PYGAME

python™

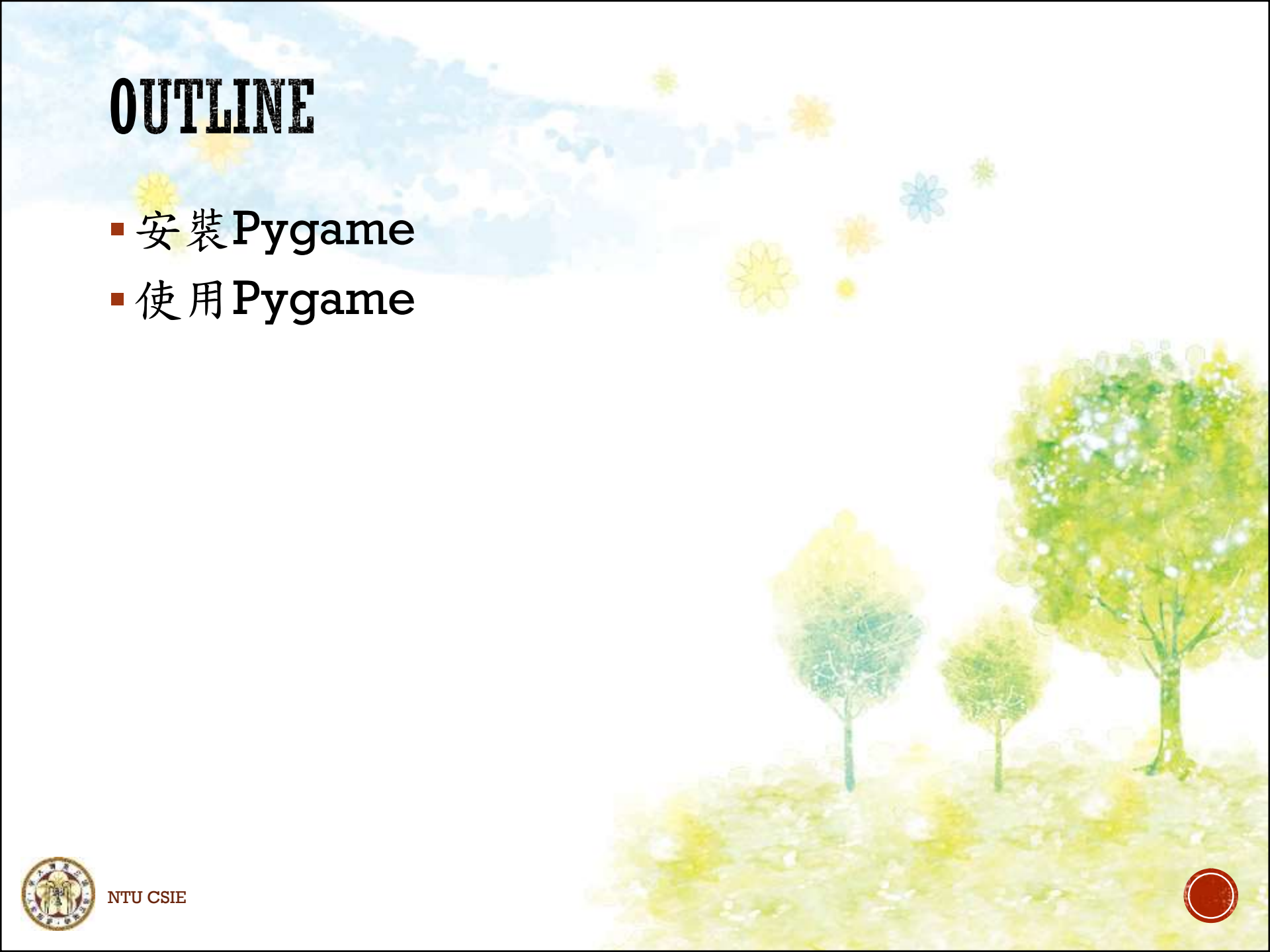
Pygame, *Games in Python – the easy way*

張傑帆 Chang, Jie-Fan

NTU CSIE

OUTLINE

- 安裝Pygame
- 使用Pygame



安裝PYGAME

- Pygame官網
- http://www.pygame.org/
- 下載適合你的python版本的pygame安裝
- 在安裝時注意路徑要設好

PYGAME

- 由於Pygame是模組庫，因此裡頭包含許多模組，以套件的方式組織，引入時可以只簡單的寫：

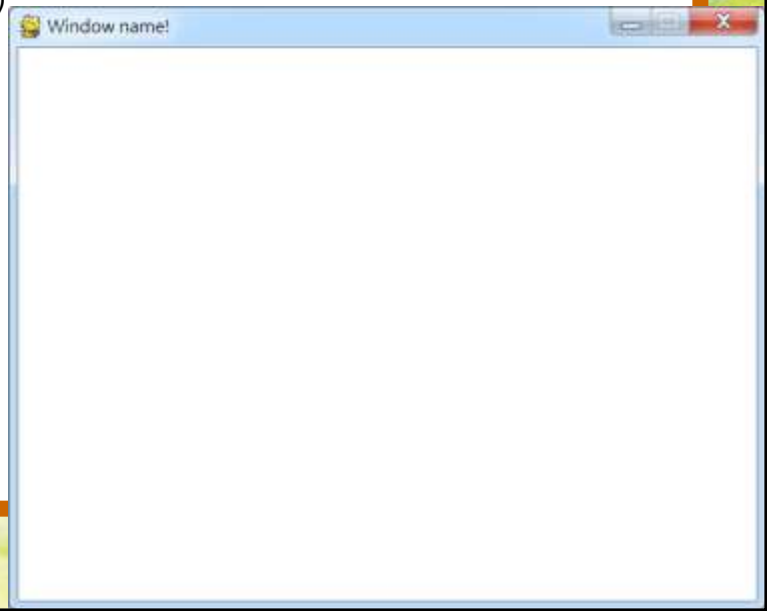
```
import pygame
```

骨架

- 基本範例，就讓我們從這開始

template00.py

```
1  from pygame import *
2  from pygame.sprite import *
3  from random import *
4
5  init()
6
7  screen = display.set_mode((640, 480))
8  display.set_caption('Window name!')
9
10 while True:
11     e = event.poll()
12     if e.type == QUIT:
13         quit()
14         break
15
16     screen.fill(Color("white"))
17     display.update()
```



