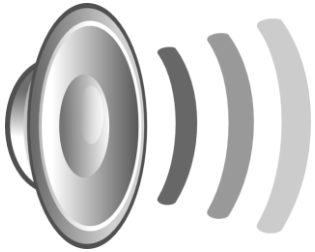
iPhone Programming

CS4347 Sound and Music Computing

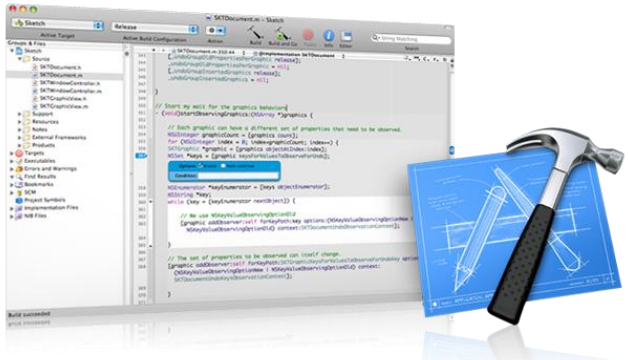
Zhou Yinsheng (yzhou86@comp.nus.edu.sg)

Jan 22nd, 2011



Acknowledgement:
Some materials are borrowed from the course of iPhone Application Development in Stanford University.

Tools to create Apps in iPhone



Xcode

Interface Builder

Mac

Instruments

Simulator/device

Sound and Music Computing

- We use iPhone to make music!
- Mobile Music Group
  - MOGFUN
  - MOGCLASS
  - MOGHEALTH

MOGCLASS

- MOGCLASS DEMO

MOGCLASS

- Do you want to create your own iPhone App or develop novel user interface for MOGCLASS?

You need to know…

- Language: Objective-C and C/C++
- Objective Oriented Programming
- Programming and testing in Xcode
- Digital Signal Processing basics
- Assembly Code (ARM)
- Computer Network
- Etc…

Today we will cover…

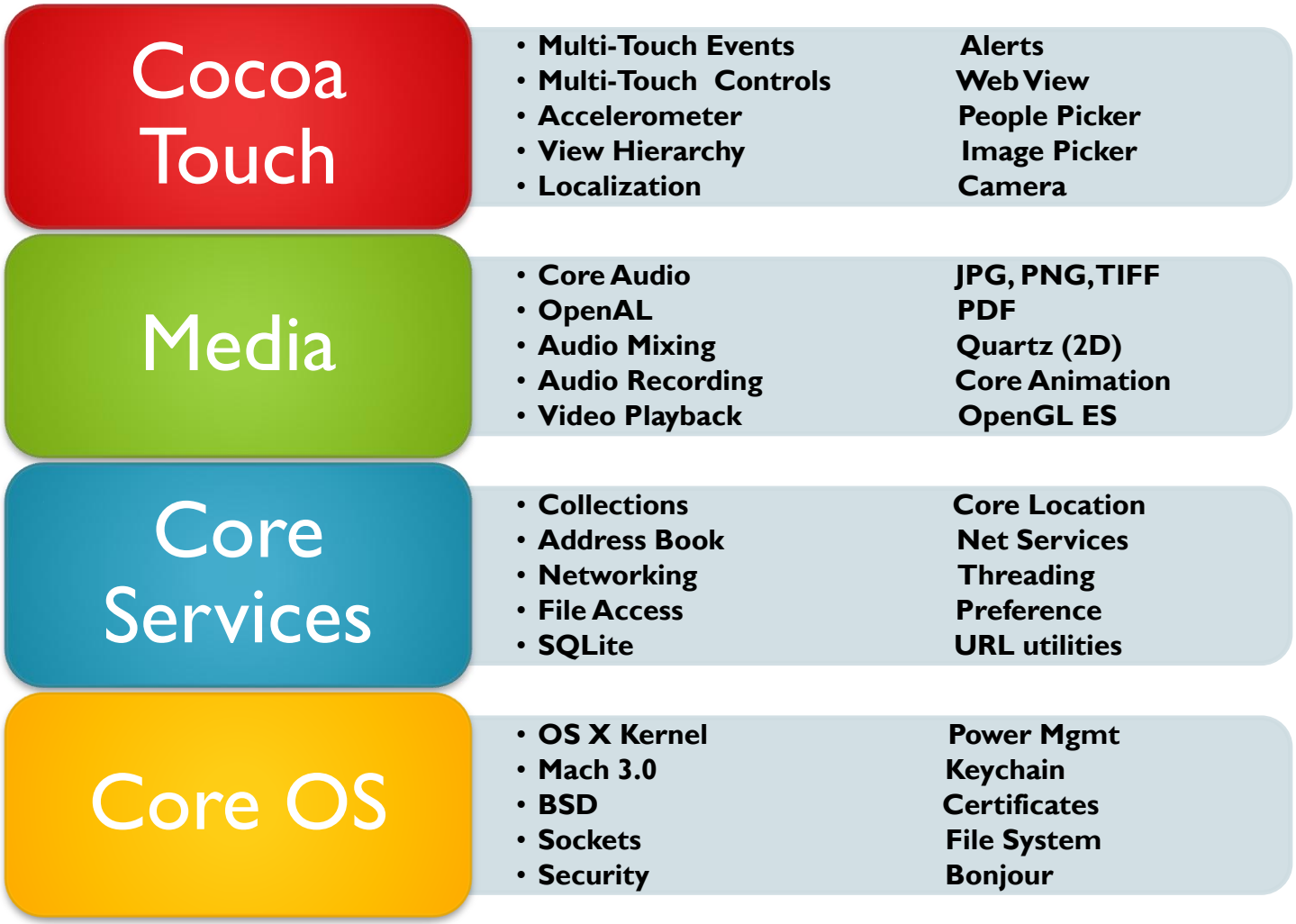
- Objective-C basics
- Building an Application
- Sensor programming (accelerometer, GPS, microphone…)
- Audio Libraries

