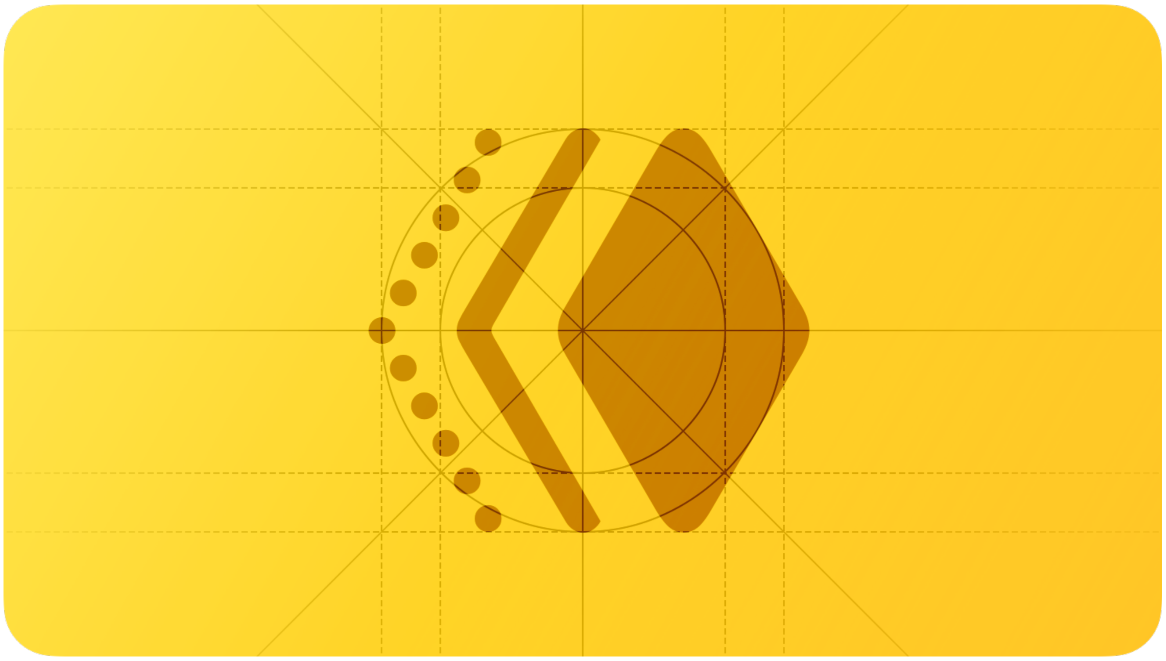
September 9, 2025
Added guidance for Liquid Glass.

Supported platforms

Motion

Beautiful, fluid motions bring the interface to life, conveying status, providing feedback and instruction, and enriching the visual experience of your app or game.



Many system components automatically include motion, letting you offer familiar and consistent experiences throughout your app or game. System components might also adjust their motion in response to factors like accessibility settings or different input methods. For example, the movement of Liquid Glass responds to direct touch interaction with greater emphasis to reinforce the feeling of a tactile experience, but produces a more subdued effect when a person interacts using a trackpad.

If you design custom motion, follow the guidelines below.

Best practices

Add motion purposefully, supporting the experience without overshadowing it. Don't add motion for the sake of adding motion. Gratuitous or excessive animation can distract people and may make them feel disconnected or physically uncomfortable.

Make motion optional. Not everyone can or wants to experience the motion in your app or game, so it's essential to avoid using it as the only way to communicate important information. To help everyone enjoy your app or game, supplement visual feedback by also using alternatives like haptics and audio to communicate.

Providing feedback

Strive for realistic feedback motion that follows people's gestures and expectations. In nongame apps, accurate, realistic motion can help people understand how something works, but feedback motion that doesn't make sense can make them feel disoriented. For example, if someone reveals a view by sliding it down from the top, they don't expect to dismiss the view by sliding it to the side.

Aim for brevity and precision in feedback animations. When animated feedback is brief and precise, it tends to feel lightweight and unobtrusive, and it can often convey information more effectively than prominent animation. For example, when a game displays a succinct animation that's precisely tied to a successful action, players can instantly get the message without being distracted from their gameplay. Another example is in visionOS: When people tap a panorama in Photos, it quickly and smoothly expands to fill the space in front of them, helping them track the transition without making them wait to enjoy the content.

