SOFTWARE SPECIFICATIONS

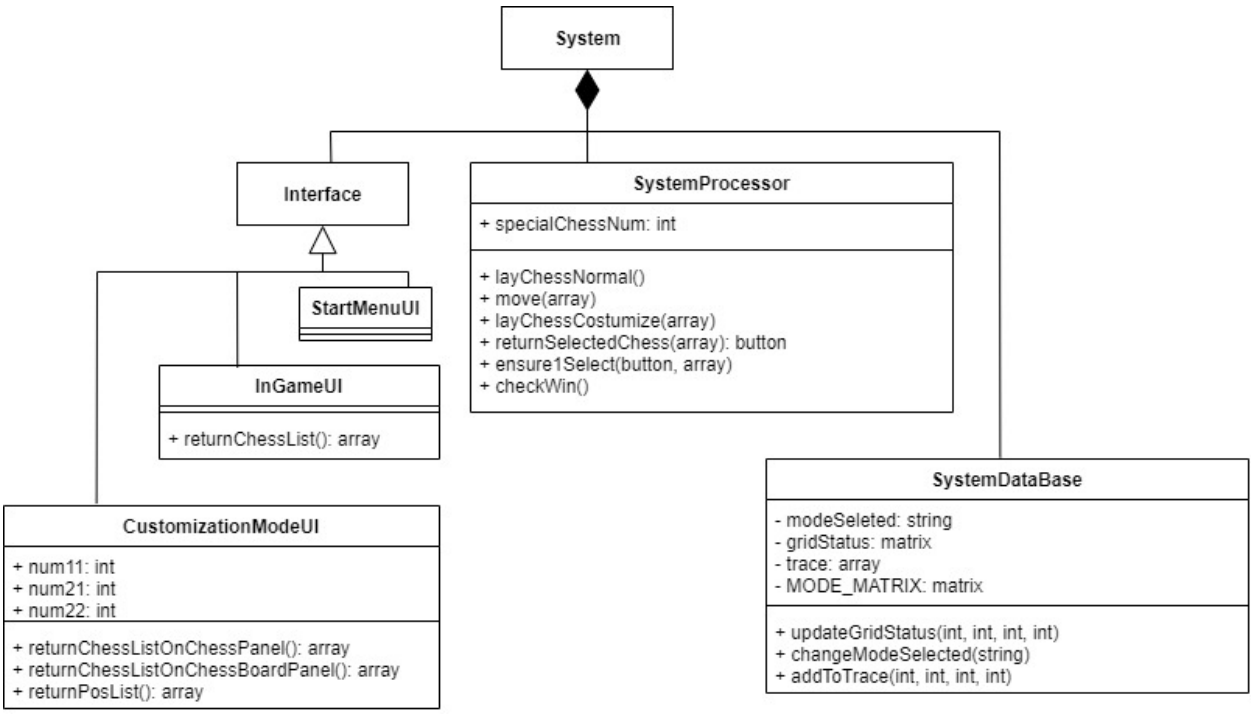
Huarong Path System

Group X
Author

Table of Contents

System Architecture

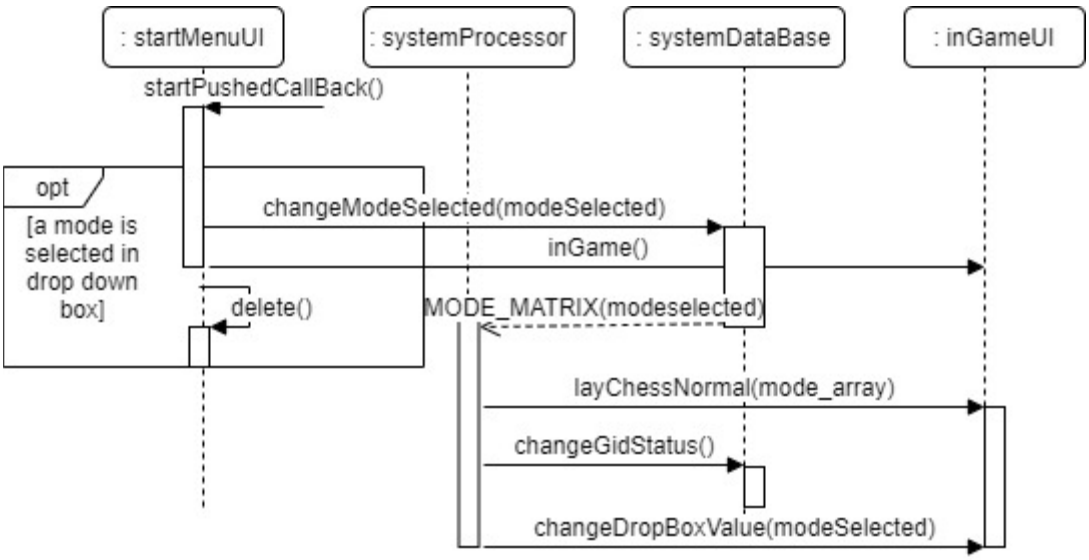
The system architecture is shown below:



Software Specifications

S1: startMenu implementation

S1.1: Select standard mode



- S1.1.1: Select a standard mode

  1. Select a standard mode in the drop-down box

2. Press ‘start’ button

a. If the selected mode is a named mode in the systemDataBase, set ‘modeSelected’ in database to the mode name. Open inGame.mlapp and delete current UI. Using an array representing this mode in database to lay chess.

* The array used to represent chessboard is 10*3. 10 means 10 chesses. In the length-3 cell, 1st number is used to represent the chess’s row, 2nd number is for column, 3rd number is for shape: 1 for a 1*1 chess (‘zu’), 2 for a 2*1 chess, 3 for a 1*2 chess, 4 for a 2*2 chess (‘CaoCao’).

b. If the selected is ‘--- --- ---’, nothing will happen.

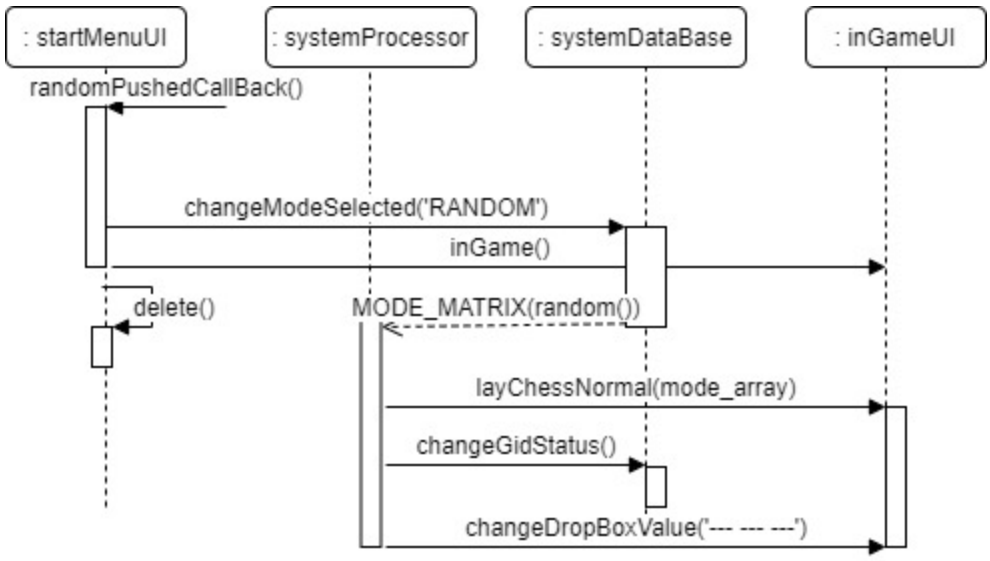
- S1.1.2: Lay chess using array

1. For each length-3 subarray in the 10*3 array, use an unused button in inGame to represent this chess through adjusting the button’s position and shape, add text on button according to the shape of the chess. Change gridStatus so the gridStatus can show where on the chessboard has been occupied correctly.

* The gridStatus used to represent chessboard occupancy is a 4*5 matrix with value of 0 or 1. 0 means there is no chess on this position, and 1 means the position has been covered by a chess.

