电子科技大学字节跳动训练营学习总结

电子科技大学计算机科学与工程学院 王浩舟

一、Objective-C 的学习

早在正式开营前，我所在的 iOS3 组组长陈壬老师就发布了《Objective-C 程序设计》《Pro+Objective-C》等学习资料，这极大地帮助了我学习 Objective-C（下文简称 OC）语言。在 OC 部分的学习中，我主要选择了《Objective-C 程序设计》进行全篇学习。其实我在高中已有信息学奥赛的编程基础，在大学也是校 ACM 队的成员，可以说比同学们要更熟悉 C++语言，也因此为学习 OC 打下了基础。由于训练营和期末时间有部分重叠，加之训练营总体时间偏短，所以我自我感觉对于 OC 的学习只是到达了能够正常使用的地步，但仍然不清楚很多 OC 语言的细节（包括一些 OC 特有的声明实现原理、编译器对代码的处理等），我打算在训练营结束后保持继续学习的劲头。

我在训练营之前完全没有接触过 iOS 开发，因此我选择从学习资料开始学习。因为我有 C++基础，所以对于资料里的很多内容能够做到当场理解，对于不理解的地方也会实践求得真知。我在 OC 学习的末尾，融会贯通所学的知识，使用接口隔离的理念编写了一个 OC 语言版本的简单线段树实现并且成功运行，这在当时令我感到十分欣喜。

就我个人的体验来说，有 C++基础的同学在学习 OC 时，应该会学到 OC 里面向对象编程中的对象函数调用方式，因为在 C++中从来没有见过[Object Function: param1 :param2 :…]这样的结构；会学到 OC 的诸多新机制，比如@property、@synthesize 之类的声明以及对应的工作机制；会学到 OC 的面向对象设计理念，包括各种继承、依赖、interface 和 implementation 在程序架构中的角色等。

就我个人的学习而言，OC 和 C 语言的差别主要在于面向对象方面的设计，其余部分都很相似。相较于 C++语言，OC 语言给我的总体印象是更严谨、更优雅的。比如 OC 里面新建一个名称为 node 的 struct 结构体之后，若要在别处声明一个该结构体的新对象，则需要用 struct node xxx 这样的语句，这在 C++中是不

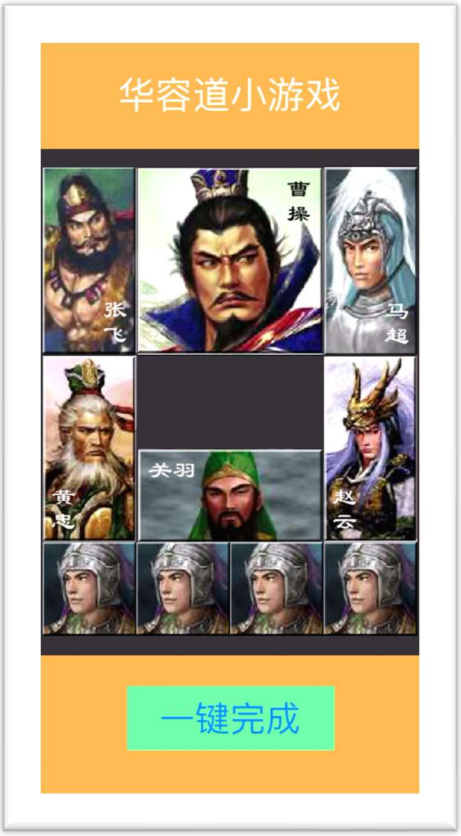
需要重复声明 struct 这个词的。此外，可能是由于苹果公司对代码相对更苛刻的要求，XCode 总是会强提醒代码中写得不符合标准的部分，尽管它不影响程序的正常运行，这也极大纠正了我的一些编程习惯。

二、iOS 的学习

有了 OC 语言基础，学习 iOS 开发就简单得多。作为 iOS 开发新手，我首先跟着《Objective-C 程序设计》的第 21 节“编写 iOS 应用程序”中所写的步骤开发出了一个非常简易的 iOS App（实际上这个程序只有一个 UILabel 控件和一个 UIButton 控件）。这个程序尽管简单，但是意义非凡，因为它成功教会了我 iOS App 开发中的界面设计、界面交互、事件绑定、代码层的界面控制等开发基础，也教会了我如何模拟器调试。后来，我通过搜索资料也学会了使用真机调试，这些新技术无疑都是令人振奋的学习成果。

