中華民國 第 50 屆中小學科學展覽會

作品說明書

高職組　電子、電機及資訊科

第三名

091005

Windows 創意教學螢幕畫布

學校名稱：國立淡水高級商工職業學校

作者：	指導老師：
職三 洪稟凱	石明金

關鍵詞：螢幕畫布、電子白板、廣播教學

摘要

當我們在課堂上，教師使用電腦與投影機來上課時，PowerPoint 投影片的放映似乎成爲主流，其中內附的畫筆功能更可以在教學中事半功倍，但我發現如果脫離了簡報程式，教師在爲電腦上的教學資料講解、畫重點時非常不容易，因此我想利用目前所學，使用 Visual Basic 2005 Express 來撰寫一個『Windows 創意教學螢幕畫布』。

透過程式，講解者可用畫筆在螢幕上畫線或方框，也有文字、圖片方塊框可以選擇，讓講解者無論在什麼視窗界面上、甚至是在影片中，都可以輕輕鬆鬆完成所需要的動作，其背景依然持續動作，完全不受影響，而且解說完的畫面可以儲存，放在網路上供學生瀏覽複習。

目前坊間有電子白板可以達到類似的功能，但是因爲價格昂貴，不易普及，尤其大部份的學校的電腦教室皆採用教學廣播系統，如果利用『**Windows 創意教學螢幕畫布**』，就可以達到講解者的需求，而且不用添加任何的硬體設備。

壹、 研究動機

最近電子白板的出現，想到我們老師如果在上課時也能使用這麼方便的東西，就可以輕鬆的標重點了，而且過程中也不會有粉筆灰。第一次知道電子白板的我，覺得它很厲害，白板第一次聽說有電子的，於是興沖沖的去找它的資料，但是實際了解後發現，電子白板之所以目前不能普及，是因爲建置的價錢實在是不便宜。

因此我在網路上想找到有關的替代方案，想讓一般的投影機、或者是各個學校都會有的電腦教室廣播系統也能利用在電腦螢幕上繪圖來達到電子白板的效果，我找到一些可以在電腦桌面上隨意畫畫寫字的程式，但是這些程式都是利用擷取螢幕畫面的方式並把擷取下來的畫面存成圖片拿來當作畫布使用，並不像電子白板一樣，能直接在螢幕上繪圖。

於是，我想做一個更簡單更方便且可以很實用的『**教學螢幕畫布**』。在不用任何的成本下，利用學校所學的程式設計，可在任何的 Windows 畫面下繪圖或標示線條，以達成向電子白板一樣的效果。

貳、 研究目的

一、 直接在電腦螢幕上作圖，而不需透過擷取圖片的方式。

二、 將電子白板原理應用在電腦螢幕上

三、 讓老師能輕易的在電腦畫面上畫線、秀圖、加上文字，使同學透過畫面更了解老師的講解內容。

四、 利用儲存功能將電腦畫面儲存，以供學生複習參考。

參、 研究設備及器材

一、 研究設備：

電腦

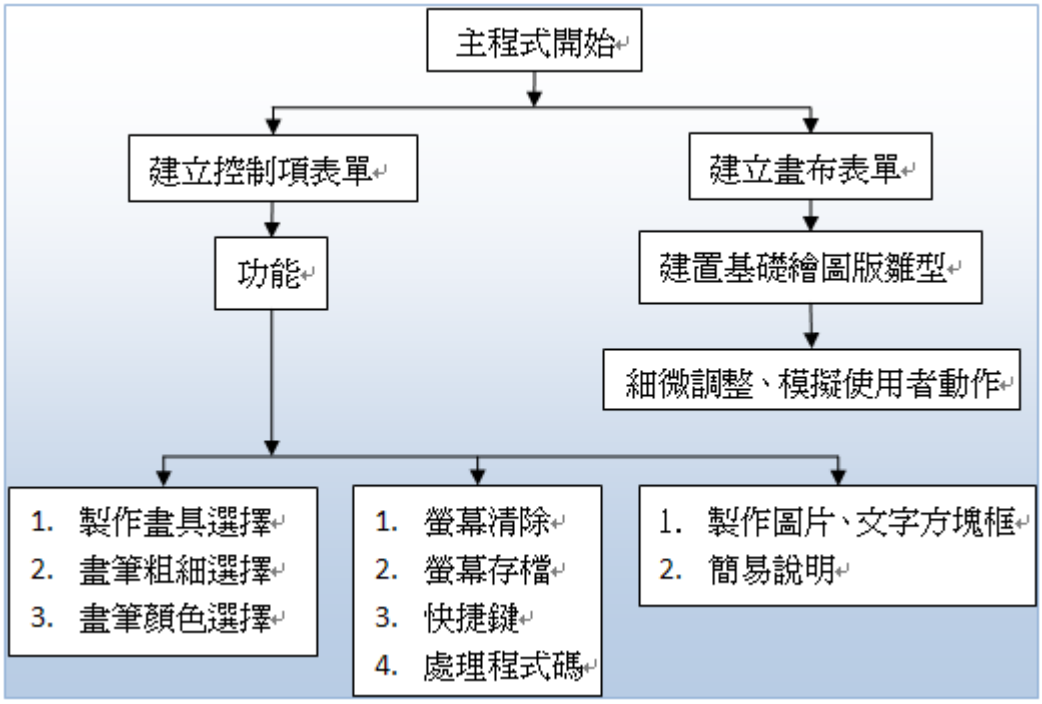
二、 研究材料：

Microsoft Visual Basic 2005 Express 版

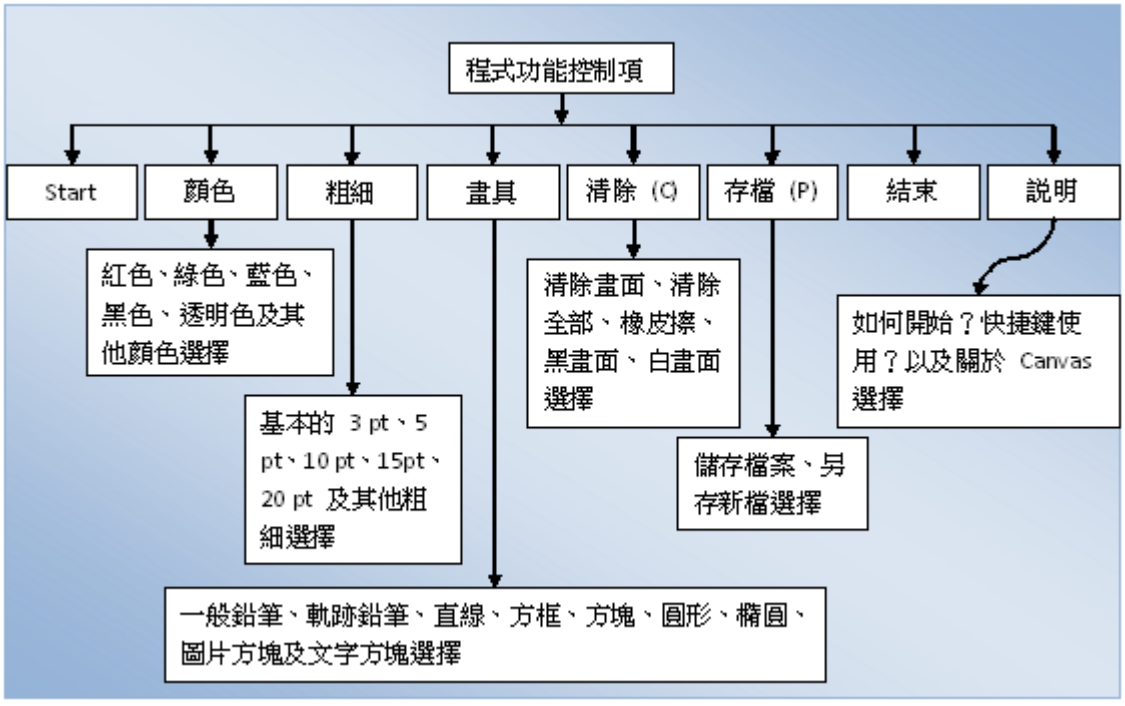
肆、 研究過程或方法

此次研究分成五部份，第一部份爲製作出螢幕繪圖板的基本雛型，第二部份爲畫具的設計與設定、顏色與粗細度選項的製做，第三部份分爲製作螢幕的清除及儲存功能、各個快捷鍵的設定與測試，控制項的重整及程式碼的排列整理，第四部分爲模擬使用者在使用上所可能的全部動作以及使用的流暢度及細調繪圖的靈敏度、畫筆的畫質，第四部份爲製作圖片及文字方塊框功能、簡易的說明選項。

