1 iPhone プログラミング入門

沼田哲史
大阪電気通信大学

iPhone プログラミングの概要

このチュートリアルでは，Apple 社より販売されている iPhone 用のアプリケーション開発について解説します．

■ iPhone というプラットフォーム

2007 年 1 月に「時折，すべてを変える革新的な製品が登場する（Every once in a while, a revolutionary product comes along that changes everything.）」として，Apple 社から主にタッチパネルのみで操作するスマートフォン iPhone が発表され，同年 6 月末にアメリカで販売が開始されました．iPhone 上で独自のアプリケーション開発が行える iPhone SDK が発表されたのが 2008 年 3 月のことです．そして 2008 年 7 月 10 月には，自作アプリケーションが販売できる App Store が iTunes 上で公開され，その翌日に次バージョンの iPhone 3G が日本でも発売されました．

2011 年 1 月現在では，326ppi の解像度を持つディスプレイを搭載した iPhone 4（図 -1），電話機能のない iPod touch，そして電子書籍リーダとして最適な本体サイズの iPad が登場しています．それらの各種のデバイスに対応するために，当初 iPhone OS と呼ばれていた iPhone 用の OS は「iOS」と呼ばれるようになり，それに伴って iPhone SDK は「iOS SDK」と名前が変えられました．なお本稿では，これらのデバイスをまとめて「iOS デバイス」と総称します．

App Store はその立ち上げからわずか 2 カ月後の 2008 年 10 月には 1 億回のダウンロードがあったと報告され，開発者にもユーザにも非常に重要なプラットフォームとして注目を集めたことが確認されました．そしておよそ 2 年後の 1 月 22 日には 100 億回のダウンロードが達成されたと報告されました．Apple 社から発表されているダウンロード回数をまとめると図 -2 のグラフのようになり，そのプラットフォームが急速な成長を見せていることが分かります．

■ iPhone のフレームワーク

iOS プログラミングの環境である Cocoa Touch は，Mac OS X プログラミングのための Cocoa がベースになっています．Cocoa は Mac OS X のために 10 年以上も使われてきた実績のある環境です．そして Cocoa Touch は Cocoa と同様に高度な MVC パラダイムをサポートしたプログラミング環境

図 -1　iPhone 4 のデバイス

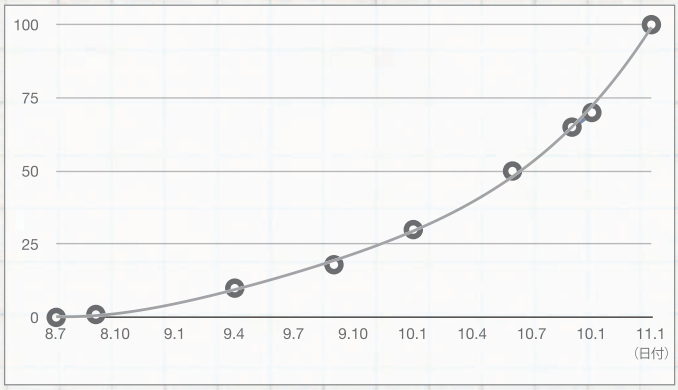

図 -2　iPhone アプリのダウンロード回数の推移（単位：億回）

Foundation	文字列処理・動的配列・連想配列などのサポートを提供する.
UIKit	ボタンなどの画面の基本になる部品の処理の実装を提供する. ページめくりやフリップといった画面遷移効果（トランジション）も簡単に実現できるように設計されている.
AV Foundation	2Dオーディオの簡単な再生をサポートする.
Open AL	3D空間を考慮したオーディオ再生をサポートする.
Core Animation	レイヤーと呼ばれる軽量の要素を使って，平面状の矩形を 3 次元空間の中で変形・移動させて，効率的にアニメーションを実現する.
OpenGL ES	2D/3D グラフィックの効率的な描画をサポートする.
Media Player	ムービーの簡単な再生をサポートする.
Core Data	オブジェクト指向的なデータ管理（O/R マッピング，シリアライズ，SQL ベースでの検索）のサポートを提供する.
Game Kit	ゲーム作成に便利な，各種ネットワーク通信（ピア検索，ペアリング，クラウド上でのスコア管理）のサポートを提供する.
Map Kit	アプリケーション埋め込みの地図表示をサポートする.

表 -1　Cocoa Touch の代表的なフレームワーク

のもと，非常に簡単なコード記述で高機能なアニメーションサポートが可能な Core Animation や，高度な音声処理のための Core Audio など，多数の有用なフレームワークを iOS デバイス用に提供します.

