## **Criterion C: Development**

1. Graphical User Interface

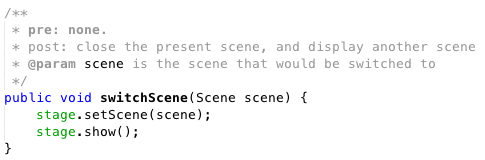
Using JavaFX

1. Algorithmic thinking
   1. Thinking abstractly
      1. Object Oriented Programming - polymorphism - multiple class with has-a relationships
      2. GUI features in JavaFX
      3. Mouse and keyboard action listeners
   2. Thinking logically
      1. Boolean expressions/algos and loop structure
      2. 3D array to represent the cells
      3. Multiple loops to go through all the indexes of the 3D array
   3. Thinking Procedurally
      1. User-defined methods
   4. Thinking ahead
      1. Error-checking mechanism in login and signup scene
         1. The accounts cannot have the same username
         2. Prevent mistyping when create account
         3. Prevent a user login both players with the same account
         4. Use of passwordFields
      2. 2D representations of the layers beside the cubic board
         1. Allow users to see the board clearly
         2. Users’ actions on the cubes will also be done on the 2D board
      3. Highlighting the cell according to the coordinate of the mouse
         1. Allow users to click the right cell
      4. isOccupied() method to prevent a cell being selected twice
         1. Make sure the game running without error
   5. Computational Thinking
      1. Math.random() method to randomly select the starting player
         1. Make the game to be fair
      2. Algorithm to check whether three cells are selected by the same user as a line - in all directions (vertical, horizontal, and diagonal)
2. Storage mechanism
   1. Use database tables in JavaDB

The informations in the database can be added and updated

(209 words)

1. Thinking abstractly
   1. GUI for abstraction and visual game board configuration



The scenes are switched by calling the method switvhScene(). All GUI components are initialized in different types, including buttons, labels, textFields, passwordFields, tableViews, imageViews, scenes, and a stage. There are scenes representing each scene. Groups containing the components for each scene using getChildren().add() method, and the groups are shown by the scenes. The position of GUI components is set by getTransforms().add(new Translate(x, y)) method. All buttons are set with actions using setOnAction(e -> method()) with user-defined methods. The size of the textFields, passwordFields, and buttons are set with setMinWidth() and setMaxWidth() methods. Some components are hidden using setVisible() method



After login successfully, hide the textFields and the Enter button, display a correct sign and wait for the other user





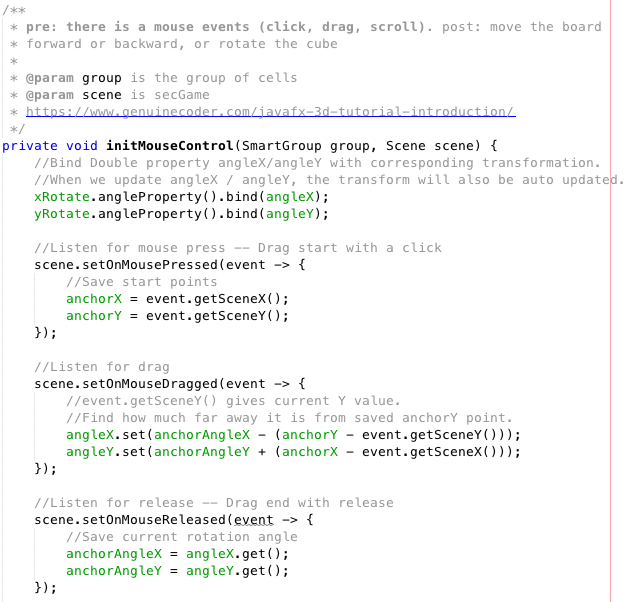
The diagram on the right side is a 2D representation of the cubic board with three layers, which assign each cell a number, and shows which cell is taken by which player using different colors.





* 1. Mouse action listeners

The dragging process for mouse control is divided into three steps: press, drag, and release. When pressing the mouse, record the initial coordinate of the mouse on the scene. Then, during the dragging (move the mouse) change the direction of the cube according to the changed coordinate of the mouse during the dragging. After dragging, the mouse is released, and record the current coordinate of the mouse for the use of the next drag. When scrolling the mouse, change the position of the cube along the z-axis, and there is a limit during scrolling to make sure that the cube would not be moved out of the sight.

