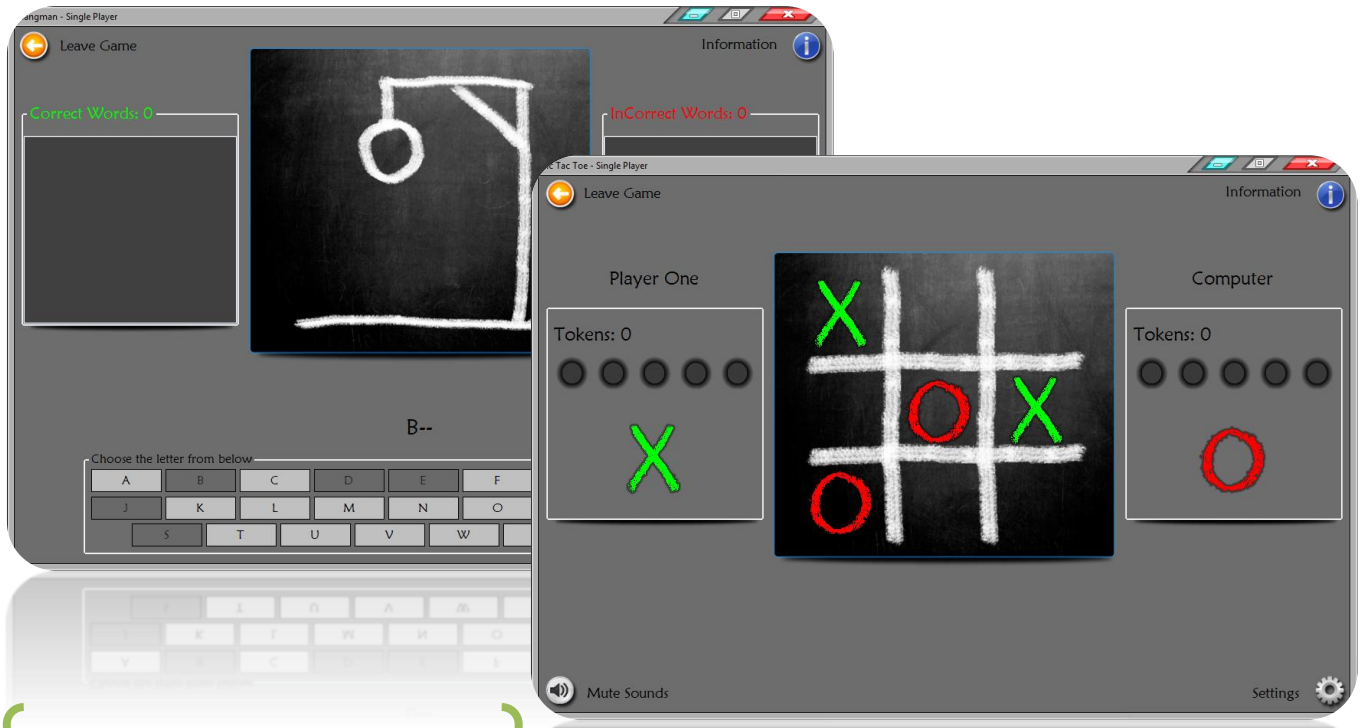
Major Project - Part 2: Terrathede Games™



Emmanuel Vaccaro

De La Salle College – Revesby Heights

Terrathede Games™

Table of Contents

1. Statement of Proposed Project:

The company, Terrathede Games™ and I have been assigned to develop an exclusive game package containing of Two Games: “Hangman” and “Tic Tac Toe” as well as our bonus VLC Media Player. Hangman will consist of three interchangeable game modes: “Single Player”, “Two Player” and “Versus”. Tic Tac Toe will consist of two interchangeable game modes: “Single Player” and “Two Player”.

1.1. Hangman

Hangman is a classic multi/single player game where player(s) try to guess a word (generated by the computer player or another player) letter by letter with a certain number of guesses.

1.1.1. Single Player:

In single player, if the word is guessed correctly, the word gets added to the “Correct Words” list. If the player gets the word incorrect, the word gets added to the “Incorrect Words” list. This game mode is endless, the player can see how many words he/she can guess in one game. The absence of score streaks in this game mode, relieves the user of rushing to beat the game and gives them the opportunity to enjoy the classic gameplay of Hangman.

1.1.2. Two Player:

In Two player, players enter words that other players have to guess in the usual way. Players switch and the player with the most amount of lives wins the round. The Player who won the round receives a gold token, the player who reaches 5 golden tokens wins the game.

1.1.3. Versus:

In Versus, it’s a race against the clock as players must guess the same word in 1 minute. The player who guesses the entire word wins the round and receives a token. The player who receives 5 golden tokens wins the game.

1.2. Tic Tac Toe:

With Tic Tac Toe, player(s) place their markers in a row of three. Once the player’s marker is set in a row of three, they are the winner of the round and receive one golden token. The player who receives 5 golden tokens wins the game.

1.2.1. Single Player:

In single player Tic Tac Toe, player one is up against the computer player who randomly selects a location on the grid to set their marker. The player with 5 tokens wins the game.

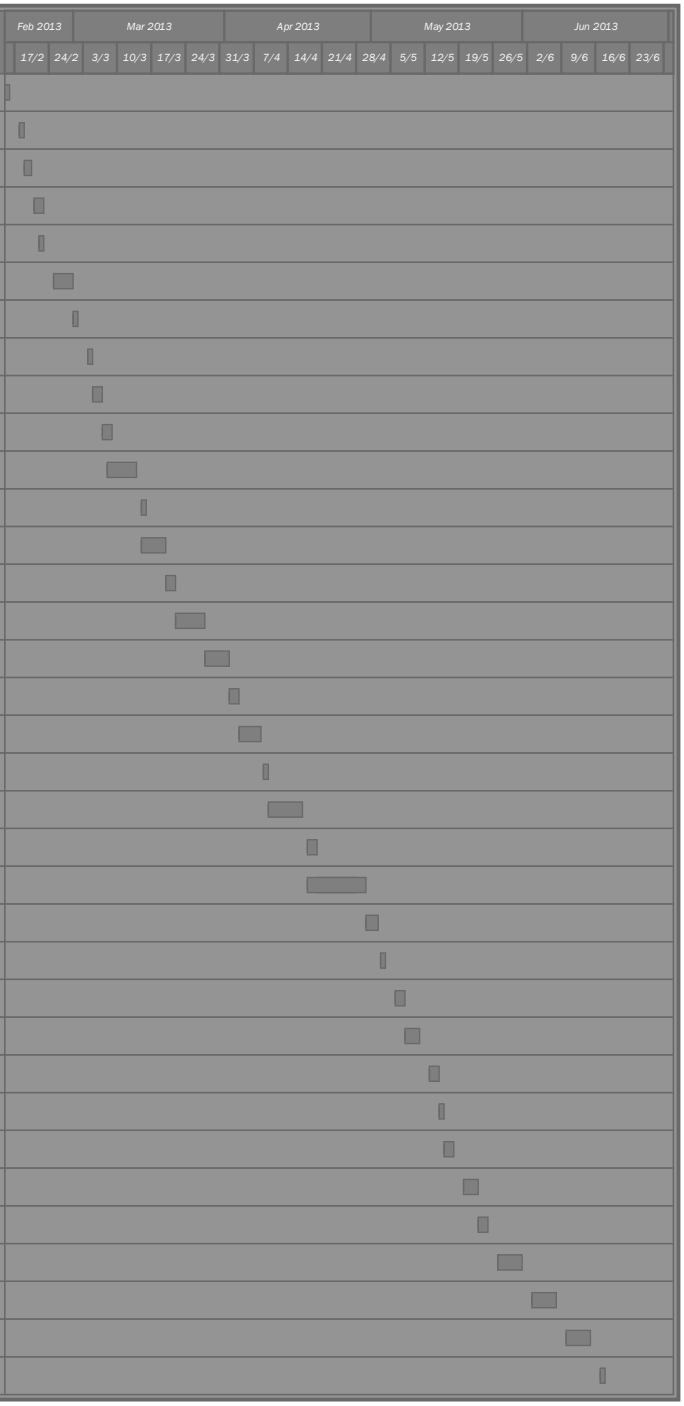
1.2.2. Two Player:

In two player Tic Tac Toe, players play against each other as they progressively place their markers in a position of their choosing. The game mode (like Single player) is interchangeable. Player’s also have access to the “Settings” menu to change music as they play the game, as opposed to hangman where it is more strategic and not timed.

2. Gantt Chart:

The following is a Gantt Chart clearly displaying specified tasks as well as their progress duration.
Note: The bonus “VLC Media Player” was created before the starting point of this Gantt Chart.

ID	Task	Start	Finish	Duration
1	Statement of proposed project	15/02/2013	15/02/2013	1d
2	Gantt Chart	18/02/2013	18/02/2013	1d
3	Screen Designs (Drawn on Paper)	19/02/2013	20/02/2013	1.5d
4	Collecting Images from Google	21/02/2013	22/02/2013	2d
5	Created Facebook Page	22/02/2013	22/02/2013	1d
6	Purchased Photoshop CS6 + Started “Photoshopping” Images	25/02/2013	28/02/2013	4d
7	VB: Main Menu	1/03/2013	1/03/2013	1d
8	VB: Hangman Mode Menu	4/03/2013	4/03/2013	1d
9	VB: Hangman Single Player	5/03/2013	6/03/2013	2d
10	VB: Hangman Two Player	7/03/2013	8/03/2013	2d
11	VB: Hangman Versus	8/03/2013	13/03/2013	4d
12	VB: Tic Tac Toe Mode Menu	15/03/2013	15/03/2013	1d
13	VB: Tic Tac Toe Single Player	15/03/2013	19/03/2013	3d
14	VB: Tic Tac Toe Two Player	20/03/2013	21/03/2013	2d
15	VB: Application Settings	22/03/2013	27/03/2013	4d
16	VB: Hangman Game Settings	28/03/2013	1/04/2013	3d
17	VB: Splash Screen	2/04/2013	3/04/2013	2d
18	VB: Mute/Unmute Button	4/04/2013	8/04/2013	2.5d
19	VB: Information Button	9/04/2013	9/04/2013	1d
20	VB: Added graphics to all forms	10/04/2013	16/04/2013	1w
21	Created Google Form & Attached to App	18/04/2013	19/04/2013	2d
22	Started Beta Testing (Recorded Results)	18/04/2013	29/04/2013	1w 3d
23	Went back and Fixed all errors recorded	30/04/2013	2/05/2013	2.5d
24	Storyboards	3/05/2013	3/05/2013	1d
25	Test Data	6/05/2013	7/05/2013	2d
26	Systems Hardware Test	8/05/2013	10/05/2013	3d
27	Deskchecks	13/05/2013	14/05/2013	2d
28	Evaluation Statement	15/05/2013	15/05/2013	1d
29	Ending Documentation	16/05/2013	17/05/2013	2d
30	Design for Blue ray Disk Cover & Label	20/05/2013	22/05/2013	3d
31	Quick Start Guide	23/05/2013	24/05/2013	2d
32	User Manual	27/05/2013	31/05/2013	1w
33	Printing all Documentation and covers	3/06/2013	7/06/2013	1w
34	Last minute program tests	10/06/2013	14/06/2013	1w
35	Hand in Assignment	17/06/2013	17/06/2013	1d



3. Storyboards:

Shown below is the navigation between screens. The screens shown below are linked by names and arrows. I.e. The splash screen is called “Splash Screen” and the arrow next to the property signifies the link proceeding to the next sub module/screen. Further information on the actions and changes within the screens can be found in the “Screen Design” section of this document (pg. 19).

3.1. Hangman:

The following storyboard is the routines in relevance to the Hangman game mode.

3.1.1. Single Player Hangman:

Splash Screen → Main Menu → Hangman Mode Menu → Hangman - Single Player → Gameplay (select letters) → Tokens Received → Leave Game → Hangman Mode Menu → Main Menu

3.1.2. Two Player Hangman:

Splash Screen → Main Menu → Hangman Mode Menu → Player One Input Box → Player Two Input Box → Hangman - Two Player → (Player Two “Enter word for player one” dialog box → Player One guess word (select letters) → Player One “Enter word for player two” dialog box” → Player Two Guess Word (select letters) → Tokens Received) → Game Over) → Leave Game → Hangman Mode Menu → Main Menu

‘The brackets signify what will be repeated depending on the player(s) selection.

3.1.3. Versus Hangman:

Splash Screen → Main Menu → Hangman Mode Menu → Player One Input Box → Player Two Input Box → Hangman - Versus → (Player One’s turn → Gameplay (select letters) → Player Two’s turn → Gameplay (select letters) → Game Over) → Leave Game → Hangman Mode Menu → Main Menu

‘The brackets signify what will be repeated depending on the player(s) selection.

3.2. Tic Tac Toe:

The following storyboard follows the typical user navigation sequence for the Tic Tac Toe game mode.

3.2.1. Single Player Tic Tac Toe:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe - Single Player → (Player One’s turn → Computer Player’s turn → Tokens Received) → Game Over → Leave Game → Tic Tac Toe Mode Menu → Main Menu

‘The brackets indicate the sequence that will be repeated depending on the player(s) selection.

3.2.2. Two Player Tic Tac Toe:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe - Two Player → (Player One’s turn → Player Two’s turn → Token’s Received) → Game Over → Leave Game → Tic Tac Toe Mode Menu → Main Menu

‘The brackets indicate the sequence that will be repeated depending on the player(s) selection.

3.3. Information Button:

The “Information” button is located on all forms of the game package. The following storyboard indicates the easiest way to access the information button on startup of the application as well as all the other possible ways of accessing the information button.

3.3.1. Main Menu:

Splash Screen → Main Menu → Information Button

3.3.2. Hangman Mode Menu:

Splash Screen → Main Menu → Hangman Mode Menu → Information Button

3.3.3. Hangman – Single Player:

Splash Screen → Main Menu → Hangman – Single Player → Information Button

3.3.4. Hangman – Two Player:

Splash Screen → Main Menu → Hangman – Two Player → Information Button

3.3.5. Hangman – Versus:

Splash Screen → Main Menu → Hangman – Versus → Information Button

3.3.6. Tic Tac Toe Mode Menu:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Information Button

3.3.7. Tic Tac Toe – Single Player:

Splash Screen → Main Menu → Tic Tac Toe – Single Player → Information Button

3.3.8. Tic Tac Toe – Two Player:

Splash Screen → Main Menu → Tic Tac Toe – Two Player → Information Button

3.4. Settings Button:

The “Settings” button is located on certain forms of the game package. It is not located on any “Hangman” modes and neither the mode selection. This is to eliminate the need of changing/playing music in the settings menu to make it a more strategic and quiet gameplay. The following storyboard indicates the various possibly ways of navigating to the settings menu.

3.4.1. Main Menu:

Splash Screen → Main Menu → Settings Button

3.4.2. Tic Tac Toe Mode Menu:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Settings Button

3.4.3. Tic Tac Toe – Single Player:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe – Single Player → Settings Button

3.4.4. Tic Tac Toe – Two Player:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe – Two Player → Settings Button

3.5. Mute/Unmute Button:

The following instructions highlight the various ways of accessing the “Mute/Unmute” button in the game package. Note: The “Mute/Unmute” button cannot be accessed within the Hangman Game Modes (Single Player, Two Player and Versus) because of the strategic gameplay method mentioned above in the “Settings” button information tab of this document.

3.5.1. Main Menu:

Splash Screen → Main menu → Mute/Unmute

3.5.2. Hangman Mode Menu:

Splash Screen → Main Menu → Hangman Mode Menu → Mute/Unmute Button

3.5.3. Tic Tac Toe Mode Menu:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Mute/Unmute Button

3.5.4. Tic Tac Toe – Single Player:

Splash Screen → Main Menu → Tic Tac Toe – Single Player → Mute/Unmute Button

3.5.5. Tic Tac Toe – Two Player:

Splash Screen → Main Menu → Tic Tac Toe – Two Player → Mute/Unmute Button

3.6. Main Menu/Leave Game Button:

The following set of instructions indicate the different ways of accessing the “Return/Leave Game” button within the game package. The button can be accessed on all forms except for the “Main Menu” because there is no other form to fall back on after the “Return” function has been met.

3.6.1. Hangman Mode Menu:

Splash Screen → Main Menu → Hangman Mode Menu → Main Menu Button

3.6.2. Hangman – Single Player:

Splash Screen → Main Menu → Hangman – Single Player → Leave Game Button

3.6.3. Hangman – Two Player:

Splash Screen → Main Menu → Hangman – Two Player → Leave Game Button

3.6.4. Tic Tac Toe Mode Menu:

Splash Screen → Main Menu → Tic Tac Toe Mode Menu → Main Menu Button

3.6.5. Tic Tac Toe – Single Player:

Splash Screen → Main Menu → Tic Tac Toe – Single Player → Leave Game Button

3.6.6. Tic Tac Toe – Two Player:

Splash Screen → Main Menu → Tic Tac Toe – Two Player → Leave Game Button

3.7. Game Settings Button:

Below is the possible way of accessing the “Game Settings” button. Note: The button is only accessible through the “Hangman Mode Menu”. This is to eliminate the possibilities of the user selecting & changing the game settings of any Hangman mode in-game.

3.7.1. Hangman Mode Menu:

Splash Screen → Main Menu → Hangman Mode Select → Game Settings Button

3.8. Screens/Images:

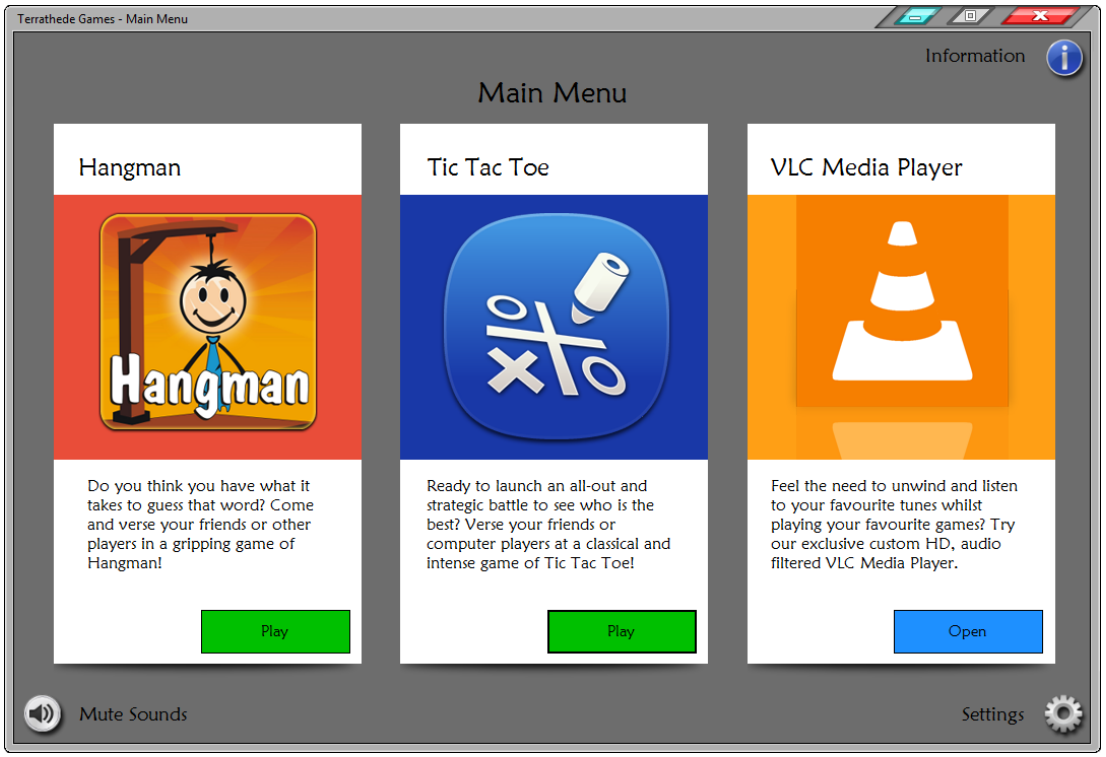
3.8.1. Splash Screen:

With the splash screen, the variety of vibrant colors sets the exciting tone of the game before being introduced to the Main Menu. It also builds up “Hype” to the essence of the game package.

Splash Screen → Main Menu
(*Proceeds to Main Menu directly after 5 seconds*)

3.8.2. Main Menu:

The Main Menu allows users to select any feature of the game package they choose.



Main Menu →

Hangman Mode Menu
(*Proceeds to “Hangman Mode Select” by clicking on “Play” button under Hangman*)

Main Menu → Tic Tac Toe Mode Menu
(*Proceeds to “Tic Tac Toe Mode Select” by clicking on “Play” button under Tic Tac Toe*)

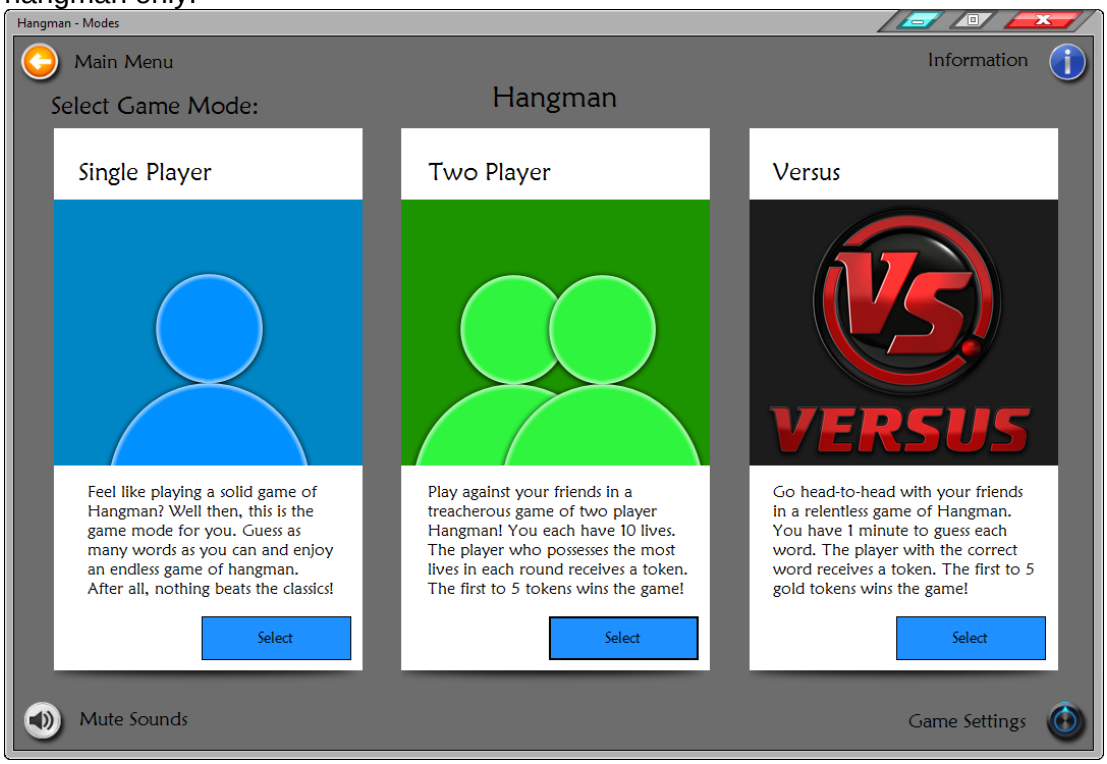
Main Menu → VLC Media Player
(*Opens up a separate “.exe” containing the VLC Media Player application*)
Main Menu → Mute/Unmute Button
(*Mutes/Unmutes all application response sounds*)

Main Menu → Information
(*Opens up a separate application, displaying the information for the game package*)

Main Menu → Settings
(*Opens up another form, displaying all the settings for the application*)

3.8.3. Hangman Mode Menu:

The Hangman Mode Menu allows users to select any game mode within the parameters of hangman only.



Hangman Mode Menu → Hangman – Single Player
(*Proceeds to "Hangman – Single Player" Mode by clicking on "Select" button under "Single Player"*)

Hangman Mode Menu → Player Names Dialog → Hangman – Two Player
(Prompts the user(s) to enter their names before *proceeding to "Hangman – Two Player" Mode by clicking on "Select" button under "Two Player"*)

Hangman Mode Menu → Player Names Dialog → Hangman – Versus
(Prompts the user(s) to enter their names before *proceeding to "Hangman – Versus" Mode by clicking on "Select" button under "Versus"*)

Hangman Mode Menu → Main Menu Button
(*Returns to the "Main Menu" by clicking on "Main Menu" button*)

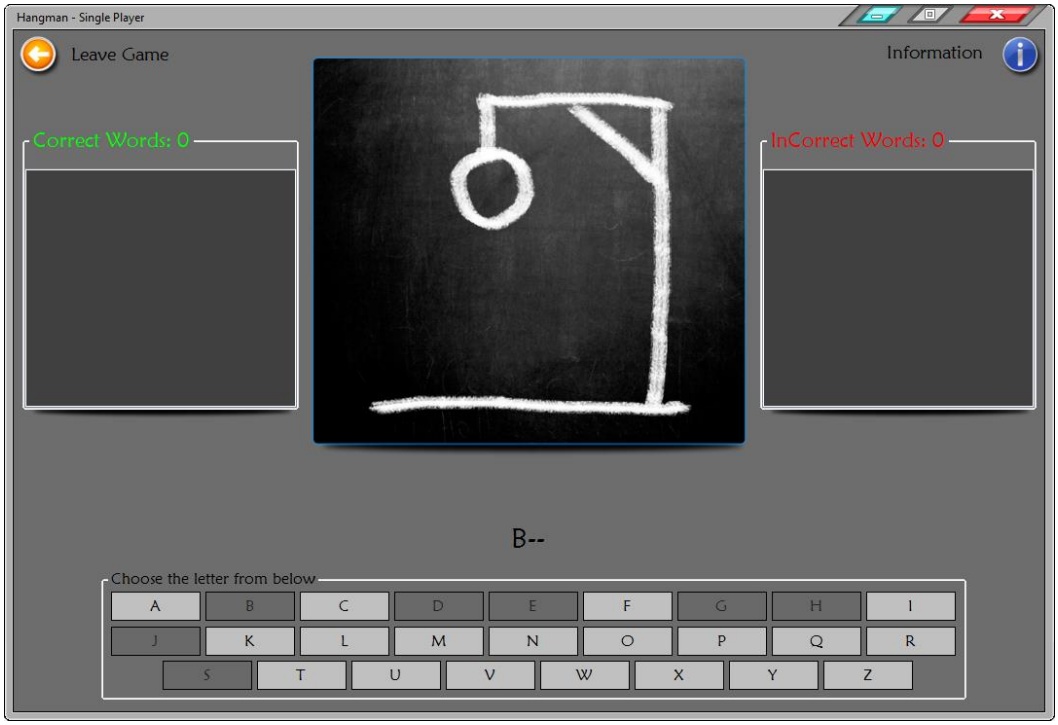
Hangman Mode Menu → Information Button
(*Opens up a separate application, displaying the information for the game package*)

Hangman Mode Menu → Mute/Unmute Button
(*Mutes/Unmutes all application response sounds*)

Hangman Mode Menu → Game Settings Button
(*Opens up another form, displaying all the game settings for Hangman*)

3.8.4. Hangman – Single Player:

The Hangman – Single Player form presents users with a classical (endless) game of "Hangman".

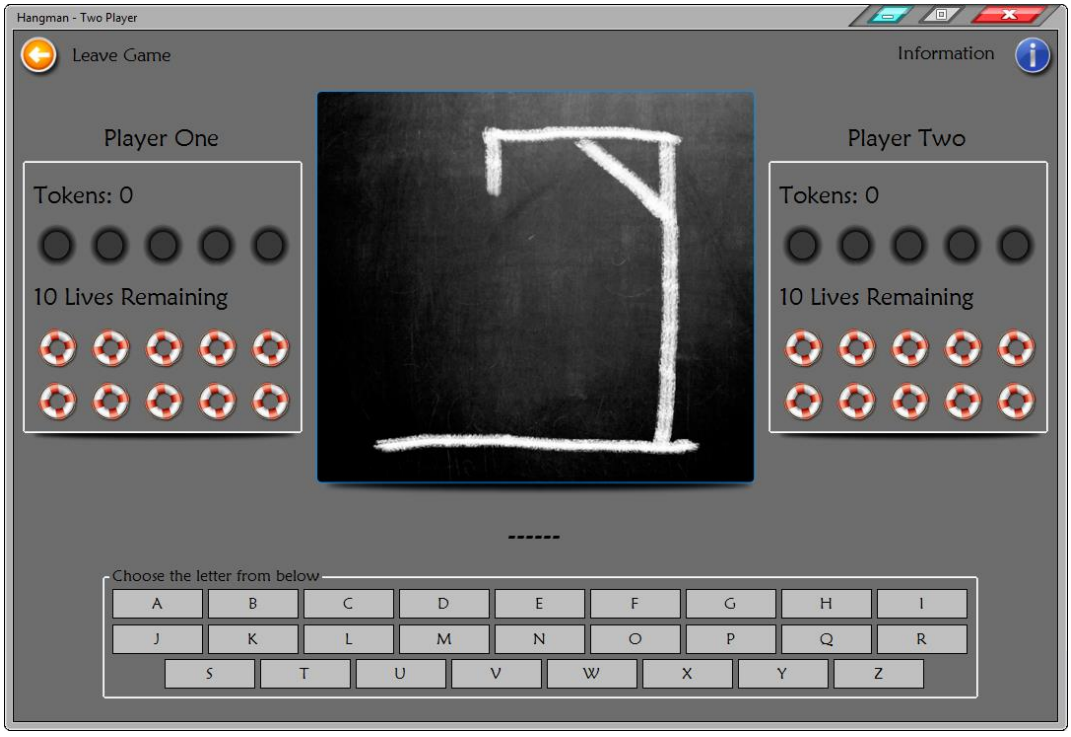


Hangman – Single Player → Leave Game Dialog → Leave Game Button
(Prompts the user to "Leave Game" then Returns to the "Hangman Mode Menu" by clicking on "Leave Game" button)

Hangman – Single Player → Information Button
(*Opens up a separate application, displaying the information for the game package*)

3.8.5. Hangman – Two Player:

The Hangman Two Player game mode prompts the user(s) to enter their name before proceeding to the game mode after the "Hangman Mode Menu".



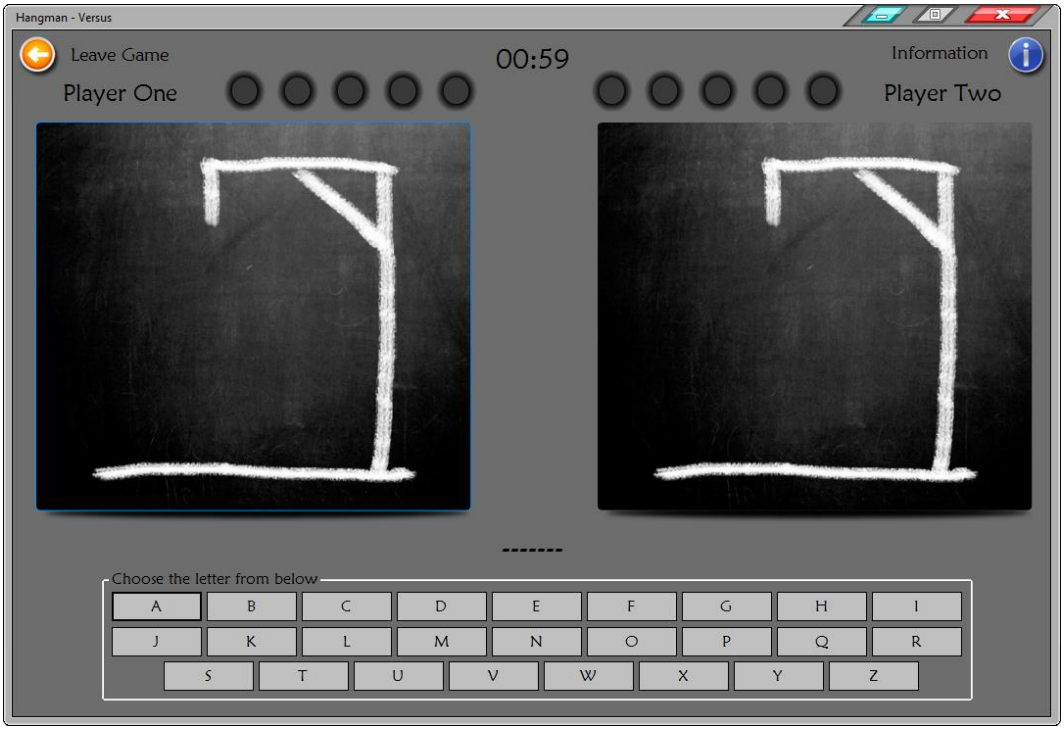
Hangman – Two Player → Leave Game Dialog → Leave Game Button
(Prompts the user(s) to "Leave Game" then Returns to the "Hangman Mode Menu" by clicking on "Leave Game" button)

Hangman – Two Player → Information Button

(*Opens up a separate application, displaying the information for the game package*)

3.8.6. Hangman – Versus:

The Hangman Versus game mode prompts the user(s) to enter their name before proceeding to the game mode after the “Hangman Mode Menu”.



Hangman – Versus → Leave Game Dialog → Leave Game Button

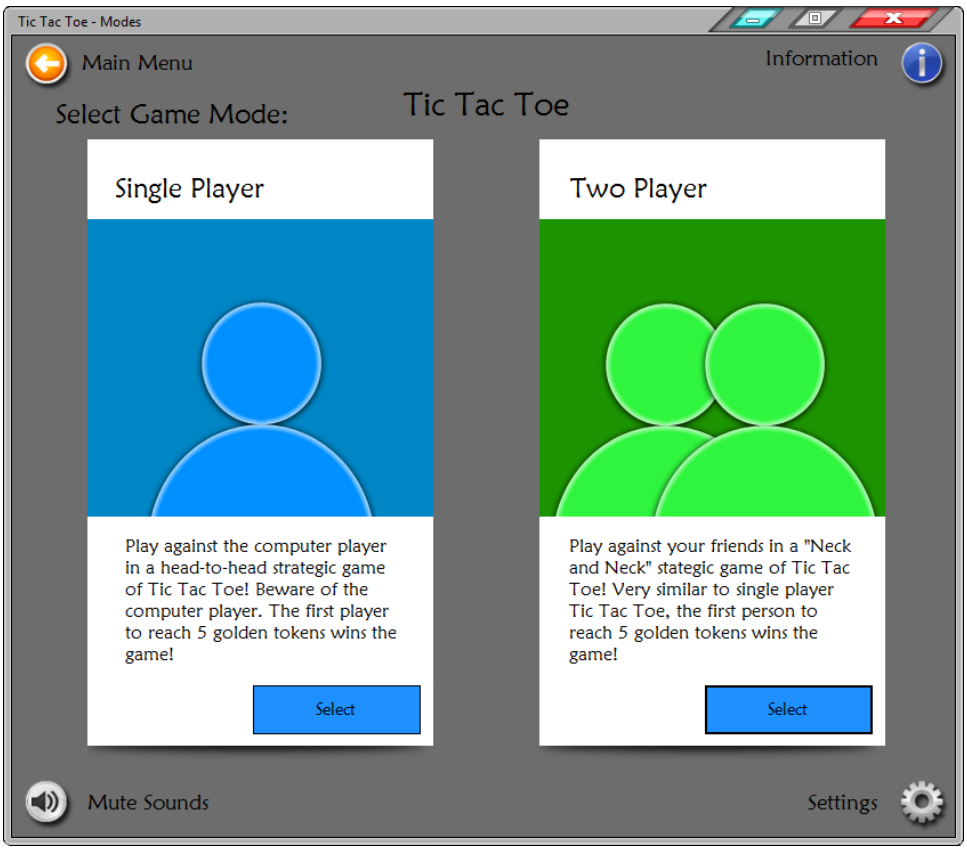
(Prompts the user(s) to “Leave Game” then Returns to the “Hangman Mode Menu” by clicking on “Leave Game” button)

Hangman – Versus → Information Button

(*Opens up a separate application, displaying the information for the game package*)

3.8.7. Tic Tac Toe Mode Menu:

The Tic Tac Toe Mode Menu allows users to select any game mode within the parameters of Tic Tac Toe only.



