A Practical Approach to Cellular Automaton

Zeyu Chen, Xibei Zhang, Yijun Zhang

May 3, 2018

Demo Link:https://www.youtube.com/watch?v=hXFlM9DHgwU

1 Introduction

1.1 Concepts of cellular automaton

A cellular automaton is a discrete model studied in computer science, mathematics, physics, complexity science, theoretical biology and microstructure modeling [1].
The concept of cellular automaton was discovered in the 1940s by Stanislaw Ulam and John von Neumann. A cellular automaton is a model of a system of 'cell' objects with the following characteristics [2]:

- The cell live on a **grid**. The simplest grid would be one-dimensional which is a line of cells.
- Each cell has a **state**. The number of state possibilities is typically finite. The simplest example has the two possibilities of 0 and 1.
- Each cell has a **neighborhood**. This is typically defined as a list of adjacent cells.

A cellular automaton consists of a regular grid of cells, each in one of a finite number of states. The grid can be in any finite number of dimensions. An initial state is selected by assigning a state for each cell. A new generation is created according to some fixed rule(a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the state of the cells in its neighborhood [1].
Game of Life is one of the 2D cellular automaton devised by the British mathematician John Horton Conway in 1970. The 'game' is a zero-play game which means that its evolution is determined by its initial state instead of further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves or creating patterns with particular properties by players. [2]

1.2 Aim of Game-of-Life

In cellular automaton, simple rules of evolution result in great and diverse complexity of behavior. The complex models in real life can be expressed succinctly via simple cellular automaton. Cellular automaton can reproduce many patterns we see in nature [3].
Models of cellular automaton are still being searched by people. In this project of game of life, we provide libraries for users to select the mode they want to use. Also, we provide brush for users to decide the initial modes they want to use for game of life.
The approaches and tools we designed in this project help users better study and learn the patterns of cellular automaton.

2 Approach

2.1 Interface Design

This project has 4 sections which are Random Mode, Seed library, Customized Mode and Stats respectively. Random Mode allow users to start the game with a random initial probability. Seed library contains some libraries that we pre-inserted. Users can select the seed they want. Customized mode contains a brush which allows users to draw the initial pattern on their own and a lib with mode they want to start with. Stats will show the parameters including FPS, time, number of live cells and generation of the current game.
The interface design flowchart is shown as below:

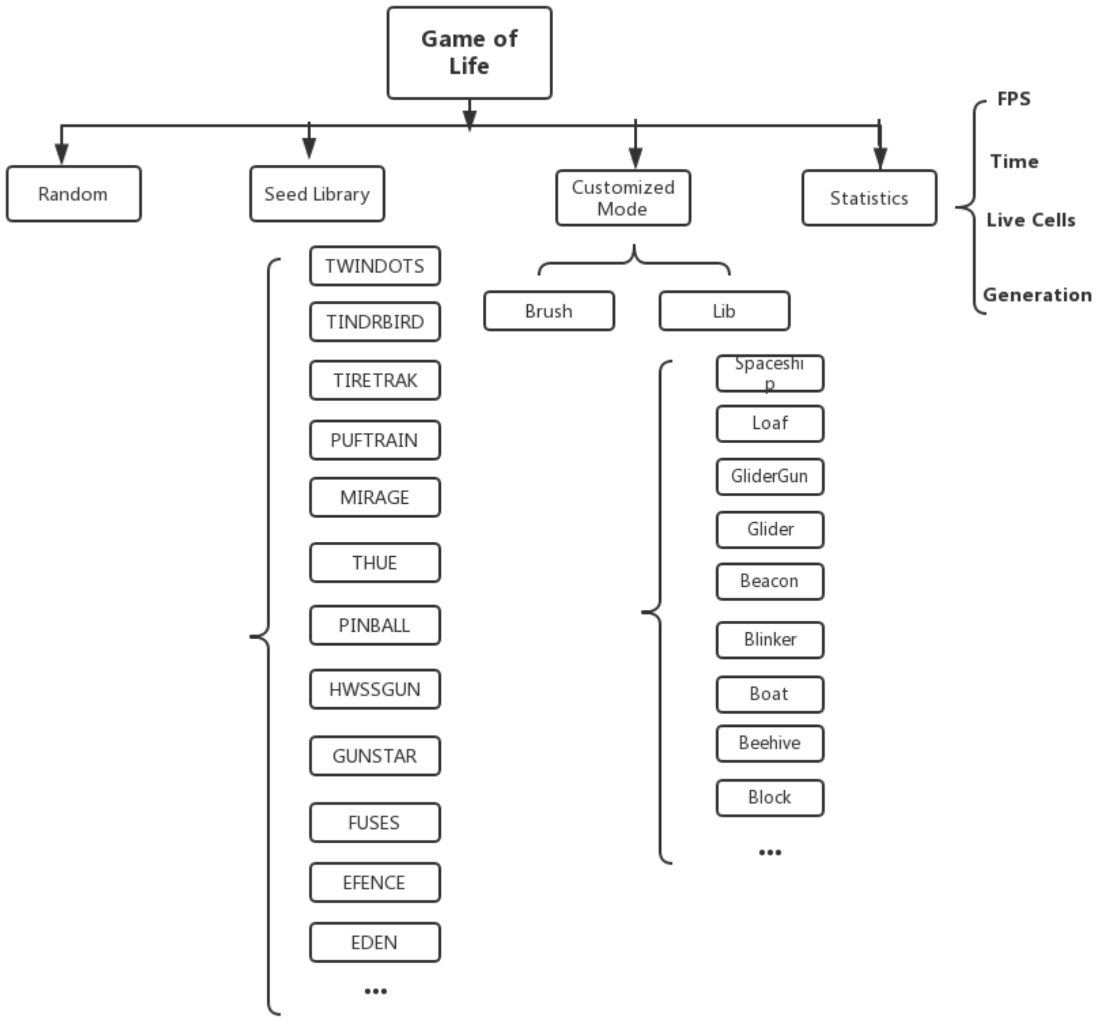


Figure 1: The interface design overview

2.2 File System

2.2.1 Loading files

We built a firm file system for our project. When new seeds are inserted into the library, one do not need to edit the codes but just inserting the seed files.
HashMap is used in the file system to do the hashing between file name and the file format, which is aimed at reading different types of file format in future.

2.2.2 Decoding files

The .LIF files if of ASCII format, which draws the pattern with "." and "*" symbols. The lines length should not exceed 80 characters. The files contains '#D some description' or '#P 8 -14'. In order to decode the files and get the patterns in the seed library, stack is used to decode the LIF files in the seed library in this project. Every time that the stack is not empty or we encounter a space or '\\', we pop the numbers and calculate the number stored in stack to get the certain mode we choose.

2.3 OpenGL

With the development of coding languages, there are plenty of implementations based on other languages like JavaScript, Python, etc. However, OpenGL is still a better choice since we can make full use of the graphic card resource to do parallel rendering.

All the versions implemented by OpenGL today are using GLUT which is a very old OpenGL library. And they do not take advantage of the graphic cards. The method we use is using GLFW library, a more extensible library, and implement it with shaders, which is much more efficient than those old versions.

OpenGL library are different under different environment. The known bug we fixed is that the window attributes(height and width) are different different in Linux and MacOS.

3 Results

The welcome page of the user interface is shown as below:

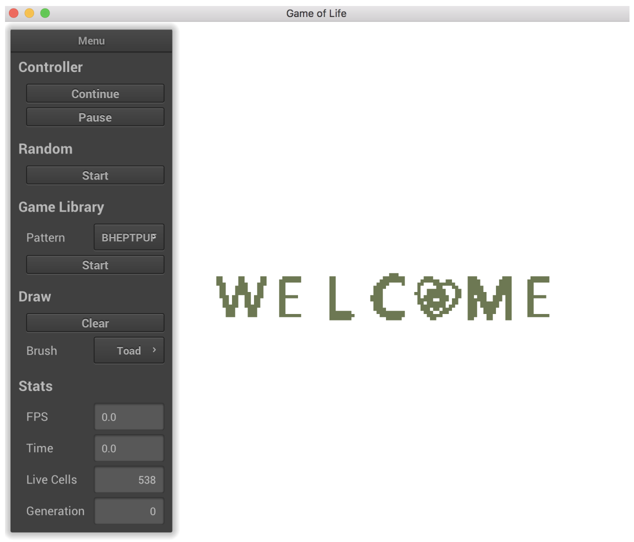
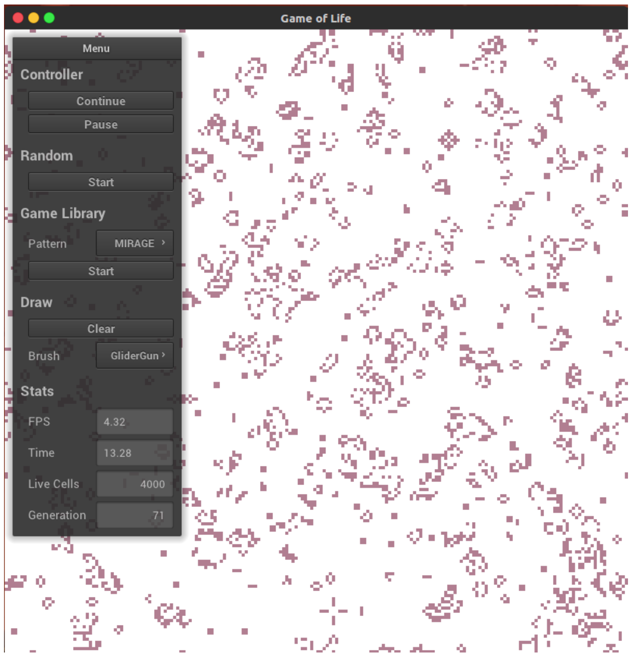


Figure 2: The welcome interface

The menu contains five main submenu

- controller is used to start or pause the current frame
  - Continue button: start the game from current frame
  - Pause button: pause the running game

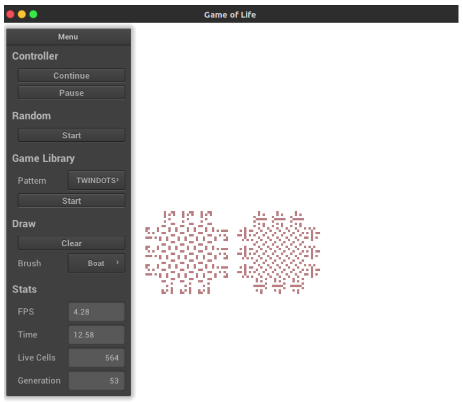
- random is used to load random pattern and start the life game by using these initial random pattern. The result is shown as below:



Pattern MIRAGE
Brush GliderGun
FPS 4.32
Time 13.28
Live Cells 4000
Generation 71

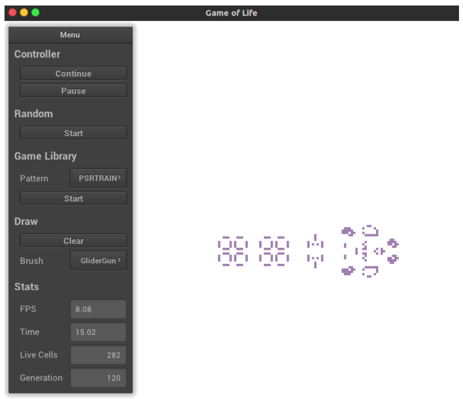
Figure 3: Screen Shot during Running Random Pattern

- game library contains various library pattern, the user could load these interesting pattern and observe the life game after click the start button. Life state is shown in Figure 4.



Pattern TWINDOTS
Brush Boat
FPS 4.28
Time 12.58
Live Cells 564
Generation 53

(a) EFENCE



Pattern PSRTRAIN
Brush GliderGun
FPS 8.08
Time 15.02
Live Cells 282
Generation 120

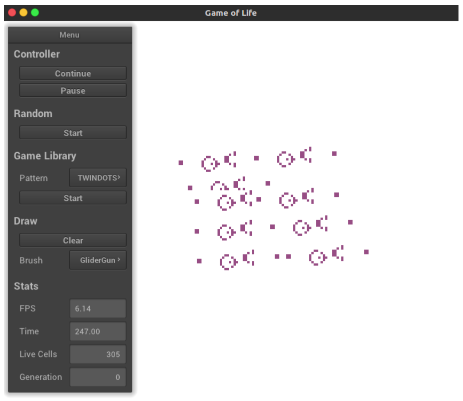
(b) PSRTRAIN

Figure 4: Game Library Examples

- Draw allows user defined their own pattern. The user can choose one or several elements to construct their own pattern and start the life game using their design pattern. Use right click to draw.
  - Clear button: clean the content in the screen

