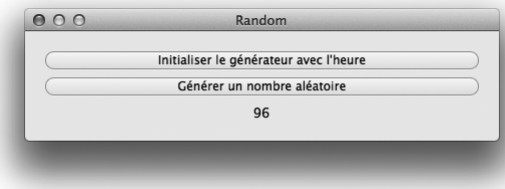
2

Premiers pas

De nombreux ouvrages débutent par des aspects philosophiques. Si nous commencions ainsi, du papier précieux serait gaspillé. À la place, ce chapitre va vous guider tout au long de l'écriture de votre première application Cocoa. Une fois celle-ci terminée, vous serez enthousiasmé et confus, mais prêt pour la philosophie.

Le premier projet est une application qui fait office de générateur de nombres aléatoires. Elle possède deux boutons : Initialiser le générateur avec l'heure et Générer un nombre aléatoire. Une zone de texte affiche le nombre généré. Pour ce simple exemple, nous devrons déterminer le bouton sur lequel l'utilisateur clique et générer une sortie. Il est possible que les explications des manipulations demandées, ainsi que leurs raisons d'être, vous semblent un tantinet vagues. Ne vous inquiétez pas. Tous les détails arriveront à point nommé. Pour le moment, nous allons simplement jouer un peu. La Figure 2.1 présente l'application terminée.

Figure 2.1
L'application terminée.



Dans Xcode

Si les outils du développeur ont été installés, vous trouverez Xcode dans le répertoire /Applications/. Faites glisser l'application sur le Dock, car vous allez devoir la lancer très souvent. Lancez Xcode. (Si vous n'avez encore jamais lancé Xcode, vous arrivez sur une page d'accueil. Conservez les valeurs par défaut et validez.)

Nous l'avons mentionné précédemment, Xcode conserve toutes les ressources qui composent votre application. Elles sont placées dans le *répertoire du projet*. La première étape du développement d'une nouvelle application consiste à créer un nouveau répertoire de projet, avec le squelette par défaut d'une application.

Créer un nouveau projet

Dans le menu File, choisissez New > Project… Dans la fenêtre qui apparaît (voir Figure 2.2), choisissez le type de projet que vous souhaitez créer : Cocoa Application. Vous noterez que bien d'autres types de projets sont également disponibles.

Figure 2.2
Choisir le type du projet.

Voici les principaux types de projets que nous utiliserons dans cet ouvrage :

- **Application.** Ce programme crée des fenêtres.
- **Tool.** Un programme qui ne possède pas d'interface utilisateur graphique. En général, un outil est un utilitaire en ligne de commande ou un démon qui s'exécute en arrière-plan.

- **Bundle ou framework.** Il s'agit d'un répertoire de ressources qui peuvent être utilisées dans une application ou un outil. Un bundle, également appelé *plug-in*, est chargé dynamiquement au moment de l'exécution. En général, une application se lie à un framework au moment de la compilation.

Pour le nom du projet, saisissez **`Random`** (voir Figure 2.3). En général, chaque mot qui compose le nom d'une application commence par une lettre majuscule. Fixez Class Prefix à Random et décochez les cases Create Document-Based Application, Use Core Data et Include Unit Tests. Vérifiez que la case Use Automatic Reference Counting est cochée. Pour tous les projets de cet ouvrage, nous utiliserons cette configuration.

