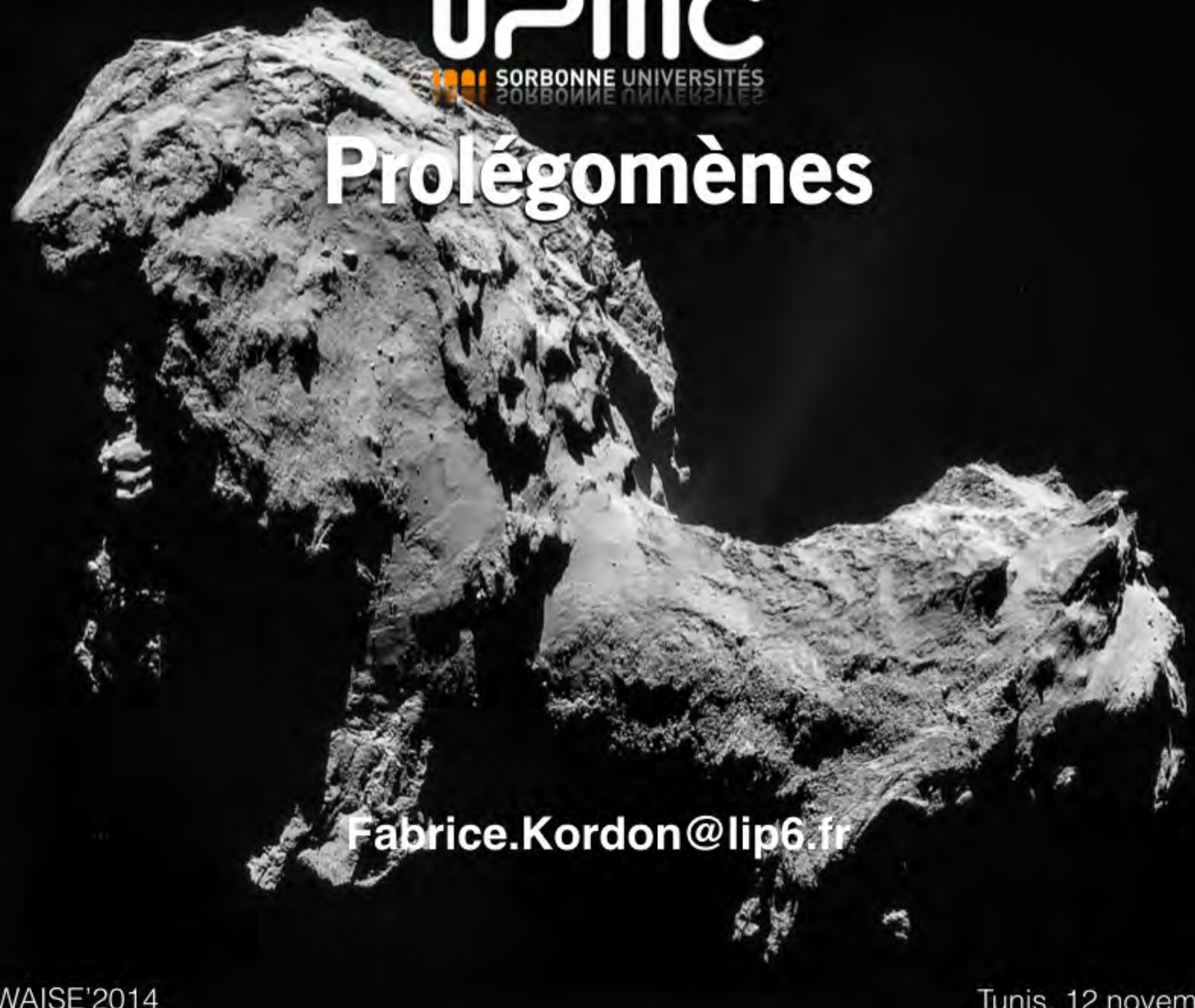
UPMC
SORBONNE UNIVERSITÉS

Mobile platforms Programming: iOS (session 1)

Tutorial
IWAISE'2014 — Tunis — November 12, 2014

UPMC
SORBONNE UNIVERSITÉS

Prolégomènes



Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

Aujourd'hui...

Mission Rosetta

- Comprendre les comètes
- Analyser Tchourioumov-Guérassimenko
  - 10 ans après le lancement de la sonde
  - À une distance de ~400x10⁶ km

La sonde Philae sur le noyau de la comète

- En ce moment, la procédure a lieu
- Site choisi nommé «Agilkia»

Et le rapport avec iOS

- C'est un système embarqué!

Un peu de publicité

- **Ce tutoriel est issu d'éléments d'un cours existant**
  - Trop long et trop riche pour une journée
- **Ressources sur le cours de M2**
  - 5 années d'existence
    - **2014: http://lip6.fr/Fabrice.Kordon/5I452-201**
  - Et sur iTunesU
    - **https://itunes.apple.com/fr/itunes-u/programmation-sur-plateformes/id919453516?mt=10**
- **Et un MOOC sur France Université Numérique**
  - Avril 2014:
    - **https://www.france-universite-numerique-mooc.fr/courses/UPMC/18001/Trimestre_2_2014/about**
  - Février et Avril 2015 : annonce bientôt

Agenda de ce tutoriel

Session 1 (2h30)

- Partie magistrale
  - **Objectif, construire une application minimale**
- Petits exercices

Session 2 (1h30)

- Partie magistrale

Session 3 (2h)

- Exercices
  - **Swift (le nouveau langage d'Apple)**
  - **Plus d'informations sur la construction d'une application «simple»**

Agenda de ce tutoriel

- **Session 1 (2h30)**
  - Partie magistrale
    - **Objectif, construire une application minimale**
  - Petits exercices
- **Session 2 (1h30)**
  - Partie magistrale
- **Session 3 (2h)**
  - Exercices
    - **Swift (le nouveau langage d'Apple)**
    - **Plus d'informations sur la construction d'une application «simple»**

Ressources en ligne!

http://www.lifl.fr/iwaise14/tutorial-ios.html

UPMC
SORBONNE UNIVERSITÉS

Qu'est-ce qu'un terminal mobile?

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

Vous avez dit «terminal mobile»?

- Ordinateur miniature…
- «Miniature» = «système embarqué»
  - Conséquence sur les ressources
- Système réactif
  - Centré sur l'interface utilisateur
- Smartphone ou tablette?
  - La taille de l'écran
  - Pour le reste, tout est «presque identique»!
    - **Mécanismes propres**
    - **Exploitation de la taille de l'écran**

L'architecture (habituelle)

- **Processeur**
  - ARM (basse consommation)
- **Disque**
  - Mémoire flash
- **Mémoire vive**
  - RAM
- **Périphériques**
  - Capteurs diverses
  - Cartes réseaux
  - etc.

iOS versus Android?

Même principes

- Java / Objective-C ou Swift
- Eclipse+plug-in / Xcode
- Brevets dans tous les coins

Qui a copié?

- 08/2005 (rachat d'Android Inc.)
- 01/2007 iPhone (annonce)
- 09/2007 iPod Touch
- 11/2007 Consortium Open Handed Alliance
- 03/2008 Xcode/SDK «iPhone Os» (beta)
- 12/2008 Android SDK
- 05/2012 Windows 8

iOS versus Android?

Même principes

- Java / Objective-C ou Swift
- Eclipse+plug-in / Xcode
- Brevets dans tous les coins

Qui a copié?

- 08/2005 (rachat d'Android Inc.)
- 01/2007 iPhone (annonce)
- 09/2007 iPod Touch
- 11/2007 Consortium Open Handed Alliance
- 03/2008 Xcode/SDK «iPhone Os» (beta)
- 12/2008 Android SDK
- 05/2012 Windows 8

Difficile de juger!!!

