

VOLUME I

# Desenvolvimento iOS com Swift para Jogos Digitais

Agesandro Scarpioni

# Xcode e SpriteKit

---

Xcode é o SW de desenvolvimento de aplicativos iOS que irá rodar nos iPhones, iPads, Apple TV e Apple Watch, SpriteKit é a engine da Apple utilizada para desenvolvimento de jogos, sua função é ajudar o desenvolvedor a aumentar sua produtividade, possui funções que facilitam a inserção de física, criação de partículas, animação, som e muitas outras coisas.



# Iniciando seu projeto

NESTE CAPÍTULO VOCÊ IRÁ:

1. Criar seu projeto.
2. Escolher o template mais apropriado.
3. Nomear e salvar seu projeto.
4. Conhecer o ambiente do Xcode.
5. SpriteKit

O primeiro passo após instalar o Xcode é criar seu projeto, ao abrir o programa escolha a segunda opção: **Create a new xcode project**.



## Welcome to Xcode

Version 7.2 (7C68)



**Get started with a playground**  
Explore new ideas quickly and easily.



**Create a new Xcode project**  
Start building a new iPhone, iPad or Mac application.



**Check out an existing project**  
Start working on something from an SCM repository.

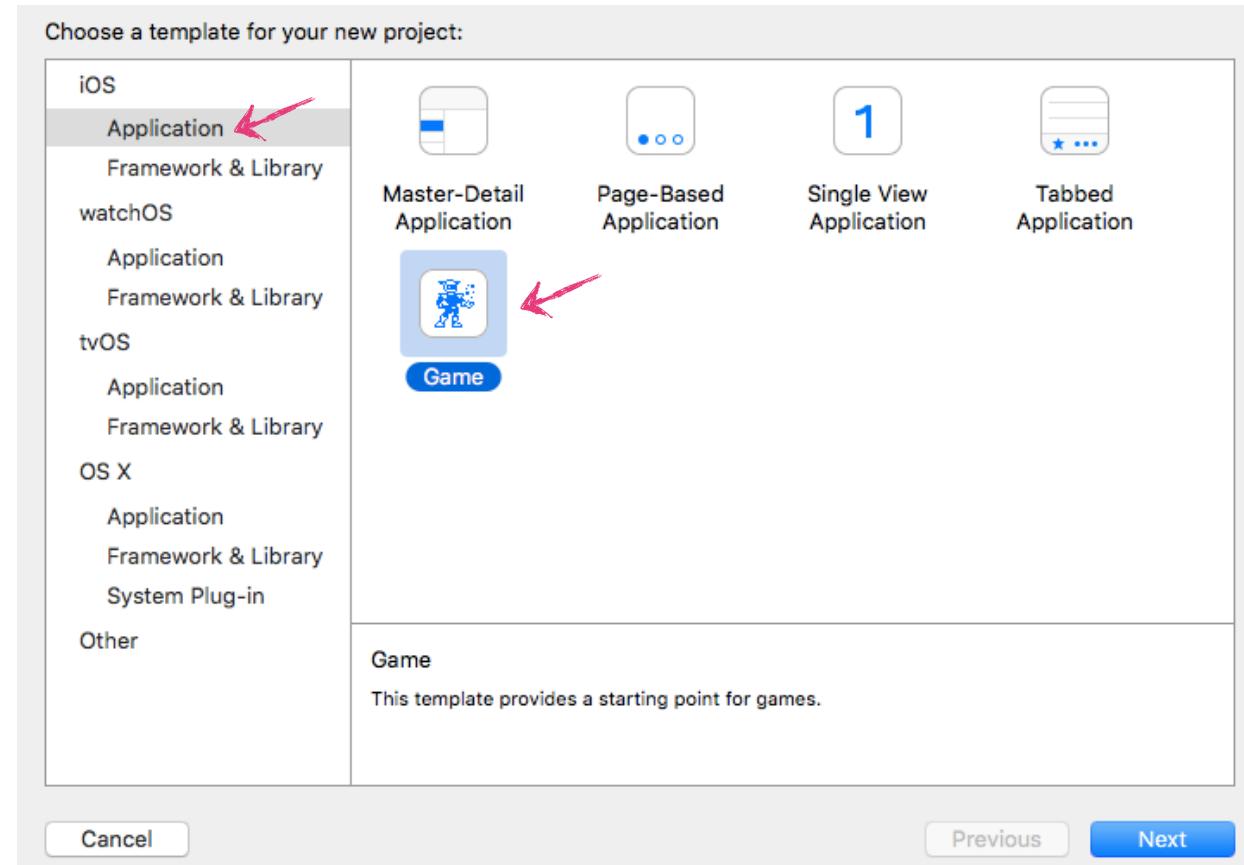
Show this window when Xcode launches

## Seção 2

# Selecionando o Template

### Selecionando o template para Jogos

Dentro do item Application de cada categoria (iOS, watchOS, tvOS e OS X), selecione a categoria **iOS -> Application**, neste item existem diversos templates, Master - Detail Application cria projetos XDDD FDFDF XX, já o Page - Based Application é para XXDFD FDFDFD FDFDF X, Single View Application é para XDFDF FDFD FDF XX, Tabbed Application para XXDFDFD X, vamos utilizar o **Game** que será o local onde faremos o uso do SpriteKit.

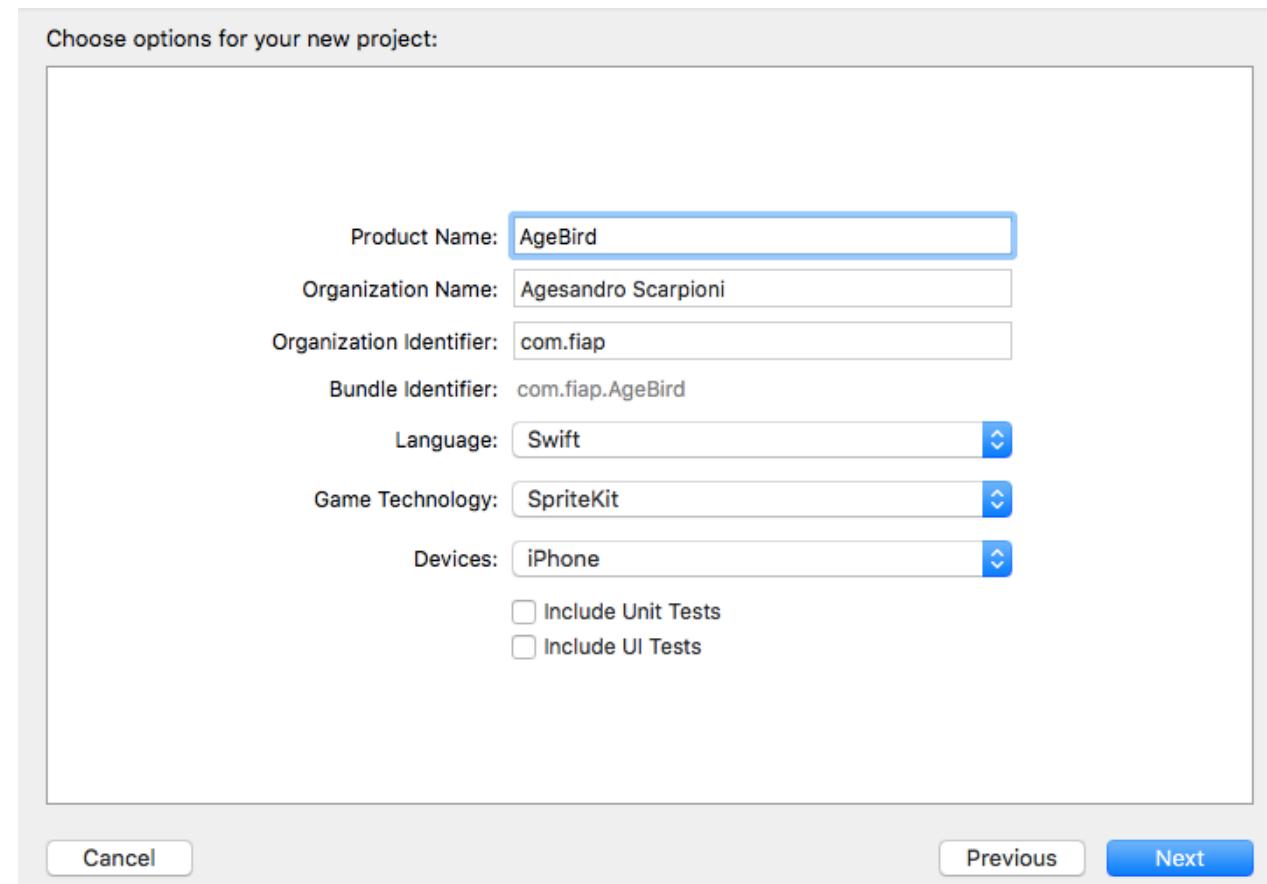


## Seção 3

# Nomeando o Projeto

### Selecionando o template para Jogos

Dentro do item Application de cada categoria (iOS, watchOS, tvOS e OS X), selecione a categoria **iOS -> Application**, neste item existem diversos templates, Master - Detail Application cria projetos XDDDFDFD XX, já o Page - Based Application é para XXDFDFD FDFDFD FDFDFD X, Single View Application é para XDFDFD FDFDFD XX, Tabbed Application para XXDFDFD X, vamos utilizar o **Game** que será o local onde faremos o uso do SpriteKit.



# Elementos Gráficos do Jogo

---

Nesse capítulo iremos aprender a inserir imagens no projeto e técnicas para posicioná-las e animá-las.



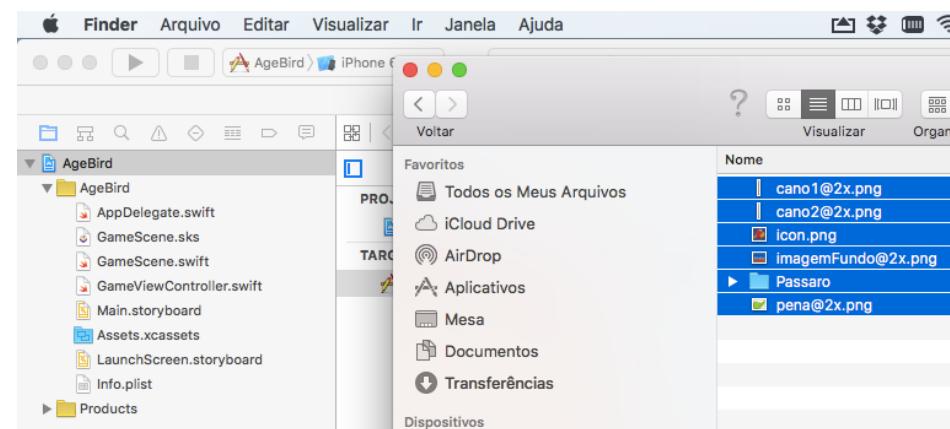
## Seção 1

# Importando imagem para o projeto

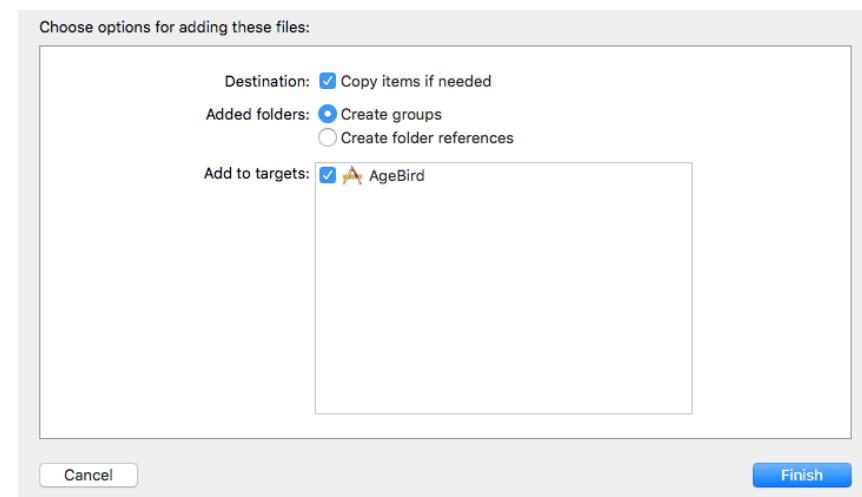
NESTE CAPÍTULO VOCÊ IRÁ:

1. Importar as imagens para o projeto
2. Adicionar uma imagem
3. Animar uma imagem
4. Alterar a escada da imagem
5. Alterar a escala da imagem
6. Adicionar a imagem de fundo a cena
7. Mover a imagem de fundo
8. Repositionar a imagem de fundo
9. Alterando o BackGroundColor do Fundo da Cena
10. Definindo um vão entre os canos
11. Criando os canos e definindo o tamanho do vão
- 12.

