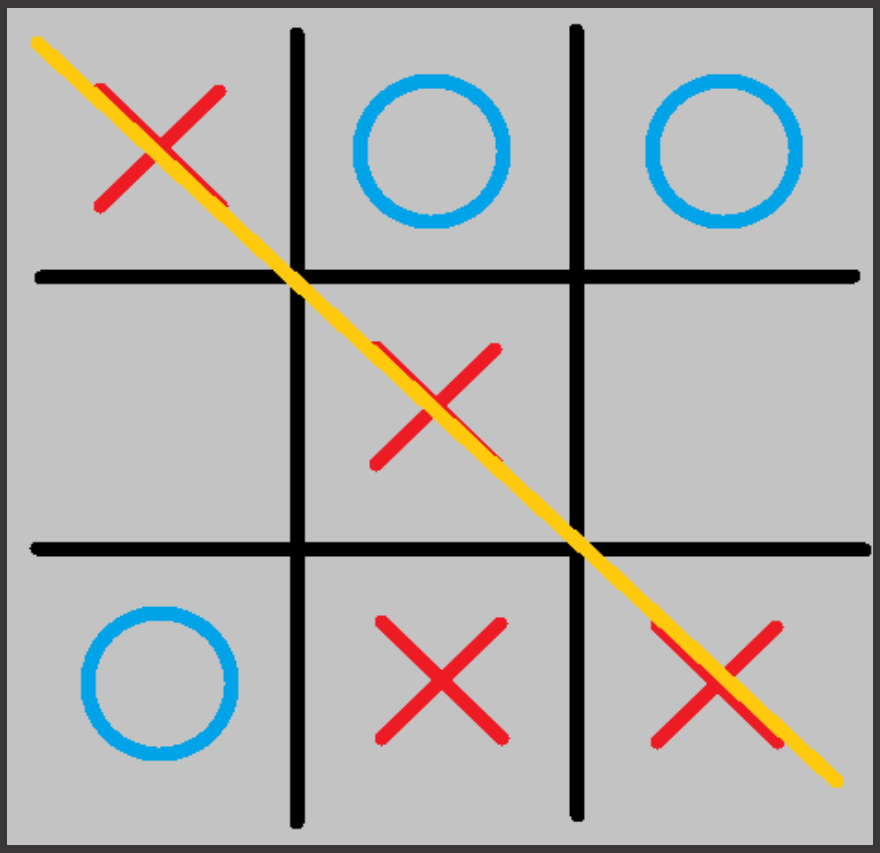
Tic-Tac-Toe



Programmer Name: *Noam Erlich*

ID: **********

Teacher: *Anatoly Peymer*

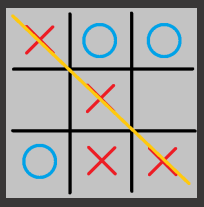
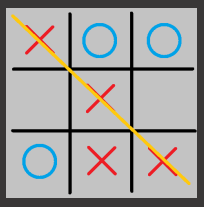


Table of Contents:



Introduction

Game: *Tic-Tac-Toe\XO*

File Name: *XO.Asm*

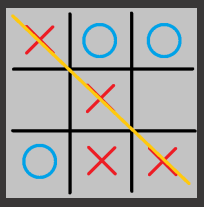
WorkSpace: *Turbo Assembler*

Develop Space: *NotePad++*

Running Space: *DosBox*

Project Topic

+ instructions



The game I developed is the famous mini game “Tic-Tac-Toe” or in is second name “XO”.

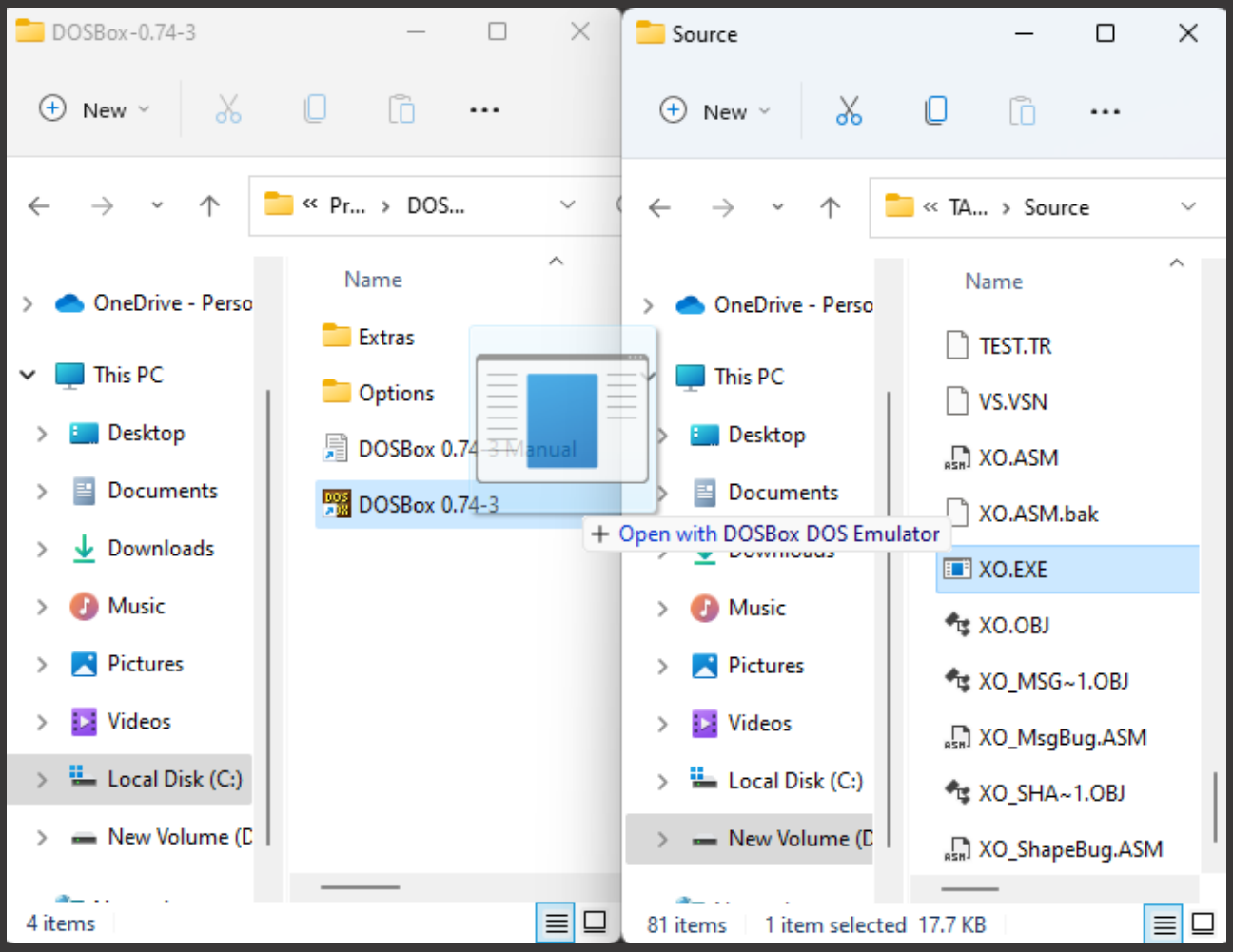
In this game 2 players are competing on a 3x3 grid line board, every round, one player needs to set his shape, “X”, or “O”, in one of the free areas on the board, the starting player is the player who choose “X”.

first player to set 3 of his shape in a row, column, or diagonal line wins.

Manual Guide

*To play the game, you will need to download the software **DOSBOX**,*

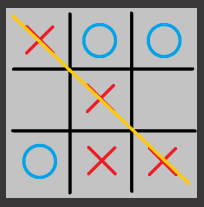
and then drag the EXE file to the DOSBOX application icon.



Menu Structure

When opening the game, the menu with the three options will show up,

1) *Play*



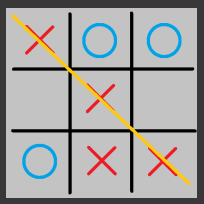
2) *Help*
3) *Exit*

To Play the game, press 1 on the keyboard.

To have more information about the game and how to play it, press 2 on the keyboard, to go inside Help category.

To Exit the game, press 3 on the keyboard.

Game Version



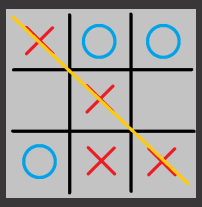
*This is the **final** and **only** game version to my game. I'm not going to update it in the close future.*

<u>This version contains 2 game modes:</u>

*1) **PvP** – Player vs Player*

*2) **PvE** – Player vs Computer*

It also contains the ability to choose the shape you want, "X" or "O", the first player to start is "X", so you have the ability to choose the starter player and not only the shape.

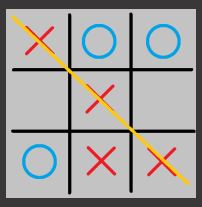


The Solution Explanation

To make the game works I needed to create logic.

Logic- *The logic is built in a structure, first of all, I need to print the menu, then check the player choice (the key bind he pressed), then print the other 2 menus and check in both of them the player choice. This is a very small part of the logic.*

The bigger part of the logic is in all of the procedures, their job, is to print the board, print the turn, change the turn, checks who wins in all the winning

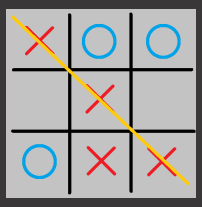


options, it also contains the logic of all the game modes.

(More about the procedures and their logic in page 37-41)

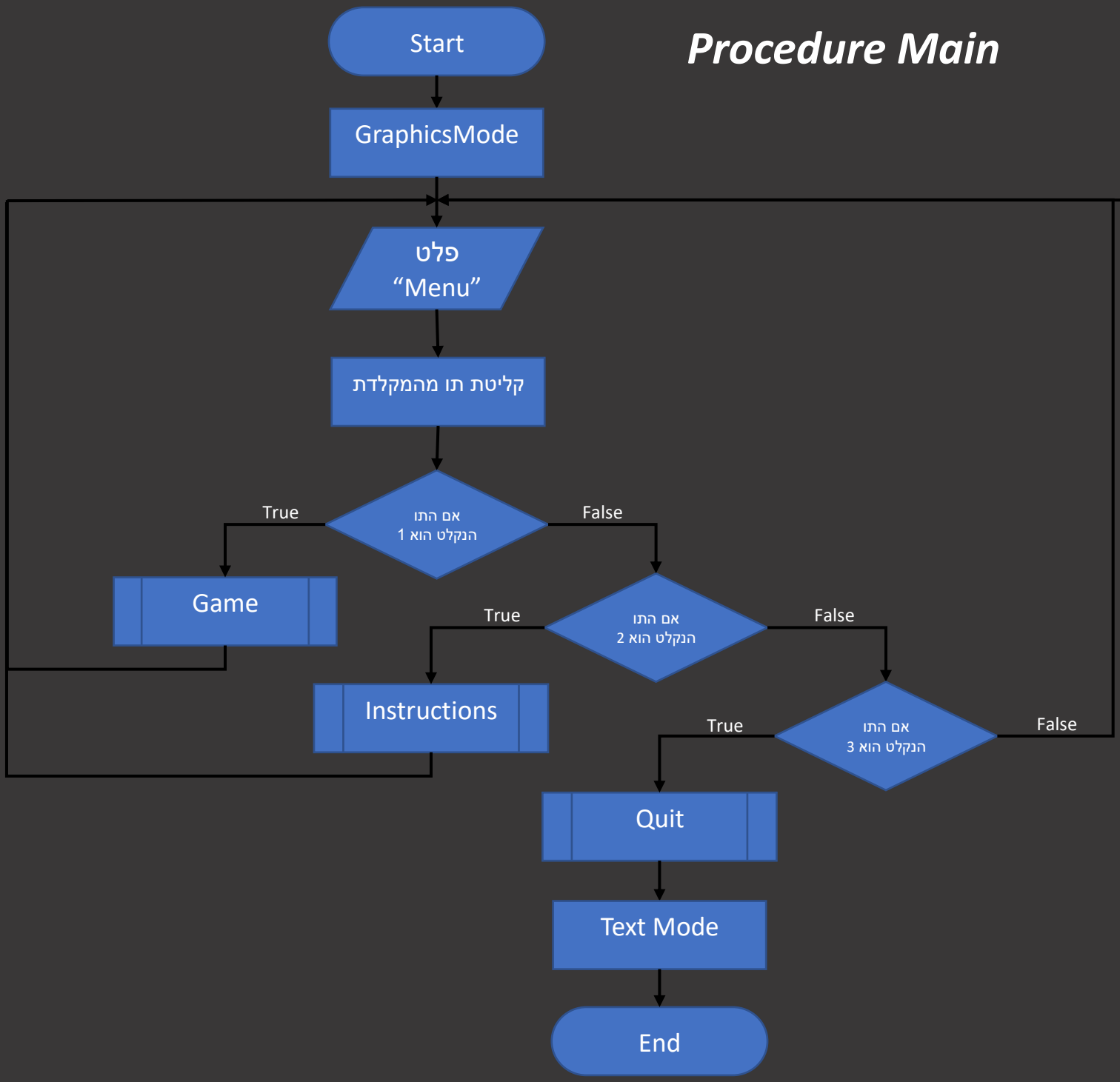
Variables- *The Variables are storage in bytes and words.*

