NSBezierPath Class Reference

Cocoa > Graphics & Imaging

2007-03-02

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

Simultaneously published in the United States and Canada.

Contents

Figures

NSBezierPath Class Reference 9

NSBezierPath Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Drawing Guide
Declared in	NSBezierPath.h
Related sample code	Dicey DockTile Sketch-112 SpeedometerView WebKitPluginStarter

Overview

An `NSBezierPath` object allows you to create paths using PostScript-style commands. Paths consist of straight and curved line segments joined together. Paths can form recognizable shapes such as rectangles, ovals, arcs, and glyphs; they can also form complex polygons using either straight or curved line segments. A single path can be closed by connecting its two endpoints, or it can be left open.

An `NSBezierPath` object can contain multiple disconnected paths, whether they are closed or open. Each of these paths is referred to as a subpath. The subpaths of an `NSBezierPath` object must be manipulated as a group. The only way to manipulate subpaths individually is to create separate `NSBezierPath` objects for each.

For a given `NSBezierPath` object, you can stroke the path's outline or fill the region occupied by the path. You can also use the path as a clipping region for views or other regions. Using methods of `NSBezierPath`, you can also perform hit detection on the filled or stroked path. Hit detection is needed to implement interactive graphics, as in rubberbanding and dragging operations.

The current graphics context is automatically saved and restored for all drawing operations involving `NSBezierPath` objects, so your application does not need to worry about the graphics settings changing across invocations.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Creating an NSBezierPath Object

`+ bezierPath` (page 14)
Creates and returns a new `NSBezierPath` object.

`+ bezierPathWithOvalInRect:` (page 15)
Creates and returns a new `NSBezierPath` object initialized with an oval path inscribed in the specified rectangle.

`+ bezierPathWithRect:` (page 15)
Creates and returns a new `NSBezierPath` object initialized with a rectangular path.

`+ bezierPathWithRoundedRect:xRadius:yRadius:` (page 16)
Creates and returns a new `NSBezierPath` object initialized with a rounded rectangular path.

`- bezierPathByFlatteningPath` (page 33)
Creates and returns a “flattened” copy of the receiver.

`- bezierPathByReversingPath` (page 34)
Creates and returns a new `NSBezierPath` object with the reversed contents of the receiver’s path.

Constructing Paths

`- moveToPoint:` (page 44)
Moves the receiver’s current point to the specified location.

`- lineToPoint:` (page 42)
Appends a straight line to the receiver’s path

`- curveToPoint:controlPoint1:controlPoint2:` (page 37)
Adds a Bezier cubic curve to the receiver’s path.

`- closePath` (page 35)
Closes the most recently added subpath.

`- relativeMoveToPoint:` (page 46)
Moves the receiver’s current point to a new point whose location is the specified distance from the current point.

- `relativeLineToPoint:` (page 45)
  Appends a straight line segment to the receiver's path starting at the current point and moving towards the specified point, relative to the current location.
- `relativeCurveToPoint:controlPoint1:controlPoint2:` (page 44)
  Adds a Bezier cubic curve to the receiver's path from the current point to a new location, which is specified as a relative distance from the current point.

Appending Common Shapes to a Path

- `appendBezierPath:` (page 27)
  Appends the contents of the specified path object to the receiver's path.
- `appendBezierPathWithPoints:count:` (page 32)
  Appends a series of line segments to the receiver's path.
- `appendBezierPathWithOvalInRect:` (page 31)
  Appends an oval path to the receiver, inscribing the oval in the specified rectangle.
- `appendBezierPathWithArcFromPoint:toPoint:radius:` (page 27)
  Appends an arc to the receiver's path.
- `appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:` (page 28)
  Appends an arc of a circle to the receiver's path.
- `appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:clockwise:` (page 29)
  Appends an arc of a circle to the receiver's path.
- `appendBezierPathWithGlyph:inFont:` (page 29)
  Appends an outline of the specified glyph to the receiver's path.
- `appendBezierPathWithGlyphs:count:inFont:` (page 30)
  Appends the outlines of the specified glyphs to the receiver's path.
- `appendBezierPathWithPackedGlyphs:` (page 31)
  Appends an array of packed glyphs to the receiver's path.
- `appendBezierPathWithRect:` (page 32)
  Appends a rectangular path to the receiver's path.
- `appendBezierPathWithRoundedRect:xRadius:yRadius:` (page 33)
  Appends a rounded rectangular path to the receiver's path.

Accessing Path Attributes

+ `defaultWindingRule` (page 19)
  Returns the default winding rule used to fill all paths.
+ `setDefaultWindingRule:` (page 24)
  Sets the default winding rule used to fill all paths.
- `windingRule` (page 54)
  Returns the winding rule used to fill the receiver's path.
- `setWindingRule:` (page 52)
  Sets the winding rule used to fill the receiver's path.
+ `defaultLineCapStyle` (page 18)
  Returns the default line cap style for all paths.

+ `setDefaultLineCapStyle:` (page 21)
  Sets the default line cap style for all paths.

- `lineCapStyle` (page 41)
  Returns the line cap style for the receiver's path.

- `setLineCapStyle:` (page 49)
  Sets the line cap style for the receiver's path.

+ `defaultLineJoinStyle` (page 18)
  Returns the default line join style for all paths.

+ `setDefaultLineJoinStyle:` (page 22)
  Sets the default line join style for all paths.

- `lineJoinStyle` (page 42)
  Returns the receiver’s line join style.

- `setLineJoinStyle:` (page 50)
  Sets the line join style for the receiver's path.

+ `defaultLineWidth` (page 19)
  Returns the default line width for the all paths.

+ `setDefaultLineWidth:` (page 23)
  Sets the default line width for all paths.

- `lineWidth` (page 43)
  Returns the line width of the receiver's path.

- `setLineWidth:` (page 51)
  Sets the line width of the receiver's path.

+ `defaultMiterLimit` (page 19)
  Returns the default miter limit for all paths.

+ `setDefaultMiterLimit:` (page 24)
  Sets the default miter limit for all paths.

- `miterLimit` (page 43)
  Returns the miter limit of the receiver's path.

- `setMiterLimit:` (page 51)
  Sets the miter limit for the receiver's path.

+ `defaultFlatness` (page 17)
  Returns the default flatness value for all paths.

+ `setDefaultFlatness:` (page 21)
  Sets the default flatness value for all paths.

- `flatness` (page 40)
  Returns the flatness value of the receiver's path.

- `setFlatness:` (page 48)
  Sets the flatness value for the receiver's path.

- `getLineDash:count:phase:` (page 40)
  Returns the line-stroking pattern for the receiver.

- `setLineDash:count:phase:` (page 49)
  Sets the line-stroking pattern for the receiver.

Drawing Paths

- `stroke` (page 52)
  Draws a line along the receiver's path using the current stroke color and drawing attributes.
- `fill` (page 39)
  Paints the region enclosed by the receiver's path.
+ `fillRect:` (page 20)
  Fills the specified rectangular path with the current fill color.
+ `strokeRect:` (page 26)
  Strokes the path of the specified rectangle using the current stroke color and the default drawing attributes.
+ `strokeLineFromPoint:toPoint:` (page 25)
  Strokes a line between two points using the current stroke color and the default drawing attributes.
+ `drawPackedGlyphs:atPoint:` (page 20)
  Draws a set of packed glyphs at the specified point in the current coordinate system.

