ASSIGNMENT 2
AZUL

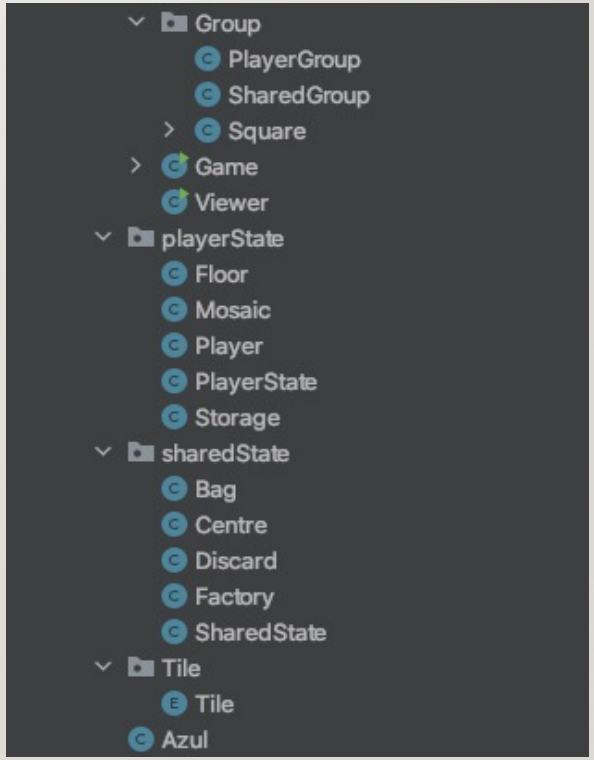
YUHUI PANG（U7211790）

JIAWEN WANG（U7111608）

QINLING ZHONG（U6616888）

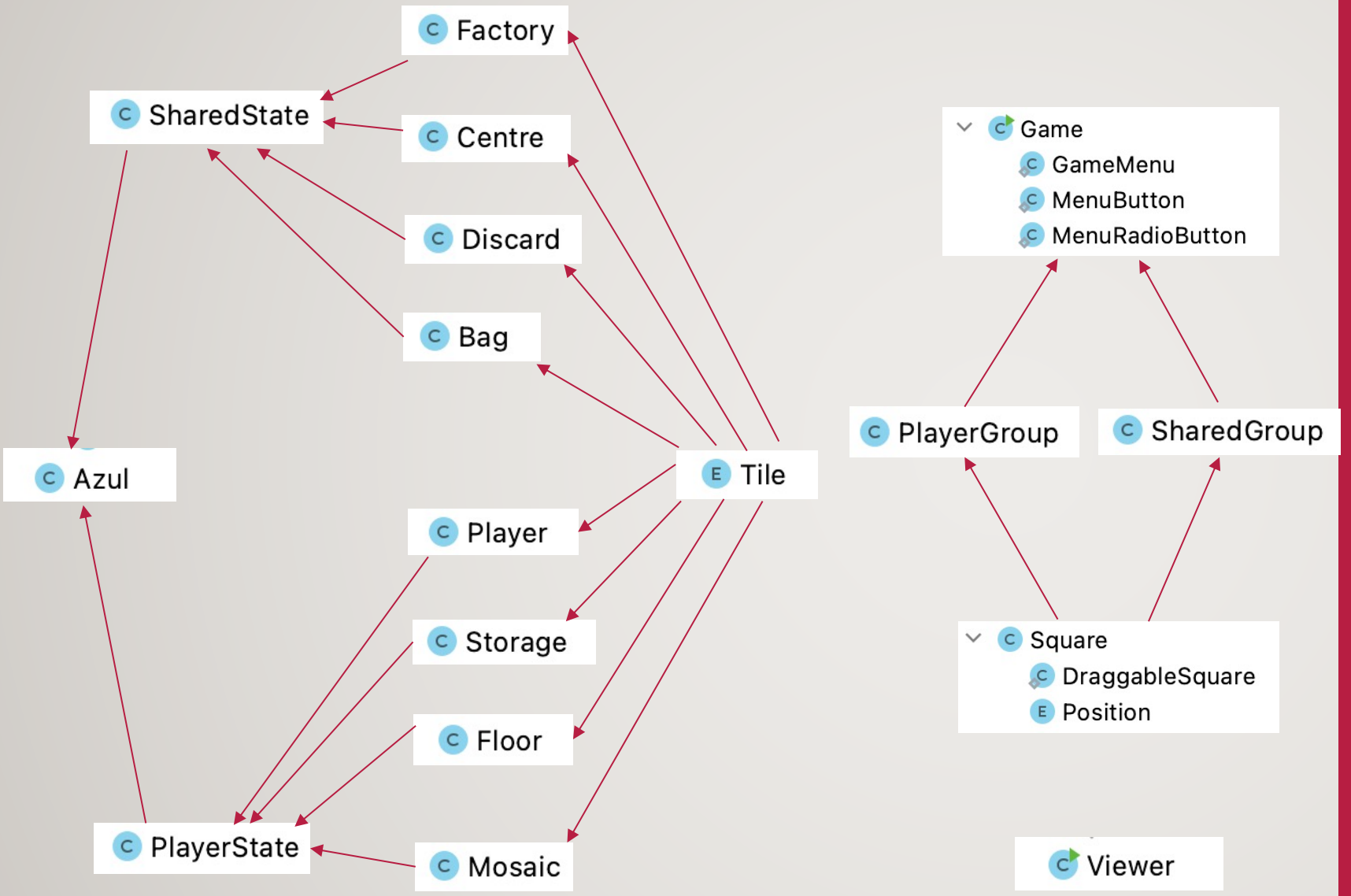
DESIGN

- Designing skeleton
- Building basic classes: constructors according to the string representation.
- Class PlayerState : combining classes Floor, Mosaic, Player and Storage, and implement a single player's player state.
- Class SharedState: combining classes Bag, Centre, Discard, Factory and implement the shared state.
- Enum Class Tile: stores Tile Id, Tile Type and Type colour.



DESIGN (CONT.)

- Playable Game: Game.java
- Class Square: draw tiles and implement dragable tiles, side and space of the squares according to the player numbers
- findNearest and highlightNearestSquare: find the nearest square and highlight the valid placements and the tiles that can be picked.
- Class PlayerGroup: draw player states according to the player
- Class SharedGroup: draw shared states according to the player number.
- Game menu, buttons, event handlers, backgroud image and bgm, sound effects.
- Variant mosaic and beginner mosaic.


Factory
SharedState
Centre
Discard
Bag
Azul
Tile
Player
Storage
Floor
PlayerState
Mosaic
Game
GameMenu
MenuButton
MenuRadioButton
PlayerGroup
SharedGroup
Square
DraggableSquare
Position
Viewer

DIAGRAM

```java
/**
 * @author Yuhui Pang
 */
public class Square extends Rectangle {

    /**
     * @author Yuhui Pang
     * the enum class that probably position the square maybe in
     */
    public enum Position{
        Bag,Centre,Discard,Factory,Floor,Mosaic,Storage;
    }
    public static double SIZE = 40;
    public static double SPACE = 5;
    public final Tile tile;
    public Position position;
    public Group group;
    public int row;
    public int col;
```

```java
/**
 * @author Yuhui Pang
 * the constructor of Square
 * @param x the x coordinate
 * @param y the y coordinate
 * @param tile the tile be represented
 * @param position current tile in which position
 * @param group current tile in which group
 * @param row the row of current tile (may not used, will equals -1)
 * @param col the col of current tile (may not used, will equals -1)
 */
public Square(double x, double y, Tile tile, Position position, Group group, int row, int col) {
    this.tile = tile;
    this.position = position;
    this.group = group;
    this.row = row;
    this.col = col;
    this.setX(x);
    this.setY(y);
    this.setWidth(SIZE);
    this.setHeight(SIZE);
    if (tile == null) {
        this.setFill(Color.GREY);
    } else {
        this.setFill(tile.getTILE_COLOR());
    }
}
```

