Instance Property

rowStride

The width, in pixels, of the underlying memory, including any additional row byte padding.

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

```
var rowStride: Int { get }
```

Available when `Format` conforms to `StaticPixelFormat`.

Discussion

Use this property when working with the `AccelerateBuffer` and `AccelerateMutableBuffer` functions. The property allows you to limit work you do on the underlying memory of a pixel buffer to just the visible pixels.

In some cases, vImage adds extra bytes at the end of each row. Use `rowStride` to find the width, in pixels, of the memory pointed to by the vImage buffer's `data` property.

For example, the following code creates a 5 x 1 `vImage.InterleavedFx2` buffer that contains the pixel values `(1, 2)`, `(3, 4)`, `(5, 6)`, `(7, 8)`, and `(9, 10)`:

```
let values: [Float] = [1, 2,  3, 4,  5, 6,  7, 8,  9, 10]
let buffer = vImage.PixelBuffer(pixelValues: values,
                                size: vImage.Size(width: 5, height: 1),
                                pixelFormat: vImage.InterleavedFx2.self)
```

In this case, the `rowStride` is 8. This equates to the buffer's `rowBytes` property of 64.

Each pixel contains two 32-bit values, which is equivalent to 8 bytes per pixel, and 8 bytes per pixel multiplied by the `rowStride` of 8 pixels equals 64.

Working with the underlying row-bytes property.

If you're accessing the underlying bytes of a pixel buffer, note that the `rowStride` ∗ `channelCount` ∗ `bitCountPerPixel` may not correspond to the `rowBytes` properly of the underlying `vImage_Buffer`. This may happen when the integer row stride multiplied by the number of bytes per pixel isn't a factor of the row bytes.

In this case, use the `withUnsafeVImageBuffer(_:)` function to access the underlying vImage buffer.

For example, the following code creates a 3-channel, 8-bit-per-channel 5 x 5 pixel buffer. The code uses the `rowStride` property to fill the buffer with the constant value 99. The pixel buffer's underlying bytes per row is 16, but because the `rowStride` property returns 5, the code calculates `5 * 3` or 15 bytes per row. Although the code correctly calculates the `5 * 5 * 3` elements, it doesn't consider the row padding.

```
let width = 5
let height = 5

let pixelBuffer = vImage.PixelBuffer(size: .init(width: width,
                                                  height: height),
                                     pixelFormat: vImage.Interleaved8x3.self)

pixelBuffer.withUnsafeMutableBufferPointer { buf in
    // The following calculation for `n` doesn't properly consider the row
    // padding and fails to fill all the elements in `buf`.
    let n = pixelBuffer.rowStride * pixelBuffer.channelCount * pixelBuffer.height

    for i in 0 ..< n {
        buf[i] = 99
    }
}
```

After the loop completes, the last four elements of the pixel buffer's `array` are `0`. For more information on how the vImage library adds additional padding to a buffer's memory, see `vImage_Buffer`.

The code below performs a similar loop to that above, but uses the `rowBytes` property to determine the length of the loop. In this case, the code successfully fills the entire buffer with the value 99.

```
pixelBuffer.withUnsafeVImageBuffer {
    // The following calculation for `n` properly considers the row
    // padding and fills all the elements in `buf`.
    let n = $0.rowBytes * Int($0.height)

    let buf = UnsafeMutableBufferPointer(
        start: $0.data.assumingMemoryBound(to: Pixel_8.self),
        count: n )

    for i in 0 ..< buf.count {
        buf[i] = 99
    }
}
```

See Also

Inspecting a pixel buffer

`var width: Int`

The width of the pixel buffer.

`var height: Int`

The height of the pixel buffer.

`var size: vImage.Size`

The size of the pixel buffer.

`var channelCount: Int`

Returns the number of channels.

`var byteCountPerPixel: Int`

Returns the number of bytes per pixel.

`var count: Int`

The total number of pixels multiplied by the number of channels in the buffer, including any row padding.

`var array: [Format.ComponentType]`

An array of `width * height * channelCount` values that's a copy of the buffer's visible contents.