Vamos preparar os recursos de imagem copiando da pasta do Finder as imagens para o projeto.



Após arrastar a pasta clique em **Copy Items if needed** e assinale a opção **Create groups**, isso garante que as imagens sejam copiadas para o projeto e não apenas referenciadas.



# Adicionar e Animar Imagem

## Adicionar uma imagem via código

É possível adicionar uma imagem à cena pelo código, para isso você precisa criar um objeto do tipo SKSpriteNode ou SKNode, passar o nome da imagem e adiciona-lo a cena pelo método addChild.

Outros métodos podem ser invocados para o objeto, como por exemplo setScale para mudar a escala da imagem, alpha para mudar o nível de visibilidade ou até zPosition para definir a profundidade que essa imagem deve ser apresentada com relação a outras imagens.

- Na linha 13 foi criado um objeto do tipo SKSpriteNode.
- Dentro do método didMoveToView na linha 18 foi passado para o objeto o nome da imagem.
- Para posicionar a figura foi definida a posição central da tela dividindo a largura por 2 e a altura por 2, como é mostrado na linha 19, em seguida atribuímos esse valor para o método position do objeto.
- Na linha 21 foi adicionado um node filho na sprite.

```
1 //  
2 // GameScene.swift  
3 // AgeBird  
4 //  
5 // Created by agesandro scarpioni on 25/01/16.  
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import SpriteKit  
10  
11 class GameScene: SKScene {  
12  
13     var passaro = SKSpriteNode()  
14  
15  
16     override func didMoveToView(view: SKView) {  
17  
18         passaro = SKSpriteNode(imageNamed: "passaro1")  
19         passaro.position = CGPointMake(self.size.width/2 , self.size.height/2)  
20         self.addChild(passaro)  
21     }  
22  
23 }  
24  
25 }
```

## Animar uma imagem

Se a imagem possuir vários frames, podemos adicioná-las dentro de um array de texturas (21 a 23) e animar esse array definindo um tempo para cada frame ao invocar o método `animateWithTextures` (25) de `SKAction`, depois basta repetir essa ação pelo `repeatActionForever` também de `SKAction` que irá receber a animação (26), quando tudo estiver montado precisamos executar no objeto `SpriteNode` o método `runAction` e passar para esse método a constante que recebeu a animação (28).



```
8
9 import SpriteKit
10
11 class GameScene: SKScene {
12
13     var passaro = SKSpriteNode()
14     var arrayPassaros:[SKTexture]=[] ↗
15
16     override func didMoveToView(view: SKView) {
17
18         passaro = SKSpriteNode(imageNamed: "passaro1")
19         passaro.position = CGPointMake(self.size.width/2 , self.size.height/2)
20
21         for (var i=1; i<=11; i++){
22             arrayPassaros.append((SKTexture(imageNamed: "passaro\((i)")))) ↗
23         }
24     }
25     let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.1)
26     let voar = SKAction.repeatActionForever(animacao)
27
28     passaro.runAction(voar)
29
30     self.addChild(passaro)
31
32 }
33 }
```

## Alterar a escala da imagem

Como citado anteriormente podemos alterar o tamanho da imagem pelo método `setScale`, nesse caso optamos por deixar a imagem com 80% do tamanho original (29), também recuamos a imagem um pouco mais à esquerda dividindo o width por 2.5 ao invés de 2.

```
9 import SpriteKit
10
11 class GameScene: SKScene {
12
13     var passaro = SKSpriteNode()
14     var arrayPassaros:[SKTexture]=[]
15
16     override func didMoveToView(view: SKView) {
17
18         passaro = SKSpriteNode(imageNamed: "passaro1") ←
19         passaro.position = CGPointMake(self.size.width/2.5 , self.size.height/2)
20
21         for (var i=1; i<=11; i++){
22             arrayPassaros.append((SKTexture(imageNamed: "passaro\((i)")))
23         }
24
25         let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.1)
26         let voar = SKAction.repeatActionForever(animacao)
27
28         passaro.runAction(voar)
29         passaro.setScale(0.8)
30         self.addChild(passaro)
31
32     }
33 }
```

## Adicionar uma imagem de fundo na cena

Vamos criar um objeto vazio para receber o fundo que seja do tipo SKSpriteNode (16), desse ponto em diante vamos adicionar o fundo de uma forma diferente do que foi feito com o passaro, nesse caso, iremos criar variáveis para receber a textura (17) e a escala (18), depois aplicamos ao objeto a textura (31), a posição (34) e a escala (35).

Também foi feito alguns testes com o tempo de animação de cada frame (28) e a escala (37).

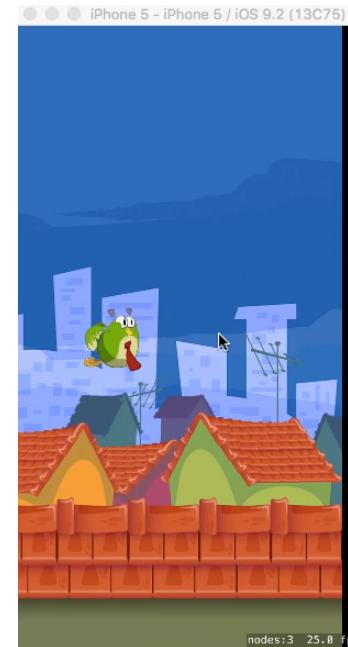
```
1 //  
2 // GameScene.swift  
3 // AgeBird  
4 //  
5 // Created by agesandro scarpioni on 25/01/16.  
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import SpriteKit  
10  
11 class GameScene: SKScene {  
12     var bird = SKSpriteNode()  
13     var arrayPassaros:[SKTexture]=[]  
14  
15     var fundo = SKSpriteNode()  
16     var texturaFundo = SKTexture(imageNamed: "imagemFundo")  
17     var escalaFundo = CGFloat(1.35)  
18  
19     override func didMoveToView(view: SKView) {  
20         bird = SKSpriteNode(imageNamed: "passaro1")  
21         bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)  
22  
23         for (var i=1; i<=12; i++){  
24             arrayPassaros.append((SKTexture(imageNamed: "passaro\"+i+"")))  
25         }  
26  
27         let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.05)  
28         let voar = SKAction.repeatActionForever(animacao)  
29         bird.runAction(voar)  
30         bird.setScale(0.85)  
31  
32         fundo = SKSpriteNode(texture: texturaFundo)  
33         fundo.position = CGPointMake(self.size.width/2, self.size.height/2)  
34         fundo.setScale(escalaFundo)  
35  
36         self.addChild(fundo)  
37  
38         self.addChild(bird)  
39  
40     }  
41  
42 }  
43  
44 }
```

```
40  
41 let moveFundo = SKAction.moveByX(-texturaFundo.size().width , y: 0, duration: 9)  
42 fundo.runAction(moveFundo)  
43
```

## Mover uma imagem de fundo

Existe um método de SKAction chamado moveByX , esse método movimenta um objeto pelo eixo x, esse método precisa de 3 valores, x, y, e o tempo que a imagem irá percorrer de uma ponta a outra.

Se deixarmos o eixo y zerado o deslocamento se dará em linha reta, e colocarmos o valor de x negativo como a largura da imagem o fundo irá se deslocar para a esquerda, se o valor de x for positivo o deslocamento ocorrerá pela direita.



*A imagem irá piscar durante a passagem do fundo porque é necessário informar que o pássaro estará em um plano a frente do fundo pelo método zPosition.*

## Reposicionar a imagem de fundo

Para reposicionar a imagem de fundo toda vez que sai da tela, vamos criar 2 constantes (repositoriarFundo e moveFundoSempre), a primeira servirá para colocar a imagem novamente na tela (38), a segunda serve para executar uma sequencia que é mover e reposicionar (39).

Troque o parâmetro do runAction para moveFundoSempre (41).

```
20     override func didMoveToView(view: SKView) {
21         bird = SKSpriteNode(imageNamed: "passaro1")
22         bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
23
24         for (var i=1; i<=12; i++){
25             arrayPassaros.append((SKTexture(imageNamed: "passaro\(" + i + ")")))
26         }
27
28         let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.05)
29         let voar = SKAction.repeatActionForever(animacao)
30         bird.runAction(voar)
31         bird.setScale(0.85)
32
33         fundo = SKSpriteNode(texture: texturaFundo)
34         fundo.position = CGPointMake(self.size.width/2, self.size.height/2)
35         fundo.setScale(escalaFundo)
36
37         let moveFundo = SKAction.moveByX(-texturaFundo.size().width * escalaFundo, y: 0, duration: 9)
38         let reposicionaFundo = SKAction.moveByX(texturaFundo.size().width * escalaFundo, y:0, duration: 0)
39         let moveFundoSempre = SKAction.repeatActionForever(SKAction.sequence([moveFundo, reposicionaFundo]))
40
41         fundo.runAction(moveFundoSempre)
42
43         self.addChild(fundo)
44
45         self.addChild(bird)
46
47
48
49     }
50 }
```



## Sincronizar a imagem do fundo

Será necessário um comando FOR que repita 3x para que a imagem de fundo seja reposicionada (41 a 49), dessa forma a imagem passando por trás do pássaro ficará contínua.