Clipping Paths

- `addClip` (page 26)
  Intersects the area enclosed by the receiver's path with the clipping path of the current graphics context and makes the resulting shape the current clipping path.
- `setClip` (page 48)
  Replaces the clipping path of the current graphics context with the area inside the receiver's path.
+ `clipRect:` (page 17)
  Intersects the specified rectangle with the clipping path of the current graphics context and makes the resulting shape the current clipping path

Hit Detection

- `containsPoint:` (page 36)
  Returns a Boolean value indicating whether the receiver contains the specified point.

Querying Paths

- `bounds` (page 34)
  Returns the bounding box of the receiver's path.
- `controlPointBounds` (page 36)
  Returns the bounding box of the receiver's path, including any control points.
- `currentPoint` (page 37)
  Returns the receiver's current point (the trailing point or ending point in the most recently added segment).
- `isEmpty` (page 41)
  Returns a Boolean value indicating whether the receiver is empty.

Applying Transformations

- `transformUsingAffineTransform:` (page 53)
  Transforms all points in the receiver using the specified transform.

Accessing Elements of a Path

- `elementCount` (page 39)
  Returns the total number of path elements in the receiver's path.
- `elementAtIndex:` (page 38)
  Returns the type of path element at the specified index.
- `elementAtIndex:associatedPoints:` (page 38)
  Gets the element type and (and optionally) the associated points for the path element at the specified index.
- `removeAllPoints` (page 46)
  Removes all path elements from the receiver, effectively clearing the path.
- `setAssociatedPoints:atIndex:` (page 47)
  Changes the points associated with the specified path element.

Caching Paths

- `cachesBezierPath` (page 35)
  Returns a Boolean value indicating whether this object maintains a cached image of its path.
- `setCachesBezierPath:` (page 47)
  Sets whether the receiver should cache its path information.

Class Methods

bezierPath

Creates and returns a new `NSBezierPath` object.

```
+ (NSBezierPath *)bezierPath
```

Return Value
A new empty path object.

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

Related Sample Code
Dicey
DockTile
Sketch-112
SpeedometerView

WebKitPluginStarter

Declared In
NSBezierPath.h

bezierPathWithOvalInRect:

Creates and returns a new NSBezierPath object initialized with an oval path inscribed in the specified rectangle.

```
+ (NSBezierPath *)bezierPathWithOvalInRect:(NSRect)aRect
```

Parameters

aRect

The rectangle in which to inscribe an oval.

Return Value
ANSBezierPath new path object with the oval path.

Discussion
If the *aRect* parameter specifies a square, the inscribed path is a circle. The path is constructed by starting in the lower-right quadrant of the rectangle and adding arc segments counterclockwise to complete the oval.

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

See Also
+ bezierPath (page 14)
- appendBezierPathWithOvalInRect: (page 31)

Related Sample Code
BindingsJoystick
Dicey
MenuItemView
Sketch-112
Worm

Declared In
NSBezierPath.h

bezierPathWithRect:

Creates and returns a new NSBezierPath object initialized with a rectangular path.

```
+ (NSBezierPath *)bezierPathWithRect:(NSRect)aRect
```

Parameters

aRect

The rectangle describing the path to create.

Return Value
A new path object with the rectangular path.

Discussion
The path is constructed by starting at the origin of *aRect* and adding line segments in a counterclockwise direction.

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

See Also
+ bezierPath (page 14)
- appendBezierPathWithRect: (page 32)
+ fillRect: (page 20)
+ strokeRect: (page 26)

Related Sample Code
Cropped Image
Link Snoop
PDF Annotation Editor
PDFKitLinker2
Sketch-112

Declared In
NSBezierPath.h

bezierPathWithRoundedRect:xRadius:yRadius:

Creates and returns a new NSBezierPath object initialized with a rounded rectangular path.

```
+ (NSBezierPath *)bezierPathWithRoundedRect:(NSRect)rect xRadius:(CGFloat)xRadius
     yRadius:(CGFloat)yRadius
```

Parameters

rect
The rectangle that defines the basic shape of the path.

xRadius
The radius of each corner oval along the x-axis. Values larger than half the rectangle's width are clamped to half the width.

yRadius
The radius of each corner oval along the y-axis. Values larger than half the rectangle's height are clamped to half the height.

Return Value
A new path object with the rounded rectangular path.

Discussion
The path is constructed in a counter-clockwise direction, starting at the top-left corner of the rectangle. If either one of the radius parameters contains the value 0.0, the returned path is a plain rectangle without rounded corners.

Availability
Available in Mac OS X v10.5 and later.

See Also
+ bezierPath (page 14)
- appendBezierPathWithRoundedRect:xRadius:yRadius: (page 33)

Related Sample Code
TrackBall

Declared In
NSBezierPath.h

clipRect:

Intersects the specified rectangle with the clipping path of the current graphics context and makes the resulting shape the current clipping path

```
+ (void)clipRect:(NSRect)aRect
```

Parameters

aRect
The rectangle to intersect with the current clipping path.

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

See Also
- addClip (page 26)
- setClip (page 48)

Related Sample Code
Sketch-112
Transformed Image

Declared In
NSBezierPath.h

defaultFlatness

Returns the default flatness value for all paths.

```
+ (CGFloat)defaultFlatness
```

Return Value
The default value for determining the smoothness of curved paths, or 0.6 if no other value has been set.

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

See Also
+ setDefaultFlatness: (page 21)
- flatness (page 40)

Declared In
NSBezierPath.h

defaultLineCapStyle

Returns the default line cap style for all paths.

```
+ (NSLineCapStyle)defaultLineCapStyle
```

Return Value
The default line cap style or NSButtLineCapStyle if no other style has been set. For a list of values, see "Constants" (page 54).

Discussion
The default line cap style can be overridden for individual paths by setting a custom style for that path using the setLineCapStyle: (page 49) method.

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

See Also
+ setDefaultLineCapStyle: (page 21)
+ defaultLineJoinStyle (page 18)
+ defaultLineWidth (page 19)
- lineCapStyle (page 41)

Declared In
NSBezierPath.h

defaultLineJoinStyle

Returns the default line join style for all paths.

```
+ (NSLineJoinStyle)defaultLineJoinStyle
```

Return Value
The default line join style or NSMiterLineJoinStyle if no other value has been set. For a list of values, see "Constants" (page 54).

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

See Also
+ setDefaultLineJoinStyle: (page 22)
+ defaultLineCapStyle (page 18)
+ defaultLineWidth (page 19)
- lineJoinStyle (page 42)

Declared In
NSBezierPath.h

defaultLineWidth

Returns the default line width for the all paths.

```
+ (CGFloat)defaultLineWidth
```

Return Value
The default line width, measured in points in the user coordinate space, or 1.0 if no other value has been set.

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

See Also
+ `setDefaultLineWidth:` (page 23)
+ `defaultLineCapStyle` (page 18)
+ `defaultLineJoinStyle` (page 18)
- `lineWidth` (page 43)

Declared In
`NSBezierPath.h`

defaultMiterLimit

Returns the default miter limit for all paths.

```
+ (CGFloat)defaultMiterLimit
```

Return Value
The default miter limit for all paths, or 10.0 if no other value has been set.

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

See Also
+ `setDefaultMiterLimit:` (page 24)
- `miterLimit` (page 43)

Declared In
`NSBezierPath.h`

defaultWindingRule

Returns the default winding rule used to fill all paths.

```
+ (NSWindingRule)defaultWindingRule
```

Return Value
The current default winding rule or `NSNonZeroWindingRule` if no default rule has been set. This value may be either `NSNonZeroWindingRule` or `NSEvenOddWindingRule`.

