



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 6

SWIFT STUDENT CHALLENGE (AKA ASSIGNMENT 4)

SWIFT STUDENT CHALLENGE

SUBTITLE

- Experienced within 3 minutes
- Your Swift playground must be built with and run on Swift Playgrounds 3.3 on iPadOS 13.4.1, Swift Playgrounds 3.3 on macOS 10.15.4, or Xcode 11.4.1 on macOS 10.15.4. If it runs on iPadOS, it must be optimized to display properly on all models of iPad Pro.
- Submissions will be judged offline. Your Swift playground should not rely on a network connection and any resources used in your Swift playground should be included locally in your .zip file.



Swift Student Challenge

Showcase your love of coding by creating an incredible Swift playground on the platform of your choice. Winners will receive an exclusive WWDC20 jacket and pin set. This challenge is open to students around the world.

[Apply now >](#)

Applying

Build your Swift playground, answer a few written prompts, provide documentation, and submit.

To be eligible for the challenge, you must:

- Be 13 years of age or older, or the equivalent minimum age in the relevant jurisdiction (for example, 16 years of age in the European Union);
- Be registered for free with Apple as an Apple developer or be a member of an educational institution.

SWIFT STUDENT CHALLENGE

- Your "story" is just as important as your code

Completing Your Submission

1. Tell us about yourself.

Sign in to the application form with the Apple ID associated with your developer account. If you're under 18 years old, you will also be asked to enter contact information for your parent or legal guardian.

You'll have the option to add details about your background and development experience. This will not influence the selection process. Tell us about any apps you have on the App Store created entirely by you as an individual, in 500 words or less. If you're 18 years of age or older and wish to share your résumé or CV with other groups at Apple, upload a PDF.

2. Provide school information.

Upload your most recent class schedule or other most recent proof of enrollment (PDF, PNG, or JPG) and the contact information for your educational supervisor. Documentation is accepted in all languages.

3. Upload and describe your Swift playground.

Upload your Swift playground from your Mac. Tell us about the features and technologies that you used in your Swift playground, in 500 words or less.

4. Provide optional information.

If you've shared or considered sharing your coding knowledge and enthusiasm for computer science with others, let us know in 500 words or less.

Dates:

- Submissions open on Tuesday, May 5, 2020 at 9:00 a.m. PDT.
- Deadline for submissions is Sunday, May 17, 2020 at 11:59 p.m. PDT.
- Applicants can view their status starting Tuesday, June 16, 2020.

SWIFT STUDENT CHALLENGE

- Your "story" is just as important as your code

Completing Your Submission

1. Tell us about yourself.

Sign in to the application form with the Apple ID associated with your developer account. If you're under 18 years old, you will also be asked to enter contact information for your parent or legal guardian.

You'll have the option to add details about your background and development experience. This will not influence the selection process. Tell us about any apps you have on the App Store created entirely by you as an individual, in 500 words or less. If you're 18 years of age or older and wish to share your résumé or CV with other groups at Apple, upload a PDF.

2. Provide school information.

Upload your most recent class schedule or other most recent proof of enrollment (PDF, PNG, or JPG) and the contact information for your educational supervisor. Documentation is accepted in all languages.

3. Upload and describe your Swift playground.

Upload your Swift playground from your Mac. Tell us about the features and technologies that you used in your Swift playground, in 500 words or less.

4. Provide optional information.

If you've shared or considered sharing your coding knowledge and enthusiasm for computer science with others, let us know in 500 words or less.

Dates:

- Submissions open on Tuesday, May 5, 2020 at 9:00 a.m. PDT.
- Deadline for submissions is Sunday, May 17, 2020 at 11:59 p.m. PDT.
- Applicants can view their status starting Tuesday, June 16, 2020.

SWIFT STUDENT CHALLENGE

WWDC 2019 Scholarship Submissions

Student submissions for the WWDC 2019 Scholarship



List of student submissions for the WWDC 2019 scholarship.

Check out the [YouTube Playlist](#) to watch the submissions.

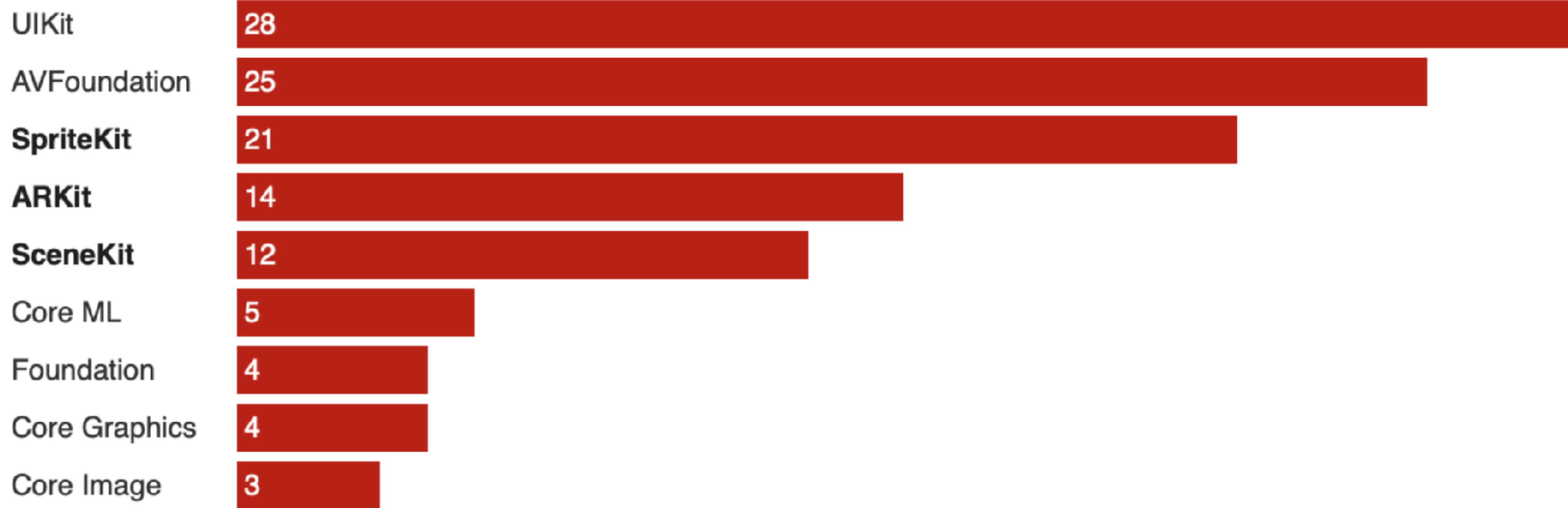
To add your own project below, just [edit](#) this file and submit a pull request! Please read the [PLEASE READ](#) section before submitting.

Name	Source	Videos	Technologies
Aaron Cheung	GitHub	Youtube	ARKit, SceneKit, AVFoundation, Foundation
Abram Situmorang	GitHub		MetalKit
Adam Giesinger	GitHub		CoreML, Vision, UIKit
Adrian Kashivskyy	GitHub		AVFoundation, CoreImage, CoreLocation, Metal

SWIFT STUDENT CHALLENGE

WWDC 2019 Scholarship Winners: AR & Animation

These are the technologies used most by scholarship winners for WWDC 2019.



SWIFT STUDENT CHALLENGE

- My thoughts for entries
 - Use SwiftUI somehow
 - SpriteKit (but not necessarily a "game")



Swift Student Challenge

Share your love of coding by creating an incredible Swift playground or app. Winners will receive an exclusive WWDC20 jacket and pin set. The challenge is open to students around the world.

[Apply now >](#)

Build your Swift playground, answer a few written prompts, provide documentation, and submit.

To be eligible for the challenge, you must:

- Be 13 years of age or older, or the equivalent minimum age in the jurisdiction (for example, 16 years of age in the European Union)
- Be registered for free with Apple as an Apple developer or be a student

SWIFT STUDENT CHALLENGE

- Assignment 4

- Create something for the Swift Student Challenge (you don't have to submit)
- Use SpriteKit (doesn't have to be the main part of your playground)



Swift Student Challenge

Share your love of coding by creating an incredible Swift playground or game. Winners will receive an exclusive WWDC20 jacket and pin set. The challenge is open to students around the world.

[Apply now >](#)

Build your Swift playground, answer a few written prompts, provide documentation, and submit.

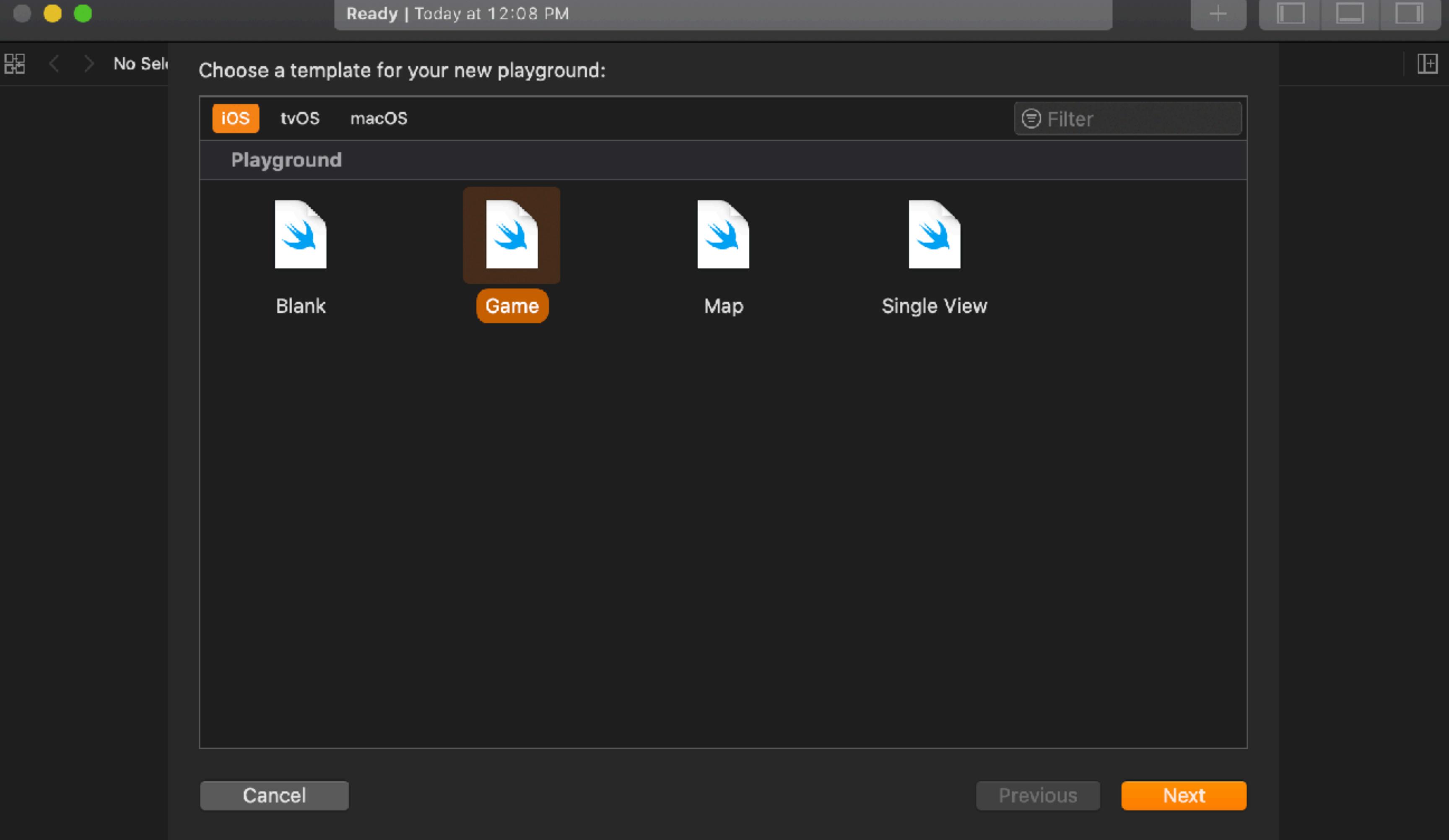
To be eligible for the challenge, you must:

- Be 13 years of age or older, or the equivalent minimum age in the jurisdiction (for example, 16 years of age in the European Union)
- Be registered for free with Apple as an Apple developer or be a student

PLAYGROUNDS

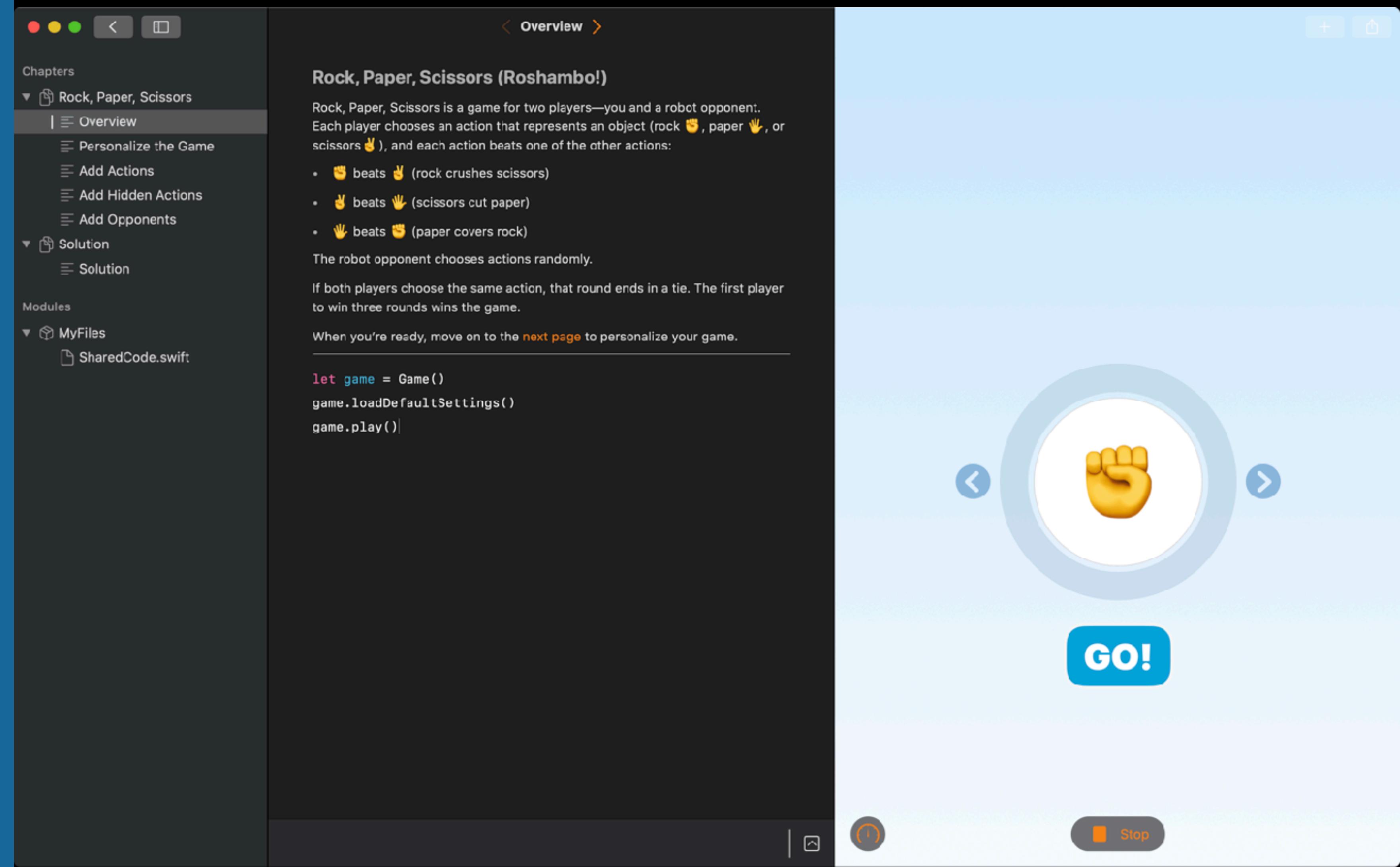
PLAYGROUNDS

- Xcode
Playground
 - Templates for
SpriteKit



PLAYGROUNDS

- Playground Books
 - Unizp Apple's samples for best overview
 - Minimal documentation



PLAYGROUNDS

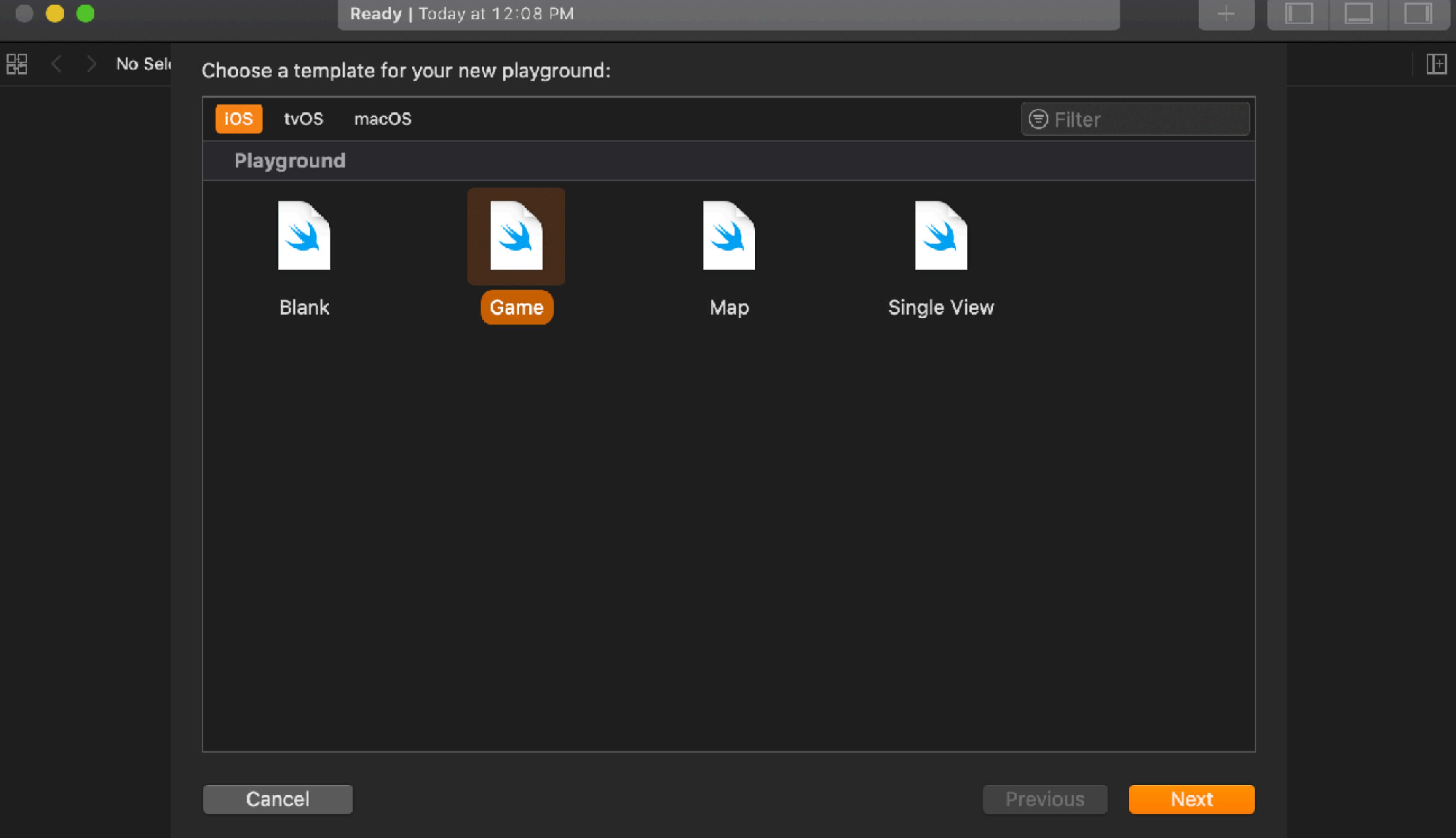
- Swift Playgrounds
- Mac
- iPad
- "Starting Points" templates for applications

The screenshot shows the Swift Playgrounds app store interface. At the top right is a "More Playgrounds" button. The main content area is divided into several sections:

- Learn to Code**: Three items:
 - Learn to Code 1**: Fundamentals of Swift, Swift 5.1 Edition. Includes a yellow cartoon character icon and a "GET" button.
 - Learn to Code 2**: Beyond the Basics, Swift 5.1 Edition. Includes a green cartoon character icon and a "GET" button.
 - Blu's Adventure**: Exploring graphics, coordinates, and touch events, Swift 5.1 Edition. Includes a blue cartoon character icon and a "GET" button.
- Challenges**: A grid of six items:
 - Brick Breaker**: Advanced, Swift 5.1 Edition. NEW badge. Includes a brick wall icon and a "GET" button.
 - Code Machine**: Beginner, Swift 5.1 Edition. Includes a code editor icon and a "GET" button.
 - Cipher**: Intermediate, Swift 5.1 Edition. Includes a lock icon and a "GET" button.
 - Sonic Workshop**: Intermediate, Swift 5.1 Edition. Includes a circuit board icon and a "GET" button.
 - Hello, Byte**: Beginner, Swift 5.1 Edition. Includes a robot head icon and a "GET" button.
 - Battleship**: Intermediate, Swift 5.1 Edition. Includes a battleship icon and a "GET" button.
- Starting Points**: A grid of four items:
 - Blank**: Swift 5.1 Edition. Includes a blank document icon and a "GET" button.
 - Shapes**: Swift 5.1 Edition. Includes a colorful geometric shapes icon and a "GET" button.
 - Graphing**: Swift 5.1 Edition. Includes a graph icon and a "GET" button.
 - Sonic Create**: Swift 5.1 Edition. Includes a city skyline icon and a "GET" button.
 - Answers**: Swift 5.1 Edition. Includes a bar chart icon and a "GET" button.
 - Puzzle World**: Swift 5.1 Edition. Includes a jigsaw puzzle icon and a "GET" button.
- Code with the Foos** [THIRD PARTY]:
 - codeSpark Academy Coding Fundamentals
 - Connect codeSpark's custom icon...
 - GET** button
- Playgrounds by Daniel Budd** [THIRD PARTY]:
 - Geometry**: Beginner, Geometry with Swift Playgrounds. Includes a protractor icon and a "GET" button.
 - Geometry Template**: Beginner, Students can create their own Geo... Includes a protractor icon and a "GET" button.
 - Coding with mBot**: Beginner, mBot with Swift Playgrounds. Includes a mBot icon and a "GET" button.
- Hacking with Swift** [THIRD PARTY]:
 - Includes a "GET" button.

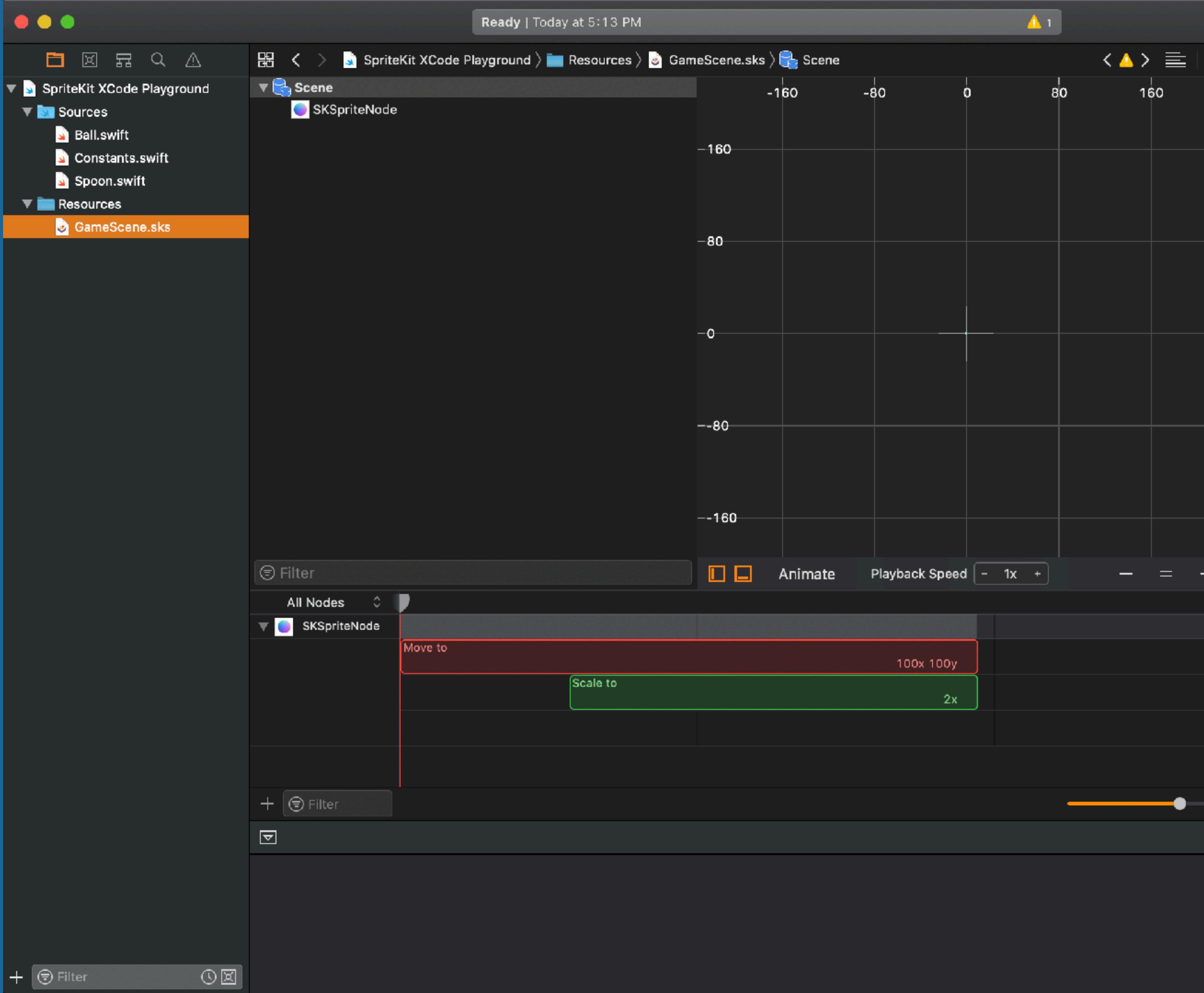
PLAYGROUNDS

- Xcode Playground
 - Easier to develop
- Swift Playgrounds App (Mac and iPad)
 - Sensors



PLAYGROUNDS

- Do as much work in Xcode as possible
 - Resources
 - Sources



Sprite Kit

SPRITE KIT

SUBTITLE

- A framework for 2D game development
 - Sprites, shapes, particles
 - Animation and physics
 - Audio, video, visual effects



SPRITE KIT

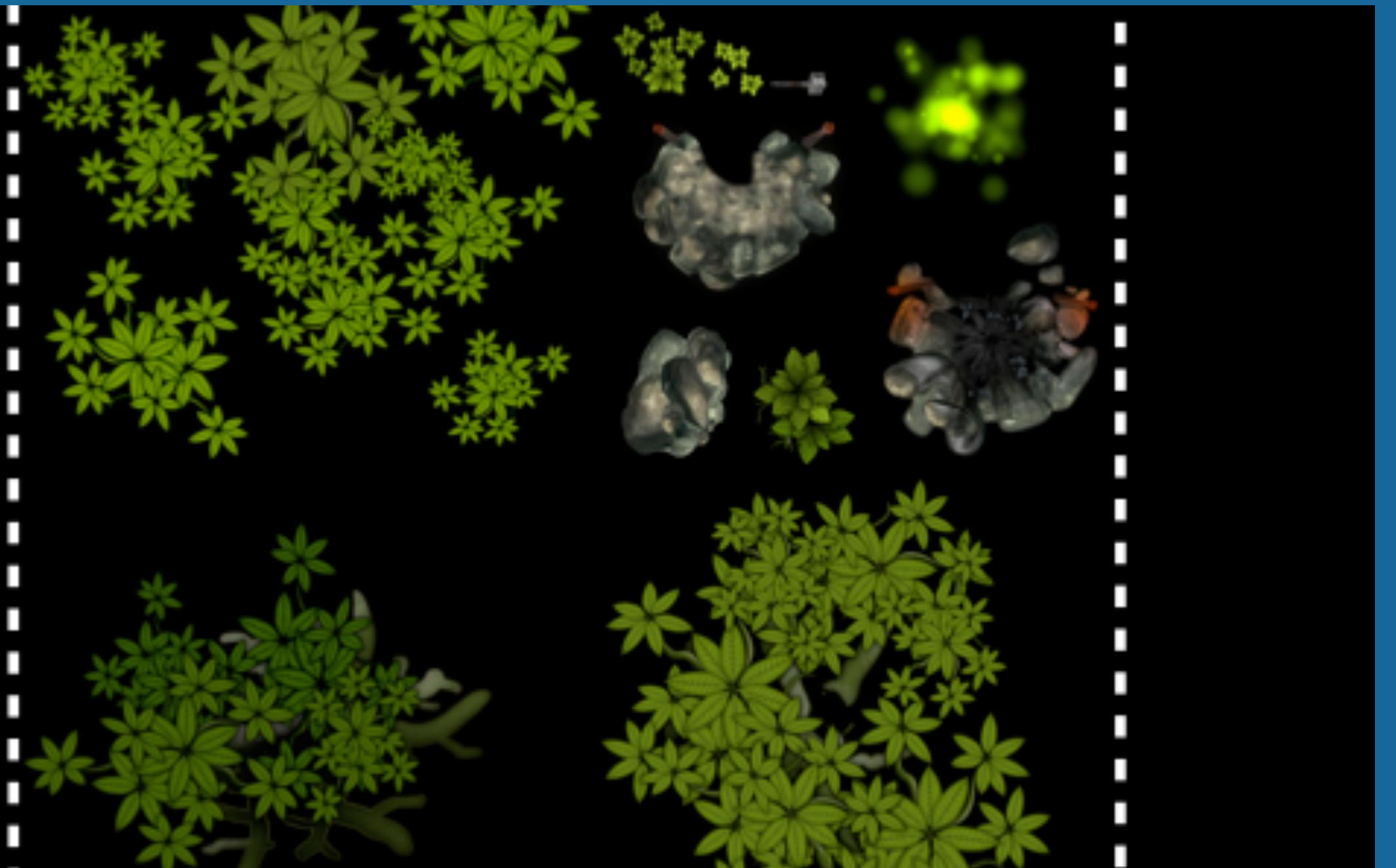
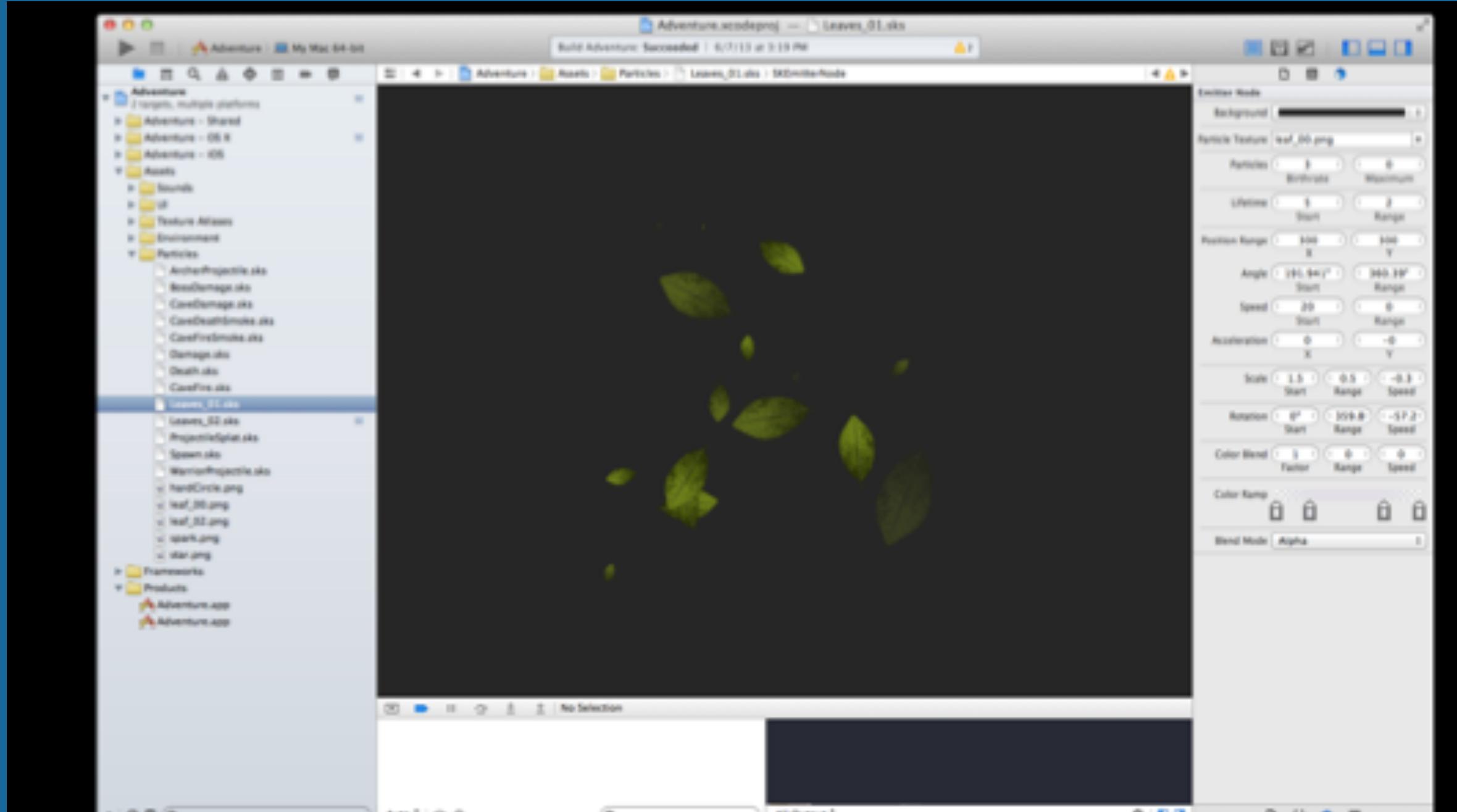


- Cross platform for iOS, tvOS and OS X

SPRITE KIT

- Alternatives
 - Cocos2D
 - Box2D (used in SpriteKit)
 - Unity
- Key Apple motivations
 - Efficiency with hardware
 - Track with iOS developments
 - Support 64bit
 - 3D touch
 - ...

