COMP39X

2019/20

3-Dimensional Game of Life

Student Name:	Jack Ashton
Student ID:	201182922
Supervisor Name:	Dr Stuart Thomason

DEPARTMENT OF
COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Dedicated to the memory of John

Acknowledgements

I would like to thank my parents, Paul and Carole, for their support throughout my studies. Thanks also to my supervisor, Stuart, for putting up with my constant questions and moments of worry while I was writing this dissertation.

I dedicate this to the memory of John Horton Conway, who sadly passed away while I was writing this dissertation, and without whom this entire project and field would not have existed.

COMP39X 2019/20

3-Dimension Game of Life

DEPARTMENT OF COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Abstract

John Horton Conway's experiments in the field of cellular automata concluded with the discovery of a set of rules for 2D cellular automata, now known as "Conway's Game of Life". This project aims to recreate Conway's Game of Life in a 3-Dimensional space and to explore how and what life forms emerge. Several unique life forms were discovered with each pattern classified by their shared characteristics. A rule was discovered that is the best candidate for a 3D equivalent of Conway's Game of Life.

Contents

Introduction

Conway’s Game of Life is a cellular automata simulation consisting of a 2D grid of cells. Each cell is represented by one of 2 possible states, Alive or Dead. Whether a cell is Alive or Dead is determined by applying a specific set of rules to each cell, based on the number of neighbours that cell has. By this method, the cells of “Conway’s Game of life” tend to evolve from its initial conditions, creating unique patterns and movement.

There are many variations on the original simulation, however, there are few that investigate how it would work in 3D. We will be implementing an optimized version of “Conway’s Game of Life” and adapting it to function in a 3-Dimensional space. In addition, we will be using the program to explore the effects of the rules on how Life forms and what, if any, unique patterns emerge.

Background

In the late 1950s, Stanislaw Ulam and John von Neumann devised a method for calculating liquid motion by dividing the liquid into discrete cells and compute the movements of each cell based on the behaviour of its neighbours. This laid the groundwork for a branch of mathematics known as cellular automata. *(Bialynicki-Birula (2004))*

In 1968, John Horton Conway conducted experiments with 2D cellular automata with different sets of rules and conditions, motivated by the works of Ulam and others years earlier. By 1970, Conway had found a simple set of rules that he called “Game of Life”, later popularized by a publication in Scientific American by Martin Gardner. *(Wolfram (2002)*

Conway’s Game of Life is a two-dimensional simulation of square cells, each of the cells is in one of two possible states, alive or dead. Each cell checks each of its eight neighbours and at each time step the following rules determine the new state of each cell:

1. Any live cell with fewer than two live neighbours dies. (Underpopulation).
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies (Overpopulation).
4. Any dead cell with exactly three live neighbours becomes a live cell (Reproduction).

Each simulation starts with an initial pattern, which the above rules are applied to create the first generation. The rules are applied iteratively to create subsequent generations.

Below are a few examples of patterns that commonly arise from these rules.
One of the most common static patterns is the "Block". All the dead neighbours do not have enough alive neighbours to become alive and the alive neighbours always have exactly 3 neighbours, therefore the pattern does not change. It also has the interesting property that if any one cell is removed from the pattern, the other 3 alive cells will cause the removed cell to become alive again.

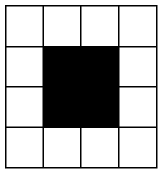


Figure 1.1

Another common pattern is the oscillator, the most common one being the "Blinker". This pattern switches between the two below states. 2 dead cells have 3 alive neighbours and therefore will become alive in the next iteration. Also 2 alive cells only be 1 alive neighbour and therefore will die in the next iteration. This causes the pattern to switch as shown below.

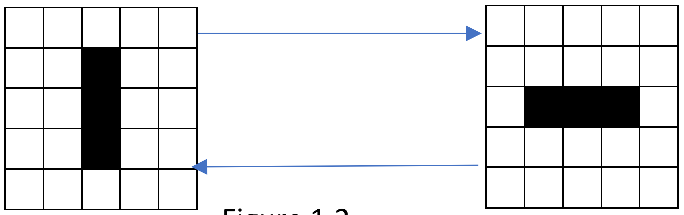


Figure 1.2

There are versions of oscillators that can traverse across the board, these patterns are called spaceships. The most common spaceship is the "Glider" shown below.

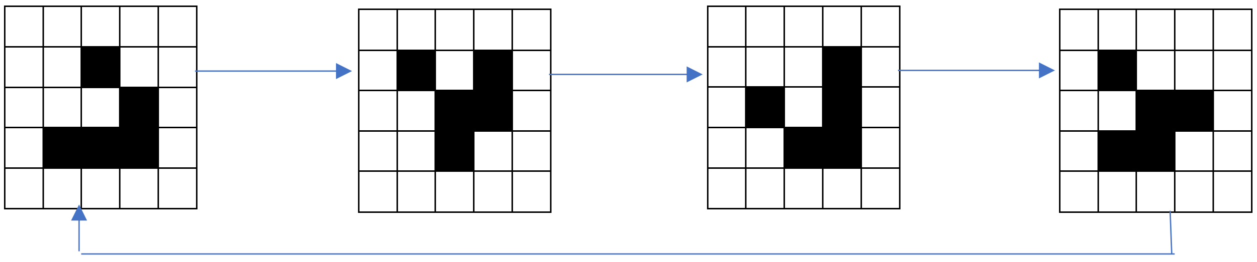


Figure 1.3

Carter Bays (1987) explored a 3-Dimensional extension of "Conway's Game of Life", showing that there existed a set of rules that closely resembled Conway's Life in 3D, and was later revisited in 2006. There has been little further investigation into this concept since.

Design

Algorithms

For representing the coordinate data, a class called Cell is created. This contains 3 integers, each corresponding to each directional axis: X, Y and Z. Cell also contains an overridden Hashcode, which is the 3 previously mentioned integers concatenated together, which will be used to identify each Cell.

This class is used as a type for data storage, 3 arrayLists of type Cell are used:

- AliveList
- CheckList
- NewAliveList

AliveList lists the Cell's that are alive, CheckList lists the Cell's that the algorithm needs to check and NewAliveList lists the Cell's that will be alive in the next step.
"Conway's Game of Life" has a set of rules that are used to determine how the cells will interact with each other, there 4 integers that represent these rules:

- LowDeath
- HighDeath
- LowBirth
- HighBirth

LowDeath and HighDeath are the lower and upper bounds of the death rule. If an alive cell has neighbours outside of this range, then the cell dies. Otherwise the cell will remain alive in the next generation. Since an alive cell has 26 possible neighbours including none, that range of values for these rules are 0 to 26.

LowBirth and HighBirth similarly are the lower and upper bounds of the birth rule. If a dead cell has neighbours between this range, then cell will come alive, otherwise it will remain dead. Similarly, these rules have 26 possible neighbours including none, however for reasons that will be explained in the implementation, the range is narrower than this.

The size of the board is determined by an integer called Extent. Typically, this value is arbitrarily large in the standard 2D Game of Life, however the board size is proportional to the cube of Extent in 3D, rather than the square of Extent in 2D, the increased processing requirements of this change limits the board size.

In 2D, an Extent of 1000, yields a total of 1 million cells with each one having 8 neighbours to check, for a total 8 million calculations per generation. In comparison, the same Extent value in 3D yields 1 billion cells with each cell having 26 neighbours to check for a total of 26 billion calculations per generation.

At a frame rate of 60 FPS, that would be 1.5 Trillion calculations per second, where the execution time for each calculation varies. This level of processing is not feasible on consumer hardware, therefore Extent is limited to a maximum of 50, although this can be increased if more processing power is available.

jMonkeyEngine is a game engine made especially for 3D Java Development. We will be using it to create this program as it is an easy to use collection of libraries that are specifically designed for rapid 3D application development.

Interface/Rendering

There will be a save/load function implemented, for allowing saving of individual iterations for examination and for use as new initial states. The current rules at the time of saving will also be saved with the iteration, so comparisons can be made on the effects of different rules on the same pattern.

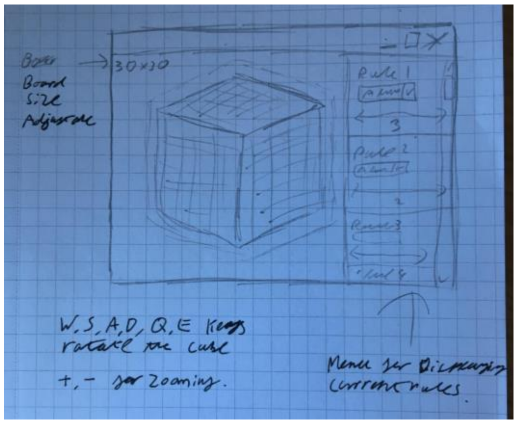
A method of representing the output of the algorithm in a 3D space is required, this should be a cube made up of smaller cubes. The large cube should have an upper bound on size, which will be dependent on testing performance. However, a size of 50 cubes per edge should be sufficient. Each small cube should be able to toggle its visibility depending on its state.

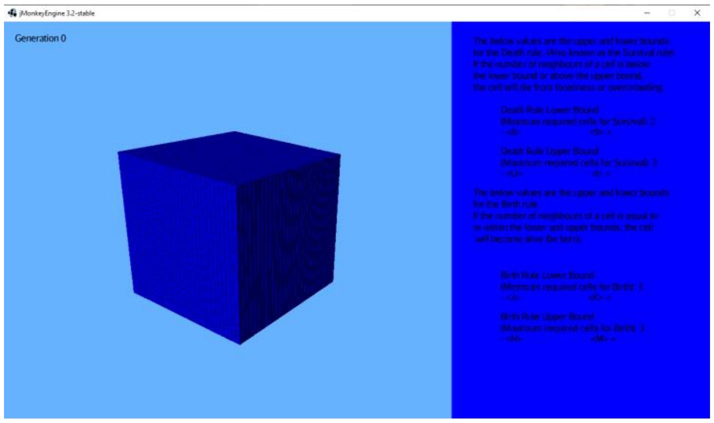
There should be a method of viewing the cube from different perspectives, therefore I will implement a way of rotating the cube or the camera around a fixed point. A method of zooming on the cube will accompany this.

Finally, the program should display the following information relating to the simulation:

- Board/Cube Size
- Number of iterations/Generations
- Current rules of the simulations

Design





The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound
(Minimum required cells for Survival): 2

Death Rule Upper Bound
(Maximum required cells for Survival): 3

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

Birth Rule Lower Bound
(Minimum required cells for Birth): 3

Birth Rule Upper Bound
(Maximum required cells for Birth): 3

Figure 2.1

Figure 2.2

Pseudocode

This method generates the larger cube as previously described. Extent is the number of cubes per edge of the larger cube. Another method will take the coordinates as its input and creates an instance of a smaller cube with a pre-set size, colour and lighting interactions. It will also store the newly generated cube in a 3D array; the position in the array will match the coordinate it represents.

```
procedure generateCube()
for (int z = 0; z < extent; z++)
    for (int y = 0; y < extent; y++)
        for (int x = 0; x < extent; x++)
            create an instance of a cube geometry at coordinates x,y,z
            attach cube to scene
        end for
    end for
end for
end procedure
```

Figure 3

Implementation

Part of the design of the algorithm was to implement the standard 2D Conway's Game of Life first, then once such algorithm has been completed, it will be modified to use 3 values for coordinates instead of 2. Early on in development it became clear that a brute-force method, while robust and simple to implement, lacks scalability. A square board with a length of 100 cells has 10,000 cells that need to be checked each step. Whereas, A cube board of the same dimensions, would be required to check 1,000,000 cells each step. A more optimized algorithm is necessary.

Since a cell can only become and remain alive when a specific number of alive cells surround it, only cells that are alive or that are adjacent to an alive cell are worth checking. A modified algorithm was therefore implemented that first produced a list of all the alive cells on the board, then using that "AliveList", a second list of cells that need to be checked can be calculated. From there, "CheckList" was used to determine if that cell should be alive or dead. This is where the rules of Conway's Game of Life are used.

This optimised algorithm accurately simulates the original, by taking in a list of the cells that are alive, it returns a list of which cells will be alive in the next step. This list is used to update the 3D rendering.

The 2D algorithm was then modified to use 3D coordinates. The test results of this algorithm are also found in the testing section. The following is an explanation of the pseudocode of the optimised algorithm for 2D GOL.

Generate Board

This method uses AliveList and converts it into an array of size "extent" called Board. Board is used for neighbour calculations.

```
procedure generateBoard()
for each cell in aliveList
    board[cell.x][cell.y] = true
end for
end procedure
```

Figure 4.1

Generate CheckList

This method uses AliveList and adds its neighbours to Checklist. CheckList is used to calculate which cells become Alive. Since a dead cell requires a specific number of alive neighbours to become alive, only dead cells that are neighbouring alive cells are considered by the algorithm. This optimizes the algorithm significantly, as only cells that are alive or have alive neighbours are checked, rather than checking all cells on the board.

```
procedure generateCheckList()
for each cell in aliveList
    add a new Cell(cell.x-1,cell.y-1) to checkList
    add a new Cell(cell.x-1,cell.y) to checkList
    add a new Cell(cell.x-1,cell.y+1)  to checkList
    add a new Cell(cell.x,cell.y-1) to checkList
    add a new Cell(cell.x,cell.y+1) to checkList
    add a new Cell(cell.x+1,cell.y-1) to checkList
    add a new Cell(cell.x+1,cell.y) to checkList
    add a new Cell(cell.x+1,cell.y+1) to checkList
end for
end procedure
```

Figure 4.2

Remove Invalid Checks

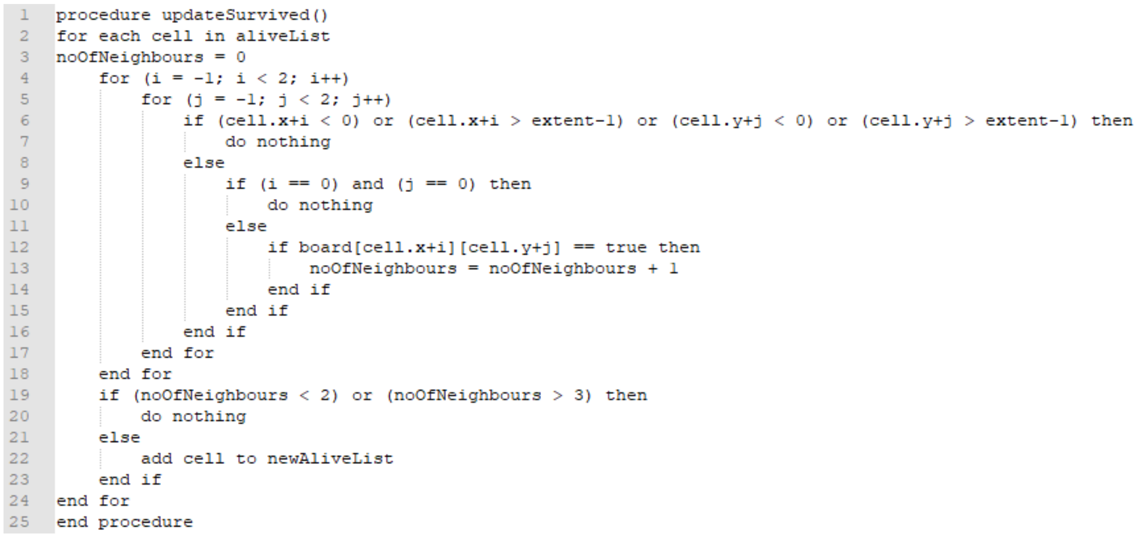
This method removes all checks that would refer to cells that appear outside the bounds of the board. If a cell is out of bounds, it is not relevant and can be discarded.

```
procedure removeInvalidChecks()
declare List<Cell> newCheckList
for each cell in checkList
    if (cell.x < 0) or (cell.x > extent-1) or (cell.y < 0) or (cell.y > extent-1) then
        do nothing
    else
        add cell to newCheckList
    end if
end for
clear checkList
add all cell in newCheckList to checkList
end procedure
```

Figure 4.3

Update Survived

This method checks the alive cells and determines if each one should be alive or dead. In Conway's Game of Life, a cell survives to the next generation if it has 2 or 3 neighbours, it dies under any other condition. Line 19 is to be changed prior to deployment to use the lowDeath and highDeath variables.

