实验9 Gesture、UIAlertController、 ScrollView

（综合性实验 2学时）

1、实验目的

1. 理解并掌握iOS多点手势识别的相关技术；

2、实验要求

1. 认真填写实验报告，要求附加部分运行界面和主要代码；
2. 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；

3、实验内容

1. Gesture
   1. 分别采用代码随机位置大小生成和直接拖拽的方式产生多个视图；
   2. 采用简单的动画进行移动；
   3. 给视图加上阴影(layer)；
   4. 可全部清空子视图；
   5. 视图支持手势（pan移动、tap删除、pinch缩放、rotation旋转）；

提示：Pinch的scale属性可用于调整frame

rotation需要用transform属性实现

代码：

//

// CircleView.swift

// 实验9-2

//

// Created by student on 2018/11/10.

// Copyright © 2018年 zoulin. All rights reserved.

//

import UIKit

@IBDesignable//使视图在main.storyboard课件可见

class CircleView: UIView {

//实现手势控制

func setup(){

//设置阴影

self.layer.cornerRadius = 50

self.layer.shadowColor = UIColor.black.cgColor

self.layer.shadowOffset = CGSize(width: 5, height: 5)

self.layer.shadowOpacity = 0.8

//设置视图的内容模式为重绘

self.contentMode = .redraw

//pan移动效果

let panRecognizer = UIPanGestureRecognizer(target: self, action: #selector(pan(recognizer:)))

self.addGestureRecognizer(panRecognizer)// 实现交互

//tap删除效果

let tapGesture = UITapGestureRecognizer(target: self, action: #selector(tap(recognizer:)))

self.addGestureRecognizer(tapGesture)

tapGesture.numberOfTouchesRequired = 1

tapGesture.numberOfTapsRequired = 2

//pinch缩放效果

let pinchGesture = UIPinchGestureRecognizer(target: self, action: #selector(pinch(recognizer:)))

self.addGestureRecognizer(pinchGesture)

//rotation旋转效果

let rotationGesture = UIRotationGestureRecognizer(target: self, action: #selector(rotation(gestrue:)))

self.addGestureRecognizer(rotationGesture)

}

//必须加上@objc ，否则不能识别，添加移动效果

@objc func pan(recognizer:UIPanGestureRecognizer){

//print("手势识别")

//设置手势识别的作用效果

switch recognizer.state{

case .changed:

fallthrough

case.ended:

let translation = recognizer.translation(in: self)

center.x += translation.x//移动距离为累加，速度太快，看不到效果

center.y += translation.y

//将每一次动作前的位移量清零

recognizer.setTranslation(.zero, in: self)

default:

break

}

}

//tap删除

@objc func tap(recognizer: UITapGestureRecognizer) {

//设置手势识别的作用效果

switch recognizer.state{

case .recognized:

print("双击删除")

default:

break

}

}

//pinch缩放

@objc func pinch(recognizer: UIPinchGestureRecognizer) {

print("手势控制缩放")

// let scale = gesture.scale

// self.transform = self.transform.scaledBy(x: scale, y: scale)

// gesture.scale = 1

//设置手势识别的作用效果

switch recognizer.state{

case .changed:

fallthrough

case.ended:

bounds.size = CGSize(width: bounds.width \* recognizer.scale, height: bounds.height \* recognizer.scale)

recognizer.scale = 1 //缩放比例控制

default:

break

}

}

//rotation旋转

@objc func rotation(gestrue: UIRotationGestureRecognizer) {

print("手势控制旋转")

let rotation = gestrue.rotation

self.transform = self.transform.rotated(by: rotation)

gestrue.rotation = 0

}

override init(frame: CGRect) {

super.init(frame: frame)

setup()

}

required init?(coder aDecoder: NSCoder) {

super.init(coder:aDecoder)

setup()

}

@IBInspectable var fillColor:UIColor?//使视图能够在main.storyboard中调整颜色

@IBInspectable var strokeColor:UIColor?

// Only override draw() if you perform custom drawing.

// An empty implementation adversely affects performance during animation.

override func draw(\_ rect: CGRect) {

// Drawing code

//画圆

let path = UIBezierPath(rect: rect)

//颜色填充

fillColor?.setFill()

strokeColor?.setStroke()

path.fill()

path.stroke()

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//

// ViewController.swift

// 实验9-2

//

// Created by student on 2018/11/10.

// Copyright © 2018年 zoulin. All rights reserved.

