



# Développement d'applications mobiles iOS

# Plan de formation

## Séance 1 (4h)

**Introduction à iOS, Objective-C / Swift et aux outils de développement**

## Séance 2 (4h)

**Capteurs mobiles sur iPhone & iPad**

**Kinan Arnaout**

**Ingénieur R&D Mobile chez Intellicore**



# Intellicore

[www.intellicore.net](http://www.intellicore.net)

[www.intellicore.tv](http://www.intellicore.tv)



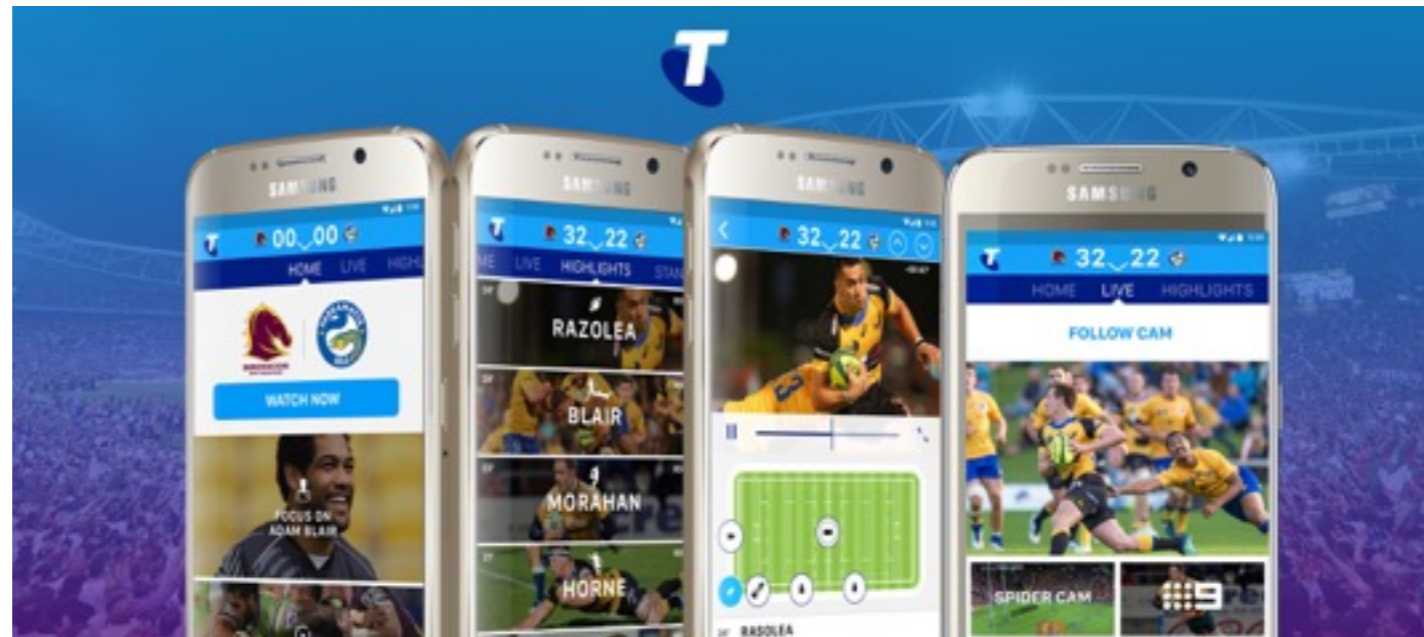


# Intellicore

## Playrz



## Applications stadium







**iOS**



# iPhone



1



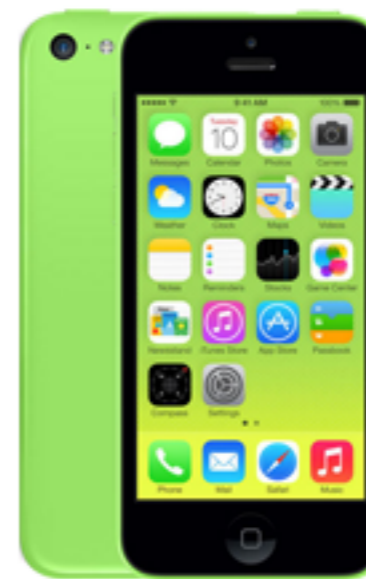
3G  
3GS



4  
4S



5



5C



5S

# Version actuelle

13M d'iPhone 6S et 6S  
+ en 3 jours



6S



6S+

# iPad



2 modèles pour 2 tailles, dotés ou non de la 3G



# Ipad Pro



# Segmentation du parc

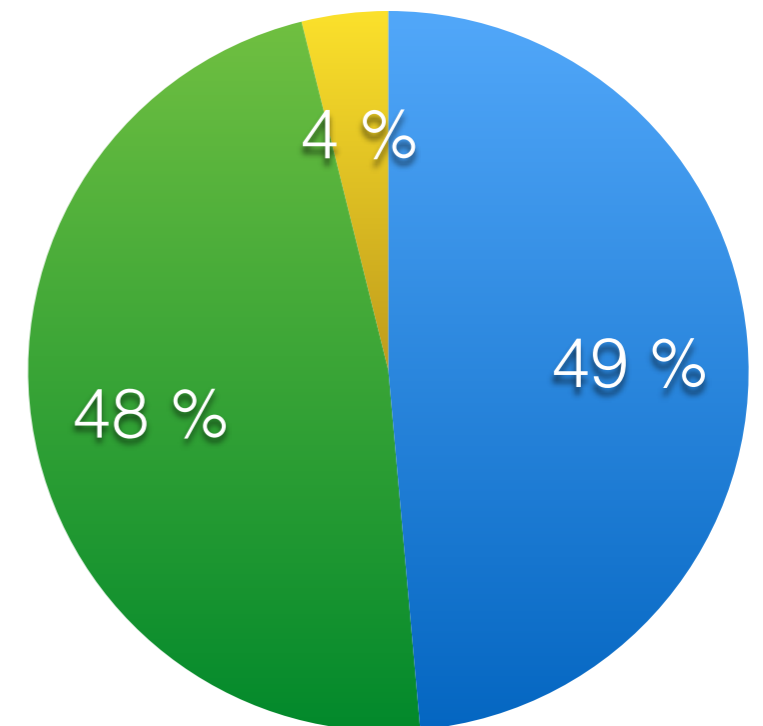
Contrairement à d'autres OS mobiles, le parc n'est pas très fragmenté

