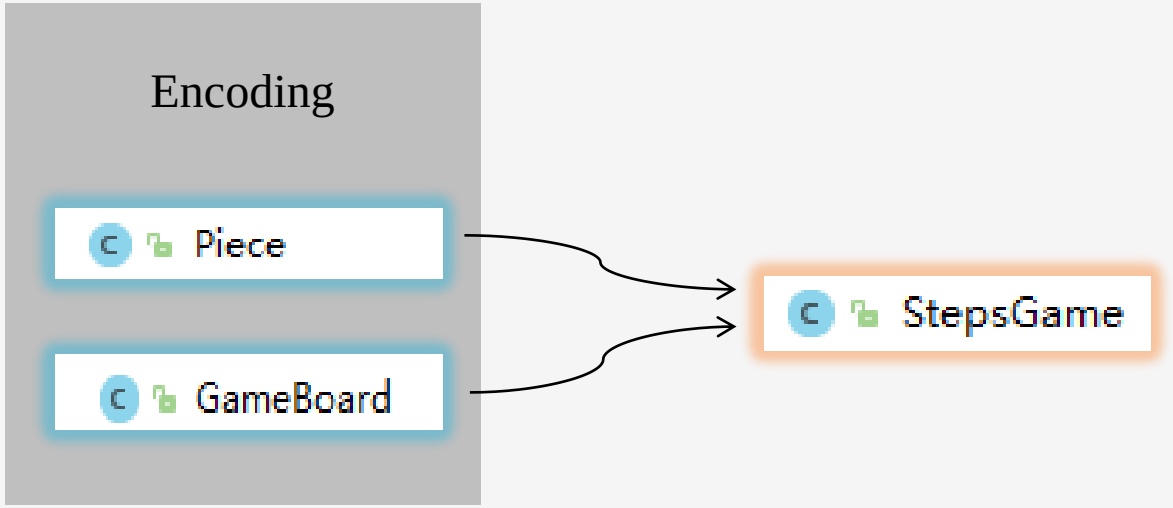
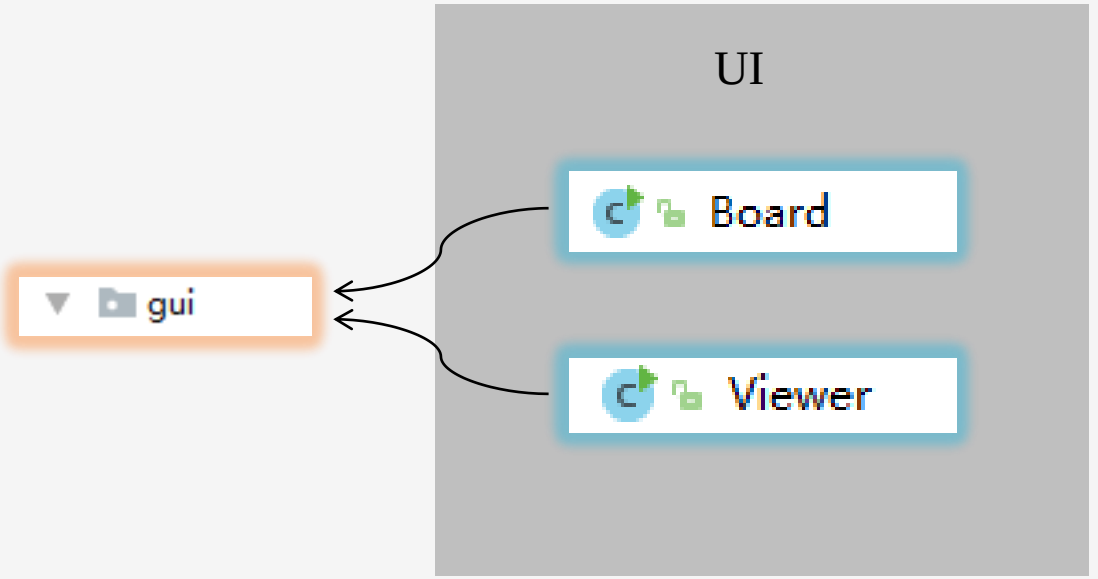
IQ-STEPS