下圖爲製作程序流程圖：



下圖爲程式功能選擇控制項設計流程圖：



一、 VB 繪圖基礎

(一) 座標格式：

要瞭解VB 的繪圖基礎與原理，必須先瞭解其座標格式。在VB 的座標格式中，表單上所採用的是類似數學上的直角座標系(x,y)，原點 (0,0)位於表單的左上角，x 座標向右增加，y 座標向下增加，x,y 皆爲整數，如圖 4-1 所示。

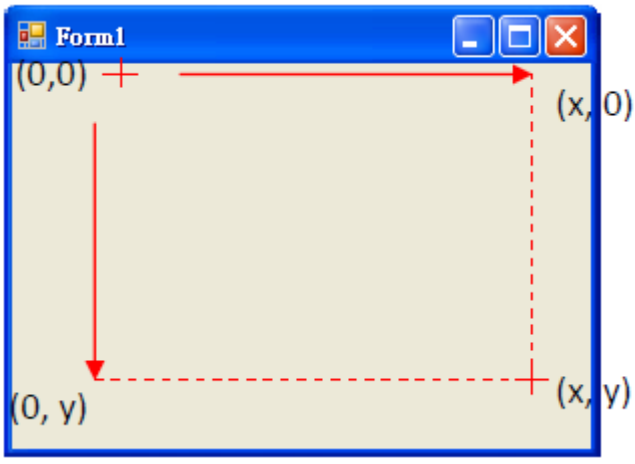


圖 4-1

在VB 的視窗程式裡，可以在特定的控制項上，以函式來繪圖，如在表單或圖片方塊上繪圖。但是在繪圖之前，您必須要透過一個繪圖物件 Graphics，才能使用 VB 所提供的繪圖方法。

用繪圖物件 Graphics 在表單上畫一條線的程式碼，如圖 4-2。

```
Private Sub Button1_Click(ByVal sender As System.Object, _
                          ByVal e As System.EventArgs) Handles Button1.Click
    Dim g As Graphics '產生一個名爲g 的Graphics 變數，但g 並未指派任何實體。
    g = Me.CreateGraphics '請Me(也就是指Form1)產生一個Graphics 物件，並指定到
    'g 的位置，所以現在的g 指的就是Form1 所派來的物件。
    g.DrawLine(Pens.Black, 10, 10, 20, 20) '請g 畫一條線，顏色爲黑色、起始點(10,10)、
    '終止點(20,20)。
    g.Dispose() 'g 使用完後，將他釋放。
End Sub
```

圖 4-2

(二) 框線繪圖方法：

使用Graphics 繪圖物件所提供的 DrawLine 函數，來繪製線段，如圖4-3。

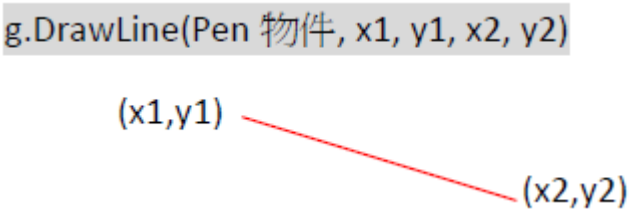


圖 4-3

說明：DrawLine 函數會根據傳入的端點座標來繪製線段，起點爲(x1,y1)，終點爲(x2,y2)。其中Pen 物件，可以從Pens 這個集合中取得，例如：Pens.Blue 代表藍色畫筆，Pens.Black 代表黑色畫筆，如圖4-4。

1. DrawRectangle 畫矩形框：

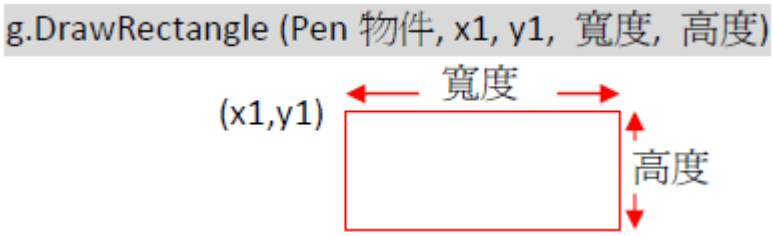


圖 4-4

2. DrawEllipse 畫橢圓形框，如圖4-5：

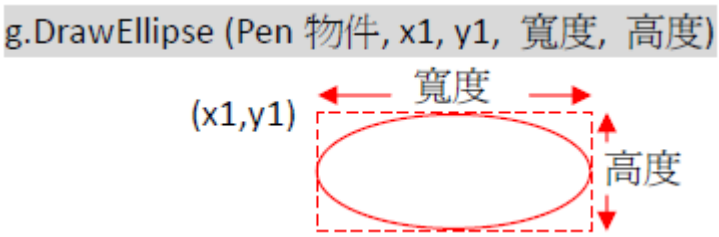


圖 4-5

3. 區塊繪圖方法 - FillRectangle 填滿矩形區，如圖 4-6：

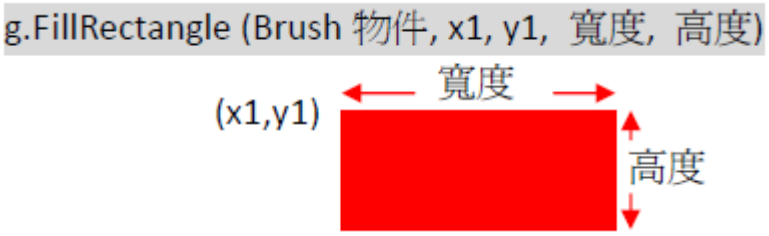


圖 4-6

假如想要繪製一個填滿的圖形，則需要使用相對應的Fill 方法，如：FillRectangle、FillEllipse FillPie，並將原先傳遞Pen物件修改爲傳遞Brush 物件即可。Brush 與Pen 物件，同樣皆爲可設定顏色之工具，唯Brush 是用來表示區塊的顏色，而Pen 用於表示框線的顏色。

二、 製做螢幕畫布的基本雛型

(一) 螢幕畫布表單基本架構

與其他普通的螢幕畫布程式不同的地方在於，本程式在畫布的部分是透明且可穿透的(如圖4-8)，這是利用指派 顏色 (Color) 給 TransparencyKey 屬性時，具有相同 背景 (BackColor) 的表單區域將透明地顯示。

圖4-7 是指定畫布表單背景與 TransparencyKey 的顏色相同。

```
Me.BackColor = Color.HotPink
Me.TransparencyKey = Color.HotPink
```

圖4-7

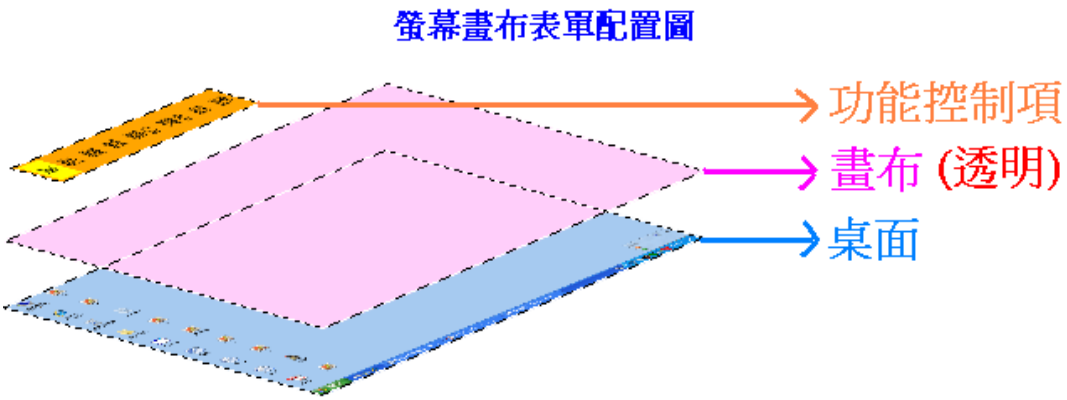
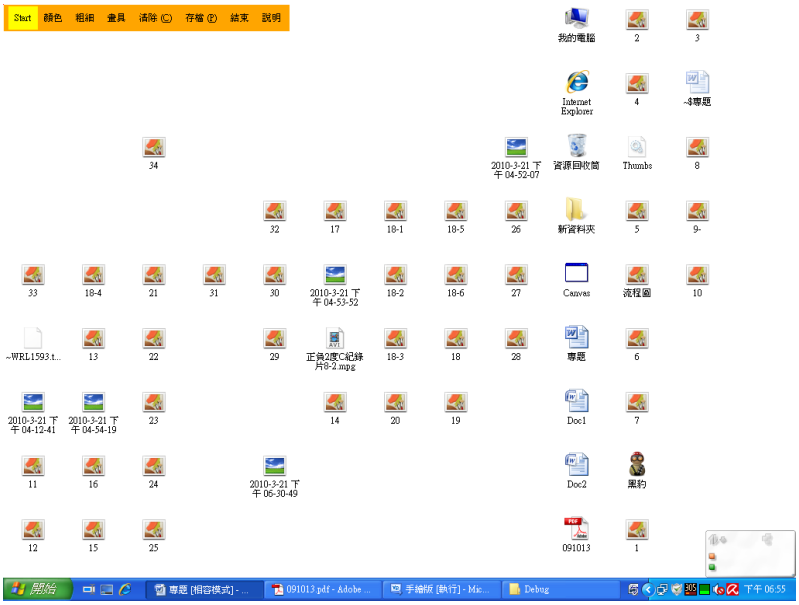


