第 11 章乒乓球遊戲

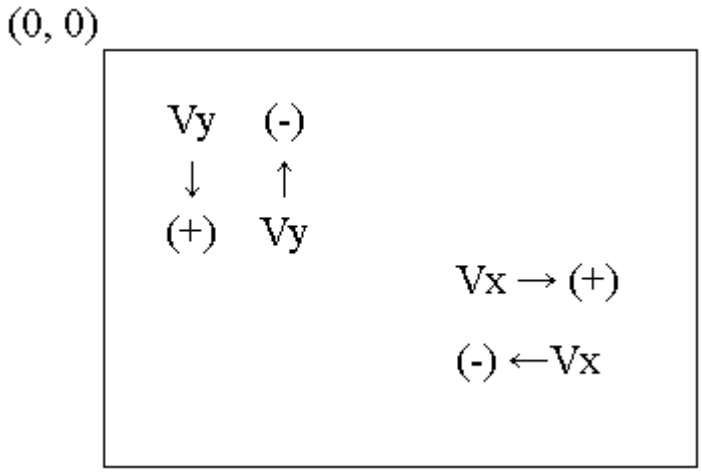
簡介：

本單元將製作一個如乒乓球一般會在表單範圍內移動，碰到邊框及擊球板都可以合理反彈的遊戲程式。主要的程式技術是如何控制球的移動、碰撞事件的處理，以及限制物件拖曳範圍的程式。

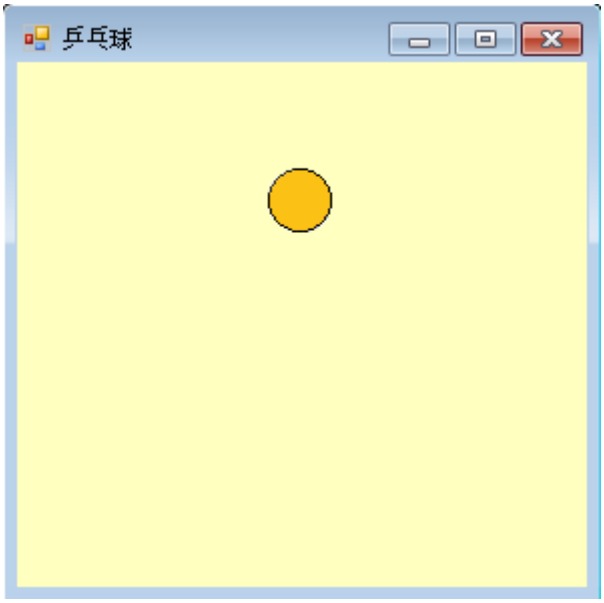
11-1 移動的球

[電腦繪圖的座標系統]

要讓物件在程式內移動基本上就是要以計時器定時改變它的座標，水平向以 Left 屬性代表，垂直向以 Top 屬性代表，改變這兩個屬性物體就移動了！每次移動的距離可以視為移動的速度，我們以 Vx 與 Vy 代表水平與垂直方向的速度，簡單的示意圖如下：



如上圖，一般的電腦繪圖座標原點在左上方，Y 軸以下為正，其餘與一般數學座標相同。Vx 為正，代表往右移動，為負代表往左；Vy 為正，代表往下，為負代表往上。我們先建立一個表單上的球形物件，請使用 PictureBox 製作，在它的 Image 屬性直接載入球的圖檔即可，但是為了讓程式簡潔起見，請將物件名稱改為「B」。如下圖：



[等速移動物件的程式]

接下來請叫用一個 timer1 物件，保持預設的屬性，寫程式如下：

```
int Vx = 5;
int Vy = 5;
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    B.Left += Vx;
    B.Top += Vy;
}
```

程式首先宣告 Vx 與 Vy 為全域變數，數值皆為正 5。接著在 timer1 事件中每次時間到(預設為 0.1 秒)就移動一個 Vx 與 Vy 值，因為都是正值，球應該會往右下方移動。最後必須在 Form_Load 事件中啟動(Start)這個 timer1。這樣就可以讓球移動了！但是球很快就會跑出視窗畫面，我們還需要繼續寫程式讓它有反彈的行為。

11-2 四面反彈的球

[碰到左邊緣時的程式]

