Comp1110/Comp6710 Presentation

Group 12g

Team

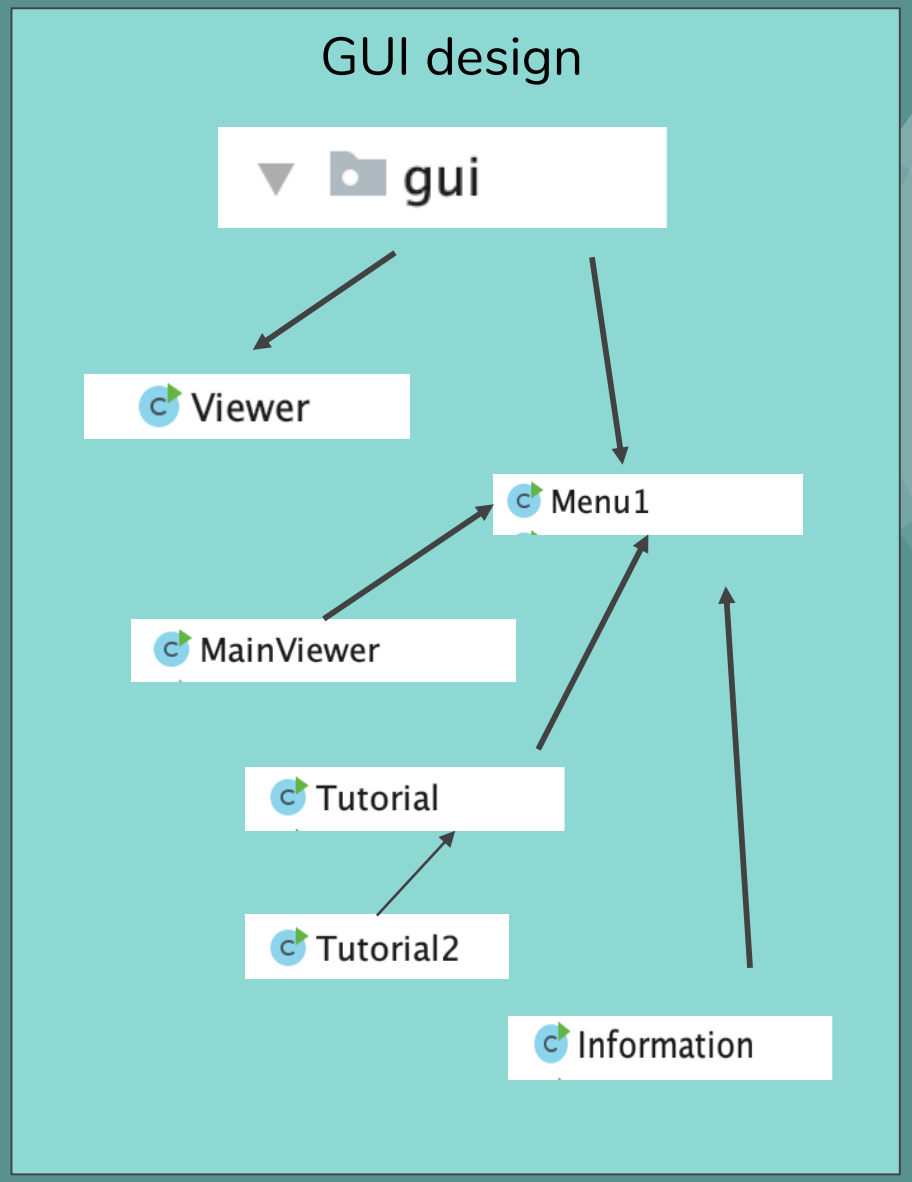
Xinyao Wang u6609958
Xinyi Zhang u6976740
Ruiqiao Jiang u6818746

