

第 6 章 秀圖軟體

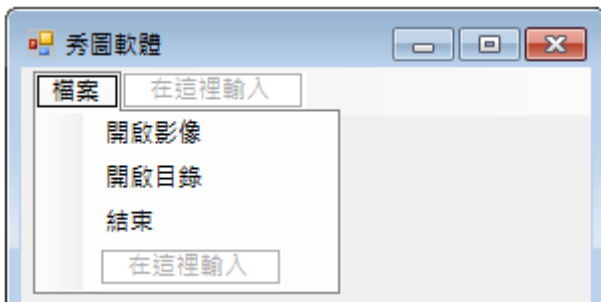
簡介：

這一章要介紹如何製作一個簡單的秀圖軟體，主要功能包括選擇開啟影像檔案，縮放以及旋轉翻轉影像的功能，過程中你將學習到操作影像物件及動態配置版面的基本概念。接著還會介紹投影片播放的功能，你可以選擇一個目錄，按鍵盤依序播放其中的影像檔案，這裡將學習到檔案目錄與檔案物件集合的基本操作方式。影像是視窗程式中不可或缺的最佳女主角，一個完全無圖的程式必定非常枯燥乏味，所以學好如何用 C#處理影像是一定要的啦！

6-1 建立功能表與狀態列

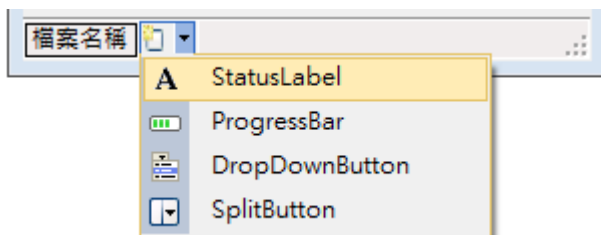
[建立主功能表]

這個程式與記事本單元一樣，主要還是由功能表(MenuStrip)操作，請先到工具箱叫出一個 MenuStrip1 物件。這次我們需要的項目較少，可以不必使用預設樣版，就直接由設計頁面直接填入以下的幾個項目。其中「結束」的程式碼請參考前面的記事本單元。



[建立狀態列]

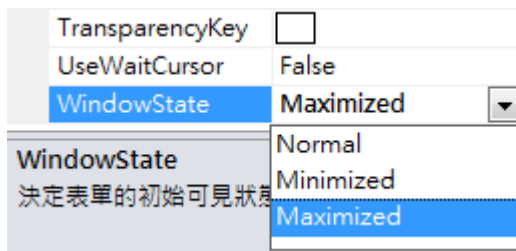
還有我們希望在視窗下方顯示出目前展示的圖檔名稱，這個依附在一般表單下方的物件稱為狀態列(StatusStrip)，在工具箱的「功能表與工具列」分類可以找到，也請加入表單，並請在其中加入一個標籤(StatusLabel)，如下圖所示。需要注意的是加入的標籤物件名字好長(toolStripStatusLabel1)不易記住，而且寫程式時它不隸屬於包住它的 statusStrip1 而是獨立於表單之內的！初學者寫程式時常常會找不到它。在此請先將他的 Text 屬性改成「檔案名稱」，否則程式未選檔案前顯示像「toolStripStatusLabel1」這樣的文字一定會讓使用者感到迷惑的。



6-2 建立影像顯示框

[設定表單的預設大小]

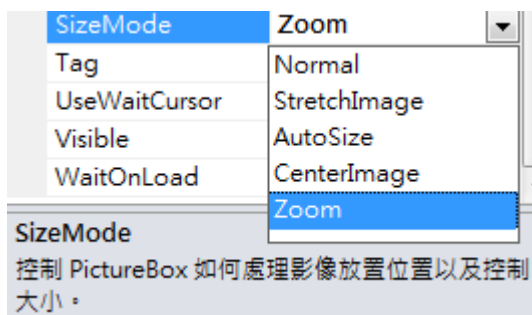
秀圖軟體就是要將影像從磁碟檔抓到程式之內，C#顯示圖檔的標準物件是 `PictureBox`，在工具箱的「通用控制項」分類之下可以找到。由於我們並不知道使用者開啟的影像會有多大，而且我們希望影像都能盡量放在螢幕的中央，所以設計階段去移動 `PictureBox` 的位置或調整它的大小並無意義，但是先將表單(不是 `PictureBox` 哦!)放到最大則是一個不錯的選擇! 所以請先點選表單本身，到它的 `WindowState` 屬性處改為 `Maximized`(最大化)，這樣程式一開始表單就會盡量放大到填滿整個螢幕。如下圖：



