Python 精通程式語言
Python遊戲設計

吳佳諺　老師

9.使用matplotlib畫圖
10.網站擷取使用Python
11.行程和執行緒
12.資料及檔案處理
13.SQLite資料庫
14.人機界面及影像處理
15.Python遊戲設計

Python遊戲設計

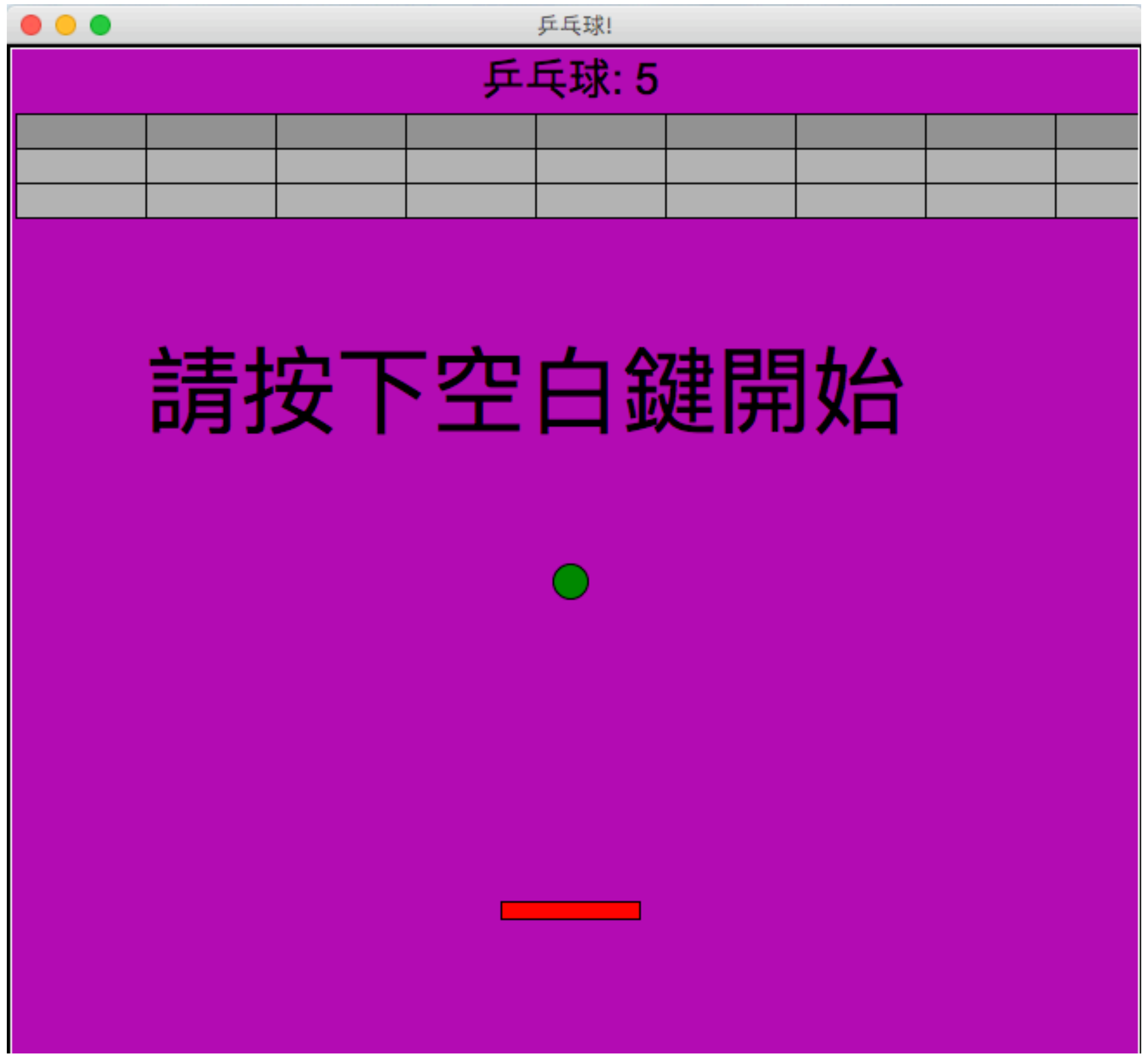
- 15-1乒乓球遊戲
- 15-2人機界面Tkinter
- 15-3乒乓球遊戲實作

15-1乒乓球遊戲

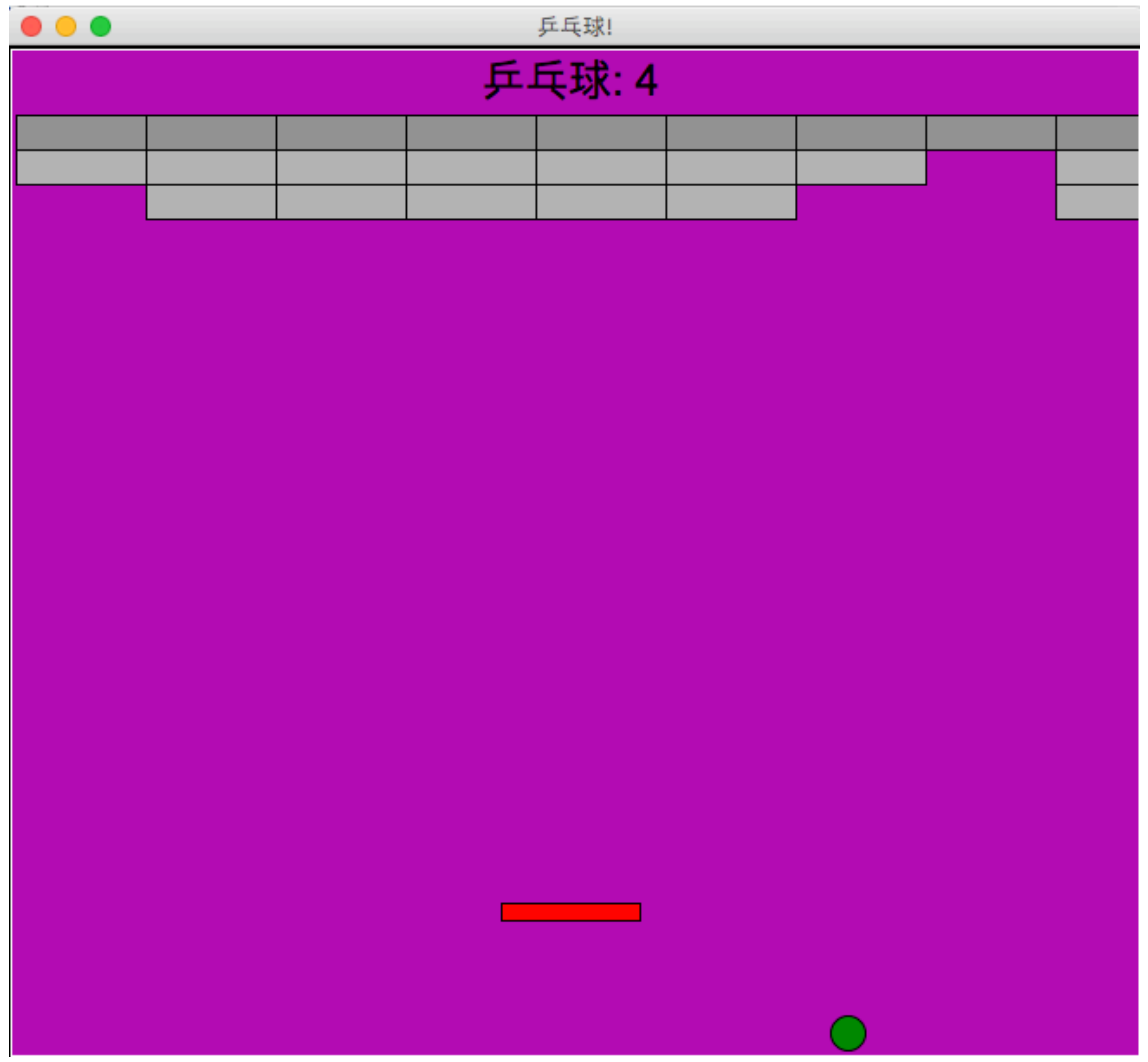
- 遊戲開始按下空白鍵
- 乒乓球掉到底下,乒乓球減少一顆
- 乒乓球撞到灰色磚塊可以消滅磚塊
- 球撞到黑色磚塊會變灰色磚塊
- 磚塊全部撞完,你贏囉
- 遊戲結束囉

