

Interactive UI

Animation, Gesture, Transition

Gesture

@GestureState 변경사항 감지

- updating
 - 제스처가 변경되는 순간, 일시적인 UI 상태
- onChanged
 - 제스처를 사용하는 동안의 UI 상태
 - 종료 후, 재설정되지 않음
- onEnded
 - 종료 후 상태 재설정

Gesture

@GestureState

- sequenced
 - 여러 제스처 상태를 순서대로 적용 가능
- simultaneously
 - 여러 제스처를 합쳐서 한 번에 인식
- exclusively
 - 여러 제스처 중 처음 시도한 하나의 제스처만 활성화

Gesture1

Gesture2

```
10 struct GestureView: View {
11     @GestureState private var isPressed = false
12     @State private var offset = CGSize.zero
13
14     var body: some View {
15         Circle()
16             .fill(isPressed ? Color.blue : Color.red)
17             .frame(width: 100, height: 100)
18             .offset(offset)
19             .gesture(
20                 LongPressGesture(minimumDuration: 0.5)
21                     .sequenced(before: DragGesture())
22                     .updating($isPressed) { value, state, _ in
23                         if case .first(true) = value {
24                             state = true
25                         }
26                     }
27                     .onEnded { value in
28                         if case .second(true, let drag?) = value {
29                             offset = drag.translation
30                         }
31                     }
32             )
33     }
34 }
```

Animation

State의 변화를 기반으로 동작

명시적

Action



withAnimation



Change State

암묵적

State Binding



Change State

Animation

명시적

```
struct AnimationView: View {
    @Namespace private var animation
    @State private var isExpanded = false

    var body: some View {
        VStack {
            if isExpanded {
                RoundedRectangle(cornerRadius: 25)
                    .fill(Color.green)
                    .matchedGeometryEffect(id: "card", in: animation)
                    .frame(width: 300, height: 300)
                    .onTapGesture {
                        withAnimation(.spring()) {
                            isExpanded.toggle()
                        }
                    }
            } else {
                RoundedRectangle(cornerRadius: 25)
                    .fill(Color.green)
                    .matchedGeometryEffect(id: "card", in: animation)
                    .frame(width: 100, height: 100)
                    .onTapGesture {
                        withAnimation(.spring()) {
                            isExpanded.toggle()
                        }
                    }
            }
        }
    }
}
```

Animation

암시적

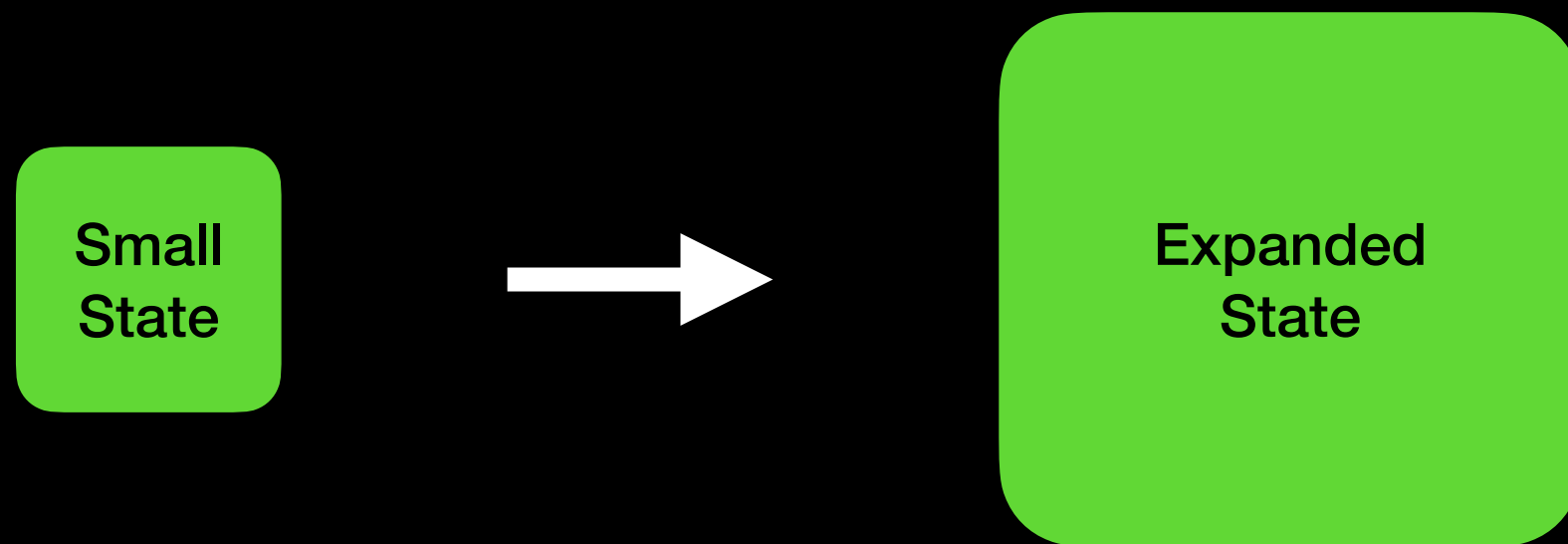
```
struct ImplicitAnimationView: View {
    @State private var isOn = false

    var body: some View {
        Circle()
            .fill(isOn ? Color.green : Color.red)
            .frame(width: isOn ? 200 : 100, height: isOn ? 200 : 100)
            .onTapGesture {
                isOn.toggle()
            }
            .animation(.easeInOut, value: isOn)
    }
}
```

Animation

matchedGeometryEffect

- 두 개의 뷰가 동일한 뷰로 인식하게 하는 기능
- 서로 다른 위치에 있어도 시스템이 자연스럽게 연결



Animation

@Namespace

- 같은 뷰라고 matchedGeometryEffect가 인식할 수 있게 함.
- id: 연결을 위한 키 값
- animation: 같은 공간(Namespace) 안에 있다는 표시

```
RoundedRectangle(cornerRadius: 25)  
    .fill(Color.green)  
    .matchedGeometryEffect(id: "card", in: animation)
```

Animation

프로젝트 별 선택 기준

- `.animation(_ , values:)`
 - 단순한 상태 변화만 줄 때
- `withAnimation`
 - 사용자 액션에 따라 명확히 제어해야 할 때
- `matchedGeometryEffect + @Namespace`
 - View 간에 자연스러운 전환이 필요할 때

Transition

`.asymmetric()`

Struct

Insertion

Removal

`.slide`

`.opacity`

Transition

AnyTransition.modifier

```
struct BlurModifier: ViewModifier {  
    let amount: CGFloat  
  
    func body(content: Content) -> some View {  
        content  
            .blur(radius: amount)  
            .opacity(1 - Double(amount / 10))  
    }  
}  
  
extension AnyTransition {  
    static var blurFade: AnyTransition {  
        .modifier(  
            active: BlurModifier(amount: 10),  
            identity: BlurModifier(amount: 0)  
        )  
    }  
}
```

```
struct TransitionAsymmetricView: View {
    @State private var show = false

    var body: some View {
        VStack(spacing: 20) {
            Button("Toggle Box") {
                withAnimation {
                    show.toggle()
                }
            }

            if show {
                RoundedRectangle(cornerRadius: 20)
                    .fill(Color.purple)
                    .frame(width: 200, height: 200)
                    .transition(.asymmetric(insertion: .slide, removal: .opacity))
            }
        }
    }
}
```