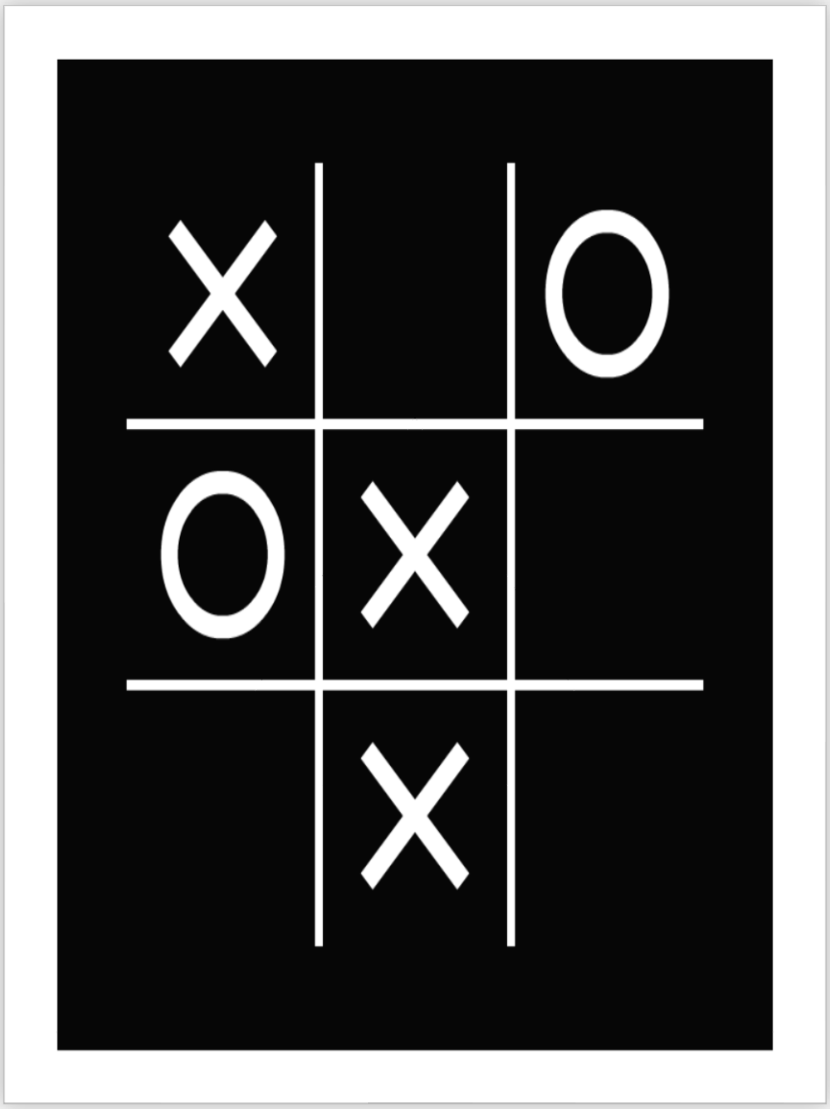


# Final Report

OXO GAME

Madikizela Will | Computer Science | June 22, 2020

## Table of Content

## Introduction

As a developer in the game development company known as Ten Eleven Game. I was approached by the company to develop a game called OXO Game, also known as tic tac toe game.

The game was expected to allow two users to connect to the server and allocate each player a character randomly i.e. O and X. Design the program-user Interface that is easy to use and understand, also state the interface conversions to the user at the beginning of the game and display the board. Should produce valid moves only and report invalid moves. The two players take turns putting their characters in the 3×3 grid board. The player who first gets three characters in a row (vertically, horizontally or diagonally) wins the game, and the other player loses the game.

I was required to develop a python-based program that will allow users to connect to the server and get their characters. The communication messages from the server to the player display to the screen while the player is playing.

