Visual C# 2010 程式設計經典

第13章

鍵盤滑鼠與共用事件

13.1 鍵盤事件

- 目前個人電腦的主要輸入工具不外是滑鼠與鍵盤。滑鼠是能在螢幕上做選取和快速定位的工具，至於鍵盤主要是用來輸入資料的工具。
- 程式中靈活運用滑鼠與鍵盤所提供事件，可使得程式生動不少。
- 譬如：在應用程式中對TextBox控制項，判斷到底是按下哪個按鍵？或判斷一些較特殊的按鍵、組合按鍵？或按上、下、左、右鍵來移動物件，按空白鍵不放開時發射子彈，放開按鍵時結束發射子彈，以上這些都必須透過鍵盤事件來處理。

- Visual C# 2010提供下面三個常用鍵盤事件讓你能輕易地完成鍵盤的處理工作：

事件名稱	說明
KeyPress	在指定物件上收到由鍵盤按下的字鍵。僅能回應按鍵動作，無法判斷目前按鍵是否按住或放開。
KeyDown	在指定物件上偵測到鍵盤有鍵被按住。
KeyUp	在指定物件上偵測到鍵盤上被按住的鍵已放開。

- 要使某個物件(或稱控制項)產生KeyDown事件、KeyUp事件或KeyPress事件，先要讓該物件取得駐點(或稱控制權)，即變成作用物件，才能接受鍵盤事件。

13.1.1 KeyPress事件

- 當在鍵盤上有做按鍵的動作時就會觸動KeyPress事件，但所按的鍵必須是具有KeyAscii碼的按鍵，才會觸動(執行)KeyPress事件。
- 至於有效的KeyPress按鍵如下表所示：

有效的按鍵	KeyAscii 碼值
可顯示的鍵盤字元	字元的 ASCII 碼
Ctrl + A 至 Ctrl + Z	1 至 26
Enter 和 Ctrl + Enter	13 和 10
Backspace 和 Ctrl + Backspace	8(倒退鍵)和 127
空白鍵	32

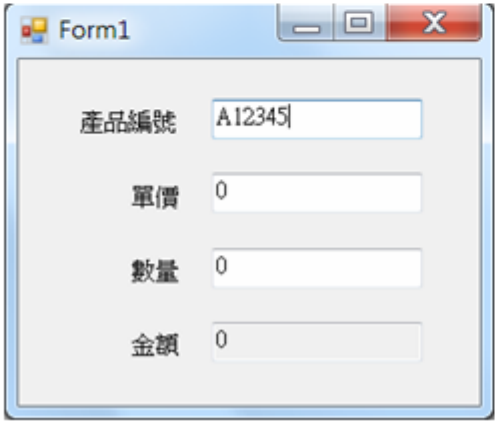
KeyPress語法：

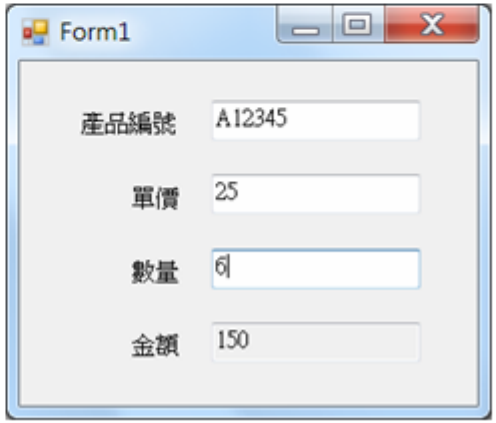
```
private void 物件_KeyPress(object sender, KeyPressEventArgs e)
{
        ……
}
```

範例 ：KeyPressDemo.sln

建立過濾字元的 TextBox 文字方塊控制項。如下圖示，產品編號的文字方塊擁有過濾字元的功能，第一個字必須輸入英文字，接著才可以在第 2~5 個字輸入數字，而其它字元則無法輸入到文字方塊內。單價和數量文字方塊限輸入數字和倒退鍵。

執行結果





操作步驟

Step 01 表單輸出入介面如下圖。

Step 02 撰寫程式碼：

```
09          // 表單載入時執行
10          private void Form1_Load(object sender, EventArgs e)
11          {
12              txtPrice.Text = "0";
13              txtQty.Text = "0";
14              txtTotal.Text = "0";
15              txtTotal.ReadOnly = true;
16          }
17          // txtId 產品編號文字方塊的 Text 屬性改變時執行
18          private void txtId_TextChanged(object sender, EventArgs e)
19          {
20              int Loc = txtId.SelectionStart;       // 儲存目前游標位置
21              // 當字母轉成大寫指定給 txtId.Text 時游標移到字串最後
22              txtId.Text = txtId.Text.ToUpper();
23              txtId.SelectionStart = Loc;           // 將游標還原到原來位置
24          }
```

```
25      // 在 txtId 產品編號文字方塊按鍵後再放開時執行
26      private void txtId_KeyPress(object sender, KeyPressEventArgs e)
27      {
28          if (txtId.Text.Length < 6)//檢查輸入的產品編號長度是否超過六個字元
29          {
30              if (txtId.SelectionStart == 0)//檢查輸入的第一個字元是否為字母
31              {   //將輸入的字母轉成大寫
32                  string S = e.KeyChar.ToString().ToUpper();
33                  if (S.CompareTo("A") < 0 || S.CompareTo("Z") > 0)
34                  {
35                   // 若輸入的第一個字元不是字母,取消輸入字元不顯示,游標停在原處
36                      e.Handled = true;
37                  }
38              }
39              else  // 若輸入的字元是第 2 個(含)以後的字元執行此段程式碼
40              {
41                  if (e.KeyChar.CompareTo('0') < 0 ||
                        e.KeyChar.CompareTo('9') > 0)
42                  {
43                      if (e.KeyChar!='\b')//若輸入的字元非數字且不是倒退鍵
44                      {
45                          e.Handled=true;//取消輸入字元不顯示,游標停在原處
46                      }
47                  }
48              }
49          }
```