CLASS SQUARE

CLASS SHAREDGROUP

```java
public class SharedState {
    /**
     * Field bag: The Bag in the SharedState
     * Field centre: The Centre in the SharedState
     * Field discard: The Discard in the SharedState
     * Field factory: The Factory in the SharedState
     * Field player: The current Player
     */
    private Bag bag;
    private Centre centre;
    private Discard discard;
    private Factory factory;
    private char player;
    public static int playerNum = 2;
    public static int factoryNum = 5;
```

```java
/**
 * @author Yuhui Pang
 */
public class PlayerGroup extends Group {

    private final double side = Square.SIZE;
    private final double space = Square.SPACE;
    private final Group player;
    private final Group storage;
    private final Group mosaic;
    private final Group floor;
```

```java
if(playerChar == 'B' || playerChar == 'D')   this.setLayoutX(600);
if(playerChar == 'A' || playerChar == 'B')   this.setLayoutY(200);
else if(playerChar == 'C' || playerChar == 'D') this.setLayoutY(480);
this.getChildren().addAll(storage,mosaic,player,floor);
```

CLASS PLAYERGROUP

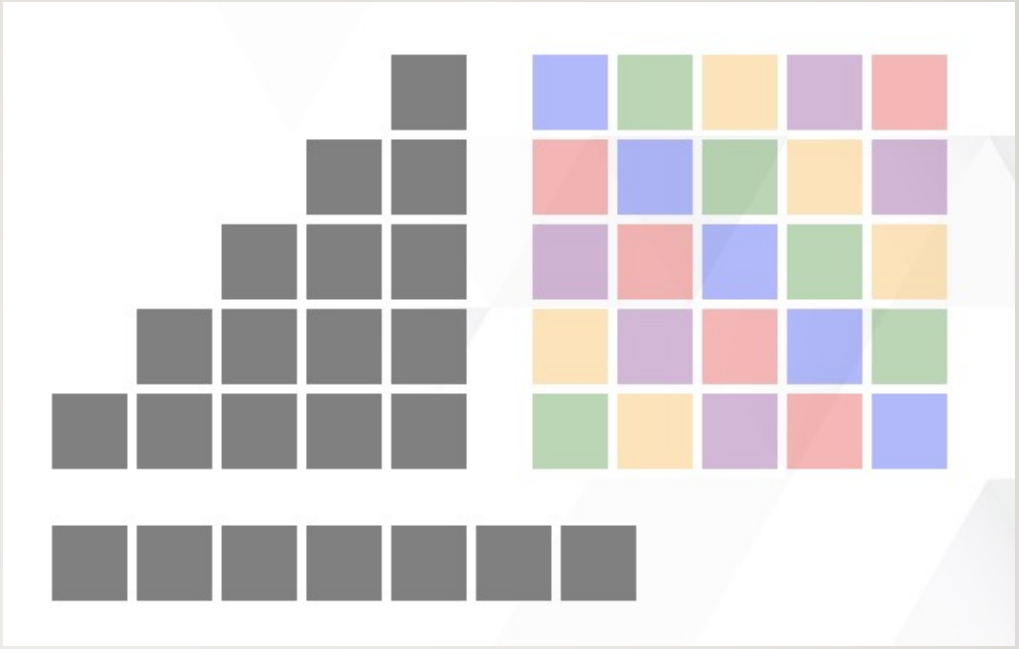
DESIGN (CONT.)

- generateAction: method for AI move
- getAllValidMoves: get all moves that can be performed according to the current game state.
- draftingHeuristic: evaluate the current game state according to the storage of the player
- Algorithm: first find all valid moves.
- If are drafting moves, return the move that will result in the highest heuristic after applying the move. If are tilling moves, return the move that will result in the highest player score after applying the move.

GENERATE ACTION

```java
public static String generateAction(String[] gameState) {
    // FIXME Task 13
    ArrayList<String> allMoves = getAllValidMoves(gameState);
    //Random rand = new Random();
    //int randomIndex = rand.nextInt(allMoves.size());
    //return allMoves.get(randomIndex);
    // FIXME Task 15 Implement a "smart" generateAction()
    char player = gameState[0].charAt(0);
    int playerNum = PlayerState.getPlayNumber(gameState[1]);
    sharedState = new SharedState(gameState[0],playerNum);
    String bestMove = allMoves.get(0);
    boolean isTillingMove = sharedState.getFactory().isEmpty() && sharedState.getCentre().isEmpty();
    for (int i = 1; i < allMoves.size(); i++) {
        String[] oldNextState = applyMove(gameState, bestMove);
        String[] newNextState = applyMove(gameState, allMoves.get(i));
        // tilling move: find the move with the highest player score
        if (isTillingMove) {
            int oldBestScore = PlayerState.getAllPlayerStates(oldNextState[1])[player - 'A'].getPlayer().getScore();
            int newNextScore = PlayerState.getAllPlayerStates(newNextState[1])[player - 'A'].getPlayer().getScore();
            if (oldBestScore < newNextScore) {
                bestMove = allMoves.get(i);
            }
        // drafting move: find the move with the highest heuristic
        } else {
            int oldBestHeuristic = draftingHeuristic(oldNextState, player);
            int newNextHeuristic = draftingHeuristic(newNextState, player);
            if (oldBestHeuristic < newNextHeuristic) {
                bestMove = allMoves.get(i);
            }
        }
    }
    return bestMove;
}
```

OVERVIEW OF THE GAME



- Single player state
- Storge(left) stores tiles, when it fills up the line performs tilling move.
- Mosaic(Right) contains 5 color's tiles, each lines color cannot be same, each column and row cannot have the same colour of tiles

SOLUTION (PART)

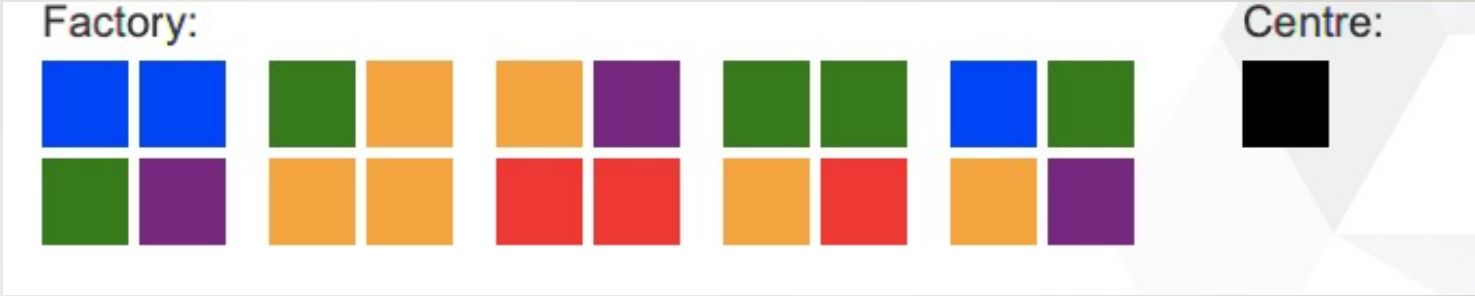
```java
public Storage(String stateStr){
    tileType = new Tile[NUMBER_ROWS];
    tileNumber = new int[NUMBER_ROWS];
    for(int i = 1; i < stateStr.length(); i += 3){
        int row = stateStr.charAt(i) - '0';
        char ch = stateStr.charAt(i+1);
        Tile tile = Tile.CharToTile(ch);
        int num = stateStr.charAt(i + 2) - '0';
        if(num > row + 1) isValid = false;
        tileType[row] = tile;
        tileNumber[row] = num;
    }
}
```

- Initialize the tile types of the Storage
- Initialize the number of tiles of each tile types

```java
public static boolean isWellFormed(String storageStr){
    if (storageStr.length() > 16 || (storageStr.length() - 1) % 3 != 0) return false;
    for (int j = 1; j < storageStr.length(); j += 3) {
        String s = storageStr.substring(j, j + 3);
        if ((s.charAt(0) < '0' || s.charAt(0) > '4') || (s.charAt(1) < 'a' || s.charAt(0) > 'e')
                || (s.charAt(2) < '0' || s.charAt(0) > '5')) return false;
    }
    return true;
}
```

- Check whether the string representation is valid.

OVERVIEW OF THE GAME



- Factory: contains 4 tiles in each factories, player perform drafting move and pick tiles to the storage
- Centre: holds first player token, keeps the remaining tiles from factories once a player picked, player can also pick tiles from centre.

