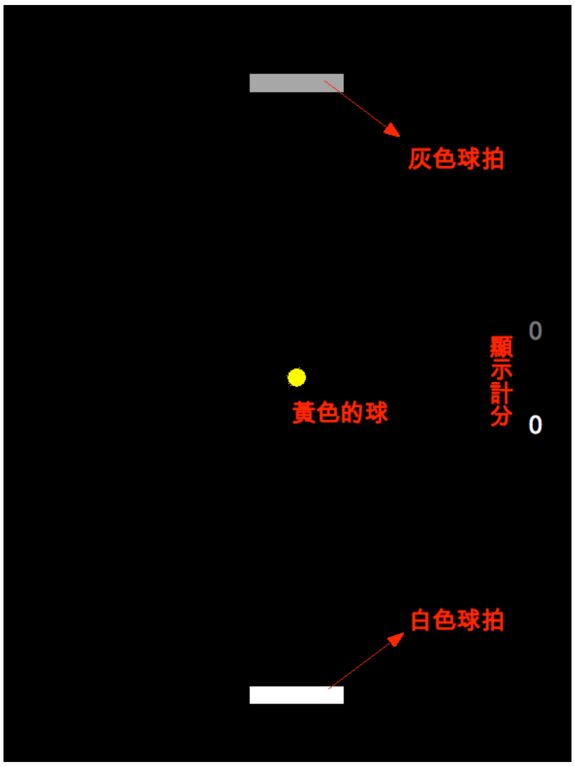
第十一章 乒乓球

桌球，又稱乒乓球，這是一種廣為人知的運動，也是早期實作出的商業電動遊戲之一，在這一章裡，我們嘗試簡化的版本，初步規劃如下圖。



視窗設定成600×800的大小，球台以黑色作為背景，黃色的球，上方灰色的球拍與下方白色的球拍，右方顯示分數。遊戲規則簡單一點，不考慮一局一局的發球回合，球由中央朝隨機方向發出後，到視窗的四個邊緣都會彈回，球拍可以直接把球擊出。如果球落在球拍後方的視窗邊緣，對手就得一分。

我們逐步發展這個程式，從模擬球的移動到將灰色的球拍交給電腦自行控制。

模擬球的移動

我們利用draw模組中的circle()函數畫出球，然後每一次重新繪圖時，球的圓心x座標及y座標各遞增1，如此產生效果彷彿平面上的球在移動，程式碼如下。

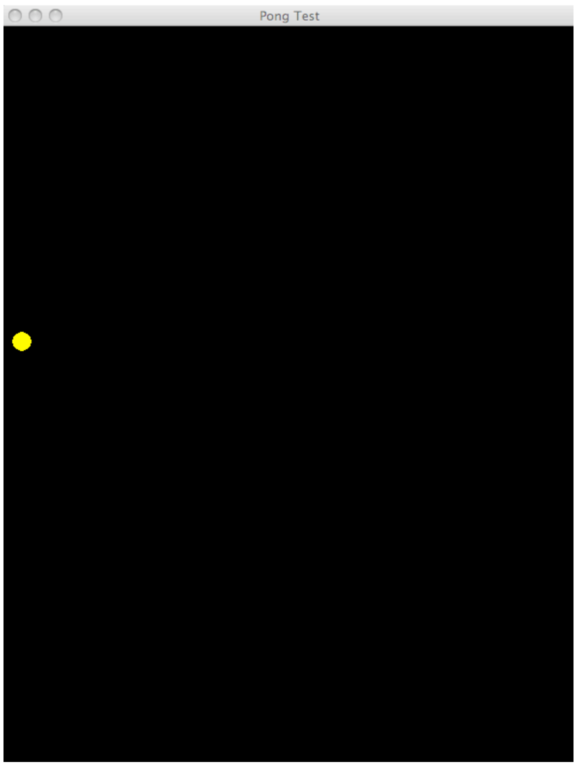
```
import pygame
from pygame.locals import *

from sys import exit

size = (600, 800)
title = "Pong Test"
black = (0, 0, 0)
yellow = (255, 255, 0)

def move(point, radius, dx, dy):
    x, y = point
    if x > 0 + radius:
        x += dx
    if x < 0 + radius:
        dx *= -1
        x = 0 + radius

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

    if y > 0 + radius:
        y += dy
    if y < 0 + radius:
        dy *= -1
        y = 0 + radius

    if y < size[1] - radius:
        y += dy
    if y > size[1] - radius:
        dy *= -1
        y = size[1] - radius

    return x, y, dx, dy

def run():
    pygame.init()

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

    ball_point = (400, 300)
    radius = 10

    dx, dy = 1, 1

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

        screen.fill(black)

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

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

        pygame.display.update()

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

在run()函數中，我們用變數ball_point儲存圓心座標，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，進到遊戲的主要迴圈後，第一次繪圖輸出後，每一次重新繪圖前實際計算圓心座標與x、y方向的變化量則交給move()函數處理。

move()函數需要四個參數，分別是圓心座標ball_point、半徑radius以及x及y方向的變化量dx與dy。ball_point在move()函數中會被分別拆成x與y座標，在我們所規劃的球台範圍內，也就是x

大於0加上半徑，小於size[0]減掉半徑，y亦同，大於0加上半徑，小於size[1]減掉半徑之中，x遞增dx而y遞增dy。

如果x或y超出這個範圍，dx或dy個別改變正負號，使球往反方向前進。最後，move()函數回傳x、y座標值，以及在正值與負值之間來回跳動的dx與dy，我們實際來看看模擬的效果吧！



隨機方向

當我們把dx與dy都先指派為1時，程式一開始執行，也就是發球的同時，球總是往右下的方向前進，而且每一次執行球的軌跡都一樣。我們可以利用以下的函數，讓發出的球朝隨機的角度。

