**课 程 设 计 报 告**

**课程设计名称：软件工程与计算**

**课程设计题目**：**Flappy Bird**

院（系）：计算机科学技术学院

专业：网络编程

班级：2015网络编程

学号：20151104681

姓名：敖婷

Github网址：ciec20151104681

指导教师：朝力萌老师

完成日期：2018.7

**目 录**

[第1章 概要设计 3](#_Toc406624046)

[1.1题目的内容与要求 3](#_Toc406624047)

[1.2总体结构 3](#_Toc406624048)

[第2章 详细设计 4](#_Toc406624049)

[第3章 调试分析 16](#_Toc406624051)

[第4章 使用说明与执行结果 17](#_Toc406624052)

[4.1运行操作及结果 17](#_Toc406624047)

[4.2课程设计小结 19](#_Toc406624048)

# 第1章概要设计

## 1.1题目的内容与要求

**内容：**

通过Xcode软件，用swift语言编写出的一款类似于FlappyBird，可以控制一个鸟儿上升或下降来通过障碍物以得分的游戏。软件可用来解压，使人放松心情。

**功能：**

1. 有游戏开始界面
2. 点开始按钮鸟开始运动
3. 可以通过点击鼠标使鸟开始上升
4. 可以一直点击鼠标使之不断上升
5. 碰撞到障碍物游戏结束
6. 通过一个障碍物可累积一分
7. 游戏结束后有最佳分数
8. 鸟的起飞、下落、碰撞、得分都有相应的音效

## 1.2总体结构

Flappy Bird

设置背景的移动

得分标签和记分板

检查是否撞击障碍物

鸟的不断起飞上升

鸟的起飞

鸟的掉落

开始场景主角出现

障碍物的移动

音效

# 第2章详细设计

### 2.1定义

let k前景地面数 = 2

let k地面移动速度:CGFloat = -150.0

let k重力:CGFloat = -1100.0

let k上冲速度:CGFloat = 400.0

var 速度 = CGPoint.zero

let k底部障碍最小乘数:CGFloat = 0.1

let k底部障碍最大乘数:CGFloat = 0.6

let k缺口乘数:CGFloat = 5

let k首次生成障碍延迟:NSTimeInterval = 2

let k每次重生障碍延迟:NSTimeInterval = 1.5

let k顶部留白:CGFloat = 20.0

let k字体名字 = "AmericanTypewriter-Bold"

var 得分标签:SKLabelNode!

var 当前分数 = 0

let k动画延迟 = 0.3

var 撞击了地面 = false

var 撞击了障碍物 = false

var 当前游戏状态: 游戏状态 = .游戏

let 世界单位 = SKNode()

let 帽子 = SKSpriteNode(imageNamed: "Sombrero")

var 游戏区域起始点: CGFloat = 0

var 游戏区域的高度: CGFloat = 0

let 主角 = SKSpriteNode(imageNamed: "Bird0")

var 上一次更新时间:NSTimeInterval = 0

var dt:NSTimeInterval = 0

let 叮的音效 = SKAction.playSoundFileNamed("ding.wav", waitForCompletion: false)

let 拍打的音效 = SKAction.playSoundFileNamed("flapping.wav", waitForCompletion: false)

let 摔倒的音效 = SKAction.playSoundFileNamed("whack.wav", waitForCompletion: false)

let 下落的音效 = SKAction.playSoundFileNamed("falling.wav", waitForCompletion: false)

let 撞击地面的音效 = SKAction.playSoundFileNamed("hitGround.wav", waitForCompletion: false)

let 砰的音效 = SKAction.playSoundFileNamed("pop.wav", waitForCompletion: false)

let 得分的音效 = SKAction.playSoundFileNamed("coin.wav", waitForCompletion: false)

### 2.2设置角色

func 设置主角(){

主角.position = CGPoint(x:size.width \* 0.2,y:游戏区域的高度 \* 0.4 + 游戏区域起始点)

主角.zPosition = 图层.游戏角色.rawValue

let offsetX = 主角.size.width \* 主角.anchorPoint.x

let offsetY = 主角.size.height \* 主角.anchorPoint.y

let path = CGPathCreateMutable()

CGPathMoveToPoint(path, nil, 3 - offsetX, 12 - offsetY)

CGPathAddLineToPoint(path, nil, 18 - offsetX, 22 - offsetY)

CGPathAddLineToPoint(path, nil, 28 - offsetX, 27 - offsetY)

