Praktikum Entwicklung von Mediensystemen mit iOS

SS 2011

Michael Rohs
michael.rohs@ifi.lmu.de
MHCI Lab, LMU München

Timeline

Date	Topic/Activity
5.5.2011	Introduction and Overview of the iOS Platform
12.5.2011	Implementing a User Interface
19.5.2011	App Architecture, Touch Input, Saving Data
26.5.2011	HTTP, Location, Sensors; Brainstorming
2.6.2011	no class (Christi Himmelfahrt)
9.6.2011	… (Milestones)
16.6.2011	
23.6.2011	no class (Fronleichnam)
30.6.2011	
07.07.2011	
14.07.2011	
21.07.2011	Project Presentation
28.07.2011	Evaluation
	Paper Writing

Today

- Alerts, Action Sheets, text input
- Application architecture
- Table views
- Multiview applications
- Touch input
- Saving data
- Exercise 2

“Hello World” Steps

- Explain #pragma
- Showing a UIAlertView
- Action sheets
  - Implement UIActionSheetDelegate in .h file
  - Construct, showInView, release
  - Implement delegate method clickedButtonAtIndex
- Text input
  - Add UITextField in Interface Builder
  - Add member variable and property to .h, synthesize in .m
  - Declare UITextFieldDelegate in .h
  - Implement delegate methods in .m, set label text on end editing
  - Set delegate in viewDidLoad method

Hello World Application Architecture

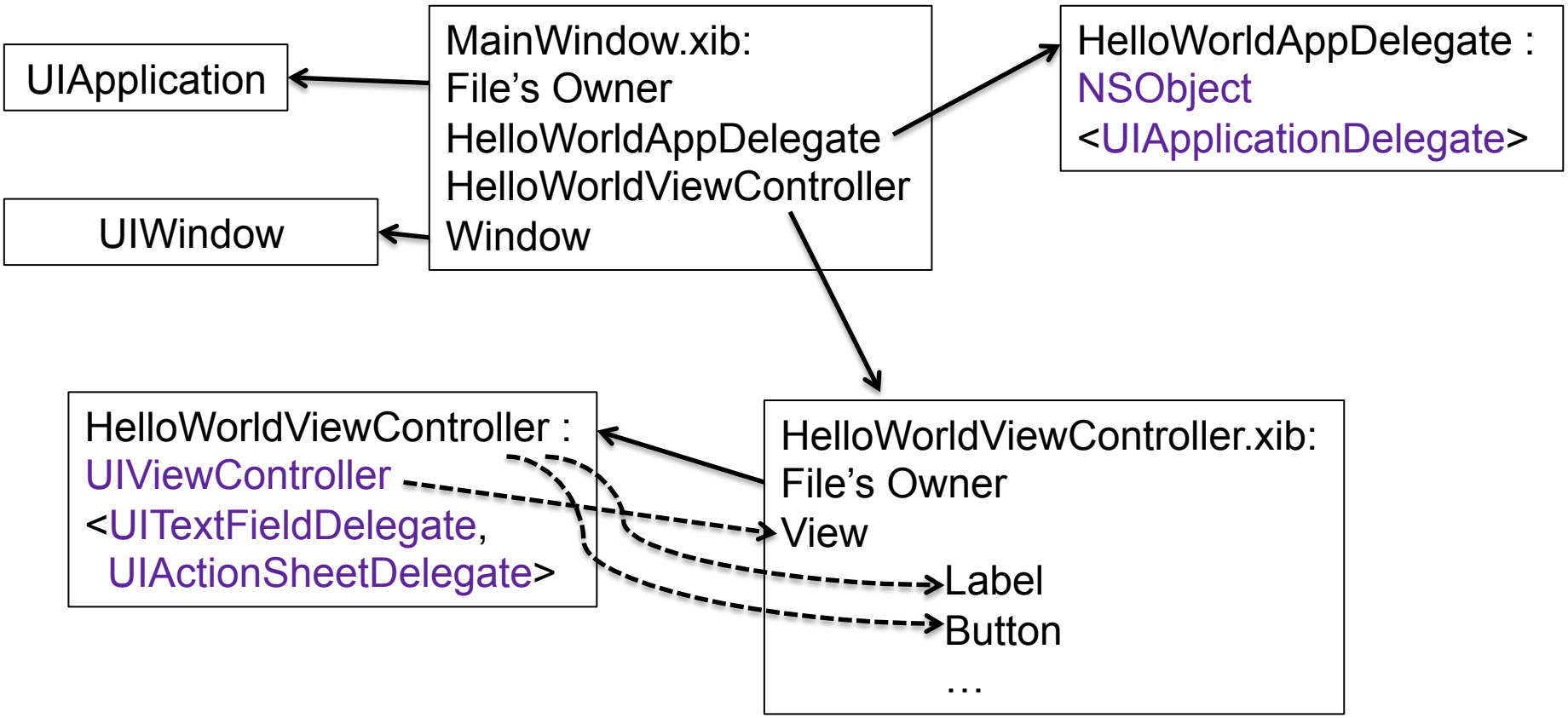
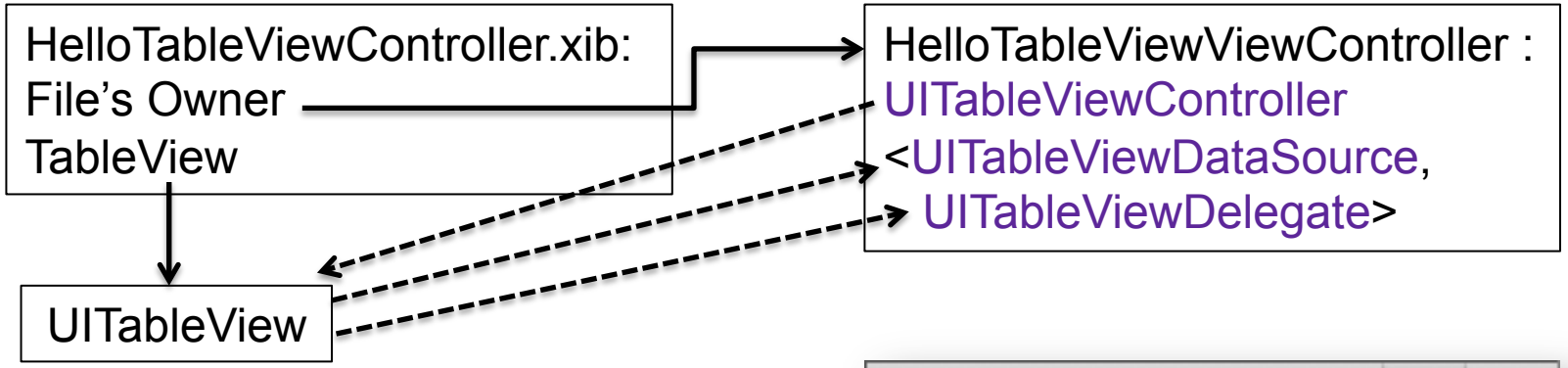
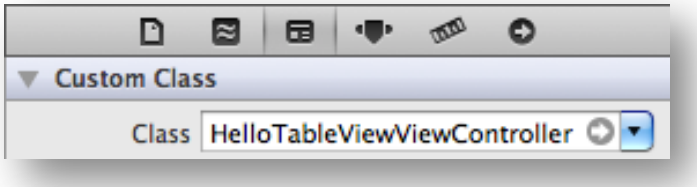
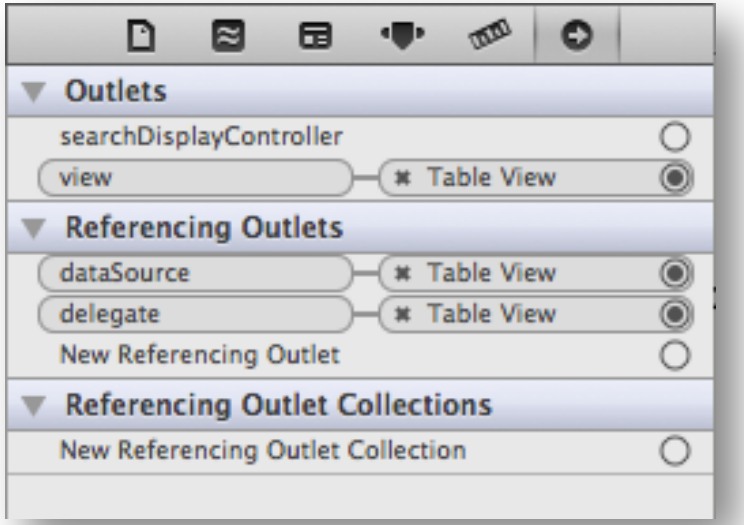


Table Views


HelloTableViewController.xib:
File's Owner
TableView
UITableView
HelloTableViewViewController :
UITableViewController
<UITableViewDataSource,
UITableViewDelegate>


Custom Class
Class
HelloTableViewViewController
Custom Class
Class
UITableView


Outlets
searchDisplayController
view
Table View
Referencing Outlets
dataSource
Table View
delegate
Table View
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection


instantiates
A
B
references
A
B

UIViewController subclasses

- View lifecycle
  - (void)viewDidLoad
  - (void)viewDidUnload
- View events
  - (void) viewWillAppear:(BOOL)animated
  - (void) viewWillDisappear:(BOOL)animated
  - (void) viewDidAppear:(BOOL)animated
  - (void) viewDidDisappear:(BOOL)animated
- Rotation settings and events

  interfaceOrientation property

  – shouldAutorotateToInterfaceOrientation:
- many more… → see documentation

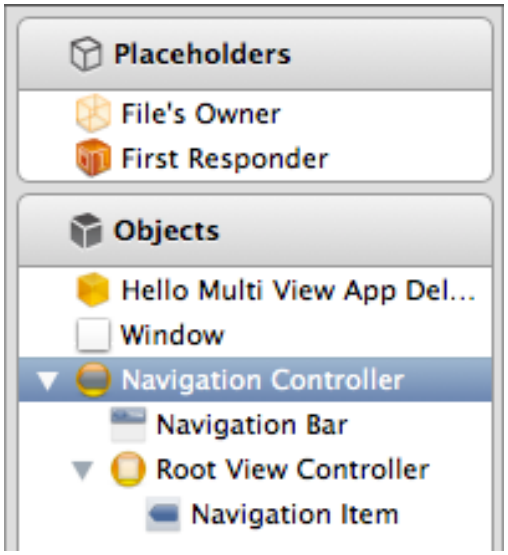
UITableViewDataSource (Protocol)

- Configuring a Table View
  - tableView:cellForRowAtIndexPath: required method
  - numberOfSectionsInTableView:
  - tableView:numberOfRowsInSection: required method
  - sectionIndexTitlesForTableView:
  - tableView:sectionForSectionIndexTitle:atIndex:
  - tableView:titleForHeaderInSection:
  - tableView:titleForFooterInSection:
- Inserting or Deleting Table Rows
  - tableView:commitEditingStyle:forRowAtIndexPath:
  - tableView:canEditRowAtIndexPath:
- Reordering Table Rows
  - tableView:canMoveRowAtIndexPath:
  - tableView:moveRowAtIndexPath:toIndexPath:

UITableViewDelegate (Protocol)

- Configuring Rows for the Table View
  - tableView:heightForRowAtIndexPath:
- Managing Accessory Views
  - tableView:accessoryButtonTappedForRowWithIndexPath:
- Managing Selections
  - tableView:{will,did}SelectRowAtIndexPath:
  - tableView:{will,did}DeselectRowAtIndexPath:
- Modifying the Header and Footer of Sections
  - tableView:viewFor{Header,Footer}InSection:
  - tableView:heightFor{Header,Footer}InSection:
- Editing Table Rows
- Reordering Table Rows

MultiView Application Architecture



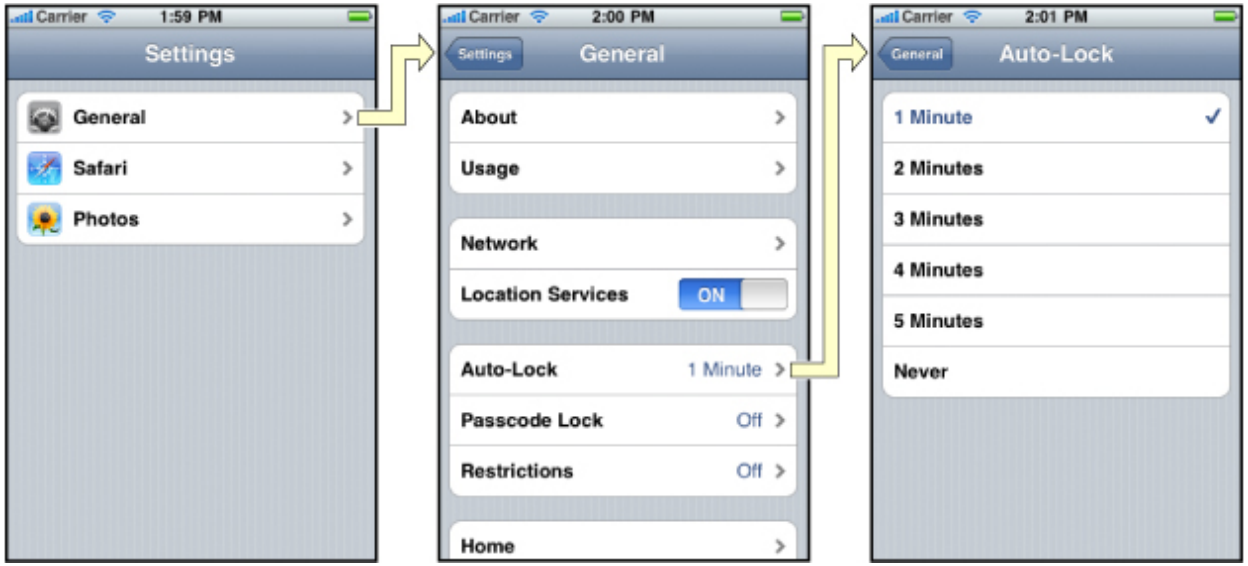
Navigation Controller Views

Source: http://developer.apple.com/library/ios/#documentation/uikit/reference/UINavigationController_Class/Reference/Reference.html

