# Introduction

Une personne qui entreprend de devenir développeur iOS va affronter trois obstacles :

- Apprendre le langage de programmation Objective-C. Du fait qu'Objective-C est une extension simple du langage C, vous en aurez une connaissance suffisante une fois lus les quatre premiers chapitres de ce livre.
- Apprendre à maîtriser les concepts essentiels : classes et messages, techniques de gestion mémoire, délégation, archivage et contrôleurs de vues. Il faut quelques jours pour assimiler ces notions ; elles vous seront devenues familières lorsque vous aurez lu la moitié de ce livre.
- Apprendre à maîtriser les infrastructures. L'objectif ultime est de savoir utiliser chacune des méthodes de chacune des classes de chaque infrastructure liée à iOS. C'est un travail de longue haleine, puisqu'il y a plus de 3000 méthodes dans plus de 200 classes dans iOS. Sans compter avec le fait que Apple rajoute des classes et des méthodes dans chaque nouvelle version de iOS. Au long de ce livre, vous découvrirez chacun des sous-systèmes qui constituent le kit SDK de iOS, mais nous n'irons pas au fond de chaque sujet. Notre but est de vous guider jusqu'au niveau à partir duquel vous êtes autonome dans la recherche d'informations, notamment en naviguant dans la documentation de référence Apple.

Nous avons utilisé le contenu de ce livre à de nombreuses reprises pour les sessions de formation que nous prodiguons dans la série *iOS Development Bootcamp* chez Big Nerd Ranch. Ce contenu a donc été suffisamment confronté à la réalité et il a aidé de nombreuses personnes à devenir des développeurs d'applications iOS. Nous espérons sincèrement qu'il vous sera utile à vous aussi.

## Prérequis intellectuels

Nous supposons dans ce livre que vous êtes motivé à apprendre comment écrire des applications iOS. Nous ne perdrons donc pas de temps à vous convaincre que les matériels iPhone, iPad et iPod touch représentent des solutions technologiques très convaincantes.

Nous supposons aussi que vous connaissez le langage C et avez quelques notions de programmation orientée objet. Dans le cas contraire, il serait sans doute préférable de lire d'abord un ouvrage dédié à la programmation C et Objective-C, comme *Programmation Objective-C* publié dans la même collection.

## Notre pédagogie

Ce livre présente les concepts majeurs de la programmation iOS de façon pratique, puisque vous allé être amené à saisir beaucoup de code source et à réaliser une dizaine d'applications fonctionnelles. En fin de livre, vous aurez acquis des connaissances et de l'expérience. Ce livre prend soin de ne pas vous noyer sous les connaissances théoriques, piège dont nous avons tous appris à nous méfier. Nous avons en effet choisi de vous proposer une approche d'apprentissage par la pratique. Les concepts et le codage vont de pair. Voici ce que des années de formation à la programmation iOS nous ont appris :

- Nous avons pu dresser la liste des concepts essentiels qu'il faut avoir maîtrisé pour débuter en programmation, et nous nous limitons à cette liste restreinte.
- Nous avons appris que les gens apprennent mieux lorsque ces concepts sont abordés au moment exact où chacun d'eux devient nécessaire.
- Nous avons appris que pour garantir la meilleure efficacité, la connaissance théorique et l'expérience pratique de la programmation doivent progresser de concert.
- Nous avons appris que l'action concrète a encore plus d'importance que nous le supposions. Nous vous demanderons souvent de commencer à saisir des instructions avant que vous ayez eu l'occasion de savoir exactement à quoi elles servent. Vous pourriez avoir l'impression de travailler un peu à l'aveuglette. Pourtant, le meilleur moyen d'apprendre à coder consiste à coder et à corriger ses erreurs. Il ne s'agit nullement d'une perte de temps, bien au contraire : en commençant à déboguer dès le départ, vous plongez vite dans l'intimité du code source. Voilà pourquoi nous vous invitons à saisir le code source bien que vous puissiez récupérer l'archive de tous les exemples disponibles sur le site de l'éditeur et faire des copier/coller. Copier/coller n'est pas programmer. Nous avons de plus hautes ambitions pour vous.