接下來我們要讓球碰壁時會作合理的反彈。第一個問題是要判斷球在何種狀態時表示「碰壁」了？第二個問題是「反彈」的意義是甚麼？我們先分析球碰左邊牆壁的過程，應該是：球往左移動(Vx 為負)直到球(B)的「左緣」(B.Left)小於視窗的左邊界(座標 x=0)，此時的反彈應該是讓 Vx 變成正數(往右)，程式碼應該像這樣：

```
if (B.Left < 0)
{
Vx = Math.Abs(Vx);
}
```

其中 Math.Abs 函數是數學上取絕對值的意思。相對的，碰到右邊界的過程應該是球往右邊走，直到球的右邊緣(B.Right)座標大於視窗可活動區的寬度(this.ClientSize.Width)，此時就要強迫 Vx 變成負數，表示之後就要往左移動了！程式碼如下：

```
if (B.Right > this.ClientSize.Width)
{
Vx = -Math.Abs(Vx);
}
```

[四面碰撞的程式]

仿照以上邏輯，我們就可以完成上(Top)、下(Bottom)、左(Left)與右(Right)四個邊界的反彈程式，並寫在 timer1 事件中物體移動的動作之後，作為檢查碰撞之用，如此我們的球應該可以合理的四面反彈了！

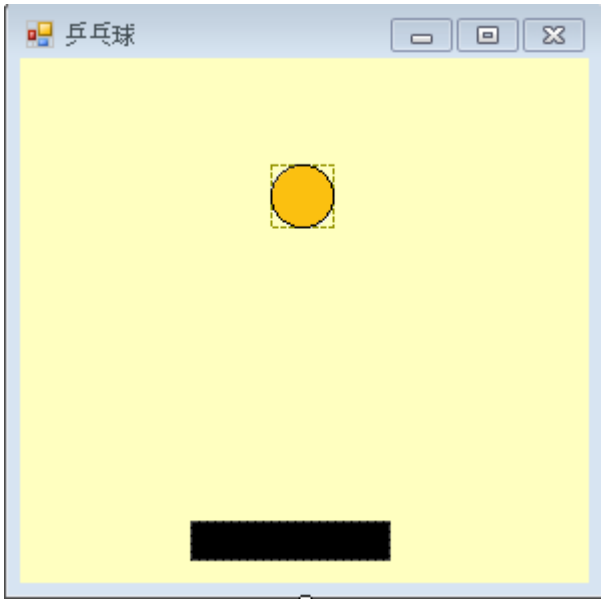
```
private void timer1_Tick(object sender, EventArgs e)
{
    B.Left += Vx;
    B.Top += Vy;
    if (B.Left < 0) { Vx = Math.Abs(Vx); } //左邊界碰撞
    if (B.Right > this.ClientSize.Width) { Vx = -Math.Abs(Vx); } //右邊界碰撞
    if (B.Top < 0) { Vy = Math.Abs(Vy); } //視窗頂部碰撞
    if (B.Bottom > this.ClientSize.Height) { Vy = -Math.Abs(Vy); } //視窗底部碰撞
}
```

其中 Right 與 Bottom 等相關屬性的詳細說明請見秀圖軟體單元 6-5 節。為了不讓程式佔據太多版面，我們也在此將一些程式指令縮減到同一行內，如前面單元談過，C#的指令是依據分號決定中止點，不是以是否斷行來區別，這樣寫是毫無問題的！

11-3 限制拖曳範圍的擊球板

[製作擊球板]

接下來我們要製作一個在視窗下方，可以被滑鼠左右拖曳的擊球板物件。此物件可以用一個 PictureBox 設定其背景為黑色，拖曳邊緣改變其位置與形狀即可完成。當然使用標籤(Label)物件設計也是可以的，方法是將 Label 的 AutoSize 屬性改為 False，刪除文字(Text)，設定背景為黑色，調整其大小與位置即可。為了程式碼的簡化請將此物件改名為 P，畫面如下：



[拖曳擊球板程式]

拖曳物件的程式我們在記事本及小畫家兩個單元都曾作過，必須使用到滑鼠壓下(MouseDown)與滑鼠移動(MouseMove)兩個副程式，還必須使用兩個全域變數，來記錄拖曳的起點座標 X 與 Y。在此因為我們希望擊球板只能左右移動，所以將 Y 方向的動作刪除。程式碼如下：

```
int mdx;
private void P_MouseDown(object sender, MouseEventArgs e)
{
    mdx = e.X;
}

private void P_MouseMove(object sender, MouseEventArgs e)
{
    if(e.Button == MouseButtons.Left)
    {
        P.Left += e.X - mdx;
    }
}
```

