TikTok-Style Comment Sheet in SwiftUI

Implementation Guide

Overview

This guide demonstrates how to replicate TikTok's signature comment sheet interaction where the video scales down and moves up as the comment sheet rises from the bottom. The effect creates a polished, coordinated animation that feels natural and responsive.

Key Techniques

- **Coordinated transforms** — the video's scaleEffect and offset are both driven by the sheet's position
- **Interactive gesture tracking** — @GestureState gives you live drag updates for smooth feel
- **Spring animations** — TikTok uses bouncy, responsive springs
- **Progress-based interpolation** — calculate a 0→1 progress from sheet position, then map to scale/offset ranges

Main View: TikTokStyleView

This is the main container view that coordinates the video scaling with the comment sheet position.

```swift
import SwiftUI

struct TikTokStyleView: View {
    @State private var sheetOffset: CGFloat = 0
    @State private var isSheetOpen = false

    let screenHeight = UIScreen.main.bounds.height
    let sheetMaxHeight: CGFloat = UIScreen.main.bounds.height * 0.7

    // How much the video scales down (0.6 = 60% of original size)
    private let minVideoScale: CGFloat = 0.6

    // Calculate video scale based on sheet position
    private var videoScale: CGFloat {
        let progress = sheetOffset / sheetMaxHeight
        return 1 - (progress * (1 - minVideoScale))
    }

    // Move video up as sheet rises
    private var videoOffsetY: CGFloat {
        let progress = sheetOffset / sheetMaxHeight
        return -progress * (screenHeight * 0.15)
    }
    var body: some View {
        ZStack(alignment: .bottom) {
            Color.black.ignoresSafeArea()

            VideoPlaceholder()
                .scaleEffect(videoScale)
                .offset(y: videoOffsetY)
                .animation(.interactiveSpring(response: 0.3), value: sheetOffset)

            // Comment button
            VStack {
                Spacer()
                HStack {
                    Spacer()
                    Button {
                        withAnimation(.spring(response: 0.35, dampingFraction: 0.8)) {
                            isSheetOpen = true
                            sheetOffset = sheetMaxHeight
                        }
                    } label: {
                        Image(systemName: "bubble.right.fill")
                            .font(.title)
                            .foregroundColor(.white)
                            .padding()
                    }
                    .padding(.trailing, 8)
                    .padding(.bottom, 100)
                }
            }
            .opacity(isSheetOpen ? 0 : 1)

            CommentSheet(
                offset: $sheetOffset,
                isOpen: $isSheetOpen,
                maxHeight: sheetMaxHeight
            )
        }
    }
}
```

Video Placeholder Component

A simple placeholder view representing the video content. Replace this with your actual video player.

```
struct VideoPlaceholder: View {
    var body: some View {
        RoundedRectangle(cornerRadius: 12)
            .fill(
                LinearGradient(
                    colors: [.purple, .blue],
                    startPoint: .topLeading,
                    endPoint: .bottomTrailing
                )
            )
            .overlay(
                Image(systemName: "play.fill")
                    .font(.system(size: 50))
                    .foregroundColor(.white.opacity(0.8))
            )
            .frame(maxWidth: .infinity, maxHeight: .infinity)
            .padding(.horizontal, 1)
    }
}
```

Comment Sheet Component

The draggable comment sheet that handles gesture recognition and provides the offset values that drive the video scaling animation.

```
struct CommentSheet: View {
    @Binding var offset: CGFloat
    @Binding var isOpen: Bool
    let maxHeight: CGFloat

    @GestureState private var dragOffset: CGFloat = 0

    private var currentOffset: CGFloat {
        max(0, maxHeight - offset - dragOffset)
    }

    var body: some View {
        VStack(spacing: 0) {
            // Handle bar
            Capsule()
                .fill(Color.gray.opacity(0.5))
                .frame(width: 40, height: 5)
                .padding(.top, 10)
                .padding(.bottom, 8)

            // Header
            HStack {
                Text("Comments")
                    .font(.headline)
                Spacer()
                Button {
                    closeSheet()
                } label: {
                    Image(systemName: "xmark.circle.fill")
                        .font(.title2)
                        .foregroundColor(.gray)
                }
            }
            .padding(.horizontal)
            .padding(.bottom, 8)

            Divider()

            // Comments list
            ScrollView {
                LazyVStack(alignment: .leading, spacing: 16) {
                    ForEach(0..<30) { i in
                        CommentRow(index: i)
                    }
                }
                .padding()
            }
        }
        .frame(height: maxHeight)
        .background(
            RoundedRectangle(cornerRadius: 20)
                .fill(Color(.systemBackground))
                .shadow(radius: 10)
        )
        .offset(y: currentOffset)
        .gesture(
            DragGesture()
                .updating($dragOffset) { value, state, _ in
                    if value.translation.height > 0 {
                        state = value.translation.height
                    }
                }
                .onEnded { value in
                    let threshold = maxHeight * 0.25
                    if value.translation.height > threshold {
                        closeSheet()
                    } else {
                        withAnimation(.spring(response: 0.35, dampingFraction: 0.8)) {
                            offset = maxHeight
                        }
                    }
                }
        )
    }

    private func closeSheet() {
        withAnimation(.spring(response: 0.35, dampingFraction: 0.8)) {
            offset = 0
            isOpen = false
        }
    }
}
```

Comment Row Component

```
struct CommentRow: View {
    let index: Int

    var body: some View {
        HStack(alignment: .top, spacing: 12) {
            Circle()
                .fill(Color.gray.opacity(0.3))
                .frame(width: 40, height: 40)

            VStack(alignment: .leading, spacing: 4) {
                Text("user_\(index)")
                    .font(.subheadline)
                    .fontWeight(.semibold)
                Text("This is a sample comment.")
                    .font(.subheadline)
                    .foregroundColor(.primary)
                Text("\(index + 1)h ago")
                    .font(.caption)
                    .foregroundColor(.gray)
            }

            Spacer()

            VStack {
                Image(systemName: "heart")
                    .font(.caption)
                Text("\(Int.random(in: 1...999))")
                    .font(.caption2)
            }
            .foregroundColor(.gray)
        }
    }
}
```

Potential Enhancements

To make the implementation even more polished:

- Add corner radius animation to the video as it scales
- Use matchedGeometryEffect for fancier transitions
- Add a dim overlay behind the sheet
- Implement velocity-based dismiss (flick to close)
- Add haptic feedback when sheet reaches thresholds

Usage

Simply add TikTokStyleView to your app's view hierarchy. The component is self-contained and handles all gesture recognition and animation coordination internally.

```
#Preview {
    TikTokStyleView()
}
```