Assignment2:
IQ-Step(draft)

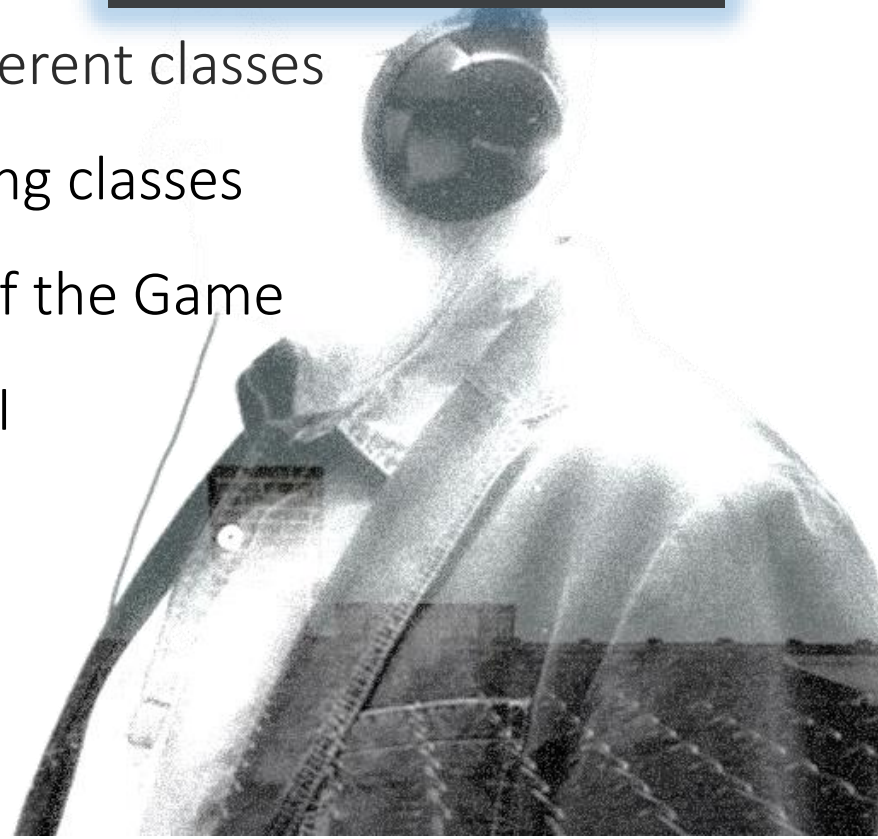
Shiqin Huo(u5949730)
Wenjun Yang(u6251843)
Xiangyi Luo(u6162693)

DESIGN

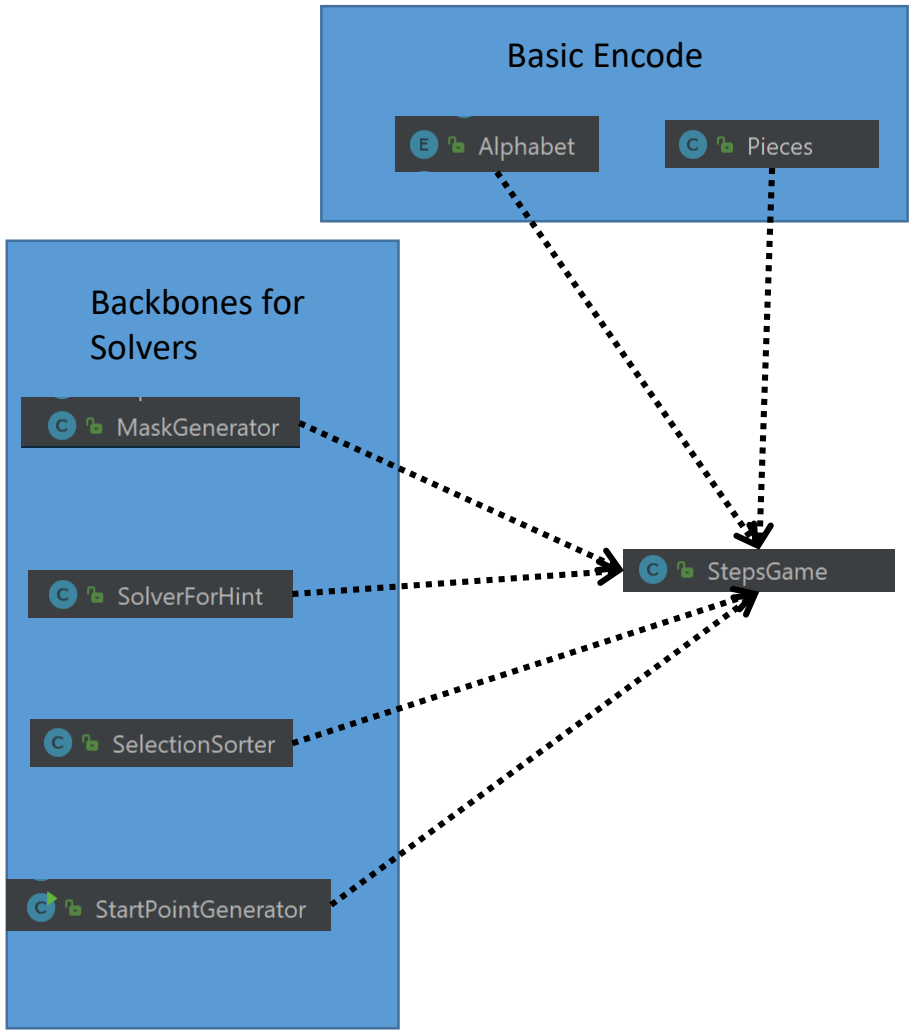
- ✓ Designing skeleton
- ✓ Building basic classes:

  basic tool methods in different classes
- ✓ Class StepGame: Combining classes

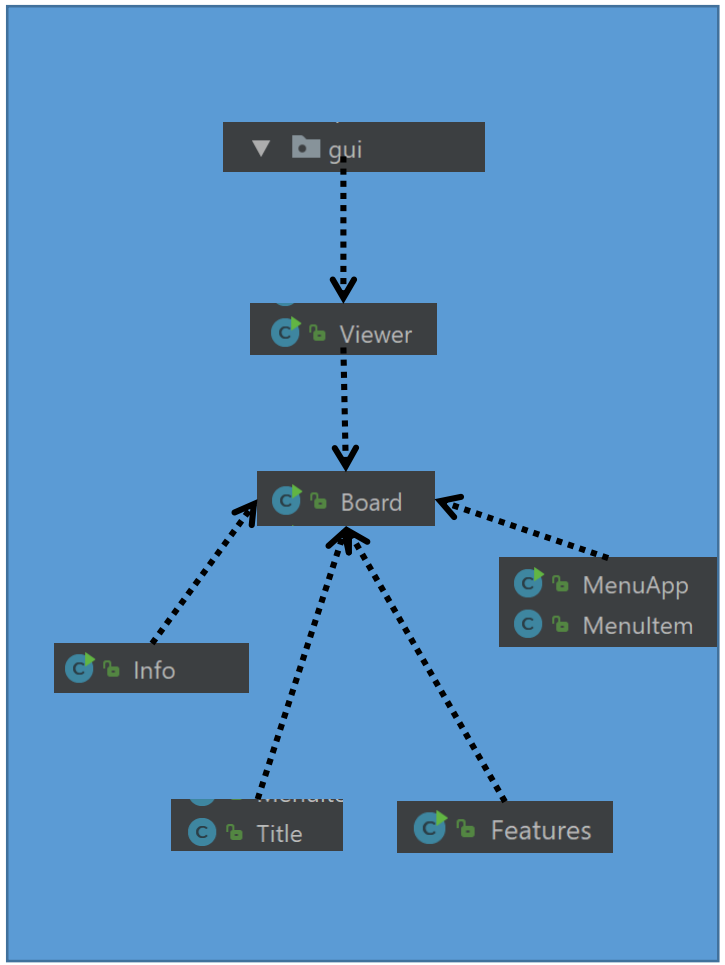
  and implement features of the Game
- ✓ Class Board: implement UI



Back-End

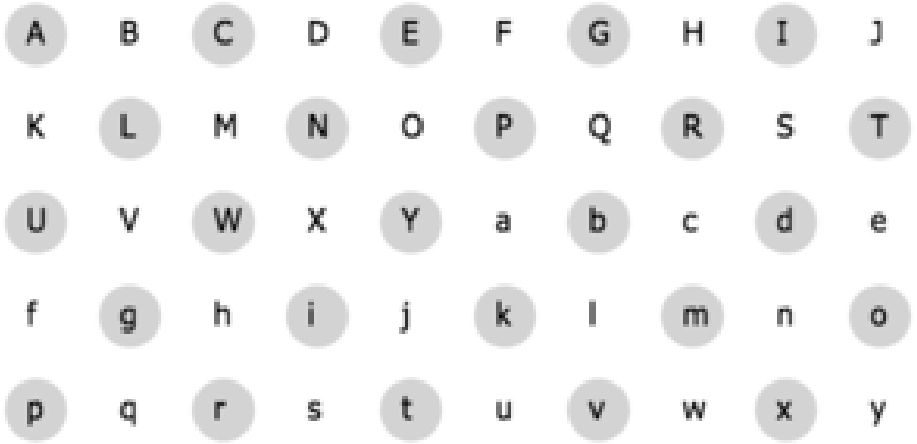


UI



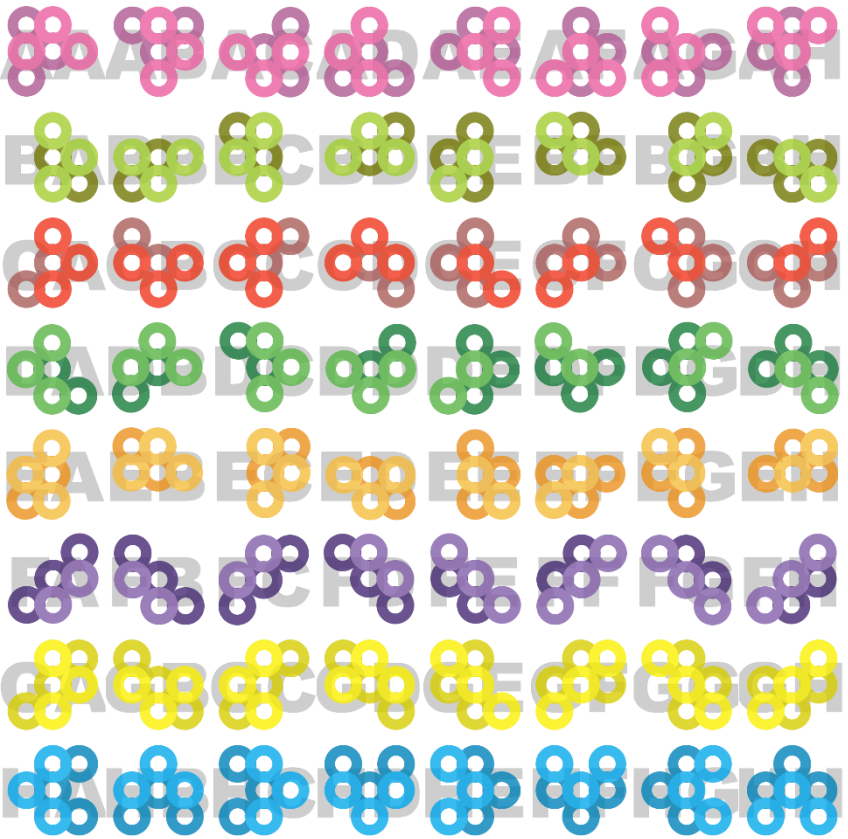
DIAGRAM

Encoding Board



- The positions on the board
- A to Y ; a to y
- Encoded as 0 to 49
- Pegs can be detected