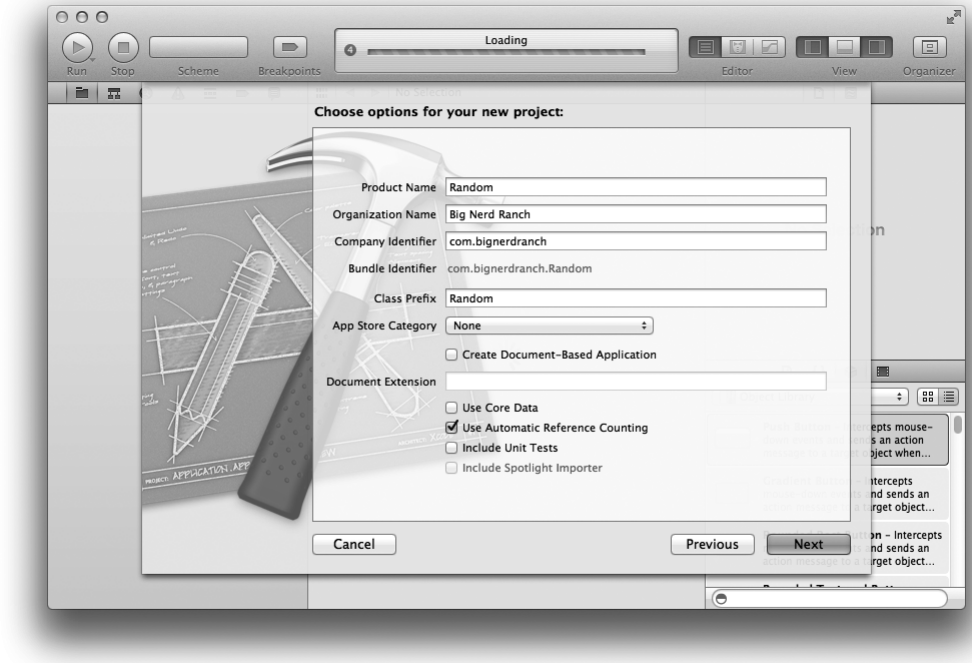


Figure 2.3

Nommer le projet.

Indiquez ensuite où sera créé le répertoire du projet. Par défaut, il s'agit de votre dossier de départ. Décochez la case Create local git repository for this project. Cliquez sur le bouton Create.

Le répertoire du projet, contenant le squelette d'une application, est créé automatiquement. À partir de ce squelette, nous allons créer le code source d'une application complète, puis compiler ce code source en application opérationnelle.

En examinant le nouveau projet dans Xcode, nous en obtenons une vue générale sur le côté gauche de la fenêtre. Chaque élément de cette vue représente un fichier du projet. Cet onglet du navigateur représente le projet ; d'autres onglets affichent des informations comme les erreurs de compilation ou les résultats d'une recherche. Nous allons modifier certains fichiers. Développez l'élément intitulé Random pour afficher la liste des fichiers qui seront compilés afin de créer une application.

Le squelette du projet qui a été créé peut être compilé et exécuté. L'application résultante est constituée d'un menu et d'une fenêtre. Pour compiler et exécuter le projet, cliquez sur l'icône de la barre d'outils libellée Run (voir Figure 2.4).

Figure 2.4
Squelette d'un projet.

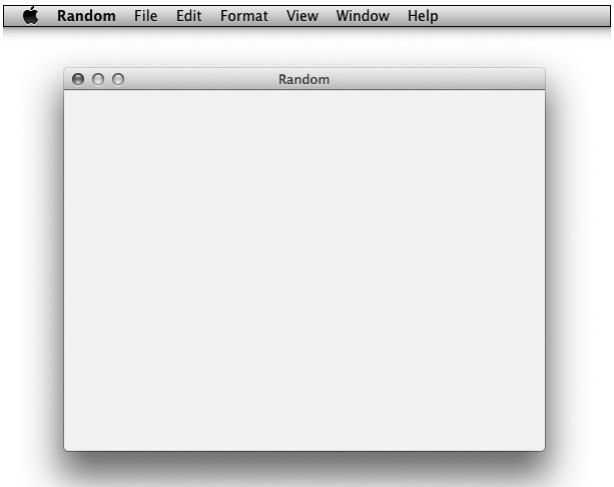
Pendant que l'application se lance, une icône sautille dans le Dock. Le nom de l'application apparaît dans le menu. Cela signifie qu'elle est active. Sa fenêtre peut être masquée par celle d'une autre application. Si vous ne la voyez pas, choisissez Hide Others dans le menu Random. Vous devez obtenir une fenêtre vide (voir Figure 2.5).

