武汉大学计算机学院
本科生课程设计报告

元胞自动机的 C++ 代码实现与分析

专　业　名　称: 数学与应用数学

课　程　名　称: 程序设计 A

指　导　教　师: 胡文斌

学　生　学　号: 2023300002015

学　生　姓　名: 王晓智

二〇二五年六月

郑 重 声 明

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠. 尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容. 对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明. 本设计报告的知识产权归属于培养单位.

本人签名: _____ 日期: _____

摘　　要

元胞自动机（Cellular Automaton, CA）是一种离散模型，它由一个格子阵列和定义在格子上的状态组成. 元胞自动机的每一个格子可以处于有限数量的状态，格子的状态在离散的时间步骤内根据一组规则进行更新. 元胞自动机广泛应用于物理学、生物学、计算机科学等领域.

在本次实验中中, 主要完成了 1. 部分元胞自动机的控制台展现代码实现;2. 利用 sdl 库在 mac 平台上实现图形界面;3. 相关分析和介绍.

由于网络上流传的代码大都是在 windows 系统上实现，需要使用到 windows.h 库，作为在 mac 上使用 sdl 库实现图形化的一个成果，还是具有一定的跨平台优越性的.

关键词: 元胞自动机; SDL; C++

目　　录

1 引言

1.1 研究背景

元胞自动机（Cellular Automata, CA）是一种时间、空间和状态均离散的动力学模型，最早由冯·诺依曼在 20 世纪 40 年代提出，用于研究自我复制机器的可能性。其核心思想是通过简单的局部规则驱动大量元胞的同步更新，从而模拟复杂系统的时空演化行为。这种“简单规则产生复杂行为”的特性，使得元胞自动机成为研究自组织、混沌、分形等非线性现象的重要工具。

1.1.1 理论发展

奠基阶段：冯·诺依曼与乌拉姆构建了首个元胞自动机模型，探索机器自我复制的理论可能性。

经典突破：1970 年康威设计的“生命游戏”（Game of Life）通过简单的生死规则（如“3 个存活邻居则生，2 或 3 个存活邻居则保持”），展现了元胞自动机在模拟生命现象中的潜力。

分类体系：沃尔夫勒姆（Stephen Wolfram）在 20 世纪 80 年代提出四类动力学行为分类（平稳型、周期型、混沌型、复杂型），为元胞自动机的理论研究提供了系统框架。

1.1.2 应用领域

元胞自动机的应用已渗透至多学科领域：生物学：模拟肿瘤生长、HIV 感染过程及群体行为（如鱼群洄游）；

物理学：用于流体力学（格子气自动机）、枝晶形成（如雪花模拟）；

计算机科学：作为并行计算模型，应用于图像处理和模式识别；

社会科学：研究经济危机传播、城市扩展等宏观现象。

1.1.3 研究意义

当前，元胞自动机在人工生命和复杂系统研究中展现出独特优势：方法论价值：通过局部规则涌现全局复杂性，为理解智能起源（如蚁群算法）提供新视角；

技术潜力：在自适应系统设计、环境模拟（如油污扩散）等领域具有广泛应用前景。

1.2 主要成果

1.2.1 规则定义与建模

元胞自动机的行为由局部规则驱动，本实验针对二维网格模型（Moore 邻域）实现以下规则体系：

- **状态集合**：S = {0,1}（0 表示死亡/空闲，1 表示存活/占据）
- **转换函数**：

$$s_i(t+1) = \begin{cases} \text{若} \sum_{j \in N_i} s_j(t) = 3 \\ \text{若} s_i(t) = 1 \wedge \sum_{j \in N_i} s_j(t) = 2 \\ \text{其他情况} \end{cases}$$

- **边界处理**：采用周期性边界条件（环形空间）

1.2.2 可视化实现方案

为直观展示元胞演化过程，设计双重可视化方案：

- **控制台输出**：通过 ASCII 字符（如” ”表示死细胞，” ”表示活细胞）实现快速原型验证
- **SDL2 图形界面**：基于像素矩阵渲染，支持实时交互与动态调参（如图4.7所示）

