Swift – Create Progress Indicator using CAReplicatorLayer

ddas.tech/swift-create-progress-indicator-using-careplicatorlayer/

November 20, 2022

This post will be a continuation to our previous post on Introduction to CAReplicatorLayer. In this post we will apply our learnings and will create a couple of **Progress Indicator using CAReplicatorLayer**.

Let's begin by creating the root layer and then adding different variations of CAReplicatorLayer to create progress indicators

Table of Contents


```
let rootLayer = CALayer()

func applicationDidFinishLaunching(_ aNotification: Notification) {
    if let frame = self.window.contentView?.frame{
    self.rootLayer.frame = frame
    self.rootLayer.backgroundColor = NSColor.black.cgColor
    self.window.contentView?.layer = self.rootLayer
    self.window.contentView?.wantsLayer = true
    }
self.progressIndicatorSampleOne()
//  self.progressIndicatorSampleTwo()
//  self.progressIndicatorSampleThree()
}
```

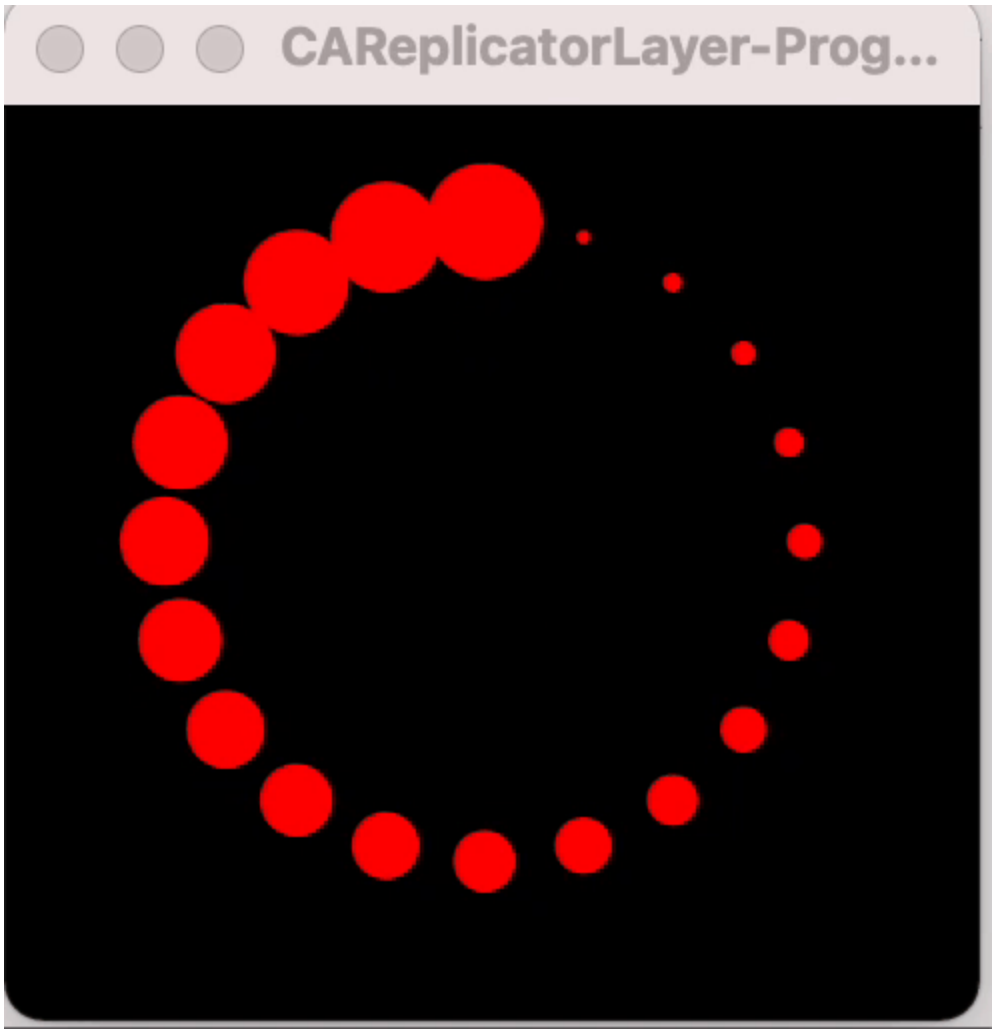
Progress Indicator using CAReplicatorLayer – opacity animation

```
//    Progress Indicator using CAReplicatorLayer – opacity animation
func progressIndicatorSampleOne(){
    let replicatorLayer = CAReplicatorLayer()
    replicatorLayer.frame = CGRect(x: 20, y: 20, width: 200, height: 200)
    replicatorLayer.borderColor = NSColor.black.cgColor
    replicatorLayer.cornerRadius = 5.0
    replicatorLayer.borderWidth = 1.0

    let circle = CALayer()
    circle.frame = CGRect(origin: CGPoint.zero,
                          size: CGSize(width: 20, height: 20))
    circle.backgroundColor = NSColor.systemGreen.cgColor
    circle.cornerRadius = 10
    circle.position = CGPoint(x: 20, y: 100)
    replicatorLayer.addSublayer(circle)

    let fadeOut = CABasicAnimation(keyPath: "opacity")
    fadeOut.fromValue = 1
    fadeOut.toValue = 0
    fadeOut.duration = 1
    fadeOut.repeatCount = Float.greatestFiniteMagnitude
    circle.add(fadeOut, forKey: nil)


    let instanceCount = 20
    replicatorLayer.instanceCount = instanceCount
    replicatorLayer.instanceDelay = fadeOut.duration / CFTimeInterval(instanceCount)

    let angle = -CGFloat.pi * 2 / CGFloat(instanceCount)
    replicatorLayer.instanceTransform = CATransform3DMakeRotation(angle, 0, 0, 1)
self.rootLayer.addSublayer(replicatorLayer)
}
```

