-users for a specific purpose, such as to develop new knowledge and skills. It could be argued that for purposes other than purely academic, there is no need to define serious games. However, while different groups of researchers use the same term, they also appear to refer to different things. For example, in [7] Serious Game definition is like *"any piece of software that merges a non-entertaining purpose (serious) with a video game structure (game)"*. In another paper [8], Serious Game is formally defined as an *"Interactive computer application, with or without significant hardware component, that has a challenging goal, is fun to play and engaging, incorporates some scoring mechanism, and supplies the user with skills, knowledge or attitudes useful in reality"*.

3. 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 the 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 a 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 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.

4. SandGame

SandGame is based on the Abelian Sandpile model, which has been considered on 3 dimensional connected torus with $(n * n * n)$ size and 2 dimensional connected lattice with $(n * n)$ size, where n is the length (nodes count) by any direction of torus or lattice. "Connected" attribute for both graph types makes each node to have equivalent parameters as for neighbors, as well as for critical height. Besides, the model will not lose any grain, which grants the game to have a winner.

Consider a 2 dimensional lattice. Every edge with random direction on the lattice is assigned an arrow parameter, also contours in the graph are excluded. Let D_i be the number of arrows directed towards to the node i. Model will become infinitely unstable if each node topples at least once during the overall toppling. As each node has 4 neighbors, we put $4 - D_i$ grains on every node i. Total grains' count will be $4 * n^2 - \sum D_i = 2 * n^2$, which is an indispensable number of grains that makes the model become infinitely unstable. With the statistical methods [9], it is shown that in order to provide an infinitely unstable state, the required average count of the grains equals $2.12588... * n^2$. By toppling $2 * n^2 - 1$ grains on a 2-dimensional lattice, users need to topple $0.1258 * n^2$ grains in average in order to make the model become infinitely unstable. The same logic applied for a 3-dimensional torus, the indispensable count of the grains will equal $3 * n^3$, meanwhile the number of toppled grains will be $3 * n^3 - 1$.

The SandGame (see Fig. 2) supports 2D/3D visualization along with the model rotation and zooming in/out possibilities. Microsoft .Net implements a strategy for web services to connect information; people; systems, and devices via software tools, thus making easier sharing and using the information between multiple websites, programs, and computers. Besides, client server architecture has been implemented to facilitate working on shared models.

So, the game is the following. There is a model with sands on random generated vertices and there are players, that should play against each other. Each player will topple a grain on the desired node of the same model by sequence, in order to make the model become infinitely unstable. The player who will succeed in that will be the winner.

4.1 SandGame Guidline

This subsection is about the playing flow for SandGame. Here it is specified how the game is played, and all the interactions are indicated. The definition is divided into three sections: Pre-construction, Actions, Winner.

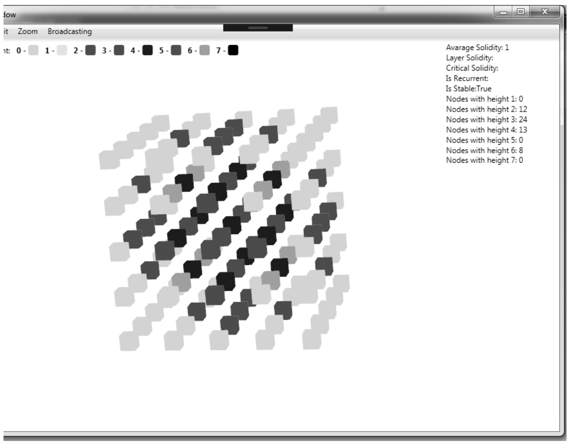
Pre-construction: For starting a new game, the player selects "File" on top panel, then chooses the "New game" option, and then inserts a size for the parameter n. A new model will be created and initialized with $3*n^3-1$ grains. To proceed with sharing the model/game, user starts broadcasting from the top panel "Broadcasting", and then selects "Start Broadcasting" by typing a name for the game (e.g., Harvard championship). Meanwhile, the other players open the channels' list on the top panel "Broadcasting"; select the "Connect to Chanel (see Fig. 4), and then choose the desired game 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 players' lateness.

