Game of Life Implementation with Extension

COMP3106
Qihang Peng
101013564

Introduction

Background

“Game of Life” is a computer program designed by John Horton Conway of Cambridge University. Martin Gardner (1914-2010), the American interesting mathematics master, introduced Conway's life game to the vast number of offenders outside the academic world through Scientific American magazine, which attracted a large number of people from all walks of life for a while. At that time, the cellular automata topic attracted the attention of scientists.

"Game of Life " is not simply a game to play. In fact, "life game" is a "cellular automaton", that is, a group of colored cells on the grid, which evolves gradually over time according to a set of rules that define the state of adjacent cells. Its inventor Conway called it "zero player, no ending".

In 1970, the basic rules of the "life game" were published in the column of Scientific American magazine. After being implemented by computer programs, it became popular in the 1970s.
All rules of the "life game" are deterministic, but the evolution of the game gives people a sense of autonomy.

Nowadays, "life game" is not only an exercise that almost all people who learn programming and algorithms can do, but also a question full of philosophy and wisdom:

- What kind of living space is most suitable for us?
- How will interactions between individual organisms affect the distribution and evolution of populations?

Prior work

Game of life is a zero-player game. It consists of a two-dimensional rectangular world, in which each grid lives a living or dead cell. The life and death of a cell at the next moment depends on the number of living or dead cells in the adjacent eight squares. If there are too many living cells in the adjacent grid, this cell will die at the next moment due to lack of resources; On the contrary, if there are too few living cells around, the cell will die because it is too lonely. In practice, players can set the number of living cells around them when it is suitable for the survival of this cell. If the number is set too high, most of the cells in the world will die because they cannot find too many living neighbours, until the whole world has no life; If the number is set too low, the world will be full of life without any change.

In practice, this number is generally 2 or 3; In this way, the whole life world will not be too desolate or crowded, but a dynamic balance. In this way, the rule of the game is: when there are 2 or 3 living cells around a square, the living cells in the square will continue to live at the next moment; Even if there are no living cells in the grid at this moment, living cells will be "born" at the next moment.

In this game, you can also set some more complex rules. For example, the current grid situation is not only determined by the father generation, but also considered by the grandfather generation. Players can also act as the "God" of the world, and randomly set the life or death of a certain grid cell to observe the impact on the world.

In the process of the game, the disordered cells will gradually evolve various delicate and tangible structures; These structures often have good symmetry, and each generation is changing its shape. Some shapes have been locked and will not change from generation to generation. Sometimes, some formed structures will be destroyed due to the "invasion" of some disordered cells. But shape and order can often emerge from chaos.

This game has been implemented by many computer programs. Many hackers in the Unix world like to play this game. They use characters to represent a cell and evolve on a computer screen. A famous example is that the GNU Emacs editor includes such a small game.

Significance of the project

In the various prevalent implementations of Game of Life, most of them were all based on the classic theory. For example, they abide following classical rules:

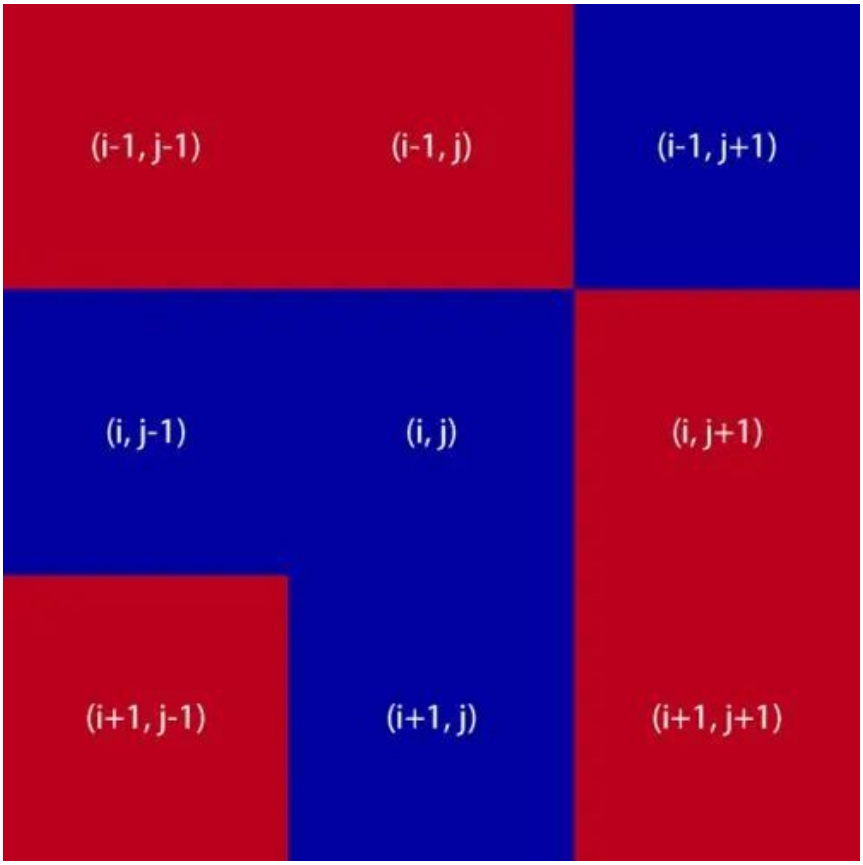
There is a cell in each square of the living space, and the adjacent eight squares around it are called neighbour cells.

