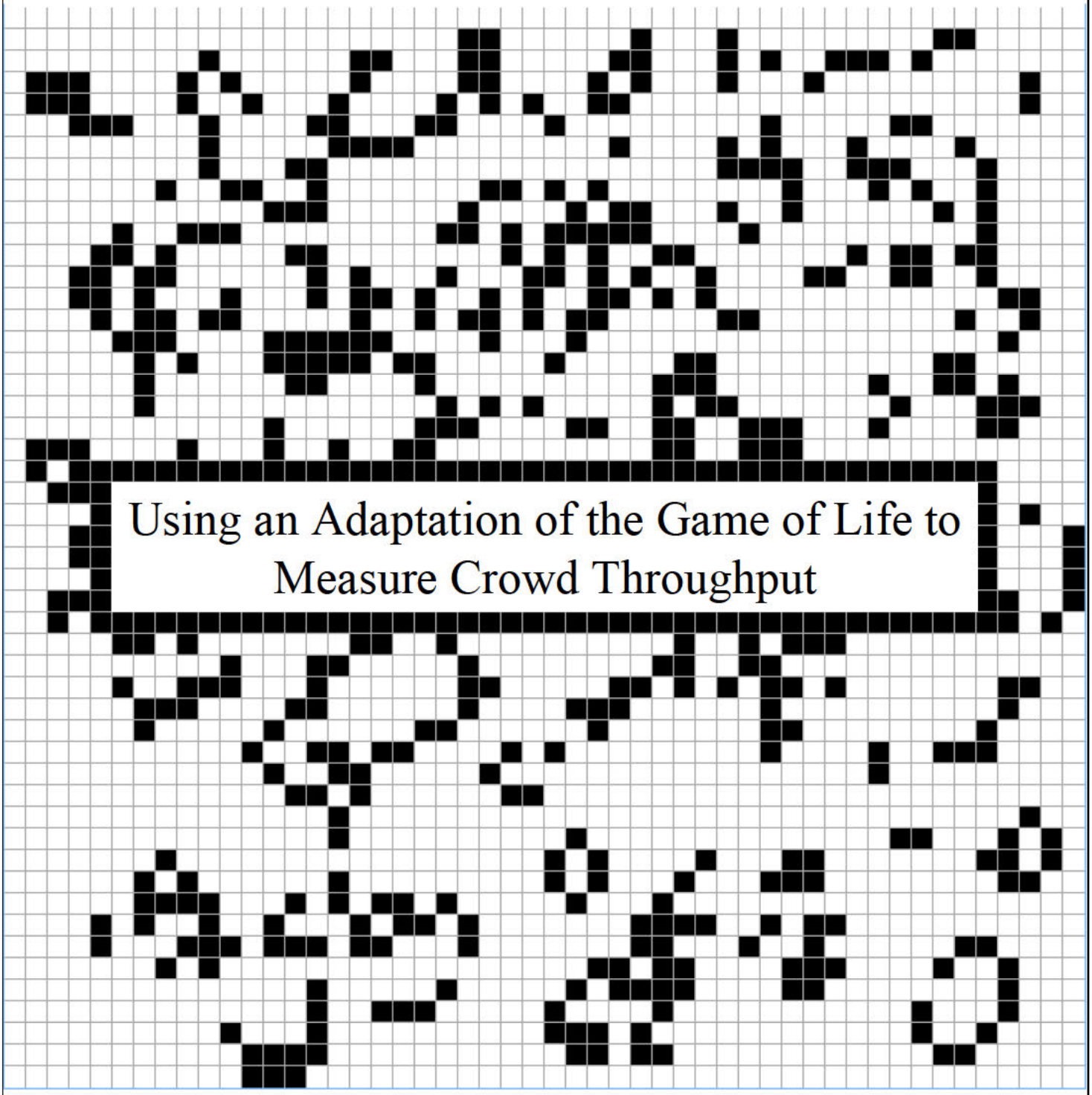

Using an Adaptation of the Game of Life to Measure Crowd Throughput

A Final Year Dissertation Submitted in Partial Fulfilment of the Degree:

BSc (Honour's) Computer Games Technology

By [redacted] [redacted] ID: [redacted]

Supervised by Dr Robert Cherry

DECLARATION

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed: _ [redacted] __

ABSTRACT

With a substantial amount of the time spent planning new buildings and walkways being allocated to considering how people will flow through them, it is an area which would benefit from a system that can effectively model the maximum viable throughput. Using a manipulated version of the Game of Life, this project attempts to address this issue by creating such a system which will be submitted alongside an accompanying report which details the stages of research and development that went into making the final product. The presentation of the results will need to be clear and concise; as such additional research into alternate methods of displaying this data will be conducted.

The product which accompanies this report can be downloaded at the following link:

https://stummuac-my.sharepoint.com/personal/[redacted]stu_mmu_ac_uk/Documents/[redacted]Product.zip

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Project Background

The Game of Life is a zero-player game which runs in cellular automaton.

The Game of Life has rules, turns (called generations), consequences and even measurable scores like any traditional game but relies on no player input so quite easily fits all the requirements to be considered a zero-player game. A cellular automaton, however, defines a range of cells that collectively form a grid which, using the traditional game comparison, acts like the game board. Each cell can be one of a range of different states that can change between generations and are often denoted by that cell's colour changing. In its simplest form, cells can be one of two states; either on or off. This is usually represented by the cell being coloured black or white.

In short, as the Game of Life is played, the rules dictate what the state each of the cells will be in the next generation using their own and their neighbours state in the current generation. Played over many generations the arrangement of cells can produce intricate and complex patterns and, at an appropriate frame rate, can appear animated.

Whilst altering the rules may only initially seem to affect the types of patterns or animations that are produced, this project will research how various adaptations of the Game of Life can perform complex computations, resolve various issues and model changes over time. This starts to highlight the problem that this project aims to investigate: to create an implementation of the Game of Life that can be used to model a real-world scenario.

1.2 The Approach

The approach to this project will be to focus on the Game of Life and how tweaking the initial ruleset of the game will affect the patterns and populations that the game produces and how it can be used to replicate more complex scenarios. Most importantly, it will focus on adaptations of the Game of Life simulation which account for non-touching neighbours or neighbours-over-distance. This will form the most important feature that must be added to the version of the game produced alongside this report.

One vital attribute will be to connect the top and bottom rows and the left and right columns to produce an infinite plain to run the game on. This will require some testing to see how well the flat plain of the Game of Life translates to a spherical grid using an implementation that projects objects horizontally and vertically towards the edges and corners of the grid.

After implementing a system to account for non-touching neighbours and work on an infinite plane the project will attempt to replicate some real-life situations which are directly affected by populations, movement and speed. This will produce data which can be collected to measure the impact of these factors on scenarios like humans moving in large crowd. Data will be plotted on graph with axis such as time (measured in generations), population and speed, and research will be carried out to look at alternate methods of displaying data from dynamic systems.

1.3 Aims and Objectives

The aim of this project as mentioned in the appended Terms of Reference is:

To Investigate Throughput Using the Game of Life and Depict the Results with Coupled Map Lattices.

The key objectives that will need to be completed for the aim to be met and for this project to be considered a success are:

- To implement a fully working version of the Game of Life;
- To expand the functionality of the Game to account for non-touching neighbours;
- Gather meaningful data which presents an insight into the relationship between overcrowding and overall movement speed and
- Research and fully understand coupled map lattices and depict the obtained data graphically through use of coupled map lattices.

1.4 Report Structure

The report section of this project will need to be coherently laid out in an order that makes logical sense and leads the reader through the development processes undertaken to satisfy the project aims.

The following chapter titles have been chosen to as the best way to achieve this:

Literature Review - This chapter is written using a culmination of the research that was undertaken whilst working on this project. As well as an in-depth explanation into the functionality of the Game of Life, there is evidence of the time that went into researching other projects that have utilised the game in similar ways. It was also necessary to gain a comprehensive understanding of coupled map lattices as they would form the primary way that the data will be presented and thus an outline of their functionality has been included in this chapter.

Design Documentation - This section of the report outlines the concepts, ideas and plans that form the code basis and the basic interface layout. The documentation should be of a high standard such that the design methodologies are clear. This is accomplished using techniques learnt in the Advanced Game Development units.

Implementation - After the design documentation had been laid out, it was possible to begin work on the first iteration of the program and the progression through multiple iterations is clearly documented in way that shows off the development process. Screenshots of code snippets are included which are used in parallel with written explanations to describe the code functionality.

Experimentation and Analysis – With the program's features all fully implemented, the best rules for modelling crowd patterns will need to be determined. With these new rules, the experiments pertaining to overcrowding, movement and speed can be carried out; the results

of which will be analysed to attempt to provide insight into the best ways to relieve traffic build up.

Evaluation – In the evaluation chapter, the project will be marked against the initial aims and objectives with considerations to its accomplishments and short comings as well as the evaluation criteria outlined in the project brief. Reasons as to why any targets that were not attained will be outlined and discussed in this chapter.

Conclusion – In the final chapter, the report will reiterate what has been achieved during the project; emphasise the positive aspects of the work and detail how problems were overcome. If it is applicable, any potential further development that could take place beyond what has been worked on here will be mentioned here too.

CHAPTER 2: LITERATURE REVIEW

2.1 Background to The Game of Life

The Game of Life is an example of cellular automaton and a zero-player game which runs on an infinite grid of cells which can each take on two states – either alive or dead. An initial arrangement of alive cells is laid out and what happens to this arrangement in the next turn, or *'generation'*, of the game is determined by a set of given rules. The rules dictate whether a cell will change its state based on how many of its eight neighbours are also alive. The three rules for Conway's Game of Life, as devised by John Conway in 1970, are as follows:

1. An alive cell with two or three neighbours that are also alive will survive to the next generation;
2. An alive cell with fewer than two or more three alive neighbours will die (of loneliness or overcrowding, respectively) in the next generation;
3. A dead cell with exactly three alive neighbours will be alive (born) in the next generation.

These rules can be applied to arrangement of alive and dead cells to produce a very wide array of patterns that are interesting to analyse. After setting these rules and determining an initial game state, the game will run indefinitely on its own without the need for any player interaction; thus, it qualifies as a zero-player game.

2.2 Cellular Automaton

Conway's Game of Life is one of the earliest examples of a cellular automaton and probably the most widely known; however, others do exist. Langton's Ant (also a zero-player game) uses a similar gird based layout made of cells that can switch between two different colours. This simulation states that an '*ant*' turns 90° to the right if it's on a white square, or 90° left on a black square, swaps the colour and moves forward one cell. Whereas the Game of Life analyses every cell of the grid to determine the next generation's layout, Langton's ant focuses on just one at a time therefore patterns tend to generate much slower. In the same way that the Game of Life replicates artificial life, Langton's Ant does too – neither simulation has concepts such as movement or reproduction implemented but do seem display these properties in their patterns as patterns and ecosystems of cells are born and die out as the game progresses

The Game of Life, in its most widely recognised form, utilises the Moore neighbourhood to analyse its eight closest cells rather than, for example, the von Neumann neighbourhood which does not consider 'diagonal neighbours' [Figure 2.1 and 2.2].

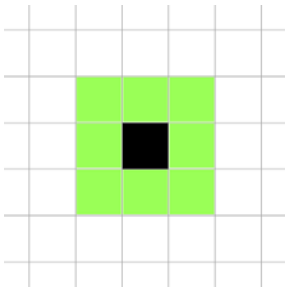

Figure 2.1: Cells highlighted within a Moore neighbourhood range of 1.

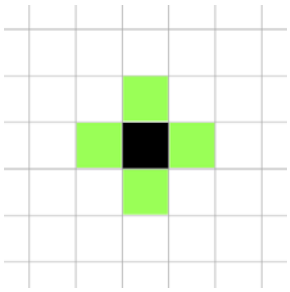

Figure 2.2: Cells highlighted within a von Neumann neighbourhood range of 1.