```
def initial():
    if randint(0, 1):
        dx = randint(0, 2)
    else:
        dx = -randint(0, 2)

    if randint(0, 1):
        dy = randint(1, 2)
    else:
        dy = -randint(1, 2)

    return dx, dy
```

這個函數借助標準模組庫中random模組的randint()函數，所以在程式的開頭不要忘了加入

```
from random import randint
```

dx的初始值設定為-2、-1、0、1、2其中之一，dy初始值設定為為-2、-1、1、2其中之一，提供發球有二十種可能的方向，若dx初始值為0，球依dy為正或負，直接往垂直方向上或下前進。

而在run()函數內，指派dx、dy初值的陳述要更改如下。

```
dx, dy = initial()
```

時間因素

但是這樣仍有一些缺點，在我們自己的電腦上看起來也許效果還可以，但是換到其他的電腦上可能會有所不同，尤其是速度較快的電腦與速度較慢的電腦之間的轉移。這樣的的問題主要是每一次重新繪圖，也就是畫面與畫面間的切換，我們並不知道到底經過多少時間。

我們可以利用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。

```
import pygame
from pygame import *

from sys import exit
from random import randint

size = (600, 800)
title = "Pong Test"
black = (0, 0, 0)
white = (255, 255, 255)
yellow = (255, 255, 0)

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

    if x > 0 + radius:
        x += dx * seconds
    if x < 0 + radius:
        dx *= -1
        x = 0 + radius

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

    if y > 0 + radius:
        y += dy * seconds
    if y < 0 + radius:
        dy *= -1
        y = 0 + radius

    if y < size[1] - radius:
        y += dy * seconds
    if y > size[1] - radius:
        dy *= -1
        y = size[1] - radius

    return x, y, dx, dy

def initial():
    if randint(0, 1):
        sx = 1
    else:
        sx = -1

    if randint(0, 1):
        sy = 1
    else:
        sy = -1

    return sx, sy

def speed():
    return float(randint(50, 100))

def run():
    pygame.init()

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

    point = (300, 400)
    radius = 10

    sign_x, sign_y = initial()
    dx = sign_x * speed()
    dy = sign_y * speed()

    clock = pygame.time.Clock()

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

        screen.fill(black)
        seconds = clock.tick(30) / 1000.0

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

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

        pygame.display.update()

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

在新的程式裡，initial()函數只有回傳+1或-1給sign_x與sign_y兩個變數，然後dx與dy分別為sign_x及sign_y乘上speed()函數。speed()函數只是簡單的利用randint()函數找出一個整數，然後用float()函數使其成為浮點數型態，事實上，speed()函數我們所要設定的是每一秒的位移量，正如其名，英文speed就是中文速度之意。

然後建立一個變數clock，其為Clock型態，用來記錄時間。進入遊戲的主要迴圈，clock使用tick()方法，並用30作為參數，這可以得到每秒畫面更新不超過30次的時間，單位為千分之一秒，所以除以1000.0得到秒數。

一般來說，這會得到1除以30等於0.033秒，然而這個程式不會佔去所有的CPU時間，因為還有作業系統或是其他程式在等待進入CPU，所以0.033是畫面更新的最小秒數。變數seconds獲得每一次畫面更新的時間，因而要作為計算圓心座標move()函數的參數之一。

move()函數中，x或y的位移量是dx或dy乘上seconds，這是由於dx或dy是速度，而seconds是秒數，速度乘以時間就等於每一次畫面更新的位移量。

加入白色球拍

加入白色的球拍很簡單，我們需要在run()函數先設定一些初始值。

```
side = (80, 12)
bat_point = (260, 740)
bat_speed = 150
```

