Programming Project assignment #2

107062313 黃寶萱

- Program performance

  從右方附圖可以看出原本 MCST 的 performance 仍不是最完美的，如果 Player1 在第 21 round 就選擇 best move: (2, 5)便可以取得勝利，不需要再多花兩 round 的時間才獲勝(play#23: Player 1 wins!)。

```
*
play# 19 , Player 1 Best Move: (5, 5)
0000000
0101110
0022211
2022000
1002010
0010020
0000210

No one wins yet
*
play# 20 , Player 2 Best Move: (5, 3)
0000000
0101110
0022211
2022020
1002010
0010020
0000210

No one wins yet
*
play# 21 , Player 1 Best Move: (5, 6)
0000010
0101110
0022211
2022020
1002010
0010020
0000210

No one wins yet
```

而在下圖的例子中，Player 2 在 round 22 便已形成 4-in-line 的局勢，如果能在下一輪 best move: (5, 0)即可獲得勝利，但最後卻是 Player 1 wins，並且從最後的 board 可以看出 Player 1 也沒有試圖阻擋 Player 2 獲得勝利，(5, 0)依舊是空的，顯示出 MCTS 仍有改善的空間。

```
*
play# 22 , Player 2 Best Move: (2, 3)
0000012
1102000
1222111
0220100
0012210
2000200
0100000

No one wins yet
```

```
*
play# 29 , Player 1 Best Move: (5, 5)
0000012
1102010
1222111
1221120
0012210
2102200
0100200

Player 1 wins!
```

右圖的情況中，可以清楚看到中間區域的部分 Player 2 有多種可能可以讓他連成 4-in-lin，或是形成 three-in-line 的局勢，但 Player 2 在 round 16 卻選擇一個不太容易連成線的位置: (4, 0)，直接放棄中間區域的布局，最後則是由 Player 1 wins。

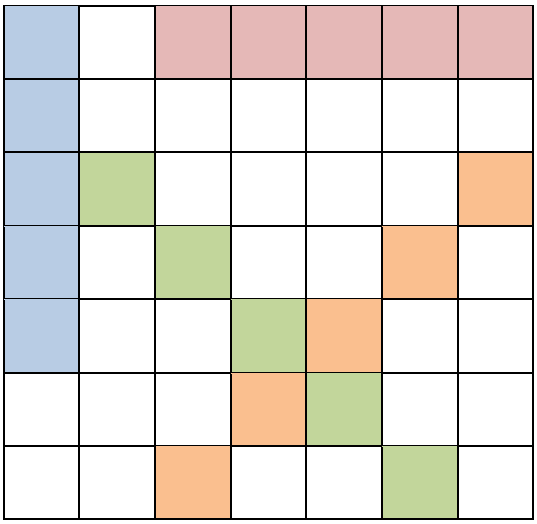
```
*
play# 14 , Player 2 Best Move: (2, 3)
0000000
0101100
0020210
2022000
0002010
0010020
0000010

No one wins yet
*
play# 15 , Player 1 Best Move: (6, 4)
0000000
0101100
0020211
2022000
0002010
0010020
0000010

No one wins yet
*
play# 16 , Player 2 Best Move: (4, 0)
0000000
0101100
0020211
2022000
0002010
0010020
0000210

No one wins yet
```

而此右圖的例子中，Player 2 則很好的判斷出 three-in-line 的情況，並在 round 14 best move: (3, 5)形成 four-in-line，下一次輪到 Player 2 時，他也直接選擇了 best move: (3, 6)贏得勝利，很好得判斷出正確的位置。

```
*
play# 14 , Player 2 Best Move: (3, 5)
0000010
0002010
0012000
0002010
2002220
0010000
0001001

No one wins yet
*
play# 15 , Player 1 Best Move: (1, 4)
0000010
0002010
0112000
0002010
2002220
0010000
0001001

No one wins yet
*
play# 16 , Player 2 Best Move: (3, 6)
0002010
0002010
0112000
0002010
2002220
0010000
0001001

Player 2 wins!
```

經過多次的測試後，最大的問題是 player 沒辦法很好的判斷出 three-in-line 或 four-in-line 的情況而錯失了許多提早結束遊戲的可能，有時甚至錯失贏得勝利的機會，而導致對手獲勝。

- function GetResult()：利用雙層 for loop 檢測 7x7 board 上的每個座標是否有五顆棋子連成一線，透過呼叫 function IsFiveInLine(x, y)來檢測。
- function IsFiveInLine(x, y)：以座標(x, y)為中心檢測所有可能連成一線的方式，我將它分為四種檢測方式(如下圖)：
  - 直向連成一線：function **checkVertical()**
  - 橫向連成一線：function **checkHorizontal()**
  - 兩種不同方向的對角連線：function **checkUpSlash()**、function **checkDownSlash()**

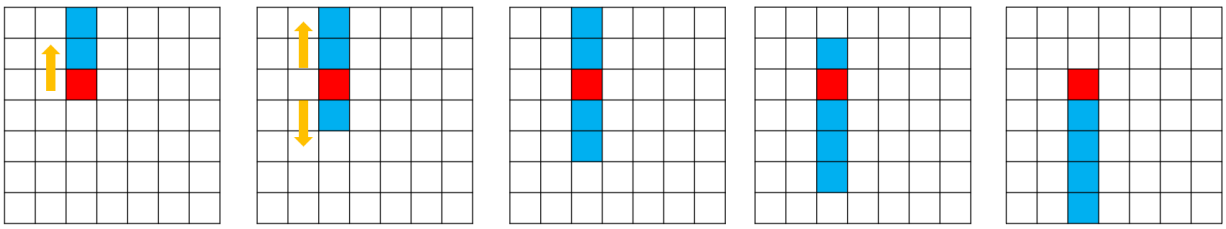


```
def IsFiveInLine(self, x, y):
    player = self.playerJustMoved

    if self.checkVertical(x, y, player)==True:
        return True
    if self.checkHorizontal(x, y, player)==True:
        return True
    if self.checkUpSlash(x, y, player)==True:
        return True
    if self.checkDownSlash(x, y, player)==True:
        return True

    return False
```