//

import UIKit

class ViewController: UIViewController {

override func viewDidLoad() {

super.viewDidLoad()

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

}

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

//定义随机添加的位置

let x = Int(arc4random())%Int(view.bounds.width)

let y = Int(arc4random())%Int(view.bounds.height)

let maxWidth: CGFloat = 150

let width = Int(arc4random() % UInt32(maxWidth))

let height = Int(arc4random() % UInt32(maxWidth))

let label = UILabel(frame: CGRect(x:x,y:y,width:width,height:height))

label.text = "hands"

label.textAlignment = .center

label.backgroundColor = UIColor.orange

label.layer.shadowColor = UIColor.gray.cgColor

label.layer.shadowOffset = CGSize(width: 5, height: 5)

label.layer.shadowOpacity = 0.8

label.isUserInteractionEnabled = true

//pan移动效果

let panRecognizer = UIPanGestureRecognizer(target: self, action: #selector(pan(recognizer:)))

label.addGestureRecognizer(panRecognizer)// 实现交互

//tap删除效果

let tapRecognizer = UITapGestureRecognizer(target: self, action: #selector(tap(recognizer:)))

label.addGestureRecognizer(tapRecognizer)

//pinch缩放效果

let pinchRecognizer = UIPinchGestureRecognizer(target: self, action: #selector(pinch(recognizer:)))

label.addGestureRecognizer(pinchRecognizer)

//rotation旋转效果

let rotationGesture = UIRotationGestureRecognizer(target: self, action: #selector(rotation(gestrue:)))

label.addGestureRecognizer(rotationGesture)

self.view.addSubview(label)

}

//必须加上@objc ，否则不能识别，添加移动效果

@objc func pan(recognizer:UIPanGestureRecognizer){

//print("手势识别")

//设置手势识别的作用效果

switch recognizer.state{

case .changed:

fallthrough

case.ended:

let translation = recognizer.translation(in: self.view)

recognizer.view?.center.x += translation.x//移动距离为累加，速度太快，看不到效果

recognizer.view?.center.y += translation.y

//将每一次动作前的位移量清零

recognizer.setTranslation(.zero, in: self.view)

default:

break

}

}

//tap删除

@objc func tap(recognizer: UITapGestureRecognizer) {

//设置手势识别的作用效果

if recognizer.state == .recognized {

recognizer.view?.removeFromSuperview()

}

}

//pinch缩放

@objc func pinch(recognizer: UIPinchGestureRecognizer) {

let scale = recognizer.scale

recognizer.view?.transform = self.view.transform.scaledBy(x: scale, y: scale)

recognizer.scale = 1

}

//rotation旋转

@objc func rotation(gestrue: UIRotationGestureRecognizer) {

print("手势控制旋转")

let rotation = gestrue.rotation

gestrue.view?.transform = self.view.transform.rotated(by: rotation)

gestrue.rotation = 1

}

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

for label in view.subviews{

if label is UILabel{

//定义随机添加的位置

UIView.animate(withDuration: 1){

//view必须加上self才不会报错

let x = Int(arc4random())%Int(self.view.bounds.width)

let y = Int(arc4random())%Int(self.view.bounds.height)

label.center.x = CGFloat(x)

label.center.y = CGFloat(y)

}

}

}

}

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

for label in view.subviews{

if label is UILabel {

label.removeFromSuperview()

