



# 擴充實戰： PyGame 遊戲開發

遊戲開發是許多程式語言很喜歡的一個領域，因為遊戲開發需要使用的技術範圍相當的廣，除了多媒體音效、圖片動畫，程式設計應用更是其中的核心。

**PyGame** 是為了讓 **Python** 能夠進行遊戲開發工作所發展出來的模組，它能幫助 **Python** 控制音效音樂、圖片動畫，並進行程式的運作，是一個十分強大，功能完整的模組。

在本章中將詳細說明 **PyGame** 的使用方式，並利用實例範例帶領讀者學習其中重要的技巧，最後再利用一個有趣又好玩的遊戲進行專題開發，讓您也可以利用 **Python** 快速的進入遊戲開發的世界。



## B.1 Pygame 入門教學

Pygame 專門是為了開發遊戲所推出的 Python 模組。它是從 Simple Directmedia Layer (SDL) 延伸發展出來的。SDL 與 DirectX 類似，以精簡方式完成許多控制聲音、影像的基礎工作，大幅簡化程式碼，使開發遊戲工作更為容易。

### B.1.1 Pygame 程式基本架構

前一章已安裝過 Pygame 模組。建立 Pygame 程式首先要匯入 Pygame 模組，語法為：

```
import pygame
```

然後啟動 Pygame 模組，語法為：

```
pygame.init()
```

接著建立繪圖視窗做為圖形顯示區域，語法為：

```
視窗變數 = pygame.display.set_mode(視窗尺寸)
```

例如建立一個寬 640、高 320 的繪圖視窗，存於 screen 變數：

```
screen = pygame.display.set_mode((640, 320))
```

繪圖視窗的原點 (0,0) 位於左上角，座標值向右、向下遞增：



Pygame 模組的 `display.set_caption` 方法可設定視窗標題，例如：

```
pygame.display.set_caption("這是繪圖視窗標題")
```



通常圖形不是直接畫在繪圖視窗中，而是在繪圖視窗建立一塊與繪圖視窗同樣大小的畫布，然後將圖形畫在畫布上。建立畫布的語法為：

```
背景變數 = pygame.Surface(screen.get_size())
背景變數 = 背景變數.convert()
```

**Surface** 方法可建立畫布，「**screen.get\_size()**」取得繪圖視窗尺寸，因此畫布會填滿繪圖視窗。畫布變數的 **convert** 方法可為畫布建立一個副本，加快畫布在繪圖視窗的顯示速度。例如建立 **background** 畫布變數：

```
background = pygame.Surface(screen.get_size())
background = background.convert()
```

畫布變數的 **fill** 方法的功能是為畫布填滿指定顏色，例如設定畫布為紅色：

```
background.fill((255,0,0))
```

建立畫布後並不會在繪圖視窗中顯示，需以視窗變數的 **blit** 方法繪製於視窗中，語法為：

```
視窗變數.blit(畫布變數, 繪製位置)
```



例如將 **background** 畫布從繪圖視窗左上角 (0,0) 開始繪製，覆蓋整個視窗：

```
screen.blit(background, (0,0))
```

最後更新繪圖視窗內容，才能顯示繪製的圖形，語法為：


```
pygame.display.update()
```

## 偵測關閉繪圖視窗

使用者若是按繪圖視窗右上角  鈕必須關閉繪圖視窗，結束程式執行，因此要以無窮迴圈檢查使用者是否按了  鈕，程式碼為：

```
1 running = True
2 while running:
3     for event in pygame.event.get():
4         if event.type == pygame.QUIT:
5             running = False
6 pygame.quit()
```

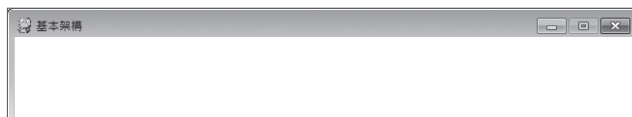


當 `running` 為 `True` 時，會重複執行 3-5 列程式檢查按鈕事件；若使用者按了  鈕會傳回「`pygame.QUIT`」，第 5 列程式將 `running` 設為 `False` 而跳出迴圈，第 6 列程式關閉繪圖視窗。

Pygame 程式基本架構整理為：

### 範例：Pygame 建立繪圖視窗

使用 Pygame 建立繪圖視窗，按  鈕會關閉視窗。



程式碼：chB\basic.py

```
1 import pygame
2 pygame.init() # 啟動 Pygame
3 screen = pygame.display.set_mode((640, 320)) # 建立繪圖視窗
4 pygame.display.set_caption(" 基本架構 ") # 繪圖視窗標題
5 background = pygame.Surface(screen.get_size()) # 建立畫布
6 background = background.convert()
7 background.fill((255,255,255)) # 畫布為白色
8 screen.blit(background, (0,0)) # 在繪圖視窗繪製畫布
9 pygame.display.update() # 更新繪圖視窗
10 running = True
11 while running: # 無窮迴圈
12     for event in pygame.event.get():
13         if event.type == pygame.QUIT: # 使用者按關閉鈕
14             running = False
15 pygame.quit() # 關閉繪圖視窗
```

## B.1.2 基本繪圖

繪製幾何圖形是遊戲模組的基本功能，設計者可將幾何圖形組合成遊戲角色。

### 繪製矩形：`pygame.draw.rect`

Pygame 繪製矩形的語法為：

```
pygame.draw.rect( 畫布, 顏色, [x座標, y座標, 寬度, 高度], 線寬 )
```

- **顏色**：由 3 個 0 到 255 整數組成，如 (255,0,0) 為紅色，(0,255,0) 為綠色，(0,0,255) 為藍色。

- **線寬**：若線寬大於 0 表示矩形邊線寬度，等於 0 表示實心矩形，預設值為 0。

例如在 (100,100) 處以線寬為 2，繪製寬度 80、高度 50 的紅色矩形：

```
pygame.draw.rect(background, (255,0,0), [100, 100, 80, 50], 2)
```

### 繪製圓形：pygame.draw.circle

Pygame 繪製圓形的語法為：

```
pygame.draw.circle(畫布, 顏色, (x座標, y座標), 半徑, 線寬)
```

例如在 (100,100) 處繪製半徑 50 的藍色實心圓形：

```
pygame.draw.circle(background, (0,0,255), (100,100), 50, 0)
```

