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 game.js
```

Code structure

- *Class GameMenu*: แสดงเมนูและรับ input เพื่อเริ่มเกม
- *Class GameTable*: จัดการกระดานของเกม, turn ของผู้เล่น
  - show_text_board
  - player_input
- *Class OthelloLogic*: จัดการเกี่ยวกับตรรกะของเกม (ตรวจว่าวางได้ที่ไหน, การกลับด้านหมาก, การรับ input ของผู้เล่น)
  - placeable
  - check_8_direction
  - check_if_that_posi_valid
  - flip
  - flip_direction

Detailed

- *Class Geme_menu*

```
class Game_menu {
        constructor() {
            let on_lunch_text = `Othello text base
    type 'exit while in game to exit the game'
    type 'start' to start the game`;
            console.log(on_lunch_text);

            const readline = require('readline');
            const ask = readline.createInterface({
                input: process.stdin,
                output: process.stdout
            });
            ask.question('type: ', (type) => {
                if (type.toLowerCase() == 'start') {
                    this.game_on = new Game_table();
                } else if (type.toLowerCase() == 'help') {
                    console.log('Help: This is a text-based Othello
game.');
                }
                ask.close();
            });}}
```

- *Class Game_table*

```
class Game_table {
  constructor() {
      this.player1 = "W";
      this.player2 = "B";
      this.current_turn = this.player1;
      this.running = true;
      this.row_col = 8;
      //create a table use 2d array
      this.table = Array.from({ length: this.row_col }, () => Array.from({
length: this.row_col }, () => "."));
      this.logic = new othello_logic(this.row_col);
      let middle_row_col = parseInt(this.row_col / 2) - 1;
      this.table[middle_row_col][middle_row_col] = "W";
      this.table[middle_row_col][middle_row_col + 1] = "B";
      this.table[middle_row_col + 1][middle_row_col] = "B";
      this.table[middle_row_col + 1][middle_row_col + 1] = "W";
      //กระดานที่แยกจากกระดานหลัก(เก็บจุดที่สามารถวางได้(ไม่เก็บในกระดานหลัก))
      this.logic_table = this.table;
      this.notend = true;
      this.show_text_board();
      this.player_input();}
```

- function show_text_board() : แสดงกระดานทุกครั้งที่เริ่มเกม และ มีผู้เล่นวางหมาก

```
show_text_board() {
    console.clear();
    //ตรวจการเปลี่ยนแปลงของกระดาน
    for (let i = 0; i < this.row_col; i++) {
        let row = "";
        for (let j = 0; j < this.row_col; j++) {
            if(this.logic_table[i][j] == this.player1 && this.table[i]
[j] == this.player2){
                this.table[i][j] = this.player1;
            } else if (this.logic_table[i][j] == this.player2 &&
this.table[i][j] == this.player1){
                this.table[i][j] = this.player2;
            }
        }
    }
    //

    //แสดงกระดาน
    this.logic_table = this.logic.placeable(this.table,
this.current_turn);
    console.log("  a  b  c  d  e  f  g  h");
    console.log("  ======================");

    for (let i = 0; i < this.row_col; i++) {
        let row = "";
        for (let j = 0; j < this.row_col; j++) {
            if (this.logic_table[i][j] == 'O' && this.table[i][j] ==
'.') {
                this.notend = false;
                row += this.logic_table[i][j] + "  ";
            }
            else {
                row += this.table[i][j] + "  ";
            }
        }
        console.log(i + '|' + row.trim());
        console.log("\n");
    }
    if(this.notend == true && this.logic.gameover_count > 0){
        console.log('gameover!');
    }
    //
}
```

- function player_input() : รับ input ตำแหน่งการวางหมากของผู้เล่น

```
player_input() {
    while (this.running) {
        if (this.logic.valid_move_avalable == false){
            break;
        }
        //switch turn
        if (this.current_turn == "W") {
            if (this.player1 == "W") {
                console.log("player 1 turn *", this.current_turn,
"*");
            } else {
                console.log("player 2 turn *", this.current_turn,
"*");
            }
        } else {
            if (this.player1 == "B") {
                console.log("player 1 turn *", this.current_turn,
"*");
            } else {
                console.log("player 2 turn *", this.current_turn,
"*");
            }
        }
        //recieve input
        let posi = prompt('enter your unit position ex: 0a :');
        if (posi.toLowerCase() == 'exit'){
            this.running = false;
            console.clear();
            console.log('you exit the game');
            continue;
        }
        //handle error input
        if (posi.length === 2 && !isNaN(posi[0]) && posi[0] >= '0' &&
posi[0] <= '7' && posi[1] >= 'a' && posi[1] <= 'h') {
            let row = parseInt(posi[0]);
            let col = posi[1].charCodeAt(0) - 'a'.charCodeAt(0);
            //เช็คว่าวางได้มั้ย และ วางหมากลงในกระดาน
            if (this.logic_table[row][col] == 'O') {
                this.logic_table = this.logic.flip(row, col);
                this.table[row][col] = this.current_turn;
                this.logic_table[row][col] = this.current_turn;
            //check if that spot avalable
            } else if (this.table[row][col] == '.') {
                console.log("this position isn't placeable please
select 'O' position");
            } else {
                this.show_text_board();
                console.log("this position already taken please choose
a new spot");
```