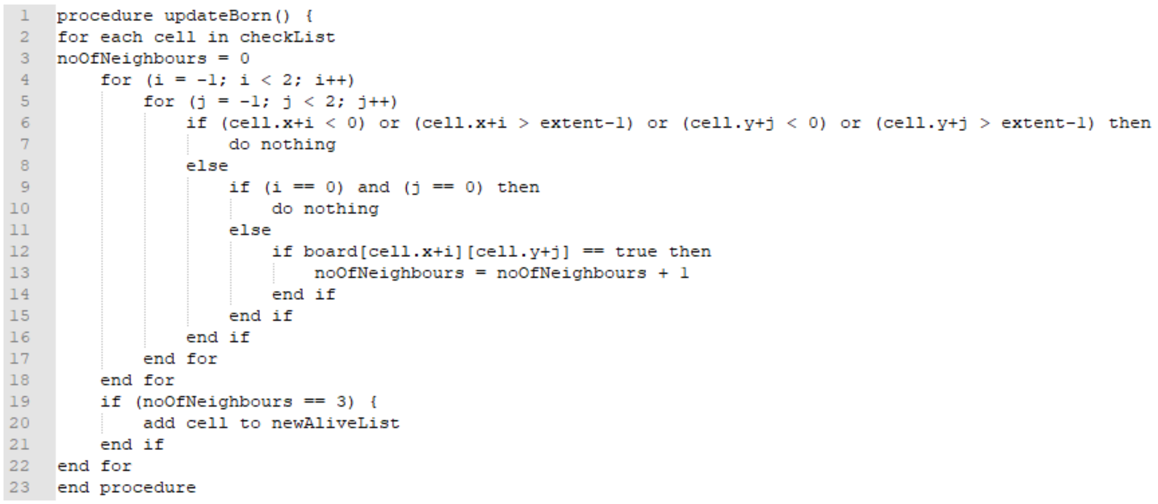


```
 1  procedure updateSurvived()
 2  for each cell in aliveList
 3  noOfNeighbours = 0
 4      for (i = -1; i < 2; i++)
 5          for (j = -1; j < 2; j++)
 6              if (cell.x+i < 0) or (cell.x+i > extent-1) or (cell.y+j < 0) or (cell.y+j > extent-1) then
 7                  do nothing
 8              else
 9                  if (i == 0) and (j == 0) then
10                      do nothing
11                  else
12                      if board[cell.x+i][cell.y+j] == true then
13                          noOfNeighbours = noOfNeighbours + 1
14                      end if
15                  end if
16              end if
17          end for
18      end for
19      if (noOfNeighbours < 2) or (noOfNeighbours > 3) then
20          do nothing
21      else
22          add cell to newAliveList
23      end if
24  end for
25  end procedure
```

Figure 4.4

Update Born

This method checks the alive cells and determines if each one should be alive or dead. Line 19 is to be changed prior to deployment to use lowBirth and HighBirth.



```
 1  procedure updateBorn() {
 2  for each cell in checkList
 3  noOfNeighbours = 0
 4      for (i = -1; i < 2; i++)
 5          for (j = -1; j < 2; j++)
 6              if (cell.x+i < 0) or (cell.x+i > extent-1) or (cell.y+j < 0) or (cell.y+j > extent-1) then
 7                  do nothing
 8              else
 9                  if (i == 0) and (j == 0) then
10                      do nothing
11                  else
12                      if board[cell.x+i][cell.y+j] == true then
13                          noOfNeighbours = noOfNeighbours + 1
14                      end if
15                  end if
16              end if
17          end for
18      end for
19      if (noOfNeighbours == 3) {
20          add cell to newAliveList
21      end if
22  end for
23  end procedure
```

Figure 4.5

Remove Duplicates

This method removes duplicate cells from NewAliveList. Since some cells may be close enough together for them to share the same neighbours, there will be some overlapping that will go undetected by generateCheckList. This will result in some cells being counted multiple times. While this does not affect the simulation results, it will affect performance and memory usage due to a gradual build-up of duplicated cells in the list. Adding NewAliveList to a LinkedHashSet will remove the duplicates.

```
procedure removeDuplicates()
declare setAlive as LinkedHashSet of type cell
add all cells of newAliveList to setAlive
clear newAliveList
add all cells of setAlive to newAliveList
end procedure
```

Figure 4.6

Make Changes

This method updates the board array with the new board state and displays the new pattern.

```
procedure makeChanges()
for each cell in aliveList
    detach cell from Scene
end for
re-initialize board
for each cell in newAliveList
    board[cell.x][cell.y][cell.z] = true
    attach cell to Scene
end for
refresh Scene
end procedure
```

Figure 4.7

Update Lists

This method updates AliveList with the contents of NewAliveList, this prepares the algorithm for the next iteration.

```
procedure updateLists()
clear aliveList
add all cells in newAliveList to aliveList
clear newAliveList
```

Figure 4.8

Run Cycle

Run cycle is a method that this executed every time a new iteration is required. This is usually executed once every frame. It is made up of all the previously mentioned methods.

```
procedure runCycle()
    generateBoard()
    generateCheckList()
    removeInvalidChecks()
    updateSurvived()
    updateBorn()
    removeDuplicates()
    makeChanges()
    updateLists()
    increment generation
    display generation
end procedure
```

Figure 4.9

Once the 3D algorithm and the interface were implemented, it was all brought together into a single program. The algorithm runs between frame updates resulting in one iteration every frame, effectively locking the simulation to the framerate. Whether or not the simulation is running is controlled via key bindings. In addition to running the simulation for as long as is required, the simulation can also be stepped by 1 generation or 100 generations. The 1 step is useful for analysing patterns in greater detail, whereas the 100 step is better suited for rapidly testing Rulesets for certain behaviours. Such behaviours are discussed later.

Since life is not spontaneously created, each simulation requires an initial pattern in order to start. Therefore, a method of loading or generating a pattern was required. This was achieved by the implementation of a load function that uses a data file that stores the rules and coordinate data for a specific pattern. Using this as the initial state has the benefit of replicating a previously discovered behaviour and it also allows us to test different rules to see if such patterns can persist between rules. It is improbable that a small set of hand-crafted patterns will yield interesting sub-patterns for each Ruleset, therefore a method for randomly generating a "cloud" of cells was implemented as shown below.

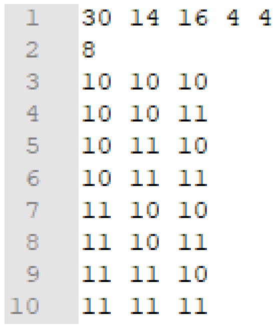
```
procedure generateRandomPat()
    initialize aliveList, newAliveList and checkList
    for z = 0 to extent-1
        for y = 0 to extent-1
            for z = 0 to extent-1
                generate a random number R
                if R < 0.3
                    boolean Random = True
                else false
                end if
                detach cell from Scene
                if random = true
                    add cell to aliveList
                    attach cell to scene
                end if
            end for
        end for
    end for
    refresh Scene
end procedure
```

Figure 4.10

When the method is executed, it overwrites the AliveList with random cell coordinates and resets the generation counter. In theory, Life should form naturally from the cloud of random cells on certain rules.

When a pattern of interest eventually arises, we should be able to save it for later study. Therefore, a save function has been implemented to accompany the load function. The filename for the save file is created using the size of the board and the current time to create a unique name, the filename can also be chosen by the user to identify patterns of interest for later. The save and load functions both use a custom file format (.gol).

The first line is made up 5 values: Extent, lowDeath, highDeath, lowBirth and highBirth. The second line states the number of alive cells and the remaining lines represent the coordinate data of all the alive cells that make up that specific pattern. The file shown below is called 30-2020-01-27-11-25-30



```
30 14 16 4 4
8
10 10 10
10 10 11
10 11 10
10 11 11
11 10 10
11 10 11
11 11 10
11 11 11
```

Figure 4.11

The rules interface allows the adjustment to the simulation rules; however, these values are not locked in once the simulation starts and therefore allows on the fly modifications to the existing rules.

Testing

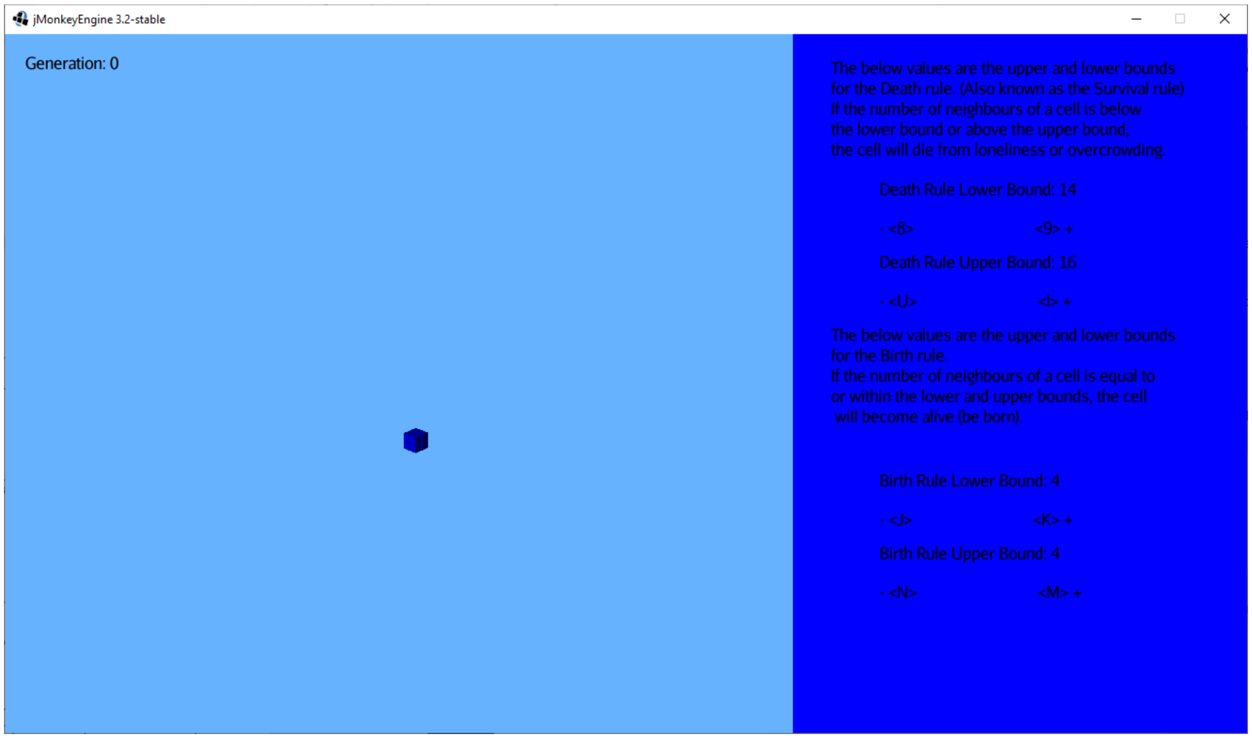
Test ID	Description	Purpose	Expected Result	Actual Result	Notes
1	Rule upper and lower bounds.	To ensure a user cannot select a rule that would be impossible or invalid.	Death Rules Lower: 0 Upper: 26 Birth Rules Lower: 1 Upper: 26	Death Rules Lower: 0 Upper: 26 Birth Rules Lower: 1 Upper: 26	Working as intended
2	Upper rule cannot be less than Lower rule and vice versa.	To ensure a user cannot select a rule that would be impossible or invalid.	Lower rule is less than Upper rule. User is prevented from selecting such rule.	Lower rule is less than Upper rule. User is prevented from selecting such rule.	Working as intended
3	Data files of the correct format are loaded correctly.	To test the Load function executed correctly and accurately interprets the data file.	File 30-2020-01-27-11-25-30-possibleglider.gol Is loaded correctly. Rule is set to (14 16 4 4).	See Figure 5.1	Working as intended
4	Load is aborted by User	To test robustness. Ensures program does not crash.	Load window is closed. Program resumes with existing data and rules.	Load window is closed. Program resumes with existing data and rules.	Working as intended
5a	Invalid/ Corrupted File is loaded. (More coordinates than Data)	To test robustness. Ensures program does not crash.	Load window is closed. Program resumes with loaded data and rules.	NumberFormat Exception on Input String "" Program terminates.	Catch added for exception. Program loads data and rules correctly. All data is loaded

5b	Invalid/ Corrupted File is loaded. (More data than Coordinates)	To test robustness. Ensures program does not crash.	Load window is Closed. Program resumes with loaded data, minus the additional data.	Load window is Closed. Program resumes with loaded data, minus the additional data. See Figure 5.2	Working as Intended. As much data as possible is loaded.
5c	Invalid/ Corrupted File is loaded. (Invalid Rules)	To test robustness. Ensures program does not crash.	Load window is Closed. Program resumes with loaded data. Invalid Rules are loaded.	Load window is Closed. Program resumes with loaded data. Invalid Rules are loaded. See Figure 5.3	Only through file editing would invalid rules be possible. Invalid rules may cause unintended behaviours but will not cause a crash.
5d	Invalid/ Corrupted File is loaded. (Missing Rule Line, 1st line of Data file)	To test robustness. Ensures program does not crash.	Load window is Closed. Program resumes with existing data and Rules.	ArrayIndexOut OfBounds Exception. Program terminates.	Catch added for exception. Program resumes with existing data and Rules.
5e	Invalid/ Corrupted File is loaded. (Missing Coordinates line, 2nd line of Data file)	To test robustness. Ensures program does not crash.	Load window is Closed. Program resumes with loaded rules. No data is loaded, Existing data is deleted.	Load window is Closed. Program resumes with loaded rules. No data is loaded, Existing data is deleted. See Figure 5.4	Working as intended. Existing data is deleted before new data is loaded. If the 2nd line is missing, a Number Format Exception occurs and no data is loaded.

6	Save Function correctly saves the current pattern and Rules.	To test the Save function executed correctly and created the save file correctly.	File is saved successfully.	File is saved successfully.	Working as intended. Save file was reloaded by the program and data checked to ensure saving occurred correctly.
7	Save is aborted by User	To test robustness. Ensures program does not crash.	Save window is closed. No new file is created.	Save window is closed. No new file is created.	Working as intended.
8a	File saved with no Filename.	To test robustness. Ensures program does not crash.	Saving with no filename is disallowed.	Saving with no filename is disallowed.	Working as intended. Save button in window will not perform any action without at least 1 character in the filename.
8b	File saved with no data.	To test robustness. Ensures program does not crash.	File is saved successfully.	File is saved successfully.	Working as intended. Blank files are useful for saving rules for later use.

9a	Birth rule/Update Born (Outside range) Rule (0 26 2 26) used. Deaths are disabled.	Ensures birth rule works as intended. Cells with alive neighbours outside the range should not be born.	Board should remain unchanged.	Board should remain unchanged. Figure 5.5	Working as intended. Ran for 100 Generations. No changes.
9b	Birth rule/Update Born (Inside range) Rule (0 26 1 1) used. Deaths are disabled.	Ensures birth rule works as intended. Cells with alive neighbours inside the range should be born.	All 26 neighbours of the starting cell are born after 1 Generation.	All 26 neighbours of the starting cell are born after 1 Generation. Figure 5.6	Working as intended There are 8 dead cell that have 1 neighbour each after the first generation. Those cells would become alive in the 2nd generation.
10a	Death rule/Update Survived (Outside range) Rule (1 1 26 26) Births are disabled.	Ensures death rule works as intended. Cells with alive neighbours outside the range should die. Pattern and rule selected to die out eventually.	Of the 3 Cells (Figure 5.7), the centre cell dies after 1 generation. The remaining cells die in the next generation.	As Expected. After 1 Generation. Figure 5.8 After 2 Generations. Figure 5.9	Working as Intended.

10b	Death rule/Update Survived (Inside range) Rule (0 1 26 26) Births are disabled.	Ensures death rule works as intended. Cells with alive neighbours inside the range should Survive. Pattern and rule selected to cause cells to die, but the pattern to survive.	Of the 3 Cells (Figure 5.7), the centre cell dies after 1 generation. The remaining cells survive in all subsequent generations.	As Expected. After 1 Generation. Figure 5.8	Working as Intended.



