“GAME OF LIFE” – CELLULAR AUTOMATON

Autor: Alexei TRUHIN
Coordonator: Mihail KULEV

Universitatea Tehnică a Moldovei

Abstract: *When we discuss about the origin of life or human being, we can't imagine the whole process and understand the structure of life. Today people can create their own world using programming. And this world is a part of cellular automaton technique.*

Index Terms: *Game of Life, Conway's Game of Life, cellular automaton.*

1. Introduction

Today, one of the most interesting directions in programming is the automatic programming. In the field of laboratory automation, the term is used to describe an algorithmic procedure that plans and executes an iterative procedure to automatically achieve a stated goal.

The basic idea of "Game of Life" is to start with a simple configuration of counters (organisms), one to a cell, then observe how it changes as you apply Conway's "genetic laws" for births, deaths, and survivals.

This simple example explains the general method of automation programming work. It shows us how people could create a new life which could evolve according to initial simple rules.

2. Rules

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors, lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The initial pattern constitutes the seed of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed—births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a tick (in other words, each generation is a pure function of the preceding one). The rules continue to be applied repeatedly to create further generations.

3. Example of patterns

A beginning pattern of three counters dies immediately unless at least one counter has two neighbors.

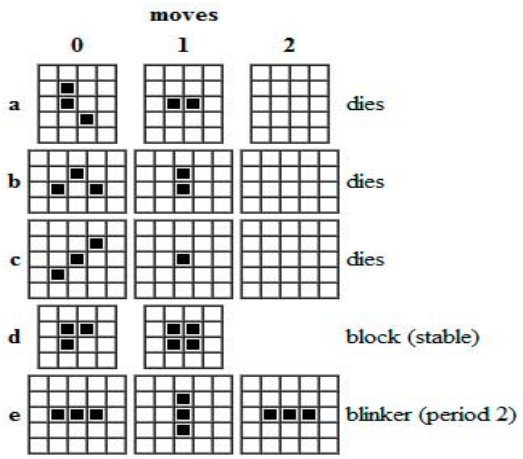

Fig. 1 Evolution of patterns of three counters

The first three [a, b, c] vanish on the second move. Pattern d becomes a stable "block" (two-by-two square) on the second move. Pattern e is the simplest of what are called "flip-flops" (oscillating figures of period 2). It alternates between horizontal and vertical rows of three. Conway calls it a "blinker".

