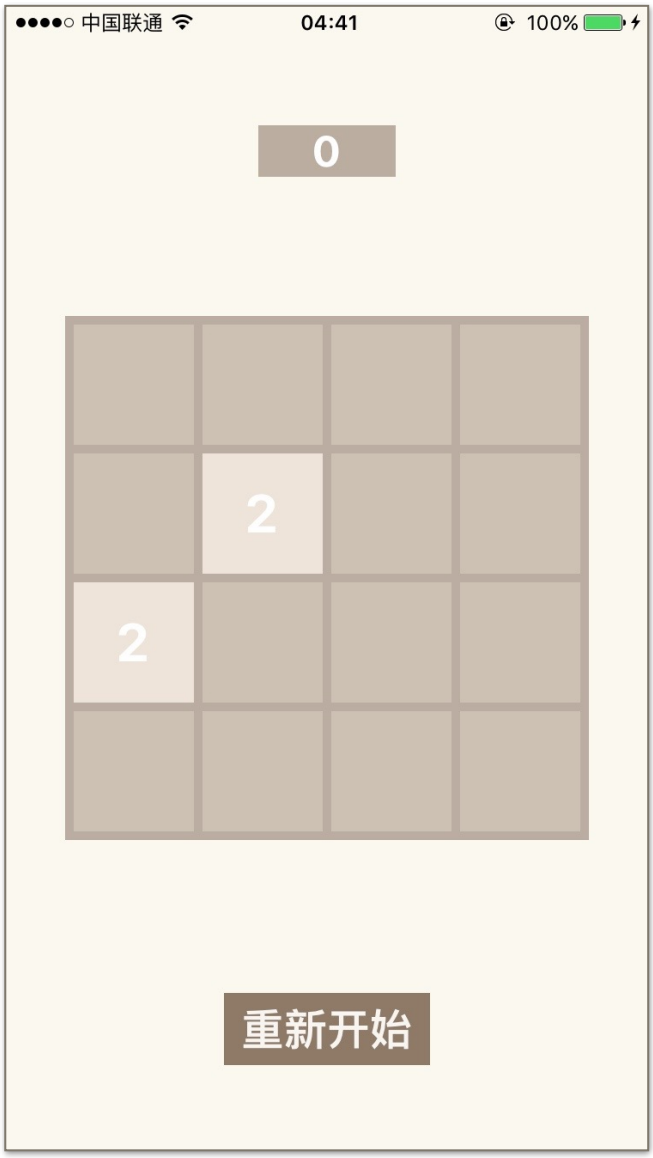
Hello, iOS
特别篇 2048小游戏[1]

随着开学越来越近，手头的事情也越来越多，所以这个系列没有继续更新，在此表示歉意。Swift的部分在开学后可能才会恢复更新。



前一段时间iOS组内曾经讨论过开学之后的招新活动，当时我们萌生了希望写个小游戏现场demo的形式调动同学们对iOS开发的兴趣的想法。在很多种小游戏之间选择了很久，最终选择了实现难度比较小、同时大家比较有兴趣的2048。4号晚上我拿到了第一版[2]，原本决定在开学后给大家做讲座之后写一个相关的教程。在我上手这套代码之后，在第一版的基础之上做了一些交互上的细微修改，形成了现在的第二版，并且迫不及待地希望给大家分享这个小应用。