SOLUTION (PART)

```
// F1aabc2abbb4ddee
public Factory(String stateStr, int playerNum) {
    this.FACTORY_NUMBER = 2 * playerNum + 1;
    int count = 0;
    this.tiles = new Tile[FACTORY_NUMBER][FACTORY_CAPACITY];
    for (int i = 1; i < stateStr.length(); i += 5) {
        int row = stateStr.charAt(i) - '0';
        for (int j = i + 1; j < i + 5; j++) {
            int col = j - i - 1;
            char ch = stateStr.charAt(j);
            this.tiles[row][col] = Tile.CharToTile(ch);
            count++;
        }
    }
    totalNum = count;
}
```

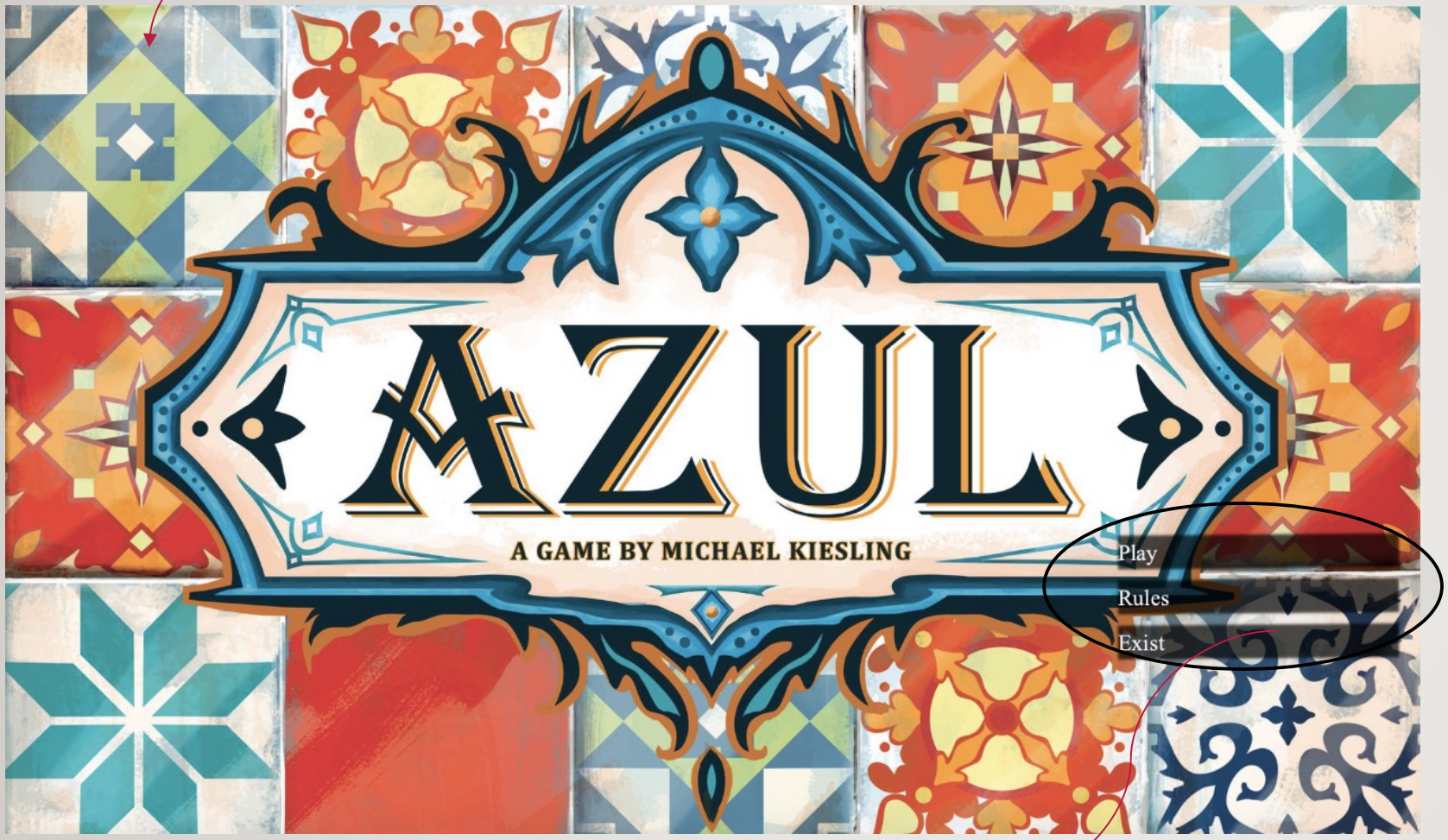
- Initialize the total number of tiles in Factory according to player number
- Initialize the tiles included in this factory valid placements

OVERVIEW OF THE GAME

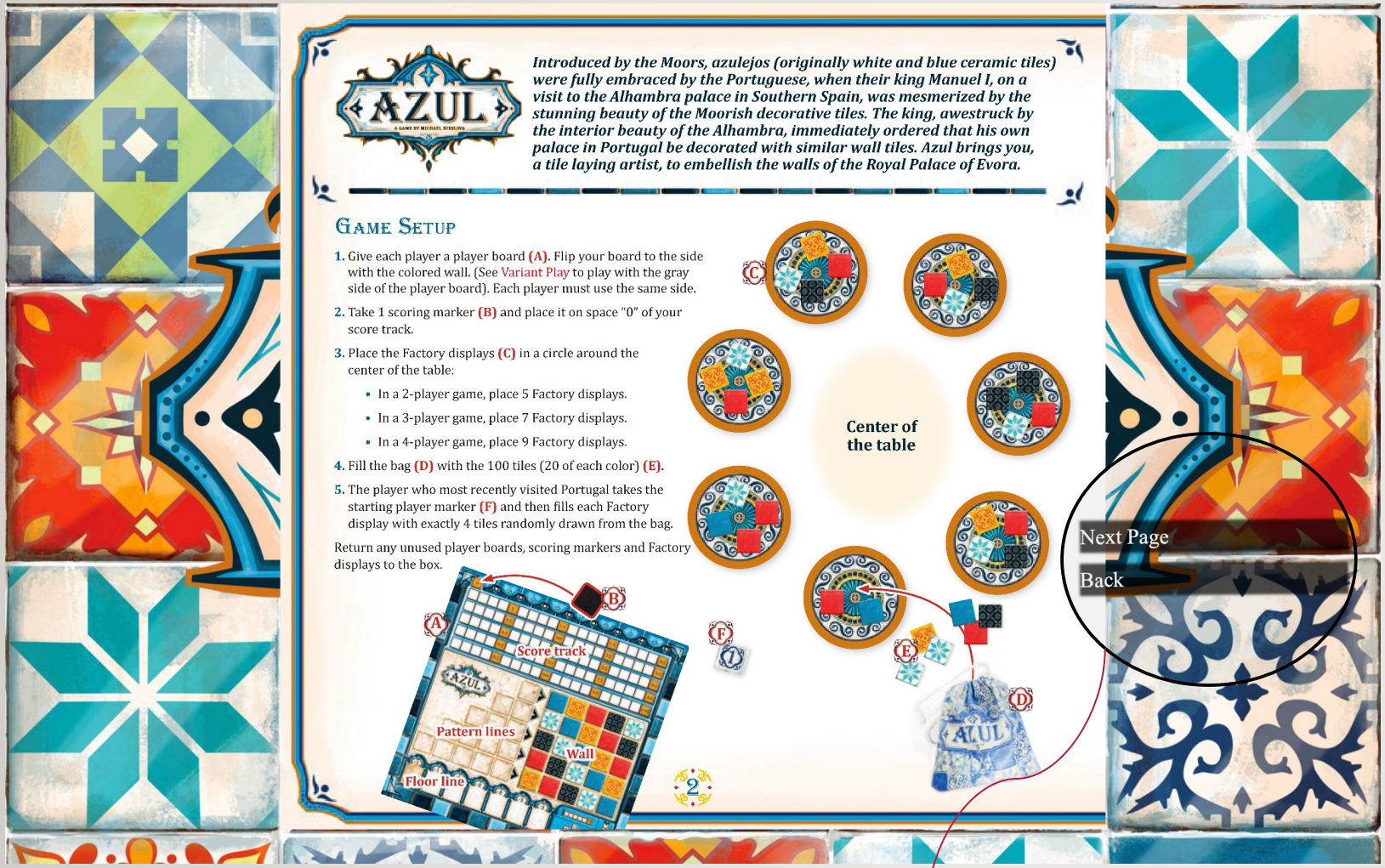
- Floor accepts redundant tiles from Factory and Centre
- Can hold tiles up to 7
- Redundant tiles in Floor will influence the mark

