Article

Optimizing your widget for accented rendering mode and Liquid Glass

Make your widget feel at home on Apple platforms and Liquid Glass by using accented rendering mode.

Overview

iPhone, iPad, Mac, and Apple Watch use Liquid Glass, a dynamic, adaptive material that also applies to widgets. When a person chooses a tinted or clear appearance for their Home Screen, the system:

- Renders your widget in the `accented` rendering mode
- Tints primary and accented content white in iOS and macOS
- Tints primary content white and accented content in the color of the watch face in watchOS
- Tints opaque images with a single white color
- Maintains opacity for transparent content and gradients, and tints them white
- Removes the background and replaces it with a themed glass or tinted color effect

By already supporting the `accented` rendering mode, some widgets don't need any further adjustments. However, you might need to update your widget to keep its text readable and make it look at home with Liquid Glass; for example, if your widget includes images, gradients, or transparent content.

Support Liquid Glass

To update your widget to support Liquid Glass:

1. Add the `widgetRenderingMode` environment variable and conditionally update your widget layout for each rendering mode as explained in the previous section.
2. Display full-color images, page, or partially transparent content only for the `fullColor` rendering mode.
3. Adjust your widget's layout as needed for the `accented` rendering mode.
4. Group your views into a primary and an accent group using the `widgetAccentable(_:)` view modifier. Views you don't mark as acceptable are part of the primary group.
5. Configure the rendering of any image using the `WidgetAccentedRenderingMode` view modifier.

Choose rendering modes for images and views

Using the `WidgetAccentedRenderingMode` view modifier, conditionally render images and views as needed:

`accented`
: Tints the image to the accent color. In iOS and macOS, primary and accent colors are white, causing the image to be a solid white color. In watchOS, the accent color matches the color on the watch face.

`desaturated`
: Desaturates the image in iOS, macOS, and watchOS.

`accentedDesaturated`
: Combines `accented` and `desaturated` accented rendering modes. In iOS and macOS, the image appears a little whiter compared to the `desaturated` rendering. In watchOS, the desaturated image takes on the color of the watch face.

`fullColor`
: Renders the image in full color, without modifications in iOS and macOS. In watchOS, the system ignores this rendering mode to make sure the widget blends in with the watch face.

> **Tip**
> Using `accented`, `desaturated`, or `accentedDesaturated` rendering modes helps the widget fit the system's cohesive look on the Home Screen. Reserve the `fullColor` rendering mode for images that represent media content, such as album artwork or a book over.

To learn more about Liquid Glass and how to design and develop interfaces that work well with the material, refer to Liquid Glass and Adopting Liquid Glass.

See Also

Layout and presentation

Supporting additional widget sizes
Offer widgets in additional contexts by adding support for various widget sizes.

Displaying the right widget background
Group your widget's background views and mark them as removable to ensure your widget appears correctly for each context and platform.

Adding StandBy and CarPlay support to your widget
Ensure that your small system family widget works well in StandBy and CarPlay.

`struct WidgetRenderingMode`
Constants that indicate the rendering mode for a widget.

`struct WidgetAccentedRenderingMode`
Constants that indicate the rendering mode for an `Image` in when displayed in a widget in `accented` mode.

`struct AccessoryWidgetBackground`
An adaptive background view that provides a standard appearance based on the the widget's environment.

`struct WidgetLocation`
Values that indicate different widget locations.