template01.py

```python
import pygame
from pygame.locals import *
import os
from sys import exit

size = (800, 600)
black = (0, 0, 0)
white = (255, 255, 255)
title = "Hello, Pygame!"
chinese_message = "來寫遊戲吧！"
message = unicode(chinese_message, "big5")

def run():
    pygame.init()

    screen = pygame.display.set_mode(size, 0, 32)
    pygame.display.set_caption(title)

    font = pygame.font.Font(os.environ['SYSTEMROOT'] + "\\Fonts\\mingliu.ttc", 80)
    text = font.render(message, True, white)

    x = (size[0]-text.get_width()) / 2
    y = (size[1]-text.get_height()) / 2

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                exit()

        screen.fill(black)
        screen.blit(text, (x, y))

        pygame.display.update()

if __name__ == "__main__":
    run()
```

PYGAME

- 建立顯示的視窗
- 對Pygame模組庫初始化，使其內的模組都能夠被利用

```
def run():
    pygame.init()
```

- 建立一個screen變數

```
screen = pygame.display.set_mode(size, 0, 32)
```

  - 第一個參數指定視窗大小
  - 第二個是顯示的特殊設定，暫時不會用到，設為0
  - 第三個則是色彩位元數的設定，設定為32位元

座標與顏色

- 變數size的型態為序對，用來儲存指定視窗大小的數值

$$size = (800, 600)$$

- 將視窗大小設定為800×600
- 被指派的size變數可以作為待會真正設置視窗函數的參數，同時代表視窗的座標
- 變數black與white的型態也是序對

```
black = (0, 0, 0)
white = (255, 255, 255)
```

PYGAME

- 標題列的顯示文字

```
pygame.display.set_caption(title)
```

- 如果要在surface物件上印出文字，就需要先建立變數font來指定載入的字型

```
font = pygame.font.Font(os.environ['SYSTEMROOT'] + "\\Fonts\\mingliu.ttc", 80)
```

- font模組是專門處理字型的模組
  - 函數Font()則是用來載入字型
  - 第一個參數型態為字串，其為字型名稱。
  - os.environ['SYSTEMROOT']為系統路徑，我們接上Fonts目錄的mingliu.ttc字型檔案名稱，這是細明體的字型。
  - 第二個參數則是設定字型大小。

- 指定顯示的文字及顏色

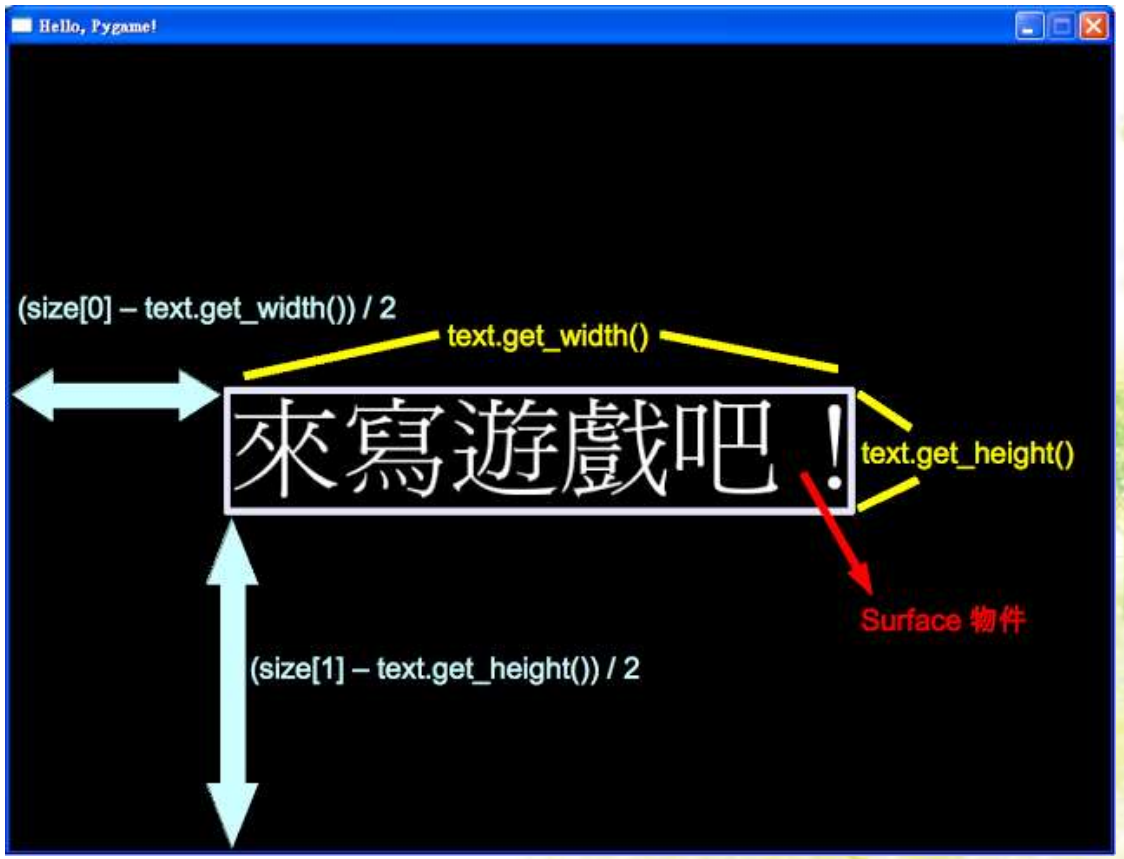
```
text = font.render(message, True, white)
```

