麗山高中資訊專題研究

Python 拈遊戲

學生：20418 呂可名

指導老師:黃履峰

目錄

圖表目錄

壹、序論

一、 研究動機

第一次接觸到這個遊戲是在高一上學期的數學研方老師所提到。當下就對它非常有興趣，也很想將它做成一個遊戲。對於開發這個遊戲，除了編寫出此程式的演算法之外，我想進一步美化此程式的使用者介面，設計不同版本及規則的遊戲。因此我選擇與 C++、Java 相較之下比較好上手的 Python 來完成此遊戲。

二、 研究目標

學習 Python 的使用者介面程式架構，並用 python 程式語言設計出圖形介面的拈遊戲。

貳、 相關研究

一、 拈遊戲簡介

拈遊戲據說來自於中國，藉由被販賣到美洲的勞工外傳。其遊戲主要利用石頭或硬幣遊玩，最為常見的玩法為將硬幣分為三列，分別為三、四、五個硬幣，兩人輪流取硬幣，每次只能取一個或以上的硬幣在同列上，玩到最後，拿到最後一個硬幣的人就贏了。

二、 拈遊戲必勝原理研究

i. 異或(exclusive or)

異或(也稱為互斥或)，是一種邏輯運算，它與一般的「或」不同的地方是，只有當兩個輸入不同的邏輯運算。通常以 XOR、⊕、^表示。

XOR	True(1)	False(0)
True(1)	False(0)	True(1)
False(0)	True(1)	False(0)

表 1　異或真質表

ii. 拈運算

整個遊戲的必勝原理利用了異或進行了一連串的二進位運算，如下表：

棒子數	二進位數	說明
3 4 5	=　011 =　100 =　101	如為 3,4,5 個，則換成二進位進行異或運算三位無皆為 0 則為不安全殘局。
+(xor)	=　010　不安全殘局	
此時第一排拿走兩個		

1	= 001	此時從第一列取走 2 個，變為 1,4,5 個，換成二進位進位進行異或運算三位皆為 0 則為安全殘局
4	= 100	
5	= 101	
+(xor)	= 000 安全殘局	

表 2　實際計算流程

✧ 安全與不安全殘局

- 安全殘局：所有排數進行異或判斷的值為 0
- 不安全殘局：所有排數進行異或判斷的值不為 0

若處於安全殘局，則對方不管怎麼拿都會處於不安全殘局。若處於不安全殘局，則需要在某些排取適當的數量才可成為安全殘局。只要一直維持在安全殘局即可獲得勝利。

三、 MVC 程式架構

為模型(Model)、視圖(View)、控制器(Controller) 三個部分所構成。

三個部分彼此環環相扣，為實作一個動態的程式設計

1. 模型(Model)：程式的核心，存放資料及一些基本演算法，並且連結整個遊戲。
2. 視圖(View)：負責呈現圖形介面的部分。
3. 控制器(Controller)：接收使用者所進行的操作，包括鍵盤輸入、滑鼠點擊等。

