第 13 章　打磚塊遊戲

簡介：

打磚塊遊戲的主要功能包括：一個碰到障礙會反彈的球，一個可以移動的球拍，以及許多被球碰撞後會消失的磚塊物件。此專案中將介紹如何用程式動態的產生大量的磚塊物件，同時設計檢查物件碰撞以及反彈行為的程式。同時也替撞擊事件加入了音效，增加一些臨場感。

13-1 四面反彈的球

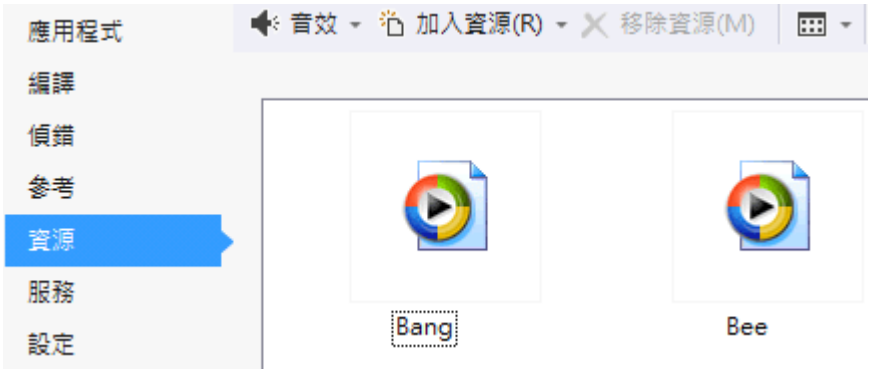
此專案前半部與上一單元乒乓球遊戲之功能大致相同，開啟專案後佈置一個 PictureBox 物件，取名為「B」，載入一個長寬皆為 32 像素點的球型圖檔，可自書附光碟的本章目錄中找到如下圖檔：

其 SizeMode 屬性必須設定為 AutoSize 讓 PictureBox 物件邊緣與圖案相符。接著佈置一個 Timer 物件，設定 Enabled 屬性為 True，雙擊該物件進入 Timer1_Tick 事件副程序，寫入如下的程式，並宣告 Vx 與 Vy 公用變數代表水平與垂直速度都等於 5，球就可以斜向移動，且碰到表單的四個邊界都會反彈了！相關程式碼解說請參考上一單元乒乓球遊戲。

```
Dim Vx As Single = 5, Vy As Single = 5 '速度值

Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    B.Left += Vx 'X方向移動
    B.Top += Vy 'Y方向移動
    If B.Left < 0 Then Vx = Math.Abs(Vx) '碰左牆
    If B.Right > Me.ClientSize.Width Then Vx = -Math.Abs(Vx) '碰右牆
    If B.Top < 0 Then Vy = Math.Abs(Vy) '碰屋頂
    If B.Bottom > Me.ClientSize.Height Then Vy = Math.Abs(Vy) '碰地板
End Sub
```

在此請注意：應該用 PictureBox 製作球與球拍物件，原因是稍後我們會用 Label 物件製作磚塊，這樣程式可以根據物件是 Label 或 PictureBox 區分是否為磚塊物件。稍後的程式也需要加入音效，因此請模仿打地鼠單元的方式加入書附光碟中的兩個音效檔案 Bang.wav 與 Bee.wav 為專案資源，如下圖：



13-2、球拍製作

佈置一個 PictureBox 物件，取名為「P」，寬與高設為 100 與 20 點， BackColor 屬性設為黑色，讓它看起來像個堅硬的擊球板。我們的目標是讓它可以在程式執行中被滑鼠拖曳，碰到球時可以讓球如碰壁一樣的反彈。拖曳此球拍的程式須寫在 P 物件的 MouseDown 與 MouseMove 事件中，且限定只能左右拖曳，並加入防止物件脫離畫面的機制，相關程式碼解說請參考上一單元乒球遊戲之說明。程式碼如下：

```
'拖曳球拍的程式
Dim mdx As Integer '拖曳起點
Private Sub P_MouseDown(sender As Object, e As MouseEventArgs) Handles P.MouseDown
    mdx = e.X '拖曳起點
End Sub
'拖曳中
Private Sub P_MouseMove(sender As Object, e As MouseEventArgs) Handles P.MouseMove
    If e.Button = Windows.Forms.MouseButtons.Left Then
        Dim X As Integer = P.Left + (e.X - mdx) '試算拖曳位置
        If X < 0 Then X = 0 '左移極限控制
        If X > Me.ClientSize.Width - P.Width Then
            X = Me.ClientSize.Width - P.Width '右移極限控制
        End If
        P.Left = X '球拍位置(不超出邊界)
    End If
End Sub
```