- There are a lot of materials about iphone online. What we will cover today is more oriented for your final project.

How to learn iPhone Programming

- Recommended Book: None! We'll use Apple documentation
- iOS Developer Center: http://developer.apple.com/devcenter/ios/index.action
  - Download the sample code and learn.
- Write the code by yourself… Trial and error.

iPhone SDK Technology Architecture



Outlines

- Objective-C basics
- Building an Application
- Sensor programming (accelerometer, GPS, microphone…)
- Audio Libraries

OOP Vocabulary

- Class: defines the grouping of data and code, the "type" of an object.
- Instance: a specific allocation of a class.
- Method: a "function" that an object knows how to perform.
- Instance Variable (or "ivar"): a specific piece of data belonging to an object.

OOP Vocabulary

- Encapsulation
  - Keep implementing private and separate from interface
- Polymorphism
  - Different objects, same interface
- Inheritance
  - Hierarchical organization, share code, customize or extend behaviors

Objective-C

- Strict superset of C
  - Mix C with ObjC
  - Or even C++ with ObjC (usually referred to as ObjC++)
- A very simple language, but some new syntax
- Single inheritance, classes inherit from one and only one superclass
- Protocols define behavior that cross classes
- Dynamic runtime
- Loosely typed, if you'd like

Class and Instance Methods

- Instance respond to instance methods

  -(id)init;

  -(float)height;

  -(void)walk;

- Classes respond to class methods

  +(id)alloc;

  +(id)person;

  +(Person*)sharedPerson;

Message Syntax

```
[receiver message];
[receiver message:argument];
[receiver message:arg1 andArg: arg2]
```

Terminology

- Message expression

  [receiver method:argument]

- Message

  [receiver method:argument]

- Selector

  [receiver method:argument]

- Method

  The code selected by a message.

Dot Syntax

- Objective-C 2.0 introduced dot syntax
- Convenient shorthand for invoking accessor methods

```
float height = [person height];
float height = person.height;

[person setHeight: newHeight];
person.height = newHeight;
```

- Follows the dots…

```
[[person child] setHeight:newHeight];
//exact the same as
person.child.height = newHeight;
```

Dynamic and static typing

- Dynamically-typed object

  id anObject

  - just id
  - Not id * (unless you really, really mean it…)
- Statically-typed object

  Person *anObject
- Objective-C provides compile-time, not runtime, type checking
- Objective-C always uses dynamic binding

Selectors identify methods by name

- A selector has type SEL

```
SEL action = [button action];
[button setAction: @selector(start:)];
```

- Conceptually similar to function pointer
- Selectors include the name and all colons, for example:

```
-(void) setName:(NSString*)name age:(int)age;
```

would have a selector:

```
SEL sel = @selector(setName:age:);
```

Working with selectors

- You can determine if an object responds to a given selector

```
id obj;
SEL sel = @selector(start:)
if ([obj respondsToSelector:sel]){
    [obj performSelector: sel withObject:self];
}
```

- This sort of introspection and dynamic messaging underlies many Cocoa design patterns

```
-(void)setTarget:(id)target;
-(void)setAction:(SEL)action;
```

Working with Classes

- You can ask an object about its class

```
Class myClass = [myObject class];
NSLog(@"My class is %@", [myObject className]);
```

- Testing for general class membership (subclasses included):

```
if ([myObject isKindOfClass:[UIControl class]]) {
    // something
}
```

- Testing for specific class membership (subclasses excluded):

```
if ([myObject isMemberOfClass:[NSString class]]) {
    // something string specific
}
```

Working with Objects

- Identity v.s. Equality
- Identity—testing equality of the pointer values

```
if (object1 == object2) {
    NSLog(@"Same exact object instance");
}
```

- Equality—testing object attributes

```
if ([object1 isEqual: object2]) {
    NSLog(@"Logically equivalent, but may
        be different object instances");
}
```

-description

- NSObject implements -description

```
- (NSString *)description;
```

- Objects represented in format strings using %@
- When an object appears in a format string, it is asked for its description

```
[NSString stringWithFormat: @"The answer is: %@",
    myObject];
```

- You can log an object's description with:

```
NSLog([anObject description]);
```

- Your custom subclasses can override description to return more specific information