Que devez-vous en conclure, cher lecteur ? Qu'il faut nous faire confiance pour adopter cette approche. Nous vous en sommes reconnaissants par avance. Notez qu'il faut aussi un peu de patience. Pendant la progression, nous veillerons à ce que vous soyez toujours à l'aise et informé de ce que nous sommes en train de préparer. Mais de temps à autre, il faudra vraiment nous laisser vous guider. Si vous craignez que cela risque de vous gêner, persévérez. Ne vous découragez pas devant un concept étrange. Souvenez-vous à ce moment que nous avons choisi de ne pas vous forcer à digérer tous les concepts d'un coup. En cas de doute, sachez que le concept sera sans doute présenté un peu plus loin, quand ce sera devenu nécessaire. Et certains points moins évidents vont s'éclairer d'eux-mêmes lorsque vous allez les mettre en pratique la première (ou la douzième) fois.

Tout le monde n'apprend pas de la même façon. Peut-être serez vous de ceux qui apprécieront notre manière de n'introduire les concepts que lorsque c'est devenu nécessaire. Mais si vous êtes de ceux que cela frustre, voici quelques conseils :

- Prenez une grande inspiration et patientez. Nous l'aborderons ce fameux concept, chemin faisant.
- Consultez l'index. Nous tolérons que vous sautiez directement plus loin dans le livre.
- Consultez la documentation Apple, un outil incontournable que vous devez prendre l'habitude savoir explorer.[1]

Si ce sont les concepts du langage Objective-C ou de la programmation objet qui vous inquiètent, tournez-vous vers un livre dédié à ce sujet, comme déjà mentionné plus haut.

# Modalités de lecture

Ce livre se fonde sur les formations que nous dispensons chez Big Nerd Ranch. Il a donc été structuré pour être assimilé d'une certaine manière.

1. N.d.T. : les lecteurs francophones verront que la documentation Apple n'est disponible qu'en anglais.

Fixez-vous des objectifs raisonnables comme "Je vais lire un chapitre par jour." Au moment de vous installer pour commencer, arrangez-vous pour être au calme pendant au moins une heure. Fermez votre messagerie, votre client Twitter et votre logiciel de tchatche. Ce n'est pas le moment de jongler entre plusieurs activités ; vous devez vous concentrer.

Pratiquez. Vous pouvez parcourir chaque chapitre à blanc d'abord si vous le voulez, mais l'apprentissage véritable réclame d'être assis et de rédiger du code source. Vous ne comprendrez bien un concept qu'après avoir écrit et, surtout, fait fonctionner un programme qui l'incorpore.

Certains projets d'exemple requièrent des fichiers complémentaires. Le premier chapitre utilise par exemple un fichier d'icône pour le projet Quiz. Vous trouverez ce fichier dans l'archive des exemples, dans le dossier des ressources. L'archive de la version américaine est à cette adresse :

**http://www.bignerdranch.com/solutions/iOSProgramming3ed.zip**

L'archive de la version française est à cette adresse :

**http://www.pearson.fr/livre/?GCOI=27440100695050**

Il existe deux grandes manières d'apprendre. Quand vous abordez la Révolution française, vous ajoutez des connaissances à tout un échafaudage de savoirs. C'est un apprentissage "facile", car même si cela peut prendre des années, vous ne tomberez jamais dans une situation de perplexité. Par contre, apprendre à programmer sous iOS est "difficile", car vous devez vous préparer à être parfois décontenancé, notamment au début. Nous avons pris soin dans ce livre de prévenir les anicroches de la phase d'apprentissage. Voici deux derniers conseils pour vous aider dans votre progression :

- Trouvez un collègue ou ami qui sait déjà écrire des applications iOS et peut répondre à vos questions. Il est particulièrement difficile de réussir dès la première fois à implanter son application sur un appareil réel si vous ne pouvez pas profiter de l'expérience d'un autre développeur.
- Dormez suffisamment. Une personne fatiguée ne mémorise pas bien ce qu'elle tente d'apprendre.