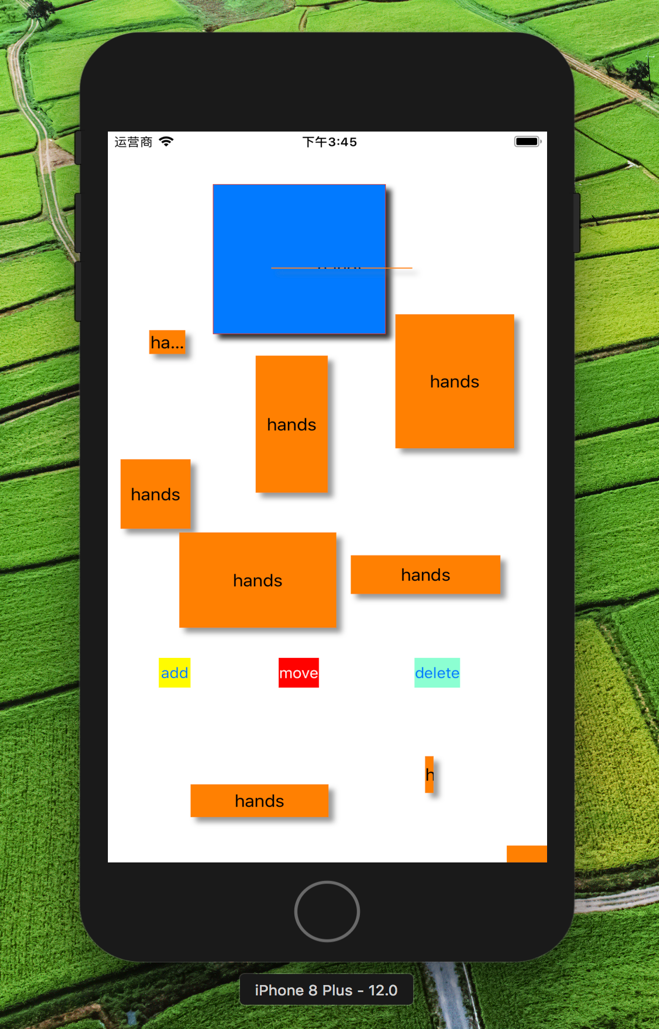
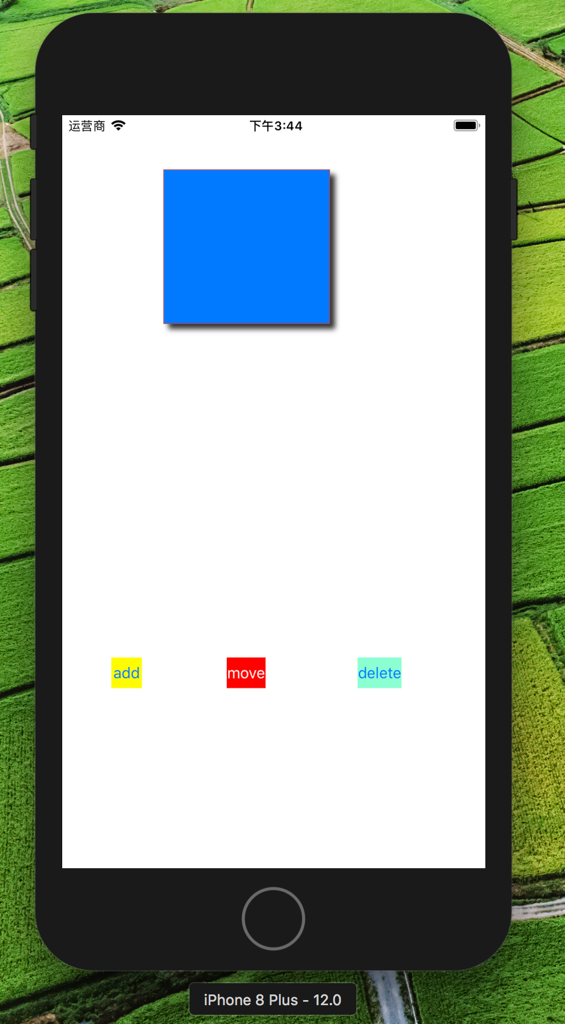
}

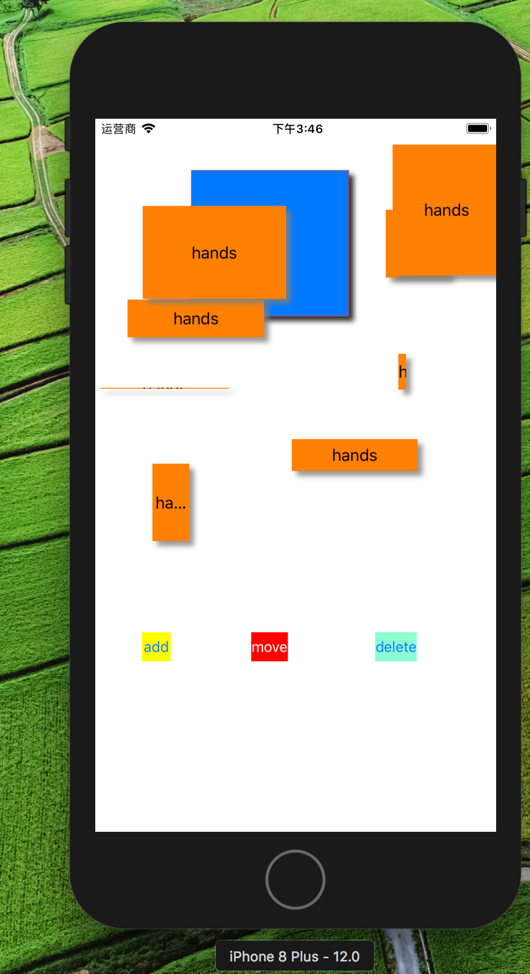
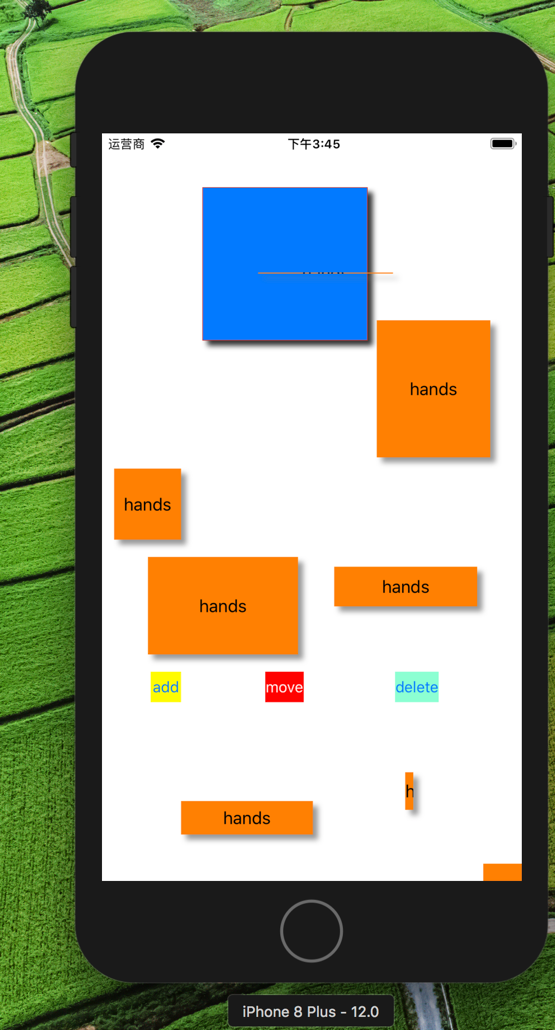
}