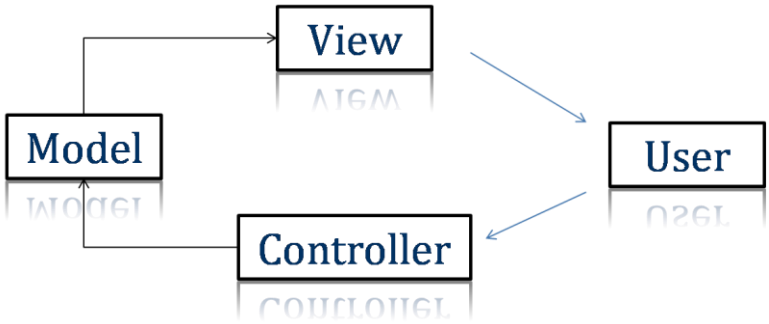

圖 1　MVC 架構

四、 Turtle、Tk、Pygame 模組

在整個遊戲主要運用到了 Python 的三個模組，分別是 Turtle、Tk 以及 Pygame。

i. Turtle

Turtle 模組是一個處理圖形介面的模組。其中包含了許多簡單又好用的函數，讓一般人能夠很快上手。

ii. Tk

是圖形介面的始祖，許多跟圖形介面有關的模組都繼承 Tk，包括 Turtle。而他能處理一些更底層的部分。

iii. Pygame

Python 用來製作遊戲的模組，可以處理圖形介面、聲音等等多媒體的效果。在此遊戲內用來播放背景音樂。

參、 遊戲實作

一、 演算法

程式的核心必勝演算法，就如同上面所說的利用異或判斷以及二進位運算來實現。

i. 程式碼

```
1  def computerzug(state):
2      nRows= len(state)  #指定排數
3      xored= 0
4      for s in state:
5          #每一排進行判斷
6          xored ^= s
7      move= randommove(state) # 不經考慮，隨機出手。
8      if xored != 0:
9          # 此情形，我方有機會勝利，
10         # 做一個動作，使得 接下來的 xored ==0，則敵方必敗。
11         for z in range(nRows):
12             s = state[z] ^ xored
13             if s <= state[z]:
14                 move= (z, s)
15                 break
16     return move
```

ii. 程式解說

```
1  xored= 0
2  for s in state:
3      #每一排進行判斷
4      xored ^= s
```

利用迴圈進行每一排的異或判斷

```
1  if xored != 0:
2          # 此情形，我方有機會勝利，
3          # 做一個動作，使得 接下來的 xored ==0，則敵方必敗。
4          for z in range(nRows):
5              s = state[z] ^ xored
6              if s <= state[z]:
7                  move= (z, s)
8                      break
```

利用異或判斷，決定該在第幾排拿掉幾根棒子。

二、 使用者介面

i. 主要類別(class)

1. NimGame

整個遊戲主要的類，包含了執行此遊戲的函數，主要實例化 MVC 及其他的類，將整個程式連結起來，

2. NimModel

為資料及演算法存放處，以及處理遊戲的一些基本運作，包括移動棒子、判斷輸贏、以及切換模式等等。

3. NimView

管理遊戲中所有畫面的顯現，包含背景、棒子、選單、音樂、文字等等，通常接收來自 NimModel 的訊息，

4. NimController

接收來自玩家使用鍵盤、滑鼠的訊息，進而將此訊息傳送到 NimModel 處理。

ii. MVC 主要功能

1. 建立視窗

NimGame 內部的兩行指令，分別是利用 Turtle、Tk 模組建立的視窗，再將這兩個視窗合併為一個視窗

```
1  self.screen =      turtle.Screen()
2  self.tk =          self.screen.getcanvas().master
```

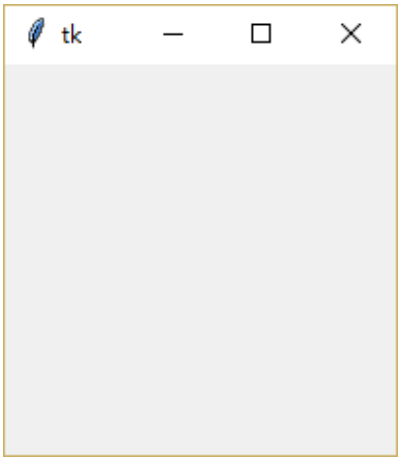

圖 2　TK 視窗

2. 製作棒子

在NimView內先指定好排數，再利用迴圈製作，並且顯現在螢幕上。

```
1  self.nRows= game.nRows
2  for row in range(self.nRows):
```

```
    for col in range(MAXSTICKS):
        self.sticks[(row, col)] = Stick(row, col, game
```

3. 電腦移動棒子、判斷輸贏

控制電腦選取棒子的主要函數，能夠切換以計算或者是隨機的方式選取，並且判斷輸贏。

```
def move(self, row, col):
        maxspalte = self.sticks[row]
        self.sticks[row] = col
        if self.game.mode == 0:
            self.game.view.notify_move(row, col,maxspalte,
self.player)
            if self.game_over():
                self.game.state = NimGame.OVER
                self.winner = self.player
                self.game.view.notify_over()
                #遊戲結束按鈕隱藏
                self.game.hint.ht()
                self.game.level.ht()
            elif self.player == 0:
                self.player = 1
                #偵測智商
                if self.game.level.value == True:
                    row, col = computerzug(self.sticks)
                else:
```

```
                    row, col = randommove(self.sticks)
                self.move(row, col)
                self.player = 0
        elif self.game.mode == 1:
            self.game.view.player_move(row, col, maxspalte,
self.player)
            if self.game_over():
                self.game.state = NimGame.OVER
                self.winner = self.player
self.game.view.player_over()
                #遊戲結束按鈕隱藏
                self.game.hint.ht()
                self.game.level.ht()
            if self.player == 0:
                self.player = 1
            else:
                self.player = 0
```

