MOV02 - Laboratorio 6

Juan Pablo Arias Mora

Febrero 2021

1 Introducción

SwiftUI es un conjunto de herramientas de interfaz de usuario que nos permite diseñar aplicaciones de forma declarativa. Esa es una forma elegante de decir que le decimos a **SwiftUI** cómo queremos que se vea y funcione nuestra interfaz de usuario, y descubre cómo hacer que eso suceda cuando el usuario interactúa con ella. En esta ocasión nos enfocaremos en transiciones entre **Views**

Para referencias futuras el código fuente final del mismo esta disponible en:
https://github.com/pabloariasmora/MOV02-Cenfotec-Demo-Labs.git

1.1 Versiones

- MacOs 10.15.7 - Inglés
- Xcode 12.4
- Simulador iPhone 12 Pro Max

2 Instrucciones

"Mi trabajo no es ponérselo fácil a la gente. Mi trabajo es hacerlos mejores." Steve Jobs

2.1 Creación de aplicación base

Paso 1: Cree un nuevo proyecto en Xcode, utilizando el template App dentro de iOS, en la ventana de opciones para la configuración especifica de la nueva aplicación.

Product Name : swuitransitions
Organization Identifier : com.cenfotec.mov02.06
Interface : SwiftUI
Life Cycle : SwiftUI App
Languge : Swift
Use Core Data: Sin Seleccionar
Include Test: Sin Seleccionar

2.2 Creando una animación

Paso 1: Abrir el archivo **ContentView.swift**. (El código en el archivo debe ser similar al que se muestra a continuación)

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, world!")
            .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
```

```
    static var previews: some View {
        ContentView()
    }
}
```

Paso 2: Dentro de la propiedad body, cambie el valor **"Hello, World!"** por el código que se muestra a continuación.

```
VStack{
    Button ("Tap Me"){
        //do nothing
    }
    Rectangle()
        .fill(Color.purple)
        .frame(
            width: 200,
            height: 200,
            alignment: .center)
}
```

Paso 3: En el **canvas**, debe hacer **click** sobre la palabra **Resume** para poder ver desplegar el **preview** de la **View** (ver figura 1).

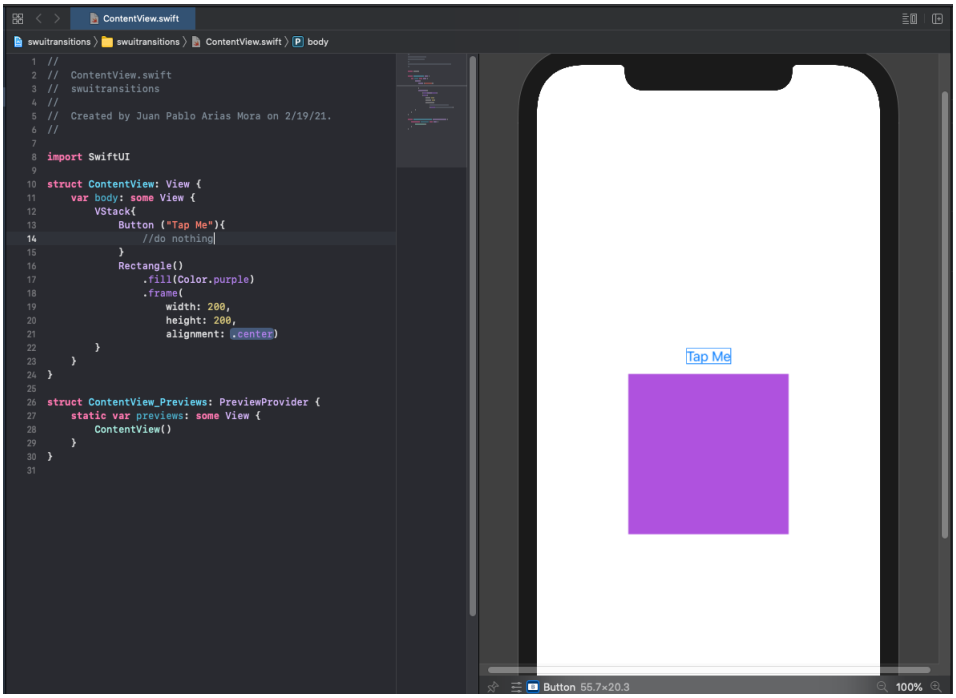

Figure 1: View Base

Paso 4: Vamos a agregar un estado del cual dependerá la visualización el cuadro morado en pantalla. Para esto agregamos justo antes de la definición del body el siguiente código.

```
@State private var isShowingSquare = false
```

Paso 5: Ahora usaremos el estado como condición para mostrar o no el cuadro morado. Esto se logra agregando un condicional antes de la definición del **Rectangle()** y encerrando entre el condicional la misma. (ver figura 2).