Structure

Rules of metro game

code

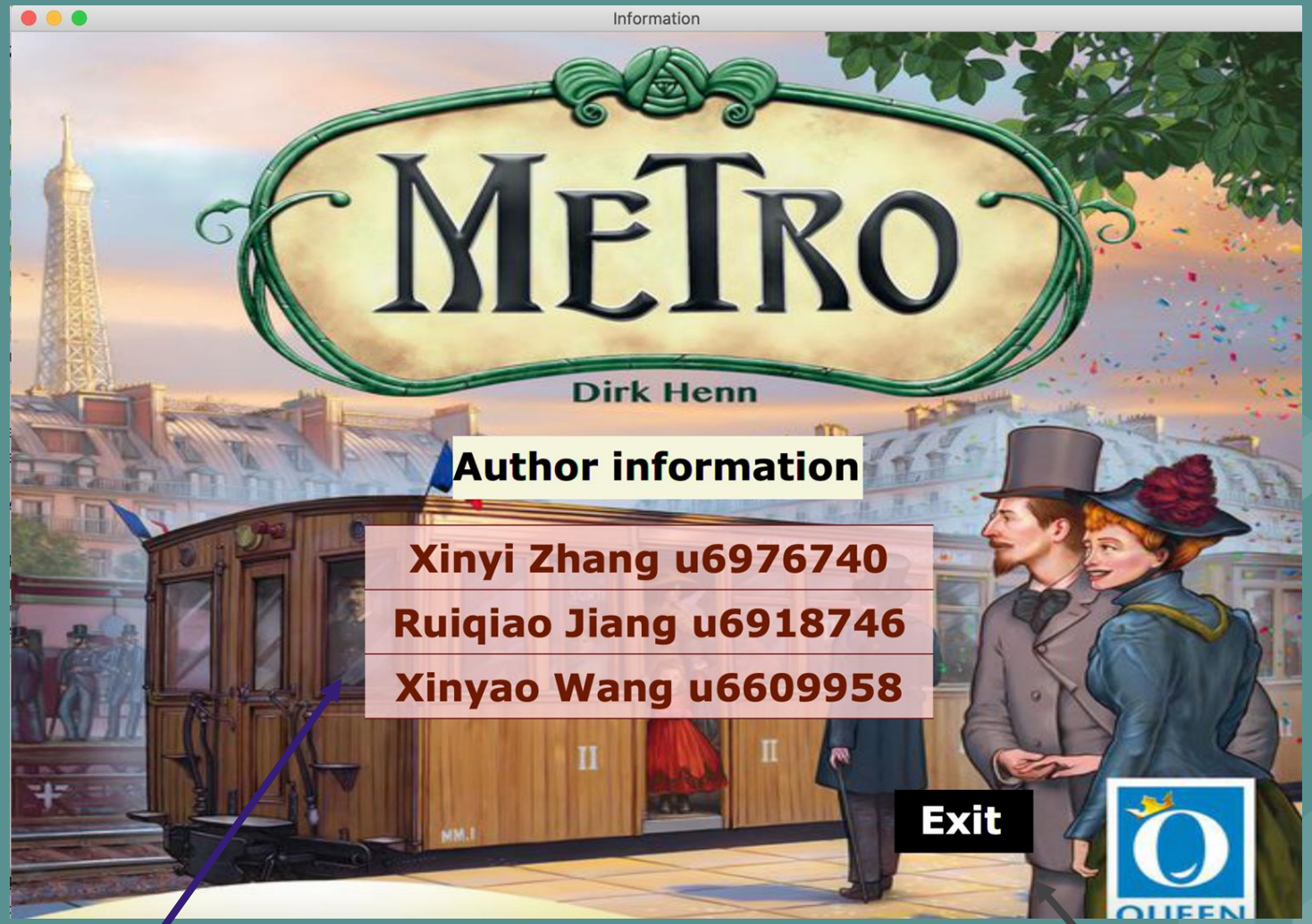
Diagram


GUI design
gui
Viewer
Menu1
MainViewer
Tutorial
Tutorial2
Information

Game Menu

Animation: the colour will change when the move the mouse and press the button

Information Page



Indicated the author information

Press to back the main viewer

Tutorial

brief tutorial of the game source from the Metro Queen Games

Metro

How To Play The Game

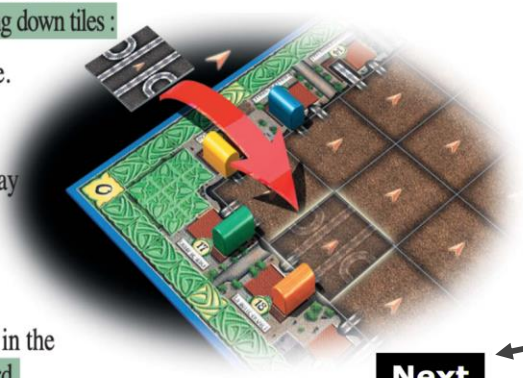
- The youngest player plays first. Each of the other players then take turns clockwise.

Placing Track tiles on the board

- Each player takes a turn placing a Track tile on the board. Whenever a player cannot, or does not want to, place the tile that he is holding, he picks up a new tile - as long as there are tiles available in the pool - which he then must place on the board.

These 4 conditions must be followed when laying down tiles :

1. Each tile must be placed on an empty square. That square must border on (be adjacent to) a tile which has already been played, or be placed along the edge of the board. A tile may not be placed next to one of the central station squares unless it completes or continues an existing track.



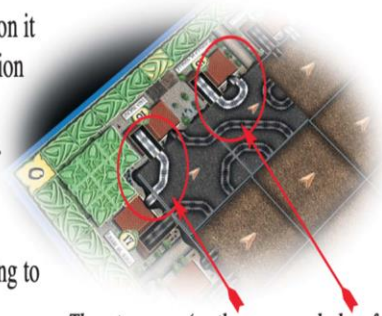
2. The red arrow on the tile must always face in the same direction as the red arrow on the board.

Next

To the next page

Metro

3. A tile may not be placed so that the tracks printed on it begin and end in the same station. The only exception would be if that tile could not be placed elsewhere (this rarely happens except at the end of the game).



These two cases (on the corner and edge of the game board) are normally not allowed.

