**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**：**利用swift3.0编写跑酷游戏**

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：20151104695

姓 名：孙武

指导教师：朝力萌

完成日期：2018.7

# 第1章 概要设计

## 1.1题目的内容与要求

**内容：**

**利用xcode 里game的项目组件，制作出一款自己喜欢的跑酷游戏。此款跑酷游戏用swift3.0语言编写。**

**功能：**

1. **能够将游戏人物显示到场景之中。**
2. **能够将游戏人物显示到场景之中并让之跑动。**
3. **能够使游戏人物通过点击鼠标完成跳跃。**
4. **能够使游戏人物通过点击两次鼠标完成二连跳。**
5. **能够使游戏人物在落地之后进行翻滚 。**
6. **能够使游戏人物通过吃苹果增加分数。**
7. **能够增加背景音乐。**

## 1.2总体结构

功夫Panda

背景图片移动

吃掉苹果增加分数

熊猫翻滚

熊猫二段跳跃

熊猫跳跃

熊猫跑动

显示熊猫在场景中

踩踏平台移动

背景音乐

# 第2章 详细设计

### 2.1 熊猫跑动功能

//跑

for var i = 1; i<=runAtlas.textureNames.count; i += 1 {

let tempName = String(format: "panda\_run\_%.2d", i)

let runTexture = runAtlas.textureNamed(tempName)

runFrames.append(runTexture)

}

### 2.2 熊猫跳跃功能

//跳

for var i = 1; i<=jumpAtlas.textureNames.count; i += 1 {

let tempName = String(format: "panda\_jump\_%.2d", i)

let jumpTexture = jumpAtlas.textureNamed(tempName)

jumpFrames.append(jumpTexture)

}

### 2.3 熊猫二段跳跃功能

// 跳的时候的点缀效果

for var i=1 ; i <= jumpEffectAtlas.textureNames.count ; i += 1 {

let tempName = String(format: "jump\_effect\_%.2d", i)

let effectexture = jumpEffectAtlas.textureNamed(tempName)

jumpEffectFrames.append(effectexture)

}

### 2.4熊猫翻滚功能

//滚

for var i = 1; i<=rollAtlas.textureNames.count; i += 1 {

let tempName = String(format: "panda\_roll\_%.2d", i)

let rollTexture = rollAtlas.textureNamed(tempName)

rollFrames.append(rollTexture)

}

### 2.5踩踏平台移动功能

import SpriteKit

class PlatformFactory:SKNode {

let textureLeft = SKTexture(imageNamed: "platform\_l")

let textureMid = SKTexture(imageNamed: "platform\_m")

let textureRight = SKTexture(imageNamed: "platform\_r")

var platforms = [Platform]()

var screenWdith:CGFloat = 0.0

var delegate:ProtocolMainscreen?

func createPlatformRandom(){

//随即平台长度

let midNum = arc4random()%4 + 1

//随机间隔

let gap:CGFloat = CGFloat(arc4random()%8 + 1)

//x坐标

let x = self.screenWdith + CGFloat(midNum\*50) + gap + 100

//y坐标

let y = CGFloat(arc4random()%200 + 200)

createPlatform(midNum, x: x, y: y)

}

func createPlatform(midNum:UInt32,x:CGFloat,y:CGFloat){

let platform = Platform()

let platform\_left = SKSpriteNode(texture: textureLeft)

platform\_left.anchorPoint = CGPointMake(0, 0.9)

let platform\_right = SKSpriteNode(texture: textureRight)

platform\_right.anchorPoint = CGPointMake(0, 0.9)

var arrPlatform = [SKSpriteNode]()

arrPlatform.append(platform\_left)

platform.position = CGPointMake(x, y)

for \_ in 1...midNum {

let platform\_mid = SKSpriteNode(texture: textureMid)

platform\_mid.anchorPoint = CGPointMake(0, 0.9)

arrPlatform.append(platform\_mid)

}

arrPlatform.append(platform\_right)

platform.onCreate(arrPlatform)

platform.name = "platform"

self.addChild(platform)

platforms.append(platform)

self.delegate?.onGetData(platform.width + x - screenWdith,theY:y)

}

func move(speed:CGFloat){

for p in platforms {

let position = p.position

p.position = CGPointMake(position.x - speed, position.y)

}

if platforms[0].position.x < -platforms[0].width{

platforms[0].removeFromParent()

platforms.removeAtIndex(0)

}

}

//清除所有的Nod

func reset(){

self.removeAllChildren()

platforms.removeAll(keepCapacity: false)

}

}

### 2.6背景图片移动功能

//场景的动态背景图类

import SpriteKit

class Background:SKNode {

//近处的背景

var arrBG = [SKSpriteNode]()