2. In inGame, change the value in drop-down box so that it is the same as modeSelected in database.

S1.2: Select random mode



- S1.2.1: Press random mode button

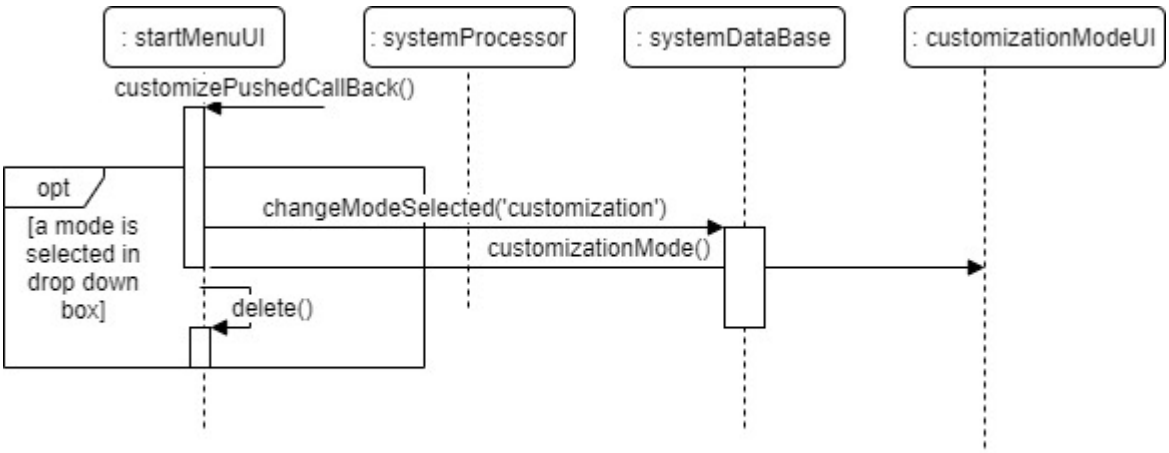
1. Set ‘modeSelected’ in database to ‘RANDOM’. Open inGame.mlapp and delete current UI. From the matrix restoring chessboards in database, randomly select an array.

- S1.2.2: Lay chess using array

  1. For each length-3 subarray in the 10*3 array, use an unused button in inGame chessboard to represent this chess through adjusting the button’s position and shape, add text on button according to the shape of the chess. Change gridStatus so the gridStatus can show where on the chessboard has been occupied correctly.