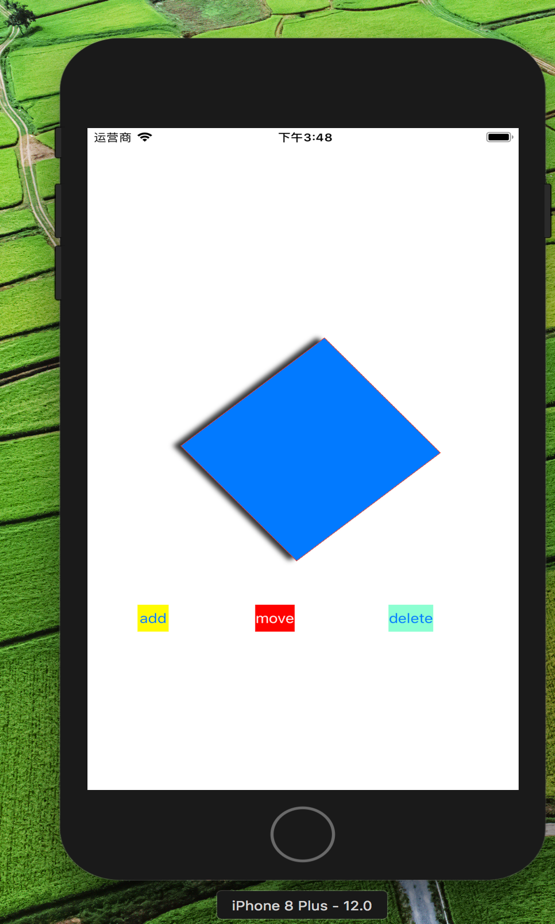
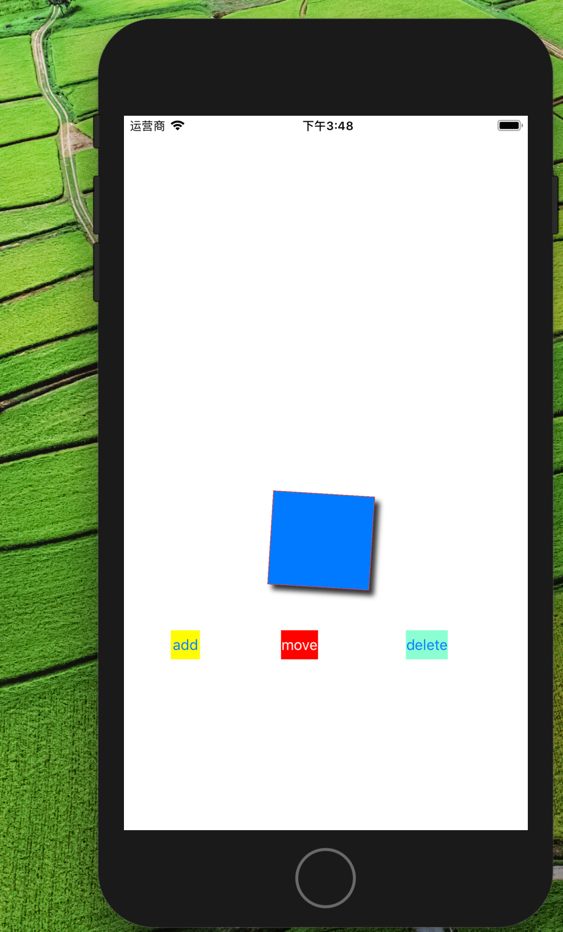
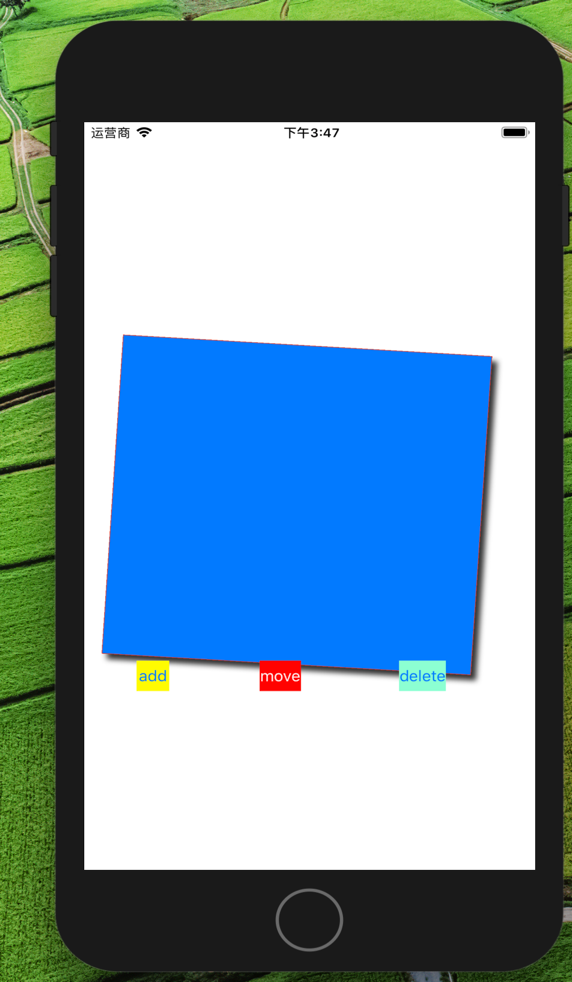
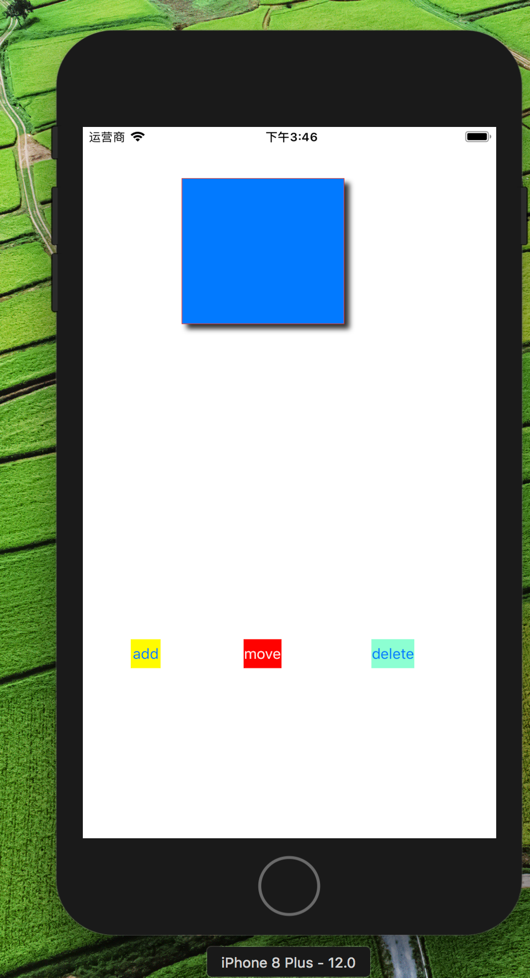
}

