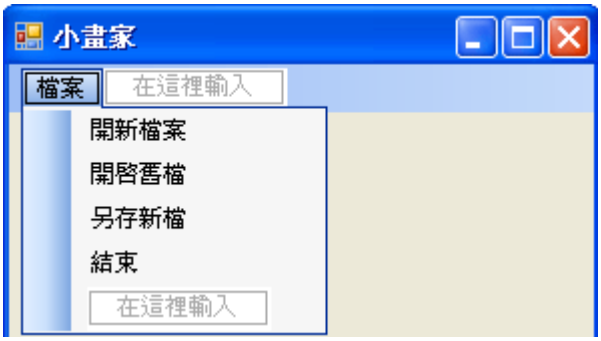
第 6 章　簡易小畫家

簡介：

上一章我們學會了一些影像物件處理的基本概念，這一章我們將繼續學習用 VB 程式繪圖的技巧。比較複雜的是在.NET Framework 架構下，繪圖的操作層次分得很細，且多半必須使用程式碼操作，包括繪圖物件的建置，畫筆及筆刷顏色與粗細的設定等等，對初學者而言有點複雜，就請耐心學習了。

6-1 建立開檔存檔功能

這個程式與上一單元一樣，主要還是由功能表(MenuStrip)操作，請先到工具箱叫出一個 MenuStrip1 物件。這次我們是要畫圖的，所以除了開啟舊檔之外，還必須有新增空白圖檔，並回存作品(影像)的功能。如果你還記得之前的單元就會知道，另存新檔是比儲存檔案較簡單的動作，本單元就省略完整儲存功能的介紹，只作「另存新檔」就好了。請設計如下：



開啟影像檔案如前一單元一樣，需要用到 PictureBox 與 OpenFileDialog 物件，本單元也要作存檔動作，需要 SaveFileDialog，請到工具箱找出它們放到表單上。PictureBox 的 SizeMode 屬性請設為 AutoSize，本單元以畫圖為主，不作縮放處理，進出的檔案都照原始大小顯示就好了！其次，OpenFileDialog 與 SaveFileDialog 物件的 Filter 也都設為「*.jpg|*.jpg」，表示我們只處理 jpg 檔案的意思。當然你是可以設定處理更多格式的，如有需要請參考秀圖軟體單元。最後請將表單的 WindowState 設定為 Maximized(放到最大)，一般影像軟體都是希望畫面越大越好的！

仿照正版小畫家，請將 PictureBox1 放到視窗的左上角，如果你無法順利將 PictureBox1 貼齊到主功能表的下方，請查看一下 MenuStrip1 的 Size->Height(高度)值，將 PictureBox1 的 Location->Y 值(頂部座標，相當於執行階段的 Top 屬性)設為與 MenuStrip1.Height 一樣就貼齊了！

首先看開啟舊檔的程式如下，與前面單元大致相同，第一行是說：如果使用者按下 Cancel(取消)鍵，就離開(Exit)此副程式(Sub)的意思，接著就是按照選擇的檔案開啟載入。

```
Private Sub 開啟舊檔ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles 開啟舊檔ToolStripMenuItem.Click
      If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.Cancel Then Exit Sub
      PictureBox1.Load(OpenFileDialog1.FileName)
End Sub
```

另存新檔是將目前 PictureBox1 裡面的影像(Image 屬性)依使用者在 SaveFileDialog1 中指定的檔名存到磁碟之中。因為我們之前已經將 Filter 設定為只處理 jpg 檔案，所以鍵入檔名時不需要再寫副檔名了！譬如你寫存入檔案「X」，就會變成 X.jpg 圖檔。請注意到本單元是要畫圖的，所以儲存的檔案可能是某圖檔被你叫出來塗鴉之後的「作品」，最好還是另存新檔，不要覆蓋原圖比較好。

```
Private Sub 另存新檔ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles 另存新檔ToolStripMenuItem.Click
      If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.Cancel Then Exit Sub
      PictureBox1.Image.Save(SaveFileDialog1.FileName, Imaging.ImageFormat.Jpeg)
End Sub
```

