Detecting Changes at Each Step of an Animation

Get notified of a property change on your node subclass and retrieve the amount of change.

Overview

Generally, actions do not call public methods on nodes. For example, if you want to subclass `SKNode` to respond to a `move(to:duration:)` action, you might consider overriding its `position` property to add a didSet observer.

```
class MovingNode: SKSpriteNode {
    override var position: CGPoint {
        didSet {
            // code to react to position change
        }
    }
}
```

However, because a move action that's running on an instance of MovingNode (defined in the code listing above) doesn't set its position, the observer isn't invoked and thus, your code is never executed.

In this case, the solution is to use `SKSceneDelegate` and compare the node's position across two of its callbacks. You save the node's initial position in `update(_:for:)`, which is called at the beginning of each frame, then calculate any change in position in `didEvaluateActions(for:)`, which is called after actions have been evaluated.

The following code demonstrates an example of this strategy:

```
let node = SKNode()
var nodePosition = CGPoint()

func update(_ currentTime: TimeInterval, for scene: SKScene) {
    nodePosition = node.position
}

func didEvaluateActions(for scene: SKScene) {
    let distance = hypot(node.position.x - nodePosition.x,
                         node.position.y - nodePosition.y)

    if distance > 0 {
        // code to react to position change
    }
}
```