In apps, generally avoid adding motion to UI interactions that occur frequently. The system already provides subtle animations for interactions with standard interface elements. For a custom element, you generally want to avoid making people spend extra time paying attention to unnecessary motion every time they interact with it.

Let people cancel motion. As much as possible, don't make people wait for an animation to complete before they can do anything, especially if they have to experience the animation more than once.

Consider using animated symbols where it makes sense. When you use SF Symbols 5 or later, you can apply animations to SF Symbols or custom symbols. For guidance, see Animations.

Leveraging platform capabilities

Make sure your game's motion looks great by default on each platform you support. In most games, maintaining a consistent frame rate of 30 to 60 fps typically results in a smooth, visually appealing experience. For each platform you support, use the device's graphics capabilities to enable default settings that let people enjoy your game without first having to change those settings.

Let people customize the visual experience of your game to optimize performance or battery life. For example, consider letting people switch between power modes when the system detects the presence of an external power source.

Platform considerations

No additional considerations for iOS, iPadOS, macOS, or tvOS.

visionOS

In addition to subtly communicating context, drawing attention to information, and enriching immersive experiences, motion in visionOS can combine with depth to provide essential feedback when people look at interactive elements. Because motion is likely to be a large part of your visionOS experience, it's crucial to avoid causing distraction, confusion, or discomfort.

As much as possible, avoid displaying motion at the edges of a person's field of view. People can be particularly sensitive to motion that occurs in their peripheral vision: in addition to being distracting, such motion can even cause discomfort because it can make people feel like they or their surroundings are moving. If you need to show an object moving in the periphery during an immersive experience, make sure the object's brightness level is similar to the rest of the visible content.

Help people remain comfortable when showing the movement of large virtual objects. If an object is large enough to fill a lot of the field of view, occluding most or all of passthrough, people can naturally perceive it as being part of their surroundings. To help people perceive the object's movement without making them think that they or their surroundings are moving, you can increase the object's translucency, helping people see through it, or lower its contrast to make its motion less noticeable.

Note

People can experience discomfort even when they're the ones moving a large virtual object, such as a window. Although adjusting translucency and contrast can help in this scenario, consider also keeping a window's size fairly small.

Consider using fades when you need to relocate an object. When an object moves from one location to another, people naturally watch the movement. If such movement doesn't communicate anything useful to people, you can fade the object out before moving it and fade it back in after it's in the new location.

In general, avoid letting people rotate a virtual world. When a virtual world rotates, the experience typically upsets people's sense of stability, even when they control the rotation and the movement is subtle. Instead, consider using instantaneous directional changes during a quick fade-out.

Consider giving people a stationary frame of reference. It can be easier for people to handle visual movement when it's contained within an area that doesn't move. In contrast, if the entire surrounding area appears to move — for example, in a game that automatically moves a player through space — people can feel unwell.

Avoid showing objects that oscillate in a sustained way. In particular, you want to avoid showing an oscillation that has a frequency of around 0.2 Hz because people can be very sensitive to this frequency. If you need to show objects oscillating, aim to keep the amplitude low and consider making the content translucent.

watchOS

SwiftUI provides a powerful and streamlined way to add motion to your app. If you need to use WatchKit to animate layout and appearance changes — or create animated image sequences — see `WKInterfaceImage`.

> **Note**
>
> All layout- and appearance-based animations automatically include built-in easing that plays at the start and end of the animation. You can't turn off or customize easing.

Resources

Related

Feedback

Accessibility

Spatial layout

Immersive experiences

Developer documentation

Animating views and transitions — SwiftUI

Videos

Enhance your UI animations **Create custom visual effects** **Design considerations for**

and transitions   with SwiftUI   vision and motion

Change log

Date	Changes
September 9, 2025	Added guidance for Liquid Glass.
June 10, 2024	Added game-specific examples and enhanced guidance for using motion in games.
February 2, 2024	Enhanced guidance for minimizing peripheral motion in visionOS apps.
June 21, 2023	Updated to include guidance for visionOS.