Creating a Look-At Constraint

Make a node automatically rotate itself based on the changing position of another node, by using orientation constraints.

Overview

A common use for orientation constraints is to make a look-at constraint. For example, you may create a look-at constraint to make a pair of eyes follow a moving object or to have a rocket point in the direction of its target.

The following code shows how to create a constraint to make a sprite node named pointer always point toward a circular-shape node named target. The texture assigned to pointer is an upward-pointing arrow and, because SpriteKit's angular coordinates have their origin at the three o'clock position, the orient constraint has an offset of 90° (pi/2 radians) so that the arrow is correctly oriented toward the target.

```
let target = SKShapeNode(circleOfRadius: 10)
target.position = CGPoint(x: 100, y: 100)
scene.addChild(target)

let pointer = SKSpriteNode(imageNamed: "arrowUp.png")
pointer.position = CGPoint(x: 200, y: 200)
scene.addChild(pointer)

let lookAtConstraint = SKConstraint.orient(to: target,
                                           offset: SKRange(constantValue:
-CGFloat.pi / 2))
pointer.constraints = [ lookAtConstraint ]
```

A further orientation constraint, zRotation(_:), can be used in combination with the look-at constraint shown above to limit rotation. When you add the code above, the arrow's rotation is limited so that it never points downward:

```
let limitLookAt = SKConstraint.zRotation(SKRange(lowerLimit: -CGFloat.pi / 2,
                                                 upperLimit: CGFloat.pi / 2))

pointer.constraints = [ lookAtConstraint, limitLookAt ]
```