Tic Tac Toe Mode Menu → Tic Tac Toe – Single Player
(*Proceeds to "Tic Tac Toe – Single Player" Mode by clicking on "Select" button under "Single Player"*)

Tic Tac Toe Mode Menu → Player Names Dialog → Tic Tac Toe – Two Player
(Prompts the user(s) to enter their names before *proceeding to "Tic Tac Toe – Two Player" Mode by clicking on "Select" button under "Two Player"*)

Tic Tac Toe Mode Menu → Main Menu Button
(*Returns to the "Main Menu" by clicking on "Main Menu" button*)

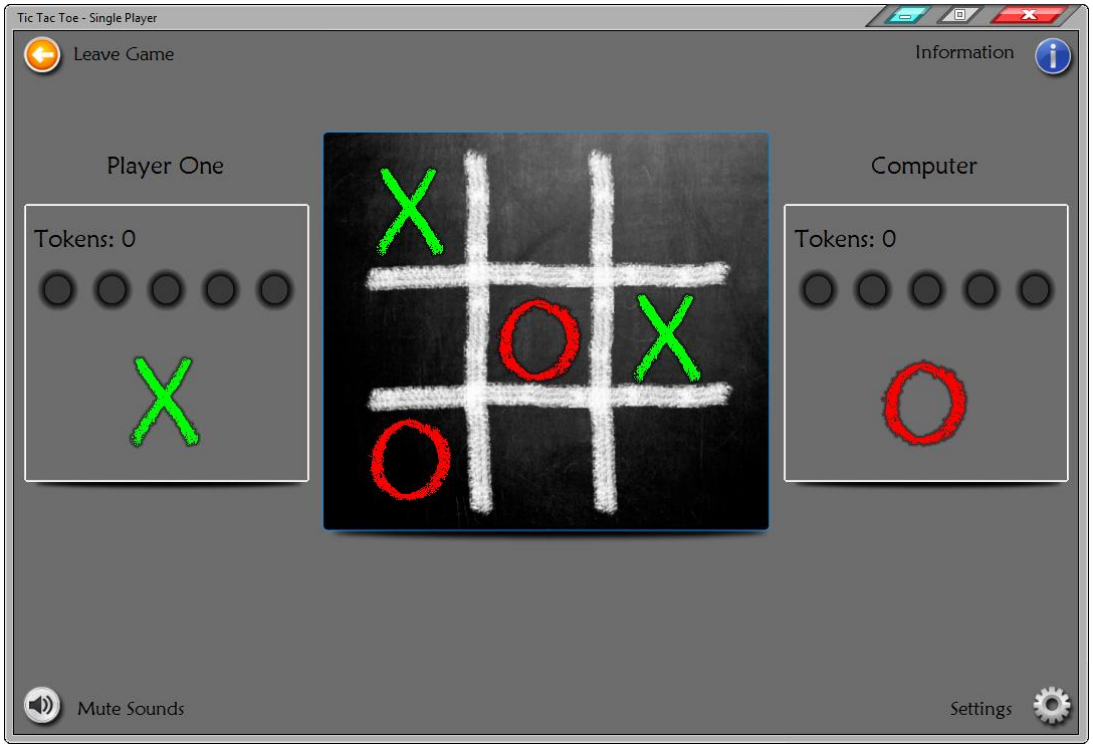
Tic Tac Toe Mode Menu → Information Button
(*Opens up a separate application, displaying the information for the game package*)

Tic Tac Toe Mode Menu → Mute/Unmute Button
(*Mutes/Unmutes all application response sounds*)

Tic Tac Toe Mode Menu → Settings Button
(*Opens up another form, displaying all the settings for the application*)

3.8.8. Tic Tac Toe – Single Player:

The Tic Tac Toe – Single Player game mode grants user's the ability to play against a computer in a classical game of Tic Tac Toe.



Tic Tac Toe – Single Player → Leave Game Dialog → Leave Game Button
(*Prompts the user to "Leave Game" then proceeds "Tic Tac Toe Mode Menu" Mode by selecting "yes" in the dialog widow)*

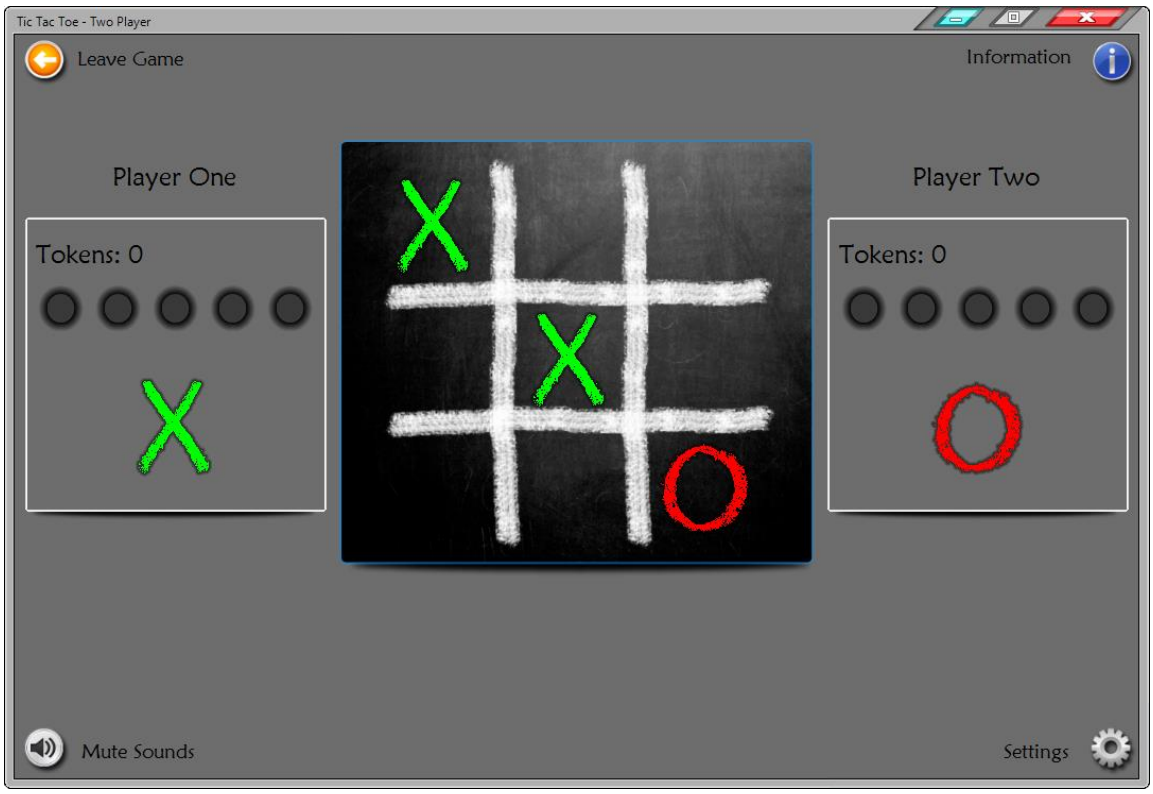
Tic Tac Toe – Single Player → Information Button
(*Opens up a separate application, displaying the information for the game package*)

Tic Tac Toe – Single Player → Mute/Unmute Button
(*Mutes/Unmutes all application response sounds*)

Tic Tac Toe – Single Player → Settings Button
(*Opens up another form, displaying all the settings for the application*)

3.8.9. Tac Toe – Two Player:

The Tic Tac Toe – Two Player game mode allows users to play against each other in a classical game of Tic Tac Toe.



Tic Tac Toe – Two Player → Leave Game Dialog → Leave Game Button
(*Prompts the user to "Leave Game" then proceeds "Tic Tac Toe Mode Menu" Mode by selecting "yes" in the dialog widow)*

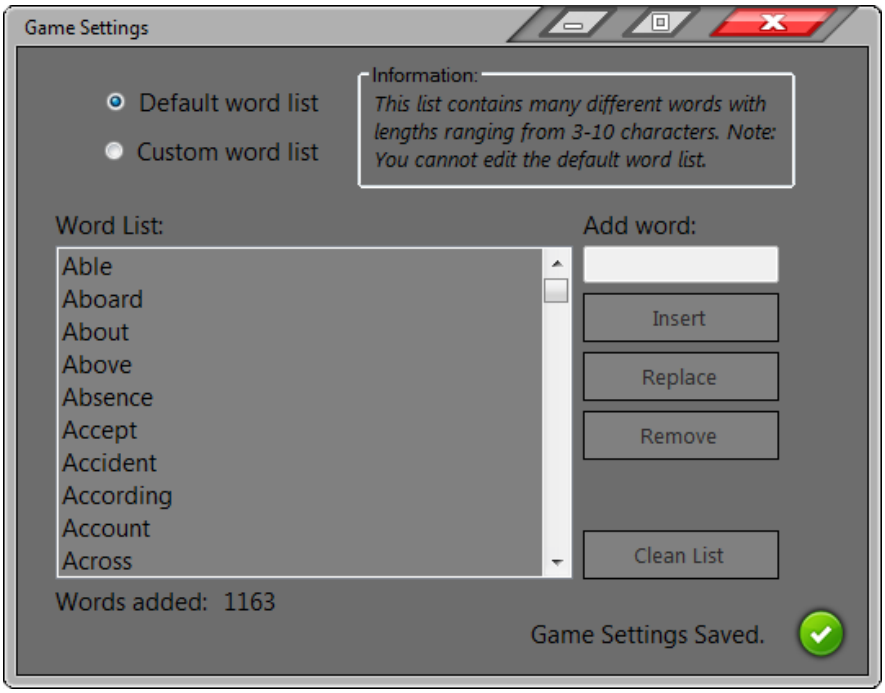
Tic Tac Toe – Two Player → Information Button
(*Opens up a separate application, displaying the information for the game package*)

Tic Tac Toe – Two Player → Mute/Unmute Button
(*Mutes/Unmutes all application response sounds*)

Tic Tac Toe – Two Player → Settings Button
(*Opens up another form, displaying all the settings for the application*)

3.8.10. Game Settings:

The Game Settings form/feature allows user(s) to select from “Default Word List”/”Custom Word list”, Insert, Replace, Remove and Clean the Custom word list.

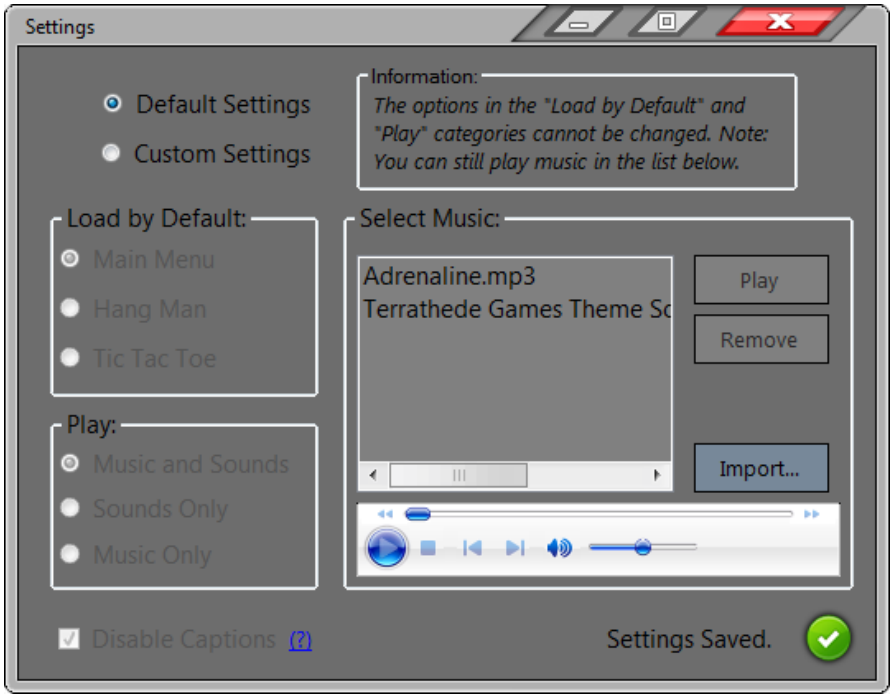


Game Settings → Close Button

(Closes the Game Settings window and proceeds to the “Hangman Mode Menu”)

3.8.11. Settings:

The Settings window allows users to adjust parameters from a range of given settings.

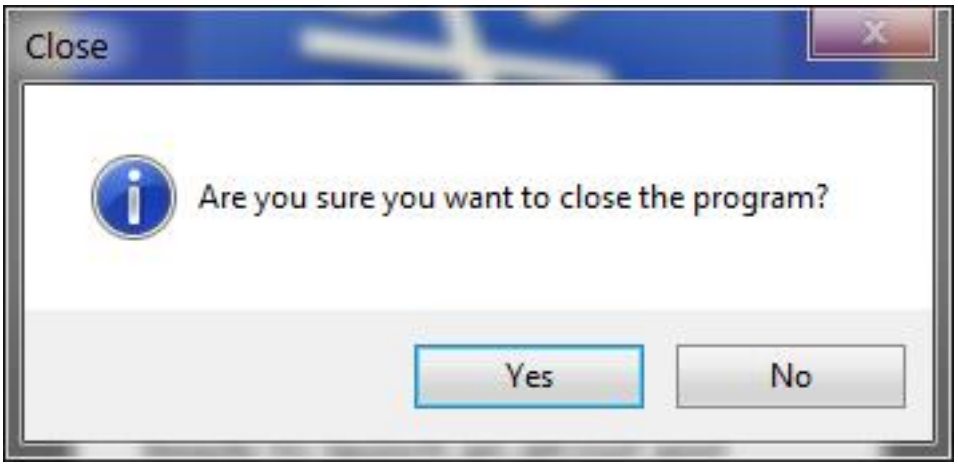


Settings → Close Button

(Closes the Settings window and returns to the window from which the user selected the “Settings” function/form)

3.8.12. Dialog box - Close:

This dialog box is displayed once the user try's the close any form within the game package.



Main Menu → Close Button

(Closes the program through the Main Menu)

Main menu → Hangman Mode Menu → Close Button

(Closes the program through the Hangman Mode Menu)

Main Menu → Hangman Mode Menu → Hangman – Single Player → Close Button

(Closes the program through the Hangman – Single Player form)

Main Menu → Hangman Mode Menu → Hangman – Two Player → Close Button

(Closes the program through the Hangman – Two Player form)

Main Menu → Hangman Mode Menu → Hangman – Versus → Close Button

(Closes the program through the Hangman – Versus form)

Main Menu → Tic Tac Toe Mode Menu → Close Button

(Closes the program through the Tic Tac Toe Mode Menu)

Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe – Single Player → Close Button

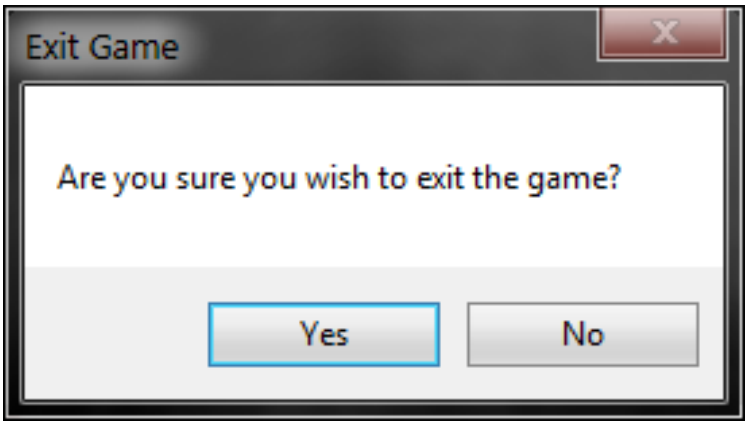
(Closes the program through the Tic Tac Toe – Single Player form)

Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe – Two Player → Close Button

(Closes the program through the Tic Tac Toe – Two Player form)

3.8.13. Dialog box – Exit Game:

This dialog box is displayed once the user try's to leave the game once the game has already started. This dialog box is only showed in all the Hangman modes and all the Tic Tac Toe modes. It does not show in any of the menus.



Main Menu → Hangman Mode Menu → Hangman – Single Player → Leave Game Button
(Displays the dialog box once the user clicks the "Leave Game" Button)

Main menu → Hangman Mode Menu → Hangman – Two Player → Leave Game Button
(Displays the dialog box once the user clicks the "Leave Game" Button)

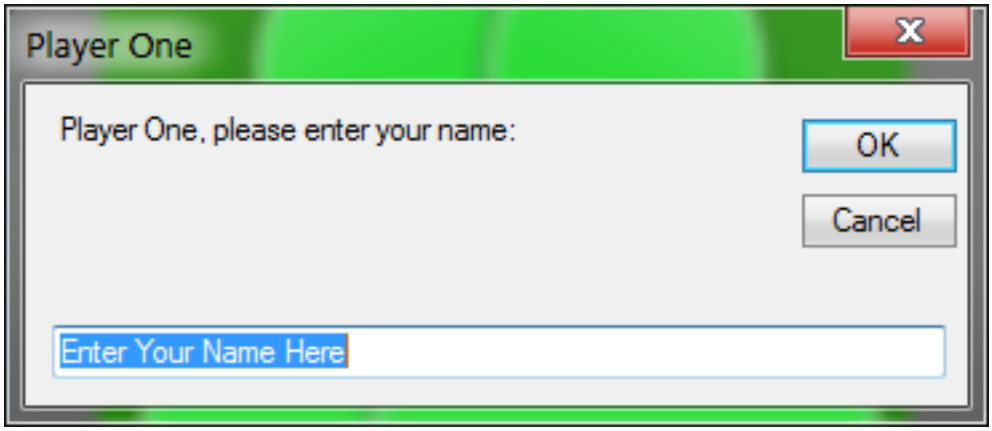
Main Menu → Hangman Mode Menu → Hangman – Versus → Leave Game Button
(Displays the dialog box once the user clicks the "Leave Game" Button)

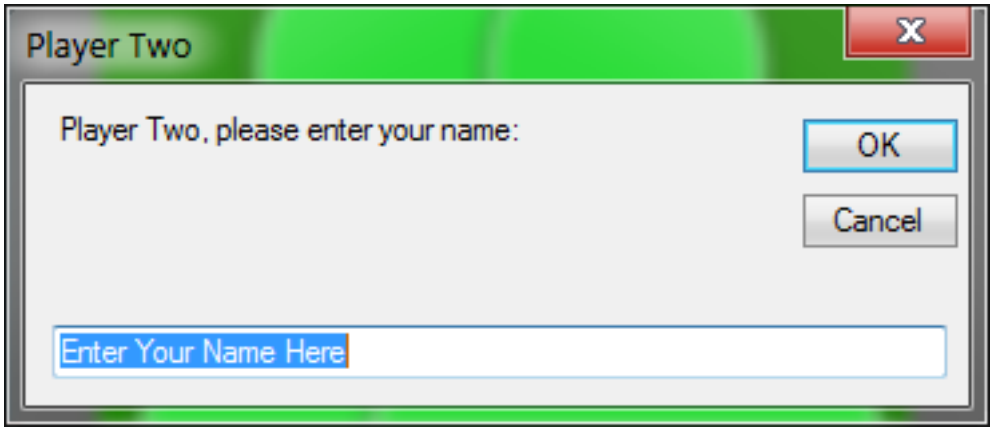
Main Menu → Hangman Mode Menu → Tic Tac Toe – Single Player → Leave Game Button
(Displays the dialog box once the user clicks the "Leave Game" Button)

Main Menu → Hangman Mode Menu → Tic Tac Toe – Two Player → Leave Game Button
(Displays the dialog box once the user clicks the "Leave Game" Button)

3.8.14. Dialog box – Player One/Player Two:

The input box (dialog) is displayed once the user(s) select a two-player game in either game (Hangman/Tic Tac Toe) or versus in Hangman. Note: Player One dialog comes before Player Two Dialog.





Main Menu → Hangman Mode Menu → Hangman – Two Player → Player One Input dialog → Player Two Input dialog

(Displays the dialog box once the user(s) click the "Hangman – Two Player" 'select' Button)

Main menu → Hangman Mode Menu → Hangman – Versus → Player One Input dialog → Player Two Input dialog

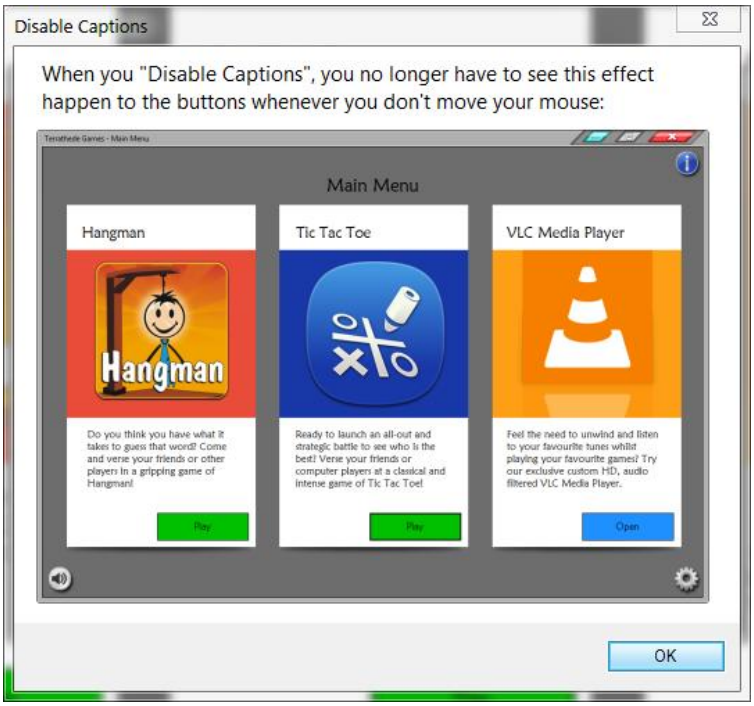
(Displays the dialog box once the user(s) click the "Hangman – Versus" 'select' Button)

Main Menu → Tic Tac Toe Mode Menu → Tic Tac Toe – Two Player → Player One Input dialog → Player Two Input dialog

(Displays the dialog box once the user(s) click the "Tic Tac Toe" 'select' Button)

3.8.15. Dialog box – Disable Captions:

This dialog shows up once the user selects the link label "(?)" in the "Settings" form.



Settings → "(?)" Link Label → Disable Captions dialog box

(Displays the dialog box once the user selects the "(?)" link label)

4. Screen Designs:

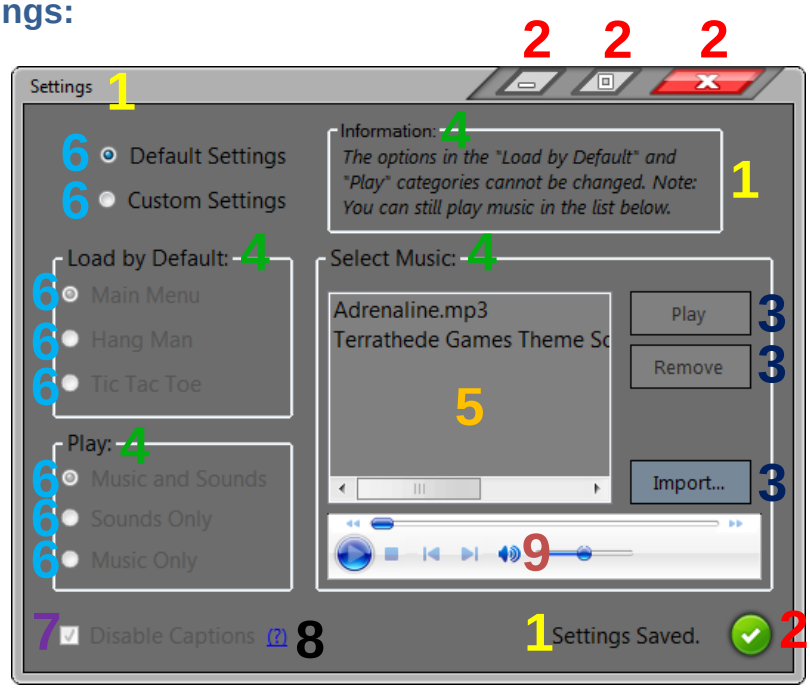
With the screen designs, appropriate screen elements as well as fonts, colors and layout of the game package were carefully chosen to allow a more aesthetic approach to the user's appeal of the program.The following information outlines all of the screen elements used in every form as well as their purpose.

4.1. Main Menu:

1. Labels
2. Picturebox
3. Buttons

The Main Menu is comprised of labels (1) and pictureboxes (3). The labels clearly display the form's interactivity to the user as well as what each feature does within the form. The pictureboxes act as graphically enhanced buttons that allow the user to easily identify them and interact with them. The buttons (3) are also easily identifiable as they are color coded and allow an easy distinction between them.

4.2. Settings:



1. Labels
2. Picturebox
3. Buttons
4. Groupboxes
5. Listboxes
6. Radio buttons
7. Checkboxes
8. Link labels
9. Media Player OCX (Active X Control)

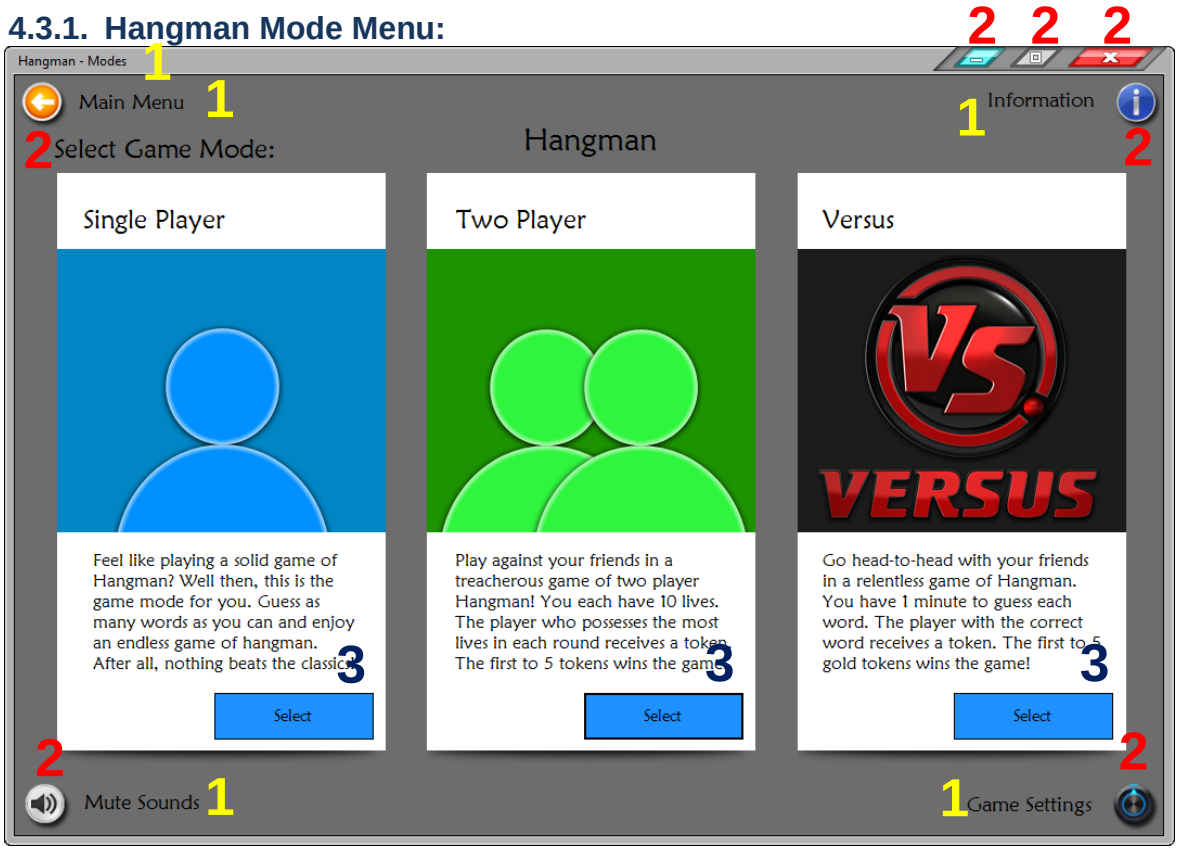
The Settings screen is the only form which is comprised of the most screen elements. The screen elements used in this form are labels (1), pictureboxes (2), buttons (3), groupboxes (4), listboxes (5), radio buttons (6), checkboxes (7), link labels (8) and a Media Player OCX (9). The labels used in this form allow the user to identify certain functions of the form that raises questions by the user. The pictureboxes are used to provide the user with a more graphical interpretation of the user interface. The buttons in this screen design are used to allow the user to navigate between functions. There is a play button which enables itself once the user has selected a file that has a ".mp3" extension within the listbox (5). The remove button allows the user to delete media files that have been imported to the program via "Music" file in the program's directory. The import button grants the user with the ability to import various media files to the listbox using the "File.Copy" method in visual basic. The media player OCX (9) is the most important part to the playback function as it is used to play the selected music from the listbox using the "Directory" method (Plays a media file using the file directory).

This form also consists of a variety of interchangeable options using radio buttons (6) and a checkbox (7). The user can choose between default settings/custom settings by selecting either radio buttons. If the user selects default settings then the program automatically changes the settings of the parameters to what they originally were when the user first installed the application and the parameters are disabled. If the user selects the custom settings radio button then all the parameters are available to be changed. They are all in an individual group box (4) to categorize the parameters so that the user can easily identify where the settings are and what they wish to change. The checkbox (7) is the only one that sits outside of a groupbox. The "Disable Captions" checkbox allows the user to either have all the labels pop up next to the picturebox on several forms or to have this option disabled (by checking the box with a tick). The link label (8) is used to allow users who are unable to identify what this unusual function does, by clicking on the "(?)" users are then presented with a dialog box outlining exactly what the function does by enabling/disabling the checkbox.

4.3. Hangman:

The following diagrams highlight the variety of different screen elements used within each form of the game Hangman.

4.3.1. Hangman Mode Menu:

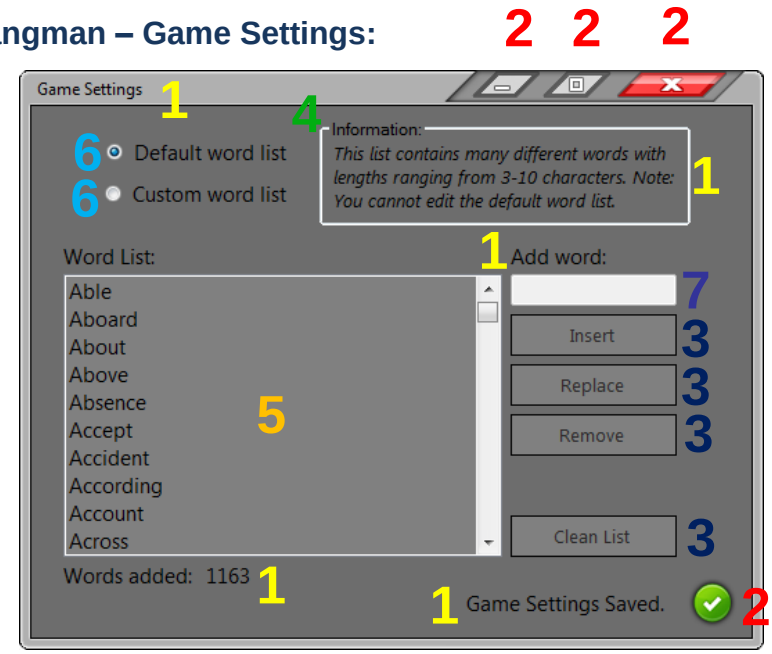


1. Labels
2. Picturebox
3. Buttons

The Hangman Mode Menu, as consistent as the main menu, provides users with easy navigation between forms. Once the user selects a button (3) then the corresponding action takes place. For example, if the user selects the button underneath the “Single Player” option, then the form opens without any prompts. However, if the user selects either “Two Player” or “Versus” hangman, then an input box will prompt the user to enter Player One’s name and Player Two’s name. This is because both the “Two Player” and “Versus” game modes require two players to enter the game.

Just like the main menu (and various other forms) the pictureboxes act as buttons. Though, there are still actual button components used in the screen design, the pictureboxes (2) provide users with a more enhanced concept of the program’s animations. With the animations, they are exactly the same throughout the whole game package so that users are familiarized with the interface to adhere to the specifications of the user and how they navigate through applications other than this one. Labels (1) are used to give appropriate visualization as to what picturebox/button leads to and what form will open during the course of this action that the user selects.