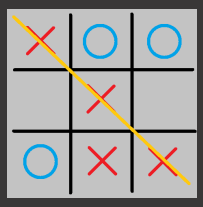
Graphics- *I'm not using graphics in my code, only text and chars, but I am using graphics mode, because the text font is bigger and better for my opinion in graphics mode.*



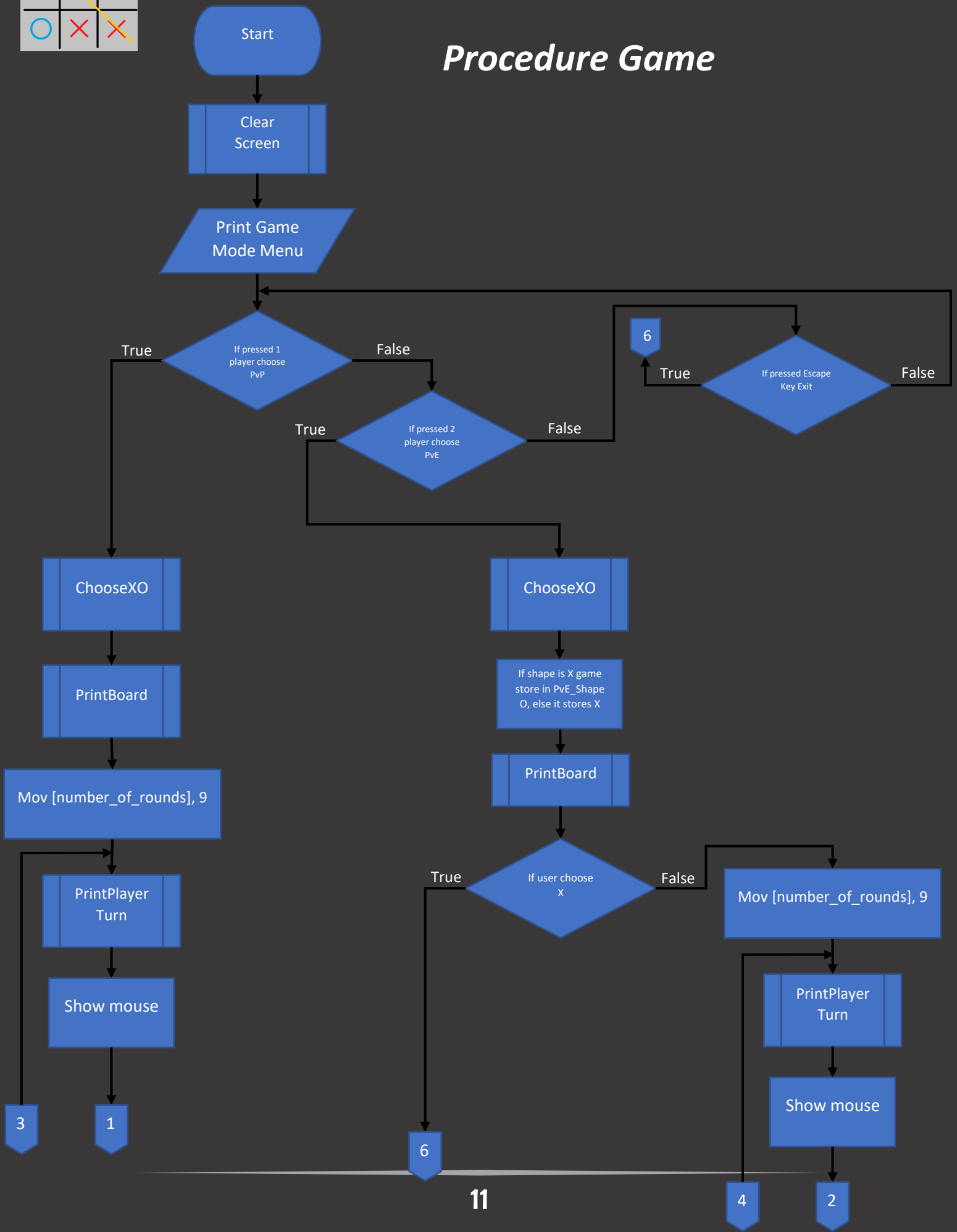
Flow Charts

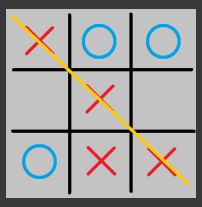
Procedure Main

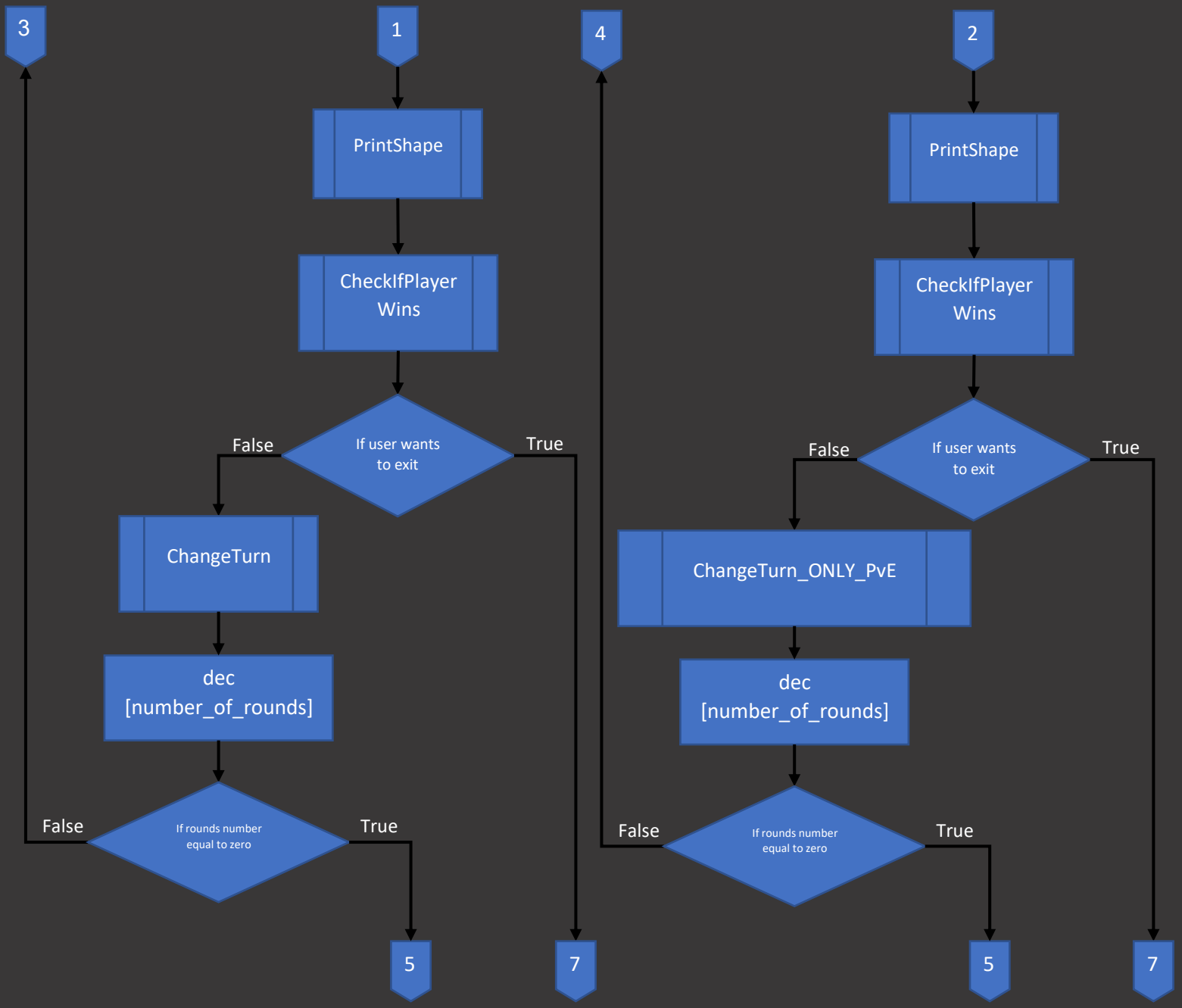


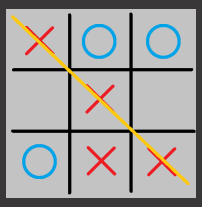


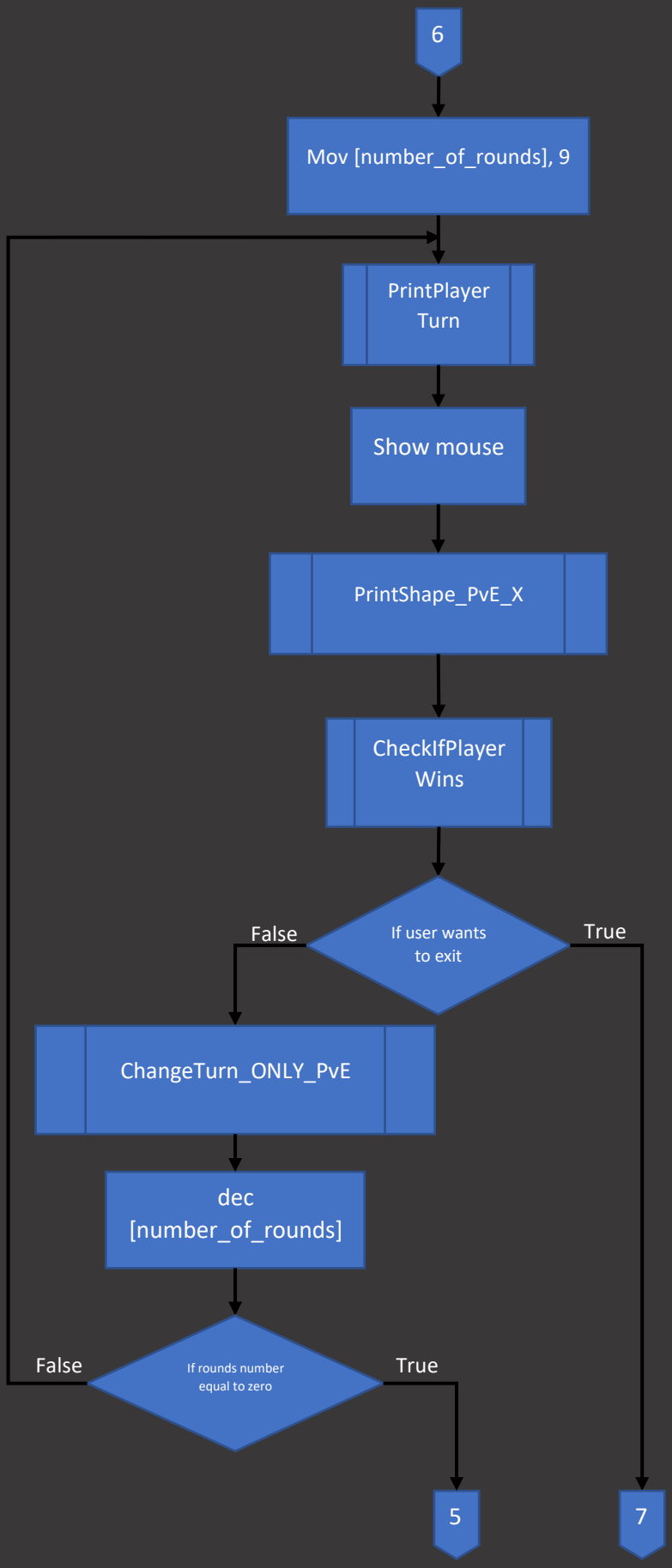
Procedure Game

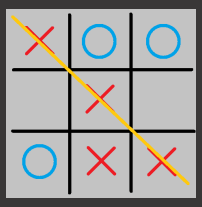


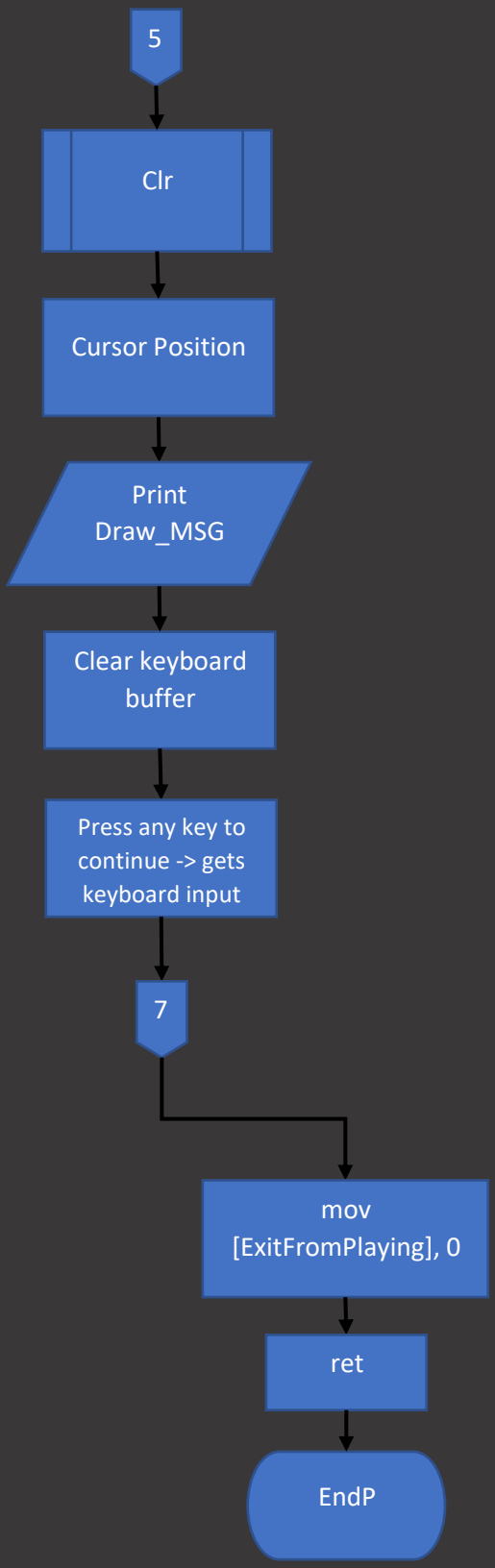


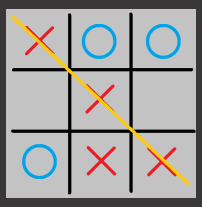

3
1
4
2
PrintShape
CheckIfPlayer Wins
If user wants to exit
False
True
ChangeTurn
dec [number_of_rounds]
If rounds number equal to zero
False
True
5
7
PrintShape
CheckIfPlayer Wins
If user wants to exit
False
True
ChangeTurn_ONLY_PvE
dec [number_of_rounds]
If rounds number equal to zero
False
True
5
7



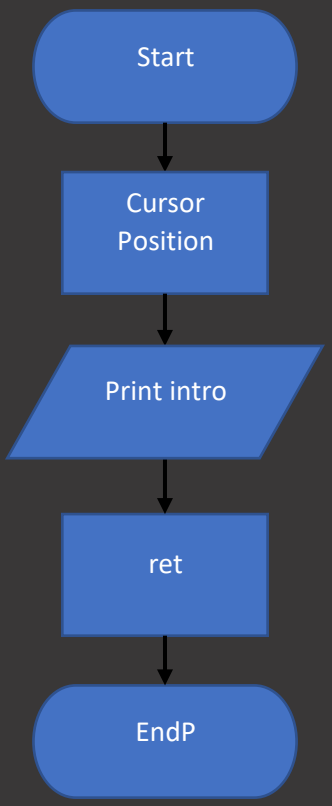

6
Mov [number_of_rounds], 9
PrintPlayer Turn
Show mouse
PrintShape_PvE_X
CheckIfPlayer Wins
If user wants to exit
False
True
ChangeTurn_ONLY_PvE
dec [number_of_rounds]
If rounds number equal to zero
False
True
5
7

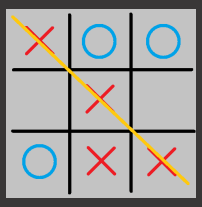



5
Clr
Cursor Position
Print
Draw_MSG
Clear keyboard
buffer
Press any key to
continue -> gets
keyboard input
7
mov
[ExitFromPlaying], 0
ret
EndP