其他兩個選項中的 `Minimized` 是指縮到最小，也就是我們常常作的動作，將某個暫時不操作的視窗程式收到視窗下方的工具列，但不是關閉程式。`Normal` 則是讓視窗維持設計時的大小。

[設定影像盒的顯示模式]

接著請置入 `pictureBox1` 物件並將它的屬性 `SizeMode` 改為 `Zoom`，如下圖。



這個屬性 `SizeMode` 的意義是選擇載入影像時顯示的模式，`Normal` 是完全不改變邊框，且保持原始影像大小，所以影像太大時會無法全部顯示；`StretchImage` 是讓影像縮放符合(塞入)現有的邊框；`AutoSize` 是自動縮放邊框以符合影像實際的大小；`CenterImage` 是不改變影像大小直接讓它置於相框中間；`Zoom` 則類似 `StretchImage`，影像會隨邊框縮放，但是可以保持原始影像的長寬比例。一般秀圖軟體縮放時通常不希望扭曲長寬比(變成哈哈鏡)，所以在此我們選擇使用「`Zoom`」模式。

6-3 讀入與顯示影像

[開啟影像的程式]

要選擇開啟影像檔案，如同記事本單元必須先置入一個 `openFileDialog1`(工具箱的對話方塊)，為了讓它只能選擇圖檔，應該將 `Filter` 屬性寫成：「影像檔案|*.jpg;*.gif;*.bmp」，表示只處理這三種常見副檔名的影像檔案，中間用分號(;)區隔。順便將預設的 `openFileDialog1` 檔名刪除，接著請雙擊功能表的「開啟影像」寫入事件副程式如下：

```
private void 開啟影像ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK){
        pictureBox1.Load(openFileDialog1.FileName);
    }
}
```

上面程式使用 `PictureBox` 的 `Load` 方法載入檔案，語法很簡單，但有時候會失靈！譬如用上述程式碼試圖載入 `GIF` 動畫時就會當掉！此時可以嘗試比較複雜的語法，先將圖檔以 `Bitmap` 物件載入再指定給 `PictureBox` 的 `Image` 屬性就可以了！原因應該是更低階程式設計的差異所致，去問微軟的工程師吧！

```
private void 開啟影像ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK){
        Bitmap bmp =new Bitmap(openFileDialog1.FileName);
        pictureBox1.Image=bmp;
    }
}
```

試試看，你已經可以執行程式開啟影像了！包括 `gif` 動畫哦！但是你應該不會滿意的，因為影像會縮進你原本沒有刻意調整的 `pictureBox1` 之內，而且位置也不會剛好置中。這時候就必須設法寫出自動調整 `pictureBox1` 的「大小」以及「位置」的程式了！

6-4 顯示影像與置中

[自訂副程式]

因為這是兩組會多次重複的動作，所以我們要將它們分離出來作成獨立的**副程式**，後續如縮放影像等等的程式就不必一直重複寫一樣的程式碼了！雖然拷貝程式碼很方便，但是如果有時候這些重複的動作必須修改時，就必須同時改很多個地方，非常麻煩又容易遺漏出錯，所以將會重複使用的程式碼盡量寫成副程式是一個良好的習慣。

[調整影像至原始大小]

首先看載入影像後調整為原始影像大小的副程式如下，請記得這不是事件副程式，必須自己手動鍵入程式標題，當然 `Visual Studio` 還是會一路有提示訊息幫助你的。這些打字時會自動出現的提示稱為 `IntelliSense`(智慧感知)，如果有時候提示沒有出現可以按下鍵盤的 `Alt+→` 就會強迫它出現了。

```
private void SizeImage()
{
    pictureBox1.Width = pictureBox1.Image.Width;
    pictureBox1.Height = pictureBox1.Image.Height;
}
```

程式碼很簡短，但是透露出很重要的概念！在此，pictureBox1 應該被視為現實世界的一個相片框(影像容器)，其中置放的「照片」就是 pictureBox1.Image！這個照片的寬與高(Image.Width & Image.Height)代表原始影像大小，請注意！即使稍後我們會將影像縮放，這兩個值也是固定不變的！它的意義是**原始影像**像素點的長與寬的點數。至於標題列的 void 原意是「空」的意思，表示此副程式沒有回傳值。在 C 語言慣例中每個副程式都**應該**有回傳值，譬如一個 Boolean 函數(True or False)、數字、字串甚至陣列等等。如果不需要回傳值也必須交代它是**空**的值(void)。

