第 15 章射擊遊戲

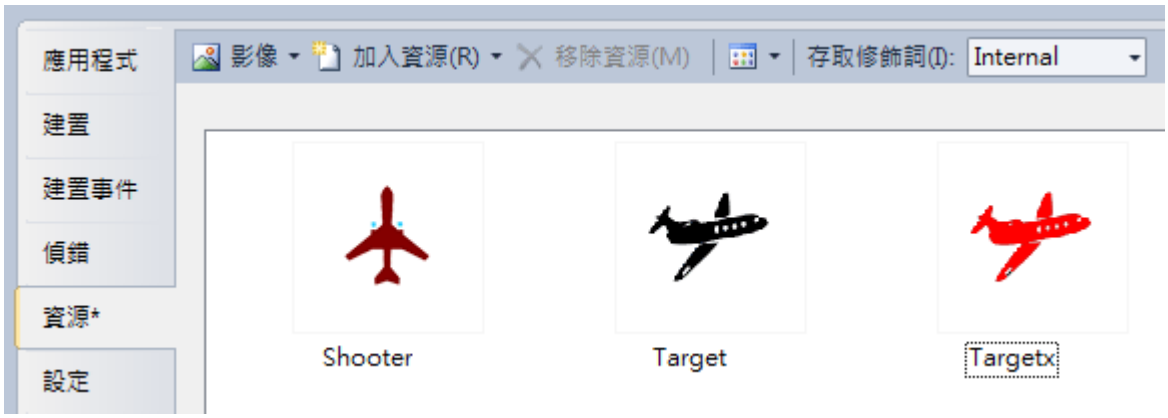
簡介：

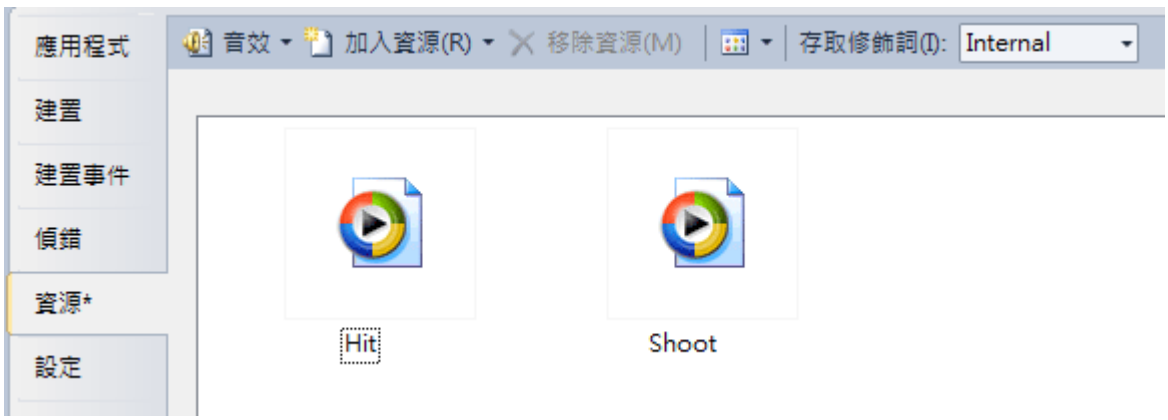
射擊遊戲是非常主流的遊戲種類，它的基本元素包括：可以移動及射擊的槍支、彈藥流失與得分的累計，以及至少一個會移動的目標，當然射擊以及擊中目標的音效也是很重要的一部分。本專案將以最精簡的程式技術，代表性的設計出以上的各項功能，讓同學們體驗一下此類程式運作的核心。

15-1 加入資源檔案

[加入圖案與音效資源]