一旦在任一方向上確定有連成一線，即馬上 return True，可以省下其他方向不必要的檢測時間，如果四個方向都沒有連成線，則 return False，讓 function GetResult()繼續下一次 iteration 的檢測，此外，在這個 function 中也會用變數 player 紀錄遊戲到目前為止最後一個擺放旗子的人是誰(1 or 2)。

- function checkVertical(x, y, player)：

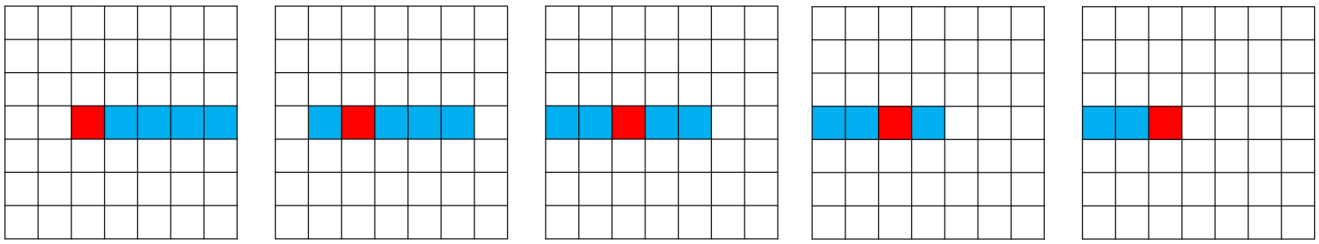
  當 function IsFiveInLine(x, y)呼叫 checkVertical(x, y, player)會把要檢測的位置座標(x, y)及剛移動的 player 傳送到這個 function，檢測方式分成 5 個步驟：



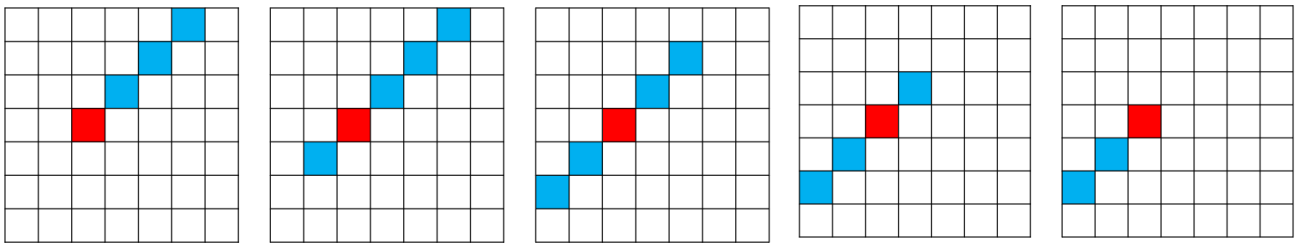
  利用 function IsOnBoard(x, y+dy) 判斷現在要檢測的位置(上圖藍色框框)是否在 7x7 board 範圍內，同時也會檢測中心點(上圖紅色框框)以及可能連成一線位置(上圖藍色框框)在 7x7 board 上是否與 player 為同一人，如果為同一人則會以對應的變數(v1, v2, v3, v4, v5)來記錄連續棋子個數，一旦檢測發現有五棵棋子連成一線，代表有 player 勝利便馬上 return True，不用繼續完成後續其他可能的檢查。

```python
def checkVertical(self, x, y, player):
    # print("vertical: ", player)
    v1 = 1
    v2 = 1
    v3 = 1
    v4 = 1
    v5 = 1
    for (dx, dy) in [(0, +1), (0, +2), (0, +3), (0, +4)]:
        if self.IsOnBoard(x, y+dy) and self.board[x][y]==player and self.board[x][y+dy]==player:
            v1 += 1
            if v1==5:
                return True
    for (dx, dy) in [(0, -1), (0, +1), (0, +2), (0, +3)]:
        if self.IsOnBoard(x, y+dy) and self.board[x][y]==player and self.board[x][y+dy]==player:
            v2 += 1
            if v2==5:
                return True
    for (dx, dy) in [(0, -2), (0, -1), (0, +1), (0, +2)]:
        if self.IsOnBoard(x, y+dy) and self.board[x][y]==player and self.board[x][y+dy]==player:
            v3 += 1
            if v3==5:
                return True
    for (dx, dy) in [(0, -3), (0, -2), (0, -1), (0, +1)]:
        if self.IsOnBoard(x, y+dy) and self.board[x][y]==player and self.board[x][y+dy]==player :
            v4 += 1
            if v4==5:
                return True
    for (dx, dy) in [(0, -4), (0, -3), (0, -2), (0, -1)]:
        if self.IsOnBoard(x, y+dy) and self.board[x][y]==player and self.board[x][y+dy]==player:
            v5 += 1
            if v5==5:
                return True
    return False
```

- 同理，function checkHorizontal()、checkUpSlash()、checkDownSlash()也是以同樣的作法檢測，下方以圖示表示。
  - checkHorizontal()



  - checkUpSlash()



- checkDownSlash()