Handle = **true,** messages are processed before reaching controls on the form.

```
50          else    //若輸的字元的長度超過 6 個字元執行此區段程式碼
51          {
52              if (e.KeyChar!='\b')//若是倒退鍵,取消輸入字元不顯示,游標停在原處
53              {
54                  e.Handled = true;
55              }
56          }
57      }
58      // txtPrice 單價文字方塊的 Text 屬性改變時執行
59      private void txtPrice_TextChanged(object sender, EventArgs e)
60      {
61          try
62          {
63              int price = int.Parse(txtPrice.Text);
64              int qty = int.Parse(txtQty.Text);
65              txtTotal.Text = (price * qty).ToString();
66          }
67          catch//若輸入的資料有誤執行此空區段程式碼,不處理所發生的錯誤
68          { }
69      }
```

```
71      private void txtPrice_KeyPress(object sender, KeyPressEventArgs e)
72      {
73          //若輸入的字元是非數字且不是倒退鍵
74          if ((e.KeyChar < '0' || e.KeyChar > '9') && (e.KeyChar != '\b'))
75          {
76              e.Handled = true;  //取消輸入字元不顯示,游標停在原處
77          }
78      }
79      // txtQty 數量文字方塊的 Text 屬性改變時執行
80      private void txtQty_TextChanged(object sender, EventArgs e)
81      {
82          try
83          {
84              int price = int.Parse(txtPrice.Text);
85              int qty = int.Parse(txtQty.Text);
86              txtTotal.Text = (price * qty).ToString();
87          }
88          catch
89          { }
90      }
91      // 在 txtQty 數量文字方塊按鍵後再放開時執行
92      private void txtQty_KeyPress(object sender, KeyPressEventArgs e)
93      {
94          if ((e.KeyChar < '0' || e.KeyChar > '9') && (e.KeyChar != '\b'))
95          {
96              e.Handled = true;
97          }
98      }
```

13.1.2 KeyDown和KeyUp事件

- 在電腦射擊遊戲中，當按下某個按鍵不放時用來發射子彈；當放開按鍵時停止發射子彈，這樣完成一個按鍵動作就做了兩件事情，我們必須將發射子彈的程式寫在KeyDown事件內。
- 而將停止發射子彈的程式寫在KeyUp事件中。
- KeyDown與KeyUp事件的功能說明如下：

語法

```
private void 物件_KeyDown(object sender, KeyEventArgs e)
```

當按下鍵盤某個按鍵不放時會觸動 KeyDown 事件。

語法

```
private void 物件_KeyUp(object sender, KeyEventArgs e)
```

當放開目前被按下的按鍵時會觸動 KeyUp 事件。

- 在KeyDown與KeyUp事件處理函式中，若按下或放開的按鍵能被偵測到，表示該鍵擁有鍵盤掃描(KeyCode)。
- 因此可經由事件處理函式內引數中的第二個引數e來取得所按的鍵之相關訊息。
- 其中可以使用e.KeyCode來取得按鍵的掃描碼。若要得知該鍵意義，可經由e.KeyCode敘述輸出。
- 譬如：e.KeyCode 用來取得字元或是透過 e.KeyValue 取得該字元的ASCII碼。也可以透過Keys列舉常數來表示該字元。下表為Keys列舉常數說明：

常用按鍵字元	KeyValue 鍵值	Keys 列舉常數
A ~ Z	65 ~ 90	Keys.A ~ Keys.Z
0 ~ 9	48 ~ 57	Keys.0 ~ Keys.9
F1 ~ F12	112 ~ 123	Keys.F1 ~ Keys.F12
↑ 、 ↓	38、40	Keys.Up、Keys.Down
← 、 →	37、39	Keys.Left 、Keys.Right
Enter ↵	13	Keys.Enter
空白鍵	32	Keys.Space
⇧ Shift	16	Keys.ShiftKey
Ctrl	17	Keys.ControlKey
Alt	18	Keys.AltKey

⇧ Shift	16	Keys.ShiftKey
Ctrl	17	Keys.ControlKey
Alt	18	Keys.AltKey
Esc	27	Keys.Escape
Backspace	8	Keys.Back
Home	36	Keys.Home
End	35	Keys.End
PgUp、PgDn	33、34	Keys.PageUp、Keys.PageDown
Ins	45	Keys.Insert
Del	46	Keys.Delete

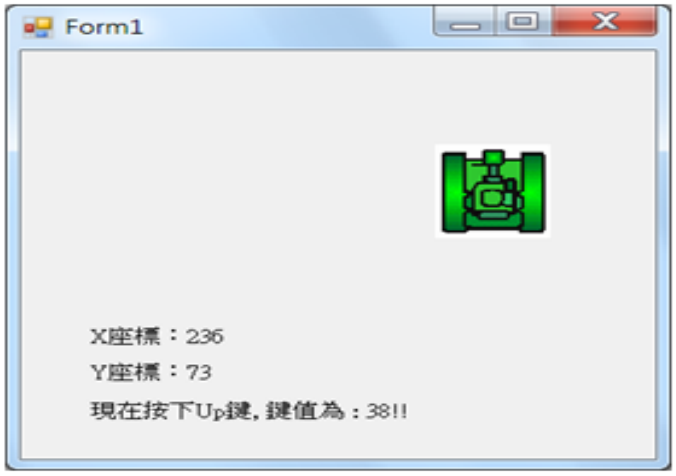
1. KeyUp和KeyDown事件處理函式能處理KeyPress事件所無法處理的按鍵，如功能鍵、編輯鍵和組合鍵。
2. KeyPress事件可以傳回一個字元的ASCII鍵碼，但無法得知目前鍵盤是持續按著，還是按一下就放開。
3. 如鍵入一個字元，會觸動上面三個事件，其發生順序為：KeyDown事件，接著為KeyPress事件，最後是KeyUp事件。