Foundation Framework

- Foundation Classes
  - NSObject
    - String
      - NSString / NSMutableString
    - Collection
      - NSArray / NSMutableArray
      - NSDictionary / NSMutableDictionary
      - NSSet / NSMutableSet
      - NSNumber
    - Others
      - NSData / NSMutableData
      - NSDate / NSCalendarDate

More OOP Info

- Tons of books and articles on OOP
- Objective-C 2.0 Programming Language

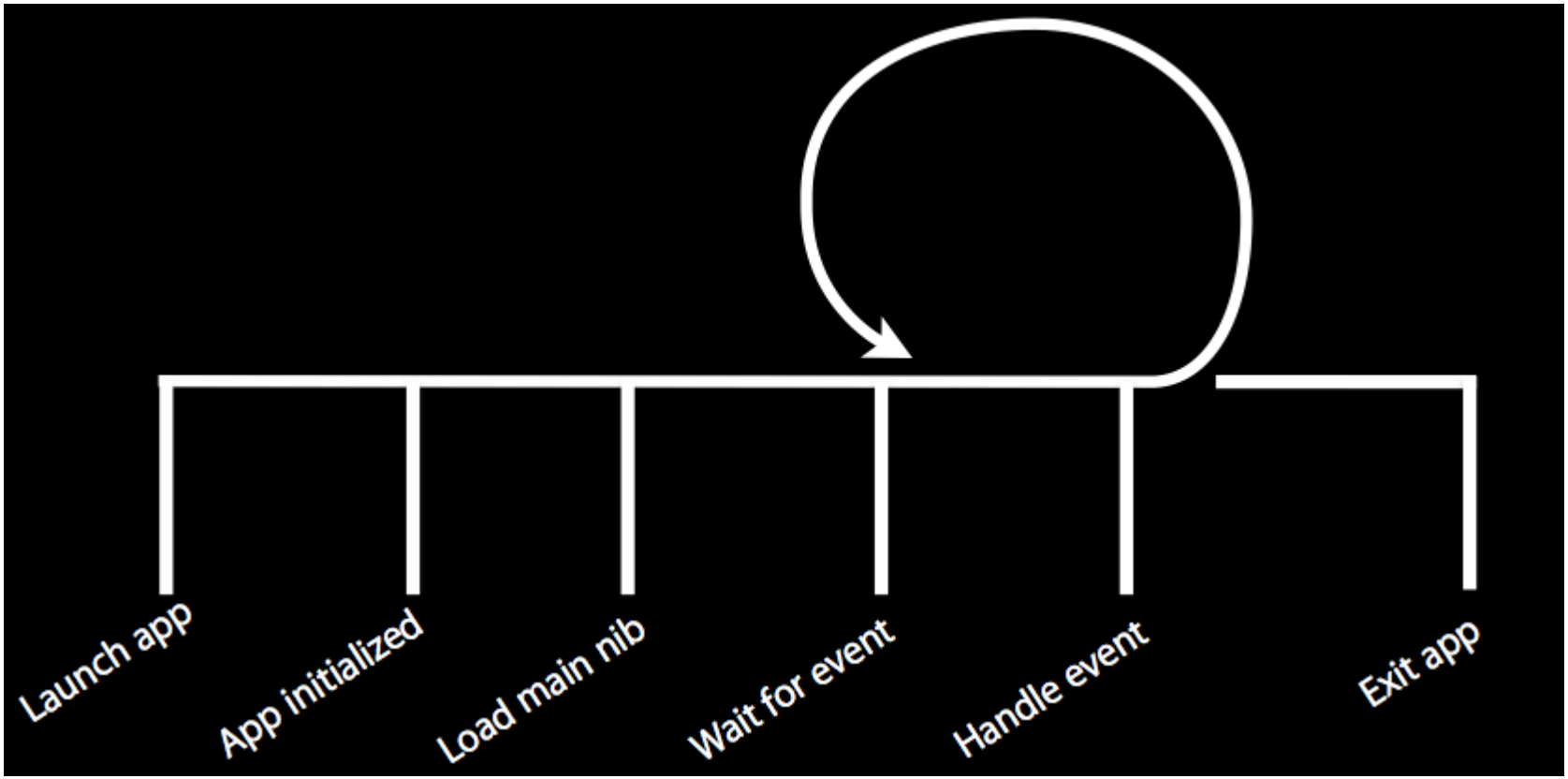
Outlines

- Objective-C basics
- Building an Application
- Sensor programming (accelerometer, GPS, microphone…)
- Audio Libraries

Anatomy of an Application

- Compiled code
  - Your code
  - Frameworks
- Nib files
  - UI elements and other objects
  - Details about object relations
- Resources (images, sounds, strings, etc)
- Info.plist file (application configuration)

