Controlling User Interaction on Nodes

Enable your node to respond to user input, like touches or mouse clicks.

Overview

SKNode subclasses UIResponder in iOS and tvOS, and NSResponder in macOS, allowing nodes to respond to user interaction events such as touches and mouse movements.

There are two strategies you can adopt when supporting user interaction.

>> Enable user interaction on a single catch-all node, for example the scene, and calculate the child node or nodes that the user is interacting with by doing custom hit tests.

>> Enable user interaction on every node you want the user to interact with, and subclass them to implement their responder functions.

Catch All User Interaction by Using a Parent Node

Listing 1 shows an example of how to handle all user interaction on a parent node—in this case, the scene itself. The TouchScene class subclasses SKScene and overrides its isUserInteractionEnabled to return true.

LISTING 1 A SCENE THAT RESPONDS TO TOUCHES

```
class TouchScene: SKScene {
    override var isUserInteractionEnabled: Bool {
        get {
            return true
        }
        set {
            // ignore
        }
    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        guard let touch = touches.first else {
            return
        }

        let location = touch.location(in: self)

        let touchedNodes = nodes(at: location)
        let frontTouchedNode = atPoint(location).name
    }
}
```

Note: Alternatively, you could override `sceneDidLoad()` and set `isUserInteractionEnabled` to true in your implementation.

The overridden `touchesBegan(_:with:)` method shows two techniques to find the nodes that have been touched. `nodes(at:)` returns an array of all of the sprites and `atPoint(_:)` returns the top-most touched node.

In this example, the child nodes should have `isUserInteractionEnabled` set to false. The parent scene is responsible for responding to user interactions.

Listing 2 shows the equivalent code for macOS.

LISTING 2 A SCENE THAT RESPONDS TO MOUSE DOWN

```
class TouchScene: SKScene {
    override var isUserInteractionEnabled: Bool {
        get {
            return true
        }
        set {
            // ignore
        }
    }

    override func mouseDown(with event: NSEvent) {
        let location = event.location(in: self)
        let touchedNodes = nodes(at: location)
        let firstTouchedNode = atPoint(location).name
    }
}
```

Enable User Interaction on All Nodes

Listing 3 shows an example of supporting user interaction on individual nodes in your scene. Instances of TouchSpriteNode are added as children of a standard scene and user interaction is handled by each independently. In the case of overlapping nodes, only one will register a user interaction. For example, of two overlapping nodes, the one with the highest zPosition receives the touch. For two nodes with an equal zPosition, the last node in the parent's children array receives the touch.

LISTING 3 A SPRITE NODE THAT RESPONDS TO TOUCHES

```
class TouchSpriteNode: SKSpriteNode {
    override var isUserInteractionEnabled: Bool {
        set {
            // ignore
        }
        get {
            return true
        }
    }

    // For macOS replace this method with `mouseDown(with:)`
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        // User has touched this node
    }
}
```

Handle User Interaction on Hidden Nodes by Using Alpha

Translucent nodes—those with an alpha of less than 1 but greater than 0—still receive user interactions. You can set a node's alpha to leastNonzeroMagnitude to make it effectively transparent and yet still respond to touches or mouse movements, although giving it a color of clear has the same effect.

Review the Bounds of Node User Interaction

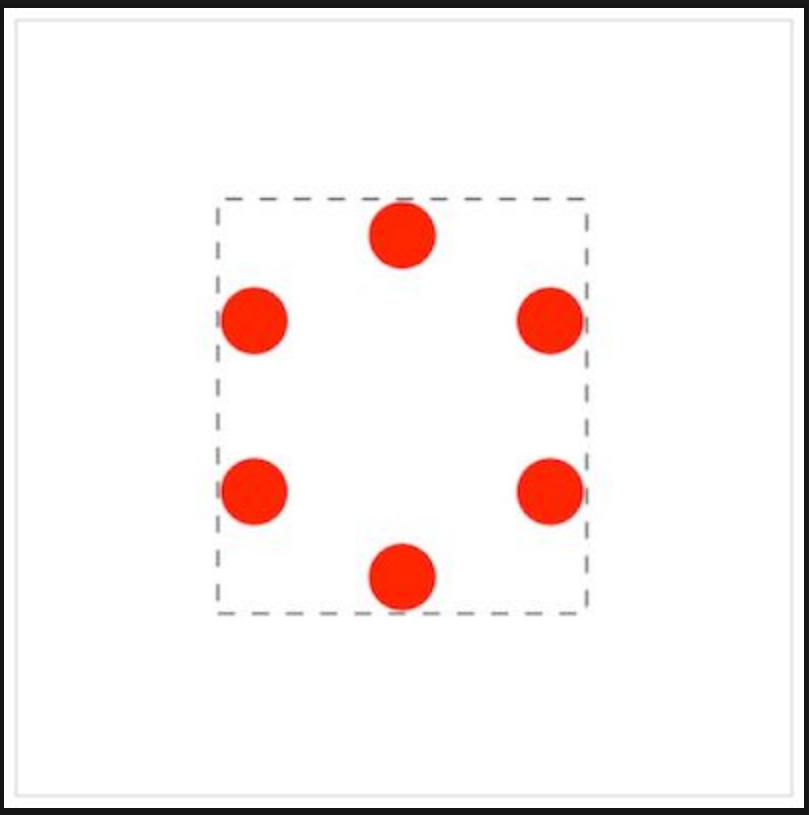
User interaction is based on the bounding box of the node. Sprite nodes containing textures with transparent areas or shape nodes with non-rectangular shapes will still report touches and mouse interactions even if the interaction is over a transparent part of their content. If you build a composite node, the region that receives user interaction events is dependent on the node tree. Listing 4 shows the code used to create a subclassed SKNode that renders six circular shape nodes forming a larger circle.

