AI in Games

COMP 1864 Project Report

001331157	Roberto Scialpi	rs9864p@gre.ac.uk

Content

Overview

The idea for this project originated from curiosity and experimentation, specifically from exploring how Cellular Automata behave under different conditions.
This led to the question: what happens if the neighbourhood definition changes while keeping the same Birth/Survival rules?

To investigate this, the project explores how Cellular Automata behaviour changes when altering neighbourhood selection and rule sets within a grid based system. Inspired by Conway's Game of Life, the project investigates the effects of non-standard neighbourhoods and introduces an alternative rule system, referred to as BoB's Game of Life.

The final stage combines both systems within the same grid to observe emergent behaviour and interaction between competing rule sets.

Concept Design

The concept of this project is to explore how changes to neighbourhood selection and rule sets affect the behaviour of Cellular Automata systems. The project is inspired by Conway's Game of Life, which traditionally uses a Moore neighbourhood and a fixed Birth/Survival rule set (B3/S23) to produce complex emergent patterns from simple rules.

The first stage of the project investigates how altering the neighbourhood check alone impacts the simulation. By applying alternative neighbourhood configurations while keeping the original Conway rule set, the system demonstrates how spatial relationships influence pattern formation and stability.

Building on this, the project introduces an alternative rule system, referred to as BoB's Game of Life, which uses different Birth and Survival conditions (B3456/S2468). This creates a second Cellular Automata behaviour that evolves independently from Conway's rules, allowing for direct comparison between the two systems.

The final stage of the project combines both rule sets within the same grid. Conway and BoB cells coexist and interact, with conflict resolution handled through priority and neighbourhood influence. This setup allows observation of how competing rule based systems evolve when sharing the same environment, highlighting emergent behaviour, dominance, and interaction patterns.

AI Techniques Used

This project applies two Artificial Intelligence techniques studied during this module: two dimensional (2D) Cellular Automata and Finite State Machines (FSMs). Both techniques rely on local rule evaluation and decentralised decision making, allowing complex behaviour to emerge from simple interactions.

2D Cellular Automata

The primary technique used in this project is 2D Cellular Automata, where the simulation space is represented as a two-dimensional grid of discrete cells. Each cell exists in one of two possible states, alive or dead, and updates its state at each simulation step based on the state of neighbouring cells.

The project is inspired by Conway's Game of Life, which traditionally uses:

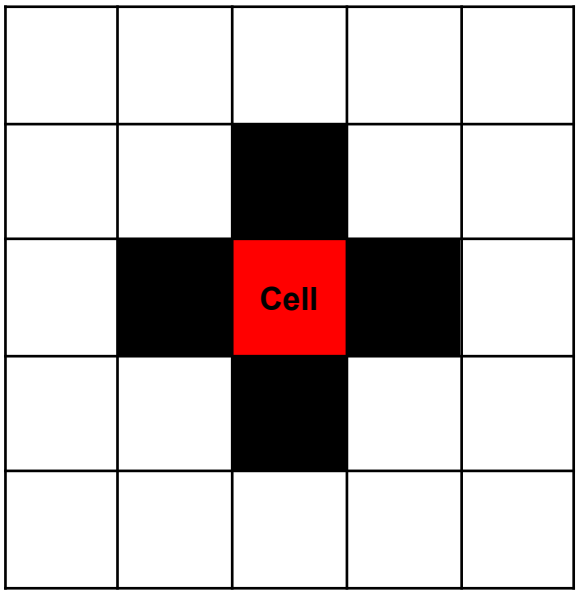
- A 2D grid
- A Moore neighbourhood
- A fixed Birth/Survival rule set (B3/S23)

This project extends the traditional model by:

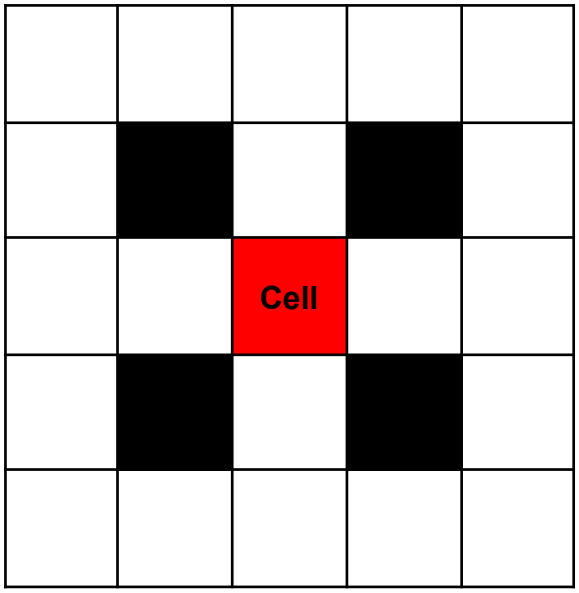
- Experimenting with alternative 2D neighbourhood configurations (Plus, Cross, directional 2x3 block)
- Maintaining synchronous updates across the grid
- Allowing multiple Cellular Automata systems to operate within the same 2D environment

These are the alternative configurations:

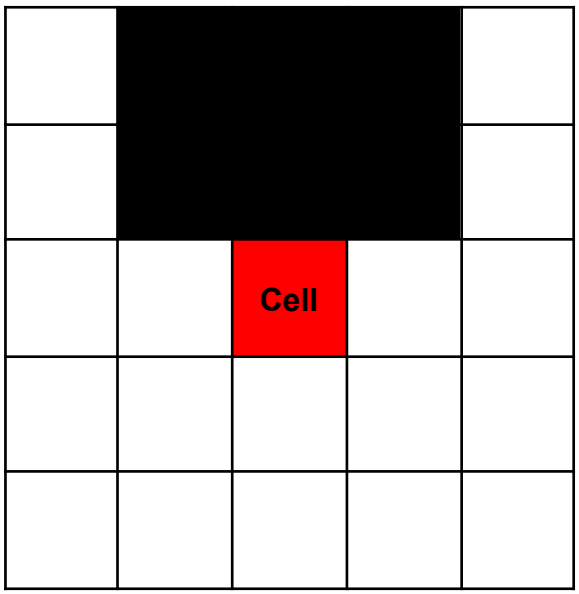
- **Plus**

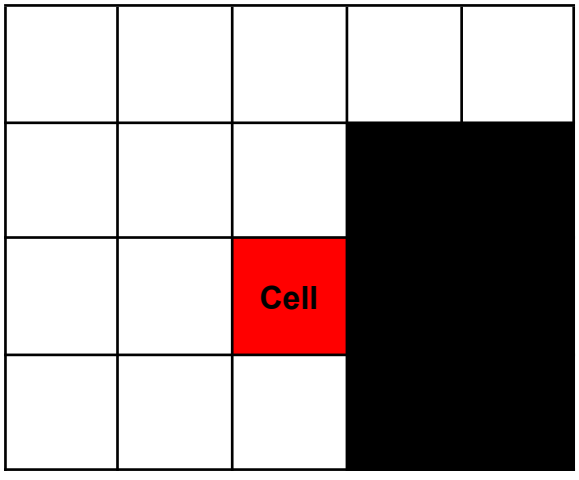


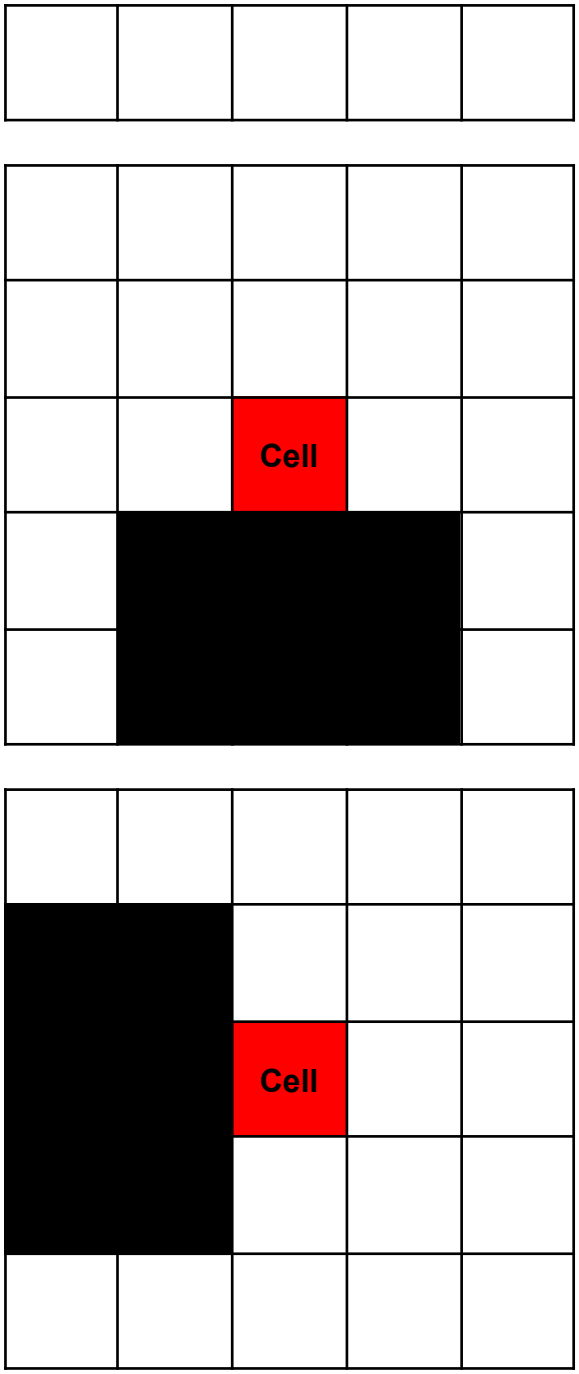
- **Cross**



- **Directional 2x3 Block**







By modifying neighbourhood definitions while keeping the grid structure constant, the project demonstrates how spatial relationships in a 2D space significantly influence pattern formation and system stability.

Finite State Machines (FSM)

Finite State Machines are applied implicitly at the level of individual cells within the 2D grid. Each cell functions as a simple FSM with two states:

- Alive
- Dead

State transitions are governed by local environmental conditions, including:

- The cell's current state
- The number of alive neighbouring cells
- The active rule set (Conway or BoB)

Rather than using explicit state diagrams, FSM behaviour is implemented through conditional logic within the Birth and Survival checks. This mirrors common game AI practices, where FSM logic is embedded directly into update rules rather than represented as standalone structures.

Technical Implementation

The system is divided into two main components:

- **Grid Manager**: responsible for grid creation, cell storage, rendering
- **Cellular Automata**: responsible for evaluating rules, neighbourhood checks, and advancing the simulation through generations.

Both scripts contain functionalities for the user input.

This separation ensures that the simulation logic remains independent from rendering, improving readability, extensibility, and maintainability.

The Grid

The grid is represented using 2D int arrays, where each cell can be either alive (1) or dead (0). To support the coexistence of multiple rule sets, separate state grids are maintained:

- One grid for Conway's Game of Life
- One grid for BoB's Game of Life

Each rule set has both a current and a next state array, allowing all updates to be calculated simultaneously before advancing the simulation. This avoids update-order bias and ensures consistent generation transitions.

Cells are instantiated as individual GameObjects in the scene, and their visual state is updated based on the active grid data. This is handled by the Grid Manager.

Cellular Automata

The Cellular Automata script is responsible for evaluating neighbourhoods, applying rule sets, and advancing the simulation through discrete generations. It operates independently from rendering, interacting with the grid only through state arrays provided by the Grid Manager.

Each simulation step is processed synchronously. For every cell in the grid, the system evaluates the surrounding neighbours, determines the next state according to the active rule set, and stores the result in a separate “next state” grid. Once all cells have been evaluated, the grids are swapped to advance the simulation to the next generation.

Neighbourhood Evaluation

Neighbourhood selection defines which surrounding cells influence a cell’s next state. The project supports multiple neighbourhood configurations, including:

- Moore (8 surrounding cells)
- Plus (up, down, left, right)
- Cross (diagonal neighbours only)
- Directional 2x3 region (Up2, Down2, Left2, Right2)

Neighbourhoods are implemented using predefined offset arrays, allowing neighbour positions to be calculated dynamically. This makes the system extensible, as additional neighbourhood types can be added without modifying the core simulation loop.

