

|  |  |  |  |
| --- | --- | --- | --- |
| 设计题目 | Swift语言做计分器 | | |
| 指导教师 | 朝力萌 | 职称 | 主任 |
| 姓 名 | 白才艺 | | |
| 学 号 | 20131105761 | | |
| github | https://github.com/2013110 | | |

**内蒙古师范大学计算机与信息工程学院**

|  |  |  |  |
| --- | --- | --- | --- |
| **实习题目** |  | | |
|  | | |
| **指导教师** |  | **职 称** |  |
| **学生姓名** |  | | |
| **学 号** |  | | |
| **日 期** |  | | |

基于Swift语言做计分器研究报告

摘要: Swift是一款易学易用的编程语言，而且它还是第一套具有与脚本语言同样的表现力和趣味性的系统编程语言。Swift的设计以安全为出发点，以避免各种常见的编程错误类别。通过学习，在xcode软件环境下实现了对简单方块游戏作计分器运行界面开发。

关键词：Swift语言；计分器；Xcode.

Calculator Research Report Based on Swift language

Swift is an easy to learn and easy to use programming language, but it is also the first set of the same performance with the scripting language and interesting system programming language. Swift design to safety as the starting point, to avoid a variety of common programming error categories. Through learning, in the Xcode software environment to achieve a simple box game for the development of the operation interface.

Key words: Swift language; score indicator; Xcode.

# 1.引言

Swift语言做计分器是本学期移动平台应用开发第二次作业，我做的是简单方块游戏计分器。Swift是第一个既满足工业标准又像脚本语言一样充满表现力和趣味的编程语言。它支持代码预览，这个革命性的特性可以允许程序员在不编译和运行应用程序的前提下运行 Swift 代码并实时查看结果。Swift 是一种新的编程语言，用于编写 iOS 和 OS X 应用。Swift 结合了 C 和 Objective-C 的优点并且不受C兼容性的限制。Swift 采用安全的编程模式并添加了很多新特性，这将使编程更简单，更灵活，也更有趣。Swift 将现代编程语言的精华和苹果工程师文化的智慧结合了起来。编译器对性能进行了优化，编程语言对开发进行了优化，两者互不干扰，鱼与熊掌兼得。Swift 既可以用于开发“hello, world”这样的小程序，也可以用于开发一套完整的操作系统。通过学习，在四周课时时间内做了简易计分器，并做了如下报告。

# 2. Swift语言做计分器主要函数

swift函数和c#,js的写法大致一直，但是与object-c写法有很大不同点。，该语言主要用的函数有：

1）func 函数名（ 参数变量：类型 ，参数变量：类型...）｛｝

说明:

1: func 是函数关键字  
 2：｛｝ 函数体

3: 参数变量是默认常量类型，不能在函数函数体里面直接修改  
 即 func A (value:String)  与 func A (let value:String)写法是相同的，即value 是常量。

2）var 参数变量 ：类型 表示参数变量是可变变量，不是常量，可以任意修改

var sumInMemory: Double = 0.0

var sumSoFar: Double = 0.0

var factorSoFar: Double = 0.0

var pendingAdditiveOperator = ""

var pendingMultiplicativeOperator = ""

var waitingForOperand = true

var displayValue: Double {

3）函数类型表示一个函数、方法或闭包的类型，它由一个参数类型和返回值类型组成，中间用箭头->隔开：

 func calculate(rightOperand: Double, pendingOperator: String) -> Bool {

**3． Swift做计算器依附Xcode软件**

Xcode 是运行在操作系统Mac OS X上的集成开发工具（IDE），由苹果公司开发。Xcode是开发OS X和iOS应用程序的最快捷的方式。Xcode 具有统一的用户界面设计，编码、测试、调试都在一个简单的窗口内完成。

**3.1 Xcode快捷键**

文件

CMD + N: 新文件； CMD + SHIFT + N: 新项目； CMD + O: 打开；

CMD + S: 保存； CMD + SHIFT + S: 另存为；

CMD + W: 关闭窗口； CMD + SHIFT + W: 关闭文件

编辑

CMD + [: 左缩进；CMD + ]: 右缩进；

CMD + CTRL + LEFT: 折叠；CMD + CTRL + RIGHT: 取消折叠；

CMD + CTRL + TOP: 折叠全部函数；CMD + CTRL + BOTTOM: 取消全部函数折叠；

CTRL + U: 取消全部折叠；

CMD + D: 添加书签； CMD + /: 注释或取消注释；

CTRL + .: 参数提示； ESC: 自动提示列表

调试

CMD + \: 设置或取消断点；

CMD + OPT + \: 允许或禁用当前断点；

CMD + OPT + B: 查看全部断点；

CMD + RETURN: 编译并运行（根据设置决定是否启用断点）；

CMD + R: 编译并运行（不触发断点）；

CMD + Y: 编译并调试（触发断点）；

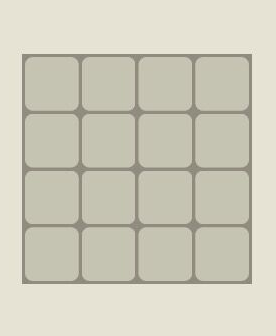
CMD + SHIFT + RETURN: 终止运行或调试；

CMD + B: 编译；

CMD + SHIFT + K: 清理；

# 3. Swift语言做计分器运行结果



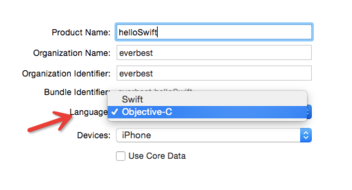


**4.创建过程与主要代码详解如下：**

1）新打开xcode，新建一个工程项目



2）在语言选择这里，现在新增了一个swift的选项，当然我们现在要选择swift，然后继续。



3）xcode”开发工具，点击“File”->“New”->“Project”项来创建一个新的工程。接着在打开的“创建工程”向导界面中，选择“OS X”系统中的“Application”项，选中“Command Line Tool”项，点击“Next”按钮来创建一个基于OS X系统的命令行应用程序。