表 -1 に，Cocoa Touch の代表的なフレームワークとその概要をまとめます．Cocoa Touch には，これ以外にも 20 以上のフレームワークが用意されています.

iPhone アプリ開発環境構築

この章では，iPhone アプリの開発環境を整えるための手順について説明します．iPhone アプリ開発のための iOS SDK は，Apple Developer として登録すれば，無料で入手できます．ただし，Intel 製の CPU が搭載された Mac 本体と，その上で動作する Mac OS X 10.6（最新バージョン）が別途必要となります.

有料・無料の別を問わず，iPhone プログラミングやアプリ販売申請などは，すべて iOS Dev Center（http://developer.apple.com/devcenter/ios/）という Web ページから行います．iOS Dev Center にログインすると，iOS SDK が無料でダウンロードできるほか，有料の iOS Developer Program に参加している場合には，開発に利用する実機登録や販売アプリの登録申請が行えます．なお，最新の iOS SDK が入った有料の Xcode 4 は，Mac 用のオンラインストア App Store から購入することもできます.

■ Apple Developer 登録

iOS Dev Center にログインして SDK をダウンロードするためには，無料の Apple Developer 登録が必要になります．http://developer.apple.com/jp/programs/register/ から登録作業を行ってください．メールアドレス・氏名・住所などの個人情報を入力し，プログラム経験や開発希望分野などのアンケートに答えて，契約条項に同意すると，Apple Developer として登録できます．登録に使用したメールアドレスとパスワードを使って iOS Dev Center にログインすると，図 -3 のような Xcode+iOS SDK のダウンロードリンクが表示されます．Xcode は Mac OS X および iPhone 開発のための統合開発環境で，高度なテキストエディタやデバッガが利用できます.

■ Xcode+iOS SDK のインストール

ダウンロードしたディスクイメージをマウ

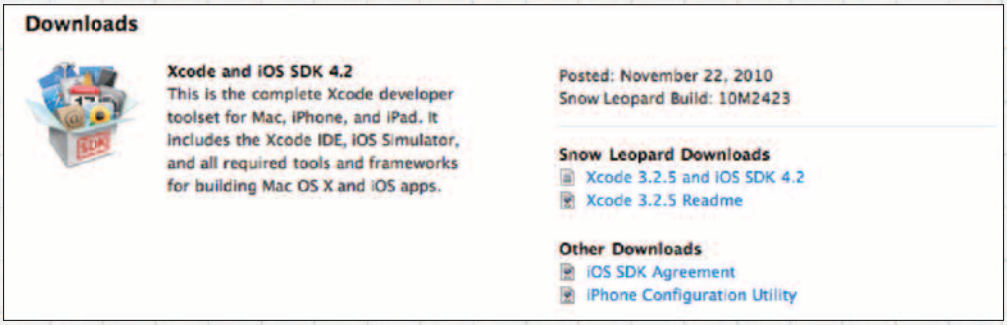

図 -3　Xcode+iOS SDK のダウンロードページ

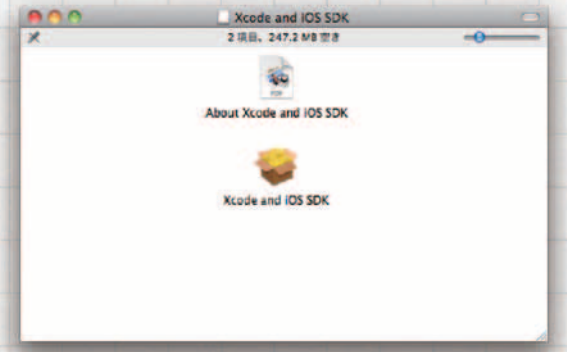

図 -4　Xcode+iOS SDK のインストーラ

ントすると，図 4 のように Xcode+iOS SDK のインストーラが出てきますので，これをダブルクリックして SDK をインストールします．インストールされたファイル一式は，デフォルトの設定では，「Macintosh HD」直下の「Developer」フォルダ以下に格納されます．

iOS SDK をインストールすると，Mac 上で動作する iOS シミュレータを使って，動作テストを行いながら iPhone や iPad 用のアプリケーションが開発できるようになります．ただし，加速度センサを使った機能やカメラを使った機能，またアプリケーション実行中に電話がかかってきたときの動作確認などはシミュレータではできませんので，iPhone や iPad などの実機を使って動作確認を行わなければいけません．

