COCOA WORKSHOP PART I

Andreas Monitzer

2009-02-17

Workshop Schedule

1. Introduction, Foundation	**2009-02-17**
2. GUI Programming	2009-02-19
3. Hands-On	2009-02-24
4. Advanced	2009-02-26

Structure

- Introduction
- Objective C
- Memory Management
- Important Foundation Classes
- Creating Classes
- Big Example 1

INTRODUCTION

About Myself

- Dipl.-Ing. Andreas Monitzer
- 27 years old
- Working as programmer
- Cocoa Programming since Mac OS X Public Beta (2000)
  - pptp-gui
  - CocoaDevCentral

My Work

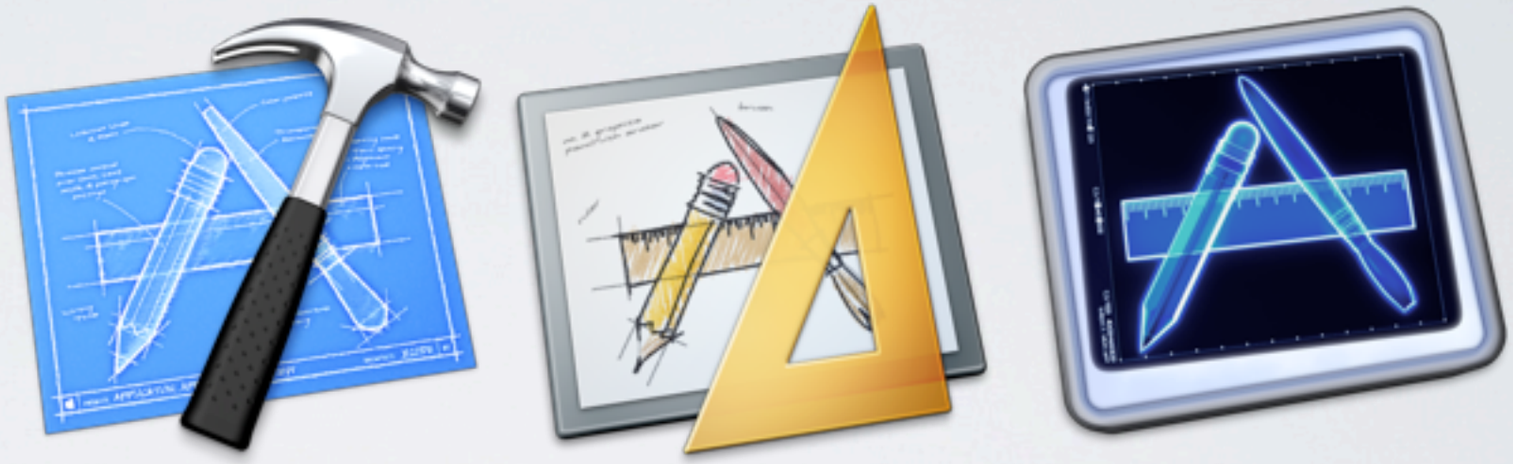
- DigiTunnel
- PulpFiction
- MailDrop 2 (never left beta)
- Adium (GSoC 2006 & 2007)

Target

- Write programs for Mac OS X in its native environment
- "How to Make a Zillion Dollars and Not Lose Your Soul" (Whil Shipley)
- Cocoa programming is fun!

Things to Learn

- Tools



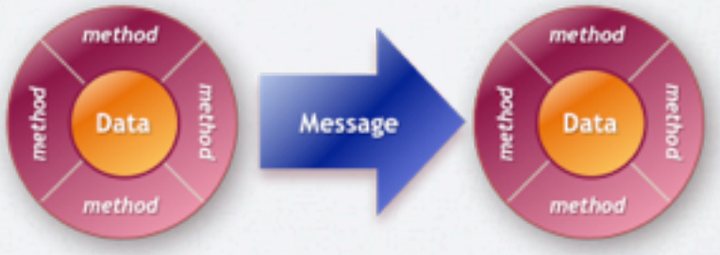
...

- Frameworks

Foundation AppKit

...

