实验编号： 4 **四川师大《IOS》实验报告 2018** 年 **9** 月 **26** 日

### **计算机科学学院** 2016 级 4 班 实验名称： 自定制视图、文件、Playground可视化

姓名：\_樊琳\_ 学号：\_2016110408\_ 指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**实验\_四\_ \_**自定制视图、文件、Playground可视化**\_**

1. 实验目的及要求
2. 掌握沙盒文件的操作；
3. 掌握自定制视图的定义及使用；
4. 掌握playground下可视化开发方法；
5. 实验内容
6. **文件缓存处理**
   1. 判断沙盒的Document目录下是否存在某文件夹，如果没有则新建一个该文件夹；
   2. 判断是否该文件夹下存在一个图片文件，如果存在该文件，读取该文件到一个图片对象中并进行显示，如果不存在则从网上下载一张图片并保存到该图片文件中；
7. **自定制视图**
   1. 从UIView中派生一个自定制的View；
   2. 绘制一个椭圆（或则自己喜欢的任何图形）；
   3. 新建视图对象并进行显示；
8. **代码版hello world（在视图控制器中加入代码）**
   1. 代码中生成label（outlet）和button；
   2. 将label和button加入根view中；
   3. button添加像self（target）发射action（selector）的操作；

实现clicked响应代码（selector）

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. **文件缓存处理**
   1. 判断沙盒的Document目录下是否存在某文件夹，如果没有则新建一个该文件夹；
   2. 判断是否该文件夹下存在一个图片文件，如果存在该文件，读取该文件到一个图片对象中并进行显示，如果不存在则从网上下载一张图片并保存到该图片文件中；

* 程序代码：

//

// ViewController.swift

// 1

//

// Created by student on 2018/12/13.

// Copyright © 2018年 fl. 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.

let manager = FileManager.default

let document = manager.urls(for: .documentDirectory, in: .userDomainMask).first?.path

let file = document?.appending("/image") //图片文件夹

//print(document!) //输出Document路径，方便查看

if manager.fileExists(atPath: file!) { //若该文件夹存在

let image = file?.appending("/bd.png") //图片文件

if manager.fileExists(atPath: image!) { //若该文件存在，则显示到界面上

do{

let data = try Data(contentsOf: URL(fileURLWithPath: image!))

let img = UIImage(data: data)

let imageView = UIImageView(image: img)

imageView.frame = CGRect(x: 0, y: 100, width: 400, height: 300)

self.view.addSubview(imageView)

} catch {

print(error)

}

} else { //若不存在，则从网络下载一个图片并保存为图片文件

let url = URL(string: "https://ss0.bdstatic.com/5aV1bjqh\_Q23odCf/static/superman/img/logo/bd\_logo1\_31bdc765.png")

do{

let data = try Data(contentsOf: url!)

try data.write(to: URL(fileURLWithPath: image!), options: .atomicWrite)

print("文件不存在，已成功创建文件")

} catch {

print(error)

}

}

} else { //若该文件夹不存在，则创建该文件夹

do {

try manager.createDirectory(atPath: file!, withIntermediateDirectories: true, attributes: nil)

print("文件夹不存在，已成功创建文件夹")

} catch {

print(error)

}

}

}

}

* 运行结果：



1. **自定制视图**
   1. 从UIView中派生一个自定制的View；
   2. 绘制一个椭圆（或则自己喜欢的任何图形）；
   3. 新建视图对象并进行显示；

* 程序代码：

class triangleview:UIView {

override func draw(\_ rect: CGRect) {

let path=UIBezierPath()

path.move(to: CGPoint(x: rect.size.width/2, y: 0))

path.addLine(to: CGPoint(x: rect.size.width, y: rect.size.height))

path.addLine(to: CGPoint(x: 0, y: rect.size.height))

path.close()

UIColor.red.setFill()

path.fill()