CGPathAddLineToPoint(path, nil, 39 - offsetX, 23 - offsetY)

CGPathAddLineToPoint(path, nil, 39 - offsetX, 9 - offsetY)

CGPathAddLineToPoint(path, nil, 25 - offsetX, 4 - offsetY)

CGPathAddLineToPoint(path, nil, 5 - offsetX, 2 - offsetY)

CGPathCloseSubpath(path)

主角.physicsBody = SKPhysicsBody(polygonFromPath: path)

主角.physicsBody?.categoryBitMask = 物理层.游戏角色

主角.physicsBody?.collisionBitMask = 0

主角.physicsBody?.contactTestBitMask = 物理层.障碍物 | 物理层.地面

世界单位.addChild(主角)

}

### 2.3设置主菜单

func 设置主菜单(){

//logo

let logo = SKSpriteNode(imageNamed: "Logo")

logo.position = CGPoint(x:size.width/2,y:size.height \* 0.8)

logo.name = "主菜单"

logo.zPosition = 图层.UI.rawValue

世界单位.addChild(logo)

//开始游戏按钮

let 开始游戏按钮 = SKSpriteNode(imageNamed: "Button")

开始游戏按钮.position = CGPoint(x:size.width \* 0.5,y:size.height \* 0.25)

开始游戏按钮.name = "主菜单"

开始游戏按钮.zPosition = 图层.UI.rawValue

世界单位.addChild(开始游戏按钮)

let 游戏 = SKSpriteNode(imageNamed: "Play")

游戏.position = CGPoint.zero

开始游戏按钮.addChild(游戏)

}

### 2.4设置背景和前景

func 设置背景() {

let 背景 = SKSpriteNode(imageNamed: "Background")

背景.anchorPoint = CGPoint(x: 0.5, y: 1.0)

背景.position = CGPoint(x: size.width/2, y: size.height)

背景.zPosition = 图层.背景.rawValue

世界单位.addChild(背景)

游戏区域起始点 = size.height - 背景.size.height

游戏区域的高度 = 背景.size.height

let 左下 = CGPoint(x: 0, y: 游戏区域起始点)

let 右下 = CGPoint(x: size.width, y: 游戏区域起始点)

self.physicsBody = SKPhysicsBody(edgeFromPoint: 左下, toPoint: 右下)

self.physicsBody?.categoryBitMask = 物理层.地面

self.physicsBody?.collisionBitMask = 0

self.physicsBody?.contactTestBitMask = 物理层.游戏角色

}

func 设置前景() {

for i in 0..<k前景地面数{

let 前景 = SKSpriteNode(imageNamed: "Ground")

前景.anchorPoint = CGPoint(x: 0, y: 1.0)

前景.position = CGPoint(x: CGFloat(i) \* 前景.size.width, y: 游戏区域起始点)

前景.zPosition = 图层.前景.rawValue

前景.name = "前景"

世界单位.addChild(前景)

}

}

### 2.5设置记分板

func 设置记分板(){

if 当前分数 > 最高分(){

设置最高分(当前分数)

}

let 记分板 = SKSpriteNode(imageNamed: "ScoreCard")

记分板.position = CGPoint(x:size.width/2,y:size.height/2)

记分板.zPosition = 图层.UI.rawValue

世界单位.addChild(记分板)

let 当前分数标签 = SKLabelNode(fontNamed: k字体名字)

当前分数标签.fontColor = SKColor(red: 11.2, green: 22.2, blue: 0.234, alpha: 29.1)

当前分数标签.position = CGPoint(x: -记分板.size.width/4,y:-记分板.size.height/3)

当前分数标签.text = "\(当前分数)"

当前分数标签.zPosition = 图层.UI.rawValue

记分板.addChild(当前分数标签)

let 最高分标签 = SKLabelNode(fontNamed: k字体名字)

最高分标签.fontColor = SKColor(red: 101.0/255.0, green: 71.0/255.0, blue: 73.0/255.0, alpha: 1.0)

最高分标签.position = CGPoint(x: 记分板.size.width/4,y:-记分板.size.height/3)

最高分标签.text = "\(最高分())"

最高分标签.zPosition = 图层.UI.rawValue

记分板.addChild(最高分标签)

let 游戏结束 = SKSpriteNode(imageNamed:"GameOver")

游戏结束.position = CGPoint(x:size.width/2,y:size.height/2 + 记分板.size.height/2 + k顶部留白 + 游戏结束.size.height/2)

