第 14 章我的工具箱

簡介：

相信同學們都已經很孰悉在設計階段使用工具箱產生新的物件，並替它們設計外觀與事件副程式。上一個單元也學到如何由程式主動產生很多磚塊物件，但是更為進階的遊戲程式常常需要在程式執行中由使用者主導產生新的物件，這些物件通常也必須具有事件副程式以回應使用者的操作，譬如至少要能被拖曳移動位置才行！

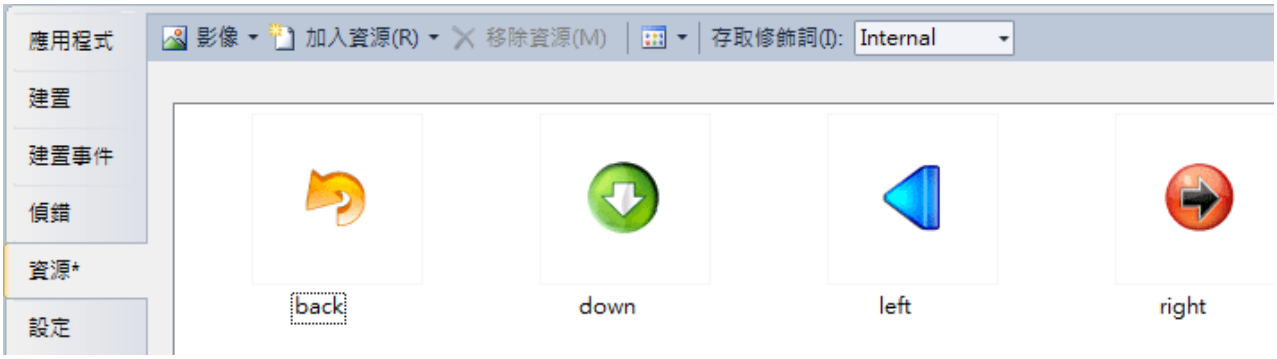
本單元的內容就是介紹如何寫出可以在程式執行中動態產生物件，且具備事件副程式的程式！我們將建立一個模擬許多軟體中工具箱的程式，使用者選擇物件後可以複製、拖曳、縮放，乃至旋轉與翻轉，這也是某些影像處理軟體中所稱「印章物件」的程式技術。

14-1 建立種子物件

[匯入圖案資源檔]

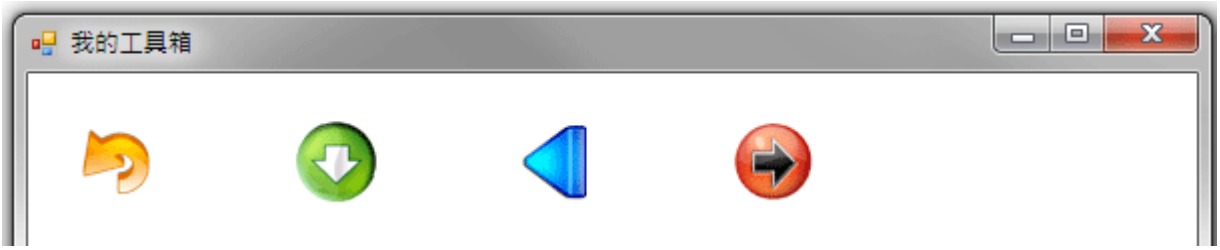
此專案最先要建立幾個圖形物件作為種子物件，類似工具箱或者印章物件的樣板，當圖形被選取後，使用者可以點擊表單空白處產生此物件的複製實體，這些複製物件預設就會有被拖曳的功能，同時還要具有跳出式選單對這個物件進行縮放、旋轉乃至刪除等等影像處理功能。

建立專案後請開啟方案總管中的 Properties 項目，進入專案屬性頁，選擇資源頁籤，選擇「加入資源」→「加入現有檔案」。加入四個圖檔如下(back, down, left & right)。當然你可以匯入其他圖案作為範例，但是最好不要太大，以免縮放測試時不方便；而且最好不要完全對稱，以免翻轉時看不出變化；同時記得動畫在目前的 C#版本可以載入，但是旋轉與翻轉後會變成靜態圖片，失去動畫特性。