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

See Also
+ setDefaultWindingRule: (page 24)
- windingRule (page 54)

Declared In
NSBezierPath.h

drawPackedGlyphs:atPoint:

Draws a set of packed glyphs at the specified point in the current coordinate system.

```
+ (void)drawPackedGlyphs:(const char *)packedGlyphs atPoint:(NSPoint)aPoint
```

Parameters

packedGlyphs

A C-style array containing one or more CGGlyph data types terminated by a NULL character.

aPoint

The starting point at which to draw the glyphs.

Discussion
This method draws the glyphs immediately.

You should avoid using this method directly. Instead, use the appendBezierPathWithGlyph:inFont: (page 29) and appendBezierPathWithGlyphs:count:inFont: (page 30) methods to create a path with one or more glyphs.

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

See Also
- appendBezierPathWithPackedGlyphs: (page 31)
- set (NSColor)

Declared In
NSBezierPath.h

fillRect:

Fills the specified rectangular path with the current fill color.

```
+ (void)fillRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the current coordinate system.

Discussion
This method fills the specified region immediately. This method uses the compositing operation returned by the compositingOperation method of NSGraphicsContext.

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

See Also
- `- appendBezierPathWithRect:` (page 32)
- `+ bezierPathWithRect:` (page 15)
- `+ strokeRect:` (page 26)
- `- compositingOperation` (NSGraphicsContext)
- `- set` (NSColor)

Related Sample Code
DragItemAround
JSPong
QTSSConnectionMonitor
ThreadsImportMovie
WhackedTV

Declared In
`NSBezierPath.h`

setDefaultFlatness:

Sets the default flatness value for all paths.

```
+ (void)setDefaultFlatness:(CGFloat)flatness
```

Parameters
flatness
The default flatness value.

Discussion
The flatness value specifies the accuracy (or smoothness) with which curves are rendered. It is also the maximum error tolerance (measured in pixels) for rendering curves, where smaller numbers give smoother curves at the expense of more computation. The exact interpretation may vary slightly on different rendering devices.

The default flatness value is 0.6, which yields smooth curves.

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

See Also
- `+ defaultFlatness` (page 17)
- `- setFlatness:` (page 48)

Declared In
`NSBezierPath.h`

setDefaultLineCapStyle:

Sets the default line cap style for all paths.

```
+ (void)setDefaultLineCapStyle:(NSLineCapStyle)lineCap
```

Parameters

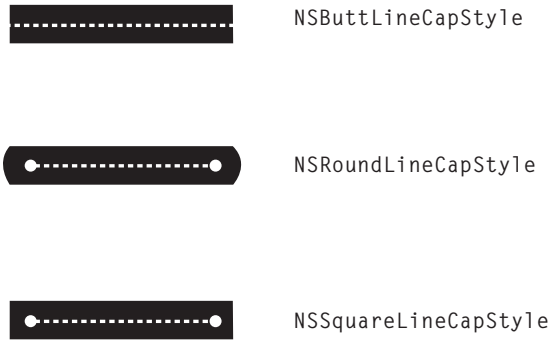
lineCap

The default line cap style. For a list of values, see "Constants" (page 54).

Discussion

The line cap style specifies the shape of the endpoints of an open path when stroked. Figure 1 (page 22) shows the appearance of the available line cap styles.

Figure 1 Line cap styles



Availability

Available in Mac OS X v10.0 and later.

See Also

`+ defaultLineCapStyle` (page 18)

`+ setDefaultLineJoinStyle:` (page 22)

`+ setDefaultLineWidth:` (page 23)

`- setLineCapStyle:` (page 49)

Declared In

`NSBezierPath.h`

setDefaultLineJoinStyle:

Sets the default line join style for all paths.

```
+ (void)setDefaultLineJoinStyle:(NSLineJoinStyle)lineJoinStyle
```

Parameters

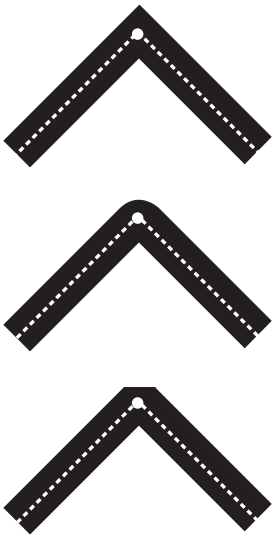
lineJoinStyle

The default line join style. For a list of values, see "Constants" (page 54).

Discussion

The line join style specifies the shape of the joints between connected segments of a stroked path. Figure 2 (page 23) shows the appearance of the available line join styles.

Figure 2 Line join styles



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

See Also
+ defaultLineJoinStyle (page 18)
+ setDefaultLineCapStyle: (page 21)
+ setDefaultLineWidth: (page 23)
+ setDefaultMiterLimit: (page 24)
- setLineJoinStyle: (page 50)

Declared In
NSBezierPath.h

setDefaultLineWidth:

Sets the default line width for all paths.

```
+ (void)setDefaultLineWidth:(CGFloat)width
```

Parameters

width

The default line width, measured in points in the user coordinate space.

Discussion
The line width defines the thickness of stroked paths. A width of 0 is interpreted as the thinnest line that can be rendered on a particular device. The actual rendered line width may vary from the specified width by as much as 2 device pixels, depending on the position of the line with respect to the pixel grid and the current anti-aliasing settings. The width of the line may also be affected by scaling factors specified in the current transformation matrix of the active graphics context.

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

See Also
+ defaultLineWidth (page 19)
+ setDefaultLineCapStyle: (page 21)
+ setDefaultLineJoinStyle: (page 22)
- setLineWidth: (page 51)

Related Sample Code
Clock Control
CocoaDragAndDrop

Declared In
NSBezierPath.h

setDefaultMiterLimit:

Sets the default miter limit for all paths.

```
+ (void)setDefaultMiterLimit:(CGFloat)limit
```

Parameters

limit

The default limit at which miter joins are converted to bevel joins.

Discussion
The miter limit helps you avoid spikes at the junction of two line segments connected by a miter join (NSMiterLineJoinStyle). If the ratio of the miter length—the diagonal length of the miter join—to the line thickness exceeds the miter limit, the joint is converted to a bevel join. The default miter limit value is 10, which converts miters whose angle at the joint is less than 11 degrees.

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

See Also
+ defaultMiterLimit (page 19)
+ setDefaultLineJoinStyle: (page 22)
- setMiterLimit: (page 51)

Declared In
NSBezierPath.h

setDefaultWindingRule:

Sets the default winding rule used to fill all paths.

```
+ (void)setDefaultWindingRule:(NSWindingRule)windingRule
```

Parameters

windingRule

The winding rule to use if no winding rule is set explicitly for a path object. This value may be either NSNonZeroWindingRule or NSEvenOddWindingRule.

Discussion
Winding rules determine how to paint (or fill) the region enclosed by a path. You use this method to set the default rule that is applied to paths that do not have a custom winding rule assigned.

For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

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

See Also
`+ defaultWindingRule` (page 19)
`- setWindingRule:` (page 52)

Declared In
`NSBezierPath.h`

strokeLineFromPoint:toPoint:

Strokes a line between two points using the current stroke color and the default drawing attributes.

