# Xcode ことはじめ－アプリが作りたい学生のために　その1－

*Startup Exercises of Xcode – For the Students to make iOS Applications, Part. 1*

奥 山　徹　　曽我部 雄樹
Tohru Okuyama　　Yuki Sogabe

**概要**

これは、スマートフォンやパッド型のコンピュータのアプリケーション（以下、「アプリ」と記す。）を作りたい学生たちとともに、Mac OS や iPad を真剣に使ったことのない教員2名が悪戦苦闘しながらも、どうにかアプリらしきものを作る基礎を学生とともに習得するまでの記録である。そのため、論文というよりは読み物としての色合いが強いが、最近流行の用語で言えば、アクティブラーニングの1つの実践例として報告する。

## 1. はじめに

もともと今年度のゼミ生は iPad[1]を使い、パッド型コンピュータでの情報処理を学びたいということで、筆者のゼミにやってきた。ゼミ生が来た当初は、Mac はおろか iPad すらない研究室で、どうやってこの要求を満たすかを悩んでいたが、どうにか 10 台の iPad mini と 2 台の MacBook Air が準備でき、今年からようやく iPad を使ったゼミを開始することができた。しかし、指導する教員も学生も iPad 初心者であり、お互いに情報を集めて、議論して、実践して、また情報を集める、という作業を繰り返し、最近では、どうにか iPad mini を使いこなせるようになってきた。

あるとき、「どうせならアプリを作りたい！！」という学生の一言から、アプリ作成講座はスタートした。アプリ作成講座は、毎週火曜日の5時限目を使い、学生たちの自主的な学習講座として実施している。そして、5 時限目以降は、それぞれが自学自習の時間として、現在はアプリ作成の基礎を学んでいる。参加している学生は、経営学部経営情報学科から4～5名、ビジネス企画学科から1～2名、歯学部歯学科から1名であり、毎回4～5名が参加している。本報告では、実際に使用した教材を使い、アプリ作成講座に関する概要を報告する。

## 2. 開発環境の設定

アプリの開発ターゲットは iPhone および iPad とした。要するに iOS に対するアプリ作成を目指すこととした。最初は情報収集から開始した。その結果、

（1） iOS アプリの開発キットである Xcode[2]が Apple Store から無料でダウンロードできる。
（2） ただし、Xcode は Mac OS 上でしか動作しない。

（3） 開発言語は Objective-C である。

（4） Interface Builder(IB)と呼ばれるビジュアル化されたユーザインターフェイスが利用できる。

などの情報を得て、早速、MacBook Air に Xcode をダウンロードするところから開始した。しかし、ここで第一の関門が待ち構えていた。それは、Apple Store における Apple ID の取得である。本来なら教育目的のための Apple ID を付与してもらうのがよいが、今回は時間の都合上、1つの Apple ID を複数の MacBook Air および iPad でホームシェアして利用した。このような利用方法が許されているかどうかについては、現在調査中であるが、とりあえず、同じ Apple ID を使い、2 台の MacBook Air に Xcode をダウンロードしようとした。しかし、ここで二番目の問題が起きた。Xcode は2G バイト以上の容量があり、学内 LAN からのインターネット接続ではダウンロードできなかった。途中まではダウンロードできるのだが、ある程度の容量をダウンロードしたところで、接続が切断されるのである。何度か試したがうまくいかず、結局、iPhone のテザリング機能[3]を利用してダウンロードすることとした。これにより筆者の利用パケット数が 7G バイトを超えてしまい、速度制限されてしまったが[4]、どうにか Xcode をダウンロードすることができた。

以下は、ダウンロードした Xcode 利用の最初のステップとして作成した簡易利用マニュアルである。

# Xcode 利用マニュアル簡易版

奥山　徹

## １．はじめに

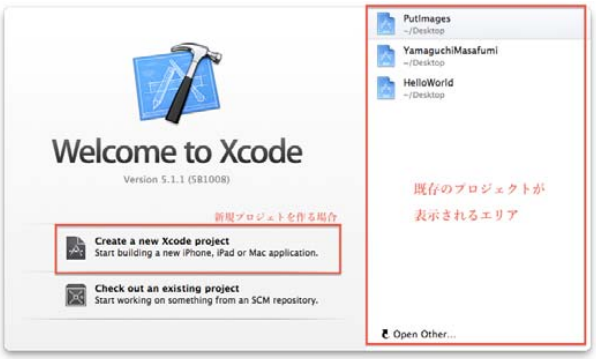
iPhone や iPad のアプリケーション（以下単に「iOS アプリ」あるいは「アプリ」と記す。）の開発は画面構成やその遷移など、多くのプログラム構成要素を含み、かなり難しい。そのため、開発には SDK(Software Development Kit)を使うことになる。SDK として、Apple が提供している Xcode が Apple Store から手軽にダウンロードできるので、それを使うのが簡単である。しかし、開発言語は Objective-C に限られてしまう。その他にも開発環境はいくつか存在するので、自分の技量と目的にあったものを選択するのがよい。ここでは、標準ともいえる Xcode の簡単な使い方を紹介する。