[建立種子物件]

將表單標題修改為『工具箱』，置入四個 PictureBox，分別載入上述四個資源圖檔，SizeMode 屬性設為 AutoSize，使圖檔顯示為原始大小。大致如下圖：



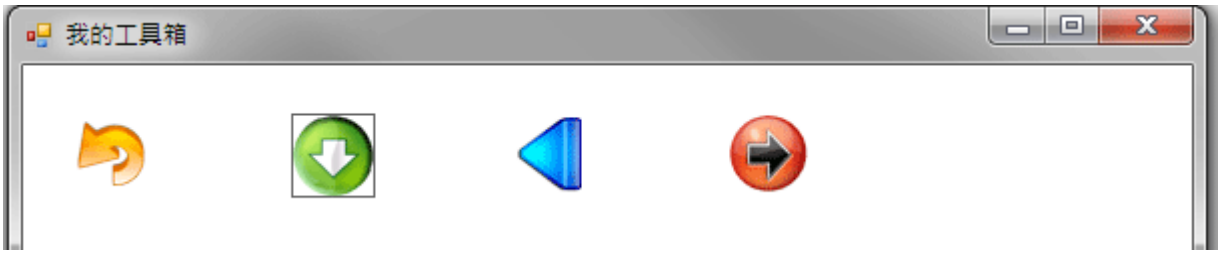
14-2 選取種子物件的程式

[物件被選取狀態的顯示]

一般工具箱操作的第一個動作是選取物件，被選的物件也必須有被點選的圖案反應。在此首先雙擊 pictureBox1，寫入 pictureBox1_Click 的事件副程式，並在 Form_Load 事件中加入 pictureBox2~pictureBox4 的 Click 事件共用這個副程式的程式碼，亦即點選任何一個 PictureBox 時都會執行同樣的程式：

```
PictureBox P;
private void pictureBox1_Click(object sender, EventArgs e)
{
    foreach (PictureBox c in this.Controls)
    {
        c.BorderStyle = BorderStyle.None;
    }
    P = (PictureBox)sender;
    P.BorderStyle = BorderStyle.FixedSingle;
}

private void Form1_Load(object sender, EventArgs e)
{
    pictureBox2.Click += new System.EventHandler(this.pictureBox1_Click);
    pictureBox3.Click += new System.EventHandler(this.pictureBox1_Click);
    pictureBox4.Click += new System.EventHandler(this.pictureBox1_Click);
}
```

上述程式首先公告一個 PictureBox P 變數，用以紀錄被點選的 PictureBox 物件(sender)，接著將表單(this.Controls)上所有 PictureBox 的邊框樣式(BorderStyle)都變成沒有(None)。唯獨被點選的影像 P 被設為有單線的邊框(BorderStyle.FixSingle)。這段程式讓物件被選取時有一個邊框明顯標示，同時程式內部也以全域變數 P 記住被選者，稍後處理時就可以據此進行。下圖為執行程式時第二圖被選的狀態。



14-3 複製物件

[用 Form1_MouseDown 事件複製物件]

我們希望的操作過程是點選圖案之後，接下來在「表單空白處」點下就可以在該處複製出一個一模一樣的物件。因此複製的程式應寫在 Form1_MouseDown 事件中。請注意！如果將程式寫在 Form1_Click 事件中，因為 Click 事件無法傳遞滑鼠的座標(沒有 e.X 與 e.Y 的參數)，因此將無法決定物件複製的位置，所以在此必須使用 MouseDown。程式碼如下：