- Language



Objective C

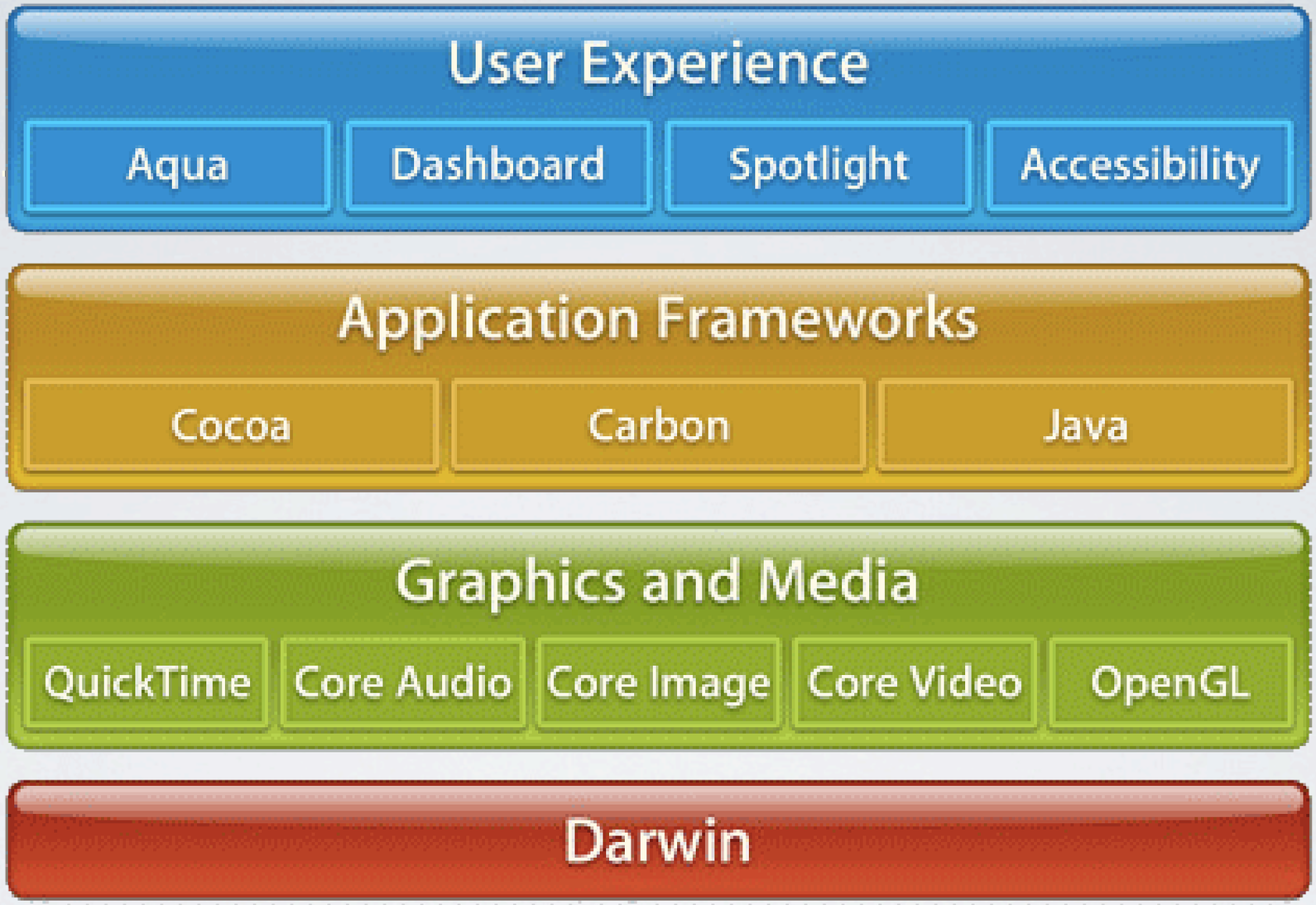
LET'S GET STARTED

Objective C

- Dynamic Object-Oriented Programming Language
- First published in 1986, pretty much simultaneously to C++
- Popularized by NeXT: NeXTSTEP/OpenStep
- Fully backwards-compatible with C
  - Objective C-specific keywords use @-prefix

Cocoa

- API for user interface development
- Written in Objective C, but can be accessed from many languages
  - Objective C, Perl, Python, Ruby, F-Script, formerly Java



MAC OS X ARCHITECTURE

Cocoa Structure

- Divided into two main frameworks (libraries)
  - Foundation
  - AppKit
- Additional frameworks available
  - WebKit, CoreFoundation, QTKit, …

