CS 4802 - Assignment 1 - Game of Life

Ceren Savasan

1 My GitHub repository

https://github.com/cerensavasan/BioVis-Assignment1.git

2 How to run my code

There are 3 .pde files that are interconnected: Life.pde, Cells.pde and Grid.pde. The actual setup() and draw() functions are in the Life.pde file.

When you unzip the file, open all 3 .pde files by double clicking any of them. To run, make sure all 3 files are open in the same Processing.exe in 3 different tabs, and press the "Play" button.

You can pause the program by pressing 'p'.

You can clear the board and kill all cells by pressing 'c'.

You can reset the board by pressing 'r'.

You can left-click on the board to give a miraculous birth to a cell. Be careful, if there are no cells nearby, it will automatically die. If it is surrounded by too many cells, it will automatically die. You can only place it near 1 or 2 other cells to ensure its survival

.

3 Sources used and modifications

I started by reading the relevant sections in the recommended book, The Nature of Code by Daniel Shiffman. A lot of the information given was helpful but I wanted to see processing code, so I looked at the following examples to decide on the structure of my version of the Game of Life:

http://sandropaganotti.com/2010/03/20/a-processing-org-game-of-life/

https://processing.org/examples/gameoflife.html

https://github.com/justinzelinsky/life

https://github.com/betel/gameOfLife_processing/tree/master/gameOfLife

All of these examples helped me decide on how to implement the different sections of my program. The first link showed me the appropriate way to traverse through the entire grid. The second

link taught me how to initialize the cells on the grid properly and how to switch between two arrays. The third link gave me the reasoning behind splitting the grid, the cell and the actual game of life parts of the code for organization purposes, even though the code did not actually work the way it is in the repository. The last link helped me debug the stagnation problem I was having after a few iterations. The cells would just get stuck in the same configuration and just switch between two sets of positions.

I started from scratch and modified the initial ideas I had based on what I saw was the best practice among all of these examples. I based the functionality mostly off of the second link, with the structure resembling the third link.

My game of life is about an evolving organism conquering the entire board, slowly but surely.

(Then I got a little distracted by the patterns created by my code and looked into cancerous cell growth and rapid population growth of an invasive predator species when they are introduced to a new environment with an abundance of prey. Here are some of the resources I discovered to be very interesting:

Cancer cell growth: https://www.youtube.com/watch?v=IeUANxFVXKc

Bell frogs: http://naldc.nal.usda.gov/naldc/download.xhtml?id=49725&content=PDF

List of all known invasive species: http://www.invasivespeciesinfo.gov/animals/main.shtml

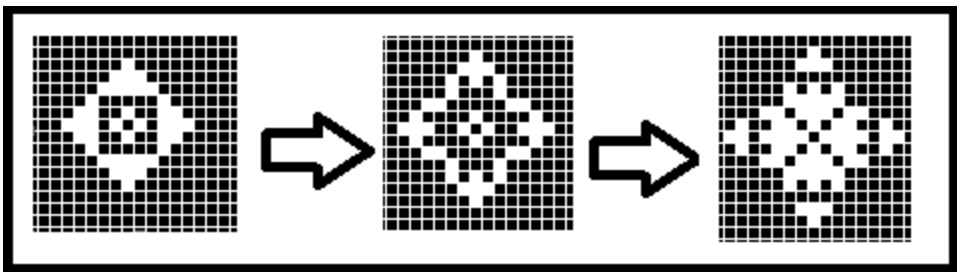
TED talk about invasive species: https://www.youtube.com/watch?v=GJW-Zgjz7G0)

4 GENERATION AND SET OF RULES THAT GOVERN BIRTHS & DEATHS

The initial position of all the cells is randomized by using random().

The rules of birth and death are the following:

- ⇒ A cell dies from overcrowding, which means that it cannot be neighbor to more than 3 other cells.
- ⇒ A cell is born when it is neighbored by 3 other live cells, if there is enough empty space around that can accommodate the birth without overcrowding.
- ⇒ A cell can die from loneliness. A cell dies if it has no neighbors.
- ⇒ A cell can cheat death by dying only to be reanimated: through the on/off toggle effect. An example of this is the following graphic:



This is a type of evolution I have added to the cell. I will explain this further in the biological aspect section.

