实验 4

创建简单计算器

实验结构

- 使用 Swift 语言创建简单计算器

实验目标

本实验结束后，你将能够：

- 在 Xcode 中使用 Swift 语言创建简单的四则运算计算器

导论

Objective-C 是 iOS 应用编写中使用的主要编程语言。Objective-C 是 C 编程语言的超集。它继承了 C 的语法和语句。

Swift 的属性和语法继承自 Objective-C。Swift 添加了新数据类型和控制流格式，能比 Objective-C 编程更为简练。

实验：创建简单计算器

Swift 同苹果 Cocoa Touch 框架一同工作，包含在 Xcode 6 中。Xcode 6 使用 Objective-C、C++、C 和 Swift 编程，让其能够运行在单个程序之内。

背景

假设你要在 Xcode 中创建一个简单四则运算计算器。为此，你希望使用 Swift 语言。本实验中，你将创建出这一计算器应用。

实验准备

开始实验之前，你需要有：

- 安装有 Xcode 的 Mac OS (developer.apple.com/xcode/downloads)
- 苹果开发者帐户

实验：推荐解决方案

1. 启动 Xcode。在 **Welcome to Xcode** 界面上（参见图 1），选择 **Create a new Xcode project** 选项，开启新项目创建。

图 1: 选择 Create a New Xcode Project

2. 在 **Choose a template for your new project** 界面，选择 **Single View Application** 模板，并点 **Next**，如图 2 所示：

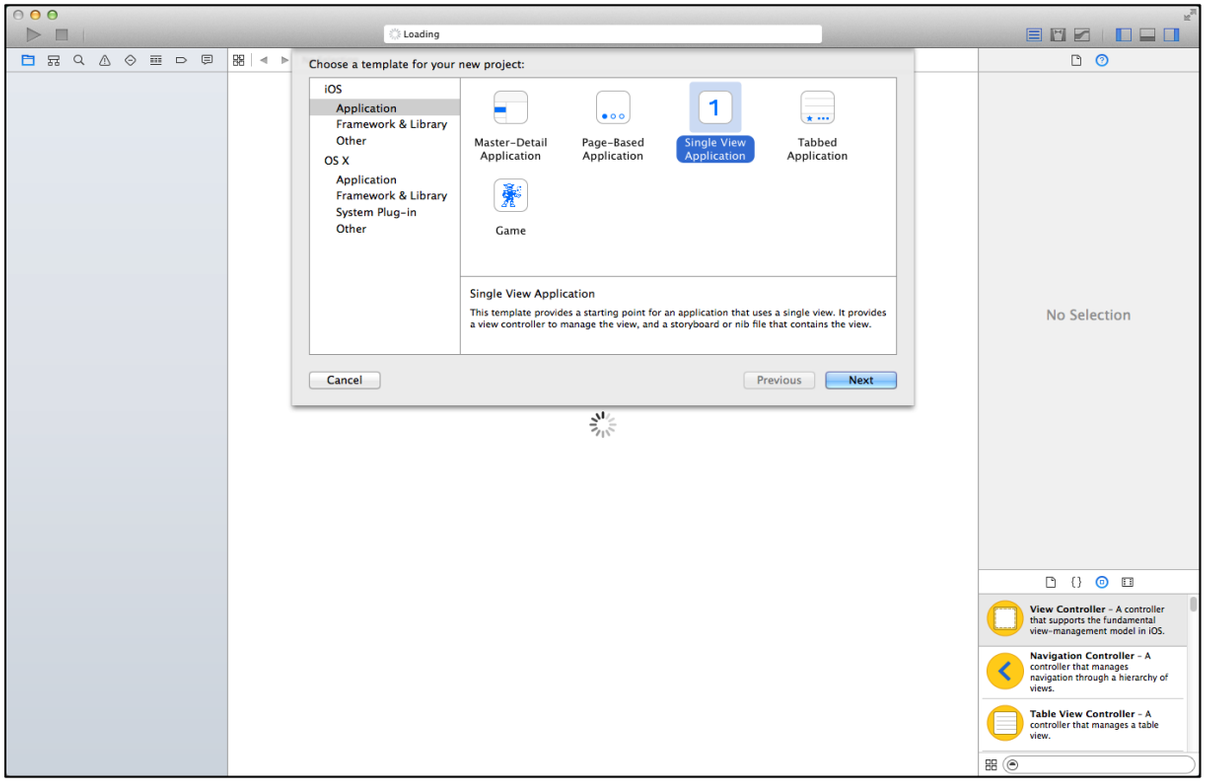


图 2: 选择 Single View Application 模板

3. 在 **Choose options for your new project** 界面上（参见图 3），键入项目名为 **SimpleCal**，**Language** 处选择 **Swift**，然后点 **Next**。