圖 4-8

下圖爲依配置所呈現的實際執行範例畫面



以下爲本螢幕畫布程式實際使用的示範 (圖 4-8-1 ~ 4-8-3)

圖 4-8-1 原本的簡報畫面

圖 4-8-2執行後在簡報上隨意繪圖

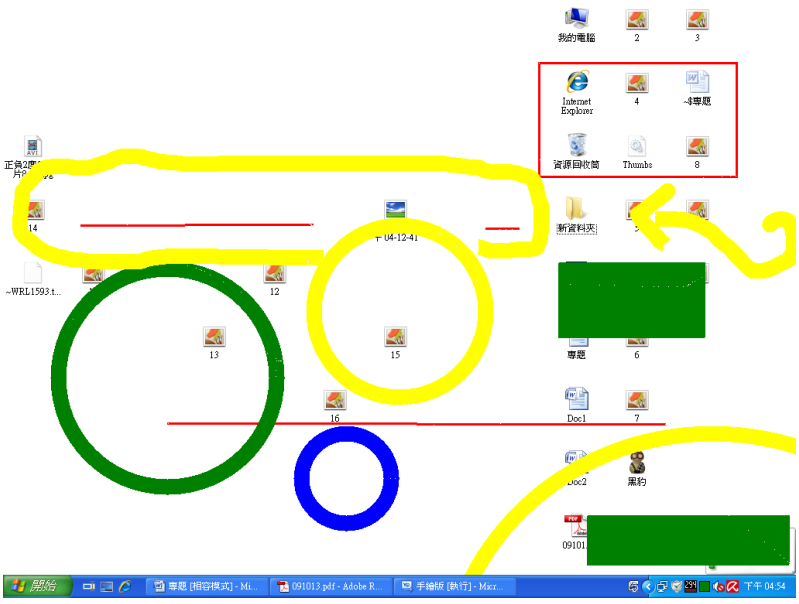


圖 4-8-3退出簡報後仍留下繪圖痕

(二) 螢幕畫布控制項基本架構

1. 畫筆顏色的設定

運用內建的顏色，在 ToolStripMenuItem 選取時更換畫筆的顏色，並判斷所勾選的選項及標示勾勾，在透明顏色選項部分爲依據 TransparencyKey 所指定的顏色，來構成透明，而 TransparencyKey 指定的顏色就是表單的背景色，而在其他選項顏色部分爲開起內建顏色選擇 (ColorDialog) 來指定畫筆顏色 (圖 4-9 爲畫筆顏色選擇的畫面)

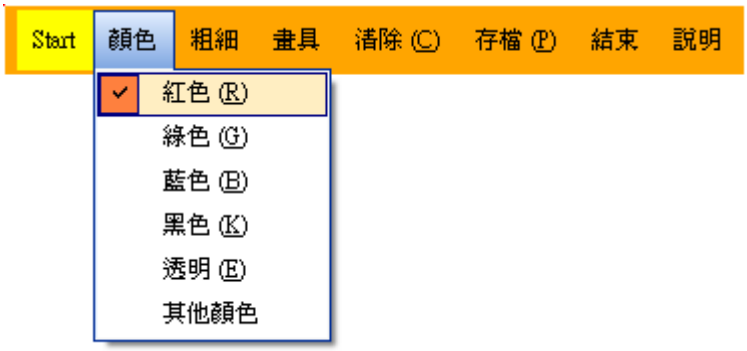


圖 4-9

圖 4-10 爲每個畫筆顏色設定的實際程式碼

編號 1： 紅色的 ToolStripMenuItem 按鍵。
編號 2~7：判斷所勾選的選項及標示選項前的勾勾，此爲選定紅色選項。
編號 8、9：指定畫筆、筆刷顏色爲紅色。
編號 10~12： 綠色、黑色以及藍色的 ToolStripMenuItem 按鍵。
編號 13： 選擇透明色的 ToolStripMenuItem 按鍵。
編號 14~19：判斷所勾選的選項及標示選項前的勾勾，此爲選定透明色選項。
編號 20、21：指定畫筆、筆刷顏色爲畫布背景色 (使 TransparencyKey 發揮作用) 。
編號 22： 其他顏色的 ToolStripMenuItem 按鍵。
編號 23： 顯示內鍵顏色選擇對話方塊 (ColorDialog)，且判斷在對話方塊上如果按下確定鍵時發生動作。
編號 24、25：指定畫筆、筆刷顏色爲顏色選擇對話框 (ColorDialog) 所選擇的顏色。
編號 26~31：判斷所勾選的選項及標示選項前的勾勾，此爲選定其他顏色選項。

```
    '顏色選擇
    Sub 紅色ToolStripMenuItem_Click(ByVal sender As System.Object,
                                    ByVal e As System.EventArgs) Handles 紅色ToolStripMenuItem.Click '1
        紅色ToolStripMenuItem.Checked = True  '2
        綠色ToolStripMenuItem.Checked = False '3
        藍色ToolStripMenuItem.Checked = False '4
        黑色ToolStripMenuItem.Checked = False '5
        透明ToolStripMenuItem.Checked = False '6
        其他顏色ToolStripMenuItem.Checked = False '7
        Form1.p.Color = Color.Red             '8
        Form1.sb.Color = Color.Red            '9
    End Sub

    Sub 綠色ToolStripMenuItem_Click ...       '10

    Sub 藍色ToolStripMenuItem_Click ...       '11

    Sub 黑色ToolStripMenuItem_Click ...       '12

    Sub 透明ToolStripMenuItem_Click(ByVal sender As System.Object,
                                    ByVal e As System.EventArgs) Handles 透明ToolStripMenuItem.Click '13
        紅色ToolStripMenuItem.Checked = False '14
        綠色ToolStripMenuItem.Checked = False '15
        藍色ToolStripMenuItem.Checked = False '16
        黑色ToolStripMenuItem.Checked = False '17
        透明ToolStripMenuItem.Checked = True  '18
        其他顏色ToolStripMenuItem.Checked = False '19
        Form1.p.Color = Form1.BackColor       '20
        Form1.sb.Color = Form1.BackColor      '21
    End Sub

    Sub 其他顏色ToolStripMenuItem_Click(ByVal sender As System.Object,
                                    ByVal e As System.EventArgs) Handles 其他顏色ToolStripMenuItem.Click '22
        If ColorDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then '23
            Form1.p.Color = ColorDialog1.Color '24
            Form1.sb.Color = ColorDialog1.Color '25
            紅色ToolStripMenuItem.Checked = False '26
            綠色ToolStripMenuItem.Checked = False '27
            藍色ToolStripMenuItem.Checked = False '28
            黑色ToolStripMenuItem.Checked = False '29
            透明ToolStripMenuItem.Checked = False '30
            其他顏色ToolStripMenuItem.Checked = True '31
        End If
    End Sub
```

圖 4-10

2. 畫筆粗細的設定

運用內建的指令來指定畫筆的 Width，在選取 ToolStripMenuItem 時更換畫筆的粗細，並判斷所勾選的選項及標示勾勾，在其他粗細選項部分，因避免再另外跳視窗來指定粗細，所以採用 TextBox 來直接輸入，指定粗細的最大值在 200 pt，(圖 4-11 爲畫筆粗細選擇的畫面)