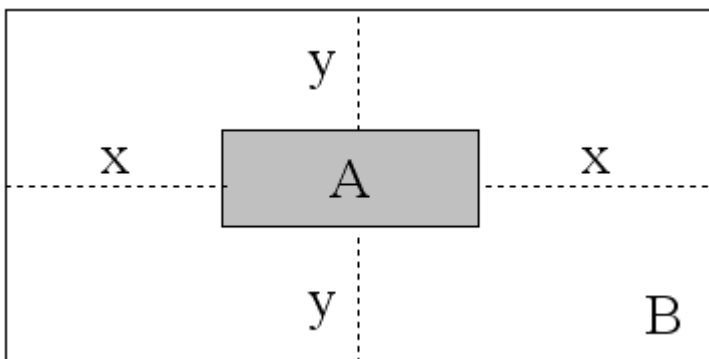
「縮放」在此的意義其實是改變「相框」的大小，因為相框被設定為 Zoom，所以它有類似凹凸透鏡的作用，可將原始影像在顯示時縮放大小。上面的副程式是希望相框設定為與原始影像一樣大，所以就使用原始影像的長寬為相框的長寬了！或許聰明的同學會發現：其實設定 SizeMode=AutoSize 就會有一樣的效果！但是這樣後面的縮放與旋轉程式就不太好寫了。你可以試試看不同的程式寫法！

[影像置中的程式]

接下來是讓影像框置中的副程式如下：

```
private void CenterImage()
{
    pictureBox1.Left = (this.ClientSize.Width - pictureBox1.Width) / 2;
    pictureBox1.Top = (this.ClientSize.Height - pictureBox1.Height) / 2;
}
```

為何如此寫？請先看看下面的示意圖，如果一個 A 物件要放到 B 物件之內，而且位置要置中，它應有的座標位置(x, y)應該像這樣：



其中 x 須為 B 的寬度減去 A 的寬度除以 2，對照程式碼 A 就是指 pictureBox1，B 就是

視窗(this)可以加入物件的矩形區域(this.ClientSize)，所以 $x=(B.Width-A.Width)/2$ 。y 座標的計算也類似。

接下來我們只要修改開啟影像的程式碼，載入影像後再呼叫上述兩個副程式如下，就可以將影像以原始大小正確顯示於視窗中央了！呼叫副程式就是直接寫副程式名稱，甚至在智慧感知提示中已經會出現它們。但是初學者常常出現的錯誤是大小寫寫錯，以及忘了加上小括號，必須記得這是程式的『動作』也就是物件導向程式設計中所稱的一個「方法」，小括號是必須的！還有就是要注意副程式寫的位置必須與其他事件副程式在同一階層，因為大括號很多，許多初學者會將自訂副程式寫到其他事件副程式內部或 Form1 類別之外，都是不可以的！

```
private void 開啟影像ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Load(openFileDialog1.FileName);
        SizeImage();
        CenterImage();
    }
}
```

6-5 位置與長寬屬性

[有關物件的位置與形狀的基本屬性]

在此說明一下有關調整物件位置與大小相關的屬性，首先必須熟悉至少下面四個屬性：

Left：左緣座標，用來調整物件的水平位置，相當於於屬性視窗的 Location->**X**。

Top：頂部座標，用來調整物件的垂直位置，相當於屬性視窗的 Location->**Y**。

Width：物件的寬度，相當於屬性視窗的 Size->**Width**。

Height：物件的高度，相當於屬性視窗的 Size->**Height**。

設計階段要調整位置大小，可以直覺地使用滑鼠在設計頁面拖曳即可，還有格式對齊工具可以使用，好像不認識這些屬性名稱好像也沒關係？但是在程式執行階段要使用程式碼動態控制時，關係可就大了！

[唯讀的位置相關屬性]

