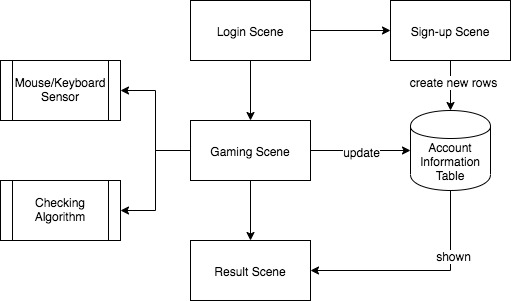
## **Criterion B: Design**

**System flowchart:**



Login Scene

* Identify the users by usernames and passwords which are stored in the database

Sign up Scene

* Allow users to create an account; add new items into the database

Gaming Scene

* The main part of this application
* Users will play tic-tac-toe in this page
* Checking algorithm will be called to check whether the game is ended and who wins

Result Scene

* Display the result of the game (who is the winner) and rank

Database

* Store the users’ username, password, number of wins and number of loses

**Software design framework**

Because the client’s problem and requirements are pre-defined (he wants a 3D version of tic-tac-toe), the developing processes are clear and will go step by step. The phases in SDLC (System Development Life Cycle) model is similar to the criteria for the IA, including the planning, designing, coding, testing, and evaluating parts.



**Mockup 1 - the login page**



* The first interface when the application is turned on
* The textFields and buttons for the two users are separate
* textFields and passwordFields
  + Users should type their username and password to log in
  + The typed text for the password should be hidden
* By clicking the enter button, the username and password in the textFields will be compared with the items in the database table
* The Sign-up button allows users to switch to the sign-up scene if they don’t have an account

**Mockup 2 - the sign-up page**

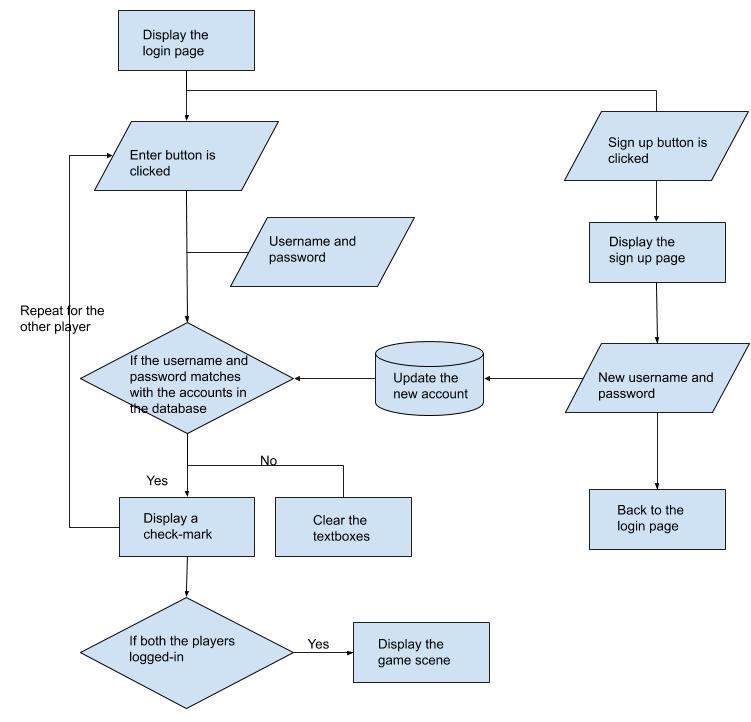


* The users should retype the password to make sure there isn’t mistyping
* If the username already exist in the database, the new account cannot be created

**Database table mockup**

|  |  |  |  |
| --- | --- | --- | --- |
| Username | Password | Total wins | Total losses |
| Tom | 12345 | 3 | 2 |
| Jerry | 54321 | 2 | 3 |
| Jeck | abcde | 5 | 2 |
| Rose | 314159 | 1 | 4 |

