Game-logic simulation based on cellular automata and flocking techniques

Sławomir Nikiel
Institute of Control and Computation Engineering
University of Zielona Góra
Address:
ul. Podgórna 50, 65-246 Poland
E-mail:S.Nikiel@issi.uz.zgora.pl

KEYWORDS
Cellular Automata, Flocking, Game Logic.

ABSTRACT

In the paper, a novel approach to simulation for game logic is proposed. It is based on the Cellular Automata augmented by grouping rules. The idea behind the proposed method uses local interaction on the cellular grid to obtain a global goal: a simulation of 'intelligent' behavior of groups. Locally acting bots organize themselves in groups to strengthen and perform attack on the game user. The proposed method can be used in action, role-playing and survival games offering real-time interaction with dozens and hundreds of bots. A prototype implementation of the method is discussed along with the performance analysis.

INTRODUCTION

We can observe the growth of the game industry. Entertainment powerhouses exploit to the limits current-gen consoles and multimedia mobile devices (Rasmusson 2007; Von Lehn and Heath 2003). They games are equipped with even better user interfaces, complex game stories and 3D graphics simulating real life environments (Teo et al. 2000). Game engines include combat, open worlds, engaging story and multiple characters (Buckland 2005; Rucker 2002; Schwab 2004). At the core of modern game, it is always the interaction with player (Johnson 2009). Current game environments offer interaction with a dozen of characters, which is sometimes explained by the rule of simplicity (Alexander 1964; Sirlin 2009). However, the action, role-playing and survival games require larger number of actors. When we look at the film industry, we can observe simulated interactions of hundreds or thousands of creatures. It is not straightforward to apply similar interaction techniques in real-time game environment. However it is possible to use potential of high-performance simulations to offer massive interactions. The capability of Cellular Automata (CA) to perform simulation of thousands of entities can be used to enhance the gamer experience. Many studies on CA have been published. The publication areas include the simulation and analysis of microwave propagation and the distribution of plants in ecological models, astrophysics and cosmology models. Some of them share real-time simulation property, very attractive from the game design point of view (Burks 1972; Choffurt and Culik 1984).

The organization of this paper is as follows. Section 2 provides the brief description of concepts and the main properties of the Cellular Automata and Flocking techniques. In Section 3, the core of the paper, the game logic and simulation procedure is presented. Software implementation of a prototype application for mobile devices is discussed in Section 4. The performance analysis is in Section 5, followed by Section 6, which is conclusion.

CELLULAR AUTOMATA AND FLOCKING

Cellular Automata methods have been developed to perform the simulation and analysis of complex interactions through relatively simple rules of inter-cell interactions. They are discrete dynamical systems whose behavior is completely specified in terms of the local relation. The CA space is typically represented by a uniform grid with each site or cell containing a limited number of information. The time of simulation advances in discrete steps, and the rules of transformation are usually expressed with a simple recipe. Given a suitable recipe, a whole hierarchy of structures and phenomena is modeled (Preston and Duff 1984; Toffoli and Margolus 1987; Wolfram 1996). The simulation procedure is based on a limited region of space that forms a mesh of cells. A number of objects is placed either randomly (with a given distribution) or manually. Each cell is empty or occupied by an object. In the simplest case, the cell is emptied or filled with a selected object with some probability. The final structure is obtained after several iterations, where each cell of the mesh is checked. More complex models can be constructed with CA. When we permit interactions between cells we obtain simulations of ecological models (Burks 1972). Cellular Automata is used to simulate large number of entities, but the interaction is always a matter of neighboring cells.

A different approach is used to simulate the complex motion of a flock of birds, herds of animals or group of fishes as seen in the natural world. This type of aggregate motion is rarely seen in computer games. The simulated flock is usually an elaboration of a particle system, with the simulated elements being the particles. The motion is created by augmented behavioral model.