```
+ (void)strokeLineFromPoint:(NSPoint)point1 toPoint:(NSPoint)point2
```

Parameters
point1
The starting point of the line.
point2
The ending point of the line.

Discussion
This method strokes the specified path immediately.

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

See Also
`- lineToPoint:` (page 42)
`- moveToPoint:` (page 44)
`+ setDefaultLineCapStyle:` (page 21)
`+ setDefaultLineWidth:` (page 23)
`- stroke` (page 52)

Related Sample Code
BindingsJoystick
Clock Control
FilterDemo
GLChildWindowDemo
WhackedTV

Declared In
`NSBezierPath.h`

strokeRect:

Strokes the path of the specified rectangle using the current stroke color and the default drawing attributes.

```
+ (void)strokeRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the current coordinate system.

Discussion

The path is drawn beginning at the rectangle's origin and proceeding in a counterclockwise direction. This method strokes the specified path immediately.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithRect:` (page 32)
+ `bezierPathWithRect:` (page 15)
+ `fillRect:` (page 20)
+ `setDefaultLineJoinStyle:` (page 22)
+ `setDefaultLineWidth:` (page 23)
- `set` (NSColor)

Related Sample Code

CocoaDragAndDrop

Declared In

`NSBezierPath.h`

Instance Methods

addClip

Intersects the area enclosed by the receiver's path with the clipping path of the current graphics context and makes the resulting shape the current clipping path.

```
- (void)addClip
```

Discussion

This method uses the current winding rule to determine the clipping shape of the receiver. This method does not affect the receiver's path.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ `clipRect:` (page 17)
- `setClip` (page 48)

Related Sample Code
BindingsJoystick
IBFragmentView
Reducer
WebKitDOMElementPlugIn

Declared In
`NSBezierPath.h`

appendBezierPath:

Appends the contents of the specified path object to the receiver's path.

```
- (void)appendBezierPath:(NSBezierPath *)aPath
```

Parameters

aPath
The path to add to the receiver.

Discussion
This method adds the commands used to create *aPath* to the end of the receiver's path. This method does not explicitly try to connect the subpaths in the two objects, although the operations in *aPath* may still cause that effect.

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

Declared In
`NSBezierPath.h`

appendBezierPathWithArcFromPoint:toPoint:radius:

Appends an arc to the receiver's path.

```
- (void)appendBezierPathWithArcFromPoint:(NSPoint)fromPoint toPoint:(NSPoint)toPoint
    radius:(CGFloat)radius
```

Parameters

fromPoint
The middle point of the angle.

toPoint
The end point of the angle.

radius
The radius of the circle inscribed in the angle.

Discussion
The created arc is defined by a circle inscribed inside the angle specified by three points: the current point, the `fromPoint` parameter, and the `toPoint` parameter (in that order). The arc itself lies on the perimeter of the circle, whose radius is specified by the `radius` parameter. The arc is drawn between the two points of the circle that are tangent to the two legs of the angle.

The arc usually does not contain the points in the *fromPoint* and *toPoint* parameters. If the starting point of the arc does not coincide with the current point, a line is drawn between the two points. The starting point of the arc lies on the line defined by the current point and the *fromPoint* parameter.

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

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

Related Sample Code
IBFragmentView

Declared In
`NSBezierPath.h`

appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:

Appends an arc of a circle to the receiver's path.

```
- (void)appendBezierPathWithArcWithCenter:(NSPoint)center radius:(CGFloat)radius
    startAngle:(CGFloat)startAngle endAngle:(CGFloat)endAngle
```

Parameters

center
Specifies the center point of the circle used to define the arc.

radius
Specifies the radius of the circle used to define the arc.

startAngle
Specifies the starting angle of the arc, measured in degrees counterclockwise from the x-axis.

endAngle
Specifies the end angle of the arc, measured in degrees counterclockwise from the x-axis.

Discussion
The created arc lies on the perimeter of the circle, between the angles specified by the *startAngle* and *endAngle* parameters. The arc is drawn in a counterclockwise direction. If the receiver's path is empty, this method sets the current point to the beginning of the arc before adding the arc segment. If the receiver's path is not empty, a line is drawn from the current point to the starting point of the arc.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

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

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:clockwise:

Appends an arc of a circle to the receiver's path.

```
- (void)appendBezierPathWithArcWithCenter:(NSPoint)center radius:(CGFloat)radius
    startAngle:(CGFloat)startAngle endAngle:(CGFloat)endAngle
    clockwise:(BOOL)clockwise
```

Parameters

center
Specifies the center point of the circle used to define the arc.

radius
Specifies the radius of the circle used to define the arc.

startAngle
Specifies the starting angle of the arc, measured in degrees counterclockwise from the x-axis.

endAngle
Specifies the end angle of the arc, measured in degrees counterclockwise from the x-axis.

clockwise
YES if you want the arc to be drawn in a clockwise direction; otherwise NO to draw the arc in a counterclockwise direction.

Discussion
The created arc lies on the perimeter of the circle, between the angles specified by the *startAngle* and *endAngle* parameters. The arc is drawn in the direction indicated by the *clockwise* parameter. If the receiver's path is empty, this method sets the current point to the beginning of the arc before adding the arc segment. If the receiver's path is not empty, a line is drawn from the current point to the starting point of the arc.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

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

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithGlyph:inFont:

Appends an outline of the specified glyph to the receiver's path.

```
- (void)appendBezierPathWithGlyph:(NSGlyph)aGlyph inFont:(NSFont *)fontObj
```

Parameters

aGlyph

The glyph to add to the path.

fontObj

The font in which the glyph is encoded.

Discussion

If the glyph is not encoded in the font specified by the *fontObj* parameter—that is, the font does not have an entry for the specified glyph—then no path is appended to the receiver.

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithGlyphs:count:inFont:` (page 30)
- `appendBezierPathWithPackedGlyphs:` (page 31)
+ `drawPackedGlyphs:atPoint:` (page 20)

Declared In

`NSBezierPath.h`

appendBezierPathWithGlyphs:count:inFont:

Appends the outlines of the specified glyphs to the receiver's path.

```
- (void)appendBezierPathWithGlyphs:(NSGlyph *)glyphs count:(NSInteger)count
    inFont:(NSFont *)fontObj
```

Parameters

glyphs

A C-style array of `NSGlyph` data types to add to the path.

count

The number of glyphs in the *glyphs* parameter.

fontObj

The font in which the glyphs are encoded.

Discussion

If the glyphs are not encoded in the font specified by the *fontObj* parameter—that is, the font does not have an entry for one of the specified glyphs—then no path is appended to the receiver.

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithGlyph:inFont:` (page 29)

- appendBezierPathWithPackedGlyphs: (page 31)
+ drawPackedGlyphs:atPoint: (page 20)

Declared In
NSBezierPath.h

appendBezierPathWithOvalInRect:

Appends an oval path to the receiver, inscribing the oval in the specified rectangle.

```
- (void)appendBezierPathWithOvalInRect:(NSRect)aRect
```

Parameters

aRect

The rectangle in which to inscribe the oval.

Discussion
Before adding the oval, this method moves the current point, which implicitly closes the current subpath. If the *aRect* parameter specifies a square, the inscribed path is a circle. The path is constructed by starting in the lower-right quadrant of the rectangle and adding arc segments counterclockwise to complete the oval.

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

Related Sample Code
Grady
SonOfSillyBalls

Declared In
NSBezierPath.h

appendBezierPathWithPackedGlyphs:

Appends an array of packed glyphs to the receiver's path.

```
- (void)appendBezierPathWithPackedGlyphs:(const char *)packedGlyphs
```

Parameters

packedGlyphs

A C-style array containing one or more CGGlyph data types terminated by a NULL character.