## ２．Xcode のダウンロード

使い方を紹介する前に、Xcode をダウンロードしておく必要がある。Apple ID を取得しているなら、その ID を使い簡単にダウンロードできる。インストール作業は特に必要なく、ダウンロードが終了した時点で、利用できるようになる。詳しくは、Xcode のダウンロードの方法を記した別のドキュメントを参照せよ。

３．Xcode の起動と簡単な使い方

起動は、Finder からでも Lanchpad からでもできるが、筆者はデスクトップにショートカットを作り、そこから起動しているが、好みの問題であるから、適当に設定して欲しい。Xcode が起動すると次のようなウィンドウが表示される。




既存のアプリ開発プロジェクトが存在する場合は、画面の右側にそのリストが表示される。一般に、ダウンロード直後にはプロジェクトは存在しないので、ここには何も表示されないはずである。今回は、既に利用している Xcode の起動画面をサンプルとしたため、３つの既存のプロジェクト（PutImages 他２つ）が表示されている。既存のプロジェクトを再度読み込む場合は、プロジェクト名をダブルクリックすればよい。

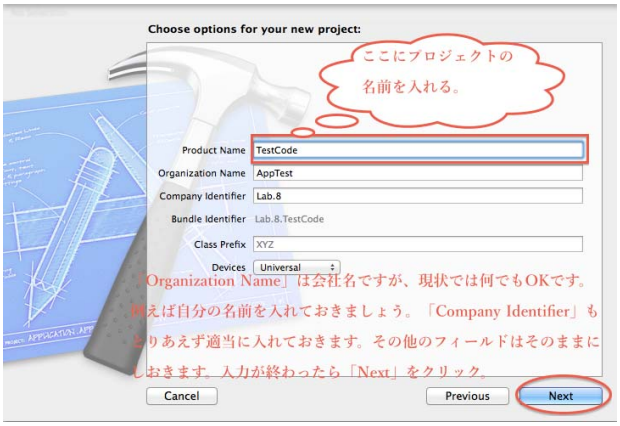
なお、Xcode のプロジェクトとは、アプリ開発のためのジョブの単位をプロジェクトと呼び、Xcode では、プロジェクトごとに開発単位を制御している。Xcode を使う最初のステップは、開発プロジェクトの名前を付けることであり、開発するアプリにふさわしい名前を付けることが重要である。

