UIView Animation

- Animierte Änderungen an Views:
  - View Hierarchie (Hinzufügen und Löschen von subviews)
  - hidden -> (undurchsichtiges) verblassen/erscheinen
  - frame -> (Größe/Position im Superview ändern)

    If the `transform` property does not contain the identity transform, modify the `bounds` or `center` properties instead.
  - transform (Verschieben, Drehen, Skalieren)
  - alpha (Transparenz)

UIView Animation

- durch Aufruf von `UIView`-Klassenmethoden
  - Übergabe eines Blockes, der das Ziel der Änderung beschreibt + Übergabe von Zeit- und weiteren Parametern der Animation
  - die im Block beschriebenen Änderungen werden SOFORT vollzogen, aber über die angegebene Zeit visuell animiert dargestellt
- einfachste Form

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                 animations:(void (^)(void))animations
// duration: in Sekunden
// animations: Block der die Änderung (ggf. an mehreren Views)
// vollzieht
```

UIView Animation

- Beispiel

```
[UIView animateWithDuration:1.0
                 animations: ^{
                     firstView.alpha = 0.0;  // fade out
                     secondView.alpha = 1.0; // fade in
}];
```

UIView Animation

- Angabe eines sog. Completion-Blocks möglich
  - führt nach (animierter Darstellung von) `animations` noch `completion` aus
  - `finished` gibt an, ob `animations` komplett ausgeführt wurde (s.u.)
  - falls `duration == 0`, wird `completion` am Beginn der nächsten Run Loop gerufen

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion
```

UIView Animation

- Beispiel

```
[UIView animateWithDuration: 0.2
        animations: ^{view.alpha = 0.0;}
        completion:^(BOOL finished){
         if (finished) [view removeFromSuperview];
}];
```

UIView Animation

- weitere Optionen möglich
  - starte Animation erst nach `delay` Sekunden (Änderung wird trotzdem SOFORT vollzogen)

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

UIView Animation

- Optionen

```
enum {
   UIViewAnimationOptionLayoutSubviews            = 1 <<  0,
   UIViewAnimationOptionAllowUserInteraction      = 1 <<  1,
   UIViewAnimationOptionBeginFromCurrentState     = 1 <<  2,
   UIViewAnimationOptionRepeat                    = 1 <<  3,
   UIViewAnimationOptionAutoreverse               = 1 <<  4,
   UIViewAnimationOptionOverrideInheritedDuration = 1 <<  5,
   UIViewAnimationOptionOverrideInheritedCurve    = 1 <<  6,
   UIViewAnimationOptionAllowAnimatedContent      = 1 <<  7,
   UIViewAnimationOptionShowHideTransitionViews   = 1 <<  8,

   UIViewAnimationOptionCurveEaseInOut            = 0 << 16,
   UIViewAnimationOptionCurveEaseIn               = 1 << 16,
   UIViewAnimationOptionCurveEaseOut              = 2 << 16,
   UIViewAnimationOptionCurveLinear               = 3 << 16,

   UIViewAnimationOptionTransitionNone            = 0 << 20,
   UIViewAnimationOptionTransitionFlipFromLeft    = 1 << 20,
   UIViewAnimationOptionTransitionFlipFromRight   = 2 << 20,
   UIViewAnimationOptionTransitionCurlUp          = 3 << 20,
   UIViewAnimationOptionTransitionCurlDown        = 4 << 20,
};
typedef NSUInteger UIViewAnimationOptions;
```

mit | verknüpfen

UIView Animation

- Beispiel (nested animation)

```
[UIView animateWithDuration:1.0
        delay: 1.0
 options:UIViewAnimationOptionCurveEaseOut
 animations:^{
     aView.alpha = 0.0;
     // Create a nested animation that has a different
     // duration, timing curve, and configuration.
     [UIView animateWithDuration:0.2
                delay:0.0
                options: UIViewAnimationOptionOverrideInheritedCurve |
                        UIViewAnimationOptionCurveLinear |
                        UIViewAnimationOptionOverrideInheritedDuration |
                        UIViewAnimationOptionRepeat |
                        UIViewAnimationOptionAutoreverse
                animations:^{
                        [UIView setAnimationRepeatCount:2.5];
                        anotherView.alpha = 0.0;
                }
                completion:nil];
 } // nach 1s: 1s aView || 0.2s * 2.5 * 2 (reverse) anotherView
```