Vamos suavizar a imagem de fundo alterando o alpha do fundo para 0.75 (46).

```
20 override func didMoveToView(view: SKView) {
21     bird = SKSpriteNode(imageNamed: "passaro1")
22     bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
23
24     for (var i=1; i<=12; i++){
25         arrayPassaros.append(SKTexture(imageNamed: "passaro\" + i + "\""))
26     }
27
28     let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.05)
29     let voar = SKAction.repeatActionForever(animacao)
30     bird.runAction(voar)
31     bird.setScale(0.85)
32
33     fundo = SKSpriteNode(texture: texturaFundo)
34     fundo.position = CGPointMake(self.size.width/2, self.size.height/2)
35     fundo.setScale(escalaFundo)
36
37     let moveFundo = SKAction.moveByX(-texturaFundo.size().width * escalaFundo, y: 0, duration: 9)
38     let reposicionaFundo = SKAction.moveByX(texturaFundo.size().width * escalaFundo, y:0, duration: 0)
39     let moveFundoSempre = SKAction.repeatActionForever(SKAction.sequence([moveFundo, reposicionaFundo]))
40
41     for var i:CGFloat=0; i<3; i++
42     {
43         fundo = SKSpriteNode(texture: texturaFundo)
44         fundo.setScale(escalaFundo)
45         fundo.position = CGPointMake(x: texturaFundo.size().width/2 + texturaFundo.size().width * i *
46             escalaFundo, y: self.frame.height/2)
47         fundo.alpha=0.75
48         fundo.runAction(moveFundoSempre)
49         self.addChild(fundo)
50
51     }
52
53     self.addChild(bird)|
```



## Alterar o BackGroundColor do fundo da cena

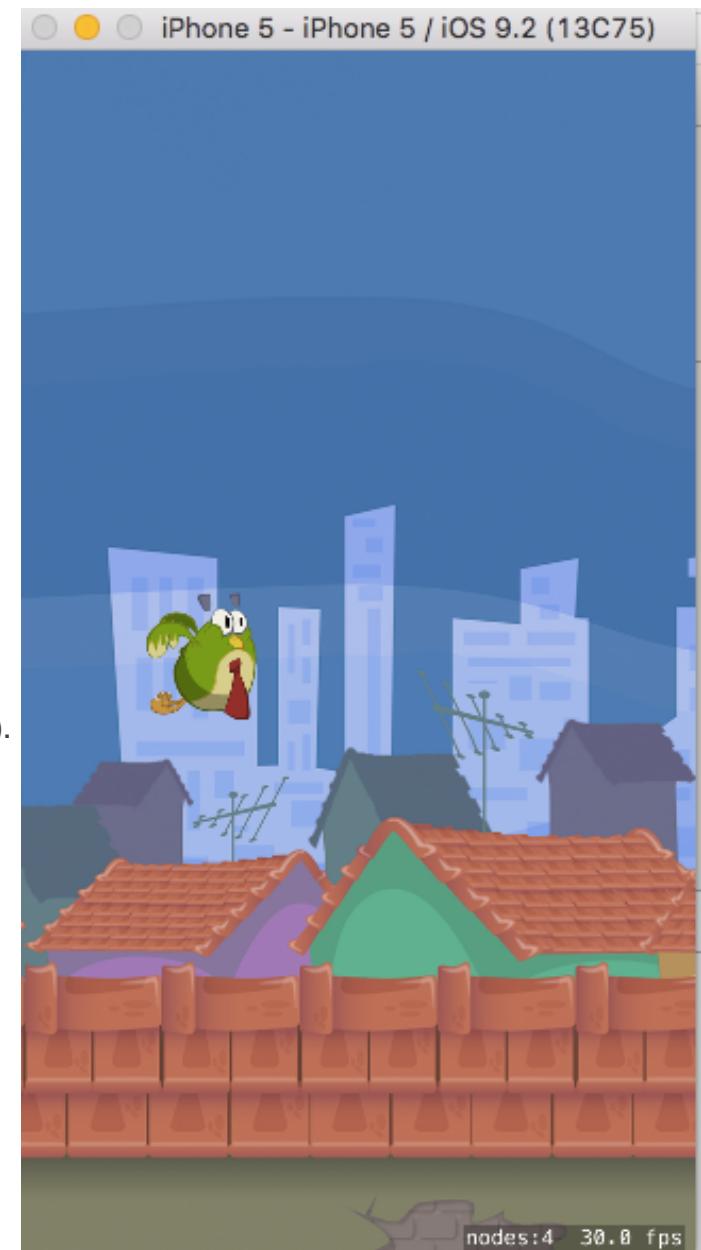
Observe pela figura ao lado que o fundo com 0.75 de alpha ficou levemente cinza, isso se deu porque a cena iniciou com um fundo cinza, vamos alterar a cor de fundo incluindo como primeira linha da função didMoveToView um backgroundColor setado em preto (24).

```
21
22     override func didMoveToView(view: SKView) {
23
24     →     self.backgroundColor = UIColor.blackColor()
25
26     bird = SKSpriteNode(imageNamed: "passaro1")
27     bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
28
```

## Definir um vão entre os canos

Vamos criar os canos e um vão entre os canos para o pássaro passar, primeiro iremos declarar uma variável que representará o espaço entre o cano. Veja a imagem abaixo (20).

```
1 // GameScene.swift
2 // AgeBird
3 //
4 //
5 // Created by agesandro scarpioni on 25/01/16.
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import SpriteKit
10
11 class GameScene: SKScene {
12
13     var bird = SKSpriteNode()
14     var arrayPassaros:[SKTexture]=[]
15
16     var fundo = SKSpriteNode()
17     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
18     var escalaFundo = CGFloat(1.35)
19
20     var alturaVao = CGFloat(50) ←
```



## Criar os canos na cena e definir o tamanho do vão

Uma altura de 2.5x o tamanho do pássaro foi definida (57), foi criado um timer para que a cada 3 segundos uma função “criarCano”, seja chamada (60).

```
55
56     self.addChild(bird)
57     alturaVao = bird.size.height * 2.5
58
59     //Criamos um timer, não se usa mais (var timer) quando apenas declaramos e não utilizamos a variável|
60     _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false) iPhone 6 - iPhone 6 / iOS 9.2 (13C75)
61
62 }
63
64 func criarCano()
65 {
66     let numeroRandom = arc4random() % UInt32(self.size.height / 3)
67     let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4
68
69     let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))
70     let apagaCano = SKAction.removeFromParent()
71     let sequenciaCano = SKAction.sequence([moveCano, apagaCano])
72
73     let imagemCanoBarra = SKTexture(imageNamed: "cano1")
74     let cano1 = SKSpriteNode(texture: imagemCanoBarra)
75     cano1.position = CGPointMake(x: self.size.width, y: alturaRandom)
76     cano1.runAction(sequenciaCano)
77
78     let imagemCanoTopo = SKTexture(imageNamed: "cano2")
79     let cano2 = SKSpriteNode(texture: imagemCanoTopo)
80     cano2.position = CGPointMake(x: self.size.width, y:(cano1.position.y + cano2.size.height + alturaVao))
81     cano2.runAction(sequenciaCano)
82
83     self.addChild(cano1)
84     self.addChild(cano2)
85
86     _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false)
87 }
88
89 }
90
```



The screenshot shows the Flappy Bird game interface. A single bird is visible in the center. Two pipes (barrels) are present: one pipe is positioned higher than the bird, and another is lower, creating a gap for the bird to pass through. The background features a dark blue sky with city buildings at the bottom. The bottom of the screen shows green ground and some foliage. The status bar indicates "nodes:6 17.6 fps".

No código acima a função “criarCano” gera um numero randômico para definir a altura do cano inferior “barra” (66 e 67), uma sequência idêntica ao movimento do fundo com 3 constantes (moveCano, apagaCano e sequênciaCano) irão ajudar no deslocamento e reaparecimento dos canos (69 a 71), o cano da barra é desenhado e posicionado (73 a 76) e acontece o mesmo com o cano do topo apenas ajustando a posição com a diferença do cano da barra e o vão (78 a 81), os dois canos são adicionados na cena (83 e 84) e o timer é chamado novamente após 3 segundos gerando uma continuidade nas aparições dos objetos (86).

## Seção 3

# Física

### Adicionar física ao pássaro

Foi definido que o pássaro terá um corpo circular (57) com um raio do tamanho da altura dividido por 2, o true em dynamic aciona a física.

```
52     fundo.runAction(moveFundoSempre)
53     self.addChild(fundo)
54 }
55
56
57 bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
58 bird.physicsBody?.dynamic = true
59
60 self.addChild(bird)
61 alturaVao = bird.size.height * 2.5
62
63 //Criamos um timer, não se usa mais (var timer) quando apenas declaramos
64 _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: #selector(timer))
65
66
67 }
68
69 func criarCano()
70 {
71     let numeroRandom = arc4random() % UInt32(self.size.height / 3)
```

## Adicionar um chão para aparar o pássaro

Foi criado um SKNode que representa o chão da cena (66), a área desse chão será o comprimento da cena com altura 1 (68) e terá uma física retangular, essa superfície terá uma dinâmica setada em false para o pássaro poder ser aparado, ao final o chão será adicionado à cena.

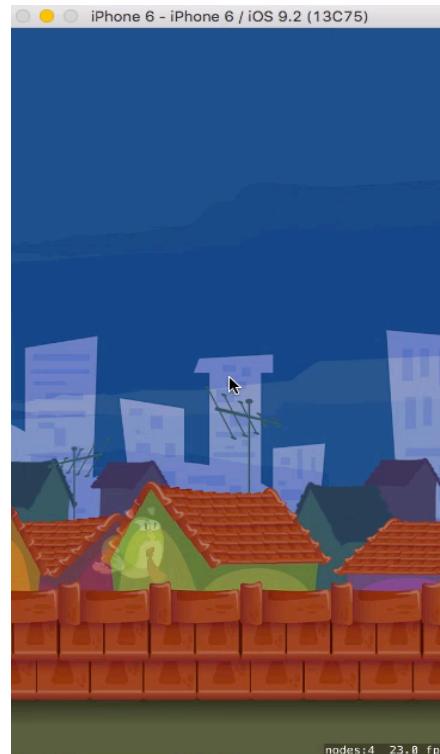
```
52         fundo.runAction(moveFundoSempre)
53         self.addChild(fundo)
54     }
55
56
57     bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
58     bird.physicsBody?.dynamic = true
59
60     self.addChild(bird)
61     alturaVao = bird.size.height * 2.5
62
63     //Criamos um timer, não se usa mais (var timer) quando apenas declaramos e não utilizamos a variável
64     _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false)
65
66     let chao = SKNode()
67     chao.position = CGPointMake(0, 0)
68     chao.physicsBody = SKPhysicsBody(rectangleOfSize: CGSizeMake(self.frame.size.width, 1))
69     chao.physicsBody?.dynamic = false
70     self.addChild(chao)
71
72 }
73
74 func criarCano()
75 {
```



## Controlar o pássaro na vertical

Abaixo da função CriarCano, crie a função touchesBegan, essa função é ativada toda vez que a tela for tocada, dentro da função vamos ajustar a velocidade do pássaro em 0 e aplicar um impulso 0 na horizontal e 150 na vertical.

```
98
99     override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
100    {
101        bird.physicsBody?.velocity = CGVectorMake(0, 0)
102        bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
103    }
104}
```



## Empinar o pássaro durante o impulso

Para empinar o pássaro é necessário utilizar a função update que é atualizada a cada novo frame, dentro dessa função (107), será capturado em uma constante chamada **num** o velocidade da física do pássaro, esse número é passado para uma função chamada chamada empinada (111) que irá controlar a rotação conforme a velocidade da física do pássaro. A função empinada foi criada para evitar que o pássaro gire muito para frente ou muito para trás, dessa forma com os valores enviados para a função controlamos essa rotação. Alterando esses valores você pode fazer com que a rotação aumente ou diminua.

```
105
106    override func update(currentTime: CFTimeInterval) {
107        let num = bird.physicsBody!.velocity.dy as CGFloat
108        bird.zRotation = self.empinada( -1, max: 0.5, valor:num * 0.001)
109    }
110
111    func empinada(min: CGFloat, max: CGFloat, valor: CGFloat) -> CGFloat {
112        if( valor > max )
113        {
114            return max
115        } else if( valor < min )
116        {
117            return min
118        } else
119        {
120            return valor
121        }
122    }
123 }
```

