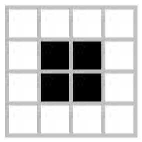
Clayton Seelenmayer Cellular Automata & Procedural Generation CSCI 422

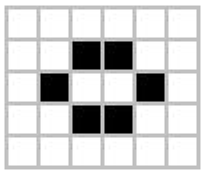
A cellular automata is a set of cells in a mathematical model with simple rules that governs their replication, translation, and destruction. It is often used to determine the structure of rigid bodies or state diagrams in modeling living organisms, or parallel processing, but by no means is this an exhaustive list of applications. Cells become saturated when context dependent data fields are filled, say colour, and play a role in how the rules replicate their behaviour.

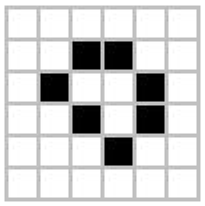
We observe patterns or notice methods that start to arise when running a cellular automata's algorithm. These phenomenons are called emergences and are a set of traits of a system that are not apparent from its components in isolation. They appear from extended iterative use of each cell's interactions, dependencies, or relationships. They form when placed together in a complex system and is a way of defining the long-term behaviour as it reproduces or breaks down. Emergence can be categorized by structures resembling those of Still-Lifes, Oscillators, and Spaceships.

For understandability, I'm going to choose Conway's Game of Life as a platform to demonstrate data manipulation in an automata like environment. Given an infinite 8-regular graph with saturatable vertices, we manipulate the data according to a ruleset by visiting each vertex for each iteration. Conway's game of life's ruleset can be condensed to the following three rules. Survival is when upon considering a particular cell, we saturate it if it has exactly 2 or 3 adjacent saturated vertices. Reproduction happens when we saturate an unsaturated vertex only when it is adjacent to exactly 3 saturated vertices. Death is where we unsaturate a vertex when the previous conditions aren't satisfied.

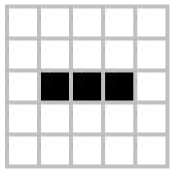
A Still-Life is a set of cells that remain saturated for a definitive or extended durations. As the simulation runs, we may notice regions of data that do not change. If all data in the entire space forms a still-life, then the system is said to be static. Still-life emergences may be destroyed if a collision with another translating phenomenon occurs. The following are a few common examples:

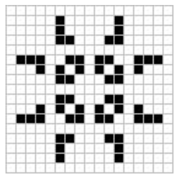

Block

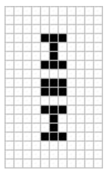

Bee-Hive


Loaf

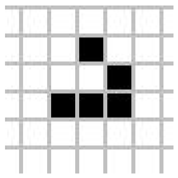
Oscillators are structures with repetitive pattern but remain bounded to a local region in space. They are a set of cells whose data behaviour remains self contained. In generating the automata, the isolated oscillator's data alterations will become predictable. The data set of which an isolated oscillator is defined over will repeat periodically. If an oscillator is completely unpredictable, then it is deemed chaotic.The following are examples of oscillators:

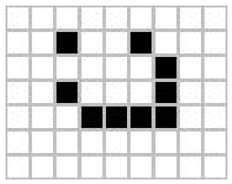

Blinker

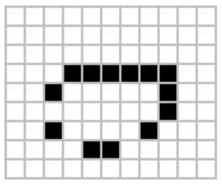

Pulsar


Penta-Decathalon (15-period)

Another interesting emergence is that of a spaceship. It is a pattern that forms on sets of data that traverses the space linearly and indefinitely until the data rules become interrupted. Similar to oscillators, spaceships have a period in which they repeat over, where the overall data governing the set of points has translated to another set with each period in a linear motion. The following are some examples of spaceships..