4. 滑鼠點擊移動棒子

在 NimController 內利用一個迴圈進行判斷每一次玩家所點擊的棒子是否被移動過，藉此來移動棒子。

```
for stick in self.sticks.values():
    if self.game.mode == 0:
        stick.onclick(stick.makemove)
    elif self.game.mode == 1:
        stick.onclick(stick.playermove
```

5. 鍵盤控制

在 NimController 內部接收鍵盤的訊息，再將此訊息傳至其他類別進行處理。

```
# 用 "space" 來 開始遊戲
    self.game.screen.onkey(self.game.model.setup, "space")

# 用 "Esc" 來 清除 畫面
    self.game.screen.onkey(self.game.view.clear, "Escape")
```

6. 文字訊息

遊戲下方所顯現的文字，使用了一些Turtle內建的文字版面設定。

```
def display(self, msg1, msg2=None):

        self.screen.tracer(False)
        self.writer.clear()

        if msg2 is not None:
            self.writer.goto(0, - SCREENHEIGHT // 2 + 48)
            self.writer.pencolor("red")
            self.writer.write(msg2, align="center",
font=("Courier",18,"bold"))
        self.writer.goto(0, - SCREENHEIGHT // 2 + 20)
```

遊戲剛開始時的文字說明。這部分是在此函數內創建了一個類(屬於 Turtle 類)，專門用來書寫文字。

```
def set_msg(self):
        msg = '''
                        Python 拈遊戲 (NimGame)


            以下是遊戲規則:

            1. 玩家與電腦輪流撿棒子，撿到最後一個的就贏了。

            2. 點擊任一棒子則玩家將會撿走該棒子右側的所有棒子。

            3. 點擊左下角的方塊會給予提示。

            4. 點擊右下角的烏龜可選擇難度(黃色代表難，橘色代表簡單)

            5. 遊戲開始前請選擇排數
        '''
        self.t = Turtle()
        self.t.pu()
        self.t.ht()
        self.t.goto(0,-50)
        self.t.write(msg, align="center", font=('Arial', 15,
'bold'))
```

