Concept Document - Winner’s TicTacToe

The product is a playable version of an adapted form of Tic-Tac-Toe, called “Winner’s Tic-Tac-Toe”. The rules and structure are similar to the normal game, with the adjusted rules of players being able to “steal” squares, making it so that there is always a winner, removing the typical issue of the regular game often ending in a tie. It has a clear interface with an interactable board. It will be designed to be played between two people, not against an ai. The purpose of the game is to be a digital, easy way to casually play a fun version of Tic-Tac-Toe with friends and family. The targeted audience includes pretty much anyone, as the rules are simple enough for most ages to understand.

Requirements - Winner’s TicTacToe

1)Req. #1. The game will have a title of the game, labeled “Winner’s Tic Tac Toe”.

2)Req. #2.1. The game will have displays for both “Player 1” and “Player 2”

3)Req. #2.2. Each display will indicate which symbol, “X” or “O”, corresponds with each player.

4)Req. #3.1. The game will have a switch labeled “Set Player Turn”, alternating the starting symbol.

5)Req. #3.2. The game’s “Set Player Turn” switch will disappear when a game is started.

6)Req. #4. The game will have a “Clear” button that removes all symbols from the board spaces.

7)Req. #5. The game will have an “Exit” button that closes the game and VI.

8)Req. #6. The game will have a “Turn” string display that shows the symbol of the current player’s turn.

9)Req. #7. The game will have a “Start Game” button.

10)Req. #7.1. The “Start Game” button will have a red border.

11)Req. #7.2. The “Start Game” button will have the board spaces become enabled and un-grayed out.

12)Req. #7.3 When the “Start Game” button is pressed, it will turn into the “End Game” button.

13)Req. #8. The game will have an “End Game” button.

14)Req. #8.1. The “End Game” button will have a green border.

15)Req. #8.2. The “End Game” button will have the “Set Player Turn” switch reappear.

16)Req. #8.3. The “End Game” button will have the board spaces become disabled and grayed out when pressed.

17)Req. #8.4. When the “End Game” button is pressed, it will turn into the “Start Game” button.

18)Req. #9. The game will have nine buttons in a 3x3 grid pattern to be the game board.

19)Req. #9.1. When a game button is pressed, it will display the symbol of the player with the active turn.

20)Req. #9.2. The active symbol will alternate between “X” and “O”.

21)Req. #9.3. The buttons will be able to be overridden if a player “steals” a game button by pressing it on their turn.

22)Req. #10. The game will have an “Instructions” display, giving a description of each part of the game interface (Set Player Turn, Start Game, Turn, End Game, Clear, Exit).

23)Req. #11. The game will have a “Rules” display, giving a description of the basic rules of the game.

24)Req. #11.1. The “Rules” display will explain that pressing a box will display the player’s symbol.

25)Req. #11.2. The “Rules” display will explain that a player can “steal” a box by putting their symbol on the other player’s symbol.

26)Req. #11.3. The “Rules” display will explain how the game is won, which is getting three of one player’s symbols in a row.

27)Req. #11.4. The “Rules” display will explain that there is always a winner.

28)Req. #12. The game will have a “Game Over” display.

29)Req. #12.1. The “Game Over” display will let the players know the game is over as well as which player won the game.

30)Req. #12.2. The “Game Over” display will appear when a win case is detected on the board, being three of one symbol in a row.

Design Document - Winner’s TicTacToe

Req. #1-2.2. will be accomplished by placing a string control on the front panel and typing the text needed .

Req. #3.1 will be accomplished by placing a slide switch on the front panel labeled “Set Player Turn”.

Req. #3.2 & 8.2 will be accomplished by creating a property node that will change the button to visible when the ‘START/END” button is pressed changing the case statement to true, allowing the property node to change the switch to invisible. When the case statement is false the property node will be changed to false making the switch visible again.

Req. #4. Will be accomplished by placing an ok button named “clear” that when pressed, will change a case structure to true, this case will initialize all the buttons to default and will assign all button texts to an empty string constant.

Req. #5 will be accomplished by placing an ok button that when pressed, will activate a display message to the user confirming if they would like to exit. When the user says yes it will activate a switch case that when true, will utilize a VI server reference to reference to the current VI then use an invoke node to close the program with FP.Close.

