1. **Pygame套件安裝與基本原理**
2. 首先，安裝Python的pygame套件：

|  |
| --- |
| **pip install pygame** |

1. **「繪圖視窗」和「畫布」是分開的：**通常不會直接在繪圖視窗畫圖，而會先畫在「畫布」上，再把畫布貼到「繪圖視窗」上。
2. **pygame的七大核心元素：**

|  |  |
| --- | --- |
| **核心元素** | **說明** |
| **pygame.Surface** | Surface 資料型別，代表一個矩形的影像(畫布)，用來繪製在螢幕上 |
| **pygame.Rect** | Rect 資料型別，用來定位矩形空間的位置和可以用來偵測物件是否碰撞的 |
| **pygame.event** | 事件模組，用來處理使用者觸發事件，包含自定義事件 |
| **pygame.font** | 文字模組，用來顯示文字，可用來顯示儀表板資料 |
| **pygame.draw** | 繪圖模組，可以畫出多邊形圖形，可當作背景物件 |
| **pygame.image** | 圖片模組，用來處理載入圖片等相關操作，可當作角色精靈（sprite） |
| **pygame.time** | 時間模組，包含控制遊戲迴圈迭代速率，確保反饋不會太快消逝 |

1. **基本語法**
2. 建立**繪圖視窗**，並設定**大小**與**標題**：

|  |
| --- |
| **繪圖視窗變數名稱 = pygame.display.set\_mode((寬, 高))** |
| **pygame.display.set\_caption("視窗標題")** |

1. 建立**畫布視窗**，並設定**大小**、**副本**與**背景顏色**：

|  |
| --- |
| **畫布變數名稱 = pygame.Surface(screen.get\_size())****畫布變數名稱 =畫布變數名稱.convert()** **畫布變數名稱.fill((R,G,B))** |

1. 在畫布上新增**文字**，並設定字體、大小、顏色與位置：

|  |
| --- |
| **字體變數名稱 = pygame.font.SysFont(字體, 文字大小)** **文字變數名稱 = 字體變數名稱.render(文字內容, 平滑度, (R,G,B), (R,G,B))****畫布變數名稱.blit(文字變數名稱, (x座標, y座標))** |

1. 在畫布上新增**幾何圖形，**線寬=0時為實心：

|  |
| --- |
| **pygame.draw.rect(畫布變數名稱, (R,G,B), [x坐標, y坐標, 寬度, 高度], 線寬)** **pygame.draw.circle(畫布變數名稱, (R,G,B), (x坐標, y坐標), 半徑, 線寬)****pygame.draw.ellipse(畫布變數名稱, (R,G,B), [x坐標, y坐標, x直徑, y直徑], 線寬)****pygame.draw.arc(畫布變數名稱, (R,G,B), [x坐標, y坐標, x直徑, y直徑], 起始角, 結束角, 線寬)****pygame.draw.line(畫布變數名稱, (R,G,B), (x坐標1, y坐標1), (x坐標2, y坐標2), 線寬)** |

1. 在畫布上新增**圖片**，：

|  |
| --- |
| **圖片變數 = pygame.image.load(圖片檔案路徑)****圖片變數 = pygame.transform.scale(圖片變數,(寬,高))****圖片變數.convert()****畫布變數名稱.blit(圖片變數, (x坐標, y坐標))** |

1. 將畫布顯示在繪圖視窗上：

|  |
| --- |
| **繪圖視窗變數名稱.blit(畫布變數名稱, (0,0))** **pygame.display.update()**  |

import sys

import pygame

# 初始化

pygame.init()