Boundary handling is managed using wrap-around logic, treating the grid as an infinite loop space. This prevents edge cases and ensures consistent behaviour across the entire grid.

Rule Set Evaluation

Two distinct rule sets are implemented within the system:
Conway’s Game of Life
Conway’s rule set follows the traditional Birth/Survival model:

- **Birth**: A dead cell becomes alive if it has exactly three alive neighbours
- **Survival**: A living cell remains alive if it has two or three alive neighbours

This rule set is applied independently of neighbourhood type, allowing experimentation with non-standard neighbourhoods while maintaining Conway’s original logic.

BoB's Game of Life

BoB's Game of Life introduces alternative Birth/Survival rule sets:

- **Moore Birth**: A dead cell becomes alive if it has between three and six alive neighbours
- **Moore Survival**: A living cell survives when there is an odd number of alive neighbours
- **Plus Birth**: A dead cell becomes alive if it has exactly two alive neighbours
- **Plus Survival**: A living cell survives if it has one or two alive neighbours
- **Cross Birth**: A dead cell becomes alive if it has exactly one alive neighbours
- **Cross Survival**: A living cell survives if it has one or two alive neighbours
- **Directional 2x3 Block Birth**: A dead cell becomes alive if it has two or three alive neighbours
- **Directional 2x3 Block Survival**: A living cell survives when it has between two and four alive neighbours

This rule set creates a significantly different evolutionary behaviour, often producing denser, less predictable patterns compared to Conway's system.

Rule selection is controlled via the user interface, allowing Conway, BoB, or both systems to be activated during runtime.

Multi-Rule Interaction in a Shared 2D Space

In the final stage of the project, both Conway and BoB rule sets operate simultaneously within the same grid. Rather than merging the rules, each system maintains its own state arrays and evaluates its next generation independently.

For each cell:

- Neighbour influence is calculated separately for Conway and BoB
- Each rule set produces a candidate next state
- A conflict resolution step determines which rule set, if any, occupies the cell

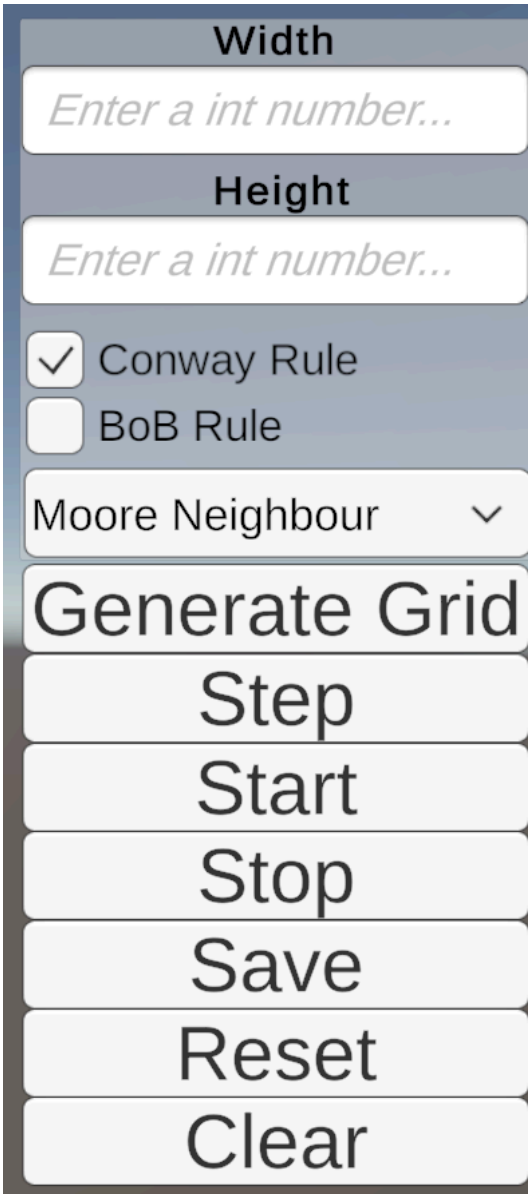
When both systems attempt to occupy the same cell, priority is resolved based on local neighbourhood influence. The system compares the number of neighbouring Conway cells and BoB cells, granting control to the rule set with the greater local presence. In the case of equal influence, the cell dies.

