COMP 1110 Assignment -2 Presentation

Group-thu08j

Group-members:

Yuqing Zhang (u6767747),

Kalai(u6555407),

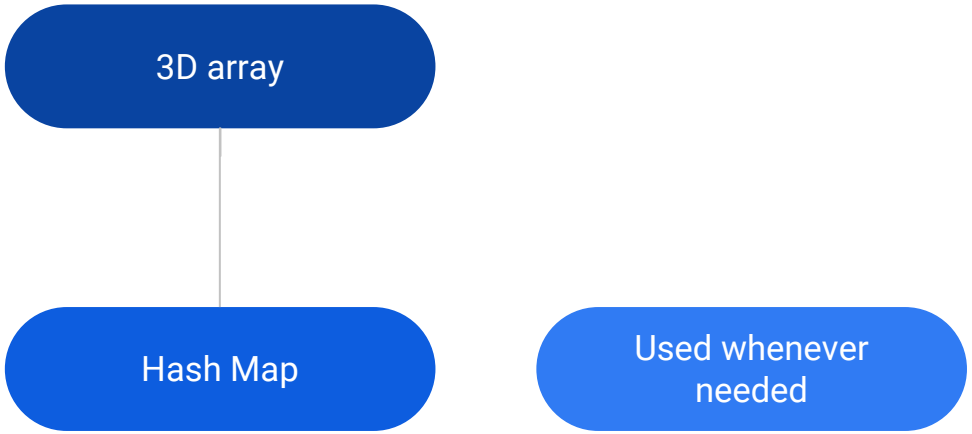
LingYu Xia(u6483756)

Introduction

- Our task is to convert a board game called IQ-twist into a computer program with the help of Java and JavaFX .
- Throughout the presentation we would like to discuss about the ways in which we incorporated the game logic in to a program using java , the building up of the user interface, the difficulties faced during the implementations ,the drawbacks and the things we could have improved.

Logic Of the Game

- Representation of Board - A Single 2-D String Array
- Representation of piece/peg - A once only initialised hashmap of just 60 elements(pieces & pegs)
- Data of the pieces - A Single 3-D String array



More about the Representation of Pieces

or	r	or
x	x	r

```
"C:\Program Files (x8
  12345678
A #r#xxxxb
B xxrxxxx#
C xxxxxxxb
D xxxxxxxb
```

og	x	x
og	g	g
x	og	x

This piece (a0) will have a
HashMap key number 0.
(a1) will have a key number of 1 & it goes like
this for the subsequent ones.

Part one — ensure the validity of the placement.

Task 5 (the backbone of placement validity)

Functions used

-placer:Places the piece(2d-array) into the board

-boardcreator - initialise the board with respective values

-rotator

-flipper

-checkBoard - checks the board for badPegs and if pieces Overlap or not.

Continuation

Overlap and Offboard are being checked as each piece is being placed .
-checkboard
check the length of each String element in the board;

check each char value of each String element;

i.e. checkboard function will make task 5 return false if any String element is in the pattern like “pgor”(badPegs) or “ry”(overlap);

Part one — building a Viewer

- ➔ Initial approach, we had used rows of lines to represent a board visually.
- ➔ Used a Gridpane ,which just required column and row indices and spans as arguments.
- ➔ Upon using setRotate over over few images(pieces) there were placed with an offset.