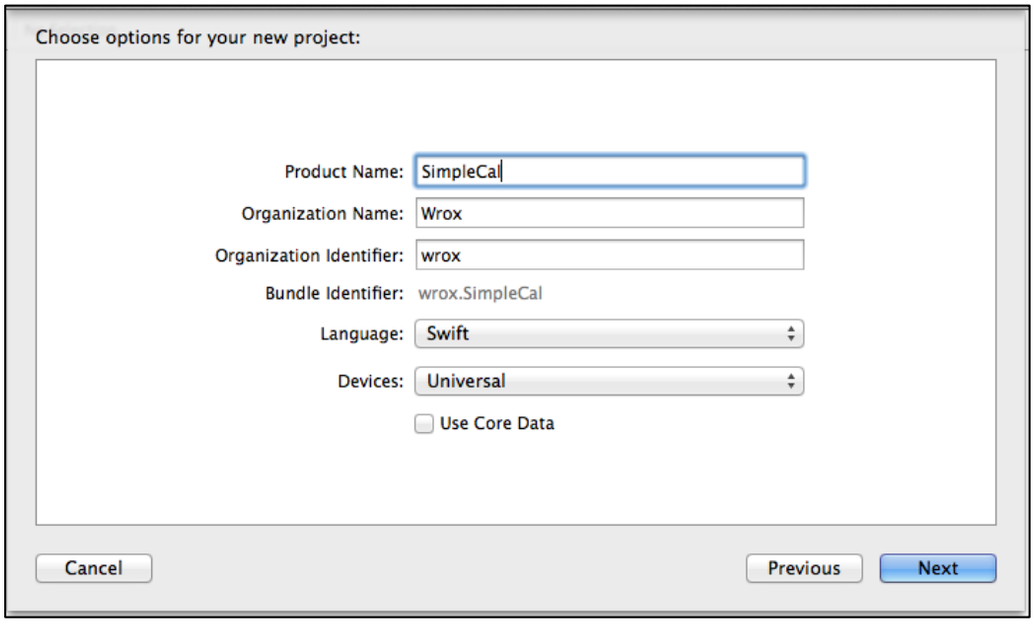


图 3: 输入新项目选项

4. 在下一个界面上（参见图 4），选择保存项目的位置，然后点 **Create** 来创建。

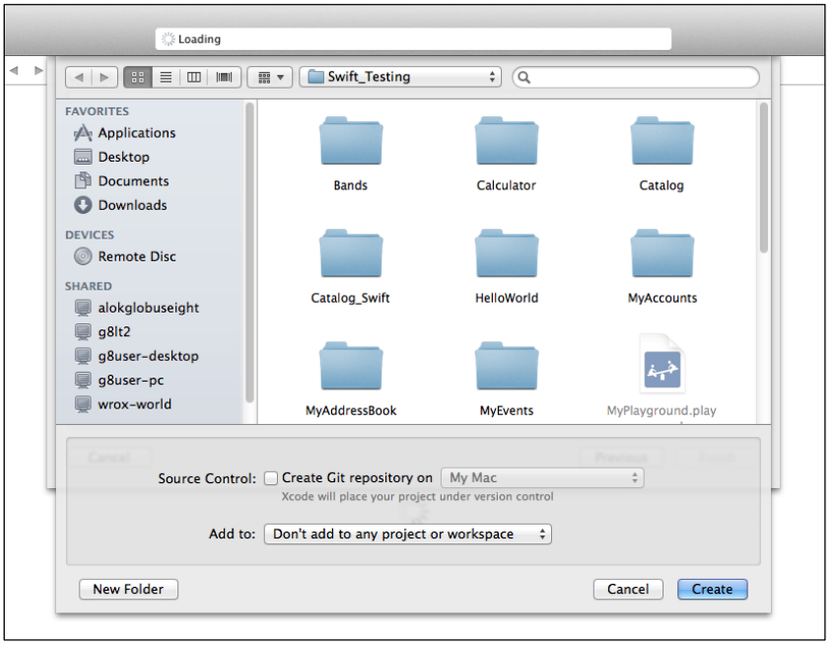


图 4: 保存项目，然后点击创建

5. Swift 的一般格式会显示出来，如图 5 所示。Swift 中有 `AppDelegate.swift`、`ViewController.swift` 和 `Main.storyboard` 文件，作为编码的主文件。