}

截图：







1. 实现UIAlertController交互
   1. 显示ActionSheet并进行交互；
   2. 显示Login Alert并进行交互；

代码：

//

// ViewController.swift

// 实验9

//

// Created by student on 2018/11/7.

// Copyright © 2018年 zoulin. All rights reserved.

//

import UIKit

class ViewController: UIViewController {

override func viewDidLoad() {

super.viewDidLoad()

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

}

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

let alert = UIAlertController(title: "action sheet", message: "this is a new message", preferredStyle: .actionSheet)

alert.addAction(UIAlertAction(title: "gray", style: .default, handler: { (action) in

print("ok clicked")

self.view.backgroundColor = UIColor.gray

}))

alert.addAction(UIAlertAction(title: "green", style: .destructive, handler: { (action) in

print("destructive clicked")

self.view.backgroundColor = #colorLiteral(red: 0.721568644, green: 0.8862745166, blue: 0.5921568871, alpha: 1)

}))

alert.addAction(UIAlertAction(title: "blue", style: .cancel, handler: { (action) in

print("cancel clicked")

self.view.backgroundColor = #colorLiteral(red: 0.4745098054, green: 0.8392156959, blue: 0.9764705896, alpha: 1)

}))

present(alert, animated: true, completion: nil)

}

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

let alert = UIAlertController(title: "Login", message: "login message", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Login", style: .default, handler: {(action) in

guard let username = alert.textFields?.first?.text,let password = alert.textFields?.last?.text else{

return

}

print("username=\(username),password=\(password)")

