Public Class FrmPlayTTT

' <=== Defining variables ===>

Dim GameBoard(,) As Short = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}

Dim PlayerTurn As Boolean = True ' True = O, False = X

Dim GameInProgress As Boolean = False

Dim TurnNumber As Integer = 0

Dim Player1Name As String = "Player 1"

Dim Player2Name As String = "Player 2"

Private Sub ParseCommandLineArgs()

If ModOpeningFiles.LoadedReplayAlready = False Then

Dim Arguments As String() = Environment.GetCommandLineArgs()

ModOpeningFiles.LoadedReplayAlready = True

Try

' MsgBox(Arguments(1))

If Arguments(1).Split(".")(Arguments(1).Split(".").Count() - 1) = "T3Replay" Then

MsgBox("Loading replayer...", , "SUCCESS")

FrmReplayViewer.Show()

FrmReplayViewer.OnStartWithFile(Arguments(1))

Me.Close()

Else

MsgBox("Not a '\*.T3Replay' file", , "FAILURE")

End If

Catch ERR As Exception

If Not (TypeOf ERR Is IndexOutOfRangeException) Then

MsgBox("Something went wrong..." & ERR.ToString(), , "FAILURE")

MsgBox("Loading game window for safety", , "FAILURE")

If Application.OpenForms().OfType(Of FrmReplayViewer).Any Then

Me.Show()

FrmReplayViewer.Close()

End If

End If

End Try

End If

End Sub

'

Private Sub FrmPlayTTT\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

ParseCommandLineArgs()

If My.Settings.SaveFolder = "" Then My.Settings.SaveFolder = Application.StartupPath

PnlGameControls.Enabled = True

PnlGameButtons.Enabled = False

LblLastMove.Text = "Last Move: N/A"

LblLastMove.ForeColor = Color.Black

ResetGameGrid()

PnlInfo.Enabled = False

BtnEndGame.Enabled = False

TurnNumber = 0

End Sub

' <=== Control buttons ===>

''' <summary>

''' Starts the game.

''' </summary>

Private Sub StartGameButtons\_Click(sender As Object, e As EventArgs) Handles BtnStartGame.Click, NewGameToolStripMenuItem.Click, CntxtNewGameItem.Click

' Check if a game is currently being played

If GameInProgress = True Then

Dim Choice As Integer = MsgBox("Are you sure you want to end the game?", MsgBoxStyle.YesNo, "Game in progress!")

If Choice <> 6 Then

' Continue the game

Exit Sub

End If

' Complain about their life choices

If PlayerTurn = True Then

MsgBox("I'm very disappointed in you, " & Player1Name)

Else

MsgBox("I'm very disappointed in you, " & Player2Name)

End If

End If

If Player1Name = "Player 1" And Player2Name = "Player 2" Then

GetPlayerNames()

End If

PnlGameButtons.Enabled = True

PnlInfo.Enabled = True

BtnEndGame.Enabled = True

ResetPlayerTurn()

BtnStartGame.Enabled = False

GameBoard = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}

ResetGameGrid()

'CheckGrid()

ModSavingFiles.ResetFileSettings()

GameInProgress = True

TurnNumber = 0

If My.Settings.CrossesGoFirst = True Then

MsgBox("Game begun. It's O's turn!")

Else

MsgBox("Game begun. It's X's turn!")

End If

End Sub

''' <summary>

''' Allows players to re-enter their names

''' </summary>

Private Sub BtnChangeNames\_Click(sender As Object, e As EventArgs) Handles BtnChangeNames.Click, CntxtChangeNamesItem.Click

GetPlayerNames()

End Sub

''' <summary>

''' Memed

''' </summary>

Private Sub GetHelpOnlineToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpOnlineToolStripMenuItem.Click

MsgBox("There is no one to help you online. Sorry", MsgBoxStyle.OkOnly, "HAHAHAHAHAHAHAHAHAHA")

End Sub

''' <summary>

''' Shows Help

''' </summary>

Private Sub GetHelpToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpToolStripMenuItem.Click, CntxtHelpItem.Click

FrmHelp.Show()

End Sub

''' <summary>

''' Allows for setting changes

''' </summary>

Private Sub SettingsToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles SettingsToolStripMenuItem.Click, CntxtSettingsItem.Click

FrmSettings.Show()

Me.Close()

End Sub

''' <summary>

''' Ends the game, but isn't happy with it

''' </summary>

Private Sub BtnEndGame\_Click(sender As Object, e As EventArgs) Handles BtnEndGame.Click, CntxtEndGameItem.Click

Dim Choice As Integer = MsgBox("Are you sure you want to end the game?", MsgBoxStyle.YesNo, "Game in progress!")

If Choice <> 6 Then

' Continue the game

Exit Sub

End If

' Complain!

If PlayerTurn = True Then

MsgBox("I'm very disappointed in you, " & Player1Name)

Else

MsgBox("I'm very disappointed in you, " & Player2Name)

End If

GameBoard = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}

GameInProgress = False

PnlGameButtons.Enabled = False

PnlInfo.Enabled = False

ResetPlayerTurn()

BtnEndGame.Enabled = False

BtnStartGame.Enabled = True

ResetGameGrid()

ModSavingFiles.ResetFileSettings()

TurnNumber = 0

End Sub

''' <summary>

''' Opens the replay viewer and closes the game

''' </summary>

Private Sub LoadPreviousGameToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles LoadPreviousGameToolStripMenuItem.Click

FrmReplayViewer.Show()

Me.Close()

End Sub

' <=== Non-Button based Subs / Functions ===>

''' <summary>

''' Asks the players for their names. Is purely for prettiness and doesn't affect gameplay. 10Char limit.

''' </summary>

Private Sub GetPlayerNames()

' Player 1's name

Dim Flag As Boolean = False

While Flag = False

Player1Name = InputBox("O's name?", "Player Names", Player1Name)

If Len(Player1Name) > 10 Then

MsgBox(Player1Name & " is too long. Please use a name that is 10 characters or less", MsgBoxStyle.OkOnly, "LENGTH TOO GREAT")

Flag = False

ElseIf Player1Name = "" Then

Player1Name = "Player 1"

Flag = True

Else

LblNameO.Text = Player1Name & " - O"

Flag = True

End If

End While

' Player 2's name

Flag = False

While Flag = False

Player2Name = InputBox("X's name?", "Player Names", Player2Name)

If Len(Player2Name) > 10 Then

MsgBox(Player2Name & " is too long. Please use a name that is 10 characters or less", MsgBoxStyle.OkOnly, "LENGTH TOO GREAT")

Flag = False

ElseIf Player2Name = "" Then

Player2Name = "Player 2"

Flag = True

Else

LblNameX.Text = Player2Name & " - X"

Flag = True

End If

End While

End Sub

''' <summary>

''' Checks if anyone has won yet

''' </summary>

Private Sub CheckForVictory()