開啟專案後打開方案總管中的 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」，畫面如下。請依序加入三個圖檔(Shooter.gif，Target.gif 與 Tagetx.gif) 以及音效檔(Shoot.wav & Hit.wav)，稍後三個圖將分別作為槍支與目標，其中 Target 是目標正常狀態，Targetx 是被擊中時的圖案，兩音效則分別為槍聲與擊中目標的音響。資源檔案加入之後會被自動分類到音效與影像兩個類別的頁面之中，請切換檢視確認，不要誤會匯入檔案失敗了！這些圖與音效檔案可自書附光碟中取得，自己設計或搜尋網路資源當然也無妨。

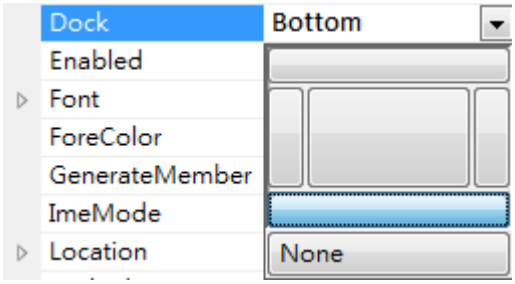




15-2 表單物件佈置

[建立操作面板]

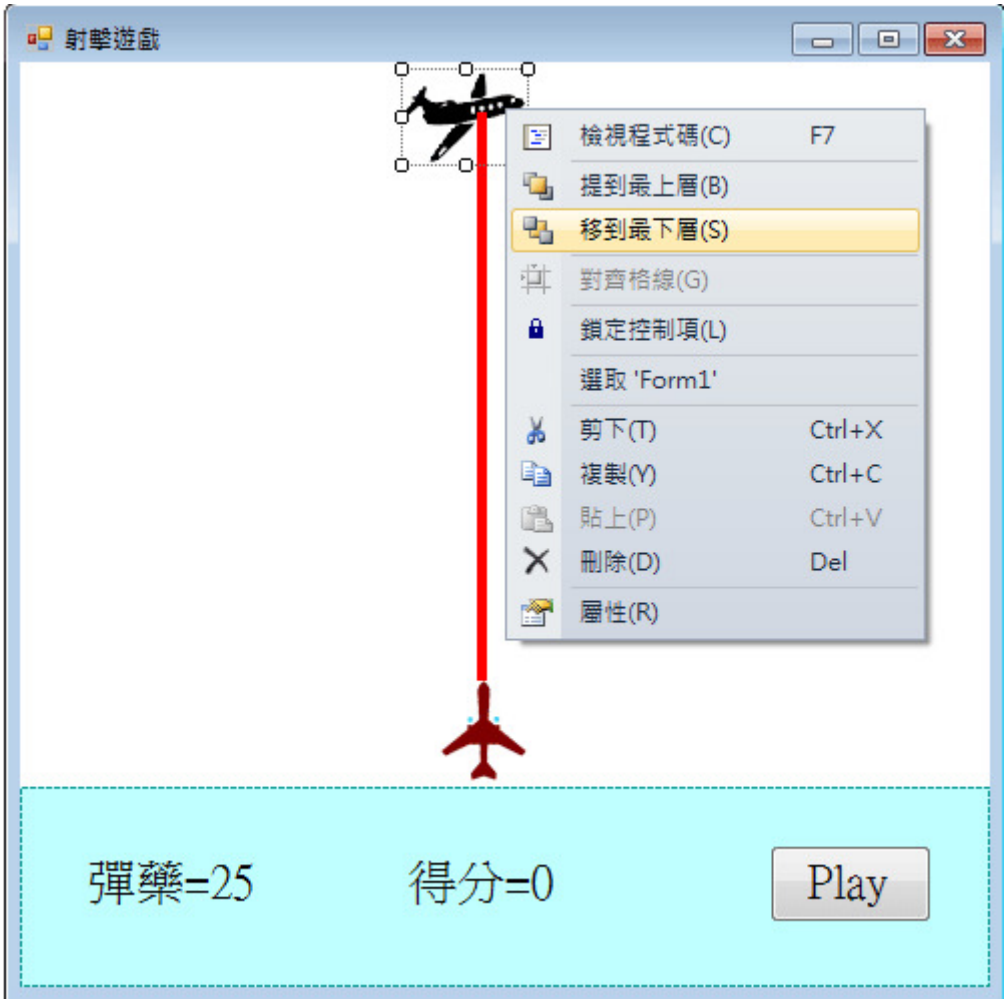
首先將設計表單拉大到約 500 像素點長寬，標題改為「射擊遊戲」，到工具箱的容器項目載入一個面板(Panel)物件，到它的 Dock 屬性處選擇置於表單下方(Bottom)。如下圖：