**Algorithm flowchart 1 - logging-in and sign-up**



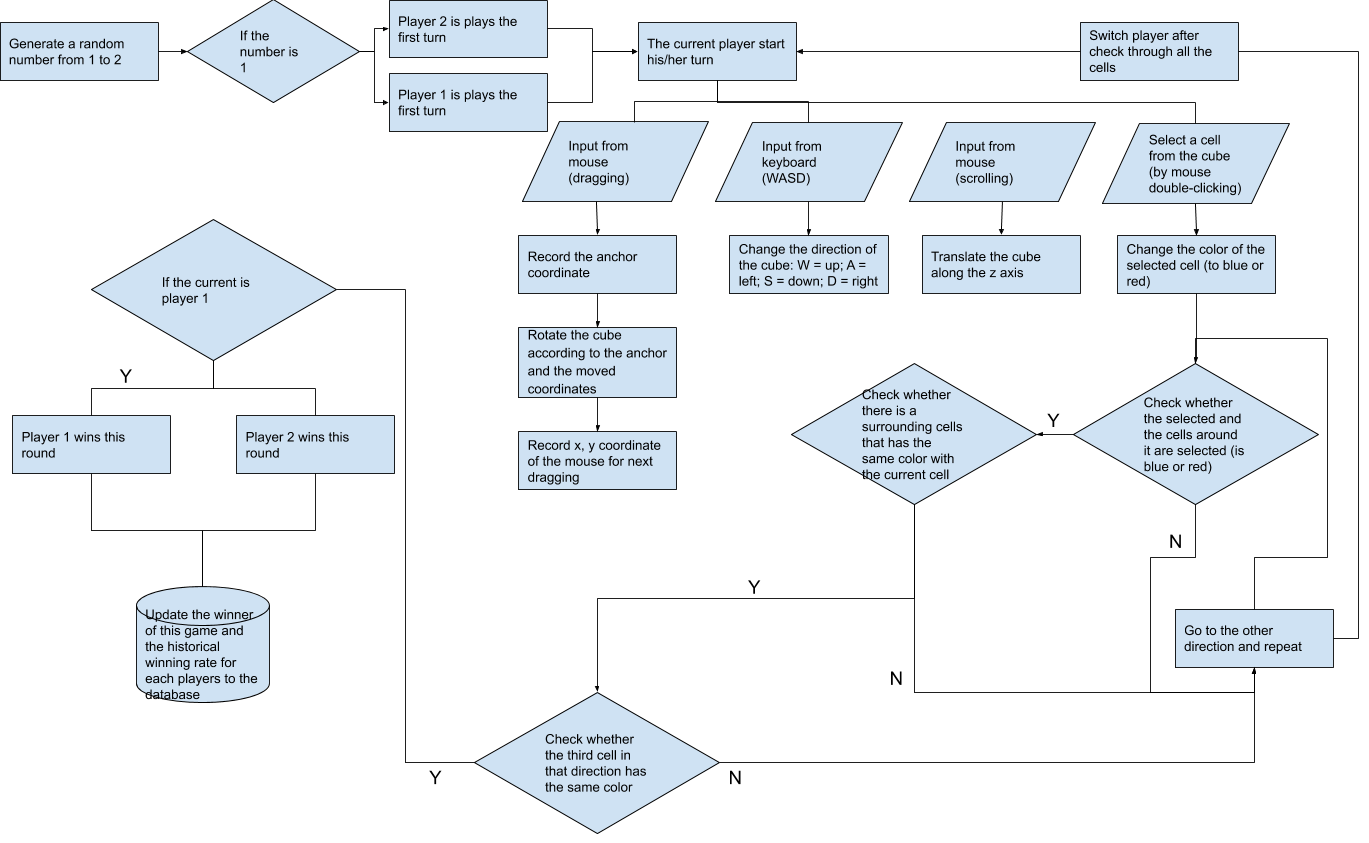
* The game will start when both the players login correctly

**Mockup 3 - the gaming page**



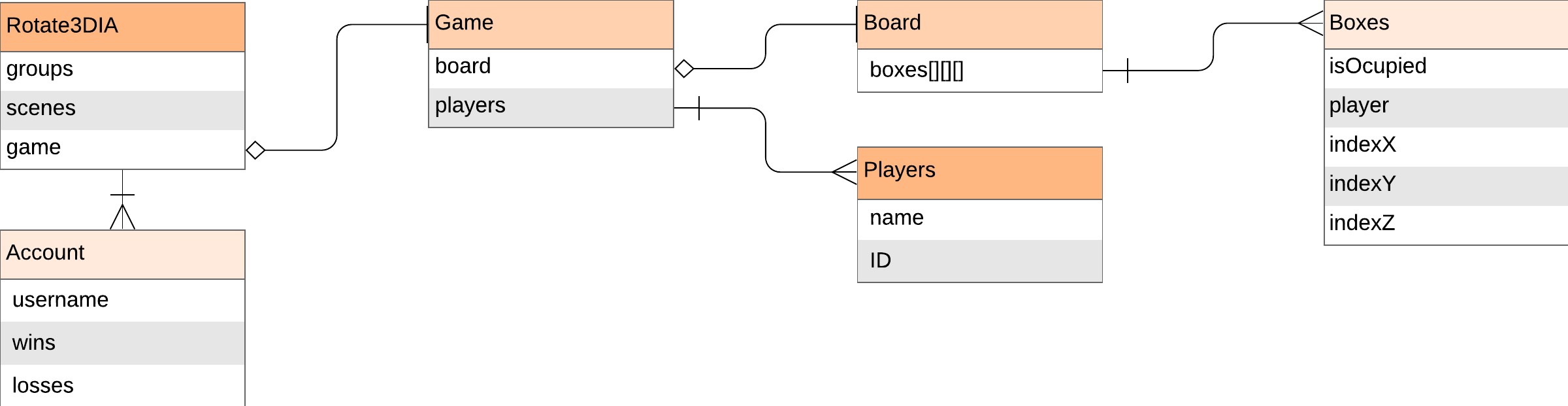
* Display after login
* Allow the users to play the game
* A game board
  + There is a cube in the middle of the scene as the board. The users can turning it to change the direction by dragging is using mouse or WASD on keyboard and move the cube further or closer by scrolling
  + After double-clicking, the colour of the cell could be changed
  + The is a 2D representation (the three layers) of the cube on the side showing each cell’s colour so that the users can see through the layers directly