}))

alert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: { (action) in

}))

alert.addTextField { (textField) in

textField.placeholder = "please input your name"

}

alert.addTextField { (textField) in

textField.placeholder = "please input your password"

textField.isSecureTextEntry = true

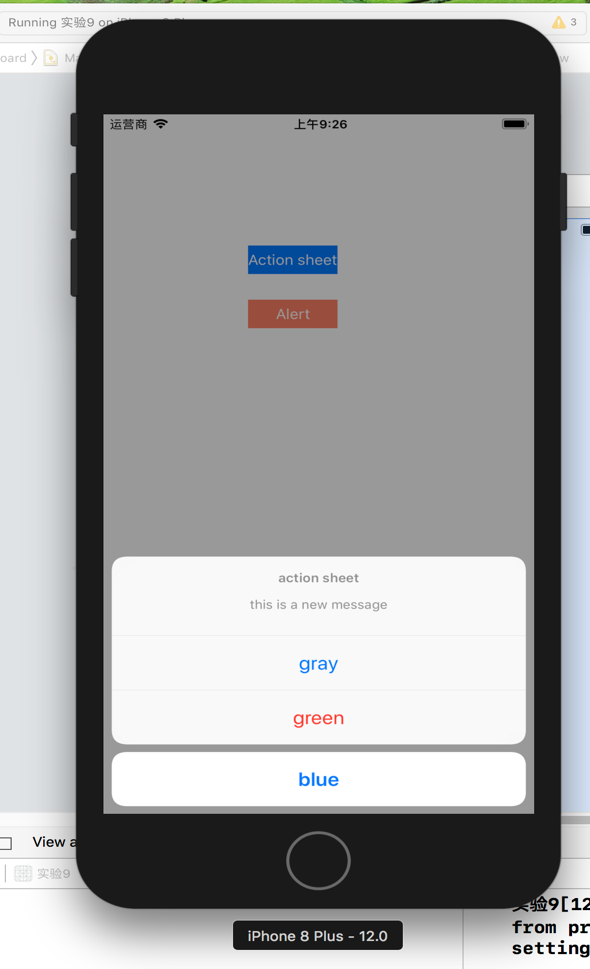
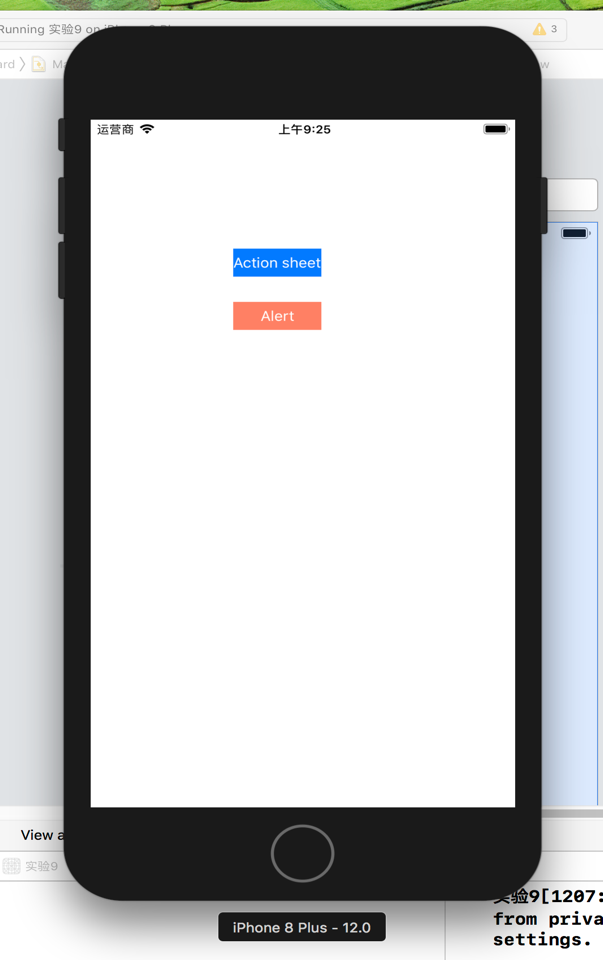
}