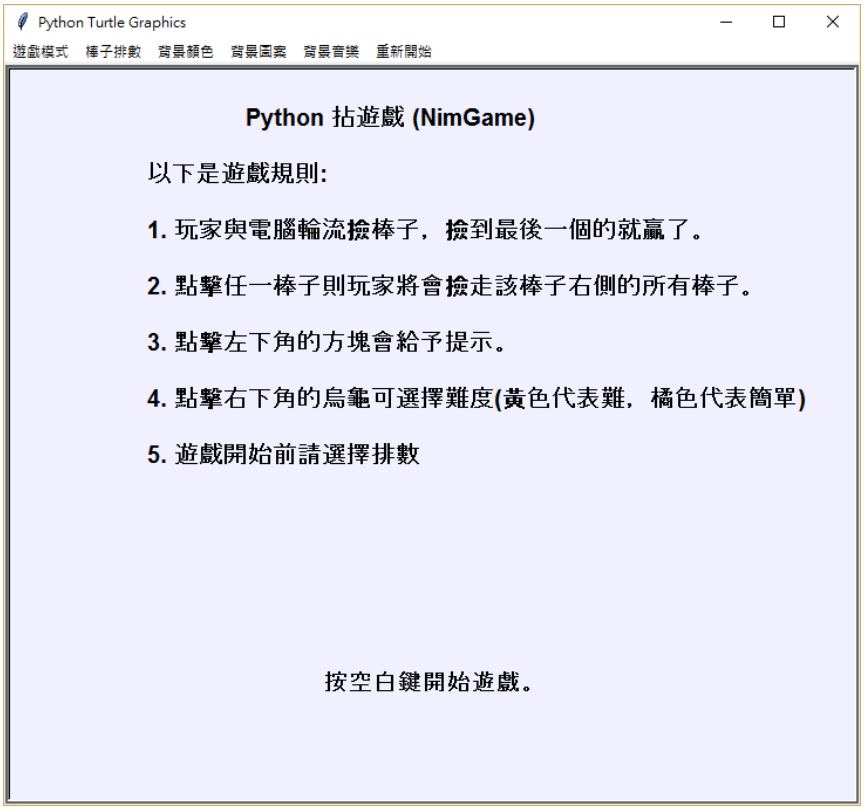

圖 3　　文字訊息

iii.　其它類別功能

1. Stick

棒子的實體，實現棒子所有的屬性、功能，以及利用內部的一些函數讓自身屬性能夠於其他地方使用。

```
1  class Stick(Turtle):
2
3      def __init__(self, row, col, game):
4          Turtle.__init__(self, visible=False)
5          self.row= row
6          self.col= col
7          self.game= game
8          self.nRows= self.game.nRows
9          x, y= self.coords(row, col)
```

```
10
11          self.shape("square")
12          a = 25.0
13          self.shapesize(HUNIT/a, WUNIT/25.0)
14          self.speed(0)
15          self.pu()
16          self.goto(x,y)
17          self.color('gray')
18      def coords(self, row, col):
19          packet, remainder = divmod(col, 5)
20          x = (3 + 11 * packet + 2 * remainder) * WUNIT
21          y = (row) * HUNIT
22          x0=  x -SCREENWIDTH//2   + WUNIT//2
23          y0= -y +SCREENHEIGHT//2  - HUNIT//2
24          return x0, y0
25
26      def makemove(self, x, y):
27          if self.game.state != NimGame.RUNNING:
28              return
29          self.game.controller.notify_move(self.row, self.col)
30
31      def playermove(self, x, y):
32          if self.game.state != NimGame.RUNNING:
33              return
34          self.game.model.player_move(self.row, self.col)
```

2. Level

為一按鈕，以滑鼠點擊的方式改變電腦的演算法處理方式，分為以必勝法則選取棒子，或是隨機選取。內部的屬性都以繼承 Turtle 的方式來呈現，並且建立一個函數 changevalue 來改變自身的值，讓其它地方能夠偵測到自身被點擊。

```
1  class Level(Turtle):
2      def __init__(self, game=None, x=0, y=0):
3          Turtle.__init__(self, visible=False)
4          self.value = True
5          self.shape('turtle')
6          self.shapesize(HUNIT/15, WUNIT/5)
7          self.speed(0)
8          self.left(90)
9          self.pu()
10         self.goto(x,y)
11         self.color('yellow')
12         self.onclick(self.changevalue)
13     def changevalue(self, x, y):
14         if(self.value==True):
15             self.value = False
16             self.color('orange')
17         else:
18             self.value = True
```

```
19              self.color('yellow')
```