这个还原度非常高的副本就是本节的主要内容。我们从审查代码的程序员这个角度[3]对程序代码进行一个简要的分析。

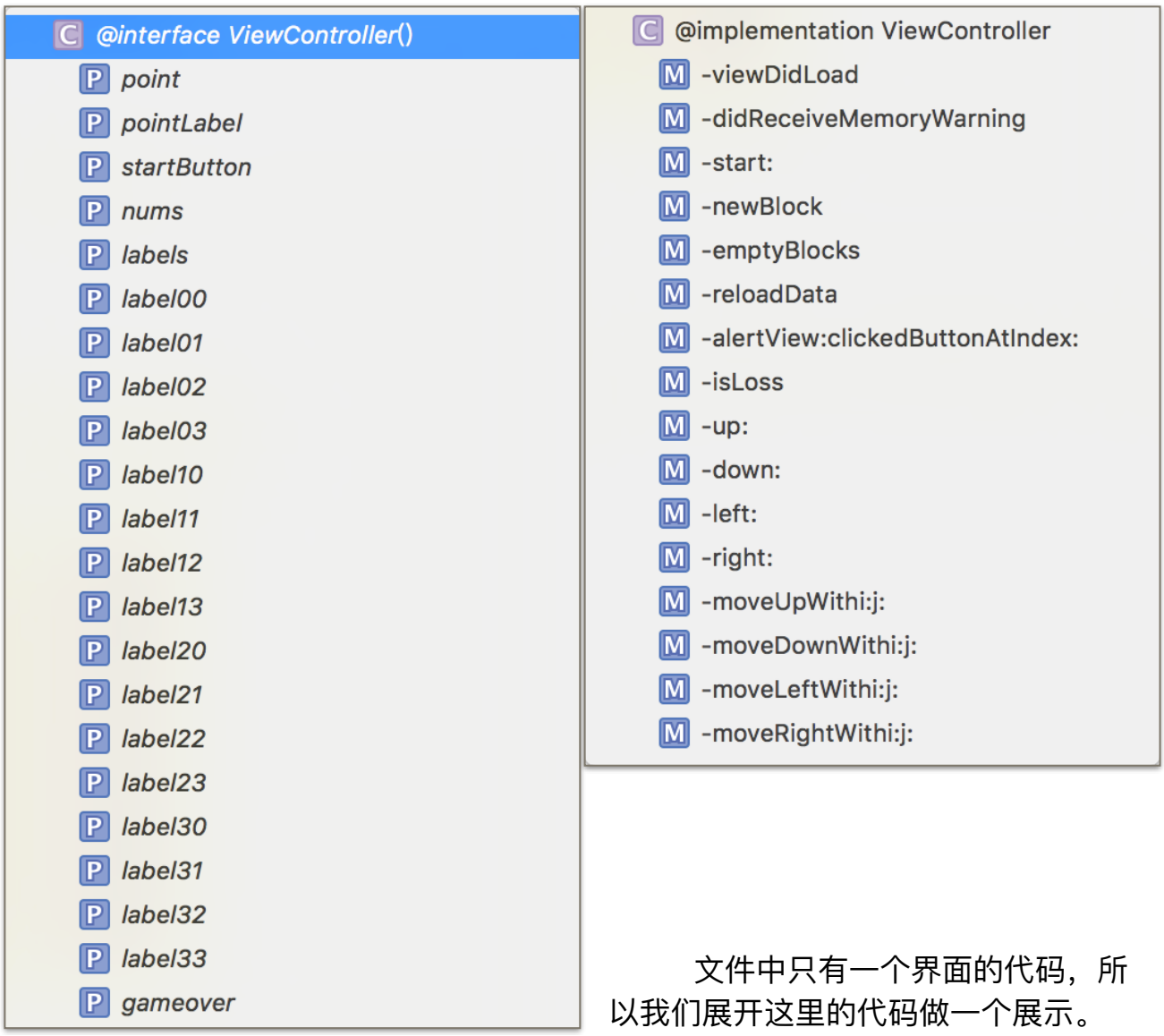
我们审查代码的时候，可以从这里了解整个文件的结构，其中包括属性、方法等基本信息。

---

[1] 2015年9月5日，基于OS X 10.11 beta 8， iOS 9 beta 5， Xcode 9 beta 5.

[2] 在此感谢创联iOS组组长高铭提供了本项目的组内开源权限，其他用途请联系原作者高铭。

[3] 在实际的开发当中，一个程序员不仅作为代码的编写者，还有可能参与到帮助其他程序员的工作当中。这时我们应该以代码审查者的角度观察其他人的代码，这种能力也是优秀程序员应该具备的。



文件中只有一个界面的代码，所以我们展开这里的代码做一个展示。

在大的项目中，我们建议把功能分块化处理。这份代码中就看到了功能分块的痕迹。让我们从刚刚进入界面调用的viewDidLoad方法开始[4]。

```
40 - (void)viewDidLoad {
41     [super viewDidLoad];
42     self.labels = @[@[self.label00, self.label01, self.label02, self.label03],
43                     @[self.label10, self.label11, self.label12, self.label13],
44                     @[self.label20, self.label21, self.label22, self.label23],
45                     @[self.label30, self.label31, self.label32, self.label33]];
46     self.nums = [[NSMutableArray alloc] init];
47     for (int i = 0; i <= 3; i++) {
48         [self.nums addObject:[[NSMutableArray alloc] init]];
49         for (int j = 0; j <= 3; j++) {
50             [self.nums[i] addObject:[NSMutableString stringWithString:@""]];
51         }
52     }
53     self.point = 0;
54     self.pointLabel.text = [NSString stringWithFormat:@"%ld", (long)self.point];
55     [self start:self];
56 }
```