Les migrations du parc sont relativement rapides

Tout développement se doit de cibler les 2 dernières versions majeures de l'OS

● iOS 8 ● iOS 7 ● Older

iOS 8 adoption  
17/10/ 2014

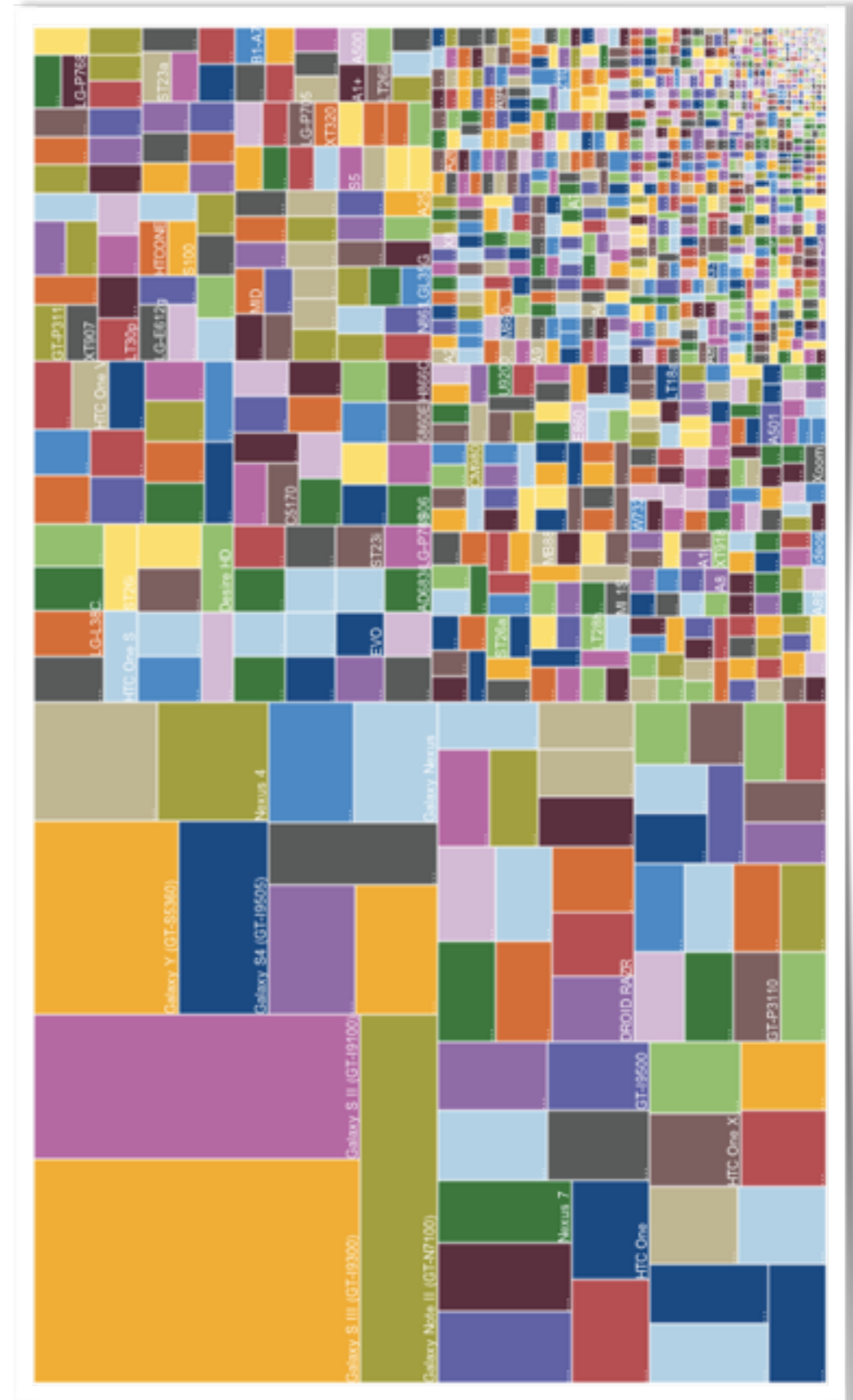
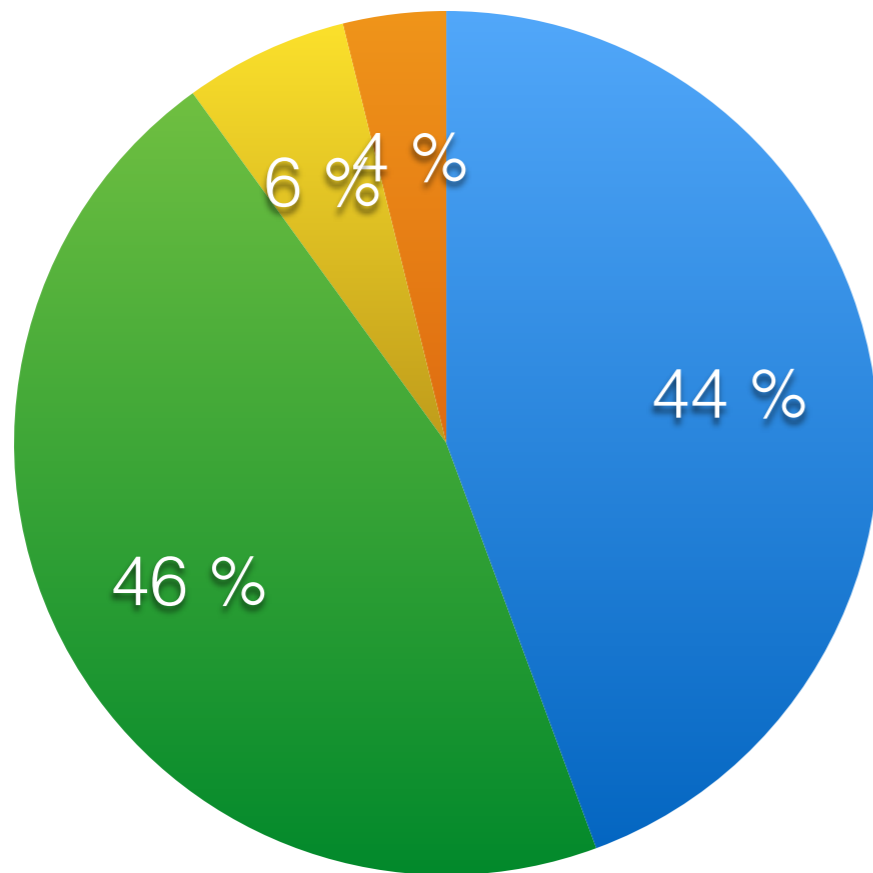




# Versus

- Kit Kat
- Jelly Bean
- Ice Cream Sandwich
- Older

Android OS versions



# App Store

> 75.000.000.000  
téléchargements

± 1.300.000 applications actives

± 210.000 développeurs

66% des apps sont gratuites



```
{
    (NSManagedObjectContext *)managedObjectContext
    if (_managedObjectContext != nil) {
        return _managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
    if (coordinator != nil) {
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithPersistentStoreCoordinator:coordinator];
    }
    return _managedObjectContext;
}

// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectContext *)managedObjectContextModel
{
    if (_managedObjectContextModel != nil) {
        return _managedObjectContextModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"Lab" withExtension:@"momd"];
    _managedObjectContextModel = [[NSManagedObjectContext alloc] initWithContentsOfURL:modelURL];
    return _managedObjectContextModel;
}

// Returns the persistent store coordinator for the application.
// If the coordinator doesn't already exist, it is created and the application's store added to it.
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
    if (_persistentStoreCoordinator != nil) {
        return _persistentStoreCoordinator;
    }

    NSURL *storeURL = [[self applicationDocumentsDirectory] URLByAppendingPathComponent:@"Lab.sqlite"];
    NSError *error = nil;
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithURL:storeURL error:&error];
    if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configurationName:nil URL:storeURL options:nil error:&error]) {
        NSLog(@"Error adding persistent store: %@", error);
    }
}

// Returns the application's main window.
- (UIWindow *)window
{
    UIWindow *window = [[UIWindow alloc] initWithFrame:[UIScreen mainScreen].bounds];
    UINavigationController *navController = [[UINavigationController alloc] initWithRootViewController:self];
    window.rootViewController = navController;
    [window makeKeyAndVisible];
    return window;
}

- (void) viewDidLoad
{
    [self loadInitialData];
}

- (void) loadInitialData
{
    NSLog(@"Loading initial data...");
}

- (void) didReceiveMemoryWarning
{
    NSLog(@"Memory warning received. Saving data...");
}

- (void) saveData
{
    NSLog(@"Saving data...");
}

- (void) deleteData
{
    NSLog(@"Deleting data...");
}

- (void) reloadData
{
    NSLog(@"Reloading data...");
}

- (void) dealloc
{
    NSLog(@" dealloc");
}

@end
```

# Objective-C

2013-04-21 19:03:25.594 Lab[17985:c07] global  
2013-04-21 19:03:25.596 Lab[17985:c07] Appli  
cont view controller at the end of

# Avez-vous déjà vu ?

```
// Extrait de la librairie open source AFNetworking

- (AFHTTPRequestOperation *)HTTPRequestOperationWithRequest:(NSURLRequest *)urlRequest
    success:(void (^)(AFHTTPRequestOperation
*operation, id responseObject))success
    failure:(void (^)(AFHTTPRequestOperation
*operation, NSError *error))failure
{
    AFHTTPRequestOperation *operation = nil;

    for (NSString *className in self.registeredHTTPOperationClassNames) {
        Class operationClass = NSClassFromString(className);
        if (operationClass && [operationClass canProcessRequest:urlRequest]) {
            operation = [(AFHTTPRequestOperation *)[operationClass alloc] initWithRequest:urlRequest];
            break;
        }
    }

    if (!operation) {
        operation = [[AFHTTPRequestOperation alloc] initWithRequest:urlRequest];
    }

    [operation setCompletionBlockWithSuccess:success failure:failure];

    operation.credential = self.defaultCredential;
#ifdef _AFNETWORKING_PIN_SSL_CERTIFICATES_
    operation.SSLPinningMode = self.defaultSSLPinningMode;
#endif
    operation.allowsInvalidSSLCertificate = self.allowsInvalidSSLCertificate;

    return operation;
}
```