4.3.2. Hangman – Game Settings:



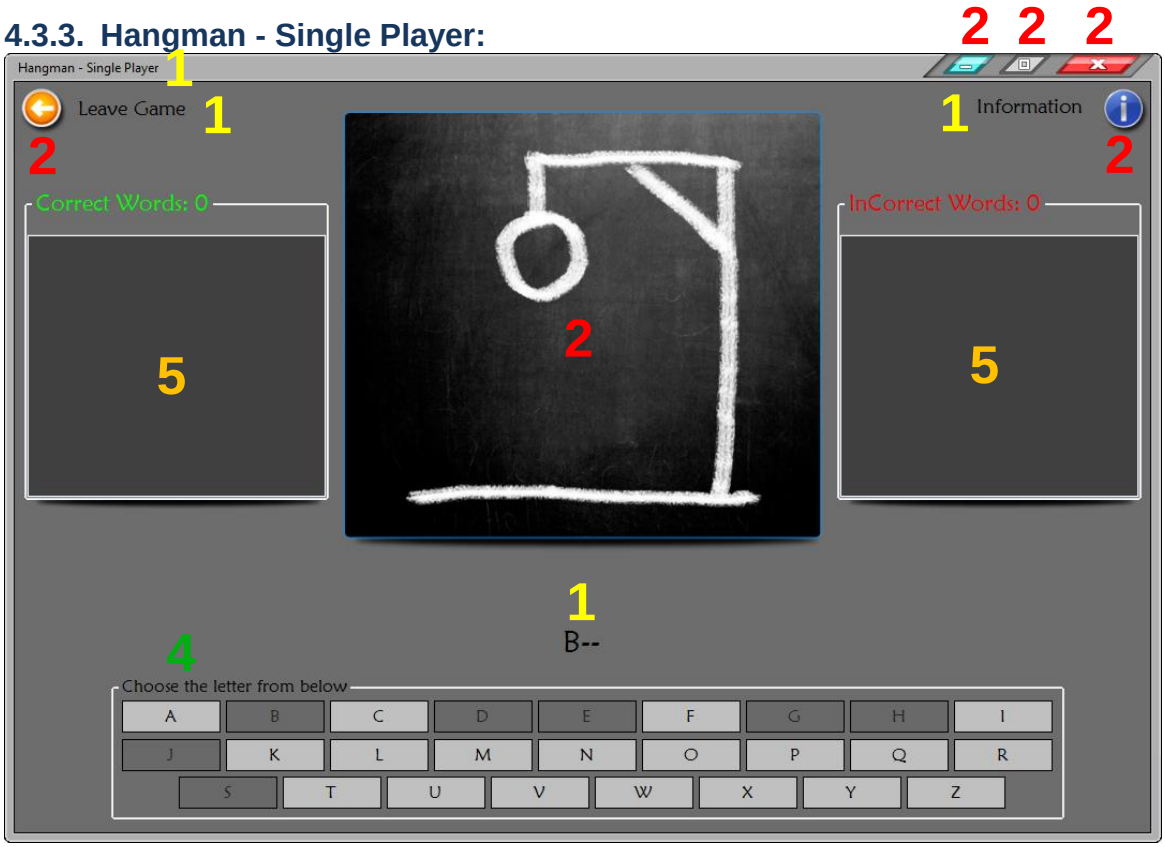
1. Labels
2. Picturebox
3. Buttons
4. Groupboxes
5. Listboxes
6. Radio buttons
7. Textboxes

The Hangman – Game Settings screen design uses a variety of screen elements. It uses labels (1), pictureboxes (2), buttons (3), groupboxes (4), listboxes (5), radio buttons (6) and textboxes (7). The labels are uniquely used in this screen design to provide the user with as much information about this form as possible. The labels aren't placed in various random places but are put I rather significant ones. The pictureboxes (like all other forms) are used to act like buttons. However, in this form, it has a more significant purpose. It uses a .gif image guided by a timer as well as the label's changing text to improve the user interface more productively.

The buttons are very important in this screen as they provide the user with a variety of different functions. The insert button allows the user to insert words that they have already entered into the textbox (7). The replace button gives the user access to replace the selected word in the listbox (5) with the word they have entered previously into the textbox. The remove button provides users with the option to delete words from the listbox providing they select the word before pressing the button. The clean list button grants users with the ability to remove all words in the custom word list and this function will automatically enter 10 random words from the default word list.

The radio buttons are essential in allowing the user to select from either the custom word list or default word list. If the custom word list radio button (6) is selected then all the buttons are enabled and you are able to see them and access their functions. If the default word list radio button is selected, users cannot use any of the button's functions. Therefore, they cannot edit the default word list. All screen elements in this screen design work coherently to provide users with a unique experience in the graphical user interface (GUI).

4.3.3. Hangman - Single Player:



1. Labels
2. Pictureboxes
3. Buttons
4. Groupboxes
5. Listboxes

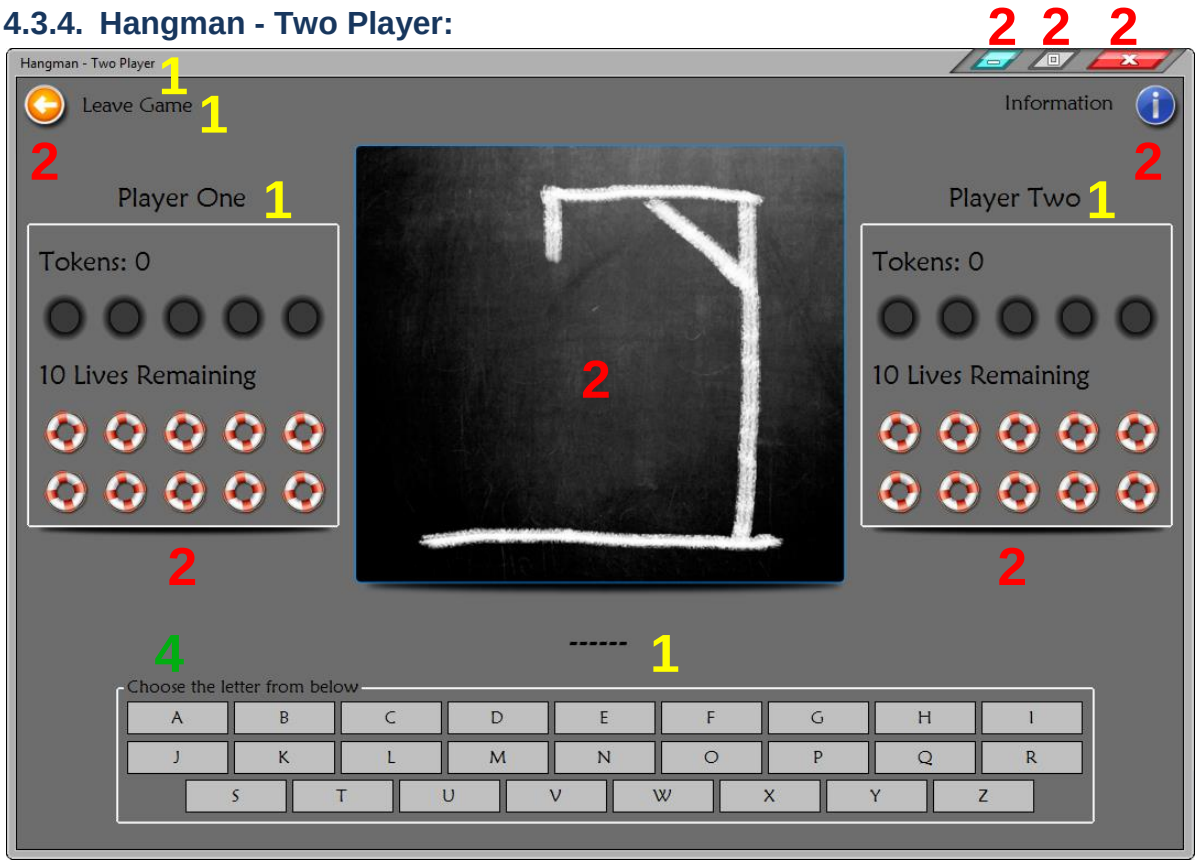
The Hangman – Single Player form is comprised of a variety of different screen elements. It consists of labels (1), pictureboxes (2), buttons (3), groupboxes (4) and Listboxes (5). The labels are self-explanatory, but in this form, the one of the labels has a huge impact to the game's quality. In this form, the label is used as a place where the word is generated in the middle of the screen and provides the user with a basic idea of what letters are missing from the word itself. When the user selects the "Start" button in this form. The word starts off with a row of hyphens "----". As the user selects the various buttons from the groupbox, if the word contains the user's selected letter then that letter gets represented in the hyphen (-) containing that letter.

The buttons (3) in this form are mainly letters in the groupbox. These buttons are very important as they serve two purposes:

1. Graphically display the letters that the user had already chosen. (Changes from "Gray" to "Dark Gray" and disables itself to signify that the word cannot be selected anymore)
2. Provide users with a widened and visible selection from the alphabet. This eliminates the need to display the letters already chosen in a label and thus, saves space in the form to add appropriate screen elements.

The groupbox (4) also serves a significant purpose in this game mode. It narrows the user's focus towards the letters that are available and keeps their line of sight within the game's parameters. The listboxes (5) in this form keep a distinction between words that have been guessed correctly and words that were guessed incorrectly. The listbox on the left (←) displays the list of words that the user has guessed correctly. Whereas, the listbox to the right (→) shows the list of words that the user has guessed incorrectly.

4.3.4. Hangman - Two Player:



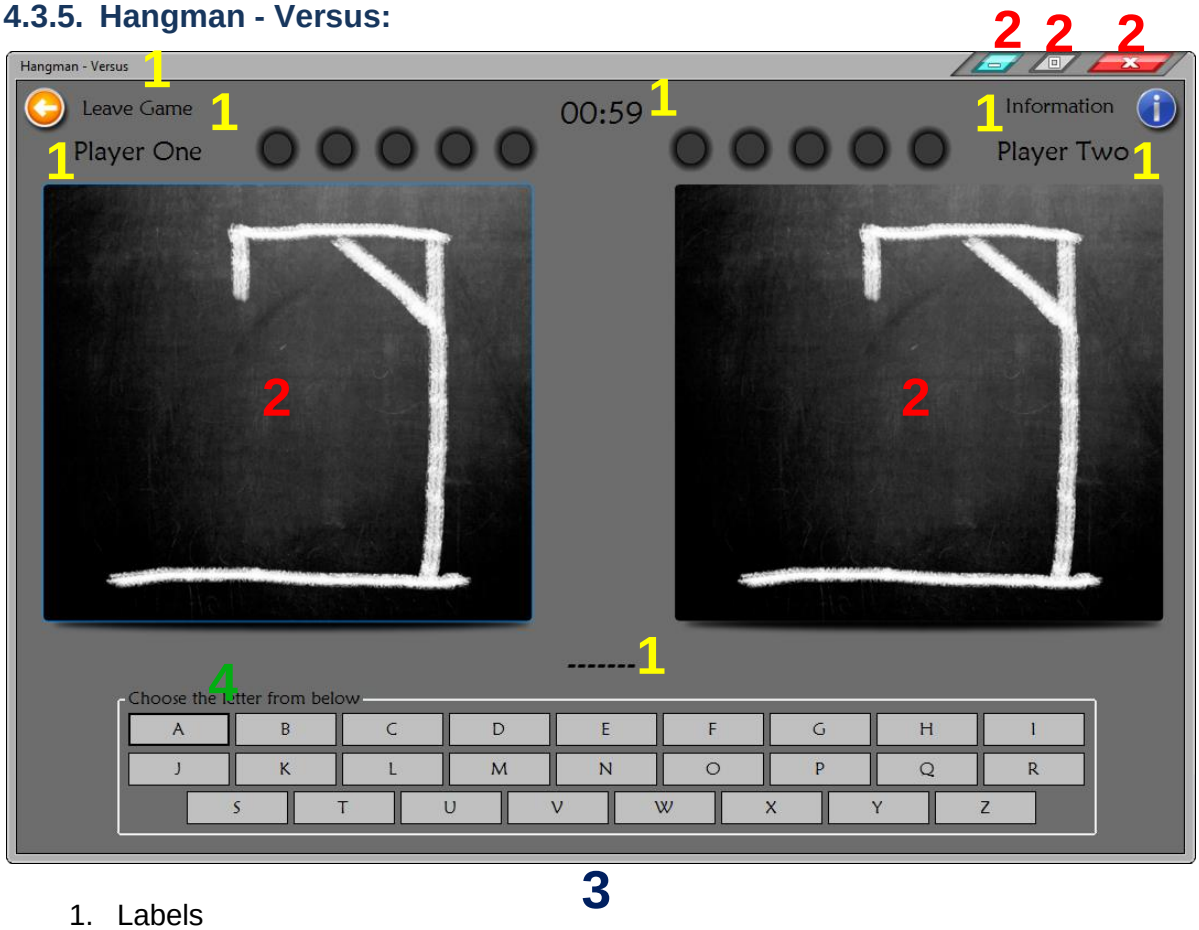
1. Labels
2. Pictureboxes
3. Buttons
4. Groupboxes

The Hangman – Two Player form consists of only 4 different screen elements. The labels used in this form serve the exact same purpose as the Hangman - single player form, it initially displays the generated word's length using hyphens and when the user selects a letter from the buttons in the groupbox (4) and if that letter is within the generated letter (whether it contains the same letter several times In the word) the correct letter(s) are presented to the user by displaying the better in the location of the hyphens (-).

As for the pictureboxes (2), they serve an enormous purpose in the qualifications of this game mode. The "lives" system used in this game use quite an irregular symbol as opposed to other games using "hearts" for lives where this uses actual "life savers". This concept was used to have a profound effect towards the user so that they can easily identify the distinction between the two principles and disregard that factor and see it as "Ironic" (that the program uses actual "life savers" other than the traditional "heart" system).

In this game mode, both players are asked to give each other words in different intervals of the game. When the game starts, player two is the first to give player one a word. Once player one gets a word, that word is placed within the length of hyphens (-) hiding the full word from the player. As the user's selects the letters from the buttons in the groupbox (4), each letter they get incorrect consequences in the deduction of 1 life point. As soon as player one gets the word correct, the players switch and player one is the one who gives player two a word to guess. However, if player one was unsuccessful in guessing the word, this results in a draw, as it would have an unfair advantage to player one by giving player two a token. Once the players have switched turns, player two mustn't lose more lives than player one or this will result in player one's receiving of a token. This concept is repeated until the player who receives a total of 5 golden tokens wins the game.

4.3.5. Hangman - Versus:



1. Labels
2. Pictureboxes
3. Buttons
4. Groupboxes

In the Hangman – Versus form, players are introduced to a fast-pacing game mode that requires both players to guess the given (generated) word as fast as possible. Labels (1) in this form are used to indicate a number of things. One major difference that this game mode has as opposed to other forms is the game clock at the top of the form. This clock represents the time that both users have left to guess the current generated word in that is represented by another label using hyphens (-). The clock starts at “1:00” and users have to both work together in solving the same word but be against each other at the same time. This game mode is quite compelling as it is designed to make opponents team up and play against each other simultaneously.

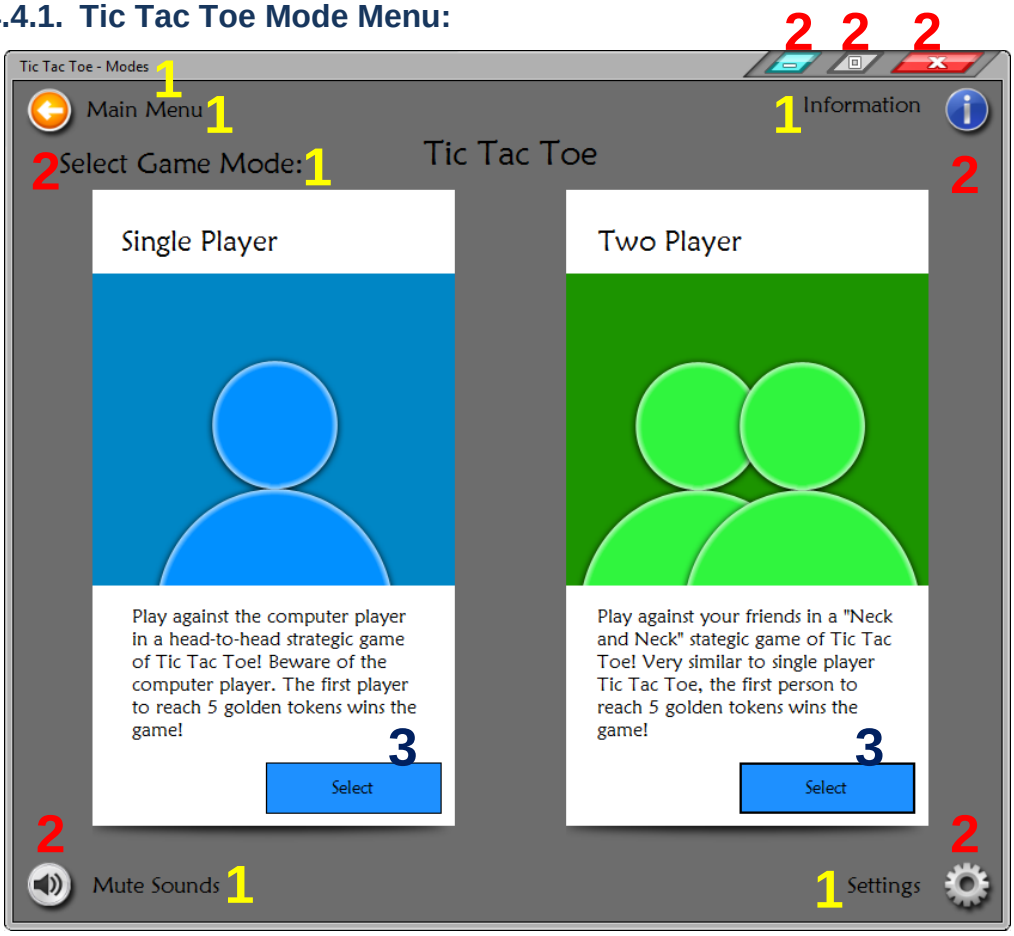
The pictureboxes have a unique twist in this form as it collects the current player’s selected letter and identifies whether that word is valid within the generated word given to both players. The player who guesses the entire word (the player with the last selected letter) receives a token which is displayed through a picturebox (the same goes with all the other hangman and tic tac toe forms). The groupbox (4) in this form has a huge impact to both users as it grabs both player’s attention whilst they are both trying to figure out what word has been generated by the word list.

The buttons (3) in this game mode hold a strong significance to the rest of the gameplay. As users take turns in guessing letters the letters that have been selected are still disabled for both users. This means that the selection of letters doesn’t reset every time a player gets a wrong letter but it keeps going to give players a little bit of a boost whilst trying to guess the generated word.

4.4. Tic Tac Toe:

The following diagrams/screen designs highlight the variety of different screen elements used within each game mode form of the game Tic Tac Toe.

4.4.1. Tic Tac Toe Mode Menu:



1. Labels
2. Pictureboxes
3. Buttons

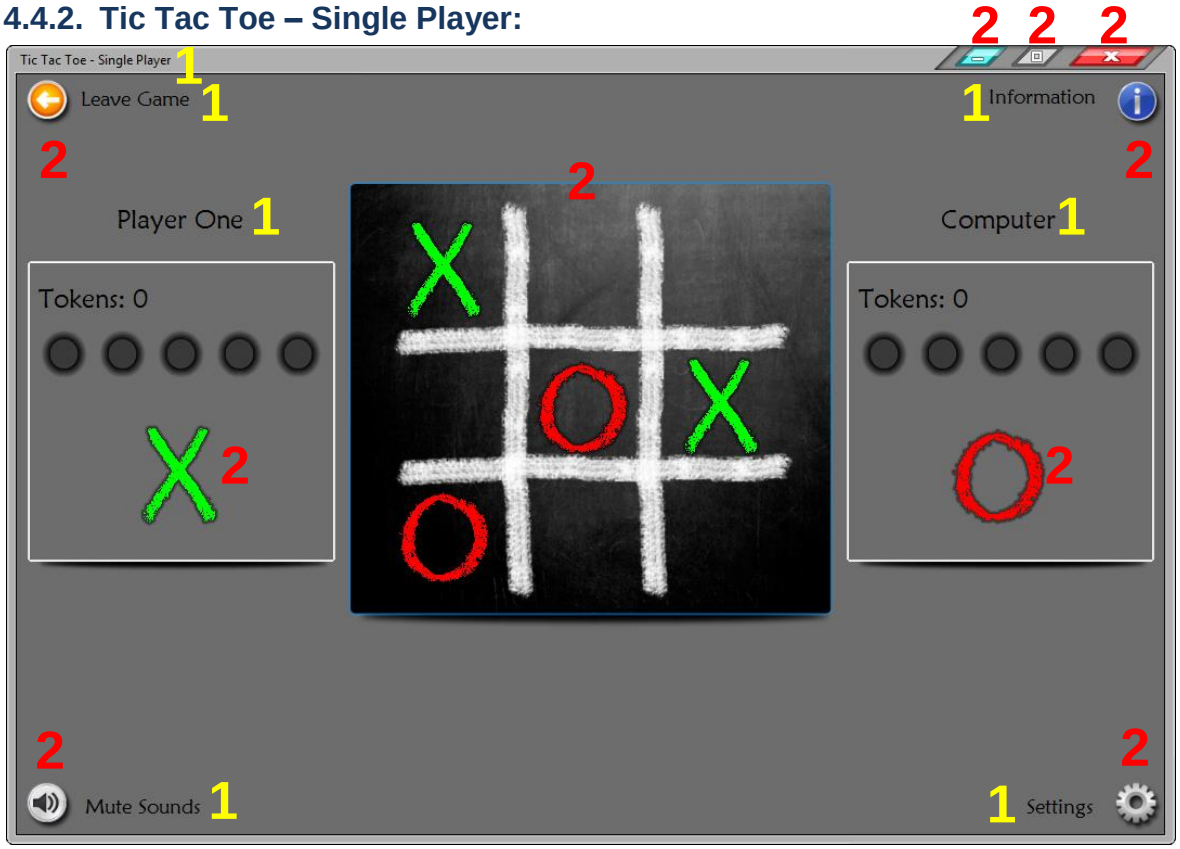
The Tic Tac Toe Mode Menu consists of only labels (1), pictureboxes (2) and buttons (3). In this screen, the labels are used to contrast what aspects of the form that the user would have difficulty in identifying (as with all forms of the game package). The pictureboxes still have a consistent layout to the form as with other forms where it provides users with a more enhanced idea of how to navigate within the form.

In this screen, there are two major factors that differentiates this form to the “Hangman Mode Menu” form:

1. The “Settings” function from the Main Menu is brought to this form as opposed to the Hangman Mode Menu which, in its place, has “Game Settings”. There is a huge significance as to why this form has “Settings” and not “Game Settings”.
2. This form no longer supports the “Versus” game mode. This is due to the fact that the game mode is not a necessarily needed in the quality of this application. Also, it gives each game a significant amount of distinction between each other, allowing users to find it aesthetically pleasing.

The buttons (3) in this screen are used to link each game with the appropriate form/sub module that opens after clicking either one of the “Select” buttons.

4.4.2. Tic Tac Toe – Single Player:

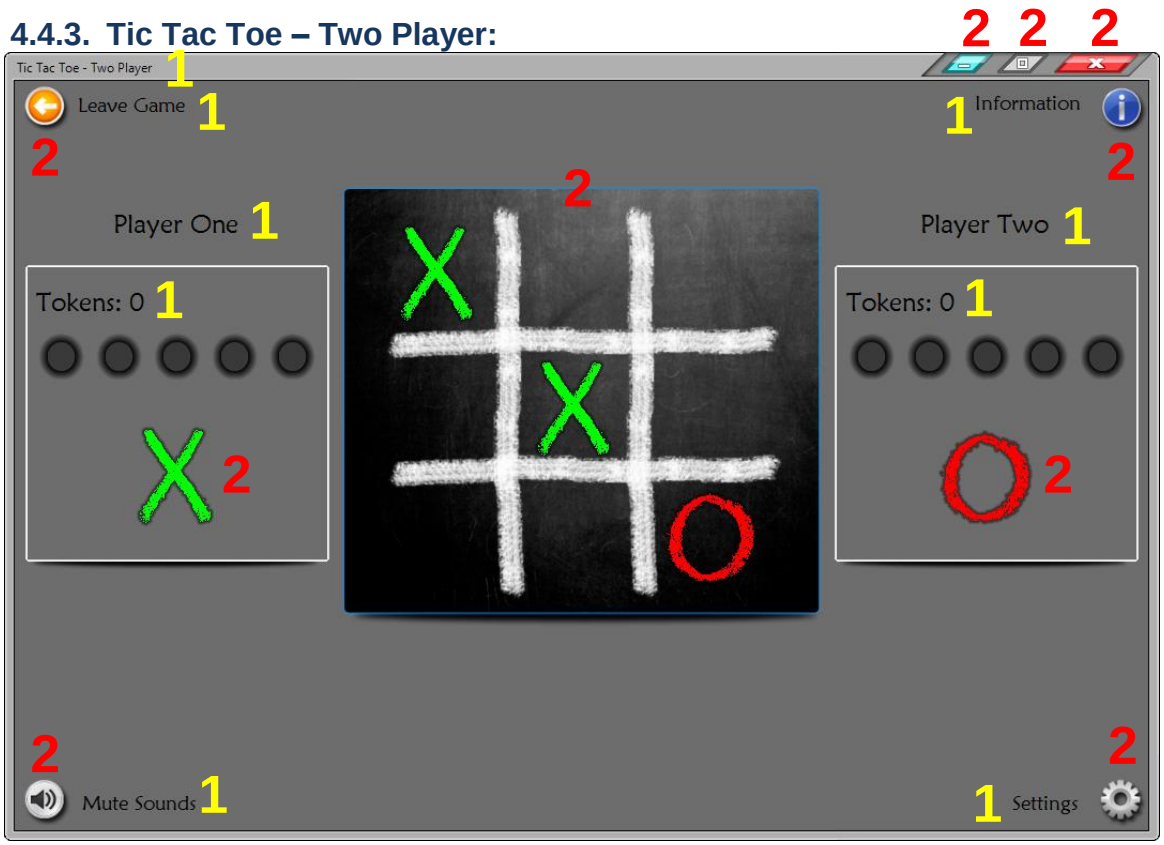


1. Labels
2. Pictureboxes

In this game mode, appropriate screen elements have been used to provide the user with a variety of different options. This game mode is not as complex as other screen designs due to the fairly low usage of screen elements. However, the purpose of this is significant in its own way. This game mode does not require a large amount of navigation. Therefore, the amount of screen elements needed is significantly low. There are pictureboxes (like in all other forms) that are used to act like buttons. They also add to the graphical quality of the user interface and provide the user with a more aesthetically pleasing screen design.

The labels used in this screen design give the user the ability to identify all the other screen elements are and briefly represents their functions to a specific event. I.e. click event. The screen design of this game mode is comprised of only labels and pictureboxes, though, the screen elements used in this form (as mentioned above) each consist of a specific purpose in their own way.

4.4.3. Tic Tac Toe – Two Player:



1. Labels
2. Pictureboxes

The Tic Tac Toe – Two Player game mode, much like single player, is comprised of only labels (1) and pictureboxes (2). Although, there is a difference between Tic Tac Toe – Two Player and Tic Tac Toe – Single Player. That is, two player requires two users to be using the variety of screen elements at the same time (switching turns). Both players are initially introduced to an empty board with only a “Start” button (just like all the games in the game package). Player one and player two are then left with just an empty board with no markers added to it (after clicking the “Start” button). Once the user(s) enter the mouse into the board, there is a small circle changing between red and green to signify whose marker is about to be placed once the user clicks on the board. This is repeated until the player who receives 5 golden tokens wins the game.

5. Algorithms:

```
BEGIN Mainprogram
        SplashScreen
        IF user selects Hangman THEN
                HangmanModeMenu
        ELSEIF user selects Tic Tac Toe THEN
                TicTacToeModeMenu
        ENDIF
END Mainprogram

BEGIN HangmanModeMenu
        IF user selects Hangman Single Player THEN
                HangmanSinglePlayer
        ELSEIF user selects Hangman Two Player THEN
                Prompt user for Player One
                Get Player One
                Prompt user for Player Two
                Get Player Two
                HangmanTwoPlayer
        ELSEIF user selects Game Settings THEN
                GameSettings
        ENDIF
END HangmanModeMenu

BEGIN HangmanSinglePlayer
        IF DefaultWordList = True THEN
                Word = DefaultWordList(RandomWord)
        ELSE
                Word = CustomWordList(RandomWord)
        ENDIF
        WordLength = Number of letters in Word
        WHILE i < WordLength
                WordtoGuess(i) = Next Letter in Word to guess
                i = i + 1
        ENDWHILE
        WordIndex = 0
        SubstringIndex = 0
        ReturnLetter = False
        Chances = 10
        Correct = False
        WHILE Chances > 0 OR Correct = False
                Get selected Letter from user
                FOR i = 0 to WordLength - 1
                        IF Word(i) = Letter THEN
                                WordIndex = i
                                WordIndex = WordIndex + 1
                                Mid(Wordtoguess, WordIndex, 1) = Letter
                                ReturnLetter = True
                        ENDIF
                NEXT i
                IF WordtoGuess = Word THEN
                        Correct = True
                        Display “Correct!”
                        Add WordtoGuess to CorrectWords
                        CorrectWords = CorrectWords + 1
                ELSEIF ReturnLetter = False THEN
                        Chances = Chances – 1
```

```
                ReturnLetter = False
            ELSE
                ReturnLetter = False
            ENDIF
        ENDWHILE
        IF Chances = 0 THEN
            Chances = 10
            Display “Incorrect!”
        ENDIF
END HangmanSinglePlayer

BEGIN HangmanTwoPlayer
        Round = 0
        SwitchedPlayer = False
        WHILE Round < 10 OR PlayerOneWinner = True OR PlayerTwoWinner = True
            IF SwitchedPlayer = False THEN
                Prompt Player Two for Word
                Get Player Two’s Word
                WordLength = Number of letters in Word
                WHILE i < WordLength
                    WordtoGuess(i) = Next Letter in Word to guess
                    i = i + 1
                ENDWHILE
                WordIndex = 0
                SubstringIndex = 0
                ReturnLetter = False
                Chances = 10
                Correct = False
                WHILE Chances > 0 OR Correct = False
                    Get selected Letter from user
                    FOR i = 0 to WordLength - 1
                        IF Word(i) = Letter THEN
                            WordIndex = i
                            WordIndex = WordIndex + 1
                            Mid(Wordtoguess, WordIndex, 1) = Letter
                            ReturnLetter = True
                        ENDIF
                    NEXT i
                    IF WordtoGuess = Word THEN
                        Correct = True
                        Display “Correct!”
                        Add WordtoGuess to CorrectWords
                        CorrectWords = CorrectWords + 1
                    ELSEIF ReturnLetter = False THEN
                        PlayerOneChances = PlayerOneChances – 1
                        ReturnLetter = False
                    ELSE
                        ReturnLetter = False
                    ENDIF
                ENDWHILE
                IF Chances = 0 THEN
                    Display “Incorrect!”
                    Give token to Player Two
                    PlayerTwoTokenNo = PlayerTwoTokenNo + 1
                    Display “Token!”
                ENDIF
                SwitchedPlayer = True
            ELSE
```

```
Prompt Player One for Word
Get Player One's Word
WordLength = Number of letters in Word
WHILE i < WordLength
        WordtoGuess(i) = Next Letter in Word to guess
        i = i + 1
ENDWHILE
WordIndex = 0
SubstringIndex = 0
ReturnLetter = False
Chances = 10
Correct = False
WHILE PlayerTwoChances > 0 OR Correct = False
        Get selected Letter from user
        FOR i = 0 to WordLength - 1
                IF Word(i) = Letter THEN
                        WordIndex = i
                        WordIndex = WordIndex + 1
                        Mid(Wordtoguess, WordIndex, 1) = Letter
                        ReturnLetter = True
                ENDIF
        NEXT i
        IF WordtoGuess = Word THEN
                Correct = True
                Display "Correct!"
                Add WordtoGuess to CorrectWords
                CorrectWords = CorrectWords + 1
        ELSEIF ReturnLetter = False THEN
                PlayerTwoChances = PlayerTwoChances – 1
                ReturnLetter = False
        ELSE
                ReturnLetter = False
        ENDIF
ENDWHILE
IF PlayerOneChances > PlayerTwoChances THEN
        Give token to Player Two
        Display "Token!"
ELSE
        Give token to Player Two
        Display "Token!"
ENDIF
IF Chances = 0 THEN
        PlayerOneChances = 10
        PlayerTwoChances = 10
        Display "Incorrect!" for Player Two
        Give token to Player One
        PlayerOneTokenNo = PlayerOneTokenNo + 1
        Display "Token!"
ENDIF
PlayerOneChances = 10
PlayerTwoChances = 10
IF PlayerOneTokenNo = 5 THEN
        PlayerOneWinner = True
ELSEIF PlayerTwoTokenNo = 5 THEN
        PlayerTwoWinner  = True
ENDIF
Round = Round + 1
SwitchedPlayer = False
```

```
            ENDIF
        ENDWHILE
        IF PlayerOneWinner = True THEN
            Display “Winner!” for Player One
            Display “Loser!” for Player Two
        ELSEIf PlayerTwoWinner = True THEN
            Display “Winner!” for Player Two
            Display “Loser!” for Player One
        ENDIF
END HangmanSinglePlayer

BEGIN TicTacToeModemenu
        IF user selects Tic Tac Toe Single Player THEN
            TicTacToeSinglePlayer
        ELSEIF user selects Tic Tac Toe Two Player THEN
            Prompt user for Player One
            Get Player One
            Prompt user for Player Two
            Get Player Two
            TicTacToeTwoPlayer
        ENDIF
END TicTacToeModemenu

BEGIN TicTacToeSinglePlayer
        Winner = False
        Round = 0
        WHILE Round =< 10 OR PlayerOneWinner = True OR ComputerPlayerWinner = True
            IF PlayerOne = True THEN
                Get Player One’s Chosen Marker Locations
                IF TopLeft = "X" And MiddleCenter = "X" And BottomRight = "X" Then
                    WonRound = True
                ElseIf TopLeft = "X" And TopCenter = "X" And TopRight = "X" Then
                    WonRound = True
                ElseIf MiddleLeft = "X" And MiddleCenter = "X" And MiddleRight = "X" Then
                    WonRound = True
                ElseIf BottomLeft = "X" And BottomCenter = "X" And BottomRight = "X" Then
                    WonRound = True
                ElseIf TopLeft = "X" And MiddleLeft = "X" And BottomLeft = "X" Then
                    WonRound = True
                ElseIf TopCenter = "X" And MiddleCenter = "X" And BottomCenter = "X" Then
                    WonRound = True
                ElseIf TopRight = "X" And MiddleRight = "X" And BottomRight = "X" Then
                    WonRound = True
                ElseIf TopRight = "X" And MiddleCenter = "X" And BottomLeft = "X" Then
                    WonRound = True
                End If
                IF WonRound = True THEN
                    Display “Token!” to Player One
                    PlayerOneTokenNo = PlayerOneTokenNo + 1
                    PlayerOne = True
                    Chosen = False
                    Round = Round + 1
                ELSE
                    PlayerOne = False
                ENDIF
                IF PlayerOneTokenNo = 5 THEN
                    PlayerOneWinner = True
                ENDIF
```

```
                ELSE
                        Chosen = False
                        TicTacToeComputerPlayer

                        IF TopLeft = "O" And MiddleCenter = "O" And BottomRight = "O" Then
                                WonRound = True
                        ElseIf TopLeft = "O" And TopCenter = "O" And TopRight = "O" Then
                                WonRound = True
                        ElseIf MiddleLeft = "O" And MiddleCenter = "O" And MiddleRight = "O" Then
                                WonRound = True
                        ElseIf BottomLeft = "O" And BottomCenter = "O" And BottomRight = "O" Then
                                WonRound = True
                        ElseIf TopLeft = "O" And MiddleLeft = "O" And BottomLeft = "O" Then
                             WonRound = True
                        ElseIf TopCenter = "O" And MiddleCenter = "O" And BottomCenter = "O" Then
                                WonRound = True
                        ElseIf TopRight = "O" And MiddleRight = "O" And BottomRight = "O" Then
                                WonRound = True
                        ElseIf TopRight = "O" And MiddleCenter = "O" And BottomLeft = "O" Then
                                WonRound = True
                        End If
                        IF WonRound = True THEN
                                Display “Token!” to Computer Player
                                ComputerPlayerTokenNo = ComputerPlayerTokenNo + 1
                                PlayerOne = True
                                Chosen = False
                                Round = Round + 1
                        ELSE
                                PlayerOne = True
                        ENDIF
                        IF ComputerPlayerTokenNo = 5 THEN
                                ComputerPlayerWinner = True
                        ENDIF
                ENDIF
        ENDWHILE
        IF PlayerOneWinner = True THEN
                Display “Winner!” to Player One
                Display “Loser!” to Computer Player
        ELSEIF ComputerPlayerWinner = True THEN
                Display “Winner!” to Computer Player
                Display “Loser!” to Player One
        ENDIF
END TicTacToeSinglePlayer


BEGIN TicTacToeComputerPlayer
        Index = 0
        WHILE Chosen = False
                Index = Random(0, AvailableNumbers(Length))
                ChosenNumber = AvailableNumbers(Index)
                AvailableNumbers(Index) = “”
                IF ChosenNumber = 1 THEN
                        TopLeft = “O”
                        Chosen = True
                ELSEIF ChosenNumber = 2 THEN
```

```
                TopCenter = “O”
                Chosen = True
            ELSEIF ChosenNumber = 3 THEN
                TopRight = “O”
                Chosen = True
            ELSEIF ChosenNumber = 4 THEN
                MiddleLeft = “O”
                Chosen = True
            ELSEIF ChosenNumber = 5 THEN
                MiddleCenter = “O”
                Chosen = True
            ELSEIF ChosenNumber = 6 THEN
                MiddleRight = “O”
                Chosen = True
            ELSEIF ChosenNumber = 7 THEN
                BottomLeft = “O”
                Chosen = True
            ELSEIF ChosenNumber = 8 THEN
                BottomCenter = “O”
                Chosen = True
            ELSEIF ChosenNumber = 9 THEN
                BottomRight = “O”
                Chosen = True
            ENDIF
        ENDWHILE
        PlayerOne = True
        Chosen = False
END TicTacToeComputerPlayer

BEGIN TicTacToeTwoPlayer
        Winner = False
        Round = 0
        SwitchPlayer = False
        WHILE Winner = False OR Round =< 10
            IF SwitchPlayer = False THEN
                Get Player One’s Chosen Marker Locations
                IF TopLeft = "X" And MiddleCenter = "X" And BottomRight = "X" Then
                    WonRound = True
                ElseIf TopLeft = "X" And TopCenter = "X" And TopRight = "X" Then
                    WonRound = True
                ElseIf MiddleLeft = "X" And MiddleCenter = "X" And MiddleRight = "X" Then
                    WonRound = True
                ElseIf BottomLeft = "X" And BottomCenter = "X" And BottomRight = "X" Then
                    WonRound = True
                ElseIf TopLeft = "X" And MiddleLeft = "X" And BottomLeft = "X" Then
                  WonRound = True
                ElseIf TopCenter = "X" And MiddleCenter = "X" And BottomCenter = "X" Then
                    WonRound = True
```

```
                    ElseIf TopRight = "X" And MiddleRight = "X" And BottomRight = "X" Then
                        WonRound = True
                    ElseIf TopRight = "X" And MiddleCenter = "X" And BottomLeft = "X" Then
                        WonRound = True
                    End If
                    IF WonRound = True THEN
                        Display “Token!” to Player One
                        PlayerOneTokenNo = PlayerOneTokenNo + 1
                        PlayerOne = True
                        Chosen = False
                        Round = Round + 1
                    ELSE
                        PlayerOne = False
                    ENDIF
                    IF PlayerOneTokenNo = 5 THEN
                        PlayerOneWinner = True
                    ENDIF
                ELSE
                    Get Player Two’s Chosen Marker Locations
                    IF TopLeft = "O" And MiddleCenter = "O" And BottomRight = "O" Then
                        WonRound = True
                    ElseIf TopLeft = "O" And TopCenter = "O" And TopRight = "O" Then
                        WonRound = True
                    ElseIf MiddleLeft = "O" And MiddleCenter = "O" And MiddleRight = "O" Then
                        WonRound = True
                    ElseIf BottomLeft = "O" And BottomCenter = "O" And BottomRight = "O" Then
                        WonRound = True
                    ElseIf TopLeft = "O" And MiddleLeft = "O" And BottomLeft = "O" Then
                        WonRound = True
                    ElseIf TopCenter = "O" And MiddleCenter = "O" And BottomCenter = "O" Then
                        WonRound = True
                    ElseIf TopRight = "O" And MiddleRight = "O" And BottomRight = "O" Then
                        WonRound = True
                    ElseIf TopRight = "O" And MiddleCenter = "O" And BottomLeft = "O" Then
                        WonRound = True
                    End If
                    IF WonRound = True THEN
                        Display “Token!” to Player Two
                        PlayerTwoTokenNo = PlayerTwoTokenNo + 1
                        PlayerOne = True
                        Chosen = False
                        Round = Round + 1
                    ELSE
                        SwitchedPlayer = False
                    ENDIF
                    IF PlayerTwoTokenNo = 5THEN
                        PlayerTwoWinner = True
                    ENDIF
                ENDIF
            ENDWHILE
            IF PlayerOneWinner = True THEN
                Dispay “Winner!” to Player One
                Display “Loser!” to Player Two
            ELSEIF PlayerTwoWinner = True THEN
                Display “Winner!” to Player Two
                Display “Loser!” to Player One
            ENDIF
END TicTacToeTwoPlayer
```