UIView Animation

- Animierte Änderung der View Hierarchie

```
+ (void)transitionFromView:(UIView *)fromView // raus
                    toView:(UIView *)toView   // rein
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                completion:(void (^)(BOOL finished))completion;
```

  - entfernt `fromView` aus der View Hierarchie und fügt `toView` ein
  - soll stattdessen nur die Property `hidden` gesetzt werden, Option `UIViewAnimationOptionShowHideTransitionViews` setzen

UIView Animation

- Animierte Änderung der View Hierarchie
  - oder alles explizit im animations-Block

```
+ (void)transitionWithView:(UIView *)containerView
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                animations:(void (^)(void))animations
                completion:(void (^)(BOOL finished))completion;
```

UIImagePickerController

- Bilder (oder Videos) vom Nutzer auswählen/ aufnehmen lassen
  - nur Portrait-Mode
  - modaler View (`presentModalViewController:animated:`)
  - fix und fertig (keine Ableitung/Änderung erlaubt) [Ausnahme: `cameraOverlayView`]

UIImagePickerController

- Benutzung
  - Erzeugen (alloc/init) und Delegate setzen
  - Konfiguration (source, kind of media, user editability).
  - modal präsentieren
  - im Delegate auf Bildauswahl reagieren

UIImagePickerController

- Konfiguration erfragen
  - hängt von der Art des Gerätes ab (ohne/mit/mehrere Kamera(s), kann Movie aufnehmen/oder nicht)
  - Quelle:

```
+ (BOOL)isSourceTypeAvailable:
(UIImagePickerControllerSourceType)sourceType;
```

```
enum {
   UIImagePickerControllerSourceTypePhotoLibrary,
   UIImagePickerControllerSourceTypeCamera,
   UIImagePickerControllerSourceTypeSavedPhotosAlbum
};
typedef NSUInteger UIImagePickerControllerSourceType;
```

UIImagePickerController

- Konfiguration erfragen
  - hängt von der Art des Gerätes ab (ohne/mit/mehrere Kamera(s), kann Movie aufnehmen/oder nicht)
  - Medientyp:

```
+ (NSArray *)availableMediaTypesForSourceType:
(UIImagePickerControllerSourceType)sourceType;
// strange API: Array of NSStrings
// look for:
// kUTTypeImage // Bilder
// kUTTypeMovie // Audio und Video
```

UIImagePickerController

- Konfiguration erfragen
  - mehrere Kameras?

```
+ (BOOL)isCameraDeviceAvailable:
(UIImagePickerControllerCameraDevice)cameraDev;
```

```
enum {
   UIImagePickerControllerCameraDeviceRear,
   UIImagePickerControllerCameraDeviceFront
};
typedef NSUInteger UIImagePickerControllerCameraDevice;
```

  - Kamera mit Blitz?

```
+ (BOOL)isFlashAvailableForCameraDevice:
(UIImagePickerControllerCameraDevice) cameraDev;
```

  - was kann diese Kamera aufnehmen?

```
+ (NSArray *) // of NSNumbers
availableCaptureModesForCameraDevice:
(UIImagePickerControllerCameraDevice) cameraDev;
```

```
enum {
   UIImagePickerControllerCameraCaptureModePhoto,
   UIImagePickerControllerCameraCaptureModeVideo
};
typedef NSUInteger UIImagePickerControllerCameraCaptureMode;
```

UIImagePickerController

- Konfiguration setzen
  - im Rahmen der erfragten Möglichkeiten, z.B.

```
UIIPC *picker = [[UIIPC alloc] init];
picker.delegate = self;
// muss UINavigationControllerDelegate impl.
if ([UIIPC isSourceTypeAvailable:UIIPCSourceTypeCamera]) {
    picker.sourceType = UIIPCSourceTypeCamera;
} // photo library als default
NSString *desired = kUTTypeMovie;
// oder kUTTypeImage oder beide
if ([[UIIPC availableMediaTypesForSourceType:picker.sourceType]
containsObject:desired]) {
    picker.mediaTypes = [NSArray arrayWithObject:desired];
    [self presentModalViewController:picker animated:YES];
} else { /* Medientyp leider nicht zu bekommen */ }
```

```
UIIPC == UIImagePickerController
```

UIImagePickerController

- Auswahl kann ggf. noch editiert werden

```
// UIImagePickerController Editability
@property BOOL allowsEditing;
// YES: editieren Möglich, der Delegate bekommt Roh- und
// geänderte Daten
```