```
                continue;
            }
            //change player
            if (this.current_turn == this.player1) {
                this.current_turn = this.player2;
            } else {
                this.current_turn = this.player1;
            }
            this.show_text_board();
        } else {
            console.log('Invalid input. Please enter a valid
position.');
        }
    }
}
```

- *Class othello_logic*

```
class othello_logic {
  constructor(table_size) {
      this.table = null;
      this.check_direction = [[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0]];
      this.current_player = null;
      this.opposite = null;
      this.size = table_size;
      this.valid_move_avalable = false;
      this.gameover_count = 0;}
```

- function ในการเช็คจุดที่วางได้

- run loop เช็คกระดานทุกช่อง

```
placeable(table1, p) {
    this.valid_move_avalable = false;
    this.table = JSON.parse(JSON.stringify(table1));
    this.current_player = p;
    if (this.current_player == 'W') {
        this.opposite = 'B';
    } else {
        this.opposite = 'W';}
    for (let i = 0; i < this.size; i++) {
        for (let j = 0; j < this.size; j++) {
            this.check_8_direction(i, j);
        }
    }
    if (this.valid_move_avalable == false){
            if (this.gameover_count>0){
                console.log('gameover')
            }else{
            this.gameover_count+=1}
        }
        return this.table;
```

- เช็ครอบทิศของแต่ละช่อง

```
check_8_direction(x, y) {
    if (this.table[x][y] !== '.') {
        return;
    }
    let valid = false;
    //[[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [-1,
0]]
    for (let direction of this.check_direction) {
        let [dix, diy] = direction;
        if (this.check_if_that_posi_valid(x, y, dix, diy)) {
            valid = true;
        }
    }

    if (valid) {
        this.table[x][y] = 'O';
        this.valid_move_avalable = true;
    }
}
```

- เช็คทิศนั้นๆจนสุดกระดาน

```
check_if_that_posi_valid(current_x, current_y, direction_x, direction_y) {
    let checking_x = current_x + direction_x;
    let checking_y = current_y + direction_y;
    let opponent_detected = false;
    //ตั้ง condition ไม่ให้เลยกระดาน
    while (checking_x >= 0 && checking_x < this.size && checking_y >= 0 &&
checking_y < this.size) {
        //ข้ามถ้าหากรอบจุดที่เช็คอยู่เป็นช่องว่าง
        if (this.table[checking_x][checking_y] === '.' ||
this.table[checking_x][checking_y] === 'O') {
            return false;
        }
        //เก็บค่าถ้าเจอฝั่งตรงข้ามระหว่างทาง
        else if (this.table[checking_x][checking_y] === this.opposite) {
            opponent_detected = true;
        }
        //ถ้าเจอตัวเอง และ ระหว่างทางเจอฝั่งตรงข้าม ให้จุดนี้สามารถวางงได้
        else if (this.table[checking_x][checking_y] === this.current_player
&& opponent_detected) {
            this.valid_move_avalable = true;
            return true;
        } else {
            return false;
        }
        checking_x += direction_x;
        checking_y += direction_y;
    }

    return false;
}
```

- function ในการกลับด้านหมาก

- เช็ครอบทิศของจุดที่ผู้เล่นทำการวางหมาก

```
flip(fx,fy){
    //[[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [-1,
0]]
    for (let direction of this.check_direction) {
        let [dix, diy] = direction;
        this.flip_direction(fx, fy, dix, diy);
    }

    return this.table
    }
```

- กลับด้านหมากในทิศนั้นๆ(หากตรวจเจอหมากฝั่งตัวเอง)

```
flip_direction(current_x, current_y, direction_x, direction_y) {
    let checking_x = current_x + direction_x;
    let checking_y = current_y + direction_y;
    let positions_to_flip = [];
    //ตั้ง condition ไม่ให้เลยกระดาน
    while (checking_x >= 0 && checking_x < this.size && checking_y >= 0 &&
checking_y < this.size) {
        if (this.table[checking_x][checking_y] === '.' ||
this.table[checking_x][checking_y] === 'O' ) {
            break
        }
        //if meet opposite player store the position
        else if (this.table[checking_x][checking_y] === this.opposite) {
            positions_to_flip.push([checking_x, checking_y]);
        } else if (this.table[checking_x][checking_y] ===
this.current_player) {
            //if meet current player use the stored position to flip
            for (let pos of positions_to_flip) {
                //change opposite to current player
                this.table[pos[0]][pos[1]] = this.current_player;
            }
            return ;
        }
        checking_x += direction_x;
        checking_y += direction_y;
    }
}
```