# Objective-C

Wrapper du langage C

Langage compilé

Dynamique

Envoi de messages à la Smalltalk

Programmation Orientée Objet

Fortement typé

Très verbeux

# Typage

## PHP

(faiblement typé)

```
$i = 123;  
$s = "Hello world!";
```

## Java

(fortement typé)

```
int i = 123;  
String s = "Hello world!";
```

## Objective-C

(fortement typé, pointeurs)

```
NSInteger i = 123;  
NSString *s = @"Hello world!";
```

# Header & implementation



```
@interface Speaker : NSObject

// Properties
@property (nonatomic, strong) NSString *firstname;
@property (nonatomic, strong) NSString *lastname;

// Methods declaration
- (void)sayHello;

@end
```

Speaker.h

API publique de la classe  
Éléments accessibles  
depuis l'extérieur



```
#import "Speaker.h"

@implementation Speaker

- (void)sayHello
{
    NSLog(@"Hello, my name is %@ %@", _firstname,
        _lastname);
}

@end
```

Speaker.m

Implémentation des  
méthodes déclarées dans  
l'interface



# Properties

```
@property (nonatomic, strong) NSString *lastname;
```

Déclaration des attributs d'une classe

Génération automatique des getters

```
- (void)whois  
{  
    NSLog(@"My firstname is %@", self.firstname); // Getter via self.  
    NSLog(@"My lastname is %@", _lastname); // Getter via _  
}
```

Génération automatique des setters

```
- (void)setup  
{  
    self.firstname = @"Cyril"; // Setter via self.  
    _lastname = @"Chandelier"; // Setter via _  
}
```



```
[NSNumber numberWithInt:12];
```

```
// ou
```

```
@12;
```

```
[NSNumber numberWithBool:YES];
```

```
// ou
```

```
@YES;
```

```
NSArray *myArray = [NSArray arrayWithObjects:obj1, obj2, obj3, nil];
```

```
// ou
```

```
NSArray *myArray = @[ obj1, obj2, obj3 ];
```

```
id obj = [myArray objectAtIndex:0];
```

```
// ou
```

```
id obj = myArray[0];
```

```
NSDictionary *dic = [NSDictionary dictionaryWithObjectsAndKeys:lastname, @"lastname", firstname, @"firstname",  
[NSDate date], @"registrationDate", nil];
```

```
// ou
```

```
NSDictionary *dic = @{  
    @"lastname": lastname,  
    @"firstname": firstname,  
    @"registrationDate": [NSDate date]  
};
```

```
id obj = [dic objectForKey:@"lastname"];
```

```
// ou
```

```
id obj = dic[@"lastname"];
```

# Messages

Envoi de commandes aux objets (équivalent aux appel de méthodes)

Messages analysés au runtime par l'objet

## Objective-C

```
// Speaker objects
Speaker *speaker1 = [[Speaker alloc] init];
Speaker *speaker2 = [[Speaker alloc] init];

// Simple, no argument
[speaker1 sayHello];

// Single argument
[speaker1 say:@"Hello"];

// Multiple argument
[speaker1 say:@"Hello" to:@"students"];

// Nested messages
[speaker1 say:@"Hello" to:[speaker2 fullName]];
```

## Java

```
// Speaker objects
Speaker speaker1 = new Speaker();
Speaker speaker2 = new Speaker();

// Simple, no argument
speaker1.sayHello();

// Single argument
speaker1.say("Hello");

// Multiple argument
speaker1.say("Hello", "students");

// Nested messages
speaker1.say("Hello", speaker2.getFullName());
```

# Protocols

Déclaration de méthodes à implémenter

Très utilisé pour le design pattern “Délégué”

```
@class Speaker;  
  
@protocol SpeakerDelegate <NSObject>  
  
@required  
- (void)speaker:(Speaker *)aSpeaker said:(NSString *)sentence;  
  
@optional  
- (void)speaker:(Speaker *)aSpeaker ask:(NSString *)question;  
  
@end
```

```
@interface Speaker : NSObject <SpeakerDelegate>  
  
@end
```

# Categories

## Ajout de fonctionnalités à une classe

```
// Speaker+Utils.h
#import "Speaker.h"

@interface Speaker (Utils)

- (NSString *)fullname;

@end
```

```
// Speaker+Utils.m

#import "Speaker+Utils.h"

@implementation Speaker (Utils)

- (NSString *)fullname
{
    return [NSString stringWithFormat:@"%@" "%@", self.firstname, self.lastname];
}

@end
```

# Blocks

Bout de code exécutable

Paramètres et types de retours

Très utilisés dans les animations

```
// Method with a block as parameter
- (void)doSomethingWithBlock:(void (^)(void))aBlock
{
    aBlock();
}

- (void)iUseBlocks
{
    // Block declaration
    void (^aBlock)(void) = ^{
        NSLog(@"Hi, i'm a block");
    };

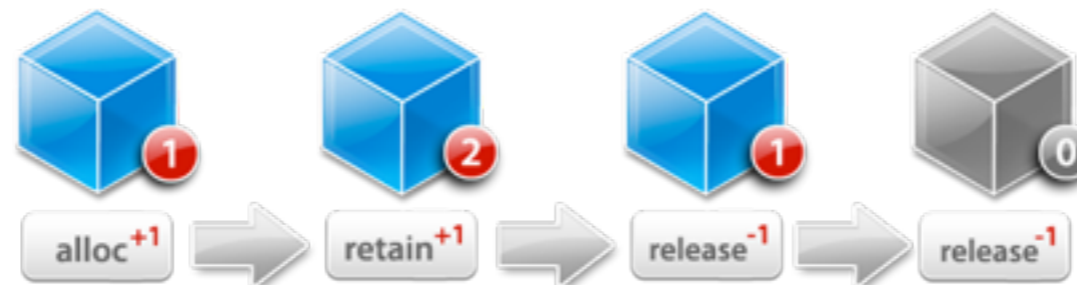
    // Execute it now
    aBlock();

    // Give it to a method
    [self doSomethingWithBlock:aBlock];
}
```

## Avant

Gestion de la mémoire de type Reference Counting

- retain
- release
- autorelease



## Maintenant

ARC (Automatic Reference Counting)

≠ Garbage Collector

ional patterns

Protocols and extensions on structs

Pattern mat

se syntax

Closures

Generics

Fast iter

e collections

Optional

tor overloading

Object orient

spaces

Tuples

Type infe



Read-Eval-Print-Loop (

mutability syntax

Multiple return types

Com

ctive playground

**Swift**

# “Objective-C without the C”

Introduit durant la WWDC 2014

Présenté comme le successeur de l'Objective-C

Plus rapide (93x) d'après des benchmarks réalisés sur des algorithmes complexes de tri et d'encryption



# Points communs

Cocoa et Cocoa Touch

Compilateur LLVM

Automatic Reference Counting

Même runtime qu'Objective-C

# Principales différences avec Objective-C

Plus besoin de ; à la fin de chaque instruction

Plus de header (fichiers .h)

Les énumérations peuvent avoir des données associées

(Re)-définition d'opérateurs

Closures

Namespaces

# Designed for safety

Les pointeurs ne sont plus exposés

Chaque case d'un switch est terminal

Les variables et les constantes sont toujours initialisées

Le typage est indispensable et contraignant pour faire attention aux algorithmes développés

# Optionals

*“there is a value, and it equals x”*

**OR**

*“there isn’t a value at all”*

Équivalent de l’utilisation de *nil* en Objective-C

Symbole “?” pour déclarer une variable ou un retour comme optionnel

Symbole “!” pour forcer un retour différent de *nil*

# Optionals

```
func cellDidEditTaskContent(cell: TodoCell, newContent: String)
{
    // Retrieve index path
    let indexPath = self.tableView.indexPathForCell(cell)
    if (indexPath != nil)
    {
        // Update related task and save
        var task = self.frc!.objectAtIndex(indexPath!) as Task

        // Update task
        TaskController.sharedInstance.updateTask(task, content: newContent)
    }
}
```

