Map Kit Framework Reference

User Experience

2010-05-11

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

Contents

Introduction

Framework	/System/Library/Frameworks/MapKit.framework
Header file directories	/System/Library/Frameworks/MapKit.framework/Headers
Companion guide	iOS Application Programming Guide
Declared in	MKAnnotation.h MKAnnotationView.h MKCircle.h MKCircleView.h MKGeometry.h MKMapView.h MKMultiPoint.h MKOverlay.h MKOverlayPathView.h MKOverlayView.h MKPinAnnotationView.h MKPlacemark.h MKPointAnnotation.h MKPolygon.h MKPolygonView.h MKPolyline.h MKPolylineView.h MKReverseGeocoder.h MKShape.h MKTypes.h MKUserLocation.h

About the Map Kit Framework

The Map Kit framework provides an interface for embedding maps directly into your own windows and views. This framework also provides support for annotating the map, adding overlays, and performing reverse-geocoding lookups to determine placemark information for a given map coordinate.

Important: The Map Kit framework uses Google services to provide map data. Use of specific classes of this framework (and their associated interfaces) binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Introduction

Classes

PART I

Classes

MKAnnotationView Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKAnnotationView.h
Related sample code	MapCallouts WeatherMap

Overview

The `MKAnnotationView` class is responsible for presenting annotations visually in a map view. Annotation views are loosely coupled to a corresponding annotation object, which is an object that corresponds to the `MKAnnotation` protocol. When an annotation's coordinate point is in the visible region, the map view asks its delegate to provide a corresponding annotation view. Annotation views may be recycled later and put into a reuse queue that is maintained by the map view.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

The most efficient way to provide the content for an annotation view is to set its `image` (page 18) property. The annotation view sizes itself automatically to the image you specify and draws that image for its contents. Because it is a view, however, you could also override the `drawRect:` method and draw your view's content manually. If you choose to override `drawRect:` directly and you do not specify a custom image in the `image` property, be aware that the width and height of the annotation view's frame are set to 0 by default. Before your custom content can be drawn, you must set the width and height to nonzero values by modifying the view's `frame` property. In general, if your content consists entirely of static images, it is more efficient to set the `image` property and change it as needed than to draw the images yourself.

Annotation views remain anchored to the map at the point specified by their associated annotation object. Although they scroll with the map contents, annotation views reside in a separate display layer and are not scaled when the size of the visible map region changes.

Annotation views support the concept of a selection state, which determines whether the view is unselected, selected, or selected and displaying a standard callout view. The user toggles between the selection states through interactions with the annotation view. In the unselected state, the annotation view is displayed but

not highlighted. In the selected state, the annotation is highlighted but the callout is not displayed. And finally, the annotation can be displayed both with a highlight and a callout. The callout view displays additional information such as a title string and controls for viewing more information. The title information is provided by the annotation object but your annotation view is responsible for providing any custom controls. For more information, see the subclassing notes.

Reusing Annotation Views

Annotation views are designed to be reused as the user (or your application) changes the visible map region. The reuse of annotation views provides significant performance improvements during scrolling by avoiding the creation of new view objects during this time critical operation. For this reason, annotation views should not be tightly coupled to the contents of their associated annotation. Instead, it should be possible to use the properties of an annotation view (or setter methods) to configure the view for a new annotation object.

Whenever you initialize a new annotation view, you should always specify a reuse identifier for that view. As annotation views are no longer needed, the map view may put them into a reuse queue. As new annotations are added to the map view, the delegate object can then dequeue and reconfigure an existing view (rather than create a new one) using the `dequeueReusableAnnotationViewWithIdentifier:` (page 46) method of `MKMapView`.

Subclassing Notes

You can use the `MKAnnotationView` class as is or subclass it to provide custom behavior as needed. The `image` (page 18) property of the class lets you set the appearance of the annotation view without subclassing directly. You might also create custom subclasses as a convenience and use them to put the annotation view in a known state. For example, the `MKPinAnnotationView` subclass initializes the contents of the annotation view to a pin image.

There are no special requirements for subclassing `MKAnnotationView`. However, the following list includes some reasons you might want to subclass and some of the methods you would override to implement the desired behavior:

- To put the annotation view into a consistent state, provide a custom initialization method. Your custom initialization method would then call `initWithAnnotation:reuseIdentifier:` (page 20) to initialize the superclass.
- To provide custom callout views, override the `leftCalloutAccessoryView` (page 18) method and use it to return the views.

If you support draggable annotation views in iOS 4.0 and later, your subclass is responsible for changing the value in the `dragState` (page 16) property to appropriate values at key transition points in the drag operation. For more information, see the description of that property.

Tasks

Initializing and Preparing the View

- `initWithAnnotation:reuseIdentifier:` (page 20)
  Initializes and returns a new annotation view.
- `prepareForReuse` (page 21)
  Called when the view is removed from the reuse queue.

Getting and Setting Attributes

`enabled` (page 17) *property*
A Boolean value indicating whether the annotation is enabled.

`image` (page 18) *property*
The image to be displayed by the annotation view.

`highlighted` (page 17) *property*
A Boolean value indicating whether the annotation view is highlighted.

`annotation` (page 14) *property*
The annotation object currently associated with the view.

`centerOffset` (page 15) *property*
The offset (in pixels) at which to display the view.

`calloutOffset` (page 15) *property*
The offset (in pixels) at which to place the callout bubble.

`reuseIdentifier` (page 19) *property*
The string that identifies that this annotation view is reusable. (read-only)

Managing the Selection State

- `setSelected:animated:` (page 22)
  Sets the selection state of the annotation view.

`selected` (page 20) *property*
A Boolean value indicating whether the annotation view is currently selected.

Managing Callout Views

`canShowCallout` (page 15) *property*
A Boolean value indicating whether the annotation view is able to display extra information in a callout bubble.

`leftCalloutAccessoryView` (page 18) *property*
The view to display on the left side of the standard callout bubble.

`rightCalloutAccessoryView` (page 19) *property*
The view to display on the right side of the standard callout bubble.

Supporting Drag Operations

`draggable` (page 16) *property*
A Boolean indicating whether the annotation view is draggable.

`- setDragState:animated:` (page 21)
Sets the current drag state for the annotation view.

`dragState` (page 16) *property*
The current drag state of the annotation view.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

annotation

The annotation object currently associated with the view.

```
@property (nonatomic, retain) id <MKAnnotation> annotation
```

Discussion
You should not change the value of this property directly. This property contains a non-`nil` value only while the annotation view is visible on the map. If the view is queued and waiting to be reused, the value is `nil`

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts
WeatherMap

Declared In
`MKAnnotationView.h`

calloutOffset

The offset (in pixels) at which to place the callout bubble.

```
@property (nonatomic) CGPoint calloutOffset
```

Discussion
This property determines the additional distance by which to move the callout bubble. When this property is set to (0, 0), the anchor point of the callout bubble is placed on the top-center point of the annotation view's frame. Specifying positive offset values moves the callout bubble down and to the right, while specifying negative values moves it up and to the left.

Availability
Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

canShowCallout

A Boolean value indicating whether the annotation view is able to display extra information in a callout bubble.

```
@property (nonatomic) BOOL canShowCallout
```

Discussion
If `YES`, a standard callout bubble is shown when the user taps a selected annotation view. The callout uses the title and subtitle text from the associated annotation object. (If the annotation's `title` (page 115) method returns an empty string, the annotation view is treated as if its `enabled` property is set to `NO`.) The callout also displays any custom callout views stored in the `leftCalloutAccessoryView` and `rightCalloutAccessoryView` properties.

Availability
Available in iOS 3.0 and later.

See Also
`@property leftCalloutAccessoryView` (page 18)
`@property rightCalloutAccessoryView` (page 19)

Related Sample Code
MapCallouts

Declared In
`MKAnnotationView.h`

centerOffset

The offset (in pixels) at which to display the view.

```
@property (nonatomic) CGPoint centerOffset
```

Discussion
By default, the center point of an annotation view is placed at the coordinate point of the associated annotation. You can use this property to reposition the annotation view as needed. This x and y offset values are measured in pixels. Positive offset values move the annotation view down and to the right, while negative values move it up and to the left.

Availability
Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

draggable

A Boolean indicating whether the annotation view is draggable.

```
@property (nonatomic, getter=isDraggable) BOOL draggable
```

Discussion
Setting this property to `YES` makes an annotation draggable by the user. If `YES`, the associated annotation object must also implement the `setCoordinate:` (page 114) method. The default value of this property is `NO`.

Setting this property to `YES`, lets the map view know that the annotation is always draggable. In other words, you cannot conditionalize drag operations by attempting to stop an operation that has already been initiated; doing so can lead to undefined behavior. Once begun, the drag operation should always continue to completion.

Availability
Available in iOS 4.0 and later.

Declared In
`MKAnnotationView.h`

dragState

The current drag state of the annotation view.

```
@property (nonatomic) MKAnnotationViewDragState dragState
```

Discussion
Applications targeting iOS 4.1 and earlier can use this property to support drag operations in custom annotation views. If your application runs in iOS 4.2 or later, you should override the `setDragState:animated:` (page 21) method and use it to manage the drag state instead.

To support drag operations, you must override the implementation of this property and update the drag state at the following times:

- When the drag state changes to `MKAnnotationViewDragStateStarting` (page 23), you should set the state to `MKAnnotationViewDragStateDragging` (page 23). If you perform an animation to indicate the beginning of a drag, you should perform that animation before changing the state. Changing the state to the new value lets the map know that your animations are done.
- When the state changes to either `MKAnnotationViewDragStateCanceling` (page 23) or `MKAnnotationViewDragStateEnding` (page 23), set the state to `MKAnnotationViewDragStateNone` (page 23). If you perform an animation at the end of a drag, you should perform that animation before changing the state.

Changing the state to the `MKAnnotationViewDragStateDragging` or `MKAnnotationViewDragStateNone` value is the way to signal to the map view that you are done with any animations you wanted to perform. For example, when a drag operation begins for a pin annotation, the `MKPinAnnotationView` class executes an animation to lift the pin off the map. Similarly, when the pin is dropped, the class performs a drop animation. Even if you do not perform any animations, you should still change the value of this property to reflect the correct state.

You must not try to abort a new drag operation by changing the state from `MKAnnotationViewDragStateStarting` to `MKAnnotationViewDragStateNone`. If you do not want your annotation view to be draggable, set the `draggable` (page 16) property to `NO`.

Availability
Available in iOS 4.0 and later.

See Also
`@property draggable` (page 16)

Declared In
`MKAnnotationView.h`

enabled

A Boolean value indicating whether the annotation is enabled.

```
@property (nonatomic, getter=isEnabled) BOOL enabled
```

Discussion
The default value of this property is `YES`. If the value of this property is `NO`, the annotation view ignores touch events and cannot be selected. Subclasses may also display the annotation contents differently depending on the value of this property.

Availability
Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

highlighted

A Boolean value indicating whether the annotation view is highlighted.

```
@property (nonatomic, getter=isHighlighted) BOOL highlighted
```

Discussion
You should not set the value of this property directly. The map view sets it in response to touch events entering or exiting the annotation view's bounds.

Availability
Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

image

The image to be displayed by the annotation view.

```
@property (nonatomic, retain) UIImage *image
```

Discussion
Assigning a new image to this property also changes the size of the view's frame so that it matches the width and height of the new image. The position of the view's frame does not change.

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
`MKAnnotationView.h`

leftCalloutAccessoryView

The view to display on the left side of the standard callout bubble.

```
@property (retain, nonatomic) UIView *leftCalloutAccessoryView
```

Discussion
The default value of this property is `nil`. The left callout view is typically used to display information about the annotation or to link to custom information provided by your application. The height of your view should be 32 pixels or less.

If the view you specify is also a descendant of the `UIControl` class, you can use the map view's delegate to receive notifications when your control is tapped. If it does not descend from `UIControl`, your view is responsible for handling any touch events within its bounds.

Availability
Available in iOS 3.0 and later.

See Also
`@property canShowCallout` (page 15)

Related Sample Code
MapCallouts

Declared In
MKAnnotationView.h

reuseIdentifier

The string that identifies that this annotation view is reusable. (read-only)

```
@property (nonatomic, readonly) NSString *reuseIdentifier
```

Discussion
You specify the reuse identifier when you create the view. You use this type later to retrieve an annotation view that was created previously but which is currently unused because its annotation is not on screen.

If you define distinctly different types of annotations (with distinctly different annotation views to go with them), you can differentiate between the annotation types by specifying different reuse identifiers for each one.

Availability
Available in iOS 3.0 and later.

Declared In
MKAnnotationView.h

rightCalloutAccessoryView

The view to display on the right side of the standard callout bubble.

```
@property (retain, nonatomic) UIView *rightCalloutAccessoryView
```

Discussion
This property is set to `nil` by default. The right callout view is typically used to link to more detailed information about the annotation. The height of your view should be 32 pixels or less. A common view to specify for this property is `UIButton` object whose type is set to `UIButtonTypeDetailDisclosure`.

If the view you specify is also a descendant of the `UIControl` class, you can use the map view's delegate to receive notifications when your control is tapped. If it does not descend from `UIControl`, your view is responsible for handling any touch events within its bounds.

Availability
Available in iOS 3.0 and later.

See Also
`@property canShowCallout` (page 15)

Related Sample Code
MapCallouts

Declared In
MKAnnotationView.h

selected

A Boolean value indicating whether the annotation view is currently selected.

```
@property (nonatomic, getter=isSelected) BOOL selected
```

Discussion
You should not set the value of this property directly. If the property contains `YES`, the annotation view is displaying a callout bubble.

Availability
Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

Instance Methods

initWithAnnotation:reuseIdentifier:

Initializes and returns a new annotation view.

```
- (id)initWithAnnotation:(id <MKAnnotation>)annotation   reuseIdentifier:(NSString
    *)reuseIdentifier
```

Parameters

annotation
The annotation object to associate with the new view.

reuseIdentifier
If you plan to reuse the annotation view for similar types of annotations, pass a string to identify it. Although you can pass `nil` if you do not intend to reuse the view, reusing annotation views is generally recommended.

Return Value
The initialized annotation view or `nil` if there was a problem initializing the object.

Discussion
The reuse identifier provides a way for you to improve performance by recycling annotation views as they are scrolled on and off of the map. As views are no longer needed, they are moved to a reuse queue by the map view. When a new annotation becomes visible, your application can request a view for that annotation by passing the appropriate reuse identifier string to the `dequeueReusableAnnotationViewWithIdentifier:` (page 46) method of `MKMapView`.

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts
WeatherMap

Declared In
`MKAnnotationView.h`

prepareForReuse

Called when the view is removed from the reuse queue.

```
- (void)prepareForReuse
```

Discussion
The default implementation of this method does nothing. You can override it in your custom annotation views and use it to put the view in a known state before it is returned to your map view delegate.

Availability
Available in iOS 3.0 and later.

See Also
`dequeueReusableAnnotationViewWithIdentifier:` (page 46) (MKMapView)

Declared In
`MKAnnotationView.h`

setDragState:animated:

Sets the current drag state for the annotation view.

```
- (void)setDragState:(MKAnnotationViewDragState)newDragState animated:(BOOL)animated
```

Parameters

newDragState
The new drag state for the annotation view.

animated
If `YES`, the change to the new drag state should be animated; otherwise, it should be made without animations.

Discussion
Applications targeting iOS 4.2 and later can override this method and use it to implement drag support for custom annotation views. As the system detects user actions that would indicate a drag, it calls this method to update the drag state. In response to these changes, your custom implementation of this method should do the following:

- When the drag state changes to `MKAnnotationViewDragStateStarting` (page 23), set the state to `MKAnnotationViewDragStateDragging` (page 23). If you perform an animation to indicate the beginning of a drag, and the *animated* parameter is `YES`, perform that animation before changing the state.
- When the state changes to either `MKAnnotationViewDragStateCanceling` (page 23) or `MKAnnotationViewDragStateEnding` (page 23), set the state to `MKAnnotationViewDragStateNone` (page 23). If you perform an animation at the end of a drag, and the *animated* parameter is `YES`, you should perform that animation before changing the state.

The default implementation of this method sets the value of the `dragState` (page 16) property to the value in the *newDragState* parameter only. Therefore, direct subclasses can simply call the inherited version of this method to change the drag state; otherwise, just change the value in the `draggable` property directly.

Changing the state to `MKAnnotationViewDragStateDragging` or `MKAnnotationViewDragStateNone` is the way to signal to the map view that you are done with any animations you wanted to perform. For example, when a drag operation begins for a pin annotation, the `MKPinAnnotationView` class executes an animation to lift the pin off the map. Similarly, when the pin is dropped, the class performs a drop animation. Even if you do not perform any animations, you should call the inherited version of this method to update the `dragState` (page 16) property.

You must not try to abort a new drag operation by changing the state from `MKAnnotationViewDragStateStarting` to `MKAnnotationViewDragStateNone`. If you do not want your annotation view to be draggable, set the `draggable` (page 16) property to `NO`.

Availability
Available in iOS 4.2 and later.

Declared In
`MKAnnotationView.h`

setSelected:animated:

Sets the selection state of the annotation view.

```
- (void)setSelected:(BOOL)selected    animated:(BOOL)animated
```

Parameters

selected
Contains the value `YES` if the view should display itself as selected.

animated
Set to `YES` if the change in selection state is animated.

Discussion
You should not call this method directly. An `MKMapView` object calls this method in response to user interactions with the annotation.

Availability
Available in iOS 3.0 and later.