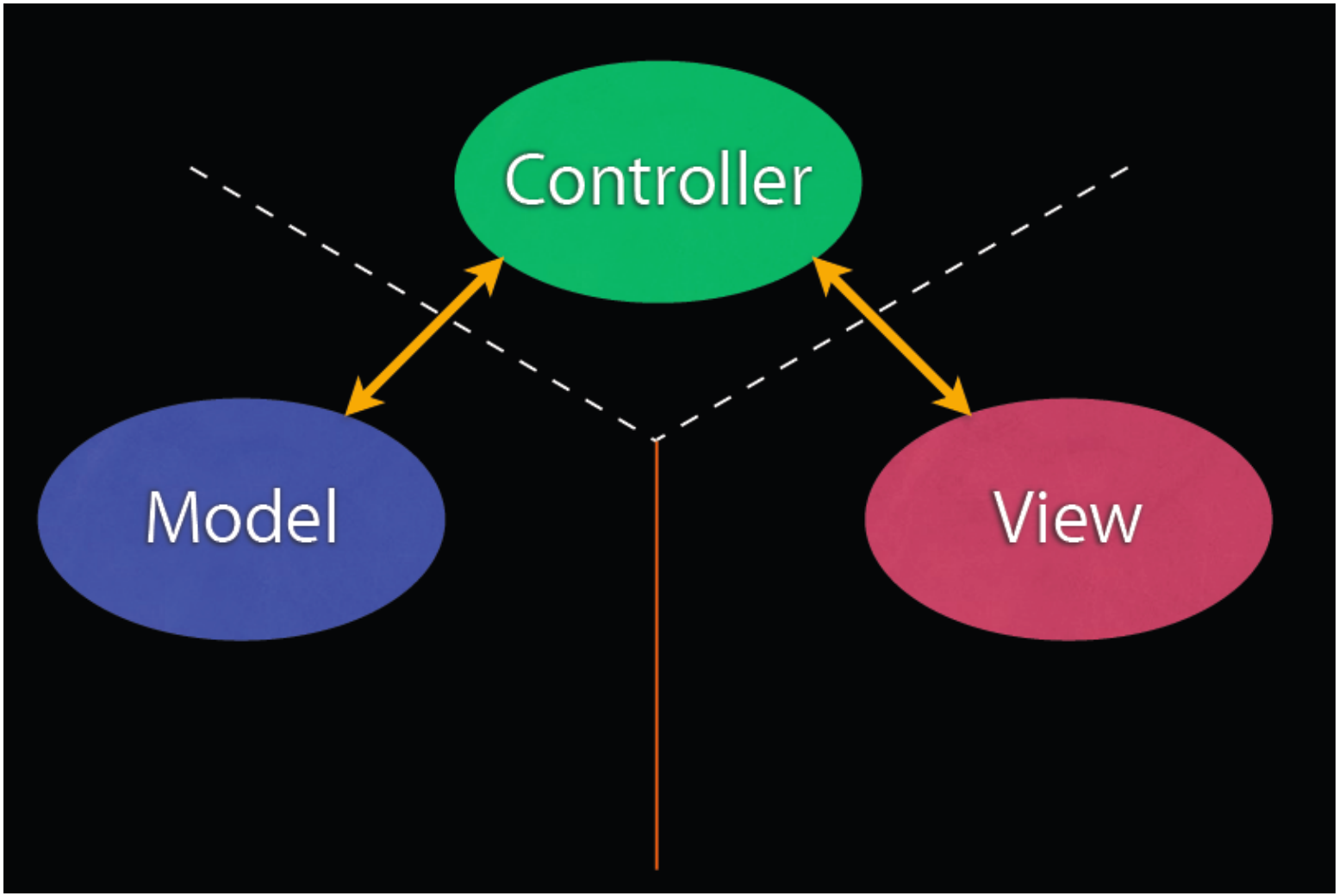
App Lifecycle



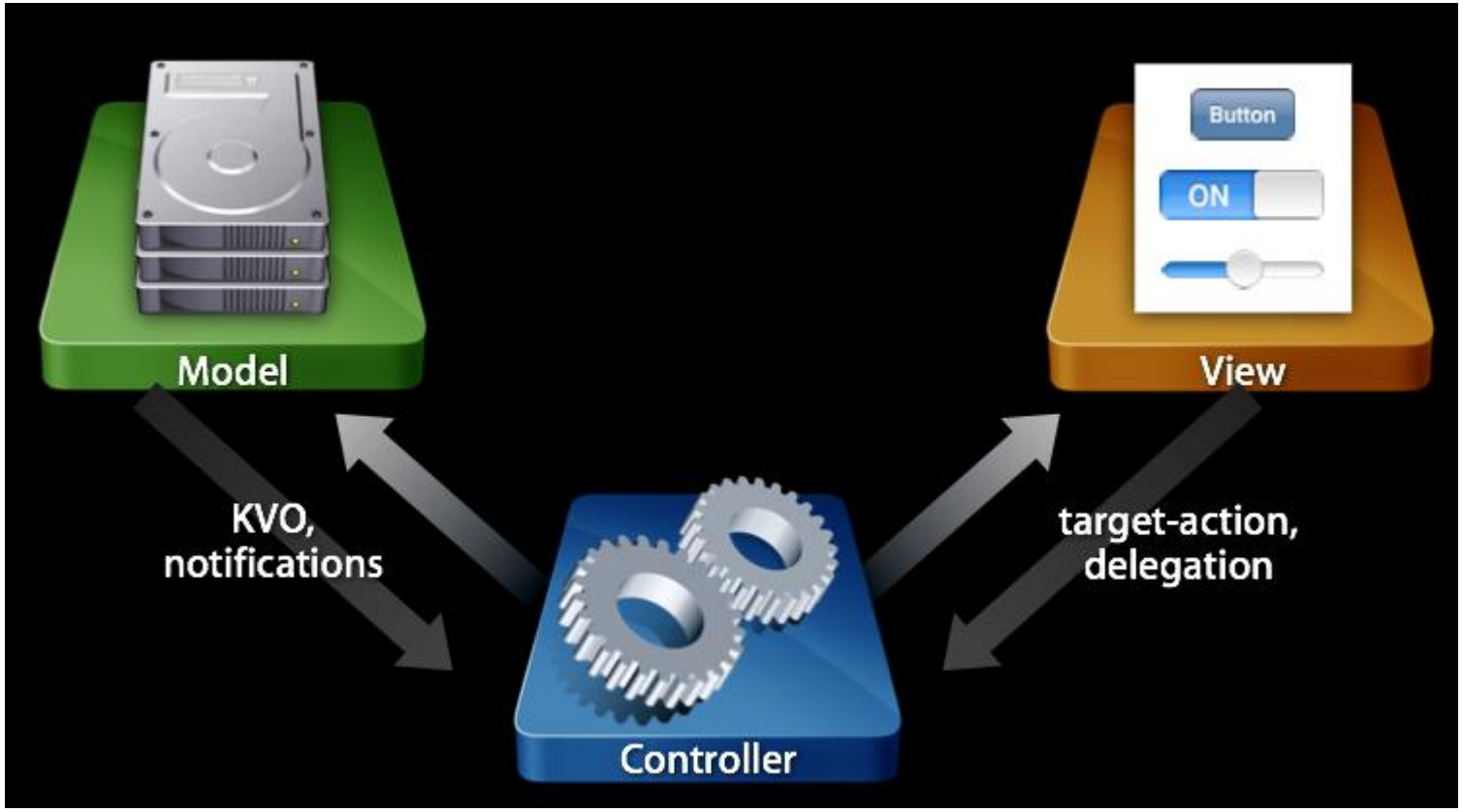
App Lifecycle

- Main function
- UIApplicationMain which Info.plist to figure out what nib to load.
- MainWindow.xib contains the connections for our application.
- AppDelegate
- ViewController.xib
- View – handle UI events

Model-View-Controller



Communication and MVC



Model

- Manages the app data and state
- Note concerned with UI or presentation
- Often persists somewhere
- Same model should be reusable, unchanged in different interfaces

View

- Present the Model to the user in an appropriate interface
- Allows user to manipulate data
- Does not store any data
  - (except to cache state)
- Easily reusable & configurable to display different data

Controller

- Intermediary between Model & View