Glider


Light spaceship


Heavy spaceship

Cellular automata forms a space in which we can memoize data onto, and record how it changes over time. In the environment provided, we have the option to saturate or unsaturate a vertex. This can be related to binary strings, where saturated vertices are given a boolean value of 1. We can manipulate the behaviour of these cells in a spatial like manner using a deterministic set of rules that may be programmed to change over time. More importantly, this means that cellular automata can be represented using Turing Machines

For Conway's Game of Life and many other automatas, it becomes very hard to predict how a system will behavior, or how to predict chaotic patterns. Starting with a vertex, if an enumeration of all nearby cells can be formed, and the data of each cell is quantifiable, then we can construct a binary encoding of data for the automata. Given two encodings for a current, and next iteration, no known algorithm exists. This falls as corollary to the halting problem for if a deciding algorithm existed, it could be reduced to solve the halting problem by comparing the Turing Machine encodings.

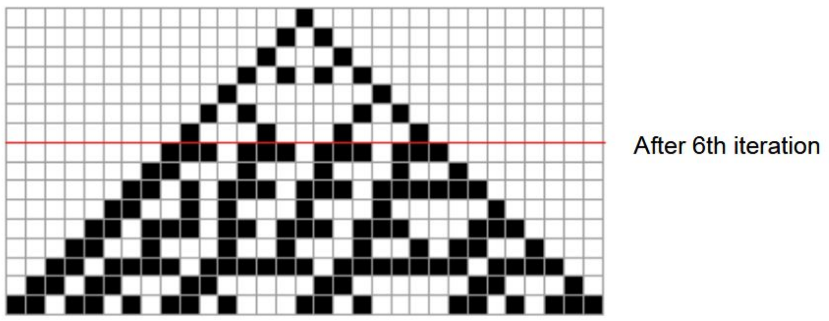
When designing more complicated automata, we can use construction methods in determining how to build an automata to suite a specific behaviour. I propose the definition of Turing Machine composition to be as follows upon input of 2 Turing Machine encodings:

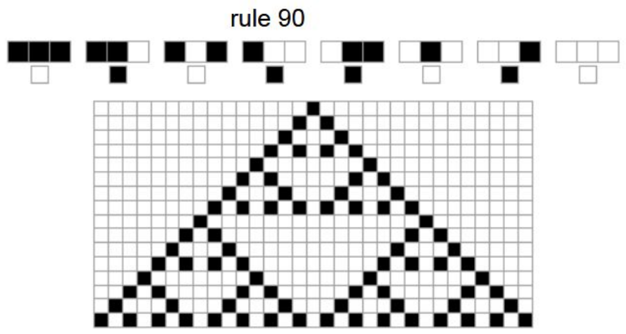
1) The new alphabet and state sets become unioned together.

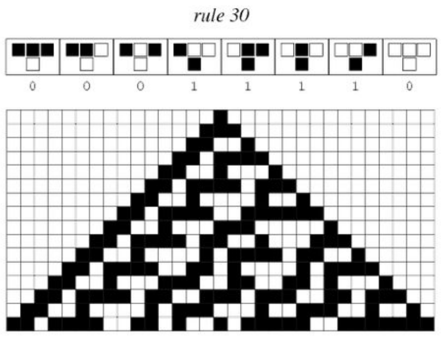
2) The start state is selected from the subordinate parameter, and the accepting states are selected from the dominant parameter.

3) Each terminating character x will be uniquely identified as a tuple (x, q) that masks the start state q in the dominant Turing Machine. All subsequent rulesets of the dominant Turing Machine will accommodate to carry this value. This preserves where the subordinate Turing Machine left off in it’s simulation.

The following example illustrates the composition of Stephen Wolfram’s “rule 90” onto that of “rule 30” at an arbitrary iteration of 6 for a 1-dimensional automata.







Now that we have a method for composing Turing Machines, we can now define a context free grammar who’s terminal points represent compatible automata rulesets. This gives the artist a method for generating a schedule on how these automata rulesets can interact. A context free grammar can yield fractal like structures for schedules in which we can compose these automatas. The following is a tree visualization of how a context free grammar can generate branching like structures despite schedules being read in the form of a linear extension.

A 3D tree representation of a CFG.

This completes the analysis of the featured method for procedural generation.