Map Kit

- Overlays: `MKOverlay` „flächenhafte“ Annotationen eines `MKMapViews`

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

  - enthaltene Objekte müssen Protokoll `MKOverlay` implementieren:

```
@protocol MKOverlay <MKAnnotation>

// From MKAnnotation, for areas this should return
// the centroid of the area.
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate;

// boundingMapRect should be the smallest rectangle that completely contains the
// overlay. For overlays that span the 180th meridian, boundingMapRect should have
// either a negative MinX or a MaxX that is greater than MKMapSizeWorld.width.

@property (nonatomic, readonly) MKMapRect boundingMapRect

@optional
// Implement intersectsMapRect to provide more precise control over when
// the view for the overlay should be shown. If omitted, MKMapRectIntersectsRect
// ([overlay boundingRect], mapRect) will be used instead.

- (BOOL)intersectsMapRect:(MKMapRect)mapRect;

@end
```

Map Kit

- `overlays` ist immutable:
  - `- (void)addOverlay:(id <MKOverlay>)overlay;`
  - `- (void)addOverlays:(NSArray *) overlays;`
  - `- (void)removeOverlay:(id <MKAnnotation>) overlay;`
  - `- (void)removeOverlays:(NSArray *) overlays;`
- benutzen `MKOverlayViews` zur Anzeige
  - Zuordnung im `MKMapViewDelegate`:

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

Map Kit

- `MKOverlayView` hat keine Default-Darstellung (wie Pin bei Annotation)
- Ableitung muss Zeichnen implementieren:

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

- fast wie drawRect, aber mit Kontext vom Rufer
- CoreGraphics zum Zeichnen verwenden (muss thread-safe sein)

Map Kit

- das Rechteck wird in Karten-Koordinaten angegeben (nicht View-Koordinaten)
- Umrechnung leicht möglich:

```
- (MKMapPoint)mapPointForPoint:(CGPoint)point;
- (MKMapRect)mapRectForRect:(CGRect)rect;
- (CGPoint)pointForMapPoint:(MKMapPoint)mapPoint;
- (CGRect)rectForMapRect:(MKMapRect)mapRect;
```

Map Kit

- weitere `MKMapView` Eigenschaften
  - auch im IB einstellbar

```
@property MKMapType mapType;

// wo ist das Gerät?
@property BOOL showsUserLocation;
```

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

```
@property (readonly) BOOL isUserLocationVisible;


@property (readonly) MKUserLocation *userLocation;
// MKUserLocation ist ein Object konform zu MKAnnotation


// Einschränkung der möglichen Interaktion
@property BOOL zoomEnabled;
@property BOOL scrollEnabled;
```

Map Kit

- welches Gebiet zeigt der `MKMapView` ?

```
@property MKCoordinateRegion region;

typedef struct {
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;

typedef struct {
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
} MKCoordinateSpan;


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

// oder nur Mittelpunkt setzen

@property CLLocationCoordinate2D centerCoordinate;

- (void)setCenterCoordinate:(CLLocationCoordinate2D)center
                   animated:(BOOL)animated;
```

Map Kit

- Umrechnung von Karten/View-Koordinaten

```
- (CGPoint)convertCoordinate:(CLLocationCoordinate2D)coord
                 toPointToView:(UIView *)view;
- (CLLocationCoordinate2D)convertPoint:(CGPoint)point
                 toCoordinateFromView:(UIView *)view;
- (CGRect)convertRegion:(MKCoordinateRegion)region
           toRectToView:(UIView *)view;
- (MKCoordinateRegion)convertRect:(CGRect)rect
                 toRegionFromView:(UIView *)view;
```

Map Kit

- weitere MKMapViewDelegate-Methoden
  - (Google-Maps müssen geladen werden …)

```
- (void)mapViewWillStartLoadingMap:(MKMapView *)sender;
- (void)mapViewDidFinishLoadingMap:(MKMapView *)sender;
- (void)mapViewDidFailLoadingMap:(MKMapView *)sender
                       withError:(NSError *)error;
```

- viele weitere Hilfsfunktionen
  - s. Dokumentation, z.B. `MKMapRectContainsPoint`, `MKMapPointForCoordinate`, usw.