## Project Management

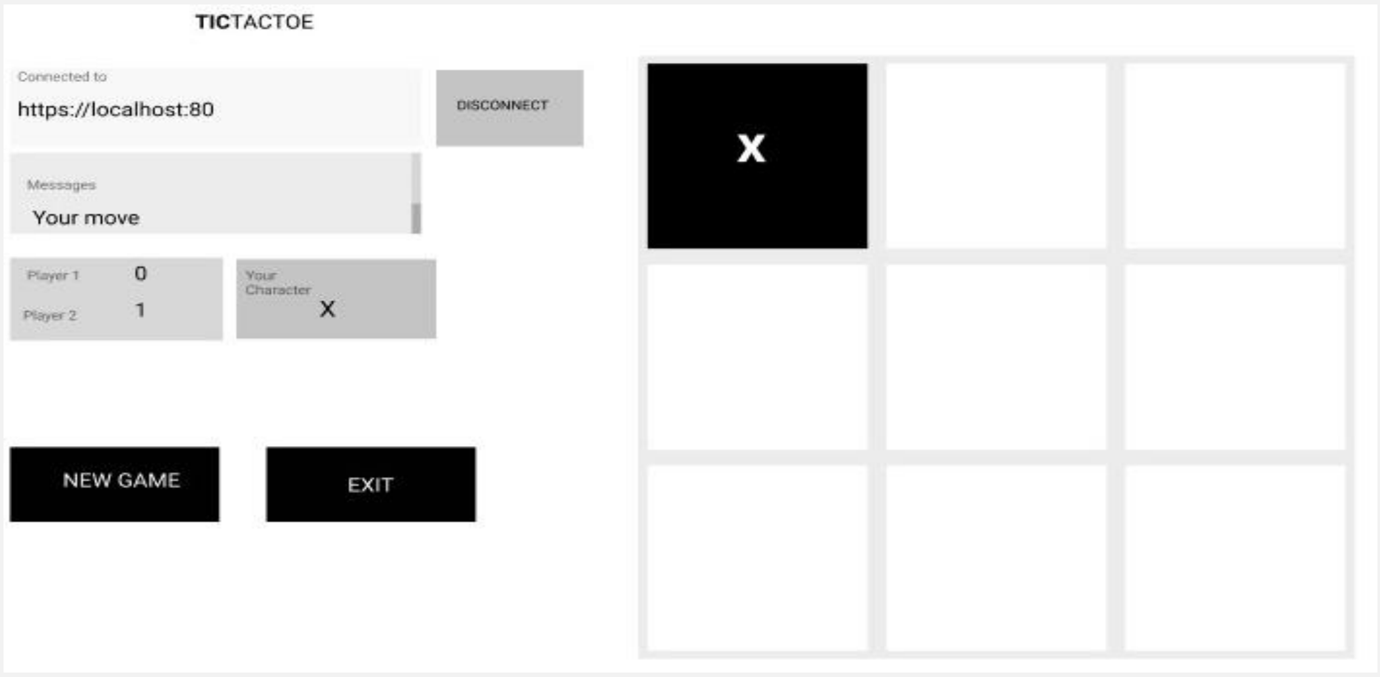
### Work Plan

The project's work was broken down into 6 weekly assignments. The Project plan was created that servers as the guideline on creating timelines and establishing deadlines for each phase of the project. The schedule plan that was designed in the project plan was used to factor different times-related and provide a good idea on what to be done next. That gave me a lot of time to plan and submit every work that was required in time.

