COMP1110 - wed18i

u6079329 - u6609337 - u6406312

James Frampton Reid - Feier Xiao - Lachlan McVicar

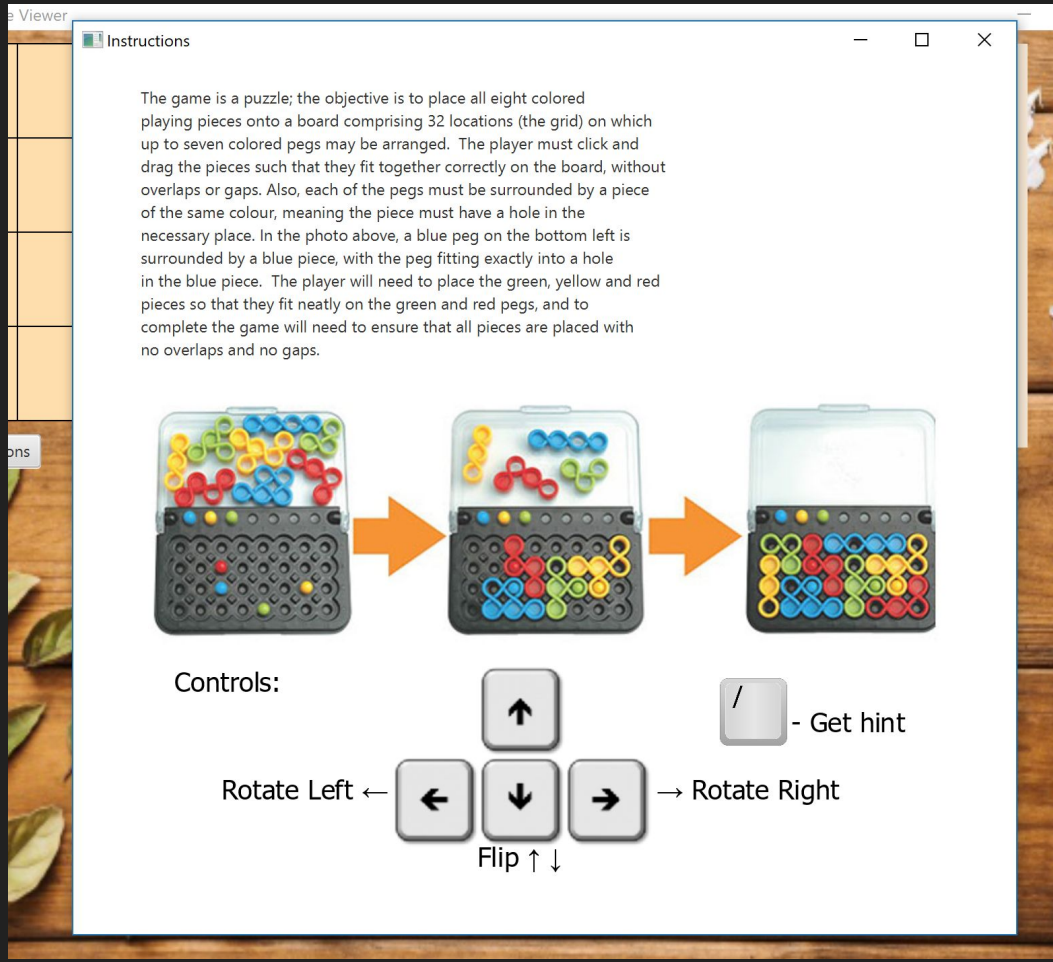
Instructions

Click on ‘Instructions’ button to bring up a new window that displays instructions

This is a whole new scene and was executed from within a scene.