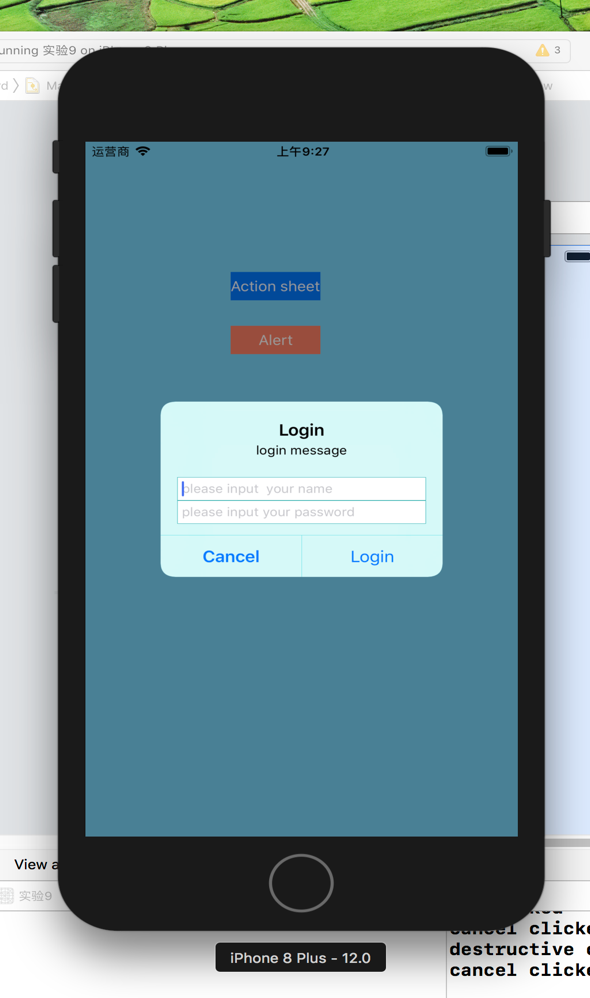
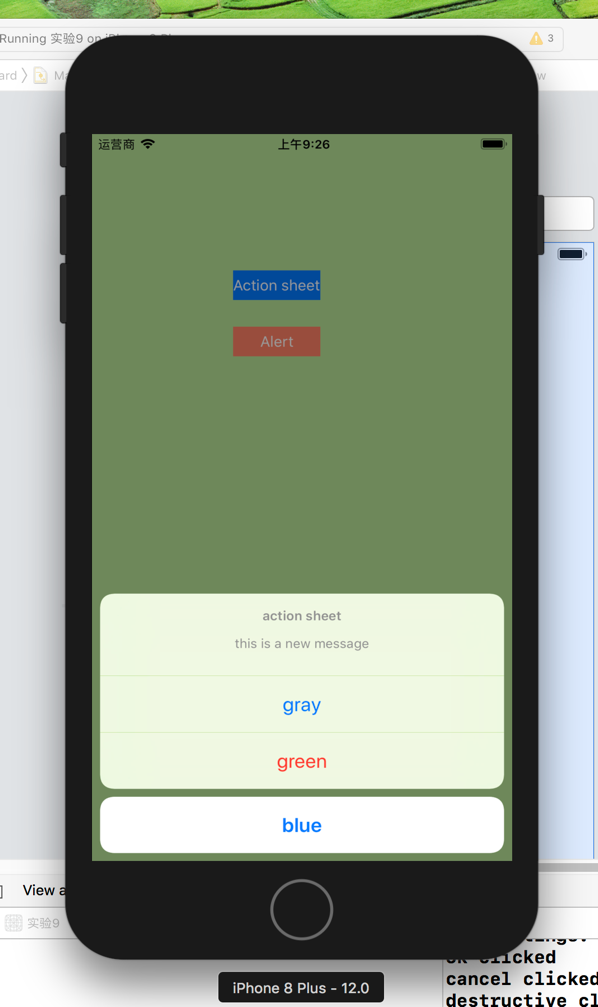
present(alert, animated: true, completion: nil)