See Also
`selectAnnotation:animated:` (page 52) (MKMapView)

Declared In
`MKAnnotationView.h`

Constants

MKAnnotationViewDragState

These constants indicate the current drag state of an annotation view.

```
typedef enum MKAnnotationViewDragState {
    MKAnnotationViewDragStateNone = 0,
    MKAnnotationViewDragStateStarting,
    MKAnnotationViewDragStateDragging,
    MKAnnotationViewDragStateCanceling,
    MKAnnotationViewDragStateEnding
} MKAnnotationViewDragState;
```

Constants

`MKAnnotationViewDragStateNone`

The view is not involved in a drag operation. The annotation view is responsible for returning itself to this state when a drag ends or is canceled.

Available in iOS 4.0 and later.

Declared in `MKAnnotationView.h`.

`MKAnnotationViewDragStateStarting`

An action occurred that indicated the view should begin dragging. The map view automatically moves annotation views to this state in response to appropriate user actions.

Available in iOS 4.0 and later.

Declared in `MKAnnotationView.h`.

`MKAnnotationViewDragStateDragging`

The view is in the middle of a drag operation and is tracking progress.

Available in iOS 4.0 and later.

Declared in `MKAnnotationView.h`.

`MKAnnotationViewDragStateCanceling`

An action occurred that indicated the view should cancel the drag operation. You can put an annotation view into this state to abort the operation.

Available in iOS 4.0 and later.

Declared in `MKAnnotationView.h`.

`MKAnnotationViewDragStateEnding`

An action occurred that indicated the view was dropped by the user. The map view automatically moves annotation views to this state in response to appropriate user actions.

Available in iOS 4.0 and later.

Declared in `MKAnnotationView.h`.

Notifications

MKAnnotationCalloutInfoDidChangeNotification

Notifies observers that the title or subtitle information of an annotation object changed. (#Deprecated. Use KVO notifications instead.)

This notification supports legacy applications and is no longer necessary. MapKit tracks changes to the title and subtitle of an annotation using KVO notifications.

Availability

Available in iOS 3.0 and later.

Declared In
`MKAnnotationView.h`

MKCircle Class Reference

Inherits from	MKShape : NSObject
Conforms to	MKOverlay MKAnnotation (MKShape) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKCircle.h

Overview

The `MKCircle` class is a concrete overlay object representing a circular area on a map. This class manages the data that defines the area and is typically used in conjunction with an `MKCircleView` object, which handles the drawing of the circular area on a map.

Tasks

Creating a Circle Overlay

`+ circleWithCenterCoordinate:radius:` (page 27)
Creates and returns an `MKCircle` object using the specified coordinate and radius.

`+ circleWithMapRect:` (page 27)
Creates and returns an `MKCircle` object where the circular area is derived from the specified rectangle.

Accessing the Overlay's Attributes

`coordinate` (page 26) *property*
The center point of the circular area, specified as a latitude and longitude. (read-only)

`radius` (page 26) *property*
The radius of the circular area, measured in meters. (read-only)

`boundingMapRect` (page 26) *property*
The bounding rectangle of the circular area. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

boundingMapRect

The bounding rectangle of the circular area. (read-only)

```
@property (nonatomic, readonly) MKMapRect boundingMapRect
```

Discussion
As latitude values move away from the equator and toward the poles, the physical distance between map points gets smaller. This means that more map points are needed to represent the same distance. As a result, the bounding rectangle of a circle overlay gets larger as the center point of that circle moves away from the equator and toward the poles.

Availability
Available in iOS 4.0 and later.

Declared In
`MKCircle.h`

coordinate

The center point of the circular area, specified as a latitude and longitude. (read-only)

```
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKCircle.h`

radius

The radius of the circular area, measured in meters. (read-only)

```
@property (nonatomic, readonly) CLLocationDistance radius
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKCircle.h`

Class Methods

circleWithCenterCoordinate:radius:

Creates and returns an `MKCircle` object using the specified coordinate and radius.

```
+ (MKCircle *)circleWithCenterCoordinate:(CLLocationCoordinate2D)coord
    radius:(CLLocationDistance)radius
```

Parameters

coord

The center point of the circle, specified as a latitude and longitude value.

radius

The radius of the circle, measured in meters from the center point.

Return Value
A circle overlay object.

Availability
Available in iOS 4.0 and later.

Declared In
`MKCircle.h`

circleWithMapRect:

Creates and returns an `MKCircle` object where the circular area is derived from the specified rectangle.

```
+ (MKCircle *)circleWithMapRect:(MKMapRect)mapRect
```

Parameters

mapRect

The map rectangle used to determine the circular area. The center point of the rectangle is used as the center point of the circle. If the rectangle is not a square, the longest side of the rectangle is used to define the radius of the resulting circle.

Return Value
A circle overlay object.

Availability
Available in iOS 4.0 and later.

Declared In
`MKCircle.h`

MKCircleView Class Reference

Inherits from	MKOverlayPathView : MKOverlayView : UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKCircleView.h

Overview

The `MKCircleView` class provides the visual representation for an `MKCircle` annotation object. This view fills and strokes the circle represented by the annotation. You can change the color and other drawing attributes of the circle by modifying the properties inherited from the `MKOverlayPathView` class. This class is typically used as is and not subclassed.

Tasks

MethodGroup

`- initWithCircle:` (page 30)
Initializes and returns a new overlay view using the specified circle overlay object.

`circle` (page 30) *property*
The circle overlay object that contains the information used to draw the overlay. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

circle

The circle overlay object that contains the information used to draw the overlay. (read-only)

```
@property (nonatomic, readonly) MKCircle *circle
```

Availability
Available in iOS 4.0 and later.

Declared In
MKCircleView.h

Instance Methods

initWithCircle:

Initializes and returns a new overlay view using the specified circle overlay object.

```
- (id)initWithCircle:(MKCircle *)circle
```

Parameters

circle

The circle overlay containing the information about the circular area to be drawn.

Return Value
A new circle overlay view.

Availability
Available in iOS 4.0 and later.

Declared In
MKCircleView.h

MKMapView Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MapKit/MKMapView.h
Related sample code	MapCallouts WeatherMap WorldCities

Overview

An `MKMapView` object provides an embeddable map interface, similar to the one provided by the Maps application. You use this class as-is to display map information and to manipulate the map contents from your application. You can center the map on a given coordinate, specify the size of the area you want to display, and annotate the map with custom information.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

When you initialize a map view, you should specify the initial region for that map to display. You do this by setting the `region` (page 38) property of the map. A region is defined by a center point and a horizontal and vertical distance, referred to as the span. The **span** defines how much of the map at the given point should be visible and is also how you set the zoom level. Specifying a large span results in the user seeing a wide geographical area and corresponds to a low zoom level. Specifying a small span results in the user seeing a more narrow geographical area and corresponds to a higher zoom level.

In addition to setting the span programmatically, the `MKMapView` class supports many standard interactions for changing the position and zoom level of the map. In particular, map views support flick and pinch gestures for scrolling around the map and zooming in and out. Support for these gestures is enabled by default but can also be disabled using the `scrollEnabled` (page 39) and `zoomEnabled` (page 41) properties.

In iOS 4.0 and later, you can also use projected map coordinates instead of regions to specify some values. When you project the curved surface of the globe onto a flat surface, you get a two-dimensional version of a map where longitude lines appear to be parallel. Locations and distances on this map are specified using the `MKMapPoint` (page 162), `MKMapSize` (page 163), and `MKMapRect` (page 163) data types. You can use these data types to specify the map's visible region and when specifying the location of overlays.

Although you should not subclass the `MKMapView` class itself, you can get information about the map view's behavior by providing a delegate object. The map view calls the methods of your custom delegate to let it know about changes in the map status and to coordinate the display of custom annotations, which are described in more detail in "Annotating the Map" (page 32). The delegate object can be any object in your application as long as it conforms to the `MKMapViewDelegate` protocol. For more information about implementing the delegate object, see *MKMapViewDelegate Protocol Reference*.

Annotating the Map

The `MKMapView` class supports the ability to annotate the map with custom information. Because a map may have potentially large numbers of annotations, map views differentiate between the annotation objects used to manage the annotation data and the view objects for presenting that data on the map.

An **annotation object** is any object that conforms to the `MKAnnotation` protocol. Annotation objects are typically implemented using existing classes in your application's data model. This allows you to manipulate the annotation data directly but still make it available to the map view. Each annotation object contains information about the annotation's location on the map along with descriptive information that can be displayed in a callout.

The presentation of annotation objects on the screen is handled by an **annotation view**, which is an instance of the `MKAnnotationView` class. An annotation view is responsible for presenting the annotation data in a way that makes sense. For example, the Maps application uses a pin icon to denote specific points of interest on a map. (The MapKit framework offers the `MKPinAnnotationView` class for similar annotations in your own applications.) You could also create annotation views that cover larger portions of the map.

Because annotation views are needed only when they are onscreen, the `MKMapView` class provides a mechanism for queueing annotation views that are not in use. Annotation views with a reuse identifier can be detached and queued internally by the map view when they move off screen. This feature improves memory use by keeping only a small number of annotation views in memory at once and by recycling the views you do have. It also improves scrolling performance by alleviating the need to create new views while the map is scrolling.

When configuring your map interface, you should add all of your annotation objects right away. The map view uses the coordinate data in each annotation object to determine when the corresponding annotation view needs to appear on screen. When an annotation moves on screen, the map view asks its delegate to create a corresponding annotation view. If your application has different types of annotations, it can define different annotation view classes to represent each type.

Adding Overlays to the Map

In iOS 4.0 and later, you can use overlays to display content over a wide portion of the map. An **overlay** is any object that conforms to the `MKOverlay` protocol. An overlay object is a data object that contains the points needed to specify the shape and size of the overlay and its location on the map. Overlays can represent shapes such as circles, rectangles, multi-segment lines, and simple or complex polygons. You can also define your own custom overlays to represent other shapes.

The presentation of an overlay on screen is handled by an **overlay view**, which is an instance of the `MKOverlayView` class. The job of an overlay view is to draw the shape representing the overlay on top of the map content. For example, an overlay that represents a bus route might have an overlay view that draws the path of the route along with icons showing the stops along that route. The Map Kit framework defines overlay views for the standard types of overlay objects and you can define additional overlay views as needed.

When configuring your map interface, you can add overlay objects at any time. The map view uses the data in each overlay object to determine when the corresponding overlay view needs to appear on screen. When an overlay moves on screen, the map view asks its delegate to create a corresponding overlay view.

Tasks

Accessing Map Properties

`mapType` (page 38) *property*
The type of data displayed by the map view.

`zoomEnabled` (page 41) *property*
A Boolean value that determines whether the user may use pinch gestures to zoom in and out of the map.

`scrollEnabled` (page 39) *property*
A Boolean value that determines whether the user may scroll around the map.

Accessing the Delegate

`delegate` (page 37) *property*
The receiver's delegate.

Manipulating the Visible Portion of the Map

`region` (page 38) *property*
The area currently displayed by the map view.

- `setRegion:animated:` (page 53)
Changes the currently visible region and optionally animates the change.

`centerCoordinate` (page 37) *property*
The map coordinate at the center of the map view.

- `setCenterCoordinate:animated:` (page 52)
Changes the center coordinate of the map and optionally animates the change.

`visibleMapRect` (page 41) *property*
The area currently displayed by the map view.

- `setVisibleMapRect:animated:` (page 53)
Changes the currently visible portion of the map and optionally animates the change.

- `setVisibleMapRect:edgePadding:animated:` (page 54)
Changes the currently visible portion of the map, allowing you to specify additional space around the edges.

Accessing the Device's Current Location

`showsUserLocation` (page 40) *property*
A Boolean value indicating whether the map may display the user location.

`userLocationVisible` (page 40) *property*
A Boolean value indicating whether the device's current location is visible in the map view. (read-only)

`userLocation` (page 40) *property*
The annotation object representing the user's current location. (read-only)

Annotating the Map

`annotations` (page 36) *property*
The complete list of annotations associated with the receiver. (read-only)

- `addAnnotation:` (page 42)
Adds the specified annotation to the map view.

- `addAnnotations:` (page 42)
Adds an array of annotations to the map view.

- `removeAnnotation:` (page 50)
Removes the specified annotation object from the map view.

- `removeAnnotations:` (page 50)
Removes the specified annotation objects from the map view.

- `viewForAnnotation:` (page 54)
Returns the annotation view associated with the specified annotation object, if any.

- `annotationsInMapRect:` (page 43)
Returns the annotation objects located in the specified map rectangle.

`annotationVisibleRect` (page 37) *property*
The visible rectangle where annotation views are currently being displayed. (read-only)

- `dequeueReusableAnnotationViewWithIdentifier:` (page 46)
Returns a reusable annotation view located by its identifier.

Managing Annotation Selections

`selectedAnnotations` (page 39) *property*
The annotations that are currently selected.

- `selectAnnotation:animated:` (page 52)
Selects the specified annotation and displays a callout view for it.

- `deselectAnnotation:animated:` (page 46)
Deselects the specified annotation and hides its callout view.

Adding and Removing Overlays

`overlays` (page 38) *property*
The overlays currently associated with the map view. (read-only)

- `addOverlay:` (page 42)
Adds a single overlay object to the map.

- `addOverlays:` (page 43)
Adds an array of overlay objects to the map.

- `removeOverlay:` (page 51)
Removes a single overlays from the map.

- `removeOverlays:` (page 51)
Removes one or more overlays from the map.

- `insertOverlay:atIndex:` (page 47)
Inserts an overlay into the list of overlay objects associated with the map.

- `exchangeOverlayAtIndex:withOverlayAtIndex:` (page 47)
Exchanges the position of two overlay objects.

- `insertOverlay:aboveOverlay:` (page 47)
Inserts one overlay on top of another.

- `insertOverlay:belowOverlay:` (page 48)
Para

- `viewForOverlay:` (page 54)
Returns the view (if any) associated with the overlay object.

Converting Map Coordinates

- `convertCoordinate:toPointToView:` (page 44)
Converts a map coordinate to a point in the specified view.

- `convertPoint:toCoordinateFromView:` (page 44)
Converts a point in the specified view's coordinate system to a map coordinate.

- `convertRegion:toRectToView:` (page 45)
  Converts a map region to a rectangle in the specified view.
- `convertRect:toRegionFromView:` (page 45)
  Converts a rectangle in the specified view's coordinate system to a map region.

Adjusting Map Regions and Rectangles

- `regionThatFits:` (page 49)
  Adjusts the aspect ratio of the specified region to ensure that it fits in the map view's frame.
- `mapRectThatFits:` (page 48)
  Adjusts the aspect ratio of the specified map rectangle to ensure that it fits in the map view's frame.
- `mapRectThatFits:edgePadding:` (page 49)
  Adjusts the aspect ratio of the specified map rectangle, incorporating the specified inset values.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

annotations

The complete list of annotations associated with the receiver. (read-only)

```
@property(nonatomic, readonly) NSArray *annotations
```

Discussion
The objects in this array must adopt the `MKAnnotation` protocol. If no annotations are associated with the map view, the value of this property is `nil`.

Availability
Available in iOS 3.0 and later.

See Also
- `addAnnotation:` (page 42)
- `addAnnotations:` (page 42)
- `removeAnnotation:` (page 50)
- `removeAnnotations:` (page 50)

Related Sample Code
WeatherMap

Declared In
`MKMapView.h`

annotationVisibleRect

The visible rectangle where annotation views are currently being displayed. (read-only)

```
@property(nonatomic, readonly) CGRect annotationVisibleRect
```

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

centerCoordinate

The map coordinate at the center of the map view.

```
@property(nonatomic) CLLocationCoordinate2D centerCoordinate
```

Discussion
Changing the value in this property centers the map on the new coordinate without changing the current zoom level. It also updates the values in the `region` property to reflect the new center coordinate and the new span values needed to maintain the current zoom level.

Changing the value of this property updates the map view immediately. If you want to animate the change, use the `setCenterCoordinate:animated:` method instead.

Availability
Available in iOS 3.0 and later.

See Also
`- setCenterCoordinate:animated:` (page 52)
`@property region` (page 38)

Declared In
`MKMapView.h`

delegate

The receiver's delegate.

```
@property(nonatomic, assign) id<MKMapViewDelegate> delegate
```

Discussion
A map view sends messages to its delegate regarding the loading of map data and changes in the portion of the map being displayed. The delegate also manages the annotation views used to highlight points of interest on the map.

The delegate should implement the methods of the `MKMapViewDelegate` protocol.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

mapType

The type of data displayed by the map view.

```
@property(nonatomic) MKMapType mapType
```

Discussion
Changing the value in this property may cause the receiver to begin loading new map content. For example, changing from `MKMapTypeStandard` to `MKMapTypeSatellite` might cause it to begin loading the satellite imagery needed for the map. If new data is needed, however, it is loaded asynchronously and appropriate messages are sent to the receiver's delegate indicating the status of the operation.

Availability
Available in iOS 3.0 and later.

Related Sample Code
WorldCities

Declared In
`MKMapView.h`

overlays

The overlays currently associated with the map view. (read-only)

```
@property(nonatomic, readonly) NSArray *overlays
```

Discussion
The objects in this array must adopt the `MKOverlay` protocol. If no overlays are associated with the map view, the value of this property is an empty array.