SPRITE KIT



- SpriteKit consists of a framework and tools
 - Particle editor
 - Texture atlas generator
 - Level editor

SPRITE KIT



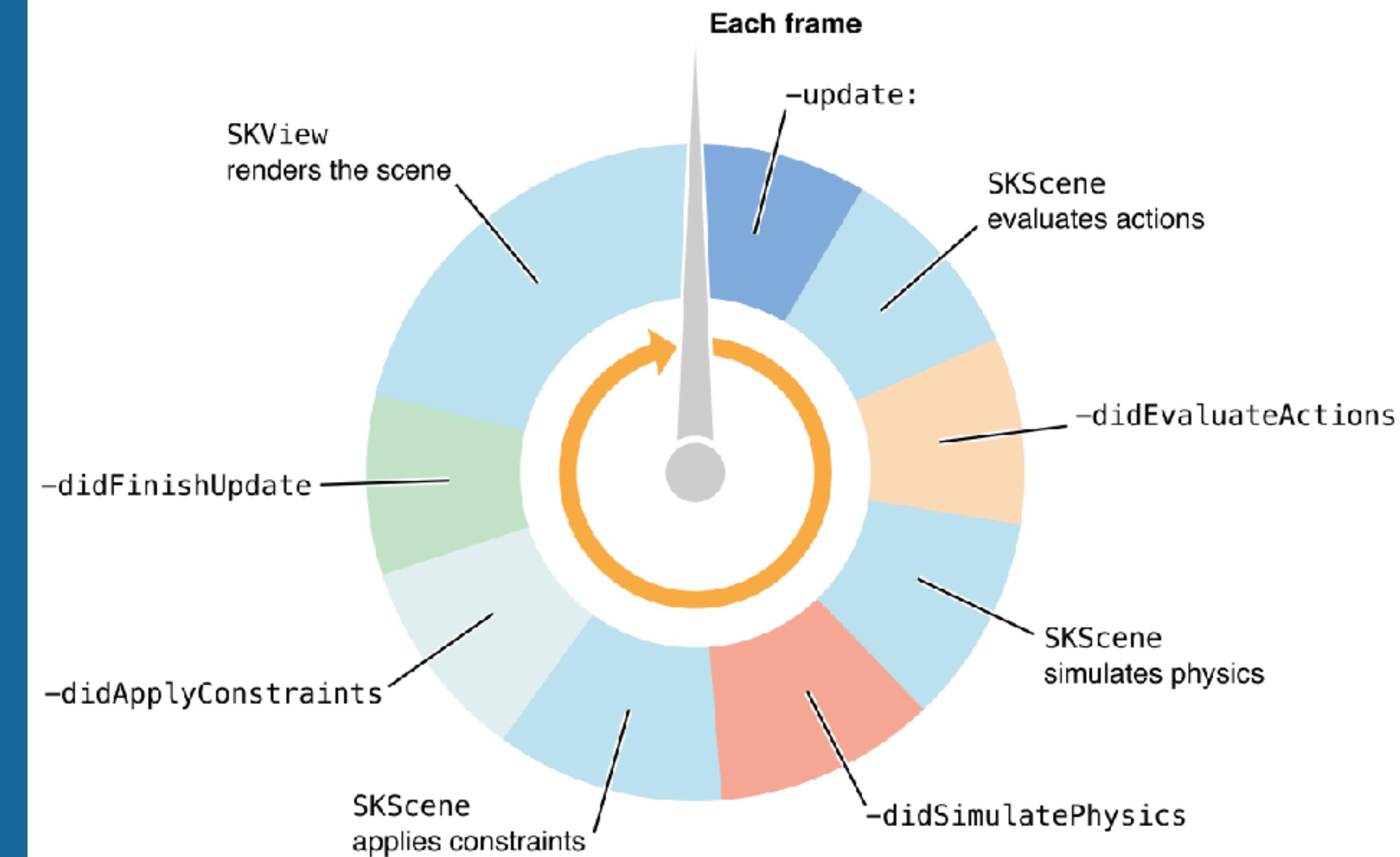
- Integrated live editor in Xcode

**SPRITE KIT IN A
NUTSHELL**

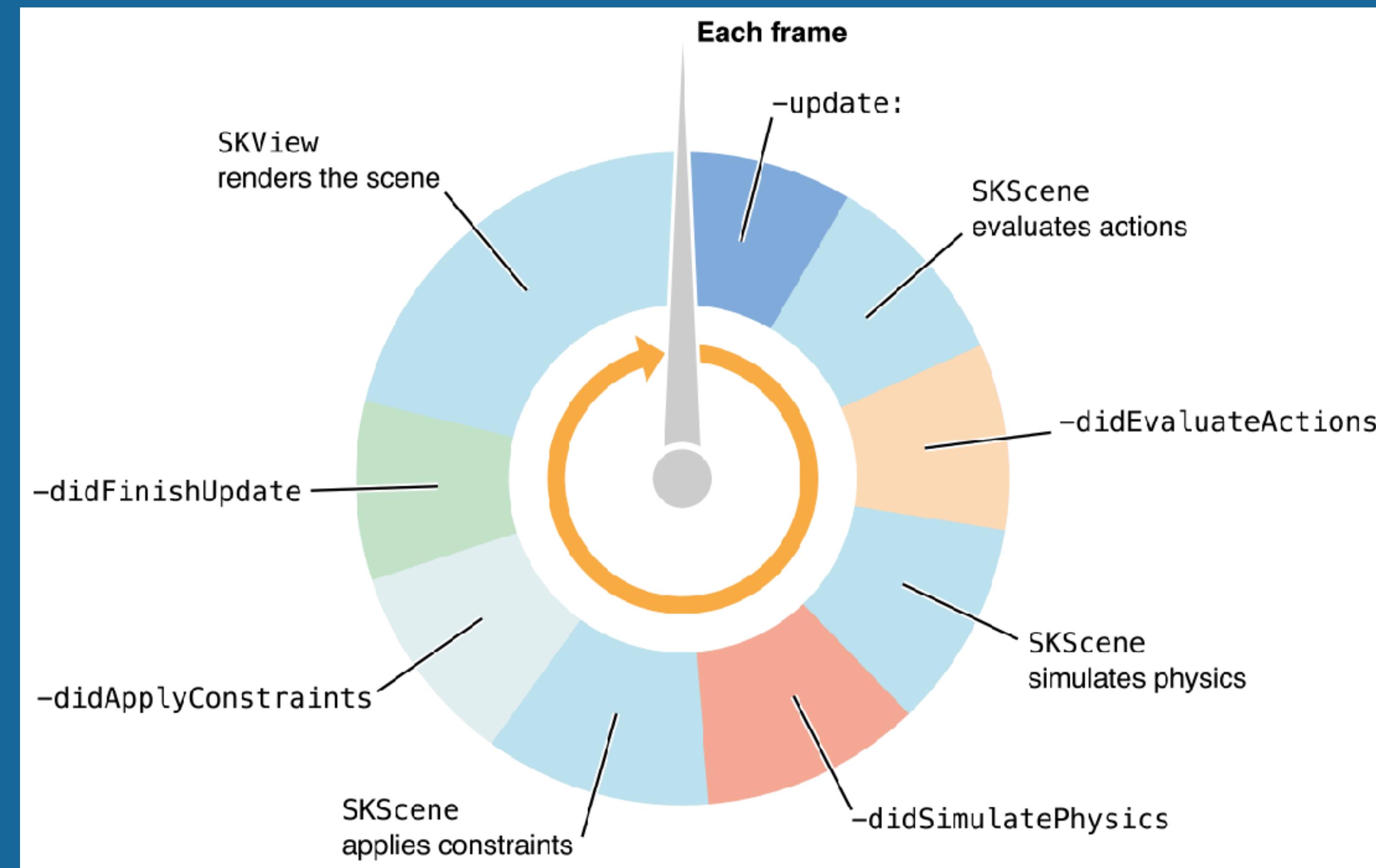
SPRITE KIT IN A NUTSHELL

SUBTITLE

- SpriteKit provides a graphics rendering and animation infrastructure to animate arbitrary textured images, or sprites
- SpriteKit uses a traditional rendering loop where the contents of each frame are processed before the frame is rendered



SPRITE KIT IN A NUTSHELL



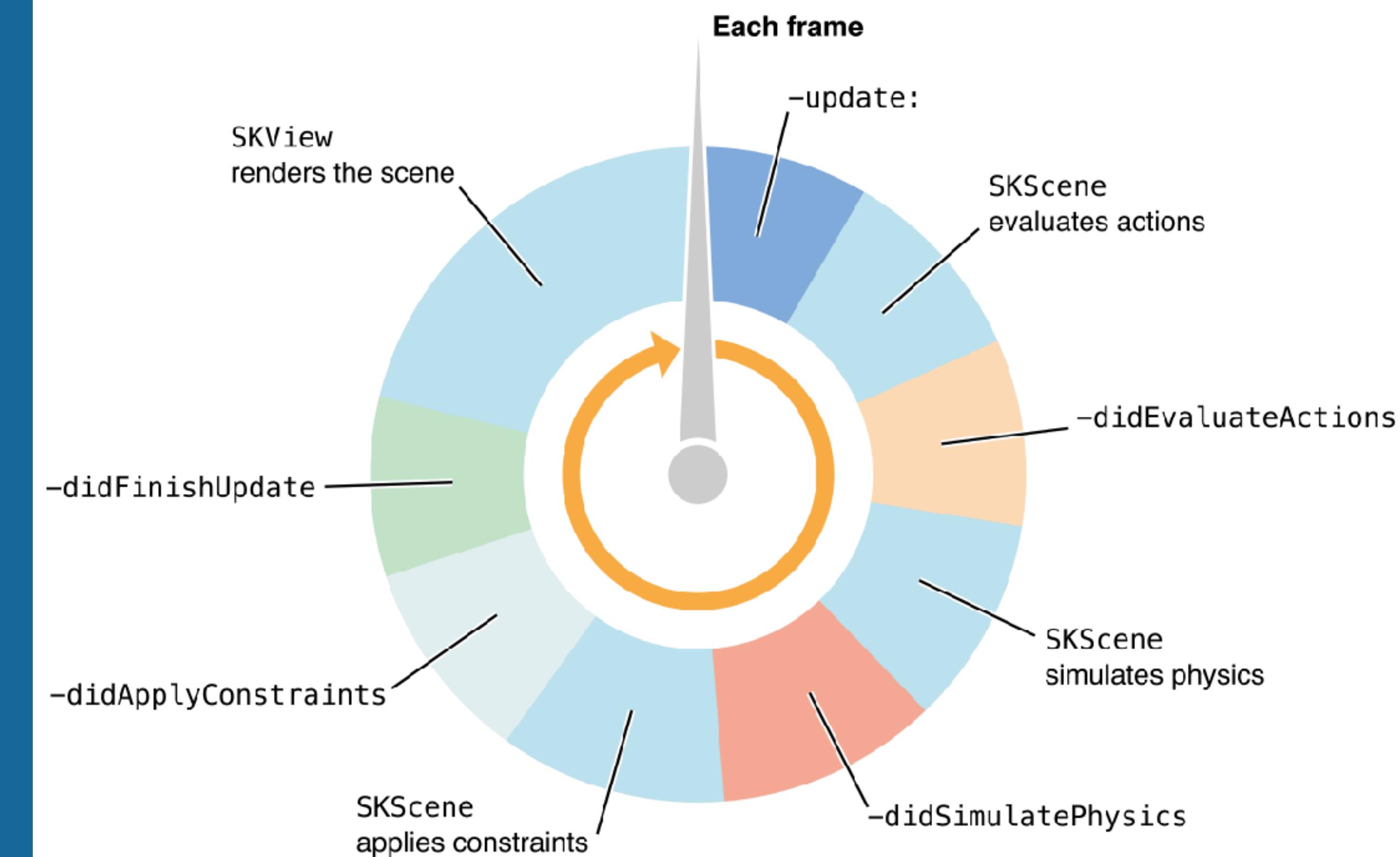
60 Frames
per second
is the goal

- Your game determines the contents of the scene and how those contents change in each frame

SPRITE KIT IN A NUTSHELL

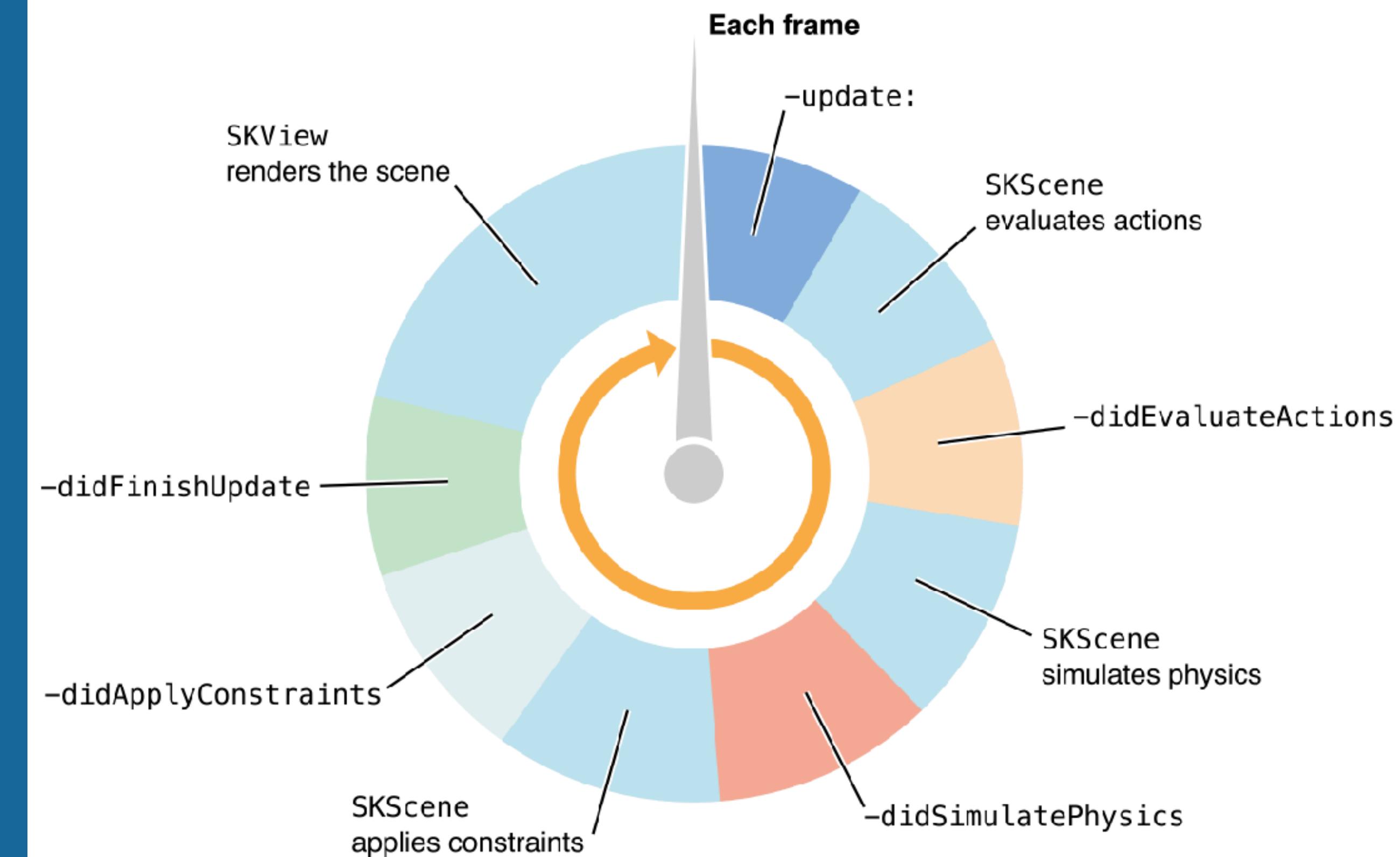
SUBTITLE

- SpriteKit is optimized
 - Render frames of animation using the graphics hardware
 - Positions of sprites can be changed arbitrarily in each frame of animation



SPRITE KIT IN A NUTSHELL

- Provides other functionality
 - Physics simulation
 - Sounds effects
 - Playback
 - Volume adjustments
 - Special effects
 - Particle emitters
 - Lighting/camera positioning
 - Video effects
 - Crop, overlay



SPRITE KIT IN A NUTSHELL

- Sprite content is drawn by presenting scenes (`SKScene`) in a sprite view (`SKView`)
 - You switch between scenes using `SKTransition`
- Physics bodies (`SKPhysicsBody`) and joints (`SKPhysicsJoint`) simulate physics in your scene



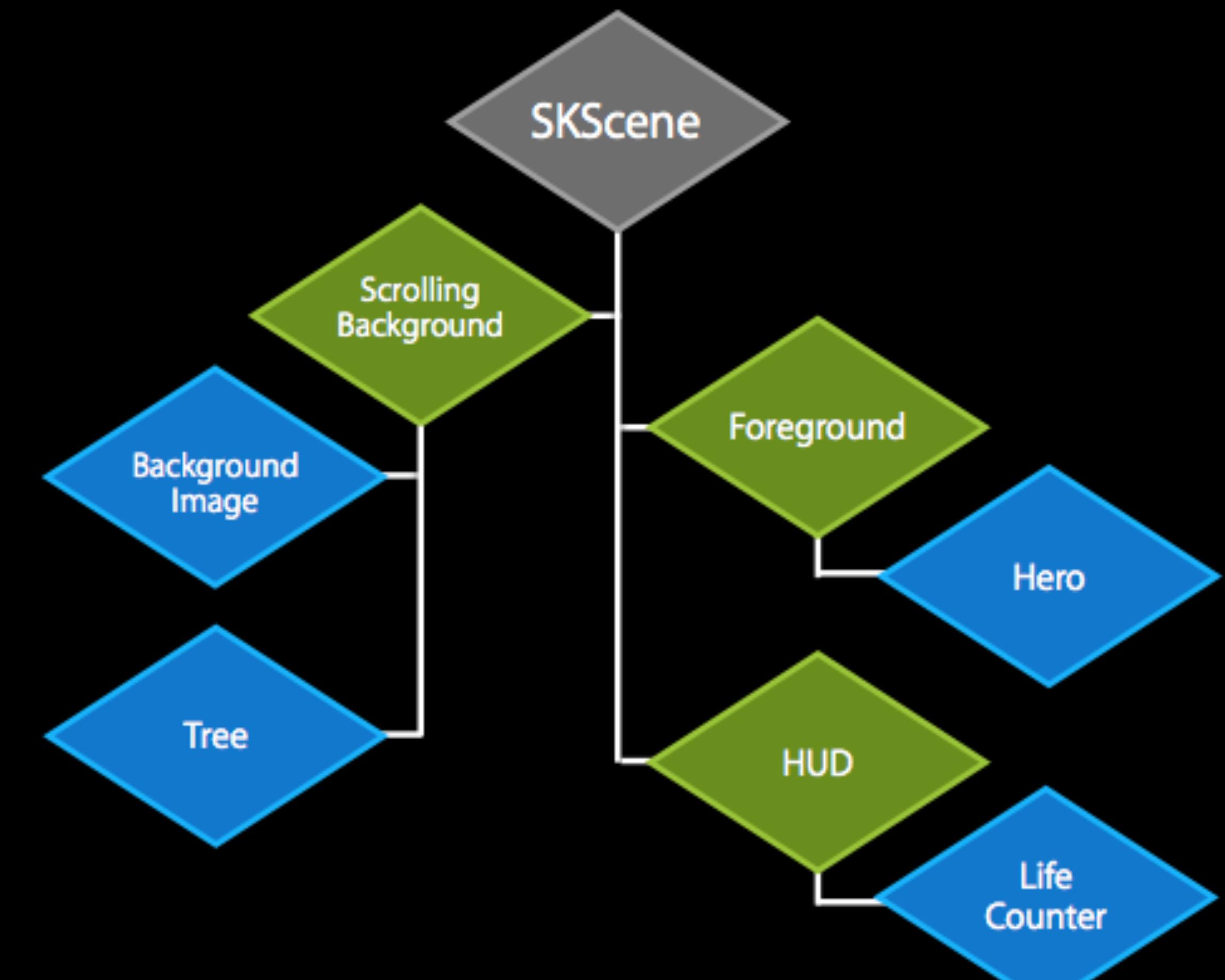
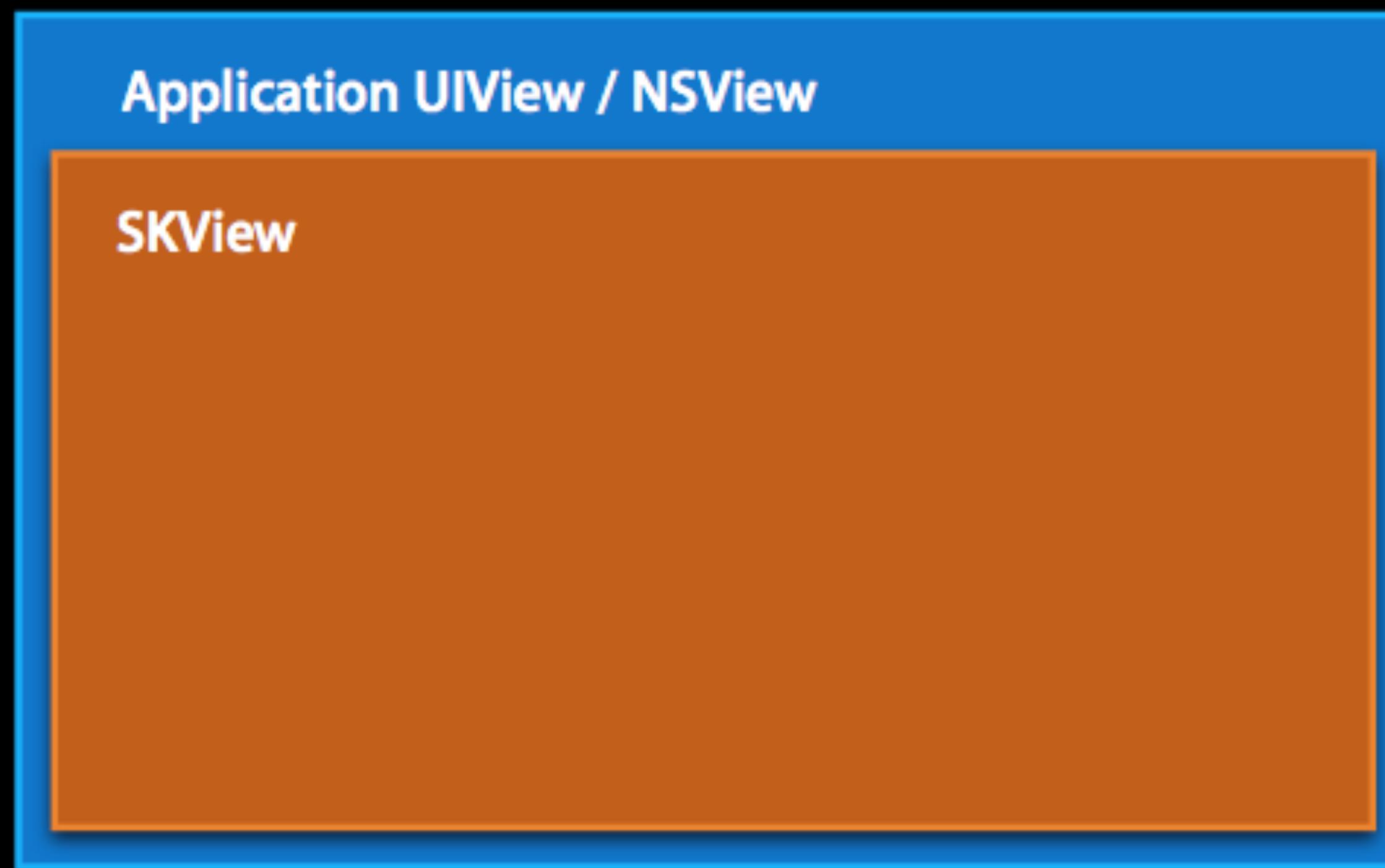
SPRITE KIT IN A NUTSHELL

- A node tree (`SKNodes`) defines what appears in a scene
 - Specialized nodes (eg. `SKEffectNode`, `SKSpriteNode`)
 - Textures are images used to efficiently render sprites
- Nodes execute actions (`SKAction`) to animate content



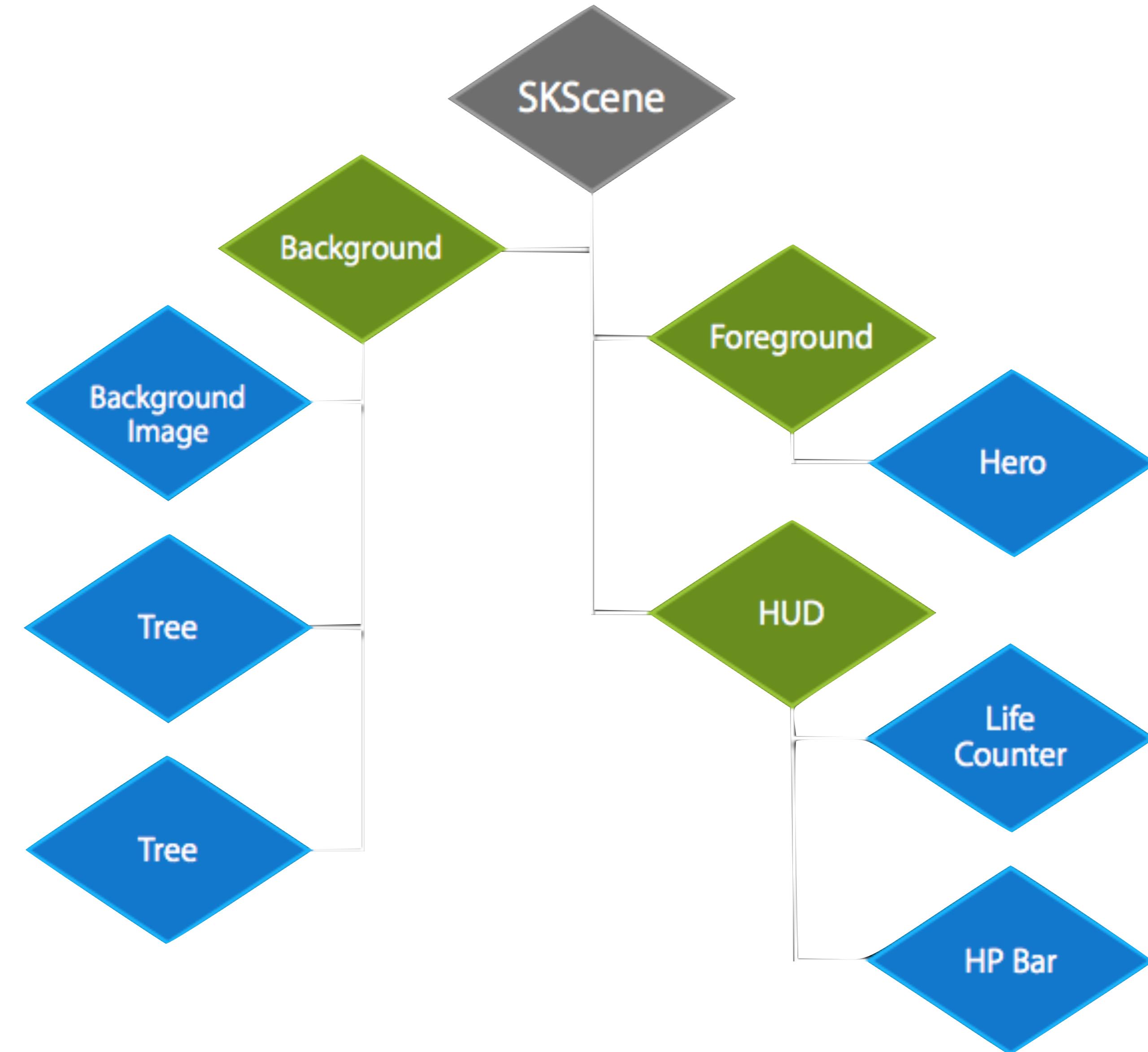
VIEW AND SCENES

VIEW AND SCENES



VIEW AND SCENES

- Scenes are constructed from nodes
 - Nodes can be visual
 - Some can be used for organization



VIEW AND SCENES

SUBTITLE

- A game may be constructed from several scenes
 - Menu
 - Leaderboard
 - Game
 - Levels
 - Achievements
 - etc.