# 設定繪圖視窗

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 建立繪圖視窗「大小」，為 800x600

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定繪圖視窗「標題」， My First Game

# 建立畫布

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定畫布大小「大小」，等於 screen 的大小

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 產生副本以加速顯示速度

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定畫布背景顏色

# 在畫布上，新增文字

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定字體、大小

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定文字內容、平滑度、文字顏色、背景顏色

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 將文字顯示在畫布上，x座標:10;y座標:10

# 在畫布上，新增幾何圖形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,[70, 70, 500, 60], 4)              # 矩形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,[70, 150, 500, 60], \_\_\_\_)          # 實心矩形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,(100,300), 50, 4)                  # 圓形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,[200,250, 150, 80], 4)             # 橢圓形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,[400, 250, 70, 150] ,5 ,1.5 , 4)   # 圓弧形

pygame.draw.\_\_\_\_(\_\_\_\_, \_\_\_\_,(550,250), (550, 400), 4)          # 直線

# 在畫布上，新增圖片

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 新增圖片

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 設定圖片大小

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 產生副本以加速顯示速度

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 將圖片顯示在畫布上 ，x座標:600;y座標:10

# 繪圖視窗上顯示

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 在繪圖視窗上，顯示畫布，並設定座標位置

pygame.display.update()                       # 更新畫面，等所有操作完成後一次更新（若沒更新，則元素不會出現）

while True: # 無窮迴圈，執行遊戲的視窗，

    for event in pygame.event.get():   # 檢查所有可能發生的事件

        if event.type == pygame.QUIT:  # 當使用者結束視窗的事件發生時，則結束程式

            pygame.quit()

            sys.exit()

**三、建立腳色、簡易動畫與碰撞偵測**

1. 設定每秒執行**次數**

|  |
| --- |
| **變數名稱 = pygame.time.Clock()****變數名稱.tick(執行次數)** |

1. 偵測使用者滑鼠操作：0表示按滑鼠左鍵，1表示按滑鼠中鍵，2表示按滑鼠右鍵

|  |
| --- |
| **pygame.mouse.get\_pressed()** |

1. 取得目前滑鼠所在的座標

|  |
| --- |
| **pygame.mouse.get\_pos()** |

import sys

import pygame

# 初始化

pygame.init()

# 設定繪圖視窗

screen = pygame.display.set\_mode((800, 600)) # 建立繪圖視窗「大小」，為 800x600

pygame.display.set\_caption("籃球移動") # 設定繪圖視窗「標題」， My First Game

# 建立畫布

canvas = pygame.Surface(screen.get\_size())

canvas = canvas.convert()

canvas.fill((0,0,0))

# 建立籃球

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 設定籃球畫布大小，圓的直徑

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 設定籃球畫布背景顏色

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 在籃球畫布上產生藍球

x,y = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    # 設定起始x,y座標

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 取得籃球畫布矩形的大小

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 設定籃球畫布矩形，以中心點為中心，設定出現的座標位置

speed = 3                    # 設定移動速度

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 重要，計時物件

while True: # 無窮迴圈，執行遊戲的視窗，

    for event in pygame.event.get():   # 檢查所有可能發生的事件

        if event.type == pygame.QUIT:  # 當使用者結束視窗的事件發生時，則結束程式

            pygame.quit()

            sys.exit()

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        # 每秒執行30次

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_              # 增加x軸的值 (水平移動)

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 更新座標位置

    if  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:   #到達左右邊界

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          #正負值交換

    # 繪圖視窗上顯示

    screen.blit(canvas, (0,0))      # 在繪圖視窗上，顯示畫布，並設定座標位置

    screen.blit(ball, \_\_\_\_\_) # 在畫布上，顯示藍色球，並設定起始位置為目前藍色球畫布的左上x,y座標

    pygame.display.update()         # 更新畫面，等所有操作完成後一次更新（若沒更新，則元素不會出現）

1. **角色類別**
2. **pygame.sprite.Sprite類別**：創造角色，需繼承pygame.sprite.Sprite類別，便於創造大量相同的角色。

class 自訂類別名稱(pygame.sprite.Sprite):

    def \_\_init\_\_(self, 參數1, 參數2, ...):

        pygame.sprite.Sprite.\_\_init\_\_(self)