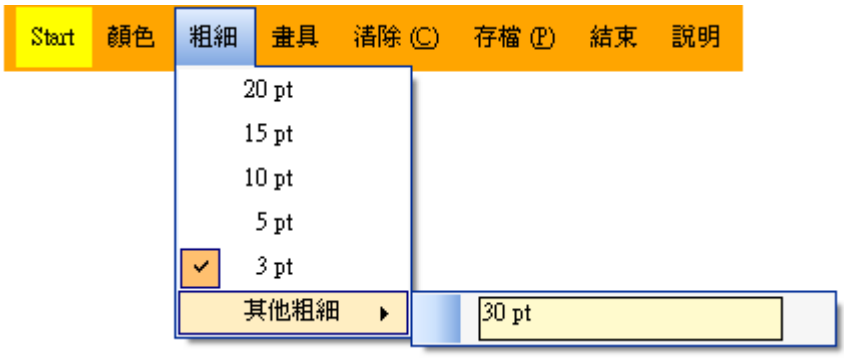


圖 4-11

圖 4-12 爲每個畫筆粗細設定的實際程式碼

編號 1：畫筆粗細 20 pt 的 ToolStripMenuItem 按鍵。

編號 2~7：判斷所勾選的選項及標示選項前的勾勾，此爲選定 20 pt 選項。

編號 8：指定畫筆粗細爲 20 pt。

編號 9~12：畫筆粗細 15 pt、10 pt、5 pt、3pt 的 ToolStripMenuItem 按鍵。

編號 13：選擇其他粗細的 ToolStripMenuItem 按鍵。

編號 14~19：判斷所勾選的選項及標示選項前的勾勾，此爲選定其他粗細選項。

編號 20：指定畫筆粗細爲 ToolStripMenuItem9 的 TextBox 框內的數字(val) 。

```
    '畫筆粗細
    Sub ToolStripMenuItem2_Click(ByVal sender As System.Object, _
                                 ByVal e As System.EventArgs) Handles ToolStripMenuItem2.Click '1
        ToolStripMenuItem2.Checked = True  '2
        ToolStripMenuItem3.Checked = False '3
        ToolStripMenuItem4.Checked = False '4
        ToolStripMenuItem5.Checked = False '5
        ToolStripMenuItem6.Checked = False '6
        其他粗細ToolStripMenuItem.Checked = False '7
        Form1.p.Width = 20 '8
    End Sub

    Sub ToolStripMenuItem3_Click ... '9

    Sub ToolStripMenuItem4_Click ... '10

    Sub ToolStripMenuItem5_Click ... '11

    Sub ToolStripMenuItem6_Click ... '12


    Private Sub ToolStripMenuItem9_Click(ByVal sender As System.Object, _
                                 ByVal e As System.EventArgs) Handles ToolStripMenuItem9.Click '13
        ToolStripMenuItem2.Checked = False '14
        ToolStripMenuItem3.Checked = False '15
        ToolStripMenuItem4.Checked = False '16
        ToolStripMenuItem5.Checked = False '17
        ToolStripMenuItem6.Checked = False '18
        其他粗細ToolStripMenuItem.Checked = True '19
        Form1.p.Width = Val(ToolStripMenuItem9.Text) '20
    End Sub
```

圖 4-12

3. 畫具的設定

(1) 基本的畫具

指定當前所要使用的畫具，而 drawtool 變數的值在開始或畫圖模式進行中被判斷以來指定不同的畫具所需的不同設定，利用選取 ToolStripMenuItem 時更換 drawtool 變數的值，並判斷所勾選的選項及標示勾勾 (圖 4-13 爲畫具選擇的畫面)

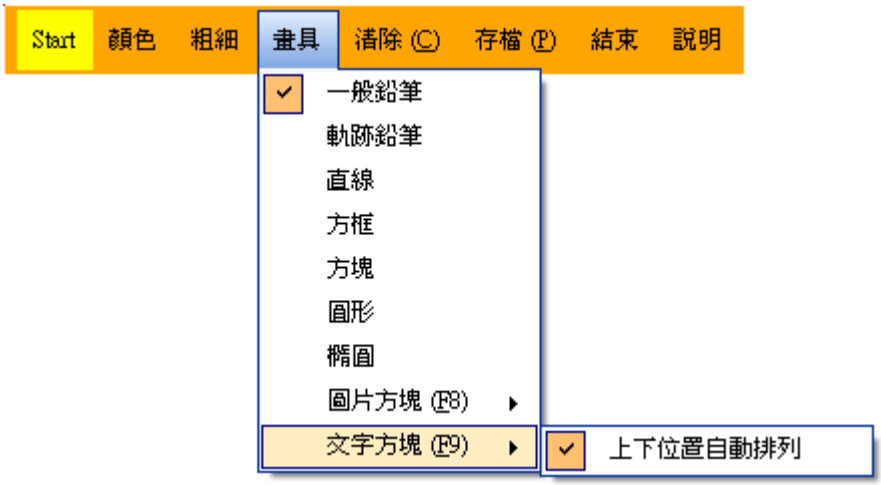


圖 4-13

圖 4-14 爲每個基本畫具變數設定的實際程式碼

編號 1：一般鉛筆的 ToolStripMenuItem 按鍵。

編號 2~10：判斷所勾選的選項及標示選項前的勾勾，此爲選定一般鉛筆選項。

編號 11：將 drawtool 變數的值設定爲 1，以供後續使用。

編號 12~17：軌跡鉛筆、直線、方框、方塊、圓形、橢圓的 ToolStripMenuItem 按鍵。

```
'畫具選擇
Sub 線ToolStripMenuItem_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles 線ToolStripMenuItem.Click '1
    線ToolStripMenuItem.Checked = True '2
    軌跡筆ToolStripMenuItem.Checked = False '3
    直線ToolStripMenuItem.Checked = False '4
    框ToolStripMenuItem.Checked = False '5
    方塊ToolStripMenuItem.Checked = False '6
    圓ToolStripMenuItem.Checked = False '7
    橢圓ToolStripMenuItem.Checked = False '8
    文字方塊ToolStripMenuItem.Checked = False '9
    圖片方塊ToolStripMenuItem.Checked = False '10
    Form1.drawtool = 1 '11
End Sub

Sub 軌跡筆ToolStripMenuItem_Click ... '12

Sub 直線ToolStripMenuItem_Click ... '13

Sub 框ToolStripMenuItem_Click ... '14

Sub 方塊ToolStripMenuItem_Click ... '15

Sub 圓ToolStripMenuItem_Click ... '16

Sub 橢圓ToolStripMenuItem_Click ... '17
```

圖4-14

(2) 進階的畫具

除了一般普通的畫筆功能外，最特別的就是軌跡鉛筆、圖片方塊以及文字方塊，(圖 4-15 爲選擇畫面)

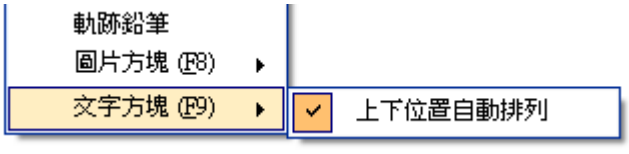


圖4-15

4. 清除的設定

選取 ToolStripMenuItem 時執行，並判斷所勾選的選項及標示勾勾 (圖 4-16 爲清除選擇畫面)

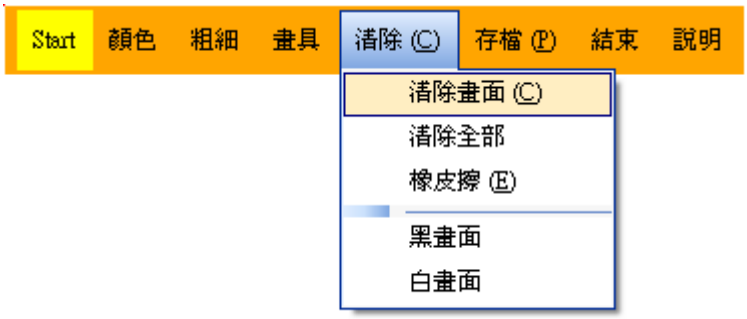


圖 4-16

圖 4-17 爲每個清除選擇的實際程式碼

編號 1： 將畫布的顏色設定爲 HotPink 以清除畫布 (TransparencyKey指定色爲HotPink)

編號 2： 重新整理畫布

編號 3、4：讓表單(功能表單、畫布表單)永遠在最上層

編號 5： 讓程式重新執行，可以直接清除所有畫面

編號 6： 呼叫 透明ToolStripMenuItem_Click(sender, e) 讓畫筆顏色變爲透明

編號 7： 判斷功能表單是否顯示，如果顯示就會在按下橡皮擦ToolStripMenuItem1_Click 的時候直接啓動畫圖模式(配合之後的快捷鍵)

```
    '清除選擇
    Private Sub 清除全部筆跡ToolStripMenuItem_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles 清除全部筆跡ToolStripMenuItem.Click
        Form1.g.Clear(Color.HotPink)                                    '1
        Refresh()                                                       '2
        Me.TopMost = True                                               '3
        Form1.TopMost = True                                            '4
    End Sub

    Private Sub 清除全部ToolStripMenuItem_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles 清除全部ToolStripMenuItem.Click
        Application.Restart()                                           '5
    End Sub

    Sub 橡皮擦ToolStripMenuItem1_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles 橡皮擦ToolStripMenuItem1.Click
        橡皮擦ToolStripMenuItem1.Checked = True
        黑畫面ToolStripMenuItem.Checked = False
        白畫面ToolStripMenuItem.Checked = False
        線ToolStripMenuItem.Checked = False
        軌跡筆ToolStripMenuItem.Checked = False
        直線ToolStripMenuItem.Checked = False
        框ToolStripMenuItem.Checked = False
        方塊ToolStripMenuItem.Checked = False
        圓ToolStripMenuItem.Checked = False
        橢圓ToolStripMenuItem.Checked = False
        文字方塊ToolStripMenuItem.Checked = False
        圖片方塊ToolStripMenuItem.Checked = False
        透明ToolStripMenuItem_Click(sender, e)                          '6
        If Me.Visible = True Then StartMenuItem1_Click(sender, e) '7
    End Sub
```