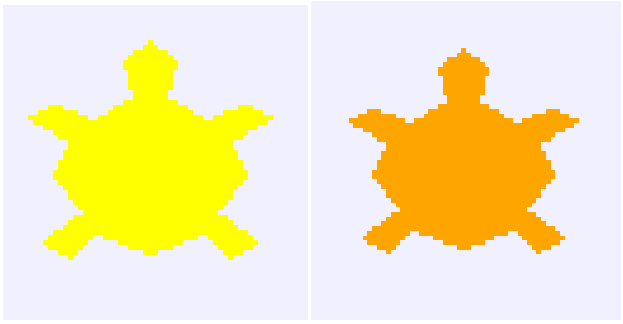

圖 4　　兩種不同型態的 Level

3. Hint

為一按鈕，當其被點擊時，螢幕會出現提示，讓玩家能夠輕鬆打敗電腦。這裡採用點擊不放才會出現指令，因此需要兩個不同的函數分別偵測點擊與放開改變其值。

```
1   class Hint(Turtle):
2       def __init__(self, game, x=0, y=0):
3           Turtle.__init__(self, visible=False)
4           self.game = game
5           self.value = True
6           self.shape('square')
7           self.shapesize(HUNIT/15, WUNIT/5)
8           self.speed(0)
9           self.pu()
10          self.goto(x,y)
11          self.color('blue')
12          self.onclick(self.changevalue)
13          self.onrelease(self.changevalue02)
14      def changevalue(self, x, y):
15          if(self.value==True):
```

```
16              self.value = False
17              self.color('violet')
18  self.game.view.writer.goto(0,-100)
19              move = computerzug(self.game.model.sticks)
20              z= '提示：第 {} 排，拿到剩下 {} 個。'.format(move[0]+1,
21  move[1])self.game.view.writer.write(z,align="center",
22  font=("Courier",14,"bold"))
23      def changevalue02(self, x, y):
24          if(self.value == False):
25              self.value = True
26              self.color('blue')
27              self.game.view.writer.undo()
```

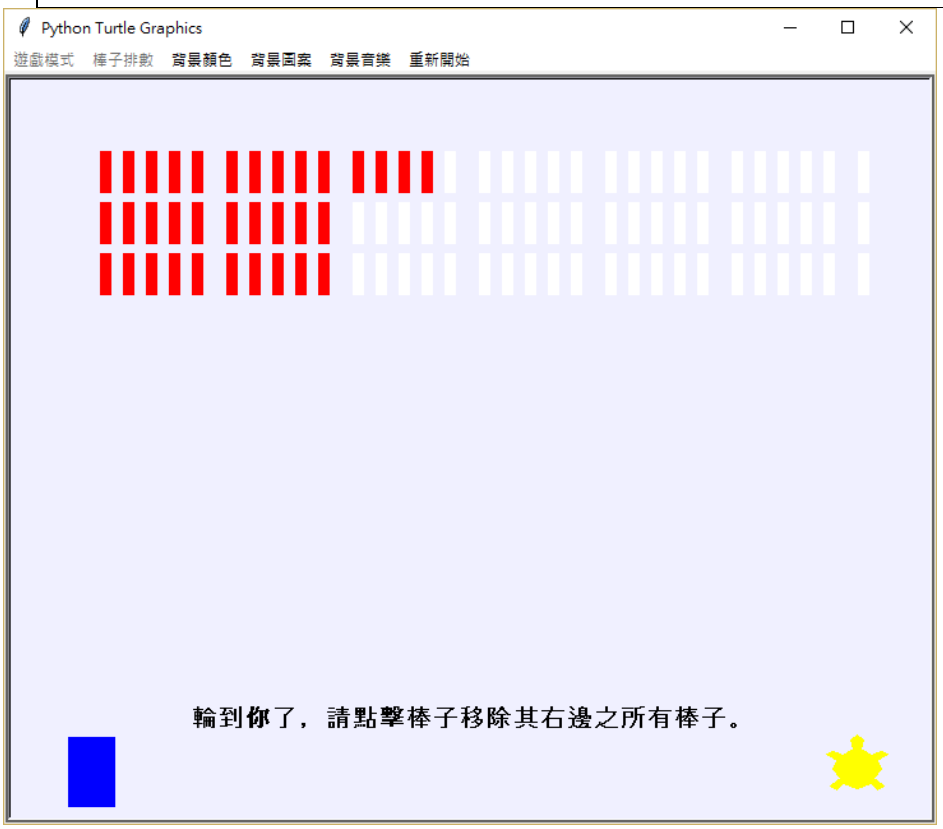

圖 5　Hint 按鈕(點擊前)

圖 6　Hint 按鈕(點擊後)

4. CvMenu

遊戲上排的選單列，所有的選單屬性及功能都在此控制，這部分使用了 TK 模組來實現。

(1) 雙人模式

增加雙人模式的功能，當模式改變後，遊戲會重新開始，是因為這部分屬於底層的更改，許多部份都必須要重新再執行一遍。

```
1  bMenu = Menu(aMenu, tearoff=0)
2          #bLabel = ['單人模式', '雙人模式']
3          bLabel = {'單人模式': 0,
4                    '雙人模式': 1}
5          keyL = sorted(bLabel)
```

```
6           for k in keyL:
7               def cmd(x=k):
8                   nRows = game.nRows
9                   mode = bLabel[x]
10                  aNimGame= NimGame(nRows, mode)
11              bMenu.add_command(label=k, command=cmd)
```

