CS142
Visualising Snowflake Growth with Cellular Automata

Lydia Shepherd (u1900130)

May 14, 2020

Contents

List of Figures

List of Tables

1 Introduction

For this project, I chose to explore further uses of cellular automata (CA) to visualise systems, in this case the growth of snowflakes. Snowflakes grow as ice crystals which take a hexagonal shape and hence a perfect ice crystal or plate snowflake is a regular hexagon with no imperfections [3]. However, the conditions under which a snowflake grows, such as the air temperature and humidity, define the trillions of different shapes it can take. Using a CA to visualise snowflake growth has been done before and so I drew inspiration from these works when creating mine. An example of these is the visualization pictured in figure 1 which is part of a website explaining how snowflakes grow and allows the user to explore the simple rules behind it [1].

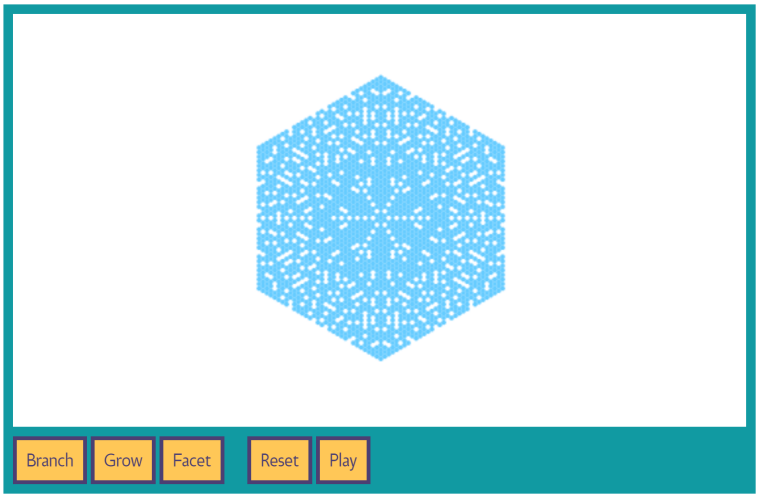

Figure 1: Image of the simulation of snowflake growth allowing user exploration[1]

I drew a lot of inspiration from this site and visualisation including inspiration for the underlying rules for snowflake growth and allowing the user to explore this phenomenon. My main aim for this project was to clearly simulate snowflake growth and allow exploration of the underlying rules. However, another aim for my project was to also inform the user of common snowflake classifications and show them how they are created. I also used videos of real snowflake growth to gain insight into this process, an image of which is shown in figure 2.

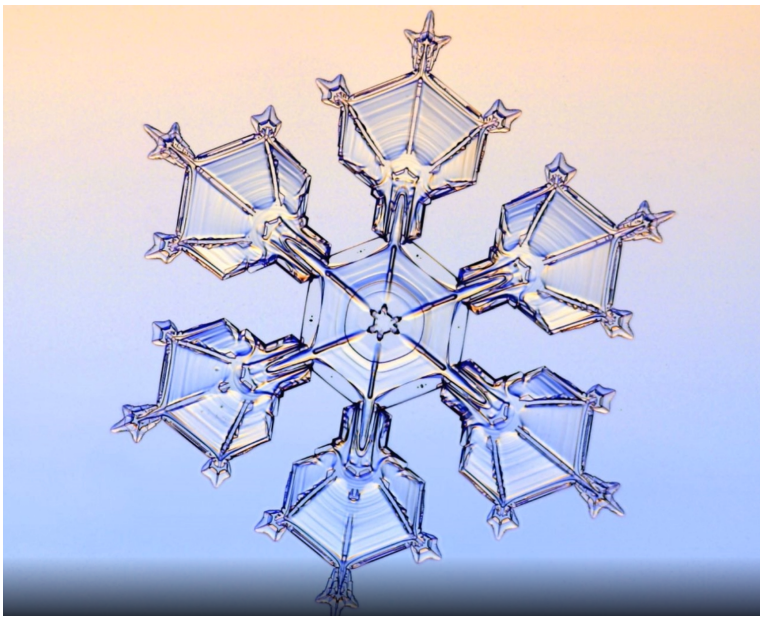

Figure 2: Image of a real snowflake taken with a microscope [2]

