Tic-Tac-Toe Part 2

Contents

Time required: 90 minutes

Tic-Tac-Toe

A classic programming problem.

Videos

If you are having trouble with this assignment, or want to see another solution:

- Tic-Tac-Toe Part 1 in Matlab

Step 1: Game Loop

All games require an infinite game loop. This loop keeps looking for player interaction.

1. Create a **while true** loop around the main game code. (Not around the functions.)
2. Add an **if** statement right after the player input.
3. If the player enters 0, the game exits the loop with the **break** statement.

Step 2: Is This a Valid Move?

A valid move can only happen if the move is available. This is indicated by a 0. 1 indicates the player, 2 indicates the computer.

1. Create a function named **isValidMove**.
2. **isValidMove** takes the current player's move and the current board matrix as arguments.
3. Use the same if decision statement in the **updateBoard** function to convert the number to row, col index.
4. Compare the move to the current board matrix location. Is it 0?
5. Return a Boolean that indicates true or false. Either the gameboard cell has a 0, or it doesn't.
6. In the main part of the program, call the **isValidMove** function right after the if statement that tests for a 0 to exit the game.
7. **isValidMove** returns a Boolean value of true or false. Use an if statement using the **isValidMove** function as an expression to decide:
    a. true: Make the move with **updateBoard.**
    b. false: Display "Invalid Move, please try again". Use the **continue** keyword to skip the loop and back to the beginning of the loop.

Step 3: Computer's Turn

The computer will make a random move on any available move which is indicated by a 0 in the matrix.

1. Create a function named: **getComputerMove()**
2. **getComputerMove()** takes the current board matrix and returns a move.
3. Use the MATLAB **find()** function to return a vector named **availableMoves** where the board elements equal 0.

```
availableMoves = find(board == 0)
```

4. Get the length of **availableMoves** moves and assign this to **lenMove**
5. Use **randi** to get a random integer from the **lenMove** variable and assign this to **randomMove.**
6. Get the **linearIndex** of the move from the **availableMoves** vector.

```
% 1-9 position of the move as an index
linearIndex = availableMoves(randomIndex);

% Convert linear index to row and column
[row, col] = ind2sub(size(board), linearIndex);
```

7. Use an if esleif structure to determine which move this row and column is. The sample code below will get you started.

```
% Convert row and column to position number (1-9) using if statements
if row == 1 && col == 1
    move = 1;
elseif row == 1 && col == 2
    move = 2;
```

8. The **move** variable will be returned.

Example run:

```
Do you want to play a game?
Let's play Tic Tac Toe
Can you beat the computer?
     0     0     0
     0     0     0
     0     0     0

Enter your move (1-9) (0 to exit): 1
Computer moves:
     1     0     0
     0     0     0
     0     0     2

Enter your move (1-9) (0 to exit): 1
Invalid move, please try again.
Enter your move (1-9) (0 to exit): 2
Computer moves:
     1     0     0
     1     0     0
     2     0     2

Enter your move (1-9) (0 to exit): 0
Thanks for playing Tic Tac Toe
```

Assignment Submission

1. Submit properly named and commented script files.
2. Attach a screenshot of the Command Window showing the successful execution of each script.
3. Attach all to the assignment in Blackboard.