Chan Xu (u6233112)
Yiwen Peng (u6071714)
Qingsen Kuang(u5917277)

Design





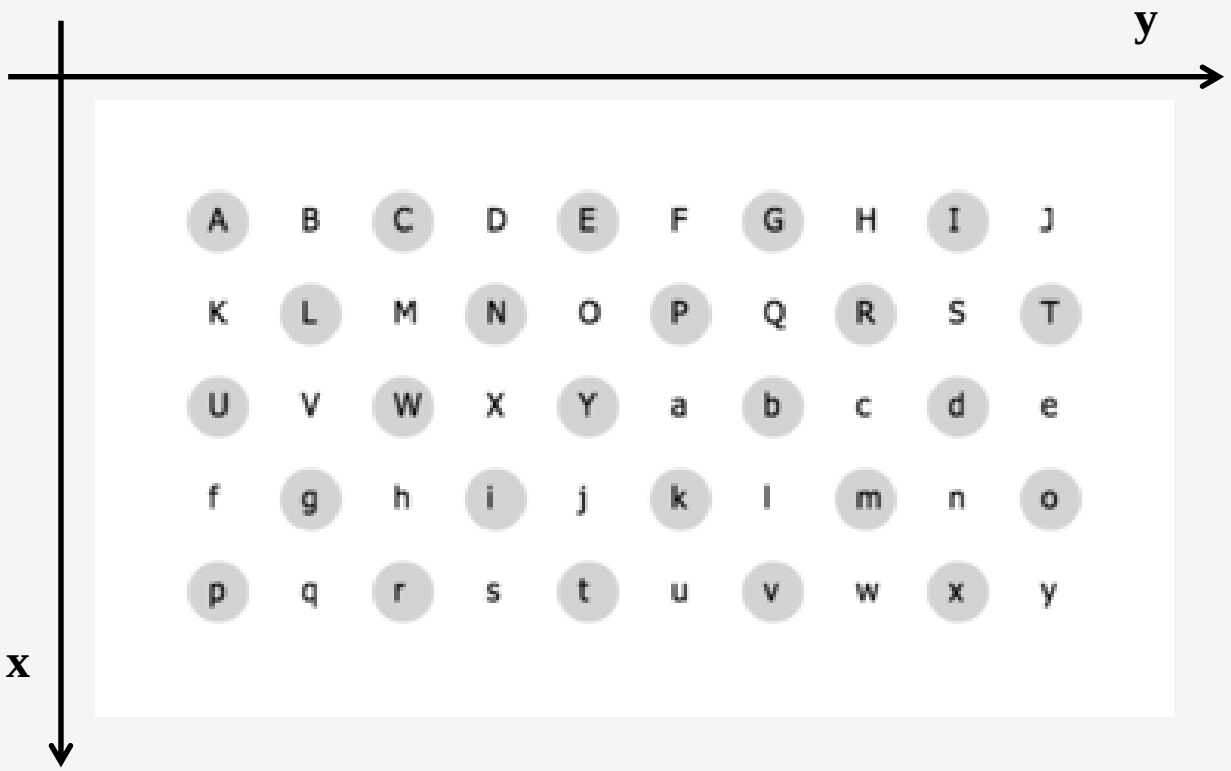
- Building basic classes:

  Piece, GameBoard
- StepsGame: Combine classes
- UI: Board class
- (include piece draggablePiece)

