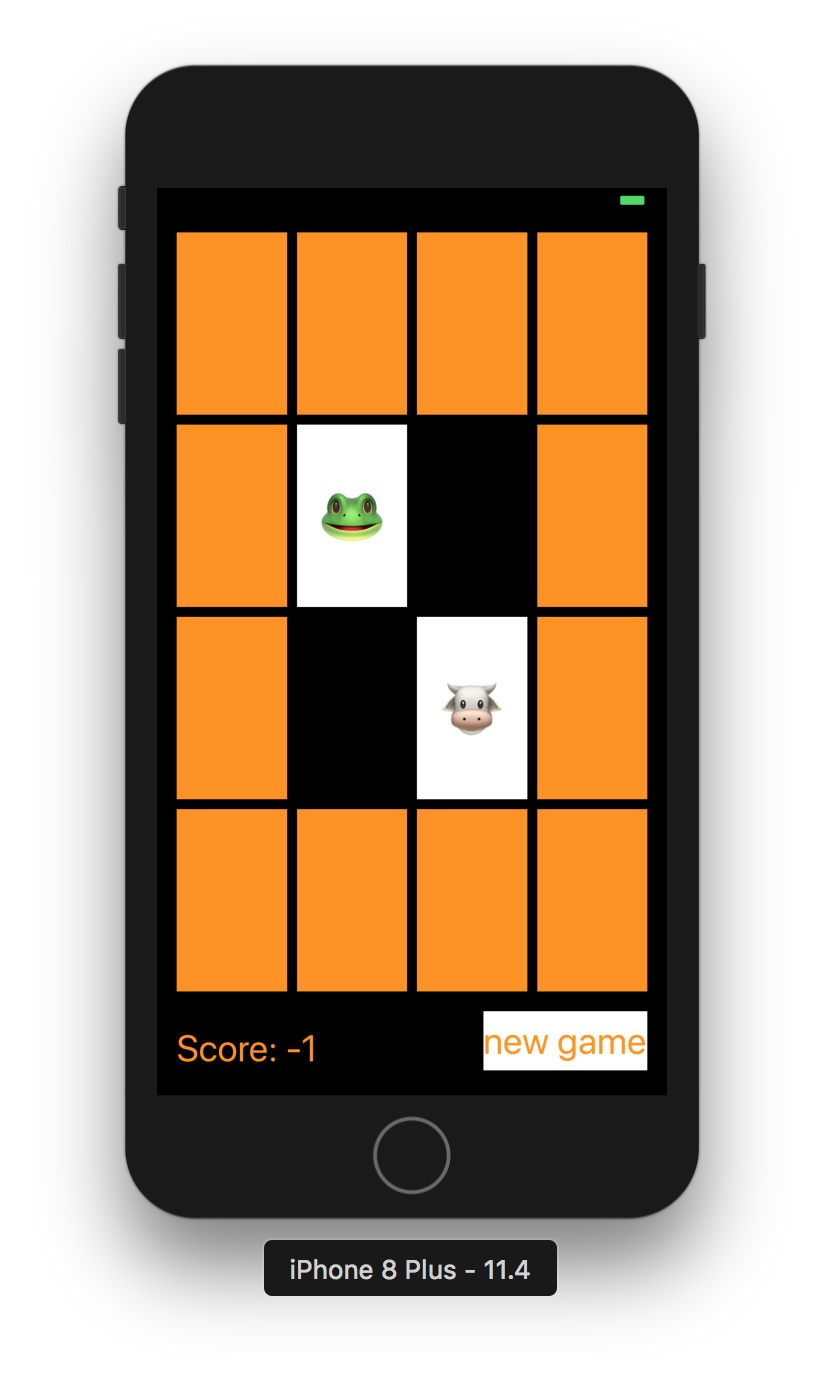
实验编号： 7 **四川师大《IOS》实验报告 2019** 年 **11** 月 **2** 日