If the regions defined by two overlays intersect one another, the order of the objects in this array determines the z-order of the corresponding overlay views that are displayed in the map. Overlay objects at the beginning of the array are placed behind those that come later in the array. Thus, the view for an overlay at index `0` is displayed behind the view for the overlay at index `1`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

region

The area currently displayed by the map view.

```
@property(nonatomic) MKCoordinateRegion region
```

Discussion
The region encompasses both the latitude and longitude point on which the map is centered and the span of coordinates to display. The span values provide an implicit zoom value for the map. The larger the displayed area, the lower the amount of zoom. Similarly, the smaller the displayed area, the greater the amount of zoom.

Changing only the center coordinate of the region can still cause the span to change implicitly. This is due to the fact that the distances represented by a span change at different latitudes and longitudes and the map view may need to adjust the span to account for the new location. If you want to change the center coordinate without changing the zoom level, use the `centerCoordinate` instead.

Changing the value of this property updates the map view immediately. When setting this property, the map may adjust the new region value so that it fits the visible area of the map precisely. This is normal and is done to ensure that the value in this property always reflects the visible portion of the map. However, it does mean that if you get the value of this property right after setting it, the returned value may not match the value you set. (You can use the `regionThatFits:` (page 49) method to determine the region that will actually be set by the map.)

If you want to animate the change in region, use the `setRegion:animated:` method instead.

Availability
Available in iOS 3.0 and later.

See Also
- `setRegion:animated:` (page 53)
- `@property centerCoordinate` (page 37)

Related Sample Code
WorldCities

Declared In
`MKMapView.h`

scrollEnabled

A Boolean value that determines whether the user may scroll around the map.

```
@property(nonatomic, getter=isScrollEnabled) BOOL scrollEnabled
```

Discussion
This property controls only user interactions with the map. If you set the value of this property to `NO`, you may still change the map location programmatically by changing the value in the `region` property.

The default value of this property is `YES`.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

selectedAnnotations

The annotations that are currently selected.

```
@property(nonatomic, copy) NSArray *selectedAnnotations
```

Discussion
Assigning a new array to this property selects the first annotation in the array only.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

showsUserLocation

A Boolean value indicating whether the map may display the user location.

```
@property(nonatomic) BOOL showsUserLocation
```

Discussion
This property does not indicate whether the user's position is actually visible on the map, only whether the map view is allowed to display it. To determine whether the user's position is visible, use the `userLocationVisible` property. The default value of this property is `NO`.

Setting this property to `YES` causes the map view to use the Core Location framework to find the current location. As long as this property is `YES`, the map view continues to track the user's location and update it periodically.

Availability
Available in iOS 3.0 and later.

See Also
`@property userLocationVisible` (page 40)

Declared In
`MKMapView.h`

userLocation

The annotation object representing the user's current location. (read-only)

```
@property(nonatomic, readonly) MKUserLocation *userLocation
```

Availability
Available in iOS 3.0 and later.

See Also
`@property showsUserLocation` (page 40)

Declared In
`MKMapView.h`

userLocationVisible

A Boolean value indicating whether the device's current location is visible in the map view. (read-only)

```
@property(nonatomic, readonly, getter=isUserLocationVisible) BOOL userLocationVisible
```

Discussion
This property uses the horizontal accuracy of the current location to determine whether the user's location is visible. Thus, this property is `YES` if the specific coordinate is offscreen but the rectangle surrounding that coordinate (and defined by the horizontal accuracy value) is partially onscreen.

If the user's location cannot be determined, this property contains the value `NO`.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

visibleMapRect

The area currently displayed by the map view.

```
@property(nonatomic) MKMapRect visibleMapRect
```

Discussion
This property represents the same basic information in the `region` (page 38) property but specified as a map rectangle instead of a region.

Changing the value of this property updates the map view immediately. If you want to animate the change, use the `setVisibleMapRect:animated:` method instead.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

zoomEnabled

A Boolean value that determines whether the user may use pinch gestures to zoom in and out of the map.

```
@property(nonatomic, getter=isZoomEnabled) BOOL zoomEnabled
```

Discussion
This property controls only user interactions with the map. If you set the value of this property to `NO`, you may still change the zoom level programmatically by changing the value in the `region` property.

The default value of this property is `YES`.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

Instance Methods

addAnnotation:

Adds the specified annotation to the map view.

```
- (void)addAnnotation:(id < MKAnnotation >)annotation
```

Parameters

annotation

The annotation object to add to the receiver. This object must conform to the `MKAnnotation` protocol. The map view retains the specified object.

Availability
Available in iOS 3.0 and later.

See Also
- `addAnnotations:` (page 42)
- `removeAnnotation:` (page 50)

Declared In
`MKMapView.h`

addAnnotations:

Adds an array of annotations to the map view.

```
- (void)addAnnotations:(NSArray *)annotations
```

Parameters

annotations

An array of annotation objects. Each object in the array must conform to the `MKAnnotation` protocol. The map view retains the individual annotation objects.

Availability
Available in iOS 3.0 and later.

See Also
- `addAnnotation:` (page 42)
- `removeAnnotations:` (page 50)

Declared In
`MKMapView.h`

addOverlay:

Adds a single overlay object to the map.

```
- (void)addOverlay:(id < MKOverlay >)overlay
```

Parameters

overlay

The overlay object to add. This object must conform to the `MKOverlay` protocol.

Discussion

The specified overlay is added to the end of the list of overlay objects. Adding an overlay causes the map view to begin monitoring the area represented by that overlay. As soon as the bounding rectangle of the overlay intersects the visible portion of the map, the map view adds a corresponding overlay view to the map. The overlay view is provided by the `mapView:viewForOverlay:` (page 124) method of the map view's delegate object.

To remove an overlay from a map, you must remove the overlay object using the `removeOverlay:` (page 51) method.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

addOverlays:

Adds an array of overlay objects to the map.

```
- (void)addOverlays:(NSArray *)overlays
```

Parameters

overlays

An array of objects, each of which must conform to the `MKOverlay` protocol.

Discussion

The specified objects are added to the end of the list of overlay objects. Adding an overlay causes the map view to begin monitoring the area represented by that overlay. As soon as the bounding rectangle of the overlay intersects the visible portion of the map, the map view adds a corresponding overlay view to the map. The overlay view is provided by the `mapView:viewForOverlay:` (page 124) method of the map view's delegate object.

To remove an overlay from a map, you must remove the overlay object using the `removeOverlay:` (page 51) method.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

annotationsInMapRect:

Returns the annotation objects located in the specified map rectangle.

```
- (NSSet *)annotationsInMapRect:(MKMapRect)mapRect
```

Parameters

mapRect

The portion of the map that you want to search for annotations.

Return Value

The set of annotation objects located in *mapRect*.

Discussion

This method offers a fast way to retrieve the annotation objects in a particular portion of the map. This method is much faster than doing a linear search of the objects in the `annotations` (page 36) property yourself.

Availability

Available in iOS 4.2 and later.

Declared In

`MKMapView.h`

convertCoordinate:toPointToView:

Converts a map coordinate to a point in the specified view.

```
- (CGPoint)convertCoordinate:(CLLocationCoordinate2D)coordinate toPointToView:(UIView
    *)view
```

Parameters

coordinate

The map coordinate for which you want to find the corresponding point.

view

The view in whose coordinate system you want to locate the specified map coordinate. If this parameter is `nil`, the returned point is specified in the window's coordinate system. If *view* is not `nil`, it must belong to the same window as the map view.

Return Value

The point (in the appropriate view or window coordinate system) corresponding to the specified latitude and longitude value.

Availability

Available in iOS 3.0 and later.

Declared In

`MKMapView.h`

convertPoint:toCoordinateFromView:

Converts a point in the specified view's coordinate system to a map coordinate.

```
- (CLLocationCoordinate2D)convertPoint:(CGPoint)point toCoordinateFromView:(UIView
    *)view
```

Parameters

point

The point you want to convert.

view
The view that serves as the reference coordinate system for the *point* parameter.

Return Value
The map coordinate at the specified point.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

convertRect:toRegionFromView:

Converts a rectangle in the specified view's coordinate system to a map region.

```
- (MKCoordinateRegion)convertRect:(CGRect)rect toRegionFromView:(UIView *)view
```

Parameters

rect
The rectangle you want to convert.

view
The view that serves as the reference coordinate system for the *rect* parameter.

Return Value
The map region corresponding to the specified view rectangle.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

convertRegion:toRectToView:

Converts a map region to a rectangle in the specified view.

```
- (CGRect)convertRegion:(MKCoordinateRegion)region toRectToView:(UIView *)view
```

Parameters

region
The map region for which you want to find the corresponding view rectangle.

view
The view in whose coordinate system you want to locate the specified map region. If this parameter is `nil`, the returned rectangle is specified in the window's coordinate system. If *view* is not `nil`, it must belong to the same window as the map view.

Return Value
The rectangle corresponding to the specified map region.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

dequeueReusableAnnotationViewWithIdentifier:

Returns a reusable annotation view located by its identifier.

```
- (MKAnnotationView *)dequeueReusableAnnotationViewWithIdentifier:(NSString
    *)identifier
```

Parameters

identifier

A string identifying the annotation view to be reused. This is the same string that you specify when initializing the annotation view using the `initWithAnnotation:reuseIdentifier:` (page 20) method.

Return Value
An annotation view with the specified identifier, or `nil` if no such object exists in the reuse queue.

Discussion
For performance reasons, you should generally reuse `MKAnnotationView` objects in your map views. As annotation views move offscreen, the map view moves them to an internally managed reuse queue. As new annotations move onscreen, and your code is prompted to provide a corresponding annotation view, you should always attempt to dequeue an existing view before creating a new one. Dequeueing saves time and memory during performance critical operations such as scrolling.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

deselectAnnotation:animated:

Deselects the specified annotation and hides its callout view.

```
- (void)deselectAnnotation:(id < MKAnnotation >)annotation animated:(BOOL)animated
```

Parameters

annotation

The annotation object to deselect.

animated

If YES, the callout view is animated off screen.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

exchangeOverlayAtIndex:withOverlayAtIndex:

Exchanges the position of two overlay objects.

```
- (void)exchangeOverlayAtIndex:(NSUInteger)index1
    withOverlayAtIndex:(NSUInteger)index2
```

Parameters

index1

The index of the first object in the `overlays` (page 38) array.

index2

The index of the second object in the `overlays` (page 38) array.

Discussion

If either overlay object has an associated view, the position of that view is updated as well. Thus, exchanging views also affects the z-order of overlay views as they appear on the map view.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

insertOverlay:aboveOverlay:

Inserts one overlay on top of another.

```
- (void)insertOverlay:(id < MKOverlay >)overlay aboveOverlay:(id < MKOverlay
    >)sibling
```

Parameters

overlay

The overlay object to insert.

sibling

An existing object in the `overlays` array. This object must exist in the array and must not be `nil`.

Discussion

This method adds the object in *overlay* to the map view and positions it relative to the specified *sibling* object in the `overlays` (page 38) array. This position causes the view associated with *overlay* to be displayed on top of the view associated with *sibling*.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

insertOverlay:atIndex:

Inserts an overlay into the list of overlay objects associated with the map.

```
- (void)insertOverlay:(id < MKOverlay >)overlay atIndex:(NSUInteger)index
```

Parameters

overlay

The overlay object to insert.

index

The index at which to insert the overlay object. If this value is greater than the number of objects in the `overlays` (page 38) property, this method appends the object to the end of the array.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

insertOverlay:belowOverlay:

Para

```
- (void)insertOverlay:(id < MKOverlay >)overlay belowOverlay:(id < MKOverlay
    >)sibling
```

Parameters

overlay

The overlay object to insert.

sibling

An existing object in the `overlays` (page 38) array. This object must exist in the array and must not be `nil`.

Discussion
This method adds the object in *overlay* to the map view and positions it relative to the specified *sibling* object in the `overlays` (page 38) array. This position causes the view associated with *overlay* to be displayed behind the view associated with *sibling*.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

mapRectThatFits:

Adjusts the aspect ratio of the specified map rectangle to ensure that it fits in the map view's frame.

```
- (MKMapRect)mapRectThatFits:(MKMapRect)mapRect
```

Parameters

mapRect

The initial map rectangle whose width and height you want to adjust.

Return Value
A map rectangle that is still centered on the same point of the map but whose width and height are adjusted to fit in the map view's frame.

Discussion
You can use this method to normalize map rectangle values before displaying the corresponding area. This method returns a new map rectangle that both contains the specified rectangle and fits neatly inside the map view's frame.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

mapRectThatFits:edgePadding:

Adjusts the aspect ratio of the specified map rectangle, incorporating the specified inset values.

```
- (MKMapRect)mapRectThatFits:(MKMapRect)mapRect edgePadding:(UIEdgeInsets)insets
```

Parameters

mapRect
The initial map rectangle whose width and height you want to adjust.

insets
The distance (measured in screen points) by which to inset the returned rectangle from the actual boundaries of the map view's frame.

Return Value
A map rectangle that is still centered on the same point of the map but whose width and height are adjusted to fit in the map view's frame minus the inset values.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

regionThatFits:

Adjusts the aspect ratio of the specified region to ensure that it fits in the map view's frame.

```
- (MKCoordinateRegion)regionThatFits:(MKCoordinateRegion)region
```

Parameters

region
The initial region whose span you want to adjust.

Return Value
A region that is still centered on the same point of the map but whose span values are adjusted to fit in the map view's frame.

Discussion
You can use this method to normalize the region values before displaying them in the map. This method returns a new region that both contains the specified region and fits neatly inside the map view's frame.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

removeAnnotation:

Removes the specified annotation object from the map view.

```
- (void)removeAnnotation:(id < MKAnnotation >)annotation
```

Parameters

annotation
The annotation object to remove. This object must conform to the MKAnnotation protocol.

Discussion
If the annotation is currently associated with an annotation view, and that view has a reuse identifier, this method removes the annotation view and queues it internally for later reuse. You can retrieve queued annotation views (and associate them with new annotations) using the dequeueReusableAnnotationViewWithIdentifier: (page 46) method.

Removing an annotation object disassociates it from the map view entirely, preventing it from being displayed on the map. Thus, you would typically call this method only when you want to hide or delete a given annotation.

Availability
Available in iOS 3.0 and later.

See Also
- removeAnnotations: (page 50)
- addAnnotation: (page 42)

Declared In
MKMapView.h

removeAnnotations:

Removes the specified annotation objects from the map view.

```
- (void)removeAnnotations:(NSArray *)annotations
```

Parameters

annotations
The array of annotations to remove. Objects in the array must conform to the MKAnnotation protocol.

Discussion
If any annotation object in the array has an associated annotation view, and if that view has a reuse identifier, this method removes the annotation view and queues it internally for later reuse. You can retrieve queued annotation views (and associate them with new annotations) using the dequeueReusableAnnotationViewWithIdentifier: (page 46) method.

Removing annotation objects disassociates them from the map view entirely, preventing them from being displayed on the map. Thus, you would typically call this method only when you want to hide or delete the specified annotations.

Availability
Available in iOS 3.0 and later.

See Also
- `removeAnnotation:` (page 50)
- `addAnnotations:` (page 42)

Declared In
`MKMapView.h`

removeOverlay:

Removes a single overlays from the map.

```
- (void)removeOverlay:(id < MKOverlay >)overlay
```

Parameters

overlay
The overlay object to remove.

Discussion
Removing an overlay object removes the corresponding overlay view, if one is currently displayed. If the specified object is not currently associated with the map view, this method does nothing.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

removeOverlays:

Removes one or more overlays from the map.

```
- (void)removeOverlays:(NSArray *)overlays
```

Parameters

overlays
An array of objects, each of which conforms to the `MKOverlay` protocol.

Discussion
Removing an overlay object removes the corresponding overlay view, if one is currently displayed. If one or more of the overlay objects are not currently associated with the map view, this method removes the objects that are associated with the map and ignores the rest.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

selectAnnotation:animated:

Selects the specified annotation and displays a callout view for it.

```
- (void)selectAnnotation:(id < MKAnnotation >)annotation animated:(BOOL)animated
```

Parameters

annotation

The annotation object to select.

animated

If `YES`, the callout view is animated into position.

Discussion

If the specified annotation is not onscreen, and therefore does not have an associated annotation view, this method has no effect.

Availability

Available in iOS 3.0 and later.

Declared In

`MKMapView.h`

setCenterCoordinate:animated:

Changes the center coordinate of the map and optionally animates the change.

```
- (void)setCenterCoordinate:(CLLocationCoordinate2D)coordinate
    animated:(BOOL)animated
```

Parameters

coordinate

The new center coordinate for the map.

animated

Specify `YES` if you want the map view to scroll to the new location or `NO` if you want the map to display the new location immediately.

Discussion

Changing the center coordinate centers the map on the new coordinate without changing the current zoom level. It also updates the values in the `region` property to reflect the new center coordinate and the new span values needed to maintain the current zoom level.

Availability

Available in iOS 3.0 and later.

See Also

`@property centerCoordinate` (page 37)

`@property region` (page 38)

Declared In

`MKMapView.h`

setRegion:animated:

Changes the currently visible region and optionally animates the change.

```
- (void)setRegion:(MKCoordinateRegion)region animated:(BOOL)animated
```

Parameters

region

The new region to display in the map view.

animated

Specify `YES` if you want the map view to animate the transition to the new region or `NO` if you want the map to center on the specified region immediately.

Discussion

Changing just the center coordinate of the region can still cause the span values to change implicitly. This is due to the fact that the distances represented by a span change at different latitudes and longitudes and the map view may need to adjust the span to account for the new location. If you want to change the center coordinate without changing the zoom level, use the `setCenterCoordinate:animated:` instead.