Actions: After connecting to a channel players can start dropping grains by selecting "Edit", and then "Add Grain" on the top panel (see Fig. 3) by giving node coordinates. Players continue these steps until one of them becomes a winner.

Winner: The player, whose dropped grain destabilizes the model resulting in all nodes toppling at least once, becomes a winner of the game. In a word, the player wins when his dropped grain forces the model to switch his state into infinitely unstable.

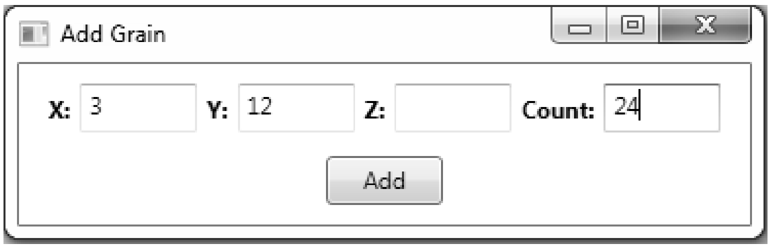
Strong background and theory behind the SandGame, makes it harder to win when you play against professional players. Each player can create his/her own strategy, but the winner will be the one, whose strategy will be better. For example, one of the simplest strategies could be toppling grains on the nodes with max heights, as we know that model will be in infinitely unstable state when each node topples at least once.

Besides, accounting of attributes is implemented for each change in state, such as: average/layer/critical solidities; the model stability/non stability; count of nodes of the same height, etc., (see Fig. 5), in order to facilitate decision making.



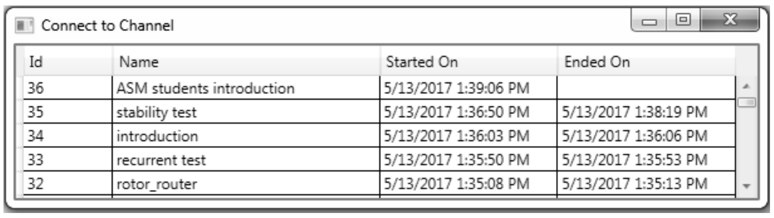
Avarage Solidity: 1
Layer Solidity:
Critical Solidity:
Is Recurrent:
Is Stable:True
Nodes with height 1: 0
Nodes with height 2: 12
Nodes with height 3: 24
Nodes with height 4: 13
Nodes with height 5: 0
Nodes with height 6: 8
Nodes with height 7: 0

Fig 1. SandGame on CA simulator.



X: 3 Y: 12 Z: Count: 24

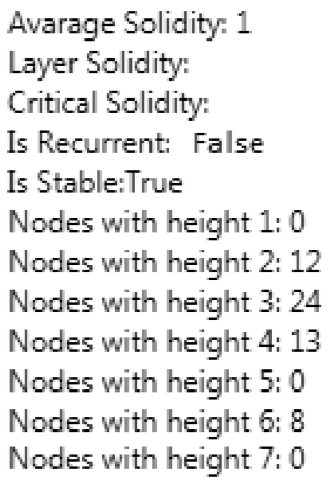
Fig 2. Dialog window for adding grain.



Connect to Channel

Id	Name	Started On	Ended On
36	ASM students introduction	5/13/2017 1:39:06 PM	
35	stability test	5/13/2017 1:36:50 PM	5/13/2017 1:38:19 PM
34	introduction	5/13/2017 1:36:03 PM	5/13/2017 1:36:06 PM
33	recurrent test	5/13/2017 1:35:50 PM	5/13/2017 1:35:53 PM
32	rotor_router	5/13/2017 1:35:08 PM	5/13/2017 1:35:13 PM

Fig 3. Dialog window for the list of games.



```
Avarage Solidity: 1
Layer Solidity:
Critical Solidity:
Is Recurrent:   False
Is Stable:True
Nodes with height 1: 0
Nodes with height 2: 12
Nodes with height 3: 24
Nodes with height 4: 13
Nodes with height 5: 0
Nodes with height 6: 8
Nodes with height 7: 0
```

Fig 4. Attributes.