既然有 **Left** 屬性代表物件的左邊緣，那麼有沒有 **Right** 屬性代表右邊緣呢？有的！雖然屬性視窗中看不到，打程式碼時的提示項目就會出現 **Right** 的選項。同樣的，**Top** 屬性的相反應該是 **Bottom**(底部)，這個屬性也是有的！少了它們寫程式時會很麻煩，譬如 A 物件的右邊必須寫成 $A.Left + A.Width$ ，下邊緣要寫成 $A.Top + A.Height$ ，很囉唆！也因此讓複雜的位置移動程式更易於出錯。

不過請記住！**Right** 與 **Bottom** 屬性是**唯讀**的！就是不能寫 $A.Bottom=200$ 這樣的程式

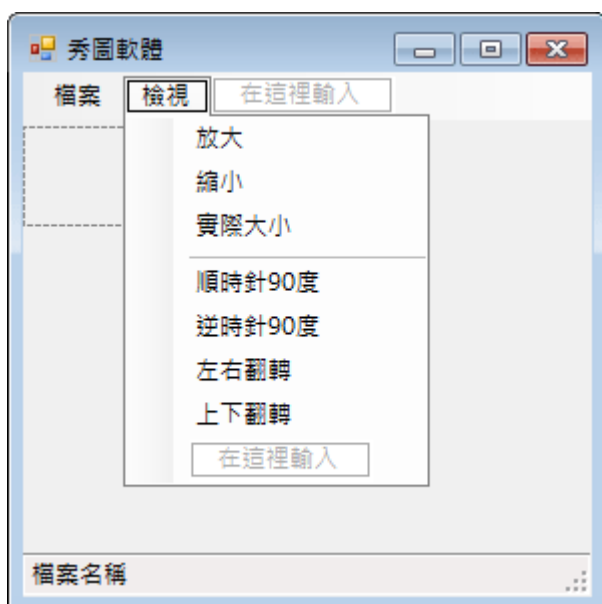
碼！要改變物件的位置還是只能改變 Left 及 Top 屬性，Right 與 Bottom 的值永遠是被動的自 Left, Width, Top, Height 等等四個屬性計算得到的，只是設計軟體會隨時自動幫你算好(Right 與 Bottom)而已。

[位置屬性的別名]

還有一個會困擾初學者的地方是在屬性視窗中看不到上述的 Left 等屬性，其中 Right 與 Bottom 是唯讀的，不在屬性視窗表示不可調整是合理的，但是 Left 與 Top 應該要出現才對呀？事實上是名稱變了！可能為了方便歸類，Left 屬性會顯示成 Location(位置)→X，Top 就變成 Location→Y 了！這兩種名稱是相通的，程式碼中也可以使用 Location.X=200 的語法。

6-6 縮放與旋轉影像

請先在功能表建立以下項目，直接在功能表的空格編輯項目時，要產生格線只需要鍵入一個減號「-」即可！在此我們用一條格線稍微區分縮放與旋轉翻轉兩類的處理功能。



[縮放影像的程式]

以下是放大處理功能的程式碼：

```
private void 放大ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Width = Convert.ToInt32(pictureBox1.Width*1.2);
    pictureBox1.Height = Convert.ToInt32(pictureBox1.Height * 1.2);
    CenterImage();
}
```

[整數與浮點數的轉換]

在此碰到一個資料型態轉換的問題，在 C#語言中資料型態定義是很嚴謹的，譬如一般物件的寬與高是 32 位元的整數(Int32)，但是如果我們希望將此數字乘以 1.2 倍時，1.2 是有

小數的數值，電腦預設會將具有小數的數字視為 64 位元長度的**倍準數(double)**，所以我們將 pictureBox1 的寬度乘以 1.2 時電腦會自動將答案變成比較精準也就是比較大的資料型態 double，此時又要將此值(答案)指定給 pictureBox1.Width(資料型態 int32)時就會出現資料簡化的情況(64 位元實數變成 32 位元整數)。C#在此會要求一定要明確的使用轉換函數，就是 Convert.ToInt32 了！很多已經會寫 VB 的老手碰到這類型態轉換的問題都會很氣惱，因為 VB 通常可以自己看著辦！C#卻很堅持一定要設計者自己寫得非常清楚。

此外，當影像框大小改變時中心點也會偏移，所以必須再呼叫一次 CenterImage 將影像置中！接下來縮小影像的程式與放大類似，實際像素功能更簡單，就是與開影像時動作一樣，以原始影像大小為大小(SizeImage)，然後置中(CenterImage)即可。程式碼如下：

```
private void 縮小ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Width = Convert.ToInt32(pictureBox1.Width / 1.2);
    pictureBox1.Height = Convert.ToInt32(pictureBox1.Height / 1.2);
    CenterImage();
}

private void 實際大小ToolStripMenuItem_Click(object sender, EventArgs e)
{
    SizeImage();
    CenterImage();
}
```

