Type Method

makeDynamicPixelBufferAndCGImageFormat(cgImage:)

Returns a new dynamic pixel format pixel buffer and Core Graphics image format structure from a Core Graphics image.

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

```
static func makeDynamicPixelBufferAndCGImageFormat(cgImage: CGImage) throws -> (pix
Image.PixelBuffer<vImage.DynamicPixelFormat>, cgImageFormat: vImage_CGImage
Format)
```

Available when `Format` is `vImage.DynamicPixelFormat`.

Parameters

`cgImage`
The source Core Graphics image.

Return Value

A new pixel buffer of type `vImage.DynamicPixelFormat` and a `vImage_CGImageFormat` that describes the ordering and number of the color channels, the size and type of the data in the color channels, and whether or not the data is premultiplied by alpha.

Discussion

Use this function to create an image format and pixel buffer from a `CGImage` instance. The pixel buffer that this function returns has a pixel format that's unknown at compile time.

Note that you can call `init(cgImage:cgImageFormat:pixelFormat:)` to create a statically typed pixel buffer from a `CGImage` instance. However, this function allows you to create your own any-to-any converter for more control over the conversion process to a statically typed buffer.

For example, the following code creates a dynamically typed pixel buffer and `vImage_CGImageFormat` from a source image. Use the returned `vImage_CGImageFormat` as the source format on an any-to-any converter. For example:

```
let srcImage = imageLiteral(resourceName: " ... ").cgImage(
    forProposedRect: nil,
    context: nil,
    hints: nil)!

let source = try vImage.PixelBuffer.makeDynamicPixelBufferAndCGImageFormat(
    cgImage: srcImage)

let destFormat = vImage_CGImageFormat(
    bitsPerComponent: 8,
    bitsPerPixel: 8,
    colorSpace: CGColorSpaceCreateDeviceGray(),
    bitmapInfo: CGBitmapInfo(rawValue: CGImageAlphaInfo.none.rawValue))!

let destBuffer = vImage.PixelBuffer(size: source.pixelBuffer.size,
                                    pixelFormat: vImage.Planar8.self)

let converter = try vImageConverter.make(sourceFormat: source.cgImageFormat,
                                         destinationFormat: destFormat)

try converter.convert(from: source.pixelBuffer, to: destBuffer)
```

On return, `destBuffer` contains an 8-bit, grayscale representation of the original image.

See Also

Creating a pixel buffer and image format

`static func makePixelBufferAndCGImageFormat(cgImage: CGImage, pixelFormat: Format.Type) throws -> (pixelBuffer: vImage.PixelBuffer<Format>, cgImageFormat: vImage_CGImageFormat)`

Returns a new pixel buffer and Core Graphics image format structure from a Core Graphics image.