新しいプロジェクトを作成するには、画面左側の、

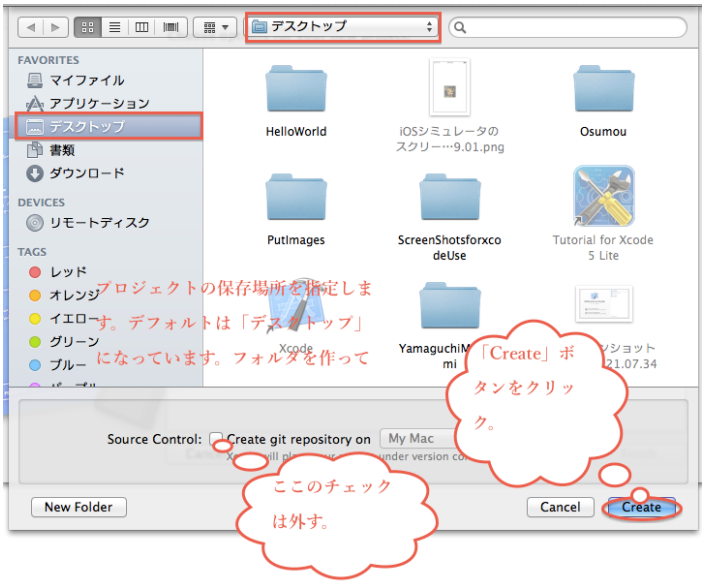
Create a new Xcode project

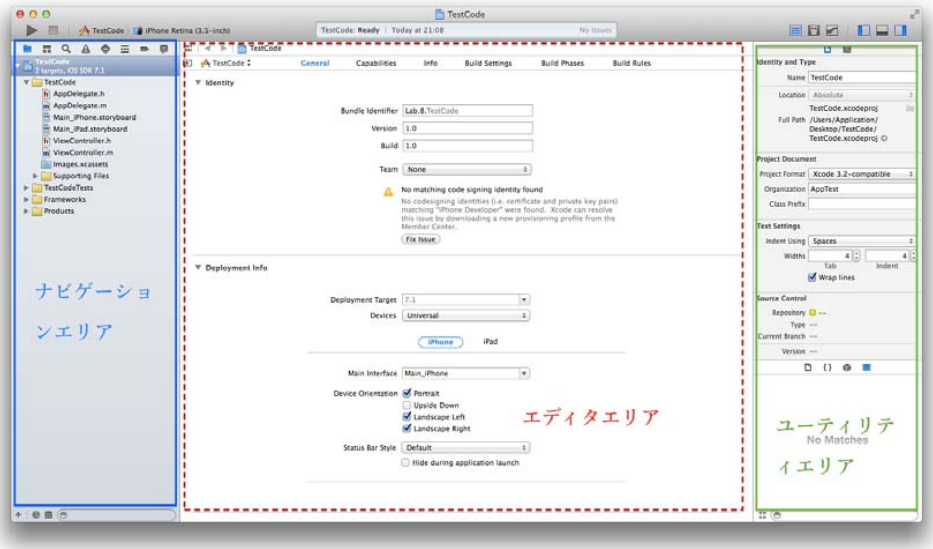
をクリックすればよい。すると、作成するアプリのテンプレート[5]の選択画面となる。

最も基本的なテンプレートである「Single View Application」を選択し、「Next」をクリックする。すると、プロジェクトに名前を付ける画面となる。ここでは、「TestCode」と名前をつけている。Organization Name と Company Identifier については、図中に注意書きを示してあるが、いまのところはそれに従って記述してよい。しかし、アプリを公表するときにはこれではダメなので、後に書き方について注意する予定である。プロジェクト名を入力すると、次にプロジェクトの保存場所を聞いてくる。デバイスは、Universal のままでよいが、あらかじめターゲットがわかっているなら、それを選択する。名前の入力が終わったなら、「Next」をクリックする。すると、プロジェクトの保存場所を聞いてくる。

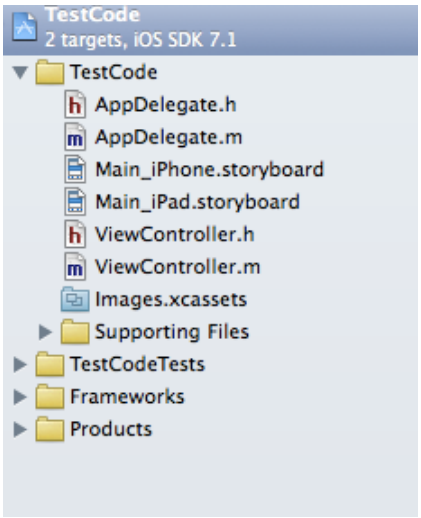



次に、プロジェクトを保存する場所を選択する。ここでは、デスクトップに保存することとして、「Source Control」のチェックを外して、「Create」をクリックする。






上の図は、Xcode のプロジェクト設定が終わり、実際にアプリを開発するための基本画面である。ここで示すように Xcode の基本画面は「ナビゲーションエリア」、「エディタエリア」、「ユーティリティエリア」の３つのエリアに分かれている。ナビゲーションエリアは、アプリのファイルやフォルダ、あるいはそれらの階層構造を示すエリアである。エディタエリアはその名の通り、プログラムなどの編集を行うエリアである。ユーティリティエリアは、プログラムで使用する UI(User Interface)などのプロパティを扱うエリアである。



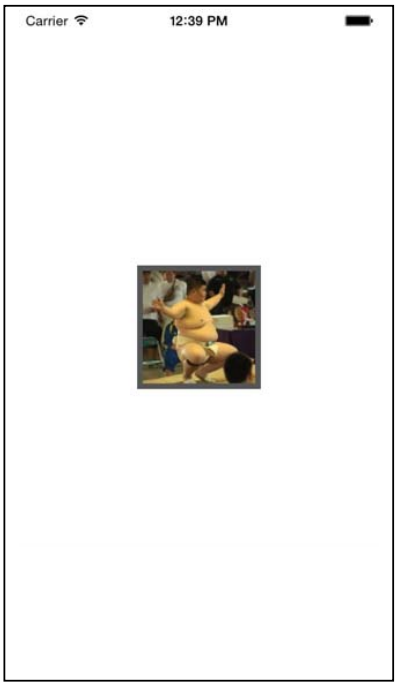

では、ナビゲーションエリアに表示されているファイルを確認しておく。拡張子「.h」のファイルはいわゆるヘッダファイルであり、各種の定義を行うものである。「.m」ファイルは、プログラムの各種処理を行うソースファイルのようである。拡張子が「.m」である理由は調査中である。「.storyboard」は、画面上で UI パーツを視覚的に配置して、画面遷移などの設計に使うツール（Interface Builder）である。ターゲットを「Universal」としたため、iPhone 用と iPad 用の 2 つの storyboard が用意されている。