' If a diagonal, row or column add to 3, O wins

' If a diagonal, row or column add to 30, X wins

Dim Total As Integer = 0

Dim Winner As String = "None"

' \/ Checks verticals

For Looper1 = 0 To 2

Total = 0

For Looper2 = 0 To 2

Total += GameBoard(Looper1, Looper2)

Next

If Total = 3 Then ' O wins

Winner = "O"

ElseIf Total = 27 Then ' X wins

Winner = "X"

End If

Next

' \/ Checks horizontals

For Looper2 = 0 To 2

Total = 0

For Looper1 = 0 To 2

Total += GameBoard(Looper1, Looper2)

Next

If Total = 3 Then ' O wins

Winner = "O"

ElseIf Total = 27 Then ' X wins

Winner = "X"

End If

Next

' \/ Checks diagonals

If (GameBoard(0, 0) + GameBoard(1, 1) + GameBoard(2, 2)) = 3 Or \_

(GameBoard(2, 0) + GameBoard(1, 1) + GameBoard(0, 2)) = 3 Then ' O wins

Winner = "O"

ElseIf (GameBoard(0, 0) + GameBoard(1, 1) + GameBoard(2, 2)) = 27 Or \_

(GameBoard(2, 0) + GameBoard(1, 1) + GameBoard(0, 2)) = 27 Then ' X wins

Winner = "X"

End If

' \/ evaluates the result

If Winner = "O" Then

'Debug.WriteLine(Player1Name.ToUpper & " WINS")

LblLastMove.Text = "O Wins!"

LblLastMove.ForeColor = Color.DarkBlue

MsgBox(Player1Name & " is the winner!" & Environment.NewLine & "Congratulations!!!", MsgBoxStyle.OkOnly, Player1Name.ToUpper & " WINS")

EndGameAfterVictoryOrDraw()

ElseIf Winner = "X" Then

'Debug.WriteLine(Player2Name.ToUpper & " WINS")

LblLastMove.Text = "X Wins!"

LblLastMove.ForeColor = Color.Maroon

MsgBox(Player2Name & " is the winner!" & Environment.NewLine & "Congratulations!!!", MsgBoxStyle.OkOnly, Player2Name.ToUpper & " WINS")

EndGameAfterVictoryOrDraw()

Else

' Check for a draw

If (GameBoard(0, 0) <> 0) And (GameBoard(0, 1) <> 0) And (GameBoard(0, 2) <> 0) And \_

(GameBoard(1, 0) <> 0) And (GameBoard(1, 1) <> 0) And (GameBoard(1, 2) <> 0) And \_

(GameBoard(2, 0) <> 0) And (GameBoard(2, 1) <> 0) And (GameBoard(2, 2) <> 0) Then

' It's a draw!

'Debug.WriteLine("IT'S A DRAW")

LblLastMove.Text = "It's a draw!"

LblLastMove.ForeColor = Color.Black

MsgBox(Player1Name & " and " & Player2Name & " drew.", MsgBoxStyle.OkOnly, "Maybe another time...")

EndGameAfterVictoryOrDraw()

Else

'Debug.WriteLine("Normal move")

End If

End If

End Sub

''' <summary>

''' Saves the game just played and resets the gameboard after the game is fully completed

''' </summary>

''' <remarks></remarks>

Private Sub EndGameAfterVictoryOrDraw()

ModSavingFiles.CreateNewReplaySave()

ModSavingFiles.ResetFileSettings()

GameInProgress = False

PnlGameButtons.Enabled = False

PnlInfo.Enabled = False

GameBoard = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}

BtnStartGame.Enabled = True

BtnEndGame.Enabled = False

ResetPlayerTurn()

TurnNumber = 0

End Sub

''' <summary>

''' Resets all the buttons to their default states

''' </summary>

Private Sub ResetGameGrid()

Btn00.Text = ""

Btn00.ForeColor = Color.Black

Btn01.Text = ""

Btn01.ForeColor = Color.Black

Btn02.Text = ""

Btn02.ForeColor = Color.Black

Btn10.Text = ""

Btn10.ForeColor = Color.Black

Btn11.Text = ""

Btn11.ForeColor = Color.Black

Btn12.Text = ""

Btn12.ForeColor = Color.Black

Btn20.Text = ""

Btn20.ForeColor = Color.Black

Btn21.Text = ""

Btn21.ForeColor = Color.Black

Btn22.Text = ""

Btn22.ForeColor = Color.Black

End Sub

''' <summary>

''' Writes the current values of the game grid to the console for debugging purposes

''' </summary>

'Private Sub CheckGrid()

' Debug.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_")

' Debug.WriteLine("|{0,3}|{1,3}|{2,3}|", GameBoard(0, 0), GameBoard(1, 0), GameBoard(2, 0))

' Debug.WriteLine("|-----------|")

' Debug.WriteLine("|{0,3}|{1,3}|{2,3}|", GameBoard(0, 1), GameBoard(1, 1), GameBoard(2, 1))

' Debug.WriteLine("|-----------|")

' Debug.WriteLine("|{0,3}|{1,3}|{2,3}|" & Environment.NewLine, GameBoard(0, 2), GameBoard(1, 2), GameBoard(2, 2))

'End Sub

''' <summary>

''' Changes the text \_colour\_ of the button and LblLastMove to reflect which player has chosen what

''' </summary>

''' <param name="Sender">The object that called the sub</param>

Private Sub ChangeColour(ByRef Sender As Object)

If Sender.Text = "O" Then

' Make it Blue (0, 0, 139)

LblLastMove.ForeColor = Color.DarkBlue

Sender.ForeColor = Color.DarkBlue

LblPlayerTurn.ForeColor = Color.Maroon

ElseIf Sender.Text = "X" Then

' Make it Red (128, 0, 0)

LblLastMove.ForeColor = Color.Maroon

Sender.ForeColor = Color.Maroon

LblPlayerTurn.ForeColor = Color.DarkBlue

Else

LblLastMove.ForeColor = Color.Black

Sender.ForeColor = Color.Black

LblPlayerTurn.ForeColor = Color.Black

End If

End Sub

''' <summary>

''' Resets the game to the correct turn

''' </summary>

Private Sub ResetPlayerTurn()

If My.Settings.CrossesGoFirst Then

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

LblPlayerTurn.ForeColor = Color.DarkBlue

Else

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

LblPlayerTurn.ForeColor = Color.Maroon

End If

LblLastMove.Text = "Last Move: N/A"

LblLastMove.ForeColor = Color.Black

End Sub

' <=== Grid Button Subs ===>

' << On Click >>

''' <summary>

''' Changes the relevant grid square to reflect the move just made, then checks for a winner

''' </summary>

Private Sub Btn00\_Click(sender As Object, e As EventArgs) Handles Btn00.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(0, 0) = 1

sender.Text = "O"

LblLastMove.Text = "O: 0,0"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(0, 0) = 9

sender.Text = "X"