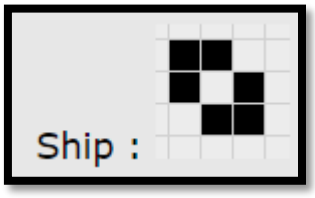
⇒ When paused, the user can manipulate the board by killing or giving birth to a cell. You can create a cell by left-clicking on any black empty space on the board. You can kill an existing cell by left-clicking on it.

⇒ You can clear the board to kill all cells.

⇒ You can reset the board and randomize all origin-cell positions to see a complete new progression.

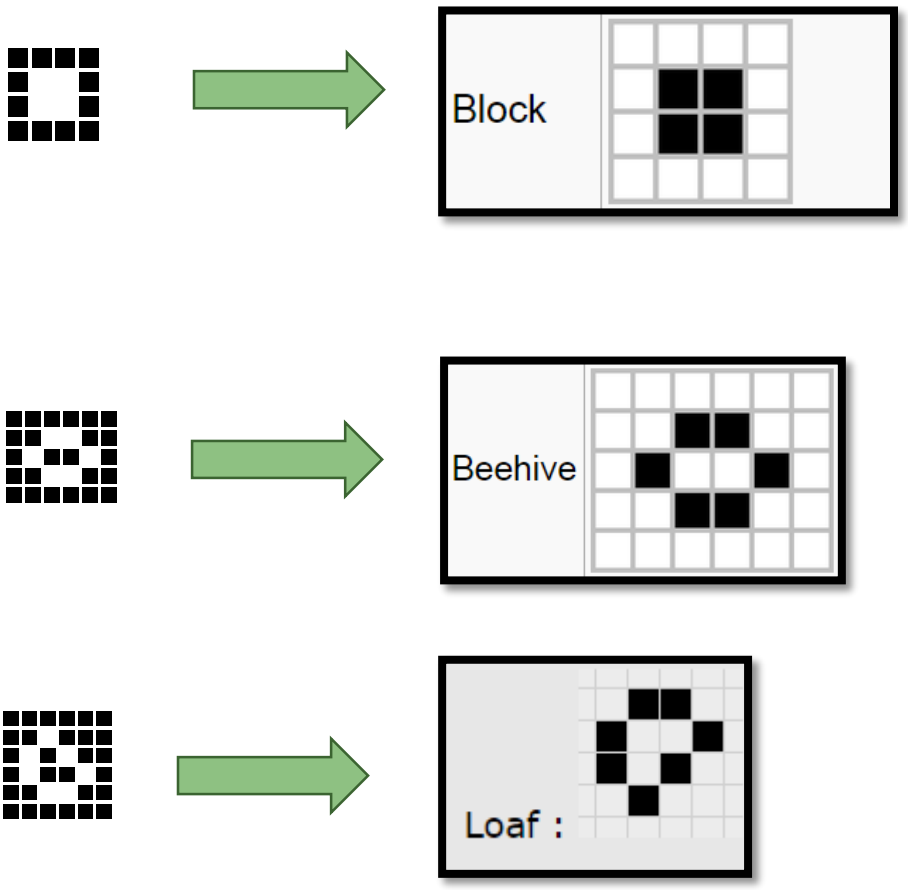
5 TECHNICAL ASPECT

This is sort of a cool way of implementing overgrowth through dying, then having the neighbors see that space has been opened, then occupying that spot, only to die again of overcrowding and repeat. The original game of life employed an algorithm that allowed stable set of cells to stay where it was because it needed 2 neighbors to stay alive but mine needs 1. Yet some cells would clash would each other and keep expanding while still staying in the general vicinity. I have not implemented a walking feature, but took a different approach on the evolution of the organism. I basically tweaked the stabilization parameters so even if I get the ship[1], or the boat[2] patterns, it iterates through and expands still.

1. 

2. 

I have a few variations of viable stabilized sets that occur. These are:



6 BIOLOGICAL ASPECT

When I discovered the pattern I achieved through some simple tweaks to the algorithm, it looked a lot like an invasive predator species. In isolated numbers, they remain stabilized. In large numbers, they expand rapidly and evolve into complex packs. The outlier cells collide with stabilized sets of cells and add them to their ranks. As long as there is prey, in this case black squares that signify empty space, they will increase in number.

This also created a species so powerful that it overtakes the entire space by rapid reproduction. A cell is not preemptive, so it is born regardless of the fact that it will die soon after from overcrowding. But the loss of these cells allows the organism, or the pack of cells, to expand on the board.