游戏结束.zPosition = 图层.UI.rawValue

世界单位.addChild(游戏结束)

let ok按钮 = SKSpriteNode(imageNamed: "Button")

ok按钮.position = CGPoint(x:size.width/2,y:size.height/2 - 记分板.size.height/2 - k顶部留白 - ok按钮.size.height/2)

ok按钮.zPosition = 图层.UI.rawValue

世界单位.addChild(ok按钮)

let ok = SKSpriteNode(imageNamed: "OK")

ok.position = CGPoint.zero

ok.zPosition = 图层.UI.rawValue

ok按钮.addChild(ok)

let 声音特效 = SKAction.sequence([

SKAction.waitForDuration(k动画延迟),

砰的音效,

SKAction.waitForDuration(k动画延迟),

砰的音效,

SKAction.waitForDuration(k动画延迟),

砰的音效,

SKAction.runBlock(切换到结束状态)

])

runAction(声音特效)

}

### 2.6设置得分标签

func 设置得分标签() {

得分标签 = SKLabelNode(fontNamed:k字体名字)

得分标签.fontColor = SKColor(red: 101.0/255.0, green: 71.0/255.0, blue: 73.0/255.0, alpha: 1.0)

得分标签.position = CGPoint(x:size.width/2,y:size.height - k顶部留白)

得分标签.verticalAlignmentMode = .Top

得分标签.text = "0"

得分标签.zPosition = 图层.UI.rawValue

世界单位.addChild(得分标签)

}

### 2.7设置、生成障碍物

func 创建障碍物(photo:String)->SKSpriteNode{

let 障碍物 = SKSpriteNode(imageNamed: photo)

障碍物.zPosition = 图层.障碍物.rawValue

障碍物.userData = NSMutableDictionary()

let offsetX = 障碍物.size.width \* 障碍物.anchorPoint.x

let offsetY = 障碍物.size.height \* 障碍物.anchorPoint.y

let path = CGPathCreateMutable()

CGPathMoveToPoint(path, nil, 4 - offsetX, 0 - offsetY)

CGPathAddLineToPoint(path, nil, 7 - offsetX, 307 - offsetY)

CGPathAddLineToPoint(path, nil, 47 - offsetX, 308 - offsetY)

CGPathAddLineToPoint(path, nil, 48 - offsetX, 1 - offsetY)

CGPathCloseSubpath(path)

障碍物.physicsBody = SKPhysicsBody(polygonFromPath: path)

障碍物.physicsBody?.categoryBitMask = 物理层.障碍物

障碍物.physicsBody?.collisionBitMask = 0

障碍物.physicsBody?.contactTestBitMask = 物理层.游戏角色

return 障碍物

}

func 生成障碍(){

let 底部障碍 = 创建障碍物("CactusBottom")

let 起始X坐标 = size.width + 底部障碍.size.width/2

let Y坐标最小值 = (游戏区域起始点 - 底部障碍.size.height/2) + 游戏区域的高度 \* k底部障碍最小乘数

let Y坐标最大值 = (游戏区域起始点 - 底部障碍.size.height/2) + 游戏区域的高度 \* k底部障碍最大乘数

底部障碍.position = CGPointMake(起始X坐标, CGFloat.random(min: Y坐标最小值,max: Y坐标最大值))

底部障碍.name = "底部障碍"

世界单位.addChild(底部障碍)

let 顶部障碍 = 创建障碍物("CactusTop")

顶部障碍.zRotation = CGFloat(180).degreesToRadians()

顶部障碍.position = CGPoint(x: 起始X坐标, y: 底部障碍.position.y + 底部障碍.size.height/2 + 顶部障碍.size.height/2 + 主角.size.height \* k缺口乘数)

顶部障碍.name = "顶部障碍"

世界单位.addChild(顶部障碍)

let X轴移动距离 = -(size.width + 底部障碍.size.width)

let 移动持续时间 = X轴移动距离/k地面移动速度

let 移动的动作队列 = SKAction.sequence([SKAction.moveByX(X轴移动距离,y :0, duration:NSTimeInterval(移动持续时间)),SKAction.removeFromParent()

])

顶部障碍.runAction(移动的动作队列)

底部障碍.runAction(移动的动作队列)