### 繪製橢圓形：pygame.draw.ellipse

Pygame 繪製橢圓形的語法為：

```
pygame.draw.ellipse(畫布, 顏色, [x座標, y座標, x直徑, y直徑], 線寬)
```

例如在 (100,100) 處以線寬為 5，繪製 x 直徑 120、y 直徑 70 的綠色橢圓形：

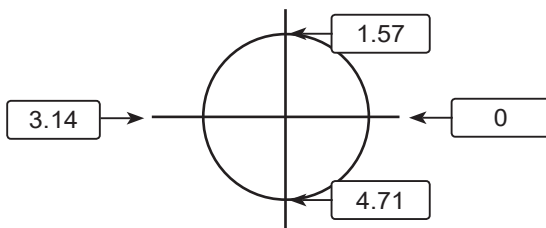
```
pygame.draw.ellipse(background, (0,255,0), [100, 100, 120, 70], 5)
```

### 繪製圓弧：pygame.draw.arc

Pygame 繪製圓弧的語法為：

```
pygame.draw.arc(畫布, 顏色, [x座標, y座標, x直徑, y直徑], 起始角, 結束角, 線寬)
```

- **起始角及結束角**：單位為弧度，以右方為 0，逆時針旋轉遞增角度。



例如在 (300,150) 處以線寬為 5，繪製直徑為 150 的紅色半圓形：

```
pygame.draw.arc(background, (255,0,0), [300, 150, 150, 150],  
0, 3.14, 5)
```



## 繪製直線：pygame.draw.line

Pygame 繪製直線的語法為：

```
pygame.draw.line( 畫布 , 顏色 , (x 座標 1, y 座標 1) , (x 座標 2, y 座標 2) , 線寬 )
```

例如在以線寬為 3、繪製由 (100,100) 到 (300,400) 的紫色直線：

```
pygame.draw.line(background, (255,0,255), (100,100), (300,400), 3)
```

## 繪製多邊形：pygame.draw.polygon

Pygame 繪製多邊形的語法為：

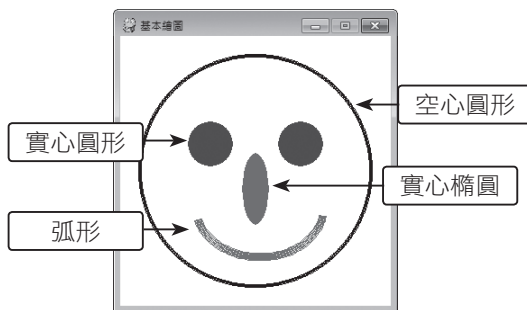
```
pygame.draw.polygon( 畫布 , 顏色 , 點座標串列 , 線寬 )
```

例如繪製由 (200,100)、(100,300)、(300,300) 三點組成的藍色實心三角形：

```
points = [(200,100), (100,300), (300,300)]  
pygame.draw.polygon(background, (0,0,255), points, 0)
```

## 範例：基本繪圖 - 人臉

以基本繪圖功能繪製人臉。



程式碼：chB\basicplot.py

```
.....略  
8 pygame.draw.circle(background, (0,0,0), (150,150), 130, 4)  
9 pygame.draw.circle(background, (0,0,255), (100,120), 25, 0)  
10 pygame.draw.circle(background, (0,0,255), (200,120), 25, 0)  
11 pygame.draw.ellipse(background, (255,0,255), [135, 130, 30, 80], 0)  
12 pygame.draw.arc(background, (255,0,0), [80, 130, 150, 120],  
    3.4, 6.1, 9)  
.....略
```

### B.1.3 載入圖片

使用幾何繪圖無法畫出精緻圖形，若有現成圖片可載入 Pygame 中直接使用。載入圖片的語法為：

```
圖片變數 = pygame.image.load( 圖片檔案路徑 )
```

圖片載入後通常會以 **convert** 方法處理，增加繪製速度，語法為：

```
圖片變數 .convert()
```

例如載入 <media> 資料夾 <img01.jpg> 圖片檔存於 image 變數：

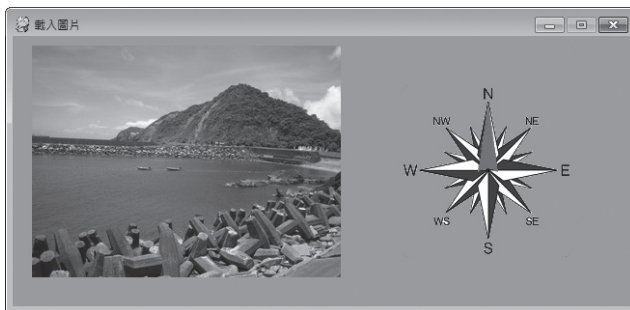
```
image = pygame.image.load("media\\img01.jpg")
image.convert()
```

Pygame 可載入的圖片類型有 JPG、PNG、GIF、BMP、PCX、TIF、LBM 等。

如果圖片經過去背處理，Pygame 顯示圖片時會呈現去背效果。

#### 範例：顯示載入圖片

載入圖片並顯示，右方為具有去背效果的圖片。



程式碼：ch15\loadpic.py

```
.....略
7 background.fill((0,255,0))
8 image = pygame.image.load("media\\img01.jpg")
9 image.convert()
10 compass = pygame.image.load("media\\compass.png")
11 compass.convert()
12 background.blit(image, (20,10))
13 background.blit(compass, (400,50))
.....略
```



## 程式說明

- 7 設定背景為綠色，方便觀察去背效果。
- 8-11 載入 <media> 資料夾 <img01.jpg> 及 <compass.png> 圖片。
- 12-13 繪製兩張圖片，其中 <compass.png> 圖片有去背效果。

## B.1.4 繪製文字

Pygame 可用繪圖的方式繪製文字，如此就可將文字與圖形合為一體。繪製文字前需先指定文字字體，語法為：

```
字體變數 = pygame.font.SysFont(字體名稱, 字體尺寸)
```

- **字體名稱**：如果要顯示中文及英文，字體名稱需用中文字體，常用的是 simhei (黑體)、fangsong (仿宋體) 等，否則無法顯示中文。

Pygame 繪製文字的語法為：

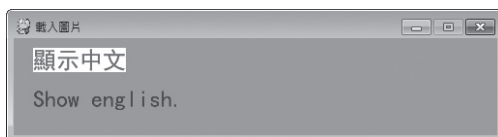
```
文字變數 = 字體變數.render(文字, 平滑值, 文字顏色, 背景顏色)
```