Sceneception

The largest image is sourced from the IQ twist website

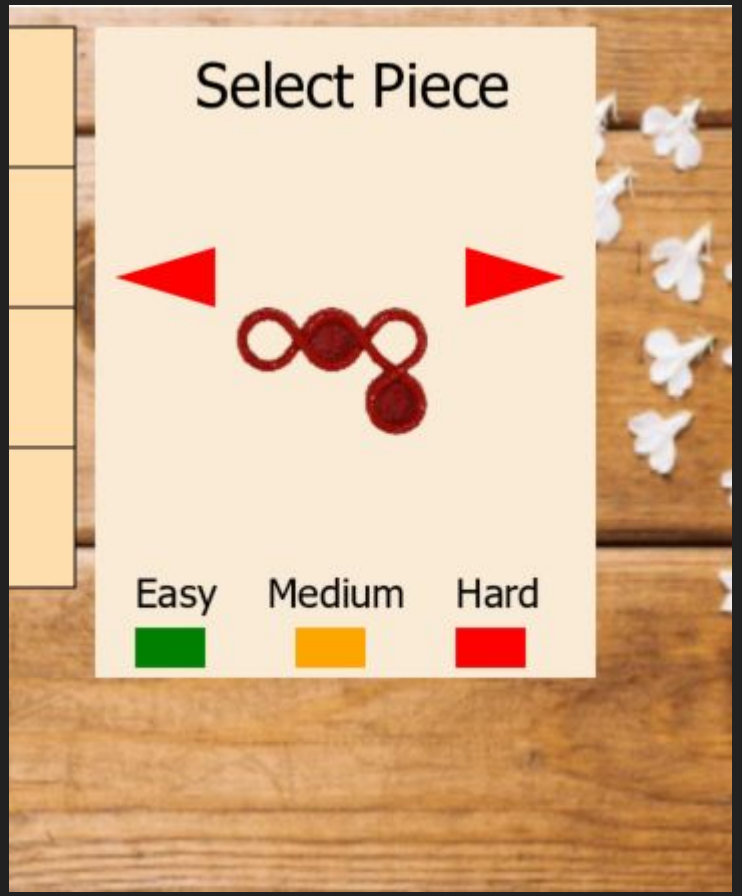


Selecting Pieces

Pieces are selected through this interface clicking on a arrow facing right will change the selected piece to the next piece and clicking on the left will change it to the previous piece.

This was done by analysing the dimensions of the image and basing the order in regards to the size of the image

Selecting duplicate pieces is impossible as it analyses the board state and skips pieces already on the board.



Score

Added ability to keep track of score.

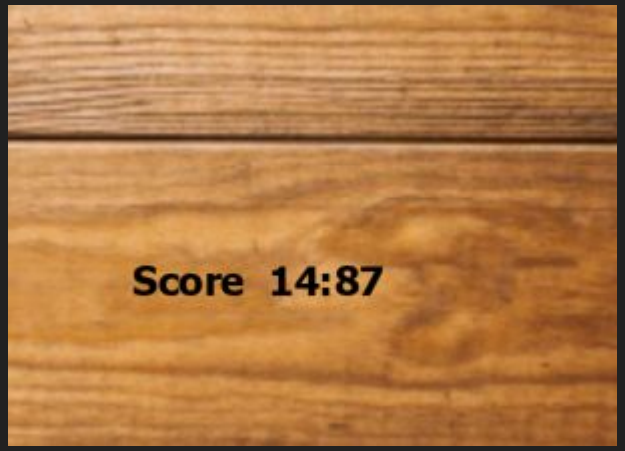
Pauses and turns green upon completion of the puzzle