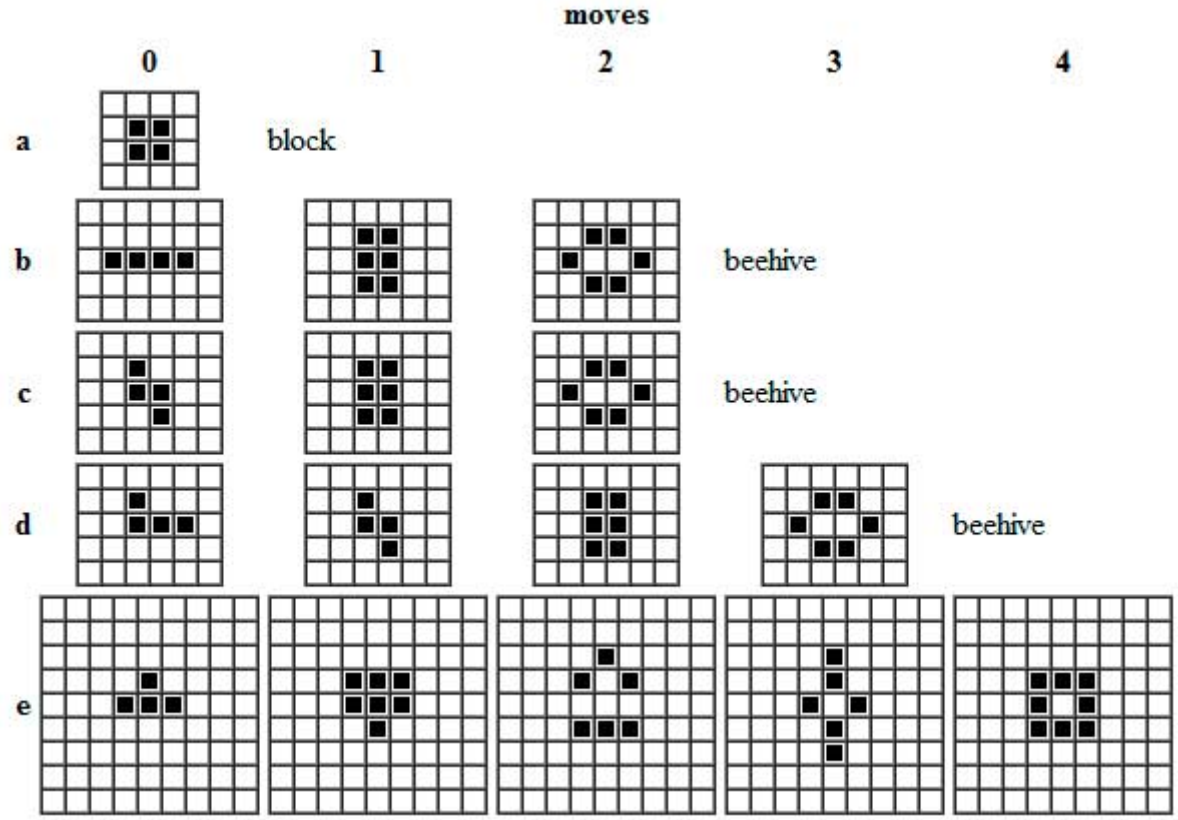

Fig. 2 Evolution of patterns of four counters

The illustration above shows the life histories of the five tetrominoes (four rookwise-connected counters). The square [a] is, as we have seen, a still-life figure. Tetrominoes [b] and [c] reach a stable figure, called a "beehive," on the second move. Beehives are frequently produced patterns. Tetromino [d]becomes a beehive on the third move. Tetromino [e] is the most interesting of the lot. After nine moves it becomes four isolated blinkers, a flip-flop called "traffic lights." It too is a common configuration. The illustration above shows the 12 commonest forms of still life.

4. Tools and code

In my work, I used the mix of OpenGL and C/C++ languages for its simplicity. It's possible through the GLUT library.

The full name of OpenGL is Open Graphics Library. It is a standard specification defining a cross-language, multi-platform API for writing applications and simulating physics, which produce 2D and 3D computer graphics.

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input.

The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. Getting started with OpenGL programming while using GLUT often takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs.

The general functions look like this:

```
int main(int argc, char ** argv){
    …
    glutInit(&argc, argv);
    glutInitWindowSize(screen,screen);
    glutInitWindowPosition(200,50);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Game of Life");
    glClearColor(0,0,0,0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, 1000, 0, 1000, -1, 1);

    createGLUTMenus();

    glutReshapeFunc( reshape );
    glutKeyboardFunc(keyboard);
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);
    glutMainLoop();
  return 0;
}
```

Fig. 3 General functions

It seems to be easy to understand and work with glut functions.

5. The program

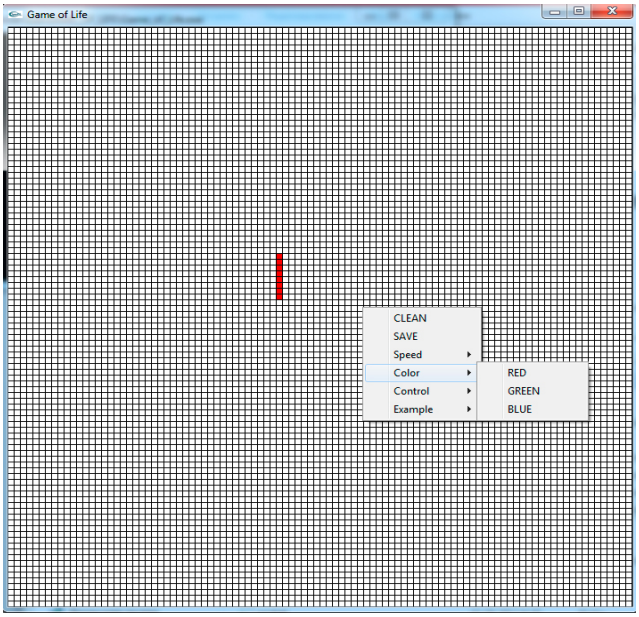

Fig. 4 Window of the program

It's a two-dimensional grid, with clean cells (dead) and colored cells (live). We can set the live cells with the mouse and change properties (like color or speed).

The general menu has 6 points:

- CLEAN – clean the grid.
- SAVE – save the cells positions in an external file
- Speed – change the speed. The numbers represent the period, in milliseconds, in which is made one move.
- Color – change the color of live cells.
- Control – the actions user can make and their hotkeys.
- Example – examples of some cell-generations. It cleans the grid and draws the example.

One of the interesting patterns is the Glider:

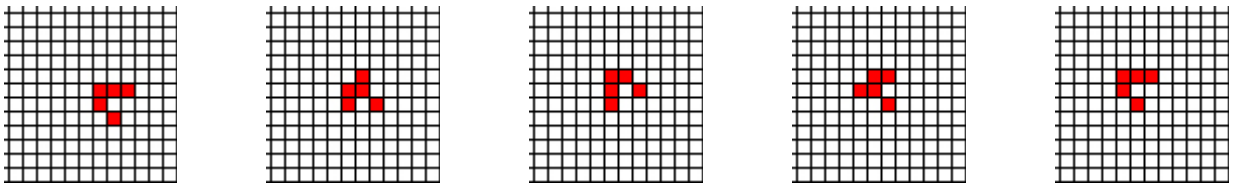

Fig. 5 Glider

One of my favorite patterns is Rocket. And it really looks like a rocket.

