特集記事

# 特集：スマートフォン・プログラミング 事始め
# Windows Phoneのプログラミング

お茶の水女子大学 お茶大アカデミックプロダクション
塚田 浩二

## 1．はじめに

Windows Phone 7（以下、Windows Phone）は、Microsoftが現在開発中のスマートフォン向けOSです。従来のWindows Mobile 6.5（以下、Windows Mobile）までとは、ユーザ・インタフェース（以下、UI）／開発方法／配布方法などさまざまな点で、大幅な変更が行われています。ここでは、それぞれの変更点について簡単に説明した上で、具体的な開発手順を紹介します。なお、本稿執筆時点ではWindows Phone対応端末はまだ発売されていない（2010年末に発売予定）ため、発売時には若干仕様が異なる可能性があります。

### 1.1 UI

従来のWindows MobileのUIは、PC用Windows OSを小画面向けに移植した印象の強いものでしたが、Windows Phoneでは、Microsoft版iPod Touchともいえるメディアプレイヤー端末Zune HD[1]のようなシンプルなUIへと大幅に変更されています。また、Windows Phoneのホーム画面には、「live tiles」というユニークなUIが採用されるようです。友人の名前のタイルを作成して初期画面に登録すると、その友人のオンラインでの活動（メール、SNS、写真共有サイトなど）に手軽にアクセスできるそうです。ちなみにこれは、我々の提案する「ソーシャル顔アイコン」システム[2]のコンセプトに近く、個人的にも注目しています。

### 1.2 ハードウェア

従来のWindows Mobileでは、多様な仕様のハードウェアに対応していましたが、Windows Phoneではハードウェアの要求仕様が厳格化され、表1に示すような高スペック端末のみをサポートします。

表1 Windows Phoneのハードウェア要求仕様

| | |
|---|---|
| ディスプレイ | WVGA (800 x 480) |
| タッチパネル | 4点以上のマルチタッチ |
| GPU | DirectX 9 対応 |
| センサー | A-GPS,加速度センサ,方位センサ，照度センサ，近接センサ |
| カメラ | デジタルカメラ |
| ハードウェアボタン | 3個（Start, Search, Back） |
| ネットワーク | 携帯電話，Wi-Fi |
| メモリ | RAM(256MB以上)とフラッシュメモリ(8GB以上) |

### 1.3 開発手法

従来のWindows Mobileでは、主に.Net Compact Frameworkを用いた.Netアプリケーションと、Windows APIを用いてC++で記述するネイティブアプリケーションの2つの開発手法が提供されていました。

一方、Windows Phoneでは、開発手法が刷新され、「Silverlight」と「XNA Framework」を用いる2つの手法が提供されます。Silverlightでは、xaml（eXtensible Application Markup Language）というXMLベースのマークアップ言語を用いて、GUIの外観／配置／イベントなどを定義し、ロジックをC#で記述します。開発環境としては主にVisualStudioを利用し、GUI関連のxamlコードの大半は自動生成されます。さらに、ベクタグラフィックスやムービーを容易に扱うことができたり、Microsoft Blend[3]という、Adobe Flashのようなデザイナー向けのxamlデザインツールも用意されており、柔軟なUI設計が可能です。

XNA Frameworkは、.Net Compact Frameworkにゲーム開発に特化した拡張ライブラリを追加した開発環境です。開発環境としては主にXNAStudioを利用します。こちらでは、原則グラフィックスを含む全てのコードを自分で記述する形になります。

このように、一般のアプリケーション（以下、アプリ）向けにはSilverlight、ゲームなどを作る場合はXNA Frameworkを利用する形になるようです。

### 1.4 互換性

従来のWindows Mobileでは、OSがアップデートされても下位互換性が保たれており、多くのアプリを継続利用可能でした。一方、Windows Phoneでは上述のように開発方法／動作方法が異なるため、従来のアプリは原則動作しません。

