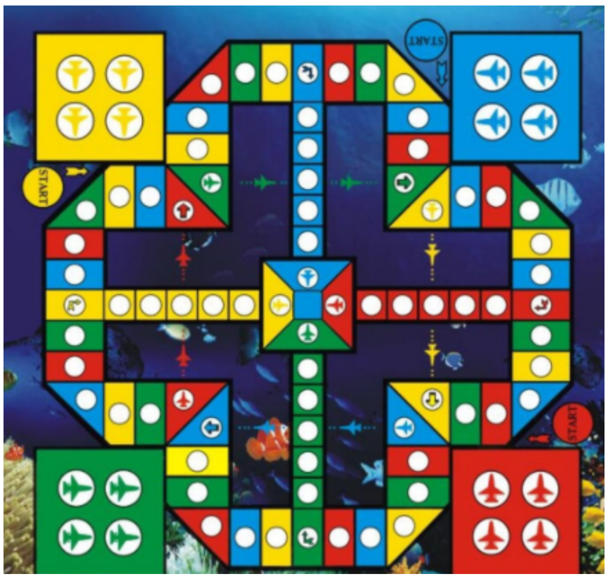
20F Java A Project - Aeroplane Chess

Introduction

Aeroplane Chess (飞行棋) is a Chinese cross-and-circle board game. In this project, you are required to program the game with customized rules. The introduction to this game below is mainly adopted from Wikipedia.

Chestboard



M’ m’ m

Objective

Two to four players each try to get all their own plane pieces from their hangars, located at the corners of the board, into the base of their own colour in the centre of the board. Each player takes a turn by rolling the two dice. On a turn a player may do the following:

- Take a piece out of the hangar onto the board. This can only be done by rolling a 6 with either of the dice (e.g., roll a 3-6, or 6-2, or 6-6).

- Move a piece that is on the board clockwise around the track. The number of spaces moved is derived from the dices with arithmetic operations, maximum 12. For example, rolling a 2 and a 4 will let you to move any one of your plane by 2, 4, 2+4=6, 4-2=2, 4*2=8, or 4/2=2 spaces. Note that a 4 and a 3 cannot make a move of int(4/3) spaces.
- When a player lands on an opponent's piece, the opponent returns that piece to its hangar.
- When a plane lands on a space of its own colour, it immediately jumps to the next space of its own colour. Any opposing planes sitting on these squares are sent back to their hangars.

Besides the above basic rules, there are additional rules for you to implement:

- There are additional shortcut squares. When a plane lands on one of these of its own colour, it may take the shortcut, and any opposing planes in the path of the shortcut are sent back to their hangars.
- If the sum of the two dices is no less than 10, whether they are used to enter or move a piece, gives that player another roll. A second sum no less than 10 gives the player a third roll with enter or move. If the player rolls a third sum no less than 10, any pieces moved by the first two steps must return to their hangar and play passes to the next player.
- When a plane lands on another plane in its own color, the player can choose to stack the pieces and move them as one piece until they reach the centre or are landed on by an opponent. When stacked pieces are sent back to their hangar by an opponent landing on them, they are no longer stacked.
- When a plane lands on an opposing plane, players determine which gets sent back to its hangar by rolling one die, with the high roll determining the winner. When one plane attacks a stack of planes, it must battle each one by rolling the die. When a stack attacks another stack, the planes battle each other with a series of successive die rolls

until only one player occupies the square. This rule replaces the third basic rule above.

Ending

A plane must fly into the centre **base on an exact roll**. When a plane does so, it is placed face down back in its own hangar, indicating that it is done for the game. If a player cannot move pieces into the centre base by an exact roll, then they must move their piece backwards according to number rolled. The first player to get all four of their planes to the centre of the board wins.The rest play until there is only one loser.

Project Requirement

Task 1: Initialize Game. (20)

- Initialize a new game, and the board of which is cross-shaped, and any overlap of different components cannot be appeared.
- All routes, landing points, two components represent two dices, and different colors should be arranged in a suitable position on board.
- The game should includes four players.
- Can restart a game by clicking a button rather than closing it and open the game again.

Task 2: Load and Save a Game (20)

- Your program should be able to load an existing game from a text (or other type) file with a pre-defined format by clicking a button. After loading, all the components should be placed at their positions given in the text (or other type) file.

    prepared following test cases by yourself:

    1. An successful case that contains stacked planes

2. An successful case that one of the four players can win the game within four steps.
3. Multiple successful cases that can test one of the eight rules with one or two steps.

- Your program should be able to perform error check.

  prepared following test cases by yourself:

  1. An error case that the count of players is not four.
  2. An error case that the count of cells in 4-players' route are not same.
  3. An error case that lacking of landing points.
  4. An error case that the count of dice is not two.

- Your program should be able to save the current game into a text (or other type) file.

Task 3: Play the game (25)

- Your program should detect the winning status of the game, and end the game when there is a winner.
- In a round, the number of steps that each airplane can take depends on the sum, difference, product and quotient of the results of the two dice. Other than that, in a round a maximum step should be set in the game.
- During one game, your program should be enable to switch between the normal mode and the cheating mode where the dice numbers can be manually selected, so that it is convenient for testing.
- All four basic rules in the Objective section are implemented.
- All four additional rules in the Objective section are implemented.

Task 4: Graphical User Interface (15)

- Your program should have a graphical user interface using Java Swing. (FX can also be accepted)

Advanced Requirements (20)

- Design Human vs. Machine Mode, and make the machine player smarter.
- Design a platform for your game, such as multi-loading, ranking list, adding start menu for different functions of the game, etc.
- Design a game mall to sell props, that can be used in game to make your game more interesting.
- Make your game looks nicer, such as changing the theme, adding sound effect, adding background music, adding more prompt label when playing the game, showing possible moves after dicing, playing animations of the moving of airplain.
- Support on-line mode in Local Area Network.
- More.