圖 4-17

圖 4-18爲黑畫面與白畫面功能的實際程式碼

黑畫面與白畫面則爲將畫布的顏色設定爲黑色或白色

```
Private Sub 黑畫面ToolStripMenuItem_Click(ByVal sender As System.Object, _
                        ByVal e As System.EventArgs) Handles 黑畫面ToolStripMenuItem.Click
    黑畫面ToolStripMenuItem.Checked = True
    白畫面ToolStripMenuItem.Checked = False
    橡皮擦ToolStripMenuItem1.Checked = False
    Form1.g.Clear(Color.Black)
    Refresh()
    Me.TopMost = True
    Form1.TopMost = False
End Sub

Private Sub 白畫面ToolStripMenuItem_Click(ByVal sender As System.Object, _
                        ByVal e As System.EventArgs) Handles 白畫面ToolStripMenuItem.Click
    黑畫面ToolStripMenuItem.Checked = False
    白畫面ToolStripMenuItem.Checked = True
    橡皮擦ToolStripMenuItem1.Checked = False
    Form1.g.Clear(Color.White)
    Refresh()
    Me.TopMost = True
    Form1.TopMost = False
End Sub
```

圖 4-18

5. 儲存的設定

選取 ToolStripMenuItem 時執行，並判斷所勾選的選項及標示勾勾 (圖 4-19 爲清除選擇畫面)

圖 4-19

圖 4-20 爲儲存設定的實際程式碼

編號 1：另存新檔的 ToolStripMenuItem 按鍵。

編號 2：將儲存對話方塊(SaveFileDialog) 的儲存格式限定爲 JPEG 格式檔

編號 3：顯示內建儲存對話方塊(SaveFileDialog)，且判斷在對話方塊上如果按下確定鍵時發生動作。

編號 4：將儲存對話方塊所選擇的路徑儲存至 saveadd 變數

編號 5：傳送 (PrtScn) 按鍵並等待執行結束

編號 6：宣告 Image 變數爲圖片

編號 7：將剛剛按下(PrtScn) 按鍵所儲存在剪貼簿的圖片，存至 Image 變數

編號 8：儲存圖片

編號 9：儲存檔案的 ToolStripMenuItem 按鍵。

編號 9-10 之間的程式碼功能與前述 (編號 5~7)相同

編號 10：自動命名成 “西元年-月-日 上/下午 時-分-秒.jpg” ，並存至主程式所在的目錄。例如：“2010-3-21 下午 04-54-19.jpg” (自動命名成當前的時間)

```
    '儲存檔案
    Private Sub 選擇儲存位置ToolStripMenuItem_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles 自訂檔名ToolStripMenuItem.Click '1
        SaveFileDialog1.Filter = "圖片檔(*.jpg)|*.jpg"                                          '2
        If SaveFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then                    '3
            saveadd = SaveFileDialog1.FileName                                                  '4
            SendKeys.SendWait("&{PRTSC}")                                                       '5
            Dim Image As Image                                                                  '6
            Image = System.Windows.Forms.Clipboard.GetDataObject.GetData _
            (System.Windows.Forms.DataFormats.Bitmap)                                           '7
            Image.Save(saveadd)                                                                 '8
        End If
    End Sub

    Public Sub SaveMenuItem_Click(ByVal sender As System.Object, _
                            ByVal e As System.EventArgs) Handles SaveMenuItem.Click             '9
        SendKeys.SendWait("&{PRTSC}")
        Dim Image As Image
        Image = System.Windows.Forms.Clipboard.GetDataObject.GetData _
        (System.Windows.Forms.DataFormats.Bitmap)
        Image.Save(Application.StartupPath & "\" & Replace(Replace _
        (Date.Now, "/", "-"), ":", "-") & ".jpg")                                               '10
    End Sub
```

圖 4-20

6. 快捷鍵的設定

爲了自訂快捷鍵而不是使用內建的快捷鍵設定，所以使用 Timer 元件，並引用User32 元件來判斷所按下的按鍵，圖 4-21 爲所有按鍵所對應的動作的實際程式碼。

```
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) _
                                                                        Handles Timer1.Tick
        If GetAsyncKeyState(Keys.PrintScreen) Then
            Form2.SaveMenuItem_Click(sender, e)
            PictureBox1.Visible = False
            startmark = 0
            If Form2.Visible = False Then t1.Abort()
        End If
        If GetAsyncKeyState(Keys.F9) Then
            Form2.文字方塊ToolStripMenuItem_Click(sender, e)
            If Form2.Visible = False Then
                If startmark = 1 Then t1.Abort()
                Form2.Show()
            End If
        End If
        If GetAsyncKeyState(Keys.F8) Then
            Form2.圖片方塊ToolStripMenuItem_Click(sender, e)
            If Form2.Visible = False Then
                If startmark = 1 Then t1.Abort()
                Form2.Show()
            End If
        End If
        If Form2.Visible = False Then
            If GetAsyncKeyState(Keys.C) Then
                g.Clear(Color.HotPink)
                Refresh()
            End If

            If GetAsyncKeyState(Keys.E) Then
                Form2.橡皮擦ToolStripMenuItem1_Click(sender, e)
            End If
            If GetAsyncKeyState(Keys.R) Then Form2.紅色ToolStripMenuItem_Click(sender, e)
            If GetAsyncKeyState(Keys.G) Then Form2.綠色ToolStripMenuItem_Click(sender, e)
            If GetAsyncKeyState(Keys.B) Then Form2.藍色ToolStripMenuItem_Click(sender, e)
            If GetAsyncKeyState(Keys.K) Then Form2.黑色ToolStripMenuItem_Click(sender, e)
            If GetAsyncKeyState(Keys.E) Then Form2.透明ToolStripMenuItem_Click(sender, e)
        End If
    End Sub
```

圖 4-21

7. 開始程式、暫停與結束

爲了方便使用，考慮到快速切換模式的問題，所以想了兩種切換方式

(1) 將滑鼠直接移至畫面左上角，即可暫停或開始繪圖模式。

(爲下圖中左上角**藍色**三角部分)

(2) 或直接使用滑鼠左鍵按下 Start 鍵，即可開始繪圖模式，暫停仍是移至左上角。

(3) 結束程式部分，直接使用滑鼠左鍵按下結束鍵即可結束程式。

圖 4-22 爲開始繪圖模式的實際程式碼

在執行緒裡放置畫筆的程式碼，啓動繪圖模式時就是啓動該執行緒

```
    '開始按鍵
    Sub StartMenuItem1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles StartMenuItem1.Click
        Me.Hide()
        Form1.startmark = 1
        Form1.PictureBox1.Visible = True
        Form1.t1 = New System.Threading.Thread(AddressOf Form1.Background)
        Form1.t1.Start()
        If 黑畫面ToolStripMenuItem.Checked = True Or _
        白畫面ToolStripMenuItem.Checked = True Then
            Form1.g.Clear(Color.HotPink)
            Refresh()
        End If
        黑畫面ToolStripMenuItem.Checked = False
        白畫面ToolStripMenuItem.Checked = False
        橡皮擦ToolStripMenuItem1.Checked = False
    End Sub
```

圖 4-22

圖 4-23 爲快速切換模式的實際程式碼 (停止或開始執行緒)

```
    Private Sub Panel1_MouseEnter(ByVal sender As Object, _
                        ByVal e As System.EventArgs) Handles Panel1.MouseEnter
        If Form2.Visible = True Then
            Form2.StartMenuItem1_Click(sender, e)
        Else
            t1.Abort()
            Form2.Show()
        End If
    End Sub
End Class
```

圖 4-23

圖 4-24 爲結束程式的實際程式碼

```
    Private Sub 結束ToolStripMenuItem_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles 結束ToolStripMenuItem.Click
        Form1.g.Dispose() '釋放畫布
        End               '結束程式
    End Sub
```

圖 4-24

三、 畫具的製作

引用 User32 元件來判斷滑鼠 X,Y 座標 及 所按下的滑鼠左鍵 (圖4-25)

```
Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" _
                                    (ByRef lpPoint As POINTAPI) As Integer

Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Integer) As Short
```

圖4-25

1. **一般鉛筆做法：**(引用 User32 元件來判斷是否按下滑鼠左鍵，程式碼如圖4-26)
   (1) PictureBox1 會先跟著滑鼠游標移動
   (2) 當按下滑鼠左鍵時，依照 PictureBox1 目前的位置(座標) 來判斷當前滑鼠的位置
   (3) 以 DrawLine 的方式來畫線
   (4) 當放開滑鼠左鍵時，重新再判斷滑鼠游標的位置 (PictureBox1再次跟著滑鼠)

```
Case 1
    If GetAsyncKeyState(1) Then
        g.DrawLine(p, s_a1, s_b1, POINTAP.x, POINTAP.y)
        s_a1 = POINTAP.x
        s_b1 = POINTAP.y
        PictureBox1.Visible = False
    ElseIf Not GetAsyncKeyState(1) Then
        MouseCheck = False
        PictureBox1.Visible = True
    End If
```