En guise de conclusion…

La bataille se joue sur l'écosystème!

	Apple	Google
Nombre d'applications	**900K** (07/2013)	**700K** (09/2014)
Téléchargements	100 x 10^9 cumulés (fin 2013)	50 x 10^9 cumulés (fin 2013)
Développeurs*	50% avec $500-/mois/app	64% avec $500-/mois/app

*: http://www.developereconomics.com/reports/developer-economics-q3-2014/

Et l'écosystème?

- Offre? nombre de développeurs? nombre d'applications?
  - **Attirer les développeurs**
  - **Générer de nouveaux usages…**

UPMC
SORBONNE UNIVERSITÉS

Quelques éléments sur la programmation embarquée

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction

Programmation embarquée implique…

- Contraintes
  - **Mémoire, énergie, CPU**
- Compilation croisée (souvent)

Programmation sur plateformes mobiles?

- Programmation embarquée…
- Programmation centrée sur l'interface utilisateur
  - **Modèle MVC**
- Programmation réactive

Compilation croisée

- **Déploiement spécifique — environnements dédiés**
- **Debug «en deux temps»**
  - Sur simulateur
  - Sur terminal

Compilation croisée

exécution native
(compilation croisée)

- **Déploiement spécifique — environnements dédiés**
- **Debug «en deux temps»**
  - Sur simulateur
  - Sur terminal

Compilation croisée

exécution native
(compilation croisée)

- **Déploiement spécifique — environnements dédiés**
- **Debug «en deux temps»**
  - Sur simulateur
  - Sur terminal

Contraintes de programmation

Ressources limitées

- Mémoire
  - Attention à la gestion (et au garbage  et les préemptions associées)
  - Android : «ramasse miette» («garbage collection») propre à Java
  - iOS, compteurs de références (ARC = Automatic Reference Counting)
  - Objective-C & Swift
- Consommation d'énergie
  - Attention aux périphériques gourmands (GPS, caméra/HDR/Flash, etc.)
- CPU (lié à la consommation d'énergie)
  - Attention aux algorithmes coûteux (jeux?)

Notion d'«événement urgent»

- Plus de mémoire ou d'énergie disponible
- Appel téléphonique (pour les «smartphones»)

Autres enjeux en programmation

Ergonomie

- Facilité d'usage!
- Respecter le modèle de réactivité («look and feel»)
  - **Charte graphique**
  - **Mécanismes dédiés**

Fiabilité

- Éviter les plantages (e.g. problèmes de mémoire)

Sécurité

- Attention à la protection des données personnelles
  - **Accès contrôlé par des API à certains éléments**

Rapidité

- Penser les aspects algorithmiques (parfois intégrés dans l'OS)

En guise de conclusion…

Bref, beaucoup de compromis…

…et Apple peut trouver dans vos erreurs des raisons de rejet de vos Apps

- **Vous devez donc penser à tout cela…**
- **C'est la clef de la réussite d'une application**
  - Avec «la bonne idée» bien sûr

UPMC
SORBONNE UNIVERSITÉS

La chaîne de production d'applications iOS

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

Mécanismes de tests et d'analyse

Développement

Mécanismes de tests et d'analyse

Développement

Tests unitaires

Mécanismes de tests et d'analyse

Développement

Tests unitaires

Analyser
(compilateur)

Le «wokflow» d'Apple

Développement

Le «wokflow» d'Apple

Le «wokflow» d'Apple

Développement

Exécution
(sur terminal)

provisioning
profile

Exécution
(simulateur)

Le «wokflow» d'Apple

Développement

Exécution
(sur terminal)

provisioning
profile

Exécution
(simulateur)

Profiler
(instruments)

Le «wokflow» d'Apple

Développement

Exécution
(sur terminal)

provisioning
profile

Exécution
(simulateur)

Profiler
(instruments)

Déploiement

En guise de conclusion…

- **Xcode = environnement phare de développement**
  - Conçu et maintenu par Apple
  - Évolue à la vitesse d'iOS (mais nécessite un Mac)
- **Autres alternatives**
  - Des possibilités (limitées) sous Windows (cf moteur de recherche)
    - **N'exploitent pas forcément les dernières nouveautés**
    - **Embarquent souvent un «moteur» intégré**
  - Environnements de développements alternatives sous MacOS
    - **Demander aux moteurs de recherche**
  - Environnements multi-plateformes
    - **Souvent limité à l' «intersection» des fonctions entre iOS et Android**
  - On peut toujours faire du HTML5
    - **Exécution dans un navigateur**
    - **Limites fortes (look and feel, etc.)**
    - **par contre, approche multi-plateforme** (peu/moins usitée)

En guise de conclusion…

- **Xcode = environnement phare de développement**
  - Conçu et ma
  - Évolue à la vi
- **Autres alter**
  - Des possibilit
    - **N'exploitent**
    - **Embarquen**
  - Environnem
    - **Demander**
  - Environnem
    - **Souvent lim**
  - On peut to
    - **Exécution**
    - **Limites fortes (look and feel, etc.)**
    - **par contre, approche multi-plateforme** (peu/moins usitée)

Quelques liens…

AppCode (http://www.jetbrains.com/objc)
Payant
∃ licences «académiques» et «open source»

Haxe (multi-plateformes), http://haxe.org

Titanium (http://www.appcelerator.com/titanium)

Corona (http://www.coronalabs.com)

UPMC
SORBONNE UNIVERSITÉS

Principes de déploiement d'une application iOS

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction

Exécuter une Application iOS

- Dans le simulateur
  - **Gratuit…**

… mais pas très pratique

- Sur le terminal
  - **Mieux…**

… mais plus compliqué

Voyons les possibilités

Déployer une application — technique 1

Localement

- Via le connecteur USB

- Requiert un abonnement développeur (provisioning profile)
- Possitibilité d'utiliser un «University program»

Déployer une application — technique 2

Via l'AppStore

- Procédure de validation (Apple)
- Apple prélève 30%!!!

- Requiert un abonnement développeur (provisioning profile)

Déployer une application — technique 3

Via un intranet + iTunes

- Génération d'un fichier «.ipa»
- Le même que pour l'AppStore mais géré explicitement

- Requiert un certificat «enterprise» (distribution profile)

Les programmes de développement

	rien	University	Enterprise	Standard
SDK	✔	✔	✔	✔
ßSDK	✘	✘	✔	✔
Forums	✔	✔	✔	✔
Simulateur	✔	✔	✔	✔
Déploiement	✘	✔	✔	✔
Distribution	✘	✘	✔	✘
AppStore	✘	✘	✘	✔
Coût	**0$**	**—**	**299$**	**99$**

Les programmes de développement

	rien	University	Enterprise	Standard
SDK	✔	✔	✔	✔
ßSDK	✘	✘	✔	✔
Forums				✔
Simulateur				✔
Déploiement				✔
Distribution	✘	✘	✔	✘
AppStore	✘	✘	✘	✔
Coût	**0$**	**—**	**299$**	**99$**

Déployer une application
(exécution hors simulateur)
requiert une identification

En guise de conclusion…

Chaîne de déploiement très cadrée

- Bien plus que sous Android
  - **On peut juste «faire un scp»…**

Avantages

- Une protection certaine
  - **Contre les malveillances (virus, etc.)**
- Un contrôle qualité
  - **Limiter les erreurs «imputables au terminal»**

Inconvénients

- Une forme de protection
  - **Contre la concurrence (marché captif)**
- Un mécanisme de déploiement complexe