VIEW AND SCENES

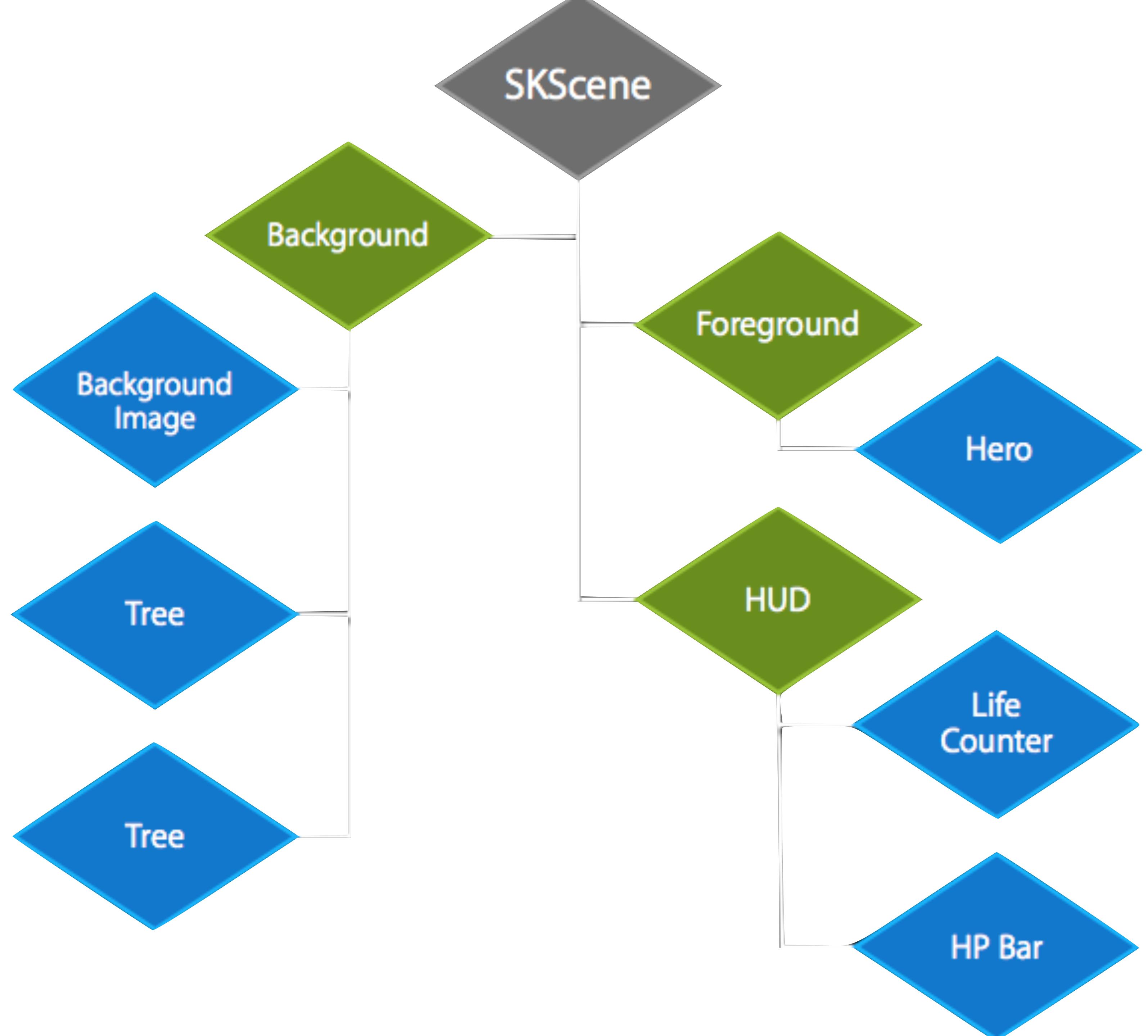


NODES

VIEW AND SCENES

- Each scene is constructed of a combination of nodes
 - Baseclass is SKNode
- Add a node

```
self.addChild()
```



SPRITES AND PARTICLES

The base class

- Used for grouping in node tree

```
@property SKNode* parent;  
@property NSArray* children;
```

- Used for positioning itself and children

```
@property CGPoint position;           // relative to parent  
@property CGFloat zRotation;          // radians  
@property CGFloat xScale, yScale;     // scaling
```

- Can control visibility

```
@property CGFloat alpha;  
@property BOOL hidden;
```

- Can run actions

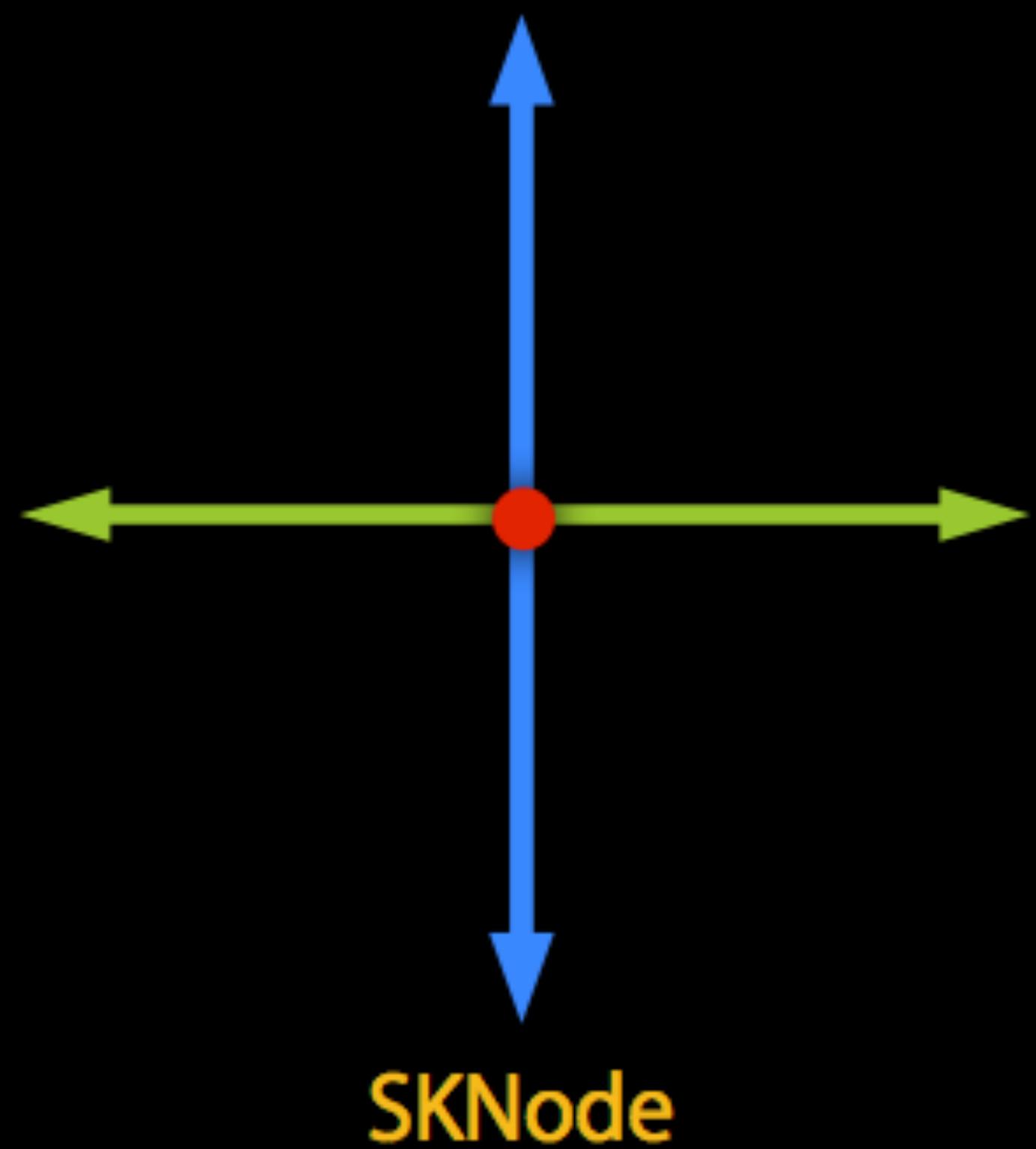


Image via Apple

NODES

```
SKSpriteNode *ship = [SKSpriteNode spriteNodeWithImageNamed:@"Spaceship"];
[ship setAnchorPoint:(CGPoint){0.5, 0.5}];
ship.position = CGPointMake(40.0,30.0);
[scene addChild:ship]
```

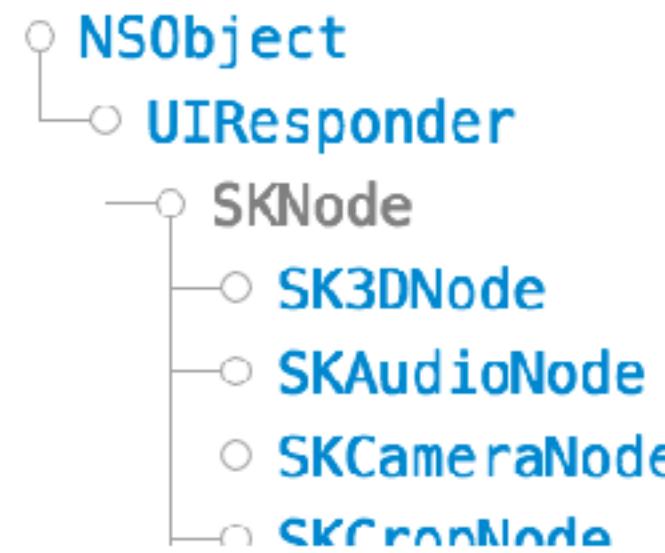
- Creates a sprite matching the image size
- Centers sprite on image center (0.5, 0.5)
- Positions sprite at (40, 30) relative to its parent
- Follows **UIImage -imageNamed:** behavior

NODES

- Specialized nodes for different appearances and behaviors

SKNode

Inherits From



Conforms To

`CVarArgType`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCoding`
`NSCopying`

Import Statement

```
SWIFT  
import SpriteKit
```

Availability

Available in iOS 7.0 and later

The `SKNode` class is the fundamental building block of most SpriteKit content. The `SKNode` class doesn't draw any visual content. Its primary role is to provide baseline behavior that the other node classes use. All visual elements in a SpriteKit-based game are drawn using predefined `SKNode` subclasses.

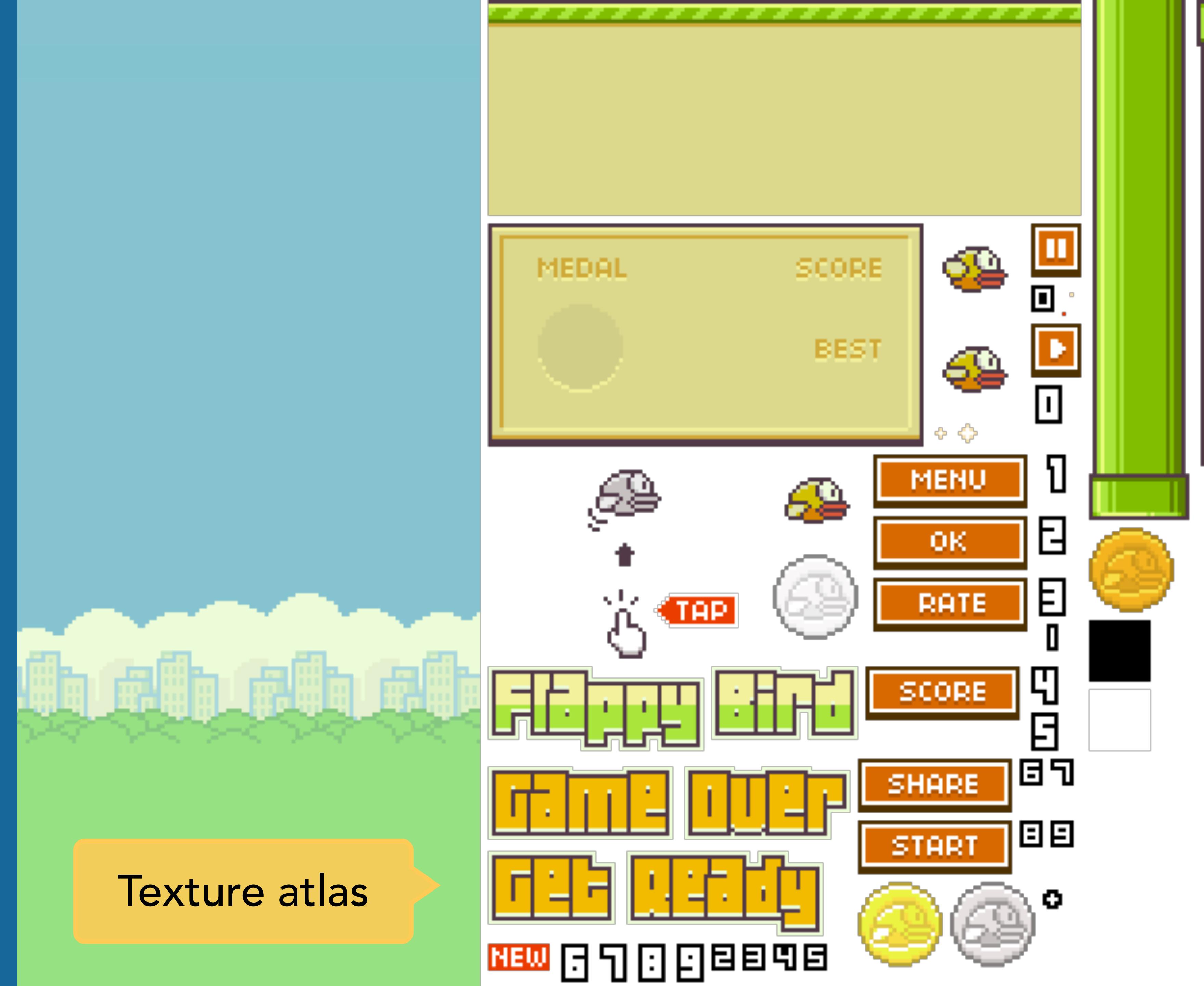
Table 1 describes the node subclasses provided in SpriteKit.

Table 1 Node subclasses

Class	Description
<code>SKSpriteNode</code>	A node that draws a textured sprite.
<code>SKVideoNode</code>	A node that plays video content.
<code>SKLabelNode</code>	A node that renders a text string.
<code>SKShapeNode</code>	A node that renders a shape based on a Core Graphics path.
<code>SKEmitterNode</code>	A node that creates and renders particles.

NODES

- SKSpriteNode
 - Draws a sprite with a texture
 - Can just be a solid color
 - Game images
 - Characters or objects in a game
 - Has an explicit size



NODES

- SKLabelNode
 - Used to display text within a game
- Examples
 - Title
 - Menu options
 - Scores
 - Time
 - "Game Over"



NODES

```
import Foundation
import SpriteKit

class Ball: SKShapeNode {

    let number: Int = Int(arc4random_uniform(10) + 1)
    var radius: CGFloat!
    var circumference: CGFloat!

    override init() {
        super.init()
        circumference = CGFloat(number * 10)
        radius = CGFloat(circumference / 2)
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    convenience init(position: CGPoint) {
        self.init()

        self.path = CGPathCreateWithEllipseInRect(CGRectMake(-radius, -radius, circumference, circumference), nil)
        self.position = position
        self.fillColor = UIColor.blueColor()
        self.strokeColor = UIColor.blueColor()
    }
}
```

Custom
SKShapeNode

NODES

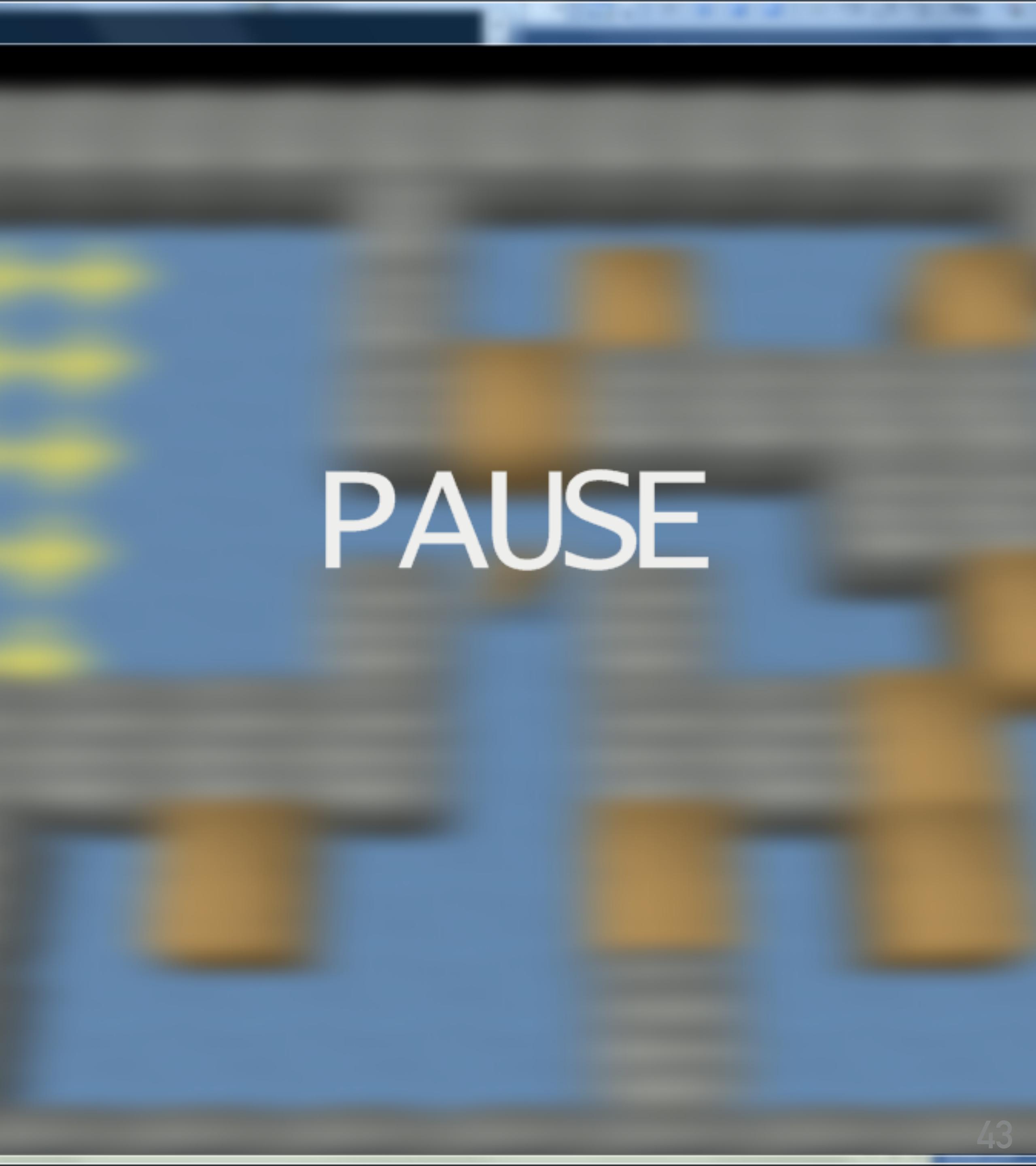
- SKEmitterNode
 - The node responsible for managing and displaying particle emitter based special effects
 - Full featured 2D particle system
 - Standard `startValue` and `speed`
 - Advanced keyframe sequence controls



NODES

SUBTITLE

- SKEffectNode
 - Allows Core Image filter effects to be applied to child nodes.
 - A sepia filter effect, for example, could be applied to all child nodes of an SKEffectNode.
- SKCropNode
 - Allows the pixels in a node to be cropped subject to a specified mask



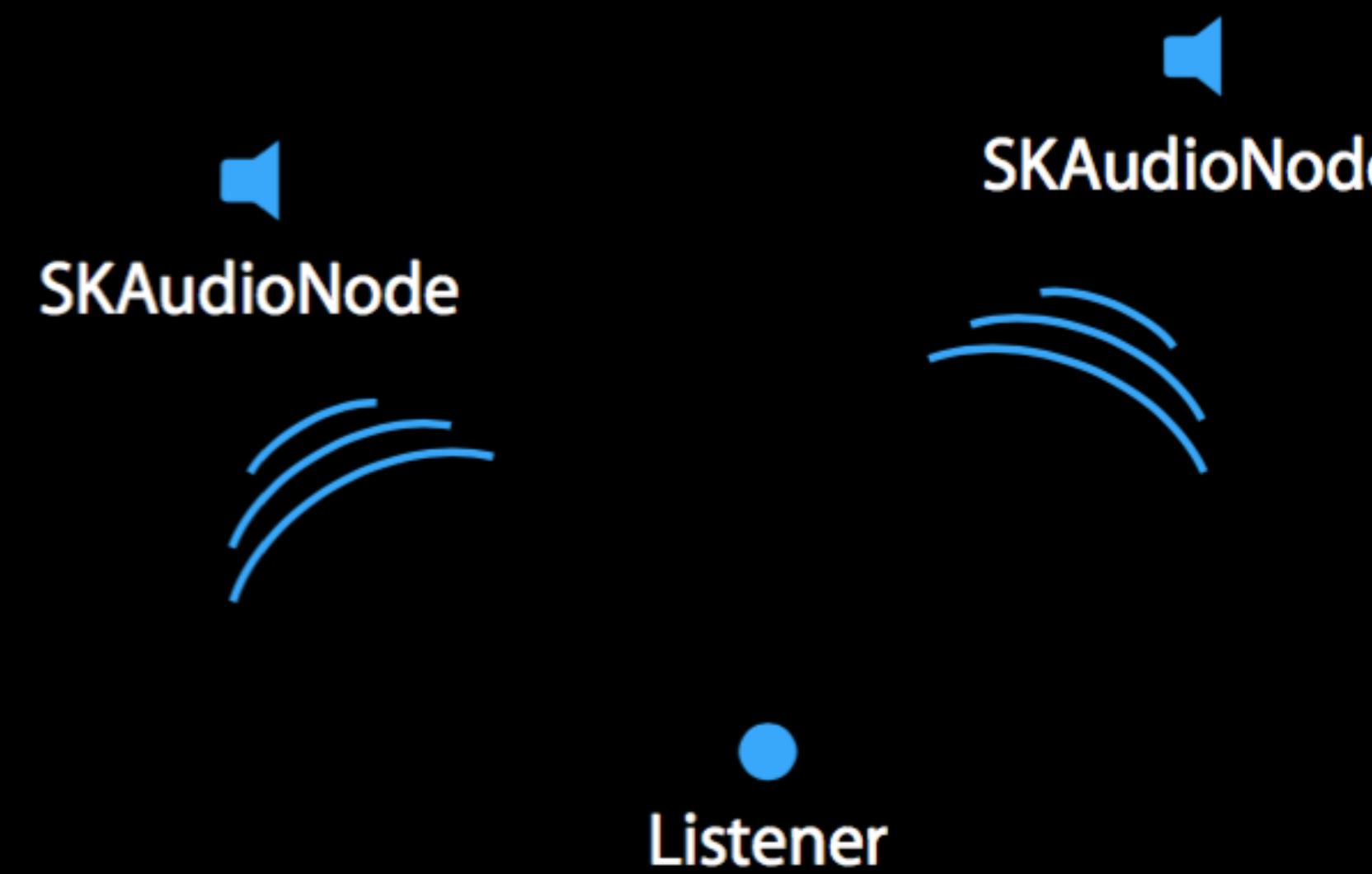
NODES

- SKLightNode
 - Adds a light sources to a scene
 - Casts of shadows when the light falls on other nodes in the same scene
- SK3DNode
 - Allows 3D assets to be embedded into 2D Sprite Kit games

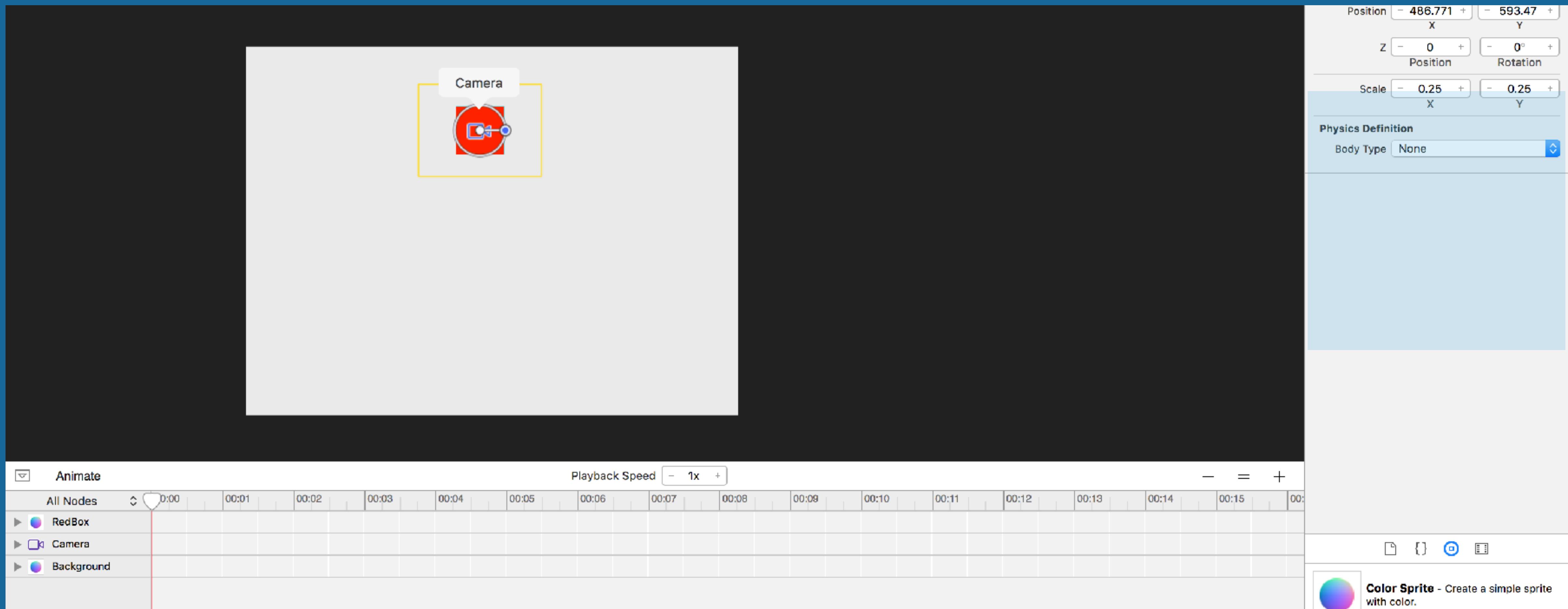


NODES

- SKAudioNode
 - Leveraging AVAudioEngine
 - Position calculated via node position
 - Listener node property on SKScene
 - Create with filename or URL



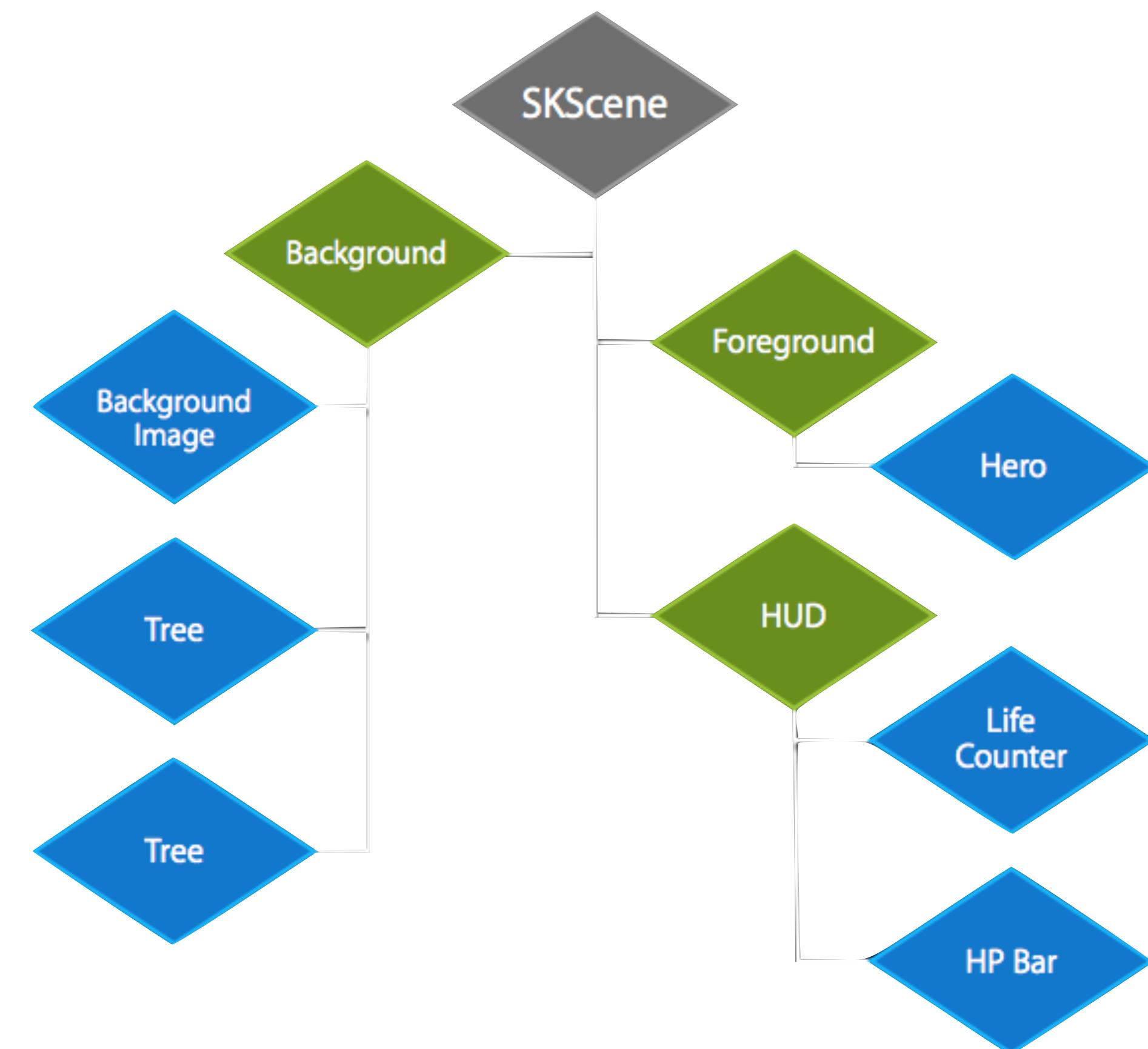
NODES



- SKCamera Node

NODES

- SKNode is often used for grouping nodes
 - Transform node subtree
 - Remove node subtree
- Children nodes inherit properties of parents



VIEW AND SCENES

- Special syntax for searching the node tree

Syntax	Description
/	When placed at the start of the search string, this indicates that the search should be performed on the tree's root node. When placed anywhere but the start of the search string, this indicates that the search should move to the node's children.
//	When placed at the start of the search string, this specifies that the search should begin at the root node and be performed recursively across the entire node tree. Otherwise, it performs a recursive search from its current position.
.	Refers to the current node.
..	The search should move up to the node's parent.
*	The search matches zero or more characters.
[characters delimited by commas or dashes]	The search matches any of the characters contained inside the brackets.
alphanumeric characters	The search matches only the specified characters.

