Context and State

↑ where all cg begins

```
//PracticeView.swift
import UIKit

@IBDesignable
class PracticeView: UIView {
    override func draw(_ rect: CGRect){
        let ctx = UIGraphicsGetCurrentContext()!
        ctx.setFillColor(UIColor.orange.cgColor)
        ctx.setStrokeColor(UIColor.blue.cgColor)
        ctx.setLineWidth(6.5)
        /*
        ctx.move(to: CGPoint(x:0, y:0))
        ctx.addLine(to: CGPoint(x:180, y:0))
        ctx.addLine(to: CGPoint(x:180, y:90))
        ctx.addLine(to: CGPoint(x:0, y:90))
        ctx.closePath()
        ctx.fillPath()
        */

        /*
        let rectPath = CGMutablePath()
        rectPath.move(to: CGPoint(x:2, y:2))
        rectPath.addLine(to: CGPoint(x:180, y:2))
        rectPath.addLine(to: CGPoint(x:180, y:90))
        rectPath.addLine(to: CGPoint(x:2, y:90))
        rectPath.closeSubpath()
        ctx.addPath(rectPath)
        ctx.strokePath()
        */
```

ctx = // fill color
// stroke color

↗ shift every point .25 to fit the line width

```
        /*
        let rectPath = UIBezierPath()
        rectPath.move(to: CGPoint(x:2, y:2))
        rectPath.addLine(to: CGPoint(x:180, y:2))
        rectPath.addLine(to: CGPoint(x:180, y:90))
        rectPath.addLine(to: CGPoint(x:2, y:90))
        rectPath.fill()
        */
        /*
        let rectPath = UIBezierPath(rect: CGRect(x:2, y:2, width:100, height:90))
        rectPath.fill
        let center = CGPoint(x: bounds.width/2, y: bounds.height/2)
        let startAngleInDegrees: CGFloat = 135 * CGFloat(Double.pi2)/180
        let endAngleInDegrees: CGFloat = 45 * CGFloat(Double.pi)/180
        let arcRadius: CGFloat = bounds.width/2
        let arcPath = UIBezierPath(arcCenter: center, radius: arcRadius - 25, startAngle: startAngleInDegrees,
            endAngle: endAngleInDegrees, clockwise: true)
        arcPath.lineWidth = 50
        UIColor.orange.setStroke()
        arcPath.stroke()
```

```
*/
/*
let trianglePath = UIBezierPath()
trianglePath.move(to: CGPoint(x:2, y:90))
tringlePath.addLine(to: CGPoint(x:100, y:90))
tringlePath.addLine(to: CGPoint(x:50, y:10))
trianglePath.close()
UIColor.blue.setFill()
trianglePath.fill()
let ovalPath = UIBezierPath(ovalIn: CGRect(x: 2 , y:2, width:100, height:100))
UIColor.green.setFill
ovalPath.fill()
*/
/*
let context = UIGraphicsGetCurrentContext()!
context.saveGState()
let ovalPath = UIBezierPath(ovalIn: CGRect(x:50, y:50, width:200 height:200))
ovalPath.addClip()
let gradientStartColor = UIColor.yellow.cgColor
let gradientEndColor = UIColor.red.cgColor
let colors = [gradientStartColor, gradientEndColor]
let colorLocations:[CGFloat] = [0.0, 1.0]
let colorSpace = CGColorSpaceCreateDeviceRGB()          Core Foundation (C related)
let gradient = CGGradient(colorsSpace: colorSpace, colors: colors as CFArray, locations: colorLocations)!
let startPoint = CGPoint.zero
let endPoint = CGPoint(x:0.0, y: bounds.height)
context.drawLinearGradient(gradient, start: startPoint, end: endPoint, options: [])
context.restoreGState()


let secondOval = UIBezierPath(ovalIn: CGRect(x:0, y:0, width: bounds.width, height: bounds.height))
UIColor.green.setFill()
secondOval.fill()                    current transform matrix
                                     ↑
                                user space    device space
*/
/*
let context = UIGraphicsGetCurrentContext()!
let shadowColor = UIColor.black.withAlphaComponent(0.7).cgColor
let shadowOffset = CGSize(width: 2.0, height: 2.0)
let shadowBlurRadius: CGFloat = 5.0
context.setShadow(offset: shadowOffset, blur: shadowBlurRadius, color: shadowColor)
let oval = UIBezierPath(ovalIn: CGRect(x:50, y:50, width: 100, height: 100))
UIColor.green.setFill()
oval.fill()
*/
let context = UIGraphicsGetCurrentContext()!
let rotationAngle: CGFloat = 45 * Pi / 180
let squareWidth: CGFloat = 80.0
let viewWidth: CGFloat = bounds.width
let viewHeight: CGFloat = bounds.height
// context.translateBy(x: viewWidth/2, y: viewHeight/2)
```

```
// context.rotate(by: rotationAngle)
// context.scaleBy(x: 1.2, y: 1.2)
var transform = CGAffineTransform(translationX: viewWidth/2, y: viewHeight/2)
transform = transform.rotated(by: rotationAngle)
transform = transform.scaledBy(x: 1.2, y: 1.2)

// context.concatenate(transform)
let squarePath = UIBezierPath(rect: CGRect(x: -squareWidth/2, y: -squareWidth/2, width: squareWidth,
    height: squareWidth))
squarePath.apply(transform)
UIColor.green.setFill()
squarePath.fill() // transform the square Path before fill
```

