NSAffineTransform Class Reference

Cocoa > Graphics & Imaging

2007-01-15

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

Simultaneously published in the United States and Canada.

Contents

NSAffineTransform Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Drawing Guide
Declared in	NSAffineTransform.h
Related sample code	DockTile SpeedometerView Transformed Image WebKitPluginStarter WebKitPluginWithJavaScript

Overview

The `NSAffineTransform` class provides methods for creating, concatenating, and applying affine transformations.

A transformation specifies how points in one coordinate system are transformed to points in another coordinate system. An affine transformation is a special type of transformation that preserves parallel lines in a path but does not necessarily preserve lengths or angles. Scaling, rotation, and translation are the most commonly used manipulations supported by affine transforms, but shearing is also possible.

Note: In Mac OS X v10.3 and earlier the `NSAffineTransform` class was declared and implemented entirely in the Application Kit framework. As of Mac OS X v10.4 the `NSAffineTransform` class has been split across the Foundation Kit and Application Kit frameworks.

Methods for applying affine transformations to the current graphics context and a method for applying an affine transformation to an `NSBezierPath` object are described in NSAffineTransform Additions in the Application Kit.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Creating an NSAffineTransform Object

`+ transform` (page 7)
Creates and returns a new `NSAffineTransform` object initialized to the identity matrix.

`- initWithTransform:` (page 8)
Initializes the receiver's matrix using another transform object and returns the receiver.

Accumulating Transformations

`- rotateByDegrees:` (page 9)
Applies a rotation factor (measured in degrees) to the receiver's transformation matrix.

`- rotateByRadians:` (page 10)
Applies a rotation factor (measured in radians) to the receiver's transformation matrix.

`- scaleBy:` (page 11)
Applies the specified scaling factor along both x and y axes to the receiver's transformation matrix.

`- scaleXBy:yBy:` (page 11)
Applies scaling factors to each axis of the receiver's transformation matrix.

`- translateXBy:yBy:` (page 14)
Applies the specified translation factors to the receiver's transformation matrix.

`- appendTransform:` (page 7)
Appends the specified matrix to the receiver's matrix.

`- prependTransform:` (page 9)
Prepends the specified matrix to the receiver's matrix.

`- invert` (page 8)
Replaces the receiver's matrix with its inverse matrix.

Transforming Data and Objects

`- transformPoint:` (page 12)
Applies the receiver's transform to the specified `NSPoint` data type and returns the results.

- `transformSize:` (page 13)
  Applies the receiver's transform to the specified `NSSize` data type and returns the results.

Accessing the Transformation Structure

- `transformStruct` (page 13)
  Returns the matrix coefficients stored in the receiver's matrix.
- `setTransformStruct:` (page 12)
  Replaces the receiver's transformation matrix with the specified values.

Class Methods

transform

Creates and returns a new `NSAffineTransform` object initialized to the identity matrix.

```
+ (NSAffineTransform *)transform
```

Return Value
A new identity transform object. This matrix transforms any point to the same point.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithTransform:` (page 8)

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSAffineTransform.h`

Instance Methods

appendTransform:

Appends the specified matrix to the receiver's matrix.

```
- (void)appendTransform:(NSAffineTransform *)aTransform
```

Parameters

aTransform

The matrix to append to the receiver.

Discussion

This method multiplies the receiver's matrix by the matrix in *aTransform* and replaces the receiver's matrix with the results. This type of operation is the same as applying the transformations in the receiver followed by the transformations in *aTransform*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- prependTransform: (page 9)

Declared In

NSAffineTransform.h

initWithTransform:

Initializes the receiver's matrix using another transform object and returns the receiver.

```
- (id)initWithTransform:(NSAffineTransform *)aTransform
```

Parameters

aTransform

The transform object whose matrix values should be copied to this object.