LblLastMove.Text = "X: 0,0"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn10\_Click(sender As Object, e As EventArgs) Handles Btn10.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(1, 0) = 1

sender.Text = "O"

LblLastMove.Text = "O: 1,0"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(1, 0) = 9

sender.Text = "X"

LblLastMove.Text = "X: 1,0"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn20\_Click(sender As Object, e As EventArgs) Handles Btn20.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(2, 0) = 1

sender.Text = "O"

LblLastMove.Text = "O: 2,0"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(2, 0) = 9

sender.Text = "X"

LblLastMove.Text = "X: 2,0"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn01\_Click(sender As Object, e As EventArgs) Handles Btn01.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(0, 1) = 1

sender.Text = "O"

LblLastMove.Text = "O: 0,1"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(0, 1) = 9

sender.Text = "X"

LblLastMove.Text = "X: 0,1"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn11\_Click(sender As Object, e As EventArgs) Handles Btn11.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(1, 1) = 1

sender.Text = "O"

LblLastMove.Text = "O: 1,1"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(1, 1) = 9

sender.Text = "X"

LblLastMove.Text = "X: 1,1"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn21\_Click(sender As Object, e As EventArgs) Handles Btn21.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(2, 1) = 1

sender.Text = "O"

LblLastMove.Text = "O: 2,1"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(2, 1) = 9

sender.Text = "X"

LblLastMove.Text = "X: 2,1"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn02\_Click(sender As Object, e As EventArgs) Handles Btn02.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(0, 2) = 1

sender.Text = "O"

LblLastMove.Text = "O: 0,2"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(0, 2) = 9

sender.Text = "X"

LblLastMove.Text = "X: 0,2"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn12\_Click(sender As Object, e As EventArgs) Handles Btn12.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(1, 2) = 1

sender.Text = "O"

LblLastMove.Text = "O: 1,2"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(1, 2) = 9

sender.Text = "X"

LblLastMove.Text = "X: 1,2"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

Private Sub Btn22\_Click(sender As Object, e As EventArgs) Handles Btn22.Click

If sender.Text = "O" Or sender.Text = "X" Then

MsgBox("This box has already been filled. You can't re-fill it", MsgBoxStyle.OkOnly, "Already played here")

Else

If PlayerTurn = True Then ' O's go

GameBoard(2, 2) = 1

sender.Text = "O"

LblLastMove.Text = "O: 2,2"

PlayerTurn = False

LblPlayerTurn.Text = "X's turn"

Else ' X's go

GameBoard(2, 2) = 9

sender.Text = "X"

LblLastMove.Text = "X: 2,2"

PlayerTurn = True

LblPlayerTurn.Text = "O's turn"

End If

TurnNumber += 1

ModSavingFiles.AddNewTurnRecord(GameBoard, TurnNumber, Not (PlayerTurn))

ChangeColour(sender)

'CheckGrid()

CheckForVictory()

End If

End Sub

' << On Mouse Over >>

''' <summary>

''' Changes the contents of the button to {X} or {O} when the mouse goes over an empty button

''' </summary>

''' <param name="sender">The button that called the event</param>

Private Sub GridButtons\_MouseEnter(sender As Object, e As EventArgs) Handles Btn00.MouseEnter, Btn10.MouseEnter, Btn20.MouseEnter, Btn01.MouseEnter, \_

Btn11.MouseEnter, Btn21.MouseEnter, Btn02.MouseEnter, Btn12.MouseEnter, Btn22.MouseEnter

If GameInProgress = True Then

If sender.Text = "O" Or sender.Text = "X" Then

Else

If PlayerTurn = True Then

sender.Text = "{O}"

sender.ForeColor = Color.FromArgb(17, 150, 157)

Else

sender.Text = "{X}"

sender.ForeColor = Color.Salmon

End If

End If

End If

End Sub

''' <summary>

''' Emptys the {X} or {O} from the button that the mouse left

''' </summary>

''' <param name="sender">The button that called the event</param>

Private Sub GridButtons\_MouseLeave(sender As Object, e As EventArgs) Handles Btn00.MouseLeave, Btn10.MouseLeave, Btn20.MouseLeave, Btn01.MouseLeave, \_

Btn11.MouseLeave, Btn21.MouseLeave, Btn02.MouseLeave, Btn12.MouseLeave, Btn22.MouseLeave

If GameInProgress = True Then

If sender.Text = "O" Or sender.Text = "X" Then

Else

sender.Text = ""

sender.ForeColor = Color.Black

End If

End If

End Sub

' <=== ===>

''' <summary>

''' When a key is pressed...

''' </summary>

''' <param name="E">The details of the key event that called the function</param>

Private Sub ArrowKey\_Press(Sender As Object, E As KeyEventArgs) Handles Me.KeyDown

Select Case E.KeyCode

Case Keys.F1

FrmHelp.Show()

End Select

End Sub

Private Overloads Sub OnDragOver(Sender As Object, DragDetails As DragEventArgs) Handles Me.DragOver

If DragDetails.Data.GetDataPresent(DataFormats.FileDrop) Then

DragDetails.Effect = DragDropEffects.Copy

End If

End Sub

Private Overloads Sub OnDragDrop(Sender As Object, Details As DragEventArgs) Handles Me.DragDrop

FrmReplayViewer.Show()

FrmReplayViewer.OnDragDrop(Sender, Details)

Me.Close()

End Sub

Private Sub OnRightClick(Sender As Object, Details As MouseEventArgs) Handles Me.MouseClick

' Right click

If Details.Button = Windows.Forms.MouseButtons.Right Then

CntxtMenuPlayGame.AutoClose = True

CntxtMenuPlayGame.Show(Control.MousePosition)

End If

End Sub

End Class

Public Class FrmReplayViewer

Dim ReplayDetails As List(Of GameBoardState) = Nothing

Dim ActiveTurn As Short = 1

''' <summary>

''' Prepares for user input

''' </summary>

Private Sub FrmReplayViewer\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

ResizeWindow(False)

PnlGameButtons.Enabled = False

PnlGameBoard.BackColor = Color.FromKnownColor(KnownColor.Control)

TxtFilePath.Text = ""

End Sub

''' <summary>

''' Get gaming

''' </summary>

Private Sub NewGameToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles NewGameToolStripMenuItem.Click

FrmPlayTTT.Show()

Me.Close()

End Sub

''' <summary>

''' Allows for setting changing

''' </summary>

Private Sub SettingsToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles SettingsToolStripMenuItem.Click

FrmSettings.Show()

Me.Close()

End Sub

''' <summary>

''' Memed

''' </summary>

Private Sub GetHelpOnlineToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpOnlineToolStripMenuItem.Click

MsgBox("There is no one to help you online. Sorry", MsgBoxStyle.OkOnly, "HAHAHAHAHAHAHAHAHAHA")