One of the main aspects of this project will be to expand the Moore neighbourhood to account for more than 'touching neighbours'. As shown in the figures below, a cell's zone of influence increases exponentially as the range (given from here on as 'r') is increased. The formula for finding the number of neighbour cells contained in a specific cell's range, r, is given by:

$$(2r + 1)^2 - 1$$

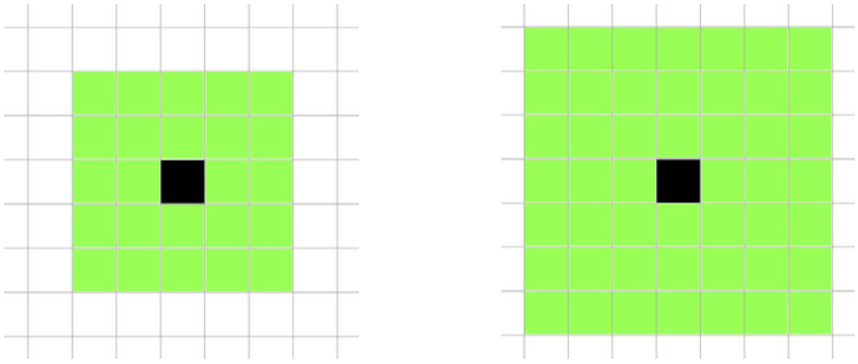

Figure 2.3: Increasing the radius of the zone of influence exponentially increases the number of neighbouring cells. These images show radii two and three.

This vastly increasing neighbour count will make searching for the nearest living neighbours of a cell require much more computing power as each iteration through the program requires exponentially more checks.

2.3 Practical Applications of the Game of Life

Small to large scale computations can be simulated with the Game of Life. This is done by utilising specific patterns within the game that can replicate simple logic gates. Gliders, an arrangement of four live cells that oscillates, and traverses one cell across every four generations, can be used transmit information over long distances. By using combinations of glider guns [Figure 2.4] projecting glider streams that intersect and interrupt each other, the basic AND, OR and NOT logic gates can be created. It is therefore possible to arrange these computational building blocks into a device that can compute information.

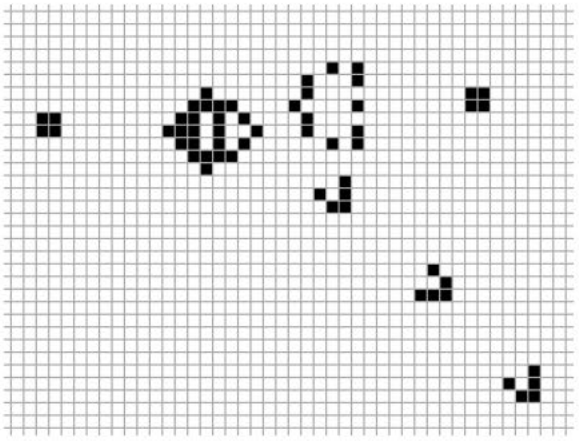

Figure 2.4: Gosper's Glider Gun emitting a stream of gliders.

However, one of the major drawbacks of modelling situations with the Game of Life is that it cannot be run in reverse as there is no way of calculating what the previous game state was. This is because many initial states can tend towards the same final state, whether it be an empty universe or an example of still life, meaning any of those initial arrangements could have been the start point. Figure 2.5 shows two different states that converge to the same state by the third generation. It is impossible to determine which starting arrangement of cells could have led to the final generation.

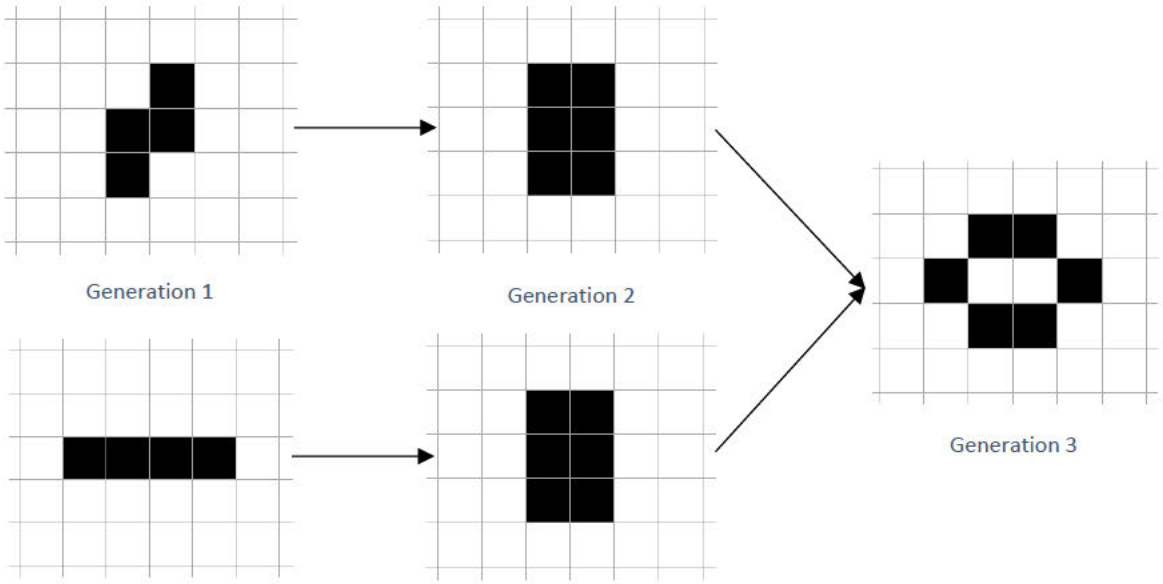

Figure 2.5: Different states of the Game of Life that converge to the same still state after three generations.

Although this can be a drawback, it can be utilised in security and encryption in the same was as very large prime numbers can. Using Figure 2.5 as an example, if Generation 3 represented a lock protecting some sensitive data, it would be impossible to determine which of the first generations, or keys, were used to produce it. The lock would only be opened when the presented with the correct key which is impossible to reverse engineer as there is no way of

determining the key just from looking at the lock thus making what's inside secure. Practically, though, a much more complex state than one with six live cells would be used. There are infinitely many states that converge to the final generation and therefore there are infinitely many keys that would need to be tested to break the lock – which would be impossible to compute [Reference 2.3].

There are also patterns that exist within the Game of life that are impossible to occur throughout the game discounting the first, player set state. For example, a grid where all, or a large number, of the cells are alive could never appear through natural gameplay. Having some layouts that will never occur could cause problems in some computational scenarios.

It is also possible to produce random numbers using cellular automata and indeed the game of life. In effect, numbers are read from a line of dead or alive cells producing a binary output of ones and zeros. In 1986, Stephen Wolfram proposed the idea that random numbers could be generated by applying cellular automata rules to an initial generation to produce a greater random sequence. He stated in 'Random Sequence Generation by Cellular Automata', that it should not be possible for a computer to revert the sequence back to its original generation which, as is mentioned here, is not possible. [Reference 2.4]

2.4 Related Works

This project, specifically, aims to investigate such real-life scenarios as people moving in large crowds and attempt to model those behaviours using a cellular automaton, the Game of Life. It will be necessary to feature different scenarios within the model such as converging crowds trying move through a small door.

In a research paper with similar aims to this one conducted on roads in New Delhi, India, researchers Mallikarjuna and Ramachandra Rao attempted to model traffic flow on the roads using cellular automata. They aimed to find out how viable it was to model traffic which contained different vehicle types with different attributes. In this case, they wanted to model traffic that contained both trucks and cars and analyse the impact they had on each other and assess the efficiencies of having roads restricted to just one vehicle type against mixed traffic. Each vehicle type had a series of attributes which included their cell dimensions, speed, acceleration and details on braking and lane changing.

Their first experiments were carried out on homogeneous traffic models containing only one of either cars or trucks. Using each of the vehicle type's attributes a simulation was produced. Vehicles attempted to travel at their max speed but were often limited when other vehicles were close.

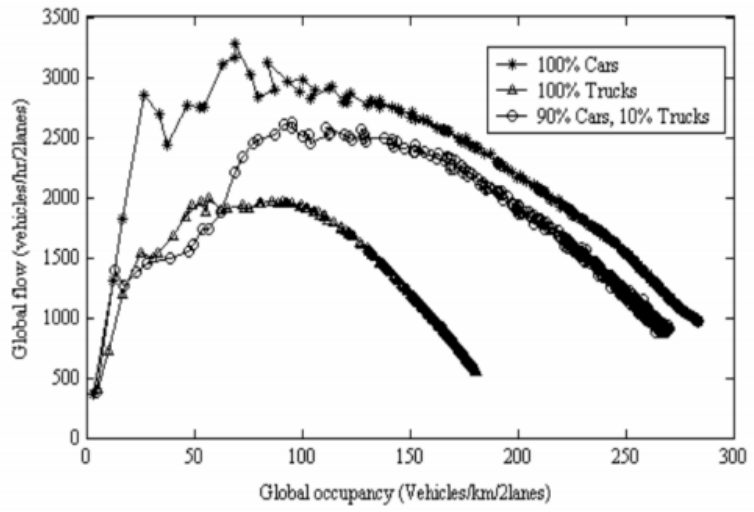

Figure 2.6: Mallikarjuna and Ramachandra Rao's finding on the relationship between traffic flow and density through use of cellular automata. [Reference 2.5]

The data shows a clear trend that is tending towards zero for traffic flow when global occupancy approaches one hundred percent. This is to be expected as if there was no space between any of the vehicles then the traffic would be at a complete stand still. They found that even including a small number of larger vehicles, shown above at ten percent, decreased traffic by a lot due to the huge increase in lane changes that were now happening. This is like how this project aims to model scenarios but using the Game of Life – by limiting speed based on how close other people are. The real-life implications of this research would be to ease traffic congestion or used in consideration when planning new roads.

This project also aims to carry out similar research to model the movement of people in a crowd. Such information would be useful to consider when planning emergency exits where congestion free walkways are important or general crowd management in narrow walkways like hallways or pavements.

Those examples work on a model where no cells are created or destroyed as the game plays out as the number of cars or people in the scenario remains constant. However, there are examples of models which are designed to demonstrate growth by constantly changing the live cell count in the system. K. C. Clarke and L. J. Gaydos utilised cellular automata in 1998 to model a projected growth of the cities of San Francisco and Washington/Baltimore. Their projections were plotted on a map which highlighted where buildings were likely to be built as those cities expanded with areas that were predicted to have the most growth were coloured in darker. This use of cellular automata in a projection model is very different as it is not a random model. A projection of this type is heavily dependent on the rules that are input to gather reliable data.

They concluded that the data was very slow to collect as where people were likely to build was based largely on a lot of different factors that were slow to input. [Reference 2.6]

2.5 Techniques Employed

The first reported mentioned in the related work section utilised a table that plotted traffic flow against the occupancy of the grid. As this project will produce data from similar inputs, it would be possible to collate the information into the same type of table. If the data produces the same shape curve, it should be possible to determine the most optimal mix between the maximum vehicles and efficient traffic flow by inspecting the data at the top of the inverse parabola.

It may also be good for this project to utilise a heat map to display data, highlighting areas where speed in crowds or on the roads was high or low which would show key areas of congestion that could be analysed to assess more efficient crowd management techniques. This is an area that my project will work in by utilising coupled map lattices.

2.6 Coupled Map Lattices

As an extension task to this project, there was an opportunity to investigate coupled map lattices as an alternative way to represent the data obtained during the investigation.

Coupled Map Lattices are one of the best methods of displaying non-linear or chaotic data types. Where a system's outputs increase exponentially or they appear random or chaotic as the game continues to run, like in the Game of Life, a coupled map lattice can be effective when used to plot the changes over that run time. Being a system that produces vastly different outcomes from very similar initial inputs, these effects can be much easier to see when the data is plotted against the total run time to produce an overall heatmap. Showing and discussing a lattice in this way would be much more insightful into the problems of traffic build up rather than just showing the final iteration of a Game of Life which would not convey the whole story.

In the case of monitoring traffic patterns, it would be expected that the grid starts out as a dark colour all over and gradually areas that had the largest build-ups of slow moving traffic would be highlighted in the lighter colour. This would produce a heat map that would make it easy to spot where the problem areas are.