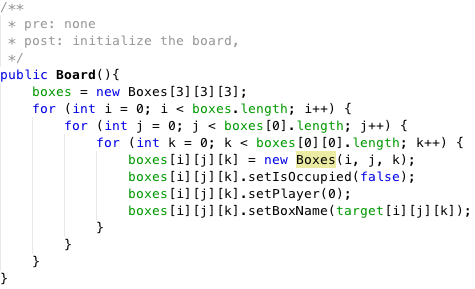


When the mouse moves, the mouse listener will find whether the mouse is on a cell. If event.getTarget() of the mouse movement is a cell, then that cell will be highlighted by changing the material of that cell to a brighter image. If the mouse moves out of that cell, the material of that cell will be changed back.



1. OOP and Data Structure
   1. 3D-Array to store the board configuration and the has-a relationship between classes

The Game class has-a Board as an object. A variable “boxes” in the Board class, which is a 3D-array, represents the cells in the game board. The object boxes stores the objects defined in the Boxes class. The methods setIsOccupied() and setPlayer() are used to store the information that whether a specific cell is selected and which player selected it, where isOccupied (boolean) and player (int) are variables in the Boxes object. The boxes array is initialized by the constructor in the Board class (a Board has multiple Boxes).



Each cell (or Boxes object) in the array are assigned to three indexes representing the cell’s position in the three directions on the board (along x, y, and z axises). These boxes objects in the array are modified according to player’s action as the game goes on.



* 1. A chacking algorithm to find the winner for each round

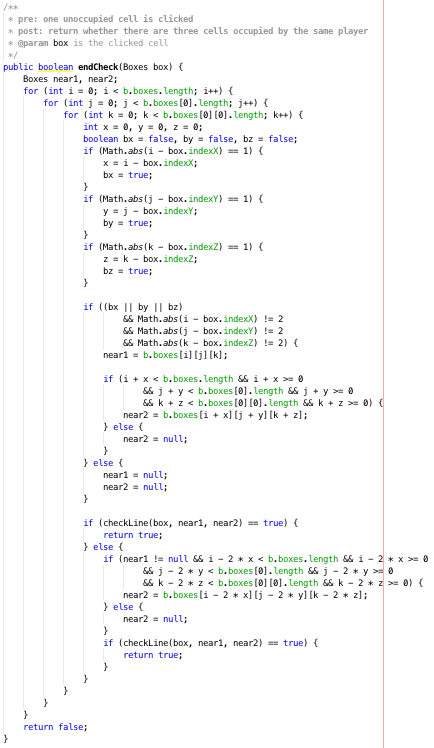
In the Game class, the method click(), result(), checkEnd(), and checkLine() are used to operate the 3D array when a certain cell is clicked.

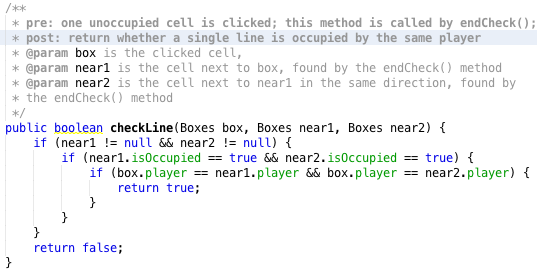
Once the player selected a cell on the board, the program will check whether there are three cells in a line are selected by the same player. From the cell [0][0][0] to [2][2][2], the program will firstly find all the neighbour cells around the selected cell. Then distinguish whether the surrounding cells are selected by the same person. If true, find the third cell on that direction and check its color.

For example, if a player selected the cell on the left-top corner (cell 1), it will find that cell 2, 4, and 5 are near the selected cell; among the three surrounding cells, only cell 2 and 5 are blue, so continue to check cell 3 and 9 which are on the same line with 2 and 5; then, because 3 is blue and 9 has not been chosen, cell 1, 2, and 3 are connected as a line and the player with blue cells wins this round.



Method endCheck() is called by the click() method once a cell is selected by a player. EndCheck() method firstly find the possible directions that can have three cells, and then call checkLine() to find whether the three cells in each direction are selected by one player by comparing the three Boxes’ player variable objects in parameters





1. Java DB to store, update, and read wins and loses for each account in a local database
   1. Add rows into the database

In the signup scene, this method is called by clicking the btnSignupEnter button. Check whether the information in txtSignupPassword and txtSignupPasswordConfirm are the same. The add a new roll into the database table according to the information in the textfFelds. The playing history (wins, losses) is all 0.