1.2.3 工程化实践

通过以下方法提升代码可维护性：

- **Makefile 自动化**：定义编译目标（debug/release），集成静态代码检查（如 cppcheck）

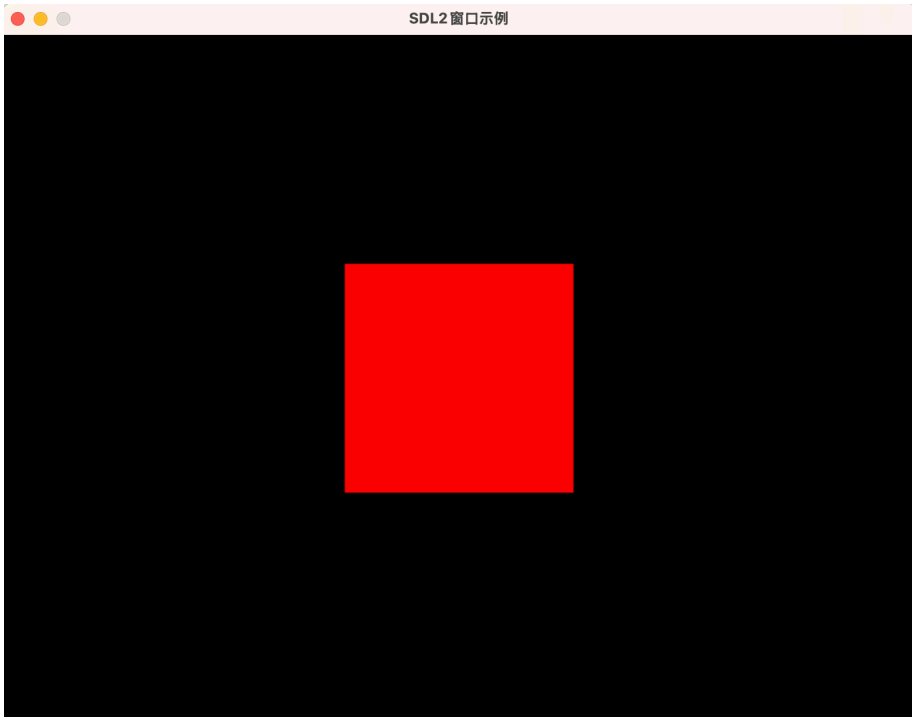

图 1.1　基于 SDL2 的可视化界面

- **模块化设计**：分离核心逻辑（CA 规则引擎）与可视化组件，遵循 SOLID 原则

2 理论基础

2.1 元胞自动机的原理

元胞自动机（Cellular Automata, CA）是一种时空离散的动力学系统模型，其核心原理可概括为以下 5 个要素:

2.1.1 基本组成

- **元胞（Cell）**：系统的基本单元，分布在规则网格（一维/二维/多维）上，每个元胞具有有限离散状态（如 $\{0, 1\}$）。
- **状态集（Set）**：可以是两种状态，用“生”、“死”, “0”、“1”, “黑”、“白”来表示；也可以是多种状态，如不同的颜色。
- **空间/元胞空间（Lattice）**：元胞在空间中分布的空间格点，可以是一维、二维或多维, 常见网格类型包括四方格（二维）、六边形格或立体网格（三维）。
- **邻居（Neighbor）**：存在于某一元胞周围，能影响该元胞在下一时刻的状态; 或者说决定元胞状态更新的局部范围，典型结构包括:
  - **Moore 邻居**（二维，8 邻域）
  - **Von Neumann 邻居**（二维，4 邻域）。
- **演化规则（Rule）**：根据元胞及其邻居元胞的状态，决定下一时刻该元胞状态的动力学函数，也可以是状态转移方程。状态转移函数 $\delta : S^k \to S$，基于当前元胞及其邻居状态计算下一时刻状态。数学表示为:

$$s_i(t+1) = \delta\left(s_{i-r}(t), \ldots, s_i(t), \ldots, s_{i+r}(t)\right)$$

  其中 r 为邻居半径。

2.1.2 演化机制

- **同步更新**：所有元胞按离散时间步并行更新状态。
- **局部性**：元胞状态仅依赖自身及邻居的当前状态，无全局信息参与。

- **边界条件**：常用周期型（环形连接）、反射型或固定值型处理有限空间的边界效应。

2.1.3 行为分类（Wolfram 分类）

1. **平稳型**：演化至均匀静止状态（不动点）。
2. **周期型**：形成稳定振荡结构（如生命游戏中的” 眨眼器”）。
3. **混沌型**：产生无序、分形等不可预测模式。
4. **复杂型**：涌现局部有序与全局混沌的混合行为（如生命游戏的” 滑翔机”）。

2.2 一维元胞自动机

一维元胞自动机（1D Cellular Automata）是最简单的元胞自动机形式.

2.2.1 基本结构

- **元胞空间**：由直线上的离散格点构成，每个格点称为一个元胞，通常用整数索引表示位置（如 $i \in \mathbb{Z}$）。
- **状态集合**：每个元胞的状态 $s_i \in S$ 为有限离散值，最常见的是二进制状态 $\{0, 1\}$。
- **邻居定义**：通常采用半径为 r 的对称邻居，例如 $r = 1$ 时，元胞 i 的邻居为 $\{i-1, i, i+1\}$。

2.2.2 演化规则

- **局部映射**：下一时刻的状态 $s_i(t+1)$ 由当前时刻自身及邻居的状态共同决定：

$$s_i(t+1) = f\big(s_{i-r}(t), \ldots, s_i(t), \ldots, s_{i+r}(t)\big)$$

  其中 f 为局部规则函数。
- **同步更新**：所有元胞的状态同时更新，时间 t 为离散步长。

2.2.3 典型示例

- **初等元胞自动机（ECA）**：状态数 $k = 2$、半径 $r = 1$ 的一维 CA，共有 256 种可能的规则（Wolfram 编号规则）。例如：
  - **Rule 30**：产生混沌模式，用于伪随机数生成。
  - **Rule 110**：具有通用计算能力，可模拟图灵机。

2.2.4 动力学行为分类（Wolfram 分类）

1. **平稳型**：全 0 或全 1。
2. **周期型**：如 0101 循环等。
3. **混沌型**：表现出无序、分形等不可预测行为。
4. **复杂型**：涌现局部有序与全局混沌的混合模式。