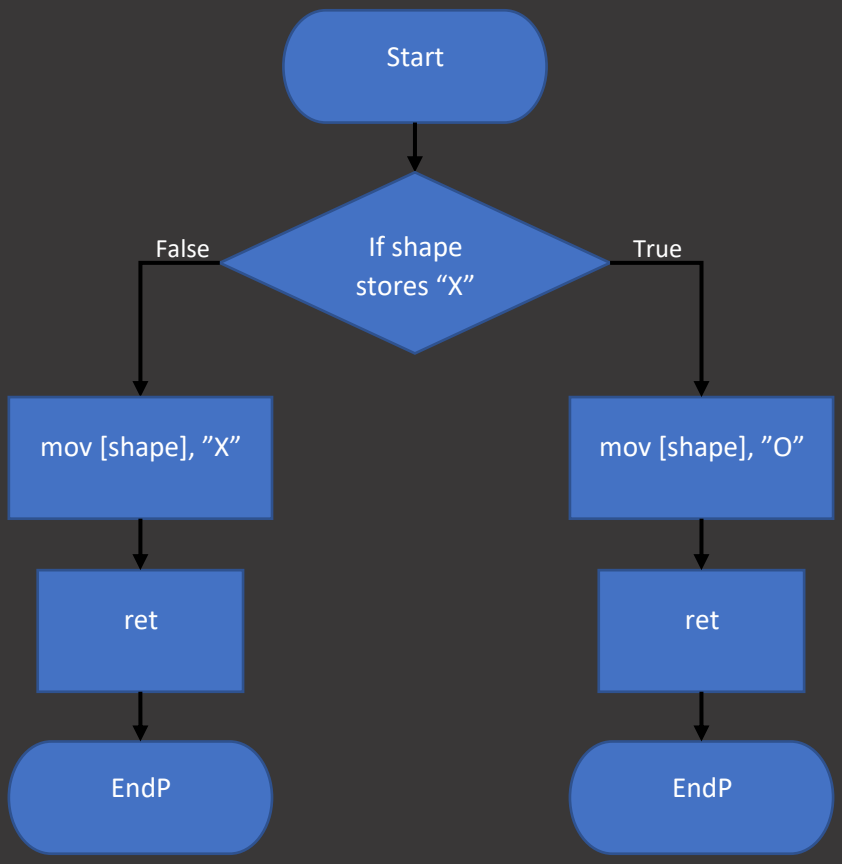


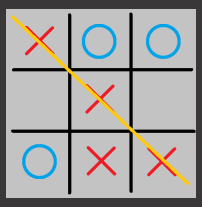
Procedure Menu



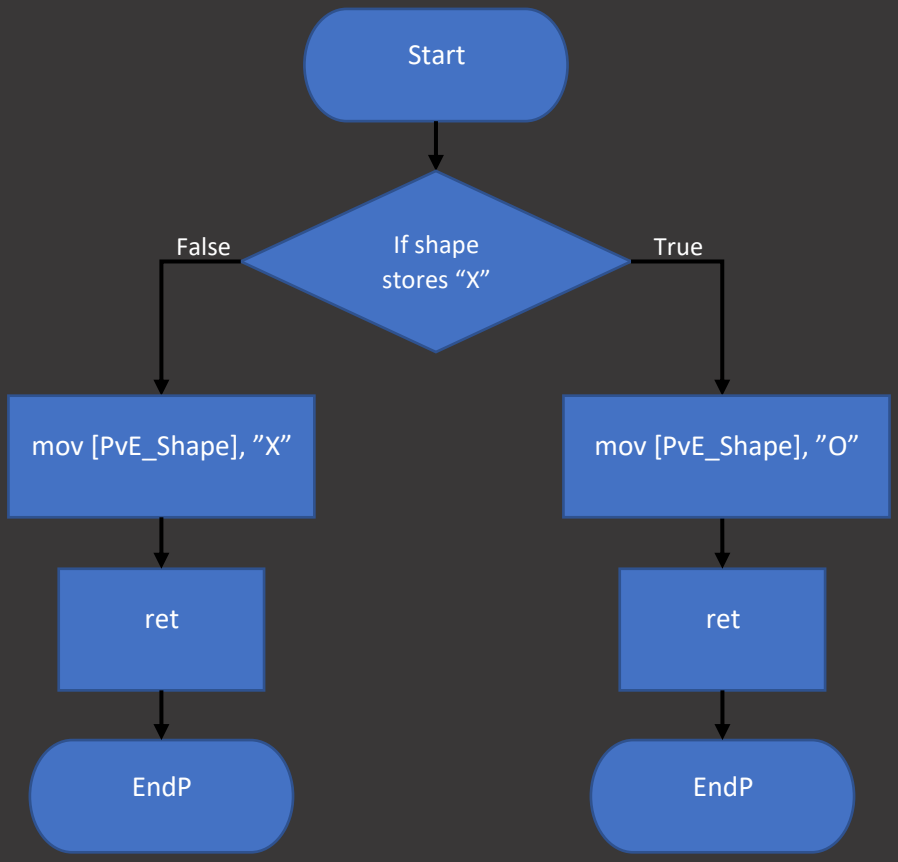


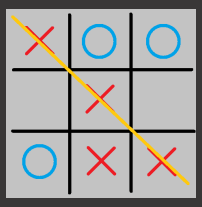
Procedure ChangeTurn



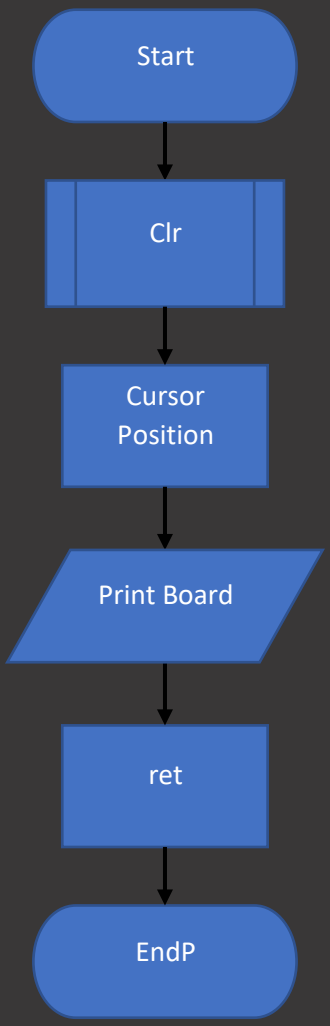


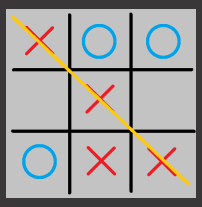
Procedure ChangeTurnONLY_PvE



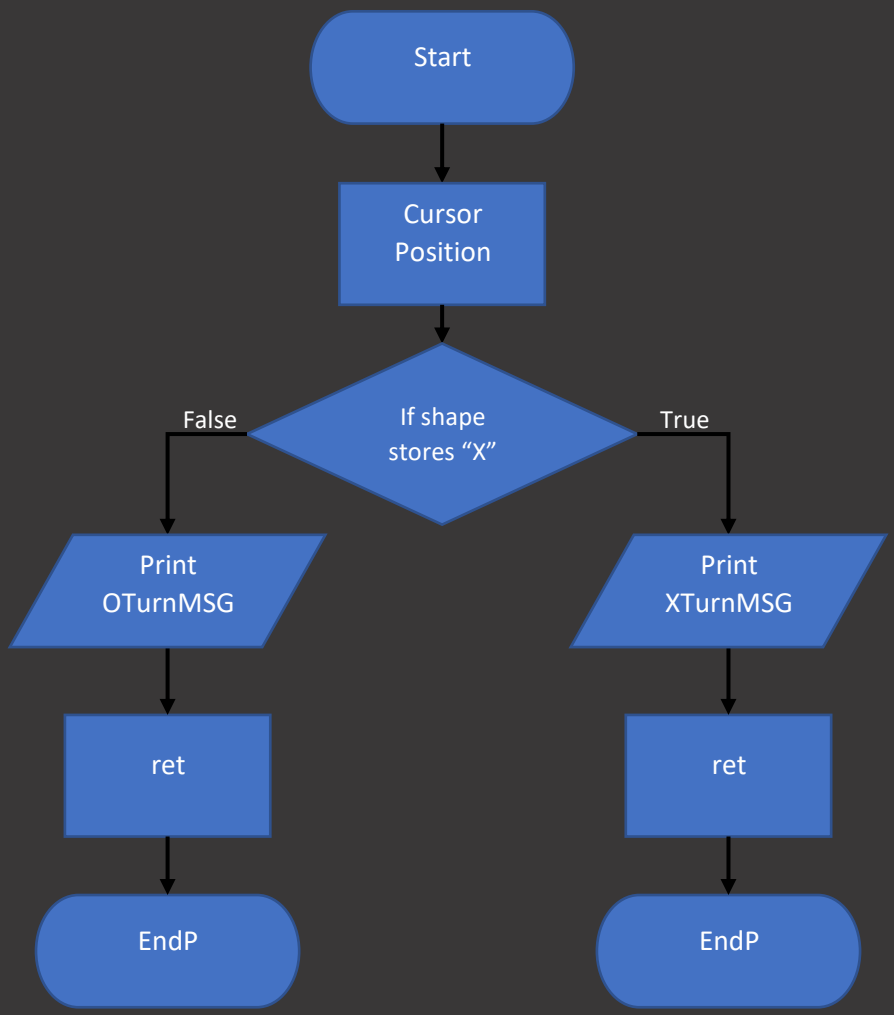


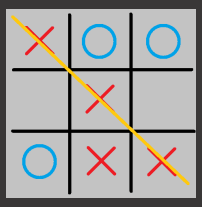
Procedure PrintBoard



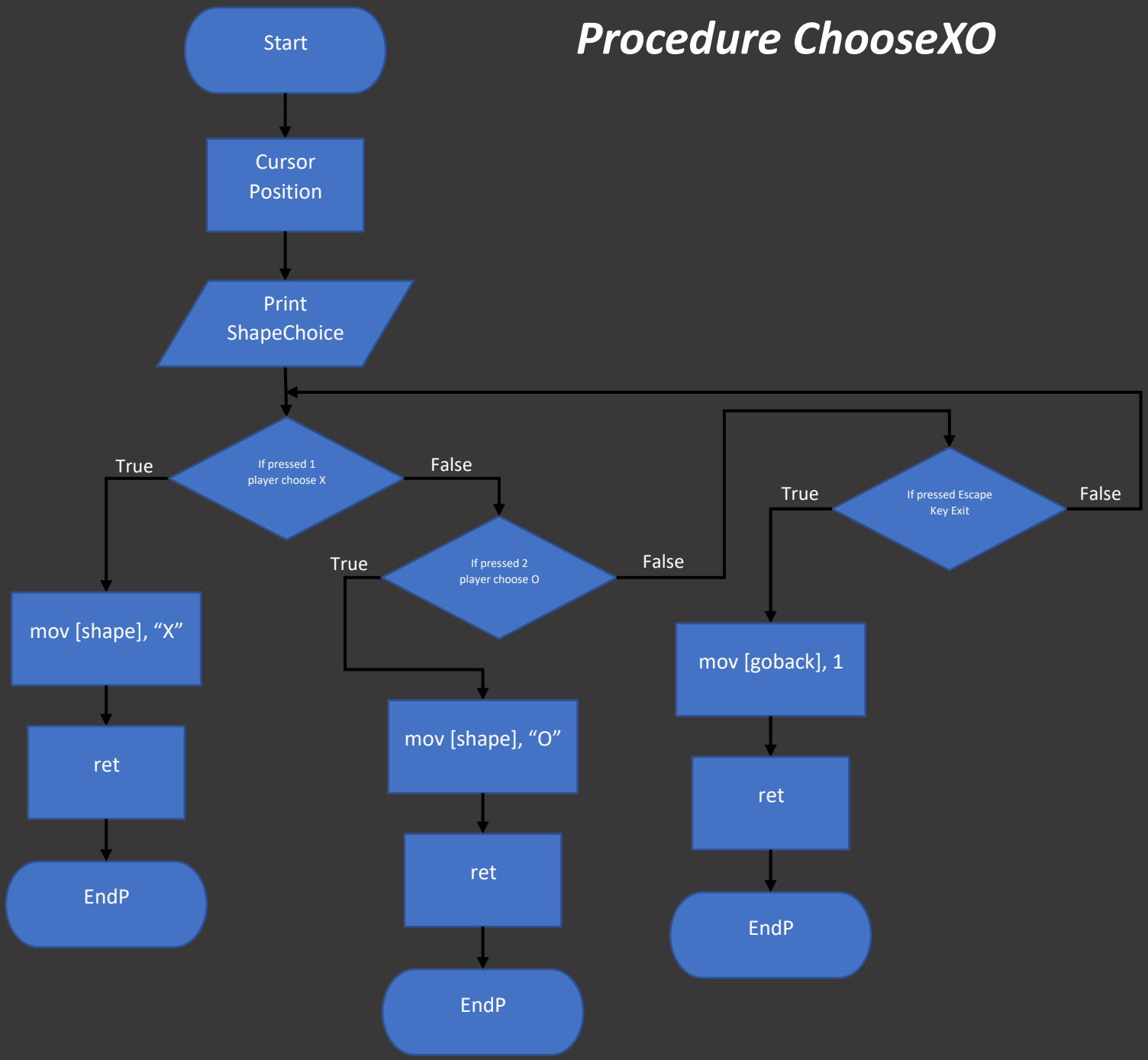


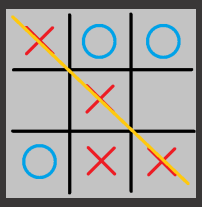
Procedure PrintPlayerTurn



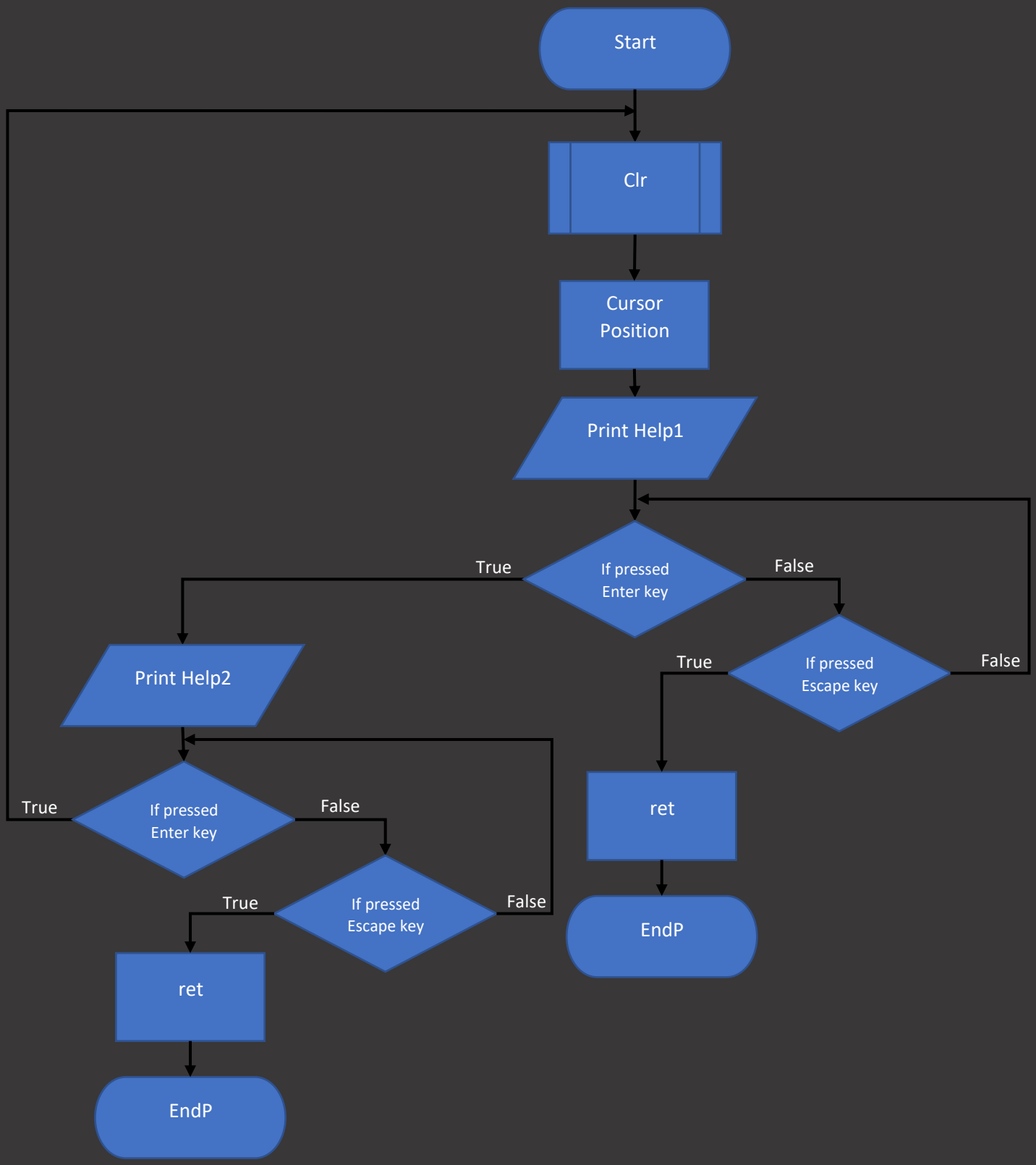