- render方法就是將文字著色於Surface物件之上
- 第一個就是字串，我們已經把所要印出的文字儲存到變數message之中
- 第二個設為True，使文字不會印成斜體字，
- 第三個參數則是指定顏色。

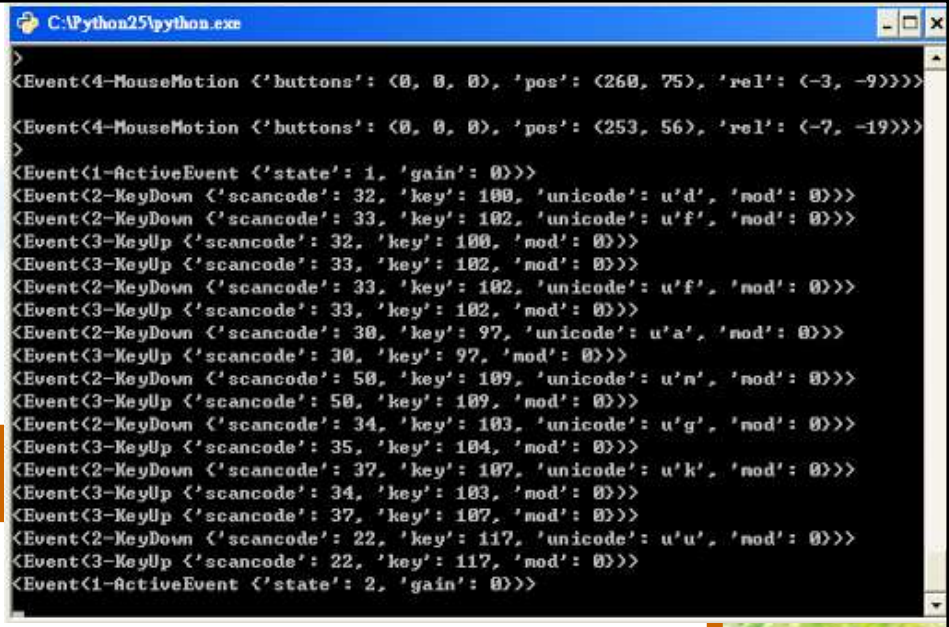
- 設定文字出現座標

```
x = (size[0]-text.get_width()) / 2
y = (size[1]-text.get_height()) / 2
```

- 顯示文字 `screen.blit(text, (x, y))`



事件測試



Event_test.py

```
import pygame
from pygame.locals import *
from sys import exit


pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption('Window name!')

while True:
    test = pygame.event.wait()
    print(test)
```

- 主要迴圈

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill(black)
    screen.blit(text, (x, y))

    pygame.display.update()
```

- 當事件的type屬性為QUIT的時候，執行exit()函數結束Pygame的視窗
- 不然while True迴圈永遠不會結束
- Surface物件利用fill()方法，充滿我們指定的黑色
- 用blit()方法，將文字轉換到視窗的物件上，第二個參數是放位置的起始座標
- update()函數更新視窗畫面

PYGAME

- 下表為我們將會用到的模組

模組名稱	描述
pygame.cursors	載入滑鼠游標的圖示。
pygame.display	控制顯示的視窗。
pygame.draw	基本的形狀繪圖。
pygame.event	事件管理。
pygame.font	載入TrueType字型。
pygame.image	載入圖形檔案。
pygame.key	鍵盤的控制。
pygame.mouse	滑鼠控制。
pygame.surface	圖形與視窗的型態。
pygame.time	管理時間與畫面更新率。

template02.py

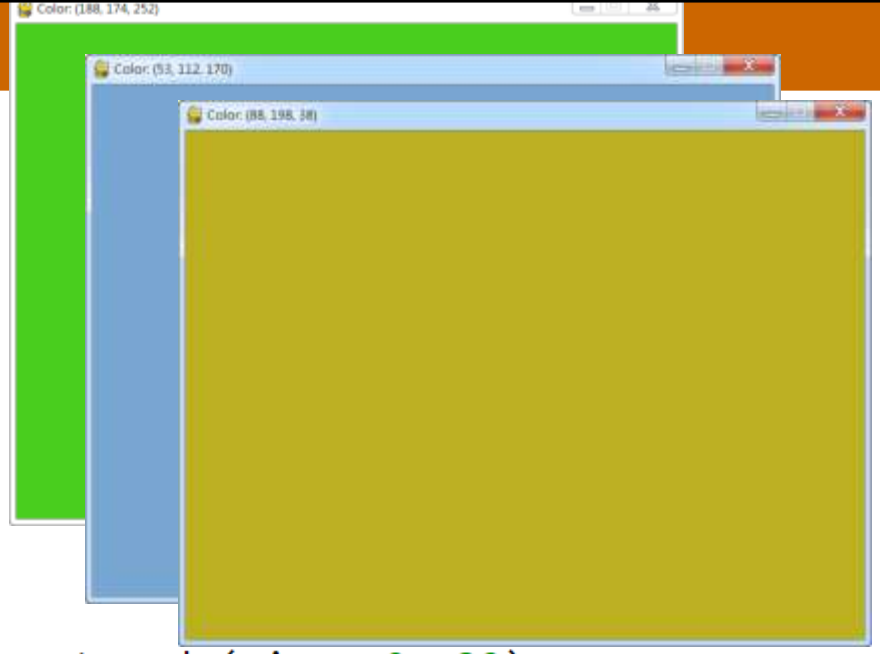
```python
import pygame
from pygame.locals import *
from sys import exit

size = (800, 600)
p1 = (0, 0)
p2 = (800, 0)
p3 = (0, 600)
p4 = (800, 600)
title = "The coordinate of 800*600 screen"
black = (0, 0, 0)
white = (255, 255, 255)

def run():
    pygame.init()

    screen = pygame.display.set_mode(size, 0, 32)
    pygame.display.set_caption(title)

    font = pygame.font.SysFont("times", 40)
    text1 = font.render(str(p1), True, white)
    text2 = font.render(str(p2), True, white)
    text3 = font.render(str(p3), True, white)
    text4 = font.render(str(p4), True, white)