  2. In inGame, change the value in drop-down box to ‘--- --- ---’.

S1.3: Select customize mode

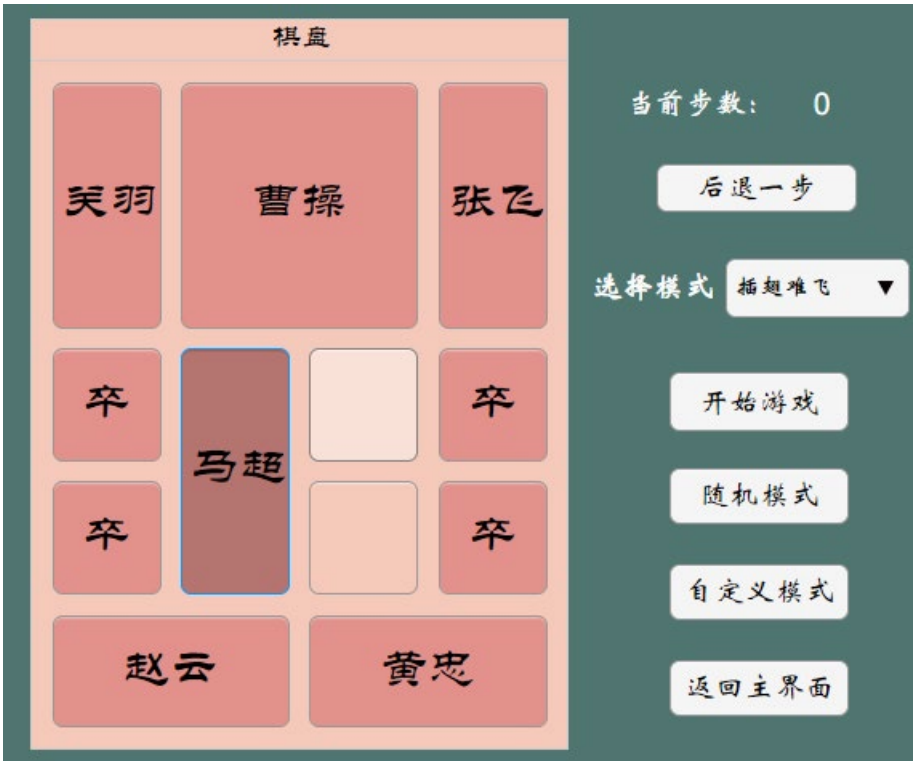


- S1.3.1: Press customize mode button

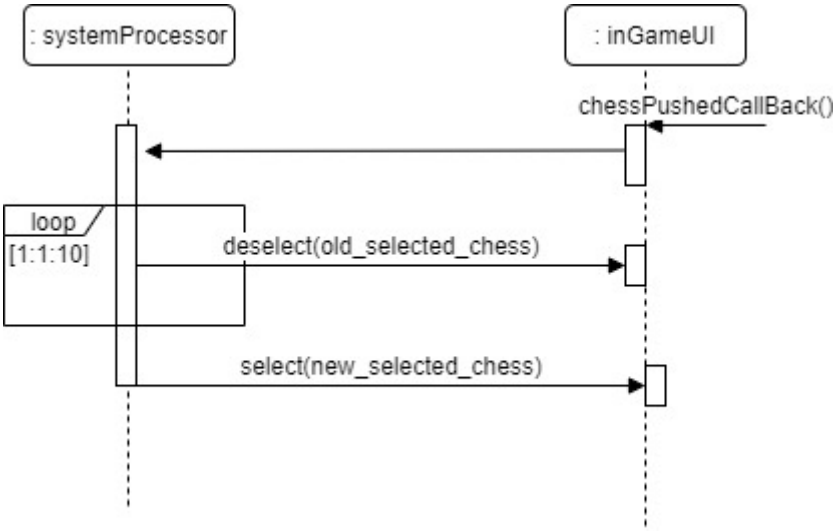
  1. Set ‘modeSelected’ in database to ‘Customization’. Change gridStatus to all ‘0’;

  2. Open customizationMode.mlapp and delete current UI.

S2: inGame implementation



S2.1: Select chess



- S2.1.1: Select a chess on chessboard

  1. Set the new selected chess's value to 'false'.

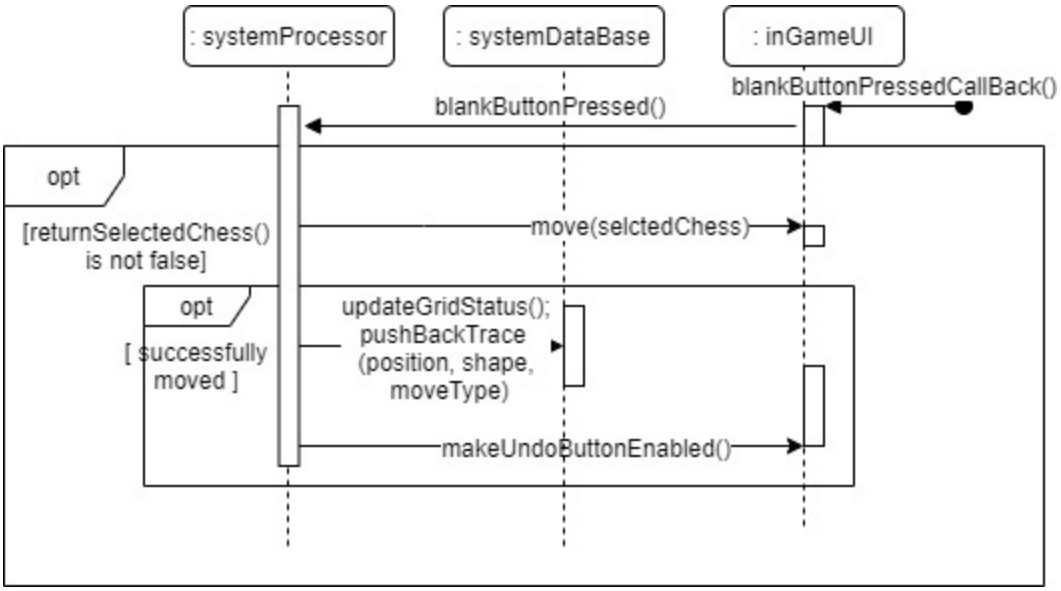
  2. Go through all chess button, check if there is already selected chess.

     a. If there is, set this old chess's value to 'false'.

     b, if there is not, nothing will happen.

