基于 iOS 平台的猜拳游戏设计制作

课程名称：iOS 程序设计

作 者 ： 艾 孜 尔 江 · 艾 尔 斯 兰

学 号 ： 1 7 0 8 1 1 6 0

时 间 ： 二 〇 一 九 年 九 月 二 十 七 日

地 点 ： 软 件 学 院 6 0 4 实 验 室

目录

实验目的

本实验是基于 iOS 平台的猜拳游戏设计及制作实验，属于典型的设计型实验类别，且属于创新型和综合型实验的结合体。故本实验的目的可以归纳为以下几点：

（1） 充分掌握苹果电脑的使用方法；

（2） 充分理解软件项目制作流程及实际开发流程；

（3） 充分理解 iOS 程序基于 Swift 语言设计制作的基础方式；

（4） 设计出用户友好的猜拳游戏机制；

（5） 制作出所所设计的猜拳游戏并能够按预期需求运行；

（6） 所设计出的程序适配与各类 iOS 设备。

实验内容

本次实验中，本人首先依据本课程的教材 ***APP-DEVELOPMENT-WITH-SWIFT*** 一书中第二十章内容进行了对本项目的初步创建，制作猜拳游戏的程序。随着该章节内容的结束，我也发现在书中的内容仅仅对于逻辑框架做了一个简单明了的叙述，而并没有给出最终成品的实现效果，在这样的情况下，就能够对于最终成品有一个发散性的定义。好在书中提及了相当的暗示。比如使用 isHidden()函数隐藏控件、使用 isEnable()函数设置 Button 状态等，本人就根据书中给出的逻辑框架，通过简单的 Label 控件、Button 控件和课程中所学习的堆栈视图（Stack View）完成了本次实验。

在设计之初，根据书中给出的猜拳游戏的机制，本人感到过多的生疏和难以实现，于是就将本次实验的内容拆分成非常零散的步骤：

（1） 通过 Label 控件、Button 控件实现游戏的布局【我在这里使用的石头、剪刀和布的呈现方式实际上都是系统自带的表情符号内容】；

（2） 通过和 Stack View 实现上述设计的布局在各类 iOS 移动设备上都能合理呈现；

（3） 初始状态下 Play Again 按钮呈隐藏状态；

（4） 通过 isHidden()函数实现在 Button 被点击之后隐藏其它两个 Button，并能够唤起先前隐藏的 Play Again 按钮；

（5） 在 Button 被点击之后上方有机器人图标（表情符号）的 Label 可以改变成为一个固定的图标表情符号；

（6） 在下方 Play Again 按钮被点击之后所有 Button 都能够从隐藏状态中被唤起；

（7） 测试以便上述步骤能够合理运行并且没有错误【实际上在这一步本人发现先前所设置的隐藏属性在最左边的 Button 中并没有奏效，出了一个临时的 Bug，为能够进展后续部分，本人在设计时确保代码方面准确无误之后便暂时忽略了这一视图上的问题。】

（8） 通过 GKRandomDistribution()函数实现上方 Label 控件在 Button 被按下之后随机呈现一个图标（表情符号）；

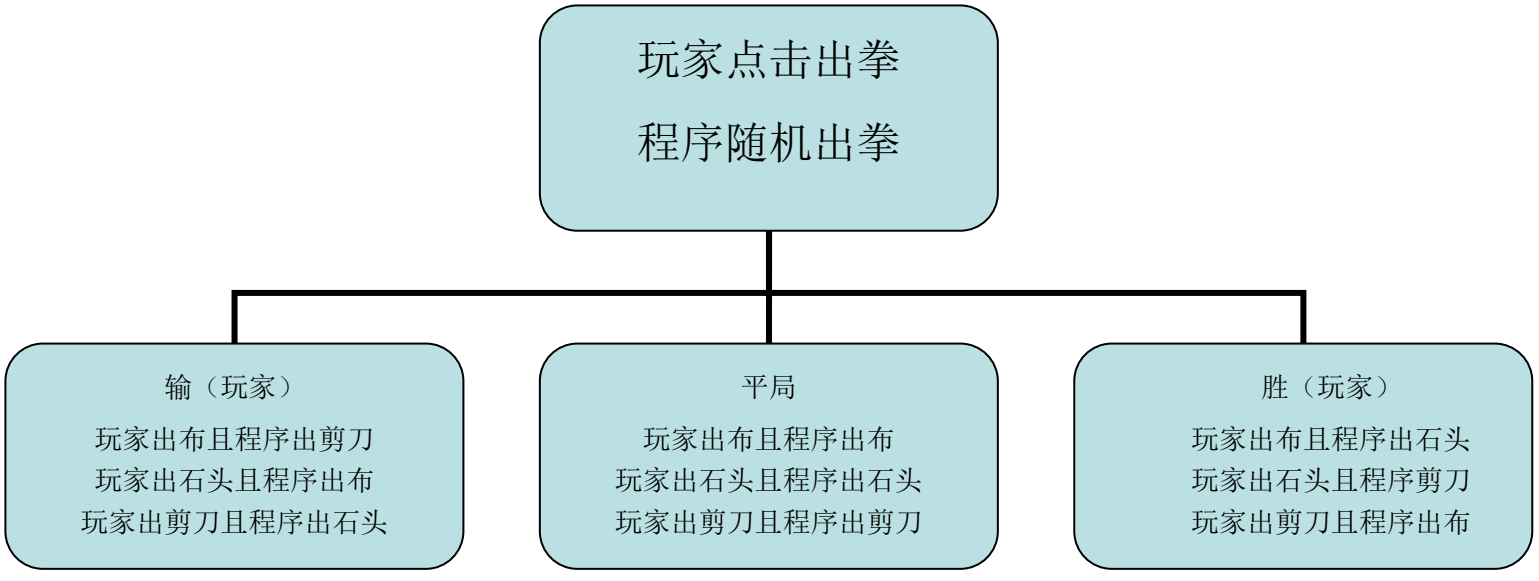
（9） 加入游戏规则——当玩家出的拳和随机呈现的一致时为平局，不同时有输赢提示在大图标下方 Label 控件中提示；