[限制擊球板拖曳範圍]

上述程式有個缺點，就是滑鼠左右拖曳時有可能將物件拖出畫面之外，如果玩家也同時放開滑鼠左鍵停止拖曳了，既然擊球板都看不見了，自然之後就很難正確的操作將它拖回畫面之內！因此應該用程式限制，使板子永遠不能離開視窗範圍。具體作法是每次拖曳動作都先試算落點，如果落點超出範圍就用程式強制物件只能停在邊緣。修改程式如下：

```
private void P_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        int x = P.Left + (e.X - mdx);
        if (x < 0) { x = 0; } //不超過左邊緣
        if (x > this.ClientSize.Width  - P.Width)
        { x = this.ClientSize.Width  - P.Width; } //不超過右邊緣
        P.Left = x;
    }
}
```

程式碼中的變數 x 就是擊球板左緣的試算值，如果小於 0 就強迫等於 0，板子的左緣會停在左邊界；稍稍複雜的是板子的右緣數學表示式必須是 x+P.Width，如果這個值大於視窗可活動區寬度(this.ClientSize.Width)，則視窗的右緣應該等於視窗可活動區寬度，數學表示式是：

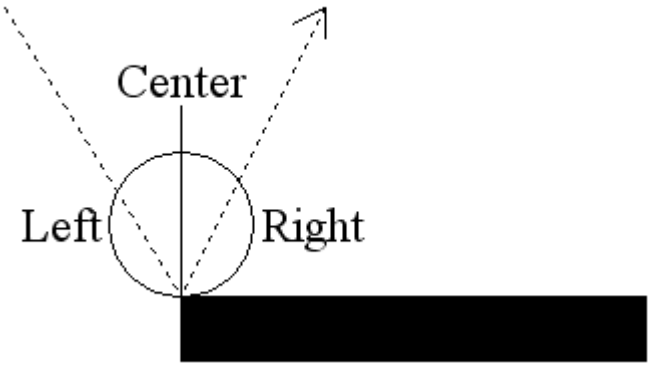
x + P.Width = this.ClientSize.Width

移項之後就是程式碼的 x = this.ClientSize.Width - P.Width 了！最後這個處理過的 x 作為 P 物件的新位置(Left)，就不會有擊球板跑出畫面的問題了！或許有人在奇怪為何不用 Right 屬性呢？原因是 Right 是不能變更的『唯讀』屬性，上面的程式也是不得已的作法。

11-4 擊球程式

[擊球板擊球反彈示意圖]

現階段的程式球遇到板子是沒反應的！要讓板子可以擊球反彈，事實上是要讓板子取代之前的視窗下方反彈的程式碼。但是必須加上限制球在板子的寬度範圍之內才能反彈，否則球就應該掉出畫面之外，死掉了！更細緻地說，球的「中央線」(Left 與 Right 屬性的平均值)必須大於板子的左邊緣(Left 屬性)座標，且小於板子的右邊緣座標(Right 屬性)；同時球的底部 Y 座標(Bottom 屬性)要小於板子的頂部 Y 座標(Top 屬性)。示意圖如下：