python

遊戲開始按下空白鍵

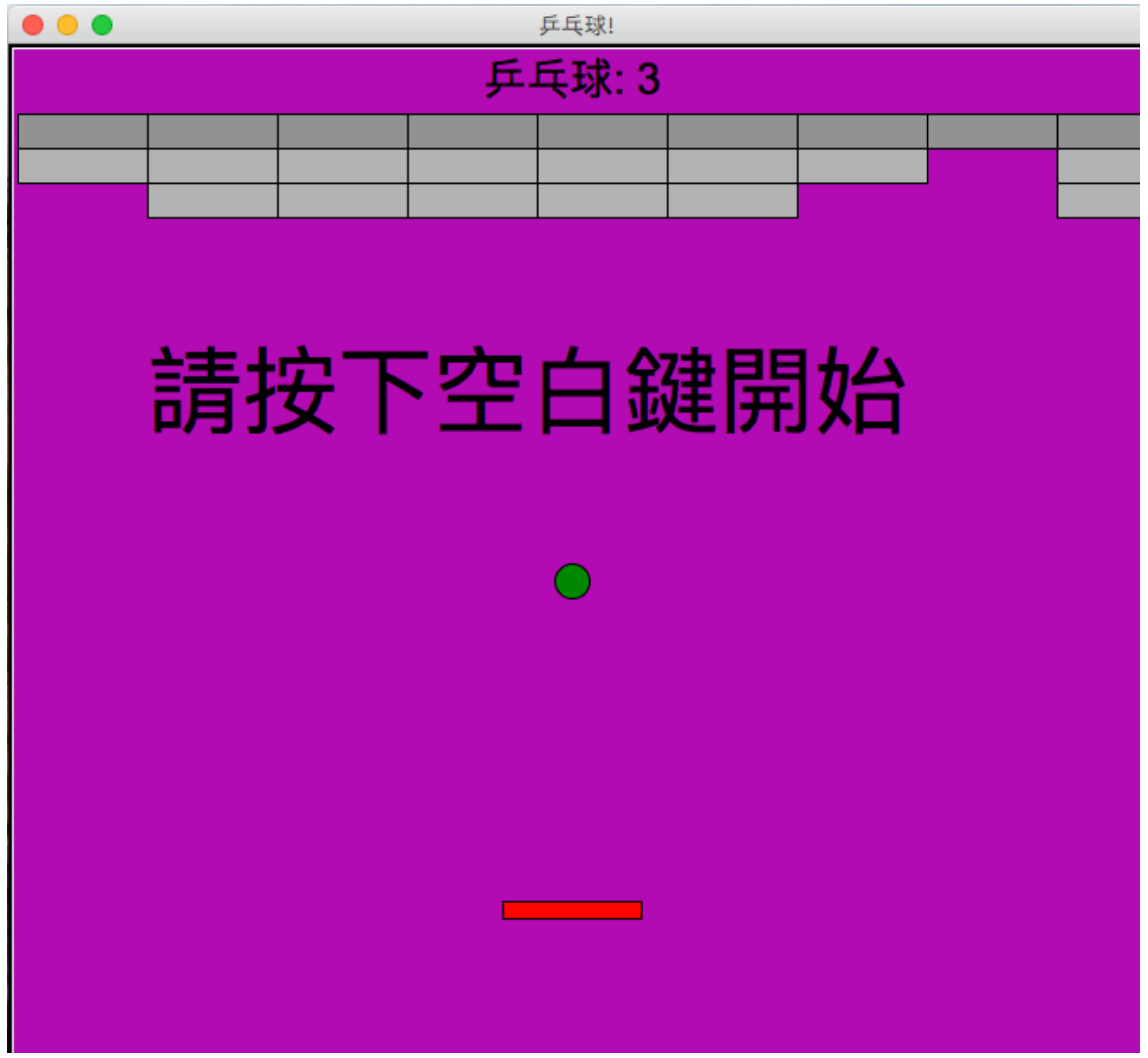


乒乓球掉到底下,乒乓球減少一顆



python

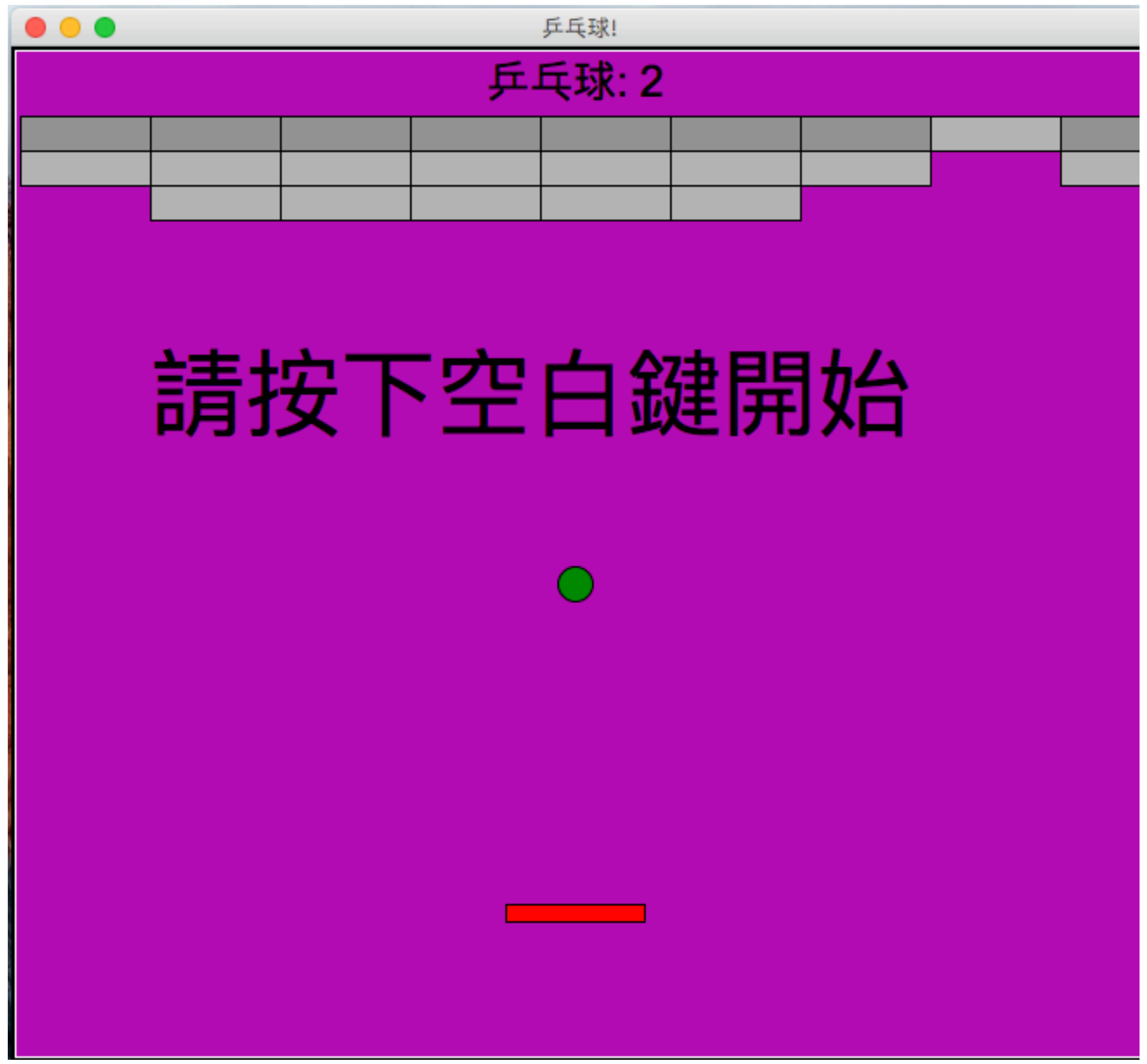
剩下乒乓球3顆


乒乓球!