圖4-26

2. **軌跡鉛筆做法：**(引用 User32 元件來判斷是否按下滑鼠左鍵，程式碼如圖4-27)
   (1) 以 User32 元件來判斷滑鼠位置
   (2) 當按下滑鼠左鍵時，直接以 DrawLine 的方式來畫線，而不透過 PictureBox1 所在位置座標判斷 (PictureBox1隱藏)

```
Case 7
    If GetAsyncKeyState(1) Then
        g.DrawLine(p, s_a1, s_b1, POINTAP.x, POINTAP.y)
        s_a1 = POINTAP.x
        s_b1 = POINTAP.y
        PictureBox1.Visible = False
    ElseIf Not GetAsyncKeyState(1) Then
        MouseCheck = False
    End If
```

圖4-27

軌跡畫筆與一般的畫筆不同在於軌跡畫筆可以直接穿透畫面、直接選取視窗，又不影響當前的使用，所以當教師在做教學時，需要常常移動物件、或選取物件時，就可以使用軌跡畫筆以減少切換使用模式的時間及方便度。(圖 4-28 爲軌跡鉛筆實際執行圖)

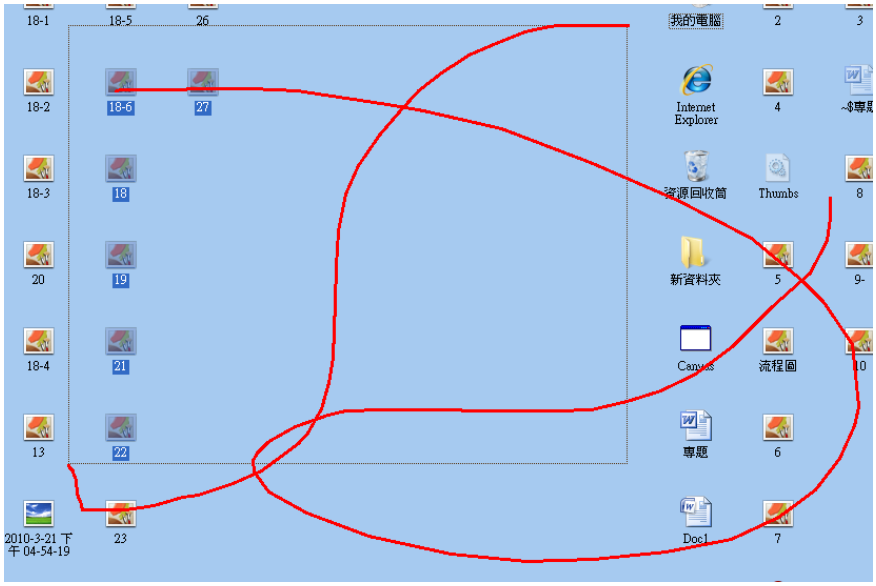


圖 4-28 軌跡鉛筆

3. **其他畫具作法：**(以 PictureBox1控制項的位置(座標)，來判斷滑鼠游標位置，程式碼如圖4-29，其他畫具如圖 4-30)
   (1) 以 MouseUp 的方式，判斷滑鼠左鍵放開時繪圖
   (2) 當 drawtool 變數爲 2 時，畫直線。drawtool 變數爲 3 時，畫方框。變數爲 4 時畫方塊。變數爲 5 時，畫圓形。變數爲 6 時，則是畫橢圓。

```
Private Sub PictureBox1_MouseUp(ByVal sender As Object, ByVal e As System.Windc
    Select Case drawtool
        Case 2
            g.DrawLine(p, s_a2, s_b2, e.X + PictureBox1.Location.X, PictureBox1
        Case 3
            g.DrawRectangle(p, s_a2, s_b2, (e.X + PictureBox1.Location.X) - s_a
        Case 4
            g.FillRectangle(sb, s_a2, s_b2, (e.X + PictureBox1.Location.X) - s_
        Case 5
            h = (e.X + PictureBox1.Location.X) - s_a2 + 1
            w = (e.Y + PictureBox1.Location.Y) - s_b2 + 1
            If w >= h Then
                h = w
            Else
                w = h
            End If
            g.DrawEllipse(p, s_a2, s_b2, w, w)
        Case 6
            g.DrawEllipse(p, s_a2, s_b2, (e.X + PictureBox1.Location.X) - s_a2
    End Select
End Sub
```

圖4-29

圖4-30 其他畫具

4. 文字方塊

爲了方便教師在上課時能補充上課的重點，且不需在黑板拿粉筆寫字或白版上寫字，甚至有些教師不熟悉使用滑鼠來在畫面直接寫字，都可以利用所提供的文字方塊來輸入，方塊可以隨時放在任何視窗上，輸入想要的任何文字，也可以直接以滑鼠拖曳位置，而這些文字可以隨意的改變顏色、大小和字型、顏色、移動大小、移除，如果有需要的話還可以直接印在螢幕上，讓教師和學生皆可遠離粉筆灰。程式碼如圖 4-31，文字方塊實際執行如圖 4-32)

(1) 以動態的方式新增 TextBox 與 ContextMenuStrip 控制項
(2) 定義新增出來的 TextBox 位置、名稱、大小、字體大小、字型、前景字顏色、且文字可輸入多行。
(3) 定義新增出來的 ContextMenuStrip 控制項項目，並連結至所要觸發的動作，有選擇字型、選擇顏色、移除方塊、釘住方塊選項。
(4) 設定 TextBox 與 ContextMenuStrip 控制項連結。
(5) 新增所定義好的動態 TextBox 至表單。
(6) 因爲要使文字方塊可以移動、變動大小，所以在最後時另外連結至三個所需觸發的動作。

```
Sub 文字方塊ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
                                                    Handles 文字方塊ToolStripMenuItem.Click
    文字方塊ToolStripMenuItem.Checked = True
    橡皮擦ToolStripMenuItem1.Checked = False
    黑畫面ToolStripMenuItem.Checked = False
    白畫面ToolStripMenuItem.Checked = False
    線ToolStripMenuItem.Checked = False
    軌跡筆ToolStripMenuItem.Checked = False
    直線ToolStripMenuItem.Checked = False
    框ToolStripMenuItem.Checked = False
    方塊ToolStripMenuItem.Checked = False
    圓ToolStripMenuItem.Checked = False
    橢圓ToolStripMenuItem.Checked = False
    圖片方塊ToolStripMenuItem.Checked = False
    Dim myTextBox As TextBox = New TextBox
    Dim myContextMenuStrip As ContextMenuStrip = New ContextMenuStrip
    myTextBox.Top = 100
    myTextBox.Name = "myTextBox1" & i.ToString
    i = i + 1
    myTextBox.Left = 30
    myTextBox.Size = New Size(300, 100)
    myTextBox.Font = New Font("Fixedsys", 60)
    myTextBox.ForeColor = Form1.sb.Color
    myTextBox.Multiline = True
    myContextMenuStrip.Items.Add("選擇字型", Nothing, AddressOf ContextMenuStrip_Click)
    myContextMenuStrip.Items.Add("選擇顏色", Nothing, AddressOf ContextMenuStrip1_Click)
    myContextMenuStrip.Items.Add("移除方塊", Nothing, AddressOf ContextMenuStrip2_Click)
    myContextMenuStrip.Items.Add("釘住方塊", Nothing, AddressOf ContextMenuStrip3_Click)
    myContextMenuStrip.BackColor = Color.White
    myTextBox.ContextMenuStrip = myContextMenuStrip
    Form1.Controls.Add(myTextBox)

    AddHandler myTextBox.MouseDown, AddressOf MyMouseDown
    AddHandler myTextBox.MouseMove, AddressOf textMouseMove
    AddHandler myTextBox.MouseLeave, AddressOf MyMouseLeave
End Sub

Sub ContextMenuStrip_Click ...
Sub ContextMenuStrip1_Click ...
Sub ContextMenuStrip2_Click ...

Sub ContextMenuStrip3_Click ...
```

圖4-31

圖 4-32 文字方塊

5. 圖片方塊

有些教學可能需要額外的圖片來做爲比較，這時候只要利用所提供的圖片方塊，直接把需要的圖拖曳至方塊內，就可以輕輕鬆鬆的展示圖片，由於這些方塊會一直處在螢幕的最上層，所以在切換視窗時，不必擔心找不到圖片，而浪費了找圖片的時間，可迅速的繼續教學，又不會中斷整個教學的時間。程式碼如圖 4-33，圖片方塊實際執行如圖 4-34)