- **平滑值**：布林值，True 表示平滑文字，文字較美觀但繪製較費時；False 表示文字可能有鋸齒，繪製速度較快。

例如繪製中文及英文文字：(<plottext.py>)

```
……略
background.fill((0,255,0)) # 背景為綠色
font1 = pygame.font.SysFont("simhei", 24)
text1 = font1.render("顯示中文", True, (255,0,0),
(255,255,255)) # 中文, 不同背景色
background.blit(text1, (20,10))
text2 = font1.render("Show english.", True,
(0,0,255), (0,255,0)) # 英文, 相同背景色
background.blit(text2, (20,50))
……略
```

執行結果：






## B.2 Pygame 動畫處理

動畫是遊戲不可或缺的元素，在遊戲中，只有角色動起來，遊戲才會擁有生命，但動畫也是遊戲設計者最感頭痛的部分。Pygame 模組透過不斷重新繪製繪圖視窗，短短幾列程式就讓圖片動起來了！

### B.2.1 動畫處理基本程式架構

Pygame 基本程式架構中，最後是以無窮迴圈檢查使用者是否按  鈕關閉繪圖視窗，設計者可將不斷重新繪製繪圖視窗的程式碼置於此無窮迴圈內。動畫處理的基本程式架構為：(<basicmotion.py>)

```
.....略
7 background.fill((255,255,255))
8
9 clock = pygame.time.Clock() # 建立時間元件
10 running = True
11 while running:
12     clock.tick(30) # 每秒執行 30 次
13     for event in pygame.event.get():
14         if event.type == pygame.QUIT:
15             running = False
16     screen.blit(background, (0,0)) # 清除繪圖視窗
17
18     pygame.display.update() # 更新繪圖視窗
19 pygame.quit() # 關閉繪圖視窗
```

第 9 列建立 `clock` 元件，第 12 列利用此元件的 `tick` 方法設定每秒重繪次數，此處設為每秒重繪 30 次。重繪次數越多，動畫會越流暢，但 CPU 負擔越重，如果超過負荷，程式可能當機。如無特殊需求，一般設為「30」。

第 16 列以 `background` 背景畫布覆蓋繪圖視窗，會將繪圖視窗中所有內容清除，讓設計者重新繪製。

設計者可將一次性設計工作如建立幾何圖形、載入圖片、變數初值設定等程式碼置於第 8 列，再將移動後繪製圖片的程式碼置於第 17 列，相當於圖片一秒會移動 30 次，造成流暢的動畫效果。



## B.2.2 水平移動的藍色球體

以一個水平移動的藍色球體簡單動畫說明動畫處理程式。

### 範例：藍色球體水平移動

開始時藍色球體位於水平中央位置並向右移動，碰到右邊界時會反彈向左移動，碰到左邊界時也會反彈向右移動。



程式碼：chB\horizontalmotion.py

……略

```
8 ball = pygame.Surface((30,30)) # 建立球矩形繪圖區
9 ball.fill((255,255,255)) # 矩形區塊背景為白色
10 pygame.draw.circle(ball, (0,0,255), (15,15), 15, 0) # 畫藍色球
11 rect1 = ball.get_rect() # 取得球矩形區塊
12 rect1.center = (320,45) # 球起始位置
13 x, y = rect1.topleft # 球左上角座標
14 dx = 3 # 球運動速度
15 clock = pygame.time.Clock()
16 running = True
17 while running:
18     clock.tick(30) # 每秒執行 30 次
19     for event in pygame.event.get():
20         if event.type == pygame.QUIT:
21             running = False
22     screen.blit(background, (0,0)) # 清除繪圖視窗
23     x += dx # 改變水平位置
24     rect1.center = (x,y)
25     if(rect1.left <= 0 or rect1.right >= screen.get_width()): # 到達左右邊界
26         dx *= -1
27     screen.blit(ball, rect1.topleft)
28     pygame.display.update()
29 pygame.quit()
```

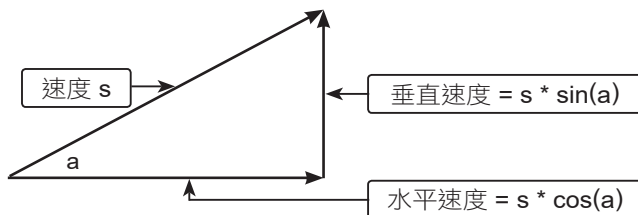
### 程式說明

- 8-14 執行一次性工作：建立球體及設定變數。
- 8-9 建立球體繪圖區並將背景設為與 `background` 背景色相同 (白色)，繪製球體時才不會呈現矩形底色。

- 10 繪製藍色球。
- 11-12 取得球體矩形區塊並設定球體起始位置。
- 13 取得球體左上角座標，用 `blit` 方法繪圖時以此點做為球體位置。
- 14 設定球體運動速度，數值越大，速度越快。
- 23-27 繪製動畫，每秒畫 30 次：改變球體位置後重繪。
- 23 改變球體水平位置：`dx` 為正時向右，`dx` 為負時向左。
- 24 更新位置。
- 25-26 球體碰到左、右邊界時改變 `dx` 正、負號即可改變球體移動方向。
- 27 重新繪製球體。

### B.2.3 自由移動的藍色球體

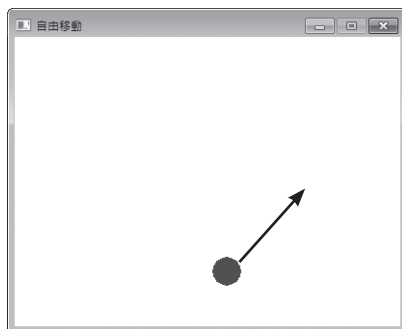
一般球體不是在單一方向移動，而是可以任意方向移動，此時就要將移動速度拆為水平速度及垂直速度，控制水平、垂直速度就能控制移動方向。水平及垂直速度可用三角函數取得，計算方式如下圖：