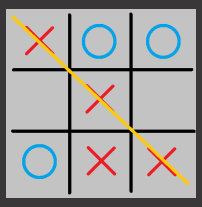
Procedure ChooseXO



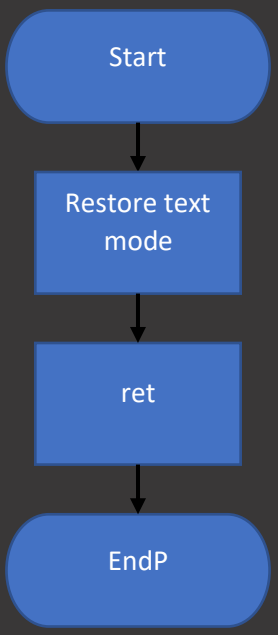


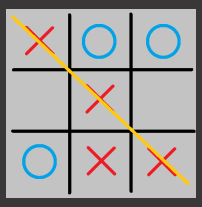
Procedure Instructions



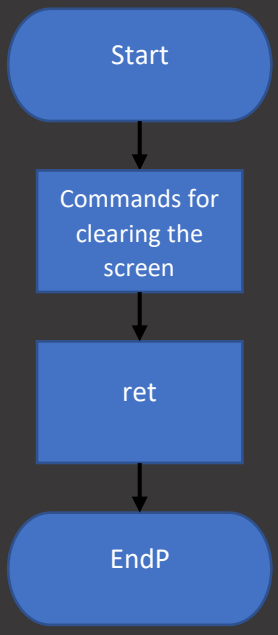


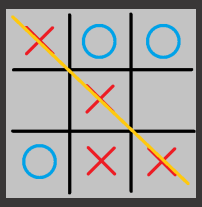
Procedure Quit



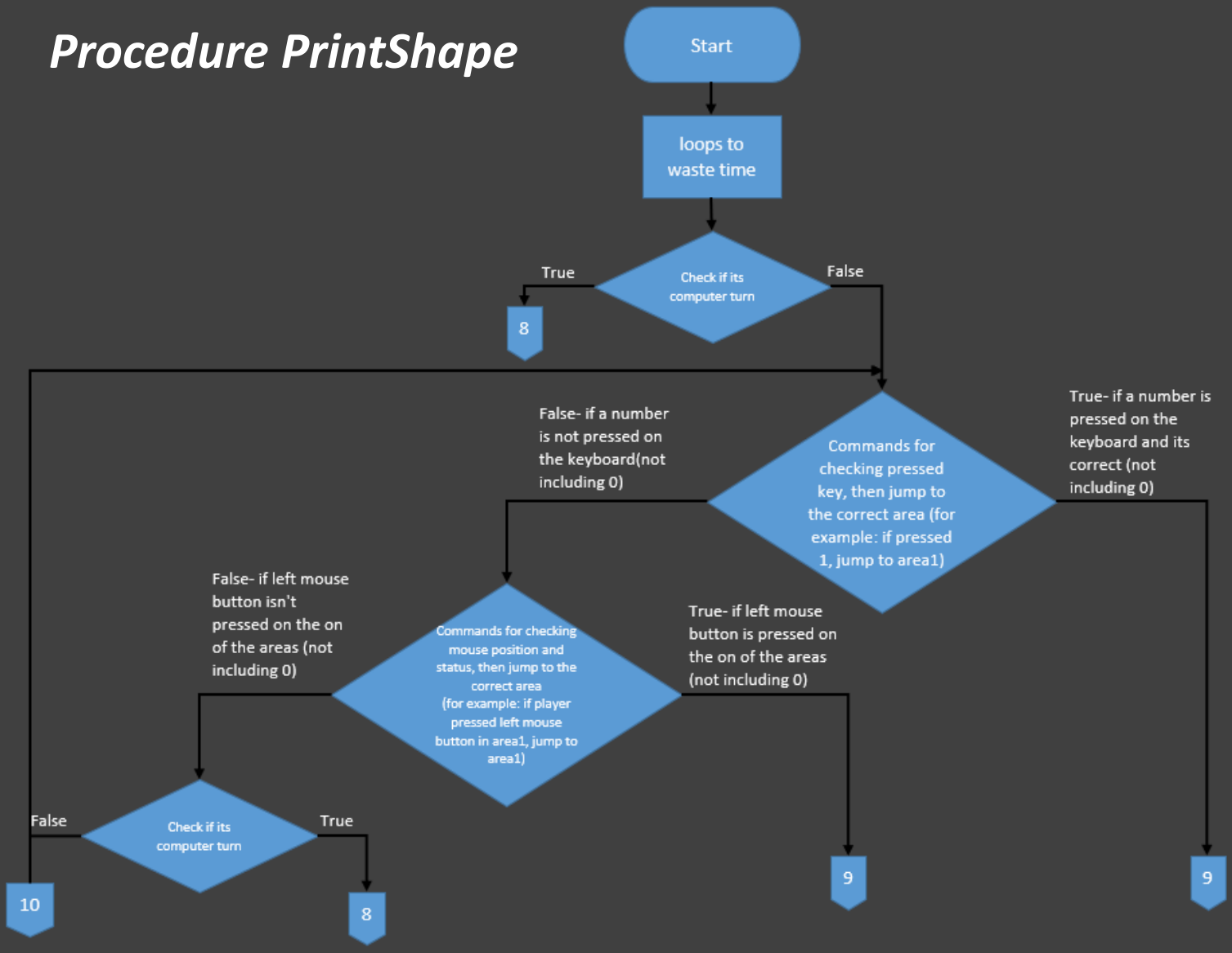


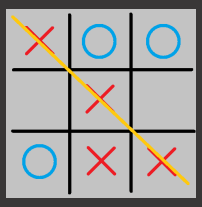
Procedure Clr

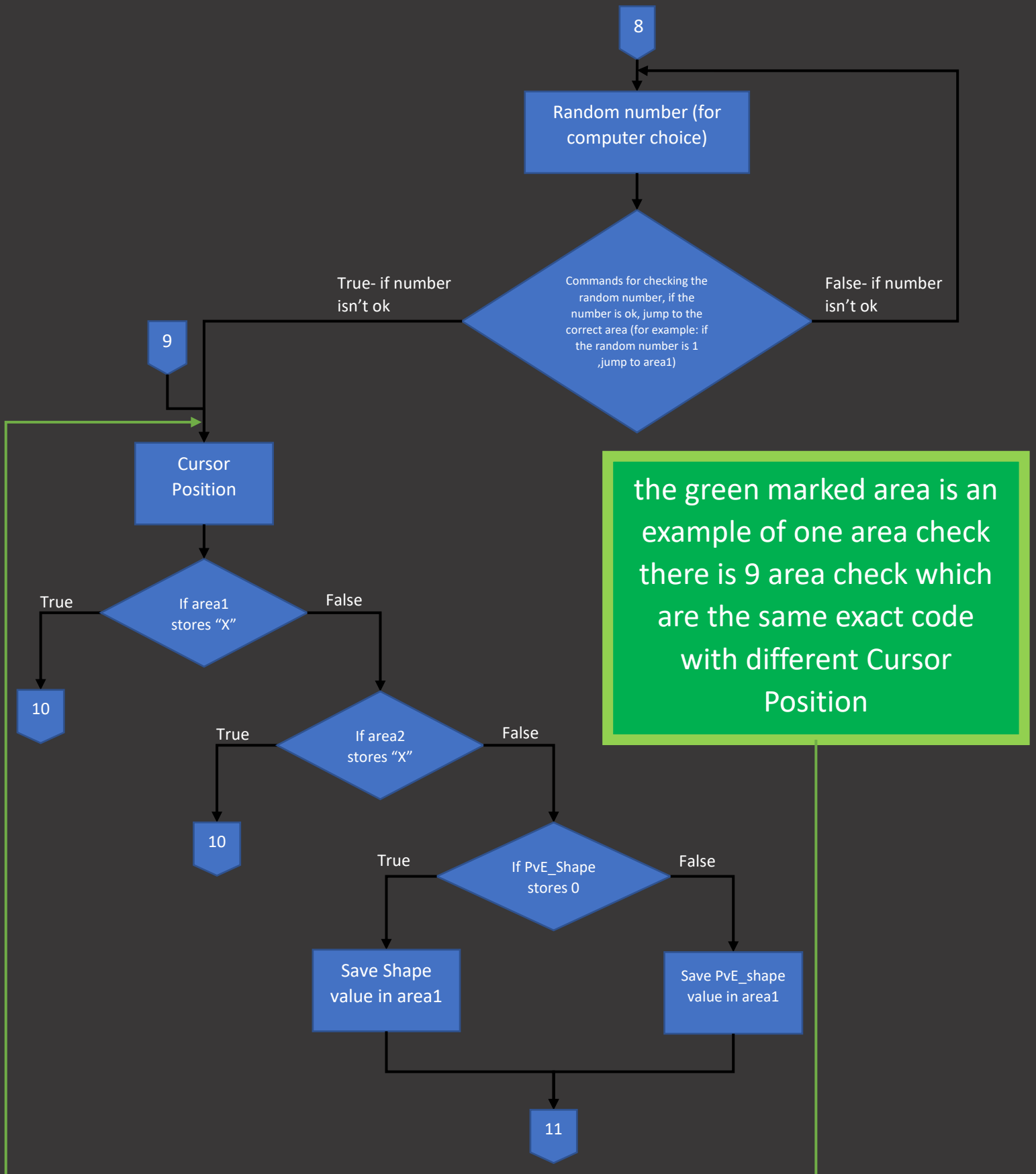


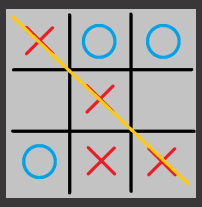


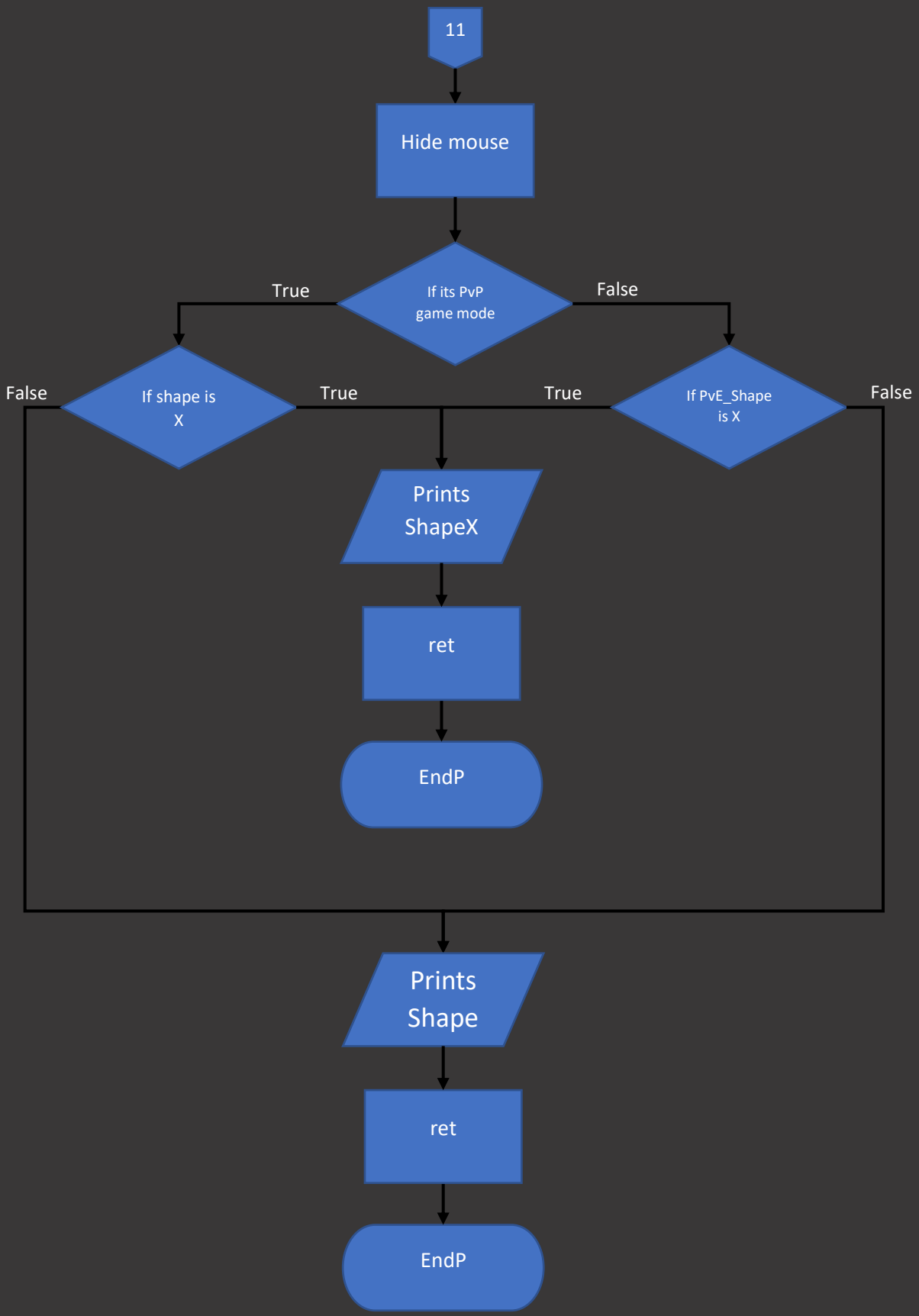
Procedure PrintShape



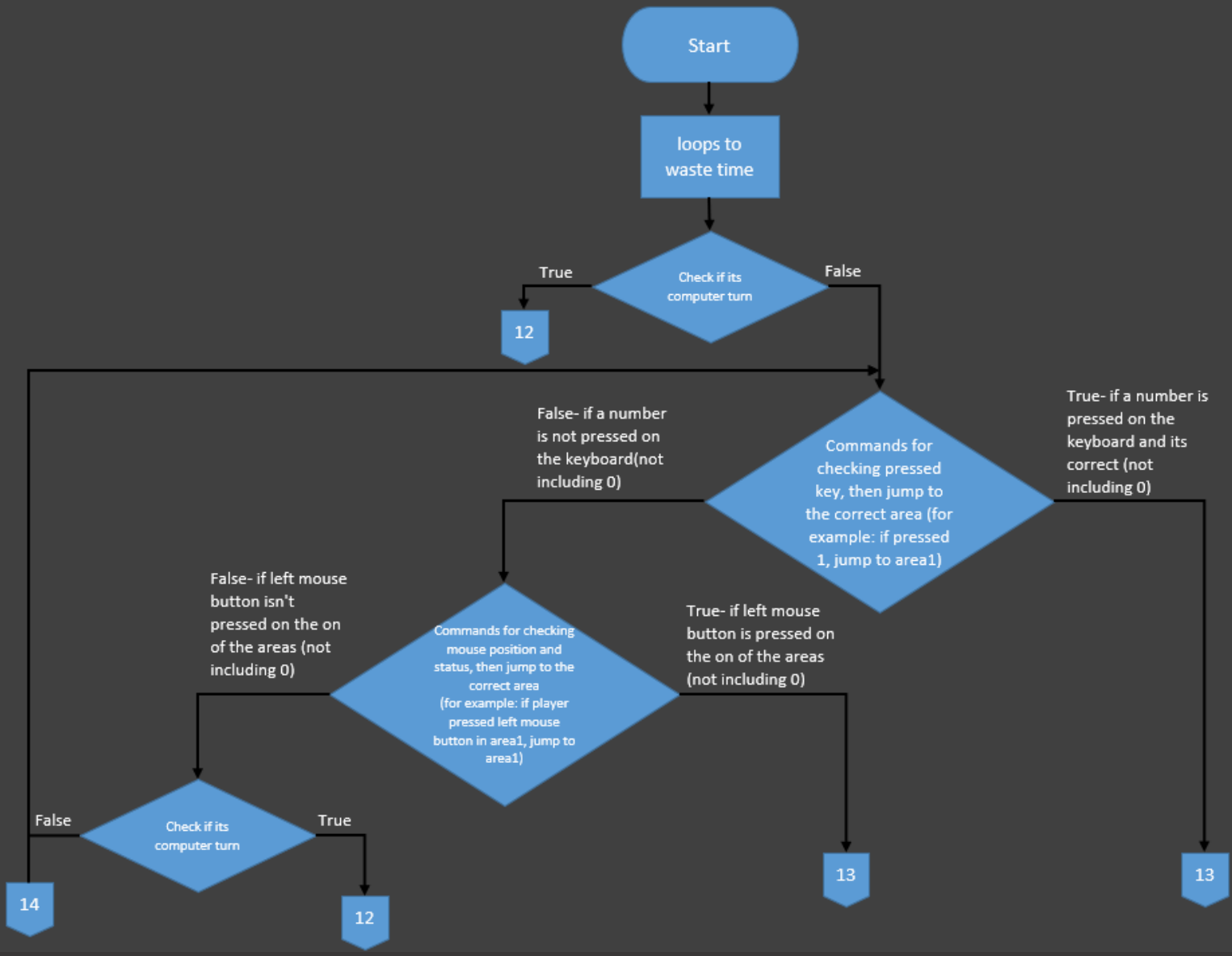


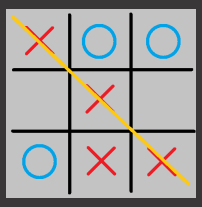

8
Random number (for computer choice)
Commands for checking the random number, if the number is ok, jump to the correct area (for example: if the random number is 1 ,jump to area1)
True- if number isn't ok
False- if number isn't ok
9
Cursor Position
If area1 stores "X"
True
False
10
If area2 stores "X"
True
False
10
If PvE_Shape stores 0
True
False
Save Shape value in area1
Save PvE_shape value in area1
11
the green marked area is an example of one area check there is 9 area check which are the same exact code with different Cursor Position