- Updates the view when the model changes
- Updates the model when the user manipulates the view
- Typically where the app logic lives

Outlines

- Objective-C basics
- Building an Application
- Sensor programming (accelerometer, GPS, microphone, …)
- Audio Libraries

Accelerometer

- What are the accelerometers?
  - Measure changes in force
- What are the uses?
- Physical Orientation vs. Interface Orientation
  - Ex: Photos & Safari

Orientation-Related Changes

- Getting the physical orientation
  - UIDevice class
    - UIDeviceOrientationDidChangeNotification
- Getting the interface orientation
  - UIApplication class
    - statusBarOrientation property
  - UIViewController class
    - interfaceOrientation property

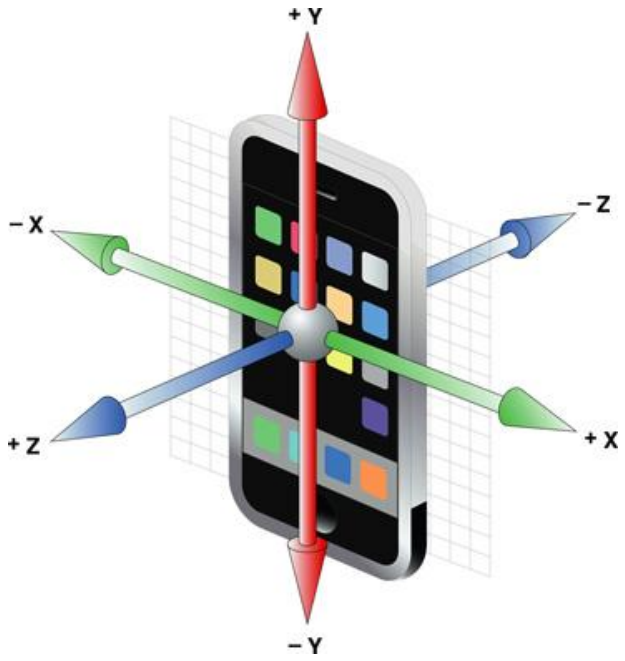
Shake Undo!