## Adicionar física aos canos para colidirem com o pássaro

As linhas abaixo são necessárias para adicionar física ao cano, dessa forma o pássaro irá colidir com o cano, deve-se indicar false no atributo dynamic para que os canos não caiam.

```
95  
96     cano1.physicsBody = SKPhysicsBody(rectangleOfSize: cano1.size)  
97     cano1.physicsBody?.dynamic = false // impede que o cano caia  
98     cano2.physicsBody = SKPhysicsBody(rectangleOfSize: cano2.size)  
99     cano2.physicsBody?.dynamic = false  
100  
  
    func criarCano()  
    {  
        77  
        78  
        79  
        80  
        81  
        82  
        83  
        84  
        85  
        86  
        87  
        88  
        89  
        90  
        91  
        92  
        93  
        94  
        95  
        96  
        97  
        98  
        99  
        100  
        101  
        102  
        103  
        104  
        105  
        106  
  
        let numeroRandom = arc4random() % UInt32(self.size.height / 3)  
        let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4  
  
        let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))  
        let apagaCano = SKAction.removeFromParent()  
        let sequenciaCano = SKAction.sequence([moveCano, apagaCano])  
  
        let imagemCanoBarra = SKTexture(imageNamed: "cano1")  
        let cano1 = SKSpriteNode(texture: imagemCanoBarra)  
        cano1.position = CGPointMake(x: self.size.width, y: alturaRandom)  
        cano1.runAction(sequenciaCano)  
  
        let imagemCanoTopo = SKTexture(imageNamed: "cano2")  
        let cano2 = SKSpriteNode(texture: imagemCanoTopo)  
        cano2.position = CGPointMake(x: self.size.width, y:(cano1.position.y + cano2.size.height +  
            alturaVao))  
        cano2.runAction(sequenciaCano)  
  
        cano1.physicsBody = SKPhysicsBody(rectangleOfSize: cano1.size)  
        cano1.physicsBody?.dynamic = false // impede que o cano caia  
        cano2.physicsBody = SKPhysicsBody(rectangleOfSize: cano2.size)  
        cano2.physicsBody?.dynamic = false  
  
        self.addChild(cano1)  
        self.addChild(cano2)  
  
        _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"),  
            userInfo: nil, repeats: false)  
    }  
  
    iPhone 5s - iPhone 5s / iOS 9.2 (13C75)  

```

## Inserir um v o para computar pontos

As linhas abaixo adicionam um vão na cena (103), um cálculo será feito para definir a posição do vão, o vão ficará deslocado um pouco mais à frente dos canos, lembre-se de adicionar o vão à cena (113).

## Seção 4

# Colisão.

### Contact Delegate - Adicionar um protocolo

Para determinar se o pássaro colidiu com o vâo, com o chão ou com o cano, precisamos invocar o protocolo SKPhysicsContactDelegate, veja como invocar esse protocolo na linha 11, para que funcione precisamos informar que a cena irá delegar o protocolo (31).

```
1 //  
2 // GameScene.swift  
3 // AgeBird  
4 //  
5 // Created by agesandro scarpioni on 25/01/16.  
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import SpriteKit  
10  
11 class GameScene: SKScene, SKPhysicsContactDelegate {  
    //na linha acima após a vírgula estamos adicionando um protocolo  
    //que serve para verificar quem colidiu com quem  
14    var bird = SKSpriteNode()  
15    var arrayPassaros:[SKTexture]=[]  
16  
27  
28    override func didMoveToView(view: SKView) {  
29        self.backgroundColor = UIColor.blackColor()  
30        //informa que a própria cena delega o protocolo  
31        self.physicsWorld.contactDelegate = self  
32
```



## Identificar os objetos

Deve-se criar um id para o pássaro, canos e vâo, dessa forma é possível computar as colisões.

```
1 //////////////////////////////////////////////////////////////////
2 // GameScene.swift
3 // AgeBird
4 //
5 // Created by agesandro scarpioni on 25/01/16.
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import SpriteKit
10
11 class GameScene: SKScene, SKPhysicsContactDelegate {
12     //na linha acima após a vírgula estamos adicionando um protocolo
13     //que serve para verificar quem colidiu com quem
14     var bird = SKSpriteNode()
15     var arrayPassaros:[SKTexture]=[]
16
17     var fundo = SKSpriteNode()
18     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
19     var escalaFundo = CGFloat(1.35)
20
21     var alturaVao = CGFloat(100)
22
23     let idBird:UInt32 = 1
24     let idCanos:UInt32 = 2
25     let idVao:UInt32 = 0 << 3
26
27
28     override func didMoveToView(view: SKView) {
29         self.backgroundColor = UIColor.blackColor()
30         //informa que a própria cena delega o protocolo
31         self.physicsWorld.contactDelegate = self
32
33         bird = SKSpriteNode(imageNamed: "passaro1")
34         bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
35 }
```

## Relacionar o bitMask para o pássaro

Antes de adicionarmos o pássaro a cena, devemos relacionar cada id aos tipos de colisões.

```
65  
66    bird.physicsBody?.categoryBitMask = idBird  
67    bird.physicsBody?.contactTestBitMask = idCanos  
68    bird.physicsBody?.collisionBitMask = idVao  
69
```

```
27    override func didMoveToView(view: SKView) {  
28        self.backgroundColor = UIColor.blackColor()  
29  
30        bird = SKSpriteNode(imageNamed: "passaro1")  
31        bird.position = CGPointMake(self.size.width/2.5, self.size.height/  
32  
33        for (var i=1; i<=12; i++){  
34            arrayPassaros.append(SKTexture(imageNamed: "passaro\(\(i)\)"))  
35        }  
36  
37        let animacao = SKAction.animateWithTextures(arrayPassaros, timePer  
38        let voar = SKAction.repeatActionForever(animacao)  
39        bird.runAction(voar)  
40        bird.setScale(0.85)  
41  
42        fundo = SKSpriteNode(texture: texturaFundo)  
43        fundo.position = CGPointMake(self.size.width/2, self.size.height/2  
44        fundo.setScale(escalaFundo)  
45  
46        let moveFundo = SKAction.moveByX(-texturaFundo.size().width * esca  
47        let reposicionaFundo = SKAction.moveByX(texturaFundo.size().width  
48        let moveFundoSempre = SKAction.repeatActionForever(SKAction.sequer  
49  
50        for var i:CGFloat=0; i<3; i++  
51        {  
52            fundo = SKSpriteNode(texture: texturaFundo)  
53            fundo.setScale(escalaFundo)  
54            fundo.position = CGPointMake(x: texturaFundo.size().width/2 + text  
55            fundo.alpha=0.75  
56            fundo.runAction(moveFundoSempre)  
57            self.addChild(fundo)  
58        }  
59  
60        bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height  
61        bird.physicsBody?.dynamic = true  
62        bird.physicsBody?.allowsRotation = false  
63        bird.physicsBody?.velocity = CGVectorMake(0, 0)  
64        bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))  
65  
66        bird.physicsBody?.categoryBitMask = idBird  
67        bird.physicsBody?.contactTestBitMask = idCanos  
68        bird.physicsBody?.collisionBitMask = idVao  
69
```



## Relacionar o bitMask para os canos e o vāo

Dentro da função criarCano antes de adicionarmos o cano e o vāo a cena, devemos relacionar cada id aos tipos de colisões.

```
83
84 func criarCano()
85 {
86     let numeroRandom = arc4random() % UInt32(self.size.height / 3)
87     let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4
88
89     let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))
90     let apagaCano = SKAction.removeFromParent()
91     let sequenciaCano = SKAction.sequence([moveCano, apagaCano])
92
93     let imagemCanoBarra = SKTexture(imageNamed: "cano1")
94     let cano1 = SKSpriteNode(texture: imagemCanoBarra)
95     cano1.position = CGPointMake(x: self.size.width, y: alturaRandom)
96     cano1.runAction(sequenciaCano)
97
98     let imagemCanoTopo = SKTexture(imageNamed: "cano2")
99     let cano2 = SKSpriteNode(texture: imagemCanoTopo)
100    cano2.position = CGPointMake(x: self.size.width, y:(cano1.position.y + cano2.size.y))
101    cano2.runAction(sequenciaCano)
102
103    cano1.physicsBody = SKPhysicsBody(rectangleOfSize: cano1.size)
104    cano1.physicsBody?.dynamic = false // impede que o cano caia
105    cano2.physicsBody = SKPhysicsBody(rectangleOfSize: cano2.size)
106    cano2.physicsBody?.dynamic = false
107
108    //muito parecido com o chão, porém vamos criar o vāo para computar pontos
109
110    let vao = SKNode()
111    //abaixo calculamos a altura do vāo que vai do final do cano1 até o começo do cano 2
112    vao.position = CGPointMake(x: self.size.width + cano1.size.width , y: cano1.position.y + cano1.size.height/2)
113    //deixa o vāo um pouco para a frente
114    vao.physicsBody = SKPhysicsBody(rectangleOfSize: CGSizeMake(1,alturaVao))
115    vao.physicsBody?.dynamic = false
116    //serve para que o vāo caminhe junto com o cano
117    vao.runAction(sequenciaCano)
118
119    cano1.physicsBody?.categoryBitMask = idCanos
120    cano2.physicsBody?.categoryBitMask = idCanos
121
122    vao.physicsBody?.collisionBitMask = idVao
123    vao.physicsBody?.categoryBitMask = idVao
124    vao.physicsBody?.contactTestBitMask = idBird
```

The code snippet shows the creation of two pipes (cano1 and cano2) and a moving floor (vao). Each pipe has a physics body with a category bit mask of idCanos. The moving floor also has a physics body with collision, category, and contact test bit masks set to idVao and idBird respectively. A red oval highlights the assignment of category bit masks to the pipes (lines 119-120), and a red arrow points from this oval to the assignment of the same bit masks to the moving floor (lines 118-125).

## **Relacionar o bitMask para o touchesBegan**

Dentro da função criarCano antes de adicionarmos o cano e o v o a cena, devemos relacionar cada id aos tipos de colisões.

## Implementar o didBeginContact

Existe uma função chamada didBeginContact que faz parte do protocolo contactDelegate que adicionamos, com essa função é possível testar de ocorreu colisão com o vāo ou com outros objetos na tela.

```
157
158     override func update(currentTime: CFTimeInterval) {
159         let num = bird.physicsBody!.velocity.dy as CGFloat
160         bird.zRotation = self.empinada( -1, max: 0.5, valor:num * 0.001)
161     }
162
163
164     func didBeginContact(contact: SKPhysicsContact)
165     {
166         if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
167             print("colidiu com o vāo")
168         }else{
169             print("Não colidiu com o vāo, acertei o chāo ou o cano")
170         }
171     }
172
173 }
```

# Placar

## Definir variáveis e objetos para o placar

Para computar pontos quando o pássaro passar pelo vôlei precisamos de um contador e um label para exibir a informação. Adicione as linhas 27 e 28.

```
8  
9 import SpriteKit  
10  
11 class GameScene: SKScene, SKPhysicsContactDelegate {  
    //na linha acima após a vírgula estamos adicionando um protocolo  
    //que serve para verificar quem colidiu com quem  
    var bird = SKSpriteNode()  
    var arrayPassaros:[SKTexture]=[]  
16  
17    var fundo = SKSpriteNode()  
18    var texturaFundo = SKTexture(imageNamed: "imagemFundo")  
19    var escalaFundo = CGFloat(1.35)  
20  
21    var alturaVao = CGFloat(100)  
22  
23    let idBird:UInt32 = 1  
24    let idCanos:UInt32 = 2  
25    let idVao:UInt32 = 0 << 3  
26  
27    → var score = 0  
28    var scoreLabel = SKLabelNode()  
29
```

## Definir um Label e suas características

Adicione um Label a cena como visto no detalhe (70), regule o alpha em 0.5 e a zPosition em 9, aproveite para deixar o pássaro como primeiro elemento da cena informando o zPosition do pássaro em 10 (78).

