```
// Draw Paths Seperately          more effecint, use GPU, not CPU
import UIKit                      CAShapeLaye is a CALayer

class ViewController: UIViewController{
    let squareShapeLayer: CAShapeLayer = CAShapeLayer()
    let triangleShapelayer: CAShapeLayer = CAShapeLayer()



    override func viewDidLoad(){
        super.viewDidLoad()
        createSquareShape()
        createTriangleShape()
    }

    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        squareShapeLayer.path = UIBezierPath(rect: (x:50, y:50, width:50, height:100)).cgPath
        let trianglePath = UIBezierPath()
        trianglePath.move(to: CGPoint(x:50, y:300))
        trianglePath.addLine(to: CGPoint(x:150, y:300))
        trianglePath.addLine(to: CGPoint(x:100, y:200))
        trianglePath.close()
        triangleShapeLayer.path = trianglePath.cgPath
    }

    func createSquareShape(){
        squareShapeLayer.fillColor = UIColor.red.cgColor
        squareShapelayer.lineWidth = 10.0
        squareShapeLayer.strokeColor = UIColor.black.cgColor
        //squareShapeLayer.lineDashPattern = [4,3]
        //squareShapeLayer.lineDashPhase = 10
        squareShapeLayer.lineJoin = CAShapeLayerLineJoin.round
        squareShapeLayer.strokeEnd = 0.7
        squareShapeLayer.lineCap = CAShapeLayerLineCap.round

        view.layer.addSublayer(squareShapeLayer)
    }

    func createTriangleShape(){
        triangleShapeLayer.fillColor = UIColor.green.cgColor
        view.layer.addSublayer(triangleShapeLayer)
    }
```

```
// Draw Paths as Combined Path
import UIKit

class ViewController: UIViewController{
    let squareShapeLayer: CAShapeLayer = CAShapeLayer()
    let triangleShapeLayer: CAShapeLayer = CAShapeLayer()


    override func viewDidLoad(){
        super.viewDidLoad()
        // createSquareShape()
        // createTriangleShape()
        combinePath()
    }


    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        squareShapeLayer.path = UIBezierPath(rect: (x:50, y:50, width:50, height:100)).cgPath
        let trianglePath = UIBezierPath()
        trianglePath.move(to: CGPoint(x:50, y:300))
        trianglePath.addLine(to: CGPoint(x:50, y:300))
        trianglePath.addLine(to: CGPoint(x:150, y:300))
        trianglePath.addLine(to: CGPoint(x:100, y:200))
        trianglePath.close()
        triangleShapeLayer.path = trianglePath.cgPath
    }

    func combinePaths(){
        let combinePath = UIBeizerPath()
        let squarePath = UIBezierPath(rect: (x:50, y:50, width:50, height:100)).cgPath
        combinePath.append(squarePath)
        let trianglePath = UIBezierPath()
        trianglePath.move(to: CGPoint(x:50, y:300))
        trianglePath.addLine(to: CGPoint(x:150, y:300))
        trianglePath.addLine(to: CGPoint(x:100, y:120))
        trianglePath.close()
        combinePath.append(trianglePath)
        let combineShape = CAShapeLayer()
        combineShape.lineWidth = 3.0
        combineShape.strokeColor = UIColor.red.cgColor
        combineShape.fillColor = nil
        combineShape.path = combinePath.cgPath
        view.layer.addSublayer(combineShape)
    }
}
```

Masks ⇒ similar to clips in Core graphics

```
import UIKit

class ShapeView : UIView{                                   CALayer = CAShapeLayer

    override func didMoveToSuperView(){
        super.didMoveToSuperView()
        let lineWidth : CGFloat = 30.0
        let shape = CAShapeLayer()
        shape.path = UIBezierPath(ovalIn : CGRect(x: lineWidth/2, y: lineWidth/2, width: bounds.width - lineWidth,
            height: bounds.height - lineWidth)).cgPath
        shape.frame = bounds
        shape.fillColor = nil
        shape.lineWidth = lineWidth
        shape.strokeColor = UIColor.black.cgColor
       // layer.addSublayer(shape)
        let gradientLayer = CAGradientLayer()
        gradientLayer.frame = bounds
        gradientLayer.colors = [UIColor.red.cgColor, UIColor.orange.cgColor]
       gradientLayer.mask = shape
       layer.addSublayer(gradientLayer)
    }
```

Transformations

```
import UIKit

class ViewController : UIViewController{

    override func viewDidLoad(){
        super.viewDidLoad()
        createShape()

    }

    override func viewDidAppear(_ animated : Bool){
        super.viewDidAppear(animated)
        createSquare()
    }
```

```
func createSquare() {
    let squarePath : CGPath = UIBezierPath(rect: CGRect(x:0, y:0, width:200, height:200)).cgPath
    let squareShape = CAShapeLayer()
    squareShape.path = squarePath
    squareShape.frame = CGRect(x: view.frame.midX-100, y: view.frame.midY-100, width:200,
        height:200)
    squareShape.fillColor = UIColor.orange.cgColor
    let angle = 45 * CGFloat(Double.pi)/180
    //squareShape.transform = CATransform3DMakeRotation(angle, 0, 0, 1.0)   anchor point is in the center if has a frame
    //squareshape.transform = CATransform3DMakeTranslation(50, 50, 0)
    squareshape.transform = CATransform3DMakeScale(0.5, 0.5, 1)

    view.layer.addSublayer(squareShape)
}
```