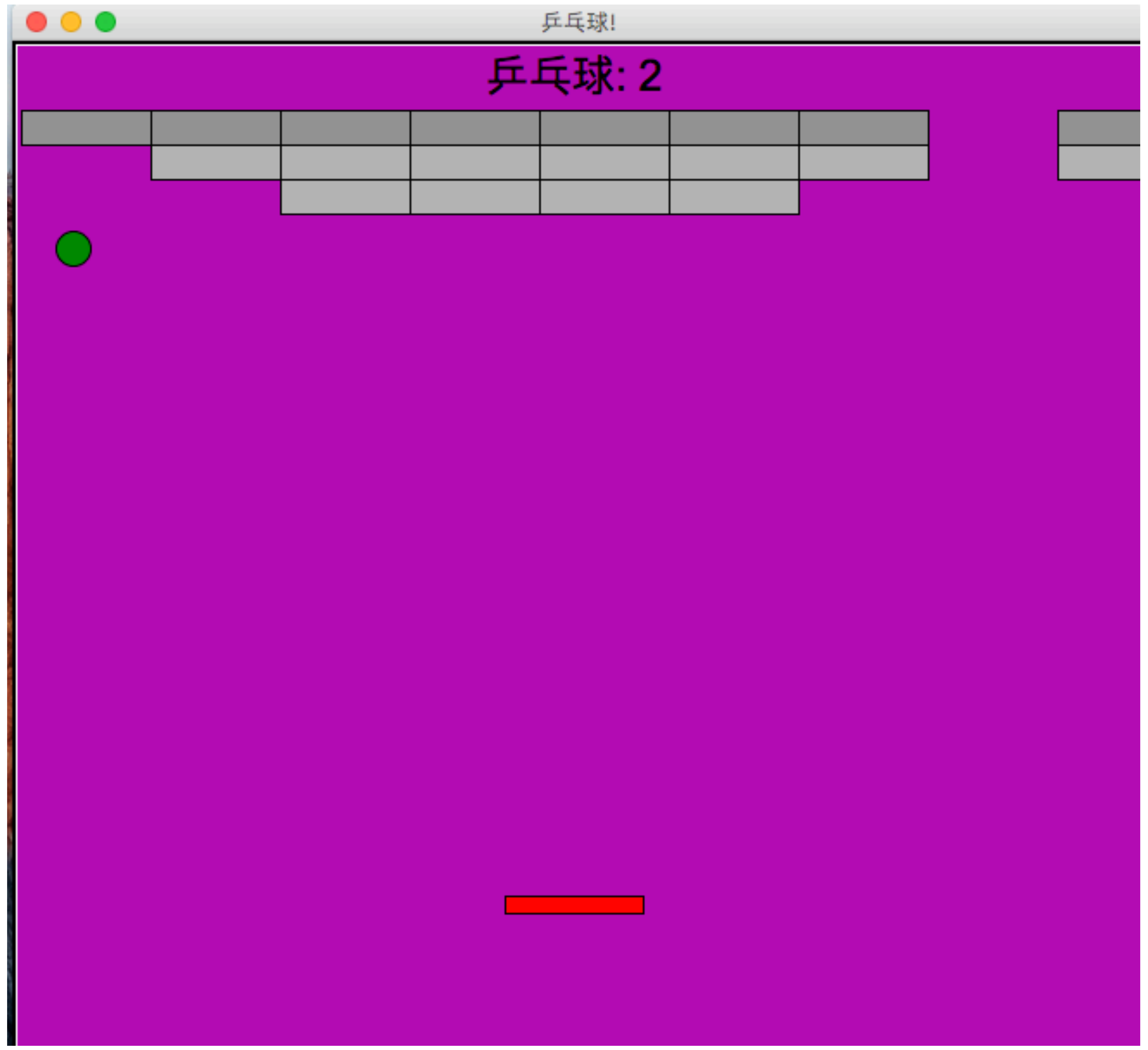
乒乓球: 3

請按下空白鍵開始

剩下乒乓球2顆

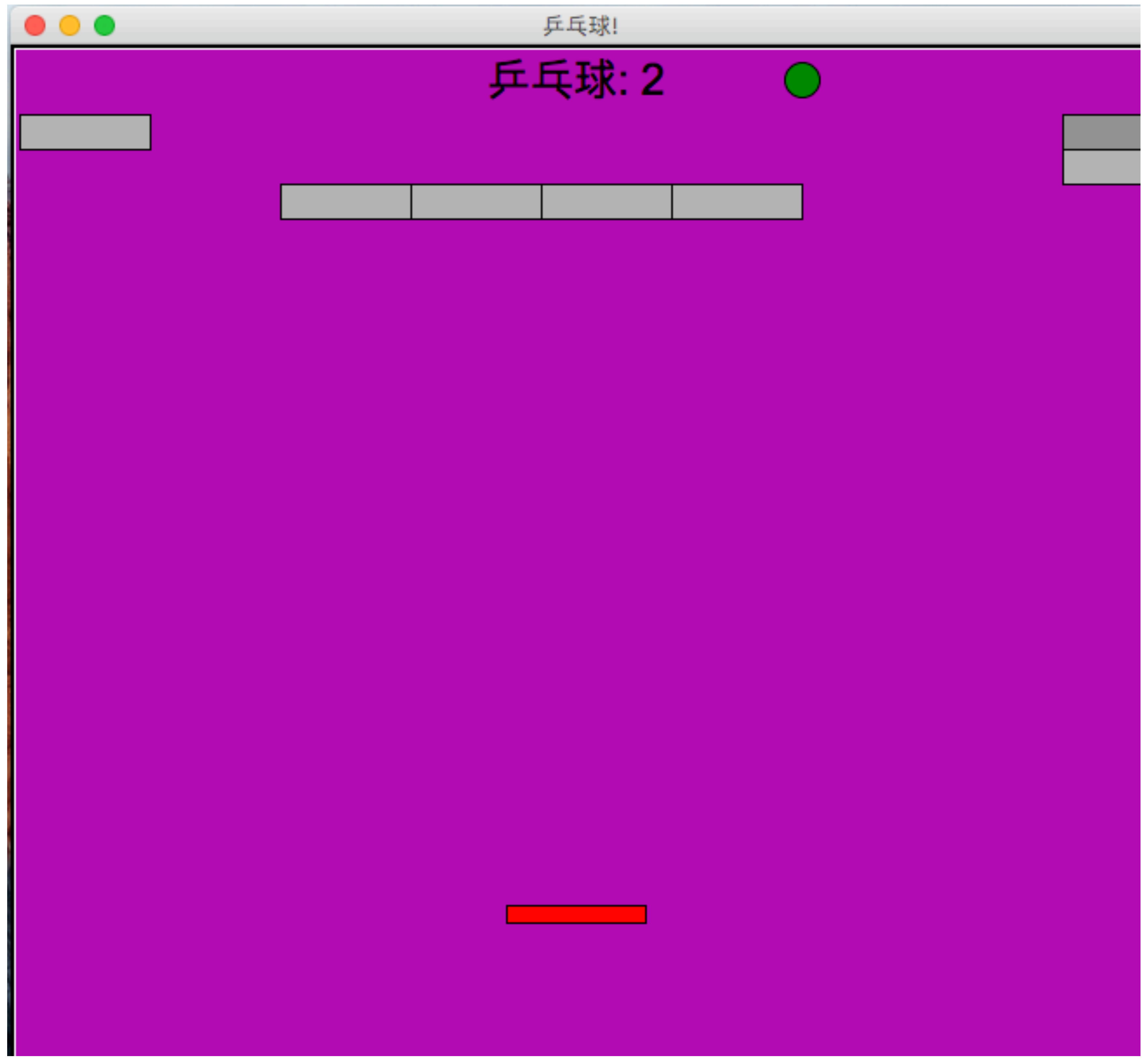


乒乓球撞到灰色磚塊可以消滅磚塊

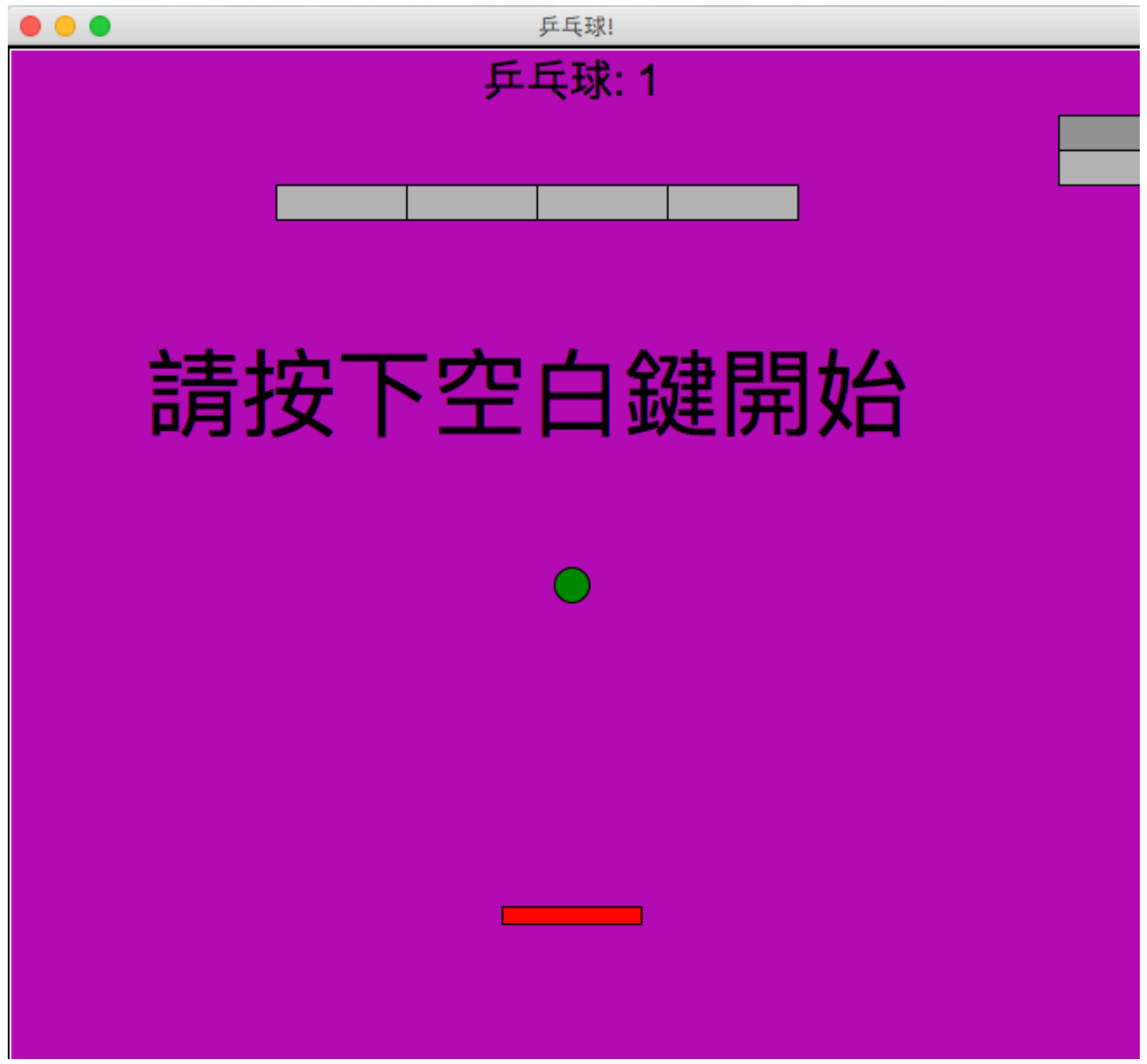


球撞到黑色磚塊會變灰色磚塊

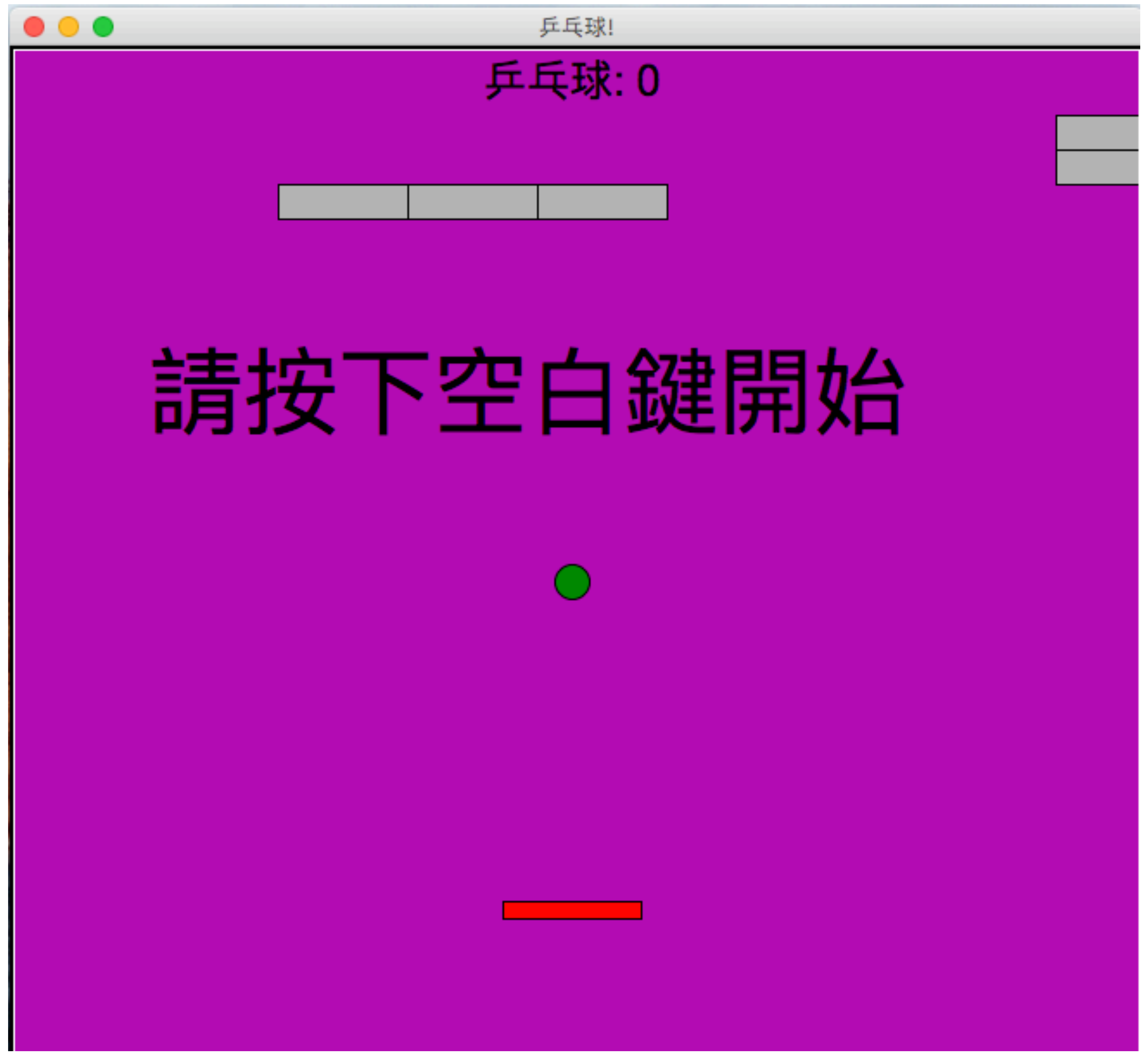
- 上面灰黑色磚塊撞到,會變灰色



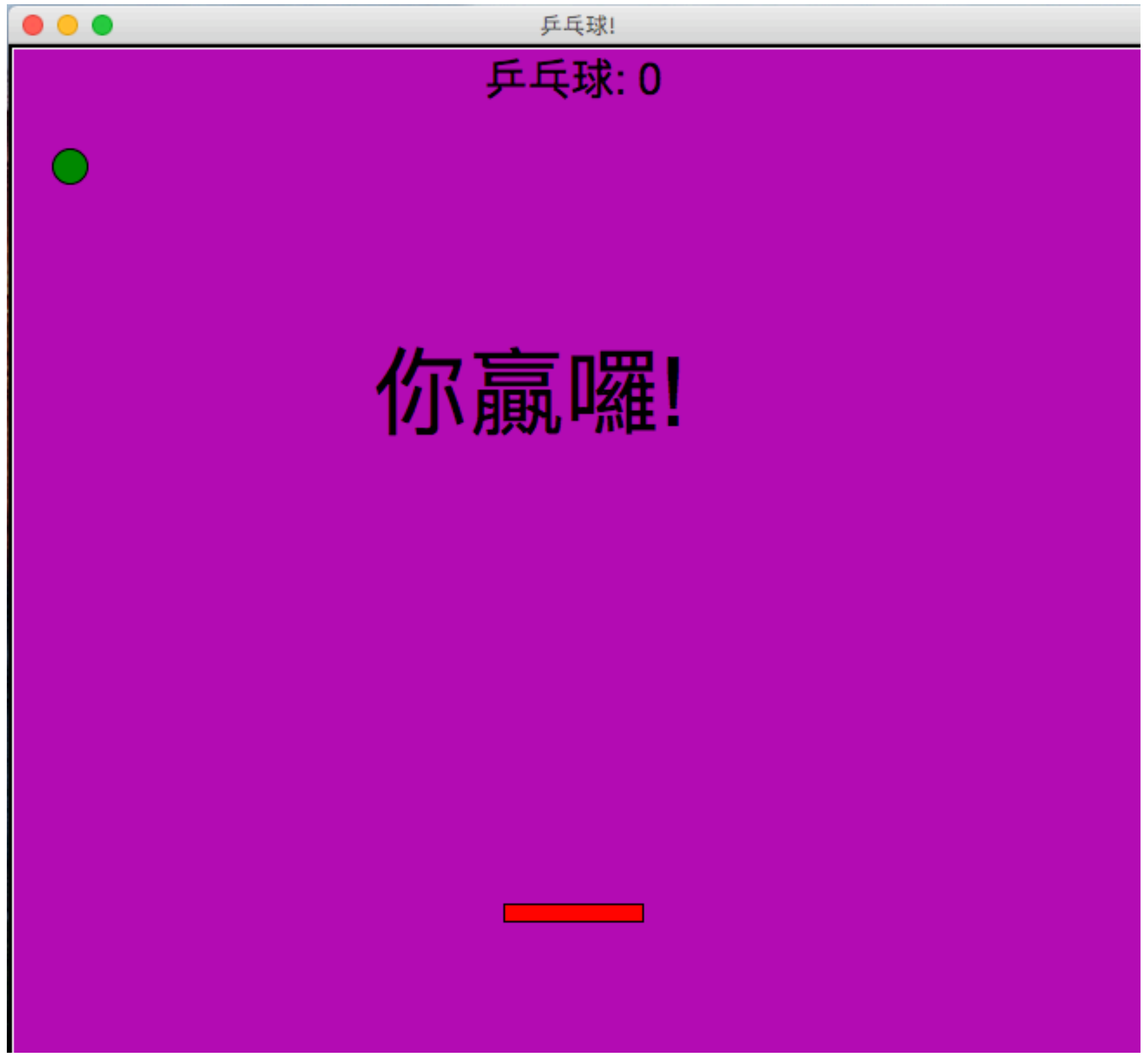
剩下乒乓球1顆



剩下乒乓球0顆

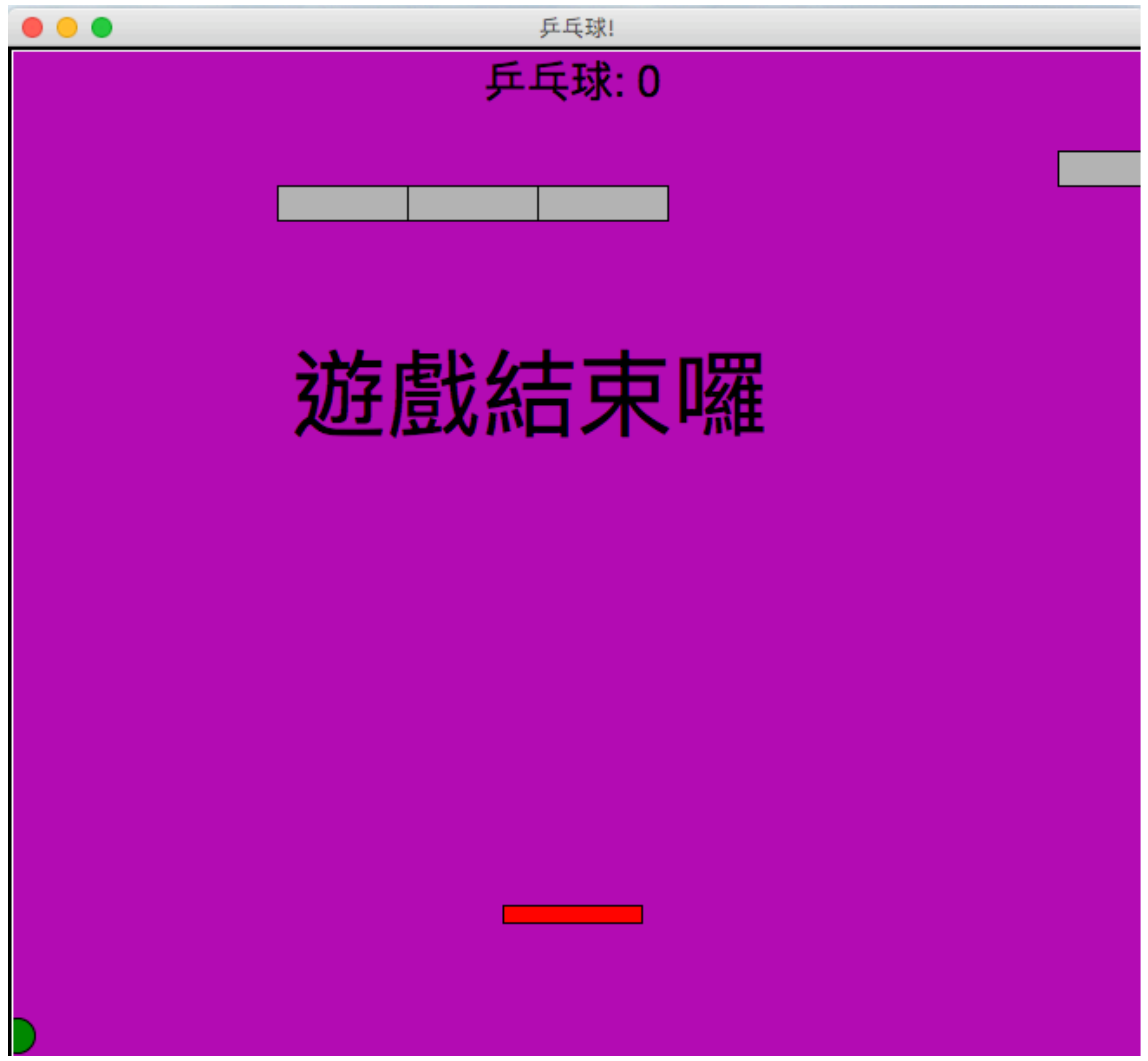


磚塊全部撞完,你贏囉



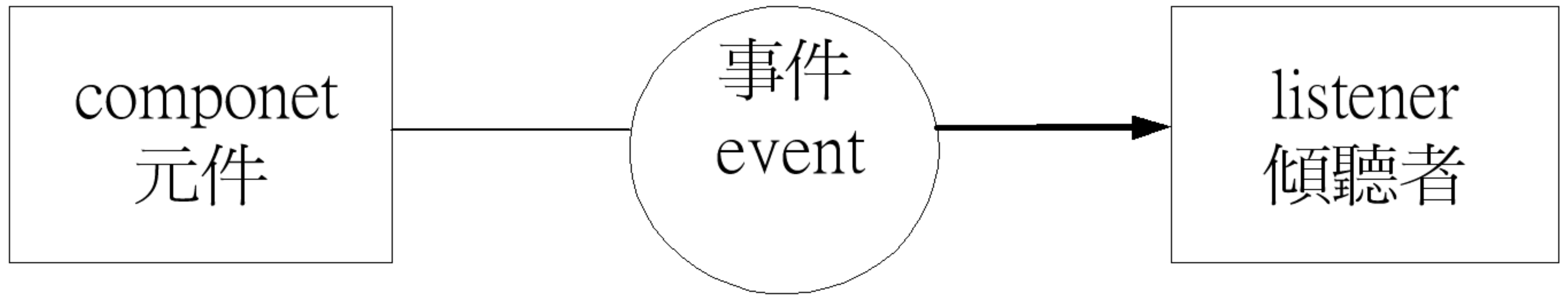
遊戲結束囉

