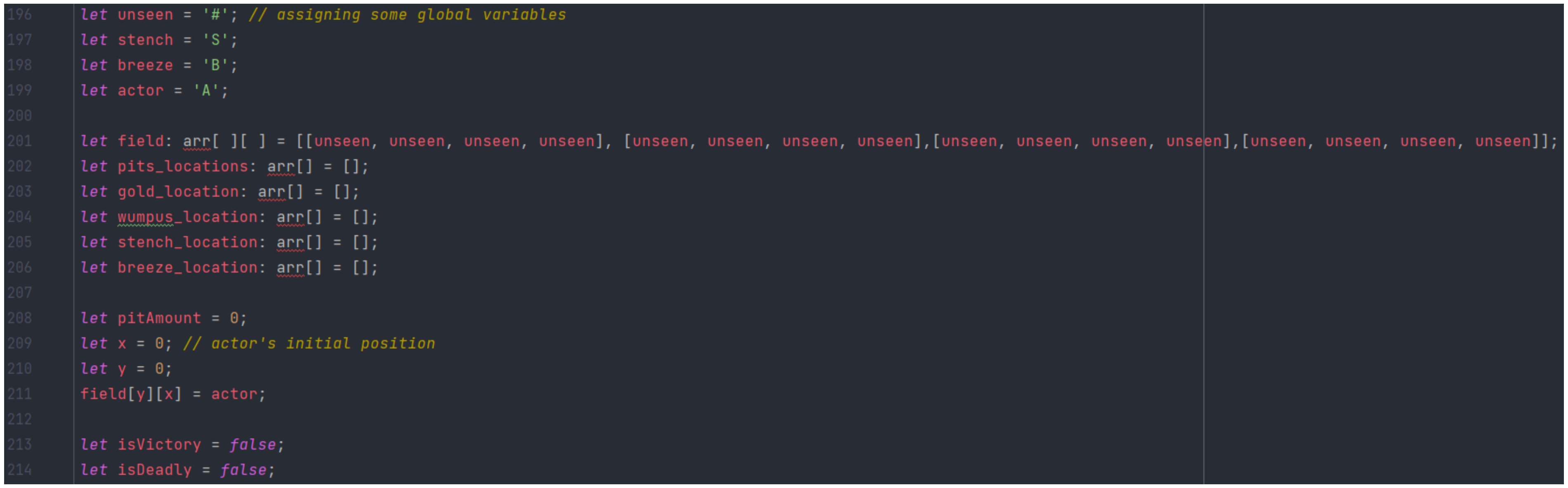
WUMPUS CONSOLE GAME

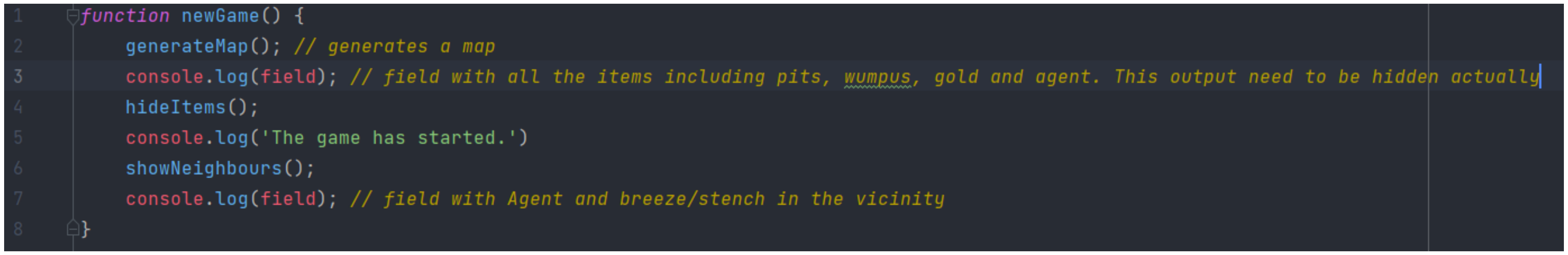
First, I need to apologise that I haven't completed the task properly. Initially, I tried to implement the whole game functionality without OOP. This part went successfully. But, the implementation of this code according to UML Class Diagram was impossible for me so far. However, let's have a look at the game:

The code starts with variable assignment (the meaning of each will be described later):



```
let unseen = '#'; // assigning some global variables
let stench = 'S';
let breeze = 'B';
let actor = 'A';

let field: arr[ ][ ] = [[unseen, unseen, unseen, unseen], [unseen, unseen, unseen, unseen],[unseen, unseen, unseen, unseen],[unseen, unseen, unseen, unseen]];
let pits_locations: arr[] = [];
let gold_location: arr[] = [];
let wumpus_location: arr[] = [];
let stench_location: arr[] = [];
let breeze_location: arr[] = [];

let pitAmount = 0;
let x = 0; // actor's initial position
let y = 0;
field[y][x] = actor;

let isVictory = false;
let isDeadly = false;
```