球拍擊球的相關程式事實上是取代之前的下邊界反彈，限定只有在球拍範圍之內才有反彈行為，否則球會直接掉落到畫面之下，遊戲也就結束了！所以必須在控制球運動的 Timer1 事件中寫出判斷是否碰到球拍，以及是否掉出邊界結束遊戲的程式。在此也同時加入撞擊球拍不同位置時可以改變水平速度(Vx)的程式，完整程式碼如下：

```
'球的運動控制
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    B.Left += Vx 'X方向移動
    B.Top += Vy 'Y方向移動
    If B.Left < 0 Then Vx = Math.Abs(Vx) '碰左牆
    If B.Right > Me.ClientSize.Width Then Vx = -Math.Abs(Vx) '碰右牆
    If B.Top < 0 Then Vy = Math.Abs(Vy) '碰屋頂
    Dim C As Single = (B.Left + B.Right) / 2 '球的中心點X座標
    If B.Bottom > P.Top And C > P.Left And C < P.Right Then '撞到球拍
        My.Computer.Audio.Play(My.Resources.Bang, AudioPlayMode.Background) '擊球音效
        Vy = -Math.Abs(Vy) '向上彈
```

```
            Dim F As Single = (C - P.Left) / P.Width '計算擊球點
            If Vx < 0 Then F = 1 - F '方向調整
            Vx = Vx * (F + 0.5) 'X速度修正
        End If
        If B.Top > Me.ClientSize.Height Then '漏接了，球掉出畫面
            Timer1.Stop()
            MsgBox("Game over!")
        End If
    End Sub    End Sub
```

程式說明詳見上一單元之乒乓球遊戲，此處並在球打到球拍時發出一個 Bang 音效。

13-3、磚塊的製作：動態產生控制項

通常這類遊戲的磚塊(目標)數目都很多，如果要在設計階段直接擺好上百個磚塊，還必須排列整齊是非常耗時費事的作法。還好表單上的物件，或稱控制項(Control)如同程式內的變數，也可以在程式進行中用程式碼的宣告方式產生，這種作法我們稱之為「**動態產生控制項**」。

只是不同於一般變數宣告，新控制項的產生必須加入「New」關鍵字。其差異在於：如果你只是宣告一個物件型別，譬如 Dim Q As Lable，程式會知道 Q 是一個 Label 類型的物件，實際呼叫使用它時卻會碰到「並未將物件參考設定為物件的執行個體」之類的錯誤訊息！這表示該物件只有登錄的名稱，並未真的在記憶體中被建立為實體(instance)，所以它是無法被實際操作的！這很像某些父母替尚未出生的孩子先命名，但實際上世間還沒有真的這個人，你也不能為這個「名字」報戶口一樣！

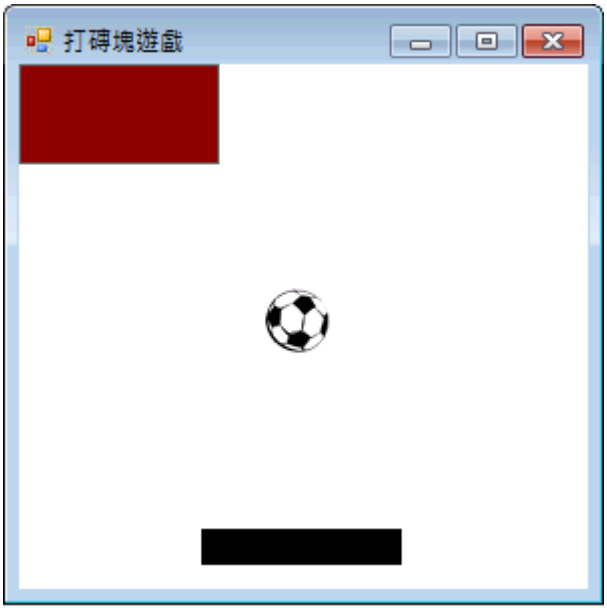
通常我們會用到動態產生控制項的原因，一是數量很大，一是不確定物件的數量與位置。所以設計程式時必須保持彈性，最好將產生物件的程式碼變成一個自訂的獨立副程序。此副程序實際的操作程序是將我們希望製作的控制項屬性以參數形式傳遞到副程序內，副程序依此「訂單」的要求製作出實體物件並加入表單，我們就有一個新物件了！