- 當乒乓球剩下0顆時,遊戲結束囉



15-2人機界面Tkinter

- 在圖形化介面，一個事件代表了一些動作，例如按下空白鍵按鈕，在鍵盤上按下左右鍵按鈕，在視窗上按下關掉視窗的按鈕，或者是任何期望引發回應的動作。



15-2-1事件驅動程式設計

- 事件驅動程式設計使用告知和回應的方法來作程式設計。
- 告知物件被稱為事件(event)。
- 使用事件和事件處理。
- 一個事件(event)就是一個物件，它作用在另外一個稱為傾聽的物件上。
- 事件的傳送稱為啟動事件。
- Canvas.bind()函數為註冊傾聽的函數物件.

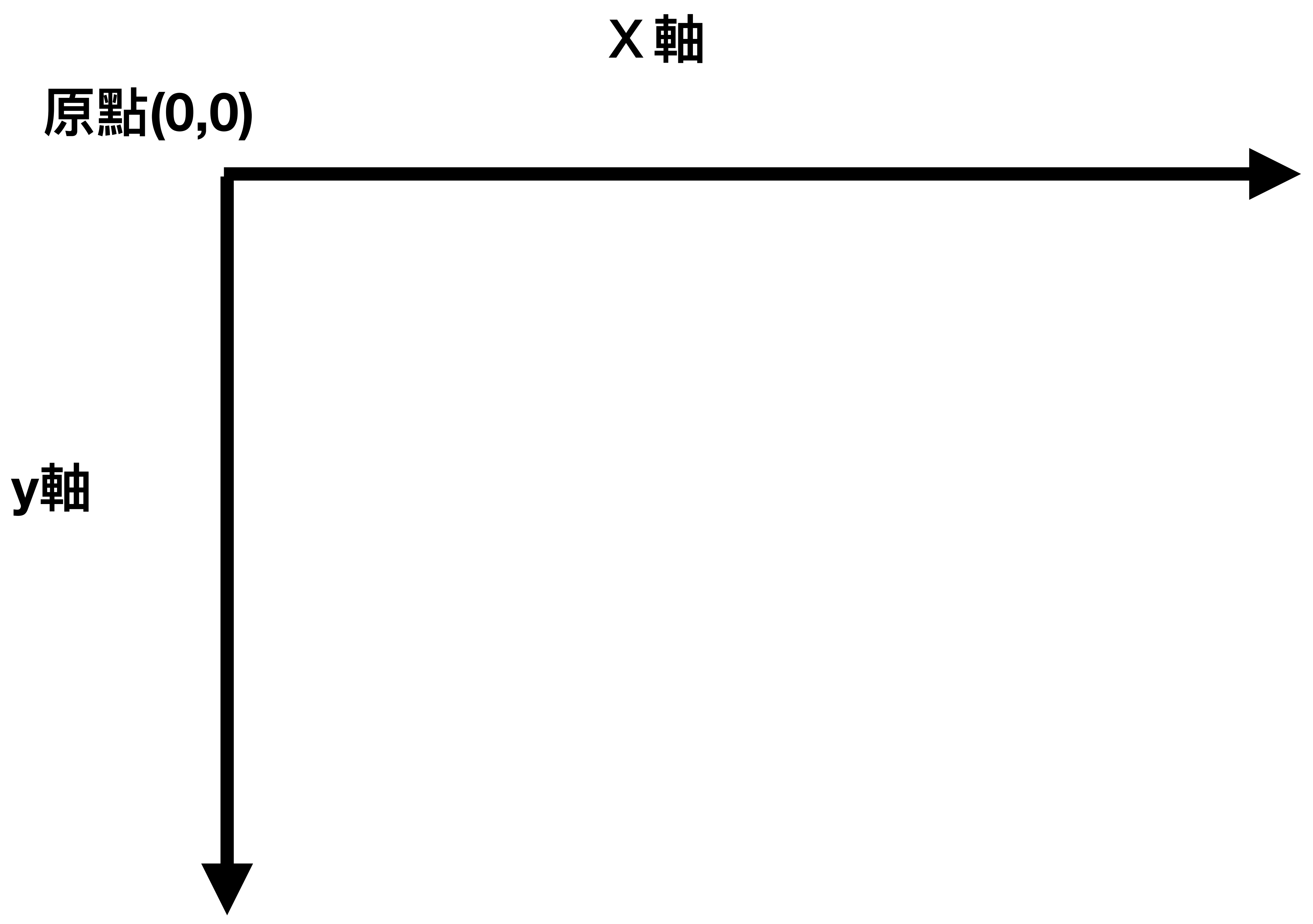
```
self.canvas.bind('<Left>',lambda _: self.paddle.move(-30))
self.canvas.bind('<Right>',lambda _: self.paddle.move(30))

