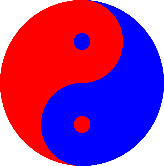


A

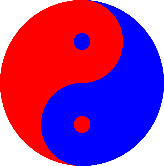
Documentation

Of the



Game

Ofentse and Mnotho



***Index***

**Introduction…………………………………………1**

**Project Management……………………………..1-6**

**Design………………………………………………...6**

* **Software design**
* **GUI Design**

**Implementations…………………………………7-9**

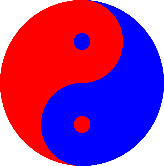
* **Low fidelity**
* **High fidelity**
* **Final GUI**

**Testing……………………………………………..10**

* **Development testing**
* **User testing**

**Conclusion………………………………………..10**

**Appendencies…………………………………..11-45**



***Introduction***

**This is a documentation for a project that required us to develop a two player game that had to be played online via an internet connection, the game that we had to develop is called SOS.SOS is a game like Tic Tac Toe but is more complex, SOS game is played on square grid with sixteen square inside the grid, the aim of the game is to create a straight sequence S-O-S among connected squares in a grid.**

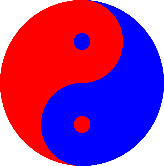
**On this document we will explain:**

* **Project management**
* **Design of the game**
* **Implementations**
* **Testing**
* **Enhancements**
* **Future Work**

***Project management***

**PROJECT PLAN and TEAM WORK:**

We drafted a time table to schedule our work accordingly So that we do not miss deadlines and due dates .On the time table each team member was allocated a specific task for a specific date.



**Time table for the Project plan:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Task | Who | Duration | Start | End |
| 1. | Text-Based Client | Mnotho and Ofentse | 7 Days | 13 April | 20 April |
| 1.1 | Creating the display\_board method, that deals with displaying the SOS board. | Mnotho and Ofentse | 2 Days | 13 April | 15 April |
| 1.2 | Creating the handle\_message method , firstly modifying it to notify when a new game is started. | Ofentse | 2 hours | 15 April | 15 April |
| 1.3 | Modifying the handle\_message method to check if the move a player made is valid or invalid and displaying a proper message . | Mnotho | 2 hours | 15 April | 15 April |
| Number | Task | Who | Duration | Start | End |
| 1.4 | Modifying the handle message method to check and notify the players when the game is over. | Ofentse | 2 hours | 17 April | 17 April |
| 1.5 | Modifying the method to check if the player wants to stop playing and terminate the program and display an appropriate method. | Mnotho | 3 hours | 18 April | 18 April |
| 2. | GUI Prototype | Mnotho and Ofentse | 7 days | 20 April | 27 April |
| 2.1 | Low-fidelity prototype. Here We’ll basically do a rough drawing of what our interface will look like, on a paper. Having combined ideas. | Mnotho and Ofentse | 1 hour | 20 April | 20 April |
| 2.2 | Consulting our team leader, Michael January about our low-fidelity prototype. | Mnotho and Ofentse | 2 hours | 21 April | 21 April |
| 2.3 | Starting with the high-fidelity prototype, the basic window operations, the resizing, the server text field and label | Ofentse | 1 hour | 22 April | 22 April |
| 2.4 | Adding the connect/disconnect button, quit/exit button, text area for server messages | Mnotho | 3 hours | 23 April | 23 April |
| 2.5 | Adding some more features, tuning the GUI | Ofentse  And Mnotho | 2 hours | 24 April | 24 April |
| Number | Task | Who | Duration | Start | End |
| 3 | Final System Alpha | Mnotho and Ofentse | 7 days | 27 April | 04 May |
| 3.1 | Get 5 people to fill out the questionnaire and use the feedback in a User Testing document. | Micheal,2 CSC1011H students and 2 non-CSC1011H students | 2 days | 27April | 29 April |
| 3.2 | Implementing the GUI. | Ofentse | 4 hours | 28 April | 28 April |
| 3.3 | Writing code to allow both users to connect if they request to(i.e clicking connect) | Mnotho and Ofentse | 8 hours | 29 April | 29 April |
| 3.4 | Modifying the interface such that it handles the new game message properly | Ofentse | 1 days | 30 April | 30 May |
| 3.5 | Creating an external thread for the play loop method so it won’t conflict with the PyQt event loop and also testing the last things modified | Mnotho | 2 days | 1 April | 3 April |
| 4. | Final System Beta | Mnotho and Ofentse | 7 days | 04May | 11 May |
| 4.1 | Writing an SOS game GUI client. | Mnotho | 1 day | 04 may | 05 May |
|  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4.3 | Testing the game by playing a few games and see if it works accordingly and debugging where necessary | Mnotho | 12 hours | 07 May | 07 May |  |
| 4.4 | Mnotho and Ofentse, each one of them implementing two separate features | Mnotho and Ofentse | 2 days | 08 May | 10 May |  |
| 4.5 | Test if the new features are properly working. | Ofentse | 4 hours | 10 May | 10 May |  |
|  |  |  |  |  |  |  |
| 5 | Final Report | Mnotho and Ofentse | 7 days | 11 May | 18 May |  |
| 5.1 | Starting the final report, beginning with the Introduction. | Mnotho and Ofentse | 1 day | 11 May | 12 May |  |
| 5.2 | This section is about how the project was managed and the experience with working with my partner. | Mnotho and Ofentse | 2 days | 13 May | 15May |  |
| Number | Task | Who | Duration | Start | End |  |
| 5.3 | Summarizing main points in the report | Mnotho  And Ofentse | 5hours | 16 May | 17May |  |
| 5.4 | Rehearsing a project presentation | Mnotho  And Ofentse | 1 day | 16 May | 17May |  |
| 5.5 | Possible enhancements that could add to user experience. | Mnotho | 8 hours | 17 may | 17may |  |

***Team Work:***

The experience of us two working as a team wasn’t good. We experienced so many problems, delays and disagreements. It’s changed how we view working in groups but after all the project was u success.