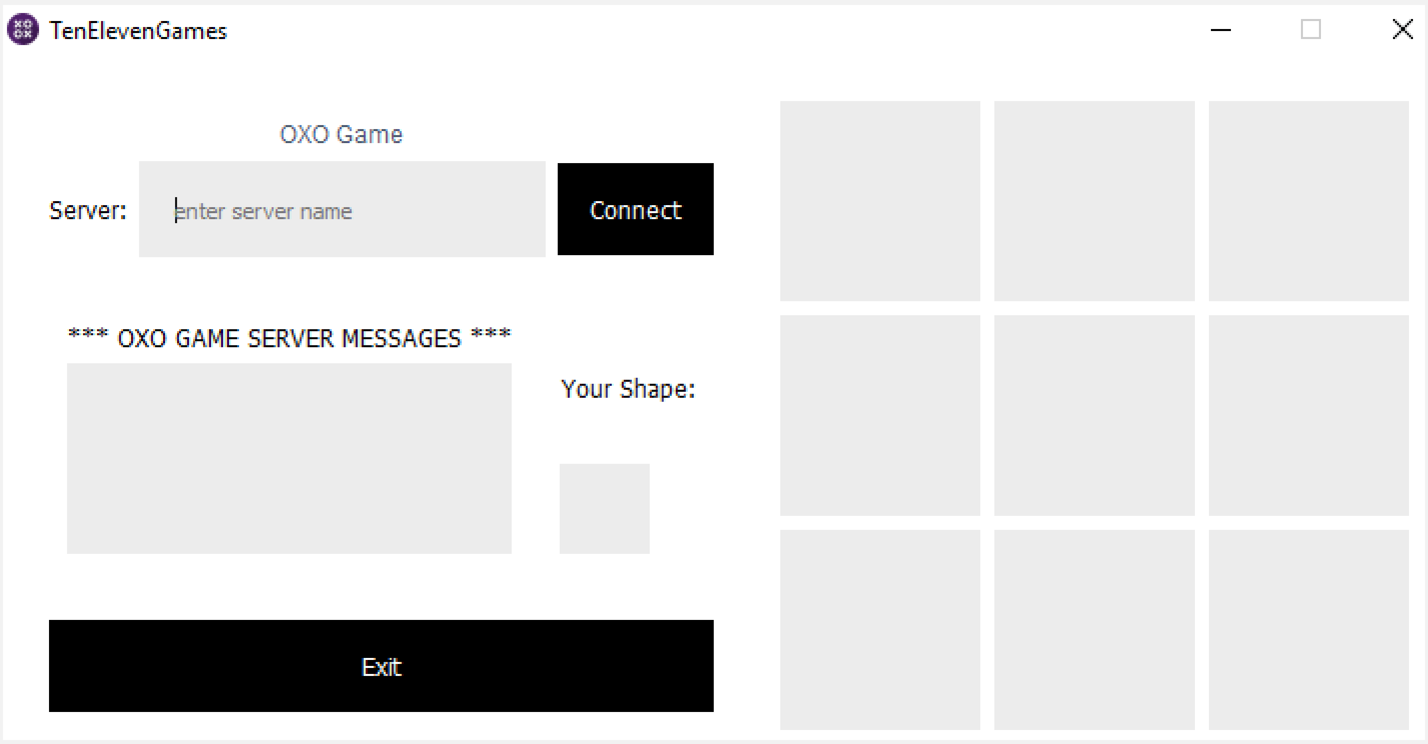
## System Design

Low Fidelity Prototype – the first idea I had when designing the GUI of the game

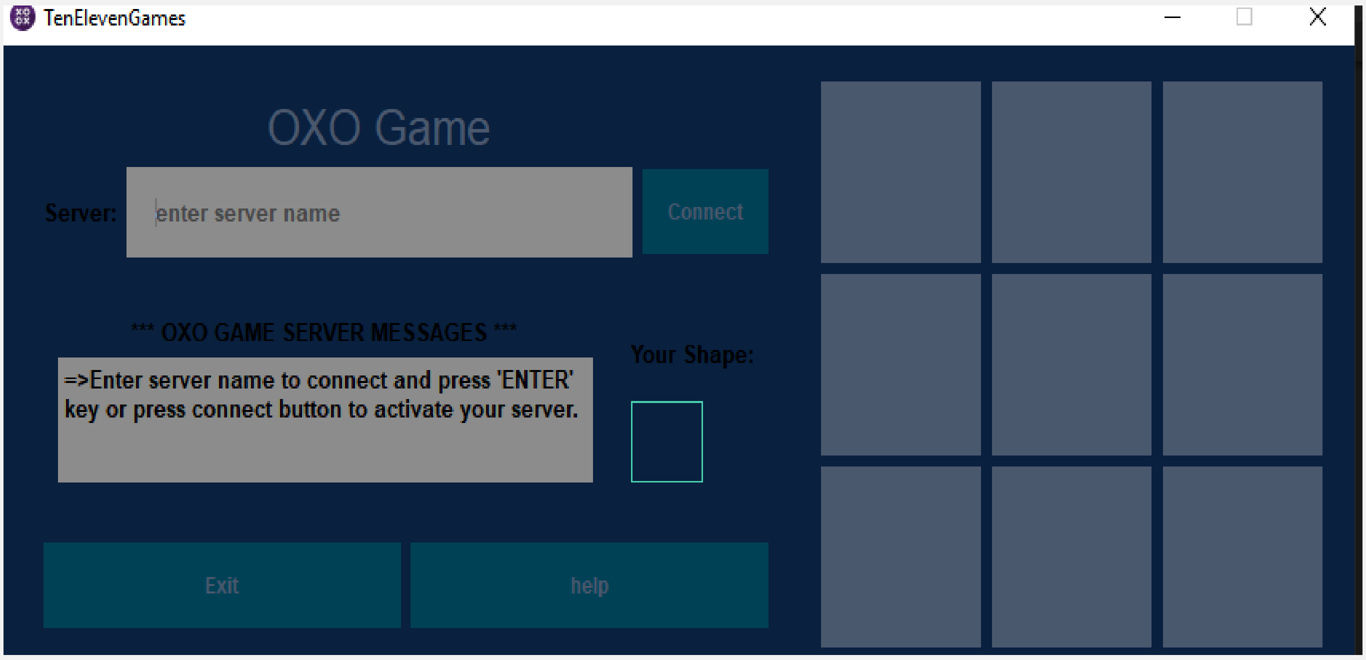
- This was designed by *figma* (it's not a functioning program)



## High Fidelity Prototype



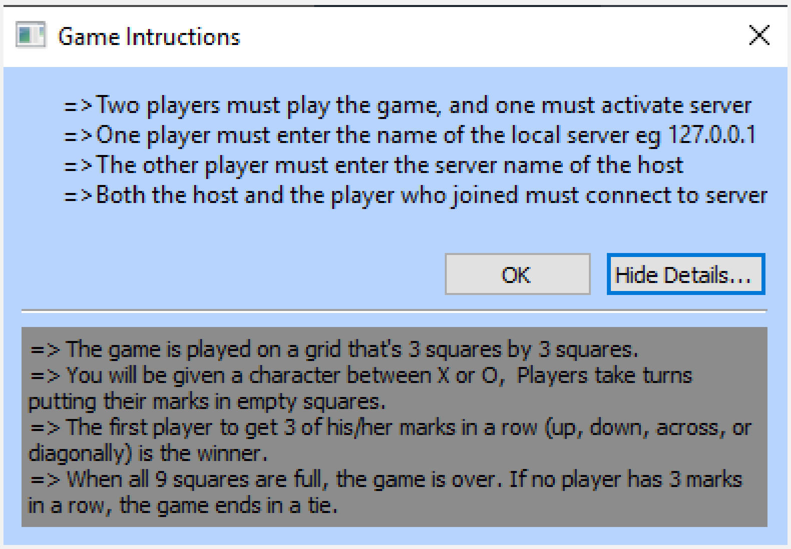
## Final Design – with all the implementations made to the game



=>Enter server name to connect and press 'ENTER' key or press connect button to activate your server.