2 Design

When designing the grid for this CA, it was clear that it needed to be hexagonal since the basic shape of ice crystals are hexagons and a snowflake is made of many ice crystals. To create the rules for the CA, I took inspiration from two sources, the visualisation shown in figure 1 and Wolfram's paper on two-dimensional cellular automata which details pattern growth ideas [4]. In real snowflake growth, there are 3 main rules, growth, faceting and branching [5]. These were the 3 rules for my CA and I used trial and error to work out the best probability for each rule to be called. Growth involves any cell with at least one neighbour that is ice becoming ice and has the lowest probability. Faceting means that any cell with at least 3 neighbours being ice becoming ice and is responsible for filling in gaps in the snowflake. Finally, branching is when any cell with one line of 2 ice cells preceding it becomes ice and hence creates the classic 6 branches of a dendrite snowflake.
The growth of the snowflake on the screen was the main focus of this project and so to draw the users' attention I used the preattentive attribute of movement as the pattern of the snowflake grows and moves on the screen [6]. I also made the grid large and central on the screen so it would be the main focus. I kept the background colours muted by using grey lines for the grid

so that the colour for ice cells would stand out more [7]. I chose a dark blue for the ice cells since blue is a common colour for representing ice and the dark shade caused high contrast so it draws the users' attention and will be easily visible for everyone regardless of any visual disabilities such as colour blindness [8].
I used buttons and scroll bars to allow the users to explore the CA and different rules. I kept the interaction simple with all elements evenly spaced at the top of the screen and made sure to make the buttons clear to the user with natural mapping [9] and I made them dark with white text in order to stand out [8]. I preferred the larger and evenly spaced buttons in my design to the version shown in figure 1 as I felt my design was clearer. The scroll bar allowed the user to select a preset snowflake pattern from 3 options: plate, needle or stellar dendrite. These are 3 common snowflake classifications which the user could watch grow on the screen. A drop-down list made sense as it is intuitive for selection processes. Additionally, the user could explore random snowflake growth by drawing their own starting configuration on the screen and using the buttons to grow it either automatically or manually. Additionally, I decided that the user should be able to save an image of the displayed snowflake since they are randomly generated and so some designs are not interesting. Therefore, the user could save their favourite snowflake designs.

3 Implementation

To create the grid, I made a Hex class which allowed each cell to be an individual hexagon with an attribute of whether it was ice or not. This class also handled the drawing of the hex to screen.

Hex
x: float y: float radius: float row: integer col: integer ANGLE: float ice: boolean previous: boolean c: color neighbours: Hex[]
Constructor(float x_in, float y_in, float radius_in, int row_in, int col_in) display() setNeighbours(int index, Hex h) getNeighbours(): Hex[] getX(): float getY(): float getRadius(): float getRow(): integer getCol(): integer wasIce(): boolean setIce(boolean b) update()

Figure 3: Class Diagram for the Hex class

Creating a hexagonal grid is more complicated since the rows and columns are offset from each other. The y offset is always the same but the x offset changes depending on whether the row is even or odd [10]. Using a simple nested loop to create the coordinates for each row and column, the 2d array of hex objects was populated as the grid [11].

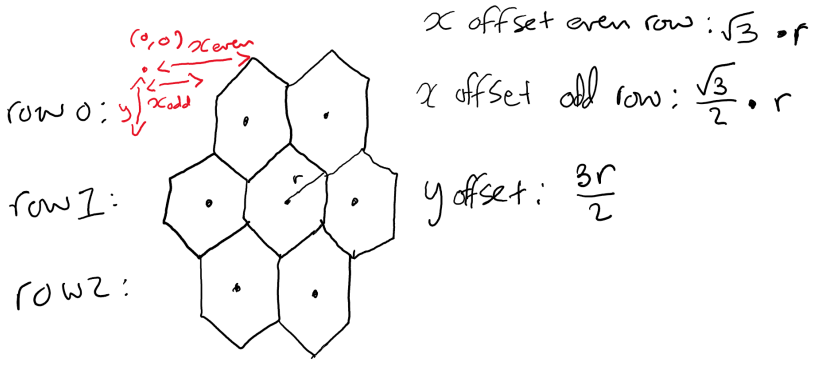

Figure 4: Diagram showing the maths behind creating a hexagonal grid and showing the 6 neighbouring cells to the centre cell.