Generation: 0

The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 14

- <8> <9> +

Death Rule Upper Bound: 16

- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

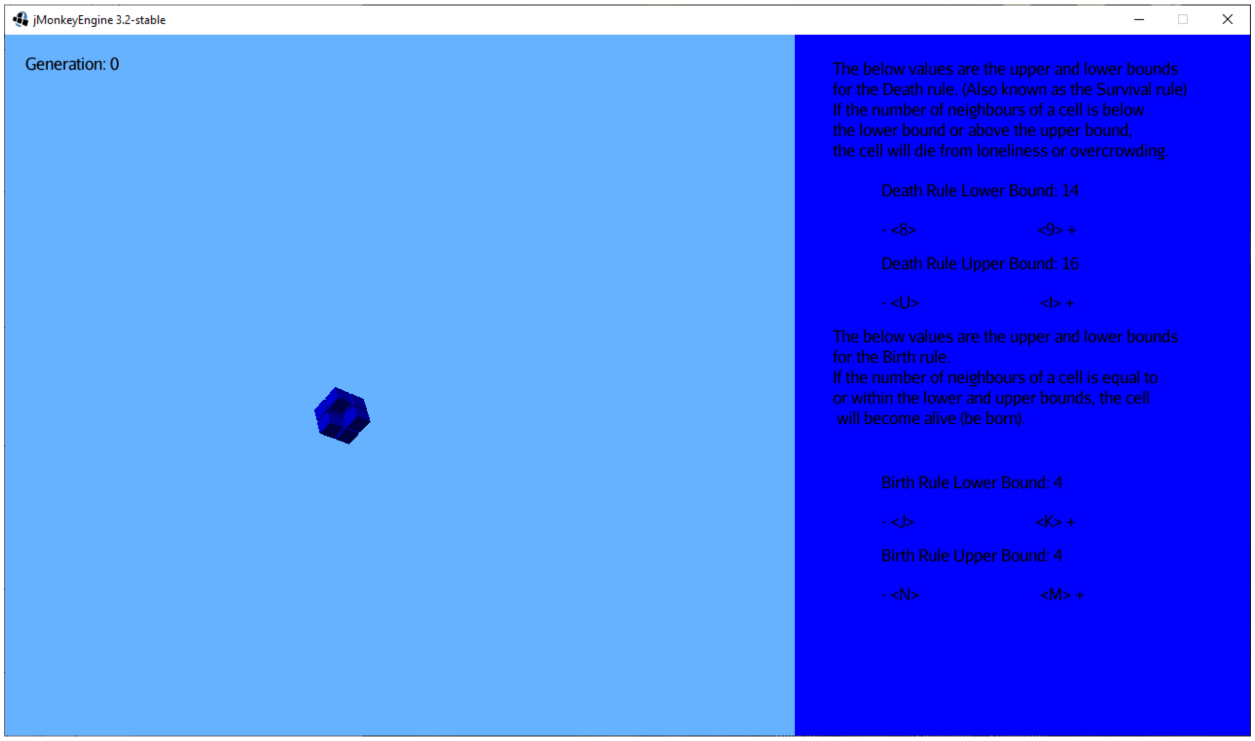
Birth Rule Lower Bound: 4

- <J> <K> +

Birth Rule Upper Bound: 4

- <N> <M> +

Figure 5.1



Generation: 0

The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 14

- <8> <9> +

Death Rule Upper Bound: 16

- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

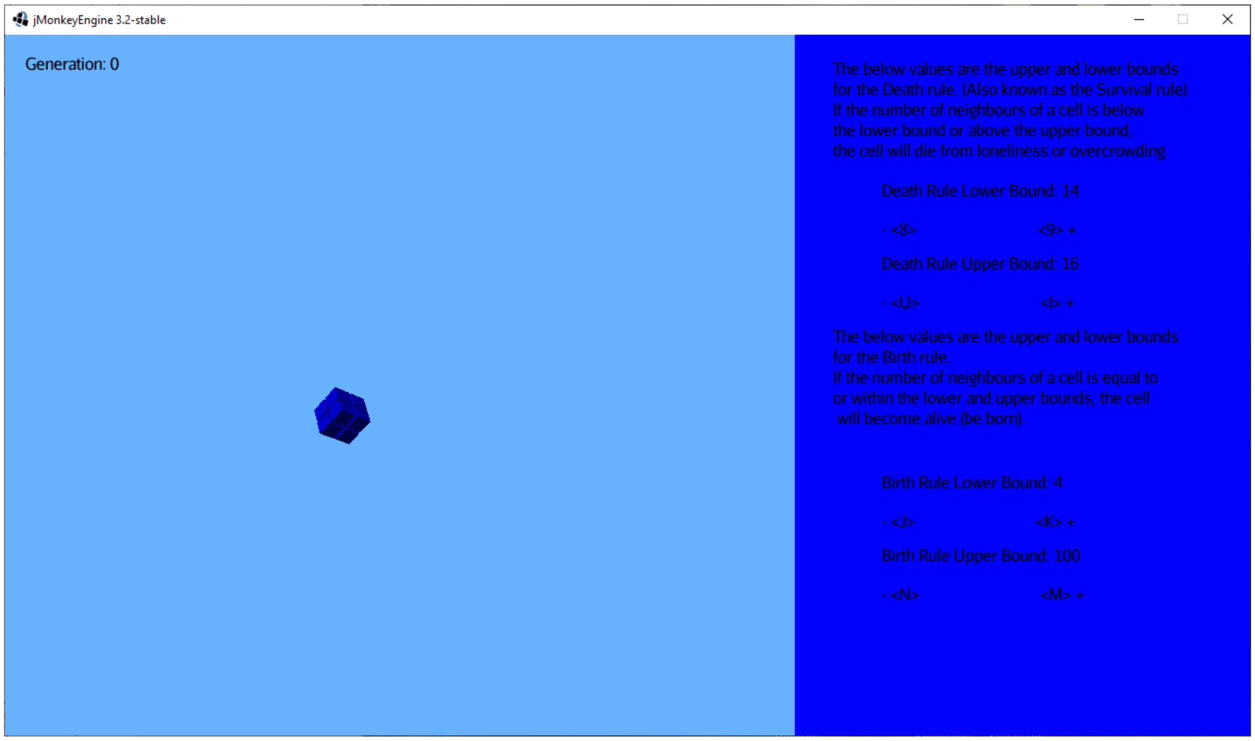
Birth Rule Lower Bound: 4

- <J> <K> +

Birth Rule Upper Bound: 4

- <N> <M> +

Figure 5.2



Generation: 0

The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 14

- <8> <9> +

Death Rule Upper Bound: 16

- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

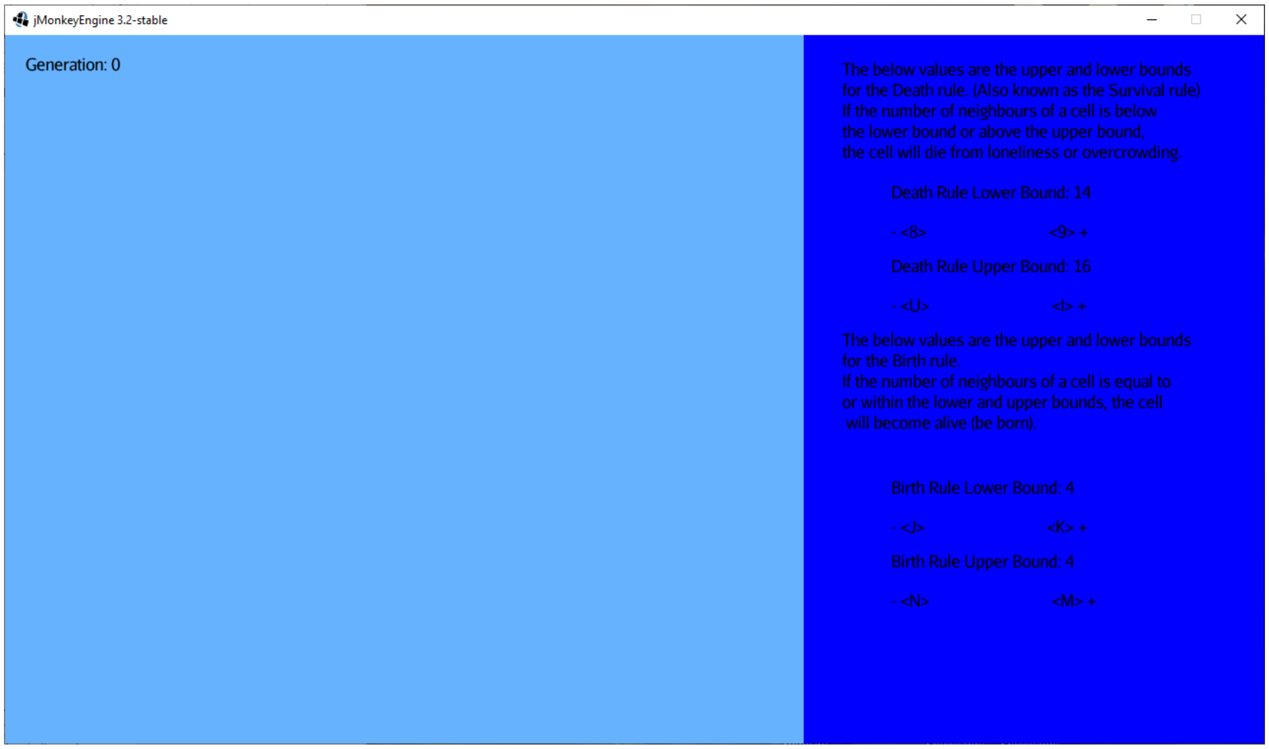
Birth Rule Lower Bound: 4

- <J> <K> +

Birth Rule Upper Bound: 100

- <N> <M> +

Figure 5.3



Generation: 0

The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 14

- <8> <9> +

Death Rule Upper Bound: 16

- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

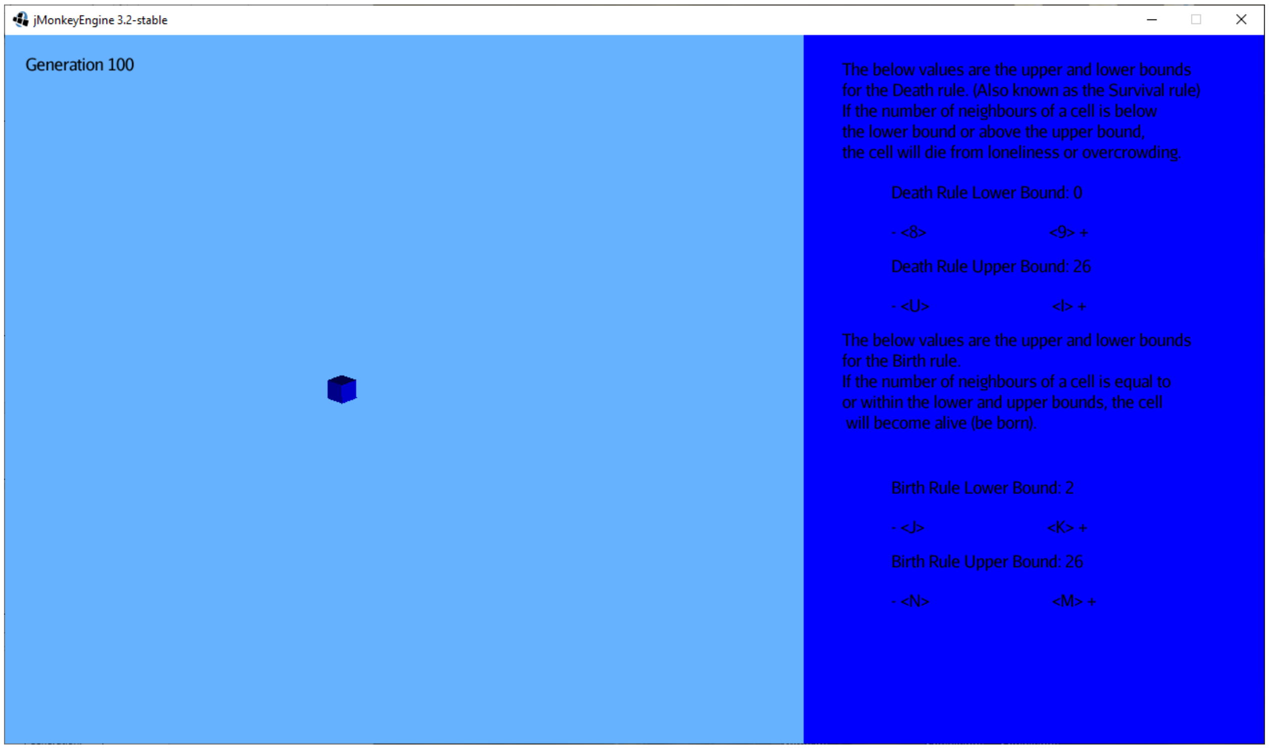
Birth Rule Lower Bound: 4

- <J> <K> +

Birth Rule Upper Bound: 4

- <N> <M> +

Figure 5.4



Generation 100

The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 0

- <8> <9> +

Death Rule Upper Bound: 26

- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

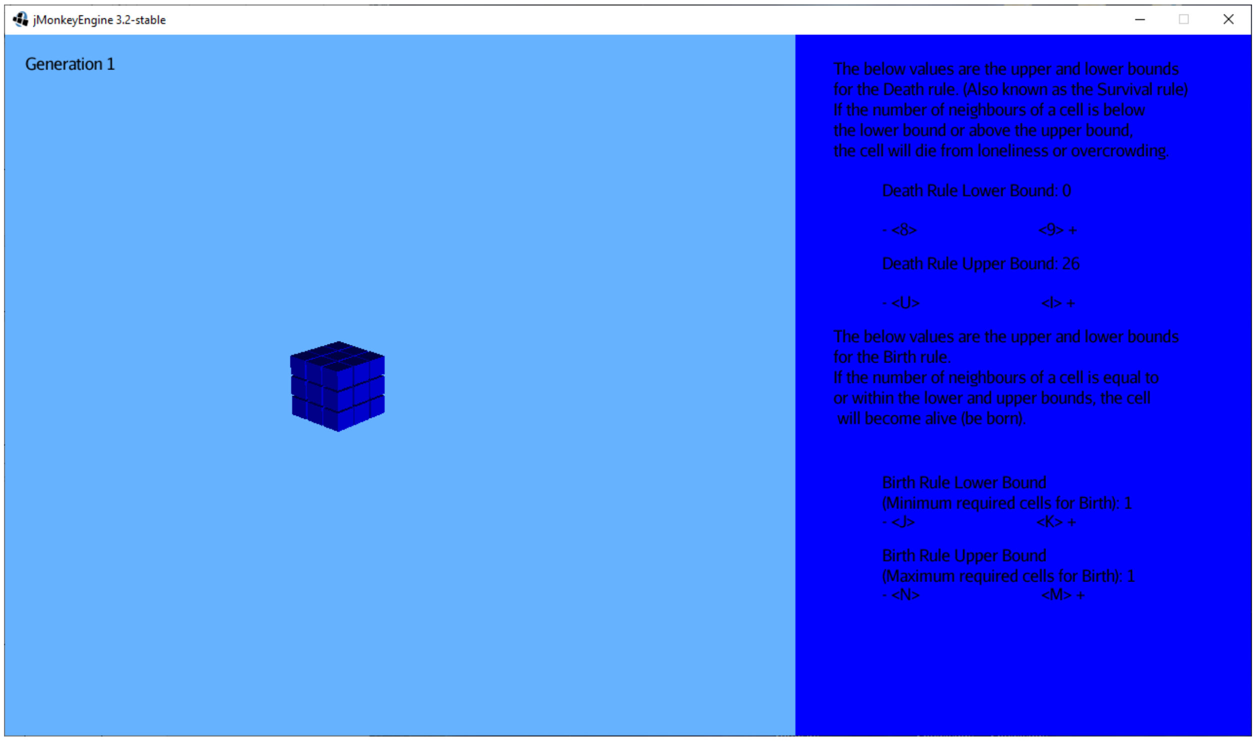
Birth Rule Lower Bound: 2

- <J> <K> +

Birth Rule Upper Bound: 26

- <N> <M> +

Figure 5.5



The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 0

- <8> <9> +

Death Rule Upper Bound: 26

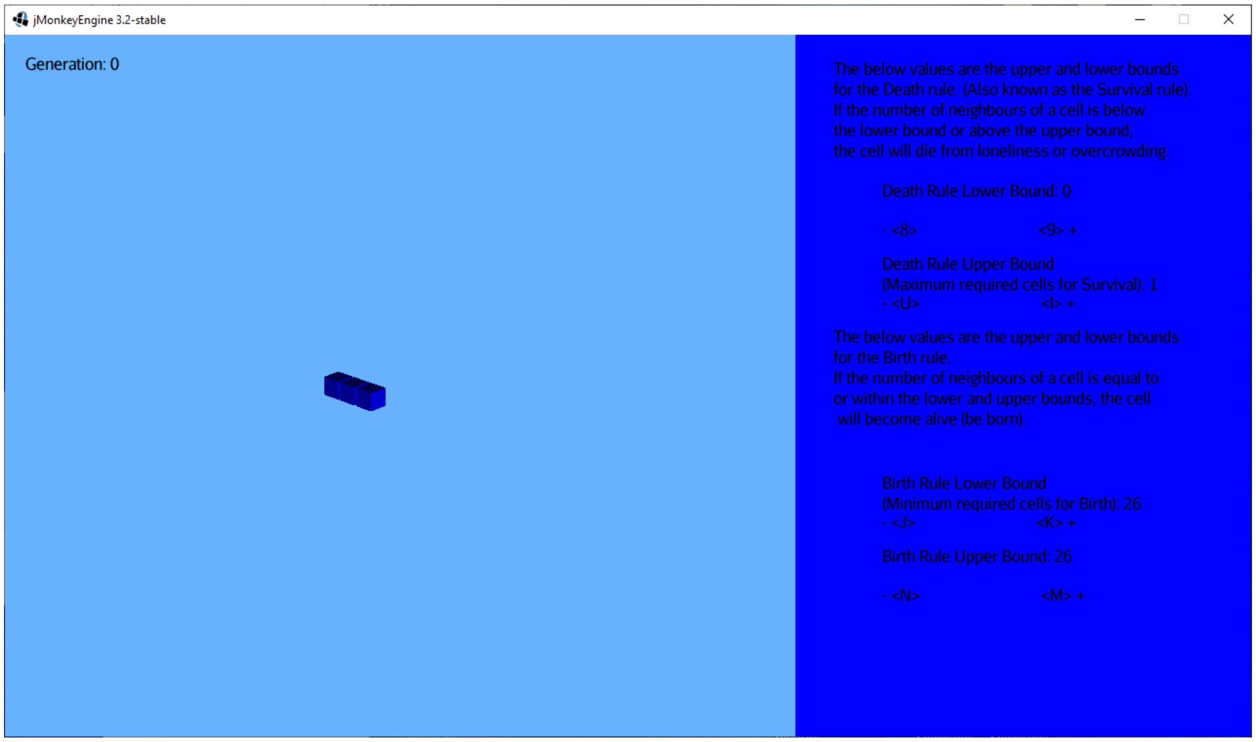
- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

Birth Rule Lower Bound
(Minimum required cells for Birth): 1
- <J> <K> +

Birth Rule Upper Bound
(Maximum required cells for Birth): 1
- <N> <M> +

Figure 5.6



The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 0

- <8> <9> +

Death Rule Upper Bound
(Maximum required cells for Survival): 1
- <U> <I> +

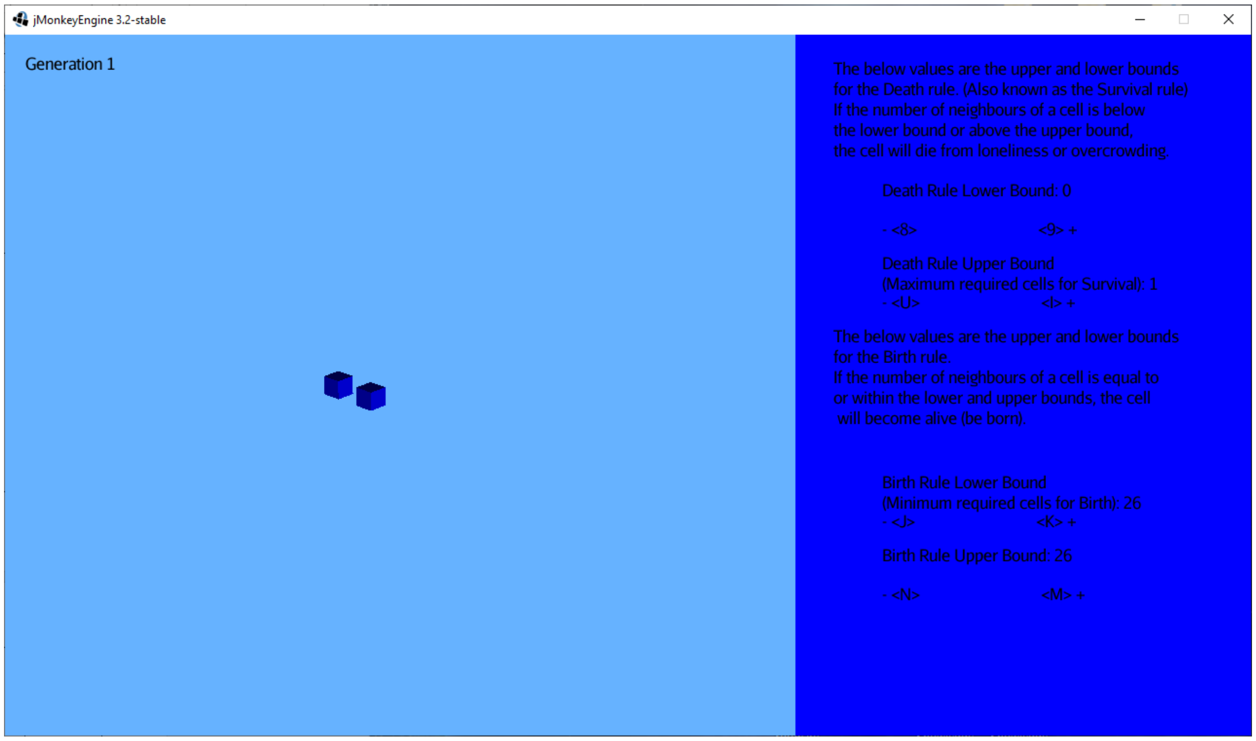
The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

Birth Rule Lower Bound
(Minimum required cells for Birth): 26
- <J> <K> +

Birth Rule Upper Bound: 26

- <N> <M> +

Figure 5.7



The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound: 0

- <8> <9> +

Death Rule Upper Bound
(Maximum required cells for Survival): 1
- <U> <I> +

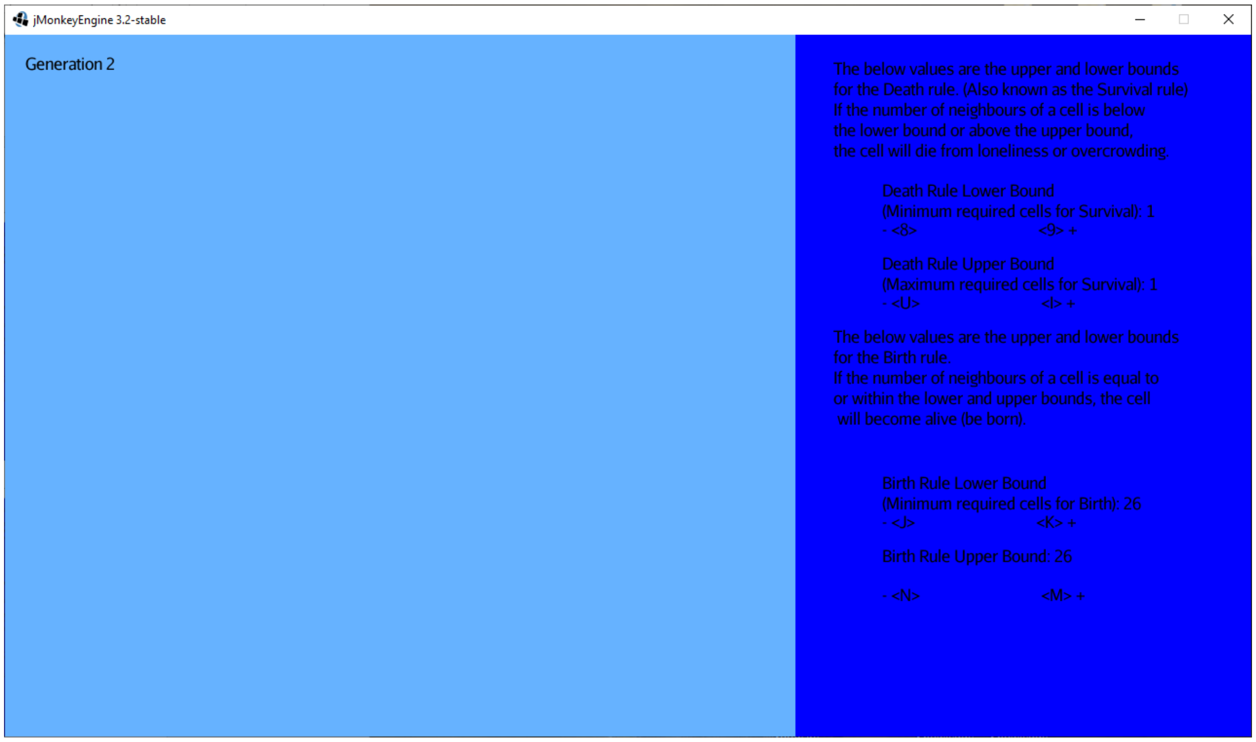
The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

Birth Rule Lower Bound
(Minimum required cells for Birth): 26
- <J> <K> +

Birth Rule Upper Bound: 26

- <N> <M> +

Figure 5.8



The below values are the upper and lower bounds
for the Death rule. (Also known as the Survival rule)
If the number of neighbours of a cell is below
the lower bound or above the upper bound,
the cell will die from loneliness or overcrowding.