在 Panel 物件中佈置兩個標籤，分別寫入「彈藥=25」(label1)以及「得分=0」(label2)，再加入一個重玩按鍵(button1)。你可以自行調整這些物件的屬性加以修飾，譬如字放大一點，或給予框線背景色等等，以強化展示效果。

[建立目標、槍枝與雷射光物件]

接著在 Panel 物件上方的區域加入三個標籤(Label)物件，將它們分別製作成槍支、目標以及雷射光線，且分別重新命名為 G(槍支)，T(目標)以及 L(雷射光)，以簡化後續的程式撰寫，如果物件的相疊層次不如預期，可以使用按右鍵跳出選單的移到最上或作下層調整。如下圖：

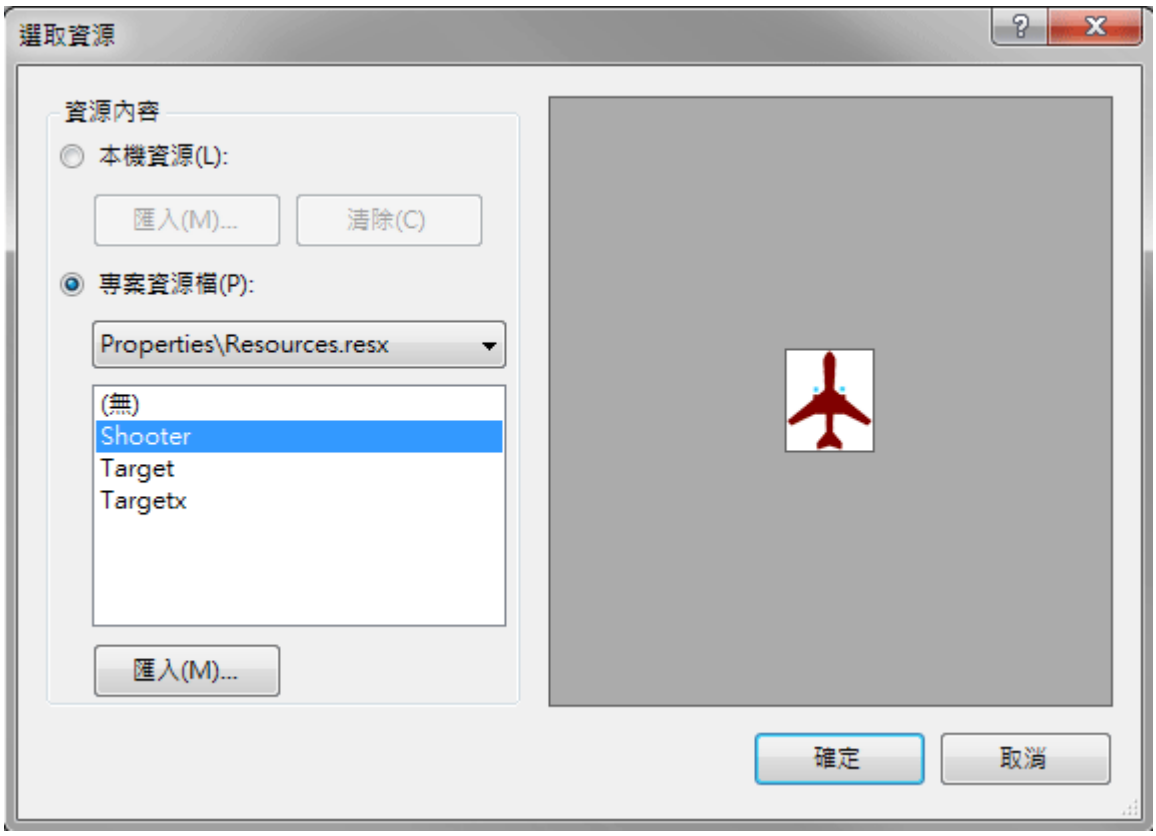


如同前面單元曾經作過的介紹，在此處如果使用 PictureBox 製作槍枝、目標甚至雷光都是一定可以的，且製作過程比較簡單。但 PictureBox 會消耗較多記憶體；標籤物件也可以顯示圖案，但是功能比較簡單，所佔記憶體也較小，在多數遊戲設計中記憶體使用越精簡表示可以容納更多元件，也可以加快執行速度，斤斤計較記憶體的用量是必要的，所以學會用標籤製作圖案物件確有實質效益！

[用標籤製作圖案的步驟]

用標籤製作圖案，首先必須將 AutoSize 屬性設定為 False，接著要將文字刪除，並在 Image 屬性中載入資源圖檔，畫面如下圖。因為標籤的 AutoSize 屬性只會依據「文字」內容縮放物件本身，不會因應「影像」大小縮放，所以必須停止標籤的 AutoSize 功能，圖案載入之後還必須參考原圖大小調整物件框線。如果要充分的精確必須在屬性欄中定義 Size 的 Width 與 Height，使它們與原圖長寬一致。至於雷射光(L)的部分，並不需要圖檔(設定值為→無)，只要設定紅色的背景色(BackColor)即可。

事實上，如下圖以本專案來說如果你直接使用「本機資源」的方式跳過資源檔案設定，直接載入圖檔結果並無差別。不過，對於較大的專案而言，養成使用資源檔案的習慣比較好，不然會有許多物件重複載入一樣的圖檔，最終很難精準的管理以節約資源。



15-3 限制範圍的槍支移動