GAME MENU

Background



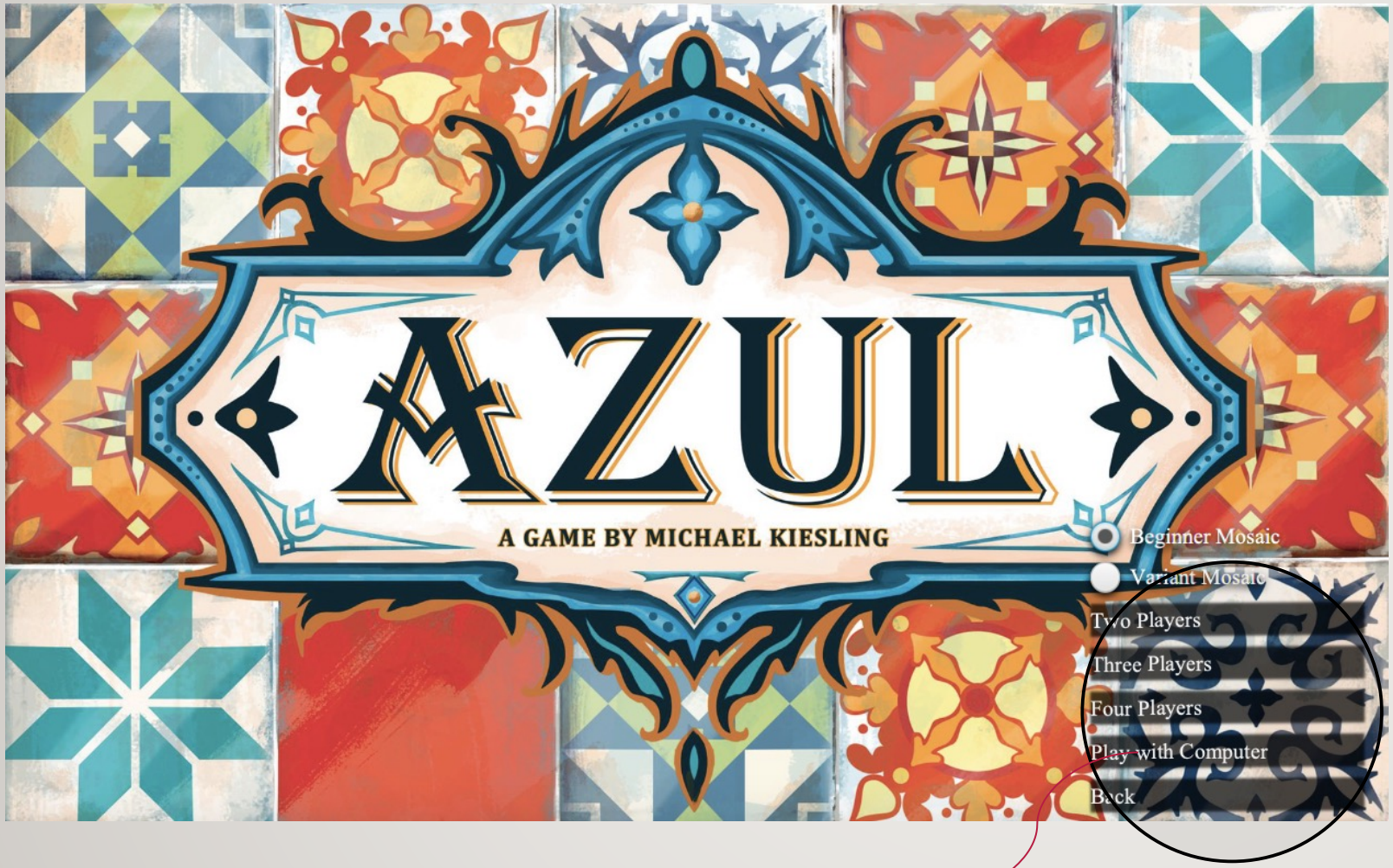
Pop-up Animation



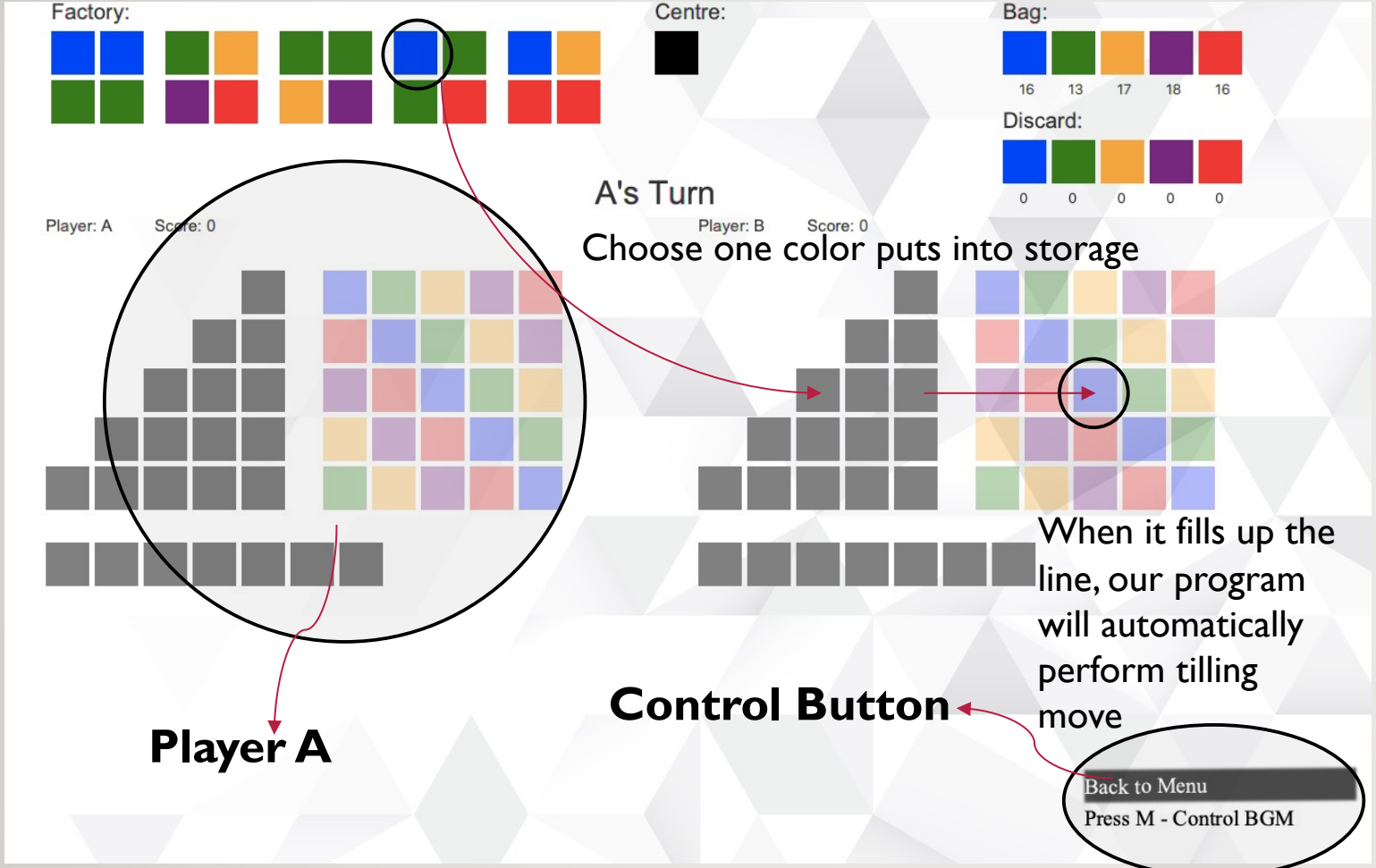
Allow users to view game rules page by page

