生命游戏技术文档

软件 41 楼昀恺 2014013407

一、前言

1.1 程序名称

生命游戏。

1.2 程序内容简介

本程序是一个用 JavaScript 及 HTML 开发的小游戏，需要打开 lifegame.html 文件在浏览器下运行。

游戏的主要规则如下：游戏有一个包含一定数量方格的棋盘，每个方格内可以包含一个活细胞或一个死细胞，包含活细胞则对应方格为白色，否则为黑色区域。游戏开始时将在棋盘内随机产生一定数量的细胞，随后每个细胞根据其自身及其周围细胞的状态进行变化，若周围有 3 个活细胞时，若自身为死细胞，则细胞复活，否则自身状态不变；若细胞周围有 2 个活细胞则该细胞状态不变；其余情况下该细胞死亡。

用户只需打开 lifegame.html 文件，并设定好棋盘的尺寸及初始化时活细胞数量并按下“确定”按钮，上述演化过程即会自动进行。演化过程中，可通过按下“暂停”按钮暂时中止细胞演化，可通过按下“继续”按钮继续进行细胞演化。

二、程序模块介绍

2.1 pos 类

模块功能简介：

该类用于存放记录细胞状态的棋盘数组中各元素的位置信息，例如：棋盘数组 map 中元素 map[0][0]对应的 pos 类对象的 row 和 col 值分别为 0 和 0。

模块包含的方法及功能介绍：

get_row() : 返回本 pos 对象的 row 值。

get_col(): 返回本 pos 对象的 col 值。

set_row(value): 设置本 pos 对象的 row 值为 value。

set_col(value): 设置本 pos 对象的 col 值为 value。

2.2 life_map 类

模块功能简介：

该类对应游戏中的棋盘，包含游戏初始化，游戏进程控制、管理各细胞信息的功能。

模块包含的属性及功能介绍：

map： 二维数组，对应游戏中的棋盘，二维数组的每个元素是一个细胞类的实例对象。

mapsize: 设置棋盘的在横向及纵向最多能容纳的细胞数，及二位数组 map 在各维上的长度。

poslist: 一维数组，每个元素对应一个 pos 类的实例对象，初始化时即在此列表中随机访问第 i 个元素并设定其细胞为活，将其从 poslist 中去除后继续随机找细胞。

ui_map_list: 二维数组，用于界面，每个元素对应一个 div 标签，而此处 div 标签在界面上代表细胞。

record_map: 二维数组，与 map 类似，在每次更新 map 时存储所有细胞更新前的信息并据此改变各细胞状态，在更新过程中，record_map 内容不发生变化。

per_border_width: 根据屏幕高度及每列细胞数量得到的每个方格的边长。

interval: ui 界面中黑色圆形淡入淡出所花的时间。

to_change_list: 一维数组，记录在本轮更新中改变了状态的细胞用于更新 ui 界面。

ispause: bool 变量，用于判断当前游戏是否被暂停。

模块包含的方法及功能介绍：

create_list(): 创建 poslist 列表，使列表包含 map 中所有元素对应的 pos 实例对象。

set_interval(value): 设定属性 interval 为输入值 value。

match_ui_array(): 将 map 中更新的结果同步到数组 ui_map_list 中，即更新 ui 界面中黑点的显隐。

ui_map_create(mapsize): 填充 ui_map_list 数组，每个元素是一个 div，同时绘制棋盘及方格。

init_map(live_num): 随机找到一定数量的 map 中的元素，将其中细胞的状态设置为“生”，

同时调用调用 ui_map_create 函数初始化了 ui_map_list 数组，初始化了 record_map 数组。

change_ceil_status(row, col, oldmap): oldmap 对应属性中的 record_map，函数根据 record_map 中的信息更新 map[row][col]位置的细胞的状态。

get_around_status(row, col, oldmap): oldmap 对应属性中的 record_map，函数根据 record_map 中的信息返回 map[row][col]对应细胞周围 8 个细胞中活细胞的数量。

