iOS Application Development

Lecture 2: Seminar Topics and Unit 1

Simon Völker & Philipp Wacker
Media Computing Group
RWTH Aachen University

hci.rwth-aachen.de/ios
Seminar

• 2 Presentations per Session

• 20 min presentation, ~10 min discussion

• 3 people per group

• Dates:
  • 18.11, 19.11, 25.11, 26.11, 2.12, 3.12, 9.12

• Order is not fixed yet!

• Finished version 1 week before the presentation

• 15 min slide and content discussions 1 week before the presentation
Seminar

• Framework overview
• Conceptional structure
• Show how to use them

• Structure:
  • Brief introduction
  • Basis steps how to use the framework
  • Demo (live coding)
  • Explain one or two advanced features and show how to use them.

• Deliverables: Slides and Demo code
Topic: Sprite Kit

• 2D Gaming Engine

• Visuals are Sprites, Shapes

• Includes:
  • Animations
  • Physics
  • Lighting
  • Particle effects

We will show a very short demo presentation
1. Core Data

- Manages the model in MVC
- Persistant data storage
- Managed objects
- Alternative to SQL

https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html


2. Scene Kit

- High-level 3D game engine
- Scene graphs
- Animations
- Physics


3. Metal

• Low level renderer

• Modern alternative to OpenGL

• Precompiled shaders

• Multithreading

• Speed increase by reducing CPU load


4. CloudKit & iCloud Drive

- Cloud storage
- Cross devices files sharing
- Storage optimization

https://developer.apple.com/icloud/

5. Core ML + Create ML

• Machine learning

• Computer vision (Vision Framework)

• Natural language processing


6. SiriKit

- Add Siri to your app
- Voice commands


7. WatchOS

• Design apps for the Apple Watch

• Native WatchOS apps


8. MapKit, CoreLocation

- Map navigation
- Location detection
- Geofencing


9. Networking on iOS

- CoreBluetooth
- Sockets
- Bonjour

10. Extensions & Inter-App communication

- Deep Linking
- App Extensions (Services)
- Files app

https://developer.apple.com/app-extensions/

11. Debugging in Xcode and Instruments

- Instruments
- Energie consumptions

12. ClassKit

- Create class exercises
- Schoolwork apps
- Student and teach classroom curriculum support


13. RealityKit & Reality Composer

- Simulate and render 3D content in AR
- Prototype AR scenes and apps

14. Bringing People into AR

- Detect people and object in a AR scene
- Body tracking


15. GameplayKit

- Gameplay logics
- Agents
- Pathfinding
- Rule systems

https://developer.apple.com/documentation/gameplaykit


16. SwiftUI

• Design Apps using simple SwiftUI framework

• Declarative syntax

• Design tools

17. AVKit

- Low-Latency HLS
- Media playback
- Multi-Camera capture for iOS


Forming Groups & Vote for the Topic

Give each topic a different rating.

Please rate your interest in each topic on a scale from 1 (highly interested) to 17 (not interested at all).

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Core Data</td>
</tr>
<tr>
<td>2. Scene Kit</td>
</tr>
<tr>
<td>3. Metal</td>
</tr>
<tr>
<td>4. CloudKit &amp; iCloud Drive</td>
</tr>
<tr>
<td>5. Core ML + Create ML</td>
</tr>
<tr>
<td>6. SiriKit</td>
</tr>
<tr>
<td>7. WatchOS</td>
</tr>
<tr>
<td>8. MapKit, CoreLocation</td>
</tr>
<tr>
<td>9. Networking on iOS</td>
</tr>
<tr>
<td>10. Extensions &amp; Inter-App communication</td>
</tr>
<tr>
<td>11. Debugging in Xcode and Instruments</td>
</tr>
<tr>
<td>12. ClassKit</td>
</tr>
<tr>
<td>13. RealityKit &amp; Reality Composer</td>
</tr>
<tr>
<td>14. Bringing People into AR</td>
</tr>
<tr>
<td>15. GameplayKit</td>
</tr>
<tr>
<td>16. Swift UI</td>
</tr>
<tr>
<td>17. AVKit</td>
</tr>
</tbody>
</table>
CHAPTER 1

Swift
History

• Introduced at WWDC 2014

• Influenced by C and Objective-C

• But designed to be simpler to learn and not dependent on older programming languages

• “Safe, fast, and expressive”

• Open source since 2015
Characteristics

• Clean syntax

• Optionals

• Type inference

• Type safety

• Automatic Reference Counting (ARC)
Characteristics

• Tuples and multiple return values

• Generics

• Fast and concise iteration over collections

• Structs that support methods, extensions, and protocols

• Map, filter, reduce, and other functional programming patterns

• Powerful error handling
Playgrounds

```swift
import UIKit

var str = "Hello, playground"
print("Hello, world!")
```