CHAPTER 3: DESIGN DOCUMENTATION

The design documentation will contain an outline of the main techniques and processes used to develop the core program of the project and detail the key stages of the development cycle from planning to production. This will include plans of the layout and developing primitive pseudo code that will form the basis of the code when it comes to making the program.

This design documentation is likely to be an ongoing document that will be completed and added to whilst the corresponding part of the project is being worked on.

3.1 The Game of Life in Pseudo-Code

To produce the first iteration of the project's code, a simple pseudo code was written outlining the steps that program would need to perform to work properly.

```
for each (cell in grid)        // check every cell each generation
{
      neighbourCount = 0;

      for each (neighbouring cell)  // check the 8 neighbour cells
      {
            if (cell is alive)
            neighbourCount ++;
      }

      if (cell is alive)       // cells with 2 or 3 neighbours survive
      {
            if ((neighbourCount < 2) || (neighbourCount > 3))
                  cell = dead;
      }
      else                     // cells with 3 neighbours are born
      {
            if (neighbourCount == 3)
                  cell = alive;
      }
}
```

This code shows the program iterates through every cell between each new generation to find out the new state of the cells. A variable is allocated to keep count of the number of alive cell neighbouring the cell that is currently being analysed. Then, with that number obtained, the next generation of alive and dead cells can be calculated by applying the game's rules.

3.2 Interface Design

To give a general idea as to what the final product may look like, an interface plan was created to highlight the features that would need to be included to make the project successful.

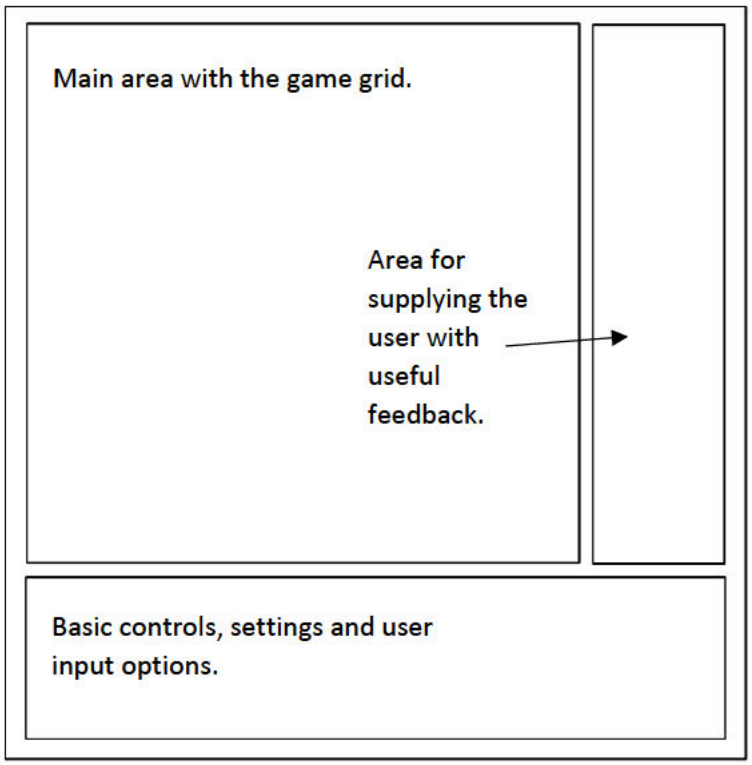

Figure 3.1: Proposed layout of the final working program.

Much of the diagram is taken up by the playing window that displays the game of life grid but space below contains the basic controls for operating the program including sliders for the grid dimensions and buttons for starting the game. The area along the right-hand side of the application is allocated for providing some useful feedback which can be collated and will form the results of the experiments. This will separate the user interface into three specific areas - one for each of: inputting initial game rules and parameters; displaying the game whilst it's running and the output and statistics of the game as it runs.

CHAPTER 4: IMPLEMENTATION

4.1 Developing the Code

Both the console application and the Windows Form Application were written in the C# programming language utilising Microsoft Visual Studio.

The first iteration of the code was developed in a simple console application to establish the basic processes before trying to flesh out the rest of the required functionality. Firstly, a set of variables that would be needed to hold the necessary information required to run the program were declared – this included the numbers of rows and columns of the game of life grid and an array to which represents the grid of cells. In this version of the code, the user if asked to input the grid's dimensions which are then used to initialise the they array's dimensions. This ability to dynamically change the grid size will carry through all the iterations of the program.

```
int gridSizeX = 0, gridSizeY = 0;
int[,] grid = new int[0, 0];

Console.WriteLine("How many rows should the grid have?");
gridSizeX = Int32.Parse(Console.ReadLine());
Console.WriteLine("How many columns should the grid have?");
gridSizeY = Int32.Parse(Console.ReadLine());

    Random random = new Random();
    grid = new int[gridSizeX, gridSizeY];
```

Figure 4.1: Declaration of initial variables some of which are set by the user.

With the grid's proportions now established, the grid can now be populated with alive and dead cells. This was easily done by iterating through the cells of the array and assigning either 1 or 0 at random with a random number generator. This forms the game of life's starting generation from which the game will be played and completes the initiation of the program.

```
for (int i = 0; i < gridSizeY; i++)
{
    for (int j = 0; j < gridSizeX; j++)
    {

        grid[j, i] = random.Next(0, 2);
    }
}
```

Figure 4.2: Randomly populating an array with 1s and 0s.

Next, a separate class named ‘Grid’ was created to keep the methods responsible for manipulating the grid together. The first method, Neighbours, analyses a cell of the array and returns how many living neighbours it has. It does this by first checking the cell is not along the edge of the grid then checks if the adjacent cell is alive which is represented by that cell being equal to ‘1’. For example, the first test checks whether the x and y coordinates are greater than zero (i.e. the cell being tested is not in the top left corner), if not the check for the cell at coordinate (y – 1, x – 1) is carried out. This function returns a number, neighbourCount, between zero and eight – as there are eight neighbouring cells.

```
private int Neighbours(int x, int y) // 8 checks. 1 for each adj cell
{
    int neighbourTotal = 0; //Cumulative tally of 'alive' neighbours

    //1. Cell Above and Left
    if (x > 0 && y > 0)//First if statement checks cell is not a 'border' cell
    {
        if (generation[y - 1, x - 1] == 1)
            neighbourTotal++;
    }
    //2. Cell Above
    if (y > 0)
    {
        if (generation[y - 1, x] == 1)
            neighbourTotal++;
    }
    //3. Cell Above and Right
    if (x < width - 1 && y > 0)
    {
        if (generation[y - 1, x + 1] == 1)
            neighbourTotal++;
    }
    //4. Cell Right
    if (x < width - 1)
    {
        if (generation[y, x + 1] == 1)
            neighbourTotal++;
    }
    //5. Cell Below and Right
    if (x < width - 1 && y < height - 1)
    {
        if (generation[y + 1, x + 1] == 1)
            neighbourTotal++;
    }
    //6. Cell Below
    if (y < height - 1)
    {
        if (generation[y + 1, x] == 1)
            neighbourTotal++;
    }
    //7. Cell Below and Right
    if (x > 0 && y < height - 1)
    {
        if (generation[y + 1, x - 1] == 1)
            neighbourTotal++;
    }
    //8. Cell Left
    if (x > 0)
    {
        if (generation[y, x - 1] == 1)
            neighbourTotal++;
    }
    return neighbourTotal;
}
```

Figure 4.3: The function used to count how many living neighbours a cell has.

To establish what the next generation of the game of life would look like, a method called NextGeneration was created which analyses each cell of the entire grid and establishes whether it should be alive or dead in the next generation based on the neighbourTotal value returned by the Neighbours method. This is where the rules of the game of life are established in the code as the next generation is dictated by what each cell’s neighbourTotal is.

In this method, there is an iteration through every cell and a call to the Neighbours method to find out its alive neighbour count which is then used to determine whether it will be alive or dead in the next generation. Cells that return a neighbour count of three are set to '1'; cells that are currently alive and have two or three live neighbours are set to '1' and all others are set to '0'.

```
public void NextGeneration()
{
    int[,] nextGeneration = new int[height, width];
    lastGeneration = (int[,])generation.Clone();
    generationCount++;

    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            if (Neighbours(x, y) == 3)
                nextGeneration[y, x] = 1;

            else if (generation[y, x] == 1 && (Neighbours(x, y) == 2 || Neighbours(x, y) == 3))
                nextGeneration[y, x] = 1;

            else
                nextGeneration[y, x] = 0;
        }
    }
    generation = (int[,])nextGeneration.Clone();
}
```

Figure 4.4: Using the neighbour count to determine what state the cell will be in the next generation.

The final key method of this class is DrawGeneration which is responsible for taking the array of ones and zeroes and drawing them in the console window to look like an arrangement of alive and dead cells in a grid. This program does this by replacing each '1' with a '#' to indicate and alive cell and each '0' with an '_' to indicate a dead cell. After each row of the array is written out, a new line is started.

```
public void DrawGeneration()
{
    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            if (generation[y, x] == 1)
                Console.Write("#");
            else
                Console.Write("_");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}
```

Figure 4.5: A function that uses the grid to draw the generation to the console.

There are also a couple of other smaller methods mostly used to provide some extra information to the user when the program is running. Firstly, the AliveCells method checks to see if there are any alive cells in the current array. This is used to terminate the program if there are no longer any alive cells. There is also a method, GenerationCount, that simply returns the generationCount variable which is increased in the NextGeneration method and used to show how many generations have been iterated through.

With all these combined, a while loop iterates while the AliveCells method returns a value greater than zero and then repeatedly calls the DrawGeneration and the NextGeneration classes. This image shows the user declaring some dimensions for the grid and the first two generations of the game after the grid has been randomly populated and one further generation has been calculated.

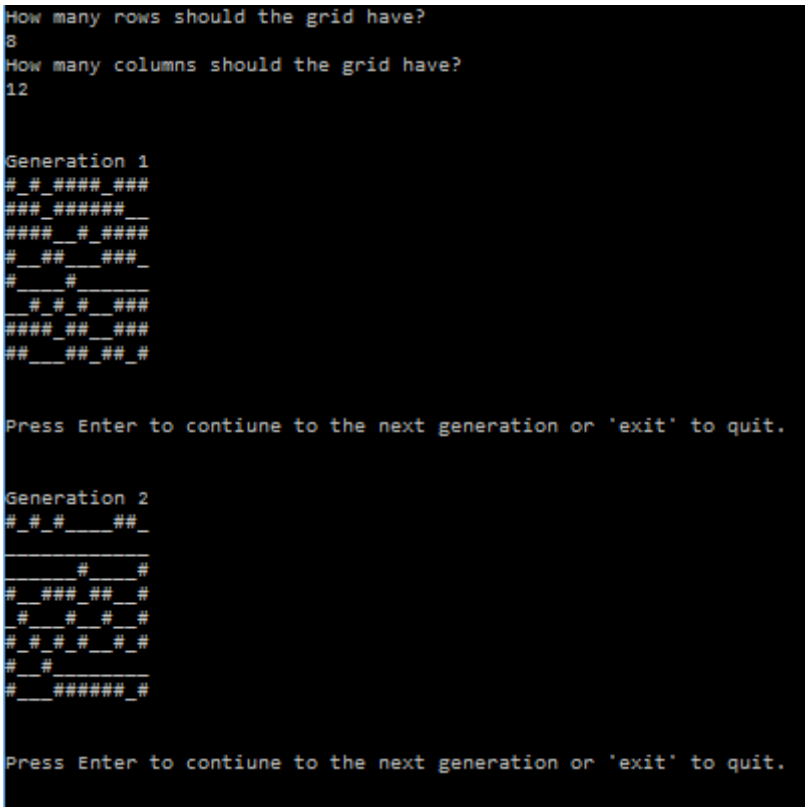

Figure 4.6: The first iteration of the program running in the console.