}

func 无限重生障碍(){

let 首次延迟 = SKAction.waitForDuration(k首次生成障碍延迟)

let 重生障碍 = SKAction.runBlock(生成障碍)

let 每次重生间隔 = SKAction.waitForDuration(k每次重生障碍延迟)

let 重生的动作队列 = SKAction.sequence([重生障碍, 每次重生间隔])

let 无限重生 = SKAction.repeatActionForever(重生的动作队列)

let 总的动作队列 = SKAction.sequence([首次延迟, 无限重生])

runAction(总的动作队列,withKey: "重生")

}

func 停止重生障碍(){

removeActionForKey("重生")

世界单位.enumerateChildNodesWithName("顶部障碍", usingBlock: { 匹配单位, \_ in

匹配单位.removeAllActions()

})

世界单位.enumerateChildNodesWithName("底部障碍", usingBlock: { 匹配单位, \_ in

匹配单位.removeAllActions()

})

}

### 2.8屏幕点击

override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {

guard let 点击 = touches.first else{

return

}

let 点击位置 = 点击.locationInNode(self)

switch 当前游戏状态{

case .主菜单:

if 点击位置.y > size.height \* 0.15{

切换到教程状态()

}

break

case .教程:

切换到游戏状态()

break

case .游戏:

主角飞一下()

break

case .跌落:

break

case .显示分数:

break

case .结束:

切换到新游戏()

break

}

}

## 2.9 更新

override func update(当前时间: CFTimeInterval) {

if 上一次更新时间 > 0 {

dt = 当前时间 - 上一次更新时间

} else {

dt = 0

}

上一次更新时间 = 当前时间

switch 当前游戏状态 {

case .主菜单:

break

case .教程:

break

case .游戏:

更新前景()

更新主角()

撞击障碍物检查()

撞击地面检查()

更新得分()

break

case .跌落:

更新主角()

撞击地面检查()

break

case .显示分数:

break

case .结束:

break

}

}

func 更新主角(){

let 加速度 = CGPoint(x: 0,y: k重力)

速度 = 速度 + 加速度 \* CGFloat(dt)

主角.position = 主角.position + 速度 \* CGFloat(dt)

//检测撞击地面时让小鸟停在地面上

if 主角.position.y - 主角.size.height/2 < 游戏区域起始点{

主角.position = CGPoint(x: 主角.position.x, y: 游戏区域起始点 + 主角.size.height/2)

}

}

func 更新前景(){

世界单位.enumerateChildNodesWithName("前景", usingBlock: {匹配单位, \_ in

if let 前景 = 匹配单位 as? SKSpriteNode{

let 地面移动速度 = CGPoint(x:self.k地面移动速度,y:0)

前景.position += 地面移动速度 \* CGFloat(self.dt)

if 前景.position.x < -前景.size.width{

前景.position += CGPoint(x:前景.size.width \* CGFloat(self.k前景地面数),y: 0)

}

}

})

}

func 更新得分(){

世界单位.enumerateChildNodesWithName("顶部障碍", usingBlock: { 匹配单位, \_ in

if let 障碍物 = 匹配单位 as? SKSpriteNode{

if let 已通过 = 障碍物.userData?["已通过"] as? NSNumber{

if 已通过.boolValue{

return

}

}

if self.主角.position.x > 障碍物.position.x + 障碍物.size.width/2{

self.当前分数 = self.当前分数 + 1

self.得分标签.text = "\(self.当前分数)"

self.runAction(self.得分的音效)

障碍物.userData?["已通过"] = NSNumber(bool:true)

}

}

})

}

## 2.10 撞击检查

func 撞击障碍物检查(){

if 撞击了障碍物 {

撞击了障碍物 = false

切换到跌落状态()

}

}

func 撞击地面检查(){

if 撞击了地面{

撞击了地面 = false

速度 = CGPoint.zero

主角.zRotation = CGFloat(-90).degreesToRadians()

主角.position = CGPoint(x:主角.position.x,y:游戏区域起始点 + 主角.size.width/2)

runAction(撞击地面的音效)

切换到显示分数状态()

}

}

## 2.11 游戏状态

func 切换到主菜单(){

当前游戏状态 = .主菜单

设置背景()

设置前景()

设置主角()

设置帽子()

设置主菜单()

}

func 切换到教程状态(){

当前游戏状态 = .教程

世界单位.enumerateChildNodesWithName("主菜单") { 匹配单位,\_ in

匹配单位.runAction(SKAction.sequence([

SKAction.fadeOutWithDuration(0.05),

SKAction.removeFromParent()

]))

}

设置得分标签()

设置教程()