When setting a new region, the map may adjust the value in the *region* parameter so that it fits the visible area of the map precisely. This is normal and is done to ensure that the value in the `region` (page 38) property always reflects the visible portion of the map. However, it does mean that if you get the value of that property right after calling this method, the returned value may not match the value you set. (You can use the `regionThatFits:` (page 49) method to determine the region that will actually be set by the map.)

Availability

Available in iOS 3.0 and later.

See Also

`@property region` (page 38)

`- setCenterCoordinate:animated:` (page 52)

Declared In

`MKMapView.h`

setVisibleMapRect:animated:

Changes the currently visible portion of the map and optionally animates the change.

```
- (void)setVisibleMapRect:(MKMapRect)mapRect animated:(BOOL)animate
```

Parameters

mapRect

The map rectangle to make visible in the map view.

animate

Specify `YES` if you want the map view to animate the transition to the new map rectangle or `NO` if you want the map to center on the specified rectangle immediately.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

setVisibleMapRect:edgePadding:animated:

Changes the currently visible portion of the map, allowing you to specify additional space around the edges.

```
- (void)setVisibleMapRect:(MKMapRect)mapRect edgePadding:(UIEdgeInsets)insets
    animated:(BOOL)animate
```

Parameters

mapRect

The map rectangle to make visible in the map view.

insets

The amount of additional space (measured in screen points) to make visible around the specified rectangle.

animate

Specify `YES` if you want the map view to animate the transition to the new map rectangle or `NO` if you want the map to center on the specified rectangle immediately.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

viewForAnnotation:

Returns the annotation view associated with the specified annotation object, if any.

```
- (MKAnnotationView *)viewForAnnotation:(id < MKAnnotation >)annotation
```

Parameters

annotation

The annotation object whose view you want.

Return Value
The annotation view or `nil` if the view has not yet been created. This method may also return `nil` if the annotation is not in the visible map region and therefore does not have an associated annotation view.

Availability
Available in iOS 3.0 and later.

Declared In
`MKMapView.h`

viewForOverlay:

Returns the view (if any) associated with the overlay object.

```
- (MKOverlayView *)viewForOverlay:(id < MKOverlay >)overlay
```

Parameters

overlay

The overlay object whose view you want.

Return Value
The view associated with the overlay object or `nil` if the overlay is not on screen.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

Constants

MKMapType

The type of map to display.

```
enum {
    MKMapTypeStandard,
    MKMapTypeSatellite,
    MKMapTypeHybrid
};
typedef NSUInteger MKMapType;
```

Constants

`MKMapTypeStandard`
Displays a street map that shows the position of all roads and some road names.
Available in iOS 3.0 and later.
Declared in `MKTypes.h`.

`MKMapTypeSatellite`
Displays satellite imagery of the area.
Available in iOS 3.0 and later.
Declared in `MKTypes.h`.

`MKMapTypeHybrid`
Displays a satellite image of the area with road and road name information layered on top.
Available in iOS 3.0 and later.
Declared in `MKTypes.h`.

MKMultiPoint Class Reference

Inherits from	MKShape : NSObject
Conforms to	MKAnnotation (MKShape) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKMultiPoint.h

Overview

The `MKMultiPoint` class is an abstract superclass used to define shapes composed of multiple points. You should not create instances of this class directly. Instead, you should create instances of the `MKPolyline` or `MKPolygon` classes. However, you can use the method and properties of this class to access information about the specific points associated with the line or polygon.

Tasks

Accessing the Points in the Shape

`points` (page 58) *property*
The array of points associated with the shape. (read-only)

`pointCount` (page 58) *property*
The number of points associated with the shape. (read-only)

Getting Coordinate Values

`- getCoordinates:range:` (page 58)
Retrieves one or more points associated with the shape and converts them to coordinate values.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

pointCount

The number of points associated with the shape. (read-only)

```
@property (nonatomic, readonly) NSUInteger pointCount
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKMultiPoint.h`

points

The array of points associated with the shape. (read-only)

```
@property (nonatomic, readonly) MKMapPoint *points
```

Discussion
The number of points in the array is specified by the `pointCount` (page 58) property.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMultiPoint.h`

Instance Methods

getCoordinates:range:

Retrieves one or more points associated with the shape and converts them to coordinate values.

```
- (void)getCoordinates:(CLLocationCoordinate2D *)coords range:(NSRange)range
```

Parameters

coords

On input, you must provide a C array of structures large enough to hold the desired number of coordinates. On output, this structure contains the requested coordinate data.

range

The range of points you want. The `location` field indicates the first point you are requesting, with `0` being the first point, `1` being the second point, and so on. The `length` field indicates the number of points you want. The array in *coords* must be large enough to accommodate the number of requested coordinates.

Discussion
This method converts the map points into coordinates before returning them to you. If you want the value of each point specified as a map point, you can access the values directly using the `points` (page 58) property.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMultiPoint.h`

MKOverlayPathView Class Reference

Inherits from	MKOverlayView : UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKOverlayPathView.h

Overview

The `MKOverlayPathView` class represents a generic overlay that draws its contents using a `CGPathRef` data type. You can use this class to implement simple path-based overlay views or subclass it to define additional drawing behaviors. The default drawing behavior of this class is to apply the object's current fill attributes, fill the path, apply the current stroke attributes, and then stroke the path.

If you subclass, you should override the `createPath` (page 66) method and use that method to build the appropriate path for the overlay. You can invalidate this path as needed and force the path to be recreated using whatever new data your subclass has obtained.

Tasks

Accessing the Drawing Attributes

`fillColor` (page 62) *property*
The fill color to use for the path.

`strokeColor` (page 65) *property*
The stroke color to use for the path.

`lineWidth` (page 64) *property*
The stroke width to use for the path.

`lineJoin` (page 64) *property*
The line join style to apply to corners of the path.

`lineCap` (page 63) *property*
The line cap style to apply to the open ends of the path.

`miterLimit` (page 64) *property*
The limiting value that helps avoid spikes at junctions between connected line segments.

`lineDashPhase` (page 63) *property*
The

`lineDashPattern` (page 63) *property*
An array of numbers indicating the dash pattern for paths.

Creating and Managing the Path

`path` (page 64) *property*
The current path to use when drawing the overlay.

- `createPath` (page 66)
Creates the path for the overlay.

- `invalidatePath` (page 67)
Releases the path associated with the receiver.

Drawing the Path

- `applyStrokePropertiesToContext:atZoomScale:` (page 66)
Applies the receiver's current stroke-related drawing properties to the specified graphics context.

- `applyFillPropertiesToContext:atZoomScale:` (page 65)
Applies the receiver's current fill-related drawing properties to the specified graphics context

- `strokePath:inContext:` (page 67)
Draws a line along the specified path.

- `fillPath:inContext:` (page 67)
Fills the area enclosed by the specified path.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

fillColor

The fill color to use for the path.

```
@property (retain) UIColor *fillColor
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

lineCap

The line cap style to apply to the open ends of the path.

```
@property CGLineCap lineCap
```

Discussion
The line cap style is applied to the start and end points of any open subpaths. This property does not affect closed subpaths. The default line cap style is `kCGLineCapButt`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

lineDashPattern

An array of numbers indicating the dash pattern for paths.

```
@property (copy) NSArray *lineDashPattern
```

Discussion
The array contains one or more `NSNumber` objects that indicate 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.

This property is set to `nil` by default, which indicates no line dash pattern.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

lineDashPhase

The

```
@property CGFloat lineDashPhase
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

lineJoin

The line join style to apply to corners of the path.

```
@property CGLineJoin lineJoin
```

Discussion
The default line join style is `kCGLineJoinMiter`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

lineWidth

The stroke width to use for the path.

```
@property CGFloat lineWidth
```

Discussion
The default value of this property is 0.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

miterLimit

The limiting value that helps avoid spikes at junctions between connected line segments.

```
@property CGFloat miterLimit
```

Discussion
The miter limit helps you avoid spikes in paths that use the `kCGLineJoinMiter` join style. If the ratio of the miter length—that is, 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 is 10, which results in the conversion of miters whose angle at the joint is less than 11 degrees.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

path

The current path to use when drawing the overlay.

```
@property CGPathRef path
```

Discussion
Getting the value of this property causes the path to be created (using the `createPath` (page 66) method) if it does not already exist. You can also assign a path object to this property explicitly.

When assigning a new path object to this property, the receiver retains the path you specify.

Availability
Available in iOS 4.0 and later.

See Also
- `createPath` (page 66)
- `invalidatePath` (page 67)

Declared In
`MKOverlayPathView.h`

strokeColor

The stroke color to use for the path.

```
@property (retain) UIColor *strokeColor
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

Instance Methods

applyFillPropertiesToContext:atZoomScale:

Applies the receiver's current fill-related drawing properties to the specified graphics context

```
- (void)applyFillPropertiesToContext:(CGContextRef)context
    atZoomScale:(MKZoomScale)zoomScale
```

Parameters

context
The graphics context used to draw the view's contents.

zoomScale
The current zoom scale used for drawing.

Discussion
This is a convenience method for applying all of the drawing properties used when filling a path. This method applies the current fill color to the specified graphics context.

Availability
Available in iOS 4.0 and later.

See Also
- `fillPath:inContext:` (page 67)

Declared In
`MKOverlayPathView.h`

applyStrokePropertiesToContext:atZoomScale:

Applies the receiver's current stroke-related drawing properties to the specified graphics context.

```
- (void)applyStrokePropertiesToContext:(CGContextRef)context
    atZoomScale:(MKZoomScale)zoomScale
```

Parameters

context
The graphics context used to draw the view's contents.

zoomScale
The current zoom scale used for drawing.

Discussion
This is a convenience method for applying all of the drawing properties used when stroking a path. This method applies the stroke color, line width, line join, line cap, miter limit, line dash phase, and line dash attributes to the specified graphics context. This method applies the scale factor in the *zoomScale* parameter to the line width and line dash pattern automatically so that lines scale appropriately.

This method does not save the current graphics state before applying the new attributes. You must save it yourself and restore it later when you are done drawing.

Availability
Available in iOS 4.0 and later.

See Also
- `strokePath:inContext:` (page 67)

Declared In
`MKOverlayPathView.h`

createPath

Creates the path for the overlay.

```
- (void)createPath
```

Discussion
The default implementation of this method does nothing. Subclasses should override it and use it to create the `CGPathRef` data type to be used for drawing. After creating the path, your implementation should then assign it to the `path` (page 64) property.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayPathView.h`

fillPath:inContext:

Fills the area enclosed by the specified path.

```
- (void)fillPath:(CGPathRef)path inContext:(CGContextRef)context
```

Parameters

path

The path to fill.

context

The graphics context in which to draw the path.

Discussion

You must set the current fill color before calling this method. Typically you do this by calling the `applyFillPropertiesToContext:atZoomScale:` method prior to drawing. If the `fillColor` (page 62) property is currently `nil`, this method does nothing.

Availability

Available in iOS 4.0 and later.

See Also

`- applyFillPropertiesToContext:atZoomScale:` (page 65)

Declared In

`MKOverlayPathView.h`

invalidatePath

Releases the path associated with the receiver.

```
- (void)invalidatePath
```

Discussion

You can call this method at any time where a change in the path information would require you to recreate the path. This method sets the `path` (page 64) property to `nil`, which causes the cached path to be released.

Availability

Available in iOS 4.0 and later.

Declared In

`MKOverlayPathView.h`

strokePath:inContext:

Draws a line along the specified path.

```
- (void)strokePath:(CGPathRef)path inContext:(CGContextRef)context
```

Parameters

path

The path to draw.

context

The graphics context in which to draw the path.

Discussion
You must set the current stroke color before calling this method. Typically you do this by calling the `applyStrokePropertiesToContext:atZoomScale:` method prior to drawing. If the `strokeColor` (page 65) property is currently `nil`, this method does nothing.

Availability
Available in iOS 4.0 and later.

See Also
`- applyStrokePropertiesToContext:atZoomScale:` (page 66)

Declared In
`MKOverlayPathView.h`

MKOverlayView Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKOverlayView.h

Overview

The `MKOverlayView` class defines the basic behavior associated with all overlay views. An overlay view provides the visual representation of an overlay object—that is, an object that conforms to the `MKOverlay` protocol. This class defines the drawing infrastructure used by the map view but does not do any actual drawing. Subclasses are expected to override the `drawMapRect:zoomScale:inContext:` (page 72) method in order to draw the contents of the overlay view.

The Map Kit framework provides several concrete instances of overlay views. Specifically, it provides overlay views for each of the concrete overlay objects. You can use one of these existing overlay views or define your own subclass if you want to draw the overlay contents differently.

Subclassing Notes

You can subclass `MKOverlayView` to create overlays based on custom shapes and content. The only method subclasses are expected to override is the `drawMapRect:zoomScale:inContext:` (page 72) method. However, if your class contains content that may not be ready for drawing right away, you should also override the `canDrawMapRect:zoomScale:` (page 71) method and use it to report when your class is ready and able to draw.

The implementation of your `drawMapRect:zoomScale:inContext:` (page 72) method must be safe to run from multiple threads simultaneously. To improve performance, the map view may tile overlays that are large enough and distribute the rendering of each tile to separate threads.

Tasks

Initializing an Overlay View

- `initWithOverlay:` (page 73)
  Initializes and returns the overlay view and associates it with the specified overlay object.

Attributes of the Overlay

`overlay` (page 71) *property*
The overlay object containing the data for drawing. (read-only)

Converting Points on the Map

- `pointForMapPoint:` (page 74)
  Returns the point in the overlay view that corresponds to specified point on the map.
- `mapPointForPoint:` (page 73)
  Returns the map point that corresponds to the specified point in the overlay view.
- `rectForMapRect:` (page 74)
  Returns the rectangle in the overlay view that corresponds to the specified rectangle on the map.
- `mapRectForRect:` (page 74)
  Returns the map rectangle that corresponds to the rectangle in the overlay view's coordinate system.

Drawing the Overlay

- `canDrawMapRect:zoomScale:` (page 71)
  Returns a Boolean value indicating whether the overlay view is ready to draw its content.
- `drawMapRect:zoomScale:inContext:` (page 72)
  Draws the contents of the overlay view.
- `setNeedsDisplayInMapRect:` (page 75)
  Invalidates the view in the given map rectangle at all zoom scales.
- `setNeedsDisplayInMapRect:zoomScale:` (page 75)
  Invalidates the view in the given map rectangle but only at the specified zoom scale.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

overlay

The overlay object containing the data for drawing. (read-only)

```
@property (nonatomic, readonly) id <MKOverlay> overlay
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayView.h`

Instance Methods

canDrawMapRect:zoomScale:

Returns a Boolean value indicating whether the overlay view is ready to draw its content.

```
- (BOOL)canDrawMapRect:(MKMapRect)mapRect zoomScale:(MKZoomScale)zoomScale
```

Parameters

mapRect
The map rectangle that needs to be updated.

zoomScale
The current scale factor applied to the map.

Return Value
`YES` if this view is ready to draw its contents or `NO` if it is not.

Discussion
Overlay views can override this method in situations where they may depend on the availability of other information to draw their contents. For example, an overlay view showing traffic information might want to delay drawing until it has all of the traffic data it needs. In such a case, it can return `NO` from this method to indicate that it is not ready.

If you return `NO` from this method, your application is responsible for calling the `setNeedsDisplayInMapRect:zoomScale:` (page 75) method when the overlay view subsequently becomes ready to draw its contents.

The default implementation of this method returns `YES`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayView.h`

drawMapRect:zoomScale:inContext:

Draws the contents of the overlay view.

```
- (void)drawMapRect:(MKMapRect)mapRect zoomScale:(MKZoomScale)zoomScale
    inContext:(CGContextRef)context
```

Parameters

mapRect
The map rectangle that needs to be updated. You can use this rectangle to limit drawing to only the portion of the view that changed.

zoomScale
The current scale factor applied to the map content. You can use this value for configuring the stroke width of lines or other attributes that might be affected by the scale of the view's content.

context
The graphics context to use for drawing the view's content.

Discussion
The default implementation of this method does nothing. Subclasses are expected to override this method (instead of the `drawRect:` method) and use it to draw the contents of the view.

In your drawing code, you should specify the position of any rendered content relative to the map itself and not relative to the view's bounds or frame. In other words, compute the position and size of any overlay content using map points and map rectangles, convert those values to `CGPoint` and `CGRect` types (using the methods of this class), and then use the converted points to build paths or specify the rendering location for items.

You should also not make assumptions that the view's frame matches the bounding rectangle of the overlay. The view's frame is actually bigger than the bounding rectangle to allow you to draw lines for things like roads that might be located directly on the border of that rectangle. For some types of content, such as gradients, this also means that you might need to apply a clipping rectangle to *context* to ensure drawing is contained to the correct area.

It is recommended that you use Core Graphics to draw any content for your overlays. If you choose to use UIKit classes and methods for drawing instead, you must push the specified graphics context onto the context stack (using the `UIGraphicsPushContext` function) before making any drawing calls. When you are done drawing, you must similarly pop the graphics context off the stack using the `UIGraphicsPopContext`. During drawing, you may draw content normally but should avoid manipulating views and other classes that are safe to use only from the application's main thread.

To improve drawing performance, the map view may tile overlays that become large enough and render the tiles from separate threads. Your implementation of this method must therefore be capable of safely running from multiple threads simultaneously. In addition, you should avoid drawing the entire contents of the overlay each time this method is called. Instead, your implementation should always take the *mapRect* parameter into consideration and avoid drawing content outside that rectangle. Failure to do so could lead to performance problems.

Availability
Available in iOS 4.0 and later.

See Also
- `canDrawMapRect:zoomScale:` (page 71)
- `pointForMapPoint:` (page 74)
- `rectForMapRect:` (page 74)

Declared In
`MKOverlayView.h`

initWithOverlay:

Initializes and returns the overlay view and associates it with the specified overlay object.