3. Set the new selected chess's value to 'true'.

S2.2: Choose direction



- S2.2.1: Press a blank button on chessboard

  1. There are 2 blank positions on chessboard for player to move selected chess.

  2. After receiving the signal that player has press a blank button, the processor will go through all chesses to check if there is already selected chess.

     a. If there is, use this selected chess.

     b. if there is not, nothing will happen.

- S2.2.2: Move chess

  1. Compare the chess's position with the blank button's position, chess's shape should be considered.

     a. If blank button is on top of selected chess, try to move selected chess up. Check the gridStatus, if gridStatus[chessColumn: chessColumn +chessLength][chessRow(1)-1] exists '1', which means there is already other chesses on top of selected chess, the up() function will not be executed. Else,

     b. If blank button is on left of selected chess, try to move selected chess left. Check the gridStatus, if gridStatus[chessRow: chessRow +chessLength][chessColumn (1)-1] exists '1', which means there is already other chesses on left of selected chess, the left() function will not be executed.

     c. If blank button is under selected chess, try to move selected chess down. Check the gridStatus, if gridStatus[chessColumn: chessColumn +chessLength][chessRow(end)+1] exists '1', which means there is already other chesses on top of selected chess, the down() function will not be executed.

d. If blank button is on right of selected chess, try to move selected chess right. Check the gridStatus, if gridStatus[chessRow: chessRow +chessLength][chessColumn (end)+1] exists '1', which means there is already other chesses on right of selected chess, the right () function will not be executed.

2. If the chess successfully moved, set its origin position in gridStatus matrix to '0' and new position to '1'. Push this movement to the back of database.trace[].

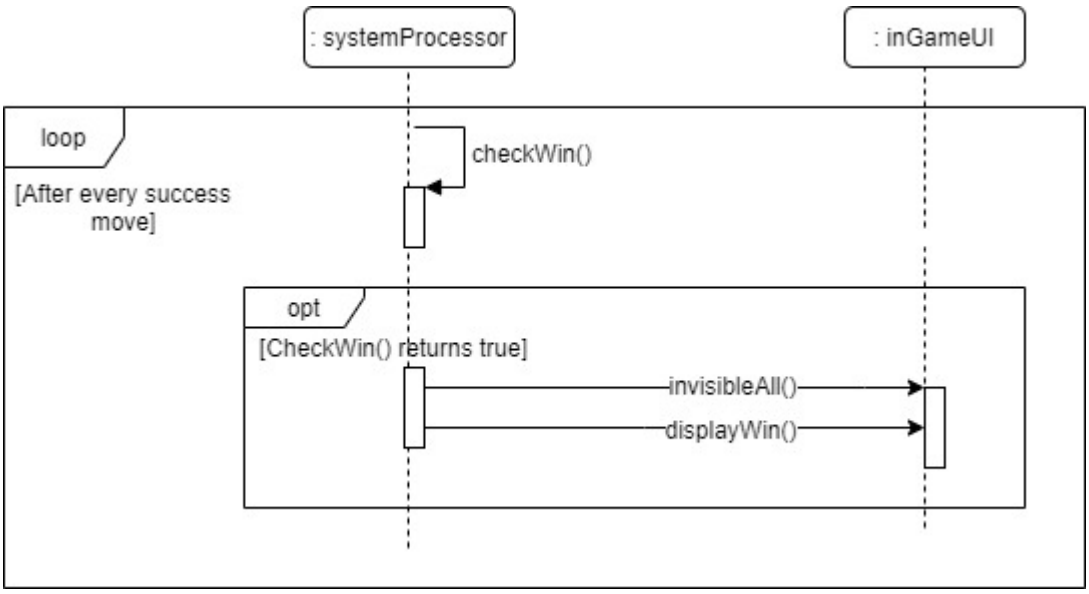
* The trace is stored in database, which is an n*4 array. In the length-4 cell, the 1st number represents the chess's row(1) after moved, the 2nd number means the chess's column(1) after moved, the 3rd number shows the shape of the moved chess, the 4th number represents the direction of movement: 1 for up, 2 for right, 3 for down, 4 for left.

