LES ANIMATIONS SWIFTUI

Introduction

POURQUOI LES ANIMATIONS ?

- Ajoutent clarté et dynamisme à l'interface utilisateur.
- Simulent des interactions naturelles pour une meilleure expérience.

PHILOSOPHIE DE SWIFTUI

- Simplifier l'ajout d'animations tout en restant performant.
- Animation = combinaison de data animée et d'algorithme d'interpolation.

CONCEPTS CLÉS

- Animatable : Définit ce qui est animé.
- Animation : Définit comment les données changent dans le temps.
- Transitions : Gère le contexte des mises à jour.

```
VStack(alignment: .center) {
    Text(item.type.rawValue)
        .font(.system(size: 68))
        .padding()
        .background(
            Rectangle()
                .fill(item.rarity.color)
            .cornerRadius(20)
        ).shadow(color: (item.rarity.color).opacity(0.8), radius: shadowRadius ? 50 : 0, x: 0, y: 0)
        .rotation3DEffect(.degrees(degrees), axis: (x: 1, y: 1, z: 0))
        .scaleEffect(isAnimating ? 1.2 : 1.0)
        //.rotation3DEffect(.degrees(rotationX), axis: (x: 1, y: 1, z: 0))
        .task {
            try? await Task.sleep(for: .seconds(0.4))
            isAppeared = true
            withAnimation(.spring(duration: 1, bounce: 0.6 )) {
                self.degrees = 360
            }
            try? await Task.sleep(for: .seconds(0.6))
            isAppeared = true
            withAnimation(.bouncy(duration: 2)) {
                shadowRadius = true
            }

        }
        .onTapGesture {
            withAnimation(.easeIn(duration: 1)) {
                isAnimating.toggle()
            }
        }
```

Animation explicite -> rotation3DEffect

```
if(item.rarity == .unique){
    Text("Objet Unique 🏆")
        .background(
            Rectangle()
                .fill(item.rarity.color)
                .frame(width: 350, height: 50)
            .cornerRadius(10)
        ).shadow(color: (item.rarity.color).opacity(0.8), radius: 20, x: 0, y: 0)
        .scaleEffect(isAnimating ? 1.2 : 1.0)
        .padding(50)
    /*Button("animated") {
        withAnimation{
            self.degrees += 180
        }
    }*/
}
```

Animation implicites -> scaleEffect

Les bases

ANIMATIONS IMPLICITES ET EXPLICITES

- Animations implicites :
  - Faciles à utiliser via le modificateur .animation.
- Exemples :
  - Redimensionner un bouton avec scaleEffect.
  - Ajouter un flou avec blur.
- Animations explicites :
- Plus flexibles grâce à withAnimation.
- Exemple : rotation 3D d’un bouton avec rotation3DEffect.

Personnalisation avec Animatable

DÉFINITION

- Un attribut animable peut interpoler ses valeurs entre deux états.
- Exemples d’attributs animables intégrés : scaleEffect, rotation3DEffect, etc.

AVANTAGES

- Gère les animations de manière performante (hors du thread principal).
- Prise en charge des types complexes comme CGSize ou CGRect.

ASTUCE PRATIQUE

- AnimatablePair permet de combiner plusieurs attributs animables.

```
.animation(.easeIn(duration: 1), value: shadowRadius)
```

Gestion fine des animations

Transitions et contrôles avancés

RÔLE DES TRANSITIONS

- Gèrent le contexte des animations dans une mise à jour.
- Propage l’animation en cours dans le graphe de dépendances.

GESTION FINE DES ANIMATIONS

- Utilisation de modificateurs d’animation avec transitions ou animation.
- Différentes animations pour l’effet d’échelle et l’ombre.
- Animation programmée (via un Binding).

Expériences complexes

EFFETS COMBINÉS

- Ajoutez des cercles pulsants autour d’un bouton avec overlay et repeatForever.
- Exemples d’animations : rebonds, transitions douces.

MODIFICATIONS CIBLÉES

- Utilisez des modificateurs pour appliquer des animations spécifiques aux sous-vues.
- Limitez les animations accidentelles en utilisant des modificateurs d’animation scindés.

```
.animation(
        Animation.easeOut(duration: 1).repeatForever(autoreverses: false),
        value: 1
    )
```