THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

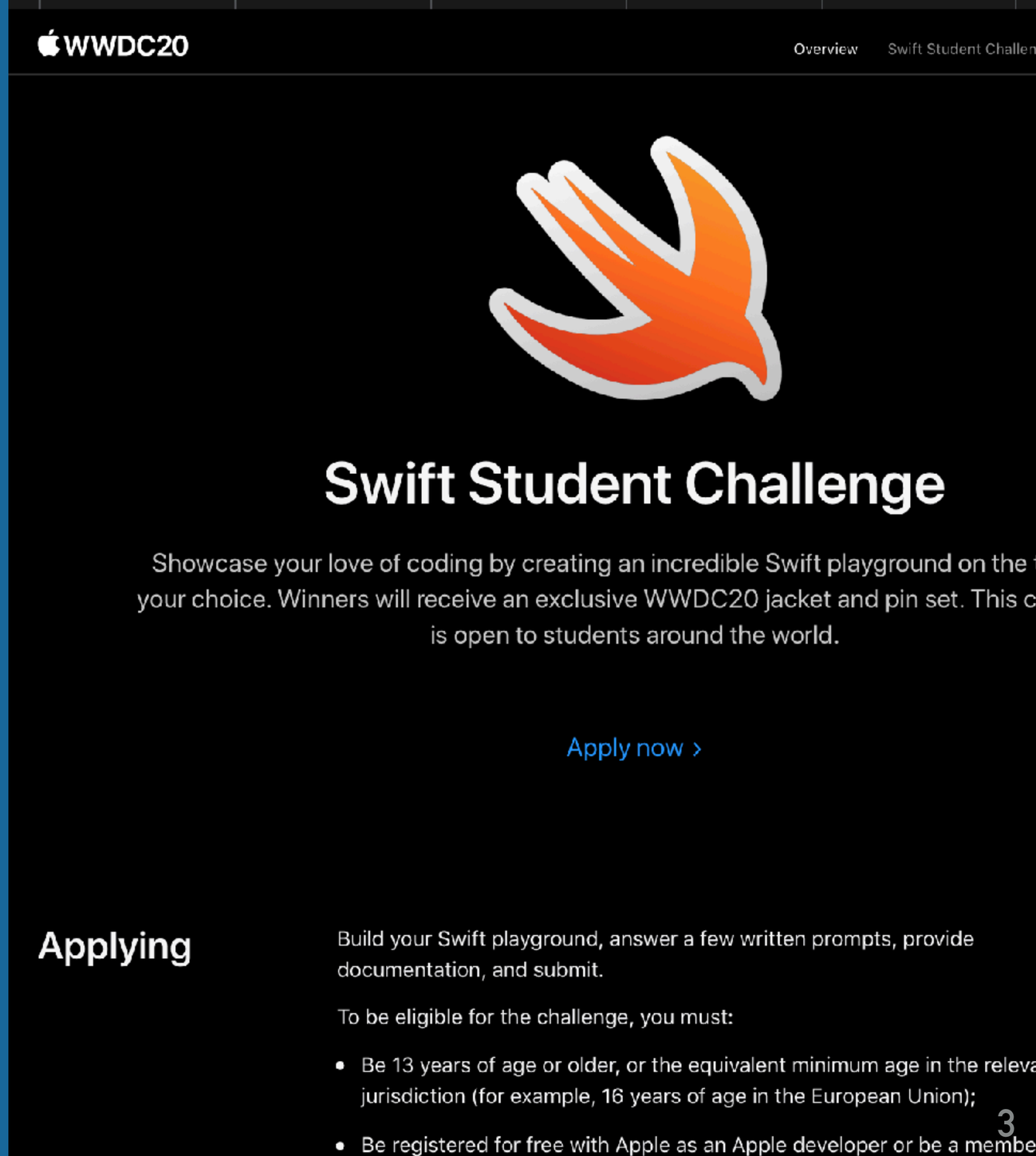
MPCS 51032 • SPRING 2020 • SESSION 6

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

- 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

- https://github.com/wwdc/2019

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.

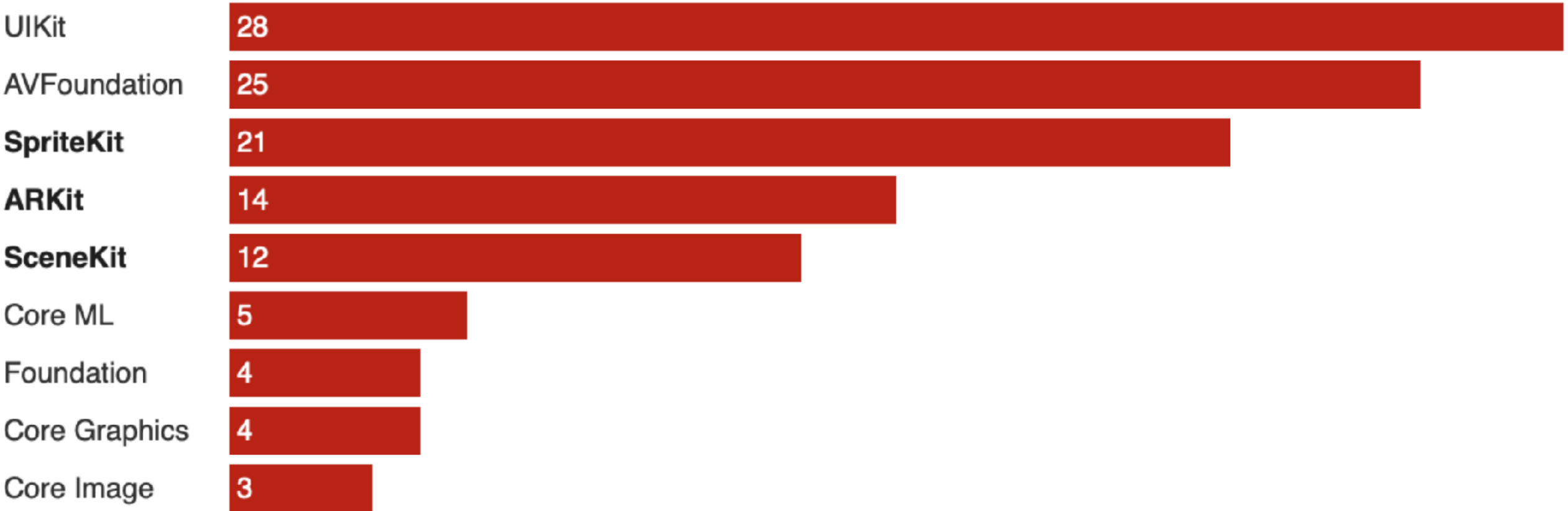
To add your own project below, just edit this file and submit a pull request! Please read the **PLEASE READ** sectio
submitting.

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

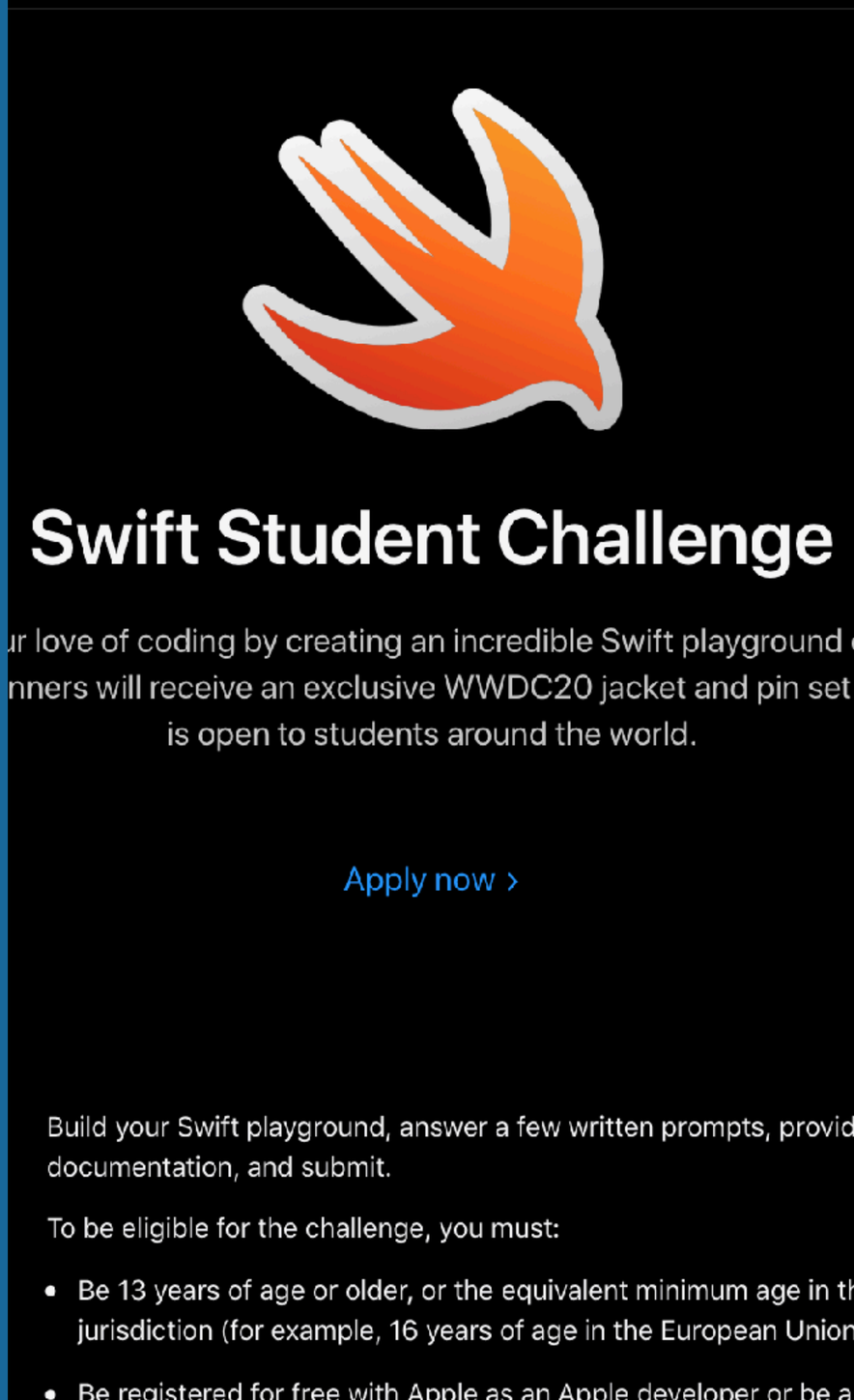
WWDC 2019 Scholarship Winners: AR & Animation

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



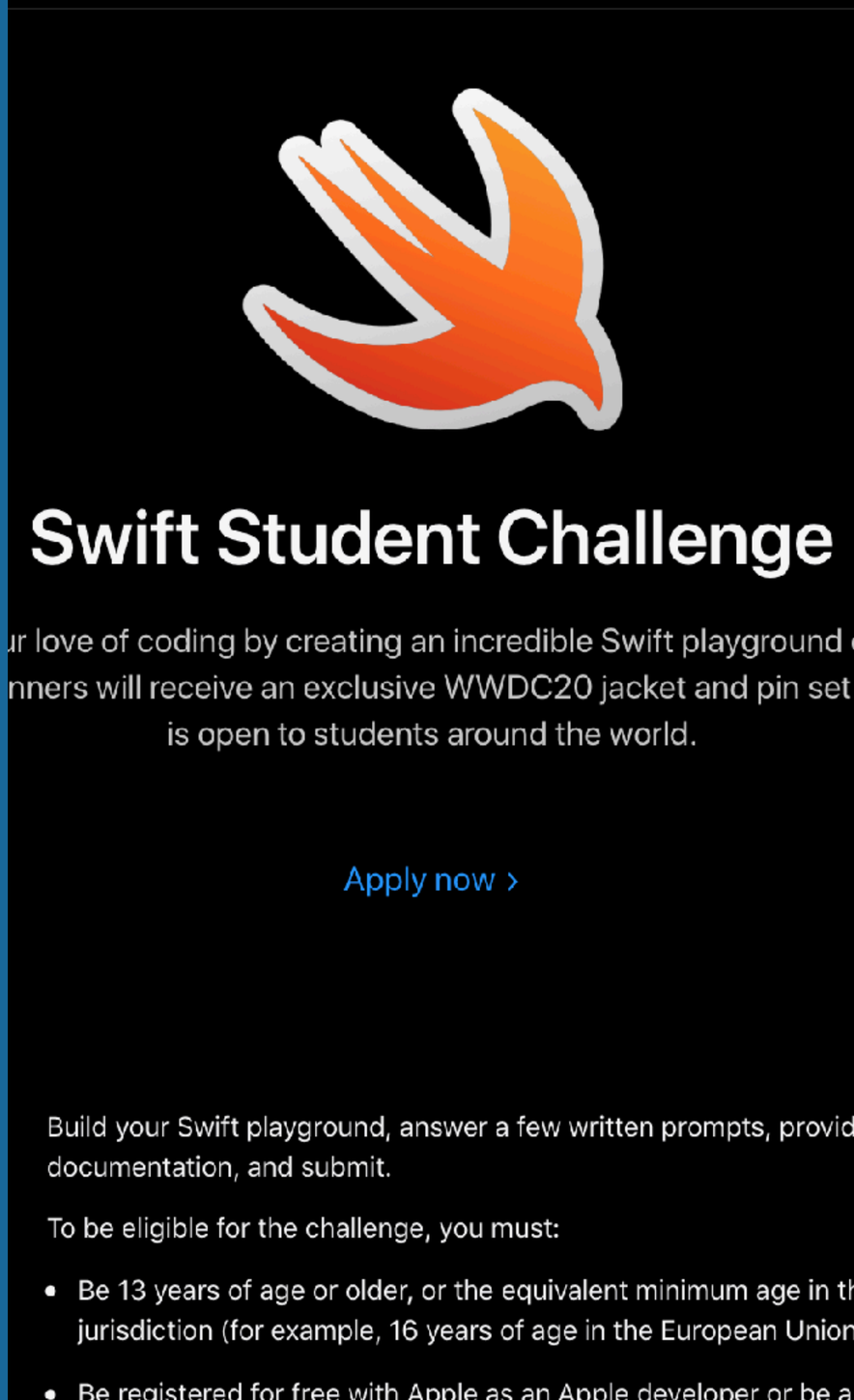
SWIFT STUDENT CHALLENGE

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



SWIFT STUDENT CHALLENGE

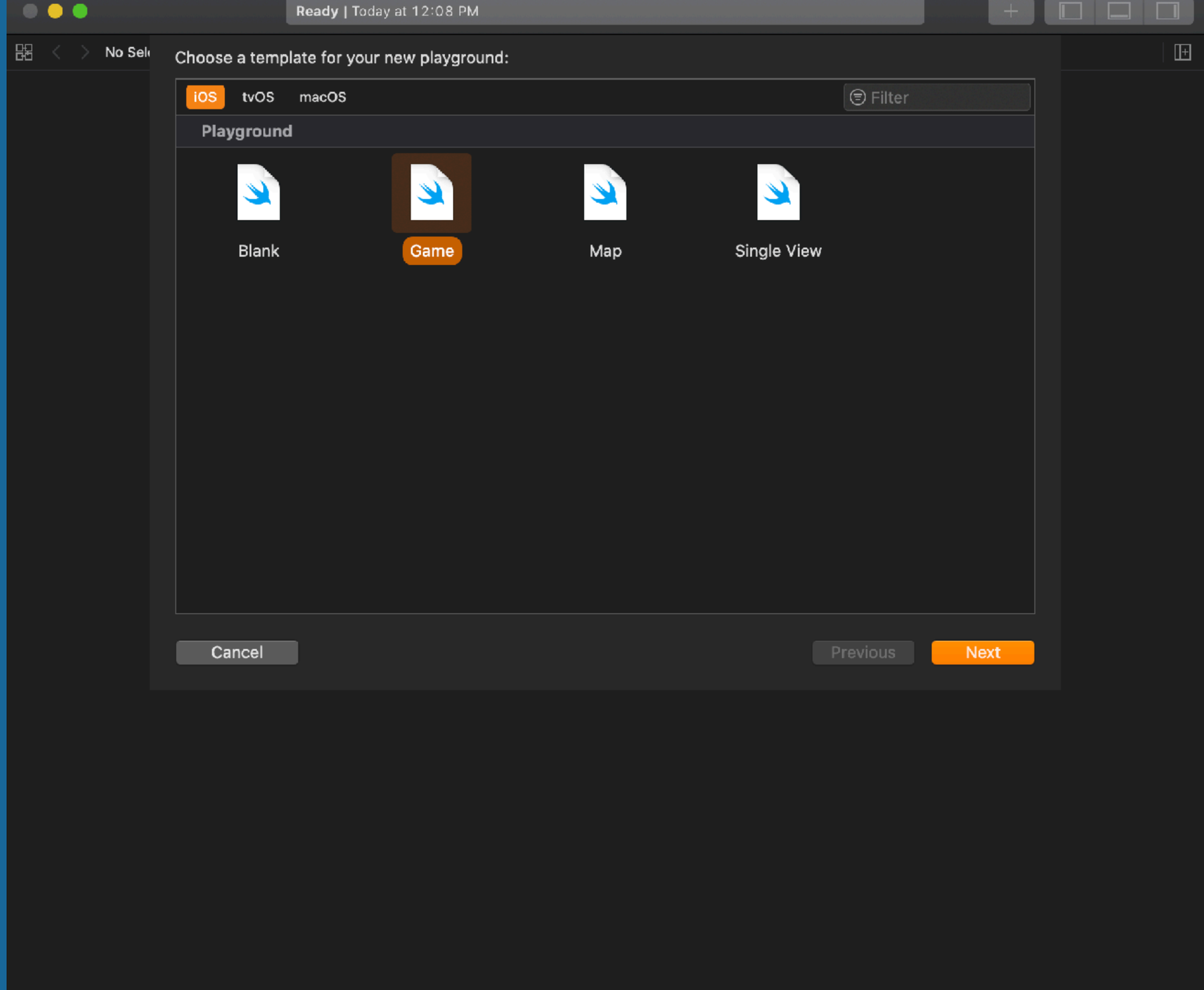
- Assignment 4
  - Create something for the Swift Student Challenge (you don't have to submit)
  - Use SpriteKit (doesn't have to be the main part of your playground)