## Implementation

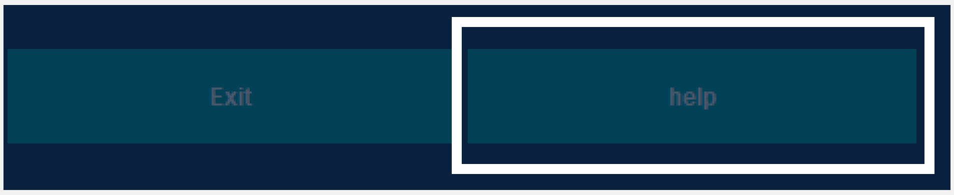
I have implemented the help button that will allow users not familiar with how to connect to the server, get assistance, and guide them on how to play the game with the game rules as well.


Game Intructions

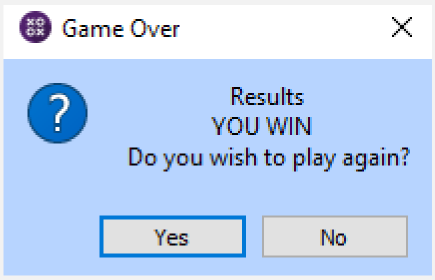
=>Two players must play the game, and one must activate server
=>One player must enter the name of the local server eg 127.0.0.1
=>The other player must enter the server name of the host
=>Both the host and the player who joined must connect to server

=> The game is played on a grid that's 3 squares by 3 squares.
=> You will be given a character between X or O, Players take turns putting their marks in empty squares.
=> The first player to get 3 of his/her marks in a row (up, down, across, or diagonally) is the winner.
=> When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.

This is the message that will pop up when the help button is clicked.

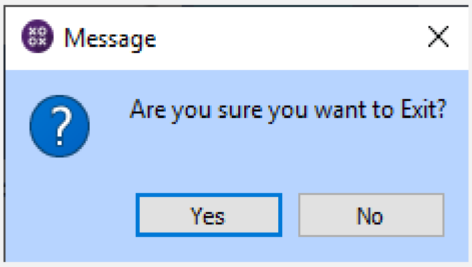


This is the pop-up message that indicates the results stating the winner or loser or weather it's a tie.


Game Over

Results
YOU WIN
Do you wish to play again?

When the player exits the game, pressing the exit button, a pop-up message will appear asking the user if they are sure to exit the game. This saves a lot of work avoiding mistakes and unexpected misuse of the buttons.

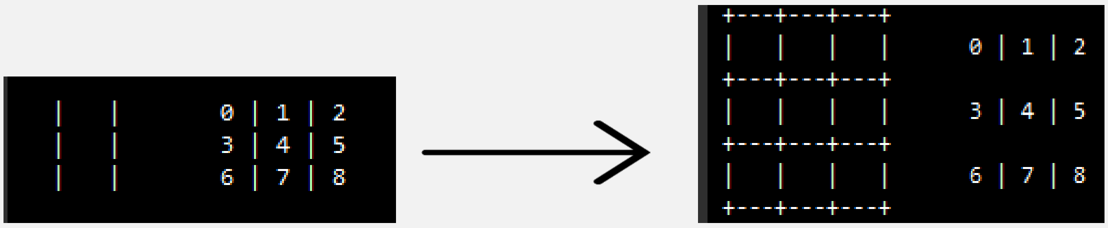

Message

Are you sure you want to Exit?

## Testing

### Text Based Client

One of the major concepts as a developer is to satisfy the user or the client, since we are developing a system for the company, we need to evaluate our program before we procced. On the text-based client, the major problem was the board design, the program was working as expected without errors. To solve the problem, I added borders (outer lines) to make it clearer.



```
 |   |        0 | 1 | 2
 |   |        3 | 4 | 5
 |   |        6 | 7 | 8
```

```
+---+---+---+
|   |   |   |      0 | 1 | 2
+---+---+---+
|   |   |   |      3 | 4 | 5
+---+---+---+
|   |   |   |      6 | 7 | 8
+---+---+---+
```

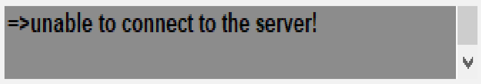
### GUI Prototype

In design, although there are several factors that have a significant influence on the outcome, colors play a major function to attract users. Color is the easiest and the most important aspect of engaging the user with the product in design. To overcome the problem, I changed the color of gui from white to dim blue. The user interface was well laid out and everything was easy to access and placed in an appropriate position.

## System Correctness

The system was tested in many ways to make the program bug free. Different scenarios were, mostly major scenarios that could course major problems in the program, make it to crush or have logical errors that will not be pleasant for the user.