        self.image = pygame.Surface([…, …])  # 建立影像(畫布)

        self.rect = self.image.get\_rect()   # 取得影像(畫布)的矩形大小大

        self.rect.center = (x座標, y座標)   # 設定初始位置

    # 可以在主程式呼叫，用來撰寫角色的動作

    def 自訂方法名稱(self):

        pass

1. **角色群組：**通常角色也會有自己的畫布，故**移動角色相當於移動畫布的概念**。角色類別的繪製方式與一般畫布不太一樣，首先需要將角色物件加入角色群組中。

|  |
| --- |
| **角色群組名稱 = pygame.sprite.Group() # 創建角色群組****角色群組名稱.add(角色物件) # 新增角色至群組****角色群組名稱.draw(繪圖視窗) # 將所有角色畫出來** |



import sys

import pygame

import random

# 初始化

pygame.init()

class Ball(pygame.sprite.Sprite):

    def \_\_init\_\_(self, speed, x, y, r, color):     # 參數為「球速」、「球x,y座標」、「半徑」、「球顏色」

        pygame.sprite.Sprite.\_\_init\_\_(self)

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 設定球速

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 產生 直徑(寬)\*直徑(高) 的矩形畫布，用來放畫好的圈圈

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 畫布背景同背景色，使之看不見畫布

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 在畫布上畫圓圈圈

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 取得image的矩形大小

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          # 設定畫圈圈時的中心點座標為x,y

    # 可以在主程式呼叫，用來移動角色

    def update(self):

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 改變球的位置

# 設定繪圖視窗

screen = pygame.display.set\_mode((800, 600))  # 建立繪圖視窗「大小」，為 800x600

pygame.display.set\_caption("Ball Game")       # 設定繪圖視窗「標題」， My First Game

# 建立畫布

canvas = pygame.Surface(screen.get\_size())    # 設定畫布大小「大小」，等於 screen 的大小

canvas = canvas.convert()                     # 產生副本以加速顯示速度

canvas.fill((0,0,0))                          # 設定畫布背景顏色

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    # 建立全部角色群組

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 重要，計時物件

while True:  #無窮迴圈

    for event in pygame.event.get():

        if event.type == pygame.QUIT: # 使用者按右上角的關閉鈕

            pygame.quit()

            sys.exit()

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     #每秒執行30次

    # 隨機添加雪球

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 建立白色球物件，X軸隨機，Y軸為0，半徑為5，白色

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 加入全部角色群組

    # 更新所有角色的狀態

    for spr in allsprite:

        spr.update()

    screen.blit(canvas, (0,0))  # 在繪圖視窗顯示畫布          (只有背景的畫布)

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_      # 將所有角色，顯示在繪圖視窗  (圓圈圈的畫布)

    pygame.display.update()     # 更新繪圖視窗

1. **打蟑螂遊戲**
2. **自訂事件**

|  |
| --- |
| 自訂事件變數名稱 = pygame.USEREVENT # 多個事件時，需將常數USEREVENT+1pygame.time.set\_timer(自訂事件變數名稱, 毫秒) # 設定事件多久發生一次 |

import sys

import pygame

import random

# 初始化

pygame.init()

class Cockroach(pygame.sprite.Sprite):

    def \_\_init\_\_(self, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):   # 設定蟑螂屬性：x座標、y座標、圖片寬度、圖片高度

        pygame.sprite.Sprite.\_\_init\_\_(self)

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 載入圖片

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 縮放圖片

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 產生副本圖片，並保留透明背景

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 回傳目前位置

        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 以左上角為中心，設定x,y座標

# 設定繪圖視窗

screen = pygame.display.set\_mode((800, 600))  # 建立繪圖視窗「大小」，為 800x600

pygame.display.set\_caption("打蟑螂")          # 設定繪圖視窗「標題」

# 建立畫布

canvas = pygame.Surface(screen.get\_size())  # 建立畫布，大小與繪圖視窗相同

canvas = canvas.convert()                   # 建立副本，以加快載入速度

canvas.fill((255,255,255))                  # 設定畫布背景顏色：白色

# 在畫布上，新增計分板

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 預設得分為0分

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 設定字體、大小

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 設定文字內容、平滑度、文字顏色、背景顏色

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 將計分板顯示在畫布上，x座標:0;y座標:0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 設定圖片寬與高為隨機大小: 100~500 之間

