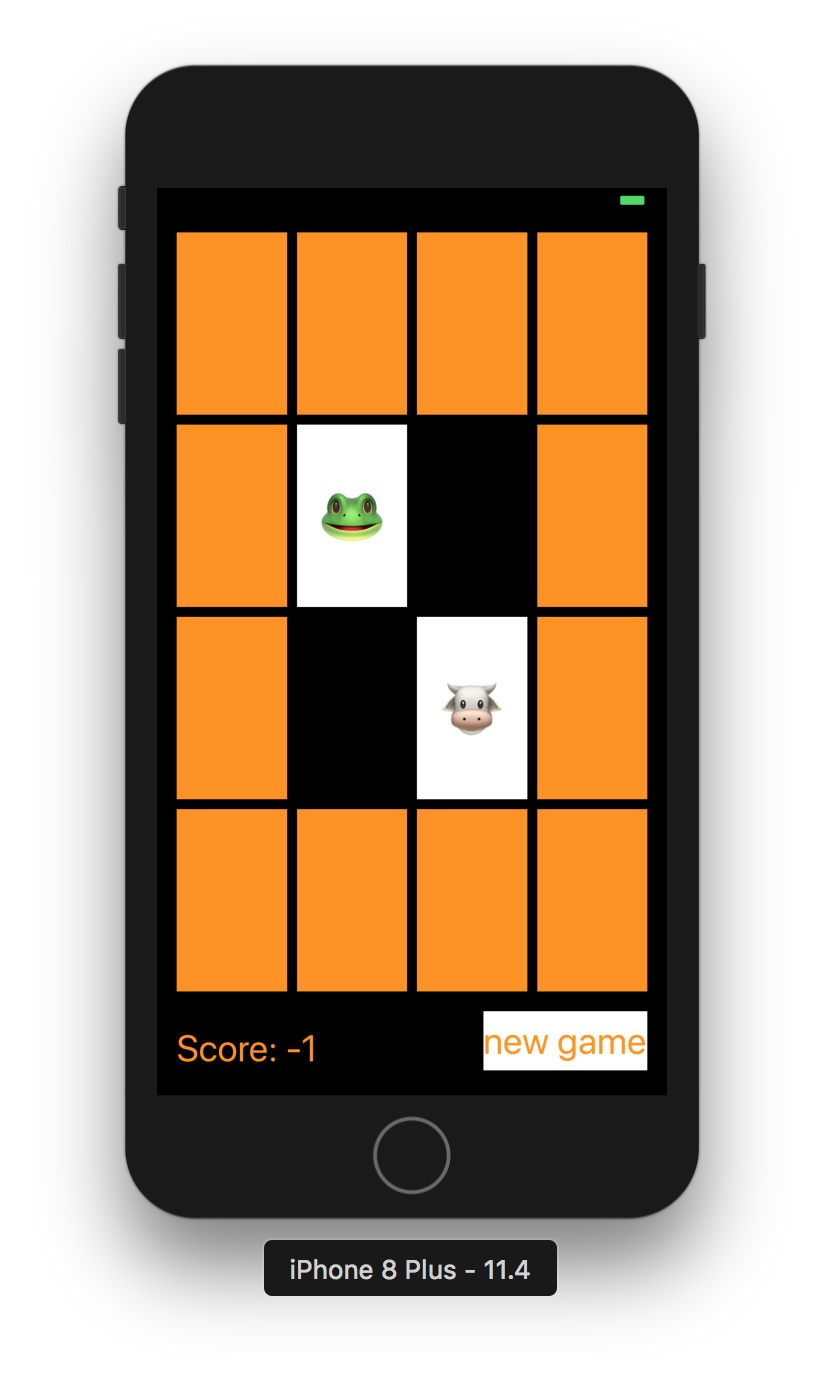
实验编号： 7 **四川师大《IOS》实验报告 2018** 年 **10** 月 **24** 日