GAME RULES

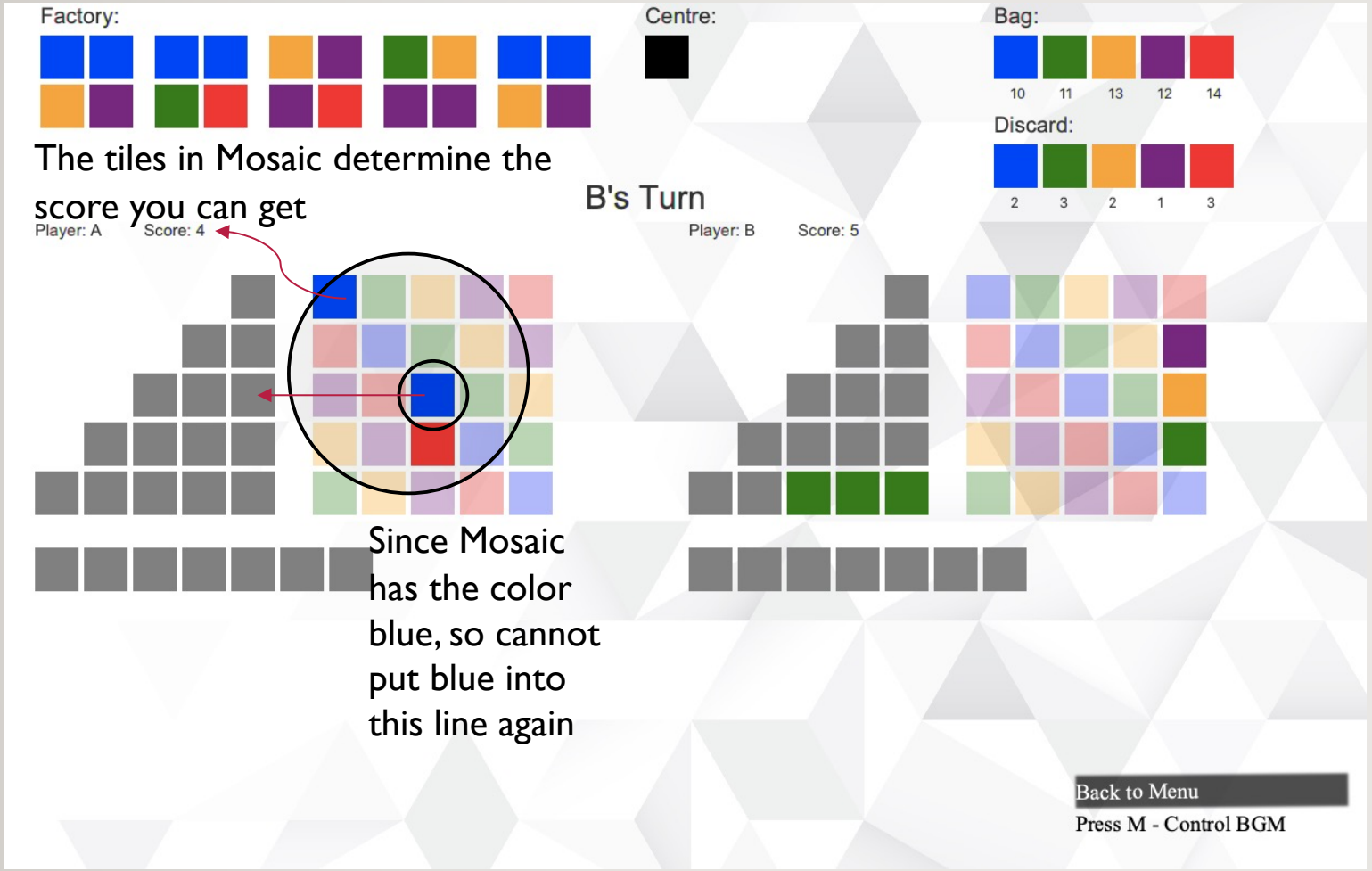
GAME STARTING



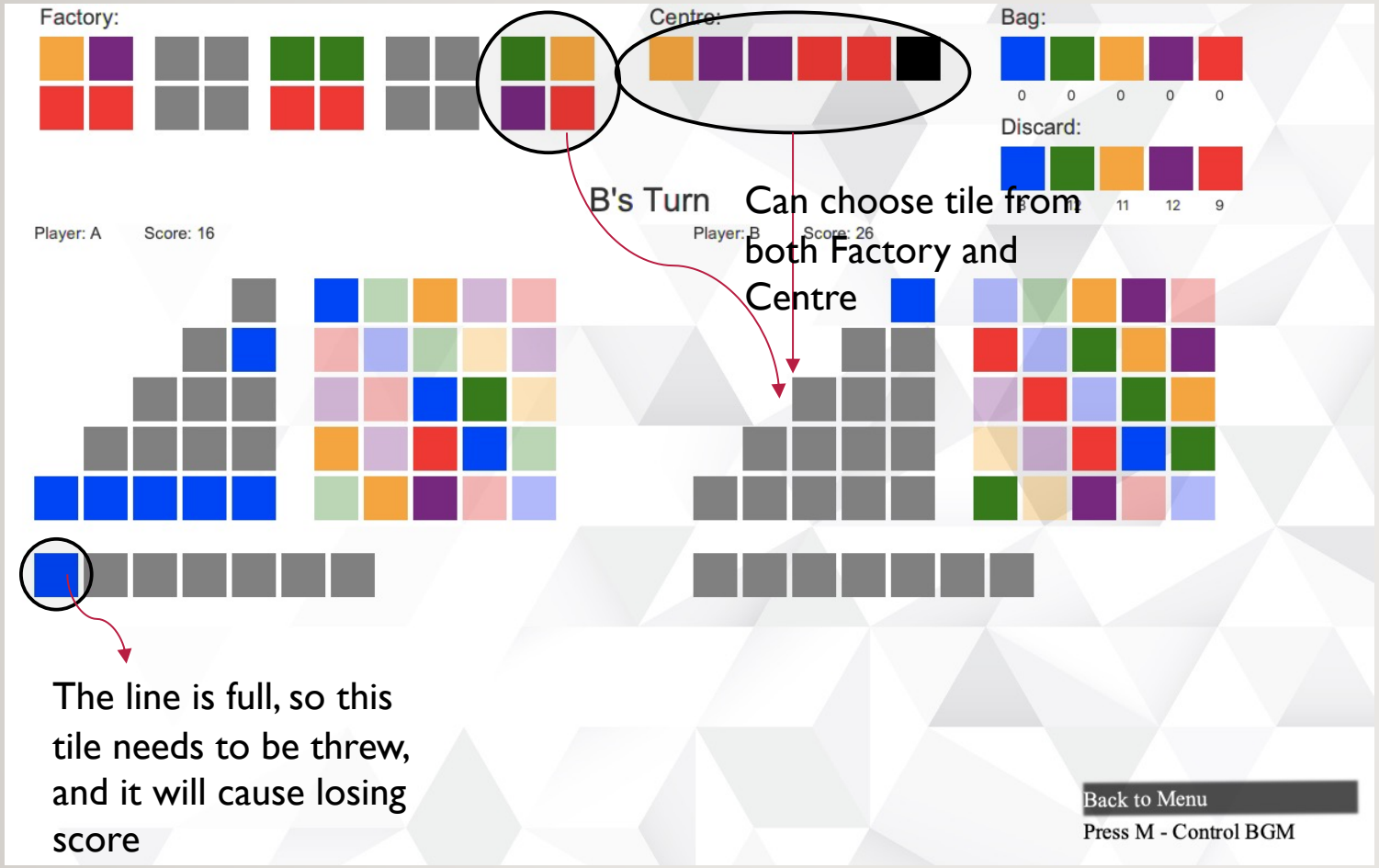
Pop-up Animation



GAME OF TWO BEGINNER MOSAIC


Factory:
Centre:
Bag:
10 11 13 12 14
Discard:
2 3 2 1 3
The tiles in Mosaic determine the score you can get
B's Turn
Player: A Score: 4
Player: B Score: 5
Since Mosaic has the color blue, so cannot put blue into this line again
Back to Menu
Press M - Control BGM