VIEW AND SCENES

```
import SpriteKit
import GameplayKit

class GameScene: SKScene {

    private var label : SKLabelNode?
    private var spinnyNode : SKShapeNode?

    override func didMove(to view: SKView) {

        // Get label node from scene and store it for use later
        self.label = self.childNode(withName: "//helloLabel") as? SKLabelNode
        if let label = self.label {
            label.alpha = 0.0
            label.run(SKAction.fadeIn(withDuration: 2.0))
        }

        // Create shape node to use during mouse interaction
        let w = (self.size.width + self.size.height) * 0.05
        self.spinnyNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: w), cornerRadius: w * 0.3)

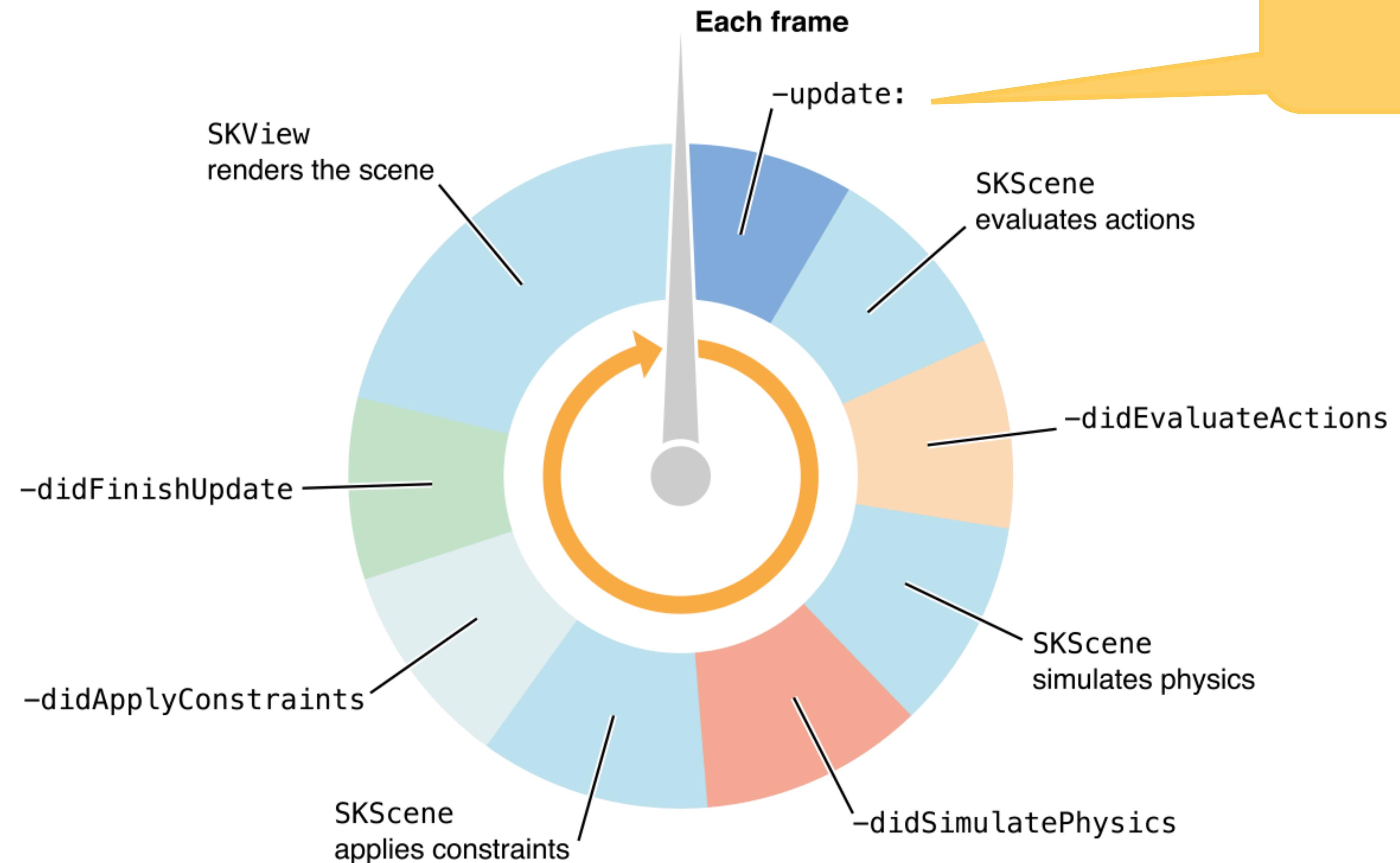
        if let spinnyNode = self.spinnyNode {
            spinnyNode.lineWidth = 2.5

            spinnyNode.run(SKAction.repeatForever(SKAction.rotate(byAngle: CGFloat(Double.pi), duration: 1)))
            spinnyNode.run(SKAction.sequence([SKAction.wait(forDuration: 5), SKAction.removeFromParent()]))
        }
    }
}
```