- UIEvent type
  - @property(readonly) UIEventType type;
  - @property(readonly) UIEventSubtype subtype;
  - UIEventTypeMotion
  - UIEventSubtypeMotionShake

Getting raw Accelerometer Data

- 3-axis data
- Configurable update frequency (10-100Hz)
- Sample Code (AccelerometerGraph)

- Class
  - UIAccelerometer
  - UIAcceleration
- Protocol
  - UIAccelerometerDelegate



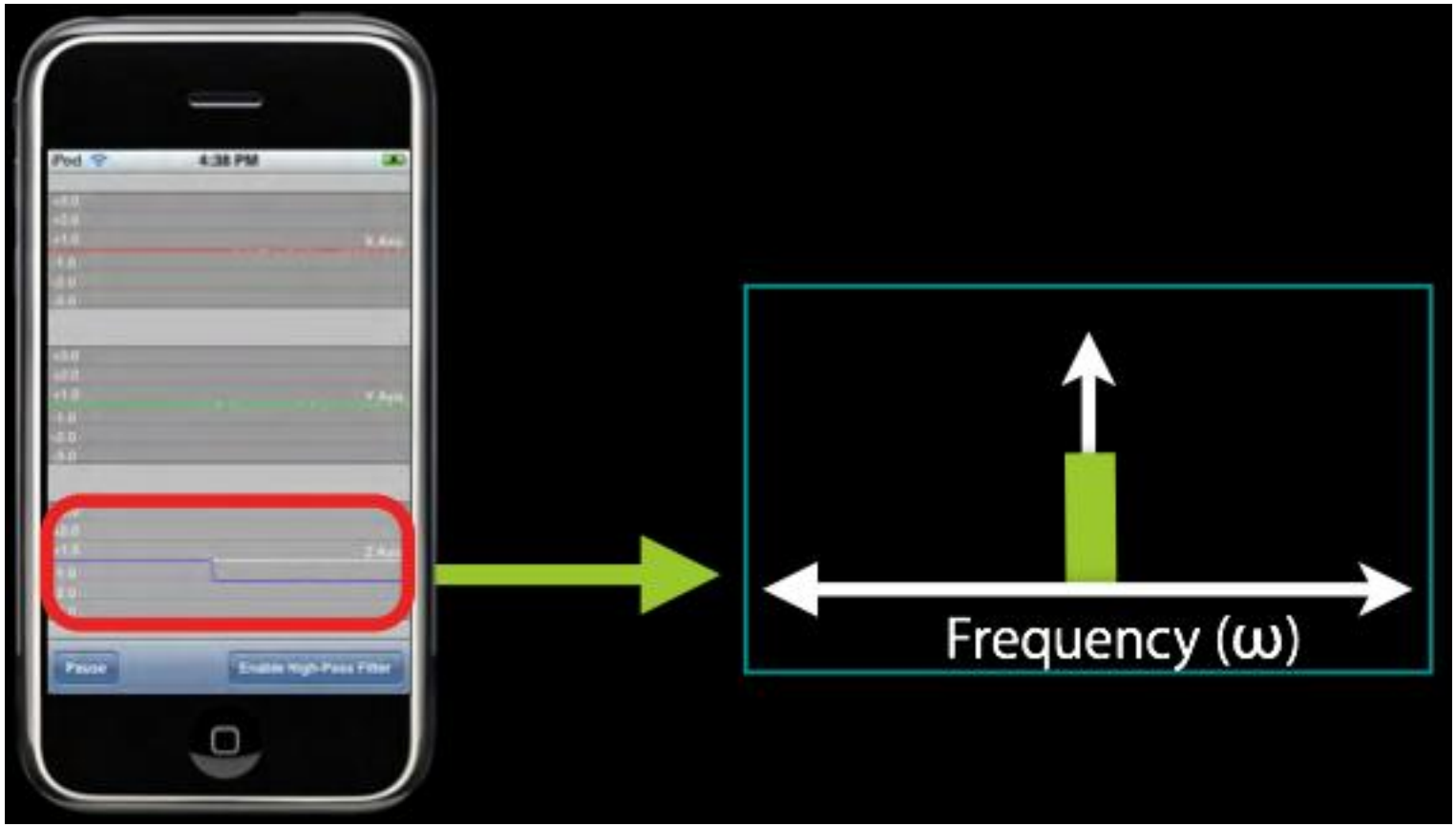
Getting raw Accelerometer Data

```
-(void)initAccelerometer{
    [[UIAccelerometer sharedAccelerometer] setUpdateInterval: (1.0 / 100)];
    [[UIAccelerometer sharedAccelerometer] setDelegate: self];
}

-(void)accelerometer: (UIAccelerometer*)accelerometer didAccelerate:
    (UIAcceleration*) acceleration {
    double x, y, z;
    x = acceleration.x;
    y = acceleration.y;
    z = acceleration.z;
    //process the data…
}

-(void)disconnectAccelerometer{
    [[UIAccelerometer sharedAccelerometer] setDelegate: nil];
}
```