En guise de conclusion…

Chaîne de déploiement très cadrée

- Bien plus que sous Android
  - **On peut juste «faire un scp»…**

Avantages

- Une protection certaine
  - **Contre les malveillances (virus, etc.)**
- Un contrôle qualité
  - **Limiter les erreurs «imputables au terminal»**

Inconvénients

- Une forme de protection
  - **Contre la concurrence (marché captif)**
- Un mécanisme de déploiement complexe

Un écosystème centré sur Apple

UPMC
SORBONNE UNIVERSITÉS

Généralités sur l'environnement de développement d'applications iOS

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

Petit historique

Xcode, l'environnement de développement d'Apple

- Au départ dédié à MacOS (iOS est une extension «naturelle»)
- Versions 3.*: à partir de 2008 — la première à supporter «iPhoneOS»
- versions 4.*: à partir de 2011
- versions 5.*: à partir de 2013
- version 6 : annoncée en juin 2014
  - **Celle que nous utilisons** (exercices partiellement préparés sur versions ß + GM)

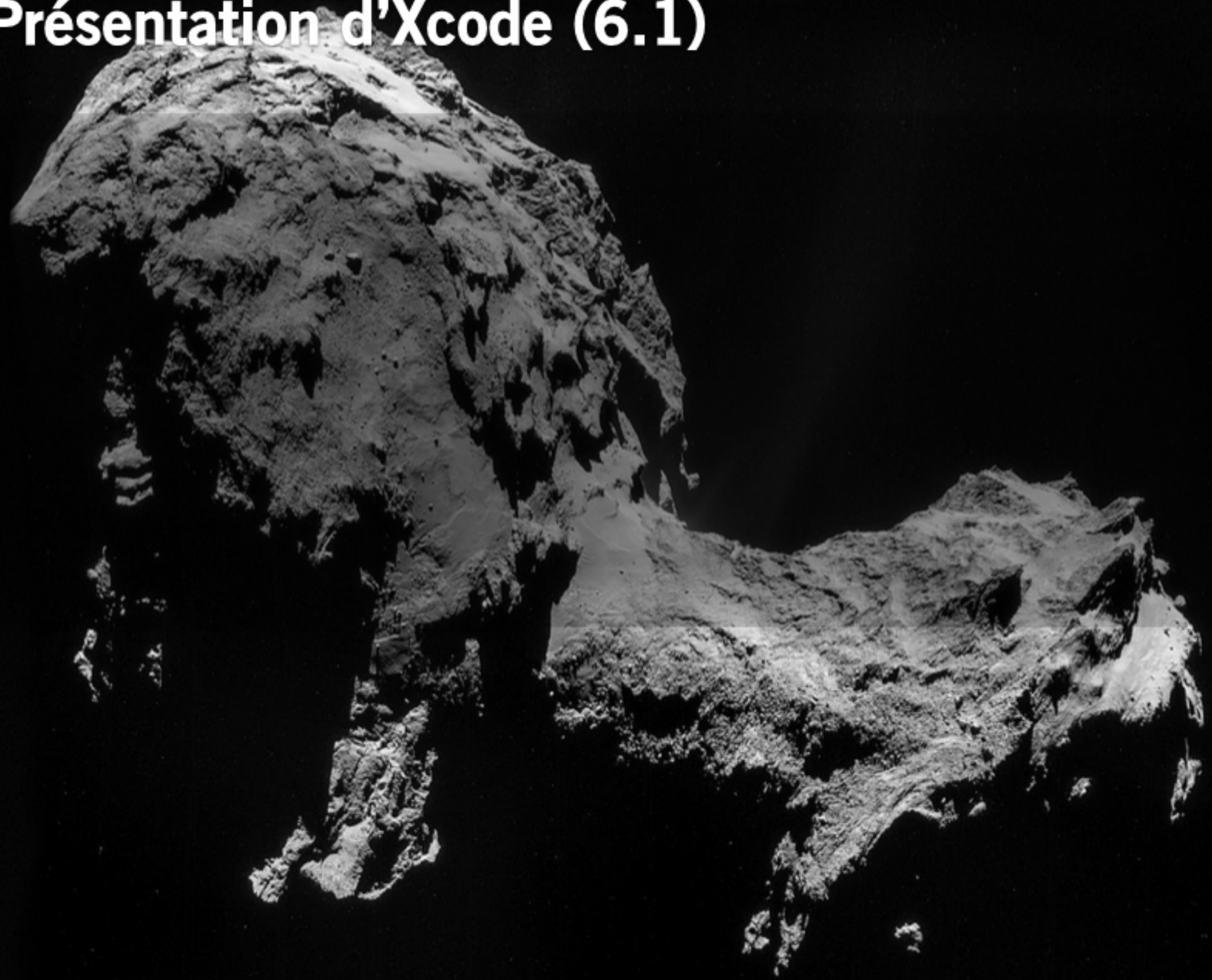
C'est un GUI + SDK

- Graphical User Interface
  - **Pour la conception…**
- Software Development Kit
  - **Environnement intégré de développement (incluant la documentation)**
  - **ici, compilation croisée AMD64->ARM**

Qu'y trouve-t-on?

- Xcode (la «porte d'entrée»)
- Les modules
  - StoryBoard
  - Simulateur/Débugger
  - Instrument («profiler»)
  - Analyse statique
  - Gestionnaires de versions intégrés
    - subversion, git
  - **Accès direct à la documentation!!!!!**

Présentation d'Xcode (6.1)



En guise de conclusion…

- Vous savez maintenant une idée de l'ensemble !!!

- Xcode s'est fortement rapproché du «style Eclipse»
  - Mais vous ne vous en plaindrez pas forcément

- Cela dit, c'est un outil de bonne qualité
  - Associé à un excellent compilateur: llvm

UPMC
SORBONNE UNIVERSITÉS

Xcode, création d'un projet

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

Les étapes de la création d'un projet

- **Facile!!! il faut répondre à des questions**
  - Choix du type d'application
  - Choix du type de terminal cible
  - Choix du langage
  - sélection de l'identifiant
  - etc.
- **Si des projets ont déjà été créés, des valeurs par défaut sont proposées**

Création d'un projet

- Dans Xcode

En guise de conclusion…

Vous savez maintenant créer un «projet vide» dans Xcode

Il va falloir apprendre à le remplir

UPMC
SORBONNE UNIVERSITÉS

Principes de construction de l'interface (mode «kindergarden»)

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction…

Problème, construction d'une interface graphique

- Délicat / notions complexes
- Travail sur l'«écosystème» = faciliter le travail
  - **Plusieurs façons de faire**

Par dessin («kindergarder» — «quiche eater*»)

- Utilisation d'un «clicodrome»…
- Permet de ne pas se soucier (maîtriser?) «ce qui se passe»

Programmatique (pour les «real programmers*»)

- Tout est possible (sortir des «chantiers battus»)
- On maîtrise finement l'interface (mais il faut savoir faire)

* http://www2.informatik.uni-halle.de/lehre/c/realprog.html

En guise d'introduction…

Problème, construction d'une interface graphique

- Délicat / notions complexes
- Travail «[illegible]système» = faciliter le travail
  - Plusie[illegible]

Dans la «vraie vie»?

On utilise souvent «les deux»

Mais…

L'un s'appuie sur l'autre…
… devinez lequel?

Par de[illegible]ter*»)

- Utilisc[illegible]
- Perm[illegible]e passe»

Prog[illegible]mmers*»)

- Tout est possible (sortir des «chan[illegible]
- On maîtrise finement l'interface (mais il faut savoir faire)

* http://www2.informatik.uni-halle.de/lehre/c/realprog.html

StoryBoard, comment ça marche?