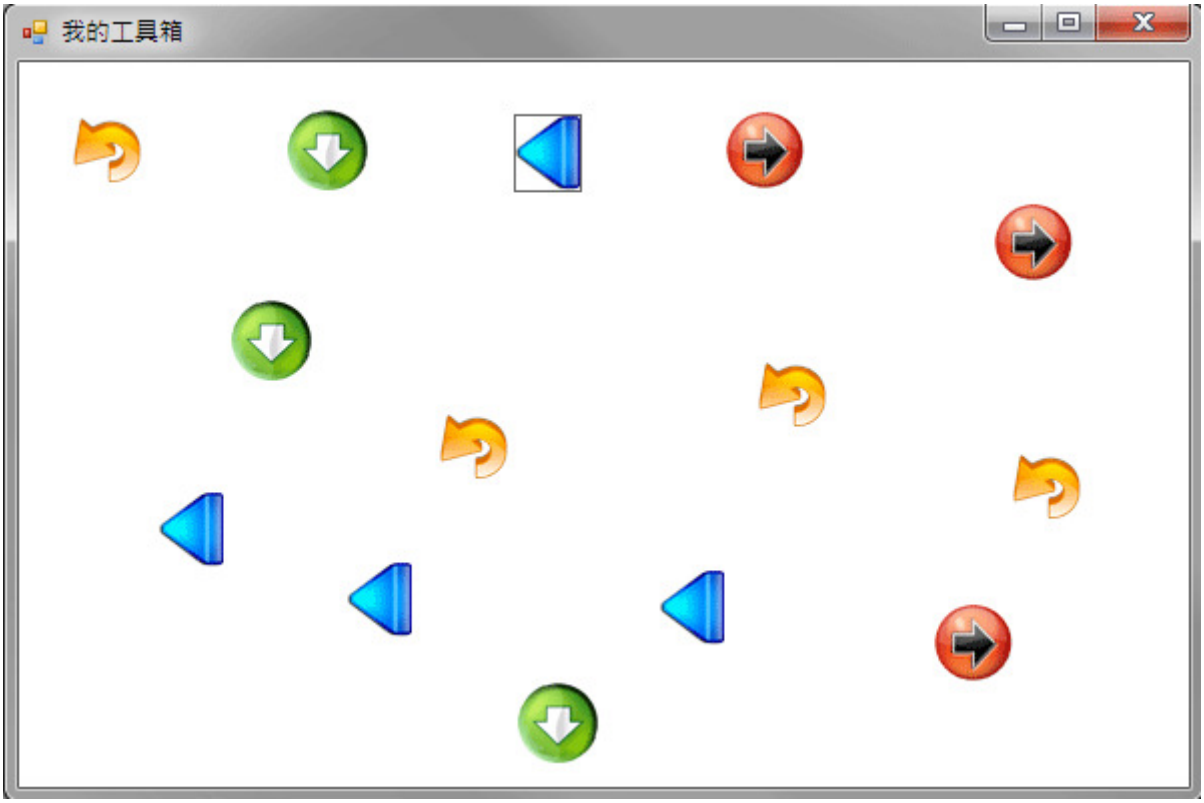
```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (P != null)
    {
        PictureBox Q = new PictureBox();
        Q.SizeMode = PictureBoxSizeMode.AutoSize;
        Q.Image = new Bitmap(P.Image);
        Q.Left = e.X - Q.Width / 2;
        Q.Top = e.Y - Q.Height / 2;
        this.Controls.Add(Q);
    }
}
```

上述程式首先檢查 P 是否已經存在，P != null 表示使用者已經點選某個圖案，null 是空的意思，如果等於(==)null 即表示沒有此物件。如果 P 存在，則宣告建置一個新的 PictureBox 物件 Q，設定影像模式(PictureBoxSizeMode)為自動縮放到原始大小(AutoSize)，其影像(Image)則決定於之前點選的物件 P 的影像(P.Image)。

[共用影像或使用新影像的差異]

在此必須注意，如果你寫成 Q.Image = P.Image;看起來沒錯，執行起來也沒錯！但是稍後你會發現所有複製品是共用同一張影像的！如果你旋轉某個物件時，所有複製品都會一起轉動！這應該不是你預期的效果吧？當然如果你真的需要這樣的結果，也可以這樣做！