### 1.5 配布方法

従来のWindows Mobileでは、アプリは作者のWebサイトなどからインストーラー（Cab形式）をダウンロードして自由にインストールする形式が主流でしたが、Windows Phoneでは、iPhoneなどと同様に原則マーケットプレース（Windows Marketplace）からアプリをダウンロード&インストールする形式に変更されます。

このように、近年のiPhone／Androidなどのスマートフォン市場の変化を強く意識して、Windows Phoneではアプリの開発手法／配布手法などが大きく変化しています。次に、Windows Phone用アプリを開発する手順について詳しく紹介します。

## 2. 開発環境の準備

Windows Phoneの開発者向けに、ポータルサイト「Windows Phone Developer Site[4]」が用意されており、開発ツールのダウンロードやMarketplaceへの登録を行うことが出来ます。

まず、Windows Phone用の開発環境のダウンロード／インストールを行います。Developer Siteの、画面右上の画像リンク「April Reflesh Get it Now!」をクリックすると、「Windows Phone Developer Tools CTP - April Refresh」のダウンロードページ[5]へ移動します。次に、ページ下部の「VMX¥vm_web.exe (3.2MB)」の右側の「Download」ボタンをクリックすると、インストーラのダウンロードが開始されます（図1）。なお、インストーラの実行にはWindows 7 or Windows Vistaが必要です（Windows XPでは動作しません。）。vm_web.exeはファイルサイズが小さいですが、Webインストーラ*1であり、開発ツールとして以下の4つを含みます。

- Visual Studio 2010 Express for Windows Phone CTP
  Silverlightを用いて、一般的なアプリ開発
を行う際の統合開発環境です。
- Windows Phone Emulator CTP
  Windows Phone用のアプリをデバッグする
際のエミュレーターです。
- Silverlight for Windows Phone CTP
  Windows Phone用のSilverlightの実行環境のようです。
- XNA Game Studio 4.0 CTP
  XNA FrameWorkを用いて、主にゲーム開発に利用する統合開発環境です。

これらの開発ツールは、無償でダウンロードすることができます。アプリを一般に配布するためのMarketplaceへの登録には、$99の年会費がかかるようです。学生向けのアカデミックライセンスでは、年会費無料でアプリを2件まで配布できます。

## 3. VisualStudioの基本とHelloWorld

ここでは、VisualStudio2010 Express for Windows Phone CTP（以下、VisualStudio2010）とWindows Phone Emulator CTP（以下、Emulator）の基本的な使い方を、HelloWorldプログラムの作成を例として説明します。

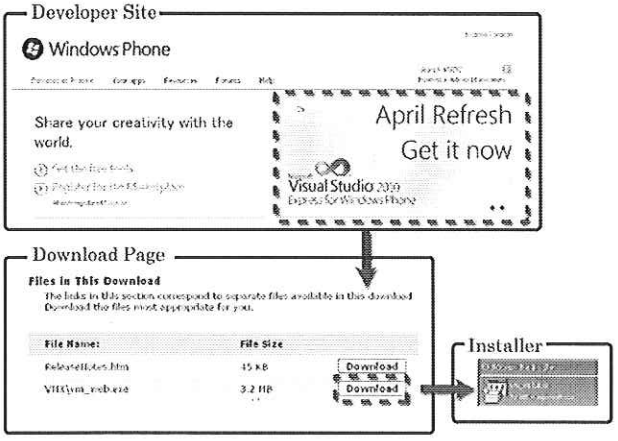


図1 Windows Phone開発環境のダウンロード

### 3.1 プロジェクトの新規作成

まず、Visual Studio 2010を起動します。デフォルトのインストール状況では、スタートメニュー→Microsoft Visual Studio 2010 Express → Microsoft Visual Studio 2010 Express for Windows Phoneをダブルクリックします。

VisualStudio2010の起動後、まずプロジェクトの新規作成を行います。「メニュー→File→New Project (or Start Page → New Project)」を選択すると、図2のようなダイアログが表示されます。このダイアログでは、主に以下の4つの項目について設定を行います。