Interface graphique

- Dessiner ses écrans
  - **Positionner des contraintes**
- Scénariser l'application
- Lier les éléments de l'interface à du code

Génération de «code Autolayout»

- Autolayout introduit avec iOS 6 (2012)
- Positionnement des éléments de l'interface de manière relative
  - **Relations entre eux**
  - **Contraintes par rapport à l'écran (bords, largeur, hauteur)**

Autolayout aussi en mode «programmatique»

- Une troisième manière de faire…

Un mot sur les résolutions…

Différent types d'écrans

- 480x320 (déprécié depuis Xcode 5.5 + en voie de disparition)
- 960x640 retina 3.5" (iPodTouch 3/4 + Iphone 4/4S)
- 1027x768 iPad 1 & 2, iPad mini)
- 1136x640 retina 4" (iPodTouch 5 + iPhone 5/5S)
- 1334x750 retina 4.7" (iPhone 6)
- 2048x1536 retina (iPad 3/4/air + iPad mini retina)
- 1920x1080 retina HD 5.5" (iPhone 6+)

Deux modes/comportements

- «petit terminal» & «grand terminal»

Rotation du terminal

- Pris en charge par StoryBoard
  - **Si bien géré!!!**

Un mot sur les résolutions…

Différent types d'écrans

- 480x320 (déprécié depuis Xcode 5.5 + en voie de disparition)
- 960x640 retina 3.5" (iPodTouch 3/4 + Iphone 4/4S)
- 1027x768 iPad 1 & 2, iPad mini)
- 1136x640 retina 4" (iPodTouch 5 + iPhone 5/5S)
- 1334x750 retina 4.7" (iPhone 6)
- 2048x1536 retina (iPad 3/4/air + iPad mini retina)
- 1920x1080 retina HD 5.5" (iPhone 6+)

Un nouvel écran à venir!!!

Deux modes/comportements

- «petit terminal» & «grand terminal»

Rotation du terminal

- Pris en charge par StoryBoard
  - **Si bien géré!!!**

Ce qu'il faut éviter!!!

La gestion «en dur» de la résolution…

- C'est moche!

En guise de conclusion…

L'importance de l'«écosystème»

- Et le «besoin de troupes» pour l'enrichir!
- Vous avez l'équivalent sous Android et sous Windows8/9

Il est important de connaître le mode «kindergarden»

- Mais il est aussi important de le dépasser
- On s'en servira «au début»

Hello World avec Xcode (démo de Storyboard)

- Dans Xcode

En guise de conclusion…

Vous connaissez les bases

de la construction d'interfaces avec StoryBoard

On peut a priori tout gérer

moi je trouve cela «bien compliqué et magique»*

* ceci est une opinion personnelle

UPMC
SORBONNE UNIVERSITÉS

Orientation du terminal & applications universelles avec Storyboard

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction…

Rappelez vous!!!

- ∃ «petit» et «grand» terminaux
- Des différences de taille dans l'écran
- Des différences de comportement
  - **Mode «paysage», apparition de menus, gestion multi-vue**

Applications dédiées à un terminal?

- Ce n'est pas raisonnable d'un point de vue maintenance
  - **Se justifie dans des cas particuliers ou si pas de sens pour un type de terminal**
- Il faut «penser Universal»
  - **Quid de l'Apple watch?**
  - **On en sait peu à ce stade…**

Universal? Orientation? comment gérer?

C'est pris en charge par «StoryBoard»

- Gestion unifiée de l'interface
- Définition de contraintes relatives
- Définition d'interfaces dédiées à un type de terminal donné
- Gestion d'une orientation donnée

Gros avantage!!!

- On «dessine» sans se préoccuper des positions
- Le système fournit des alertes
  - **Conformité à l'interface?**
  - **Contraintes oubliées**

Mais attention

- Une fausse manipulation est facile!!!

Préliminaire à la démonstration…

Il a des gens qui adorent

StoryBoard (et son ancêtre InterfaceBuilder)

on permis à des «djeuns»

de développer avec succès des applications…

…et de «devenir riche»

Préliminaire à la démonstration…

Il a des gens qui adorent

Story on ancêtre InterfaceBuilder)

on p «djeuns»

de d ivec succès des applications…

…et riche»

Préliminaire à la démonstration…

Il a des gens qui adorent

Story on ancêtre InterfaceBuilder)

on p «djeuns»

de d ivec succès des applications…

…et riche»

Préliminaire à la démonstration…

Il a des gens qui

Story

on p

de d

…et

on ancêtre

«djeu

ivec

riche»

Cela devient plus difficile…

Il y a une exigence de qualité…

Quoique parfois…

Préliminaire à la démonstration…

Il a de

Sta n ancêtre

on djeu

de ec s…

…et che»

Cela devient plus difficile…

Il y a une exigence de qualité…

Quoique parfois…

Démonstration…

- Contraintes & rotations
- Différentier les orientations
- Différentier les terminaux

En guise de conclusion…

Nous n'avons vu que «les bases»…

… vu mon niveau en StoryBoard

Pour en savoir plus!!!

UPMC
SORBONNE UNIVERSITÉS

Guide minimum de survie en Objective-C

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction…

Historique

- 1983
- S'inspire de C et de smalltalk
- Premier utilisateur: NextStep (NeXT)

Nous aborderons…

- Principes du langage
- Principaux éléments de syntaxe
- La classe NSString
  - **Vous en aurez rapidement besoin**

Objective-C, concepts de base

- Méthodes de classes et d'instances
- Communication entre classes

Principes d'Objective-C

- **Basé sur C** 
- **Sur ensemble strict de C**
  - Nouvelle syntaxe, nouveaux types
    - «[c m]», @property, @interface, @implementation, @synthesize, ...
    - id, class, selector
  - Héritage simple, «protocole» (délégation = framework)
  - Typage (relativement) fort (par rapport à C), mécanismes d'introspection...
- **Classes et instances sont des objets**
  - Méthodes de classes
  - Méthodes d'instances

Les méthodes

Les instances répondent à des méthodes d'instances

```
- (id) init;
- (float) temperature;
- (void) kekchose;
```

Les classes répondent à des méthodes de classe

```
+ (id) alloc;
+ (UIDevice*) currentDevice;
```

Usage régulier dans les API d'iOS

En objective-C, tout est message

- Syntaxe

```
[dest message];
[dest message:argument];
[dest message:arg1 etAussi:arg2];
```

- Exemple

```
meteo *logParis;

[logParis tempsPourDemain];
[logParis ajoutTemperature:13.0];
[logParis tempsMatin:@"beau" apresMidi:@"nuages"];
```

Morphologie d'un message


```
[logParis ajoutTemperature:13.0];
```

```
[logParis tempsMatin:@“Beau” apresMidi:@“Nuages”];
```

Morphologie d'un message

Message

```
[logParis ajoutTemperature:13.0];
```

```
[logParis tempsMatin:@“Beau” apresMidi:@“Nuages”];
```

Morphologie d'un message

un sélecteur

Message

```
[logParis ajoutTemperature:13.0];
```

un sélecteur

```
[logParis tempsMatin:@“Beau” apresMidi:@“Nuages”];
```

Morphologie d'un message

argument

Message

un sélecteur

```
[logParis ajoutTemperature:13.0];
```

un sélecteur

```
[logParis tempsMatin:@"Beau" apresMidi:@"Nuages"];
```

argument

Le «type» sélecteur

- Mécanisme de construction de sélecteurs

- Extrêmement utile pour référencer des méthodes

```
SEL sel1 = @selector(coucouLesGars);
SEL sel1 = @selector(totoEstContent:);
SEL sel2 = @selector(tempsMatin:ApresMidi:);
```