```
- (id)initWithOverlay:(id <MKOverlay>)overlay
```

Parameters

overlay

The overlay object to use when drawing the overlay on the map. This object provides the data needed to draw the overlay's shape. This object is retained by the overlay view.

Return Value
An initialized overlay object.

Discussion
Upon initialization, the frame of the overlay view is set to `CGRectZero`. The map view sets the size and position of the view at display time, and you should not change those values yourself.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayView.h`

mapPointForPoint:

Returns the map point that corresponds to the specified point in the overlay view.

```
- (MKMapPoint)mapPointForPoint:(CGPoint)point
```

Parameters

point

The point in the view's coordinate system that you want to convert.

Return Value
The point on the two-dimensional map projection corresponding to the specified point.

Availability
Available in iOS 4.0 and later.

See Also
- `pointForMapPoint:` (page 74)

Declared In
`MKOverlayView.h`

mapRectForRect:

Returns the map rectangle that corresponds to the rectangle in the overlay view's coordinate system.

```
- (MKMapRect)mapRectForRect:(CGRect)rect
```

Parameters

rect

The rectangle specified in the receiver's coordinate system.

Return Value

The rectangle on the two-dimensional map project that corresponds to the specified view rectangle.

Availability

Available in iOS 4.0 and later.

See Also

- rectForMapRect: (page 74)

Declared In

MKOverlayView.h

pointForMapPoint:

Returns the point in the overlay view that corresponds to specified point on the map.

```
- (CGPoint)pointForMapPoint:(MKMapPoint)mapPoint
```

Parameters

mapPoint

A point on the two-dimensional map projection. If you have a coordinate value (latitude and longitude), you can use the MKMapPointForCoordinate (page 143) function to convert that coordinate to a map point.

Return Value

The point in the receiver's coordinate system that corresponds to the map point.

Availability

Available in iOS 4.0 and later.

See Also

- mapPointForPoint: (page 73)

Declared In

MKOverlayView.h

rectForMapRect:

Returns the rectangle in the overlay view that corresponds to the specified rectangle on the map.

```
- (CGRect)rectForMapRect:(MKMapRect)mapRect
```

Parameters

mapRect

A rectangle on the two-dimensional map projection.

Return Value

The rectangle specified in the receiver's coordinate system.

Availability

Available in iOS 4.0 and later.

See Also

- mapRectForRect: (page 74)

Declared In

MKOverlayView.h

setNeedsDisplayInMapRect:

Invalidates the view in the given map rectangle at all zoom scales.

```
- (void)setNeedsDisplayInMapRect:(MKMapRect)mapRect
```

Parameters

mapRect

The portion of the overlay that needs to be updated. This value is specified using a map rectangle and not view coordinates. You can convert from a view rectangle to a map rectangle using the mapRectForRect: (page 74) method.

Discussion

Marking a rectangle as invalid causes that portion of the view to be redrawn during the next update cycle. This method invalidates the overlay regardless of the current zoom scale associated with the map.

Availability

Available in iOS 4.0 and later.

Declared In

MKOverlayView.h

setNeedsDisplayInMapRect:zoomScale:

Invalidates the view in the given map rectangle but only at the specified zoom scale.

```
- (void)setNeedsDisplayInMapRect:(MKMapRect)mapRect zoomScale:(MKZoomScale)zoomScale
```

Parameters

mapRect

The portion of the overlay that needs to be updated. This value is specified using a map rectangle and not view coordinates. You can convert from a view rectangle to a map rectangle using the mapRectForRect: (page 74) method.

zoomScale

The zoom scale for which you want to invalidate the overlay.

Discussion
Marking a rectangle as invalid causes that portion of the view to be redrawn during the next update cycle. This method invalidates the overlay only when it is drawn at the specified zoom scale.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlayView.h`

MKPinAnnotationView Class Reference

Inherits from	MKAnnotationView : UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKAnnotationView.h
Related sample code	MapCallouts

Overview

The `MKPinAnnotationView` class provides a concrete annotation view that displays a pin icon like the ones found in the Maps application. Using this class, you can configure the type of pin to drop and whether you want the pin to be animated into place.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Tasks

Getting and Setting Attributes

`pinColor` (page 78) *property*
The color of the pin head.

`animatesDrop` (page 78) *property*
A Boolean value indicating whether the annotation view is animated onto the screen.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

animatesDrop

A Boolean value indicating whether the annotation view is animated onto the screen.

```
@property (nonatomic) BOOL animatesDrop
```

Discussion
When this property is `YES`, the map view animates the appearance of pin annotation views by making them appear to drop onto the map at the target point. This animation occurs whenever the view transitions from offscreen to onscreen.

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
`MKPinAnnotationView.h`

pinColor

The color of the pin head.

```
@property (nonatomic) MKPinAnnotationColor pinColor
```

Discussion
The Maps application uses different pin colors for different types of map annotations. Your own map annotation should use the available pin colors in the same way. For a description of when to use each type of pin, see the constants of "`MKPinAnnotationColor`" (page 79).

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
`MKPinAnnotationView.h`

Constants

MKPinAnnotationColor

The supported colors for pin annotations.

```
enum {
    MKPinAnnotationColorRed = 0,
    MKPinAnnotationColorGreen,
    MKPinAnnotationColorPurple
};
typedef NSUInteger MKPinAnnotationColor;
```

Constants

`MKPinAnnotationColorRed`

The head of the pin is red. Red pins indicate destination points on the map.

Available in iOS 3.0 and later.

Declared in `MKPinAnnotationView.h`.

`MKPinAnnotationColorGreen`

The head of the pin is green. Green pins indicate starting points on the map.

Available in iOS 3.0 and later.

Declared in `MKPinAnnotationView.h`.

`MKPinAnnotationColorPurple`

The head of the pin is purple. Purple pins indicate user-specified points on the map.

Available in iOS 3.0 and later.

Declared in `MKPinAnnotationView.h`.

MKPlacemark Class Reference

Inherits from	NSObject
Conforms to	MKAnnotation NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKPlacemark.h

Overview

A `MKPlacemark` object stores placemark data for a given latitude and longitude. Placemark data includes information such as the country, state, city, and street address associated with the specified coordinate. Placemark objects are typically generated by a `MKReverseGeocoder` object, although you can also create them explicitly yourself.

A placemark is also an annotation and conforms to the `MKAnnotation` protocol, whose properties and methods include the placemark coordinate and other information. Because they are annotations, you can add them directly to the map view.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Tasks

Initializing a Placemark Object

`- initWithCoordinate:addressDictionary:` (page 85)
Initializes and returns a placemark object using the specified coordinate and Address Book dictionary.

Accessing the Placemark Attributes

`addressDictionary` (page 82) *property*
A dictionary containing the Address Book keys and values for the placemark. (read-only)

`thoroughfare` (page 85) *property*
The street address associated with the placemark. (read-only)

`subThoroughfare` (page 85) *property*
Additional street-level information for the placemark. (read-only)

`locality` (page 83) *property*
The city associated with the placemark. (read-only)

`subLocality` (page 84) *property*
Additional city-level information for the placemark. (read-only)

`administrativeArea` (page 83) *property*
The state associated with the placemark. (read-only)

`subAdministrativeArea` (page 84) *property*
Additional administrative area information for the placemark. (read-only)

`postalCode` (page 84) *property*
The postal code associated with the placemark. (read-only)

`country` (page 83) *property*
The name of the country associated with the placemark. (read-only)

`countryCode` (page 83) *property*
The abbreviated country name. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

addressDictionary

A dictionary containing the Address Book keys and values for the placemark. (read-only)

```
@property (nonatomic, readonly) NSDictionary *addressDictionary
```

Discussion
The keys in this dictionary are those defined by the Address Book framework and used to access address information for a person. For a list of the strings that might be in this dictionary, see the "Address Property" constants in *ABPerson Reference*.

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

administrativeArea

The state associated with the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *administrativeArea
```

Discussion
If the placemark location was Apple's headquarters, the value for this property would be the string "CA" or "California".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

country

The name of the country associated with the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *country
```

Discussion
If the placemark location was Apple's headquarters, the value for this property would be the string "United States".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

countryCode

The abbreviated country name. (read-only)

```
@property (nonatomic, readonly) NSString *countryCode
```

Discussion
This string is the standard abbreviation used to refer to the country. For example, if the placemark location was Apple's headquarters, the value for this property would be the string "US".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

locality

The city associated with the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *locality
```

Discussion
If the placemark location was Apple's headquarters, the value for this property would be the string "Cupertino".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

postalCode

The postal code associated with the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *postalCode
```

Discussion
If the placemark location was Apple's headquarters, the value for this property would be the string "95014".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

subAdministrativeArea

Additional administrative area information for the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *subAdministrativeArea
```

Discussion
Subadministrative areas typically correspond to counties or other regions that are then organized into a larger administrative area or state. For example, if the placemark location was Apple's headquarters, the value for this property would be the string "Santa Clara", which is the county in California that contains the city of Cupertino.

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

subLocality

Additional city-level information for the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *subLocality
```

Discussion
This property contains additional information, such as the name of the neighborhood or landmark associated with the placemark. It might also refer to a common name that is associated with the location.

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

subThoroughfare

Additional street-level information for the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *subThoroughfare
```

Discussion
Subthroughfares provide information such as the street number for the location. For example, if the placemark location was Apple's headquarters (1 Infinite Loop), the value for this property would be the string "1".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

thoroughfare

The street address associated with the placemark. (read-only)

```
@property (nonatomic, readonly) NSString *thoroughfare
```

Discussion
The street address contains the street name. For example, if the placemark location was Apple's headquarters, the value for this property would be the string "Infinite Loop".

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

Instance Methods

initWithCoordinate:addressDictionary:

Initializes and returns a placemark object using the specified coordinate and Address Book dictionary.

```
- (id)initWithCoordinate:(CLLocationCoordinate2D)coordinate
    addressDictionary:(NSDictionary *)addressDictionary
```

Parameters

coordinate
The map coordinate to associate with the placemark.

addressDictionary

A dictionary containing keys and values from an Address Book record. For a list of strings that you can use for the keys of this dictionary, see the "Address Property" constants in *ABPerson Reference*. All of the keys in should be at the top level of the dictionary.

Return Value
An initialized `MKPlacemark` object.

Discussion
You can create placemark objects manually for entities for which you already have address information, such as contacts in the Address Book. Creating a placemark object explicitly avoids the need to query the reverse geocoder object for the same information.

Availability
Available in iOS 3.0 and later.

Declared In
`MKPlacemark.h`

MKPointAnnotation Class Reference

Inherits from	MKShape : NSObject
Conforms to	MKAnnotation (MKShape) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKPointAnnotation.h

Overview

The `MKPointAnnotation` class defines a concrete annotation object located at a specified point. You can use this class, rather than define your own, in situations where all you want to do is associate a point on the map with a title.

Tasks

Accessing the Annotation's Location

`coordinate` (page 87) *property*
The coordinate point of the annotation, specified as a latitude and longitude.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

coordinate

The coordinate point of the annotation, specified as a latitude and longitude.

```
@property (nonatomic, assign) CLLocationCoordinate2D coordinate
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKPointAnnotation.h`

MKPolygon Class Reference

Inherits from	MKMultiPoint : MKShape : NSObject
Conforms to	MKOverlay MKAnnotation (MKShape) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKPolygon.h

Overview

The `MKPolygon` class represents a shape consisting of one or more points that define a closed polygon. The points are connected end-to-end in the order they are provided. The first and last points are connected to each other to create the closed shape.

When creating a polygon, you can mask out portions of the polygon by specifying one or more interior polygons. Areas that are masked by an interior polygon are not considered part of the polygon's occupied area.

Tasks

Creating a Polygon Overlay

`+ polygonWithPoints:count:` (page 91)
Creates and returns an `MKPolygon` object from the specified set of map points.

`+ polygonWithPoints:count:interiorPolygons:` (page 91)
Creates and returns an `MKPolygon` object from the specified set of map points and interior polygons.

`+ polygonWithCoordinates:count:` (page 90)
Creates and returns an `MKPolygon` object from the specified set of coordinates.

`+ polygonWithCoordinates:count:interiorPolygons:` (page 91)
Creates and returns an `MKPolygon` object from the specified set of coordinates and interior polygons.

Accessing the Interior Polygons

`interiorPolygons` (page 90) *property*
The array of polygons nested inside the receiver. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

interiorPolygons

The array of polygons nested inside the receiver. (read-only)

```
@property (readonly) NSArray *interiorPolygons
```

Discussion
When a polygon is rendered on screen, the area occupied by any interior polygons is masked out and not considered part of the polygon.

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolygon.h`

Class Methods

polygonWithCoordinates:count:

Creates and returns an `MKPolygon` object from the specified set of coordinates.

```
+ (MKPolygon *)polygonWithCoordinates:(CLLocationCoordinate2D *)coords
    count:(NSUInteger)count
```

Parameters
coords
The array of coordinates defining the shape. The data in this array is copied to the new object.
count
The number of items in the *coords* array.

Return Value
A new polygon object.

Availability
Available in iOS 4.0 and later.

Declared In
MKPolygon.h

polygonWithCoordinates:count:interiorPolygons:

Creates and returns an MKPolygon object from the specified set of coordinates and interior polygons.

```
+ (MKPolygon *)polygonWithCoordinates:(CLLocationCoordinate2D *)coords
    count:(NSUInteger)count interiorPolygons:(NSArray *)interiorPolygons
```

Parameters

coords
The array of coordinates defining the shape. The data in this array is copied to the new object.

count
The number of items in the *coords* array.

interiorPolygons
An array of MKPolygon objects that define one or more cutout regions for the receiver's polygon.

Return Value
A new polygon object.

Availability
Available in iOS 4.0 and later.

Declared In
MKPolygon.h

polygonWithPoints:count:

Creates and returns an MKPolygon object from the specified set of map points.

```
+ (MKPolygon *)polygonWithPoints:(MKMapPoint *)points count:(NSUInteger)count
```

Parameters

points
The array of map points defining the shape. The data in this array is copied to the new object.

count
The number of items in the *points* array.

Return Value
A new polygon object.

Availability
Available in iOS 4.0 and later.

Declared In
MKPolygon.h

polygonWithPoints:count:interiorPolygons:

Creates and returns an MKPolygon object from the specified set of map points and interior polygons.

```
+ (MKPolygon *)polygonWithPoints:(MKMapPoint *)points count:(NSUInteger)count
    interiorPolygons:(NSArray *)interiorPolygons
```

Parameters

points

The array of map points defining the shape. The data in this array is copied to the new object.

count

The number of items in the *points* array.

interiorPolygons

An array of `MKPolygon` objects that define one or more cutout regions for the receiver's polygon.

Return Value
A new polygon object.

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolygon.h`

MKPolygonView Class Reference

Inherits from	MKOverlayPathView : MKOverlayView : UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKPolygonView.h

Overview

The `MKPolygonView` class provides the visual representation for an `MKPolygon` annotation object. This view fills and strokes the area represented by the annotation. You can change the color and other drawing attributes of the polygon by modifying the properties inherited from the `MKOverlayPathView` class. This class is typically used as is and not subclassed.

Tasks

MethodGroup

`- initWithPolygon:` (page 94)
Initializes and returns a new overlay view using the specified polygon overlay object.

`polygon` (page 94) *property*
The polygon overlay object that contains the information used to draw the overlay. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

polygon

The polygon overlay object that contains the information used to draw the overlay. (read-only)

```
@property (nonatomic, readonly) MKPolygon *polygon
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolygonView.h`

Instance Methods

initWithPolygon:

Initializes and returns a new overlay view using the specified polygon overlay object.

```
- (id)initWithPolygon:(MKPolygon *)polygon
```

Parameters

polygon

The polygon overlay containing the information about the area to be drawn. This object must have at least three points defining the polygon in order for this view to draw the corresponding path.

Return Value
A new polygon overlay view.

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolygonView.h`

MKPolyline Class Reference

Inherits from	MKMultiPoint : MKShape : NSObject
Conforms to	MKOverlay MKAnnotation (MKShape) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKPolyline.h

Overview

The `MKPolyline` class represents a shape consisting of one or more points that define connecting line segments. The points are connected end-to-end in the order they are provided. The first and last points are not connected to each other.

Tasks

Creating a Polyline Overlay

`+ polylineWithPoints:count:` (page 96)
Creates and returns an `MKPolyline` object from the specified set of map points.

`+ polylineWithCoordinates:count:` (page 95)
Creates and returns an `MKPolyline` object from the specified set of coordinates.

Class Methods

polylineWithCoordinates:count:

Creates and returns an `MKPolyline` object from the specified set of coordinates.

```
+ (MKPolyline *)polylineWithCoordinates:(CLLocationCoordinate2D *)coords
    count:(NSUInteger)count
```

Parameters

coords

The array of coordinates defining the shape. The data in this array is copied to the new object.

count

The number of items in the *coords* array.

Return Value
A new polyline object.

Availability
Available in iOS 4.0 and later.

Declared In
MKPolyline.h

polylineWithPoints:count:

Creates and returns an `MKPolyline` object from the specified set of map points.

```
+ (MKPolyline *)polylineWithPoints:(MKMapPoint *)points count:(NSUInteger)count
```

Parameters

points

The array of map points defining the shape. The data in this array is copied to the new object.

count

The number of items in the *points* array.