變數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「while True:」迴圈之中，如下。

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

要移動球拍，也就是等同於改變bat_point的值，這裡，我們希望用鍵盤的左右兩個方向鍵進行控制，於是用變數pressed_key記錄從鍵盤按下了什麼鍵。

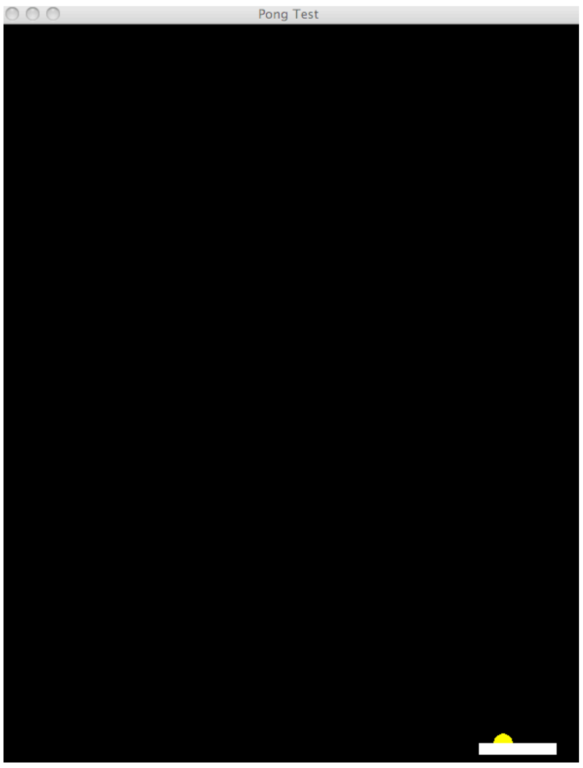
```
pressed_key = pygame.key.get_pressed()
```

然後用另一個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)
```



batcontrol()函數所回傳的就是作為球拍長方形Rect物件的起始座標。這個函數是如何作用的呢，首先將bat_point的值拆成x與y，函數裡只會變動x的值，y值保持不變。如果使用者按下左方向鍵，x會依次在每個畫面遞減bat_speed乘以seconds，這就是球拍向左的位移量，如果x最後遞減小於0，x就設為0，使球拍不會超出球台範圍。

如果使用者按得是右方向鍵，球拍以類似的方式往右移動，如果x的值加上球拍的長度大於球台的最左側，也就是x+side[0]大於size[0]，x就設為size[0]-side[0]。

來試看看吧！

用球拍擊球

有沒有發現嘗試用球拍接球，球會直接穿過球拍，好像球從球拍後面漏掉一樣，這是為什麼呢？因為我們的程式是對球跟球拍處理各自的繪圖，因而當球跟球拍的座標重複時，兩者產生的情況變程式重疊顯示，而球的座標一直在改變，導致看起來像是球穿過球拍一樣。

我們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，然而我們需要注意，這個colliderect()方法是個布林函數，只會回傳真，及兩個Rect物件相遇，或是假，也就是兩個Rect物件沒有相遇。

不過我們還需要一個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

    if randint(0, 1):
        dx *= -1.0
    else:
        dx *= 1.0
    dy *= -1
    x += dx * (seconds + 0.1)
    y += dy * (seconds + 0.1)

    return x, y, dx, dy
```

總共需要四個參數，point為球的圓心座標，其餘則是dx、dy與seconds。在rebound()函數內，二分之一的機率dx會改變正負值，如果dx改變正負值，球就會往球拍的方向反彈，而如果正負值沒有改變，x會持續遞增或遞減，就像到球台邊緣的反彈一般。

最後rebound()函數同move()函數回傳x、y、dx及dy。而在主要的遊戲迴圈內，碰撞需要以下的程式碼處理。
```
if bat.colliderect(ball):
    ball_x, ball_y, ball_dx, ball_dy = \
                        rebound(ball_point, ball_dx, ball_dy, seconds)
else:
    ball_x, ball_y, ball_dx, ball_dy = \
                   move(ball_point, radius, ball_dx, ball_dy, seconds)
```