Encoding Pieces

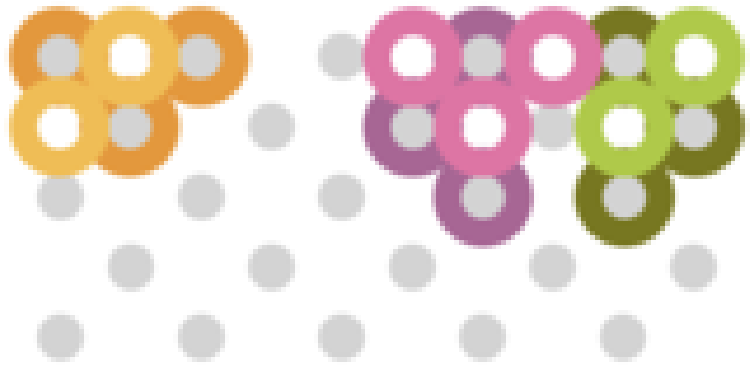


A string contains:
Three Chars

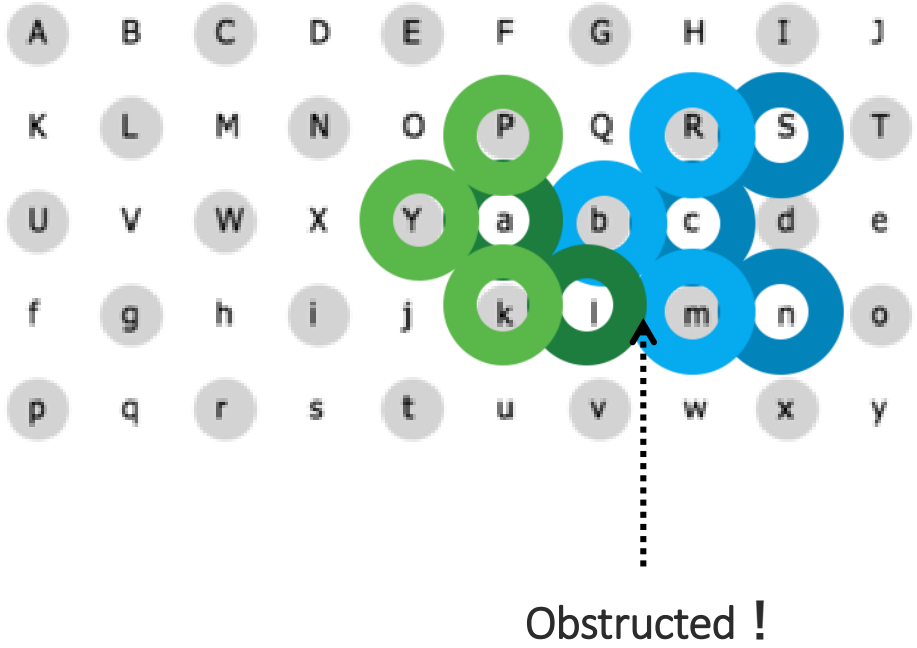
1 The shape
2 The rotation
3 The position
(on the board)

Checking Validity

- The piece is in the board range;
- At most eight pieces are on the board;
- Each shape should be used at most once;

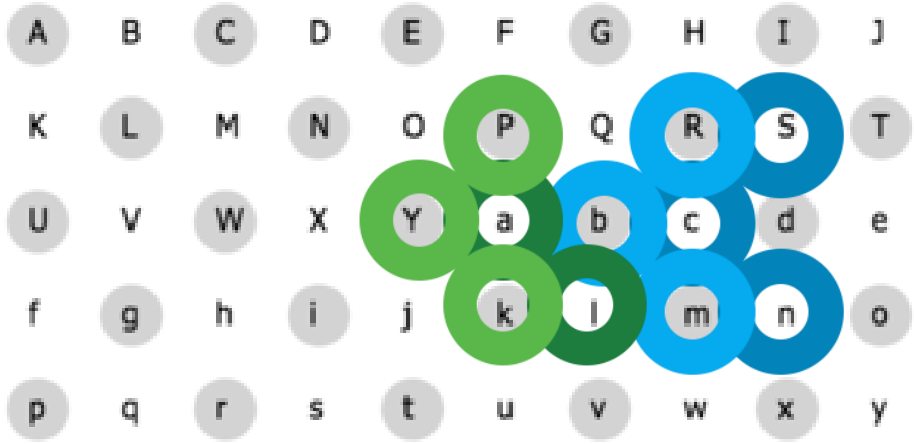


Checking Obstruction



- Obstruction:
- If a bottom ring is placed after a top ring, says obstructed.

Checking Obstruction

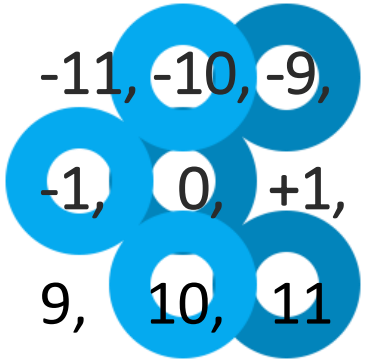


Obstructed！

- How to detect obstruction?
- If a peg position has been located
- Then the adjacent non-peg positions are not allowed to be located at the next move

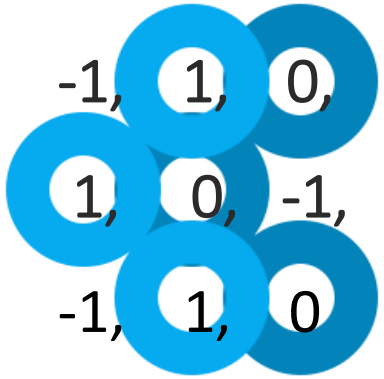
Checking Obstruction

```
/*Reflect the relationship of the 9 grids*/
private final static int[][] grid3x3 = {
        {-11, -10, -9},
        {-1, 0, +1},
        {+9, +10, +11}
};
```



- Firstly, think about represent pieces in 3x3 grids.(Get idea from lecture code in Class Boggle)