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 設定圖片出現的x座位置:   0 ~ 繪圖視窗最大寬度 之間

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 設定圖片出現的y座標位置: 100 ~ 繪圖視窗最大高度 之間

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   # 產生一隻蟑螂物件

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 新增自訂事件，若有二個事件，則pygame.USEREVENT + 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 設定自訂事件多久發生一次 (單位:毫秒)

hit = None

while True: # 無窮迴圈，執行遊戲的視窗，

    for event in pygame.event.get():   # 檢查所有可能發生的事件

        if event.type == pygame.QUIT:  # 當使用者結束視窗的事件發生時，則結束程式

            pygame.quit()

            sys.exit()

        elif event.type == \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  :  # 當發生自訂事件時

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 將原本的蟑螂物件刪除，並產生新的

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 重新設定背景顏色刷，以便刪除hit

        elif event.type == pygame.MOUSEBUTTONDOWN:  # 判斷使用者是否按下滑鼠事件

            if  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 判斷是否點擊於圖片上

                \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 設定擊中蟑螂時的文字 (Hit!)

                \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 顯示Hit!

                \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  # 分數+5分

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 設定新分數

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 顯示新分數

    # 繪圖視窗上顯示

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 將畫布顯示在繪圖視窗上

    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     # 將蟑螂顯示在繪圖視窗上

    pygame.display.update()

1. **角色移動與碰撞**
2. 鍵盤偵測

|  |
| --- |
|  **from pygame.locals import \*****變數名稱 = pygame.key.get\_pressed()** **變數名稱[K\_按鍵名稱]** |

1. 時間計時器

|  |
| --- |
| **變數名稱 = pygame.time.get\_ticks()** |

1. 角色與角色碰撞(矩形檢測)，精準檢測可用pygame.sprite.collide\_rect \_ratio()

|  |
| --- |
|  **變數名稱 = pygame.sprite.collide\_rect(角色物件1,角色物件2)** |

1. 角色與角色碰撞(圓形檢測)，精準檢測可用pygame.sprite.collide\_circle \_ratio()

|  |
| --- |
|  **變數名稱 = pygame.sprite.collide\_circle(角色物件1,角色物件2)** |

1. 角色與角色碰撞(像素檢測)，精準檢測可用pygame.sprite.collide\_mask \_ratio()

|  |
| --- |
|  **變數名稱 = pygame.sprite.collide\_mask(角色物件1,角色物件2)** |

1. 角色與群組碰撞，移除值：True刪除 / False不刪除。精準檢測可用pygame.sprite.spritecollideany()

|  |
| --- |
| **變數名稱 = pygame.sprite.spritecollide(角色物件,角色群組,移除值)** |

import sys

import pygame

import random

from pygame.locals import \*

class MySprite(pygame.sprite.Sprite):

    def \_\_init\_\_(self):

        pygame.sprite.Sprite.\_\_init\_\_(self)

        self.image = None

        self.master\_image = None

        self.rect = None              # 儲存裁切後圖片，顯示的座標位置與大小

        self.old\_frame = -1           # 上一個窗格編號

        self.frame = 0                # 目前窗格編號

        self.frame\_width = 1          # 儲存裁切後圖片的寬

        self.frame\_height = 1         # 儲存裁切後圖片的高

        self.first\_frame = 0          # 第一個窗格編號

        self.last\_frame = 0           # 最後一個窗格編號

        self.columns = 1              # 圖片欄數

        self.last\_time = 0            # 儲存執行完窗格後的時間

        self.direction = 0            # 角色移動方向

        self.x = 100                  # 角色的x座標

        self.y = 100                  # 角色的y座標

        self.speed = 1.5

    def load(self, filename, width, height, columns):

