MKMapView Class Reference

User Experience

2010-05-11

Contents

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	MKMapView.h MKTypes.h

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 12) 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 12) and `zoomEnabled` (page 14) 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`, `MKMapSize`, and `MKMapRect` 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 6). 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 11) *property*
  The type of data displayed by the map view.

`zoomEnabled` (page 14) *property*
  A Boolean value that determines whether the user may use pinch gestures to zoom in and out of the map.

`scrollEnabled` (page 12) *property*
  A Boolean value that determines whether the user may scroll around the map.

Accessing the Delegate

`delegate` (page 11) *property*
  The receiver's delegate.

Manipulating the Visible Portion of the Map

`region` (page 12) *property*
  The area currently displayed by the map view.

– `setRegion:animated:` (page 25)
  Changes the currently visible region and optionally animates the change.

`centerCoordinate` (page 10) *property*
  The map coordinate at the center of the map view.

– `setCenterCoordinate:animated:` (page 25)
  Changes the center coordinate of the map and optionally animates the change.

`visibleMapRect` (page 14) *property*
  The area currently displayed by the map view.

– `setVisibleMapRect:animated:` (page 26)
  Changes the currently visible portion of the map and optionally animates the change.

– `setVisibleMapRect:edgePadding:animated:` (page 26)
  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 13) *property*
A Boolean value indicating whether the map may display the user location.

`userLocationVisible` (page 14) *property*
A Boolean value indicating whether the device's current location is visible in the map view. (read-only)

`userLocation` (page 13) *property*
The annotation object representing the user's current location. (read-only)

Annotating the Map

`annotations` (page 10) *property*
The complete list of annotations associated with the receiver. (read-only)

`– addAnnotation:` (page 15)
Adds the specified annotation to the map view.

`– addAnnotations:` (page 15)
Adds an array of annotations to the map view.

`– removeAnnotation:` (page 22)
Removes the specified annotation object from the map view.

`– removeAnnotations:` (page 23)
Removes the specified annotation objects from the map view.

`– viewForAnnotation:` (page 27)
Returns the annotation view associated with the specified annotation object, if any.

`annotationVisibleRect` (page 10) *property*
The visible rectangle of the map view. (read-only)

`– dequeueReusableAnnotationViewWithIdentifier:` (page 18)
Returns a reusable annotation view located by its identifier.

Managing Annotation Selections

`selectedAnnotations` (page 13) *property*
The annotations that are currently selected.

`– selectAnnotation:animated:` (page 24)
Selects the specified annotation and displays a callout view for it.

`– deselectAnnotation:animated:` (page 19)
Deselects the specified annotation and hides its callout view.

Adding and Removing Overlays

`overlays` (page 11) *property*
The overlays currently associated with the map view. (read-only)

`– addOverlay:` (page 16)
Adds a single overlay object to the map.

- `addOverlays:` (page 16)
  Adds an array of overlay objects to the map.
- `removeOverlay:` (page 23)
  Removes a single overlays from the map.
- `removeOverlays:` (page 24)
  Removes one or more overlays from the map.
- `insertOverlay:atIndex:` (page 20)
  Inserts an overlay into the list of overlay objects associated with the map.
- `exchangeOverlayAtIndex:withOverlayAtIndex:` (page 19)
  Exchanges the position of two overlay objects.
- `insertOverlay:aboveOverlay:` (page 20)
  Inserts one overlay on top of another.
- `insertOverlay:belowOverlay:` (page 21)
  Para
- `viewForOverlay:` (page 27)
  Returns the view (if any) associated with the overlay object.

Converting Map Coordinates

- `convertCoordinate:toPointToView:` (page 17)
  Converts a map coordinate to a point in the specified view.
- `convertPoint:toCoordinateFromView:` (page 17)
  Converts a point in the specified view's coordinate system to a map coordinate.
- `convertRegion:toRectToView:` (page 18)
  Converts a map region to a rectangle in the specified view.
- `convertRect:toRegionFromView:` (page 18)
  Converts a rectangle in the specified view's coordinate system to a map region.

Adjusting Map Regions and Rectangles

- `regionThatFits:` (page 22)
  Adjusts the aspect ratio of the specified region to ensure that it fits in the map view's frame.
- `mapRectThatFits:` (page 21)
  Adjusts the aspect ratio of the specified map rectangle to ensure that it fits in the map view's frame.
- `mapRectThatFits:edgePadding:` (page 21)
  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 15)
- `addAnnotations:` (page 15)
- `removeAnnotation:` (page 22)
- `removeAnnotations:` (page 23)

Declared In
`MKMapView.h`

annotationVisibleRect

The visible rectangle of the map view. (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 25)

`@property region` (page 12)

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.

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. 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 25)
`@property centerCoordinate` (page 10)

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 14)

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 13)

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 12) 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 15)
`– removeAnnotation:` (page 22)

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 15)

`- removeAnnotations:` (page 23)

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:` 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 23) 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:` 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 23) method.

Availability
Available in iOS 4.0 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:` 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 11) array.

`index2`

The index of the second object in the `overlays` (page 11) 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 11) 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 11) 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 11) 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 11) 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 18) 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 23)
`– addAnnotation:` (page 15)

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 18) 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 22)
`– addAnnotations:` (page 15)

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 10)
- `@property region` (page 12)

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.

Availability

Available in iOS 3.0 and later.

See Also

`@property region` (page 12)
`- setCenterCoordinate:animated:` (page 25)

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`.

Document Revision History

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

Date	Notes
2010-05-11	Added symbols introduced in iOS 4.0.
2009-07-27	Updated the description of the annotations property.
2009-05-12	New document that describes the class for managing an embeddable map interface.