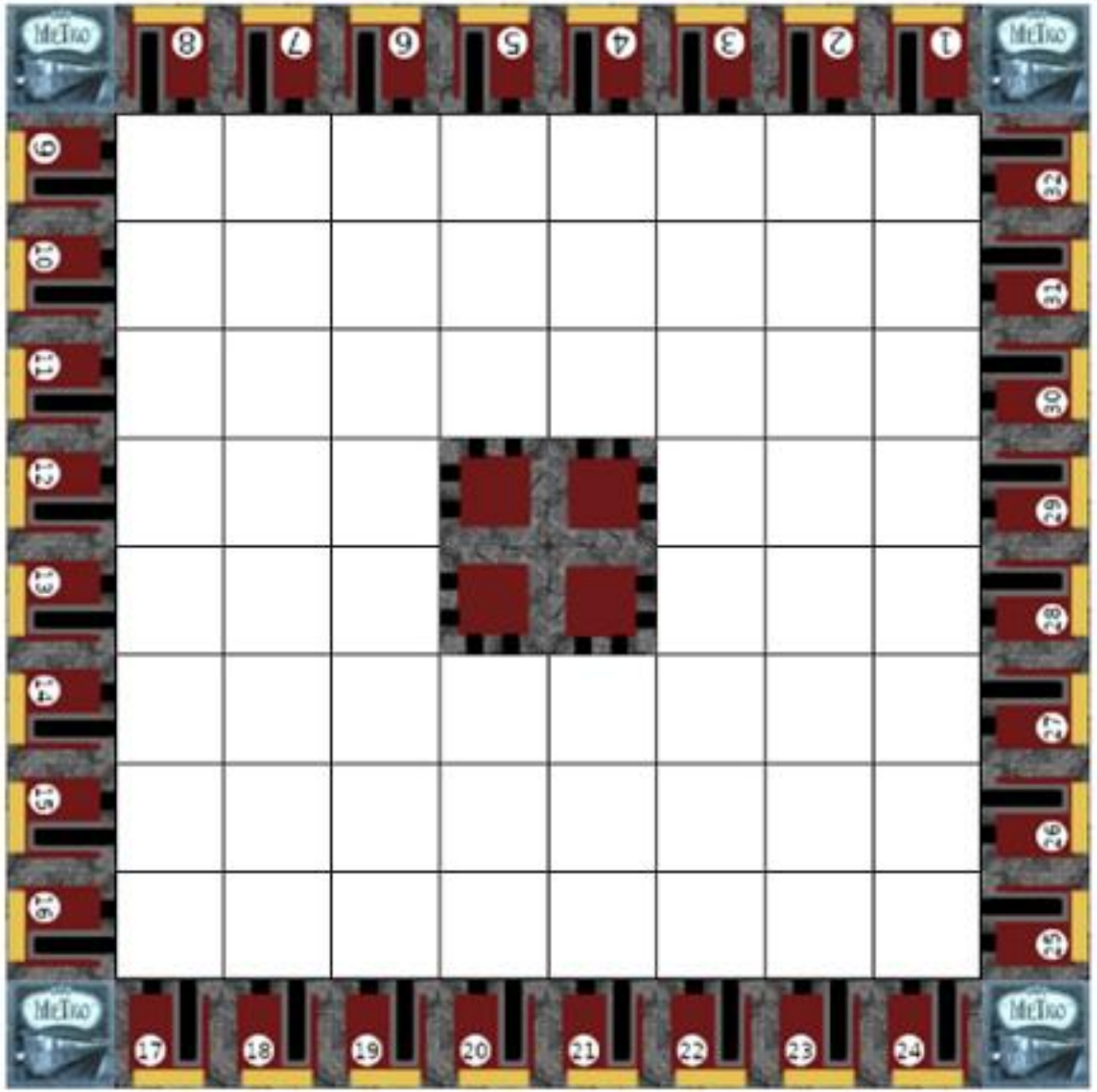
4. A tile may be placed next to an opponent's Metro subway departure point or next to one of his Metro lines, but the tile must be placed according to the previous placement rules.

- After laying down a tile, the player picks up a new tile, as long as one is available, unless he is already holding a tile. A player may only have one tile in his hand at the end of his turn.
- Players whose Metro stations are all finished by a completed network of lines must continue to place their tiles elsewhere on the game board.

Back

Back to Main viewer

Viewe r



One tile randomly selected by the computer, When we clicked on the tile picture ,the size of the tile will become the same as the gird on the board, and it can be dragged onto the board.

Number of player:

Input the number of players here(1-6)

pc move:

Input the number of tiles you want the computer to automatically generate on the board after you place a tile correctly

player1: player2:

player3: player4:

player5: player6:

Display the score of each player, automatically update after dragging the tile to the correct position.

Exit

Return to main interface

Task 2

determine whether a piece placement is well-formed

```
 *
 * @param piecePlacement A String representing the piece to be placed
 * @return True if this string is well-formed
 *  @author Ruiqiao Jiang
 */
public static boolean isPiecePlacementWellFormed(String piecePlacement) {



    // FIXME Task 2: determine whether a piece placement is well-formed
    if (!Pattern.matches( regex: "[a-d]{4}[0-7]{2}", piecePlacement))
        return false;
    return true;
}
```

