Tetris 線上遊戲專題報告

環境設置

我們為了能在網頁中運行不同環境製作的小遊戲我們必須先在網頁的 cord 中載入所需的程式庫並且要啟動。

```
<!-- 導入 Brython 程式庫 -->
<script src="/static/brython.js"></script>
<script src="/static/brython_stdlib.js"></script>
```

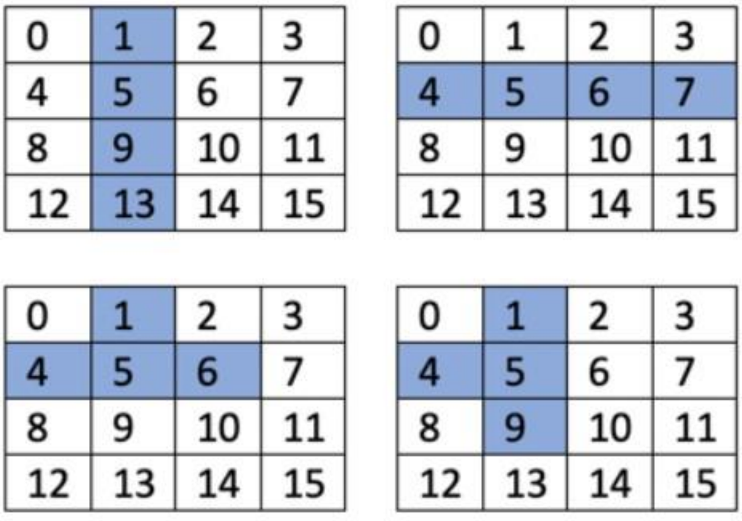
先將已存在於近端中的 Brython 程式庫導入網頁的頁面中，讓我們小遊戲中的程式碼能夠順利運作

```
<!-- 啟動 Brython -->
<script>window.onload=function(){brython();}</script>
```

導入程式庫後再來就是啟動 Brython，這樣在使用網頁時就能讀取到小遊戲

方塊圖形設定

我們將要製作的圖形以數字的矩陣列出圖形的範圍包括圖形選轉之後的矩陣一同排列出來。



```
figures = [
    [[1, 5, 9, 13], [4, 5, 6, 7]],
    [[4, 5, 9, 10], [2, 6, 5, 9]],
    [[6, 7, 9, 10], [1, 5, 6, 10]],
    [[1, 2, 5, 9], [0, 4, 5, 6], [1, 5, 9, 8], [4, 5, 6, 10]],
    [[1, 2, 6, 10], [5, 6, 7, 9], [2, 6, 10, 11], [3, 5, 6, 7]],
    [[1, 4, 5, 6], [1, 4, 5, 9], [4, 5, 6, 9], [1, 5, 6, 9]],
    [[1, 2, 5, 6]],
]
```

選取數字排列出的圖形就是我們想要的圖形，這邊先完成了我們小遊戲中會使用到的所有方塊圖形。

我們在進行俄羅斯方塊的小遊戲時方塊的掉落是隨機的所以我們在設定方塊的掉落時必須加入隨機的元素。

```
class Figure:
    ...
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.type = random.randint(0, len(self.figures) - 1)
        self.color = random.randint(1, len(colors) - 1)
        self.rotation = 0
```

這邊使用隨機產生並將方塊顏色及方塊的類型設為隨機產生的內容，這樣就能有不同顏色及不同方塊類型隨機產生。

```
class Figure:
    ...
    def image(self):
        return self.figures[self.type][self.rotation]

    def rotate(self):
        self.rotation = (self.rotation + 1) %
len(self.figures[self.type])
```

我們為了要能夠快速得到方塊圖形翻轉後的圖形，產生方塊後綁定設定回傳的圖形為下一個同一形狀但是不同方向的圖形(方塊設定時的陣列)。

遊戲設定

方塊的設定後進行遊戲本體的編寫，先用一些變量設定遊戲的初始狀態。

```
class Tetris:
    level = 2
    score = 0
    state = "start"
    field = []
    height = 0
    width = 0
    x = 100
    y = 60
    zoom = 20
    figure = None
```

其中的 field 我們將作為是否有方塊的判定，有方塊時就會有顏色也就代表不是空的部分。

```
class Tetris:
    ...
    def __init__(self, height, width):
        self.height = height
        self.width = width
        for i in range(height):
            new_line = []
            for j in range(width):
                new_line.append(0)
            self.field.append(new_line)
```

將遊戲的狀態重整，準備進行新的遊戲進行。

```
class Tetris:
    ...
    def new_figure(self):
        self.figure = Figure(3, 0)
```

創建新方塊時方塊的起始位置設定。

```
class Tetris:
    ...
    def intersects(self):
        intersection = False
        for i in range(4):
            for j in range(4):
                if i * 4 + j in self.figure.image():
                    if i + self.figure.y > self.height - 1 or \
                            j + self.figure.x > self.width - 1 or \
                            j + self.figure.x < 0 or \
                            self.field[i + self.figure.y][j +
self.figure.x] > 0:
                        intersection = True
        return intersection
```

為了確定方塊在移動或是旋轉的過程中並沒有與其他方塊或是邊界相互有接觸，我們檢測方塊的 4x4 的矩陣範圍是否有重疊的範圍。
有了這個功能也就能夠限制方塊在有其他方塊時的移動並在疊在其他方塊時固定方塊的位置。

```
class Tetris:
    ...
    def freeze(self):
        for i in range(4):
            for j in range(4):
                if i * 4 + j in self.figure.image():
                    self.field[i + self.figure.y][j + self.figure.x] =
self.figure.color
        self.break_lines()
        self.new_figure()
        if self.intersects():
            game.state = "gameover"
```

檢測固定的方塊有相交時表示遊戲結束。

```
class Tetris:
    ...
    def break_lines(self):
        lines = 0
        for i in range(1, self.height):
            zeros = 0
            for j in range(self.width):
                if self.field[i][j] == 0:
                    zeros += 1
            if zeros == 0:
                lines += 1
                for i1 in range(i, 1, -1):
                    for j in range(self.width):
                        self.field[i1][j] = self.field[i1 - 1][j]
        self.score += lines ** 2
```

當有一行的方塊被填滿我們就要進行消除

```
class Tetris:
    ...
    def go_space(self):
        while not self.intersects():
            self.figure.y += 1
        self.figure.y -= 1
        self.freeze()

    def go_down(self):
        self.figure.y += 1
        if self.intersects():
            self.figure.y -= 1
            self.freeze()

    def go_side(self, dx):
        old_x = self.figure.x
        self.figure.x += dx
        if self.intersects():
            self.figure.x = old_x

    def rotate(self):
        old_rotation = self.figure.rotation
        self.figure.rotate()
        if self.intersects():
            self.figure.rotation = old_rotation
```

最後對方塊的移動方式進行編寫。

參考資料

https://levelup.gitconnected.com/writing-tetris-in-python-2a16bddb5318

https://mde.tw/wcm2022_guide/content/w14%20Tetris.html