Zhazira Umirkalykova

Computer Simulation Lab 2-3

Report Game of Life

The Game of Life is a cellular automaton created by British mathematician John Conway in 1970. It is a zero-player game, this means that the revolution of the next generation depends on the initial state, which is given at the beginning of the game or getting randomly by a program. This game was coded multiple time in different programming languages. In this laboratory work, I tried to code Game of Live in JavaScript using HTML properties for the suitable interface.

First, in order to start coding Game of Life, following rules of the game should be covered. According to Conway's Game of Life: "The universe of the Game of Life is an infinite two-dimensional [orthogonal](https://en.wikipedia.org/wiki/Orthogonal) grid of square cells, each of which is in one of two possible states, alive or dead, or "populated" or "unpopulated". Every cell interacts with its eight [neighbors](https://en.wikipedia.org/wiki/Moore_neighborhood), which are the cells that are horizontal, 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 underpopulation.
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 overpopulation.
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."

Why I chose JS, is because in JavaScript it was easier for me to draw the output screen of the game rather than in the other language. I used HTML5, JS, and random() function, etc. for simulation the Game of Life. Below will be the specification how the program works, how I tried to illustrate it.

First, in the <body> tag event onLoad='init();' used, where onLoad event occurs when an object has been loaded. It this case onLoad event runs the init() function while the program is loaded. Moreover, The HTML5 <canvas> tag is used to draw graphics, usually JavaScript. As long as, the <canvas> element has no drawing abilities of its own, the getContext() method is used, which returns an object that provides methods and properties for drawing on the canvas. This reference will cover the properties and methods of the getContext("2d") object, which can be used to draw text, lines, boxes, circles, and more - on the canvas. Hense, in the init() function, where I tried to check first the blinker pattern, if the blinder and random grids exist, the getContext("2d") object will be used.

Initially, to take control and see how the program works, it is necessary to in manual gives the initial value for the first generation, which is:

0 1 0

0 1 0

0 1 0

According to the rules given in the task above, the next generation should be:

0 0 0

1 1 1

0 0 0 this cycle should be repeated all the way down.

I chose the blinker pattern because I can control the initial input, according which I can predict the next generation and check whether the conditions truly work, without any bugs.

The function GameOfLife() responsible for the defining the board of the grid, which means that it will run over the array; find the board; stop the game. In addition, the next job of this function is a randomly defining first generation of a random grid. This will build by two methods: the round() method, which rounds the nearest integer number; the random() method, which returns a random number from 0 up to 1, not including it. Therefore, I will use it together like Math.round(Math.random()). There will be in two loops which are needed to go to the matrix and defining the next generation by the function nextGen. This function will be the main part of the program as it checks the neighbors of the cell. In order to do it, four for loops will be used which are go to the neighbor and run the counter to define the next generation.

There is a print function, where will be used the fixed parameter of width and height, it will be 8 in this simulation of the game. Finally, it will go through the cells checking cell and deciding whether it alive or dead. It will be determined according to the 4 rules, which were mentioned above. In order to illustrate the alive cells, I will use the fillStyle property of the JS, which returns a color of the cell, in this case, if it is the alive cell, the output of it will be white. Otherwise, the cell will be filled in black.

Finally, in the output will be two grids: the blinker pattern grid, with the 3x3 matrix; the random grid with the 8x8 matrix or the zero-player Game of Life. It is breaks or stops when the alive cells reach the board of the grid, or all cells will die.