我們先建立如下的製造磚塊的副程序，並在 Form_Load 事件中試用一下，在表單的左上角座標(0,0)處產生一個磚塊，程式碼與執行畫面如下：

```
'製作指定位置磚塊的副程序
Sub Brick(X As Integer, Y As Integer)
    Dim Q As New Label '建置新的磚塊物件
    With Q
        .Width = 100 '寬
        .Height = 50 '高
        .BackColor = Color.DarkRed '磚紅色
        .BorderStyle = BorderStyle.FixedSingle '邊框
        .Left = X '座標X
        .Top = Y '座標Y
    End With
    Me.Controls.Add(Q) '磚塊加入表單
End Sub

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Brick(0, 0) '左上角的一個磚塊
```

```
End Sub
```



上述程式碼首先建立一個名為 Brick 的副程序，副程序內部就是定義磚塊應有的各項屬性，其中只有磚塊位置是經由 X 與 Y 參數自副程序外部傳進來的。當然我們也可以使用 PictureBox 之類的其他物件作為磚塊，但是考慮到記憶體的使用量，應該盡量使用如 Label 這樣功能少，記憶體資源消耗也少的輕量級物件來製作大量的物件。

Brick 副程序先用關鍵字 New 宣告並建置一個稱為 Q 的 Label 物件，接著使用 With…End With 的語法連續宣告它的多個屬性，在此直接寫成 Q.Left，Q.Top…等等也無不可，但使用 With 語法結構較為清晰。這些屬性中除了位置(X，Y)由外部傳入之外，其他都是副程序自定的，所以產品外觀都一樣。至於 Me.Controls.Add(Q)的語法是將 Q 物件加入表單(Me)的控制項集合(Controls)之內，這樣才能在表單上顯示出來，如同我們在設計階段將工具箱物件佈置到表單上一樣。

13-4、磚牆的製作與起始畫面調整

接下來我們要建置遊戲需要的所有磚塊，目標是在視窗的上方做出 5X5 共 25 個磚塊的磚牆。這可以在 Form_Load 事件中用兩層的巢狀迴圈控制 X 與 Y 座標，再重複呼叫 Brick 副程序即可，程式碼如下：

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    For i As Integer = 0 To 4 '五個橫列
        For j As Integer = 0 To 4 '五個直行
            Brick(100 * i, 50 * j) '製作磚塊(100x50點大小)
        Next
    Next
End Sub
```

上述迴圈的內層改變 Y 座標(j)，所以會在固定的 X 位置產生一排垂直排列的磚塊，外層迴圈改變 X 座標，在不同的 X 位置再產生另一排磚塊，掌握好磚塊的寬(100)與高(50)就會變成整齊砌好的磚牆了。

但是因為我們事前並未刻意調整視窗大小以符合這面磚牆，所以上述程式產生的磚牆可能會超出視窗範圍，或只佔據視窗的左上角；球與球拍也可能身陷在磚塊之中，使畫面變得不合理。在設計階段要先計算精準這些動態產生物件的關係位置也是可以，但是挺麻煩的！因此可以使用動態調整視窗及既有物件的技巧，以程式計算方式調整畫面。首先我們要使視窗的可編輯區域切齊磚牆的寬度，程式碼如下：