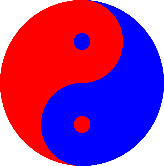
***Design:***

***Software Design:***

Python 3.4 and PYQT4 were used to develop the SOS game, since our game had to be played via an internet connection, we choose to use the STOMP {Simple/Streaming Text Oriented Messaging Protocol} because it text based and more simple and easier to use

***Gui Design:***

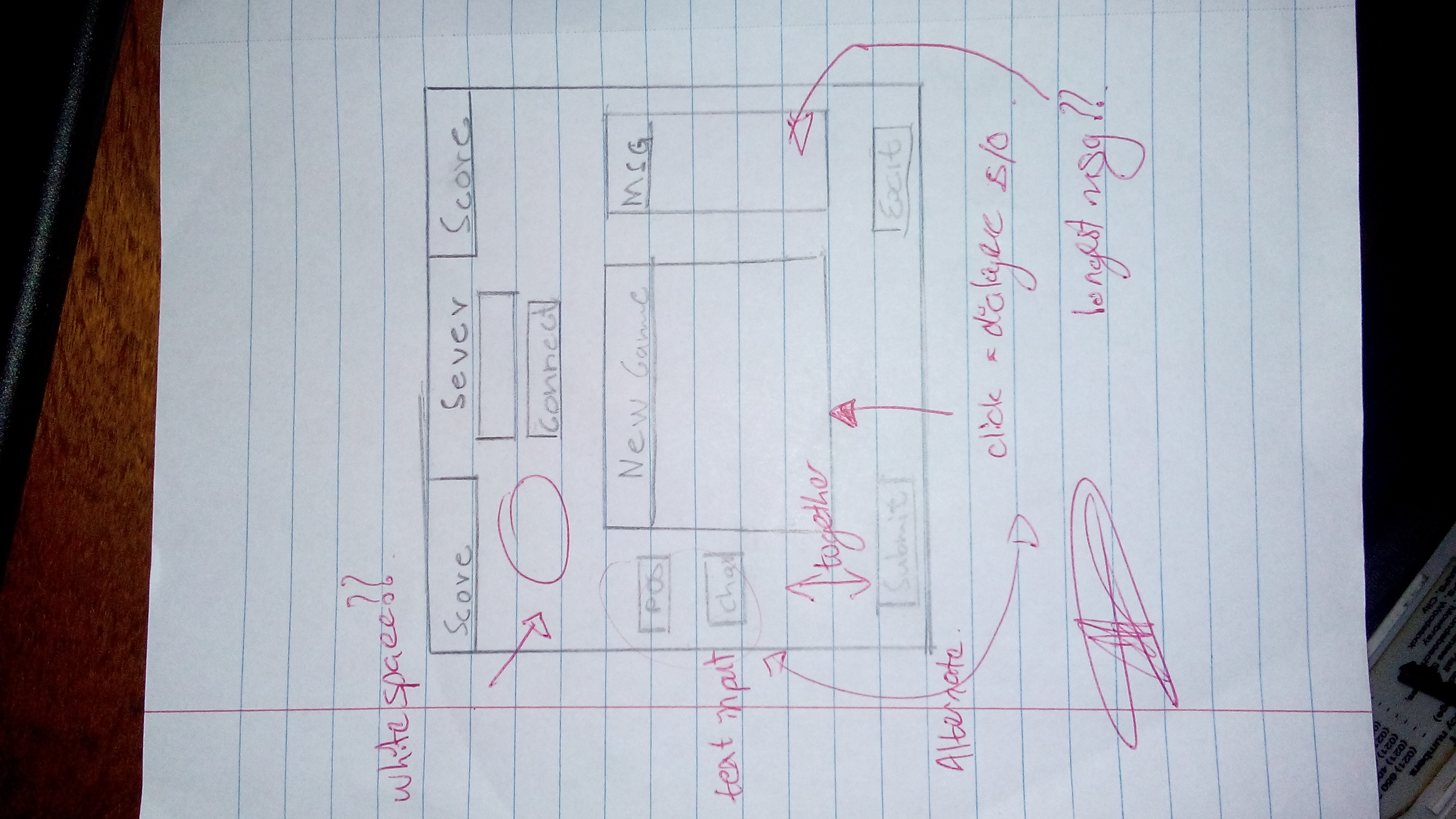
The Gui had three development stages ,the low fidelity prototype which was a simple user interface with a basic layout for SOS game but it no dynamic functionality for the game to run , the high fidelity prototype ,for the high fidelity prototype most the features from low fidelity prototype design where unchanged but a lot of features we added to it to it to make the design run the sos game in the background while it updates the user interface

