Othello Text-Based Game Documentation

คำนำ :

- Code นี้สร้างมาเพื่อทำให้สามารถเล่น othello ได้ใน terminal(text based) เขียนด้วย Javascript (othello คือ boardgame 8*8 ที่ให้ผู้เล่นทั้งสองฝ่ายผลัดกันวางหมากบนกระดานและหากหมากของฝ่ายตรงข้ามถูกล้อม หมากตัวนั้นจะถูกเปลี่ยนเป็นของฝั่งล้อม)

requirements :

- ต้องมี git และ nodejs ในเครื่อง
- Git
- Node.js

Download :

- เปิด terminal/cmd แล้วรันโค้ดต่อไปนี้

```
cd path ของ folder ที่ต้องการ clone งานนี้
git clone https://github.com/lemuruz/othello_text.git
cd othello_text
npm install
```

Run the game:

- เปิด terminal/cmd แล้วรันโค้ดต่อไปนี้

```
cd 'path ของ folder ที่ clone งานนี้ไว้' (เช่น /media/user/HDD)
node gamev2.js
```

Improvement from v1:

- เพิ่ม bot เข้ามาในเกม
- ตกแต่งกระดานกับหมากให้สวยและมองง่ายมากขึ้น
- จัดระเบียบ code ทำให้เข้าใจง่ายขึ้น

Code structure

- *Class BOARD*: จัดการ ตัวเกมั้งหมด logic
  - printBoard()
  - switchPlayer()
  - isValidMove(coordinates)
  - placePiece(x, y)
  - checkBoard()
  - gameOver()
  - playGame()
- *Class PLAYER*: เก็บtoken ของผู้เล่น รับ input
  - playerInput()
- *Class BOT*: จัดการlogic ของ bot return ตำแหน่ง
  - bestMove(board)

Detailed

- Class BOARD

```
class BOARD {
    constructor(player2) {
        this.board = Array.from({ length: 8 }, () => Array.from({ length: 8
}, () => "▧"));
        this.player1 = new PLAYER('○');
        this.player2 = player2;
        this.currentPlayer = this.player1;
        this.board[3][3] = this.player1.playerToken;
        this.board[3][4] = this.player2.playerToken;
        this.board[4][3] = this.player2.playerToken;
        this.board[4][4] = this.player1.playerToken;

        this.playGame();
}}
```

- module printBoard() แสดงกระดานทุกครั้งที่เริ่มเกม และ มีผู้เล่นวางหมาก โดยการใช้ for loop เพื่อทำให้สามารถบรับแต่งกระดานได้ง่ายขึ้น

```
printBoard() {
    console.clear()
    this.checkBoard();
    console.log("  a b c d e f g h");
    console.log("  ================");
    for (let i = 0; i < 8; i++) {
        let row = "";
        for (let j = 0; j < 8; j++) {
            row += this.board[i][j] + "";
        }
        console.log(i + '|' + row.trim());
    }
}
```

- module switchPlayer() : ทำการเปลี่ยน currentPlayer เป็นอีฝั่งนึง

```
switchPlayer() {
    this.currentPlayer = this.currentPlayer === this.player1 ? this.player2
: this.player1;
}
```

- module isValidMove(coordinates) : รับค่าตำแหน่งจากผู้เล่น แล้วทำการเช็คว่า ตำแหน่งนั้นๆสามารถวางได้หรือไม่

```
isValidMove(coordinates) {
    if (!coordinates) {
        return false;
    }
    //เปลี่ยนตัวหนังสือเป็นตัวเลข
    let col = coordinates[1].charCodeAt(0) - 'a'.charCodeAt(0);

    if (this.board[coordinates[0]][col] === '⭕') {
        this.placePiece(parseInt(coordinates[0]), col);
        return true;
    } else {
        console.log('This coordinate is invalid, please select "⭕"');
        return false;
    }
}
```

