

# 生命游戏程序 技术文档

软件 42 蒋梦青 2014013443

## 一、概述

本程序主要使用 HTML 和 JavaScript 在 Web 前端搭建了可视化的生命游戏程序，并通过 mocha 测试工具对代码进行重构，即运行测试驱动的开发，提高代码质量。

程序界面如下：



点击 **start** 开始游戏，开始时按照输入框中的数字进行随机初始化活细胞，点击 **pause** 暂停，点击 **reset** 则再次根据输入框中的数字进行初始化。

按钮下方显示两个信息，**generation** 表示迭代的次数，**population** 表示现存的活细胞数量。

最下方的方块阵为可视化的生命游戏界面，白色表示活细胞，黑色表示死细胞，每 200 毫秒进行一次迭代。

## 二、技术实现

### a) 绘制地图

通过 JavaScript 向“draw-animation”容器中 **appendChild**，添加 60\*60 的小方格容器（即<div>标签结点），并修改每个方格的尺寸、位置、背景颜色，用 **map** 数组进行记录，同时使用 **mask** 矩阵来记录地图中细胞的死活情况，便于计算。

本程序原本使用 **webGL** 来绘制矩形，但经过比较之后，控制 **dom** 结点的方法效率更高，运行起来流畅不卡顿。

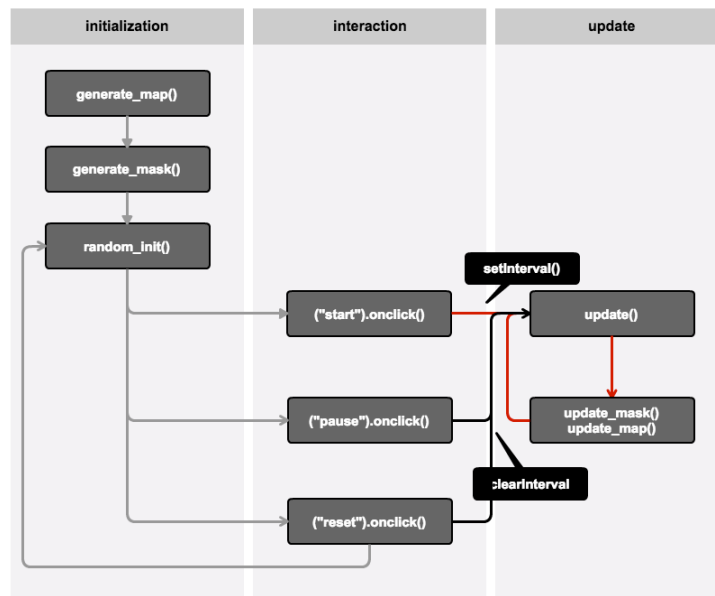
### b) 交互与动画实现

设置 **start**, **pause**, **reset** 三个按钮的 **onclick** 函数，用两个全局 **flag** 变量 **running** 和 **init**（分别表示是否运行和是否初始化）来游戏的状态，进

行 init 与 update。

其中 start 按钮的 onclick 函数中包含 setInterval 函数，每 200 毫秒执行一次 update，对细胞状态和地图颜色进行更新。

c) 程序架构



### 三、单元测试方案

a) 单元测试环境

基于 node.js 的单元测试，使用 mocha 和 chai 工具，通过浏览器浏览测试结果。

b) 测试用例设计

测试单元：initialization 单元（包含 `generate_map`, `generate_mask` 和 `random_init`），interaction 单元（模拟按钮点击和输入值改变），update 单元（主要测试 `update` 的逻辑）。

下面分单元进行叙述：

i. initialization

测试用例主要是检测：1. 输出矩阵的维度；2. 输出矩阵的初始化值；3. Dom 结点添加是否正常；3. 随机初始化活细胞是否满足数量要求。

ii. interaction






分别调用

```
document.getElementById("start").onclick();
document.getElementById("pause").onclick();
document.getElementById("reset").onclick();
document.getElementById("input").value=num.toString();
```

通过检查返回值和观察地图变化来进行测试

iii. update

由于生命游戏的迭代规则，存在一些比较特殊的生命细胞集团，这些特殊的生命细胞集团分为 3 类：

- 静止细胞，例如： (Block),  (Loaf)
- 周期细胞，例如：  (Blinker, 周期变化)
- 飞船细胞，例如： (Glider, 不断平移)

用以上四个用例作为输入来检测 `update` 单元的运行逻辑。

c) 测试结果

用上述用例进行测试，测试结果均为通过。具体可查看 `test.html`。

d) 运行测试的方法

根据 `mocha` 工具的官方文档，本程序搭建了运行测试和查看结果的 `HTML` 网页，用浏览器打开 `test.html` 即可。

## 四、部署

本程序已发布到网页 [jmq14.github.io/GameOfLife/](http://jmq14.github.io/GameOfLife/) 上，可点击该链接直接查看。