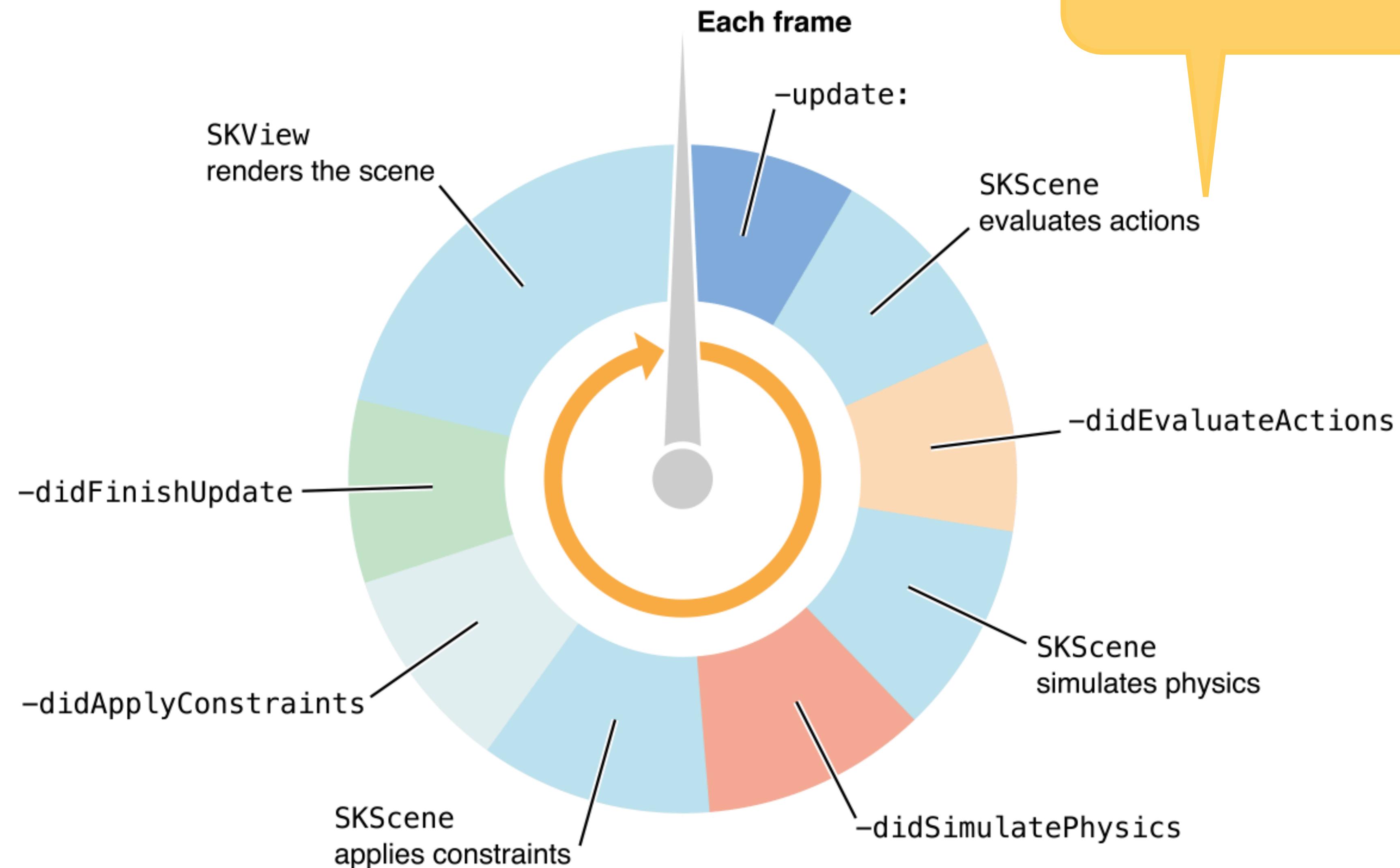
Label Node

GAME RENDERING LOOP

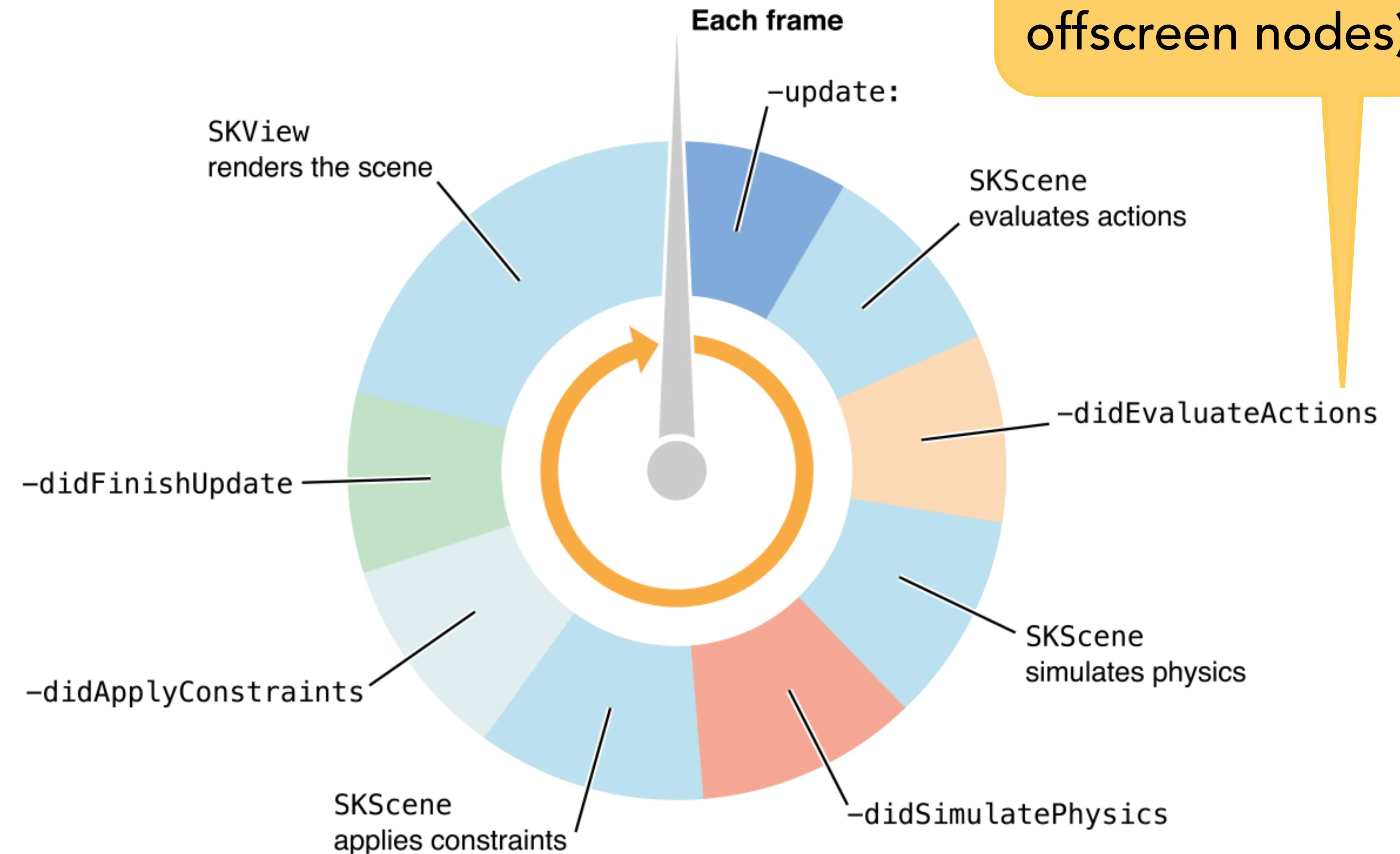
GAME RENDERING LOOP



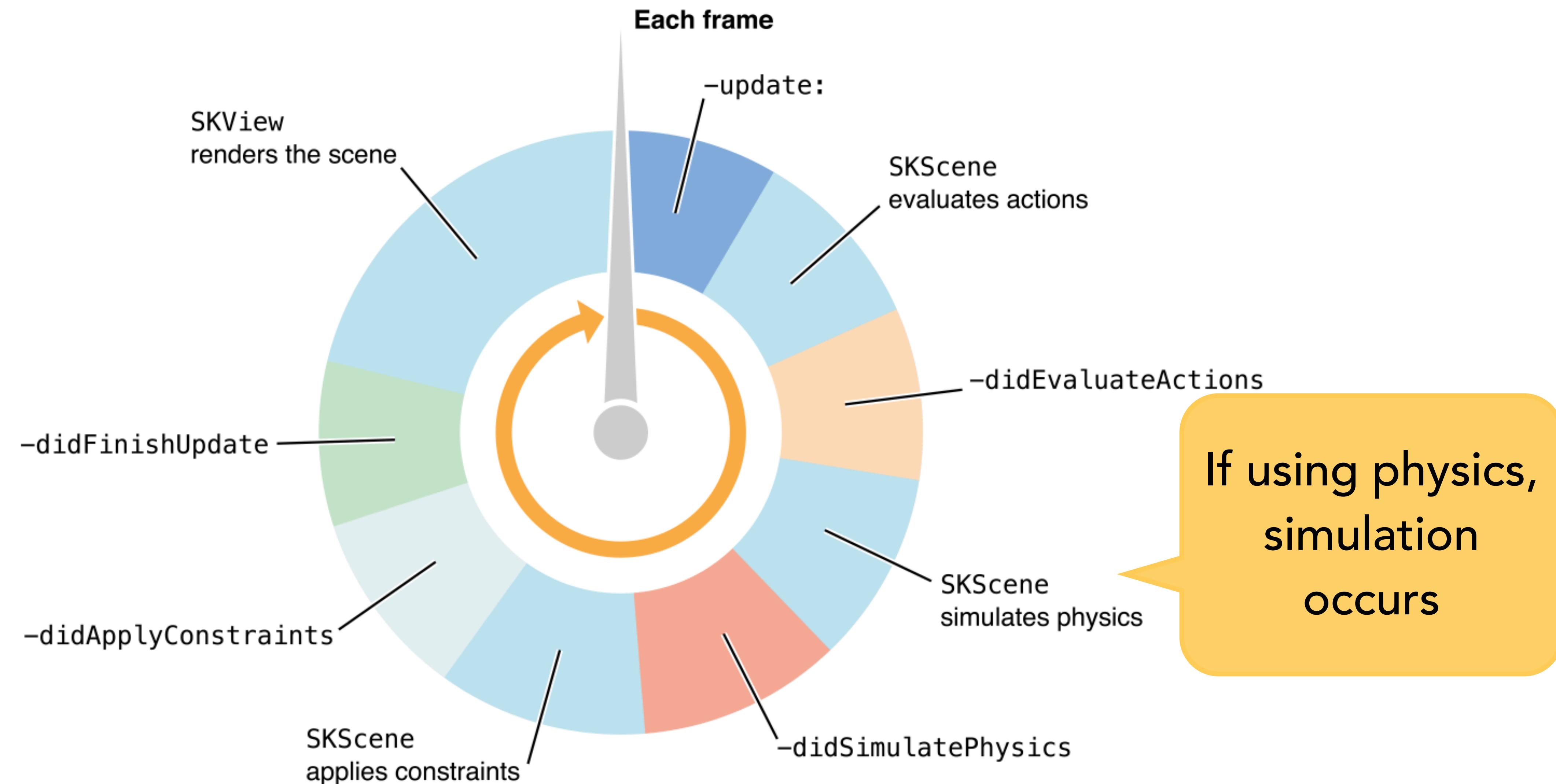
GAME RENDERING LOOP



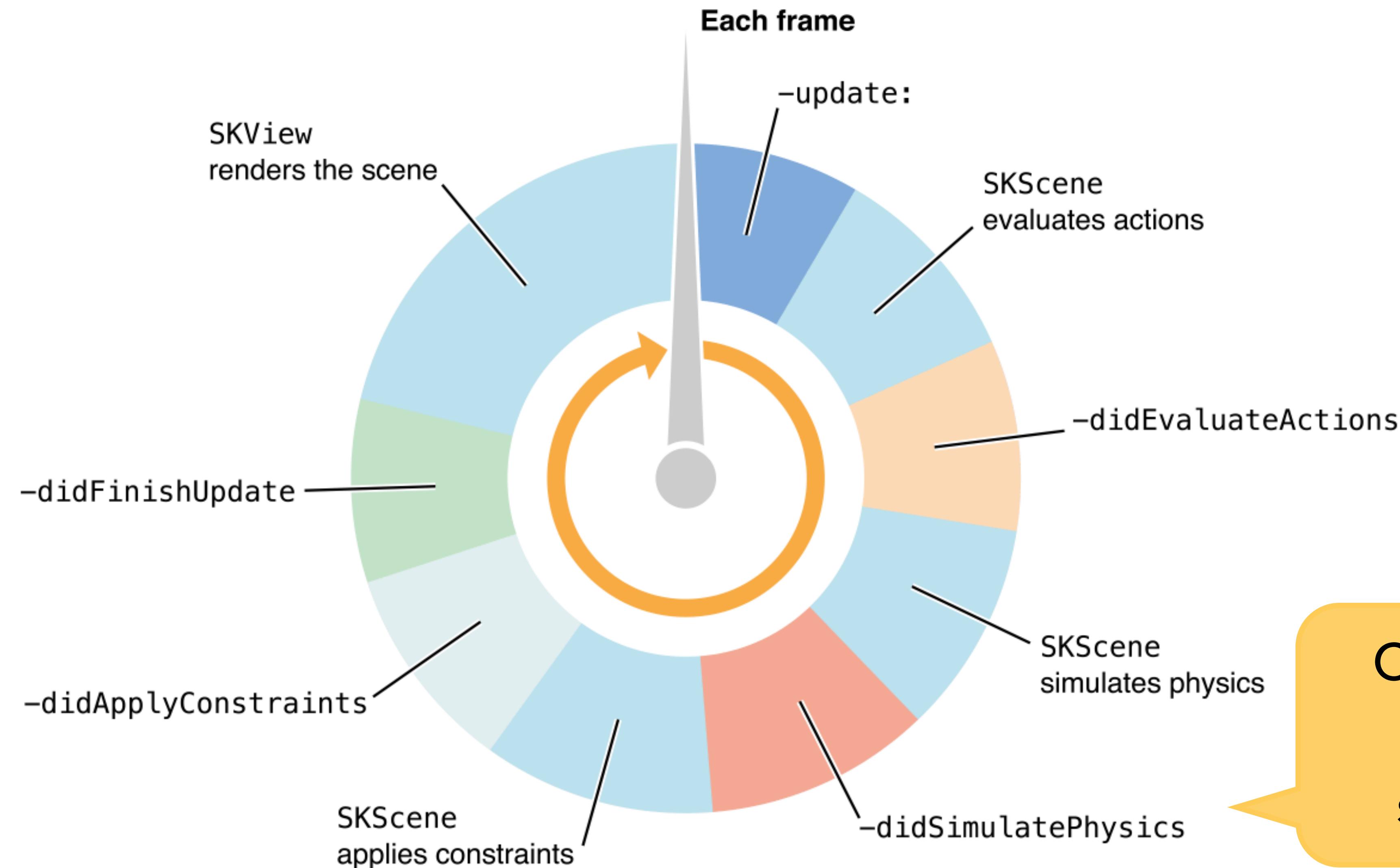
GAME RENDERING LOOP



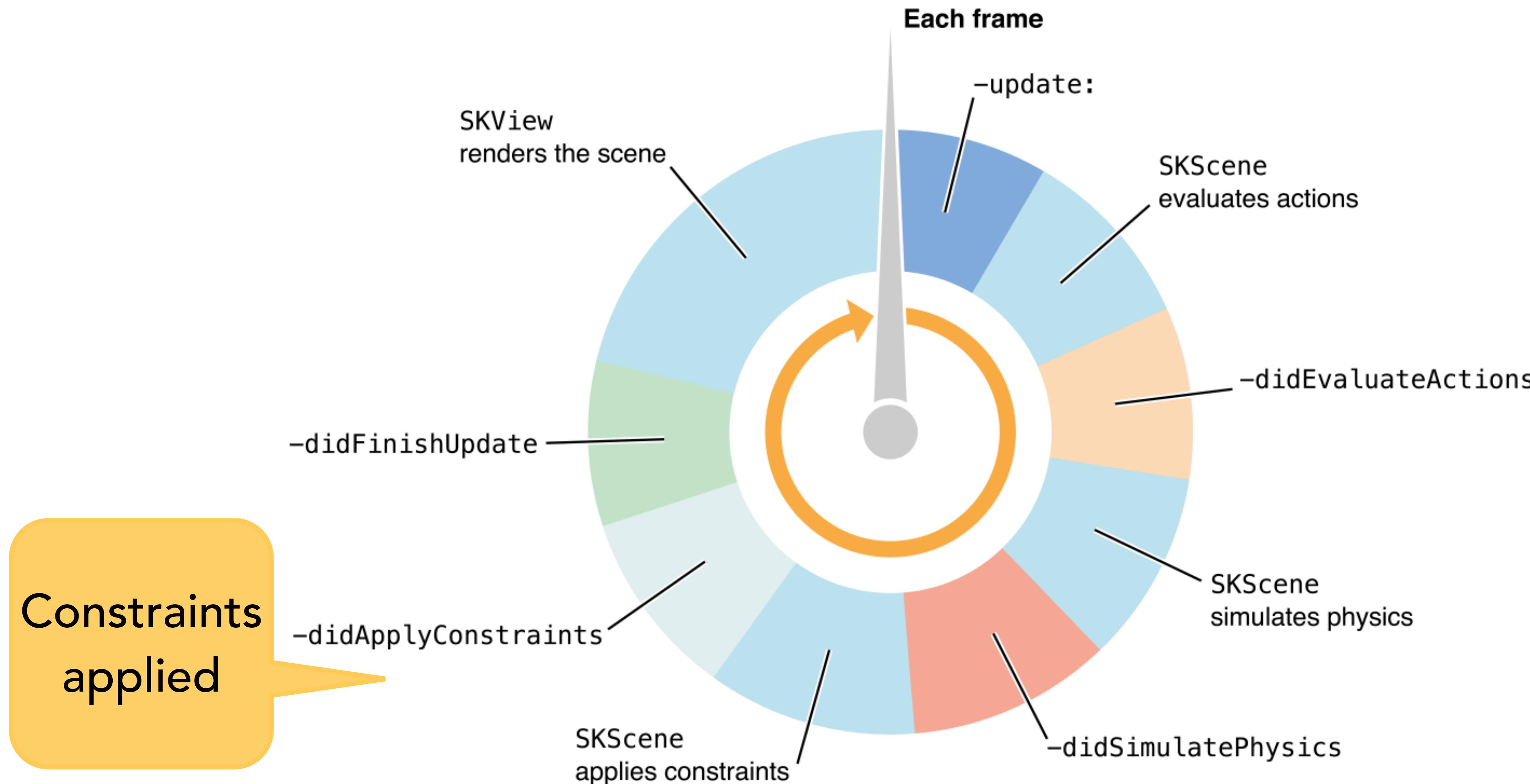
GAME RENDERING LOOP



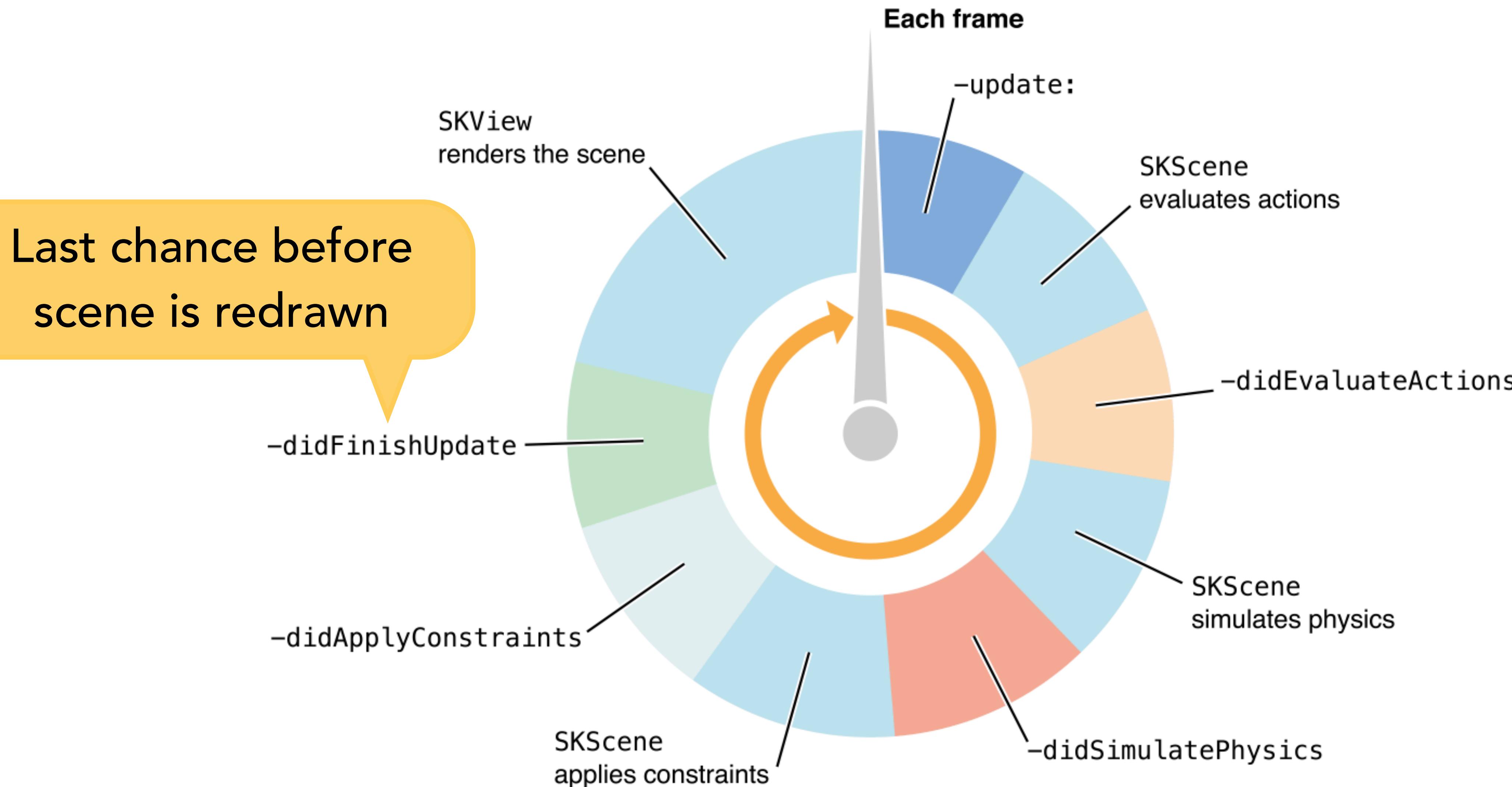
GAME RENDERING LOOP



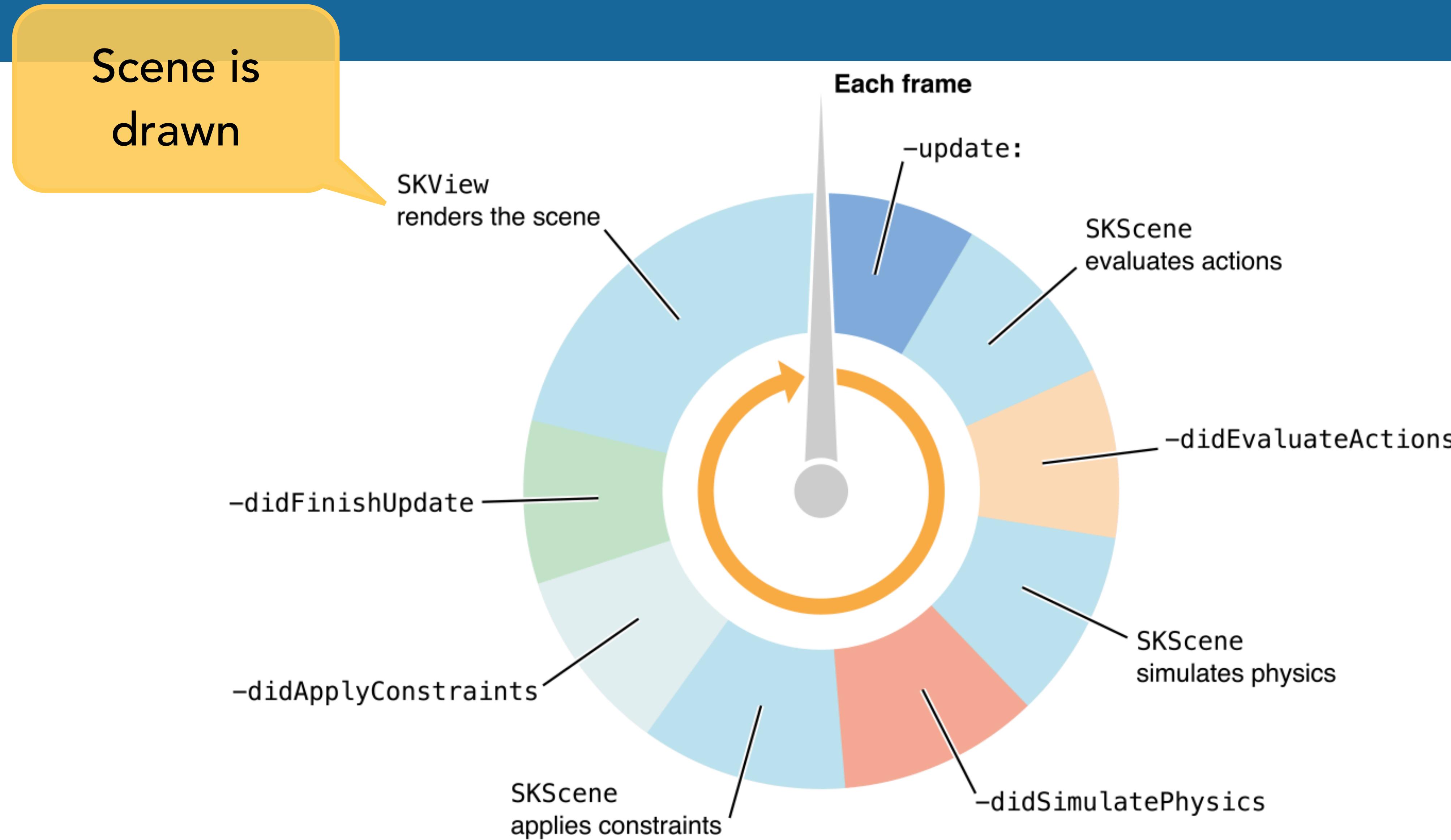
GAME RENDERING LOOP



GAME RENDERING LOOP

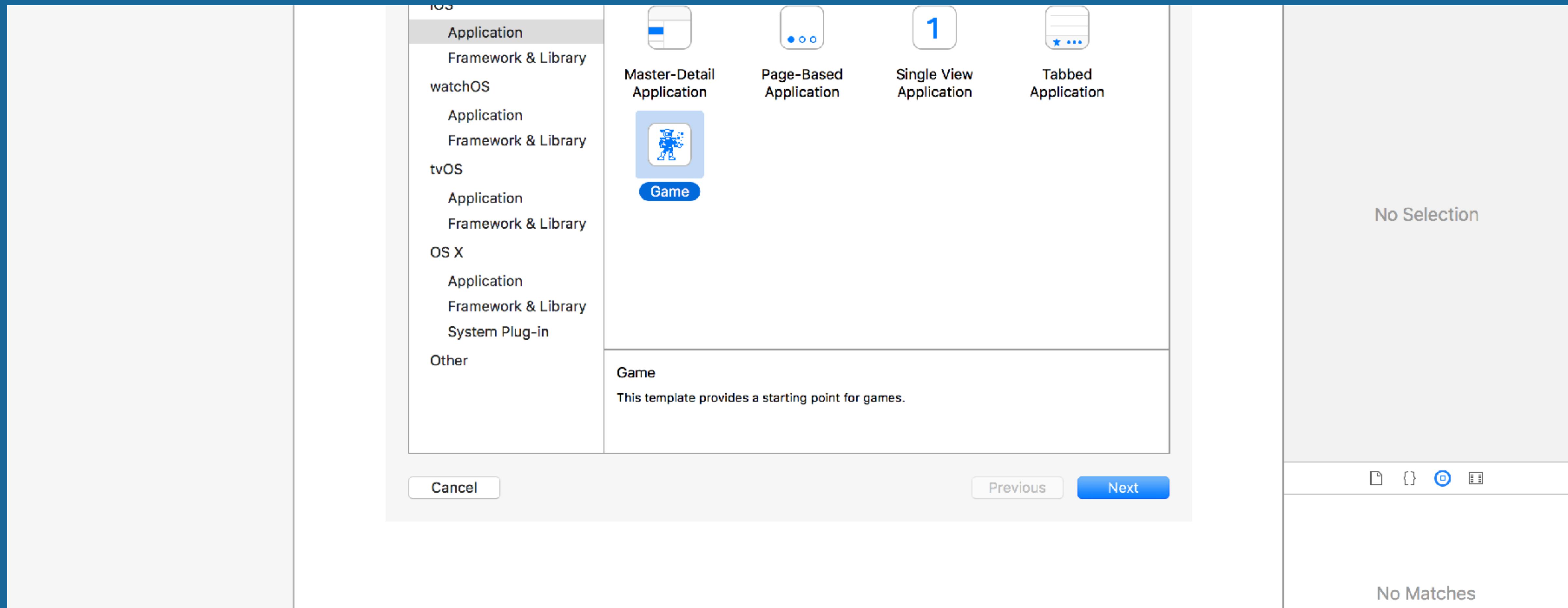


GAME RENDERING LOOP



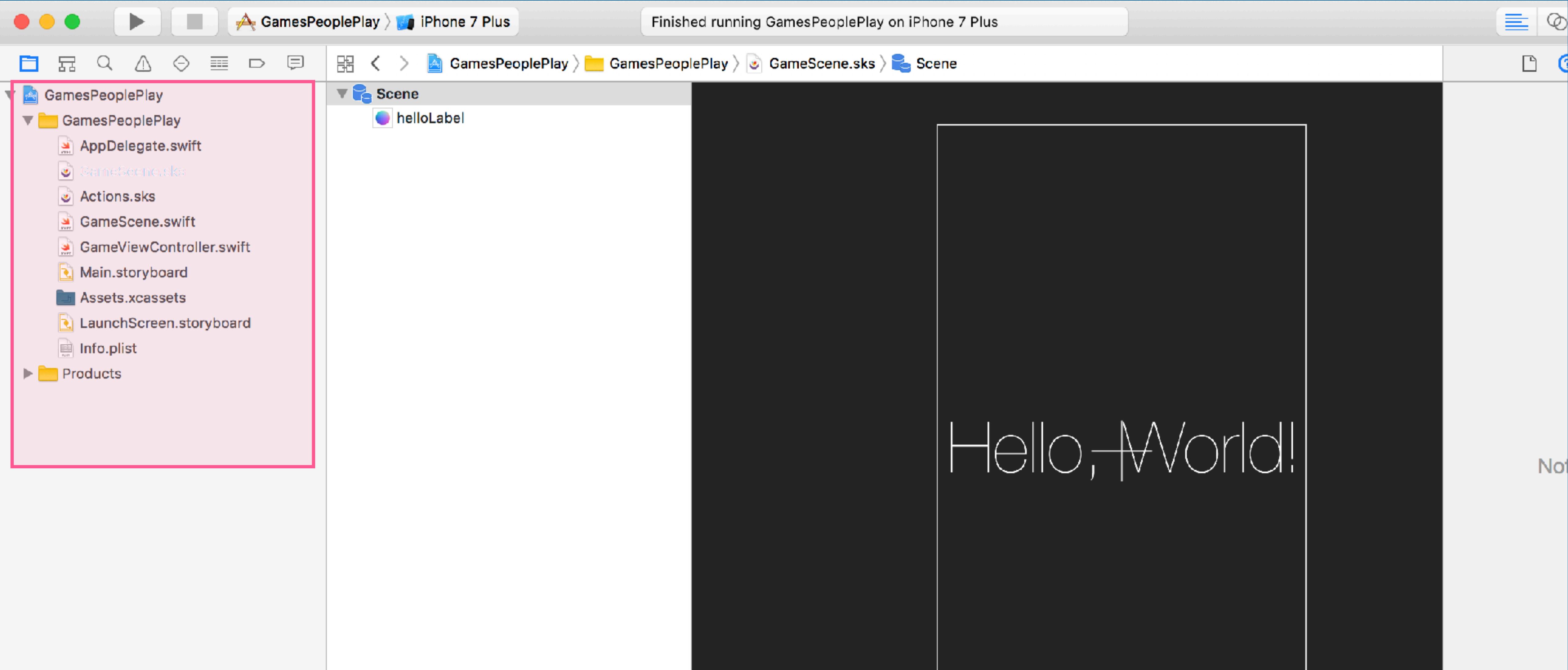
CREATING A SPRITE KIT PROJECT

CREATING A SPRITE KIT PROJECT

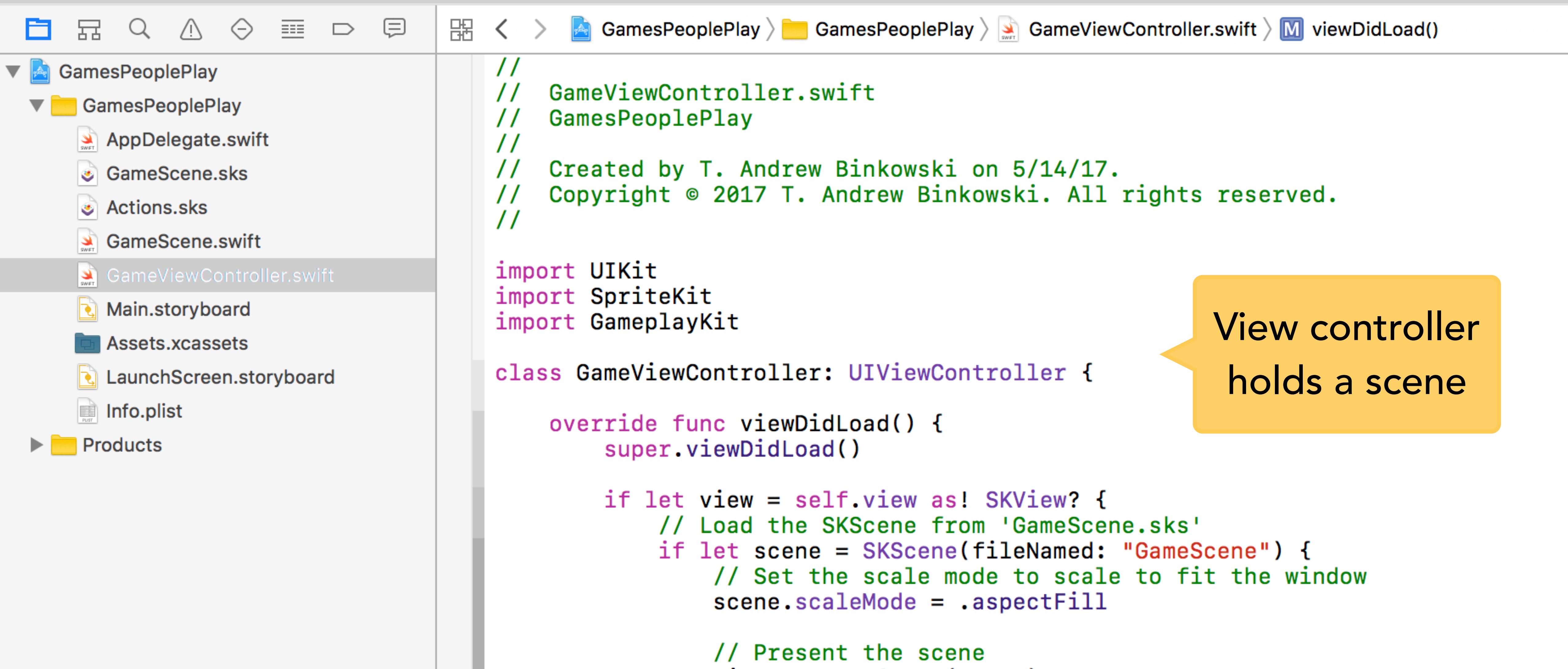


- Game template in Xcode

CREATING A SPRITE KIT PROJECT



CREATING A SPRITE KIT PROJECT



The screenshot shows the Xcode interface with the project navigation bar at the top. Below it is the file browser on the left, showing the project structure:

- GamesPeoplePlay (project folder)
- GamesPeoplePlay (group folder)
- AppDelegate.swift
- GameScene.sks
- Actions.sks
- GameScene.swift
- GameViewController.swift (selected)
- Main.storyboard
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist
- Products

The GameViewController.swift file contains the following Swift code:

```
// GameViewController.swift
// GamesPeoplePlay
//
// Created by T. Andrew Binkowski on 5/14/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import UIKit
import SpriteKit
import GameplayKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let view = self.view as! SKView? {
            // Load the SKScene from 'GameScene.sks'
            if let scene = SKScene(fileNamed: "GameScene") {
                // Set the scale mode to scale to fit the window
                scene.scaleMode = .aspectFill

                // Present the scene
            }
        }
    }
}
```

A yellow callout bubble on the right side of the screen contains the text: "View controller holds a scene".

VIEW AND SCENES

```
import UIKit
import SpriteKit
import GameplayKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let view = self.view as! SKView? {
            // Load the SKScene from 'GameScene.sks'
            if let scene = SKScene(fileNamed: "GameScene") {
                // Set the scale mode to scale to fit the window
                scene.scaleMode = .aspectFill

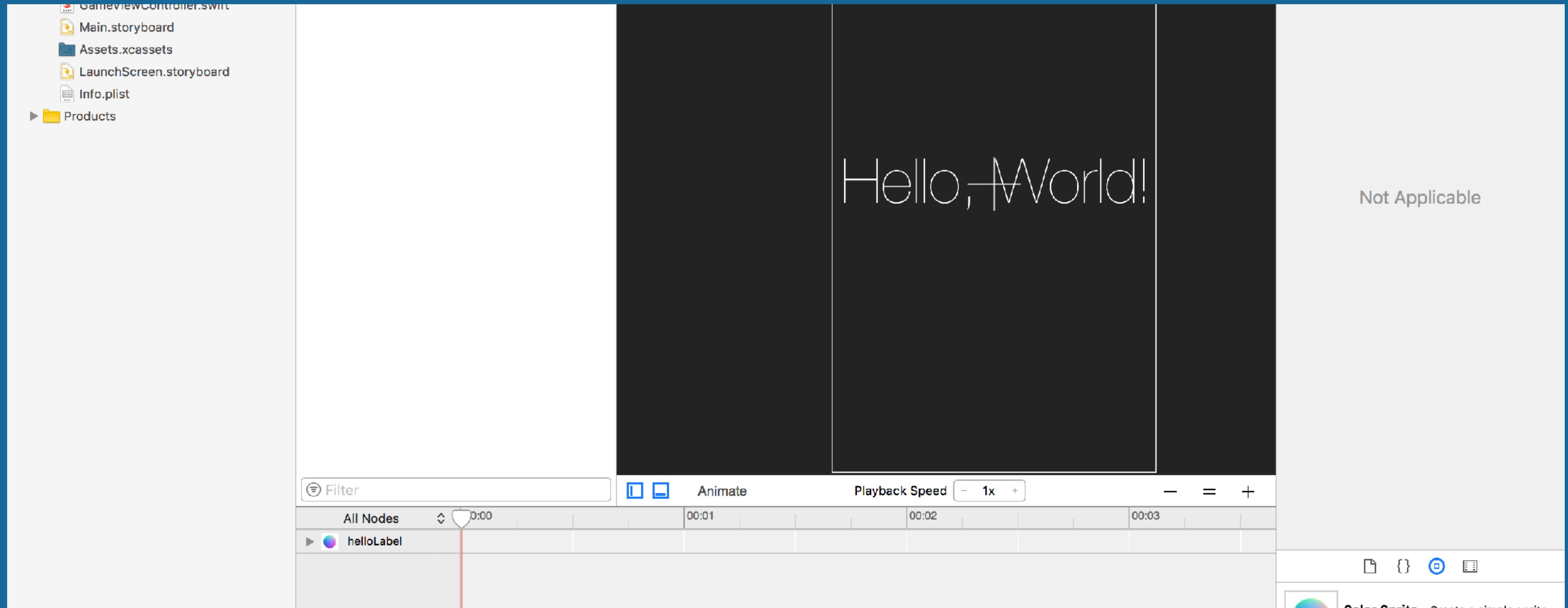
                // Present the scene
                view.presentScene(scene)
            }
            view.ignoresSiblingOrder = true

            view.showsFPS = true
            view.showsNodeCount = true
        }
    }
}
```

Cast scene
to view

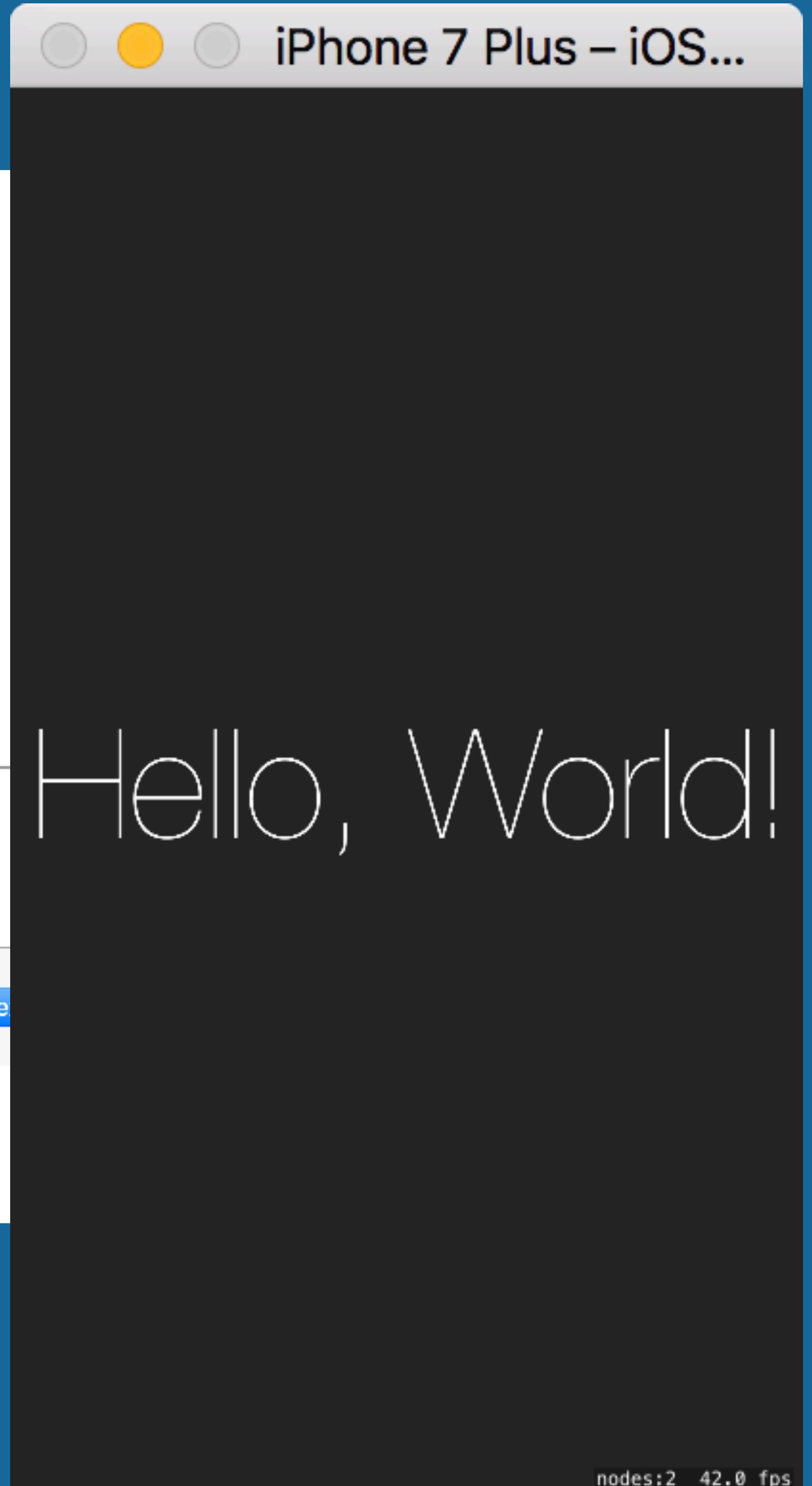
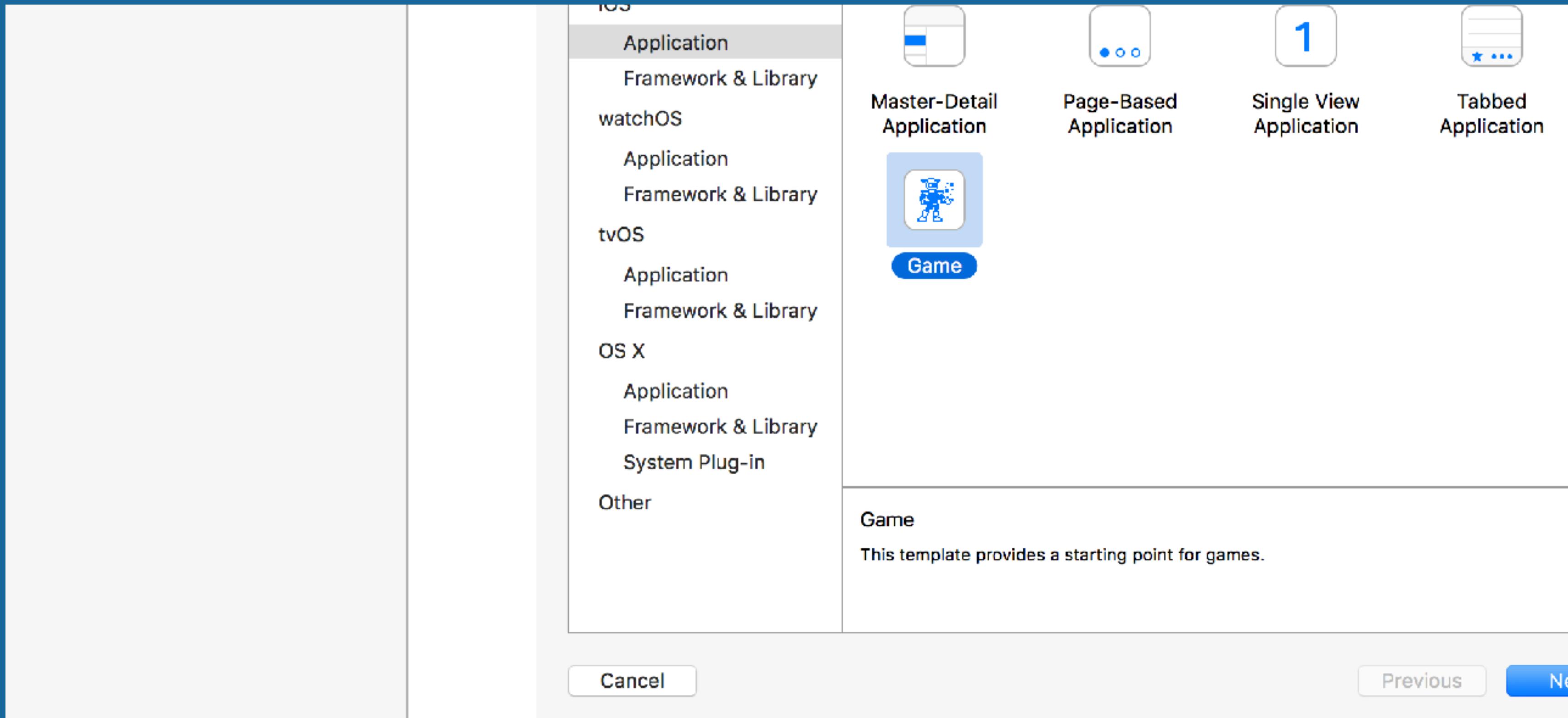
- Default template assigns SKView to view controller's main view

VIEW AND SCENES



- Scene editor

VIEW AND SCENES



- Game template in Xcode

SCENE EDITOR

SCENE EDITOR

Scene

helloLabel
SKSpriteNode

Scene

Name name
Camera
Color
Size iPhone 6s Plus
Landscape
Scale 1
Anchor Point X 0.5 Y 0.5
Gravity X 0 Y -9.8
Preview Show Camera Boundaries
 Show Physics Boundaries
 Use Camera Node

Color Sprite - Create a simple sprite with color.

Empty - Create an empty node.

Light - Create a basic light.

Customize for device

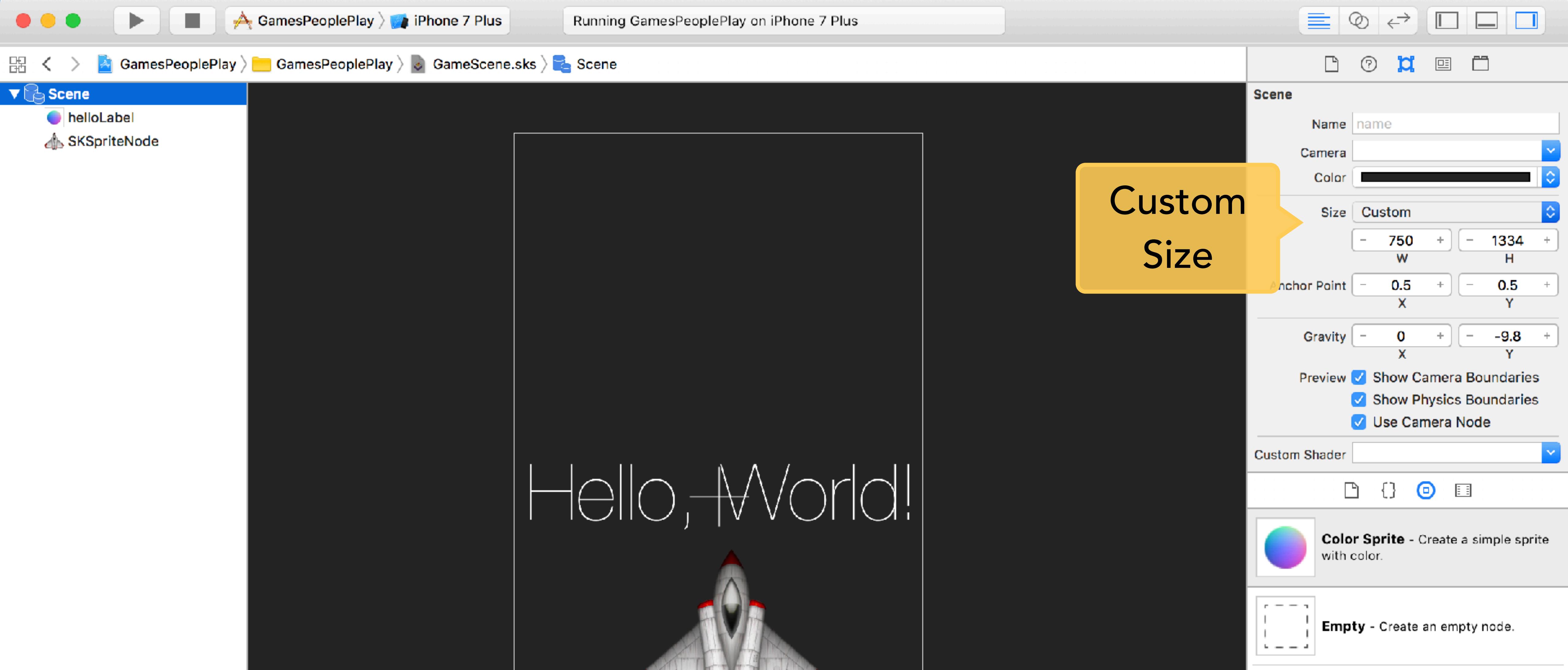
Hello, World!

Color Sprite - Create a simple sprite with color.

Empty - Create an empty node.

Light - Create a basic light.

SCENE EDITOR



SCENE EDITOR

The Scene Editor interface is shown, featuring a central canvas and various panels on the left and right.

Scene Panel: Shows the scene structure with two nodes:

- helloLabel (SKLabelNode)
- SKSpriteNode

Inspector Panel (right): Displays properties for the selected SKSpriteNode (name: name). The Scale is set to 1.0, and the Anchor Point is at (0.5, 0.5). Gravity is set to (0, -9.8). Preview options include Show Camera Boundaries, Show Physics Boundaries, and Use Camera Node, all of which are checked.

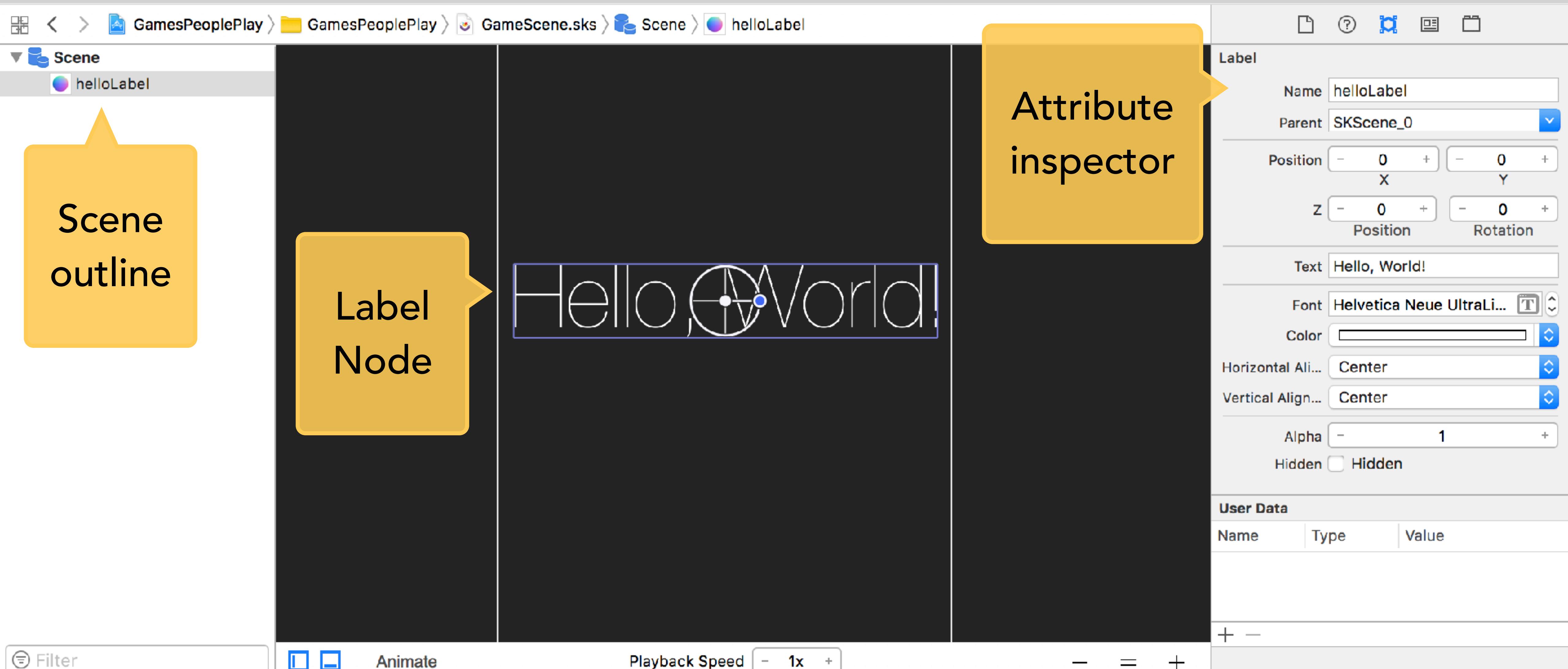
Text Labels: A yellow callout box contains the text "Scaled coordinate space".

Bottom Panel: Shows node creation options: Color Sprite (Create a simple sprite with color) and Empty (Create an empty node).

SCENE EDITOR



SCENE EDITOR



SCENE EDITOR

Scene outline

Hello, World!

SpriteKit
object
library

Animate Panel

- 750 + - 1334 +
W H

Anchor Point - 0.5 + - 0.5 +
X Y



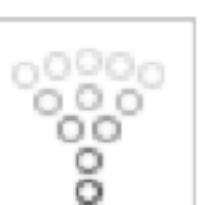
Color Sprite - Create a simple sprite with color.



Empty - Create an empty node.



Light - Create a basic light.



Emitter - Create a basic emitter.



Label - Create a basic label.



Shape Node (Square) - Create a square shape node.