GAME START



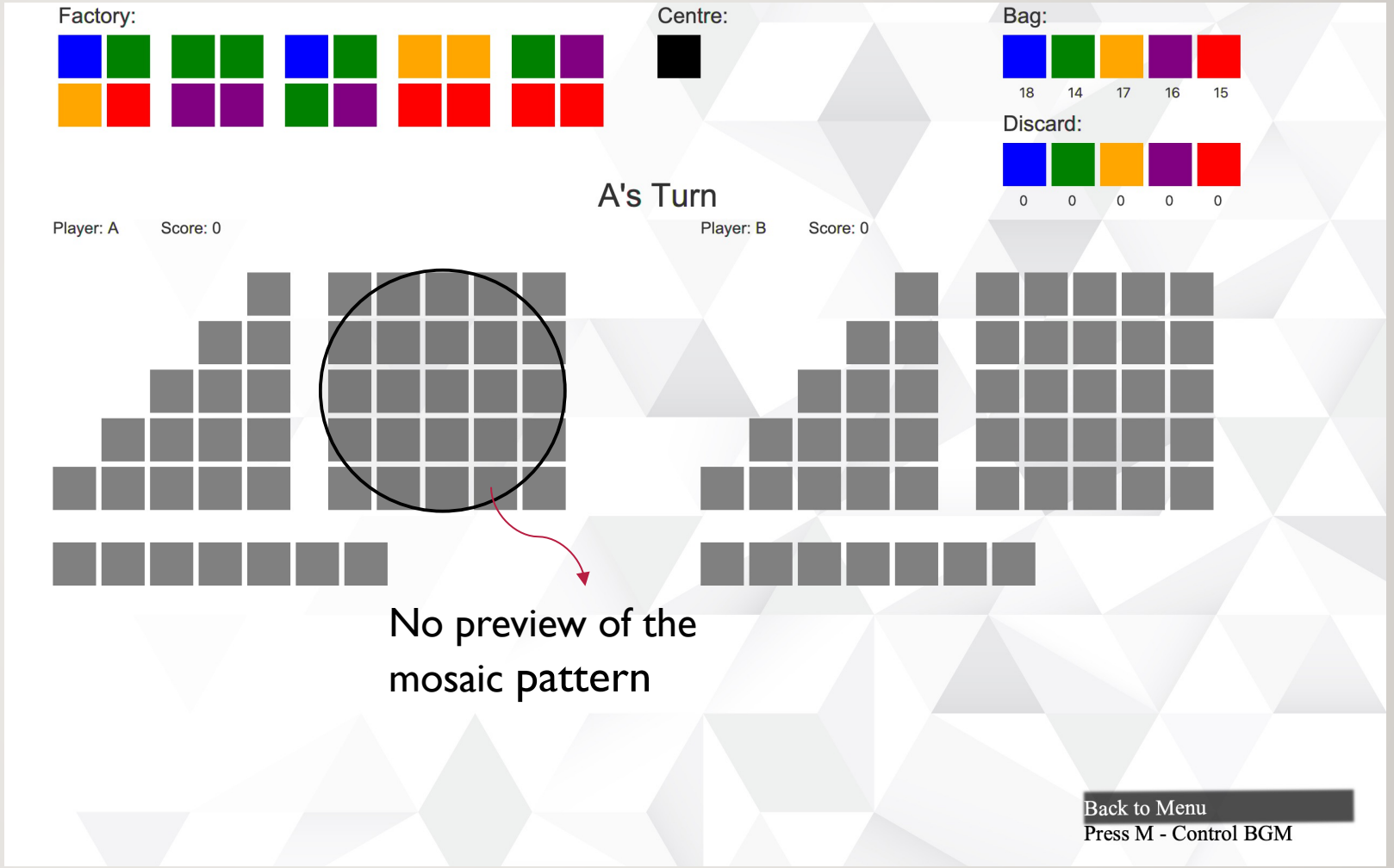
GAME PROCESS


Factory:
Centre:
Bag:
0 0 0 0 0
Discard:
13 10 13 14 9
The score can be updated automatically
A's Turn
Player: A
Score: 52
Player: B
Score: 27
Player A wins!
Press ESC - back to the game menu
Back to Menu
Press M - Control BGM