(1) 以動態的方式新增 PictureBox 與 ContextMenuStrip 控制項
(2) 定義新增出來的 PictureBox 位置、名稱、大小、字體大小、字型、前景字顏色、且可以支援滑鼠直接拖曳。
(3) 判斷是否有圖片等待複製貼上，如果有，直接讀取需複製的參數，並執行貼上作業、清空判斷，把貼上的圖移至最上層，如果沒有則繼續執行，且圖片大小調爲最適 PictureBox框 (圖片方塊)大小。
(4) 定義新增出來的 ContextMenuStrip 控制項項目，並連結至所要觸發的動作，有選擇圖片、複製圖片、移除方塊、釘住方塊選項。
(5) 設定 PictureBox 與 ContextMenuStrip 控制項連結。
(6) 新增所定義好的動態 PictureBox 至表單。
(7) 因爲要使圖片方塊可以移動、變動大小、具有拖曳功能、複製圖片、且自動調整圖片大小，所以在最後時另外連結至六個所需觸發的動作。

```
Sub 圖片方塊ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    圖片方塊ToolStripMenuItem.Checked = True
    文字方塊ToolStripMenuItem.Checked = False
    橡皮擦ToolStripMenuItem1.Checked = False
    黑畫面ToolStripMenuItem.Checked = False
    白畫面ToolStripMenuItem.Checked = False
    線ToolStripMenuItem.Checked = False
    軌跡筆ToolStripMenuItem.Checked = False
    直線ToolStripMenuItem.Checked = False
    框ToolStripMenuItem.Checked = False
    方塊ToolStripMenuItem.Checked = False
    圓ToolStripMenuItem.Checked = False
    橢圓ToolStripMenuItem.Checked = False
    Dim myPictureBox As PictureBox = New PictureBox
    Dim myContextMenuStrip As ContextMenuStrip = New ContextMenuStrip
    myPictureBox.Name = "myPictureBox1" & i2.ToString
    i2 = i2 + 1
    If Not ps = "" Then
        myPictureBox.ImageLocation = ps
        myPictureBox.Size = New Size(bx, by)
        myPictureBox.BringToFront()
        ps = ""
    Else
        myPictureBox.Size = New Size(30, 30)
    End If
    myPictureBox.Top = 100
    myPictureBox.Left = 30
    myPictureBox.SizeMode = PictureBoxSizeMode.StretchImage
    myPictureBox.BackColor = Color.Black
    myContextMenuStrip.Items.Add("選擇圖片", Nothing, AddressOf ContextMenuStrip4_Click)
    myContextMenuStrip.Items.Add("複製圖片", Nothing, AddressOf ContextMenuStrip7_Click)
    myContextMenuStrip.Items.Add("移除方塊", Nothing, AddressOf ContextMenuStrip5_Click)
    myContextMenuStrip.Items.Add("釘住方塊", Nothing, AddressOf ContextMenuStrip6_Click)
    myPictureBox.AllowDrop = True
    myPictureBox.ContextMenuStrip = myContextMenuStrip
    Form1.Controls.Add(myPictureBox)
    AddHandler myPictureBox.MouseDown, AddressOf MyMouseDown
    AddHandler myPictureBox.MouseMove, AddressOf picMouseMove
    AddHandler myPictureBox.MouseLeave, AddressOf MyMouseLeave
    AddHandler myPictureBox.DragDrop, AddressOf pb_DragDrop
    AddHandler myPictureBox.DragEnter, AddressOf pb_DragEnter
    AddHandler myPictureBox.MouseDoubleClick, AddressOf MyMouseDoubleClick
End Sub
```

圖4-33

圖 4-34 圖片方塊

四、 特殊功能

1. **改變文字框或圖片框大小** (與VB設計模式中的控制項一樣具有八個調整點)，程式碼如圖4-35)

   編號 1：動態讀取當前文字框或圖片框的名稱

   編號 2：如果上下位置自動排列選項有勾選，則將當前方塊框移至最上層

   編號 3：當按下滑鼠左鍵

   編號 4~5 之間：判斷滑鼠游標在方塊框的位置

```
    Private Sub textMouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs)
1:      tname = DirectCast((sender), TextBox).Name.ToString()
2:      If 自動排列ToolStripMenuItem1.Checked = True Then t.BringToFront()
3:      If e.Button = MouseButtons.Left Then
4:          Select Case m_MousePointPosition
                Case EnumMousePointPosition.MouseDrag
                    sender.Location = New Point(sender.Left + e.X - p0.X, sender.Top + e.Y - p0.Y)
                Case EnumMousePointPosition.MouseSizeBottom
                    sender.Size = New Size(sender.Width, sender.Height + e.Y - p1.Y)
                    p1 = New Point(e.X, e.Y)
                Case EnumMousePointPosition.MouseSizeBottomRight
                    sender.Size = New Size(sender.Width + e.X - p1.X, sender.Height + e.Y - p1.Y)
                    p1 = New Point(e.X, e.Y)
                Case EnumMousePointPosition.MouseSizeRight
                    sender.Size = New Size(sender.Width + e.X - p1.X, sender.Height)
                    : p1 = New Point(e.X, e.Y)
                Case EnumMousePointPosition.MouseSizeTop
                    sender.Location = New Point(sender.Left, sender.Top + (e.Y - p0.Y))
                    sender.Size = New Size(sender.Width, sender.Height - (e.Y - p0.Y))
                Case EnumMousePointPosition.MouseSizeLeft
                    sender.Location = New Point(sender.Left + e.X - p0.X, sender.Top)
                    sender.Size = New Size(sender.Width - (e.X - p0.X), sender.Height)
                Case EnumMousePointPosition.MouseSizeBottomLeft
                    sender.Location = New Point(sender.Left + e.X - p0.X, sender.Top)
                    sender.Size = New Size(sender.Width - (e.X - p0.X), sender.Height + e.Y - p1.Y)
                    p1 = New Point(e.X, e.Y)
                Case EnumMousePointPosition.MouseSizeTopRight
                    sender.Location = New Point(sender.Left, sender.Top + (e.Y - p0.Y))
                    sender.Size = New Size(sender.Width + (e.X - p1.X), sender.Height - (e.Y - p0.Y))
                    p1 = New Point(e.X, e.Y)
                Case EnumMousePointPosition.MouseSizeTopLeft
                    sender.Location = New Point(sender.Left + e.X - p0.X, sender.Top + (e.Y - p0.Y))
                    sender.Size = New Size(sender.Width - (e.X - p0.X), sender.Height - (e.Y - p0.Y))
5:                  End Select
```

圖4-35

編號 6~7 之間：改變滑鼠游標 (依據滑鼠游標在方塊框的位置)，程式碼如圖4-36

```
6:          m_MousePointPosition = MousePointPosition(sender.Size, e)  '判斷游標的位置狀態
            Select Case m_MousePointPosition    '改變游標
                Case EnumMousePointPosition.MouseSizeNone
                    Me.Cursor = Cursors.Arrow       '箭頭
                Case EnumMousePointPosition.MouseDrag
                    Me.Cursor = Cursors.SizeAll     '四方向
                Case EnumMousePointPosition.MouseSizeBottom
                    Me.Cursor = Cursors.SizeNS      '南北
                Case EnumMousePointPosition.MouseSizeTop
                    Me.Cursor = Cursors.SizeNS      '南北
                Case EnumMousePointPosition.MouseSizeLeft
                    Me.Cursor = Cursors.SizeWE      '東西
                Case EnumMousePointPosition.MouseSizeRight
                    Me.Cursor = Cursors.SizeWE      '東西
                Case EnumMousePointPosition.MouseSizeBottomLeft
                    Me.Cursor = Cursors.SizeNESW    '東北到南西
                Case EnumMousePointPosition.MouseSizeBottomRight
                    Me.Cursor = Cursors.SizeNWSE    '東南到西北
                Case EnumMousePointPosition.MouseSizeTopLeft
                    Me.Cursor = Cursors.SizeNWSE    '東南到西北
                Case EnumMousePointPosition.MouseSizeTopRight
                    Me.Cursor = Cursors.SizeNESW    '東北到南西
7:                  End Select
```