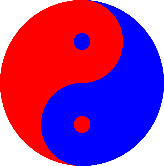


***Design implementations:***

***Low Fidelity:***

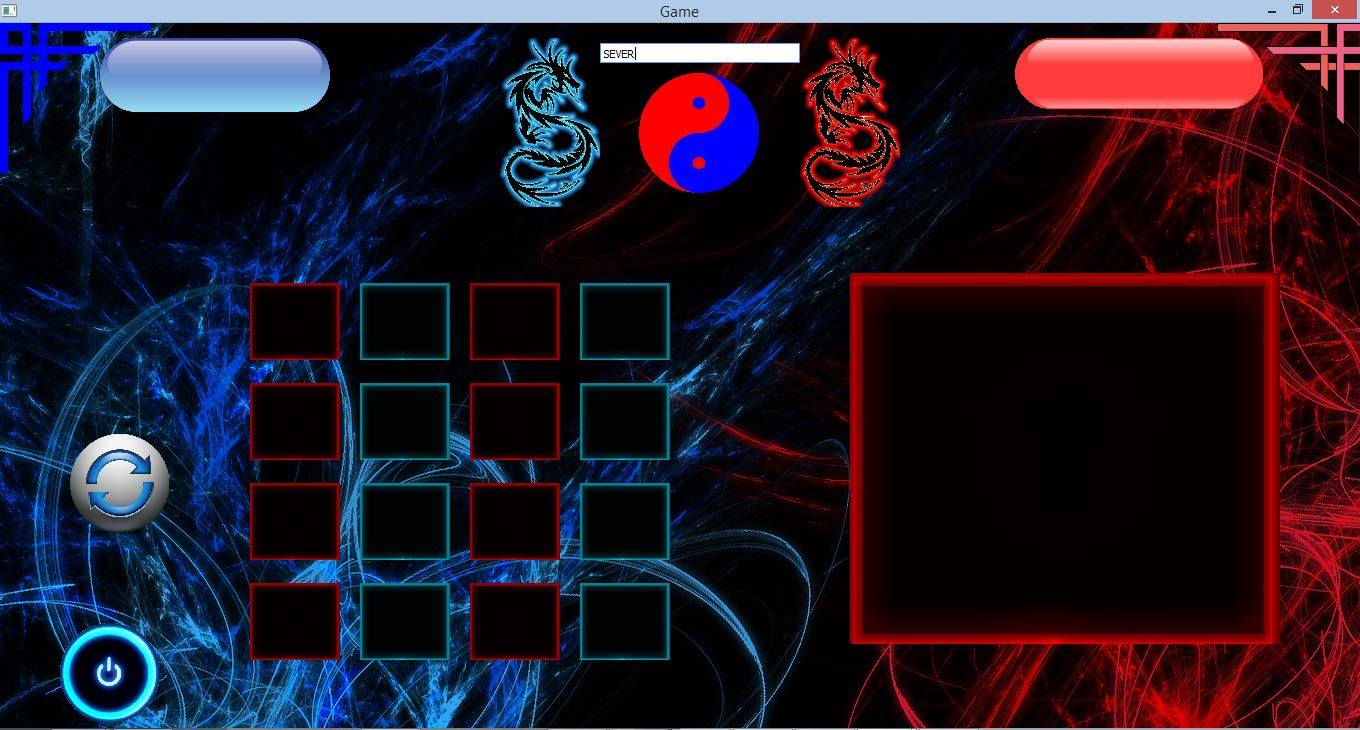
This is our low fidelity GUI ,our aim was to optimize the GUI for touch enabled desktop users and keyboard users ,the draft shows a simple layout with combo boxes, line edit and buttons



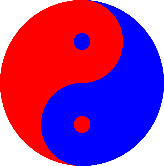


***High Fidelity GUI:***

The high fidelity GUI was meant to show a simple SOS game user interface layout but without the game running .all the features from the low fidelity from GUI were but the combo boxes were removed from the interface so that it simple looking yet more user friendly. A theme of red against blue was used that inspired by Chinese traditional art.

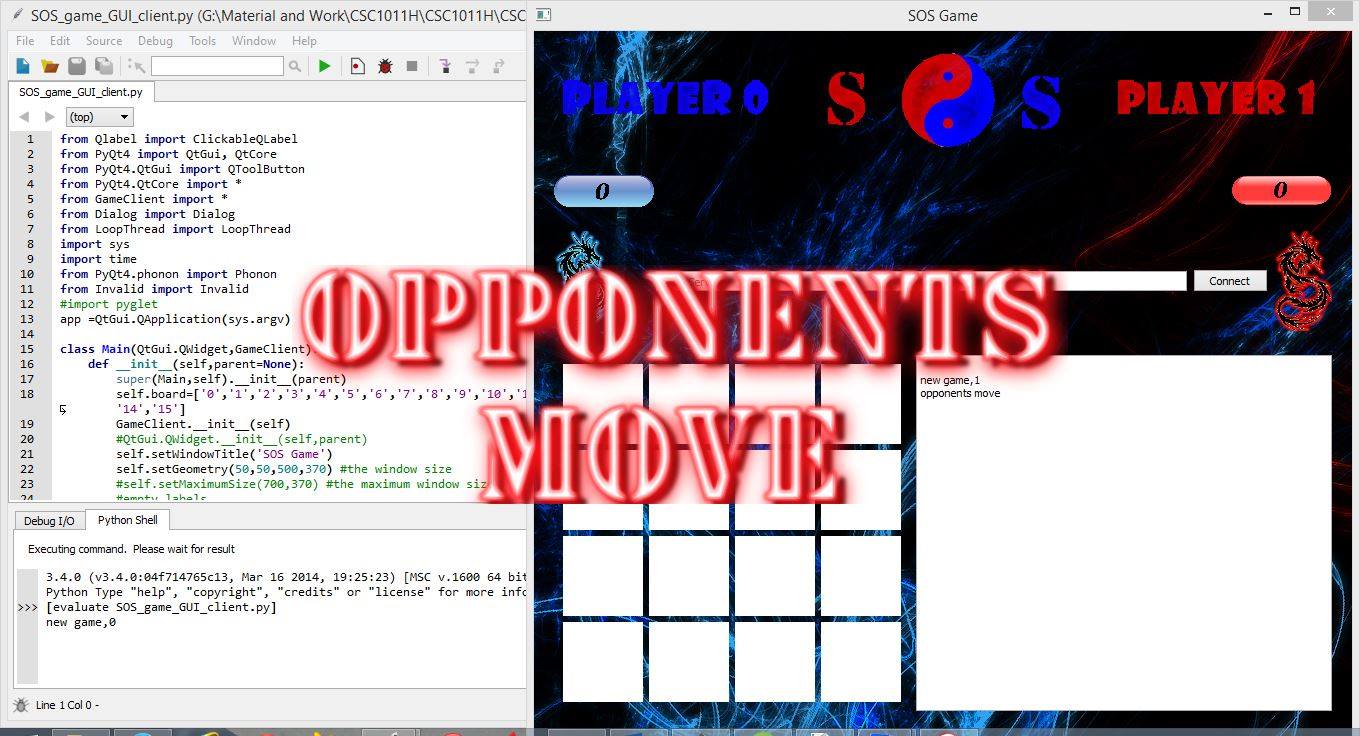


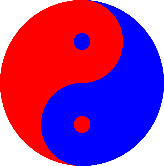
The GUI had some minor issues, absolute positioning was used so the GUI was not adjustable and it could only run only on full screen mode.