（10） 通过 view.backgroundColor 的设定来实现各类结局时背景颜色的变化；

（11） 通过 view.backgroundColor 的设定来实现点击 Play Again 后背景颜色变回白色。

（12） 测试整个程序的运行情况并撰写实验报告。

在这样的步骤拆分之后说是有的事务变得相对清晰明了。尽管教员在课程中多次提及本实验中的关键的、比较难的步骤是猜拳输赢逻辑的框架，但是就本实验中的猜拳游戏来说，本人认为最为关键的部分还是在脑海中理清开发步骤并使之有条理地进展，毕竟本次游戏的输赢机制相对比较容易并且每个人都对于猜拳游戏的逻辑规则耳熟能详。但为响应本次实验作业的提交要求，本人还是保守地将猜拳游戏的输赢逻辑框架罗列如下：



主要的实现代码如下：

```
//  Sign.swift
//  Guessor
//  Created by apple27 on 2019/9/26.
//  Copyright © 2019 年 bjutsoft. All rights reserved.

import Foundation
import GameplayKit

//设置随机生成 0 到 2 的数字
let randomChoice = GKRandomDistribution(lowestValue: 0, highestValue: 2)

func randomSign() -> String {    //实现随机数字所对应的出拳表情符号
    let sign = randomChoice.nextInt()
    if sign == 0 {
        return "✊"
    }
    else if sign == 1 {
        return "✋□"
    }
    else {
        return "✌□"
    }
}
```

```
//  ViewController.swift
//  Guessor
//  Created by apple27 on 2019/9/26.
//  Copyright © 2019 年 bjutsoft. All rights reserved.
import UIKit
class ViewController: UIViewController {

    @IBOutlet weak var Robot: UILabel!
    @IBOutlet weak var Hint: UILabel!
    @IBOutlet weak var Play: UIButton!

    @IBOutlet weak var PlayAgain: UIButton!
    @IBOutlet weak var Scissora: UIButton!
    @IBOutlet weak var Mux: UIButton!
    @IBOutlet weak var Paoer: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        PlayAgain.isHidden = true
        //Mux.frame = CGRect(x: 0, y: 0, width: 75, height: 176)   手动设置控件的坐标位
置
    }

    func Init(){
        Robot.text = "□"
        Scissora.isHidden = false
        Mux.isHidden = false
```

```
        Paoer.isHidden = false
        PlayAgain.isHidden = true
        view.backgroundColor = .white
        Hint.text = "Scissors, rock and paper. "

        //Enable all the choice in the game starting view
        Mux.isEnabled = true
        Scissora.isEnabled = true
        Paoer.isEnabled = true

    }


    @IBAction func RockClicked(_ sender: Any) {
        var gameSign = randomSign()

        Scissora.isHidden = true
        Paoer.isHidden = true
        PlayAgain.isHidden = false
        Robot.text = gameSign

        //The Rock can't be clicked in the game running view
        Mux.isEnabled = false    //设置出拳之后结局状态下玩家之前所选按钮锁定
```

//【我个人认为这其实是多此一举，因为玩家想要出同样的拳直接再次点击结局状态下之前出过的拳就可以了，没有必要点击 Play Again 再回到初始界面再在初始界面下选择同样的拳。当用户想出不一样的拳的时候他才点击 Play Again 返回初始界面选择另外的拳即可。所以 Play Again 并不是朱教员课上所述那样“形同虚设”。】

```
        if gameSign == "✊□" {
```

```
            Hint.text = "You lose!"
            view.backgroundColor = .red
        }
        else if gameSign == "✌□"{
            Hint.text = "You win!"
            view.backgroundColor = .green
        }
        else{
            Hint.text = "In equal!"
            view.backgroundColor = .yellow
        }
        print("Mux")


        print(Mux.frame)
    }

    @IBAction func ScissorsClicked(_ sender: Any) {
        var gameSign = randomSign()

        print(Scissora.isHidden)
        Mux.isHidden = true
        Paoer.isHidden = true
        PlayAgain.isHidden = false
        Robot.text = gameSign

        if gameSign == "✊" {
            Hint.text = "You lose!"
            view.backgroundColor = .red
```

```
        }
        else if gameSign == "✋□"{
            Hint.text = "You win!"
            view.backgroundColor = .green
        }
        else{
            Hint.text = "In equal!"
            view.backgroundColor = .yellow
        }

        //The Scissora can't be clicked in the game running view
        Scissora.isEnabled = false
        //print("Scissora")
        //print(Scissora.frame)   测试时打桩查看“剪刀”控件的位置
}

@IBAction func PaperClicked(_ sender: Any) {
        var gameSign = randomSign()
        Mux.isHidden = true
        Scissora.isHidden = true
        PlayAgain.isHidden = false
        Robot.text = gameSign

        if gameSign == "✌□" {
            Hint.text = "You lose!"
            view.backgroundColor = .red
        }
        else if gameSign == "✊"{
            Hint.text = "You win!"
```

```
            view.backgroundColor = .green
        }
        else{
            Hint.text = "In equal!"
            view.backgroundColor = .yellow
        }

        //The Paper can't be clicked in the game running view
        Paoer.isEnabled = false
        print(Paoer.frame)
    }


    @IBAction func Play(_ sender: Any) {
         Init()
    }

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

}
```