- Utile pour l'introspection aussi

```
- (BOOL) respondToSelector:(SEL)sel;
```

Premiers éléments utiles

- NSLog
- NSString

Générer une trace sur la console…

Instruction NSLog

- Prend une NSString en paramètre
- Concaténation/substitutions gérées par NSString (cf plus loin)

Exemple

```
NSLog(@"Toto est content");
```

Les chaînes de caractères

NSString plutôt que char*

- C'est une classe (support d'Unicode)
- Toujours utilisé…

Évolution de la syntaxe

- En C: "Hello World"
- En Objective-C: @"Hello World"

```
NSString *maChaine = @"Hello World";
```

Manipulation des NSString

- **Les NSString ne sont pas modifiables!!!**
  - Il existe les NSMutableString pour cela

```
NSString *maChaine = @"Hello World";

NSString *meteo = @"beau";
NSString *msg = [NSString stringWithFormat:@"Il fait %@", meteo];
```

```
NSLog(@"Le vent souffle à %d km/h : %d sur l'échelle de Beaufort",
      [logParis vent],
      [meteo equivalentBeaufort:[logParis vent]]);
```

```
NSString *myString = @"Hello";
NSString *fullString;
fullString = [myString stringByAppendingString:@" world!"];
```

- **Pour le reste...**

En guise de conclusion…

- Vous disposez (presque 😉) des bases pour construire votre première application en Objective-C

One more thing… Objective-C & Swift?

- **L'un va-t-il remplacer l'autre?**
  - Dans l'esprit d'Apple sans doute… mais c'est plus compliqué 
- **État des lieux**
  - Un existant colossal *(rappel, 900K applications!)*
    - **TIOBE Index (Objective-C): 2004 = 39ème — 2009 = 30ème — 2014 = 3ème (~10%)**
    - **TIOBE Index (Swift) : introduction en 18ème position (coté buzz?)**
  - Aaron Hillegass (Juin 2014)
    - **«les développeurs iOS ont besoin de connaître Objective-C»**
    - **Aspects patrimoine (iOS contient encore des couches en C)**
    - **Facilité de lecture?**
  - Il y aura sans doute une longue cohabitation des deux langages

- **Désolé, vous devez connaître les deux** 

UPMC
SORBONNE UNIVERSITÉS

Guide minimum de survie en Swift

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

En guise d'introduction…

Historique

- Conçu depuis 2010 (en secret)
- Lancé en fanfare le 2 juin 2014 (WWDC)
- Version 1.0 «stabilisée» le 9 septembre 2014

Nous aborderons…

- Principes du langage
- Principaux éléments de syntaxe
- Les chaînes de caractères
  - **Vous en aurez rapidement besoin**

Swift, concepts de base

- Classes et appel de méthodes (super basique)

Principes de Swift

- Langage Fonctionnel et Objet
- Intègre les dernières techniques de compilation
  - «Facile» (compact à écrire)
    - Plein d'implicite!
    - Tout est déduit de vos expressions
    - Sera-ce lisible?
  - Fortement typé (inférence de type)
- S'inspire d'Objective-C
  - Mais intègre certains mécanismes
    - ARC
- S'appuie sur les mêmes bibliothèques
  - Donc vous vous y retrouverez…

Méthodes de classes*

- Contrairement à Objective-C, un seul type de méthodes (sur les instances)

```
class Compteur {
    var compte: Int = 0
    func incremente() {
        compte++
    }
    func ajoute(n: Int) {
        compte += n
    }
    func afficheCompte() {
        println("Compte : \(compte)")
    }
    func val () -> Int {
        return compte
    }
}
```

Usage régulier dans les API d'iOS

- Plus sur les classe plus tard

* exemple tiré de http://www.swiftcode.fr/antiseche-swift/

Appel de méthodes

Syntaxe pointée (plus «classique»)

```
var monCompte = Compteur()
monCompte.incremente()
monCompte.ajoute(2)
monCompte.afficheCompte()
println(monCompte.val())
```

```
class Compteur {
    var compte: Int = 0
    func incremente() {
        compte++
    }
    func ajoute(n: Int) {
        compte += n
    }
    func afficheCompte() {
        println("Compte : \(compte)")
    }
    func val () -> Int {
        return compte
    }
}
```

Premiers éléments utiles

- print / println
- Gestion des chaînes de caractères

Générer une trace sur la console…

Instruction println et print

- Prend une chaîne de caractères en paramètre
- Concaténation/substitutions gérées en natif (cf plus loin)

Exemple

```
println ("Toto est content")
```

- Quelques observations…

Les chaînes de caractères

Le type string existe mais se déduit (merci le typage dynamique)

- Support d'Unicode intégral
- Comme pour toutes les variables, deux modes
  - **Immuable ou modifiable**

Exemple

- En C: "Hello World"
- En Objective-C: @"Hello World"

```
let immuable  = "Toto l'immobile"
var variable  = "Je peux changer"
let duchinois = "Hànzi 漢字"
```

Manipulation des chaînes de caractères

Les variables déclarées «let» ne sont pas modifiables

- Il faut les déclarer en «var»

```
let maChaine = "Hello World"

let meteo = "beau"
let msg = "Il fait "+meteo

var logParis = meteoclass()
println("Le vent souffre à \(logParis.vent) km/m: ",
        "\(logParis.equivalentBeaufort(logParis.vent))",
        " sur l'échelle de beaufort")

let myString = "Hello"
var fullString : String
fullString = myString+" World!"
```

Pour le reste...

En guise de conclusion…

- Vous disposez (presque 😉) des bases pour construire votre première application en Swift

UPMC
SORBONNE UNIVERSITÉS

Votre première application, «Bonrevoir»