***Final GUI Design:***

On our final GUI Design we had lot of changes that were implemented the .Labels were added, the play again button was removed but the theme was kept the same





***TESTING:***

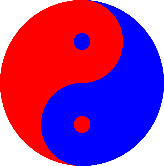
***Development testing:***

Each time a new feature was implemented on code for the GUI and the SOS game functionality it was tested, for example, when the sound feature was added to GUI for the first time it could not play because an external python module was used, the solution to the problem was to use a PyQT4 build in media play module called phonon

***User Testing:***

Most of the users had problems using the high fidelity GUI because it did not have any labels to explain what it is happening or what is expected from them, most of them had negative comments about the functionality of high fidelity GUI but they had positive comments about theme of Layout.

After user testing comments were reviewed, changes were implemented on the GUI to enhance user experience then we gave a second round of testing for the users to enjoy themselves ,We had positive comments from users after that because the help button for instruction on how to use the GUI were there and labels were also added.



***Enhancements:***

* Sound (PHONON): for notifications
* Clickable Images(MousePressedEvent): for the gameplay
* Flash plash: for notifications
* Massage box : for alerts (if what to quit the game)

Future enhancement:

* Animations for more user interaction
* Multiple player mode( more than two players to players)
* Multiple themes

***Conclusion:***

The project was an SOS game. The game was basically for use with a mouse, the formation of SOS’s. The road to getting the project done was indeed long and the team had a lot of difficulties and issues. Through it all, the game was finished and it was quite a learning experience.

***Appendences:***

from Qlabel import ClickableQLabel

from PyQt4 import QtGui, QtCore

from PyQt4.QtGui import QToolButton

from PyQt4.QtCore import \*

from GameClient import \*

from Dialog import Dialog

from LoopThread import LoopThread

import sys

import time

from PyQt4.phonon import Phonon

from Invalid import Invalid

#import pyglet

app =QtGui.QApplication(sys.argv)

class Main(QtGui.QWidget,GameClient):

def \_\_init\_\_(self,parent=None):

super(Main,self).\_\_init\_\_(parent)

self.board=['0','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15']

GameClient.\_\_init\_\_(self)

#QtGui.QWidget.\_\_init\_\_(self,parent)

self.setWindowTitle('SOS Game')

self.setGeometry(50,50,500,370) #the window size

#self.setMaximumSize(700,370) #the maximum window size

#empty\_labels

self.how\_to\_label=QtGui.QLabel('')

self.empty2=QtGui.QLabel('')

#pop up buttons to click when you want to play

self.st=QtGui.QPixmap("S")

self.st= self.st.scaled(150, 200, QtCore.Qt.KeepAspectRatio)

self.sd=QtGui.QLabel(self)

self.sd.setPixmap(self.st)

self.so=QtGui.QPixmap("O")

self.so= self.so.scaled(150, 200, QtCore.Qt.KeepAspectRatio)

self.sdo=QtGui.QLabel(self)

self.sdo.setPixmap(self.so)

self.grid2=QtGui.QGridLayout()

self.grid2.addWidget(self.sd,0,0) #adding the buttons to be clicked on the grid

self.grid2.addWidget(self.sdo,0,1)

self.grid2wid=QtGui.QWidget()

self.grid2wid.setLayout(self.grid2)

#background

self.picture = QtGui.QPalette(self) # Background image

self.picture.setBrush(QtGui.QPalette.Background,

QtGui.QBrush(QtGui.QPixmap("back.jpg")))

self.setPalette(self.picture)

#layouts

self.grid\_one=QtGui.QGridLayout() #the grid layout for images,S, O

self.vbox\_1=QtGui.QVBoxLayout()

self.score\_hbox=QtGui.QHBoxLayout() #the hbox layout for the scores

self.conn\_hbox=QtGui.QHBoxLayout() #the hbox for the server label, line edit and the button

self.grid\_vbox1\_layout=QtGui.QHBoxLayout() #the layout for the grid\_one and vbox1 widgets

self.bottom\_hbox=QtGui.QHBoxLayout() #the hbox layout for the bottom part, for the buttons (shut and play)

self.top\_hbox=QtGui.QHBoxLayout() #the upper most hbox layout

self.edit\_box=QtGui.QTextEdit()

self.main\_layout=QtGui.QVBoxLayout()

#Labels

self.server=QtGui.QLabel("Server")

self.position=QtGui.QLabel("Position")

self.character=QtGui.QLabel("Character")

self.score0=QtGui.QLabel("Player0:")

self.score1=QtGui.QLabel("Player1:")

self.drag\_1=QtGui.QLabel() #blue dragon

self.drag\_2=QtGui.QLabel() #red dragon

self.score\_zero=QtGui.QLabel()

self.score\_one=QtGui.QLabel()

self.top\_left\_dec=QtGui.QLabel()

self.top\_right\_dec=QtGui.QLabel()

self.sides\_label=QtGui.QLabel()

self.shut\_label=QtGui.QLabel()

self.shut\_pix = QtGui.QPixmap("Close.png")

self.shut\_label.setPixmap(self.shut\_pix) #the label next to the shutting down button will be a close image

# self.shut\_label.setAutoFillBackground(True)

self.play\_again\_label=QtGui.QLabel()

self.play\_a\_pix=QtGui.QPixmap("PlayAgain.png")

self.play\_again\_label.setPixmap(self.play\_a\_pix) #the label next to this one will be a play\_again image