}

func 切换到游戏状态(){

当前游戏状态 = .游戏

世界单位.enumerateChildNodesWithName("教程") { 匹配单位,\_ in

匹配单位.runAction(SKAction.sequence([

SKAction.fadeOutWithDuration(0.05),

SKAction.removeFromParent()

]))

}

无限重生障碍()

主角飞一下()

}

func 切换到跌落状态(){

当前游戏状态 = .跌落

runAction(SKAction.sequence([

摔倒的音效,

SKAction.waitForDuration(0.1),

下落的音效

]))

主角.removeAllActions()

停止重生障碍()

}

func 切换到显示分数状态(){

当前游戏状态 = .显示分数

主角.removeAllActions()

停止重生障碍()

设置记分板()

}

func 切换到新游戏(){

runAction(砰的音效)

let 新的游戏场景 = GameScene.init(size:size)

let 切换特效 = SKTransition.fadeWithColor(SKColor.blackColor(), duration: 0.05)

view?.presentScene(新的游戏场景,transition: 切换特效)

}

func 切换到结束状态(){

当前游戏状态 = .结束

}

## 2.12 分数

func 最高分() -> Int {

return NSUserDefaults.standardUserDefaults().integerForKey("最高分")

}

func 设置最高分(最高分:Int) {

NSUserDefaults.standardUserDefaults().setInteger(最高分, forKey: "最高分")

NSUserDefaults.standardUserDefaults().synchronize()

}

## 2.13 物理引擎

func didBeginContact(碰撞双方: SKPhysicsContact) {

let 被撞对象 = 碰撞双方.bodyA.categoryBitMask == 物理层.游戏角色 ? 碰撞双方.bodyB : 碰撞双方.bodyA

if 被撞对象.categoryBitMask == 物理层.地面{

撞击了地面 = true

}

if 被撞对象.categoryBitMask == 物理层.障碍物{

撞击了障碍物 = true

}

}

# 调试分析

## 调试中出现的问题有：

* 地面场景不循环显示

if 前景.position.x < -前景.size.width {

前景.position += CGPoint(x: 前景.size.width \* CGFloat(self.k前景地面数), y: 0)

}

* 主角静止不动

主角.position = 主角.position + 速度 \* CGFloat(dt)

* 撞击障碍物游戏不停止

if 撞击了障碍物 {

撞击了障碍物 = false

切换到跌落状态()

}

* 障碍物显示的位置不正确

底部障碍.position = CGPoint(x: 起始X坐标, y: CGFloat.random(min: Y坐标最小值, max: Y坐标最大值))

* 没有音效 音效文件位置错误
* 无法开始新的游戏

主角.removeAllActions()

case .结束:

切换到新游戏()

Break

# 第4章使用说明与执行结果

## 4.1 运行操作及结果

(1)开始界面



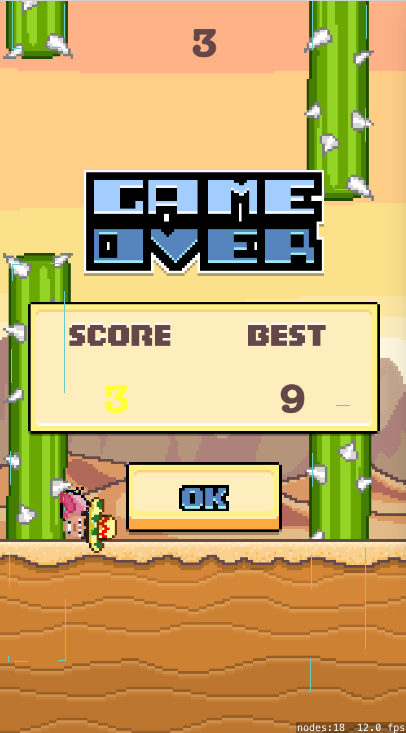
(2)教程界面



(3)游戏开始



(4)结束界面

****

## 4.2 课程设计小结

**在实验中发现——分析——总结的问题有：**对swift语言的学习还是不够，很多还需要自己下去学习。这几个星期，在老师的帮助下，利用百度和老师的指导还学会了很多其他的知识。最终在老师和同学的耐心帮助和指导下，成功完成了程序的编写，再次感谢大家！

**参考文献：**

**[1]** **https://www.jianshu.com/nb/2038131**

**[2]** SwiftV课堂 > Swift初级教程 > 「从零开始」做一个 FlappyBird 那样的游戏

**http://www.swiftv.cn/course/il0dj77w**