To find the neighbouring cells in a hexagonal grid depended fully on whether

the cell was in an odd or even row. I wrote a function that would populate the neighbours array of each hex by using a series of if-statements. The neighbours array contains copies of the hex objects so that their previous state can be examined. It had to be the previous state since the current state would be changing as the rules were run and so the previous state of the grid had to be used for determining the next state. When designing the rules themselves, I made some adaptations to the original designs in order to improve the quality of the snowflakes created. The growth rule was simply any cell neighbouring ice becomes ice and stayed that way but I adapted the faceting rule so that it only facets when the hex has 3 or 4 neighbouring ice cells. This meant that gaps could be left in the snowflake to model imperfections in the crystal or rime (water) droplets. I used random number generation in order to select which rule was used when the simulation was run with branching having an 78% probability, faceting 15% and growing 7%. In my opinion, this created more accurate snowflake patterns.

Upon startup, the grid has a simple hexagon pattern of ice cells on it which allows an easy starting point for users to grow snowflakes. The 'run' button at the top runs the above rules 25 times to generate a snowflake pattern. I chose a frame rate of 5 for this so that the user can see each step happening but it still flows smoothly. If the user would prefer, they could use the 'step' button to call a random one of the rules and watch a snowflake grow through manual steps. This allows for a deeper exploration of snowflake growth. I also added individual buttons for each rule so that the user could try out any combination of the rules to grow their own snowflake. The main structure of the code is dictated by 5 states. In the main state 0, the sketch waits for input either a button press or selection from the drop-down list. State 1 is only reached when the 'run' button is selected as it uses the draw loops to call the rule running function 25 times. For the other buttons, a simple function call is required to execute the required rule. The 'reset' button removes the displayed snowflake and returns to the starting configurations, as well as clearing any selection in the drop-down list.

The drop-down list allows the user to select from 3 different preset snowflakes and if one is selected the corresponding program state is reached which calls the correct function. A problem I encountered when listening for both button selections and drop-down list selections was that the list always seemed to have an option selected automatically even when it was a button that was clicked. To avoid this, I put the drop-down list options at the end of the if-statement so it would only trigger if none of the buttons triggered it. All of the buttons and the drop-down list were made using the ControlP5 library

[12]. The original font used by the ControlP5 library was very hard to read and was too small for the buttons so I created my own font to use which was larger and easier to read.

I additionally listened for `mouseClicked` events to check if the user had clicked in any of the hexes on the grid. If they clicked on the grid, it then toggled that hex to ice or not so that the user could try their own starting configurations. The `keyPressed` event allowed the user to press 's' to save an image of the currently displayed snowflake.