Death Rule Lower Bound
(Minimum required cells for Survival): 1
- <8> <9> +

Death Rule Upper Bound
(Maximum required cells for Survival): 1
- <U> <I> +

The below values are the upper and lower bounds
for the Birth rule.
If the number of neighbours of a cell is equal to
or within the lower and upper bounds, the cell
will become alive (be born).

Birth Rule Lower Bound
(Minimum required cells for Birth): 26
- <J> <K> +

Birth Rule Upper Bound: 26

- <N> <M> +

Figure 5.9

During testing, a previously unknown bug was discovered when the following pattern (Figure 5.10) was applied to Rule (5 6 5 5). The pattern was originally found by Carter Bays and is a "H" shaped oscillator of period 3 and it was used in the testing to verify that the algorithm would function as expected when a previously discovered pattern and Ruleset was input.

After 1 generation, the pattern was simulated as expected. However, in the 2nd Generation, 2 cells did not die (Figure 5.11). The pattern died within 3 generations of this event. Upon inspection of the source code it was discovered that a variable representing the "Y" coordinate of a cell, was reading the "X" coordinate instead. This resulted in the removal of the wrong/non-existent cells and incorrect neighbour calculations. This was a simple logic error and was quickly changed to read from the correct variable.

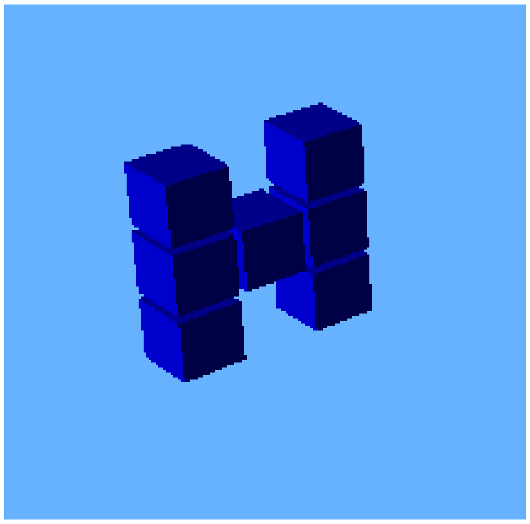


Figure 5.10

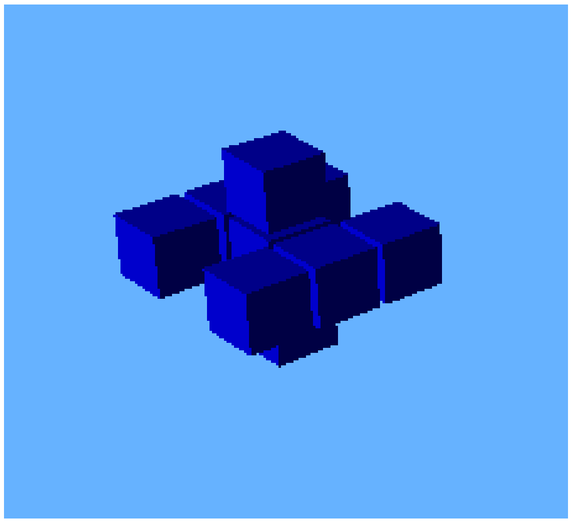


Figure 5.11

Experimentation

The secondary goal of this project is to experiment with Rulesets and discover interesting patterns. However, since each cell has 26 neighbours, there are 27 possibilities for each of the 4 rules, which results in 531,441 unique Rulesets to analyse. This number can be reduced given each pair of rules forms a range of neighbour values for their respective functions, (i.e. lowDeath and highDeath form the death rule range.). With this in mind, any rule where the lower bound is higher than the upper bound can be eliminated. This can be given by the formula, where n is the number options for each individual rule:

$$x = \frac{n(n+1)}{2}$$

Substituting 27 for n, gives us 378 combinations for each pair of rules. The product of each pair gives us the total number of possible combinations: 142,884. An improvement on 531,441, however this can be reduced further.

Since the algorithm requires that a cell must have at least 1 neighbour for it to be checked and that a birth rule of {0,0} (hereby referred to as B{0,0}) implies that only a cell with no neighbours can become alive, B{0,0} will never be satisfied. This in turn will prevent any births from occurring, which in the majority of cases will cause the rapid death of all cells, with the exception of cells that don't satisfy the death rule, in which case the pattern will be static. B{0,0} can therefore be discarded.

Additionally, since B{0,x} requires at least 1 alive neighbour in order to be considered by the algorithm and B{1,x} requires at least 1 alive neighbour to become alive when it is considered, both of these rules have the same effect on the simulation. Therefore, any birth rule that has a lower bound of 0 should be discarded. The ability to select such a rule has been disabled by default.

A similar argument can be applied to a death rule of {0,26} (hereby referred to as D{0,26}), as this implies that cells cannot die. Therefore, any pattern would either be stable or explode, depending on the birth rule. Therefore D{0,26} can also be discarded.

On a side note, a Ruleset of {B{0,0} D{0,26}} (hereby referred to as (0 0 0 26)) is stable for any pattern. Since with these rules, no births or deaths can occur, it has the effect of disabling the interactions between cells. The application of this as a default Ruleset for seed patterns could be useful as a preventative measure for accidental simulation continuation, or as a start point for a systematic approach to testing Rulesets. It was highly likely that Rulesets close to these extremes will exhibit similar behaviours, however, further analysis was required to confirm.

Since birth rules that have a lower bound of 0 are no longer considered, the number of combinations for the birth rule is now 351, down from 378. Death rule combinations is down to 377. This reasoning has allowed us to eliminate 10,557 Rulesets from consideration, with 132,327 left to test.

Upon deeper analysis of the algorithm, a set of special case Rulesets became apparent:

B{x,7}, B{x,11}, B{x,17}, B{8,y}, B{12,y}, B{18,y}
D{0,7}, D{0,11}, D{0,17}, D{7,26}, D{11,26}, D{17,26}

These special cases are based on the possible number of neighbours for a cell on the edge of the board. Considering this algorithm does not include a board wrap, a cell that exists on the edge, face or corner will always have fewer neighbours than other cells. For example, a birth rule B{8,x>8}, will result in the corner cells never becoming alive as they only have a maximum of 7 neighbours. A further example, a death rule D{0,17} will result in every cell on the edge, corner and face surviving, creating a hollow cube structure with an outer shell of made of static cells.

Since the board size is bounded by processing limitations and that the standard 2D Conway's Game of Life is traditionally played on an arbitrarily large board or theoretically infinite. The effect of reduced neighbours has not been explored in depth and is not representative of Conway's GOL, therefore, we should only consider Rulesets were neighbour constraints are not a major factor. In that case, any Ruleset that requires 18 or more neighbours will be revisited in the future. That brings down the number of Rulesets to 153 for Birth rules and 171 for Death rules, which eliminates 106,164 Rulesets leaving 26,163 remaining.

Data

The number of potential Rulesets has now been reduced to a small enough subset that systematic testing was possible. The testing of each rule was carried out in the following manner:

1. A Randomised "Seed" Pattern is generated
2. The Generated pattern is run with one of the potential Rulesets for 100 Generations
3. After 100 Generations, the resulting pattern was analysed and classified as 1 of 5 possible states. (Static, Dead, Exploded, Repeating or Other)
4. The process was repeated 3 times for each Ruleset and the majority state was recorded. If no majority exists, the process was repeated until a majority appears or otherwise labelled "Other".

Any pattern that dies out should be considered an invalid rule as if there are no cells then no further changes to the pattern can be made. Also, the rules for 2D Game of life should not result in Explosive growth. *(Conway (1999)).* Therefore, any pattern that Explodes or Dies are invalid rules. These will be coloured Red.

Rulesets that form oscillating patterns have the most potential as Game of life analogues and therefore were coloured Green. Static patterns show promise and require further analysis as oscillating patterns may exist within these rules, the nature of the rule may prevent repeating patterns from forming frequently or naturally. Static patterns and patterns that cannot be classified are coloured Yellow.
The following graphic is the result of the above testing.

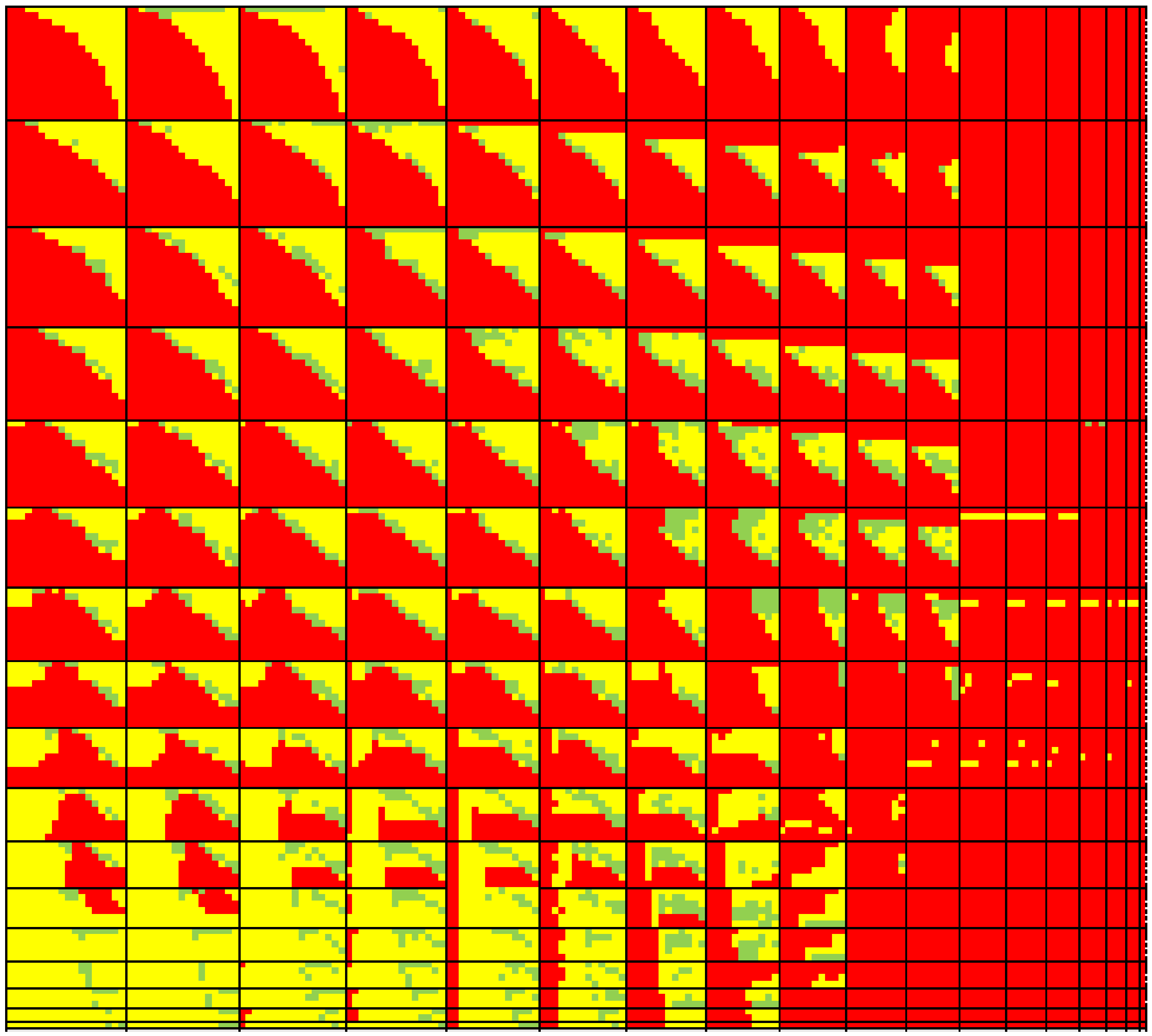


Figure 6

Throughout the above testing, several patterns were observed to exhibit similar and classifiable behaviours. As shown in Figure 6, most of the Rulesets result in either explosive growth or death. Repeating patterns (Green) tend to rise for rules on the boundary of the Red and Yellow regions and with less frequency, which indicates that such behaviours are highly sensitive to its rules.

The Statics

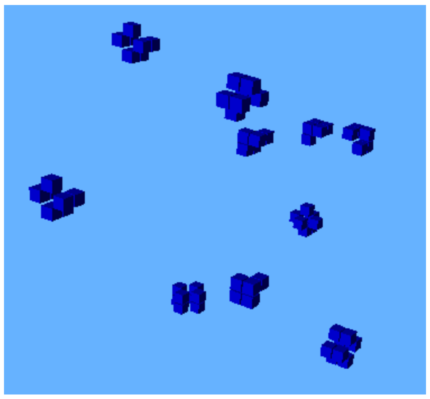


Figure 7.1.1

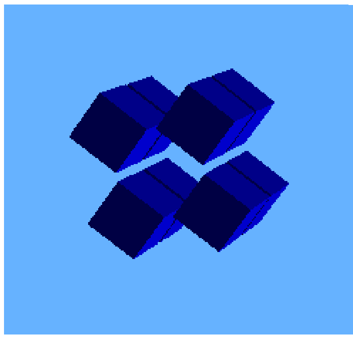


Figure 7.1.2

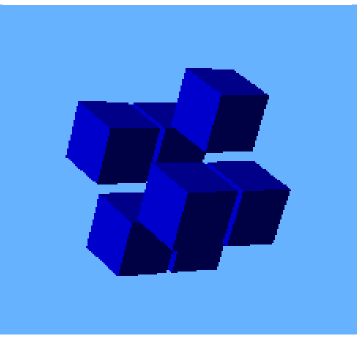


Figure 7.1.3

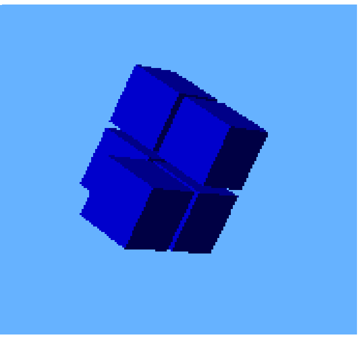


Figure 7.1.4

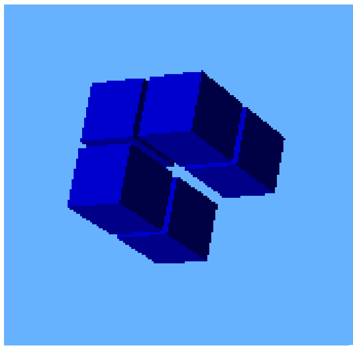


Figure 7.1.5

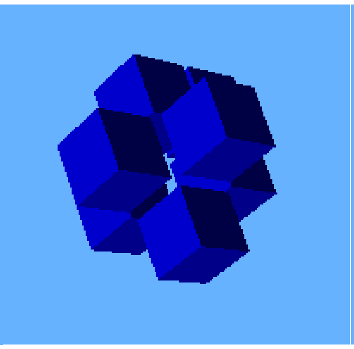


Figure 7.1.6

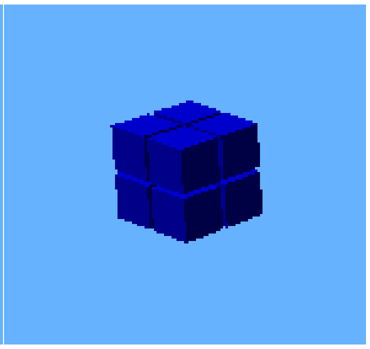


Figure 7.1.7

Most patterns resulting from Rule (4 5 6 6) tend to leave the same few static patterns behind. These five patterns also appeared in rules (3 5 6 6), (4 7 6 6), (4 5 7 7) and (5 6 5 5).
Rules (4 5 6 6) and (5 6 5 5) also produced oscillating patterns, albeit not as frequently as the static patterns.
Figure 7.1.7 is another stable pattern when rule (7 7 7 7) is applied. This pattern is the equivalent of a 2D Block pattern.

The Oscillators

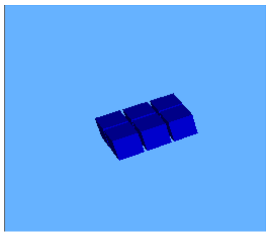


Figure 7.2.1

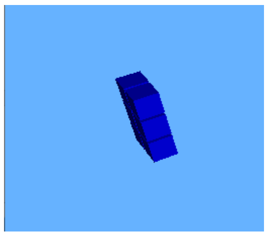


Figure 7.2.2

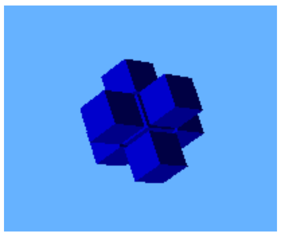


Figure 7.2.3

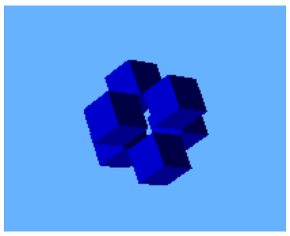


Figure 7.2.4

Rule (4 5 6 6) occasionally produces the above oscillators. For Figure 7.2.1, the oscillator is equivalent of a 2D blinker.

