南開大學
Nankai University

高级语言程序设计 2-2

大作业项目报告

梅骏逸

学号 ：2111876

班级 ：0972

2022 年 5 月 3 日

目录

一、 作业题目

(一) 概述

本项目实现了一个对一维、二维、三维元胞自动机（Cellular Automata）的模拟器——Casim（**C**ellular **A**utomata **SIM**ulator）。可以通过 lua 语言脚本对模拟器中元胞自动机的规则进行自定义，从而兼顾模拟器的灵活性和规则的可读性。同时可在程序中对观察视角、元胞自动机显示模式、元胞自动机区域大小进行调整。

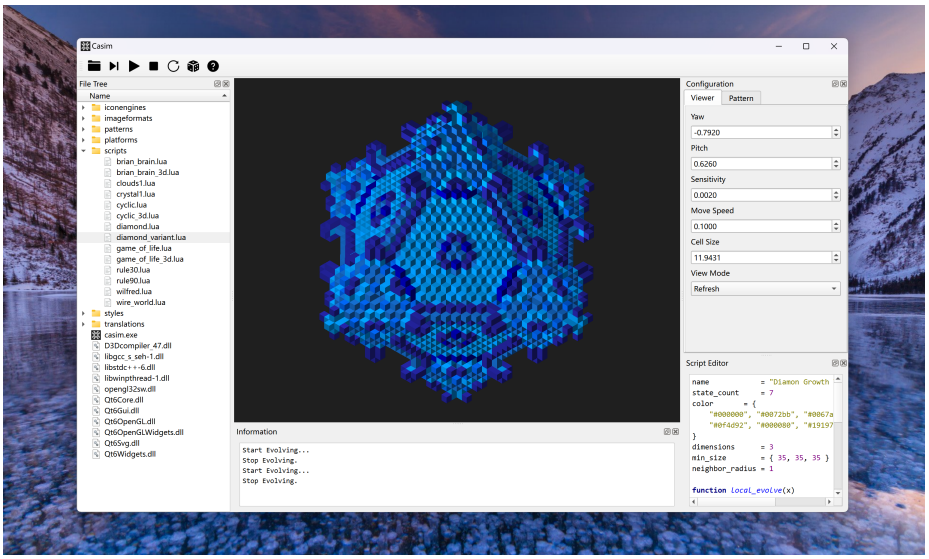

图 1: Casim 主界面运行时截图

(二) 关于元胞自动机

一般地，元胞自动机可以由元胞、元胞状态、邻域和状态更新规则构成，用公式可以表示为：

$$A = (L, d, S, N, f)$$

其中 L 为元胞空间（即所有元胞的集合），d 代表元胞自动机的维数，S 是元胞自动机离散且有限的状态集合，N 为某个邻域内元胞状态的集合，f 为元胞自动机局部状态的更新规则。

简单的元胞自动机规则在特定的初始模式下可以表现出非常复杂的行为，因此，元胞自动机在各种领域的建模都起到了很大的作用，对元胞自动机的运行过程进行可视化能够使得其运行过程更加直观，这就是本项目希望能够达到的效果。

二、 开发环境

操作系统：Windows 11 22H2
编程语言：C++, Lua, GLSL
使用的库：xtensor, xtl, opengl, freeglut, qt, lua
编译器：MinGW g++

三、 主要实现方式

(一) 模拟器核心逻辑

模拟器所支持的 Lua 代码是以 Lua 5.4.4 版本的标准定义的，并且通过 lua 库对脚本进行解释运行。在脚本中必须说明元胞自动机的名字、维数、状态数、状态对应的显示颜色，最小要求的区域大小以及邻域半径。

一个可以在模拟器中运行的脚本至少应当具有如下的几个部分（以 Rule 30 为例）

Rule 30 规则描述脚本

```lua
name            = "Elementary Rule 30"
color           = { "#000000", "#ffffff" }
state_count     = 2
dimensions      = 1
min_size        = { 100 }
neighbor_radius = 1

function local_evolve(x)
    return (x[1] ~ (x[2] | x[3])) -- p xor (q or r)
end
```

每次更新时将局部状态（根据 Neighbor Radius 即邻域半径获取）以一维数组的方式通过压栈传入 local_evolve 函数从而获取下一代当前位置的元胞的状态。

由于使用脚本定义元胞自动机的规则，所以模拟器支持元胞自动机的模式就能够更加灵活，用户只需要通过撰写简单的 lua 脚本就能够定义用于可视化的元胞自动机。

(二) 模式的载入与保存

对于每一个规则都会有特定的模式（Pattern），同一个规则下的不同模式可能表现出在时间上的不同特性，所以模拟器实现了对模式的保存与读取功能。

每一种模式都使用多维数组的方式进行存储。通过 xtensor 库以 npy[1]文件的方式进行读写。

(三) 元胞自动机的可视化

在模拟器中，所有维数的元胞自动机中的元胞都以空间中的方块表示，所以模拟器中的显示器（Viewer）一直都是以 3D 的形式运行。

(四) 3D 显示模式的操作及实现

3D 显示模式主要是通过 opengl 以及 qt 中的 QOpenGLWidget 来实现的，同时可以通过鼠标对观察视角进行拖动，也可以通过键盘进行移动。鼠标滚轮可以对团进行放大和缩小.