「AppDelegate」ファイルは、アプリ全体の調整を行うファイルであり、「ViewController」ファイルは、主に画面設計用のファイルと推測されるが、詳しくは調査中である。

Xcode5 から新しく追加された画面リソース管理のための「Images.xcassets」の使い方は、現在調査中である。「Supporting Files」フォルダは各種のサポートファイルが入っており、「プロジェクト名 Tests」フォルダには各種のテストファイルが、「Frameworks」には各種の便利なツールが格納されている。「Products」フォルダには、実際のアプリが格納される。

## ４．おわりに

以上、Xcode の起動から、プロジェクト作成の方法、そして、各エリアの説明をしてきたが、アプリのコード入力が終わったら、左上の三角（矢印？）マークをクリックする。すると、コードのコンパイルが始まり、成功するとエミュレータが起動して、作成したアプリの検証が可能となる。エラーがあると、エディタエリアに詳細なエラーメッセージが表示されるので、それを参考にしながらデバッグしていく。なお、ソースコードデバッガなど強力なデバッグツールが用意されているが、それらの使い方は、今後順を追って説明していく。なお、最後の図は iPhone エミュレータの画面の例である。



### 3. Hello World プロジェクト

最初に作成したのが「Hello World」プロジェクトと命名したアプリの実装である。これは、カーニハンとリッチーによる有名な C の教科書[6]の最初に出てくる実装例であり、ディスプレイ上に、

hello, world

という文字列を表示させるだけのプログラムに由来する。C のソースコードは、

```
main( )
{
        printf(“hello, world¥n”);
}
```

という簡単なものである。さて、C 言語でこのように実装される（現在の C 言語では、このように簡単な書き方はしない。）のであるが、これを表示するアプリをどのようにして作るのかが最初の課題である。

さて、とりあえず動かすことが可能なソースコードを探してみる。その結果、「ViewController.m」に次のコードを加えればよい（下のソースコードの「// ここから」から「// ここまでを追加」の部分）という記述を発見した[7]。

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // ここから
    UILabel *label = [[UILabel alloc] init];
    label.text = @"Hello World！";
    [label sizeToFit];
    label.center = self.view.center;
    [self.view addSubview:label];
    // ここまでを追加
}
```

そこで、このソースコードを「ViewControll.m」に追加する。図 1 は、プログラムを入力している画面であり、Xcode にはこのような強力な入力アシストシステムが搭載されていて、途中まで入力すると該当する可能性のあるコードの一覧を表示してくれるので、入力はとても楽である。

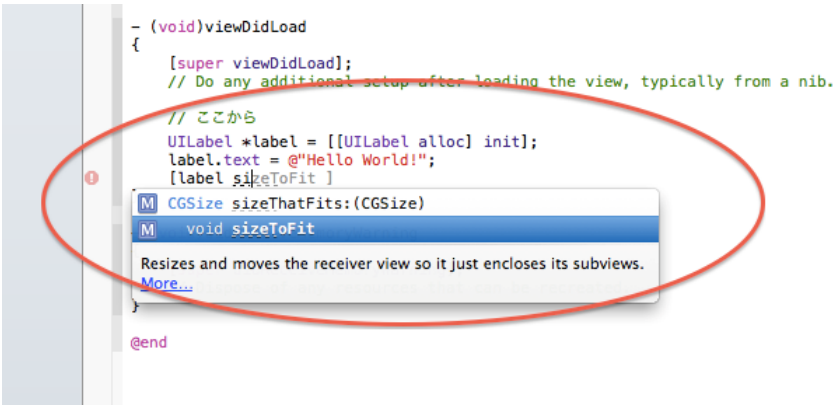



図 1. Xcode における入力アシストシステム

まずは、ナビゲーションエリアで、「ViewController.m」をクリックし、エディタエリアにソースコードを表示する。その中に、上で示した追加分のソースコードを入力する。入力が終わったら、図 2 の左上の1と示された三角のアイコンをクリックする。すると、2で示されたように「Build Succeeded」と表示され、ビルドが無事に終わったことがわかる。なお、エラーが発生したら、エディタエリアにエラーが表示されるので、それを手がかりに、エラーを修正する。

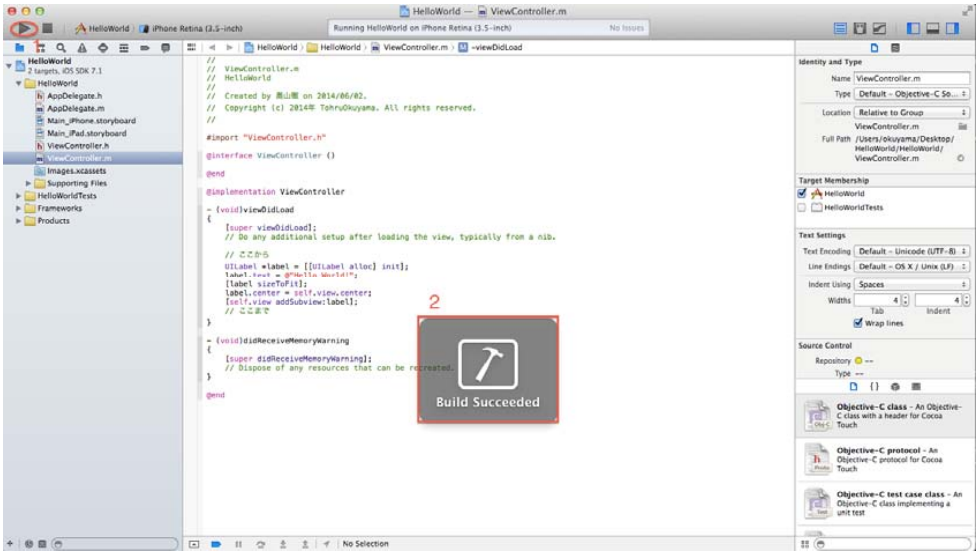



図 2. ソースコードのコンパイル

正常にビルドされると、やがて、図 3 の iOS エミュレータが起動して、アプリを実行してくれる。

さて、このようにアプリは正常に動いたのであるが、なんとなく釈然としない。いきなり、このようにガリガリとコードを書かなければならないのであろうか。それにしても、画面に表示させるだけのプログラムにしては大変である。これが、複雑なプログラムを作成することとなると、すべてのコードを手で書くのは、あ


図 3. iOS エミュレータ

まりにも非効率的である。

以下、いくつかの簡単なアプリを作成していくうちに、この問題については、簡単に答えが出た。次節でそれについて紹介するが、その前に、ソースコードの内容を見ておこう。最初の行で、「UILabel」クラスの、「label」というオブジェクトを生成し、初期化している。