# Structure du livre

Chaque chapitre du livre aborde un ou plusieurs concepts du développement iOS grâce à un exemple concret. En fin de chapitre, nous vous proposons des défis à relever, pour poursuivre la pratique du codage. Relevez-en quelques-uns au moins, car ils permettent de consolider les connaissances acquises au cours du chapitre. Presque tous les chapitres offrent aussi une ou deux sections "Pour les plus curieux" qui approfondissent certains concepts du chapitre.

Le Chapitre 1 présente la programmation iOS en vous guidant au long de la création d'une petite application nommée Quiz. Cela vous donne l'occasion de découvrir l'atelier Xcode, le simulateur iOS, la création de projet et les fichiers associés. Le chapitre contient une description de la structure MVC (Modèle-Vue-Contrôleur) et son intérêt en développement iOS.

Les Chapitres 2 et 3 sont un peu plus théoriques car ils ne sont pas illustrés par la création d'une application complète. Nous y présentons les fondamentaux du langage Objective-C et de la gestion mémoire. En revanche, nous en profitons pour démarrer un nouveau projet nommé RandomPossessions qui sera poursuivi dans la suite.

Dans les Chapitres 4 et 5, vous découvrez la géolocalisation et l'affichage de cartes avec les infrastructures Core Location et MapKit en créant une application de cartographie nommée Whereami. Vous allez aussi expérimenter le concept essentiel de délégation et découvrir les protocoles, les infrastructures (frameworks), les diagrammes d'objets, le débogueur et la documentation Apple.

Les Chapitres 6 et 7 se concentrent sur l'interface utilisateur iOS en créant les deux projets liés Hypnosister et HypnoTime. Vous apprendrez concrètement à utiliser les vues, les contrôleurs, le défilement, le zoom et la navigation entre écrans avec une barre d'onglets.

Dans le Chapitre 8, nous créons une autre application de petite envergure nommée HeavyRotation pour découvrir les notifications et le mécanisme d'autorotation. Nous verrons aussi les retailles automatiques (*autoresizing*) qui permettent au projet **`HeavyRotation`** de se rendre compatible iPad ainsi que la nouvelle option d'iOS6 Use Autolayout.

Le Chapitre 9 est le premier d'une longue série au cours de laquelle nous allons concevoir une vaste application : Homepwner. (Ce n'est pas une faute de frappe anglaise, comme nous l'expliquerons.) Cette application gère la liste de tous les objets de valeur que vous possédez et qu'il faudrait communiquer à votre assureur en cas de catastrophe. Homepwner va vous accompagner jusqu'au Chapitre 17.

Vous allez construire des tables et des vues tables dans les Chapitres 9, 10 et 15 en utilisant des contrôleurs de vues et des sources de données. Vous verrez comment afficher des données tabulaires dans une vue, permettre à l'utilisateur de modifier les données et embellir l'interface.

Le Chapitre 11 profite de l'expérience acquise dans le Chapitre 7 au niveau de la navigation. Vous découvrirez la classe **`UINavigationController`** et doterez le projet Homepwner d'une capacité à descendre dans les détails (*drill-down*) et d'une barre de navigation.

Dans le Chapitre 12, vous apprenez à prendre des photos, à les afficher et à les sauvegarder, en ayant recours aux classes **`NSDictionary`** et **`UIImagePickerController`**.

Le Chapitre 13 présente le contrôleur de surfenêtre **`UIPopoverController`** destiné à l'iPad et les contrôleurs de vues modaux. Nous en profitons pour rendre le projet Homepwner universel (pouvant fonctionner tant sur iPhone que sur iPad).

Le Chapitre 14 s'intéresse à la sauvegarde et au rechargement des données. Nous verrons comment archiver les données du projet Homepwner grâce au protocole `NSCoding`. Nous expliquons aussi les états applicatifs (actif, suspendu, arrière-plan) et les transitions entre états.

Le Chapitre 16 constitue une introduction à Core Data, ce qui permettra de sauvegarder les données du projet Homepwner avec un objet **`NSManagedObjectContext`**.