实验环境和器材

实验中所使用的软件环境为苹果系统下的 XCode，基于 iOS 10.0 开发。

实验步骤(项目的设计与实现)

本实验的主要步骤如下：

（1） 设计构思最终效果；

（2） 拆分各类功能，撰写随机生成数据的函数；

（3） 设计视图布局；

（4） 实现布局在所有 iOS 移动设备上的适配问题；

（5） 链接布局中的各类控件并撰写 Button 的在被点击时会发生的代码；

（6） 测试程序是否满足基本要求；

（7） 整合代码使先前所撰的随机数据生成的那个类融入于主程序部分；

（8） 测试程序是否满足最终需求；

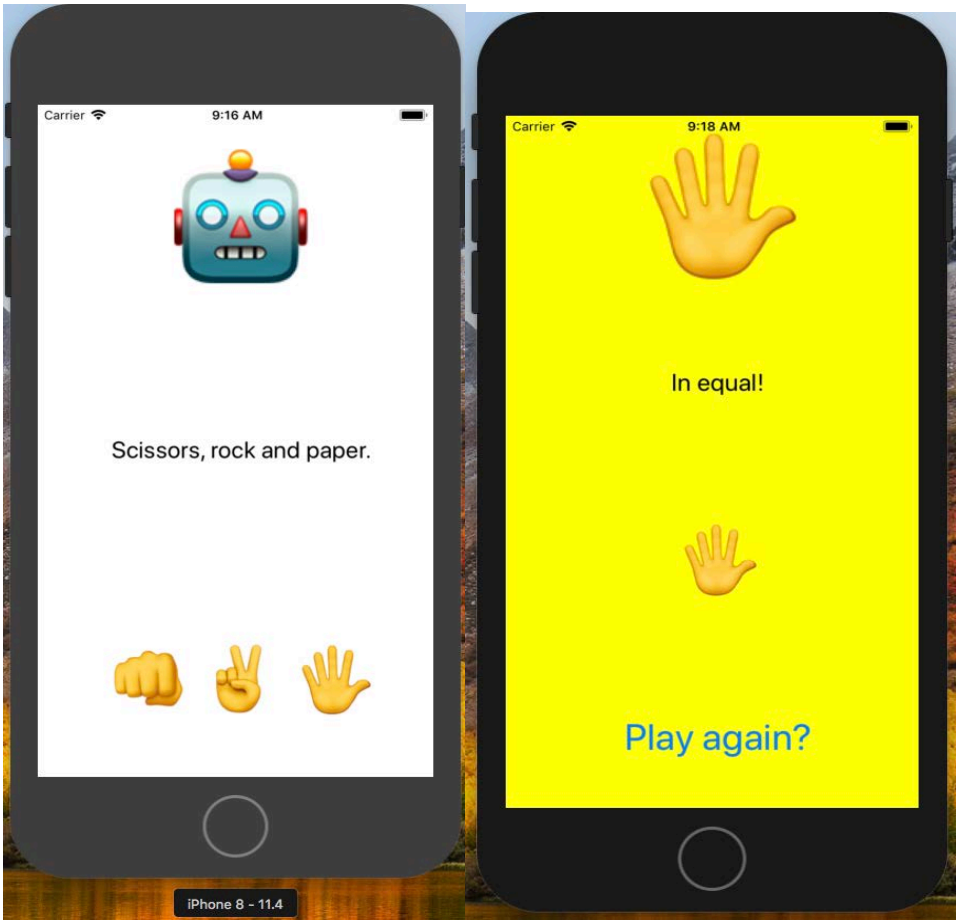
（9） 进一步完善程序，添加一些锦上添花的效果（比如颜色改变）；