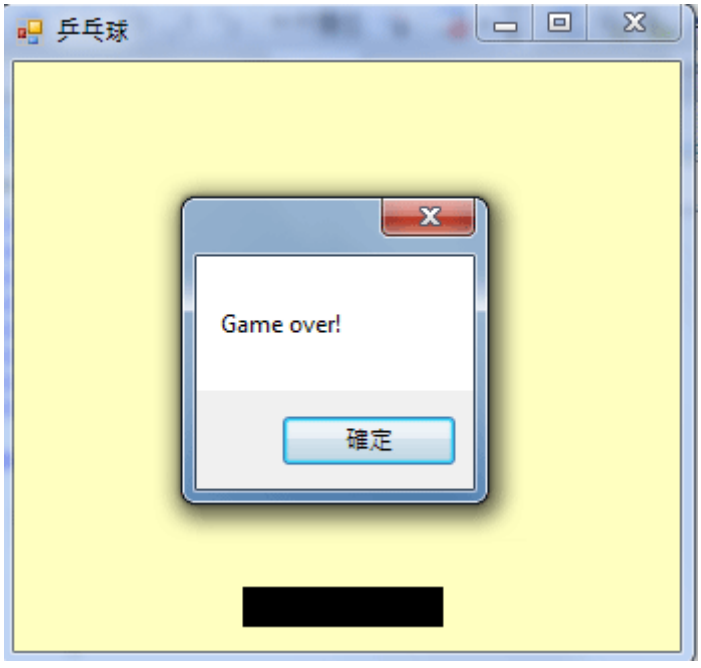


[擊球板接球與漏接的程式]

同時間，因為球有可能掉出畫面，我們也要加上遊戲停止的檢查程式，必須將 timer1 程式修改如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
    B.Left += Vx;
    B.Top += Vy;
    if (B.Left < 0) { Vx = Math.Abs(Vx); } //左邊界碰撞
    if (B.Right > this.ClientSize.Width) { Vx = -Math.Abs(Vx); } //右邊界碰撞
    if (B.Top < 0) { Vy = Math.Abs(Vy); } //視窗頂部碰撞
    if (B.Bottom > P.Top) //低於擊球板高度時(可能碰撞板子)
    {
        int C = (B.Left + B.Right) / 2; //計算球中心點
        if (C >= P.Left && C <= P.Right) //擊中擊球板
        {
            Vy = -Math.Abs(Vy); //反彈
        }
        else
        {
            if (B.Top > this.ClientSize.Height) //掉出視窗外
            {
                timer1.Stop();
                MessageBox.Show("Game over!");
            }
        }
    }
}
```

上述程式在球的底部座標(B.Bottom)大於板子的頂部(P.Top)時開始進一步檢查球的中央線座標 C = (B.Left + B.Right) / 2 是不是在板子的 X 範圍內 C >= P.Left && C <= P.Right，如果是！就是碰到板子需要反彈，否則就看球的頂部是不是已經跑出畫面之外(B.Top > this.ClientSize.Height)，如果是！就是死掉了！結束畫面大致如下：

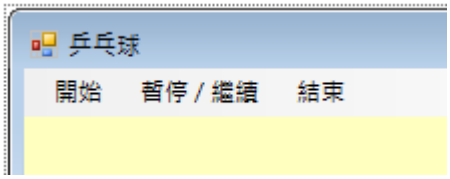


11-5 遊戲控制程式

[插入功能表]

接下來我們要使用功能表控制遊戲的「開始」、「暫停」以及「結束」。請先加入一個

主功能表(menuStrip1)，並寫入控制項目如下：



[重啟遊戲的功能]

程式開始應該啟動計時器，但是啟動遊戲也可能是重新開始，球在前面一輪遊戲中可能已經掉出畫面之外，所以必須重設球的位置，程式碼可以設計如下：就是將球的 X 與 Y 座標概略放在畫面中央偏高一點，再啟動 timer1。

```
private void 開始ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    B.Left = this.ClientSize.Width  / 2;
    B.Top = this.ClientSize.Height  / 3;
    timer1.Start();
}
```

[暫停與繼續的切換]

「暫停與繼續」是個切換開關，如果程式進行中就必須關閉 timer1 (Enabled=False)；相對的，如果是暫停狀態就必須將 timer1 啟動(Enabled=True)。一個簡單的設計方式是使用"！"運算子將 Enabled 屬性的 Boolean 值(True or False)變成相反！結束程式則使用標準的 Application.Exit()離開程式。這兩個控制項目程式如下：

```
private void 暫停繼續ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    timer1.Enabled = !(timer1.Enabled);
}