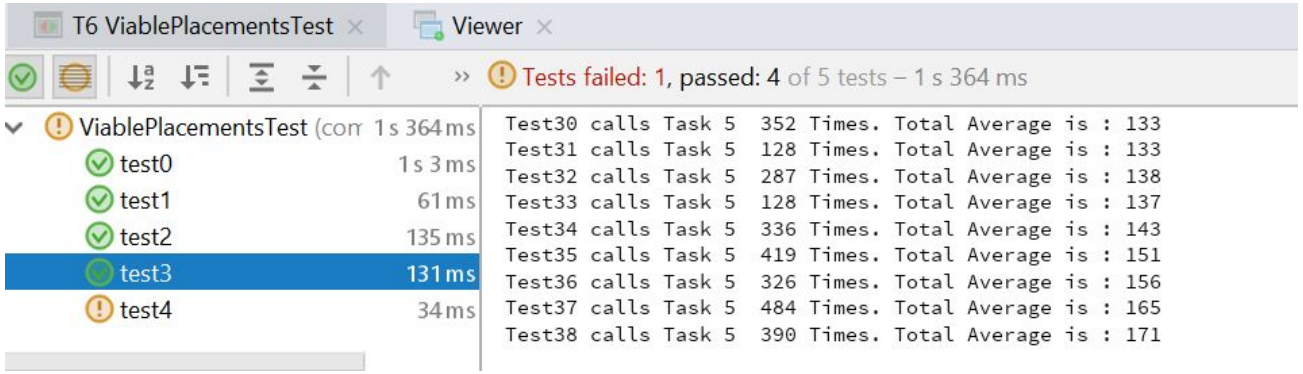
More on the Viewer

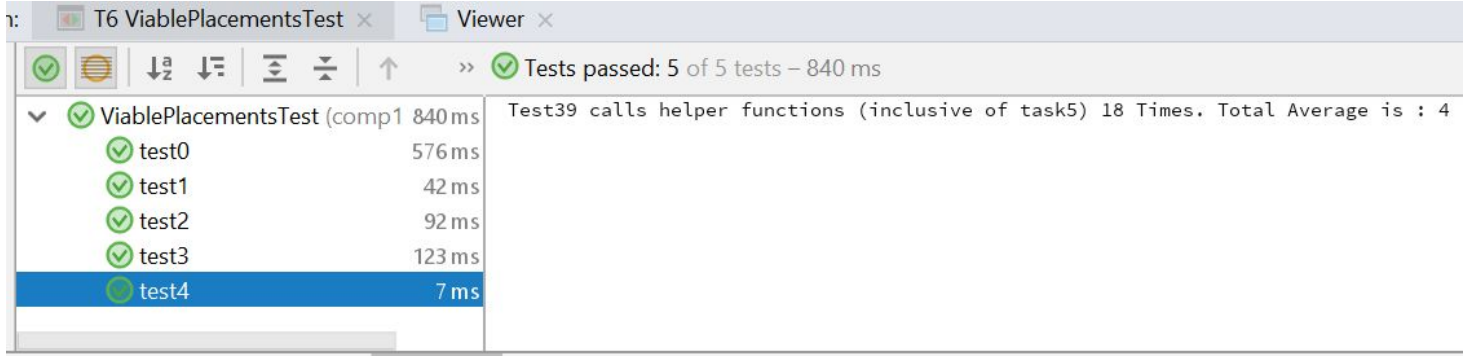
```
/*TRANSLATION DATA
   R90        R270
 a -45        -45
 b -45        -45
 c -140       -140
 d -45        -45
 e  0           0
 f -45        -45
 g 0            0
 h 0            0
   */
```

- setTranslation was used to fix the offsets produced by the images.
- The data about the translation values were manually found out for this task.

Part two — get viable piece placements

- Initially ,we had brute forced to create piece encodings and then ran Task 5 several times for validity check.These were the average number of calls it made to Task 5.





- We had managed to reduce the number of function calls to a smaller number by using the current codes.
- What does it do?

1)Find the the range of indices of the empty Grids

2)Uses different containers - Contains the keys for the pieces HashMap

3)These container will updated and reduced accordingly.

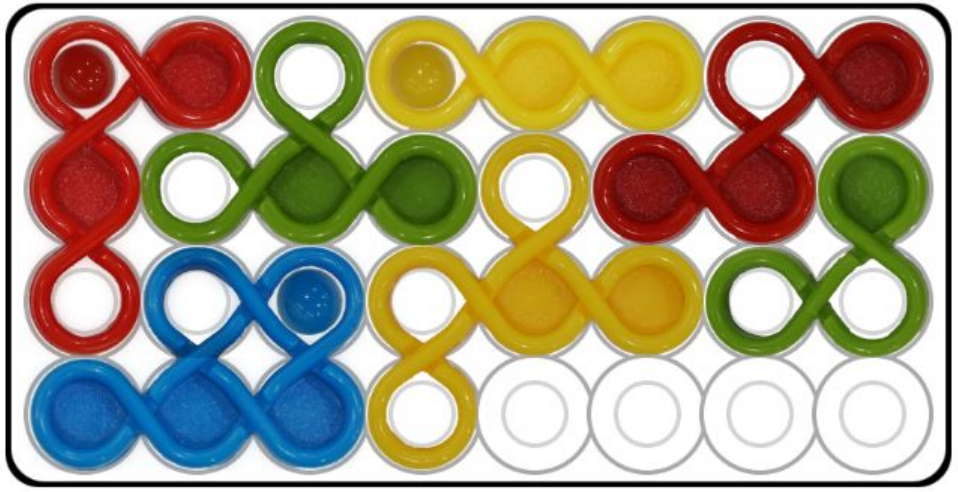
```
Containers

0)        1)        2)        3)      4)
 X X       X X X     X X X     X       X X X X
 X X       X X X     X X X     X
 X X                 X X X     X
                               X
```