6. Test Data:

The test data displayed below will be used in the deskchecks. The deskchecks will accurately determine whether or not the output is expected throughout all the games starting with the Game Settings for hangman.

6.1. Hangman:

6.1.1. Game Settings:

Test Data	Purpose
• “abCdEfghiJ”	This data will be used to test if the correct letter casing is changed during the “Insert” process.
• “aBc!@#678901234567890”	This data is used to test if the signs and symbols are rejected by the program.
• “”	This will be used to test whether or not the dialog box shows up prompting the user to enter a word if they have not already done so.
A B C D E F G	The capitalized letters and whether or not they return with the first letter as the capital letter as well as no spaces between 1 letter.
Switching between radio buttons	Used to test if the proper buttons disable themselves and re-enable themselves upon switching between radio buttons.
Using the “Replace” button	Used to test if the word has to be entered in order for the word to be replaced. Also used to test if the word entered replaces the selected word.
Using the “Clean List” button	This is going to be used to test whether or not the custom word list fills up with 10 random words from the default word list.

6.1.2. Hangman – Single Player:

Test Data	Purpose
Selecting letters from Groupbox buttons	To test if the correct letters have been chosen from the groupbox
Using Default word List	This is going to be used to test whether the program identifies whether or not the “Default Word List” has been chosen.
Using Custom Word List	This is going to be used to test whether the program identifies whether or not the “Custom Word List” has been chosen.

6.1.3. Hangman – Two Player:

Test Data	Purpose
• ‘12345’ • ‘ ’ – (space) • ‘Letters’ • ‘Spa ce’ • ‘Numbers12345’ • ‘Morethantwentyonecharacters’ –(24 characters) • ‘’ – (nothing) • ‘!@#$%^’	When entering a new word for player 1, there are certain restrictions. Basically there can only be letters; no spaces, no special characters and no numbers. Also the word cannot exceed 20 characters and must be at least 1. The test data used is numbers only, spaces, letters only (under 20 characters) , letters only (over 20 characters), nothing at all, special characters, numbers and letters, letters and spaces.

6.2. Tic Tac Toe:

6.2.1. Tic Tac Toe – Single Player:

Test Data	Purpose
Computer Player's reaction time	The computer player should be able to pick a location on the grid in a split second
Marker disable picturebox after clicking	After choosing a spot to place the marker, the user should not be able to click the same box.
Player One token increment	Player One's tokens should increase if the user wins the round.
Computer Player token increment	Computer Player's tokens should increase if the computer wins the round.

6.2.2. Tic Tac Toe – Two Player:

Test Data	Purpose
Green marker	The marker should appear when its player one's turn and when the user hovers over the picture boxes.
Red Marker	A red marker should appear when its player Two's turn and player two hovers over the pictureboxes
Player One token increment	Player One's tokens should increase if the user wins the round.
Player Two token increment	Player Two's tokens should increase if the computer wins the round.
Marker disable picturebox after clicking	After choosing a spot to place the marker, the user should not be able to click the same box.

7. Deskchecks:

7.1. Game Settings:

Game Settings		
Input Data	**Output**	**Expected Output**
abCdEfghiJ		
	Abcdefghij	Abcdefghi
A B C D E F G		
	Abcdefg	Abcdefg
Input Data	**Output**	**Expected Output**
aBc!@#678901234567890		
	Dialog box ("The word you have entered is Invalid")	Dialog box("The word you have entered is Invalid")
""		
	Dialog box ("The word you have entered is Invalid")	Dialog box ("The word you have entered is Invalid")
Function/Input Data	**Output**	**Expected Output**
Click Default word list		
	All buttons + Textbox disabled	All buttons & Textbox disabled
Click Custom word list		
	All buttons + Textbox enabled	All buttons & Textbox enabled
Selected List word (to replace)		
ABCDEFGHI		
Using the "Replace" button		
	Abcdefghi	Abcdefghi
Using the "Clean List" button		
	Add 10 random words to the list from default list	10 random words entered into list box

7.2. Hangman – Single Player:

Hangman - Single Player		
Function/Input Data	**Output**	**Expected Output**
Selecting letters from groupbox buttons		
	B----	Until a chosen letter is shown in a random location of a hyphen
Using default word list		
	Aboard	At least one word from the DefaultWordList.txt document
Using custom word list		
	Terrathede	One word I entered into the Custom Word List

7.3. Hangman – Two Player:

Hangman - Two Player		
Input Data	**Output**	**Expected Output**
12345		
	Dialog box	Dialog box
Spa ce		
	Dialog box	Dialog box
Letters		
	"------"	"------"
Numbers12345		
	Dialog box	Dialog box
MoreThanTwentyCharacters		
	Dialog box	Dialog box
"-(Nothing)"		
	Dialog box	Dialog box

7.4. Tic Tac Toe – Single Player:

Tic Tac Toe - Single Player		
Function/Input Data	**Output**	**Expected Output**
Player one selects a marker position		
	Chosen = 4 (Random Number)	A random "Chosen Number" between 1-9
Clicking a picturebox		
	Picturebox location = disabled	Selected picturebox is disabled
Player one wins the round		
	Token increased by 1	Token = Token + 1 (Token = 1)
Computer Player wins the round		
	Token increased by 1	Token = Token + 1 (Token = 1)

7.5. Tic Tac Toe – Two Player:

Tic Tac Toe - Two Player		
Function/Input Data	**Output**	**Expected Output**
Player One's turn + Hover over boxes		
	Green marker shows in all boxes	Green marker shows in all boxes
Player Two's turn + Hover over boxes		
	Red marker shows in all boxes	Red marker shows in all boxes
Player one wins the round		
	Token increased by 1	Token = Token + 1 (Token = 1)
Player two wins the round		
	Token increased by 1	Token = Token + 1 (Token = 1)
Player One clicks a box		
Player Two clicks a box	Chosen box disables	Disables Chosen box
	Chosen box disables	Disables Chosen box

8. Source Code Printout:

8.1. Splash Screen:

```
Public Class frmSplashScreen 'frmSplashScreen form code
    Dim file(10000) As String
    Dim MaxCounter As Integer
    Dim Counter As Integer = 1
    Dim Filename(10000) As String
    Dim AppPath As String = Application.StartupPath
    Dim fileNames = My.Computer.FileSystem.GetFiles(AppPath,
FileIO.SearchOption.SearchAllSubDirectories)
    Private Sub ListFiles() 'ListFiles Subroutine code
        MaxCounter = fileNames.Count 'Sets the "MaxCounter" integer value to the amount of
items in the "fileNames" array
        For i = 0 To MaxCounter - 1 'Executes a set of commands until the integer variable
"i" is equal to the same value as the "MaxCounter" integer variable
            file(i) = fileNames(i) 'Writes the "fileNames" array's index of "i" to "file"
array's index of "i"
        Next
        CleanString() 'Calls the "CleanString" subroutine
    End Sub
    Private Sub CleanString() 'CleanString Subroutine code
        For i = 0 To MaxCounter - 1 'Executes a set of commands until the integer variable
"i" is equal to the same value as the "MaxCounter" integer variable
            file(i) = file(i).Substring(file(i).LastIndexOf("\") + 1) 'Writes the "fileNames"
array's index of "i" to "file" array's index of "i" without the full file directory (using
substring)
            Filename(i) = file(i) 'Writes the "fileNames" array's index of "i" to "file"
array's index of "i"
        Next
        FileTimer.Start() 'Starts the "FileTimer" timer tick event
    End Sub
    Private Sub LoadSettings() 'LoadSettings Subroutine code
        If My.Settings.MusicAndSounds = True Then 'Checks if application setting's
"MusicAndSounds" boolean variable is set to true
            frmMainMenu.btnMuteUnMute.Visible = True 'Removes the "btnMuteUnmute" button from
the "frmMainMenu" form
            frmHangmanModeMenu.btnMuteUnmute.Visible = True 'Shows the "btnMuteUnmute" button
on the "frmHangmanModeMenu" form
            frmTicTacToeModeMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeModeMenu" form
            frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeSinglePlayer" form
            frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = True
            'frmTicTacToeVersus.btnMuteUnMute.Visible = True
        End If
        If My.Settings.SoundsOnly = True Then 'Checks if application setting's "SoundsOnly"
boolean variable is set to true
            frmMainMenu.btnMuteUnMute.Visible = True 'Removes the "btnMuteUnmute" button from
the "frmMainMenu" form
            frmHangmanModeMenu.btnMuteUnmute.Visible = True 'Shows the "btnMuteUnmute" button
on the "frmHangmanModeMenu" form
            frmTicTacToeModeMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeModeMenu" form
            frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeSinglePlayer" form
            frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeTwoPlayer" form
            'frmTicTacToeVersus.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button on the "frmTicTacToeVersus" form
            frmSettings.OCXMusicPlayer.URL = Nothing 'Clears the Media Player's playlist
        End If
        If My.Settings.MusicOnly = True Then 'Checks if application setting's "MusicOnly"
boolean variable is set to true
```

```
            frmMainMenu.btnMuteUnMute.Visible = False 'Removes the "btnMuteUnmute" button
from the "frmMainMenu" form
            frmHangmanModeMenu.btnMuteUnmute.Visible = False 'Removes the "btnMuteUnmute"
button from the "frmHangmanModeMenu" form
            frmTicTacToeModeMenu.btnMuteUnMute.Visible = False 'Removes the "btnMuteUnmute"
button from the "frmTicTacToeModeMenu" form
            frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = False 'Removes the
"btnMuteUnmute" button from the "frmTicTacToeSinglePlayer" form
            frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = False 'Removes the "btnMuteUnmute"
button from the "frmTicTacToeTwoPlayer" form
            'frmTicTacToeVersus.btnMuteUnMute.Visible = False 'Removes the "btnMuteUnmute"
button from the "frmTicTacVersus" form
        End If
    End Sub

    Private Sub frmSplashScreen_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmSplashScreen Form Load code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        With picLoadingGif 'Executes a set of commands regarding the "picLoadingGif" picture
box
            .Image = My.Resources.Loader 'Changes the Image property of "picLoadingGif" to
"Loader" in resources
            .SizeMode = PictureBoxSizeMode.CenterImage 'Changes the SizeMode property of
"picLoadingGif" to "CenterImage"
        End With
        DefaultFormTimer.Start() 'Starts the "DefaultFormTimer" timer tick event
        LoadSettings() 'Calls the "LoadSettings" subroutine
        ListFiles() 'Calls the "ListFiles" subroutine
    End Sub

    'Timers
    Private Sub DefaultFormTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DefaultFormTimer.Tick 'DefaultFormTimer Timer Tick code
        For i = 0 To MaxCounter 'Executes a set of commands until the integer variable "i" is
equal to the same value as the "MaxCounter" integer variable
            file(i) = "" 'Clears "file" array's index "i"
            Filename(i) = "" 'Clears "Filename" array's index "i"
        Next
        DefaultFormTimer.Stop() 'Stops the "DefaultFormTimer" timer tick event
        If My.Settings.DefaultSettings = True Then 'Checks if "DefaultSettings" boolean
variable in application settings is set to True
            Me.Hide() 'Hides the form
            frmMainMenu.Show() 'Shows the "frmMainMenu" form
            Exit Sub 'Exits the subroutine
        End If
        If My.Settings.MainMenu = True Then 'Checks if "MainMenu" boolean variable in
application settings is set to True
            Me.Hide() 'Hides the form
            frmMainMenu.Show() 'Shows the "frmMainMenu" form
        End If
        If My.Settings.HangMan = True Then 'Checks if "HangMan" boolean variable in
application settings is set to True
            Me.Hide() 'Hides the form
            frmHangmanModeMenu.StartPosition = FormStartPosition.CenterScreen 'Sets the
"frmHangManModeMenu" start position property to "CenterScreen", starting the form in the
center of the screen
            frmHangmanModeMenu.Show() 'Shows the "frmHangManModeMenu" form
        End If
        If My.Settings.TicTacToe = True Then 'Checks if "TicTacToe" boolean variable in
application settings is set to True
            Me.Hide() 'Hides the form
```

```
            frmTicTacToeModeMenu.StartPosition = FormStartPosition.CenterScreen 'Sets the
"frmTicTacToeModeMenu" start position property to "CenterScreen", starting the form in the
center of the screen
            frmTicTacToeModeMenu.Show() 'Shows the "frmTicTacToeModeMenu" form
        End If
    End Sub
    Private Sub FileTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles FileTimer.Tick 'FileTimer Timer Tick code
        Counter = Counter + 1 'Increments the "Counter" integer variable
        If Counter = MaxCounter Then 'Checks if the integer variable "Counter" is the same
value as the integer variable "MaxCounter"
            lblFile.Text = "axvlc.dll" 'Changes the "lblFile" label text to "axvlc.dll"
            FileTimer.Stop() 'Stops the "FileTimer" timer tick event
        Else
            lblFile.Text = Filename(Counter) 'Writes the "Filename" array's index value
"Counter" to the "lblFile" label
        End If
    End Sub
End Class
```

8.2. Main Menu:

```
Imports System.IO 'Form makes reference to the "System.IO" namespace
Public Class frmMainMenu 'frmMainMenu Form Class
    Public PlayerFlag As Boolean = False
    Public FirstPlayer As Boolean = False
    Public SecondPlayer As Boolean = False
    Public PlayerOne As String
    Public PlayerTwo As String
    Public WithEvents player As New System.Media.SoundPlayer
    Dim AppPath As String = Application.StartupPath
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Dim CaptionCounter As Integer
    Dim FullMusicDirectory As String = Application.StartupPath & "\Music\"
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
```

```
        End Select
    End Sub
    Private Sub frmMainMenu_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load 'frmMainMenu Form Load code
        If My.Settings.DisableCaptions = False Then 'Checks if the application setting's
boolean variable "DisableCaptions" is set to false
            MouseMoveTimer.Start() 'Starts the "MouseMoveTimer" timer
        End If
        If My.Settings.Mute = False Then 'Checks if the user wants the sounds to be muted on
all form
            lblMute.Text = "Mute Sounds" 'Sets the "lblMute" label to "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        Else
            lblMute.Text = "UnMute Sounds" 'Sets the "lblMute" label to "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        End If
    End Sub
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown 'btnMinimize Button
MouseDown code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed 'Changes the
background image of the "btnMinimize" button when the mouse is down
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter 'btnMinimize Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_MouseScrollover 'Sets the soundplayer to the
"MouseScrollover" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted 'Changes the
background image of the "btnMinimize" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave 'btnMinimize Button MouseLeave code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp 'btnMinimize Button MouseUp
code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormMinimizing 'Sets the soundplayer to the
"FormMinimizing" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized 'Minimizes the form by changing the form's
"FormWindowState" properties to "Minimized"
    End Sub
    'Exit
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed  'Changes the background
image of the "btnClose" button when the mouse is down
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
```

```
            player.Stream = My.Resources.sound_MouseScrollover 'Sets the soundplayer to the
"MouseScrollover" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
background image of the close button to highlighted when the curser enters the button
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button MouseLeave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer to the
"FormClosing" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed 'Changes the background
image of the "btnInfo" button when the mouse is down
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        lblInfo.Visible = True 'Shows the "lblInfo" label when mouse enters the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_MouseScrollover 'Sets the soundplayer to the
"MouseScrollover" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted 'Changes the
background image of the info button to highlighted when the curser enters the button
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False 'Hides the "lblInfo" label when mouse leaves the picture box
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer to the
"FormClosing" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
```

```
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    Private Sub btnSettings_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseDown 'btnSettings Button
MouseDown code
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Pushed  'Changes the
background image of the "btnSettings" button when the mouse is down
    End Sub
    Private Sub btnSettings_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseEnter 'btnSettings Button MouseEnter code
        lblSettings.Visible = True 'Shows the "lblSettings" label when mouse enters the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Highlighted 'Changes the
background image of the settings button to highlighted when the curser enters the button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_MouseScrollover 'Sets the soundplayer to the
"MouseScrollover" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnSettings_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseLeave 'btnSettings Button MouseLeave code
        lblSettings.Visible = False 'Hides the "lblSettings" label when mouse leaves the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnSettings_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseUp 'btnSettings Button MouseUp
code
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer to the
"FormSelect" WAV file in the resources
            player.Play() 'Plays the sound file
        End If
        frmSettings.Show() 'Shows the "frmSettings" form
    End Sub
    Private Sub btnMuteUnMute_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseUp 'btnMuteUnMute Button
MouseUp code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            My.Settings.Mute = True 'Sets the application setting's boolean variable "Mute"
to true
            lblMute.Text = "UnMute Sounds" 'Sets the "lblMute" label's text value to "UnMute
Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Sets the
"btnMuteUnmute" picturebox's background image to "UnMute_Button" in resources
        Else
            My.Settings.Mute = False 'Sets the "Mute" variable in application settings to
false
            lblMute.Text = "Mute Sounds" 'Sets the "lblMute" label's text value to "Mute
Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Sets the
"btnMuteUnmute" picturebox's background image to "Mute_Button" in resources
        End If
        My.Settings.Save() 'Saves the application settings
    End Sub
    Private Sub btnMuteUnMute_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseDown 'btnMuteUnMute Button
MouseDown code
```

```
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Pushed 'Sets the
"btnMuteUnmute" picturebox's background image to "UnMute_Button_Pushed" in resources
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Pushed 'Sets the
"btnMuteUnmute" picturebox's background image to "Mute_Button_Pushed" in resources
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseEnter 'btnMuteUnMute Button MouseEnter code
        lblMute.Visible = True 'Shows the "lblMute" label
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_MouseScrollover 'Sets the soundplayer to the
"MouseScrollover" WAV file in the resources
            player.Play() 'Plays the sound file
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Highlighted 'Changes the
background image of the mute button to highlighted when the curser enters the button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Highlighted 'Changes
the background image of the mute button to highlighted when the curser enters the button
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseLeave 'btnMuteUnMute Button MouseLeave code
        lblMute.Visible = False 'Shows the "lblMute" label
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Changes the background
image of the mute button to original image when the curser leaves the button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Changes the
background image of the mute button to original image when the curser leaves the button
        End If
    End Sub
    'Hangman
    Private Sub btnHangMan_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnHangMan.Click 'btnHangMan Button Click code
        frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormOpening 'Sets the player's stream to the
"sound_FormOpening" WAV file in resources
            player.Play() 'Plays the sound file
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Changes the
background image of the mute button to original image when the curser leaves the button
            frmHangmanModeMenu.btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button
'Sets the frmHangmanModeMenu's "btnMuteUnMute" picturebox's background image to
"UnMute_Button" in resources
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Changes the background
image of the mute button to original image when the curser leaves the button
            frmHangmanModeMenu.btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button 'Sets
the frmHangmanModeMenu's "btnMuteUnMute" picturebox's background image to "Mute_Button" in
resources
        End If
        Me.Hide() 'Hides the current form
        frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
    End Sub
    Private Sub btnTicTacToe_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnTicTacToe.Click 'btnTicTacToe Button Click code
        frmTicTacToeModeMenu.Location = New Point(Me.Location.X + 119, Me.Location.Y) 'Sets
the "frmTicTacToeModeMenu" form's location to the current form's location
```

```
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            player.Stream = My.Resources.sound_FormOpening 'Sets the player's stream to the
"sound_FormOpening" WAV file in resources
            player.Play() 'Plays the sound file
            frmTicTacToeModeMenu.lblMute.Text = "Mute Sounds" 'Sets the "lblMute" label's
text value to "UnMute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
'Changes the background image of the mute button to original image when the curser leaves the
button
        Else
            frmTicTacToeModeMenu.lblMute.Text = "UnMute Sounds" 'Sets the "lblMute" label's
text value to "Mute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
'Changes the background image of the mute button to original image when the curser leaves the
button
        End If
        Me.Hide() 'Hides the form
        frmTicTacToeModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
    End Sub
    'VLCMediaPlayer
    Private Sub btnVLCMediaPlayer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnVLCMediaPlayer.Click 'btnVLCMediaPlayer Button Click code
        Dim ProcessDirectory As String = AppPath & "\VLCMediaPlayer" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"VLCMediaPlayer.exe"
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the VLCMediaPlayer.exe
(executable file)
    End Sub

    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblMute.Visible = False 'Hides the "lblMute" label
                lblSettings.Visible = False 'Hides the "lblSettings" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblSettings.Visible = True 'Shows the "lblSettings" label
            lblInfo.Visible = True 'Shows the "lblInfo" label
            If btnMuteUnMute.Visible = True Then 'Checks if the "btnMuteUnMute" picturebox is
showing
                lblMute.Visible = True 'Shows the "lblMute" label
            End If
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.3. Hangman Mode Menu:

```
Public Class frmHangmanModeMenu 'frmHangmanModeMenu form code
    Dim CaptionCounter As Integer
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Dim AppPath As String = Application.StartupPath
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Sub GetPlayerNames() 'GetPlayerNames Private Sub code
        If frmMainMenu.PlayerFlag = True Then
            Dim MessageBoxResult As String
            MessageBoxResult = MsgBox("Player One: " & frmMainMenu.PlayerOne & vbCrLf &
"Player Two: " & frmMainMenu.PlayerTwo & vbCrLf & vbCrLf & "Would you like to use the same
player names?", vbYesNo + vbInformation, "Player Names") 'Checks if there were names already
entered into the application before
            If MessageBoxResult = vbNo Then
                frmMainMenu.PlayerFlag = False 'Sets PlayerFlag to False
                frmMainMenu.FirstPlayer = False 'Sets FirstPlayer to False
                frmMainMenu.SecondPlayer = False 'Sets SecondPlayer to False
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then 'Checks if FirstPlayer is set to false
            Dim FirstPlayerInputBoxResult As String
            FirstPlayerInputBoxResult = InputBox("Player One, please enter your name:",
"Player One", "Enter Your Name Here") 'Prompts the user for Player One's Name
            If FirstPlayerInputBoxResult = "Enter Your Name Here" Then
                MsgBox("You have not entered a name. Please enter your name to continue",
vbInformation, "Invalid Name")  'Tells the user if the name entered is invalid
                Exit Sub
            End If
            If FirstPlayerInputBoxResult = "" Then 'Checks if the input box is empty
                Exit Sub
            Else
```

```
                frmMainMenu.PlayerOne = FirstPlayerInputBoxResult
                frmMainMenu.FirstPlayer = True
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then
            Exit Sub
        End If
        If frmMainMenu.SecondPlayer = False Then
            Dim SecondPlayerInputBoxResult As String
            SecondPlayerInputBoxResult = InputBox("Player Two, please enter your name:",
"Player Two", "Enter Your Name Here") 'Prompts the user for Player Two's Name
            If SecondPlayerInputBoxResult = "Enter Your Name Here" Then
                MsgBox("You have not entered a name. Please enter your name to continue",
vbInformation, "Invalid Name") 'Tells the user if the name entered is invalid
                Exit Sub
            End If
            If SecondPlayerInputBoxResult = "" Then 'Checks if the input box is empty
                Exit Sub
            Else
                frmMainMenu.PlayerTwo = SecondPlayerInputBoxResult
                frmMainMenu.SecondPlayer = True
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then
            Exit Sub
        End If
        frmMainMenu.PlayerFlag = True
    End Sub
    Private Sub btnMuteUnMute_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnmute.MouseUp 'btnMuteUnmute MouseUp
code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            My.Settings.Mute = True 'Sets the "Mute" variable in application settings to True
            lblMute.Text = "UnMute Sounds"
            btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button
        Else
            My.Settings.Mute = False 'Sets the "Mute" variable in application settings to
False
            lblMute.Text = "Mute Sounds"
            btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button
        End If
        My.Settings.Save()
    End Sub
    Private Sub btnMuteUnMute_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnmute.MouseDown 'btnMuteUnmute MouseDown
code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button_Pushed
        Else
            btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button_Pushed
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnmute.MouseEnter 'btnMuteUnmute MouseEnter code
        lblMute.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button_Highlighted 'Changes the
background image of the mute button to highlighted when the curser enters the button
        Else
            btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button_Highlighted 'Changes
the background image of the mute button to highlighted when the curser enters the button
        End If
    End Sub
```

```
    Private Sub btnMuteUnMute_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnmute.MouseLeave 'btnMuteUnmute MouseLeave code
        lblMute.Visible = False
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button 'Changes the background
image of the mute button to highlighted when the curser enters the button
        Else
            btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button 'Changes the
background image of the mute button to highlighted when the curser enters the button
        End If
    End Sub
    'Minimize
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown 'btnMuteUnmute MouseDown
code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed 'Changes the
"btnMinimize" picturebox's background to another image in resources
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter 'btnMinimize MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted 'Changes the
"btnMinimize" picturebox's background to another image in resources
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave 'btnMinimize MouseLeave code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
picturebox's background to another image in resources
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp 'btnMinimize MouseUp code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
picturebox's background to another image in resources
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed 'Changes the "btnClose"
picturebox's background to another image in resources
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
"btnClose" picturebox's background to another image in resources
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
```

```
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        lblInfo.Visible = True 'Shows the "lblInfo" label
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False 'Hides the "lblInfo" label
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo"
picturebox's background to another image in resources
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
```

```
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        lblMainMenu.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblMainMenu.Visible = False
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        frmMainMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmMainMenu" form's location to the current form's location
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "sound_FormReturning" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            frmMainMenu.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Changes the
"btnMuteUnMute" picturebox's background to another image in resources
        Else
            frmMainMenu.btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Changes
the "btnMuteUnMute" picturebox's background to another image in resources
        End If
        frmMainMenu.Show() 'Shows the "frmMainMenu" form
        Me.Dispose() 'Closes the current form
    End Sub
    Private Sub btnSettings_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnGameSettings.MouseDown 'btnGameSettings
Button MouseDown code
        btnGameSettings.BackgroundImage = My.Resources.GameSettings_Button_Pushed 'Changes
the "btnGameSettings" picturebox's background to another image in resources
    End Sub
    Private Sub btnSettings_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnGameSettings.MouseEnter 'btnGameSettings Button MouseEnter code
        lblGameSettings.Visible = True
        btnGameSettings.BackgroundImage = My.Resources.GameSettings_Button_Highlighted
'Changes the "btnGameSettings" picturebox's background to another image in resources
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnSettings_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnGameSettings.MouseLeave 'btnGameSettings Button MouseLeave code
        lblGameSettings.Visible = False
        btnGameSettings.BackgroundImage = My.Resources.GameSettings_Button 'Changes the
"btnGameSettings" picturebox's background to another image in resources
    End Sub
    Private Sub btnSettings_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnGameSettings.MouseUp 'btnGameSettings Button
MouseUp code
```

```
        btnGameSettings.BackgroundImage = My.Resources.GameSettings_Button 'Changes the
"btnGameSettings" picturebox's background to another image in resources
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        frmGameSettings.Show() 'Shows the "frmGameSettings" form
    End Sub
    'SinglePlayer
    Private Sub btnSinglePlayer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelectSinglePlayer.Click 'btnSelectSinglePlayer Button Click
code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.Hide()
        frmHangmanSinglePlayer.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanSinglePlayer" form's location to the current form's location
        frmHangmanSinglePlayer.Show() 'Shows the "frmHangmanSinglePlayer" form
    End Sub
    'TwoPlayer
    Private Sub btnTwoPlayer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelectTwoPlayer.Click 'btnSelectTwoPlayer Button Click code
        GetPlayerNames()
        If frmMainMenu.FirstPlayer = False Or frmMainMenu.SecondPlayer = False Then
            Exit Sub
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.Hide()
        frmHangmanTwoPlayer.lblPlayerOne.Text = frmMainMenu.PlayerOne
        frmHangmanTwoPlayer.lblPlayerTwo.Text = frmMainMenu.PlayerTwo
        frmHangmanTwoPlayer.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
location of the "frmHangmanTwoPlayer" form to the current form's location
        frmHangmanTwoPlayer.Show() 'Shows the "frmHangmanTwoPlayer" form
    End Sub
    'Versus
    Private Sub btnSelectVersus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelectVersus.Click 'btnSelectVersus button click event code
        GetPlayerNames() 'Calls the "GetPlayerNames" subroutine
        If frmMainMenu.FirstPlayer = False Or frmMainMenu.SecondPlayer = False Then 'Checks
if both players have entered their names
            Exit Sub 'Exits the subroutine
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.Hide() 'Hides the current form
        frmHangmanVersus.lblPlayerOne.Text = frmMainMenu.PlayerOne
        frmHangmanVersus.lblPlayerTwo.Text = frmMainMenu.PlayerTwo
        frmHangmanVersus.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
location of the "frmHangmanVersus" form to the current form's location
        frmHangmanVersus.Show() 'Shows the "frmHangmanVersus" form
    End Sub
```

```
    'Timers
    Private Sub SoundTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles SoundTimer.Tick 'SoundTimer Tick code
        frmMainMenu.btnClose.Enabled = True
        frmMainMenu.btnMaximize.Enabled = True
        frmMainMenu.btnMinimize.Enabled = True
        SoundTimer.Stop()
    End Sub