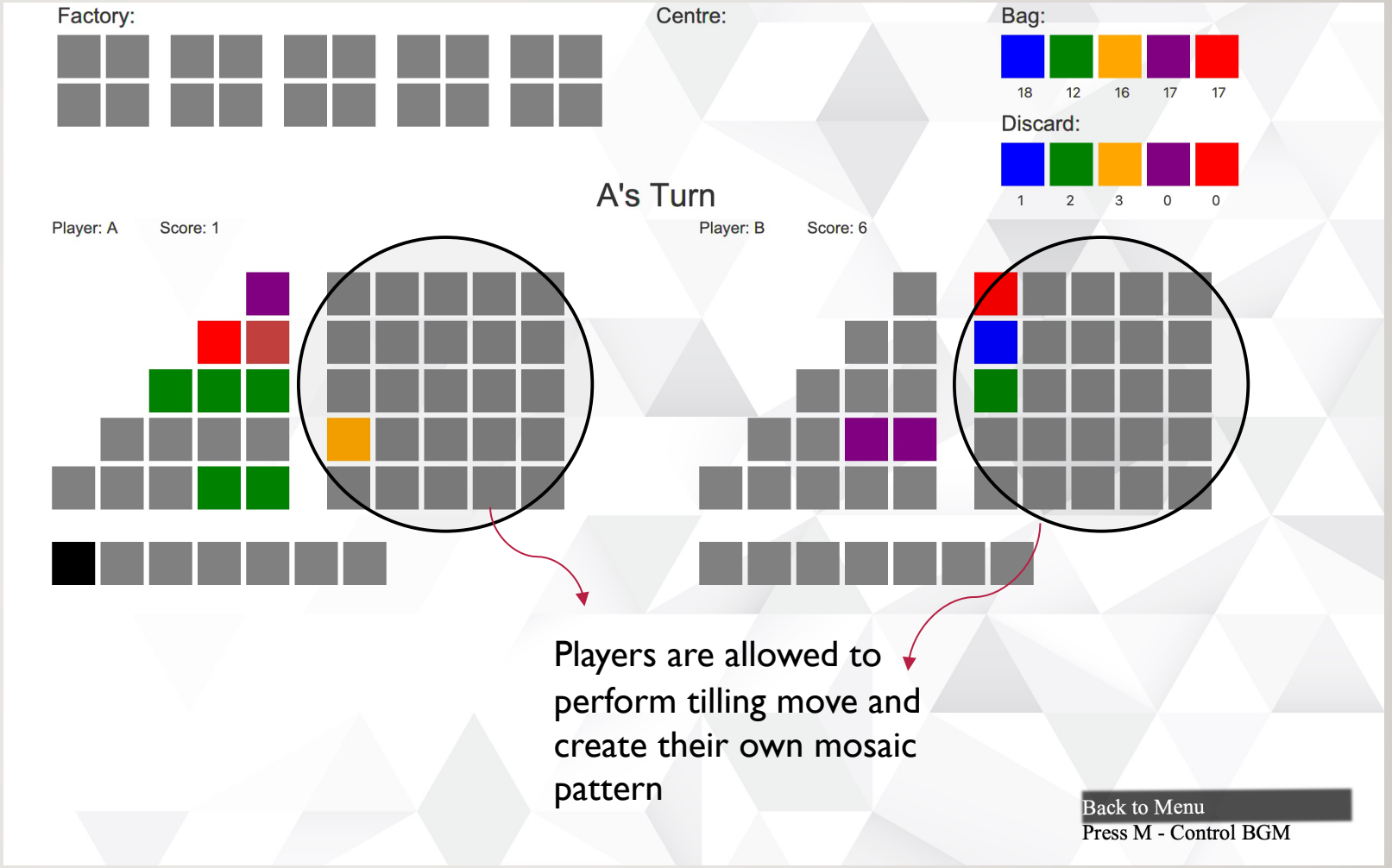
GAME END

VARIANT MOSAIC

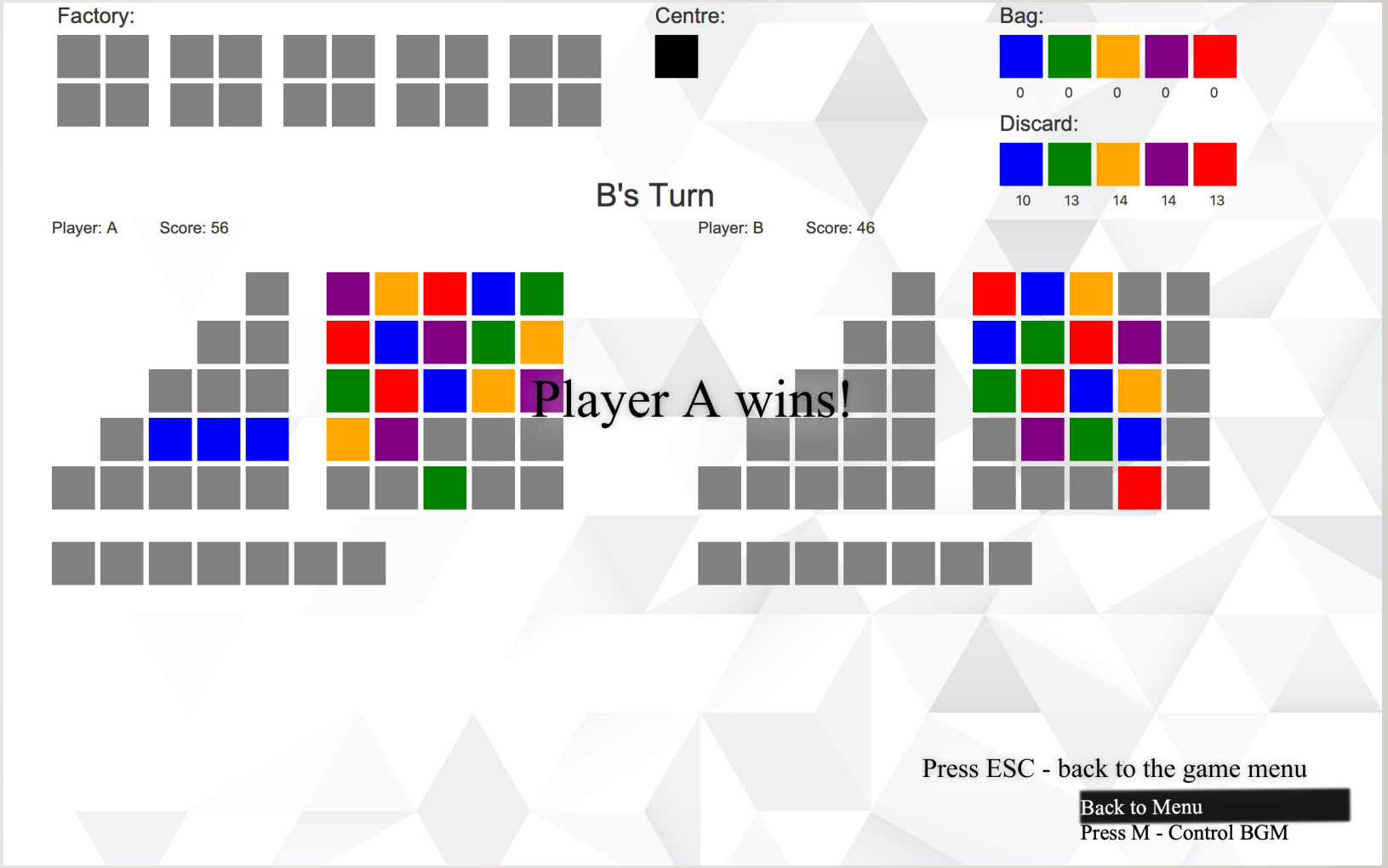
GAME START



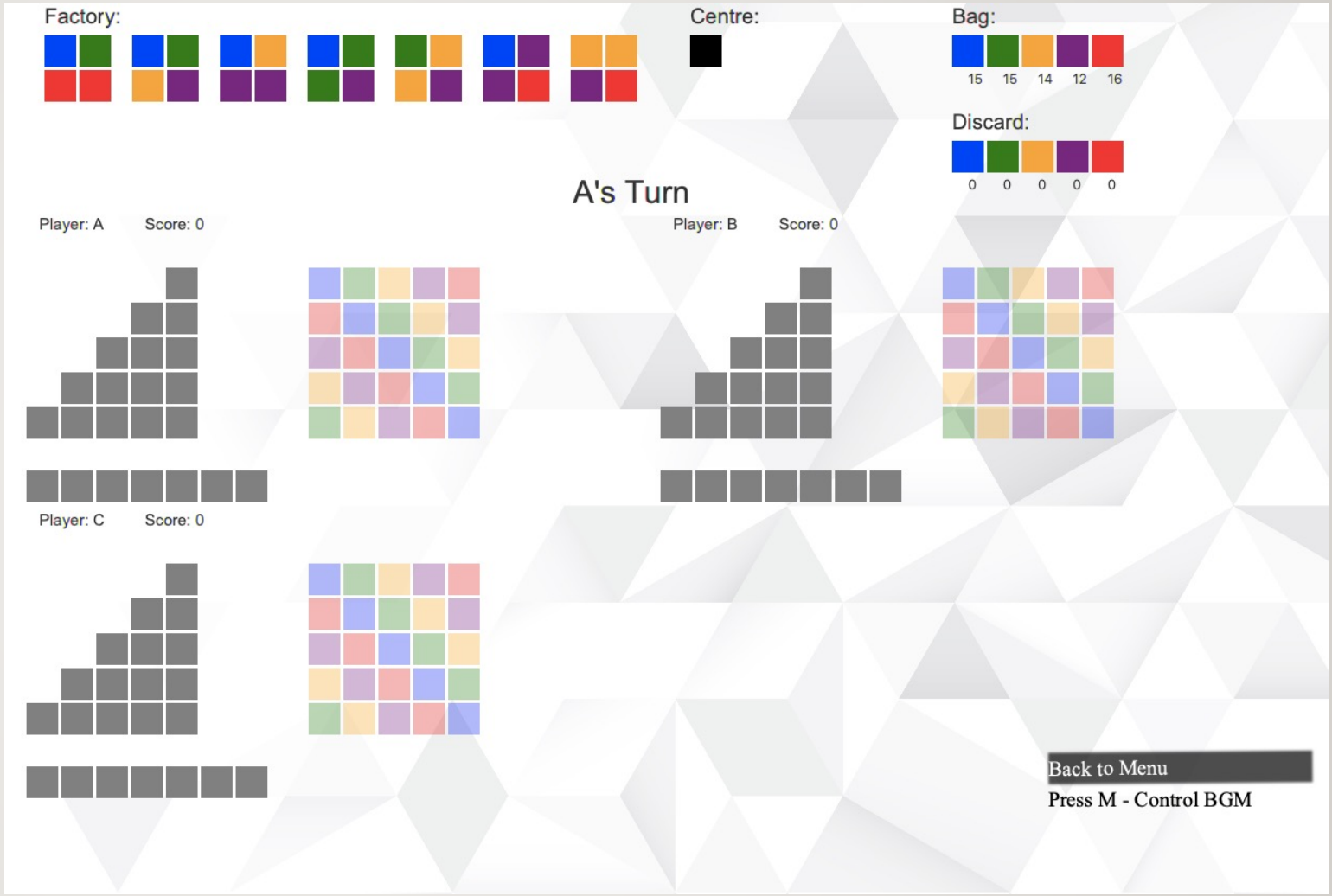
TILLING MOVE



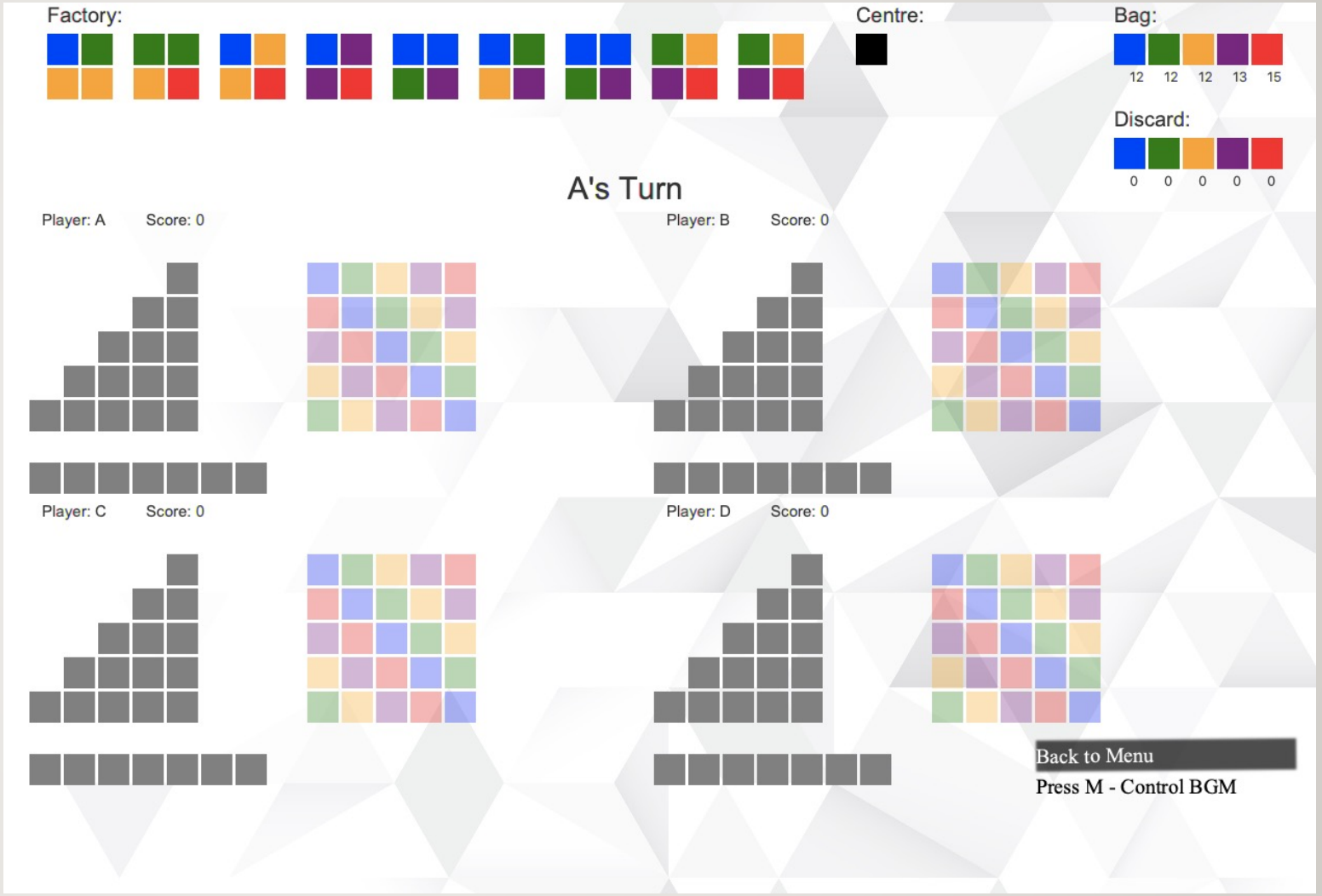
GAME END

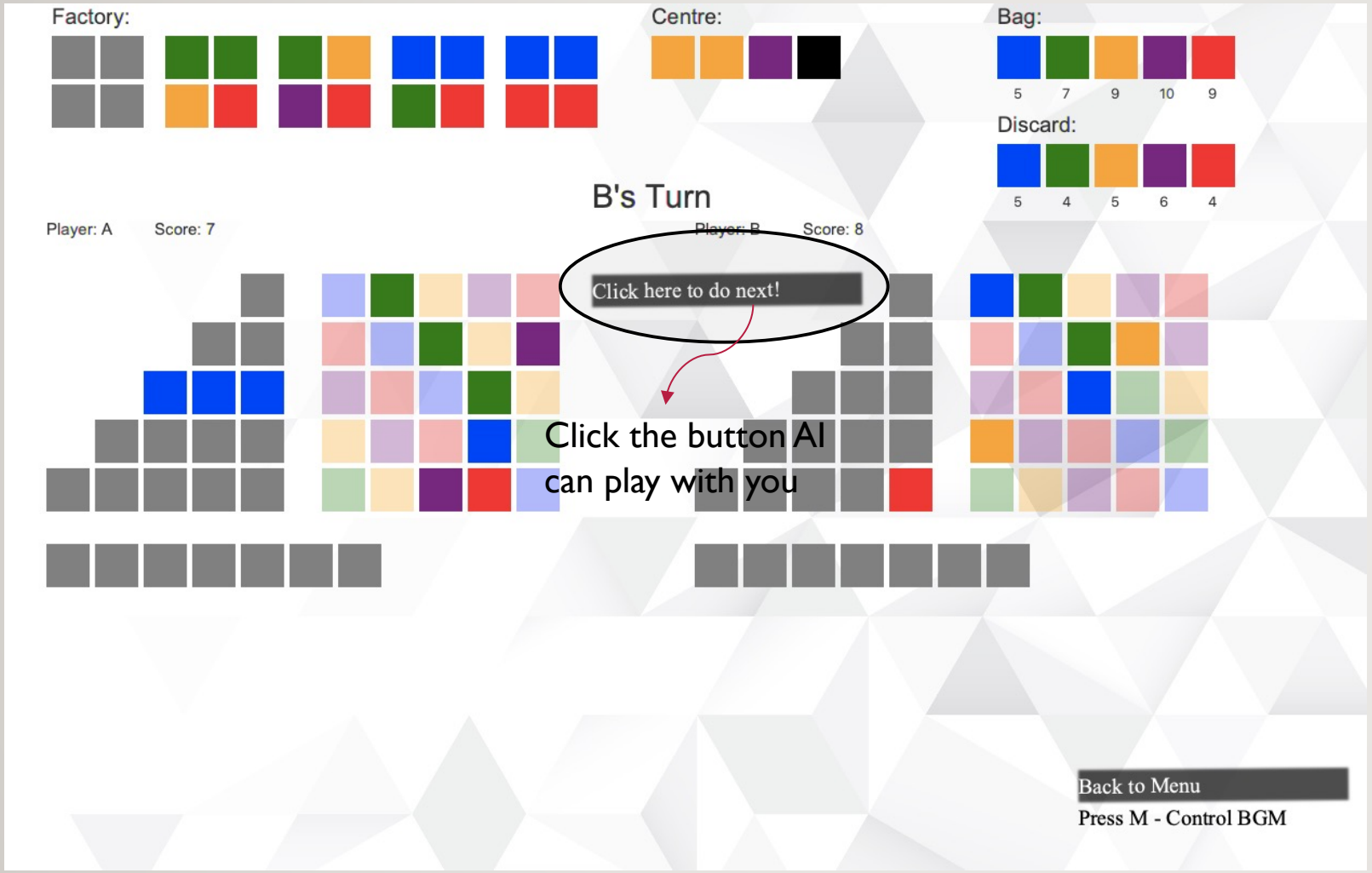

Factory:
Centre:
Bag:
0 0 0 0 0
Discard:
10 13 14 14 13
B's Turn
Player: A Score: 56
Player: B Score: 46
Player A wins!
Press ESC - back to the game menu
Back to Menu
Press M - Control BGM

GAME OF THREE



GAME OF FOUR


Factory:
Centre:
Bag:
12 12 12 13 15
Discard:
0 0 0 0 0
A's Turn
Player: A Score: 0
Player: B Score: 0
Player: C Score: 0
Player: D Score: 0
Back to Menu
Press M - Control BGM


Factory:
Centre:
Bag:
5 7 9 10 9
Discard:
5 4 5 6 4
B's Turn
Player: A Score: 7