}

}

override func loadView() {

view = UIView(frame: CGRect(x: 0, y: 0, width: 500, height: 500))

view.backgroundColor=UIColor.green

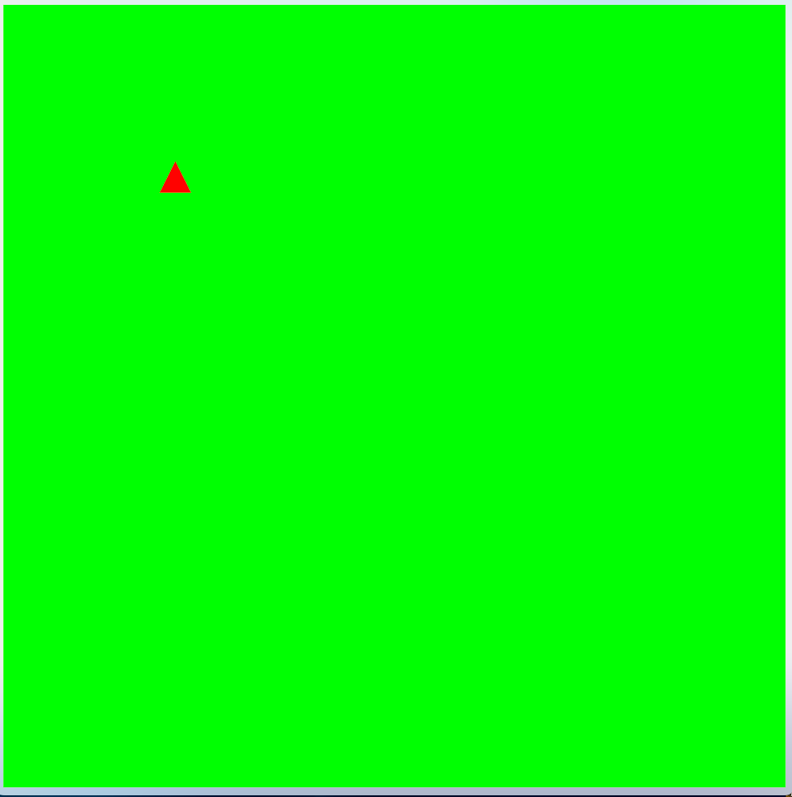
triangular=triangleview(frame: CGRect(x: 100, y: 100, width: 20, height: 20))

triangular.backgroundColor=UIColor.clear

view.addSubview(triangular)

}

* 运行结果：



1. **代码版hello world（在视图控制器中加入代码）**
   1. 代码中生成label（outlet）和button；
   2. 将label和button加入根view中；
   3. button添加像self（target）发射action（selector）的操作；

实现clicked响应代码（selector）

* 程序代码：

//

// AppDelegate.swift

// helloworld\_fl

//

// Created by student on 2018/10/17.

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

//

import UIKit

import CoreData

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

// Override point for customization after application launch.

window=UIWindow(frame:UIScreen.main.bounds)

window?.backgroundColor=UIColor.red

window?.rootViewController=firstViewController()

window?.makeKeyAndVisible()

return true

}

func applicationWillResignActive(\_ application: UIApplication) {

// Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

// Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

}

func applicationDidEnterBackground(\_ application: UIApplication) {

// Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

// If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

}

func applicationWillEnterForeground(\_ application: UIApplication) {

// Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

}

func applicationDidBecomeActive(\_ application: UIApplication) {

// Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

}

func applicationWillTerminate(\_ application: UIApplication) {

// Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

// Saves changes in the application's managed object context before the application terminates.

self.saveContext()

}

// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {

/\*

The persistent container for the application. This implementation

creates and returns a container, having loaded the store for the

application to it. This property is optional since there are legitimate

error conditions that could cause the creation of the store to fail.

\*/

let container = NSPersistentContainer(name: "helloworld\_fl")

container.loadPersistentStores(completionHandler: { (storeDescription, error) in

if let error = error as NSError? {

// Replace this implementation with code to handle the error appropriately.

// fatalError() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

/\*

Typical reasons for an error here include:

\* The parent directory does not exist, cannot be created, or disallows writing.

\* The persistent store is not accessible, due to permissions or data protection when the device is locked.

\* The device is out of space.

\* The store could not be migrated to the current model version.

Check the error message to determine what the actual problem was.

\*/

fatalError("Unresolved error \(error), \(error.userInfo)")

}

})

return container

}()

// MARK: - Core Data Saving support

func saveContext () {

let context = persistentContainer.viewContext

if context.hasChanges {

do {

try context.save()

} catch {

// Replace this implementation with code to handle the error appropriately.

// fatalError() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

let nserror = error as NSError

fatalError("Unresolved error \(nserror), \(nserror.userInfo)")

}

}

}

}

//

// firstViewController.swift

// helloworld\_fl

//

// Created by student on 2018/10/17.