4 Resulting Visualisation

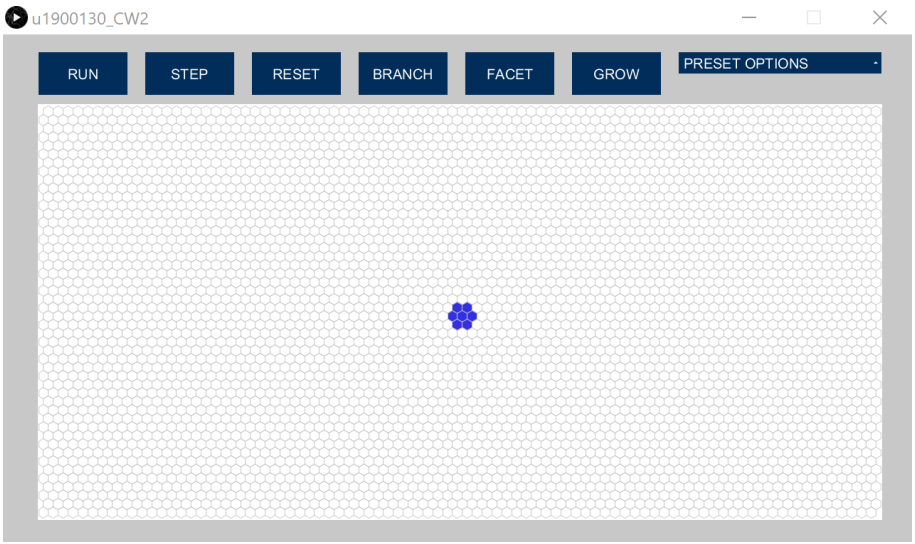

Figure 5: Screenshot showing the starting configuration of the visualisation.

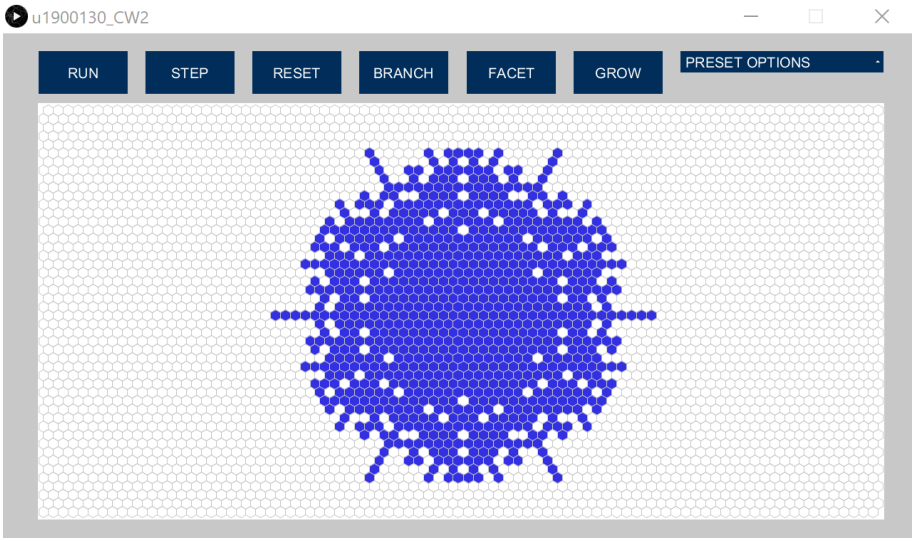

Figure 6: Screenshot showing the snowflake resulting from pressing the 'run' button.

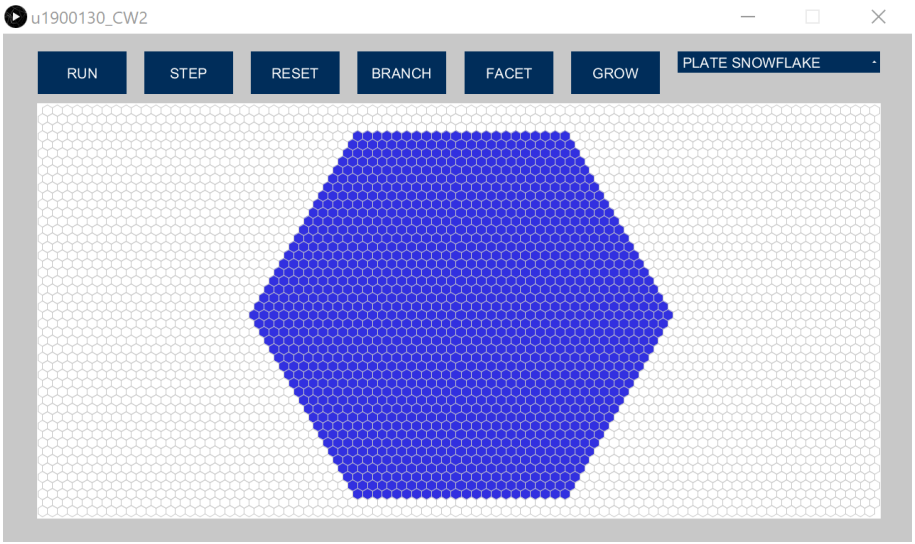

Figure 7: Screenshot showing the 'plate' snowflake configuration.