顯示計分

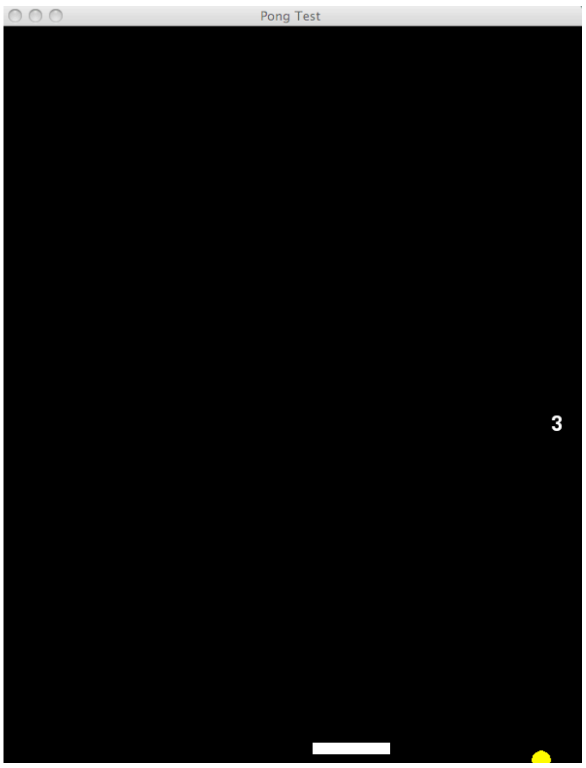
我們要處理計分，在run()函數先加入以下的設定。
```
score_point = (size[0]-text.get_width()-20, size[1]/2)
score = 0
font = pygame.font.SysFont("arial", 32)
```

變數score_point為分數顯示的位置，score累計分數，同樣的，顯示文字就樣先建立Font型態的變數font。

假設接到球就加一分，我們直接在「while True:」迴圈內作分數累計。
```
if bat.colliderect(ball):
    ball_x, ball_y, ball_dx, ball_dy = \
                        rebound(ball_point, ball_dx, ball_dy, seconds)
    score += 1
else:
    ball_x, ball_y, ball_dx, ball_dy = \
                   move(ball_point, radius, ball_dx, ball_dy, seconds)
```

當然，也要作顯示分數的繪圖工作。
```
text = font.render(str(score), True, white)
screen.blit(text, score_point)
```

來看看效果吧！



加入灰色球拍

加入灰色球拍如同加入白色球拍一樣，主要是把視窗下方的座標改成上方的座標，比較令人煩惱的，就是如何讓電腦自行控制接球。

其實這就如同batcontrol()函數一般的計算座標，我們定義另一個playercontrol()函數來計算灰色球拍的座標。

```
def playercontrol(player_point, ball_point, speed, side, seconds):
    player_x, player_y = player_point
    ball_x, ball_y = ball_point

    if ball_y < 400:
        if player_x < ball_x + side[0]:
            player_x += speed * seconds

            if player_x < 0:
                player_x = 0
            elif player_x + side[0] > size[0]:
                player_x = size[0] - side[0]

        if player_x > ball_x:
            player_x -= speed * seconds

            if player_x < 0:
                player_x = 0
            elif player_x + side[0] > size[0]:
                player_x = size[0] - side[0]

    return player_x, player_y
```

五個參數中，player_point為灰色球拍的座標，ball_point則是球的座標，在函數中player_point及ball_point分別被拆成x及y兩個變數。我們假設球通過球台的一半，也就是ball_y大於400時，灰色球拍才進行接球的判斷。

怎麼作判斷的呢？球拍的x座標與球的x座標加上球拍長度比較，如果球拍的x座標小於球的x座標加上球拍長度，這是說球拍現階段在球的左方，因而球拍向右移動，又如果球拍的x座標大於球的x座標加上球拍長度，球拍就要向左移動。

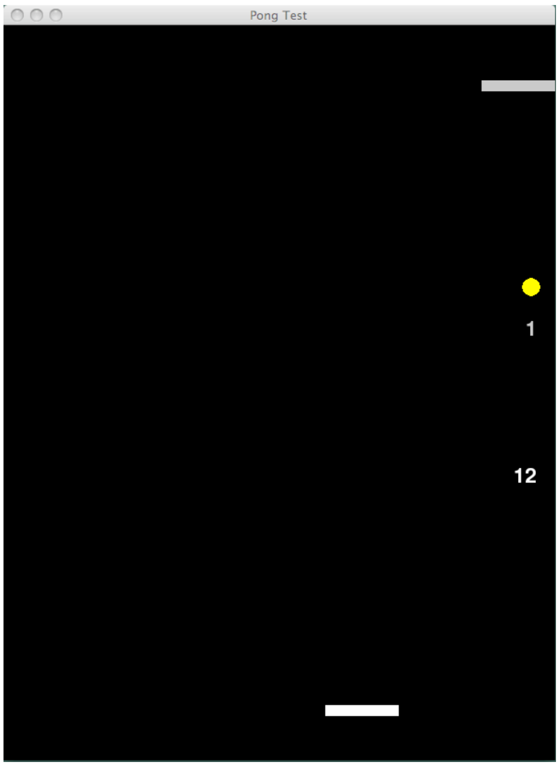
當然我們不希望球拍移動超出球台範圍，所以假如player_x小於0就將player_x設為0，又如果player_x加上side[0]大於size[0]，player_x則設為size[0]-size[0]。最後函數回傳的就是調整過的灰色球拍的座標。