#text boxes

self.pos\_field=QtGui.QLineEdit()

self.char\_field=QtGui.QLineEdit()

self.server\_field=QtGui.QLineEdit()

self.server\_field.setPlaceholderText("Enter Server")

#images/pixmaps

self.S\_pix = QtGui.QPixmap('S.gif').scaled(80, 80, QtCore.Qt.KeepAspectRatio) #the S image, scaled to be a bit bigger

self.O\_pix = QtGui.QPixmap('O.gif').scaled(80, 80, QtCore.Qt.KeepAspectRatio) #the O image

self.Blank = QtGui.QPixmap('blank.gif').scaled(80, 80, QtCore.Qt.KeepAspectRatio) #the blank image

self.dragon\_blue=QtGui.QPixmap("TribalBlue.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #a pixmap with the dragon image that has been resized to fit in thewindow

self.player\_div=QtGui.QPixmap("Heading\_11.png").scaled(1200, 100, QtCore.Qt.KeepAspectRatio) #this image allocates to you the sides of the players

self.red\_dec=QtGui.QPixmap("redc.png").scaled(70, 70, QtCore.Qt.KeepAspectRatio) #the red corner decoration

self.blue\_dec=QtGui.QPixmap("bluec.png").scaled(70, 70, QtCore.Qt.KeepAspectRatio) #the blue corner decoration

self.dragon\_red=QtGui.QPixmap("TribalRed.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #same here, but with the red dragon

self.drag\_1.setPixmap(self.dragon\_blue)

self.drag\_2.setPixmap(self.dragon\_red)

self.top\_left\_dec.setPixmap(self.blue\_dec)

self.top\_right\_dec.setPixmap(self.red\_dec)

self.sides\_label.setPixmap(self.player\_div)

#player zero scores

'''Score\_PlayerNumber\_Score'''

self.score\_0\_0 = QtGui.QPixmap("recblue\_0.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #the blue record image, score\_(player)0\_score(0)

self.score\_0\_1 = QtGui.QPixmap("recblue\_1.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #player 0 score pixmap when the score is 1

self.score\_0\_2 = QtGui.QPixmap("recblue\_2.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_3 = QtGui.QPixmap("recblue\_3.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_4 = QtGui.QPixmap("recblue\_4.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_5 = QtGui.QPixmap("recblue\_5.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_6 = QtGui.QPixmap("recblue\_6.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_7 = QtGui.QPixmap("recblue\_7.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_0\_8 = QtGui.QPixmap("recblue\_8.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

#the score: pixmap dictionary for player0 scores

'''The score is the key and the pixmap is the value, this will be used for displaying the score for each player based on the score they have'''

self.p0\_scores = { 0:self.score\_0\_0 , 1:self.score\_0\_1 , 2: self.score\_0\_2 , 3: self.score\_0\_3 , 4:self.score\_0\_4 , 5:self.score\_0\_5, 6:self.score\_0\_6, 7:self.score\_0\_7, 8:self.score\_0\_8}

#player one scores

'''Score\_PlayerNumber\_Score'''

self.score\_1\_0=QtGui.QPixmap("recred\_0.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #the red record image, the same as the above scores pixmpas

self.score\_1\_1=QtGui.QPixmap("recred\_1.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #the red record image

self.score\_1\_2=QtGui.QPixmap("recred\_2.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_3=QtGui.QPixmap("recred\_3.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_4=QtGui.QPixmap("recred\_4.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_5=QtGui.QPixmap("recred\_5.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_6=QtGui.QPixmap("recred\_6.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_7=QtGui.QPixmap("recred\_7.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

self.score\_1\_8=QtGui.QPixmap("recred\_8.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio)

#the score: pixmap dictionary for player1 scores

'''The score is the key and the pixmap is the value, this will be used for displaying the score for each player based on the score they have'''

self.p1\_scores = { 0:self.score\_1\_0 , 1:self.score\_1\_1 , 2: self.score\_1\_2 , 3: self.score\_1\_3 , 4:self.score\_1\_4 , 5:self.score\_1\_5, 6:self.score\_1\_6, 7:self.score\_1\_7, 8:self.score\_1\_8} #each and every score with it's corresponding pixmap

#the scores images initially to be both zero

self.score\_zero.setPixmap(self.score\_0\_0)

self.score\_one.setPixmap(self.score\_1\_0) #setting the pixmaps accordingly

##zero1= 0,1, zero0=0,0 , their positions as they will be put in the grid##

'''I used the Qlabel class to make these buttons clickable'''

#first row

self.zero0 = ClickableQLabel(self) #clickable labels to hold pixmap,

self.zero1 = ClickableQLabel(self) #" " " " topmiddle

self.zero2 = ClickableQLabel(self)

self.zero3 = ClickableQLabel(self)

#second row

self.one0 = ClickableQLabel(self)

self.one1 = ClickableQLabel(self)

self.one2 = ClickableQLabel(self)

self.one3 = ClickableQLabel(self)

#third row

self.two0 = ClickableQLabel(self)

self.two1 = ClickableQLabel(self)

self.two2 = ClickableQLabel(self)

self.two3 = ClickableQLabel(self)

#fourth row

self.three0 = ClickableQLabel(self)

self.three1 = ClickableQLabel(self)

self.three2 = ClickableQLabel(self)

self.three3 = ClickableQLabel(self)

#the list of these labels:

self.empty\_labels= [ self.zero0, self.zero1 , self.zero2 , self.zero3 , self.one0 ,self.one1 , self.one2 ,self.one3 , self.two0 ,self.two1 , self.two2 , self.two3 , self.three0 , self.three1, self.three2, self.three3 ]

'''These are all the empty clickable image labels'''

#buttons

self.connect\_button=QtGui.QPushButton("Connect")

self.play\_pix=QtGui.QPixmap("play\_again.png").scaled(70, 70, QtCore.Qt.KeepAspectRatio)

#self.play\_again=ClickableQLabel(self)

#self.play\_again.setPixmap(self.play\_pix)