上述程式中的參數 **Imaging.ImageFormat.Jpeg** 是必要的！就是指示電腦要以 Jpeg 檔案格式儲存，Jpeg 與 jpg 意義相同。可能是為了嚴謹吧？程式不會因為我們在 SaveFileDialog1 物件中講過一次要使用 jpg 就自動在此以 jpg(Jpeg)格式存檔。當然這也表示你可以在此寫其他格式，如 BMP，那就會有一個名為 jpg 實為 bmp 格式的怪檔案了！應該有些軟體會被你欺騙而打不開該檔案吧？有可能！但其實各種格式的影像在檔案內容的最前面都有識別字元，不會只靠副檔名來分辨的！你能騙到的應該只是程式使用者而已。

6-2 建立新影像

小畫家程式最常用的應該不是開啟舊檔來玩，而是開新檔案繪圖，甚至程式一開始就會開個空白的新檔案給我們使用！所以開新檔案與代表程式起始的 Form_Load 事件應該共用下面建立新影像的程式：

```
Private Sub 開新檔案ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles 開新檔案ToolStripMenuItem.Click, Me.Load
      PictureBox1.Image = New Bitmap(800, 600)
      Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
      g.Clear(Color.White)
End Sub
```

上面的程式首先請注意到標題列的 Handles 之後除了「開新檔案」功能的 Click 事件之外，還用逗點接續了一個 Me.Load 事件名稱，Me 是 VB 程式預設自稱「本表單」的代名詞，所以 Me.Load 就相當於 Form_Load 事件了！表示程式一開始「也會」執行這個副程式。在此的 Me.Load 應該是你寫好「開新檔案」副程式之後自己加上去的！

接著是宣告一個新的，長 800 點寬 600 點的影像給 PictureBox1。請回憶上一單元教過的概念：PictureBox 物件是一個「相框」，預設是沒有照片(Image)的，所以在此必須宣告影像物件給它使用，否則後續就無法畫圖了！這樣還不夠，預設新宣告的影像是無色(透明)的！

所以即使你給了影像，看起來 PictureBox1 裡面還是沒有東西！當然，如果你設計時給了 PictureBox1 物件一個背景色，你會「以為」有影像，但是對電腦來說那是「相框底」的顏色，你還是無法在這上面畫圖的！

結論是：你必須給這個新影像一個底色，一般就是白色囉！不過讓無色的影像變成白色，這已經算是「繪圖」動作了，所以必須宣告繪圖物件 Graphics，這個物件的概念是它依附於一個影像物件，可以接受指令去畫很多東西，譬如直線、方塊、圓圈甚至是刷出底(背景)色等等。當然你叫它畫的東西就會直接呈現於它依附的影像物件，如果這個影像物件就是 PictureBox1.Image 當然你就會看到結果了！上面程式最後一行的 g.Clear(Color.White)就是使用繪圖物件 g 將 PictureBox1.Image 的底色刷成白色(Color.White)的意思了！試試看！打開程式你就會有一張白紙了！