1. When the current cell is in an annihilation state, when there are 3 living cells around, the cell will become a living state after iteration (simulated reproduction).
2. When the current cell is in the survival state, when less than 2 neighbouring cells survive, the cell becomes an annihilation state (the number is rare).
3. When the current cell is in a living state, when there are more than 3 living cells around, the cell becomes an annihilation state (too many).
4. When the current cell is in a living state, when there are 2 or 3 living cells around, the cell remains as it is.

In short, living cells are regarded as' 1 'and dead cells as' 0'. The accumulation of eight neighbours and Sum determine the state of the next round:

Reproduction: Dead Cell=0, if Sum=3, Cell=1.
"Rare": Live Cell=1, if Sum<2, Cell=0.
"Too many": Live Cell=1. If Sum>3, Cell=0.
"Normal": Live Cell=1. If Sum=2 or 3, Cell=1.

The given cells (i, j) in the simulation are accessed using a two-dimensional array [i] [j], where i and j are subscripts of rows and columns, respectively. The game is built in a grid of 9 squares, each cell has 8 adjacent cells, as shown in the following figure.In a given time period, the value of a given cell depends on the state of its neighbours in the previous time period.



The rules are classic, however, they have limitations. Firstly, they only consider the surrounding neighbours' life status; secondly, they fail to take into account such as natural elements such as resources or weather; thirdly, they didn't incorporate the rules that in a mimic living society, there should always be the rule of natural selection. For example, herbivores should consume plants/ grass and carnivore consume herbivores.

Methods

Rule-based system

Python is the main programming language chosen to implement this extended version of the Game of life and this implementation leverages the Expert system, particularly, the rule-based system rationale (Xu et al., 2020). Expert system is one of the main research fields of artificial intelligence, which is a computer program for business processing based on industry rules(Fai et al., 2015). These rules are derived from industry knowledge and are used to describe the actions to be performed under specific conditions, and define the impact of related actions on data. Therefore, the expert system can use its reasoning ability to draw conclusions or perform relevant analysis tasks. In the expert system, the knowledge needed to solve the problem is stored in the knowledge base as a rule set, forming a knowledge base system.

One of the representative products of the medical expert system is MYCIN, which is used to help doctors diagnose and treat bacterial infections. In other industries, expert systems are also widely used to analyze geophysical data and find oil and metal deposits in the geological exploration industry. In addition, the expert system is also used in financial investment, banking and telecommunications industries.

1. Advantages of rule-based expert system

In rule-based expert system, knowledge base and rule execution component are its core modules, which are called expert system kernel. The expert system kernel and the rules it executes are relatively independent. This means that the expert system kernel can be used to deal with problems in different industries without fundamental changes. At the same time, adding or modifying rules in the expert system will not affect the function of the system kernel.

The language used to describe rules is closely related to the language used by industry experts to describe problem solutions. When industry experts use language to build a rule, they create a record of business knowledge, which can be shared among industry personnel. The collection and management of knowledge by expert system ensures the continuity of system operation, and the use of rules can be ensured even in the absence of industry experts.
In addition, managing rules by the system can ensure that the knowledge base is easier to update without the participation of programmers, which reduces the cost of software operation and maintenance, and also ensures that changes in rules are based on the needs of industry personnel.

Finally, the knowledge stored and retrieved by the expert system is far greater than that memorized by a single person, and there will be no errors. The rules provided by the expert system must also be created by industry experts through precise modelling.

2. Rule based expert system architecture

Rule-based expert system includes two main parts: knowledge base and expert system kernel. A knowledge base is a collection of rules, which can be a file system in the form of metadata encoding, or a rule base stored in a relational database. The expert system kernel is a logical module for creating, editing, and executing rules.