Figure 7.2.3 is made from a single cell with another cell on each of its 6 faces. Since each outer cell either has 4 or 5 cells at any point in time, the outer cell will remain alive. The inner cell has 6 neighbours and therefore will die in the next generation, however since the centre cell has 6 neighbours it will become alive again as soon as it dies. The pattern has a period of 2 and has similarities to a Beacon oscillator in 2D Game of Life.

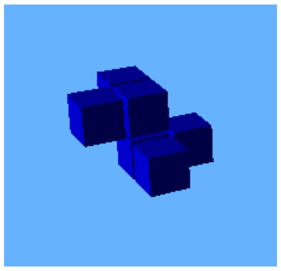


Figure 7.3.1

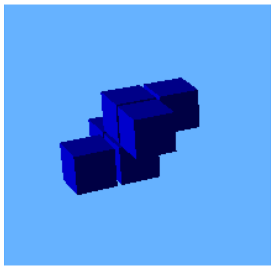


Figure 7.3.2

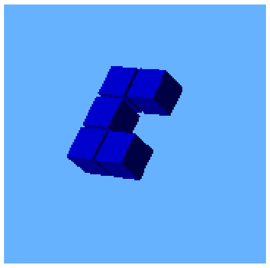


Figure 7.3.3

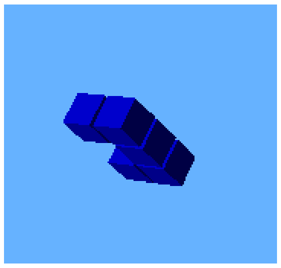


Figure 7.3.4

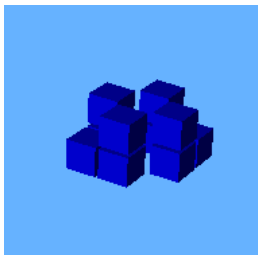


Figure 7.3.5

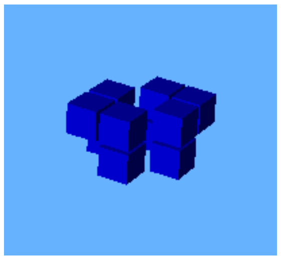


Figure 7.3.6

Rule (5 6 5 5) also produces the oscillators shown by Figure 7.2.1 and 7.2.3. This rule also produces three additional oscillators shown by Figure 7.3.1, 7.3.3 and 7.3.5. The first oscillator has a period of 2 and each generation appears to be the mirror image of each other. The second oscillator (Figure 7.3.3) also has a period of 2. Since none of the 5 initial cells have 5 neighbours, all will die in the next generation, however there are 5 dead cells that have 5 alive neighbours and will therefore become alive.

Both generations of this oscillator are identical in structure with only the orientation differing. The third oscillator is similar to a previously found static pattern (Figure 7.1.2) with 2 additional cells.

Each additional cell has 4 alive neighbours and therefore will die in the next generation, however there is a corresponding dead cell that has 5 neighbours, which will therefore come alive, resulting in Figure 7.3.6.

The Fluidics

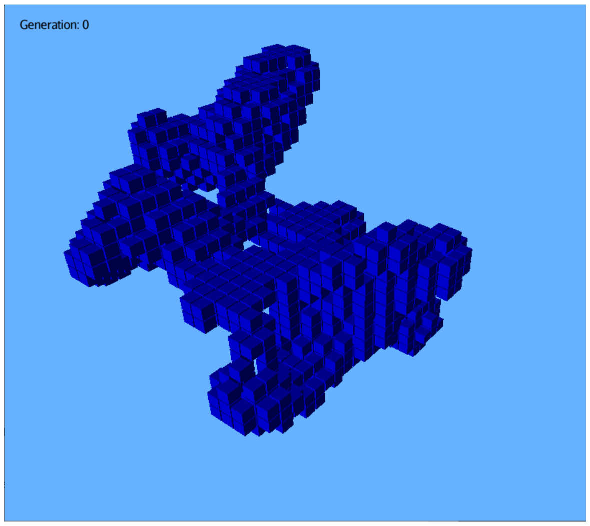


Figure 7.4.1

Unlike Rule (8,18,10,12), Rule (8,15,11,12) always forms a single connected structure. The larger bubbles of cells tend to be connected by thinner strands of cells. However, like Rule (8,18,10,12), this pattern is also Static.

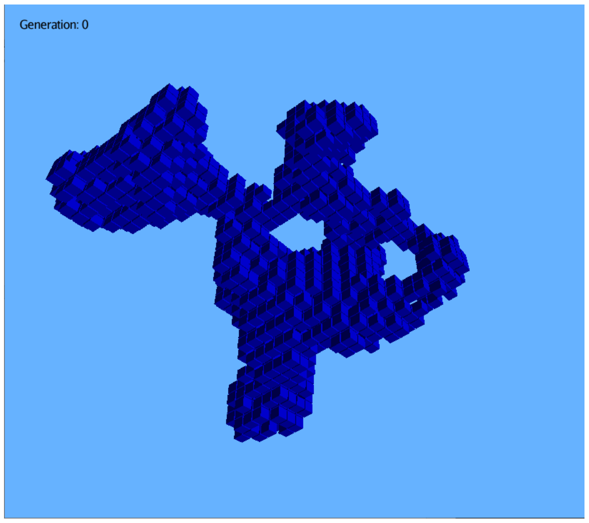


Figure 7.4.2

Alternate view from rear, the strand-like connecting structures are more clearly visible.

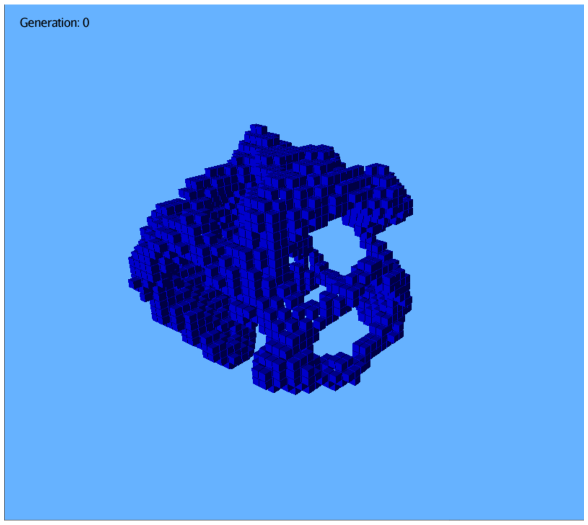


Figure 7.5.1

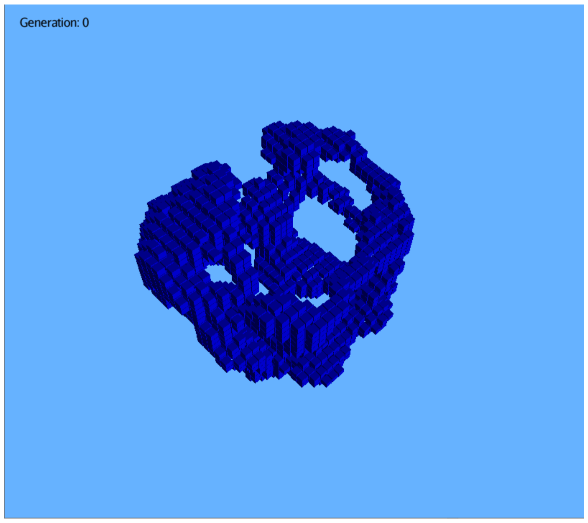


Figure 7.5.2

Rule (8 15 11 13) forms a Static pattern similar in structure to the patterns generated by Rule (8 15 11 12). The strand-like structures visible in Figure 7.4.2 are more exaggerated in this pattern.

Alternative view of Figure 7.5.1

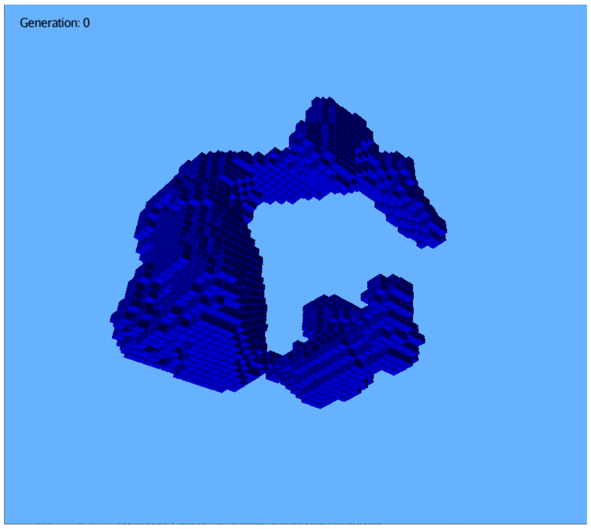


Figure 7.6.1

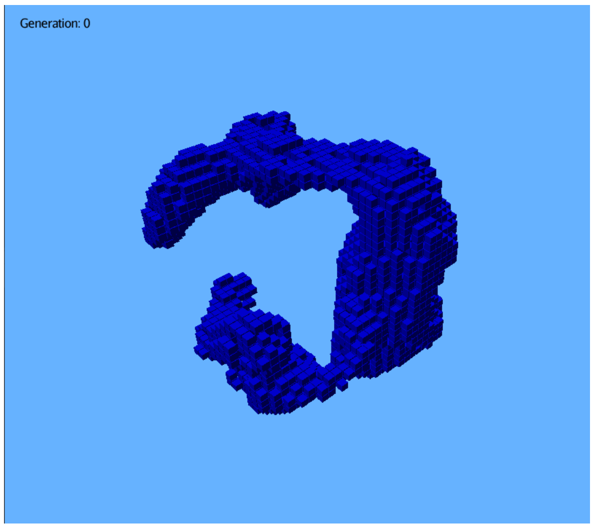


Figure 7.6.2

This pattern was also generated using Rule (8 15 11 13). This pattern has thicker strands connecting the bubble-like structures. This pattern is also very different to the patterns represented by Figures 7.5.1 and 7.5.2 which were generated using the same Ruleset, in that the strands do not form loop structures and that the pattern has a smoother surface in comparison. Figure 7.6.2 is an alternate view of Figure 7.6.1 from the opposite side.

The Snakes

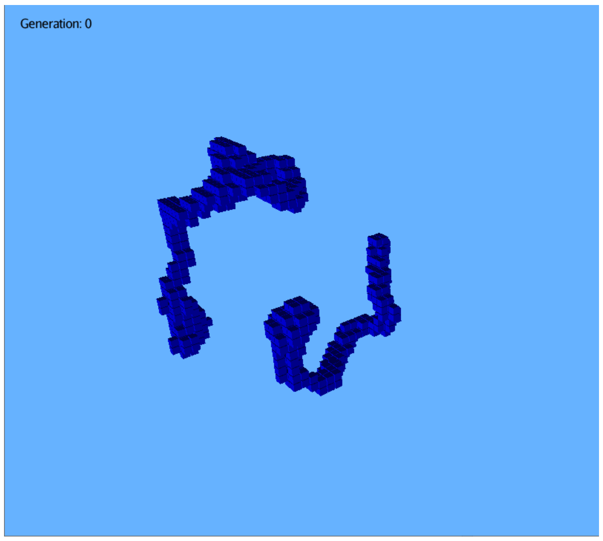


Figure 7.7.1

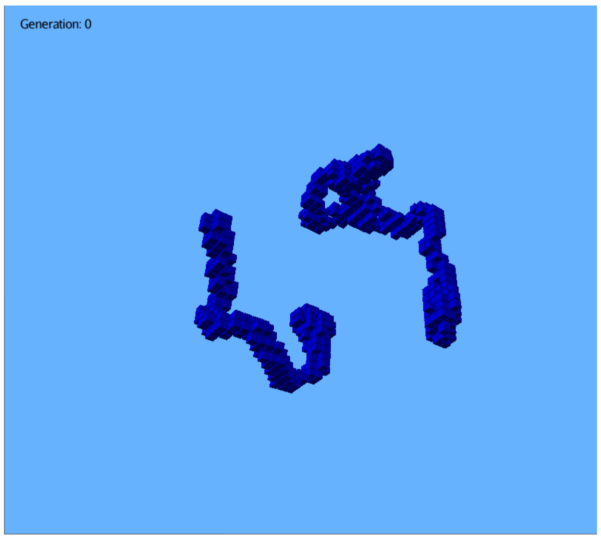


Figure 7.7.2

Rule (8 15 11 14) forms a static pattern. This rule forms snake-like structures, very similar to the connecting strands that arise from Rule (8 15 11 13) shown by Figure 7.5.1 and Figure 7.5.2. Earlier experimentation with the birth rule range has consistently showed that as the range increases, the more opportunities for life to develop increases and therefore it would be more likely for such rule to see explosive growth. With Rule (8 15 11 14), we should expect to see a similar structure to Rule (8 15 11 13) with a higher cell density, however the opposite happens, the large bubble-like structures tend to die out leaving the snake-like structures remaining. Figure 7.7.2 is an alternate view.

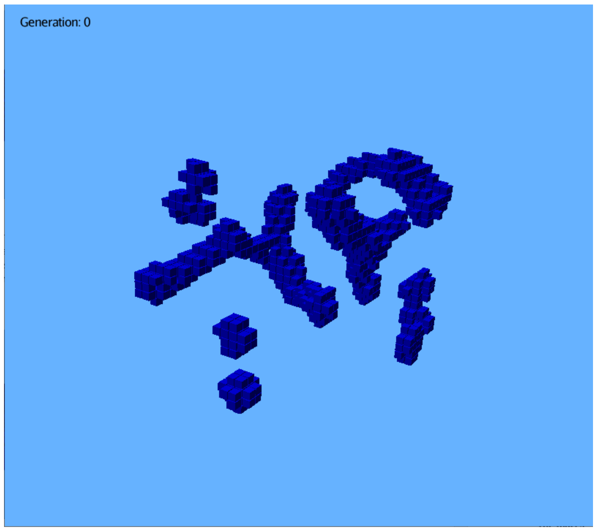


Figure 7.8.1

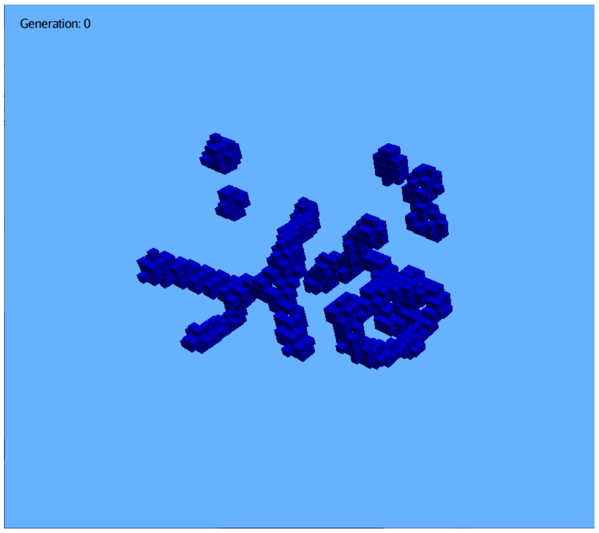


Figure 7.8.2

Rule (8 12 12 17) shares similarities with Rule (8 15 11 14). The notable difference between the rules is the appearance of small isolated static bubbles of cells surrounding the snake-like structures.

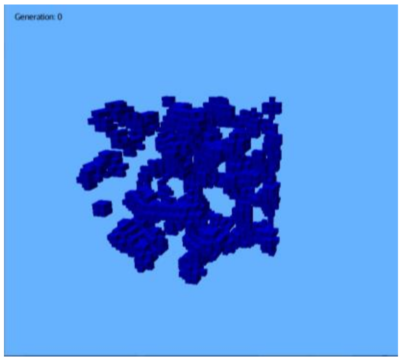


Figure 7.9.1

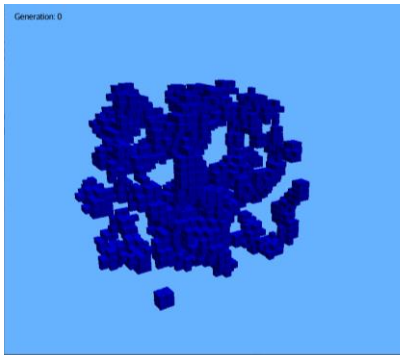


Figure 7.9.2

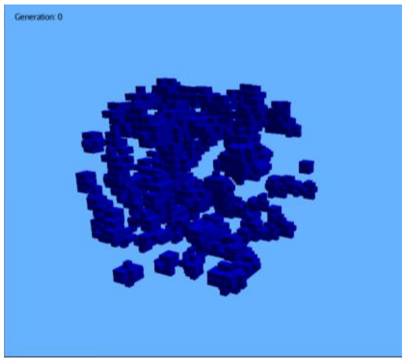


Figure 7.9.3

Rule (7 13 13 15) forms a pattern that appears to be a denser version of the pattern shown in Figure 7.8.1. On closer inspection, all of the snake-like structures are connected into a single structure with static bubbles placed around the structure.

The Mazes

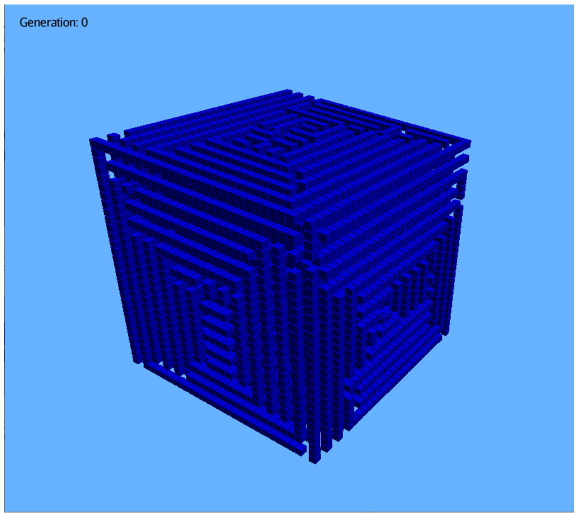


Figure 7.10.1

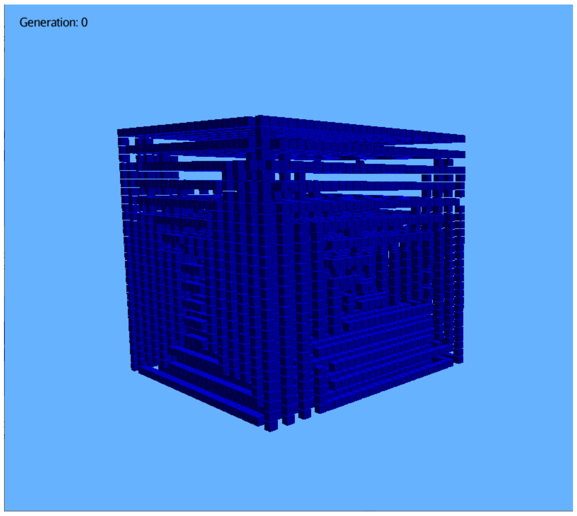


Figure 7.10.2

Rule (0 3 2 2) results in a static maze-like structure as shown by Figure 7.10.1. This Ruleset creates lines of cells with a thickness of 1 cell, separated by at least 1 cell. This creates a maze-like shape on each side of board’s outer edge. In addition to each line of cells being separated by a 1 cell gap, each layer of cube is also separated by a 1 cell gap, with each layer forming another maze. Figure 7.10.2 clearly shows the distinct layers of the cube.