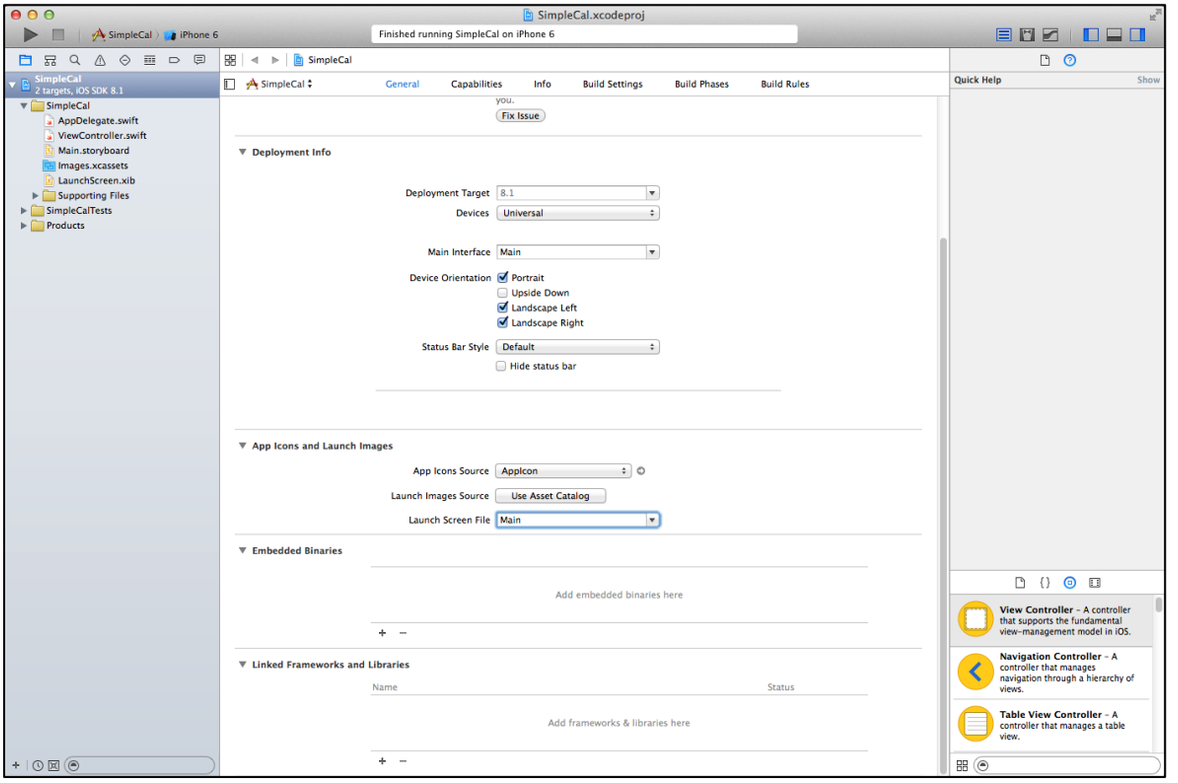


图 5: Swift 的一般格式

6. 选择 `Main.storyboard` 文件，使用 **wAny:hAny** 选项调整 Storyboard 尺寸，如图 6 所示：

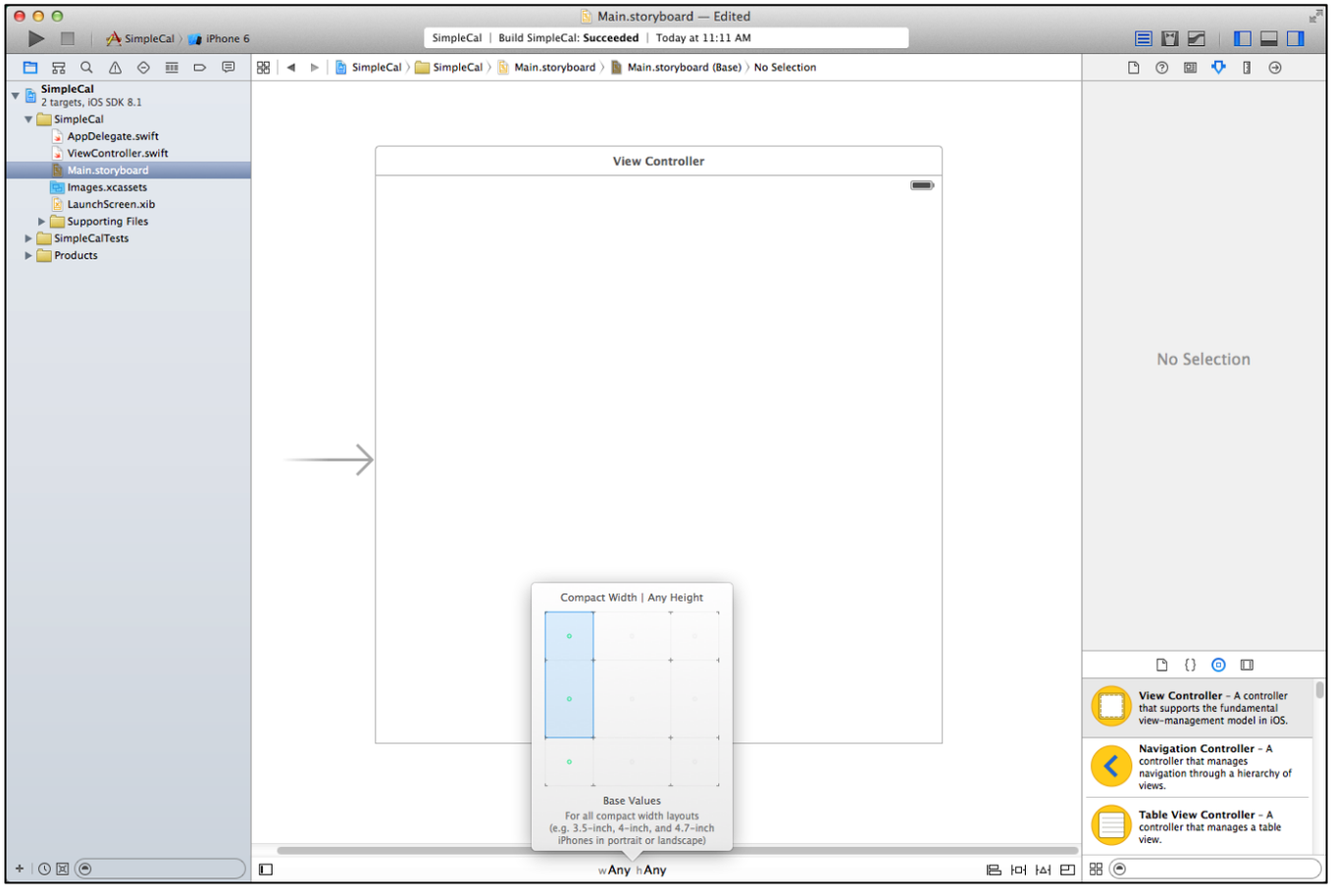
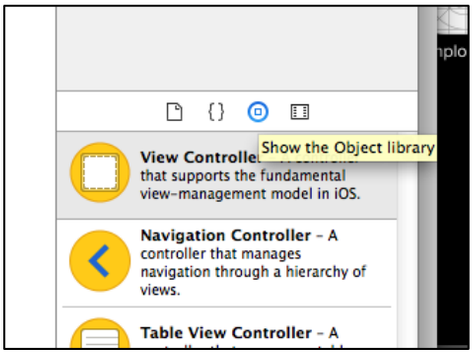


