vImage Data Types and Constants Reference

Performance > Graphics & Imaging

2007-07-12

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

vImage Data Types and Constants Reference

Framework:	Accelerate/vImage
Companion guide	vImage Programming Guide
Declared in	vImage_Types.h

Overview

The data types and constants defined in this document are used by vImage functions. The primary vImage data type is the vImage buffer, which contains a pointer to image data along with other image data information. The vImage framework also defines data types for planar and interleaved pixel types, a resampling callback filter, and an affine transform. It provides constants that specify errors that can be returned by vImage functions and flags that you can pass to a function to specify a variety of processing options.

Data Types

vImagePixelCount

A type for the number of pixels.

```
typedef unsigned long  vImagePixelCount;
```

Discussion
For LP64 (ppc64) this is a 64-bit quantity.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vImage_Types.h`

vImage_Buffer

The basic data structure used by vImage functions for passing image data.

```
typedef struct vImage_Buffer
{
    void        *data;
    vImagePixelCount    height;
    vImagePixelCount    width;
    size_t    rowBytes;
}vImage_Buffer;
```

Fields

`data`

A pointer to memory for image data. The image data can be in planar (Planar8, PlanarF) or interleaved (ARGB8888, ARGBFFFF, RGBA8888, or RGBAFFFF) formats. If you are using the vImage buffer to provide an image, then the pointer should point to the top left pixel of the image. If you are providing the vImage buffer to a function that fills the memory with image data (that is, as a destination buffer), the pointer must point to an area of memory that is an appropriate size for the destination buffer. Specifically, size of the memory, in bytes, must be at least the height of the image data multiplied by the number of row bytes.

`height`

The number of pixels in one column of the image.

`width`

The number of pixels in one row of the image.

`rowBytes`

The number of bytes in a pixel row. This is the distance, in bytes, between the start of one row of the image and the start of the next. This quantity must be at least the width times the pixel size, where pixel size depends on the image format. You can provide a larger value, in which case the extra bytes will extend beyond the end of each row of pixels.

You may want to provide a larger value for one of two reasons: to improve performance, or to describe an image within a larger image without copying the data. The extra bytes are not considered part of the image represented by the vImage buffer.

Discussion

vImage functions will not attempt to read pixel data outside the area described by the height and width fields of the vImage buffer. The function will also not write data outside that area.

Declared In

`vImage_Types.h`

vImage_AffineTransform

A structure for values that represent an affine transformation.

```
typedef struct vImage_AffineTransform
{
    float a, b, c, d;
    float tx, ty;
}vImage_AffineTransform;
```

Discussion

This structure represents the 3x2 matrix :

$$\begin{pmatrix} a & b \\ c & d \\ tx & ty \end{pmatrix}$$

The vImage affine transform structure is just like the Quartz `CGAffineTransform` data structure. *CGAffineTransform Reference* describes functions for creating and manipulating matrixes of this form.

Declared In
`vImage_Types.h`

vImage_Error

A type for image errors.

```
typedef  ssize_t  vImage_Error;
```

Discussion
"Error Codes" (page 9) describes the constants that use this type.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

vImage_Flags

A type for processing options.

```
typedef uint32_t vImage_Flags;
```

Discussion
"Processing Flags" (page 11) describes the constants that use this type.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

Pixel_8

A type for an 8-bit planar pixel value

```
typedef uint8_t Pixel_8;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

Pixel_F

A type for a floating-point planar pixel value

```
typedef float Pixel_F;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

Pixel_8888

A type for an interleaved, 8 bits per channel pixel value.

```
typedef uint8_t Pixel_8888[4];
```

Discussion
For example, `uint8_t[4] = { alpha, red, green, blue }` for ARGB data.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

Pixel_FFFF

A type for an interleaved, floating-point pixel value.

```
typedef float Pixel_FFFF[4];
```

Discussion
For example, `float[4] = { alpha, red, green, blue }` for ARGB data.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

GammaFunction

A type for a gamma function.

```
typedef void * GammaFunction;
```

Discussion
You use this data type when you create a gamma function. See `vImageCreateGammaFunction`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vImage_Types.h`

ResamplingFilter

A pointer to a resampling filter callback function.

```
typedef void * ResamplingFilter;
```

Discussion
You pass a resampling filter callback to a shear function. The resampling filter pointer can point to a structure that contains a function, rows of precalculated values, flag settings, and so on. The shear function requires that the structure contains a scale factor.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`vImage_Types.h`

Constants

Error Codes

Error codes returned by vImage functions.

```
enum
{
    kvImageNoError                  =   0,
    kvImageRoiLargerThanInputBuffer =   -21766,
    kvImageInvalidKernelSize        =   -21767,
    kvImageNoEdgeStyleSpecified     =   -21768,
    kvImageInvalidOffset_X          =   -21769,
    kvImageInvalidOffset_Y          =   -21770,
    kvImageMemoryAllocationError    =   -21771,
    kvImageNullPointerArgument      =   -21772,
    kvImageInvalidParameter         =   -21773,
    kvImageBufferSizeMismatch       =   -21774,
    kvImageUnknownFlagsBit           =    -21775
};
```

Constants
`kvImageNoError`

The vImage function completed without error.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageRoiLargerThanInputBuffer`
: The region of interest, as specified by the `srcOffsetToROI_X` and `srcOffsetToROI_Y` parameters and the height and width of the destination buffer, extends beyond the bottom edge or right edge of the source buffer.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageInvalidKernelSize`
: Either the kernel height, the kernel width, or both, are even.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageNoEdgeStyleSpecified`
: The function requires you to set at least one edge option flags: `kvImageCopyInPlace`, `kvImageBackgroundColorFill`, or `kvImageEdgeExtend`, but none is set. See "Processing Flags" (page 11).