self.canvas.bind('<space>', lambda _: self.start_game())
```

傾聽的函數物件start_game()

```
def start_game(self):
    self.canvas.unbind('<space>')
    self.canvas.delete(self.text)
    self.paddle.ball = None
    self.game_loop()
```

python

15-2-2座標軸



使用tkinter套件

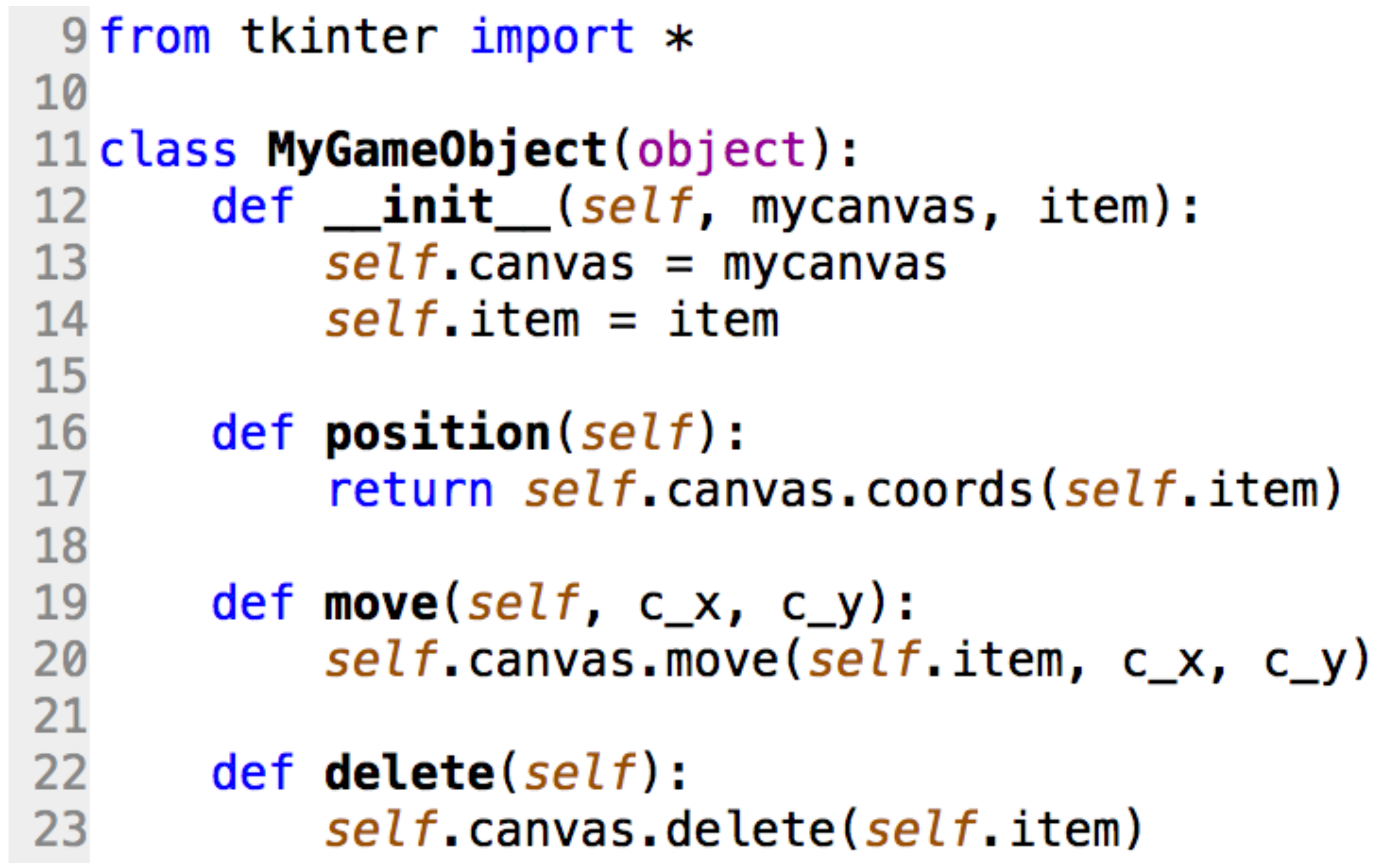
- 從tkinter人機界面套件輸入所有模組
- from tkinter import *
- #建立最上層視窗
- master = Tk()

