COMP 1110/6710

Thu15b:
Yuqing Zhai (u6865190)
Yufan Zhou (u6213488)
Huhan Jiang (u6090688)

Assignment Objectives

- The dices have highway and railway connection
- The player has to roll four ordinary dices seven times
- Fill the board with the dices, and the connection have to be valid
- Each round of game can use only one special dice, and you can only use three times of the special dices in the game
- The goal is to establish a network of routes as long as possible and connect exits as more as possible
- The final score is decided by the longest railway/highway, how many exits it connects, and how many central grids it covers

Design Approach

1. Tile
2. Method：new string [add a tile string into placement string]
3. Give viable pieces method
4. Use recursive helping method to get all solutions

1. Tiles

```java
    // which exit can the four direction of tile connect - up,right,down,left
    // highway->1, railway->2, no exit->0
    static int S0[][]={{1,1,2,1},{1,1,1,2},{2,1,1,1},{1,2,1,1},{1,1,2,1},{1,1,1,2},{2,1,1,1},{1,2,1,1}};
    static int S1[][]={{1,2,2,2},{2,1,2,2},{2,2,1,2},{2,2,2,1},{1,2,2,2},{2,1,2,2},{2,2,1,2},{2,2,2,1}};
    static int S2[][]={{1,1,1,1},{1,1,1,1},{1,1,1,1},{1,1,1,1},{1,1,1,1},{1,1,1,1},{1,1,1,1},{1,1,1,1}};
    static int S3[][]={{2,2,2,2},{2,2,2,2},{2,2,2,2},{2,2,2,2},{2,2,2,2},{2,2,2,2},{2,2,2,2},{2,2,2,2}};
    static int S4[][]={{1,2,2,1},{1,1,2,2},{2,1,1,2},{2,2,1,1},{1,1,2,2},{2,1,1,2},{2,2,1,1},{1,2,2,1}};
    static int S5[][]={{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1}};
    static int A0[][]={{2,0,0,2},{2,2,0,0},{0,2,2,0},{0,0,2,2},{2,2,0,0},{0,2,2,0},{0,0,2,2},{2,0,0,2}};
    static int A1[][]={{2,0,2,0},{0,2,0,2},{2,0,2,0},{0,2,0,2},{2,0,2,0},{0,2,0,2},{2,0,2,0},{0,2,0,2}};
    static int A2[][]={{2,2,2,0},{0,2,2,2},{2,0,2,2},{2,2,0,2},{2,0,2,2},{2,2,0,2},{2,2,2,0},{0,2,2,2}};
    static int A3[][]={{1,1,1,0},{0,1,1,1},{1,0,1,1},{1,1,0,1},{1,0,1,1},{1,1,0,1},{1,1,1,0},{0,1,1,1}};
    static int A4[][]={{1,0,1,0},{0,1,0,1},{1,0,1,0},{0,1,0,1},{1,0,1,0},{0,1,0,1},{1,0,1,0},{0,1,0,1}};
    static int A5[][]={{1,0,0,1},{1,1,0,0},{0,1,1,0},{0,0,1,1},{1,1,0,0},{0,1,1,0},{0,0,1,1},{1,0,0,1}};
    static int B0[][]={{1,0,2,0},{0,1,0,2},{2,0,1,0},{0,2,0,1},{1,0,2,0},{0,1,0,2},{2,0,1,0},{0,2,0,1}};
    static int B1[][]={{1,2,0,0},{0,1,2,0},{0,0,1,2},{2,0,0,1},{1,0,0,2},{2,1,0,0},{0,2,1,0},{0,0,2,1}};
    static int B2[][]={{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1},{1,2,1,2},{2,1,2,1}};

    // To get the connection type
    // Author: Yuqing Zhai
    public static int getvalue(String tilePlacementStringA,int x,int y)
    {
        int ans=0;
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='0') ans=S0[x][y];
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='1') ans=S1[x][y];
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='2') ans=S2[x][y];
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='3') ans=S3[x][y];
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='4') ans=S4[x][y];
        if(tilePlacementStringA.charAt(0)=='S'&&tilePlacementStringA.charAt(1)=='5') ans=S5[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='0') ans=A0[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='1') ans=A1[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='2') ans=A2[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='3') ans=A3[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='4') ans=A4[x][y];
        if(tilePlacementStringA.charAt(0)=='A'&&tilePlacementStringA.charAt(1)=='5') ans=A5[x][y];
        if(tilePlacementStringA.charAt(0)=='B'&&tilePlacementStringA.charAt(1)=='0') ans=B0[x][y];
        if(tilePlacementStringA.charAt(0)=='B'&&tilePlacementStringA.charAt(1)=='1') ans=B1[x][y];
        if(tilePlacementStringA.charAt(0)=='B'&&tilePlacementStringA.charAt(1)=='2') ans=B2[x][y];
        return ans;
    }
```