3. Make 'Undo' button enabled.

- S2.2.3: Lay blank button

1. Go through gridStatus, put 2 blank buttons on the 2 positions with value '0'.

S2.3: Declare win if wins



- S2.3.1: Check win

  1. After every success movement, go through chesses on chessboard.

     a. If the chess's shape is 2*2

        (i) If the chess's on the exit, declare win.

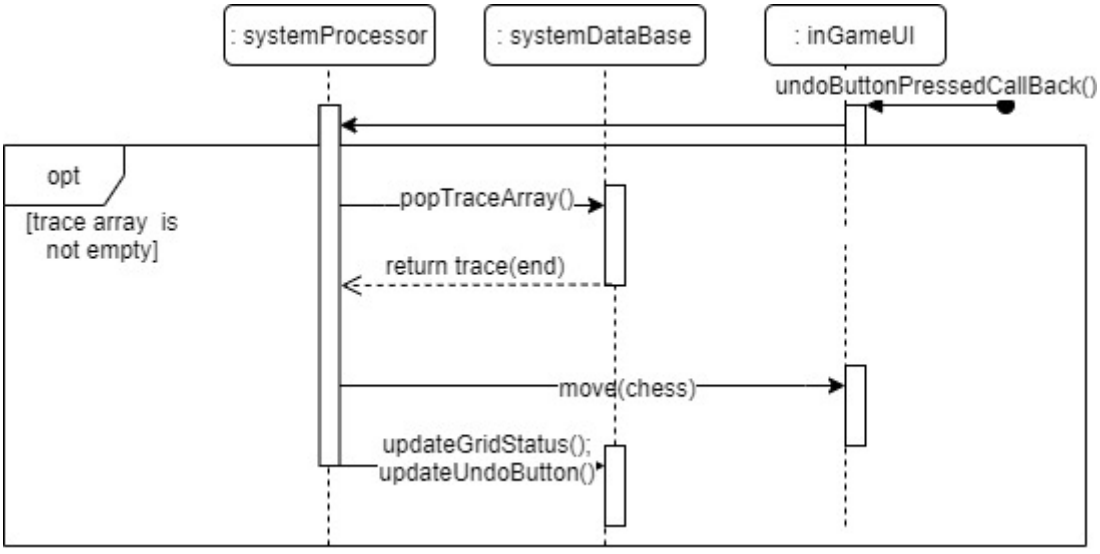
        (ii)Else, nothing will happen.

- S2.3.2: Declare win

  1. Make all button on chessboard invisible.

  2. Make 'You Win !' label visible.

S2.4: Undo



- S2.4.1: Pop trace
  1. If the trace array is not empty: pop last line from trace array.
- S2.4.2: Move Chess to opposite direction
  1. Use the array's information to move chess on corresponding position. Direction is the 4th number of the array: 1 to move down, 2 to move right, 3 to move up, 4 to move left.
  2. As the move function used above is same as normal move function, which will add a line of this move to trace array anyway. So we need to manually delete the last line of trace array.
- S2.4.3: Update 'Undo' button