    Private Sub frmHangmanModeMenu_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmHangmanModeMenu form Load code
        If My.Settings.DisableCaptions = False Then
            MouseMoveTimer.Start()
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            lblMute.Text = "Mute Sounds"
            btnMuteUnmute.BackgroundImage = My.Resources.Mute_Button
        Else
            lblMute.Text = "UnMute Sounds"
            btnMuteUnmute.BackgroundImage = My.Resources.UnMute_Button
        End If
    End Sub
   Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblMainMenu.Visible = False 'Hides the "lblMainMenu" label
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblMute.Visible = False 'Hides the "lblMute" label
                lblGameSettings.Visible = False 'Hides the "lblGameSettings" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblMainMenu.Visible = True  'Shows the "lblMainMenu" label
            lblInfo.Visible = True  'Shows the "lblInfo" label
            lblGameSettings.Visible = True  'Shows the "lblGameSettings" label
            If btnMuteUnmute.Visible = True Then 'Checks if the "btnMuteUnMute" picturebox is
showing
                lblMute.Visible = True  'Shows the "lblMute" label
            End If
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick  'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.4. Hangman – Single Player:

```
Public Class frmHangmanSinglePlayer 'frmHangmanSinglePlayer form code
    Dim word As String = ""
    Dim Reader As IO.StreamReader
    Dim EndOfList As Integer
    Dim CurrentLetter As String
    Dim OpacityCounter As Integer
    Dim CaptionCounter As Integer
    Dim Correct As Boolean = False
    Dim ExitFlag As Boolean = True
    Dim CorrectWords As Integer = 0
    Dim InCorrectWords As Integer = 0
    Dim CurrentMousePosition As String
    Dim CountDownCounter As Integer = 6
    Dim PlayerOneChances As Integer = 12
    Dim appPath As String = Application.StartupPath()
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub

    Function GuessedLetter(ByVal letter As String)
        Dim ReturnLetter As Boolean = False
        Dim x As Integer
        For i = 0 To word.Length - 1
            If word.Substring(i, 1) = letter.ToLower Then
                x = i
                x = x + 1
                Mid(lblGuess.Text, x, 1) = letter
                ReturnLetter = True
            End If
        Next i
        Return ReturnLetter
    End Function

    Public Function PlayerOne_GetImage(ByVal bounds As Size) As Image
        'Draw each piece of the man,
        'as applicable
```

```
        Dim img As New Bitmap(bounds.Width, bounds.Height)
        If PlayerOneChances < 0 Then
            img = My.Resources.HangMan_CurrentPlayer_10
        End If
        If PlayerOneChances >= 0 Then
            img = My.Resources.HangMan_CurrentPlayer_10
        End If
        If PlayerOneChances > 1 Then
            img = My.Resources.HangMan_CurrentPlayer_9
        End If
        If PlayerOneChances > 2 Then
            img = My.Resources.HangMan_CurrentPlayer_8
        End If
        If PlayerOneChances > 3 Then
            img = My.Resources.HangMan_CurrentPlayer_7
        End If
        If PlayerOneChances > 4 Then
            img = My.Resources.HangMan_CurrentPlayer_6
        End If
        If PlayerOneChances > 5 Then
            img = My.Resources.HangMan_CurrentPlayer_5
        End If
        If PlayerOneChances > 6 Then
            img = My.Resources.HangMan_CurrentPlayer_4
        End If
        If PlayerOneChances > 7 Then
            img = My.Resources.HangMan_CurrentPlayer_3
        End If
        If PlayerOneChances > 8 Then
            img = My.Resources.HangMan_CurrentPlayer_2
        End If
        If PlayerOneChances > 9 Then
            img = My.Resources.HangMan_CurrentPlayer_1
        End If
        If PlayerOneChances > 10 Then
            img = My.Resources.HangMan_CurrentPlayer_0
        End If
        If PlayerOneChances > 11 Then
            img = My.Resources.HangMan_CurrentPlayer
        End If
        'Release the Graphics object
        'Return the image
        Return img

    End Function
    Public Sub Guess()
        Try
            lblGuess.Focus()
            PicPlayerOne.BackgroundImage = Nothing
            If GuessedLetter(CurrentLetter) = False Then
                PlayerOneChances = PlayerOneChances - 1
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
            Else
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
            End If
            If word.ToUpper = lblGuess.Text.ToUpper Then
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    lblGuess.ForeColor = Color.Lime 'Sets the "lblGuess" label's forecolor to
Lime
                    lblGuess.Text = word.ToUpper 'Converts all characters in the "lblGuess"
label to UPPERCASE
                    WinnerSoundTimer.Start() 'Starts the "WinnerSoundTimer" timer's tick
event
                End If
                PicPlayerOne.Image = My.Resources.HangMan_Correct
```

```
                DisableLetters()
                lstCorrect.Items.Add(word.ToUpper)
                CorrectWords = CorrectWords + 1
                bxCorrectWords.Text = "Correct Words: " & CorrectWords
                GameTimer.Start()
            End If
            If PlayerOneChances <= 1 Then
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    lblGuess.ForeColor = Color.Maroon 'Sets the "lblGuess" label's forecolor
to Maroon
                    lblGuess.Text = word.ToUpper 'Converts all characters in the "lblGuess"
label to UPPERCASE
                    picWrongArrow.Visible = True 'Shows the "picWrongArrow" picture box
                    LoserSoundTimer.Start() 'Starts the "LoserSoundTimer" timer's tick event
                End If
                PicPlayerOne.Image = My.Resources.HangMan_Wrong
                PicPlayerOne.BackgroundImage = My.Resources.HangMan_CurrentPlayer_10
                DisableLetters()
                lstIncorrect.Items.Add(word.ToUpper)
                InCorrectWords = InCorrectWords + 1
                bxIncorrectWords.Text = "InCorrect Words: " & InCorrectWords
                GameTimer.Start()
            End If
        Catch ex As Exception
            MsgBox("Error: Please report this to Emmanuel Vaccaro")
        End Try
    End Sub
    Private Sub EnableLetters() 'EnableLetters Private Sub code
        'Changes all of the button's backcolor to Silver
        btnA.BackColor = Color.Silver
        btnB.BackColor = Color.Silver
        btnC.BackColor = Color.Silver
        btnD.BackColor = Color.Silver
        btnE.BackColor = Color.Silver
        btnF.BackColor = Color.Silver
        btnG.BackColor = Color.Silver
        btnH.BackColor = Color.Silver
        btnI.BackColor = Color.Silver
        btnJ.BackColor = Color.Silver
        btnK.BackColor = Color.Silver
        btnL.BackColor = Color.Silver
        btnM.BackColor = Color.Silver
        btnN.BackColor = Color.Silver
        btnO.BackColor = Color.Silver
        btnP.BackColor = Color.Silver
        btnQ.BackColor = Color.Silver
        btnR.BackColor = Color.Silver
        btnS.BackColor = Color.Silver
        btnT.BackColor = Color.Silver
        btnU.BackColor = Color.Silver
        btnV.BackColor = Color.Silver
        btnW.BackColor = Color.Silver
        btnX.BackColor = Color.Silver
        btnY.BackColor = Color.Silver
        btnZ.BackColor = Color.Silver
        'Re-enables all of the buttons
        btnA.Enabled = True
        btnB.Enabled = True
        btnC.Enabled = True
        btnD.Enabled = True
        btnE.Enabled = True
        btnF.Enabled = True
        btnG.Enabled = True
        btnH.Enabled = True
        btnI.Enabled = True
```

```
        btnJ.Enabled = True
        btnK.Enabled = True
        btnL.Enabled = True
        btnM.Enabled = True
        btnN.Enabled = True
        btnO.Enabled = True
        btnP.Enabled = True
        btnQ.Enabled = True
        btnR.Enabled = True
        btnS.Enabled = True
        btnT.Enabled = True
        btnU.Enabled = True
        btnV.Enabled = True
        btnW.Enabled = True
        btnX.Enabled = True
        btnY.Enabled = True
        btnZ.Enabled = True
    End Sub
    Private Sub DisableLetters()
        'Changes all of the button's backcolor to DimGray
        btnA.BackColor = Color.DimGray
        btnB.BackColor = Color.DimGray
        btnC.BackColor = Color.DimGray
        btnD.BackColor = Color.DimGray
        btnE.BackColor = Color.DimGray
        btnF.BackColor = Color.DimGray
        btnG.BackColor = Color.DimGray
        btnH.BackColor = Color.DimGray
        btnI.BackColor = Color.DimGray
        btnJ.BackColor = Color.DimGray
        btnK.BackColor = Color.DimGray
        btnL.BackColor = Color.DimGray
        btnM.BackColor = Color.DimGray
        btnN.BackColor = Color.DimGray
        btnO.BackColor = Color.DimGray
        btnP.BackColor = Color.DimGray
        btnQ.BackColor = Color.DimGray
        btnR.BackColor = Color.DimGray
        btnS.BackColor = Color.DimGray
        btnT.BackColor = Color.DimGray
        btnU.BackColor = Color.DimGray
        btnV.BackColor = Color.DimGray
        btnW.BackColor = Color.DimGray
        btnX.BackColor = Color.DimGray
        btnY.BackColor = Color.DimGray
        btnZ.BackColor = Color.DimGray
        'Disables all of the buttons
        btnA.Enabled = False
        btnB.Enabled = False
        btnC.Enabled = False
        btnD.Enabled = False
        btnE.Enabled = False
        btnF.Enabled = False
        btnG.Enabled = False
        btnH.Enabled = False
        btnI.Enabled = False
        btnJ.Enabled = False
        btnK.Enabled = False
        btnL.Enabled = False
        btnM.Enabled = False
        btnN.Enabled = False
        btnO.Enabled = False
        btnP.Enabled = False
        btnQ.Enabled = False
        btnR.Enabled = False
        btnS.Enabled = False
```

```
        btnT.Enabled = False
        btnU.Enabled = False
        btnV.Enabled = False
        btnW.Enabled = False
        btnX.Enabled = False
        btnY.Enabled = False
        btnZ.Enabled = False
    End Sub
    Public Sub GameStart()
        ExitFlag = False
        Dim rand As New Random
        picWrongArrow.Visible = False
        lblGuess.ForeColor = Color.Black
        EnableLetters()
        EndOfList = lstWords.Items.Count
        PicPlayerOne.Image = Nothing
        PicPlayerOne.BackgroundImage = My.Resources.HangMan_CurrentPlayer
        lblGuess.Text = ""
        word = lstWords.Items.Item(rand.Next(0, EndOfList))
        word = word.ToLower
        For i = 0 To word.Length - 1
            If word(i) = " " Then
                lblGuess.Text &= " "
            Else
                lblGuess.Text &= "-"
            End If
        Next i
        PlayerOneChances = 12
        PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
        CurrentLetter = ""
    End Sub
    Private Sub frmHangmanSinglePlayer_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If My.Settings.DisableCaptions = False Then
            MouseMoveTimer.Start()
        End If
        DisableLetters()
        If My.Settings.HangmanDefaultGameList = False Then
            Dim rand As New Random
            Reader = New IO.StreamReader(appPath & "\CustomWordList.txt")
            While (Reader.Peek() > -1)
                lstWords.Items.Add(Reader.ReadLine)
            End While
            Reader.Close()
        Else
            Dim rand As New Random
            Reader = New IO.StreamReader(appPath & "\DefaultWordList.txt")
            While (Reader.Peek() > -1)
                lstWords.Items.Add(Reader.ReadLine)
            End While
            Reader.Close()
        End If
        EndOfList = lstWords.Items.Count 'Changes the "EndOfList" integer to the ammount of
words in the "lstWords" listbox
    End Sub
    Private Sub frmHangmanSinglePlayer_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles MyBase.KeyPress
        If Asc(e.KeyChar) = 13 And btnStart.Visible = True Then
            btnStart_MouseUp(Nothing, Nothing)
        End If
        If (Asc(e.KeyChar) = 65 Or Asc(e.KeyChar) = 97) And btnA.Enabled = True Then 'Checks
if the button pressed on the keyboard is either "A" or "a" and checks if the button is
enabled
            btnA_Click(Nothing, Nothing) 'Calls the "btnA_Click" event
```

```
        ElseIf (Asc(e.KeyChar) = 66 Or Asc(e.KeyChar) = 98) And btnB.Enabled = True Then
'Checks if the button pressed on the keyboard is either "B" or "b" and checks if the button
is enabled
            btnB_Click(Nothing, Nothing) 'Calls the "btnB_Click" event
        ElseIf (Asc(e.KeyChar) = 67 Or Asc(e.KeyChar) = 99) And btnC.Enabled = True Then
'Checks if the button pressed on the keyboard is either "C" or "c" and checks if the button
is enabled
            btnC_Click(Nothing, Nothing) 'Calls the "btnC_Click" event
        ElseIf (Asc(e.KeyChar) = 68 Or Asc(e.KeyChar) = 100) And btnD.Enabled = True Then
'Checks if the button pressed on the keyboard is either "D" or "d" and checks if the button
is enabled
            btnD_Click(Nothing, Nothing) 'Calls the "btnD_Click" event
        ElseIf (Asc(e.KeyChar) = 69 Or Asc(e.KeyChar) = 101) And btnE.Enabled = True Then
'Checks if the button pressed on the keyboard is either "E" or "e" and checks if the button
is enabled
            btnE_Click(Nothing, Nothing) 'Calls the "btnE_Click" event
        ElseIf (Asc(e.KeyChar) = 70 Or Asc(e.KeyChar) = 102) And btnF.Enabled = True Then
'Checks if the button pressed on the keyboard is either "F" or "f" and checks if the button
is enabled
            btnF_Click(Nothing, Nothing) 'Calls the "btnF_Click" event
        ElseIf (Asc(e.KeyChar) = 71 Or Asc(e.KeyChar) = 103) And btnG.Enabled = True Then
'Checks if the button pressed on the keyboard is either "G" or "g" and checks if the button
is enabled
            btnG_Click(Nothing, Nothing) 'Calls the "btnG_Click" event
        ElseIf (Asc(e.KeyChar) = 72 Or Asc(e.KeyChar) = 104) And btnH.Enabled = True Then
'Checks if the button pressed on the keyboard is either "H" or "h" and checks if the button
is enabled
            btnH_Click(Nothing, Nothing) 'Calls the "btnH_Click" event
        ElseIf (Asc(e.KeyChar) = 73 Or Asc(e.KeyChar) = 105) And btnI.Enabled = True Then
'Checks if the button pressed on the keyboard is either "I" or "i" and checks if the button
is enabled
            btnI_Click(Nothing, Nothing) 'Calls the "btnI_Click" event
        ElseIf (Asc(e.KeyChar) = 74 Or Asc(e.KeyChar) = 106) And btnJ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "J" or "j" and checks if the button
is enabled
            btnJ_Click(Nothing, Nothing) 'Calls the "btnJ_Click" event
        ElseIf (Asc(e.KeyChar) = 75 Or Asc(e.KeyChar) = 107) And btnK.Enabled = True Then
'Checks if the button pressed on the keyboard is either "K" or "k" and checks if the button
is enabled
            btnK_Click(Nothing, Nothing) 'Calls the "btnK_Click" event
        ElseIf (Asc(e.KeyChar) = 76 Or Asc(e.KeyChar) = 108) And btnL.Enabled = True Then
'Checks if the button pressed on the keyboard is either "L" or "l" and checks if the button
is enabled
            btnL_Click(Nothing, Nothing) 'Calls the "btnL_Click" event
        ElseIf (Asc(e.KeyChar) = 77 Or Asc(e.KeyChar) = 109) And btnM.Enabled = True Then
'Checks if the button pressed on the keyboard is either "M" or "m" and checks if the button
is enabled
            btnM_Click(Nothing, Nothing) 'Calls the "btnM_Click" event
        ElseIf (Asc(e.KeyChar) = 78 Or Asc(e.KeyChar) = 110) And btnN.Enabled = True Then
'Checks if the button pressed on the keyboard is either "N" or "n" and checks if the button
is enabled
            btnN_Click(Nothing, Nothing) 'Calls the "btnN_Click" event
        ElseIf (Asc(e.KeyChar) = 79 Or Asc(e.KeyChar) = 111) And btnO.Enabled = True Then
'Checks if the button pressed on the keyboard is either "O" or "o" and checks if the button
is enabled
            btnO_Click(Nothing, Nothing) 'Calls the "btnO_Click" event
        ElseIf (Asc(e.KeyChar) = 80 Or Asc(e.KeyChar) = 112) And btnP.Enabled = True Then
'Checks if the button pressed on the keyboard is either "P" or "p" and checks if the button
is enabled
            btnP_Click(Nothing, Nothing) 'Calls the "btnP_Click" event
        ElseIf (Asc(e.KeyChar) = 81 Or Asc(e.KeyChar) = 113) And btnQ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Q" or "q" and checks if the button
is enabled
            btnQ_Click(Nothing, Nothing) 'Calls the "btnQ_Click" event
```

```
        ElseIf (Asc(e.KeyChar) = 82 Or Asc(e.KeyChar) = 114) And btnR.Enabled = True Then
'Checks if the button pressed on the keyboard is either "R" or "r" and checks if the button
is enabled
            btnR_Click(Nothing, Nothing) 'Calls the "btnR_Click" event
        ElseIf (Asc(e.KeyChar) = 83 Or Asc(e.KeyChar) = 115) And btnS.Enabled = True Then
'Checks if the button pressed on the keyboard is either "S" or "s" and checks if the button
is enabled
            btnS_Click(Nothing, Nothing) 'Calls the "btnS_Click" event
        ElseIf (Asc(e.KeyChar) = 84 Or Asc(e.KeyChar) = 116) And btnT.Enabled = True Then
'Checks if the button pressed on the keyboard is either "T" or "t" and checks if the button
is enabled
            btnT_Click(Nothing, Nothing) 'Calls the "btnT_Click" event
        ElseIf (Asc(e.KeyChar) = 85 Or Asc(e.KeyChar) = 117) And btnU.Enabled = True Then
'Checks if the button pressed on the keyboard is either "U" or "u" and checks if the button
is enabled
            btnU_Click(Nothing, Nothing) 'Calls the "btnU_Click" event
        ElseIf (Asc(e.KeyChar) = 86 Or Asc(e.KeyChar) = 118) And btnV.Enabled = True Then
'Checks if the button pressed on the keyboard is either "V" or "v" and checks if the button
is enabled
            btnV_Click(Nothing, Nothing) 'Calls the "btnV_Click" event
        ElseIf (Asc(e.KeyChar) = 87 Or Asc(e.KeyChar) = 119) And btnW.Enabled = True Then
'Checks if the button pressed on the keyboard is either "W" or "w" and checks if the button
is enabled
            btnW_Click(Nothing, Nothing) 'Calls the "btnW_Click" event
        ElseIf (Asc(e.KeyChar) = 88 Or Asc(e.KeyChar) = 120) And btnX.Enabled = True Then
'Checks if the button pressed on the keyboard is either "X" or "x" and checks if the button
is enabled
            btnX_Click(Nothing, Nothing) 'Calls the "btnX_Click" event
        ElseIf (Asc(e.KeyChar) = 89 Or Asc(e.KeyChar) = 121) And btnY.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Y" or "y" and checks if the button
is enabled
            btnY_Click(Nothing, Nothing) 'Calls the "btnY_Click" event
        ElseIf (Asc(e.KeyChar) = 90 Or Asc(e.KeyChar) = 122) And btnZ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Z" or "z" and checks if the button
is enabled
            btnZ_Click(Nothing, Nothing) 'Calls the "btnZ_Click" event
        End If
    End Sub
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized 'Minimizes the form
```

```
    End Sub
    Private Sub btnStart_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseDown 'btnStart Button MouseDown
code
        btnStart.BackgroundImage = My.Resources.Start_Button_Pushed 'Changes the background
image of the "btnStart" button when the mouse is down
    End Sub
    Private Sub btnStart_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseEnter 'btnStart Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.BackgroundImage = My.Resources.Start_Button_Highlighted 'Changes the
background image of the "btnStart" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnStart_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseLeave 'btnStart Button MouseLeave code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnStart_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseUp 'btnStart Button MouseUp code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
        PicPlayerOne.BackgroundImage = My.Resources.HangMan_Player
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.Visible = False
        lblGuess.Visible = True
        GameStart()
    End Sub
    'Close
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
```

```
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    'Info
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        lblInfo.Visible = True
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False
        btnInfo.BackgroundImage = My.Resources.Info_Button
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = appPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        lblLeaveGame.Visible = True 'Shows the label
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
```

```
        lblLeaveGame.Visible = False 'Hides the label
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        If ExitFlag = True Then 'Checks if "ExitFlag" is set to True
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
            Exit Sub 'Exits the subroutine
        End If
        Dim MessageBoxResult As String 'Declares "MessageBoxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you wish to exit the game?", vbYesNo, "Exit
Game") 'Prompts the user if they wish to exit the game
        If MessageBoxResult = vbYes Then 'Checks if the messagebox selection returns "vbYes"
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            ExitFlag = True 'Sets the boolean variable "ExitFlag" to True
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
        End If
    End Sub
    'Alphabet Buttons
    Private Sub btnA_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnA.Click 'btnA button mouse click event
        btnA.BackColor = Color.DimGray 'Sets the "btnA" button's backcolor to DimGray
        btnA.Enabled = False 'Disables the "btnA" button
        CurrentLetter = "A" 'Sets the "CurrentLetter" string variable to "A"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnB_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnB.Click 'btnB button mouse click event
        btnB.BackColor = Color.DimGray 'Sets the "btnB" button's backcolor to DimGray
        btnB.Enabled = False 'Disables the "btnB" button
        CurrentLetter = "B" 'Sets the "CurrentLetter" string variable to "B"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnC_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnC.Click 'btnC button mouse click event
        btnC.BackColor = Color.DimGray 'Sets the "btnC" button's backcolor to DimGray
        btnC.Enabled = False 'Disables the "btnC" button
        CurrentLetter = "C" 'Sets the "CurrentLetter" string variable to "C"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnD_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnD.Click 'btnD button mouse click event
        btnD.BackColor = Color.DimGray 'Sets the "btnD" button's backcolor to DimGray
        btnD.Enabled = False 'Disables the "btnD" button
        CurrentLetter = "D" 'Sets the "CurrentLetter" string variable to "D"
        Guess() 'Calls the "Guess" subroutine
```

```
    End Sub
    Private Sub btnE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnE.Click 'btnE button mouse click event
        btnE.BackColor = Color.DimGray 'Sets the "btnE" button's backcolor to DimGray
        btnE.Enabled = False 'Disables the "btnE" button
        CurrentLetter = "E" 'Sets the "CurrentLetter" string variable to "E"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnF.Click 'btnF button mouse click event
        btnF.BackColor = Color.DimGray 'Sets the "btnF" button's backcolor to DimGray
        btnF.Enabled = False 'Disables the "btnF" button
        CurrentLetter = "F" 'Sets the "CurrentLetter" string variable to "F"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnG_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnG.Click 'btnG button mouse click event
        btnG.BackColor = Color.DimGray 'Sets the "btnG" button's backcolor to DimGray
        btnG.Enabled = False 'Disables the "btnG" button
        CurrentLetter = "G" 'Sets the "CurrentLetter" string variable to "G"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnH_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnH.Click 'btnH button mouse click event
        btnH.BackColor = Color.DimGray 'Sets the "btnH" button's backcolor to DimGray
        btnH.Enabled = False 'Disables the "btnH" button
        CurrentLetter = "H" 'Sets the "CurrentLetter" string variable to "H"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnI_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnI.Click 'btnI button mouse click event
        btnI.BackColor = Color.DimGray 'Sets the "btnI" button's backcolor to DimGray
        btnI.Enabled = False 'Disables the "btnI" button
        CurrentLetter = "I" 'Sets the "CurrentLetter" string variable to "I"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnJ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnJ.Click 'btnJ button mouse click event
        btnJ.BackColor = Color.DimGray 'Sets the "btnJ" button's backcolor to DimGray
        btnJ.Enabled = False 'Disables the "btnJ" button
        CurrentLetter = "J" 'Sets the "CurrentLetter" string variable to "J"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnK.Click 'btnK button mouse click event
        btnK.BackColor = Color.DimGray 'Sets the "btnK" button's backcolor to DimGray
        btnK.Enabled = False 'Disables the "btnK" button
        CurrentLetter = "K" 'Sets the "CurrentLetter" string variable to "K"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnL_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnL.Click 'btnL button mouse click event
        btnL.BackColor = Color.DimGray 'Sets the "btnL" button's backcolor to DimGray
        btnL.Enabled = False 'Disables the "btnL" button
        CurrentLetter = "L" 'Sets the "CurrentLetter" string variable to "L"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnM_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnM.Click 'btnM button mouse click event
        btnM.BackColor = Color.DimGray 'Sets the "btnM" button's backcolor to DimGray
        btnM.Enabled = False 'Disables the "btnM" button
        CurrentLetter = "M" 'Sets the "CurrentLetter" string variable to "M"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnN.Click 'btnN button mouse click event
```

```
        btnN.BackColor = Color.DimGray 'Sets the "btnN" button's backcolor to DimGray
        btnN.Enabled = False 'Disables the "btnN" button
        CurrentLetter = "N" 'Sets the "CurrentLetter" string variable to "N"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnO.Click 'btnO button mouse click event
        btnO.BackColor = Color.DimGray 'Sets the "btnO" button's backcolor to DimGray
        btnO.Enabled = False 'Disables the "btnO" button
        CurrentLetter = "O" 'Sets the "CurrentLetter" string variable to "O"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnP.Click 'btnP button mouse click event
        btnP.BackColor = Color.DimGray 'Sets the "btnP" button's backcolor to DimGray
        btnP.Enabled = False 'Disables the "btnP" button
        CurrentLetter = "P" 'Sets the "CurrentLetter" string variable to "P"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnQ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnQ.Click 'btnQ button mouse click event
        btnQ.BackColor = Color.DimGray 'Sets the "btnQ" button's backcolor to DimGray
        btnQ.Enabled = False 'Disables the "btnQ" button
        CurrentLetter = "Q" 'Sets the "CurrentLetter" string variable to "Q"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnR_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnR.Click 'btnR button mouse click event
        btnR.BackColor = Color.DimGray 'Sets the "btnR" button's backcolor to DimGray
        btnR.Enabled = False 'Disables the "btnR" button
        CurrentLetter = "R" 'Sets the "CurrentLetter" string variable to "R"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnS.Click 'btnS button mouse click event
        btnS.BackColor = Color.DimGray 'Sets the "btnS" button's backcolor to DimGray
        btnS.Enabled = False 'Disables the "btnS" button
        CurrentLetter = "S" 'Sets the "CurrentLetter" string variable to "S"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnT_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnT.Click 'btnT button mouse click event
        btnT.BackColor = Color.DimGray 'Sets the "btnT" button's backcolor to DimGray
        btnT.Enabled = False 'Disables the "btnT" button
        CurrentLetter = "T" 'Sets the "CurrentLetter" string variable to "T"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnU_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnU.Click 'btnU button mouse click event
        btnU.BackColor = Color.DimGray 'Sets the "btnU" button's backcolor to DimGray
        btnU.Enabled = False 'Disables the "btnU" button
        CurrentLetter = "U" 'Sets the "CurrentLetter" string variable to "U"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnV_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnV.Click 'btnV button mouse click event
        btnV.BackColor = Color.DimGray 'Sets the "btnV" button's backcolor to DimGray
        btnV.Enabled = False 'Disables the "btnV" button
        CurrentLetter = "V" 'Sets the "CurrentLetter" string variable to "V"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnW_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnW.Click 'btnW button mouse click event
        btnW.BackColor = Color.DimGray 'Sets the "btnW" button's backcolor to DimGray
        btnW.Enabled = False 'Disables the "btnW" button
        CurrentLetter = "W" 'Sets the "CurrentLetter" string variable to "W"
```

```
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnX_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnX.Click 'btnX button mouse click event
        btnX.BackColor = Color.DimGray 'Sets the "btnX" button's backcolor to DimGray
        btnX.Enabled = False 'Disables the "btnX" button
        CurrentLetter = "X" 'Sets the "CurrentLetter" string variable to "X"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnY_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnY.Click 'btnY button mouse click event
        btnY.BackColor = Color.DimGray 'Sets the "btnY" button's backcolor to DimGray
        btnY.Enabled = False 'Disables the "btnY" button
        CurrentLetter = "Y" 'Sets the "CurrentLetter" string variable to "Y"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnZ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnZ.Click 'btnZ button mouse click event
        btnZ.BackColor = Color.DimGray 'Sets the "btnZ" button's backcolor to DimGray
        btnZ.Enabled = False 'Disables the "btnZ" button
        CurrentLetter = "Z" 'Sets the "CurrentLetter" string variable to "Z"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnA_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnZ.MouseUp, btnY.MouseUp, btnX.MouseUp,
btnW.MouseUp, btnV.MouseUp, btnU.MouseUp, btnT.MouseUp, btnS.MouseUp, btnR.MouseUp,
btnQ.MouseUp, btnP.MouseUp, btnO.MouseUp, btnN.MouseUp, btnM.MouseUp, btnL.MouseUp,
btnK.MouseUp, btnJ.MouseUp, btnI.MouseUp, btnH.MouseUp, btnG.MouseUp, btnF.MouseUp,
btnE.MouseUp, btnD.MouseUp, btnC.MouseUp, btnB.MouseUp, btnA.MouseUp
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_select2 'Sets the soundplayer to
the "sound_select2" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub CountDownTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CountDownTimer.Tick
        CountDownCounter = CountDownCounter - 1 'Counts down the variable "CountDownCounter"
        If CountDownCounter = 5 Then 'Checks if the variable "CountDownCounter" is equal to 5
            PicPlayerOne.Image = My.Resources.Hangman_Four
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the
soundplayer to the "sound_FormSelect" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 4 Then 'Checks if the variable "CountDownCounter" is equal to 4
            PicPlayerOne.Image = My.Resources.Hangman_Three
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the
soundplayer to the "sound_FormSelect" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 3 Then 'Checks if the variable "CountDownCounter" is equal to 3
            PicPlayerOne.Image = My.Resources.Hangman_Two
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the
soundplayer to the "sound_FormSelect" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
```

```
        If CountDownCounter = 2 Then 'Checks if the variable "CountDownCounter" is equal to 2
            PicPlayerOne.Image = My.Resources.Hangman_One
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the
soundplayer to the "sound_FormSelect" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 1 Then 'Checks if the variable "CountDownCounter" is equal to 1
            PicPlayerOne.Image = Nothing
            GameStart() 'Calls the "GameStart" subroutine
            CountDownTimer.Stop() 'Stops the "CountDownTimer" timer
            CountDownCounter = 6 'CountDownCounter is set to 6
        End If
    End Sub
    Private Sub GameTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles GameTimer.Tick 'GameTimer Timer Tick code
        PicPlayerOne.Image = Nothing
        GameStart() 'Calls the "GameStart" subroutine
        GameTimer.Stop() 'Stops the "GameTimer" timer
    End Sub
    Private Sub WinnerSoundTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles WinnerSoundTimer.Tick 'WinnerSoundTimer Timer Tick code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer
to the "sound_FormSelect" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer's event
    End Sub
    Private Sub LoserSoundTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LoserSoundTimer.Tick 'LoserSoundTimer Timer Tick code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_rejected 'Sets the soundplayer to
the "sound_rejected" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        LoserSoundTimer.Stop() 'Stops the "LoserSoundTimer" timer's event
    End Sub
    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblLeaveGame.Visible = False 'Hides the "lblLeaveGame" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop()  'Stops the "CaptionTimer" timer
            lblInfo.Visible = True 'Shows the "lblInfo" label
            lblLeaveGame.Visible = True 'Shows the "lblLeaveGame" label
```

```
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub

End Class
```