- #從tkinter人機界面套件輸入所有模組
- from tkinter import *
- class MyGame(Frame):#繼承自父類別Frame
-     #初始化視窗
-         master = Tk()
-         master.title('乒乓球!')
-         super(MyGame, self).__init__(master)
- #建立最上層視窗
- master = Tk()
- #建立視窗實體
- game = MyGame()

15-3乒乓球遊戲實作

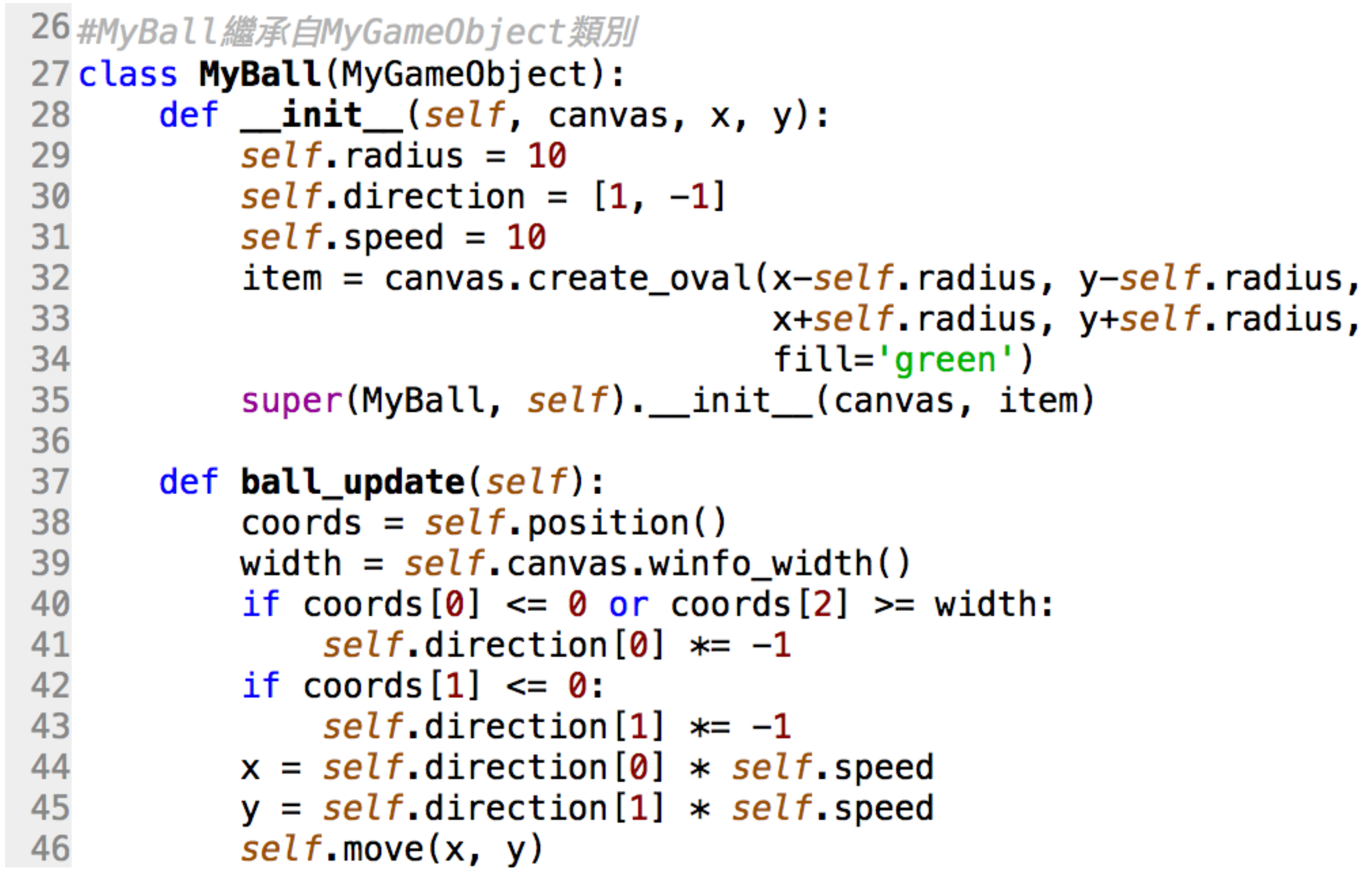
- MyGameObject類別
- MyBall繼承自MyGameObject類別
- MyPaddle類別繼承自MyGameObject類別
- MyBrick類別繼承自MyGameObject類別
- MyGame類別繼承自Frame類別
- 開始遊戲

MyGameObject類別



```python
from tkinter import *

class MyGameObject(object):
    def __init__(self, mycanvas, item):
        self.canvas = mycanvas
        self.item = item

    def position(self):
        return self.canvas.coords(self.item)

    def move(self, c_x, c_y):
        self.canvas.move(self.item, c_x, c_y)

    def delete(self):
        self.canvas.delete(self.item)
```

MyBall繼承自 MyGameObject類別

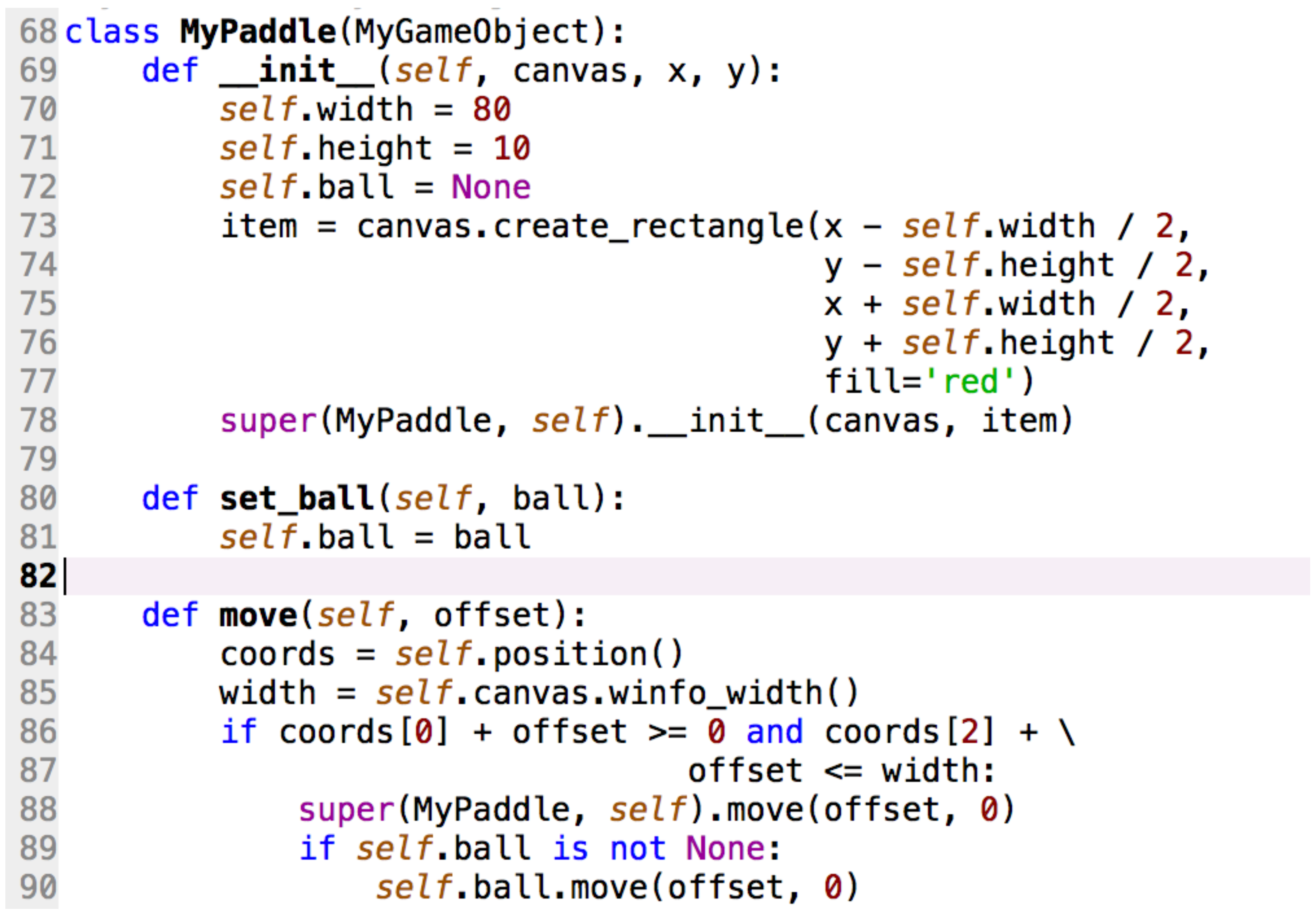


```python
#MyBall繼承自MyGameObject類別
class MyBall(MyGameObject):
    def __init__(self, canvas, x, y):
        self.radius = 10
        self.direction = [1, -1]
        self.speed = 10
        item = canvas.create_oval(x-self.radius, y-self.radius,
                                  x+self.radius, y+self.radius,
                                  fill='green')
        super(MyBall, self).__init__(canvas, item)

    def ball_update(self):
        coords = self.position()
        width = self.canvas.winfo_width()
        if coords[0] <= 0 or coords[2] >= width:
            self.direction[0] *= -1
        if coords[1] <= 0:
            self.direction[1] *= -1
        x = self.direction[0] * self.speed
        y = self.direction[1] * self.speed
        self.move(x, y)
```