■ iOS Developer Program への参加

実機を使った開発には，有料の iOS Developer Program への参加が必要です．参加には，年間 10,800 円の参加費がかかります（2011 年 1 月時点）．このプログラムに参加することで，実機での開発に加えて，App Store での販売もできるようになります．

iOS プログラミングの基礎知識

この章では，iOS プログラミングを始めるにあたって必要な基礎知識を解説します．

■ Xcode と Interface Builder

iOS プログラミングは，高度な MVC パラダイムに基づき，「インタフェースのデザイン」と「機能の実装」の 2 つの作業を通じて行います．また統合開発環境（IDE）では当然のことですが，プロジェクトの管理も必要となります．

iOS SDK では，プロジェクトの管理と機能の実装を，Xcode というツールを使って行います．また，インタフェースのデザインには Interface Builder というツールを使います．Xcode は，バージョン管理を行いながらソースコードが編集できる統合開発環境で，アプリケーションのビルド（実行ファイルの作成）ができるだけでなく，デバッグ実行のためのデバッガも備えています（図 -5）．また Interface Builder は，GUI 部品をドラッグ＆ドロップでビューに追加し，各部品の設定を最終的な見た目どおりに編集することのできるユーザインタフェースの設計ツールです（図 -6）．

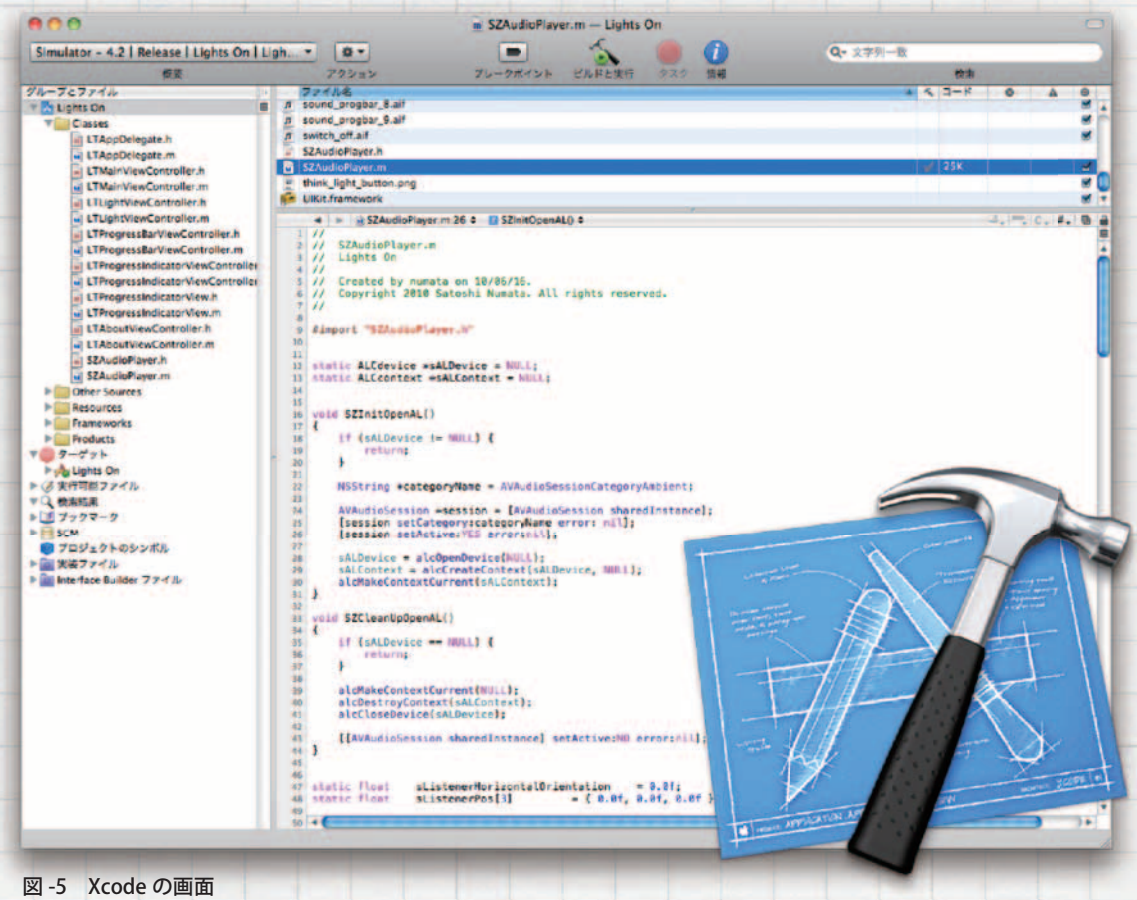

図 -5　Xcode の画面

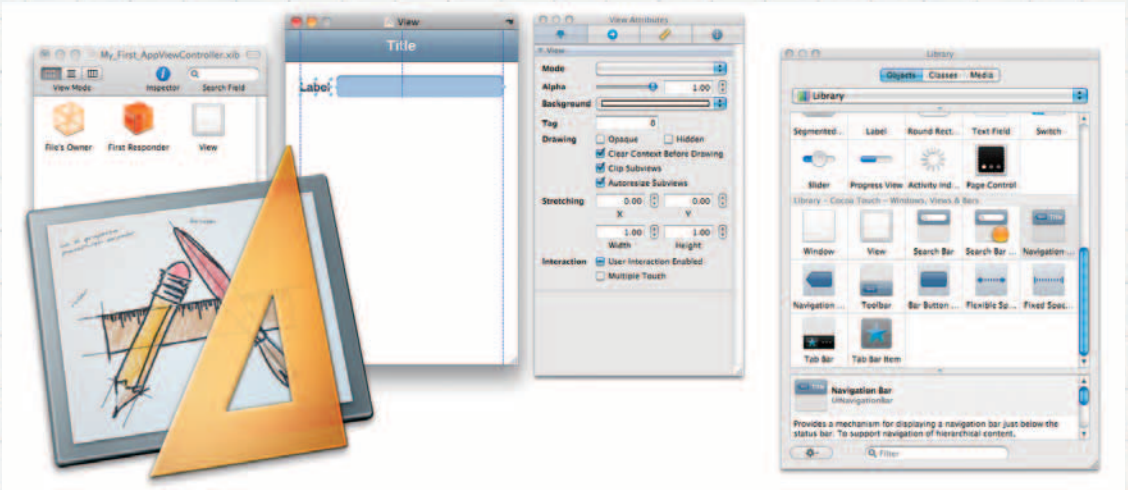

図 -6　Interface Builder の画面

■ Objective-C の文法

本誌の読者の方の多くは，すでに C# や C++ や Java などを使ったクラスベースのオブジェクト指向プログラミングに慣れていることと想定して，Objective-C の文法を簡単に説明します．なお，Objective-C は C 言語をベースとした言語ですので，必要な個所では，通常通り「#include < ～ >」で C 言語の標準ライブラリのヘッダファイルを読み込んで，標準ライブラリの関数を呼び出すことができます．

Objective-C では，クラスベースのオブジェクト指向プログラミングを行います．クラス定義は，C++ と同様に宣言部と実装部に

```
#import <Foundation/Foundation.h>

@interface MyClass : NSObject {
    // インスタンス変数の宣言
    int   score;
}

// メソッド宣言
- (void)sayHello;

@end
```

図 -7　Objective-C クラスの宣言部