[旋轉影像的程式]

至於旋轉或翻轉就不是處理「相框」，而是要處理相框內的「照片」了！在此就是 pictureBox1 物件內的 Image 屬性，必須讓 Image 真的轉動才行！還好，旋轉與翻轉影像在 C#中有現成的指令可以用，所以程式碼相當精簡，先試試看順時針 90 度旋轉程式如下：

```
private void 順時針90度ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image.RotateFlip(RotateFlipType.Rotate90FlipNone);
    pictureBox1.Refresh();
}
```

首先是將 Image 用 RotateFlip(旋轉翻轉)指令加上 Rotate90(旋轉 90 度)FlipNone(不翻轉)的參數執行影像的旋轉動作，此時通常不會自動顯示 Image 物件的改變，所以還必須加入一個「Refresh」指令。但是除非你的影像剛好是正方形，上面的旋轉結果一定會有點怪異，就是 pictureBox1 的長與寬還是一樣的，影像又一定會擠入現有框線內，且遵守長寬比保持一樣的 Zoom 模式，所以顯示的影像就會變小了！要顯示正確的結果必須將影像的長與寬也隨著影像的旋轉而調換，而且還要再執行一次置中。完整程式碼如下：

```

private void 順時針90度ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image.RotateFlip(RotateFlipType.Rotate90FlipNone);
    int w = pictureBox1.Width;
    pictureBox1.Width = pictureBox1.Height;
    pictureBox1.Height = w;
    CenterImage();
    pictureBox1.Refresh();
}

```

[讓兩個數字互換的程式]

「調換」長寬的程式技巧是先宣告一個變數記住寬度，再將寬度變成與高度相同，最後將剛剛記住的寬度賦予高度的屬性，有點繁瑣！但程式內的兩個變數內容要交換時，這是標準程序！接下來的逆時針 90 度程式可以完全仿照，只要將 Rotate 值變成 270 度就可以了！此地的 Rotate 都是指順時針旋轉，所以旋轉 270 度就等於逆時針轉 90 度了！

```

private void 逆時針90度ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image.RotateFlip(RotateFlipType.Rotate270FlipNone);
    int w = pictureBox1.Width;
    pictureBox1.Width = pictureBox1.Height;
    pictureBox1.Height = w;
    CenterImage();
    pictureBox1.Refresh();
}

```

[翻轉影像的程式]

旋轉是 Rotate，翻轉就是 Flip，左右翻轉算是 X 方向，就是 FlipX；上下翻轉時就是 Y 方向的 FlipY 了！不過翻轉時長寬不必改變，也就無須重新置中，所以程式碼簡短一點，兩種翻轉程式如下：

```

private void 左右翻轉ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image.RotateFlip(RotateFlipType.RotateNoneFlipX);
    pictureBox1.Refresh();
}

private void 上下翻轉ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image.RotateFlip(RotateFlipType.RotateNoneFlipY);
    pictureBox1.Refresh();
}

```

6-7 投影片播放

[開啟一個目錄播放其中影像]

秀圖軟體常常可以使用上下或者 PageUp / PageDown 按鍵連續播放同一個目錄內的影像

檔案，我們也來作看看。事實上這個功能必須先取得整個目錄的檔案資訊，但不是載入全部的影像哦！只是檔名與大小等資訊而已。如果我們如前面的程式，只選擇一個檔案開啟，理論上電腦也可以知道該檔所在的目錄，用程式去取得整個目錄的資訊並開始作連續播放的動作，但是程式比較囉嗦，我們暫且將這個功能作為本單元的進階挑戰。在此先使用直接選擇目錄進行投影片播放的程序來設計我們的功能。

[使用選取目錄的對話方塊]

與前一單元的記事本程式相似，接下來的程式必須使用到一般專案非預設載入的檔案處理程式庫 **System.IO**，所以請先將此命名空間以 **using** 指令加入專案程式碼的最前面區塊中。接下來在表單上置入一個「目錄選擇」對話方塊 **FolderBrowserDialog**，位置也是在工具箱的「對話方塊」分類之中。接著在我們設計的「開啟目錄」功能內寫程式如下：

```
FileInfo[] F;
private void 開啟目錄ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        DirectoryInfo D = new DirectoryInfo(folderBrowserDialog1.SelectedPath);
        F = D.GetFiles("*.jpg");
        if (F.Length > 0)
        {
            pictureBox1.Load(F[0].FullName);
            SizeImage();
            CenterImage();
        }
    }
}
```