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

Each simulated element of flock is designed as an independent entity that moves according to its local perception of the surrounding environment, the laws of simulated physics and a set of pre-programmed behaviors. The composite motion of the simulated flock of birds is the result of the cross-interaction of the relatively simple behaviors of the individual simulated birds (Reynolds 1987).

This paper explores an approach based on combination of CA simulation and flocking/grouping rules as an alternative to scripting the logic of bot navigation individually.

GAME LOGIC BASED ON CA

Game logic is usually perceived to be an Artificial Intelligence (AI) of the virtual enemy (*ro*-bot) (Buckland 2005). In particular it is responsible for the bot navigation and its decision making. This Section describes and game logic based on CA and grouping.

Derivation of the CA and Flocking Algorithms

The starting point for game logic based on CA is the automaton grouping cells of the same characteristics augmented by 'infection' capability. The result of this iteration is processed by another automaton equipped with the rules triggering attack for 'large' groups of bots. The control of bots is performed by Cellular Automata acting on a two-dimensional grid, where each cell can be a virtual actor. Three general rules govern the simulation:

- The grouping rule- based on the Moore neighborhood.
- The infection rule- two 'zombies' in the Moore neighborhood of 'human' infect him and turn into another 'zombie'.
- The attack rule- when the group of 'zombies' is large enough, it can attack the player's avatar. In the counting process, similar to Margolus neighboring is used.

Grouping and Infection Processes

Cellular Automata operates on a two-dimensional grid size $\boldsymbol{n}$ x $\boldsymbol{m}$ (to simplify $\boldsymbol{n} = \boldsymbol{m}$). Cells can have any number of states $\boldsymbol{s}$. The border conditions are described as periodical. The Moore neighborhood with excluded central cell is used.
The automaton rule- one iteration consist of repeated $\boldsymbol{n^2}$ following steps:

1. Randomly pick cell **C**, with less than **S** neighbors with similar state.
2. In the vicinity of **C** randomly choose neighboring cell **N**
3. if **C** -cell state is equal to **N** -cell state then go to step 6.
4. if **N** -cell is empty (state=0) – the growth factor **G** is counted: the number of neighboring cells with **C**-cell state on location **N** diminished by the number of neighboring cells with **C**-cell state on location **C**. If **G >= 0** then **C** is 'moved' to **N** (change **C**- cell state to 0, a **N**-cell state to **C**-state) and go to step 6.
5. Count the growth factor **G**: number of neighboring cells to **C** -cell on location **N** diminished by number of neighboring cells to **N** on location **C**. If **G ≠ 0** then swap **N** and **C** cells.
   Infection is performer by: If **C** is 'human' and **N** has **G** neighbors then **C**-cell changes its state to **N** – is changed into the 'zombie'. The value of **G** determines the 'infecting' power.
6. End if iteration

Attacks

Attack of 'zombies' is performed by moving the group of infected bots towards the game player's avatar. The 'infection' is caused by touching a given number of 'zombies' **G**. The group of 'zombie' bots should be large enough to perform the assault. Another 'triggering' condition is the limited movement of the avatar. The estimation of the zombie group count is performed by the automaton working on the output of the grouping algorithm presented in the previous Section.
The automaton defines a similar to Margolus neighborhood, the number of cells (9) is larger than in original (4) version. The size of zombie groups can be changed according to a given game level (with the parameter **S**- discussed in the next Section). The automaton can set the preferred direction of movement for group cells.

Choosing S-parameter

The algorithm was tested in a prototype test-application to help choose experimentally the appropriate **S** value. This parameter determines the minimum number of neighboring cells counted in the grouping process. The most interesting results were obtained for **S**=3 (after 500 iterations), as illustrated on Fig. 1.0

The grouping procedure worked when number of active cells is larger than 40. Smaller number of cells results in chaotic behavior (Fig. 1.0.a). For larger number of cells, the influence of **S** on grouping process is clearly visible. Smaller values of **S** result in numerous but smaller flocks of cells (Fig. 1.0.d) while larger values of **S** (6 or 7) end up in two big Groups of each type of cell (Fig. 1.0.b and 1.0.c).