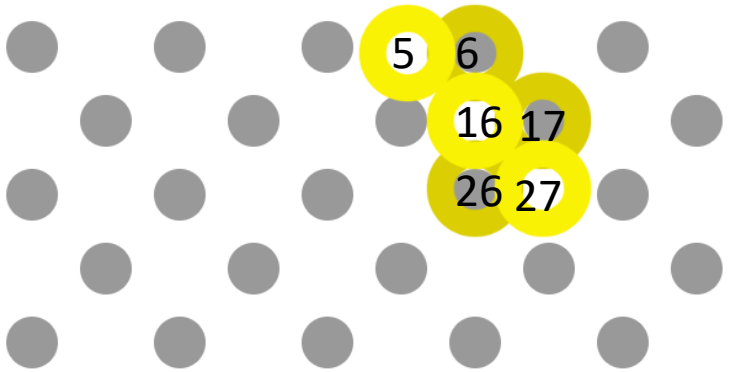
Checking Obstruction



- Secondly,
- 1 for top ring
- 0 for bottom ring
- -1 for nothing

Checking Obstruction

Example:
Placement: GGQ
Next piece: DFO

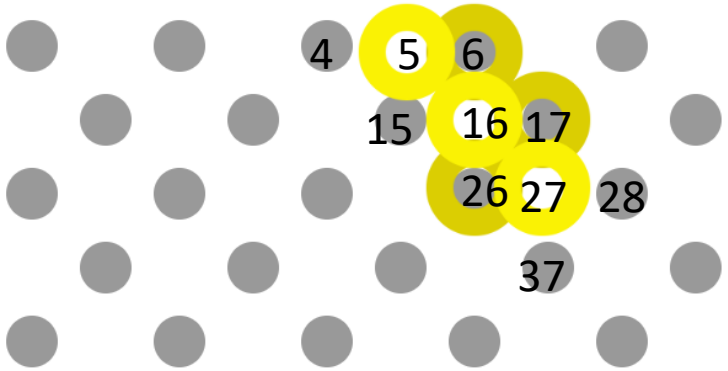


In this example used positions are:
[16, 17, 5, 6, 26, 27]

- Thirdly,
- Recording all the positions the given placement has used.
- Store in a HashSet

Checking Obstruction

Example:
Placement: GGQ
Next piece: DFO



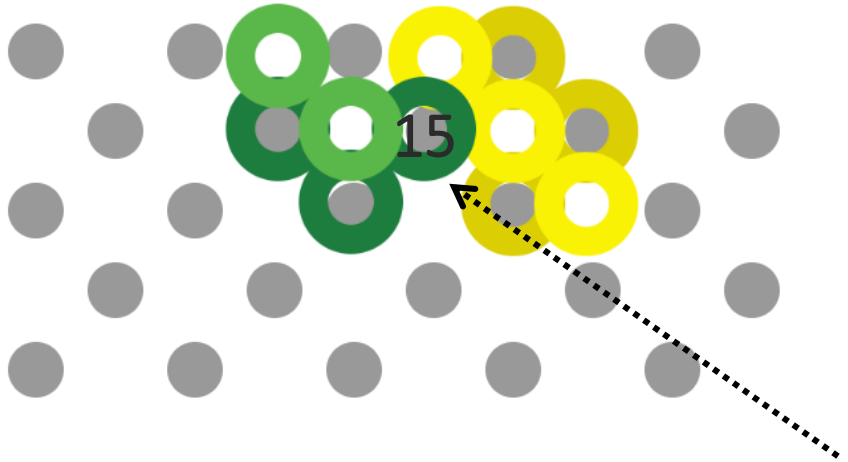
In this example used positions :
[16, 17, 5, 6, 26, 27]

Then we can compute the positions which cannot be used at next move：
[16, 17, 4, 5, 37, 6, 26, 27, 28, 15]

- According to the algorithm mentioned :
- If a peg position has been located
- Then the adjacent non-peg positions are not allowed to be located at the next move

Checking Obstruction

Example:
Placement: GGQ
Next piece: DFO



In this example used positions :
[16, 17, 5, 6, 26, 27]

Then we can compute the positions which cannot be used at next move：
[16, 17, 4, 5, 37, 6, 26, 27, 28, 15]

Now check positions the DFO will take:
[3, 24, 13, 14, 15]
The pieces is going to use 15 which is not allowed.

Hence, the notObstruct method return *false !*

Reordering

Example:
Placement: DFOGGQEDI
Objective: DFOGGQEDIBAkFHnHCiAALCAg

Reordering

Example:
Placement: DFOGGQEDI
Objective: DFOGGQEDIBAkFHnHCiAALCAg

In this case,
We get four different orders for one objective:

[DFOGGQEDIBAkHCiFHnAALCAg,
DFOGGQEDIBAkFHnHCiAALCAg,
DFOGGQEDIBAkHCiAALCAgFHn,
DFOGGQEDIBAkHCiAALFHnCAg]

MaskGenerator

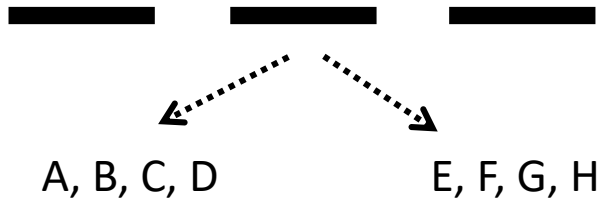
```java
public static ArrayList<String> maskGenerator1(char first){
    //All the positions where a piece of placement which starts in A, C, D, F, G, H can be placed on
    char[] ACDFGH1 = {'L', 'N', 'P', 'R', 'W', 'Y', 'b', 'd', 'g', 'i', 'k', 'm'};
    char[] ACDFGH2 = {'M', 'O', 'Q', 'S', 'V', 'X', 'a', 'c', 'h', 'j', 'l', 'n'};
    ArrayList<String> newArr = new ArrayList<>();
    //different possible states for one single mask
    char[] second1 = {'A','B','C','D'};  char[] second2 = {'E','F','G','H'};
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'A','B','C','D'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH1.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second1[i])+String.valueOf(ACDFGH1[j]));
        }
    }
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'E','F','G','H'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH2.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second2[i])+String.valueOf(ACDFGH2[j]));
        }
    }
    return newArr;
}
```

A, B, C…H

Getting Solution (unordered)

MaskGenerator

```java
public static ArrayList<String> maskGenerator1(char first){
    //All the positions where a piece of placement which starts in A, C, D, F, G, H can be placed on
    char[] ACDFGH1 = {'L', 'N', 'P', 'R', 'W', 'Y', 'b', 'd', 'g', 'i', 'k', 'm'};
    char[] ACDFGH2 = {'M', 'O', 'Q', 'S', 'V', 'X', 'a', 'c', 'h', 'j', 'l', 'n'};
    ArrayList<String> newArr = new ArrayList<>();
    //different possible states for one single mask
    char[] second1 = {'A','B','C','D'};  char[] second2 = {'E','F','G','H'};
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'A','B','C','D'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH1.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second1[i])+String.valueOf(ACDFGH1[j]));
        }
    }
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'E','F','G','H'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH2.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second2[i])+String.valueOf(ACDFGH2[j]));
        }
    }
    return newArr;
}
```