Bien qu'elle ne fasse pas grand-chose, l'application n'en est pas moins opérationnelle. Même l'impression fonctionne. Son code ne comprend qu'une seule ligne ; examinons-la. Quittez Random et revenez dans Xcode.

La fonction *main*

Développez le dossier Supporting Files et sélectionnez main.m en cliquant simplement dessus. Le code apparaît dans l'éditeur (voir Figure 2.6). Si vous double-cliquez sur le nom du fichier, il s'ouvrira dans une nouvelle fenêtre. En raison du grand nombre de fichiers manipulés dans une journée, toutes ces fenêtres qui s'ouvrent peuvent devenir rapidement pénibles. Il est parfois préférable de travailler dans une seule fenêtre.

Figure 2.5
Exécuter le projet.



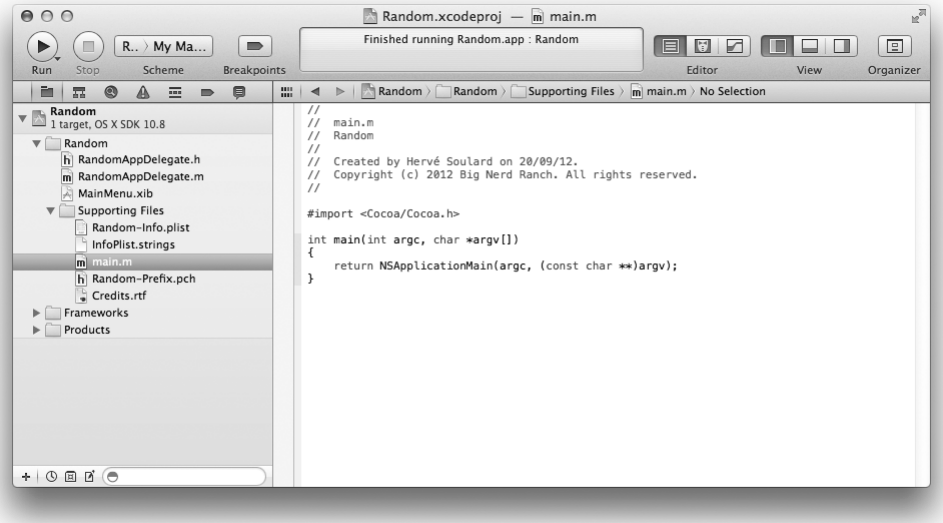


Figure 2.6
La fonction `main()`.

En règle générale, nous ne modifions pas le fichier main.m dans un projet d'application. La fonction **`main()`** par défaut appelle simplement **`NSApplicationMain()`**, qui, à son tour, charge les objets qui constituent l'application. La section suivante explique comment **`NSApplicationMain()`** connaît les objets qu'elle doit charger.

Dans Interface Builder

Dans le navigateur de projet, sous RANDOM, vous trouvez le fichier nommé MainMenu.xib. Cliquez dessus pour l'ouvrir dans l'éditeur Interface Builder. Ensuite, dans la barre d'outils, cliquez sur la vue UTILITIES de façon à afficher le panneau droit (voir Figure 2.7).