- bei Videos können Qualität und Dauer gesetzt werden

```
@property NSTimeInterval videoMaximumDuration; // in Sekunden
@property UIImagePickerController QualityType qualityType;
```

```
enum {
   UIImagePickerControllerQualityTypeHigh    = 0,
   UIImagePickerControllerQualityType640x480 = 3,
   UIImagePickerControllerQualityTypeMedium  = 1,  // default value
   UIImagePickerControllerQualityTypeLow     = 2
};
typedef NSUInteger UIImagePickerControllerQualityType;
```

UIImagePickerController

- Delegate bekommt Auswahl/Aufnahme

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info

/* A dictionary containing the original image and the edited
image, if an image was picked; or a filesystem URL for the movie,
if a movie was picked. Your delegate object's implementation of
this method should pass the specified media on to any custom code
that needs it, and should then dismiss the picker view. */y
```

- oder Abbruch der Auswahl

```
- (void)imagePickerControllerDidCancel:
                            (UIImagePickerController *)picker;
/* Your delegate's implementation of this method should dismiss
the picker view by calling the dismissModalViewControllerAnimated:
method of the parent view controller. Implementation of this
method is optional, but expected. */
```

UIImagePickerController

- alle Daten werden aus der info extrahiert:

  Schlüssel sind NSStrings:

```
NSString *const UIImagePickerControllerMediaType;     // kUTTypeImage/kUTTypeMovie
NSString *const UIImagePickerControllerOriginalImage; // original UIImage
NSString *const UIImagePickerControllerEditedImage;   // editiertes UIImage
NSString *const UIImagePickerControllerCropRect;      // Bildausschnitt
NSString *const UIImagePickerControllerMediaURL;      // Movie URL
NSString *const UIImagePickerControllerReferenceURL;  // orig. Bild als File
NSString *const UIImagePickerControllerMediaMetadata; // Metadaten
```

- Aufnahme (mit Metadaten) als Bild im Photoalbum speichern

```
/* ALAssetsLibrary */
- (void)writeImageToSavedPhotosAlbum:(CGImageRef)imageRef
                            metadata:(NSDictionary *)metadata
                     completionBlock:(ALAssetsLibraryWriteImageCompletionBlock)block

// UIImage: @property(nonatomic, readonly) CGImageRef CGImage
```

UIImagePickerController

```
typedef void (^ALAssetsLibraryWriteImageCompletionBlock)
                              (NSURL *assetURL, NSError *error);
```

- Beispiel

```
- (void)imagePickerController:(UIImagePickerController *)picker
    didFinishPickingMediaWithInfo:(NSDictionary *)info {

    UIImage *image = [info valueForKey: UIImagePickerControllerOriginalImage];
    NSDictionary *metadata =
                    [info valueForKey:UIImagePickerControllerMediaMetadata];

    ALAssetsLibrary *library = [[ALAssetsLibrary alloc] init];

    [library writeImageToSavedPhotosAlbum: image.CGImage
                                 metadata:metadata
                          completionBlock:^(NSURL *assetURL, NSError *error)
                                          { }
    ]; // asyncronous write within a separate thread

    [library release];

}
```

By the way: toll-free bridged types

Core Foundation type	Foundation class
CFArrayRef	NSArray
CFAttributedStringRef	NSAttributedString
CFCalendarRef	NSCalendar
CFCharacterSetRef	NSCharacterSet
CFDataRef	NSData
CFDateRef	NSDate
CFDictionaryRef	NSDictionary
CFErrorRef	NSError
CFLocaleRef	NSLocale
CFMutableArrayRef	NSMutableArray
CFMutableAttributedStringRef	NSMutableAttributedString
CFMutableCharacterSetRef	NSMutableCharacterSet
CFMutableDataRef	NSMutableData
CFMutableDictionaryRef	NSMutableDictionary
CFMutableSetRef	NSMutableSet
CFMutableStringRef	NSMutableString
CFNumberRef	NSNumber
CFReadStreamRef	NSInputStream
CFRunLoopTimerRef	NSTimer
CFSetRef	NSSet
CFStringRef	NSString
CFTimeZoneRef	NSTimeZone
CFURLRef	NSURL
CFWriteStreamRef	NSOutputStream

Sound

- diverse Möglichkeiten auf unterschiedlichsten Abstraktionsniveaus:
  - Core Audio
  - AVFoundation
  - Media Player
  - OpenAL
  - …
- hier nur ‚simple Sounds abspielen/aufnehmen', 2 Varianten
  - System Sound API (für ultra-simple "sound effects")
  - AVAudioPlayer/AVAudioRecorder

System Sound API

- ganz kurze, nichtwiederholbare Sounds, keine Lautstärkeregelung
  AIFF or WAV (uncompressed formats).
- Sound registrieren