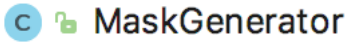


Getting Solution (unordered)

MaskGenerator

```java
public static ArrayList<String> maskGenerator1(char first){
    //All the positions where a piece of placement which starts in A, C, D, F, G, H can be placed on
    char[] ACDFGH1 = {'L', 'N', 'P', 'R', 'W', 'Y', 'b', 'd', 'g', 'i', 'k', 'm'};
    char[] ACDFGH2 = {'M', 'O', 'Q', 'S', 'V', 'X', 'a', 'c', 'h', 'j', 'l', 'n'};
    ArrayList<String> newArr = new ArrayList<>();
    //different possible states for one single mask
    char[] second1 = {'A','B','C','D'};  char[] second2 = {'E','F','G','H'};
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'A','B','C','D'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH1.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second1[i])+String.valueOf(ACDFGH1[j]));
        }
    }
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'E','F','G','H'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH2.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second2[i])+String.valueOf(ACDFGH2[j]));
        }
    }
    return newArr;
}
```

Getting Solution (unordered)



```java
public static ArrayList<String> maskGenerator1(char first){
    //All the positions where a piece of placement which starts in A, C, D, F, G, H can be placed on
    char[] ACDFGH1 = {'L', 'N', 'P', 'R', 'W', 'Y', 'b', 'd', 'g', 'i', 'k', 'm'};
    char[] ACDFGH2 = {'M', 'O', 'Q', 'S', 'V', 'X', 'a', 'c', 'h', 'j', 'l', 'n'};
    ArrayList<String> newArr = new ArrayList<>();
    //different possible states for one single mask
    char[] second1 = {'A','B','C','D'};  char[] second2 = {'E','F','G','H'};
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'A','B','C','D'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH1.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second1[i])+String.valueOf(ACDFGH1[j]));
        }
    }
    //connect the type of mask, state of mask and position of mask when the state of mask is in {'E','F','G','H'}
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < ACDFGH2.length; j++){
            newArr.add(String.valueOf(first)+String.valueOf(second2[i])+String.valueOf(ACDFGH2[j]));
        }
    }
    return newArr;
}
```

Map

Key : type of mask Value : possible positions correspond to mask in the key

Getting Solution (unordered)

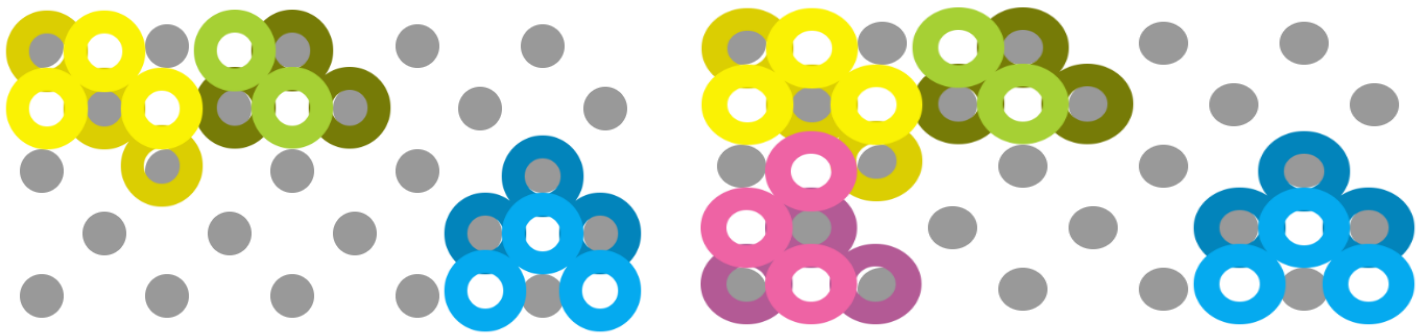
Algorithm

1. Delete all the states of existing masks on the board.

2. Append the remaining masks with different states
   on the board, judge whether it is valid by
   FUNCTION notObstruct and isPlacementSequenceValid.
   If it is valid, link the candidate to the original placement
   and add them into a ArrayList. Otherwise, abandon the states.

3. Do it recursively.

Getting Solution (unordered)

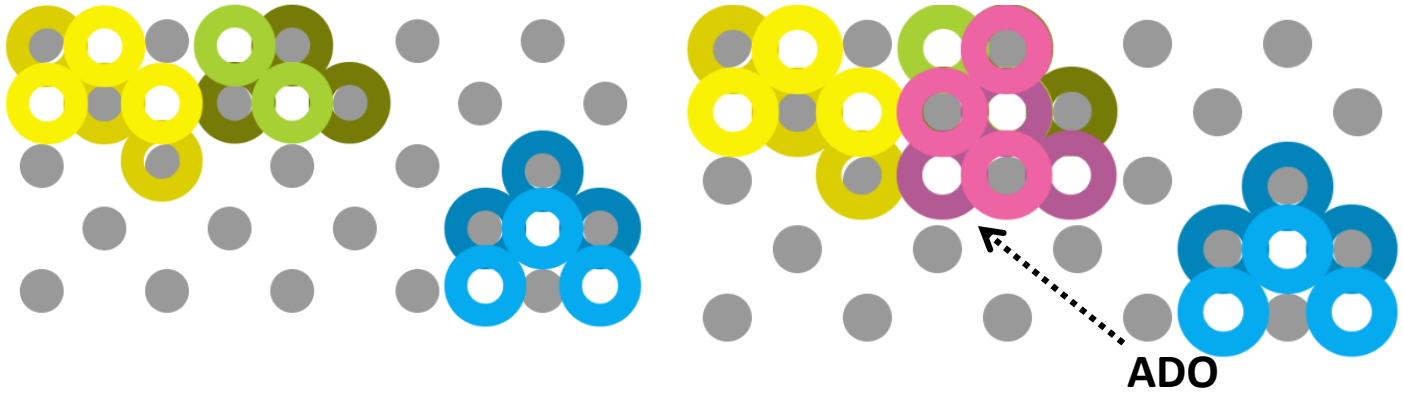


HHnBFOGDL

ADg

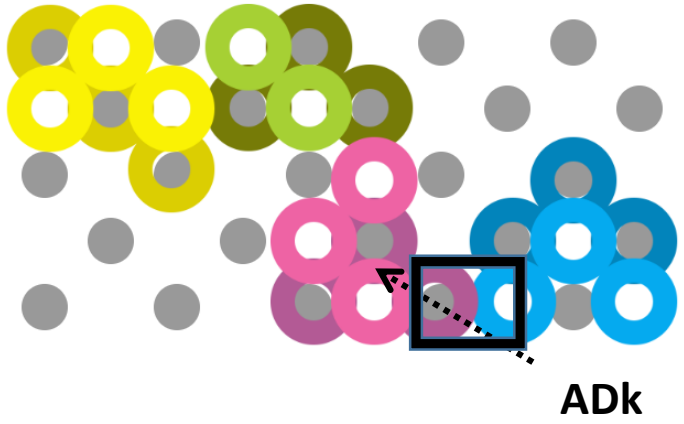
HHnBFOGDLADg

Getting Solution (unordered)