2. Well Formed Placement

```java
public static boolean isTilePlacementWellFormed(String tilePlacementString) {

    // whether it contains only 5 characters
    if(tilePlacementString.length()/5 !=1 || tilePlacementString.length()%5 !=0){
        return false;
    }

    // whether the first character represents a die A or B, or a special tile S
    if(tilePlacementString.charAt(0) != 'A' && tilePlacementString.charAt(0) != 'B' &&
            tilePlacementString.charAt(0) != 'S'){
        return false;
    }

    // when the first character is A or S, whether the second character between 0-5
    if(tilePlacementString.charAt(0) == 'A' && tilePlacementString.charAt(1)>'5'){
        return false;
    }

    if(tilePlacementString.charAt(0) == 'S' && tilePlacementString.charAt(1)>'5'){
        return false;
    }

    // when the first character is B, whether the second character between 0-2
    if(tilePlacementString.charAt(0) == 'B' && tilePlacementString.charAt(1)>'2'){
        return false;
    }

    // whether the third character is between A and G
    if(tilePlacementString.charAt(2) > 'G' || tilePlacementString.charAt(2) < 'A'){
        return false;
    }

    // whether the fourth character is between 0 and 6
    if(!(tilePlacementString.charAt(3) == '0' || tilePlacementString.charAt(3) == '1'||
            tilePlacementString.charAt(3) == '2'||tilePlacementString.charAt(3) == '3'||
            tilePlacementString.charAt(3) == '4'||tilePlacementString.charAt(3) == '5'||
            tilePlacementString.charAt(3) == '6')){
        return false;
    }

    // whether the fifth character is between 0 and 7
    if(tilePlacementString.charAt(4) > '7'){
        return false;
    }
    // FIXME Task 2: determine whether a tile placement is well-formed
    return true;
}
```

3. Is Board String Well Formed

```java
    public static boolean isBoardStringWellFormed(String boardString) {

        // whether a board string is null or empty
        if(boardString == null || boardString == ""){
            return false;
        }

        // whether a board string has exactly N five-character tile placements, and not too long
        if(boardString.length()%5 != 0) {
            return false;
        }else if(boardString.length()/5 > 31){
            return false;
        }

        int count = 0;
        for(int i=0; i<boardString.length(); i+=5){
            // slice the board string into a single one
            String newString = boardString.substring(i,i+4);
            char margin = boardString.charAt(i+4);
            newString = newString + margin;
            // test whether it is well performed
            if(!isTilePlacementWellFormed(newString)){
                return false;
            }
            // count the number of special pieces
            if(newString.charAt(0) == 'S'){
                count++;
            }
        }

        // if special pieces is more than 3 then it is not valid
        if(count > 3){
            return false;
        }else{
            return true;
        }
        // FIXME Task 3: determine whether a board string is well-formed
    }
```