        self.master\_image = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 載入圖片並產生副本

        self.frame\_width =  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定裁切後圖片的高

        self.frame\_height = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定裁切後圖片的高

        self.rect = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定裁切後圖片，顯示的座標位置與大小

        self.columns = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 設定圖片總共有幾欄數

        # 計算總共有幾個窗格

        rect = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 取得原始圖片的大小與座標位置

        self.last\_frame = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # 總窗格數 = (將原始圖片的寬 / 人物小大的寬度 取商數) - 1

    def update(self, current\_time, fps=60):

        if \_\_\_\_\_\_\_\_\_  >  \_\_\_\_\_\_\_\_\_:    # 如果當前時間大於fps(60)，則往下一個窗格編號

            \_\_\_\_\_\_\_\_\_

            if \_\_\_\_\_\_\_\_\_ > \_\_\_\_\_\_\_\_\_:       # 若目前窗格編號 > 最後一個窗格編號，則必須回到第一個窗格編號

                \_\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_

            \_\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_         # 儲存目前時間

        if self.frame != self.old\_frame:     # 若窗格編號改變時，則重新取得x,y座標

            frame\_x = (self.frame % self.columns) \* self.frame\_width        # 取得x座標，x座標規律為0,32,64,128

            frame\_y = (self.frame // self.columns) \* self.frame\_height      # 取得y座標

            rect = frame\_x, frame\_y, self.frame\_width, self.frame\_height

            self.image = self.master\_image.subsurface(rect)    # 繼續裁剪圖片

            self.old\_frame = self.frame

# 初始化

pygame.init()

# 設定繪圖視窗

screen = pygame.display.set\_mode((800, 600))  # 建立繪圖視窗「大小」，為 800x600

pygame.display.set\_caption("人物移動")       # 設定繪圖視窗「標題」

clock = pygame.time.Clock()

fps = 24

player = \_\_\_\_\_\_\_\_\_

player.load(\_\_\_\_\_\_\_\_\_)

groups = \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            sys.exit()

    clock.tick(fps)

    ticks = pygame.time.get\_ticks()

    keys = \_\_\_\_\_\_\_\_\_    # 回傳使用者按下的按鍵

    if \_\_\_\_\_\_\_\_\_: \_\_\_\_\_\_\_\_\_     # 若為esc，則離開遊戲

    elif \_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_:

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

    elif \_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_:

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

    elif \_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_:

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

    elif \_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_:

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

        \_\_\_\_\_\_\_\_\_

    else:

        \_\_\_\_\_\_\_\_\_

    player.first\_frame = \_\_\_\_\_\_\_\_\_     # 設定角色動作的第一個窗格編號 = 動作編號 \* 欄數

    player.last\_frame = \_\_\_\_\_\_\_\_\_  # 設定角色動作的最後一個窗格編號 = 第一個窗格編號+欄數-1

    if player.frame < player.first\_frame:                      # 檢查若目前窗格編號<第一個窗格的編號，則設定目前窗格為第一個窗格的編號

        player.frame = player.first\_frame

    if not player\_moving:    # 當角色沒有移動的時候，停止更新窗格數(幀幅)

        player.frame = player.first\_frame = player.last\_frame

    else:                    # 當角色移動的時候

        if player.x < 0:

            \_\_\_\_\_\_\_\_\_

        elif player.x > 800:

            \_\_\_\_\_\_\_\_\_

        if player.y < 0:

            \_\_\_\_\_\_\_\_\_

        elif player.y > 600:

            \_\_\_\_\_\_\_\_\_

        player.rect = (player.x,player.y)

    screen.fill((44, 169, 240))

    groups.update(ticks)

    groups.draw(screen)

    pygame.display.update()