```
UILabel *label = [[UILabel alloc] init];
```

クラスとオブジェクトの関係は、オブジェクト指向プログラミングの基本であるので、ここでは省略する。詳しくは、著者がオブジェクト指向を学んだメーヤーの著書[8]などを参照して欲しい。なお、著者は Eiffel という言語でオブジェクト指向プログラミングを学んだため、C++や Objective-C のようなＣからの派生言語のプログラミングは苦手である。iOS8 からは、Swift と呼ばれるより洗練された言語で開発できるようになる[9]。Swift では、よりオブジェクト指向らしいプログラミングができることを期待したい。ところで、「UILabel」というクラスは、「A variable sized amount of static text.」のためのオブジェクトを定義するクラスであり、もっとも基本的なテキストビューを作成するものの1つである。

次からは、label オブジェクトのプロパティを定義する部分であり、最初は「Hello World!」というテキストを設定し、次は、ビューの大きさに表示をフィットさせることを指示し、文字表示の中心座標を指定している。いずれも、UILabel クラスに標準で定義されているプロパティである。

```
label.text = @"Hello World！";
[label sizeToFit];
label.center = self.view.center;
```

そして、最後のこの行は、表示のための「self.view」にサブビューとして label の内容を貼り付ける操作であり、これにより指定されたラベル（文字列）が画面に表示されることになる。