End Sub

''' <summary>

''' Gives help

''' </summary>

Private Sub GetHelpToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpToolStripMenuItem.Click

FrmHelp.Show()

End Sub

''' <summary>

''' Loads a replay from a [x].T3Replay file

''' </summary>

''' <param name="sender">The object that called the sub</param>

''' <param name="e">The event that the sub was called with</param>

Private Sub LoadPreviousGame\_Click(sender As Object, e As EventArgs) Handles BtnLoadPreviousReplay.Click, LoadPreviousGameToolStripMenuItem.Click

Dim FilePath As String = ModOpeningFiles.OpenReplayFilePath()

If FilePath = "ERROR: No file chosen" Then

MsgBox(FilePath, MsgBoxStyle.OkOnly, "FAILURE")

FileChosen(False)

PnlGameButtons.Enabled = False

ElseIf FilePath = "ERROR: File version out of date" Then

MsgBox(FilePath, MsgBoxStyle.OkOnly, "FAILURE")

FileChosen(False)

PnlGameButtons.Enabled = False

Else

ReplayDetails = ModOpeningFiles.ParseTextFile(FilePath)

Dim Failed As Boolean = False

For Each Element In ReplayDetails

If Element.GameBoardLine = "ERROR" Then

MsgBox("ERROR: Corrupted file")

FileChosen(False)

Failed = True

End If

Next

If Failed = False Then

FileChosen(True)

PnlGameButtons.Enabled = True

BtnNextMove.Enabled = True

BtnToLastMove.Enabled = True

ActiveTurn = 1

EvaluateBoard()

End If

End If

BtnPrevMove.Enabled = False

BtnToFirstMove.Enabled = False

End Sub

''' <summary>

''' Prepares the game board for replays

''' </summary>

''' <param name="ReadyOrNot">Whether or not we are ready to look at a replay</param>

Public Sub FileChosen(ByVal ReadyOrNot As Boolean)

ResizeWindow(ReadyOrNot)

If ReadyOrNot = True Then

PnlGameButtons.Enabled = True

PnlGameBoard.BackColor = Color.FromKnownColor(KnownColor.GradientInactiveCaption)

TrBrMoves.Maximum = ReplayDetails.Count - 1

TrBrMoves.Value = 0

TrBrMoves.Refresh()

PnlGameBoard.Refresh()

DrawGridLines()

Else

ReplayDetails = Nothing

LblTotalTurns.Text = 0

TxtFilePath.Text = ""

PnlGameBoard.BackColor = Color.FromKnownColor(KnownColor.Control)

PnlGameButtons.Enabled = False

PnlGameBoard.Refresh()

End If

End Sub

''' <summary>

''' Draws the lines onto the blank grid

''' </summary>

Private Sub DrawGridLines()

Dim PanelGraphics As Graphics = PnlGameBoard.CreateGraphics()

Dim Pen As System.Drawing.Pen = Pens.Black

' Vericals

Dim XLine1 As Integer = 86

Dim XLine2 As Integer = 173

Dim XLine3 As Integer = 0

Dim XLine4 As Integer = 260

Dim Y1Verts As Integer = 0

Dim Y2Verts As Integer = 260

PanelGraphics.DrawLine(Pen, XLine1, Y1Verts, XLine1, Y2Verts)

PanelGraphics.DrawLine(Pen, XLine2, Y1Verts, XLine2, Y2Verts)

PanelGraphics.DrawLine(Pen, XLine3, Y1Verts, XLine3, Y2Verts)

PanelGraphics.DrawLine(Pen, XLine4, Y1Verts, XLine4, Y2Verts)

' Horizontals

Dim YLine1 As Integer = 86

Dim YLine2 As Integer = 173

Dim YLine3 As Integer = 0

Dim YLine4 As Integer = 260

Dim X1Horis As Integer = 0

Dim X2Horis As Integer = 260

PanelGraphics.DrawLine(Pen, X1Horis, YLine1, X2Horis, YLine1)

PanelGraphics.DrawLine(Pen, X1Horis, YLine2, X2Horis, YLine2)

PanelGraphics.DrawLine(Pen, X1Horis, YLine3, X2Horis, YLine3)

PanelGraphics.DrawLine(Pen, X1Horis, YLine4, X2Horis, YLine4)

PanelGraphics.Dispose()

End Sub

''' <summary>

''' Resizes the window cos prettiness

''' </summary>

''' <param name="ReadyOrNot">Whether or not a file has been successfully selected</param>

Private Sub ResizeWindow(ByVal ReadyOrNot As Boolean)

If ReadyOrNot = True Then

Me.MaximumSize = New System.Drawing.Size(300, 476)

Me.Size = Me.MaximumSize

Me.MinimumSize = Me.MaximumSize

Else

Me.MinimumSize = New System.Drawing.Size(300, 95)

Me.Size = Me.MinimumSize

Me.MaximumSize = Me.MinimumSize

End If

End Sub

' <=== ===>

''' <summary>

''' Changes the current turn being viewed

''' </summary>

Private Sub BtnToFirstMove\_Click(Optional sender As Object = Nothing, Optional e As EventArgs = Nothing) Handles BtnToFirstMove.Click

ActiveTurn = 1

BtnToFirstMove.Enabled = False

BtnPrevMove.Enabled = False

BtnNextMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End Sub

''' <summary>

''' Changes the current turn being viewed

''' </summary>

Private Sub BtnPrevMove\_Click(Optional sender As Object = Nothing, Optional e As EventArgs = Nothing) Handles BtnPrevMove.Click

ActiveTurn -= 1

If ActiveTurn = 1 Then

BtnToFirstMove.Enabled = False

BtnPrevMove.Enabled = False

End If

If ActiveTurn < ReplayDetails.Count Then

BtnNextMove.Enabled = True

BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End Sub

''' <summary>

''' Changes the current turn being viewed

''' </summary>

Private Sub BtnNextMove\_Click(Optional sender As Object = Nothing, Optional e As EventArgs = Nothing) Handles BtnNextMove.Click

ActiveTurn += 1

If ActiveTurn = ReplayDetails.Count Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

End If

If ActiveTurn > 1 Then

BtnToFirstMove.Enabled = True

BtnPrevMove.Enabled = True

End If

EvaluateBoard()

End Sub

''' <summary>

''' Changes the current turn being viewed

''' </summary>

Private Sub BtnToLastMove\_Click(Optional sender As Object = Nothing, Optional e As EventArgs = Nothing) Handles BtnToLastMove.Click

ActiveTurn = ReplayDetails.Count

BtnToLastMove.Enabled = False

BtnNextMove.Enabled = False

BtnToFirstMove.Enabled = True

BtnPrevMove.Enabled = True

EvaluateBoard()

End Sub

' <=== ===>

''' <summary>

''' Loop through the replaydetails array and evaluate which postion should go where

''' </summary>

Private Sub EvaluateBoard()

LblTurnNumber.Text = CStr(ActiveTurn)