### **计算机科学学院** 2017 级 4 班 实验名称： Game单MVC \_

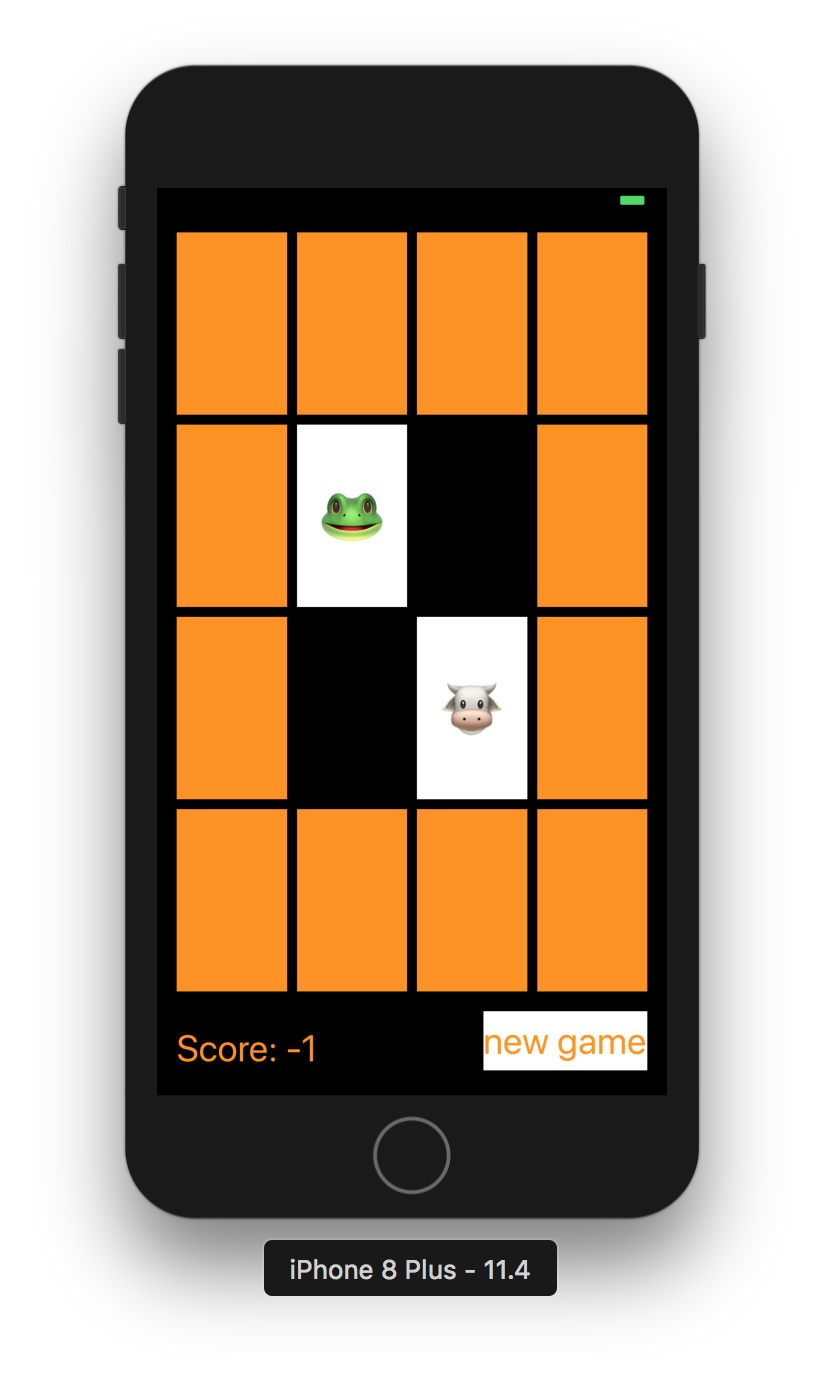
姓名：\_李晋\_\_ 学号：\_2017110517\_\_\_指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**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. **实验要求**
5. 认真填写实验报告，要求附加部分运行界面和主要代码；
6. 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；
7. **实验内容**
8. 参照Stanford视频1和2完成一个game（Concentration）的制作；
9. 在(1)的基础上进一步完成Stanford Assignment 1的完整要求；
10. 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代码：

import Foundation

struct Card

{

    var isFaceUp = false

    var isMatched = false

    var identifier: Int

    static var identifierFactory = 0

    static func getUniqueIdentifier() -> Int{

        Card.identifierFactory += 1

        return Card.identifierFactory

    }

    init(){

        self.identifier = Card.getUniqueIdentifier()

    }

}

Concentration代码：

import Foundation

class Concentration

{

    //所有卡片的数组

    var cards = [Card]()

    //记录是否有一张而且仅有一张卡朝上的卡片的索引，类型为可选类型

    var indexOfOneAndOnlyFaceUpCard: Int?