Some other options have also been added to the start of the program that allow the user to open some predefined noteworthy game of life pattern such as gliders and blinkers to add some increased functionality. In the next iteration of the program there will be a need for a more graphical display of the grid; an option to alter neighbour counts to account for non-touching neighbours and link opposite sides to produce an infinite plain.

4.2 Conversion to a Graphical Application

It was now necessary to produce a graphical version of the program which allowed for much more interactivity. A lot of the background code carried over quite easily and could form the basis of some more in-depth functionality.

Rather than having a function that prints either a '#' or an '_', a function must be written that draws a filled square for alive cells and an empty square for dead cells. Utilising C#'s graphics controller, the following function [Figure 4.7] was written within the Cell class which checks whether itself is set to alive and draws an appropriately filled or empty rectangle.

```
public void Draw(Graphics graphics)
{
    if(Alive)
        graphics.FillRectangle(BlackBrush, x, y, width, height);
    else
        graphics.DrawRectangle(GrayPen, x, y, width, height);
}
```

Figure 4.7: A function that draws either a filled or empty cell.

An option to turn on debugging text was also added now which allows displays the current state, the neighbour count and what the state the cell will take in the next generation. A check box toggle option is put in the controls box to allow this text to be toggled on and off at any time. Figure 4.8 shows these two features working together in the new visual version of the game.

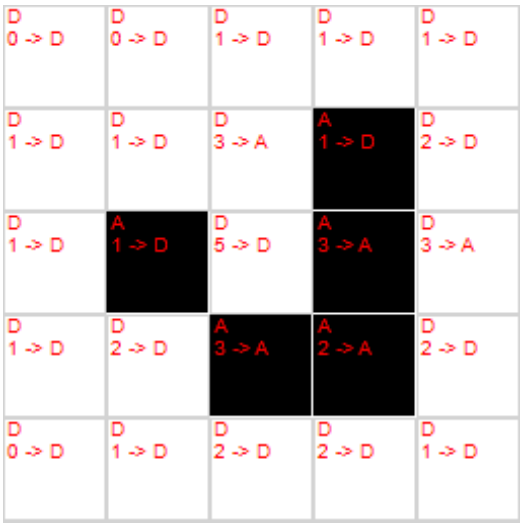

Figure 4.8: The game running with debug text info text turned on.

4.3 Producing an Infinite Plain

To model more complex situations and as per the project aims it was necessary at this point to produce an infinite grid. Doing so would allow the game run on an effectively infinite or spherical plain. This would involve linking the edges such that when neighbouring cells are searched for, cells along the edges would have neighbours on the other side of the grid.

A new function was created with refactored code for counting neighbours which now accounts for, and performs a search for cells along the edge of the grid. This is done by allocating new variables for below, above, left and right then checking if they are set to values outside the range of the game's grid. Any that are will be reassigned to reference the appropriate cell on the other side of the grid [Figure 4.9].

```
int NeighboursInfinite(int x, int y)
{
    int AliveNeighbours = 0;
    int Below = y + 1;
    int Above = y - 1;
    int Right = x + 1;
    int Left = x - 1;

    if (Below > MAX_ROWS - 1)
        Below = 0;
    if (Above < 0)
        Above = MAX_ROWS - 1;
    if (Right > MAX_COLS - 1)
        Right = 0;
    if (Left < 0)
        Left = MAX_COLS - 1;

    //Check cell on the right.
    if (this.Cells[Right, y].Alive)
        AliveNeighbours++;
```

Figure 4.9: The new neighbours function that links edges of the grid.

Again, a toggle was added to the controls box which allows the user to switch between using an infinite game grid and the classic finite style.

4.4 Expanding the Functionality to Work Over Distance

As per another one of the main objectives in this project, it was necessary to add functionality which allows the program to search for a cell's living neighbours who aren't directly touching. As discussed in the Chapter 2, increasing the range exponentially increases the number of cells that need to be checked and as such a search diameter of seven was decided as an upper limit. This restriction was decided on due to multiple factors. Firstly, performance rates took a large hit with every increase in search range and the next step up would have required eighty searches per cell in the grid up from the current maximum of forty-seven. Secondly, with a greater search range matched with an infinite grid, it caused issues where a cell was neighbours with another cell twice or a cell was its own neighbour.

When the infinite plain option is selected, users can choose to increase the size of the search range using the numeric up/down box in the controls box. It should be noted that it is not possible to set the dimensions of the grid to a size that is smaller than the search range that has been set. Figure 4.10 shows the game using a search range of three and working with the debug text feature enabled which displays the cell's alive neighbour count and its state in the next generation.

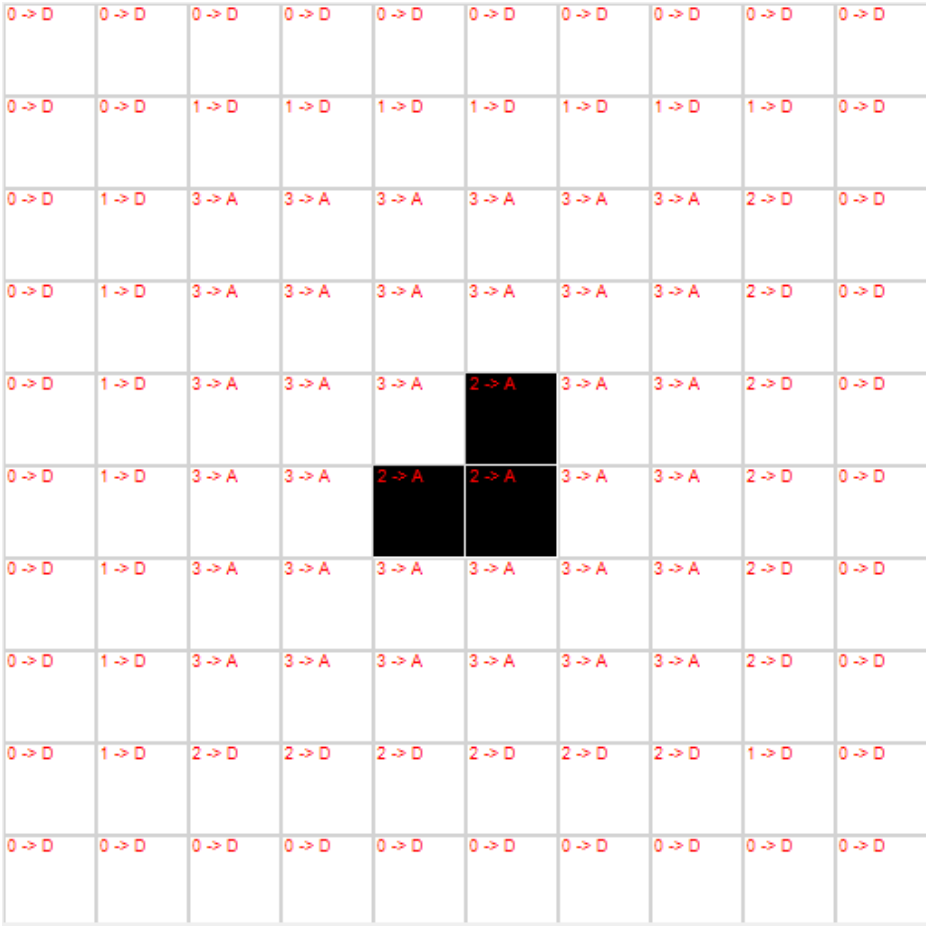

Figure 4.10: Game of Life running with a search range of three.

4.5 Adding a Coupled Map Lattice Feature

It was now necessary to add a feature that could produce a coupled map lattice at any time during the playing of the game. For this to be possible each cell would need to track how many times it had been alive throughout the game. A variable was added to the Cell class that was then increased every time that the cell was set to alive. Using this, and knowing the total number of generations, it was possible to calculate at what percentage of the total game each had been set to alive.

With this new information about each cell, the game could now draw the coupled map lattices. Cells that had never, or very rarely been alive, were coloured in a dark red and cells that were alive frequently alive were set as a lighter colour. Utilising a range of colours between dark red and light yellow for all the percentages produced the desired lattice and heatmap effect, an example of which can be seen in Figure 4.11. Over the course of this game's life, areas that are frequently dead are coloured in darker than those that are alive more often. Again, this is an option that can be toggled on and off at any time with the check box in the controls. This Coupled Map Lattice shows the 100^{th} generation of a game on an 80 x 100 sized grid where the initial generation was a checkerboard pattern.

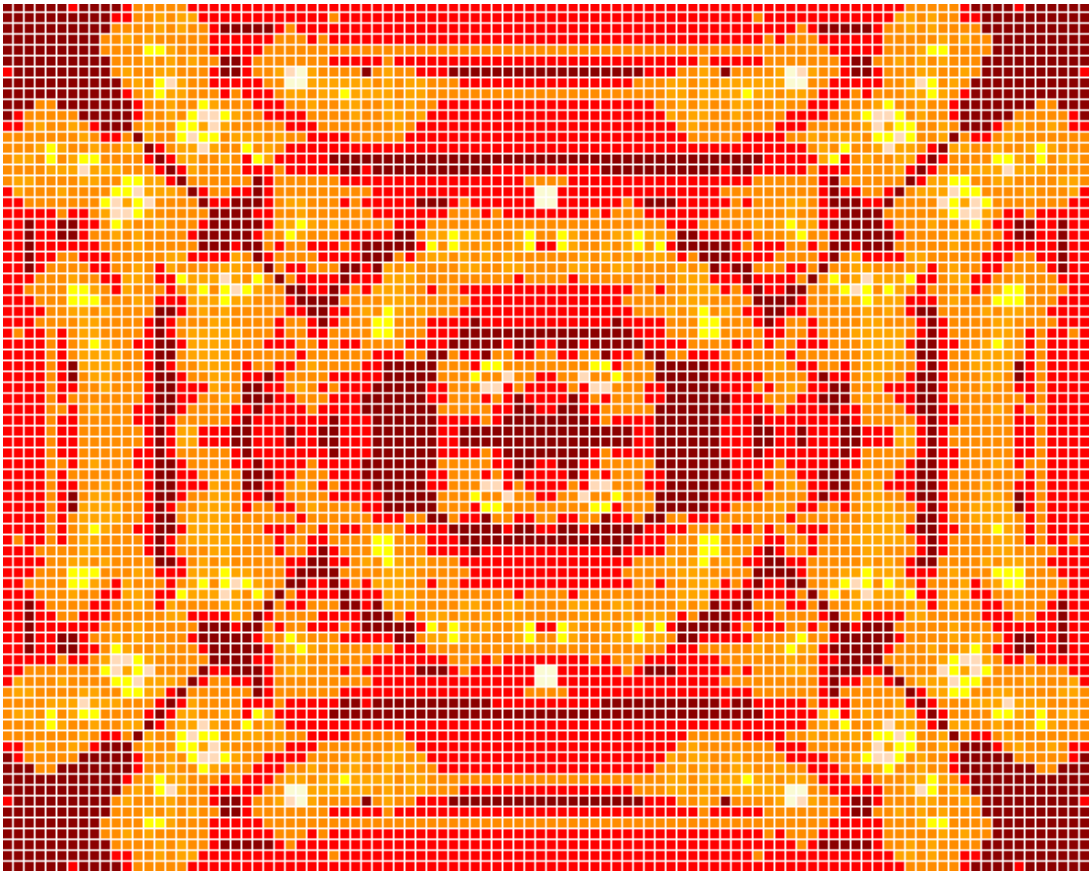

Figure 11: The game running with the coupled map lattice option enabled.

4.6 Finishing Touches

At this point it became apparent that a new type of cell would be required outside of what the current alive or dead cell typing could support. It was necessary to implement a '*wall'* cell state that would ignore rules of the game stay present indefinitely. This would be used to contain or restrict the human cell's movements when traversing through complex environments like a room or hallway.

Implementing this feature was achieved by replacing the cell class' Alive Boolean which recorded whether a cell was alive (true) or dead (false). This was replaced with an Alive integer

which recorded the cell's state with a number value, rather than a true or false, details of which are recorded in Table 4.1. Wall blocks can be placed by right clicking in any cell.

int Alive	State	Appearance
0	Dead	White
1	Alive	Black
2	Wall	Black lattice

Table 4.1: The three states that a cell can take.