TrBrMoves.Value = ActiveTurn - 1

PnlGameBoard.Refresh()

DrawGridLines()

For Looper = 0 To 8

Dim X As Integer

Dim Y As Integer

' <=======

Select Case Looper

Case 0

' 0, 0

X = 2

Y = 2

Case 1

' 0, 1

X = 2

Y = 88

Case 2

' 0, 2

X = 2

Y = 175

Case 3

' 1, 0

X = 88

Y = 2

Case 4

' 1, 1

X = 88

Y = 88

Case 5

' 1, 2

X = 88

Y = 175

Case 6

' 2, 0

X = 175

Y = 2

Case 7

' 2, 1

X = 175

Y = 88

Case Else

' 2, 2

X = 175

Y = 175

End Select

' <=======

LblTotalTurns.Text = ReplayDetails.Count

If ReplayDetails(ActiveTurn - 1).BoardArray(Looper) = "1" Then

DrawNought(X, Y)

ElseIf ReplayDetails(ActiveTurn - 1).BoardArray(Looper) = "9" Then

DrawCross(X, Y)

End If

Next

End Sub

''' <summary>

''' Draw a nought at the given coordinates

''' </summary>

''' <param name="X">The X Coordinate</param>

''' <param name="Y">The Y coordinate</param>

Private Sub DrawNought(ByVal X As Integer, ByVal Y As Integer)

Dim PanelGraphics As Graphics = PnlGameBoard.CreateGraphics()

Dim Pen As New System.Drawing.Pen(Brushes.DarkBlue)

Pen.Color = Color.DarkBlue

Pen.Width = 3

PanelGraphics.DrawEllipse(Pen, New Rectangle(X, Y, 83, 83))

PanelGraphics.Dispose()

End Sub

''' <summary>

''' Draw a cross at the given coordinates

''' </summary>

''' <param name="X">the X coordinate</param>

''' <param name="Y">The Y coordinate</param>

Private Sub DrawCross(ByVal X As Integer, ByVal Y As Integer)

Dim PanelGraphics As Graphics = PnlGameBoard.CreateGraphics()

Dim Pen As New System.Drawing.Pen(Brushes.Maroon)

Pen.Color = Color.Maroon

Pen.Width = 3

PanelGraphics.DrawLine(Pen, X, Y, (X + 82), (Y + 82))

PanelGraphics.DrawLine(Pen, (X + 82), Y, X, (Y + 82))

PanelGraphics.Dispose()

End Sub

''' <summary>

''' When keys are pressed, do stuff

''' </summary>

''' <param name="Sender">The object that sent the key references</param>

''' <param name="E">the details of the keypress</param>

Private Sub ArrowKey\_Press(Sender As Object, E As KeyEventArgs) Handles Me.KeyDown

Select Case E.KeyCode

Case Keys.F1

FrmHelp.Show()

Case Keys.F5

Try

Dim EmptyBoard As New GameBoardState With {.GameBoardLine = ""}

If ReplayDetails(3).BoardArray = EmptyBoard.BoardArray Then

Else

PnlGameBoard.Refresh()

EvaluateBoard()

End If

Catch ERR As Exception

' Do nothing

End Try

Case Keys.D

If BtnNextMove.Enabled = True Then

BtnNextMove\_Click()

End If

Case Keys.A

If BtnPrevMove.Enabled = True Then

BtnPrevMove\_Click()

End If

Case Keys.S

If BtnToFirstMove.Enabled = True Then

BtnToFirstMove\_Click()

End If

Case Keys.W

If BtnToLastMove.Enabled = True Then

BtnToLastMove\_Click()

End If

Case Keys.D1

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 1) Then

ActiveTurn = 1

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = False

BtnToFirstMove.Enabled = False

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case Keys.D2

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 2) Then

ActiveTurn = 2

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case Keys.D3

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 3) Then

ActiveTurn = 3

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case Keys.D4

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 4) Then

ActiveTurn = 4

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case Keys.D5

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 5) Then

ActiveTurn = 5

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 5 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case Keys.D6

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 6) Then

ActiveTurn = 6

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 6 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case Keys.D7

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 7) Then

ActiveTurn = 7

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 7 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case Keys.D8

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 8) Then

ActiveTurn = 8

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 8 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case Keys.D9

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 9) Then

ActiveTurn = 9

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 9 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

End Select

End Sub

''' <summary>

''' Change the mouse icon when files are dragged over so people know they can be released here

''' </summary>

''' <param name="Sender">The object that sent the data</param>

''' <param name="DragDetails">The information about the drag</param>

Private Overloads Sub OnDragOver(Sender As Object, DragDetails As DragEventArgs) Handles Me.DragEnter

If DragDetails.Data.GetDataPresent(DataFormats.FileDrop) Then

DragDetails.Effect = DragDropEffects.Copy

End If

End Sub

''' <summary>

''' When a file is dragged over the form

''' </summary>

''' <param name="Sender">The object that sent the data</param>

''' <param name="DragDetails">The deets of teh drag ting</param>

Public Overloads Sub OnDragDrop(Sender As Object, DragDetails As DragEventArgs) Handles Me.DragDrop

Dim FilePaths() As String = DragDetails.Data.GetData(DataFormats.FileDrop)

If FilePaths(0).Split(".")(FilePaths.Count) = "T3Replay" Then

ReplayDetails = ModOpeningFiles.ParseTextFile(FilePaths(0))

FileChosen(True)

Dim Failed As Boolean = False

For Each Element In ReplayDetails

If Element.GameBoardLine = "ERROR" Then

FileChosen(False)

MsgBox("ERROR: Corrupted file")

Failed = True

Exit Sub

End If

Next

If Failed = False Then

PnlGameButtons.Enabled = True

BtnNextMove.Enabled = True

BtnToLastMove.Enabled = True

ActiveTurn = 1

BtnPrevMove.Enabled = False

BtnToFirstMove.Enabled = False

TxtFilePath.Text = FilePaths(0)

EvaluateBoard()

End If

Else

MsgBox("Not Accepted: File type not supported. Please use a '\*.T3Replay' file")

End If

End Sub

''' <summary>

''' When a file is dragged onto the "T3.exe" file, prepare to replay it

''' </summary>

''' <param name="FilePath">The file path of the [first] file dragged onto the "T3.exe" file</param>

Public Sub OnStartWithFile(FilePath)

ReplayDetails = ModOpeningFiles.ParseTextFile(FilePath)

FileChosen(True)

For Each Element In ReplayDetails

If Element.GameBoardLine = "ERROR" Then

FileChosen(False)

MsgBox("ERROR: Corrupted file")

Exit Sub

End If

Next

PnlGameButtons.Enabled = True

BtnNextMove.Enabled = True

BtnToLastMove.Enabled = True

ActiveTurn = 1

BtnPrevMove.Enabled = False

BtnToFirstMove.Enabled = False

TxtFilePath.Text = FilePath

EvaluateBoard()