**Enter wrong server name** – the program responded accordingly by displaying the error message in the Text screen.



```
=>unable to connect to the server!
```

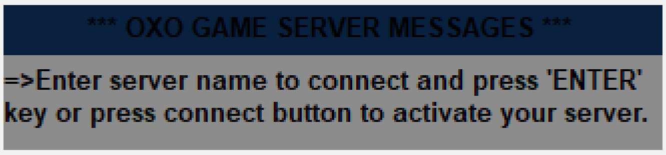
**Press the board play button before connecting to the server** – before connecting to server and getting your icon, the program does not allow the user to press the board play button (the buttons are disabled)

**Pressed connect button multiple times** – after connecting to server the program disables the connect button, disallowing the users to press the button multiple times

**Edit the text edit that displays messages from the server** – the Text box could not be edited, at all.

## Enhancements

Connect the keyboard to the game – I have linked the connect button to the keyboard, "ENTER" button. Displayed a line of instruction to the user that will assist the user understand how to use it.



```
*** OXO GAME SERVER MESSAGES ***
=>Enter server name to connect and press 'ENTER'
key or press connect button to activate your server.
```

Disable the play board buttons – when it's the opponent's turn to play.

Play sound – the program can play sound when you place the character to the game board button, also play different sounds when the player WINS or LOSES.

## Future Work

Artificial Intelligence can be used for this game in order to play only single player, in other words the server that was given should be able to allow the user to play against the computer.

Enable users to change the color of the game to the color of their needs.

Collect the score when players play multiple times

## Conclusion

In the conclusion of the project, I would like to say that Python is a fun programming language and while creating a project like this, it has not just been a good experience, but it also helped in the development of my creativity and logical thinking and also enhanced my programming skills, also gave me the taste of the really world problems. I would be more than happy to work on the other projects in Python because it’s just amazing to work with Python. The program is working, and it’s also bug-free. The Project was handled nice and smooth although it was super challenging, I managed to do all the tusks in time.