Player: B Score: 8
Click here to do next!
Click the button AI can play with you
Back to Menu
Press M - Control BGM

PLAY WITH AI

Figure 1:https://www.123rf.com/

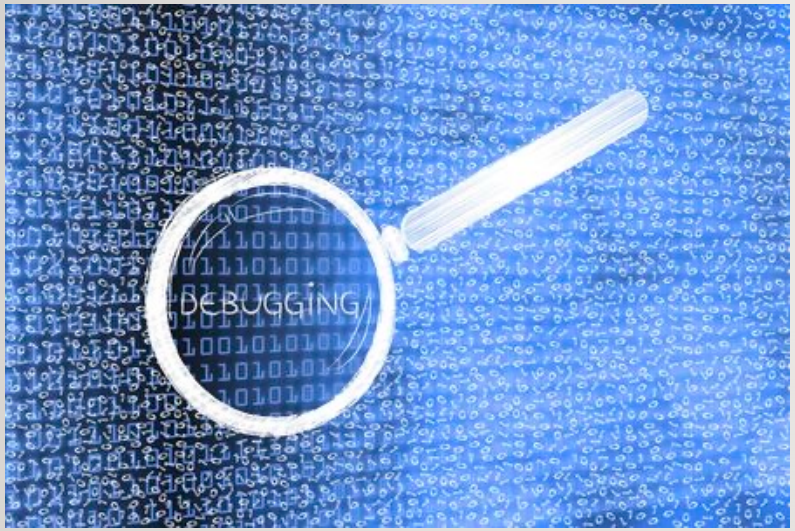


Figure 2:https://itchronicles.com/

ABOUT INSUFFICIENT

- We did not have sufficient time to clear up the code
- The AI is not as smart as we wish, we did not have time to study and apply algorithms such as Monte Carlo Tree Search and Minimax.

COOPERATE

DISCUSS

- Each time we need to complete a task we will communicate with each other and share ideas

WORKING

- We assign tasks to each other by considering person's ability
- Our cooperation is efficient and friendly

THANKS

YUHUI PANG（U7211790）

JIAWEN WANG（U7111608）

QINLING ZHONG（U6616888）