Figure 4.12 shows the final working program with all its features implemented. It should now be possible, with a few tweaks to the rules, to use this program to model the scenarios set out in the aims of the project.

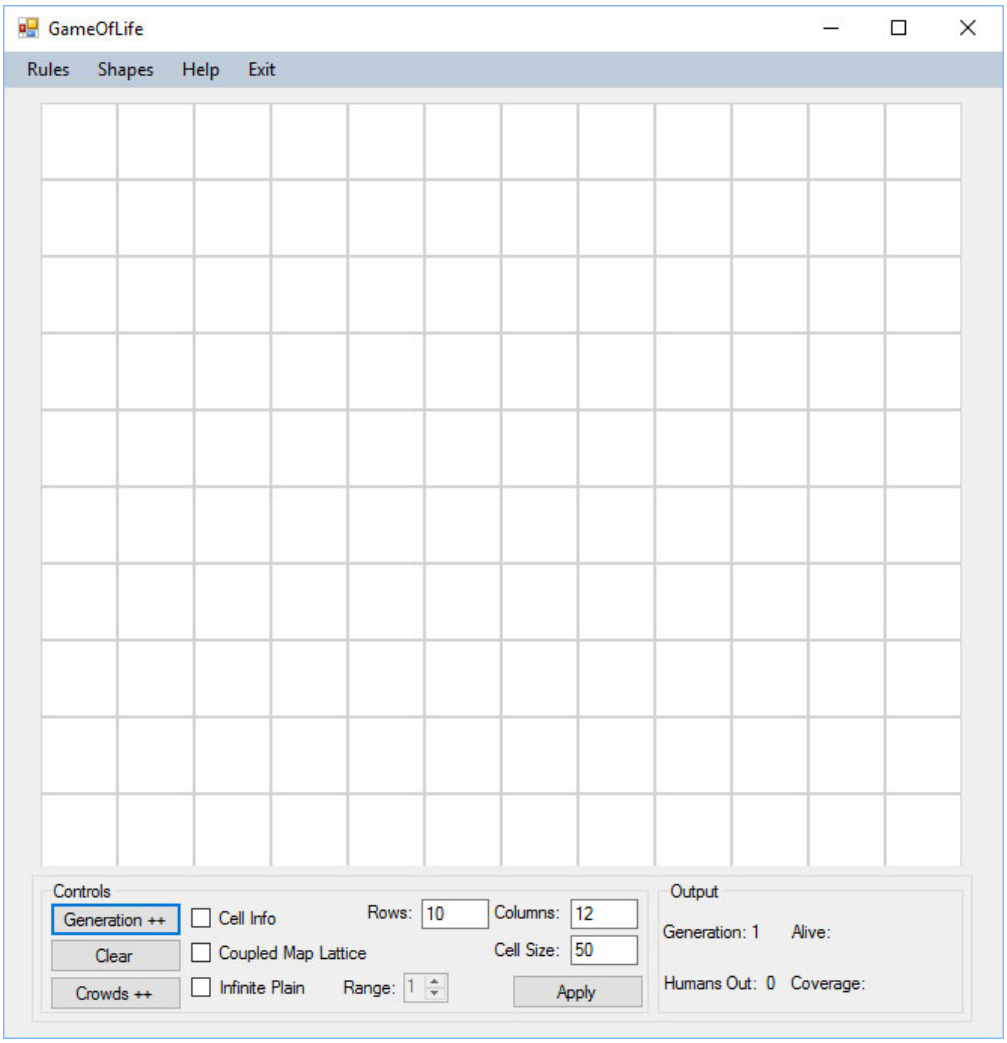

Figure 4.12: The product fully finished and implemented.

CHAPTER 5: EXPERIMENTATION

It is now possible, with the completed program, to begin to form system that can model the system outlined in the objectives of this project. This will mean perfecting a set of rules that give a true representation of crowd traffic patterns. This chapter describes how the rules were decided and analyses the results obtained from running the models.

5.1 Declaring the Rules

In each scenario, a room or environment was constructed and an arrangement of 'human' cells were spread loosely around the end furthest from the exit. The game will then play one generation at a time with the humans moving in the direction of the exit. Using the previously devised rules for searching for neighbours, humans will give priority to less crowded areas; moving forward only when the next step has fewer neighbours than where they are currently stood. When the game has played all the way through, the coupled map lattice will be produced and it should be apparent where the biggest build-up of traffic was. A prediction at this point would be that there are build-ups at the start and around key chokepoints.

This first environment to be in which the simulation would be run and was intentionally kept simple to test these new rules with the aim of producing a more complex room later. Each cell of the room contains a direction indicating the fastest route out of the room as shown in Figure 5.1.

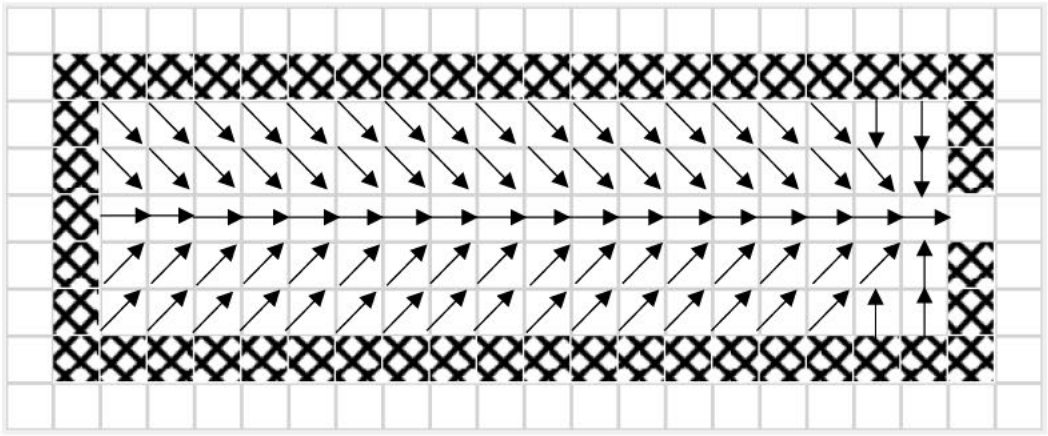

Figure 5.1: The Hallway layout with arrows indicating the route towards the exit.

A more complex room was then designed to further test these behaviours. This room contains two conjoining routes which should have an interesting effect on the movement

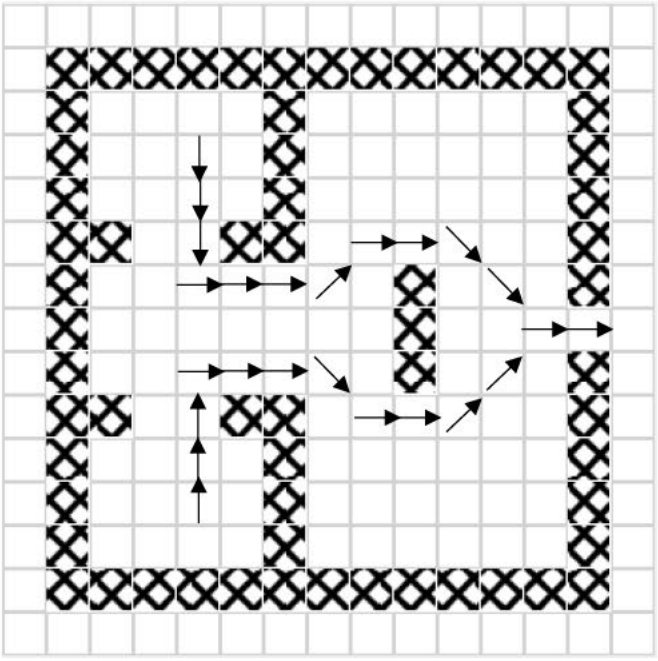

Figure 5.2: The Room layout with the two critical paths shown

5.2 Results

The first experiment that was carried out aimed to model humans walking down a hallway and determine the impact that overcrowding had on the overall time taken to traverse the given distance. As multiple human cells cannot occupy the same grid square, it would be assumed that as the distance they are told to keep apart increases the overall time taken would increase to. Each time that the test was run, the distance the human cells had to keep apart was increased and every time one of the ten humans reached the exit, the current generation was recorded. The first test however, used a search range of zero; this meant if the cell in front was available then the human would move there without any consideration to overcrowding. The data is also shown graphically in Figure 5.3.

Hallway										
Test Number	Search Range	Generations Taken for Humans to Leave								
		1	2	3	4	5	6	7	8	9	10
1	0	20	21	22	23	24	25	26	27	28	29
2	1	20	22	24	26	28	30	32	34	36	38
3	2	20	23	26	29	32	35	38	41	44	47

Table 5.1: Results from running the game on the Hallway layout.

The next series of results are from the test carried out on the Room environment. Using the same rules as before, the humans were split evenly with five starting in the upper left room and five in the lower left room.

Room										
Test Number	Search Range	Generations Taken for Humans to Leave								
		1	2	3	4	5	6	7	8	9	10
4	0	14	15	16	18	19	20	22	24	25	27
5	1	14	16	18	20	22	24	26	28	30	32
6	2	14	16	18	21	23	25	28	30	33	36

Table 5.2: Results from running the game on the Room layout.

5.3 Analysis

As would have been expected, cells at the front of the crowd moved away first leaving those at the back stationary for multiple generations, whereas cells that start off at the front were always able to move to the exit regardless of the search range. However, since the program had to be altered to include a first-come-first-serve system to prevent humans moving into the same cell, a very clear bias towards cells that are searched first can be observed.

It can be seen from the results that the first human to get out took the same time regardless of the search range. This would be expected as its path to the exit was clear never obstructed. Subsequent humans took longer to leave due to more restricted opportunities to move forward.

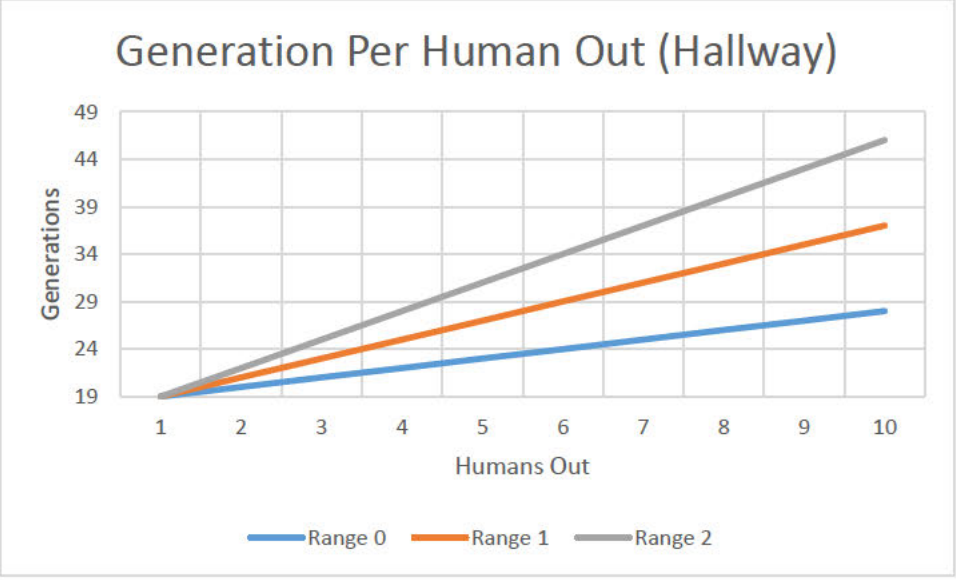

Figure 5.3: Data collected from the Hallway experiment displayed on a graph.