接著由滑鼠點選的位置(e.X 與 e.Y)決定新 PictureBox Q 的位置(Left 與 Top 屬性)，上述算法會將物件的中心點設為滑鼠所點的位置。最後此物件必須被加入(Add)到表單(this)的控制項集合(Controls)之中，這樣就可以成功的複製一個物件到指定位置了！執行程式多次點擊就可以複製很多物件，當然還可以隨時切換圖案種類，如下為測試操作畫面之一：



14-4 產生拖曳功能

[使用樣板物件撰寫拖曳程式]

這個階段的目標是讓所有新產生的複製物件都可以被拖曳，但是在設計階段它們根本還不存在，怎麼寫它們的事件副程式呢？答案是先用一個已經存在的 PictureBox 模擬先寫好事件副程式的樣板(可參考五子棋單元)。在之前的小畫家等單元中我們已經學過拖曳程式的寫法，如下我們必須將程式寫在 pictureBox1 的 MouseDown 以及 MouseMove 事件之中：

```
int mdx, mdy;
private void pictureBox1_MouseDown(object  sender, MouseEventArgs e)
{
    mdx = e.X;
    mdy = e.Y;
    P = (PictureBox)sender;
}

private void pictureBox1_MouseMove(object  sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        PictureBox Q = (PictureBox)sender;
        Q.Left += e.X - mdx;
        Q.Top += e.Y - mdy;
    }
}
```

[在 MouseDown 事件同時完成物件點選]

上述程式與之前的拖曳程式稍有不同的是 P = (PictureBox)sender;這一句！意義是將事件觸發者 sender 轉換(宣告)為 PictureBox 物件 P，亦即使用者點壓此物件時就將此物件視為 P 物件，這樣就同時完成了點選物件的動作，之後我們就知道**正在處理**哪一個物件了！必須記得執行期間可能有很多新產生的物件會在視窗之中，所以 P 不只可能是原來的四個樣板之一，也可能是任何一個動態產生的新 PictureBox！隨時確定程式碼使用的對象(P)，在此範例中很重要。

先測試一下上面的程式是不是確實可以讓 pictureBox1 在執行階段被拖曳。但是記得這只是模擬用的，事實上我們並不希望當作工具箱樣板之一的 pictureBox1 真的能被拖曳，所以請打開 Form1.Designer.cs 檔案，找到 pictureBox1 物件相關程式中的這兩行**刪除**(或前面加雙斜線變成註解)：

```
this.pictureBox1.MouseMove += new System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseMove_1);
this.pictureBox1.MouseDown += new System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown_1);
```

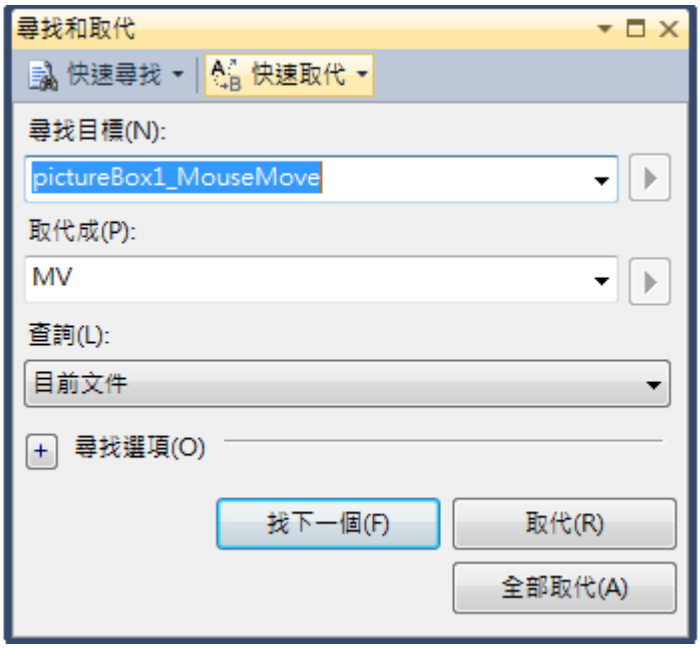
這表示 pictureBox1 的 MouseDown 與 MouseMove 將不會使用事件副程式，當然就不會動了！接下來要讓這兩個原本屬於 pictureBox1 的副程式變成新增物件的 MouseDown 與 MouseMove 副程式，這樣它們就可以被拖曳了！作法是在新增物件時與修改屬性一樣，告知電腦新物件要使用這兩個事件副程式！請修改新增物件的 Form1_MouseDown 副程式如下。