範例 ：KeyDownKeyUpDemo.sln

利用上、下、左、右鍵來控制坦克圖片的移動。透過按住鍵盤的 ↑、↓、←、→ 方向鍵，來控制坦克圖片上、下、左、右移動的方向，當坦克圖片移動超過表單上、下、左、右邊界時，即會從相反的邊界進入。此外在表單的左下角，會以英文字 Up、Down、Left、Right 來提示目前所按的方向鍵，並在其後顯示該方向鍵的掃描碼。

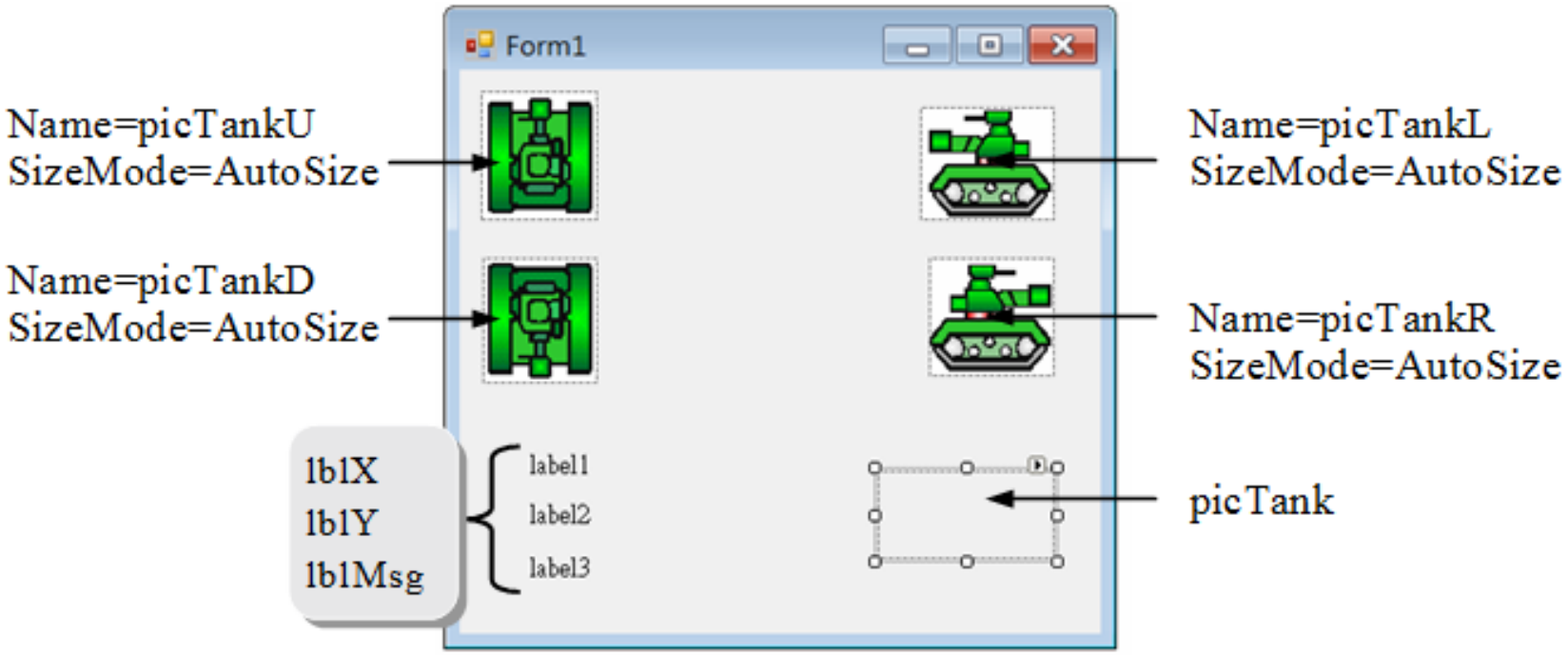
執行結果

1. 在上圖按 ↑ 鍵，圖片向上移動，並在左下角顯示 Up 鍵和其掃描碼 38。
2. 當坦克圖片移動超過表單上、下、左、右邊界時，即會從相反的邊界出現移動。例如：坦克圖片移動到表單右邊界外時，馬上會從左邊界出現往右移動。
3. 若所按的鍵不是 ↑、↓、←、→ 鍵時，只要該按鍵具有掃描碼，會在左下角顯示該按鍵的鍵值。