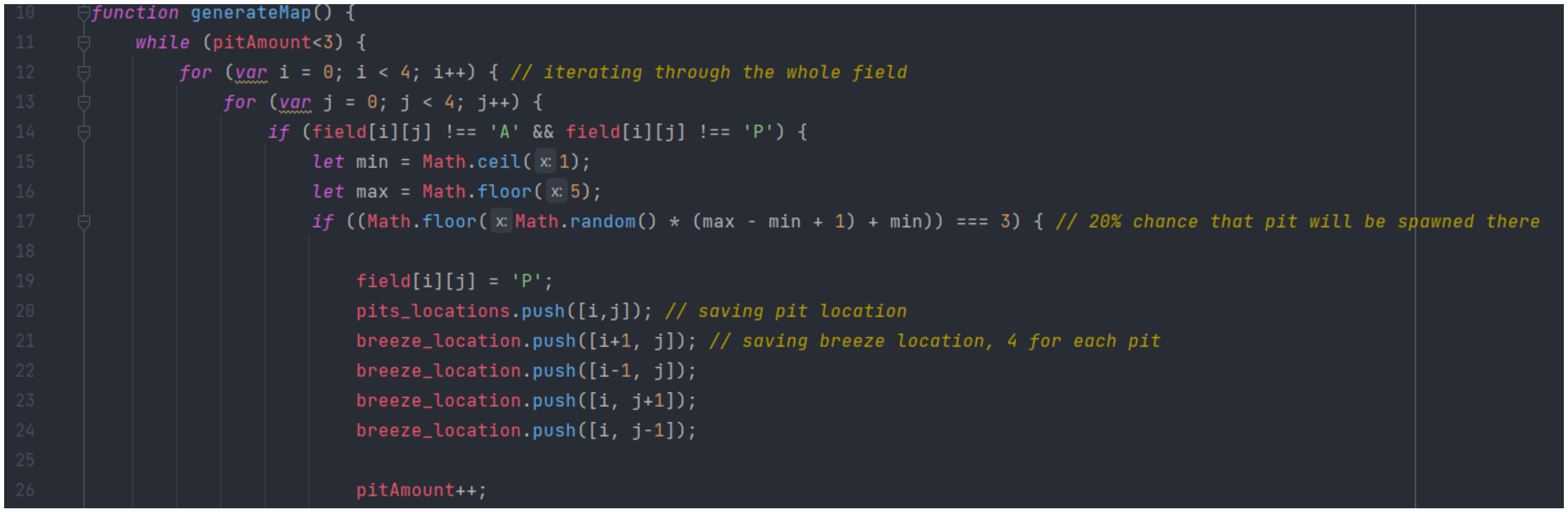
After calling newGame() function, we start the game. Function fulfils 3 steps: generates the map, hides items supposed to be hidden from player, and shows breeze / stench in neighbour cells.



```
function newGame() {
    generateMap(); // generates a map
    console.log(field); // field with all the items including pits, wumpus, gold and agent. This output need to be hidden actually
    hideItems();
    console.log('The game has started.')
    showNeighbours();
    console.log(field); // field with Agent and breeze/stench in the vicinity
}
```

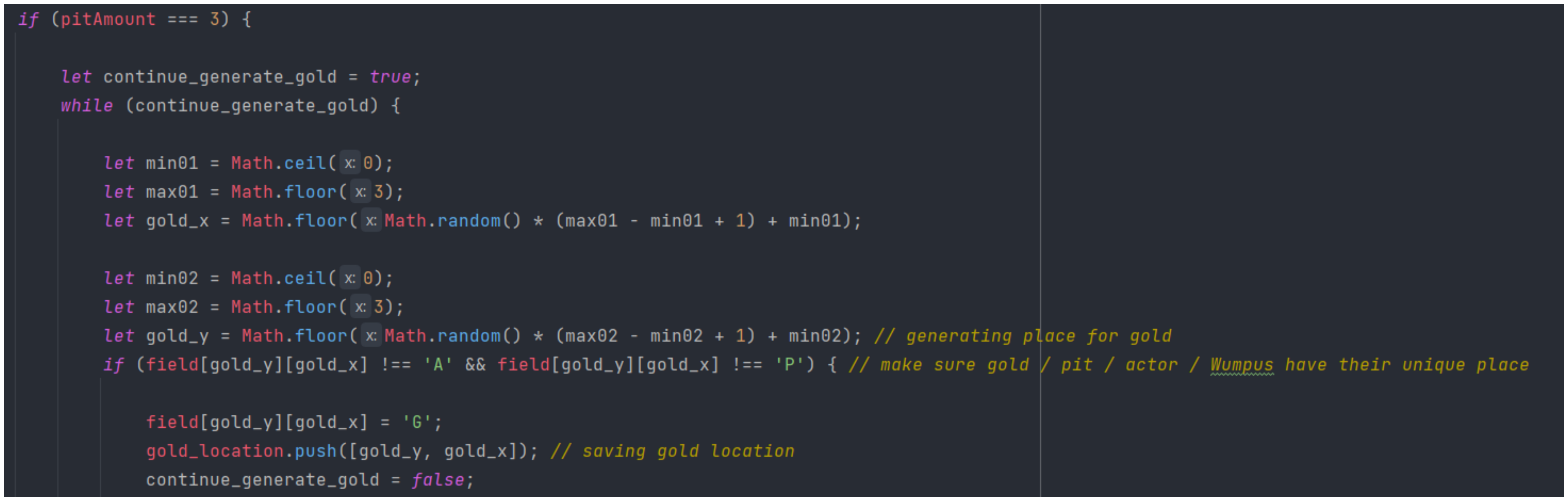
GenerateMap() function:

First snippet of generateMap() function generates 3 random located pits:



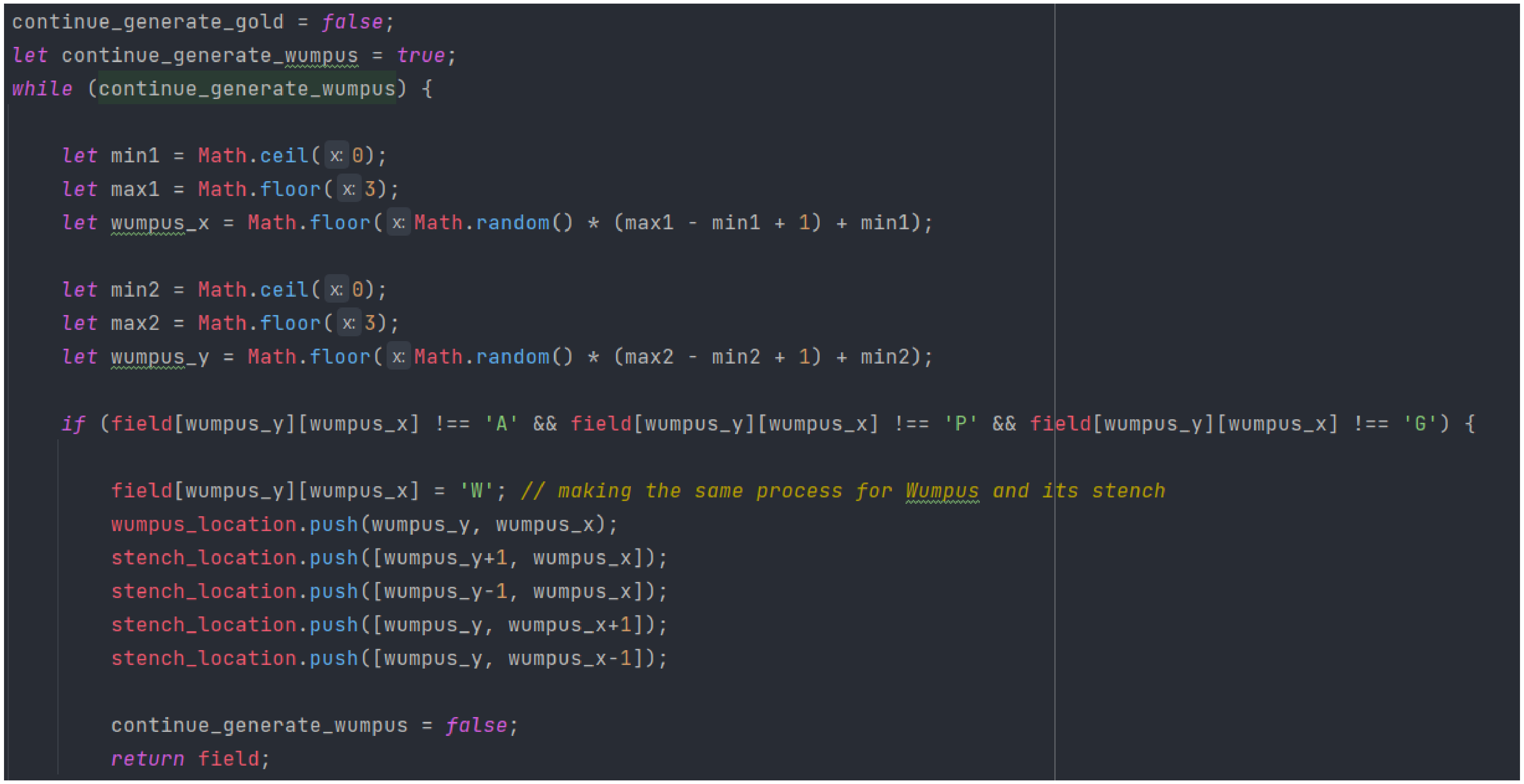
```
function generateMap() {
    while (pitAmount<3) {
        for (var i = 0; i < 4; i++) { // iterating through the whole field
            for (var j = 0; j < 4; j++) {
                if (field[i][j] !== 'A' && field[i][j] !== 'P') {
                    let min = Math.ceil( x: 1);
                    let max = Math.floor( x: 5);
                    if ((Math.floor( x: Math.random() * (max - min + 1) + min)) === 3) { // 20% chance that pit will be spawned there

                        field[i][j] = 'P';
                        pits_locations.push([i,j]); // saving pit location
                        breeze_location.push([i+1, j]); // saving breeze location, 4 for each pit
                        breeze_location.push([i-1, j]);
                        breeze_location.push([i, j+1]);
                        breeze_location.push([i, j-1]);

                        pitAmount++;
```

Snippet that generates gold:



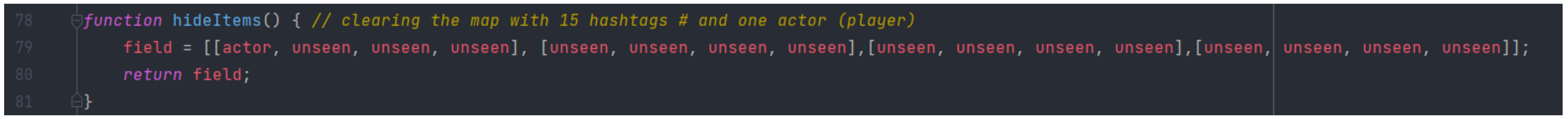
```
if (pitAmount === 3) {

    let continue_generate_gold = true;
    while (continue_generate_gold) {

        let min01 = Math.ceil( x: 0);
        let max01 = Math.floor( x: 3);
        let gold_x = Math.floor( x: Math.random() * (max01 - min01 + 1) + min01);

        let min02 = Math.ceil( x: 0);
        let max02 = Math.floor( x: 3);
        let gold_y = Math.floor( x: Math.random() * (max02 - min02 + 1) + min02); // generating place for gold
        if (field[gold_y][gold_x] !== 'A' && field[gold_y][gold_x] !== 'P') { // make sure gold / pit / actor / Wumpus have their unique place

            field[gold_y][gold_x] = 'G';
            gold_location.push([gold_y, gold_x]); // saving gold location
            continue_generate_gold = false;
```

Repeating the same process for Wumpus location generation:



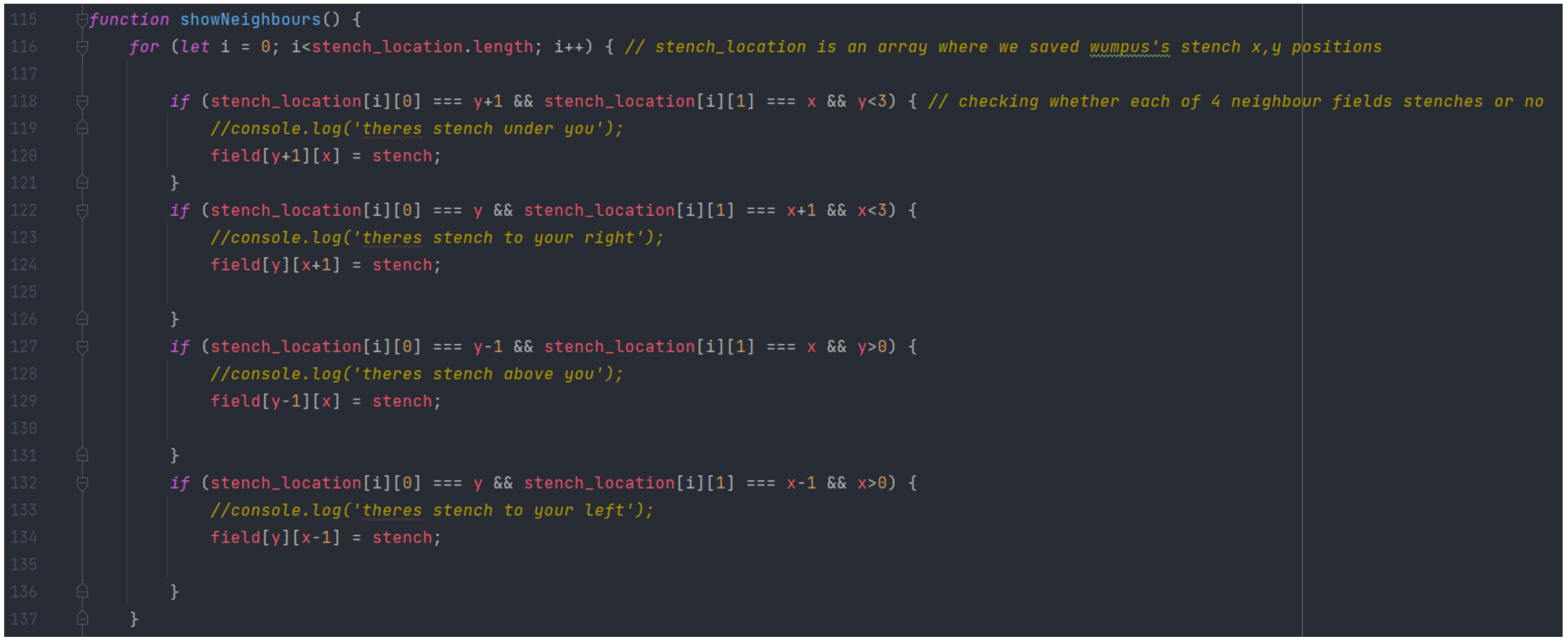
```
continue_generate_gold = false;
let continue_generate_wumpus = true;
while (continue_generate_wumpus) {

    let min1 = Math.ceil( x: 0);
    let max1 = Math.floor( x: 3);
    let wumpus_x = Math.floor( x: Math.random() * (max1 - min1 + 1) + min1);

    let min2 = Math.ceil( x: 0);
    let max2 = Math.floor( x: 3);
    let wumpus_y = Math.floor( x: Math.random() * (max2 - min2 + 1) + min2);

    if (field[wumpus_y][wumpus_x] !== 'A' && field[wumpus_y][wumpus_x] !== 'P' && field[wumpus_y][wumpus_x] !== 'G') {

        field[wumpus_y][wumpus_x] = 'W'; // making the same process for Wumpus and its stench
        wumpus_location.push(wumpus_y, wumpus_x);
        stench_location.push([wumpus_y+1, wumpus_x]);
        stench_location.push([wumpus_y-1, wumpus_x]);
        stench_location.push([wumpus_y, wumpus_x+1]);
        stench_location.push([wumpus_y, wumpus_x-1]);

        continue_generate_wumpus = false;
        return field;
```

After the map has been generated, we need to hide pits, gold and Wumpus from the player:



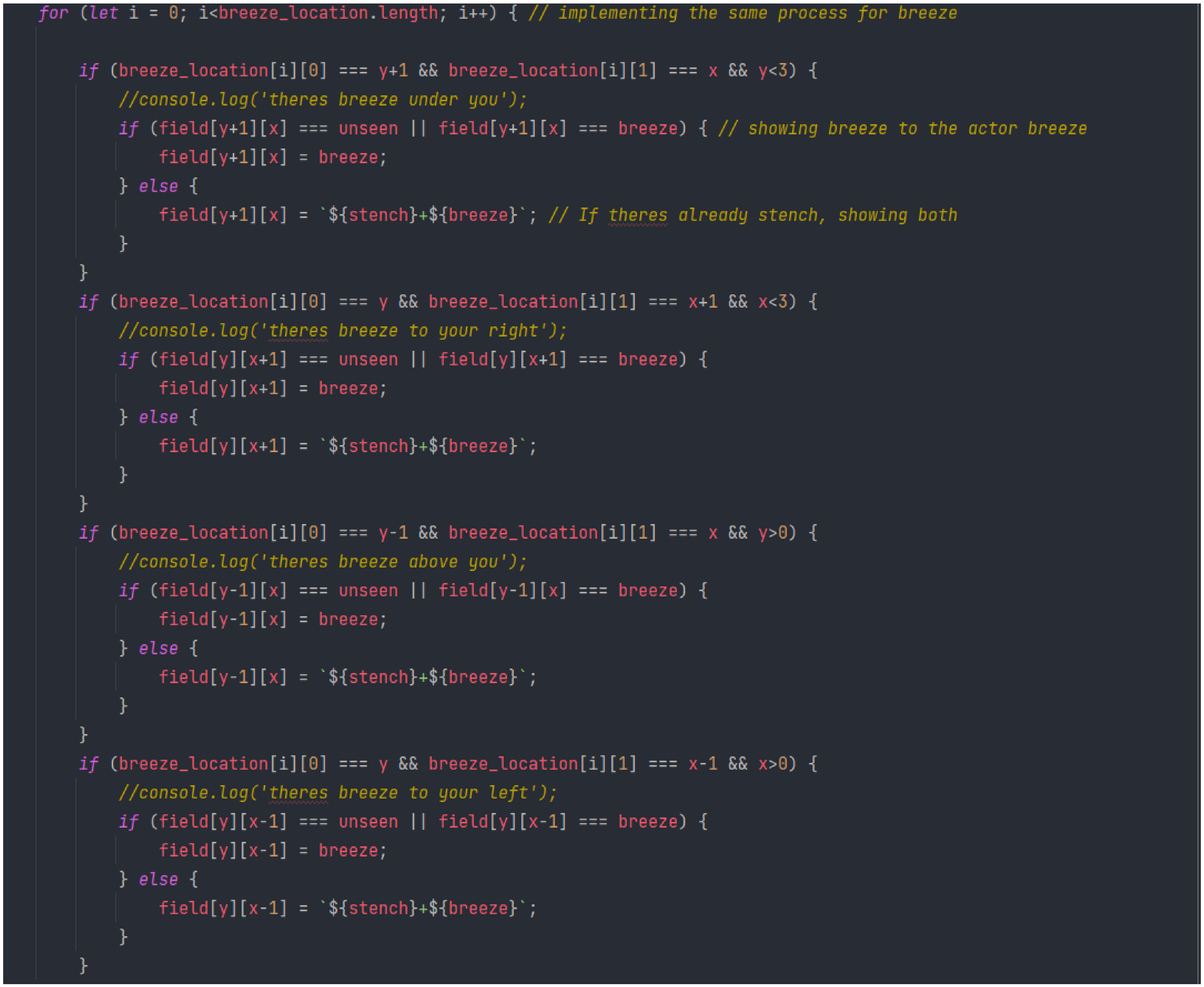
```
function hideItems() { // clearing the map with 15 hashtags # and one actor (player)
    field = [[actor, unseen, unseen, unseen], [unseen, unseen, unseen, unseen],[unseen, unseen, unseen, unseen],[unseen, unseen, unseen, unseen]];
    return field;
}
```

Checking whether the actor feels stench or no. If yes, program displays it around the actor:



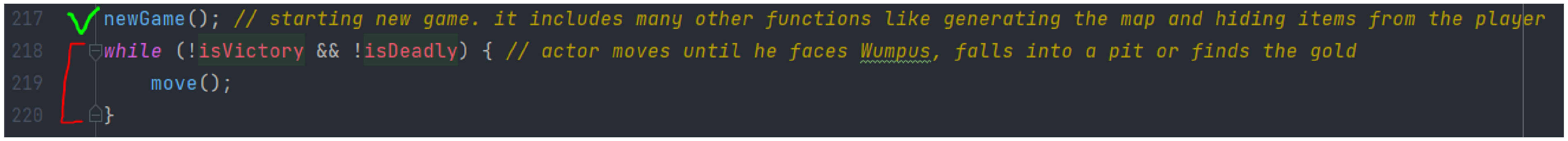
```
function showNeighbours() {
    for (let i = 0; i<stench_location.length; i++) { // stench_location is an array where we saved wumpus's stench x,y positions

        if (stench_location[i][0] === y+1 && stench_location[i][1] === x && y<3) { // checking whether each of 4 neighbour fields stenches or no
            //console.log('theres stench under you');
            field[y+1][x] = stench;
        }
        if (stench_location[i][0] === y && stench_location[i][1] === x+1 && x<3) {
            //console.log('theres stench to your right');
            field[y][x+1] = stench;

        }
        if (stench_location[i][0] === y-1 && stench_location[i][1] === x && y>0) {
            //console.log('theres stench above you');
            field[y-1][x] = stench;

        }
        if (stench_location[i][0] === y && stench_location[i][1] === x-1 && x>0) {
            //console.log('theres stench to your left');
            field[y][x-1] = stench;

        }
    }
```