[拖曳槍枝物件的程式]

槍支移動就是程式執行中的拖曳動作，稍稍不同的是在此例中我們限制槍支只能在畫面下方水平的移動，否則玩家將槍支直接拿到目標旁邊開槍就不必瞄準了！哈哈！當然，如果不小心動作太大將槍支移出畫面之外抓不回來，也是很傷腦筋的事情，所以如同之前的乒乓球單元，也必須限制槍枝移動的範圍不可超出表單。此部分程式碼的邏輯說明可參考乒乓球單元。以下為槍支移動的程式碼：

```
int mdx;
private void G_MouseDown(object sender, MouseEventArgs e)
{
    mdx = e.X;
}
private void G_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
       int X = G.Left + e.X - mdx;
       if (X < 0) { X = 0; }
       if (X+G.Width > this.ClientSize.Width) { X = this.ClientSize.Width - G.Width; }
       G.Left = X;
       L.Left = G.Left + G.Width / 2 - L.Width / 2;
    }
}
```

首先是宣告公用變數 mdx，在 MouseDown 事件中將拖曳起點記錄在 mdx；在

MouseMove 事件中先確定滑鼠為拖曳狀態(e.Button=…Left)，之後宣告一個變數 X，試算一下槍支預期的新位置，就是 G.Left 加上 X 座標移動量(終點的 e.X 減去起點的 mdx)。如果超出左右邊界時強制 X 要在邊界值內，最後才將槍支(G)移動到此 X 的位置。

[調整雷射光追隨槍枝的位置]

由於雷射光應該始終與槍支位置一致，所以最後要將雷射光(L.Left)移到槍支的中心軸位置(G.Left + (G.Width - L.Width) / 2)。測試一下程式會發現槍枝與雷射光連成一體移動，到了邊緣會『卡住』無法移出畫面。