Encoding Board

- Positions of board: A – Y & a – y
- Encoded to 0 – 49
- Convert to x: 0 – 4

  y: 0 – 9



Encoding Pieces

- A piece string: Three chars

  1.The shape

  2. The orientation

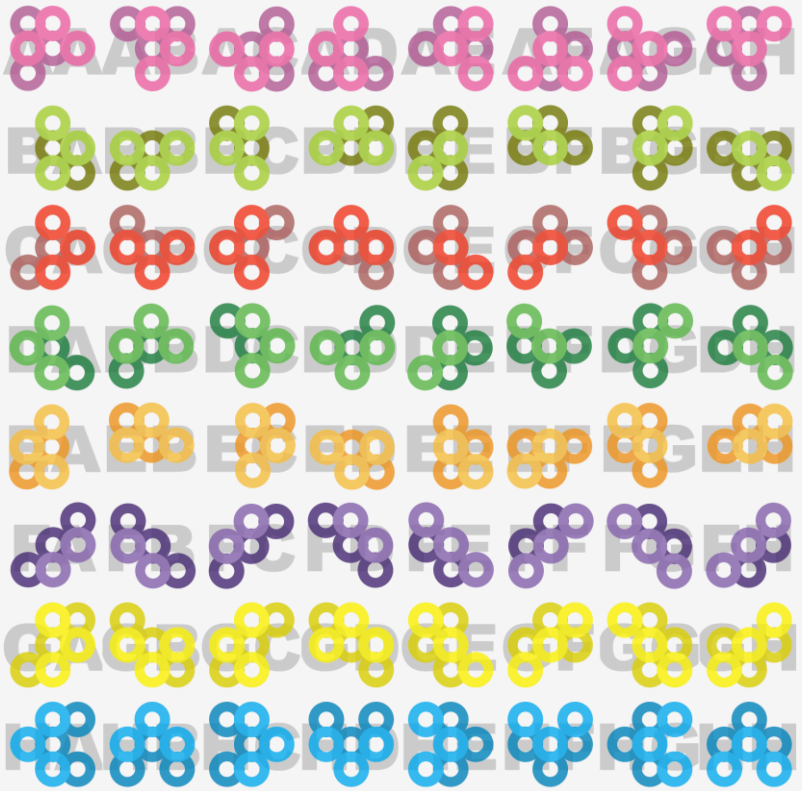
  3. The location (on the board)

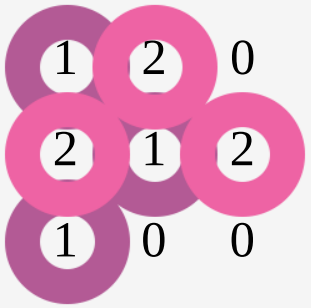
- Represented by number: ‘0’,’1’,’2’

  ‘0’ means no ring

  ‘1’ means low layer ring

  ‘2’ means high layer ring





Pieces Validity

- The piece is in the range of the board
- At most eight pieces can be put into board
- Each piece can be used at most once

Checking Obstruction

Obstruction:

- The low layer ring is placed after a high layer ring.

Obstruction!

Checking Obstruction

How to fix obstruction ?

- Firstly,
- Get the coordinates.

```
/* get the coordinates of home ring in board. */
private class Coord {
    int x, y;

    Coord(int pos) {
        x = pos / 10;
        y = pos % 10;
    }

    Coord(int X, int Y) {
        this.x = X;
        this.y = Y;
    }
}
```

Checking Obstruction

How to fix obstruction ?

- Secondly,
- Place low rings and check neighbors
- How to check neighbors?

- If the ring types of neighbor coordinates is high, this ring cannot be placed.

```
private boolean checkNeighbours(int currentX, int currentY, int ringType) {
    if (ringType == Piece.Low)
    {
        Coord[] neighbours = new Coord[]
                {
                        new Coord( X: currentX - 1, currentY),
                        new Coord( X: currentX + 1, currentY),
                        new Coord(currentX, Y: currentY + 1),
                        new Coord(currentX, Y: currentY - 1)
                };

        for (Coord pos : neighbours)
        {
            if (    0 <= pos.x && pos.x <= 4 &&
                    0 <= pos.y && pos.y <= 9 &&
                    board[pos.x][pos.y] == Piece.High)
            {
                return false;
            }
        }
    }
    return true;
}
```

Checking Obstruction

How to fix obstruction ?