# Appendices

## OXO_Game.py

```python
# main gui class
# Will Madikizela

import sys
from PyQt5.Qt import Qt
from PyQt5.QtWidgets import*
from PyQt5.QtCore import*
from PyQt5.QtGui import*
from time import*
from PyQt5.QtMultimedia import*
from GameClient import*

class LoopThread(QThread):
    # create a signal
    msg_signal = pyqtSignal(str)
    def __init__ (self):
        QThread.__init__(self)
    def run(self):
        while True:
            msg = OXO.receive_message()
            # emit signal
            if len(msg):self.msg_signal.emit(msg)

class OxoGame(QWidget, GameClient): # inherits from QWidgets and gameclient
    # parent difines parent widget
    def __init__ (self, parent = None):
        # super class constructor
        QWidget.__init__(self, parent)
        GameClient.__init__(self)
        # window name
        self.setWindowTitle("TenElevenGames")
        # backgroung color of the window
        self.setPalette(QPalette(QColor("#09203F")))
        # x, y, widgth and height
        self.setGeometry(270, 117, 500, 350)
        # game icon
        self.setWindowIcon(QIcon("gameIcon.png"))
        # Disable maximizing window
        self.setFixedSize(850, 350)
        # default font
        self.setFont(QFont("arial", 10, weight = QFont.Bold))


        self.sounds = dict(play=QSound("play.wav"),
                           win=QSound("win.wav"),
                           lose=QSound("lose.wav"))

        # Declare variables
        # heading label
        self.heading  = QLabel("OXO Game")
        # set font of the heading label
        self.heading.setFont(QFont("Simplifica", 20, 10))
        # set color of the heading label to greyish blue
        self.heading.setStyleSheet("color: #4A586E ")
        # set the heading label to center
        self.heading.setAlignment(Qt.AlignCenter)

        # server label
        self.server_input_label = QLabel("Server:")
        # server input
        self.server_url_input = QLineEdit()
        # reurn pressed to allow the user to press 'Enter' key to connect
        self.server_url_input.returnPressed.connect(self.connect)
        # allow a button to clear all the text at once
        self.server_url_input.setClearButtonEnabled(True)
        # place holder in the input for server for more intructions
        self.server_url_input.setPlaceholderText("enter server name")
        # button to connect to server
        self.connect_button = QPushButton("Connect")
        # set Object Name for connect name
        self.connect_button.setObjectName("connect")
        # connect the button to report action when pressed
        self.connect_button.clicked.connect(self.connect)


        # labels indicating messages from server
        self.messages_from_server_label = QLabel("***OXO GAME SERVER MESSAGES***")

        # align the message label to centre
        self.messages_from_server_label.setAlignment(Qt.AlignCenter)
        # messages from the server
        self.messages_from_server = QTextEdit("")
        # disable editing the textedit
        self.messages_from_server.setReadOnly(True)

        # label indicating the players character
        self.character_label = QLabel("Your Shape:")
        # align the label to centre
        self.character_label.setAlignment(Qt.AlignCenter)

        # character of the player
        # x character
        self.o = QIcon("nought.png")
        # o character
        self.x = QIcon("cross.png")
        # blank character
        self.null = QIcon("blank.gif")

        # the button to place the character
        self.character = QPushButton()
               # oject name to specify the character button
               self.character.setObjectName("character")
               # set text to Fasle to allow icon to be inserted to the button
               self.character.setText("")
               # set the size of the icon to 45 heigth and 45 widgth
               self.character.setIconSize(QSize(45, 45))
               # set the size of the button to 45 height and 45 widgth
               self.character.setFixedSize(45, 45)
               self.character.setEnabled(True)

               # button to exit the game
               self.exit_button = QPushButton("Exit")
               # help button
               self.help_button = QPushButton("help")
               # set Object for help button
               self.help_button.setObjectName("help")
               # set ObjectName for exit button
               self.exit_button.setObjectName("exit")
               # connect the button to report action when pressed
               self.exit_button.clicked.connect(self.closeEvent)
               self.help_button.clicked.connect(self.help_button_clicked)

               # board buttons
               self.board = QGridLayout()
               # customize horizontal spacing between buttons to 7px
        self.board.setHorizontalSpacing(7)
        # customize vertical spacing between buttons to 7px
        self.board.setVerticalSpacing(7)
        self.counter = 0
        # iterate at a range of 3 to allow 3 columns
        for self.column in range(3):
            # iterate at a range of 3 to allow 3 rows
            for self.row in range(3):
                # set board button
                self.board_play_button = QToolButton()
                # Fixed size of the button to 100 height and 100 widgth
                self.board_play_button.setFixedSize(100, 100)
                # Fixed size of the icon in the button to 150 height and 150 widgth
                self.board_play_button.setIconSize(QSize(150, 150))
                # set text to Fasle to allow icon to be inserted to the button
                self.board_play_button.setText("")
                # track each button by giving it, its object name
                self.board_play_button.setObjectName(str(self.counter))
                self.board.addWidget(self.board_play_button, self.column, self.row)
                # increment counter
                self.counter += 1
        # create widget for the board game
        self.board_widget = QWidget()
        self.board_widget.setLayout(self.board)
        # get all the buttons in the board widget
        self.allButtons = self.board_widget.findChildren(QToolButton)
        # iterate each button
        for button in self.allButtons:
            # connect each button to report feedback when pressed
            button.clicked.connect(self.buttons)


        """ empty variables that will hold the decision and the shape of the user """
        self.decision = None
        self.shape = None

        self.loop_thread = LoopThread()
        self.loop_thread.msg_signal.connect(self.handle_message)

        #################### LAY OUT MANAGEMENT ###########################

        # input info grid
        self.server_heading_grid = QGridLayout()
        # heading
        self.server_heading_grid.addWidget(self.heading, 0, 1, 1, 1)
        # server input label
        self.server_heading_grid.addWidget(self.server_input_label, 1, 0)
        # server input
        self.server_heading_grid.addWidget(self.server_url_input, 1, 1)
        # connect button
        self.server_heading_grid.addWidget(self.connect_button, 1, 2)
        self.server_heading_grid_widget = QWidget()
        self.server_heading_grid_widget.setLayout(self.server_heading_grid)

        # grid layout for player information
        self.detail_character_grid = QGridLayout()
        # label for messages from server