15-4 雷射光的視覺效果

[按 Z 鍵發射雷射光]

射擊時雷射光(L)是主角，基本上它應該是隱形的(初始設定 Visible=False)，我們希望按下鍵盤的 Z 鍵時表示發射，每次發射雷射光會伴隨音效出現一個短暫的時間，隨即消失。要作到這些效果先要設定表單的 KeyPreview 屬性為 True 讓表單具備處理鍵盤事件的能力。接著要載入一個 timer1 元件，設定 Interval=200 即 1/5 秒，這是預定雷射光每次顯示的時間。下面是相關的程式碼：在表單的鍵盤事件 KeyDown 中，先檢查如果是 Z 鍵表示射擊，先讓雷射光出現，啟動計時器；在計時器事件中，一到了預定的 1/5 秒，會將雷射光隱藏，同時也將此計時器停止。現在我們有了一支可以移動並能射擊的槍支了！

```
 private void timer1_Tick(object sender, EventArgs e)
{
    L.Visible = false;
    timer1.Stop();
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Z)
    {
        L.Visible = true;
        timer1.Start();
    }
}
```

[定時熄滅的雷射光]

在此或許有人會問：既然用 KeyDown 事件開啟雷射光，用 KeyUp 關閉雷射不就好了？為何要用這種比較複雜的計時器控制法？原因是：如果這樣設計，就會有玩家一直壓著發射鍵盤，永遠保持發射狀態，看起來不太正常，就是玩家必須再按鍵一次才能再度發射。不過一樣的功能確實有很多種設計的方法，如同數學題目常常也有多種解法，真正好的程式其實必須考慮多種寫法找出執行時最有效率，視覺上合理，程式碼也最為精簡的寫法。

試試看！現階段的程式你已經可以拖曳槍枝，隨時按下 Z 鍵射擊，射擊時還會產

生雷射光一閃而逝的效果，當然雷射光要能一直依附在槍枝正前方才對！

15-5 移動的目標

[製作循環往返的移動目標]

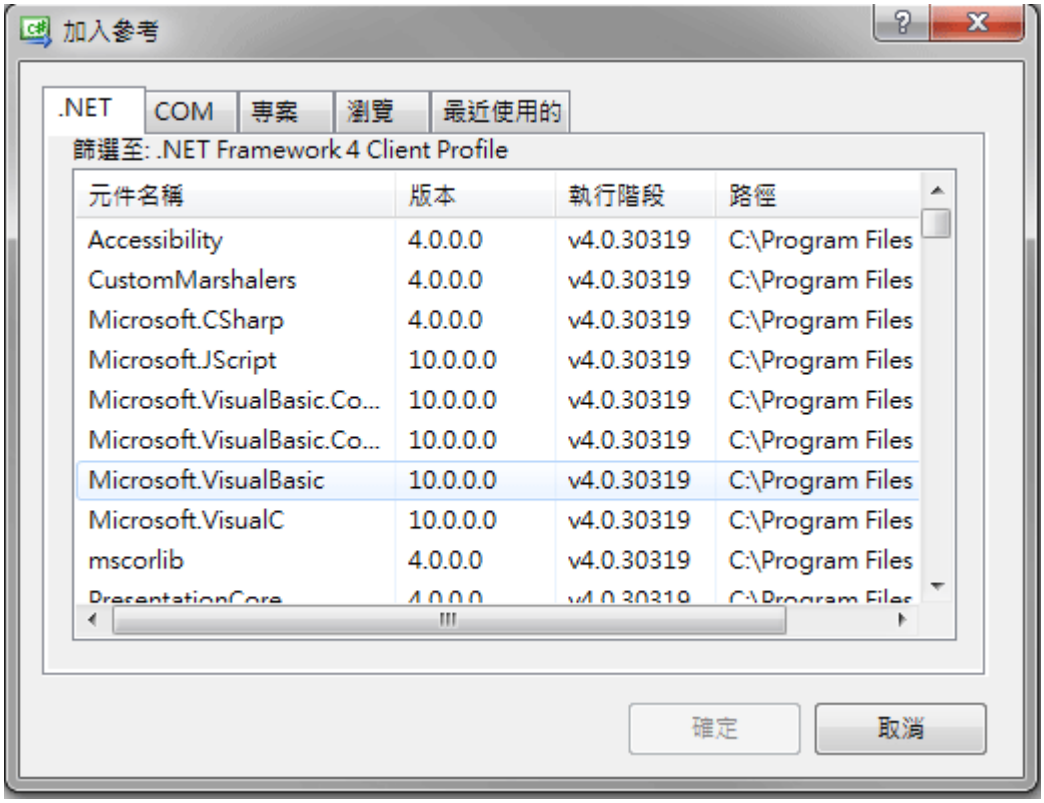
接下來要作一個會移動的目標(T)，在此例中是一架機頭朝右的飛機，飛出畫面之後還會再度從另一邊重複出現。這一部分程式與之前螢幕保護程式中移動的小時鐘非常類似，可以翻閱參考。在此請載入一個 timer2 物件，使用預設的 Interva=100，並直接開啟它(使屬性 Enabled=True)。timer2 的功能是定時往右移動目標(T.Left+=5)，我們預計讓飛機朝右邊飛去，但是如果飛出畫面之後目標要移到畫面最左邊之外，再度飛入畫面之中。程式碼如下：

```
private void timer2_Tick(object sender, EventArgs e)
{
    T.Left += 5;
    if (T.Left > this.ClientSize.Width)
    { T.Left = -T.Width; }
}
```