Hello, world!
Variables and Constants

- Variables are declared with `var`

```swift
var x = 100
```

- Constants are declared with `let`

```swift
let pi = 3.14
// error! a constant can't be changed afterwards
```
```swift
import UIKit

let x = 1
var y = 2

y = 3

x = 4  // Cannot assign to value: 'x' is a 'let' constant
```
Type Inference

• Swift automatically chooses the adequate data type for a variable/constant

```swift
var x = 100
    x = 99.5 // error! x is of type Integer

var x = 99.5
    x = 100 // correct (x = 100.0, type: Double)
```

• You can also explicitly specify the type

```swift
var aString : String
```
Data Types & Type Inference

```swift
var x1 = 100 // Int
var x2 = 0.5 // Double
var x3 = x1 + x2 // error! can’t add Int and Double
var x3 = Double(x1) + x2 // works! explicit type casting to Double
var x4 = 0.5 + 100 // works! compiler adds before setting the data type
print("x4 = \(x4)") // x4 = 100.5
var 😄 = "LOL" // String
```
Optionals

- By default, variables and constants cannot be nil

- Optionals: variables that can also be nil

  ```swift
  var opt: Int? = 3
  opt = nil
  ```

- Normal variables and Optionals are incompatible to each other

  ```swift
  var number = Int("abc")    // nil. Type of Number: Int?
  print(number + 3)          // error! Int? != Int
  var i: Int = number         // error!
  ```
Tuples

• Tuples can have multiple elements of different types

```swift
var tuple = (1, 2.0, "hello", true)
var (a,b,c,d) = tuple // a = 1, …
print(tuple.2) // "hello" (starts at 0)
```
//: Playground - noun: a place where people can play

```swift
import UIKit

let x = 1
var y = 2

y = 3
// x = 4

var tuple = (1, 2.0, "hello", true)
var (a, b, c, d) = tuple
print(tuple.2)
print(b)

print(tuple)
```

```
Int 0
Double 1
String 2
Bool 3
```
Control Flow

• If/else

```swift
var x=3
if x<0 {
    print("x is negative")
} else if x==0 {
    print ("x is zero")
} else {
    print("x is positive")
}
```

• Ternary Operator

```swift
var largest: Int
let a = 15
let b = 4
if a > b {
    largest = a
} else {
    largest = b
}
largest = a > b ? a : b
```
• Switch

```
let pt = (0.0, 0.0)

switch pt {
    case (0,0):
        print("Origin.")
    case (_,0):
        print("On x-axis.")
    case (0,_):
        print("On y-axis.")
    default:
        print("Elsewhere.")
}
```

```
switch distance {
    case 0...9:
        print("You are close")
    case 10...500:
        print("Take a car")
    default:
        print("Too far away.")
}
```
CHAPTER 4
Development Environment
Xcode

```swift
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        print("Hey iOS class")
        print("This is the Editor")
        print("To the left is the Navigator")
        print("The Toolbar is on top")
        print("The Debugging area is below")
        print("To the right is the Utility area")
    }
}
```
Playground Demo

```swift
import UIKit

var str = "Hello, playground"
print("Hello, world!")
```

Hello, world!
Xcode

• 5 areas
  1. Editor
  2. Toolbar
  3. Navigator
  4. Debugging
  5. Utility
.xcodeproj File

• Settings file for your project
Building/Running

No Device
- No devices connected to 'My Mac'...

Build Only Device
- Generic iOS Device

iOS Simulators
- iPad Air (3rd generation)
- iPad Pro (9.7-inch)
- iPad Pro (11-inch)
- iPad Pro (12.9-inch) (3rd generation)
- iPhone 8
- iPhone 8 Plus
- iPhone 11
- iPhone 11 Pro
- iPhone 11 Pro Max

Add Additional Simulators...
Warnings & Errors

- Warnings don’t prevent your app from compiling & running
  - Code that never gets executed
  - Variable that does not change
  - Deprecated code

- Errors prevents the app from building
  - Invalid code (typo, variable declaration, function calling)
  - Xcode often provides suggestions & fixes
Debugging

- Set breakpoints for execution on simulator and device
- Continue, Step over, Step into, Step out
Documentation

• Quick Help (Option+Click)
• Documentation Browser
• Programming Guides
Interface Builder

- Visually define your UI
Outlets & Actions

• Connect your UI elements with your code: Right-click + Drag

```swift
override func didReceiveMemoryWarning(super) {
    super.didReceiveMemoryWarning
    // Dispose of any resources recreated.
}
```
Outlets & Actions

- **IBOutlet**
  - Access the UI element from code
  
  ```swift
  @IBOutlet weak var textLabel: UILabel!
  ```

- **IBAction**
  - Receive UI events
  
  ```swift
  @IBAction func buttonPressed(_ sender: Any) {}
  ```
Summary

• Basic concept of Swift

• Data Types, Control Flow, Tuples

• Development Environment
  • Xcode
  • Playgrounds

• Tomorrow: Unit 2: Strings, Classes and Structs…