# Constantes et variables

```
// Constant > immutable  
let aConstant = 10  
  
// Variable > mutable  
var aVariable = 10
```

Swift met un point d'honneur à déclarer ses variables avec la mutabilité qui y convient le plus

Il est conseillé de toujours déclarer ses variables en tant que constante, puis de mettre à jour le type si nécessaire uniquement

# Type Safety & Type Inference

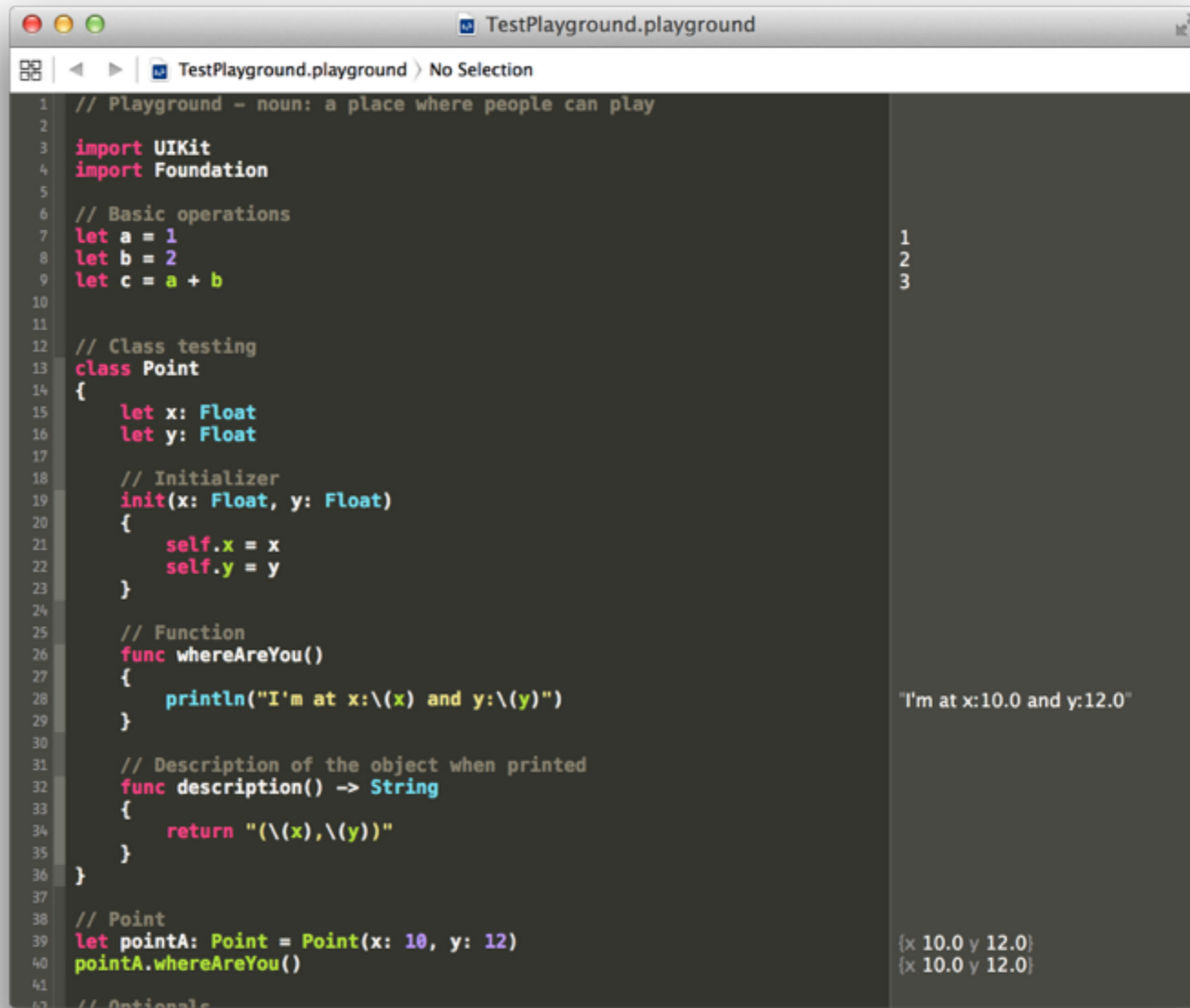
```
let pi = 3.14159
// pi is inferred to be of type Double

let pi : Double = 3.14159
// No type inference
```

Le type des variables et des constantes est “deviné” par le compilateur

Une fois le type attribué, une variable ne peut pas changer de type

# Playground



```
1 // Playground - noun: a place where people can play
2
3 import UIKit
4 import Foundation
5
6 // Basic operations
7 let a = 1
8 let b = 2
9 let c = a + b
10
11
12 // Class testing
13 class Point
14 {
15     let x: Float
16     let y: Float
17
18     // Initializer
19     init(x: Float, y: Float)
20     {
21         self.x = x
22         self.y = y
23     }
24
25     // Function
26     func whereAreYou()
27     {
28         println("I'm at x:\(x) and y:\(y)")
29     }
30
31     // Description of the object when printed
32     func description() -> String
33     {
34         return "\(x), \(y)"
35     }
36 }
37
38 // Point
39 let pointA: Point = Point(x: 10, y: 12)
40 pointA.whereAreYou()
41
42 // Optionals
```

1  
2  
3

"I'm at x:10.0 and y:12.0"





{x 10.0 y 12.0}  
{x 10.0 y 12.0}



# En savoir plus

2 ebooks gratuits, écrits et distribués par Apple

**BOOKS**

 <p>The Swift Programming Language</p> 	<p><b>The Swift Programming Language</b></p> <p>Apple Inc. &gt;</p> <p>★★★★★ (100)</p> <p>Programming</p> <p>2 Jun, 2014</p> <p>Read ▾</p>	 <p>Using Swift with Cocoa and Objective-C</p> 	<p><b>Using Swift with Cocoa and Objective-C</b></p> <p>Apple Inc. &gt;</p> <p>★★★★★ (15)</p> <p>Programming</p> <p>2 Jun, 2014</p> <p>Read ▾</p>
---	--	---	---