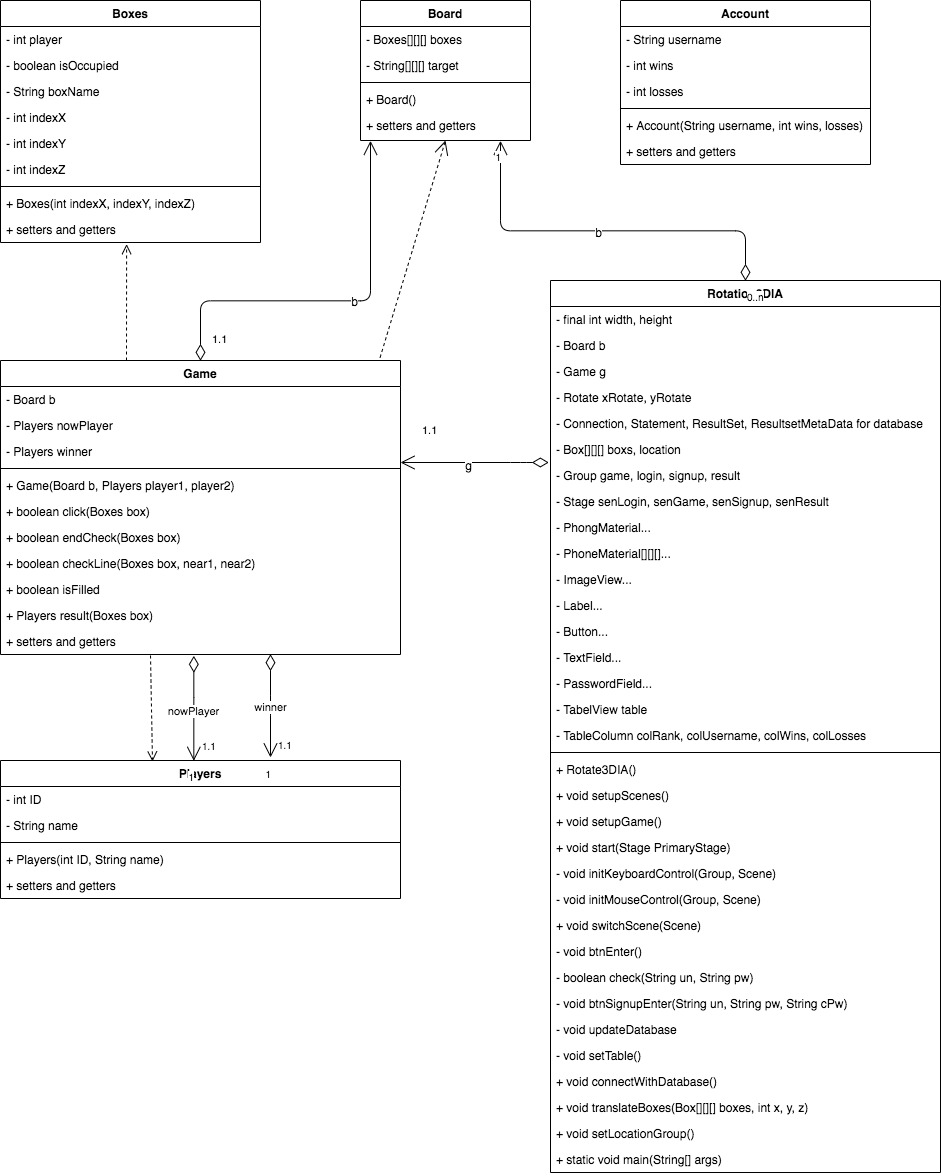
**Algorithm flowchart 2 - actions during the game**



* As entering to the gaming page, randomly select the first player
* During his/her turn, the user can either turning the cubes (by keyboard or mouse) or selecting a cell by double-clicking
* Once a user clicks a cell, the colour of the cell will change to blue or red; then, check whether there are three cells in a line are connected
* The cells (cubes) are stored in a 2D-array
* Users cannot select occupied cells
* Once there is a winner of the round, update the total wins and losses for the player to the database