End Sub

Private Sub TrBrMoves\_Scroll(sender As Object, e As EventArgs) Handles TrBrMoves.Scroll

Select Case TrBrMoves.Value

Case 0

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 1) Then

ActiveTurn = 1

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = False

BtnToFirstMove.Enabled = False

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case 1

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 2) Then

ActiveTurn = 2

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case 2

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 3) Then

ActiveTurn = 3

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case 3

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 4) Then

ActiveTurn = 4

BtnNextMove.Enabled = True

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

BtnToLastMove.Enabled = True

EvaluateBoard()

End If

End If

Case 4

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 5) Then

ActiveTurn = 5

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 5 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case 5

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 6) Then

ActiveTurn = 6

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 6 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case 6

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 7) Then

ActiveTurn = 7

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 7 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case 7

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 8) Then

ActiveTurn = 8

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 8 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

Case 8

If (ReplayDetails IsNot Nothing) Then

If (ReplayDetails.Count >= 9) Then

ActiveTurn = 9

BtnPrevMove.Enabled = True

BtnToFirstMove.Enabled = True

If ReplayDetails.Count = 9 Then

BtnNextMove.Enabled = False

BtnToLastMove.Enabled = False

Else : BtnNextMove.Enabled = True : BtnToLastMove.Enabled = True

End If

EvaluateBoard()

End If

End If

End Select

End Sub

End Class

Public Class FrmHelp

''' <summary>

''' Display the help in a textbox

''' </summary>

Private Sub BtnGamePlayHelp\_Click(sender As Object, e As EventArgs) Handles BtnGamePlayHelp.Click

RtbHelp.Refresh()

RtbHelp.Text = "How to play." & Environment.NewLine & \_

"Beginning a game" & Environment.NewLine & \_

"-> 1) Click the 'Start Game' button or 'New Game' in the 'File' menu" & Environment.NewLine & \_

"--[ If a game is already in progress you will be prompted to continue or abandon it. If a game is abandoned, the replay is not saved" & Environment.NewLine & \_

"-> 2) If you have not done so previously, enter the names of the players" & Environment.NewLine & Environment.NewLine & \_

"Playing a game" & Environment.NewLine & \_

"-> 1) O's go first, so that player should click on the tile they wish to play" & Environment.NewLine & \_

"-> 2) X's get the second turn, so they should click on any unoccupied square they wish to move to" & Environment.NewLine & \_

"-> 3) Take turns to play tiles until either a player gets 3 of their icons in a row / column / diagonal (a Win) or no tiles are left to play (a Draw)" & Environment.NewLine & \_

"--[ If a tile has already been occupied, you will be unable to select it for your move" & Environment.NewLine & Environment.NewLine & \_

"Ending a game" & Environment.NewLine & \_

"-> 1) Click the 'End Game' button. You will be asked whether or not you wish to end the game" & Environment.NewLine & \_

"-> 2) Click 'Yes' to end the game. Clicking 'No' will return you to the game" & Environment.NewLine & \_

"--[ If a game is abandoned, the replay is not saved" & Environment.NewLine & Environment.NewLine & \_

"Changing player names" & Environment.NewLine & \_

"-> 1) Click 'Change Names'" & Environment.NewLine & \_

"-> 2) Enter Player 1's chosen name" & Environment.NewLine & \_

"-> 3) Enter Player 2's chosen name" & Environment.NewLine & \_

"--[ Clicking 'Cancel' will set the name to 'Player [x]'" & Environment.NewLine & \_

"--[ Names must be between 1 and 10 characters in length" & Environment.NewLine & Environment.NewLine & \_

"Instant replays" & Environment.NewLine & \_

"-> Drag and drop a '\*.T3Replay' file into the window to load a replay instantly"

End Sub

''' <summary>

''' Moar Halp

''' </summary>

Private Sub BtnReplayHelp\_Click(sender As Object, e As EventArgs) Handles BtnReplayHelp.Click

RtbHelp.Refresh()

RtbHelp.Text = "Replaying previous games." & Environment.NewLine & \_

"1) Click 'Select a file' or drag and drop a '\*.T3Replay' file into the window" & Environment.NewLine & \_

"2) Navigate to the '\*.T3Replay' file and open it" & Environment.NewLine & \_

"--[ This step will not be necessary if you have used the drag and drop feature" & Environment.NewLine & \_

"3) Choose any of the following" & Environment.NewLine & \_

"-> '1ST': Jump to the first move" & Environment.NewLine & \_

"--[ Disabled on load" & Environment.NewLine & \_

"-> 'PREV': Step back one move" & Environment.NewLine & \_

"--[ Disabled on load" & Environment.NewLine & \_

"-> 'NXT': Step forward one move" & Environment.NewLine & \_

"-> 'LAST': Jump to the final move" & Environment.NewLine & \_

"-> Scroll Bar: Scroll to the desired move" & Environment.NewLine & \_

"-> Number Keys: Select the desired move to view" & Environment.NewLine & \_

"-> WASD Keys: Scroll to the desired move" & Environment.NewLine & \_

"-[ The number in the middle of the buttons shows you what turn you are currently viewing" & Environment.NewLine & \_

"[ If you want to open another '\*.T3Replay' file, simply click 'Select a file' or Drag and Drop a '\*.T3Reply' file again" & Environment.NewLine & \_

"[ Alternatively, should you wish to be awesome, drag and drop a '\*.T3Replay' file over the T3.exe file itself!!!"

End Sub

''' <summary>

''' Yup. Still dishing out help

''' </summary>

Private Sub BtnSettingsHelp\_Click(sender As Object, e As EventArgs) Handles BtnSettingsHelp.Click

RtbHelp.Refresh()

RtbHelp.Text = "Changing settings." & Environment.NewLine & \_

"Available Settings" & Environment.NewLine & \_

"-> Autosave replays" & Environment.NewLine & \_

"--> Whether or not the application should ask to save replays" & Environment.NewLine & \_

"-> Noughts go first" & Environment.NewLine & \_

"--> Change whether or not Noughts get the first move" & Environment.NewLine & \_

"---[Soon to be 'Random player goes first'" & Environment.NewLine & \_

"-> Save folder" & Environment.NewLine & \_

"--> Change" & Environment.NewLine & \_

"---> Select a folder to save replay files to" & Environment.NewLine & \_

"--> Reset" & Environment.NewLine & \_

"---> Resets the replay folder to the file that the application started in" & Environment.NewLine & Environment.NewLine & \_

"Future settings" & Environment.NewLine & \_

"-> Themes - Change the appearance of the game" & Environment.NewLine & \_

"-> Random player goes first - Game picks a random player to begin playing"

End Sub

End Class

Public Class FrmSettings

Dim ChangeSettingsAllowed As Boolean = False

''' <summary>

''' Sets the checkboxes to the correct mode depending on the current settings in My.Settings