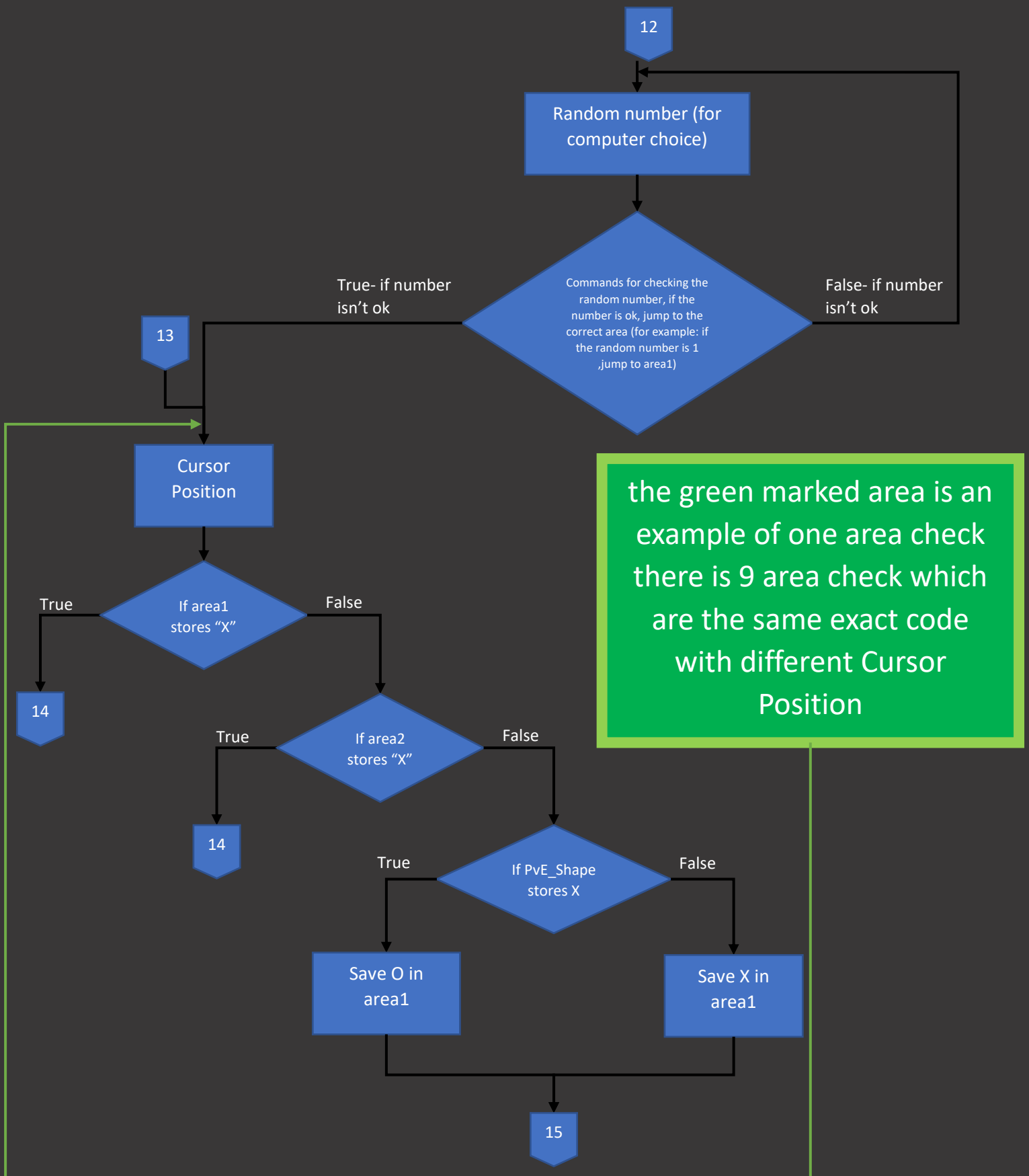



11
Hide mouse
If its PvP game mode
True
False
If shape is X
False
True
True
If PvE_Shape is X
False
Prints ShapeX
ret
EndP
Prints Shape
ret
EndP

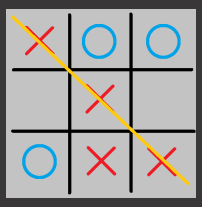
Procedure PrintShape_PvE_X

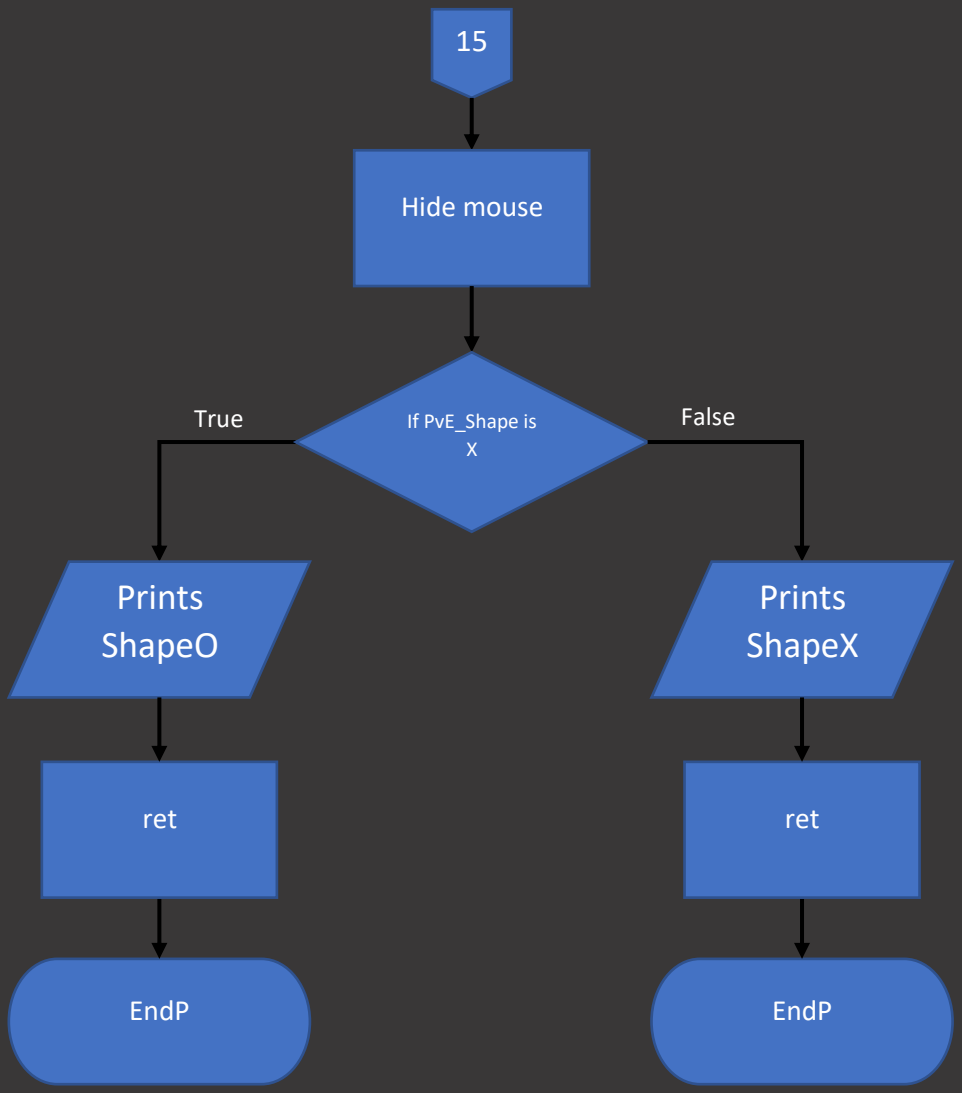


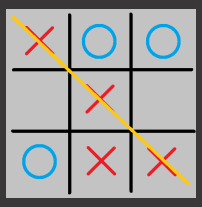




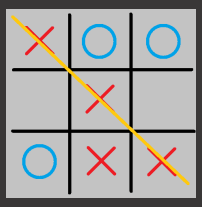
the green marked area is an example of one area check there is 9 area check which are the same exact code with different Cursor Position



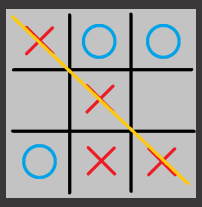

15
Hide mouse
If PvE_Shape is X
True
False
Prints ShapeO
Prints ShapeX
ret
ret
EndP
EndP

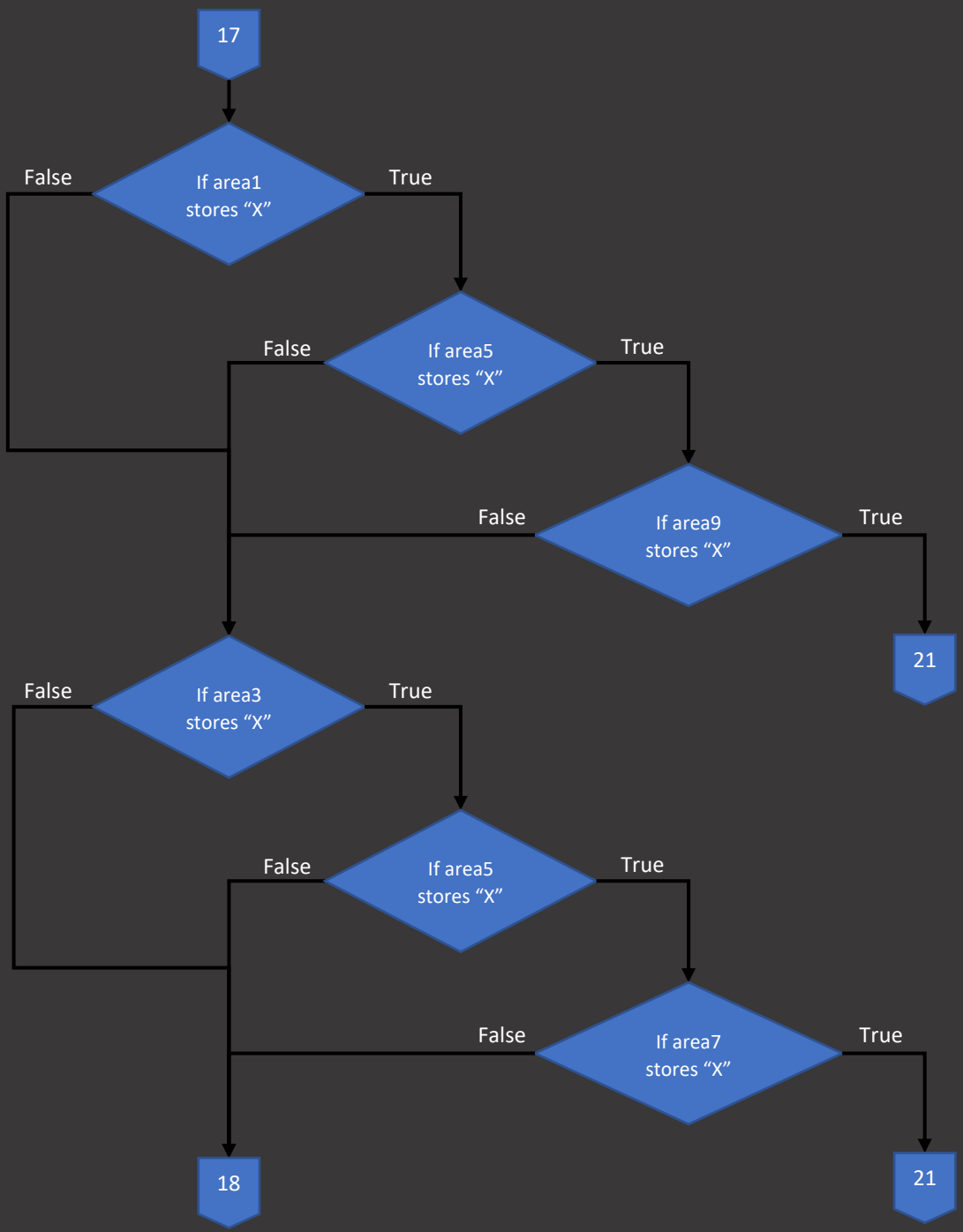


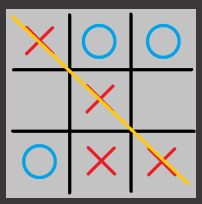
Procedure CheckIfPlayerWins



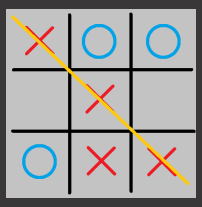

16
If area1 stores “X”
False
True
If area4 stores “X”
False
True
If area7 stores “X”
False
True
21
If area2 stores “X”
False
True
If area5 stores “X”
False
True
If area8 stores “X”
False
True
21
If area3 stores “X”
False
True
If area6 stores “X”
False
True
If area9 stores “X”
False
True
17
21