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                exit()

        screen.fill(black)

        screen.blit(text1, (0, 0))
        screen.blit(text2, (p2[0]-text2.get_width(), 0))
        screen.blit(text3, (0, p3[1]-text3.get_height()))
        screen.blit(text4, (p4[0]-text4.get_width(), p4[1]-text4.get_height()))

        pygame.display.update()

if __name__ == "__main__":
    run()
```

RGB值

- 即紅綠藍，用0到255的表示色階

顏色名稱	紅色（R）	綠色（G）	藍色（B）	序對值	實際顏色
黑色	0	0	0	(0, 0, 0)	
湛藍	0	0	128	(0, 0, 128)	
綠色	0	128	0	(0, 128, 0)	
褐紅	128	0	0	(128, 0, 0)	
青綠	0	128	128	(0, 128, 128)	
紫色	128	0	128	(128, 0, 128)	
橄欖	128	128	0	(128, 128, 0)	
灰色	128	128	128	(128, 128, 128)	
紅色	255	0	0	(255, 0, 0)	
萊姆綠	0	255	0	(0, 255, 0)	
藍色	0	0	255	(0, 0, 255)	
銀灰	192	192	192	(192, 192, 192)	
紫紅	255	0	255	(255, 0, 255)	
黃色	255	255	0	(255, 255, 0)	
碧綠	0	255	255	(0, 255, 255)	
白色	255	255	255	(255, 255, 255)	

template03.py



```
import pygame
from pygame.locals import *
from sys import exit
from random import randint
from time import sleep

size = (800, 600)

def run():
    pygame.init()

    screen = pygame.display.set_mode(size, 0, 32)

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                exit()

        color = (randint(0, 255), randint(0, 255), randint(0, 255))
        screen.fill(color)
        pygame.display.set_caption("Color: " + str(color))
        sleep(1)

        pygame.display.update()

if __name__ == "__main__":
    run()
```

動畫效果

- 假如我們在while True迴圈之前加入以下

i = 0

- 迴圈裡面加入

print (i)

i += 1

```
x = (size[0]-text.get_width()) / 2
y = (size[1]-text.get_height()) / 2

i=0
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            quit()

    screen.fill(black)
    screen.blit(text, (x, y))
    pygame.display.update()

    print(i)
    i+=1
```

動畫效果

- 不斷印出遞增的變數i
- Pygame視窗依舊不變
- 維持螢幕的顯示
- 非常短的時間之內
  用非常快的速度在螢幕上輸出
- 程式只做了兩個繪圖工作
  - 第一項繪圖是讓視窗充滿黑色
  - 第二項繪圖則是把字串放置到視窗中央
- 程式就是不停的利用**while True**迴圈做這兩件繪圖工作

```
screen.fill(black)
screen.blit(text, (x, y))
pygame.display.update()
```

PYGAME

- 假如隨著迴圈逐次改變繪圖的座標位置
- 也就製造了動畫效果。
- 我們試著讓文字左右移動，到視窗的邊緣再回來

```
x = (size[0]-text.get_width()) / 2
y = (size[1]-text.get_height()) / 2

dx = 1
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            quit()

    screen.fill(black)
    screen.blit(text, (x, y))
    pygame.display.update()