**Outils de développement**



# Get a Mac

Macbook

Macbook Pro

iMac

Mac Pro

Mac Mini

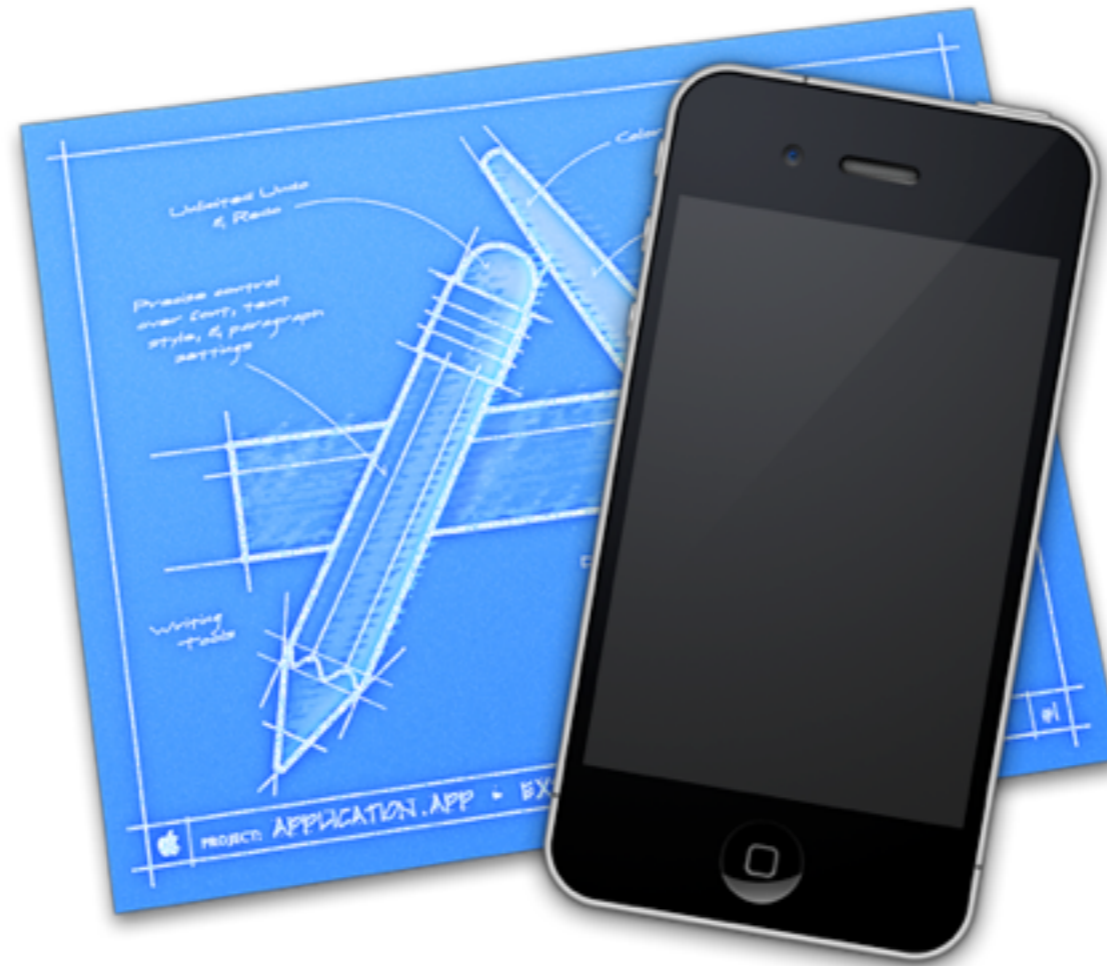




# Xcode



# Simulateur iPhone & iPad



# Programme développeur

*\$99 / 80€*

Valable pendant **1 an** (renouvelable)