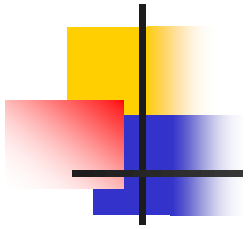
操作步驟

Step 01 表單輸出入介面如下圖。



註 將書附光碟 ch13 資料夾下的 tankU.bmp, tankD.bmp, tankL.bmp, tankR.bmp 四個坦克圖檔指定給 picTankU, picTankD, picTankL, picTankR 圖片方塊控制項的 Image 屬性。

```
10        // 宣告 locX, locY 用來存放坦克車開始的座標
11        int locX, locY;
12
13        // 表單載入時執行
14        private void Form1_Load(object sender, EventArgs e)
15        {
16            // 取得 picTank 的 X, Y 座標並指定給 locX, locY
17            locX = picTank.Location.X;
18            locY = picTank.Location.Y;
19            lblX.Text = "X 座標：" + picTank.Location.X;
20            lblY.Text = "Y 座標：" + picTank.Location.Y;
21            lblMsg.Text = "請按上下左右鍵控制坦克！";
22            picTank.SizeMode = PictureBoxSizeMode.AutoSize;
23            picTank.Image = picTankU.Image;
24            picTankU.Visible = false;  // 坦克往上圖隱藏
25            picTankD.Visible = false;  // 坦克往下圖隱藏
26            picTankL.Visible = false;  // 坦克往左圖隱藏
27            picTankR.Visible = false;  // 坦克往右圖隱藏
28        }
29        // 在表單按下鍵盤不放時執行
30        private void Form1_KeyDown(object sender, KeyEventArgs e)
31        {
32            switch (e.KeyCode)
33            {
34                case Keys.Up:        // 判斷是否按鍵盤上鍵
35                    picTank.Image = picTankU.Image;
```



```
36                  if ((picTank.Top + picTank.Height) <= 0)
37                  {
38                      picTank.Top = this.Height;
39                  }
40                  else
41                  {
42                      picTank.Top -= 10;
43                  }
44                  break;
45              case Keys.Down:       // 判斷是否按鍵盤下鍵
46                  picTank.Image = picTankD.Image;
47                  if (picTank.Top >= this.Height)
48                  {
49                      picTank.Top = 0 - picTank.Height;
50                  }
51                  else
52                  {
53                      picTank.Top += 10;
54                  }
55                  break;
56              case Keys.Left:     // 判斷是否按鍵盤左鍵
57                  picTank.Image = picTankL.Image;
58                  if (picTank.Width + picTank.Left <= 0)
59                  {
60                      picTank.Left = this.Width;
61                  }
62                  else
63                  {
64                      picTank.Left -= 10;
65                  }
```

```
66                      break;
67                  case Keys.Right:    // 判斷是否按鍵盤右鍵
68                      picTank.Image = picTankR.Image;
69                      if (picTank.Left >= this.Width)
70                      {
71                          picTank.Left = 0 - picTank.Width;
72                      }
73                      else
74                      {
75                          picTank.Left += 10;
76                      }
77                      break;
78              }
79              lblX.Text = "X 座標：" + picTank.Location.X;
80              lblY.Text = "Y 座標：" + picTank.Location.Y;
81              lblMsg.Text = "現在按下" + e.KeyCode.ToString() +
                    "鍵, 鍵值為 : " + e.KeyValue.ToString() + "!!";
82          }
83          // 在表單放開鍵盤的按鍵時執行
84          private void Form1_KeyUp(object sender, KeyEventArgs e)
85          {
86              lblX.Text = "X 座標：" + picTank.Location.X;
87              lblY.Text = "Y 座標：" + picTank.Location.Y;
88              lblMsg.Text = "請按上下左右鍵控制坦克！";
89          }
```

13.2 滑鼠事件
13.2.1 滑鼠事件簡介

- 在Windows Form應用程式的環境下，使用滑鼠操作讓使用者能很輕易地用來點取各種選項和按鈕、移動物件的圖示和插入點、編輯文件或是執行各種應用程式。
- 這些動作都可以交由下列八個滑鼠事件來處理：

事件名稱	說明
Click	在物件上按滑鼠左鍵一下。
DoubleClick	在物件上快按滑鼠左鍵兩下。
MouseDown	在物件上偵測到有滑鼠鍵被按住。
MouseEnter	滑鼠指標進入物件的範圍內會觸發此事件。
MouseHover	滑鼠指標停駐在物件上會觸發此事件。
MouseLeave	滑鼠指標離開物件時會觸發此事件。
MouseMove	在物件上偵測到滑鼠正在移動。
MouseUp	在物件上偵測到已按住之滑鼠鍵被放開。

13.2.2 Click與DoubleClick事件

- 當你按下滑鼠左鍵再放開滑鼠的動作就會觸動Click（按一下）事件。
- 例如：我們經常按一下「命令按鈕」來啟動相關的功能；亦可在表單中或任何型態的控制項上按一下來表示選取該「表單」或「控制項」。
- Click事件的使用時機有下列三種：

1. 用來選取物件
   例如移動滑鼠指標到某個圖示上按一下，使圖示名稱反白，或從清單方塊中的某選項上按一下使該選項反白。
2. 使物件獲得控制權，以利由鍵盤鍵入資料
   例如移動滑鼠指標到文字方塊內，按一下滑鼠左鍵，使文字方塊內出現閃爍的插入點游標。
3. 執行指令
   例如按命令按鈕、圖示鈕或功能表內功能選項。

- 當你快速按下滑鼠左鍵再放開滑鼠的動作連續兩次就稱為DoubleClick(按兩下)事件。DoubleClick事件的使用時機如下：

1. 開啟資料夾視窗
   例如移動滑鼠到「我的電腦」圖示上快按滑鼠左鍵二下，開啟「我的電腦」視窗。
2. 執行應用程式
   例如在資料夾視窗中有程式檔圖示與文件檔圖示。若用滑鼠指標在該程式檔圖示上快按二下，即會執行該程式檔而開啟該程式視窗。若用滑鼠指標在該文件檔圖示上快按二下，則會開啟能處理該資料檔的程式視窗，並將該文件檔的資料顯示在程式視窗的編輯區內。
3. 快速選取清單方塊選項
   例如在「開啟舊檔」對話方塊中，在某檔案名稱上快按滑鼠左鍵二下，則可直接開啟該檔案，而不必按[開啟]鈕。

13.2.3 MouseDown、MouseUp與Click事件

- 若在滑鼠任一鍵上面按一下，馬上會觸動MouseDown事件。若滑鼠鍵由按下的狀態再放開，馬上觸動MouseUp事件。無論滑鼠是否有被按下，只要移動滑鼠，馬上觸動MouseMove事件。
- 當按下滑鼠左鍵再放開，會同時觸動MouseDown、MouseUp和Click三個事件，其先後次序為MouseDown事件，接著為MouseUp事件，最後為Click事件。
- 在滑鼠事件中透過e.Button 來判斷按下滑鼠哪個按鈕。如下：

1. 若偵測到按滑鼠左鍵，則 e.Button==MouseButtons.Left
2. 若偵測到按滑鼠中鍵，則 e.Button==MouseButtons.Middle
3. 若偵測到按滑鼠右鍵，則 e.Button==MouseButtons.Right