至於分數就得交給move()函數處理，這裡我們把move()函數更名為ballcontrol()函數。
```
def ballcontrol(point, radius, dx, dy, seconds, bat_score, player_score):
    x, y = point

    if x > 0 + radius:
        x += dx * seconds
    if x < 0 + radius:
        dx *= -1
        x = 0 + radius

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

    if y > 0 + radius:
        y += dy * seconds
    if y < 0 + radius:
        dy *= -1
        y = 0 + radius
        bat_score += 1

    if y < size[1] - radius:
        y += dy * seconds
    if y > size[1] - radius:
        dy *= -1
        y = size[1] - radius
        player_score += 1

    return x, y, dx, dy, bat_score, player_score
```
白色球拍的分數由變數bat_score記錄，灰色球拍則由變數player_score記錄，我們會發現計分的程式控制很簡單，當y小於0加半徑，bat_score遞增1，也就是白色球拍加一分，而當y大於size[1]加半徑，player_score遞增1，由電腦控制的灰色球拍就加了一分。

到這部份為止的程式碼整理如下。
```
import pygame
from pygame.locals import *

from sys import exit
from random import randint

size = (600, 800)
title = "Pong Test"

black = (0, 0, 0)
gray = (192, 192, 192)
```

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

def ballcontrol(point, radius, dx, dy, seconds, bat_score, player_score):
    x, y = point

    if x > 0 + radius:
        x += dx * seconds
    if x < 0 + radius:
        dx *= -1
        x = 0 + radius

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

    if y > 0 + radius:
        y += dy * seconds
    if y < 0 + radius:
        dy *= -1
        y = 0 + radius
        bat_score += 1

    if y < size[1] - radius:
        y += dy * seconds
    if y > size[1] - radius:
        dy *= -1
        y = size[1] - radius
        player_score += 1

    return x, y, dx, dy, bat_score, player_score

def rebound(point, dx, dy, seconds):
    x, y = point

    if randint(0, 1):
        dx *= -1.0
    else:
        dx *= 1.0

    dy *= -1

    x += dx * (seconds + 0.1)
    y += dy * (seconds + 0.1)

    return x, y, dx, dy


def initial():
    if randint(0, 1):
        sx = 1
    else:
        sx = -1

    if randint(0, 1):
        sy = 1
    else:
        sy = -1
    return sx, sy

def speed():
    return float(randint(50, 100))

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

def playercontrol(player_point, ball_point, speed, side, seconds):
    player_x, player_y = player_point
    ball_x, ball_y = ball_point

    if ball_y < 400:
        if player_x < ball_x + side[0]:
            player_x += speed * seconds

            if player_x < 0:
                player_x = 0
            elif player_x + side[0] > size[0]:
                player_x = size[0] - side[0]

        if player_x > ball_x:
            player_x -= speed * seconds

            if player_x < 0:
                player_x = 0
            elif player_x + side[0] > size[0]:
                player_x = size[0] - side[0]


    return player_x, player_y

def run():
    pygame.init()

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

    ball_point = (300, 400)
    ball_x, ball_y = ball_point
    radius = 10.

    side = (80, 12)
    bat_point = (260, 740)
    player_point = (260, 60)
    bat_speed = 150

    sign_x, sign_y = initial()
    ball_dx = sign_x * speed()
    ball_dy = sign_y * speed()

    score_point = (700, 500)
    bat_score = 0
    player_score = 0
    font = pygame.font.SysFont("arial", 32)

    clock = pygame.time.Clock()

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

        screen.fill(black)
        seconds = clock.tick(30) / 1000.0

        ball = pygame.draw.circle(screen, yello, ball_point, radius)
        bat = pygame.draw.rect(screen, white, Rect(bat_point, side))
        player = pygame.draw.rect(screen, gray, Rect(player_point, side))

        player_x, player_y = player_point

        bat_text = font.render(str(bat_score), True, white)
        player_text = font.render(str(player_score), True, gray)
        screen.blit(bat_text, (size[0]-bat_text.get_width()-20, \
                                                   size[1]/2+80))
        screen.blit(player_text, \
                      (size[0]-player_text.get_width()-20, size[1]/2-80))