[4] 这里涉及到了界面的生命周期的概念，我们暂时跳过。

这个方法里面，做了16个文本框的初始化，同时把它们添加到了一个线性表中管理。另外对于分数也做了初始化。

```
62 - (IBAction)start:(id)sender {
63     self.gameover.hidden = YES;
64     for (int i = 0; i <= 3; i++) {
65         for (int j = 0; j <= 3; j++) {
66             self.nums[i][j] = [NSMutableString stringWithString:@""];
67         }
68     }
69     self.point = 0;
70     self.pointLabel.text = [NSString stringWithFormat:@"%ld", (long)self.point];
71     [self newBlock];
72     [self newBlock];
73 }
74
75 - (void)newBlock {
76     NSMutableArray *emptyArr = [self emptyBlocks];
77     if (emptyArr.count == 0) {
78         return;
79     }
80     long n = random() % emptyArr.count;
81     long m = random() % 10;
82     if (m <= 6) {
83         [emptyArr[n] setString:@"2"];
84     } else {
85         [emptyArr[n] setString:@"4"];
86     }
87     [self reloadData];
88 }
```

开始方法绑定在了重新开始按钮上，展现的是初始化的过程。下面的新书块方法，模拟了可能生成2和4的方法，其中random()生成了随机数[5]。我们注意到，76行引用了emptyBlocks方法。对于没有见过的方法，想要快速定位可以使用代码搜索，快捷键是Command+F，输入emptyBlcoks，对应的内容将会高亮处理。

```
90 - (NSMutableArray *)emptyBlocks {
91     NSMutableArray *emptyArr = [[NSMutableArray alloc] init];
92     for (int i = 0; i <= 3; i++) {
93         for (int j = 0; j <= 3; j++) {
94             if ([self.nums[i][j] isEqualToString:@""]) {
95                 [emptyArr addObject:self.nums[i][j]];
96             }
97         }
98     }
99     return emptyArr;
100 }
```

哦，原来这个方法就在下面[6]。

---

[5] 你可以思考一下这个方法里生成的随机数起到了什么作用。

[6] 源代码中，emptyBlocks方法就在newBlock方法的下面。但是在大段的代码中，寻找特定的代码的确是非常困难的。

```
102 - (void)reloadData {
103     for (int i = 0; i <= 3; i++) {
104         for (int j = 0; j <= 3; j++) {
105             ((UILabel *)self.labels[i][j]).text = [NSString stringWithFormat:@"%@", self.nums[i][j]];
106             int n = [((UILabel *)self.labels[i][j]).text intValue];
107             switch (n) {
108                 case 2:
109                     ((UILabel *)self.labels[i][j]).backgroundColor = [UIColor colorWithRed:238.0/255.0
                            green:228.0/255.0 blue:218.0/255.0 alpha:1.0];
110                     break;
111                 case 4:
112                     ((UILabel *)self.labels[i][j]).backgroundColor = [UIColor colorWithRed:237.0/255.0
                            green:224.0/255.0 blue:200.0/255.0 alpha:1.0];
113                     break;
```

reloadData方法包含着刷新界面内容的相关代码。其中这个switch...case...的内容包含了各种数字的背景颜色信息，由于代码量比较大，所以不一一列举了。

```
151     self.pointLabel.text = [NSString stringWithFormat:@"%ld", (long)self.point];
152     if ([self isLoss]) {
153         self.gameover.hidden = NO;
154         UIAlertView *av = [[UIAlertView alloc] initWithTitle:@"Game Over" message:[NSString
                stringWithFormat:@"您的分数是：%ld", (long)self.point] delegate:self cancelButtonTitle:@"再来一局"
                otherButtonTitles:nil];
155         [av show];
156     }
```

这里面包含了我做出的一些细微修改。在游戏结束之后(判断语句中的isLoss满足)，会弹出一个提示框显示分数并再来一局。注意av[7]的delegate是self，所以对于这个ViewController的委托应该设置为UIAlertViewDelegate。并在第155行显示这个提示框。

```
159 - (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex {
160     if (buttonIndex == 0) {
161         [self start:nil];
162     }
163 }
```

所谓的委托，体现在接下来的方法。对于一个提示框来说，它的按钮是具有一定的顺序的。本例中的提示框只有一个按钮，所以其代号[8]为0，对于0号按钮相应start这个方法。

```
197 - (IBAction)down:(id)sender {
198     for (int j = 0; j <= 3; j++) {
199         for (int i = 2; i >= 0; i--) {
200             [self moveDownWithi:i j:j];
201         }
202     }
203     [self newBlock];
204 }
```