8.5. Hangman – Two Player:

```
Public Class frmHangmanTwoPlayer 'frmHangmanTwoPlayer form code
    Dim Alphabet As String
    Dim Word As String = ""
    Dim WordEntered As String
    Dim CurrentLetter As String
    Dim Toggle As Boolean = False
    Dim ExitFlag As Boolean = True
    Dim BeepSound As Boolean = False
    Dim PlayerFlash As Boolean = False
    Dim DisplayToken As Boolean = False
    Dim PlayerOneTokenNo As Integer = 0
    Dim PlayerTwoTokenNo As Integer = 0
    Dim CountDownCounter As Integer = 6
    Dim PlayerOneGuesses As Integer = 10
    Dim PlayerTwoGuesses As Integer = 10
    Dim SwitchedPlayer As Boolean = False
    Dim AppPath As String = Application.StartupPath
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Public Function PlayerOne_GetImage(ByVal bounds As Size) As Image 'PlayerOne_GetImage
function code
        'Draw each piece of the man,
        'as applicable
        Dim img As New Bitmap(bounds.Width, bounds.Height)
```

```
        If PlayerOneGuesses = 0 Then
            img = My.Resources.HangMan_CurrentPlayer_10
        End If
        If PlayerOneGuesses > 0 Then
            img = My.Resources.HangMan_CurrentPlayer_8
        End If
        If PlayerOneGuesses > 1 Then
            img = My.Resources.HangMan_CurrentPlayer_7
        End If
        If PlayerOneGuesses > 2 Then
            img = My.Resources.HangMan_CurrentPlayer_6
        End If
        If PlayerOneGuesses > 3 Then
            img = My.Resources.HangMan_CurrentPlayer_5
        End If
        If PlayerOneGuesses > 4 Then
            img = My.Resources.HangMan_CurrentPlayer_4
        End If

        'Release the Graphics object
        'Return the image
        Return img

    End Function
    Public Function PlayerTwo_GetImage(ByVal bounds As Size) As Image 'PlayerTwo_GetImage
function code
        Dim img As New Bitmap(bounds.Width, bounds.Height)
        If PlayerTwoGuesses = 0 Then
            img = My.Resources.HangMan_CurrentPlayer_10
        End If
        If PlayerTwoGuesses > 0 Then
            img = My.Resources.HangMan_CurrentPlayer_8
        End If
        If PlayerTwoGuesses > 1 Then
            img = My.Resources.HangMan_CurrentPlayer_7
        End If
        If PlayerTwoGuesses > 2 Then
            img = My.Resources.HangMan_CurrentPlayer_6
        End If
        If PlayerTwoGuesses > 3 Then
            img = My.Resources.HangMan_CurrentPlayer_5
        End If
        If PlayerTwoGuesses > 4 Then
            img = My.Resources.HangMan_CurrentPlayer_4
        End If
        Return img
    End Function
    Function CurrentGuess(ByVal Letter As String) 'CurrentGuess function code
        Dim ReturnValid As Boolean = False
        Dim Index As Integer
        For i = 0 To Word.Length - 1
            If Word.Substring(i, 1) = Letter.ToUpper Then
                Index = i
                Index = Index + 1
                Mid(lblGuess.Text, Index, 1) = Letter
                ReturnValid = True
            End If
        Next i
        Return ReturnValid
    End Function
    Public Sub Guess() 'Guess subroutine code
        lblGuess.Focus() 'Focuses on the "lblGuess" label
        picGameStatus.BackgroundImage = Nothing
        If SwitchedPlayer = False Then
            If CurrentGuess(CurrentLetter) = False Then
                PlayerOneGuesses = PlayerOneGuesses - 1
```

```
                GetLives() 'Calls the GetLives subroutine
                picGameStatus.BackgroundImage = PlayerOne_GetImage(picGameStatus.Size)
                If PlayerOneGuesses = 1 Then
                    lblPlayerOneLives.Text = PlayerOneGuesses & " Life Remaining"
                Else
                    lblPlayerOneLives.Text = PlayerOneGuesses & " Lives Remaining"
                End If
            Else
                picGameStatus.BackgroundImage = PlayerOne_GetImage(picGameStatus.Size)
            End If
            If Word.ToUpper = lblGuess.Text.ToUpper Then
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                    frmMainMenu.player.Play() 'Plays the sound file
                End If
                lblGuess.ForeColor = Color.Lime
                lblGuess.Text = Word.ToUpper 'Converts all characters in the "lblGuess" label
to UPPERCASE
                DisplayTimer.Start()
            End If
            If PlayerOneGuesses < 1 Then
                lblGuess.ForeColor = Color.Maroon 'Sets the "lblGuess" label's forecolor to
Maroon
                lblGuess.Text = Word.ToUpper 'Converts all characters in the "lblGuess" label
to UPPERCASE
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    frmMainMenu.player.Stream = My.Resources.sound_rejected
                    frmMainMenu.player.Play() 'Plays the sound file
                End If
                DisplayTimer.Start()
            End If
        Else
            If CurrentGuess(CurrentLetter) = False Then
                PlayerTwoGuesses = PlayerTwoGuesses - 1
                GetLives() 'Calls the GetLives subroutine
                picGameStatus.BackgroundImage = PlayerTwo_GetImage(picGameStatus.Size)
                If PlayerTwoGuesses = 1 Then
                    lblPlayerTwoLives.Text = PlayerTwoGuesses & " Life Remaining"
                Else
                    lblPlayerTwoLives.Text = PlayerTwoGuesses & " Lives Remaining"
                End If
            Else
                picGameStatus.BackgroundImage = PlayerTwo_GetImage(picGameStatus.Size)
            End If
            If Word.ToUpper = lblGuess.Text.ToUpper Then
                lblGuess.ForeColor = Color.Lime
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                    frmMainMenu.player.Play() 'Plays the sound file
                    CheckWinner() 'Calls the CheckWinner subroutine
                    Exit Sub
                End If
            End If
            If PlayerTwoGuesses < PlayerOneGuesses Then
                lblGuess.ForeColor = Color.Maroon 'Sets the "lblGuess" label's forecolor to
Maroon
                lblGuess.Text = Word.ToUpper 'Converts all characters in the "lblGuess" label
to UPPERCASE
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    frmMainMenu.player.Stream = My.Resources.sound_rejected
                    frmMainMenu.player.Play() 'Plays the sound file
                End If
```

```
                CheckWinner() 'Calls the CheckWinner subroutine
                Exit Sub
            End If
            If PlayerTwoGuesses < 1 Then
                If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                    frmMainMenu.player.Stream = My.Resources.sound_rejected
                    frmMainMenu.player.Play() 'Plays the sound file
                End If
                CheckWinner() 'Calls the CheckWinner subroutine
            End If
        End If
    End Sub

    Public Sub CheckWinner() 'CheckWinner subroutine code
        If PlayerOneTokenNo = 5 Then
            GetTokens()
            picGameStatus.Image = My.Resources.HangMan_GameOver
            PicPlayerOneStatus.Image = My.Resources.Hangman_Player_One_Winner
            PicPlayerTwoStatus.Image = My.Resources.Hangman_Player_Two_Loser
            PlayerOneTokenNo = 0
            NewGame()
            Exit Sub
        End If
        If PlayerTwoTokenNo = 5 Then
            GetTokens()
            picGameStatus.Image = My.Resources.HangMan_GameOver
            PicPlayerOneStatus.Image = My.Resources.Hangman_Player_One_loser
            PicPlayerTwoStatus.Image = My.Resources.Hangman_Player_Two_Winner
            PlayerTwoTokenNo = 0
            NewGame()
            Exit Sub
        End If
        If PlayerOneGuesses = PlayerTwoGuesses Then
            picGameStatus.Image = My.Resources.HangMan_Draw
            frmMainMenu.player.Stream = My.Resources.sound_rejected
            frmMainMenu.player.Play()
            WinnerTimer.Start()
            Exit Sub
        ElseIf PlayerOneGuesses > PlayerTwoGuesses Then
            PlayerOneTokenNo = PlayerOneTokenNo + 1
            DisplayToken = False
        ElseIf PlayerOneGuesses < PlayerTwoGuesses Then
            PlayerTwoTokenNo = PlayerTwoTokenNo + 1
            DisplayToken = True
        End If
        If DisplayToken = False Then
            DisableLetters() 'Calls the DisableLetters subroutine
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing
            frmMainMenu.player.Play()
            picPlayerOneToken.Visible = True
            GetTokens() 'Calls the GetTokens subroutine
            WinnerTimer.Start()
        Else
            DisableLetters() 'Calls the DisableLetters subroutine
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing
            frmMainMenu.player.Play()
            picPlayerTwoToken.Visible = True
            GetTokens() 'Calls the GetTokens subroutine
            WinnerTimer.Start()
        End If
    End Sub
    Public Sub SwitchPlayers() 'SwitchPlayers subroutine code
        StopTimers() 'Calls the StopTimers subroutine
        lblPlayerOneLives.ForeColor = Color.Black
        lblPlayerTwoLives.ForeColor = Color.Black
```

```
        If SwitchedPlayer = False Then
            SwitchedPlayer = True
        Else
            SwitchedPlayer = False
        End If
    End Sub
    Public Sub GetLives() 'GetLives subroutine code
        If SwitchedPlayer = False Then
            If PlayerOneGuesses < 1 Then
                Player1_Life1.BackgroundImage = My.Resources.No_Token
                LifeFlashTimer.Stop()
                lblPlayerOneLives.ForeColor = Color.Black
            End If
            If PlayerOneGuesses < 2 Then
                Player1_Life2.BackgroundImage = My.Resources.No_Token
                PlayerFlash = False
                LifeFlashTimer.Start()
            End If
            If PlayerOneGuesses < 3 Then
                Player1_Life3.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 4 Then
                Player1_Life4.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 5 Then
                Player1_Life5.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 6 Then
                Player1_Life6.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 7 Then
                Player1_Life7.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 8 Then
                Player1_Life8.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 9 Then
                Player1_Life9.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerOneGuesses < 10 Then
                Player1_Life10.BackgroundImage = My.Resources.No_Token
            End If
        Else
            If PlayerTwoGuesses < 1 Then
                Player2_Life1.BackgroundImage = My.Resources.No_Token
                lblPlayerTwoLives.ForeColor = Color.Black
            End If
            If PlayerTwoGuesses < 2 Then
                Player2_Life2.BackgroundImage = My.Resources.No_Token
                lblPlayerTwoLives.ForeColor = Color.Red
                PlayerFlash = True
                LifeFlashTimer.Start()
            End If
            If PlayerTwoGuesses < 3 Then
                Player2_Life3.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerTwoGuesses < 4 Then
                Player2_Life4.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerTwoGuesses < 5 Then
                Player2_Life5.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerTwoGuesses < 6 Then
                Player2_Life6.BackgroundImage = My.Resources.No_Token
            End If
            If PlayerTwoGuesses < 7 Then
```

```
            Player2_Life7.BackgroundImage = My.Resources.No_Token
        End If
        If PlayerTwoGuesses < 8 Then
            Player2_Life8.BackgroundImage = My.Resources.No_Token
        End If
        If PlayerTwoGuesses < 9 Then
            Player2_Life9.BackgroundImage = My.Resources.No_Token
        End If
        If PlayerTwoGuesses < 10 Then
            Player2_Life10.BackgroundImage = My.Resources.No_Token
        End If
    End If
End Sub
Private Sub EnableLetters() 'EnableLetters subroutine code
    btnA.BackColor = Color.Silver
    btnB.BackColor = Color.Silver
    btnC.BackColor = Color.Silver
    btnD.BackColor = Color.Silver
    btnE.BackColor = Color.Silver
    btnF.BackColor = Color.Silver
    btnG.BackColor = Color.Silver
    btnH.BackColor = Color.Silver
    btnI.BackColor = Color.Silver
    btnJ.BackColor = Color.Silver
    btnK.BackColor = Color.Silver
    btnL.BackColor = Color.Silver
    btnM.BackColor = Color.Silver
    btnN.BackColor = Color.Silver
    btnO.BackColor = Color.Silver
    btnP.BackColor = Color.Silver
    btnQ.BackColor = Color.Silver
    btnR.BackColor = Color.Silver
    btnS.BackColor = Color.Silver
    btnT.BackColor = Color.Silver
    btnU.BackColor = Color.Silver
    btnV.BackColor = Color.Silver
    btnW.BackColor = Color.Silver
    btnX.BackColor = Color.Silver
    btnY.BackColor = Color.Silver
    btnZ.BackColor = Color.Silver
    'Re-enables all of the buttons
    btnA.Enabled = True
    btnB.Enabled = True
    btnC.Enabled = True
    btnD.Enabled = True
    btnE.Enabled = True
    btnF.Enabled = True
    btnG.Enabled = True
    btnH.Enabled = True
    btnI.Enabled = True
    btnJ.Enabled = True
    btnK.Enabled = True
    btnL.Enabled = True
    btnM.Enabled = True
    btnN.Enabled = True
    btnO.Enabled = True
    btnP.Enabled = True
    btnQ.Enabled = True
    btnR.Enabled = True
    btnS.Enabled = True
    btnT.Enabled = True
    btnU.Enabled = True
    btnV.Enabled = True
    btnW.Enabled = True
    btnX.Enabled = True
    btnY.Enabled = True
```

```
        btnZ.Enabled = True
    End Sub
    Private Sub DisableLetters() 'DisableLetters subroutine code
        btnA.BackColor = Color.DimGray
        btnB.BackColor = Color.DimGray
        btnC.BackColor = Color.DimGray
        btnD.BackColor = Color.DimGray
        btnE.BackColor = Color.DimGray
        btnF.BackColor = Color.DimGray
        btnG.BackColor = Color.DimGray
        btnH.BackColor = Color.DimGray
        btnI.BackColor = Color.DimGray
        btnJ.BackColor = Color.DimGray
        btnK.BackColor = Color.DimGray
        btnL.BackColor = Color.DimGray
        btnM.BackColor = Color.DimGray
        btnN.BackColor = Color.DimGray
        btnO.BackColor = Color.DimGray
        btnP.BackColor = Color.DimGray
        btnQ.BackColor = Color.DimGray
        btnR.BackColor = Color.DimGray
        btnS.BackColor = Color.DimGray
        btnT.BackColor = Color.DimGray
        btnU.BackColor = Color.DimGray
        btnV.BackColor = Color.DimGray
        btnW.BackColor = Color.DimGray
        btnX.BackColor = Color.DimGray
        btnY.BackColor = Color.DimGray
        btnZ.BackColor = Color.DimGray
        btnA.Enabled = False
        btnB.Enabled = False
        btnC.Enabled = False
        btnD.Enabled = False
        btnE.Enabled = False
        btnF.Enabled = False
        btnG.Enabled = False
        btnH.Enabled = False
        btnI.Enabled = False
        btnJ.Enabled = False
        btnK.Enabled = False
        btnL.Enabled = False
        btnM.Enabled = False
        btnN.Enabled = False
        btnO.Enabled = False
        btnP.Enabled = False
        btnQ.Enabled = False
        btnR.Enabled = False
        btnS.Enabled = False
        btnT.Enabled = False
        btnU.Enabled = False
        btnV.Enabled = False
        btnW.Enabled = False
        btnX.Enabled = False
        btnY.Enabled = False
        btnZ.Enabled = False
    End Sub
    Public Sub ClearTokens() 'ClearTokens subroutine code
        Player1_GoldToken1.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken2.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken3.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken4.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken5.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken1.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken2.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken3.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken4.BackgroundImage = My.Resources.No_Token
```

```
        Player2_GoldToken5.BackgroundImage = My.Resources.No_Token
    End Sub
    Public Sub RestartLives() 'RestartLives subroutine code
        PlayerOneGuesses = 10
        PlayerTwoGuesses = 10
        Player1_Life1.BackgroundImage = My.Resources.LifeIcon
        Player1_Life2.BackgroundImage = My.Resources.LifeIcon
        Player1_Life3.BackgroundImage = My.Resources.LifeIcon
        Player1_Life4.BackgroundImage = My.Resources.LifeIcon
        Player1_Life5.BackgroundImage = My.Resources.LifeIcon
        Player1_Life6.BackgroundImage = My.Resources.LifeIcon
        Player1_Life7.BackgroundImage = My.Resources.LifeIcon
        Player1_Life8.BackgroundImage = My.Resources.LifeIcon
        Player1_Life9.BackgroundImage = My.Resources.LifeIcon
        Player1_Life10.BackgroundImage = My.Resources.LifeIcon
        Player2_Life1.BackgroundImage = My.Resources.LifeIcon
        Player2_Life2.BackgroundImage = My.Resources.LifeIcon
        Player2_Life3.BackgroundImage = My.Resources.LifeIcon
        Player2_Life4.BackgroundImage = My.Resources.LifeIcon
        Player2_Life5.BackgroundImage = My.Resources.LifeIcon
        Player2_Life6.BackgroundImage = My.Resources.LifeIcon
        Player2_Life7.BackgroundImage = My.Resources.LifeIcon
        Player2_Life8.BackgroundImage = My.Resources.LifeIcon
        Player2_Life9.BackgroundImage = My.Resources.LifeIcon
        Player2_Life10.BackgroundImage = My.Resources.LifeIcon
    End Sub
    Public Sub GetTokens() 'GetTokens subroutine code
        If DisplayToken = False Then
            If PlayerOneTokenNo > 0 Then
                Player1_GoldToken1.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerOneTokenNo > 1 Then
                Player1_GoldToken2.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerOneTokenNo > 2 Then
                Player1_GoldToken3.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerOneTokenNo > 3 Then
                Player1_GoldToken4.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerOneTokenNo > 4 Then
                Player1_GoldToken5.BackgroundImage = My.Resources.Gold_Token
            End If
            lblPlayerOneTokens.Text = "Tokens: " & PlayerOneTokenNo
        Else
            If PlayerTwoTokenNo > 0 Then
                Player2_GoldToken1.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerTwoTokenNo > 1 Then
                Player2_GoldToken2.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerTwoTokenNo > 2 Then
                Player2_GoldToken3.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerTwoTokenNo > 3 Then
                Player2_GoldToken4.BackgroundImage = My.Resources.Gold_Token
            End If
            If PlayerTwoTokenNo > 4 Then
                Player2_GoldToken5.BackgroundImage = My.Resources.Gold_Token
            End If
            lblPlayerTwoTokens.Text = "Tokens: " & PlayerTwoTokenNo
        End If
    End Sub
    Public Sub StopTimers() 'StopTimers subroutine code
        CountDownTimer.Stop() 'Stops CountDownTimer
        TokenTimer.Stop() 'Stops TokenTimer
```

```
        LifeFlashTimer.Stop() 'Stops LifeFlashTimer
    End Sub
    Public Sub Restart() 'Restart subroutine code
        picGameStatus.Image = Nothing
        StopTimers() 'Calls the StopTimers subroutine
        DisableLetters() 'Calls the DisableLetters subroutine
        btnStart.Visible = True
        lblGuess.Visible = False
        lblGuess.Text = "Game Starting..."
        'Changes the back color of all buttons
    End Sub
    Public Sub GameStart()
        Word = WordEntered
        BeepSound = False
        lblGuess.ForeColor = Color.Black
        EnableLetters() 'Calls the EnableLetters subroutine
        GetTokens() 'Calls the GetTokens subroutine
        picGameStatus.Image = Nothing
        picGameStatus.BackgroundImage = My.Resources.HangMan_CurrentPlayer_4
        For i = 0 To Word.Length - 1
            If Word(i) = " " Then
                lblGuess.Text &= " "
            Else
                lblGuess.Text &= "-"
            End If
        Next i
        CurrentLetter = ""
    End Sub
    Public Sub NewRound()
        RestartLives()  'Calls the RestartLives subroutine
        StopTimers()  'Calls the StopTimers subroutine
        DisableLetters()  'Calls the DisableLetters subroutine
        lblGuess.Visible = False 'Hides the label
        btnStart.Visible = True 'Shows the label
        lblPlayerOne.ForeColor = Color.Black
        lblPlayerTwo.ForeColor = Color.Black
        lblPlayerOneLives.Text = "10 Lives Remaining"
        lblPlayerTwoLives.Text = "10 Lives Remaining"
        PlayerOneGuesses = 10
        PlayerTwoGuesses = 10
        RestartLives()
        SwitchedPlayer = False
    End Sub
    Public Sub NewGame()
        StopTimers()
        DisableLetters()
        lblGuess.Visible = False
        btnStart.Visible = True
        lblGuess.Text = "Game Starting..."
        PlayerOneTokenNo = 0
        PlayerTwoTokenNo = 0
        lblPlayerOneTokens.Text = "Tokens: 0"
        lblPlayerTwoTokens.Text = "Tokens: 0"
        PlayerOneGuesses = 10
        PlayerTwoGuesses = 10
        lblPlayerOneLives.Text = "10 Lives Remaining"
        lblPlayerTwoLives.Text = "10 Lives Remaining"
        SwitchedPlayer = False
    End Sub

    Private Sub frmHangmanTwoPlayer_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles MyBase.KeyPress
        If Asc(e.KeyChar) = 13 And btnStart.Visible = True Then
            btnStart_MouseUp(Nothing, Nothing)
        End If
```

```
        If (Asc(e.KeyChar) = 65 Or Asc(e.KeyChar) = 97) And btnA.Enabled = True Then 'Checks
if the button pressed on the keyboard is either "A" or "a" and checks if the button is
enabled
            btnA_Click(Nothing, Nothing) 'Calls the "btnA_Click" event
        ElseIf (Asc(e.KeyChar) = 66 Or Asc(e.KeyChar) = 98) And btnB.Enabled = True Then
'Checks if the button pressed on the keyboard is either "B" or "b" and checks if the button
is enabled
            btnB_Click(Nothing, Nothing) 'Calls the "btnB_Click" event
        ElseIf (Asc(e.KeyChar) = 67 Or Asc(e.KeyChar) = 99) And btnC.Enabled = True Then
'Checks if the button pressed on the keyboard is either "C" or "c" and checks if the button
is enabled
            btnC_Click(Nothing, Nothing) 'Calls the "btnC_Click" event
        ElseIf (Asc(e.KeyChar) = 68 Or Asc(e.KeyChar) = 100) And btnD.Enabled = True Then
'Checks if the button pressed on the keyboard is either "D" or "d" and checks if the button
is enabled
            btnD_Click(Nothing, Nothing) 'Calls the "btnD_Click" event
        ElseIf (Asc(e.KeyChar) = 69 Or Asc(e.KeyChar) = 101) And btnE.Enabled = True Then
'Checks if the button pressed on the keyboard is either "E" or "e" and checks if the button
is enabled
            btnE_Click(Nothing, Nothing) 'Calls the "btnE_Click" event
        ElseIf (Asc(e.KeyChar) = 70 Or Asc(e.KeyChar) = 102) And btnF.Enabled = True Then
'Checks if the button pressed on the keyboard is either "F" or "f" and checks if the button
is enabled
            btnF_Click(Nothing, Nothing) 'Calls the "btnF_Click" event
        ElseIf (Asc(e.KeyChar) = 71 Or Asc(e.KeyChar) = 103) And btnG.Enabled = True Then
'Checks if the button pressed on the keyboard is either "G" or "g" and checks if the button
is enabled
            btnG_Click(Nothing, Nothing) 'Calls the "btnG_Click" event
        ElseIf (Asc(e.KeyChar) = 72 Or Asc(e.KeyChar) = 104) And btnH.Enabled = True Then
'Checks if the button pressed on the keyboard is either "H" or "h" and checks if the button
is enabled
            btnH_Click(Nothing, Nothing) 'Calls the "btnH_Click" event
        ElseIf (Asc(e.KeyChar) = 73 Or Asc(e.KeyChar) = 105) And btnI.Enabled = True Then
'Checks if the button pressed on the keyboard is either "I" or "i" and checks if the button
is enabled
            btnI_Click(Nothing, Nothing) 'Calls the "btnI_Click" event
        ElseIf (Asc(e.KeyChar) = 74 Or Asc(e.KeyChar) = 106) And btnJ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "J" or "j" and checks if the button
is enabled
            btnJ_Click(Nothing, Nothing) 'Calls the "btnJ_Click" event
        ElseIf (Asc(e.KeyChar) = 75 Or Asc(e.KeyChar) = 107) And btnK.Enabled = True Then
'Checks if the button pressed on the keyboard is either "K" or "k" and checks if the button
is enabled
            btnK_Click(Nothing, Nothing) 'Calls the "btnK_Click" event
        ElseIf (Asc(e.KeyChar) = 76 Or Asc(e.KeyChar) = 108) And btnL.Enabled = True Then
'Checks if the button pressed on the keyboard is either "L" or "l" and checks if the button
is enabled
            btnL_Click(Nothing, Nothing) 'Calls the "btnL_Click" event
        ElseIf (Asc(e.KeyChar) = 77 Or Asc(e.KeyChar) = 109) And btnM.Enabled = True Then
'Checks if the button pressed on the keyboard is either "M" or "m" and checks if the button
is enabled
            btnM_Click(Nothing, Nothing) 'Calls the "btnM_Click" event
        ElseIf (Asc(e.KeyChar) = 78 Or Asc(e.KeyChar) = 110) And btnN.Enabled = True Then
'Checks if the button pressed on the keyboard is either "N" or "n" and checks if the button
is enabled
            btnN_Click(Nothing, Nothing) 'Calls the "btnN_Click" event
        ElseIf (Asc(e.KeyChar) = 79 Or Asc(e.KeyChar) = 111) And btnO.Enabled = True Then
'Checks if the button pressed on the keyboard is either "O" or "o" and checks if the button
is enabled
            btnO_Click(Nothing, Nothing) 'Calls the "btnO_Click" event
        ElseIf (Asc(e.KeyChar) = 80 Or Asc(e.KeyChar) = 112) And btnP.Enabled = True Then
'Checks if the button pressed on the keyboard is either "P" or "p" and checks if the button
is enabled
            btnP_Click(Nothing, Nothing) 'Calls the "btnP_Click" event
```

```
        ElseIf (Asc(e.KeyChar) = 81 Or Asc(e.KeyChar) = 113) And btnQ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Q" or "q" and checks if the button
is enabled
            btnQ_Click(Nothing, Nothing) 'Calls the "btnQ_Click" event
        ElseIf (Asc(e.KeyChar) = 82 Or Asc(e.KeyChar) = 114) And btnR.Enabled = True Then
'Checks if the button pressed on the keyboard is either "R" or "r" and checks if the button
is enabled
            btnR_Click(Nothing, Nothing) 'Calls the "btnR_Click" event
        ElseIf (Asc(e.KeyChar) = 83 Or Asc(e.KeyChar) = 115) And btnS.Enabled = True Then
'Checks if the button pressed on the keyboard is either "S" or "s" and checks if the button
is enabled
            btnS_Click(Nothing, Nothing) 'Calls the "btnS_Click" event
        ElseIf (Asc(e.KeyChar) = 84 Or Asc(e.KeyChar) = 116) And btnT.Enabled = True Then
'Checks if the button pressed on the keyboard is either "T" or "t" and checks if the button
is enabled
            btnT_Click(Nothing, Nothing) 'Calls the "btnT_Click" event
        ElseIf (Asc(e.KeyChar) = 85 Or Asc(e.KeyChar) = 117) And btnU.Enabled = True Then
'Checks if the button pressed on the keyboard is either "U" or "u" and checks if the button
is enabled
            btnU_Click(Nothing, Nothing) 'Calls the "btnU_Click" event
        ElseIf (Asc(e.KeyChar) = 86 Or Asc(e.KeyChar) = 118) And btnV.Enabled = True Then
'Checks if the button pressed on the keyboard is either "V" or "v" and checks if the button
is enabled
            btnV_Click(Nothing, Nothing) 'Calls the "btnV_Click" event
        ElseIf (Asc(e.KeyChar) = 87 Or Asc(e.KeyChar) = 119) And btnW.Enabled = True Then
'Checks if the button pressed on the keyboard is either "W" or "w" and checks if the button
is enabled
            btnW_Click(Nothing, Nothing) 'Calls the "btnW_Click" event
        ElseIf (Asc(e.KeyChar) = 88 Or Asc(e.KeyChar) = 120) And btnX.Enabled = True Then
'Checks if the button pressed on the keyboard is either "X" or "x" and checks if the button
is enabled
            btnX_Click(Nothing, Nothing) 'Calls the "btnX_Click" event
        ElseIf (Asc(e.KeyChar) = 89 Or Asc(e.KeyChar) = 121) And btnY.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Y" or "y" and checks if the button
is enabled
            btnY_Click(Nothing, Nothing) 'Calls the "btnY_Click" event
        ElseIf (Asc(e.KeyChar) = 90 Or Asc(e.KeyChar) = 122) And btnZ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Z" or "z" and checks if the button
is enabled
            btnZ_Click(Nothing, Nothing) 'Calls the "btnZ_Click" event
        End If
    End Sub
    'Buttons
    'Minimize
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
```

```
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized
    End Sub
    'Start
    Private Sub btnStart_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseDown 'btnStart Button MouseDown
code
        btnStart.BackgroundImage = My.Resources.Start_Button_Pushed 'Changes the background
image of the "btnStart" button when the mouse is down
    End Sub
    Private Sub btnStart_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseEnter 'btnStart Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.BackgroundImage = My.Resources.Start_Button_Highlighted 'Changes the
background image of the "btnStart" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnStart_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseLeave 'btnStart Button MouseLeave code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnStart_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseUp 'btnStart Button MouseUp code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
        If SwitchedPlayer = False Then
            Dim InputBoxResult As String
            InputBoxResult = InputBox(lblPlayerTwo.Text & ", please enter a word for " &
lblPlayerOne.Text & " to guess.", "Enter Word", "Enter Your Word Here")
            If InputBoxResult = "Enter Your Word Here" Or InputBoxResult = "" Then
                Exit Sub
            End If
            WordEntered = InputBoxResult.ToUpper
            Dim Valid As Boolean = True
            For a = 1 To 26
                If InputBoxResult.Contains(lstSymbol.Items(a)) Then
                    Valid = False
                End If
                If Valid = False Then
                    Exit For
                End If
            Next
            If Valid = False Then
                MsgBox("The word you entered is invalid. Words entered must contain
alphabetical letters only. Please re-enter your word again.", vbExclamation + vbOKOnly,
"Invalid Word")
                btnStart_MouseUp(sender, e)
                Exit Sub
            End If
            WordEntered = InputBoxResult.ToUpper
            picGameStatus.BackgroundImage = My.Resources.HangMan_Player
        Else
            Dim InputBoxResult As String
            InputBoxResult = InputBox(lblPlayerOne.Text & ", please enter a word for " &
lblPlayerTwo.Text & " to guess.", "Enter Word", "Enter Your Word Here")
            If InputBoxResult = "Enter Your Word Here" Or InputBoxResult = "" Then
```

```
                Exit Sub
            End If
            WordEntered = InputBoxResult.ToUpper
            Dim Valid As Boolean = True
            For a = 1 To 26
                If InputBoxResult.Contains(lstSymbol.Items(a)) Then
                    Valid = False
                End If
                If Valid = False Then
                    Exit For
                End If
            Next
            If Valid = False Then
                MsgBox("The word you entered is invalid. Words entered must contain
alphabetical letters only. Please re-enter your word again.", vbExclamation + vbOKOnly,
"Invalid Word")
                btnStart_MouseUp(sender, e)
                Exit Sub
            End If
            WordEntered = InputBoxResult.ToUpper
            picGameStatus.BackgroundImage = My.Resources.HangMan_Player
        End If
        CountDownTimer.Start()
        picGameStatus.Image = My.Resources.Hangman_Five
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.Visible = False
        lblGuess.Visible = True
        ExitFlag = False
        Word = WordEntered
    End Sub
    'Close
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If BeepSound = False Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
```

```
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    'Info
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        If BeepSound = False Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        lblInfo.Visible = True
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave
        lblInfo.Visible = False
        btnInfo.BackgroundImage = My.Resources.Info_Button
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button
        If My.Settings.Mute = False Then
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed
    End Sub
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        lblLeaveGame.Visible = True
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblLeaveGame.Visible = False
```

```
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        If ExitFlag = True Then 'Checks if ExitFlag is set to True
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
            Exit Sub
        End If
        Dim MessageBoxResult As String
        MessageBoxResult = MsgBox("Are you sure you wish to exit the game?", vbYesNo, "Exit
Game") 'Prompts the user if they wish to exit game
        If MessageBoxResult = vbYes Then
            lblPlayerOneLives.Text = "10 Lives Remaining" 'Displays "10 Lives Remaining" in
lblPlayerOneLives label
            lblPlayerOneLives.Text = "10 Lives Remaining" 'Displays "10 Lives Remaining" in
lblPlayerOneLives label
            PlayerOneGuesses = 10 'Sets PlayerOneGuesses to 10
            PlayerTwoGuesses = 10 'Sets PlayerTwoGuesses to 10
            CountDownCounter = 6 'Sets CountDownCounter to 10
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
            ExitFlag = True 'Sets ExitFlag to True
        End If
    End Sub

    'Alphabet
    Private Sub btnA_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnA.Click 'btnA button mouse click event
        btnA.BackColor = Color.DimGray 'Sets the "btnA" button's backcolor to DimGray
        btnA.Enabled = False 'Disables the "btnA" button
        CurrentLetter = "A" 'Sets the "CurrentLetter" string variable to "A"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnB_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnB.Click 'btnB button mouse click event
        btnB.BackColor = Color.DimGray 'Sets the "btnB" button's backcolor to DimGray
        btnB.Enabled = False 'Disables the "btnB" button
        CurrentLetter = "B" 'Sets the "CurrentLetter" string variable to "B"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnC_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnC.Click 'btnC button mouse click event
        btnC.BackColor = Color.DimGray 'Sets the "btnC" button's backcolor to DimGray
        btnC.Enabled = False 'Disables the "btnC" button
        CurrentLetter = "C" 'Sets the "CurrentLetter" string variable to "C"
        Guess() 'Calls the "Guess" subroutine
```

```
    End Sub
    Private Sub btnD_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnD.Click 'btnD button mouse click event
        btnD.BackColor = Color.DimGray 'Sets the "btnD" button's backcolor to DimGray
        btnD.Enabled = False 'Disables the "btnD" button
        CurrentLetter = "D" 'Sets the "CurrentLetter" string variable to "D"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnE.Click 'btnE button mouse click event
        btnE.BackColor = Color.DimGray 'Sets the "btnE" button's backcolor to DimGray
        btnE.Enabled = False 'Disables the "btnE" button
        CurrentLetter = "E" 'Sets the "CurrentLetter" string variable to "E"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnF.Click 'btnF button mouse click event
        btnF.BackColor = Color.DimGray 'Sets the "btnF" button's backcolor to DimGray
        btnF.Enabled = False 'Disables the "btnF" button
        CurrentLetter = "F" 'Sets the "CurrentLetter" string variable to "F"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnG_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnG.Click 'btnG button mouse click event
        btnG.BackColor = Color.DimGray 'Sets the "btnG" button's backcolor to DimGray
        btnG.Enabled = False 'Disables the "btnG" button
        CurrentLetter = "G" 'Sets the "CurrentLetter" string variable to "G"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnH_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnH.Click 'btnH button mouse click event
        btnH.BackColor = Color.DimGray 'Sets the "btnH" button's backcolor to DimGray
        btnH.Enabled = False 'Disables the "btnH" button
        CurrentLetter = "H" 'Sets the "CurrentLetter" string variable to "H"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnI_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnI.Click 'btnI button mouse click event
        btnI.BackColor = Color.DimGray 'Sets the "btnI" button's backcolor to DimGray
        btnI.Enabled = False 'Disables the "btnI" button
        CurrentLetter = "I" 'Sets the "CurrentLetter" string variable to "I"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnJ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnJ.Click 'btnJ button mouse click event
        btnJ.BackColor = Color.DimGray 'Sets the "btnJ" button's backcolor to DimGray
        btnJ.Enabled = False 'Disables the "btnJ" button
        CurrentLetter = "J" 'Sets the "CurrentLetter" string variable to "J"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnK.Click 'btnK button mouse click event
        btnK.BackColor = Color.DimGray 'Sets the "btnK" button's backcolor to DimGray
        btnK.Enabled = False 'Disables the "btnK" button
        CurrentLetter = "K" 'Sets the "CurrentLetter" string variable to "K"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnL_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnL.Click 'btnL button mouse click event
        btnL.BackColor = Color.DimGray 'Sets the "btnL" button's backcolor to DimGray
        btnL.Enabled = False 'Disables the "btnL" button
        CurrentLetter = "L" 'Sets the "CurrentLetter" string variable to "L"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnM_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnM.Click 'btnM button mouse click event
```

```
        btnM.BackColor = Color.DimGray 'Sets the "btnM" button's backcolor to DimGray
        btnM.Enabled = False 'Disables the "btnM" button
        CurrentLetter = "M" 'Sets the "CurrentLetter" string variable to "M"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnN.Click 'btnN button mouse click event
        btnN.BackColor = Color.DimGray 'Sets the "btnN" button's backcolor to DimGray
        btnN.Enabled = False 'Disables the "btnN" button
        CurrentLetter = "N" 'Sets the "CurrentLetter" string variable to "N"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnO.Click 'btnO button mouse click event
        btnO.BackColor = Color.DimGray 'Sets the "btnO" button's backcolor to DimGray
        btnO.Enabled = False 'Disables the "btnO" button
        CurrentLetter = "O" 'Sets the "CurrentLetter" string variable to "O"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnP.Click 'btnP button mouse click event
        btnP.BackColor = Color.DimGray 'Sets the "btnP" button's backcolor to DimGray
        btnP.Enabled = False 'Disables the "btnP" button
        CurrentLetter = "P" 'Sets the "CurrentLetter" string variable to "P"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnQ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnQ.Click 'btnQ button mouse click event
        btnQ.BackColor = Color.DimGray 'Sets the "btnQ" button's backcolor to DimGray
        btnQ.Enabled = False 'Disables the "btnQ" button
        CurrentLetter = "Q" 'Sets the "CurrentLetter" string variable to "Q"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnR_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnR.Click 'btnR button mouse click event
        btnR.BackColor = Color.DimGray 'Sets the "btnR" button's backcolor to DimGray
        btnR.Enabled = False 'Disables the "btnR" button
        CurrentLetter = "R" 'Sets the "CurrentLetter" string variable to "R"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnS.Click 'btnS button mouse click event
        btnS.BackColor = Color.DimGray 'Sets the "btnS" button's backcolor to DimGray
        btnS.Enabled = False 'Disables the "btnS" button
        CurrentLetter = "S" 'Sets the "CurrentLetter" string variable to "S"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnT_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnT.Click 'btnT button mouse click event
        btnT.BackColor = Color.DimGray 'Sets the "btnT" button's backcolor to DimGray
        btnT.Enabled = False 'Disables the "btnT" button
        CurrentLetter = "T" 'Sets the "CurrentLetter" string variable to "T"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnU_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnU.Click 'btnU button mouse click event
        btnU.BackColor = Color.DimGray 'Sets the "btnU" button's backcolor to DimGray
        btnU.Enabled = False 'Disables the "btnU" button
        CurrentLetter = "U" 'Sets the "CurrentLetter" string variable to "U"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnV_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnV.Click 'btnV button mouse click event
        btnV.BackColor = Color.DimGray 'Sets the "btnV" button's backcolor to DimGray
        btnV.Enabled = False 'Disables the "btnV" button
        CurrentLetter = "V" 'Sets the "CurrentLetter" string variable to "V"
```

```
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnW_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnW.Click 'btnW button mouse click event
        btnW.BackColor = Color.DimGray 'Sets the "btnW" button's backcolor to DimGray
        btnW.Enabled = False 'Disables the "btnW" button
        CurrentLetter = "W" 'Sets the "CurrentLetter" string variable to "W"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnX_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnX.Click 'btnX button mouse click event
        btnX.BackColor = Color.DimGray 'Sets the "btnX" button's backcolor to DimGray
        btnX.Enabled = False 'Disables the "btnX" button
        CurrentLetter = "X" 'Sets the "CurrentLetter" string variable to "X"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnY_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnY.Click 'btnY button mouse click event
        btnY.BackColor = Color.DimGray 'Sets the "btnY" button's backcolor to DimGray
        btnY.Enabled = False 'Disables the "btnY" button
        CurrentLetter = "Y" 'Sets the "CurrentLetter" string variable to "Y"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnZ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnZ.Click 'btnZ button mouse click event
        btnZ.BackColor = Color.DimGray 'Sets the "btnZ" button's backcolor to DimGray
        btnZ.Enabled = False 'Disables the "btnZ" button
        CurrentLetter = "Z" 'Sets the "CurrentLetter" string variable to "Z"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    'Alphabet Sounds
    Private Sub btnA_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnZ.MouseUp, btnY.MouseUp, btnX.MouseUp,
btnW.MouseUp, btnV.MouseUp, btnU.MouseUp, btnT.MouseUp, btnS.MouseUp, btnR.MouseUp,
btnQ.MouseUp, btnP.MouseUp, btnO.MouseUp, btnN.MouseUp, btnM.MouseUp, btnL.MouseUp,
btnK.MouseUp, btnJ.MouseUp, btnI.MouseUp, btnH.MouseUp, btnG.MouseUp, btnF.MouseUp,
btnE.MouseUp, btnD.MouseUp, btnC.MouseUp, btnB.MouseUp, btnA.MouseUp
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_select2
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
    End Sub
    Private Sub LifeFlashTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LifeFlashTimer.Tick 'LifeFlashTimer Timer tick code
        If PlayerFlash = False Then
            If lblPlayerOneLives.ForeColor = Color.Black Then
                lblPlayerOneLives.ForeColor = Color.Red
            Else
                lblPlayerOneLives.ForeColor = Color.Black
            End If
        Else
            If lblPlayerTwoLives.ForeColor = Color.Black Then
                lblPlayerTwoLives.ForeColor = Color.Red
            Else
                lblPlayerTwoLives.ForeColor = Color.Black
            End If
        End If
    End Sub
    Private Sub CountDownTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CountDownTimer.Tick 'CountDownTimer Timer tick code
        CountDownCounter = CountDownCounter - 1
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
```

```
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        If CountDownCounter = 5 Then 'Checks if the CountDownCounter is set to 5
            picGameStatus.Image = My.Resources.Hangman_Four
        End If
        If CountDownCounter = 4 Then 'Checks if the CountDownCounter is set to 4
            picGameStatus.Image = My.Resources.Hangman_Three
        End If
        If CountDownCounter = 3 Then 'Checks if the CountDownCounter is set to 3
            picGameStatus.Image = My.Resources.Hangman_Two
        End If
        If CountDownCounter = 2 Then 'Checks if the CountDownCounter is set to 2
            picGameStatus.Image = My.Resources.Hangman_One
        End If
        If CountDownCounter = 1 Then 'Checks if the CountDownCounter is set to 1
            EnableLetters()  'Calls the EnableLetters subroutine
            picGameStatus.Image = Nothing 'Clears picturebox
            lblGuess.Text = "" 'Clears lblGuess label
            GameStart()  'Calls the GameStart subroutine
            CountDownTimer.Stop() 'Stops CountDownTimer
            CountDownCounter = 6 'Sets CountDownCounter to 6
        End If
    End Sub
    Private Sub frmHangmanTwoPlayer_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmHangmanTwoPlayer form load code
        If My.Settings.DisableCaptions = False Then
            MouseMoveTimer.Start() 'Starts the MouseMoveTimer
        End If
        DisableLetters() 'Calls the DisableLetters subroutine
    End Sub
    Private Sub DisplayTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DisplayTimer.Tick 'DisplayTimer Timer tick code
        GetLives() 'Calls the DisableLetters subroutine
        SwitchPlayers() 'Calls the SwitchPlayers subroutine
        Restart() 'Calls the Restart subroutine
        lblGuess.ForeColor = Color.Black
        picPlayerOneToken.Visible = False
        picPlayerTwoToken.Visible = False
        picGameStatus.Image = Nothing
        DisplayTimer.Stop() 'Stops the "DisplayTimer" timer tick event
    End Sub

    Private Sub WinnerTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles WinnerTimer.Tick 'WinnerTimer Timer tick code
        Restart()
        NewRound()
        lblGuess.ForeColor = Color.Black
        picPlayerOneToken.Visible = False
        picPlayerTwoToken.Visible = False
        picGameStatus.Image = Nothing
        WinnerTimer.Stop() 'Stops the "DisplayTimer" timer tick event
    End Sub
    Dim CaptionCounter As Integer
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString()  'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
```

```
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblLeaveGame.Visible = False 'Hides the "lblLeaveGame" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString()  'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblInfo.Visible = True 'Shows the "lblInfo" label
            lblLeaveGame.Visible = True 'Shows the "lblLeaveGame" label
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.6. Hangman – Versus:

```
Public Class frmHangmanVersus 'frmGameSettings form code
    Dim rand As New Random
    Dim Reader As IO.StreamReader
    Dim EndOfList As Integer
    Dim CurrentLetter As String
    Dim GameTime As Integer = 60
    Dim TokenTime As Integer = 1
    Dim SoundTime As Integer = 2
    Dim Winner As Boolean = False
    Dim Toggle As Boolean = False
    Dim ExitFlag As Boolean = True
    Dim BothLose As Boolean = False
    Dim BeepSound As Boolean = False
    Dim PlayerToken As Boolean = False
    Dim PlayerOneChances As Integer = 7
    Dim PlayerTwoChances As Integer = 7
    Dim CountDownCounter As Integer = 6
    Dim PlayerOneTokenNo As Integer = 0
    Dim PlayerTwoTokenNo As Integer = 0
    Dim CaptionCounter As Integer
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Dim CurrentPlayer As Boolean = True
    Dim NotCurrentPlayer As Boolean = True
    Dim PlayerWithToken As String = "No One"
    Dim appPath As String = Application.StartupPath()
    Dim word As String = ""
    Friend WithEvents beep As New System.Media.SoundPlayer
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
```

```
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Function PlayerOne_GetImage(ByVal bounds As Size) As Image 'PlayerOne_GetImage
Private Function code
        Dim img As New Bitmap(bounds.Width, bounds.Height)
        If CurrentPlayer = True Then 'Checks if current player is player one
            If PlayerOneChances < 0 Then
                img = My.Resources.HangMan_CurrentPlayer_10
            End If
            If PlayerOneChances >= 0 Then
                img = My.Resources.HangMan_CurrentPlayer_10
            End If
            If PlayerOneChances > 1 Then
                img = My.Resources.HangMan_CurrentPlayer_9
            End If
            If PlayerOneChances > 2 Then
                img = My.Resources.HangMan_CurrentPlayer_8
            End If
            If PlayerOneChances > 3 Then
                img = My.Resources.HangMan_CurrentPlayer_7
            End If
            If PlayerOneChances > 4 Then
                img = My.Resources.HangMan_CurrentPlayer_6
            End If
            If PlayerOneChances > 5 Then
                img = My.Resources.HangMan_CurrentPlayer_5
            End If
            If PlayerOneChances > 6 Then
                img = My.Resources.HangMan_CurrentPlayer_4
            End If
            If PlayerOneChances > 7 Then
                img = My.Resources.HangMan_CurrentPlayer_3
            End If
            If PlayerOneChances > 8 Then
                img = My.Resources.HangMan_CurrentPlayer_2
            End If
            If PlayerOneChances > 9 Then
                img = My.Resources.HangMan_CurrentPlayer_1
            End If
            If PlayerOneChances > 10 Then
                img = My.Resources.HangMan_CurrentPlayer_0
            End If
            If PlayerOneChances > 11 Then
                img = My.Resources.HangMan_CurrentPlayer
            End If
        Else
            If PlayerOneChances < 0 Then
                img = My.Resources.HangMan_Player_10
            End If
```

```
            If PlayerOneChances >= 0 Then
                img = My.Resources.HangMan_Player_10
            End If
            If PlayerOneChances > 1 Then
                img = My.Resources.HangMan_Player_9
            End If
            If PlayerOneChances > 2 Then
                img = My.Resources.HangMan_Player_8
            End If
            If PlayerOneChances > 3 Then
                img = My.Resources.HangMan_Player_7
            End If
            If PlayerOneChances > 4 Then
                img = My.Resources.HangMan_Player_6
            End If
            If PlayerOneChances > 5 Then
                img = My.Resources.HangMan_Player_5
            End If
            If PlayerOneChances > 6 Then
                img = My.Resources.HangMan_Player_4
            End If
            If PlayerOneChances > 7 Then
                img = My.Resources.HangMan_Player_3
            End If
            If PlayerOneChances > 8 Then
                img = My.Resources.HangMan_Player_2
            End If
            If PlayerOneChances > 9 Then
                img = My.Resources.HangMan_Player_1
            End If
            If PlayerOneChances > 10 Then
                img = My.Resources.HangMan_Player_0
            End If
            If PlayerOneChances > 11 Then
                img = My.Resources.HangMan_Player
            End If
        End If
        Return img 'Returns the image
    End Function
    Private Function PlayerTwo_GetImage(ByVal bounds As Size) As Image 'PlayerTwo_GetImage
Private Function code
        Dim img As New Bitmap(bounds.Width, bounds.Height)
        If NotCurrentPlayer = True Then 'Checks if the Player Two is the current player
            If PlayerTwoChances < 0 Then
                img = My.Resources.HangMan_CurrentPlayer_10
            End If
            If PlayerTwoChances >= 0 Then
                img = My.Resources.HangMan_CurrentPlayer_10
            End If
            If PlayerTwoChances > 1 Then
                img = My.Resources.HangMan_CurrentPlayer_9
            End If
            If PlayerTwoChances > 2 Then
                img = My.Resources.HangMan_CurrentPlayer_8
            End If
            If PlayerTwoChances > 3 Then
                img = My.Resources.HangMan_CurrentPlayer_7
            End If
            If PlayerTwoChances > 4 Then
                img = My.Resources.HangMan_CurrentPlayer_6
            End If
            If PlayerTwoChances > 5 Then
                img = My.Resources.HangMan_CurrentPlayer_5
            End If
            If PlayerTwoChances > 6 Then
                img = My.Resources.HangMan_CurrentPlayer_4
```

```
            End If
            If PlayerTwoChances > 7 Then
                img = My.Resources.HangMan_CurrentPlayer_3
            End If
            If PlayerTwoChances > 8 Then
                img = My.Resources.HangMan_CurrentPlayer_2
            End If
            If PlayerTwoChances > 9 Then
                img = My.Resources.HangMan_CurrentPlayer_1
            End If
            If PlayerTwoChances > 10 Then
                img = My.Resources.HangMan_CurrentPlayer_0
            End If
            If PlayerTwoChances > 11 Then
                img = My.Resources.HangMan_CurrentPlayer
            End If
        Else
            If PlayerTwoChances < 0 Then
                img = My.Resources.HangMan_Player_10
            End If
            If PlayerTwoChances >= 0 Then
                img = My.Resources.HangMan_Player_10
            End If
            If PlayerTwoChances > 1 Then
                img = My.Resources.HangMan_Player_9
            End If
            If PlayerTwoChances > 2 Then
                img = My.Resources.HangMan_Player_8
            End If
            If PlayerTwoChances > 3 Then
                img = My.Resources.HangMan_Player_7
            End If
            If PlayerTwoChances > 4 Then
                img = My.Resources.HangMan_Player_6
            End If
            If PlayerTwoChances > 5 Then
                img = My.Resources.HangMan_Player_5
            End If
            If PlayerTwoChances > 6 Then
                img = My.Resources.HangMan_Player_4
            End If
            If PlayerTwoChances > 7 Then
                img = My.Resources.HangMan_Player_3
            End If
            If PlayerTwoChances > 8 Then
                img = My.Resources.HangMan_Player_2
            End If
            If PlayerTwoChances > 9 Then
                img = My.Resources.HangMan_Player_1
            End If
            If PlayerTwoChances > 10 Then
                img = My.Resources.HangMan_Player_0
            End If
            If PlayerTwoChances > 11 Then
                img = My.Resources.HangMan_Player
            End If
        End If
        Return img 'Returns the image
    End Function
    Function GuessedLetter(ByVal Letter As String) 'GuessedLetter Function code
        Dim ReturnLetter As Boolean = False
        Dim x As Integer
        For i = 0 To word.Length - 1
            If word.Substring(i, 1) = Letter.ToLower Then
                x = i
                x = x + 1
```

```
                Mid(lblGuess.Text, x, 1) = Letter
                ReturnLetter = True
            End If
        Next i
        Return ReturnLetter
    End Function
    Private Sub Guess() 'Guess Private Sub code
        lblGameTime.Focus() 'Focuses on the label
        PicPlayerOne.BackgroundImage = Nothing 'Clears the Image
        PicPlayerTwo.BackgroundImage = Nothing 'Clears the Image
        If Toggle = False Then 'Checks if Toggle is set to false
            If GuessedLetter(CurrentLetter) = False Then
                PlayerOneChances = PlayerOneChances - 1
                CurrentPlayer = False
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
                NotCurrentPlayer = True
                Toggle = True 'Set's "Toggle" to true
                PicPlayerTwo.BackgroundImage = PlayerTwo_GetImage(PicPlayerTwo.Size)
            Else
                CurrentPlayer = True
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
                NotCurrentPlayer = False
                Toggle = False
                PicPlayerTwo.BackgroundImage = PlayerTwo_GetImage(PicPlayerTwo.Size)
            End If
            If word.ToUpper = lblGuess.Text.ToUpper Then
                If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
                    If My.Settings.Mute = False Then 'Checks if "Mute" variable in
application settings is set to false
                        frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                        frmMainMenu.player.Play() 'Plays the sound file
                    End If
                    lblGuess.ForeColor = Color.Lime
                    lblGuess.Text = word.ToUpper
                End If
                PlayerOneTokenNo = PlayerOneTokenNo + 1
                If PlayerOneTokenNo > 4 Then
                    GetTokens() 'Calls the "GetTokensRestart" subroutine
                    StopTimers() 'Calls the "StopTimers" subroutine
                    PlayerOneTokenNo = 0 'Sets "PlayerOneTokenNo" to 0
                    PlayerTwoTokenNo = 0 'Sets "PlayerTwoTokenNo" to 0
                    Restart() 'Calls the "Restart" subroutine
                    DisableLetters() 'Calls the "DisableLetters" subroutine
                    DisableButtons() 'Calls the "DisableButtons" subroutine
                    ButtonEnableTimer.Start() 'Starts ButtonEnableTimer
                    Winner = True 'Set's "Winner" to true
                    WinnerSoundTimer.Start() 'Starts WinnerSoundTimer
                    PicPlayerOne.Image = My.Resources.Hangman_Player_One_Winner
                    PicPlayerTwo.Image = My.Resources.Hangman_Player_Two_Loser
                    Exit Sub
                End If
                PlayerOneChances = 7 'Sets "PlayerOneChances" to 7
                PlayerTwoChances = 7 'Sets "PlayerTwoChances" to 7
                PlayerToken = True
                GetTokens() 'Calls the "GetTokensRestart" subroutine
                StopTimers() 'Calls the "StopTimers" subroutine
                DisableLetters() 'Calls the "DisableLetters" subroutine
                WinnerSoundTimer.Start() 'Starts WinnerSoundTimer
                PicPlayerOne.Image = My.Resources.Hangman_Token
                PlayerWithToken = "Player 1"
                TokenTimer.Start() 'Starts TokenTimer
                Exit Sub
            End If
        Else
            If GuessedLetter(CurrentLetter) = False Then
                PlayerTwoChances = PlayerTwoChances - 1
```

```
                CurrentPlayer = True  'Set's "CurrentPlayer" to true
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
'Retrieves the image from the function
                Toggle = False  'Set's "Toggle" to False
                NotCurrentPlayer = False  'Set's "NotCurrentPlayer" to False
                PicPlayerTwo.BackgroundImage = PlayerTwo_GetImage(PicPlayerTwo.Size)
'Retrieves the image from the function
            Else
                CurrentPlayer = False 'Set's "CurrentPlayer" to False
                PicPlayerOne.BackgroundImage = PlayerOne_GetImage(PicPlayerOne.Size)
'Retrieves the image from the function
                Toggle = True 'Set's "Toggle" to true
                NotCurrentPlayer = True 'Set's "NotCurrentPlayer" to true
                PicPlayerTwo.BackgroundImage = PlayerTwo_GetImage(PicPlayerTwo.Size)
'Retrieves the image from the function
            End If
            If word.ToUpper = lblGuess.Text.ToUpper Then 'Checks if the player got the word
correct
                If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
                    If My.Settings.Mute = False Then 'Checks if "Mute" variable in
application settings is set to false
                        frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                        frmMainMenu.player.Play() 'Plays the sound file
                    End If
                    lblGuess.ForeColor = Color.Lime 'Changes the color of the label
                    lblGuess.Text = word.ToUpper 'Displays the word in the label
                End If
                PlayerTwoTokenNo = PlayerTwoTokenNo + 1 'Increments PlayerTwoTokenNo
                If PlayerTwoTokenNo > 4 Then 'Checks if PlayerTwoTokenNo is greater than 4
                    StopTimers() 'Calls the "StopTimers" subroutine
                    GetTokens() 'Calls the "GetTokens" subroutine
                    PlayerOneTokenNo = 0 'Sets PlayerOneTokenNo to 0
                    PlayerTwoTokenNo = 0 'Sets PlayerTwoTokenNo to 0
                    Restart() 'Calls the "Restart" subroutine
                    DisableLetters() 'Calls the "DisableLetters" subroutine
                    DisableButtons() 'Calls the "DisableButtons" subroutine
                    ButtonEnableTimer.Start()
                    Winner = True 'Set's "Winner" to true
                    WinnerSoundTimer.Start()
                    PicPlayerOne.Image = My.Resources.Hangman_Player_One_loser
                    PicPlayerTwo.Image = My.Resources.Hangman_Player_Two_Winner
                    Exit Sub
                End If
                PlayerOneChances = 7 'Sets "PlayerOneChances" to 7
                PlayerTwoChances = 7 'Sets "PlayerTwoChances" to 7
                PlayerToken = True 'Set's "PlayerToken" to true
                GetTokens()  'Calls the "GetTokens" subroutine
                StopTimers()  'Calls the "StopTimers" subroutine
                DisableLetters()  'Calls the "DisableLetters" subroutine
                WinnerSoundTimer.Start()
                PicPlayerTwo.Image = My.Resources.Hangman_Token
                PlayerWithToken = "Player 2"
                TokenTimer.Start()
                Exit Sub
            End If
        End If
        If PlayerOneChances = 1 And PlayerTwoChances = 1 Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            PlayerOneChances = 7 'Sets "PlayerOneChances" to 7
            PlayerTwoChances = 7 'Sets "PlayerTwoChances" to 7
            BothLose = True 'Set's "BothLose" to true
            StopTimers()  'Calls the "StopTimers" subroutine
```

```
            DisableLetters()  'Calls the "DisableLetters" subroutine
            WinnerSoundTimer.Start()
            picWrongArrow.Visible = True 'Shows the picturebox
            lblGuess.ForeColor = Color.Maroon 'Changes the label's color to maroon
            lblGuess.Text = word.ToUpper 'Changes the label's case to upper
            TokenTimer.Start() 'Starts TokenTimer
            Exit Sub 'Exits subroutine
        End If
    End Sub
    Private Sub GetTokens()
        If PlayerOneTokenNo > 0 Then
            Player1_GoldToken1.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerOneTokenNo > 1 Then
            Player1_GoldToken2.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerOneTokenNo > 2 Then
            Player1_GoldToken3.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerOneTokenNo > 3 Then
            Player1_GoldToken4.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerOneTokenNo > 4 Then
            Player1_GoldToken5.BackgroundImage = My.Resources.Gold_Token
        End If

        If PlayerTwoTokenNo > 0 Then
            Player2_GoldToken1.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerTwoTokenNo > 1 Then
            Player2_GoldToken2.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerTwoTokenNo > 2 Then
            Player2_GoldToken3.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerTwoTokenNo > 3 Then
            Player2_GoldToken4.BackgroundImage = My.Resources.Gold_Token
        End If
        If PlayerTwoTokenNo > 4 Then
            Player2_GoldToken5.BackgroundImage = My.Resources.Gold_Token
        End If
    End Sub
    Private Sub DisableButtons() 'DisableButtons subroutine code
        btnStart.Enabled = False
        btnReturn.Enabled = False
        btnInfo.Enabled = False
        btnClose.Enabled = False
        btnMinimize.Enabled = False
    End Sub
    Private Sub EnableLetters() 'EnableLetters subroutine code
        'Changes back color of all buttons to Silver
        btnA.BackColor = Color.Silver
        btnB.BackColor = Color.Silver
        btnC.BackColor = Color.Silver
        btnD.BackColor = Color.Silver
        btnE.BackColor = Color.Silver
        btnF.BackColor = Color.Silver
        btnG.BackColor = Color.Silver
        btnH.BackColor = Color.Silver
        btnI.BackColor = Color.Silver
        btnJ.BackColor = Color.Silver
        btnK.BackColor = Color.Silver
        btnL.BackColor = Color.Silver
        btnM.BackColor = Color.Silver
        btnN.BackColor = Color.Silver
        btnO.BackColor = Color.Silver
```

```
        btnP.BackColor = Color.Silver
        btnQ.BackColor = Color.Silver
        btnR.BackColor = Color.Silver
        btnS.BackColor = Color.Silver
        btnT.BackColor = Color.Silver
        btnU.BackColor = Color.Silver
        btnV.BackColor = Color.Silver
        btnW.BackColor = Color.Silver
        btnX.BackColor = Color.Silver
        btnY.BackColor = Color.Silver
        btnZ.BackColor = Color.Silver
        'Re-enables all of the buttons
        btnA.Enabled = True
        btnB.Enabled = True
        btnC.Enabled = True
        btnD.Enabled = True
        btnE.Enabled = True
        btnF.Enabled = True
        btnG.Enabled = True
        btnH.Enabled = True
        btnI.Enabled = True
        btnJ.Enabled = True
        btnK.Enabled = True
        btnL.Enabled = True
        btnM.Enabled = True
        btnN.Enabled = True
        btnO.Enabled = True
        btnP.Enabled = True
        btnQ.Enabled = True
        btnR.Enabled = True
        btnS.Enabled = True
        btnT.Enabled = True
        btnU.Enabled = True
        btnV.Enabled = True
        btnW.Enabled = True
        btnX.Enabled = True
        btnY.Enabled = True
        btnZ.Enabled = True
    End Sub
    Private Sub DisableLetters() 'DisableLetters subroutine code
        'Changes back color of all buttons to DimGray
        btnA.BackColor = Color.DimGray
        btnB.BackColor = Color.DimGray
        btnC.BackColor = Color.DimGray
        btnD.BackColor = Color.DimGray
        btnE.BackColor = Color.DimGray
        btnF.BackColor = Color.DimGray
        btnG.BackColor = Color.DimGray
        btnH.BackColor = Color.DimGray
        btnI.BackColor = Color.DimGray
        btnJ.BackColor = Color.DimGray
        btnK.BackColor = Color.DimGray
        btnL.BackColor = Color.DimGray
        btnM.BackColor = Color.DimGray
        btnN.BackColor = Color.DimGray
        btnO.BackColor = Color.DimGray
        btnP.BackColor = Color.DimGray
        btnQ.BackColor = Color.DimGray
        btnR.BackColor = Color.DimGray
        btnS.BackColor = Color.DimGray
        btnT.BackColor = Color.DimGray
        btnU.BackColor = Color.DimGray
        btnV.BackColor = Color.DimGray
        btnW.BackColor = Color.DimGray
        btnX.BackColor = Color.DimGray
        btnY.BackColor = Color.DimGray
```

```
        btnZ.BackColor = Color.DimGray
        'Disables all of the buttons
        btnA.Enabled = False
        btnB.Enabled = False
        btnC.Enabled = False
        btnD.Enabled = False
        btnE.Enabled = False
        btnF.Enabled = False
        btnG.Enabled = False
        btnH.Enabled = False
        btnI.Enabled = False
        btnJ.Enabled = False
        btnK.Enabled = False
        btnL.Enabled = False
        btnM.Enabled = False
        btnN.Enabled = False
        btnO.Enabled = False
        btnP.Enabled = False
        btnQ.Enabled = False
        btnR.Enabled = False
        btnS.Enabled = False
        btnT.Enabled = False
        btnU.Enabled = False
        btnV.Enabled = False
        btnW.Enabled = False
        btnX.Enabled = False
        btnY.Enabled = False
        btnZ.Enabled = False
    End Sub
    Private Sub ClearTokens() 'ClearTokens subroutine code
        'Obtains the image from resources for pictureboxes below
        Player1_GoldToken1.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken2.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken3.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken4.BackgroundImage = My.Resources.No_Token
        Player1_GoldToken5.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken1.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken2.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken3.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken4.BackgroundImage = My.Resources.No_Token
        Player2_GoldToken5.BackgroundImage = My.Resources.No_Token
    End Sub
    Private Sub Restart() 'Restart subroutine code
        PicPlayerOne.Image = Nothing 'Clears picture box
        PicPlayerTwo.Image = Nothing 'Clears picture box
        PicPlayerOne.BackgroundImage = My.Resources.HangMan_Player
        PicPlayerTwo.BackgroundImage = My.Resources.HangMan_Player
        lblGameTime.ForeColor = Color.Black 'Changes label's forecolor to black
        lblGameTime.Text = "01:00" 'Changes label's text
        lblGuess.Text = "Getting Word..." 'Changes label's text
        ClearTokens() 'Calls the "ClearTokens" subroutine
        GetTokens() 'Calls the "GetTokens" subroutine
        StopTimers() 'Calls the "StopTimers" subroutine
        DisableLetters() 'Calls the "DisableLetters" subroutine
        btnStart.Visible = True 'Shows start button
        lblGuess.Visible = False 'Hides guess label
    End Sub
    Private Sub GameStart() 'GameStart subroutine code
        BeepSound = False 'Set's "BeepSound" variable to false
        picWrongArrow.Visible = False 'Hides picturebox
        lblGuess.ForeColor = Color.Black 'Changes label's forecolor to black
        EnableLetters() 'Calls the "EnableLetters" subroutine
        EndOfList = lstWords.Items.Count
        GetTokens() 'Calls the "GetTokens" subroutine
        PicPlayerOne.Image = Nothing
        PicPlayerTwo.Image = Nothing
```

```
        PicPlayerOne.BackgroundImage = My.Resources.HangMan_CurrentPlayer_4
        PicPlayerTwo.BackgroundImage = My.Resources.HangMan_Player_4
        GameTimer.Start() 'Starts GameTimer
        GameTime = 60 'Sets GameTime to 60
        lblGameTime.Text = "01:00" 'Changes label's text
        lblGuess.Text = "" 'Clears label's text
        word = lstWords.Items.Item(rand.Next(0, EndOfList)) 'Gets next random word
        word = word.ToLower 'Sets the random word to lowercase
        For i = 0 To word.Length - 1 'Executes a set of commands a certain amount of times
            If word(i) = " " Then 'Checks for spaces in the word
                lblGuess.Text &= " " 'Changes label's text
            Else
                lblGuess.Text &= "-" 'Changes label's text
            End If
        Next i
        PlayerOneChances = 7 'Sets "PlayerOneChances" to 7
        PlayerTwoChances = 7 'Sets "PlayerOneChances" to 7
        CurrentPlayer = True 'Set's "CurrentPlayer" to True
        NotCurrentPlayer = True 'Set's "NotCurrentPlayer" to True
        CurrentLetter = "" 'Clears label's text
    End Sub
    Private Sub StopTimers() 'StopTimers subroutine code
        GameTimer.Stop() 'Stops the "GameTimer"
        BeepTimer.Stop() 'Stops the "BeepTimer"
        ColorTimer.Stop() 'Stops the "ColorTimer"
        CountDownTimer.Stop() 'Stops the "CountDownTimer"
        TokenTimer.Stop() 'Stops the "TokenTimer"
        WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer"
        ButtonEnableTimer.Stop() 'Stops the "ButtonEnableTimer"
        lblGameTime.ForeColor = Color.Black 'Changes label's forecolor to black
    End Sub

    Private Sub frmHangmanVersus_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles Me.KeyPress 'frmHangmanVersus Keypress event
code
        If Asc(e.KeyChar) = 13 And btnStart.Visible = True Then 'Checks if "enter key" has
been pushed
            btnStart_MouseUp(Nothing, Nothing) 'Calls the btnstart button's MouseUp event
        End If
        If (Asc(e.KeyChar) = 65 Or Asc(e.KeyChar) = 97) And btnA.Enabled = True Then 'Checks
if the button pressed on the keyboard is either "A" or "a" and checks if the button is
enabled
            btnA_Click(Nothing, Nothing) 'Calls the "btnA_Click" event
        ElseIf (Asc(e.KeyChar) = 66 Or Asc(e.KeyChar) = 98) And btnB.Enabled = True Then
'Checks if the button pressed on the keyboard is either "B" or "b" and checks if the button
is enabled
            btnB_Click(Nothing, Nothing) 'Calls the "btnB_Click" event
        ElseIf (Asc(e.KeyChar) = 67 Or Asc(e.KeyChar) = 99) And btnC.Enabled = True Then
'Checks if the button pressed on the keyboard is either "C" or "c" and checks if the button
is enabled
            btnC_Click(Nothing, Nothing) 'Calls the "btnC_Click" event
        ElseIf (Asc(e.KeyChar) = 68 Or Asc(e.KeyChar) = 100) And btnD.Enabled = True Then
'Checks if the button pressed on the keyboard is either "D" or "d" and checks if the button
is enabled
            btnD_Click(Nothing, Nothing) 'Calls the "btnD_Click" event
        ElseIf (Asc(e.KeyChar) = 69 Or Asc(e.KeyChar) = 101) And btnE.Enabled = True Then
'Checks if the button pressed on the keyboard is either "E" or "e" and checks if the button
is enabled
            btnE_Click(Nothing, Nothing) 'Calls the "btnE_Click" event
        ElseIf (Asc(e.KeyChar) = 70 Or Asc(e.KeyChar) = 102) And btnF.Enabled = True Then
'Checks if the button pressed on the keyboard is either "F" or "f" and checks if the button
is enabled
            btnF_Click(Nothing, Nothing) 'Calls the "btnF_Click" event
        ElseIf (Asc(e.KeyChar) = 71 Or Asc(e.KeyChar) = 103) And btnG.Enabled = True Then
'Checks if the button pressed on the keyboard is either "G" or "g" and checks if the button
is enabled
```

```
            btnG_Click(Nothing, Nothing) 'Calls the "btnG_Click" event
        ElseIf (Asc(e.KeyChar) = 72 Or Asc(e.KeyChar) = 104) And btnH.Enabled = True Then
'Checks if the button pressed on the keyboard is either "H" or "h" and checks if the button
is enabled
            btnH_Click(Nothing, Nothing) 'Calls the "btnH_Click" event
        ElseIf (Asc(e.KeyChar) = 73 Or Asc(e.KeyChar) = 105) And btnI.Enabled = True Then
'Checks if the button pressed on the keyboard is either "I" or "i" and checks if the button
is enabled
            btnI_Click(Nothing, Nothing) 'Calls the "btnI_Click" event
        ElseIf (Asc(e.KeyChar) = 74 Or Asc(e.KeyChar) = 106) And btnJ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "J" or "j" and checks if the button
is enabled
            btnJ_Click(Nothing, Nothing) 'Calls the "btnJ_Click" event
        ElseIf (Asc(e.KeyChar) = 75 Or Asc(e.KeyChar) = 107) And btnK.Enabled = True Then
'Checks if the button pressed on the keyboard is either "K" or "k" and checks if the button
is enabled
            btnK_Click(Nothing, Nothing) 'Calls the "btnK_Click" event
        ElseIf (Asc(e.KeyChar) = 76 Or Asc(e.KeyChar) = 108) And btnL.Enabled = True Then
'Checks if the button pressed on the keyboard is either "L" or "l" and checks if the button
is enabled
            btnL_Click(Nothing, Nothing) 'Calls the "btnL_Click" event
        ElseIf (Asc(e.KeyChar) = 77 Or Asc(e.KeyChar) = 109) And btnM.Enabled = True Then
'Checks if the button pressed on the keyboard is either "M" or "m" and checks if the button
is enabled
            btnM_Click(Nothing, Nothing) 'Calls the "btnM_Click" event
        ElseIf (Asc(e.KeyChar) = 78 Or Asc(e.KeyChar) = 110) And btnN.Enabled = True Then
'Checks if the button pressed on the keyboard is either "N" or "n" and checks if the button
is enabled
            btnN_Click(Nothing, Nothing) 'Calls the "btnN_Click" event
        ElseIf (Asc(e.KeyChar) = 79 Or Asc(e.KeyChar) = 111) And btnO.Enabled = True Then
'Checks if the button pressed on the keyboard is either "O" or "o" and checks if the button
is enabled
            btnO_Click(Nothing, Nothing) 'Calls the "btnO_Click" event
        ElseIf (Asc(e.KeyChar) = 80 Or Asc(e.KeyChar) = 112) And btnP.Enabled = True Then
'Checks if the button pressed on the keyboard is either "P" or "p" and checks if the button
is enabled
            btnP_Click(Nothing, Nothing) 'Calls the "btnP_Click" event
        ElseIf (Asc(e.KeyChar) = 81 Or Asc(e.KeyChar) = 113) And btnQ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Q" or "q" and checks if the button
is enabled
            btnQ_Click(Nothing, Nothing) 'Calls the "btnQ_Click" event
        ElseIf (Asc(e.KeyChar) = 82 Or Asc(e.KeyChar) = 114) And btnR.Enabled = True Then
'Checks if the button pressed on the keyboard is either "R" or "r" and checks if the button
is enabled
            btnR_Click(Nothing, Nothing) 'Calls the "btnR_Click" event
        ElseIf (Asc(e.KeyChar) = 83 Or Asc(e.KeyChar) = 115) And btnS.Enabled = True Then
'Checks if the button pressed on the keyboard is either "S" or "s" and checks if the button
is enabled
            btnS_Click(Nothing, Nothing) 'Calls the "btnS_Click" event
        ElseIf (Asc(e.KeyChar) = 84 Or Asc(e.KeyChar) = 116) And btnT.Enabled = True Then
'Checks if the button pressed on the keyboard is either "T" or "t" and checks if the button
is enabled
            btnT_Click(Nothing, Nothing) 'Calls the "btnT_Click" event
        ElseIf (Asc(e.KeyChar) = 85 Or Asc(e.KeyChar) = 117) And btnU.Enabled = True Then
'Checks if the button pressed on the keyboard is either "U" or "u" and checks if the button
is enabled
            btnU_Click(Nothing, Nothing) 'Calls the "btnU_Click" event
        ElseIf (Asc(e.KeyChar) = 86 Or Asc(e.KeyChar) = 118) And btnV.Enabled = True Then
'Checks if the button pressed on the keyboard is either "V" or "v" and checks if the button
is enabled
            btnV_Click(Nothing, Nothing) 'Calls the "btnV_Click" event
        ElseIf (Asc(e.KeyChar) = 87 Or Asc(e.KeyChar) = 119) And btnW.Enabled = True Then
'Checks if the button pressed on the keyboard is either "W" or "w" and checks if the button
is enabled
            btnW_Click(Nothing, Nothing) 'Calls the "btnW_Click" event
```