```
Me.Width = (Me.Width - Me.ClientSize.Width) + 100 * 5
```

其中的 Me 代表視窗，Me.Width 代表視窗外框寬度，Me.**ClientSize**.Width 代表的是視窗**內部**可編輯區的寬度，兩者相減就是視窗左右邊框的寬度總和。我們的目標是讓視窗的可編輯區切齊磚牆的寬度，因為每塊磚寬 100 乘以 5 等於 500，磚牆總寬度為 500，Me.ClientSize.Width 也應該等於 600 點寬，可惜這個 Me.ClientSize.Width 屬性是唯讀的，無法直接設定！所以我們只能改變視窗外框的寬度(Me.Width)來間接調整它，就是使得 Me.Width 等於磚牆寬度加上邊框的寬度。

同理，高度的處理也類似，但是必須加上可以容納球與球拍的高度空間(200 點)。程式碼如下：

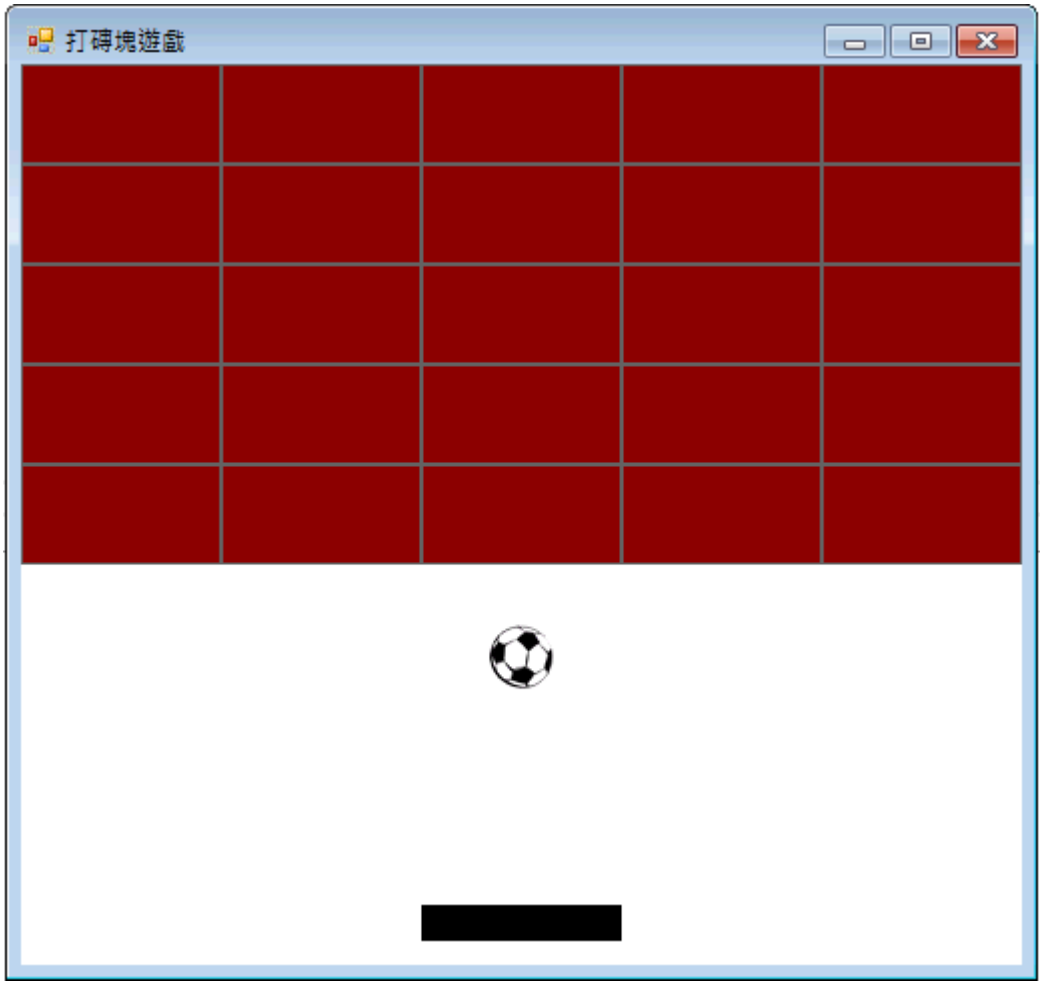
```
Me.Height = (Me.Height - Me.ClientSize.Height) + 50 * 5 + 200
```

接下來是球與球拍的位置調整程式：

```
B.Top = 50 * 5 + 30 '調整球的高度，五排磚塊下30點
B.Left = (Me.ClientSize.Width - B.Width) / 2 '球水平置中
P.Top = Me.ClientSize.Height - 30 '調整球拍的高度，離底30點
P.Left = (Me.ClientSize.Width - P.Width) / 2 '球拍水平置中
```

球(B)的高度是磚牆高度加上 30，較接近磚牆的位置；拍子(P)的高度是可編輯區高度減 30，較接近視窗的底部。水平位置都應該置中，計算式是可編輯區的寬度減去物件寬度除以 2！完整的 Form_Load 程式碼與執行程式畫面如下：

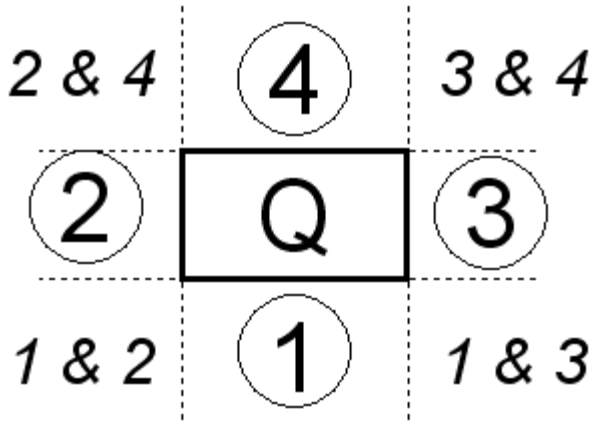
```
'表單載入
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    For i As Integer = 0 To 4 '五個橫列
        For j As Integer = 0 To 4 '五個直行
            Brick(100 * i, 50 * j) '製作磚塊(100x50點大小)
        Next
    Next
    Me.Width = (Me.Width - Me.ClientSize.Width) + 100 * 5 '調整表單寬度
    Me.Height = (Me.Height - Me.ClientSize.Height) + 50 * 5 + 200 '調整表單高度
    B.Top = 50 * 5 + 30 '調整球的高度，五排磚塊下30點
    B.Left = (Me.ClientSize.Width - B.Width) / 2 '球水平置中
    P.Top = Me.ClientSize.Height - 30 '調整球拍的高度
    P.Left = (Me.ClientSize.Width - P.Width) / 2 '球拍水平置中
    Timer1.Start() '啟動球的運動
End Sub
```



13-5、磚塊碰撞偵測

到此為止程式執行時球會在磚牆上面劃過，但是不會對磚塊有任何作用或反應，我們的目標是讓球與磚塊碰撞時會產生合理的反彈，同時磚塊也要消失！以寫程式的角度來說：如何檢查出球與磚塊是否產生碰撞？而且是碰到了磚塊的上下左右哪一邊？最為

困難，也就是所謂的「碰撞偵測」程式。我們的數學邏輯可以從下面的示意圖講起：



中央的 Q 代表磚塊，四面的圓圈代表球。如果如球②的右緣(Right 屬性)小於磚塊的左緣(Left 屬性)，代表球確定偏左沒有發生碰撞；其餘方向皆可依此類推，就可以排除所有確定「無」碰撞的狀況，通過這些測試之後就是確定「有」碰撞了！事實上就是球與磚塊確定有部分重疊。我們可以開始建立一個檢查球與某個磚塊是否有碰撞的自訂副程序 chkHit，初步程式碼如下：

