Implementation of cellular automata for real-time generation of infinite cave levels

Nikita Kalinskiy
Software Engineering Bachelors
Innopolis University
Innopolis, Russia
n.kalinskiy@innopolis.ru

***Abstract*—This paper describes a simple but very efficient method of generating an infinite map for a Cave Crawler game based on the cellular automata (CA). The algorithm has proven to generate playable and well-designed tunnel-based maps. It has very low generation time, permitting realtime content generation.**

***Index Terms*—procedural world generation, cellular automata**

I. Introduction

Procedural content generators are bits of code written into the game that can create new pieces of game content at any time - even when the game is running! Game developers have tried to procedurally generate everything from 3D worlds to musical soundtracks. Adding some generation to your game is a great way to plug in extra value: players love it because they get new, unpredictable and exciting content each time they play. That is why PCG is extremely useful in Cave Crawler games. *Cave Crawler* is a four player, co-op, game. The game world is displayed in the same manner as in *Diablo* (bird-eye view), yet all four players play on the same screen. Cave Crawler is an abusive game in that players are led to believe that they are progressing towards the end of the game, while there is in fact no end to the cave tunnels. In order to achieve this effect reliably, the game must procedurally generate all the maps as well as the enemies the players have to face.

This paper presents an efficient real-time PCG approach for generating infinite maps consisting of cave tunnels. The motivation for this algorithm is that the core mechanic for the test-bed game requires infinite map generation. The PCG approach proposed utilizes cellular automata (CA). Cellular automata self-organize and determine the existence of floors and rocks in the cave map and offer a simple, yet efficient and reliable solution to real-time cave-map generation.

II. Cellular Automata Approach

A. Grid generation

The Cellular Automata PCG(CA-PCG) algorithm consists of three phase: generation of the grid, generation of its adjacent grids and connecting not accessible grids via tunnels. At this approach we consider the grid size to be 50x50 cells.

Firstly, a central grid that is a base point of CA-PCG, is populated with cells. Each cell has data about its own location in the grid(x, y coordinates) and the cell type(floor, rock or wall). All cells are generated as floor at the beginning. During their generation the cell can change its type to rock with an $r\%$(in this study r equals to 35%). This is done using a pseudo-random generator with a uniform distribution.

When all cells in the grid are generated, CA performs n iterations over the grid. In this approach n is set to 3. While iterating over the grid, CA calculates every cell *neighborhood value* aggregating the number of cell neighbors which are rocks. CA operates with a Moore neighborhood with a distance M. After checking all neighbors of a cell its *neighborhood value* is tested against the T value - rocks threshold. If the *neighborhood value* is greater or equal to T then the cell's type is changed to *rock*.

When CA completes all iterations the walls cells have to be defined. Walls are rock cells that have at least one floor cell in its neighborhood.

B. Content representation

Due to specifics of the CA-PCG the content is represented by four parameters:

- r: initial percentage of rocks in the generated grid
- T: rocks threshold against which neighborhood value is tested
- n: number of CA iterations
- M: Moore neighborhood size

We can also store the seed for each grid's pseudo-random generator so that using it and four parameters the algorithm can regenerate the same grid.

C. Adjacent Grids Generation

After the generation of the base grid its north, south, east and west grids are created using the same procedure as described in section 3.A. After pointers to these grids are stored in the base grid the final phase of the algorithm starts. Each grid is checked if it is accessible from adjacent grids and if not a tunnel is created by finding the closest floor cells to the border and connecting via tunnel of predefined width(e.g. 3 cells).

The last step is to perform additional n CA iterations of the whole 5 grids to remove any inconsistencies between grids and smooth-out the generated tunnels.

