Extracted from:

Apple Game Frameworks and Technologies

Build 2D Games with SpriteKit & Swift

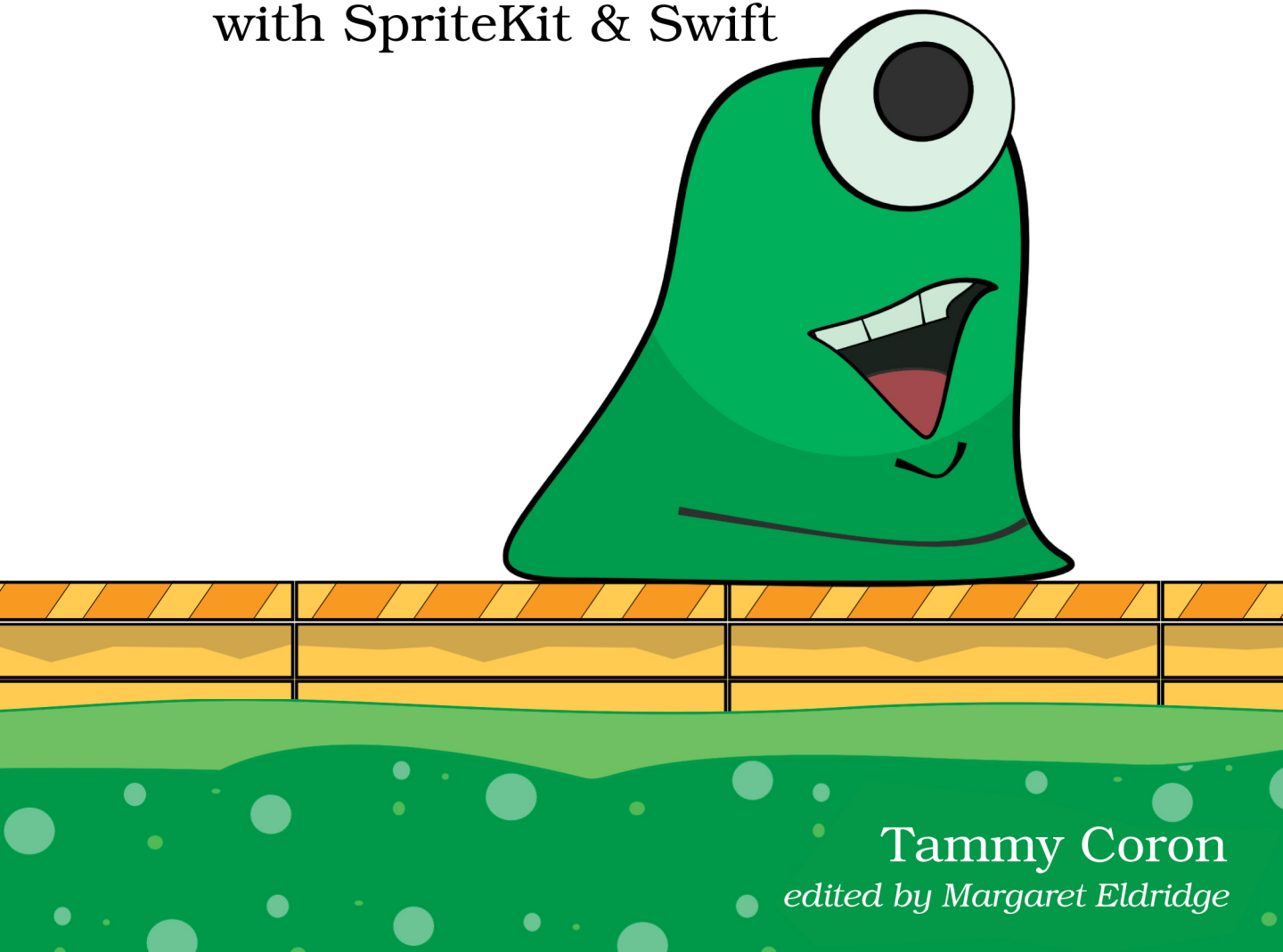
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Apple Game Frameworks and Technologies

Build 2D Games with SpriteKit & Swift



Tammy Coron

edited by Margaret Eldridge

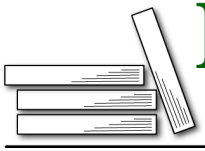
Apple Game Frameworks and Technologies

Build 2D Games with SpriteKit & Swift

Tammy Coron

The Pragmatic Bookshelf

Raleigh, North Carolina



Pragmatic Bookshelf

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Margaret Eldridge
Copy Editor: Katharine Dvorak
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-784-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—April 2021

To my children, Travis and Jake, and to my husband, Bill: I love you guys so very, very much. Life wouldn't be the same without you.