Bar Graph

```
// BarGraphView.swift

import UIKit
@Designable
class: BarGraphView: UIView(){

    struct BarGraphData{
        var label: String
        var value: Int
    }
    let barGraphData: [BarGraphData] = [
        BarGraphData(Label: "Mon", value: 10),
        ↓ similar ones

    }
    let barWidth: CGFloat = 20.0



    overide func draw(_ rect: CGRect){
        let margin: CGFloat = 0.1 * bounds.width
        let graphMidPoint: Int = Int((bounds.height - (2*margin))/2)
        let linePath = UIBezierPath()
        linePath.move(to: CGPoint(x:0, y: margin))
        linePath.addLine(to: CGPoint(x: bounds.width, y: margin))
        linePath.move(to: CGPoint(x:0, y: CGFloat(graphMidPoint) + margin))
        linePath.addLine(to: CGPoint(x: bounds.width, y: CGFloat(graphMidPoint) + margin))
        linePath.move(to: CGPoint(x:0, bounds.height - margin))
        linePath.addLine(to: CGPoint(x: bounds.width, y: bounds.height - margin))
        UIColor.black.setStroke()
        linePath.lineWidth = 0.5
        linePath.stroke()
        let font = UIFont(name: "Avenir-Medium", size: 14)!
```

```
let labelAttributes = [NSAttributedStringKey.font: font,
                       NSAttributedStringKey.foregroundColor: UIColor.black]
let highestDataValue = barGraphData.max{$0.value < $1.value}!.value
let verticalTopLabel = String(highestDataValue) as NSString
verticalTopLabel.draw(at: CGPoint(x:0, y: margin), withAttributes: labelAttributes)
let midDataValue = Int(highestDataValue/2)
let verticalMidLabel = String(midDataValue) as NSString
verticalMidLevel.draw(at: CGPoint(x:0, y: CGFloat(graphMidPoint) + margin, withAttributes: labelAttributes)
let verticalBottomLabel = String(0) as NSString
verticalBottomLabel.draw(at: CGPoint(x:0, y: bounds.height - margin), withAttributes: labelAttributes)
    let numberOfBars:Int = barGraphData.count
    let numberOfGaps:Int = numberOfBars - 1
    let barGap: CGFloat = ((bounds.width - (barWidth * CGFloat(numberOfBars))) - (2 * margin)) / numberOfGaps
    for i in 0...barGraphData.count-1{
        let xPos = CGFloat(i) * (barWidth + barGap) + margin
        let barHeight:CGFloat = CGFloat(barGraphData[i].value)/CGFloat(highestDataValue) * (bounds.height - margin * 2)
        let yPos = bounds.height - barHeight - margin
        let bar = UIBezierPath(rect: CGRect(x: xPos, y: yPos, width: barWidth, height: barHeight))
        let barColor = UIColor(red: randomColorValue(), green: randomColorValue(), blue: randomColorValue(), alpha: 1.0)
        barColor.setFill()
        bar.fill
        let label = barGraphData[i].label as NSString
        let textPos = CGPoint(x: xPos, y: bounds.height - margin)
        label.draw(at: textPos, withAttributes: labelAttributes)
    }
}

func randomColorValue() -> CGFloat{
    return CGFloat(arc4random())/CGFloat(UInt32.max)
}
```