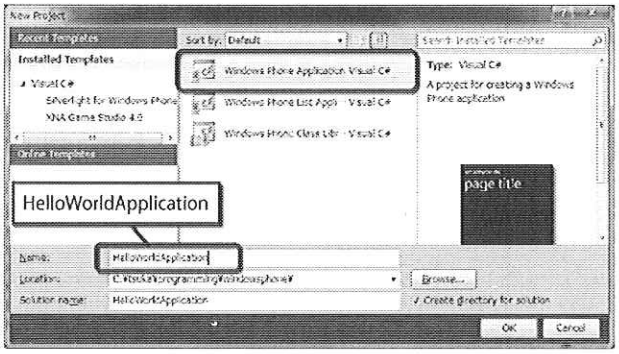


図2 新規プロジェクトの設定／作成

- Template：プロジェクトの種類を選択します。今回は一般的なアプリを製作するため、一番上の「Windows Phone Application Visual C#」を選びます。
- Name：プロジェクトの名前を入力します。今回は「HelloWorldApplication」とします。
- Location：プロジェクトの保存先です。デフォルトのままでも、任意に変更しても構いません。
- Solution name：ソリューション*2の名前です。プロジェクト名と連動して自動的に名前がつきます。

これらの設定を行い「OK」ボタンを押すと、プロジェクトの新規作成が完了し、図3のような画面が開かれます。ここではまず、VisualStudioの基本的な機能について簡単に説明します。

- デザイン編集画面
  GUIを視覚的に編集できます。

*1 インストール中に必要ファイルを随時Webからダウンロードする方式。

*2 一つ～複数のプロジェクトを含むVisualStudioでのファイルの管理単位。今回の例では一つのソリューションに一つのプロジェクトしか含まれないので、特に意識しなくても構いません。

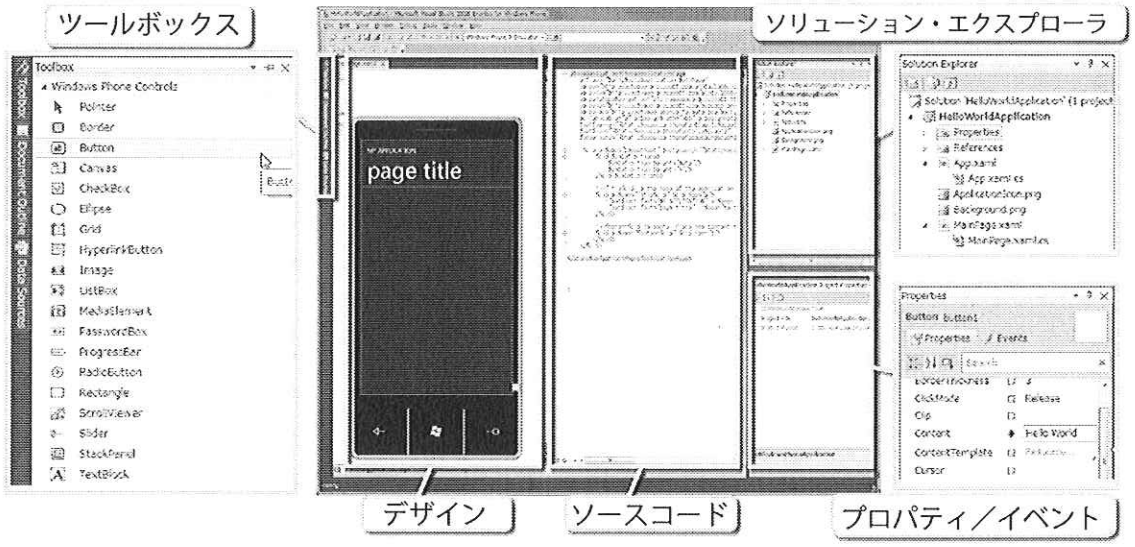


図3　VisualStudio2010の外観と機能

・ソースコード編集画面
デザイン編集画面で編集中のGUIのソースコード（*.xaml）や、ロジックを記述するC#のソースコード（*.xaml.cs）を確認／編集できます。
・ツールボックス
各コンポーネントをドラッグ&ドロップすることでデザイン編集画面に追加できます。
・プロパティ／イベント
デザイン編集画面で選択したコンポーネントのプロパティを編集したり、イベントを追加することができます。
・ソリューション・エクスプローラ
プロジェクトに含まれるファイルを確認したり、新規に追加することが出来ます。