```
[self.view addSubview:label];
```

**4. より洗練された方法**

上でも記したとおり、アプリ開発では、はたしてこのようにソースコードを直接入力していくことになるのだろうか。答えは、「否」である。Xcode にはより洗練された表示画面設計の手段が提供されている。それが、「Interface Builder（IB）」である。具体的には、ナビゲーションエリアの、「.storyboard」というものを利用して、表示画面やスイッチ、ボタンの配置、動作などを設計することができる。ここでは、Hello World

プロジェクトを storyboard を使い実現してみる。

まずは、Xcode に新しいプロジェクトを作成する。テンプレートは「Single View Application」とし、プロジェクト名は「HelloWorld2」としておく。そして、基本画面となったら、ここで、ナビゲーションエリアの「Main_iPhone.storyboard」をクリックする。すると、図 4 のように iPhone の画面を模した画像があらわれる。これが「Interface Builder(IB)」である。

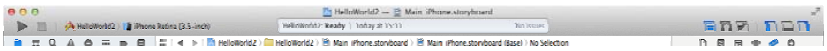


図 4. Interface Builder

ここにラベルオブジェクトを貼り付ける。貼り付けるには、図 5 の画面右下のオブジェクトライブラリから、「Label」を選択する。

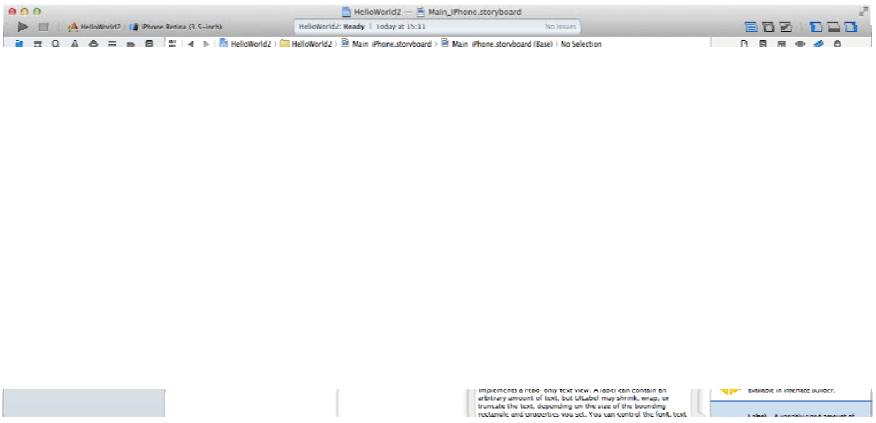


図 5. オブジェクトライブラリの Label を選択

オブジェクトライブラリをスクロールし、「Label」を見つけたら、図 6 の通り IB の iPhone の画像の上にドラッグ＆ドロップする。

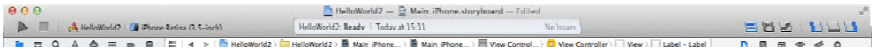


図 6. Label オブジェクトのドラッグ＆ドロップ

これで、ファイルオブジェクトが IB 上に貼り付けられる。貼り付けられたら、オブジェクトのアトリビュートインスペクターを開く。（図 7 の右上の四角の部分をクリックする。）

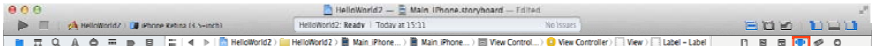


図 7. オブジェクトのアトリビュートインスペクター

すると、オブジェクトに定義されている各種のプロパティを変更できるようになるので、図 8 の右側の text の label という部分を「hello, world」（あえて、カーニハンらの最初のプログラムの文字列に合わせてある。）を入力する。

図 8. Label オブジェクトの text の変更

ラベルの表示エリア(ビュー)が小さく、すべて表示できないので、サイズインスペクター（図 9 の右上の四角1の部分）をクリックして、ビューのサイズを変更する。ここでは、幅だけを変更するので、図 9 の 2 の部分の幅の値を大きくする。図 9 の 3 のビューが文字列をすべて表示できるようになるまで、幅を大きくする。