Only starts once the puzzle has been generated

Resets upon creating a new puzzle

Is actually an invisible rectangle hurtling at the speed of light

Why? It was more economical in regards to the animation



Difficulty levels

Selecting easy generates a puzzle with 6 pegs

Medium generates 4 pegs

Hard generates 3 pegs

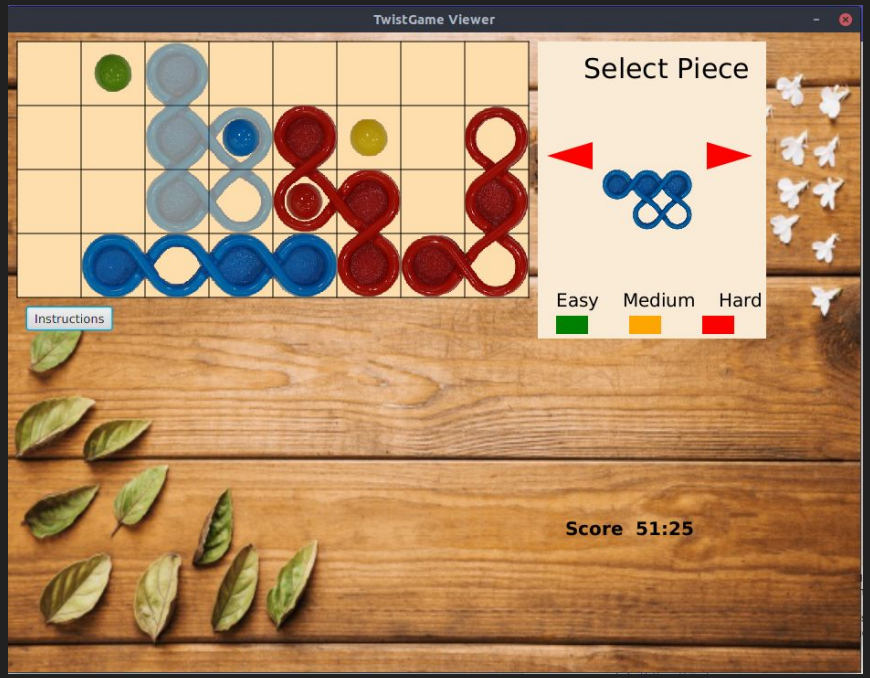
All generated puzzles have a unique solution

GetHint

Generate and toggle spooky ghost hints with the ‘/’ key

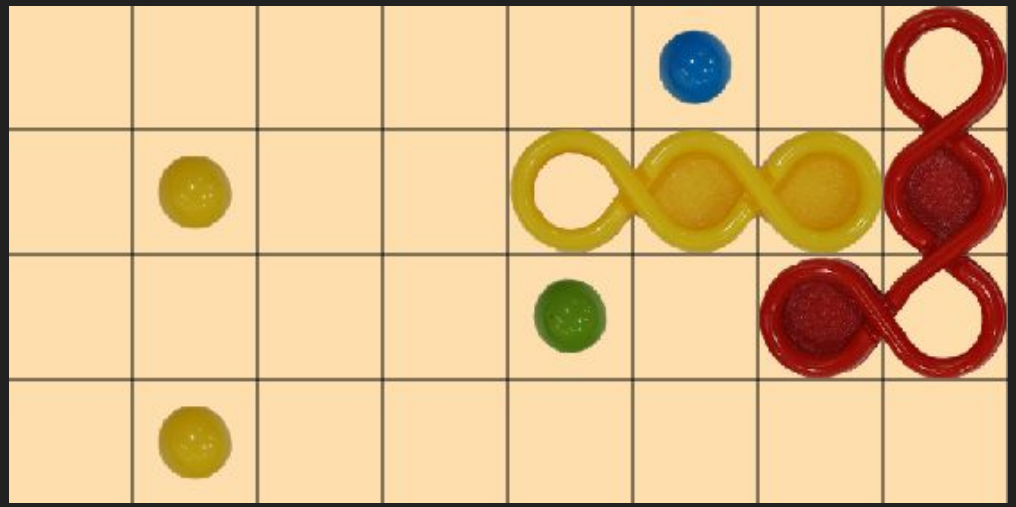
Each solution is unique to each generated challenge