[DirectoryInfo 與 FileInfo 物件]

上述程式先在公用區域(副程式框架的外面)宣告全域變數**陣列 FileInfo[] F**。如字面的意義，**FileInfo** 表示檔案的資訊，而且是一「組」，而非一個檔案的資訊，這在程式語言中稱為陣列，**C#**陣列的宣告方式是在資料型態之後加上一對**方括號**(或稱中括號)，之後是陣列的名稱。當我們要取用陣列成員時就是以 **F[0]**, **F[1]**, **F[2]**...等名稱呼叫，括號內的數字稱為「陣列索引」，規定須從 **0** 開始；也因此最後一個成員的索引值就是總數減一！此地宣告時方括號內為空值表示成員個數未定，要看後面的程式來決定實際成員個數。

開啟目錄的程式與開啟檔案相似，只是最終選取的是一個目錄而非個別的檔案。當使用者選好目錄之後，上述程式宣告一個名為 **D** 的 **DirectoryInfo**(目錄資訊)變數，並從 **D** 中擷取副檔名為 **jpg** 的影像檔案集結成為 **F** 陣列。如果 **F** 陣列的成員數(**F.Length**)大於零(有圖檔)就將第一個成員(**F[0]**)載入 **pictureBox1**，當然也是要調整大小(**SizeImage**)與位置(**CenterImage**)。

上面的程式已經可以成功的顯示目錄中的第一張圖(如果有 **jpg** 檔案的話)，但如何翻頁呢？事實上就是要將剛剛顯示的 **F[0]**依序變成 **F[1]**, **F[2]**...等等。這必須要有一個公用變數 **N** 來記住目前播放到第幾張圖片？接著就可以用鍵盤的 **PageUp** 與 **PageDown** 增減這個 **N** 值。如果超過合理範圍，譬如變成負數時要輪回到最大的索引值；相反的，如果大於最大索

引就要輪回到 0 !

[顯示上下一頁的程式]

在此要寫鍵盤程式，所以必須先將表單的 `KeyPreview` 屬性設為 `True`(預設為 `False`，請參考小小電子琴單元)，接著在表單的 `KeyDown` 事件寫程式如下：

```
int N = 0;
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.PageDown:
            N += 1;
            if (N > F.Length - 1) { N = 0; }
            break;
        case Keys.PageUp:
            N -= 1;
            if (N < 0) { N = F.Length - 1; }
            break;
    }
    pictureBox1.Load(F[N].FullName);
    SizeImage();
    CenterImage();
}
```

上述程式中先宣告一個全域的整數變數 `N`，接著 `F.Length` 是 `F` 陣列的成員個數，也就是可播放影像檔的總數，`F.Length-1` 是最大的索引值(總影像數減一)。先用鍵盤增減 `N` 值，並檢查 `N` 可能超出範圍的情況，太大就回歸到 0，太小就跳到最大索引值。到了可以播放的「下一張」之後，由後續程式依索引載入圖檔，一個簡單的投影片播放程式就完成了！而且個別的圖片依舊享有前面製作的縮放、旋轉與翻轉的功能。

6-8 狀態列顯示檔案資訊

[顯示播放影像檔名]

前面我們已經準備好了狀態列上的一個標籤要顯示檔名，卻一直還未使用。簡單的作法是在每次 `pictureBox1.Load` 指令執行時將檔名寫到標籤 `toolStripStatusLabel1` 之中，開啟單一檔案時是這樣：

```
pictureBox1.Load(openFileDialog1.FileName);
toolStripStatusLabel1.Text = openFileDialog1.FileName;
```

開啟目錄第一張圖時是這樣：

```
pictureBox1.Load(F[0].FullName);
toolStripStatusLabel1.Text = F[0].Name;
```

開啟「下一張」圖片時是這樣：

```
pictureBox1.Load(F[N].FullName);  
toolStripStatusLabel1.Text = F[N].Name;
```

[顯示影像總數與目前序號]