private void 結束ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    Application.Exit();
}
```

[調整乒乓球可活動區]

執行程式後一定會發現有點奇怪，球會跑到功能表(menuStrip1)的上面！這是因為之前的頂部反彈程式設定『頂部』在表單可用區的最頂端，Y 座標為 0 處，但是現在多了一個 menuStrip1 所以必須調整頂部反彈程式成為：

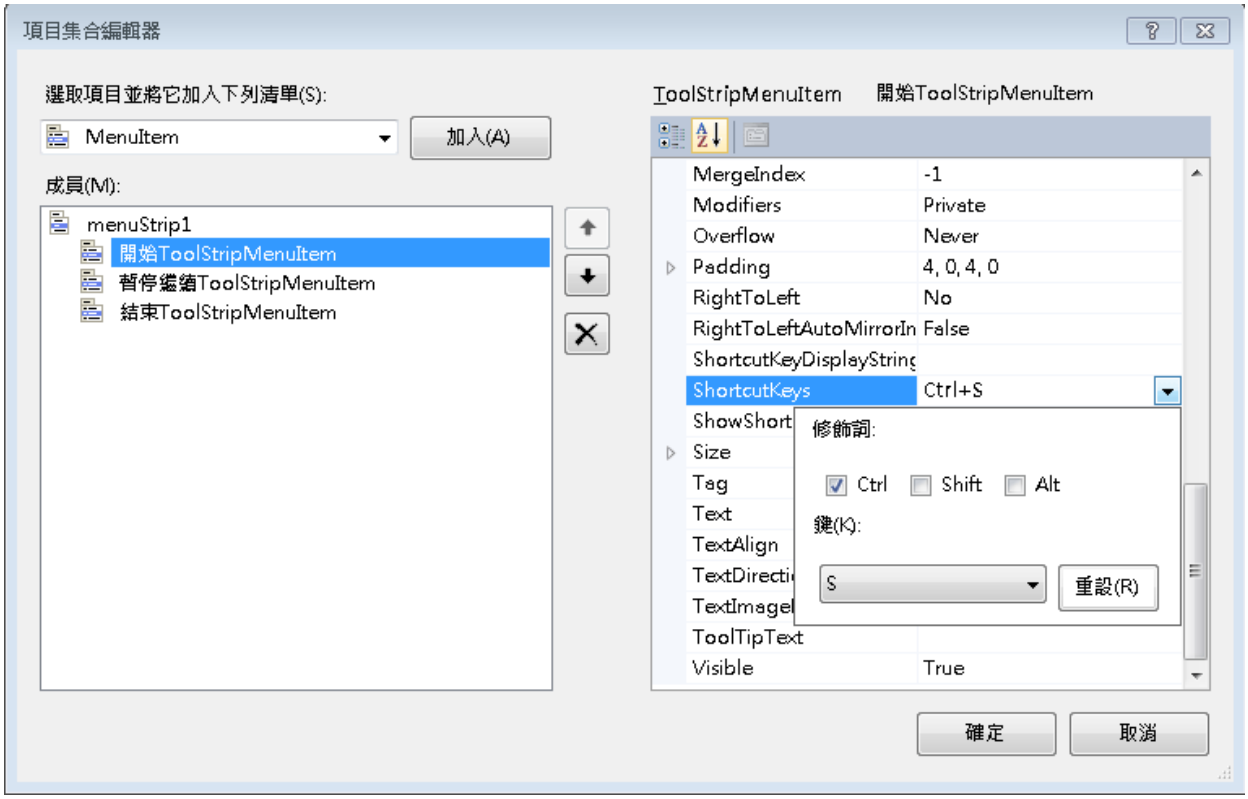
```
if (B.Top < menuStrip1.Height) { Vy = Math.Abs(Vy); } //視窗頂部碰撞
```

不僅如此，球的移動事實上是跳躍式的，即使如上修改之後，球在撞擊邊緣時還是會有一點侵犯到功能表，所以最好將 menuStrip1 物件的層次移到最上層，這樣看起來就會正常多了，操作畫面如下：

11-6 建立快捷鍵

[設定功能表項目的快捷鍵]

目前設計其實還是不夠方便！因為遊戲中滑鼠必須一直控制著擊球板，如何有空去點選功能表來暫停程式呢？此時快捷鍵(Shortcut Key)就有用了！請對著功能表的空白處按下右鍵，在跳出式功能表中選擇「編輯項目」會出現如下的視窗：

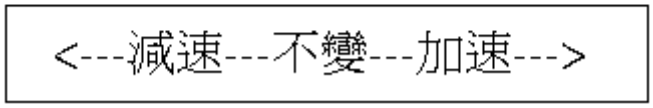


請設定 Ctrl+S(Start)為遊戲開始，Ctrl+P(Pause)為遊戲暫停與繼續，Ctrl+X(eXit)為結束程式，這些都是遵循一般軟體的習慣，你當然可以自己設定任何鍵，不過為了怕玩家不習慣，寫程式在此最好不要標新立異。作好之後程式進行中使用鍵盤也可以控制程式了！不過應該將 Form_Load 事件的啟動 timer1 的程式刪除，這樣程式一開始球是靜止的，按功能表或 Ctrl+S 才會啟動。

11-7 改變反彈方向

[模擬變方向擊球的反應]

擊球板擊球時，以目前程式進行只能單純以 45 度角反彈，頗為單調。如果可以控制板子的擊球點，適度改變 Vx 速度的大小，就可以讓球的反彈角度有所變化了，這可以大大增加遊戲的好玩程度哦！不過這是一個有一點難的向量問題，請先看看如下的示意圖：



我們的目標是當球落在擊球板中心點偏向球來的方向時，讓 Vx 略為減速，落在球離開的方向時，讓 Vx 略為加速。計算方法還是要依據球的中心點位置 C，先算出球落於板子的相對位置比例：F = (C - P.Left) / P.Width，依據 Vx 的方向，如果是逆向(Vx < 0)的就將這個 F 比例逆轉(1.0 - F)。接著再將 F + 0.5 就會是一個介於 0.5~1.5 之間的比例，將它乘以 Vx 就是微調之後的速度了！如上圖的狀況，擊中板子正中央時倍數為 1！剛好為完全不改變 Vx，偏右(板子遠側)時會加速，偏左(板子近側)時會減速，程式碼如下：

