Function

vImageNewResamplingFilterForFunctionUsingBuffer(_:_:_:_:_:_:)

Creates a resampling filter object that encapsulates a resampling kernel function that you provide.

iOS 5.0+ | iPadOS 5.0+ | Mac Catalyst 13.1+ | macOS 10.3+ | tvOS 5.0+ | visionOS 1.0+ | watchOS 1.0+

```
func vImageNewResamplingFilterForFunctionUsingBuffer(
    _ filter: ResamplingFilter,
    _ scale: Float,
    _ kernelFunc: ((UnsafePointer<Float>?, UnsafeMutable
Pointer<Float>?, UInt, UnsafeMutableRawPointer?) -> Void)!,
    _ kernelWidth: Float,
    _ userData: UnsafeMutableRawPointer!,
    _ flags: vImage_Flags
) -> vImage_Error
```

Parameters

`filter`

A pointer to a buffer. On return, the buffer contains a resampling filter object. You must allocate the memory for this buffer yourself. Call the function `vImageGetResamplingFilterSize(_:_:_:_:)` to obtain the size of this buffer.

`scale`

A scale factor to associated with the resampling filter object. Shear functions to which you pass the resampling filter object use this factor when performing a shear operation. The shear function applies the scale factor to the entire image, in a direction appropriate to the shear function, either horizontal or vertical.

`kernelFunc`

A pointer to your custom resampling kernel function. If this pointer is `NULL`, vImage creates a resampling filter object that corresponds to its default kernel. The kernel function must remain valid for the life of the resampling filter object.

If you need precise control over the memory used for a resampling filter object but want to use the default, pass `NULL` as the `kernelFunc` parameter. This causes vImage to create a resampling filter object that corresponds to its default kernel. You can then determine where in memory the resampling filter object is located. You can reuse the buffer to reduce memory allocation, and so on. Note that a resampling filter object created in this way isn't necessarily the same as one created by the function `vImageNewResamplingFilter(_:_:)`. You must still deallocate the object yourself; you can't pass it to the function `vImageDestroyResamplingFilter(_:)`.

`kernelWidth`

A bounding value for the domain of your resampling kernel function. The shearing operation passes x-values to your function that are in the range `–kernelWidth` and `+kernelWidth`, inclusive.

`userData`

A pointer to custom data that you want to use when calculating your resampling kernel function. When vImage invokes your resampling kernel function, this pointer is passed to the function. The data can be anything you want to use in calculating your resampling kernel function — a table, a list of pointers to related functions, or so on. The data must remain valid for the life of the resampling kernel object.

If your resampling kernel function does nor require user data, pass `NULL`.

`flags`

The options to use when creating the resampling filter object. You must set exactly one of the following flags to specify how vImage handles pixel locations beyond the edge of the source image: `kvImageBackgroundColorFill` or `kvImageEdgeExtend`.

Set the `kvImageHighQualityResampling` flag if you want vImage to use a higher quality, but slower, resampling filter.

If your code implements its own tiling or its own multithreading, pass `kvImageDoNotTile`.

This function ignores the `kvImageLeaveAlphaUnchanged` flag.

Return Value

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

Discussion

This function creates a reusable resampling filter object that you can pass to a shear function. Before calling this function, you must allocate a buffer to contain the resampling filter object returned by this function. You can get the necessary size by calling the function `vImageGetResamplingFilterSize(_:_:_:_:)`. When you no longer need this object, you're responsible for deallocating its memory. Don't use the function `vImageDestroyResamplingFilter(_:)` to deallocate custom resampling filter objects; it's not designed to deallocate them.

If you need precise control over the memory used for a resampling filter object but want to use the default, pass `NULL` as the `kernelFunc` parameter. This causes vImage to create a resampling filter object that corresponds to its default kernel. You can then determine where in memory the resampling filter object is located. You can reuse the buffer to reduce memory allocation, and so on. Note that a resampling filter object created in this way isn't necessarily the same as one created by the function `vImageNewResamplingFilter(_:_:)`. You must still deallocate the object yourself; you can't pass it to the function `vImageDestroyResamplingFilter(_:)`.

See Also

Resampling filters

```
func vImageNewResamplingFilter(Float, vImage_Flags) -> ResamplingFilter
!
```

Creates a resampling filter object that corresponds to the default kernel supplied by the vImage framework.

```
func vImageGetResamplingFilterExtent(ResamplingFilter, vImage_Flags) ->
vImagePixelCount
```

Returns the maximum sampling radius for a resampling filter.

```
func vImageGetResamplingFilterSize(Float, ((UnsafePointer<Float>?,
UnsafeMutablePointer<Float>?, UInt, UnsafeMutableRawPointer?) -> Void
)!, Float, vImage_Flags) -> Int
```

Returns the minimum size, in bytes, for the buffer needed by the new resampling filter function.

```
func vImageDestroyResamplingFilter(ResamplingFilter!)
```

Disposes of a resampling filter object.