Generation of Mazelike Levels via Neuroevolution and Cellular Automata

Kaleb Osiel Alanis

Introduction

As technology becomes more robust and powerful, video games become more ambitious, especially in regards to the scale of games, with some like Minecraft having a potential world size which spans larger than the planet Neptune [1]. But with the introduction of larger scale games, the cost of adding content directly in such games intrinsically increases with such scale. My research is in response to David Ashlock's paper "Search-Based Procedural Generation of Maze-Like Levels". In Ashlock's paper, he describes methods to evolve direct and indirect representations of mazes with fitness functions of his design [2]. My effort is similar in that my goal was to evolve procedures that would generate such maze-like levels, rather than evolve the actual levels themselves. Specifically, I research how an Artificial Neural Network (ANN) performs with such task. I also see how see how cellular automata perform with ANN's serving as cell state rules.

Motivation

As previously mentioned, David Ashlock's methods of creating maze-like levels are very useful and create very interesting levels. So what would be the motivation in evolving functional methods to generate these levels? When I first sought this project, my initial motivation was not procedural content generation, but mapping alternative parameters to the generation of levels, specifically, player skill. In doing so, one would effectively map difficulty to the procedural generation of levels.

Definitions

To gather a better understanding on my paper and research, consider the following definitions which will facilitate in the articulation of my maze generation methods and fitness function design.

Definition 1:

w is the maze's world space. w is represented as a matrix of characters which is $N \times N$, dimensions can be assigned by the programmer, but in my experiments, $N = 64$. In my maze's representation, a member of w, called a *cell*, has a binary set of states where cell $x \in \{'*', '\#'\}$. $'*', '\#'$ represent a level floor and a level wall or obstacle, respectively.

Definition 2:

r, c are the row and column coordinate of the world space, respectively. $w[r][c]$ is a world cell. It then follows that $w[r][c]_{\in w} = x$ where $x \in \{'*', '\#'\}$.

Definition 3:

h is an array representation of a cellular automata neighborhood. The specifics of the cellular automata neighborhoods used in my experiments are given later.

Definition 4.1:

The function $taxicab(x_a, x_b)$ takes in two cells $x_a, x_b \in w$ and returns the shortest taxicab distance (manhattan distance) between them. If there is no traversable path between the two cells, the function returns ∞.

Definition 4.2:

A *cluster* is a set of cells S such that given a cell x:

$$x \in S \rightarrow \{x \in w \wedge x = '*' \wedge \forall_{y \in w, S}\,(taxicab(x,y) \neq \infty)\}$$

Definition 4.3:

A cluster S's *area* is the cardinality of S.

Definition 4.4:

A cluster S's *size* is the largest element of the set

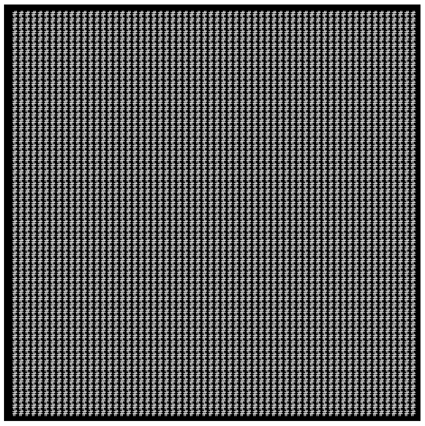
$$\{taxicab(x,y) | \forall_{x \in w} \forall_{y \in w}\,[x \neq y \wedge taxicab(x,y) \neq \infty]\}$$

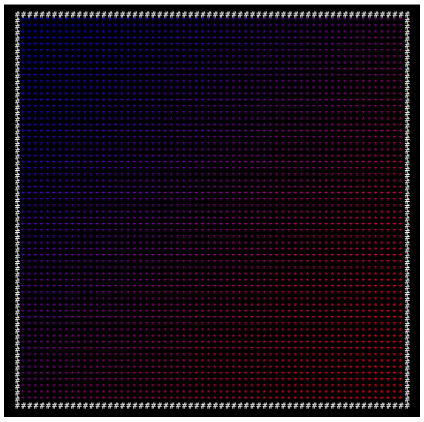
Definition 5:

A maze is *feasible* when the majority f the floor tiles in are in one cluster, which allows a player to traverse most of the maze itself. A maze is *completely feasible* when all floor tiles are in a single cluster.