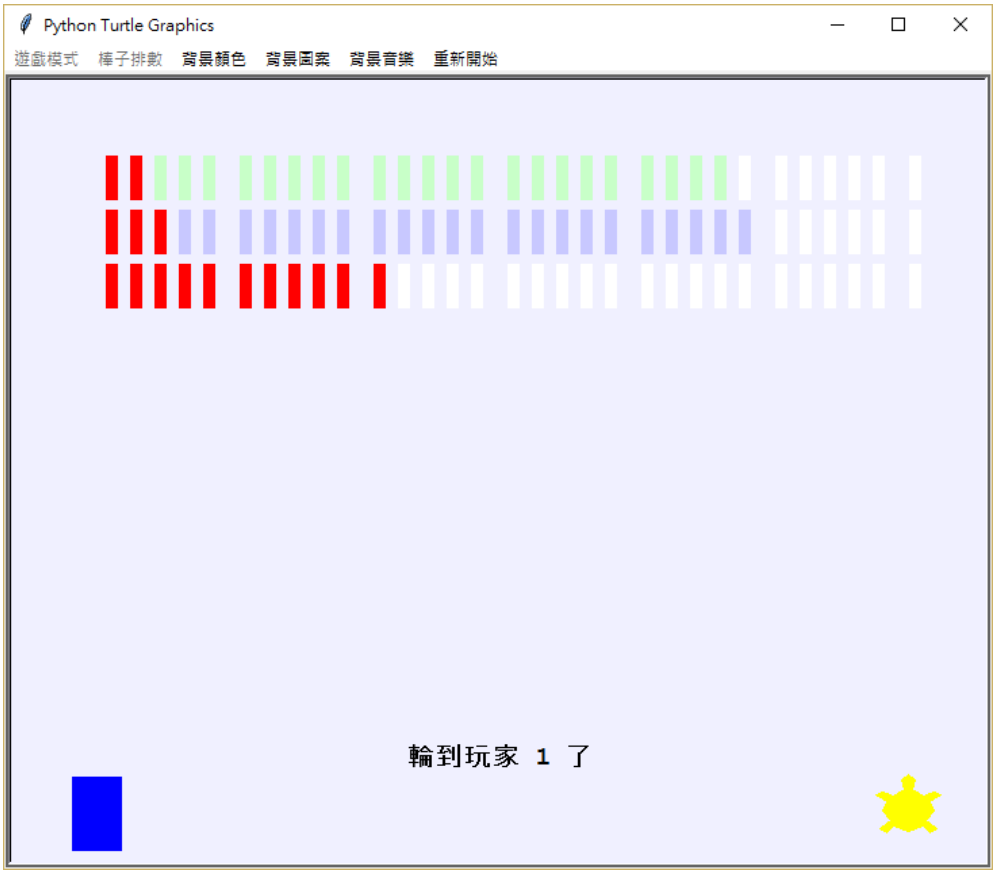

圖 7　雙人模式

(2) 更改排數

建立一個列表將排數分別以字串的形式輸入進選單列，同樣的更改後也會使遊戲重新開始。

```
1   cMenu = Menu(aMenu, tearoff=0)
2           cLabel = ['3', '4', '5', '6', '7', '8', '9', '10']
3           for i in cLabel:
```

```
4           def cmd(x=i):
5               nRows = int(x)
6               mode = game.mode
7               aNimGame= NimGame(nRows, mode)
8           cMenu.add_command(label=i, command=cmd)
```