Today: Foundation

- Memory Management, Collections, Event Handling, …
- Can be used for command line tools
- Objective C is pretty useless without Foundation

OBJECTIVE C

Classes

- Similar to Java
  - No multiple inheritance
  - Java interfaces / C++ pure abstract classes = ObjC protocols
  - Object introspection (very frequently used)
- Split into interface and implementation
- type "`id`" = any object

ObjC messaging

- Parameters to methods are interleaved with the name

  - ":" prefix

```
[object abc:param1 def:param2];
```

  Method name: "`abc:def:`"

Declaration:

```
- (void)abc:(type)param1 def:(type)param2
```

Constructors/ Destructors

- Convention only!
- Constructor

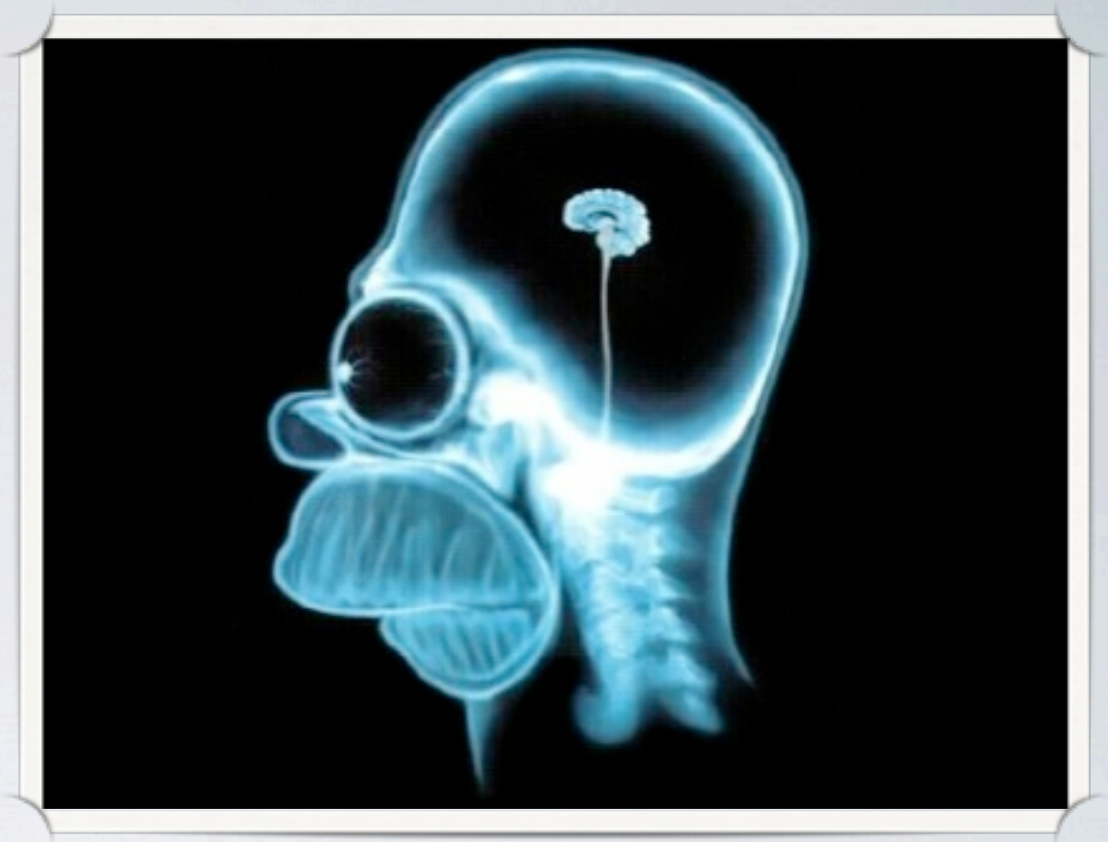
  `[[classname alloc] init]` or `[[classname alloc] initWith...]`

- Destructor

  `- (void)dealloc`

MEMORY MANAGEMENT

Memory Management



- Two modes:
  - Reference Counting
  - Garbage Collection

- Reference Counting keywords
  - “`alloc`”, “`new`”, “`copy`”
  - “`retain`”
  - “`release`”
  - Autorelease Pool

Garbage Collection

- Does the memory management for you
- Not transparent like in Java!
  - Keeps a list of root objects and their dependencies
- strong references, weak references