2.2.5 应用场景

- **物理模拟**：一维晶格中的粒子扩散、波动传播。
- **密码学**：Rule 30 用于生成加密序列。
- **理论计算**：证明计算普适性（如 Rule 110）。

2.2.6 NaSch 模型（Nagel-Schreckenberg 模型）

NaSch 模型是由德国学者 Kai Nagel 和 Michael Schreckenberg 于 1992 年提出的一维交通流元胞自动机模型，用于模拟车辆在单车道上的行驶行为。

模型将道路离散化为等距元胞，每个元胞为空或容纳一辆车，车辆速度 $v_n \in \{0, 1, \ldots, v_{\max}\}$。每个时间步按以下四步并行更新：

1. **加速:**$v_n \leftarrow \min(v_n + 1, v_{\max})$（司机倾向于加速至最大速度）
2. **减速:**$v_n \leftarrow \min(v_n, d_n)$（避免碰撞，$d_n$ 为与前车间距）
3. **随机慢化:** 以概率 p 令 $v_n \leftarrow \max(v_n - 1, 0)$（模拟驾驶员行为差异）
4. **位置更新:**$x_n \leftarrow x_n + v_n$（车辆按新速度移动）

图 2.1　NaSch 模型下的仿真

2.2.6.1　关键特性

- **自发拥堵**：随机慢化规则可引发交通波的传播，模拟真实拥堵现象.
- **相变行为**：流量-密度曲线呈现自由流、拥堵相变，临界密度约为 0.185[1]。
- **周期性边界**：常用环形道路假设以简化模拟。

2.2.6.2　应用与扩展

- **基础场景**：单车道交通流模拟，如高速公路。
- **扩展模型**：STCA（双车道换道模型）、交叉口信号控制等。
- **实际应用**：交通管理策略评估、拥堵机理分析。

一维元胞自动机虽结构简单，但通过局部规则的迭代能展现丰富的全局行为，是研究复杂系统自组织现象的基础模型。

2.3　二维元胞自动机

二维元胞自动机是一种在二维规则网格上运行的离散动力系统模型

2.3.1　基本结构

- **元胞空间**：由二维方形网格构成，每个网格点代表一个元胞，状态 $s_{i,j} \in S$（如 $\{0,1\}$ 表示死亡/存活）。
- **邻居定义**：
  - **冯・诺依曼邻居**：上下左右 4 个正交相邻元胞
  - **摩尔邻居**：包含对角相邻的 8 个元胞（生命游戏采用此结构）

[1] https://www.docin.com/p-2693167189.html

2.3.2 演化规则

- 状态更新函数：

$$s_{i,j}(t+1) = \delta\left(\{s_{k,l}(t) \mid (k,l) \in N_{i,j}\}\right)$$

  其中 $N_{i,j}$ 表示 (i,j) 的邻居集合。
- **同步更新**：所有元胞状态同时改变

2.3.3 经典模型

- **生命游戏（Game of Life）**：
  - 存活条件：2 或 3 个存活邻居
  - 死亡条件：邻居数 <2（孤独）或 >3（拥挤）
  - 繁殖条件：死亡元胞有恰好 3 个存活邻居
- **森林火灾模型**：模拟火势蔓延，状态包括「树木」「燃烧」「空地」

2.3.4 行为分类（Wolfram 分类）

1. 平稳型：演化至均匀状态（如全存活或全死亡）
2. 周期型：形成振荡结构（如生命游戏中的”滑翔机”）
3. 混沌型：产生无序分形模式
4. 复杂型：涌现稳定结构与混沌的混合