MyGameObject類別

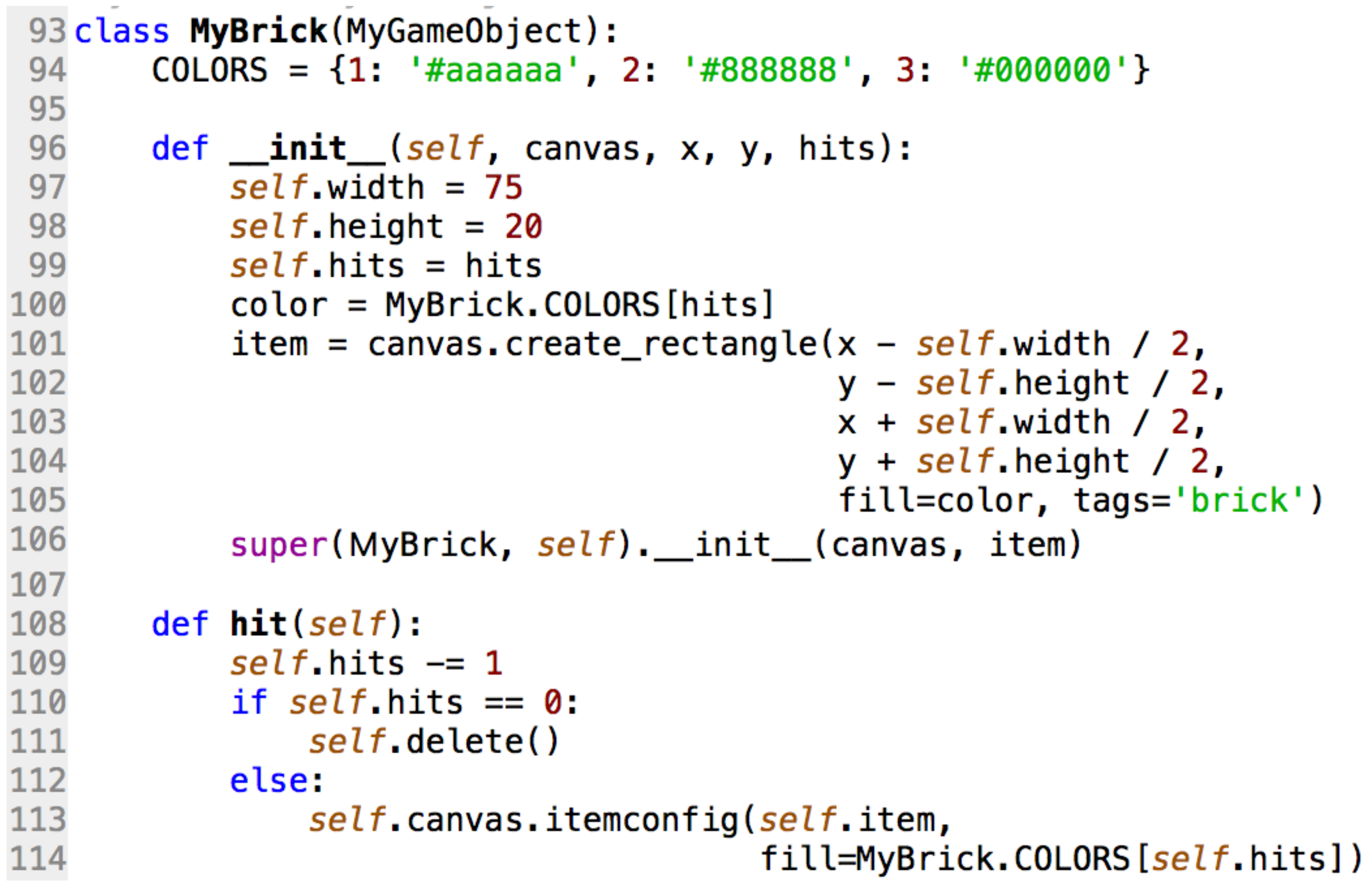
```python
    def ball_collide(self, game_objects):
        coords = self.position()
        x = (coords[0] + coords[2]) * 0.5
        if len(game_objects) > 1:
            self.direction[1] *= -1
        elif len(game_objects) == 1:
            game_object = game_objects[0]
            coords = game_object.position()
            if x > coords[2]:
                self.direction[0] = 1
            elif x < coords[0]:
                self.direction[0] = -1
            else:
                self.direction[1] *= -1

        for game_object in game_objects:
            if isinstance(game_object, MyBrick):
                game_object.hit()
```

MyPaddle類別繼承自 MyGameObject類別



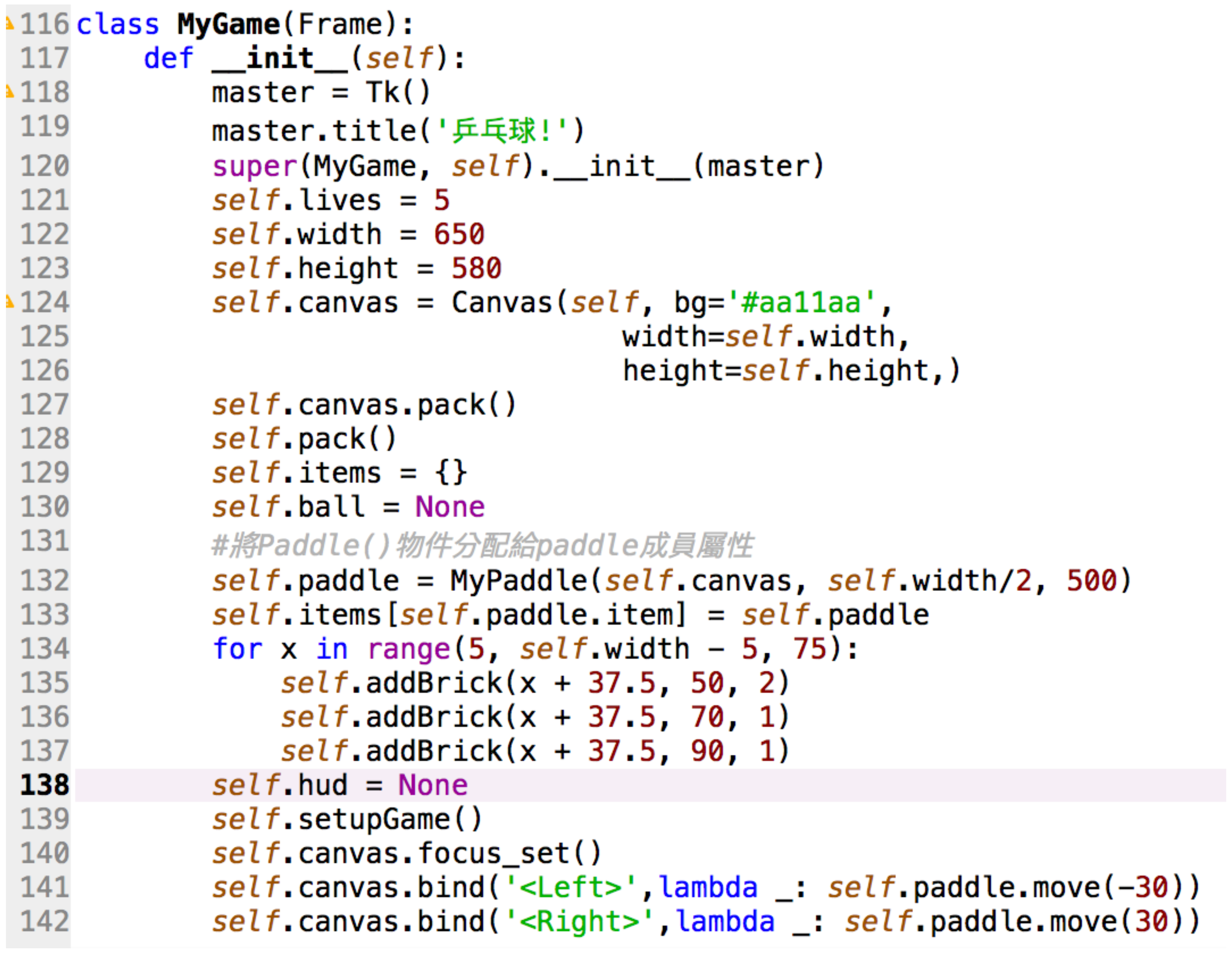
```python
class MyPaddle(MyGameObject):
    def __init__(self, canvas, x, y):
        self.width = 80
        self.height = 10
        self.ball = None
        item = canvas.create_rectangle(x - self.width / 2,
                                       y - self.height / 2,
                                       x + self.width / 2,
                                       y + self.height / 2,
                                       fill='red')
        super(MyPaddle, self).__init__(canvas, item)

    def set_ball(self, ball):
        self.ball = ball

    def move(self, offset):
        coords = self.position()
        width = self.canvas.winfo_width()
        if coords[0] + offset >= 0 and coords[2] + \
                                offset <= width:
            super(MyPaddle, self).move(offset, 0)
            if self.ball is not None:
                self.ball.move(offset, 0)
```