圖 4-36

編號 8~9 之間：開始判斷觸發移動方塊框或調整方塊框大小 (依據滑鼠游標在方塊框的位置) ，程式碼如圖4-37

```
    Private Function MousePointPosition(ByVal Size As Size, ByVal e _
    As System.Windows.Forms.MouseEventArgs) As EnumMousePointPosition
8:      Const Band = 10

        If e.X >= -1 * Band And e.X <= Size.Width And e.Y >= -1 * Band And e.Y <= Size.Height Then
            If e.X < Band Then
                If e.Y < Band Then
                    Return EnumMousePointPosition.MouseSizeTopLeft
                ElseIf e.Y > -1 * Band + Size.Height Then
                    Return EnumMousePointPosition.MouseSizeBottomLeft
                Else
                    Return EnumMousePointPosition.MouseSizeLeft
                End If
            ElseIf e.X > -1 * Band + Size.Width Then
                If e.Y < Band Then
                    Return EnumMousePointPosition.MouseSizeTopRight
                ElseIf e.Y > -1 * Band + Size.Height Then
                    Return EnumMousePointPosition.MouseSizeBottomRight
                Else
                    Return EnumMousePointPosition.MouseSizeRight
                End If
            Else
                If e.Y < Band Then
                    Return EnumMousePointPosition.MouseSizeTop
                ElseIf e.Y > -1 * Band + Size.Height Then
                    Return EnumMousePointPosition.MouseSizeBottom
                Else
                    Return EnumMousePointPosition.MouseDrag
                End If
            End If
        Else
            Return EnumMousePointPosition.MouseSizeNone
9:      End If
    End Function
```

圖 4-37

2. **拖曳圖片檔案功能**，程式碼如圖 4-38

編號 1：讀取當前滑鼠所拖曳的檔案的路徑，且直接載入至當前的圖片方塊中

編號 2：讀取當前滑鼠所拖曳的檔案的路徑，再次載入(這裡爲複製圖片所需)

編號 3：如果檔案不爲圖檔，則顯示 Msgbox 對話框。

編號 4、5：當滑鼠在圖片方塊中放開滑鼠左鍵則顯示圖片至方塊中。

```
    Private Sub pb_DragDrop(ByVal sender As Object, ByVal e As System.Windows.Forms.DragEventArgs)
        Try
1:          p.Image = Image.FromFile(CType(e.Data.GetData(DataFormats.FileDrop), Array).GetValue(0) _
            .ToString)
2:          p.ImageLocation = CType(e.Data.GetData(DataFormats.FileDrop), Array).GetValue(0).ToString
        Catch ex As Exception
3:          MessageBox.Show("Error Doing Drag/Drop")
        End Try
    End Sub

    Private Sub pb_DragEnter(ByVal sender As Object, ByVal e As System.Windows.Forms.DragEventArgs)
4:      If (e.Data.GetDataPresent(DataFormats.FileDrop)) Then
5:          e.Effect = DragDropEffects.Copy
        End If
    End Sub
```

圖 4-38

伍、 研究結果

我以兩個表格來比較，表一是與一般坊間的電子白板來做比較，教學螢幕畫布與電子白板最大的差異爲完全不需要建置的經費，即可完成電子白板所擁有的功能，只要使用學校的電腦教室教學廣播系統，甚至是跟電子白板一樣使用單槍投影，一樣可以達到互動性，也可以邊講述邊教學。

表一、創意教學螢幕畫布與電子白板比較

	智****E****B****	S****B****	Windows 創意教學螢幕畫布
建置經費(不含單槍)	5 萬元	10 萬元	0 元
多媒體呈現	OK	OK	OK
講述式教學	OK	OK	OK
要達到互動性	困難	容易	容易
達到互動性配合的硬體	必須(一筆可觀的經費)	不用	不用
學生參與的資訊能力	需要	不需要	不需要
互動性展現的適合年級	四、五、六	低、中、高	皆可

表二是與網路上常見的螢幕繪圖程式比較，網路上這些類似的程式，可以在螢幕上繪圖，也可以達到教學的目的，但是幾乎都不能輸入文字、畢竟都是國外製作的，如果有輸入文字的程式，大部分也只能輸入英文，有的繪圖的畫面可以存檔，但絕大部分都不行，最重要的是在實用性上，網路上的程式幾乎不能在動態的背景上繪圖，也不能一邊繪圖一邊捲動頁面，但是這些不可能的任務，『Windows 創意教學螢幕畫布』每一樣都辦的到，且做得更好。

表二、創意教學螢幕畫布與一般繪圖軟體比較

	ZoomIt	Screen Marker	Pointofix	Windows 創意教學螢幕畫布
輸入文字	英文	不可	**中英文**	**任何文字**
插入圖片	不可	不可	不可	**可**
畫面存檔	可	不可	可	可
播放影片	不可	**可**	不可	**可**
捲動頁面	不可	**可**	不可	**可**
系統資源	**需求低**	需求很高	**需求低**	**需求低**
安裝模式	單一檔案執行	單一檔案執行	單一檔案執行	單一檔案執行

陸、 討論

經由程式執行的過程，發現有幾個問題，需要進一步解決：

一、 畫具在判斷座標時，造成大量 CPU 資源損耗的問題？

由於此程式的畫布是透明的，所以無法直接透過表單來讀取滑鼠座標，然而畫筆需要座標才能判斷當前的位置，所以我透過媒介來判斷當前滑鼠的座標，一開始是希望能”即時”讀取到所需的座標值，而使用 Timer 來不斷讀取，進而造成資源損耗，甚至有時會當機，後來想到可以利用以前在網路上學到的多執行緒來處理這部分，現在也成功削減了許多浪費掉的 CPU 資源。

二、 筆跡有時候在碰到別的視窗會消失？

我想到既然筆跡無法存在畫面上(因爲我的表單是透明的)，那何不把畫布表單設定爲永遠在最上層，這樣處理過後筆跡果然會留在螢幕上，但有時候仍會有部分消失的情況，因爲不像普通的畫布可以直接在圖片畫布表單上重繪，這個部份還需在加強。

三、 橡皮擦和透明顏色有時候會失效?

一般的畫布可能沒有這個問題，因爲那些程式都是以圖片作爲底，橡皮擦或者是透明的顏色只需使用白色來覆蓋痕跡就好了，但是我的畫布是透明的底，無法使用白色來覆蓋繪圖的痕跡，所以我使用之前所提到的 TransparencyKey 屬性來處理，可惜很不幸的都測試失敗了，於是我到網路上找了些有關的資料，查到的結果原來是 TransparencyKey 屬性自己本身有錯誤，就是當監視器的色彩深度設定爲大於 24 位元的數值，會失效。由於網路上也沒有很好的解決方法，所以我把表單背景色改成表單原來的顏色，不過這樣處理過後，反而比較少失效了。

柒、 結論

經由本研究的過程，已經把電子白板的功能融入 Windows 創意教學螢幕畫布中，不但操作簡單，不需要增加任何的硬體設備，使用者可以在任何的 windows 畫面(含桌面、套裝軟體、簡報)中畫線條、圖形或插入圖片。教師可利用電腦教室內的廣播系統，把 windows 畫面傳給每一個學生。而且如果搭配手寫版，程式的教學效果會更好。

在製作程式的過程中，爲了能更了解教師在教學使用上的需求，所以我常常把修改好的程式拿給他們使用，我也把他們的建議做了一些改善，例如：當在教學時，常常需要補充重點，因此我做了文字和圖片方塊功能，使重點可以更清楚的呈現。而有些教師希望能邊畫重點還能邊動作電腦的操作，因此我對於這個問題而特別製做了軌跡鉛筆。在快捷鍵部分也依照使用的需求而設定，例如：清除、橡皮擦、換顏色的快捷鍵分別 C、E、R。而且爲了能使切換模式的動作快速，做了兩種切換的方式，直接把滑鼠游標移至螢幕左上角即可快速切換。

未來螢幕畫布如果能具備螢幕錄影功能，把教學的過程整個錄製下來，且如果將來技術成熟，也許就能夠把當前的繪圖痕跡所記錄，當在捲動螢幕畫面時，繪圖痕跡就能夠隨著滑鼠滾輪移動，可以省去在當前頁面繪圖完後還需清除的麻煩，當這些都完成後，整體可能會有更高的價值。

捌、參考資料及其他

一、認識 hwnd(Handle of Window) - 探索電腦 - Jeremy 的部落格
http://jeremyblog.twbbs.org/2009/07/hwndhandle-of-window.html

二、用VB 來教繪圖應用的經驗 國立台灣師大附中 李啓龍 老師
http://icerc.tnssh.tn.edu.tw/download/epaper/epaper24/20080430.pdf

三、電子白板的使用經驗與選擇 - rainlan's pLog
http://blog.ilc.edu.tw/blog/blog/2/post/1172/25924

四、如何於執行期拖曳移動控制項 - 強力鄉頭 VB BLOG - 點部落
http://www.dotblogs.com.tw/PowerHammer/archive/2009/10/01/10872.aspx

五、微軟 MSDN 線上論壇
六、微軟技術社群討論區

【評語】091005

專題作品製作過程詳實，表達能力生動活潑，且能察覺上課中老師使用多媒體教學的不便之處，並利用所學的專業知識解決問題，值得嘉許。

教學螢幕畫布要成為更完整之應用軟體，需再加深加廣，可多參考其他多媒體廣播教學軟體的各項功能，會更完整。