        # ball control
        if bat.colliderect(ball):
            ball_x, ball_y, ball_dx, ball_dy = \
                           rebound(ball_point, ball_dx, ball_dy, seconds)
        elif player.colliderect(ball):
            ball_x, ball_y, ball_dx, ball_dy = \
                           rebound(ball_point, ball_dx, ball_dy, seconds)
        else:
            ball_x, ball_y, ball_dx, ball_dy, bat_score, player_score = \
                       ballcontrol(ball_point, radius, ball_dx, ball_dy,\
                                         seconds, bat_score, player_score)

        ball_point = (ball_x, ball_y)

        # user's bat
        pressed_key = pygame.key.get_pressed()
        bat_point = batcontrol(pressed_key, bat_point, bat_speed, \
                                                    side, seconds)

        # artificial intelligence
        player_point = playercontrol(player_point, ball_point, \
                                         bat_speed, side, seconds)

        pygame.display.update()

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

來試看看吧！



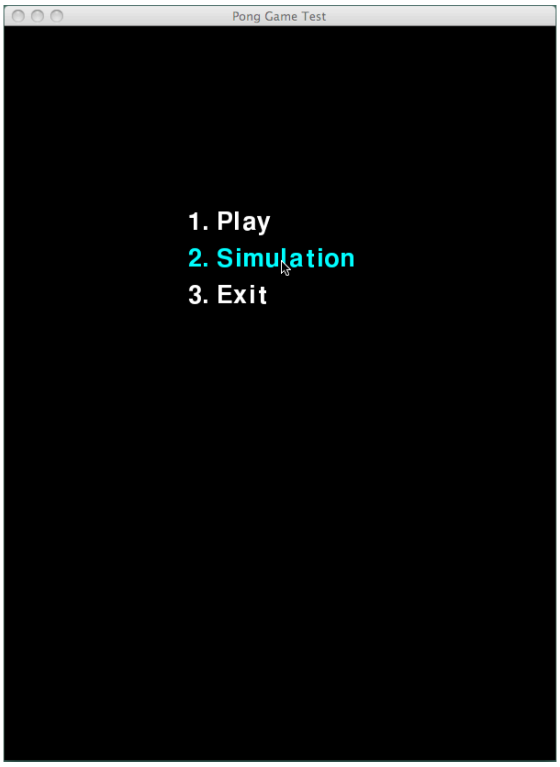
加入選單

嗯，灰色球拍有點笨，不過的確會去接球。我們可以試者寫出另一個playercontrol()函數，來控制白色球拍，讓電腦自己跟自己模擬這個乒乓球遊戲。等一等，我們先來作一個進入遊戲的選單程式。

選單程式要怎麼寫呢？所有的程式控制其實有點像我們寫過七巧板程式，顯示選項文字，滑鼠移動到選項文字的上方隨之改變顏色，如果在某個選項上按下滑鼠右鍵，就進行該選項的功能。

我們暫時不考慮點擊滑鼠右鍵的事情，選單程式舉例如下。

```
import pygame
from pygame.locals import *

from sys import exit
from random import randint

size = (600, 800)
title = "Pong Game Test"

black = (0, 0, 0)
gray = (128, 128, 128)
white = (255, 255, 255)
yellow = (255, 255, 0)
aqua = (0, 255, 255)

def run():
    pygame.init()

    screen = pygame.display.set_mode(size, 0, 32)
    pygame.display.set_caption(title)
    prompt1 = "1. Play"
    prompt2 = "2. Simulation"
    prompt3 = "3. Exit"
    prompt4 = """ """

    font1 = pygame.font.SysFont("arial", 40)
    clock = pygame.time.Clock()

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

        screen.fill(black)

        text1 = font1.render(prompt1, True, white)
        text1_1 = font1.render(prompt1, True, aqua)
        text2 = font1.render(prompt2, True, white)
        text2_1 = font1.render(prompt2, True, aqua)
        text3 = font1.render(prompt3, True, white)
        text3_1 = font1.render(prompt3, True, aqua)

        x, y = pygame.mouse.get_pos()

        if 200 <= x <= 200 + text1.get_width() and \
                                  200 <= y <= 200 + text1.get_height():
            screen.blit(text1_1, (200, 200))
            screen.blit(text2, (200, 240))
            screen.blit(text3, (200, 280))
        elif 200 <= x <= 200 + text2.get_width() and \
                                  240 <= y <= 240 + text1.get_height():
            screen.blit(text1, (200, 200))
            screen.blit(text2_1, (200, 240))
            screen.blit(text3, (200, 280))
        elif 200 <= x <= 200 + text3.get_width() and \
                                  280 <= y <= 280 + text1.get_height():
            screen.blit(text1, (200, 200))
            screen.blit(text2, (200, 240))
            screen.blit(text3_1, (200, 280))
        else:
            screen.blit(text1, (200, 200))
            screen.blit(text2, (200, 240))
            screen.blit(text3, (200, 280))

        pygame.display.update()

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



執行程式會顯示如右的結果。

如何在按下滑鼠右鍵後進行另一個功能呢？簡單一點的解決方法，把選單程式中的run()函數更名為menu()函數，使用者控制的程式也更名為userplay()函數，這兩個函數都各自包含一個「while True:」迴圈，而在menu()函數裡依需求呼叫userplay()函數，在userplay()函數中按下ESC鍵，就會回到menu()函數，也就是呼叫menu()函數。

我們另外寫一個電腦模擬的simulateplay()函數加入選

單，程式碼如下，請自行參考。