`kvImageInvalidOffset_X`
: The `srcOffsetToROI_X` parameter that specifies the left edge of the region of interest is greater than the width of the source image.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageInvalidOffset_Y`
: The `srcOffsetToROI_Y` parameter that specifies the top edge of the region of interest is greater than the height of the source image.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageMemoryAllocationError`
: An attempt to allocate memory failed.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageNullPointerArgument`
: A pointer parameter is `NULL` and it must not be.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageInvalidParameter`
: Invalid parameter.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageBufferSizeMismatch`
: The function requires the source and destination buffers to have the same height and the same width, but they do not.

  Available in Mac OS X v10.3 and later.

  Declared in `vImage_Types.h`.

`kvImageUnknownFlagsBit`
The flag is not recognized.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

Declared In
`vImage_Types.h`

Processing Flags

Flags that specify options for vImage functions.

```
enum
{
    kvImageNoFlags                  =   0,
    kvImageLeaveAlphaUnchanged      =   1,
    kvImageCopyInPlace              =   2,
    kvImageBackgroundColorFill      =   4,
    kvImageEdgeExtend               =   8,
    kvImageDoNotTile                =   16,
    kvImageHighQualityResampling    =   32,
    kvImageTruncateKernel           =   64,
    kvImageGetTempBufferSize        =   128
};
```

Constants
`kvImageNoFlags`
Do not set any flags.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageLeaveAlphaUnchanged`
Operate on red, green, and blue channels only. When you set this flag, the alpha value is copied from source to destination. You can set this flag only for interleaved image formats.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageCopyInPlace`
Copy the value of the edge pixel in the source to the destination. When you set this flag, and a convolution function is processing an image pixel for which some of the kernel extends beyond the image boundaries, vImage does not computer the convolution. Instead, the value of the particular pixel unchanged; it's simply copied to the destination image. This flag is valid only for convolution operations. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageBackgroundColorFill`

A background color fill. The associated value is a background color (that is, a pixel value). When you set this flag, vImage assigns the pixel value to all pixels outside the image. You can set this flag for convolution and geometry functions. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageEdgeExtend`

Extend the edges of the image infinitely. When you set this flag, vImage replicates the edges of the image outward. It repeats the top row of the image infinitely above the image, the bottom row infinitely below the image, the first column infinitely to the left of the image, and the last column infinitely to the right. For spaces that are diagonal to the image, vImage uses the value of the corresponding corner pixel. For example, for all pixels that are both above and to the left of the image, the upper-leftmost pixel of the image (the pixel at row 0, column 0) supplies the value. In this way, vImage assigns every pixel location outside the image some value. You can set this flag for convolution and geometry functions. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageDoNotTile`

Do not use vImage internal tiling routines. When you set this flag, vImage turns off internal tiling. Set this flag if you want to perform your own tiling or your own multithreading, or to use the minimum or maximum filters in place.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageHighQualityResampling`

Use a higher quality, slower resampling filter for for geometry operations—shear, scale, rotate, affine transform, and so forth.

Available in Mac OS X v10.3 and later.

Declared in `vImage_Types.h`.

`kvImageTruncateKernel`

Use the part of the kernel that overlaps the image. This flag is valid only for convolution operations. When you set this flag, vImage restricts calculations to the portion of the kernel overlapping the image. It corrects the calculated pixel by first multiplying by the sum of all the kernel elements, then dividing by the sum of the kernel elements that are actually used. This preserves image brightness at the edges.

For integer kernels:

```
real_divisor = divisor * (sum of used kernel elements) / (sum of kernel
elements)
```

The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.

Kernel truncation is not robust for certain kernels. It can ail when any rectangular segment of the kernel that includes the center, and at least one of the corners, sums to zero. You typically see this with emboss or edge detection filters, or other filters that are designed to find the slope of a signal. For those kinds of filters, you should use the `kvImageEdgeExtend` option instead.

Available in Mac OS X v10.4 and later.

Declared in `vImage_Types.h`.

`kvImageGetTempBufferSize`

Get the minimum temporary buffer size for the operation, given the parameters provided. When you set this flag, the function returns the number of bytes required for the temporary buffer. A negative value specifies an error.

Available in Mac OS X v10.4 and later.

Declared in `vImage_Types.h`.

Discussion

You can pass multiple flags to a function by adding the flag values together. For example, to leave alpha unchanged and turn off tiling, you can pass:

```
kvImageLeaveAlphaUnchanged + kvImageDoNotTile
```

Three of the flags are mutually exclusive: `kvImageCopyInPlace`, `kvImageBackgroundColorFill`, and `kvImageEdgeExtend`. Never pass more than one of these flag values in the same flag parameter.

When passing flags to a function, do not set values for flags that are not used by the function. If the function requires you to set certain flag values, do so. For example, for the convolution function, you must set exactly one of `kvImageCopyInPlace`, `kvImageBackgroundColorFill`, and `kvImageEdgeExtend`. Otherwise the function may return an error. If you don't want to set flag values, pass `kvImageNoFlags`.

Declared In

`vImage_Types.h`

Document Revision History

This table describes the changes to *vImage Data Types and Constants Reference*.

Date	Notes
2007-07-12	Updated for Mac OS X v10.4.
	The content in this document was formerly part of *Optimizing Image Processing With vImage*.
	Updated the data types in `vImage_Buffer` (page 5).
	Revised the description of `kvImageTruncateKernel` (page 13).
	Added `GammaFunction` (page 8), `ResamplingFilter` (page 9), and `kvImageUnknownFlagsBit` (page 11).

Document Revision History

Index