在学会了上述技术后，我开始着手华容道游戏的开发。开发成果链接

(1) 界面设计



华容道的整体界面设计如右图所示，其主要包含的内容如下：

标题：华容道小游戏

游戏区域：10 个图形方块，每一个方块都有包括 4 个方向的、用于识别滑动手势并做出对应响应的 UISwipeGestureRecognizer。图形方块的图像来源于互联网。

一键完成按钮：按下后游戏自动基于游戏当前地图进行 BFS 搜索，并会自动基于游戏当前地图走完后续完成步骤。在此过程中玩家将无法操作游戏区域的方块或再次点击一键完成按钮。

(2) 代码编写

由于 storyboard 布局方式（拖动控件设计界面）不适合界面控件密集的情况（在本次开发中每个方块需要构建四个 UISwipeGestureRecognizer），所以我结合 storyboard 和代码布局两种方式，兼具了界面可视设计与代码布局的优势。具体实现方式是除了 UISwipeGestureRecognizer 以外的控件使用 storyboard 布局设计，而对于 10 个方块对应的 40 个 UISwipeGestureRecognizer 都在代码内实现布局。

代码所涉及到的算法内容主要包括自动求解的 BFS 算法、游戏地图哈希算法、方块移动合法性判断、游戏结束判断等。实现难度都不大，但我学到了 NSMutableDictionary、NSMutableArray、NSString、NSValue 等类的基本特性。在本次开发实践中，NSMutableDictionary 主要用于游戏地图哈希算法；NSMutableArray 主要实现 BFS 的队列，NSValue 主要用于把程序内单独声明的 struct 存进 NSMutableArray 中。

除了算法外，代码还涉及到对界面的操作部分。由于 iOS 开发是面向对象的开发，所以在自动求解中根据解操作对应控件对象这部分的代码就稍显繁琐（同样繁琐的是 viewDidLoad 函数中声明 40 个 UISwipeGestureRecognizer 的部分）。不过这一部分的开发也让我厘清了 iOS 开发的 UIView 的很多特性，比如控件位置可以用包括四个参数的 CGRect 类表示等。

(3) 遇到的问题和解决方式

在华容道的开发中我遇到了主线程循环中操作控件并不会对应显示在屏幕上的问题。具体表现为：我在一键完成按钮对应的 IBAction 函数中编写了一个 For 循环，这个循环用于在用户界面中可视化展示 BFS 搜得的结果，但实践中发现 For 循环代码中确实改变了图像方块的 Rect 参数（即使用 setFrame 函数），但这并不会展现在用户界面中，而是会在 For 循环结束后全部一同变更。在用户侧，程序表现为假死一段时间后突然方块都移动到游戏结束的位置，并提示游戏结束。

这无疑是完全不符合设计理念的。所以我首先考虑是不是这个 For 循环导致 IBAction 函数没有返回，所以阻塞了主线程，进而导致 UI 绘制的问题。基于这个考虑，我为 For 循环单独开设了一个子线程，但在调试中发现无用且报错。经

过查阅资料后才发现 iOS 开发只能在主线程中更新界面，不允许在子线程更新。这个方案只好作罢。

因此，For 循环仍然位于主线程的 IBAction 函数内。我再次考虑在 For 循环内异步调用强制重绘界面的函数，也就是 setNeedsDisplay、setNeedsLayout 和 layoutIfNeeded 等函数，但都无用，作罢。

我继续尝试使用 dispatch_after 构建动画更新队列，也就是按照等差数列构建一系列时间，然后把各个 setFrame 函数结合对应时间放入队列中。但实践中发现这种实现方法只能实现动画前几步的正确显示，然后就会几步合为一步进行方块移动，这无疑也不符合设计理念。我自己学习后大致判断这是因为 iOS 界面的刷新周期导致的，在向陈壬老师请教后也知道了 dispatch_after 函数并不能实现完全准时的队列，只好作罢。

最后我学习了 iOS 的动画机制，把 setFrame 函数放在 animateWithDuration 函数中，终于实现了正确的动画效果。

三、总结

这是我第一次参与 iOS 开发。很庆幸自己是在字节大牛 mentor 的带领下入了 iOS 开发的门，少走了很多弯路。iOS 开发着实有趣，我自己实现的简单动画效果更是在我以前的开发中从没有过的新奇体验。在结营后，我确信自己会继续保持对 iOS 开发的高度热情，深入学习 OC 和 iOS。立一个 flag：我希望有朝一日能有成为 iOS 独立开发者的实力。