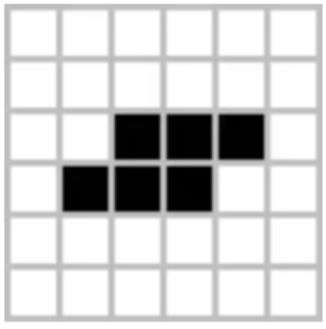

图 2.2　二维元胞自动机中的一类振荡器

2.3.5 应用领域

- 物理模拟：流体动力学、晶体生长
- 生物建模：细胞分化、流行病传播
- 社会科学：交通流、城市规划
- 图像处理：模式识别、纹理生成

二维元胞自动机通过局部规则驱动全局复杂行为，是研究涌现现象的重要工具。其 MATLAB/Python 实现通常涉及邻域卷积和状态矩阵迭代。

3　实验环境介绍

3.1　主机环境

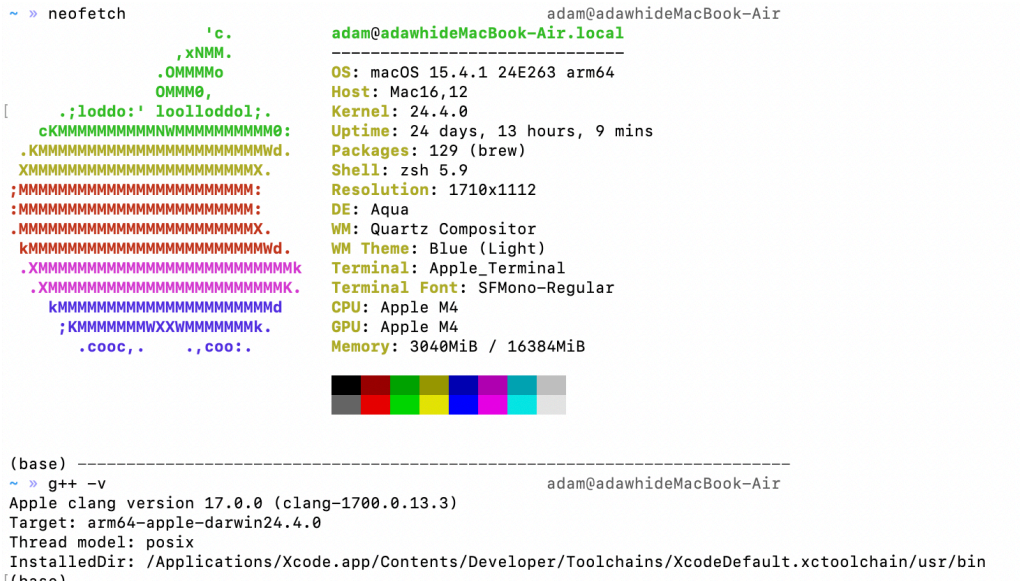

图 3.1　实验环境

3.2　软件环境:SDL 库

SDL（Simple DirectMedia Layer）是一套开源的跨平台多媒体开发库，由 Sam Lantinga 于 1998 年开发，使用 C 语言编写。它主要用于游戏、模拟器和媒体播放器等多媒体应用的开发，支持 Windows、Linux、Mac OS X 等多种操作系统

4　设计与实现

4.1　结构设计

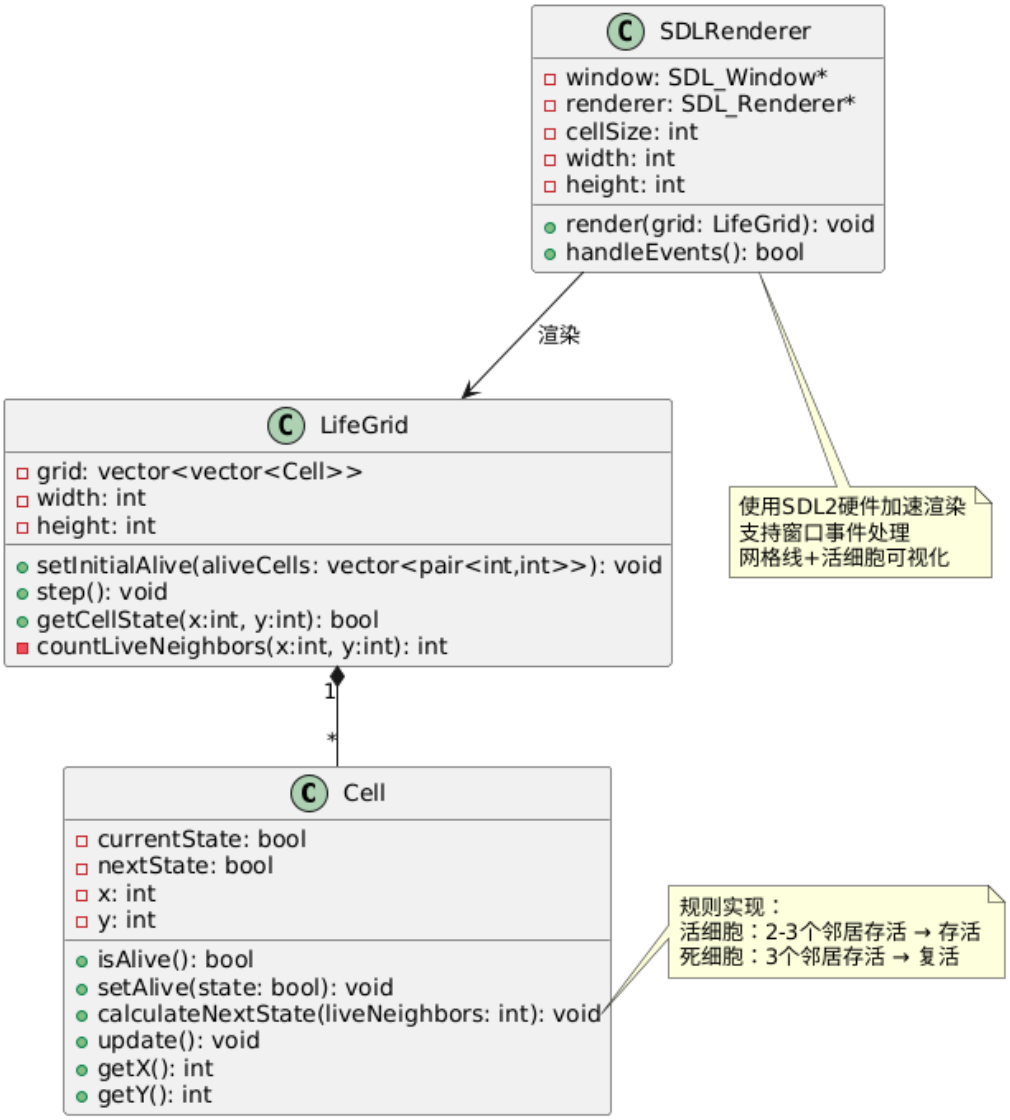

图 4.1　类的构造

4.2 类的设计与实现

Cell 类：

封装细胞状态转换逻辑（规则 1-4）

使用双缓冲模式（current/nextState）避免迭代冲突

坐标属性支持环形边界检测

LifeGrid 类：

使用 STL vector 实现动态二维网格

countLiveNeighbors 实现环形边界处理（模运算）

分离状态计算 (step) 与数据获取 (getCellState)

