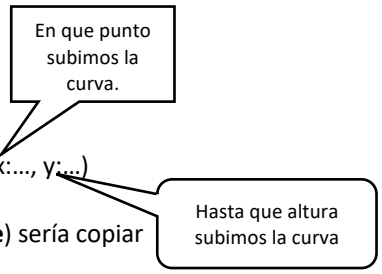
GRÁFICOS EN SWIFTUI

EL PATH

- Un path se crea a través de los puntos que vamos uniendo, desde un punto inicial… hasta uno final de cierre.
  Path(){…}
  - Punto de inicio -> **path.move**
  - Seguir dibujando -> **path.addline**
  - Cerrar el camino si no se ha puesto la última línea -> **path.closeSubPath**
  - Pintarlo de un color = .**fill**(Color.blue)
  - En vez de llenar el interior crear un bore = .**stroke**(Color: …, lineWidht: Int)
  - Para hacer una curva = **path.addQuadCurve**(to:CGPoint(x:…, y:…), control: CGPoint(x:…, y:…)



- Se puede usar **Zstack** para combinar o superponer más de 1 figura.
- Parece que no son compatibles y la única forma de hacer las dos cosas (**el .fill y el .stroke**) sería copiar 2 veces el mismo código y superponerlos embebiendo ambos en una **Zstack**.

ESTRUCTURAS CIRCULARES O ARCOS (TARTAS)

- SwiftUI tien preparada una **api** para que los desarrolladores dibujen arcos Path() { path in ….} (ver ejemplos abajo)
- Para hacer una **tarta** embebemos en un Zstack varios path y vamos jugando con distintos ángulos.
  - Si queremos dar importancia y separar un **quesito** en concreto le aplicamos un **offset**() después del .**fill**()
  - Luego, como ya estamos en un **Zstack**, podemos copiar de nuevo el **path** debajo y aplicarle un .**stroke**()
  - Y en el **path** que hicimos el .**stroke**(), como ese no está relleno por dentro podemos aplicarle un **texto** haciendo un .**overlay**()
  - En este ejemplo, al haber muchos **path** y casi todos iguales es recomendable documentarlos para que cuando vuelvas dentro de un tiempo a ver el código te sea más fácil recordarlo.
- Podemos hacer un **rectángulo** sin tener que delimitar todos los puntos así:
  pathAddRect(CGRect(x:30, y:130, width: 200, height:120) -> (Punto de inicio, anchura y altura)

PROTOCOLO SHAPE:

Podemos estandarizar una figura que hayamos creado para luego usarla en más sitios y adaptarla simplemente con el parámetro de entrada (en este caso un rectángulo **CGRect**) creando una estructura.
Para hacer el círculo más personalizado se puede declarar en la **struct** una variable (porcentaje del círculo) de tipo **CGFloat** de 1.0 por ejemplo y luego poner los puntos de partida y final en función de esa variable.
Ver ejemplo en código más abajo.
Por último comentar que se pueden usar también las shapes (formas) que SwiftUI ya tiene predefinidos, como el círculo, cuadrado, rectángulo….

```
struct ContentView: View {
   var body: some View {
      VStack {
         HStack{
            ZStack{
            Path(){ path in
               path.move(to: CGPoint(x: 30, y: 130))
               path.addLine(to: CGPoint(x:130, y: 130))
               path.addLine(to: CGPoint(x:130, y: 150))
               path.addLine(to: CGPoint(x:50, y: 150))
               path.addLine(to: CGPoint(x:50, y: 170))
               path.addLine(to: CGPoint(x:100, y: 170))
               path.addLine(to: CGPoint(x:100, y: 190))
               path.addLine(to: CGPoint(x:50, y: 190))
               path.addLine(to: CGPoint(x:50, y: 250))
               path.addLine(to: CGPoint(x:30, y: 250))
               //path.addLine(to: CGPoint(x:30, y: 30))
               path.closeSubpath()
            }
            .stroke(Color.blue, lineWidth: 10 )

            Path(){ path in
               path.move(to: CGPoint(x: 30, y: 130))
               path.addLine(to: CGPoint(x:130, y: 130))
               path.addLine(to: CGPoint(x:130, y: 150))
               path.addLine(to: CGPoint(x:50, y: 150))
```

```
            path.addLine(to: CGPoint(x:50, y: 170))
            path.addLine(to: CGPoint(x:100, y: 170))
            path.addLine(to: CGPoint(x:100, y: 190))
            path.addLine(to: CGPoint(x:50, y: 190))
            path.addLine(to: CGPoint(x:50, y: 250))
            path.addLine(to: CGPoint(x:30, y: 250))
            //path.addLine(to: CGPoint(x:30, y: 30))
            path.closeSubpath()
            }
            .fill(Color.orange)
        }
          Path(){ path in
            path.move(to: CGPoint(x: 30, y: 130))
            path.addQuadCurve(to: CGPoint(x: 130, y: 130), control: CGPoint(x: 80, y: 70))

            path.addLine(to: CGPoint(x:130, y: 130))
            path.addLine(to: CGPoint(x:130, y: 150))
            path.addQuadCurve(to: CGPoint(x: 50, y: 150), control: CGPoint(x: 85, y: 110))
            path.addLine(to: CGPoint(x:50, y: 150))
            path.addQuadCurve(to: CGPoint(x: 50, y: 230), control: CGPoint(x: 20, y: 190))
            path.addLine(to: CGPoint(x:50, y: 230))
            path.addQuadCurve(to: CGPoint(x: 110, y: 230), control: CGPoint(x: 85, y: 270))

            path.addLine(to: CGPoint(x:110, y: 230))
            path.addLine(to: CGPoint(x:110, y: 200))
            path.addLine(to: CGPoint(x:80, y: 200))
            path.addLine(to: CGPoint(x:80, y: 180))
            path.addLine(to: CGPoint(x:130, y: 180))
            //path.addQuadCurve(to: CGPoint(x: 30, y: 200), control: CGPoint(x: 80, y: 300))
            path.addLine(to: CGPoint(x:130, y: 250))
            path.addQuadCurve(to: CGPoint(x: 30, y: 250), control: CGPoint(x: 80, y: 310))
            path.addLine(to: CGPoint(x:30, y: 250))
            //path.addQuadCurve(to: CGPoint(x: 30, y: 250), control: CGPoint(x: 80, y: 300))

            path.closeSubpath()
            path.addQuadCurve(to: CGPoint(x: 30, y: 250), control: CGPoint(x: -10, y: 190))

          }
          //.fill(Color.orange)
          .stroke(Color.blue, lineWidth: 5)
        }

        /*Path(){ path in
          path.move(to: CGPoint(x: 60, y: 130))
          path.addQuadCurve(to: CGPoint(x: 200, y: 130), control: CGPoint(x: 130, y: 60))
          path.addRect(CGRect(x: 30, y: 130, width: 200, height: 120))
        }*/
        Text("Prueba")
          .font(.system(.title, design: .rounded))
          .fontWeight(.bold)
          .frame(width: 250, height: 150)
          .background(FunnyLabel(perCircle: 0.8).fill(Color.orange))

      }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
struct FunnyLabel: Shape {
    var perCircle = 1.0

    func path(in rect: CGRect) -> Path {
        var path =  Path()

        path.move(to: CGPoint(x: rect.size.width * (1 - perCircle)/2, y: 0))
```

```
        path.addQuadCurve(to: CGPoint(x: (rect.size.width * (1 - (1 - perCircle)/2)), y: 0), control: CGPoint(x: rect.size.width/2, y: rect.size.width*(-
0.5)))
        path.addRect(CGRect(x: 0, y: 0, width: rect.size.width, height: rect.size.height))

        return path
    }
}
```