設定移動方向與水平線的夾角  $a$ ，即可用 `cos`、`sin` 函數取得水平、垂直速度。

#### 範例：藍色球體自由移動

開始時藍色球體以隨機角度向右上移動，撞到邊緣會反彈繼續移動。





程式碼：chB\freemotion.py

```
.....略
14 direction = random.randint(20,70) # 起始角度
15 radian = math.radians(direction) # 轉為弧度
16 dx = 5 * math.cos(radian) # 球水平運動速度
17 dy = -5 * math.sin(radian) # 球垂直運動速度
.....略
25 screen.blit(background, (0,0)) # 清除繪圖視窗
26 x += dx # 改變水平位置
27 y += dy # 改變垂直位置
28 rect1.center = (x,y)
29 if(rect1.left <= 0 or rect1.right >=
    screen.get_width()): # 到達左右邊界
30     dx *= -1 # 水平速度變號
31 elif(rect1.top <= 5 or rect1.bottom >=
    screen.get_height()-5): # 到達上下邊界
32     dy *= -1 # 垂直速度變號
33 screen.blit(ball, rect1.topleft)
34 pygame.display.update()
35 pygame.quit()
```

## 程式說明

- 1-13 與前一範例相同：建立藍色球體。
- 14 以亂數設定起始角度，如此每次執行程式球體移動路徑才會不同。
- 15 三角函數的參數單位是弧度，此列程式將角度度數轉換為弧度。
- 16-17 計算水平、垂直速度，因向右上移動，所以垂直速度為負值。
- 26-27 同時改變水平及垂直位移，就會讓球在指定方向移動。
- 29-30 撞到左、右邊界就將水平速度變號。
- 31-32 撞到上、下邊界就將垂直速度變號。

## B.2.4 角色類別 (Sprite)

Pygame 遊戲中有許多元件會重複使用，例如射擊太空船遊戲，外星太空船可能多達數十艘，只要建立「角色類別」，即可創造多個相同物件。

Pygame 角色類別是最被遊戲設計者稱道的功能，它不但能複製多個物件，還能進行動畫繪製、碰撞偵測等。建立角色類別的基本語法為：

```
class 角色名稱(pygame.sprite.Sprite):  
    屬性 1 = 值 1  
    屬性 2 = 值 2  
    .....  
    def __init__(self, 參數 1, 參數 2, .....):  
        pygame.sprite.Sprite.__init__(self)  
        程式碼
```

屬性 1、屬性 2、……可有可無，通常做為類別中不同函式的共用變數。

「\_\_init\_\_」為類別建構式，類別中一定要有此函式，建立物件時會執行此函式，僅執行一次，通常用於建立圖形、載入圖片、初始化變數值等。

類別中可自行撰寫特定功能函式，例如繪製動畫功能，再於物件中呼叫。

以角色類別建立的角色物件無法直接在畫布中顯示，必須加入角色群組才能繪製。建立角色群組的語法為：

```
角色群組名稱 = pygame.sprite.Group()
```

使用角色群組的 **add** 方法可將角色物件加入角色群組，語法為：

```
角色群組名稱.add( 角色物件 )
```

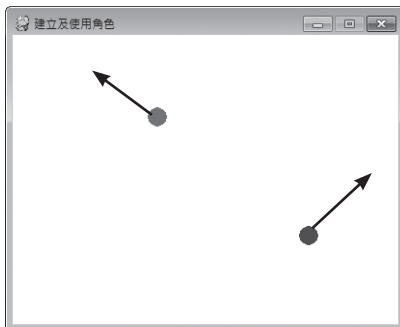
角色群組中可包含多個角色物件。最後使用角色群組的 **draw** 方法可將群組內全部角色物件繪製到畫布上，語法為：

```
角色群組名稱.draw( 畫布 )
```

以前一範例自由移動的球體為例，說明建立球體角色類別程式。

### 範例：以角色製作自由移動球體

紅色及藍色球體角色會獨立自由移動，碰到邊界會反彈。





程式碼：chB\sprite.py

```
1 import pygame, random, math
2
3 class Ball(pygame.sprite.Sprite):
4     dx = 0 #x 位移量
5     dy = 0 #y 位移量
6     x = 0 # 球 x 座標
7     y = 0 # 球 y 座標
8
9     def __init__(self, speed, srx, sry, radium, color):
10         pygame.sprite.Sprite.__init__(self)
11         self.x = srx
12         self.y = sry
13         self.image = pygame.Surface([radium*2, radium*2]) #繪製球體
14         self.image.fill((255,255,255))
15         pygame.draw.circle(self.image, color, (radium,radium), radium, 0)
16         self.rect = self.image.get_rect() # 取得球體區域
17         self.rect.center = (srx,sry) # 初始位置
18         direction = random.randint(20,70) # 移動角度
19         radian = math.radians(direction) # 角度轉為強度
20         self.dx = speed * math.cos(radian) # 球水平運動速度
21         self.dy = -speed * math.sin(radian) # 球垂直運動速度
22
23     def update(self):
24         self.x += self.dx # 計算球新餘標
25         self.y += self.dy
26         self.rect.x = self.x # 移動球圖形
27         self.rect.y = self.y
28         if(self.rect.left <= 0 or self.rect.right >=
29             screen.get_width()): # 到達左右邊界
30             self.dx *= -1 # 水平速度變號
31         elif(self.rect.top <= 5 or self.rect.bottom >=
32             screen.get_height()-5): # 到達上下邊界
33             self.dy *= -1 # 垂直速度變號
```

### 程式說明

- 3 角色類別名稱為 Ball。
- 4-7 x、y 位移量及球的 x、y 座標在 update 函式也要使用，故設為屬性。

- 9            參數 `speed` 為球體移動速度，`srx`、`sry` 為球體初始位置，`radius` 為球體半徑，`color` 為球體顏色。
- 11-12       設定 `x`、`y` 屬性值為參數 `srx`、`sry`，即球體位置。
- 13-17       畫出球體圖形並設定初始位置。
- 18-21       以亂數取得初始移動角度並計算水平、垂直移動速度。
- 23-31       自訂球體移動功能函式：`update` 函式。
- 24-27       計算球體新位置並將球體移到新位置。
- 18-31       碰到邊界反彈。

程式碼：chB\sprite.py (續)

```

.....略
40 allsprite = pygame.sprite.Group() # 建立角色群組
41 ball1 = Ball(8, 100, 100, 20, 20, (0,0,255)) # 建立藍色球物件
42 allsprite.add(ball1) # 加入角色群組
43 ball2 = Ball(6, 200, 250, 20, 20, (255,0,0)) # 建立紅色球物件
44 allsprite.add(ball2)
.....略
52 screen.blit(background, (0,0)) # 清除繪圖視窗
53 ball1.update() # 物件更新
54 ball2.update()
55 allsprite.draw(screen)
56 pygame.display.update()
57 pygame.quit()

