IceMan - Geometric Transforms

ice.Image LinearTransform(*matrix*)

This operation performs all coordinate transformations expressible as a 3x3 matrix. These include affine transformations (in which parallel lines stay parallel), translations and non-affine perspective transformations.

The resampling filter is a Catmull-Rom bicubic, whose width is calculated appropriately.

Parameters

matrix

3x3 transformation matrix (list).

Example

? Unknown Attachment ? Unknown Attachment

In the example below, i is of type ice.Image.

```
matrix = [1, 0.3, 0, 0.1, 1, 0, 0, 0, 1.0]
result = i.LinearTransform(matrix
```

ice.Image Translate(*point, filterType*)

Translate an image with appropriate resampling. Integral pixel translate operations are automatically optimized.

Parameters

point

Translate by a specified number of pixels (tuple)

filter

resampling filter to use (int). Can be one of:

- ice.constants.FILTER_BILINEAR
- ice.constants.FILTER_BSPLINE
- ice.constants.FILTER_CATROM
- ice.constants.FILTER_LANCZOS
- ice.constants.FILTER_MITCHELL_NETRAVALI
- ice.constants.FILTER_POINTSAMPLE

Defaults to FILTER_CATROM (CatmullRom).

Example

? Unknown Attachment ? Unknown Attachment

```
filter = ice.constants.FILTER_BILINEAR
amount = (23.5. 60.6)
result = orig.Translate(amount, filter)
```

This is a special case of *LinearTransform.*

ice.Image Move(*point*)

Returns a copy of the operand image with its world coordinates moved by an integral amount. The copy will share data with the source, no resampling will be performed

Parameters

point

Move by a specified number of pixels (tuple)

ice.Image Scale(*scale, filter*)

Geometrically scale an image with appropriate resampling.

Parameters

scale

Scale by specified factors in x and y (list)

filter

resampling filter to use (int). Defaults to CatmullRom.

Example

? Unknown Attachment ? Unknown Attachment

```
filter = ice.constants.FILTER_LANCZOS
amount = [0.5. 0.5]
result = orig.Scale(amount, filter)
```

This is a special case of *LinearTransform*.

ice.Image Rotate(*degrees, filter*)

Rotate an image about *(0, 0)*. The angle of rotation is measured counterclockwise.

Parameters

degrees

Angle of rotation in degrees (float)

filter

resampling filter to use (int). Defaults to CatmullRom

Example

? Unknown Attachment ? Unknown Attachment

This is a special case of *LinearTransform.*

ice.Image Resize(*scale*)

Fast geometric scaling using only point-sampling. Useful for previews and interactive display.

Parameters

scale

Scale by specified factors in x and y (list)

ice.Image Reformat(newBox, preserveAR, crop, anamorph)

This is an operation that fulfills a common need: that of changing the size of an image prior to saving. The arguments are self-explanatory: letter- or window-boxing is automatically performed if aspect ratio is to be preserved and cropping is not enabled. The *anamorph* argument is unity when no anamorphic display is intended: larger than unity when it is.

⚠ This is not a "native" operation: it internally comprises other operations arranged to yield the result desired.

Parameters

newBox

Desired box (list).

preserveAR

Should the aspect ratio be preserved (bool)? default = true.

crop

Should the image be cropped to preserve aspect ratio (bool)? default = true.

anamorph

If the final image is destined for anamorphic display, the "stretch factor" (float). default = 1.0.

ice.Image Flip(*x, y, transpose*)

This operation is optimized to perform the eight possible "unity-scale" transformations. These are illustrated below.

Parameters

x

Flip in x (bool)

y

Flip in y (bool)

t*ranspose:*

Transpose the image (bool)

Example

? Unknown Attachment

```
# No-op
result = m1_1.Flip(False, False, False)
```

? Unknown Attachment

```
# Transpose axes
result = m1_1.Flip(False, False, True)
```

? Unknown Attachment

```
# Flip in Y
result = m1_1.Flip(False, True, False)
```

? Unknown Attachment

```
# Flip in Y and transpose axes
result = m1_1.Flip(False, True, True)
```

? Unknown Attachment

```
# Flip in X
result = m1_1.Flip(True, False, False)
```

? Unknown Attachment

```
# Flip in X and transpose axes
result = m1_1.Flip(True, False, True)
```

? Unknown Attachment

```
# Flip in X and Y
result = m1_1.Flip(True, True, False)
```

? Unknown Attachment

```
# Flip in X and Y and transpose
result = m1_1.Flip(True, True, True)
```

ice.Image DisplacementWarp(*warpImg, minMax, filterType, filterScale*)

General displacement warp operation. *WarpImg* is a two-channel vector field containing the offset of the source pixel for each result pixel. The actual offset is given by:

```
O' = o * (max  min) + min
```

The type of the filter is normally best set to FILTER_CATROM. Since it is not possible to analytically determine a single best filter width, *filterScale* should be chosen to yield the best possible combination of sharpness and anti-aliasing. 1.0 is a good place to start: smaller values yield wider filters (and less sharpness).

Parameters

warpImg

Displacement image (ice.Image)

minMax

Displacements corresponding to 0 and 1 (list). defaults to [0, 1].

filterType

Filter type. Defaults to CatmullRom.

filterScale

Scale equivalent for filter width (float) defaults to 1.0

Example

Original Image:

? Unknown Attachment

Displacement Image:

? Unknown Attachment

Warped/Resultant Image:

? Unknown Attachment

```
minMax = [0, 10]
filter = ice.constants.FILTER_MITCHELL_NETRAVALI
fScale = 1.0
result = fruit.DisplacementWarp(disp, minMax, filter, fScale)
```