SDLRenderer 类：

封装 SDL2 初始化/销毁逻辑（RAII 模式）

事件处理与渲染解耦

细胞渲染使用 `SDL_Rect` 填充优化性能

设计模式应用

观察者模式：SDLRenderer 观察 LifeGrid 状态变化

策略模式：细胞规则可通过继承 Cell 类扩展

外观模式：SDLRenderer 封装 SDL2 复杂 API

4.3 实际界面

以下是部分代码运行过程中的实际演示

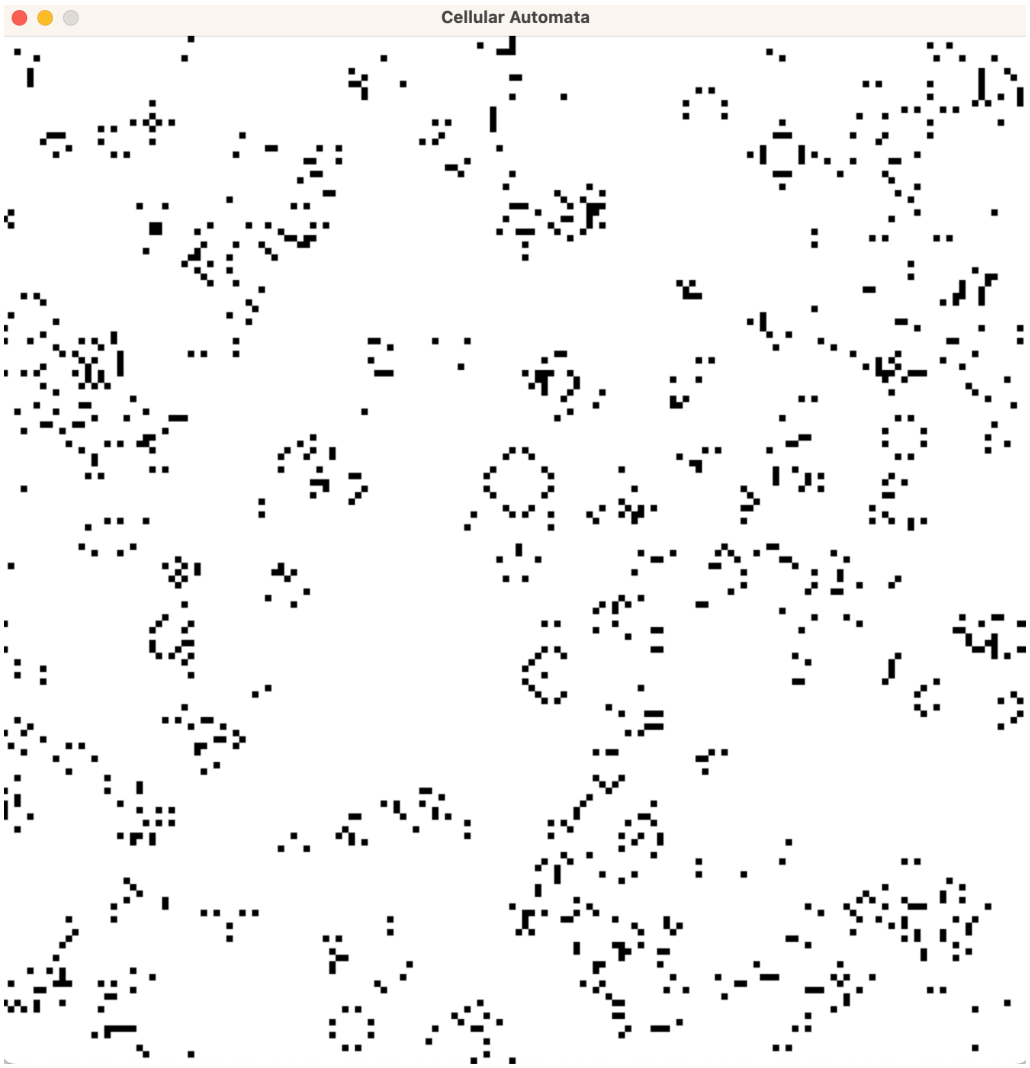

图 4.2　box

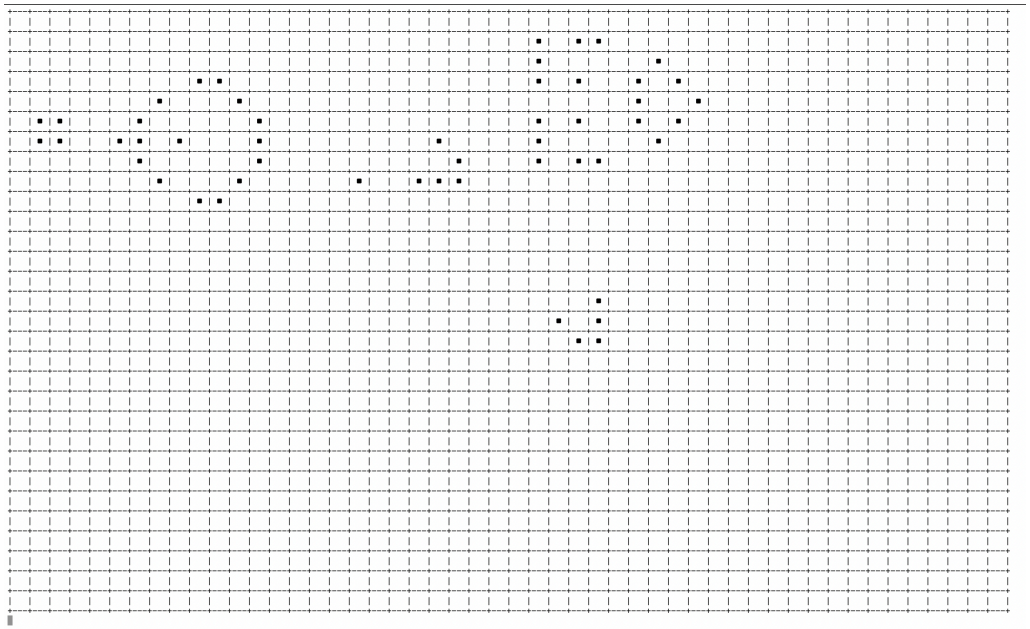

图 4.3　gameoflife

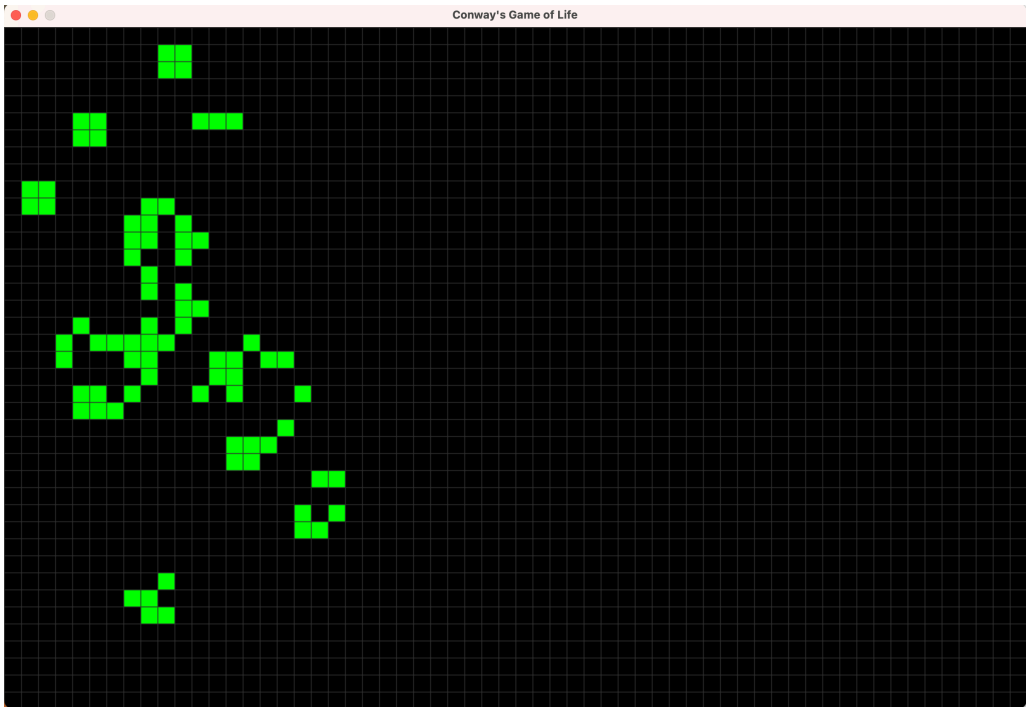

图 4.4　gameoflifesdl2

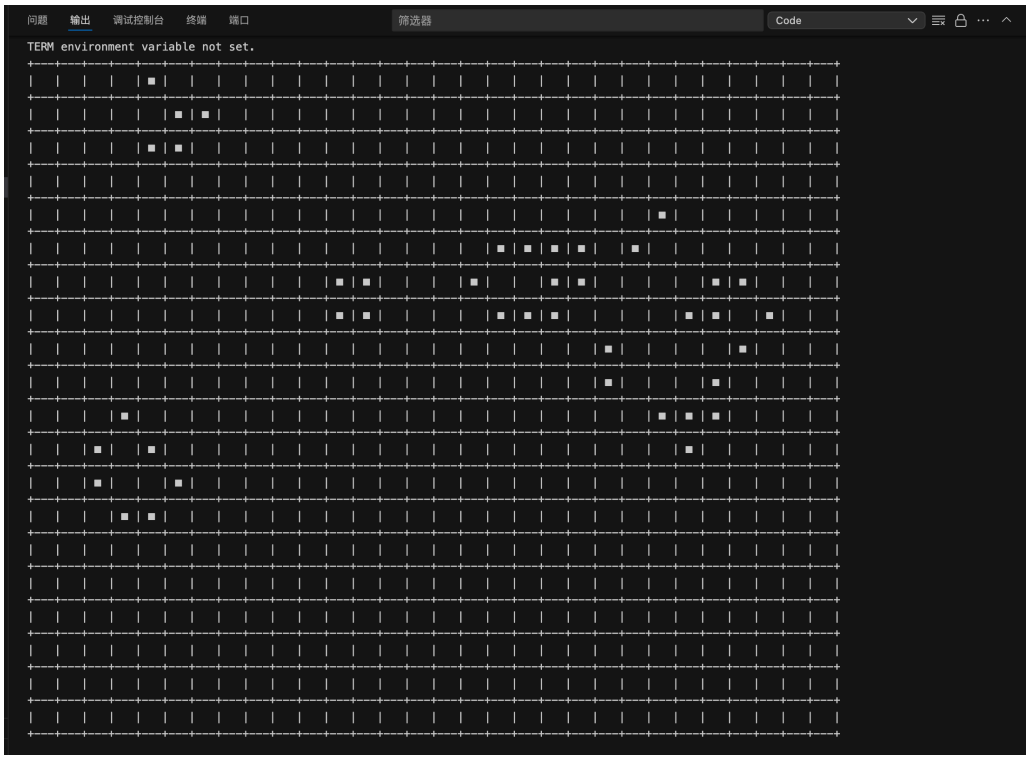

图 4.5　lifegame

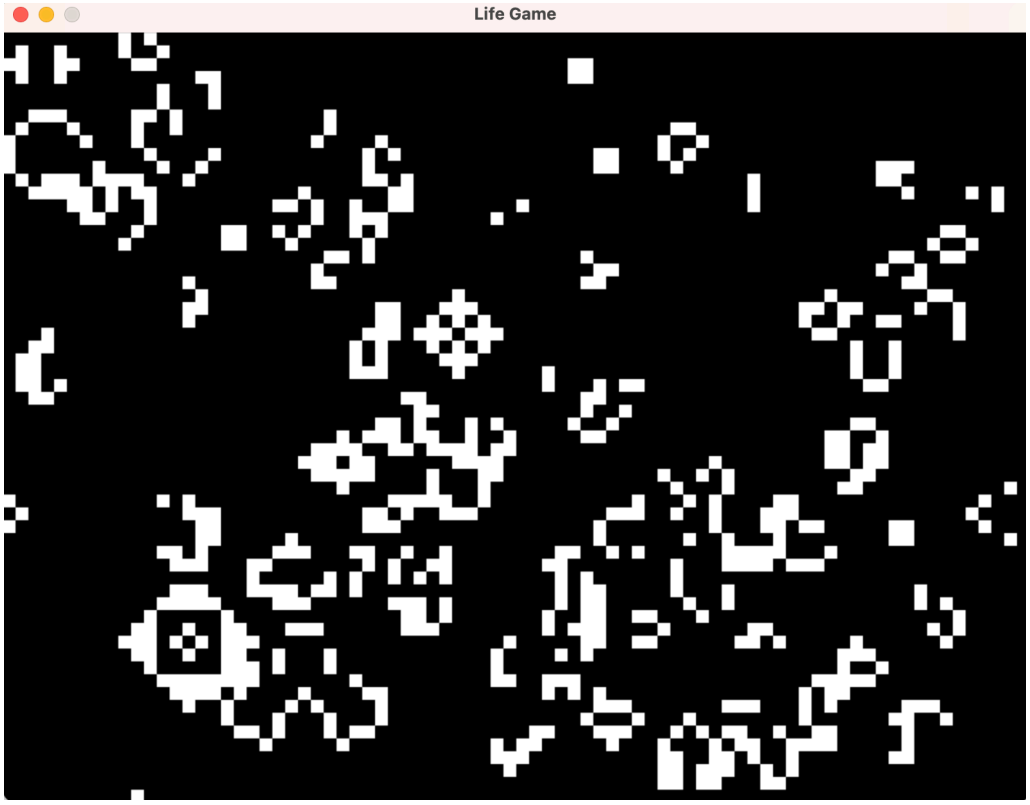