```

#### 程式說明

- 40            建立角色群組變數 `allsprite`。
- 41-44       建立兩個球體 (藍色及紅色) 並加入角色群組。
- 53-54       每秒執行球體移動函式 (`update`) 30 次。
- 55            繪製所有球體角色 (藍色及紅色球體)。

### B.2.5 碰撞偵測

角色物件提供了數個碰撞偵測方法，可對角色物件碰撞做各種不同形式的偵測，較常用的碰撞偵測有兩種：



## 角色物件與角色物件的碰撞

偵測兩個角色物件的碰撞使用 `collide_rect` 方法，語法為：

```
偵測變數 = pygame.sprite.collide_rect(角色物件1, 角色物件2)
```

- **偵測變數**：布林值，`True` 表示兩角色物件發生碰撞，`False` 表示沒有碰撞。

## 角色物件與角色群組的碰撞

偵測一個角色物件與角色群組的碰撞使用 `spritecollide` 方法，語法為：

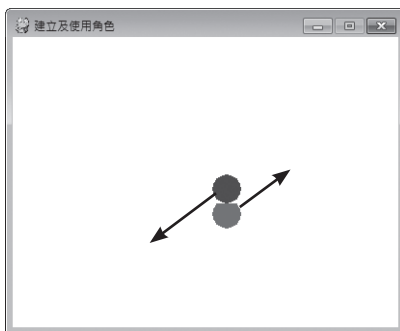
```
偵測變數 = pygame.sprite.spritecollide(角色物件, 角色群組, 移除值)
```

- **偵測變數**：傳回在角色群組中發生碰撞的角色物件串列，由串列長度可知是否發生碰撞：串列長度為 `0` 表示未發生碰撞，大於 `0` 表示發生碰撞。
- **移除值**：布林值，`True` 表示會將發生碰撞的角色物件從角色群組中移除，`False` 表示不從角色群組中移除。

下面範例使用前一小節建立的球體角色說明兩個角色物件碰撞，角色物件與角色群組碰撞則在下一節「應用：打磚塊遊戲」中再說明。

### 範例：球體物件碰撞

紅色及藍色球體角色會獨立自由移動，碰到邊界會反彈，互撞也會反彈。



`Ball` 角色類別新增 `collidebounce` 函式做為發生碰撞時的處理程式碼。

程式碼：chB\collide.py

```
1 import pygame, random, math
2
3 class Ball(pygame.sprite.Sprite):
4     .....略
```



```

33     def collidebounce(self):
34         self.dx *= -1

```

### 程式說明

- 33-34 球體碰撞後水平速度變號造成反彈。

無窮迴圈中加入偵測碰撞程式碼。

程式碼：chB\collide.py (續)

```

.....略
55     ball1.update()    # 物件更新
56     ball2.update()
57     allsprite.draw(screen)
58     result = pygame.sprite.collide_rect(ball1, ball2)
59     if result == True:
60         ball1.collidebounce()
61         ball2.collidebounce()
62     pygame.display.update()
63     pygame.quit()

```

### 程式說明

- 58 進行兩球體物件碰撞偵測。
- 59 傳回值為 True 表示發生碰撞，執行 60-61 列。
- 60-61 兩球體都反彈。

## B.2.6 鍵盤事件

使用者可透過鍵盤按鍵來操控遊戲中角色運作，取得鍵盤事件的方法有兩種：

### pygame.KEYDOWN、pygame.KEYUP 事件

pygame.KEYDOWN 是當使用者按下鍵盤時觸發，pygame.KEYUP 是當使用放開鍵盤時觸發，語法為：

```

for event in pygame.event.get():
    if event.type == 鍵盤事件:
        if event.key == pygame. 鍵盤常數:
            處理程式碼

```

- 鍵盤事件：pygame.KEYDOWN 或 pygame.KEYUP。
- 鍵盤常數：每個按鍵有對應的鍵盤常數，如「0」按鍵的鍵盤常數為「K\_0」。



## pygame.key.get\_pressed 事件

`pygame.key.get_pressed` 會傳回當前所有鍵狀態的串列，若指定鍵的值為 `True` 就表示該鍵被按，語法為：

```
按鍵變數 = pygame.key.get_pressed()
if 按鍵變數 [pygame. 鍵盤常數]:
    處理程式碼
```

常用的按鍵與鍵盤常數對應表：

按鍵	鍵盤常數	按鍵	鍵盤常數
0 到 9	K_0 到 K_9	向上鍵	K_UP
a 到 z	K_a 到 K_z	向下鍵	K_DOWN
F1 到 F12	K_F1 到 K_F12	向右鍵	K_RIGHT
空白鍵	K_SPACE	向左鍵	K_LEFT
Enter 鍵	K_RETURN	Esc 鍵	K_ESCAPE
定位鍵	K_TAB	後退鍵	K_BACKSPACE
「+」鍵	K_PLUS	「-」鍵	K_MINUS
「Insert」鍵	K_INSERT	「Home」鍵	K_HOME
「End」鍵	K_END	「Caps Lock」鍵	K_CAPSLOCK
右方「Shift」鍵	K_RSHIFT	「PgUp」鍵	K_PAGEUP
左方「Shift」鍵	K_LSHIFT	「PgDn」鍵	K_PAGEDOWN
右方「Ctrl」鍵	K_RCTRL	右方「Alt」鍵	K_RALT
左方「Ctrl」鍵	K_LCTRL	左方「Alt」鍵	K_LALT

### 範例：鍵盤控制球體移動

按鍵盤向右鍵，藍球會向右移動，按住向右鍵不放球體會快速向右移動，若到達邊界則停止移動；按向左鍵藍球會向左移動，操作方式與向右鍵相同。



程式碼：chB\keyevent.py

```
.....略
22     keys = pygame.key.get_pressed() # 檢查按鍵被按
23     if keys[pygame.K_RIGHT] and rect1.right <
        screen.get_width(): # 按向右鍵且未達右邊界
24         rect1.centerx += dx # 向右移動
25     elif keys[pygame.K_LEFT] and rect1.left > 0:
        # 按向左鍵且未達左邊界
26         rect1.centerx -= dx # 向左移動
27     screen.blit(background, (0,0)) # 清除繪圖視窗
28     screen.blit(ball, rect1.topleft)
29     pygame.display.update()
30     pygame.quit()
```