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (P != null)
    {
        PictureBox Q = new PictureBox();
        Q.SizeMode = PictureBoxSizeMode.AutoSize;
        Q.Image = new Bitmap(P.Image);
        Q.Left = e.X - Q.Width / 2;
        Q.Top = e.Y - Q.Height / 2;
        Q.MouseDown += new MouseEventHandler(this.pictureBox1_MouseDown);
        Q.MouseMove += new MouseEventHandler(this.pictureBox1_MouseMove);
        this.Controls.Add(Q);
    }
}
```

[簡化與置換副程式名稱]

很簡單吧？與共用事件副程式的寫法完全一樣，試試看！現在所有新增物件都可以被拖來拖去了！有沒有覺得像 pictureBox1_MouseDown 這樣的名字太長很麻煩？我就常常這樣覺得！其實可以使用程式碼頁面的編輯取代功能將所有的 pictureBox1_MouseDown 變成 MD，pictureBox1_MouseMove 變成 MV！反正在此以後 picturBox1_MouseDown 本來就不是給 pictureBox1 使用嘛！就改了它們吧？快捷操作方式是先選取想置換的字串，鍵入 Ctrl+H 鍵盤按鍵，出現如下視窗，接下來應該不必說明了。

```
(this.MD);
(this.pictureBox1_MouseMove);



s e)



nder, MouseEventArgs e)

seButtons.Left)
```



相關程式碼現在應該變成這樣：

```
    int mdx, mdy;
    private void MD(object sender, MouseEventArgs e)
    {
        mdx = e.X;
        mdy = e.Y;
        P = (PictureBox)sender;
    }

    private void MV(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            PictureBox Q = (PictureBox)sender;
            Q.Left += e.X - mdx;
            Q.Top += e.Y - mdy;
        }
}

private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (P != null)
    {
        PictureBox Q = new PictureBox();
        Q.SizeMode = PictureBoxSizeMode.AutoSize;
        Q.Image = new Bitmap(P.Image);
        Q.Left = e.X - Q.Width / 2;
        Q.Top = e.Y - Q.Height / 2;
        Q.MouseDown += new MouseEventHandler(this.MD);
        Q.MouseMove += new MouseEventHandler(this.MV);
        this.Controls.Add(Q);
    }
}
```

14-5 縮放與刪除功能

[製作跳出式選單]

接下來我們想對產生的物件作更多的處理，包括縮放大小以及刪除它！最簡單的方式是建立一個跳出式選單，將功能程式寫在裡面，再將它與新物件連結即可。首先請到工具箱的「功能表與工具列」分類下找出一個 ContextMenuStrip 物件，編寫如下的三個項目：

相對應的功能副程式程式碼後：

```
private void 刪除ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    P.Dispose();
}

private void 放大ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    P.SizeMode = PictureBoxSizeMode.StretchImage;
    P.Width *= 2;
    P.Height *= 2;
}

private void 縮小ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    P.SizeMode = PictureBoxSizeMode.StretchImage;
    P.Width /= 2;
    P.Height /= 2;
}
```

請注意這段程式碼是用於新增物件被按右鍵時會跳出選單，然後產生作用的程式，如前所述，對物件按右鍵時就是一個 MouseDown 動作，就會讓該物件變成表單公告的 P 變數！所以幾個副程式都是假設以 P 為處理對象，事實上就是可以用於任何被點選的物件。影像縮放程式請參考簡易秀圖程式的單元，刪除物件就是 Dispose！

[連結物件與跳出式選單]

但是上述選單(ContextMenuStrip1)如何連結到新增的物件呢？如同增加新增物件的一個屬性而已，請在前面新增 Q 物件的程式碼中加入一個屬性指定 ContextMenuStrip 即可(粗體字部分)：