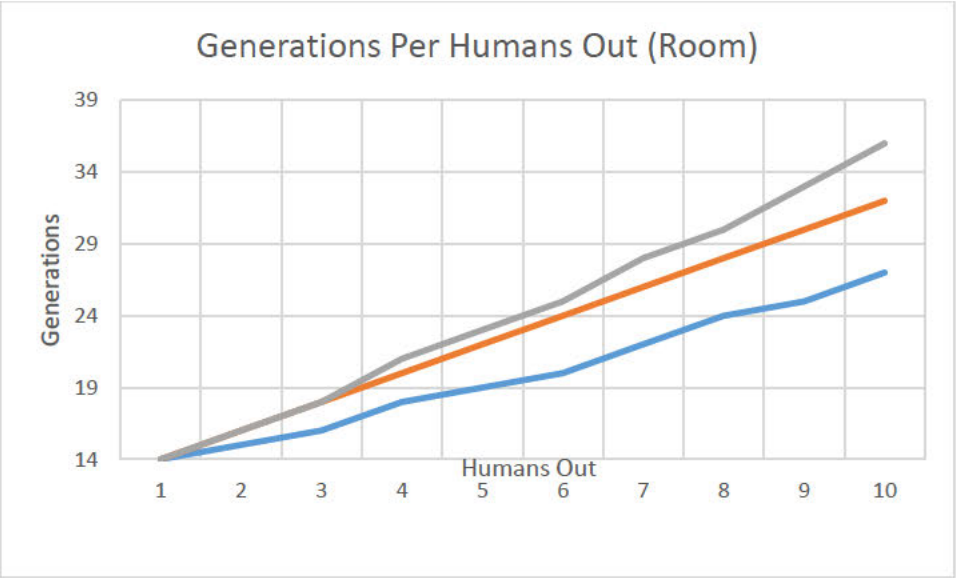

Figure 5.4: Data collected from the Hallway experiment displayed on a graph.

The following figure shows a coupled map lattice for the search range of zero test that was carried out on the Hallway environment. The coupled map lattices show a very good summary of how the dynamic system changed over time; better than an image taken of a single generation could. The lighter coloured cells show the expected build up around the start that occurs while the initial crowd attempts to disperse. However, it also shows, quite clearly, the cells in the top half as darker as there was less build up there due to the advantage of being allowed to move first.

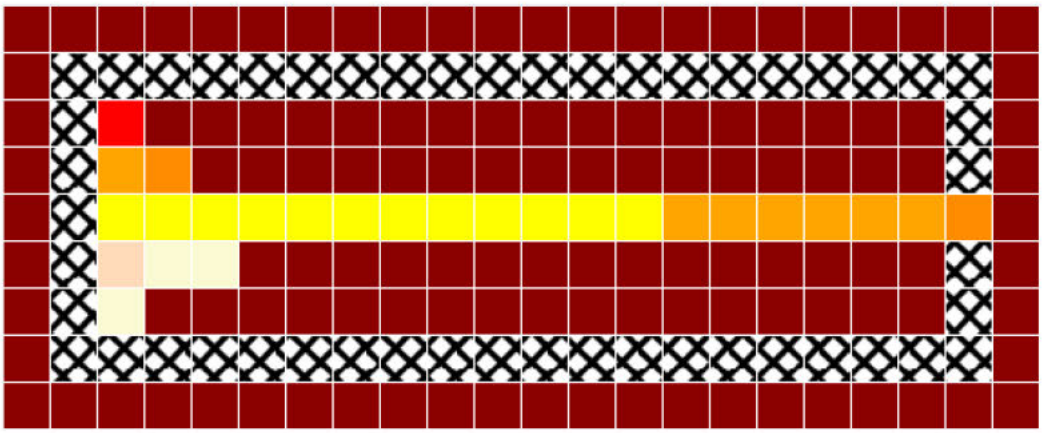

Figure 5.5: Results from running the experiment in the Hallway with a search range of 0.

As can be seen from the coupled map lattice in Figure 5.6, there is a build-up of traffic highlighted my lighter colours - most notably at the beginnings and the point where the two paths converge. Again, there is a clear bias against cells that are analysed later. Whilst the build ups around the two starting areas are similar, the area before the convergence is much lighter on the bottom path. This happens because the humans on the bottom path get stuck in this area until all the humans of the upper path have passed.

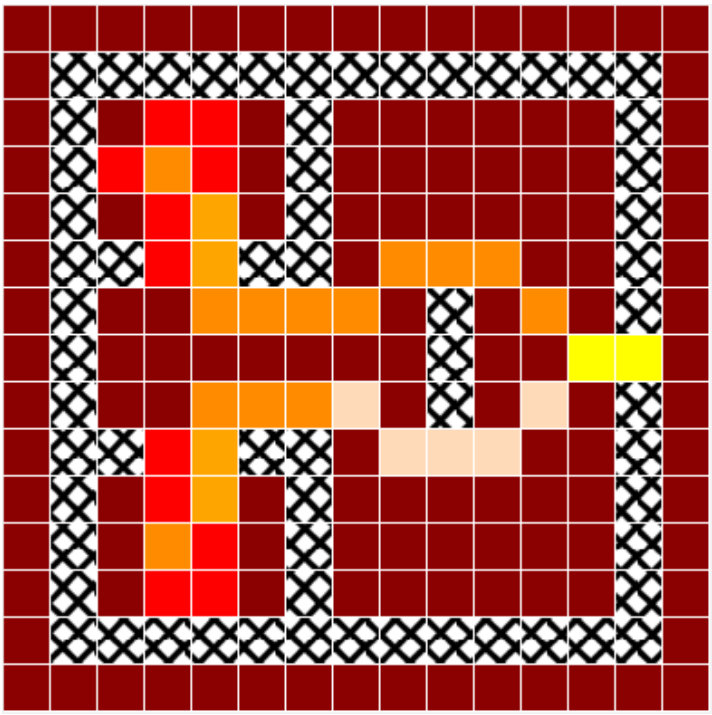

Figure 5.6: Results from running the experiment in the Room with a search range of 0.

The following evaluation and conclusion chapters will discuss in more depth the effectiveness of these experiments model the various scenarios as well as the successes and failings of all aspects of the project compared to its original aims and objectives.

CHAPTER 6: EVALUATION

This evaluation will look back through the project and reflect on how successfully the aims set out in the introduction have been tackled and if the objectives have been satisfied. Both the achievements and limitations of what has been created.

6.1 How Successful is the Work?

The successful implementation of the Game of Life with all the accompanying features is one of the most successful aspects of this project. On its own, the program serves as a viable representation of the Game of Life and features such as toggling cells between alive and dead by clicking, clearing the board, changing the dimensions of the cells and the grid, showing cell information and the output information that is displayed all contribute to its standalone functionality.

The first major implementation that was required to take this program beyond what the original Game of Life could do was a system that searched further than direct, touching neighbours. This was successfully implemented up to a search range of three, taking the number of neighbouring cells that each cell has from eight up to forty-seven. Cells could infer how far they were from other cells and this would become a crucial feature when attempting to model crowd behaviours. Along with another added feature which connected the top and bottom and left and right edges, the project was well under way to successfully meeting the objectives laid out in the introduction.

The tertiary task of considering an alternate method of displaying data lead to researching coupled map lattices. It was observed how effective they could be at conveying information taken from a dynamic system that changed over time so implementing such a feature into this program became one of the primary goals. With this feature then successfully implemented, it was then possible to utilise coupled map lattices to better analyse results gained from carrying out the experiments had this feature not been included.

The program achieves the goals it set out to obtain but ultimately falls short of being a true representation of the scenarios that it attempts to model. Unfortunately, the more closely the original aspects of the Game of Life are adhered to, the harder it is to model crowd patterns in this way. Too many conflicting factors come in to play within the Game of Life and human movement that are just not compatible when attempting to design a system which uses both. In the Game of Life, all cells are analysed independently of each other however with humans this cannot happen as it would mean two humans could move into the same cell – there must be some check that a human hasn't already claimed a cell heading into the next generation. This opens another problem in that whichever cell is searched first gets the opportunity to claim open cells first which would never happen in the Game of Life.

CHAPTER 7: CONCLUSION

7.1 What has Been Achieved?

The initial objective in this project was to implement a fully working version of the Game of Life. Not being an incredibly complex program, this was achieved with relative ease although the version was primitive and had very few features. The program went through multiple iterations, each time adding features most notably: becoming a graphical application from a console application; the ability to click cells and change their state at will; ability to dynamically change the game's dimensions and the ability for the game to work over an infinite plain by joining the opposite edges. As it stands the program is a successful implementation of a cellular automata with support for multiple rulesets beyond Conway's Game of Life.

However, meeting the rest of the objectives would require further development. As a background task to this project, coupled map lattices were suggested as useful alternative method of displaying data gathered from dynamic systems like the Game of Life. As such, an option to toggle the current game to a CML-style layout was added. This resulted in an image that told a better story of that game's lifetime much better than a still image of the final frame did.

The last objective was to allow the system to search for neighbours beyond those that are touching. This was accomplished up to a range of three, searching the neighbouring forty-eight cells up from just eight with the default rules.

With these objectives met, it was now possible to begin experimenting with the inputs to produce a system that models movements in crowds and round off the final objective of this project.

7.2 Limitations of this Project

Whilst this program meets its initial aims, it is with some liberties that it can be called a perfect representation of crowd movements. Firstly, as humans operating within the rules of this program will never move to a cell that is more crowded than the one they are currently in, it would not be possible for them to move past each other in opposite directions in a narrow walkway. This would not be possible as they would have to move closer to each other at least at some point. With some further development, it would be interesting to implement an option to allow people moving towards different destinations; though such a system is out of the scope of this project.

One major drawback of using the Game of Life to model crowd movement is that cells in automata act completely independent of each other unlike human behaviour. In the Game of Life, each cell's future state is calculated at the same time before the next generation is drawn however doing this in the model of human movement would allow two humans to move into the same cell in the same generation. To counter that problem, this program must utilise a first-come-first-serve rule where a cell must check that it's not moving into another human but also

that it's not moving into a cell that has already been claimed by a different cell earlier on in the current generation. This bias to the cell that is analysed first is not present in the real Game of Life as every cell has the same opportunity to progress.

Another smaller drawback of this model would be that movement in a diagonal version is the same as one vertical and one horizontal move. This would allow a person to traverse a room diagonally in the same time it takes to travel horizontally – a clearly much greater distance.

This leads on to an area of further development that would be nice to implement. For example, if a human wants to move in a diagonal direction but that cell is already occupied, it would be good to then check if a movement horizontal or vertical is available. This would make the model more natural but as a diagonal move is worth two moves in a straight direction, this would have little impact on the program in its current state. If this project was to be reproduced, or added to with any of these further development options, it would be suggested that the program be remade with a different application type that has a better graphical performance as the current program pushes to the limit what can be accomplished in C# Windows Form Applications.

As highlighted in the evaluation, the aspects of program came together as expected, it was packed with features like an infinite plain, search over distance and coupled map lattices and provided useful information for analysing but unfortunately fell short of truly replicating human crowd movements. Attempting to adhere to closely to the rules set out in the Game of Life held back what could be accomplished with crowd movements whereas, attempting to implement a true crowd model demanded features that are not present in the Game of Life.

REFERENCES

Images not given a source reference are screenshots taken from the product accompanying this report.

Chapter 2 References

[2.1] Tyler, T. *The Moore neighbourhood.* Cellular Automata FAQ [Online] [Accessed on 21st November 2016] http://cell-auto.com/neighbourhood/moore/

[2.2] Tyler, T. *How do you do computations with the Game of Life?* Cellular Automata FAQ [Online] [Accessed on 23rd November 2016] http://cafaq.com/lifefaq/index.php?open=how_compute#how_compute

[2.3] Standford University. *'Practical Applications'.* A Discussion of THE GAME OF LIFE [Online] http://web.stanford.edu/~cdebs/GameOfLife/

[2.4] Wolfram, S. (1986) 'Random Sequence Generation by Cellular Automata' *Advances in Applied Mathematics.* 7 pp. 123-169.

[2.5] Mallikarjuna, C. and Ramachandra Rao, K. (2007) 'Identification of A Suitable Cellular Automata Model for Mixed Traffic' *Journal of the Eastern Asia Society for Transportation Studies.* 7 pp. 2454–2469.