Dans le Chapitre 17, vous découvrirez les concepts et techniques de l'internationalisation et de la localisation avec la classe **`NSLocale`**, les tables de chaînes et **`NSBundle`**. Ce chapitre celui dans lequel vous terminerez le projet Homepwner.

Dans le Chapitre 18, vous utiliserez **`NSUserDefaults`** pour sauvegarder les préférences utilisateur.

Les Chapitres 19 et 20 proposent de créer une application de dessin nommée TouchTracker afin d'expérimenter les événements de type toucher. Vous verrez comment ajouter des capacités de détection multi-touch et comment vous servir de **`UIGestureRecognizer`** pour gérer des gestes spécifiques. Vous aborderez à cette occasion les notions de premier répondeur et de chaîne de répondeurs tout en approfondissant vos connaissances sur **`NSDictionary`**.

Dans le Chapitre 21, vous verrez comment profiter de l'outil d'analyse de code INSTRUMENTS qui permet d'optimiser les performances de vos applications. Le chapitre présente aussi les schèmes Xcode et l'analyseur statique.

Les Chapitres 22 et 23 présente les calques et l'infrastructure Core Animation en ajoutant des animations à l'application HypnoTime du Chapitre 7. Vous y verrez les animations interpolantes et les objets d'animation tels que **`CABasicAnimation`** et **`CAKeyframeAnimation`**.

Le Chapitre 24 présente une nouveauté de iOS qui permet de construire une application sans écrire quasiment de code : le storyboard. Dans le projet d'exemple nommé StoryTime, vous allez utiliser la classe **`UIStoryboard`** et en savoir plus sur les avantages et inconvénients des storyboards dans la construction d'applications.

Avec le Chapitre 25, vous plongez dans le vaste monde des services web en commençant la création de l'application Nerdfeed qui va interroger un flux d'actualités RSS sur un serveur grâce aux classes **`NSURLConnection`** et **`NSXMLParser`**. Le projet va afficher une page web dans un objet **`UIWebView`**.

Dans le Chapitre 26, vous découvrirez le contrôleur à vues de division (bivolet) **`UISplitView-Controller`** pour doter le projet Nerdfeed d'une interface à deux volets afin de tirer profit de la vaste surface d'affichage des iPad.

Le Chapitre 27 présente les avantages et modalités d'emploi des blocs de code qui deviennent de plus en plus précieux dans le SDK de iOS. Vous allez créer une petite application de test nommée Blocky, ce qui va vous préparer à incorporer des blocs de code dans le projet Nerdfeed dès le chapitre suivant.

Dans les Chapitres 28 et 29, vous allez remettre en cause l'architecture de gestion de données du projet Nerdfeed en sorte de lui faire adopter l'approche de programmation MVCS (Modèle-Vue-Contrôleur-Stock). Vous découvrirez la logique de requête web et verrez comment bien concevoir une application qui doit communiquer avec une sources de données externe.

Enfin, le Chapitre 30 vous montre comment adapter une application pour lui faire tirer profit du mécanisme iCloud de stockage en ligne afin de synchroniser et sauvegarder des données utilisables depuis plusieurs appareils sous iOS.

# Conventions de programmation

Ce livre contient beaucoup de code. Nous avons cherché à faire en sorte que sa rédaction et la conception sous-jacente soient exemplaires en suivant tant que faire se peut les bonnes pratiques en vigueur. Pourtant, nous nous sommes parfois écartés des habitudes d'écriture que vous pouvez voir dans les exemples de code Apple ou dans d'autres livres. Ces différences ne vous diront sans doute rien pour l'instant, mais nous pensons qu'il est indispensable de les décrire avant de vous laisser commencer la lecture du premier chapitre :

- Il existe une seconde manière d'écrire les appels aux méthodes d'accès qui se nomme la *syntaxe point*. Nous expliquerons comment elle s'écrit, mais nous ne l'utiliserons pas dans ce livre. Pour nous, comme pour les débutants, cette syntaxe point a tendance à réduire la lisibilité fonctionnelle du code source.
- Dans nos sous-classes dérivées de **`UIViewController`**, nous faisons toujours de **`init`** l'initialisateur désigné. Selon nous, le créateur d'une instance de la classe n'a pas à connaître le nom du fichier XIB utilisé par un contrôleur de vue, ni même de savoir qu'il en utilise un.