    //匹配成功时加分标准

    var addPoint = false

    func chooseCard(at index: Int){

        if !cards[index].isMatched{

            if let matchIndex = indexOfOneAndOnlyFaceUpCard,matchIndex != index{

                //check if cards match

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

                    cards[matchIndex].isMatched = true

                    cards[index].isMatched = true

                    addPoint = true

                }

                cards[index].isFaceUp = true

                indexOfOneAndOnlyFaceUpCard = nil

            }else{

                //either no cards or 2 cards are face up

                for flipDownIndex in cards.indices{

                    cards[flipDownIndex].isFaceUp = false

                }

                cards[index].isFaceUp = true

                indexOfOneAndOnlyFaceUpCard = index

            }

        }

    }

    init(numberOfParisOfCards: Int){

        for \_ in 0..<numberOfParisOfCards{

            let card = Card()

//            cards.append(card)

//            cards.append(card)

            //下列式子等价于注释

            cards += [card,card]

        }

        // TODO:Shuffle the cards

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

    }

}

ViewController代码：

import UIKit

class ViewController: UIViewController {

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

    //得分情况

    @IBOutlet weak var flipCountLabel: UILabel!

    //所有卡片关联的出口，所有卡片的集合

    @IBOutlet var cardButtons: [UIButton]!

    //加分

    var scores = 0 { didSet { flipCountLabel.text = "Scores : \(scores)" } }

    //所有UIButton所关联的action

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

        //点击卡片开始

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

            game.chooseCard(at: cardNumber)

            updateViewFromModel()

        }else{

            print("chosen card was no in the cardButtons")

        }

    }

    //从模型更新视图,如果点击没翻的卡片则翻出

    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 = UIColor.white

                //匹配成功则加分

                if game.addPoint{

                    scores += 10

                    game.addPoint = false

                }

            }else{

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

                button.backgroundColor = card.isMatched ? UIColor.clear :  colorLiteral(red: 1, green: 0.5285960501, blue: 0.1228347408, alpha: 1)

            }

        }

    }

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

        for index in cardButtons.indices{

            game.cards[index].isMatched = false

            game.cards[index].isFaceUp = false

            cardButtons[index].setTitle("",for:UIControlState.normal)

            cardButtons[index].backgroundColor =  colorLiteral(red: 1, green: 0.5285960501, blue: 0.1228347408, alpha: 1)

        }

        scores = 0

    }

    var emojiChoices = ["🐳","🌚","🌏","🌝","🍟","⚽️","🌽","🍩"]

    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] ?? "?"

    }

    override func viewDidLoad() {

        super.viewDidLoad()

        // Do any additional setup after loading the view, typically from a nib.