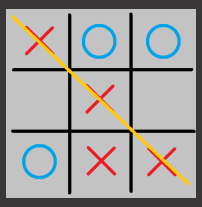

17
If area1 stores “X”
False
True
If area5 stores “X”
False
True
If area9 stores “X”
False
True
21
If area3 stores “X”
False
True
If area5 stores “X”
False
True
If area7 stores “X”
False
True
18
21



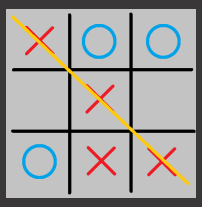

18
If area1 stores “O”
False
True
If area2 stores “O”
False
True
If area3 stores “O”
False
True
22
If area4 stores “O”
False
True
If area5 stores “O”
False
True
If area6 stores “O”
False
True
22
If area7 stores “O”
False
True
If area8 stores “O”
False
True
If area9 stores “O”
False
True
19
22

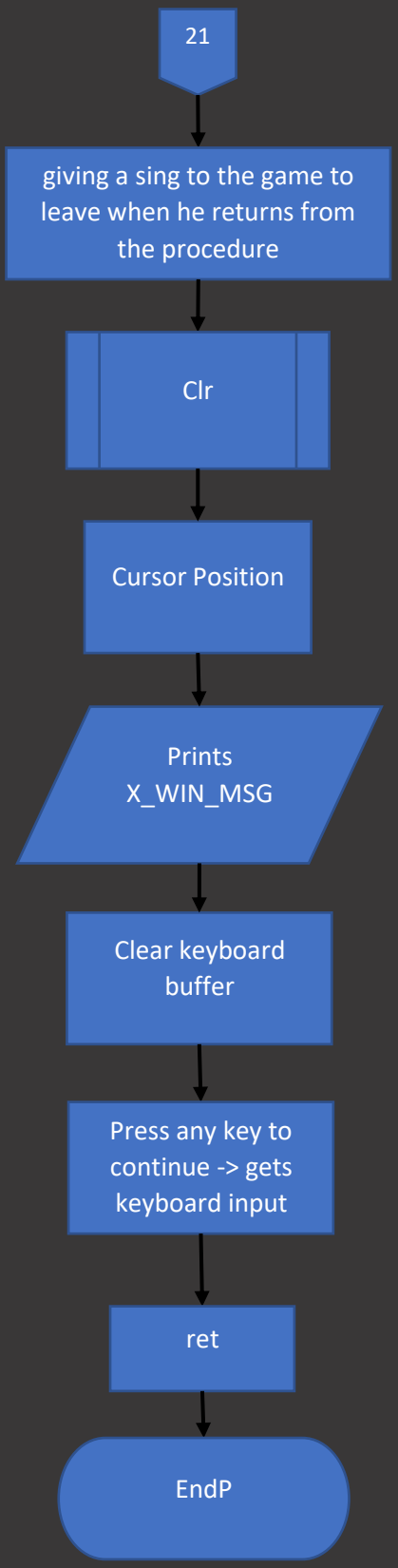


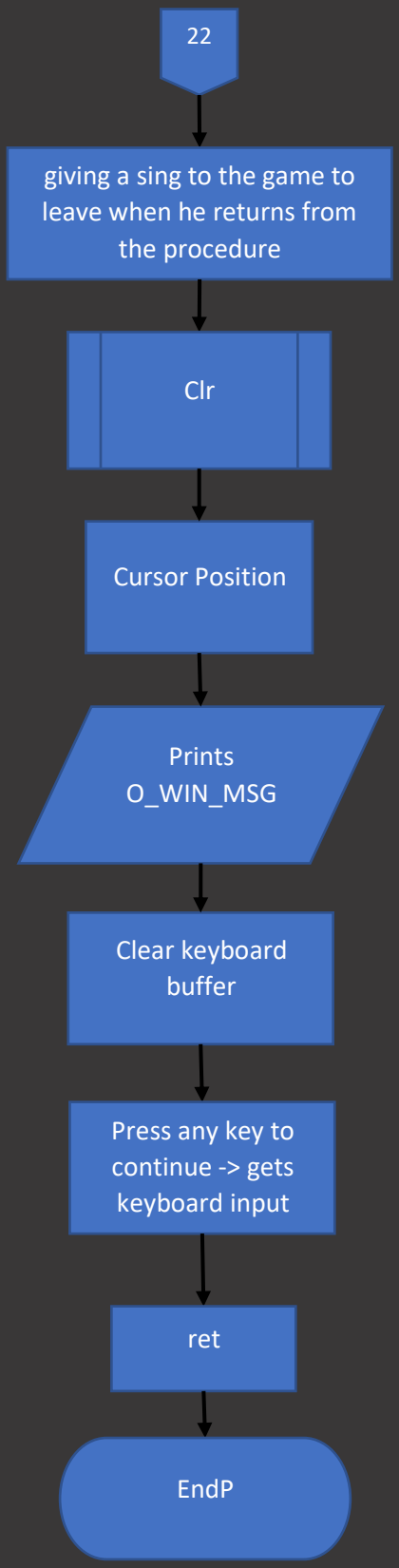

19
If area1 stores “O”
False
True
If area4 stores “O”
False
True
If area7 stores “O”
False
True
22
If area2 stores “O”
False
True
If area5 stores “O”
False
True
If area8 stores “O”
False
True
22
If area3 stores “O”
False
True
If area6 stores “O”
False
True
If area9 stores “O”
False
True
20
22

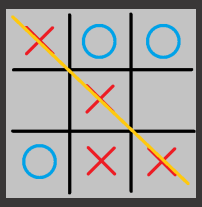



20
If area1 stores “O”
False
True
If area5 stores “O”
False
True
If area9 stores “O”
False
True
22
If area3 stores “O”
False
True
If area5 stores “O”
False
True
If area7 stores “O”
False
True
22
ret
EndP




21
giving a sing to the game to leave when he returns from the procedure
Clr
Cursor Position
Prints X_WIN_MSG
Clear keyboard buffer
Press any key to continue -> gets keyboard input
ret
EndP


22
giving a sing to the game to leave when he returns from the procedure
Clr
Cursor Position
Prints O_WIN_MSG
Clear keyboard buffer
Press any key to continue -> gets keyboard input
ret
EndP



Procedure Explanation

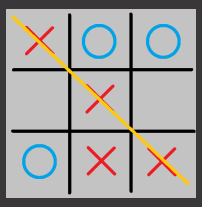
The program is built from couple of parts:

Label Main- *Contains the Commands to call procedure Menu and the choice to play, help, or exit.*

Proc Menu- *Prints the menu.*

Proc Game- *Contains most of the program, using a lot of procedures inside of it, and another 2 small choice menus.*

Proc Quit - *Restores to Text mode and returns to the place it was called (in Main label) to Exit.*

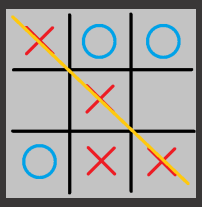


Proc ChangeTurn- *Changes the player turns. if his shape was "X" after 1 turn the procedure changes his shape to "O". I'm using it for PvP game modes logic.*

Proc ChangeTurnONLY_PvE- *do the same as ChageTurn, I created it only for PvE game modes logic.*

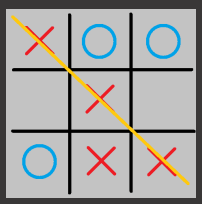
Proc PrintBoard- *Prints the board on the screen.*

Proc PrintPlayerTurn- *checks which player supposed to play, "X" or "O", then prints a suitable message on the screen, "X turn" or "O turn".*



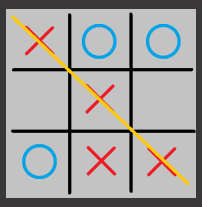
***Proc PrintShape**- This procedure is being called in Game procedure. it Contains half of the logic of the game with the procedure PrintShape_PvE_X together, and its job is to get the input of the player and checks if it is suitable for the game and its course, then it prints the correct shape in the suitable area.*

***Proc PrintShape_PvE_X-** Does the same as PrintShape does but a little different. This procedure is being called in Game procedure for a unique game mode, PvE, when player choose X, it was created because when player choose this game mode, it didn't work well on PrintShape procedure, so I created this procedure to fix this game mode only.*



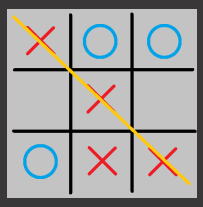
Proc CheckIfPlayerWins- *This procedure contains the second half of the game logic.*
Its main purpose is to check if one of the players won the game.
it has all the different variants of finding the winner (8 different ways per shape).
*2 shapes -> 2*8 different ways = 16*

Proc ChooseXO- *This procedure prints the shape choice menu, gets input from the player (his choice), and sets variable shape to "X" or "O". It takes a big part of the logic and without it, the whole logic structure won't work.*



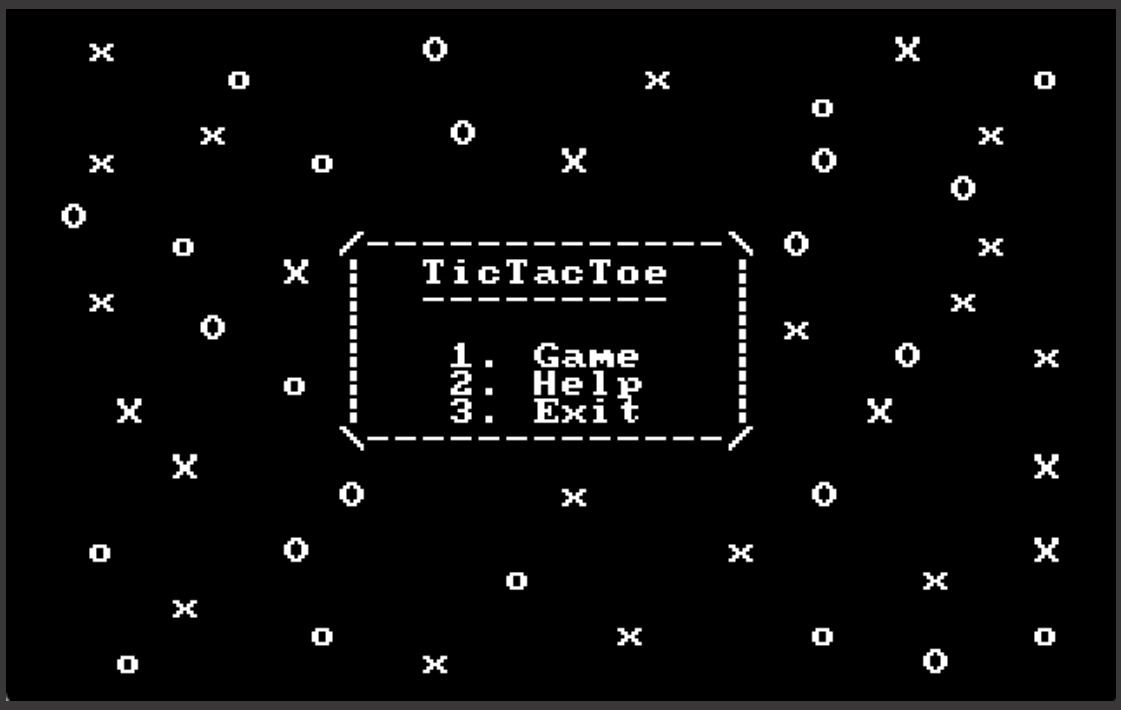
Proc Instructions- *Its job is to print the instructions if player chose to in the main menu. It has two pages that can be swapped by pressing Enter key, and to exit player needs to press Escape Key.*

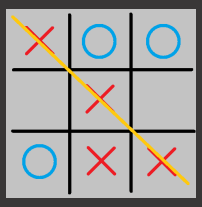
Proc Clr- *Its job is to clear the screen from text and colors. Its not really clearing the screen, its actually painting the screen in one single color (the color which the programmer chose, I chose black).*



Running Examples

Main Menu:

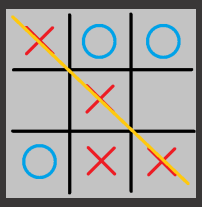




Help:

```
----------------------------------------
INSTRUCTIONS: Page 1
XO Game or TicTacToe(American English),
is a minigame for two players
who take turns marking the spaces in
three-by-three grid with X or O.
The player who succeeds in placing
three of their marks in a horizontal,
vertical, or diagonal row is the winner.
----------------------------------------
        Next Page -> (press Enter)
         To Go Back Press ESCAPE
```

```
----------------------------------------
INSTRUCTIONS: Page 2
To place shape in the chosen area press
the adjusted keybind (1,2,3,etc.) or
press on your left mouse button, and
to exit the game, press escape or
right mouse button.
----------------------------------------
     Previous Page -> (press Enter)
         To Go Back Press ESCAPE
```