15-6 加入射擊與擊中效果

[加入播放音效程式的參考]

射擊時當然要加入聲音才好玩，而且射中與否必須有所區別，讓擊中目標的玩家得到更多樂趣。如打地鼠單元說過的，C#沒有自己預設的音效播放程式，所以必須向 VB 借用！要用音效首先必須先到功能表的「專案」項目選取「加入參考」，出現選擇視窗後到.NET 頁籤選擇「Microsoft Visual Basic」



還要到程式碼前端加上兩個 using 指令：

```
using Microsoft.VisualBasic;
using Microsoft.VisualBasic.Devices;
```

接著請在程式碼的公用區域，就是在副程式外面加上一行宣告「Computer」物件 C 的程式碼，稍後就用這個物件來播出音效：

```
Computer C = new Computer();
```

[擊中與否的音效設計]

我們是用鍵盤發射武器的，所以相關程式應該由 KeyDown 事件加以修改。判斷是否「擊中」目標的關鍵程式是：如果雷射光的整個寬度都在目標的寬度範圍內，就是雷射光的「左緣」(L.Left)大於目標的「左緣」(T.Left)；且光的右緣(L.Right)小於目標的右緣(T.Right)，就可視為擊中！應該給予一個較精采的撞擊音效(Hit.wav)，而且被擊中的目標圖案也要變成 Targetx；否則就使用簡單的發射音效(Shoot.wav)。程式碼修改如下：

```
private void Form1_KeyDown(object  sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Z)
    {
        L.Visible = true;
        timer1.Start();
        if (L.Left > T.Left && L.Right < T.Right)
        {
            C.Audio.Play(Properties.Resources.Hit,  AudioPlayMode.Background);
            T.Image = Properties.Resources.Targetx;
        }
        else
        {
            C.Audio.Play(Properties.Resources.Shoot,  AudioPlayMode.Background);
        }
    }
}
```

if 條件式中的"&&"記號是"而且"的意思，很多程式語言會使用 AND。上面程式的射中條件就是：雷射光的左緣(L.Left)大於(>)目標的左緣(T.Left)，而且(&&)雷射光的右緣(L.Right)小於(<)目標的右緣(T.Right)。播放音效則是使用我的電腦(Computer)物件 C 的音響(Audio)播放(Play)某個檔案；因為我們已經將要播的音效納入為專案資源(Resources) ，所以呼叫 Properties.Resource 即可。射中時播 Hit(擊中)，沒射中播 Shoot。