Applications illimitées sur l'App Store

Indispensable pour déployer sur device



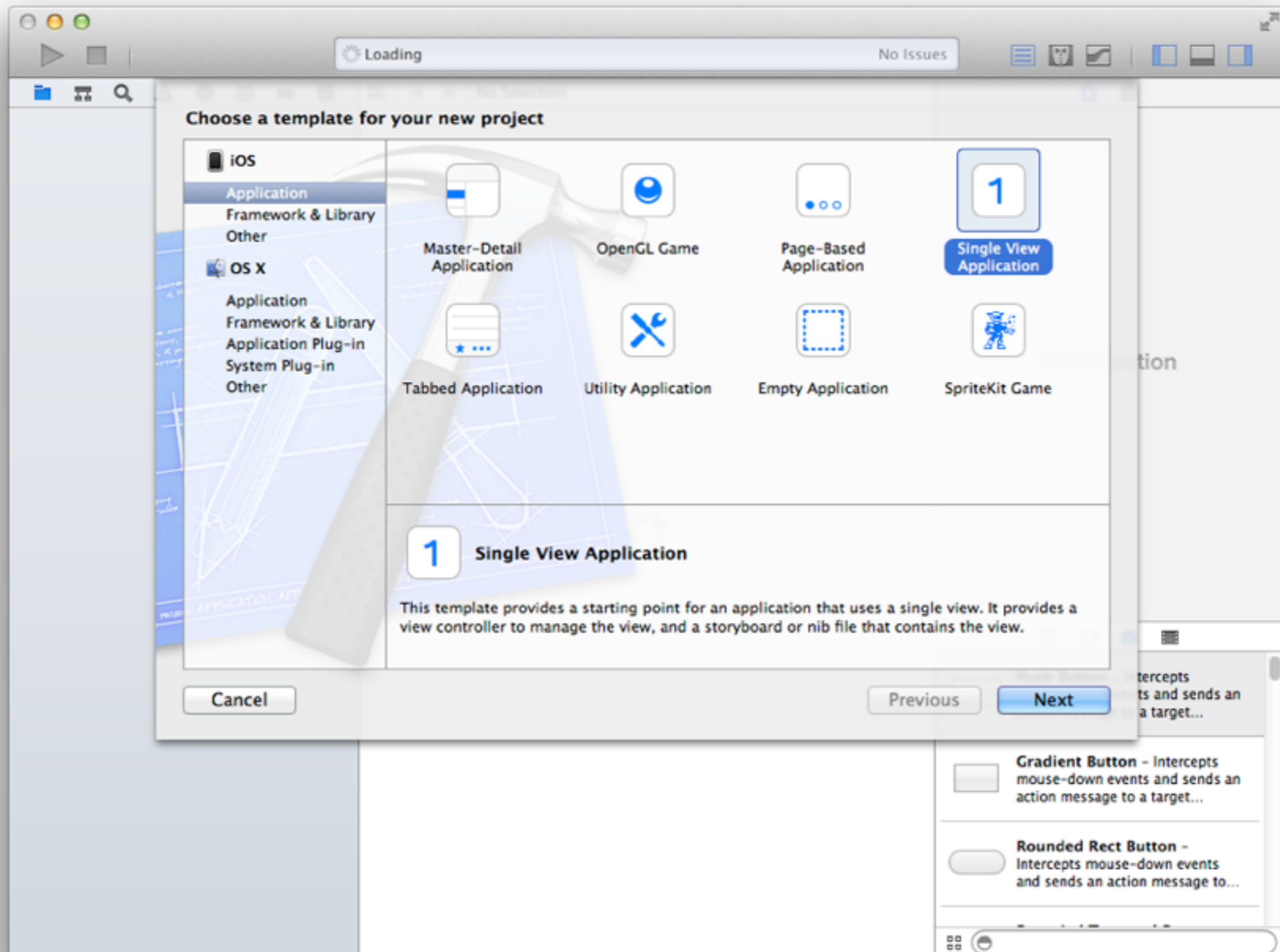




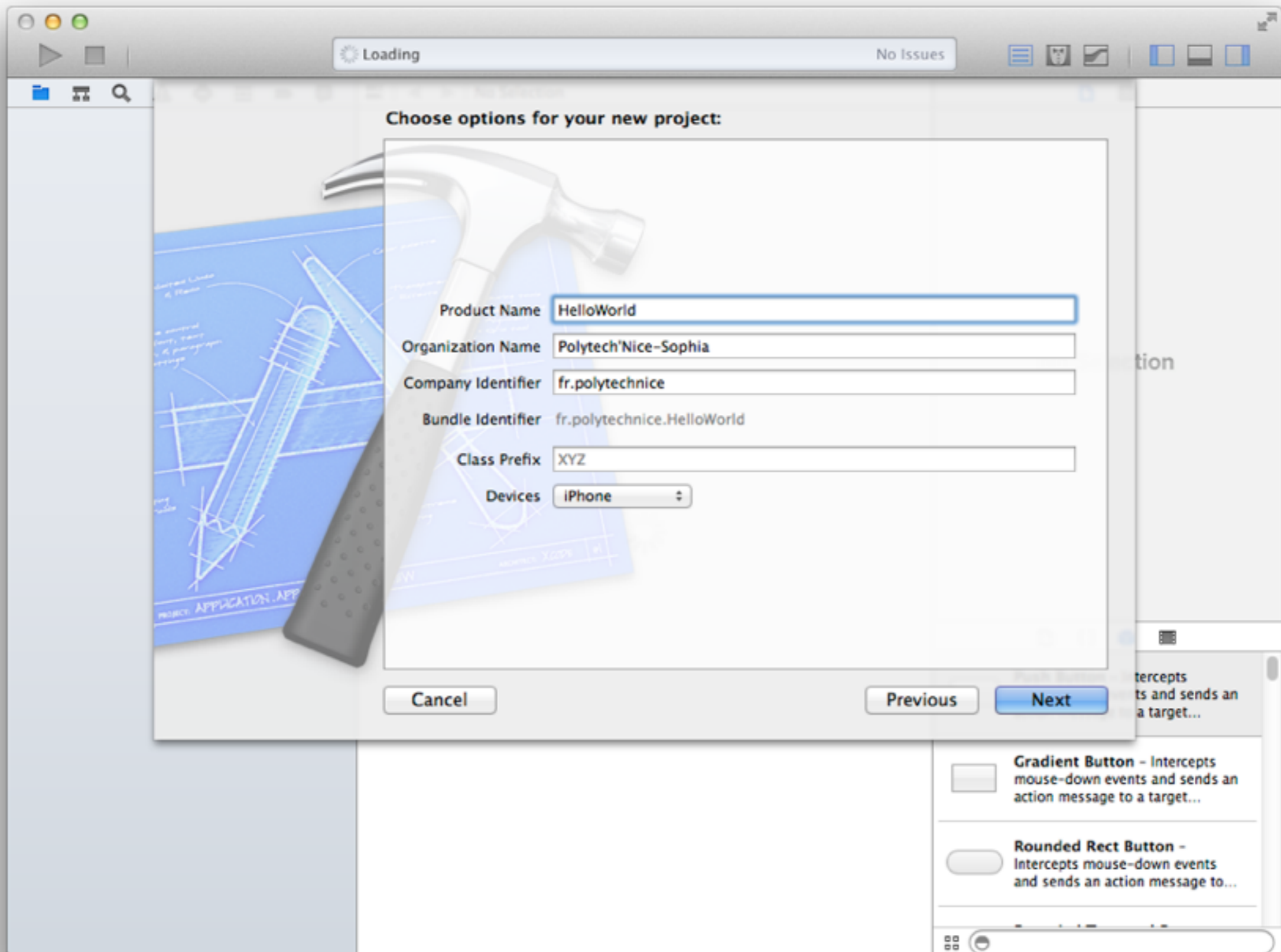
**Premier projet**



# Création à partir d'un template



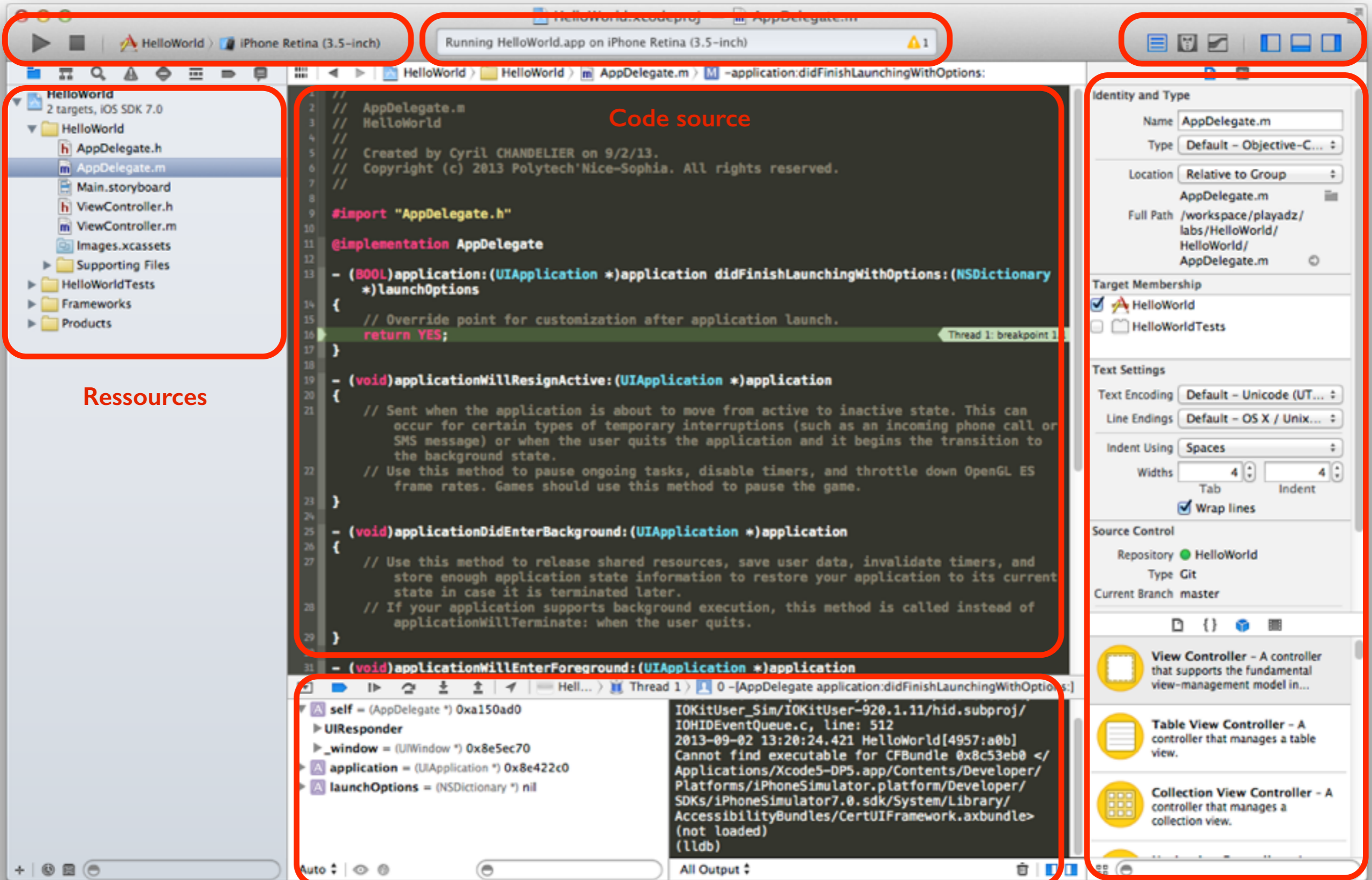
# Informations générales



Lancer le projet / Choix de la cible

Indicateur d'activité

Configuration de la fenêtre



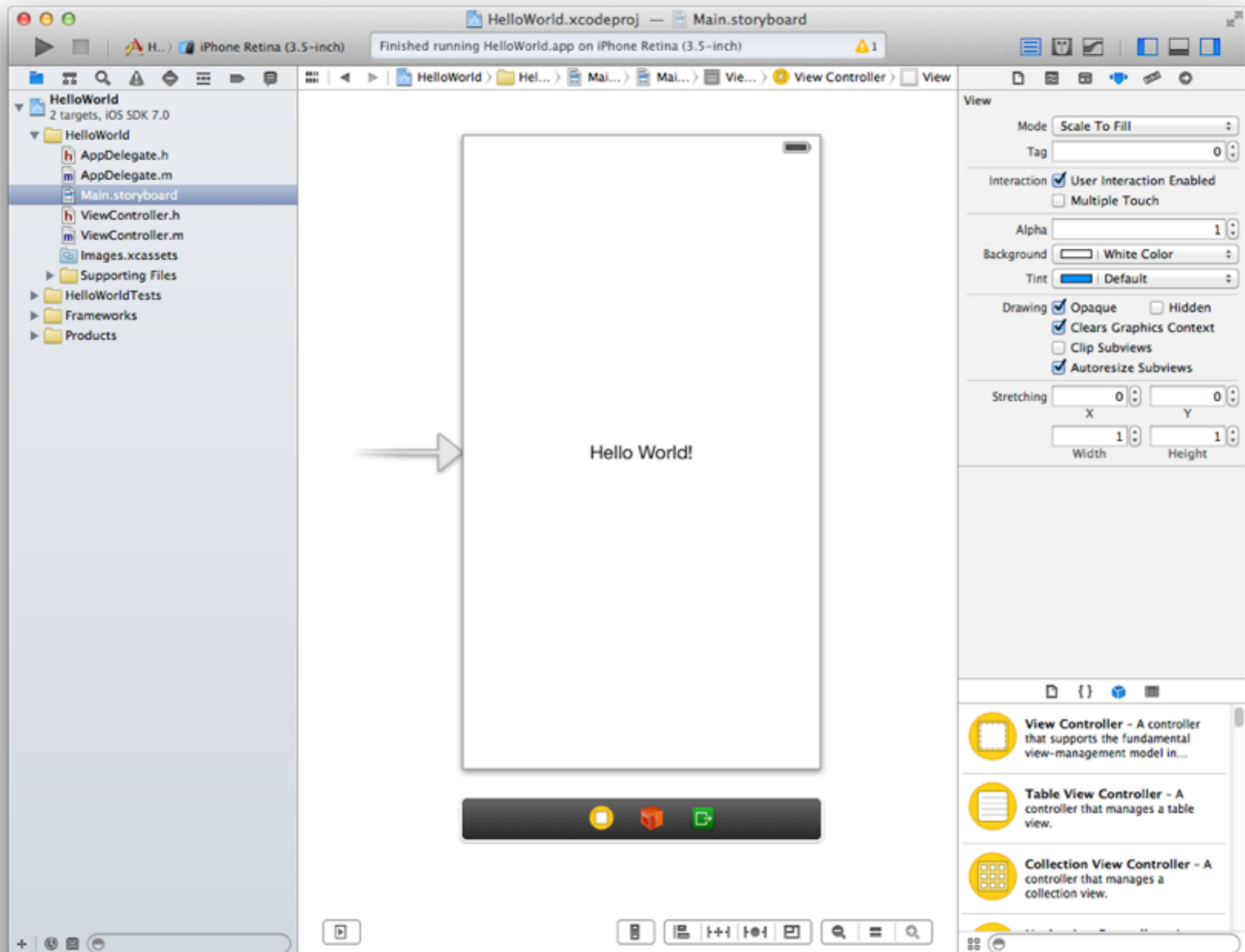
Ressources

Code source

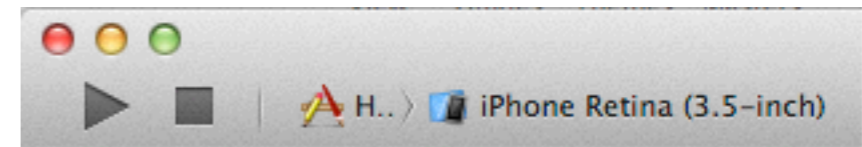
Debugger

Inspecteur





# Lancement de l'application



Build and Run

Compilation du projet et  
des dépendances

Ouverture du simulateur

Premier projet lancé!







# Persistance des données



# PLIST

## Sauvegarde de données dans un fichier sur le disque

Key	Type	Value
▼ Root	Array	(4 items)
▼ Item 0	Dictionary	(3 items)
title	String	Home
viewController	String	HomeViewController
image	String	picto_home.png
▼ Item 1	Dictionary	(3 items)
title	String	Posts
viewController	String	PostListViewController
image	String	picto_posts.png
▼ Item 2	Dictionary	(3 items)
title	String	Shops
viewController	String	ShopListViewController
image	String	picto_shops.png
▼ Item 3	Dictionary	(3 items)
title	String	Contact
viewController	String	ContactViewController
image	String	picto_contact.png

```
// Get saved data
NSBundle *mainBundle = [NSBundle mainBundle];
NSString *filePath = [mainBundle pathForResource:@"Menu" ofType:@"plist"];
NSArray *menus = [NSArray arrayWithContentsOfFile:filePath];
for(NSDictionary *menu in menus)
{
    // Do something
}
```

```
// Save data
NSArray *arrayToSave = @[ @"A", @"B", @"C" ];
[arrayToSave writeToFile:@"save.plist" atomically:YES];
```

# User Defaults

Sauvegarde des préférences et réglages de l'utilisateur (entre autres)

Gère la sauvegarde d'objets

