Swift CAReplicatorLayer Sample Code

ddas.tech/swift-careplicatorlayer-sample-code/

November 20, 2022

In this post we will check the CAReplicatorLayer behavior. **CAReplicatorLayer** is essentially a layer that creates a specified number of copies of its sublayers, each copy potentially having geometric, temporal, and color transformations applied to it.

We can use a CAReplicatorLayer object to build complex layouts based on a single source layer that is replicated with transformation rules that can affect the position, rotation color, and time.

Table of Contents

Let's begin with creating the root layer and then we will add CAReplicatorLayer to it.

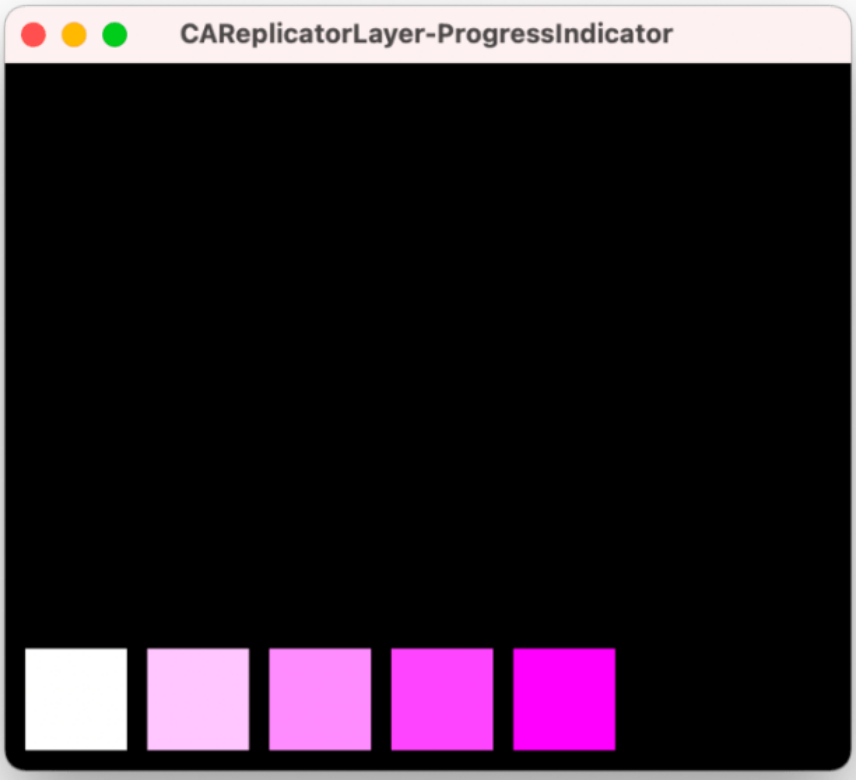
```
let rootLayer = CALayer()

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.replicatorLayerSampleOne()
//  self.replicatorLayerSampleTwo()
//  self.replicatorLayerSampleThree()
}
```

CAReplicatorLayer X Transformation and Single Color Offset

Lets create the first sample code

This creates a square of size 50 * 50 and replicates that 5 times and each instance has a color offset (in the example we are offsetting the green color).
Each of the instance is added to the replicator layer at a horizontal distance of 10

```
func replicatorLayerSampleOne(){
   let replicatorLayer = CAReplicatorLayer()
   let coloredSquare = CALayer()
   coloredSquare.backgroundColor = NSColor.white.cgColor
   coloredSquare.frame = CGRect(x: 10, y: 10, width: 50, height: 50)

   let instanceCount = 5
   replicatorLayer.instanceCount = instanceCount
   replicatorLayer.instanceTransform = CATransform3DMakeTranslation(60, 0, 0)

   replicatorLayer.instanceGreenOffset = -0.2
   replicatorLayer.addSublayer(coloredSquare)
   rootLayer.addSublayer(replicatorLayer)

}
```