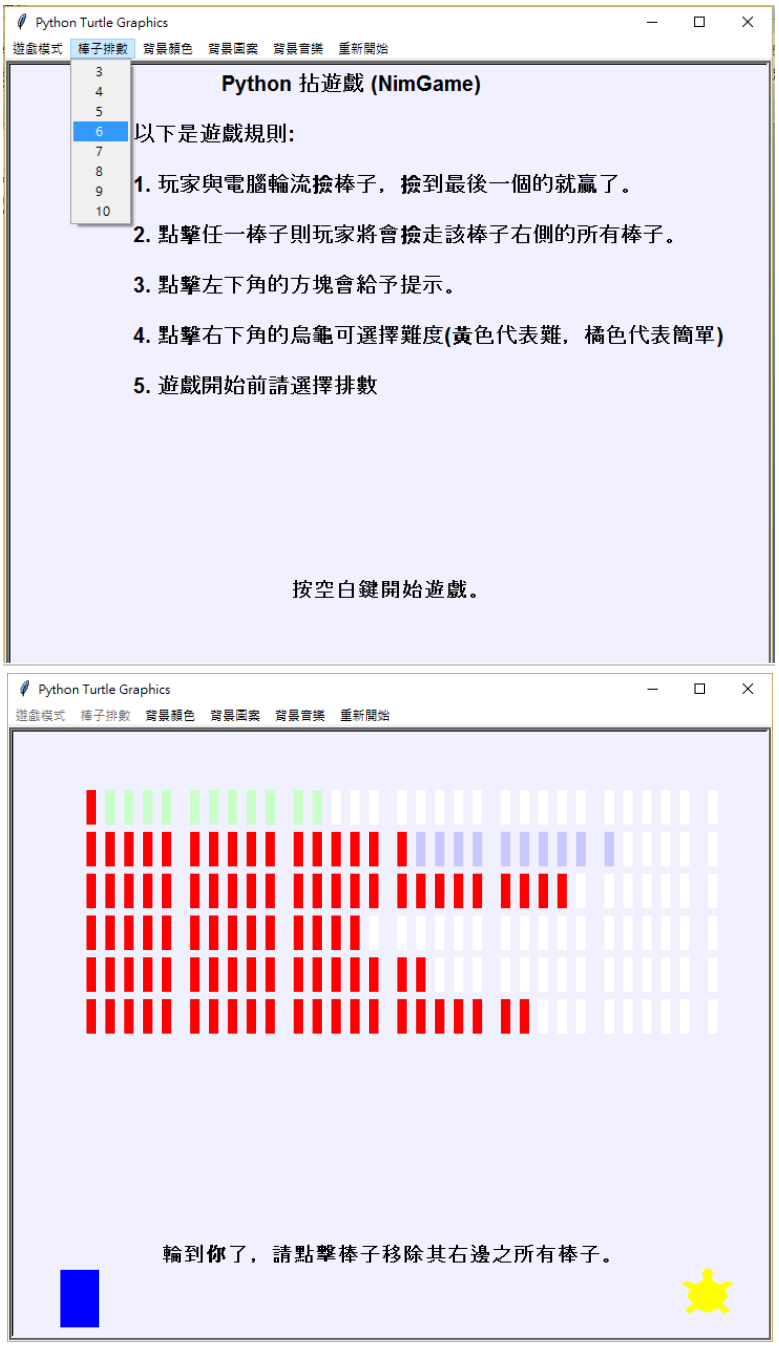

圖 8　更改排數

(3) 背景顏色及圖案

設定了四種不同的背景顏色及圖案，這部分使用了一種叫字典(dictionary)的資料處存型態，可使用文字與屬性對應。

```
dMenu = Menu(aMenu, tearoff=0)


        dLabel= {'顏色 1': '#F0F0FF',
                 '顏色 2': '#DDDDDD',
                 '顏色 3': '#FFFFBB',
                 '顏色 4': '#EEFFBB'}
        keyL = sorted(dLabel)
        for k in keyL:
            def cmd(x=k):
                game.screen.bgcolor(dLabel[x])
            dMenu.add_command(label= k, command= cmd)
        #-------------------------------------------------
        eMenu = Menu(aMenu, tearoff=0)

        eLabel = {'無圖案': 'nopic',
                  '圖案 1': 'pictures/background.png',
                  '圖案 2': 'pictures/dog01.png',
                  '圖案 3': 'pictures/calvin.png'}
        keyL = sorted(eLabel)
        for k in keyL:
            def cmd(x=k):
                game.screen.bgpic(eLabel[x])
```

```
23          eMenu.add_command(label = k, command=cmd)
```