- Nous créons systématiquement nos instances de contrôleurs de vues par instructions et non comme le font certains programmeurs, par un fichier XIB. Nous considérons que cette technique visuelle rend le code source résultant difficile à lire et à mettre au point.
- Nous démarrons quasiment tous nos projets à partir du modèle le plus simple, celui de l'application vide (EMPTY). Les autres modèles de projets ajoutent quantité de lignes d'instructions préprogrammées que vous devrez en général commencer par supprimer. De plus, cela va à l'encontre de notre règle précédente. Nous considérons qu'ils représentent une mauvaise solution pour démarrer vos projets dans les meilleures conditions.

Nous sommes persuadés qu'en suivant ces règles, le code produit sera plus lisible et plus simple à déboguer. Lorsque vous aurez terminé l'étude de ce livre (et donc travaillé selon nos préconisations), vous pourrez faire des essais pour voir s'il y a un quelconque avantage à vous écarter de nos règles.

# Conventions typographiques

Pour augmenter la lisibilité de ce livre, nous avons défini des conventions de présentation et de mise en valeur des éléments remarquables dans le texte.

Les noms de classes, d'objets, de méthodes, de fonctions et de messages sont imprimés en police **`Courier gras`**. Les noms de classes et d'objets commencent par une majuscules, ceux des méthodes et des messages par une minuscule. Voici un exemple : "Dans le corps de la méthode **`loadView`** de la classe **`RexViewController`**, ajoutez un appel à la fonction **`NSLog`** pour afficher la valeur sur la console."

Les données membres, variables, constantes et types de données sont imprimés dans la même police `Courier`, mais pas en gras, comme ceci : "La variable `fido` est du type `float`. Vous lui donnez comme valeur initiale `M_PI`."

Les noms des commandes et options de menu sont imprimées en PETITES CAPITALES, comme dans cet exemple : "Démarrez Xcode et ouvrez le menu FILE pour choisir NEW puis PROJECT."

Les noms de fichiers, de dossiers et de projets adoptent une police sans-serif, comme dans "Enregistrez le fichier sous le nom essai.txt."

Les extraits et blocs de code source sont bien sûr toujours imprimés en police Courier. Les instructions nouvelles que vous devez saisir au clavier apparaissent en gras et les lignes existantes à supprimer sont imprimées en style biffé. Dans l'extrait suivant, les quatre premières lignes sont rappelées pour vous repérer : vous devez supprimer la cinquième puis saisir les deux suivantes avant les trois suivantes déjà présentes.

```
@interface QuizAppDelegate : NSObject <UIApplicationDelegate> { // Lignes
  int currentQuestionIndex;                                     // existantes
  // Objets du modèle                                           // à
  NSMutableArray *questions;                                    // conserver
  NSMutableArray *answers;                  // Ligne à supprimer

  // Les objets vues                        // Ligne à ajouter
  IBOutlet UILabel *questionField;          // Ajouter
  IBOutlet UILabel *answerField;            // Ligne existante
  UIWindow *window;                         // etc.
}
```

# Prérequis techniques

Il vous faut un ordinateur Mac Intel assez récent pour développer des applications iOS. Vous commencerez par récupérer et installer le kit de développement iOS SDK depuis le site Apple, qui comprend l'atelier de production Xcode (environnement de développement intégré, EDI), le simulateur iOS et quelques autres outils indispensables. Ces outils sont gratuits moyennant la création d'un compte de développeur.

Il est fortement conseillé de devenir membre du programme *iOS Developer Program* (80 euros/an) pour les trois raisons suivantes :

- Les membres ont accès en premier aux dernières versions des outils.
- Il est impossible d'implanter une application sur un appareil réel si vous n'avez pas signé cette application, ce qui n'est possible que si vous êtes membre. Vous devrez donc le devenir pour pouvoir tester vos projets autrement que dans le simulateur.
- Il faut être membre pour pouvoir publier une application dans la boutique Apple Store.