    x += dx
    if (x + text.get_width()) > size[0] or x < 0:
        dx *= -1
```

- 迴圈之前加入

  dx = 1

- 迴圈裡面加入

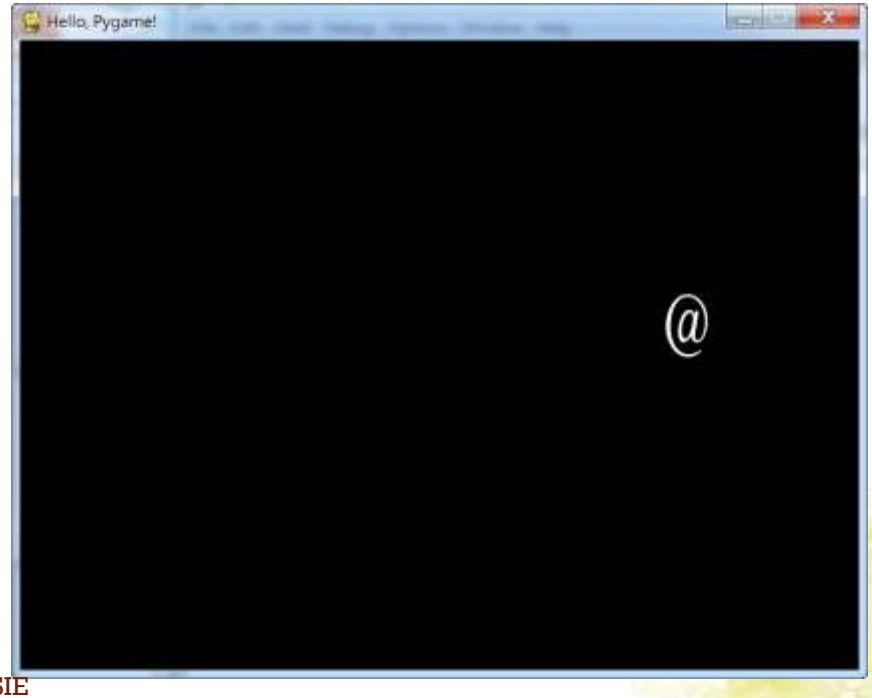
  x += dx

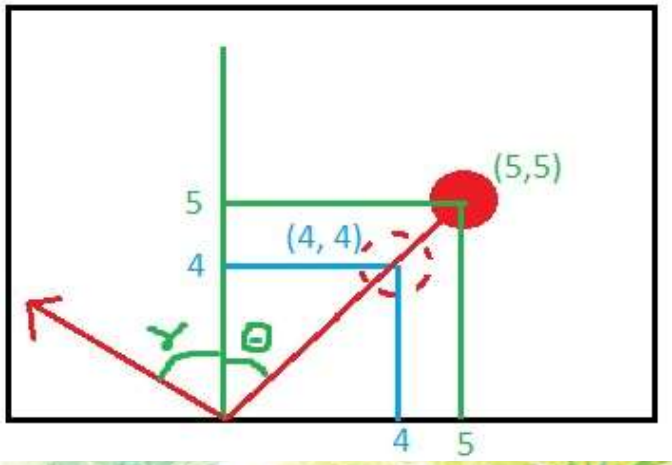
  if (x + text.get_width()) > size[0] or x < 0:

  dx *= -1

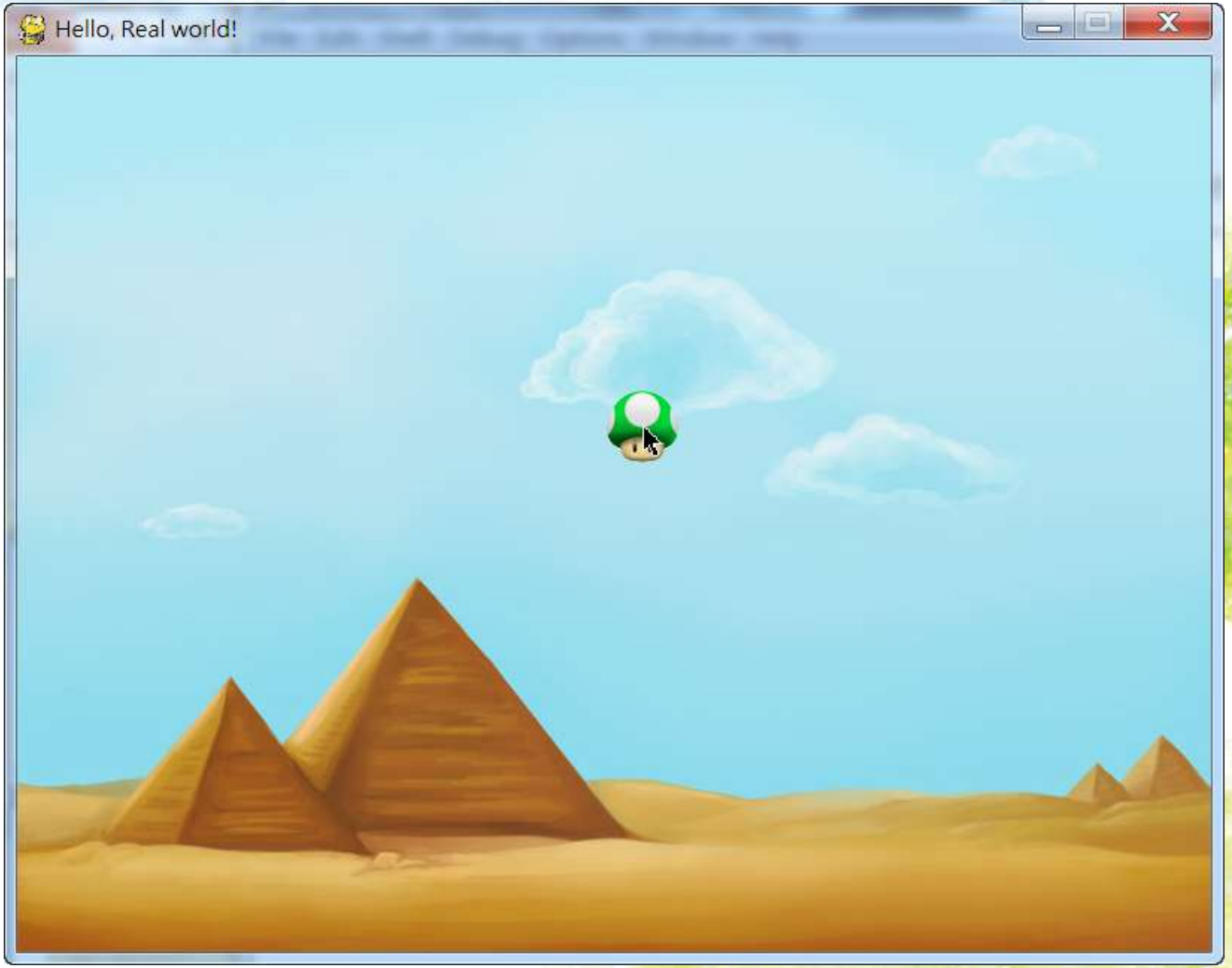
小練習

- 將文字改成一個字或符號
- 試著讓它呈對角線運動
- 碰到邊線便會反彈





載入圖片



Load_image01.py

```
import pygame
from pygame.locals import *
from sys import exit

screen_size = (800, 600)
title = "Hello, Real world!"
background_image = "background.png"
cursor_image = "cursor.png"

def mouse_pos(img):
    x, y = pygame.mouse.get_pos()
    x -= img.get_width() / 2
    y -= img.get_height() / 2
    return x, y

def run():
    pygame.init()

    screen = pygame.display.set_mode(screen_size, 0, 32)
    pygame.display.set_caption(title)

    background = pygame.image.load(background_image).convert()
    cursor = pygame.image.load(cursor_image).convert_alpha()