Discussion
You should avoid using this method directly. Instead, use the appendBezierPathWithGlyph:inFont: (page 29) and appendBezierPathWithGlyphs:count:inFont: (page 30) methods to append glyphs to a path.

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

See Also
+ drawPackedGlyphs:atPoint: (page 20)

Related Sample Code
DockTile

SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithPoints:count:

Appends a series of line segments to the receiver's path.

```
- (void)appendBezierPathWithPoints:(NSPointArray)points count:(NSInteger)count
```

Parameters

points

A C-style array of NSPoint data types, each of which contains the end point of the next line segment.

count

The number of points in the *points* parameter.

Discussion
This method interprets the points as a set of connected line segments. If the current path contains an open subpath, a line is created from the last point in that subpath to the first point in the points array. If the current path is empty, the first point in the points array is used to set the starting point of the line segments. Subsequent line segments are added using the remaining points in the array.

This method does not close the path that is created. If you wish to create a closed path, you must do so by explicitly invoking the receiver's closePath (page 35) method.

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

Declared In
NSBezierPath.h

appendBezierPathWithRect:

Appends a rectangular path to the receiver's path.

```
- (void)appendBezierPathWithRect:(NSRect)aRect
```

Parameters

aRect

The rectangle describing the path to create.

Discussion
Before adding the rectangle, this method moves the current point to the origin of the rectangle, which implicitly closes the current subpath (if any). The path is constructed by starting at the origin of *aRect* and adding line segments in a counterclockwise direction. The final segment is added using a closePath (page 35) message.

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

See Also
+ bezierPathWithRect: (page 15)
+ fillRect: (page 20)
+ strokeRect: (page 26)

Related Sample Code
Cropped Image
IBFragmentView
TrackBall

Declared In
NSBezierPath.h

appendBezierPathWithRoundedRect:xRadius:yRadius:

Appends a rounded rectangular path to the receiver's path.

```
- (void)appendBezierPathWithRoundedRect:(NSRect)rect xRadius:(CGFloat)xRadius
    yRadius:(CGFloat)yRadius
```

Parameters

rect
The rectangle that defines the basic shape of the path.

xRadius
The radius of each corner oval along the x-axis. Values larger than half the rectangle's width are clamped to half the width.

yRadius
The radius of each corner oval along the y-axis. Values larger than half the rectangle's height are clamped to half the height.

Discussion
The path is constructed in a counter-clockwise direction, starting at the top-left corner of the rectangle. If either one of the radius parameters contains the value `0.0`, the returned path is a plain rectangle without rounded corners.

Availability
Available in Mac OS X v10.5 and later.

See Also
+ bezierPathWithRoundedRect:xRadius:yRadius: (page 16)

Declared In
NSBezierPath.h

bezierPathByFlatteningPath

Creates and returns a "flattened" copy of the receiver.

```
- (NSBezierPath *)bezierPathByFlatteningPath
```

Return Value
A new path object whose contents are a flattened version of the receiver's path.

Discussion
Flattening a path converts all curved line segments into straight line approximations. The granularity of the approximations is controlled by the path's current flatness value, which is set using the `setDefaultFlatness:` (page 21) method.

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

Declared In
`NSBezierPath.h`

bezierPathByReversingPath

Creates and returns a new `NSBezierPath` object with the reversed contents of the receiver's path.

```
- (NSBezierPath *)bezierPathByReversingPath
```

Return Value
A new path object whose contents are a reversed version of the receiver's path.

Discussion
Reversing a path does not necessarily change the appearance of the path when rendered. Instead, it changes the direction in which path segments are drawn. For example, reversing the path of a rectangle (whose line segments are normally drawn starting at the origin and proceeding in a counterclockwise direction) causes its line segments to be drawn in a clockwise direction instead. Drawing a reversed path could affect the appearance of a filled pattern, depending on the pattern and the fill rule in use.

This method reverses each whole or partial subpath in the path object individually.

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

Declared In
`NSBezierPath.h`

bounds

Returns the bounding box of the receiver's path.

```
- (NSRect)bounds
```

Return Value
The rectangle that encloses the path of the receiver. If the path contains curve segments, the bounding box encloses the curve but may not enclose the control points used to calculate the curve.

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

See Also
`- controlPointBounds` (page 36)

Related Sample Code
DockTile
ImageMapExample
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

cachesBezierPath

Returns a Boolean value indicating whether this object maintains a cached image of its path.

```
- (BOOL)cachesBezierPath
```

Return Value
YES if the path maintains a cached image; otherwise, NO.

Discussion
Caching of paths currently has no effect, so method always returns NO.

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

See Also
- setCachesBezierPath: (page 47)

Declared In
NSBezierPath.h

closePath

Closes the most recently added subpath.

```
- (void)closePath
```

Discussion
This method closes the current subpath by creating a line segment between the first and last points in the subpath. This method subsequently updates the current point to the end of the newly created line segment, which is also the first point in the now closed subpath.

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

See Also
- fill (page 39)

Related Sample Code
DockTile
IBFragmentView
SpeedometerView

WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

containsPoint:

Returns a Boolean value indicating whether the receiver contains the specified point.

```
- (BOOL)containsPoint:(NSPoint)aPoint
```

Parameters
aPoint
The point to test against the path, specified in the path object's coordinate system.

Return Value
YES if the path's enclosed area contains the specified point; otherwise, NO.

Discussion
This method checks the point against the path itself and the area it encloses. When determining hits in the enclosed area, this method uses the non-zero winding rule (NSNonZeroWindingRule). It does not take into account the line width used to stroke the path.

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

Related Sample Code
Dicey
ImageMapExample

Declared In
NSBezierPath.h

controlPointBounds

Returns the bounding box of the receiver’s path, including any control points.

```
- (NSRect)controlPointBounds
```

Return Value
The rectangle that encloses the receiver's path. If the path contains curve segments, the bounding box encloses the control points of the curves as well as the curves themselves.

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

See Also
- bounds (page 34)

Declared In
NSBezierPath.h

currentPoint

Returns the receiver's current point (the trailing point or ending point in the most recently added segment).

```
- (NSPoint)currentPoint
```

Return Value
The point from which the next drawn line or curve segment begins.

Discussion
If the receiver is empty, this method raises `NSGenericException`.

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

See Also
- `closePath` (page 35)
- `curveToPoint:controlPoint1:controlPoint2:` (page 37)
- `lineToPoint:` (page 42)
- `moveToPoint:` (page 44)

Declared In
`NSBezierPath.h`

curveToPoint:controlPoint1:controlPoint2:

Adds a Bezier cubic curve to the receiver's path.

```
- (void)curveToPoint:(NSPoint)aPoint controlPoint1:(NSPoint)controlPoint1
    controlPoint2:(NSPoint)controlPoint2
```

Parameters

aPoint
The destination point of the curve segment, specified in the current coordinate system

controlPoint1
The point that determines the shape of the curve near the current point.

controlPoint2
The point that determines the shape of the curve near the destination point.

Discussion
You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

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

See Also
- `closePath` (page 35)
- `lineToPoint:` (page 42)
- `relativeCurveToPoint:controlPoint1:controlPoint2:` (page 44)
+ `setDefaultFlatness:` (page 21)

Related Sample Code
CocoaVideoFrameToGWorld
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

elementAtIndex:

Returns the type of path element at the specified index.

```
- (NSBezierPathElement)elementAtIndex:(NSInteger)index
```

Parameters
index
The index of the desired path element.

Return Value
The type of the path element. For a list of constants, see "NSBezierPathElement" (page 54).