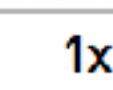
Filter



Animate



Playback Speed

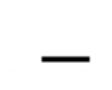


-

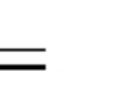
1x



+



-



=



+

All Nodes

0:00

00:01

00:02

00:03

helloLabel

ADDING NODES

SCENE EDITOR

GamesPeoplePlay

- GamesPeoplePlay
- AppDelegate.swift
- GameScene.sks
- Actions.sks
- GameScene.swift
- GameViewController.swift
- Main.storyboard
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist

Products

AppIcon

Spaceship

Spaceship

Image

1x 2x 3x

Universal

No Quick Help

Search Documentation

Spaceship asset

Color Sprite - Create a simple sprite with color.

Empty - Create an empty node

SCENE EDITOR

The screenshot shows the Xcode Scene Editor interface for a game project named "GamesPeoplePlay". The project structure is visible in the top navigation bar: GamesPeoplePlay > GamesPeoplePlay > GameScene.sks > Scene > SKSpriteNode.

The main canvas displays the game scene with the following elements:

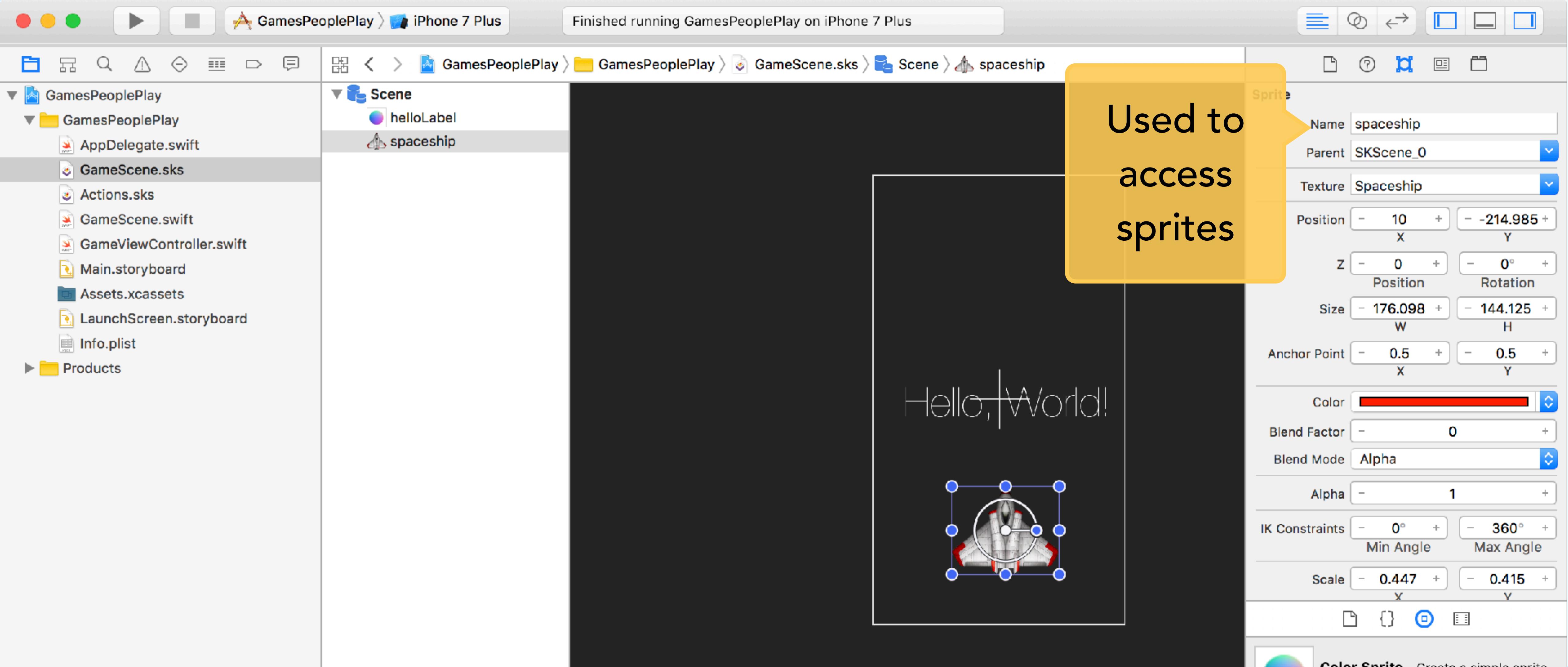
- A large white "Hello, World!" label centered at the top.
- An SKSpriteNode representing a spaceship, which is currently selected. It has a white and red triangular shape with a circular cockpit area. A blue selection outline with handles is around it.

The right-hand panel contains the Inspector for the selected SKSpriteNode, with the following settings:

- Sprite** section:
 - Name: name
 - Parent: SKScene_0
 - Texture: Spaceship (highlighted with a yellow callout)
 - Position: -22.662, -269.341
 - Size: 394, 347
 - Anchor Point: 0.5, 0.5
 - Color: Red
 - Blend Factor: 0
 - Blend Mode: Alpha
- Buttons: Create Color Sprite, Create Empty, Create Light
- Color Sprite: Description: Create a simple sprite with color.
- Empty: Description: Create an empty node.
- Light: Description: Create a basic light.

A yellow callout box points from the text "Texture" to the "Spaceship" dropdown in the Inspector.

SCENE EDITOR



SCENE EDITOR

GamesPeoplePlay > iPhone 7 Plus Finished running GamesPeoplePlay on iPhone 7 Plus

GamesPeoplePlay > GamesPeoplePlay > GameScene.sks > Scene > spaceship

Sprite

Name: spaceship
Parent: SKScene_0
Texture: Spaceship
Position: X: -10, Y: -214.985
Z: 0, Rotation: 0°

Optional
SKSpriteNode

```
// Create a node from asset
let spaceship: SKSpriteNode = self.childNode(withName: "spaceship") as! SKSpriteNode
spaceship.xScale = 0.2
spaceship.yScale = 0.2
```

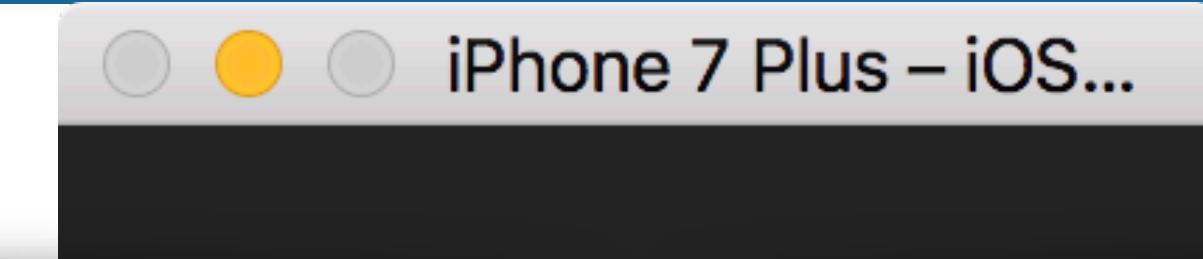
The screenshot shows the Xcode interface with the Scene Editor open. In the top bar, it says "GamesPeoplePlay > iPhone 7 Plus" and "Finished running GamesPeoplePlay on iPhone 7 Plus". Below that, the file path is "GamesPeoplePlay > GamesPeoplePlay > GameScene.sks > Scene > spaceship". On the left, the Project Navigator shows files like AppDelegate.swift, GameScene.sks, Actions.sks, GameScene.swift, GameViewController.swift, Main.storyboard, and Assets.xcassets. The main canvas shows a gray scene with a white rectangle and a small red and white spaceship sprite node. The right side has a Inspector pane with tabs for Sprite, Color, and Physics. The Sprite tab shows settings for the "spaceship" node: Name is "spaceship", Parent is "SKScene_0", Texture is "Spaceship", Position is X: -10, Y: -214.985, Z: 0, Rotation: 0°. A yellow callout bubble points from the text "Optional SKSpriteNode" in the code editor below to the "spaceship" node in the scene. The code editor contains the following Swift code:

```
// Create a node from asset
let spaceship: SKSpriteNode = self.childNode(withName: "spaceship") as! SKSpriteNode
spaceship.xScale = 0.2
spaceship.yScale = 0.2
```

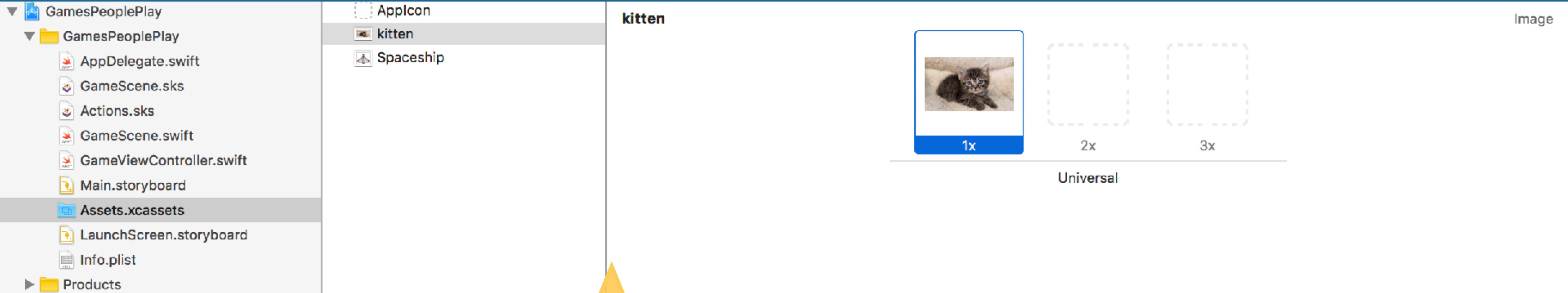
The code creates a node from an asset named "spaceship" and scales it down by 0.2 on both axes.

SCENE EDITOR

```
// Create a node from asset
let spaceship: SKSpriteNode = self.childNode(withName: "spaceship") as! SKSpriteNode
spaceship.xScale = 0.2
spaceship.yScale = 0.2
```



SCENE EDITOR



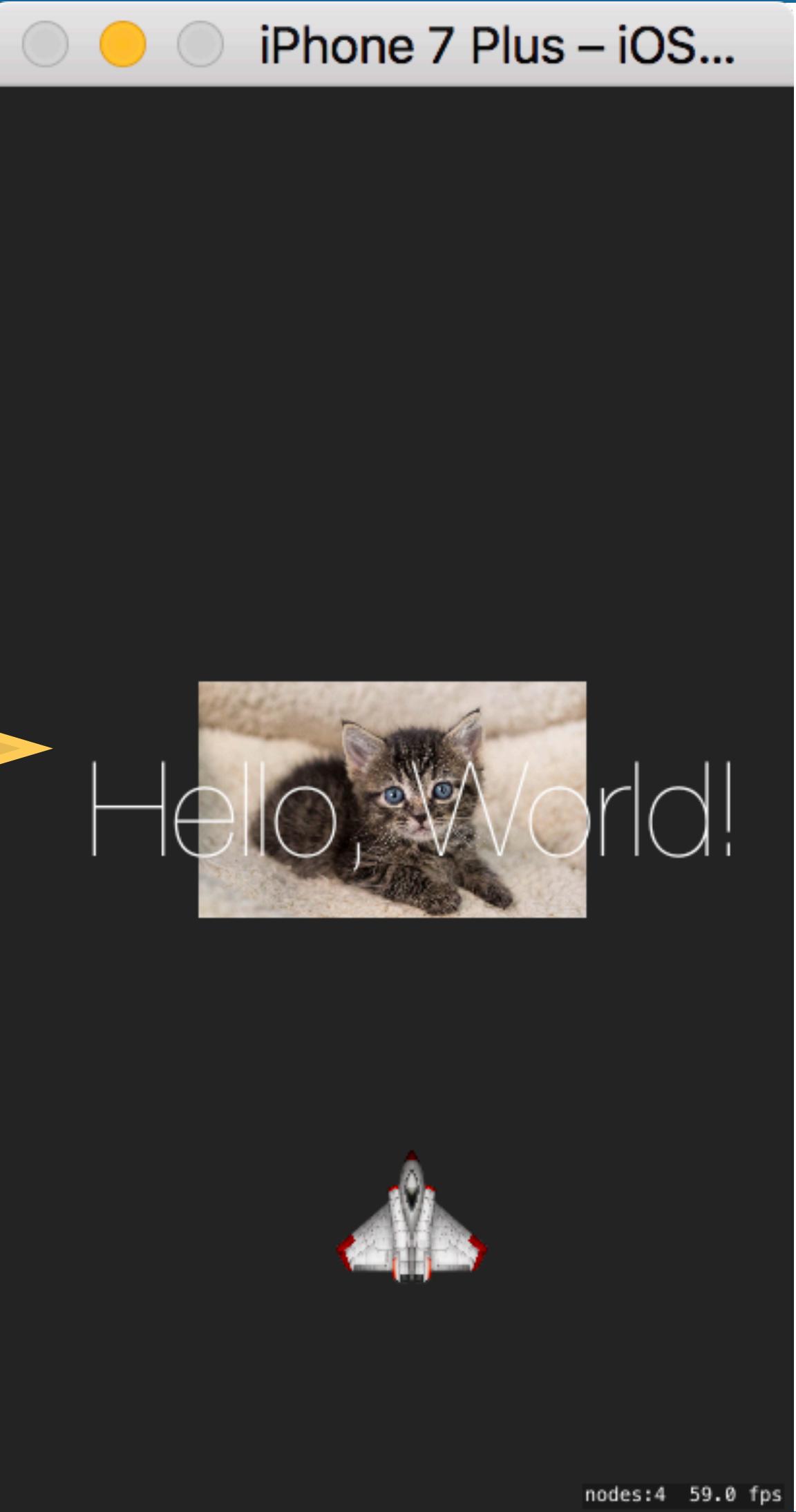
Create a kitten node
at runtime

SCENE EDITOR

```
override func didMove(to view: SKView) {  
  
    // Create a node from asset  
    let spaceship: SKSpriteNode = self.childNode(withName: "spaceship") as! SKSpriteNode  
    spaceship.xScale = 0.2  
    spaceship.yScale = 0.2  
  
    // Add a kitten node from custom asset  
    let kitten: SKSpriteNode = SKSpriteNode(imageNamed: "kitten")  
    self.addChild(kitten)  
    kitten.xScale = 0.2  
    kitten.yScale = 0.2  
  
    // Get label node from scene and store it for use later  
    self.label = self.childNode(withName: "//helloLabel") as? SKLabelNode  
    if let label = self.label {  
        label.alpha = 0.0
```

SCENE EDITOR

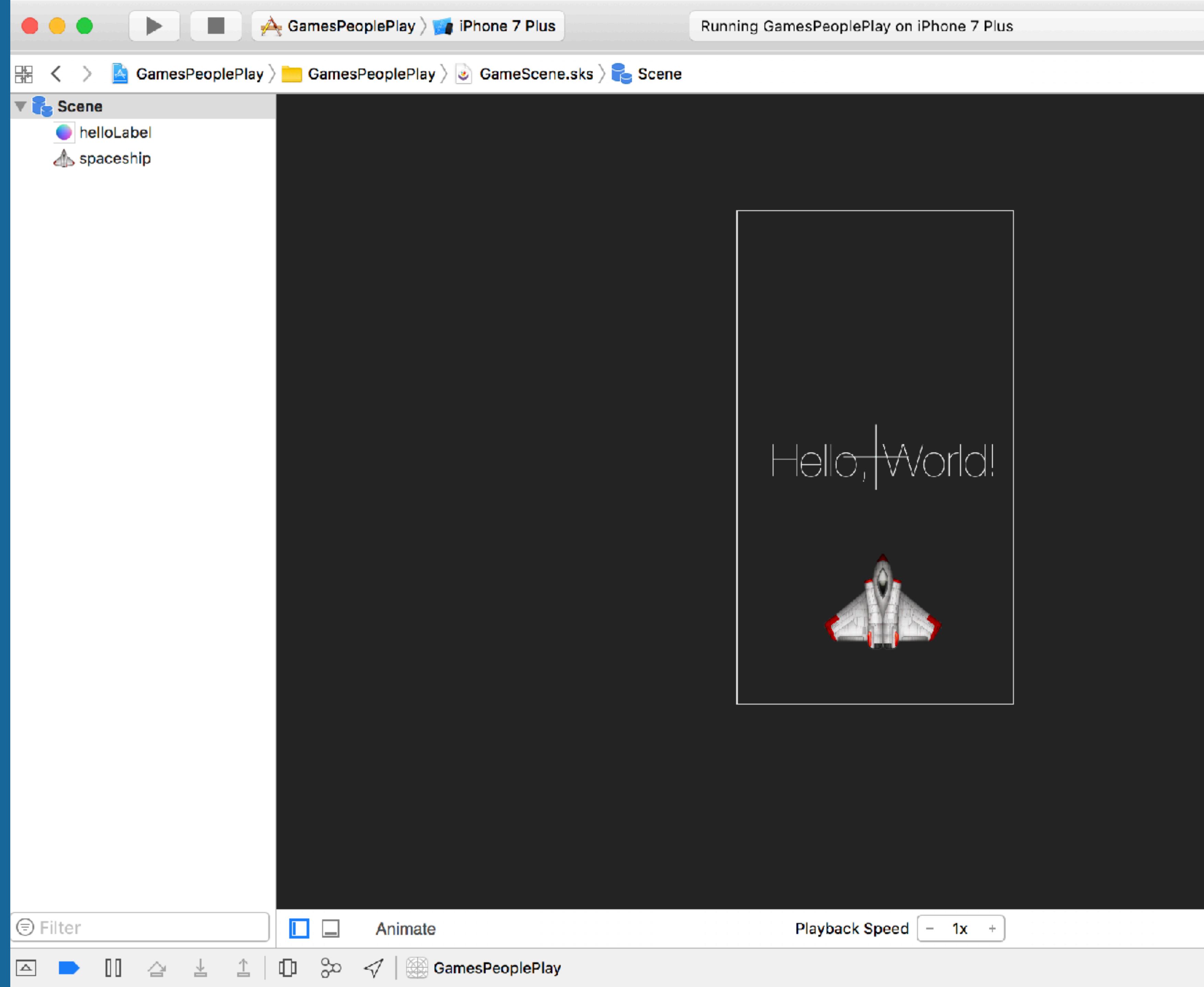
Add at origin



nodes:4 59.0 fps

SCENE EDITOR

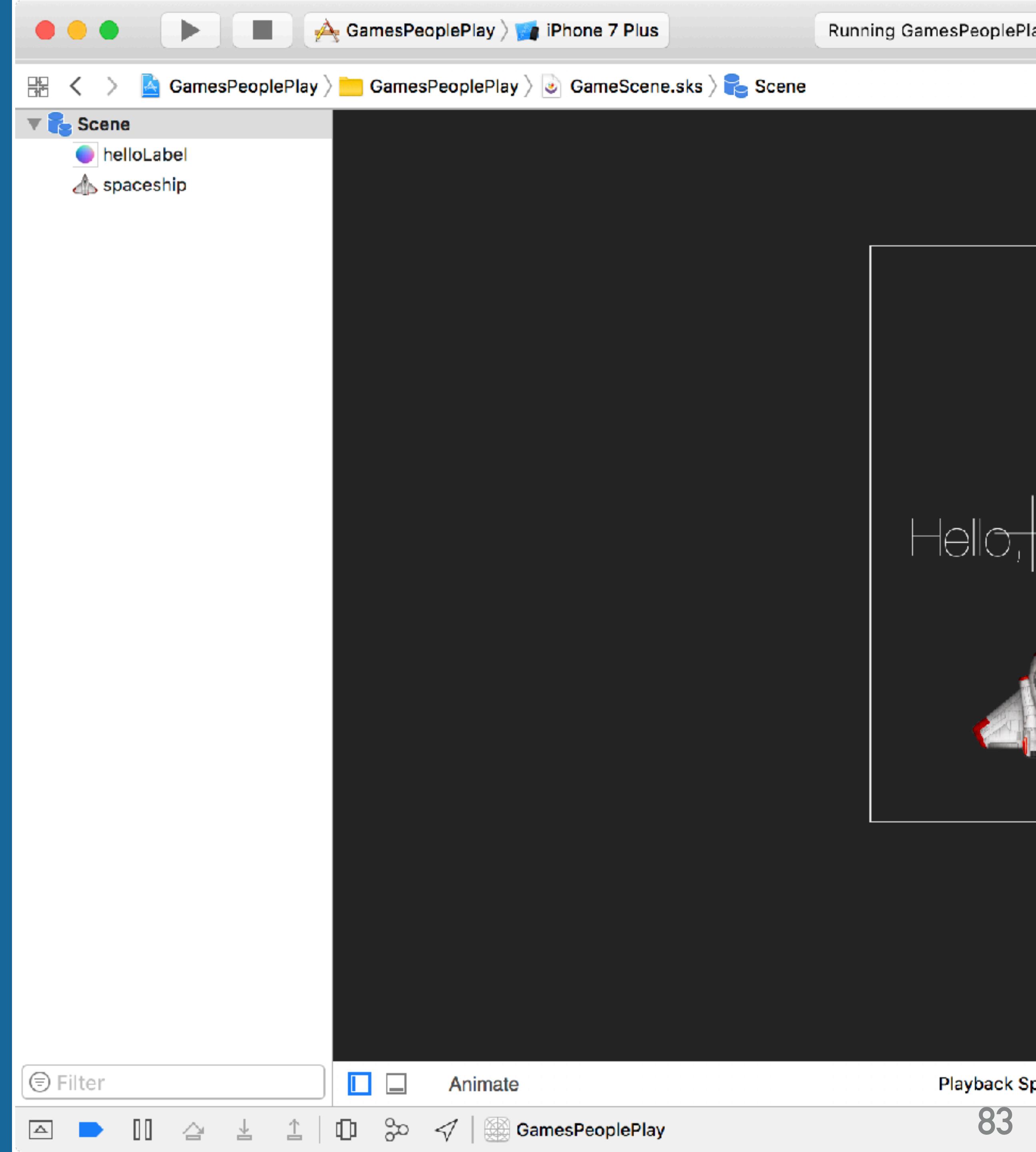
- Parents are nodes that contain other nodes
 - Children are those other nodes
- A child must have a parent object to be drawn to the screen



SCENE EDITOR

SUBTITLE

- Parents and children can be managed
 - Programmatically
 - Document outline



SCENE EDITOR

- Special syntax for searching the node tree to find a node

Syntax	Description
/	When placed at the start of the search string, this indicates that the search should be performed on the tree's root node. When placed anywhere but the start of the search string, this indicates that the search should move to the node's children.
//	When placed at the start of the search string, this specifies that the search should begin at the root node and be performed recursively across the entire node tree. Otherwise, it performs a recursive search from its current position.
.	Refers to the current node.
..	The search should move up to the node's parent.
*	The search matches zero or more characters.

```
// Get label node from scene and store it for use later
self.label = self.childNode(withName: "//helloLabel") as? SKLabelNode
if let label = self.label {
    label.alpha = 0.0
    label.run(SKAction.fadeIn(withDuration: 2.0))
}
```

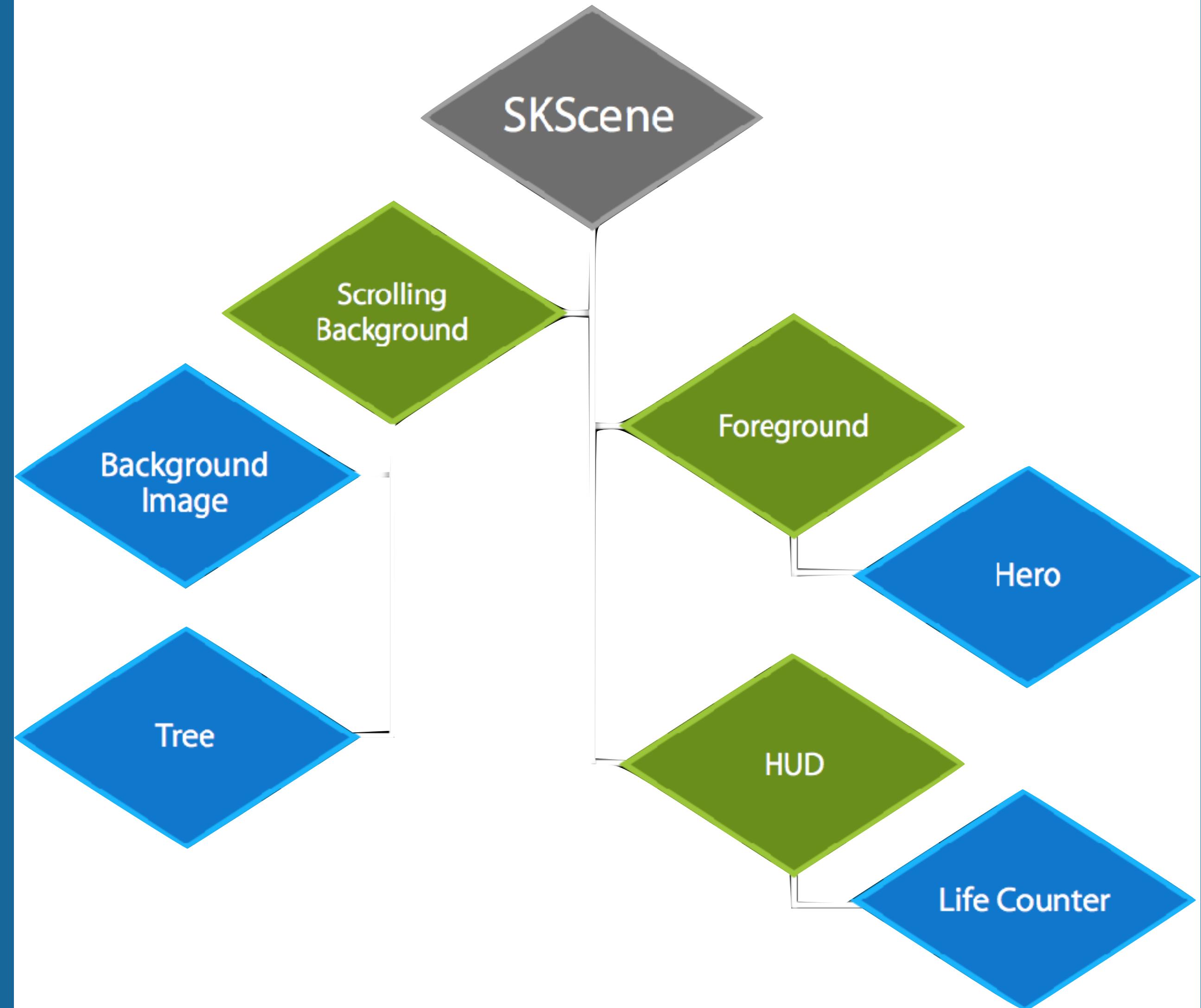
ined inside the brackets.

characters

ADDING A SCENE

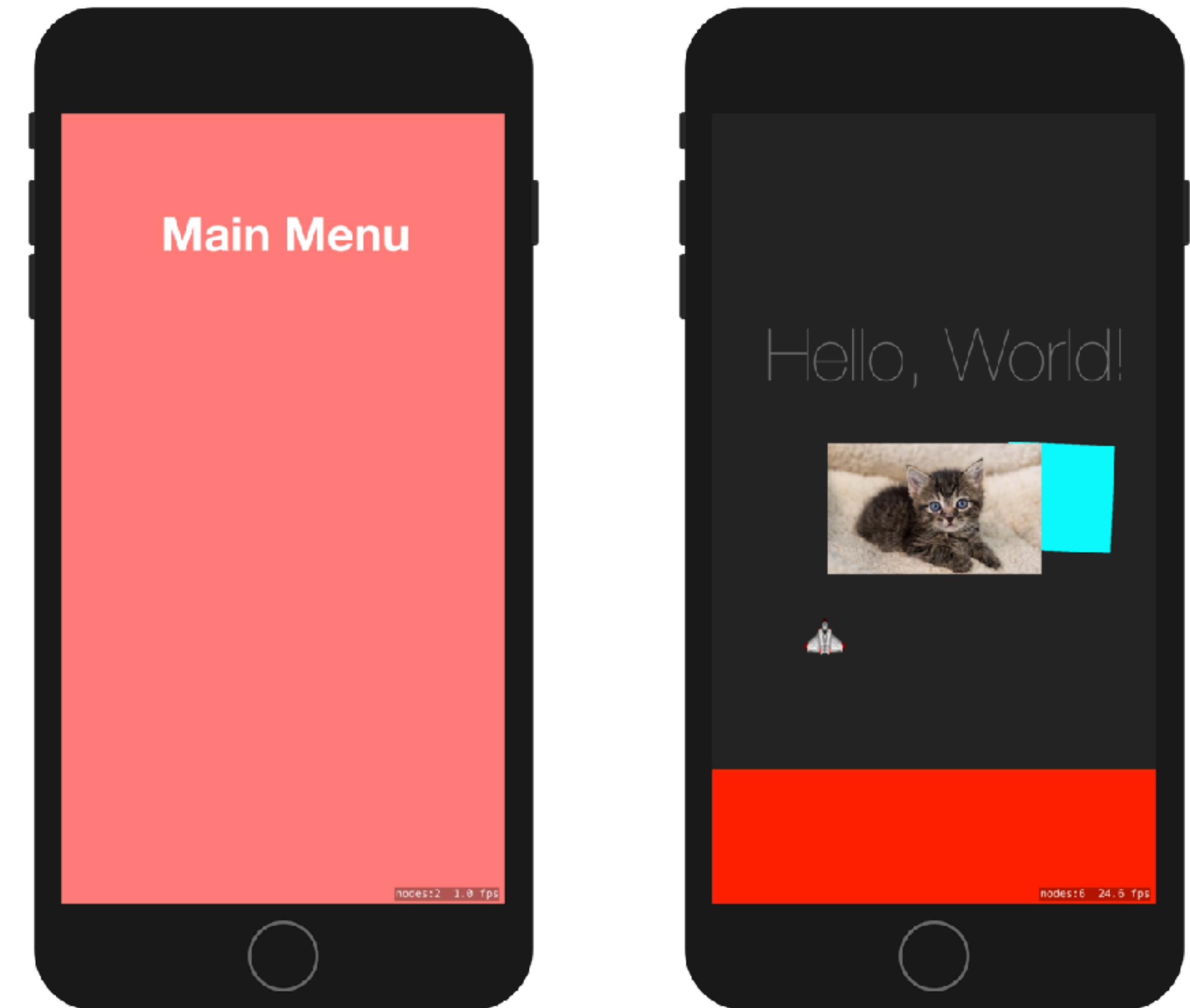
SCENE TRANSITIONS

- Root node of the scene graph
- Displayed by a SKView
- Inherits from SKEffectNode
Runs per-frame loop
 - -update:
 - -didEvaluateActions
 - -didSimulatePhysics



SCENE TRANSITIONS

- Typical to create different scenes for
 - Main menu
 - Game options
 - Gameplay
 - Game Level
 - Achievement Alert
 - Game over



SCENE TRANSITIONS

SUBTITLE

- Transition between scenes using `SKTransition`
 - Performs a transition between current and new SKScene
 - Standard transitions
 - Cross fade, fade thru color
 - Doors closing, doors opening
 - Flip vertical, flip horizontal Move in, push in, reveal
 - CIFilter transitions



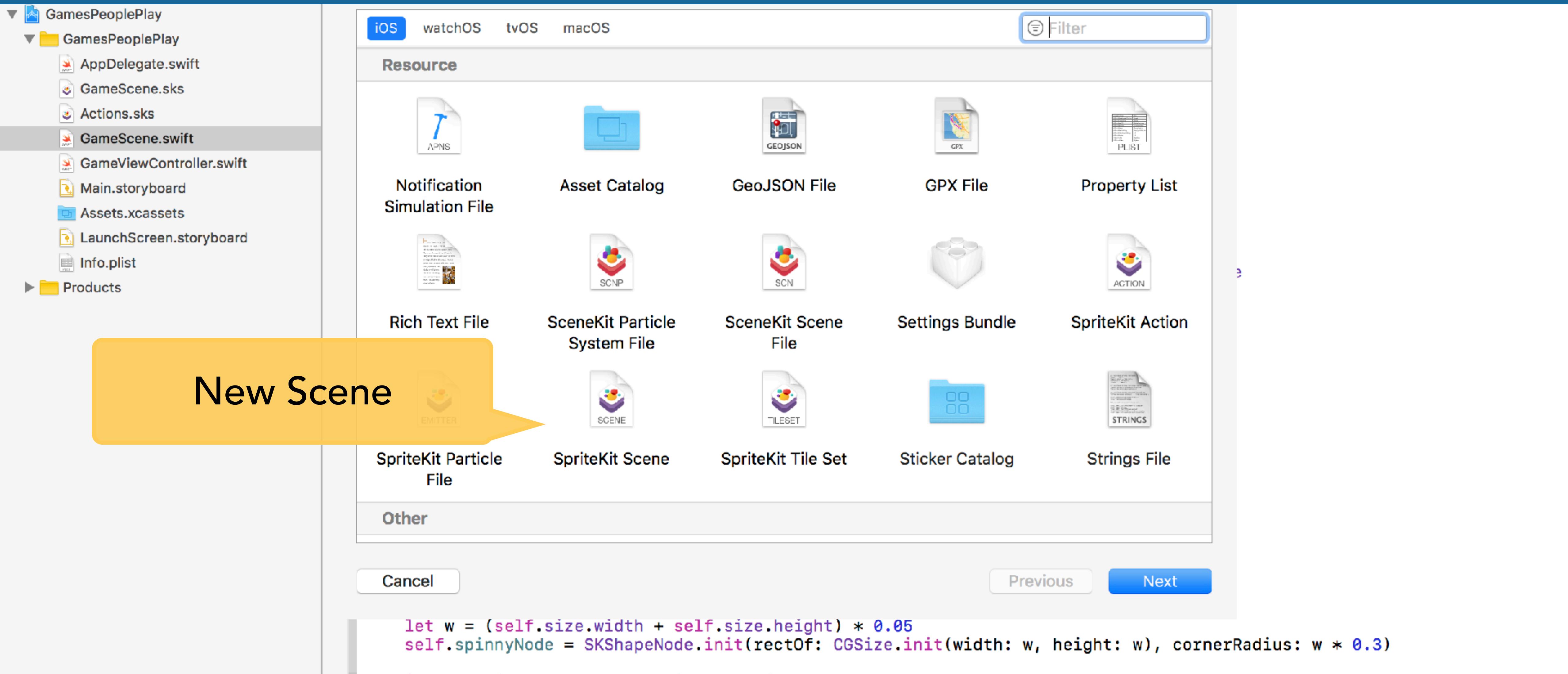
Hello, World!