PLAYGROUNDS

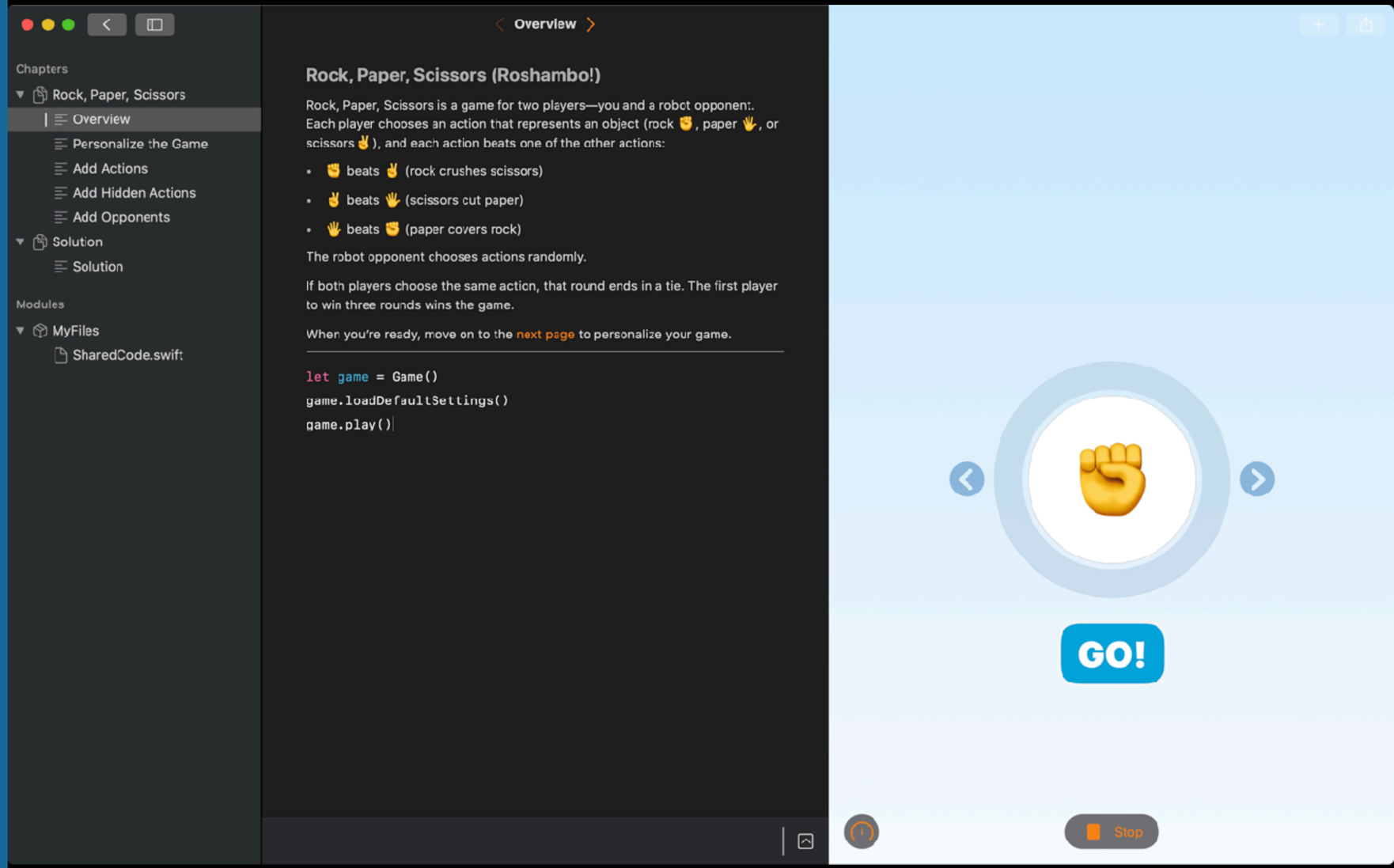
PLAYGROUNDS

- Xcode Playground
  - Templates for SpriteKit



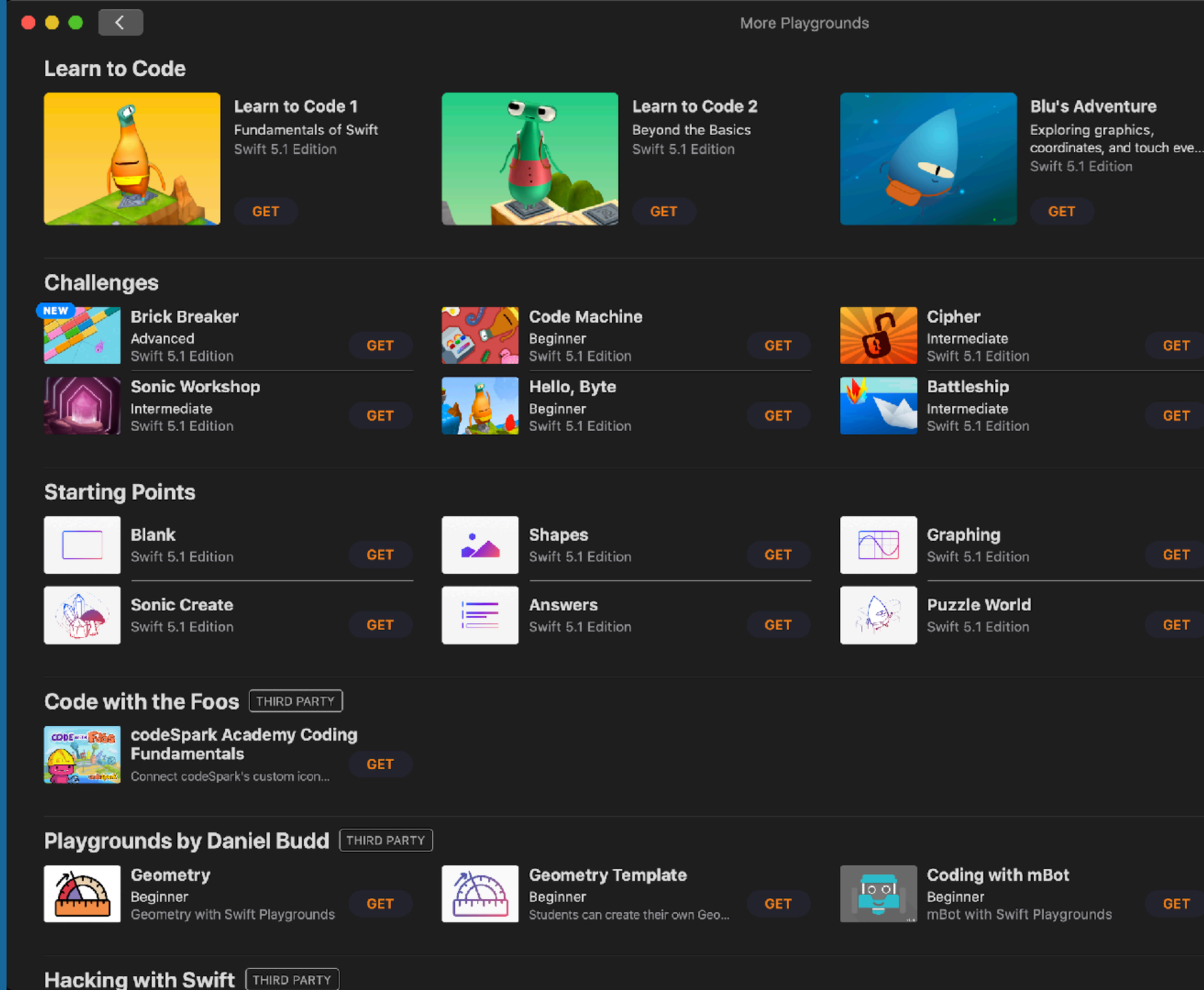
PLAYGROUNDS

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



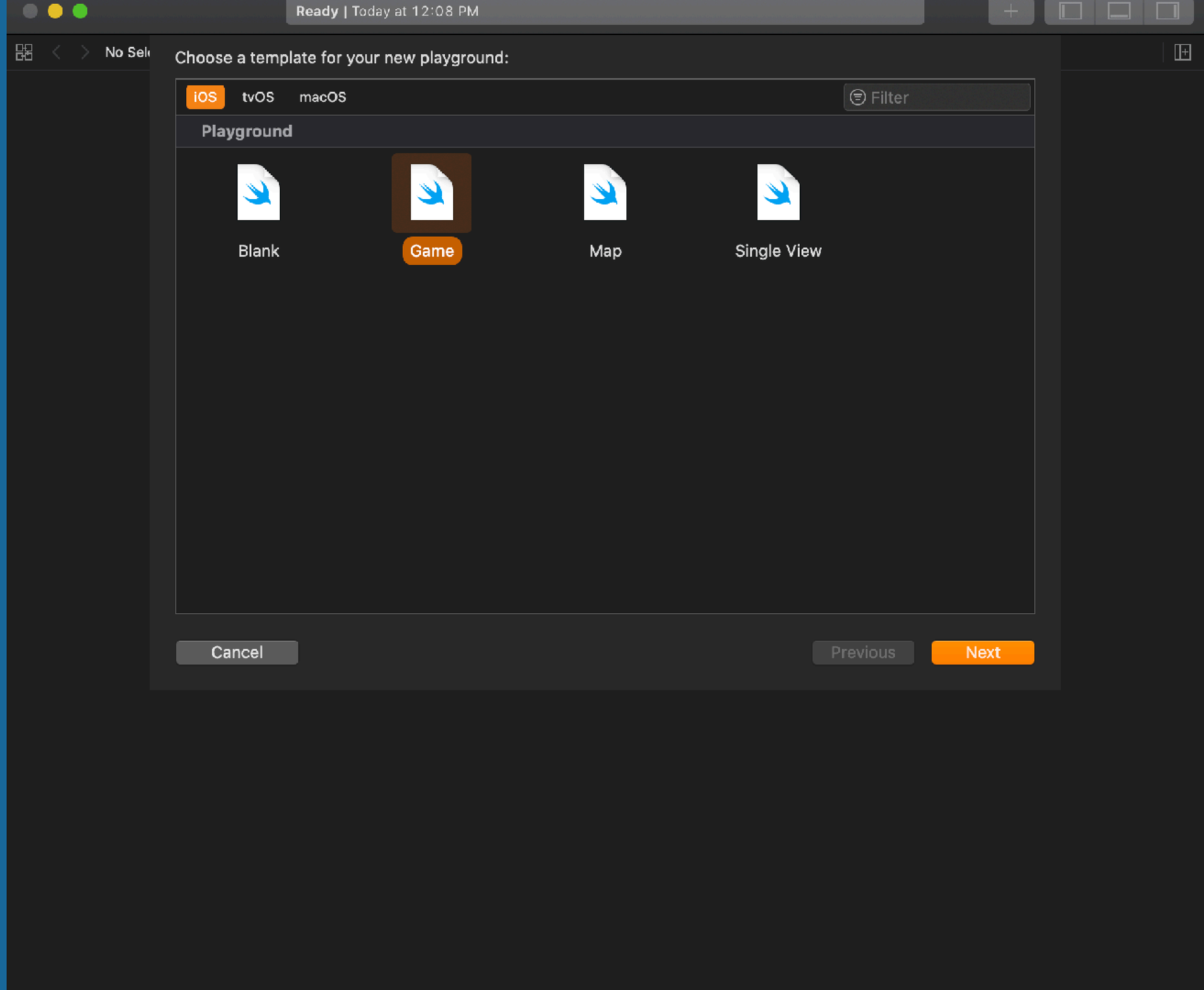
PLAYGROUNDS

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



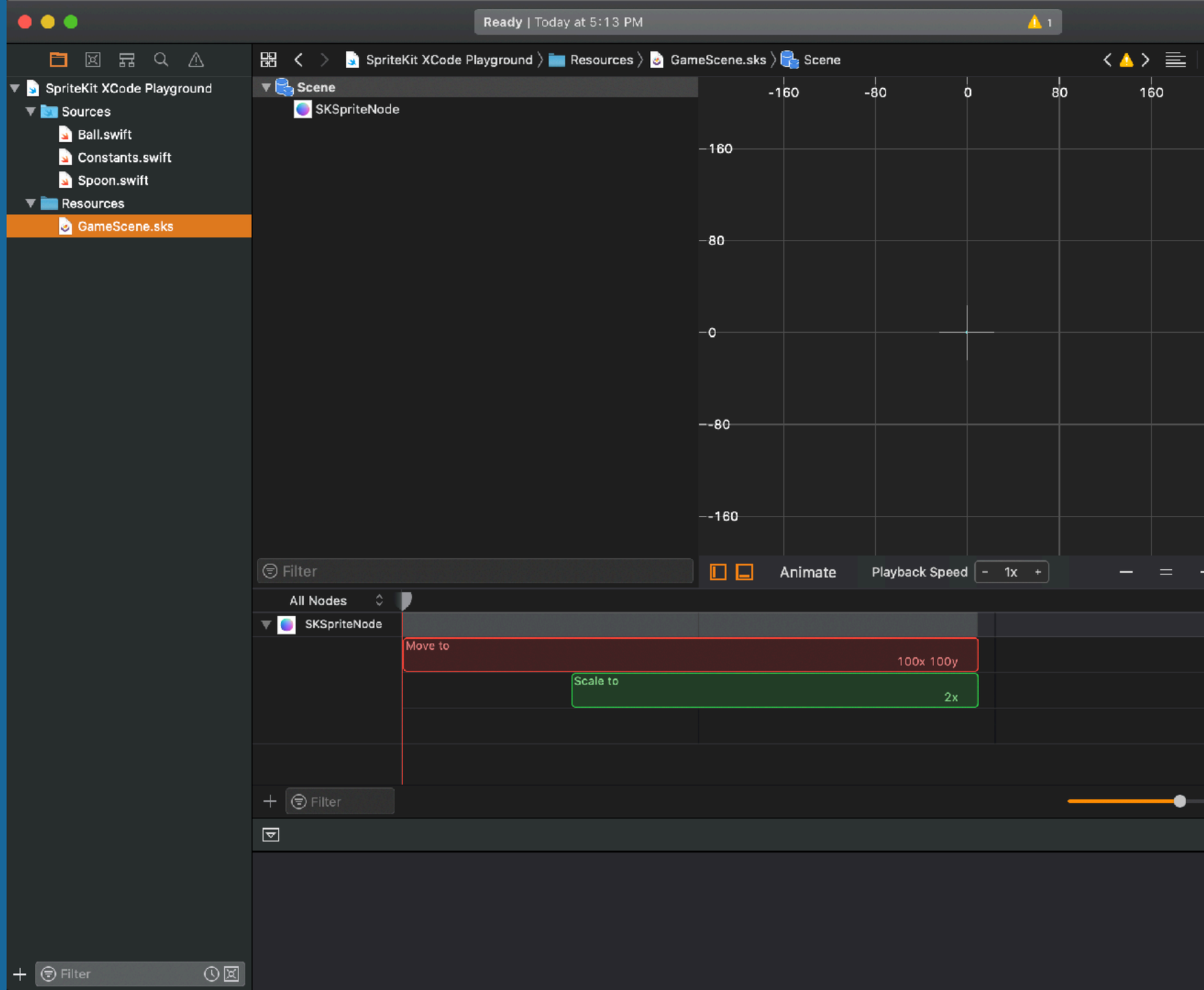
PLAYGROUNDS

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



PLAYGROUNDS

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



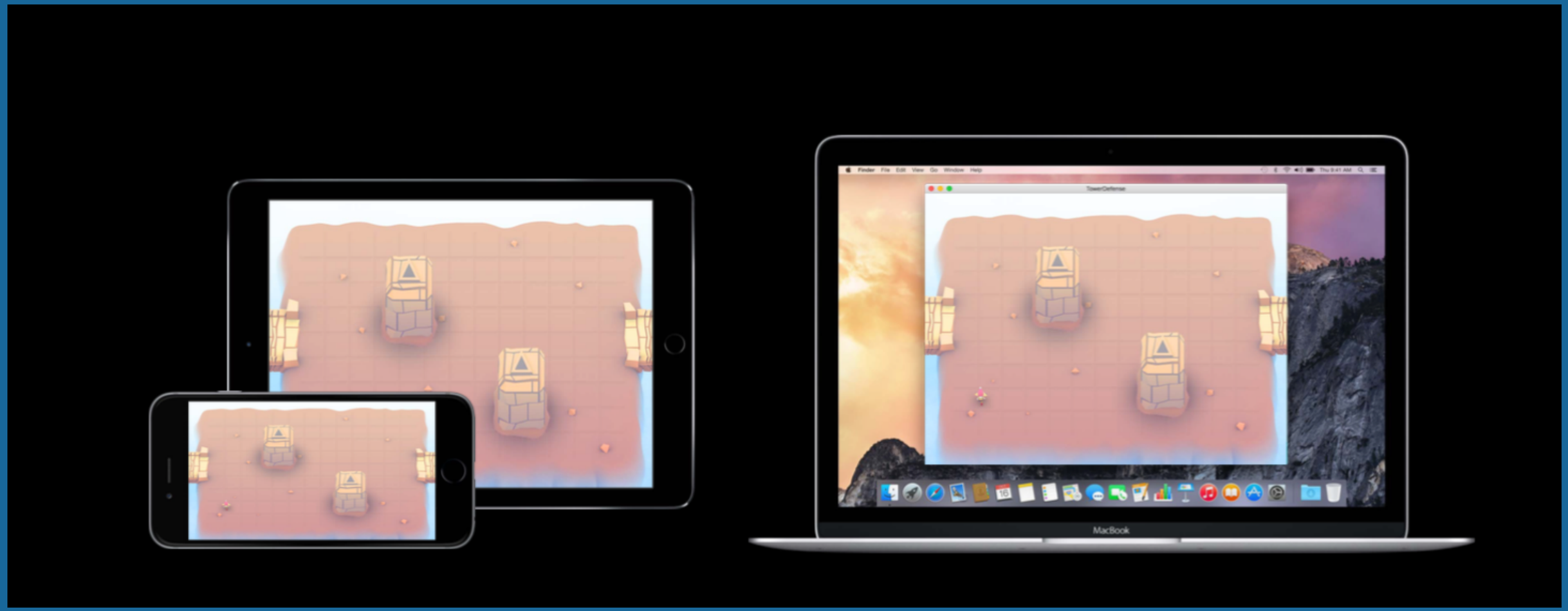
SPRITE KIT

SPRITE KIT

SUBTITLE

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

SPRITE KIT

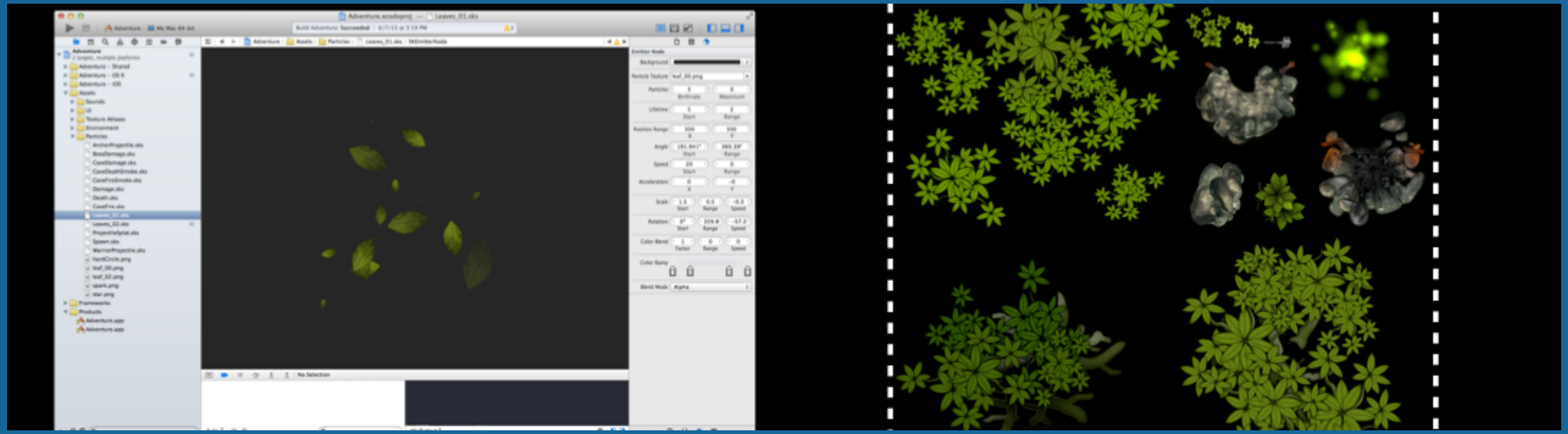


- Cross platform for iOS, tvOS and OS X

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

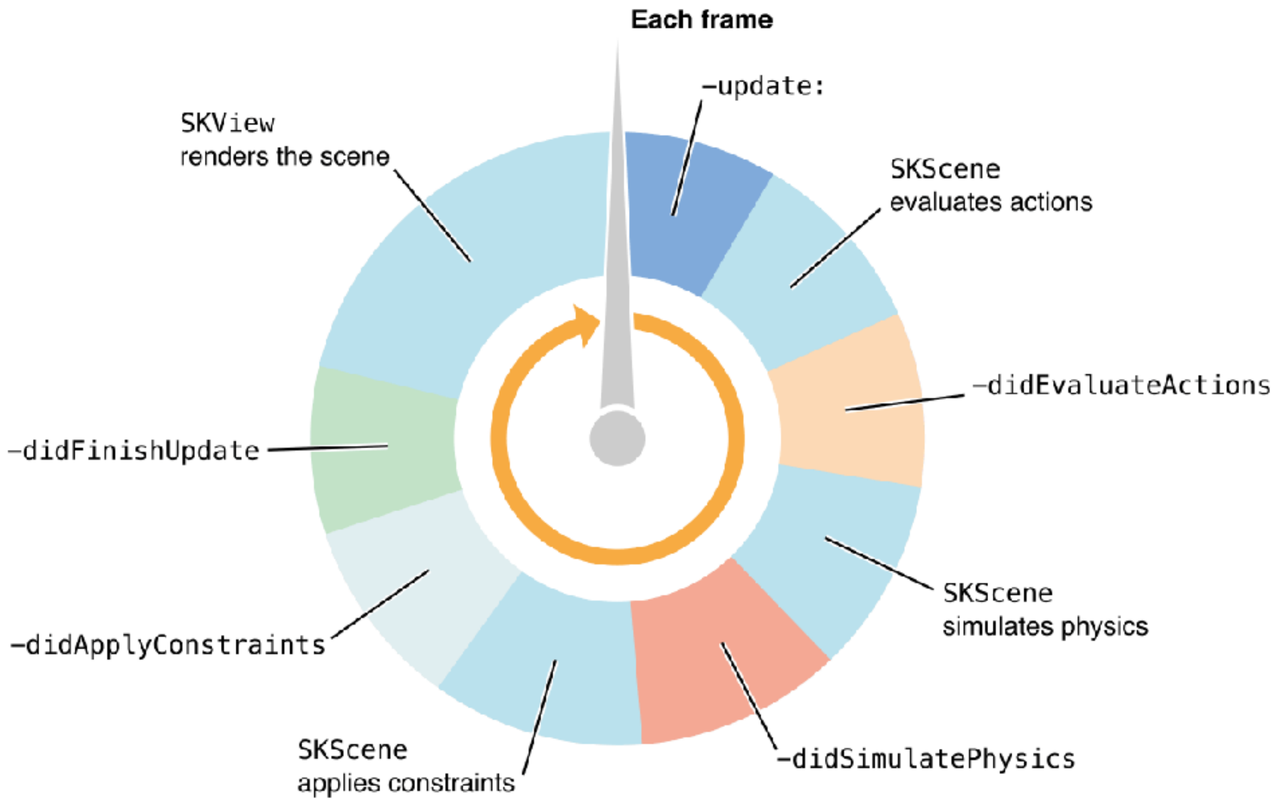
- Integrated live editor in Xcode

SPRITE KIT IN A NUTSHELL

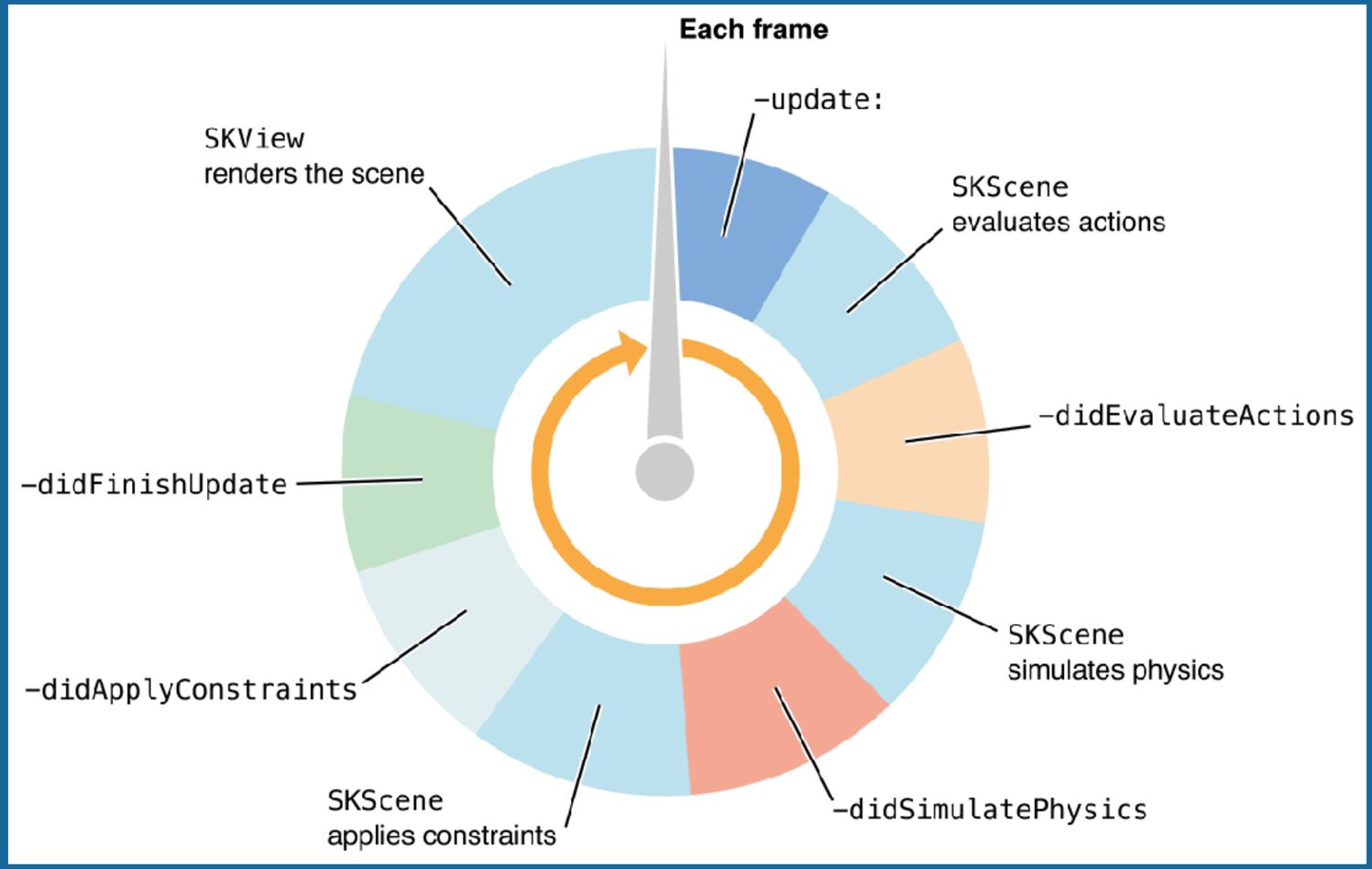
SPRITE KIT IN A NUTSHELL

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



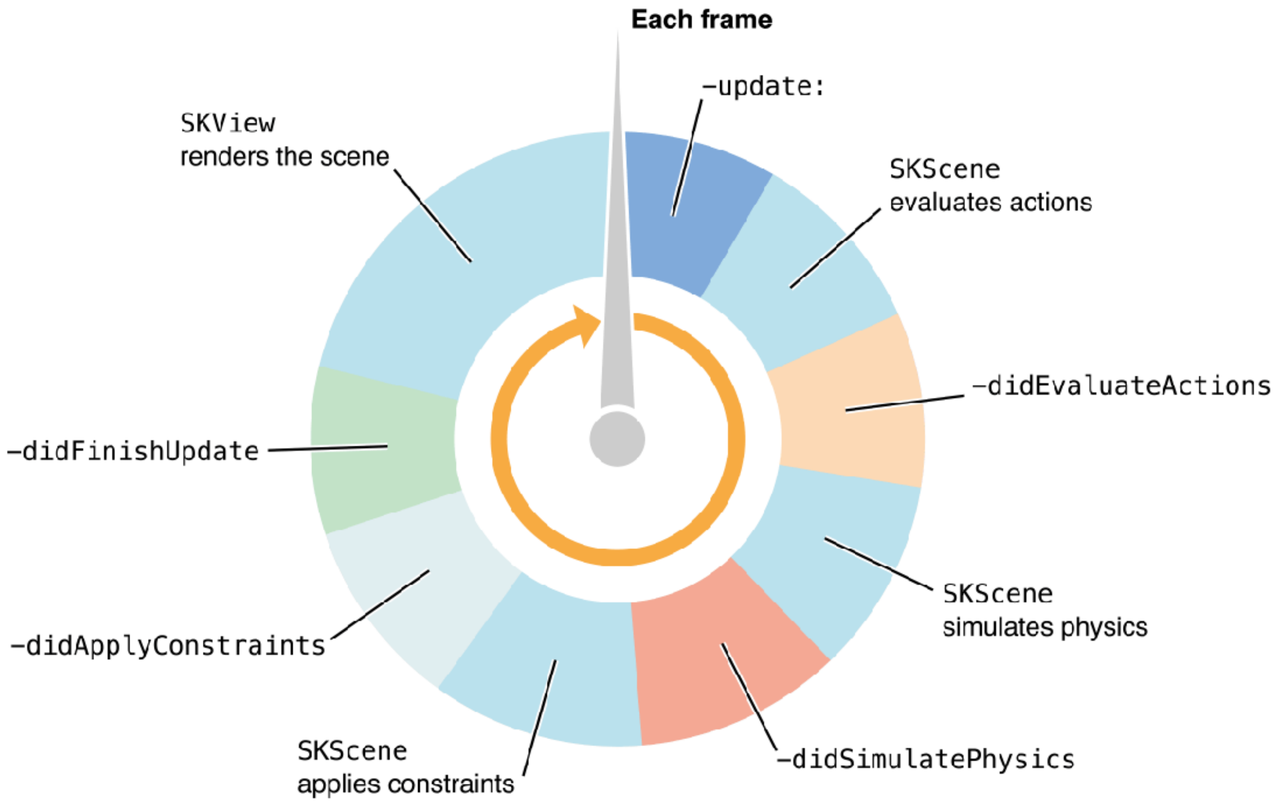
60 Frames per second is the goal

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

SPRITE KIT IN A NUTSHELL

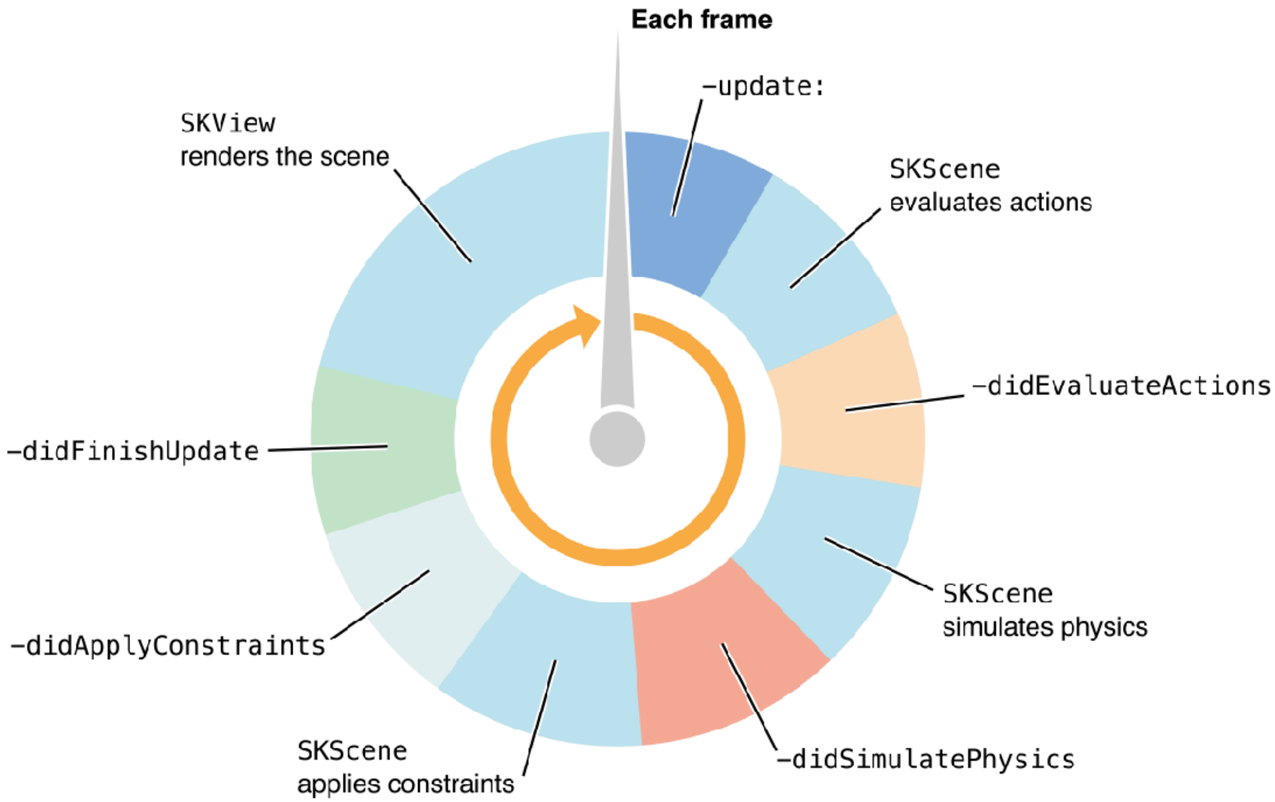
SUBTITLE

- 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

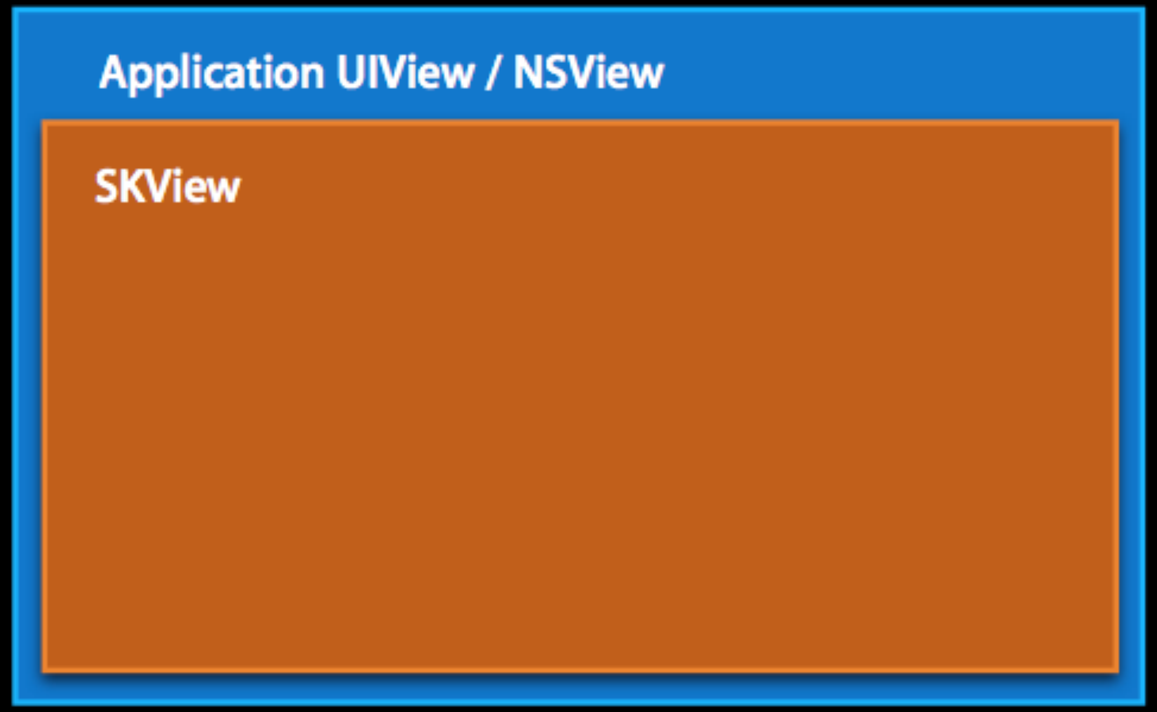
- Sprite content is drawn by presenting scenes (`SKScene`) in a sprite view (`SKView`)
  - You switch between scenes using `SKTransition`
- Physics bodies (SKPhysicsBody) and joints (SKPhysicsJoint) simulate physics in your scene

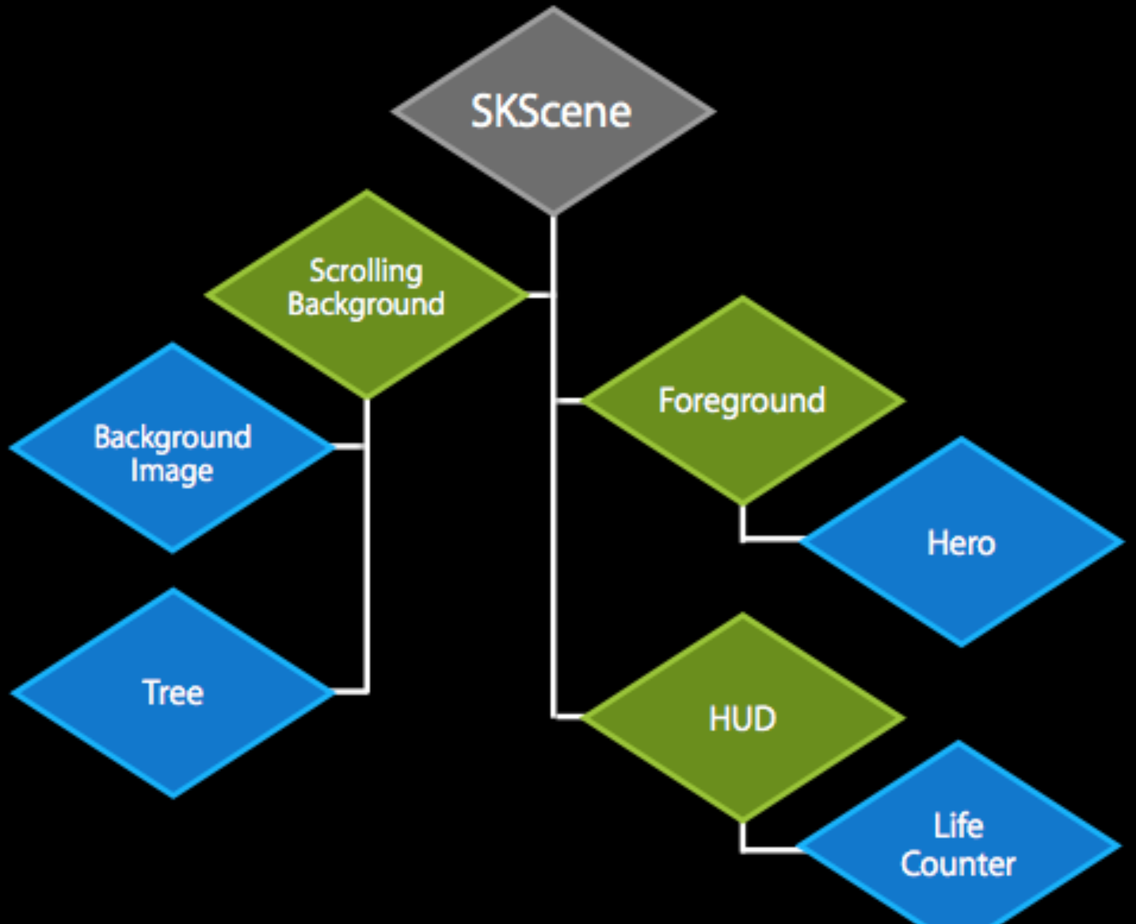
SPRITE KIT IN A NUTSHELL

- 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

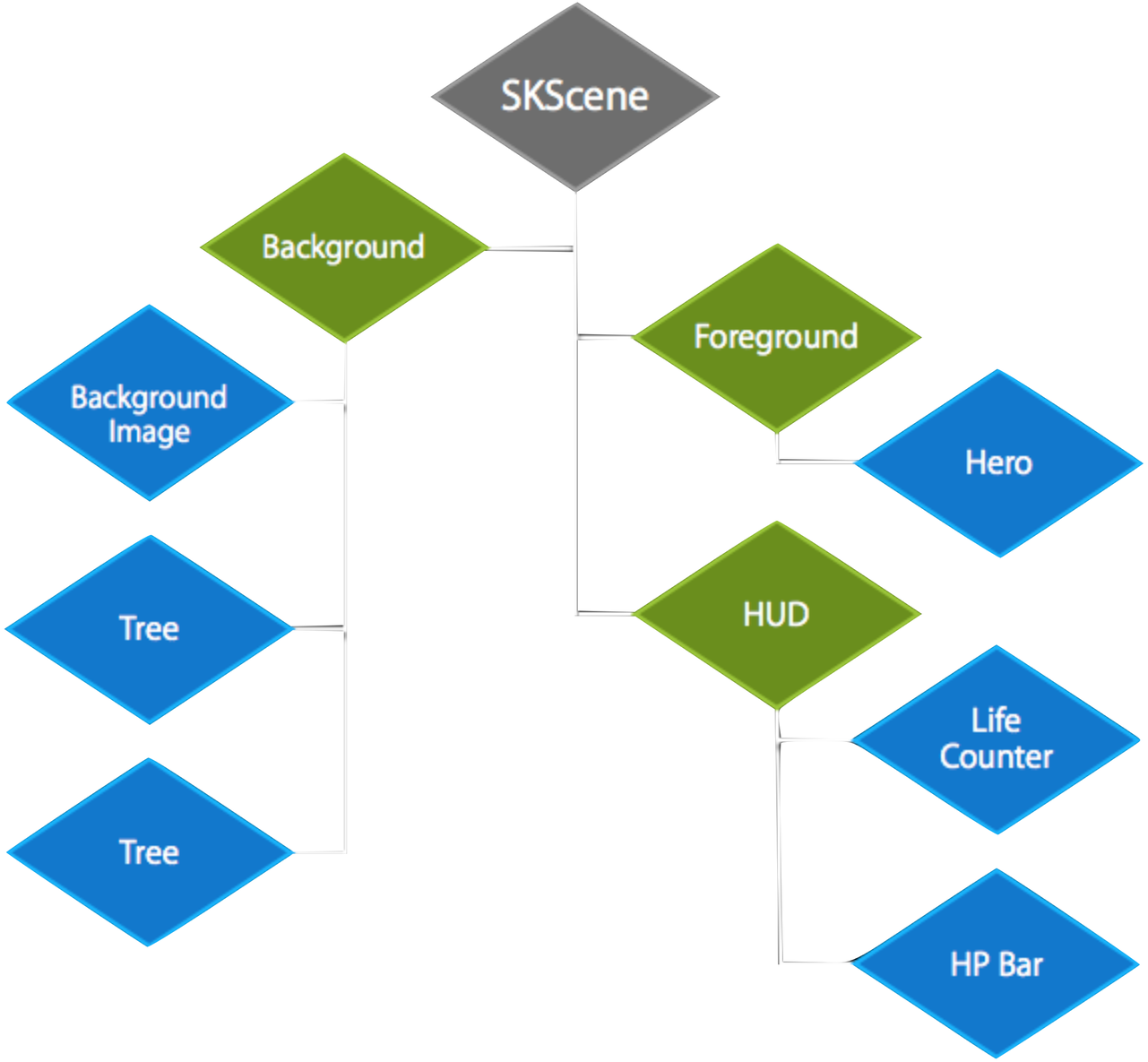
VIEW AND SCENES





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.