Fabrice.Kordon@lip6.fr

Tutoriel IWAISE'2014

Tunis, 12 novembre 2014

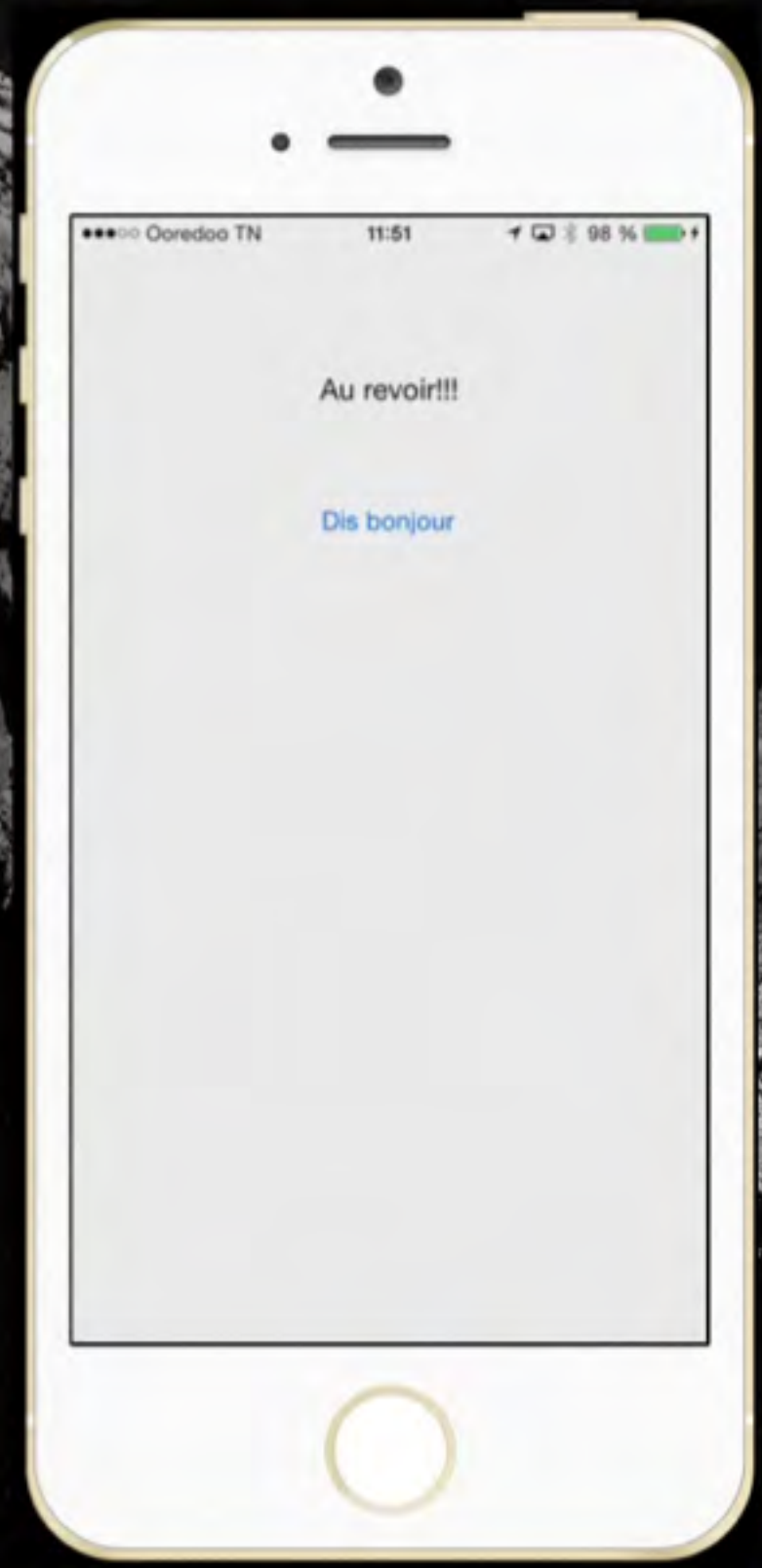
En guise d'introduction

Objectifs de l'exercice

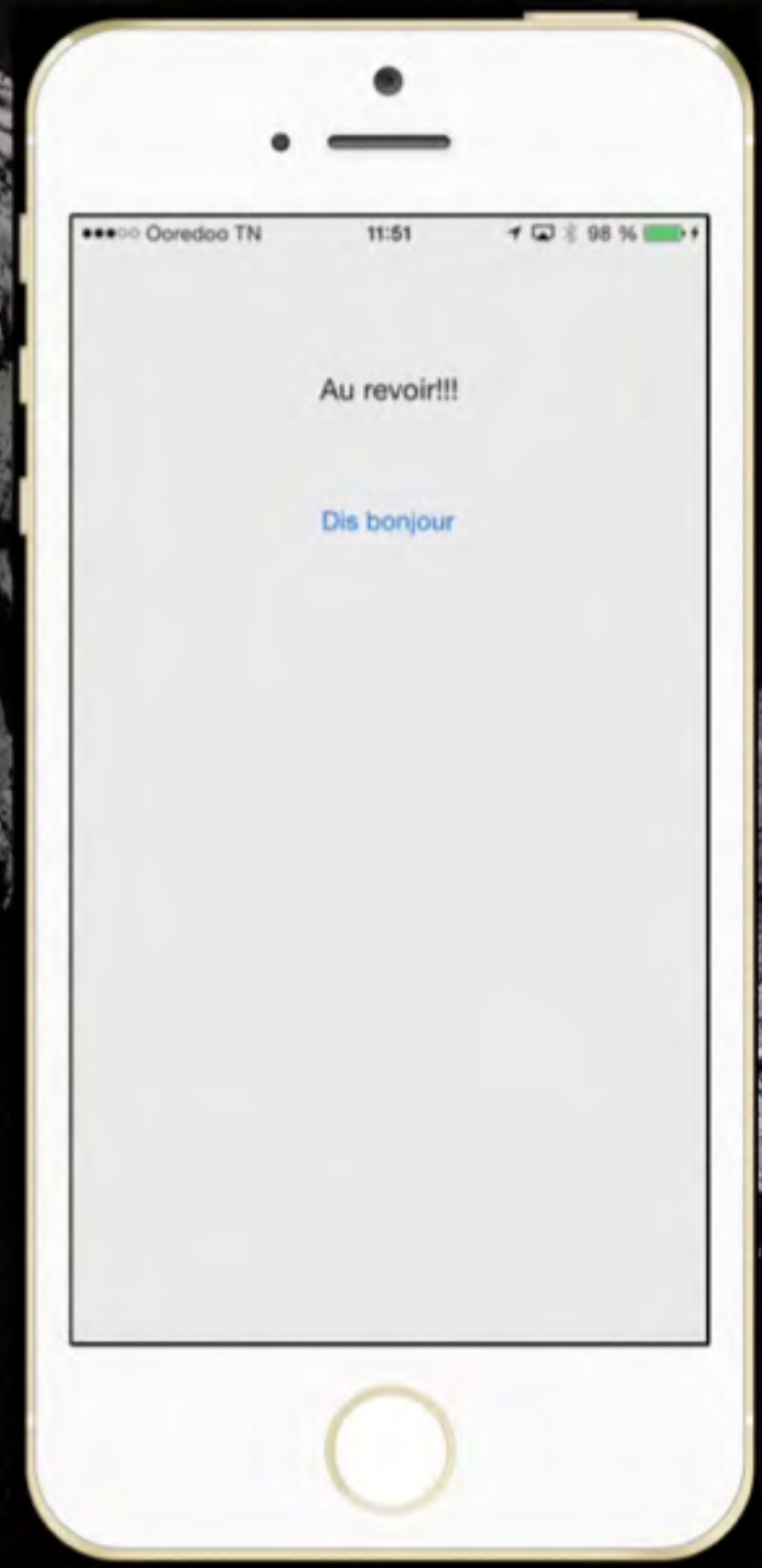
- Premier contact avec l'environnement
- Votre première application
- Jouer avec les interactions
  - **Bouton à la fois «action» et «outlet»**
- Jouer avec l'orientation
  - **Contraintes communes**

Bref, enfin «jouer avec l'environnement»

«BonRevoir» — petite démonstration



«BonRevoir» — petite démonstration



«BonRevoir» — et la rotation?

«BonRevoir» — et la rotation?

Quelques éléments utiles

Le Label (UILabel)

- Attribut «text»
  - Objective-C : NSString
  - Swift : chaîne de caractères

Le bouton (UIButton)

- Attribut «Title» de type UILabel
  - Il y a une «indirection» via le UILabel…

Difficulté

- Mettre à jour le texte du bouton
  - Objective C: setTitle: forState:
  - Swift: SetTitle (2 paramètres)

En guise de conclusion...