Definition 6:

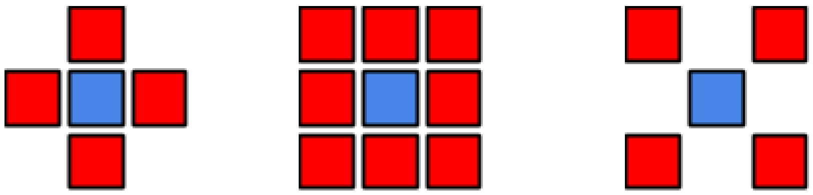
Maze *triviality* is at the current moment hard to define, resulting in a "*know it when I see it*" metric being used most of the time. The best way to describe triviality is the complexity of the maze's geography. Cul-de-sacs, dead ends, reconvergent paths are examples of complex geography. The most trivial of mazes are shown below:





Cellular Automata Neighborhoods and Regions

Cellular Automata *regions* are the stimuli a cell uses to determine its change of state. Regions are the collection of states of other member cells in a world space. *Neighborhoods* are regions composed of the members of the world space that are adjacent to such cell (i.e it's "neighbors"). In my experiments which I use cellular automata to generate maze-like levels, I use 3 different neighborhoods as stimuli. Their description is best shown in the diagrams below:



Maze Generation Methods

The following methods are used to generate maze-like levels:

Method M_1 : ANN Painting

This method traces through the world matrix w . A neural network is given two inputs, r, c and returns a real x such that $\{0 \le x \le 1\}$. We can abstract the neural network as a function, so consider the following algorithm:

$$ANN(r,c) < 0.5 \rightarrow '*'$$
$$ANN(r,c) \geq 0.5 \rightarrow '\#'$$
$$\forall_{w[r][c] \in w}(w[r][c] = ANN(r,c))$$

Method M_2 : Cellular Automata with static ANN defined rules

This method first fills the world matrix w with a random binary noise. A neural network is given a cellular automata neighborhood and returns a real x such that $\{0 \le x \le 1\}$. The algorithm is as follows:

$$ANN(h) < 0.5 \rightarrow '*'$$
$$ANN(h) \geq 0.5 \rightarrow '\#'$$
$$\forall_{w[r][c] \in w}(w[r][c] = ANN(h))$$

Method M_3 : Cellular Automata with dynamic ANN defined rules

This method first fills the world matrix w with a random binary noise. A neural network is given a cellular automata neighborhood and the current row and column value and returns a real x such that $\{0 \le x \le 1\}$. The algorithm is as follows:

$$ANN(r,c,h) < 0.5 \rightarrow '*'$$
$$ANN(r,c,h) \geq 0.5 \rightarrow '\#'$$

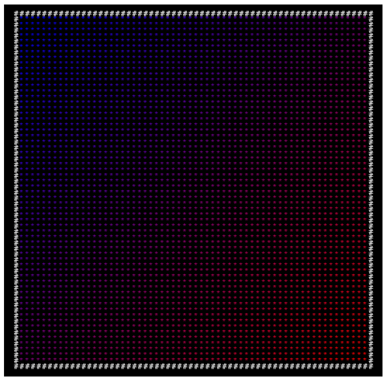
$$\forall_{w[r][c] \in w}(w[r][c] \ = \ ANN(r, c, h))$$