    }

    override func didReceiveMemoryWarning() {

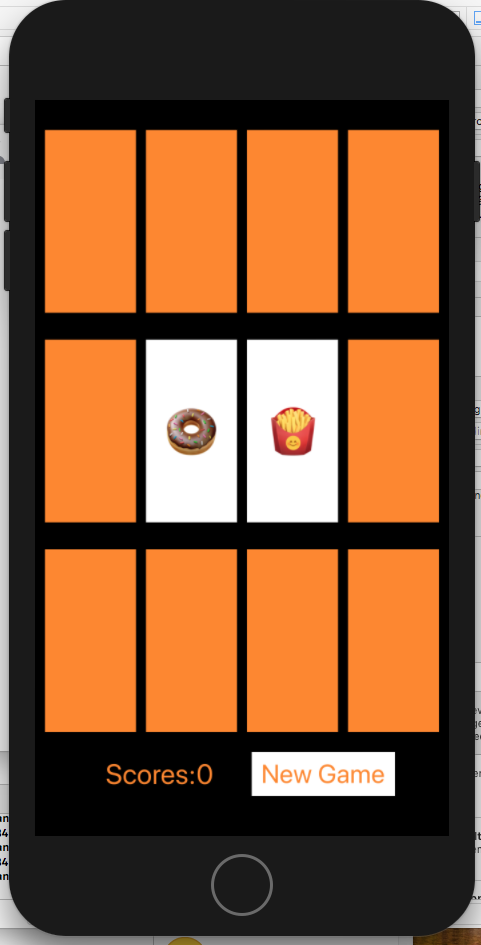
        super.didReceiveMemoryWarning()

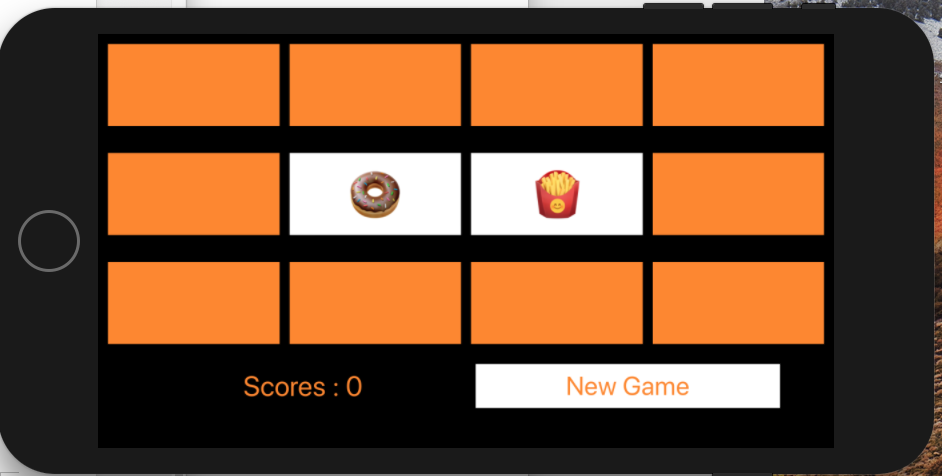
        // Dispose of any resources that can be recreated.

    }

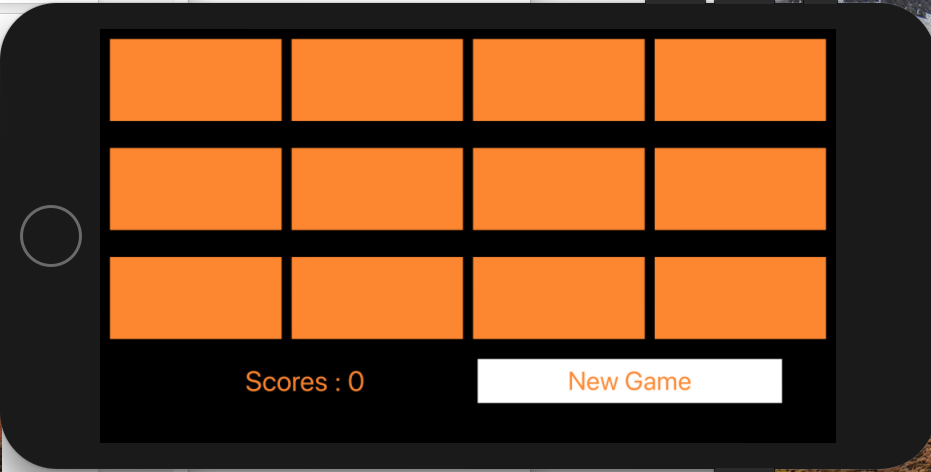
}

* 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）**
2. 通过此次实验感受到了软件开发的魅力，能够跟着视频写一个游戏出来非常有成就感。
3. 进一步掌握了MVC的思想，利用stackView使布局更加简单，在实验过程中，熟练控件与代码的关联。
4. 通过此次实验学会了代码调试

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。