### **计算机科学学院** 2016 级 4 班 实验名称： Game单MVC \_

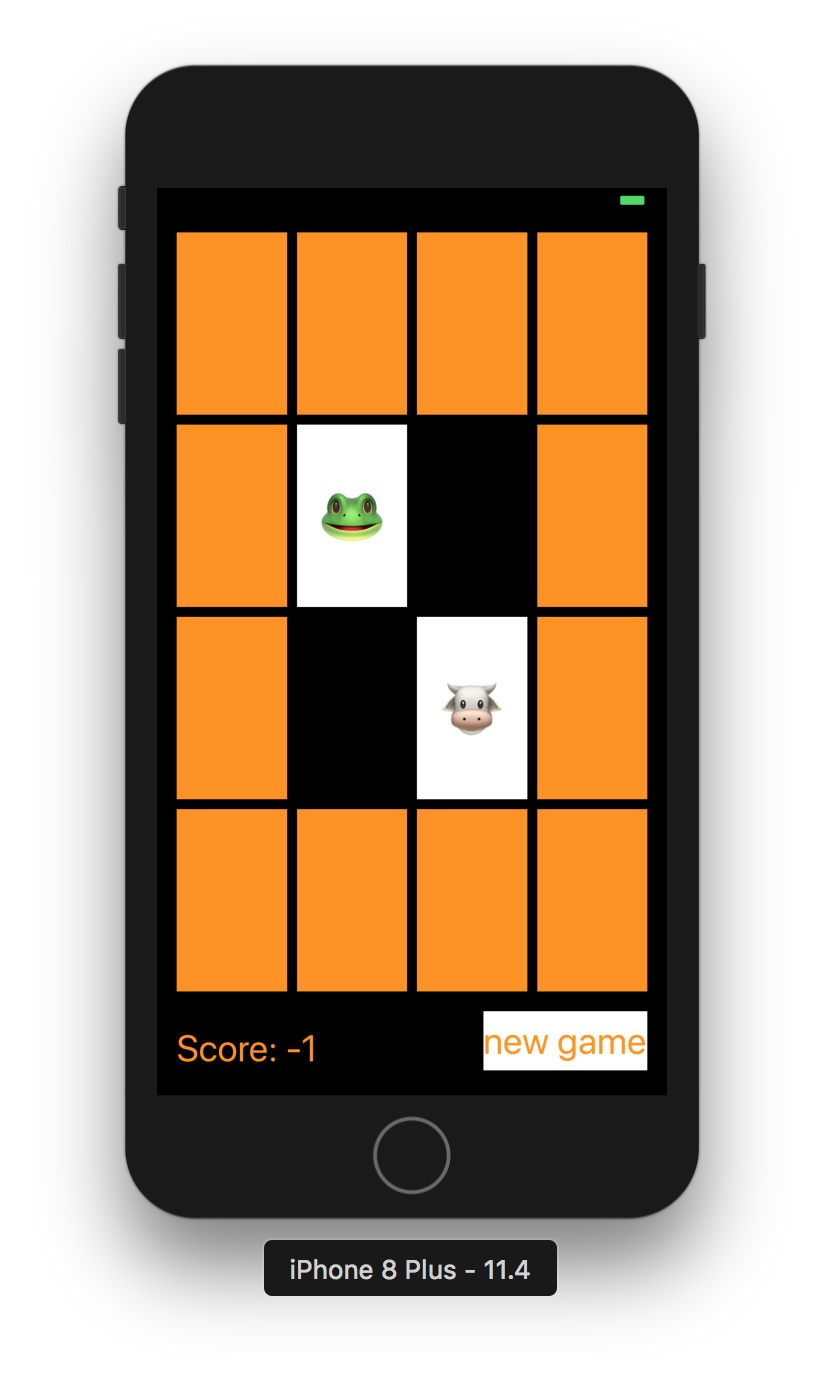
姓名： 郭周倩 学号： 2016110413 指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**实验 七 \_\_\_\_\_\_** Game单MVC **\_\_\_\_\_\_\_\_**