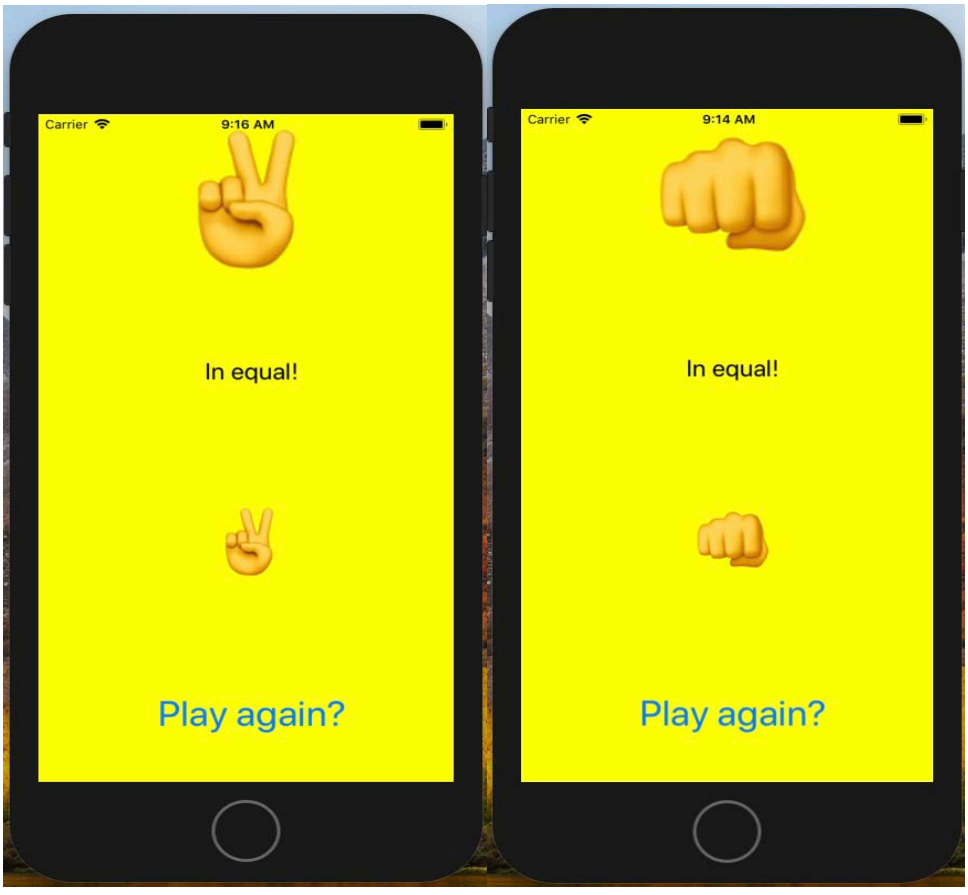
（10） 测试程序是否可以发布，如发现错误，回溯上述完善和调试的过程；

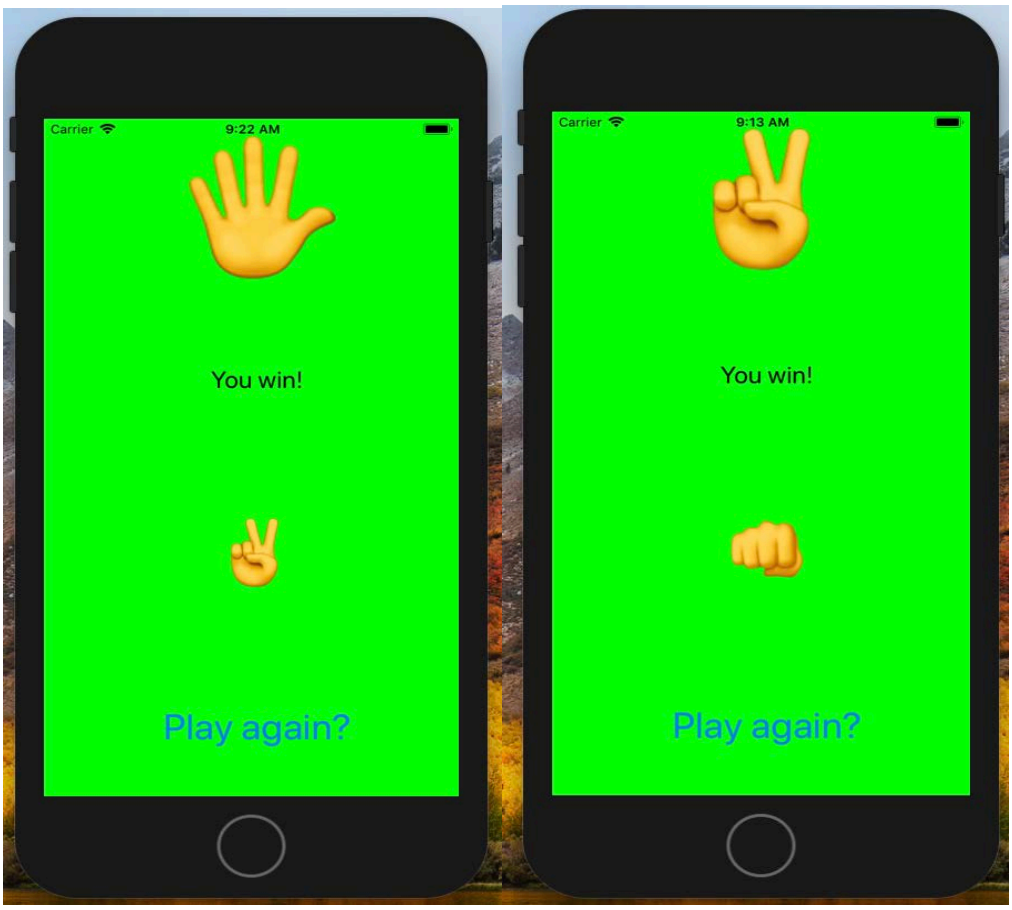
（11） 撰写实验报告。

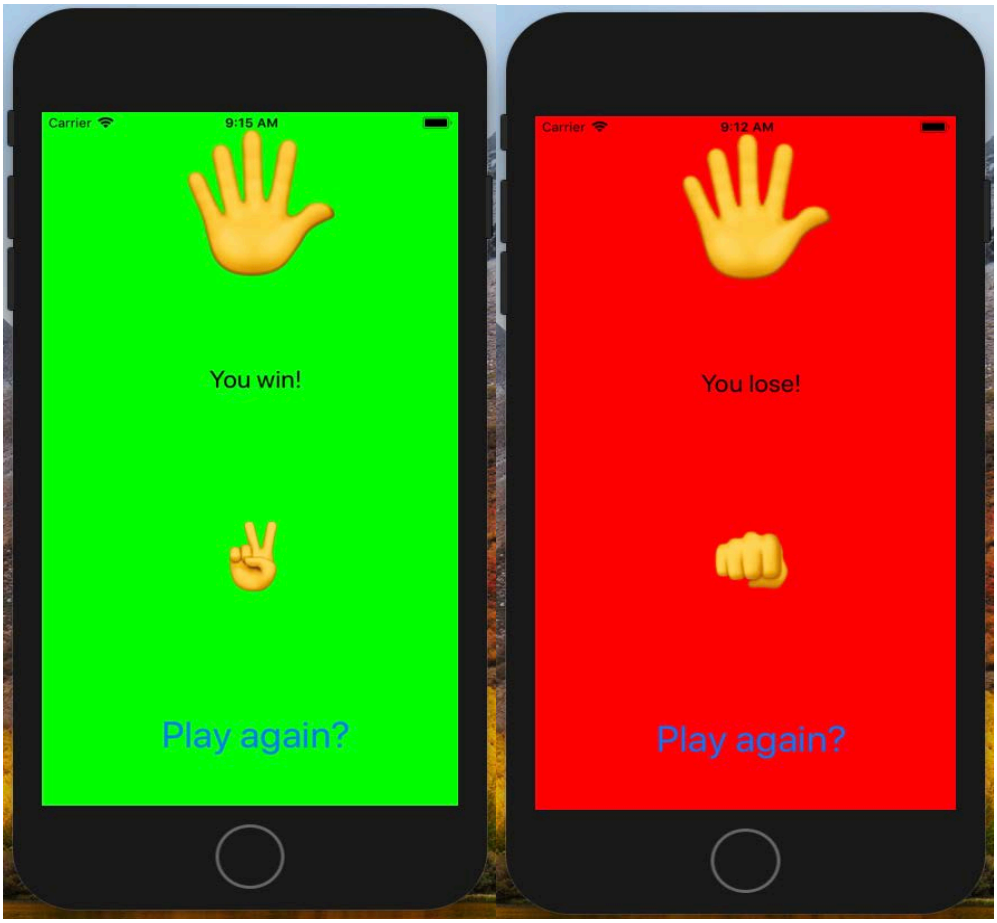
实验结果 (项目测试)