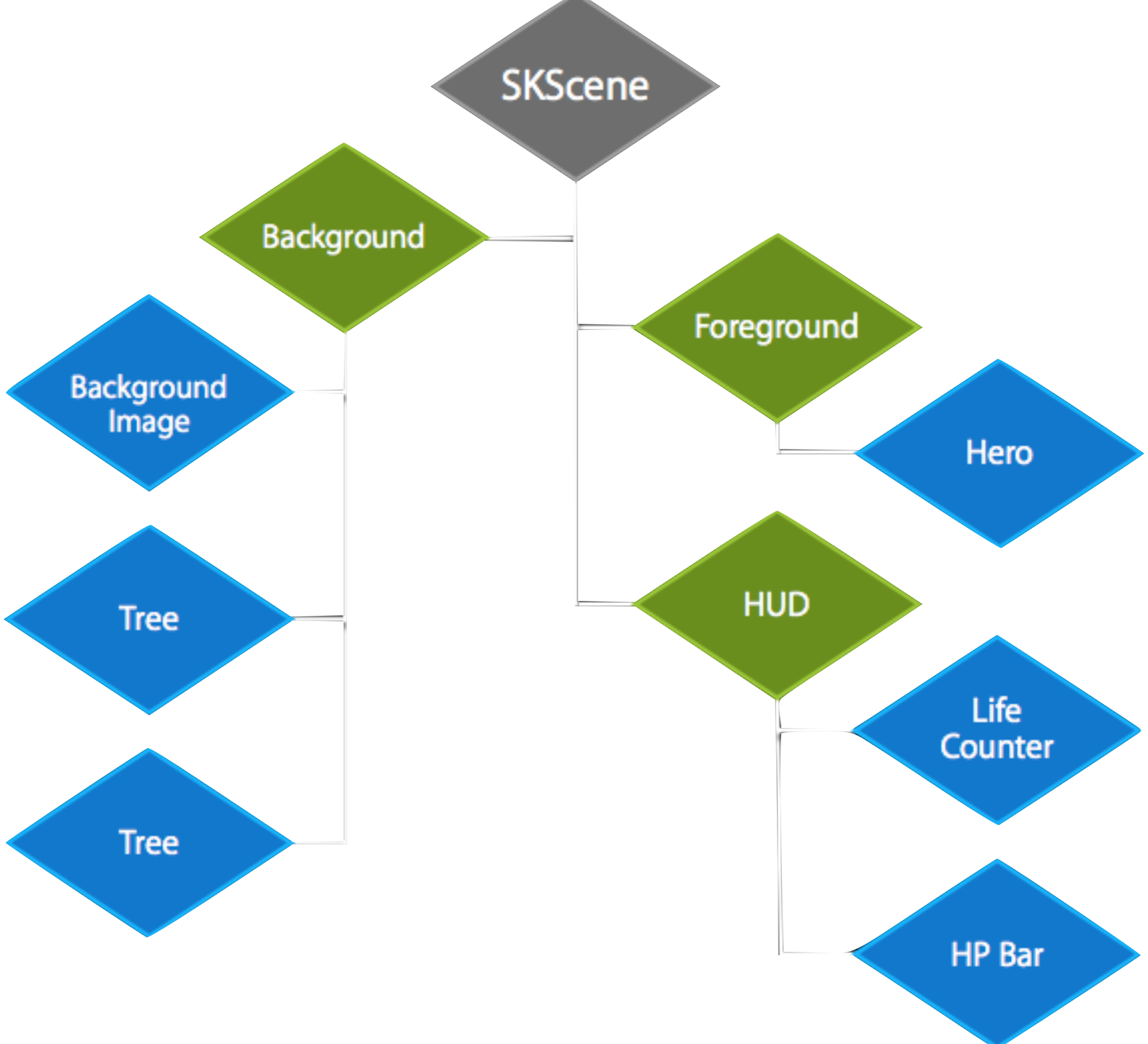
VIEW AND SCENES

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

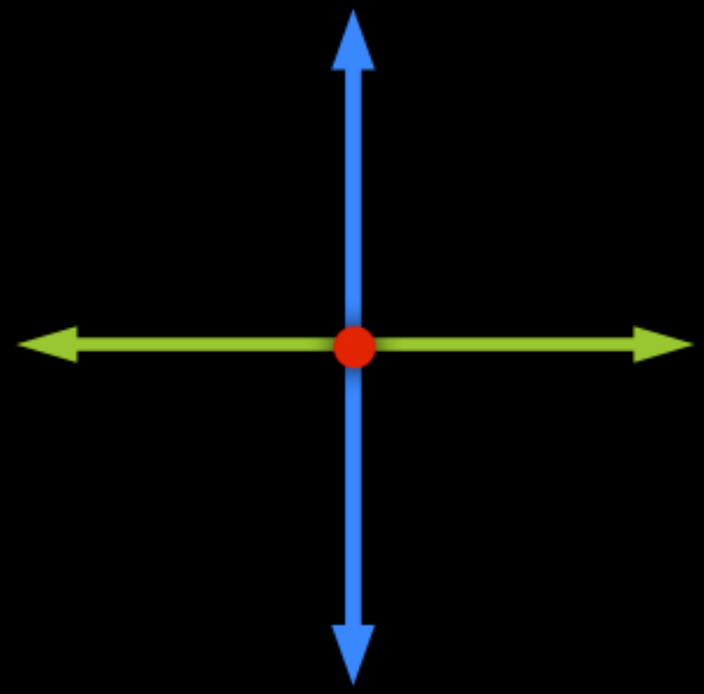


Image via Apple

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

- NSObject
  - UIResponder
    - SKNode
      - SK3DNode
      - SKAudioNode
      - SKCameraNode
      - SKCropNode

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
`SKSpriteNode`	A node that draws a textured sprite.
`SKVideoNode`	A node that plays video content.
`SKLabelNode`	A node that renders a text string.
`SKShapeNode`	A node that renders a shape based on a Core Graphics path.
`SKEmitterNode`	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”

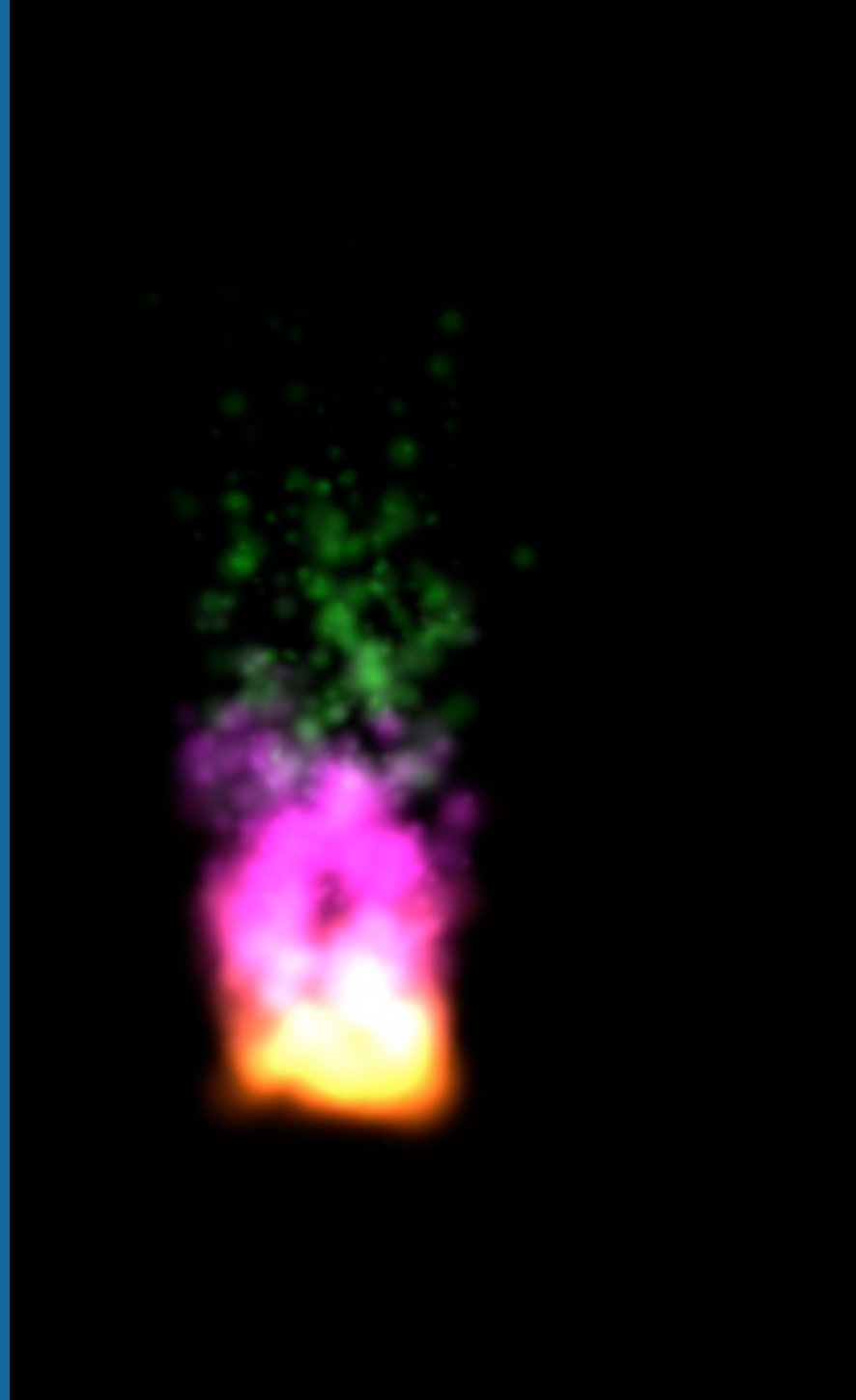
NODES

```
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

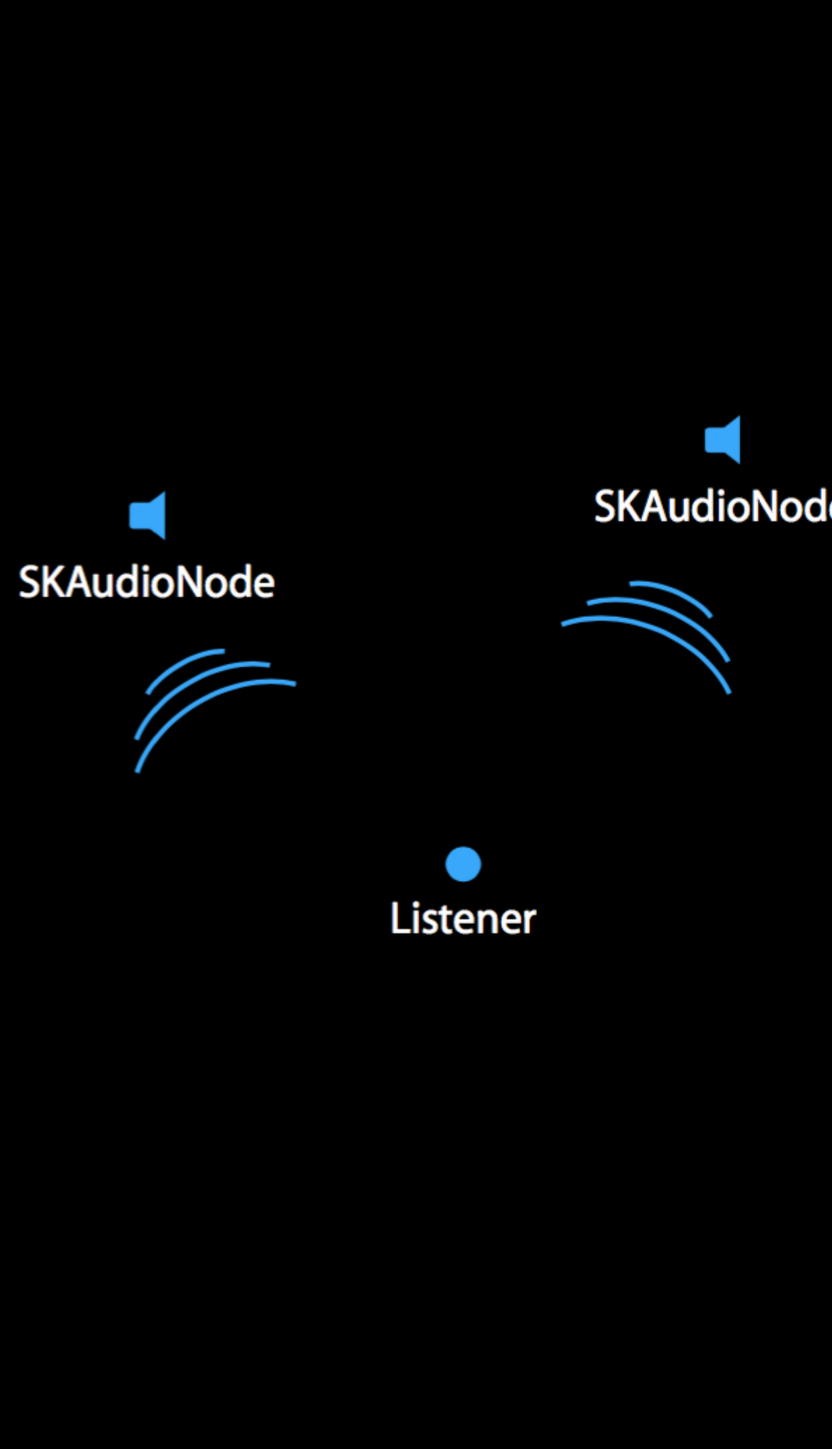
- SKEffectNode
  - Allows Core Image filter effects to be applied to child nodes.
  - A sepia filter effect, for example, could be applied to all child nodes of an SKEffectNode.
- SKCropNode
  - Allows the pixels in a node to be cropped subject to a specified mask

NODES

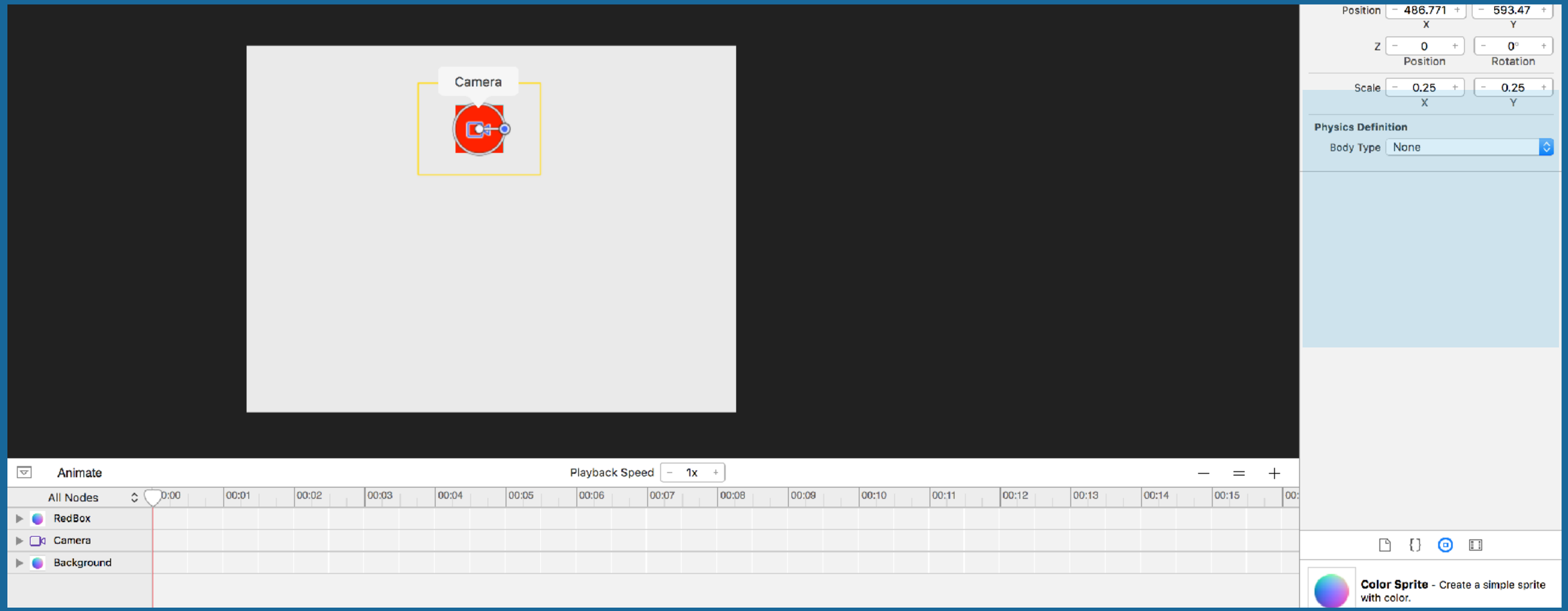
- SKLightNode
  - Adds a light sources to a scene
    - Casts of shadows when the light falls on other nodes in the same scene
- SK3DNode
  - Allows 3D assets to be embedded into 2D Sprite Kit games

NODES

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



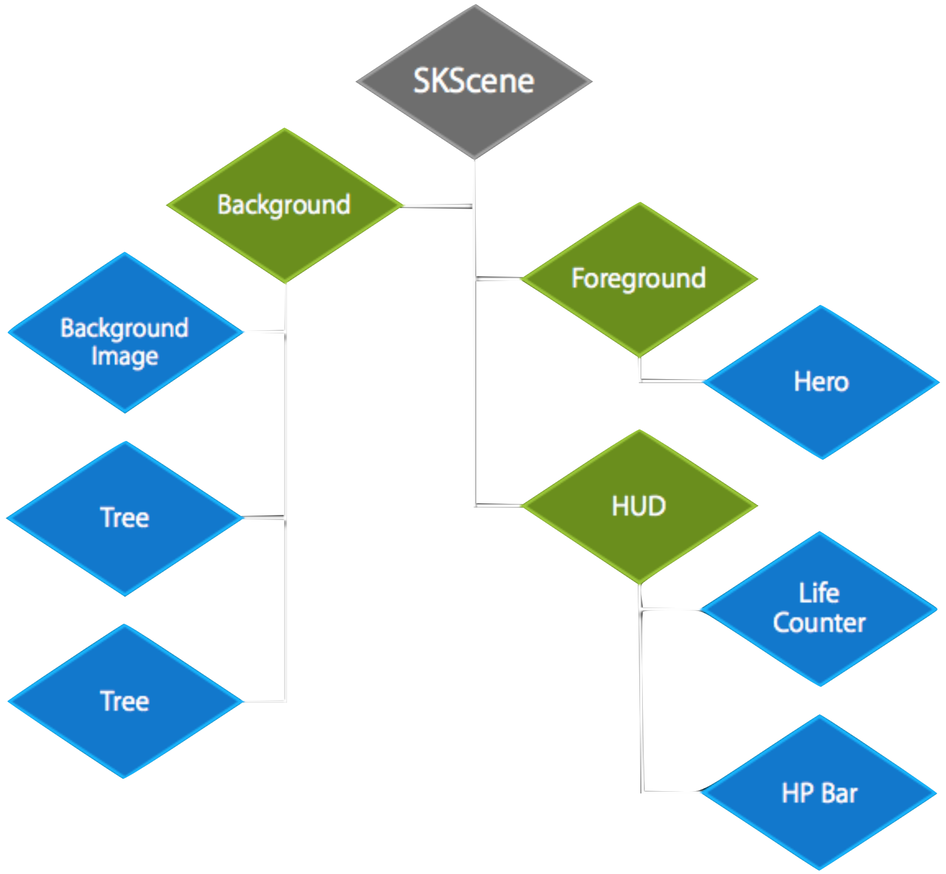
NODES



- 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

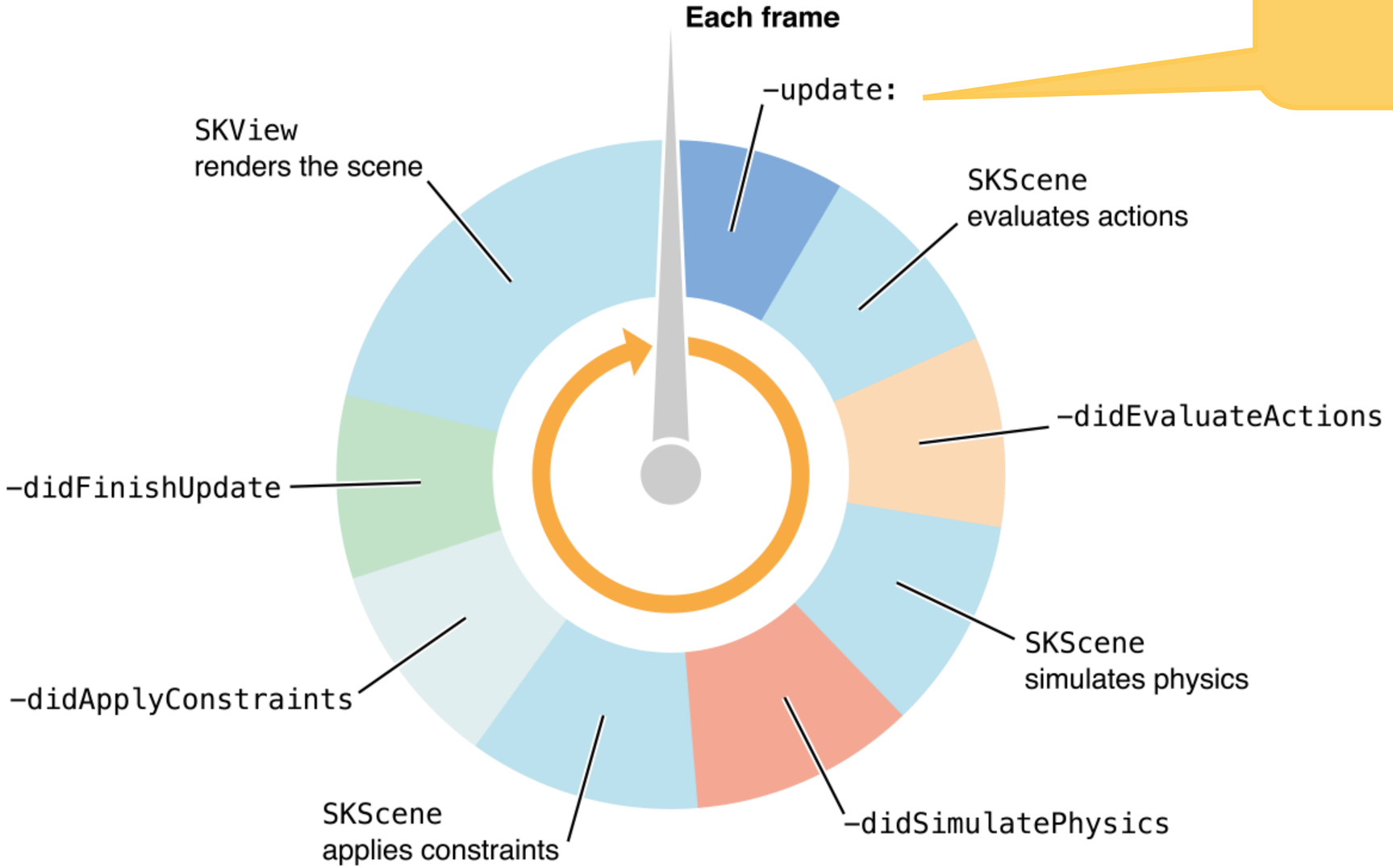
```
import SpriteKit
import GameplayKit