Getting Solution (unordered)

HHnBFOGDL



ADk

Abandon!

Getting Solution (unordered)

Test on computer in N109
Around 1 min

Test on computer in N114
Around 1 min 20s

Test on own Mbp
Around 1 min 40s

Getting Solution (unordered)

Why was it slow?

We make comparisons with another group:
FUNCTION - isPlacementSequenceValid
Ours: 8ms to tell a piece placement is invalid
Theirs: 15ms
Ours: 24ms to tell a piece placement is valid
Theirs: 8ms

Improvement

Improve the efficiency of FUNCTION

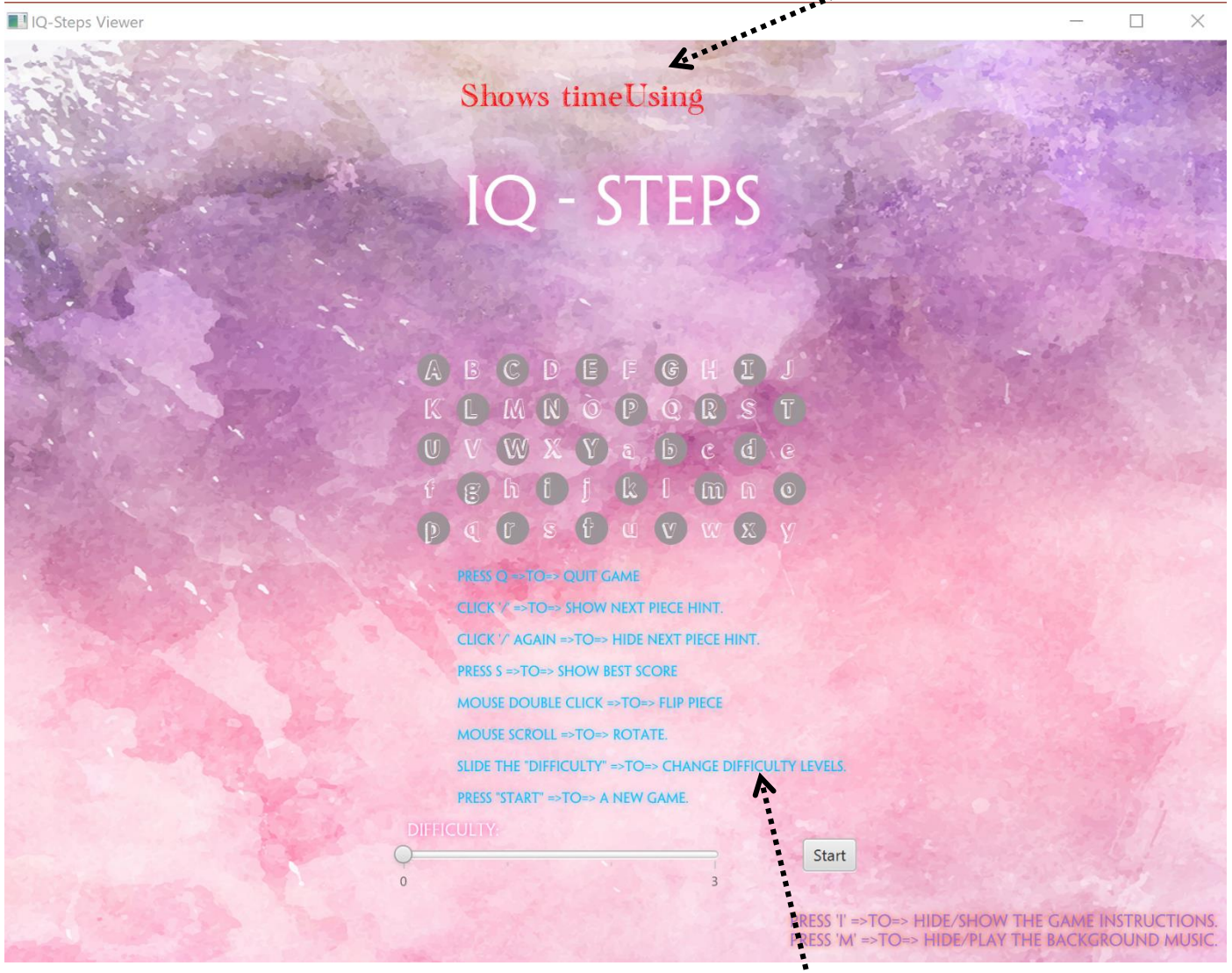
notObstruct and isPlacementSequenceValid !!