These spectral pieces allow you to see the pegs underneath



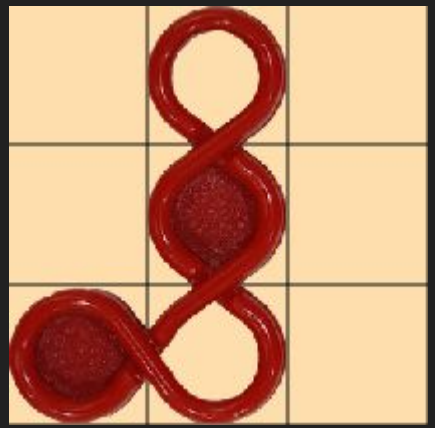
Solving a Board State

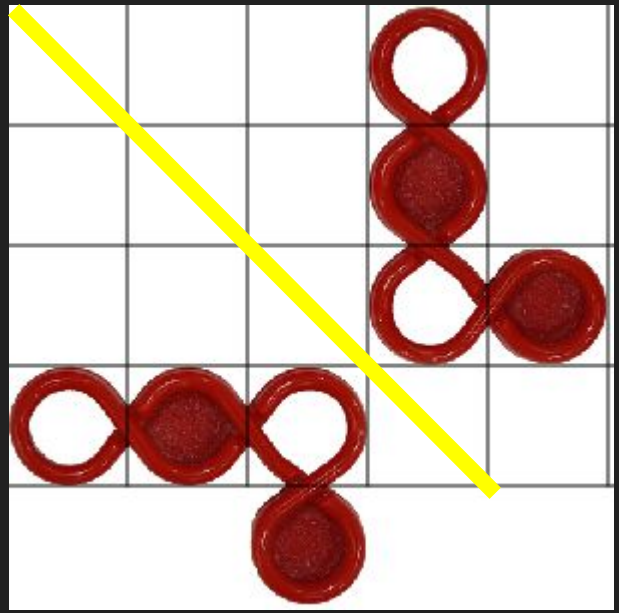
- Use Recursion to place pieces
- Use a blackList to ensure a placement is only checked ONCE
- Up to 720x faster
- Other Minor Optimisations, try to front load the work



Rotating Pieces

- No obvious function
- Reflection -> Flip -> Translation





Dictionary

At the beginning, we tried to iterate all possible initiate peg placement, and put it into the getSolutions function to check whether it will return only one solutions.

But no matter how we optimized our project, we couldn’t run through them. There are too much possibilities.

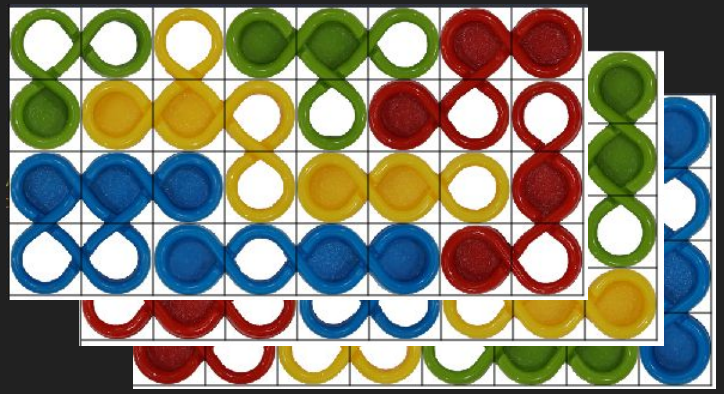
Then we changed the way we look at this problem