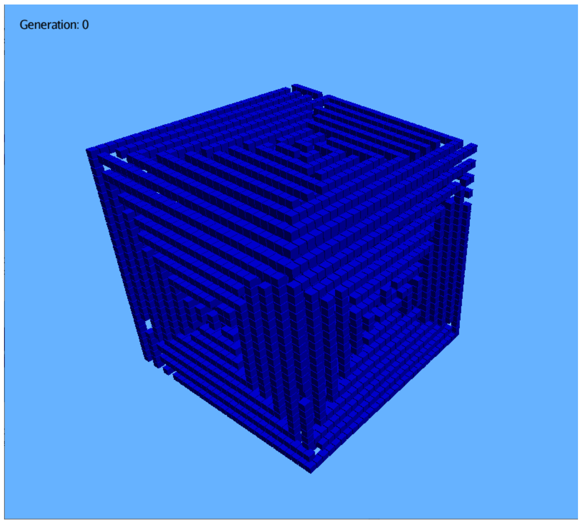


Figure 7.11.1

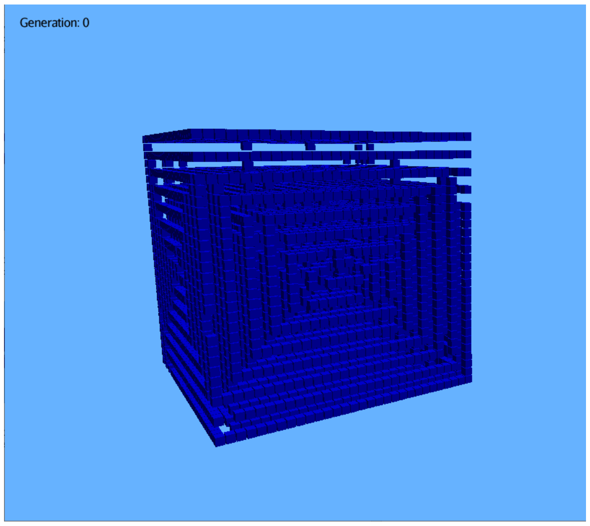


Figure 7.11.2

Rule (1 3 2 2) is like Rule (0 3 2 2). This rule results in simpler static maze on the surface of the cube, however as shown in the side view (Figure 7.11.2), the cube becomes denser closer to the centre.

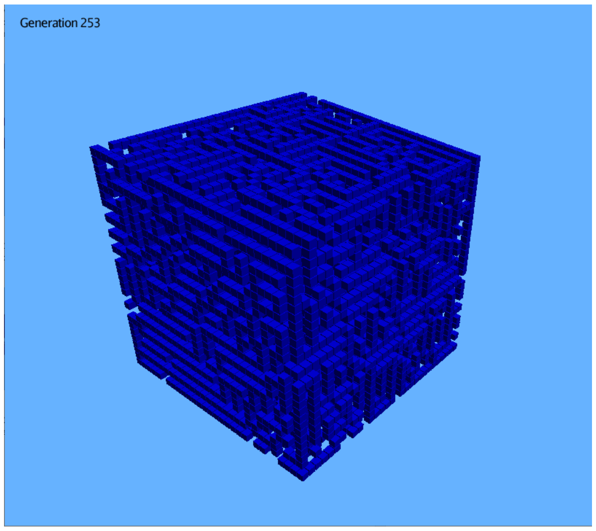


Figure 7.12.1

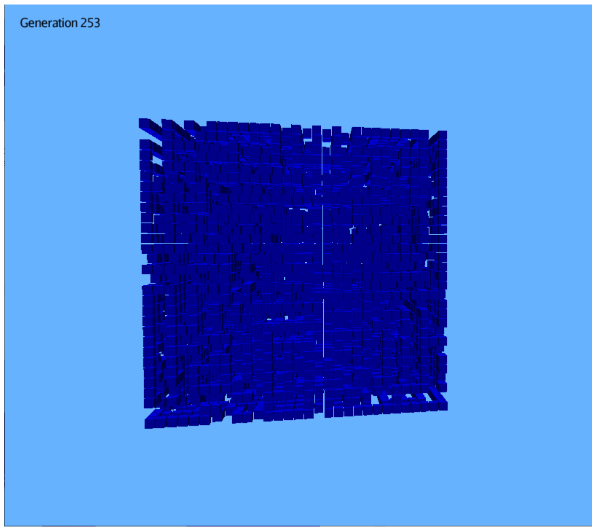


Figure 7.12.2

Rule (0 5 3 3) also forms a static maze-like structure. This pattern is more complex compared to previous maze-like structures. This pattern also has a uniform density with no clear gaps between cube layers.

The Unknowns

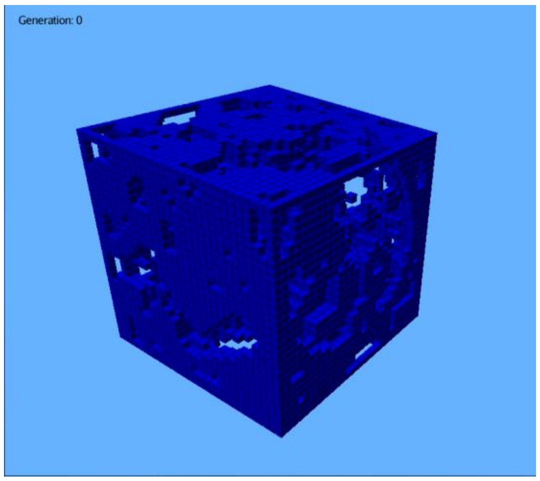


Figure 7.13.1

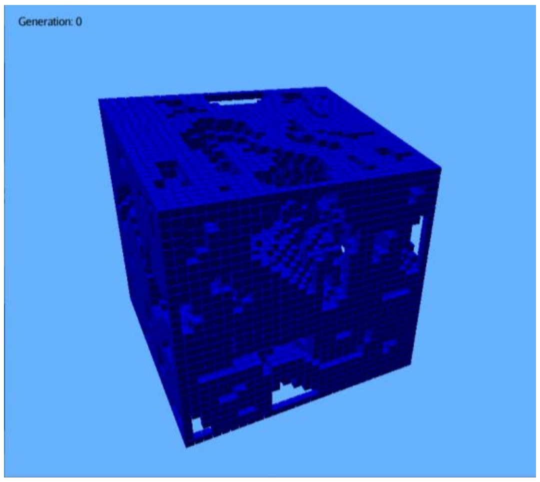


Figure 7.13.2

Rule (12 12 1 11) forms what could be described as a cross section of a cave system. The edges and corners of the cube is filled with cells giving the appearance of a frame. The cube itself is filled with a dense structure of cells. The pattern shown in Figure 7.13.1 is the result after 100 generations from random starting pattern. The pattern will continue to change with subsequent generations, never filling the cube further nor falling into an oscillating pattern. When classifying this rule for the purposes of the rule table, this rule was labelled as Explodes. While this pattern does not grow when bounded by the board, when unbounded it would grow unrestricted.

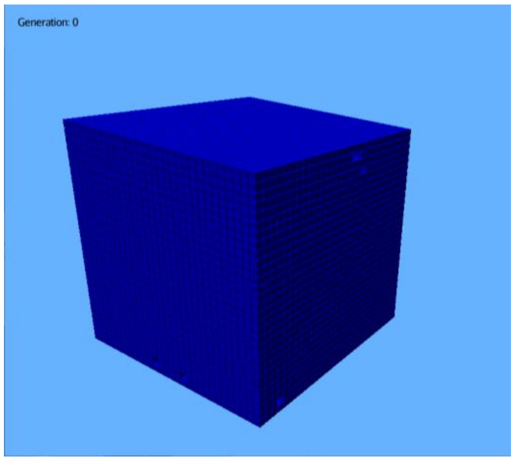


Figure 7.14.1

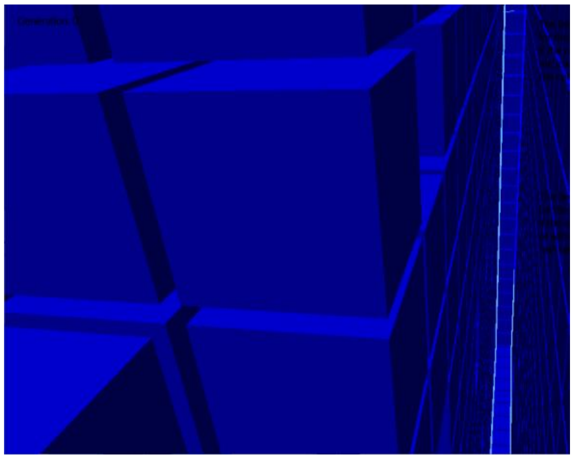
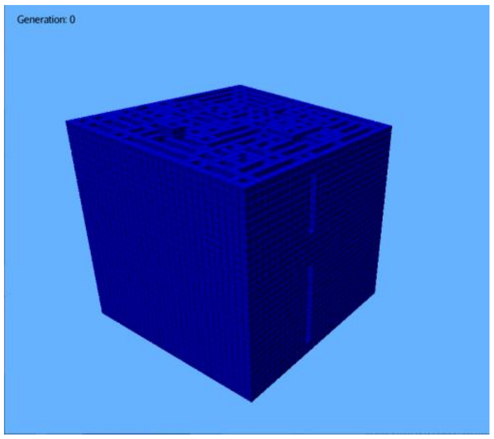


Figure 7.14.2

Rule (0 17 1 8) forms what appears to be a solid cube (Figure 7.14.1), however the outer layer of the cube forms a shell-like structure surrounding a smaller cube with a 1 cell gap between the two. (Figure 7.14.2). The inner cube also has a solid outer shell of cells, the interior of this inner cube is chaotic yet static.



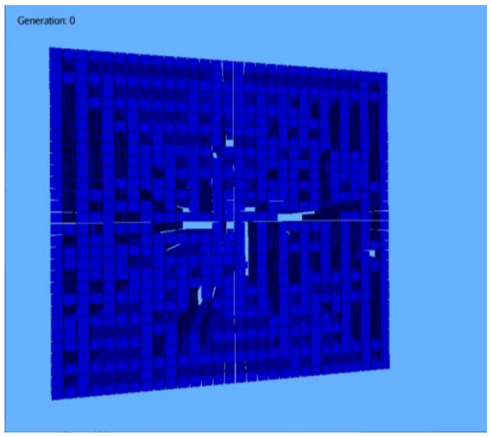
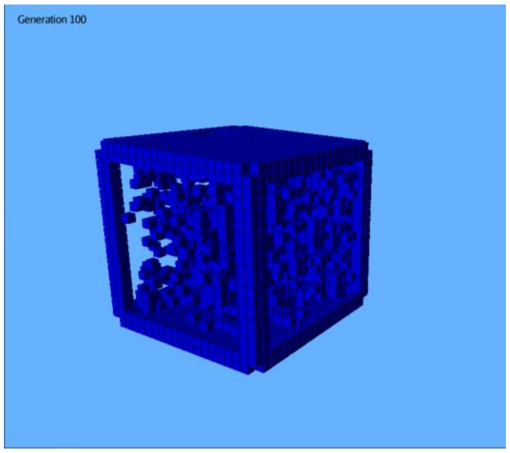


Figure 7.15.1 Figure 7.15.2

Rule (3 17 2 5) produces static patterns with no apparent unique properties. Early on through development, this pattern (Figure 7.15.1) was discovered, the pattern is static and has a few noteworthy properties. Four sides of the cube are (almost) completely covered with cells. The remaining sides have similarities with the pattern shown by Figure 7.12.1 with its maze-like structure. A top down view of one of these sides shows that this pattern is consistent throughout the structure, the opposite side shows an almost identical pattern. While reproduction of this specific structure with this set of rules has proven fruitless, it serves as proof that the potential for new, undiscovered patterns and structures is higher than previously explored.



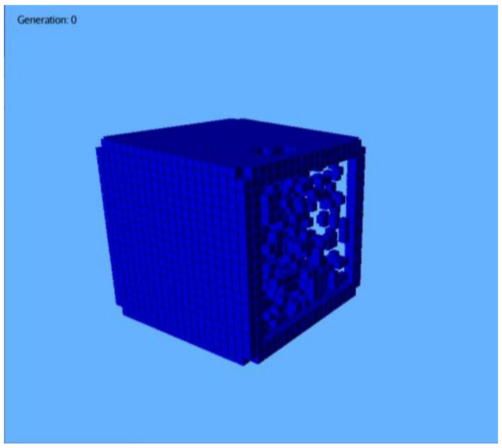


Figure 7.16.1 Figure 7.16.2

Rule (8 16 2 2) explodes immediately and like Rule (3 17 2 5) has no apparent unique properties. This pattern (Figure 7.16.1) exists in a semi-static state with this Ruleset. The outer frame of the cube is static and three of the cube's sides are filled in and static. The inside of the cube is a chaotic cloud of cells, that never settles down. The cloud does not interact with the frame of the cube. This pattern cannot be classified as it demonstrates properties of both explosive growth and static structure.

The Glider

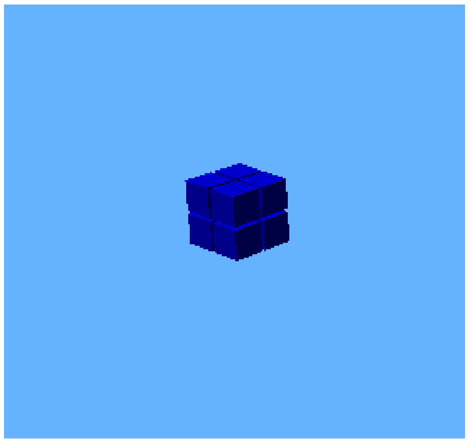


Figure 7.17.1

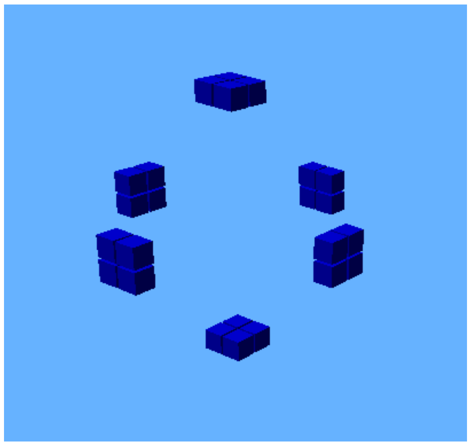


Figure 7.17.2

While searching for a 3D equivalent of the 2D static Block pattern. We found that when the initial pattern (Figure 7.17.1) is applied to Rule (14 16 4 4), a new pattern arises that is a strong candidate for a Glider. The 1x2x2 shape creates two copies either side of it, after 5 generations all but one of the copies cancel out in each direction leaving six patterns.

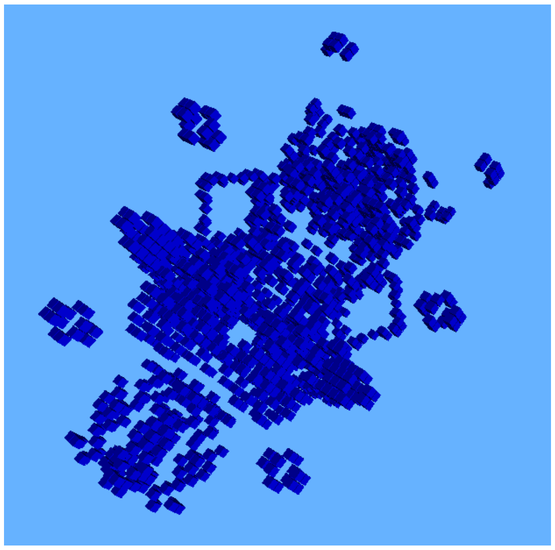


Figure 7.17.3

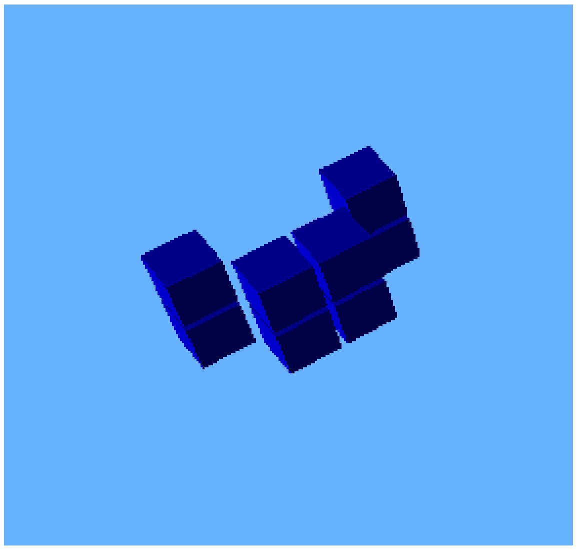


Figure 7.17.4

To test if this pattern is a true glider or if it grows without bound, we hand-crafted the pattern and ran it for 100 generation using the same rules. The result was that the pattern grows without bounds, however another pattern emerged in generation 57 (Figure 7.17.3).

The pattern that emerged (Figure 7.17.4) was observed moving out from the chaotic cloud of cells in the direction of the four forward cells. The pattern died upon reaching the board boundary. Each alive cell has less than 14 neighbours and therefore dies as per the Ruleset, however there are 8 dead cells that have exactly 4 cells and therefore will become alive in the next generation. The next generation of this pattern corresponds to pattern being shifted 1 cell in the direction of the 4 centre cells.

Since this pattern was spawned from a hand-crafted pattern it cannot be determined to be a naturally occurring pattern with this Ruleset. However, considering the highest number of alive neighbours any cell in this pattern could have is 5, any Ruleset where the death rules are both 6 at the minimum and the birth rules are both 4 (i.e. (>5 >5 4 4)) will be capable of supporting this Glider.

Referring to Figure 6, there are 78 rules where this glider is supported, all of which are classified as Explodes. This implies that either that this pattern does not occur naturally or that it can occur naturally and that these rules require a larger board and more extensive testing to confirm such.