プロジェクトを新規作成した状態で、ソリューション・エクスプローラを見ると、多数のファイルが自動的に追加されていることを確認できます（図3右上）。

この中で、開発者が直接編集するファイルは、「MainPage.xaml」「MainPage.xaml.cs」の二点となります。MainPage.xamlは、Windows Phoneアプリの外観（GUIの配置、デザイン、イベント定義など）の記述を行います。一方、MainPage.xaml.csは、各イベントに対応するロジックなどを記述します。

### 3.2　ボタンの配置／設定

ここでは、画面中央にボタンを配置し、ラベルとサイズを変更した上で、クリックイベントを追加します。

1. ツールボックスで「Button」コンポーネントを選択し、ドラッグ&ドロップでデザイン画面に配置します。
2. デザイン画面で配置したコンポーネントを選択して、プロパティを以下のように変更します（図3右下）。
   ・Content = "Hello World"
   ・Width = "Auto"
3. 最後に、ボタンをクリックした際のイベントを定義します。プロパティ／イベント画面の「Event」タブを押してイベント表示に切り替え、「Click」の行をダブルクリックします[*3]。なお、Buttonコンポーネントでは、Clickはデフォルトイベントなので、デザイン画面でコンポーネントを直接ダブルクリックする形でも構いません。

この時点で、MainPage.xamlには、以下のようなコードが追加されます。下線は今回の変更部分です。

```
<Grid x:Name="ContentGrid" Grid.Row="1">
    <Button Content="Hello World" Height="70"
     HorizontalAlignment="Left" Margin="141,43,0,0"
     Name="button1" VerticalAlignment="Top" Width="Auto"
     Click="button1_Click"/>
</Grid>
```

### 3.3　ロジックの記述

前述のようにClickイベントを追加すると、MainPage.xaml.csが自動的に開かれ、「button1_Click」という関数が追加されます。ここでは、デフォルトで配置される画面上部の2つのTextBlockのラベルを入れ替える処理を以下のように記述します。

[*3] Buttonコンポーネントでは、Clickはデフォルトイベントなので、デザイン画面でコンポーネントを直接ダブルクリックする形でも構いません。

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    textBlockListTitle.Text = "Hello World!";
    textBlockPageTitle.Text = "Hello Windows Phone!";
}
```

最後に、キーボードから「F5」キーを押すと、プログラムをコンパイルして、Emulator上でデバッグを行うことができます。図4に、プログラムをEmulatorで起動している様子を示します。中央に「Hello World」というボタンが表示され、ボタンをクリックすると、上部のTextBlockの文字が「Hello World!」に変化します。

Emulatorの下部には、実機のハードウェアボタンを想定した3つのボタンが並んでいます。左の「Back」ボタンを押すと、実行中のプログラムを終了してEmulatorがホーム画面に戻り、VisualStudioのデバッグモードも終了します。


図4 HelloWorldプログラムをEmulatorで実行

## 4. Webブラウザ

次に、多少高度な事例として、Webブラウザアプリの開発について紹介します。

### 4.1 プロジェクトの新規作成

プロジェクト名を「WebBrowserApplication」として、プロジェクトを新規作成します。他の設定はHelloWorldのケースと同一です。

### 4.2 コンポーネントの配置／設定

今回は、TextBox、Button、WebBrowserの3つのコンポーネントを利用します。TextBoxは、URLの入力／表示に利用します。Buttonは、入力したURLへと移動するトリガとして利用します。WebBrowserは、実際にWebページを表示します。以下に、作業手順をまとめます。

1. 前述のTextBox／Button／WebBrowserの各コンポーネントをツールボックスからデザイン画面へと配置します。レイアウトは、図5を参考にしてください。
2. 各コンポーネントのプロパティを以下のように変更します。
   - TextBox: textBox1
     - Text = http://mobiquitous.com/（任意）
   - Button: button1
     - Content = Go
3. 次に、TextBoxに表示されたURLをWebBrowserで表示するためのイベントを追加します。Buttonをダブルクリックして「Click」イベントを追加し、作成された関数「button1_Click」内に以下のようなコードを記述します。

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    webBrowser1.Navigate(new Uri(textBox1.Text,
                                 UriKind.Absolute));
}
```