        self.detail_character_grid.addWidget(self.messages_from_server_label, 2, 0)
        # messages from server
        self.detail_character_grid.addWidget(self.messages_from_server, 3, 0)
        self.detail_character_grid_widget = QWidget()
        self.detail_character_grid_widget.setLayout(self.detail_character_grid)

        # character layout information
        self.character_layout = QGridLayout()
        # character display label
        self.character_layout.addWidget(self.character_label, 0, 0)
        # character of the player or user
        self.character_layout.addWidget(self.character, 1, 0)
        self.character_layout_widget  = QWidget()
        self.character_layout_widget.setLayout(self.character_layout)

        # layout for all the details of the user or player
        self.detail = QHBoxLayout()
        # layout for player information
        self.detail.addWidget(self.detail_character_grid_widget)
        # character layout information
        self.detail.addWidget(self.character_layout_widget)
        self.detail_widget = QWidget()
        self.detail_widget.setLayout(self.detail)

        # the exit layout
        self.button_horizontalBox = QHBoxLayout()
        # exit button
        self.button_horizontalBox.addWidget(self.exit_button)
        self.button_horizontalBox.addWidget(self.help_button)
        self.button_horizontalBox_widget = QWidget()
        self.button_horizontalBox_widget.setLayout(self.button_horizontalBox)


        vbox = QVBoxLayout()
        vbox.addWidget(self.server_heading_grid_widget)
        vbox.addWidget(self.detail_widget)
        vbox.addWidget(self.button_horizontalBox_widget)
        vbox_widget  = QWidget()
        vbox_widget.setLayout(vbox)

        ##############################  MAIN LAYOUT  #################################

        self.main_layout = QHBoxLayout()
        self.main_layout.addWidget(vbox_widget)
        self.main_layout.addWidget(self.board_widget)
        self.main_layout_widget = QWidget()
        self.setLayout(self.main_layout)

        # disable the grid button
        self.board_widget.setEnabled(False)

    def clear_board(self):
        # clears the board game to allow NewGame option
        for button in self.allButtons: # iterate each button
            button.setText("")
            button.setIcon(QIcon())
            button.setEnabled(True)

    def closeEvent(self):
        # generates a question when exit button is clicked
        reply = QMessageBox.question(
            self, "Message",
            "Are you sure you want to Exit?",
            QMessageBox.Yes | QMessageBox.No
        )
        if reply == QMessageBox.Yes:
            self.exit()
        else:
            pass

    def keyPressEnter(self, event):
        # close application from Escape key
        if event.key() == Qt.key_Escape:
            self.close()

    # fuction for connect button
    def connect(self):
        try:
            self.connect_to_server(self.server_url_input.text())
            # disables the connect button after connected succesfully
            self.connect_button.setEnabled(False)
            self.messages_from_server.insertPlainText("=>Successfully connected to the server.\n")
        except:self.messages_from_server.insertPlainText('=>unable to connect to the server!\n')
        else:self.loop_thread.start()

    # fuction for game board buttons
    def buttons(self):
        self.button = self.sender()
        self.sounds["play"].play()
        # sends the infomation of the clicked button to server
        self.send_message(self.button.objectName())

    def input_play_again(self,decision):
        # pop up message after the Game is over, asking if the users are willing to play again
        self.user_response = QMessageBox.question(
            self,
            "Game Over",
            "                 Results \n              "+ decision +"\nDo you wish to play again?",
            QMessageBox.Yes|QMessageBox.No
        )

        # assign each button to a variable to anable the server to communicate
        if self.user_response == QMessageBox.Yes:
            self.feedback = 'y'
        else:
            self.feedback = 'n'

    def handle_message(self, msg):

        # indicates the new game is about to start
        if msg[:msg.find(",")] == "new game":

            # disable the game board
            self.board_widget.setEnabled(False)

            # gets the shape from the message
            self.shape = msg[-1]
            if self.shape == 'O':
                self.character.setIcon(self.o)
            else:
                self.character.setIcon(self.x)

            # a statement indicating the player's character
            self.messages_from_server.insertPlainText("=>The game is about to begin your character is " + self.shape + "\n")
```

```python
                    self.messages_from_server.moveCursor(QTextCursor.End)

                # indicates it’s the clients move
                elif msg == "your move":

                    # Enable the game board
                    self.board_widget.setEnabled(True)

                    # let the player know it’s their time to move
                    self.messages_from_server.insertPlainText("=>It's your turn to move\n")
                    self.messages_from_server.moveCursor(QTextCursor.End)

                # indicates it’s the opponents move
                elif msg == "opponents move":

                    # disable the game board
                    self.board_widget.setEnabled(False)

                    # let the player know, the opponent player is about to move
                    self.messages_from_server.insertPlainText("=>Waiting for the opponent to move...\n")
                    self.messages_from_server.moveCursor(QTextCursor.End)

                # position chosen is valid
                elif msg[:msg.find(",")] == "valid move":

                    # get position from player
                    self.position = msg[-1]

                    # locate the button that is clicked
                    self.clicked_button = self.board_widget.findChild(QToolButton, str(self.position))

                    # Place an Icon to the clicked button
                    # msg[-1], is the shape
                    if msg[-3]== "X":
                        self.clicked_button.setIcon(self.x)
                    elif msg[-3] == "O":
                        self.clicked_button.setIcon(self.o)

                # position chosen is invalid
                elif msg == "invalid move":

                    # position unavailable on the board
                    self.messages_from_server.insertPlainText("=>You can't go there. Try again.\n")
                    self.messages_from_server.moveCursor(QTextCursor.End)

                # indicates that the game is over
                elif msg[:msg.find(",")] == "game over":

                    # disable the game board
                    self.board_widget.setEnabled(False)

                    # get results from server log for the winner
                    self.winner = msg[-1] # X

                    # checks which character wins
                    if self.shape == self.winner:

                        # set decision, to let know the user won
                        self.decision = "YOU WIN"

                        # play sound that indicates that the play won
                        self.sounds["win"].play()

                        # print the decision to the text that shows messages from server