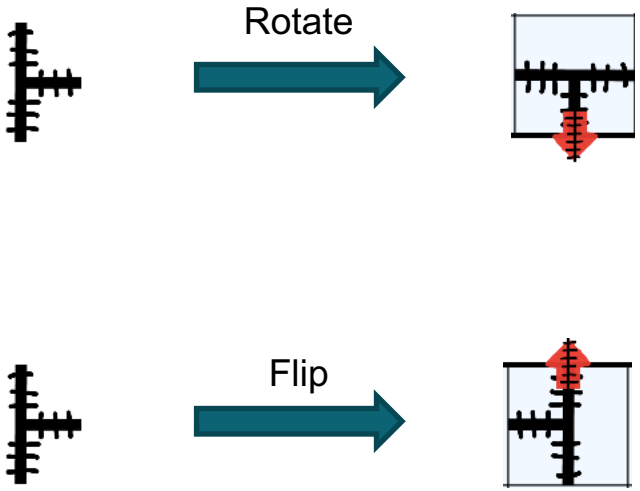
4. Are Connected Neighbors

```java
243     public static boolean areConnectedNeighbours(String tilePlacementStringA, String tilePlacementStringB) {
244
245         // first judge neighbour
246         if (areNeighbours(tilePlacementStringA, tilePlacementStringB) == -1) {
247             return false;
248         }
249         // If they are neighbours, whether they can successfully connect.
250         if(tilePlacementStringA.charAt(2) - tilePlacementStringB.charAt(2) == 1) {
251             if(areConnected(tilePlacementStringA,tilePlacementStringB, indexA: 0, indexB: 2))
252                 return true;
253         }
254
255         if(tilePlacementStringA.charAt(2)-tilePlacementStringB.charAt(2) == -1) {
256             if(areConnected(tilePlacementStringA,tilePlacementStringB, indexA: 2, indexB: 0))
257                 return true;
258         }
259
260         if(tilePlacementStringA.charAt(3)-tilePlacementStringB.charAt(3) == 1) {
261             if(areConnected(tilePlacementStringA,tilePlacementStringB, indexA: 3, indexB: 1))
262                 return true;
263         }
264
265         if(tilePlacementStringA.charAt(3)-tilePlacementStringB.charAt(3) == -1) {
266             if(areConnected(tilePlacementStringA,tilePlacementStringB, indexA: 1, indexB: 3))
267                 return true;
268         }
269         // FIXME Task 5: determine whether neighbouring placements are connected