```
if isShowingSquare{
   //Código del rectangulo
}
```

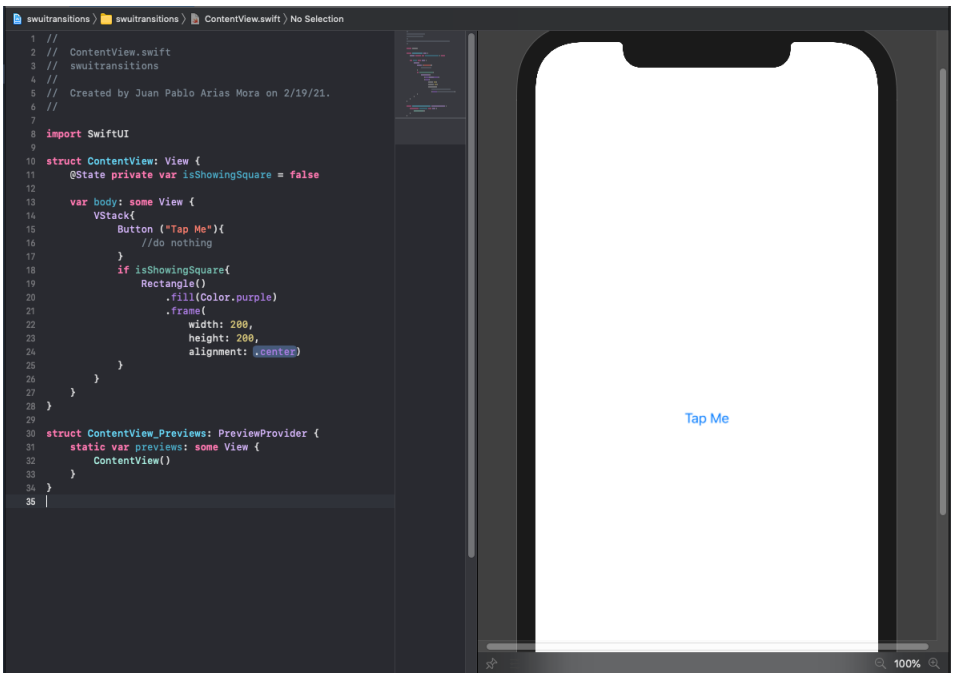

Figure 2: View Con Condicional

Paso 6: Compilar y ejecutar el código, una vez este corriendo si damos click sobre el botón **Tap Me**, notaremos que no interactúa con la variable **isShowingSquare**, esto debido a que en su acción tenemos solamente comentarios, por tanto debemos agregar el siguiente código remplazando la linea **//do nothing** (ver figura 3).

```
self.isShowingSquare.toggle()
```

Figure 3: Agregando Acciones a Botones

Paso 7: Compilar y ejecutar el código, una vez este corriendo si damos **Click** sobre el botón **Tap Me**, el resultado del mismo debe ser mostrar el cuadrado morado utilizando una aparición abrupta, debido a que no tiene ninguna animación. Y si volvemos a dar **Click** veremos como vuelve a desaparecer.

Paso 8: Agregar una animación de un **View** puede llegar a ser tan simple como agregar un **Wrapper State Change** alrededor de la acción de **toogle**. Como se muestra en el siguiente código: (ver figura 5)

```
withAnimation{
    self.isShowingSquare.toggle()
}
```

```swift
import SwiftUI

struct ContentView: View {
    @State private var isShowingSquare = false

    var body: some View {
        VStack{
            Button ("Tap Me"){
                withAnimation{
                    self.isShowingSquare.toggle()
                }
            }
            if isShowingSquare{
                Rectangle()
                    .fill(Color.purple)
                    .frame(
                        width: 200,
                        height: 200,
                        alignment: .center)
                    .transition(.scale)
            }
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Figure 4: withAnimation

Paso 9: Compilar y ejecutar el código, una vez este corriendo si damos **Click** sobre el botón **Tap Me**, el resultado del mismo debe ser mostrar nuevamente el cuadrado, pero esta vez con un efecto **Fade** y a su vez desplazando hacia arriba el botón **Tap Me** para dar espacio al cuadrado.

Paso 10: Todavía existe una mejoría que podemos realizar sobre el código, y es agregarle al cuadrado un **Transition Modifier**, esto se logra agregando al finalizar definición del **Rectangle()** el siguiente código:

```
.transition(.scale)
```

```swift
import SwiftUI

struct ContentView: View {
    @State private var isShowingSquare = false

    var body: some View {
        VStack{
            Button ("Tap Me"){
                withAnimation{
                    self.isShowingSquare.toggle()
                }
            }
            if isShowingSquare{
                Rectangle()
                    .fill(Color.purple)
                    .frame(
                        width: 200,
                        height: 200,
                        alignment: .center)
                    .transition(.scale)
            }
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Figure 5: Transiciones

Paso 11: Compilar y ejecutar el código, una vez este corriendo si damos **Click** sobre el botón **Tap Me**, el resultado debe verse mucho mejor, ya que el cuadrado entra al **View** de forma escalada.

Paso 12: Existen casos en los cuales como programadores la animación de entrada de un objeto debe ser diferente a la de salida, esto para mejorar la UX. Para poder genera ese efecto en el código debemos reemplazar la linea que contiene **.transition** por el siguiente código: (ver figura 6)

```
.transition(.asymmetric(insertion: .scale, removal: .opacity))
```

Paso 13: Compilar y ejecutar el código, una vez este corriendo si damos **Click** sobre el botón **Tap Me**, el resultado debe presentar en pantalla una animación de escalamiento al entrar y una de FadeOut al salir.

```swift
import SwiftUI

struct ContentView: View {
    @State private var isShowingSquare = false

    var body: some View {
        VStack{
            Button ("Tap Me"){
                withAnimation{
                    self.isShowingSquare.toggle()
                }
            }
            if isShowingSquare{
                Rectangle()
                    .fill(Color.purple)
                    .frame(
                        width: 200,
                        height: 200,
                        alignment: .center)
                    .transition(.asymmetric(insertion: .scale, removal: .opacity))
            }
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Figure 6: Transiciones asimétricas