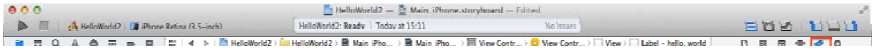


図 9. オブジェクトのサイズインスペクター

それでは、これを実行してみると、図 10 のように、簡単に文字列を表示できるようになった。

このように、IB を使えばビジュアル化されたインターフェイスを使い、種々のオブジェクトを簡単に画面に貼り付けて、そのプロパティを変更できる。これを使えば、詳細な動作を規定する部分だけをプログラムすればよく、アプリ作成の効率は格段によくなる。

ところで、このままでは画面上に置いたオブジェクトをプログラムのソースコード内に反映させることができない。そこで、次節でその方法を述べて、第一回目の報告を終了する。

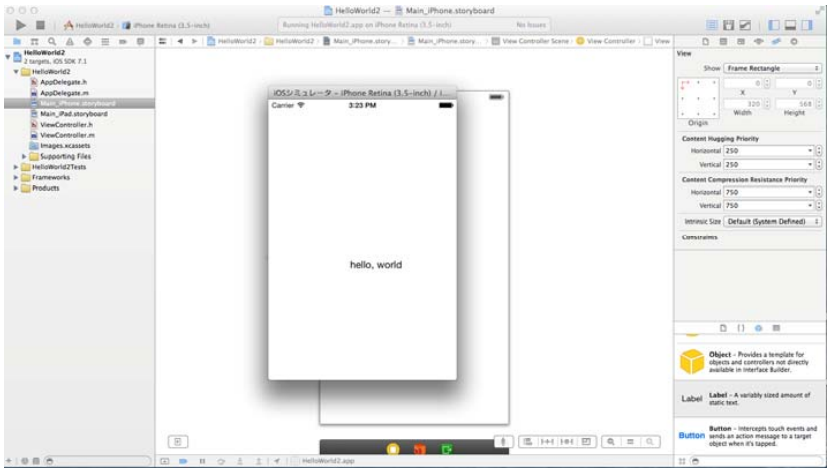



図 10. プロジェクトの実行画面

## 5. オブジェクトとソースコードの結合

IB（storyboard）上に貼り付けられたオブジェクトはプロジェクト内では有効だが、このオブジェクトをプログラムから参照することができない。そのため、オブジェクトとソースコードの結合が必要となる。以下に、その手順を示す。

まずは、label オブジェクトを IB に貼り付けた状態で、アシスタントエディタ（図 11 の右上の四角の部分 1をクリックする。）で、「ViewController.m」を開く。

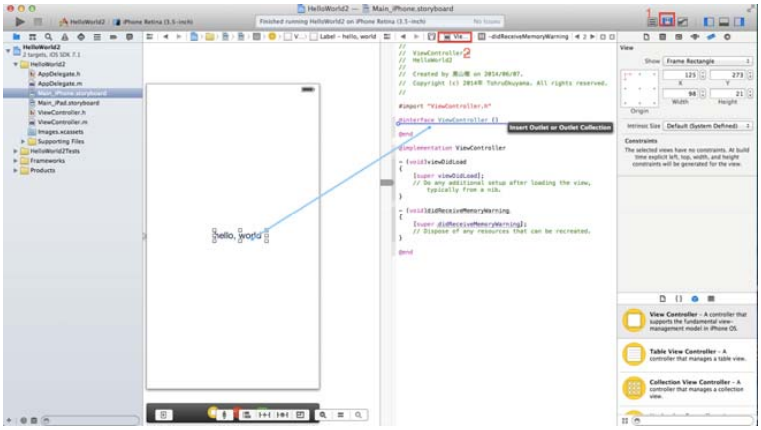


図 11. オブジェクトとソースコードの結合

もし、「ViewController.m」が開かなかったら、2の部分のファイル名をクリックして、当該ファイルを選択する。次に、IB 内の結合するオブジェクトをクリックし、「control」キーを押したまま、ドラッグする。すると、図 11 のように、結合のための矢印が示されるので、「Insert Outlet or Outlet Collection」と表示される位置(図の白黒反転した文字列が表示されている部分)で、ドロップする。すると、図 12 のように、結合するオブジェクトの名称をつけるように指示されるので、名前をつけて「connect」をクリックする。ここでは、「helloLabel」としたが、任意の名前を付けてよい。これが、ソースコード内のオブジェクトの参照名となる。

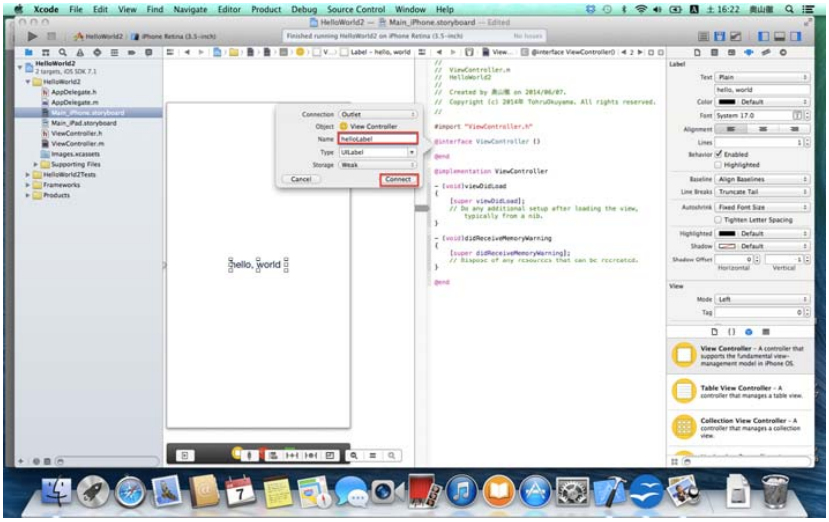


図 12. オブジェクトとソースコードの結合

すると、結合を示すソースコードが自動的に挿入される。コードの前の（図 13 の丸の中）記号は、これが IB 上のオブジェクトと結合していることを表している。（実際には、ヘッダファイルに結合するのが一般的である。）

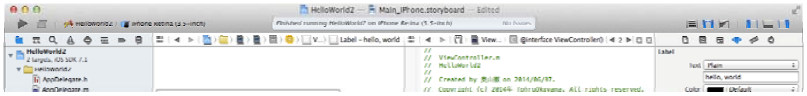


図 13. 結合されたオブジェクト

最後に、ソースコードの追加が、このオブジェクトに実際に影響するかどうかを確かめる。図 14 の四角で囲んだ部分を ViewDidLad の部分に付け加える。