Conclusion

Rule (4 5 6 6) and (5 6 5 5) both contain the same static patterns and oscillators, but (5 6 5 5) contains additional oscillators. Yet, neither have been observed to create gliders. Rule (14 16 4 4) and (6 6 4 4) have both been proven to support a glider, however neither of these rules have been observed creating a glider naturally.

Patterns have emerged from Rule (7 7 7 7) and (4 5 6 6) that can be considered equivalent to two of Conway's Patterns. Figure 7.1.7 is equivalent to a 2D "Block" and Figure 7.2.1 is the equivalent of a 2D "Blinker".

Of the rules tested, Rule (5 6 5 5) displays behaviour closest to that of Conway's Game of Life, with its diverse mix of static and oscillating patterns. However, its notable lack of an observed glider is evidence that a rule that exhibits closer behaviour to Conway's Game of Life exists.

In addition, given that Rulesets with upper and lower bounds greater than 18 were not explored, what this shows is that there is potential for further research into this field.

BCS Project Criteria & Self-Reflection

An ability to apply practical and analytical skills gained during the degree programme.

Over the past couple of years, I have gained an in-depth understanding of Conway's Game of Life. In my first year of the degree, I was introduced to the algorithm simulation and was tasked to write the program in the Haskell programming language. The very next year, one of my assignments was to implement a version of Conway's Game of Life in the C programming language, during this year I also created a 3D Arkanoid clone using the jMonkeyEngine SDK for one of my assignments. I believe this gives me a significant advantage in this project.

Innovation and/or Creativity, and Synthesis of information, ideas and practices to provide a quality solution together with an evaluation of that solution

As mentioned earlier, there hasn't been much experimentation with a 3D Conway's Game of life, other than Carter Bays early insight in 1987, to my knowledge. I feel that there is a lot of research potential that still hasn't been explored in-depth. One common factor among all versions of Conway's Game of life and Cellular automata is that the rules for life tend to be fixed and are only modified prior to the simulation commencing. This is something that this could be looked into as an advancement of the concept, the idea of Dynamic Rulesets. In addition to seeing what patterns emerge from different Rulesets, how do patterns of one set interact with a different set of rules? Further, what if the rules were changed constantly throughout the simulation at specified points? For example, oscillating between 2 sets of rules every other iteration.

That your project meets a real need in a wider context

While the project itself doesn't meet a real need, the data that this project generates through explored rules and patterns, there are academics in this field that may find the data interesting and may conclude there may be worth in revisiting the applications of a 3D Game of Life.

An ability to self-manage a significant piece of work

To ensure that I stayed on track, I had created a Gantt chart outlining my progress throughout the project. I used Github to keep my work backed up and synchronized, ensuring I stayed organised. In addition, I had regular progress checks with my supervisor for this project to ensure that I had sufficient time to complete my goals and deciding which of said goals are achievable.

Critical self-evaluation of the process

The main criticism I have about the process is that I felt at times I was too ambitious about my ideas for this project. One idea as previously mentioned was the implementation of what I like to call “Dynamic Rulesets”. In its entirety, this would have allowed multiple Rulesets to applied to the simulation periodically, the potential for new patterns and lifeforms to emerge is huge and could possibly have lead to a new branch of research in the field of Cellular Automata. However, while this idea showed great potential, it was never implemented into the final program as it wasn’t within the scope of this project and the development of such would have caused delays. In the end, this idea was scaled back to just allowing the user to change the Ruleset on-the-fly. This idea was far more manageable and greatly assisted in the analysis of patterns, mainly what rules would cause a pattern to die/explode.

References

- *Bialynicki-Birula, Iwo; Bialynicka-Birula, Iwona (2004) p. 9. Modelling Reality: How Computers Mirror Life.*
- *Wolfram, Stephen (2002) p. 876-877. A New Kind of Science.*
- *Bays, Carter (1987) p. 373–400. Candidates for the Game of Life in Three Dimensions.*
- *Bays, Carter (2006) p. 381–386. A Note About the Discovery of Many New Rules for the Game of Three-Dimensional Life.*
- *Conway, John (1999) Unknown.*

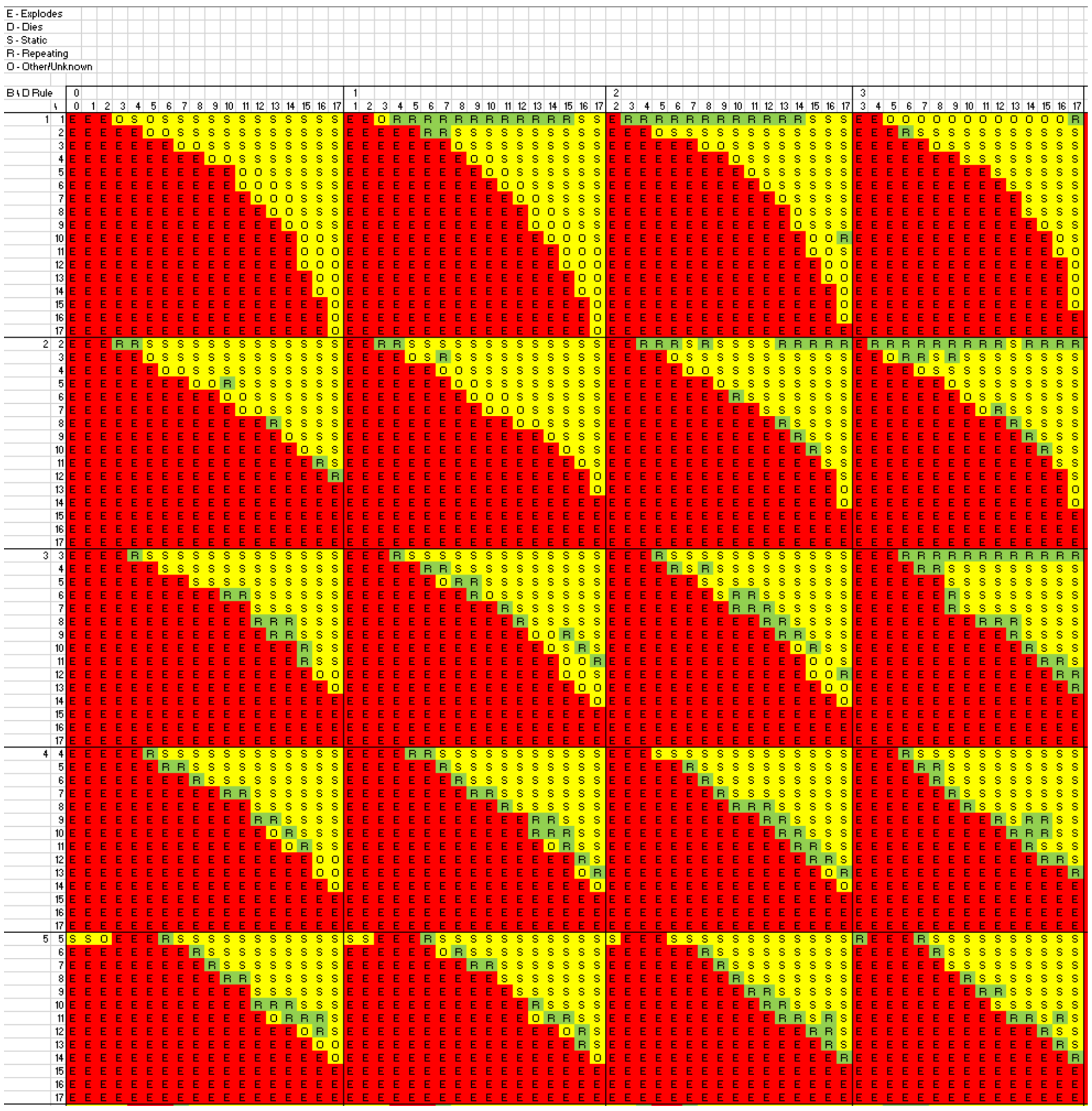
Appendices

A note on Figures 7.15.1 & 7.16.1:

Due to these patterns being found early on in development and the failure to reproduce a similar pattern with these rules, the origins of these patterns cannot be accurately determined. This pattern may have emerged naturally from another Ruleset or as part of data file corruption.

Complete Table of Results (Figure 6)

Table divided into sections.



E - Explodes
D - Dies
S - Static
R - Repeating
O - Other/Unknown

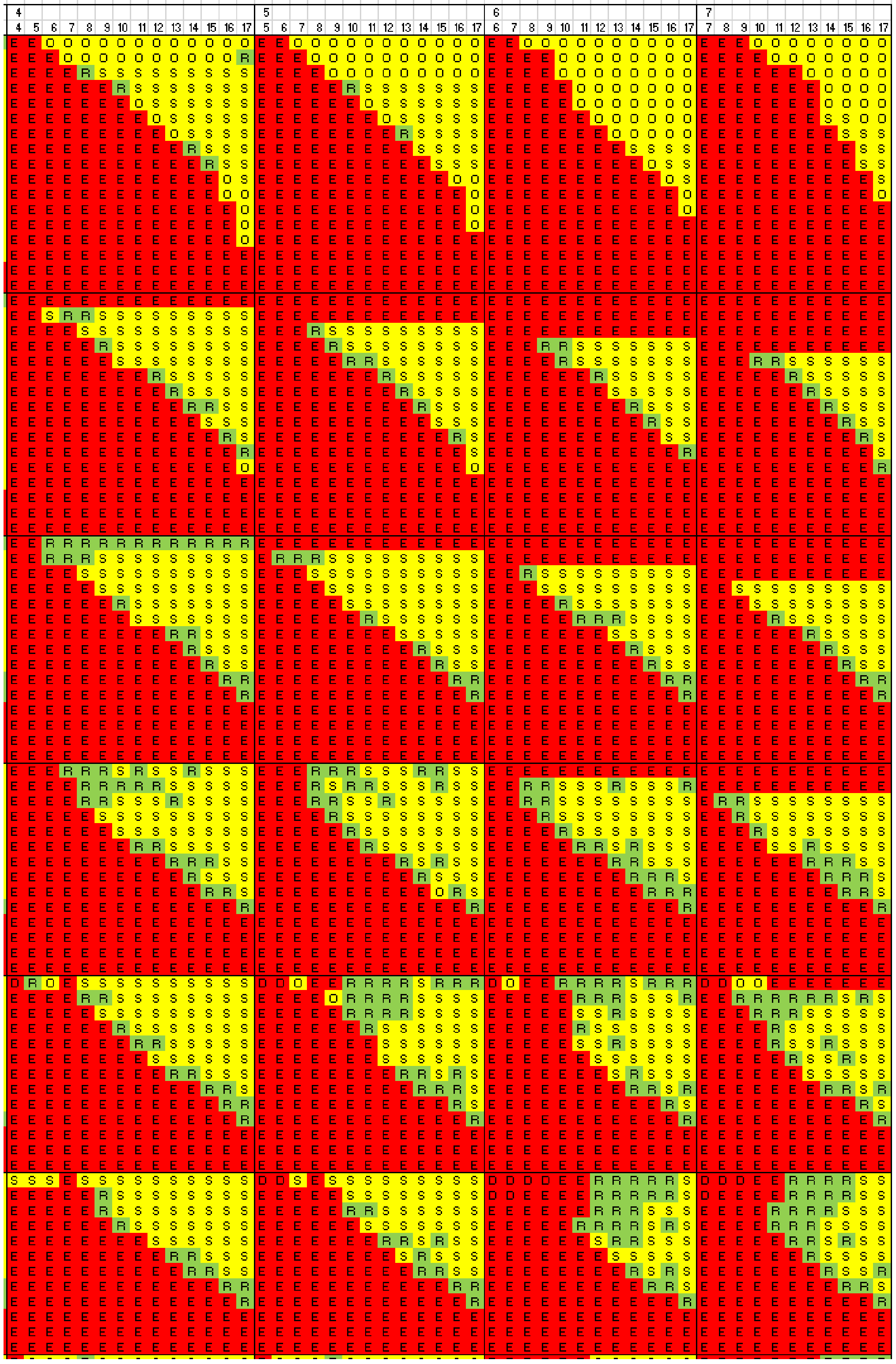
B \ D Rule		0																	
	\	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	E	E	E	O	S	O	S	S	S	S	S	S	S	S	S	S	S	S
	2	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	E	O	O	O	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
2	2	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	E	O	S	S	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	O	O	R	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
3	3	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
4	4	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
5	5	S	S	O	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	R	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E

B \ D Rule		1																
	\	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	E	E	O	R	R	R	R	R	R	R	R	R	R	R	R	S	S
	2	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	E	E	E	O	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
2	2	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	O	S	R	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	O	O	O	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	O	O	O	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	O	O	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
3	3	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	O	R	R	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	R	O	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	O	O	R	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S	R	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	R
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
4	4	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
5	5	S	S	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	O	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E

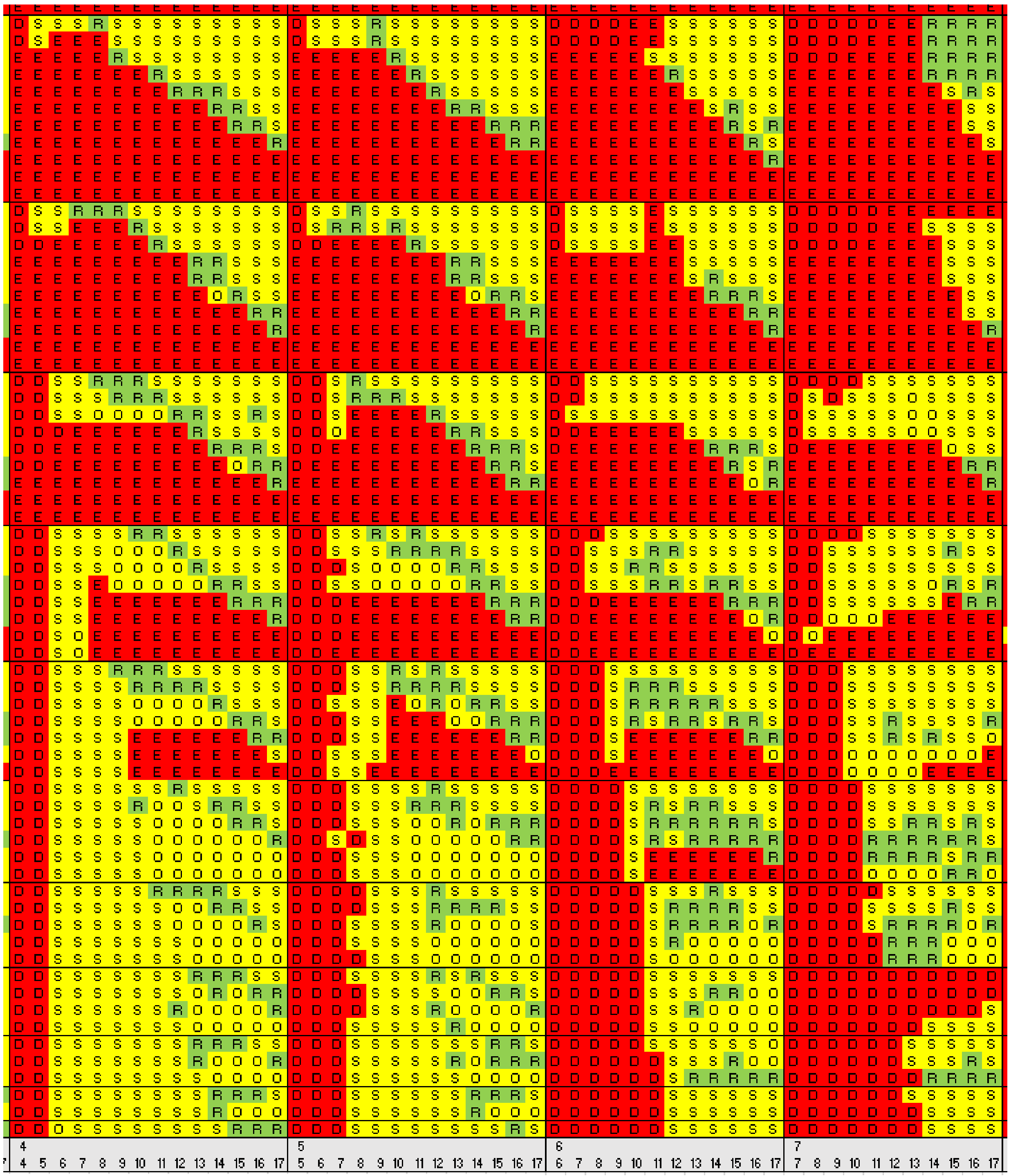
B \ D Rule		2															
	\	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	E	R	R	R	R	R	R	R	R	R	R	R	R	S	S	S
	2	E	E	E	O	S	S	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	E	E	O	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	E	O	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	O	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	O	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	R
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
2	2	E	E	R	R	R	S	R	S	S	S	S	R	R	R	R	R
	3	E	E	E	E	O	S	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	O	O	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	O	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	E	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
3	3	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	R	S	R	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	S	R	R	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	R	R	R	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	R
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
4	4	E	E	E	S	S	S	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	R	R	R	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
5	5	S	E	E	E	S	S	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	R	R	S	R	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E

B \ D Rule		3														
	\	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	E	E	O	O	O	O	O	O	O	O	O	O	O	O	R
	2	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S
	3	E	E	E	E	E	O	S	S	S	S	S	S	S	S	S
	4	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	O	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	E	O	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
2	2	E	R	R	R	R	R	R	R	R	R	S	R	R	R	R
	3	E	E	O	R	R	S	R	S	S	S	S	S	S	S	S
	4	E	E	E	E	O	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	O	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	E	O	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	E	E	O	R	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
3	3	E	E	E	R	R	R	R	R	R	R	R	R	R	R	R
	4	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	R	R	R	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
4	4	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S
	5	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	R	S	R	R	S	S
	10	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
5	5	R	E	E	E	R	S	S	S	S	S	S	S	S	S	S
	6	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	7	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	R	R	S	R	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E