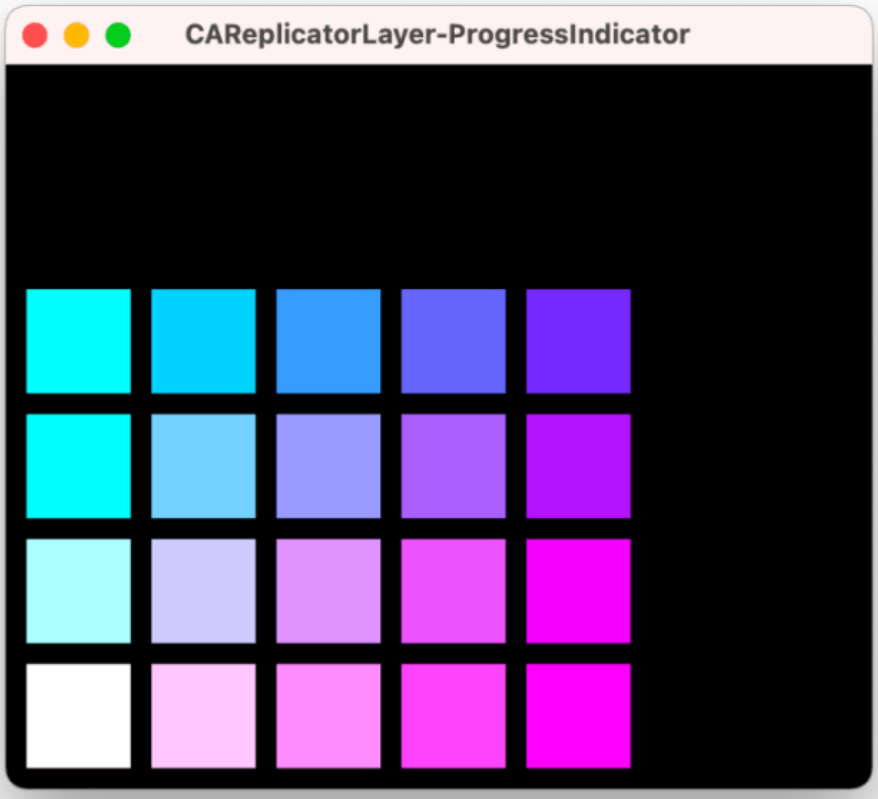


CAReplicatorLayer Sample with X transformation and color offset

CAReplicatorLayer X, Y Transformation and Two Color Offset

Next we will do **Nesting replicator Layers in X and Y Axis**

```
func replicatorLayerSampleTwo(){
    let replicatorLayer = CAReplicatorLayer()
    let rowLayer = CAReplicatorLayer()

    replicatorLayer.instanceCount = 4
    rowLayer.instanceCount = 5

    let coloredSquare = CALayer()
    coloredSquare.backgroundColor = NSColor.white.cgColor
    coloredSquare.frame = CGRect(x: 10, y: 10, width: 50, height: 50)

    rowLayer.instanceTransform = CATransform3DMakeTranslation(60, 0, 0)
    rowLayer.addSublayer(coloredSquare)
    rowLayer.instanceGreenOffset = -0.2

    replicatorLayer.instanceTransform = CATransform3DMakeTranslation(0, 60, 0)
    replicatorLayer.instanceRedOffset = -0.2
    replicatorLayer.addSublayer(rowLayer)

    rootLayer.addSublayer(replicatorLayer)

}
```

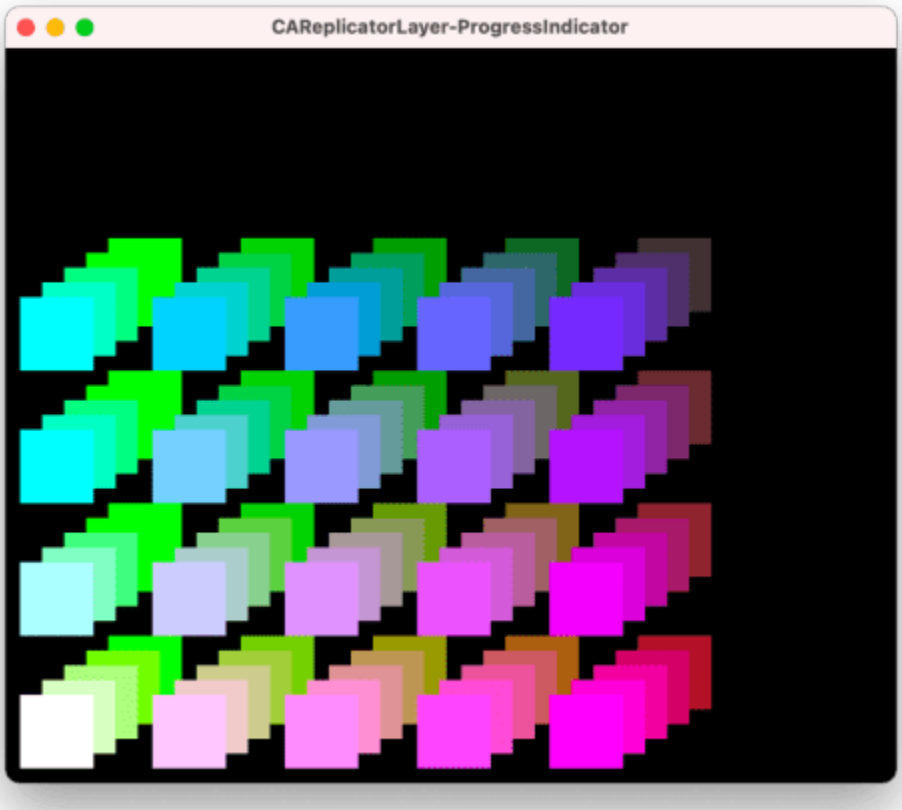


CAReplicatorLayer Sample with X and Y transformation and color offset

CAReplicatorLayer X, Y and Z Transformation and Multi Color Offset

Nesting replicator Layers in X, Y and Z axis

```
func replicatorLayerSampleThree(){
    let replicatorLayer = CAReplicatorLayer()
    let rowLayer = CAReplicatorLayer()
    let zLayer = CAReplicatorLayer()

    replicatorLayer.instanceCount = 4
    rowLayer.instanceCount = 5
    zLayer.instanceCount = 5

    let coloredSquare = CALayer()
    coloredSquare.backgroundColor = NSColor.white.cgColor
    coloredSquare.frame = CGRect(x: 10, y: 10, width: 50, height: 50)

    zLayer.instanceTransform = CATransform3DMakeTranslation(15, 10, -30)
    zLayer.instanceBlueOffset = -0.2
    zLayer.instanceRedOffset = -0.1

    zLayer.addSublayer(coloredSquare)

    rowLayer.instanceTransform = CATransform3DMakeTranslation(90, 0, 0)
    rowLayer.addSublayer(zLayer)
    rowLayer.instanceGreenOffset = -0.2

    replicatorLayer.instanceTransform = CATransform3DMakeTranslation(0, 90, 0)
    replicatorLayer.instanceRedOffset = -0.2
    replicatorLayer.addSublayer(rowLayer)

    rootLayer.addSublayer(replicatorLayer)

}
```



CAReplicatorLayer Sample with X, Y and Z transformation and multi color offset

Now that we know how the CAReplicatorLayer works time to apply it to create some progress indicators in the next post Progress Indicator using CAReplicatorLayer