この時点で、F5キーを押してEmulatorを起動すれば、TextBoxに入力されたURLを、Buttonを押すことでWebBrowser上に表示することができます*1。

### 4.3 画面回転への対応

次に、画面の描画方向が縦／横に変化した際に、各GUIコンポーネントのサイズや位置を自動調整する方法について説明します。これは、スマートフォンではおなじみの、加速度センサと連動して、端末の持ち方によって画面方向を切り替える機能に対応するものです。

具体的には、各コンポーネントの「Height」「Width」「HorizontalAlignment」「VerticalAlignment」という4つのプロパティを編集します。以下に、最終的なxmalのソースコードを記します。

```
<Grid x:Name="ContentGrid" Grid.Row="1">
  <TextBox Name="textBox1"  Width="Auto" Height="Auto"
   HorizontalAlignment="Stretch" VerticalAlignment="Top"
   Text="http://mobiquitous.com/"  Margin="2,6,71,0"  />
  <Button  Name="button1" Content="Go"  Width="86" Height="70"
   HorizontalAlignment="Right"  VerticalAlignment="Top"
   Margin="0,6,3,0" Click="button1_Click" />
  <browser:WebBrowser Name="webBrowser1"  Width="Auto"
   Height="Auto" HorizontalAlignment="Stretch"
   VerticalAlignment="Stretch" Margin="13,75,0,0" />
</Grid>
```

*1 本稿執筆時点のWebBrowserコンポーネントは、日本語表示には対応していませんでしたが、近い将来解決されると思われます。

### 4.4 URL／タイトルの更新

最後に、現在閲覧しているWebページのURLとタイトルを随時更新するコードを記述します。まず、画面上部の2つのTextBlockのプロパティを、以下のように変更します。

- TextBlock: textBlockPageTitle
  Text=Web Browser
- TextBlock: textBlockListTitle
  FontSize="36"

次に、WebBrowserを選択し、プロパティ／イベント画面から「Navigated」イベント（あるWebページの読み込みが完了したことを通知する）を追加します。MainPage.xaml.csに「webBrowser1_Navigated」という関数が追加されますので、ここに以下のようなコードを記述します。

```
private void webBrowser1_Navigated(object sender,
NavigatedEventArgs e)
{
  //現在の URL を更新
  textBox1.Text = e.Uri.ToString();
  //ページの全内容を文字列として取得
  string htmlContent = webBrowser1.SaveToString();
  //正規表現でページタイトルを抽出
  Match m = Regex.Match(htmlContent, "<title>(.+)</title>",
  RegexOptions.IgnoreCase);
  if (m.Success)
  {
   //ページタイトルを更新
   textBlockListTitle.Text = m.Groups[1].ToString();
  }
}
```