Gère certains types primitifs (NSInteger, float, double, BOOL)

```
// User defaults
NSUserDefaults *userDefaults = [NSUserDefaults standardUserDefaults];

// Save data
[userDefaults setBool:YES forKey:@"SOUND_ENABLED"];
[userDefaults synchronize];

// Get the saved data
BOOL soundEnabled = [userDefaults boolForKey:@"SOUND_ENABLED"];
if(soundEnabled)
{
    // Do something
}
```

# Core Data

OPM

- r

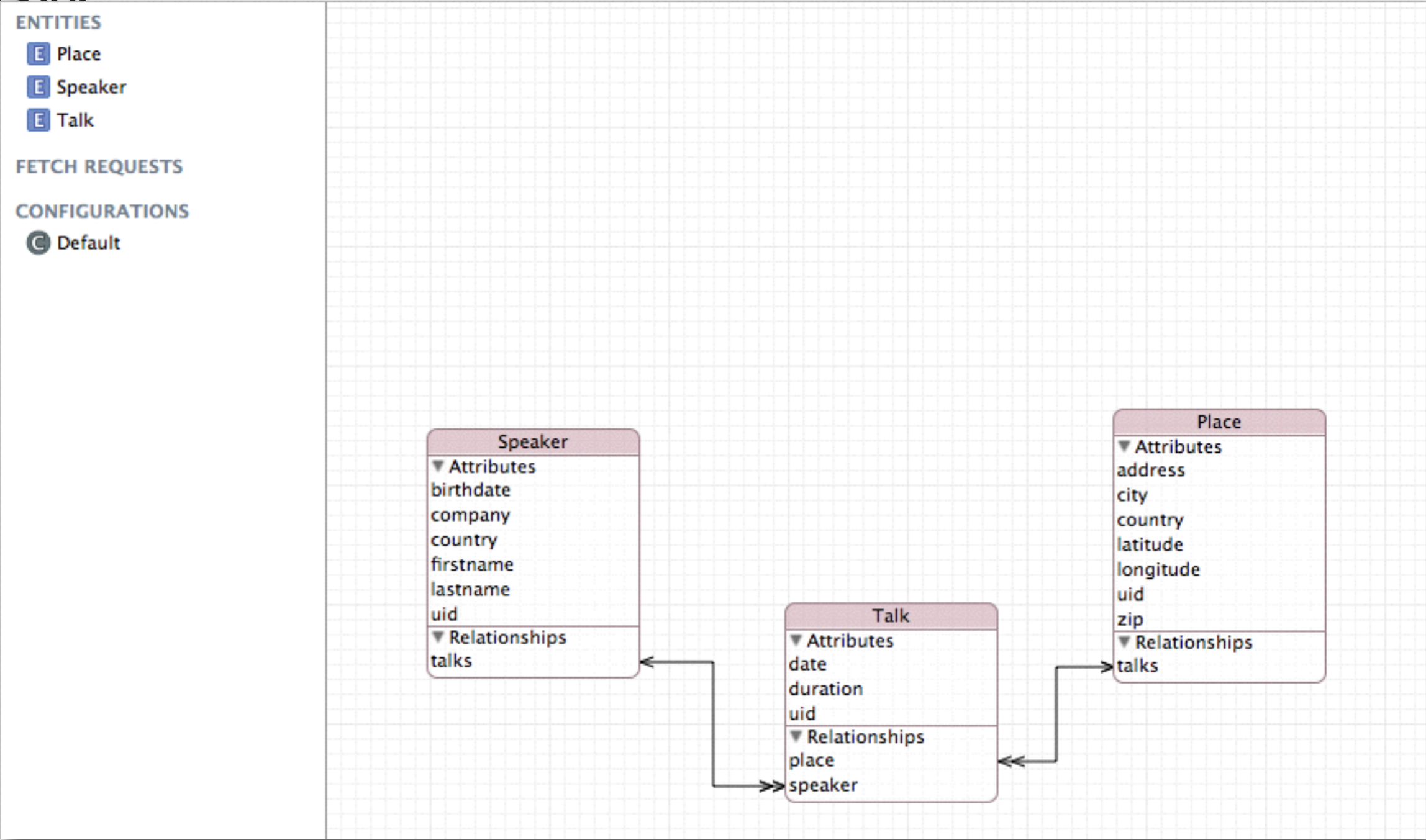
- M

- M

St

- b

- f



- memory

Synchronisation de donnée dans le cloud

Lié à l'**Apple ID** de l'utilisateur (compte Apple)

Synchronisation instantanée entre les devices

Nouveau : CloudKit (iOS 8)

> private / public databases (eq. backend + WS)

> asset storage

> authentication via l'Apple ID (eq. Facebook Connect)







# Déploiement



# Portail développeur

Certificates, Identifiers & Profiles

Cyril CHANDELIER ▾



iOS Apps



[Certificates](#)



[Identifiers](#)



[Devices](#)



[Provisioning Profiles](#)

**Learn More**

 [App Distribution Guide](#)



Mac Apps

**Join the Mac Developer Program**  
Get everything you need to develop, sign, and distribute your apps.

[Learn more](#)

[Join now](#)



Safari Extensions

**Join the Safari Developer Program**  
Create a signing certificate for your Safari extensions.

[Learn more](#)

[Join now](#)



# Gestion des apps

Apple iTunes Connect

Add New App

## Manage Your Apps


### Recent Activity

See All 


iOS App Recent Activity

3 Total




 Nanashi 1.1



 LASIDO 1.5



 Jellyfish Alerts 1.2

### Search

Name :

Apple ID :

SKU :

Status :

Search

<http://itunesconnect.apple.com>

# Testflight

Service racheté par Apple en 2013

Intégré à iTunes Connect depuis Septembre 2014

Gestion de beta testing (utilisateurs, feedback, etc.)