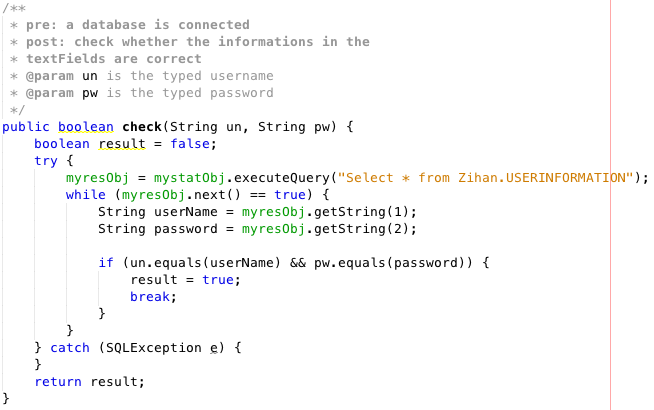


Sample database table: There are already six accounts in the database, and each account is stored as an row in a table. The username and password columns are varchars, used for user authorization. The other two columns are integers, recording each user’s playing history



* 1. User authorization

This method is called by btnEnter1() and btnEnter2() to check whether the username and password in the textFields are the correct with the database. A while loop is used to go through each roll of in the database table, and the getString(int) method return the information in the first (username) and the second (password) column.



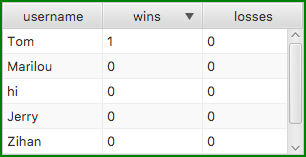
* 1. Update the wins and losses in the database and output the user’s information as a ranking table from the database

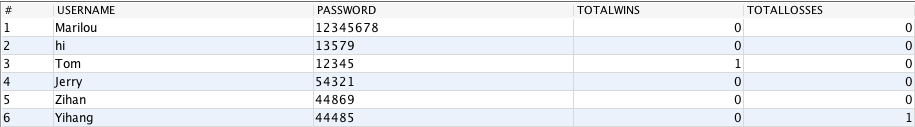
When a player wins a round, a while loop will go through all the rows (accounts) in the database table and find the username of the winner and loser and find their rows. Then the winner’s total wins in the database will added by one and the loser’s total losses will be added by one using updataInt() method. MyresObj.getInt() returns the value in the column for a specific row.



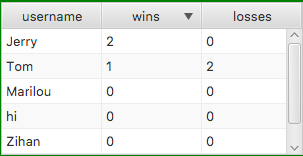
For example, if Tom plays this game with Jerry, before they play:

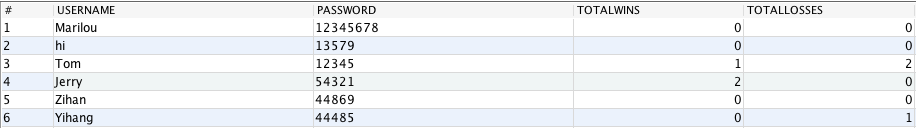
Tom has one win and Jerry has 0.

(GUI)

(Local file)

After two rounds, if Jerry wins both rounds, Tom have 1 win and 2 losses, and Jerry has 2 wins and 0 losses.

(GUI)

(Local file)

(1052 words)

Works Cited

*YouTube*, YouTube, www.youtube.com/watch?v=HLx1ySEVAgQ.

*YouTube*, YouTube, www.youtube.com/watch?v=CITiIS0OL1k.

Genuine Coder. “JavaFX 3D Tutorial #1 - Introduction to 3D Development.” *Genuine Coder*, 27 Sept. 2018, www.genuinecoder.com/javafx-3d-tutorial-introduction/.

Genuine Coder. “JavaFX 3D Tutorial #12 – Depth Buffer.” *Genuine Coder*, 13 Oct. 2018, www.genuinecoder.com/javafx-3d-complex-shapes-and-depth-buffer/.

Genuine Coder. “JavaFX 3D Tutorial #2 - Camera Systems.” *Genuine Coder*, 27 Sept. 2018, www.genuinecoder.com/javafx-3d-camera-systems/.

Genuine Coder. “JavaFX 3D Tutorial #3 – 3D Object Transform (Rotation) with Keyboard Input.” *Genuine Coder*, 27 Sept. 2018, www.genuinecoder.com/javafx-3d-transform-objects-with-keyboard-input/.

Genuine Coder. “JavaFX 3D Tutorial #4 – 3D Object Transform (Rotation) with Mouse.” *Genuine Coder*, 27 Sept. 2018, www.genuinecoder.com/javafx-3d-tutorial-object-transform-rotation-with-mouse/.

Genuine Coder. “JavaFX 3D Tutorial #5 – More Fine Mouse Control - Zoom in and Zoom Out.” *Genuine Coder*, 27 Sept. 2018, www.genuinecoder.com/javafx-3d-mouse-control-zoom/.