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                exit()

        screen.blit(background, (0, 0))
        screen.blit(cursor, mouse_pos(cursor))

        pygame.display.update()

if __name__ == "__main__":
    run()
```

PYGAME載入圖片

- 背景圖透過變數background，以image模組中的load()函數來載入圖像檔案，套用convert()方法
- 滑鼠游標透過變數cursor進行處理，同樣以load()函數載入圖像檔案，這個圖檔包括透明的影像格式，因此是用convert_alpha()轉換

```
screen = pygame.display.set_mode(screen_size, 0, 32)
pygame.display.set_caption(title)
```

- 主要迴圈中，仍需透過blit()方法將兩個圖像轉換到screen，同時須指定圖像的左上角的起始座標

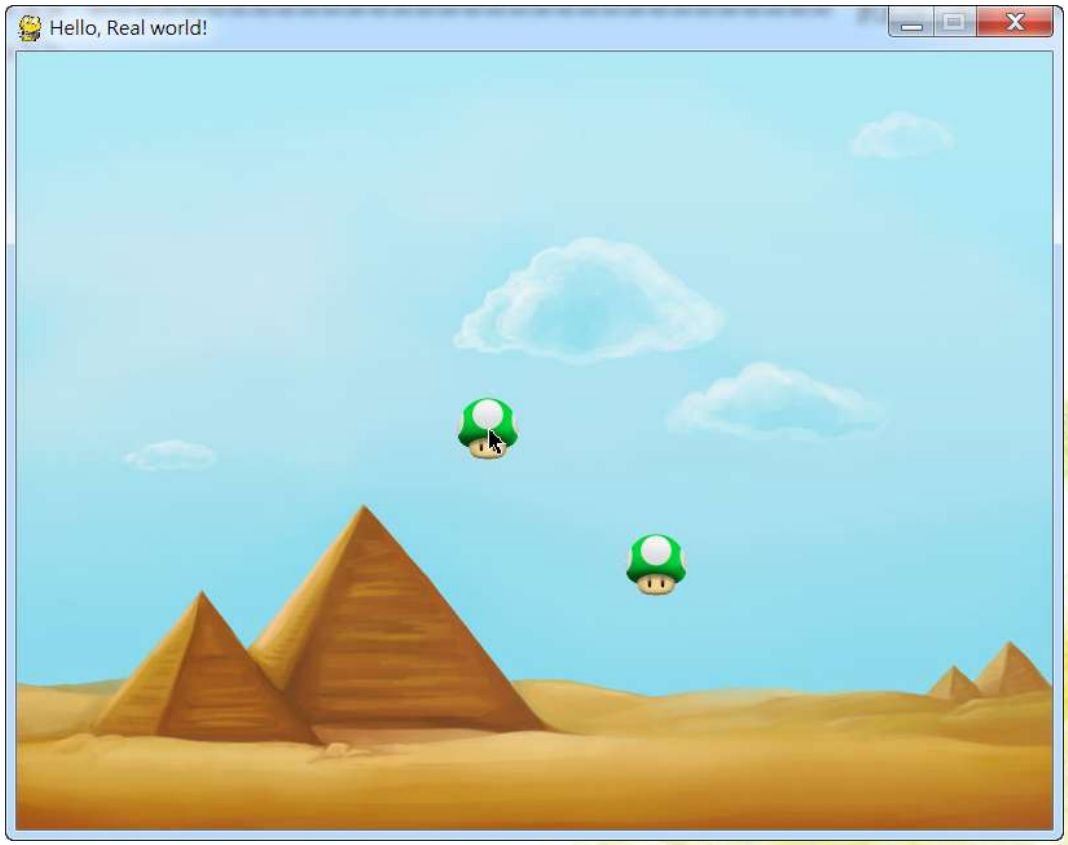
```
screen.blit(background, (0, 0))
screen.blit(cursor, mouse_pos(cursor))
```

- 滑鼠游標的位置並沒有固定，利用mouse_pos()處理滑鼠座標

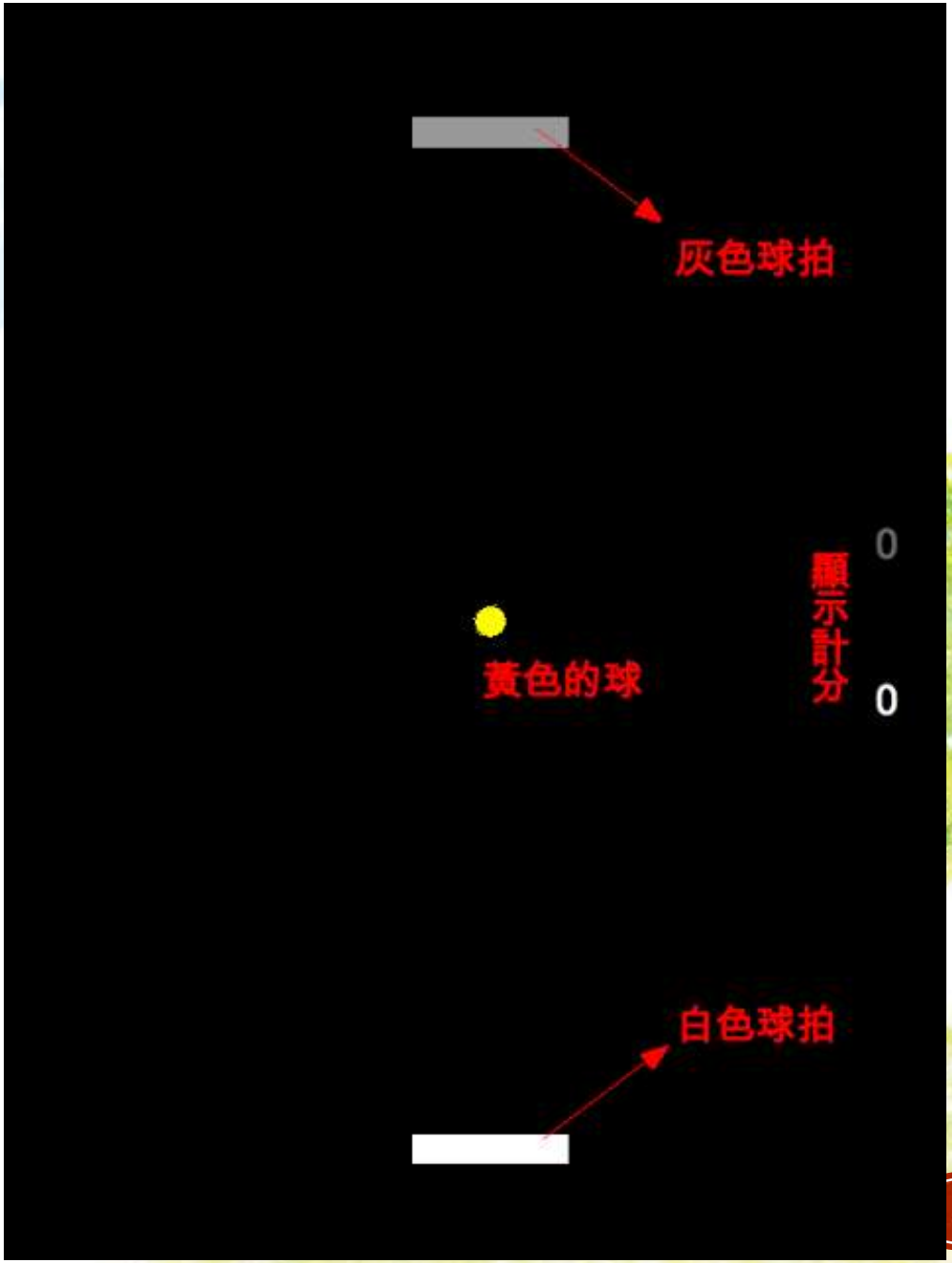
```
def mouse_pos(img):
    x, y = pygame.mouse.get_pos()
    x -= img.get_width() / 2
    y -= img.get_height() / 2
    return x, y
```

小練習

- 請加載入一小圖片
- 使其如同上一小練習一般對角運動

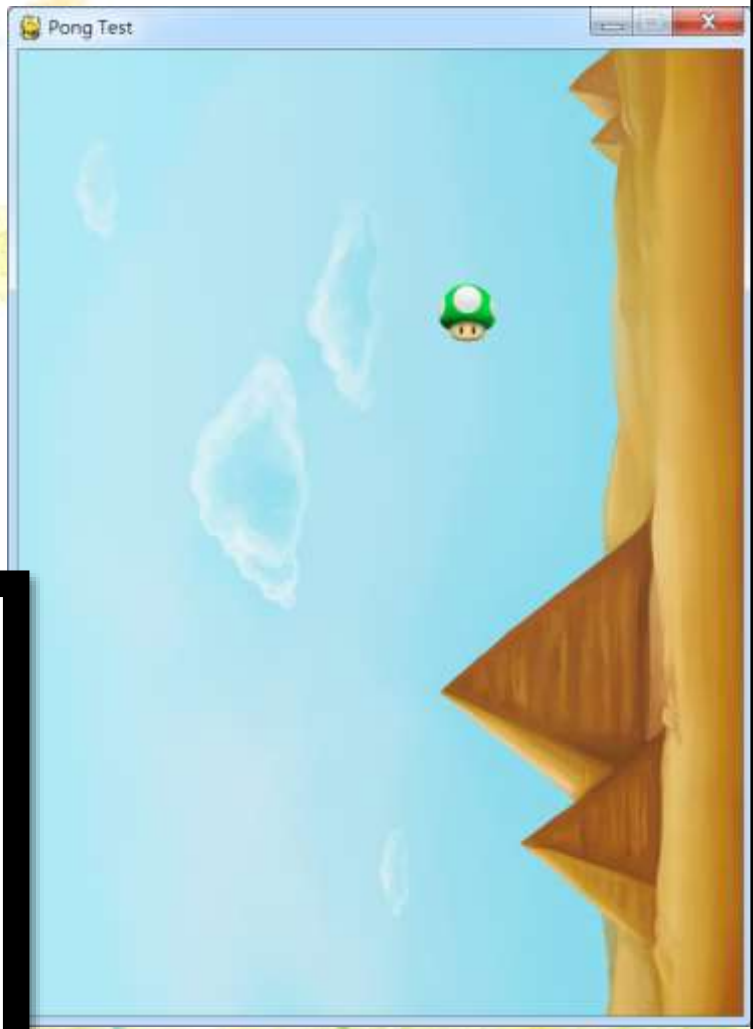


氣浮飛碟球



- 把設定座標移動改成函式
- 變數radius儲存圓的半徑
- x及y方向的變化量分別用變數dx及dy儲存
- 計算圓心座標與x、y方向變化量則交給move()處理