由于要兼容 2D 和 1D 的元胞自动机，故 3D 显示时的投影采用了正射投影法（Orthoraphic Projection）, 因此对视角远近的改变（即鼠标滚轮缩放）被看作为元胞本身大小的变化，每次改变大小时改变投影矩阵并刷新显示区域达到显示效果。

[1]npy 是 Numpy 存储数组的一种文件格式,此处以 npy 文件形式存储元胞状态集合,所以元胞模式也可以通过 Python 和 Numpy 库（或兼容的其它库）进行读写

同时鼠标滚轮的缩放会以鼠标位置为依据，将鼠标所指向的位置相对固定地进行缩放，其计算方法是根据屏幕大小以及鼠标在显示区域中的位置计算出 3D 观察视角的位置并进行更新从而达到效果。

1. 元胞自动机的显示模式

对于一维与二维的元胞自动机，模拟器提供了两种显示模式：刷新（Refresh）和累积（Accumulate）。仍然以 Rule 30 为例，Rule 30 属于基本（Elementary）元胞自动机，这一类元胞自动机在刷新模式下只能在一条线上表现出不同模式，但是在累积模式下，就会出现复杂的形态。

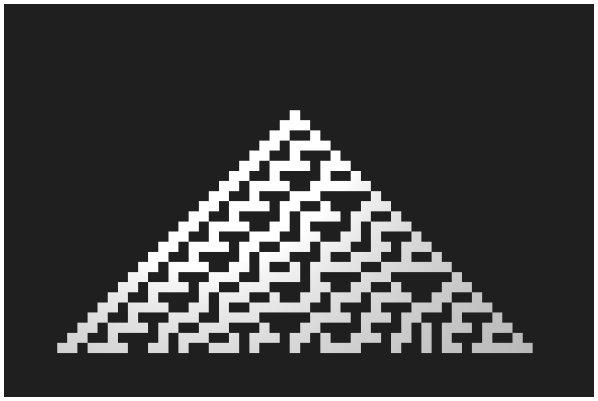

图 2: 累积模式下 Rule 30 展现出的复杂模式

而对于二维元胞自动机来说，累积模式也可以呈现出惊艳的效果（以 Brian's Brain）为例

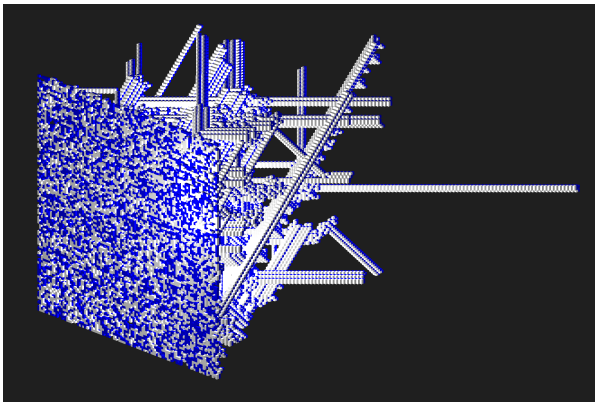

图 3: 累积模式下 Brian's Brain 的效果

2. 元胞自动机的显示区域

理想中的元胞自动机的区域应当是无限大的，也就是可以容纳无限个元胞，但是在这个模拟器中采用的是固定大小的元胞空间，在区域外的自动视为默认状态传入局部状态表。

四、 测试

项目测试表如下[2]

[2]其中核心逻辑测试的所有规则脚本源码均位于项目文件夹中的 scripts 文件夹中

模块名称	测试内容	结果	备注
3D 显示模块	生命游戏（Pattern）	PASS	测试 Pattern 显示效果
	Cyclic（Pattern）		
	布莱恩大脑（Pattern）		
	第三十号规则（Pattern）		
3D 操作模块	鼠标拖动	PASS	测试交互操作
	鼠标滚轮缩放		
	键盘移动		
	键盘缩放		
元胞自动机脚本解释及核心逻辑	生命游戏	PASS	测试核心逻辑
	生命游戏		
	布莱恩大脑		
	布莱恩大脑（3D）		
	Clouds1		
	Crystal1		
	Cyclic		
	Cyclic (3D)		
	Diamond		
	Diamond Variant		
	第三十号规则		
	第九十号规则		
	Wilfred		
	Wireworld		

五、 总结

本项目结合使用 qt 等库实现了一个元胞自动机模拟器并且可以从脚本语言的代码（Lua）中动态解释运行规则，同时支持多维元胞自动机的可视化[3]。

[3]截至本报告完成时，项目已有 26 次提交（commit）符合大作业要求。Github 地址, Gitee 地址, Bilibili 讲解地址