LISTING 4 A COMPOSITE NODE THAT RESPONDS TO TOUCHES

```
class TouchCompositeNode: SKNode {
    override var isUserInteractionEnabled: Bool {
        set {
            // ignore
        }
        get {
            return true
        }
    }

    let tau = CGFloat.pi * 2

    required init(color: SKColor, radius: CGFloat = 100) {
        super.init()

        stride(from: 0, to: tau, by: tau / 6).forEach {

            let node = SKShapeNode(circleOfRadius: 20)

            node.fillColor = color
            node.position = CGPoint(x: sin($0) * radius,
                                    y: cos($0) * radius)

            addChild(node)
        }
    }

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

    // For macOS replace this method with `mouseDown(with:)`
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        // User has touched this node
    }
}
```

If you were to add an instance of TouchCompositeNode to an SKScene with no other nodes behind it, all touches inside its calculateAccumulatedFrame() would be reported. Figure 1 shows the accumulated frame of TouchCompositeNode. Any user events within the dashed line will call touchesBegan(_:with:).

FIGURE 1 USER EVENTS REPORTED INSIDE A NODE'S ACCUMULATED FRAME



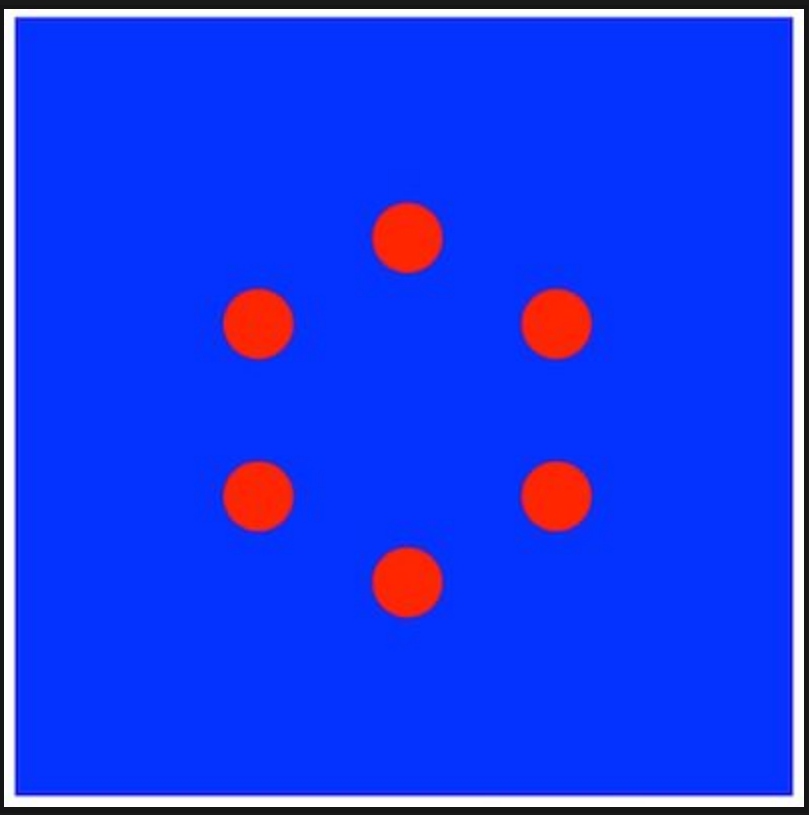
However, if you were to place an instance of TouchCompositeNode above another node using, for example, the code in Listing 5, only touches on its child nodes would be reported.

LISTING 5 PLACING A COMPOSITE NODE OVER A BACKGROUND

```
let composite = TouchCompositeNode(color: .red)
composite.position = CGPoint(x: 400, y: 400)

let backgroundNode = SKSpriteNode(color: .blue,
                                  size: CGSize(width: 500, height: 500))
backgroundNode.position = CGPoint(x: 400, y: 400)

scene.addChild(backgroundNode)
scene.addChild(composite)
```

In this case, only touches or mouse events over the red spots shown in Figure 2 will call `touchesBegan(_:with:)`.

FIGURE 2 USER EVENTS REPORTED OVER A NODE'S NON-TRANSPARENT CONTENT



> **Important:** Nodes that have user interaction disabled will block touch and mouse events on nodes behind them where they overlap.