The diagram illustrates the addition of a label to a scene. A red oval highlights the section of code from line 76 to line 83. A red arrow points from this highlighted area to the right-hand column of code, which contains the detailed configuration for the scoreLabel.

```
71
72     bird.physicsBody?.categoryBitMask = idBird
73     bird.physicsBody?.contactTestBitMask = idCanos
74     bird.physicsBody?.collisionBitMask = idVao
75
76     //adicionando o label à cena
77     scoreLabel.fontName = "Verdana"
78     scoreLabel.fontSize = 200
79     scoreLabel.text = "0"
80     scoreLabel.position = CGPointMake(CGRectGetMidX(self.frame), self.frame.size.height/2)
81     scoreLabel.alpha=0.5
82     scoreLabel.zPosition=9 //posição em z quanto maior mais à frente
83     self.addChild(scoreLabel)
84
85     self.addChild(bird)
86     alturaVao = bird.size.height * 2.5
87
88     //Criamos um timer, não se usa mais (var timer) quando apenas declaramos e não utilizamos a
89     _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"),
90
91     let chao = SKNode()
```

69	//adicionando o label à cena
70	scoreLabel.fontName = "Verdana"
71	scoreLabel.fontSize = 200
72	scoreLabel.text = "0"
73	scoreLabel.position = CGPointMake(CGPointMake(CGRectGetMidX(self.frame), self.frame.size.height/2))
74	scoreLabel.alpha=0.5
75	scoreLabel.zPosition=9 //posição em z quanto maior mais à frente
76	self.addChild(scoreLabel)
77	bird.zPosition = 10
78	self.addChild(bird)
79	alturaVao = bird.size.height * 2.5
80	

## Acumular e exibir pontos

Dentro da área do if que avalia se o pássaro colidiu com o vāo, adicione um ponto ao score e exiba o resultado no texto do Label.

```
175
176 func didBeginContact(contact: SKPhysicsContact)
177 {
178     if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
179         print("colidiu com o vāo")
180         score+=1
181         scoreLabel.text = "\(score)"
182     }else{
183         print("Não colidiu com o vāo, acertei o chāo ou o cano")
184     }
185 }
186
```

# Início de Jogo

### Definir um controle para o início do Jogo

Crie uma variável boleana e a inicie com false como indicado na linha 30

```
1 //  
2 // GameScene.swift  
3 // AgeBird  
4 //  
5 // Created by agesandro scarpioni on 25/01/16.  
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import SpriteKit  
10  
11 class GameScene: SKScene, SKPhysicsContactDelegate {  
    //na linha acima após a vírgula estamos adicionando um protocolo  
    //que serve para verificar quem colidiu com quem  
    var bird = SKSpriteNode()  
    var arrayPassaros:[SKTexture]=[]  
16  
    var fundo = SKSpriteNode()  
18    var texturaFundo = SKTexture(imageNamed: "imagemFundo")  
19    var escalaFundo = CGFloat(1.35)  
20  
    var alturaVao = CGFloat(100)  
22  
    let idBird:UInt32 = 1  
    let idCanos:UInt32 = 2  
    let idVao:UInt32 = 0 << 3  
26  
    var score = 0  
28    var scoreLabel = SKLabelNode()  
29  
30    var comeceu:Bool = false  
31
```

## Iniciar o jogo ao toque da tela

Dentro da função touchesBegan monte um if para iniciar o jogo apenas quando a tela for clicada, inclua o B

```
141 |
142     override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
143     {
144         if !comecou{
145             comecou=true
146             bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
147             bird.physicsBody?.dynamic = true
148             bird.physicsBody?.allowsRotation = false
149             bird.physicsBody?.velocity = CGVectorMake(0, 0)
150             bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
151
152             bird.physicsBody?.categoryBitMask = idBird
153             bird.physicsBody?.contactTestBitMask = idCanos
154             bird.physicsBody?.collisionBitMask = idVao
155
156         }else{
157             bird.physicsBody?.velocity = CGVectorMake(0, 0)
158             bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
159         }
160
161     }
162 }
```

## Implementar o Update

Ao rodar o programa dará um erro conforme a imagem logo abaixo, isso ocorre porque como foi informado no slide anterior que se a tela não sofrer um clique o programa não inicia, durante as atualizações do frame (update), ele tenta pegar a velocidade do pássaro para fazer a empinada e não consegue. Corrija inserindo o if na linha 178 e a chave na linha 181.

The screenshot shows the Xcode interface during a debug session. The code editor displays the following Swift code:

```
176
177     override func update(currentTime: CFTimeInterval) {
178         let num = bird.physicsBody!.velocity.dy as CGFloat
179         bird.zRotation = self.empinada( -1, max: 0.5, valor:num * 0.001)
180     }
181
182
183     func didBeginContact(contact: SKPhysicsContact)
184     {
185         if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
186             print("colidiu com o v\u00e3o")
187             score++
188             scoreLabel.text = "\(score)"
189         }else{
190             print("N\u00e3o colidiu com o v\u00e3o, acertei o ch\u00e3o ou o cano")
191         }
192     }
193 }
```

The line 178 is highlighted in orange, indicating it's the source of the error. A tooltip above the line says "Thread 1: EXC\_BAD\_INSTRUCTION (code=EXC\_I386\_INVOP, subcode=0x0)". The debugger sidebar shows variables:

- currentTime = (CFTimeInterval) 16633.497577415001
- self = (AgeBird.GameScene) 0x00007f86304da600
- num = (CGFloat) 6.9275074980248534E-310

The bottom right shows the error message from LLDB:

```
requesting subtype without specifying idiom
2016-01-26 01:36:05.237 AgeBird[10568:268490] CUICatalog: Invalid Request:
requesting subtype without specifying idiom
fatal error: unexpectedly found nil while unwrapping an Optional value
(lldb)
```

The screenshot shows the Xcode code editor with two red arrows pointing to specific lines of code. The first arrow points to line 178, and the second arrow points to line 181. The code is identical to the one shown in the previous screenshot, except for the annotations.

```
176
177     override func update(currentTime: CFTimeInterval) {
178         if comecou{
179             let num = bird.physicsBody!.velocity.dy as CGFloat
180             bird.zRotation = self.empinada( -1, max: 0.5, valor:num * 0.001)
181         }
182     }
183 }
```

## Implementar o criarCano

Faça o mesmo tipo de if na área demarcada em azul, vejo o exemplo abaixo imaginando que a área em azul foi recortada e colada na linha 95 após o if montado.

```
90 }
91
92 func criarCano()
93 {
94     let numeroRandom = arc4random() % UInt32(self.size.height / 3)
95     let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4
96
97     let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))
98     let apagaCano = SKAction.removeFromParent()
99     let sequenciaCano = SKAction.sequence([moveCano, apagaCano])
100
101    let imagemCanoBarra = SKTexture(imageNamed: "cano1")
102    let cano1 = SKSpriteNode(texture: imagemCanoBarra)
103    cano1.position = CGPointMake(x: self.size.width, y: alturaRandom)
104    cano1.runAction(sequenciaCano)
105
106    let imagemCanoTopo = SKTexture(imageNamed: "cano2")
107    let cano2 = SKSpriteNode(texture: imagemCanoTopo)
108    cano2.position = CGPointMake(x: self.size.width, y:(cano1.position.y + cano2.size.height + alturaVao))
109    cano2.runAction(sequenciaCano)
110
111    cano1.physicsBody = SKPhysicsBody(rectangleOfSize: cano1.size)
112    cano1.physicsBody?.dynamic = false // impede que o cano cai
113    cano2.physicsBody = SKPhysicsBody(rectangleOfSize: cano2.size)
114    cano2.physicsBody?.dynamic = false
115
116    //muito parecido com o chão, porém vamos criar o vao para computar pontos
117
118    let vao = SKNode()
119    //abixo calculamos a altura do vao que vai do final do cano1 até o começo do cano 8
120    vao.position = CGPointMake(x: self.size.width + cano1.size.width , y: cano1.position.y + cano1.size.height/2)
121    //deixa o vao um pouco para a frente
122    vao.physicsBody = SKPhysicsBody(rectangleOfSize: CGSizeMake(1,alturaVao))
123    vao.physicsBody?.dynamic = false
124    //serve para que o vao caminhe junto com o cano
125    vao.runAction(sequenciaCano)
126
127    cano1.physicsBody?.categoryBitMask = idCanos
128    cano2.physicsBody?.categoryBitMask = idCanos
129
130    vao.physicsBody?.collisionBitMask = idVao
131    vao.physicsBody?.categoryBitMask = idVao
132    vao.physicsBody?.contactTestBitMask = idBird
133
134    self.addChild(cano1)
135    self.addChild(cano2)
136    self.addChild(vao)// adiciona o vao à cena
137
138    _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false)
139
140
141
142 override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
143     if !comecou{
144         comeceu = true
145     }
146 }
```

```
91
92
93 func criarCano()
94 {
95     if comecou{
96         |
97     }
98     _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false)
99 }
```

Veja como ficará a implementação do criarCano após a variável chamada comecou receber o true. Isso foi feito para evitar que os canos passem sem antes o usuário ter clicado na tela para iniciar o jogo.

```
90    }
91
92    func criarCano()
93    {
94        if comecou{
95            let numeroRandom = arc4random() % UInt32(self.size.height / 3)
96            let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4
97
98            let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))
99            let apagaCano = SKAction.removeFromParent()
100           let sequenciaCano = SKAction.sequence([moveCano, apagaCano])
101
102           let imagemCanoBarra = SKTexture(imageNamed: "cano1")
103           let cano1 = SKSpriteNode(texture: imagemCanoBarra)
104           cano1.position = CGPointMake(x: self.size.width, y: alturaRandom)
105           cano1.runAction(sequenciaCano)
106
107           let imagemCanoTopo = SKTexture(imageNamed: "cano2")
108           let cano2 = SKSpriteNode(texture: imagemCanoTopo)
109           cano2.position = CGPointMake(x: self.size.width, y:(cano1.position.y + cano2.size.height + alturaVao))
110           cano2.runAction(sequenciaCano)
111
112           cano1.physicsBody = SKPhysicsBody(rectangleOfSize: cano1.size)
113           cano1.physicsBody?.dynamic = false // impede que o cano caia
114           cano2.physicsBody = SKPhysicsBody(rectangleOfSize: cano2.size)
115           cano2.physicsBody?.dynamic = false
116
117           //muito parecido com o chão, porém vamos criar o vão para computar pontos
118
119           let vao = SKNode()
120           //abaixo calculamos a altura do vão que vai do final do cano1 até o começo do cano B
121           vao.position = CGPointMake(x: self.size.width + cano1.size.width , y: cano1.position.y + cano1.size.height/2)
122           //deixa o vao um pouco para a frente
123           vao.physicsBody = SKPhysicsBody(rectangleOfSize: CGSizeMake(1,alturaVao))
124           vao.physicsBody?.dynamic = false
125           //serve para que o vao caminhe junto com o cano
126           vao.runAction(sequenciaCano)
127
128           cano1.physicsBody?.categoryBitMask = idCanos
129           cano2.physicsBody?.categoryBitMask = idCanos
130
131           vao.physicsBody?.collisionBitMask = idVao
132           vao.physicsBody?.categoryBitMask = idVao
133           vao.physicsBody?.contactTestBitMask = idBird
134
135           self.addChild(cano1)
136           self.addChild(cano2)
137           self.addChild(vao)// adiciona o vao à cena
138
139    }
140    _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"), userInfo: nil, repeats: false)
141
142 }
```