Discussion
Path elements describe the commands used to define a path and include basic commands such as moving to a specific point, creating a line segment, creating a curve, or closing the path. The elements are stored in the order of their execution.

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

See Also
- elementCount (page 39)
- elementAtIndex:associatedPoints: (page 38)
- bezierPathByReversingPath (page 34)

Declared In
NSBezierPath.h

elementAtIndex:associatedPoints:

Gets the element type and (and optionally) the associated points for the path element at the specified index.

```
- (NSBezierPathElement)elementAtIndex:(NSInteger)index
    associatedPoints:(NSPointArray)points
```

Parameters
index
The index of the desired path element.

points

On input, a C-style array containing up to three `NSPoint` data types, or `NULL` if you do not want the points. On output, the data points associated with the specified path element.

Return Value
The type of the path element. For a list of constants, see "NSBezierPathElement" (page 54).

Discussion
If you specify a value for the points parameter, your array must be large enough to hold the number of points for the given path element. Move, close path, and line segment commands return one point. Curve operations return three points.

For curve operations, the order of the points is controlPoint1 (*points*[0]), controlPoint2 (*points*[1]), endPoint (*points*[2]).

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

See Also
- `elementCount` (page 39)
- `elementAtIndex:` (page 38)

Declared In
`NSBezierPath.h`

elementCount

Returns the total number of path elements in the receiver's path.

```
- (NSInteger)elementCount
```

Return Value
The number of path elements.

Discussion
Each element type corresponds to one of the operations described in "Path Elements".

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

See Also
- `elementAtIndex:` (page 38)
- `elementAtIndex:associatedPoints:` (page 38)

Related Sample Code
Cropped Image

Declared In
`NSBezierPath.h`

fill

Paints the region enclosed by the receiver's path.

```
- (void)fill
```

Discussion
This method fills the path using the current fill color and the receiver's current winding rule. If the path contains any open subpaths, this method implicitly closes them before painting the fill region.

The painted region includes the pixels right up to, but not including, the path line itself. For paths with large line widths, this can result in overlap between the fill region and the stroked path (which is itself centered on the path line).

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

See Also
- `- stroke` (page 52)
- `- windingRule` (page 54)
- `- set` (NSColor)

Related Sample Code
Cropped Image
Dicey
DockTile
SpeedometerView
WebKitPluginStarter

Declared In
`NSBezierPath.h`

flatness

Returns the flatness value of the receiver's path.

```
- (CGFloat)flatness
```

Return Value
The flatness value of the path. If no value is set, this method returns the default flatness value.

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

See Also
- `- setFlatness:` (page 48)
- `+ defaultFlatness` (page 17)

Declared In
`NSBezierPath.h`

getLineDash:count:phase:

Returns the line-stroking pattern for the receiver.

```
- (void)getLineDash:(CGFloat *)pattern count:(NSInteger *)count phase:(CGFloat
    *)phase
```

Parameters

pattern

On input, a C-style array of floating point values, or `nil` if you do not want the pattern values. On output, this array contains the lengths (measured in points) of the line segments and gaps in the pattern. The values in the array alternate, starting with the first line segment length, followed by the first gap length, followed by the second line segment length, and so on.

count

On input, a pointer to an integer or `nil` if you do not want the number of pattern entries. On output, the number of entries written to *pattern*.

phase

On input, a pointer to a floating point value or `nil` if you do not want the phase. On output, this value contains the offset at which to start drawing the pattern, measured in points along the dashed-line pattern. For example, a phase of 6 in the pattern 5-2-3-2 would cause drawing to begin in the middle of the first gap.

Discussion

The array in the *pattern* parameter must be large enough to hold all of the returned values in the pattern. If you are not sure how many values there might be, you can call this method twice. The first time you call it, do not pass a value for *pattern* but use the returned value in *count* to allocate an array of floating-point numbers that you can then pass in the second time.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setLineDash:count:phase:` (page 49)

Declared In

`NSBezierPath.h`

isEmpty

Returns a Boolean value indicating whether the receiver is empty.

```
- (BOOL)isEmpty
```

Return Value

`YES` if the receiver contains no path elements; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBezierPath.h`

lineCapStyle

Returns the line cap style for the receiver's path.

```
- (NSLineCapStyle)lineCapStyle
```

Return Value
The receiver's line cap style. For a list of values, see "Constants" (page 54). If this value is not set for the receiver, the default line cap style is returned.

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

See Also
`+ defaultLineCapStyle` (page 18)
`+ setDefaultLineCapStyle:` (page 21)
`- setLineCapStyle:` (page 49)

Declared In
`NSBezierPath.h`

lineJoinStyle

Returns the receiver's line join style.

```
- (NSLineJoinStyle)lineJoinStyle
```

Return Value
The receiver's line join style. For a list of values, see "Constants" (page 54). If this value is not set for the receiver, the default line join style is returned.

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

See Also
`+ defaultLineJoinStyle` (page 18)
`+ setDefaultLineJoinStyle:` (page 22)
`- setLineJoinStyle:` (page 50)

Declared In
`NSBezierPath.h`

lineToPoint:

Appends a straight line to the receiver's path

```
- (void)lineToPoint:(NSPoint)aPoint
```

Parameters
aPoint
The destination point of the line segment, specified in the current coordinate system.

Discussion
This method creates a straight line segment starting at the current point and ending at the point specified by the *aPoint* parameter. The current point is the last point in the receiver's most recently added segment.

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

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

See Also
- `closePath` (page 35)
- `curveToPoint:controlPoint1:controlPoint2:` (page 37)

Related Sample Code
IBFragmentView
ImageMapExample
Polygons
Sketch-112
Squiggles

Declared In
`NSBezierPath.h`

lineWidth

Returns the line width of the receiver's path.

```
- (CGFloat)lineWidth
```

Return Value
The line width of the receiver, measured in points in the user coordinate space.

Discussion
If no value was set explicitly for the receiver, this method returns the default line width.

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

See Also
- `setLineWidth:` (page 51)
+ `defaultLineWidth` (page 19)

Declared In
`NSBezierPath.h`

miterLimit

Returns the miter limit of the receiver's path.

```
- (CGFloat)miterLimit
```

Return Value
The miter limit of the path. If no value is set, this method returns the default miter limit.

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

See Also
- `setMiterLimit:` (page 51)
+ `defaultMiterLimit` (page 19)

Declared In
`NSBezierPath.h`

moveToPoint:

Moves the receiver's current point to the specified location.

```
- (void)moveToPoint:(NSPoint)aPoint
```

Parameters

aPoint
A point in the current coordinate system.

Discussion
This method implicitly closes the current subpath (if any) and sets the current point to the value in *aPoint*. When closing the previous subpath, this method does not cause a line to be created from the first and last points in the subpath.

For many path operations, you must invoke this method before issuing any commands that cause a line or curve segment to be drawn.

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

See Also
- `closePath` (page 35)
- `curveToPoint:controlPoint1:controlPoint2:` (page 37)
- `lineToPoint:` (page 42)

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

relativeCurveToPoint:controlPoint1:controlPoint2:

Adds a Bezier cubic curve to the receiver's path from the current point to a new location, which is specified as a relative distance from the current point.

```
- (void)relativeCurveToPoint:(NSPoint)aPoint controlPoint1:(NSPoint)controlPoint1
    controlPoint2:(NSPoint)controlPoint2
```

Parameters

aPoint

The destination point of the curve segment, interpreted as a relative offset from the current point.

controlPoint1

The point that determines the shape of the curve near the current point, interpreted as a relative offset from the current point.

controlPoint2

The point that determines the shape of the curve near the destination point, interpreted as a relative offset from the current point.