Pushing a new View onto the View Stack

- Loading and pushing the new view controller

```
MyDetailViewController *d = [[MyDetailViewController alloc]
initWithNibName:@"MyDetailViewController" bundle:nil];
d.labelText = [data objectAtIndex:indexPath.row];
[self.navigationController pushViewController:d animated:YES];
[d release];
```



Source: http://developer.apple.com/library/ios/#documentation/uikit/reference/UINavigationController_Class/Reference/Reference.html

Touch Input

- Overwrite methods in UIView or UIImageView:

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch *touch = [touches anyObject];
    CGPoint p = [touch locationInView:self];
    traceCount = 0;
    trace[traceCount++] = p;
    [self updateDisplay];
}
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;
```

Accessing Application Directories

- Application sandbox: can only access own app folder

```
NSString *homeDir = NSHomeDirectory();
NSString *tmpDir = NSTemporaryDirectory();
NSArray *paths = NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, YES);
path = [paths objectAtIndex:0];
```

- Accessing data bundled as an application resource

```
NSString *fileName = [homeDir
stringByAppendingPathComponent:@"/Test.app/mydata.dat"];
```

Loading and Saving Binary Data

- NSData is a container for bytes
- Loading arbitrary binary data

```
NSData *d = [[NSData alloc] initWithContentsOfFile:fileName];
NSMutableData *m = [NSData dataWithContentsOfFile:fileName];
```

- Accessing the data

```
const char* b = [d bytes]; // d is immutable → cannot be modified
char* c = [m mutableBytes]; // m is mutable → can be modified
```

- Saving arbitrary binary data

```
[c writeToFile:fileName atomically:YES];
```

- Appending to mutable data object

```
[m appendBytes:myBytes length:myBytesCount];
```

Binary Loading & Saving Code Snippet

```
NSString *fileName = @"/.../myTestFile.dat";
NSData *d = [[NSData alloc] initWithContentsOfFile:fileName];
const char* b = [d bytes];
// use the data, cannot modify
[d release];
NSMutableData *m = [[NSData alloc] initWithContentsOfFile:fileName];
char* c = [m mutableBytes];
c[0] = 42; // modify the data (direct access to data)
char *myBytes = "123";
int myBytesCount = strlen(myBytes);
[m appendBytes:myBytes length:myBytesCount];
[m writeToFile:fileName atomically:YES];
[m release];
```

Loading XML Data

- XML data and property lists for structured data

```
NSDictionary *d = [NSDictionary dictionaryWithContentsOfFile: f];
```

- Predefined elements dict, array, string, key, integer, etc.
- Example (a dictionary containing an array of dictionaries)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>images</key>
        <array>
                <dict>
                        <key>title</key><string>My Image Title</string>
                        <key>image</key><string>MyImage.png</string>
                </dict>
                <dict>
                        <key>title</key><string>Another Title</string>
                        <key>image</key><string>AnotherImage.png</string>
                </dict>
        </array>
</dict>
</plist>
```

Loading and Saving Object Hierarchies

- Declaring objects as archiveable by implementing NSCopying protocol
  - initWithCoder, encodeWithCoder
- Handle archiving in these methods
  - All objects handled by coder need to conform to NSCopying
- NSKeyedArchiver to save object hierarchy
- NSKeyedUnarchiver to load object hierarchy

Declaring Classes as Archiveable

```
@interface MyClass : NSObject <NSCopying> {
    NSString *lastName;
    NSMutableArray *firstNames;
}
- (id) initWithCoder:(NSCoder *)decoder {
    self = [super init];
    self.lastName = [decoder decodeObjectForKey:@”lastName"];
    self.firstNames = [decoder decodeObjectForKey:@”firstNames"];
    return self;
}
- (void) encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:lastName forKey:@”lastName"];
    [encoder encodeObject:firstNames forKey:@"firstNames"];
}
```

Saving Object Hierarchies

- Archiving (simple version, one root)

```
[NSKeyedArchiver archiveRootObject:myRoot toFile:myFile];
```

- Archiving (complex version, multiple roots)

```
NSMutableData *data = [[NSMutableData alloc] init];
NSKeyedArchiver *archiver = [[NSKeyedArchiver alloc]
                                    initForWritingWithMutableData:data];
[archiver encodeObject:myRoot1 forKey:@”myRoot1"];
[archiver encodeObject:myRoot2 forKey:@”myRoot2"];
[archiver finishEncoding];
[data writeToFile:myFile atomically:YES];
[archiver release];
[data release];
```

Loading an Object Hierarchy

- Unarchiving an object hierarchy

```
self.object = [NSKeyedUnarchiver unarchiveObjectWithFile:fileName];
```

- Root object needs to be retained after unarchiving

  In the example above it is a retained property