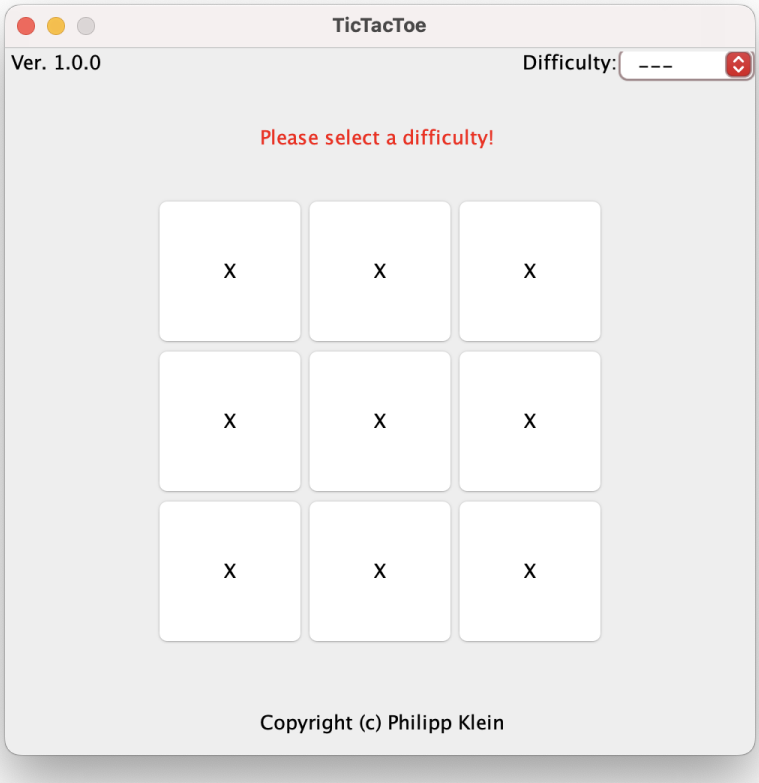
# TicTacToe

written by Philipp Klein (phi.klein17@gmail.com)
Version 1.0.0

# Contents

# 1 How to use the software

When you first launch the application, you'll find yourself on this page:




On the top right, you can select the level of difficulty, and how hard the computer will place on the game board. If currently no difficulty is selected, you'll see '- - -' in the selection field. In that state, you won't be able to click any button and start the game. At the end of each game, a reset button will appear at the bottom of the board.

If you have selected a difficulty, just press any field button, making your first placement on the board and officially starting the game. During the game, you won't be able to change the difficulty!

## 2 How does the software work?

The function of the software will be explained with a custom game. For the computer to orientate on the board, movement variables are used to track the progress of the game:

```
Horizontal lines :  a_horizontal,b_horizontal,c_horizontal
Vertical lines   :  a1_vertical,a2_vertical,a3_vertical
Cross lines      :  a1_cross,c1_cross
```

These get updated, every time the player or computer makes a move. The player increases and the computer decreases the movement variables with each move. If one or more variables are $= -2$ the computer will be able to win in the next move. If one or more variables is $= 2$ the player can win in the next move. Here's an example game with the difficulty 'Hard':

Assuming we make our first move in the top left corner:

| | | |
|---|---|---|
| X | | |
| | | |
| | | |

```
a_horizontal = a1_vertical = a1_cross = 1.
```

Only when the difficulty is set to 'Hard' the computer will use the method `firstMove()`. This method tries to limit the player to the best at the beginning of the game. Since we've made a placement in the middle, the computer will counter, by making it's move in the middle of the field.

| | | |
|---|---|---|
| X | | |
| | O | |
| | | |

```
b_horizontal = a2_vertical = c1_cross = -1.
a_horizontal = a1_vertical = 1.
a1_cross = 0.
```

If we had placed e.g. on $A2$, the computer would counter by making a move on either the left or right side of $A2(A1, A3)$. To conclude, the `firstMove()` method places the move in the middle when we make a move in a corner, and if we, e.g., place it in the middle of the board or in the middle of one of the outer borders of the board `(a_horizontal, a1_vertical, a3_vertical, c_horizontal)`, the computer will make the move in a corner, directly next to the move. Now, our next move is placed on b1:

| | | |
|---|---|---|
| X | | |
| X | O | |
| | | |

```
c1_cross = a2_vertical = -1.
b_horizontal = a1_cross = 0.
a1_vertical = 2 (!!!)
a_horizontal = 1.
```

The computer now detects via the method `opponentPossibleWinInterfere()` that the player can win on the next move ($C1$). So it makes its placement at $C1$:

| | | |
|---|---|---|
| X | | |
| X | O | |
| O | | |

```
a_horizontal = c_horizontal = a1_vertical  = 1.
b_horizontal = a1_cross = 0.
c1_cross = -2 (!!!)
a2_vertical = -1.
```

As you can see, `c1_cross = -2`, so the computer can win in the next move ($A3$). But first, we block this opportunity by making our placement on $A3$.

| | | |
|---|---|---|
| X | | X |
| X | O | |
| O | | |

```
a1_vertical = c_horizontal = a3_vertical = 1.
b_horizontal = a1_cross = 0.
c1_cross = a2_vertical = -1.
a_horizontal = 2 (!!!)
```

The computer detects `a_horizontal = 2`, so he blocks it by making its move on $A2$:

| | | |
|---|---|---|
| X | O | X |
| X | O | |
| O | | |

```
a_horizontal = c_horizontal  = a1_vertical = a3_vertical = 1.
b_horizontal = a1_cross = 0.
a2_vertical = -2 (!!!)
c1_cross = -1.
```

Of course, we won't let the computer win, so we'll make our placement on $C2$:

| | | |
|---|---|---|
| X | O | X |
| X | O | |
| O | X | |

```
a_horizontal = c_horizontal  = a1_vertical = a3_vertical = 1.
b_horizontal = a1_cross = c_horizontal = 0.
a2_vertical = c1_cross = -1
```

Since there's no direct danger for the computer, he'll use the `limitPlayerMovement()` method, which checks if a corner is free, to limit the player's movements. If no corner is available, the computer calls the `makeRandomMove()` method. In our case, the method `limitPlayerMovement()` chooses the bottom left corner ($C3$):

| | | |
|---|---|---|
| X | O | X |
| X | O | |
| O | X | O |

The only move we have left, is the field $B3$:

| | | |
|---|---|---|
| X | O | X |
| X | O | X |
| O | X | O |

A tie!