Project #1 – Cella Rule 150

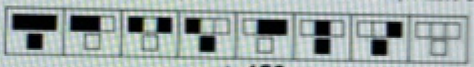
Introduction

This project is to write a program to display the generational progress of Wolfram's Rule-150 cellular automaton. The program will be written in Javascript and P5.JS (for animation) with an HTML web page for display.

The cella "growth generations" will be shown in a 2D grid of black and white cells. Each row after the top will show the next generation; the top row being the initial setup generation. Each generation is a stage in the "life" of the cellular automaton.

Rule-150

Wolfram's Rule-150 (from his 2002 book "A New Kind of Science") is based on a 1D array where each cell is "active". What happens to it depends on its current binary state (1 or 0, white or black – or the two colors of your choice that are distinguishable by us) and the states of its two neighbors; 3 cells in all. With 3 binary cells controlling what happens to the center cell in the next generation, there are 8 possible 3-cell configurations (8 = 2^3). A Rule needs to specify what happens to a cell with each of those 8 neighborly configurations. Rule-150 looks like this, (showing a section for each of the 8 3-cell triples) with the top row of each section being the old generation 3-cell triple, and the bottom cell representing what the "growth-change" was for the center cell of the old 3-cell triple in the creation of new generation (black is filled -- state 1, white is empty – state 0):



rule **150** Wolfram, 2002

This Rule format is interpreted as follows: leftmost is a section top 3-cell triple containing all black (1's, AKA filled cells). If the center cell and its two neighbors are in state 1, then in the next generation, the cell will change to state 0 (white, clear). Put differently, if the binary number represented by the 3 cells is 111 = 7; then the middle cell changes to state 0 in the next generation. A similar analysis is used for the other 3-cell triple states. Now, if we treat the next generation 1-cell states as 8 bits of a binary number, then these 8 1-cell configurations generate (from left to right) the binary bits 01011010, which equals a decimal 150. Hence, this set of 8 state transition rules based on 3-cell triple is called Wolfram's Rule-150.

Setup

Your program should initialize a 401 by 401 square grid to have all cells full (state 1). Each row will represent a growth generation, with the top row being the initial generation. Then set the top row's 201st (the center) cell in state 0. This represents the initial (0th == "seeded") generation. Make your team name visible on the web page above the grid. (If you wish, you can reverse the colors if that seems more pleasing.)

Running

After setup, your program should show on the next row down the next (#1) generation of Rule-150 operating on every row cell. It does this by analyzing each 3-cell triple, say, from the left – moving a 3-cell "window" to the right by 1 cell, and generating the correct center cell state on the row below the triple-cell. It would be convenient if you placed a cell border (say of yellow or red) just inside the cell of the next generation that you are creating so that it is easy to follow which cells are being generated. You can use the P5.JS draw() function to perform this animation change.

Borders: You should presume that a "neighboring cell" off the right or left side of the grid is empty. (This limits the right and left border cells to only 4 of the 8 possible configurations, each.) Similarly, continue running until all 400 rows (generations) have been shown. Then stop updating the animation.