图 6: 调整 Storyboard 尺寸

7. 在实用工具面板中选择库选择器栏，通过选择标签来设计计算器的前端用户界面，如图 7 所示：



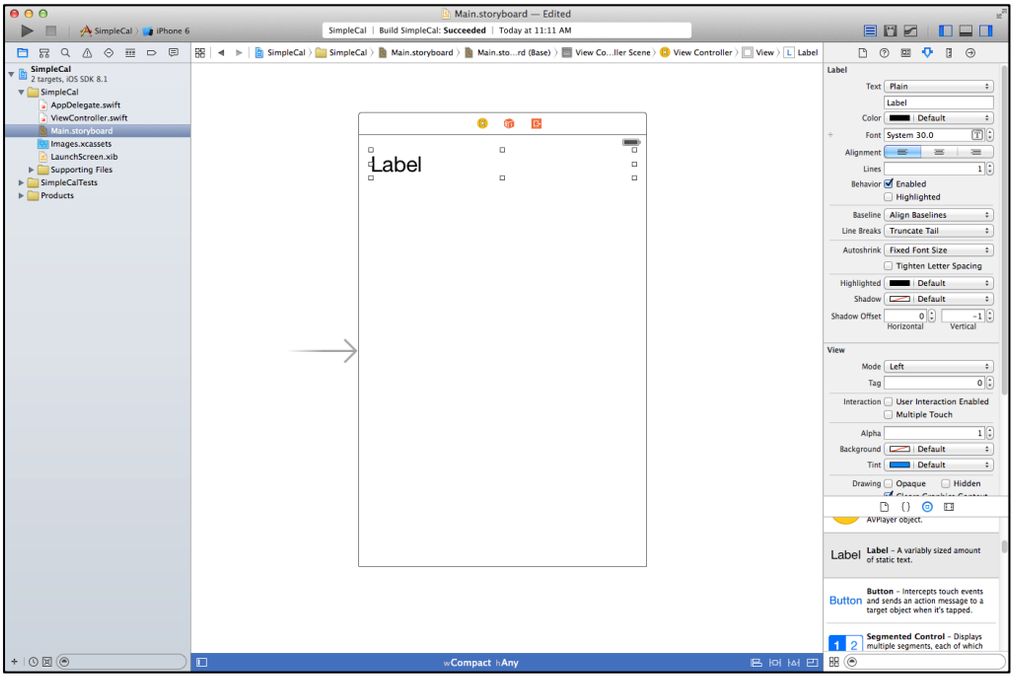
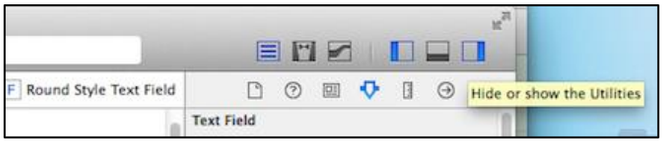


