Informatique
IUT Nord Franche-Comté

Technical report

Jessy MOUGAMADALY, Lucas DA SILVEIRA, Theo GASNER, Ewan HUMBERT,

Yassine CHAID

SAE 1-2-6

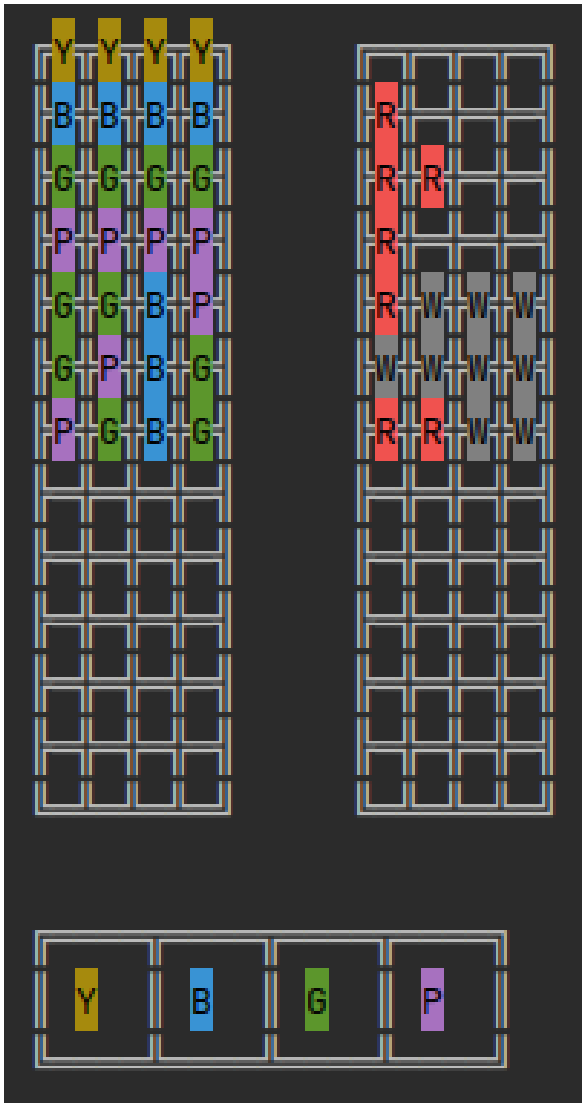

Table des matières

Technical report on the implementation of the Mastermind Board Game

Objectives

The objective of this project is to implement a playable version of the board game Mastermind for computers. The game should allow the user to fight against a human or a robot. The implementation must be well structured and written in a way that can be maintained. It should be possible to propose two versions of the game with different A.I..

A.I. Presentation

It's a quite difficult to create strategies that are sure to win in all cases in the Mastermind, there are different strategies that exist but they are not flawless and the win rate depends on the configuration of the current game for example, the A.I. will have more chances to win a game in cases where we have very few colors, in contrast, it will be harder to win the more colors we have. That's not all, in our version of Mastermind not only the number of colors can be changed but also the number of columns and rows, the more columns, the more the possibilities so the harder it gets to find the secret code, but the more rows we have the more chances we get to find the code, so we can expand the number of tries.

For our project we have settled for two different strategies, the two are clearly not flawless, especially the second one, but it will be further developed in the following part.

First A.I. : Color tester

The strategy for the first A.I. is follow :

- The I.A. will collect all the colors playable in the current game
- It will try for each color a completed row of color
- It will get the number of correct guess (matching and correct colors)

Jessy MOUGAMADALY, Lucas DA SILVEIRA, Théo GASNER, Ewan HUMBERT, Yassine CHAID

Etudiants en première année de BUT Informatique

Université de Franche-Comté
IUT Nord Franche-Comté
19 Av. du Maréchal Juin
90016 Belfort

- Retain the corresponding color in the required number of times to create a the right set of colors for the code
- If the number is equal to the number of columns, the A.I. will generate the possible permutations for the determined set of colors
- Once all the permutations are foud, the A.I. will try each one and it will be depended on luck to find the right answer before the number of completed rows equal the maximum number of rows.

As we can see, this A.I. is interesting to find the correct code by testing each color, but the necessary number of try to find the code will greatly depend on how many colors is in the code and the number of columns, in the case there are not enough rows, it will be impossible to find the right code and the A.I. will always lose.

Second A.I.

This A.I. is partly inspired by Donald Knuth's strategy, but was adapted to a variable number of rows, colors and columns. The strategy for this A.I :

- The I.A. will collect all the colors playable in the current game
- It will generate all the possible answer with these colors and the number of columns
- It will try a random possibility in the list of possibilities and check the number of correct guess
- If the number of guess equals to the number of columns, but not all the colors are place in the right spot, it will generate a list of all the permutations and try each one until it finds the correct one or reach the maximum number of completed rows
- Else if the number is not equal to the number of columns, it will remove the last answer from the list of possibilities and all the permutations possible with the last answer, and try a new answer randomly from the list of possibilities

Now, you can see a major flaw in the strategy of this A.I., the flaw is that for all the try that does not equal the number of columns, it will just remove the answer and all the permutations of this answer in list to the possibilities, and that's a big problem because there is no analysis from the A.I. to determine some pattern in the wrong answer to remove similar answers in the list of possibilities and thus shorten greatly the list of possibilities. So we can say that this A.I. is partly

random and greatly depends on luck, but if we improve the way the A.I. excludes the answers that are not possible, we think that it will greatly improve the win rate of the A.I..

Implementation

The Mastermind board game has been implemented in Java using the Boardifier library. Boardifire allows to create enhanced graphics for board games, making them more attractive and enjoyable to play. Several design patterns were used in the implementation, such as the Observer pattern to handle user input and the Singleton pattern for the Board class. Boardifire also allows developers to customize all aspects of their board game, including the board, pieces and rules.

The implementation includes several key features such as the ability to choose the difficulty, the ability to track the number of attempts made by the user, and the ability to provide feedback on each guess. Code snippets are provided below to show some of the key elements of the implementation.

Entry/Trace files

For the purpose of testing the game, we have generated a number of entry/trace files that we can be used as input files for the game by doing as follow :

```
java Mastermind 0 --cols=a --colors=b --rows=c  --aimode=0 < file
```

The “a” has to be replaced by the number of columns.

The “b” has to be replaced by the number of colors.

The “c” has to be replaced by the number of rows.

Each of those paremeters depend on the entry/trace file used, you can find the information about the parameters in the name of the folder containing the trace files. Each entry/trace file is as follow :

GBYP

GGGG

BBBB

YYYY

PPPP

BYGP

YBPG

PYBG

PBYG

GBYP

Each line corresponds to a answer (so an user input), in this case, the first line is the code and the lines after are the tries to find the code. The number of lines of a trace file is between 2 and the maximum number of rows + 1.

Conclusion

The implementation of the board game Mastermind was a success and all goals were achieved. The use of object-oriented programming principles and design patterns resulted in a well-structured and maintainable code base. The tests performed confirmed the accuracy and reliability of the implementation. In future projects, we encourage you to add more difficulty levels and add new game modes to make the game more challenging and engaging for users.