範例：MouseEventDemo.sln

使用功能表與圖片方塊控制項製作簡易繪圖的小畫家程式。本例以圖片方塊當畫布，透過滑鼠事件當按滑鼠左鍵拖曳滑鼠即可在畫布上繪圖寫字。放開滑鼠即提筆不畫。小畫家程式提供的功能表選項說明如下：

執行結果

1. 執行功能表的 [檔案/開檔] 可將目前專案 bin/Debug 資料夾下的 myPic.jpg 圖檔放入圖片方塊中顯示。
2. 執行功能表的 [檔案/存檔] 可將在圖片方塊(本例使用圖片方塊當做繪圖的畫布)所進行繪圖或寫字以檔名 myPic.jpg 儲存到目前專案 bin/Debug 資料夾下。
3. 執行功能表的 [檔案/清除] 可將圖片方塊畫布清成白色，以便讓使用者重新繪圖。
4. 執行功能表的 [檔案/結束] 可結束程式。
5. 功能表的 [畫筆粗細] 有 1 Pixels、2 Pixels、3 Pixels 功能選項可用來設定畫筆的粗細。
6. 功能表的 [畫筆顏色] 有 黑、紅、綠、藍功能選項可用來設定畫筆的顏色。


Form1
檔案
畫筆粗細
畫筆顏色
開檔
存檔
清除
結束

程式分析

為配合本例介紹如何使用滑鼠在 PictureBox 圖片方塊控制項上面繪圖，先介紹基本繪圖要領：

1. 先建立圖形物件 bmp，接著使用 bmp 來當做畫布，畫布名稱為 g，圖形大小為 320 x 210。

```
Bitmap bmp = new Bitmap(320, 210);              // 建立 bmp 圖形物件
Graphics g = Graphics.FromImage(bmp) ;          // 使用 bmp 來當畫布 g
```

2. 將畫布 g 清成白色畫布。

```
g.Clear(Color.White);
```

3. 將畫布(即圖形物件 bmp)貼到 pictureBox1 圖片控制項上面。

```
pictureBox1.Image = bmp;
```

4. 在畫布上兩點座標(50,50)、(100,100) 繪製一條畫筆寬度為 1 Pixels 的紅色線段：

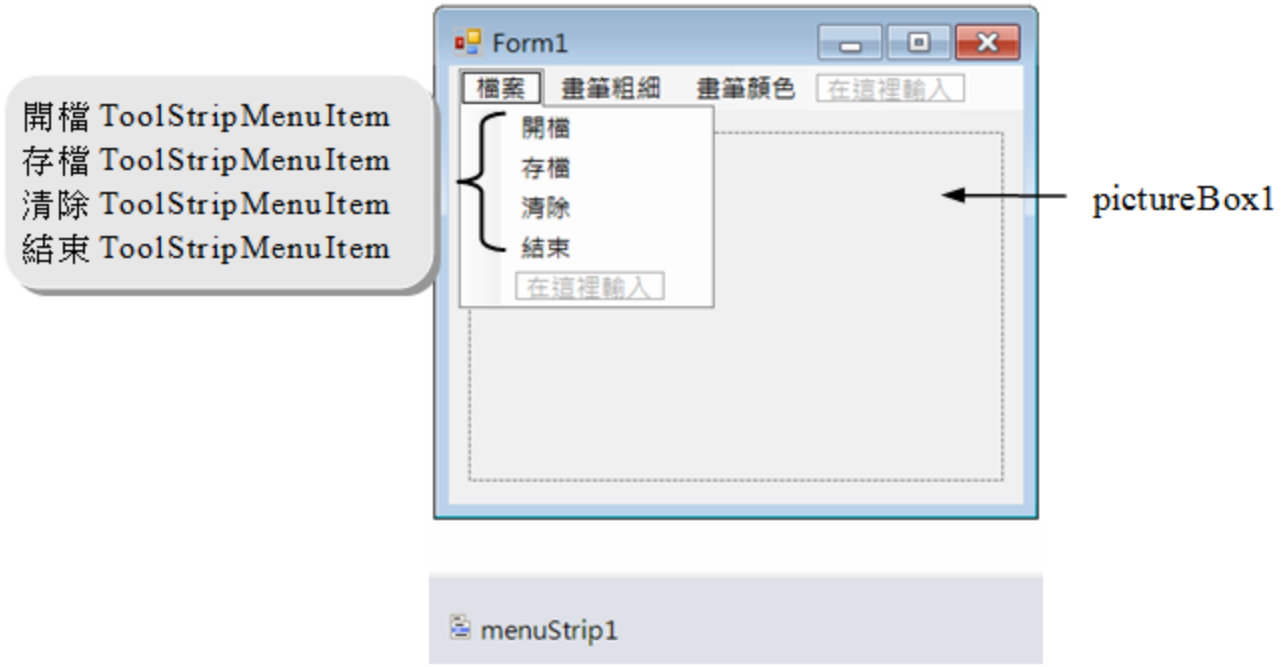
```
int point = 1;                              // 畫筆寬度
Pen p1 = new Pen(Color.Red, point);         // 建立寬度為 1 pixel 的紅色畫筆
g.DrawLine(p1, 50, 50, 100, 100);           // 在畫布上畫一條直線
pictureBox1.Image = bmp ;
```

5. 清畫布變成黃色畫布：

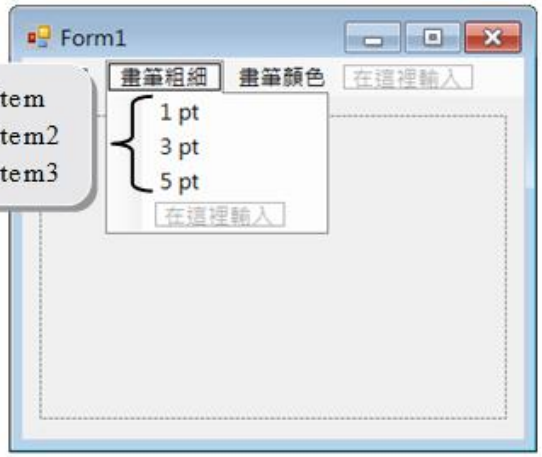
```
g.Clear(Color.Yellow);
pictureBox1.Refresh();
```