Jusqu'à 1000 beta testers

<http://www.testflight.com/>

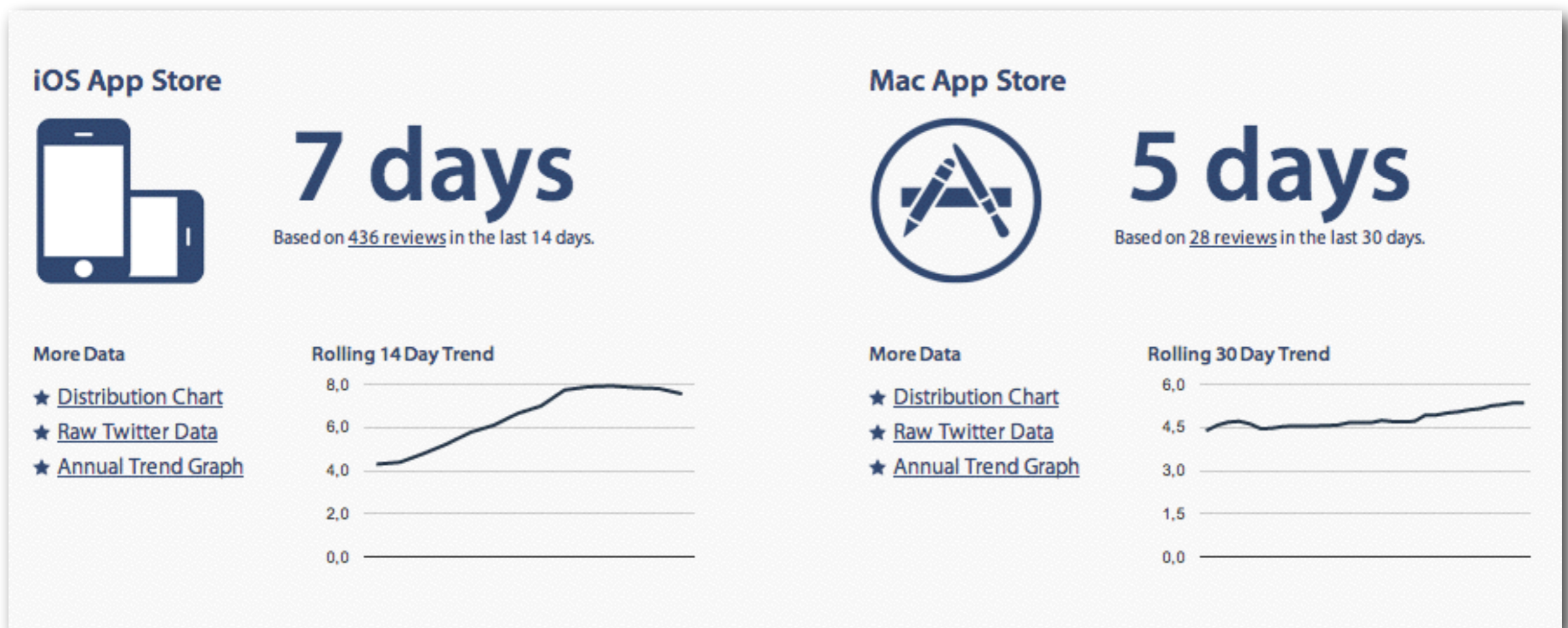
# Validation

3 à 15 jours (sauf exception)

Ne pas utiliser d'API privées (non documentées) sous peine de rejet

Attention au respect des guidelines Apple

<http://reviewtimes.shinydevelopment.com/>







**Frameworks majeurs**



Composants d'UI de base

Accéléromètre

Mouvements du device

Afficher/prendre des photos/vidéos

[https://developer.apple.com/library/ios/documentation/uikit/reference/UIKit\\_Framework/index.html](https://developer.apple.com/library/ios/documentation/uikit/reference/UIKit_Framework/index.html)

# MapKit + CoreLocation

Affichage de cartes

Géo-localisation

Reverse geocoding

Boussole



Classements

Scores

Achievements

Joueur contre joueur (tour par tour, temps réel)

P2P

[https://developer.apple.com/library/ios/documentation/GameKit/Reference/GameKit\\_Collection/\\_index.html](https://developer.apple.com/library/ios/documentation/GameKit/Reference/GameKit_Collection/_index.html)

# AddressBook / AddressBookUI

Accéder au répertoire

Manipuler les contacts

AddressBookUI fournit les interfaces natives

<https://developer.apple.com/library/ios/documentation/ContactData/Conceptual/AddressBookProgrammingGuideforiPhone/Introduction.html>

## Partager sur les réseaux sociaux

- Twitter ( $\geq$  iOS 5)
- Facebook ( $\geq$  iOS 6)
- Weibo ( $\geq$  iOS 6)

[https://developer.apple.com/library/ios/documentation/Social/Reference/Social\\_Framework/\\_index.html](https://developer.apple.com/library/ios/documentation/Social/Reference/Social_Framework/_index.html)



## Gestion des achats In-App (In-App Purchase)

- consommables
- débloquer une partie de l'application
- abonnements

[https://developer.apple.com/library/mac/documentation/StoreKit/Reference/StoreKit\\_Collection/\\_index.html](https://developer.apple.com/library/mac/documentation/StoreKit/Reference/StoreKit_Collection/_index.html)





**Librairies externes**



# CocoaPods

```
$ edit Podfile
platform :ios, '6.0'
pod 'JSONKit', '~> 1.4'
pod 'Reachability', '~> 3.0.0'

$ pod install
```



**CocoaPods**

The best way to manage library dependencies in Objective-C projects.

<http://cocoapods.org/>



# Cocoa Controls



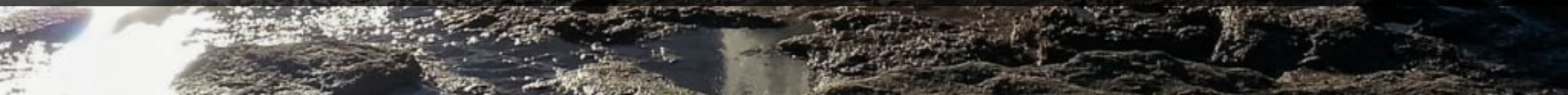
The screenshot displays the Cocoa Controls website interface. At the top left is the Cocoa Controls logo, featuring a cocoa pod icon and the text "cocoa CONTROLS". To the right of the logo, it states "1585 open source and commercial UI components for iOS and OS X." Below this, there are filter options for "Sort" (Date, Rating, Apps), "License" (All, Apache 2.0, BSD, CC BY 3.0, CC BY-SA 3.0, Commercial, Custom, Eclipse Public License), "Platform" (All, iOS, OS X), and "Filter CocoaPods" (Yes, No). The main content area shows three preview cards for different UI components: a menu with blue, black, and red backgrounds; a collection of buttons and alerts including "Default", "Primary", "Success", "Info", "Warning", "Danger", "Bookmark", "Done", "Delete", and "Download"; and a card titled "Simple keyframe animations" featuring a blue unicorn illustration.



[www.cocoacontrols.com](http://www.cocoacontrols.com)



**Divers**



# Documentation

Depuis Xcode

⌘ ⇧ 0 ou depuis le menu **Aide**

Depuis un navigateur

<http://developer.apple.com/library/ios/>



# Guidelines

## **iOS Human Interface Guidelines**

<https://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html>

## **App Review Guidelines**

<https://developer.apple.com/appstore/resources/approval/guidelines.html>

## **In-App Purchase Guidelines**

<https://developer.apple.com/in-app-purchase/In-App-Purchase-Guidelines.pdf>

## Statistiques

<http://www.appexplorer.com/stats/>

<http://mixpanel.com/trends/>

## Classement TIOBE

<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>





**Questions ?**



<http://tryobjective.codeschool.com>

Ne pas hésitez à copier le code dans Xcode sur vos machines pour voir le comportement réel une fois compiler (quelques petites différences)