```
double F = ((double)C - (double)P.Left) / (double)P.Width;
if (Vx < 0) { F = (1.0 - F); }
F = F + 0.5;
Vx = (int)Math.Round(Vx * F);
```

請將上面程式碼插入板子擊中球時的程式區塊內，程式碼裡面為顧及到資料型別的轉換加入了許多型別轉換函數，主要是何時應該作 double 的計算，何時又必須轉為整數等等，C#對於型別是非常挑剔的。試試看！我們現在可以打出類似乒乓或網球的「切球」效果了！

11-8 進階挑戰

一、如何加入音效？

提示：參考打地鼠遊戲。

二、如何加入速度調整的介面？

提示：在功能表內加入下拉式選單。

三、如何加入直接用鍵盤而非快捷鍵控制遊戲的功能？

提示：增加鍵盤事件執行功能表中的遊戲控制程式。

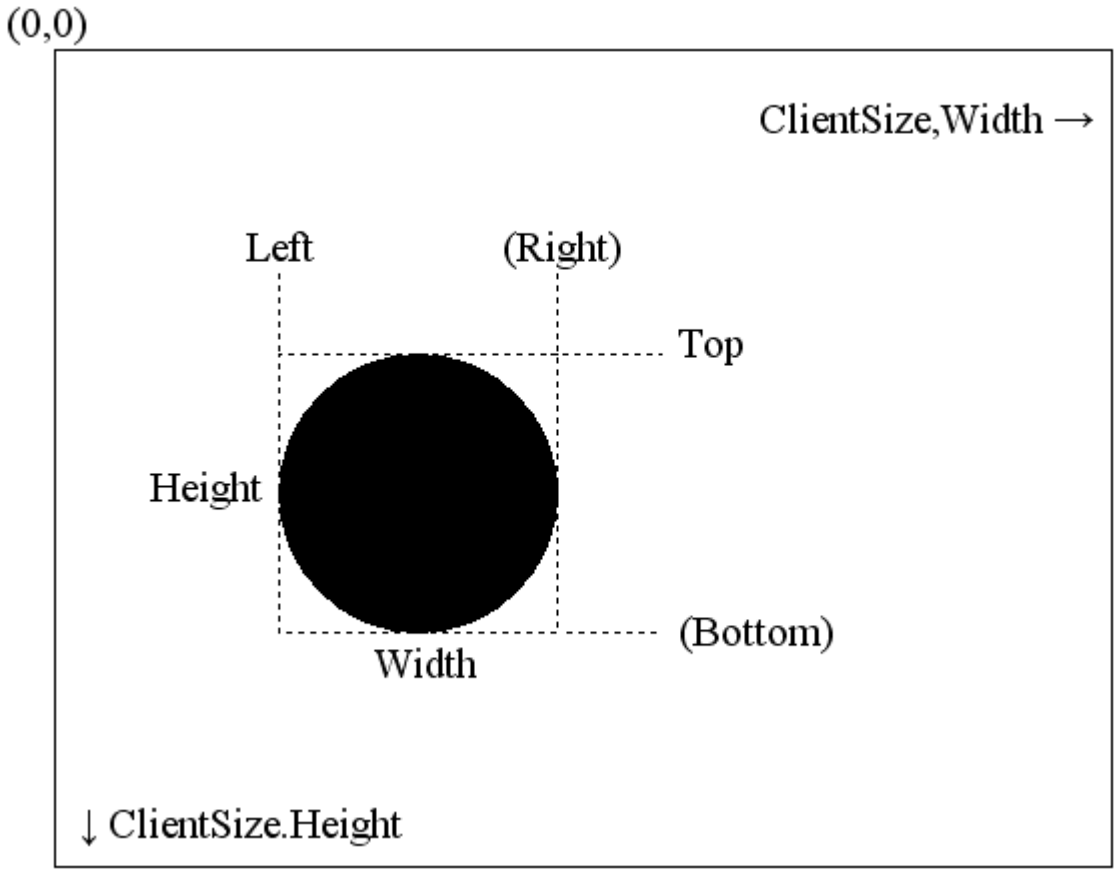
課後閱讀

細說碰撞程式

相信對於多數同學而言，這個單元最難理解的是所謂的「碰撞」程式，但是不可避免的，遊戲類的程式會有很多這類的狀況必須處理。在此就多花一點時間討論碰撞程式的一些細節吧！

[代表物件邊緣的屬性]

在本單元所謂的碰撞都是一個獨立的物件移動中去碰到某一個平面，基本上工具箱的物件都被預設為矩形，即使你將一個球形影像放入矩形的 label1 中，寫程式時電腦認定的物件「邊緣」也還是 label1 的矩形邊緣，而不是視覺上看到的圓形。所謂「碰撞」就是物件的邊緣與某個面(界線)有交會的情況，所以我們首先必須掌握的是如何描述物件「邊緣」的技巧，簡單的示意圖如下：