操作步驟

Step 01 表單輸出入介面如下圖。

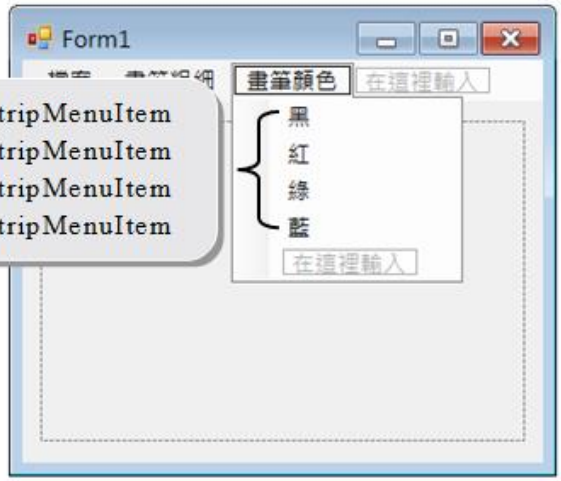


pixelsToolStripMenuItem
pixelsToolStripMenuItem2
pixelsToolStripMenuItem3



menuStrip1

黑 ToolStripMenuItem
紅 ToolStripMenuItem
綠 ToolStripMenuItem
藍 ToolStripMenuItem



menuStrip1

```
13          Bitmap bmp;         // 宣告圖形物件
14          int oldX, oldY;     // 記錄滑鼠指標 X、Y 座標
15          int PenPoint;       // 宣告 PenPoint 表示畫筆粗細
16          Color PenColor;     // 宣告 PenColor 表示畫筆顏色
17
18          // 表單載入時執行
19          private void Form1_Load(object sender, EventArgs e)
20          {
21              bmp = new Bitmap(320, 210);  //建立圖形物件大小 320*210
22              Graphics g = Graphics.FromImage(bmp);  // 建立畫布物件 g
23              PenColor = Color.Black;
24              PenPoint = 3;
25              g.Clear(Color.White);    // 將畫布清為白色
26              pictureBox1.Image = bmp;// 畫布貼到 pictureBox1 圖片方塊控制項上
27              pictureBox1.Refresh();  // 更新 pictureBox1 圖片方塊控制項
28          }
```

```
29        // 執行 [檔案/開檔] 指令執行
30        private void 開檔ToolStripMenuItem_Click
              (object sender, EventArgs e)
31        {
32            try
33            {
34                FileStream f = new FileStream("myPic.jpg", FileMode.Open);
35                bmp = new Bitmap(f);
36                f.Close();
37                pictureBox1.Image = bmp;
38            }
39            catch (Exception ex)
40            {
41                MessageBox.Show("目前專案無圖檔，請先繪圖後再存檔");
42            }
43        }
44        // 執行 [檔案/存檔] 指令執行
45        private void 存檔ToolStripMenuItem_Click
              (object sender, EventArgs e)
46        {
47            bmp.Save("myPic.jpg");
48        }
```

```
49        // 執行 [檔案/清除] 指令執行
50        private void 清除ToolStripMenuItem_Click
                (object sender, EventArgs e)
51        {
52            Graphics g = Graphics.FromImage(bmp);   // 建立畫布物件 g
53            g.Clear(Color.White);                   // 將畫布清為白色
54            pictureBox1.Image = bmp;
55        }
56        // 執行 [檔案/結束] 指令執行
57        private void 結束ToolStripMenuItem_Click
              (object sender, EventArgs e)
58        {
59            Application.Exit();
60        }
61        // 在 pictureBox1 圖片方塊上按滑鼠鍵
62        private void pictureBox1_MouseDown
              (object sender, MouseEventArgs e)
63        {
64            oldX = e.X;
65            oldY = e.Y;
66        }
```

```
67          // 在 pictureBox1 圖片方塊移動滑鼠會執行
68          private void pictureBox1_MouseMove
                (object sender, MouseEventArgs e)
69          {
70              //  判斷是否按下滑鼠左鍵
71              if (e.Button == MouseButtons.Left)
72              {
73                  Graphics g = Graphics.FromImage(bmp);
74                  Pen p = new Pen(PenColor, PenPoint);
75                  g.DrawLine(p, oldX, oldY, e.X, e.Y);//在 bmp 畫布上畫一條直線
76                  pictureBox1.Image=bmp;// 畫布貼到 pictureBox1 圖片方塊控制項上
77                  oldX = e.X;             // 將目前畫筆座標當作下次畫筆的起點
78                  oldY = e.Y;
79              }
80          }
81          // 設定畫筆粗細為 1 pt
82          private void pixelsToolStripMenuItem_Click
                (object sender, EventArgs e)
83          {
84              PenPoint = 1;
85          }
86          // 設定畫筆粗細為 3 pt
87          private void pixelsToolStripMenuItem2_Click
                (object sender, EventArgs e)
88          {
89              PenPoint = 3;
90          }
```

```
91          // 設定畫筆粗細為 5 pt
92          private void pixelsToolStripMenuItem3_Click
                (object sender, EventArgs e)
93          {
94              PenPoint = 5;
95          }
96          // 設定畫筆顏色為黑色
97          private void 黑ToolStripMenuItem_Click
                (object sender, EventArgs e)
98          {
99              PenColor = Color.Black;
100         }
101         // 設定畫筆顏色為紅色
102         private void 紅ToolStripMenuItem_Click(object sender, EventArgs e)
103         {
104             PenColor = Color.Red;
105         }
106         // 設定畫筆顏色為綠色
107         private void 綠ToolStripMenuItem_Click(object sender, EventArgs e)
108         {
109             PenColor = Color.Green;
110         }
111         // 設定畫筆顏色為藍色
112         private void 藍ToolStripMenuItem_Click(object sender, EventArgs e)
113         {
114             PenColor= Color.Blue;
115         }
```