## Definir o zPosition dos canos

Como o pássaro foi marcado como posição z que representa a profundidade do objeto na cena em 10, o placar em 9, posicione os canos em 8, veja a imagem abaixo:

```
136  
137     cano1.zPosition = 8  
138     cano2.zPosition = 8|  
139     self.addChild(cano1)  
140     self.addChild(cano2)  
141     self.addChild(vao)// adiciona o vao à cena  
142
```

# Fim de Jogo

### Criar um controle para o fim de jogo

Crie uma variável booleana chamada terminou como mostra na linha 31, ela terá um comportamento idêntico a variável iniciou, porém ela receberá true quando o pássaro colidir com o chão ou com o cano.

```
9 import SpriteKit
10
11 class GameScene: SKScene, SKPhysicsContactDelegate {
12     //na linha acima após a vírgula estamos adicionando um protocolo
13     //que serve para verificar quem colidiu com quem
14     var bird = SKSpriteNode()
15     var arrayPassaros:[SKTexture]=[]
16
17     var fundo = SKSpriteNode()
18     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
19     var escalaFundo = CGFloat(1.35)
20
21     var alturaVao = CGFloat(100)
22
23     let idBird:UInt32 = 1
24     let idCanos:UInt32 = 2
25     let idVao:UInt32 = 0 << 3
26
27     var score = 0
28     var scoreLabel = SKLabelNode()
29
30     var comecou:Bool = false
31     var terminou:Bool = false
32
```

## Implementar o término do jogo em didBeginContact e em criarCano

Melhore o if da função criar cano para que os canos sejam criados se o programa já começou e ainda não terminou, isso é visto na linha (98) da tela abaixo, para que funcione implemente um true para a variável terminou como visto na linha (206).

```
95  
96    func criarCano()  
97    {  
98        if comeceu && !terminou {  
99            let numeroRandom = arc4random() % UInt32(self.size.height / 3)  
100           let alturaRandom = CGFloat(numeroRandom) - self.size.height / 4  
101  
102           let moveCano = SKAction.moveByX(-self.frame.size.width * 2, y: 0, duration: NSTimeInterval(self.size.width / 100))  
103           let apagaCano = SKAction.removeFromParent()  
104           let sequenciaCano = SKAction.sequence([moveCano, apagaCano])  
105  
  
197  
198    func didBeginContact(contact: SKPhysicsContact)  
199    {  
200        if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {  
201            print("colidiu com o vāo")  
202            score++  
203            scoreLabel.text = "\(score)"  
204        }else{  
205            print("Não colidiu com o vāo, acertei o chão ou o cano")  
206            terminou = true  
207        }  
208    }  
209 }
```

## Implementar o término do jogo em touchesBegan

Melhore o if da função touchesBegan adicionando um else (164) e um if de não terminou na linha (166).

```
147  
150    override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?)  
151    {  
152        if !comecou{  
153            comecou=true  
154            bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)  
155            bird.physicsBody?.dynamic = true  
156            bird.physicsBody?.allowsRotation = false  
157            bird.physicsBody?.velocity = CGVectorMake(0, 0)  
158            bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))  
159  
160            bird.physicsBody?.categoryBitMask = idBird  
161            bird.physicsBody?.contactTestBitMask = idCanos  
162            bird.physicsBody?.collisionBitMask = idVao  
163  
164        }else{  
165  
166            if !terminou{  
167                bird.physicsBody?.velocity = CGVectorMake(0, 0)  
168                bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))  
169            }  
170        }  
171    }  
172  
173    }  
174 }
```



## Criar objeto para receber as cenas

Crie um objMoveCena como SKNode (32).

```
8 import SpriteKit
9
10
11 class GameScene: SKScene, SKPhysicsContactDelegate {
12     //na linha acima após a vírgula estamos adicionando um protocolo
13     //que serve para verificar quem colidiu com quem
14     var bird = SKSpriteNode()
15     var arrayPassaros:[SKTexture]=[]
16
17     var fundo = SKSpriteNode()
18     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
19     var escalaFundo = CGFloat(1.35)
20
21     var alturaVao = CGFloat(100)
22
23     let idBird:UInt32 = 1
24     let idCanos:UInt32 = 2
25     let idVao:UInt32 = 0 << 3
26
27     var score = 0
28     var scoreLabel = SKLabelNode()
29
30     var comecou:Bool = false
31     var terminou:Bool = false
32     var objMoveCena = SKNode()
33
```

## Adicionar o fundo da tela a um objeto de cena

No ponto indicado em didMoveToView, comente a linha onde foi adicionado o fundo à cena(68) e vamos adicionar o fundo ao objeto objMoveCena(71).

```
35
36     override func didMoveToView(view: SKView) {
37         self.backgroundColor = UIColor.blackColor()
38         //informa que a própria cena delega o protocolo
39         self.physicsWorld.contactDelegate = self
40
41         bird = SKSpriteNode(imageNamed: "passaro1")
42         bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
43
44         for (var i=1; i<=12; i++){
45             arrayPassaros.append(SKTexture(imageNamed: "passaro\((i)")))
46         }
47
48         let animacao = SKAction.animateWithTextures(arrayPassaros, timePerFrame: 0.05)
49         let voar = SKAction.repeatActionForever(animacao)
50         bird.runAction(voar)
51         bird.setScale(0.85)
52
53         fundo = SKSpriteNode(texture: texturaFundo)
54         fundo.position = CGPointMake(self.size.width/2, self.size.height/2)
55         fundo.setScale(escalaFundo)
56
57         let moveFundo = SKAction.moveByX(-texturaFundo.size().width * escalaFundo, y: 0, duration: 9)
58         let reposicionaFundo = SKAction.moveByX(texturaFundo.size().width * escalaFundo, y:0, duration: 0)
59         let moveFundoSempre = SKAction.repeatActionForever(SKAction.sequence([moveFundo, reposicionaFundo]))
60
61         for var i:CGFloat=0; i<3; i++
62     {
63             fundo = SKSpriteNode(texture: texturaFundo)
64             fundo.setScale(escalaFundo)
65             fundo.position = CGPointMake(x: texturaFundo.size().width/2 + texturaFundo.size().width * i * escalaFundo, y: self.frame.height/2)
66             fundo.alpha=0.75
67             fundo.runAction(moveFundoSempre)
68             //self.addChild(fundo)
69             //ao invés de adicionar o fundo à cena veja linha acima que comentei, adicione o fundo
70             //a um objeto como na linha abaixo, dessa forma quando o programa terminar paramos o objeto
71             objMoveCena.addChild(fundo)
72
73
74 }
```



Veja nessa página como ficou a alteração do trecho da página anterior.

```
60
61     for var i:CGFloat=0; i<3; i++
62     {
63         fundo = SKSpriteNode(texture: texturaFundo)
64         fundo.setScale(escalaFundo)
65         fundo.position = CGPointMake(x: texturaFundo.size().width/2 + texturaFundo.size().width * i *
66             escalaFundo, y: self.frame.height/2)
67         fundo.alpha=0.75
68         fundo.runAction(moveFundoSempre)
69         //self.addChild(fundo)
70         //ao invés de adicionar o fundo à cena veja linha acima que comentei, adicione o fundo
71         //a um objeto como na linha abaixo, dessa forma quando o programa terminar paramos o objeto
72         objMoveCena.addChild(fundo)
73
74 }
```

Adicione o novo objeto que contém o fundo da tela à cena

```
35
36     override func didMoveToView(view: SKView) {
37         self.addChild(objMoveCena) //adicionando o objeto de fundo a cena
38
39         self.backgroundColor = UIColor.blackColor()
40         //informa que a própria cena delega o protocolo
41         self.physicsWorld.contactDelegate = self
42
43         bird = SKSpriteNode(imageNamed: "passaro1")
44         bird.position = CGPointMake(self.size.width/2.5, self.size.height/2)
45
```

Na função criarCano comente as linhas onde o cano e o v o foram adicionados ´ cena, e adicione esses objetos ao como filhos de objMoveCena linhas 148 a 155.

```
144
145
146     cano1.zPosition = 8
147     cano2.zPosition = 8
148     //self.addChild(cano1)
149     //self.addChild(cano2)
150     //self.addChild(vao)// adiciona o vao ´ cena
151     //adicionar todos elementos de cano e v o ao objMoverCena
152     //dessa forma o cano tamb m ir  parar
153     objMoveCena.addChild(cano1)
154     objMoveCena.addChild(cano2)
155     objMoveCena.addChild(vao)// adiciona o vao ´ cena
156 }
157 _ = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector: Selector("criarCano"),
158                                         userInfo: nil, repeats: false)
159 }
160 override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?)
161 {
162     if !comecou{
```

## Criar um Label para o fim de jogo

Inclua um label para o Game Over como mostra a linha (33).

```
10 class GameScene: SKScene, SKPhysicsContactDelegate {
11     //na linha acima após a vírgula estamos adicionando um protocolo
12     //que serve para verificar quem colidiu com quem
13     var bird = SKSpriteNode()
14     var arrayPassaros:[SKTexture]=[]
15
16     var fundo = SKSpriteNode()
17     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
18     var escalaFundo = CGFloat(1.35)
19
20     var alturaVao = CGFloat(100)
21
22     let idBird:UInt32 = 1
23     let idCanos:UInt32 = 2
24     let idVao:UInt32 = 0 << 3
25
26     var score = 0
27     var scoreLabel = SKLabelNode()
28
29     var comecou:Bool = false
30     var terminou:Bool = false
31     var objMoveCena = SKNode()
32     var gameOverLavel = SKLabelNode()
33
34
```

## Parar a cena e adicionar um texto de fim de Jogo

Pare a cena como mostra informando 0 no speed (218) e inclua no selector (219) a chamada à função textoGameOver (224)

```
208
209 func didBeginContact(contact: SKPhysicsContact)
210 {
211     if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
212         print("colidiu com o v o")
213         score++
214         scoreLabel.text = "\(score)"
215     }else{
216         print("N o colidiu com o v o, acertei o ch o ou o cano")
217         terminou = true
218         objMoveCena.speed = 0
219         _ = NSTimer.scheduledTimerWithTimeInterval(1, target: self, selector: Selector("textoGameOver"),
220                                         userInfo: nil, repeats: false)
221     }
222 }
223
224 func textoGameOver(){
225     gameOverLavel.fontName = "Verdana"
226     gameOverLavel.fontSize = 25
227     gameOverLavel.text = "Toque para reiniciar!"
228     gameOverLavel.position = CGPointMake(CGRectGetMidX(self.frame), self.frame.size.height/2)
229     gameOverLavel.zPosition=11 //posi o em z quanto maior mais ´a frente
230     self.addChild(gameOverLavel)
231 }
232
```

# Transição de cenas

### Definir controle para iniciar o jogo

Para reiniciar o jogo inclua a variável booleana tocouNaTelaParaReinicar como false (34).

```
8 import SpriteKit
9
10 class GameScene: SKScene, SKPhysicsContactDelegate {
11     //na linha acima após a vírgula estamos adicionando um protocolo
12     //que serve para verificar quem colidiu com quem
13     var bird = SKSpriteNode()
14     var arrayPassaros:[SKTexture]=[]
15
16     var fundo = SKSpriteNode()
17     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
18     var escalaFundo = CGFloat(1.35)
19
20     var alturaVao = CGFloat(100)
21
22     let idBird:UInt32 = 1
23     let idCanos:UInt32 = 2
24     let idVao:UInt32 = 0 << 3
25
26     var score = 0
27     var scoreLabel = SKLabelNode()
28
29     var comeceu:Bool = false
30     var terminou:Bool = false
31     var objMoveCena = SKNode()
32     var gameOverLavel = SKLabelNode()
33     var tocouNaTelaParaReinicar:Bool = false
34
35
```