```
def move(point, radius, dx, dy):
    x, y = point

    x += dx
    if(x + radius)>size[0] or x < 0:
        dx*= -1

    y += dy
    if(y + radius)>size[1] or y < 0:
        dy*= -1

    return x, y, dx, dy
```



繪製圓球

- 設定座標半徑

```
ball_point = (400, 300)
radius = 10
```

- 在主迴圈中繪製圓球

```
ball = pygame.draw.circle(screen, yellow, ball_point, radius)
```

- 請將小圖改成繪製的圓球

時間因素

- 畫面與畫面間的切換，我們並不知道到底經過多少時間
- 可以利用pygame模組庫中的time模組，控制所經過的時間，然後依比例計算出位移量

```
def move(point, radius, dx, dy, sec):
    x, y = point

    x += dx*sec
    if(x + radius)>size[0] or x < 0:
        dx*= -1

    y += dy*sec
    if(y + radius)>size[1] or y < 0:
        dy*= -1

    return x, y, dx, dy
```

- 回圈前

clock = pygame.time.Clock()

- 回圈前

seconds = clock.tick(30) / 1.0

x, y, dx, dy = move(ball_point, radius, dx, dy, seconds)

加入白色球拍

- 設定一些初始值

```
side = (80, 12)
bat_point = (260, 740)
bat_speed = 1
```

- side用作球拍的長方形Rect物件的邊長
- bat_point則是左上角座標
- bat_speed則是球拍移動的速度
- 畫出球拍的程式碼要加入迴圈之中，如下。

```
bat = pygame.draw.rect(screen, white, Rect(bat_point, side))
```

移動球拍

- 等同於改變bat_point的值
- 用鍵盤的左右兩個方向鍵進行控制
- 用pressed kev記錄從鍵盤按下了什麼鍵

```
pressed_key = pygame.key.get_pressed()
```

- 用個batcontrol()函數控制移動

```
def batcontrol(key, point, speed, side, seconds):
    x, y = point

    if key[K_LEFT]:
        x -= speed * seconds
    if x < 0:
        x = 0
    elif key[K_RIGHT]:
        x += speed * seconds
    if x + side[0] > size[0]:
        x = size[0] - side[0]

    return x, y
```

- 總共需要五個參數，我們在遊戲的主要迴圈呼叫如下

```
bat_point = batcontrol(pressed_key, bat_point, bat_speed, side, seconds)
```

用球拍擊球

- 如何製造看起像是用球拍擊球的效果呢？
- Rect物件有內建的colliderect()方法處理兩個Rect物件碰撞的問題

```
if bat.colliderect(ball):
    x, y, dx, dy = rebound(ball_point, dx, dy, seconds)
else:
    x, y, dx, dy = move(ball_point, radius, dx, dy, seconds)
```

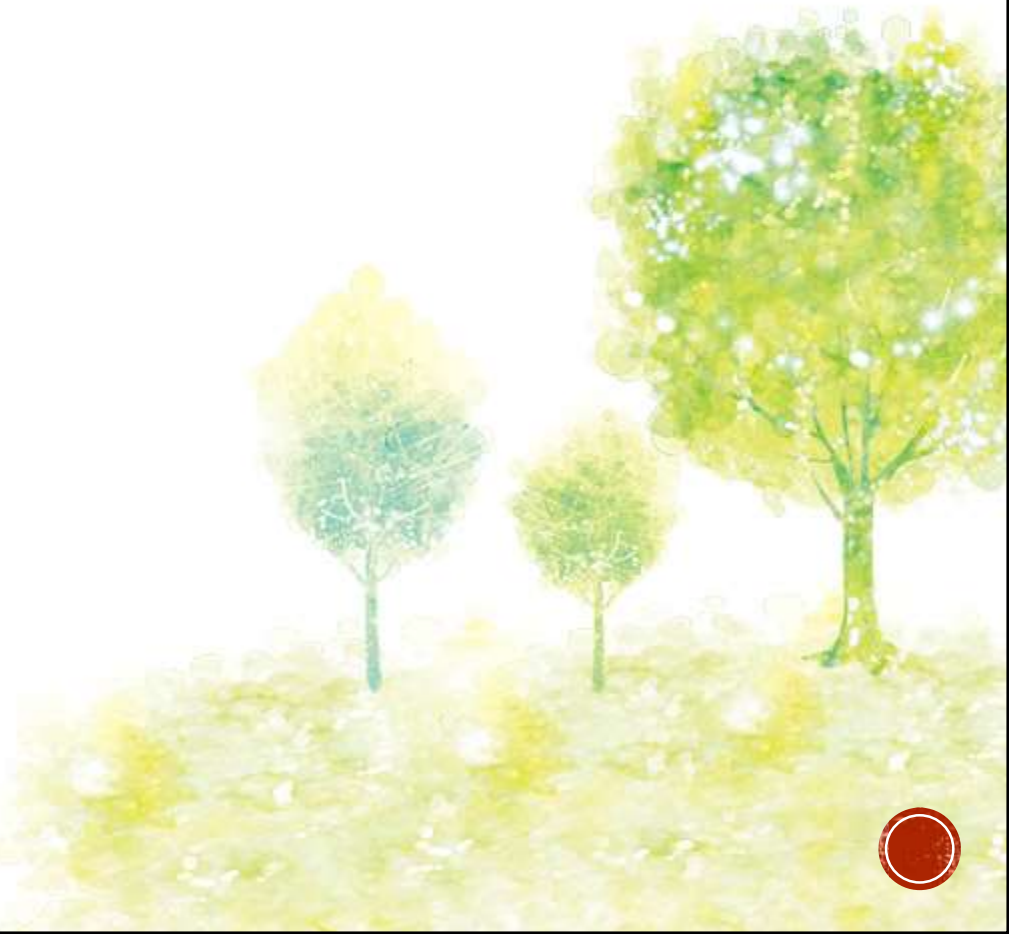
- 我們還需要一個rebound()函數，來處理兩個Rect物件碰撞後的情況

```
def rebound(point, dx, dy, seconds):
    x, y = point
    #dx *=-1
    dy *=-1
    #x += dx*5
    y += dy*5
    return x, y, dx, dy
```


Pong Test

回家練習

- 請加入另一球拍使雙方可以互打



PYGAME TUTORIAL

- Tutorial：
  http://www.pygame.org/docs/
- Example(Recommended)：
  http://www.pygame.org/docs/tut/chimp/ChimpLineByLine.html
- Cheat sheet(常用指令筆記)：
  http://inventwithpython.com/pygamecheatsheet.png
- 程式語言教學誌：
  http://pydoing.blogspot.tw/