270         return false;
271     }
```

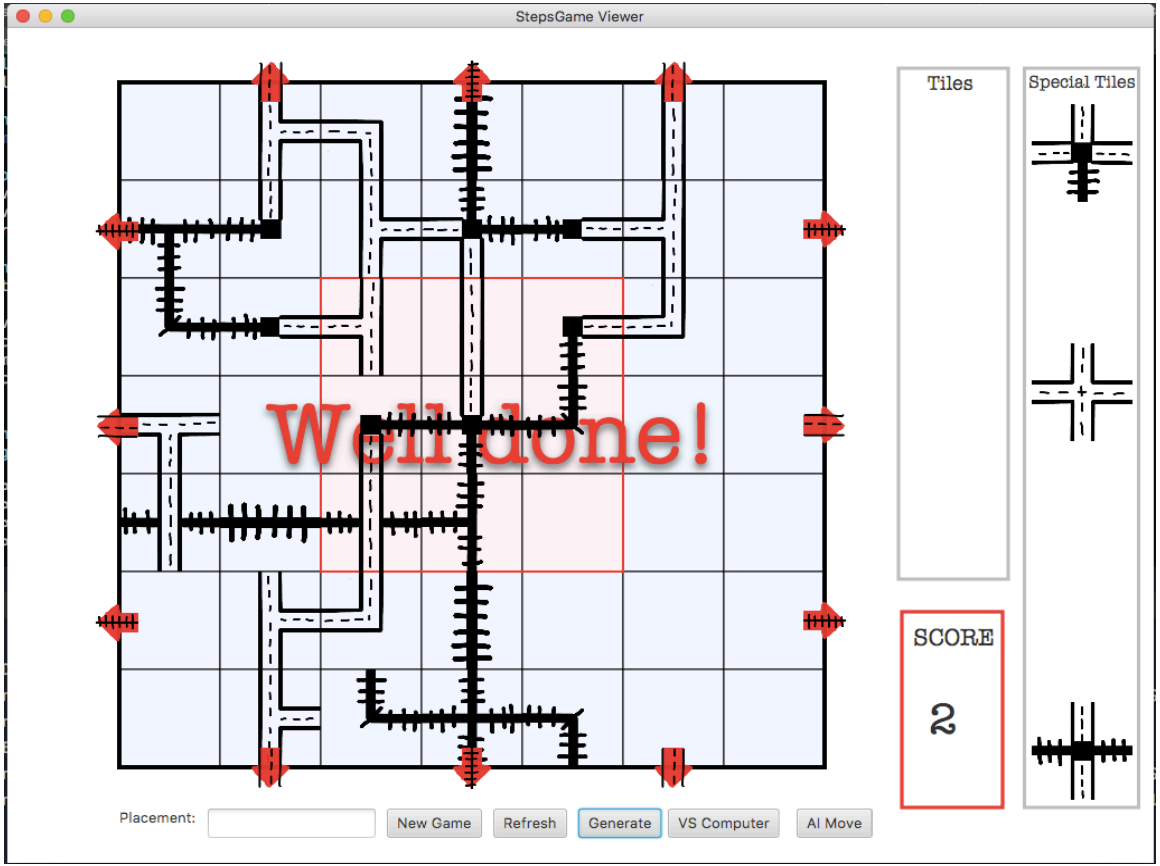
5. Get solutions

```java
    public static int getBasicScore(String boardString) {
        getGroups(boardString);

        int ans = 0;
        // Number of Exits connected to route | 2 | 3 | 4  | 5 |  6 |  7 |  8 |  9 |  10 | 11 | 12 |
        // Points Awarded                     | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 45 |
        for(int i=0;i<group.size();i++){
            int exits=0;
            int error=0;
            for(int j=0;j+4<group.get(i).length();j=j+5){
                String temp=group.get(i).substring(j,j+5);
                for(int k=0;k<4;k++) {
                    if(mapNode.get(temp).direction[k]==-1) exits++;
                    if(mapNode.get(temp).direction[k]==-2){
                        error++;
                    }
                }
            }
            if(exits==2) ans+=4;
            if(exits==3) ans+=8;
            if(exits==4) ans+=12;
            if(exits==5) ans+=16;
            if(exits==6) ans+=20;
            if(exits==7) ans+=24;
            if(exits==8) ans+=28;
            if(exits==9) ans+=32;
            if(exits==10) ans+=36;
            if(exits==11) ans+=40;
            if(exits==12) ans+=45;

            ans-=error;
        }
        //centre grid that are covered
        ans += getCentreGrids(boardString);
        // FIXME Task 8: compute the basic score
        return ans;
    }
```

