Will Bales

Dr. Wan

MATH 215

Cellular Automata in Generating Novel/Seeded Mazes

*Description of Accomplishments in The Project:*

There were several goals laid out at the beginning of the project. The goals were: writing code that would be able to draw a maze of any size with matplotlib, using the method of cellular automata to create said maze, having every point being connected to every other point along some arbitrary path, having every maze seeded in such a way that if the same seed and dimensions were entered, the algorithm would produce the exact same maze every time, and finally giving room for the possibility of creating an animation of the generation. All of these goals were accomplished through a rigorous process of trial and error. However, in addition to accomplishing all things that were laid out, there was the added bonus of finishing early enough to have time to optimize my code. I was able to shorten run times by eliminating the execution of cells in state 3 that were surrounded by already connected cells. This made it possible for the creation of an animation with a reasonable runtime of 2 hours… In addition to accomplishing the project goals, I was also able to gain a deeper understanding of cellular automata, matplotlib, numpy, classes, animation, seeding, code optimization, debugging, probability, and python overall.

*Interesting Features:*

There are several interesting features such as: seeding, strange spikes at the bottom of every maze—likely a seed issue as I checked my code, branching probability, and the capability of creating strangely shaped mazes—albeit matplotlib is not the best with this.

*Description of Challenges Faced and Resolved:*

Just looking at the goals that were laid out at the beginning it may seem as if they are not incredibly numerous; however, these goals are not as easy to accomplish as they may seem. One of the preliminary issues I was faced with was understanding how to use cellular automata to generate a maze. The concept was slightly foreign to me; so it was a learning curve. However, after I understood the general concept, I began focusing on constructing each level of my code. At the first level I began constructing my cell class. It was difficult to figure out what attributes the cells needed as there were so many different ways I could have approached creating the cell class. However, I soon realized that this aspect would only be finalized towards the end of the project. Once all of the pieces came together I became certain of how the cell class needed to be organized. The hardest aspect of creating the cell class was by far creating the method cell task that would govern the actions and interactions between cells. However, to be fair, this part of the code is the heart and soul of the algorithm. Moving further, there were small bugs with creating a class method that would draw an individual cell; however, they were easily fixed. Designing and debugging the maze class was similarly, if not more difficult than designing the cell class. Some difficult tasks that were resolved included: generating a string form of the maze that showed cell states, designing an initialization method, designing a create maze method that performed all cycles of maze creation, and working out a way to animate a large maze. The latter two of the list proved to be substantially more difficult than all other challenges. The reason why designing the create maze method was difficult was because it required data from every level of my code; transferring this data was tedious and required a lot of thought. In addition to this, the method was also the culmination of every other process in the project; thus, it was the ultimate test for bugs. Finally, the reason why animation was so difficult was largely because of the way that my plotting was originally designed. The design was not compatible with any of the matplotlib libraries relating to animation. This forced me to have to use imperfect methods to produce a gif, such as using a website online.

*Other Relevant Details:*

Much of this code's inspiration came from the pdf listed in sources. Additionally, there were aspects of this code that could have been changed to make it more visually appealing, such as in the case of matplotlib; however, this would have required extensive redesigns. Also, I believe the final product still gets many of the messages across.

*Sources:*

<https://justinparrtech.com/JustinParr-Tech/wp-content/uploads/Creating%20Mazes%20Using%20Cellular%20Automata_v2.pdf>

<https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html>

<https://ezgif.com/maker>

<https://www.youtube.com/shorts/Ann21Y0kmVg>