- Thirdly,
- Place high rings

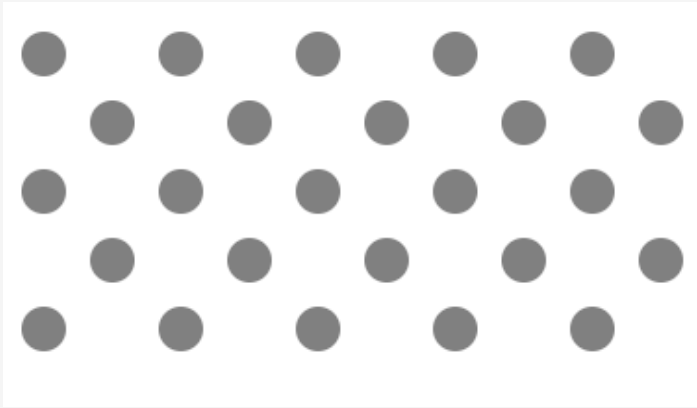
How to place rings ?

- Using “for loop” twice to put rings into the board.
- Using cellNumber of the piece ring to replace the number of the corresponding positions on the board (placed the ring successfully).


```
private boolean placeRing(int[][] currentShape, int ringType, Coord coord) {
    int x = coord.x;
    int y = coord.y;

    for (int i = -1; i <= 1; i++) {
        for (int j = -1; j <= 1; j++) {
            int cellNumber = currentShape[i + 1][j + 1];
            int currentX = x + i;
            int currentY = y + j;
            if (cellNumber == ringType)
            {
                if (    0 <= currentX && currentX <= 4 &&
                        0 <= currentY && currentY <= 9)
                {
                    if (    board[currentX][currentY] == 0 &&
                            checkNeighbours(currentX, currentY, ringType))
                    {
                        board[currentX][currentY] = cellNumber;
                    }else
                    {
                        return false;
                    }
                }else
                {
                    return false;
                }
            }
        }
    }
    return true;
}
```

Make Board and load image



Loop

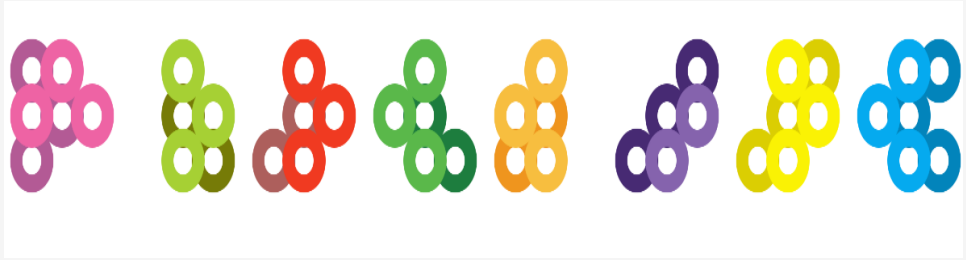
- Use for loop to make a board

By adding each circle to the Group

- Use for loop to load the image

(load half of them, the other are showed by flipping).

- Set position in Piece class



Make draggable piece

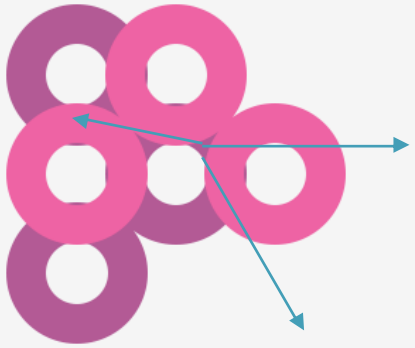
(get ideas from assignment 1)

Double click

How to interact

- Draggable class :set the property
- Extend from Piece :get location, size
- Handle event for each piece:

1. Rotate(scroll)
2. Flip(double click)
3. Draggable(press, drag)



Drag the picture

scroll

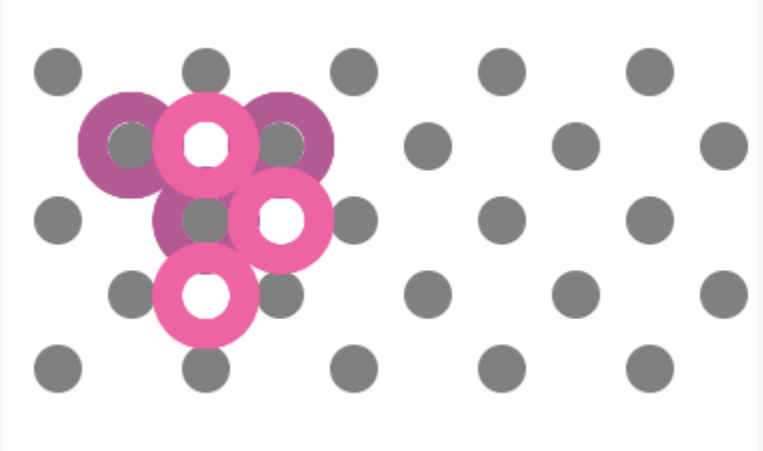
Method to justify

- On board
- Is valid

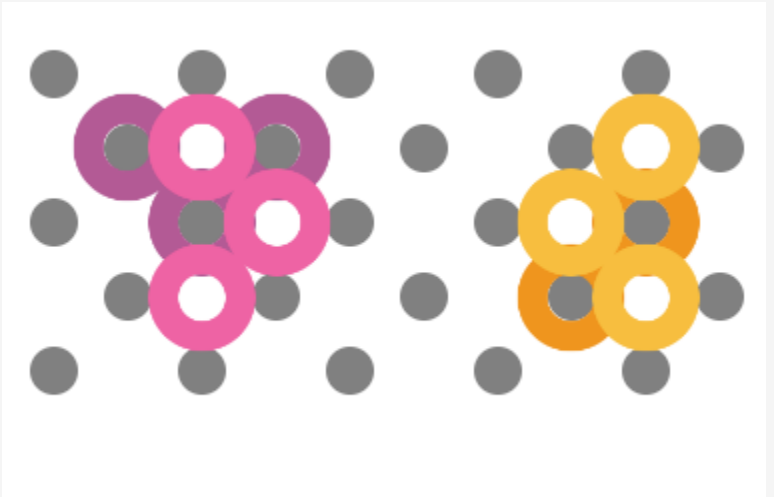
```
if(onBoard()){
    snapToGrid();
}
else {
    snapToHome();
}
if(StepsGame.isPlacementSequenceValid(Board.this.toString(onBoardPiece))){
    snapToGrid();
}else{
    snapToHome();
}
```

```
    private boolean onBoard(){
        return getLayoutX()>=(START_X-80)&&getLayoutX()<=(START_X+WIDTH+Space-Piece_Size)
                &&getLayoutY()>=(START_Y-80)&&getLayoutY()<=(START_Y+(Space*5+r)-Piece_Size);
```