6-3 塗鴉

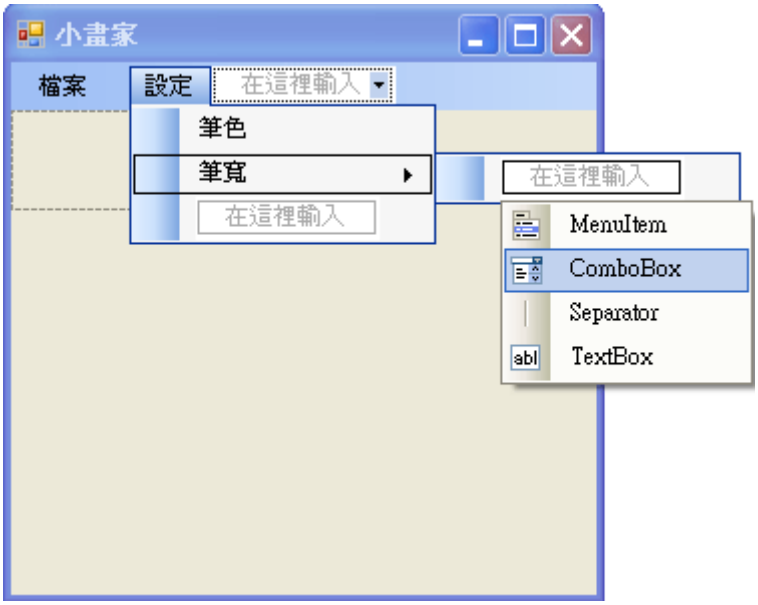
有了空白的畫圖紙第一件事當然就是想給它亂塗鴉一下的啦！要寫塗鴉的程式必須先分析一下細部動作：先是滑鼠對著畫面的一個點(起點)壓下左鍵，這就是 MouseDown 事件；接著使用者繼續壓住滑鼠左鍵移動滑鼠到新的一點(終點)，就是 MouseMove 事件，每次的移動我們都要立即畫一個小線段，從起點到終點，畫完之後將終點變成起點，滑鼠繼續移動時就會繼續畫隨後的小線段。這些小線段連起來就是你的塗鴉了！程式碼如下：

```
Dim x0 As Integer, y0 As Integer

Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseDown
        x0 = e.X
        y0 = e.Y
End Sub

Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseMove
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
            g.DrawLine(Pens.Black, x0, y0, e.X, e.Y)
            x0 = e.X
            y0 = e.Y
            PictureBox1.Refresh()
        End If
End Sub
```

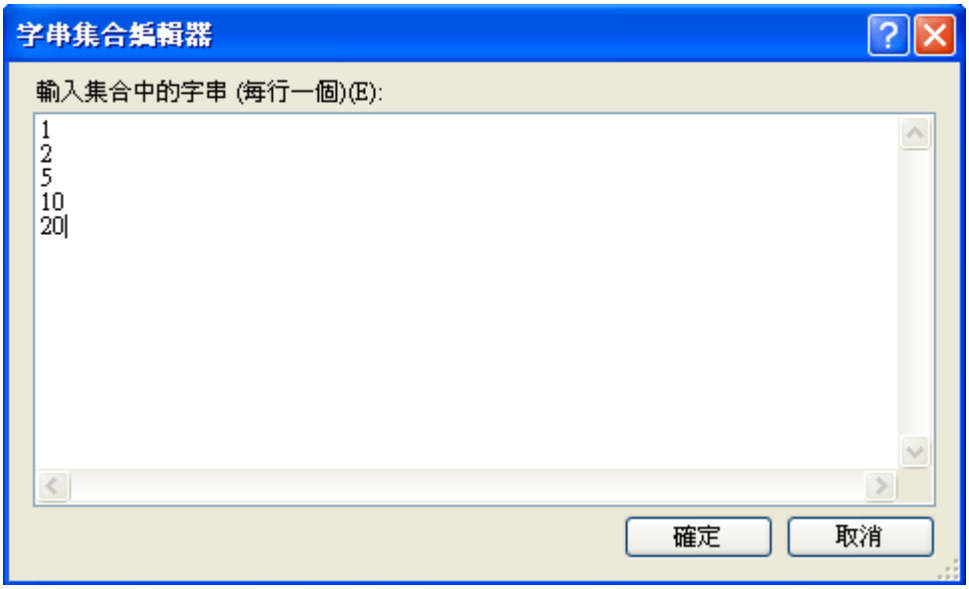
上面程式碼中的 x0 與 y0 就是繪圖起點的座標，當 MouseDown 事件發生時先記住這個位置，所謂 e.X 與 e.Y 就是滑鼠目前位置的座標。接下來滑鼠移動了！我們首先宣告繪圖物件 g，當然是要它指向目前的 PictureBox1.Image 這個影像了！請繪圖物件 g 畫一個線段(DrawLine)，使用黑色的筆(Pens.Black)，起終點座標為(x0, y0)-(e.X, e.Y)，畫完之後將終點變成新的起點(x0 = e.X，y0 = e.Y)。試試看！塗鴉動作應該可以執行了！當然別忘了可以使用剛剛寫的存檔程式將作品典藏哦！