Getting Solution (unordered)

Game Menu


Welcome to IQ STEPS!
IQ - STEPS
START
DIFFICULTY
RELOAD
INFORMATION
EXIT TO DESKTOP
Backgroud
Pop – up effect

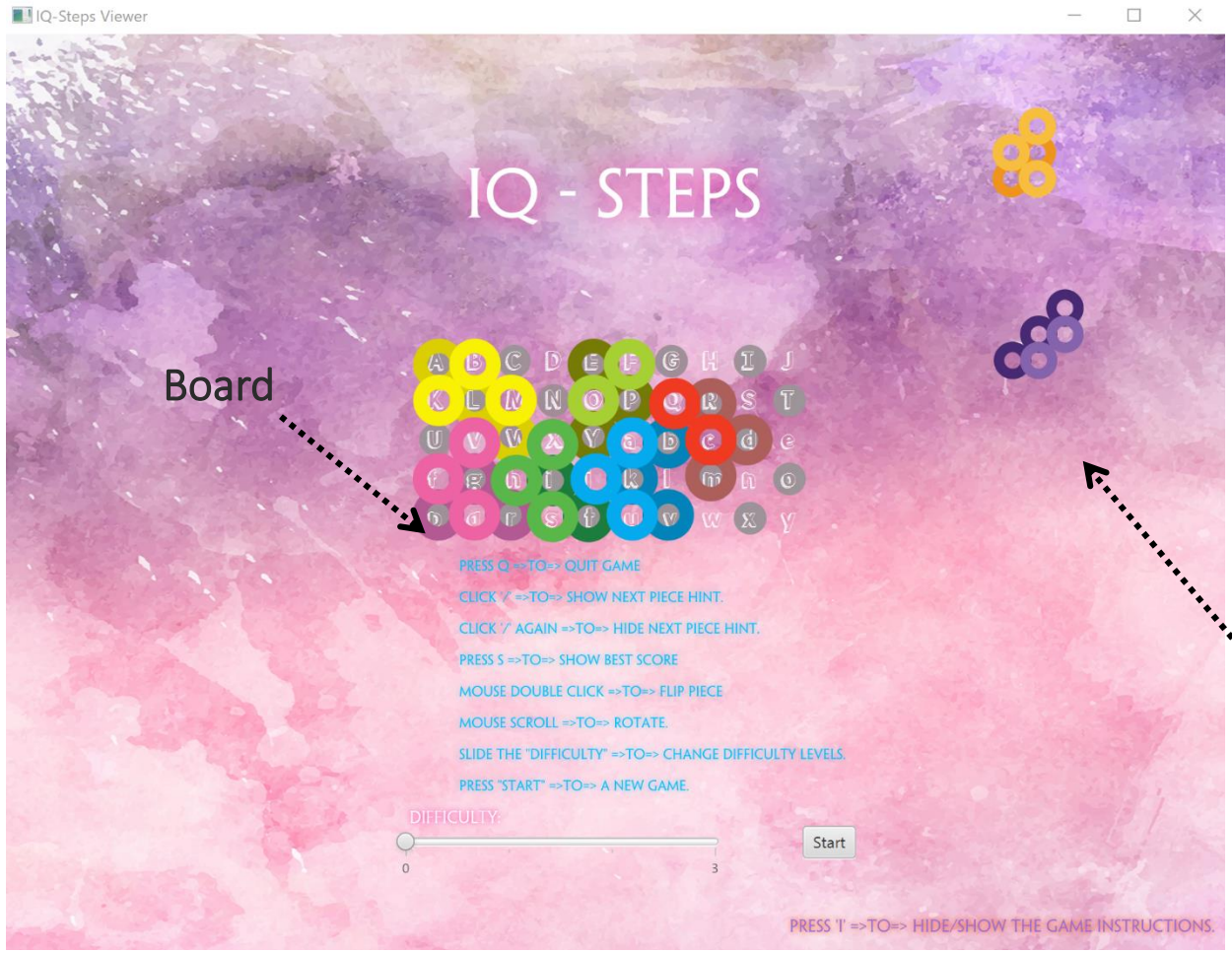
Timer



Clear Instruction for new user

Game Starting

Game Process

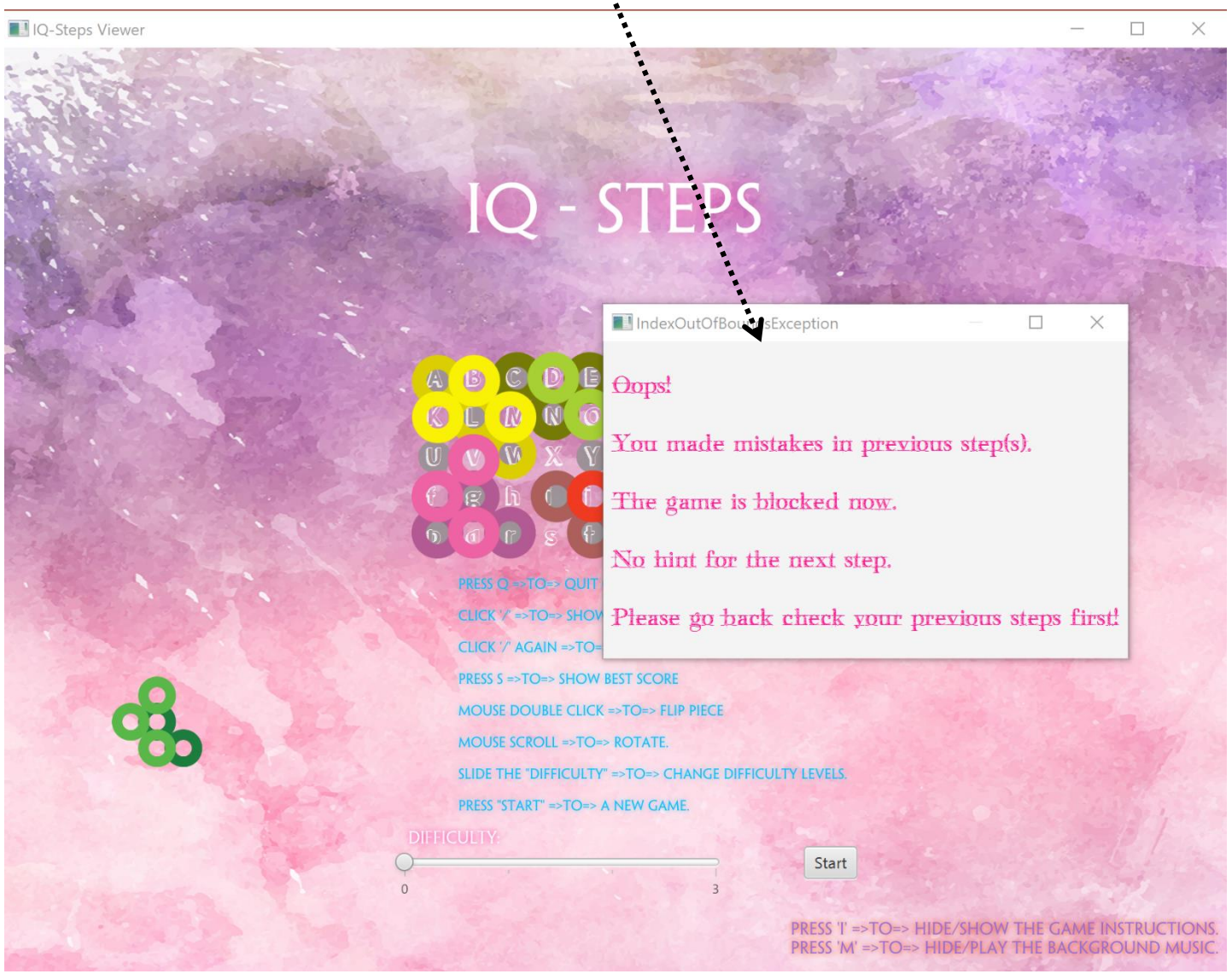


- Starting layer: all pieces at home
- Draggable layer
- Double click: reloading flip piece