13.3 控制項共用事件

- 假設表單上某些控制項的事件處理函式皆相同，若逐一對每個控制項撰寫相同的程式碼，不但增加程式碼的長度而且難以維護，這樣豈不是很沒有效率嗎？
- 此時可以建立一個方法(即共用事件處理函式)讓控制項的事件彼此共用來解決這個問題。
- 至於建立共用事件有下列兩種方式：

1. 在設計階段透過屬性視窗工具列的  事件圖示鈕來建立。
2. 直接在執行階段透過程式碼來增刪控制項的共用事件。

13.3.1 如何使用屬性視窗建立共用事件

- 譬如：欲將btn2~btn3 兩個按鈕的Click事件被觸發時都會以btn1_Click事件處理函式，其操作方式如下：

Step 01 首先在程式編碼視窗寫好 btn1_Click 事件處理函式內的程式碼，用來當作控制項的共用事件。

```
private void btn1_Click(object sender, EventArgs e)
{
        ……
}
```

Step 02 將 btn2 的 Click 事件指到 btn1_Click 事件處理函式

在表單中點選 btn2 按鈕成為作用控制項，接著移動滑鼠到屬性視窗中點選工具列中的  事件圖示鈕，接著切換到事件清單並點選 Click 事件，由下拉鈕清單中選取 btn1_Click 事件處理函式當作共用事件。

Step 03 將 btn3 的 Click 事件也指到 btn1_Click()事件

操作方式同上一步驟。

範例 ：EventHandlerDemo.sln

練習建立按鈕事件的共用事件處理函式。如下圖，根據你所按的按鈕，標籤控制項會顯示你所按下的按鈕名稱，本範例三個按鈕之一被按下時，皆觸發 按鈕一 鈕的 btn1_Click 事件處理函式。

執行結果

▲按下 按鈕一

▲按下 按鈕二

▲按下 按鈕三

操作步驟

Step 01 表單輸出入介面如下圖。

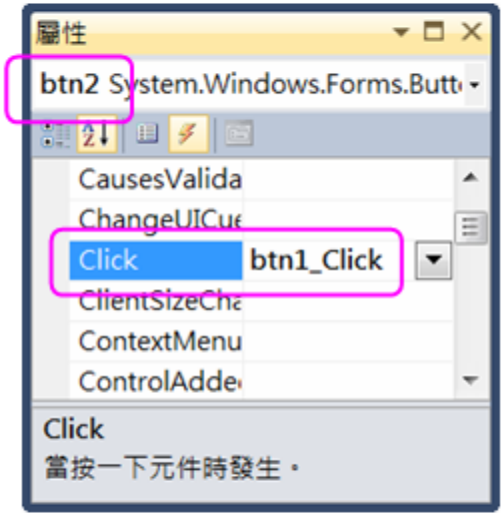
Step 02 撰寫程式碼：

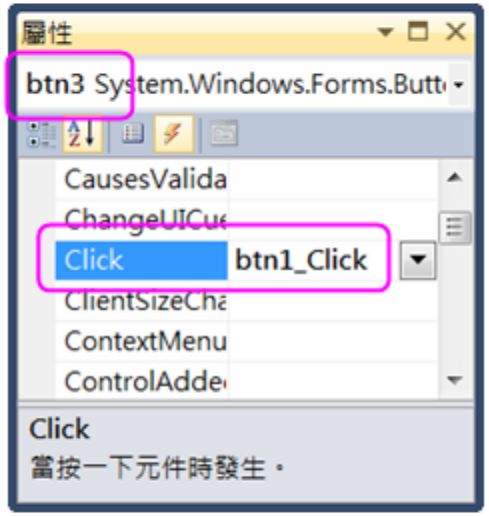
程式碼 FileName:Form1.cs

```
01 namespace EventHandlerDemo
02 {
03     public partial class Form1 : Form
04     {
05         public Form1()
06         {
07             InitializeComponent();
08         }
09         // 表單載入時執行
10         private void Form1_Load(object sender, EventArgs e)
11         {
12             lblShow.Text = "";
13         }
14
15         private void btn1_Click(object sender, EventArgs e)
16         {
17             Button btnHit; // 宣告 btnHit 為 Button 類別物件
18             // 將 sender 轉型成 Button 類別物件，接著再指定給 btnHit
19             // 此時 btnHit 即代表是使用者按下的按鈕
20             btnHit = (Button)sender;
21             lblShow.Text = " 目前按下 " + btnHit.Text;
22         }
23     }
24 }
```

Step 03 指定共用事件

本例希望按下 btn2 和 btn3 鈕觸發該鈕的 Click 事件時皆會執行 btn1_Click 事件處理函式。因此如下兩圖請將 btn2 及 btn3 的 Click 事件設為 btn1_Click。





Step 04 執行程式

① 若 [按鈕一] 被按下，會顯示"目前按下 按鈕一"。

② 若 [按鈕二] 被按下，會顯示"目前按下 按鈕二"。

③ 若 [按鈕三] 被按下，會顯示"目前按下 按鈕三"。

13.3.2 如何新增和刪除控制項的事件

Visual C# 2010 在程式執行階段中可以透過 EventHandler 型別來新增或取消當控制項事件被觸發時所要執行的事件處理函式，該事件必須符合一般定義事件處理函式的引數規則才有效。譬如自行定義一個 MyEvent 事件處理函式。其寫法如下：

```
private void MyEvent(object sender, EventArgs e)
{
        ……
}
```