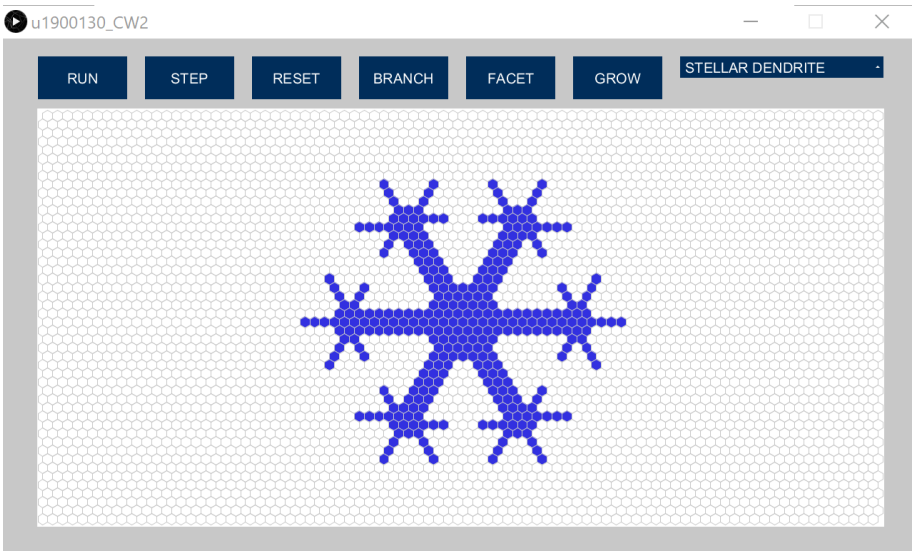

Figure 8: Screenshot showing the 'stellar dendrite' snowflake configuration.

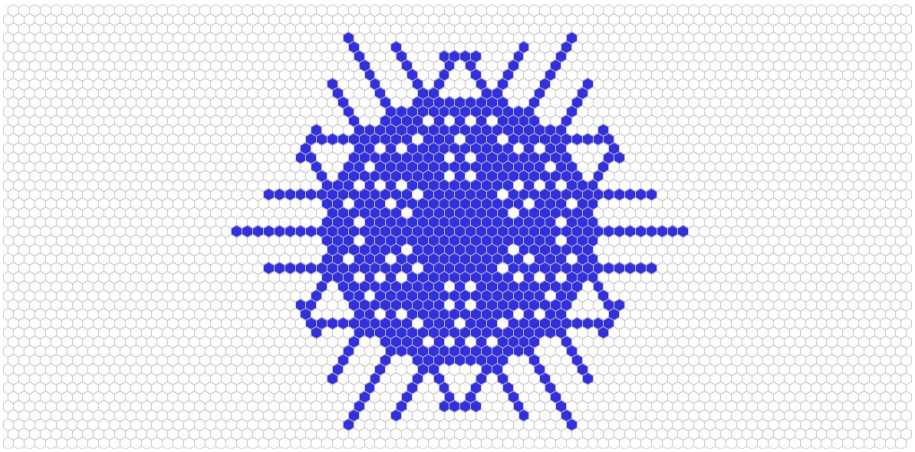

Figure 9: An example of the image saved when the user presses 's'.

5 Evaluation

I carried out an evaluation of this project following on from the aims discussed in section 1 and using the 5 evaluation criteria from Introduction to Information Visualisation [8]. I also got feedback from four users based on their experience testing it.