Task 3

```
Map<String, Integer> counts = getDeckMap();
List<Integer> positions = new ArrayList<>();
for (int i = 0; i < placement.length(); i += 6) {
    if (placement.substring(i).length() < 6) {
        return false;
    }
    if (!isPiecePlacementWellFormed(placement.substring(i, i + 6))) {
        return false;
    }
    int posistion = Integer.parseInt(placement.substring(i + 4, i + 6));
    if (!positions.contains(posistion)) {
        positions.add(posistion);
    } else {
        return false;
    }
    String key = placement.substring(i, i + 4);
    if ((counts.put(key, counts.get(key) - 1)) == 0) {
        return false;
```

determine whether a placement sequence is well-formed

Task 5

draw a random tile from deck

```
    Map<String, Integer> counts = getDeckMap();
    for (int i = 0; i < placementSequence.length(); i += 6) {
        String key = placementSequence.substring(i, i + 4);
        counts.put(key, counts.get(key) - 1);
    }
    for (int i = 0; i < totalHands.length(); i += 4) {
        String key = totalHands.substring(i, i + 4);
        counts.put(key, counts.get(key) - 1);
    }
    List<String> randomList = new ArrayList<>();
    for (String key : counts.keySet()) {
        if (counts.get(key) > 0) {
            randomList.add(key);
        }
    }
    return randomList.get(new Random().nextInt(randomList.size()));
}
```

Task6-9

Task 6: determine whether a placement sequence is valid
Task 7: determine the current score for the game
Task 9: generate a valid move

```
// FIXME Task 6: determine whether a placement sequence is valid
if (!isPlacementSequenceWellFormed(placementSequence)) {
    return false;
}

Set alreadyPositions = new HashSet();
for (int i = 0; i < placementSequence.length(); i += 6) {
    String position = placementSequence.substring(i + 4, i + 6);
    String direction = placementSequence.substring(i, i + 4);
    // Condition Duplication
    if (alreadyPositions.contains(position)) {
        return false;
    } else {
        int rowNumber = Integer.parseInt(placementSequence.substring(i + 4, i +
        int colNumber = Integer.parseInt(placementSequence.substring(i + 5, i +
        String oneDirection = placementSequence.substring(i, i + 1);
        String twoDirection = placementSequence.substring(i + 1, i + 2);
        String threeDirection = placementSequence.substring(i + 2, i + 3);
        String fourDirection = placementSequence.substring(i + 3, i + 4);

        // judge if it is the centre station
        if (Arrays.asList(new String[]{"33", "34", "43", "44"}).contains(positio
```

Task6

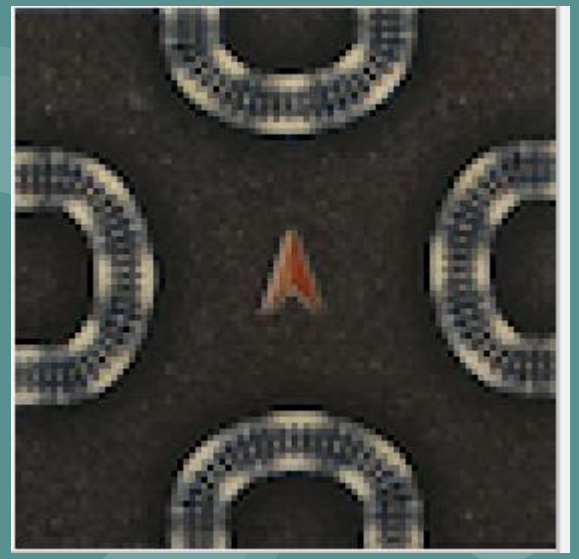
abcd16acbd12

Variables
abcd-direction
16-position
1-row number
6-col number
a-oneDirection
b-twoDirection
c-threeDirection
d-fourDirection