図5に、最終的なWebBrowserプログラムをEmulatorで動作させる様子を示します。


図5　WebBrowserプログラムの機能と外観

## 5. センサ

Silverlightを用いた開発では、Windows Phoneデバイスに内蔵されるさまざまなセンサも手軽に扱うことが可能になるようです。残念ながら、Emulator上ではセンサ類の動作を実験できないので、現時点ではコードを試す方法がありませんが、参考までにセンサの基本的な取り扱い方を紹介します。センサ関連のクラスは、GUI部品を持たないため、基本的にMainPage.xaml.csファイルに直接コードを記述します。以下、加速度センサとA-GPSを扱うための最小限のコードを紹介します。

```
//センサを扱うための名前空間
using Microsoft.Devices.Sensors;
…
//加速度センサオブジェクトの宣言
AccelerometerSensor accelerometer = null;
…
//加速度センサの開始／停止
void StartStopAccelerometer(){
 if (accelerometer == null)
 {
   //加速度センサ初期化
   accelerometer = new AccelerometerSensor();
   //センサデータ更新時のイベントハンドラを追加
   accelerometer.ReadingChanged += new EventHandler
                    <AccelerometerReadingAsyncEventArgs>
                    (accelerometer_ReadingChanged);
   //加速度センサ計測開始
   accelerometer.Start();
 }
 else
 {
   //加速度センサ携速停止
   accelerometer.Stop();
   accelerometer = null;
 }
}
//加速度センサの更新時のイベント処理
void accelerometer_ReadingChanged(object sender,
                    AccelerometerReadingAsyncEventArgs e)
  {
   //加速度センサの値を変数に格納
   double x = e.Value.X;
   double y = e.Value.Y;
   double z = e.Value.Z;
  }
```

```
//位置情報を扱う名前空間
using System.Device.Location;
…
//位置情報の更新を監視するクラス
GeoCoordinateWatcher watcher = null;
…
//位置情報計測開始
private void StartLocationService( )
{
 // GPS の精度を設定して初期化.
 watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
 //位置更新イベント発生の閾値（メートル）
 watcher.MovementThreshold = 20;
 //位置情報更新を通知するイベントハンドラを追加
 watcher.PositionChanged += new EventHandler
                <GeoPositionChangedEventArgs<GeoCoordinate>>
                (watcher_PositionChanged);
 //位置情報取得開始.
 watcher.Start();
}
…
//位置情報更新時のイベント処理
void watcher_PositionChanged(object sender,
                GeoPositionChangedEventArgs<GeoCoordinate> e)
{
  //緯度・経度を変数に格納
  double latitude = e.Position.Location.Latitude;
  double longitude = e.Position.Location.Longitude;
}
```

## 6. おわりに

本稿では、新しいスマートフォン向けプラットホームWndows Phoneについて、従来のWindows Mobileと比較したさまざまな変更点や、基本的な開発手順、及びいくつかの開発事例について紹介しました。VisualStudioを中心とした扱いやすい開発環境は、特にWindows系プログラマにとっては大きなメリットであり、iPhoneやAndroidと比較しても一日の長があるように感じます。実機の登場は2010年末とまだ少し先になりますが、期待して発表を見守りたいと思います。

## 参考文献


[1] Zune HD: http://www.zune.net/en-US/products/zunehd/

[2] 神原啓介, 塚田浩二: ソーシャル顔アイコン, ソフトウェア科学会 WISS2009 論文集, pp.53-56, 2009. available: http://sappari.org/faceicon.html

[3] Microsoft Blend: http://www.microsoft.com/japan/products/expression/

[4] Windows Phone Developer Site: http://developer.windowsphone.com/

[5] Windows Phone Developer Tools CTP -April Refresh: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=cabcd5ed-7dfc-4731-9d7e-3220603cad14

[6] Code Samples for Windows Phone: http://msdn.microsoft.com/en-us/library/ff431744%28VS.92%29.aspx


## 著者紹介


**塚田 浩二（つかだ こうじ）：**

1977年生。2000年慶應義塾大学環境情報学部卒業。2005年同大学大学院政策・メディア研究科博士課程修了。同年、独立行政法人産業技術総合研究所研究員。2008年4月より、お茶の水女子大学特任助教。ユビキタス・インタフェースの研究・開発に従事。プロトタイピング、ガジェット収集・発明に興味を持つ。博士（政策・メディア）。

Web：http://mobiquitous.com/