分かれ，宣言部をヘッダファイル（～ .h）に記述し，実装部を実装ファイル（～ .m）に記述します．図 -7 と図 -8 に，クラス定義の例を示します．定義するクラスと，拡張子を除いた部分のファイル名は，同じになります．

このクラス定義例において，「#import」というのは，C 言語の「#include」とほぼ同等の機能を持つヘッダファイル読み込みのための宣言文です．Cocoa Touch では Foundation フレームワークが必要となりますので，そのヘッダファイルを読み込みます．

Objective-C では，クラス宣言は「@interface」と「@end」の間に記述します．「@interface」の後ろには「定義クラス名 : 親クラス名」の形式で，クラスの名前と継承元を指定します．「NSObject」というのは，Cocoa Touch におけるすべてのクラスのベースクラスとなるクラスの名前です．Java であれば java.lang.Object，C# であれば System.Object，VC++ の MFC であれば CObject に相当するクラスです．原則として，iOS プログラミングのクラス宣言では，親クラスの名前を指定します．

メソッド宣言は，「+（戻り値の型）メソッド名」または「-（戻り値の型）メソッド名」とします．「+」から始まるメソッドはクラスメソッドを，「-」から始まるメソッドはインスタンスメソッドを表します．

なお，引数が 1 つあるメソッドは，「-(void)moveX:(int)x;」というように，メソッド名の後ろにコロンを 1 つ付け，その後ろに「(型名) 引数名」の形式で宣言します．メソッド名が 2 つ以上ある場合には，「- (void)moveX:(int)x y:(int)y;」という形で引数を増やします．

```
#import "MyClass.h"

@implementation MyClass

- (void)sayHello {
    NSLog(@"Hello, world!!");
}

@end
```

図 -8　Objective-C クラスの実装部

```
#import "MyClass.h"

@implementation Foo

- (void)doTest {
    MyClass *obj = [[MyClass alloc] init];
    [obj sayHello];
    [obj release];
}

@end
```

図 -9　MyClass クラスの使用例

図 -8 のように，Objective-C のクラス実装では，まず #import 文を使って宣言部のヘッダファイルを読み込みます．そして，「@implementation」と「@end」の間に実装コードを記述します．この実装コードでは，NSLog() という，標準出力に文字列を出力する関数を使って「Hello, World!!」という文字列を書き出しています．