SOFTWARE IMPLEMENTATION

Game Rules

Based on classical (win-lose) game economy, a set of rules for a prototype game was constructed:

- The main task of the game player is to save in limited time a given number of 'humans' from infection changing them into 'zombies'.

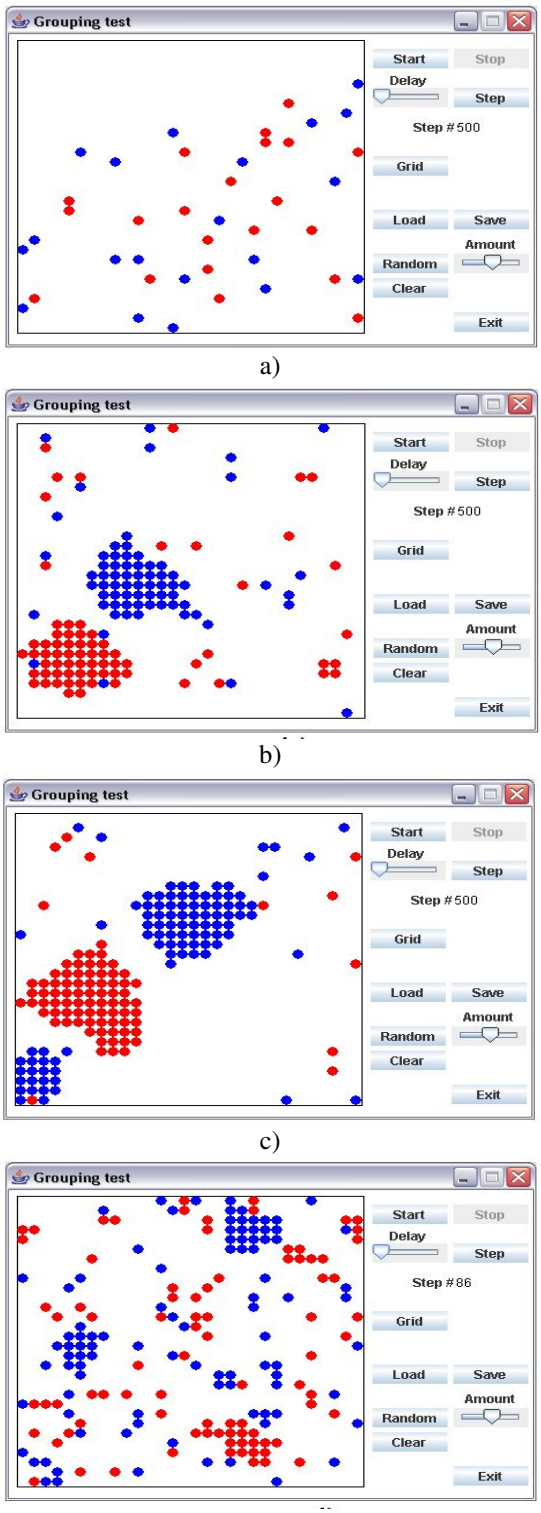

Figure 1.0 Experimental Results Obtained for the Grouping Simulation (after 500 Iterations): a) 20 Active Cells, S = 3; b) 70 Cells, S = 7; c) 90 Cells, S = 6; d) 90 Cells, S = 3.

- The player and 'humans' are attacked by 'zombies' that can form flocks to cumulate their 'strength'.
- During the game, individual 'zombies' are trying to group and if there is no apparent activity of the player they try to attack him.
- Contact of player's avatar and 'zombies' decreases its 'life power'. Game is over when it reaches zero. When a given number of 'zombies' contact 'humans' they turn them into new 'zombies'.
- The avatar can 'refill' life power picking up the 'med-aids' appearing in various places.
- The avatar has two weapons: the first one, with limited ammunition, can be used to destroy other game actors. The ammunition can be collected over the game area. The second one, unlimited, 'heals' zombies and turn them into 'humans'.
- When a 'human' is destroyed it increases the required number of 'humans' to save.
- There is a 'teleport' facility on game area, enabling teleportation to a random game spot.