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)))
```

Label Node

GAME RENDERING LOOP

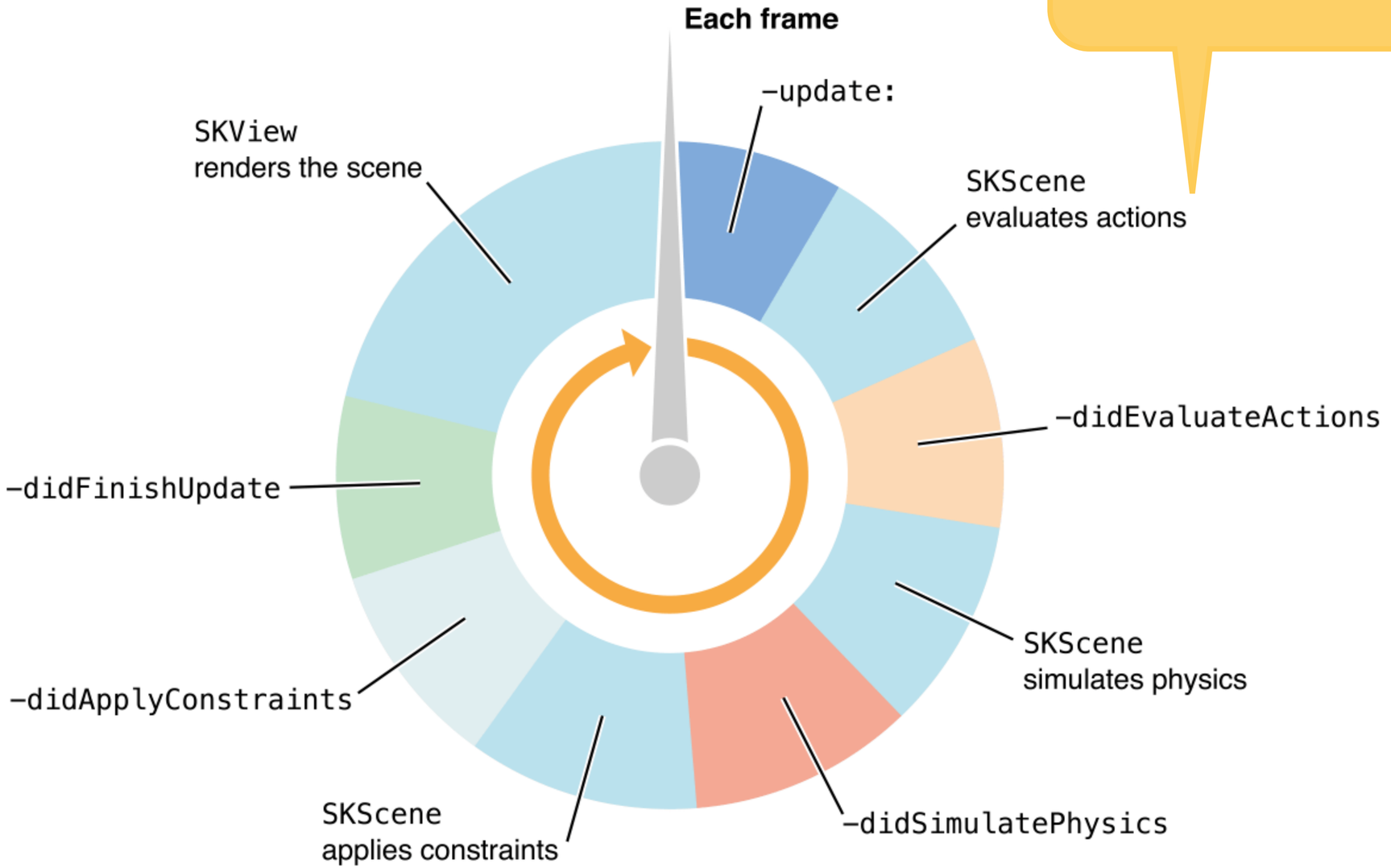
GAME RENDERING LOOP

Your game update (spawn enemies, update score)

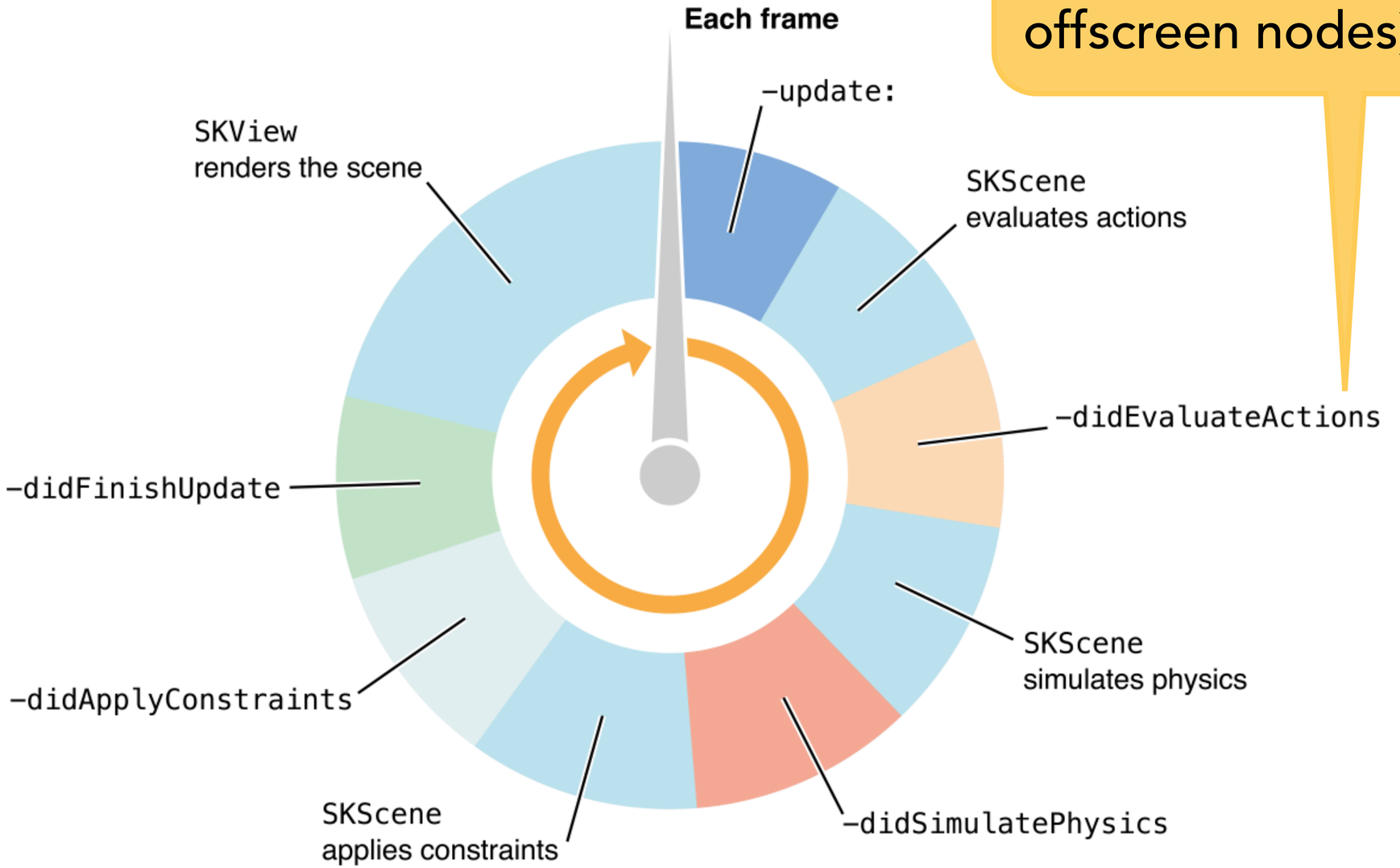


GAME RENDERING LOOP

Actions are evaluated (animations)

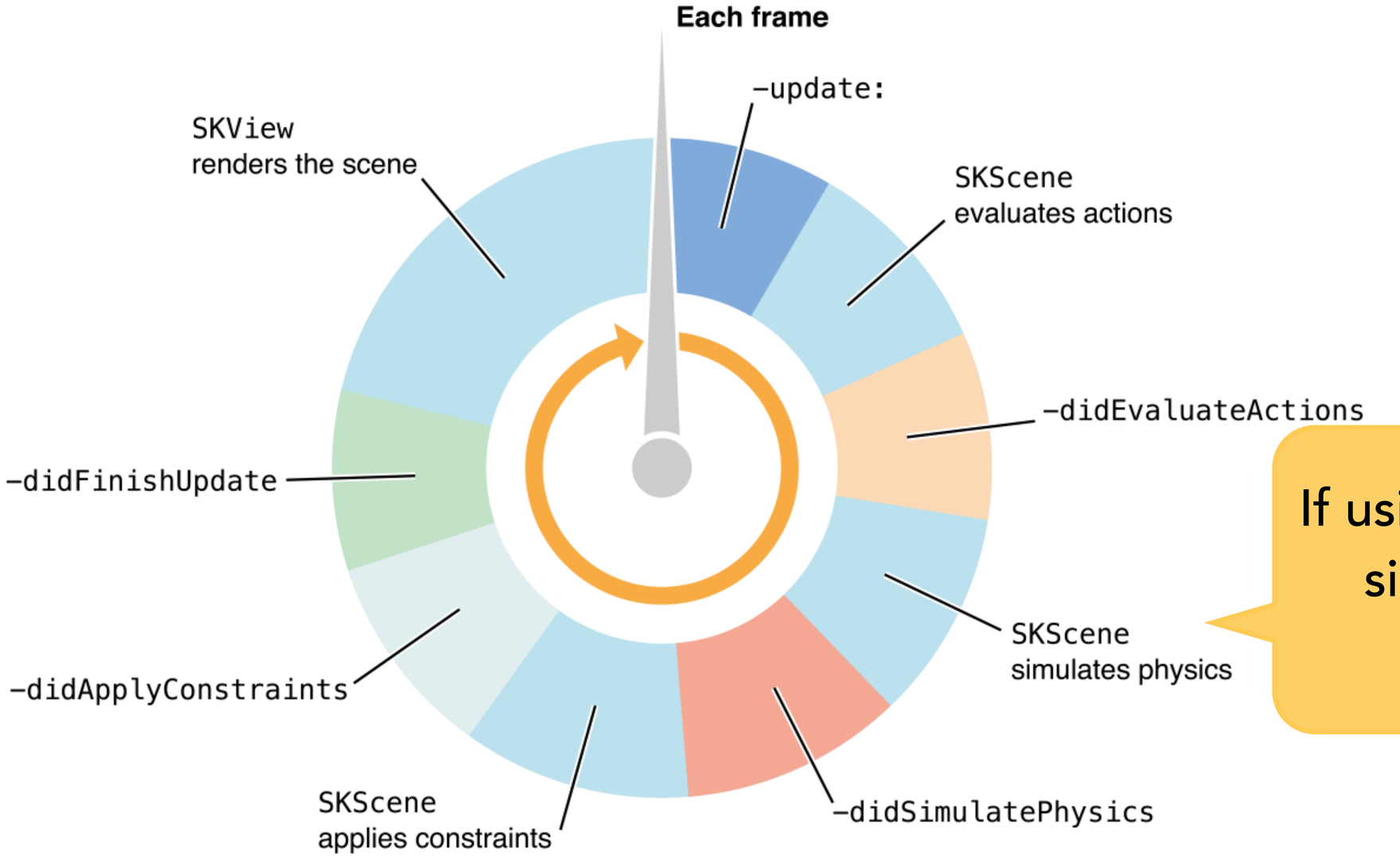


GAME RENDERING LOOP



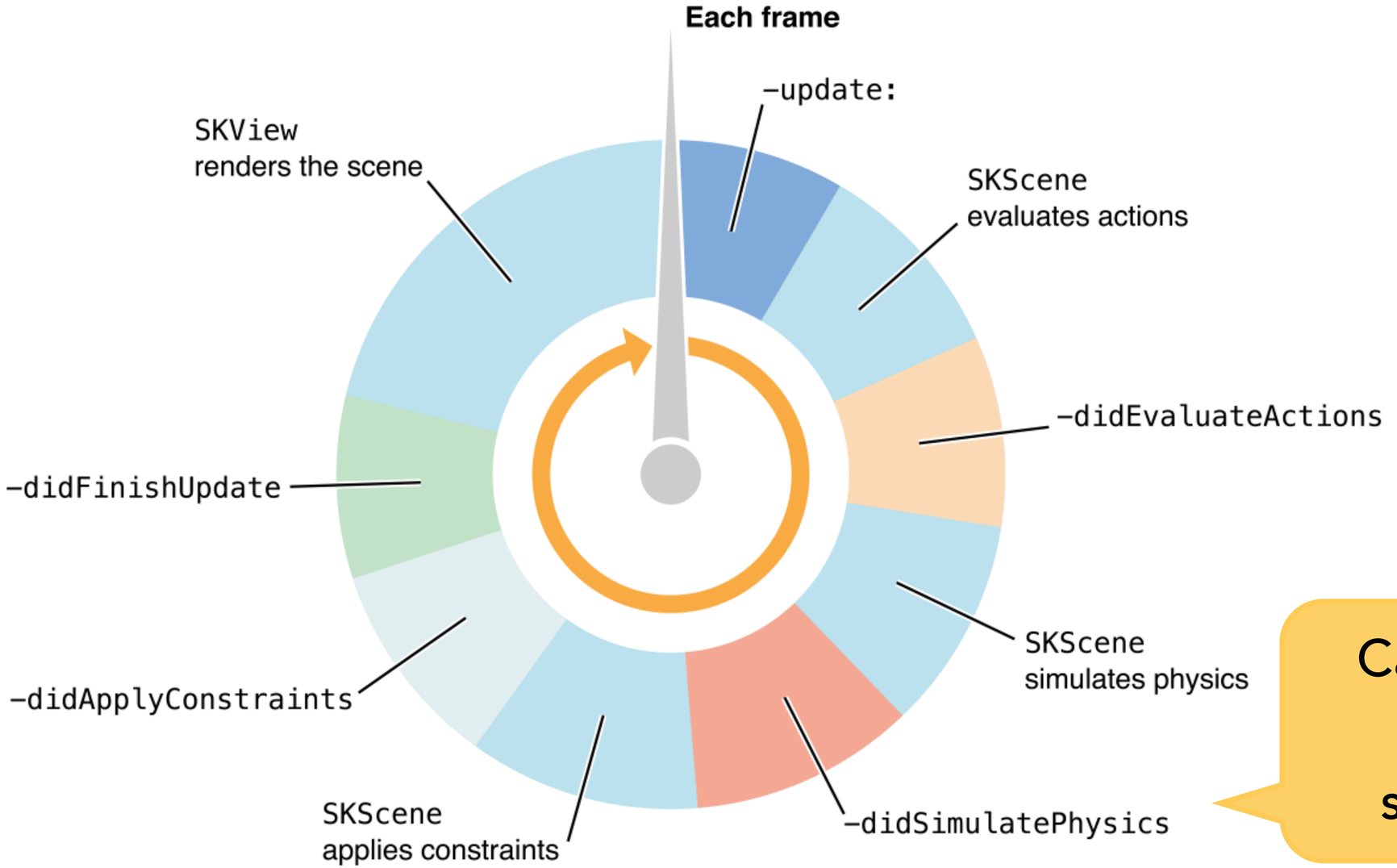
Called after actions (remove offscreen nodes)

GAME RENDERING LOOP



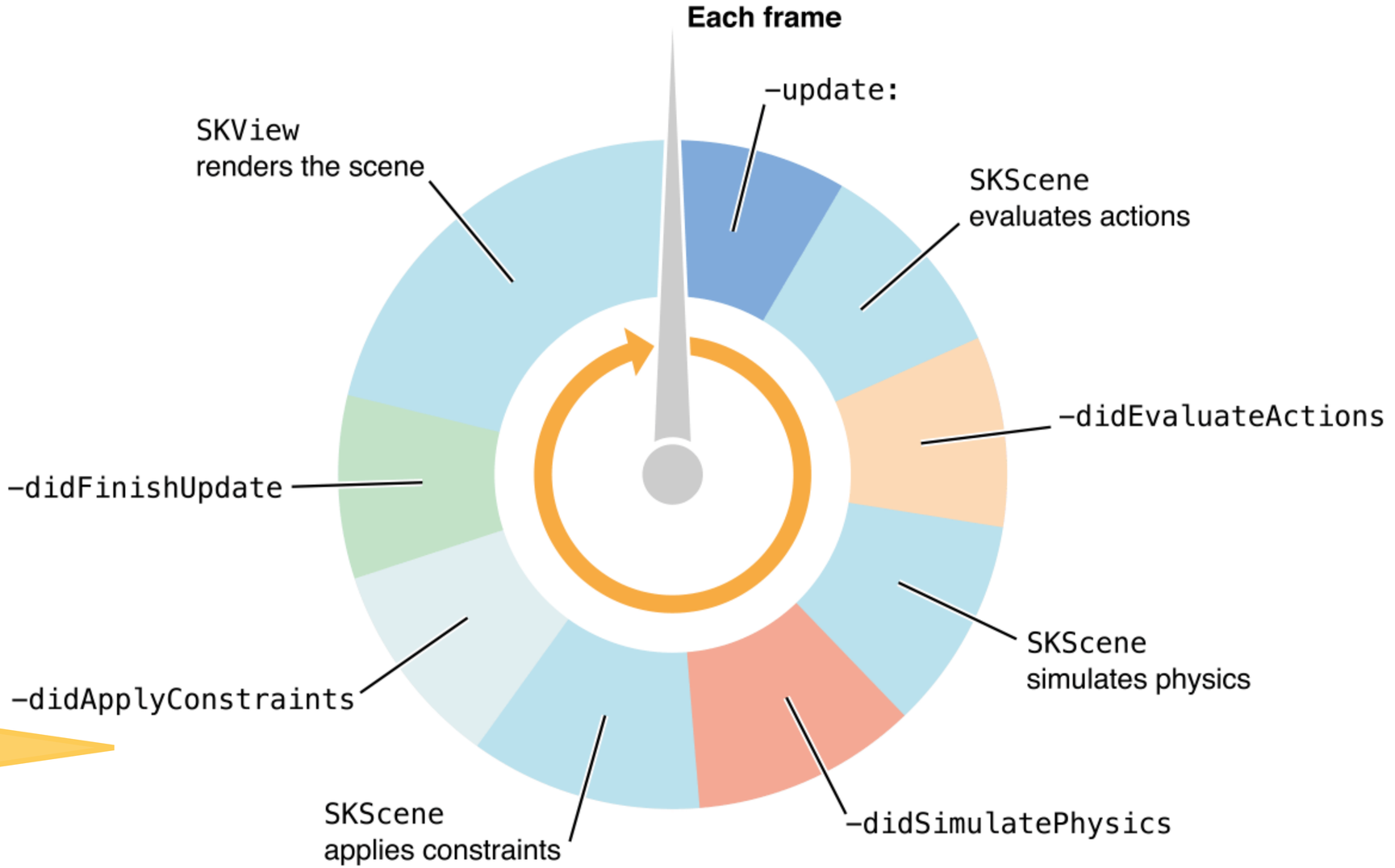
If using physics, simulation occurs

GAME RENDERING LOOP



Called after physics simulated

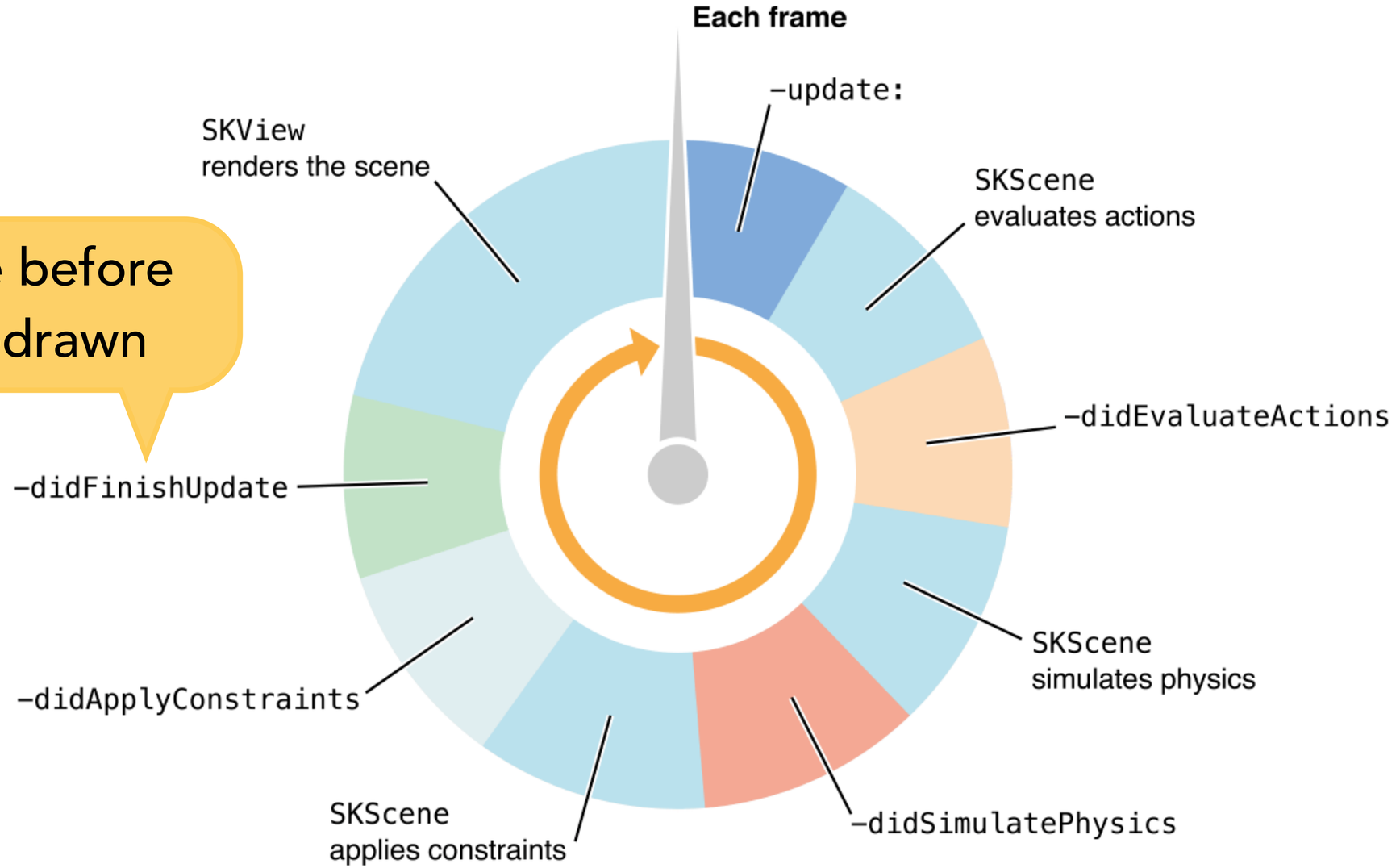
GAME RENDERING LOOP



Constraints applied

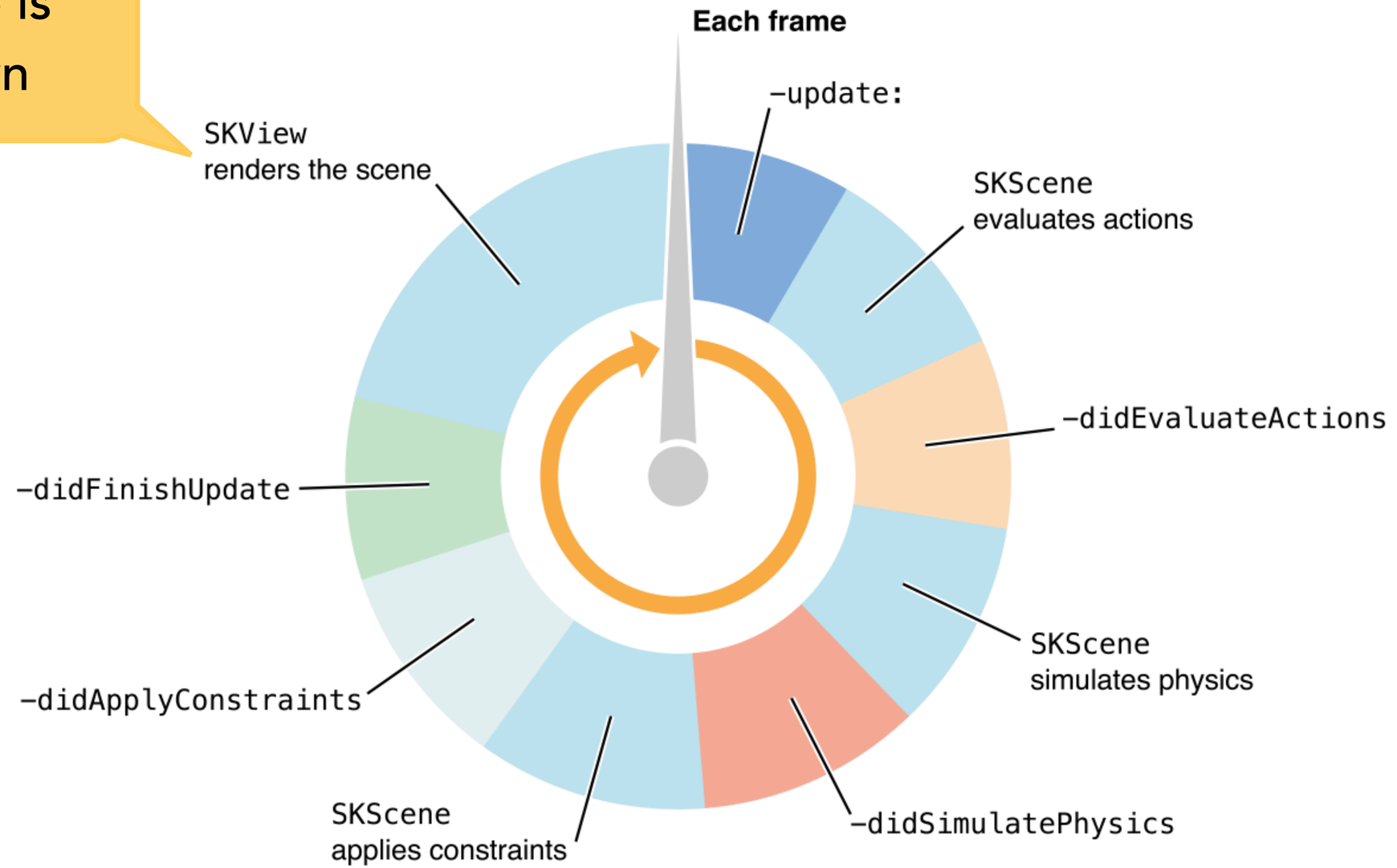
GAME RENDERING LOOP

Last chance before scene is redrawn



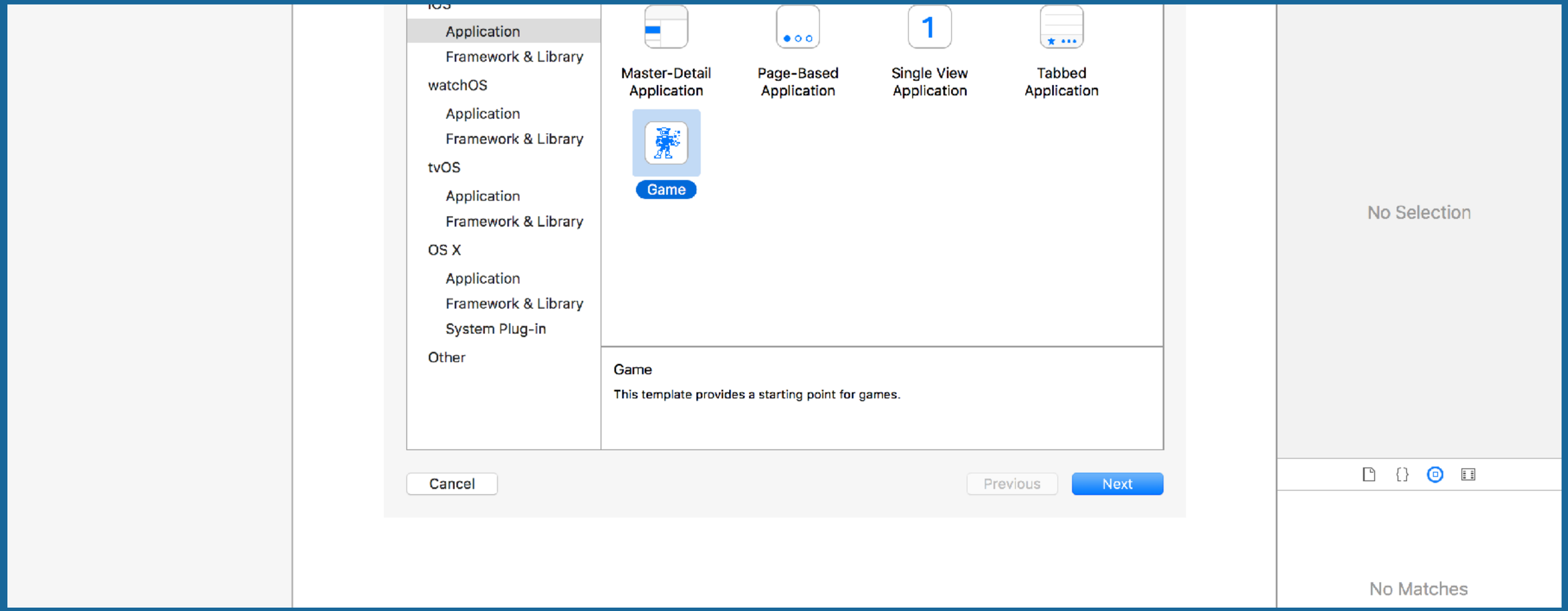
GAME RENDERING LOOP

Scene is drawn



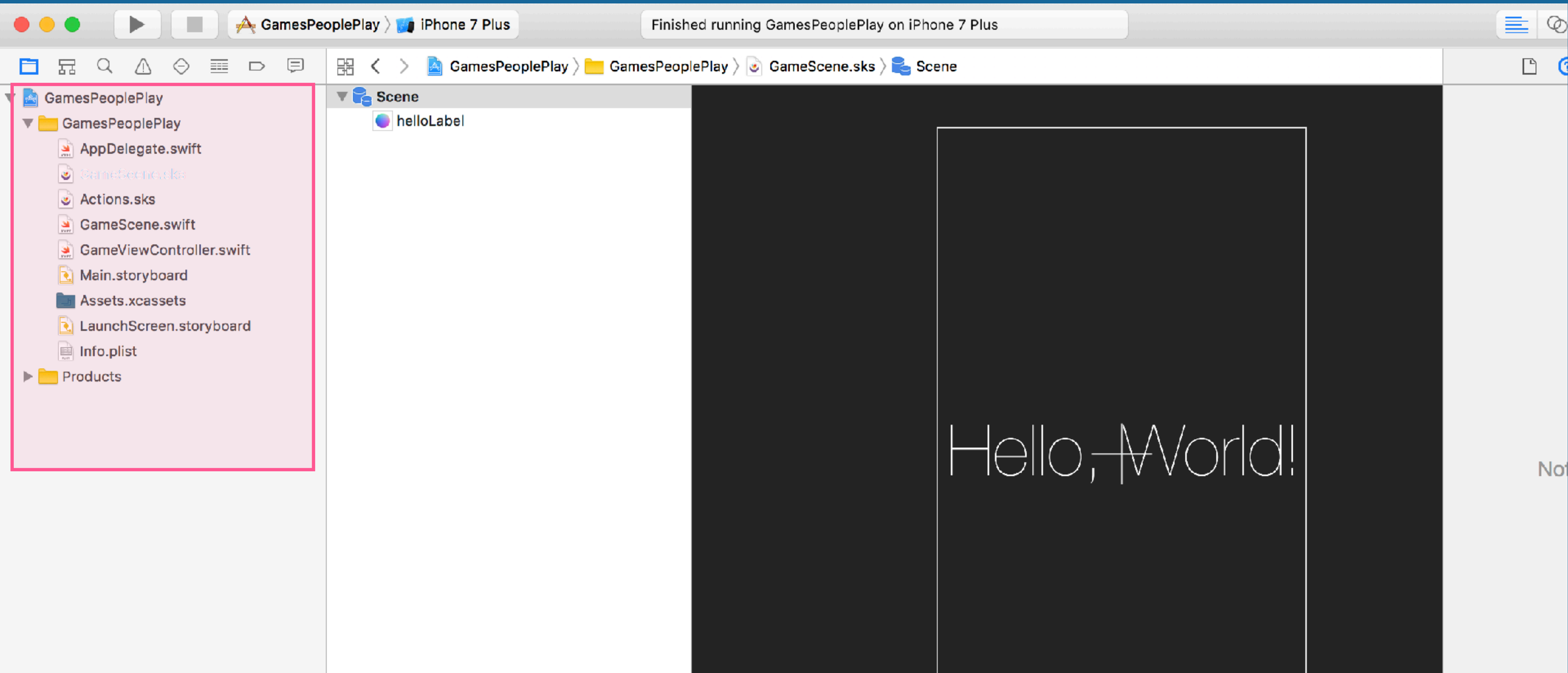
CREATING A SPRITE KIT PROJECT

CREATING A SPRITE KIT PROJECT

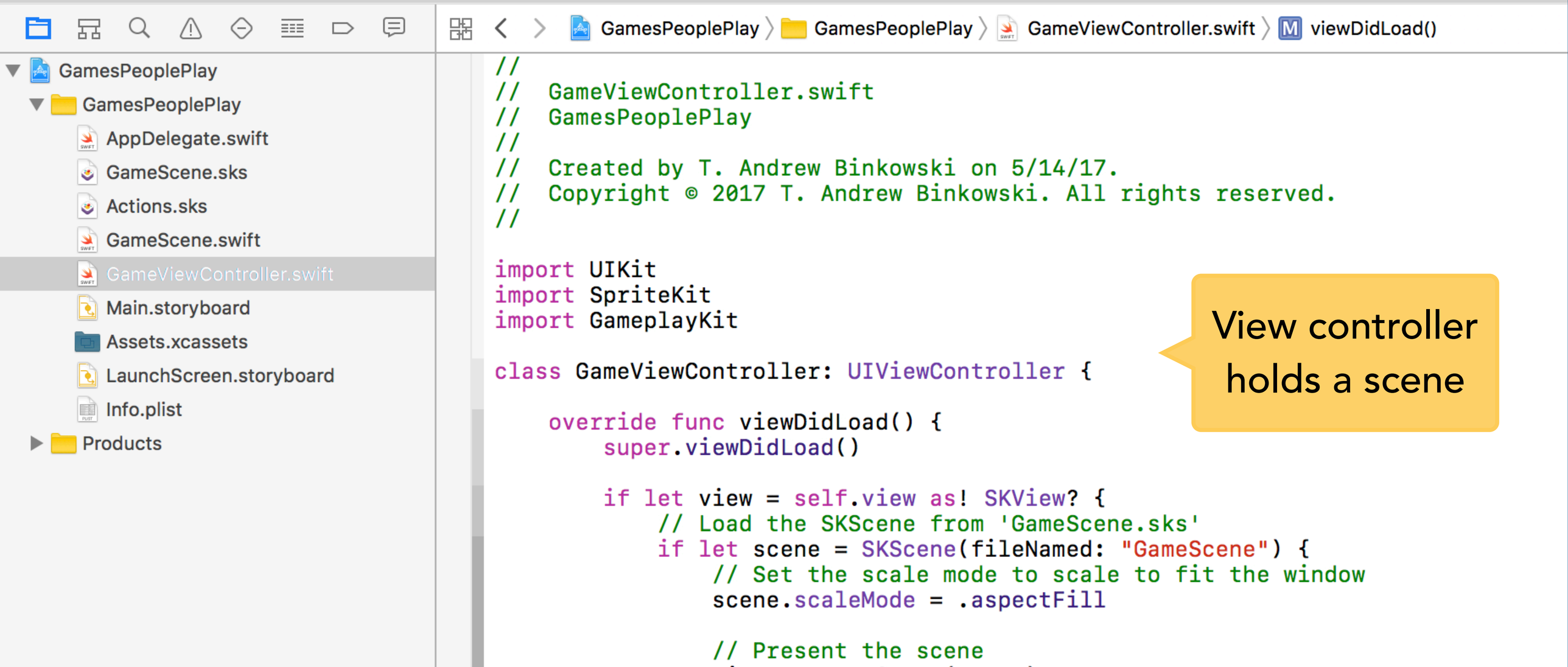


- Game template in Xcode

CREATING A SPRITE KIT PROJECT



CREATING A SPRITE KIT PROJECT



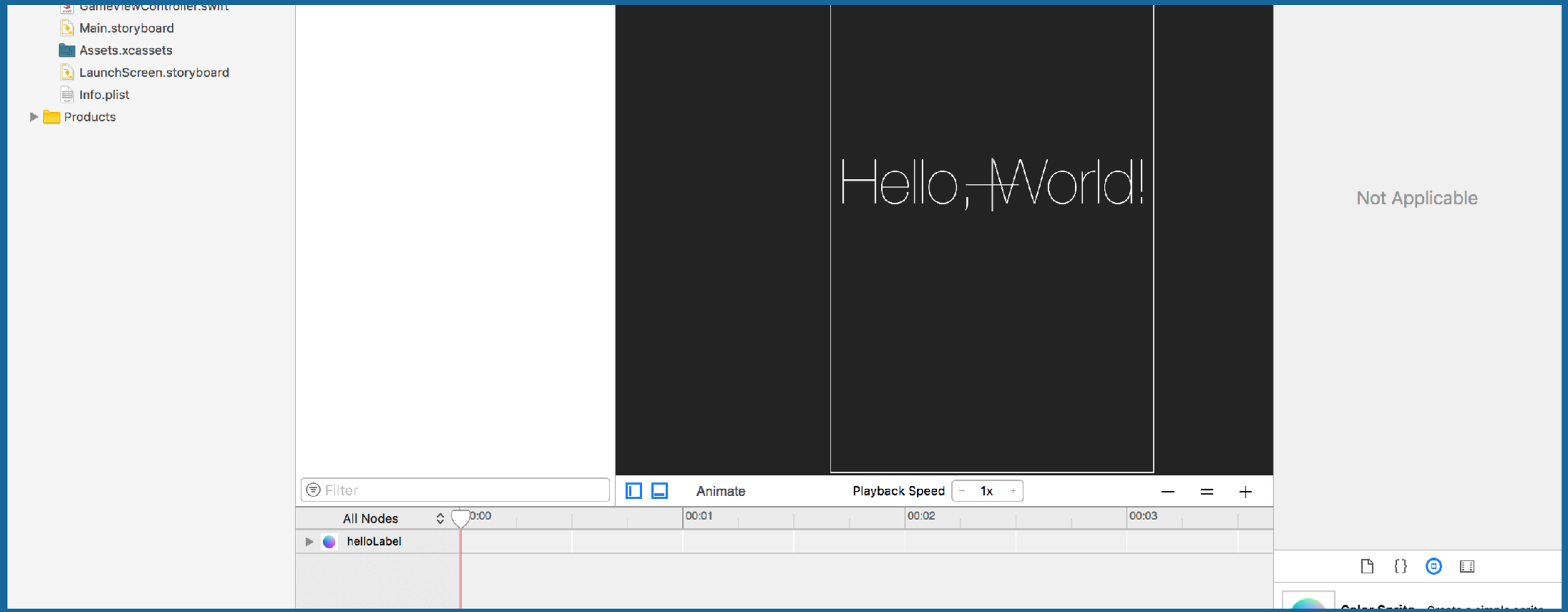
VIEW AND SCENES