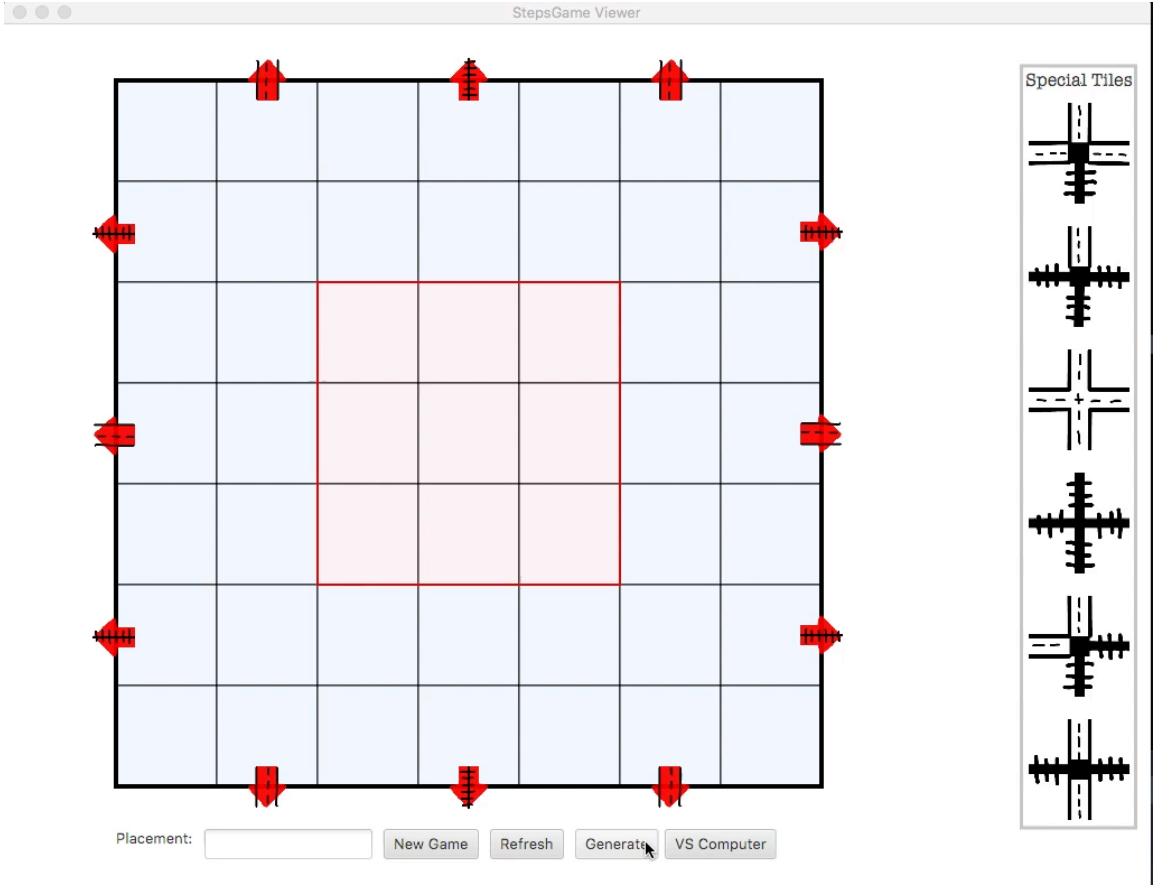
Rotating and flipping



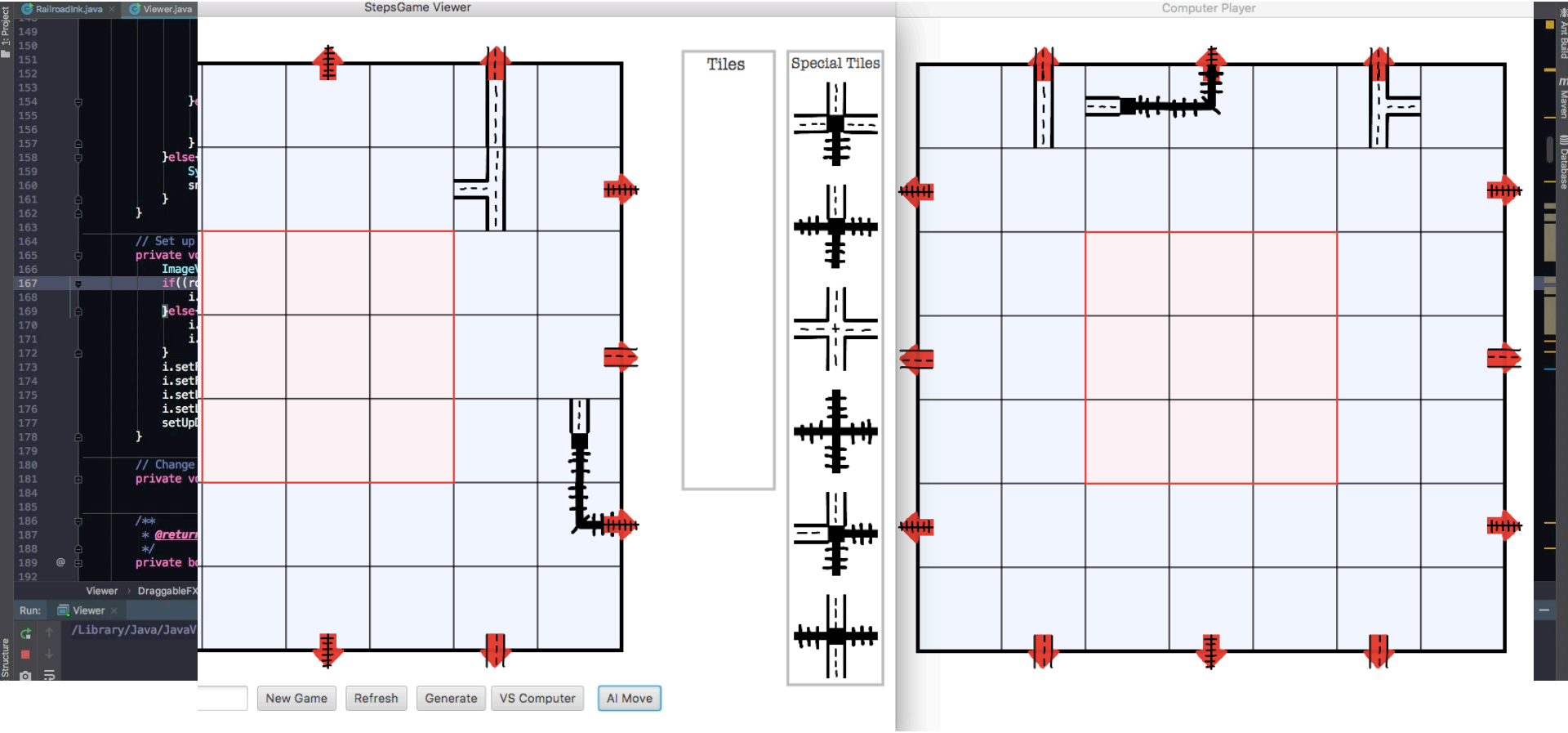
Solutions


StepsGame Viewer
Tiles
Special Tiles
Well done!
SCORE
2