- `(void)finalize`

`__strong`

IMPORTANT FOUNDATION CLASSES

NSObject

- Base class
- Implements reference counting, object introspection, etc

NSString

```
- (NSUInteger)length

- (unichar)characterAtIndex:(NSUInteger)index

+ (NSString*)stringWithFormat:(NSString*)fmt, …

- (BOOL)isEqualToString:(NSString*)aString
```

- NSMutableString

```
  - (void)appendString:(NSString*)aString
```

Collection Classes

- NSString
  - Unicode string handling
- NSURL
  - URL handling
- NSData
  - Binary data
- NSArray
  - An ordered list of objects
- NSDictionary
  - A key/value mapping
- NSSet
  - An unordered list of objects

Example

```
int main(int argc, const char *argv[]) {
  NSString *myString = @"abc";

  [myString length];

  [myString substringToIndex:2];

  NSLog(@"string: %@", myString);

  return [myString length];
}
```

NSURL

```
+ (NSURL*)URLWithString:(NSString*)URLString

+ (NSURL*)fileURLWithPath:(NSString*)path
```

NSData

```
+ (NSData*)dataWithBytes:(const void *)bytes length:
(NSUInteger)length

- (NSUInteger)length

- (const void *)bytes
```

NSArray

```
+ (NSArray*)arrayWithObjects:…

- (id)objectAtIndex:(NSUInteger)index

- (NSUInteger)count
```

- NSMutableArray

```
  - (void)addObject:(id)obj

  - (void)removeObjectAtIndex:(NSUInteger)index
```

NSDictionary

- "Hashtable"

```
+ (NSDictionary*)dictionaryWithObjectsAndKeys:…
```

```
- (id)objectForKey:(id)key
```

- NSMutableDictionary

```
- (void)setObject:(id)object forKey:(id)key
```

```
- (void)removeObjectForKey:(id)key
```

NSSet

```
+ (NSSet*)setWithObjects:…

- (BOOL)containsObject:(id)obj

- (id)anyObject
```

- NSMutableSet

  ```
  - (void)addObject:(id)obj

  - (void)removeObject:(id)obj
  ```

NSFileHandle

- Reference to a file

```
- (NSData *)readDataToEndOfFile

- (void)writeData:(NSData *)data

+ (id)fileHandleWithStandardInput

+ (id)fileHandleWithStandardOutput
```

NSRunLoop

- Main loop replacement
- Handles timers and sockets

```
- (void)run
```

NSNotification

- Observer pattern

```
[[NSNotificationCenter defaultCenter]
            addObserver:(id)notificationObserver
               selector:(SEL)notificationSelector
                   name:(NSString *)notificationName
                 object:(id)notificationSender]

[[NSNotificationCenter defaultCenter]
    postNotificationName:(NSString *)notificationName
                  object:(id)notificationSender
                userInfo:(NSDictionary *)userInfo]
```

CREATING CLASSES

Interface

```
@interface <class> : <superclass> {
  <ivars>
}

<methods>

@end
```

Implementation

```
@implementation <class>

<methods>

@end
```

Properties

- New in Objective C 2.0 and Leopard
- Automatic accessor generation
- interface: `@property (<options>) <type> <name>;`
- implementation: `@synthesize <name>;`

Protocol

```
@protocol NSCopying

- (id)copyWithZone:(NSZone *)zone;

@end
```

```
@interface NSArray : NSObject <NSCopying,
NSMutableCopying, NSCoding, NSFastEnumeration>

- (NSUInteger)count;
- (id)objectAtIndex:(NSUInteger)index;

@end
```

Categories

- Add methods to an existing class

```
@interface NSData (XMPPStreamAdditions)

- (NSData *)md5Digest;

- (NSData *)sha1Digest;

- (NSString *)hexStringValue;

- (NSString *)base64Encoded;
- (NSData *)base64Decoded;

@end
```

BIG EXAMPLE 1

Task for Thursday

- Write a tool that reads in a file, encrypts it using ROT13 and writes the result back to a different file
  - Hint: Read in the whole file at once and write everything at once
  - Be careful about memory leaks!

References

- Hillegas, Aaron: "Cocoa Programming for Mac OS X" ISBN: 978-0321503619 (Deutsch 978-3826659607)
- http://www.wilshipley.com/blog/WWDC_Student_Talk.pdf
- http://developer.apple.com/mac/