6-4 調整畫筆的顏色與粗細

剛剛的程式是使用系統預設的黑筆，粗細也是預設的一個像素點寬，如果要用自己設定的筆就要宣告一個「筆物件」來繪圖。像繪圖物件一樣，其實筆也可以隨時宣告隨時用，你不必像使用實體的筆一樣，擔心這樣會很「浪費」！哈哈！請先在功能表加入以下功能項目：



請注意到上圖最右邊我們是將新選項的右端選單拉開加入一個 ComboBox 這是一個下拉式選單，建立好之後編輯其 Items(項目)集合屬性如下圖，當然項目就是畫筆粗細的選擇，你可以自由發揮。寫好這個集合之後也要將 ComboBox 的 Text 屬性改為 1，這是筆的預設粗細值。



還有選擇顏色時需要一個 ColorDialog 對話方塊，也請準備好，位置在工具箱的「對話方塊」分類。這樣就可以開始寫程式了！請雙擊「筆色」進入程式碼頁面，寫程式如下：

```
Private Sub 筆色ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 筆色ToolStripMenuItem.Click
    ColorDialog1.ShowDialog()
End Sub
```

上述程式只是讓選色方塊顯示出來供使用者選色而已，至於選畫筆粗細的介面可以直接操作，連開啟程式都不必寫了！但這樣還是不會產生改變畫筆的效果，程式必須修改，使用這兩個操作的結果去「製作」需要的畫筆來執行繪圖。請將塗鴉部分的 MouseMove 事件副程式修改如下：

```
Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseMove
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
            Dim p As New Pen(ColorDialog1.Color, Val(ToolStripComboBox1.Text))
            g.DrawLine(p, x0, y0, e.X, e.Y)
            x0 = e.X
            y0 = e.Y
            PictureBox1.Refresh()
        End If
End Sub
```

與原來程式只有兩行的差異，一是宣告一支新畫筆(New Pen)，筆色來自 ColorDialog1 的 Color 屬性，筆的粗細則來自下拉式選單(ToolStripComboBox1)的文字(Text)所代表的數字(Val)；接著就用這支筆(p)置換原來系統預設的黑筆(Pens.Black)。如果 ToolStripComboBox1 的超長名稱對你來說有些困擾，建議你到表單點選該物件，再到屬性欄中拷貝它的 Name 屬性來寫程式，就保證不會打錯字了！你要直接將它改名也可以的。試試看！現在你可以任意改變筆色與粗細了！

6-5 畫直線

小畫家可以畫直線、矩形或圓形等形狀，與塗鴉不同的是他們在拖曳過程中只會顯示一條線或一個形狀，不會一移動就一直畫，到最終放開滑鼠時也只有最後的一個圖案會留下！這幾個功能的核心技巧很相似，本單元只示範拉直線的作法，為了簡化操作介面就使用滑鼠「右鍵」表示拖曳直線，左鍵保持為塗鴉之用。

這個程式技巧的關鍵在於當使用者按下滑鼠時要記住當時的畫面(存為一個影像物件)，每次滑鼠移動時都會再次拷貝這個影像(沒有直線)為繪圖目標，並從滑鼠壓下的那一點畫一條直線到目前滑鼠位置，這樣每一個 MouseMove 動作都只會畫出一條線的影像，因為下次移動時又是從沒有這條線的原來影像(MouseDown 的時候)重新複製來繪製。這樣一來，每一次移動滑鼠的小移動就要進行一次圖像轉移，好像讓電腦很操勞，會不會因此產生停格呢？早期的 VB6 是有可能，現在的程式內部影像管理效能好多了，何況螢幕畫面本來就每秒更新數十次，放心吧！絕對沒事的！