```
'檢查球與磚塊或牆壁碰撞的程式
Function chkHit(Q As Label) As Boolean
    If B.Right < Q.Left Then Return False '偏左未碰到
    If B.Left > Q.Right Then Return False '偏右未碰到
    If B.Top > Q.Bottom Then Return False '偏下未碰到
    If B.Bottom < Q.Top Then Return False '偏上未碰到
    Return True '回傳有碰撞
End Function
```

其中帶入的參數 Q 就是要檢查有無與球 B 碰撞的物件，沒碰到就回傳 False，四個偏差狀況都沒發生時，就回傳 True。但是在此我們不只需要知道碰到了沒？還必須知道它們是碰到上下左右哪一個邊界？這樣才能正確地讓球往正確的方向反彈。譬如撞到磚塊左邊就應該往左彈，右邊就往右彈等等。

表面上好像可以用上面四個判斷式的反向判斷得知碰撞的方向，譬如 B.Right < Q.Left 表示確定未碰到左緣，是不是 B.Right >= Q.Left 就代表撞到左邊緣了呢？其實不然！譬如下圖的狀況比較可能是撞到右邊，但是也符合 B.Right >= Q.Left 的條件：

此時可以加入一個條件就是球的前一個位置，也就是減去一個 Vx 或 Vy 的位置是「不」符合該條件的，但是現在卻符合了！表示剛剛才碰到磚塊的這一面。譬如左側碰撞的條件可以是：

B.Right >= Q.Left And **(B.Right - Q.Left) <= Math.Abs(Vx)**

其中粗體字部分就是限制 B.Right - Q.Left 的差值小於等於一個 Vx，表示兩者重疊部分小於一個速度值，那麼回溯上一個位置時應該是未碰撞的。依此類推，我們就可以偵測出球是碰到磚塊的哪一面，然後給他們正確的反彈方向了！完整的 chkHit 副程序變成這樣：