Return Value

A new transform object initialized with the matrix values of *aTransform*.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ transform (page 7)

Related Sample Code

DockTile

SpeedometerView

WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In

NSAffineTransform.h

invert

Replaces the receiver's matrix with its inverse matrix.

```
- (void)invert
```

Discussion
Inverse matrices are useful for undoing the effects of a matrix. If a previous point (x,y) was transformed to (x',y'), inverting the matrix and applying it to point (x',y') yields the point (x,y).

You can also use inverse matrices in conjunction with the `concat` method to remove the effects of concatenating the matrix to the current transformation matrix of the current graphic context.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSAffineTransform.h`

prependTransform:

Prepends the specified matrix to the receiver's matrix.

```
- (void)prependTransform:(NSAffineTransform *)aTransform
```

Parameters

aTransform
The matrix to prepend to the receiver.

Discussion
This method multiplies the matrix in *aTransform* by the receiver's matrix and replaces the receiver's matrix with the result. This type of operation is the same as applying the transformations in *aTransform* followed by the transformations in the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- appendTransform: (page 7)

Declared In
`NSAffineTransform.h`

rotateByDegrees:

Applies a rotation factor (measured in degrees) to the receiver's transformation matrix.

```
- (void)rotateByDegrees:(CGFloat)angle
```

Parameters

angle
The rotation angle, measured in degrees.

Discussion
After invoking this method, applying the receiver's matrix turns the axes counterclockwise about the current origin by *angle* degrees, in addition to performing all previous transformations.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rotateByRadians:` (page 10)
- `scaleBy:` (page 11)
- `scaleXBy:yBy:` (page 11)
- `translateXBy:yBy:` (page 14)

Related Sample Code
DockTile
PDF Annotation Editor
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSAffineTransform.h`

rotateByRadians:

Applies a rotation factor (measured in radians) to the receiver's transformation matrix.

```
- (void)rotateByRadians:(CGFloat)angle
```

Parameters
angle
The rotation angle, measured in radians.

Discussion
After invoking this method, applying the receiver's matrix turns the axes counterclockwise about the current origin by *angle* radians, in addition to performing all previous transformations.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rotateByDegrees:` (page 9)
- `scaleBy:` (page 11)
- `scaleXBy:yBy:` (page 11)
- `translateXBy:yBy:` (page 14)

Related Sample Code
Polygons
TextLayoutDemo

Declared In
`NSAffineTransform.h`

scaleBy:

Applies the specified scaling factor along both x and y axes to the receiver's transformation matrix.

```
- (void)scaleBy:(CGFloat)scale
```

Parameters

scale

The scaling factor to apply to both axes. Specifying a negative value has the effect of inverting the direction of the axes in addition to scaling them. A scaling factor of 1.0 scales the content to exactly the same size.

Discussion

After invoking this method, applying the receiver's matrix modifies the unit lengths along the current x and y axes by a factor of *scale*, in addition to performing all previous transformations.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rotateByDegrees: (page 9)
- rotateByRadians: (page 10)
- scaleXBy:yBy: (page 11)
- translateXBy:yBy: (page 14)

Related Sample Code

Aperture Edit Plugin - Borders & Titles

CIAnnotation

Polygons

Transformed Image

Declared In

NSAffineTransform.h

scaleXBy:yBy:

Applies scaling factors to each axis of the receiver's transformation matrix.

```
- (void)scaleXBy:(CGFloat)scaleX yBy:(CGFloat)scaleY
```

Parameters

scaleX

The scaling factor to apply to the x axis.

scaleY

The scaling factor to apply to the y axis.

Discussion

After invoking this method, applying the receiver's matrix modifies the unit length on the x axis by a factor of *scaleX* and the y axis by a factor of *scaleY*, in addition to performing all previous transformations. A value of 1.0 for either axis scales the content on that axis to the same size.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `rotateByDegrees:` (page 9)
- `rotateByRadians:` (page 10)
- `scaleBy:` (page 11)
- `translateXBy:yBy:` (page 14)

Related Sample Code
Sketch-112

Declared In
`NSAffineTransform.h`

setTransformStruct:

Replaces the receiver's transformation matrix with the specified values.

```
- (void)setTransformStruct:(NSAffineTransformStruct)aTransformStruct
```

Parameters

aTransformStruct

The structure containing the six transform values you want the receiver to use.

