Macintosh™
Cellular Automata

Thai Truong
@Play Corp.
truo8166@sundance.SJSU.EDU

Introduction

What is cellular automata and why do we need it? Wentian Li says that cellular automata are simple models to simulate complex behaviors of many degrees of freedom dynamical systems. Let me preface my next statement by saying that although we use the word simple, I am only scratching the surface of the field of cellular automata study. In laymen terms, it means the use of simple mathematical rules to simulate things that might occur at the cellular level. For the purpose of keeping the program simple, I am only going to look at one-dimensional cellular automata. There are of course, two and three-dimensional cellular automata but I won't get into it.

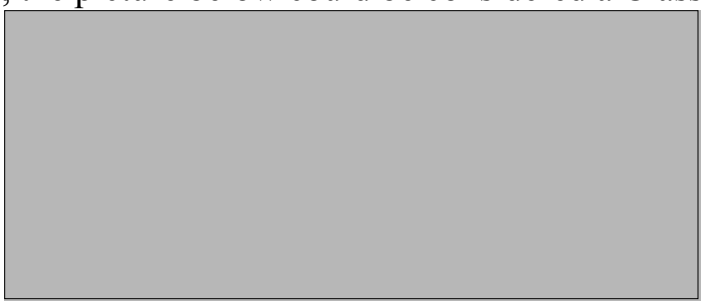
One-dimensional cellular automata is divided into four classes according to their behavior:

Class 1: Class one cellular automata are based on simple rules. They have the tendency to die out after a few generations.

Class 2: Class two cellular automata are based on periodic rules. They start out randomly but quickly become periodic. This can be seen in their repeating pattern.

Class 3: Class three cellular automata are based on random rules. They have the distinct appearance of having basically junk on the screen with no definable shapes.

Class 4: Class four cellular automata are based on complex rules. Complex because no one really knows the criteria for it to be considered a complex rule. Even my picture does not do it justice because it could still be considered a class 2 because the patterns are periodic and repeating. Li's explanation of "complex" is in the sense that they go through long transient period before settling down on the equilibrium states, or in the sense of the diversity of possible inhomogeneous configurations which they produce from different initial conditions. He also says that Class four rules are located between simple rules and random rules. Hence, the picture below could be considered a Class four or a Class two.



One-Dimensional Cellular Automata

In one dimensional cellular automata we only consider the cells in one line at a time. The other two adjacent lines are either used to calculate the results from or to receive the calculated results. Let us consider rule 30. Rule 30 is used when we want to look at a cell with only two neighbors, one on each side. Depending on the state of the left (L) and right (R) cells as well as the center (C) cell, the result will vary. If the L cell is 0 and the R cell is 0 and the C cell itself is 0 then the result will be zero. If the L cell is 0 and the R cell is 1 and the C cell is 0 then the result is 1 and so on.

L	C	R	Rule 30
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

If we were to apply rule 30 on a row of pixels on the screen, we would only concern ourselves with one pixel and its two neighbor at a time. The picture (below) shows what rule 30 produces if the it started out with only one pixel in the center (1 represents a pixel that is on and 0 is off) as row 1 shows.

1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
4	0	0	0	0	0	1	1	0	1	1	1	1	0	0	0	0
5	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0
6	0	0	0	1	1	0	1	1	1	1	0	1	1	1	0	0
7	0	0	1	1	0	0	1	0	0	0	0	1	0	0	1	0
8	0	1	1	0	1	1	1	1	0	0	1	1	1	1	1	0
9	1	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1

In row 2, we begin to see more pixels that are on. Look at the picture below and left, the top row represent a portion of row 1. The three rows below it represent the results in row 2 that we calculated from rule 30. The rows are 3 blocks in width only because we want to limit our scope to just three cells. If we look at only the three cells in row 1 that is within the border of the second row and compare those three cells to the L, C, and R values in rule 30, we can see that the result is 1. Try comparing the rest of the rows with rule 30.