包括塗鴉在內的繪圖動作程式現階段請修改如下：

```
Dim x0 As Integer, y0 As Integer
Dim bg As Bitmap
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseDown
        x0 = e.X
        y0 = e.Y
        bg = New Bitmap(PictureBox1.Image)
End Sub


Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseMove
        Select Case e.Button
            Case Windows.Forms.MouseButtons.Left
                Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
                Dim p As New Pen(ColorDialog1.Color, Val(ToolStripComboBox1.Text))
                g.DrawLine(p, x0, y0, e.X, e.Y)
                x0 = e.X
                y0 = e.Y
                PictureBox1.Refresh()
            Case Windows.Forms.MouseButtons.Right
                Dim bf As New Bitmap(bg)
                Dim g As Graphics = Graphics.FromImage(bf)
                Dim p As New Pen(ColorDialog1.Color, Val(ToolStripComboBox1.Text))
                g.DrawLine(p, x0, y0, e.X, e.Y)
                PictureBox1.Image = bf
        End Select
End Sub
```

上述程式拉直線的部分在 Case…Right，首先是宣告一個名為 bg 的點陣圖物件(料型態名稱為 Bitmap)等同於 Image 物件，作為暫存 PictureBox1.Image 的地方。在 MouseDown 事件時用 bg 記住當時的畫面(PictureBox1.Image)；在每次 MouseMove 時判斷如為「左鍵」按下就執行塗鴉，「右鍵」按下則執行畫直線。畫直線的動作包括：複製一個影像物件 bf，內容與 bg 相同，讓繪圖動作畫在這個 bf 中，再將 bf 賦予 PictureBox1.Image 屬性，就會看到直線了！但是下一個 Move 動作時又會從與 bg 相同的影像開始，所以直到放開滑鼠才會讓最後的直線固定下來。當然，下一次再執行 MouseDown 想進一步繪圖時就會包含這條直線，因為它已經是背景(PictureBox1.Image)的一部分了！因此你就可以連續畫出很多條直線囉！

如果你想如正版小畫家一樣，畫出方塊或橢圓需要改變的只是 DrawLine 的那一部分指令，改成 DrawRectangle 就是矩形；DrawEllipse 就是橢圓。如果要畫成實心的方或圓形就改成 FillRectangle 或 FillEllipse 即可，甚至有個 FillPie 的指令可以畫出扇形的形狀。

6-6 使用工具提示顯示顏色值

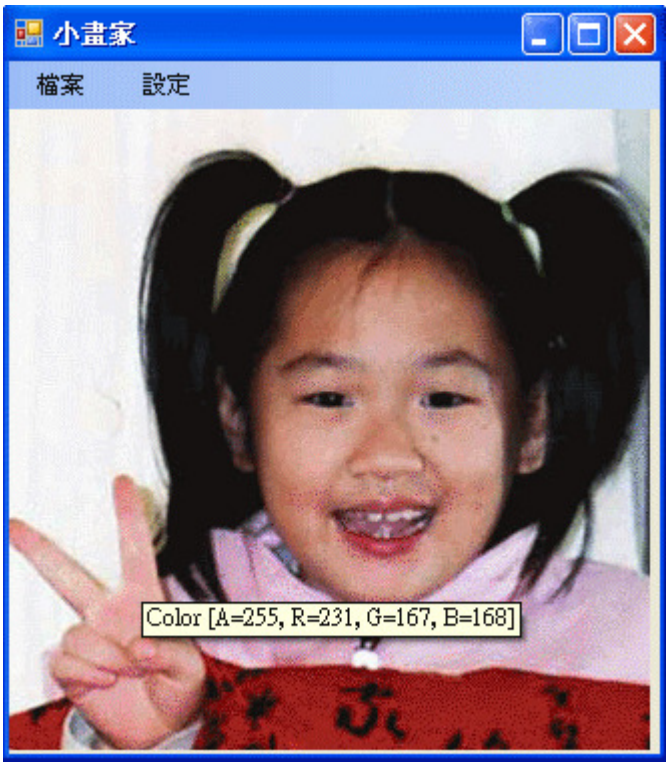
在此介紹一個工具箱「通用控制項」分類中的 ToolTip 物件，它的直接翻譯就是「工具提示」。相信所有用過電腦的人都會知道滑鼠移到某個物件上面時，常常會有個小小的黃色文字方塊「溫柔」的浮現出來，告訴你這個東西是幹嘛用的等等資訊，那就是工具提示了！請先到工具箱的「一般控制項」中找出它加入表單(ToolTip1)。請注意！工具提示不是猛跳出來的那種視窗哦！那個東西叫做訊息框(MsgBox)稍後也會介紹的。