```
'檢查球與磚塊或牆壁碰撞的程式
Function chkHit(Q As Label) As Boolean
    If B.Right < Q.Left Then Return False '偏左未碰到
    If B.Left > Q.Right Then Return False '偏右未碰到
    If B.Top > Q.Bottom Then Return False '偏下未碰到
    If B.Bottom < Q.Top Then Return False '偏上未碰到
    '碰撞目標左側(剛剛越過左邊界)往左彈
    If B.Right >= Q.Left And (B.Right - Q.Left) <= Math.Abs(Vx) Then Vx = -Math.Abs(Vx)
    '碰撞目標右側(剛剛越過右邊界)往右彈
    If B.Left <= Q.Right And (Q.Right - B.Left) <= Math.Abs(Vx) Then Vx = Math.Abs(Vx)
    '碰撞目標底部(剛剛越過底邊界)往下彈
    If B.Top <= Q.Bottom And (Q.Bottom - B.Top) <= Math.Abs(Vy) Then Vy = Math.Abs(Vy)
    '碰撞目標頂部(剛剛越過頂邊界)往上彈
    If B.Bottom >= Q.Top And (B.Bottom - Q.Top) <= Math.Abs(Vy) Then Vy = -Math.Abs(Vy)
    Q.Dispose() '刪除磚塊物件
    Return True '回傳有碰撞
End Function
```

如上程式註解，四個反彈面偵測到之後就設定應有的速度方向反應，球就會合理的反彈了！因為我們希望磚塊被打到是會消失的，所以處理完球的速度改變之後就將磚塊物件刪除(Dispose)，並回傳一個 True，表示撞到了！

13-6、使用碰撞偵測程式

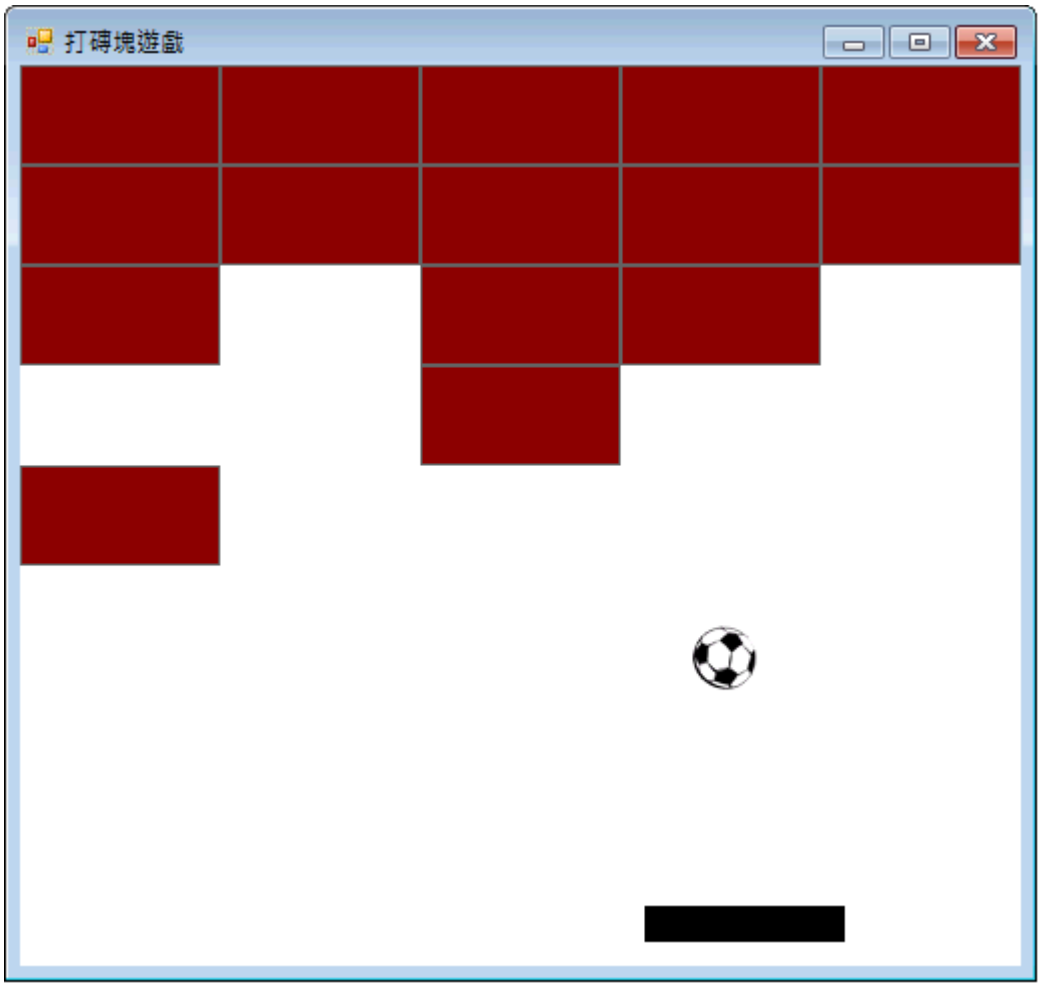
至此，其實我們都還無法測試 chkHit 是否正常運作，必須在 Timer1_Tick 事件之末加上執行所有磚塊碰撞檢查的程式碼：