User interface

The user interface is used to process service requests from users or from other systems. The user interface forwards these service requests to the respective kernel processing module. For example, if an industry expert requests to create or edit rules, the user interface will forward the request to the knowledge base editor module.

Knowledge Base Editor

The knowledge base editor can be a plain text editor, a graphical editor, or a mixture of the two. This module is used to add and modify rules for the knowledge base.

Rule Converter

Rules cannot be directly executed. They must be converted from human readable forms to forms that can be parsed by the rule engine. This transformation of rule form is realized by rule converter.

When a rule is converted from its original form to a machine-readable form, it needs to parse the text format to obtain the data structure and form an abstract syntax tree (AST). AST is an abstract data type, which aims to enable the rule engine to parse rules more simply and effectively. This abstract data type is very expressive and can be used to support the creation of very complex and powerful rules. In order to store in the knowledge base, the rule AST should be transformed into an equivalent physical storage form. The way of information description in the knowledge base depends on the storage technology of the knowledge base. Relational databases can provide a very convenient and effective way to store metadata. The metadata here is equivalent to the AST in the actual case and can be stored in the table set. The specific model used to store metadata must be able to support the use of data queries to quickly build AST.

Rules Engine

The rule engine (called the inference engine in AI technology) is responsible for executing the rules of the knowledge base. The rule engine module retrieves rules from the knowledge base, converts the rules into AST, and then submits the relevant AST results to the rule interpreter for execution. The rule engine translator traverses the AST and executes the actions specified in the rules.

A case study: MYCIN

MYCIN is an expert system developed by Stanford University for the diagnosis and treatment of bacterial infections. MYCIN system was developed in the early 1970s. As an expert consultation system, MYCIN has solved a series of technical problems in the application of expert systems, which has an important impact on the development of expert systems. Most of the current expert systems are rule-based expert systems designed and developed with reference to MYCIN. MYCIN is considered a classical system.

The MYCIN system has never been applied in practice, not because of its performance, but because the system integration technology at that time could not support the system operation. MYCIN is a stand-alone operation system, which requires users to input all information related to patients' diseases, which greatly limits the application and development of the system.

The great influence of MYCIN system also lies in the powerful functions of its knowledge expression and reasoning scheme. However, with the development of MYCIN, there is also an

insurmountable difficulty, namely, extracting the required knowledge from the work field of industry experts and transforming it into a rule base, which forms a knowledge acquisition bottleneck.

The clinical consultation process of MYCIN system simulates the diagnosis and treatment process of human beings. See Figure 3 for the detailed process. Physician users (non experts) submit their patient data, receive feedback on clinical recommendations, and feedback information through the internal explanation mechanism. For example, question answering and advisory advice fed back via a generic question solver or an inference state checker. All decisions are based on the domain knowledge required by industry experts, that is, static knowledge. A group of computer programs, namely rule parsers, use these knowledge and patient data to form clinical conclusions and treatment suggestions through logical analysis.

Figure 3. MYCIN System Composition and Information Transfer

The design goal of MYCIN system has three aspects: ① to put forward useful suggestions in clinical practice; ② Explain the decision when necessary; ③ Obtain industry knowledge directly from industry experts. Correspondingly, MYCIN system consists of three related parts:
(1) The consulting system makes use of the knowledge base and patient data entered by doctors to form diagnostic suggestions.
(2) The explanation system includes a general question solver and a reasoning state checker. During the consultation process, when the user asks for the reason and reason for the conclusion, he explains the reasoning process used.
(3) The knowledge acquisition system is used to support industry experts to update the static knowledge base of MYCIN system, without requiring industry experts to master computer programming.