在此我們希望使用工具提示顯示滑鼠所在位置的顏色資訊。典型數位影像的結構是每個像素點都是由 RGB，也就是紅綠藍三原色以不同的亮度值組成，每種色光亮度的值介與 0~255 之間。要取得一個像素點的顏色資訊是使用 GetPixel 的指令將某一像素點從 Image 物件中擷取出來，當然必須指定是影像上 X 與 Y 座標分別為多少的那一點。

我們想將顯示程式寫在 MouseMove 事件中，之前此事件副程式已經用於畫直線及塗鴉，它們分別已佔用了滑鼠的右與左鍵，這裡就將顯示顏色資訊的程序寫在「未按鍵」(None)的狀況，程式碼如下：

```
Case Windows.Forms.MouseButtons.None
    bg = New Bitmap(PictureBox1.Image)
    Dim C As Color = bg.GetPixel(e.X, e.Y)
    ToolTip1.SetToolTip(PictureBox1, C.ToString)
```

首先宣告 bg 與目前的 PictureBox1.Image 相同，接著在 bg 內滑鼠所在的位置取出一個像素點 C，所謂的像素點(Pixel)在數位影像中的實質意義就是一個點的顏色資訊，所以資料型態宣告為 Color。最後呼叫 ToolTip1 設定工具提示(SetToolTip)於 PictureBox1 物件之內，提示內容為該處像素點(C)的內容(ToString)。試試看畫面應該如下：



在此我們有點偷懶的直接將顏色變數轉成字串(C.ToString)，你會發現轉出來的除了 RGB 值之外還有一個 A 值，這是與顏色「透明度」相關的值！255 表示完全不透明，而 0 就是完全透明了！事實上也可以使用 C.R 取得紅色，C.G 取得綠色，C.B 取得藍色等與語法來操作這些資訊。

GetPixel 的使用其實是非常重要的一個里程碑，這表示我們知道如何取得影項中任何一

個像素點的詳細資訊，事實上你還可以用相對的 SetPixel 函數將顏色改變之後再「放」回影像裡面去！有了它們，任何高級的影像處理程式都可以依據這個邏輯使用 VB 程式來寫了！不要再道聽塗說影像處理程式非用 C++或 MatLab 等程式寫才可以了！其實經過「.NET Framework」的良好整合，現在的 VB 與 C++速度是完全一樣快的！加上語法容易，我認為 VB 才是寫影像處理程式的最佳語言呢！至少我多年來都是用它作影像辨識研究的！

6-7、進階挑戰

一、如何模仿畫直線的方式繪製出方塊或圓圈？
　　提示：使用繪圖物件 Graphics 的 DrawRectangle, DrawEllipse, Fill…等功能。