4）输入“Project Name”（工程名称）和“Language”（语言），选择使用“swift”语言进行创建应用程序，点击“Next”按钮。直接点击“Create”按钮创建应用程序。

**4.1 主要代码及注释**

class NumbertailGameController : UIViewController {

var demension : Int //2048游戏中每行每列含有多少个块

var threshold : Int //最高分数，判断输赢时使用

let boardWidth: CGFloat = 260.0 //游戏区域的长度和高度

let thinPadding: CGFloat = 3.0 //游戏区里面小块间的间距

let viewPadding: CGFloat = 10.0 //计分板和游戏区块的间距

let verticalViewOffset: CGFloat = 0.0 //初始化属性

int(demension d : Int , threshold t : Int) {

demension = d < 2 ? 2 : d

threshold = t < 8 ? 8 : t

super.init(nibName: nil, bundle: nil)

view.backgroundColor = UIColor(red : 0xE6/255, green : 0xE2/255, blue : 0xD4/255, alpha : 1)

}

}

@IBAction func setupGame(sender: UIButton) {

let game = NumbertailGameController(demension: 4 , threshold: 2048)

self.presentViewController(game, animated: true , completion: nil)

}

import UIKit

class GamebordView : UIView {

var demension : Int //每行(列)区块个数

var tileWidth : CGFloat //每个小块的宽度

var tilePadding : CGFloat //每个小块间的间距

//初始化

init(demension d : Int, titleWidth width : CGFloat, titlePadding padding : CGFloat, backgroundColor : UIColor, foregroundColor : UIColor ) {

demension = d

tileWidth = width

tilePadding = padding

let totalWidth = tilePadding + CGFloat(demension)\*(tilePadding + tileWidth)

super.init(frame : CGRectMake(0, 0, totalWidth, totalWidth))

self.backgroundColor = backgroundColor

}

}

override func viewDidLoad() {

super.viewDidLoad()

setupGame()

}

func setupGame(){

let viewWidth = view.bounds.size.width

let viewHeight = view.bounds.size.height

func xposition2Center(view v : UIView) -> CGFloat{

let vWidth = v.bounds.size.width

return 0.5\*(viewWidth - vWidth)

}

func yposition2Center(order : Int , views : [UIView]) -> CGFloat {

assert(views.count > 0)

let totalViewHeigth = CGFloat(views.count - 1)\*viewPadding +

views.map({$0.bounds.size.height}).reduce(verticalViewOffset, combine: {$0 + $1})

let firstY = 0.5\*(viewHeight - totalViewHeigth)

var acc : CGFloat = 0

for i in 0..<order{

acc += viewPadding + views[i].bounds.size.height

}

return acc + firstY

}

//获取具体每一个区块的边长，块数

let width = (boardWidth - thinPadding\*CGFloat(demension + 1))/CGFloat(demension)

let gamebord = GamebordView(

demension : demension,

titleWidth: width,

titlePadding: thinPadding,

backgroundColor: UIColor(red : 0x90/255, green : 0x8D/255, blue : 0x80/255, alpha : 1),

foregroundColor:UIColor(red : 0xF9/255, green : 0xF9/255, blue : 0xE3/255, alpha : 0.5)

)

let views = [gamebord]

//

var f = gamebord.frame

f.origin.x = xposition2Center(view: gamebord)

f.origin.y = yposition2Center(0, views: views)

gamebord.frame = f

view.addSubview(gamebord)

}

func setColor(backgroundColor bgcolor : UIColor, foregroundColor forecolor : UIColor){

self.backgroundColor = bgcolor

var xCursor = tilePadding

var yCursor : CGFloat

for \_ in 0..<demension{

yCursor = tilePadding

for \_ in 0..<demension {

let tileFrame = UIView(frame : CGRect(x: xCursor, y: yCursor, width: tileWidth, height: tileWidth))

tileFrame.backgroundColor = forecolor

tileFrame.layer.cornerRadius = 8

addSubview(tileFrame)

yCursor += tilePadding + tileWidth

}

xCursor += tilePadding + tileWidth

}

}

**5.小结**

本文中简要阐述了在Xcode软件下通过Swift语言做简易计分器的主要流程与对Xcode软件、Swift语言 的概述。感谢老师提供的良好上机环境与耐心指导，通过课程学习使我对Swift语言更加熟悉并能对其简单运用，收获颇多

**[参考文献]：**

### [1] <http://www.swiftv.cn/> [中国最大的Swift视频学习站](https://www.baidu.com/link?url=YWfJag2HG06GNbHmb9ljkZYnMWqACKJzaxcAQXjGNqu&wd=&eqid=ea5c3fd2000276ce000000035774bfa2)