UITextField, UITextView

- editierbare Texte
- nur wenn wirklich nötig benutzen, alternative Eingaben (Gestures) immer bevorzugen
- Handhabbarkeit am besten am Gerät testen (nicht im Simulator mit Host-Tastatur)
- `UITextField`: kurze Eingaben (eine Zeile)
  - vieles einstellbar: Farbe, Font, Größe, Ausrichtung

UITextField

- (virtuelle) Tastatur erscheint, sobald ein `UITextField` zum ‚first responder' wird
  - Klick drauf oder
  - programmatisch: `[textField becomeFirstResponder];`
- (virtuelle) Tastatur verschwindet erst beim Aufruf von `[textField resignFirstResponder];`
- Tastatur verdeckt (von unten) den View:
  - selbst geeignet verschieben, oder (besser)
  - so anordnen, dass Eingabefeld dennoch sichtbar

UITextField

- Tastatur kann konfiguriert werden:
  - Einstellungen am UITextInputTraits-Protokoll (von UITextView implementiert):

```
@property UITextAutocapitalizationType autocapitalizationType;
// words, sentences, etc.
```

```
typedef enum {
    UITextAutocapitalizationTypeNone,
    UITextAutocapitalizationTypeWords,
    UITextAutocapitalizationTypeSentences,
    UITextAutocapitalizationTypeAllCharacters,
} UITextAutocapitalizationType;
```

```
@property UITextAutocorrectionType autocorrectionType;
//UITextAutocorrectionTypeYES/NO

@property UIReturnKeyType returnKeyType;
// Go, Search, Google, Done, etc.

@property BOOL secureTextEntry;
// for passwords, for example

@property UIKeyboardType keyboardType;
// ASCII, URL, PhonePad, etc.
```

UITextField

- Tastatur unterstützt korrekte Eingaben auf Basis eines Wörterbuches
  - Input-Toleranz ‚wahrscheinlicher nächster' Tasten wird (unsichtbar) vergrößert
- wie kommt man an den eingegebenen Text?
  - im `UITextFieldDelegate:`

```
- (BOOL)textFieldShouldReturn:(UITextField *)sender;
// sent when return key is pressed

(void)textFieldDidEndEditing:(UITextField *)sender;
// This method is called after the text field resigns
// its first responder status.
// NSString * sender.text
```

UITextView

- längere (mehrzeilige) editierbare Texte
- ansonsten ähnlich wie `UITextField` mit eigenem Delegate:

```
- (BOOL)textViewShouldReturn:(UITextField *)sender;
- (void)textViewDidEndEditing:(UITextField *)sender;
```

Modal View Controllers

- temporäre und blockierende (auf Eingabe wartende) Views
- vorheriger View komplett verdeckt bzw. inaktiv

Modal View Controllers

- Beispiel: Auswahl einer Adresse (ohne die es nicht (sinnvoll) weitergeht:

user class, not inUIKit

```
- (void)lookupAddress {
    AddressLookupViewController *alvc =
            [[AddressLookupViewController alloc] init];
    [self presentModalViewController:alvc animated:YES];
    [alvc release];
}
```

- wie/wann geht es weiter:
  - erst wenn am Controller, der `presentModalViewController` gerufen hat,
    `- (void)dismissModalViewControllerAnimated:(BOOL)animated;`
    gerufen wird (NICHT am alvc)

Modal View Controllers

- Art der Animation kann variieren:

```
/*UIViewController*/ @property UIModalTransitionStyle
modalTransitionStyle;

// mit Varianten:

UIModalTransitionStyleCoverVertical
// slides up and down from bottom of screen

UIModalTransitionStyleFlipHorizontal
// flips the current view controller view over for this one

UIModalTransitionStyleCrossDissolve
// old fades out as new fades in

UIModalTransitionStylePartialCurl
// only if presenter is full screen (and no more modal allowed)
```

Modal View Controllers

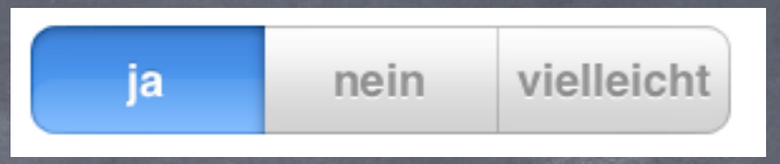
- aber der modale View Controller soll sicher keinen Rückverweis auf seinen Rufer haben!
- wie dann? Delegation!
  - nach einem selbst definierten Protokoll
  - die Implementation des modalen View Controllers muss dann explizit eine Delegate-Methode rufen

```
- (void)lookupAddress {
    ALVC *alvc = [[ALVC alloc] init];
    alvc.delegate = self; // the current Controller
    [self.presentModalViewController:alvc animated:YES];
    [alvc release];
}
```