A votre tour de travailler

Si vous êtes courageux

faites le en Objective C et en Swift

UPMC
SORBONNE UNIVERSITÉS

Et pour ceux qui vont vite...
«Couleur»

Fabrice.Kordon@lip6.fr

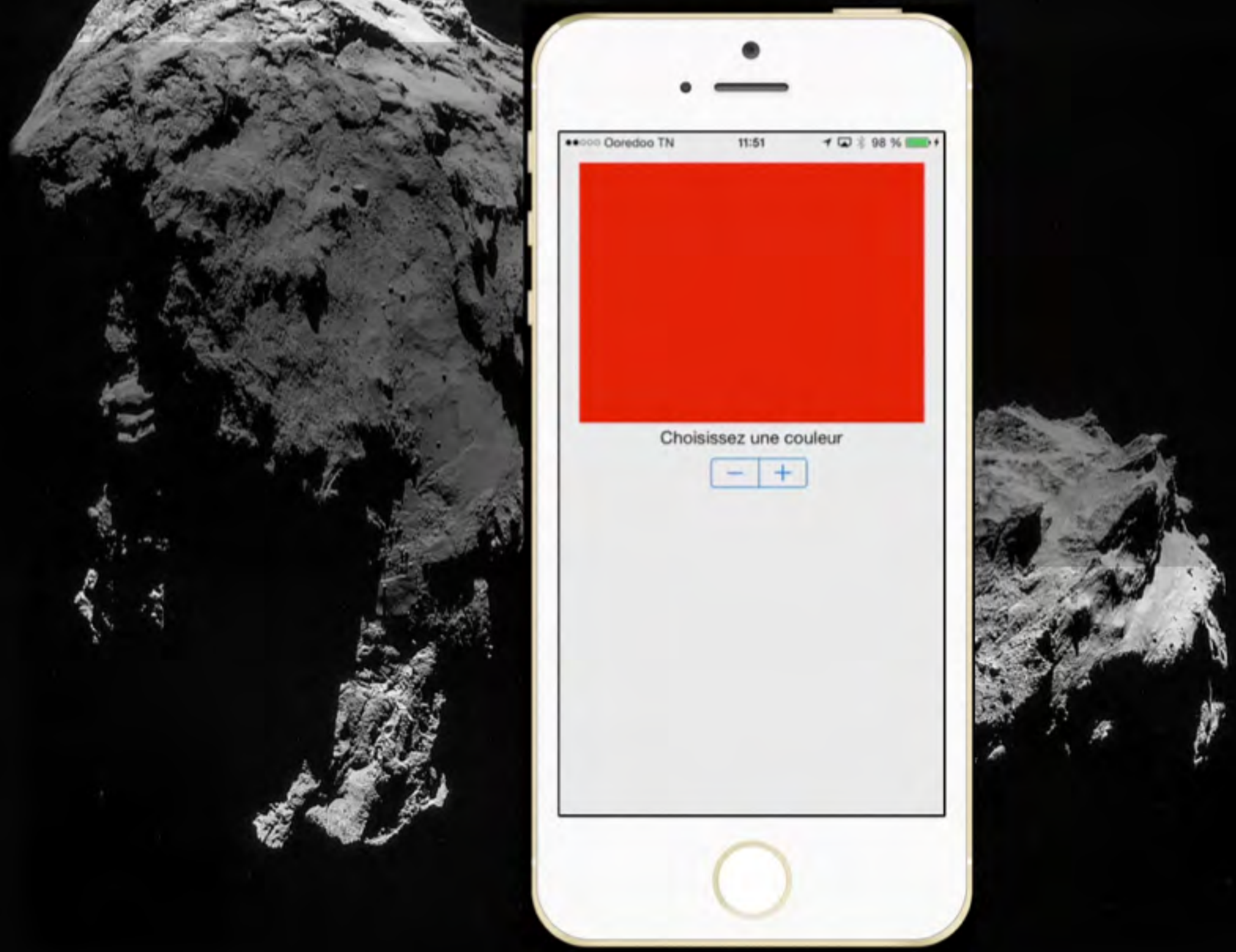
En guise d'introduction

Objectifs de l'exercice

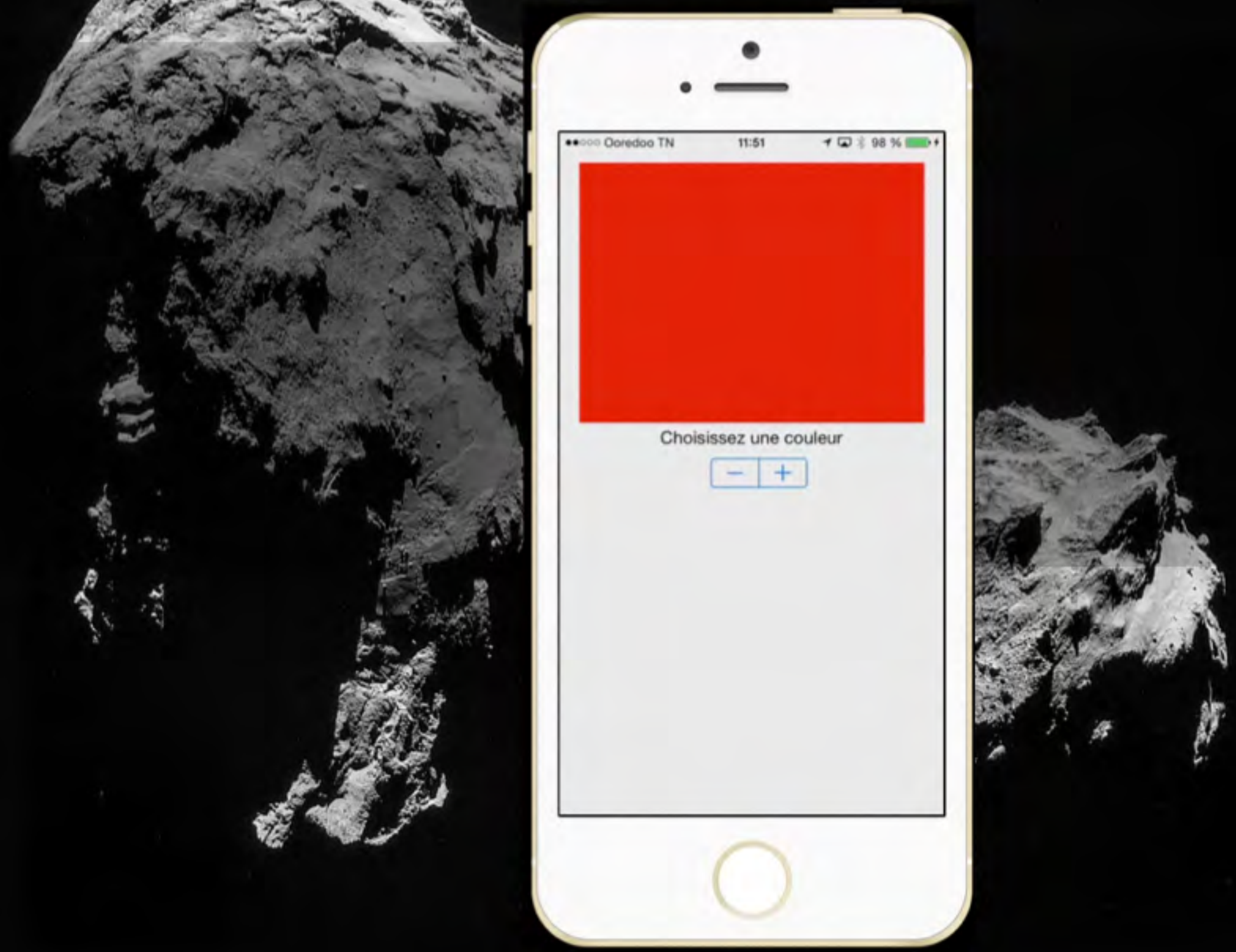
- Premier contact avec l'environnement
- Découverte d'un nouveau «widget» — le stepper
- Jouer avec les présentation
  - **Orientation différente entre vue portrait et vue paysage**

Bref, enfin «jouer avec l'environnement»