```
PictureBox Q = new PictureBox();
Q.SizeMode = PictureBoxSizeMode.AutoSize;
Q.Image = P.Image;
Q.Left = e.X - Q.Width / 2;
Q.Top = e.Y - Q.Height / 2;
Q.MouseDown += new MouseEventHandler(this.MD);
Q.MouseMove += new MouseEventHandler(this.MV);
Q.ContextMenuStrip = this.contextMenuStrip1;
this.Controls.Add(Q);
```

寫好以上程式你應該可以按右鍵選擇縮放或刪除程式執行中新增的物件了！如下圖：

14-6 旋轉與翻轉功能

[更多影像處理功能]

接下來我們繼續在跳出式選單內加入左右翻轉、上下翻轉與 90 度旋轉三個選項，並分別寫程式如下：

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

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

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

三者的作法類似，就是直接對選取物件 P 的 Image 屬性操作，用 RotateFlip 方法將 Image 物件旋轉(Rotate)或翻轉(Flip)，最後重新整理 P 影像框物件即可。處理的效果如下圖，可以看出圖中被複製的圖案都經過翻轉或旋轉了！：

14-7 進階挑戰

一、如何製作物件相疊時移到上層或下層的程式介面？

提示：可以使用物件的 BringToFront 與 SendToBack 方法，加入跳出式選單。

二、如何製作用滑鼠雙擊目標以刪除物件的程式？

提示：建立新增物件的 DoubleClick 事件副程式。

三、如何製作讓物件可以不按比例任意變形的程式介面？

提示：建立可拖曳的小物件為物件的控制端點，如小畫家等軟體。

課後閱讀

動態物件的事件副程式

我們已經知道程式由很多的「物件」組成，前幾個單元也學會除了從工具箱取得樣板製造物件之外，也可以用程式碼宣告(建置)產生物件。當然也可以用修改它們的屬性，如打磚塊單元一樣讓磚塊排列整齊。但是別忘了，完整的物件內容包括「屬性」、「方法」與「**事件**」三種成員。其中屬性與方法是寫在物件的樣板(專業術語叫做 Class 類別)程式之內，「屬性」多半可以修改它們的值(也有些是唯讀的)，而「方法」如 pictureBox1.Refresh()需要時直接叫用就可以了！但是「事件」則需要先建立事件副程式，並給予程式碼才會有用。如何讓我們宣告產生的動態物件也可以擁有需要的事件副程式，就是使用程式碼控制動態物件的終極必殺技了！

[事件副程式的 sender 與 e 參數]

在此，我們必須預先寫好動態物件的事件副程式，最重要的問題是必須搞定事件副程式的參數 sender 與 e！任何事件副程式都會有這兩個參數，sender 指觸發事件的物件，這比較單純，一句 object sender 的宣告就解決了！但是 e 其實不是"一個"而是"一組"參數，不同的事件會有不同的參數集合，譬如鍵盤 KeyDown 事件就必須寫 KeyEventArgs e；滑鼠事件就必須是 MouseEventArgs e；其他事件通常要寫 EventArgs e 等等(Args 是 Arguments 參數的縮寫，Event 是事件的意思)。