Checking whether the actor feels breeze or no. If yes, program display it around the actor. If there's breeze and stench concurrently, program displays both breeze and stench:



```
for (let i = 0; i<breeze_location.length; i++) { // implementing the same process for breeze

    if (breeze_location[i][0] === y+1 && breeze_location[i][1] === x && y<3) {
        //console.log('theres breeze under you');
        if (field[y+1][x] === unseen || field[y+1][x] === breeze) { // showing breeze to the actor breeze
            field[y+1][x] = breeze;
        } else {
            field[y+1][x] = `${stench}+${breeze}`; // If theres already stench, showing both
        }
    }
    if (breeze_location[i][0] === y && breeze_location[i][1] === x+1 && x<3) {
        //console.log('theres breeze to your right');
        if (field[y][x+1] === unseen || field[y][x+1] === breeze) {
            field[y][x+1] = breeze;
        } else {
            field[y][x+1] = `${stench}+${breeze}`;
        }
    }
    if (breeze_location[i][0] === y-1 && breeze_location[i][1] === x && y>0) {
        //console.log('theres breeze above you');
        if (field[y-1][x] === unseen || field[y-1][x] === breeze) {
            field[y-1][x] = breeze;
        } else {
            field[y-1][x] = `${stench}+${breeze}`;
        }
    }
    if (breeze_location[i][0] === y && breeze_location[i][1] === x-1 && x>0) {
        //console.log('theres breeze to your left');
        if (field[y][x-1] === unseen || field[y][x-1] === breeze) {
            field[y][x-1] = breeze;
        } else {
            field[y][x-1] = `${stench}+${breeze}`;
        }
    }
```

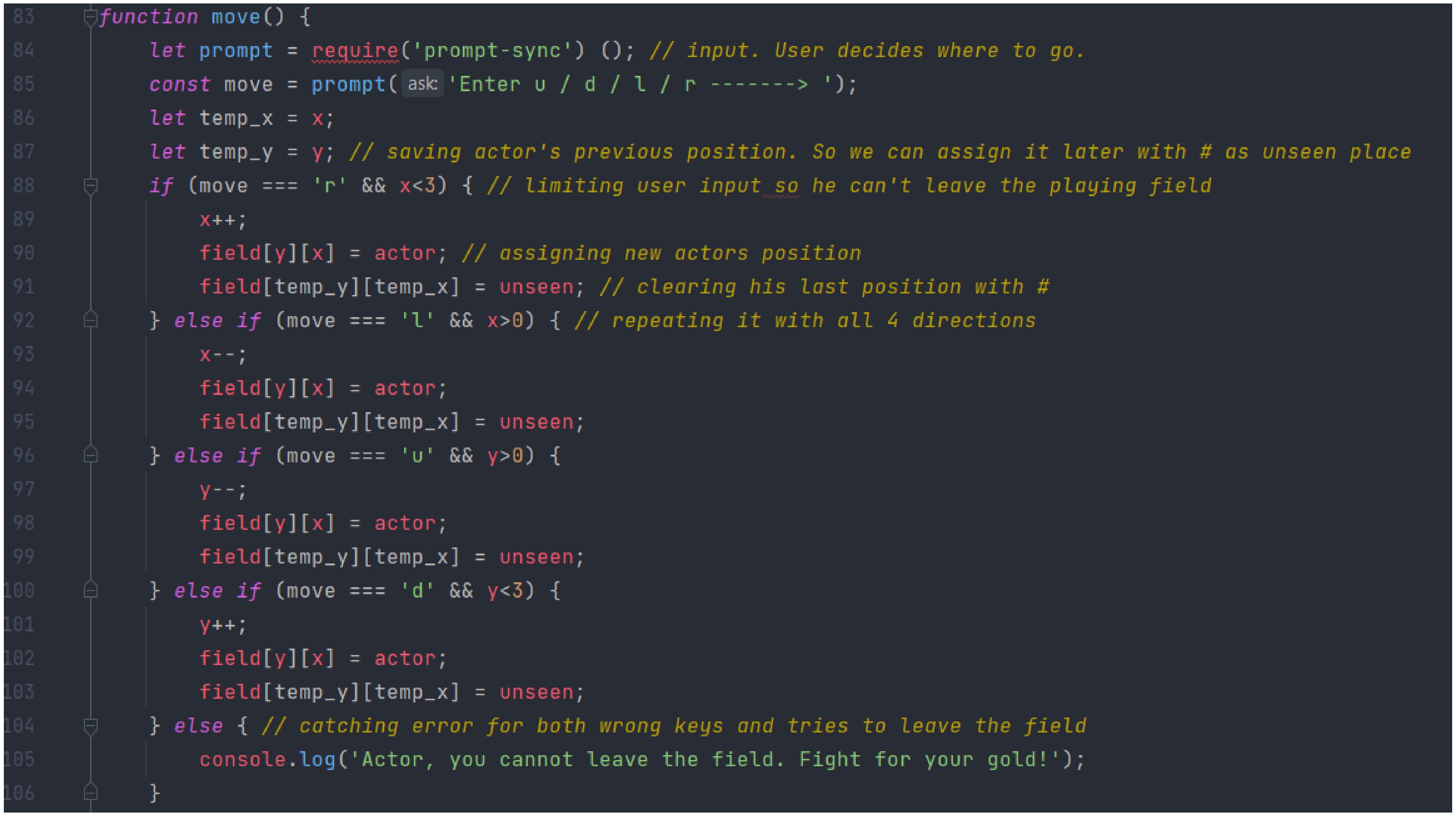
Now, newGame() function has already ended successfully. While loop calls move() function until the actor haven't faced gold, pit or Wumpus:



```
newGame(); // starting new game. it includes many other functions like generating the map and hiding items from the player
while (!isVictory && !isDeadly) { // actor moves until he faces Wumpus, falls into a pit or finds the gold
    move();
}
```

