实验编号： 10 **四川师大《IOS》实验报告 2019** 年 **11** 月 **19** 日

### **计算机科学学院** 2017 级 4 班 实验名称： 视图动画 \_

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

**实验 十 \_\_\_**视图动画**\_\_\_\_\_**

1. **实验目的及要求**
2. 掌握视图基本动画的原理和使用；
3. 掌握动力学动画的原理和使用；
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. 基于UIView.animation实现如下动画效果：
   1. 改变视图的位置
   2. 改变视图的大小
   3. 改变视图的transform
9. 基于UIView.transition实现如下动画效果：
   1. 改变视图的背景颜色；
   2. 切换两个子视图，观察切换后视图层次的变化情况；
10. 基于UIDynamicAnimtor实现简单的力学动画：
    1. 可以掉落方块；
    2. 有碰撞；
    3. 可以反弹；
11. **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）**
12. 基于UIView.animation实现如下动画效果：
    1. 改变视图的位置
    2. 改变视图的大小
    3. 改变视图的transform
       * 程序代码：

import UIKit

class ViewController: UIViewController {

@IBOutlet weak var myView: UIView!

override func viewDidLoad() {

super.viewDidLoad()

}

//改变color

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

UIView.animate(withDuration: 3) {

self.myView.backgroundColor = self.randomColor()

}

}

//改变位置

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

UIView.animate(withDuration: 2, delay: 0, options: [.curveEaseInOut], animations: {

self.myView.center.x += 50

self.myView.center.y += 50

}, completion: nil)

}

//放大缩小scale

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(scaleX: 1.5, y: 1.5)

}

}

//移动transition

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(translationX: -100, y: -100)

}

}

//旋转rotation

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(rotationAngle: 90)

}

}

//随机颜色值

func randomColor() -> UIColor {

let red = CGFloat(arc4random()) / CGFloat(Int32.max)

let green = CGFloat(arc4random()) / CGFloat(Int32.max)

let blue = CGFloat(arc4random()) / CGFloat(Int32.max)

return UIColor(red: red, green: green, blue: blue, alpha: 1)

}

}



1. 基于UIView.transition实现如下动画效果：
   1. 改变视图的背景颜色；
   2. 切换两个子视图，观察切换后视图层次的变化情况；
      * 程序代码：

import UIKit

class ViewController: UIViewController {

@IBOutlet weak var myView: UIView!

var newView: UIView? = nil

override func viewDidLoad() {

super.viewDidLoad()

}

//改变颜色

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

if let \_ = myView{

UIView.transition(with: myView, duration: 2, options: .transitionFlipFromBottom, animations: {

self.myView.backgroundColor = UIColor.green

}, completion: nil)

}

}

//改变视图

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

if let \_ = myView{

newView = UIView(frame: myView.frame)

newView!.backgroundColor = UIColor.blue

UIView.transition(from: myView, to: newView!, duration: 2, options: .transitionCurlUp, completion: nil)

}

}

}

* + - 运行结果：



1. 基于UIDynamicAnimtor实现简单的力学动画：
   1. 可以掉落方块；
   2. 有碰撞；
   3. 可以反弹；
      * 程序代码：

import UIKit

class ViewController: UIViewController {

//11111111重力动画import UIKit

class ViewController: UIViewController {

@IBOutlet weak var myView: UIView!

override func viewDidLoad() {

super.viewDidLoad()

}

//改变color

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

UIView.animate(withDuration: 3) {

self.myView.backgroundColor = self.randomColor()

}

}

//改变位置

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

UIView.animate(withDuration: 2, delay: 0, options: [.curveEaseInOut], animations: {

self.myView.center.x += 50

self.myView.center.y += 50

}, completion: nil)

}

//放大缩小scale

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(scaleX: 1.5, y: 1.5)

}

}

//移动transition

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(translationX: -100, y: -100)

}

}

//旋转rotation

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

UIView.animate(withDuration: 2) {

self.myView.transform = CGAffineTransform(rotationAngle: 90)

}

}

//随机颜色值

func randomColor() -> UIColor {

let red = CGFloat(arc4random()) / CGFloat(Int32.max)

let green = CGFloat(arc4random()) / CGFloat(Int32.max)

let blue = CGFloat(arc4random()) / CGFloat(Int32.max)

return UIColor(red: red, green: green, blue: blue, alpha: 1)

}

}

lazy var animator = UIDynamicAnimator(referenceView: self.view)

let gravity = UIGravityBehavior() //重力

let collision = UICollisionBehavior() //碰撞

override func viewDidLoad() {

super.viewDidLoad()

//333333333333,设置这个碰撞生效

collision.translatesReferenceBoundsIntoBoundary = true

//画圆

let layer = CAShapeLayer()

layer.path = UIBezierPath(ovalIn: view.bounds).cgPath

layer.fillColor = UIColor.clear.cgColor //背景色透明

layer.strokeColor = UIColor.black.cgColor //边线颜色

view.layer.addSublayer(layer)

//77777777777777 碰撞加边界

collision.addBoundary(withIdentifier: NSString("随意写"), for: UIBezierPath(ovalIn: view.bounds))

}

//返回一个随机颜色值

func randColor() -> CGFloat{

return CGFloat(arc4random()) / CGFloat(UInt32.max)

}

//添加UIlabel控件

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

let width = Int(view.bounds.width/10) //设置为屏幕宽度的10分之1

let randx = Int((arc4random() % 10)) \* width //10以内的整数乘宽度

let rect = CGRect(x:randx,y:50,width: width,height: width)

let labelAdd = UILabel(frame: rect)

labelAdd.backgroundColor = UIColor(red: randColor(), green: randColor(), blue: randColor(), alpha: 1)

//view.addSubview(labelAdd)

view.insertSubview(labelAdd, at: 0)

//让加入的空间不挡住文字，加在0层

//2222222222222

gravity.addItem(labelAdd)

collision.addItem(labelAdd)

}

//添加动画

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

//44444444444

animator.addBehavior(gravity)

animator.addBehavior(collision)

}

//移除动画

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

//55555555555

animator.removeBehavior(gravity)

animator.removeBehavior(collision)

}

//666666666666666

//设置重力方向

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

gravity.gravityDirection = CGVector(dx: -1, dy: 0)

}

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

gravity.gravityDirection = CGVector(dx: 1, dy: 0)

}

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

gravity.gravityDirection = CGVector(dx: 0, dy: -1)

}

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

gravity.gravityDirection = CGVector(dx: 0, dy: 1)

}

}









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. 学习三种不同的动画，知道了这三种动画的应用场景，与使用方式。
3. 还有更多的动画实验中没有做到，它们的功能都十分强大。
4. 熟悉了前面实验内容，用代码绘制椭圆
5. 设置碰撞效果是要设置这个碰撞生效，collision.translatesReferenceBoundsIntoBoundary = true，不然就没有效果

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。