This design allows each system to evolve naturally until interaction occurs, producing emergent behaviour.

User Interaction and Control

The system includes UI controls that allow the user to:

- Deciding the Grid size (default is 10x10)
- Select the active rule set(s)
- Change neighbourhood configuration
- Step through generations or run the simulation continuously
- Stop the simulation
- Save the current state while not simulating (it saves at the beginning the first simulation)
- Reset to the saved set up
- Clear the whole grid
- Edit cell states directly via left mouse click (selected rule set determines what cell you activate, if both active, BoB is the dominant check box)



Core Problems

The first core problem was understanding the fundamentals of Cellular Automata and the underlying principles of Conway's Game of Life. As this was a new concept and largely unexplored territory for the project, it was essential to first understand how simple local rules can lead to complex emergent behaviour.

This was addressed by studying the module material alongside external references and existing Conway's Game of Life implementations. This initial research phase established a clear understanding of neighbourhood checks, Birth/Survival rules, and generation updates, which formed the foundation for all subsequent development.

Once a working implementation of Conway's Game of Life was achieved, the main challenge became extending the system to support multiple Cellular Automata behaviours within the same environment. The initial approach was to create a secondary class to handle a different behaviour, resulting in two independent systems: Conway's Game of Life and BoB's Game of Life.

During the development of BoB's Game of Life, several approaches were considered. One early idea involved using large lookup tables or dictionaries containing binary state transitions, allowing outcomes to depend not only on the number of alive neighbours but also on their relative positions. While this approach offered fine grained control, it quickly became complex and difficult to maintain within the project scope. As a result, a design decision was made to retain the core structure of Conway's Game of Life and instead introduce alternative Birth and Survival conditions, combined with directional neighbourhood checks. This allowed BoB's Game of Life to remain expressive while keeping the system manageable.

The most significant technical challenge arose when attempting to make both rule sets operate within the same grid. Initially, grid generation and simulation logic were tightly coupled, making it difficult for multiple systems to interact with the same data safely. This led to a refactor where grid creation was managed by a Grid Manager, while all rule evaluation and simulation logic were handled by the Cellular Automata system.

Further issues emerged when managing references between multiple scripts operating on shared data. To resolve this, the design was simplified by consolidating rule evaluation into a single Cellular Automata script, with rule selection handled dynamically. This ensured that both Conway and BoB behaviours could be evaluated consistently while operating on shared grid data.

The final problem was conflict resolution when both rule sets attempted to occupy the same cell. This was solved by introducing a majority based neighbourhood influence system. When both Conway and BoB produced a living candidate for the

same cell, the system compared the number of neighbouring cells belonging to each rule set. The rule set with the greater local influence gained control of the cell in the next generation.

Summary

This project explored how Cellular Automata behaviour changes when neighbourhood definitions, rule sets, and interactions are modified within a shared 2D grid. Starting from a standard implementation of Conway's Game of Life, the system was extended to support alternative neighbourhoods, a second rule system (BoB's Game of Life), and finally interaction between both rule sets within the same environment. The results highlight how small local rule changes can lead to significantly different global and emergent behaviours.

A key limitation of the system is performance, as both rule sets are evaluated per cell each generation, which limits scalability for larger grids. Additionally, the majority-based conflict resolution can cause unstable boundary behaviour, where cells frequently switch ownership. While acceptable for experimentation, this reduces long-term pattern stability. BoB's behaviour is also less expressive than originally planned, as more complex directional and FSM-based rules were simplified to fit the scope.

Future improvements could include performance optimisation, richer FSM-style behaviours for BoB, additional competing rule sets, and analytical tools such as population graphs. Overall, the project met its objectives and provided valuable insight into multi-rule AI systems and emergent behaviour through iterative development and experimentation.

References

OpenAI (2025) ChatGPT (GPT-5.2) [Large language model]. Used for spell check and to improve clarity and grammar. Available at: https://chat.openai.com/ [Accessed: 31 December 2025]