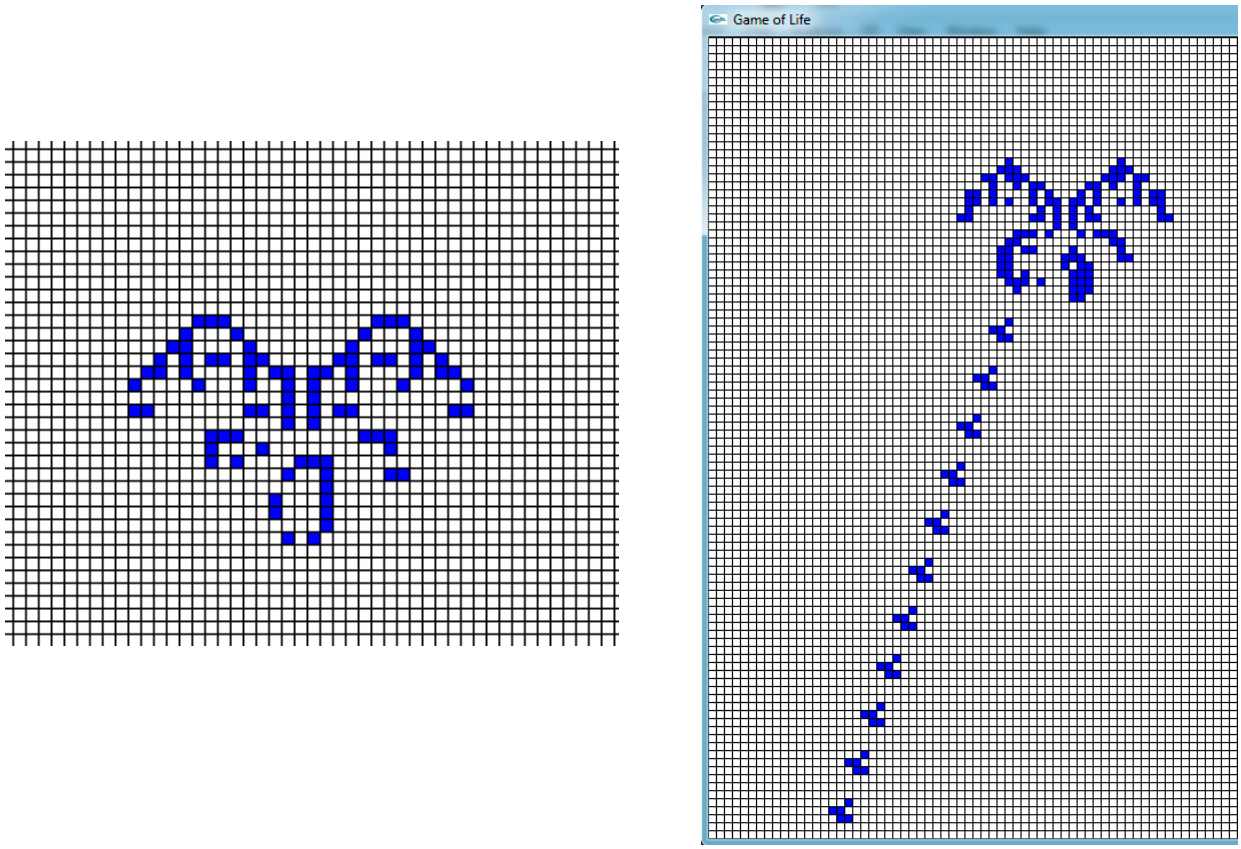

Fig. 6 Rocket: initial and after some generations state

It's an infinite cell pattern; it moves up and leaves after him gliders that look like gas from the rocket.

6. Conclusion

Elaboration of this project started from a simple game at first look and advanced in bigger idea with a strong and important significance. Game of Life is not just a game; it's a base of a new world of automaton. This example explains the base meaning of automaton programming.

References

1. *Conway's Game of Life* Available: http://en.wikipedia.org/wiki/Conway's_Game_of_Life
2. *OpenGL from wikipedia.*, Available: http://en.wikipedia.org/wiki/OpenGL
3. *OpenGL from the original site*, Available: http://www.opengl.org
4. *Information about Game of Life and patterns*, Available: http://conwaylife.com/wiki/Main_Page