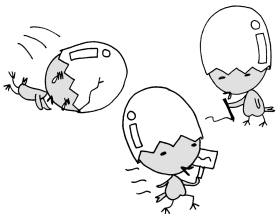
作って学ぶ VISUAL BASIC .NET

瀬戸 遥
SETO, Haruka
http://www.big.or.jp/~seto/
http://hp.vector.co.jp/authors/VA006682/



第7回 いろいろな四角形を描画してみよう

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ❑ Other:

Level

★ ★ ★ ★ ★

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥TAMAHIYOディレクトリに収録しています。

¥FIGURE
今回のサンプル

引き続きグラフィックスに挑戦

前回は、WindowsのGDI＋と.NET Frameworkクラスライブラリの Graphicsクラスを使って、文字をグラフィックスとして描画してみました。

引き続き、今月もこのGDI＋の機能を使って、パターンやグラデーションでの塗りつぶしといった、グラフィックス描画の基本的なプログラミングにチャレンジしてみましょう。

プログラムで描画する内容

今回は、PictureBoxコントロールに四角形を描画します。

描画の種類は、以下のとおりです。

①単純な枠線のみの四角形
②単色で塗りつぶした四角形
③クロスハッチのパターンで塗りつぶした四角形
④グラデーションで塗りつぶした影付き四角形
⑤写真をテクスチャに使用して塗りつぶした文字

これらの図形描画を、フォームに配置したボタンを押すことで実行します。ユーザーインターフェイスは、図1のように簡単なものにします。フォームにPictureBoxコントロールとButtonコントロールを5つ配置するだけです。

前回、グラフィックスを描画するには、「描画対象オブジェクトのPaintイベントプロシージャで処理を実行する」と説明しました。

今回も同様に、PictureBoxコントロールのPaintイベントプロシージャでこれらの図形を描画しますが、ひとつのイベントプロシージャでいろいろな図形を描画するために、変数をひとつ用意し、**Select**

図1：今回のサンプルの実行画面

Caseという条件分岐ステートメントを利用して実行します。

また、Paintイベントプロシージャ内でコードがゴチャゴチャにならないように、各図形の描画処理をひとつひとつ独自のSubプロシージャに作成し、それを呼び出す形にしてみます。

プログラムのメインの処理

プログラムのメインの処理は、ボタンのClickイベントプロシージャと、PictureBoxコントロールのPaintイベントプロシージャで行ないます。

●変数宣言とClickイベントプロシージャ

まず、モジュールレベルの変数をひとつ用意します。この変数は、描画実行用の各ボタンが押されると、そのボタンの番号を格納します。今回は“1”から“5”までの数字しか格納しませんから、最小のメモリ消費量となるByte型（0から255までの整数を格納する）で宣言します。

```
Dim Pos As Byte
```

次に、各Buttonコントロールの Clickイベントプロシージャで、この変数「Pos」に数字を代入し、PictureBoxコントロールのRefreshメソッドを実行して、強制的にPaintイベントを発生させます。

```
Private Sub Button1_Click（略） Handles Button1.Click
 Pos = 1
 Me.PictureBox1.Refresh()
End Sub

（略：Button2～4のClickイベント）

Private Sub Button5_Click（略） Handles Button5.Click
 Pos = 5
 Me.PictureBox1.Refresh()
End Sub
```

●Paintイベントプロシージャの処理

次に、PictureBoxのPaintイベントプロシージャを作成します。

```
Private Sub PictureBox1_Paint（略） Handles
PictureBox1.Paint

End Sub
```

Paintイベントプロシージャでは各図形の描画を実行しますが、これは変数Posの値とSelect Caseステートメントを使用し、値が“1”であれば枠線のみの四角形を、値が“2”であれば単色塗りつぶしで四角形を描画する、というように変数の値で処理を分岐するようにします。

```
Select Case Pos
  Case 1
    RectDraw(e)
  Case 2
    SolidRect(e)
  Case 3
    HatchRect(e)
  Case 4
    GradRect(e)
  Case 5
    TextureString(e)
End Select
```

各描画処理をこのプロシージャ内に全部記述してしまうと、かなりの量のコードとなり処理がわかりにくくなってしまいます。そこで、各描画処理ごとに独自のSubプロシージャを作成し、これを呼び出す形にしています。