```
import pygame
from pygame.locals import *

from sys import exit
from random import randint

size = (600, 800)
title = "Pong Game Test"

black = (0, 0, 0)
gray = (128, 128, 128)
white = (255, 255, 255)
yellow = (255, 255, 0)
aqua = (0, 255, 255)

prompt = {1:"1. Play", 2:"2. Simulation", 3:"3. Exit"}

def ballcontrol(point, radius, dx, dy, seconds, bat_score, player_score):
    x, y = point

    if x > 0 + radius:
        x += dx * seconds
    if x < 0 + radius:
        dx *= -1
        x = 0 + radius

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

    if y > 0 + radius:
        y += dy * seconds
    if y < 0 + radius:
        dy *= -1
        y = 0 + radius
        bat_score += 1

    if y < size[1] - radius:
        y += dy * seconds
    if y > size[1] - radius:
        dy *= -1
        y = size[1] - radius
        player_score += 1

    return x, y, dx, dy, bat_score, player_score

def rebound(point, dx, dy, seconds):
    x, y = point

    if randint(0, 1):
        dx *= -1.0
    else:
        dx *= 1.0

    dy *= -1

    x += dx * (seconds + 0.1)
```

```
        y += dy * (seconds + 0.1)

        return x, y, dx, dy

    def initial():
        if randint(0, 1):
            sx = 1
        else:
            sx = -1

        if randint(0, 1):
            sy = 1
        else:
            sy = -1

        return sx, sy

    def speed():
        return float(randint(50, 100))

    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

    def player1control(player_point, ball_point, speed, side, seconds):
        player_x, player_y = player_point
        ball_x, ball_y = ball_point

        if ball_y < 400:
            if player_x < ball_x + side[0]:
                player_x += speed * seconds

                if player_x < 0:
                    player_x = 0
                elif player_x + side[0] > size[0]:
                    player_x = size[0] - side[0]

            if player_x > ball_x:
                player_x -= speed * seconds

                if player_x < 0:
                    player_x = 0
                elif player_x + side[0] > size[0]:
                    player_x = size[0] - side[0]

        return player_x, player_y

    def player2control(player_point, ball_point, speed, side, seconds):
        player_x, player_y = player_point
        ball_x, ball_y = ball_point
        if ball_y > 400:
            if player_x < ball_x + side[0]:
                player_x += speed * seconds

                if player_x < 0:
                    player_x = 0
                elif player_x + side[0] > size[0]:
                    player_x = size[0] - side[0]

            if player_x > ball_x:
                player_x -= speed * seconds

                if player_x < 0:
                    player_x = 0
                elif player_x + side[0] > size[0]:
                    player_x = size[0] - side[0]

        return player_x, player_y

    def userplay(screen):

        ball_point = (300, 400)
        ball_x, ball_y = ball_point
        radius = 10.

        side = (80, 12)
        bat_point = (260, 740)
        player_point = (260, 60)
        bat_speed = 150

        sign_x, sign_y = initial()
        ball_dx = sign_x * speed()
        ball_dy = sign_y * speed()

        score_point = (700, 500)
        bat_score = 0
        player_score = 0
        font = pygame.font.SysFont("arial", 32)

        clock = pygame.time.Clock()

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

            screen.fill(black)
            seconds = clock.tick(30) / 1000.0

            ball = pygame.draw.circle(screen, yellow, ball_point, radius)
            bat = pygame.draw.rect(screen, white, Rect(bat_point, side))
            player = pygame.draw.rect(screen, gray, Rect(player_point, side))

            player_x, player_y = player_point

            bat_text = font.render(str(bat_score), True, white)
            player_text = font.render(str(player_score), True, gray)
            screen.blit(bat_text, (size[0]-bat_text.get_width()-20, \
                                                            size[1]/2+80))
            screen.blit(player_text, (size[0]-player_text.get_width()-20,\
                                                            size[1]/2-80))