图 4.6　lifegame2

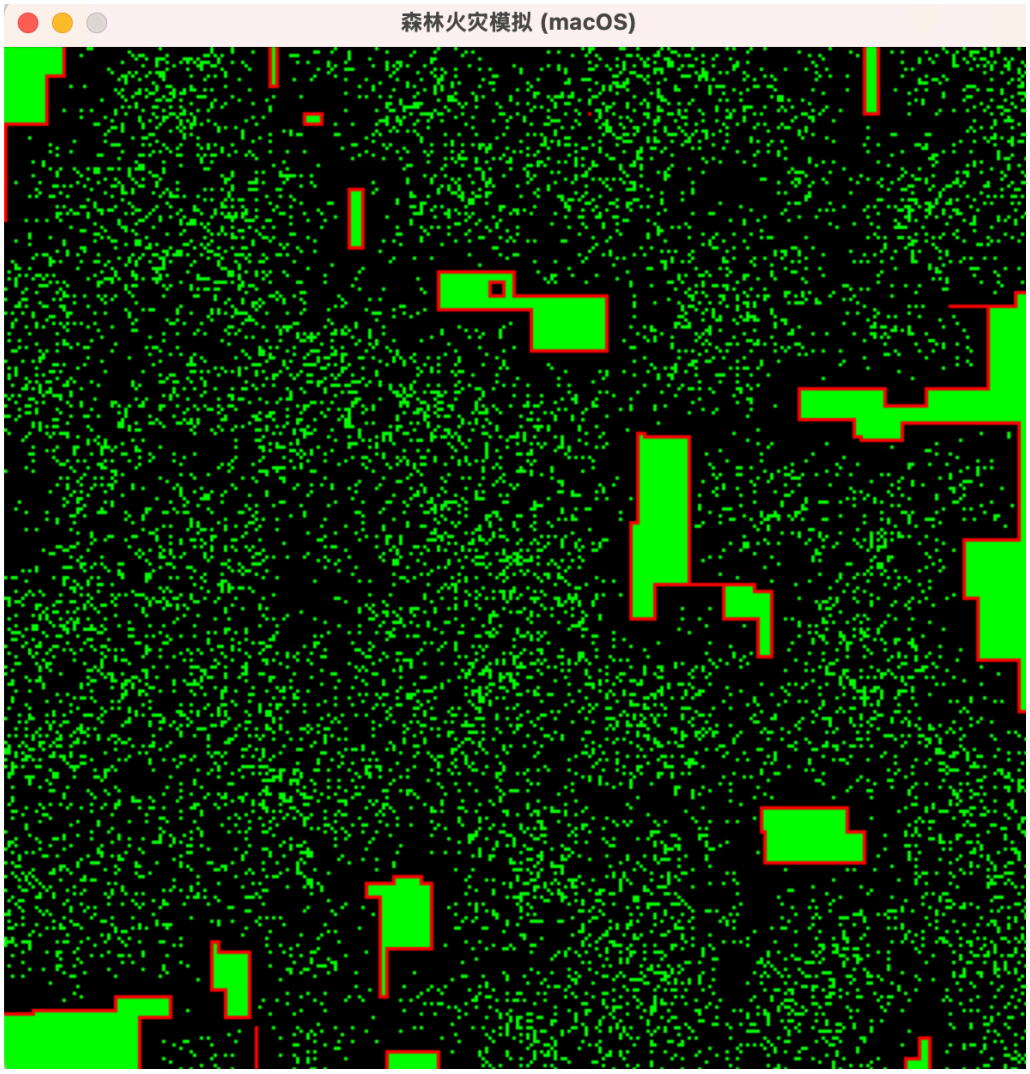

图 4.7　模拟火烧

5 总结与展望

此程序在类上和结构的实现思路并不复杂„ 严格来讲只有三个类, 可以认为只要花时间都可以轻松写出来, 在此程序的实践过程中, 主要的困难在于 mac 系统上导入 SDL 库的极度困难性, 因此我在实现的过程中特意写出每一个实现方法的编译代码 or makefile 文件, 期望以此实现泛用性, 不过不得不提,mac 上 cpp 的图形界面库软件实在太少 (? 说的可能不太严谨) 了, 这一点远远不及 windows 上的方便性.

特别的,mac 上 cpp 编译带库非常困难, 特别是使用 homebrew 安装的 sdl 库, 在位置不明晰的确定下, 编译时往往会出现找不到 #include <SDL2/SDL.h> 的情况, 非常的痛苦所以我的代码中包含了一个 `the_way_to_use_sdl_undermac`，以提供一个在 mac 上使用 SDL 库的成功案例作为参考程序，也是编写 Makefile 的重要参考。