6	6	S	S	S	S	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	S	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	S	S	S	S	S	S	S	S	S	S
	7	S	S	O	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	O	E	E	E	E	E	O	R	R	S	S	S	S	S	S	S	S	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S
	8	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S
	9	E	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	R	O	R	S	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
7	7	S	S	S	S	R	R	E	O	E	S	S	S	S	S	S	S	S	S	S	S	S	S	R	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	E	E	E	S	S	S	S	S	S	S	S	S	D	S	R	R	R	S	S	S	S	S	S	S	S	S	S
	8	S	S	S	S	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	E	E	E	E	R	S	S	S	S	S	S	S	S	D	S	E	E	E	E	R	S	S	S	S	S	S	S	S
	9	S	S	S	O	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S	S	O	E	E	E	E	E	E	R	S	S	S	S	S	S	S	D	O	E	E	E	E	E	E	S	S	S	S	S	S	S	S	D	E	E	E	E	E	E	R	S	S	S	S	S	S	S
	10	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	O	S	S	S	S	S	S	E	E	E	E	E	E	E	E	R	R	S	S	S	S	S	S	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S
	11	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S	E	E	E	E	E	E	E	E	E	R	R	S	S	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	R	S	S	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
8	8	S	S	S	S	S	R	R	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	R	R	E	O	S	S	S	S	S	S	S	S	S	S	S	S	S	R	E	E	R	S	S	S	S	S	S	S	S	D	S	S	R	R	R	S	S	S	S	S	S	S	S	S
	9	S	S	S	S	S	S	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S	S	S	O	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	E	E	E	E	R	S	S	S	S	S	S	S	D	S	S	R	E	E	E	R	S	S	S	S	S	S	S
	10	S	S	S	S	S	O	E	E	E	E	E	O	S	S	S	S	S	S	S	S	S	S	O	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	E	E	E	E	E	R	S	S	S	S	S	S	D	S	S	E	E	E	E	E	R	R	S	S	S	S	S
	11	S	S	S	S	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	O	E	E	E	E	E	E	E	O	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	R	R	S	S	S	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	R	R	R	S	S	E	E	E	E	E	E	E	E	E	E	R	R	S	S	S
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	S	E	E	E	E	E	E	E	E	E	E	E	O	R	S	S
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	R	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
9	9	S	S	S	S	S	S	R	R	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	O	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R	O	S	S	S	S	S	S	S	S	D	S	S	S	R	R	R	R	S	S	S	S	S	S	S
	10	S	S	S	S	S	S	R	O	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	E	E	O	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R	O	R	R	S	S	S	S	S	S	D	S	S	S	R	O	R	R	R	S	S	S	S	S	S
	11	S	S	S	S	S	S	S	O	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S	S	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	O	E	O	O	O	R	S	S	S	S	S	D	S	S	S	E	O	O	O	O	R	S	S	S	S	S
	12	S	S	S	S	S	S	O	O	E	E	E	E	E	E	R	S	S	S	S	S	S	S	S	O	E	E	E	E	E	O	R	R	S	S	S	S	S	S	S	S	E	E	E	E	E	E	R	S	S	S	S	D	S	S	S	E	E	E	E	E	E	R	R	S	S	S
	13	S	S	S	S	S	S	E	E	E	E	E	E	E	E	O	R	S	S	S	S	S	S	O	E	E	E	E	E	E	E	E	O	O	S	S	S	S	S	S	S	E	E	E	E	E	E	E	O	R	S	S	D	S	S	E	E	E	E	E	E	E	E	E	R	S	S
	14	O	O	O	O	O	E	E	E	E	E	E	E	E	E	E	E	R	S	O	O	O	O	E	E	E	E	E	E	E	E	E	E	E	R	R	D	S	S	O	O	E	E	E	E	E	E	E	E	O	R	R	D	S	E	E	E	E	E	E	E	E	E	E	R	R	R
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
10	10	S	S	S	S	S	S	S	S	R	O	O	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	O	O	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	S	S	S	S	S	S	S	D	S	S	S	S	R	R	R	R	S	S	S	S	S	S
	11	S	S	S	S	S	S	S	S	O	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	R	R	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	S	R	O	O	S	S	S	S	S	S	D	S	S	S	S	S	R	R	R	R	S	S	S	S	S
	12	S	S	S	S	S	S	S	S	E	E	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	E	O	O	O	R	S	S	S	S	D	S	S	S	S	S	O	O	O	O	R	S	S	S	S
	13	S	S	S	S	S	S	S	S	E	E	E	E	E	E	O	R	S	S	S	S	S	S	S	S	S	E	E	E	E	E	E	R	R	S	S	S	S	S	S	S	S	E	E	O	O	O	O	O	R	R	S	D	S	S	S	S	E	O	O	O	O	O	R	R	S	R
	14	S	S	S	S	S	S	S	O	E	E	E	E	E	E	E	O	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	O	R	R	S	S	S	S	S	S	E	E	E	E	E	E	E	R	R	R	D	S	S	S	S	E	O	O	O	O	O	O	R	R	R
	15	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	R	D	S	S	S	S	E	E	E	E	E	E	E	E	E	R
	16	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	O	E	E	E	E	E	E	E	E	E	E	D	S	S	S	S	E	E	E	E	E	E	E	E	E	E
	17	S	S	S	S	O	O	E	E	E	E	E	E	E	E	E	E	E	E	S	S	O	O	O	O	E	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	O	E	E	E	E	E	E	E	E	E	E	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E
11	11	S	S	S	S	S	S	S	S	R	R	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	E	E	R	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	S	S	S	S	S	S	D	S	S	S	S	R	R	R	R	R	S	S	S	S	S
	12	S	S	S	S	S	S	S	S	S	R	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	R	R	E	E	E	R	S	S	S	S	S	S	S	S	S	S	S	R	R	O	O	R	S	S	S	S	D	S	S	S	S	S	R	R	R	R	R	S	S	S	S
	13	S	S	S	S	S	S	S	S	S	O	E	E	E	E	R	R	S	S	S	S	S	S	S	S	S	S	O	E	E	E	E	R	R	S	S	S	S	S	S	S	S	R	S	O	O	R	O	R	S	S	S	D	S	S	S	S	S	R	O	O	O	O	R	R	S	S
	14	S	S	S	S	S	S	S	S	S	E	E	E	E	E	E	O	R	S	S	S	S	S	S	S	S	S	E	E	E	E	E	E	O	R	R	S	S	S	S	S	S	S	S	O	O	O	O	O	R	R	R	D	S	S	S	S	S	S	O	O	O	O	O	R	R	R
	15	S	S	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	R	S	S	S	S	S	S	S	O	E	E	E	E	E	E	R	R	S	S	S	S	S	S	E	E	E	E	E	E	E	R	R
	16	S	S	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	O	E	E	E	E	E	E	E	O	D	S	S	S	S	S	E	E	E	E	E	E	E	E	O
	17	S	S	S	S	S	S	S	S	O	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	S	S	S	S	S	S	S	O	E	E	E	E	E	E	E	E	D	S	S	S	S	S	E	E	E	E	E	E	E	E	E
12	12	S	S	S	S	S	S	S	S	R	R	R	E	E	E	E	E	S	S	S	S	S	S	S	S	S	S	R	R	E	R	E	E	E	S	S	S	S	S	S	S	S	S	S	R	O	O	R	S	S	S	S	S	S	S	S	S	S	S	R	R	R	R	S	S	S	S
	13	S	S	S	S	S	S	S	S	S	R	R	O	E	E	E	E	S	S	S	S	S	S	S	S	S	S	R	O	O	E	E	E	E	E	O	S	S	S	S	S	S	S	S	R	O	O	R	R	S	S	S	D	S	S	S	S	S	S	R	R	R	R	R	S	S	S
	14	S	S	S	S	S	S	S	S	S	S	S	O	E	E	E	E	E	O	S	S	S	S	S	S	S	S	S	O	O	E	E	E	E	E	E	S	S	S	S	S	S	S	S	R	O	O	O	O	R	R	S	D	S	S	S	S	S	S	R	O	O	O	O	R	R	S
	15	S	S	S	S	S	S	S	S	S	S	S	O	O	E	E	E	E	E	S	S	S	S	S	S	S	S	S	O	O	O	E	E	E	E	E	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	R	D	S	S	S	S	S	S	S	O	O	O	O	O	O	R
	16	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O
	17	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O	S	S	S	S	S	S	S	S	O	O	O	O	O	O	O
13	13	S	S	S	S	S	S	S	S	S	S	R	R	R	R	R	R	R	S	S	S	S	S	S	S	S	S	S	S	R	R	R	R	R	R	S	S	S	S	S	S	S	S	S	S	R	R	R	S	S	S	S	D	D	S	S	S	S	S	R	R	R	R	R	S	S	S
	14	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	R	O	O	O	R	S	S	D	S	S	S	S	S	S	S	R	O	R	O	R	S	S
	15	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	O	O	O	O	R	S	D	S	S	S	S	S	S	S	R	O	O	O	O	R	R
	16	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	R	D	S	S	S	S	S	S	S	S	O	O	O	O	O	O
	17	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	O	D	S	S	S	S	S	S	S	S	O	O	O	O	O	O
14	14	S	S	S	S	S	S	S	S	S	S	S	R	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	D	S	S	S	S	S	S	S	S	S	R	R	R	R	S	S	D	S	S	S	S	S	S	S	R	R	R	R	R	S	S
	15	S	S	S	S	S	S	S	S	S	S	S	R	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	R	O	O	O	O	R	S	S	S	S	S	S	S	S	S	R	O	O	O	O	R	S
	16	S	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O
	17	S	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	S	S	S	S	S	S	S	S	S	R	O	O	O	O	O
15	15	S	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	R	S	S	S	S	S	S	S	S	S	S	S	S	S	O	O	R	R	R	S	S	S	S	S	S	S	S	S	S	S	R	R	R	R	R	D	D	S	S	S	S	S	S	S	S	R	R	R	R	S
	16	S	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	S	S	S	S	S	S	S	S	S	S	R	R	O	O	O	O	D	S	S	S	S	S	S	S	S	S	R	O	O	O	O
	17	S	S	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	D	S	S	S	S	S	S	S	S	S	S	O	O	O	O
16	16	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R	O	S	S	S	S	S	S	S	S	S	S	S	S	S	S	O	R	R	R	D	D	S	S	S	S	S	S	S	S	S	S	S	R	R	S	D	D	S	S	S	S	S	S	S	S	S	S	S	R	R
	17	S	S	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	S	S	S	S	S	S	S	S	S	S	S	S	S	O	O	O	O	D	S	S	S	S	S	S	S	S	S	S	S	R	O	O	O	D	D	S	S	S	S	S	S	S	S	S	R	R	O	O
17	17	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R	S	R	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R	R	R	D	S	S	S	S	S	S	S	S	S	S	S	S	S	R	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	R
		0																		1																	2																3														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



(Birth rules from (1 1) to (6 17))



(Birth rules from (7 7) to (17 17))

8										9									10								11							12						13					14				15			16		17		
8	9	10	11	12	13	14	15	16	17	9	10	11	12	13	14	15	16	17	10	11	12	13	14	15	16	17	11	12	13	14	15	16	17	12	13	14	15	16	17	13	14	15	16	17	14	15	16	17	15	16	17	16	17	17		
E	E	E	O	O	O	O	O	O	O	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	1	1
E	E	E	E	O	O	O	O	O	O	E	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		2
E	E	E	E	E	O	O	O	O	O	E	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		3
E	E	E	E	E	E	O	O	O	O	E	E	E	E	E	E	O	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		4
E	E	E	E	E	E	O	O	O	O	E	E	E	E	E	E	O	O	O	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		5
E	E	E	E	E	E	O	O	O	O	E	E	E	E	E	E	O	O	O	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		6
E	E	E	E	E	E	E	S	S	O	E	E	E	E	E	E	O	O	O	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		7
E	E	E	E	E	E	E	O	S	S	E	E	E	E	E	E	E	S	S	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	E	E	E	E	E	S	S	E	E	E	E	E	E	E	S	S	E	E	E	E	E	E	O	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	2	2
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		3
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		4
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		5
E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		6
E	E	E	R	S	S	S	S	S	S	E	E	E	E	E	E	R	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		7
E	E	E	E	E	R	S	S	S	S	E	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	E	E	E	S	S	S	S	E	E	E	E	E	R	S	S	S	E	E	E	E	E	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	E	E	E	R	S	S	E	E	E	E	E	E	S	S	S	E	E	E	E	E	E	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	E	E	E	S	R	E	E	E	E	E	E	E	S	S	E	E	E	E	E	E	O	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	3	3
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		4
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		5
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		6
E	E	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		7
E	E	E	R	S	S	S	S	S	S	E	E	E	R	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	E	E	R	R	S	S	S	E	E	E	E	R	R	S	S	S	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	E	E	R	S	S	S	E	E	E	E	E	R	S	S	S	E	E	E	E	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	E	E	S	S	S	E	E	E	E	E	E	S	S	S	E	E	E	E	E	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	E	E	S	R	E	E	E	E	E	E	E	E	S	E	E	E	E	E	E	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	4	4
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		5
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		6
E	S	S	R	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		7
E	E	R	S	S	S	S	S	S	S	E	R	S	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	S	S	S	S	S	S	S	E	E	R	S	S	S	S	S	S	E	R	R	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	E	R	S	R	S	S	E	E	E	E	R	S	R	S	S	E	E	E	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	E	R	R	R	S	E	E	E	E	E	R	R	S	S	E	E	E	E	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	E	R	S	R	E	E	E	E	E	E	R	R	R	E	E	E	E	E	R	S	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	O	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	R	D	R	D	D	D	D	D	D	5	5
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		6
E	E	R	R	R	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		7
E	E	E	R	S	S	S	S	S	S	E	E	S	R	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	S	S	S	R	S	S	S	E	E	R	S	S	S	S	S	S	E	R	S	S	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	S	S	S	S	S	S	E	E	E	R	S	S	S	S	S	E	E	S	R	R	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	S	R	S	S	S	E	E	E	E	R	R	R	S	S	E	E	E	S	R	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	R	R	R	S	E	E	E	E	E	R	R	R	S	E	E	E	E	R	R	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	E	R	S	E	E	E	E	E	E	E	R	R	E	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17
D	D	D	D	D	D	D	D	D	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	6	6
E	E	E	E	R	R	R	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	O	D	D	O	O	O	D	D	D	D	D	D	D	D	D	D		7
E	E	E	R	R	R	S	R	S	S	E	E	R	R	R	R	R	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		8
E	E	E	R	R	R	R	S	S	S	E	E	R	O	R	R	S	S	S	E	E	R	S	R	S	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		9
E	E	E	E	S	R	S	R	S	S	E	E	E	R	R	S	S	S	S	E	E	R	R	S	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		10
E	E	E	E	E	S	S	S	S	R	E	E	E	E	S	S	S	S	S	E	E	E	R	S	S	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		11
E	E	E	E	E	E	R	S	S	S	E	E	E	E	E	R	S	S	R	E	E	E	E	R	R	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		12
E	E	E	E	E	E	E	R	R	S	E	E	E	E	E	E	R	R	S	E	E	E	E	E	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		13
E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		14
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		15
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		16
E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E		17

7	7	D	D	D	D	E	E	R	R	R	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	8	D	D	D	E	E	E	R	R	R	R	D	O	E	E	E	R	R	R	R	D	D	D	O	O	E	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	9	D	D	E	E	E	E	R	R	R	R	D	E	E	E	E	R	R	R	R	E	E	E	E	R	R	R	R	O	O	O	E	E	E	E	O	O	O	E	E	E	O	O	O	E	E	O	O	O	E	O	E	O	O	O	D
	10	E	E	E	E	E	E	S	R	R	R	E	E	E	E	E	S	R	R	R	E	E	E	E	S	R	R	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	11	E	E	E	E	E	E	E	S	R	S	E	E	E	E	E	E	R	S	S	E	E	E	E	E	R	R	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	12	E	E	E	E	E	E	E	S	S	S	E	E	E	E	E	E	S	R	S	E	E	E	E	E	S	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	13	E	E	E	E	E	E	E	E	S	R	E	E	E	E	E	E	E	S	S	E	E	E	E	E	E	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	14	E	E	E	E	E	E	E	E	S	R	E	E	E	E	E	E	E	E	S	E	E	E	E	E	E	S	S	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
8	8	D	D	D	D	D	D	D	E	E	R	D	D	D	D	D	D	D	E	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	9	D	D	D	D	D	E	E	E	E	R	D	D	D	D	E	E	E	E	R	D	D	D	E	E	E	O	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	10	D	D	D	D	E	E	E	E	E	R	D	D	D	E	E	E	E	E	E	D	D	E	E	E	E	O	R	D	O	E	E	E	E	E	D	O	O	O	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	11	D	D	D	D	E	E	E	E	E	R	D	D	D	E	E	E	E	E	E	D	D	E	E	E	E	E	R	D	O	E	E	E	E	E	O	E	E	E	E	E	O	O	E	E	E	E	E	E	E	E	E	E	O	E	D
	12	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	13	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	R	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	14	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
9	9	D	D	D	D	D	D	D	D	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	10	D	D	D	D	D	D	O	E	S	S	D	D	D	D	D	D	D	E	E	D	D	D	D	D	E	E	E	D	D	D	D	E	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	11	D	D	D	D	D	D	E	E	S	S	D	D	D	D	D	D	E	E	E	D	D	D	D	O	E	E	E	D	D	D	O	E	E	E	D	D	O	E	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	12	D	D	D	D	D	D	E	E	S	S	D	D	D	D	D	D	E	E	E	D	D	D	D	E	E	E	E	D	D	D	E	E	E	E	D	D	E	E	E	E	D	O	E	E	E	D	D	E	E	D	D	D	D	D	D
	13	D	D	D	D	D	D	E	E	E	R	D	D	D	D	D	D	E	E	E	D	D	D	D	E	E	E	E	D	D	D	E	E	E	E	D	D	E	E	E	E	D	E	E	E	E	O	E	E	E	O	E	E	D	E	D
	14	E	E	E	E	E	E	E	E	E	E	D	D	D	D	D	D	E	E	E	O	O	O	O	E	E	E	E	O	O	O	E	E	E	E	O	O	E	E	O	E	O	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	15	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	16	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
10	10	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	11	D	D	D	D	D	D	S	S	S	S	D	D	D	D	D	D	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	12	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	S	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	13	D	D	D	D	D	D	D	D	S	S	D	D	D	D	D	D	D	O	O	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	14	D	D	D	D	D	D	D	D	E	O	D	D	D	D	D	D	D	O	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	15	D	O	O	O	O	D	D	D	E	E	D	D	D	D	D	D	D	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	O	E	E	E	E	E	O	O	E	E	O	E	E	E	E	E	E	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	E
	17	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	E
11	11	D	D	D	D	D	D	D	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	12	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	13	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	14	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	15	D	D	D	D	D	S	S	S	S	S	D	D	D	D	D	D	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	D	D	S	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	S	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
12	12	D	D	D	D	D	D	D	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	13	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	14	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	15	D	D	D	D	D	D	D	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	D	D	D	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	D	S	R	R	R	R	R	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
13	13	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	14	D	D	D	D	D	D	D	D	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	15	D	D	D	D	D	D	D	D	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	D	D	D	D	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	D	D	S	R	R	R	R	R	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
14	14	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	15	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	D	D	D	D	D	D	S	D	D	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	D	D	D	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
15	15	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	16	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
16	16	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
	17	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
17	17	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
		8										9									10								11							12						13					14				15			16		17
		8	9	10	11	12	13	14	15	16	17	9	10	11	12	13	14	15	16	17	10	11	12	13	14	15	16	17	11	12	13	14	15	16	17	12	13	14	15	16	17	13	14	15	16	17	14	15	16	17	15	16	17	16	17	17