Modal View Controllers

```
// (One of) ALVC's (user defined) delegate method(s)
- (void)addressLookupViewController:(ALVC *)sender
                    didSelectAddress:(Address *)anAddress
{
    // do something with the address the user selected (anAddress)
    [self dismissModalViewControllerAnimated:YES];
    // take sender off screen & release it
}

// + some Action in ALVC !!!
- (void)reactOnSomeThingInALVC {
   Address * lastSelectedAddress = ...;
   [self.delegate addressLookupViewController: self
                                  didSelectAddress: lastSelectedAddress];
}
```



UISegmentedControl

- Radio Buttons in mehreren Styles

```
@property UISegmentedControlStyle segmentedControlStyle;
```

```
typedef enum {
    UISegmentedControlStylePlain,
    UISegmentedControlStyleBordered,
    UISegmentedControlStyleBar,
    UISegmentedControlStyleBezeled,
} UISegmentedControlStyle;
```

- DI mit einem `NSArray` von `NSString`s oder `UIImage`s

```
NSArray *itemsArray = [NSArray arrayWithObjects:
                                    @“First”, @“Second”, nil];
UISegmentedControl *myControl =
        [[UISegmentedControl alloc] initWithItems:itemsArray];
```

  - bzw. nachträglich setzen/lesen:

```
- (void)setImage:(UIImage *)image forSegmentAtIndex:(int)index;
- (NSString *)titleForSegmentAtIndex:(int)index;
```

UISegmentedControl

- eine Target/Action in der die getroffene Auswahl erfragt werden kann

```
@property NSInteger selectedSegmentIndex;
// UISegmentedControlNoSegment if nothing is selected.
```

- Zuordnung der Action per IB oder

```
[segmentedControl addTarget:self // me the controller
                     action:@selector(action:)
           forControlEvents:UIControlEventValueChanged];
```

UIActionSheet und UIAlertView

- zwei Arten von "pop up and ask the user something" Mechanismen (speziell modal)

UIActionSheet und UIAlertView

- `UIActionSheet`: Wie weiter ?
- DI:

```
-(id)initWithTitle:(NSString *)title
          delegate:(id <UIActionSheetDelegate>)delegate
 cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
 otherButtonTitles:(NSString *)otherButtonTitles, ...;
```

- weitere Varianten hinzufügen

```
- (void)addButtonWithTitle:(NSString *)buttonTitle;
```

UIActionSheet und UIAlertView

- UIActionSheet anzeigen:

```
UIActionSheet *actionSheet = [[UIActionSheet alloc]
initWithTitle:...];


[actionSheet showInView:(UIView *)];
// centers the view on iPad (so don't use this on iPad)


[actionSheet showFromRect:(CGRect) inView:(UIView *) animated:
(BOOL)];
// good on iPad


[actionSheet showFromBarButtonItem:(UIBarButtonItem *) animated:
(BOOL)];
// good on iPad
```

UIActionSheet und UIAlertView

- Dissmiss? Automatisch nach Auswahl!
- oder (selten) programmatisch:

```
- (void)dismissWithClickedButtonIndex:(NSInteger)index
                             animated:(BOOL)animated;
// It is generally recommended to call this on
// UIApplicationDidEnterBackgroundNotification.
```

- Auswahl erfragen im Delegate:

```
- (void)actionSheet:(UIAlertView *)sender
clickedButtonAtIndex:(NSInteger)index;
```

UIActionSheet und UIAlertView

- Indizierung:

```
// spezielle Indizes:
@property NSInteger cancelButtonIndex;
@property NSInteger destructiveButtonIndex;

// andere Indizes in der Reihenfolge aus dem DI:
@property NSInteger firstOtherButtonIndex; // -1: keine others
@property NSInteger numberOfButtons; // alle!

- (NSString *)buttonTitleAtIndex:(NSInteger)index;
```

UIActionSheet und UIAlertView

- UIAlertView
  - Warnung über abnormale Zustände
  - asynchron/disruptiv
- DI

```
-(id)initWithTitle:(NSString *)title
           message:(NSString *)message
           // zusätzlich zum UIActionSheet
          delegate:(id <UIAlertViewDelegate>)delegate
 cancelButtonTitle:(NSString *)cancelButtonTitle
 otherButtonTitles:(NSString *)otherButtonTitles, ...;
```

UIActionSheet und UIAlertView

- weitere Buttons hinzufügen

```
- (void) addButtonWithTitle:(NSString *)buttonTitle;
```

- Anzeigen

```
UIAlertView *alertView = [[UIAlertView alloc]
initWithTitle: ...];

[alertView show];
// anders als UIActionSheet, immer in der Mitte des Displays
```

- Abfragen (`UIAlertViewDelegate`)

```
(void)alertView:(UIAlertView *)alertView
clickedButtonAtIndex:(NSInteger)buttonIndex; // ab 0 == cancel
```