Initializer

init(cgImage:cgImageFormat:pixelFormat:)

Returns a new pixel buffer initialized from a Core Graphics image.

iOS 16.0+ | iPadOS 16.0+ | Mac Catalyst | macOS 13.0+ | tvOS 16.0+ | visionOS | watchOS 9.0+

```
init(
    cgImage: CGImage,
    cgImageFormat: inout vImage_CGImageFormat,
    pixelFormat: Format.Type = Format.self
) throws
```

Available when `Format` conforms to `InitializableFromCGImage` and `StaticPixelFormat`.

Parameters

`cgImage`
The source image.

`cgImageFormat`
The format of the image. Pass an empty `vImage_CGImageFormat` to specify that the function populates `cgImageFormat` with the properties of the image. Pass a populated `vImage_CGImageFormat` to specify that the function converts the image data to the format you specify.

`pixelFormat`
The pixel format of the initialized buffer.

Mentioned in

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

Discussion

For example, the following code creates a single-channel, 8-bit per pixel buffer from a CGImage of unknown bit depth.

When you pass a populated cgImageFormat, the init(cgImage:cgImageFormat:pixelFormat:) initializer performs the conversion from the CGImage instance's format to the vImage_CGImageFormat that you specify.

Note the bitsPerComponent, bitsPerPixel, colorSpace, and bitmapInfo.

```
let cgImage = [ ... ]

let pixelFormat = vImage.Planar8.self

var imageFormat = vImage_CGImageFormat(
    bitsPerComponent: pixelFormat.bitsPerComponent, // 8
    bitsPerPixel: pixelFormat.bitsPerPixel,         // 8
    colorSpace: CGColorSpaceCreateDeviceGray(),
    bitmapInfo: CGBitmapInfo(rawValue: CGImageAlphaInfo.none.rawValue))!

let buffer = vImage.PixelBuffer(
   cgImage: cgImage,
   cgImageFormat: &imageFormat,
   pixelFormat: pixelFormat)
```

The following code shows a similar workflow, but it creates a 32-bit per channel RGBA buffer:

```
let cgImage = [ ... ]

let pixelFormat = vImage.InterleavedFx4.self

var imageFormat = vImage_CGImageFormat(
    bitsPerComponent: pixelFormat.bitsPerComponent, // 32
    bitsPerPixel: pixelFormat.bitsPerPixel,         // 32 * 4
    colorSpace: CGColorSpaceCreateDeviceRGB(),
    bitmapInfo: CGBitmapInfo(rawValue:
```

```
                                                kCGBitmapByteOrder32Host.rawValue |
                                                CGBitmapInfo.floatComponents.rawValue |
                                                CGImageAlphaInfo.noneSkipLast.rawValue))

 let buffer = vImage.PixelBuffer(
     cgImage: cgImage,
     cgImageFormat: &imageFormat,
     pixelFormat: pixelFormat)
```

If you pass an empty `cgImageFormat`, the `init(cgImage:cgImageFormat:pixelFormat:)` initializer populates the `vImage_CGImageFormat` with the properties of the `CGImage` instance. For example, the following code initializes a pixel buffer from a 8-bit-per-channel, 4-channel RGB image:

```
let cgImage = [ ... ]

var cgImageFormat = vImage_CGImageFormat()

let buffer = try vImage.PixelBuffer(
    cgImage: cgImage,
    cgImageFormat: &cgImageFormat,
    pixelFormat: vImage.Interleaved8x4.self)

// Prints "8  32  1 (`CGColorSpaceModel.rgb.rawValue`)".
print(cgImageFormat.bitsPerComponent,
      cgImageFormat.bitsPerPixel,
      cgImageFormat.colorSpace.takeRetainedValue().model.rawValue)
```

Important

If you specify a populated `vImage_CGImageFormat`, its bits per component and bits per pixel must match those of the buffer's `pixelFormat`. If you specify an empty `vImage_CGImageFormat`, the bits per component and bits per pixel of the `CGImage` must match those of the buffer's `pixelFormat`.