Dictionary

1. Get all the possible solutions without any pegs

   Send “” to the getSolutions function

   5992 possible solutions

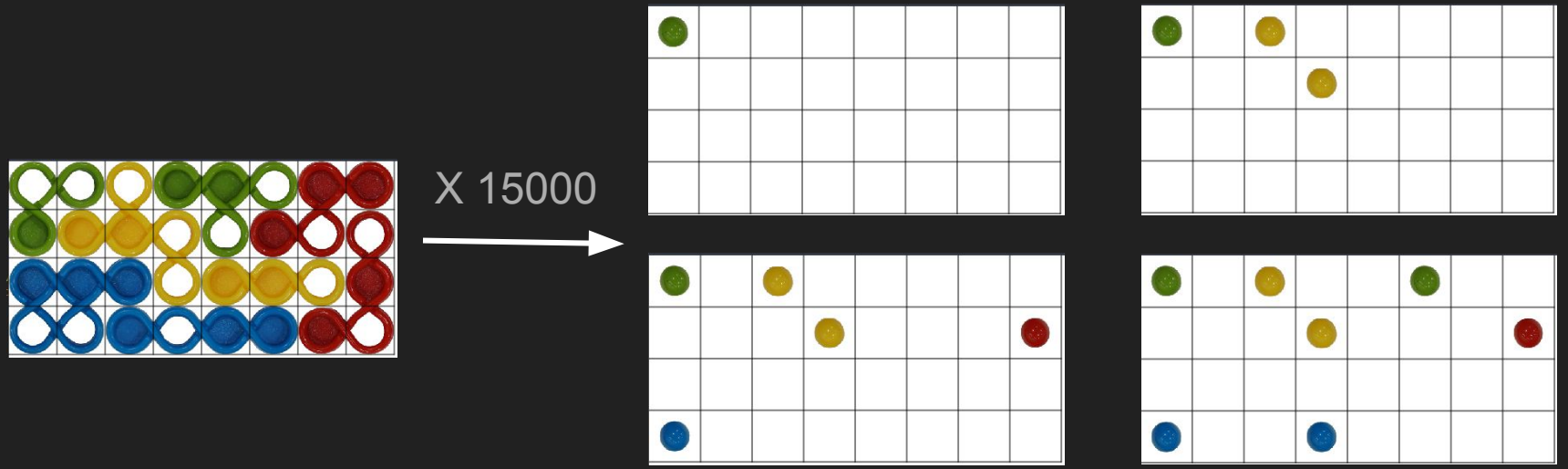


X 5992

Dictionary

2. Put pegs on that solution (Get the possible combinations of the pegs)

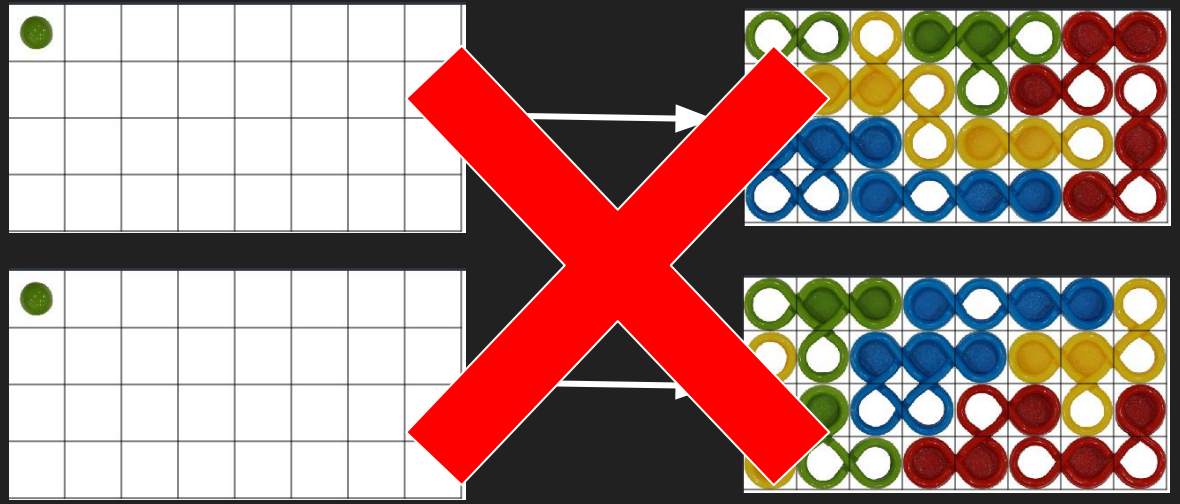
   15000 possible peg placements for every possible solutions



Dictionary

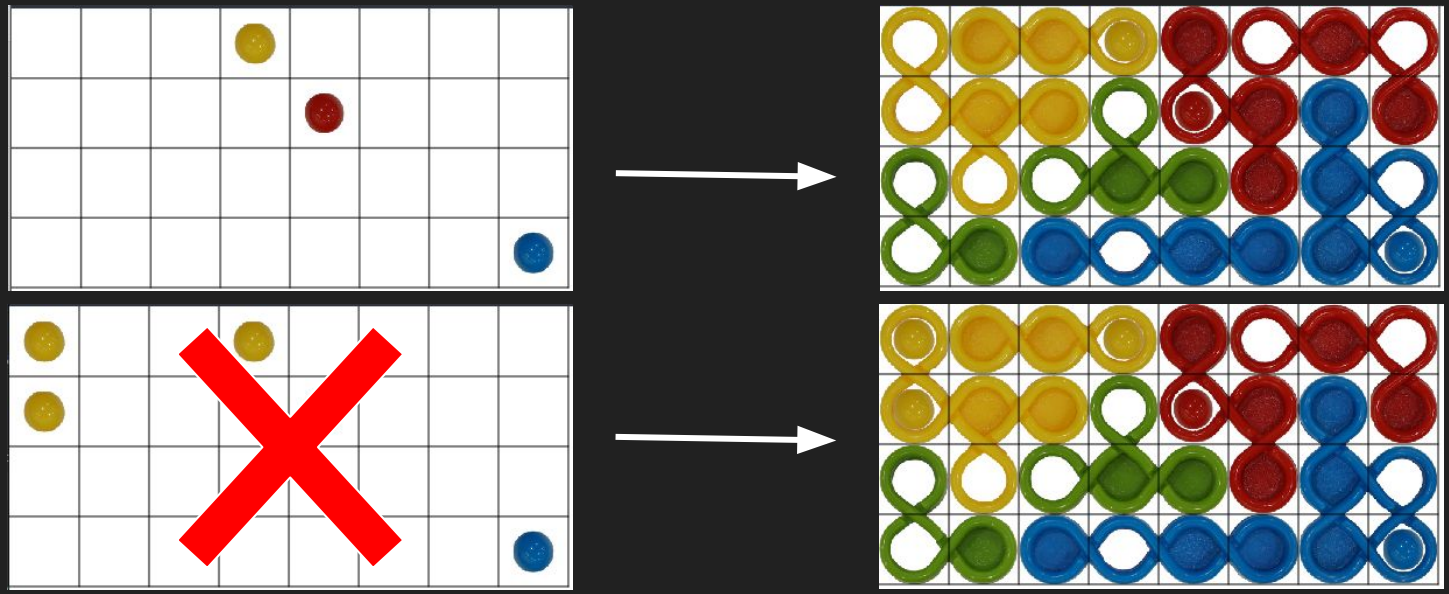
3. Get rid of the peg placement if it is existed in the key of the HashMap, so that the peg placement will have a unique solution