图 7: 选择 UILabel，用于计算器输出界面

8. 下面用计算器按钮填充 Storyboard。要添加按钮到计算器，你只需要将按钮从实用工具面板拖放到 Storyboard，如图 8 和图 9 所示：



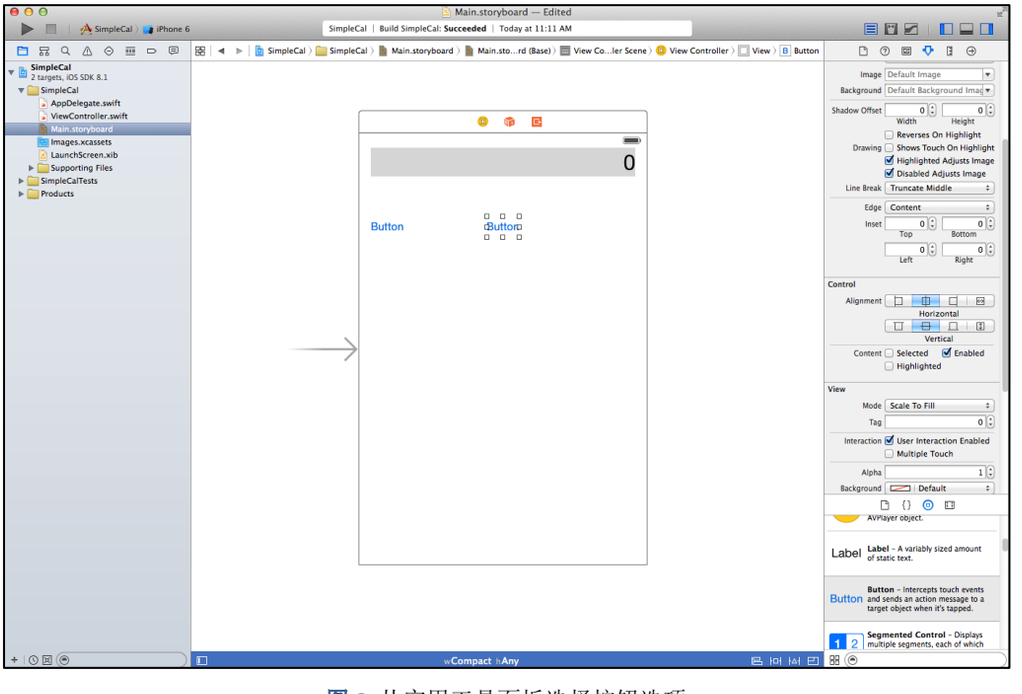


图 8: 从实用工具面板选择按钮选项

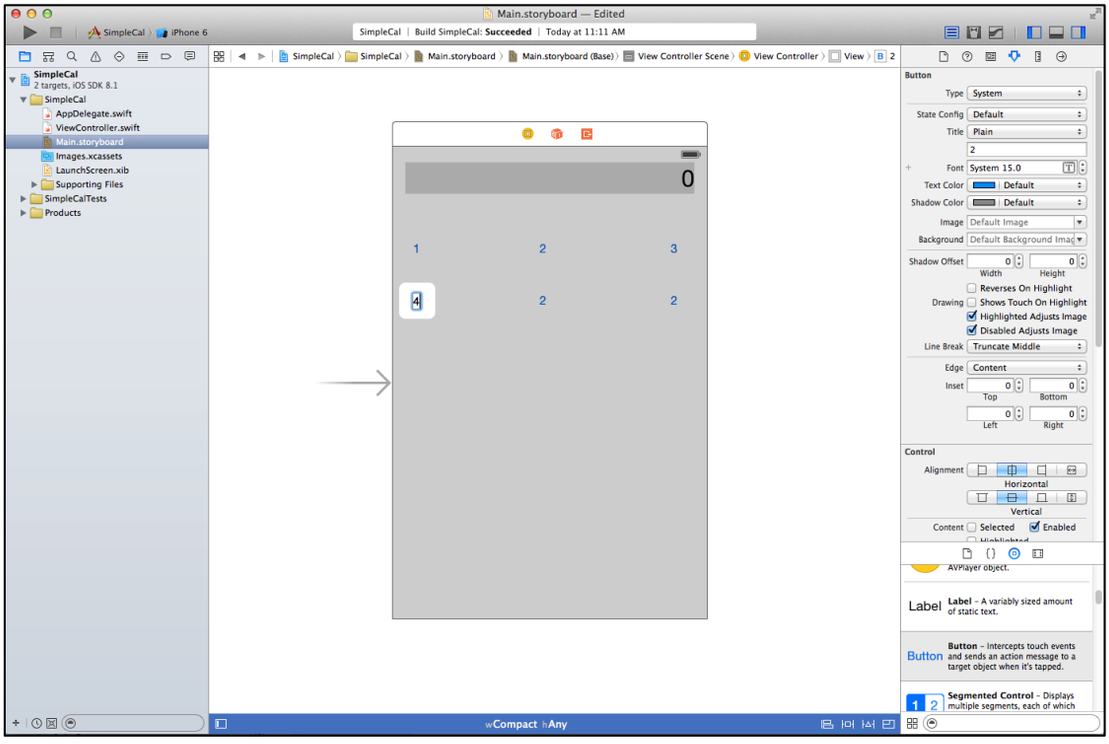


图 9: 将按钮拖放到 Storyboard 并为其命名

9.按照这种方式设计四个运算按钮，用于计算器的四则运算功能。使用属性检查器来调整按钮的文本大小（参见图 10）。

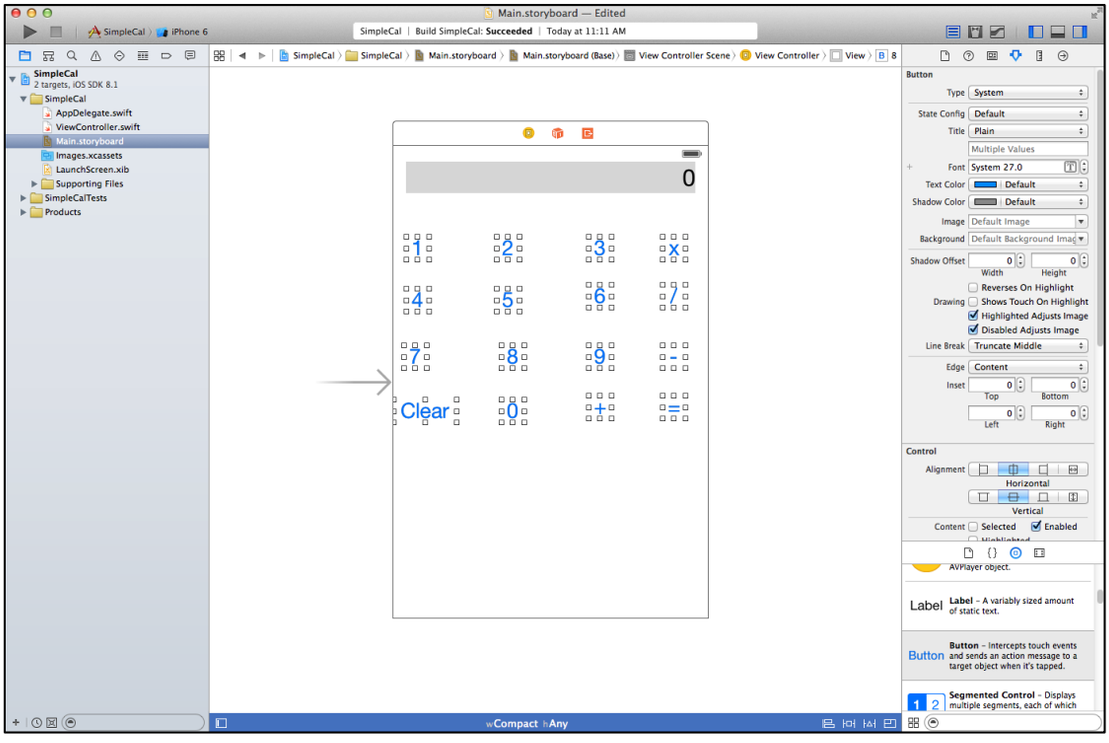


图 10: 选择按钮并调整尺寸

10. 点 **Run** 按钮运行程序，这会显示计算器用户界面，如图 11 所示：

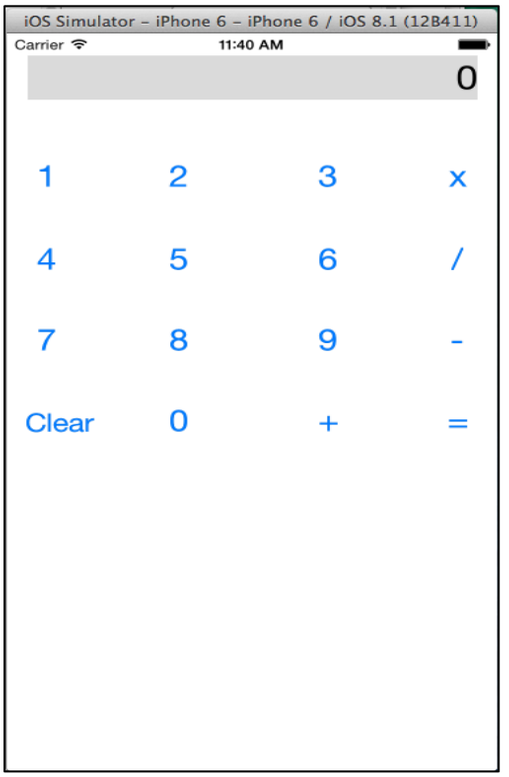


图 11: 计算器用户界面

11. 下面需要添加每个运算符按钮背后的逻辑，确保用户界面能够作为计算器发挥作用。点辅助编辑器来分屏显示，如图 12 所示：

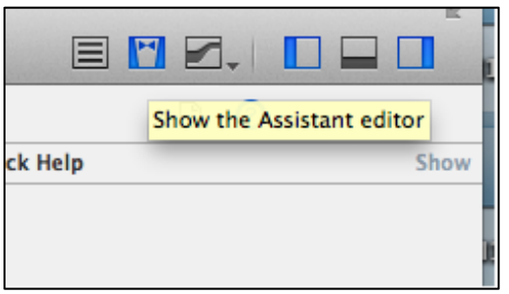


图 12: Click the Assistant Editor