Discussion

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `closePath` (page 35)
- `curveToPoint:controlPoint1:controlPoint2:` (page 37)
- `relativeLineToPoint:` (page 45)
- `relativeMoveToPoint:` (page 46)

Declared In

`NSBezierPath.h`

relativeLineToPoint:

Appends a straight line segment to the receiver's path starting at the current point and moving towards the specified point, relative to the current location.

```
- (void)relativeLineToPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point whose coordinates are interpreted as a relative offset from the current point.

Discussion

The destination point is relative to the current point. For example, if the current point is (1, 1) and *aPoint* contains the value (1, 2), a line segment is created between the points (1, 1) and (2, 3).

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 35)
- `lineToPoint:` (page 42)
- `relativeLineToPoint:` (page 45)
- `relativeMoveToPoint:` (page 46)

Declared In
`NSBezierPath.h`

relativeMoveToPoint:

Moves the receiver's current point to a new point whose location is the specified distance from the current point.

```
- (void)relativeMoveToPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point whose coordinates are interpreted as a relative offset from the current point.

Discussion

This method implicitly closes the current subpath (if any) and updates the location of the current point. For example, if the current point is (1, 1) and *aPoint* contains the value (1, 2), the previous subpath would be closed and the current point would become (2, 3). When closing the previous subpath, this method does not cause a line to be created from the first and last points in the subpath.

You must set the path's current point (using the `moveToPoint:` (page 44) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 35)
- `relativeCurveToPoint:controlPoint1:controlPoint2:` (page 44)
- `relativeLineToPoint:` (page 45)

Declared In
`NSBezierPath.h`

removeAllPoints

Removes all path elements from the receiver, effectively clearing the path.

```
- (void)removeAllPoints
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
`NSBezierPath.h`

setAssociatedPoints:atIndex:

Changes the points associated with the specified path element.

```
- (void)setAssociatedPoints:(NSPointArray)points atIndex:(NSInteger)index
```

Parameters

points

A C-style array containing up to three `NSPoint` data types. This parameter must contain the correct number of points for the path element at the specified index. Move, close path, and line segment commands require one point. Curve operations require three points.

index

The index of the path element you want to modify.

Discussion

You can use this method to change the points associated with a path quickly and without recreating the path. You cannot use this method to change the type of the path element.

The following example shows you how you would modify the point associated with a line path element. The path created by this example results in a path with two elements. The first path element specifies a move to point (0, 0) while the second creates a line to point (100, 100). It then changes the line to go only to the point (50,50) using this method:

```
NSBezierPath *bezierPath = [NSBezierPath bezierPath];
NSPoint newPoint = NSMakePoint(50.0, 50.0);

[bezierPath moveToPoint: NSMakePoint(0.0, 0.0)];
[bezierPath lineToPoint: NSMakePoint(100.0, 100.0)];

// Modifies the point added by lineToPoint: method (100.0, 100.0)
// to the new point (50.0, 50.0)
[bezierPath setAssociatedPoints: &newPoint atIndex: 1];
```

Note: If you specify too few points for a path element of type `NSCurveToBezierPathElement`, the behavior of this method is undefined.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBezierPath.h`

setCachesBezierPath:

Sets whether the receiver should cache its path information.

```
- (void)setCachesBezierPath:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should cache its path information; otherwise, `NO`.

Discussion

Caching of paths currently has no effect.

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

See Also
- `cachesBezierPath` (page 35)

Declared In
`NSBezierPath.h`

setClip

Replaces the clipping path of the current graphics context with the area inside the receiver's path.

```
- (void)setClip
```

Discussion
You should avoid using this method as a way of adjusting the clipping path, as it may expand the clipping path beyond the bounds set by the enclosing view. If you do use this method, be sure to save the graphics state prior to modifying the clipping path and restore the graphics state when you are done.

This method uses the current winding rule to determine the clipping shape of the receiver. This method does not affect the receiver's path.

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

See Also
- `addClip` (page 26)
+ `clipRect:` (page 17)
- `saveGraphicsState` (NSGraphicsContext)
- `restoreGraphicsState` (NSGraphicsContext)

Related Sample Code
PDF Annotation Editor

Declared In
`NSBezierPath.h`

setFlatness:

Sets the flatness value for the receiver's path.

```
- (void)setFlatness:(CGFloat)flatness
```

Parameters
flatness
The flatness value for the path.

Discussion
The flatness value specifies the accuracy (or smoothness) with which curves are rendered. It is also the maximum error tolerance (measured in pixels) for rendering curves, where smaller numbers give smoother curves at the expense of more computation. The exact interpretation may vary slightly on different rendering devices.

The default flatness value is 0.6, which yields smooth curves.

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

See Also
- `flatness` (page 40)
+ `setDefaultFlatness:` (page 21)

Declared In
`NSBezierPath.h`

setLineCapStyle:

Sets the line cap style for the receiver's path.

```
- (void)setLineCapStyle:(NSLineCapStyle)lineCapStyle
```

Parameters
lineCapStyle
The line cap style to use with the receiver. For a list of values, see "Constants" (page 54).

Discussion
The line cap style specifies the shape of the endpoints of an open path when stroked. Figure 1 (page 22) shows the appearance of the available line cap styles.

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

See Also
+ `defaultLineCapStyle` (page 18)
+ `setDefaultLineCapStyle:` (page 21)
- `lineCapStyle` (page 41)

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

setLineDash:count:phase:

Sets the line-stroking pattern for the receiver.

```
- (void)setLineDash:(const CGFloat *)pattern count:(NSInteger)count
    phase:(CGFloat)phase
```

Parameters

pattern

A C-style array of floating point values that contains the lengths (measured in points) of the line segments and gaps in the pattern. The values in the array alternate, starting with the first line segment length, followed by the first gap length, followed by the second line segment length, and so on

count

The number of values in *pattern*.

phase

The offset at which to start drawing the pattern, measured in points along the dashed-line pattern. For example, a phase of 6 in the pattern 5-2-3-2 would cause drawing to begin in the middle of the first gap

Discussion

For example, to produce a supermarket coupon type of dashed line:

```
array[0] = 5.0; //segment painted with stroke color
array[1] = 2.0; //segment not painted with a color

[path setLineDash: array count: 2 phase: 0.0];
```

In the above example, if you set *phase* to 6.0, the line dash would begin exactly six units into *pattern*, which would start the pattern in the middle of the first gap.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `getLineDash:count:phase:` (page 40)

Declared In

`NSBezierPath.h`

setLineJoinStyle:

Sets the line join style for the receiver's path.

```
- (void)setLineJoinStyle:(NSLineJoinStyle)lineJoinStyle
```

Parameters

lineJoinStyle

The line join style to use for the receiver's path. For a list of values, see "Constants" (page 54).

Discussion

The line join style specifies the shape of the joints between connected segments of a stroked path. Figure 2 (page 23) shows the appearance of the available line join styles.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `+ defaultLineJoinStyle` (page 18)
- `+ setDefaultLineJoinStyle:` (page 22)
- `- lineJoinStyle` (page 42)

Related Sample Code
DockTile
PDFKitLinker2
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

setLineWidth:

Sets the line width of the receiver's path.

```
- (void)setLineWidth:(CGFloat)lineWidth
```

Parameters
lineWidth
The line width to use for the receiver, measured in points in the user coordinate space.