如上圖中黑色的球形影像物件，它的邊緣有 Left, Right, Top 與 Bottom 四個屬性可用，其中 Right 與 Bottom 兩個屬性在程式操作之中是「唯讀」的，也就是說你可以寫程式使用它們，但是不能改變它們的值，如果真想改變必須同時修改物件的 Left 與 Width 屬性值(或 Top 與 Height)，就是說 Right = Left + Width，同時 Bottom = Top + Height，系統隨時會計算出 Right 與 Bottom 值供你使用，但是不准你直接修改它們！如果你覺得這樣很麻煩，想

像一下以前的程式沒有這兩個屬性，每次需要「物件・Right」就必須寫「物件・Left +物件.Width」複雜度就更高了！一般人想看懂程式也會更困難。

[ClientSize 的概念]

其次，程式中屢屢出現的 ClientSize(客戶使用區)也可以從上圖中一目了然，就是視窗可以繪圖的區域，操作它們時必須掌握的是繪圖座標原點在左上角，於是視窗的左邊緣當然是 X=0，右邊緣就要寫成 X=this.ClientSize.Width；上邊緣是 Y=0，下緣則是 Y=this.ClientSize.Height。如果你忘了它寫成 this.Width 或 this.Height 代表的意義就是視窗的外框寬或高度了！

[物件碰撞的數學意義]

接下來想想兩者交會的狀況，當球往右移時，Right 邊界首先會有一個時間剛好重疊或略略超越 ClientSize.Width，這就可以視為碰撞到右牆了！條件式當然就是：

B.Right >= this.ClientSize.Width

在此不厭其詳的說明其實是想提醒讀者，抄程式是很容易的，完全理解則不太容易。如果你不能將上面這些機制變成腦中清晰的圖像與「動畫」，下次碰到要寫碰撞程式時還是會很困難的。請大家務必花點時間充分理解並孰悉它們吧！

此外，如果你覺得這些還難不倒你，可以想像一下真實世界的碰撞，應該不會只有垂直與水平面的彈跳，如果是球在斜坡上反彈呢？如果加上重力加速度的效應呢？這些現象要讓它們看來真實，程式技術中就必須加上許多物理與數學的知識了！所以想學習碰撞程式這還只是一個最簡單的開始呢！