Sample Code

Supporting HDR images in your app

Load, display, edit, and save HDR images using SwiftUI and Core Image.

Download

iOS 17.0+ | iPadOS 17.0+ | macOS 14.0+ | Xcode 15.0+

Overview

This sample code project shows how to read, write, edit, and display HDR images using SwiftUI, UIKit, AppKit, Core Image, and Core Graphics. It loads a film strip of multiple images from on-disk or from the Photos library. Then it allows you to select, edit, and save an image in HDR. The sample uses several new APIs in various frameworks to correctly handle HDR images in a complete HDR workflow.

> **Note**
>
> This sample code project is associated with WWDC23 session 10181: Support HDR images in your app.

Configure the sample code project

Before you run this sample code project in Xcode, ensure you're using iOS 17 or later for the iOS target, and macOS 14 or later for the Mac target.

Enable HDR on image views

To turn on HDR in the SwiftUI image view, the sample uses the `allowedDynamicRange` modifier in the film strip to show limited HDR headroom.

```
FilmStrip(assets: $assets, selectedAsset: $selectedAsset)
    .allowedDynamicRange(.constrainedHigh)
    .frame(height: geometry.size.height / 10.0)
```

For the UIKit main image view, the sample adds the `preferredImageDynamicRange` property to the `UIImageView`.

```
let view = UIImageView()
view.preferredImageDynamicRange = .high
```

For AppKit, the sample adds the property to the `NSImageView`.

```
let view = NSImageView()
view.preferredImageDynamicRange = .high
```

Edit HDR using Core Image

To ensure Core Image reads Gain Map HDR images as HDR, the sample sets the `CIImageOption .expandToHDR` property to `true`. To modify the HDR image data, it uses `CIFilter` filters.

```
let ciOptions: [CIImageOption: Any] = [.applyOrientationProperty: true, .expandToHDR
```

The sample saves the image to disk if it's an on-disk file to begin with. The sample uses a `CIContext` and `CGImageDestination` to render a `CIImage` into a `CGImage` and write it to a 10-bit HEIC image file.

```
let cgImage = context.createCGImage(image,
                                    from: image.extent,
                                    format: .RGB10,
                                    colorSpace: colorspace ?? CGColorSpace(name: CGC
                                    deferred: true)
```

The sample writes the edited HDR image data back to the Photos library by writing a 10-bit HEIC image to the `renderedContentURL` of a `PHContentEditingOutput` object. The sample only uses this path when an image comes from the Photos library using the `PhotosPicker`.

```
guard let outputURL = try? output.renderedContentURL(for: .heic) else {
    print("Failed to obtain HEIC output URL.")
    return nil
}
```

See Also

Image creation

`func jpegData(compressionQuality: CGFloat) -> Data?`

Returns a data object that contains the image in JPEG format.

`func pngData() -> Data?`

Returns a data object that contains the specified image in PNG format.

~~`func UIGraphicsBeginImageContext(CGSize)`~~

Creates a bitmap-based graphics context and makes it the current context.

Deprecated

~~`func UIGraphicsGetImageFromCurrentImageContext() -> UIImage?`~~

Returns an image from the contents of the current bitmap-based graphics context.

Deprecated

~~`func UIGraphicsEndImageContext()`~~

Removes the current bitmap-based graphics context from the top of the stack.

Deprecated