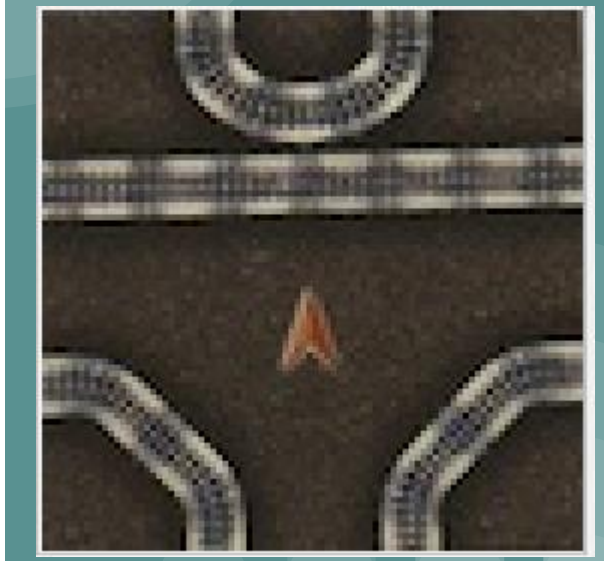
```java
// judge if it is the centre station
if (Arrays.asList(new String[]{"33", "34", "43", "44"}).contains(position)) {
    return false;
} else if (rowNumber == 0 || rowNumber == 7 || colNumber == 0 || colNumber == 7) {

    if (direction.equals("dddd") && alreadyPositions.size() == 0) {
        alreadyPositions.add(position);
        continue;
    }
    if (rowNumber == 0 && colNumber != 0 && colNumber != 7) {
        if (oneDirection.equals("d")) {
            return false;
        }
    } else if (rowNumber == 7 && colNumber != 0 && colNumber != 7) {
        if (threeDirection.equals("d")) {
            return false;
        }
    } else if (colNumber == 0 && rowNumber != 0 && rowNumber != 7) {
        if (fourDirection.equals("d")) {
            return false;
        }
    } else if (colNumber == 7 && rowNumber != 0 && rowNumber != 7) {
        if (twoDirection.equals("d")) {
            return false;
```