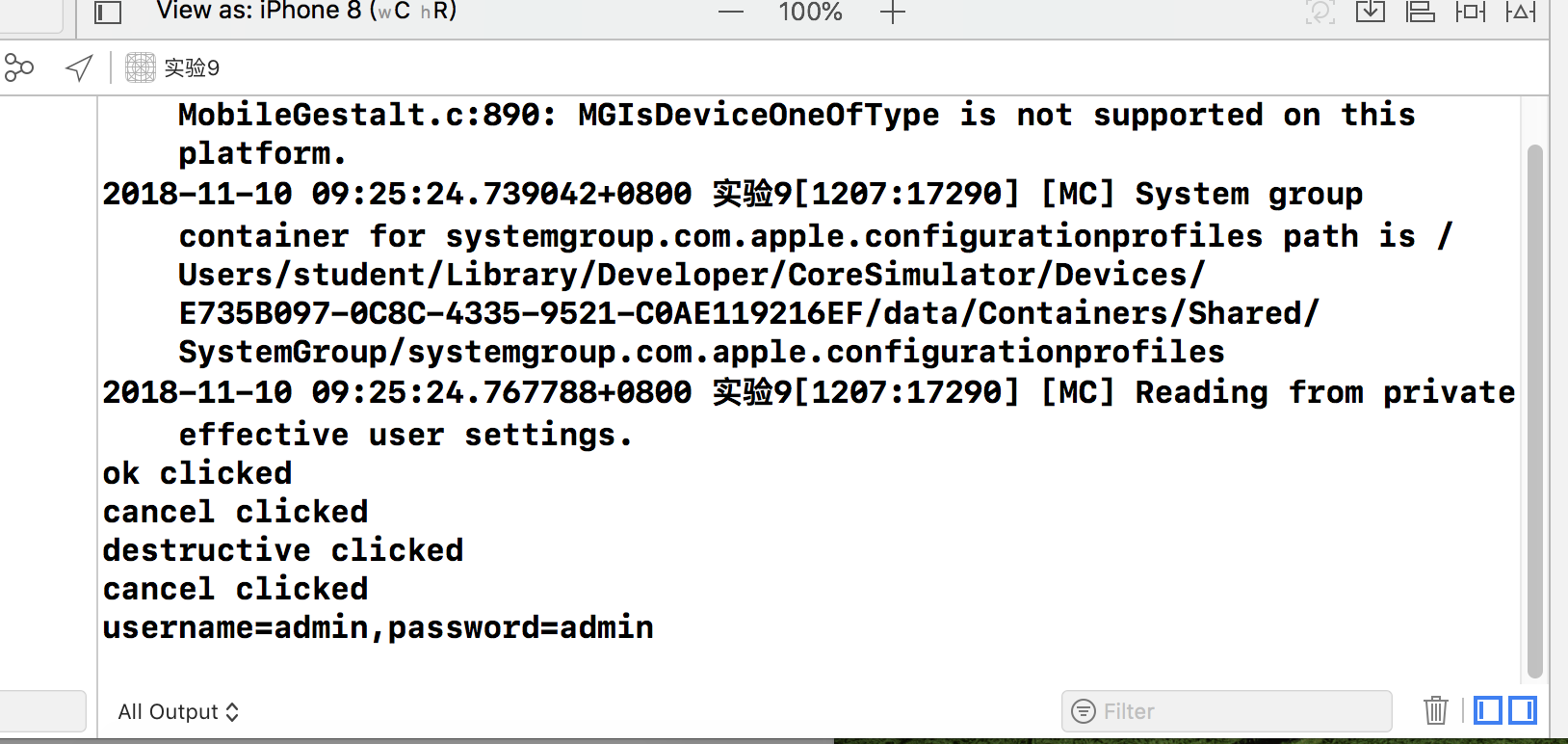
}

}

截图：







1. 一个界面使用两个scrollView
   1. 在一个scrollView中可进行多张图片横屏滚动浏览(相册)，需要有pagecontrol进行提示；
   2. 在另一个scrollView中可放大缩小；

提示：需用delegate

代码：

//

// ViewController.swift

// 实验9-2

//

// Created by student on 2018/11/7.

// Copyright © 2018年 zoulin. All rights reserved.

//

import UIKit

class ViewController: UIViewController,UIScrollViewDelegate {

@IBOutlet weak var scrollView: UIScrollView!

@IBOutlet weak var pageControll: UIPageControl!

override func viewDidLoad() {

super.viewDidLoad()

//实现图片缩放

// let imageView = UIImageView(image:UIImage(named:"1"))

// scrollView.addSubview(imageView)

// scrollView.contentSize = imageView.bounds.size

// scrollView.minimumZoomScale = 0.2

// scrollView.maximumZoomScale = 5

// scrollView.delegate = self

for i in 1...7 {

let imageView = UIImageView(image: UIImage(named: "\(i)"))

imageView.contentMode = .scaleAspectFit

imageView.frame = CGRect(x: CGFloat(i-1) \* scrollView.bounds.width, y: 0, width: scrollView.bounds.width, height: scrollView.bounds.height)

scrollView.addSubview(imageView)

}

scrollView.contentSize = CGSize(width: scrollView.bounds.width \* 7, height: scrollView.bounds.height)

scrollView.isPagingEnabled = true

scrollView.showsHorizontalScrollIndicator = false

pageControll.numberOfPages = 7

pageControll.currentPage = 0

scrollView.isPagingEnabled = true

scrollView.showsHorizontalScrollIndicator = false

}

func pageControllClicked(\_ sender: UIPageControl) {

let currentPage = sender.currentPage

let rect = CGRect(x: CGFloat(currentPage) \* scrollView.bounds.width, y: 0, width: scrollView.bounds.width, height: scrollView.bounds.height)

scrollView.scrollRectToVisible(rect, animated: true)

}

func scrollViewDidEndDecelerating(\_ scrollView: UIScrollView) {

let currentPage = scrollView.contentOffset.x/scrollView.bounds.width

pageControll.currentPage = Int(currentPage)

}

// func viewForZooming(in scrollView: UIScrollView) -> UIView? {

// return scrollView.subviews.first

// }

}

截图：