1. 实验目的及要求
2. 实现一款功能完整的game（Concentration）；
3. 掌握单MVC的主要思想；
4. 认真填写实验报告，要求附加部分运行界面和主要代码；
5. 对设计好的程序，检查输出是否符合预期，如有错请分析错误原因并解决；
6. 实验内容
7. 参照Stanford视频1和2完成一个game（Concentration）的制作；
8. 在(1)的基础上进一步完成Stanford Assignment 1的完整要求；
9. 采用autolayout布局解决横竖屏自适应如下所示；



1. 实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）
2. 参照Stanford视频1和2完成一个game（Concentration）的制作；
3. 在(1)的基础上进一步完成Stanford Assignment 1的完整要求；
4. 采用autolayout布局解决横竖屏自适应如下所示；



* 程序代码：

Card:

import Foundation

struct Card {

    var isFaceUp = false{

        didSet {

            if oldValue && !isFaceUp {

                isSeen = true

            }

        }

    }

    var isMatched = false

    var identifier: Int

    private(set) var isSeen = false

    static var identifierFactory = 0

    static func getUniqueIdentifier() -> Int{

        identifierFactory += 1

        return identifierFactory

    }

    init() {

        self.identifier = Card.getUniqueIdentifier()

    }

}

Concentration:

import Foundation

class Concentration{

    var cards = [Card]()

    var score = 0

    var indexOfOneAndOnlyFaceUpCard: Int?

    func chooseCard(at index:Int){

        if !cards[index].isMatched{

            if let machIndex = indexOfOneAndOnlyFaceUpCard, machIndex != index{

                if cards[machIndex].identifier == cards[index].identifier{

                    cards[machIndex].isMatched = true

                    cards[index].isMatched = true

                    score += 2

                }else{

                    if cards[machIndex].isSeen {

                        score -= 1

                    }

                    if cards[index].isSeen {

                        score -= 1

                    }

                }

                cards[index].isFaceUp = true

                indexOfOneAndOnlyFaceUpCard = nil

            }else{

                for flipDownIndex in cards.indices{

                    cards[flipDownIndex].isFaceUp = false

                }

                cards[index].isFaceUp = true

                indexOfOneAndOnlyFaceUpCard = index

            }

        }

    }

    init(numberOfPairsOfCards: Int) {

        for \_ in 1...numberOfPairsOfCards{

            let card = Card()

            cards += [card,card]

        }

        cards.sort { \_,\_ in arc4random\_uniform(2) > 0 }

    }

}

import UIKit

class ViewController: UIViewController {

    lazy var game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1 ) / 2)

    @IBOutlet var cardButtons: [UIButton]!

    @IBOutlet weak var flipCountLabel: UILabel!

    @IBAction func touchCard(\_ sender: UIButton) {

        if let cardNumber = cardButtons.index(of: sender) {

            game.chooseCard(at: cardNumber)

            updateViewFromModel()

        }else{

            print("choosen cards was not in cardButtons")

        }

    }

    func updateViewFromModel() {

        for index in cardButtons.indices {

            let button = cardButtons[index]

            let card = game.cards[index]

            if card.isFaceUp {

                button.setTitle(emoji(for: card), for: UIControlState.normal)

                button.backgroundColor =  colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)

            }else{

                button.setTitle("", for: UIControlState.normal)

                button.backgroundColor = card.isMatched ?  colorLiteral(red: 1, green: 1, blue: 1, alpha: 0) :  colorLiteral(red: 1, green: 0.5763723254, blue: 0, alpha: 1)

            }

        }

        flipCountLabel.text = "Score: \(game.score)"

    }

    var themes = [0:[",",",",",",",",",","],

                  1:[",",",",",",",",",","],

                  2:["⚽️",",","⚾️",",",",",",","],

                  3:[",",",",",",",",",","],

                  4:["⌚️",",",",","⌨️",",",",","☎️"],

                  5:[",",",",",",",",",","]]

    lazy var emojiChoices = themes[0]!

    var emoji = [Int:String]()

    func emoji(for card: Card) -> String {

        if emoji[card.identifier] == nil, emojiChoices.count > 0{

            let randomIndex = Int(arc4random\_uniform(UInt32(emojiChoices.count)))

            emoji[card.identifier] = emojiChoices.remove(at: randomIndex)

        }

        return emoji[card.identifier] ?? "?"