''' </summary>

''' <param name="sender">The object that called the sub</param>

''' <param name="e">The arguments with which the sub was called</param>

Private Sub FrmSettings\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

TxtSaveFolder.Text = My.Settings.SaveFolder

If My.Settings.AutomaticallySaveReplays = True Then

CbxAutosave.Checked = True

Else

CbxAutosave.Checked = False

End If

'Debug.WriteLine(CStr(My.Settings.AutomaticallySaveReplays))

If My.Settings.CrossesGoFirst = True Then

CbxNoughtsFirst.Checked = True

Else

CbxNoughtsFirst.Checked = False

End If

'Debug.WriteLine(CStr(My.Settings.CrossesGoFirst))

ChangeSettingsAllowed = True

End Sub

Private Sub NewGameToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles NewGameToolStripMenuItem.Click

FrmPlayTTT.Show()

Me.Close()

End Sub

Private Sub GetHelpOnlineToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpOnlineToolStripMenuItem.Click

MsgBox("There is no one to help you online. Sorry", MsgBoxStyle.OkOnly, "HAHAHAHAHAHAHAHAHAHA")

End Sub

Private Sub GetHelpToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles GetHelpToolStripMenuItem.Click

FrmHelp.Show()

End Sub

Private Sub LoadPreviousGameToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles LoadPreviousGameToolStripMenuItem.Click

FrmReplayViewer.Show()

Me.Close()

End Sub

''' <summary>

''' Change the My.Settings.AutomaticallySaveReplays depending on the new state of CbxAutosave

''' </summary>

''' <param name="sender">The object that called the sub</param>

''' <param name="e">The event arguments when the sub was called</param>

Private Sub CbxAutosave\_CheckedChanged(sender As Object, e As EventArgs) Handles CbxAutosave.CheckedChanged

If ChangeSettingsAllowed Then

If CbxAutosave.Checked Then

My.Settings.AutomaticallySaveReplays = True

Else

My.Settings.AutomaticallySaveReplays = False

End If

'Debug.WriteLine(CStr(My.Settings.AutomaticallySaveReplays))

End If

End Sub

''' <summary>

''' Change the My.Settings.CrossesGoFirst depending on the new state of CbxNoughtsFirst

''' </summary>

''' <param name="sender">The object that called the sub</param>

''' <param name="e">The event arguments when the sub was called</param>

Private Sub CbxNoughtsFirst\_CheckedChanged(sender As Object, e As EventArgs) Handles CbxNoughtsFirst.CheckedChanged

If ChangeSettingsAllowed = True Then

If CbxNoughtsFirst.Checked Then

My.Settings.CrossesGoFirst = True

Else

My.Settings.CrossesGoFirst = False

End If

'Debug.WriteLine(CStr(My.Settings.CrossesGoFirst))

End If

End Sub

''' <summary>

''' Allows the user to select a new folder to save replays to

''' </summary>

Private Sub BtnChangeSaveFolder\_Click(sender As Object, e As EventArgs) Handles BtnChangeSaveFolder.Click

TxtSaveFolder.Text = ModUtilities.ChangeSaveFolder()

End Sub

''' <summary>

''' Sets the save path to the folder containing the application

''' </summary>

Private Sub BtnResetDefaultFilepath\_Click(sender As Object, e As EventArgs) Handles BtnResetDefaultFilepath.Click

My.Settings.SaveFolder = Application.StartupPath

MsgBox("Reset save path to " & My.Settings.SaveFolder, MsgBoxStyle.OkOnly, "SUCCESS")

End Sub

End Class

''' <summary>

''' A nice easy way of accessing the gamedata for each turn

''' </summary>

Public Class GameBoardState

Implements IEquatable(Of GameBoardState)

' <== A line in the .T3Replay file ==>

Private GameBoardContent As String = ""

''' <summary>

''' Allows for the setting and getting of the gameboard value

''' </summary>

''' <value>POTATOES</value>

''' <returns>The raw game board state</returns>

Public Property GameBoardLine As String

Get

If GameBoardContent = "" Then

Return "EMPTY"

Else

Return GameBoardContent

End If

End Get

Set(value As String)

If Len(value) <> 16 And Len(value) <> 17 Then GameBoardContent = "ERROR" : Exit Property

If Len(value.Split("|")(0)) <> 9 Then GameBoardContent = "ERROR" : Exit Property

If Len(value.Split("|")(1)) <> 1 Then GameBoardContent = "ERROR" : Exit Property

Dim Failed As Boolean = True

For Looper = 1 To 9

If value.Split("|")(1) = CStr(Looper) Then Failed = False

Next

If Failed = True Then GameBoardContent = "ERROR" : Exit Property

If value.Split("|")(2) <> "True" And value.Split("|")(2) <> "False" Then GameBoardContent = "ERROR" : Exit Property

GameBoardContent = value

End Set

End Property

' <== ==>

' <== The contents of the board after the turn ==>

''' <summary>

''' Finds and returns the board after the turn

''' </summary>

''' <value>The contents of the board in form "111999000"</value>

''' <returns>The contents of the board in form "111999000"</returns>

Public ReadOnly Property BoardArray As String

Get

If GameBoardContent = "ERROR" Then

Return "ERROR"

End If

Dim TempGameBoard As String = GameBoardContent

Dim FoundBoard As String() = TempGameBoard.Split("|")

Dim PenultiBoard As String = FoundBoard(0)

Dim FinalBoard As String = ""

For X = 0 To (Len(PenultiBoard) - 1) Step 1

FinalBoard &= CStr(PenultiBoard(X))

Next

Return FinalBoard

End Get

End Property

' <== ==>

' <== ==>

''' <summary>

''' Gets the turn number for the latest move

''' </summary>

''' <value>Turn number as integer [logical minimum: 5, logical maximum: 9]</value>

''' <returns>Turn number as integer</returns>

Public ReadOnly Property TurnNumber As Integer

Get

If GameBoardContent = "ERROR" Then

Return "ERROR"

End If

Dim TempGameBoard As String = GameBoardContent

Dim FinalBoard As String() = TempGameBoard.Split("|")

Return Val(FinalBoard(1))

End Get

End Property

' <== ==>

''' <summary>

''' Finds the player who made the move

''' </summary>

''' <value>The player who's turn it was</value>

''' <returns>The player who made the move this turn</returns>

Public ReadOnly Property PlayerTurn As Boolean

Get

If GameBoardContent = "ERROR" Then

Return "ERROR"

End If

Dim TempGameBoard As String = GameBoardContent

Dim FinalBoard As String() = TempGameBoard.Split("|")

If FinalBoard(2) = "True" Then

Return True

Else : Return False

End If

End Get

End Property

' <== ==>

''' <summary>

''' Compares the primary content of two GameBoardState objects in a GameBoardState [= / !=] GameBoardState logic situation