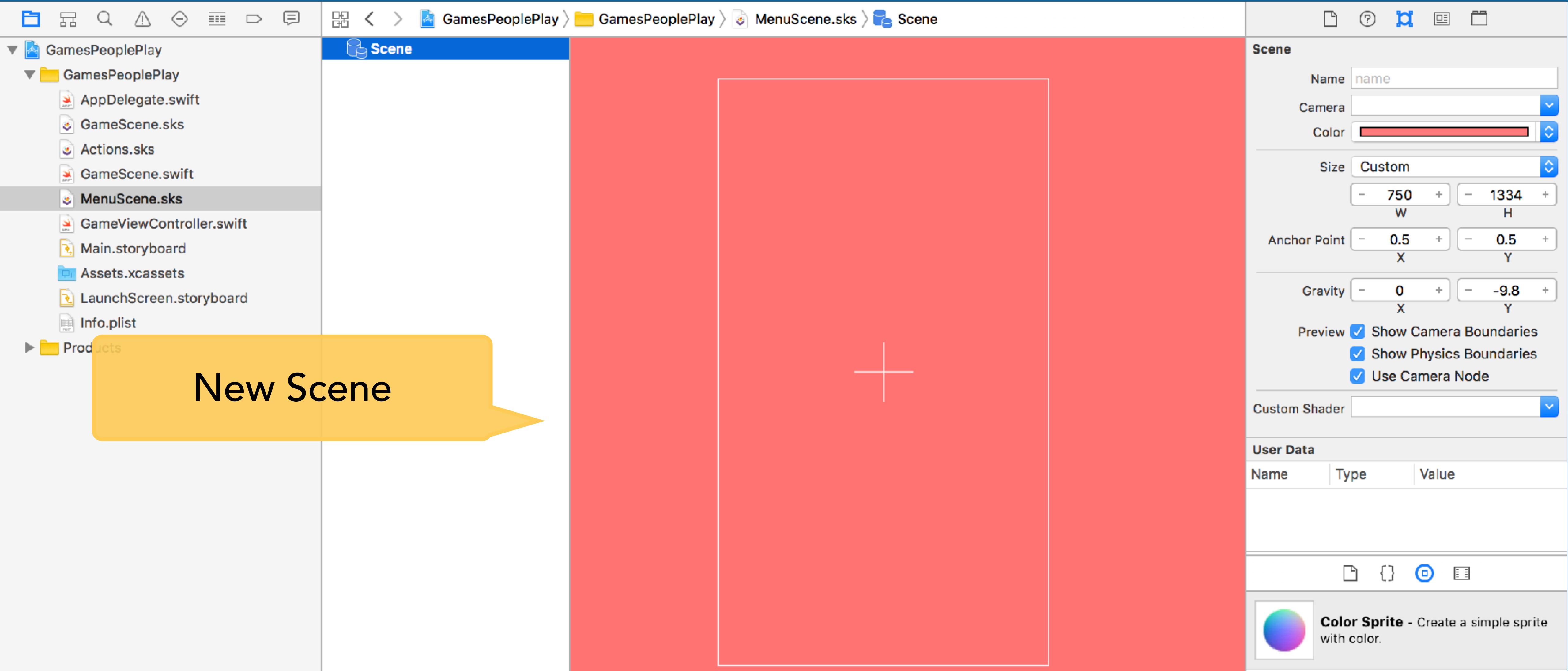
VIEW AND SCENES

```
override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {  
  
    // Create a new ball scene from the scene file  
    guard let ballScene = BallScene(fileNamed: "BallScene") else {  
        print("Trouble creating ball scene")  
        return  
    }  
  
    // Set the transition  
    let transition = SKTransition.doorOpenVerticalWithDuration(1.0)  
  
    // Present it  
    view?.presentScene(ballScene, transition: transition)  
}
```

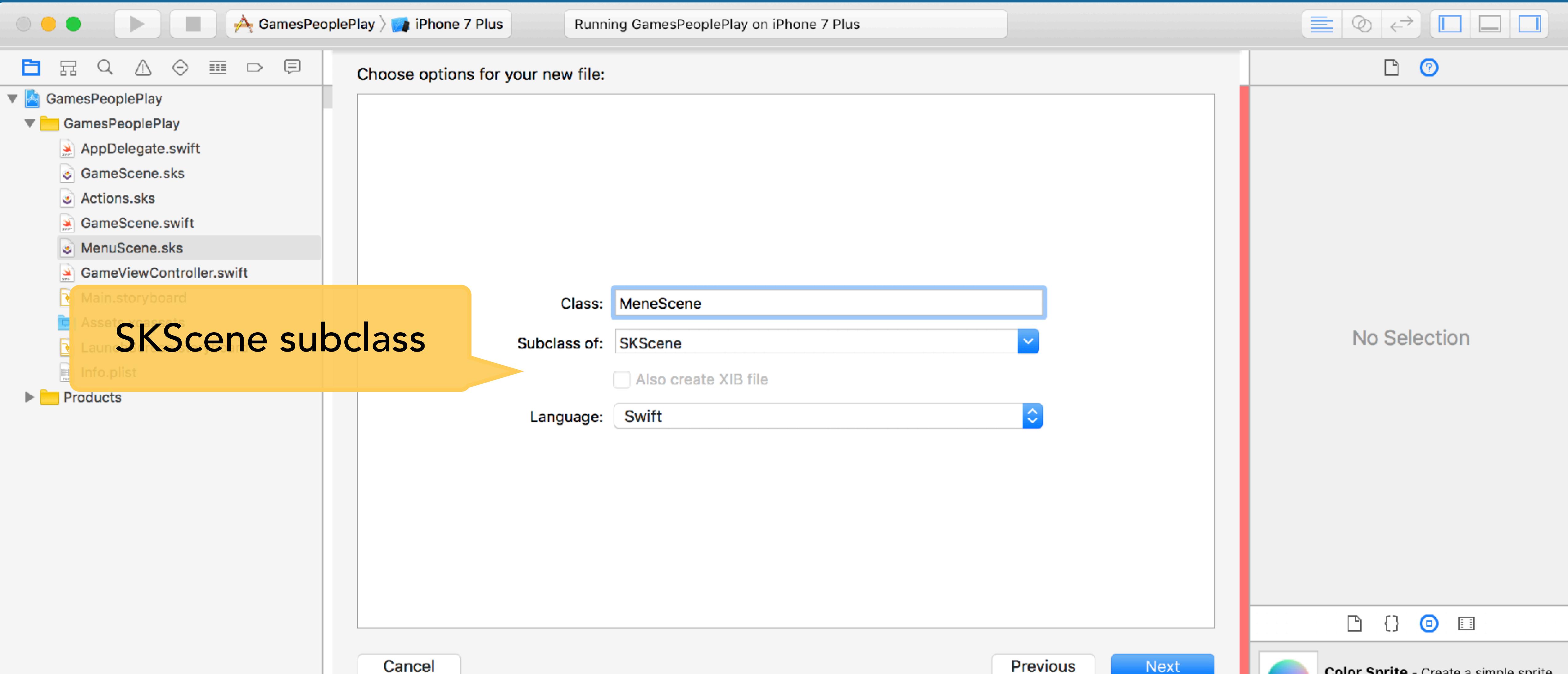
ADDING A SCENE



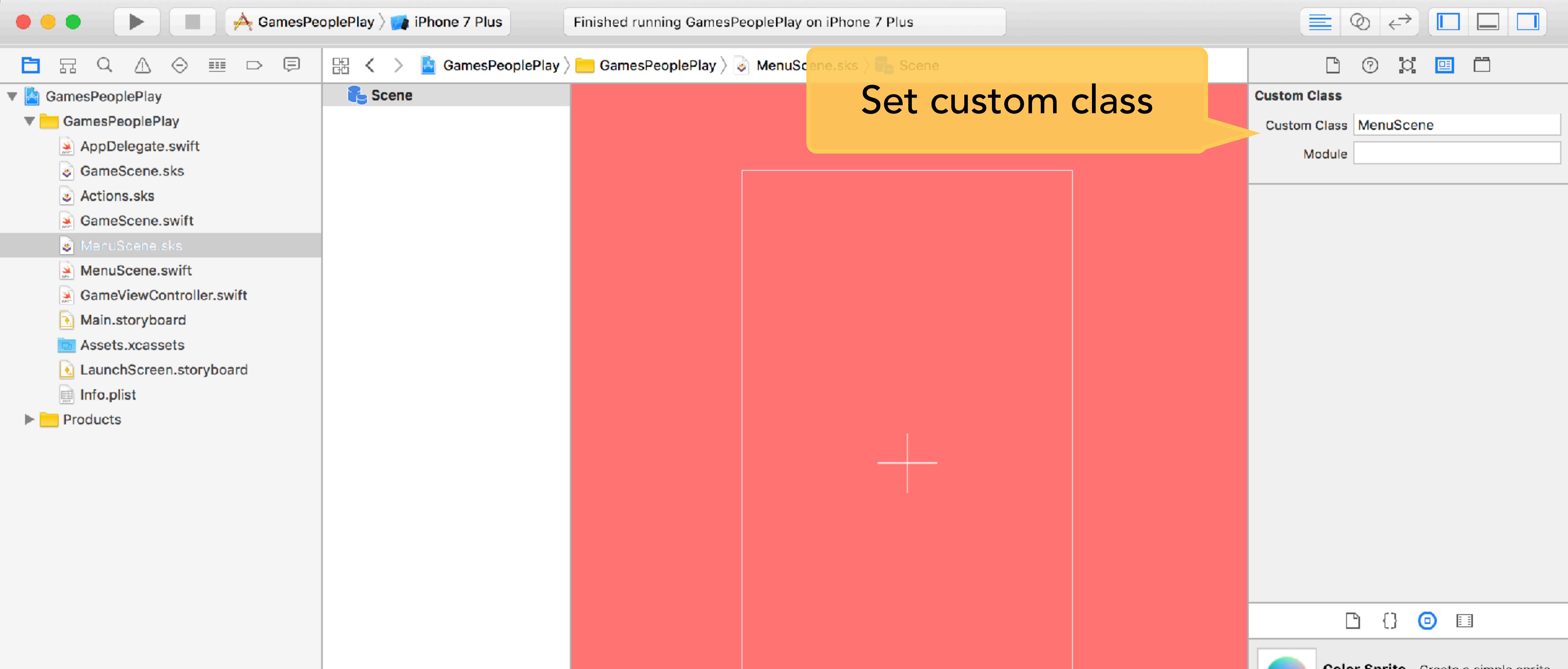
ADDING A SCENE



ADDING A SCENE



ADDING A SCENE



ADDING A SCENE

GamesPeoplePlay

- AppDelegate.swift
- GameScene.sks
- Actions.sks
- GameScene.swift
- MenuScene.sks
- MenuScene.swift**
- GameViewController.swift
- Main.storyboard
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist

```
// GamesPeoplePlay
//
// Created by T. Andrew Binkowski on 5/15/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import SpriteKit

class MenuScene: SKScene {

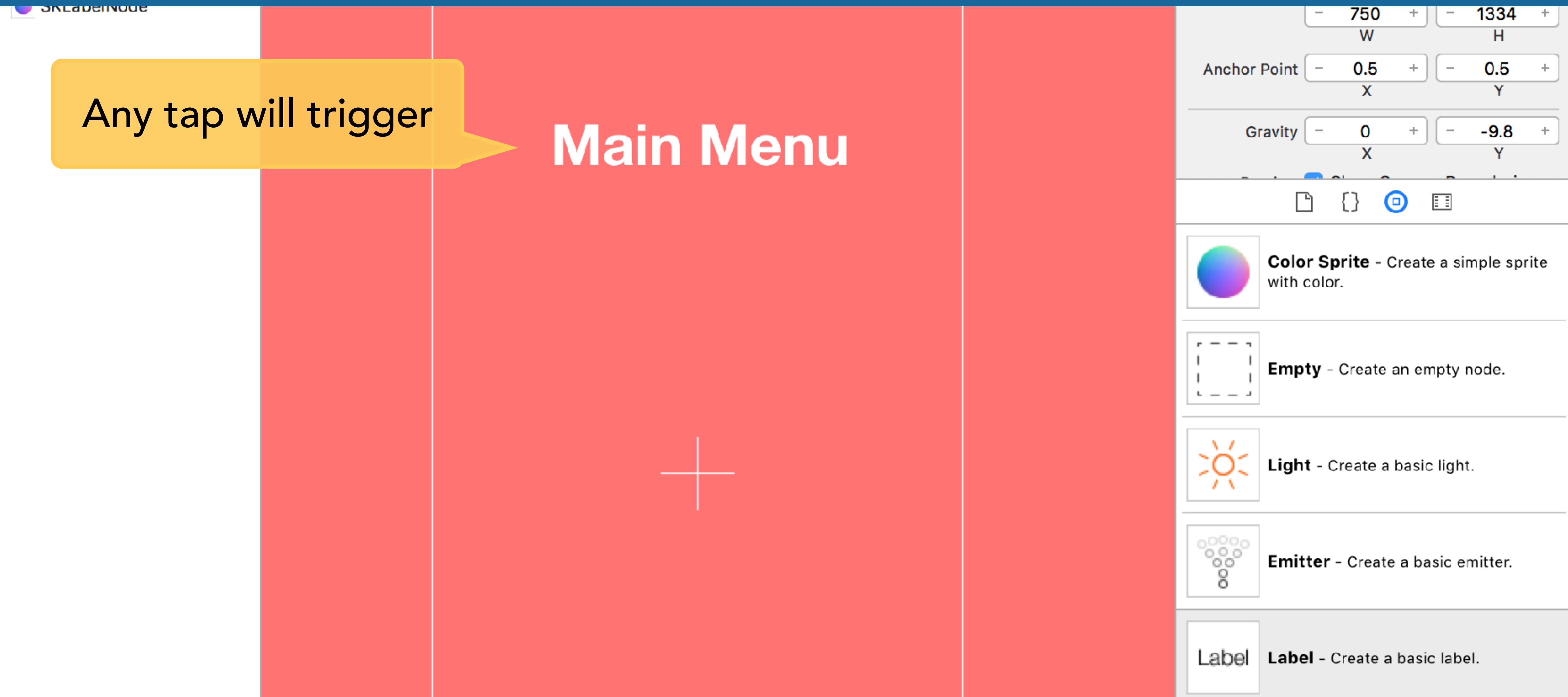
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        if let view = self.view {
            // Load the SKScene from 'GameScene.sks'

            if let scene = SKScene(fileNamed: "GameScene") {
                // Set the scale mode to scale to fit the window
                scene.scaleMode = .aspectFill

                // Present the scene
                view.presentScene(scene, transition: .crossFade(withDuration: 1.0))
            }
        }
    }
}
```

Load by identifier

ADDING A SCENE



SCENE TRANSITIONS

```
// Allow old scene's animations to continue during transition  
moveIn.pausesOutgoingScene = NO;
```

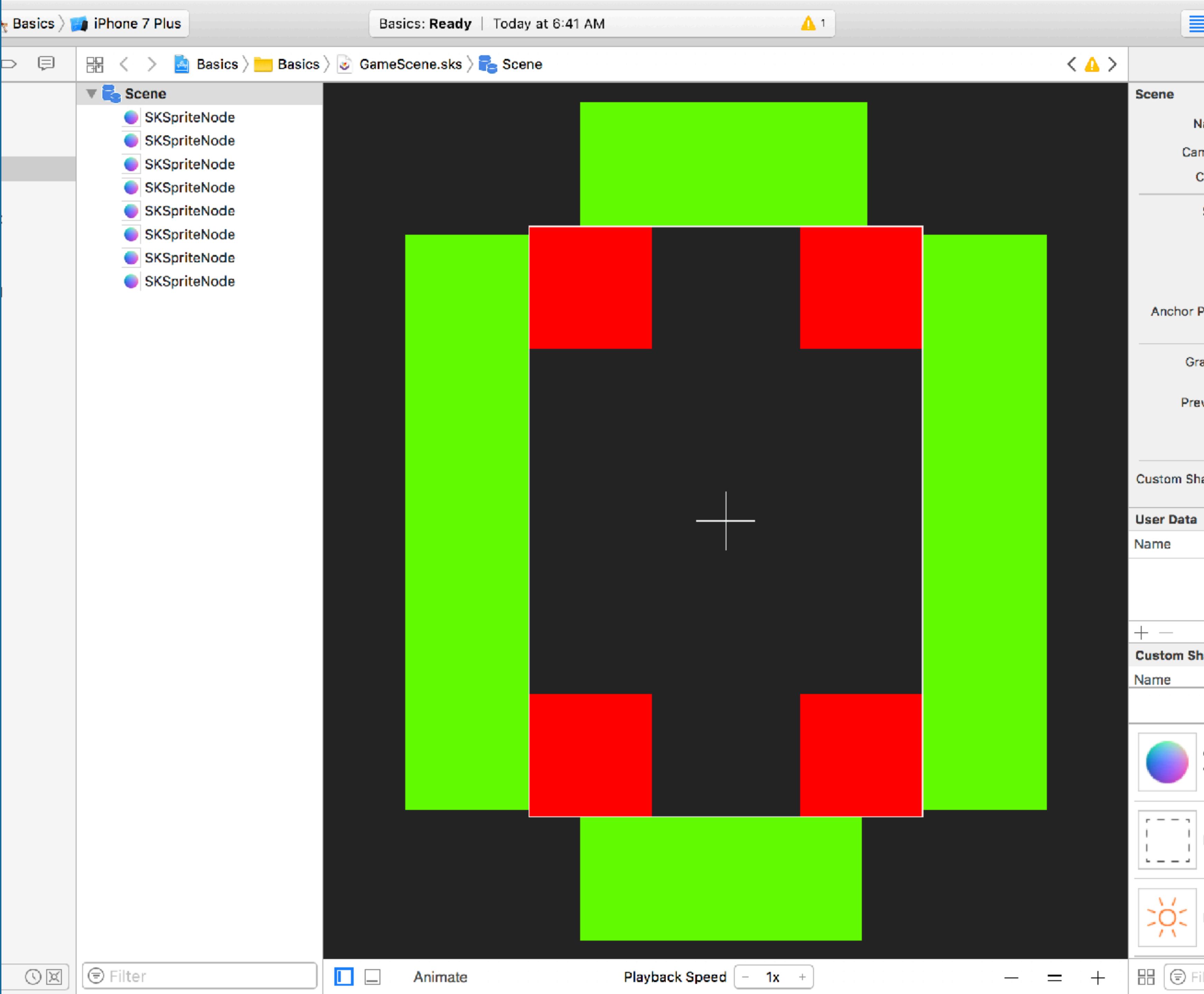
By default both scenes
will pause

- Option to pause rendering of either scene
 - Chain image based effects

SCALING MODES

SCALING MODES

- Different screen sizes need to be handled
- Automatic scaling
 - Decisions need to be made at design time



SCALING MODES

- `scaleMode`
- How the scene is mapped to the view that presents it

```
class GameViewController: UIViewController {  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        if let view = self.view as! SKView? {  
            // Load the SKScene from 'GameScene.sks'  
            if let scene = SKScene(fileNamed: "GameScene") {  
                // Set the scale mode to scale to fit the window  
                scene.scaleMode = .resizeFill  
  
                // Present the scene  
                view.presentScene(scene)  
            }  
  
            view.ignoresSiblingOrder = true  
  
            view.showsFPS = true  
            view.showsNodeCount = true  
        }  
  
        override var shouldAutorotate: Bool {  
            return true  
        }  
  
        override var supportedInterfaceOrientations: UIInterfaceOrientationMask {  
            if UIDevice.current.userInterfaceIdiom == .phone {  
                return .allButUpsideDown  
            } else {  
                return .all  
            }  
        }  
    }  
}
```

SCALING MODES

case `fill`

Each axis of the scene is scaled independently so that each axis in the scene exactly maps to the length of that axis in the view.

Stretch

case `aspectFill`

The scaling factor of each dimension is calculated and the larger of the two is chosen. Each axis of the scene is scaled by the same scaling factor. This guarantees that the entire area of the view is filled but may cause parts of the scene to be cropped.

Cropped

case `aspectFit`

The scaling factor of each dimension is calculated and the smaller of the two is chosen. Each axis of the scene is scaled by the same scaling factor. This guarantees that the entire scene is visible but may require letterboxing in the view.

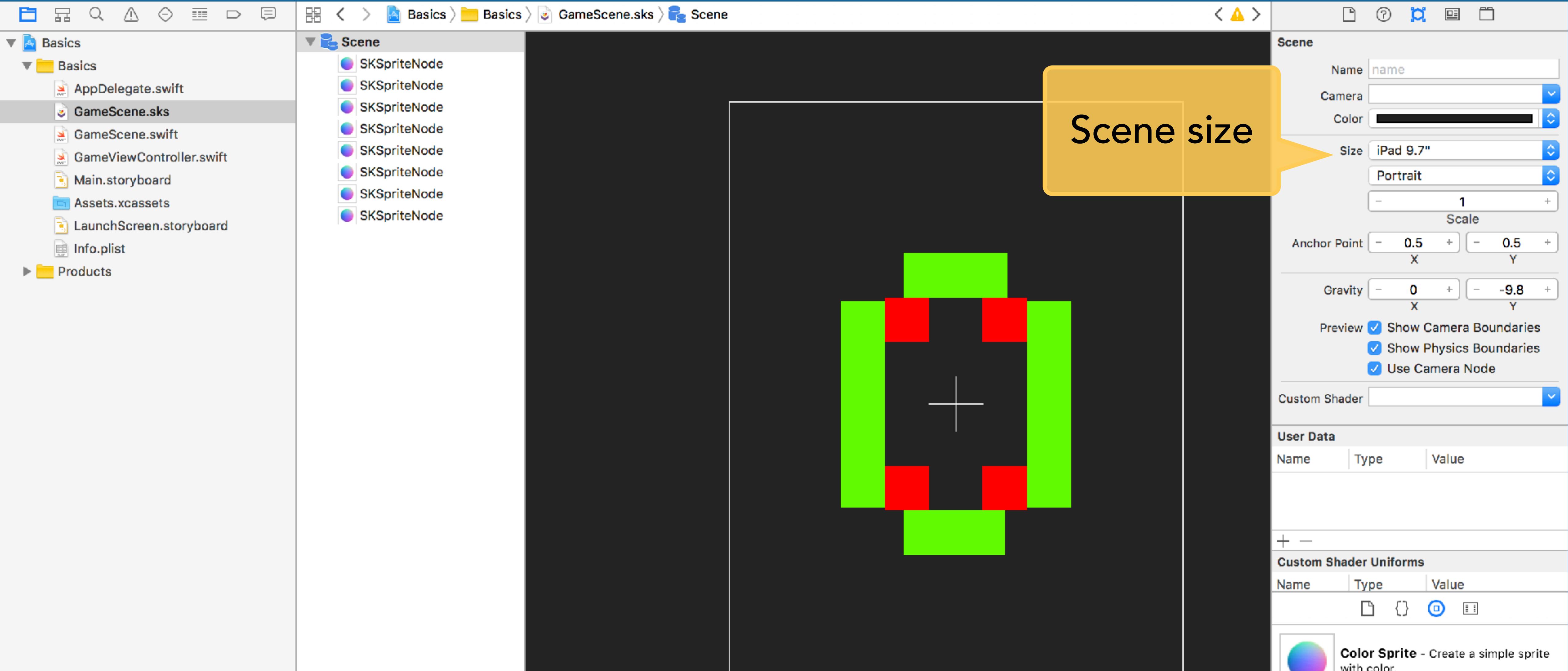
Letterbox

case `resizeFill`

The scene is not scaled to match the view. Instead, the scene is automatically resized so that its dimensions always match those of the view.

