API Collection

Core Graphics interoperability

Pass image data between the Core Graphics framework and the vImage library.

Overview

The vImage library uses the `CGImage` class as the main type to consume and produce still images. A `CGImage` instance may originate from `NSImage` or `UIImage` images, or from a `CGContext` drawing destination.

A typical Core Graphics-based vImage workflow consists of:

1. Selecting a source image, such as a Core Graphics-backed `UIImage` instance.
2. Initializing a vImage buffer from the image's bitmap data.
3. Performing an operation on the vImage buffer, such as scaling or adjusting gamma.
4. Creating a destination image from the operation result with the same image format as the source image.

vImage provides the following functions that simplify interoperation with Core Graphics:

- `vImageBuffer_InitWithCGImage(_:_:_:_:)` initializes a vImage buffer with the contents of a Core Graphics image.
- `vImageCreateCGImageFromBuffer(_:_:_:_:_:)` creates a Core Graphics image from a vImage buffer.

The following code shows a passthrough function that accepts a `CGImage` image, populates a vImage buffer from the image, and generates a `CGImage` image from the buffer.

In this example, the call to `vImageBuffer_InitWithCGImage(_:_:_:_:)` populates the `vImage_CGImageFormat` and the `vImage_Buffer` variables with the properties of the source image:

```
static func passThrough(sourceImage: CGImage) -> CGImage? {

    var format = vImage_CGImageFormat()
    var buffer = vImage_Buffer()

    defer {
        buffer.free()
    }

    vImageBuffer_InitWithCGImage(
        &buffer,
        &format,
        nil,
        sourceImage,
        vImage_Flags(kvImageNoFlags))

    // Perform image-processing operations on `buffer`.

    let destinationCGImage = vImageCreateCGImageFromBuffer(
        &buffer,
        &format,
        nil,
        nil,
        vImage_Flags(kvImageNoFlags),
        nil)

    return destinationCGImage?.takeRetainedValue()
}
```

Pass a fully initialized `vImage_CGImageFormat` to specify that `vImageBuffer_InitWithCGImage(_:_:_:_:_:)` converts the source `CGImage` image to the format that `format` describes. The following example converts the source image to a three-channel, 8-bit-per-channel RGB image:

```
static func passThrough(sourceImage: CGImage) -> CGImage? {

    var format = vImage_CGImageFormat(
        bitsPerComponent: 8,
        bitsPerPixel: 8 * 3,
        colorSpace: CGColorSpaceCreateDeviceRGB(),
        bitmapInfo: CGBitmapInfo(rawValue: CGImageAlphaInfo.none.rawValue),
        renderingIntent: .defaultIntent)!
```

```
    var buffer = vImage_Buffer()

    defer {
        buffer.free()
    }

    vImageBuffer_InitWithCGImage(
        &buffer,
        &format,
        nil,
        sourceImage,
        vImage_Flags(kvImageNoFlags))

    // Perform image-processing operations on RGB888 `buffer`.

    let destinationCGImage = vImageCreateCGImageFromBuffer(
        &buffer,
        &format,
        nil,
        nil,
        vImage_Flags(kvImageNoFlags),
        nil)

    return destinationCGImage?.takeRetainedValue()
}
```

Topics

Initializing vImage buffers from Core Graphics images

`func vImageBuffer_InitWithCGImage(UnsafeMutablePointer<vImage_Buffer>, UnsafeMutablePointer<vImage_CGImageFormat>, UnsafePointer<CGFloat>!, CGImage, vImage_Flags) -> vImage_Error`

Initializes a vImage buffer with the contents of a Core Graphics image.

Creating Core Graphics images from vImage buffers

```
func vImageCreateCGImageFromBuffer(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_CGImageFormat>, ((UnsafeMutableRawPointer?, UnsafeMutableRawPointer?) -> Void)!, UnsafeMutableRawPointer!, vImage_Flags, UnsafeMutablePointer<vImage_Error>!) -> Unmanaged<CGImage>!
```

Creates a Core Graphics image from a vImage buffer.

Creating Core Graphics image formats

```
struct vImage_CGImageFormat
```

The description of a Core Graphics image.

Querying Core Graphics image format attributes

```
func vImageCGImageFormat_IsEqual(UnsafePointer<vImage_CGImageFormat>!, UnsafePointer<vImage_CGImageFormat>!) -> Bool
```

Returns a Boolean value that indicates whether two vImage Core Graphics image formats are equal.

```
func vImageCGImageFormat_GetComponentCount(UnsafePointer<vImage_CGImageFormat>) -> UInt32
```

Calculates the number of color and alpha channels for a specified image format.

Creating Core Graphics color spaces

```
func vImageCreateRGBColorSpaceWithPrimariesAndTransferFunction(UnsafePointer<vImageRGBPrimaries>, UnsafePointer<vImageTransferFunction>, CGColorRenderingIntent, vImage_Flags, UnsafeMutablePointer<vImage_Error>!) -> Unmanaged<CGColorSpace>!
```

Creates an RGB color space based on primitives from Y′CbCr specifications.

```
struct vImageRGBPrimaries
```

A representation of the chromaticity of primaries that define a color space.

```
struct vImageTransferFunction
```

A transfer function to convert from linear to nonlinear RGB.

```
func vImageCreateMonochromeColorSpaceWithWhitePointAndTransferFunction(UnsafePointer<vImageWhitePoint>, UnsafePointer<vImageTransferFunction>, CGColorRenderingIntent, vImage_Flags, UnsafeMutablePointer<vImage_Error>!) -> Unmanaged<CGColorSpace>!
```

Creates a monochrome color space based on primitives from Y′CbCr specifications.

`struct` `vImageWhitePoint`

A representation of a white point according to the CIE 1931 color space.