Press “/” and get semitransparent hint (not available if the last move was wrong)

Game Hint

Exception window if Player make mistake in previous step.



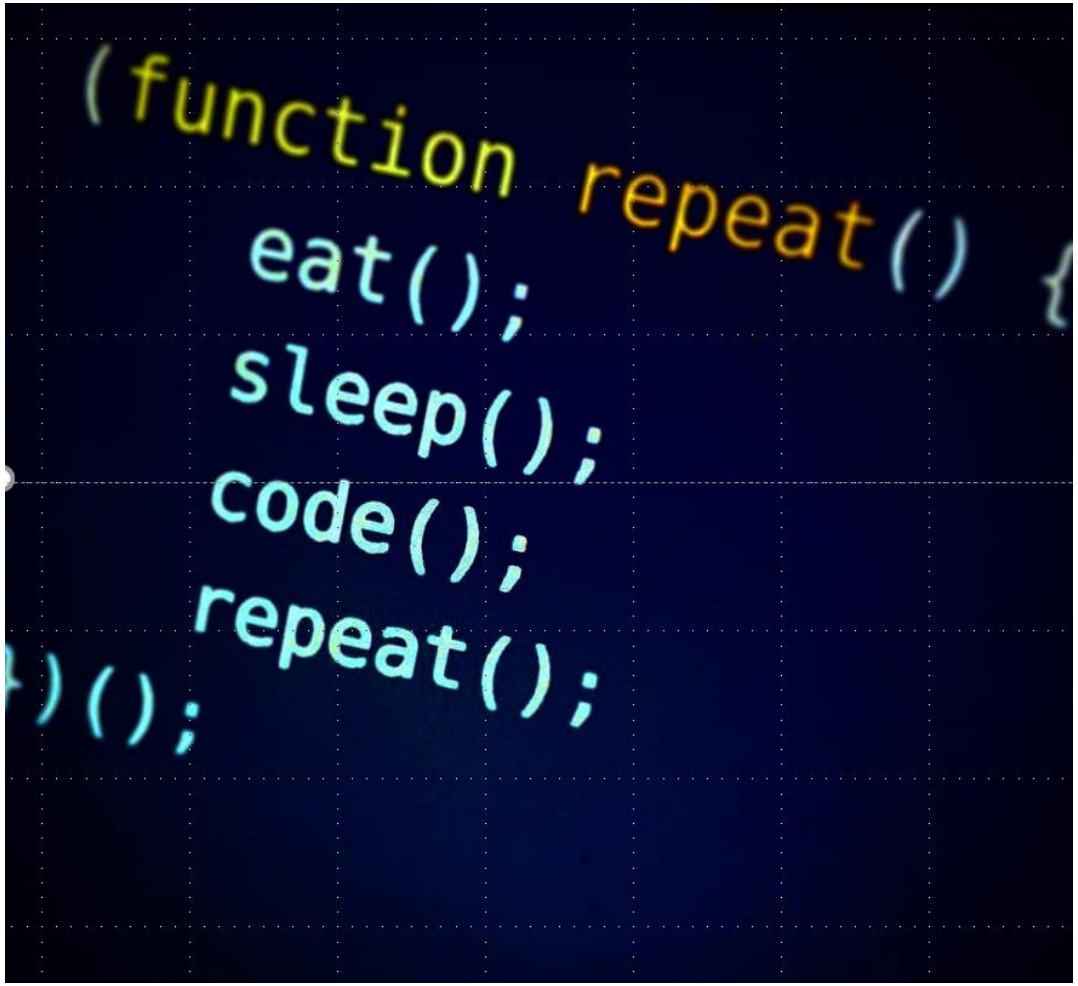
Game Hint

Best Score


IQ-Steps Viewer
IQ - STEPS
Showing best Score
The best score for l...
11.9 s!
PRESS S =>TO=> SHOW BEST SCORE
MOUSE DOUBLE CLICK =>TO=> FLIP PIECE
MOUSE SCROLL =>TO=> ROTATE.
SLIDE THE "DIFFICULTY" =>TO=> CHANGE DIFFICULTY LEVELS.
PRESS "START" =>TO=> A NEW GAME.
DIFFICULTY:
0
3
Start
PRESS 'I' =>TO=> HIDE/SHOW THE GAME INSTRUCTIONS.
PRESS 'M' =>TO=> HIDE/PLAY THE BACKGROUND MUSIC.

Game Complete

Information window



Finally,
Lots of coding,
debugging

More Fun

About Sound Effect

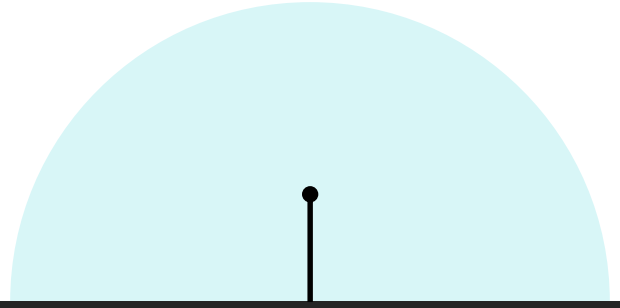
When player put piece in invalid;
When player complete the game;
When…

About UI

Artistic background;
Pop-up effect;
……

About Cooperation

As designers, we communicated with each other and share ideas!



THANKS

Designed BY
Shiqin Huo
Wenjun Yang
Xiangyi Luo