- module placePiece(x, y) : ถูกเรียกใช้โดย module isValidMove(coordinates) ,รับค่าเป็นตำแหน่ง แล้วทำการวางหมากในตำแหน่งที่รับมา จากนั้นทำการเช็ครอบตำแหน่งที่จะทำการวาง(8ทิศ)(เช็คจนสุดกระดานของแต่ละทิศ)หากเจอหมากฝั่งตัวเองอยู่ในทิศนั้นๆจะทำการกลับด้านหมากฝั่งตรงข้ามที่อยู่ระหว่างหมากฝ่ายเรา

```
placeToken(x, y) {
    let board = this.board;
    let opposite = this.currentPlayer === this.player1 ?
this.player2.playerToken : this.player1.playerToken;
    let currentPlayer = this.currentPlayer.playerToken;
    let checkDirection = [[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0]];

    for (let direction of checkDirection) {
        let [dix, diy] = direction;
        flipDirection(x, y, dix, diy, opposite, currentPlayer);
}
    this.board[x][y] = currentPlayer;


    function flipDirection(current_x, current_y, direction_x, direction_y,
opposite, currentplayerToken) {
        let positions_to_flip = [];
        let checking_x = current_x + direction_x;
        let checking_y = current_y + direction_y;

        while (checking_x >= 0 && checking_x < 8 && checking_y >= 0 &&
checking_y < 8) {
            if (board[checking_x][checking_y] === '▧' || board[checking_x]
[checking_y] === '◯') {
                break;
            } else if (board[checking_x][checking_y] === opposite) {
                positions_to_flip.push([checking_x, checking_y]);
            } else if (board[checking_x][checking_y] ===
currentplayerToken) {
                for (let pos of positions_to_flip) {
                    board[pos[0]][pos[1]] = currentplayerToken;
                }
                return;
            }
            checking_x += direction_x;
            checking_y += direction_y;
        }
    }
}
```

- module checkBoard() : ทำการเช็คว่าตำแหน่งไหนเป็นตำแหน่งที่สามารถวางได้บ้าง โดยการรันloopเช็คทุกช่องบนกระดานและทำการเช็ครอบตำแหน่งนั้นๆ(8ทิศ)(เช็คจนสุดกระดานของแต่ละทิศ)หากมีหมากฝ่ายเราอยู่ และ มีแต่หมากฝั่งตรงข้ามที่อยู่ระหว่างหมากฝ่ายเรา(ไม่มีช่องว่าง) กำหนดให้ตำแหน่งที่เช็คอยู่สามารถวางได้โดยการแทนที่ตำแหน่งบนกระดานด้วยวงกลมสีแดง

```
checkBoard() {
    let board = this.board;
    for (let i = 0; i < 8; i++) {
        for (let j = 0; j < 8; j++) {
            if (this.board[i][j] === '🟢') {
                this.board[i][j] = '▧';
            }
        }
    }
    let hasEmptySpace = false;
    let checkDirection = [[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0]];
    let opposite = this.currentPlayer === this.player1 ?
this.player2.playerToken : this.player1.playerToken;
    let currentPlayer = this.currentPlayer.playerToken;

    for (let i = 0; i < 8; i++) {
        for (let j = 0; j < 8; j++) {
            if (this.board[i][j] === '▧') {
                let valid = false;
                for (let direction of checkDirection) {
                    let [dix, diy] = direction;
                    if (checkIfThatDirectionValid(i, j, dix, diy, opposite,
currentPlayer)) {
                        valid = true;
                        break;
                    }
                }
                if (valid) {
                    hasEmptySpace = true;
                    this.board[i][j] = '🟢';
                }
            }
        }
    }
    if (!hasEmptySpace) {
        this.gameOver();
    }
    function checkIfThatDirectionValid(current_x, current_y, direction_x,
direction_y, opposite, currentToken) {

        let checking_x = current_x + direction_x;
        let checking_y = current_y + direction_y;
        let opponent_detected = false;
```

```
        while (checking_x >= 0 && checking_x < 8 && checking_y >= 0 &&
checking_y < 8) {
            if (board[checking_x][checking_y] === '▧' || board[checking_x]
[checking_y] === '◯') {
                return false;
            } else if (board[checking_x][checking_y] === opposite) {
                opponent_detected = true;
            } else if (board[checking_x][checking_y] === currentToken &&
opponent_detected) {
                return true;
            } else {
                return false;
            }
            checking_x += direction_x;
            checking_y += direction_y;
        }
        return false;
    }
}
```