//远处的背景

var arrFar = [SKSpriteNode]()

override init() {

super.init()

let farTexture = SKTexture(imageNamed: "background\_f1")

let farBg0 = SKSpriteNode(texture: farTexture)

farBg0.position.y = 150

farBg0.zPosition = 9

farBg0.anchorPoint = CGPointMake(0, 0)

let farBg1 = SKSpriteNode(texture: farTexture)

farBg1.position.y = 150

farBg1.zPosition = 9

farBg1.anchorPoint = CGPointMake(0, 0)

farBg1.position.x = farBg1.frame.width

let farBg2 = SKSpriteNode(texture: farTexture)

farBg2.position.y = 150

farBg2.zPosition = 9

farBg2.anchorPoint = CGPointMake(0, 0)

farBg2.position.x = farBg2.frame.width\*2

self.addChild(farBg0)

self.addChild(farBg1)

self.addChild(farBg2)

arrFar.append(farBg0)

arrFar.append(farBg1)

arrFar.append(farBg2)

let texture = SKTexture(imageNamed: "background\_f0")

let bg0 = SKSpriteNode(texture: texture)

bg0.anchorPoint = CGPointMake(0, 0)

bg0.position.y = 70

bg0.zPosition = 10

let bg1 = SKSpriteNode(texture: texture)

bg1.anchorPoint = CGPointMake(0, 0)

bg1.position.y = 70

bg1.zPosition = 10

bg1.position.x = bg0.frame.size.width

self.addChild(bg0)

self.addChild(bg1)

arrBG.append(bg0)

arrBG.append(bg1)

}

required init?(coder aDecoder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

func move(speed:CGFloat){

//近景

for bg in arrBG {

bg.position.x -= speed

}

if arrBG[0].position.x + arrBG[0].frame.size.width < speed {

arrBG[0].position.x = 0

arrBG[1].position.x = arrBG[0].frame.size.width

}

//远景

for far in arrFar {

far.position.x -= speed/4

}

if arrFar[0].position.x + arrFar[0].frame.size.width < speed/4 {

arrFar[0].position.x = 0

arrFar[1].position.x = arrFar[0].frame.size.width

arrFar[2].position.x = arrFar[0].frame.size.width \* 2

}

}

}

### 2.7吃掉苹果增加分数功能

//

// AppleFactory.swift

// panda

//

// Created by s20151104695 on 2018/7/4.

// Copyright © 2018年 s20151104695. All rights reserved.

//

import SpriteKit

class AppleFactory:SKNode{

let appleTexture = SKTexture(imageNamed: "apple")

var sceneWidth:CGFloat = 0.0

var arrApple = [SKSpriteNode]()

var timer = NSTimer()

var theY:CGFloat = 0.0

override init() {

super.init()

}

required init?(coder aDecoder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

func onInit(width:CGFloat, y:CGFloat) {

self.sceneWidth = width

self.theY = y

timer = NSTimer.scheduledTimerWithTimeInterval( 0.2, target: self, selector: #selector(AppleFactory.createApple), userInfo: nil, repeats: true)

}

func createApple(){

let random = arc4random() % 10

if random > 8 {

let apple = SKSpriteNode(texture: appleTexture)

apple.physicsBody = SKPhysicsBody(rectangleOfSize: apple.size)

apple.physicsBody!.restitution = 0

apple.physicsBody!.categoryBitMask = BitMaskType.apple

apple.physicsBody!.dynamic = false

apple.anchorPoint = CGPointMake(0, 0)

apple.zPosition = 40

apple.position = CGPointMake(sceneWidth+apple.frame.width , theY + 150)

arrApple.append(apple)

self.addChild(apple)

}

}

func move(speed:CGFloat){

for apple in arrApple {

apple.position.x -= speed

}

if arrApple.count > 0 && arrApple[0].position.x < -20{

arrApple[0].removeFromParent()

arrApple.removeAtIndex(0)

}

}

func reSet(){

self.removeAllChildren()

arrApple.removeAll(keepCapacity: false)

}

}

### 2.8 背景音乐功能

//

// SoundManager.swift

// panda

//

// Created by s20151104695 on 2018/7/4.

// Copyright © 2018年 s20151104695. All rights reserved.