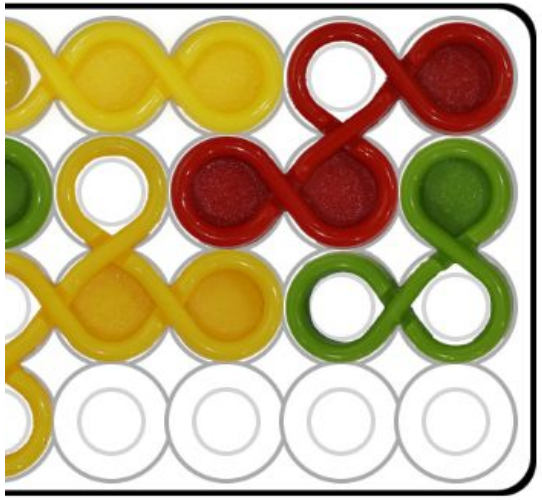
```
ppContainer[0][]=[0,2,4,6,8,10,12,14,24,26,28,30,(32 to 39 inclusive),40,42,44,46,56,58]
ppContainer[1][]=[1,3,5,7,9,11,13,15,25,27,29,31,(32 to 39 inclusive),41,43,45,47,57,59]
ppContainer[2][]=[48 to 55]
ppContainer[3][]=[17,19]
ppContainer[4][]=[16,18]
```

4)Then the container will be masked over the board , and then a piece encoding is produced accordingly and task 5 is called.

Example:Uses container 3 and 4 and masks over board and get piece encoding.

It tries each encoding with task 5 codes, if it is valid then adds it to the set(output).





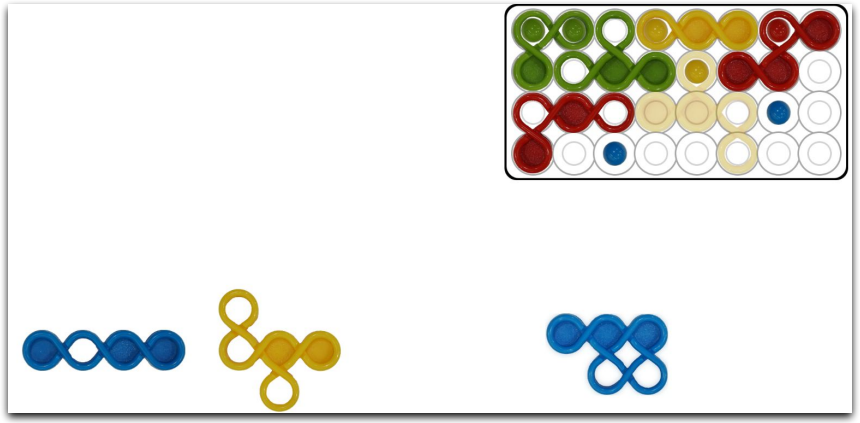
Part two — finding all solutions from start

- 1)getFormalPieces(): divide the placement String into 4-character piece Strings;
- 2)combination(): combine all the different possible pieces to form a solution;
- 3)reorderPlacementString(): reorder the pieces in the right alphabet order;
- Difficulties & Improvements: too many combinations stored in an arraylist causing OutOfMemoryException.Then we check whether the combination is valid first then add it to the list.

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/java
initial placement: c1A3d2A6e2C3f3C2g4A7h6D0j2B0j1C0k3C0l4B0l5C0
solution 0: a6A0b6B0c1A3d2A6e2C3f3C2g4A7h6D0
solution 1: a7A7b6A5c1A3d2A6e2C3f3C2g4A7h6D0