TARTA

```
struct pieView: View {
    var body: some View {
        ZStack {
            // Quesito superior
            Path(){path in

                path.move(to: CGPoint(x: 200, y: 250))
                path.addArc(center: CGPoint(x: 200, y: 250), radius: 150, startAngle: Angle(degrees: 0), endAngle: Angle(degrees: 180), clockwise:
true)
            }
            .fill(Color.orange)
            // Quesito gris izquierdo
            Path(){path in

                path.move(to: CGPoint(x: 200, y: 250))
                path.addArc(center: CGPoint(x: 200, y: 250), radius: 150, startAngle: Angle(degrees: 180), endAngle: Angle(degrees: 140), clockwise:
true)
            }
            .fill(Color(UIColor.systemGray))
            // Quesito inferior destacado
            Path(){path in

                path.move(to: CGPoint(x: 200, y: 250))
                path.addArc(center: CGPoint(x: 200, y: 250), radius: 150, startAngle: Angle(degrees: 140), endAngle: Angle(degrees: 60), clockwise:
true)
            }
            .fill(Color(UIColor.systemRed))
            .offset(x: -2 , y: 10 )

            Path(){path in

                path.move(to: CGPoint(x: 200, y: 250))
                path.addArc(center: CGPoint(x: 200, y: 250), radius: 150, startAngle: Angle(degrees: 140), endAngle: Angle(degrees: 60), clockwise:
true)
                path.closeSubpath()
            }
            .stroke(Color.blue,lineWidth: 5)
            .offset(x: -2 , y: 10 )
            .overlay((Text("22 %"))
                    .font(.system(.title, design:.rounded))
                    .bold()
                    .offset(x: -10, y:  -20)


            )
            // Quesito verde derecho
            Path(){path in

                path.move(to: CGPoint(x: 200, y: 250))
                path.addArc(center: CGPoint(x: 200, y: 250), radius: 150, startAngle: Angle(degrees: 60), endAngle: Angle(degrees: 0), clockwise: true)
            }
            .fill(Color(UIColor.systemGreen))
        }
    }
}

struct pieView_Previews: PreviewProvider {
    static var previews: some View {
        pieView()
```

```
    }
}
```

INDICADOR DE PROGRESO (ESTILO FITBIT)

```
struct progressIndicator: View {

    private var trackGradient = LinearGradient(gradient: Gradient(colors: [Color(red: 50/255, green: 150/255, blue: 230/255), Color(red: 70/255, green: 180/255, blue: 240/255)]), startPoint: .trailing, endPoint: .leading)

    var body: some View {
        ZStack{
            Circle()
                .stroke(Color(.systemGray5), lineWidth: 10)
                .frame(width:200, height: 200)

            Circle()
                .trim(from: 0, to: 0.75)
                .stroke(trackGradient, lineWidth: 20)
                .frame(width:200, height: 200)
                .overlay(VStack{
                    Text("65 %")
                        .font(.system(size: 50, design: .rounded)).fontWeight(.bold)
                    Text("Número de pasos")
                        .font(.system(.body, design: .rounded))
                        .bold().foregroundColor(Color(.systemTeal))
                })
        }
    }
}
```

DIAGRAMA EN FORMA DE DONUT

```
struct DonutView: View {
    var body: some View {
        ZStack{
            Circle()
                .trim(from: 0.0, to: 0.3)
                .stroke(Color.brown, lineWidth: 70)

                .overlay(Text("30%")
                            .font(.title)
                            .fontWeight(.bold)
                            .foregroundColor(.white)
                            .offset(x: 87, y: 90)
                        )

            Circle()
                .trim(from: 0.3, to: 0.4)
                .stroke(Color.red, lineWidth: 60)

            Circle()
                .trim(from: 0.4, to: 0.8)
                .stroke(Color.yellow, lineWidth: 60)

            Circle()
                .trim(from: 0.8, to: 1.0)
                .stroke(Color.orange, lineWidth: 60)


        }.frame(width: 250, height: 250)
    }
}
struct DonutView_Previews: PreviewProvider {
    static var previews: some View {
        DonutView()
    }
}
```