本程序的主要作用在于实现个人严格控制下的元胞自动机的初步仿真和初步界面的实现, 预计在后面需要进一步完善, 主要目标有添加键盘控制功能, 考虑添加机器学习要素研究其规律的想法, 不过感觉这要用 python 方便多了 (各方面 ww)

参考文献

[1] 知乎文章. https://zhuanlan.zhihu.com/p/113204715.

[2] 相关介绍 https://xinshoujiajia.github.io/chANKOS/chp5/content2.html.

[3] GitHub 项目. https://github.com/Windxy/CA_SEIR.

[4] GitHub 项目. https://github.com/DevilHamster/Cellular-Automata?tab=readme-ov-file.

[5] GitHub 项目. https://github.com/zhangerfa/Trands_cellular_automata.

[6] GitHub 项目. https://github.com/zhan3333/python-Cellular-automata/blob/master/cellular_automata.py.

[7] GitHub 项目. https://github.com/amao-9/celluar_aotumation/blob/main/box.py.

[8] 知乎问答. https://www.zhihu.com/question/20711603.

[9] 知乎文章. https://zhuanlan.zhihu.com/p/93738666.

[10] 知乎文章. https://zhuanlan.zhihu.com/p/621070746.

[11] 知乎文章. https://zhuanlan.zhihu.com/p/45026142.

教师评语评分

评语:

评分:

评阅人:

年　　月　　日

(备注:对该实验报告给予优点和不足的评价,并给出百分制评分。)