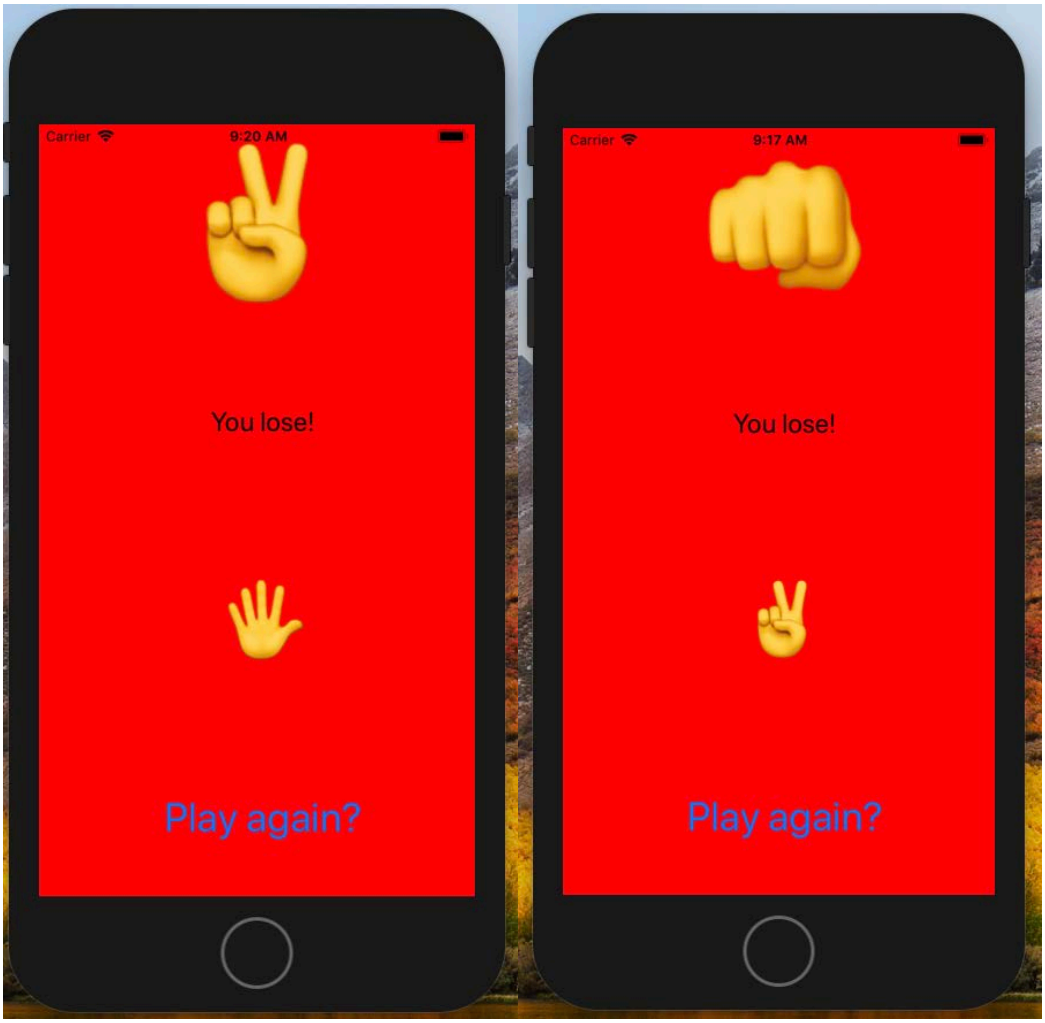
在测试过程中本人发现使用了 Stack View 之后最左边的一个按钮在本应当能够被呈现出来的时候却销声匿迹，经过调试也没有发现该控件消失的丝毫线索。最终在通过 frame()函数查看其坐标后发现，其在视图中的坐标在被点击之后其 frame()函数的第三个选项——宽度变成了 0。通过强制设置，还是没有奏效。最终在第二天清晨把原先的 Stack View 删去重新做了一遍，该问题得到解决。估计之前是在 Stack View 的使用过程中出现了错点一些选项的事故，或许这可能是 XCode 本身出现的一个漏洞。但总而言之，重新设置一遍后本程序就圆满完成了。下方是程序运行时的截图：


Carrier
9:16 AM
Scissors, rock and paper.
iPhone 8 - 11.4
Carrier
9:18 AM
In equal!
Play again?

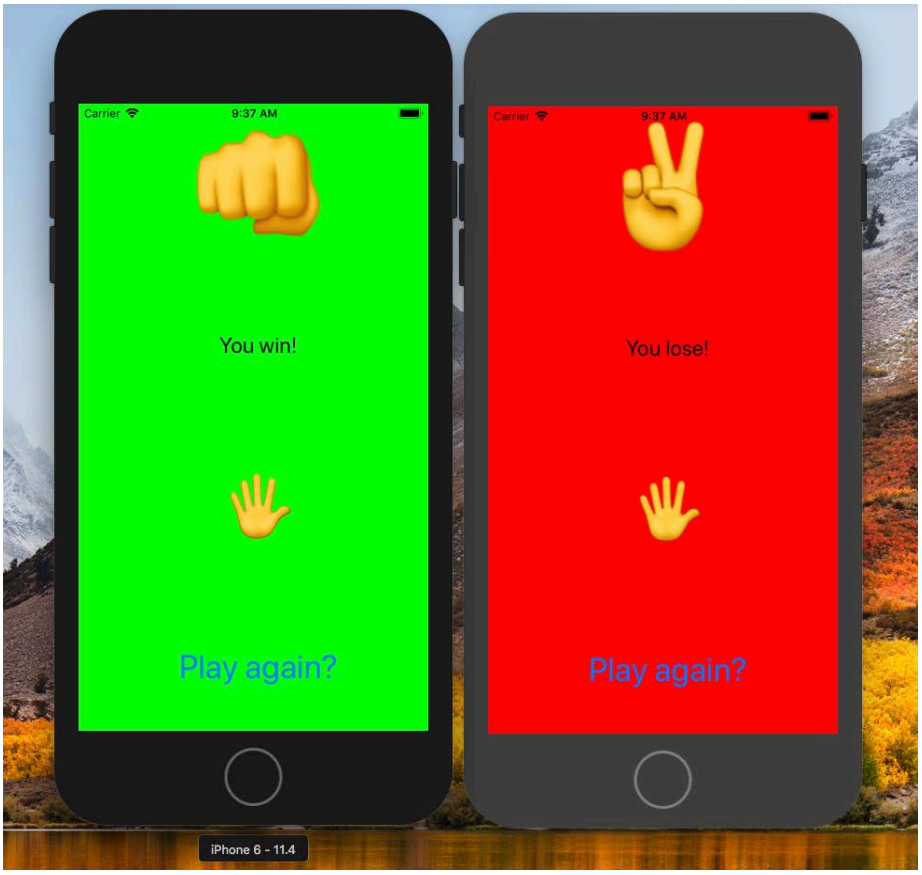

Carrier
9:16 AM
In equal!
Play again?
Carrier
9:14 AM
In equal!
Play again?