- dacc.jpg
- dada.jpg
- dbba.jpg
- dbcd.jpg
- ddbc.jpg
- dddd.jpg

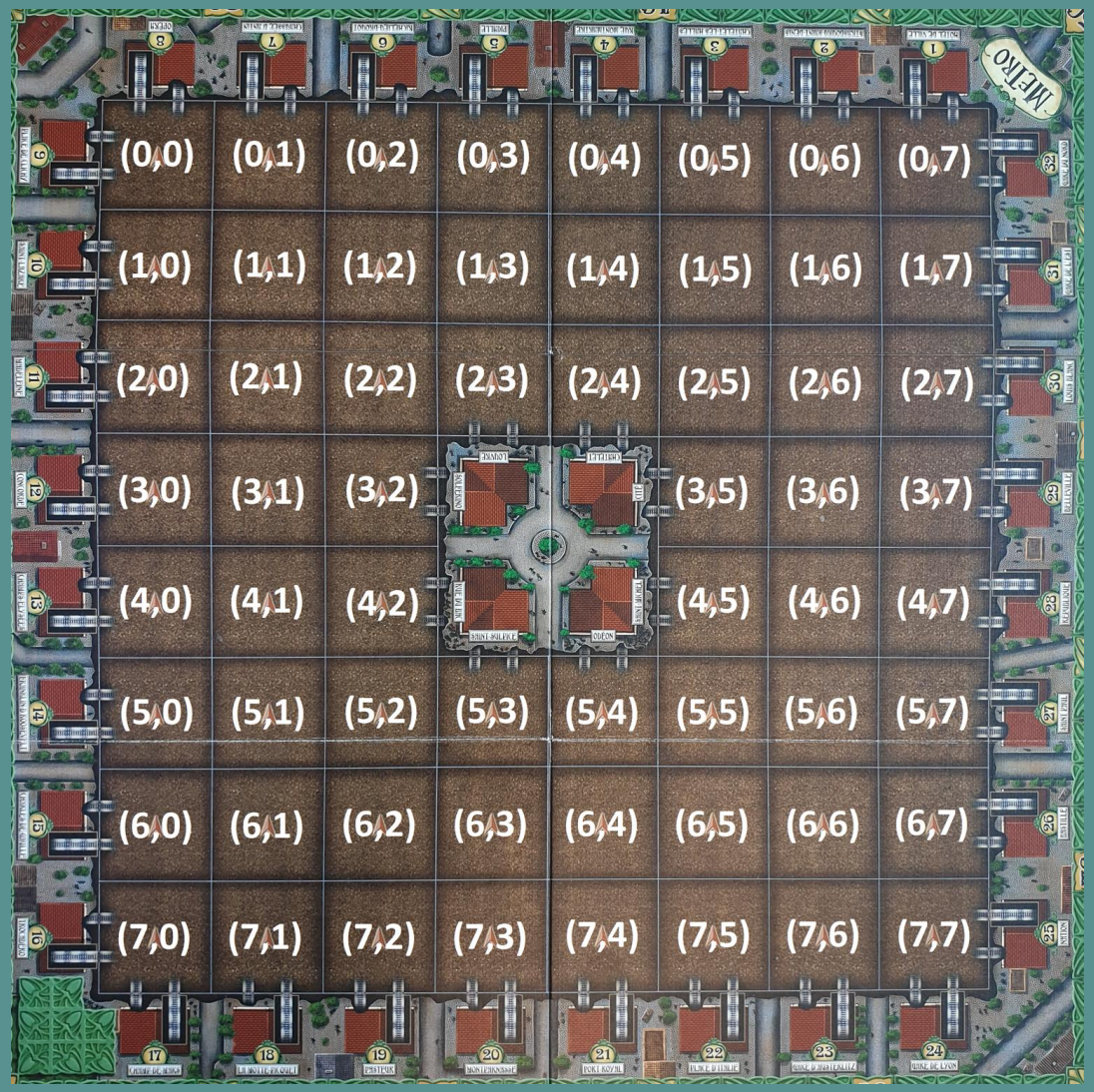


dddd



dacc

```
        }
    }
    if (rowNumber == 7 && colNumber == 7) {
        if (twoDirection.equals("d") && threeDirection.equals("d")) {
            return false;
        }
    }
    if (rowNumber == 0 && colNumber == 0) {
        if (oneDirection.equals("d") && fourDirection.equals("d")) {
            return false;
        }
    }
    if (rowNumber == 7 && colNumber == 0) {
        if (threeDirection.equals("d") && fourDirection.equals("d")) {
            return false;
        }
    }
    if (rowNumber == 0 && colNumber == 7) {
        if (oneDirection.equals("d") && twoDirection.equals("d")) {
            return false;
        }
    }
```


(0,0) (0,1) (0,2) (0,3) (0,4) (0,5) (0,6) (0,7)
(1,0) (1,1) (1,2) (1,3) (1,4) (1,5) (1,6) (1,7)
(2,0) (2,1) (2,2) (2,3) (2,4) (2,5) (2,6) (2,7)
(3,0) (3,1) (3,2) (3,5) (3,6) (3,7)
(4,0) (4,1) (4,2) (4,5) (4,6) (4,7)
(5,0) (5,1) (5,2) (5,3) (5,4) (5,5) (5,6) (5,7)
(6,0) (6,1) (6,2) (6,3) (6,4) (6,5) (6,6) (6,7)
(7,0) (7,1) (7,2) (7,3) (7,4) (7,5) (7,6) (7,7)

```
} else {
    boolean flag = false;
    if ((rowNumber - 1) != -1 && alreadyPositions.contains((rowNumber - 1) + "" + colNumber)) {
        flag = true;
    } else if ((rowNumber + 1) != 8 && alreadyPositions.contains((rowNumber + 1) + "" + colNumber)) {
        flag = true;
    } else if ((colNumber - 1) != -1 && alreadyPositions.contains(rowNumber + "" + (colNumber - 1))) {
        flag = true;
    } else if ((colNumber + 1) != 8 && alreadyPositions.contains(rowNumber + "" + (colNumber + 1))) {
        flag = true;
    }
    if (!flag) {
        return false;
    }
    alreadyPositions.add(position);
}
```

Thanks