[2.6] Clarke, K. C. and Gaydos L. J. (1998) 'Loose-coupling a cellular automaton model and GIS: long-term urban growth prediction for San Francisco and Washington/Baltimore' *International Journal of Geographical Information Science,* 12:7 pp. 699-714,

Appendix A References

[A1] University of Toronto. (2002) *Nim Strategy*. [Online] [Accessed 2nd October 2016] http://www.cdf.toronto.edu/~ajr/270/probsess/03/strategy.html

[A2] Wójtowicz, M. (2001) *Cellular Automata rules lexicon.* Family Life. [Online] [Accessed 1st October 2016] http://www.mirekw.com/ca/rullex_life.html

[A3] Sokół, J. (2015) *Chaos and game theory*. [Online] [21/10/2016] https://prezi.com/pkz7oyabsar0/chaos-and-game-theory/

APENDICES

Appendix A: Terms of Reference

Project Context

The phrase ‘zero-player game’, to most people, may sound quite redundant. How is it possible that a game could be played without any players? Zero-player, in computer games, suggests no human interaction and instead utilises artificial intelligence which still sticks to a list of rules as any other game would have. John Conway’s Game of Life is an example of such a game which runs on a grid of cells and enforces a set of user inputted rules but after that the game will run completely independent of any player interaction and creating very complex patterns.

It is possible to refer to many other games, or for them to be thought of in terms of, as zero-player games. Just simply adhering to the rules laid out by the game can sometimes enforce the next move the player must make. This is most easily observed in, for example, noughts and crosses or connect four, where it is common place for a player to have no choice but to play a certain move lest their opponent instantly win on the next turn; or in games run entirely off dice rolls such as Snakes and Ladders which, apart from the need for a player to physically roll the die, do not require any player choice. Often, strings of alternate players’ moves are dictated for them simply by the rules of the game and being forced into a specific move.

This brings up the idea, at least in part, of games that are contested between multiple players being zero-player games. In a game where both players are playing the single most optimal move on their turn, there is no longer any need for player choice and, by extension, any players. This leads onto ‘perfect play’, or ‘solved games’, where it is possible to implement a strategy that without fail allows a player to always win. The strategy to always at least win or draw a game of noughts and crosses is widely known and as such the player going first is at the advantage and can play ‘perfect’ moves that will never end in a loss. Whereas in the ‘Nim Game 21’, the player going second will always win if implementing a perfect play strategy [Reference A1]. This idea can be extended to more complex games like chess or Go, however neither have been perfectly solved and therefore there is no computer program that is unbeatable.

Aim

Investigate Throughput Using the Game of Life and Project the Results onto Coupled Map Lattices.

Related Work

Much time has been spent experimenting with alternate rulesets to research different patterns those games can produce. A game can be referred to with a simple notation which states: a Game’s rules are given by ‘Sx/By’ where ‘x’ is the list of neighbour counts required for survival

and ‘y’ is a similar list but for neighbours required for new cell generation. As cells in the Game of Life require either two or three live neighbours to survive and a dead cell requires three live neighbours to be born, the notation for Conway’s rules can be written as S23/B3 [A1].

There are some simple adaptations on the original rules which changed the neighbours required for survival and birth which, when implemented into a Game of Life simulation, produce very different results. Just changing the neighbour counts required produces vastly different patterns. Patterns created by these changes in rules tend to fall in three main categories – either chaotic, exploding or expanding. Conway’s Game of Life displays a chaotic pattern; it is difficult to predict what the result of an initial arrangement will be but don’t tend towards a fuller playing grid. Exploding patterns like Gnarl (S1/B1), often have very few neighbour counts as such the patterns they produce ‘explode’ – taking up a lot of space on the grid very quickly. Whereas expanding patterns, are usually more generous with the survivability as seen with Flakes (S012345678/B3). This creates patterns that expand from the initial input like exploding patterns but at a much slower rate. Also, the expanding patterns produce still life in the centre but exploding pattern’s centres are still evolving as the pattern reaches further out.

‘Chaotic’ gameplay is defined in a more general state as dynamic systems where small changes in the initial state produce vast changes in later stages – also known as the butterfly effect [A3] - which makes it different from randomness as chaotic system’s futures are already determined. Therefore, no pattern type is ever described as random as, due to the nature of the Game of Life, all patterns are derived from their initial state.

There are also adaptations of the rules that add a ‘lives’ count to each cell. An alive cell is given extra chances after satisfying the neighbour count for dying. These such rulesets are given by ‘Sx/By/Cz’ where ‘z’ represents the number of lives each cell is given when it’s born. As the cell decreases in lives, it cycles through the colours before dying and being filled by the final colour.

Solution

With a fully implemented simulation of the Game of Life, the project will aim to setup scenarios which will produce measurable outcomes. Ideally, the user of the program would be able to tweak the rules to influence how the entities react and how much of a difference changes in the rules can make. While pursuing the cars in traffic idea, the user of the program would need to limit speed depending on how close vehicles are to each other and examine the best ruleset that can be implemented to mitigate traffic build-up by measuring overall speed. This would be like the experiment carried out for people moving in a large crowd or moving into a bottleneck. It would be interesting to see what impact neighbour counts has on speed in both scenarios when the data is represented in coupled map lattice.

Learning Objectives

Working on various elements of my project will also allow me to work towards the Computer Games Technology course-specific learning outcomes.

I will draw on my knowledge of computer programming and graphics software to implement an interactive system which I can use to gather data for my project. Doing so I will be meeting the second and third of my course's learning outcome and further my knowledge on these topics:

- *To become knowledgeable in the use and development of computer graphics software/game middleware tools and to be able to apply this knowledge to the implementation of real-time interactive systems;*

- *To learn structured approaches to computer programming;*

My research and study of the theory behind the Game of Life and its behaviour systems will meet learning outcome four. This outcome relates to enhancing playability of games by analysing such systems. The data I collect from tweaking the simple rules of the Game of Life can be adapted to apply to much more complex games in such that I can increase playability in future games:

- *To learn how the theories and techniques of behavioural systems can be used to enhance the playability and sophistication of computer games;*

Working on this project will further my skills in project management which is a crucial skill required to work in game design as outlined in the final course learning outcome. To produce a successful project, I must work to tight schedule, meet deadlines and liaise with my project supervisor regularly which led to me creating a work schedule:

- *To gain an appreciation of the multidisciplinary environment in which commercial games are designed and produced and to acquire skills in project management and team working.*

Tasks and Timetable

This is a Gantt chart that depicts the appropriate amount of time that should be spent on each element of the project throughout the year as seen below.

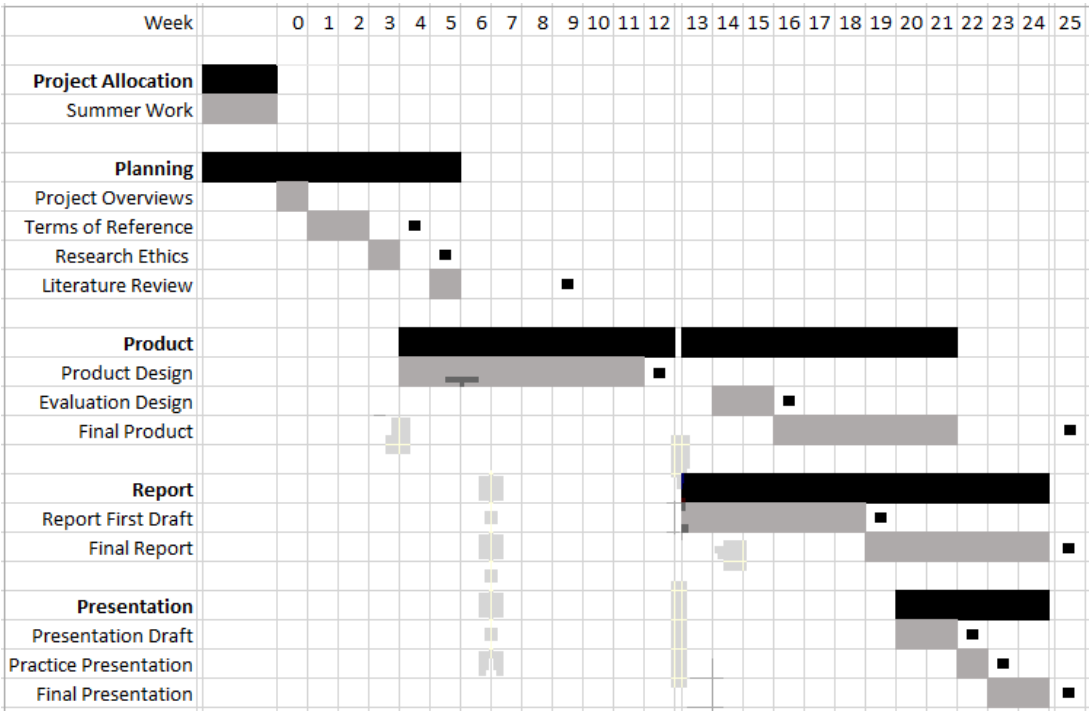

Figure A1: Gantt chart showing time allocated to all tasks involved. Time spent on a task is highlighted in grey. Various due dates are highlighted by a black square.

Over the summer, before returning to university, I spent some time planning my project and creating a simple prototype program, after receiving my allocated project topic.

The first portion of work comes in the form of planning. I've allocated five weeks to planning the project in which time I'll complete the Terms of Reference, the Research Ethics and Literature Review documents. I will spend the remaining eight weeks before Christmas working on my product design before submitting it on the last week with a product prototype.

In the New Year, I will begin work on the first draft of my report which will draw on work from the planning stage, the product evaluation report and the analysis of the data I've collected. The final section involves producing slides for project presentation which will require draft slides and a practiced presentation in Week 23. The product, final report and the final slides are due in on Week 25 and the presentation will be held.

Appendix B: Ethics Forms

To show consideration to the university's standard of ethics, a set of forms have been completed prior to any work taking place.

Included in these ethics forms are:

A Research Insurance Checklist;

A Risk Assessment Cover Sheet and

A Completed and Signed Ethics Checklist.

Research Insurance Checklist

Overview

Manchester Metropolitan University holds insurance policies to cover claims for negligence arising from the conduct of the institution's normal business. This includes research undertaken by undergraduate and postgraduate students as part of their academic qualification as well as research carried out by staff.

If you are an undergraduate student, postgraduate student or staff researcher at the institution, you must complete all relevant sections of the checklist on the following pages to identify whether your application requires referral to the university's Insurance Officer.

Completing and submitting the checklist will ensure that your research study has appropriate insurance cover in place **before** it begins. Please submit your completed Research Insurance Checklist along with your Ethics Checklist and/or Application for Ethical Approval to your Faculty Research Officer.

Referral to the Insurance Officer

If your research falls into any of the categories listed in Section 2 and/or Section 3 of the checklist, the Faculty Research Officer will send the following information to the Insurance Officer at insurance1@mmu.ac.uk:

- Insurance Checklist
- Ethics Checklist and/or Application for Ethical Approval Form
- Participant Information Sheet(s) (if applicable)
- Participant Consent Form(s) (if applicable)
- Risk Assessment

The Insurance Officer will liaise with the insurers to gain approval. Please note some types of research may require additional insurance, which may incur an additional cost to the Faculty.

Research studies must not commence until insurance and all other relevant authorisations and/or approvals are given.

Travel Insurance

Manchester Metropolitan University has a policy to provide worldwide travel insurance for members of staff and students travelling in connection with their course or on an approved University trip. This includes travel undertaken in connection with undertaking a research study. You must complete the online travel insurance form to register for travel insurance and should do this at least two weeks before your departure date.

Please visit the Financial and Legal webpage for details.

High Risk Countries

Please visit the AIG Travel Guard website to identify whether the overall rating for the country you are travelling to is 'High Risk' or more severe. Please contact your Faculty Research Officer for guidance on accessing the relevant information on the website.

Research Insurance Checklist

ADMINISTRATIVE DETAILS

Lead Investigator Name (Title/Forename/Surname) Click here to enter text.