**Entity relationship diagram**



**UML - judging algorithm and relationships**

Variables in Game class

* A Game has-a Board and several Players objects
* “Winner” is the username of the winner player

Methods in Game class

* Game(Board b) - initialize all variables and randomly select a starting player
* Click(Cells cell) - when a player clicks an unoccupied cell, change the characteristics of the cell; then call the endCheck() method
* endCheck(Cell cell) - a loop calls the checkLine() method to check whether three cells are occupied by the same player for each direction (horizontal, vertical, and diagonal) of the clicked cell.
* checkLine(Cells cell, near1, near2) - check whether the same player occupies the clicked cell and the other two cells in that line (near1 and near2)
* isField() - check if the game is a draw
* result(Cells cell) - return the username of the winner

**Pseudocode 1:**

click()

If the cell does not belong to any player, then

The cell is occupied by the current player

Call the endCheck() method, if return true, then

End the game

Else

Pass the turn to the other player

Else print “this box is occupied”

End method

**Pseudocode 2:**

endCheck(selectedCell)

Declare two local variables near1, near2 (cells)

Three for-loops to go through every cells on the board

Local variables x, y, z for the index of the cell in the array

Distinguish the cells near the selectedCell

Using if statements to find the position of the selectedCell on a line (whether on the end or middle of the line)

For loop to go through all directions to find the cell near the selectedCell - near 1 and the third cell on that direction - near 2

Call checkLine(selectedCell, near1, near2) method to find whether the clicked cell, near1, near2 are selected by the same player

If checkLine(selectedCell, near1, near2) is true, then

Return true

End loop

Return false

End method

**Pseudocode 3:**

checkLine(selectedCell, near1, near2)

If near1 and near2 cells are not null, then

If near1, near2, and the selectedCell are selected by the same player, then

Return true

Else return false

Else return false

End method

**Pseudocode 4:**

initMouseControl()

when mouse press

Record x, y coordinate of the mouse as anchors

when mouse drag

Rotate the cube according to the anchor coordinates and the new coordinates of the mouse

when mouse release

Record x, y coordinate of the mouse for next dragging

when mouse scroll

Translate the cube along the z axis

when mouse double-click

Run click()

when mouse move

If mouse is on the cube, then

Highlight the cell - change the surface image

End method

**Mockup 4 - the result page**



* The table displays the top five player’s information (username, wins, and losses)
* The users can quit the application by clicking the button

|  |  |  |
| --- | --- | --- |
| Test Plan | | |
|  | Action to be tested | Testing method and Expected output |
| 1 | Open the application and display the first scene | Click run file in Netbeans - a window is shown in the middle of the desktop; all GUI components are displayed in the correct place with the right size. |
| 2 | Functions of the GUI components in the login scene | Type texts into the textFields and passwordFields - Text can be typed into the fields. The text in the textFields will be shown, but the text in the passwordFields will be hidden as ••••••  Click “enter” and “sign-up” buttons - The enter button and sign-up button can be clicked with the intended functions |
| 3 | Login features in the login scene (for both player1 and player2) | Enter pre-registered accounts - the textFields and “enter” button will be hidden, an image of correct is displayed  Enter incorrect username/password or unregistered username - the texts in the fields are deleted, a label shows that the username or the password is wrong |
| 4 | Sign-up a new account | Register with a pre-registered user’s username - the text in the passwordFields will be deleted; a label will show that the username already exists  Enter different text in the password and confirm password fields - the text in the passwordFields will be deleted; a label will show that the two passwords are not the same  Register with a new username (the password and confirmed password are the same) and view the database table - after clicking “enter” button, the login scene is shown again, and a new row is added into the database table with the correct username and password (other information are 0)  Click back - back to the login scene |
| 5 | Turning and scrolling the cubic board | Drag the cube using a mouse - the cube will turn to the direction where the mouse moves to  Scrolling the mouse - the cube will move closer or further from the screen (bigger or smaller) in a range  Drag the cube using WASD on the keyboard - the cube will turn to the intended direction |
| 6 | Clicking the cells | Bothe the players click the cells - after one player double-clicks the cell, the cell will become blue, another player is red. |
| 7 | The 2D representation of the cubic board | Place the mouse on the cubes and move the mouse: the selected cell will be highlighted; the related cell on the 2D representation will be highlighted. |
| 8 | Winning condition | The first player connects his/her three cells in a horizontal line (number 1 2 3): a button is shown, on which it displays the username of player 1  Vertically (cell 1 10 19): same with above  Diagonally (cell 1 11 21): same with above  Diagonally in different layers (cell 1 14 27): same with above |
| 9 | The result scene | Click the “go to result page” button: A table with three rows and three columns is displayed. Player 1’s “wins” cell is added by one, Player 2’s “losses” cell is added by one. |
| 10 | Quit the application | Click the “quit” button: the window is closed. |