Discussion
The line width defines the thickness of the receiver's stroked path. A width of 0 is interpreted as the thinnest line that can be rendered on a particular device. The actual rendered line width may vary from the specified width by as much as 2 device pixels, depending on the position of the line with respect to the pixel grid and the current anti-aliasing settings. The width of the line may also be affected by scaling factors specified in the current transformation matrix of the active graphics context.

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

See Also
- `lineWidth` (page 43)
+ `setDefaultLineWidth:` (page 23)

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

setMiterLimit:

Sets the miter limit for the receiver's path.

```
- (void)setMiterLimit:(CGFloat)miterLimit
```

Parameters

miterLimit

A value indicating the limit at which miter joins are converted to bevel joins.

Discussion

The miter limit helps you avoid spikes at the junction of two line segments connected by a miter join (`NSMiterLineJoinStyle`). If the ratio of the miter length—the diagonal length of the miter join—to the line thickness exceeds the miter limit, the joint is converted to a bevel join. The default miter limit value is 10, which converts miters whose angle at the joint is less than 11 degrees.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `miterLimit` (page 43)
+ `setDefaultMiterLimit:` (page 24)

Declared In

`NSBezierPath.h`

setWindingRule:

Sets the winding rule used to fill the receiver's path.

```
- (void)setWindingRule:(NSWindingRule)aWindingRule
```

Parameters

aWindingRule

The winding rule to use for the path. This value may be either `NSNonZeroWindingRule` or `NSEvenOddWindingRule`.

Discussion

For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

Availability

Available in Mac OS X v10.0 and later.

See Also

- `fill` (page 39)
- `windingRule` (page 54)
+ `setDefaultWindingRule:` (page 24)

Related Sample Code

Cropped Image

Declared In

`NSBezierPath.h`

stroke

Draws a line along the receiver's path using the current stroke color and drawing attributes.

```
- (void)stroke
```

Discussion
The drawn line is centered on the path with its sides parallel to the path segment. This method uses the current drawing attributes associated with the receiver. If a particular attribute is not set for the receiver, this method uses the corresponding default attribute.

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

See Also
- `fill` (page 39)
+ `setDefaultLineCapStyle:` (page 21)
+ `setDefaultLineJoinStyle:` (page 22)
- `set` (NSColor)

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

transformUsingAffineTransform:

Transforms all points in the receiver using the specified transform.

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

Parameters
aTransform
The transform to apply to the path.

Discussion
This method applies the transform to the path's points immediately. The following code translates a line from 0,0 to 100,100 to a line from 10,10 to 110,110.

```
NSBezierPath *bezierPath = [NSBezierPath bezierPath];
NSAffineTransform *transform = [NSAffineTransform transform];

[bezierPath moveToPoint: NSMakePoint(0.0, 0.0)];
[bezierPath lineToPoint: NSMakePoint(100.0, 100.0)];

[transform translateXBy: 10.0 yBy: 10.0];
[bezierPath transformUsingAffineTransform: transform];
```

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

Related Sample Code
DockTile

SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

windingRule

Returns the winding rule used to fill the receiver's path.

```
- (NSWindingRule)windingRule
```

Return Value
The winding rule for the path. This value may be either `NSNonZeroWindingRule` or `NSEvenOddWindingRule`.

Discussion
This value overrides the default value returned by `defaultWindingRule` (page 19).

For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

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

See Also
- `fill` (page 39)
- `setWindingRule:` (page 52)
+ `defaultWindingRule` (page 19)

Declared In
NSBezierPath.h

Constants

NSBezierPathElement

Basic path element commands.

```
typedef enum {
    NSMoveToBezierPathElement,
    NSLineToBezierPathElement,
    NSCurveToBezierPathElement,
    NSClosePathBezierPathElement
} NSBezierPathElement;
```

Constants

`NSMoveToBezierPathElement`

Moves the path object's current drawing point to the specified point.

This path element does not result in any drawing. Using this command in the middle of a path results in a disconnected line segment.

Contains 1 point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSLineToBezierPathElement`

Creates a straight line from the current drawing point to the specified point.

Lines and rectangles are specified using this path element.

Contains 1 point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSCurveToBezierPathElement`

Creates a curved line segment from the current point to the specified endpoint using two control points to define the curve.

The points are stored in the following order: controlPoint1, controlPoint2, endPoint. Ovals, arcs, and Bezier curves all use curve elements to specify their geometry.

Contains 3 points.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSClosePathBezierPathElement`

Marks the end of the current subpath at the specified point.

Note that the point specified for the Close Path element is essentially the same as the current point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

Discussion

These commands are enough to define all of the possible path shapes. Each command has one or more points that contain information needed to position the path element. Most path elements use the current drawing point as the starting point for drawing. For more details, see Paths.

Declared In

`NSBezierPath.h`

NSLineJoinStyle

These constants specify the shape of the joints between connected segments of a stroked path.

```
typedef enum {
    NSMiterLineJoinStyle = 0,
    NSRoundLineJoinStyle = 1,
    NSBevelLineJoinStyle = 2
} NSLineJoinStyle;
```

Constants

`NSBevelLineJoinStyle`

Specifies a bevel line shape of the joints between connected segments of a stroked path.

See the `setDefaultLineJoinStyle:` (page 22) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSMiterLineJoinStyle`

Specifies a miter line shape of the joints between connected segments of a stroked path.

See the `setDefaultLineJoinStyle:` (page 22) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSRoundLineJoinStyle`

Specifies a round line shape of the joints between connected segments of a stroked path.

See the `setDefaultLineJoinStyle:` (page 22) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

Declared In

`NSBezierPath.h`

NSLineCapStyle

These constants specify the shape of endpoints for an open path when stroked.

```
typedef enum {
    NSButtLineCapStyle   = 0,
    NSRoundLineCapStyle  = 1,
    NSSquareLineCapStyle = 2
} NSLineCapStyle;
```

Constants

`NSButtLineCapStyle`

Specifies a butt line cap style for endpoints for an open path when stroked.

See the `setDefaultLineCapStyle:` (page 21) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSSquareLineCapStyle`

Specifies a square line cap style for endpoints for an open path when stroked.

See the `setDefaultLineCapStyle:` (page 21) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSRoundLineCapStyle`
Specifies a round line cap style for endpoints for an open path when stroked.

See the `setDefaultLineCapStyle:` (page 21) method for an example of the appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

Declared In
`NSBezierPath.h`

NSWindingRule

These constants are used to specify the winding rule a Bezier path should use.

```
typedef enum {
    NSNonZeroWindingRule = 0,
    NSEvenOddWindingRule = 1
} NSWindingRule;
```

Constants
`NSNonZeroWindingRule`
Specifies the non-zero winding rule.

Count each left-to-right path as +1 and each right-to-left path as -1. If the sum of all crossings is 0, the point is outside the path. If the sum is nonzero, the point is inside the path and the region containing it is filled. This is the default winding rule.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSEvenOddWindingRule`
Specifies the even-odd winding rule.

Count the total number of path crossings. If the number of crossings is even, the point is outside the path. If the number of crossings is odd, the point is inside the path and the region containing it should be filled.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

Discussion
These constants are described in more detail in Paths.

Declared In
`NSBezierPath.h`

Document Revision History

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

Date	Notes
2007-03-02	Updated links to point to Cocoa Drawing Guide. Updated for Mac OS X v10.5.
2006-10-03	Corrected the discussion of setLineDash:count:phase: to say that line dash phrases represent an exact distance, not an approximate one.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

G

I

L

M

N

R

S

T

W