這樣就行了嗎？未必！譬如與 KeyDown 非常類似的 KeyPress 事件就要寫 Key**Press**EventArgs e，天啊！誰會記得這些細微差異呢？所以最簡單安全的作法就是使用一個設計階段的物件去模擬產生同類的事件副程式。譬如你要替動態產生的 TextBox 配置 TextBox.TextChanged 事件，就先用一個設計階段的 TextBox 來寫程式，經過一般建立事件副程式程序產生的程式框架一定會正確的定義 e 的參數集合。寫好程式之後如果實際上不需要這個物件，你可以將該物件刪除，或者只將該物件指定使用該事件副程式的程式碼刪除，如本單元範例內容所示。

有了正確格式的副程式，接下來要定義某物件使用該事件副程式只要宣告新物件使用該副程式即可，但是又有一個小障礙出現！請看本單元的範例是：

Q.MouseDown += new **MouseEventHandler**(this.pictureBox1_MouseDown);

注意到了嗎？粗體字的部分又隱含了參數集合的種類定義，隨便寫也是不行的！所以最好不要直接刪除前面的模擬物件與事件副程式連結的程式碼，就是將它註解掉即可！稍後新物件要連結事件副程式時還是必須參考它的！

[為何需要學動態物件的處理？]

簡單說，當你學會了這一招，你已經可以自由自在地使用程式碼產生任何功能完整的物件，完全不必使用工具箱！必須知道，尤其是遊戲程式中，常常都會因應遊戲過程要產生不

同數量、種類，甚至具備不同事件副程式反應的物件，學會這些技巧，你就不會被迫在設計階段就預先設計好幾百個『可能會用到』的物件在表單上，大部分都隱藏(Visible=false)著備用，真的要用時又發現剛好數量不夠！而且，這樣作會讓你的程式記憶體負擔很重，跑得很慢，也令你在設計時好像在垃圾堆裡面工作一樣，東西多得層層疊疊都找不到，非常狼狽！

程式設計的資源回收

[用 Dispose 刪除物件]

本單元在刪除物件時還用到一個重要的指令 Dispose！多數同學都知道如果想讓物件「消失」可以使用「物件.Visible=false」的程式碼，在本單元中如果你堅持用這個方式讓物件消失對於使用者來說視覺上是沒差！

但是請注意 Visible=false 只是「隱藏」物件而不是刪除物件，必要時將 Visible=true 物件就會再度出現。如果你已經確定不需要再使用這個物件應該盡量用 Dispose 刪除物件，原因是不要留下太多垃圾來塞爆記憶體。以本單元為例，使用者有可能長時間開著這個程式，反覆的增加很多物件又「刪除」它們，如果曾經建立的物件一直都只是「隱藏」，很快的記憶體內就會有幾百，甚至幾千張完全不會再使用到的圖片，想要不當機都很難的！

[記憶體省著用]

我們接連幾個單元學到了與動態產生物件相關的程式，未曾提到的一個重要原因是這可以讓記憶體的使用精簡，既然我們在需要時才建置(採購？)物件，不用時也該妥善處理垃圾，不然程式用著用著莫名其妙就會因為記憶體不足而當掉，這就不是學習使用動態產生物件的初衷了！

[節省電腦資源不是過時觀念]

或許有人會說現在的電腦越來越厲害，哪會在乎這些小浪費？但是各位應該注意到近年電腦環境的改變是變得輕薄短小，智慧型手機與平板電腦盛行，其實記憶體的掌控技術比以往都更為重要！不然你的程式就會無法在這些小型電腦上使用。也因此，這些其實多半被定義為進階技術的動態物件程式技巧在本書中列為很重要的一部分！雖然日後實際寫作手機程式時可能不是使用 C#語言，但是內行人都知道，程式語言概念都是相通的！最常用來寫手機程式的 Java 語言與 C#相似度大約有 90%，所以一切學習都是值得的！

談點有關人生哲學的題外話，我已經清楚交代我的太太小孩說：如果我先走了，直接將我的遺體火化，然後合法的灑到海裡、河流或任何易於飄散分解回歸大自然循環的地方就好！千萬不要土葬或留個骨灰罈擺在靈骨塔之類的！要紀念我，有照片文字等等不占空間的電腦資訊(或者此書)就很夠了！原因和前述 Dispose 的概念很有關係，我如果已經不在人世了，還堅持要佔據地球上的一塊空間長達幾百幾千年，我想地球很快就會被墳墓與靈骨塔塞爆了！而且最終墓地還是會被拆除掩埋，又有甚麼尊嚴可言？死都死了，就直接讓出空間給後人用吧！包括我已經不用的身體，能捐就捐，不能捐的也快點讓大自然去資源回收吧！