#the shut button for closing the game ang the how to play

self.how\_to\_button=ClickableQLabel(self)

self.how\_to\_pix=QtGui.QPixmap("HowToPlay.png").scaled(100, 100, QtCore.Qt.KeepAspectRatio) #the how\_to\_play button / image

self.how\_to\_button.setPixmap(self.how\_to\_pix)

self.shut\_button=ClickableQLabel(self) #the button(image)for closing the game

self.shut\_pix=QtGui.QPixmap("shut.png").scaled(70, 70, QtCore.Qt.KeepAspectRatio)

self.shut\_button.setPixmap(self.shut\_pix)

self.how\_to\_lpix=QtGui.QPixmap("HowTo.png").scaled(250, 100, QtCore.Qt.KeepAspectRatio) #the pixmap for the how to play label image

self.how\_to\_label.setPixmap(self.how\_to\_lpix)

'''Adding widgets to layouts'''

#top\_hbox

# self.top\_hbox.addWidget(self.empty1) #empty labels to place the image in the middle

self.top\_hbox.addWidget(self.sides\_label)

self.top\_hbox.addWidget(self.empty2)

self.top\_hbox.addStretch(100)

#shutdown hbox, for the shutdown button

#self.bottom\_hbox.addWidget(self.play\_again) #button

#self.bottom\_hbox.addWidget(self.play\_again\_label) #play again label

self.bottom\_hbox.addWidget(self.shut\_button) #button

self.bottom\_hbox.addWidget(self.shut\_label) #shut down game label

self.bottom\_hbox.addWidget(self.how\_to\_button)

self.bottom\_hbox.addWidget(self.how\_to\_label)

self.bottom\_hbox.addStretch(150)

#adding widgets to the grid\_one layout

self.grid\_one.addWidget(self.zero0,0,0) #adding the blank images to their corresponding positions

self.grid\_one.addWidget(self.zero1,0,1)

self.grid\_one.addWidget(self.zero2,0,2)

self.grid\_one.addWidget(self.zero3,0,3)

#row 2

self.grid\_one.addWidget(self.one0,1,0)

self.grid\_one.addWidget(self.one1,1,1)

self.grid\_one.addWidget(self.one2,1,2)

self.grid\_one.addWidget(self.one3,1,3)

#row3

self.grid\_one.addWidget(self.two0,2,0)

self.grid\_one.addWidget(self.two1,2,1)

self.grid\_one.addWidget(self.two2,2,2)

self.grid\_one.addWidget(self.two3,2,3)

#row4

self.grid\_one.addWidget(self.three0,3,0)

self.grid\_one.addWidget(self.three1,3,1)

self.grid\_one.addWidget(self.three2,3,2)

self.grid\_one.addWidget(self.three3,3,3)

#conn\_hbox

self.conn\_hbox.addWidget(self.drag\_1)

self.conn\_hbox.addWidget(self.server) #the server label

self.conn\_hbox.addWidget(self.server\_field) #the line edit for entering the IP adress

self.conn\_hbox.addWidget(self.connect\_button)

self.conn\_hbox.addWidget(self.drag\_2)

#score\_hbox

#self.score\_hbox.addWidget(self.score0)

self.score\_hbox.addWidget(self.score\_zero)

#self.score\_hbox.addWidget(self.score1)

self.score\_hbox.addStretch(40)

self.score\_hbox.addWidget(self.score\_one)

#self.score\_one.setText("NIl")

'''layout widgets'''

'''Here we take a layout and put it as a widget so it could be added to another layout'''

#the top hbox

self.top\_hbox\_wid=QtGui.QWidget()

self.top\_hbox\_wid.setLayout(self.top\_hbox)

#for the shut and play again buttons

self.bottom\_hbox\_wid=QtGui.QWidget()

self.bottom\_hbox\_wid.setLayout(self.bottom\_hbox)

#grid\_one

#the grid with the images

self.grid\_one\_wid=QtGui.QWidget() #the grid widget

self.grid\_one\_wid.setLayout(self.grid\_one) #setting the layout to be grid\_one

#vbox1

self.vbox1\_wid=QtGui.QWidget() #the vbox1

self.vbox1\_wid.setLayout(self.vbox\_1) #making vbox1 layout a widget

#bottom\_hbox

#self.top\_hbox\_wid=QtGui.QWidget()

#self.top\_hbox\_wid.setLayout(self.top\_hbox) #same thing here

#conn\_hbox

self.conn\_hbox\_wid=QtGui.QWidget()

self.conn\_hbox\_wid.setLayout(self.conn\_hbox)

#score\_hbox

self.score\_hbox\_wid=QtGui.QWidget()

self.score\_hbox\_wid.setLayout(self.score\_hbox)

#adding the vbox1 and grid\_one layout widgets to their hbox,

'''I added the text edit box instead of the vbox1'''

#this one takes the layout widgets and adds them

self.grid\_vbox1\_layout.addWidget(self.grid\_one\_wid)

self.grid\_vbox1\_layout.addWidget(self.edit\_box)

#grid\_vbox1\_layout

self.grid\_vbox1\_layout\_wid=QtGui.QWidget() #the same layouts out with other layout widgets is made a widget for, so we could add it to the main layout

self.grid\_vbox1\_layout\_wid.setLayout(self.grid\_vbox1\_layout)

'''The main layout'''

#adding the other layouts(widgets) to the main Vbox layout

self.main\_layout.addWidget(self.top\_hbox\_wid)

self.main\_layout.addWidget(self.score\_hbox\_wid)

self.main\_layout.addWidget(self.conn\_hbox\_wid)

self.main\_layout.addWidget(self.grid\_vbox1\_layout\_wid)

self.main\_layout.addWidget(self.bottom\_hbox\_wid)

self.setLayout(self.main\_layout)

#connecting to the server

self.connect\_button.clicked.connect(self.Connect\_client)