Actual implementation

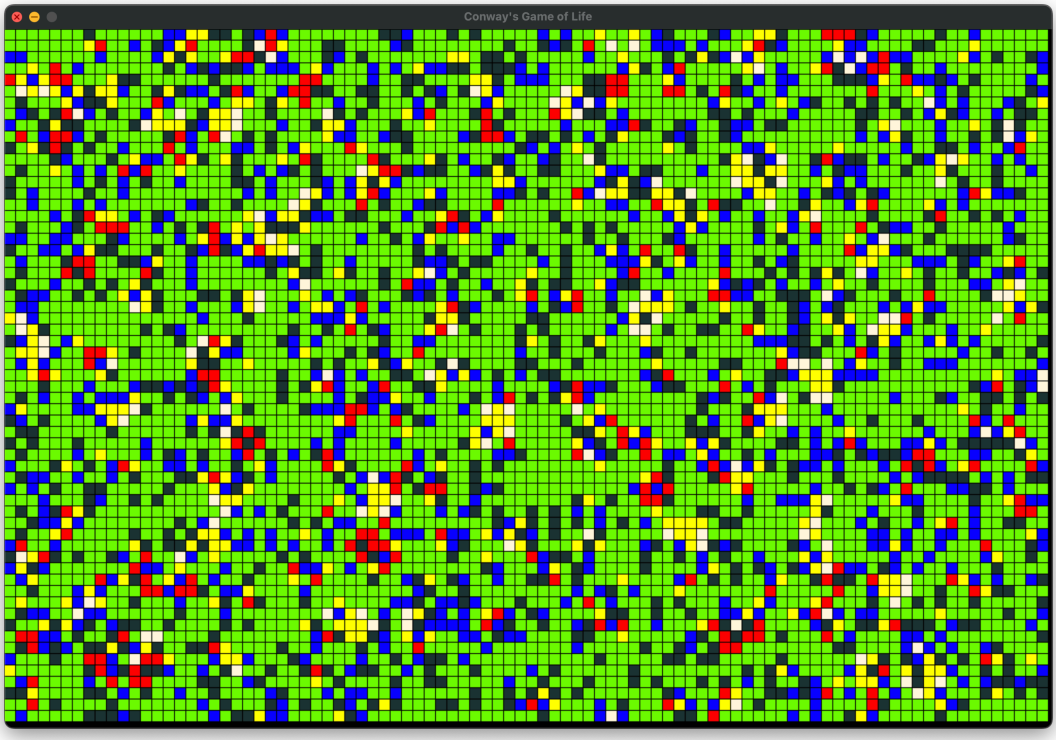
There were some additions to the game that were considered in project proposal but not actually implemented in the final implementation. Some of them were experimented but removed, some were never implemented after some thoughts. Firstly, weather system was removed, due to how fast everything reproduced in the spring, then in the winter resources were consumed instantly resulting in a dead planet. Rules of consuming were also removed, because after a lot of experiments, it was found that it was extremely difficult to balance the consume rate and reproduction rate of the spices and resources. Spices movement were never implemented because movement is the same as dying out on one side of a pack and reproduce on the other side of the pack. Due to time limitation, player control of the initial world is not implemented. The first part of the implementation centred around the actual entity to represent a life in the extension. Previously, all major implementation of game of life simply use a Boolean value of True/False to represent the life status of on object on the grid. The extension involves creating a new class to represent a cell, and use an instance variable to represent its kind:
1. Herbivore
2. Carnivore
3. Grass
4. Barren
5. Water
6. Inactive

Notice that there is an inactive state for the cells. Its purpose is that, logically, one type of cell

should not turn into another type of cell. Inactive cells(dead cells) act as the space for cells to reproduce.
Implemented rules of cells:
Herbivore and Carnivore implements the rule of living cells from the classic Game of Life, only that they have to live around the resources they require(herbivore needs grass and water, carnivore needs herbivore and water).
Grass, barren and water cells are similar to dead cells except that they cannot be reproduced on.
Inactive cells are the dead cells that can be reproduced on.
Generate world:
Each cell is generated completely randomly. This is where it could be improved and will be mentioned below.

Results

After implementing, experimenting and adjusting rules, the final result of the implementation game is produce an acceptable result. After a few cycles, the world randomly generated can reach a balance state where changes can be see in the world but they are relatively stable(looped). The reason there are no spaceship(a type of pattern in classic game of life, that seems to be moving around the world) is the existence of the grass barren and water. Since they cannot be reproduced on, they act like a barrier that prevent the living cells to move.
The below picture is a screen shot of the game in the balance state. Yellow represents herbivore, red represents carnivore, green represents grass, white represents barren, blue represents water, and the dark cells are inactive cells.
Closing the game also will display the average number of herbivore and carnivore cells. It is interesting to see that it is almost always the case that there are more herbivore than carnivore,



which is quite true in real life in a wild environment.