Swift CAReplicatorLayer – Opacity Animation

Progress Indicator using CAReplicatorLayer – scale animation

```
//Progress Indicator using CAReplicatorLayer – scale animation
func progressIndicatorSampleTwo(){

    let replicatorLayer = CAReplicatorLayer()
    replicatorLayer.frame = CGRect(x: 20, y: 20, width: 200, height: 200)
    replicatorLayer.borderColor = NSColor.black.cgColor
    replicatorLayer.cornerRadius = 5.0
    replicatorLayer.borderWidth = 1.0

    let circle = CALayer()
    circle.frame = CGRect(origin: CGPoint.zero,
                          size: CGSize(width: 30, height: 30))
    circle.backgroundColor = NSColor.red.cgColor
    circle.cornerRadius = 15
    circle.position = CGPoint(x: 20, y: 100)
    replicatorLayer.addSublayer(circle)

    let shrinkAnimation = CABasicAnimation(keyPath: "transform.scale")
    shrinkAnimation.fromValue = 1
    shrinkAnimation.toValue = 0.1
    shrinkAnimation.duration = 1
    shrinkAnimation.repeatCount = Float.greatestFiniteMagnitude
    circle.add(shrinkAnimation, forKey: nil)


    let instanceCount = 20
    replicatorLayer.instanceCount = instanceCount
    replicatorLayer.instanceDelay = shrinkAnimation.duration /
CFTimeInterval(instanceCount)

    let angle = -CGFloat.pi * 2 / CGFloat(instanceCount)
    replicatorLayer.instanceTransform = CATransform3DMakeRotation(angle, 0, 0, 1)
    rootLayer.addSublayer(replicatorLayer)
}
```



Swift CAReplicatorLayer – Scale Animation

Progress Indicator using CAReplicatorLayer – opacity and scale animation

```
//Progress Indicator using CAReplicatorLayer – opacity and scale animation
func progressIndicatorSampleThree(){

    let replicatorLayer = CAReplicatorLayer()
    replicatorLayer.frame = CGRect(x: 20, y: 20, width: 200, height: 200)
    replicatorLayer.borderColor = NSColor.black.cgColor
    replicatorLayer.cornerRadius = 5.0
    replicatorLayer.borderWidth = 1.0

    let circle = CALayer()
    circle.bounds = CGRect(x: 0, y: 0, width: 30, height: 8)
    circle.backgroundColor = NSColor.blue.cgColor
    circle.cornerRadius = 2
    circle.position = CGPoint(x: 20, y: 100)
    replicatorLayer.addSublayer(circle)

    let fadeOut = CABasicAnimation(keyPath: "opacity")
    fadeOut.fromValue = 1
    fadeOut.toValue = 0
    fadeOut.duration = 1
    fadeOut.repeatCount = Float.greatestFiniteMagnitude
    circle.add(fadeOut, forKey: nil)

    let shrinkAnimation = CABasicAnimation(keyPath: "transform.scale")
    shrinkAnimation.fromValue = 1
    shrinkAnimation.toValue = 0.1
    shrinkAnimation.duration = 1
    shrinkAnimation.repeatCount = Float.greatestFiniteMagnitude

    circle.add(fadeOut, forKey: nil)
    circle.add(shrinkAnimation, forKey: nil)


    let instanceCount = 20
    replicatorLayer.instanceCount = instanceCount
    replicatorLayer.instanceDelay = 1 / CFTimeInterval(instanceCount)

    let angle = -CGFloat.pi * 2 / CGFloat(instanceCount)
    replicatorLayer.instanceTransform = CATransform3DMakeRotation(angle, 0, 0, 1)
    rootLayer.addSublayer(replicatorLayer)
}
```

Swift Progress Indicator using CAReplicatorLayer using scale and opacity animation