Return Value
A new polyline object.

Availability
Available in iOS 4.0 and later.

Declared In
MKPolyline.h

MKPolylineView Class Reference

Inherits from	MKOverlayPathView : MKOverlayView : UIView : UIResponder : NSObject
Conforms to	NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKPolylineView.h

Overview

The `MKPolylineView` class provides the visual representation for an `MKPolyline` annotation object. This view strokes the path represented by the annotation. (This class does not fill the area enclosed by the path.) You can change the color and other drawing attributes of the path by modifying the properties inherited from the `MKOverlayPathView` class. This class is typically used as is and not subclassed.

Tasks

MethodGroup

`- initWithPolyline:` (page 98)
Initializes and returns a new overlay view using the specified polyline overlay object

`polyline` (page 98) *property*
The polyline overlay object that contains the information used to draw the overlay. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

polyline

The polyline overlay object that contains the information used to draw the overlay. (read-only)

```
@property (nonatomic, readonly) MKPolyline *polyline
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolylineView.h`

Instance Methods

initWithPolyline:

Initializes and returns a new overlay view using the specified polyline overlay object

```
- (id)initWithPolyline:(MKPolyline *)polyline
```

Parameters

polyline

The polyline overlay object containing the information about the path to be stroked. This object must have at least two points defined in order for this view to draw the corresponding path.

Return Value
A new polyline overlay view.

Availability
Available in iOS 4.0 and later.

Declared In
`MKPolylineView.h`

MKReverseGeocoder Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKReverseGeocoder.h

Overview

The `MKReverseGeocoder` class provides services for converting a map coordinate (specified as a latitude/longitude pair) into information about that coordinate, such as the country, city, or street. A reverse geocoder object is a single-shot object that works with a network-based map service to look up placemark information for its specified coordinate value.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

The Google terms of service require that the reverse geocoding service be used in conjunction with a Google map; take this into account when designing your application's user interface.

Each Map Kit application has a limited amount of reverse geocoding capacity, so it is to your advantage to use reverse geocode requests sparingly. Here are some rules of thumb for using this class most effectively:

- Send at most one reverse-geocoding request for any one user action.
- If the user performs multiple actions that involve reverse-geocoding the same location, reuse the results from the initial reverse-geocoding request instead of starting individual requests for each action.
- When you want to update the location automatically (such as when the user is moving), reissue the reverse-geocoding request only when the user's location has moved a significant distance and after a reasonable amount of time has passed. For example, in a typical situation, you should not send more than one reverse-geocode request per minute.
- Do not start a reverse-geocoding request at a time when the user will not see the results immediately. For example, do not start a request if your application recently resigned the active state (possibly because of an interruption such as a phone call) and is waiting to become active again.

An iOS-based device must have access to the network in order for the reverse geocoder object to return valid information. The reverse geocoder returns information through its associated delegate object, which is an object that conforms to the `MKReverseGeocoderDelegate` protocol. If the reverse geocoder is unable to retrieve the requested information, it similarly reports the error to its delegate object. For more information on this protocol, see *MKReverseGeocoderDelegate Protocol Reference*.

Tasks

Initializing the Reverse Geocoder

- `initWithCoordinate:` (page 102)
  Initializes the reverse geocoder with the specified coordinate value.

Accessing Reverse Geocoder Attributes

`delegate` (page 101) *property*
The reverse geocoder's delegate object.

`coordinate` (page 101) *property*
The coordinate whose placemark data you want to retrieve. (read-only)

`placemark` (page 101) *property*
The result of the reverse-geocoding operation. (read-only)

Managing the Search

- `start` (page 103)
  Starts the reverse-geocoding process asynchronously.

`querying` (page 101) *property*
A Boolean value indicating whether the receiver is in the middle of reverse-geocoding its coordinate (read-only)

- `cancel` (page 102)
  Cancels a pending reverse-geocoding request.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

coordinate

The coordinate whose placemark data you want to retrieve. (read-only)

```
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate
```

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

delegate

The reverse geocoder's delegate object.

```
@property (nonatomic, assign) id<MKReverseGeocoderDelegate> delegate
```

Discussion
A reverse-geocoder object sends messages to its delegate regarding the successful (or unsuccessful) acquisition of placemark data. You must provide a delegate object to receive this data.

For more information about the `MKReverseGeocoderDelegate` protocol, see *MKReverseGeocoderDelegate Protocol Reference*.

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

placemark

The result of the reverse-geocoding operation. (read-only)

```
@property (nonatomic, readonly) MKPlacemark *placemark
```

Discussion
The value of this property is `nil` by default. After a successful reverse-geocoding operation, it is set to the placemark object that was generated.

Availability
Available in iOS 3.2 and later.

Declared In
`MKReverseGeocoder.h`

querying

A Boolean value indicating whether the receiver is in the middle of reverse-geocoding its coordinate (read-only)

```
@property (nonatomic, readonly, getter=isQuerying) BOOL querying
```

Discussion
This property contains YES if the process is ongoing or NO if the process is done or has not yet been initiated.

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

Instance Methods

cancel

Cancels a pending reverse-geocoding request.

```
- (void)cancel
```

Discussion
You can use this method to cancel a pending request and free up the resources associated with that request. If the request has already returned or has not yet begun, calling this method has no effect.

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

initWithCoordinate:

Initializes the reverse geocoder with the specified coordinate value.

```
- (id)initWithCoordinate:(CLLocationCoordinate2D)coordinate
```

Parameters
coordinate
The map coordinate whose placemark information you want to retrieve.

Return Value
An initialized `MKReverseGeocoder` object.

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

start

Starts the reverse-geocoding process asynchronously.

```
- (void)start
```

Discussion
You should call this method only once to begin the reverse-geocoding process. This method submits the coordinate value to the map server asynchronously and returns. Once the process is complete, the results are delivered to the associated delegate object.

Availability
Available in iOS 3.0 and later.

Declared In
`MKReverseGeocoder.h`

MKShape Class Reference

Inherits from	NSObject
Conforms to	MKAnnotation NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKShape.h

Overview

The `MKShape` class is an abstract class that defines the basic properties for all shape-based annotation objects. This class must be subclassed and cannot be used as is. Subclasses are responsible for defining the geometry of the shape and providing an appropriate value for the `coordinate` (page 114) property inherited from the `MKAnnotation` protocol.

Tasks

Accessing the Shape Attributes

`title` (page 106) *property*
The title of the shape annotation.

`subtitle` (page 106) *property*
The subtitle of the shape annotation.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

subtitle

The subtitle of the shape annotation.

```
@property (copy) NSString *subtitle
```

Discussion
This string is displayed in the callout for the associated annotation view. The default value of this property is `nil`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKShape.h`

title

The title of the shape annotation.

```
@property (copy) NSString *title
```

Discussion
This string is displayed in the callout for the associated annotation view. The default value of this property is `nil`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKShape.h`

MKUserLocation Class Reference

Inherits from	NSObject
Conforms to	MKAnnotation NSObject (NSObject)
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKUserLocation.h
Related sample code	MapCallouts

Overview

The `MKUserLocation` class defines a specific type of annotation that identifies the user's current location. You do not create instances of this class directly. Instead, you retrieve an existing `MKUserLocation` object from the `userLocation` (page 40) property of the map view displayed in your application.

Important: The MapKit framework uses Google services to provide map data. Use of this class and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Tasks

Determining the User's Position

`location` (page 108) *property*
The current location of the device. (read-only)

`updating` (page 109) *property*
A Boolean value indicating whether the user's location is currently being updated. (read-only)

Accessing the User Annotation Text

`title` (page 108) *property*
The title to display for the user location annotation.

`subtitle` (page 108) *property*
The subtitle to display for the user location annotation.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

location

The current location of the device. (read-only)

```
@property (readonly, nonatomic) CLLocation *location
```

Discussion
This property contains `nil` if the map view is not currently showing the user location or if the user's location has not yet been determined.

Availability
Available in iOS 3.0 and later.

Declared In
`MKUserLocation.h`

subtitle

The subtitle to display for the user location annotation.

```
@property (retain, nonatomic) NSString *subtitle
```

Availability
Available in iOS 3.0 and later.

Declared In
`MKUserLocation.h`

title

The title to display for the user location annotation.

```
@property (retain, nonatomic) NSString *title
```

Availability
Available in iOS 3.0 and later.

Declared In
`MKUserLocation.h`

updating

A Boolean value indicating whether the user's location is currently being updated. (read-only)

```
@property (readonly, nonatomic, getter=isUpdating) BOOL updating
```

Availability
Available in iOS 3.0 and later.

Declared In
`MKUserLocation.h`

Protocols

MKAnnotation Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKAnnotation.h
Related sample code	MapCallouts WeatherMap

Overview

The `MKAnnotation` protocol is used to provide annotation-related information to a map view. To use this protocol, you adopt it in any custom objects that store or represent annotation data. Each object then serves as the source of information about a single map annotation and provides critical information, such as the annotation's location on the map. Annotation objects do not provide the visual representation of the annotation but typically coordinate (in conjunction with the map view's delegate) the creation of an appropriate `MKAnnotationView` object to handle the display.

An object that adopts this protocol must implement the `coordinate` (page 114) property. The other methods of this protocol are optional.

Important: The MapKit framework uses Google services to provide map data. Use of this protocol and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Tasks

Position Attributes

`coordinate` (page 114) *property*
The center point (specified as a map coordinate) of the annotation. (required) (read-only)

`- setCoordinate:` (page 114)
Sets the new center point of the annotation.

Title Attributes

- `title` (page 115)
  Returns the string containing the annotation's title.
- `subtitle` (page 115)
  Returns the string containing the annotation's subtitle.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

coordinate

The center point (specified as a map coordinate) of the annotation. (required) (read-only)

```
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate
```

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
`MKAnnotation.h`

Instance Methods

setCoordinate:

Sets the new center point of the annotation.

```
- (void)setCoordinate:(CLLocationCoordinate2D)newCoordinate
```

Parameters
`newCoordinate`
  The new center point for the annotation.

Discussion
Annotations that support dragging should implement this method to update the position of the annotation.

Availability
Available in iOS 4.0 and later.

Declared In
MKAnnotation.h

subtitle

Returns the string containing the annotation's subtitle.

```
- (NSString *)subtitle
```

Return Value
The subtitle string.

Discussion
This string is displayed in the callout for the associated annotation view.

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
MKAnnotation.h

title

Returns the string containing the annotation's title.

```
- (NSString *)title
```

Return Value
The title string.

Discussion
Although the implementation method is optional, if you support the selection of annotations in your map view, you are expected to provide an implementation. This string is displayed in the callout for the associated annotation view.

Availability
Available in iOS 3.0 and later.

Related Sample Code
MapCallouts

Declared In
MKAnnotation.h

MKMapViewDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MapKit/MKMapView.h
Related sample code	MapCallouts WeatherMap WorldCities

Overview

The `MKMapViewDelegate` protocol defines a set of optional methods that you can use to receive map-related update messages. Because many map operations require the `MKMapView` class to load data asynchronously, the map view calls these methods to notify your application when specific operations complete. The map view also uses these methods to request annotation and overlay views and to manage interactions with those views.

Before releasing an `MKMapView` object, for which you have set a delegate, remember to set that object's `delegate` (page 37) property to `nil`. One place you can do this is in the `dealloc` method where you dispose of the map view.

Important: The MapKit framework uses Google services to provide map data. Use of this protocol and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Tasks

Responding to Map Position Changes

`- mapView:regionWillChangeAnimated:` (page 123)
Tells the delegate that the region displayed by the map view is about to change.

- `mapView:regionDidChangeAnimated:` (page 123)
  Tells the delegate that the region displayed by the map view just changed.

Loading the Map Data

- `mapViewWillStartLoadingMap:` (page 126)
  Tells the delegate that the specified map view is about to retrieve some map data.
- `mapViewDidFinishLoadingMap:` (page 125)
  Tells the delegate that the specified map view successfully loaded the needed map data.
- `mapViewDidFailLoadingMap:withError:` (page 125)
  Tells the delegate that the specified view was unable to load the map data.

Tracking the User Location

- `mapViewWillStartLocatingUser:` (page 126)
  Tells the delegate that the map view will start tracking the user's position.
- `mapViewDidStopLocatingUser:` (page 126)
  Tells the delegate that the map view stopped tracking the user's location.
- `mapView:didUpdateUserLocation:` (page 122)
  Tells the delegate that the location of the user was updated.
- `mapView:didFailToLocateUserWithError:` (page 121)
  Tells the delegate that an attempt to locate the user's position failed.

Managing Annotation Views

- `mapView:viewForAnnotation:` (page 124)
  Returns the view associated with the specified annotation object.
- `mapView:didAddAnnotationViews:` (page 120)
  Tells the delegate that one or more annotation views were added to the map.
- `mapView:annotationView:calloutAccessoryControlTapped:` (page 119)
  Tells the delegate that the user tapped one of the annotation view's accessory buttons.

Dragging an Annotation View

- `mapView:annotationView:didChangeDragState:fromOldState:` (page 120)
  Tells the delegate that the drag state of one of its annotation views changed.

Selecting Annotation Views

- `mapView:didSelectAnnotationView:` (page 122)
  Tells the delegate that one of its annotation views was selected.
- `mapView:didDeselectAnnotationView:` (page 121)
  Tells the delegate that one of its annotation views was deselected.

Managing Overlay Views

- `mapView:viewForOverlay:` (page 124)
  Asks the delegate for the overlay view to use when displaying the specified overlay object.
- `mapView:didAddOverlayViews:` (page 121)
  Tells the delegate that one or more overlay views were added to the map.

Instance Methods

mapView:annotationView:calloutAccessoryControlTapped:

Tells the delegate that the user tapped one of the annotation view's accessory buttons.

```
- (void)mapView:(MKMapView *)mapView   annotationView:(MKAnnotationView *)view
    calloutAccessoryControlTapped:(UIControl *)control
```

Parameters

mapView
The map view containing the specified annotation view.

view
The annotation view whose button was tapped.

control
The control that was tapped.

Discussion
Accessory views contain custom content and are positioned on either side of the annotation title text. If a view you specify is a descendant of the `UIControl` class, the map view calls this method as a convenience whenever the user taps your view. You can use this method to respond to taps and perform any actions associated with that control. For example, if your control displayed additional information about the annotation, you could use this method to present a modal panel with that information.

If your custom accessory views are not descendants of the `UIControl` class, the map view does not call this method.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapView:annotationView:didChangeDragState:fromOldState:

Tells the delegate that the drag state of one of its annotation views changed.

```
- (void)mapView:(MKMapView *)mapView annotationView:(MKAnnotationView
    *)annotationView didChangeDragState:(MKAnnotationViewDragState)newState
       fromOldState:(MKAnnotationViewDragState)oldState
```

Parameters

mapView
The map view containing the annotation view.

annotationView
The annotation view whose drag state changed.

newState
The new drag state of the annotation view.

oldState
The previous drag state of the annotation view.

Discussion
The drag state typically changes in response to user interactions with the annotation view. However, the annotation view itself is responsible for changing that state as well.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapView:didAddAnnotationViews:

Tells the delegate that one or more annotation views were added to the map.

```
- (void)mapView:(MKMapView *)mapView   didAddAnnotationViews:(NSArray *)views
```

Parameters

mapView
The map view that added the annotation views.

views
An array of `MKAnnotationView` objects representing the views that were added.

Discussion
By the time this method is called, the specified views are already added to the map.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapView:didAddOverlayViews:

Tells the delegate that one or more overlay views were added to the map.

```
- (void)mapView:(MKMapView *)mapView didAddOverlayViews:(NSArray *)overlayViews
```

Parameters

mapView

The map view that added the overlay views.

overlayViews

An array of `MKOverlayView` objects representing the views that were added.

Discussion

By the time this method is called, the specified views are already added to the map.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

mapView:didDeselectAnnotationView:

Tells the delegate that one of its annotation views was deselected.

```
- (void)mapView:(MKMapView *)mapView didDeselectAnnotationView:(MKAnnotationView
    *)view
```

Parameters

mapView

The map view containing the annotation view.

view

The annotation view that was deselected.

Discussion

You can use this method to track changes in the selection state of annotation views.

Availability

Available in iOS 4.0 and later.

Declared In

`MKMapView.h`

mapView:didFailToLocateUserWithError:

Tells the delegate that an attempt to locate the user's position failed.

```
- (void)mapView:(MKMapView *)mapView didFailToLocateUserWithError:(NSError *)error
```

Parameters

mapView

The map view that is tracking the user's location.

error
An error object containing the reason why location tracking failed.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapView:didSelectAnnotationView:

Tells the delegate that one of its annotation views was selected.

```
- (void)mapView:(MKMapView *)mapView didSelectAnnotationView:(MKAnnotationView
    *)view
```

Parameters
mapView
The map view containing the annotation view.

view
The annotation view that was selected.

Discussion
You can use this method to track changes in the selection state of annotation views.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapView:didUpdateUserLocation:

Tells the delegate that the location of the user was updated.

```
- (void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation
    *)userLocation
```

Parameters
mapView
The map view that is tracking the user's location.

userLocation
The location object representing the user's latest location.

Discussion
While the `showsUserLocation` (page 40) property is set to `YES`, this method is called whenever a new location update is received by the map view.

This method is not called if the application is currently running in the background. If you want to receive location updates while running in the background, you must use the Core Location framework.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapView:regionDidChangeAnimated:

Tells the delegate that the region displayed by the map view just changed.