こうして定義したクラスのインスタンスを作成して，メソッドを呼び出して利用するには，図 -9 のように，まずクラス宣言のヘッダファイルを読み込みます．そして MyClass クラスに対して alloc メソッドを呼び出してインスタンスの生成を行い，さらにそのインスタンスに対して init メソッドを呼び出して初期化を行います．このように，Objective-C のメソッド呼び出しは，C# や

```
#import <UIKit/UIKit.h>

@interface MyClass : UIViewController {
    // アウトレット変数の宣言
    IBOutlet UISlider  *mySlider;
}

// アクション・メソッドの宣言
- (IBAction)doMyAction;

@end
```

図 -10　アウトレット変数とアクション・メソッドの宣言（ヘッダファイル）

図 -11　アウトレット変数とアクション・メソッド

Java だと「obj.method()」という形式で記述するところを，半角の角括弧を使って「[obj method]」という形式で記述します．

インスタンスが作成できたら，そのインスタンスに対して sayHello メソッドを呼び出します．

なお，iOS ではガベッジコレクションは提供されていませんので，インスタンスが不要になった時点で，明示的に release メソッドを呼び出して解放しなければいけません．

■ アウトレット変数とアクション・メソッド

iOS プログラムでは，アウトレット変数と呼ばれる特殊な変数と，アクション・メソッドと呼ばれる特殊なメソッドを利用することで，インタフェース部品を操作します．図 -10 に示すように，アウトレット変数は変数宣言の頭に「IBOutlet」というキーワードを付けて宣言します．またアクション・メソッドは，戻り値の型を書く代わりに「IBAction」と書くことで宣言します．

後半で実際のプログラム例を見ていただきますが，アウトレット変数というのは，プログラムコードから GUI 部品を参照して操作するための変数です．またアクション・メソッドというのは，GUI 部品がユーザにタッチされたアクションに応じて呼び出されるメソッドです．

これらを使って，多くの iPhone アプリは，図 -11 に示すような形態で動作します．まずユーザが iPhone を操作することにより，ボタンやスライダなどが動かされます．それに応じてアクション・メソッドが自動的に呼び出されます．そしてその結果を画面に表示するために，アウトレット変数を操作することで，ラベルなどの部品を操作して見た目を変更します．

カラーエディタの作成

この章では，iOS SDK を使った実際のプログラム例を見てみましょう．1 つのスライダを動かして色相を変更して色を作る，簡単なアプリケーションを作成してみます．

■ プロジェクトの作成

まずはアプリケーションの基礎となる，空のプロジェクトを作成します．Xcode を起動して，メニューから「ファイル」-「新規プロジェクト …」を選択してください．図 -12 のようにベースとなるテンプレートを選択する画面が表示されますので，最もシンプルな「View-based Application」を選んで「選択 …」ボタンを押し，次の画面でプロジェクト名を入力して保存します．