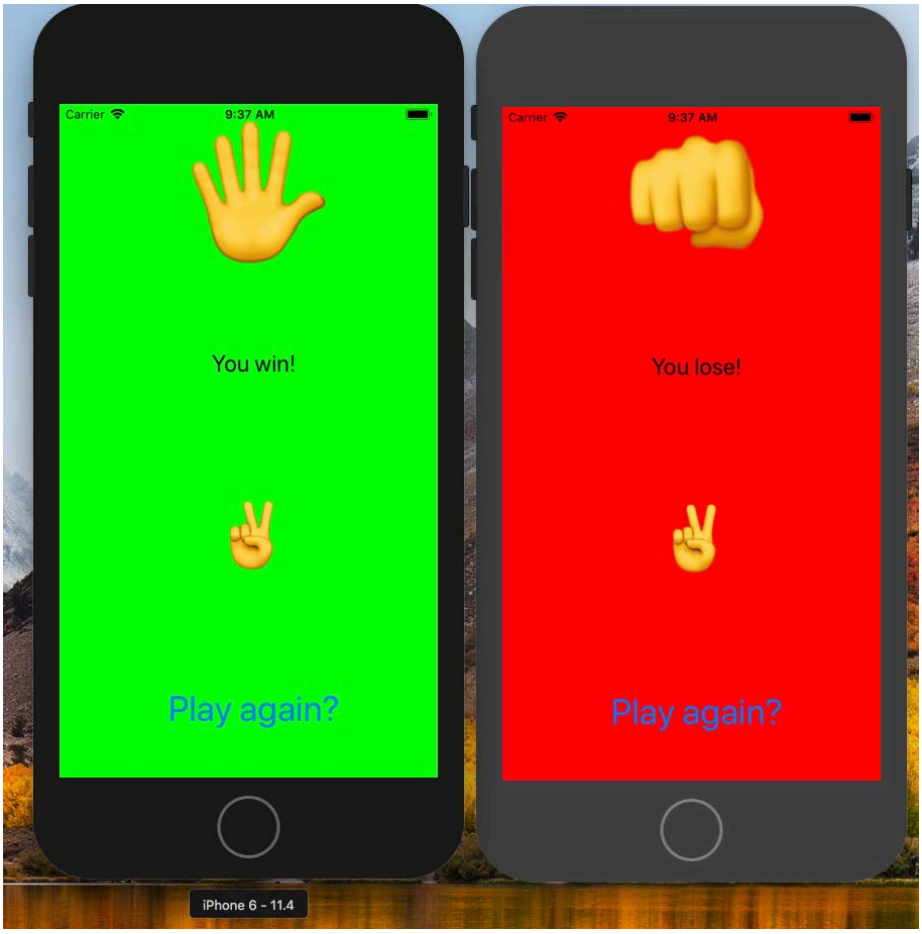

Carrier
9:22 AM
You win!
Play again?
Carrier
9:13 AM
You win!
Play again?

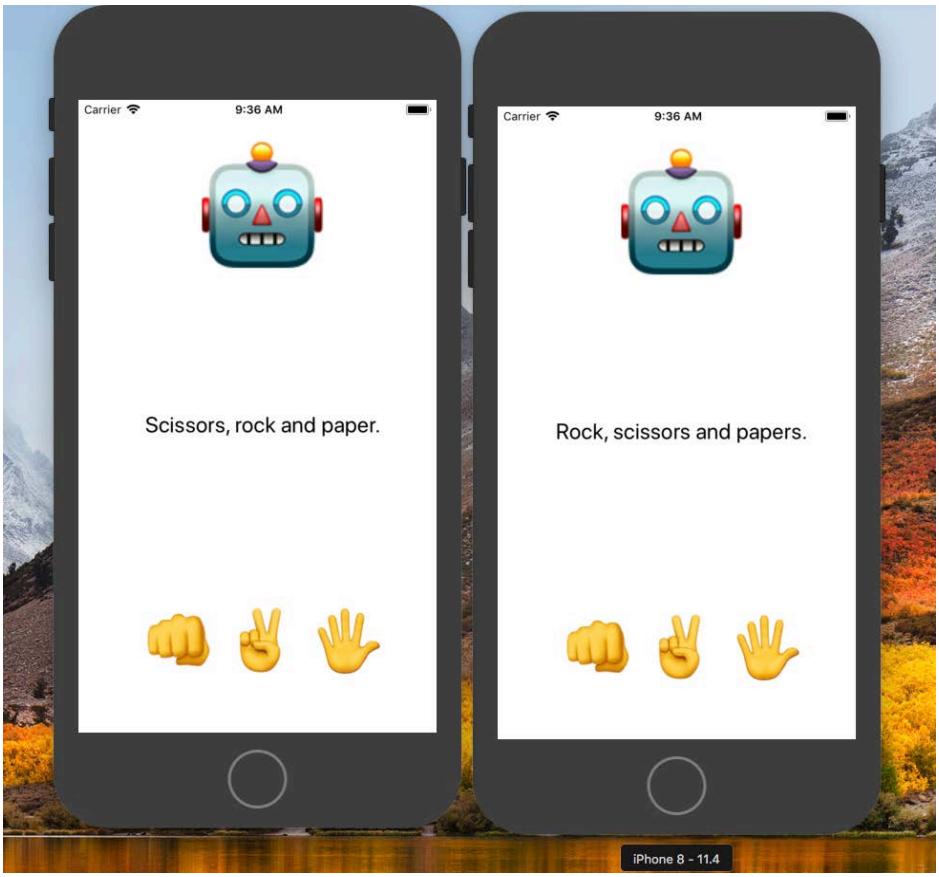

Carrier
9:15 AM
You win!
Play again?
Carrier
9:12 AM
You lose!
Play again?