Na função textoGameOver reverta para true a variável tocouNaTelaParaReinicar (234).

```
226
227 func textoGameOver(){
228     gameOverLavel.fontName = "Verdana"
229     gameOverLavel.fontSize = 25
230     gameOverLavel.text = "Toque para reiniciar!"
231     gameOverLavel.position = CGPointMake(CGRectGetMidX(self.frame), self.frame.size.height/2)
232     gameOverLavel.zPosition=11 //posição em z quanto maior mais à frente
233     self.addChild(gameOverLavel)
234     tocouNaTelaParaReinicar=true
235 }
236
```

## Implementar transição de cena para início de jogo

Na função touchesBegan inclua o if e else (163 até 168) para testar se a tela foi clicada e caso positivo uma transição entre a cena atual e a próxima cena será apresentada.

```
160
161     override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
162
163         if tocouNaTelaParaReiniciar{
164             let transicao = SKTransition.doorwayWithDuration(1)
165             let proximaCena = GameScene(size: self.size)
166             proximaCena.scaleMode = .AspectFill
167             self.view?.presentScene(proximaCena, transition: transicao)
168         }else{
169             if !comecou{
170                 comecou=true
171                 bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
172                 bird.physicsBody?.dynamic = true
173                 bird.physicsBody?.allowsRotation = false
174                 bird.physicsBody?.velocity = CGVectorMake(0, 0)
175                 bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
176
177                 bird.physicsBody?.categoryBitMask = idBird
178                 bird.physicsBody?.contactTestBitMask = idCanos
179                 bird.physicsBody?.collisionBitMask = idVao
180             }else{
181                 if !terminou{
182                     bird.physicsBody?.velocity = CGVectorMake(0, 0)
183                     bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
184
185                 }
186             }
187         }
188     }
189 }
```

## Controle de erro

O erro ocorreu porque por mais que uma vez foi adicionado o GameOverLabel, observe na linha 222 a chamada após 1 segundo, o else da linha 218 deverá ser melhorado.

The screenshot shows the Xcode interface with the Swift code for a game's physics contact handling and game over logic. The code includes a didBeginContact method and a textGameOver function. In the textGameOver function, there is a commented-out line that attempts to add the game over label again. The debugger output window at the bottom right shows an uncaught exception due to attempting to add a node with a parent.

```
211
212     func didBeginContact(contact: SKPhysicsContact)
213     {
214         if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
215             print("colidiu com o v\u00e3o")
216             score++
217             scoreLabel.text = "\(score)"
218         }else{
219             print("N\u00e3o colidiu com o v\u00e3o, acertei o ch\u00e3o ou o cano")
220             terminou = true
221             objMoveCena.speed = 0
222             _ = NSTimer.scheduledTimerWithTimeInterval(1, target: self, selector: Selector("textoGameOver"),
223                                         userInfo: nil, repeats: false)
224         }
225     }
226
227     func textoGameOver(){
228         gameOverLabel.fontName = "Verdana"
229         gameOverLabel.fontSize = 25
230         gameOverLabel.text = "Toque para reiniciar!"
231         gameOverLabel.position = CGPointMake(CGRectGetMidX(self.frame), self.frame.size.height/2)
232         gameOverLabel.zPosition=11 //posi\u00e7\u00e3o em z quanto maior mais \u00e1 frente
233         self.addChild(gameOverLabel)
234         //toqueParaTelaParaReiniciar=true
235     }
236 }
```

Terminating app due to uncaught exception  
'NSInvalidArgumentException', reason: 'Attempted to add  
a SKNode which already has a parent: <SKLabelNode>  
name:'(null)' text:'Toque para reiniciar!'  
fontName:'Verdana' position:{512, 384}'  
\*\*\* First throw call stack:

No local abaixo do else (219) monte um if para testar que só irá rodar a frase e chamar a função “textoGameOver” apenas se ainda não terminou, logo em seguida a variável terminou recebe true e o programa não entra mais nessa parte onde a função é chamada.

```
211
212 func didBeginContact(contact: SKPhysicsContact)
213 {
214     if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
215         print("colidiu com o vāo")
216         score++
217         scoreLabel.text = "\(score)"
218     }else{
219         if !terminou{
220             print("Nāo colidiu com o vāo, acertei o chāo ou o cano")
221             terminou = true
222             objMoveCena.speed = 0
223             _ = NSTimer.scheduledTimerWithTimeInterval(1, target: self, selector:
224                 Selector("textoGameOver"), userInfo: nil, repeats: false)
225         }
226     }
227 }
```

## Criar efeito de background

Para um efeito piscante a função chamada piscaBackGround irá remover as Actions que serão apresentadas, foi criado uma espera de meio segundo e 3 blocos que irão alternar entre cores banco, vermelho e preto, isso ocorrerá repetidas vezes durante 4 vezes. Para que isso funcione é necessário chamar a função piscaBackGround quando o jogo terminar. Veja a chamada na próxima página.

```
239
240     func piscaBackGround()
241     {
242
243         self.removeActionForKey("flash")
244         let espera = SKAction.waitForDuration(0.05)
245         let bgBranco = SKAction.runBlock({() in self.setBgBranco()})
246         let bgVermelho = SKAction.runBlock({() in self.setBgVermelho()})
247         let bgNormal = SKAction.runBlock({() in self.setBgNormal()})
248
249         let sequenceOfActions = SKAction.sequence([bgBranco,espera,bgVermelho,espera,bgNormal])
250         let repeatSequence = SKAction.repeatAction(sequenceOfActions, count: 4);
251         self.runAction(repeatSequence, withKey: "flash")
252     }
253
254     func setBgVermelho()
255     {
256         self.backgroundColor = UIColor.redColor()
257     }
258     func setBgBranco()
259     {
260         self.backgroundColor = UIColor.whiteColor()
261     }
262     func setBgNormal()
263     {
264         self.backgroundColor = UIColor.blackColor()
265     }
266
```

## Chamar o efeito de background

Para chamar o efeito criado na página anterior invoque o método piscaBackGround (223).

```
211
212     func didBeginContact(contact: SKPhysicsContact)
213     {
214         if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
215             print("colidiu com o v o")
216             score++
217             scoreLabel.text = "\(score)"
218         }else{
219             if !terminou{
220                 print("N o colidiu com o v o, acertei o ch o ou o cano")
221                 terminou = true
222                 objMoveCena.speed = 0
223                 → self.piscaBackGround()
224                 _ = NSTimer.scheduledTimerWithTimeInterval(1, target: self, selector:
225                     Selector("textoGameOver"), userInfo: nil, repeats: false)
226             }
227         }
228     }
```



## Capítulo 3

# Audio

---

Nesse capítulo iremos tratar de sons e músicas de fundo, definindo onde será o melhor local para executar os códigos de audio.



# Manipulando os arquivos de audio

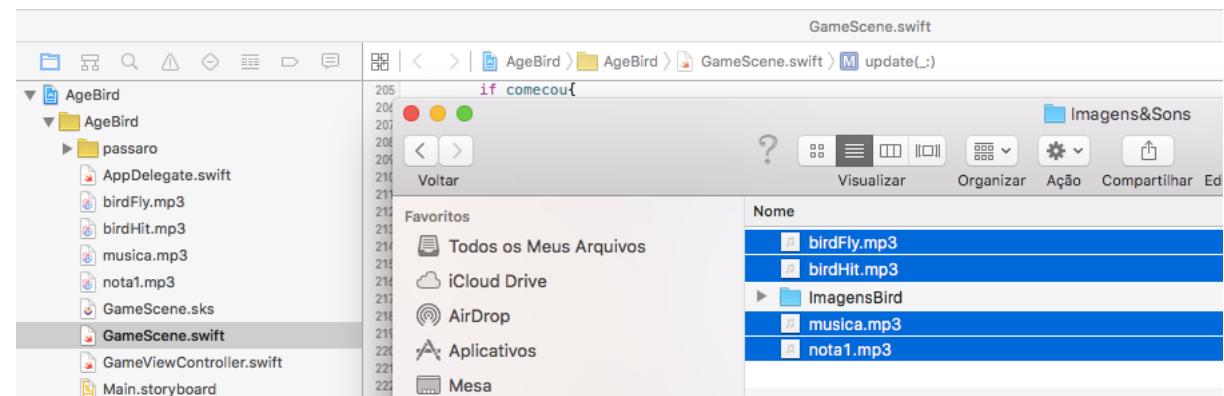
### NESTE CAPÍTULO VOCÊ IRÁ:

1. Carregar o som de fundo
2. Implementar o som para o score e para a colisão
3. Implementar o som para o vôo do pássaro
4. Definir o local e executar o som de fundo

Nessa seção iremos definir as actions que representarão os sons do jogos e definir quais são os melhores locais para a implementação. O primeiro passo é importar os arquivos de som para o projeto.

### Carregando os arquivos de som

Primeiro passo é carregar os arquivos de som para o projeto, use o finder para arrastar os arquivos de audio para o projeto.



## Definir na cena quais são os arquivos de audio

Defina 3 variáveis que representarão o som de voar, colidir e pontuar na cena do spritekit.

```
9 import SpriteKit
10
11 class GameScene: SKScene, SKPhysicsContactDelegate {
12     //na linha acima após a vírgula estamos adicionando um protocolo
13     //que serve para verificar quem colidiu com quem
14     var bird = SKSpriteNode()
15     var arrayPassaros:[SKTexture]=[]
16
17     var fundo = SKSpriteNode()
18     var texturaFundo = SKTexture(imageNamed: "imagemFundo")
19     var escalaFundo = CGFloat(1.35)
20
21     var alturaVao = CGFloat(100)
22
23     let idBird:UInt32 = 1
24     let idCanos:UInt32 = 2
25     let idVao:UInt32 = 0 << 3
26
27     var score = 0
28     var scoreLabel = SKLabelNode()
29
30     var comecou:Bool = false
31     var terminou:Bool = false
32     var objMoveCena = SKNode()
33     var gameOverLavel = SKLabelNode()
34     var tocouNaTelaParaReinic平ar:Bool = false
35
36     var somVoar = SKAction.playSoundFileNamed("birdFly.mp3", waitForCompletion: false)
37     var somColidir = SKAction.playSoundFileNamed("birdHit.mp3", waitForCompletion: false)
38     var somScore = SKAction.playSoundFileNamed("nota1.mp3", waitForCompletion: false)
39
40
```

## Efeito de som para o score e para a colisão

Dentro da função didBeginContact inclua a linha para o som do score self.runAction (224), repita o mesmo comando para o som da colisão do pássaro com o chão ou o cano na linha (231).

```
217
218     func didBeginContact(contact: SKPhysicsContact)
219     {
220         if contact.bodyA.categoryBitMask == idVao || contact.bodyB.categoryBitMask == idVao {
221             print("colidiu com o vāo")
222             score++
223             scoreLabel.text = "\(score)"
224             self.runAction(somScore)
225         }else{
226             if !terminou{
227                 print("Não colidiu com o vāo, acertei o chāo ou o cano")
228                 terminou = true
229                 objMoveCena.speed = 0
230                 self.piscaBackGround()
231                 self.runAction(somColidir)
232                 _ = NSTimer.scheduledTimerWithTimeInterval(1, target: self, selector:
233                     Selector("textoGameOver"), userInfo: nil, repeats: false)
234             }
235         }
236     }
```