Create an Endless Scrolling Background

Have you ever played a game where the main character stays in one spot, but the background moves from right to left, making it appear as if the player is traveling to the right? This popular technique, known as an *endless scrolling background*, isn't only for player movement—in Gloop Drop, you'll use it to create an infinite flow of gloop beneath the platform.

Adding Image Resources for the Gloop Flow

Select the Assets.xcassets asset catalog and drag the three files, flow_1@1x.png, flow_1@2x.png, and flow_1@3x.png from the resources/Images folder into the root of the Assets.xcassets asset catalog, as shown in the image on page 8.

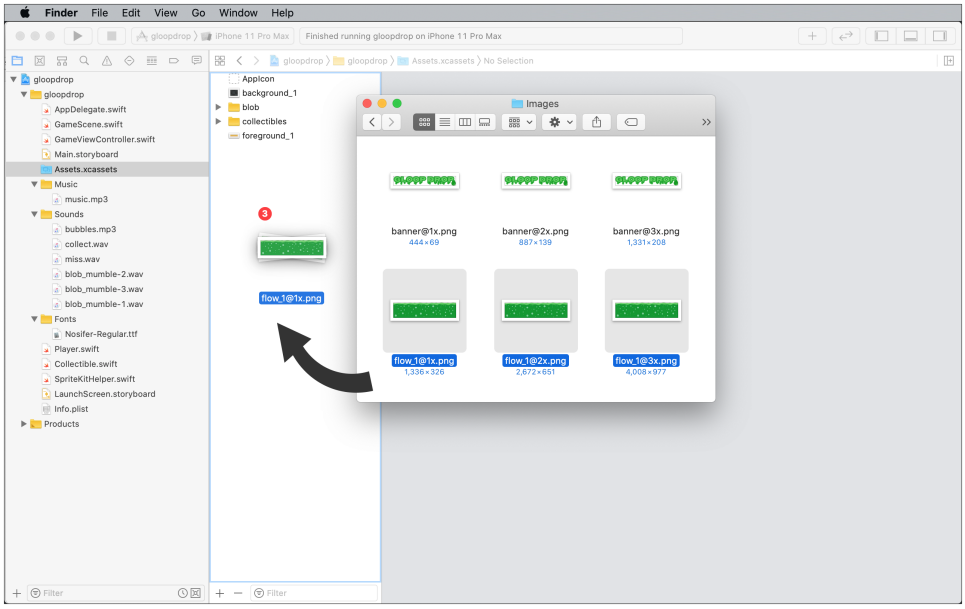
This action creates a flow_1 image set. You'll use this image set to create an endless scrolling background.

Creating Extensions for Scrolling the Background

While it's entirely possible to create a method or two inside the GameScene class to handle scrolling, you'll instead create extensions to handle it so that you can use them elsewhere in your code or in other projects.

Open the SpriteKitHelper.swift file, and to add another extension for the SKSpriteNode class, add the following code below the line that reads extension SKSpriteNode {:

```
// Used to create an endless scrolling background
func endlessScroll(speed: TimeInterval) {

  // Set up actions to move and reset nodes
  let moveAction = SKAction.moveBy(x: -self.size.width,
                                   y: 0, duration: speed)
  let resetAction = SKAction.moveBy(x: self.size.width,
                                    y: 0, duration: 0.0)

  // Set up a sequence of repeating actions
  let sequenceAction = SKAction.sequence([moveAction, resetAction])
  let repeatAction = SKAction.repeatForever(sequenceAction)

  // Run the repeating action
  self.run(repeatAction)
}
```



This code creates a sequence of actions that runs repeatedly. The first action, moveAction, takes the sprite node and moves it to the specified x-position at the desired speed. It uses the width of the node to determine how far to move it. In this case, the image width is 1336, which is the same width as the scene. So, a value of -1336 will move this sprite node to the far left, just outside the screen. The second action, resetAction, then *resets* the x-position back to the far right. These two actions endlessly repeat in a sequence with the help of the sequenceAction and repeatAction actions.

That's a lot of information to take in, but stay with me—there are still a few more things to do.

Your next task is to add a new extension for the SKNode class. Still inside the SpriteKitHelper.swift file, add the following code above the line that reads extension SKSpriteNode {:

```
extension SKNode {

  // Used to set up an endless scroller
  func setupScrollingView(imageNamed name: String, layer: Layer,
                          blocks: Int, speed: TimeInterval) {

    // Create sprite nodes; set positions based on the node's # and width
    for i in 0..<blocks {
      let spriteNode = SKSpriteNode(imageNamed: name)
      spriteNode.anchorPoint = CGPoint.zero
      spriteNode.position = CGPoint(x: CGFloat(i) * spriteNode.size.width,
```

```
                                    y: 0)
      spriteNode.zPosition = layer.rawValue
      spriteNode.name = name

      // Use the custom extension to scroll
      spriteNode.endlessScroll(speed: speed)

      // Add the sprite node to the container
      addChild(spriteNode)
    }
  }
}
```

This code is at the heart of the endless scrolling routine. You can set several options, including the image name to use for the sprite node, the z-position (layer), the number of blocks, and the speed. It then uses this information to create and add a new sprite node, which ultimately calls the endlessScroll(speed:) extension method.