  1. If the trace array is empty now, make 'undo' button disabled.

S2.5: Select other modes

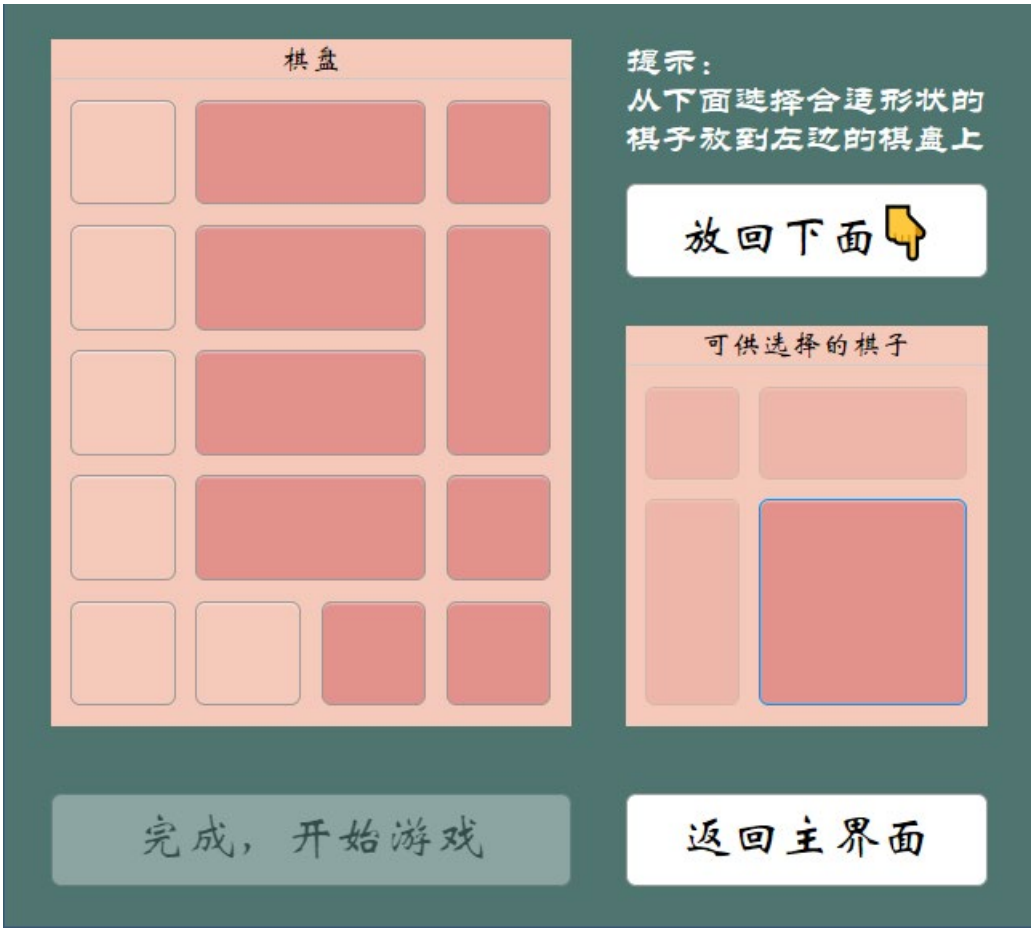
Please refer to *startMenu implementation*.

S2.6: Back to StartMenu

- S2.6.1: Press 'back to startMenu' button.

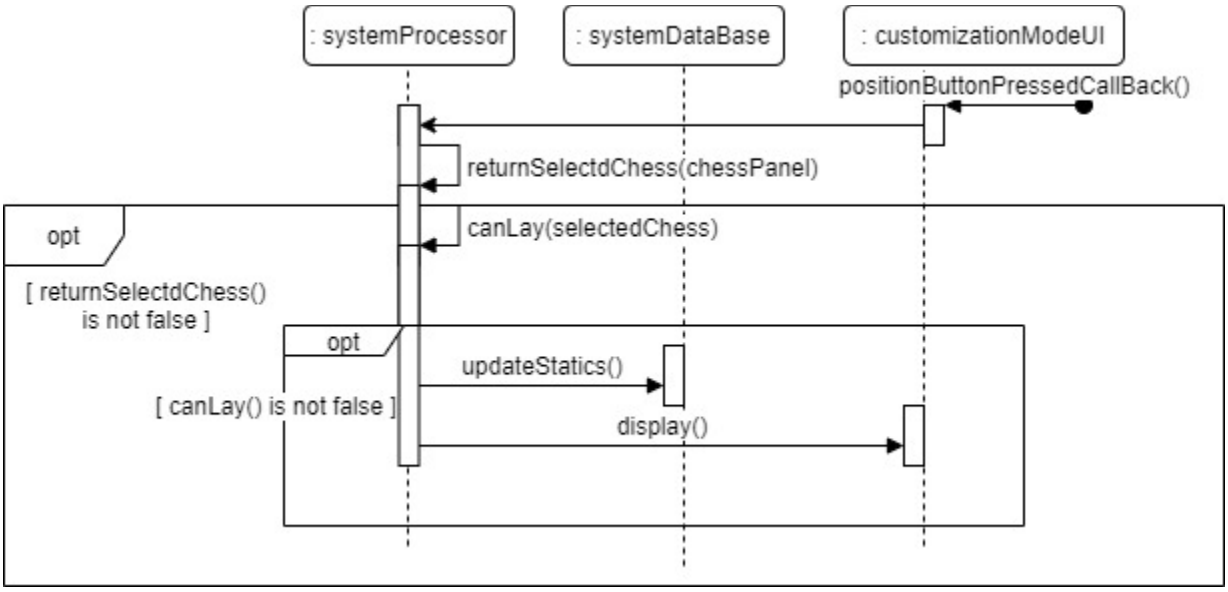
  1. Delete current UI, open startMenu UI.