```
- (void)mapView:(MKMapView *)mapView   regionDidChangeAnimated:(BOOL)animated
```

Parameters

mapView
The map view whose visible region changed.

animated
If YES, the change to the new region was animated.

Discussion
This method is called whenever the currently displayed map region changes. During scrolling, this method may be called many times to report updates to the map position. Therefore, your implementation of this method should be as lightweight as possible to avoid affecting scrolling performance.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapView:regionWillChangeAnimated:

Tells the delegate that the region displayed by the map view is about to change.

```
- (void)mapView:(MKMapView *)mapView   regionWillChangeAnimated:(BOOL)animated
```

Parameters

mapView
The map view whose visible region is about to change.

animated
If YES, the change to the new region will be animated. If NO, the change will be made immediately.

Discussion
This method is called whenever the currently displayed map region changes. During scrolling, this method may be called many times to report updates to the map position. Therefore, your implementation of this method should be as lightweight as possible to avoid affecting scrolling performance.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapView:viewForAnnotation:

Returns the view associated with the specified annotation object.

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView   viewForAnnotation:(id
    <MKAnnotation>)annotation
```

Parameters

mapView

The map view that requested the annotation view.

annotation

The object representing the annotation that is about to be displayed. In addition to your custom annotations, this object could be an `MKUserLocation` object representing the user's current location.

Return Value

The annotation view to display for the specified annotation or `nil` if you want to display a standard annotation view.

Discussion

Rather than create a new view each time this method is called, you should use the `dequeueReusableAnnotationViewWithIdentifier:` (page 46) method of the `MKMapView` class to see if an existing annotation view of the desired type already exists. If one does exist, you should update the view to reflect the attributes of the specified annotation and return it. If a view of the appropriate type does not exist, you should create one, configure it with the needed annotation data, and return it.

If the object in the annotation *parameter* is an instance of the `MKUserLocation` class, you can provide a custom view to denote the user's location. To display the user's location using the default system view, return `nil`.

If you do not implement this method, or if you return `nil` from your implementation for annotations other than the user location annotation, the map view uses a standard pin annotation view.

Availability

Available in iOS 3.0 and later.

Declared In

`MKMapView.h`

mapView:viewForOverlay:

Asks the delegate for the overlay view to use when displaying the specified overlay object.

```
- (MKOverlayView *)mapView:(MKMapView *)mapView viewForOverlay:(id
    <MKOverlay>)overlay
```

Parameters

mapView

The map view that requested the overlay view.

overlay

The object representing the overlay that is about to be displayed.

Return Value

The view to use when presenting the specified overlay on the map. If you return `nil`, no view is displayed for the specified overlay object.

Discussion
If you support the presentation of overlays, you must implement this method and provide the views for your overlay objects.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapViewDidFailLoadingMap:withError:

Tells the delegate that the specified view was unable to load the map data.

```
- (void)mapViewDidFailLoadingMap:(MKMapView *)mapView  withError:(NSError *)error
```

Parameters

mapView
The map view that started the load operation.

error
The reason that the map data could not be loaded.

Discussion
This method might be called in situations where the device does not have access to the network or is unable to load the map data for some reason. It may also be called if a request for additional map tiles comes in while a previous request for tiles is still pending. You can use this message to notify the user that the map data is unavailable.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapViewDidFinishLoadingMap:

Tells the delegate that the specified map view successfully loaded the needed map data.

```
- (void)mapViewDidFinishLoadingMap:(MKMapView *)mapView
```

Parameters

mapView
The map view that started the load operation.

Discussion
This method is called when the map tiles associated with the current request have been loaded. Map tiles are requested when a new visible area is scrolled into view and tiles are not already available. Map tiles may also be requested for portions of the map that are not currently visible. For example, the map view may load tiles immediately surrounding the currently visible area as needed to handle small pans by the user.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapViewDidStopLocatingUser:

Tells the delegate that the map view stopped tracking the user's location.

```
- (void)mapViewDidStopLocatingUser:(MKMapView *)mapView
```

Parameters

mapView
The map view that stopped tracking the user's location.

Discussion
This method is called when the value of the `showsUserLocation` (page 40) property changes to `NO`.

Availability
Available in iOS 4.0 and later.

Declared In
MKMapView.h

mapViewWillStartLoadingMap:

Tells the delegate that the specified map view is about to retrieve some map data.

```
- (void)mapViewWillStartLoadingMap:(MKMapView *)mapView
```

Parameters

mapView
The map view that began loading the data.

Discussion
This method is called whenever a new group of map tiles need to be downloaded from the server. This typically occurs whenever you expose portions of the map by panning or zooming the content. You can use this method to mark the time that it takes for the map view to load the data.

Availability
Available in iOS 3.0 and later.

Declared In
MKMapView.h

mapViewWillStartLocatingUser:

Tells the delegate that the map view will start tracking the user's position.

```
- (void)mapViewWillStartLocatingUser:(MKMapView *)mapView
```

Parameters

mapView
The map view that is tracking the user's location.

Discussion
This method is called when the value of the `showsUserLocation` (page 40) property changes to `YES`.

Availability
Available in iOS 4.0 and later.

Declared In
`MKMapView.h`

MKOverlay Protocol Reference

Conforms to	MKAnnotation
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 4.0 and later.
Declared in	MKOverlay.h

Overview

The `MKOverlay` protocol defines a specific type of annotation that represents both a point and an area on a map. Overlay objects are essentially data objects that contain the geographic data needed to represent the map area. For example, overlays can take the form of common shapes such as rectangles and circles. They can also describe polygons and other complex shapes.

You use overlays to layer more sophisticated content on top of a map view. For example, you could use an overlay to show the boundaries of a national park or trace a bus route along city streets. The Map Kit framework defines several concrete classes that conform to this protocol and define standard shapes.

Because overlays are also annotations, they have similar usage pattern to annotations. When added to a map view using the `addOverlay:` (page 42) method, that view detects whenever the overlay's defined region intersects the visible portion of the map. At that point, the map view asks its delegate to provide a special overlay view to draw the visual representation of the overlay. If you add an overlay to a map view as an annotation instead, it is treated as an annotation with a single point.

Tasks

Describing the Overlay Geometry

`coordinate` (page 130) *property*
The approximate center point of the overlay area. (required) (read-only)

`boundingMapRect` (page 130) *property*
The projected rectangle that encompasses the overlay. (required) (read-only)

Determining Map Intersections

`- intersectsMapRect:` (page 131)
Returns a Boolean indicating whether the specified rectangle intersects the receiver's shape.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

boundingMapRect

The projected rectangle that encompasses the overlay. (required) (read-only)

```
@property (nonatomic, readonly) MKMapRect boundingMapRect
```

Discussion
This property contains the smallest rectangle that completely encompasses the overlay area. Implementers of this protocol must set this area when implementing their overlay class. The rectangle should be specified using projected coordinates—that is, coordinates obtained by projecting the globe onto a two-dimensional surface.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlay.h`

coordinate

The approximate center point of the overlay area. (required) (read-only)

```
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate
```

Discussion
This point is typically set to the center point of the map's bounding rectangle. It is used as the anchor point for any callouts displayed for the annotation.

Availability
Available in iOS 4.0 and later.

Declared In
`MKOverlay.h`

Instance Methods

intersectsMapRect:

Returns a Boolean indicating whether the specified rectangle intersects the receiver's shape.

```
- (BOOL)intersectsMapRect:(MKMapRect)mapRect
```

Parameters

mapRect

The rectangle to intersect with the receiver's area.

Return Value

`YES` if any part of the map rectangle intersects the receiver's shape or `NO` if it does not.

Discussion

You can implement this method to provide more specific bounds checking for an overlay. If you do not implement it, the bounding rectangle is used to detect intersections.

Availability

Available in iOS 4.0 and later.

Declared In

`MKOverlay.h`

MKReverseGeocoderDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MapKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	MKReverseGeocoder.h

Overview

The `MKReverseGeocoderDelegate` protocol defines the interface for receiving messages from an `MKReverseGeocoder` object. You use this protocol to receive the placemark information for a given coordinate or to retrieve any errors that occurred during the reverse-geocoding process.

Delegates must implement both methods of this protocol.

Important: The MapKit framework uses Google services to provide map data. Use of this protocol and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

The Google terms of service require that the reverse geocoding service be used in conjunction with a Google map; take this into account when designing your application's user interface.

Each Map Kit application has a limited amount of reverse geocoding capacity, so it is to your advantage to use reverse geocode requests sparingly. For more information about when to initiate reverse-geocoding requests, see *MKReverseGeocoder Class Reference*.

Tasks

Processing Placemark Searches

`- reverseGeocoder:didFindPlacemark:` (page 134)
Tells the delegate that a reverse geocoder successfully obtained placemark information for its coordinate. (required)

Instance Methods

reverseGeocoder:didFailWithError:

Tells the delegate that the specified reverse geocoder failed to obtain information about its coordinate. (required)

```
- (void)reverseGeocoder:(MKReverseGeocoder *)geocoder didFailWithError:(NSError *)error
```

Parameters

geocoder
The reverse geocoder object that was unable to complete its request.

error
An error object indicating the reason the request did not succeed.

Availability
Available in iOS 3.0 and later.

Declared In
MKReverseGeocoder.h

reverseGeocoder:didFindPlacemark:

Tells the delegate that a reverse geocoder successfully obtained placemark information for its coordinate. (required)

```
- (void)reverseGeocoder:(MKReverseGeocoder *)geocoder didFindPlacemark:(MKPlacemark *)placemark
```

Parameters

geocoder
The reverse geocoder object that completed its request successfully.

placemark
The object containing the placemark data.

Discussion
You can get the map coordinate for the associated request from either the reverse geocoder object or from the placemark object, which itself supports the `MKAnnotation` protocol.

Availability
Available in iOS 3.0 and later.

Declared In
MKReverseGeocoder.h

Functions

Map Kit Functions Reference

Framework: MapKit/MapKit.h
Declared in MKTypes.h

Overview

This document describes the functions found in the Map Kit framework.

Functions by Task

The functions of the MapKit framework provide convenient ways to package map-related data structures.

Important: The MapKit framework uses Google services to provide map data. Use of these functions and the associated interfaces binds you to the Google Maps/Google Earth API terms of service. You can find these terms of service at http://code.google.com/apis/maps/iphone/terms.html.

Making Coordinate Structures

`MKCoordinateSpanMake` (page 142)
Creates a new `MKCoordinateSpan` (page 161) from the specified values.

`MKCoordinateRegionMake` (page 141)
Creates a new `MKCoordinateRegion` (page 162) from the specified coordinate and span values.

`MKCoordinateRegionMakeWithDistance` (page 141)
Creates a new `MKCoordinateRegion` (page 162) from the specified coordinate and distance values.

Making Map Point Structures

`MKMapPointMake` (page 143)
Creates a new `MKMapPoint` (page 162) structure from the specified values.

`MKMapSizeMake` (page 155)
Creates a new `MKMapSize` (page 163) structure from the specified values.

`MKMapRectMake` (page 152)
Creates a new `MKMapRect` (page 163) structure from the specified values.

Converting Between Data Types

`MKMapPointForCoordinate` (page 143)
Returns the map point data structure that corresponds to the specified coordinate.

`MKCoordinateForMapPoint` (page 140)
Returns the latitude and longitude that corresponds to the specified map point.

`MKCoordinateRegionForMapRect` (page 140)
Returns the region that corresponds to the specified map rectangle.

Getting Map Units

`MKMetersPerMapPointAtLatitude` (page 156)
Returns the distance spanned by one map point at the specified latitude.

`MKMapPointsPerMeterAtLatitude` (page 144)
Returns the number of map points that represent one meter at the given latitude.

`MKMetersBetweenMapPoints` (page 155)
Returns the number of meters between two map points.

`MKRoadWidthAtZoomScale` (page 156)
Returns the width (in screen points) of roads on a map at the specified zoom level.

Getting Points Along a Map Rectangle

`MKMapRectGetMinX` (page 148)
Returns the minimum x-axis value of the specified rectangle.

`MKMapRectGetMinY` (page 149)
Returns the minimum y-axis value of the specified rectangle.

`MKMapRectGetMidX` (page 148)
Returns the mid-point along the x-axis of the specified rectangle.

`MKMapRectGetMidY` (page 148)
Returns the mid-point along the y-axis of the specified rectangle.

`MKMapRectGetMaxX` (page 147)
Returns the maximum x-axis value of the specified rectangle.

`MKMapRectGetMaxY` (page 147)
Returns the maximum y-axis value of the specified rectangle.

`MKMapRectGetWidth` (page 149)
Returns the width of the map rectangle.

`MKMapRectGetHeight` (page 147)
Returns the height of the map rectangle.

Comparing Map Values

`MKMapPointEqualToPoint` (page 143)
Returns a Boolean value indicating whether the two map points are equal.

`MKMapSizeEqualToSize` (page 154)
Returns a Boolean value indicating whether the two map sizes are equal.

`MKMapRectEqualToRect` (page 146)
Returns a Boolean value indicating whether the two map rectangles are equal

`MKMapRectContainsPoint` (page 144)
Returns a Boolean value indicating whether the specified map point lies within the rectangle.

`MKMapRectContainsRect` (page 145)
Returns Boolean value indicating whether one rectangle contains another.

`MKMapRectIntersectsRect` (page 151)
Returns a Boolean value indicating whether two rectangles intersect each other.

`MKMapRectIsNull` (page 152)
Returns a Boolean indicating whether the specified rectangle is null.

`MKMapRectIsEmpty` (page 151)
Returns a Boolean value indicating whether the specified rectangle has no area.

Modifying Map Rectangles

`MKMapRectUnion` (page 154)
Returns a rectangle representing the union of the two rectangles.

`MKMapRectIntersection` (page 150)
Returns the rectangle representing the intersection of two rectangles.

`MKMapRectInset` (page 150)
Returns the specified rectangle inset by the specified amounts.

`MKMapRectOffset` (page 153)
Returns a rectangle whose origin point is shifted by the specified amount.

`MKMapRectDivide` (page 145)
Divides the specified rectangle into two smaller rectangles.

Getting Strings for Map Values

`MKStringFromMapPoint` (page 157)
Returns a formatted string for the specified map point.

`MKStringFromMapSize` (page 158)
Returns a formatted string for the specified map size.

`MKStringFromMapRect` (page 157)
Returns a formatted string for the specified map rectangle.

Determining Map Boundaries

`MKMapRectSpans180thMeridian` (page 153)
Returns a Boolean value that indicates whether the specified map rectangle crosses the 180th meridian.

`MKMapRectRemainder` (page 153)
Normalizes the portion of the specified rectangle that lies outside the world map boundaries.

Functions

MKCoordinateForMapPoint

Returns the latitude and longitude that corresponds to the specified map point.

```
CLLocationCoordinate2D MKCoordinateForMapPoint(
    MKMapPoint mapPoint
);
```

Parameters

mapPoint
The map point value that corresponds to the desired point on a two-dimensional map projection.

Return Value
The coordinate structure containing the latitude and longitude values for the specified point.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKCoordinateRegionForMapRect

Returns the region that corresponds to the specified map rectangle.

```
MKCoordinateRegion MKCoordinateRegionForMapRect(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle that corresponds to the desired region on a two-dimensional map projection.

Return Value
The region structure specifying the latitude, longitude, and span values for the specified rectangle.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKCoordinateRegionMake

Creates a new MKCoordinateRegion (page 162) from the specified coordinate and span values.

```
UIKIT_STATIC_INLINE MKCoordinateRegion MKCoordinateRegionMake(
    CLLocationCoordinate2D centerCoordinate,
    MKCoordinateSpan span
);
```

Parameters

centerCoordinate

The center point of the region.

span

The horizontal and vertical span representing the amount of map to display. The size of the span also reflects the current zoom level.

Return Value
A region with the specified values.

Availability
Available in iOS 3.0 and later.

Declared In
MKGeometry.h

MKCoordinateRegionMakeWithDistance

Creates a new MKCoordinateRegion (page 162) from the specified coordinate and distance values.

```
MKCoordinateRegion MKCoordinateRegionMakeWithDistance(
    CLLocationCoordinate2D centerCoordinate,
    CLLocationDistance latitudinalMeters,
    CLLocationDistance longitudinalMeters
);
```

Parameters

centerCoordinate

The center point of the new coordinate region.

latitudinalMeters

The amount of north-to-south distance (measured in meters) to use for the span.

longitudinalMeters

The amount of east-to-west distance (measured in meters) to use for the span.

Return Value
A region with the specified values.

Availability
Available in iOS 3.0 and later.

Declared In
MKGeometry.h

MKCoordinateSpanMake

Creates a new MKCoordinateSpan (page 161) from the specified values.

```
UIKIT_STATIC_INLINE MKCoordinateSpan MKCoordinateSpanMake(
    CLLocationDegrees latitudeDelta,
    CLLocationDegrees longitudeDelta
);
```

Parameters

latitudeDelta

The amount of north-to-south distance (measured in degrees) to use for the span. Unlike longitudinal distances, which vary based on the latitude, one degree of latitude is approximately 111 kilometers (69 miles) at all times.

longitudeDelta

The amount of east-to-west distance (measured in degrees) to use for the span. The number of kilometers spanned by a longitude range varies based on the current latitude. For example, one degree of longitude spans a distance of approximately 111 kilometers (69 miles) at the equator but shrinks to 0 kilometers at the poles.

Return Value
A span with the specified delta values.

Availability
Available in iOS 3.0 and later.

Declared In
MKGeometry.h

MKMapPointEqualToPoint

Returns a Boolean value indicating whether the two map points are equal.