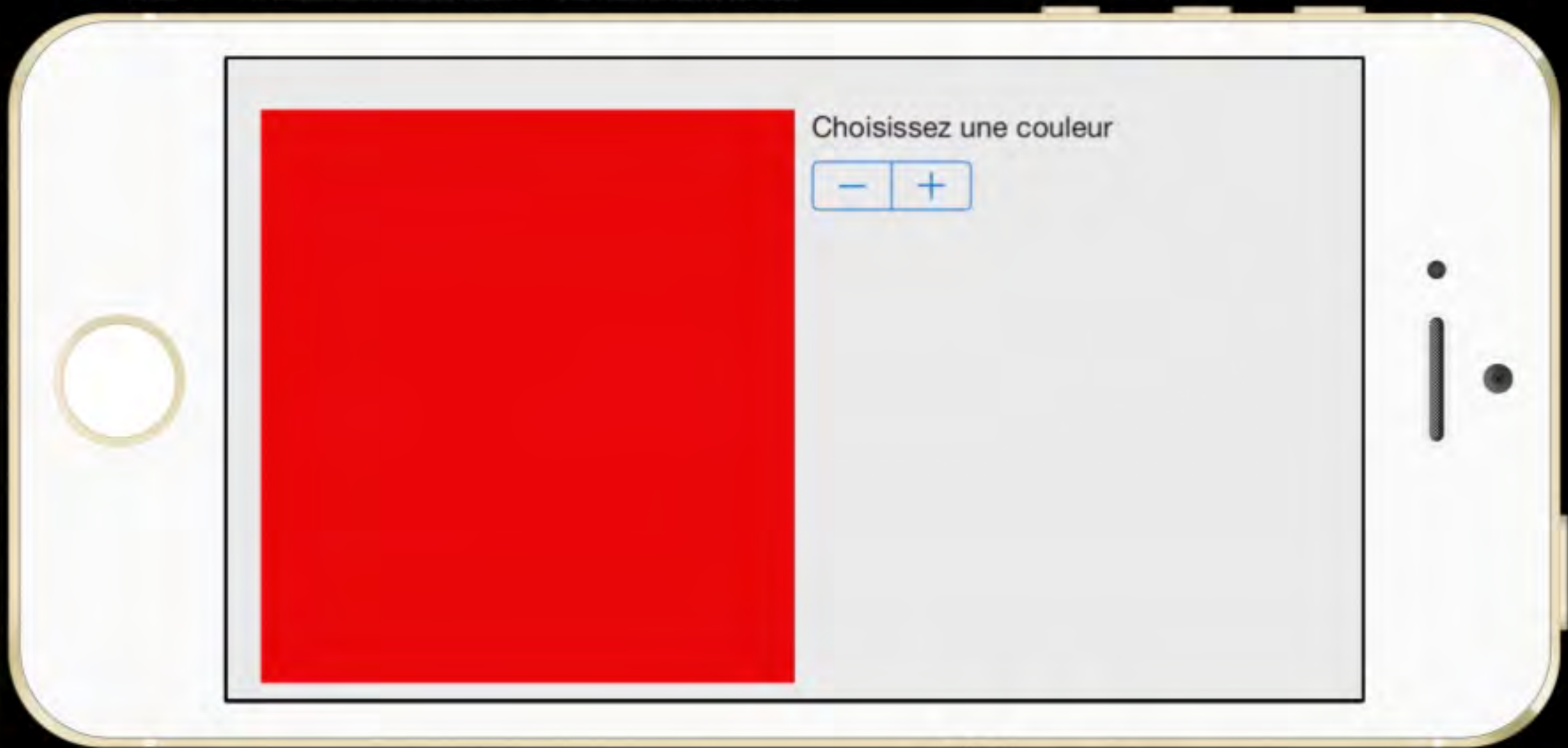
«Couleur» — petite démonstration



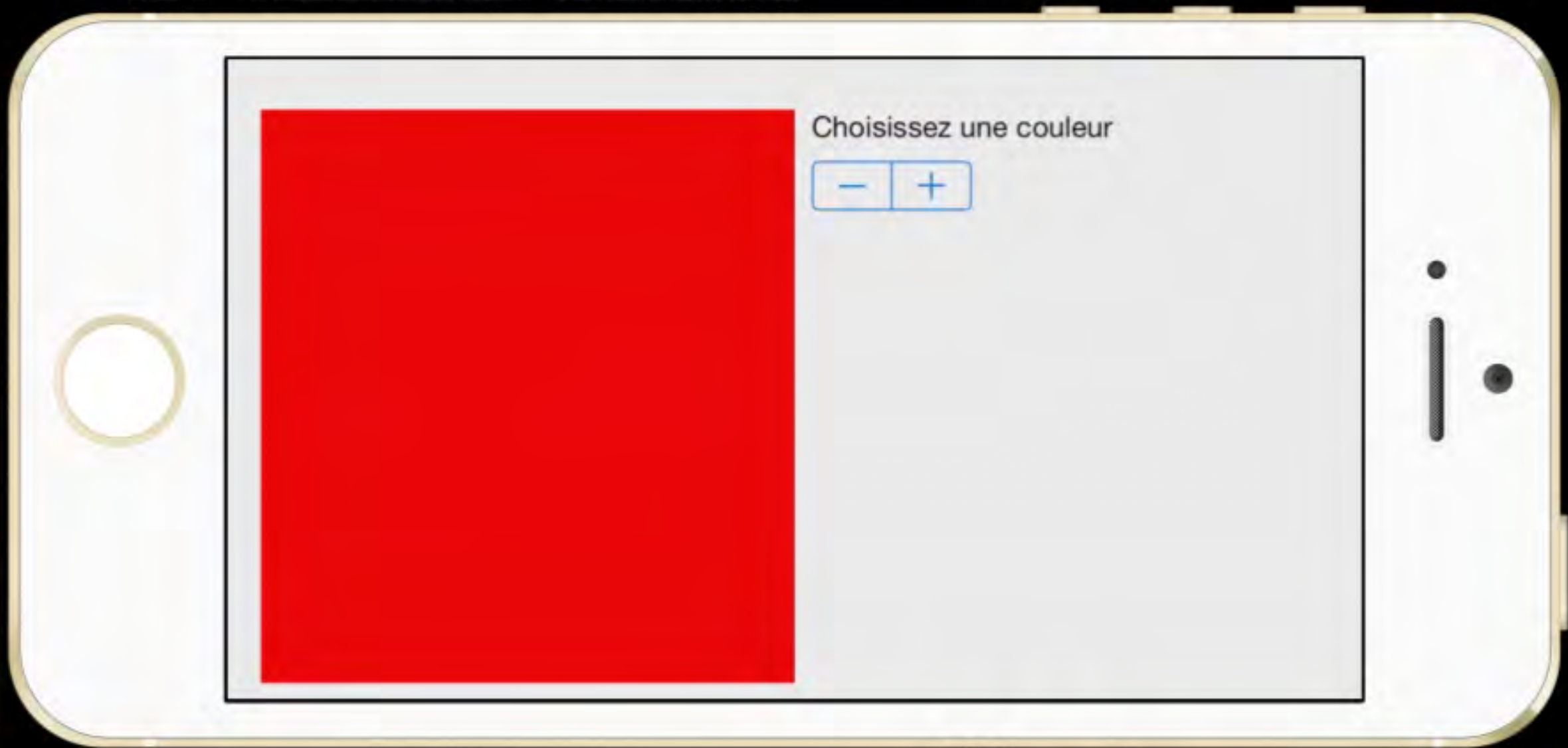
«Couleur» — petite démonstration



«Couleur» — et la rotation?



«Couleur» — et la rotation?



Quelques éléments utiles

Le «stepper» (UIStepper)

- Attribut «value» (double, nombre réel en double précision)
- Attributs «maximumValue» et «minimalValue»
- Attribut «continuous»

Une vue (UIView)

- Pour en utiliser le fond
- Attribut backgroundColor
  - **Objective-C : setBackgroundColor (valeur de type UIColor)**
  - **Swift : backgroundColor (valeur de type UIColor)**

Difficulté

- Changer la présentation en fonction de l'orientation!

En guise de conclusion...

A votre tour de travailler

Si vous êtes courageux

faites le en Objective C et en Swift