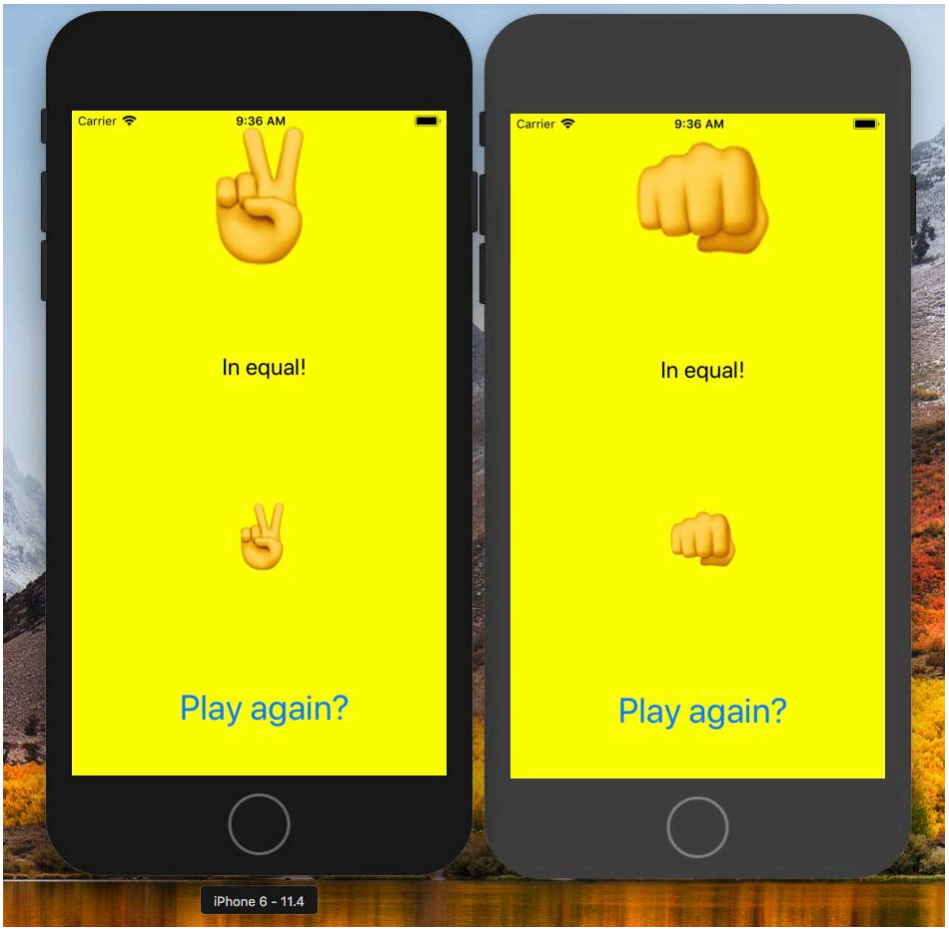

Carrier
9:20 AM
You lose!
Play again?
Carrier
9:17 AM
You lose!
Play again?

不同机子上的适配情况；


Carrier
9:37 AM
You win!
Play again?
Carrier
9:37 AM
You lose!
Play again?
iPhone 6 - 11.4


Carrier
9:37 AM
You win!
Play again?
Carrier
9:37 AM
You lose!
Play again?
iPhone 6 - 11.4


Carrier
9:36 AM
Scissors, rock and paper.
Carrier
9:36 AM
Rock, scissors and papers.
iPhone 8 - 11.4


Carrier
9:36 AM
In equal!
Play again?
Carrier
9:36 AM
In equal!
Play again?
iPhone 6 - 11.4

讨论

在本次实验中，我习得了程序的开发过程，并且最终达到了本实验报告中“实验目的”所叙述的所有内容，唯在“充分掌握苹果电脑使用技巧”方面还有待进一步熟悉和深入，希望在之后的练习中能够在各个方面应该会更上一层楼。

※结论

结论不是具体实验结果的再次罗列，也不是对今后研究的展望，而是针对这一实验所能验证的概念、原则或理论的简明总结，是从实验结果中归纳出的一般性、概括性的判断，要简练、准确、严谨、客观。

※鸣谢

本次实验中，主要感谢朱培毅老师对于本人提及的拙劣问题不厌其烦地解答与其本程序所出现的漏洞进行的调试，通过他对本软件在初始阶段出现漏洞的调试，本人对于打桩测试方法进行了进一步的重温和巩固，并了解到通过 frame() 函数查看或设置控件在视图中所处位置的技巧。同时也对其诲人不倦之精神受到了进一步地濡染。还感谢身边的同学在程序最终效果上的思路引导。

※参考资料

[1] 课程教材 ***APP-DEVELOPMENT-WITH-SWIFT***。