■ インタフェースのデザイン

次にインタフェースをデザインします．Xcode のプロジェクトにはいくつかのファイルが含まれていますが，図 -13 のように「Resources」グループの中の

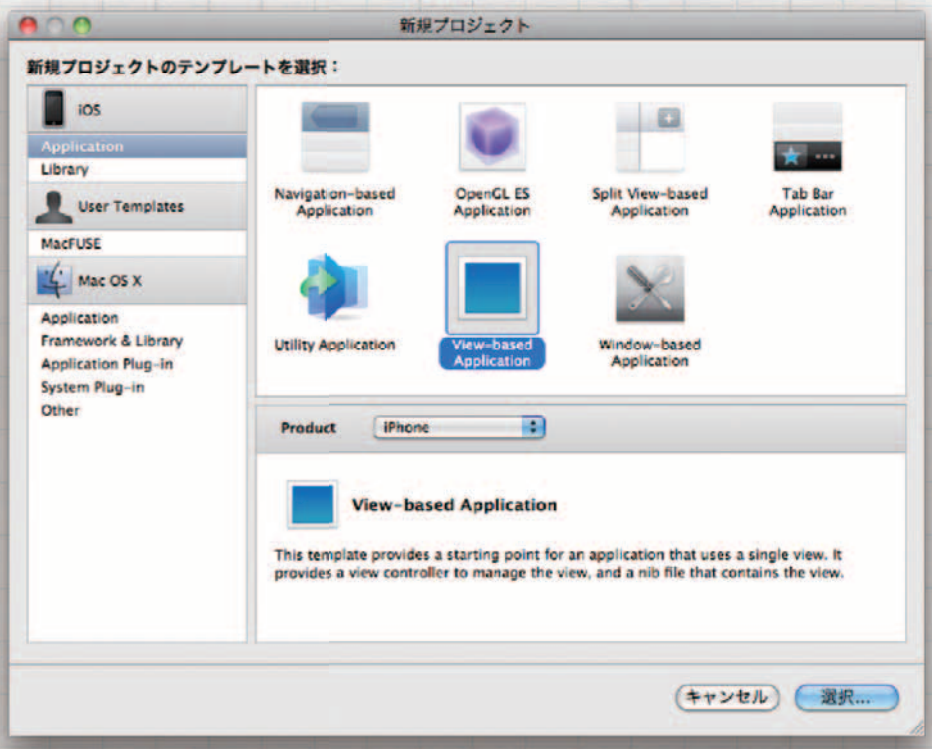

図 -12　新規プロジェクトの作成

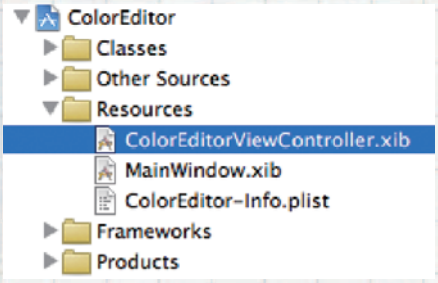

図 -13　インタフェース定義の XIB ファイル

「ColorEditorViewController.xib」というファイルを探し，このファイルをダブルクリックして開きます．この「xib」という拡張子がついた XIB ファイルがインタフェースを定義しているファイルです．XIB ファイルは Interface Builder で開かれます．

Interface Builder では，「Library」パネルに並んでいる各種 GUI 部品をビュー上にドラッグ＆ドロップで追加できます．図 -14 のように，スライダをドラッグ＆ドロップでビューに追加してください．

■ アクションの用意

次に Xcode に戻って，先ほどの手順で追加したスライダが動かされたときに呼び出されるアクションを用意します．スライダを配置したインタフェース定義のファイルが ColorEditorViewController.xib でしたので，同じファイル名を持つ ColorEditorViewController.h にアクション・メソッドの宣言を追加します．ColorEditorViewController.h を開いて，図 -15 のように「colorChanged:」というアクション・メソッドの宣言を追加してください．このアクション・メソッドでは，値が変更されたスライダを参照できるように，アクション元のスライダを引数として受け取るようにしています．

アクション・メソッドの宣言を追加したら，忘れずに「ファイル」メニューから「保存」を選択して，変更を保存しておいてください．

なお，本稿で作成するカラーエディタのサンプルでは，簡単のためにアウトレット変数は使用しません．

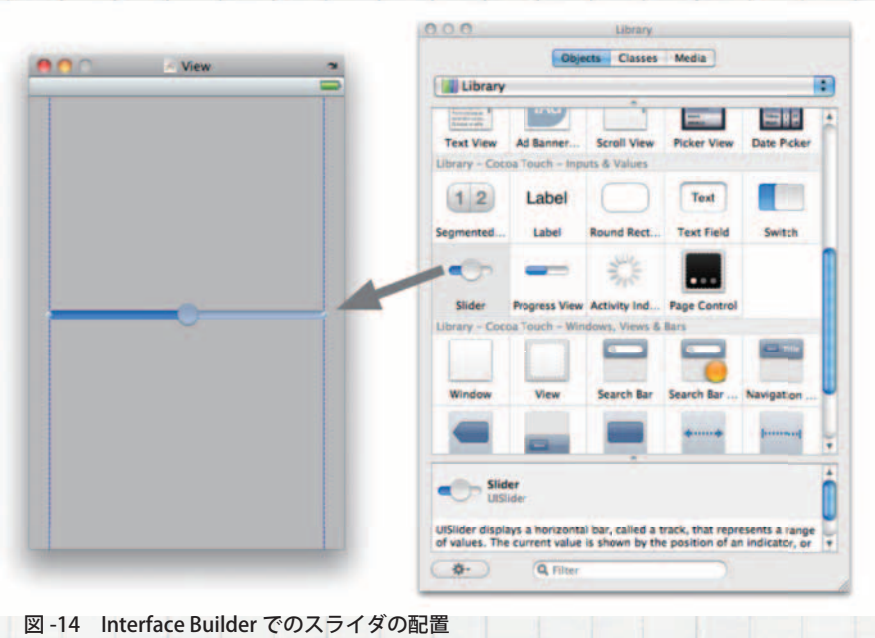

図 -14　Interface Builder でのスライダの配置

```
#import <UIKit/UIKit.h>

@interface ColorEditorViewController : UIViewController {

}

- (IBAction)colorChanged:(UISlider *)slider;

@end
```