#if zero1 is clicked, then the zero1\_Clicked() method will be called

#connections to be modified

self.connect(self.zero0, SIGNAL('clicked()'), self.Clicked\_zero0)

self.connect(self.zero1, SIGNAL('clicked()'), self.Clicked\_zero1)

self.connect(self.zero2, SIGNAL('clicked()'), self.Clicked\_zero2)

self.connect(self.zero3, SIGNAL('clicked()'), self.Clicked\_zero3)

self.connect(self.one0, SIGNAL('clicked()'), self.one0\_Clicked)

self.connect(self.one1, SIGNAL('clicked()'), self.one1\_Clicked)

self.connect(self.one2, SIGNAL('clicked()'), self.one2\_Clicked)

self.connect(self.one3, SIGNAL('clicked()'), self.one3\_Clicked)

#row3 connections

self.connect(self.two0, SIGNAL('clicked()'), self.two0\_Clicked)

self.connect(self.two1, SIGNAL('clicked()'), self.two1\_Clicked)

self.connect(self.two2, SIGNAL('clicked()'), self.two2\_Clicked)

self.connect(self.two3, SIGNAL('clicked()'), self.two3\_Clicked)

#row4 connections

self.connect(self.three0, SIGNAL('clicked()'), self.three0\_Clicked)

self.connect(self.three1, SIGNAL('clicked()'), self.three1\_Clicked)

self.connect(self.three2, SIGNAL('clicked()'), self.three2\_Clicked)

self.connect(self.three3, SIGNAL('clicked()'), self.three3\_Clicked)

#other buttons connection

self.connect(self.shut\_button, SIGNAL('clicked()'), self.shut\_win) #when the shut button is clicked, the shut\_win module is ran

self.connect(self.how\_to\_button, SIGNAL('clicked()'), self.how\_to\_play)

#the play thread

self.mess\_thread = LoopThread() #the threat object

self.mess\_thread.message\_signal.connect(self.handle\_message) #connecting the signal to the handle\_message method

def play\_sound(self,clip): #plays the clicked sound

self.mediaObject = Phonon.MediaObject(self)

self.audioOutput = Phonon.AudioOutput(Phonon.MusicCategory, self)

Phonon.createPath(self.mediaObject, self.audioOutput)

# self.mediaObject.stateChanged.connect(self.handleStateChanged)

self.mediaObject.setCurrentSource(Phonon.MediaSource(clip))

self.mediaObject.play()

#this function shows the message whether it's your move

def show\_msg(self,pic,showtime):

self.splash = QtGui.QSplashScreen(QtGui.QPixmap(pic))

# SplashScreen will be in the center of the screen by default.

self.splash.show()

# Close the SplashScreen after the specified secs (ms)

QtCore.QTimer.singleShot(showtime, self.splash.close)

def clear\_interface(self): #this method clears the images and set them to blank

#the list of the clickable labels just incase i want to aplly something to all of them

'''setting the initial scores'''

self.score\_zero.setPixmap(self.score\_0\_0)

self.score\_one.setPixmap(self.score\_1\_0)

'''reseting all the pixmaps to blank'''

#row1

self.zero0.setPixmap(self.Blank)

self.zero1.setPixmap(self.Blank)

self.zero2.setPixmap(self.Blank)

self.zero3.setPixmap(self.Blank)

#row2

self.one0.setPixmap(self.Blank)

self.one1.setPixmap(self.Blank)

self.one2.setPixmap(self.Blank)

self.one3.setPixmap(self.Blank)

#row3

self.two0.setPixmap(self.Blank)

self.two1.setPixmap(self.Blank)

self.two2.setPixmap(self.Blank)

self.two3.setPixmap(self.Blank)

#row4

self.three0.setPixmap(self.Blank)

self.three1.setPixmap(self.Blank)

self.three2.setPixmap(self.Blank)

self.three3.setPixmap(self.Blank)

#adding the blank to all the labels initially

def Connect\_client(self):

self.mess\_thread.connecter(self.server\_field.displayText()) #connecting using the thread because the connection is a loop on it's own

self.server\_field.setText('')

self.clear\_interface() #not neccessary cause new game will do this

self.mess\_thread.start() #start the playloop

self.show\_msg("Connected.png",1500) #show the connected image for 1.5 s

self.play\_sound("Connected.wav") #play the connected sound

#row1

def how\_to\_play(self):

QtGui.QMessageBox.information(self, "SOS game", "SOS is a game similar to tic-tac-toe, where by you fill in boxes with either S or O to make the word SOS, which gives you a point. The player with the most points is a winner.\nHow to play: To make a move, you click on an empty box (white) and then click either an \'s\' or an \'o\' to play." )

#some information about how to play

def handle\_message(self,msg):

print(msg)

self.edit\_box.setText( self.edit\_box.toPlainText() + "\n" + msg) #displaying the server messages on the text edit

self.checking=msg

if msg[:3]=="new": #if the first 3 ketters of msg are 'new' , then we have a 'new game,N' message.. that tells us that it's a new game

#clearing the board

self.clear\_interface()

#QtGui.QMessageBox.information(self, "SOS", "A new has started.\nHINT: Click on a box to make a move" )

self.show\_msg('NewGame.png',1000)

if msg=="your move": #if the server sends me that it's my move

print(msg)

time.sleep(1) #wait for a second before a notification

##QtGui.QMessageBox.information( self, "SOS", "Your Move" ) #a dialog that tells you that it's your move

self.show\_msg('YourMove.png',1500) #show the your move picture

if msg[:9]=="opponents": #if the server sent 'opponents move', we know it's the opponents move

#Tell the user it's the other players move

self.show\_msg('OpponentsMove.png',1500)

time.sleep(1)

if msg[:3]=="gam":

self.show\_msg("GameOver\_note.png",2000) #show that it's game over

self.split\_msg= msg.split(',') # now we havethe message as a list [GameOver,W,S0,S1], W- For the winner, S0- Player0 score, S1- Player1 score