S3: customizationMode Implementation



* The chessboard is covered by 2 layers of buttons. The upper is position button layer (visible at start), and the other is chess button layer (invisible at start). When there exists a chess on the position, the position button will become invisible, and a proper chess button will become visible.

S3.1: Drag chess



- S3.1.1: Select position button on chessboard

  1. If a position button on chessboard is selected

     a. Check if any chess in '可供选择的棋子' is selected.

     b. If no chess in '可供选择的棋子' is selected, function will directly return.

  2. Use selected chess's shape and position button's position, together with gridStatus to determine if the selected chess can lay on the chessboard.

     a. If canLay(), a proper chess button will be adjusted to suit the required position and shape and become visible, and position button on this position will become invisible.

     b. If canLay() returns false, function will directly return.

- S3.1.2: Update statics and display

  1. GridStatus will be updated according to current chessboard, and the number of different chesses will also be updated. (This is stored in 3 variables in cutomizationMode.mlapp).

  2. Update '可供选择的棋子' according to the number of different chesses. If number of 1*1 shape chesses reaches 4, then the 1*1 chess button will be disabled. If number of 1*2 shape chesses add number of 2*1 shape chesses reaches 5, then the 1*2 and 2*1 chess buttons will be disabled. If number of 2*2 shape chess reaches 1, then the 2*2 chess button will be disabled.

  3. Update finish button according to the number of all chesses. If the number of all visible chesses on chessboard reaches 10, the finish button will become enabled.

S3.2: Put Chess Back

- S3.2.1: Press putBack button

  1. If putBack button is pressed: go through all visible chesses on chessboard to check if any chess is selected.

     a. If a chess is selected, the chess will become invisible and its value will be changed to 'false'. The position button(s) on its position will be visible.

     b. If no chess is selected, the function will directly return.

- S3.2.2: Update statics and display

  1. Change the number of corresponding chess according to the selected chess's shape. Make this chess of this shape in '可供选择的棋子' enabled.
  2. Update gridStatus according to current chessboard.

  3. Make finish button disabled.

S3.3: Commit and start games

- S3.2.1: Press finished button

  1. For each chess in customized chessboard, use a 3*1 cell to represent its row, column and shape. The 10 chesses will make up a 10*3 array, which is same format as the array stored in database.

  2. Open inGame.mlapp ,delete current UI

- S3.2.2: Lay chess using array

  1. For each length-3 subarray in the 10*3 array, use an unused button in inGame to represent this chess through adjusting the button's position and shape, add text on button according to the shape of the chess. Change gridStatus so the gridStatus can show where on the chessboard has been occupied correctly.

  2. In inGame, change the value in drop-down box to '--- --- ---'.

S3.4: Back to StartMenu

- S3.4.1: Press 'back to startMenu' button.
  1. Please refer to *Back to StartMenu*.

S4: find_solution implementation

Uppaal is used to find solution to any layout.

If want to see the uppaal model, please refer to model checking in validation document.