```
'檢查磚塊碰撞情況
For Each q In Me.Controls '每一個控制項
    If TypeOf (q) Is Label Then '如果是Label(磚塊)
        If chkHit(q) Then '檢查是否擊中磚塊
            My.Computer.Audio.Play(My.Resources.Bee, AudioPlayMode.Background) '破磚音效
        End If
    End If
Next
```

這是一個特殊的迴圈，For Each q In Me.Controls 的意義是依序呼叫每一個表單上的控制項，並暫定其名稱為 q 的意思。接下來用判斷式 If TypeOf (q) Is Label 來判斷此控制項的型態是否為標籤(Label)？是的話就應該是磚塊，必須執行 chkHit 作碰撞偵測。如果有碰到磚塊就發出破磚的音效 Bee。

最後必須強調，因為程式畢竟是跳躍式的移動，與類比的世界中連續運動的物理現象還是可能有差異，不合理的情況還是不時會出現，要減少這種問題合理的方向是將 Timer 的 Interval 屬性減少(執行速度變快)，一次移動的距離(Vx 與 Vy)也減少，代價則

是電腦資源使用較多。最後看看可能的遊戲過程畫面吧！



13-7、進階挑戰

一、如何加入遊戲啟動、暫停或變速等控制介面？
　　提示：建立主功能表加入各項功能程式，可參考乒乓球遊戲單元。

二、如何加入計分機制？
　　提示：宣告公用變數記錄積分，於磚塊擊破時加分。

三、如何建立不同外觀的磚塊，且區隔計分？
　　提示：修改 Brick 副程序的內容，並將不同的預期計分寫入 Tag 屬性。

專案設計頁面與程式碼

Form1：


```
Public Class Form1
    '表單載入
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        For i As Integer = 0 To 4 '五個橫列
            For j As Integer = 0 To 4 '五個直行
                Brick(100 * i, 50 * j) '製作磚塊(100x50點大小)
            Next
        Next
        Me.Width = (Me.Width - Me.ClientSize.Width) + 100 * 5 '調整表單寬度
        Me.Height = (Me.Height - Me.ClientSize.Height) + 50 * 5 + 200 '調整表單高度
        B.Top = 50 * 5 + 30 '調整球的高度，五排磚塊下30點
        B.Left = (Me.ClientSize.Width - B.Width) / 2 '球水平置中
        P.Top = Me.ClientSize.Height - 30 '調整球拍的高度
        P.Left = (Me.ClientSize.Width - P.Width) / 2 '球拍水平置中
        Timer1.Start() '啟動球的運動
    End Sub
    '製作指定位置磚塊的副程序
    Sub Brick(X As Integer, Y As Integer)
        Dim Q As New Label '建置新的磚塊物件
        With Q
            .Width = 100 '寬
            .Height = 50 '高
            .BackColor = Color.DarkRed '磚紅色
            .BorderStyle = BorderStyle.FixedSingle '邊框
            .Left = X '座標X
            .Top = Y '座標Y
        End With
        Me.Controls.Add(Q) '磚塊加入表單
    End Sub
    Dim Vx As Single = 5, Vy As Single = 5 '速度值
```

```
'球的運動控制
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    B.Left += Vx 'X方向移動
    B.Top += Vy 'Y方向移動
    If B.Left < 0 Then Vx = Math.Abs(Vx) '碰左牆
    If B.Right > Me.ClientSize.Width Then Vx = -Math.Abs(Vx) '碰右牆
    If B.Top < 0 Then Vy = Math.Abs(Vy) '碰屋頂
    Dim C As Single = (B.Left + B.Right) / 2 '球的中心點X座標
    If B.Bottom > P.Top And C > P.Left And C < P.Right Then '撞到球拍
        My.Computer.Audio.Play(My.Resources.Bang, AudioPlayMode.Background) '擊球音效
        Vy = -Math.Abs(Vy) '向上彈
        Dim F As Single = (C - P.Left) / P.Width '計算擊球點
        If Vx < 0 Then F = 1 - F '方向調整
        Vx = Vx * (F + 0.5) 'X速度修正
    End If
    If B.Top > Me.ClientSize.Height Then '漏接了，球掉出畫面
        Timer1.Stop()
        MsgBox("Game over!")
    End If
    '檢查磚塊碰撞情況
    For Each q In Me.Controls '每一個控制項
        If TypeOf (q) Is Label Then '如果是Label(磚塊)
            If chkHit(q) Then '檢查是否擊中磚塊
                My.Computer.Audio.Play(My.Resources.Bee, AudioPlayMode.Background) '破磚音效
            End If
        End If
    Next
End Sub
'檢查球與磚塊或牆壁碰撞的程式
Function chkHit(Q As Label) As Boolean
    If B.Right < Q.Left Then Return False '偏左未碰到
    If B.Left > Q.Right Then Return False '偏右未碰到
    If B.Top > Q.Bottom Then Return False '偏下未碰到
    If B.Bottom < Q.Top Then Return False '偏上未碰到
    '碰撞目標左側(剛剛越過左邊界)往左彈
    If B.Right >= Q.Left And (B.Right - Q.Left) <= Math.Abs(Vx) Then Vx = -Math.Abs(Vx)
    '碰撞目標右側(剛剛越過右邊界)往右彈
    If B.Left <= Q.Right And (Q.Right - B.Left) <= Math.Abs(Vx) Then Vx = Math.Abs(Vx)
    '碰撞目標底部(剛剛越過底邊界)往下彈
    If B.Top <= Q.Bottom And (Q.Bottom - B.Top) <= Math.Abs(Vy) Then Vy = Math.Abs(Vy)
    '碰撞目標頂部(剛剛越過頂邊界)往上彈
    If B.Bottom >= Q.Top And (B.Bottom - Q.Top) <= Math.Abs(Vy) Then Vy = -Math.Abs(Vy)
    Q.Dispose() '刪除磚塊物件
    Return True '回傳有碰撞
End Function
'拖曳球拍的程式
Dim mdx As Integer '拖曳起點
Private Sub P_MouseDown(sender As Object, e As MouseEventArgs) Handles P.MouseDown
    mdx = e.X '拖曳起點
End Sub
'拖曳中
Private Sub P_MouseMove(sender As Object, e As MouseEventArgs) Handles P.MouseMove
    If e.Button = Windows.Forms.MouseButtons.Left Then
        Dim X As Integer = P.Left + (e.X - mdx) '試算拖曳位置
        If X < 0 Then X = 0 '左移極限控制
        If X > Me.ClientSize.Width - P.Width Then
            X = Me.ClientSize.Width - P.Width '右移極限控制
        End If
```

