

Les animations dans SwiftUI

Aurélien FILLION



Qu'est-ce qu'une animation ?

Une animation est une transition visuelle fluide entre deux états d'une interface utilisateur.

Plusieurs avantages :

- attirer l'oeil
- rendre l'interface plus dynamique

SwiftUI simplifie la création d'animations avec des outils intégrés pour animer les vues et leurs modifications d'état. Grâce à son approche déclarative, SwiftUI permet de définir des animations de manière concise et expressive, ce qui aide le développement d'interfaces utilisateur animées.

Animation implicite

Déclenchées automatiquement lors de la modification d'un état, sans spécification explicite de l'animation.

Utilisation du modificateur `.animation()` pour animer les changements d'état.

Dans cet exemple, la largeur du rectangle s'anime en douceur lors du changement de l'état `isExpanded`.

```
@State private var isExpanded = false

var body: some View {
    VStack {
        Rectangle()
            .frame(width: isExpanded ? 200 : 100, height: 100)
            .animation(.easeInOut, value: isExpanded)

        Button("Toggle") {
            isExpanded.toggle()
        }
    }
}
```

Animations explicites

Déclenchées intentionnellement par le développeur pour animer des changements spécifiques.

Utilisation de la fonction `withAnimation` pour encapsuler les modifications d'état à animer.

Chaque pression sur le bouton dans l'exemple, fait pivoter la vue de 45 degrés avec une animation de type ressort.

```
@State private var rotation: Double = 0

var body: some View {
    Button("Rotate") {
        withAnimation(.spring()) {
            rotation += 45
        }
    }
    .rotationEffect(.degrees(rotation))
}
```

Personnalisation ses Animations

On peut personnaliser nos animations comme on le souhaite

On peut ajouter des modificateurs comme `.delay()` pour retarder le début de l'animation ou `.repeatCount()` pour répéter l'animation un certain nombre de fois.

Il existe différents attributs qu'on peut rajouter pour customiser les animations par exemple :

`.linear` : Vitesse constante du début à la fin.

`.easeIn` : Démarrage lent puis accélération.

`.easeOut` : Démarrage rapide puis décélération.

`.easeInOut` : Combinaison de `easeIn` et `easeOut`.

`.spring` : Animation avec effet de ressort, offrant un mouvement plus naturel.

Cet exemple crée une animation qui commence après un délai d'une seconde, dure deux secondes, se répète trois fois et revient à son état initial à chaque répétition.

```
.animation(  
    Animation.easeInOut(duration: 2)  
        .delay(1)  
        .repeatCount(3, autoreverses: true),  
    value: scale  
)
```



Les transitions et Effets Visuels

Transitions :

- Permettent d'animer l'apparition et la disparition des vues.
- Utilisation du modificateur `.transition()` avec des effets prédéfinis comme `.slide`, `.opacity`, ou des transitions personnalisées.

Effets visuels :

- Combinaison de transformations comme la rotation, le redimensionnement et les changements d'opacité pour créer des animations.



Merci

Aurélien FILLION