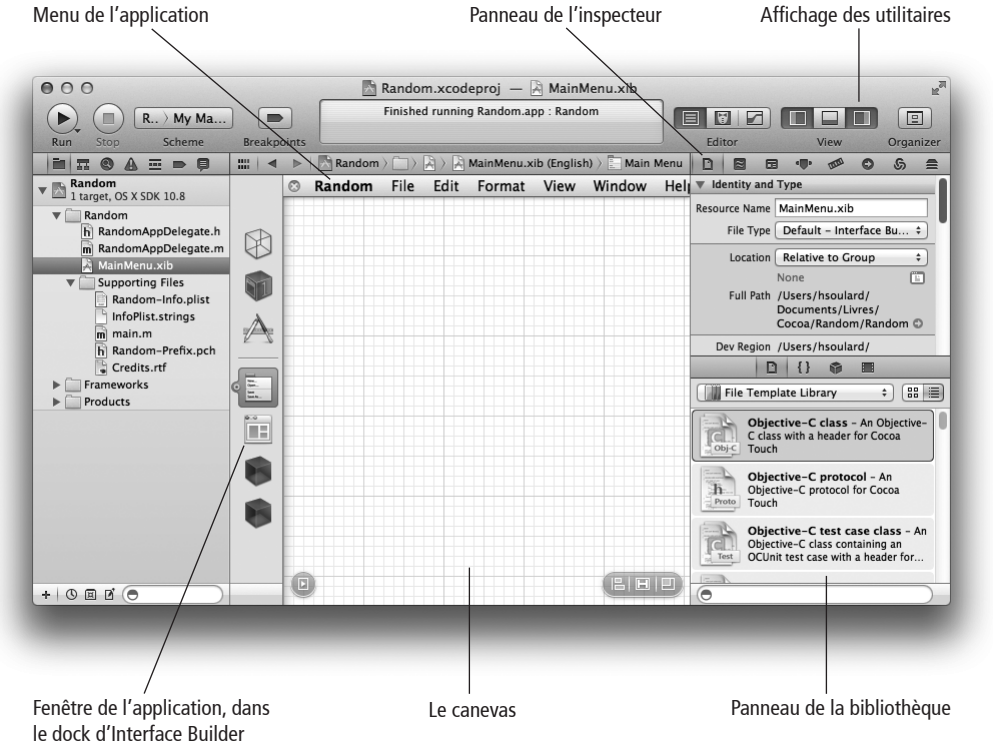


Figure 2.7
MainMenu.xib.

Interface Builder permet de créer et de modifier les objets de l'interface utilisateur de l'application, comme les fenêtres et les boutons. Nous pouvons également créer des instances de nos propres classes et établir des connexions entre ces instances et les objets standard de l'interface utilisateur. Lorsqu'un utilisateur interagit avec les objets de l'interface, les connexions définies entre ces objets et nos classes déclenchent l'exécution de notre code. Interface Builder enregistre ces objets et leurs connexions dans un fichier XIB.

La zone des utilitaires

La zone des utilitaires comprend deux panneaux : l'inspecteur et la bibliothèque. Le panneau de l'inspecteur affiche les paramètres qui correspondent au fichier ou à l'objet Interface Builder actuellement sélectionné. Le panneau de la bibliothèque propose des modèles de fichiers, des extraits de code, des objets et des médias que nous pouvons employer dans un projet. Nous pouvons faire glisser les widgets depuis la bibliothèque d'objets vers notre interface utilisateur.

Par exemple, si nous souhaitons un bouton, il suffit de le faire glisser depuis la zone de la bibliothèque.

La fenêtre vierge

Cliquez sur l'icône de la fenêtre dans le dock d'Interface Builder. La fenêtre vierge qui apparaît représente une instance de la classe **`NSWindow`** qui se trouve dans notre fichier XIB (voir Figure 2.8).