Contact Email Address Click here to enter text.

Full Title of the Research Click here to enter text.

SECTION 1 – TECHNIQUES, TESTING AND INTERVENTIONS

Does your research study involve:

☐ **Physically invasive techniques?**

This refers to any test in which the skin of the participant is broken or an implement is inserted into any opening of the human body (e.g. eyes, ears, nose, mouth, lungs, stomach, rectum, vagina and urethra) or involves the taking of body samples such as saliva, hair, urine, faeces, sputum, skin, nails, or taking biopsies of any form for any purpose, or any form of scanning such as DEXA scans, Ultrasound scans, MRI, fMRI, CT, or PET scanning.

☐ **Ingestion of food stuffs or drugs?**

This refers to the consumption of any substance which may impact on psychological or physical state. Substances may include but are not limited to food, beverages or drugs.

☐ **Physical testing?**

This refers to any test in which a participant must perform an action resulting in the use of any muscle of the body and/or involves the use of scanning procedures, eye-trackers, mounted body cameras, sensors or electrodes, or the taking of swabs from any cavity of the body, respiratory challenge testing or recording of peak flows, EEG, ECG, Exercise ECG, Treadmill work

☐ **Psychological intervention?**

This refers to any test which purposely alters the mood of the participant or involves administering personality inventories, or any other form of psychological test.

OR

✓ **I confirm that my research does not fall into any of the above categories *(please go straight to Section 3)***

SECTION 2 – CLINICAL TRIALS INSURANCE

Please complete this section only if you ticked one of the boxes in Section 1.

Does your research study involve:

- ☐ **Pregnant persons as participants with procedures other than blood samples being taken from them?**
- ☐ **Children aged five or under with procedures other than blood samples being taken from them?**
- ☐ **Activities being undertaken by the lead investigator or any other member of the study team in a country outside of the UK? *If 'Yes', please refer to the 'Travel Insurance' guidance on Page 1 of this form.***

OR

- ✓ **I confirm that my research does not fall into any of the above categories**

SECTION 3 – OTHER HAZARDS

Does your research study involve:

- ☐ **Working with Hepatitis, Human T-Cell Lymphotropic Virus Type iii (HTLV iii), or Lymphadenopathy Associated Virus (LAV) or the mutants, derivatives or variations thereof or Acquired Immune Deficiency Syndrome (AIDS) or any syndrome or condition of a similar kind?**
- ☐ **Working with Transmissible Spongiform Encephalopathy (TSE), Creutzfeldt-Jakob Disease (CJD), variant Creutzfeldt-Jakob Disease (vCJD) or new variant Creutzfeldt-Jakob Disease (nvCJD)?**
- ☐ **Working in hazardous areas or high risk countries? *Please refer to the 'High Risk Countries' guidance on Page 1 of this form.***
- ☐ **Working with hazardous substances outside of a controlled environment?**
- ☐ **Working with persons with a history of violence, substance abuse or a criminal record?**

OR

- ✓ **I confirm that my research does not fall into any of the above categories**

The MANCHESTER METROPOLITAN UNIVERSITY
Faculty of Science and Engineering

RISK ASSESSMENT COVER SHEET

REFERENCE NUMBER: NPC / /

SCHOOL: Computing, Mathematics & Digital Technology

TITLE OF WORK: Final Year Project: The Game of Life

LOCATION OF WORK: John Dalton Building computing facilities, Home computer.

INTENDED ACTIVITIES:

General use of computers to develop and test software. Method sheets and work schedules not applicable

PERSONS AT RISK

(List names of all individuals (including status e.g. staff/student), and/or unit(s) / course(s) undertaking the activity. For students please indicate course and level, for staff give contact email / phone number):

Undergraduate students.

HAZARDS (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form):

Repetitive Strain Injury – work related upper limb disorder

Back injury resulting from improper posture

Eye strain

Fatigue Stress

Possible risk from 240v electrical mains supply

Are these hazards necessary in order to achieve the objectives of the activity? **Yes**

Hazard Rating (delete as appropriate): **Low**

HAZARDOUS SUBSTANCES/MATERIALS USED AND HAZARD CLASSIFICATION
(appropriate COSHH data sheets / risk assessments must be attached to this form**):**
ALL CONTAINERS OF HAZARDOUS SUBSTANCES SHOULD BEAR CORRECT HAZARD WARNING LABELS.

NAME OF MATERIAL *Please provide also approximate quantity and concentration if applicable.*	**HAZARD CLASS**	**HAZARD LABEL**	**DISPOSAL** *Hazardous materials must not be removed from laboratories. List disposal arrangements for* <u>all materials listed below</u> *in the location where* <u>the work will be carried out:</u>

RISK CONTROL METHODS (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form):

The hazards identified above are controlled by:

Facilities review when laboratories are commissioned
Induction session on H&S given to students by Technical Services Manager
School H&S information given in Student handbook
Posters in laboratories
PAT testing of equipment after three years
Annual H&S inspections

The laboratory workstations, whilst not legally required to be DSE compliant, (the continuous usage is too low to present risk) are fully compliant with current legislation. Monitors and keyboards are adjustable, chairs are adjustable and the lighting designed for both computer usage and associated reading activity. In each laboratory, there is an adjustable desk, suitable for wheelchair users, usually located in the next to the door.

Hazard Rating with Control Methods (delete as appropriate): **Low**

Will any specific **training** *be required (if* YES *give details)?* N/A

Are there any specific **first aid** *issues (if* YES *give details)?* N/A

PROCEDURE FOR EMERGENCY SHUT-DOWN (if applicable):

In the event of fire, flood or other emergency, evacuation of the laboratory would take place and the technical staff would subsequently make an assessment of the necessity of switch-off. As overall system control is vested in a separate server room, there would be little physical harm to any device in directly cutting the power to the mains for each individual lab.

Re-start of the lab may present problems of a technical nature but would not affect the personal safety or health of any individual.

IF OFF-SITE INDICATE ANY OTHER ISSUES (e.g. associated with: individual's health and dietary requirements (obtain off-site health forms for all participating individuals and indicate where this information will be located); social activities, transportation, ID requirements; permissions for access and sampling).

Not applicable – this form applies only to the laboratories listed

	NAME	**STAFF/STUDENT No.**	**DATE**
Originator	[redacted]	[redacted]	02/11/2016
Supervisor	Robert Cherry		
Technical Manager			
Divisional / School Health and Safety Coordinator (p.p. HoS)	N. Costen	01900261	

DATE TO BE REVIEWED BY: September 2017

ETHICS CHECKLIST

This checklist must be completed **before** commencement of **any** research project. This includes projects undertaken by **staff and by students as part of a UG, PGT or PGR programme**. Please attach a Risk Assessment.

Please also refer to the University's Academic Ethics Procedures; Standard Operating Procedures and the University's Guidelines on Good Research Practice

Full name and title of applicant:	
University Telephone Number:	
University Email address:	
Status: All staff and students involved in research are strongly encouraged to complete the Research Integrity Training which is available via the Staff and Research Student Moodle areas	Undergraduate Student ☐ Postgraduate Student: Taught ☐ Postgraduate Student: Research ☐ Staff ☐
Department/School/Other Unit:	
Programme of study (if applicable):	
Name of DoS/Supervisor/Line manager:	
Project Title:	
Start & End date (cannot be retrospective):	
Number of participants (if applicable):	
Funding Source:	
Brief description of research project activities (300 words max):	

	YES	**NO**
Does the project involve NHS patients or resources? If 'yes' please note that your project may need NHS National Research Ethics Service (NRES) approval. Be aware that research carried out in a NHS trust also requires governance approval. Click here to find out if your research requires NRES approval Click here to visit the National Research Ethics Service website To find out more about Governance Approval in the NHS click here	☐	☐
Does the project require NRES approval?	☐	☐
If yes, has approval been granted by NRES? Attach copy of letter of approval. Approval cannot be granted without a copy of the letter.	☐	☐

NB Question 2 should only be answered if you have answered YES to Question 1. All other questions are mandatory.	**YES**	**NO**
1. Are you are gathering data from people?		
For information on why you need informed consent from your participants please click here		
2. If you are gathering data from people, have you:		
a. attached a participant information sheet explaining your approach to their involvement in your research and maintaining confidentiality of their data?		
b. attached a consent form? (not required for questionnaires)		
Click here to see an example of a participant information sheet and consent form		
3. Are you gathering data from secondary sources such as websites, archive material, and research datasets?		
Click here to find out what ethical issues may exist with secondary data		
4. Have you read the guidance on data protection issues?		
a. Have you considered and addressed data protection issues – relating to storing and disposing of data?		
b. Is this in an auditable form? (can you trace use of the data from collection to disposal)		
5. Have you read the guidance on appropriate research and consent procedures for participants who may be perceived to be vulnerable?		
a. Does your study involve participants who are particularly vulnerable or unable to give informed consent (e.g. children, people with learning disabilities, your own students)?		
6. Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited (e.g. students at school, members of self-help group, nursing home residents)?		
Click for an example of a PIS and information about gatekeepers		
7. Will the study involve the use of participants' images or sensitive data (e.g. participants personal details stored electronically, image capture techniques)?		
Click here for guidance on images and sensitive data		
8. Will the study involve discussion of sensitive topics (e.g. sexual activity, drug use)?		
Click here for an advisory distress protocol		
9. Could the study induce psychological stress or anxiety in participants or those associated with the research, however unlikely you think that risk is?		
Click here to read about how to deal with stress and anxiety caused by research procedures		
10. Will blood or tissue samples be obtained from participants?		
Click here to read how the Human Tissue Act might affect your work		
11. Is your research governed by the Ionising Radiation (Medical Exposure) Regulations (IRMER) 2000?		
Click here to learn more about IRMER		
12. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?		
Click here to read about how participants need to be warned of potential risks in this kind of research		
13. Is pain or more than mild discomfort likely to result from the study? Please attach the pain assessment tool you will be using.		

Click here to read how participants need to be warned of pain or mild discomfort resulting from the study and what do about it.		
14. Will the study involve prolonged or repetitive testing or does it include a physical intervention?	☐	☑
Click here to discover what constitutes a physical intervention and here to read how any prolonged or repetitive testing needs to managed for participant wellbeing and safety		
15. Will participants to take part in the study without their knowledge and informed consent? If yes, please include a justification.	☐	☑
Click here to read about situations where research may be carried out without informed consent		
16. Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	☐	☑
Click here to read guidance on payment for participants		
17. Is there an existing relationship between the researcher(s) and the participant(s) that needs to be considered? For instance, a lecturer researching his/her students, or a manager interviewing her/his staff?	☐	☑
Click here to read guidance on how existing power relationships need to be dealt with in research procedures		
18. Have you undertaken Risk Assessments for each of the procedures that you are undertaking?	☐	☑
19. Is any of the research activity taking place outside of the UK?	☐	☑
20. Does your research fit into any of the following security sensitive categories: • commissioned by the military • commissioned under an EU security call • involve the acquisition of security clearances • concerns terrorist or extreme groups If Yes, please complete a Security Sensitive Information Form	☐	☑

I understand that if granted, this approval will apply to the current project protocol and timeframe stated. If there are any changes I will be required to review the ethical consideration(s) and this will include completion of a 'Request for Amendment' form.

☑ have attached a Risk Assessment
☑ have attached an Insurance Checklist

If the applicant has answered **YES** to **ANY** of the questions **5a – 17** then they must complete the MMU Application for Ethical Approval.

Signature of Applicant: [redacted] Date:_____(DD/MM/YY)

***Independent Approval* for the above project is (please check the appropriate box):**
Granted
☐ I confirm that there are no ethical issues requiring further consideration and the project can commence.

Not Granted
☐ I confirm that there are ethical issues requiring further consideration and will refer the project protocol to the Faculty Research Group Officer.

Signature:_____Date:_____(DD/MM/YY)

Print Name: _____Position:_____
Approver: Independent Scrutiniser for UG and PG Taught/ PGRs RD1 Scrutiniser/ Faculty Head of Ethics for staff.