之后我们可以想象到，这个应用需要辨识4中手势，即向上下左右划。理论上讲，这四个方法是较为一致的，所以我们列举向下这个例子。

---

[7] UIAlertView的简写。

[8] 即AlertView的属性buttonIndex.

```
- (void)moveDownWithi:(int)i j:(int)j {
    if (i == 3) {
        return;
    } else if ([self.nums[i][j] isEqualToString:@""]) {
        return;
    } else if ([self.nums[i+1][j] isEqualToString:@""]) {
        [self.nums[i+1][j] setString:[NSString stringWithFormat:@"%@", self.nums[i][j]]];
        [self.nums[i][j] setString:@""];
        [self moveDownWithi:i+1 j:j];
    } else if ([self.nums[i+1][j] isEqualToString:self.nums[i][j]]) {
        [self.nums[i+1][j] setString:[NSString stringWithFormat:@"%d", 2*[self.nums[i+1][j] intValue]]];
        self.point = self.point + [self.nums[i+1][j] intValue];
        [self.nums[i][j] setString:@""];
        return;
    } else {
        return;
    }
}
```

刚才的手势方法中，调用了另一个移动的方法，这个方法应该是这个程序的算法核心。这段代码的分析交给读者处理，需要注意的是，这里发生了文本框中的字符和数字运算之间的转换。

```
- (BOOL)isLoss {
    NSMutableArray *emptyArr = [self emptyBlocks];
    if (emptyArr.count != 0) {
        return NO;
    } else {
        for (int i = 0; i <= 2; i++) {
            for (int j = 0; j <= 3; j++) {
                if ([self.nums[i][j] isEqualToString:self.nums[i+1][j]]) {
                    return NO;
                }
            }
        }
        for (int j = 0; j <= 2; j++) {
            for (int i = 0; i <= 3; i++) {
                if ([self.nums[i][j] isEqualToString:self.nums[i][j+1]]) {
                    return NO;
                }
            }
        }
    }
    return YES;
}
```

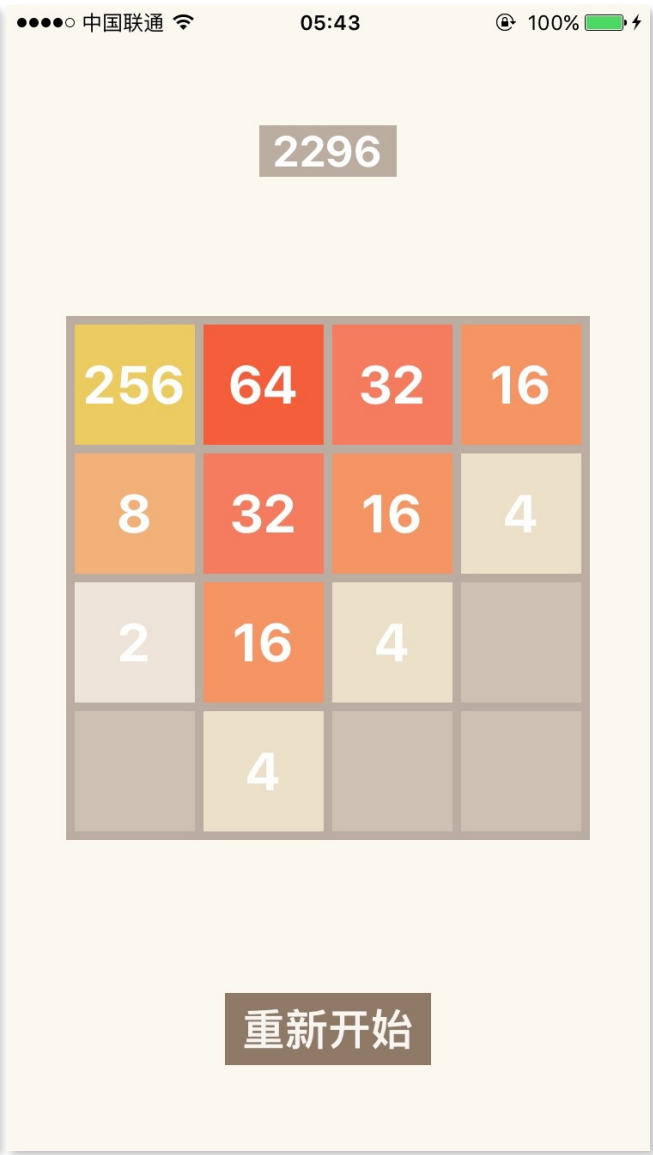
最后是判定游戏结束的方法。算法核心是屏幕内数字必须是占满的，同时不能有其他可以移动的元素，则游戏结束。

这个应用的代码未来可能会放在GitHub上，具体的信息可以关注我和高铭的GitHub主页：

Everyb0dyLies： https://github.com/Everyb0dyLies

ZHRMoe： https://github.com/ZHRMoe


●●●●○ 中国联通
05:42
100%
936
128 16 8 2
Game Over
您的分数是：936
再来一局
4 2 4 2
重新开始


●●●●○ 中国联通
05:43
100%
2296
256 64 32 16
8 32 16 4
2 16 4
4
重新开始

特别篇 **EOF**