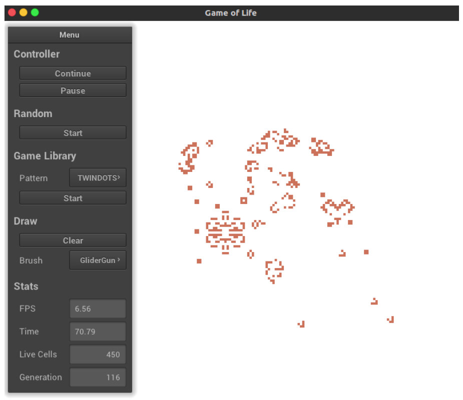
  Two designed patterns and their states during running time is shown in Figure 4.

- Stats exhibits the information of the life game.
  - FPS is the frame which runs during each second;
  - Time is the running time of the life game;
  - Live Cells is live cells in current frame;
  - Generation is nth states from the start state.



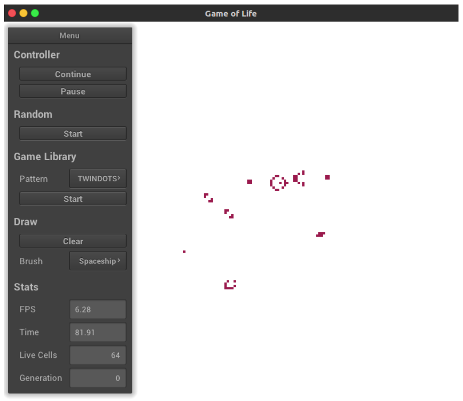
Pattern TWINDOTS
Brush GliderGun
FPS 6.14
Time 247.00
Live Cells 305
Generation 0

(a) Pattern1



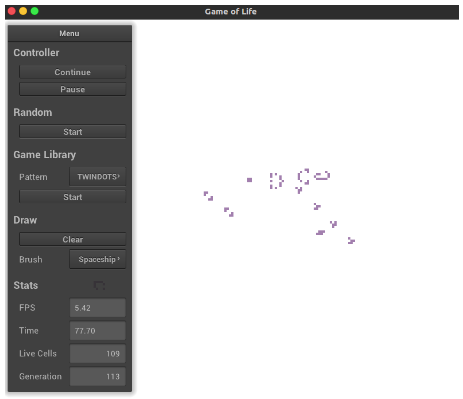
Pattern TWINDOTS
Brush GliderGun
FPS 6.56
Time 70.79
Live Cells 450
Generation 116

(b) Screen Shot during Running Pattern1



Pattern TWINDOTS
Brush Spaceship
FPS 6.28
Time 81.91
Live Cells 64
Generation 0

(c) Pattern2



Pattern TWINDOTS
Brush Spaceship
FPS 5.42
Time 77.70
Live Cells 109
Generation 113

(d) Screen Shot during Running Pattern2

Figure 5: User Design Pattern

3.1 Efficiency

To perform the experiment, we run the software on a platform with 1.6GHz X 8CPU, Graphics Intel Kabylake GT1.5, OS Type 64-bit, Disk 104.3GB with OPENGL 4.6. The operating system is Linux.

Size	100	130	160	190	220	250	280
Time	0.0416	0.047	0.071	0.104	0.141	0.168	0.214
Size	310	340	370	400	430	460	
Time	0.260	0.260	0.337	0.364	0.498	0.478	

Table 1: Elapse Time per Frame by Board Size

The game board is a square area, the size in the table means the game board is the area of size x size and the time is the elapsed time per frame.

The table shows that the time is increasing smoothly with the increase of the board size(roughly estimated as O(n)).

4 Contribution

Zeyu Chen: Builds the whole program framework, realizes the OpenGL implementation. Design .brush files and decoding. Helps with draw mode implementation. Makes demo video.

Xibei Zhang: Uses nanogui to build the user interface and constructs the whole file system. Makes majority contribution to the report.

Yijun Zhang: Focuses on .LIF(1.05) files decoding and implements draw mode. Makes contribution to the results part of the report.

References

[1] Wikipedia contributors, “Cellular automaton — Wikipedia, the free encyclopedia,” 2018, [Online; accessed 3-May-2018].

[2] Daniel Shiffman, *The nature of Code*, 2012.

[3] Vitaliy Kaurov, “Why are scientists interested in cellular automata?,” .