```
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

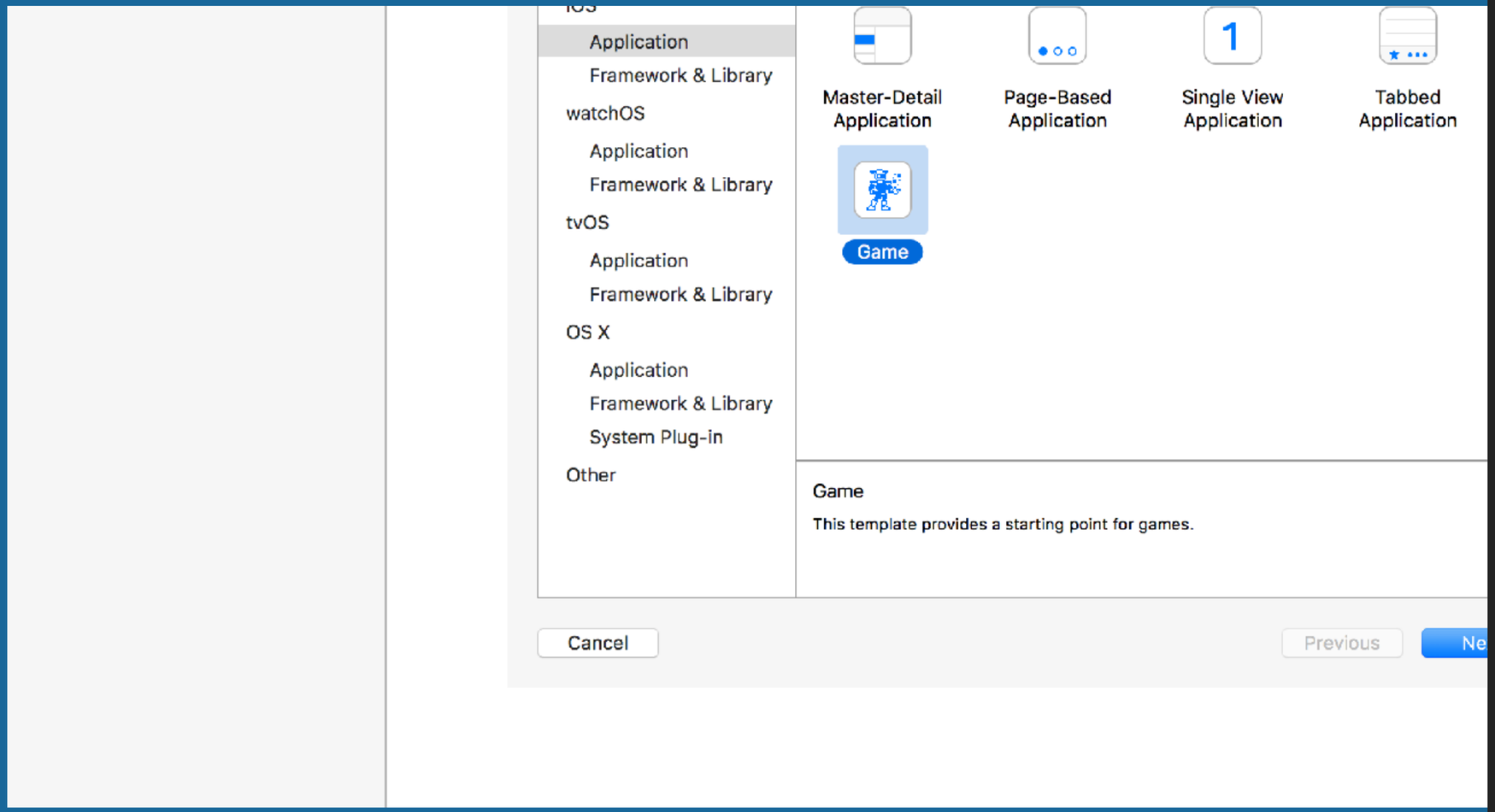
- Default template assigns SKView to view controller's main view