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

//

import UIKit

class firstViewController: UIViewController {

var lable: UILabel!

override func viewDidLoad() {

super.viewDidLoad()

self.view.backgroundColor=UIColor.yellow

lable=UILabel(frame:CGRect(x: 100, y: 100, width: 200, height: 100))

lable.text="hello world"

lable.textAlignment = .center

lable.backgroundColor=UIColor.cyan

lable.center=self.view.center

self.view.addSubview(lable)

let button = UIButton(frame:CGRect(x: 150, y: 500, width: 100, height: 50))

button.setTitle("click me", for: .normal)

button.setTitle("i am happy", for: .highlighted)

button.setTitleColor(UIColor.red, for: .normal)

view.addSubview(button)

// Do any additional setup after loading the view.

button.addTarget(self, action: #selector(btnclicked), for: .touchUpInside)

self.title="first"

}

@IBAction func btnclicked(){

lable.text="i am clicked"

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

/\*

// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

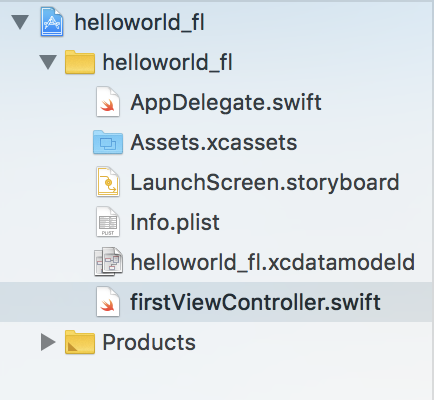
// Get the new view controller using segue.destinationViewController.

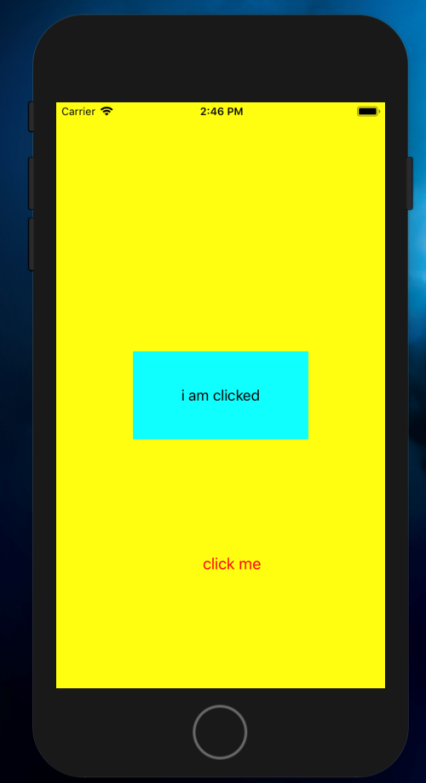
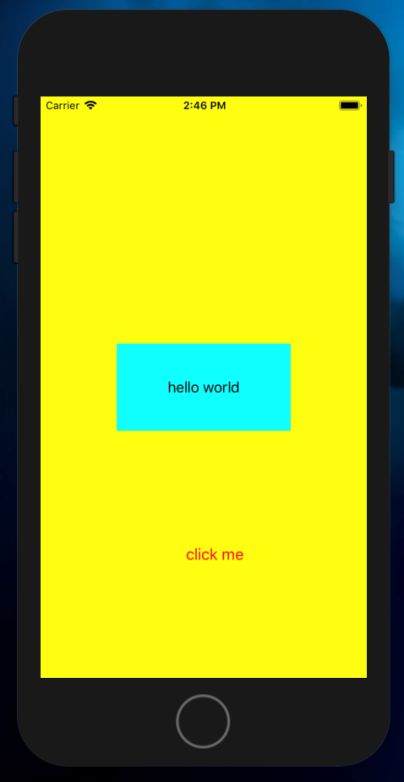
// Pass the selected object to the new view controller.

}

\*/

}

****

* 运行结果：
* 

1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

这次实验对我们的要求有：掌握沙盒文件的操作；掌握自定制视图的定义及使用；掌握playground下可视化开发方法；其实并不是很难，主要问题在沙盒的理解上，因为我们常常用windows系统，跟iOS系统的文件管理并不一样，所以要多花时间在上面。

Github地址：

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。