''' </summary>

''' <param name="other">Another GameBoardState object</param>

''' <returns>Whether the two game board states are equal</returns>

Public Overloads Function Equals(other As TicTacToe.GameBoardState) As Boolean Implements IEquatable(Of GameBoardState).Equals

If other Is Nothing Then

Return False

End If

Return (Me.GameBoardContent.Equals(other.GameBoardContent))

End Function

' <== ==>

End Class

Module ModOpeningFiles

Friend LoadedReplayAlready As Boolean = False

' <=== Loading a replay ===>

''' <summary>

''' Allows the user to select a replay file

''' </summary>

''' <returns>The file path or an error code</returns>

Public Function OpenReplayFilePath()

Dim FileDialogue As OpenFileDialog = New OpenFileDialog()

Dim strFileName As String

FileDialogue.Title = "Open Replay"

FileDialogue.InitialDirectory = My.Settings.SaveFolder

FileDialogue.Filter = "Replay files (\*.T3Replay)|\*.T3Replay"

FileDialogue.FilterIndex = 1

FileDialogue.RestoreDirectory = False

FileDialogue.CheckPathExists = True

FileDialogue.CheckFileExists = True

FileDialogue.Multiselect = False

If FileDialogue.ShowDialog() = DialogResult.OK Then

strFileName = FileDialogue.FileName

FrmReplayViewer.TxtFilePath.Text = strFileName

' Check if file is right version

Dim StreamReader As New IO.StreamReader(strFileName)

If StreamReader.ReadLine() = "T3Replay - " & Application.ProductVersion Then

StreamReader.Close()

Return strFileName

Else

StreamReader.Close()

FrmReplayViewer.TxtFilePath.Text = ""

Return "ERROR: File version out of date"

End If

End If

FrmReplayViewer.TxtFilePath.Text = ""

Return "ERROR: No file chosen"

End Function

''' <summary>

''' Runs through a .T3Replay file and appends the gameboardstate for each turn to the list

''' </summary>

''' <param name="FilePath">The file path of the .T3Replay file</param>

Public Function ParseTextFile(FilePath As String)

Dim ReplayDetails As New List(Of GameBoardState)()

Dim StreamReader As New IO.StreamReader(FilePath)

Dim FirstLineDataBecauseINeededToAdvanceTheReaderByOneLineForReasons As String = StreamReader.ReadLine()

While StreamReader.Peek() <> -1

Dim Line As New GameBoardState() With {.GameBoardLine = StreamReader.ReadLine()}

ReplayDetails.Add(Line)

End While

StreamReader.Close()

Return ReplayDetails

End Function

End Module

Imports System.IO

Imports System.Text

Module ModSavingFiles

' <=== Saving a game ===>

Dim SaveFilePath As String = Application.StartupPath & "\"

Dim ToWriteToFile As String = "T3Replay - " & Application.ProductVersion

''' <summary>

''' Emptys the string to write to the file and resets the save file path

''' </summary>

Public Sub ResetFileSettings()

SaveFilePath = My.Settings.SaveFolder & "\"

ToWriteToFile = "T3Replay - " & Application.ProductVersion

End Sub

''' <summary>

''' Generates a new save file name to prevent overwriting previous saves

''' </summary>

Private Sub NewSaveFilePath()

Dim Flag As Boolean = False

Dim FileNumber As Integer = 0

SaveFilePath = My.Settings.SaveFolder & "\"

While Flag = False

If File.Exists(SaveFilePath & CStr(FileNumber) & ".T3Replay") Then

FileNumber += 1

Else

Flag = True

SaveFilePath &= CStr(FileNumber) & ".T3Replay"

End If

End While

'Debug.WriteLine(SaveFilePath)

End Sub

''' <summary>

''' Adds the details of the move made to the turn record

''' </summary>

''' <param name="GameBoard">The current image of the game board</param>

''' <param name="TurnNumber">The turn number</param>

''' <param name="PlayerTurn">Whose turn it is (True = O, False = X)</param>

Public Sub AddNewTurnRecord(ByVal GameBoard As Short(,), TurnNumber As Integer, PlayerTurn As Boolean)

Dim Appendment As String = Environment.NewLine

For Looper = 0 To 2

For Looper2 = 0 To 2

Appendment &= CStr(GameBoard(Looper, Looper2))

Next

Next

Appendment &= "|" & CStr(TurnNumber) & "|" & CStr(PlayerTurn)

'Debug.WriteLine(Appendment)

ToWriteToFile &= Appendment

End Sub

''' <summary>

''' Write to the chosen file

''' </summary>

''' <returns>Whether or not the save was successful</returns>

Private Function SaveNewReplay()

Try

If File.Exists(SaveFilePath) = False Then ' If the file exists, don't write to it

Dim StreamWriter As New IO.StreamWriter(SaveFilePath, False)

StreamWriter.Write(CStr(ToWriteToFile))

StreamWriter.Close()

' Open the file to read from - debugging

'Dim StreamReader As New IO.StreamReader(SaveFilePath)

'Dim readText As String = StreamReader.ReadToEnd()

'StreamReader.Close()

'Debug.WriteLine(readText)

' end debugging

Return "Success"

Else

Return "Failure: File (" & SaveFilePath & ") already exists"

End If

Catch Err As Exception

Return "Failure: " & CStr(Err.Message)

End Try

End Function

''' <summary>

''' Concatenates all data into a file and saves it

''' </summary>

Public Sub CreateNewReplaySave()

If My.Settings.AutomaticallySaveReplays = True Then

NewSaveFilePath() ' Generate a new savefile path

Dim SaveStatus As String = SaveNewReplay() ' Write all data to new savefile and return what happened

MsgBox(SaveStatus, MsgBoxStyle.OkOnly, "Save status...")

ResetFileSettings() ' Empty the variables

Else

Dim Choice As Integer = MsgBox("Would you like to save the replay", MsgBoxStyle.YesNo, "Save your replay?")

If Choice = 6 Then

NewSaveFilePath() ' Generate a new savefile path

Dim SaveStatus As String = SaveNewReplay() ' Write all data to new savefile and return what happened

MsgBox(SaveStatus, MsgBoxStyle.OkOnly, "Save status...")

ResetFileSettings() ' Empty the variables

Else

MsgBox("Save not created", MsgBoxStyle.OkOnly)

End If

End If

End Sub

End Module

Module ModUtilities

''' <summary>

''' Changes where the replays are saved to

''' </summary>

''' <returns>The filepath of the new folder</returns>

Public Function ChangeSaveFolder()

Dim FolderBrowser As FolderBrowserDialog = New FolderBrowserDialog

FolderBrowser.Description = "Select the folder to save to"

FolderBrowser.ShowNewFolderButton = True

If FolderBrowser.ShowDialog() = DialogResult.OK Then

My.Settings.SaveFolder = FolderBrowser.SelectedPath

Return FolderBrowser.SelectedPath

End If

MsgBox("No folder selected, save path unchanged")

Return My.Settings.SaveFolder

End Function

End Module