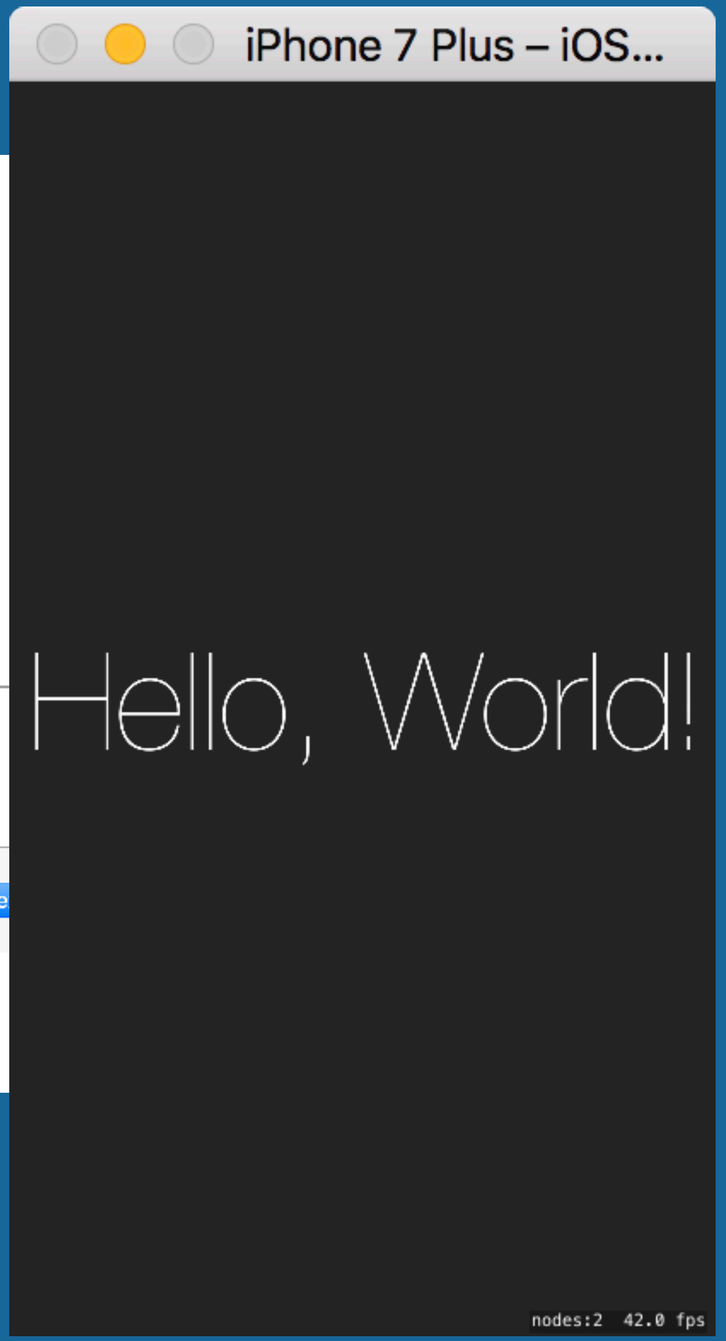
VIEW AND SCENES



- Scene editor

VIEW AND SCENES

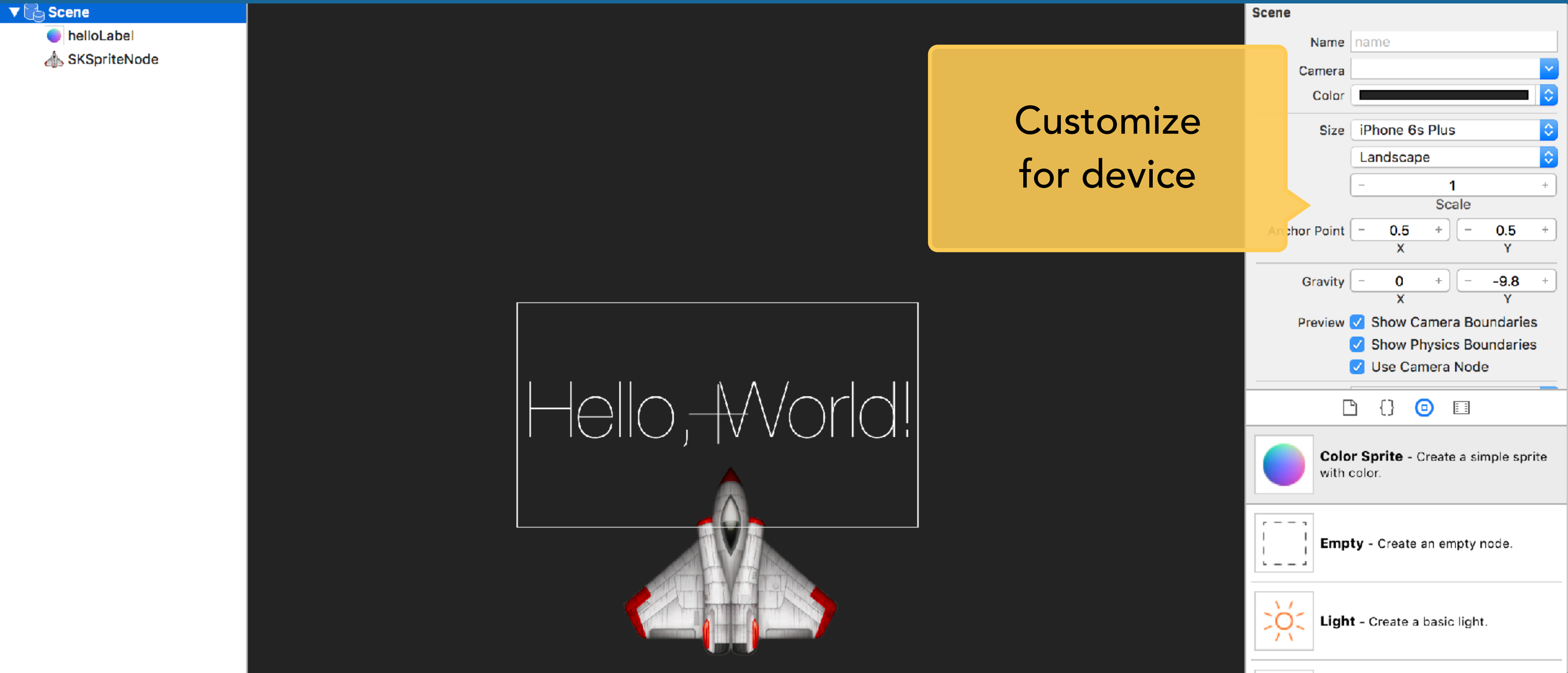




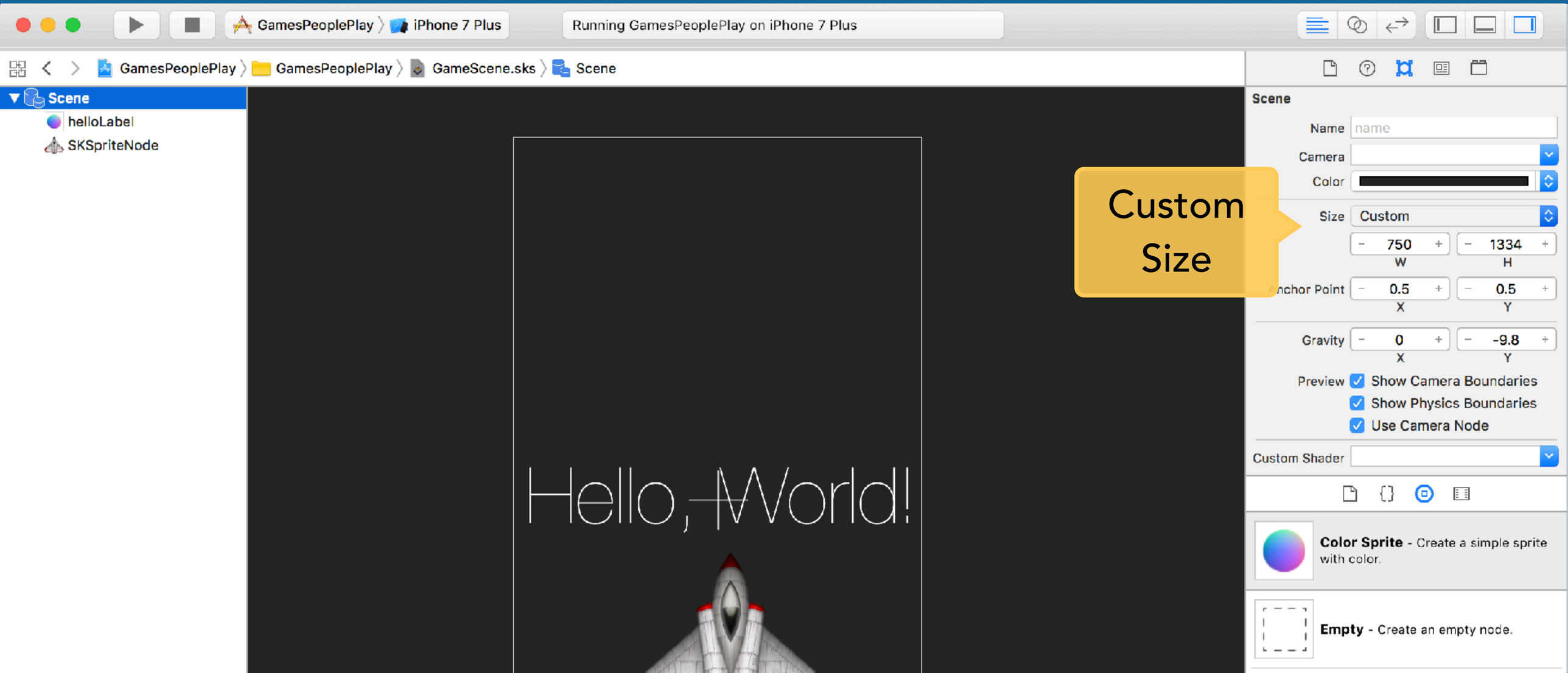
- Game template in Xcode

SCENE EDITOR

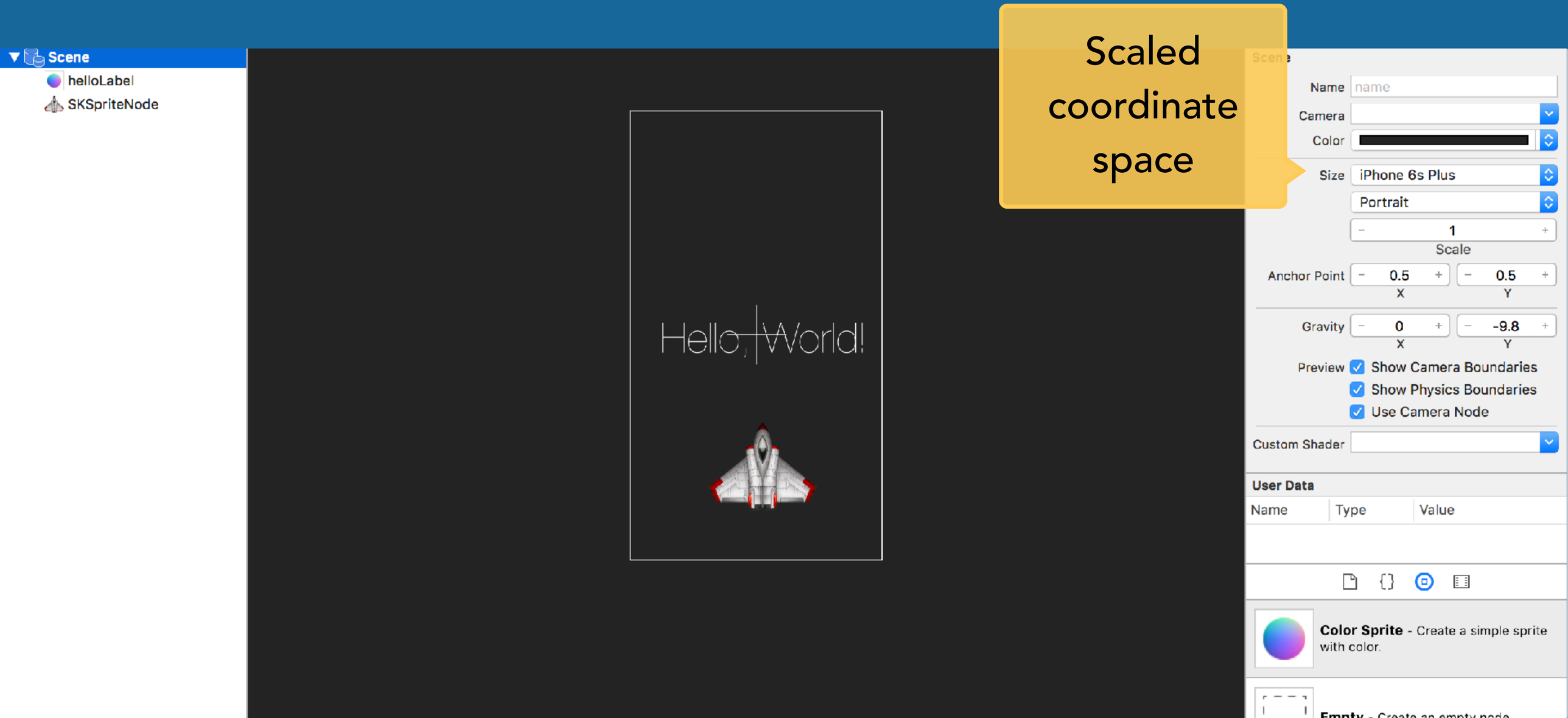
SCENE EDITOR



SCENE EDITOR

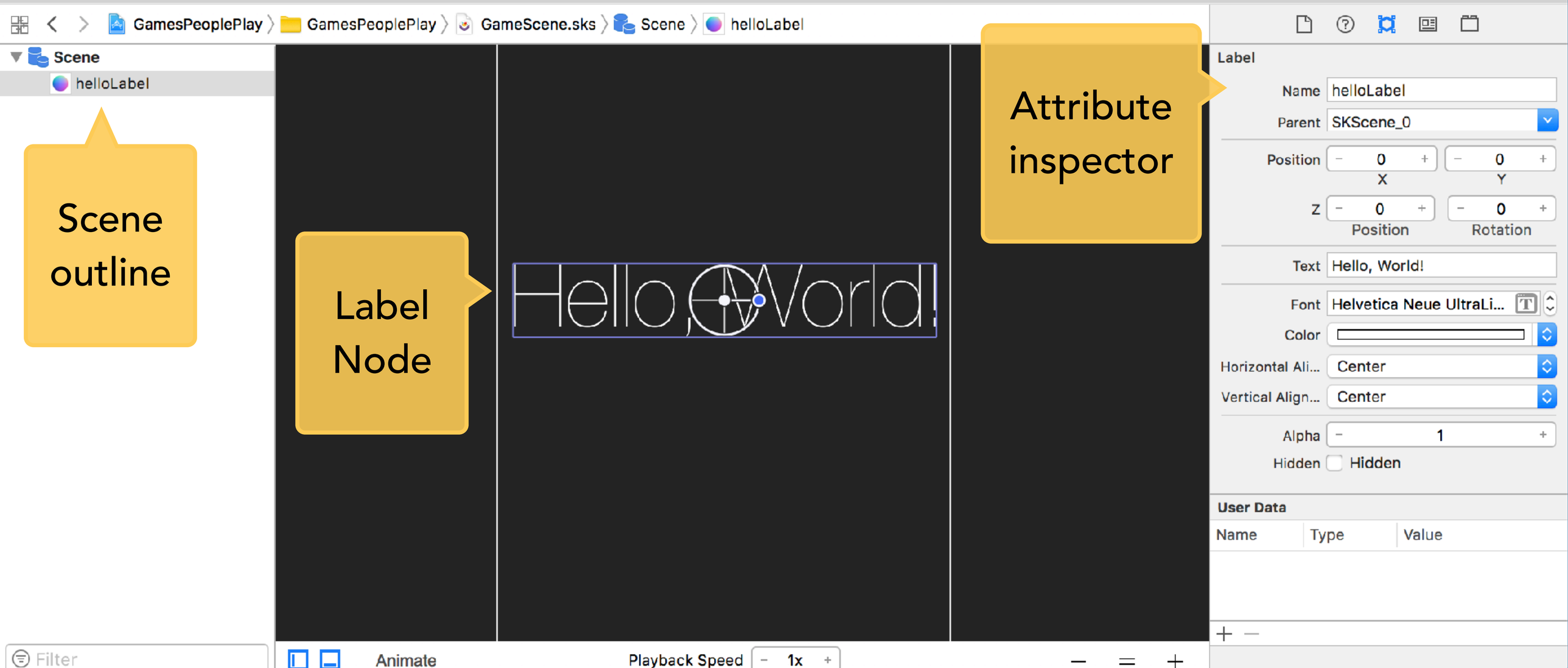


SCENE EDITOR

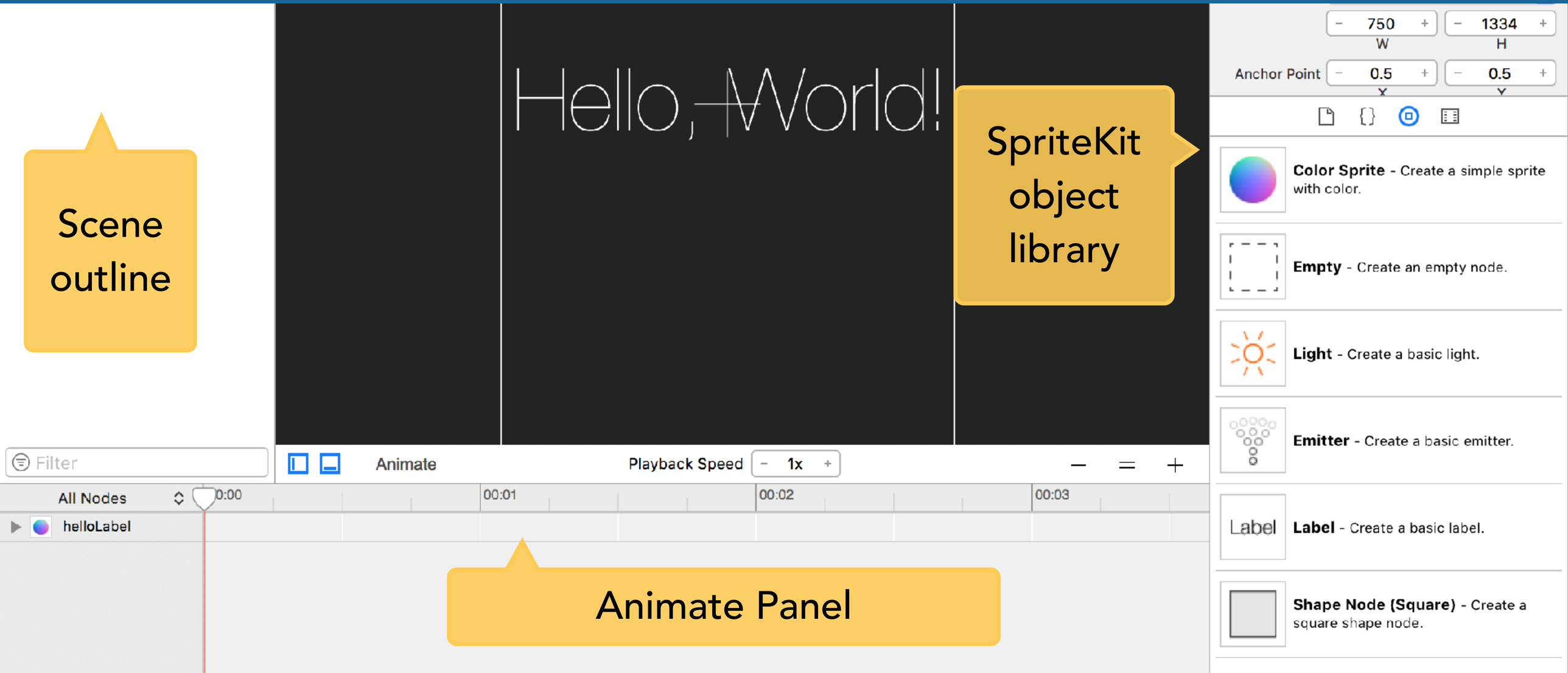


SCENE EDITOR

SCENE EDITOR

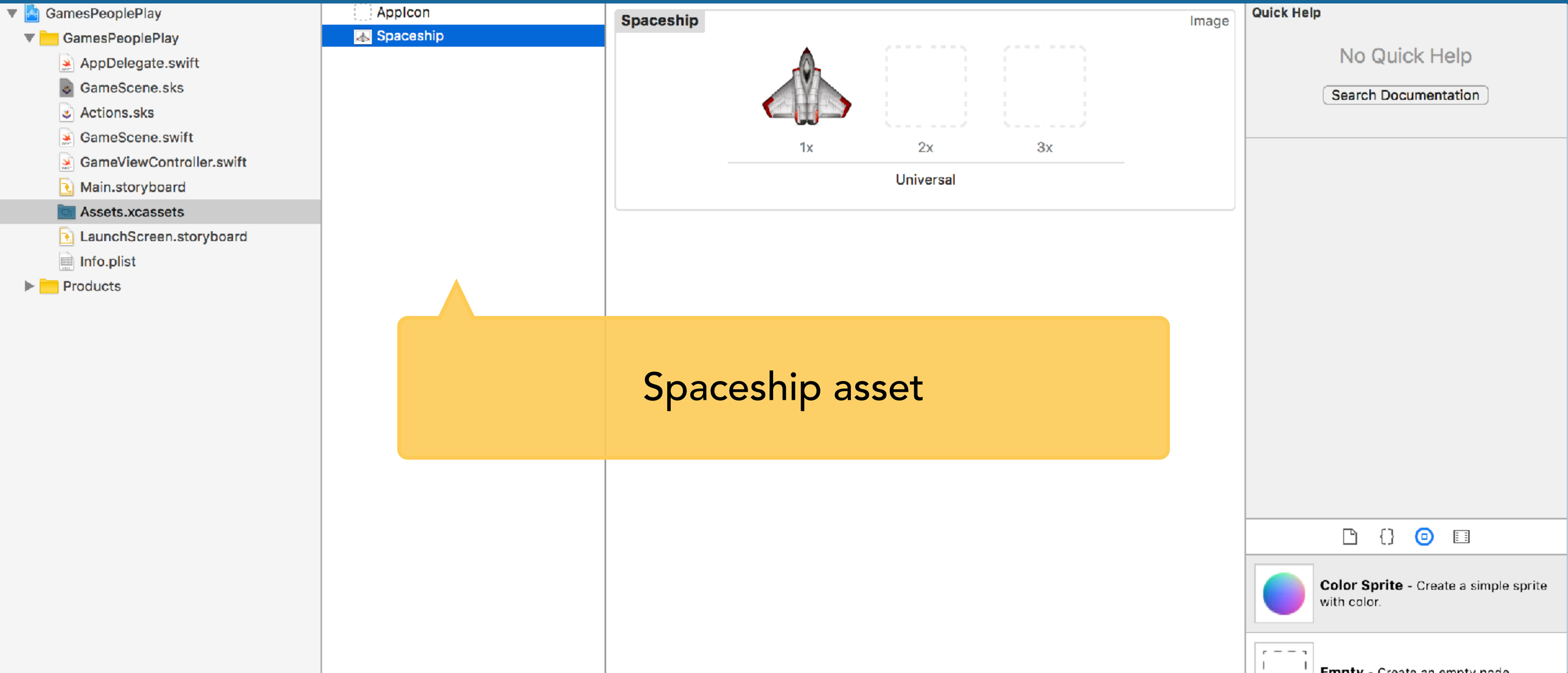


SCENE EDITOR



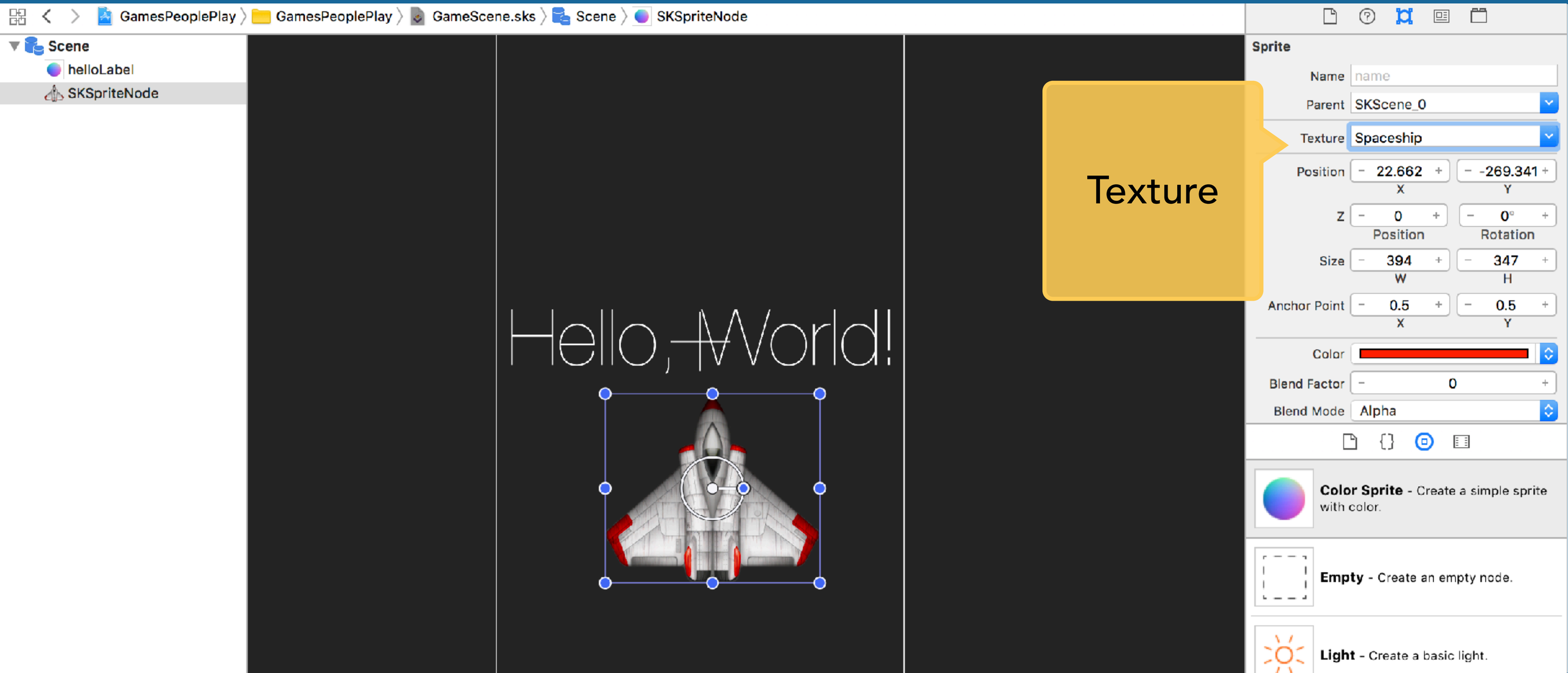
ADDING NODES

SCENE EDITOR

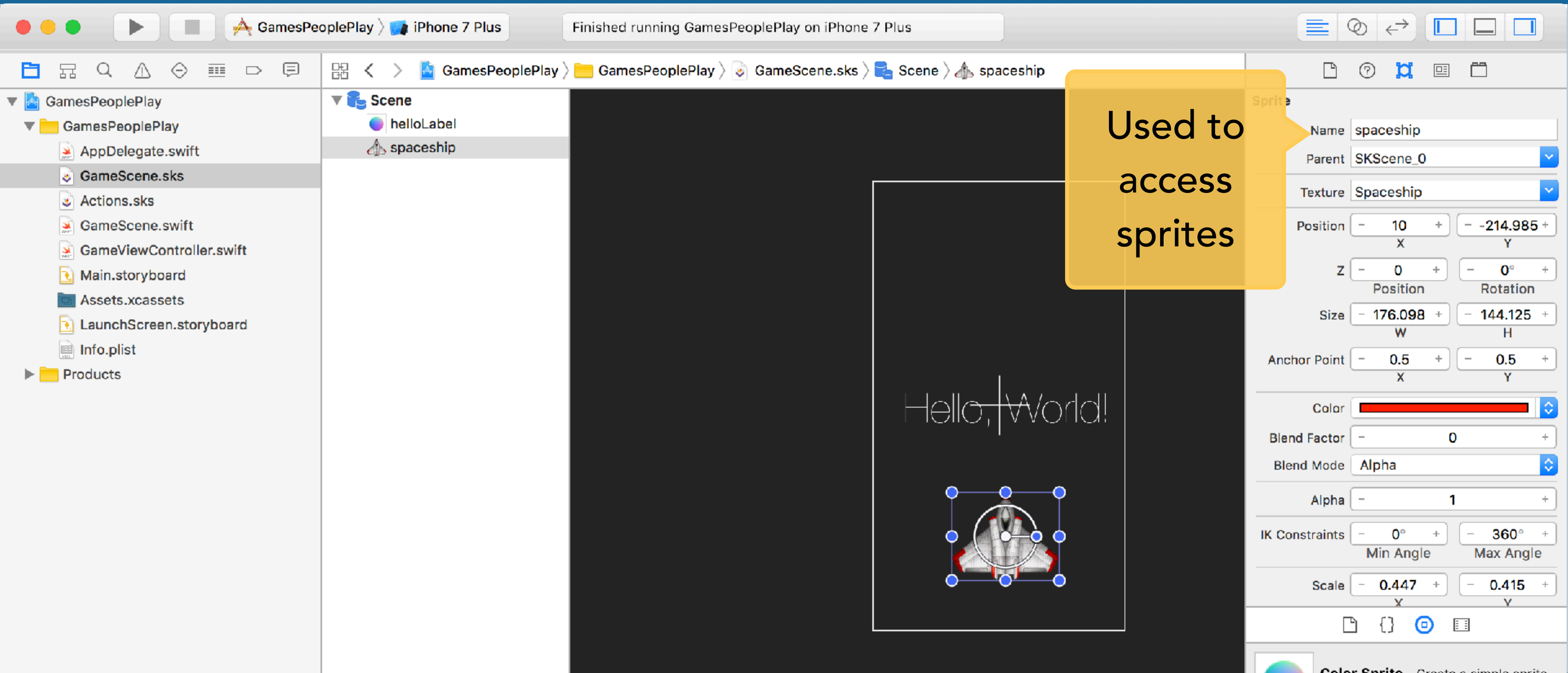


Spaceship asset

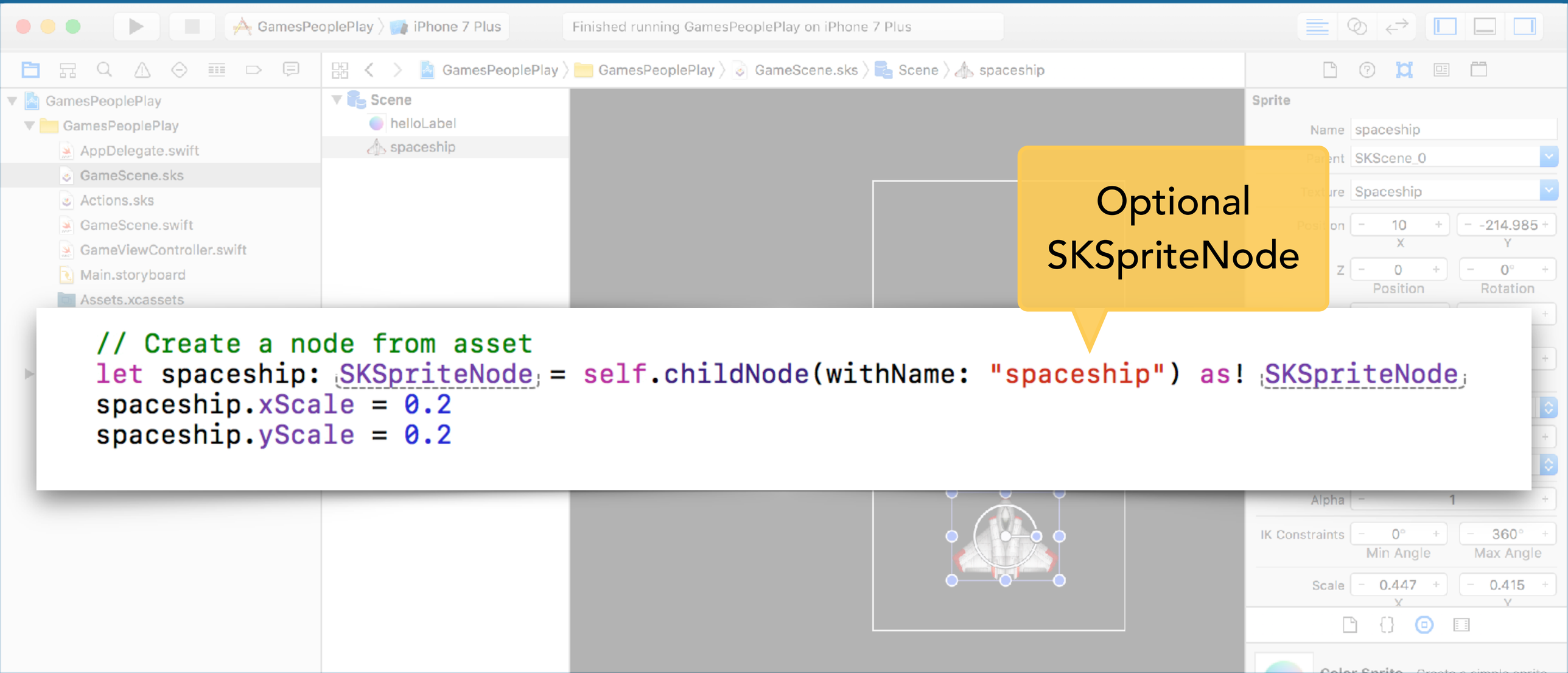
SCENE EDITOR



SCENE EDITOR

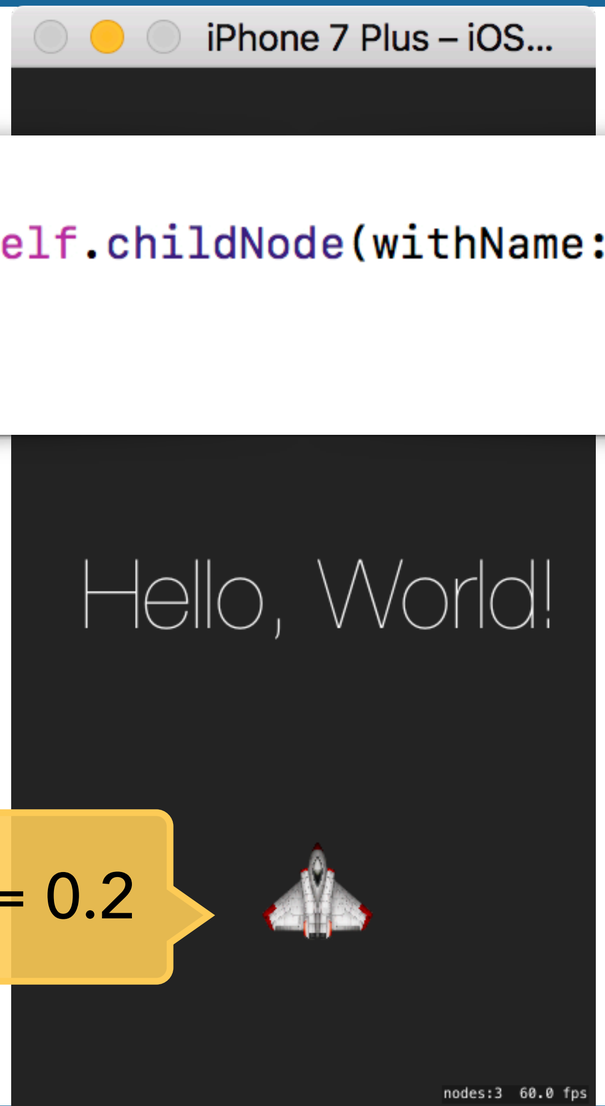


SCENE EDITOR



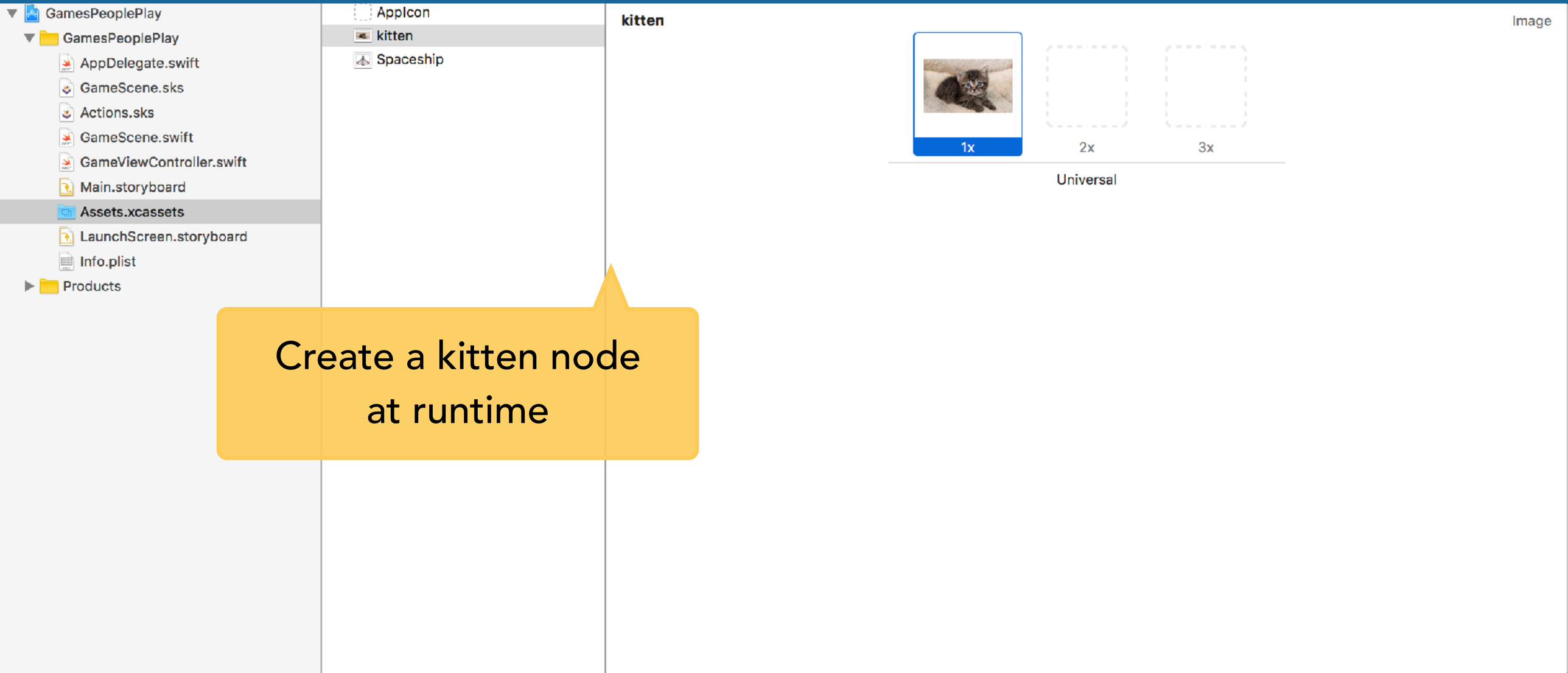
SCENE EDITOR

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



Scale = 0.2

SCENE EDITOR

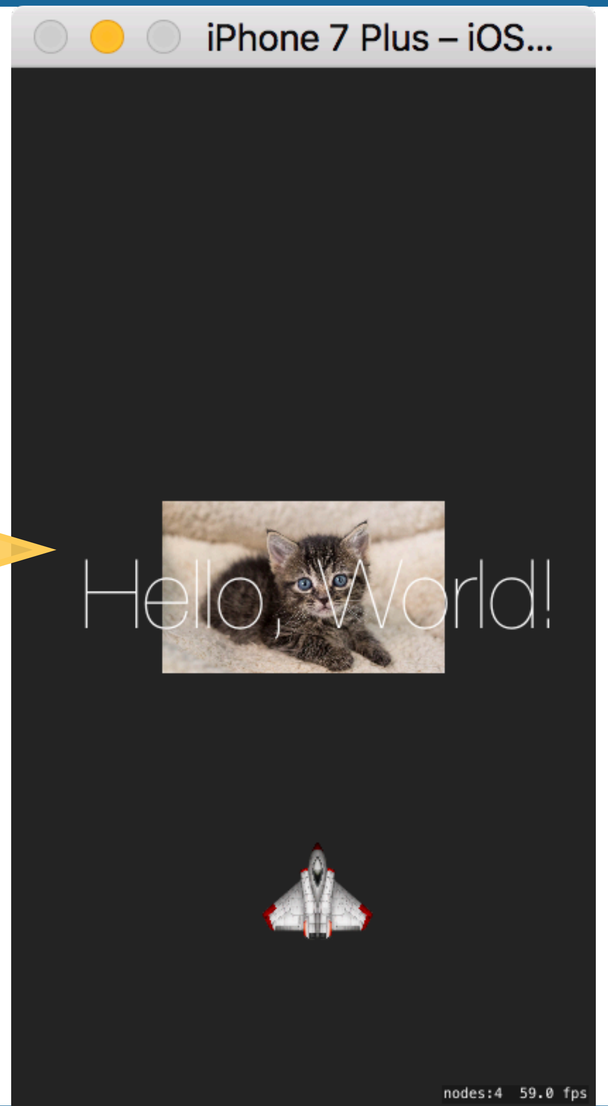


SCENE EDITOR

```
override func didMove(to view: SKView) {

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

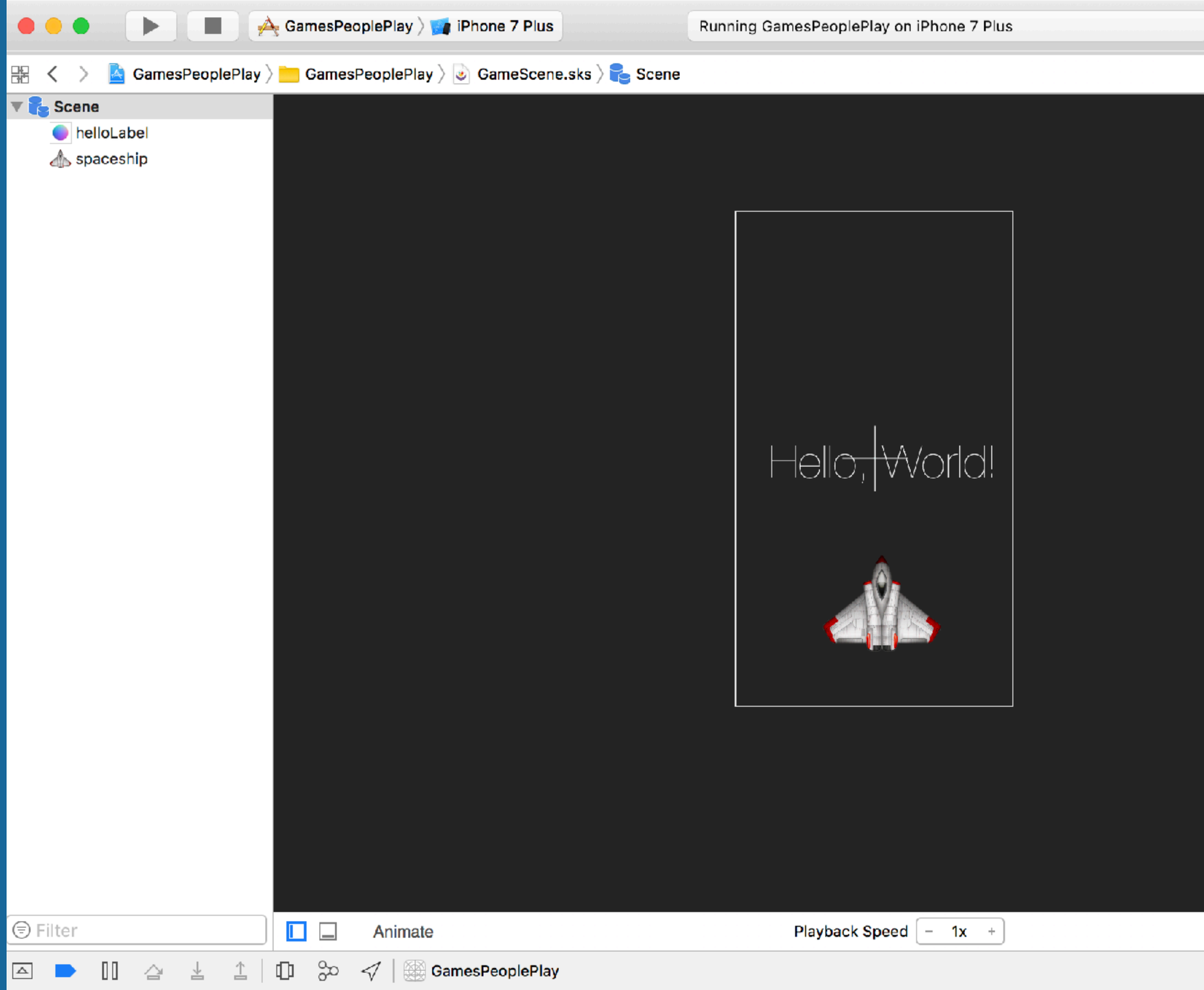
SCENE EDITOR

Add at origin



SCENE EDITOR

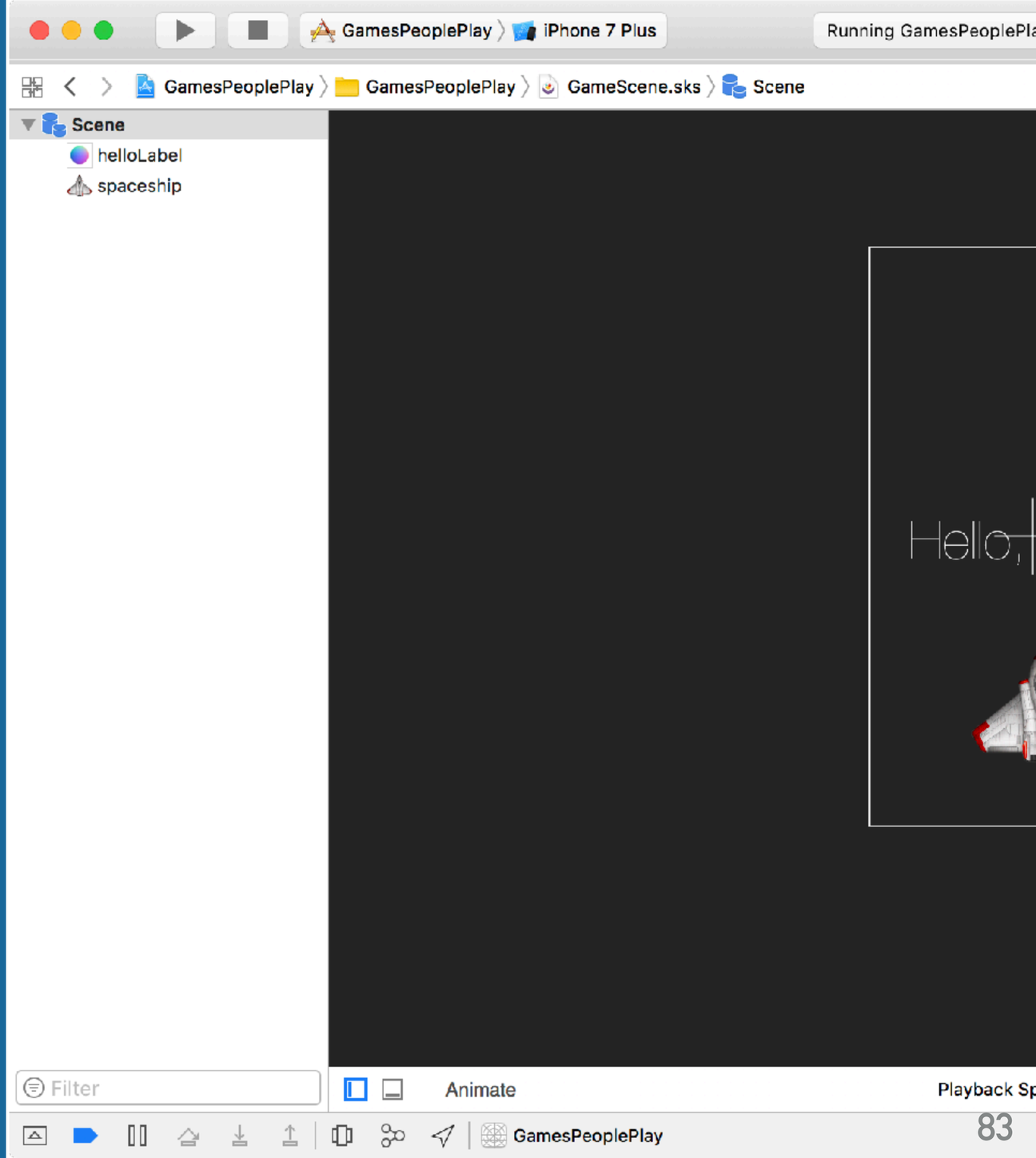
- Parents are nodes that contain other nodes
  - Children are those other nodes
- A child must have a parent object to be drawn to the screen



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

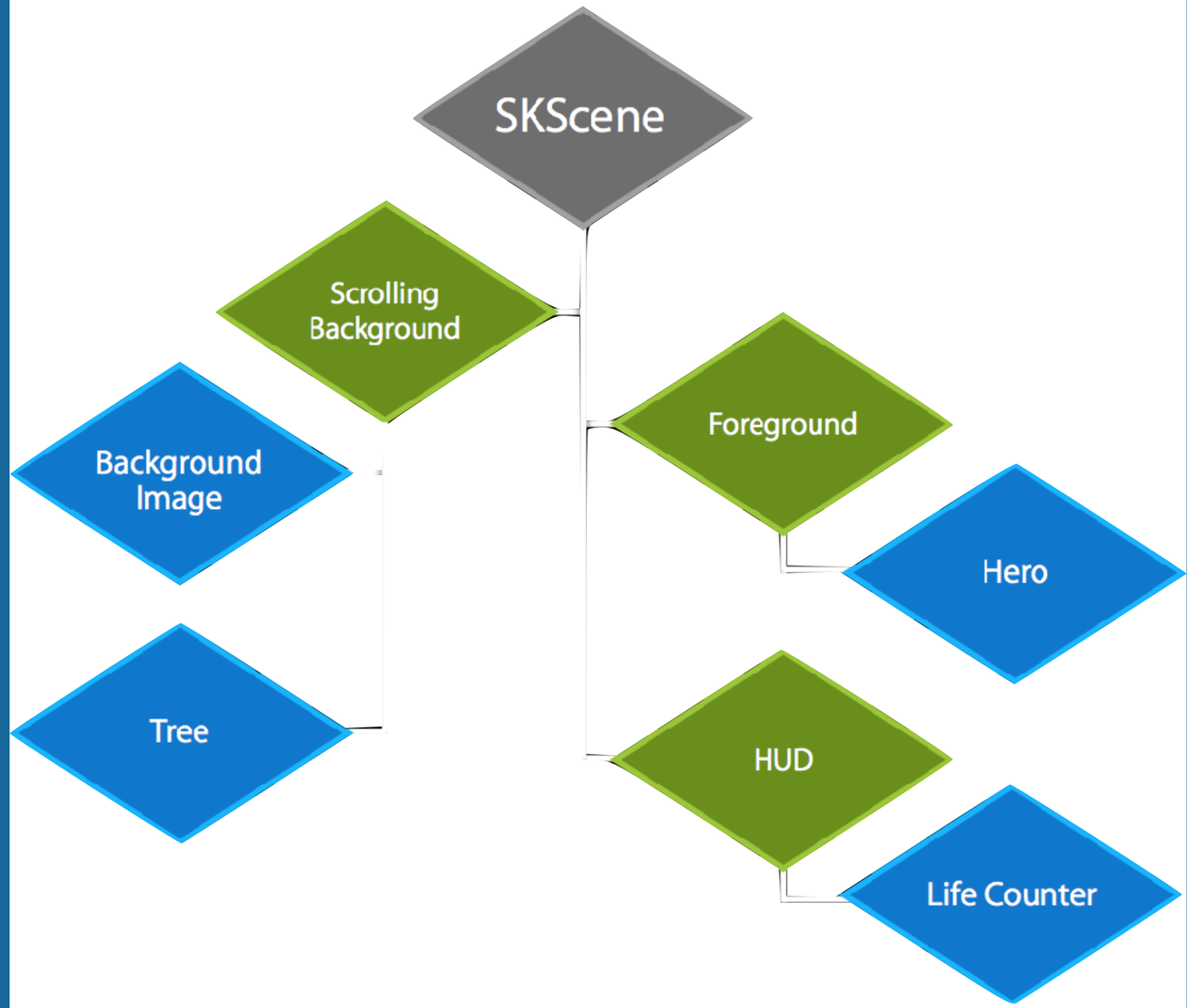
Syntax	Description
/	When placed at the start of the search string, this indicates that the search should be performed on the tree's root node. When placed anywhere but the start of the search string, this indicates that the search should move to the node's children.
//	When placed at the start of the 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.
[characters	ined inside the brackets.
characters	s.