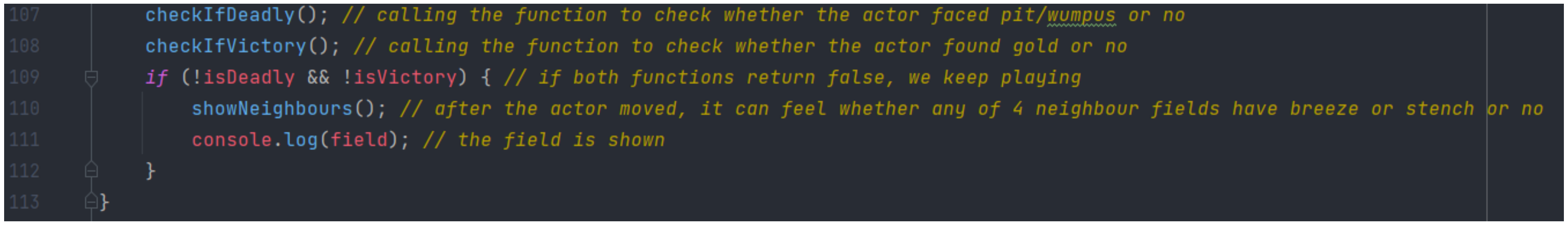
Move() function

The snippet of move() function. 84-87 lines implements user input and saves the previous actor's location. The latter part of the code determines the direction where the actor moves. Of course, the actor can't escape the playing 4x4 field:



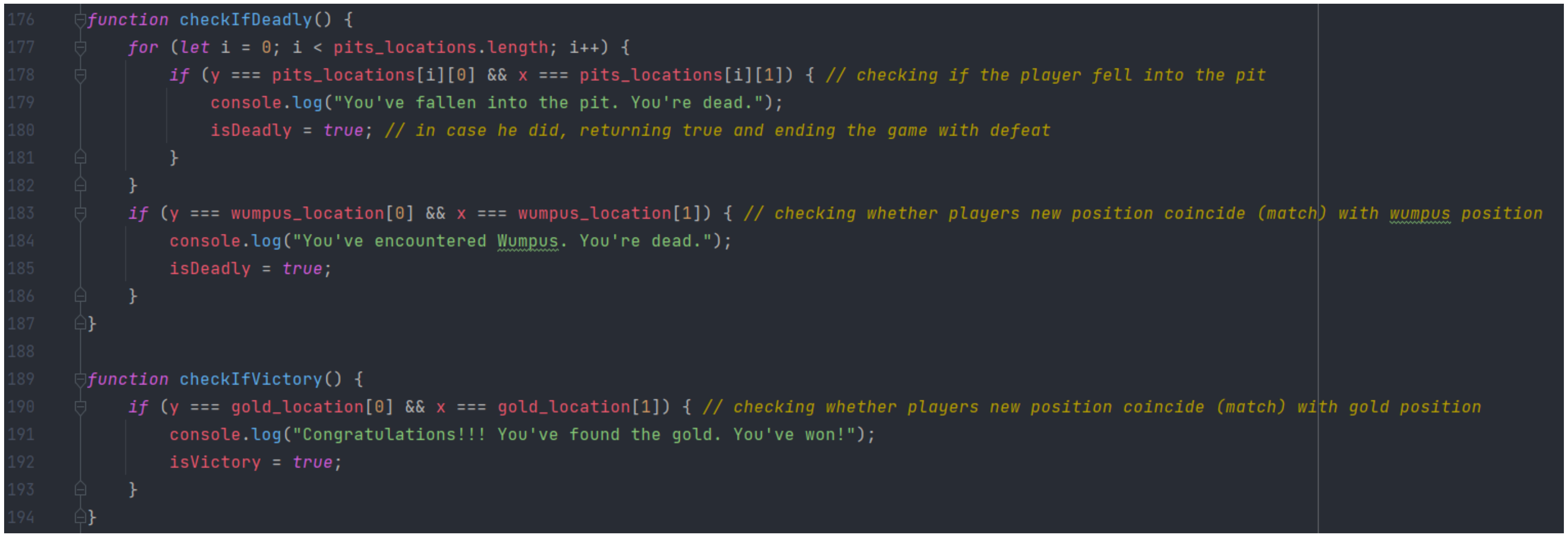
```
83   function move() {
84       let prompt = require('prompt-sync') (); // input. User decides where to go.
85       const move = prompt( ask: 'Enter u / d / l / r -------> ');
86       let temp_x = x;
87       let temp_y = y; // saving actor's previous position. So we can assign it later with # as unseen place
88       if (move === 'r' && x<3) { // limiting user input so he can't leave the playing field
89           x++;
90           field[y][x] = actor; // assigning new actors position
91           field[temp_y][temp_x] = unseen; // clearing his last position with #
92       } else if (move === 'l' && x>0) { // repeating it with all 4 directions
93           x--;
94           field[y][x] = actor;
95           field[temp_y][temp_x] = unseen;
96       } else if (move === 'u' && y>0) {
97           y--;
98           field[y][x] = actor;
99           field[temp_y][temp_x] = unseen;
100      } else if (move === 'd' && y<3) {
101          y++;
102          field[y][x] = actor;
103          field[temp_y][temp_x] = unseen;
104      } else { // catching error for both wrong keys and tries to leave the field
105          console.log('Actor, you cannot leave the field. Fight for your gold!');
106      }
```