   26 MILLION !!!



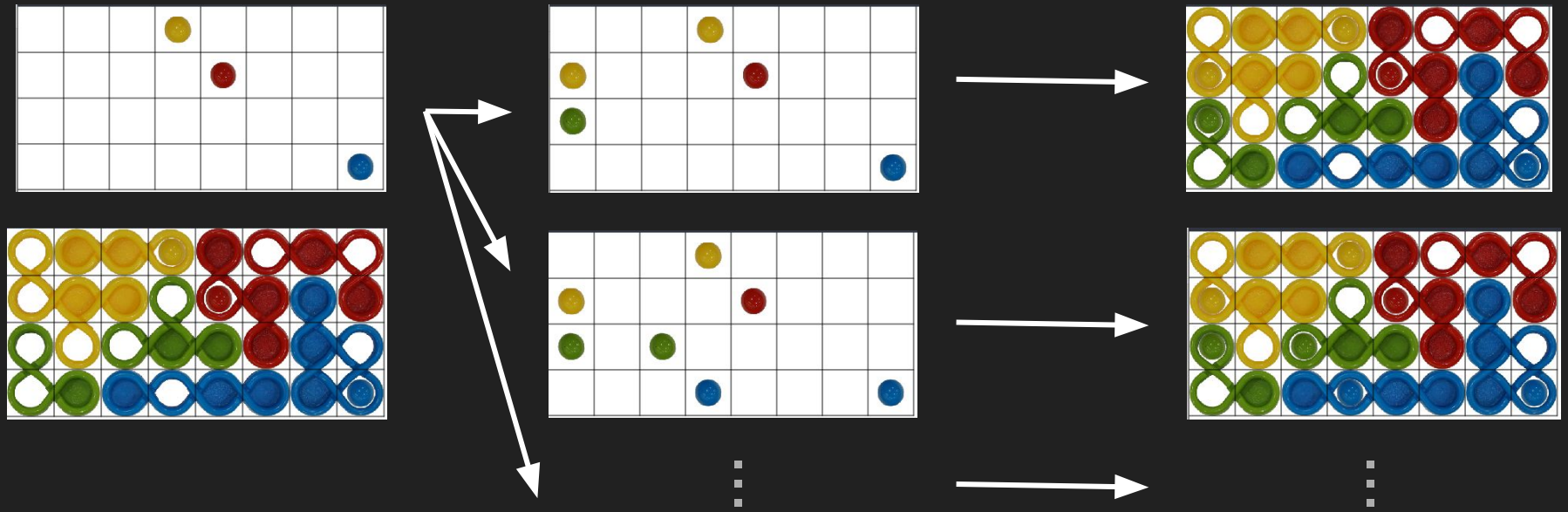
Dictionary

Unfortunately, we couldn’t process the 26 million of data using Java, which will throw an out of memory exception, otherwise we can reduce the size of the dictionary, by getting the minimal subset.



Dictionary

Then we found out all the three peg placements and add more pegs (according to the difficulty selected) to form a new placement during the game. If we can have the minimal subset, it would be complete and perfect



Thank you for listening !