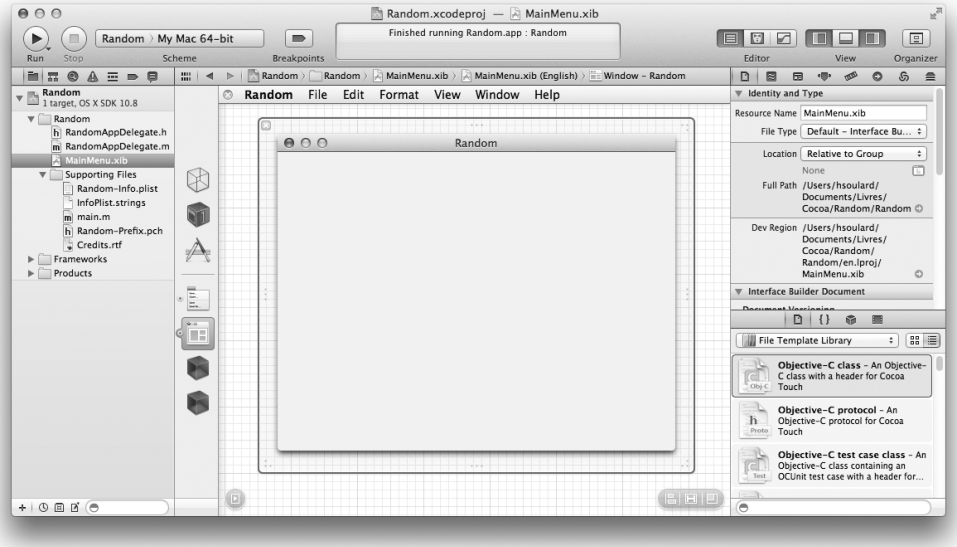


Figure 2.8

Instance de `NSWindow`.

Lorsque nous déposons dans cette fenêtre des objets provenant de la bibliothèque, ils sont ajoutés au fichier XIB. Après la création des instances de ces objets et la modification de leurs attributs, l'enregistrement d'un fichier XIB revient à lyophiliser (*archiver*) les objets dans le fichier. Lors de l'exécution de l'application, le fichier XIB est *désarchivé* par **`NSRunApplication()`** et les objets sont réanimés. Une application plus complexe pourra inclure plusieurs fichiers XIB, dont le chargement se fera au besoin.

Après que l'application a terminé le chargement des objets, elle attend simplement que l'utilisateur effectue une action. Lorsqu'il clique sur un widget ou saisit du texte, le code de l'application est appelé automatiquement. Si vous n'avez jamais écrit une application qui possède une interface utilisateur graphique, la répartition des tâches suivante risque de vous surprendre : le contrôle se trouve dans les mains de l'utilisateur, votre code réagit simplement à ses actions.

Pour les plus curieux : fichiers XIB et NIB

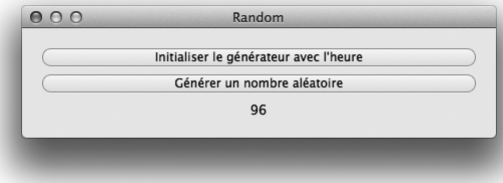
Un fichier XIB est une représentation XML des objets de l'interface utilisateur et de leurs connexions. Lors de la compilation de l'application, le fichier XIB est transformé en fichier NIB.

Il est plus facile de travailler avec le fichier XIB, notamment pour le contrôle des fichiers sources, mais le fichier NIB est de taille plus réduite et est plus facile à analyser. C'est pourquoi une application est fournie avec un fichier NIB. De façon générale, nous manipulerons uniquement des fichiers XIB, tandis que nos applications utiliseront uniquement des fichiers NIB. Cependant, la plupart des développeurs emploient les termes XIB et NIB de façon interchangeable. (Note : "NIB" signifie "NeXT Interface Builder" et "NS" représente "NeXTSTEP".)

Agencer l'interface

Nous allons la présenter pas à pas, mais n'oubliez pas que l'objectif est de créer une interface utilisateur semblable à celle illustrée à la Figure 2.9.

Figure 2.9
L'interface terminée.



Dans la liste de sélection d'une bibliothèque, choisissez Cocoa. Faites glisser un bouton depuis la fenêtre Library (voir Figure 2.10) vers la fenêtre vierge. Pour que cela soit plus facile, vous pouvez sélectionner le groupe Cocoa > Controls dans la partie supérieure de la fenêtre Library ou saisir **`button`** dans le champ de recherche.