Req. #6 will be accomplished by connecting the set player turn button to a switch case, when true a string constant “X” will be leading into a string indicator that is connected with the text in the front panel.

Req. #7 will be accomplished by placing an ok button labeled “Start Game”

Req. #7.1 & 8.1 will be accomplished by going to the buttons properties option and selecting the appearance tab, there we will set a red colored border to off and green to on.

Req. # 7.2 & 8.3 will be accomplished by placing an ok button leading to a case statement, when false all buttons will be set to disabled and greyed out by sending the enum constant value to the property node. When true, the same will happen but instead of disabled and greyed out we change the setting to enabled.

Req. # 7.4, 8, 8.4 will be accomplished by going to the properties of the button and in the appearance tab, changing the on text to “END GAME” and off text to “START GAME”.

Req. # 9 will be accomplished by creating 9 ok buttons, changing their appearance in the properties setting, and aligning them into a 3x3 grid.

Req. # 9.1 will be accomplished by connecting each button to a switch case that, when true, will activate an indicator property node that will assign the button the symbol of the player who placed it. To know whose turn it is, the property node is connected to the case statement that is keeping track of the set player turn button.

Req. #9.2 will be accomplished by having a control property node in each of the buttons case statements that will go through a not gate and into an indicator property node that will change the state of the set player turn switch, effectively changing the current status between each player.

Req. #9.3 will be accomplished by setting the button to still be enabled even when a player has pressed the button, when the next player clicks the button when it is already occupied, the button will act as if it hasn’t been pressed and will change the text in the button to the other player's symbol.

Req. #10-11.4. will be accomplished by placing a string control on the front panel and typing the text needed.

Req. #12 will be accomplished by playing a string control on the front panel and typing the text needed, it will only be seen while the game has been won, by sending a boolean true to the visibility property node in a true case statement it will appear.

Req. #12.1 will be accomplished by sending each buttons current symbol through a subVI that will check to see if either player's symbols have 3 in a row in any position, if this is true this will send a signal to the case statement that will change the game over display text to either “player 1 wins” or “player 2 wins”.

Req. #12.2 will be accomplished by sending the players symbol into a case statement, this case statement will be turned to true only if a specific win case buttons all have some value in them, these values are all checked inside the case to see if they all equal each other, if they do then the signal will be passed on through a series of or comparisons that will let through a value of 1 when any win case signal is detected. This will be sent to a button that will turn on only when player 2 has won. The signals from all the win case statements will be collated into a binary string and collated into a number, this will be sent to a case statement that will decide which case wins, which is then compared with the winning players symbol. The signal is sent to a case statement depending on which player won and will display either “player 1 wins” or “player 2 wins”.

The Tic Tac Toe game will be a 3x3 grid of buttons that will display the players symbol when pressed. The player's symbol will be chosen by a switch that will either assign them a X or an O. The objective is for a player to get a 3-in-a-row with their respective symbol. The game will then display that a certain player has won. The players will have instructions telling them how to play the game. There will be a clear button that will clear the board of all symbols. It will have an exit button that will close the game. It will have a start game button that will allow players to make moves on the grid. It will have an end game button to stop the game.

Materials needed: Computer, LabVIEW 2018, Windows OS, 2 Persons, WinnersTicTacToe\_Final.vi, TicTacTie\_Checker.vi.

Requirements: 3 programmers knowledgeable in LabVIEW, schedule of delivery dates.

Concept Doc - September 11th

Requirements Doc - September 25th

Design Doc - October 23th

Functional Blk Dgrm - October 9th

Module Design - October 20th

GUI Design - October 10th

Implementation Log - November 20th

GUI Operational - October 14th

Functional Prototype - November 6th

Prototype Review - November 8th

Test Plan - October 9th

Test Results - November 15th

Areas of risk:

The areas of risk are not being able to identify a user input by not being able to use event structures.