Process finished with exit code 0
```

Part two — Implementation of hints

- gameState- A string which holds the current game placement values.
- Based on the gameState and the Solutions which were already created Hints are given.
- We made sure that the hints were dispersed randomly and were different at each time the user presses ”/”.
- Instead of using the Random class -- Collection.Shuffle was used.

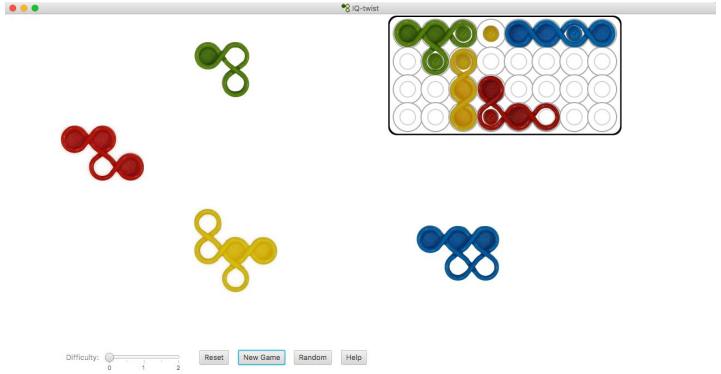


Part two — interesting starting placements

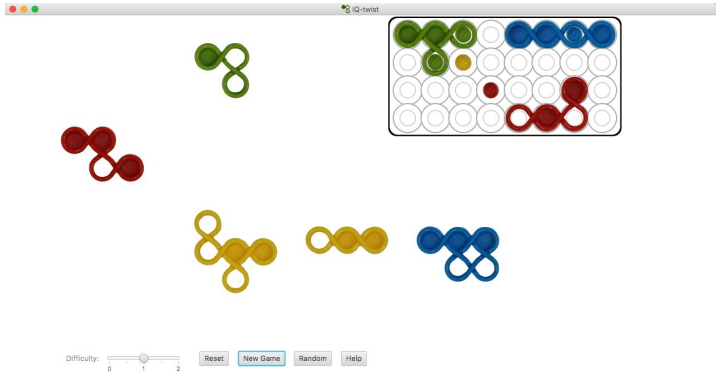
- 1)manually find four solutions that are valid;
- 2)use pieceCreator() to randomly select one of the strings, and randomly choose a 2-pieces string from it;
- 3)implement getSolutions() to find all possible solutions based on the string above;
- 4)choose randomly number of pieces from the list based on the difficulty levels the users choose to form the starting placements’
- Drawbacks: we don’t consider all the possible starting placements, it’s not thoughtful and easy to find a pattern.

Part two — interesting starting placements

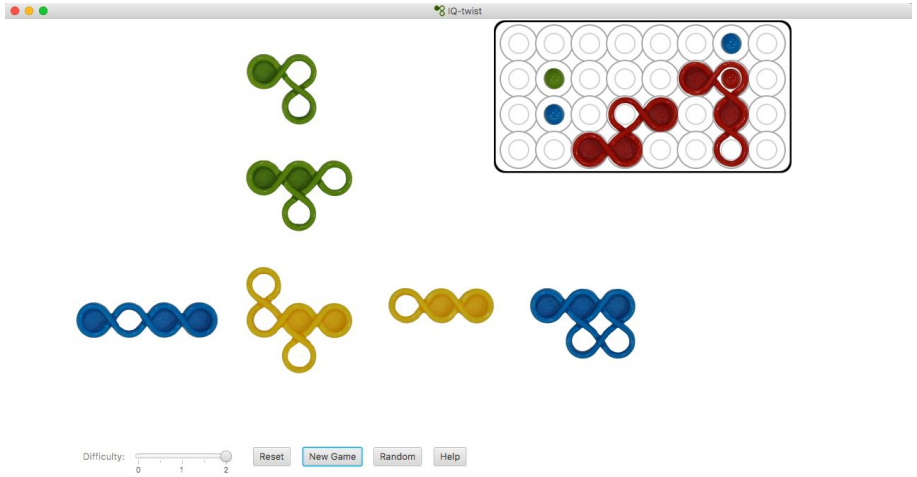
Level 0:



Level 1:



Level 2:



Part two — basic working Twist game

- 1)We build a GUI and add a game board, pieces and several button;
- 2)Linked the previous function to this GUI;
- 3)Implemented Inheritance (References from Assignment 1 Javafx codes)
  - Inner class Piece
  - within the inner class there is another class called eventPiece(which handled the draggable interface)

DrawBacks

- The GUI we built is a little simple and crude;
- The Randomness of the Starting placement was limited.
- The User need to wait for seconds to start a new game;

What we could have improved

- Improve the GUI and make it nicer;
- Enlarge the dictionary of interesting starting placements

Conclusion

- We did a good group work though we had conflicts during the programming
- We all gained a capability of efficient communication, we can show our ideas clearly to our teammates and discuss about the opinions
- Our programming skills got a great improvement since we were always trying to optimize our programmes, and have a good experience in converting logics into codes
- A deeper understanding of structural programming

Thank you for Listening !