#### 程式說明

- 22 取得所有按鍵狀態。
- 23-24 使用者按向右鍵且球尚未到達右邊界就將球向右移動。
- 25-26 使用者按向左鍵且球尚未到達左邊界就將球向左移動。

## B.2.7 滑鼠事件

使用者除了可透過鍵盤按鍵來操控遊戲中角色，也可透過滑鼠來操控。滑鼠事件分為按鈕事件及滑動事件兩大類：

### 滑鼠按鈕事件

`pygame.mouse.get_pressed` 會傳回滑鼠按鈕狀態串列，若指定按鈕的值為 `True` 就表示該按鈕被按，語法為：

```
按鈕變數 = pygame.mouse.get_pressed()
if 按鈕變數[按鈕索引]:
    處理程式碼
```

- 按鈕索引：0 表示按滑鼠左鍵，1 表示按滑鼠滾輪，2 表示按滑鼠右鍵。

### 滑鼠滑動事件

`pygame.mouse.get_pos` 會傳回目前滑鼠位置座標串列，語法為：

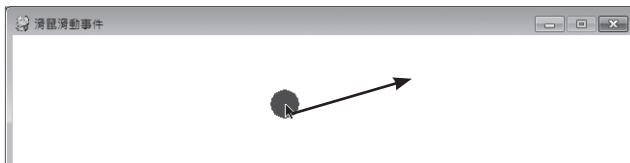
```
位置變數 = pygame.mouse.get_pos()
```

- 位置變數：串列第一個元素為 `x` 座標，第二個元素為 `y` 座標。



## 範例：藍色球隨滑鼠移動

開始時藍色球不會移動，按滑鼠左鍵後滑動滑鼠，球會跟著滑鼠移動；按滑鼠右鍵後，球不會跟著滑鼠移動。



程式碼：chB\mouseevent.py

```
.....略
16 playing = False # 開始時球不能移動
17 while running:
18     clock.tick(30) # 每秒執行 30 次
19     for event in pygame.event.get():
20         if event.type == pygame.QUIT:
21             running = False
22     buttons = pygame.mouse.get_pressed()
23     if buttons[0]: # 按滑鼠左鍵後球可移動
24         playing = True
25     elif buttons[2]: # 按滑鼠右鍵後球不能移動
26         playing = False
27     if playing == True: # 球可移動狀態
28         mouses = pygame.mouse.get_pos() # 取得滑鼠座標
29         rect1.centerx = mouses[0] # 移動滑鼠
30         rect1.centery = mouses[1]
.....略
```

### 程式說明

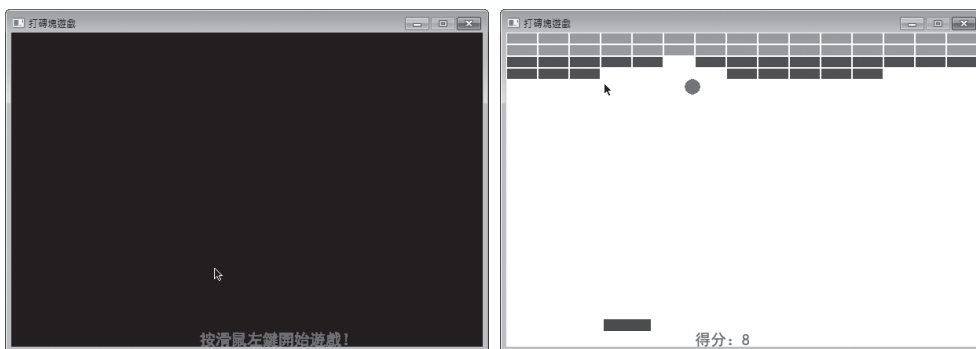
- 16      `playing` 做為球是否可移動旗標：`True` 為可移動，`False` 不可移動。
- 22      檢查滑鼠按鈕是否被按。
- 23-24   按滑鼠左鍵就設 `playing` 為 `True`，表示球體可移動。
- 25-26   按滑鼠右鍵就設 `playing` 為 `False`，表示球體不能移動。
- 27-30   若 `playing` 為 `True` 就取得滑鼠座標，球體移到滑鼠位置。

## B.3 實戰：打磚塊遊戲

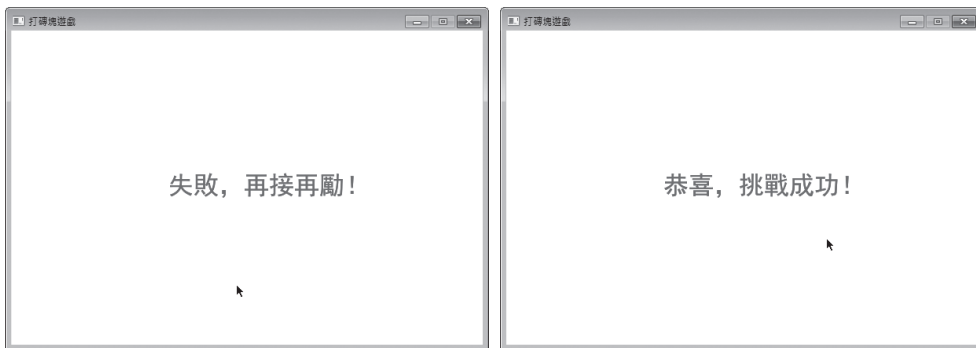
三十餘年前，街頭電玩遊戲機中最流行的遊戲就是「打磚塊」。時至今日，雖然網路連線遊戲日新月異，「打磚塊」這款小遊戲仍在許多人心中佔有一席之地，本應用製作一個簡單的打磚塊遊戲。

### B.3.1 應用程式總覽

開始時下方會顯示「按滑鼠左鍵開始遊戲」訊息，使用者按滑鼠左鍵就顯示遊戲畫面。使用者移動滑鼠控制滑板，滑板只能左右移動，位置與滑鼠 x 座標相同；共有 60 個磚塊，被球撞到的磚塊會消失，同時分數會增加，球撞到磚塊及滑板會發出不同音效。