Ring Graph

```
// viewcontroller.swift
import UIKit

class ViewController : UIViewController{
    @IBOutlet weak var ringGraph : RingGraphView!
    @IBOutlet weak var slider : UISlider!


    override func viewDidLoad(){
        super.viewDidLoad()
    }

    override func viewDidAppear(_ animated: Boo()
        super.viewDidLoad(animated)
        ringGraph.progress = CGFloat(slider.value)
}

@IBAction func sliderDidChange(_ sender: UISlider){
        ringGraph.progress = CGFloat(sender.value)
}
```

```
//RingGraphView.swift

import UIKit

class RingGraphView : UIView{

    var progress : CGFloat = CGFloat(0.5){
        didSet{
            if progress >= 1.0{
                progress = 1.0
            }
            else if progress <= 0.0{
                progress = 0.0
            }
            ringLayer.strokeEnd = progress
        }
    }
```

```
let backdropLayer : CAShapeLayer = CAShapeLayer()
let ringWidth: CGFloat = CGFloat(40.0)
var halfRingWidth : CGFloat{
    return ringWidth/2
}
let ringLayer : CAShapeLayer = CAShapeLayer()


override func didMoveToSuperView(){
    super.didMoveToSuperView()
    let circlePath : CGPath = UIBezierPath(ovalIn: CGRect(x: halfRingWidth, y: halfRingWidth,
        width: bounds.width - ringWidth, height: bounds.height - ringWidth)).cgPath
    let offset = CGFloat(90*Double.pi/180)
    backdropLayer.path = circlePath
    backdropLayer.lineWidth = ringWidth
    backdropLayer.strokeEnd = 1.0
    bakdropLayer.fillColor = nil
    backdropLayer.strokeColor = UIColor(red: 112/255, green: 25/255, blue: 88/255, alpha: 1.0).cgColor
    layer.addSublayer(backdropLayer)

    ringLayer.path = circlePath
    ringLayer.frame = CGRect(x:0, y:0, width: bounds.width, height: bounds.height)
    ringLayer.lineWidth = ringWidth
//  ringLayer.strokeColor = UIColor(red: 231/255, green: 51/255, blue 38/255, alpha: 1.0).cgColor
```

```
        // ringLayer.fillColor = nil
        ringLayer.strokeColor = UIColor.black.cgColor

        ringLayer.fillColor = nil
        ringLayer.strokeEnd = 0.5
        ringLayer.lineCap = CAShapeLayerLineCap.round
        ringLayer.transform = CATransform3DMakeRotation(offset, 0, 0, -1.0)
      // layer.addSublayer(ringLayer)
        let gradientLayer = CAGradientLayer()
        gradientLayer.colors = [UIColor(red: 231/255, green: 51/255, blue: 38/255, alpha: 1.0).cgColor,
                UIColor(red: 231/255, green: 51/255, blue: 38/255, alpha: 1.0).cgColor]
        gradientLayer.frame = bounds
        gradientLayer.locations = [0.0, 1.0]
        gradientLayer.mask = ringLayer
        layer.addSublayer(gradientLayer)

    }
}


//Marching Ants
import UIKit                CABasicAnimation

class ViewController: UIViewController{
    override func viewDidLoad() {
        super.viewDidLoad()

    }

    override func viewDidAppear(_ animated: Bool)
        super.viewDidAppear(animated)
        createAntsLoading()
    }

    func createAntsLoading(){
        let antsLoader = CAShapeLayer()
        let lineWidth: CGFloat = 6.0
        antsLoader.lineWidth = lineWidth
        antsLoader.strokeColor = UIColor.purple.cgColor
        antsLoader.path = UIBezierPath(ovalIn: CGRect(x: lineWidth/2, y: lineWidth/2, width: 200 - lineWidth,
            height: 200 - lineWidth)).cgPath
        antsLoader.frame = CGRect(x: view.frame.midX-100, y: view.frame.midY-100, width: 200, height: 200)
        let radius = (200 - lineWidth)/2
        let circumference = 2* CGFloat(Double.pi) *radius
        let phaseSize = circumference/200          // we want 200 ants
        antsLoader.lineDashPattern = [NSNumber(value: Float(phaseSize))]
        antsLoader.fillColor = nil
        view.layer.addSublayer(antsLoader)
```

```
let rotationalAnimation = CABasicAnimation(keyPath: "transform")
rotationalAnimation.byValue = CATransform3DMakeRotation(CGFloat(Double.pi), 0, 0, 1.0)
rotationalAnimation.duration = 10.0
rotationalAnimation.repeatCount = .greatestFiniteMagnitude
rotationalAnimation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)
cmtsLoader.add(rotationalAnimation, forKey: nil)
 let fadeInandOutAnimation = CABasicAnimation(keyPath: "opacity")
 fadeInandOutAnimation.fromValue = 1.0
 fadeInandOutAnimation.toValue = 0.0
 fadeInandOutAnimation.duration = 0.75
 fadeInandOutAnimation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)
 fadeInandOutAnimation.repeatCount = .greatestFiniteMagnitude
 fadeInandOutAnimation.autoreverses = true
 loadingText.layer.add(fadeInAndOutAnimation, forKey: nil)
}
```