Discussion
The matrix is of the form shown in "Manipulating Transform Values", and the six-element structure defined by the `NSAffineTransformStruct` structure is of the form:

```
{m11, m12, m21, m22, tX, tY}
```

The `NSAffineTransformStruct` structure is an alternate representation of a transformation matrix that can be used to specify matrix values directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithTransform:` (page 8)
- `transformStruct` (page 13)

Related Sample Code
Transformed Image

Declared In
`NSAffineTransform.h`

transformPoint:

Applies the receiver's transform to the specified `NSPoint` data type and returns the results.

```
- (NSPoint)transformPoint:(NSPoint)aPoint
```

Parameters

aPoint

The point in the current coordinate system to which you want to apply the matrix.

Return Value

The resulting point after applying the receiver's transformations.

Availability

Available in Mac OS X v10.0 and later.

See Also

- transformSize: (page 13)

Declared In

NSAffineTransform.h

transformSize:

Applies the receiver's transform to the specified NSSize data type and returns the results.

```
- (NSSize)transformSize:(NSSize)aSize
```

Parameters

aSize

The size data to which you want to apply the matrix.

Return Value

The resulting size after applying the receiver's transformations.

Discussion

This method applies the current rotation and scaling factors to *aSize*; it does not apply translation factors. You can think of this method as transforming a vector whose origin is (0, 0) and whose end point is specified by the value in *aSize*. After the rotation and scaling factors are applied, this method effectively returns the end point of the new vector.

This method is useful for transforming delta or distance values when you need to take scaling and rotation factors into account.

Availability

Available in Mac OS X v10.0 and later.

See Also

- transformPoint: (page 12)

Declared In

NSAffineTransform.h

transformStruct

Returns the matrix coefficients stored in the receiver's matrix.

```
- (NSAffineTransformStruct)transformStruct
```

Return Value
The structure containing the receiver's six matrix values.

Discussion
The matrix is of the form shown in "Manipulating Transform Values", and the six-element structure defined by the `NSAffineTransformStruct` structure is of the form:

```
{m11, m12, m21, m22, tX, tY}
```

The `NSAffineTransformStruct` structure is an alternate representation of a transformation matrix that can be used to specify matrix values directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithTransform:` (page 8)
- `setTransformStruct:` (page 12)

Related Sample Code
Transformed Image

Declared In
`NSAffineTransform.h`

translateXBy:yBy:

Applies the specified translation factors to the receiver's transformation matrix.

```
- (void)translateXBy:(CGFloat)deltaX yBy:(CGFloat)deltaY
```

Parameters

deltaX
The number of units to move along the x axis.

deltaY
The number of units to move along the y axis.

Discussion
Subsequent transformations cause coordinates to be shifted by *deltaX* units along the x axis and by *deltaY* units along the y axis. Translation factors do not affect `NSSize` values, which specify a differential between points.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rotateByDegrees:` (page 9)
- `rotateByRadians:` (page 10)
- `scaleBy:` (page 11)
- `scaleXBy:yBy:` (page 11)

Related Sample Code
DockTile

PDF Annotation Editor
Sketch-112
SpeedometerView
Squiggles

Declared In
`NSAffineTransform.h`

Constants

NSAffineTransformStruct

This type defines the three-by-three matrix that performs an affine transform between two coordinate systems.

```
typedef struct _NSAffineTransformStruct {
    float m11, m12, m21, m22;
    float tX, tY;
} NSAffineTransformStruct;
```

Fields

`m11 , m12, m21, m22`
Elements of a two-by-two matrix for rotation, scale, and shear transformations.

`tX, tY`
x and y translation elements

Discussion
For more details, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSAffineTransform.h`

Document Revision History

This table describes the changes to *NSAffineTransform Class Reference*.

Date	Notes
2007-01-15	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index