- gameOver() : ถูกเรียกใช้เมื่อจบเกม,ทำการเช็คคะแนนของแต่ละฝั่งและแสดงผลแพ้ชนะ จากนั้นหยุดการทำงานของเกม

```
    gameOver() {
        let player1Count = 0;
        let player2Count = 0;
        gameIsRunning = false;
        console.clear();

        for (let i = 0; i < 8; i++) {
            for (let j = 0; j < 8; j++) {
                if (this.board[i][j] == this.player1.playerToken) {
                    player1Count += 1;
                } else if (this.board[i][j] == this.player2.playerToken) {
                    player2Count += 1;
                }
            }
        }
        console.log('Game over');
        console.log('Player 1', this.player1.playerToken, 'score:',
player1Count);
        console.log('Player 2', this.player2.playerToken, 'score:',
player2Count);
        if (player1Count > player2Count) {
            console.log('Player 1 WINS!');
        } else if (player1Count < player2Count) {
            console.log('Player 2 WINS!');
        } else {
            console.log('TIE!');
        }
}
```

- module playGame() : ใชในการรันเกมโดยการใช้while loop และทำการสั่งให้ PLAYER หรือ BOT ให้รับ/ส่งตำแหน่ง

```
//async ใช้ในการ delay การทำงานของบอท
async playGame() {
    while (gameIsRunning) {
        this.printBoard();
        let input = null;
        if (this.currentPlayer instanceof PLAYER) {
            input = this.currentPlayer.playerInput();
        } else {
            console.log('Bot turn');
            await delay(1500);
            input = this.currentPlayer.bestMove(this.board);
        }

        if (this.isValidMove(input)) {
            this.switchPlayer();
        }
    }
}
```

- Class PLAYER

```
class PLAYER {
constructor(token) {
    this.playerToken = token;
}}
```

- module playerInput() : รับค่าจากผู้เล่น,เช็คความถูกต้องของค่าที่รับมา หากไม่ถูกให้เรียกใช้ module นี้ซ้ำ จากนั้น return ค่าไปให้ BOARD

```
playerInput() {
    if (!gameIsRunning) {return false;}
    console.log('Player', this.playerToken, 'turn');
    let input = prompt('Enter your move (e.g., 0a): ');
    if (input.length === 2 && !isNaN(input[0]) && input[0] >= '0' &&
input[0] <= '7' && input[1] >= 'a' && input[1] <= 'h') {
        return input;
    } else if (input === 'exit') {
        gameIsRunning = false;
        return false;
    } else {
        console.log('Invalid input');
        return this.playerInput();
    }
}
```

- Class BOT

```
class BOT {
    constructor(token, difficulty) {
        this.playerToken = token;
        this.difficulty = difficulty;
        this.scoreBoard = [
    [100, -10, 10, 5, 5, 10, -10, 100],
    [-10, -20, 1, 1, 1, 1, -20, -10],
    [10, 1, 3, 3, 3, 3, 1, 10],
    [5, 1, 3, 1, 1, 3, 1, 5],
    [5, 1, 3, 1, 1, 3, 1, 5],
    [10, 1, 3, 3, 3, 3, 1, 10],
    [-10, -20, 1, 1, 1, 1, -20, -10],
    [100, -10, 10, 5, 5, 10, -10, 100]
    ];}}
```

- module bestMove(board) : รับค่าจาก BOARD จากนั้นทำการคำนวนตาม logic ของ bot ที่ตั้งไว้จากนั้น return ตำแหน่งที่ได้ให้ BOARD

- bot lv1 จะเป็น bot แบบสุ่ม
- bot lv2 จะทำการเทียบตำแหน่งที่มีคะแนนมากกว่าในการวางและวางในตำแหน่งที่มีคะแนนสูงกว่า(ใน othello การวางหมากที่ขอบหรือมุดจะทำให้มีโมโอกาศถูกอีกฝ่ายกินหมากน้อยกว่า)

```
    bestMove(board) {
        let placeableCoordinates = [];
        for (let i = 0; i < 8; i++) {
            for (let j = 0; j < 8; j++) {
                if (board[i][j] === '⚪') {
                    placeableCoordinates.push(i.toString() +
String.fromCharCode('a'.charCodeAt(0) + j));
                }
            }}
        switch (this.difficulty) {
            case 'lv1':
                return placeableCoordinates.random();
            case 'lv2':
                let highestScoreCoordinates = placeableCoordinates[0];
                for (let i of placeableCoordinates) {
                    if (this.scoreBoard[highestScoreCoordinates[0]]
[highestScoreCoordinates[1].charCodeAt(0) - 'a'.charCodeAt(0)] <
                        this.scoreBoard[i[0]][i[1].charCodeAt(0) -
'a'.charCodeAt(0)]) {
                        highestScoreCoordinates = i;
                    }}
                return highestScoreCoordinates;
        }}
```