Simulation

Simulation and testing environment for the Java prototype application was as follows:
Mobile phone SE K300i – with CLDC1.1 and MIDP2.0, with 30MHz Java processor (an equivalent to Intel Pentium III 540MHz), 512KB stack RAM, with screen of 128x128 pixels (Hi-color).

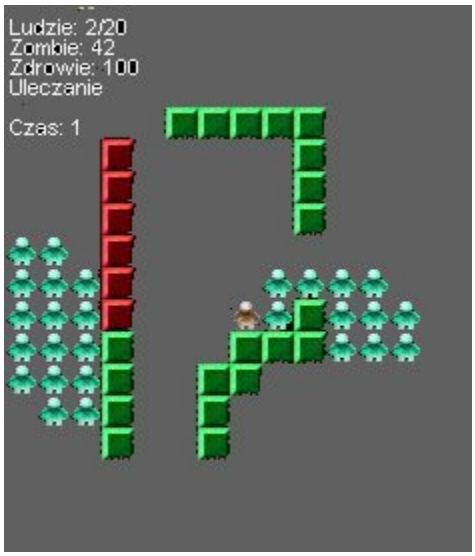

Figure 2.0 A Sample Screen Shot Depicting the Game Prototype.

The prototype application performed well, achieving real-time frame rate during the simulation. There were no visible differences for different RAM sizes (128 and 512KB).

SYSTEM ANALYSIS

In this paper, a game logic simulation is based on Cellular Automata and Flocking rules. Cellular

Automata has sufficient performance to simulate in real-time hundreds and thousands of interactions. The 'intelligence' of flocks give the additional feature for bots to gather in groups to prepare 'attacks' at gamer avatar and to disperse in case of his 'violent' response. The simulation process has to leverage the number of drones with their collective behavior. Mobile solution presented in previous section, is the most demanding programming environment, as far as application efficiency is considered (Brecken et al. 2003; Leiterman 2003, Mulholland and Murphy 2003). Thus, the working mobile game logic model opens doors to more intricate simulations in PC and console-based applications.

Game Logic

Given game logic has some drawbacks, the random choice of a cell on the CA grid with large number of 'empty' cells is a waste of processing power. This can be changed by introducing and checking lists of non-empty cells instead of the entire grid. Another problem is a possibility that the same cell is chosen several times during one iteration (performs the 'Lévy flight'). The possible solution to that problem is to attach to active cells information index with limitation of movement for each iteration.

Another drawback of the prototype implementation is the discreet movement of bots on the game arena. This is the result of straightforward implementation of CA grid to the game space. The solution to that problem may be the "off-the-screen" calculation of CA and then fluent animation of moving bots. Extension of grid size may also reduce the movement aliasing effect. Another problem that occurred during the simulation, is chaotic movement and 'looping' of bots around the obstacles. This is visible in the straightforward implementation of CA, more elaborated versions should reduce that artifact.

Efficiency Analysis

The CA algorithm is able to perform simulation of large number of cells on the grid. The performance of proposed CA plus flocking game logic simulation is examined in terms of its evaluation parameters. The performance comparison is done in terms of frame rate at Table 1.0 and 2.0. The Tables show time comparison of the 'clear' CA grid and filled CA grid and needed generation time.