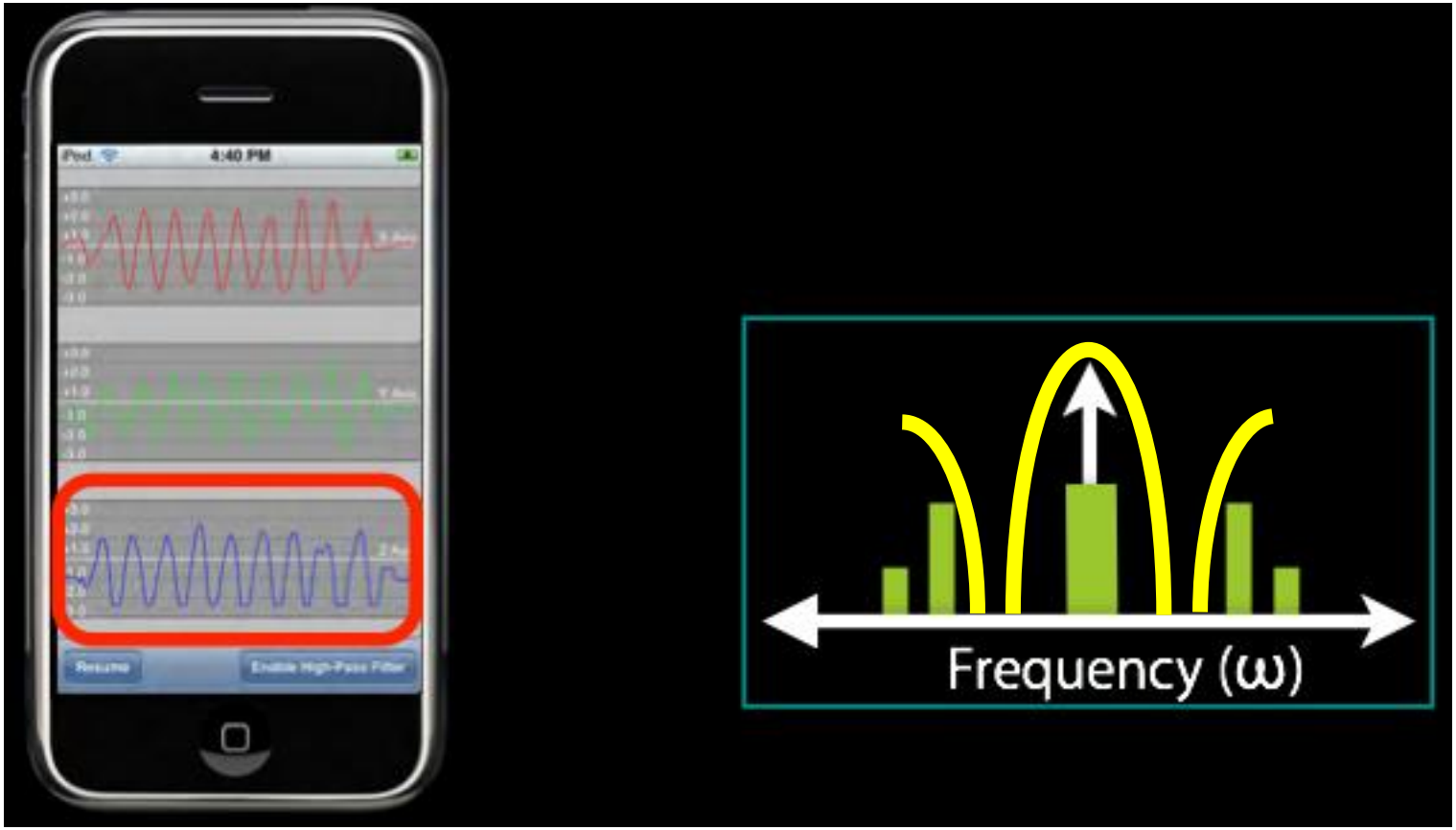
Filtering Accelerometer Data

- Low-pass filter
  - Isolates constant acceleration
  - Used to find the device orientation
- High-pass filter
  - Shows instantaneous movement only
  - Used to identify user-initiated movement

Filtering Accelerometer Data



Filtering Accelerometer Data



Applying Filters

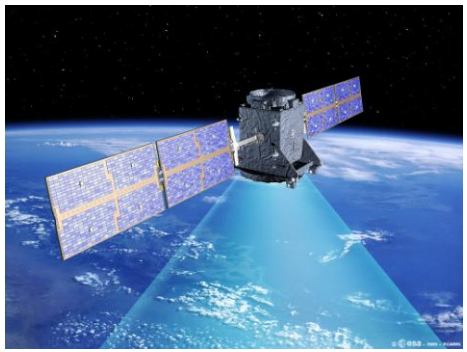
- Simple low-pass filter example

```
#define FILTERFACTOR 0.1
Value = (newAcceleration * FILTERFACTOR) + (previousValue *
(1.0 – FILTERFACTOR));
previousValue = value;
```

- Simple high-pass filter example

```
lowpassValue = (newAcceleration * FILTERFACTOR) +
(previousValue * (1.0 – FILTERFACTOR));
previousLowPassValue = lowPassValue;
highPassValue = newAcceleration – lowPassValue;
```