12. 右键按住，拖拽计算器界面的标签到 `ViewController.swift` 区域。在出现的弹窗中（参见图 13），进行下面这些操作：
    1. **Connection** 字段选择 **Outlet**
    2. **Name** 字段输入 **Output Screen**
    3. **Type** 字段选择 **UILabel**
    4. **Event** 中选择 **Touch Up Inside**
    5. **Argument** 选择为 **Weak**
    6. 点 **Connect**

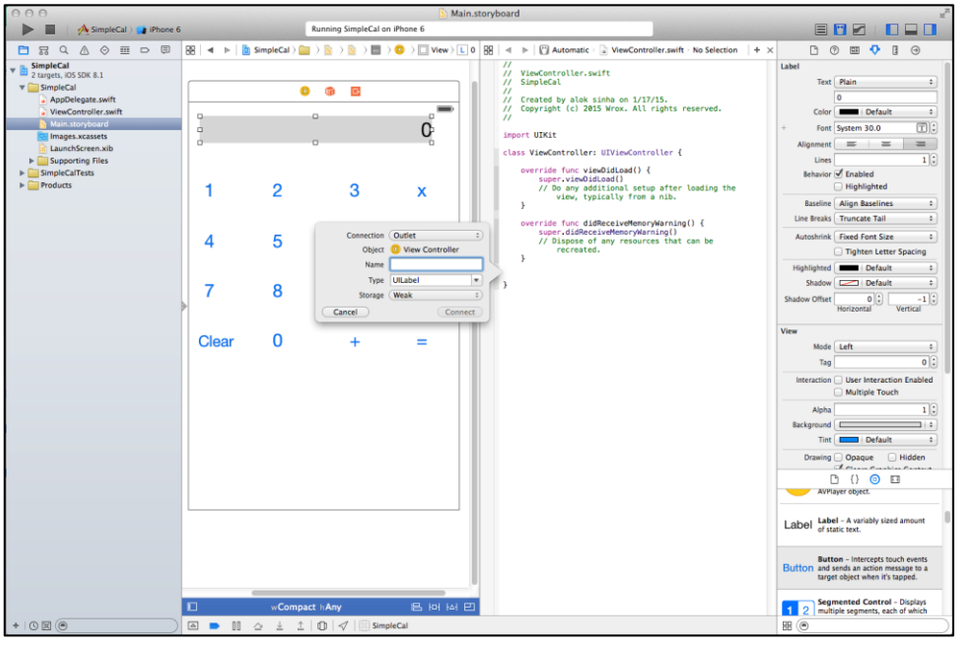


图 13: 右键按住并拖拽标签到 ViewController

13. 下一步，右键按住，拖拽每一个计算机按钮到 ViewController.swift 区域。在出现的弹窗中（参见图 14），进行下面这些操作：
    1. **Connection** 字段选择 **Action**
    2. **Name** 字段输入 **calButton**
    3. 从 **Type** 下拉列表选择 **AnyObject**
    4. **Event** 字段中选择 **Touch Up Inside**
    5. **Argument** 字段中选择 **Sender**
    6. 点 **Connect**