```
// 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))
}
```

ADDING A SCENE

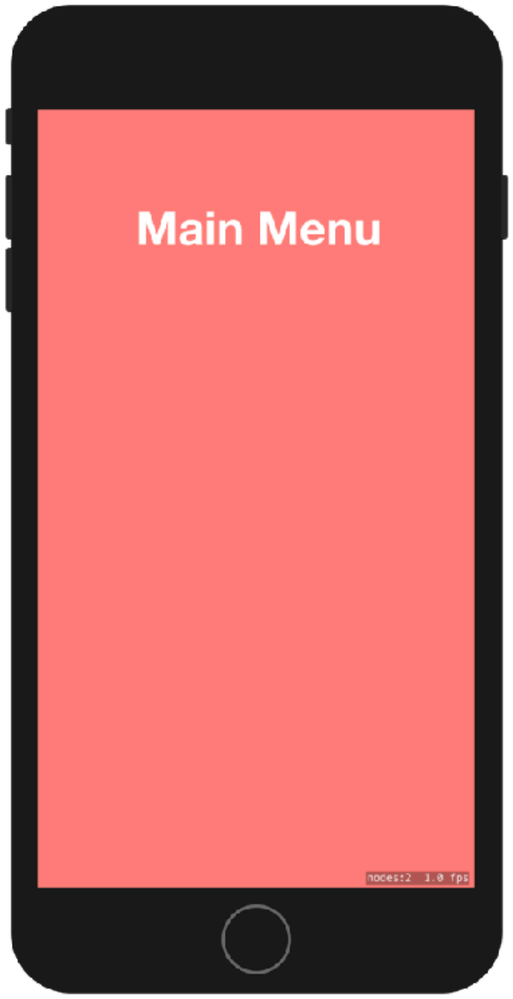
SCENE TRANSITIONS

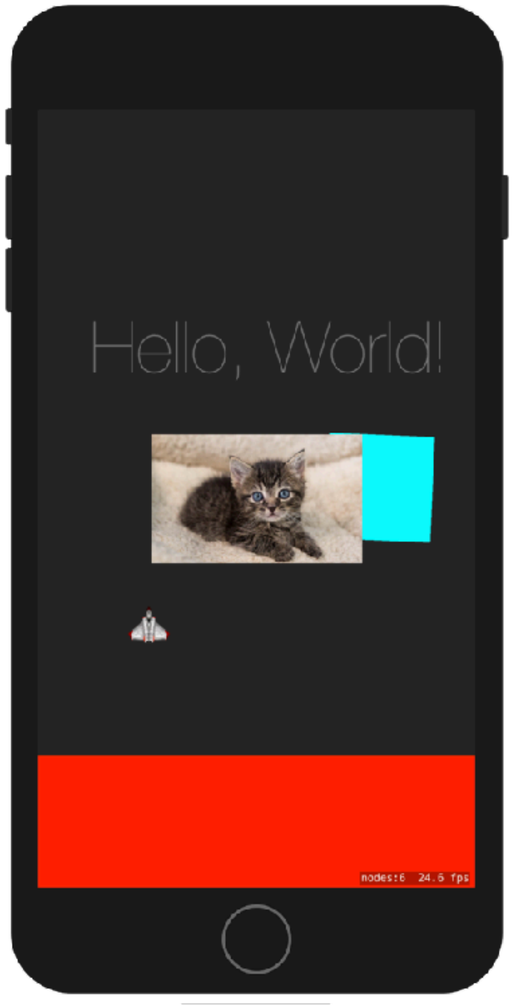
- Root node of the scene graph
- Displayed by a SKView
- Inherits from SKEffectNode
  Runs per-frame loop
  - -update:
  - -didEvaluateActions
  - -didSimulatePhysics



SCENE TRANSITIONS

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





SCENE TRANSITIONS

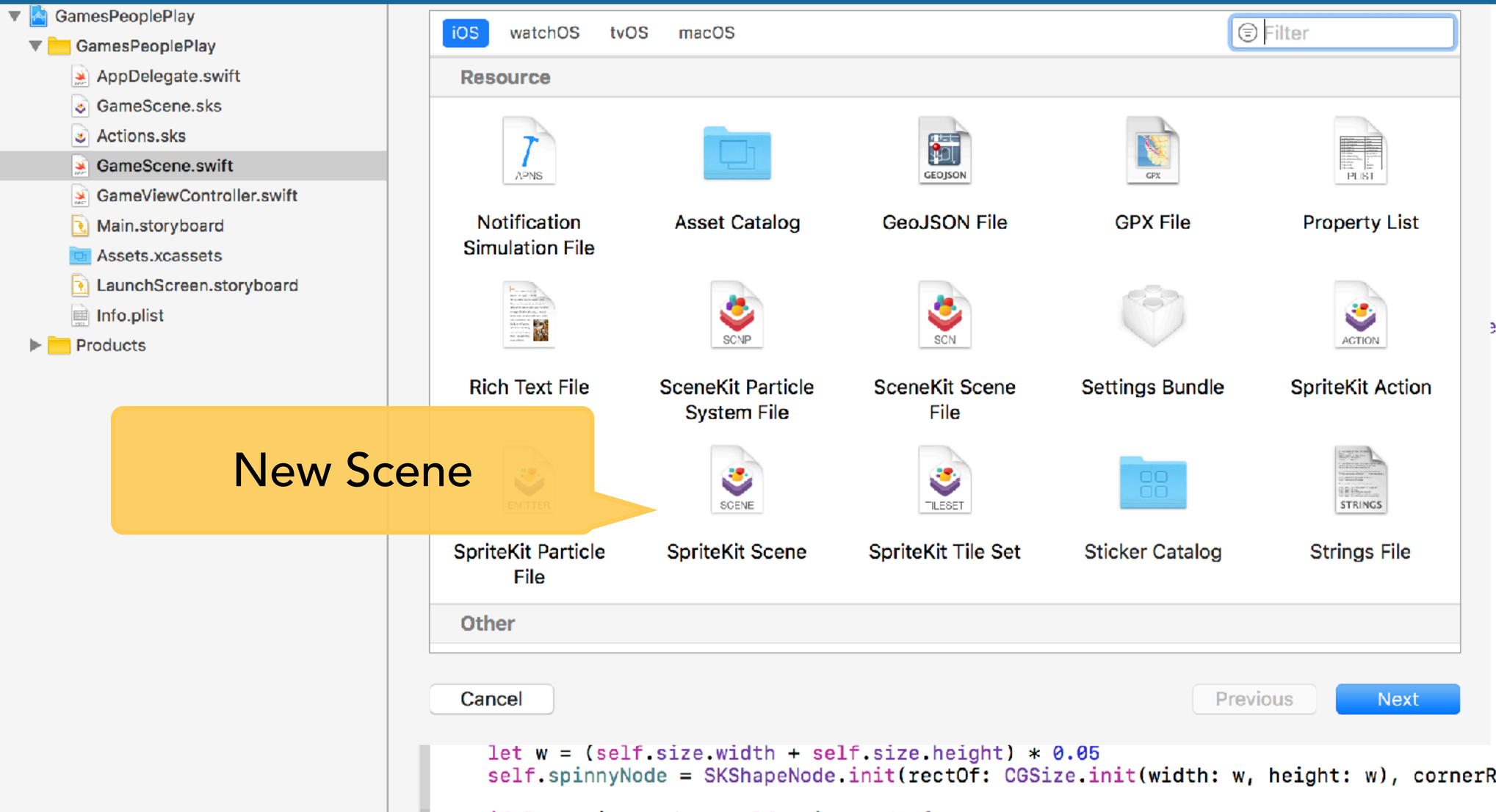
SUBTITLE

- 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

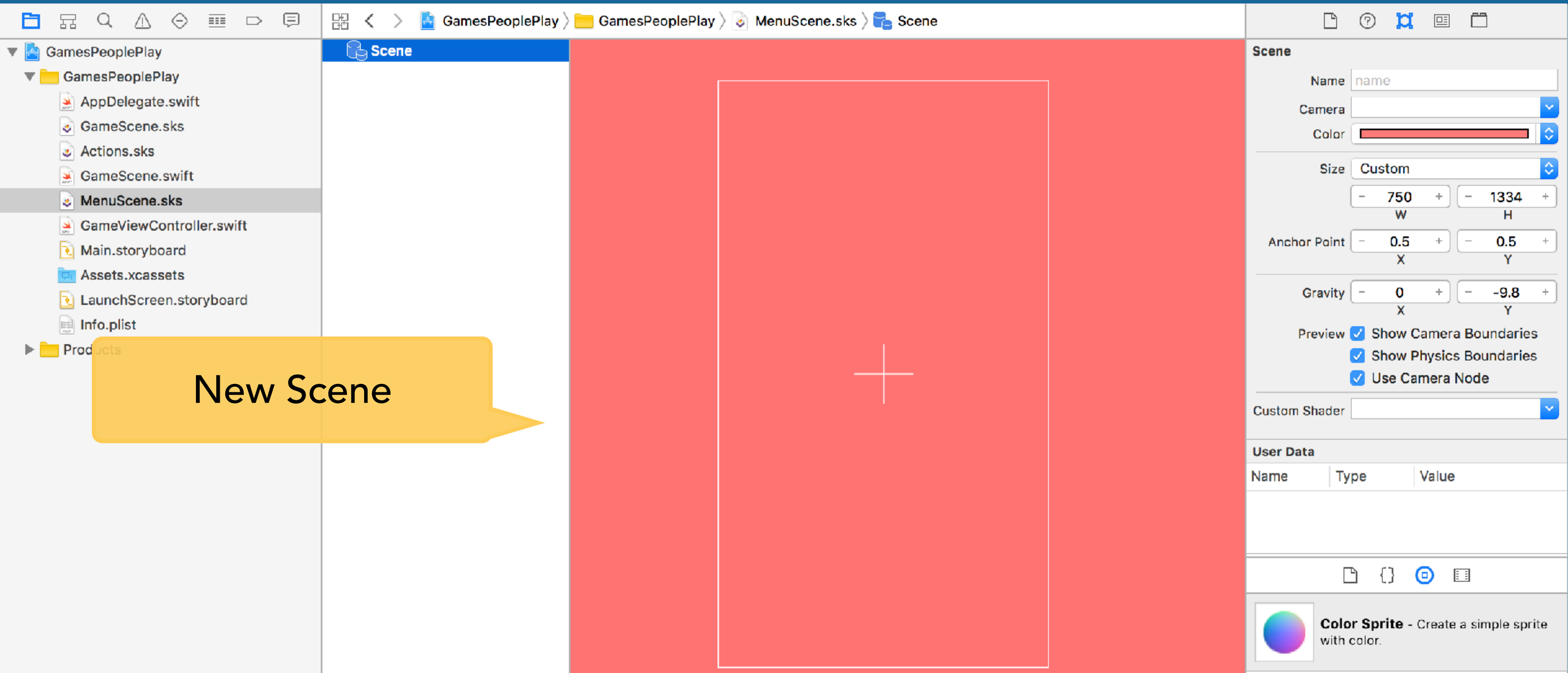
VIEW AND SCENES

```
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.doorsOpenVerticalWithDuration(1.0)

    // Present it
    view?.presentScene(ballScene, transition: transition)

}
```

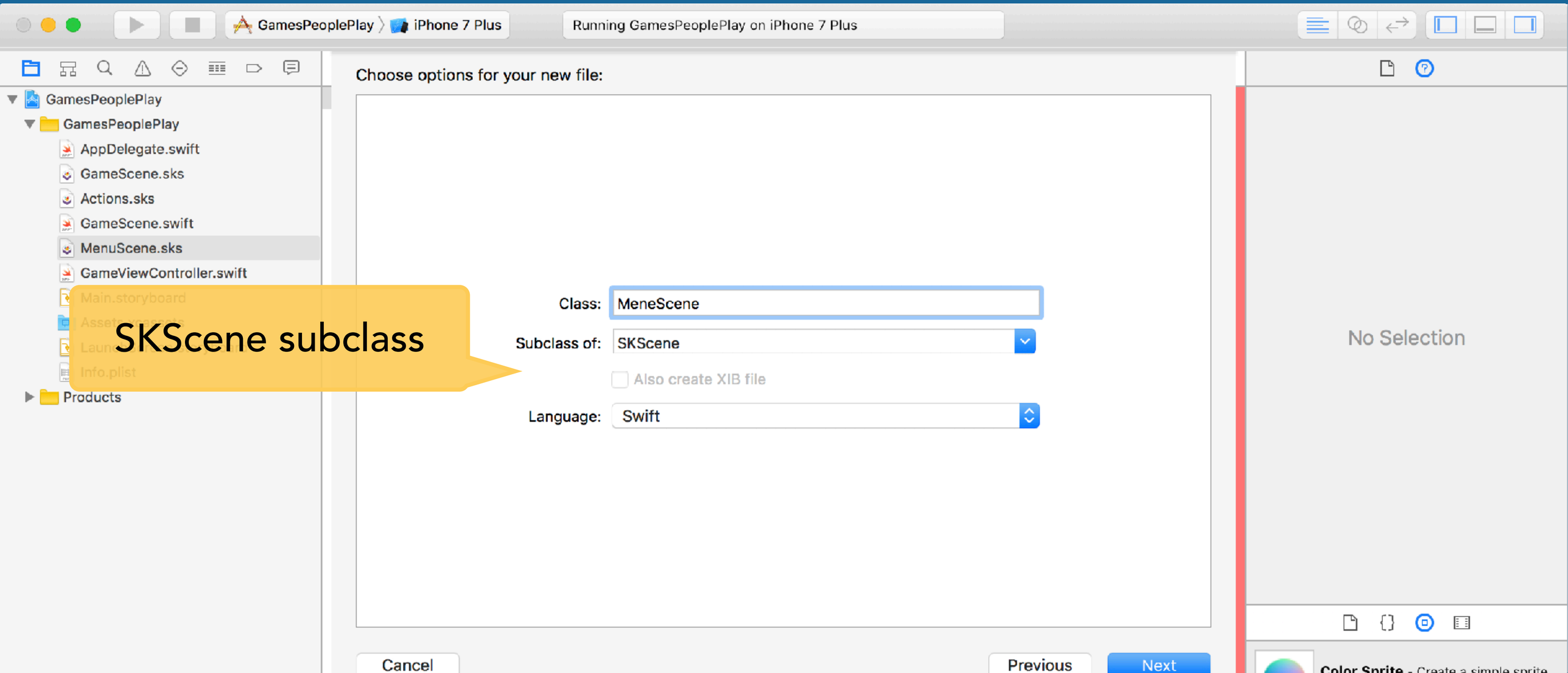
ADDING A SCENE



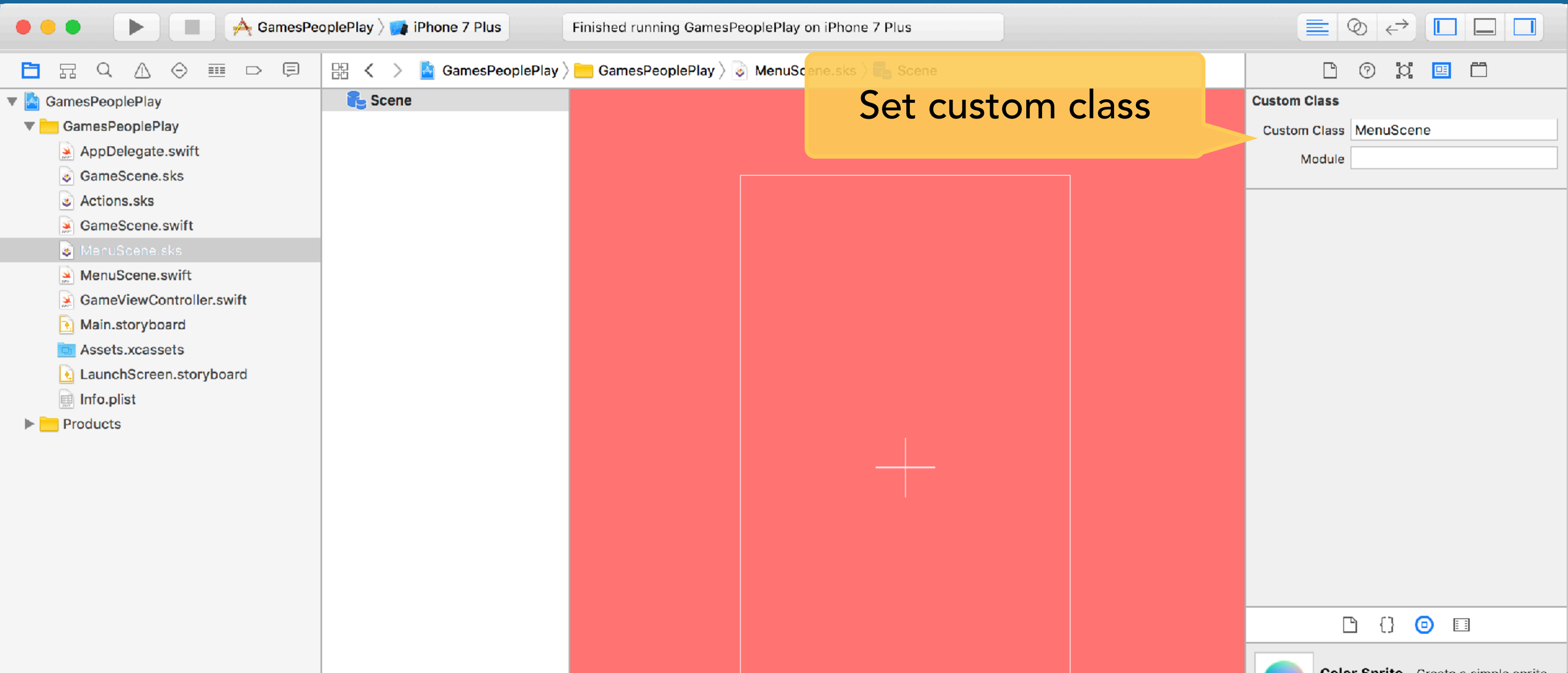
ADDING A SCENE



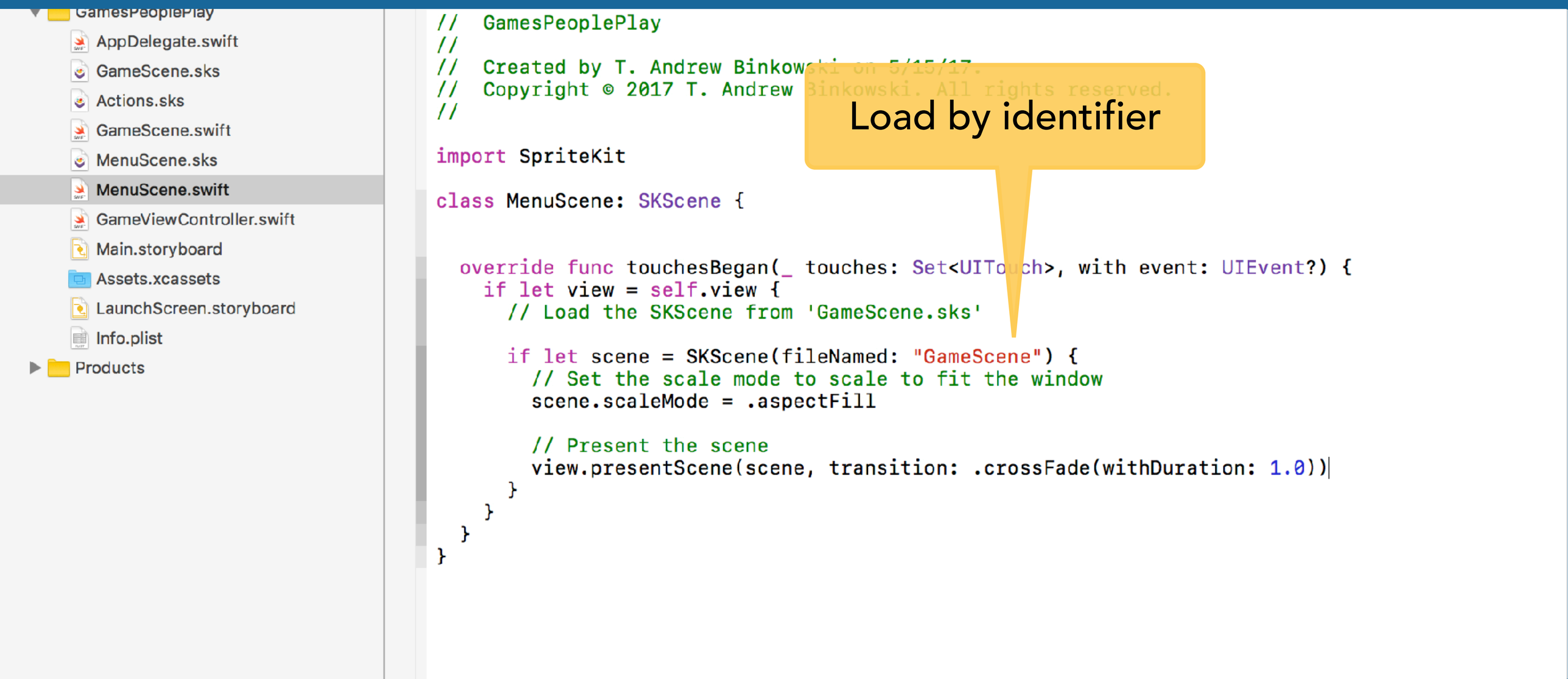
ADDING A SCENE



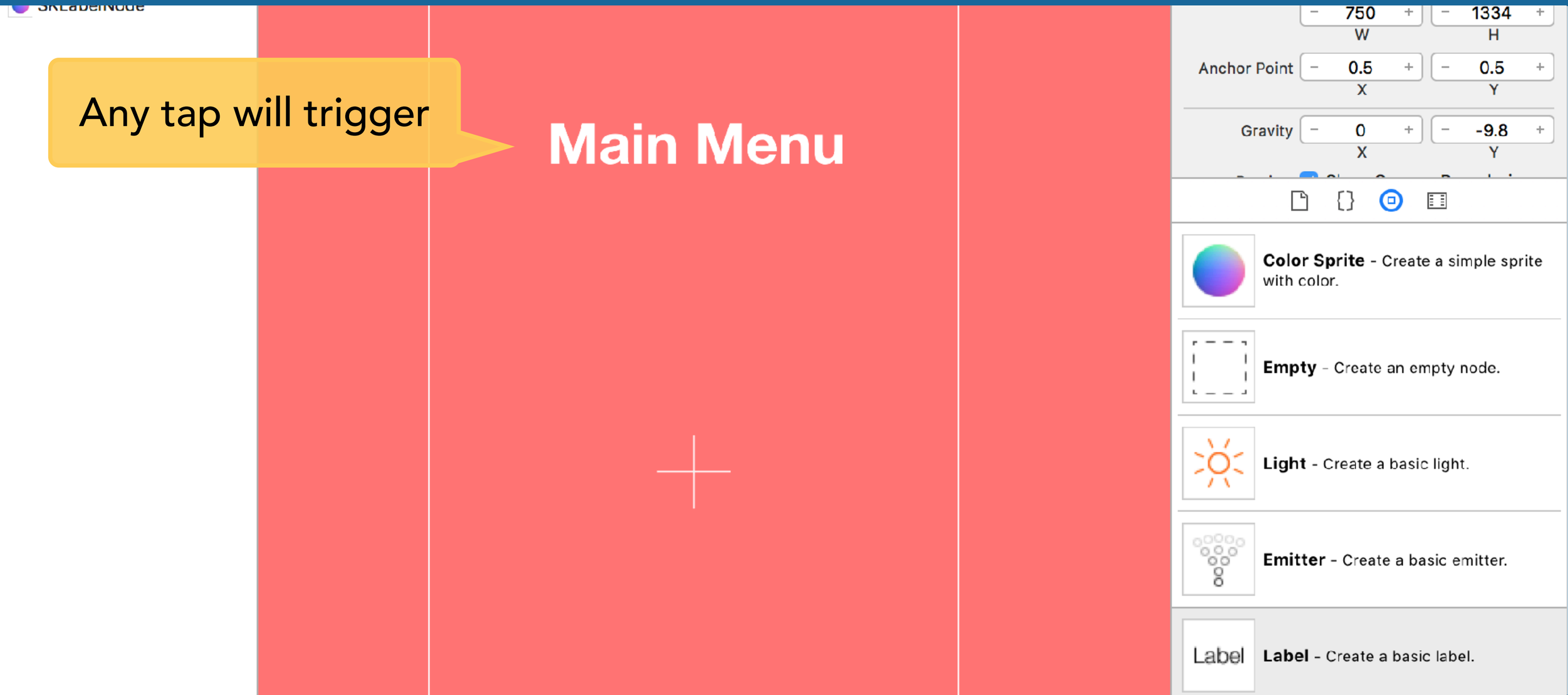
ADDING A SCENE



ADDING A SCENE



ADDING A SCENE



SCENE TRANSITIONS

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

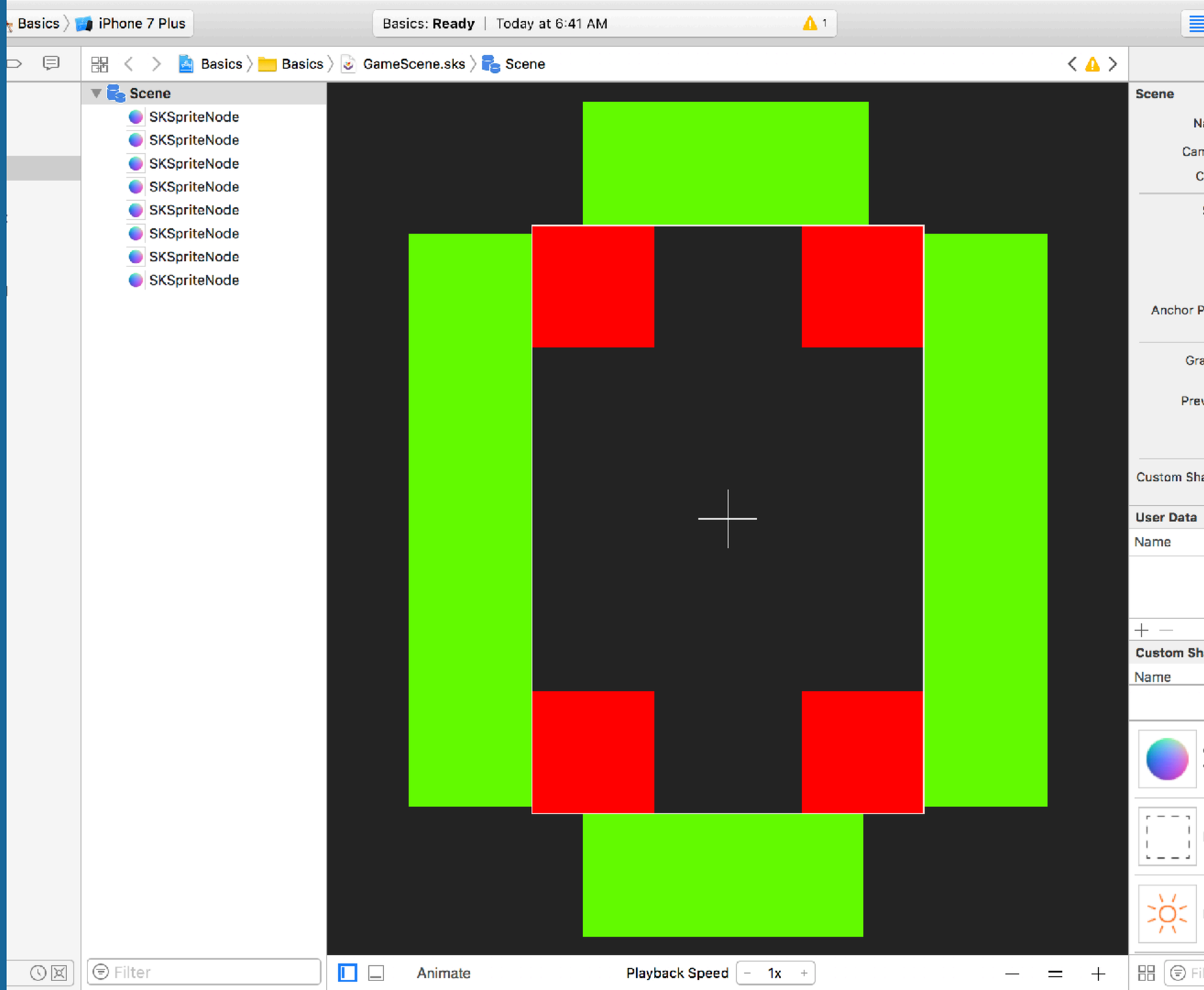
By default both scenes will pause

- 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.

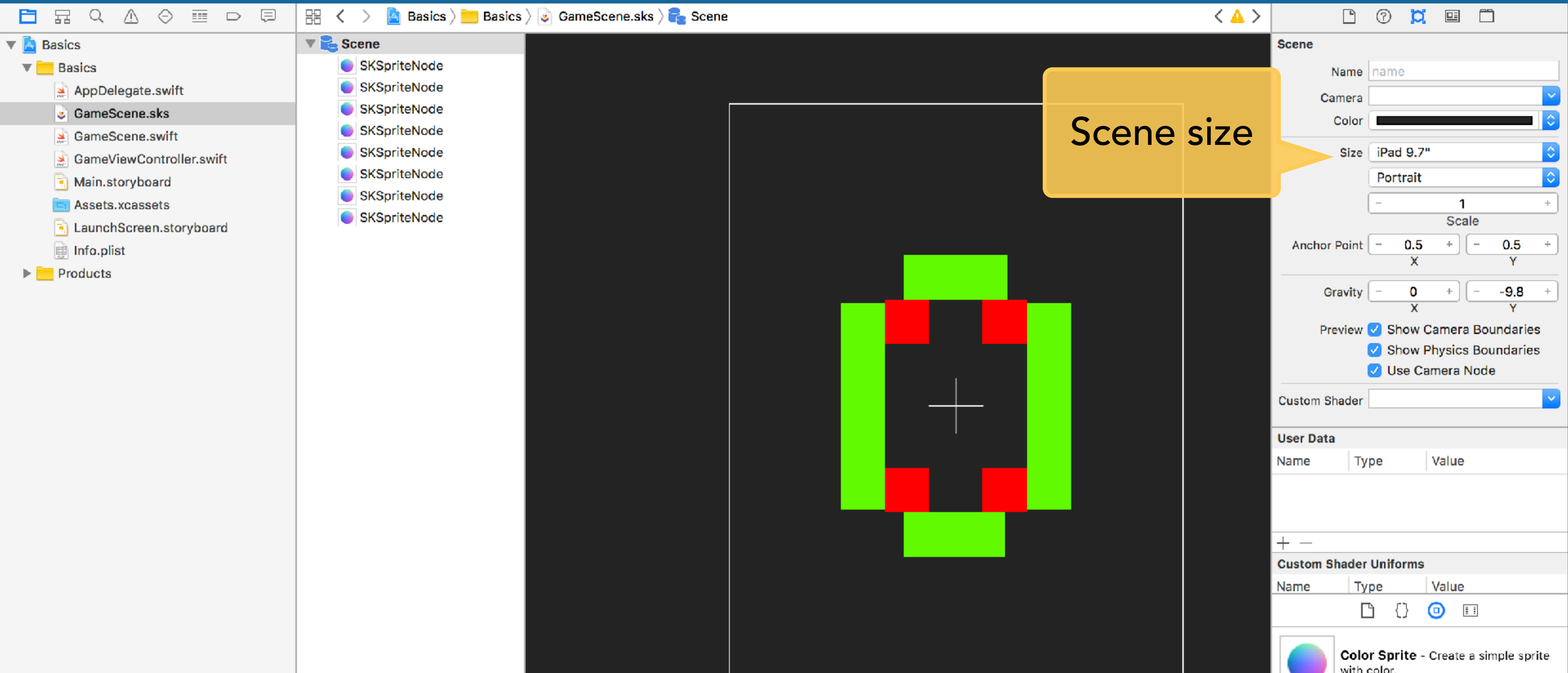
Letterbox

`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.

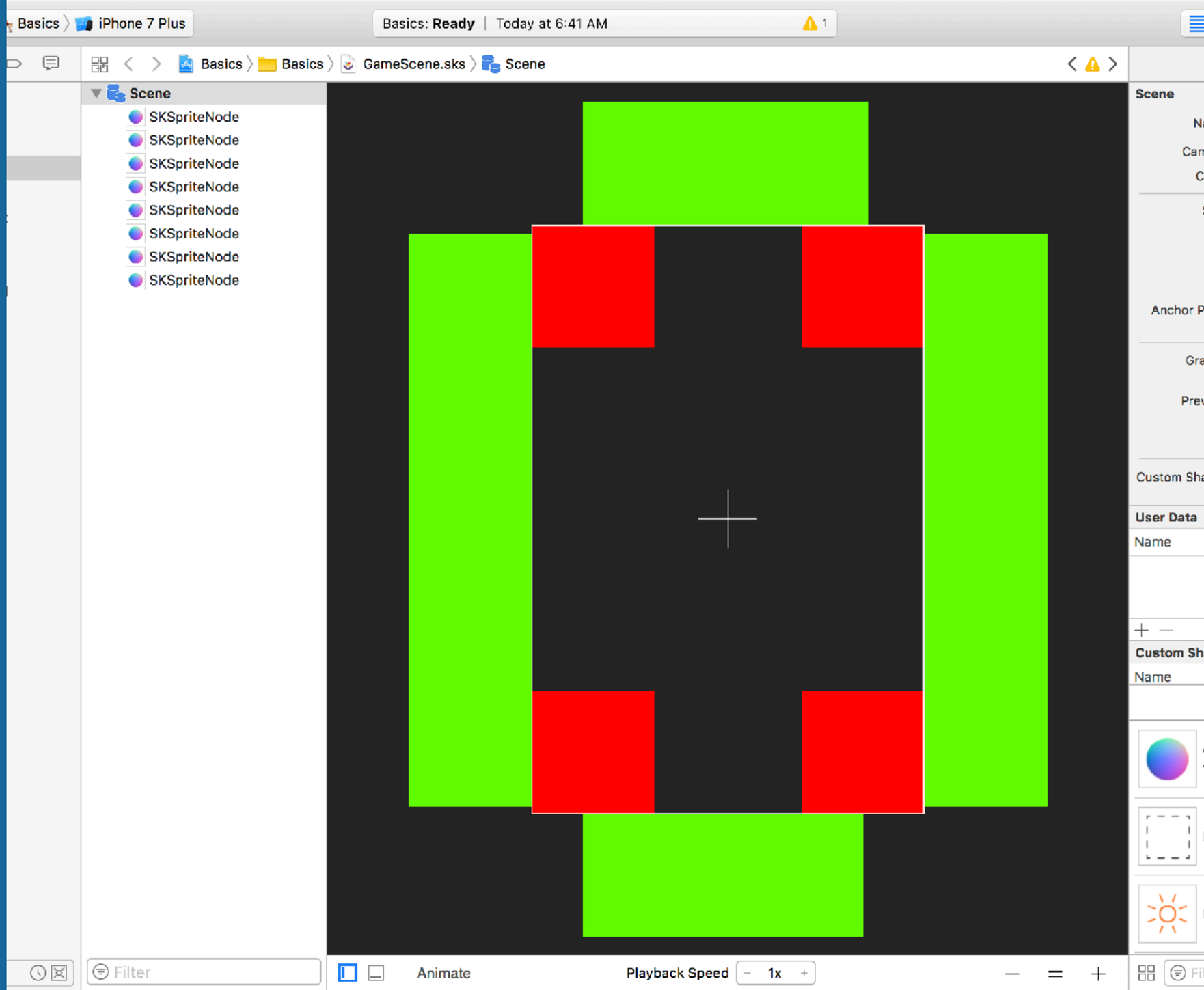
Letterbox

SCALING MODES



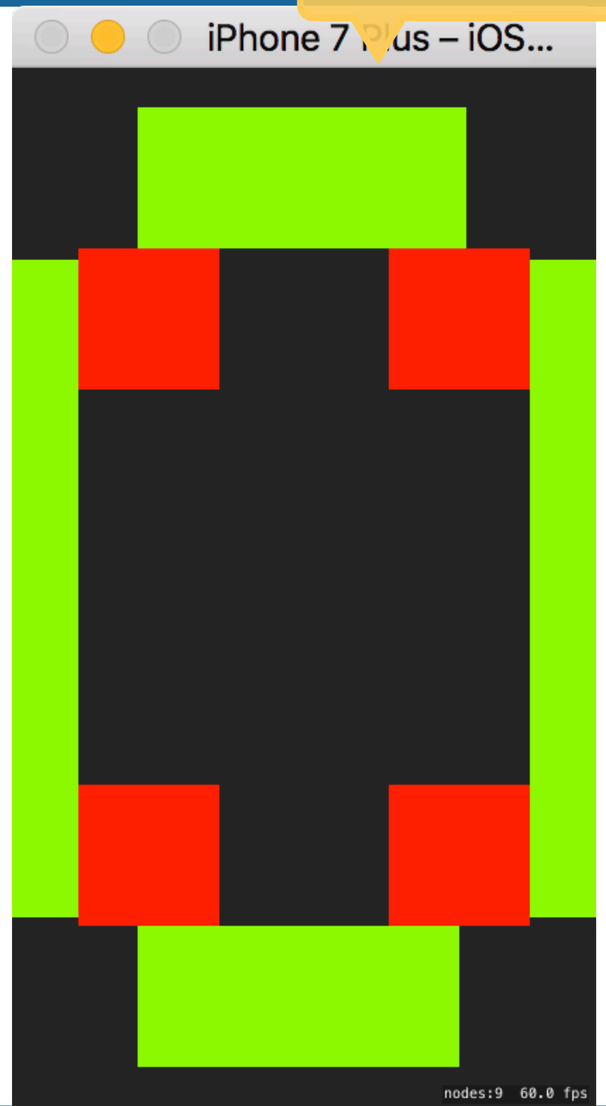
SCALING MODES

- iPhone 4s - red squares



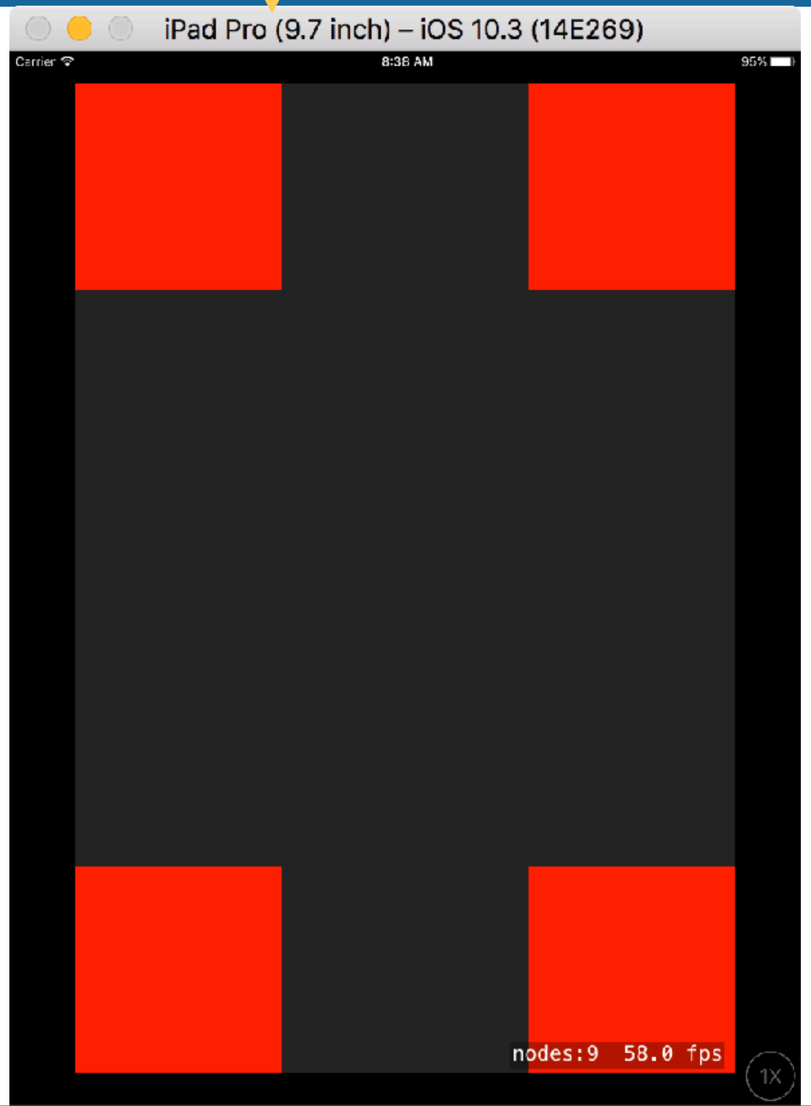
SCALING MODES

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

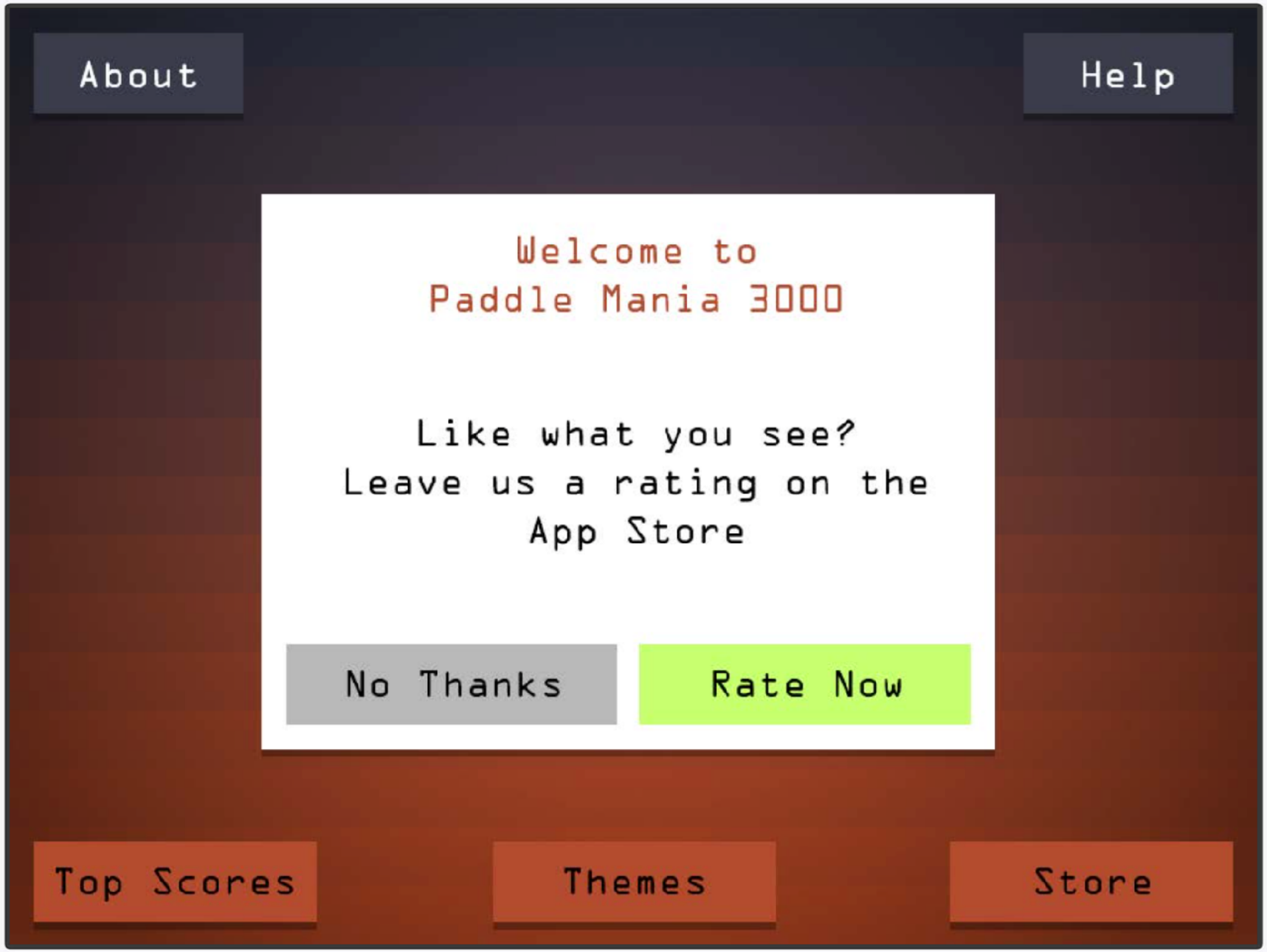
```
class GameViewController: UIViewController {