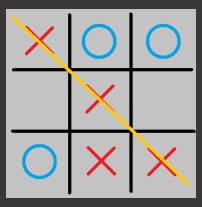


Menu2: Choose Gamemode

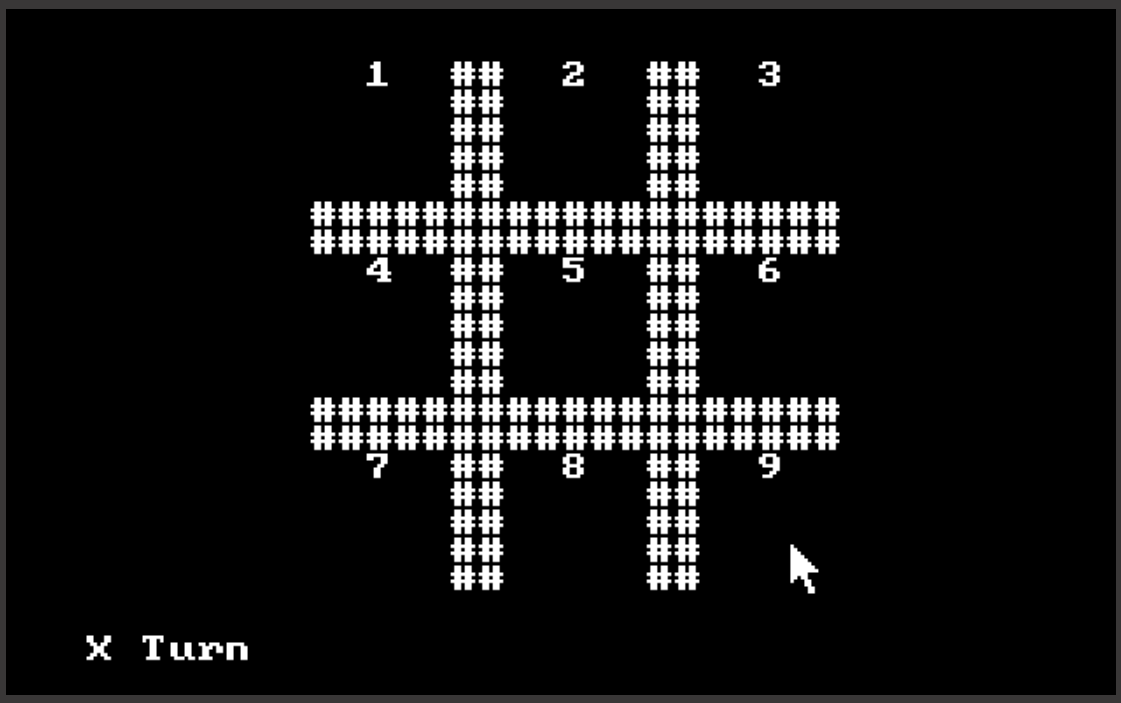
```
Choose Gamemode
--------------------------------
1. PvP - Player vs Player
2. PvE - Player vs Computer
--------------------------------
To Go Back Press ESCAPE
```

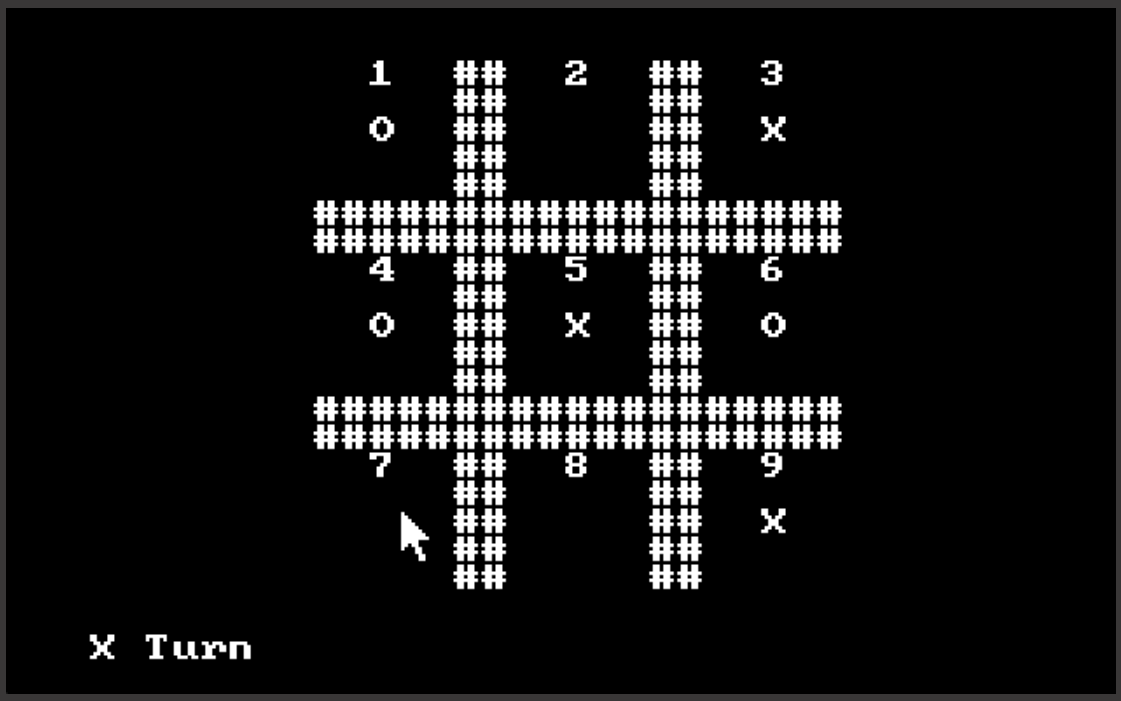
Menu3: Choose Shape

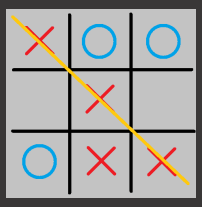
```
Choose Shape
--------------------------------
1. X
2. O
--------------------------------
To Go Back Press ESCAPE
```



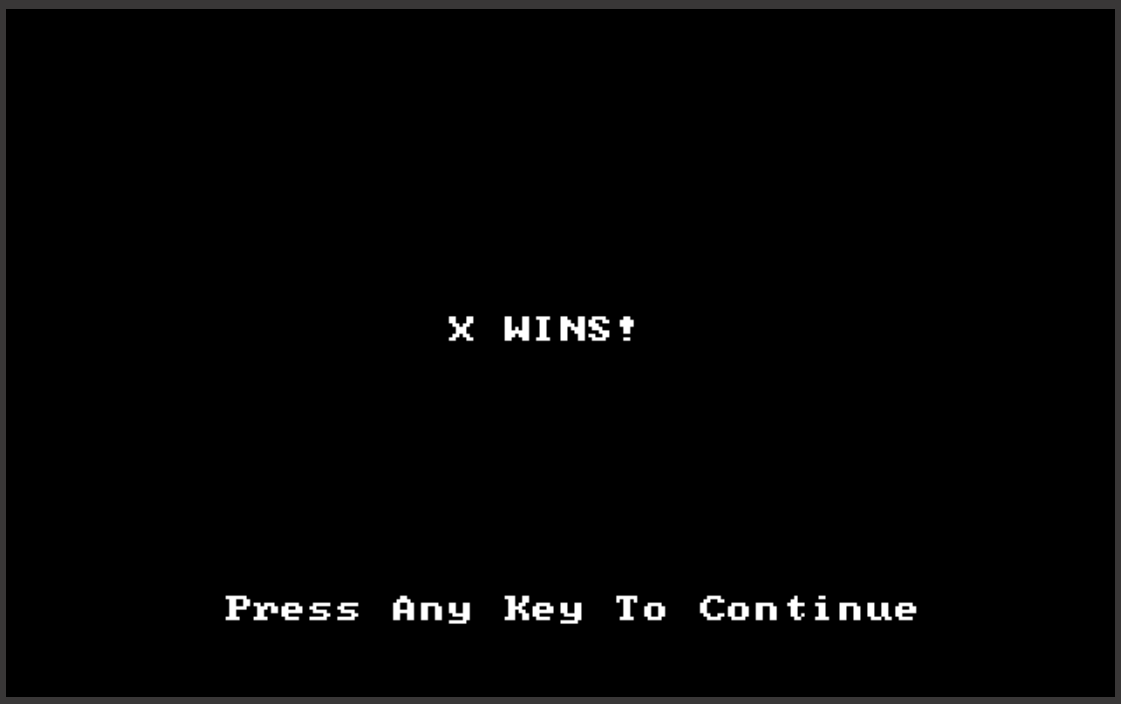
Board (while playing the game):

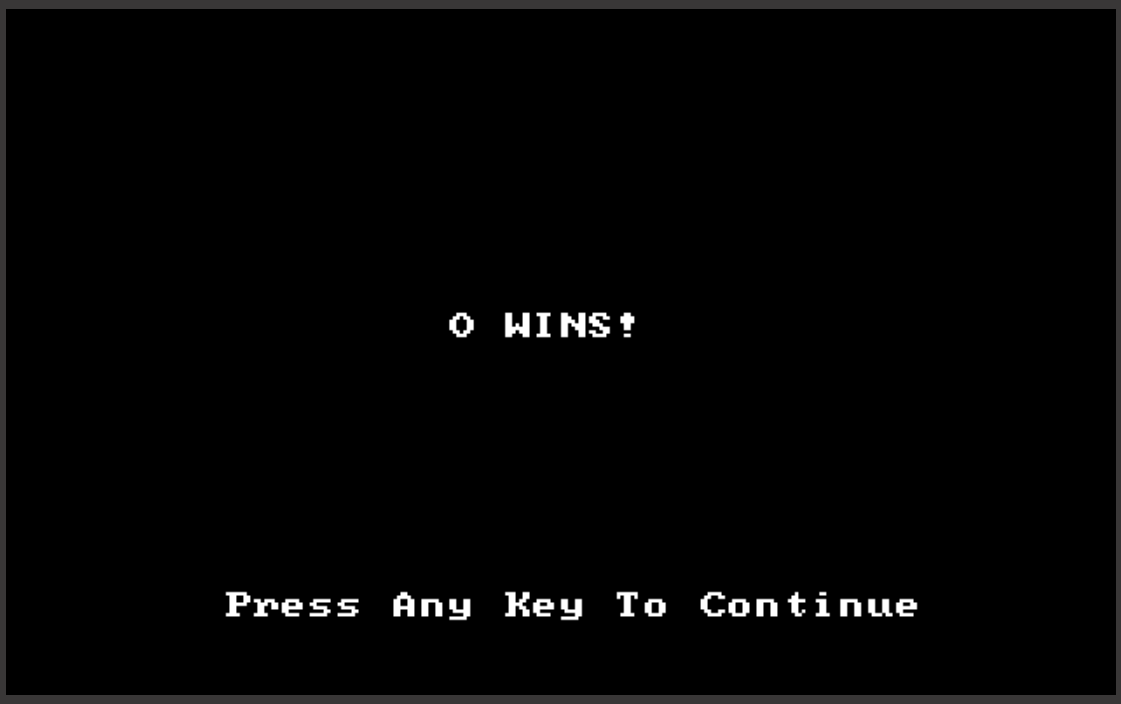


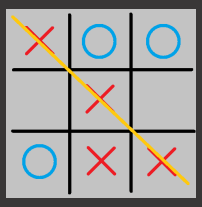




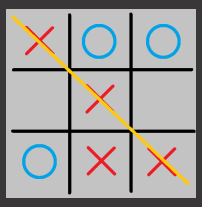
Win Messages:







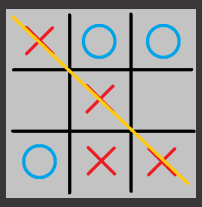
Draw Message:



Personal Summary

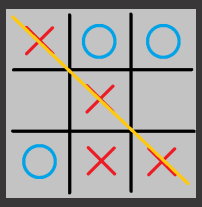
To sum up this project, I enjoyed so much creating a game, especially in a difficult language. I enjoyed studying a new programing language. I enjoyed solving difficult algorithmic problems and learning from that. And I enjoyed learning materials independently and succeeding on my own without the help of others.

The main thing that I Learned was to be with more self-confidence, and being independent, I started realizing that when I have more self-confidence, I'm more independent and I can still succeed on my own and without help



from others. For example: In the start of the year Assembly was medium level

for me, not hard, but not too easy, it was exhausting, but because I wanted to succeed I become more motivated, and today I can write very hard algorithms without that much of difficulty. Also today I like this language a lot, and I'm felling smart using it because for most of the people its hard to understand it, that's why I'm even helping others that struggling with it, and I'm teaching them and explaining part of the algorithm, I'm not doing the code for them, because I know it wont help them if I do all the work for them.



Thanks

***A BIG Thanks** to my amazing teacher Anatoly Peymer, who helped me all this long way until today, and taught me a lot of tools for life and for programming. I enjoyed in every lesson with you, and every lesson you make me like programming more and more.*

***Also thanks a lot** to Liam Mark, and Maor Reem for helping me solve problems and teaching me new things for having a better project.*