Discussion

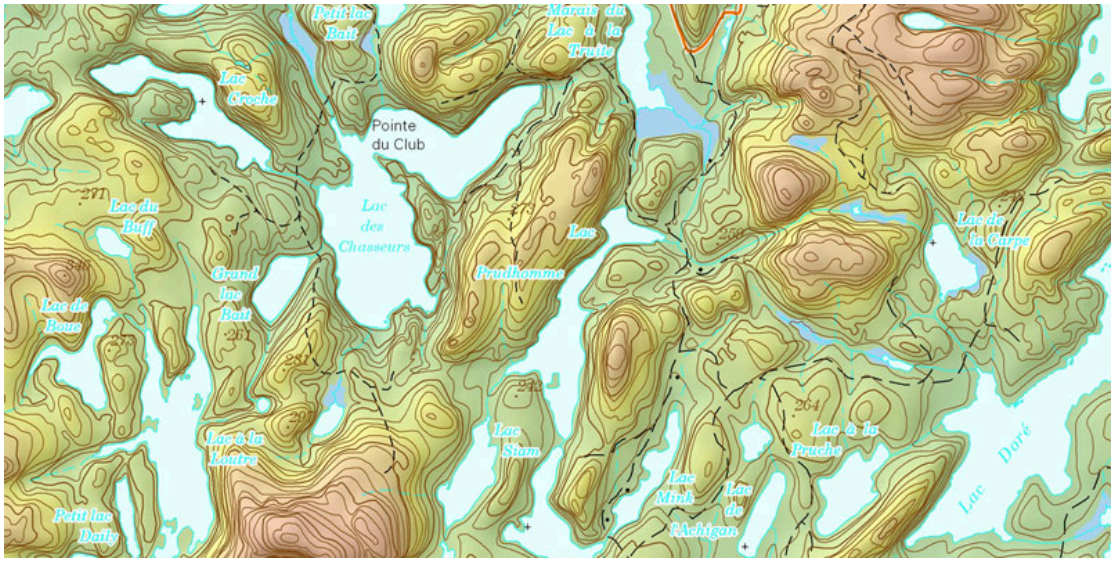
The extension mainly involves the creation of a standalone object “Cell” to better specify the species of the cell, and also the extension of the classical rule set to cater for the interaction of these extended species.

The current result looks promising, since it captures come intrinsic natural rules out there in the wild. However, there are some apparent limitations. First of all, the whole Game of life implementation is still based on looping structure and each round the main logic will scan each cell of the two-dimensional grid. This type of program logic is resource-hog and inefficient and not very aligning with the real nature. A more realistic implementation would be using a observer design pattern, with each cell register a event, with random timer, so that at any random time, a herbivore will check to see if there are plants/grass near by and starts eating, so that the whole virtual world will be rendered much more realistic. Secondly, from the AI perspective and pertaining to the expert system aforementioned, there should be a knowledge base, database, reference engine, interpretation engine, and a GUI. Currently, the implementation has a GUI, a reference engine. There is no standalone knowledge base, database or interpretation engine. Currently, we embed such knowledge in the lengthy if-else branching within the Python code itself, a future improvement could be made to include at least an knowledge base, which will consist of any number of customized rule, in the form of “If.. Else…” to capture a more realistic production and consumption rule of the virtual world we depicted. So that we could eliminate the knowledge from the Python code itself. Thirdly, with the expert system, a further enhancement could be added to the current system in providing prediction for each cell (Naser & Al-Bayed, 2016), for example, if a neighbour cell is likely to be a carnivore, then a suggestion move will be provided to the current herbivore cell. However, this improvement requires a radical change of the whole GUI since it requires cell movement.

Improvement

Although, this game of is quite a simple system to implement, there are still room for the game to improve. As mentioned above, each cell is generated completely randomly, which is not very realistic in real world, especially the environment. For example, river could be a stream of blue line instead of separated dots in the world. There is a solution for this, which can also solve the problem of lack of player interaction(setting initial state of the world). Player could be given the option to upload a picture of an environment similar to the picture below:

Source: https://www.macarte.ca/topographic.php



Then by reading the colours of the pixels, the program generates a world based on the RGB value of the pixels.

Summary

The classical game of life has many implications to us, from both the underlying principle to how to extend it with AI expert system to better mimic the ecosystem of the virtual world. Current extension has achieved certain improvement in taking consideration of specifies, resources and interaction/consumption between them. There is still much room possible for improvement, and especially with an aim for making it conform to a solid expert system implementation.

References

Naser, S.S., & Al-Bayed, M.H. (2016). Detecting Health Problems Related to Addiction of Video Game Playing Using an Expert System.

Xu, X., Yan, X., Sheng, C., Yuan, C., Xu, D., & Yang, J. (2020). A Belief Rule-Based Expert System for Fault Diagnosis of Marine Diesel Engines. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50, 656-672.

Fai, T.C., Wahidin, L.S., Khalil, S.N., Tamaldin, N., Hu, J., & Rauterberg, G. (2015). THE APPLICATION OF EXPERT SYSTEM: A REVIEW OF RESEARCH AND APPLICATIONS.