//

import SpriteKit

import AVFoundation

class SoundManager :SKNode{

var bgMusicPlayer = AVAudioPlayer()

let jumpAct = SKAction.playSoundFileNamed("jump\_from\_platform.mp3", waitForCompletion: false)

let loseAct = SKAction.playSoundFileNamed("lose.mp3", waitForCompletion: false)

let rollAct = SKAction.playSoundFileNamed("hit\_platform.mp3", waitForCompletion: false)

let eatAct = SKAction.playSoundFileNamed("hit.mp3", waitForCompletion: false)

func playBackgroundMusic(){

let bgMusicURL:NSURL = NSBundle.mainBundle().URLForResource("apple", withExtension: "mp3")!

bgMusicPlayer = try! AVAudioPlayer(contentsOfURL: bgMusicURL)

bgMusicPlayer.numberOfLoops = -1

bgMusicPlayer.prepareToPlay()

bgMusicPlayer.play()

}

func stopBackgroundMusic(){

if bgMusicPlayer.playing{

bgMusicPlayer.stop()

}

}

func playDead(){

self.runAction(loseAct)

}

func playJump(){

self.runAction(jumpAct)

}

func playRoll(){

self.runAction(rollAct)

}

func playEat(){

self.runAction(eatAct)

}

# 调试分析

## 调试中出现的问题有：

1、数字超出数组容量

2、swift2.0与swift3.0语法不同，需要改错。

3、熊猫不能移动。

4、熊猫不能跳跃。

5、熊猫不能滚动。

6、熊猫一直在天上移动，不执行踩踏平台动作。

7、熊猫可以一直跳跃，出现bug。

8、平台不能移动。

9、背景不能移动。

10、踩踏平台重复显示，，屏幕上全是平台，导致内存过多没有及时释放内存，导致内存消耗过大，严重浪费计算机资源。

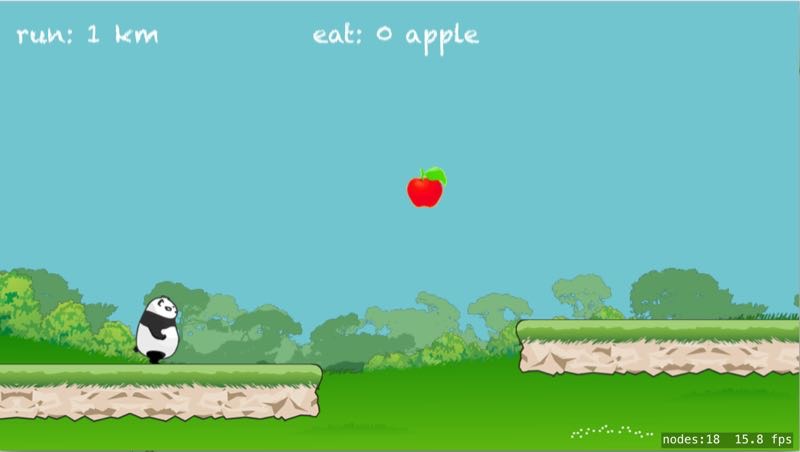
11、不显示苹果图标。

12、背景音乐在游戏开始时不播放。

# 第4章 使用说明与执行结果

一、**运行操作及结果**

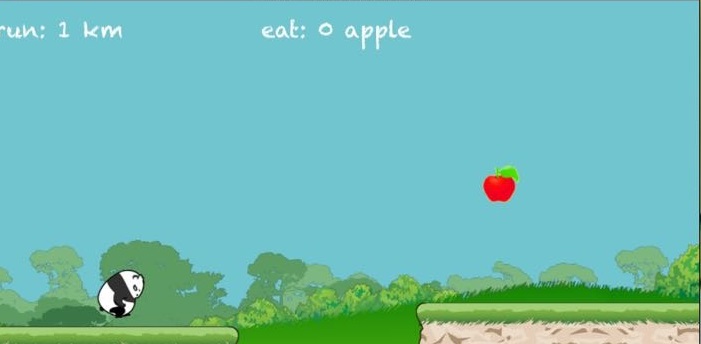
跑，游戏开始运行

****

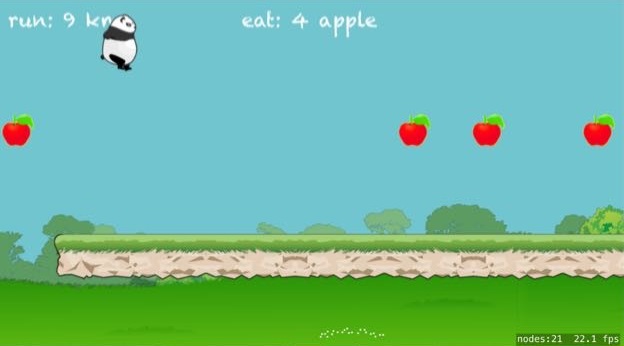
游戏结束画面

****

熊猫滚动

****

熊猫跳跃

****

**二．课程设计小结**

**在实验中发现——分析——总结的问题有：**

**1、合理利用计算机内存**

**2、合理优化代码，提高程序运行速度**

**3、学会swift3.0 与swift2.0 语法的区别。**

**参考文献：**

**[1] http://www.swiftv.cn/course/hxbazkq4**

**[2]** [SwiftV课堂](http://www.swiftv.cn/) > [Swift进阶教程](http://www.swiftv.cn/course/explore/hw4ytn7s) > 【老镇出品】Swift游戏实战-跑酷熊猫