Criteria	Requirements	How were they met?
Functionality	• Simulate snowflake growth. • Allow users to explore snowflake growth. • Inform on basic snowflake classifications.	The CA clearly simulates snowflake growth using 3 simple rules and the low framerate allows the steps to be seen. Through these rules, around a trillion unique snowflakes can be created. The buttons allow users to play with the rules and create their own snowflakes and starting configurations. The options list allows the user to explore 3 common snowflake classifications but a nice extension to this would be adding an information box about these classifications.
Effectiveness	Provide better knowledge of snowflake growth	The use of buttons to allow each rule to be run separately is an effective way to see the mechanics behind this system. It is clear to see the rules required to create different styles of snowflakes. After testing, the users said they found the exploration to be informative since they had no prior knowledge of snowflake growth.
Efficiency	Provide a focussed and clear simulation	The colour design choices helped make this simulation very clear since the dark blue snowflake stood out. Using a CA model gave it a very simple design and it made the growth easy to watch. Users commented that watching the formation of the snowflake was interesting and there were no distractions since it was centred. One user mentioned it would be useful to be told what rule was being run in each step when the 'run' button was pressed. I tried to add this but it made the screen messy and was a distraction from the snowflake.

Usability	User interaction is simple and intuitive	After testing, all users agreed that the UI design was clear and easy to use. They also commented that the exploration was very enjoyable and found watching the snowflakes grow moreish. The simple row of buttons at the top made the design consistent and intuitive. One user commented that the drop-down list was not intuitive to scroll to find the third option and so I added a note into the readme file. Some users didn't understand the use of toggling the grid cells to change the starting configuration. Allowing the users to save a snowflake was a nice addition but it would be better if it could save multiple snowflakes to different files.
Usefulness	Serves a purpose	This CA clearly simulates semi-realistic snowflake growth. With a few additions, it could make a good educational tool since users found it to be informative.

Table 1: Table evaluating the success of this visualisation

6 Conclusion

Overall, I achieved my overall aim for this project and used a CA to simulate snowflake growth. It provides a simple UI to allow for intuitive interaction so that users can easily explore the rules and styles of snowflake growth. The CA itself uses the three rules with different probabilities so that it can create realistic snowflakes as well as creating some aesthetically pleasing snowflakes. Since there are around a trillion (3^{25}) different snowflakes that can be created through the 'run' button, not all look pretty but are still accurate snowflakes. I also partially achieved my aim to inform on different snowflake classifications through the preset options drop-down list.
I thought of a few extensions and improvements to this project including adding an additional information box to the screen for the different clas-

sifications. This would inform the user of the temperature and humidity required for that classification of snowflake. Since printing the rule being used for each step of the snowflake drew the user's attention away from the grid, I thought it could be added to a separate file so that if the user was interested they could examine it. Another small extension would be to improve the functionality of the save function so that it can save to multiple files rather than just one. Additionally, it could be extended to allow the user to change the probability of each rule being run so they could see how it affects the created snowflakes. I chose not to include some of these ideas since I felt it would overwhelm the user and make the design less intuitive.

References

[1] Andrew Taylor. Snowflakes. https://www.andrewt.net/maths/snowflakes/, December 2018. Last Accessed: 29-04-2020.

[2] Kenneth G. Libbrecht. Growing snowflakes. http://www.snowcrystals.com/videos/videos.html. Last Accessed: 29-04-2020.

[3] Kenneth G. Libbrecht. Guide to snowflakes. http://www.snowcrystals.com/guide/guide.html. Last Accessed: 29-04-2020.

[4] Stephen Wolfram and Norman H. Packard. Two dimensional cellular automata. *Journal of Statistical Physics*, 38(5/6):901–946, 1985.

[5] Kenneth G. Libbrecht. Snowflake science. http://www.snowcrystals.com/science/science.html. Last Accessed: 29-04-2020.

[6] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2012.

[7] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press USA, 2001.

[8] Riccardo Mazza. *Introduction to Information Visualization*. Springer Publishing Company, Incorporated, 2009.

[9] A.Cairo. *The Functional Art:An introduction to information graphics and visualization*. New Riders, 2012.

[10] Amit Patel. Hexagonal grids. https://www.redblobgames.com/grids/hexagons/. Last Accessed: 29-04-2020.

[11] Louis Christodoulou. Tinkering in processing – drawing a hexagonal grid. http://louisc.co.uk/?p=2554. Last Accessed: 29-04-2020.

[12] Andreas Schlegel. controlP5. http://www.sojamo.de/libraries/controlP5/. Last Accessed: 24-04-2020.