Cellular Automata

What is a cellular automaton? There are of course many interpretations, but typically a cellular automaton is defined on a discrete grid, such as a vector in 1 D or an array in 2D. The discrete locations are called cells and typically they can only have a few values, typically a 0 or a 1. There needs to be a constructed set of rules that tells what happens to each cell based only on its nearest neighbors. Time is also discrete; each time step all cells are updated using the rules. That is it! Sounds trivial, but it is not.

In the 1930s, Alan Turing managed to prove that a simple theoretical machine that read a tape with a small number of symbols (he used: blank, 0, 1) and a set of rules on what to do at each symbol, such as go left, go right, erase or perhaps change its value, could in theory do any mathematical computation. The Turing machine is famous, but quite impractical. However, in the early 1940s, Ulam and von Nuemann at Los Alamos started using slightly more complex cellular 'machines' to do practical problems, especially in fluid dynamics.

To start thinking about cellular automata, we will look at a canonical form of automata. Consider a long vector of 0's and 1's. Each 0 or 1 occupies one location or cell. We now generate a new vector by looking at each group of 3 cells, starting at the left and computing a value in the new vector. We jog one cell right in the old vector each time, and eventually fill the new vector. By convention, we pad the ends with 0s so we can just fill the new vector end cells with zeros.

The calculation of the new vector cell is done with a 'lookup' table, that contains the rules on how to calculate a new cell based only on the old cell and its nearest neighbors. Here is an example of a lookup table, in the top row are all the possible groups of 3 cells, and underneath is the value of the new vector cell in the middle:

111, 110, 101, 100, 011, 010, 001, 000
x0x x0x x0x x1x x1x x1x x1x x0x

The 'x's are cells whose value is determined by the next (overlapping) group of 3. To illustrate, here are 3 vectors, the middle one is encoded from the one above, and the third one is encoded from the second, using the table above. You could continue creating new lines (vectors) from the line above *ad infinitum*.

0000100101110000
0001111101001000
0011000001111100
....

Believe it or not, this is out first cellular automaton. The entire automaton consists of generating new vectors from old, repetitively. What is staggering is that this simple 'machine' is 'Turing Complete', in other words, you can compute anything that is computable!!

To glimpse the richness of this automaton, we will apply the lookup table above to a vector with a single 1, and the rest 0s. We will apply the automata to:

.....000000001000000000.....

and see what we get? We will plot out each new row to form a 2-D plot, with the vectors forming the horizontals, and the vertical represents increasingly new vectors.

Although you can prove that you can calculate anything(!) with a cellular automaton, the catch (and it is a huge one) is that it is very difficult to dream up the rules which will do what you want. Additionally, many rules that may work, take huge amounts of time. (the wireworld computer is a great example)