Set position

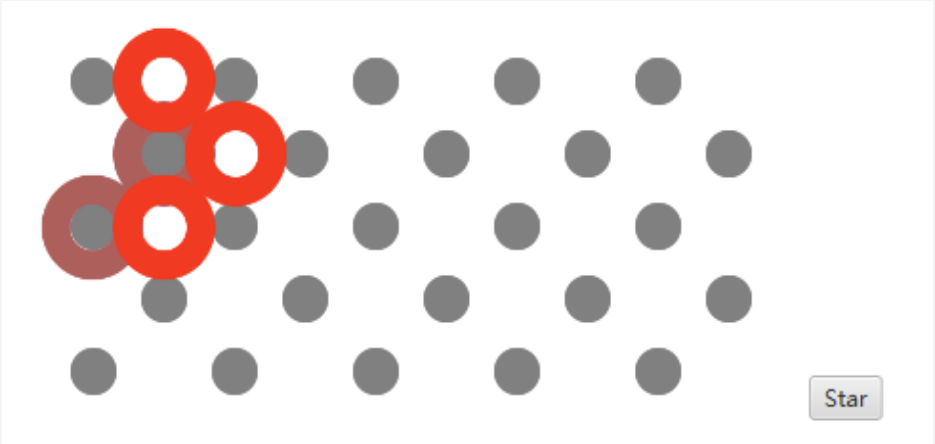


- Snap to the peg
- Not valid come back to original place



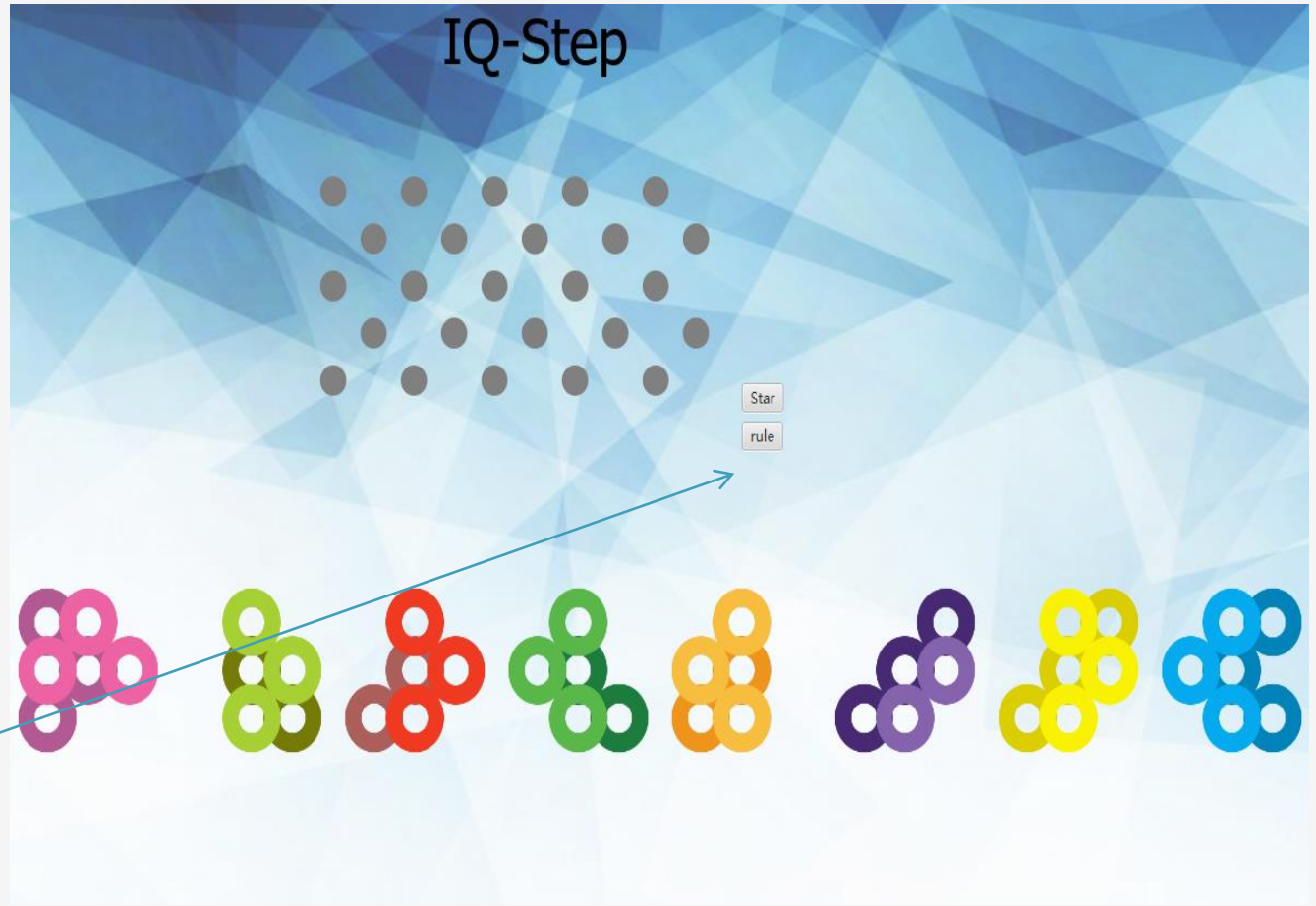
Start game

- Start game
- Select random piece on the board
- Click start button



Game Features