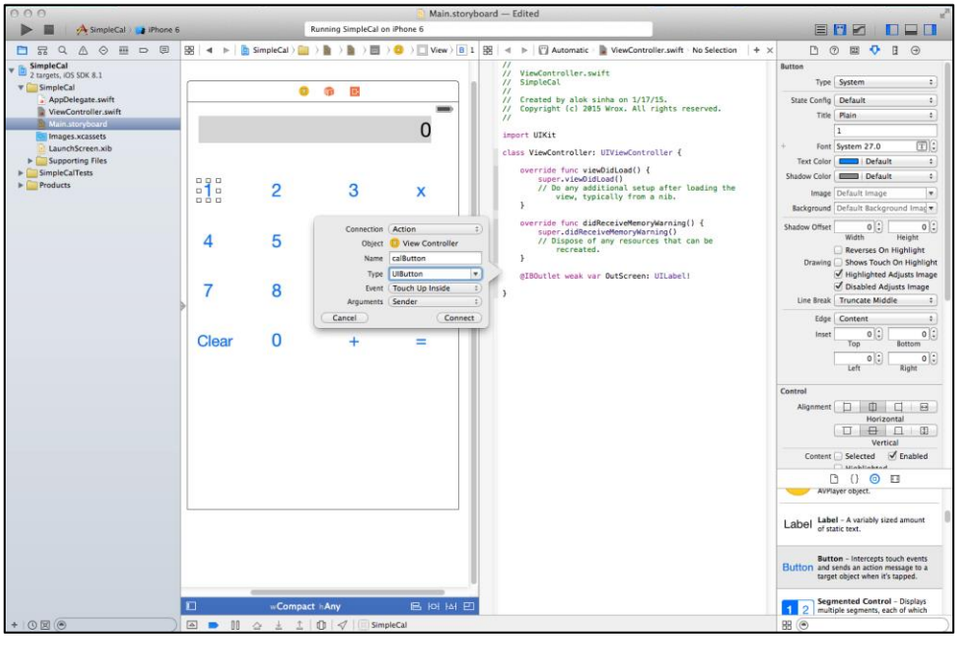


图 14: 右键按住并拖拽每一个计算器按钮到 ViewController

14. 下一步，右键按住，拖拽每一个运算按钮到 ViewController.swift 区域。在出现的弹窗中（参见图 15），进行下面这些操作：
    1. 从 **Connection** 下拉列表选择 **Action**
    2. **Name** 字段输入 **operator**
    3. 从 **Type** 下拉列表选择 **AnyObject**
    4. **Event** 字段中选择 **Touch Up Inside**
    5. **Argument** 字段中选择 **Sender**
    6. 点 **Connect**