Placement:
New Game
Refresh
Generate
VS Computer
AI Move

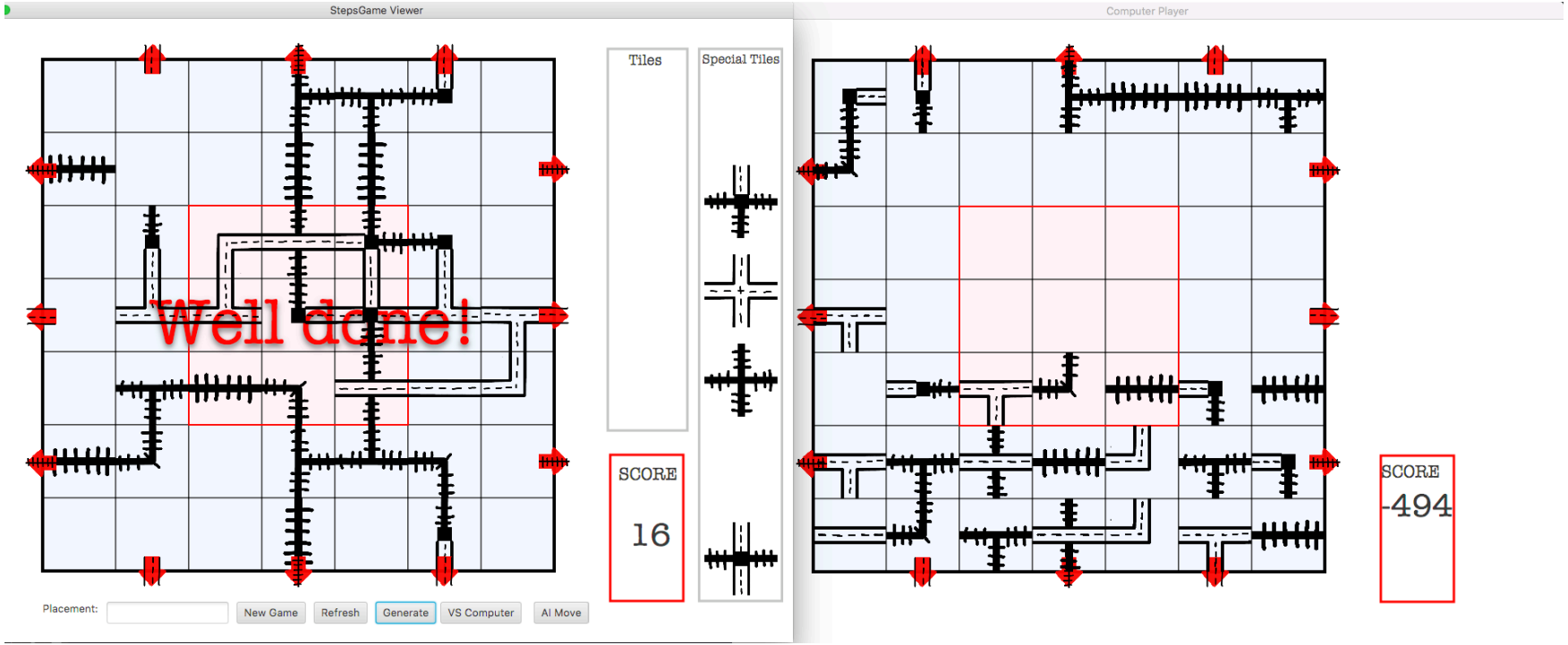
Single Player



Multi-player



Interesting part



Thank you

Any questions?