```
BOOL MKMapPointEqualToPoint(
    MKMapPoint point1,
    MKMapPoint point2
);
```

Parameters

point1
The first map point.

point2
The second point.

Return Value

`YES` if the `x` and `y` values in both points are exactly the same or `NO` if one or both values are different.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapPointForCoordinate

Returns the map point data structure that corresponds to the specified coordinate.

```
MKMapPoint MKMapPointForCoordinate(
    CLLocationCoordinate2D coordinate
);
```

Parameters

coordinate
The coordinate containing the latitude and longitude values for the desired point.

Return Value

The map point value that corresponds to the specified coordinate on a two-dimensional map projection.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapPointMake

Creates a new `MKMapPoint` (page 162) structure from the specified values.

```
MKMapPoint MKMapPointMake(
    double x,
    double y
);
```

Parameters

x

The point along the east-west axis of the map projection.

y

The point along the north-south axis of the map projection.

Return Value
A map point with the specified values.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapPointsPerMeterAtLatitude

Returns the number of map points that represent one meter at the given latitude.

```
double MKMapPointsPerMeterAtLatitude(
    CLLocationDegrees latitude
);
```

Parameters

latitude

The latitude for which to return the value.

Return Value
The number of map points that span one meter.

Discussion
The number of map points per meter increases as the latitude approaches the poles.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectContainsPoint

Returns a Boolean value indicating whether the specified map point lies within the rectangle.

```
BOOL MKMapRectContainsPoint(
    MKMapRect rect,
    MKMapPoint point
);
```

Parameters

rect

The map rectangle being tested.

point

The point to check.

Return Value

`YES` if the rectangle is not null or empty and the point is located inside the rectangle; otherwise, `NO`.

Discussion

A point is considered to be inside the rectangle if its coordinates lie inside the rectangle or on the minimum X or minimum Y edge.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectContainsRect

Returns Boolean value indicating whether one rectangle contains another.

```
BOOL MKMapRectContainsRect(
    MKMapRect rect1,
    MKMapRect rect2
);
```

Parameters

rect1

The containing rectangle.

rect2

The rectangle that might be contained in *rect1*.

Return Value

`YES` if *rect2* is null or lies entirely inside *rect1*; otherwise, returns `NO` if *rect1* is null or does not completely enclose *rect2*.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectDivide

Divides the specified rectangle into two smaller rectangles.

```
void MKMapRectDivide(
    MKMapRect rect,
    MKMapRect *slice,
    MKMapRect *remainder,
    double amount,
    CGRectEdge edge
);
```

Parameters

rect

The rectangle to divide.

slice

On input, a pointer to a map rectangle. On output, this parameter contains the portion of *rect* that was removed.

remainder

On input, a pointer to a map rectangle. On output, this parameter contains the remaining portion of *rect* that was not removed.

amount

The amount of *rect* to remove along the specified edge. If this value is negative, it is set to `0`.

edge

The edge from which to remove the specified amount.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectEqualToRect

Returns a Boolean value indicating whether the two map rectangles are equal

```
BOOL MKMapRectEqualToRect(
    MKMapRect rect1,
    MKMapRect rect2
);
```

Parameters

rect1

The first map rectangle.

rect2

The second map rectangle.

Return Value
`YES` if the rectangles are exactly the same or `NO` if the origin point or size values are different.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectGetHeight

Returns the height of the map rectangle.

```
double MKMapRectGetHeight(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The rectangle's height.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectGetMaxX

Returns the maximum x-axis value of the specified rectangle.

```
double MKMapRectGetMaxX(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The maximum x-axis value.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectGetMaxY

Returns the maximum y-axis value of the specified rectangle.

```
double MKMapRectGetMaxY(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The maximum y-axis value.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectGetMidX

Returns the mid-point along the x-axis of the specified rectangle.

```
double MKMapRectGetMidX(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The midpoint value along the x-axis.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectGetMidY

Returns the mid-point along the y-axis of the specified rectangle.

```
double MKMapRectGetMidY(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The midpoint value along the y-axis.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMapRectGetMinX

Returns the minimum x-axis value of the specified rectangle.

```
double MKMapRectGetMinX(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The minimum x-axis value.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectGetMinY

Returns the minimum y-axis value of the specified rectangle.

```
double MKMapRectGetMinY(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The minimum y-axis value.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectGetWidth

Returns the width of the map rectangle.

```
double MKMapRectGetWidth(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to test.

Return Value
The rectangle's width.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectInset

Returns the specified rectangle inset by the specified amounts.

```
MKMapRect MKMapRectInset(
    MKMapRect rect,
    double dx,
    double dy
);
```

Parameters

rect
The original rectangle.

dx
The amount (in map points) to subtract from both sides along the x-axis.

dy
The amount (in map points) to subtract from both sides along the y-axis.

Return Value
The inset rectangle. If the original rectangle was null, that rectangle is returned instead.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectIntersection

Returns the rectangle representing the intersection of two rectangles.

```
MKMapRect MKMapRectIntersection(
    MKMapRect rect1,
    MKMapRect rect2
);
```

Parameters

rect1
The first rectangle.

rect2
The second rectangle.

Return Value
The rectangle representing the intersection of the two rectangles or `MKMapRectNull` (page 167) if there is no intersection.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectIntersectsRect

Returns a Boolean value indicating whether two rectangles intersect each other.

```
BOOL MKMapRectIntersectsRect(
    MKMapRect rect1,
    MKMapRect rect2
);
```

Parameters

rect1
The first rectangle to test.

rect2
The second rectangle to test.

Return Value
YES if *rect1* and *rect2* intersect each other or NO if they do not intersect or either rectangle is null.

Discussion
The rectangles are not considered to be intersecting if the only intersection occurs along an edge. To be considered a true intersection, the rectangles must both enclose a single rectangular area whose width and height are both greater than 0.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectIsEmpty

Returns a Boolean value indicating whether the specified rectangle has no area.

```
BOOL MKMapRectIsEmpty(
    MKMapRect rect
);
```

Parameters

rect
The rectangle to test.

Return Value
YES if the rectangle is null or its width or height are equal to 0; otherwise, NO.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRectIsNull

Returns a Boolean indicating whether the specified rectangle is null.

```
BOOL MKMapRectIsNull(
    MKMapRect rect
);
```

Parameters

rect
The rectangle to test.

Return Value

`YES` if the rectangle is null or `NO` if it is not null.

Discussion

A rectangle is considered null if its origin point contains an invalid or infinite value.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectMake

Creates a new `MKMapRect` (page 163) structure from the specified values.

```
MKMapRect MKMapRectMake(
    double x,
    double y,
    double width,
    double height
);
```

Parameters

x
The point along the east-west axis of the map projection to use for the origin.

y
The point along the north-south axis of the map projection to use for the origin.

width
The width of the rectangle (measured using map points).

height
The height of the rectangle (measured using map points).

Return Value

A map rectangle with the specified values.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectOffset

Returns a rectangle whose origin point is shifted by the specified amount.

```
MKMapRect MKMapRectOffset(
    MKMapRect rect,
    double dx,
    double dy
);
```

Parameters

rect

The original rectangle.

dx

The amount (in map points) by which to shift the x-coordinate of the origin point.

dy

The amount (in map points) by which to shift the x-coordinate of the origin point.

Return Value

The offset rectangle. If the original rectangle was null, that rectangle is returned instead.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectRemainder

Normalizes the portion of the specified rectangle that lies outside the world map boundaries.

```
MKMapRect MKMapRectRemainder(
    MKMapRect rect
);
```

Parameters

rect

The rectangle to check.

Discussion

For a rectangle that lies on the 180th meridian, this function isolates the portion that lies outside the boundary, wraps it to the opposite side of the map, and returns that rectangle.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectSpans180thMeridian

Returns a Boolean value that indicates whether the specified map rectangle crosses the 180th meridian.

```
BOOL MKMapRectSpans180thMeridian(
    MKMapRect rect
);
```

Parameters

rect

The rectangle to test.

Return Value

`YES` if the rectangle spans the 180th meridian or `NO` if it is contained wholly within the world map.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapRectUnion

Returns a rectangle representing the union of the two rectangles.

```
MKMapRect MKMapRectUnion(
    MKMapRect rect1,
    MKMapRect rect2
);
```

Parameters

rect1

The first rectangle.

rect2

The second rectangle.

Return Value

A rectangle whose area encompasses the two rectangles and the space between them.

Discussion

If either rectangle is null, this method returns the other rectangle. The origin point of the returned rectangle is set to the smaller of the x and y values for the two rectangles. Similarly, the size and width of the rectangle are computed by taking the maximum x and y values and subtracting the x and y values for the new origin point.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapSizeEqualToSize

Returns a Boolean value indicating whether the two map sizes are equal.

```
BOOL MKMapSizeEqualToSize(
    MKMapSize size1,
    MKMapSize size2
);
```

Parameters

size1

The first map size.

size2

The second map size.

Return Value

`YES` if the `width` and `height` values in both sizes are exactly the same or `NO` if one or both values are different.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMapSizeMake

Creates a new `MKMapSize` (page 163) structure from the specified values.

```
MKMapSize MKMapSizeMake(
    double width,
    double height
);
```

Parameters

width

The distance (measured using map points) along the east-west axis of the map projection.

height

The distance (measured using map points) along the north-south axis of the map projection.

Return Value

A map size with the specified values.

Availability

Available in iOS 4.0 and later.

Declared In

`MKGeometry.h`

MKMetersBetweenMapPoints

Returns the number of meters between two map points.

```
CLLocationDistance MKMetersBetweenMapPoints(
    MKMapPoint a,
    MKMapPoint b
);
```

Parameters

a

The first map point.

b

The second map point.

Return Value
The number of meters between the specified map points.

Discussion
This distance reflects the actual distance between the two points on the surface of the globe, taking into account the curvature of the Earth.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKMetersPerMapPointAtLatitude

Returns the distance spanned by one map point at the specified latitude.

```
CLLocationDistance MKMetersPerMapPointAtLatitude(
    CLLocationDegrees latitude
);
```

Parameters

latitude

The latitude for which to return the value.

Return Value
The distance (in meters) spanned by a single map point.

Discussion
The distance between map points decreases as the latitude approaches the poles. This relationship parallels the relationship between longitudinal coordinates at different latitudes.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKRoadWidthAtZoomScale

Returns the width (in screen points) of roads on a map at the specified zoom level.

```
CGFloat MKRoadWidthAtZoomScale(
    MKZoomScale zoomScale
);
```

Parameters

zoomScale

The scale factor currently applied to the map view.

Return Value
The width of roads, measured in screen points. You can use the returned value to set the width of lines in drawing code that traces the path of a road.

Availability
Available in iOS 4.0 and later.

Declared In
MKOverlayView.h

MKStringFromMapPoint

Returns a formatted string for the specified map point.

```
NSString *MKStringFromMapPoint(
    MKMapPoint point
);
```

Parameters

point

The map point to format.

Return Value
A formatted string containing the x and y coordinates of the map point.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKStringFromMapRect

Returns a formatted string for the specified map rectangle.

```
NSString *MKStringFromMapRect(
    MKMapRect rect
);
```

Parameters

rect

The map rectangle to format.

Return Value
A formatted string containing the origin and size values of the map rectangle.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKStringFromMapSize

Returns a formatted string for the specified map size.

```
NSString *MKStringFromMapSize(
    MKMapSize size
);
```

Parameters

size

The map size to format.

Return Value
A formatted string containing the width and height values of the map size.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

Data Types

PART IV

Data Types

Map Kit Data Types Reference

Framework: MapKit/MapKit.h

Overview

This document describes the data types found in the Map Kit framework.

Data Types

MKCoordinateSpan

A structure that defines the area spanned by a map region.

```
typedef struct {
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
} MKCoordinateSpan;
```

Fields

`latitudeDelta`

The amount of north-to-south distance (measured in degrees) to display on the map. Unlike longitudinal distances, which vary based on the latitude, one degree of latitude is always approximately 111 kilometers (69 miles).

`longitudeDelta`

The amount of east-to-west distance (measured in degrees) to display for the map region. The number of kilometers spanned by a longitude range varies based on the current latitude. For example, one degree of longitude spans a distance of approximately 111 kilometers (69 miles) at the equator but shrinks to 0 kilometers at the poles.

Discussion

You use the delta values in this structure to indicate the desired zoom level of the map, with smaller delta values corresponding to a higher zoom level.

Availability

Available in iOS 3.0 and later.

Declared In

`MKGeometry.h`

MKCoordinateRegion

A structure that defines which portion of the map to display.

```
typedef struct {
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;
```

Fields

`center`

The center point of the region.

`span`

The horizontal and vertical span representing the amount of map to display. The span also defines the current zoom level used by the map view object.

Availability
Available in iOS 3.0 and later.

Declared In
`MKGeometry.h`

MKMapPoint

A point on a two-dimensional map projection.

```
typedef struct {
    double x;
    double y;
} MKMapPoint;
```

Fields

`x`

The location of the point along the x-axis of the map.

`y`

The location of the point along the y-axis of the map.

Discussion
If you project the curved surface of the globe onto a flat surface, what you get is a two-dimensional version of a map where longitude lines appear to be parallel. Such maps are often used to show the entire surface of the globe all at once. An `MKMapPoint` data structure represents a point on this two-dimensional map.

The actual units of a map point are tied to the underlying units used to draw the contents of an `MKMapView`, but you should never need to worry about these units directly. You use map points primarily to simplify computations that would be complex to do using coordinate values on a curved surface. By converting to map points, you can perform those calculations on a flat surface, which is generally much simpler, and then convert back as needed. You can map between coordinate values and map points using the `MKMapPointForCoordinate` (page 143) and `MKCoordinateForMapPoint` (page 140) functions.

When saving map-related data to a file, you should always save coordinate values (latitude and longitude) and not map points.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapSize

Size information as measured on a two-dimensional map projection.

```
typedef struct {
    double width;
    double height;
} MKMapSize;
```

Fields
`width`
The width value. The units of this value are map points.
`height`
The height value. The units of this value are map points.

Discussion
If you project the curved surface of the globe onto a flat surface, what you get is a two-dimensional version of a map where longitude lines appear to be parallel. Such maps are often used to show the entire surface of the globe all at once. An `MKMapSize` data structure represents a horizontal and vertical distance as measured on this two-dimensional map.

Availability
Available in iOS 4.0 and later.

Declared In
MKGeometry.h

MKMapRect

A rectangular area as measured on a two-dimensional map projection.

```
typedef struct {
    MKMapPoint origin;
    MKMapSize size;
} MKMapRect;
```

Fields
`origin`
The origin point of the rectangle.
`size`
The width and height of the rectangle, starting from the origin point.

Discussion
If you project the curved surface of the globe onto a flat surface, what you get is a two-dimensional version of a map where longitude lines appear to be parallel. Such maps are often used to show the entire surface of the globe all at once. An `MKMapRect` data structure represents a rectangular area as seen on this two-dimensional map.

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

MKZoomScale

A scale factor being used in conjunction with a map.

```
typedef CGFloat MKZoomScale;
```

Availability
Available in iOS 4.0 and later.

Declared In
`MKGeometry.h`

Constants

PART V

Constants

Map Kit Constants Reference

Framework: MapKit/MapKit.h

Overview

This document describes the constants found in the Map Kit framework

Constants

Null Map Rectangle

The null map rectangle

```
const MKMapRect MKMapRectNull;
```

Constants

`MKMapRectNull`

You can use this constant when you want to specify an invalid map rectangle.

Available in iOS 4.0 and later.

Declared in `MKGeometry.h`.

World Map Constants

Map constants for the two-dimensional map projection.

```
const MKMapSize MKMapSizeWorld;
const MKMapRect MKMapRectWorld;
```

Constants

`MKMapSizeWorld`

Specifies the width and height (in map points) of the world in the two-dimensional map projection.

Available in iOS 4.0 and later.

Declared in `MKGeometry.h`.

`MKMapRectWorld`

The map rectangle that represents the world in the two-dimensional map projection.

Available in iOS 4.0 and later.

Declared in `MKGeometry.h`.

Error Domain

The error domain for Map Kit.

```
NSString *MKErrorDomain
```

Constants

`MKErrorDomain`

The Map Kit error domain.

Available in iOS 3.0 and later.

Declared in `MKTypes.h`.

MKErrorCode

Error constants for the Map Kit framework.

```
enum MKErrorCode {
    MKErrorUnknown = 1,
    MKErrorServerFailure,
    MKErrorLoadingThrottled,
    MKErrorPlacemarkNotFound,
};
```

Constants

`MKErrorUnknown`

An unknown error occurred.

Available in iOS 3.0 and later.

Declared in `MKTypes.h`.

`MKErrorServerFailure`

The map server was unable to return the desired information.

Available in iOS 3.0 and later.

Declared in `MKTypes.h`.

`MKErrorLoadingThrottled`

The data was not loaded because data throttling is in effect.

Available in iOS 3.0 and later.

Declared in `MKTypes.h`.

`MKErrorPlacemarkNotFound`

The specified placemark could not be found.

Available in iOS 3.0 and later.

Declared in `MKTypes.h`.

Document Revision History

This table describes the changes to *Map Kit Framework Reference*.

Date	Notes
2010-05-11	Added new classes and protocols introduced in iOS 4.0.
2009-05-12	New document that describes the classes, methods, and functions of the MapKit framework.

Document Revision History