在程式執行階段中欲將 btn1 的 Click 事件委託給上面定義的 MyEvent 事件處理函式執行。若使用「+=」來指定，表示新增事件處理；若使用「-=」來指定，表示取消事件處理，兩者寫法如下：

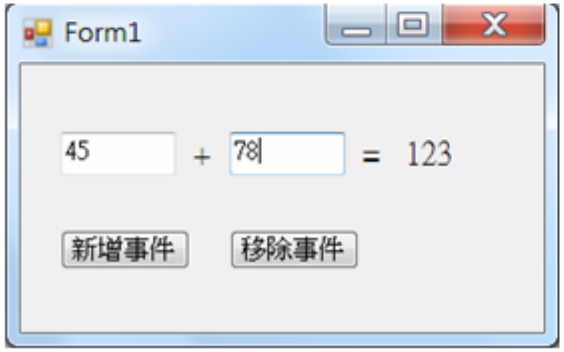
```
// 指定當 btn1 的 Click 事件被觸發時，會執行 MyEvent 事件處理函式
btn1.Click += new EventHandler(MyEvent);
```

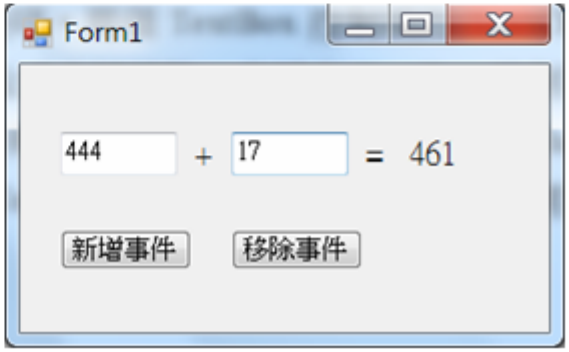
```
// 將處理 btn1 的 Click 事件的 MyEvent 事件處理函式移除
btn1.Click -= new EventHandler(MyEvent);
```

範例：AddRemoveEventDemo.sln

當表單中兩個 TextBox 控制項的 TextChanged 事件被觸發時，會執行 MyTextChanged 事件處理函式，MyTextChnaged 此事件處理函式可進行兩數相加並將其結果顯示在標籤控制項中。若按下 移除事件 鈕時，兩個 TextBox 控制項的 TextChanged 事件所指定的 MyTextChanged 事件處理函式會被移除，此時在 TextBox 控制項輸入資料將無法進行兩數相加。若按下 新增事件 鈕時，兩個 TextBox 控制項指定 MyTextChanged 為 TextChanged 事件的處理函式，此時在 TextBox 控制項輸入資料即可進行兩數相加。

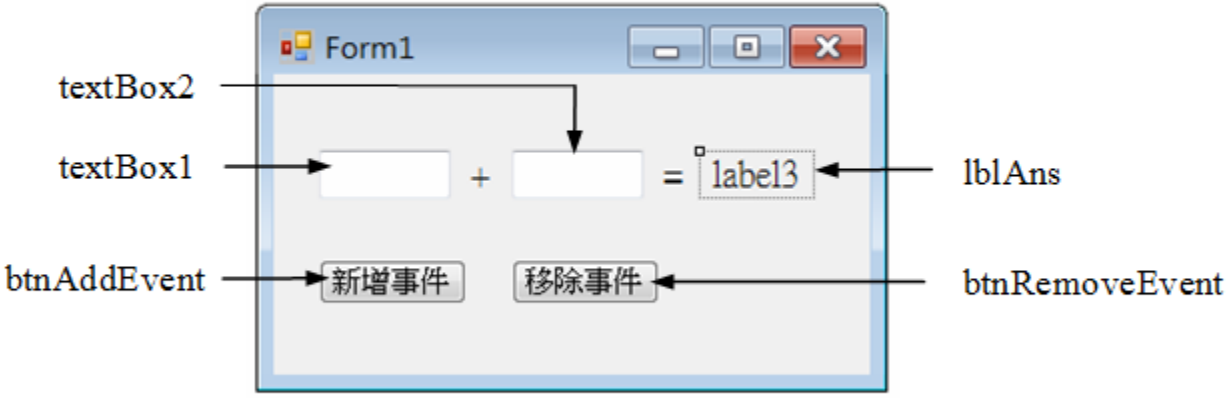
執行結果



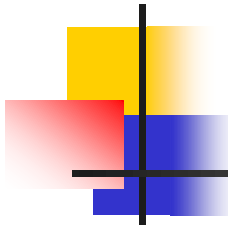


操作步驟

Step 01 表單輸出入介面如下圖。



Step 02 撰寫程式碼：



```
09        // 表單載入時執行
10        private void Form1_Load(object sender, EventArgs e)
11        {
12            lblAns.Text = "0";
13            textBox1.TextChanged +=new EventHandler(MyTextChanged);
14            textBox2.TextChanged +=new EventHandler(MyTextChanged);
15        }
16        // 自訂 MyTextChanged 事件處理函式
17        private void MyTextChanged(object sender, EventArgs e)
18        {
19            try
20            {
21                int n1, n2;
22                n1 = int.Parse(textBox1.Text);
23                n2 = int.Parse(textBox2.Text);
24                lblAns.Text = (n1 + n2).ToString();
25            }
26            catch (Exception ex)
27            {
28            }
29        }
```

```
30          // 按下 [新增事件] 鈕執行
31          private void btnAddEvent_Click(object sender, EventArgs e)
32          {
33              textBox1.TextChanged += new EventHandler(MyTextChanged);
34              textBox2.TextChanged += new EventHandler(MyTextChanged);
35          }
36          // 按下 [移除事件] 鈕執行
37          private void btnRemoveEvent_Click(object sender, EventArgs e)
38          {
39              textBox1.TextChanged -= new EventHandler(MyTextChanged);
40              textBox2.TextChanged -= new EventHandler(MyTextChanged);
41          }
```