Advanced Programming - Assignment 3

Thomas Brandt, Alek Dimitrov, Virgjil Karaja, Andrew Kolenchenko, Anka Mkheidzhe, Monica Panigrahy

October 2022

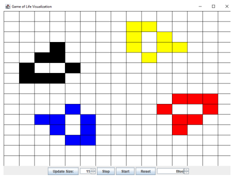
1 Conway's Game of Life (Rainbow variant)

We implemented Conway's Game of Life game, while adding the option to input several different colors in the grid cells and making it so that newborn cells take the average color of their parents. The game plays fully on a custom GUI that allows the user to input the grid dimensions (From 2x2 grid up to 100x100), the initial color of the cells via a slider, as well as having the buttons 'Step' and 'Start'. The 'Step' function simply forwards the game by one state and the 'Start' function continuously updates the game state every 400 milliseconds. Finally, to make the interface more user friendly, we made it so that cells can be colored simply by clicking on them once (and the corresponding color will be the one chosen from the slider) or removed by coloring a cell as 'Dead'[1].

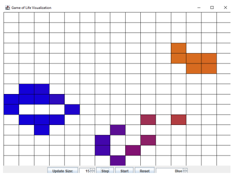
Our game follows the same general rules as Conway's Game of Life - Cells are either living or dead, with dead cells springing to life if neighbored by 3 living cells and living cells dying when neighbored by more than 3 or less than 2 living cells. Our approach only differs in that cells are also colored based on the colors of their parent cells, for which we make use of four public classes (plus one 'main' class to run our program) each describing a different level of our game - the individual cell and its properties (incl. color/position), the state of the grid of cells (incl. game dimension & all cell statuses), the window the grid is displayed in, and finally the GUI that incorporates the grid window along with user controls. Our approach has a time complexity of O(N^2) (with N being the side length of our grid) whenever updating/resetting/initializing our grid of cells (only O(1) whenever changing a single cell) as we must iterate through all NxN squares to either iterate the Game of Life or initialize cells in place.

2 Gallery

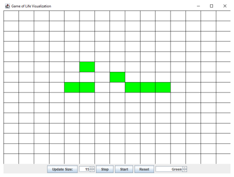
In the gallery, we present several interesting initial states, as well as the resulting coloring that happens occurs later



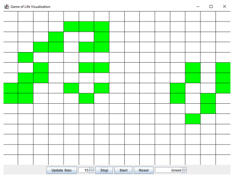
Initial state



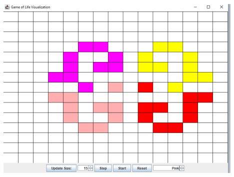
Turn 40



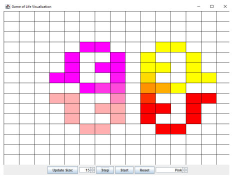
Initial state



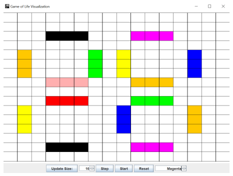
Turn 40



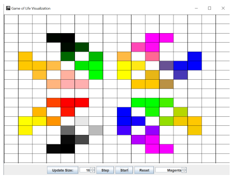
Initial Cloverleaf



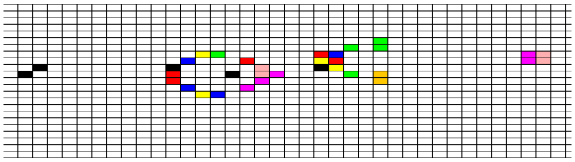
Cloverleaf Turn 40



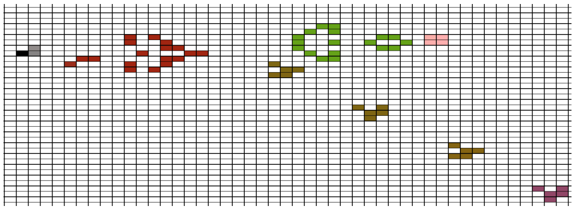
Initial Pulser



Pulser midpulse



(a) Initial Gosper glider gun



(b) Later state w/ created gliders

[1] We would also like to make special mention of the code of Prof. Paul Bouman, whose 'Creature World' code examples were especially helpful to us.