    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

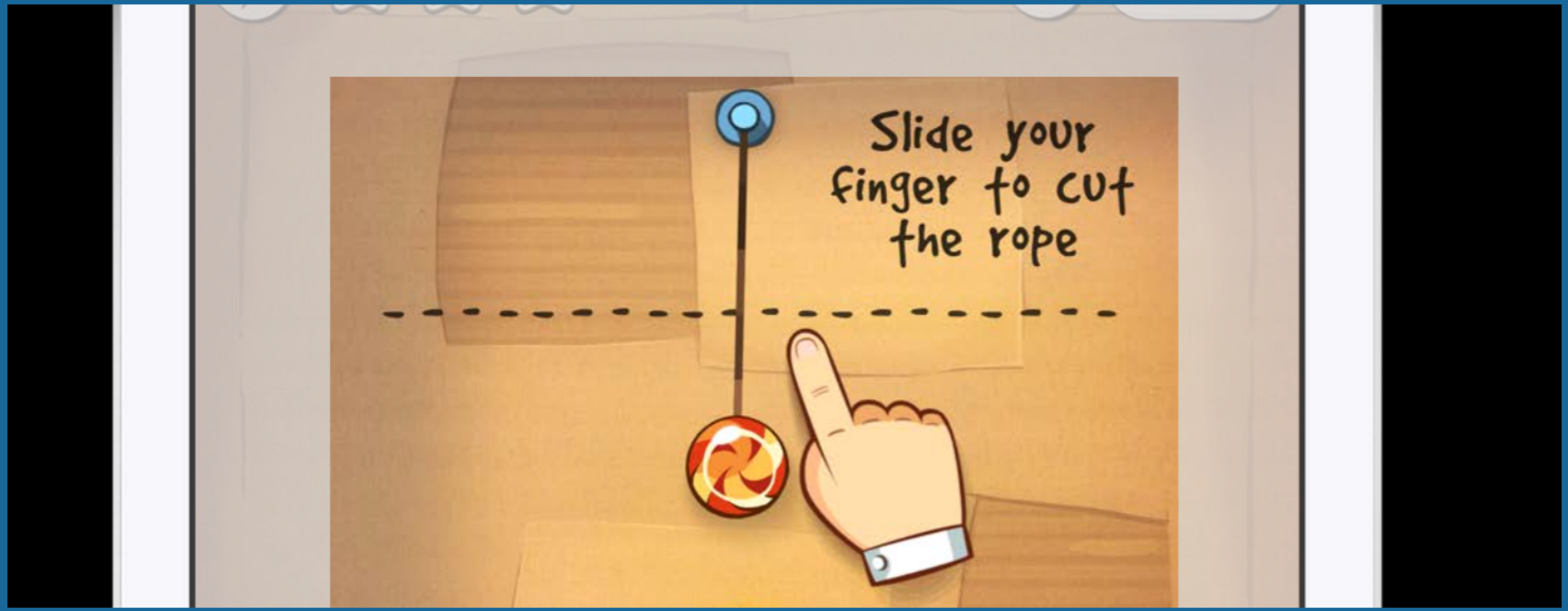
APPLE'S INGREDIENTS OF GREAT GAMES

GREAT GAMES

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

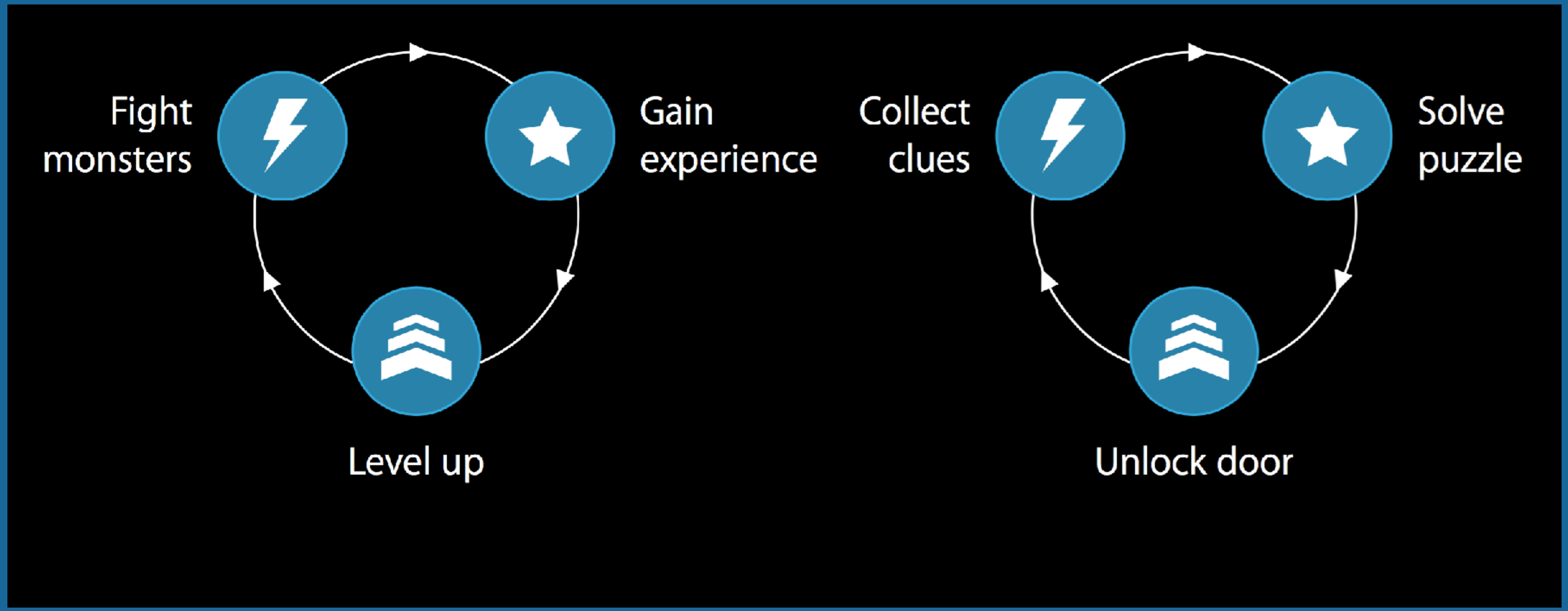


GREAT GAMES



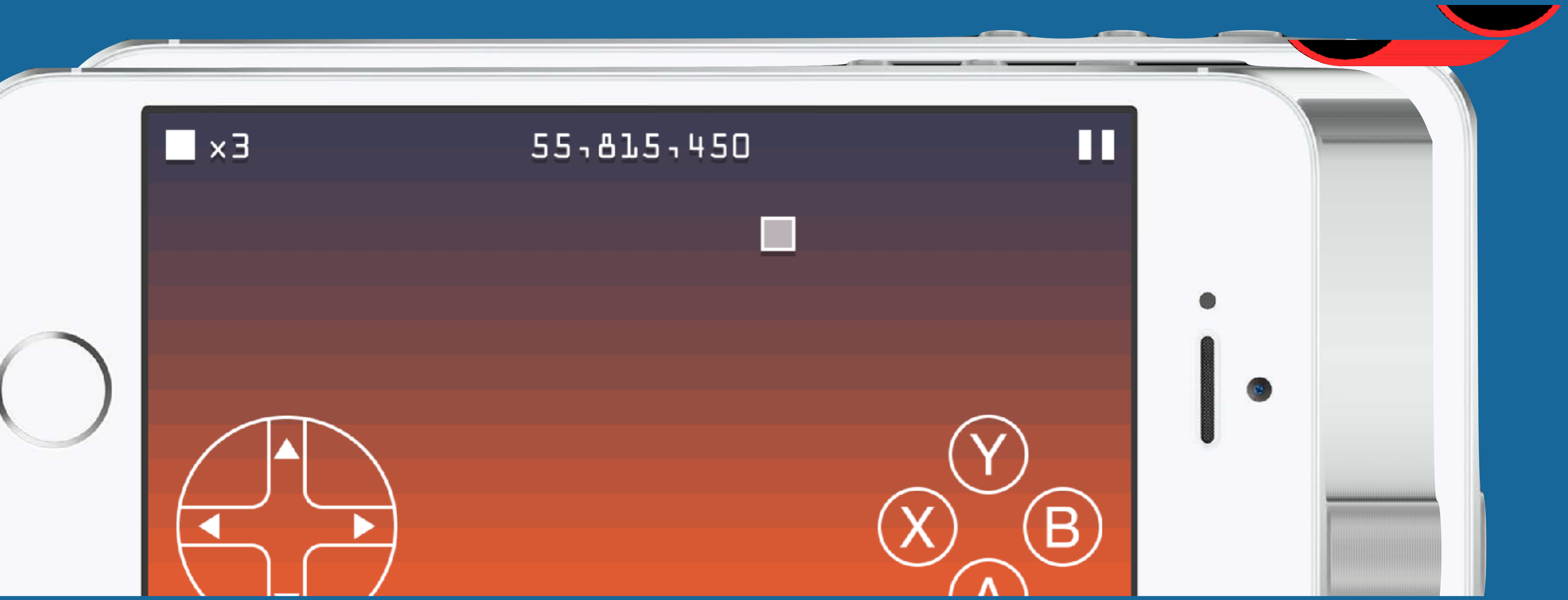
- Be a good teacher

GREAT GAMES



- Tune your core loop

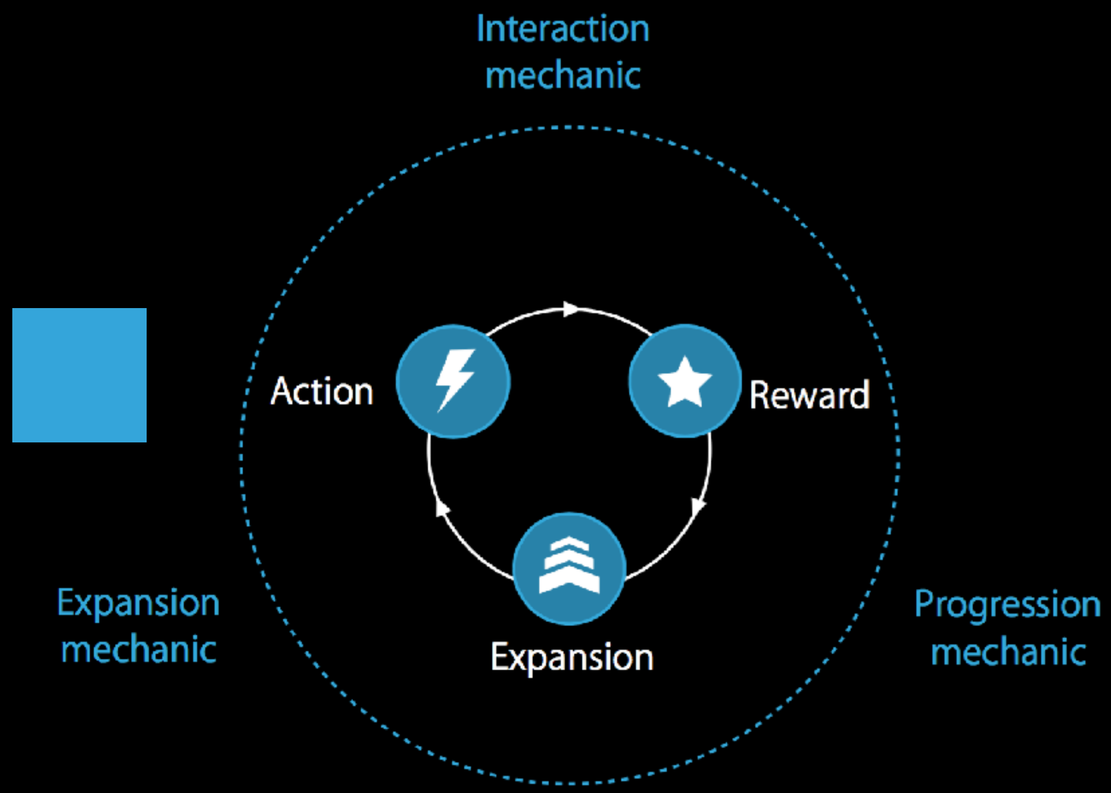
GREAT GAMES



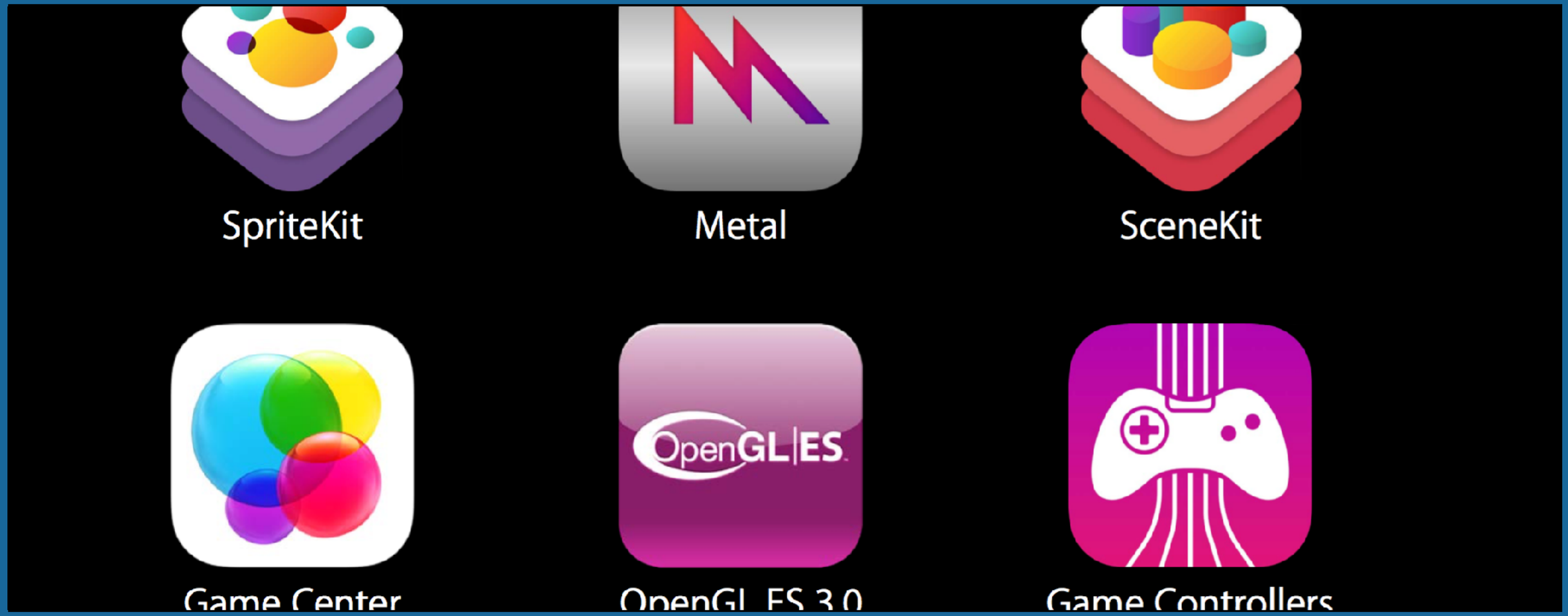
- Design for touch

GREAT GAMES

- Foster Engagement



GREAT GAMES



- Use latest technologies

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 6