除了檔名其實我們常常還會想知道已經放到總共幾張投影片的第幾張了(如 7/21)? 其實這就是剛剛程式中的變數 $N+1$ ，以及 `F.Length` 兩個數字，程式碼的寫法應該修改如下：

```
toolStripStatusLabel1.Text = F[N].Name + "(" + (N + 1).ToString() + "/" + F.Length.ToString() + ")";
```

其中 N 是索引值， $+1$ 之後就是第幾張，`F.Length` 則是可播放的總張數。請注意在這裡 `ToString`(轉字串)是必要的，否則文字"/"與數字相加是會讓程式當掉的！

6-9 進階挑戰

一、如何顯示目前播放影像的長與寬點數？(難度：低)

提示：顯示 `pictureBox1.Image.Width` 與 `Image.Height`。

二、如何選擇單一檔案後就可以有按上下一頁播放的功能？(難度：高)

提示：設法自檔案名稱取得其目錄名稱。

課後閱讀

一、影像物件的概念

在掌握影像程式的運作中最重要的概念是必須認識「影像物件」的意義！影像物件如果以變數宣告的方式來看，它應該是一個稱為 **Bitmap**(點陣圖)的資料類別。這個 **Bitmap** 物件是很多包含顏色資訊的像素點的集合，本單元的 `pictureBox1.Image`，事實上就是一個 **Bitmap** 物件。當影像以電子檔案的形式存在磁碟中時，通常會有各種壓縮格式，如 **JPG** 或 **GIF** 等等，但是被讀入(`pictureBox1.Load`)程式中時檔案會被翻譯成完整的點陣圖物件(存在記憶體中)。這個物件接下來就可以被程式旋轉、翻轉或者縮放。寫程式時要觀念清楚最重要的就是搞清楚某行程式到底是在處理 **Bitmap**、**PictureBox**(相框)或者影像檔案(如*.jpg)，這些物件在程式執行過程中意義都是不同的！

事實上本單元的處理都還只是整張影像的變化，真正的影像處理還可以將點陣圖中的個別點一一拿出來處理，譬如將彩色變黑白；重新取樣變成更多(或更少)點數的點陣圖等等。下一單元則是會介紹使用繪圖物件來改變點陣圖的內容，就是改變點陣圖內像素點的顏色以達到繪圖的效果。

二、目錄與檔案物件

本單元還介紹了一個有趣的新技術！就是將整個磁碟「目錄」的資訊讀入程式，再一一取出裡面所有圖檔的資訊來播放影像。在我們的程式邏輯中磁碟目錄的資訊也是一個「物件」(**DirectoryInfo**)，如同前面的影像檔案與影像物件(**Bitmap**)的關係一樣，我們的邏輯是先用類似檔案總管的對話方塊選取目錄，程式再宣告一個 **DirectoryInfo** 物件指向這個磁碟目錄，當然此物件就會讀取目錄裡面的資訊，包括有哪些檔案及這些檔案的摘要資訊等等，但不包括個別檔案的實質(圖片)內容。程式碼像這樣：

```
DirectoryInfo D = new DirectoryInfo(folderBrowserDialog1.SelectedPath);
```

接下來取得檔案群資訊的寫法也很特別，檔案資訊一如目錄資訊也是一個物件類別，稱為 **FileInfo**。既然是多個檔案，就必須是一個陣列，而非單一物件了，所以宣告時加個括號表示它是有多個成員的，但是一開始並不確定有幾個檔，所以括號內是空的：

```
FileInfo[] F;
```

最後使用目錄資訊物件(**DirectoryInfo**)的 **GetFiles**(取得檔案)方法，將目錄資訊內的檔案個別資訊灌到 **F** 陣列中就完成了檔案資訊的「分裝」工作了！

```
F = D.GetFiles("*.jpg")
```

這種語法相當簡潔，但是必須有清楚的觀念加上一點點想像力才能完全理解。有了這個範例為基礎，你可以想出來如果只讀一個檔案資訊到個別的 **FileInfo** 物件的程式如何寫嗎？應該這樣就可以了：

```
FileInfo F = New FileInfo(檔案路徑)
```

既然是檔案「資訊」，想當然耳的由這個物件 **F** 可以知道很多這個檔的資訊：

名稱(路徑)：**F.FullName**

檔案大小：**F.Length**

副檔名：**F.Extension**

.....

最後你可以猜一猜 **F.Directory** 的意思嗎？這可是解決本單元進階挑戰二的關鍵哦！