            # ball control
            if bat.colliderect(ball):
                ball_x, ball_y, ball_dx, ball_dy = \
                              rebound(ball_point, ball_dx, ball_dy, seconds)
            elif player.colliderect(ball):
                ball_x, ball_y, ball_dx, ball_dy = \
                              rebound(ball_point, ball_dx, ball_dy, seconds)
            else:
                ball_x, ball_y, ball_dx, ball_dy, bat_score, player_score = \
                             ballcontrol(ball_point, radius, ball_dx, ball_dy,\
                                             seconds, bat_score, player_score)

            ball_point = (ball_x, ball_y)

            # user's bat
            pressed_key = pygame.key.get_pressed()
            bat_point = batcontrol(pressed_key, bat_point, bat_speed, \
                                                                 side, seconds)

            # artificial intelligence
            player_point = player1control(player_point, ball_point, \
                                                      bat_speed, side, seconds)

            if pygame.key.get_pressed()[K_ESCAPE]:
                menu(screen, prompt)

            pygame.display.update()

    def simulateplay(screen):
        ball_point = (300, 400)
        ball_x, ball_y = ball_point
        radius = 10.

        side = (80, 12)
        bat_point = (260, 740)
        player_point = (260, 60)
        bat_speed = 150

        sign_x, sign_y = initial()
        ball_dx = sign_x * speed()
        ball_dy = sign_y * speed()

        score_point = (700, 500)
        bat_score = 0
        player_score = 0
        font = pygame.font.SysFont("arial", 32)

        clock = pygame.time.Clock()

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

            screen.fill(black)
            seconds = clock.tick(30) / 1000.0
            ball = pygame.draw.circle(screen, yellow, ball_point, radius)
            bat = pygame.draw.rect(screen, white, Rect(bat_point, side))
            player = pygame.draw.rect(screen, gray, Rect(player_point, side))

            player_x, player_y = player_point

            bat_text = font.render(str(bat_score), True, white)
            player_text = font.render(str(player_score), True, gray)
            screen.blit(bat_text, (size[0]-bat_text.get_width()-20, \
                                                               size[1]/2+80))
            screen.blit(player_text, (size[0]-player_text.get_width()-20,\
                                                               size[1]/2-80))

            # ball control
            if bat.colliderect(ball):
                ball_x, ball_y, ball_dx, ball_dy = \
                                rebound(ball_point, ball_dx, ball_dy, seconds)
            elif player.colliderect(ball):
                ball_x, ball_y, ball_dx, ball_dy = \
                                rebound(ball_point, ball_dx, ball_dy, seconds)
            else:
                ball_x, ball_y, ball_dx, ball_dy, bat_score, player_score = \
                             ballcontrol(ball_point, radius, ball_dx, ball_dy,\
                                             seconds, bat_score, player_score)

            ball_point = (ball_x, ball_y)

            # artificial intelligence
            player_point = player1control(player_point, ball_point, \
                                                     bat_speed, side, seconds)
            bat_point = player2control(bat_point, ball_point, bat_speed, \
                                                                side, seconds)

            if pygame.key.get_pressed()[K_ESCAPE]:
                menu(screen, prompt)

            pygame.display.update()

    def menu(screen, prompt):

        font = pygame.font.SysFont("arial", 40)

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

            text1 = font.render(prompt[1], True, white)
            text1_1 = font.render(prompt[1], True, aqua)
            text2 = font.render(prompt[2], True, white)
            text2_1 = font.render(prompt[2], True, aqua)
            text3 = font.render(prompt[3], True, white)
            text3_1 = font.render(prompt[3], True, aqua)

            x, y = pygame.mouse.get_pos()

            screen.fill(black)

            if 200 <= x <= 200 + text1.get_width() and \
                                     200 <= y <= 200 + text1.get_height():
```

```
                screen.blit(text1_1, (200, 200))
                screen.blit(text2, (200, 240))
                screen.blit(text3, (200, 280))
                if pygame.mouse.get_pressed()[0]:
                    userplay(screen)
            elif 200 <= x <= 200 + text2.get_width() and \
                                         240 <= y <= 240 + text1.get_height():
                screen.blit(text1, (200, 200))
                screen.blit(text2_1, (200, 240))
                screen.blit(text3, (200, 280))
                if pygame.mouse.get_pressed()[0]:
                    simulateplay(screen)
            elif 200 <= x <= 200 + text3.get_width() and \
                                         280 <= y <= 280 + text1.get_height():
                screen.blit(text1, (200, 200))
                screen.blit(text2, (200, 240))
                screen.blit(text3_1, (200, 280))
                if pygame.mouse.get_pressed()[0]:
                    exit()
            else:
                screen.blit(text1, (200, 200))
                screen.blit(text2, (200, 240))
                screen.blit(text3, (200, 280))

            pygame.display.update()

    def run():
        pygame.init()

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

        menu(screen, prompt)


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