GPS



- Classes
  - CLLocationManager
  - CLLocation
- Protocol
  - CLLocationManagerDelegate

Getting a Location

- Starting the location service

```
-(void)initLocationManager{
        CLLocationManager*  locManager = [[CLLocationManager alloc] init];
        locManager.delegate = self;
        [locManager startUpdatingLocation];
}
```

- Using the event data

```
-(void)locationManager: (CLLocationManager*)manager didUpdateToLocation:
(CLLocation*)newLocation fromLocation: (CLLocation*)oldLocation{
        NSTimerInterval howRecent = [newLocation.timestamp timeIntervalSinceNow];
        if(howRecent < -10) return;

        if(newLocation.horizontalAccuracy > 100) return;

        //Use the coordinate data.
        double lat = newLocation.coordinate.latitude;
        double lon = newLocation.coordinate.longitude;
}
```

Getting a Heading

- Geographic North
  - CLLocationDirection trueHeading
- Magnetic North
  - CLLocationDirection magneticHeading

```
-(void)locationManager: (CLLocationManager *)manager
didUpdateHeading:(CLHeading*)newHeading{
    //Use the coordinate data.
    CLLocationDirection heading = newHeading.trueHeading;
    CLLocationDirection magnetic = newHeading.magneticHeading;
}
```

Microphone

- We will cover it in the audio library…

Outlines

- Objective-C basics
- Building an Application
- Sensor programming (accelerometer, GPS, microphone…)
- Audio Libraries

Audio Libraries

- System Sound API – short sounds
- AVAudioPlayer – ObjC, simple API
- Audio Session - Audio Toolbox
- Audio Queue - Audio Toolbox
- Audio Units
- OpenAL
- MediaPlayer Framework

AVAudioPlayer

- Play longer sounds (> 5 seconds)
- Locally stored files or in-memory (no network streaming)
- Can loop, seek, play, pause
- Provides metering
- Play multiple sounds simultaneously
- Cocoa-style API
  - Initialize with file URL or data
  - Allows for delegate
- Supports many more formats
  - Everything the AudioFile API supports
- Sample Code: avTouch

AVAudioPlayer

- Create from file URL or data

```
AVAudioPlayer *player;
NSString *path = [[NSBundle mainBundle] pathForResource:… ofType:…];
NSURL *url = [NSURL fileURLWithPath:path];
player = [[AVAudioPlayer allow] initWithContentsOfURL:url];
[player prepareToPlay];
```

- Simple methods for starting/stopping

```
If(!player.playing){
    [player play];
}else{
    [player pause];
}
```

Audio Sessions

- Handles audio behavior at the application, inter-application, and device levels
  - Ring/Silent switch?
  - iPod audio continue?
  - Headset plug / unplug?
  - Phone call coming?
- Sample Code: avTouch & aurioTouch

Audio Sessions

- Six audio session categories
  - Ambient
  - Solo Ambient (Default session)
  - Media playback
  - Recording
  - Play and record
  - Offline audio processing

Audio Queue

- Audio File Stream Services & Audio Queue Services
- Supports wider variety of formats
- Finer grained control over playback
  - Streaming audio over network
  - Cf: AVAudioPlayer(local)
- Allows queueing of consecutive buffers for seamless playback
  - Callback functions for reusing buffers
- Sample code: SpeakHere

Audio Units

- For serious audio processing
- Graph-based audio
  - Rate or format conversion
  - Real time input/output for recording & playback
  - Mixing multiple streams
  - VoIP (Voice over Internet Protocol).
- Very, very powerful.
- Sample code: aurioTouch

Audio Units

- **Lowest** programming layer in iOS audio stack
  - Real-time playback of synthesized sounds
  - Low-latency I/O
  - Specific audio unit features
- Otherwise, look first at the Media Player, AV Foundation, OpenAL, or Audio Toolbox!

Audio Units

- Ex: Karaoke app in iPhone
  - real-time input from mic
  - Real-time output to speaker
  - Audio Unit provides excellent responsiveness
  - Audio Unit controls audio flow to do pitch tracking, voice enhancement, iPod equalization, and etc.

OpenAL

- High level, cross-platform API for 3D audio mixing
  - Great for games
  - Mimics OpenGL conventions
- Models audio in 3D space
  - Buffers: Container for Audio
  - Sources: 3D point emitting Audio
  - Listener: Position where Sources are heard
- Sample code: oalTouch
- More information: http://www.openal.org/

MediaPlayer Framework

- Tell iPod app to play music
- Access to entire music library
  - For playback, not processing

- Easy access through MPMediaPickerController
- Deeper access through Query APIs

- Sample code: AddMusic

Others

- Accelerate Framework
  - C APIs for vector and matrix math, digital signal processing large number handling, and image processing
  - vDSP programming guide

- Bonjour and NSStream
- The Synthesis ToolKit in C++ (STK) and OSC

More Info for Beginners

- iPhone Application Programming (Standford University - iTunes University)
- Dan Pilone & Tracey Pilone. Head First iPhone Development.

- Thank you and good luck to your final projects!