III. Evaluation

To test the efficiency of the CA approach we designed a performance measure as follows. Given that the operation occurs in realtime, computational effort via CPU time defines a meaningful heuristic of performance. The computational cost of CA operations on a Moore neighborhood is on the order of $(2M + 1)^2$ where M is the Moore neighborhood size. Given that the operation is performed on all the cells of the grid the computational effort equals $n(wh(2M + 1)^2)$, where w and h are the width and height of the grid, respectively and n is the number of CA iterations.

Random cave maps are generated in 1.410^{-4} milliseconds (average value out of 10 runs) while the CA-based algorithm generates the map in 4.110^{1} milliseconds on average making both very efficient for realtime PCG.

Figure 1 shows a map with a random generation of cells in the grid while Figure 2 shows a map that is generated using CA-PCG approach. Rocks and wall are colored red and white while floor has gray color. All other colored cells represents tunnels. It is obvious that CA generate playable, good-looking maps at a very low computational cost, whereas purely random generation does not. Such an outcome provides the first indication of the efficiency and appropriateness of the algorithm for real-time map generation.

Figure 3 depicts a 3 3 map generated with the proposed algorithm. This map was built in only 349 ms indicating the appropriateness of the CA algorithm for realtime PCG.

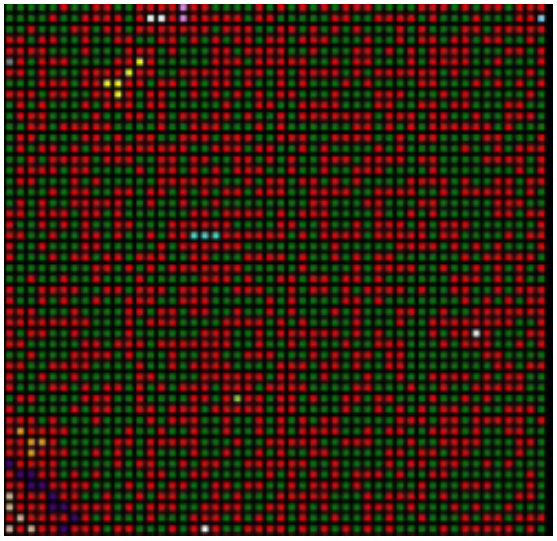

Fig. 1. Map that is generated during the grid initialization before CA iterations.

IV. Conclusions

The CA-PCG has been proven to be a very efficient algorithm with low computational cost and time that is very crucial for a real-time generation of infinite maps for 2D Cave Crawler games. The approach generates playable and good-looking maps with tunnels so that a player cannot stuck in one cave. At the same time it has good controllability since its four parameters can be easily tuned even via visual evaluation to meet the requirements of a game creator.

Fig. 2. Map generated with CA-PCG approach.

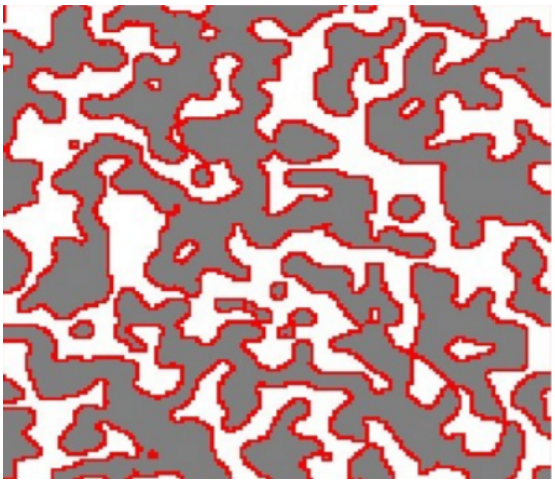

Fig. 3. 3x3 CA-PCG generated map with M=2, T=13, n=4, r=50%.

References

[1] Johnson, Lawrence, Georgios N. Yannakakis, and Julian Togelius. "Cellular automata for real-time generation of infinite cave levels." Proceedings of the 2010 Workshop on Procedural Content Generation in Games. ACM, 2010.

[2] Nikita Kalinskiy. Cellular automata implementaion, https://github.com/KN878/CACaves