MyBrick類別繼承自 MyGameObject類別



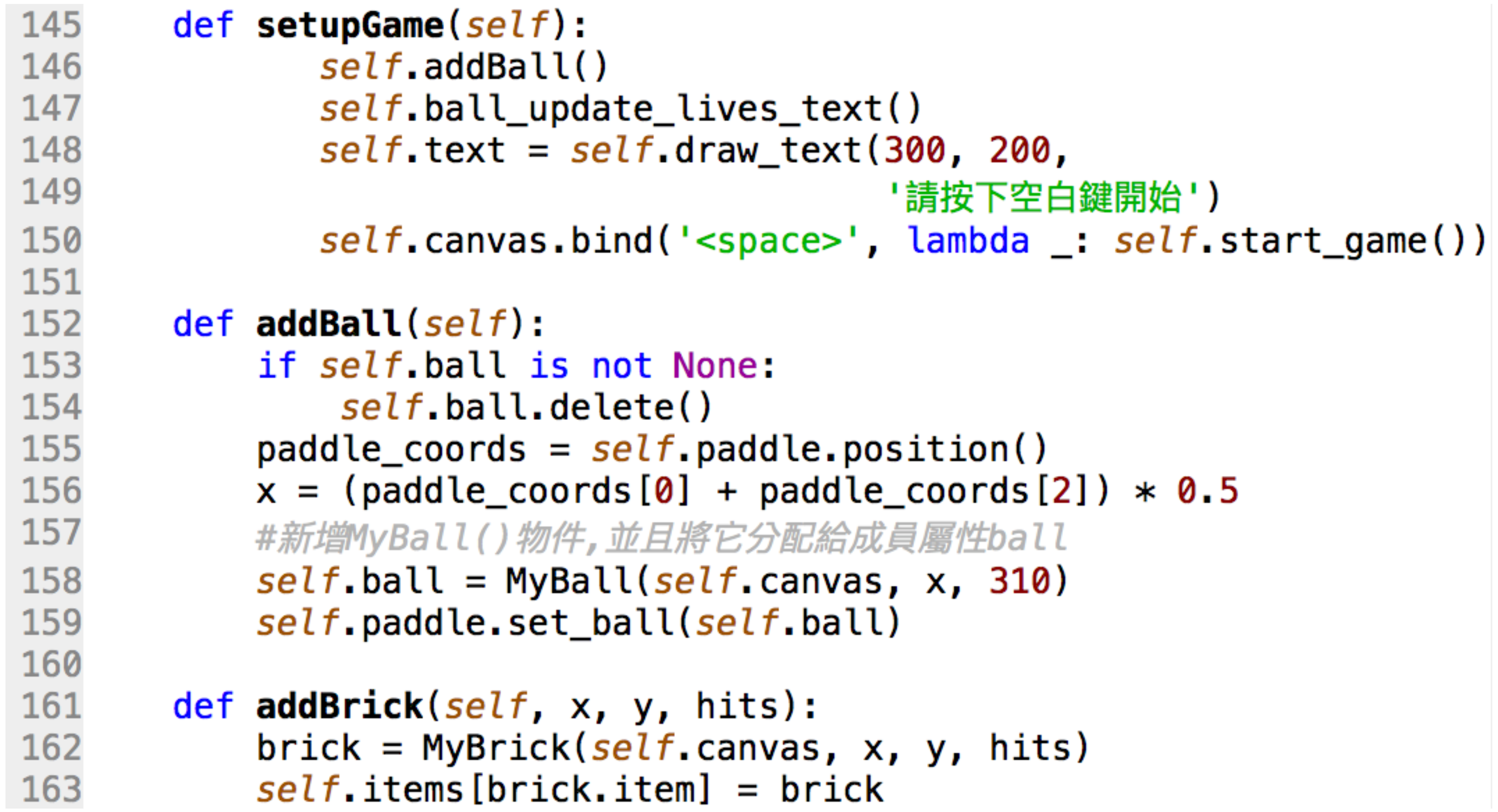
```python
class MyBrick(MyGameObject):
    COLORS = {1: '#aaaaaa', 2: '#888888', 3: '#000000'}

    def __init__(self, canvas, x, y, hits):
        self.width = 75
        self.height = 20
        self.hits = hits
        color = MyBrick.COLORS[hits]
        item = canvas.create_rectangle(x - self.width / 2,
                                       y - self.height / 2,
                                       x + self.width / 2,
                                       y + self.height / 2,
                                       fill=color, tags='brick')
        super(MyBrick, self).__init__(canvas, item)

    def hit(self):
        self.hits -= 1
        if self.hits == 0:
            self.delete()
        else:
            self.canvas.itemconfig(self.item,
                                   fill=MyBrick.COLORS[self.hits])
```

MyGame類別繼承自Frame類別



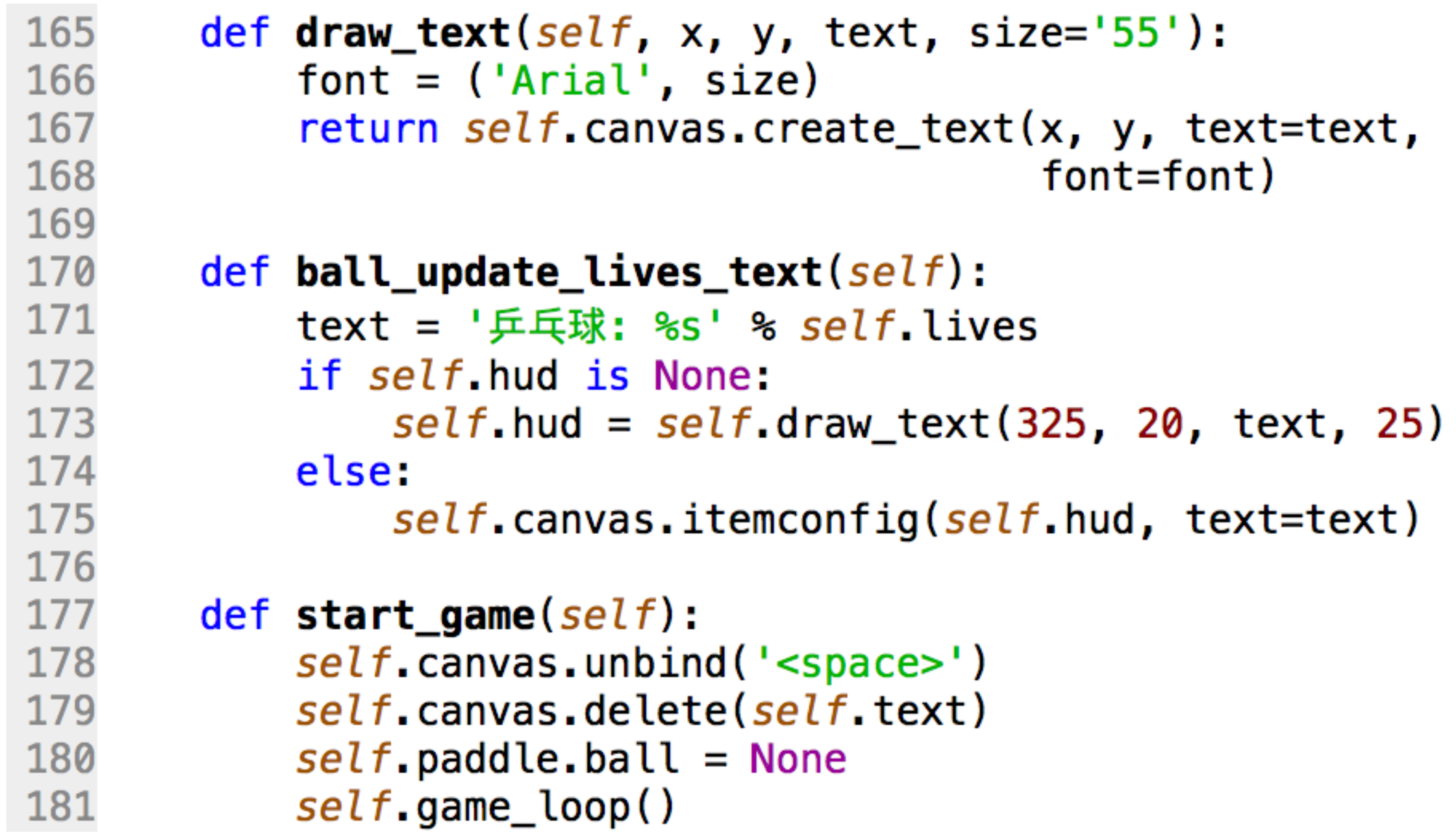
```python
class MyGame(Frame):
    def __init__(self):
        master = Tk()
        master.title('乒乓球!')
        super(MyGame, self).__init__(master)
        self.lives = 5
        self.width = 650
        self.height = 580
        self.canvas = Canvas(self, bg='#aa11aa',
                                width=self.width,
                                height=self.height,)
        self.canvas.pack()
        self.pack()
        self.items = {}
        self.ball = None
        #將Paddle()物件分配給paddle成員屬性
        self.paddle = MyPaddle(self.canvas, self.width/2, 500)
        self.items[self.paddle.item] = self.paddle
        for x in range(5, self.width - 5, 75):
            self.addBrick(x + 37.5, 50, 2)
            self.addBrick(x + 37.5, 70, 1)
            self.addBrick(x + 37.5, 90, 1)
        self.hud = None
        self.setupGame()
        self.canvas.focus_set()
        self.canvas.bind('<Left>',lambda _: self.paddle.move(-30))
        self.canvas.bind('<Right>',lambda _: self.paddle.move(30))
```

MyGame類別



```python
    def setupGame(self):
           self.addBall()
           self.ball_update_lives_text()
           self.text = self.draw_text(300, 200,
                                      '請按下空白鍵開始')
           self.canvas.bind('<space>', lambda _: self.start_game())

    def addBall(self):
        if self.ball is not None:
            self.ball.delete()
        paddle_coords = self.paddle.position()
        x = (paddle_coords[0] + paddle_coords[2]) * 0.5
        #新增MyBall()物件,並且將它分配給成員屬性ball
        self.ball = MyBall(self.canvas, x, 310)
        self.paddle.set_ball(self.ball)

    def addBrick(self, x, y, hits):
        brick = MyBrick(self.canvas, x, y, hits)
        self.items[brick.item] = brick
```

MyGame類別



```python
    def draw_text(self, x, y, text, size='55'):
        font = ('Arial', size)
        return self.canvas.create_text(x, y, text=text,
                                       font=font)

    def ball_update_lives_text(self):
        text = '乒乓球: %s' % self.lives
        if self.hud is None:
            self.hud = self.draw_text(325, 20, text, 25)
        else:
            self.canvas.itemconfig(self.hud, text=text)

    def start_game(self):
        self.canvas.unbind('<space>')
        self.canvas.delete(self.text)
        self.paddle.ball = None
        self.game_loop()
```

MyGame類別

```python
    def game_loop(self):
        self.checkCollisions()
        num_bricks = len(self.canvas.find_withtag('brick'))
        if num_bricks == 0:
            self.ball.speed = None
            self.draw_text(300, 200, '你贏囉！')
        elif self.ball.position()[3] >= self.height:
            self.ball.speed = None
            self.lives -= 1
            if self.lives < 0:
                self.draw_text(300, 200, '遊戲結束囉')
            else:
                self.after(1000, self.setupGame)
        else:
            self.ball.ball_update()
            self.after(50, self.game_loop)

    def checkCollisions(self):
        ball_coords = self.ball.position()
        items = self.canvas.find_overlapping(*ball_coords)
        objects = [self.items[x] for x in items if x in self.items]
        self.ball.ball_collide(objects)
```

50ms後再次執行

game_loop()函數

開始遊戲

```python
if __name__ == '__main__':
    game = MyGame()
    game.mainloop()
```

- Thanks.