如果球體碰到下邊界就表示球體已出界，顯示「失敗，再接再勵！」訊息並結束程式；若全部磚塊都消失則顯示「恭喜，挑戰成功！」訊息並結束程式。





## B.3.2 球體、磚塊、滑板角色類別

本遊戲主角是球體、磚塊及滑板，都設計為角色類別。首先是球體角色，與16.2.4節的球體角色類似：

程式碼：chB\brickgame.py

```
1 import pygame, random, math, time
2
3 class Ball(pygame.sprite.Sprite): # 球體角色
4     dx = 0 # x 位移量
5     dy = 0 # y 位移量
6     x = 0 # 球 x 座標
7     y = 0 # 球 y 座標
8     direction = 0 # 球移動方向
9     speed = 0 # 球移動速度
10
11     def __init__(self, sp, srx, sry, radium, color):
12         pygame.sprite.Sprite.__init__(self)
13         self.speed = sp
14         self.x = srx
15         self.y = sry
16         self.image = pygame.Surface([radium*2, radium*2]) # 繪製球體
17         self.image.fill((255,255,255))
18         pygame.draw.circle(self.image, color, (radium,radium), radium, 0)
19         self.rect = self.image.get_rect() # 取得球體區域
20         self.rect.center = (srx,sry) # 初始位置
21         self.direction = random.randint(40,70) # 移動角度
22
23     def update(self): # 球體移動
24         radian = math.radians(self.direction) # 角度轉為弧度
25         self.dx = self.speed * math.cos(radian) # 球水平運動速度
26         self.dy = -self.speed * math.sin(radian) # 球垂直運動速度
27         self.x += self.dx # 計算球新座標
28         self.y += self.dy
29         self.rect.x = self.x # 移動球圖形
30         self.rect.y = self.y
31         if(self.rect.left <= 0 or self.rect.right >=
32             screen.get_width()-10): # 到達左右邊界
33             self.bouncelr()
34         elif(self.rect.top <= 10): # 到達上邊界
35             self.rect.top = 10
36             self.bounceup()
```

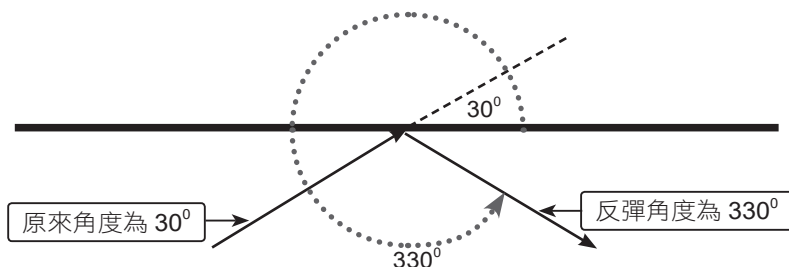
```

36         if(self.rect.bottom >= screen.get_height()-10):
37             # 到達下邊界出界
38             return True
39         else:
40             return False
41     def bounceup(self): # 上邊界反彈
42         self.direction = 360 - self.direction
43
44     def bouncelr(self): # 左右邊界反彈
45         self.direction = (180 - self.direction) % 360

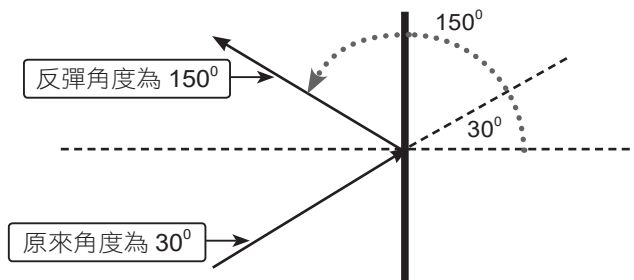
```

### 程式說明

- 11-21 建立球體圖形、設定初始位置、隨機設定起始移動方向。
- 23-39 球體移動自訂函式。
- 24-26 根據移動方向計算水平及垂直速度。
- 27-30 計算球體新座標並移到新位置。
- 31-32 球體碰到左、右邊界時進行反彈。
- 33-35 球體碰到上邊界時進行反彈：34 列在碰到上邊界時設其 y 座標為 10，避免連續碰撞上邊界。
- 36-39 若球體碰到下邊界就傳回 True 表示球出界，否則傳回 False 表示球未出界。
- 41-42 球體碰到上邊界時的反彈處理函式：「反彈角度 =  $360^\circ$  - 原來角度」。例如下圖原來角度為  $30^\circ$  度，反彈角度為  $330^\circ$  度。



- 44-45 球體碰到左、右邊界時的反彈處理函式：「反彈角度 =  $180^\circ$  - 原來角度」。因為反彈角度可能得到負值，所以再用除以 360 的餘數將其轉為正數。例如下圖原來角度為  $30^\circ$  度，反彈角度為  $150^\circ$  度。



磚塊及滑板類別程式碼為：

程式碼：chB\brickgame.py (續)

```

47 class Brick(pygame.sprite.Sprite): # 磚塊角色
48     def __init__(self, color, x, y):
49         pygame.sprite.Sprite.__init__(self)
50         self.image = pygame.Surface([38, 13]) # 磚塊 38x13
51         self.image.fill(color)
52         self.rect = self.image.get_rect()
53         self.rect.x = x
54         self.rect.y = y
55
56 class Pad(pygame.sprite.Sprite): # 滑板角色
57     def __init__(self):
58         pygame.sprite.Sprite.__init__(self)
59         self.image = pygame.image.load("media\\pad.png") # 滑板圖片
60         self.image.convert()
61         self.rect = self.image.get_rect()
62         self.rect.x = int((screen.get_width() -
63                             self.rect.width)/2) # 滑板位置
64         self.rect.y = screen.get_height() -
65                             self.rect.height - 20
66
67     def update(self): # 滑板位置隨滑鼠移動
68         pos = pygame.mouse.get_pos() # 取得滑鼠座標
69         self.rect.x = pos[0] # 滑鼠 x 座標
70         if self.rect.x > screen.get_width() -
71             self.rect.width: # 不要移出右邊界
72             self.rect.x = screen.get_width() - self.rect.width

```

#### 程式說明

- 47-54 磚塊角色類別：建立磚塊圖形並設定初始位置，參數 color 為磚塊顏色，x、y 為磚塊初始位置。



- 56-69 滑板角色類別。
- 59 滑板使用 <pad.png> 圖片做為滑板圖形。
- 65-69 滑板隨滑鼠移動位置自訂函式：滑板的 y 座標不會改變，故不需設定，僅在 67 列設定 x 座標即可。

### B.3.3 自訂函式及主程式

球出界或全部磚塊消失都會結束程式，撰寫 `gameover` 自訂函式來執行結束程式功能。

程式碼：chB\brickgame.py (續)

```
71 def gameover(message): # 結束程式
72     global running
73     text = font1.render(message, 1, (255,0,255)) # 顯示訊息
74     screen.blit(text, (screen.get_width()/2-100,screen.get_height()/2-20))
75     pygame.display.update() # 更新畫面
76     time.sleep(3) # 暫停 3 秒
77     running = False # 結束程式
```