                        self.messages_from_server.insertPlainText("=>Game Over!\n=>Thank you for playing," + self.decision + "\n")
                        self.messages_from_server.moveCursor(QTextCursor.End)

                    # if there is no winner - it's a Tie
                    elif self.winner == "T" :

                        # set decision to tie
                        self.decision = "IT'S A TIE"

                        # print the decision to the text that shows messages from server
                        self.messages_from_server.insertPlainText("=>Game Over!\n=>Thank you for playing,It's a Tie :) \n")
                        self.messages_from_server.moveCursor(QTextCursor.End)

                    # checks which character looses
                    else:

                        # play sound that indicates that the play lost
                        self.sounds["lose"].play()

                        # set decision, to let know the user lost
                        self.decision = "YOU LOST"

                        # print the decision to the text that shows messages from server
                        self.messages_from_server.insertPlainText("=>Game Over!\n=>Thank you for playing," + self.decision + "\n")
                        self.messages_from_server.moveCursor(QTextCursor.End)

                # see if the player wants to play again
                elif msg == "play again":

                    # show the pop_message, ask the users if they are willing to play again
                    self.input_play_again(self.decision)

                    # if the user wants to play again
                    if self.feedback == 'y':
                        try:self.send_message(self.feedback)
                        except:self.server_connection_lost()
                        else:self.clear_board()

                    # if the user does not want to play again
                    else:
                        self.feedback == 'n'
                        try:self.send_message(self.feedback)
                        except:self.server_connection_lost()

                # terminate the game
                elif msg == "exit game":

                    # message displayed if the other player exit game or is taking too long to play
                    self.messages_from_server.insertPlainText("=>One of the players does not wish to continue\n")
                    self.messages_from_server.moveCursor(QTextCursor.End)
                    self.close()
                    # passage time
                    sleep(5)

            # when the opponent disconnects from the game
```

```python
    def player_lost_connection(self):
        # displays the text to the server message text box
        self.messages_from_server.insertPlainText("=>Looks like your opponent disconnected\n")
        # closes the server
        self.socket.close()
        self.messages_from_server.moveCursor(QTextCursor.End)
        # clears the board
        self.clear_board()
        self.socket = socket(AF_INET, SOCK_STREAM)
        self.board_widget.setEnabled(False) # disable the board
        self.connect_button.setEnabled(False) # disable the connect button

    # when the host closes the server
    def server_connection_lost(self):
        # displays the text to the server text box
        self.messages_from_server.insertPlainText("=>server disconnected!\n")
        # closes the server
        self.socket.close()
        self.messages_from_server.moveCursor(QTextCursor.End)
        # clears the board
        self.clear_board()
        self.socket = socket(AF_INET, SOCK_STREAM)
        self.board_widget.setEnabled(False) # disable the board
        self.connect_button.setEnabled(False) # disable the connect button

    # gives the user all the details they need to connect to server and how to play game
    def help_button_clicked(self):
        # create a pop-up dialog
        message = Message()
        # set title of the window
        message.setWindowTitle("Game Instructions")
        # the first text that will appear when the window shows up
        message.setText("""=>Two players must play the game, and one must activate server
=>One player must enter the name of the local server eg 127.0.0.1
=>The other player must enter the server name of the host
=>Both the host and the player who joined must connect to server""")
        # message in the button for more details about the game
        message.setDetailedText("""=> The game is played on a grid that's 3 squares by 3 squares.
=> You will be given a character between X or O,  Players take turns putting their marks in empty squares.
=> The first player to get 3 of his/her marks in a row (up, down, across, or diagonally) is the winner.
=> When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.""")
        message.exec_()

    def play_loop(self):
        while True:
            msg = self.receive_message()
            if len(msg): self.handle_message(msg)
            else:
                self.player_lost_connection()
                break

    # fuction for exit button
    def exit(self):
        self.close()

# Global styles
stylesheet = """
#connect {
```

```
            background: #004156;
            border: none;
            padding: 16px;
            color: #4A586E;
            font-size: 12px;
            font-weight: bold;
        }
        #exit {
            background: #004156;
            border: none;
            padding: 16px;
            color: #4A586E;
            font-size: 12px;
            font-weight: bold;
        }
        #help{
            background: #004156;
            border: none;
            padding: 16px;
            color: #4A586E;
            font-size: 12px;
            font-weight: bold;
        }
        #help::hover {
            border: 1px solid #48CFAF;
            color: #01142F;
        }

        #character {
            background: #09203F;
            border: 1px solid #48CFAF;
            padding: 16px;
        }
        #exit::hover {
            border: 1px solid #48CFAF;
            color: #01142F;
        }
        #connect::hover {
            border: 1px solid #48CFAF;
            color: #01142F;
        }
        QToolButton {
            background: #4A586E;
            padding: 16px;
            outline: none;
            border: none;

        }
        QToolButton:pressed {
            background: #D5D5D5;
            padding: 16px;
            outline: none;
        }
        QToolButton::hover{
            border: 1px solid #48CFAF;
            color: #01142F;

        }
        QToolBuuton:checked {
            background: #ECECEC;
            padding: 16px;
            outline: none;

        }
        QTextEdit {
            height: 32px;
            background: #8C8C8C;
            height: 32px ;
            outline: none;
            border: none;
        }
        QLineEdit {
            background: #8C8C8C;
            padding: 16px;
            outline: none;
            border: none;
        }
        QMessageBox {
            background: #B7D4FF;
            text-align: center;
            color: red;
            outline: none;

        }
        """
        app = QApplication(sys.argv)
        OXO = OxoGame()

        def main():
            app.setStyleSheet(stylesheet)
            OXO.show()
            OXO.messages_from_server.insertPlainText("=>Enter server name to connect and press 'ENTER' key or press connect button to activate your server.\n")
            sys.exit(app.exec_())
        main()
```