Si vous avez décidé de consacrer de votre temps pour étudier ce livre, la question de devenir membre du *iOS Developer Program* ne se pose même pas. Foncez visiter la page **http://developer.apple.com/programs/ios/**.

De quel appareil sous iOS devez-vous disposer ? Les projets de la première moitié du livre sont destinés aux iPhone, mais pourront bien sûr fonctionner sur iPad dans une petite fenêtre de la taille de celle d'un iPhone. Ce n'est pas optimal, mais nous sommes à ce moment en train d'apprendre iOS et de découvrir les fondamentaux du SDK qui ne varient pas d'un type d'appareil à l'autre. En revanche, dans la suite du livre, nous découvrons des options spécifiques aux iPad et nous verrons enfin comment rendre une application universelle (capable de s'adapter à une famille de machines iOS ou l'autre).

Vous voici mis en appétit ? Parfait. C'est parti !

# Notes de la version française

Cette édition française se base sur la troisième édition anglaise qui est parue avant la publication de iOS 6. Nous avons donc procédé en accord avec les auteurs américains aux retouches indispensables suites aux évolutions décrites dans la section suivante.

## iOS 5, iOS 6 et XCode 4.5

Rappelons que iOS 5 a introduit le très important mécanisme de gestion automatique des références mémoire ARC (*Automatic Reference Counting*) qui est actif par défaut sous iOS.

Parmi les évolutions iOS 6, celles qui ont un impact sur le contenu de ce livre sont peu nombreuses et faciles à délimiter :

- Une option Use Autolayout dans le volet File Inspector, active par défaut. Sous son effet, vous ne pouvez plus décider manuellement des contraintes de retaille des vues par les éléments *struts* et *springs* dans le volet Size Inspector. Décochez cette option pour pouvoir piloter les retailles à l'ancienne manière.
- Une gestion allégée de la libération mémoire des vues qui quittent l'écran (messages **`viewWillUnload`** et **`viewDidUnload`**). Nous donnons des précisions dans l'Annexe.

- Une évolution du mécanisme de gestion des autorotations de vues, mais la compatibilité permet d'utiliser l'approche en vigueur.

## Fichier archive des exemples

En programmation, toute autre langue que l'anglais possède un avantage pédagogique : les identifiants définis par le programmeur se distinguent aisément des invariants imposés par le langage, car ils sont traduits. Le seul problème est que cette version localisée ne peut plus être aisément synchronisée en cas d'évolution importante des codes sources originaux.

C'est pourquoi seul le premier chapitre a été localisé en français (projet QuizF). Cet exercice suffit à distinguer les éléments qui sont définis librement par le programmeur de ceux imposés par le langage.

Vous pourrez ainsi télécharger d'éventuelles mises à jour des exemples américains sur le site **www.bignerdranch.com**, par exemple à l'occasion de la sortie de iOS 7, ou sur celui de l'éditeur français (**www.pearson.fr**). La version anglaise (projet Quiz) du Chapitre 1 est néanmoins fournie dans le même fichier archive.

Prenez le temps de prendre connaissance du contenu du fichier LISEZMOI.txt présent dans la racine du fichier archive.

## Configuration matérielle

Il vous faut une machine sous Mac OS X suffisamment récente pour accepter la mise à jour vers la version du système minimale requise par la version de l'atelier Xcode que vous désirez installer.

Prenons comme exemple votre objectif de développer pour iOS 6.

- Pour créer des projets iOS 6, il faut au minimum Xcode 4.5.
- Pour installer Xcode 4.5, il faut au minimum le système Mac OS X 10.7 Lion.
- Pour installer ou mettre à jour vers Lion, il vous faut une machine assez récente, bien sûr animée par un processeur Intel Core 2 Duo au minimum. Selon la gamme Mac qui vous concerne (iMac, MacBook, etc.), l'âge maximal de la machine pouvant être mise à jour vers Lion ou Mountain Lion (10.8) diffère. A l'heure où nous écrivons ces lignes, la période charnière se situe entre 2008 et 2009.