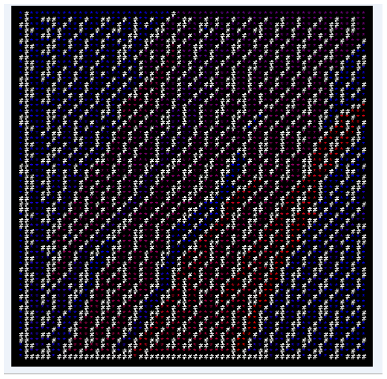
Experiments

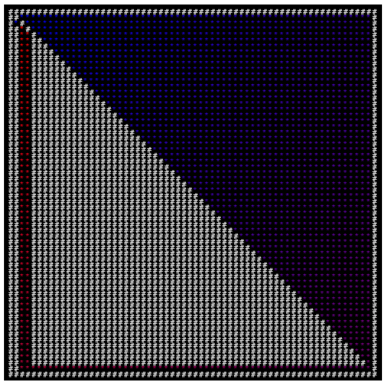
Using the Braincraft implementation of NEAT, I evolved neural structures in each of the methods in respect to various fitness functions. The evolution would run for 3000 generations and the best child would then be displayed. I used two fitness functions to guide the evolution. The first was *area* of a maze's largest cluster. The second was the *size* of the largest cluster in a maze. All three methods were evolved with both fitnesses and the two methods which involve cellular automata were evolved using the 3 different cellular automata neighborhoods as inputs.

Results

Out of the three methods, the second method perfomed the best with the current fitness functions. It resulted in the most complex mazes with reconvergent branching and cul-de-sacs being made. The first method performed the worst, mostly resulting spin-offs of the most trivial mazes previously shown. The third method gives interesting mazes but often filled the world space with blocks of walls. Only the second fitness function would result in interesting mazes. The first would result in trivial mazes almost always. The use of different cellular automata neighborhoods also had a significant impact on the generation of levels. The first two neighborhoods resulted in long mazes, either in a zig-zag or labyrinth design. The last neighborhood resulted in an expansive, feasible maze with wall tiles sparsely placed in the world. Here are the typical results of the three methods, methods 1 through 3 going from left to right:







Discussion and Conclusion

A frustrating aspect of my research is the ANN's inability to produce complex mazes, a task which it intuitively should be able to perform. I have hypothesizes that because of how NEAT operates, a fitness guided evolution of the neural structure prematurely converges the network to result in trivial mazes. One way to resolve this might be to begin the evolution with random fitness and then focus on a particular fitness. I hypothesize that this might circumvent the premature convergence to trivial mazes. I also believe that ANN's have a difficult time producing complex, feasible mazes. Evolving toward a completely feasible maze is difficult because the most common example is trivial.

A neural network guided cellular automata is an interesting concept but is incapable in producing modular structure such as rooms. It can create mazes but the intrinsic nature of cellular automata's randomness is a quality not desired in creating levels. Another problem with this approach is that there is no method to decide how many times the cellular automata should evolve. A better use for this method can be to evolve a cellular automata which can be used to make an already created level more "rugged" or cavernous.

The dynamic cellular automata approach shows promise, just not in the creating levels. The inspiration for this approach was to see if the rules which guide cellular automata generation could change in respect to the location of the cell in the world space. This could lead to interesting results in other research efforts where the use of cellular automata is more significant than generating mazes.

I must conclude that a more detailed and complex level can be created with a direct approach such as those described by Ashlock [2]. A functional approach to create levels could be changed into a composition of more direct methods and a neural network to map difficulty to the fitness of such method's evolution, but using a neural network to create such levels directly is not effective.

Future Work

I plan on continuing my efforts in mapping difficulty to procedural level generation. I will do this by completing my roguelike game and exploring different ways in creating my mazes. Another aspect which determines difficulty is the placement of loot and enemies in the game world and am exploring ways to do so. A possible solution to adding modular structures to my levels is the use of L-Systems in the construction of the levels. I am currently designing a more comprehensive L-System implementation to facilitate in this effort. My experimentation with cellular automata has also inspired me to explore its use in games and expanding its implementation to use regions, other than neighborhoods, as stimuli and allow for changes in cell rules in the middle of evolution.

Work Cited

[1]"The Overworld." - *Minecraft Wiki*. N.p., n.d. Web. 15 Dec. 2014.

[2] "Search-Based Procedural Generation of Maze-Like Levels." *IEEE Xplore*. N.p., n.d. Web. 15 Dec. 2014.