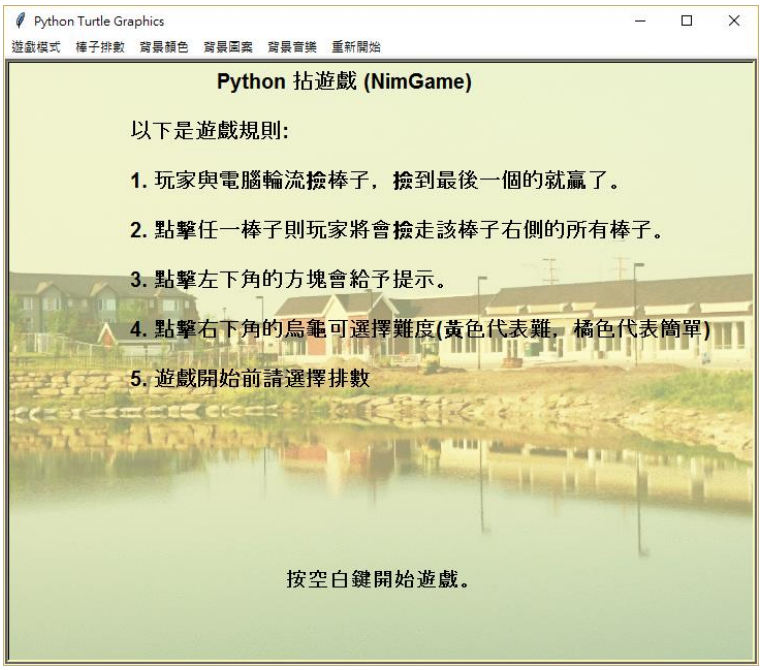

圖 9　不同的背景顏色與圖案

(4) 背景音樂

音樂的部分利用到了 pygame 模組，內部的 mixer()函數用來控制音樂的開始、切換、以及停止。

```
1   fMenu = Menu(aMenu, tearoff=0)
2         fLabel= {'音樂停止': '__0__.mid',
3                  '音樂 01': 'musics/background.mid',
4                  '音樂 02': 'musics/canon.mid',
5                  '音樂 03': 'musics/John_Cena.mp3'}
6         keyL = sorted(fLabel)
7         for k in keyL:
8             def cmd(x=k):
9                 musicFile= fLabel[x]
```

```
10              if musicFile != '__0__.mid':
11                  pygame.mixer.music.load(musicFile)
12                  pygame.mixer.music.play(-1, 0.0)
13              else:
14                  pygame.mixer.music.stop()
15          fMenu.add_command(label= k, command= cmd)
```

(5) 遊戲重新開始

使遊戲重新開始。

```
1  def cmd():
2          aNimGame= NimGame(game.nRows, game.mode)
3      aMenu.add_command(label= '重新開始', command= cmd)
4
5      game.tk.config(menu= aMenu)
```

肆、 結論與未來展望

一、 結論

經過了一年的專題研究課程，相信自己已經成長了不少，許多能力包括寫程式、查資料、做簡報以及寫小論文都有所提升。在一開始，尋找自己的主題以及研究方向困擾了我很長一段時間，這遠比開始著手進行寫程式還要困難許多。然而到了最後，看到自己所創作出來的作品，就覺得這一切的努力都值得了。

二、 未來展望

- 增加更多玩家模式
- 新的遊戲規則設計
- 繼續修改程式錯誤
- 美化遊戲介面

伍、 參考資料

一、 網路資訊

Nim - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Nim

Exclusive or - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Exclusive_or

拈及其各種變形遊戲

http://episte.math.ntu.edu.tw/articles/mm/mm_03_2_02/

rykids

http://rykids.blogspot.tw/

二、 參考書籍

Chris Roffey(2014). *Python: Next Steps*

Sakis Kasamapalis(2015.08). *Python 設計模式深入探討*

Brett Slatkin(2015.08). *Effective Python*

張凱慶(2014.08). *Python 入門指南*

三、報告之投影片：(下頁開始)

四、完整程式列表：(列於報告之投影片之後)