update_map(): 更新 map 中每个元素（即细胞）的状态，更新完毕后用新的 map 中各元素的状态设置 record_map 中对应位置细胞的状态。

print_self(): 输出 map 中各细胞的状态到控制台，用于调试。

2.3 ceil 类

模块功能简介：

代表细胞，保存了细胞的状态及在 map 中的位置，实现了一些获取及修改细胞信息的方法。

模块包含的属性及功能介绍：

status: 表示此细胞的状态，0 代表死，1 代表生。

row: 表示此细胞在 map 数组中的行号。

col: 表示此细胞在 map 数组中的列号。

模块包含的方法及功能介绍：

set_status(value): 设置细胞状态为 value。

get_status(): 返回细胞状态。

get_row(): 返回细胞行号。

get_col(): 返回细胞列号。

2.4 game_timer 对象

模块功能简介：

控制细胞演化，包含一个计时器，每隔一定时间控制所有细胞进行一次演化。

模块包含的属性及功能介绍：

update_interval: 相邻两次演化间的时间间隔。

mapsize: 设定的 map 的各维度的长度。

live_num: 初始化时要产生的活细胞的数量。

timer: 控制演化的计时器。

gamemap: 棋盘对象。

istest: bool 类型对象，为了在测试时避免出现图像而引入的参数。

模块包含的方法及功能介绍：

init(): 初始化游戏，包括设定演化时间间隔、棋盘尺寸、初始化活细胞数量、创建棋盘对象、初始化棋盘、设定棋盘的 interval 值。

start_timer(): 初始化并设定计时器。

call_timer_function(): 调用细胞演化的函数。

game_pause(): 暂停细胞演化。

game_continue(): 继续细胞演化。

三、主要功能的实现方法介绍

3.1 初始化时随机活细胞的生成

先对二维数组的每个元素根据其横纵坐标生成一个对应的 pos 类对象，例如 new pos(0, 0), new pos(0, 1)等，然后将这些对象全部压入一个数组 poslist 中。当要随机确定几个位置生成活细胞时，取一个 0 到 poslist 长度 - 1 的随机数，将其作为下标在 poslist 中对应的 pos 对象取出，取出的方法为：将其余 poslist 最后一个元素交换位置，然后弹出 poslist 的最后一个元素。将这个对象对应的细胞的状态设置为生，然后继续对 poslist 执行相同操作，直到设定为生的细胞适量达到要求为止。

3.2 细胞状态的更新

程序中使用 map 数组存储所有细胞的实例对象，使用 record_map 数组存储所有细胞本次更新前的实例对象状态，每次更新时，遍历所有细胞，使每个细胞根据 record_map 中的信息根据规则更新自身状态，并将更新结果存储到 map 中对应位置。当所有细胞状态更新完后，将 map 中信息拷贝至 record_map 中，用于下次更新。

3.3 ui 界面更新

ui 界面中用黑色背景的 div 代表活细胞，白色背景的 div 代表死细胞，所有表示细胞的 div 存在 ui_map_list 数组中，并以 map 一一对应。每次细胞更新后，将状态发生改变的细胞的 pos 对象存到名为 to_change_list 数组中，所有细胞完成一次更新后，将 to_change_list 数组中每个元素对应的细胞进行状态更新以实现棋盘的更新，以此办法来减少每次更新 ui 界面花费的时间。

四、单元测试方案

4.1 单元测试环境

本程序使用 JavaScript 单元测试框架 mocha 进行单元测试。

4.2 测试用例设计

对于所有设计的函数的测试，首先都包含一个判断它是不是函数的测试，其次，若它有至少一个参数，则测试其参数数量是否与设定相同。

对于类属性存取的函数如 set_row()，set_col()，get_row()，get_col()等函数，我采用的用例是 row 等于 0，col 等于 1 的对象，以避免将 row 和 col 弄混，例如 row=col=0 的测试用例就难以判断 get_row()输出的是 row 还是 col。

对于其他实现具体逻辑功能函数，包含对其输出结果正确性的测试。对于 life_map 类中功能函数的测试，需要先创建一个 life_map 对象，再用其调用函数进行测试。此处，当创建的对象的 mapsize 小于 3 时，由于每个细胞周围的细胞数量不到 8 个，虽然能测试，但测试效果不好。而当 mapsize 大于等于 3 时，随着 mapsize 的增加，程序逻辑的正确性不会随 mapsize 变化有很大变化。因此，测试中创建的 life_map 的 mapsize 均为 3 或 4 这样的边界值。

4.3 测试结果

测试结果如下图所示：

共进行了 54 个测试，成功了 54 个。

4.4 运行测试的方法

编写 lifegame_test.js 和 index.html 文件进行测试，测试时，在浏览器中打开 index.html 查看测试结果，根据结果对 lifegame.js 对应的函数进行相应的修改或不修改。每次只在 lifegame_test.js 中新增一条测试，然后写对应代码，调试至测试通过后再新增下一条测试，直至测试包含当前的所有需求且程序通过了所有测试，即完成测试和代码书写。

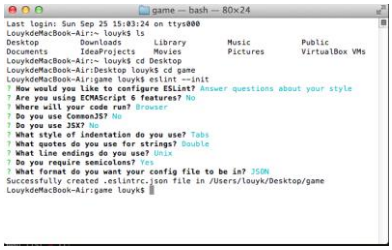
五、代码风格一致性测试

5.1 测试工具

eslint。

5.2 测试方法及结果

测试时采用的标准如下图所示：



经过测试及修改后实现了一致性较好的代码风格。