```
[_helloLabel setText:@"Hello World!"];
[_helloLabel setBackgroundColor:[UIColor blackColor]];
[_helloLabel setTextColor:[UIColor whiteColor]];
```

最初は、text プロパティを変更するコードであり、ここでは、「Hello World!」に変更している。次は、backgroundcolor プロパティを変更してビューのバックグラウンドカラーを黒としている。最後は、textcolor プロパティを変更し、文字の色を白としている。以上のソースコードを追加した後、ビルドすると図 14 に示したとおり、黒のラベルビューの領域に、白い文字で「Hello World!」と表示されている。

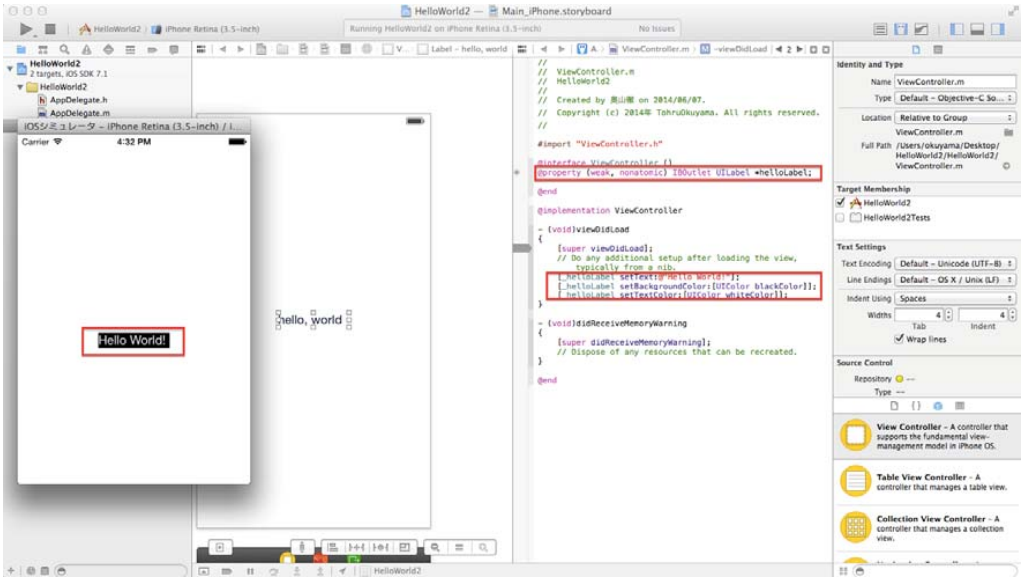



図 14. オブジェクトのプログラムからの参照

## 6. 終わりに

以上、Xcode を使ってアプリを作成するための最も基礎的な部分を紹介した。実際の作成講座では、すでに画像を表示して、それをクリックすることで、別の画像に切り替えるようなアプリを作成するところまで進んでいる。まだまだ、始まったばかりの講座で、学生はオブジェクト指向が何者かもわからない状態でひたすらソースコードとにらめっこしているところである。

今後は、より複雑なアプリを開発することで、iOS 関連のアプリ作成のコツをつかみ、スキルアップして欲しいと願っている。

文献など

[1] Apple,「あなたの物語はなんですか？ －iPad 紹介ページ。
URL: https://www.apple.com/jp/ipad/
[2] Apple Developer, “Xcode - The complete toolset for building great apps.”.
URL: https://developer.apple.com/xcode/
[3] Cool, “Tethering by iPhone5/iPhone5s au”.
URL: http://www.infraeye.com/study2/smartphone3.html
[4] Appllio,「iPhone でテザリングする設定方法と料金、通信速度制限への注意点」、2014 年 5 月.
URL: http://appllio.com/how-to-use-tethering-iphone-career-charge
[5] 小室 啓、「[iOS 7]アプリ開発入門 1からはじめる iOS 7 - はじめての Xcode 5」、2013 年 9 月.
URL: http://dev.classmethod.jp/references/first-step-xcode5/
[6] B.W.カーニハン、D.M.リッチー 著、石田晴久 訳、『プログラミング言語 C』、共立出版、1981 年 7 月、7頁.
[7] 平井祐樹, iOS アプリ開発初心者に捧ぐ開発環境 Xcode の概要とインストール (4/4), 2012 年 11 月.
URL: http://www.atmarkit.co.jp/ait/articles/1211/07/news017_4.html
[8] B.メーヤー 著、二木厚吉 監訳、『オブジェクト指向入門』、アスキー出版局、1990 年 11 月.
[9] S. Sekine、「アップル、新プログラミング言語 Swift を発表。レガシーを廃して高速化した iOS/OS X 開発用」、2014 年 6 月. URL: http://japanese.engadget.com/2014/06/02/swift-ios-os-x/

奥山 徹 （経営学部経営情報学科教授）
曽我部 雄樹 （経営学部ビジネス企画学科講師）