Function

vImageBuffer_InitWithCGImage(_:_:_:_:)

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

iOS 7.0+ | iPadOS 7.0+ | Mac Catalyst 13.1+ | macOS 10.9+ | tvOS 7.0+ | visionOS 1.0+ | watchOS 1.0+

```
func vImageBuffer_InitWithCGImage(
    _ buf: UnsafeMutablePointer<vImage_Buffer>,
    _ format: UnsafeMutablePointer<vImage_CGImageFormat>,
    _ backgroundColor: UnsafePointer<CGFloat>!,
    _ image: CGImage,
    _ flags: vImage_Flags
) -> vImage_Error
```

Parameters

buf
The destination vImage buffer. On output, an initialized buffer with all fields populated.

format
A vImage_CGImageFormat structure. Pass an empty structure to specify that the function populates the format with the properties of the Core Graphics image. Pass a populated structure to specify that the function converts the Core Graphics image to the format.

backgroundColor
If the source image contains alpha information and the format doesn't contain alpha information, this function flattens the source image against this parameter.

image

The source Core Graphics image.

`flags`
The options to use when performing the operation. Pass `kvImageNoAllocate` if the destination buffer references existing data; otherwise, pass `kvImageNoFlags`.

Return Value

`kvImageNoError`; otherwise, one of the error codes in Data Types and Constants.

Mentioned in

- Converting bitmap data between Core Graphics images and vImage buffers
- Optimizing image-processing performance

Discussion

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()
}
```