The rest of the move() function:



```
    checkIfDeadly(); // calling the function to check whether the actor faced pit/wumpus or no
    checkIfVictory(); // calling the function to check whether the actor found gold or no
    if (!isDeadly && !isVictory) { // if both functions return false, we keep playing
        showNeighbours(); // after the actor moved, it can feel whether any of 4 neighbour fields have breeze or stench or no
        console.log(field); // the field is shown
    }
}
```

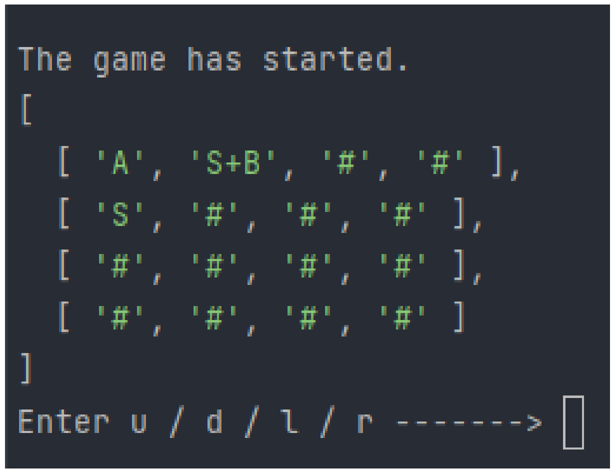
2 functions that determine whether the game continues or no. If they both return false, while loop continues and the actor will continue exploring the playing field:



```
function checkIfDeadly() {
    for (let i = 0; i < pits_locations.length; i++) {
        if (y === pits_locations[i][0] && x === pits_locations[i][1]) { // checking if the player fell into the pit
            console.log("You've fallen into the pit. You're dead.");
            isDeadly = true; // in case he did, returning true and ending the game with defeat
        }
    }
    if (y === wumpus_location[0] && x === wumpus_location[1]) { // checking whether players new position coincide (match) with wumpus position
        console.log("You've encountered Wumpus. You're dead.");
        isDeadly = true;
    }
}

function checkIfVictory() {
    if (y === gold_location[0] && x === gold_location[1]) { // checking whether players new position coincide (match) with gold position
        console.log("Congratulations!!! You've found the gold. You've won!");
        isVictory = true;
    }
}
```

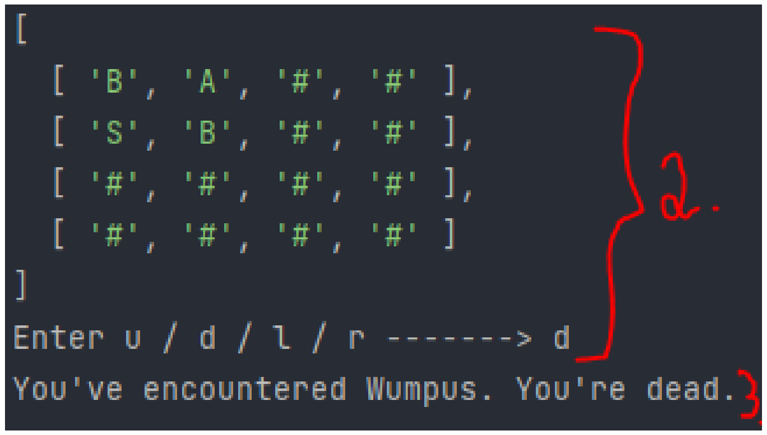
Input and Output

How the output & input actually look like:



```
The game has started.
[
  [ 'A', 'S+B', '#', '#' ],
  [ 'S', '#', '#', '#' ],
  [ '#', '#', '#', '#' ],
  [ '#', '#', '#', '#' ]
]
Enter u / d / l / r ------->
```

Second and third iteration. Wumpus location was: field[1][1]. In the second iteration I decided to move downwards, and faced the wumpus. Third iteration informs me that I lost the game:



```
[
  [ 'B', 'A', '#', '#' ],
  [ 'S', 'B', '#', '#' ],
  [ '#', '#', '#', '#' ],
  [ '#', '#', '#', '#' ]
]
Enter u / d / l / r -------> d
You've encountered Wumpus. You're dead.
```