二、如何製作調整畫紙大小的功能？
　　提示：可用記事本單元的 GroupBox 小視窗方式輸入影像的寬高值。

三、如何製作復原功能？
　　提示：使用多個背景影像物件陣列記住每個步驟的影像畫面，需要復原時回貼即可。

課後閱讀

相信多數同學學到本單元一定會被 Bitmap，Graphics 甚至 Pens，Brush 等等東西弄得很煩。畫幾條線而已，怎麼弄得那麼複雜啊？確實如此，在美好的 VB6 時代，畫一條線真的只要一個指令：**Form.Line(X1,Y1)-(X2,Y2)**，就搞定了！連我都好懷念。但是在 VB 與其他微軟語言整合在「.NET Framework」程式平台之後(2002 年)，繪圖的語法模式就遷就原來的 Visual C 語言模式變成現在這個樣子！原因當然不是因為好用，而是執行效率高！同樣的繪圖動作，很奇妙的，複雜語法的新版 VB 會比舊版的 VB6 快上很多倍(約 3~10 倍)！詳細的原因不容易說清楚，但是新的，也就是比較複雜的語法其實較為接近電腦真實執行繪圖的過程，我們就當是學得比較專業一些吧？目前我們也沒得選擇，必須這樣來寫繪圖程式。可以作的事情是在此先努力弄清楚這幾個物件的意義吧！

一、影像容器、Bitmap 與 Graphics、Pen 與 Brush

上一單元課後閱讀介紹過 Bitmap 物件，它是被繪圖的目標，如同一張圖畫紙，或者已經照好的相片！Graphics 呢？它好像是個繪圖機、畫家或者說繪圖的「代理人」，宣告它時必須提到它是依附於(用來畫)哪一個 Bitmap 物件的。也就是說如果沒有指定的 Bitmap，Graphics 其實是不能單獨的存在！下行程式就是說明 g 物件依附於 PicturBox.Image 這個影像。

```
Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
```

接下來繪圖的動作都必須指定這個 Graphics(g)去執行，但是操作個別像素點的 GetPixel 與 SetPixel 指令例外，它們可以直接讀寫像素點(Pixel)的內容，不必用到 Graphicss。Graphics 物件必然的會將所有的繪圖動作畫到它依附的那個 Bitmap 物件之上。但是煩人的陷阱是這個畫好的 Bitmap 物件還是只存在電腦的記憶體裡面，如果沒清楚地交代要顯示在哪一個影像「容器」(如 PictureBox)裡面，或者忘了更新容器內的影像內容，使用者還是會看不到結果的！下面的程式碼就是在作這些事情：

```
PictureBox1.Image = New Bitmap(800, 600)

PictureBox1.Refresh()
```

上面的 PictureBox1 就是影像的「容器」，最典型的容器當然是 PictureBox，其他還有表單的 BackgroundImage，Label 物件也有 Image 屬性，總之可以載入影像(有 Image 屬性)的物件都可以當影像容器。總之，請記得影像物件畫好後一定要放到某個容器內，而且影像用程式改變之後還得寫個指令更新(Refresh)一下才行，否則還是看不到結果的！

二、Pen 與 Brush

Pen 是筆，Brush 是刷子！要畫出線條狀的東西(Draw)得用筆，要將一個區塊塗滿(Fill)一種顏色就要用刷子！這就是它們基本的使用邏輯。因此如果繪圖物件要 Draw 一個方塊，表示是畫一個方塊形狀的邊框，就應該用 Pen，如同本單元的內容，Pen 可以自己宣告定義

筆色與筆寬，但是預設就會有許多的筆可以用，它們的名字會是這樣：Pens.Black，Pens.Red…等等。因此畫個黑框矩形的程式就會是這樣：

g.**Draw**Rectangle(Pens.Black, x1, y1, width, height)

要畫個實心的(Fill)黑方塊呢？程式就是：

g.**Fill**Rectangle(Brushes.Black, x1, y1, width, height)

如果想自己定義筆刷的顏色呢？很奇怪的！必須宣告的物件不是 Brush 或 Brushes，而是「**SolidBrush**」！程式如下：

Dim B As New SolidBrush(Color.Yellow)

g.**Fill**Rectangle(B, x1, y1, width, height)

確實有夠麻煩，但是懂得以上幾個東西的基本意義與用法之後，希望你以後要寫繪圖程式時不再覺得這麼摸不著頭緒！很遺憾的是目前多數市面書籍似乎都講不太清楚這幾個物件的用法，確實讓很多人因此放棄了使用 VB 寫好繪圖程式的企圖心，又在說必須使用 C++ 才能繪圖了！