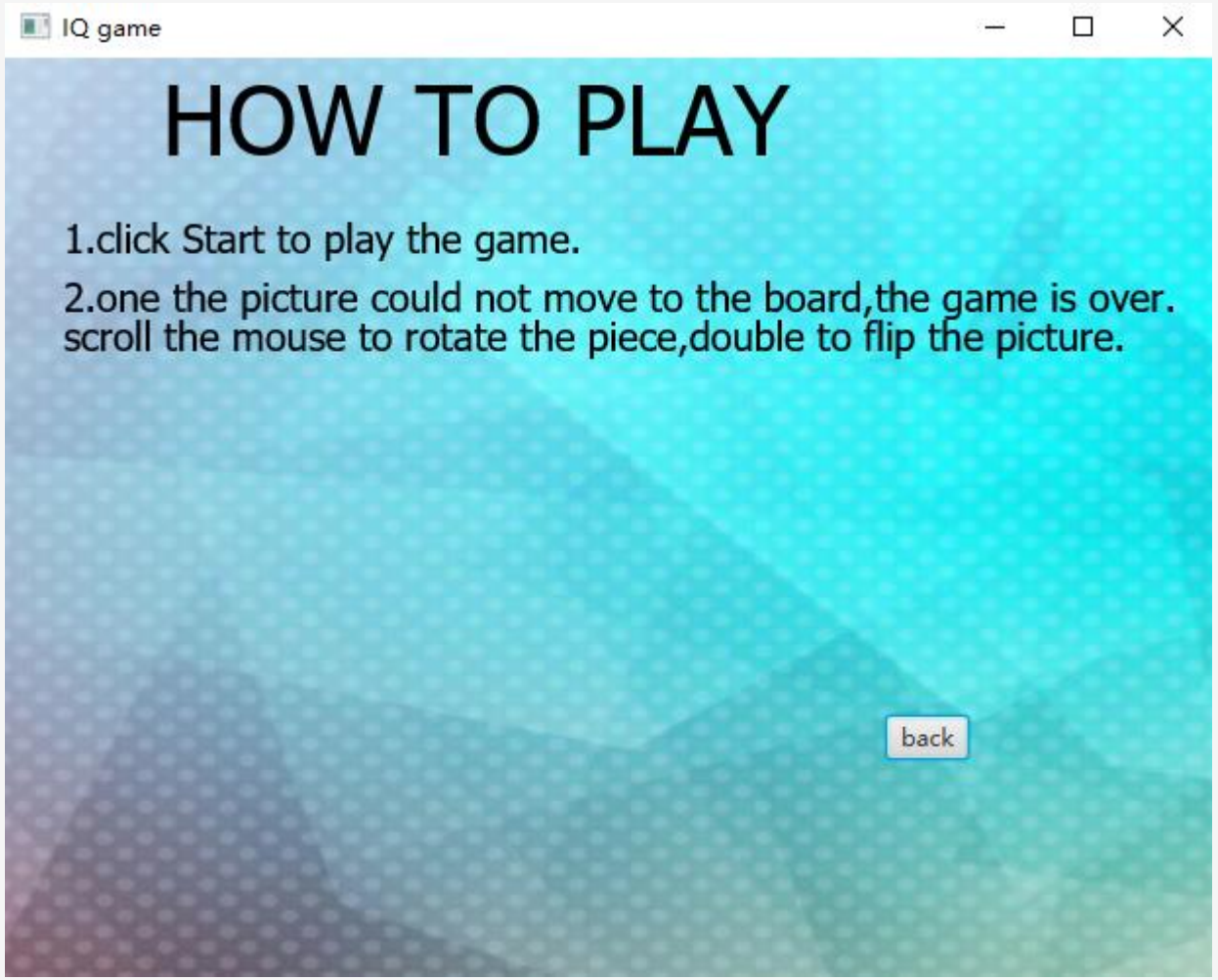
- Add title
- Add background
- Add rule button to switch the scene



Rule button

Game Features

- Add “back” button come back to game



THANKS

Chan Xu (u6233112)
Yiwen Peng (u6071714)
Qingsen Kuang(u5917277)