5. SandGame from Developer Viewpoint

SandGame was developed under CA simulator, using .Net and `C#`. To facilitate the collaborative play in the global network, Microsoft Azure has been used. From the developers viewpoint, the SandGame 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.

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();

//Adds 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

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

//Loads 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 few 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.

6. Conclusion

In this paper, a serious game, namely, "SandGame", has been presented. The goal of the SandGame is to provide joint study/research of Sandpile on 2D/3D graphs. Features developed currently, are: simulation of ASM; visualization within 2D and 3D space; shared play on the same model at the same time within global networks; models' attributes counting. SandGame has been developed on the concept of the multiuser simulator (CA simulator), which was introduced and implemented in a way to make the solution available and fitting to any other type of cellular automata. Perspectives on the work will be outlined in the near future in order to make the game environment 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, for what the technique described here [10] can be used.

Acknowledgement

The author is grateful to Prof. Yu. H. Shoukourian, Dr. S. 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.

References

[1] Ch. Meng-Tzu, Ch. Jhih-Hao, C. Sheng-Ju and Ch. Shin-Yen, "The use of serious games in science education: a review of selected empirical research from 2002 to 2013", *Journal of Computers in Education*, no. 01 pp. 353-375, 2015.

[2] 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. 381–384, 1987.

[3] 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.

[4] 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.

[5] 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, pp. 260-262, 2013.

[6] H. Nahapetyan, J.-Pierre Jessel, S. Poghosyan and Yu. Shoukourian,"A Multi user and multi purpose ca simulator",*Phys. Rev. Lett.*, vol.59, no. 4, pp. 381–384, 1987. Proc. CSIT Conference 2017, Yerevan Armenia, Sept. 23-27, pp. 260-262.

[7] Djaouti Damien; Alvarez Julian; Jessel Jean-Pierre. "Classifying Serious Games: the G/P/S model" IRIT University of Toulouse, France.

[8] M. Graafland, J. M. Schraagen and M. P. Schijven "Systematic review of serious games for medical education and surgical skills training", Department of Surgery, Academic Medical Centre, Amsterdam, and Netherlands Organization for Applied Scientific Research, Soesterberg, The Netherlands.

[9] A. Fey, L. Levine and D.B. Wilson, "Approach to criticality in sandpiles", Phys. Rev. E 82, 031121 Published 15 September 2010.

[10] 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, 117–125, 2016.

Submitted 14.08.2017, accepted 28.11.2017.

Բազմաօգտատեր "Լուրջ խաղ"-ի օրինակ դիտարկված ավազակույտի եռաչափ մոդելի վրա

Հ. Նահապետյան

Ամփոփում

Այս հոդվածի նպատակն է դիտարկել բջջային ավտոմատների մոդելները որպես հիմք որոշակի "Լուրջ խաղեր"-ի համար: Որպես բջջային ավտոմատի օրինակ ընտրվել է Աբելյան ավազակույտի մոդելը եռաչափ գրաֆի համար: Ստեղծվել է ցանցային բազմաօգտատեր խաղային համակարգ (SandGame) օգտագործելով .Net and C# լեզուն:

Пример многопользовательской игры на трехмерной модели песчаной кучи

Г. Нагапетян

Аннотация

Цель этой статьи - рассмотреть модели клеточных автоматов как основу определенных серьезных игр. В качестве примера клеточных автоматов выбрана модель песчаной кучи на трехмерных графах. Разработана сетевая многопользовательская игровая среда (SandGame) с использованием .Net и языка C#.