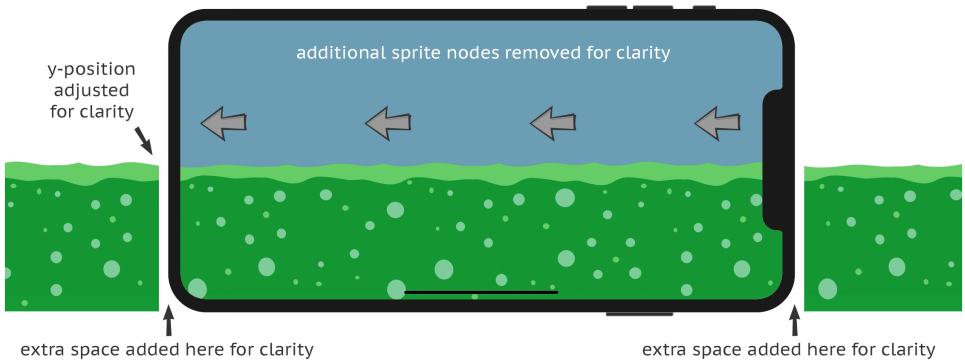
With these two extensions in place, you're ready to get scrolling.

Open the GameScene.swift file and add the following code above the line that reads // MARK: - GAME FUNCTIONS:

```
// MARK: - Gloop Flow & Particle Effects

func setupGloopFlow() {

  // Set up flowing gloop
  let gloopFlow = SKNode()
  gloopFlow.name = "gloopFlow"
  gloopFlow.zPosition = Layer.foreground.rawValue
  gloopFlow.position = CGPoint(x: 0.0, y: -60)

  // Use extension for endless scrolling
  gloopFlow.setupScrollingView(imageNamed: "flow_1",
                               layer: Layer.foreground,
                               blocks: 3, speed: 30.0)

  // Add flow to scene
  addChild(gloopFlow)
}
```

This code pulls it all together by creating a new gloopFlow node to hold the scrolling background sprites. It then uses the setupScrollingView(imageNamed:layer:blocks:speed:) extension method to initiate the scroll.

To get a better idea of how this all works, look at the following image:



In this example, which has a few modifications added for clarity, you can see how the three images are used to show a continuous flow of gloop that moves to the left.

The last step is to call this new setupGloopFlow() method. At the bottom of the didMove(to:) method, add the following code:

```
// Set up the gloop flow
setupGloopFlow()
```

Build and run the project, and you'll see an endless flow of gooey green gloop just beneath the platform as shown in the image on page 11.

You're well on your way to making this gloop flow look good, but it's missing something: ambient sound.

Adding Ambient Sound to the Gloop Flow

Still inside the GameScene.swift file, add the following new property below the existing musicAudioNode declaration:

```
let bubblesAudioNode = SKAudioNode(fileNamed: "bubbles.mp3")
```

This code initializes another SKAudioNode object using the bubbles.mp3 file that you added in Adding Resources for the Sound Effects, on page ?.

With the musicAudioNode object, you ran an action to first lower the volume to 0.0, and then you added an action to gradually increase the volume, giving you that fade-in effect. This time, instead of adjusting the volume, you'll write an action that delays adding the node to the scene.

In the didMove(to:) method, below the action that adjusts the volume of the musicAudioNode object, add the following code:

```
// Run a delayed action to add bubble audio to the scene
```


```
run(SKAction.wait(forDuration: 1.5), completion: { [unowned self] in
  self.bubblesAudioNode.autoplayLooped = true
  self.addChild(self.bubblesAudioNode)
})
```

For reference, you'll end up with something like this:

```
// Use an action to adjust the audio node's volume to 0
musicAudioNode.run(SKAction.changeVolume(to: 0.0, duration: 0.0))

// Run a delayed action on the scene that fades in the music
run(SKAction.wait(forDuration: 1.0), completion: { [unowned self] in
  self.audioEngine.mainMixerNode.outputVolume = 1.0
  self.musicAudioNode.run(SKAction.changeVolume(to: 0.75, duration: 2.0))
})
```

```
// Run a delayed action to add bubble audio to the scene
run(SKAction.wait(forDuration: 1.5), completion: { [unowned self] in
  self.bubblesAudioNode.autoplayLooped = true
  self.addChild(self.bubblesAudioNode)
})
```

Build and run the project and listen to the sweet, sweet ambient sounds of bubbling gloop as it flows endlessly beneath the platform.

If you think adding ambient sound is cool, you're in for a real treat.