    }

    @IBAction func newGame(\_ sender: Any) {

        game = Concentration(numberOfPairsOfCards: (cardButtons.count + 1 ) / 2)

        let them = Int(arc4random\_uniform(UInt32(themes.keys.count)))

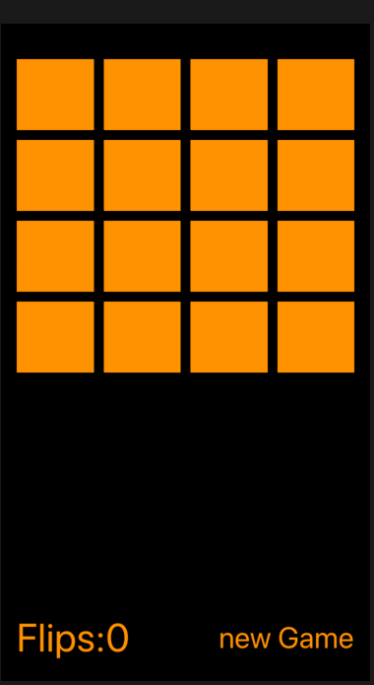
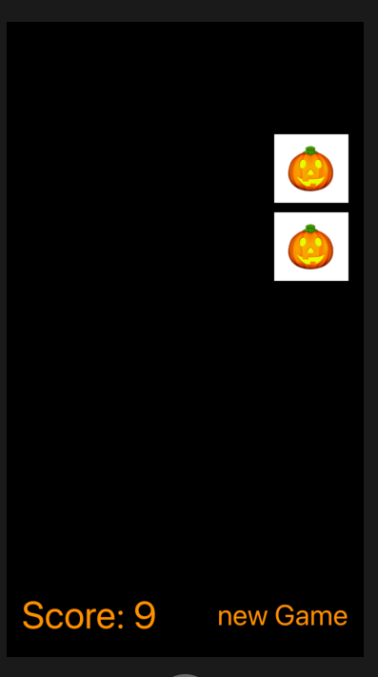
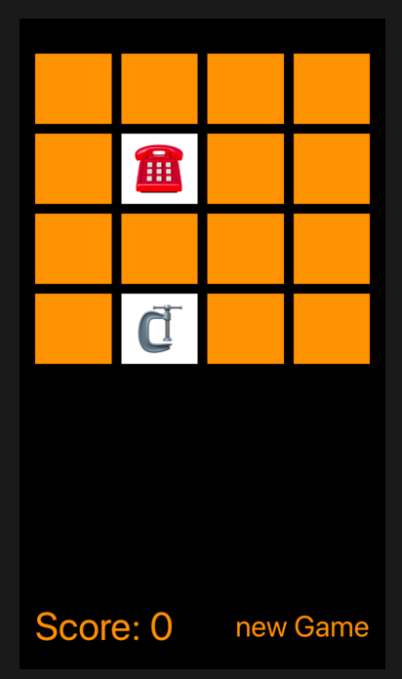
        emojiChoices = themes[them]!

        updateViewFromModel()

    }

}

* 运行截图：

1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

Github地址：<https://github.com/Gzqqqqq/IOSExperimentReport.git>

本次试验是跟着斯坦福教程学做一个小游戏，因为是跟着视频一步一步的学着做的，所以并没有存在很大的问题，跟着视频学做了一遍以后，主要用到的技术有，页面的布局，横屏以及竖屏，按钮的点击事件，每次点击按钮以后都会触发一个事件，而这个事件就是是否抵消翻牌。还有一个就是卡片计数，点击一次就加一，记分数就是，走两步未抵消就减一，抵消就加二。总的来说，做这个游戏收获还是蛮大的。跟进一步的了解啦iOS的开发机制。

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。