[擊中目標時圖案的改變]

另外目標被射中射還會變個圖案(T.Image = Properties.Resources.Targetx;)，黑色飛機瞬時變成紅色！當然被擊中之後希望不是自此就保持 Trargetx 的樣貌，所以應該將 timer1 事件變成這樣，使圖案在射擊結束之後恢復。

```
 private void timer1_Tick(object  sender, EventArgs e)
{
    L.Visible = false;
    T.Image = Properties.Resources.Target;
    timer1.Stop();
}
```

此時測試程式擊中畫面大致如下，你也應該會聽到擊中與未擊中時不同的音效。

15-7 彈藥與計分

[彈藥遞減與分數遞增的控制與顯示]

合理的射擊遊戲彈藥數量必須有限制，越打越少，打完遊戲就結束！同時也必須計算擊中次數作為分數的依據。在此遊戲中我們限制彈藥數目為 25，分數自 0 分起算。首先，必須宣告彈藥(B)與分數(S)為全域變數，預設值分別為 25 與 0。每次擊發武器之前必須先確定還有彈藥(B>0)，才能執行所有與射擊相關的程式；每射擊一次彈藥數目 B 減 1，並顯示於看板(label1)。擊中目標時則是分數 S 加 1，也要顯示於看板(label2)。KeyDown 程式繼續擴增為：

```
int B=25;
int S=0;
private void Form1_KeyDown(object  sender, KeyEventArgs  e)
{
    if (B > 0)
    {
        if (e.KeyCode == Keys.Z)
        {
            if (L.Visible == false)
            {
                B -= 1;
                label1.Text = "彈藥=" + B.ToString();
                L.Visible = true;
                timer1.Start();
                if (L.Left > T.Left && L.Right < T.Right)
                {
                    C.Audio.Play(Properties.Resources.Hit,  AudioPlayMode.Background);
                    S += 1;
                    label2.Text = "得分=" + S.ToString();
                    T.Image = Properties.Resources.Targetx;
                }
                else
                {
                   C.Audio.Play(Properties.Resources.Shoot,  AudioPlayMode.Background);
                }
            }
        }
    }
}
```

[重玩機制：彈藥與分數復原]

如果想要不重新啟動程式重玩一次的話，要作的事情應該包括彈藥重新填滿(B=25)以及分數歸零(S=0)，當然也必須顯示於看板讓使用者知道狀況，這些就是重玩按鍵(button1)的內容如下：

```
private void button1_Click(object  sender, EventArgs  e)
{
    S = 0;
    label2.Text = "得分=" + S.ToString();
    B = 25;
    label1.Text = "彈藥=" + B.ToString();
}
```

最終設計完成的遊戲過程畫面如下：

15-8 進階挑戰

一、如何製作打中會扣分的目標(打到友軍)？
提示：可製作多目標，具有不同之 Tag 值為計分依據(友軍為負數)。

二、如何製作可以選擇武器圖案的介面？
提示：可使用 RadioButton 物件選擇圖檔切換武器或目標圖案。

三、如何製作打到會消失的目標？
提示：可使用物件的 Visible 屬性，或 Dispose 方法控制。

課後閱讀

談談 C#程式碼的架構

大多數的 C 語言書籍會在更前面就介紹很多的物件導向與程式專案結構等理論概念，但本書至此應該已證明：即使你不太懂那些理論，一樣可以製作出很多有用的 C# 程式。但是當你製作的程式功能逐漸複雜龐大時，對於程式架構理論就必須有較深入的了解，才不會成為你逐步升級為專家的絆腳石。在此我們還是不準備講得非常完整深奧，只談談與目前專案有關的一些概念。