```
SystemSoundID mySound;
NSString *soundFilePath [[NSBundle mainBundle]
                         pathForResource:@“mySound” ofType:@“caf”];
NSURL *soundFileURL = [NSURL URLForString:soundFilePath];
AudioServicesCreateSystemSoundID(
                                 (CFURLRef)soundFileURL, &mySound);
```

- Abspielen

```
AudioServicesPlaySystemSound(mySound);
```

- Freigeben

```
AudioServicesDisposeSystemSound(mySound);
```

System Sound API

- Vibrieren

```
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
// oder
AudioServicesPlayAlertSound(mySound);
// depends on “vibrate with ring” setting
```

- unkomprimierte Klänge aus komprimierten machen:
  - Unix Kommando

```
/usr/bin/afconvert -f aiff -d BEI16 input.mp3 output.aif
diverse Formate und Kodierungen
Big Endian Integer 16 Bits
```

AVAudioPlayer

- komplexere (aber immer noch kleine) Klänge (keine Tracks)
  - z.B. zyklische Hintergrundmusik
  - Looping, Positionieren, Abspielen und Anhalten
  - Lautstärke einstellbar
  - mehrere (miexed) Sounds zugleich möglich
  - OO API
  - noch viel mehr Fileformate
- Einrichten

```
NSString *soundFilePath = [[NSBundle mainBundle]
                          pathForResource:@“mySound” ofType:@“mp3”];
NSURL *soundFileURL = [NSURL URLForString:soundFilePath];
AVAudioPlayer *player = [[AVAudioPlayer alloc]
                          initWithContentsOfURL:soundFileURL];
```

AVAudioPlayer

- Abspielen/Anhalten

```
[player play];
[player pause];
```

- Scrubbing (schnell vor/zurück)

```
- (void)scrub:(UISlider *)sender {
    player.currentTime = player.duration * sender.value;
    // assuming slider goes from 0 to 1
}
```

- Lautstärke

```
player.volume = 0.75;   // from 0 to 1
```

AVAudioPlayer

- AVAudioPlayerDelegate
  - wird informiert, wenn Player unterbrochen wurde (z.B. durch eingehenden Anruf)

```
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)sender
                       successfully:(BOOL)success;

- (void)audioPlayerBeginInterruption:(AVAudioPlayer *)sender;

- (void)audioPlayerEndInterruption:(AVAudioPlayer *)sender;

- (void)audioPlayerDecodeErrorDidOccur:(AVAudioPlayer *)sender
                                 error:(NSError *)error;
```

AVAudioRecorder

- Einrichten

```
NSURL *soundFileURL = ...;
// wohin schreiben
AVAudioRecorder *recorder = [[AVAudioRecorder alloc]
                    initWithURL:soundFileURL
                       settings:(NSDictionary *)settings
                          error:(NSError **)errorOut];
```

- settings: sample rate, number of channels, usw., nil für defaults.

AVAudioRecorder

- Aufnahme Start/Stop z.B.

```
- (IBAction) toggleRecorder {
    if (!recorder.recording) {
        [recorder record];
    } else {
        [recorder pause];
    }
}
```

- Aufnahmelänge (seit Start, bei Stop == 0)

```
@property (readonly) NSTimeInterval currentTime;
```

MediaPlayer Framework

- z.B. für Songs aus der iPod-Library
  - einen `MPMediaPickerController` oder eine `MPMediaQuery` erzeugen
  - `MPMusicPlayerController` erzeugen, um die ausgewählten Titel abzuspielen
- aber auch für Filme

MPMoviePlayer[View]Controller

- Filme abspielen
  - unterstützt H.264 Baseline und MPEG-4 Part 2

```
/* MPMoviePlayerController*/
- initWithContentsOfURL:(NSURL *)movieURL;
```

  - verwendet nutzerdefinierten View für den eigentlichen Playerframe oder
  - vordefinierten full-screen `MPMoviePlayerViewController`

MPMoviePlayerViewController

- mit speziellen (per Category in MP an UIViewController angehängten) Methoden

```
// present fully&modally
[self presentMoviePlayerViewControllerAnimated:
(MPMovePlayerViewController *)controller];

// dismiss
[self dismissMoviePlayerViewControllerAnimated];
```