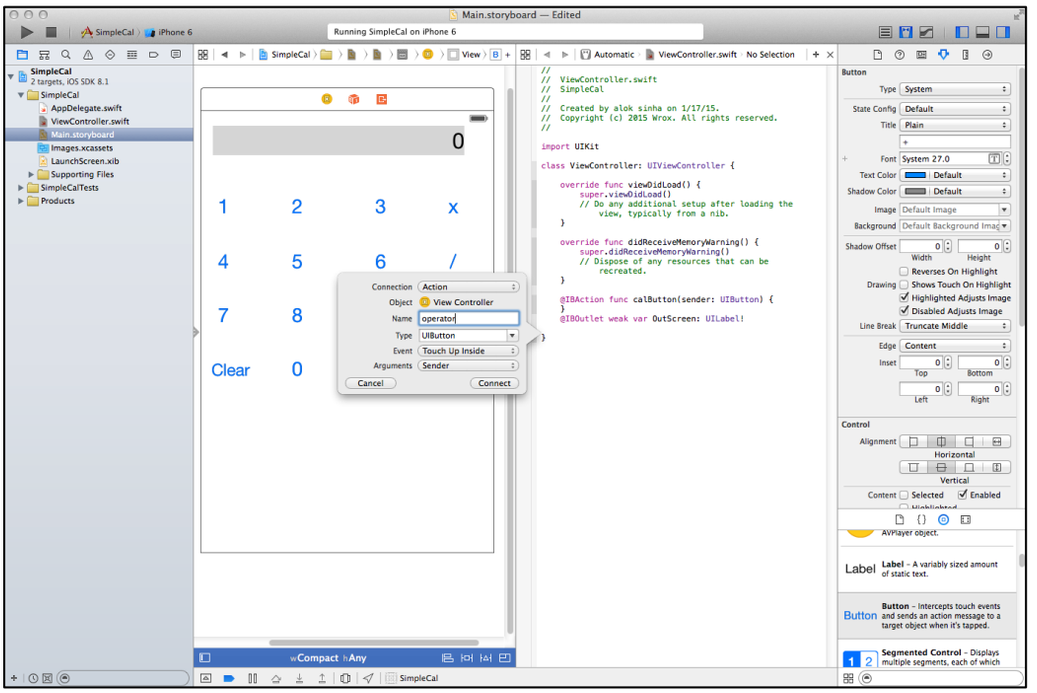


图 15: 右键按住并拖拽每一个运算按钮到 ViewController

15. 类似地，你还可以为等号按钮指定动作，如图 16 所示：

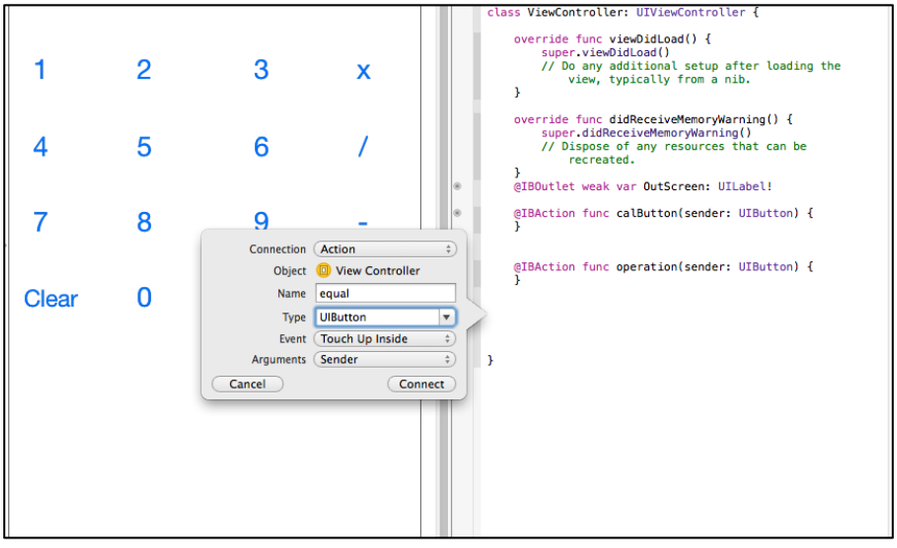


图 16: 右键按住并拖拽等号按钮到 ViewController.swift

16. 然后通过导航面板，选择 ViewController.swift 并添加如下代码：

```
// code for operation calButton:
@IBActionfunccalButton(sender:AnyObject){
    var number = sender.currentTitle
    ifisTyping{
    OutScreen.text = OutScreen.text!+number!!
}else{
    OutScreen.text = number
    isTyping = true
}
}

// code for operation Action button:
@IBActionfuncoperation(sender:Anyobject){
    isTyping= false
    firstnu= OutScreen.text!.toInt()!
    operation = sender.currentTitle!!
}

// code for equal button:
@IBActionfuncequal(sender: AnyObject) {
secondnum = OutScreen.text!.toInt()!
isTyping= false


if (operation == "+")
{  result = firstnum + secondnum
OutScreen.text = "\(result)"
println(result)}
```

```
else if (operation == "-" )
        {   result = firstnum - secondnum
OutScreen.text = "\(result)"
println(result) }

else if (operation == "x")
{  result = firstnum * secondnum
OutScreen.text = "\(result)"
println(result)}

else if (operation == "/")
        {
result = firstnum / secondnum
OutScreen.text = "\(result)"
println(result)
        }
}
}

// code for clearing the calculator OutputScreen:
@IBActionfuncClear(sender: AnyObject) {
firstnum = 0
secondnum = 0
        operation = ""
isTyping = false
result = 0
OutScreen.text = "\(result)"
    }
```

17. 最后，通过点击 **Run** 按钮来运行模拟器，图 17 显示了输出结果：

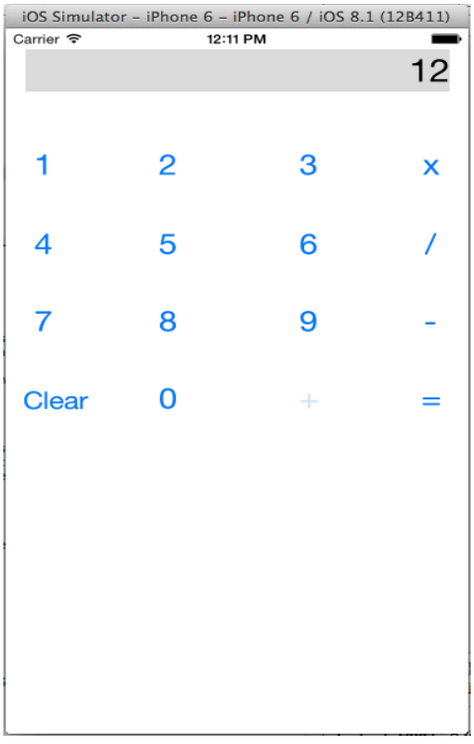


图 17: 代码输出结果

源码

```
Code in ViewController.swift

importUIKit

classViewController: UIViewController {

varisTyping = false
varfirstnum = 0
varsecondnum = 0
var result = 0
var operation = ""

    @IBOutlet weak varOutScreen: UILabel!

overridefuncviewDidLoad() {
super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

    }
overridefuncdidReceiveMemoryWarning() {
super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
```

```
    @IBActionfunccalButton(sender: AnyObject) {
var number = sender.currentTitle

ifisTyping {
OutScreen.text = OutScreen.text! + number!!

        } else {
OutScreen.text = number;
isTyping = true
        }
}
    @IBActionfuncClear(sender: AnyObject) {
firstnum = 0
secondnum = 0
        operation = ""
isTyping = false
result = 0
OutScreen.text = "\(result)"
    }

    @IBActionfuncoperation(sender: AnyObject) {
isTyping = false
firstnum = OutScreen.text!.toInt()!
operation = sender.currentTitle!!
println(operation)
    }

    @IBActionfuncequal(sender: AnyObject) {
secondnum = OutScreen.text!.toInt()!
isTyping = false


if (operation == "+")
{  result = firstnum + secondnum
OutScreen.text = "\(result)"
println(result)}

else if (operation == "-" )
        {   result = firstnum - secondnum
OutScreen.text = "\(result)"
println(result) }

else if (operation == "x")
{  result = firstnum * secondnum
OutScreen.text = "\(result)"
println(result)}
```

```
else if (operation == "/")
        {
result = firstnum / secondnum
OutScreen.text = "\(result)"
println(result)
        }
    }
}
```