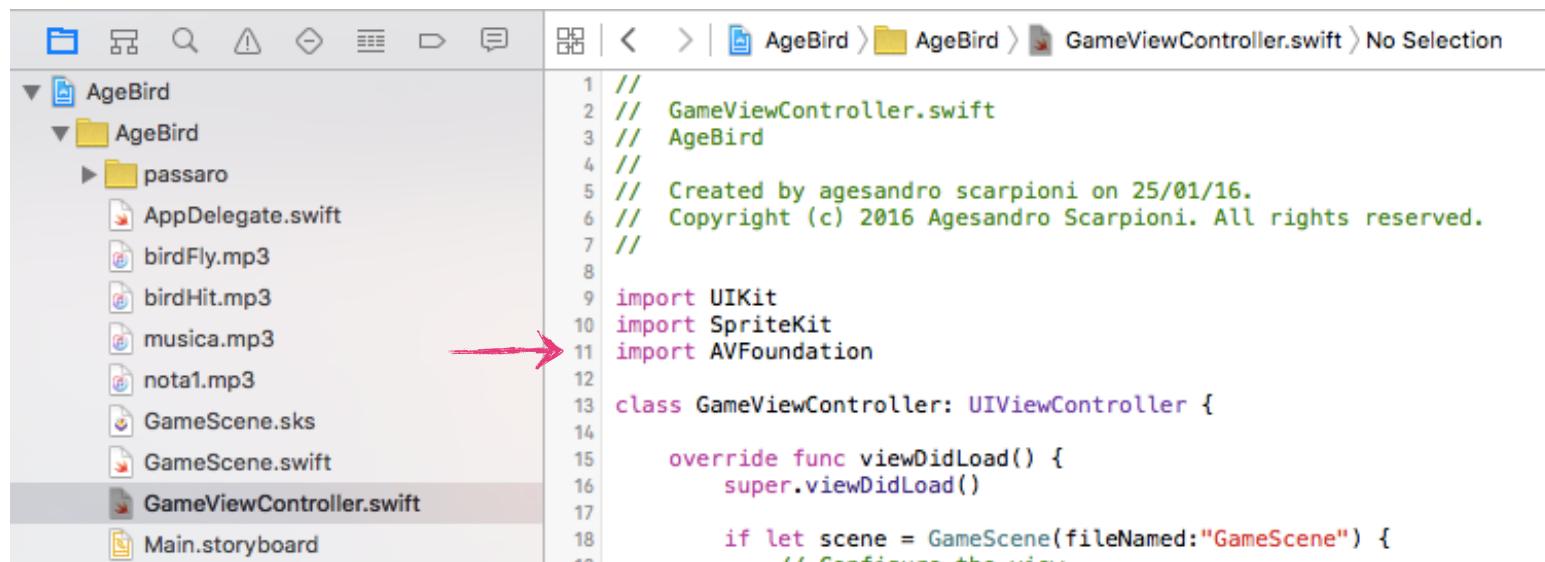
## Efeito de som para o voo

Dentro da função touchesBegan inclua a linha para o som do voo do pássaro (189) invocando para a cena o runAction do audio “somVoar”.

```
165
166     override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
167
168         if tocouNaTelaParaReiniciar{
169             let transicao = SKTransition.doorwayWithDuration(1)
170             let proximaCena = GameScene(size: self.size)
171             proximaCena.scaleMode = .AspectFill
172             self.view?.presentScene(proximaCena, transition: transicao)
173         }else{
174             if !comecou{
175                 comecou=true
176                 bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
177                 bird.physicsBody?.dynamic = true
178                 bird.physicsBody?.allowsRotation = false
179                 bird.physicsBody?.velocity = CGVectorMake(0, 0)
180                 bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
181
182                 bird.physicsBody?.categoryBitMask = idBird
183                 bird.physicsBody?.contactTestBitMask = idCanos
184                 bird.physicsBody?.collisionBitMask = idVao
185             }else{
186                 if !terminou{
187                     bird.physicsBody?.velocity = CGVectorMake(0, 0)
188                     bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
189                     self.runAction(somVoar)
190
191                 }
192             }
193         }
194     }
195 }
```

## Som de fundo

Uma cena da spritekit não é o melhor local para inserir a música de fundo, imagine um jogo que você deve passar de uma cena para outra e você não quer que a música inicie novamente. Nesse caso o melhor a fazer é inserir a música de fundo na cena pai, para isso a primeira coisa a fazer é importar o AVFoundation, veja linha (11).



```
1 //////////////////////////////////////////////////////////////////
2 //////////////////////////////////////////////////////////////////
3 //////////////////////////////////////////////////////////////////
4 //////////////////////////////////////////////////////////////////
5 //////////////////////////////////////////////////////////////////
6 //////////////////////////////////////////////////////////////////
7 //////////////////////////////////////////////////////////////////
8
9 import UIKit
10 import SpriteKit
11 import AVFoundation
12
13 class GameViewController: UIViewController {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17
18         if let scene = GameScene(fileNamed:"GameScene") {
19             // Configure the view
20
21             // Set the scale mode to aspect fill
22             scene.scaleMode = .AspectFill
23
24             // Present the scene as the view controller's root content.
25             self.view.presentScene(scene)
26
27             self.view.backgroundColor = UIColor.black
28
29         }
30
31     }
32
33 }
```

Crie um objeto chamado player que servirá para tocar a música de fundo(16). A função abaixo que inicia na linha 17 e vai até a linha 23 será a responsável por tocar a música de fundo continuamente ajustando o número de loops em -1 (20) e com o volume ajustado em 0.35 (21). Para que a música seja iniciada quando a cena pai for carregada, em viewDidLoad invoque o método play do backgroundMusic (26).

```
1 ////
2 // GameViewController.swift
3 // AgeBird
4 //
5 // Created by agesandro scarpioni on 25/01/16.
6 // Copyright (c) 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10 import SpriteKit
11 import AVFoundation
12
13 class GameViewController: UIViewController {
14
15
16     var player:AVAudioPlayer = AVAudioPlayer()
17     lazy var backgroundMusic: AVAudioPlayer = {
18         let url = NSBundle.mainBundle().URLForResource("musica", withExtension: "mp3")
19         let player = try? AVAudioPlayer(contentsOfURL: url!)
20         player!.numberOfLoops = -1 //igual a infinito
21         player!.volume=0.35
22         return player!
23 }()
24
25     override func viewDidLoad() {
26         super.viewDidLoad()
27         backgroundMusic.play()
28 }
```

# Partículas

---

Nesse capítulo você irá tratar de emissão de partículas e as configurações de seus atributos como velocidade e quantidade de emissão de partículas



# Criando uma função para partículas

### NESTE CAPÍTULO VOCÊ IRÁ:

1. Aprender a configurar uma função de partículas
2. Implementar a função de partículas
3. Encontrar o local apropriado para chamar a função de partículas

### Implementando a função criarParticulas

Criar um objeto para receber a textura da pena (279) e um objeto do tipo SKEMitterNode para emitir as partículas (280), associar a textura da pena ao emissor (281) e definir que a posição da emissão da partícula será no centro da imagem do pássaro (282).

Existem alguns atributos do objeto necessários para configurar a partícula como particleBirthRate (283) que define a taxa de criação de partículas, numParticlesToEmit (284) que define a quantidade de partículas emitidas, particleLifetime (285) que define o tempo de vida da partícula em segundos, xAcceleration e yAcceleration (287) e (288) que define a aceleração das partículas no eixo x e y, particleSpeed (290) que define a velocidade de cada partícula, particleSpeedRange (291) que define o intervalo de velocidade de emissão da partícula, particleRotationSpeed e particleRotationRange (293 e 294) que juntos controlam a velocidade e o intervalo de rotação da partícula, emissionAngle e emissionAngleRange (295 e 296) que juntos definem o angulo e o intervalo de emissão das partículas, por final temos o particleColorAlphaSpeed e particleColorAlphaRange (298 e 299) que juntos definem o Alpha (opacidade) e o intervalo do Alpha.

## Implementando a função criarParticulas

Abaixo segue a implementação da função criar particulas, além da criação dos objetos pena e emissor e a configuração dos atributos do emissor, existem duas linhas (301 e 302), a primeira define a animação de Alpha em particleAlphaSequence que faz com que a partícula vá desaparecendo, e particleScaleSequence que define a animação da escala. Ao final o emissor que é do tipo SKEmitterNode é adicionado à cena.

```
274 //Falar do editor de particular fazendo tudo visualmente
275 //File --> New --> Resource --> SpriteKit Particle File
276 func criaParticulas()
277 {
278     let pena:SKTexture = SKTexture(imageNamed: "pena")
279     let emissorPenas:SKEmitterNode = SKEmitterNode()
280     emissorPenas.particleTexture = pena
281     emissorPenas.position = CGPointMake(bird.position.x+bird.size.width/2,bird.position.y+bird.size.
282                                         height/2)
283     emissorPenas.particleBirthRate = 100
284     emissorPenas.numParticlesToEmit = 7;
285     emissorPenas.particleLifetime = 1.3
286
287     emissorPenas.xAcceleration = 0
288     emissorPenas.yAcceleration = 0
289
290     emissorPenas.particleSpeed = 100
291     emissorPenas.particleSpeedRange = 200
292
293     emissorPenas.particleRotationSpeed = -10
294     emissorPenas.particleRotationRange = 4
295     emissorPenas.emissionAngle = 3
296     emissorPenas.emissionAngleRange = 3.14
297
298     emissorPenas.particleColorAlphaSpeed = 0.1
299     emissorPenas.particleColorAlphaRange = 1
300
301     emissorPenas.particleAlphaSequence = SKKeyframeSequence(keyframeValues: [1,0], times: [0,1])
302     emissorPenas.particleScaleSequence = SKKeyframeSequence(keyframeValues: [1,0], times: [0,1])
303
304     self.addChild(emissorPenas)
305
306 }
307 }
```

## Chamar a função criarParticulas em touchesBegan

Em dois pontos da função touchesBegan deve-se chamar a função criarParticulas, o local ideal para chamar a função é após a aplicação do impulso (180 e 189).

```
165
166     override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
167     {
168         if tocouNaTelaParaReiniciar{
169             let transicao = SKTransition.doorwayWithDuration(1)
170             let proximaCena = GameScene(size: self.size)
171             proximaCena.scaleMode = .AspectFill
172             self.view?.presentScene(proximaCena, transition: transicao)
173         }else{
174             if !comecou{
175                 comecou=true
176                 bird.physicsBody = SKPhysicsBody(circleOfRadius: bird.size.height / 2)
177                 bird.physicsBody?.dynamic = true
178                 bird.physicsBody?.allowsRotation = false
179                 bird.physicsBody?.velocity = CGVectorMake(0, 0)
180                 bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
181                 self.criaParticulas()
182
183                 bird.physicsBody?.categoryBitMask = idBird
184                 bird.physicsBody?.contactTestBitMask = idCanos
185                 bird.physicsBody?.collisionBitMask = idVao
186             }else{
187                 if !terminou{
188                     bird.physicsBody?.velocity = CGVectorMake(0, 0)
189                     bird.physicsBody?.applyImpulse(CGVectorMake(0, 150))
190                     self.runAction(somVoar)
191                     self.criaParticulas()
192                 }
193             }
194         }
195     }
```