**CENG 317 PROJECT REPORT**

In this Project we aimed to do a classical tic-tac-toe game on a three-by-three board, Consider a classical tic-tac-toe game on a three-by-three board, where the winner places three marks in a row, horizontally, vertically or diagonally.

**Describing The Program**

Tic-tac-toe is typically programmed as a state machine. Depending on the current-state and the player's move, the game goes into the next-state. In this example, I use a variable currentState to keep track of the current-state of the game,and define named-constants to denote the various states of the game (PLAYING, DRAW, CROSS\_WON, and NOUGHT\_WON). A method called updateGame() is defined, which will be called after every move to update this currentState,

by checking the status of the game-board.

In this Project we have 2 classes in one class such as DrawCanvas, TicTacToe.

In TicTacToe class we have 2 enum ,a constructor and metdods.

**Metdods are :**

-mouseClicked(): mouse-clicked handler and it return type is void.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*@Override*

**public void mouseClicked(MouseEvent e) {** // mouse-clicked handler

int mouseX = e.getX();

int mouseY = e.getY();

// Get the row and column clicked

int rowSelected = mouseY / *CELL\_SIZE*;

int colSelected = mouseX / *CELL\_SIZE*;

if (currentState == *GameState*.*PLAYING*) {

if (rowSelected >= 0 && rowSelected < *ROWS* && colSelected >= 0

&& colSelected < *COLS* && board[rowSelected][colSelected] == *Seed*.*EMPTY*) {

board[rowSelected][colSelected] = currentPlayer; // Make a move

updateGame(currentPlayer, rowSelected, colSelected); // update state

// Switch player

currentPlayer = (currentPlayer == *Seed*.*CROSS*) ? *Seed*.*NOUGHT* : *Seed*.*CROSS*;

}

} else { // game over

initGame(); // restart the game

}

// Refresh the drawing canvas

repaint(); // Call-back paintComponent().

}

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-initGame(): Initialize the game-board contents and the status. It doesnt’t take any parameter and return type is void.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**void** **initGame**() {

**for** (**int** **row** = 0; row < ***ROWS***; ++row) {

**for** (**int** **col** = 0; col < ***COLS***; ++col) {

board[row][col] = *Seed*.***EMPTY***; // all cells empty

}

}

currentState = *GameState*.***PLAYING***; // ready to play

currentPlayer = *Seed*.***CROSS***; // cross plays first

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-updateGame(): it takes 3 parameters and return type is void.Update the currentState after the player with "theSeed" has placed on

(rowSelected, colSelected).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**void** **updateGame**(*Seed* theSeed, **int** rowSelected, **int** colSelected) {

**if** (hasWon(theSeed, rowSelected, colSelected)) { // check for win

currentState = (theSeed == *Seed*.***CROSS***) ? *GameState*.***CROSS\_WON*** : *GameState*.***NOUGHT\_WON***;

} **else** **if** (isDraw()) { // check for draw

currentState = *GameState*.***DRAW***;

}

// Otherwise, no change to current state (still GameState.PLAYING).

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-isDraw(): It doesn’t take any parameter and return type is boolean so it returns true if it is a draw (i.e., no more empty cell)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**boolean** **isDraw**() {

**for** (**int** **row** = 0; row < ***ROWS***; ++row) {

**for** (**int** **col** = 0; col < ***COLS***; ++col) {

**if** (board[row][col] == *Seed*.***EMPTY***) {

**return** **false**; // an empty cell found, not draw, exit

}

}

}

**return** **true**; // no more empty cell, it's a draw

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-hasWon(): It takes integer parameters such as :Seed theSeed, int rowSelected, int colSelected and it’s return type is boolean and return true if the player with "theSeed" has won after placing at(rowSelected, colSelected).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**boolean** **hasWon**(*Seed* theSeed, **int** rowSelected, **int** colSelected) {

**return** (board[rowSelected][0] == theSeed // 3-in-the-row

&& board[rowSelected][1] == theSeed

&& board[rowSelected][2] == theSeed

|| board[0][colSelected] == theSeed // 3-in-the-column

&& board[1][colSelected] == theSeed

&& board[2][colSelected] == theSeed

|| rowSelected == colSelected // 3-in-the-diagonal

&& board[0][0] == theSeed

&& board[1][1] == theSeed

&& board[2][2] == theSeed

|| rowSelected + colSelected == 2 // 3-in-the-opposite-diagonal

&& board[0][2] == theSeed

&& board[1][1] == theSeed

&& board[2][0] == theSeed);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In DrawCanvas class we have a method such as paintComponent which is taken a parameter and it doesn’t return anything.Inner class DrawCanvas (extends JPanel) used for custom graphics drawing.The Main method is in this class as well.

