TRANSICIONES

Vamos a crear una tarjeta para hacer traducciones y transicionar de un idioma a otro.
Una vez que tenemos una **transición** y la **combinamos** con otra incluso con otras dos como en nuestro ejemplo (**Offset**, **Scale** y **Opacity**), si prevemos que podemos usar esta transición en más de una ocasión lo mejor nos es hacer una **extensión** de la clase **AnyTransition** creado una propiedad estática (static var) y solo tendremos que llamarla cuando nos haga falta. Además lo bueno de las extensiones es que para llamarla al poner .**transition** y abrir paréntesis y poner un punto ya nos va a aparecer la extensión con el nombre que le hemos dado simplemente para seleccionarla, en nuestro ejemplo **(.transition(.offsetSacaledOpacityOut)**, esto es así porque hemos creado la propiedad **static**, con lo cual ésta pasa a ser una propiedad de la clase **AnyTransition** y las **subclases** no podrán cambiarla.

```
struct ContentView: View {
   @State private var showTransition = false
      var body: some View {
         VStack{ // Vamos a crear en una vertical stack un par de tarjetas cuadradas.
            Rectangle()
               .frame(width: 250, height: 250)
               .foregroundColor(Color(.systemOrange))
               .cornerRadius(25)
               .overlay(
                  Text("Apple")
                     .font(.system(.largeTitle, design: .rounded).bold())
                     .foregroundColor(.white)
               )
            if showTransition { // hacemos este if para indicar que si showTransition es true aparezca la traducción
               Rectangle()
                  .frame(width: 250, height: 250)
                  .foregroundColor(Color(.systemGreen))
                  .cornerRadius(25)
                  .overlay(
                     Text("Manzana")
                        .font(.system(.largeTitle, design: .rounded).bold())
                        .foregroundColor(.white)
                  )
                  .transition(.asimmetricTransition) //aquí como hemos creado dos posibilidades abajo en la extensión podemos elegir la
que queramos.
            }
         }
         .onTapGesture {
            withAnimation(Animation.spring()){ //hacemos la animación explícita para que se vea más animada

            self.showTransition.toggle()
         }
      }
}
struct ContentView_Previews: PreviewProvider {
   static var previews: some View {
      ContentView()
   }
}
}
extension AnyTransition{

   static var offsetScaledOpacityOut: AnyTransition {
      AnyTransition.offset(x: 700, y: -0).combined(with: .scale).combined(with: .opacity) // con esta transición la segunda tarjeta
aparece y desaparece en horizontal hacia la derecha. Si le ponemos -700 lo haría hacia la izquierda. Luego la combinamos con un .scale
y además con una .opacity. Para que nos deje combinar mas de una 1 y no nos de error tenemos que llamar mejor al principio a la clase
AnyTransition.
   }
   static var asimmetricTransition: AnyTransition{
      .asymmetric(insertion: .scale(scale: 0, anchor: .bottom), removal: .offset(x: 700, y: 0)) //esto se trataría de una transición
asimétrica, es decir, podemos elegir una transición diferente para cuando aparece y para cuando ser retira, aunque en el ejemplo
parece que no ha funcionado muy bien.
   }
}
```