```
        ElseIf (Asc(e.KeyChar) = 88 Or Asc(e.KeyChar) = 120) And btnX.Enabled = True Then
'Checks if the button pressed on the keyboard is either "X" or "x" and checks if the button
is enabled
            btnX_Click(Nothing, Nothing) 'Calls the "btnX_Click" event
        ElseIf (Asc(e.KeyChar) = 89 Or Asc(e.KeyChar) = 121) And btnY.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Y" or "y" and checks if the button
is enabled
            btnY_Click(Nothing, Nothing) 'Calls the "btnY_Click" event
        ElseIf (Asc(e.KeyChar) = 90 Or Asc(e.KeyChar) = 122) And btnZ.Enabled = True Then
'Checks if the button pressed on the keyboard is either "Z" or "z" and checks if the button
is enabled
            btnZ_Click(Nothing, Nothing) 'Calls the "btnZ_Click" event
        End If
    End Sub
    Private Sub frmHangmanVersus_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmHangmanVersus form load event code
        If My.Settings.DisableCaptions = False Then 'Checks if the application setting's
boolean variable "DisableCaptions" is set to False
            MouseMoveTimer.Start() 'Starts MouseMoveTimer
        End If
        DisableLetters() 'Calls the "DisableLetters" subroutine
        If My.Settings.HangmanDefaultGameList = False Then 'Checks if the application
setting's boolean variable "HangmanDefaultGameList" is set to False
            Dim rand As New Random
            Reader = New IO.StreamReader(appPath & "\CustomWordList.txt") 'Streams the reader
in the directory of "CustomWordList.txt"
            While (Reader.Peek() > -1)
                lstWords.Items.Add(Reader.ReadLine) 'Adds the readline of the text document
to the listbox
            End While
            Reader.Close()
        Else
            Dim rand As New Random
            Reader = New IO.StreamReader(appPath & "\DefaultWordList.txt") 'Streams the
reader in the directory of "DefaultWordList.txt"
            While (Reader.Peek() > -1)
                lstWords.Items.Add(Reader.ReadLine) 'Adds the readline of the text document
to the listbox
            End While
            Reader.Close() 'Closes the reader
        End If
        EndOfList = lstWords.Items.Count 'Sets the end of the list to the mount of items in
the listbox
    End Sub
    Private Sub btnStart_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseDown 'btnStart Button MouseDown
code
        btnStart.BackgroundImage = My.Resources.Start_Button_Pushed 'Changes the background
image of the "btnStart" button when the mouse is down
    End Sub
    Private Sub btnStart_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseEnter 'btnStart Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.BackgroundImage = My.Resources.Start_Button_Highlighted 'Changes the
background image of the "btnStart" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnStart_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseLeave 'btnStart Button MouseLeave code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
    End Sub
```

```
    Private Sub btnStart_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseUp 'btnStart Button MouseUp code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
        PicPlayerOne.BackgroundImage = My.Resources.HangMan_Player
        PicPlayerTwo.BackgroundImage = My.Resources.HangMan_Player
        CountDownTimer.Start()
        PicPlayerOne.Image = My.Resources.Hangman_Five
        PicPlayerTwo.Image = My.Resources.Hangman_Five
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.Visible = False
        lblGuess.Visible = True
        ExitFlag = False
    End Sub
    'Close
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    'Minimize
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed 'Changes the
"btnMinimize" picturebox's background to another image in resources
    End Sub
```

```
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted 'Changes the
"btnMinimize" picturebox's background to another image in resources
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
picturebox's background to another image in resources
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
picturebox's background to another image in resources
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        Me.WindowState = FormWindowState.Minimized 'Minimizes current form
    End Sub
    'Info
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        lblInfo.Visible = True
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo"
picturebox's background to another image in resources
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo"
picturebox's background to another image in resources
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
```

```
                frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        Dim ProcessDirectory As String = appPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        lblLeaveGame.Visible = True
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted 'Changes the
"btnReturn" picturebox's background to another image in resources
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblLeaveGame.Visible = False
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        If ExitFlag = True Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Restart()
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
            Exit Sub
        End If
        Dim MessageBoxResult As String
        MessageBoxResult = MsgBox("Are you sure you wish to exit the game?", vbYesNo, "Exit
Game")
        If MessageBoxResult = vbYes Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Restart()
```

```
            ExitFlag = True
            frmHangmanModeMenu.Location = New Point(Me.Location.X, Me.Location.Y) 'Sets the
"frmHangmanModeMenu" form's location to the current form's location
            frmHangmanModeMenu.Show() 'Shows the "frmHangmanModeMenu" form
            Me.Dispose() 'Closes the current form
        End If
    End Sub

    'Alphabet
    Private Sub btnA_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnA.Click 'btnA button mouse click event
        btnA.BackColor = Color.DimGray 'Sets the "btnA" button's backcolor to DimGray
        btnA.Enabled = False 'Disables the "btnA" button
        CurrentLetter = "A" 'Sets the "CurrentLetter" string variable to "A"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnB_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnB.Click 'btnB button mouse click event
        btnB.BackColor = Color.DimGray 'Sets the "btnB" button's backcolor to DimGray
        btnB.Enabled = False 'Disables the "btnB" button
        CurrentLetter = "B" 'Sets the "CurrentLetter" string variable to "B"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnC_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnC.Click 'btnC button mouse click event
        btnC.BackColor = Color.DimGray 'Sets the "btnC" button's backcolor to DimGray
        btnC.Enabled = False 'Disables the "btnC" button
        CurrentLetter = "C" 'Sets the "CurrentLetter" string variable to "C"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnD_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnD.Click 'btnD button mouse click event
        btnD.BackColor = Color.DimGray 'Sets the "btnD" button's backcolor to DimGray
        btnD.Enabled = False 'Disables the "btnD" button
        CurrentLetter = "D" 'Sets the "CurrentLetter" string variable to "D"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnE_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnE.Click 'btnE button mouse click event
        btnE.BackColor = Color.DimGray 'Sets the "btnE" button's backcolor to DimGray
        btnE.Enabled = False 'Disables the "btnE" button
        CurrentLetter = "E" 'Sets the "CurrentLetter" string variable to "E"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnF.Click 'btnF button mouse click event
        btnF.BackColor = Color.DimGray 'Sets the "btnF" button's backcolor to DimGray
        btnF.Enabled = False 'Disables the "btnF" button
        CurrentLetter = "F" 'Sets the "CurrentLetter" string variable to "F"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnG_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnG.Click 'btnG button mouse click event
        btnG.BackColor = Color.DimGray 'Sets the "btnG" button's backcolor to DimGray
        btnG.Enabled = False 'Disables the "btnG" button
        CurrentLetter = "G" 'Sets the "CurrentLetter" string variable to "G"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnH_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnH.Click 'btnH button mouse click event
        btnH.BackColor = Color.DimGray 'Sets the "btnH" button's backcolor to DimGray
        btnH.Enabled = False 'Disables the "btnH" button
        CurrentLetter = "H" 'Sets the "CurrentLetter" string variable to "H"
        Guess() 'Calls the "Guess" subroutine
    End Sub
```

```
    Private Sub btnI_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnI.Click 'btnI button mouse click event
        btnI.BackColor = Color.DimGray 'Sets the "btnI" button's backcolor to DimGray
        btnI.Enabled = False 'Disables the "btnI" button
        CurrentLetter = "I" 'Sets the "CurrentLetter" string variable to "I"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnJ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnJ.Click 'btnJ button mouse click event
        btnJ.BackColor = Color.DimGray 'Sets the "btnJ" button's backcolor to DimGray
        btnJ.Enabled = False 'Disables the "btnJ" button
        CurrentLetter = "J" 'Sets the "CurrentLetter" string variable to "J"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnK.Click 'btnK button mouse click event
        btnK.BackColor = Color.DimGray 'Sets the "btnK" button's backcolor to DimGray
        btnK.Enabled = False 'Disables the "btnK" button
        CurrentLetter = "K" 'Sets the "CurrentLetter" string variable to "K"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnL_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnL.Click 'btnL button mouse click event
        btnL.BackColor = Color.DimGray 'Sets the "btnL" button's backcolor to DimGray
        btnL.Enabled = False 'Disables the "btnL" button
        CurrentLetter = "L" 'Sets the "CurrentLetter" string variable to "L"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnM_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnM.Click 'btnM button mouse click event
        btnM.BackColor = Color.DimGray 'Sets the "btnM" button's backcolor to DimGray
        btnM.Enabled = False 'Disables the "btnM" button
        CurrentLetter = "M" 'Sets the "CurrentLetter" string variable to "M"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnN.Click 'btnN button mouse click event
        btnN.BackColor = Color.DimGray 'Sets the "btnN" button's backcolor to DimGray
        btnN.Enabled = False 'Disables the "btnN" button
        CurrentLetter = "N" 'Sets the "CurrentLetter" string variable to "N"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnO.Click 'btnO button mouse click event
        btnO.BackColor = Color.DimGray 'Sets the "btnO" button's backcolor to DimGray
        btnO.Enabled = False 'Disables the "btnO" button
        CurrentLetter = "O" 'Sets the "CurrentLetter" string variable to "O"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnP.Click 'btnP button mouse click event
        btnP.BackColor = Color.DimGray 'Sets the "btnP" button's backcolor to DimGray
        btnP.Enabled = False 'Disables the "btnP" button
        CurrentLetter = "P" 'Sets the "CurrentLetter" string variable to "P"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnQ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnQ.Click 'btnQ button mouse click event
        btnQ.BackColor = Color.DimGray 'Sets the "btnQ" button's backcolor to DimGray
        btnQ.Enabled = False 'Disables the "btnQ" button
        CurrentLetter = "Q" 'Sets the "CurrentLetter" string variable to "Q"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnR_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnR.Click 'btnR button mouse click event
        btnR.BackColor = Color.DimGray 'Sets the "btnR" button's backcolor to DimGray
```

```
        btnR.Enabled = False 'Disables the "btnR" button
        CurrentLetter = "R" 'Sets the "CurrentLetter" string variable to "R"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnS.Click 'btnS button mouse click event
        btnS.BackColor = Color.DimGray 'Sets the "btnS" button's backcolor to DimGray
        btnS.Enabled = False 'Disables the "btnS" button
        CurrentLetter = "S" 'Sets the "CurrentLetter" string variable to "S"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnT_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnT.Click 'btnT button mouse click event
        btnT.BackColor = Color.DimGray 'Sets the "btnT" button's backcolor to DimGray
        btnT.Enabled = False 'Disables the "btnT" button
        CurrentLetter = "T" 'Sets the "CurrentLetter" string variable to "T"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnU_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnU.Click 'btnU button mouse click event
        btnU.BackColor = Color.DimGray 'Sets the "btnU" button's backcolor to DimGray
        btnU.Enabled = False 'Disables the "btnU" button
        CurrentLetter = "U" 'Sets the "CurrentLetter" string variable to "U"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnV_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnV.Click 'btnV button mouse click event
        btnV.BackColor = Color.DimGray 'Sets the "btnV" button's backcolor to DimGray
        btnV.Enabled = False 'Disables the "btnV" button
        CurrentLetter = "V" 'Sets the "CurrentLetter" string variable to "V"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnW_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnW.Click 'btnW button mouse click event
        btnW.BackColor = Color.DimGray 'Sets the "btnW" button's backcolor to DimGray
        btnW.Enabled = False 'Disables the "btnW" button
        CurrentLetter = "W" 'Sets the "CurrentLetter" string variable to "W"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnX_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnX.Click 'btnX button mouse click event
        btnX.BackColor = Color.DimGray 'Sets the "btnX" button's backcolor to DimGray
        btnX.Enabled = False 'Disables the "btnX" button
        CurrentLetter = "X" 'Sets the "CurrentLetter" string variable to "X"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnY_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnY.Click 'btnY button mouse click event
        btnY.BackColor = Color.DimGray 'Sets the "btnY" button's backcolor to DimGray
        btnY.Enabled = False 'Disables the "btnY" button
        CurrentLetter = "Y" 'Sets the "CurrentLetter" string variable to "Y"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnZ_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnZ.Click 'btnZ button mouse click event
        btnZ.BackColor = Color.DimGray 'Sets the "btnZ" button's backcolor to DimGray
        btnZ.Enabled = False 'Disables the "btnZ" button
        CurrentLetter = "Z" 'Sets the "CurrentLetter" string variable to "Z"
        Guess() 'Calls the "Guess" subroutine
    End Sub
    Private Sub btnA_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnZ.MouseUp, btnY.MouseUp, btnX.MouseUp,
btnW.MouseUp, btnV.MouseUp, btnU.MouseUp, btnT.MouseUp, btnS.MouseUp, btnR.MouseUp,
btnQ.MouseUp, btnP.MouseUp, btnO.MouseUp, btnN.MouseUp, btnM.MouseUp, btnL.MouseUp,
btnK.MouseUp, btnJ.MouseUp, btnI.MouseUp, btnH.MouseUp, btnG.MouseUp, btnF.MouseUp,
btnE.MouseUp, btnD.MouseUp, btnC.MouseUp, btnB.MouseUp, btnA.MouseUp
```

```
        If BeepSound = False Then 'Checks if "BeepSound" variable is set to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_select2
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
    End Sub
    Private Sub GameTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles GameTimer.Tick
        GameTime = GameTime - 1
        lblGameTime.Text = GameTime.ToString("00:00")
        If GameTime = 30 Then
            BeepTimer.Start()
        End If
        If GameTime = 29 Then
            BeepSound = True 'Sets "BeepSound" variable to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                beep.Stream = My.Resources.wagermatchend_converted
                beep.Play()
            End If
        End If
        If GameTime < 1 Then
            BeepTimer.Stop() 'Stops BeepTimer
            GameTimer.Stop() 'Stops GameTimer
        End If
        If GameTime = 0 Then
            BeepSound = False 'Sets "BeepSound" variable to false
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            PlayerOneChances = 7 'Sets PlayerOneChances to 7
            PlayerTwoChances = 7 'Sets PlayerTwoChances to 7
            BothLose = True 'Sets the boolean variable BothLose to True
            StopTimers() 'Calls the StopTimers subroutine
            DisableLetters() 'Calls the StopTimers subroutine
            WinnerSoundTimer.Start() 'Starts WinnerSoundTimer
            picWrongArrow.Visible = True 'Hides picturebox
            lblGuess.ForeColor = Color.Maroon
            lblGuess.Text = word.ToUpper
            TokenTimer.Start() 'Starts timer
        End If
    End Sub
    Private Sub BeepTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BeepTimer.Tick 'BeepTimer Tick code
        ColorTimer.Start() 'Starts ColorTimer
        BeepTimer.Stop() 'Stops BeepTimer
    End Sub
    Private Sub ColorTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ColorTimer.Tick 'ColorTimer Tick code
        If lblGameTime.ForeColor = Color.Black Then
            lblGameTime.ForeColor = Color.Red
        Else
            lblGameTime.ForeColor = Color.Black
        End If
    End Sub
    Private Sub CountDownTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CountDownTimer.Tick 'CountDownTimer Tick code
        CountDownCounter = CountDownCounter - 1
        If CountDownCounter = 5 Then
            PicPlayerOne.Image = My.Resources.Hangman_Four
            PicPlayerTwo.Image = My.Resources.Hangman_Four
```

```
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 4 Then
            PicPlayerOne.Image = My.Resources.Hangman_Three
            PicPlayerTwo.Image = My.Resources.Hangman_Three
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 3 Then
            PicPlayerOne.Image = My.Resources.Hangman_Two
            PicPlayerTwo.Image = My.Resources.Hangman_Two
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 2 Then
            PicPlayerOne.Image = My.Resources.Hangman_One
            PicPlayerTwo.Image = My.Resources.Hangman_One
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormSelect
                frmMainMenu.player.Play() 'Plays the sound file
            End If
        End If
        If CountDownCounter = 1 Then
            PicPlayerOne.Image = Nothing
            PicPlayerTwo.Image = Nothing
            GameStart()
            CountDownTimer.Stop()
            CountDownCounter = 6
        End If
    End Sub
    Private Sub TokenTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TokenTimer.Tick 'TokenTimer Tick code
        TokenTime = TokenTime - 1
        SoundTime = SoundTime - 1
        If PlayerWithToken = "Player 1" Then
            PicPlayerOne.Image = My.Resources.Hangman_Token
        End If
        If PlayerWithToken = "Player 2" Then
            PicPlayerTwo.Image = My.Resources.Hangman_Token
        End If
        If TokenTime = 0 And PlayerToken = True Then
            PlayerToken = False 'Sets PlayerToken to False
            PlayerWithToken = "No One"
            PicPlayerOne.Image = Nothing
            PicPlayerTwo.Image = Nothing
            GameStart()  'Calls the GameStart subroutine
            TokenTime = 1
            TokenTimer.Stop() 'Stops TokenTimer
        End If
        If SoundTime = 0 Then 'Checks if SoundTime is 0
            PlayerWithToken = "No One"
            PicPlayerOne.Image = Nothing
            PicPlayerTwo.Image = Nothing
            GameStart() 'Calls the GameStart subroutine
            SoundTime = 2
```

```
            TokenTimer.Stop() 'Stops TokenTimer
        End If
    End Sub
    Private Sub WinnerSoundTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles WinnerSoundTimer.Tick
        If Winner = False Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            ButtonEnableTimer.Start() 'Starts ButtonEnableTimer
            WinnerSoundTimer.Stop() 'Stops WinnerSoundTimer
            Winner = False 'Sets Winner to False
        Else
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_setback
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            WinnerSoundTimer.Stop() 'Stops WinnerSoundTimer
            Winner = False 'Sets Winner to False
        End If
        If BothLose = True Then
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            BothLose = False 'Sets BothLose to False
            WinnerSoundTimer.Stop() 'Stops WinnerSoundTimer
        End If
    End Sub
    Private Sub ButtonEnableTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonEnableTimer.Tick
        btnStart.Enabled = True
        btnReturn.Enabled = True
        btnInfo.Enabled = True
        btnClose.Enabled = True
        btnMinimize.Enabled = True
    End Sub

    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString()  'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblLeaveGame.Visible = False 'Hides the "lblLeaveGame" label
                lblInfo.Visible = False 'Hides the "lblInfo" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblLeaveGame.Visible = True 'Shows the "lblLeaveGame" label
```

```
            lblInfo.Visible = True 'Shows the "lblInfo" label
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.7. Tic Tac Toe Mode Menu:

```
Public Class frmTicTacToeModeMenu 'frmTicTacToeModeMenu
    Dim AppPath As String = Application.StartupPath
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Sub btnPlay1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSelectSinglePlayer.Click 'btnSelectSinglePlayer click code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
            frmTicTacToeSinglePlayer.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
            frmTicTacToeSinglePlayer.btnMuteUnMute.BackgroundImage =
My.Resources.UnMute_Button
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.Hide()
        frmTicTacToeSinglePlayer.Location = New Point(Me.Location.X - 119, Me.Location.Y)
'Sets the "frmTicTacToeSinglePlayer" form's location to the current form's location
        frmTicTacToeSinglePlayer.Show() 'Shows the "frmTicTacToeSinglePlayer" form
    End Sub
```

```
    'Mute/Unmute Button Events
    Private Sub btnMuteUnMute_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseUp  'btnMuteUnMute MouseUp
code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            My.Settings.Mute = True 'Sets the "Mute" variable in application settings to True
            lblMute.Text = "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        Else
            My.Settings.Mute = False 'Sets the "Mute" variable in application settings to
False
            lblMute.Text = "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        End If
        My.Settings.Save()
    End Sub
    Private Sub btnMuteUnMute_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseDown 'btnMuteUnMute MouseDown
code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Pushed
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Pushed
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseEnter 'btnMuteUnMute MouseEnter code
        lblMute.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Highlighted 'Changes the
background image of the mute button to highlighted when the curser enters the button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Highlighted 'Changes
the background image of the mute button to highlighted when the curser enters the button
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseLeave 'btnMuteUnMute MouseLeave code
        lblMute.Visible = False
        If My.Settings.Mute = False Then 'Checks if "Mute" variable is set to False
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Changes the background
image of the mute button to highlighted when the curser enters the button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Changes the
background image of the mute button to highlighted when the curser enters the button
        End If
    End Sub
    'Minimize Button Events
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown 'btnMinimize MouseDown
code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter 'btnMinimize MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted
```

```
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave 'btnMinimize MouseLeave code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp 'btnMinimize MouseUp code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized
    End Sub
    'Exit Button Events
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    'Info Button Events
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        lblInfo.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
```

```
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False
        btnInfo.BackgroundImage = My.Resources.Info_Button
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp 'btnInfo Button MouseUp code
        btnInfo.BackgroundImage = My.Resources.Info_Button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    'Return Button Events
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed
    End Sub
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        lblMainMenu.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblMainMenu.Visible = False
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        frmMainMenu.Location = New Point(Me.Location.X - 119, Me.Location.Y)  'Sets the
"frmMainMenu" form's location to the current form's location
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormReturning
            frmMainMenu.player.Play() 'Plays the sound file
            frmMainMenu.lblMute.Text = "Mute Sounds"
            frmMainMenu.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            frmMainMenu.lblMute.Text = "UnMute Sounds"
            frmMainMenu.btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
        frmMainMenu.Show() 'Shows the "frmMainMenu" form
        Me.Close()
```

```
    End Sub
    Private Sub btnSettings_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseDown 'btnSettings Button
MouseDown code
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Pushed
    End Sub
    Private Sub btnSettings_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseEnter  'btnSettings Button MouseEnter code
        lblSettings.Visible = True
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Highlighted
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnSettings_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseLeave 'btnSettings Button MouseLeave code
        lblSettings.Visible = False
        btnSettings.BackgroundImage = My.Resources.Settings_Button
    End Sub
    Private Sub btnSettings_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseUp 'btnSettings Button MouseUp
code
        btnSettings.BackgroundImage = My.Resources.Settings_Button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        frmSettings.Show() 'Shows the "frmSettings" form
    End Sub
    Private Sub frmTicTacToeModeMenu_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmTicTacToeModeMenu form Load code
        If My.Settings.DisableCaptions = False Then
            MouseMoveTimer.Start()
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            lblMute.Text = "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            lblMute.Text = "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
    End Sub
    Private Sub GetPlayerNames() 'GetPlayerNames subroutine code
        If frmMainMenu.PlayerFlag = True Then
            Dim MessageBoxResult As String
            MessageBoxResult = MsgBox("Player One: " & frmMainMenu.PlayerOne & vbCrLf &
"Player Two: " & frmMainMenu.PlayerTwo & vbCrLf & vbCrLf & "Would you like to use the same
player names?", vbYesNo + vbInformation, "Player Names")
            If MessageBoxResult = vbNo Then
                frmMainMenu.PlayerFlag = False
                frmMainMenu.FirstPlayer = False
                frmMainMenu.SecondPlayer = False
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then
            Dim FirstPlayerInputBoxResult As String
            FirstPlayerInputBoxResult = InputBox("Player One, please enter your name:",
"Player One", "Enter Your Name Here")
            If FirstPlayerInputBoxResult = "Enter Your Name Here" Then
```

```
                MsgBox("You have not entered a name. Please enter your name to continue",
vbInformation, "Invalid Name")
                Exit Sub
            End If
            If FirstPlayerInputBoxResult = "" Then
                Exit Sub
            Else
                frmMainMenu.PlayerOne = FirstPlayerInputBoxResult
                frmMainMenu.FirstPlayer = True
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then
            Exit Sub
        End If
        If frmMainMenu.SecondPlayer = False Then
            Dim SecondPlayerInputBoxResult As String
            SecondPlayerInputBoxResult = InputBox("Player Two, please enter your name:",
"Player Two", "Enter Your Name Here")
            If SecondPlayerInputBoxResult = "Enter Your Name Here" Then
                MsgBox("You have not entered a name. Please enter your name to continue",
vbInformation, "Invalid Name")
                Exit Sub
            End If
            If SecondPlayerInputBoxResult = "" Then
                Exit Sub
            Else
                frmMainMenu.PlayerTwo = SecondPlayerInputBoxResult
                frmMainMenu.SecondPlayer = True
            End If
        End If
        If frmMainMenu.FirstPlayer = False Then
            Exit Sub
        End If
        frmMainMenu.PlayerFlag = True
    End Sub
    Private Sub btnTwoPlayer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelectTwoPlayer.Click 'btnSelectTwoPlayer Click code
        GetPlayerNames()
        If frmMainMenu.FirstPlayer = False Or frmMainMenu.SecondPlayer = False Then
            Exit Sub
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormOpening 'Sets the soundplayer
to the "FormOpening" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        frmTicTacToeTwoPlayer.lblPlayerOne.Text = frmMainMenu.PlayerOne
        frmTicTacToeTwoPlayer.lblPlayerTwo.Text = frmMainMenu.PlayerTwo
        Me.Hide()
        frmTicTacToeTwoPlayer.Location = New Point(Me.Location.X - 119, Me.Location.Y) 'Sets
the "frmTicTacToeTwoPlayer" form's location to the current form's location
        frmTicTacToeTwoPlayer.Show() 'Shows the "frmTicTacToeTwoPlayer" form
    End Sub
    Dim CaptionCounter As Integer
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
```

```
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblMainMenu.Visible = False 'Hides the "lblMainMenu" label
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblMute.Visible = False 'Hides the "lblMute" label
                lblSettings.Visible = False 'Hides the "lblSettings" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblMainMenu.Visible = True 'Shows the "lblMainMenu" label
            lblInfo.Visible = True 'Shows the "lblInfo" label
            lblSettings.Visible = True 'Shows the "lblSettings" label
            If btnMuteUnMute.Visible = True Then
                lblMute.Visible = True 'Shows the "lblMute" label
            End If
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.8. Tic Tac Toe - Single Player:

```
Public Class frmTicTacToeSinglePlayer 'frmTicTacToeSinglePlayer form code
    Dim TopLeft As String
    Dim TopRight As String
    Dim TopCenter As String
    Dim MiddleLeft As String
    Dim BottomLeft As String
    Dim MiddleRight As String
    Dim BottomRight As String
    Dim list As New ArrayList
    Dim MiddleCenter As String
    Dim BottomCenter As String
    Dim WinningPlayer As String
    Dim Draw As Boolean = False
    Dim Chosen As Boolean = False
    Dim Winner As Boolean = False
    Dim ExitFlag As Boolean = True
    Dim PlayerOne As Boolean = True
    Dim Countdowntimer As Integer = 9
    Dim PlayerOneTokenNo As Integer = 0
    Dim WonRound As Boolean = False 'Checks if a player has won the game or not
    Dim PlayerTwoWinner As Boolean = False
    Dim PlayerOneWinner As Boolean = False
    Dim ComputerPlayerTokenNo As Integer = 0
    Dim AppPath As String = Application.StartupPath
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
```

```
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Sub ComputerPlayer() 'ComputerPlayer Subroutine code
        While Chosen = False 'Executes a set of commands while the boolean variable "Chosen"
is equal to false
            Dim RandomNumber As New Random 'Declares "RandomNumber" as a new random value
generator
            Dim Index As Integer 'Declares "Index" as an integer variable
            Dim ChosenNumber As Integer 'Declares "ChosenNumber" as an integer variable
            If list.Count <= 0 Then 'Checks if there are no words in the "list" list box
                Exit Sub 'Exits the subroutine
            End If
            Index = RandomNumber.Next(0, list.Count - 1) 'Assigns the next "RandomNumber" of
the available list items to the "Index" variable
            ChosenNumber = list(Index) 'Gets the chosen number from the "list" list box
"index"
            list.RemoveAt(Index) 'Removes the "ChosenNumber" index from the "list" list box
            If ChosenNumber = 1 Then 'Checks if "ChosenNumber" is 1
                Call btnTopLeft_MouseUp(Nothing, Nothing) 'Calls the "btnTopLeft" MouseUp
event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 2 Then 'Checks if "ChosenNumber" is 2
                Call btnTopCenter_MouseUp(Nothing, Nothing) 'Calls the "btnTopCenter" MouseUp
event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 3 Then 'Checks if "ChosenNumber" is 3
                Call btnTopRight_MouseUp(Nothing, Nothing) 'Calls the "btnTopRight" MouseUp
event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 4 Then 'Checks if "ChosenNumber" is 4
                Call btnMiddleLeft_MouseUp(Nothing, Nothing) 'Calls the "btnMiddleLeft"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 5 Then 'Checks if "ChosenNumber" is 5
                Call btnMiddleCenter_MouseUp(Nothing, Nothing) 'Calls the "btnMiddleCenter"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 6 Then 'Checks if "ChosenNumber" is 6
                Call btnMiddleRight_MouseUp(Nothing, Nothing) 'Calls the "btnMiddleRight"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 7 Then 'Checks if "ChosenNumber" is 7
                Call btnBottomLeft_MouseUp(Nothing, Nothing) 'Calls the "btnBottomLeft"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 8 Then 'Checks if "ChosenNumber" is 8
```

```
                Call btnBottomCenter_MouseUp(Nothing, Nothing) 'Calls the "btnBottomCenter"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            ElseIf ChosenNumber = 9 Then 'Checks if "ChosenNumber" is 9
                Call btnBottomRight_MouseUp(Nothing, Nothing) 'Calls the "btnBottomRight"
MouseUp event
                Chosen = True 'Sets the boolean variable "Chosen" to True
            End If
        End While
        PlayerOne = True 'Sets the boolean variable "PlayerOne" to True
        Chosen = False 'Sets the boolean variable "Chosen" to False
    End Sub
    Private Sub TokenSoundStream() 'TokenSoundStream Subroutine code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub CheckPlayerOneTurn() 'CheckPlayerOneTurn Subroutine code
        ExitFlag = False 'Sets the boolean variable "ExitFlag" to False
        WonRound = False 'Sets the boolean variable "WonRound" to False
        Draw = False 'Sets the boolean variable "Draw" to False
        If TopLeft = "X" And MiddleCenter = "X" And BottomRight = "X" Then 'Checks if the
"TopLeft","MiddleCenter" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "X" And TopCenter = "X" And TopRight = "X" Then 'Checks if the
"TopLeft","TopCenter" and "TopRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf MiddleLeft = "X" And MiddleCenter = "X" And MiddleRight = "X" Then 'Checks if
the "MiddleLeft","MiddleCenter" and "MiddleRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf BottomLeft = "X" And BottomCenter = "X" And BottomRight = "X" Then 'Checks if
the "BottomLeft","BottomCenter" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "X" And MiddleLeft = "X" And BottomLeft = "X" Then 'Checks if the
"TopLeft","MiddleLeft" and "BottomLeft" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopCenter = "X" And MiddleCenter = "X" And BottomCenter = "X" Then 'Checks if
the "TopCenter","MiddleCenter" and "BottomCenter" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "X" And MiddleRight = "X" And BottomRight = "X" Then 'Checks if the
"TopRight","MiddleRight" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "X" And MiddleCenter = "X" And BottomLeft = "X" Then 'Checks if the
"TopRight","MiddleCenter" and "BottomLeft" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        End If
        If WonRound = True Then 'Checks if the boolean variable "WonRound" is set to True
            PlayerOneTokenNo = PlayerOneTokenNo + 1 'Increments the "PlayerOneTokenNo"
integer variable
            TokenSoundStream() 'Calls the "TokenSoundStream" subroutine
            DisplayPlayerOneWinner() 'Calls the "DisplayPlayerOneWinner" subroutine
            GetTokens() 'Calls the "GetTokens" subroutine
            CheckDraw() 'Calls the "CheckDraw" subroutine
            If Draw = True Then 'Checks if the boolean variable "Draw" is set to true
                Draw = False 'Sets the boolean variable "Draw" to False
            End If
        Else
            PlayerOne = False 'Sets the boolean variable "PlayerOne" to False
            Chosen = False 'Sets the boolean variable "Chosen" to False
            CheckDraw() 'Calls the "CheckDraw" subroutine
            ComputerPlayer() 'Calls the "ComputerPlayer" subroutine
        End If
    End Sub
```

```
    Private Sub CheckComputerPlayerTurn() 'CheckComputerPlayerTurn Subroutine code
        ExitFlag = False 'Sets the boolean variable "ExitFlag" to False
        WonRound = False 'Sets the boolean variable "WonRound" to False
        Draw = False 'Sets the boolean variable "Draw" to False
        If TopLeft = "O" And MiddleCenter = "O" And BottomRight = "O" Then 'Checks if the
"TopLeft","MiddleCenter" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "O" And TopCenter = "O" And TopRight = "O" Then 'Checks if the
"TopLeft","TopCenter" and "TopRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf MiddleLeft = "O" And MiddleCenter = "O" And MiddleRight = "O" Then 'Checks if
the "MiddleLeft","MiddleCenter" and "MiddleRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf BottomLeft = "O" And BottomCenter = "O" And BottomRight = "O" Then 'Checks if
the "BottomLeft","BottomCenter" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "O" And MiddleLeft = "O" And BottomLeft = "O" Then 'