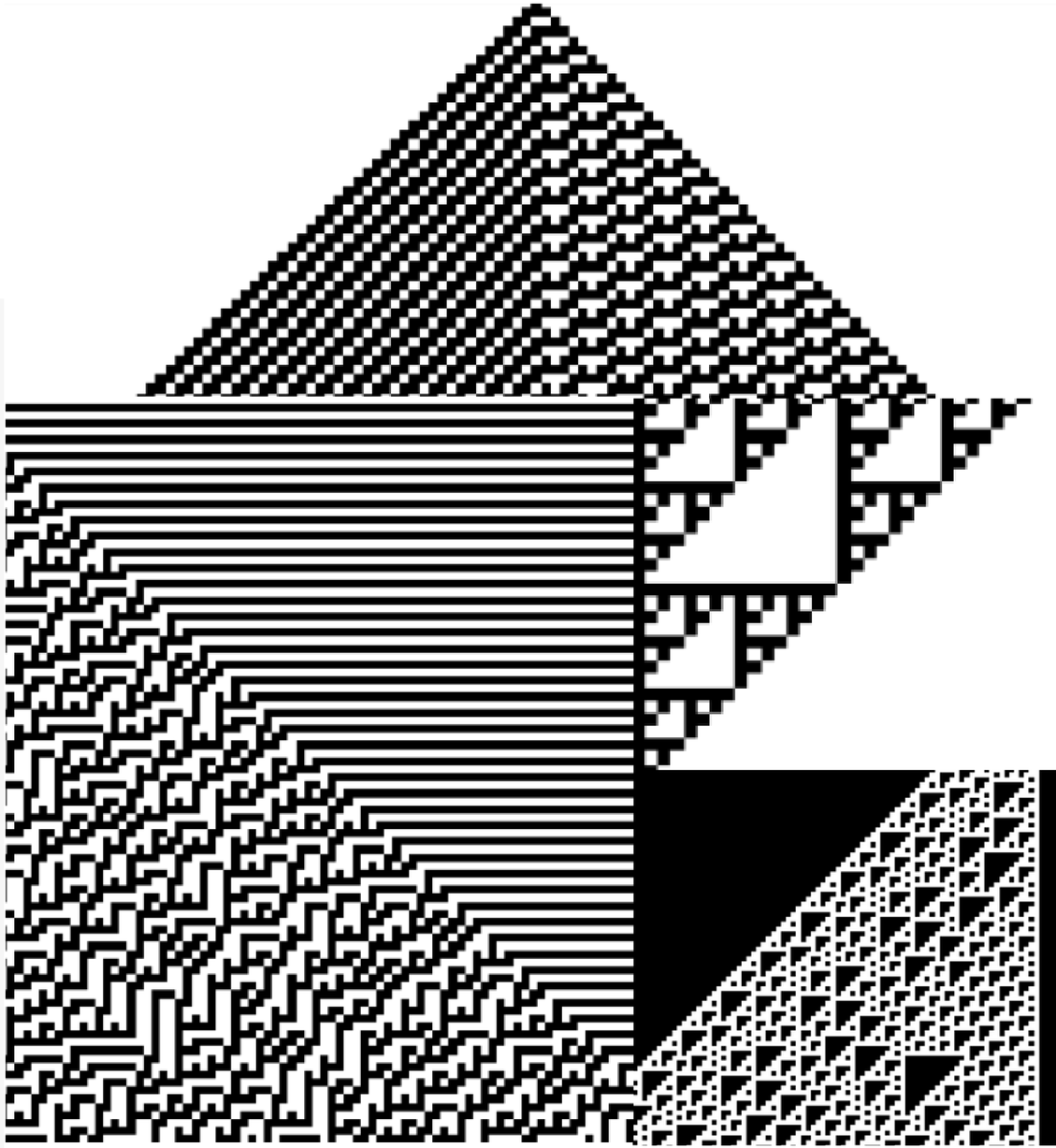
ACTIVITY 10. COMPLEXITY SCIENCE: CELLULAR AUTOMATA

Julian Christopher L. Maypa
2020-07587
App Physics 157 WFY-FX-1

Objectives

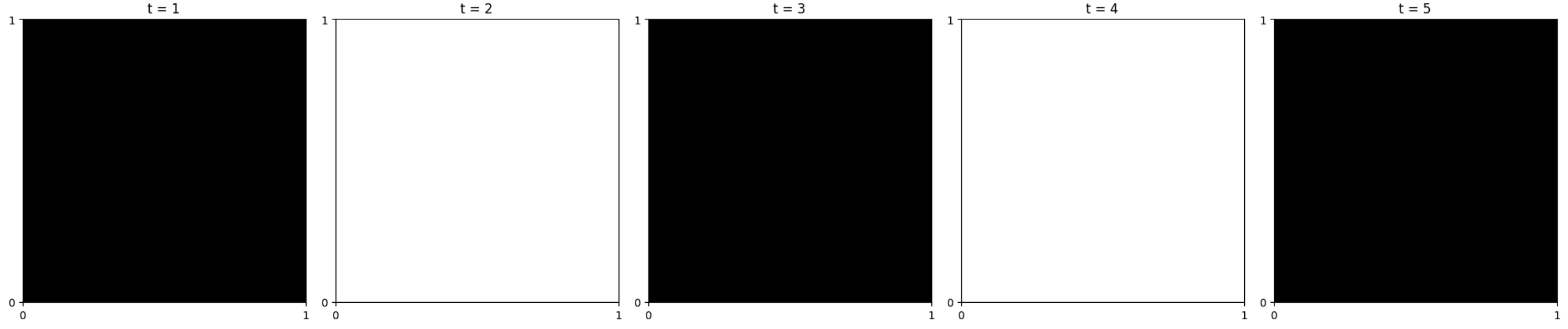
- [x] Define a cellular automaton
- [x] Demonstrate Wolfram’s 1D CA models.
- [x] Implement and Apply CAs.



Cellular Automaton

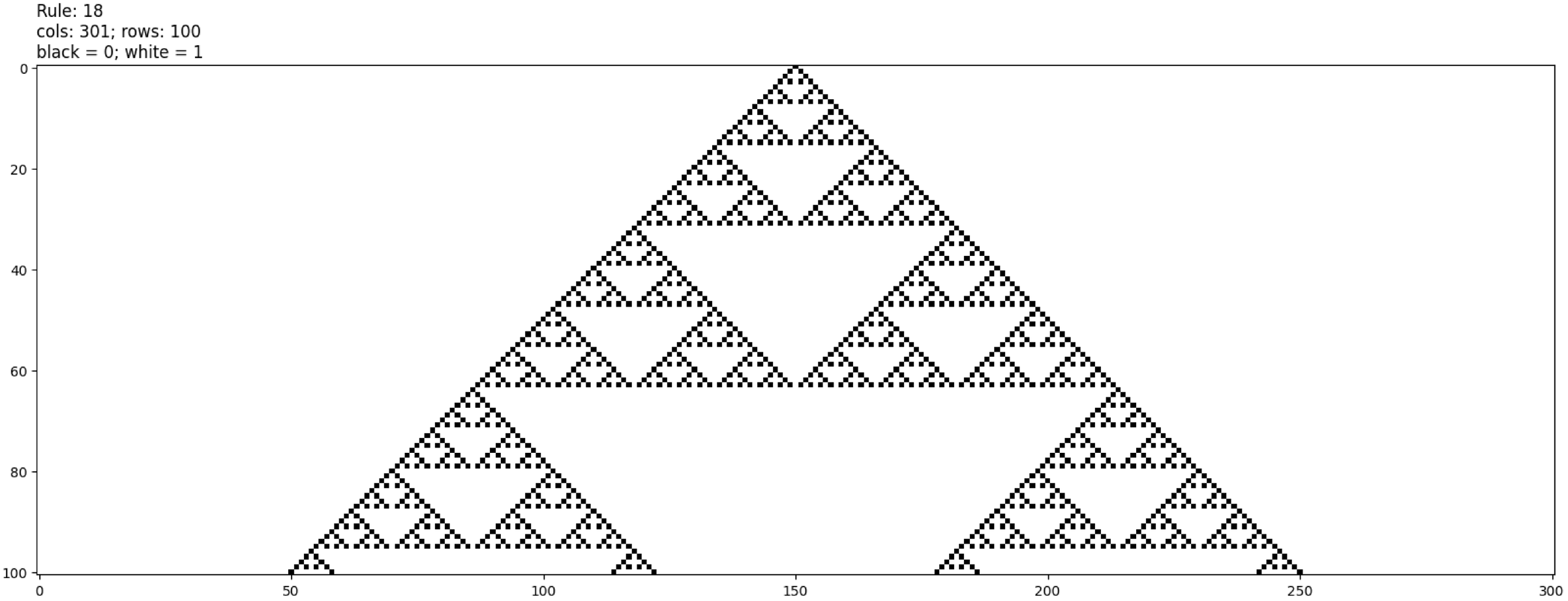
The most basic demonstration of a cellular automaton is a blinking pixel. Below is a matrix containing a single element. Black represents 0 and white represents 1. The pixel is governed by a single rule which is $x = (t + 1)\%2$. Where x is the state and t is the time step. So at $t = 1$, the pixel is set to 0 or in other words, turned off. At the next time step, the pixel is turned on. And this patterns goes on for an infinite number of time steps. This analogy can be extend to multiple dimensions, where during each time step, the previous state is morphed based on a rule.

black = 0, white =1



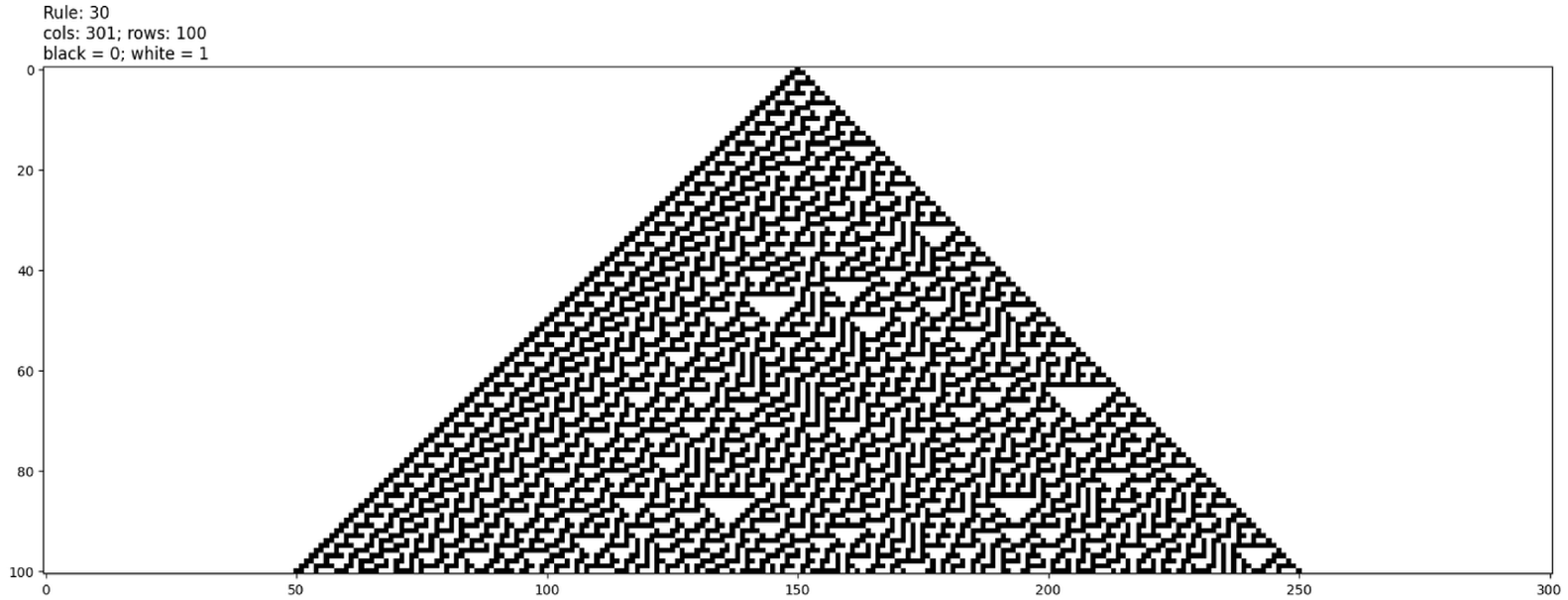
Wolfram’s 1D CA models

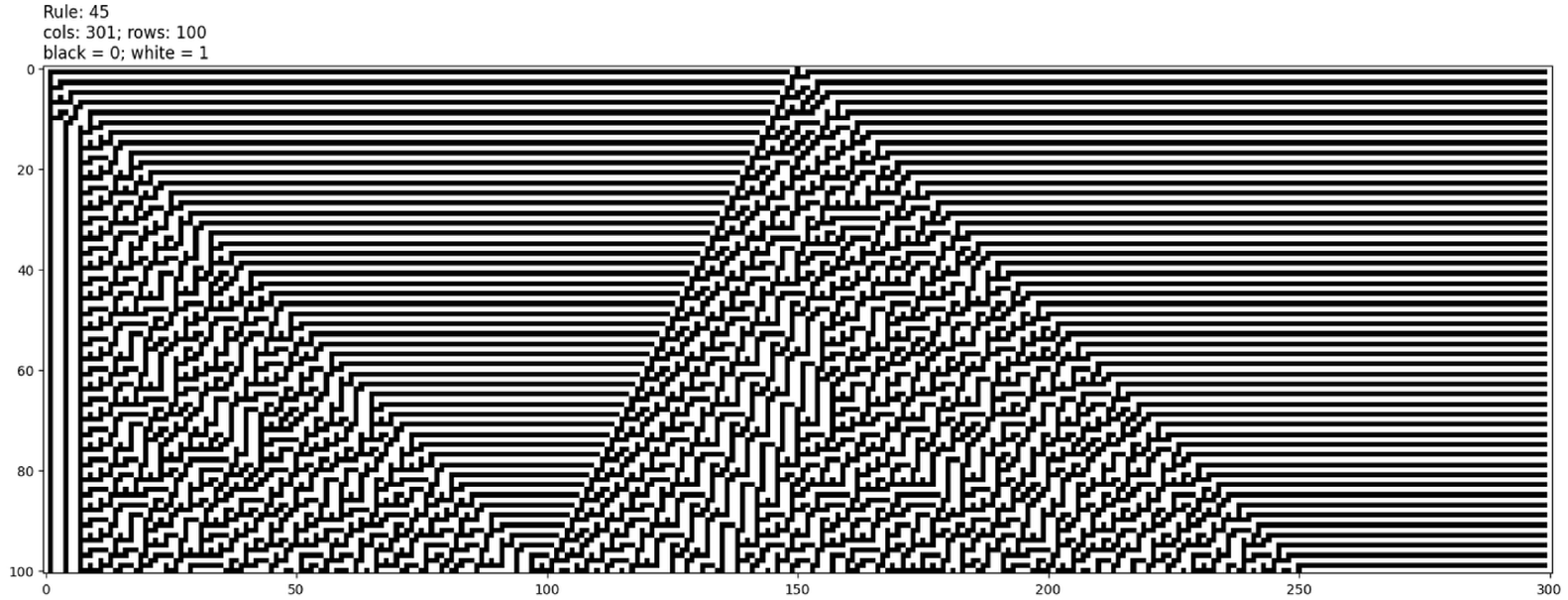
I then implemented Wolfram's 1D cellular automata models. In the succeeding slides, the models show determinism, randomness, and universality.



Determinism

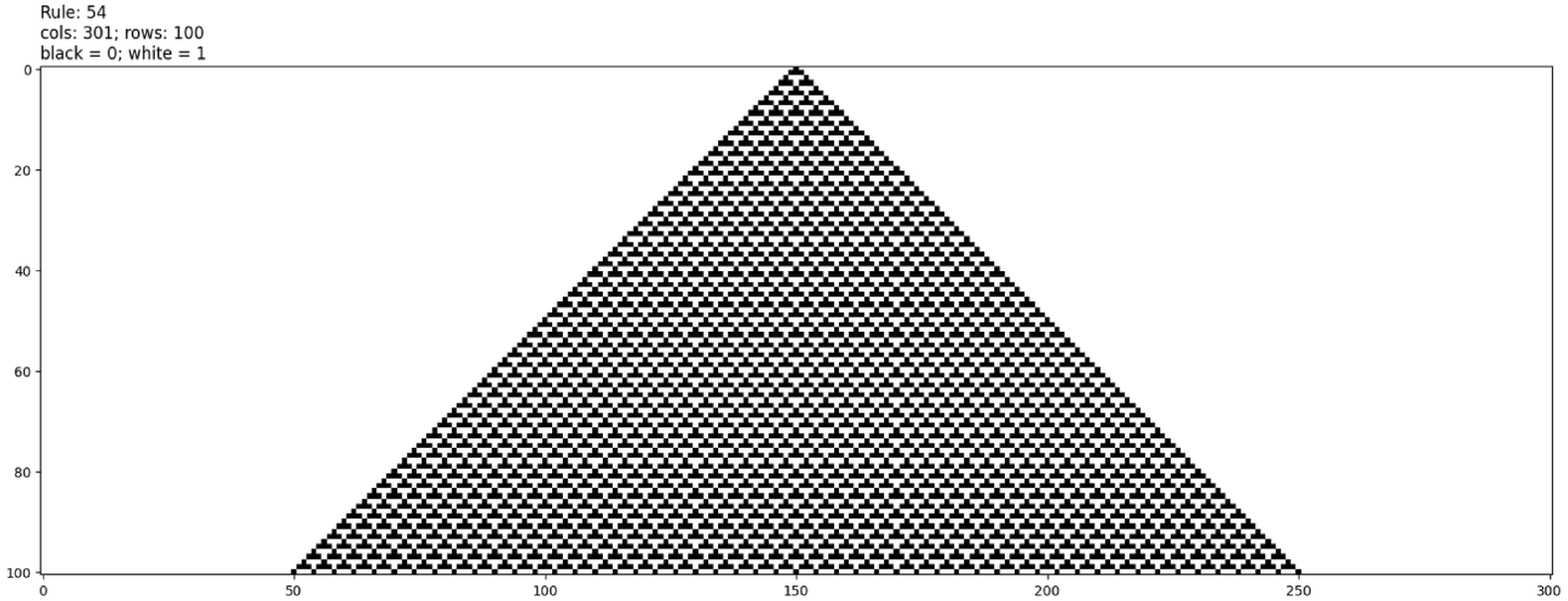
Cellular automata models are deterministic because the next generations are always determined by the previous generations. Each generation is always related to its previous generation, and the next generation is not randomly generated.

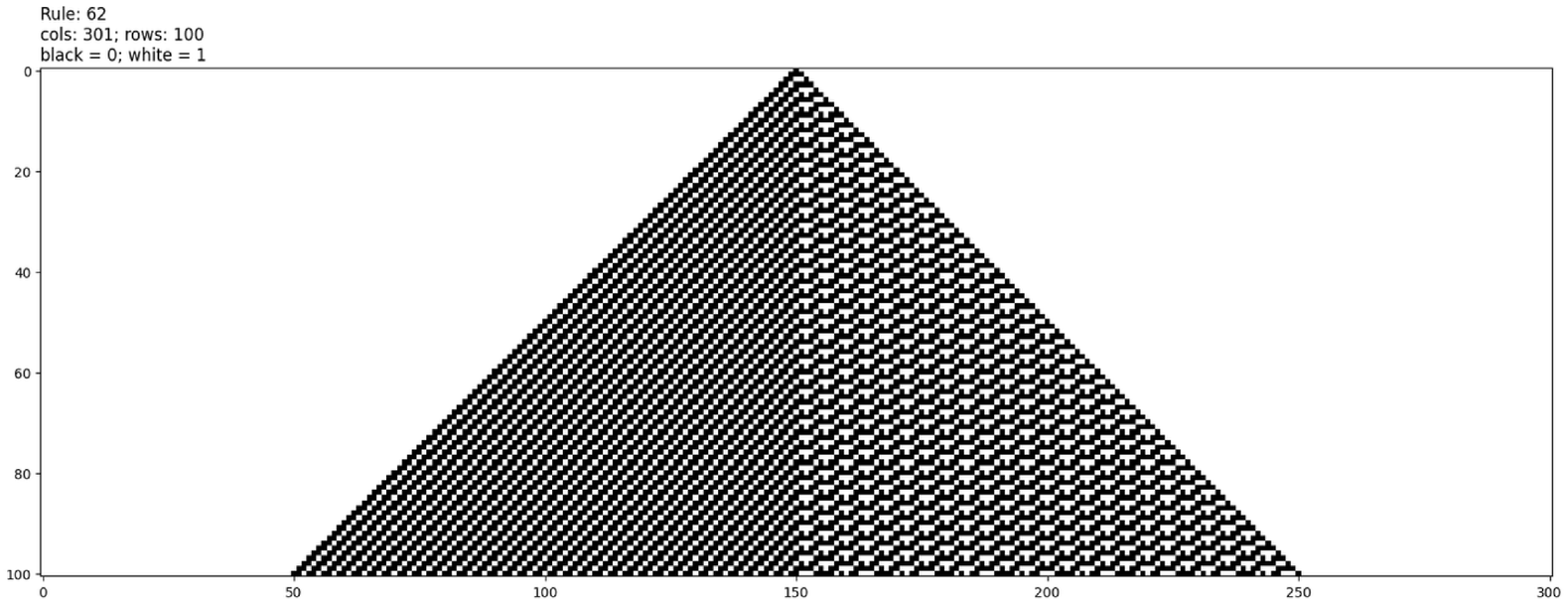




Randomness

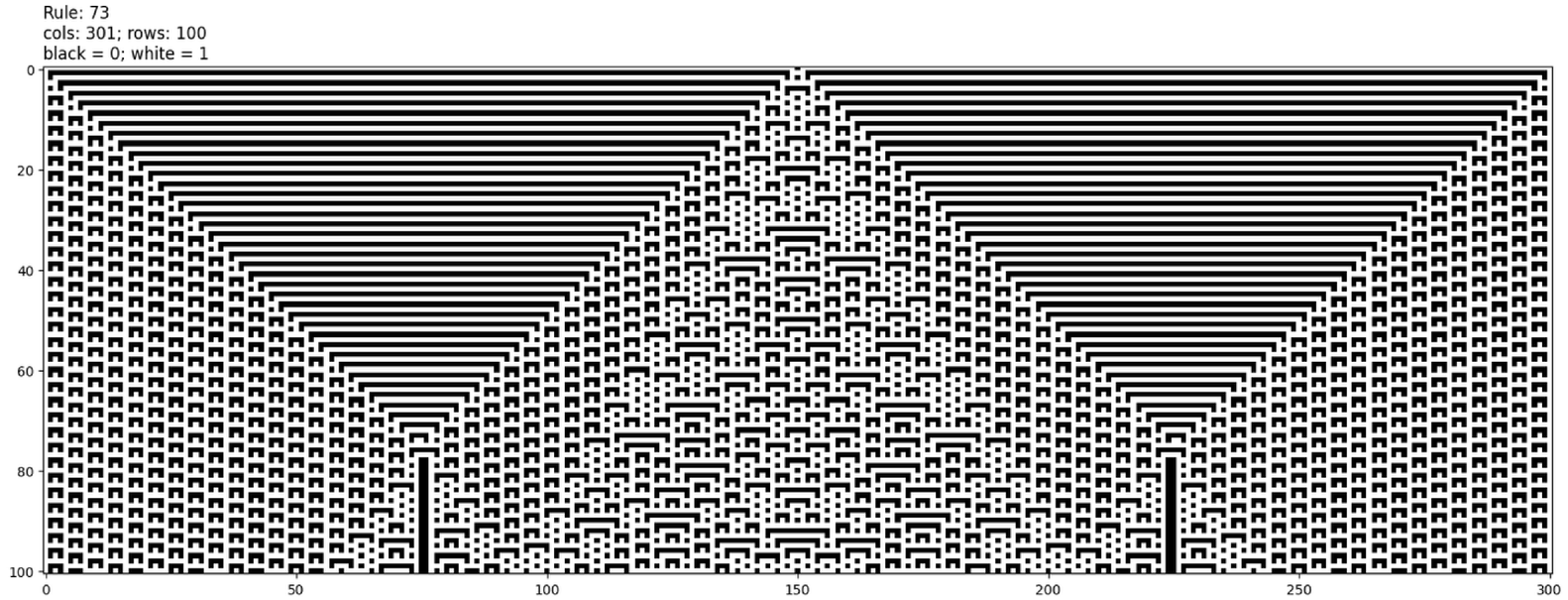
A cellular automata can demonstrate randomness. Some rules produce a symmetric and repeating pattern. But some rules have a non-repeating pattern, such as Rule 110. As more generations are added, more random patterns are created.

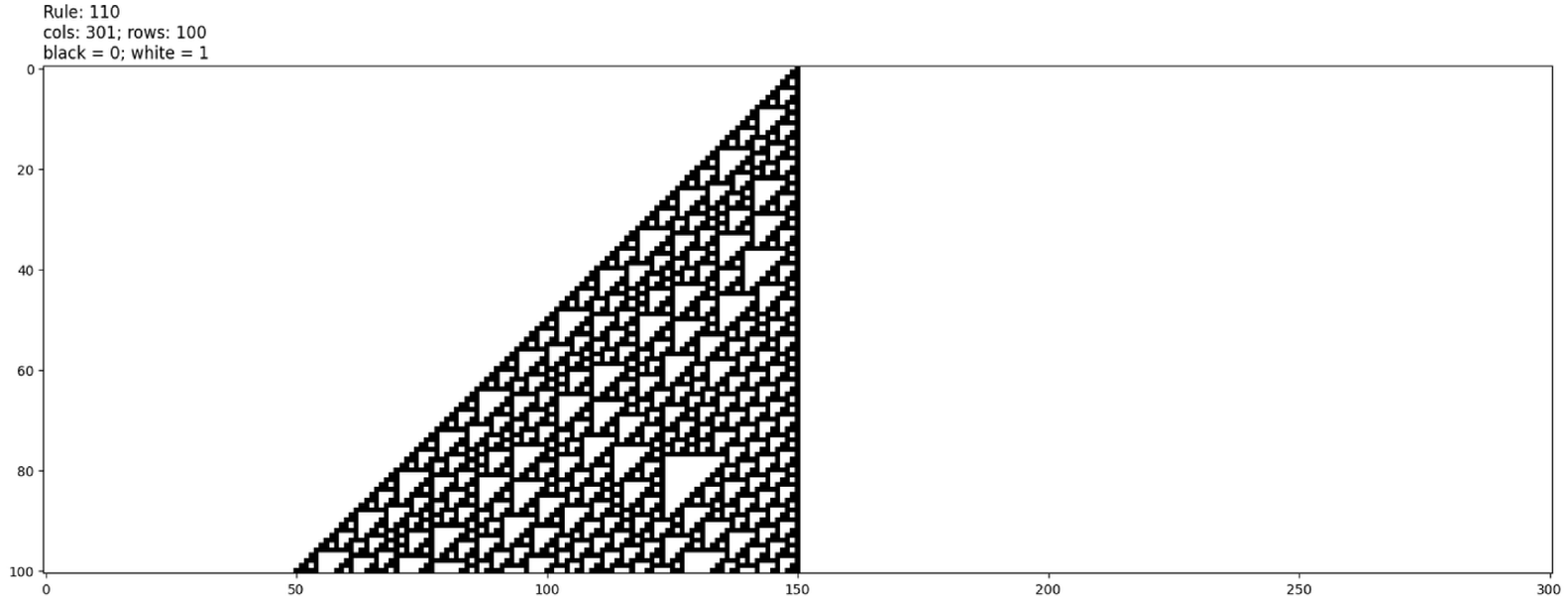




Universality

The cellular automata, especially Rule 110 (which can be seen in the right), can also serve as a model for all the solutions in a Turing machine. This is also why Rule 110 of the cellular automata is called Turing complete. Since all Turing functions can solve any computational problem, then any Turing complete cellular automata can be used to model any computational problem.

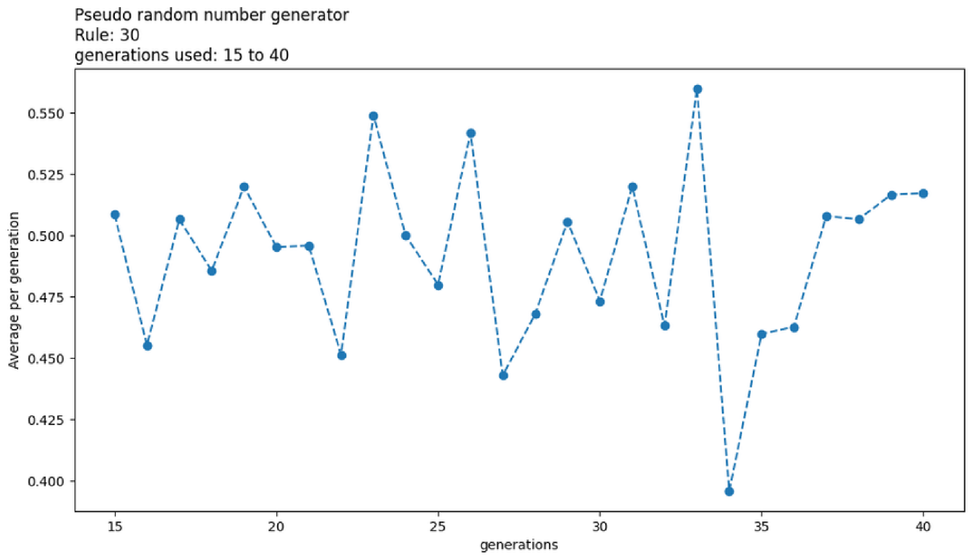


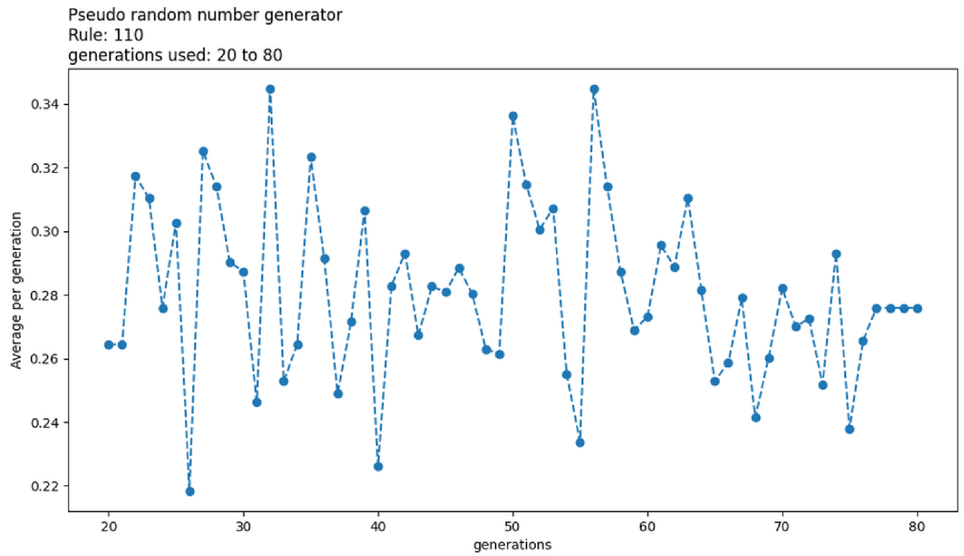


An application of a CA

I tried to make a pseud-random number generator using a cellular automata. What the generator does is that it creates an array with *n* elements ranging from 0 to 1. The number of elements *n* is equal to the number of columns of the CA. I then multiply the first row of the CA to the array and get its average. I repeat this process for a certain number of rows and the output numbers serve as the pseudo-random numbers. The graph of the pseudo-random numbers can be seen below:

Looking at the graphs, it looks somewhat erratic and has no visible pattern. Thus I have successfully shown that CAs can be applied in pseudo-random number generators.





Reflection

Overall, I believe that my output for the different rules of the cellular automata are correct because I cross referenced them with the patterns found online. The simulation was easy to make and it was relatively easy to understand. I also did not really get stuck in any part of this activity.

I'd like to thank my instructors, Sir Rene Principe Jr. and Sir Kenneth Leo, for guiding me throughout the activity. I would also like to thank my professor, Ma'am Jing, for guiding me in my coding while my classmates and I worked in R202. I would also like to acknowledge my classmates: Abdel, Johnenn, Jonabel, Richmond, Lovely, Hans, Genesis, Jeruine, Rusher, and Ron for helping me complete this activity.

Self Grade

Technical Correctness	I understood the lesson and met all the objectives. My results are complete and I got the expected results.	35
Quality of Presentation	The images I added to this report are of good quality and all the graphs are properly labelled. My code is also properly organized and labelled.	35
Self Reflection	I got the expected results, and acknowledged the contributions of my peers while doing this activity. I also properly cited online references.	30
Initiative	Apart from doing the required tasks, I also helped my classmates with their code and helped them by cross-referencing my results with theirs.	10
	Total	110

References

[1] Downey, A.B. (2016). *Think Complexity*. Green Tea Press