```
            P.Left = X '球拍位置(不超出邊界)
        End If
    End Sub
End Class
```

課後閱讀

一、談談動態產生物件

本單元最有趣的新技術應該是使用程式碼製造大量磚塊的程式，比較專業的術語叫作「動態產生物件」。因為我們已經習慣在設計階段自工具箱拉出物件，而在程式碼中宣告數值或文字變數，這讓我們有個錯誤的印象認為這是截然不同的兩件事情！變數與物件是「完全不同的東西」。但事實上所有的物件也都可以如一般變數一樣直接用程式碼宣告產生，只是這些「物件」通常比文數字資料結構大而複雜，要正確的讓它們成為表單上可以正常運作的一份子，步驟會多一點。

第一個差別是文數字等等變數我們稱為「基本資料型態」，使用如 Dim i As Integer 這樣的宣告之後這個變數就是實際可以使用的「東西」，你可以定義它是等於 3 還可以將它與其他數字相加等等。也就是電腦已經配置了記憶體位置給這個變數 i，你讓 i=3 的時候記憶體的這個位置的值就變成 3 了！

但是對於非基本資料型態的東西，如果只用 Dim Q As Label 這樣的宣告，只是告訴電腦它的資料類型，就是電腦在紀錄本上記載「名稱 Q 的東西是個 Label」，但事實上電腦並不會因此在記憶體中給它一個實際的位置，當你繼續當 Q 是個實體物件操作時不是出現錯誤訊息就是物件根本看不到，因為它實際上不存在(只有一個空名牌)！關鍵是要使用 New 來建置物件「實體」，因此動態產生物件的第一關是必須用 Dim Q As New Label 這種語法宣告新物件。

接著我們必須用程式碼去定義原本我們習慣用屬性視窗定義的一些屬性，包括物件的大小、顏色與位置等等！但是即使你設定好了屬性完整的物件，它也只是深藏在記憶體中，必須將它「加入表單」才會變成可見的物件。程式碼就是：Me.Controls.Add(Q)！

一、談談碰撞偵測程式

碰撞偵測在遊戲程式中可以說是非常重要的技術，但是極少有書籍會認真地介紹怎麼寫這種程式。其實任何可以打打殺殺的遊戲都一定有它的存在，所以遊戲公司的程式師是一定會寫的！只是不願意告訴我們這些外行人而已。本書的碰撞程式也是作者教學多年中所嘗試的多個版本之一。不是最好，只是比較簡單容易理解的一種而已，實際寫法應該不下幾十種，讀者可以自己試試新的邏輯推理方式。

因為碰撞是一個蠻複雜的數學問題，老實說即使會寫程式的高手要寫出完全合理的碰撞程式都很困難。即使你的程式可以交代所有的碰撞行為，如何節省運算還是一個大問題。想一想，每次球移動一小步就必須檢查它和所有的磚塊有無碰撞？這當然是很消耗電腦資源的！因此實務上應該還必須加上一些快速檢查的機制，不會每一個磚塊每次都做完整的碰撞偵測才對。