[using, namespace 與 class]

我們先看看 C#程式碼的階層架構，最前面是一連串的 using 宣告引用需要的函式庫分類，接著是一個 namespace WindowsFormsApplication1 標題，這是一個專案(或稱命名空間 namespace)的開始，理論上一個專案裡面可以容納多個 class(翻譯成「類別」)，而我們目前已經很習慣使用的表單(Form)就是一種視窗程式的基本類別。一個專案程式碼剛開始的樣貌大概如下：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

[何謂 partial？]

請注意到 **partial** 這個關鍵字，就是說這個類別(class)的程式碼在此(Form1.cs)只是**一部分**，其他部分主要就是在我們已經多次處理過的 Form1.Drsigner.cs 這個檔案了！打開這個檔案裏面的類別也是以 partial class Form1 開始的！類似這樣：

```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// 設計工具所需的變數。
        /// </summary>
        private System.ComponentModel.IContainer components = null;
```

這樣做的好處是不必把太多程式碼集中在一個檔內，增加視覺負擔，而且也可以分工，讓複雜表單的建置能分成幾個部份進行。物件屬性與事件的基本設定基本上由 Form1.Designer.cs 負責，程式邏輯部分則由 Form1.cs 負責。也就是我們在設計頁面作的事情會自動變成 Designer 的程式碼，多數必須我們主導設計的程式邏輯則放在 Form1.cs 內！

[所有控制項也都是類別(class)]

在物件導向程式概念中，類別與物件的意義相似。事實上如果我們可以看到工具箱物件的原始程式碼，應該也都類似表單程式的樣子！包括物件的屬性、方法與事件等等，都可以是類別內的成員，屬性類似類別內宣告的一個變數，方法與事件則是一些各自獨立的公用(Public)副程式。換言之，我們工作的視窗其實就是我們建立與設計的一個「類別」！類別之內可以有其他的類別，本身也可以被其他類別使用，就是這個彈性的架構讓開發複雜結構的程式更容易，也更易於讓開發團隊分工合作。

變數有效的範圍

在前面的單元我們已經使用過很多的變數宣告，如 int S;之類的語法，基本上我們可以在 public partial class Form1 : Form 之下的大括號範圍內的任何地方宣告變數，但是宣告的位置與適用範圍是有差異的！

[區域變數]

宣告變數當然是有需要使用，但如上所述，程式碼是有階層架構的，變數也是！一般來說我們宣告的變數只能在同階層，及以下的階層使用。在此階層(或者說程式單元)以上(或說以外)就不具意義。譬如在某個副程式 A 內宣告的變數「X」與另一個副程式 B 內宣告的變數 X 是不同的變數，猶如不同班級中同座號的人其實並不相同一樣！甚至如果你在一個 If 程式區塊(或者 for 迴圈)內宣告一個變數，這個變數也只能在此區塊內有效！出了程式區塊就不能視它為一個存在的變數。

[全域變數]

那麼如果我希望一個變數可以通用於多個副程式怎麼辦？就是必須將它宣告在所有副程式之外！也就是它的地位須與副程式平行，是同一個「階層」的東西！我們就稱它為全域變數了。譬如本單元的計分計時變數就是如此，我們在很多個副程式中都必須隨時參考分數與時間，此時就不能將變數宣告在個別的副程式內。至於需不需要宣告到程式最前面，對 C#來說只要在副程式之外，class 邊界之內就可以。如果各個副程式是教室，副程式之間的空間就是走廊，任何走廊上出現的宣告就都是效果相同的全域(Global)變數，也因此你可以將變數宣告在主要使用它們的副程式前面，比較容易閱讀。譬如我們幾個單元中的拖曳程式多半就是將需要共用的 mdx, mdy 等變數寫在 MouseDown 與 MouseMove 事件副程式的前面。