In our main method we call TicTacToe constructor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TicTacToe**(**int** playersNum) {

canvas = **new** DrawCanvas(); // Construct a drawing canvas (a JPanel)

canvas.setPreferredSize(**new** Dimension(***CANVAS\_WIDTH***, ***CANVAS\_HEIGHT***));

// The canvas (JPanel) fires a MouseEvent upon mouse-click

canvas.addMouseListener(**new** MouseAdapter() {

…

// Setup the status bar (JLabel) to display status message

statusBar = **new** JLabel(" ");

statusBar.setFont(**new** Font(**Font**.***DIALOG\_INPUT***, **Font**.***BOLD***, 15));

statusBar.setBorder(**BorderFactory**.*createEmptyBorder*(2, 5, 4, 5));

**Container** **cp** = getContentPane();

cp.setLayout(**new** BorderLayout());

cp.add(canvas, **BorderLayout**.***CENTER***);

cp.add(statusBar, **BorderLayout**.***PAGE\_END***); // same as SOUTH

setDefaultCloseOperation(**JFrame**.***EXIT\_ON\_CLOSE***);

pack(); // pack all the components in this JFrame

setTitle("Tic Tac Toe");

setVisible(**true**); // show this JFrame

board = **new** *Seed*[***ROWS***][***COLS***]; // allocate array

initGame(); // initialize the game board contents and game variables

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

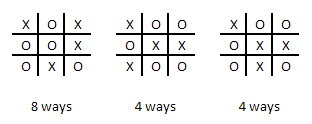
**Analysis**

The following table shows the probability of 0 to 9 X cards, and the probability of a tie given that number of Xs. The first column is the number of Xs out of 9. The second column is the probability of the given number of Xs on the card. The third column is the conditional probability of a tie, given the number of Xs for that row. The fourth column is the product of the second and third column. The lower right cell shows the probability of a tie is 6.3735%.

### Tie Probability in Tic Tac Toe

|  |  |  |  |
| --- | --- | --- | --- |
| **NUMBER XS** | **PROBABILITY** | **CONDITIONAL PROBABILITY OF A TIE** | **PRODUCT** |
| 0 | 0.001631 | 0 | 0 |
| 1 | 0.015898 | 0 | 0 |
| 2 | 0.067526 | 0 | 0 |
| 3 | 0.163991 | 0 | 0 |
| 4 | 0.250955 | 0.126984 | 0.031867 |
| 5 | 0.250955 | 0.126984 | 0.031867 |
| 6 | 0.163991 | 0 | 0 |
| 7 | 0.067526 | 0 | 0 |
| 8 | 0.015898 | 0 | 0 |
| 9 | 0.001631 | 0 | 0 |
| Total | 1.000000 |  | 0.063735 |

If there are 4 Xs and 5 Os, then there are [combin](http://wizardofodds.com/gambling/glossary/#combin)(9,4)=126 ways to place the Xs and Os on the board. Of those 126, 16 result in a tie. The image below shows the three tie patterns, and the number of ways each can be rotated or reflected. So the probability of a tie, given 4 or 5 Xs, is 16/126 = 12.70%.



Once it can be accepted that the probability of a tie is 6.3735%, the rest is easy. The probability that there is not a tie is 100%-6.3735% = 93.6265%. The probability of X or O winning is each half of that, or 46.8133%.

The following table shows the probability and contribution to the return of each outcome of the X and O bets. The lower right cell shows a house edge of 3.19%.

### X/O Bet Return — Four Decks

|  |  |  |  |
| --- | --- | --- | --- |
| **EVENT** | **PROBABILITY** | **PAYS** | **RETURN** |
| Win | 0.468133 | 1 | 0.468133 |
| Tie | 0.063735 | -0.5 | -0.031867 |
| Loss | 0.468133 | -1 | -0.468133 |
| Total | 1.000000 |  | -0.031867 |

The following table shows the probability and contribution to the return of each outcome of the Tie bet. The lower right cell shows a house edge of 4.40%.

### Tie Bet Return — Four Decks

|  |  |  |  |
| --- | --- | --- | --- |
| **EVENT** | **PROBABILITY** | **PAYS** | **RETURN** |
| Win | 0.063735 | 14 | 0.892286 |
| Loss | 0.936265 | -1 | -0.936265 |
| Total | 1.000000 |  | -0.043980 |