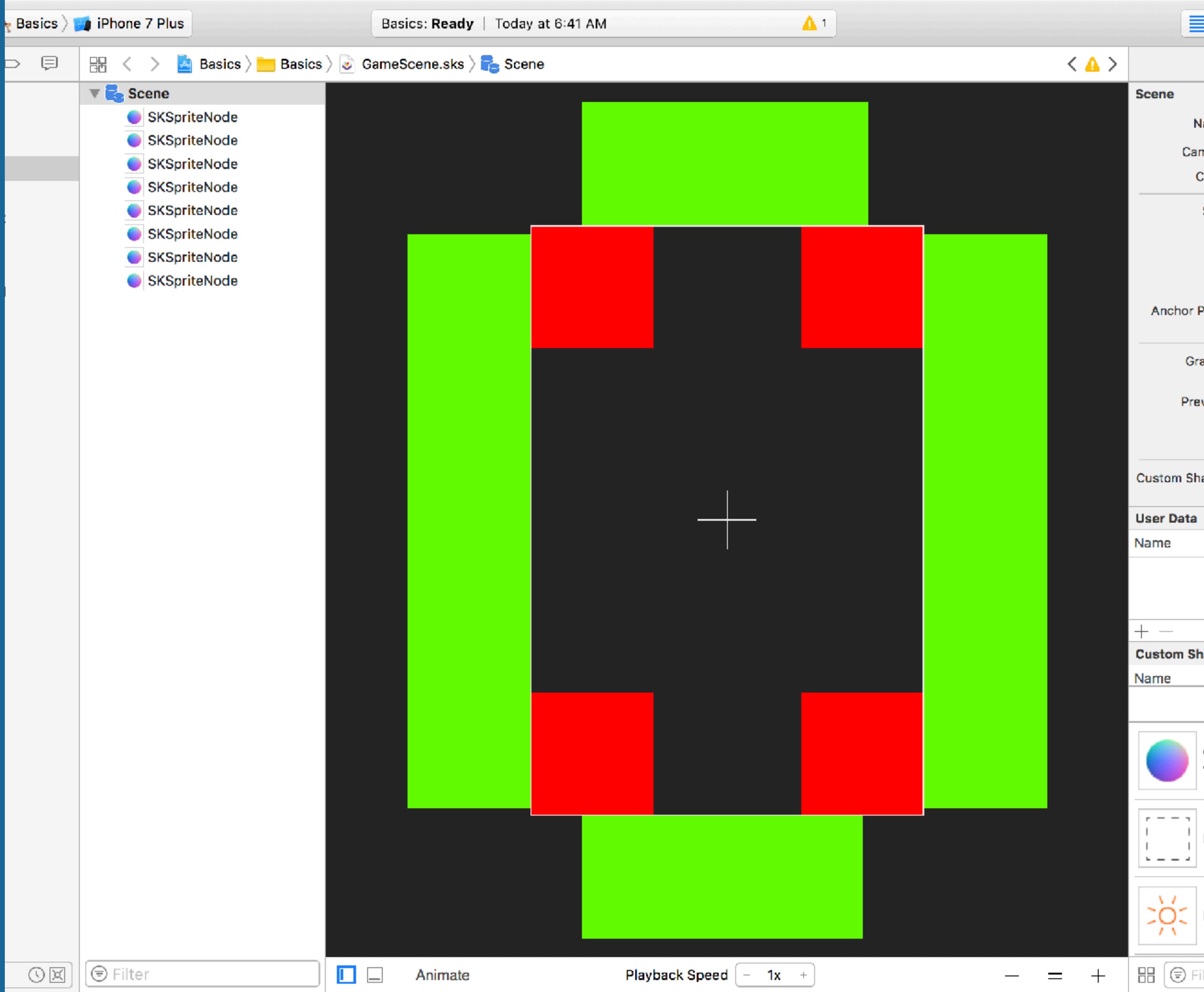
Letterbox

SCALING MODES



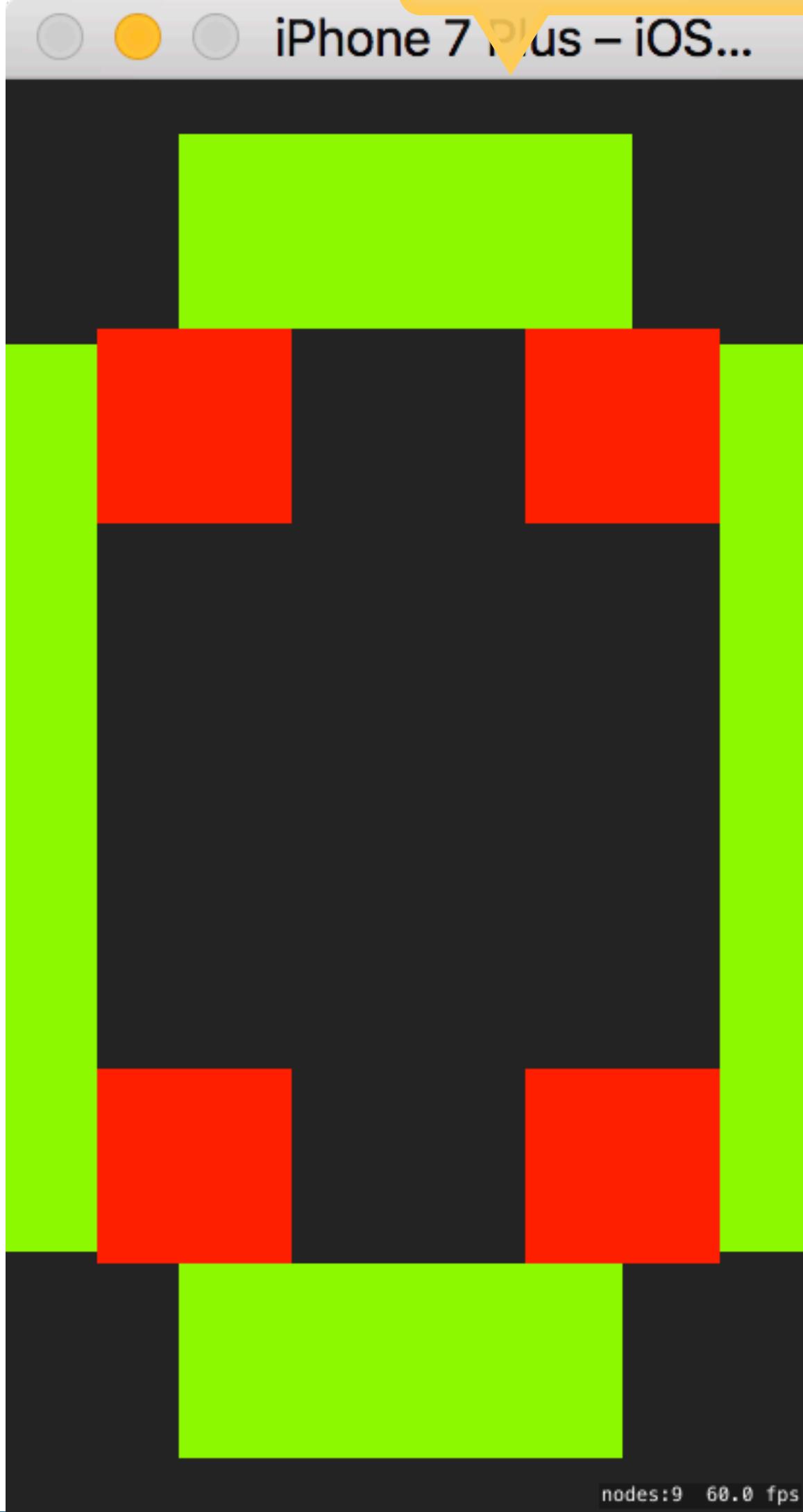
SCALING MODES

- iPhone 4s - red squares



SCALING MODES

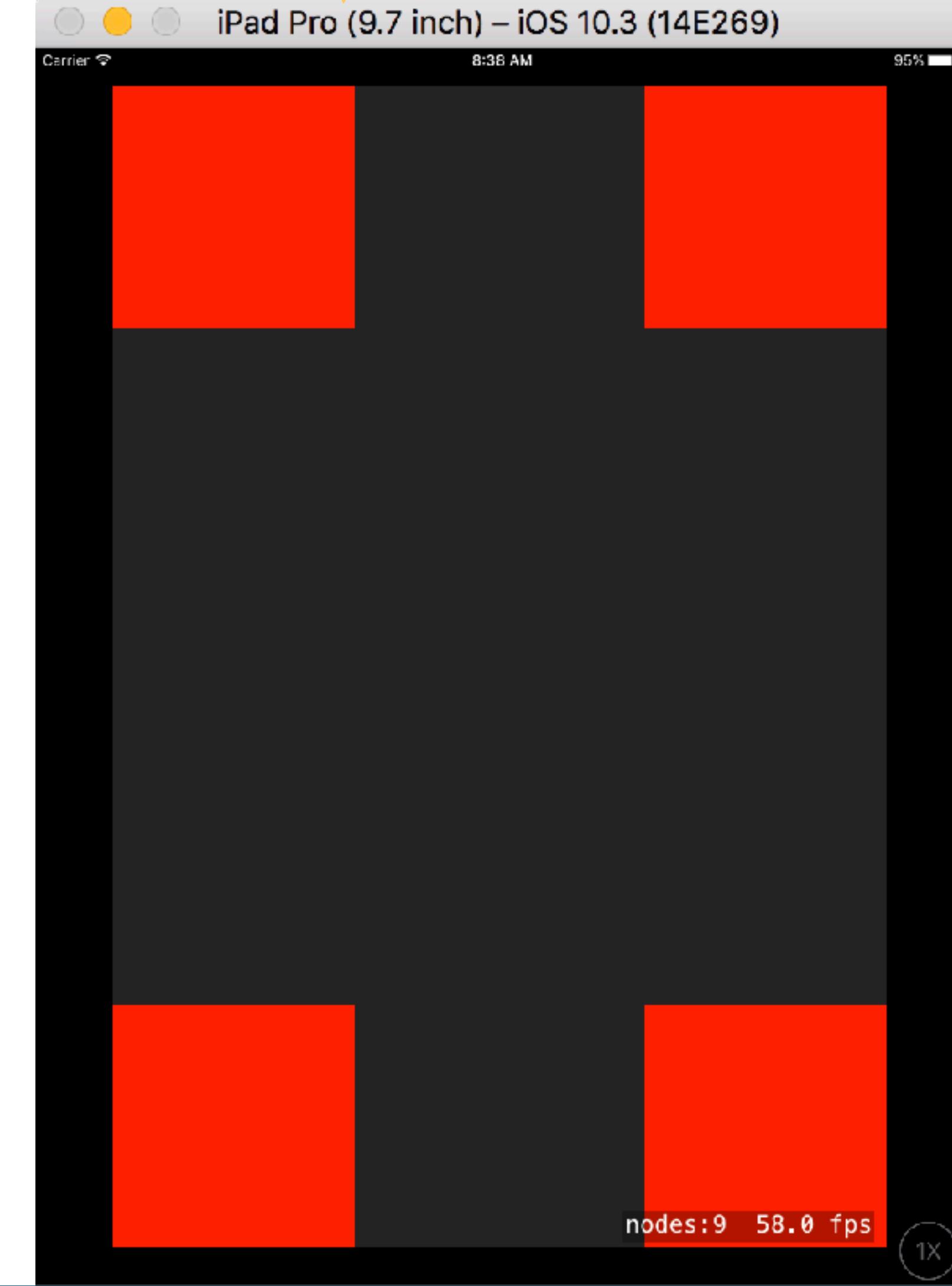
4s resizeFill on 7+



4s resizeFill on SE



4s on iPad



SCALING MODES

- `scaleMode`
- How the scene is mapped to the view that presents it

```
class GameViewController: UIViewController {  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        if let view = self.view as! SKView? {  
            // Load the SKScene from 'GameScene.sks'  
            if let scene = SKScene(fileNamed: "GameScene") {  
                // Set the scale mode to scale to fit the window  
                scene.scaleMode = .resizeFill  
  
                // Present the scene  
                view.presentScene(scene)  
            }  
  
            view.ignoresSiblingOrder = true  
  
            view.showsFPS = true  
            view.showsNodeCount = true  
        }  
  
        override var shouldAutorotate: Bool {  
            return true  
        }  
  
        override var supportedInterfaceOrientations: UIInterfaceOrientationMask {  
            if UIDevice.current.userInterfaceIdiom == .phone {  
                return .allButUpsideDown  
            } else {  
                return .all  
            }  
        }  
    }  
}
```

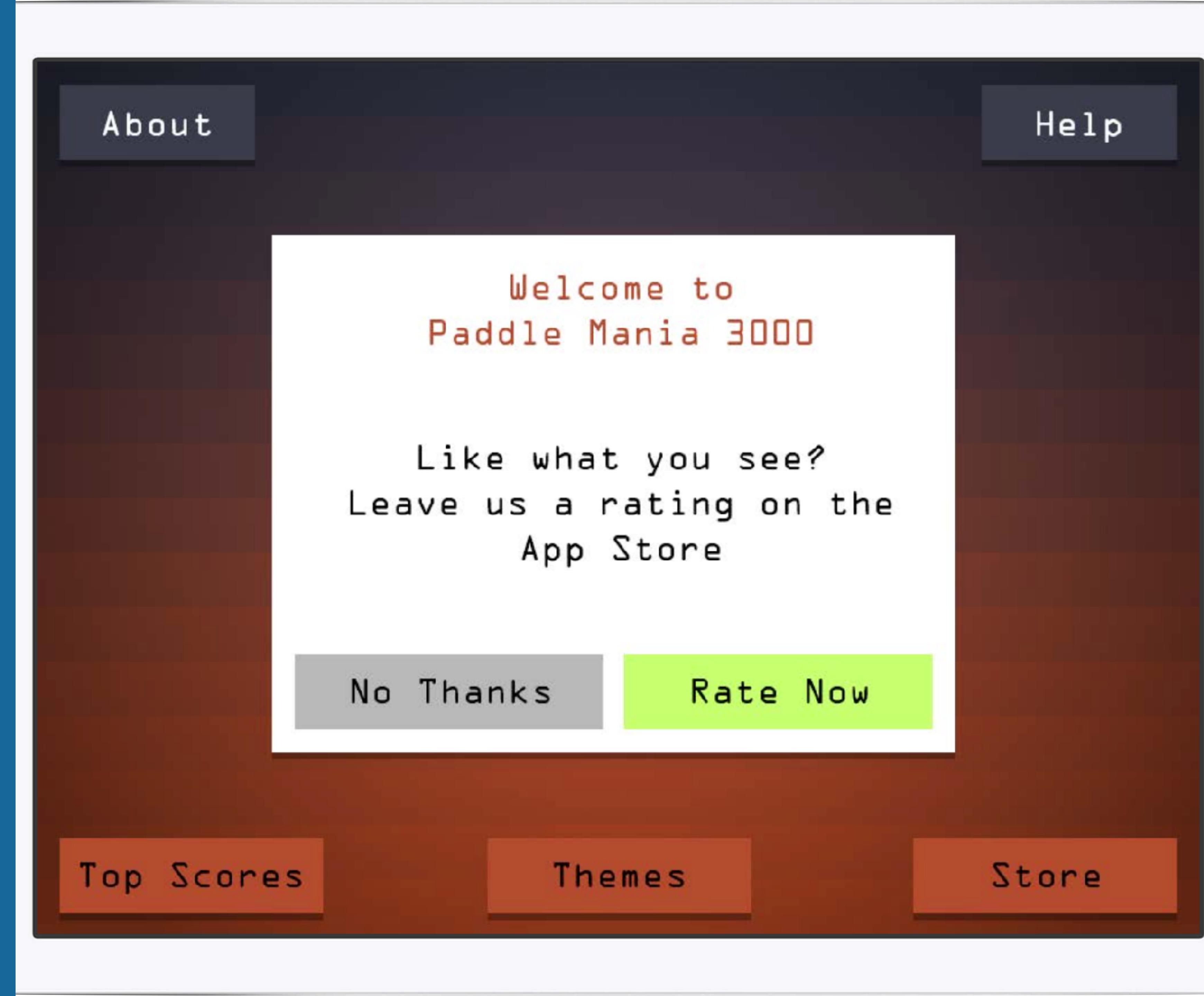
SCALING MODES



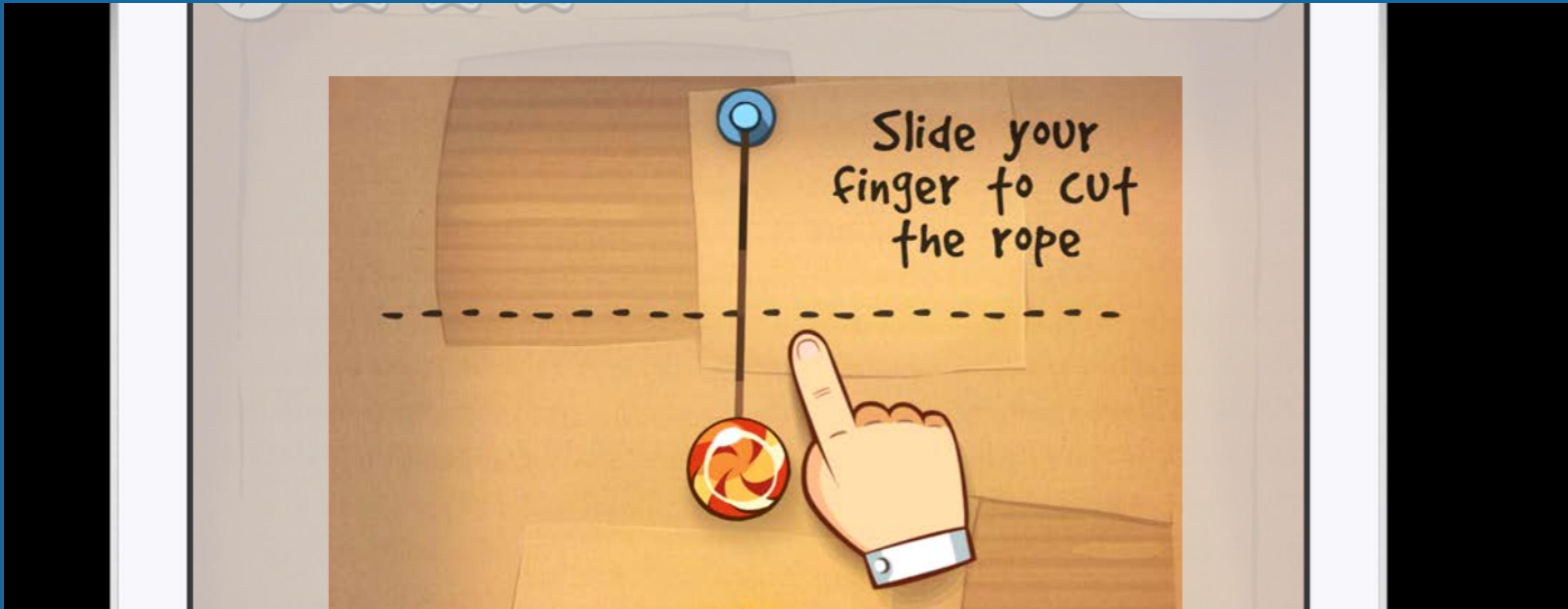
APPLE'S INGREDIENTS OF GREAT GAMES

GREAT GAMES

- Remove friction
 - On-boarding
 - Loading assets
 - Prompting for reviews

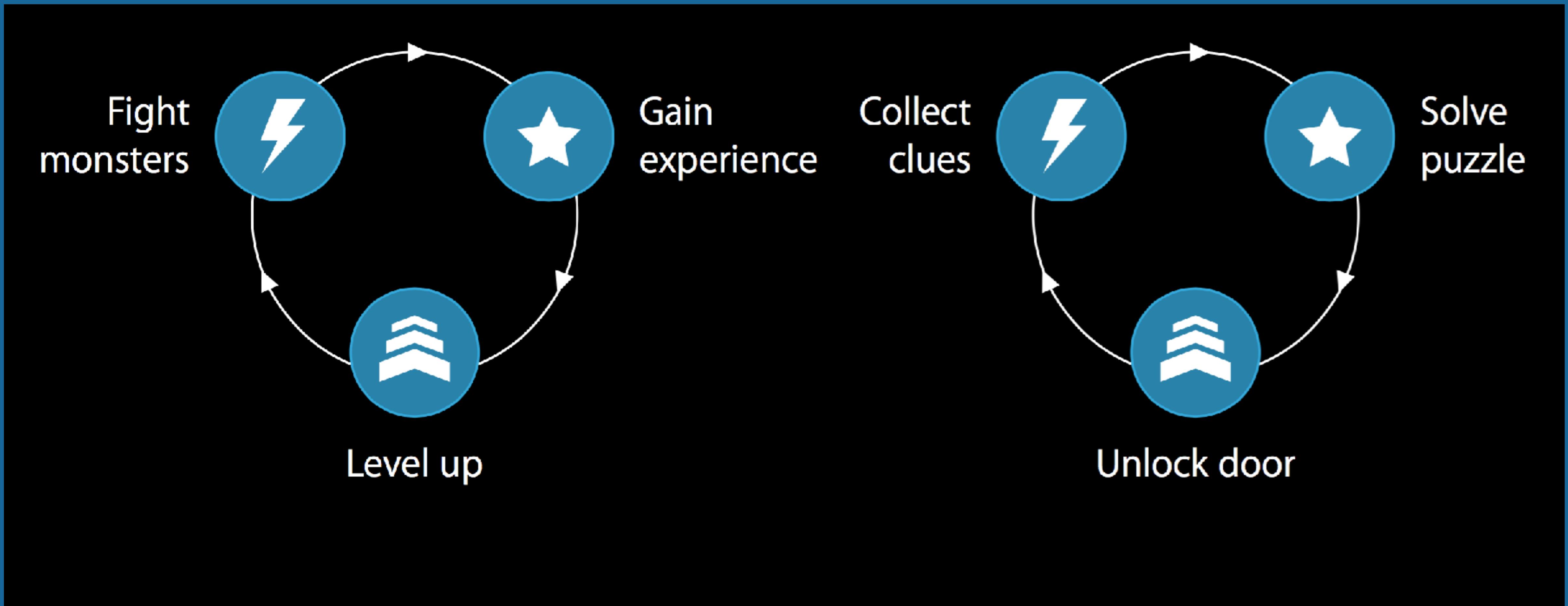


GREAT GAMES



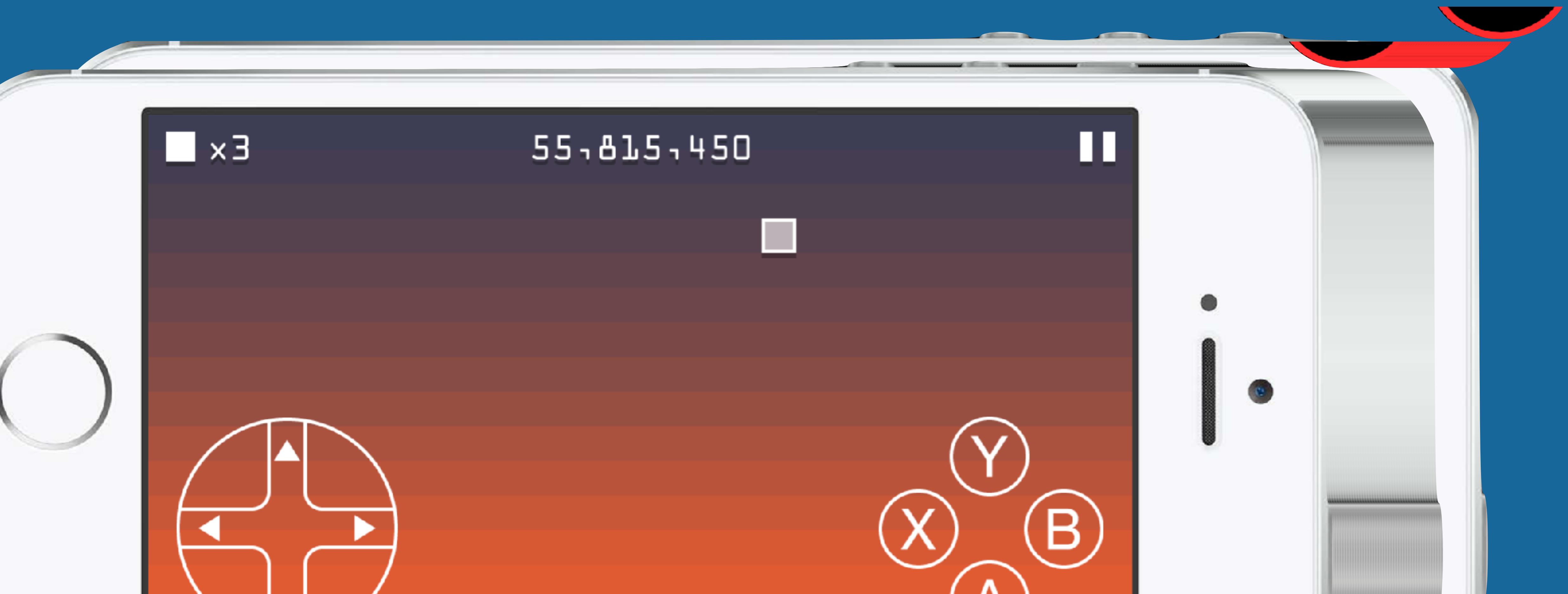
- Be a good teacher

GREAT GAMES



- Tune your core loop

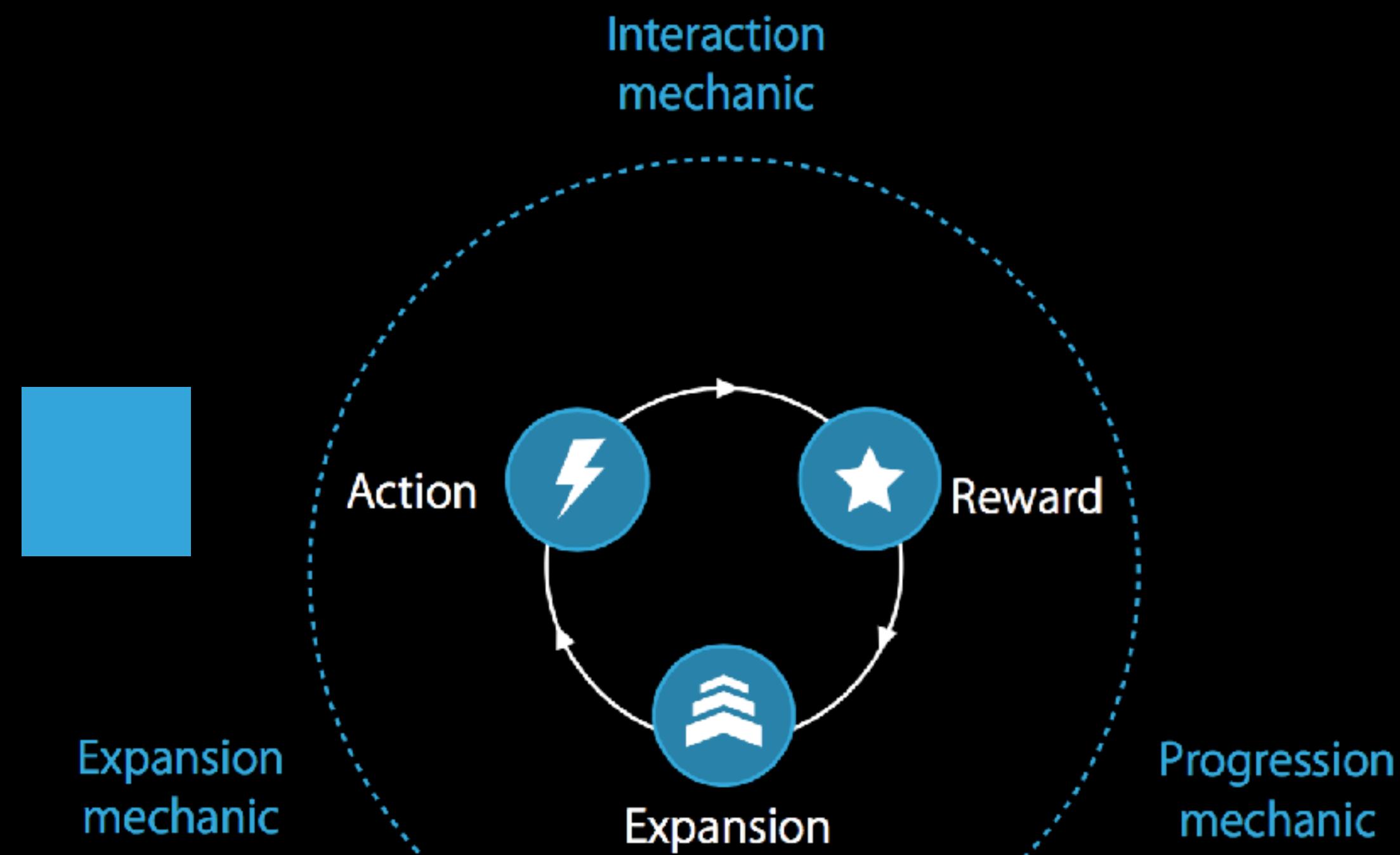
GREAT GAMES



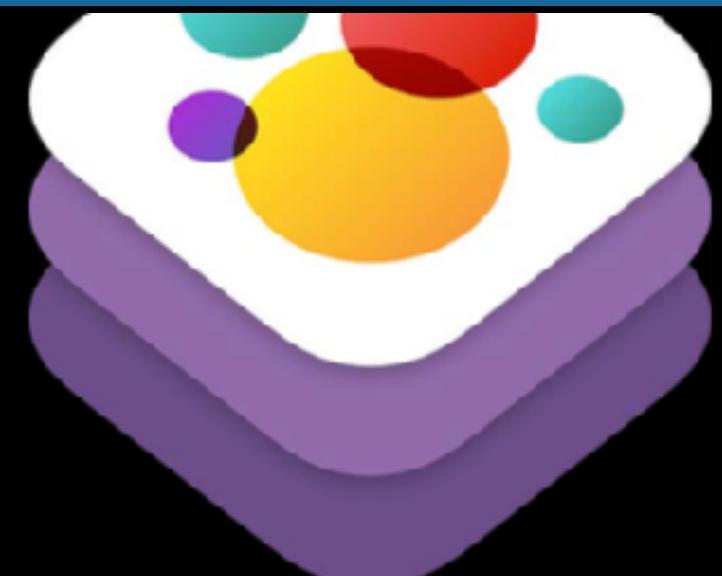
- Design for touch

GREAT GAMES

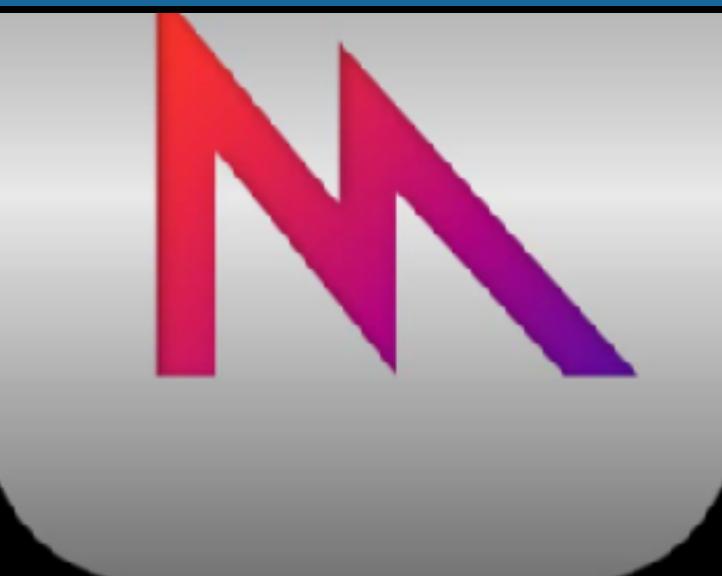
- Foster Engagement



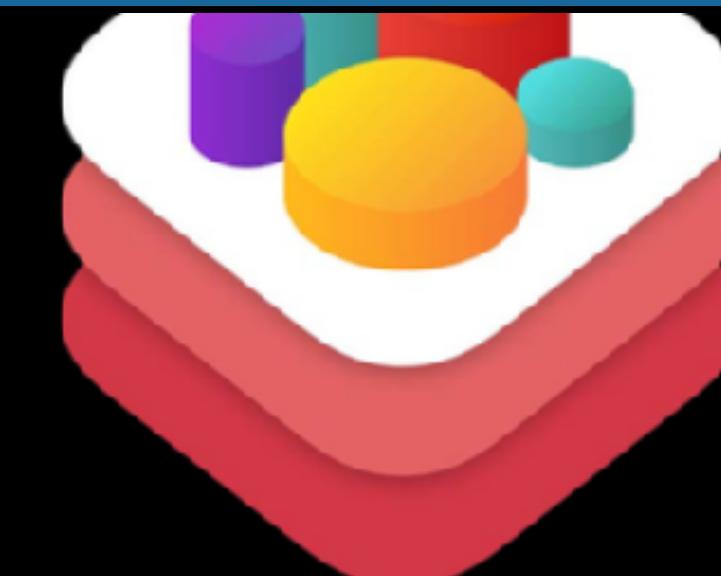
GREAT GAMES



SpriteKit



Metal



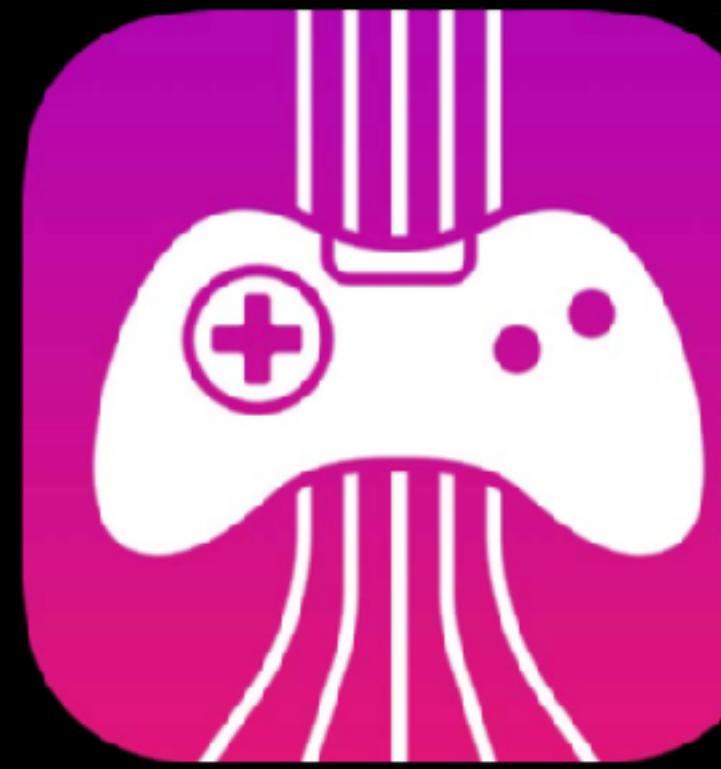
SceneKit



Game Center



OpenGL ES 3.0



Game Controllers

- Use latest technologies



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 6