- 71 參數 `message` 是要顯示的訊息文字。
- 72 及 77 設定主程式 `running` 為 `False` 可結束程式，所以要宣告全域變數。
- 73-74 繪製訊息。
- 75 更新畫面後訊息才會顯示。
- 76 等 3 秒鐘再結束程式。

主程式建立各種角色及初值設定。

程式碼：chB\brickgame.py (續)

```
79 pygame.init()
80 score = 0 # 得分
81 font = pygame.font.SysFont("SimHei", 20) # 下方訊息字體
82 font1 = pygame.font.SysFont("SimHei", 32) # 結束程式訊息字體
83 soundhit = pygame.mixer.Sound("media\\hit.wav") # 接到磚塊音效
84 soundpad = pygame.mixer.Sound("media\\pad.wav") # 接到滑板音效
85 screen = pygame.display.set_mode((600, 400))
86 pygame.display.set_caption("打磚塊遊戲")
87 background = pygame.Surface(screen.get_size())
88 background = background.convert()
89 background.fill((255,255,255))
```



```
90 allsprite = pygame.sprite.Group() # 建立全部角色群組
91 bricks = pygame.sprite.Group() # 建立磚塊角色群組
92 ball = Ball(10, 300, 350, 10, (255,0,0)) # 建立紅色球物件
93 allsprite.add(ball) # 加入全部角色群組
94 pad = Pad() # 建立滑板球物件
95 allsprite.add(pad) # 加入全部角色群組
96 clock = pygame.time.Clock()
97 for row in range(0, 4): # 3 列方塊
98     for column in range(0, 15): # 每列 15 磚塊
99         if row == 0 or row == 1: # 1,2 列為綠色磚塊
100             brick = Brick((0,255,0), column * 40 + 1, row * 15 + 1)
101             if row == 2 or row == 3: # 3,4 列為藍色磚塊
102                 brick = Brick((0,0,255), column * 40 + 1, row * 15 + 1)
103             bricks.add(brick) # 加入磚塊角色群組
104             allsprite.add(brick) # 加入全部角色群組
105 msgstr = "按滑鼠左鍵開始遊戲！" # 起始訊息
106 playing = False # 開始時球不會移動
107 running = True
```

- 80 score 記錄得分，每打到一個磚塊得 1 分。
- 81-82 設定兩個字體大小，記得字型使用「SimHei」顯示中文。
- 83-84 載入碰撞磚塊及滑板的音效檔。
- 90-91 建立儲存全部角色及磚塊角色的角色群組。
- 92-95 分別建立球體角色及滑板角色，並加入全部角色群組。
- 97-102 以迴圈建立 4 列 15 行磚塊：磚塊長寬為 38x13，100 及 102 列設定磚塊位置為 40x15，留 2 像素做為磚塊間隔。
- 103-104 磚塊要同時加入全部角色群組及磚塊角色群組，全部角色群組用於繪製圖形，磚塊角色群組用於偵測與球體的碰撞。
- 105 設定程式開始時顯示的訊息。
- 106 設定程式開始時球體不會移動。

主程式無窮迴圈動畫程式碼。

程式碼：chB\brickgame.py (續)

```
108 while running:
109     clock.tick(30)
110     for event in pygame.event.get():
111         if event.type == pygame.QUIT:
112             running = False
113     buttons = pygame.mouse.get_pressed() # 檢查滑鼠按鈕
114     if buttons[0]: # 按滑鼠左鍵後球可移動
115         playing = True
116     if playing == True: # 遊戲進行中
117         screen.blit(background, (0,0)) # 清除繪圖視窗
118         fail = ball.update() # 移動球體
119         if fail: # 球出界
120             gameover("失敗，再接再勵！")
121         pad.update() # 更新滑板位置
122         hitbrick = pygame.sprite.spritecollide(ball,
123             bricks, True) # 檢查球和磚塊碰撞
124         if len(hitbrick) > 0: # 球和磚塊發生碰撞
125             score += len(hitbrick) # 計算分數
126             soundhit.play() # 球撞磚塊聲
127             ball.rect.y += 20 # 球向下移
128             ball.bounceup() # 球反彈
129             if len(bricks) == 0: # 所有磚塊消失
130                 gameover("恭喜，挑戰成功！")
131         hitpad = pygame.sprite.collide_rect(ball,
132             pad) # 檢查球和滑板碰撞
133         if hitpad: # 球和滑板發生碰撞
134             soundpad.play() # 球撞滑板聲
135             ball.bounceup() # 球反彈
136         allsprite.draw(screen) # 繪製所有角色
137         msgstr = "得分：" + str(score)
138         msg = font.render(msgstr, 1, (255,0,255))
139         screen.blit(msg, (screen.get_width()/2-60,
140             screen.get_height()-20)) # 繪製訊息
141     pygame.display.update()
142     pygame.quit()
```

- 113-115 檢查滑鼠按鈕，若使用者按滑鼠左鍵就將 `playing` 設為 `True`，表示開始遊戲。
- 116-135 當 `playing` 設為 `True` 時才執行 117-135 列程式進行遊戲。



- 118-120 球體使用 `update` 函式移動後就檢查球體是否出界，若傳回 `True` 表示出界，結束程式並顯示失敗訊息。
- 121 滑板使用 `update` 函式隨滑鼠移動。
- 122 檢查球體與磚塊群組是否碰撞：注意第 3 個參數要設為 `True`，如此在發生碰撞後，被撞的磚塊才會被移除（同時從磚塊角色群組及全部角色群組移除）。
- 123 `hitbrick` 會傳回被撞磚塊串列，「`len(hitbrick)`」表示被撞磚塊數量，若大於 0 表示發生碰撞。
- 124-125 計算分數及播放音效。
- 126 球撞到磚塊後常會再連續撞旁邊磚塊，將球下移避免此現象。
- 127 球撞到磚塊後反彈。
- 128-129 若全部磚塊都消失就結束程式並顯示成功訊息。
- 130 檢查球體與滑板是否碰撞。
- 131-133 若球體與滑板發生碰撞就播放音效並將球反彈。
- 134 重繪所有角色。
- 135 若遊戲進行中就設定訊息為所得分數。
- 136-137 在螢幕下方繪製訊息。