図 -15　アクション・メソッド宣言の追加

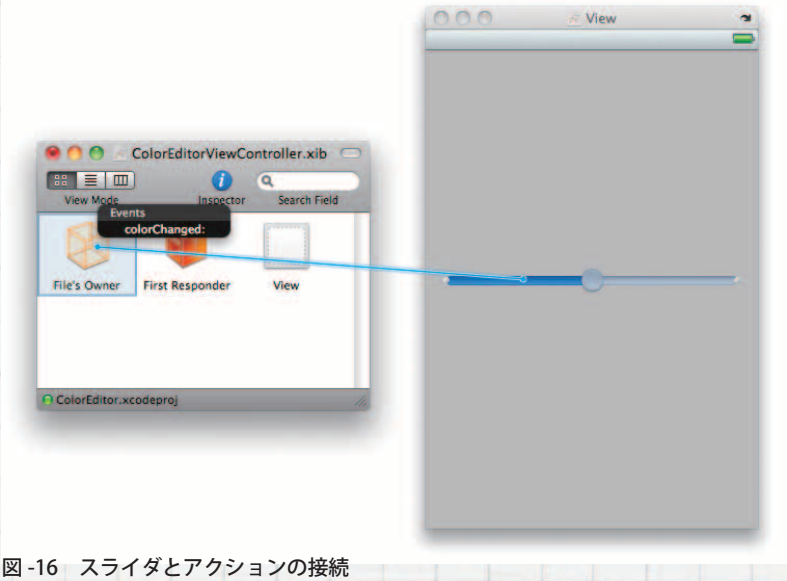

図 -16　スライダとアクションの接続

■ インタフェース部品とコードの接続

再び Interface Builder に移って，ユーザによってスライダが動かされたときに，先ほど追加したアクションが呼び出されるように設定します．スライダを右クリック（あるいは control キーを押したままクリック）し，マウスボタンを押したままマウスカーソルを移動させます．するとスライダから線が伸びますので，そのまま「File's Owner」というアイコンまでマウスを移動させて指を離すと，スライダから接続可能なアクションの一覧メニューが表示されます．このメニューで，先ほど追加した「colorChanged:」メソッドを選択してください（図 -16）．

Interface Builder での作業はこれで終わりです．Interface Builder のメニューから「File」-「Save」を選択して，変更内容を保存してください．

■ アクションの実装

最後に Xcode に戻り，アクション・メソッドの実装コードを書きます．スライダの部品は，デフォルトでは 0.0 ～ 1.0 の値を取るように設定されています．そこでこの値を色相とする色を表すオブジェクトを作成し，その色をビューの背景色として設定します（図 -17）．

■ 実行テスト

それでは iOS シミュレータを使って，ここまで作成したアプリケーションを実行してみましょう．アプリケーションの実行は，Xcode から行います．まず iOS シミュレータで実行するために，ツールバーの「概要」メニューで「Simulator」を選択します（図 -18）．そして「ビルド」メニューから「ビルドと実行」を選択してください．

すると図 -19 のように iOS シミュレータが起動して，アプリケーションの実行が始ま

ります．マウスを使ってスライダを動かして，スライダの値に応じて画面の背景色が変わることを確認してください．

作った iPhone アプリを実機で動かそう

この章では，作成したアプリケーションを実機で動かす方法について解説します．

上にも書いたように，自作アプリケーションを実機で動作させるためには，有料の iOS Developer Program への参加が必要です．iOS Developer Program に参加すると，iOS Dev Center（http://developer.apple.com/devcenter/ios/）からデバイスの登録とアプリケーション識別子の登録ができるようになります．

iOS デバイスを登録するためには，デバイス固有の識別子（UUID）が必要となります．UUID は，iTunes 上で見ることのできるデバイスの概要から，「シリアル番号」と書いてある部分を 1 回クリックすることで表示されます（図 -20）．UUID が表示された状態でメニューから「編集」-「コピー」を選択すると，クリップボードに UUID がコピーされますので，この情報を使って iOS Dev Center でデバイスを登録します．

iOS Dev Center で登録するアプリケーション識別子には，ドメイン名を持っている場合には，そのドメイン名を逆順にしたものを登録します．たとえば「ipsj.or.jp」というドメイン名を持っている場合には，「jp.or.ipsj.apps.*」といった識別子を登録します．ドメイン名を持っていない場合には，「myapps.*」といった識別子で構いません．

iOS Dev Center でデバイスとアプリケーション識別子の登録が完了すると，それらの情報をひとまとめにした「プロビジョニング・プロファイル」というファイル（～.mobileprovison）を作成してダウンロード

```
...
@implementation ColorEditorViewController

...
- (IBAction)colorChanged:(UISlider *)slider
{
    float hue = [slider value];
    UIColor *color = [UIColor colorWithHue:hue
                                saturation:1.0f
                                brightness:1.0f
                                     alpha:1.0f];
    [self.view setBackgroundColor:color];
}

@end
```