Table 1.0 Test Results for a Mobile Phone

	Map 30x30 No 'zombies' and 'humans'	Map 60x60 No 'zombies' and 'humans'
No.	**FPS**	**FPS**
1	56,70	54,60
2	56,34	55,09
3	55,27	55,03
4	57,00	55,10
5	55,90	55,15
Average:	56,24	54,99
	Map 30x30 20 'zombies' and 5 'humans'	Map 60x60 20 'zombies' and 5 'humans'
No.	**FPS**	**FPS**
1	45,01	44,99
2	45,13	45,54
3	45,90	45,01
4	45,67	45,49
5	45,78	45,37
Average:	45,50	45,28

Table 2.0 Test Results for a Mobile Phone

	Map 30x30 40 'zombies' and 10 'humans'	Map 60x60 40 'zombies' and 10 'humans'
No.	**FPS**	**FPS**
1	34,67	34,86
2	34,55	34,87
3	34,70	34,57
4	34,23	34,01
5	34,17	34,17
Average:	34,46	34,50
	Map 30x30 80 'zombies' and 20 'humans'	Map 60x60 80 'zombies' and 20 'humans'
No.	**FPS**	**FPS**
1	24,96	24,39
2	25,01	24,45
3	23,91	24,58
4	24,78	24,10
5	24,59	24,17
Average:	24,65	24,34

CONLUSIONS

The game programming delivers constant challenges in the field of simulating logic and behaviour of bots. The paper proposes an alternative approach based on Cellular Automata and flocking techniques. The novel scheme utilizes local interaction of bots on the cellular grid to obtain a global goal- the 'intelligent' behavior of bot groups. The performance of the method allows user interaction with dozens and hundreds of bots, extending programming possibilities in action, role playing and survival games. It is possible to greatly enrich the interaction in gaming. The author plans to develop further the method in order to support more demanding first-person shooter games.

ACKNOWLEDGEMENTS

Application prototype results are part of M.Sc. diploma thesis by Mr. Michal Jackowski, entitled "Implementation of CA model to game logic in mobile systems", at University of Zielona Góra, Poland, 2006.

REFERENCES

Alexander, C. 1964. *Notes on the synthesis of form*. Harvard University Press.

Brecken, D.; Barker B. and L. Vanhelsuwe L. 2003. *Developing Games in Java*. New Riders Publishing.

Buckland, M. 2005. *Programming Game AI by Example.* Wordware Publishing.

Burks, E. 1972. *Essays on Cellular Automata.* University of Illinois Press, Champaign, IL.

Choffurt, C. and I.K. Culik. 1984. "On real-time cellular automata and trellis automata" *Acta Informatica*, No. 21, 393–407.

Johnson, S. 2009. "Asynchronous design". *Game Developer*, (March issue), 64-65.

Leiterman, J.C. 2003. *Vector game math processors*, Wordware Publishing.

Mulholland, A. and G. Murphy. 2003. *Java 1.4 game programming*. Wordware Publishing.

Preston, K. Jr. and J.M.B. Duff 1984. *Modern Cellular Automata*. Plenum Press, New York.

Rasmusson, J. 2007. *Multimedia in mobile phones, architectures and trends*. Ericsson (Mobile platforms), Lund.

Reynolds, C. 1987. "Flocks, herds and schools: a distributed behavioral model". *Computer Graphics*, 21(4), 25-34.

Rucker, R. 2002. *Software engineering and computer games*. Addison Wesley.

Schwab, B. 2004. *AI game engine programming*, Charles River Media.

Sirlin, D. 2009. "Subtractive design". *Game Developer*, March issue, 23-28.

Teo, L.; J. Byrne and D. Ngo D. 2000."A method for determining the properties of multi-screen interfaces". *International Journal of Applied Mathematics and Computer Science*, Vol. 10, No. 2., 413-427.

Toffoli, T. and N. Margolus. 1987. *Cellular Automata machines, a new environment for modeling*. MIT Press, Cambridge, MA.

Von Lehn, D. and C. Heath. 2003. "Displacing the object: mobile technologies and interpretative resources". *In Proceedings ICHIM Paris*.

Wolfram, S. 1996. *Theory and Application of Cellular Automata*. World Scientific Publ., Singapore.

AUTHOR BIOGRAPHY

SŁAWOMIR NIKIEL is currently the Professor at the Institute of Control and Computation Engineering, Department of Electrical Technology, Computer Science and Telecommunication, University Of Zielona Góra, Poland. His research interest include virtual reality systems, game programming and multimedia.