Implementation Log:

-Added game title

-Added Set Player Turn switch and created 3x3 grid of buttons

-Added Exit and Start game buttons

-Added switch cases connected to the buttons

-Implemented game exit

-Added set player turn cases

-Implemented 3x3 grid to change to the players symbol

-Created alternating player algorithm

-Requires mechanical action to be changed to switch until released which causes previously placed symbols to change to the current symbol.

-Alternating symbol algorithm removed, button case statements were modified to with property nodes to “not” the current state of the set player turn switch

-Implemented buttons disabled when end game is pressed

-Implemented buttons enabled when end game is pressed

-Implemented win case statements

-Added player 1 or 2 win check

-Added collate win cases

-Added win case states if statement

-Added X and O win checker

-Added custom icons

-Added Rules and Instructions

Test Plan - Winner’s TicTacToe

Req. #1-2.2. We will test these requirements by making sure they appear appropriately on the front panel. We will also close and reopen the program to make sure the text is saved.

Req. #3.1 We will test this requirement by making sure the symbol displayed on the “Turn” display alternates when the switch is alternated.

Req. #3.2 We will test this requirement by selecting “START GAME” and seeing if the switch will remain visible or not.

Req. #4. We will test this requirement by making sure all of the strings displayed on the buttons turn blank when the button is pressed. We will also make sure to see if the buttons revert back to an “off” default state.

Req. #5 We will test this requirement by making sure the program goes through the proper steps for exiting the VI when the button is pressed. We should get an “Are you sure?” related tab, that when verified closes the program.

Req. #6 We will test this requirement by making sure the “Turn” display shows either an “X” or “O”. We will test to see if the symbol placed on a button matches the symbol that was shown on the “Turn” display.

Req. #7 We will test this by making sure the “START GAME” button appears accordingly on the front panel, with the correct label.

Req. #7.1 The front panel should display the start game button with a red border

Req. # 7.2 We will test this requirement by making sure that the buttons are enabled when the “START GAME” button is pressed, allowing them to be interactable. We will test every single button, also making sure that none of them are grayed out after the “START GAME” button is pressed.

Req. # 7.3 & 8 We will test this requirement by selecting the start game button to see if it will change into “END GAME” when pressed.

Req. #8.1 The front panel should display the start game button with a green border

Req# 8.2 We will test this requirement by selecting “end game” and seeing if the switch will become visible or not.

Req. # 8.3 We will test these requirements by making sure all of the buttons are disabled and grayed out after the “END GAME” button is pressed. We will test every individual button, making sure they aren’t interactable, as well as checking that every single button is grayed out.

Req. # 8.4 We will test this requirement by selecting the end game button to see if it will change into “start game” when pressed.

Req. # 9 We will test this requirement by making sure there are nine functioning buttons in a 3x3 grid pattern that appear accordingly on the front panel.

Req. # 9.1 We will test this requirement by making sure that when a button is pressed, it will show either an “X” or “O” depending on the current player’s turn. When the button is pressed the text label in the middle of the button should simply change from a blank or the opposite player symbol into the current player’s symbol. We will test this with every button, with both the symbols “X” and “O”.

Req. #9.2 We will test this requirement by making sure the symbols shown on the buttons alternate between “X” and “O” after every press. If a button is pressed and shows an “X”, the next button pressed should show an “O”, then followed by an “X”, etc. We will test this on every single button to make sure there are no exceptions.

Req. #9.3 We will test this requirement by making sure that a symbol label of each and every button can be overridden by the opposite symbol. We will test every single button, starting with an “O” then changing it directly to an “X”, then vice versa starting with an “X” and directly changing it to an “O”.

Req. #10-11.4. We will test these requirements by making sure they appear appropriately on the front panel. We will also close and reopen the program to make sure the text is saved.

Req. #12 We will test this requirement by making sure the “GAME OVER” display appears accordingly on the front panel.

Req. #12.1 We will test this requirement by testing if the display shows the correct string, either saying “PLAYER 1 WINS” or “PLAYER 2 WINS”, based on which player won the game. We will test different win cases with each symbol to see if the display shows the correct winner.

Req. #12.2 We will test this requirement by testing every possible win case, being the three horizontal cases, three vertical cases, and two diagonal cases, for each symbol, making sure the “GAME OVER” display appears when any of the cases are met.