図 -17　アクション・メソッドの実装

図 -18　シミュレータ実行の設定

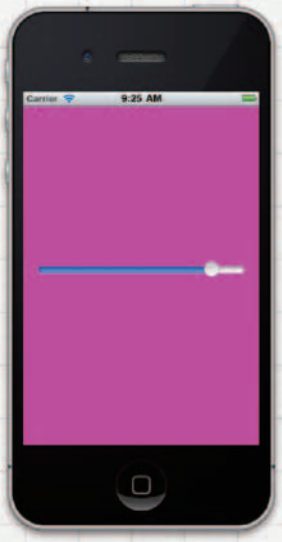

図 -19　iOS シミュレータでのカラーエディタの実行テスト

図 -20　iTunes による iOS デバイスの UUID の確認

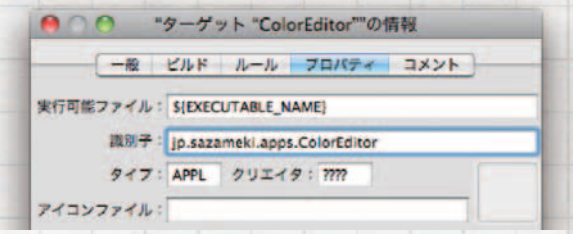

図 -21　プロジェクトのアプリケーション識別子の設定

図 -22　ビルドに使用するプロファイルの選択

できるようになります．このファイルを Xcode の Dock アイコンにドラッグ＆ドロップして登録することで，実機上で実行できるようになります．

Xcode の「プロジェクト」メニューから「アクティブターゲット “ColorEditor” を編集」を選択して，「プロパティ」の設定で，先ほど iOS Dev Center で設定したアプリケーション識別子の「*」の部分をアプリケーション名に変えて設定します（図 -21）．

Xcode の「プロジェクト」メニューから「アクティブターゲット “ColorEditor” を編集」を選択して，「プロパティ」の設定で，先ほど iOS Dev Center で設定したアプリケーション識別子の「*」の部分をアプリケーション名に変えて設定します（図 -22）．

以上の設定が完了したら，ツールバーの「概要」メニューで「Device」を選択します．「ビルド」メニューから「ビルドと実行」を選択すると，アプリケーションがデバイス上にインストールされて実行が始まります．

iPhone アプリの配布・販売について

App Store で配布するためには，App Store 専用のプロファイルを新たに作成して Xcode にインストールし，そのプロファイルを使ってビルドした実行ファイルが必要になります．そしてアプリケーションの説明文，アイコン，キーワード，カテゴリ，サポート URL などを設定して，アプリケーションの登録申請を行います．

2010 年 6 月に開催された WWDC で Steve Jobs 氏が発表したデータによれば，アプリケーションの 95% が 7 日以内に承認されているとのことです．筆者の経験的にも，時差を含めて 8 日以内には承認されています．なお，アプリケーションのアップデート時にも，同じ手順を踏む必要があり，承認まで同程度の時間がかかります．なお，承認されなかった 5% のアプリケーションで最も多いのは，予期せぬクラッシュや説明文のとおり動かないものだと説明されています．

なお，iPhone 用に作成したアプリケーションは，基本的には iPad でも iPod touch でも実行できることが前提となります．そのため，iPhone 実機だけでなく，iPad や iPod touch 上での動作も確認しておく方が良いでしょう．筆者の経験では，iPhone での使用を想定して「電話をかける」ボタンを用意していて，iPod touch でもこのボタンが使えるようになっていることを指摘されてアプリケーションが承認されなかったことがあります．

本稿では，iPhone アプリの開発について，簡単に概略をまとめました．本稿が皆さんの iPhone アプリ開発の足掛かりとなることを期待しています．

（平成 23 年 2 月 11 日受付）

沼田哲史（正会員）　numata@dg.osakac.ac.jp
1978 年 1 月生．2005 年大阪大学大学院情報科学研究科にて博士（情報科学）取得．同年より大阪電気通信大学総合情報学部デジタルゲーム学科講師．著書に「実践 iPad/iPhone ゲームプログラミング」（秀和システム），「iPad/iPhone アプリを作る前に知っておきたい 70 の常識」（秀和システム）．