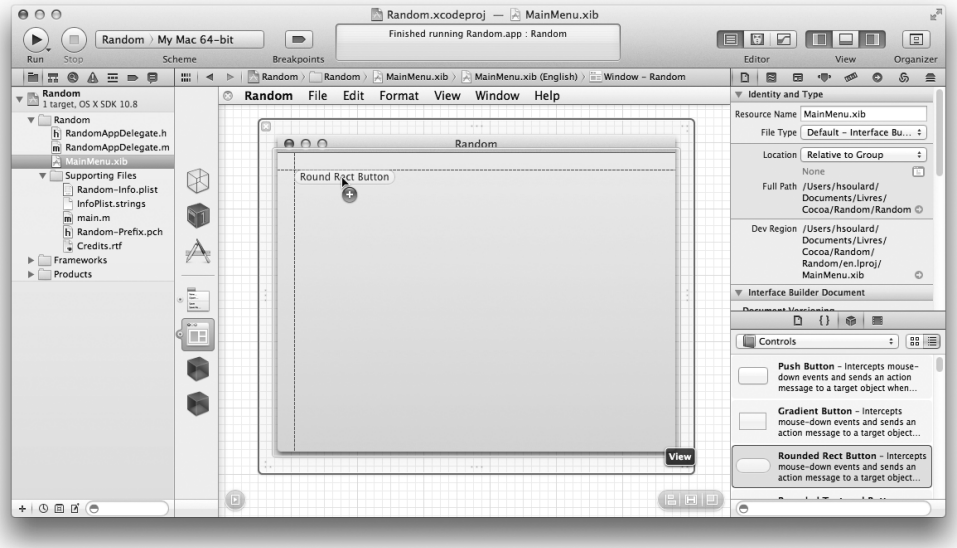


Figure 2.10
Faire glisser un bouton.

Double-cliquez sur le bouton pour fixer son intitulé à INITIALISER LE GÉNÉRATEUR AVEC L'HEURE.

Copiez et collez le bouton. Fixez l'intitulé du nouveau bouton à GÉNÉRER UN NOMBRE ALÉATOIRE. Faites glisser le champ de texte LABEL (voir Figure 2.11) et déposez-le sur la fenêtre.

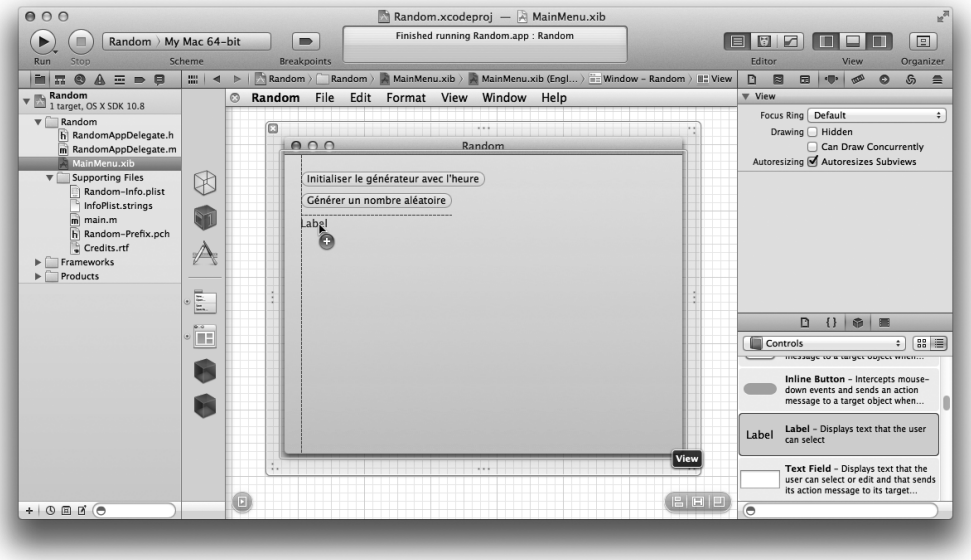


Figure 2.11

Faire glisser un champ de texte.

Pour que le champ de texte soit aussi large que les boutons, faites glisser ses bords gauche et droit vers les côtés de la fenêtre. Notez les lignes bleues qui apparaissent lorsque vous approchez du bord de la fenêtre. Ces guides vous aident à respecter les règles des interfaces graphiques utilisateur définies par Apple.

Diminuez la taille de la fenêtre en déplaçant les poignées transparentes qui l'entourent.

Pour que le contenu du champ de texte soit centré, vous devez utiliser l'inspecteur d'attributs. Sélectionnez le champ de texte puis l'onglet ATTRIBUTES INSPECTOR en partie supérieure du panneau de l'inspecteur. Cliquez sur le bouton de centrage (voir Figure 2.12).