Game of Life
项目文档

作者

邹豪风
张凯

项目概述

本项目以结对编程的开发模式，实现了一个网页游戏 Game of Life，并对游戏进行了系统的模块测试。

项目实现

- 本项目通过 Javascript 实现了一个网页游戏 Game of Life；
- 项目使用了 Bootstrap 提供的 CSS 样式及组件；
- 项目使用了 HTML5 画布进行游戏主界面的绘制；
- 游戏主界面包含一系列方格，每个方格表示一个细胞生存单元；其中黑色方格表示细胞死亡，白色方格表示细胞存活；在游戏未开始或游戏暂停时，可以通过"每边细胞个数"滑动条按钮设置每一条边可容纳的细胞个数，从而对界面的细胞总数进行控制；
- 游戏同时支持随机放置细胞及手动放置细胞；在游戏未开始或游戏暂停时，可以通过点击"随机"按钮随机放置细胞，生成的细胞密度可以通过"细胞密度"滑动条按钮进行控制，也可以通过手动点击游戏主界面方格进行单个细胞状态的修改，可以通过点击"清空"按钮清除所有细胞；
- 游戏在放置好细胞后可以通过点击"开始"按钮进行细胞的自动繁衍，细胞繁衍的周期可以通过"繁衍周期"滑动条按钮进行控制，在细胞繁衍的过程中可以通过点击"暂停"按钮暂停细胞的繁衍。

项目部署

- 本项目部署在 Github Pages 上；
- 项目的 Git 远程仓库地址为 https://github.com/nezharen/GameOfLife/，分支为 gh-pages；
- 项目的 Github Pages 地址为 http://nezharen.github.io/GameOfLife/。

项目测试

- 本项目使用 QUnit 框架进行单元测试。点击在游戏主页面上方的"单元测试"选项，可以进行测试并查看测试页面。测试函数包含在 test.js 文件中。
- 项目测试部分主要针对游戏的初始化、算法逻辑及游戏状态正确性等进行了测试。具体来说，进行了六项测试：

- （1）测试了初始化函数，检查了地图矩阵初始状态、地图边界长度等数据的合法性；
- （2）测试了随机生成地图函数，检查执行随机函数 random 后，生成的细胞密度是否与输入密度值相符；
- （3）测试了不同细胞密度下，细胞执行一次进化操作后的状态正确性。测试函数输入不同的初始密度数值，包括了 0 和 100%这样的边界值，并通过随机函数生成初始地图，而后依据游戏规则进行测试；
- （4）在上面一项测试的基础上，测试了细胞执行多次进化操作的正确性。测试执行了 100 次进化操作，并在每次进化操作后检查正确性；
- （5）测试了一组特殊的输入数据。这组数据中细胞初始时分布在地图的边界（四角）上，主要以检查边界情况为目的；
- （6）由于游戏异步进行，因此进行了异步测试，使用 setTimeout 方法将进化操作延时进行并进行测试。

项目分工

- 张凯：结对编程中主要负责主程序界面设计及逻辑编写；
- 邹豪风：结对编程中主要负责检查；辅助部分逻辑、界面编写；负责单元测试。

项目反思

- 关于结对编程的感受

张凯：结对编程对于这种小型项目的开发具有比较大的意义。小型项目往往体系结构不大，在实现难度上相对较低，因此有更多的时间和精力去关注代码质量。但正是结对编程的名字决定了它的应用对象是一对开发者，因此其适用范围是非常窄的。当开发一个相对较大的项目，项目有十几个甚至几十个开发者时，再实行这种一人编码一人检查的模式将会造成很大的资源浪费。尤其是在当今的协作开发版本控制软件已经相对成熟的情况之下，结对编程相比之下恐怕没有比较强的竞争力。

邹豪风：结对编程是一个很好的主意。正如邹欣所说，结对编程能够增加我们的编程专注度，及时发现各种细节错误。实时的讨论也能够带来更加优化的算法、

更加简洁的程序代码。结对编程需要牺牲一个人的编程时间，而程序员的时间是宝贵的。但如果结对编程能够提高编程的效率，也不妨尝试之。

- 关于单元测试的理解

利用此次作业机会，我们接触了单元测试工具 QUnit。仔细了解后发现该工具仅仅是提供了一个测试平台，具体的单元测试，包括测试对象、测试函数、数据的设计依然必须由我们自己来做。

从我们的角度来看，单元测试是一种黑盒测试。测试人员应该首先站在一个“局外人“的角度，不在意函数内部如何实现，只关心函数执行后的返回值及程序中变量的状态是否正确。单元测试的好处是，如果进行全局测试，即使发现了错误，也往往难以查找。而进行单元测试，能够将偌大的程序拆分为粒度更小的部分，这有助于我们定位产生错误的函数。尽管编写、设计单元测试函数将会花费更多的精力，但是能够带来更加可靠、安全的软件。