S0=self.split\_msg[2] #player1's score

S1=self.split\_msg[3] #player2's score

Winner=self.split\_msg[1]

time.sleep(1.5)

if Winner=="T": #T is sent when it's a tie, so we'll print it to both the clients that it's tie

self.show\_msg("Tie.png",3000)

time.sleep(1.5)

elif Winner =="0":

self.show\_msg('Player0\_wins.png',3000)

time.sleep(1.5)

elif Winner == "1":

self.show\_msg('Player1\_wins.png',3000)

time.sleep(1.5)

if msg[:5]=="valid": #if the first FIVE elements of the string are 'valid', we know that it's valid move so we further proceed

self.msg\_list=msg.split(',') # we have a list, [valid move,P,C,S1,S2]

self.P = self.msg\_list[1] #the position

#print(self.P)

self.C = self.msg\_list[2] #the charactor

#print(self.C)

self.board[int(self.P)] = str(self.C) #replace the charactor in the specified position in the list with the specified charactor

S0=self.msg\_list[3] #player0's score

S1=self.msg\_list[4] #player1's score

if self.C=="S": #if the character was S

self.empty\_labels[int(self.P)].setPixmap(self.S\_pix)

elif self.C=="O": # if the character was O

self.empty\_labels[int(self.P)].setPixmap(self.O\_pix) #change the image to O

#print('\n')

#print("Player 0:",S0)

#print("Palyer 1:",S1)

self.score\_zero.setPixmap(self.p0\_scores[int(S0)]) #displaying the appropiate score, by taking the score and getting the corresponding pixmap from the scores dictionary

self.score\_one.setPixmap(self.p1\_scores[int(S1)]) #same here

if msg == "invalid move": #if the move is considered to be invalid by the server, then We'll notify the user

self.play\_sound("invalid\_move.wav")

QtGui.QMessageBox.information( self, "SOS", "Invalid move. Please try a different position." ) #a dialog that tells you that it's an invalid move

#a dialog that tells you that it's your move

#self.show\_msg('Invalidmove.png',1500)

if msg[:4] =='play': #now we know we have to prompt users to play a new game

#QtGui.QMessageBox.information( self, "SOS", "Game Over \n\n(Press Play Again to restart)" )

reply = QtGui.QMessageBox.question(None,'SOS','Do you want to play again?',QtGui.QMessageBox.Yes, QtGui.QMessageBox.No)

if reply == QtGui.QMessageBox.Yes:

self.mess\_thread.message\_sender("y") #send a yes if the user wants to play again

elif reply== QtGui.QMessageBox.No:

self.mess\_thread.message\_sender("n") #else, send a no

if msg =='exit game':

print('Game Over')

def play\_again\_clicked(self):

reply = QtGui.QMessageBox.question(None,'Confirm Restart','Are you sure you want to quit and restart?',QtGui.QMessageBox.Yes, QtGui.QMessageBox.No)

if reply == QtGui.QMessageBox.Yes:

self.clear\_interface() #clears the interface because the play\_again button has been selected

elif reply== QtGui.QMessageBox.No:

pass

def make\_move(self,pos): #for making moves, using the thread it's called everytime a button is clicked

self.play\_sound("clicked.wav") #plays the clicked sound

if self.checking=="your move":

self.grid2wid.show()

self.sd.mousePressEvent= self.s\_play

self.sdo.mousePressEvent= self.o\_play

#if self.checking=="invalid move":

#self.play\_sound("invalid\_move.wav")

#print("Invalid Move")

#print('\n')

#self.show\_msg("Invalidmove.png",1500)

##time.sleep(1)

if self.checking[:9]=="opponents":

self.show\_msg('OpponentsMove',1000)

self.pos=pos #setting self.pos to the position of the clicked space

def o\_play(self,event):

self.char="O"

self.grid2wid.close()

self.mess\_thread.message\_sender(str(self.pos)+","+str(self.char).upper())

def s\_play(self,event):

self.char="S"

self.grid2wid.close()

self.mess\_thread.message\_sender(str(self.pos)+","+str(self.char).upper())

def shut\_win(self):

reply = QtGui.QMessageBox.question(None,'Confirm Exit...','Are you sure you want exit?',QtGui.QMessageBox.Yes, QtGui.QMessageBox.No) #asking if the user really wants to exit

if reply == QtGui.QMessageBox.Yes: #if they click yes, close the window

self.close()

elif reply== QtGui.QMessageBox.No:

pass

#you can only make a move if it's your move

def Clicked\_zero0(self):

print("zero")

pos=0 #the position if this button was clicked

self.make\_move(pos)

#self.show\_msg()

def Clicked\_zero1(self):

pos=1

self.make\_move(pos)

def Clicked\_zero2(self):

pos=2

self.make\_move(pos)

def Clicked\_zero3(self):

pos=3

self.make\_move(pos)

#row2

def one0\_Clicked(self):

pos=4

self.make\_move(pos)

def one1\_Clicked(self):

pos=5

self.make\_move(pos)

def one2\_Clicked(self):

pos=6

self.make\_move(pos)

def one3\_Clicked(self):

pos=7

self.make\_move(pos)

#row3

def two0\_Clicked(self):

pos=8

self.make\_move(pos)

def two1\_Clicked(self):

pos=9

self.make\_move(pos)

def two2\_Clicked(self):

pos=10

self.make\_move(pos)

def two3\_Clicked(self):

pos=11

self.make\_move(pos)

#row4

def three0\_Clicked(self):

pos=12

self.make\_move(pos)

def three1\_Clicked(self):

pos=13

self.make\_move(pos)

def three2\_Clicked(self):

pos=14

self.make\_move(pos)

def three3\_Clicked(self):

pos=15

self.make\_move(pos)

def main():

main=Main()

main.clear\_interface() #so it could show

#input('Press enter to exit.')

main.show()

sys.exit(app.exec\_())

main()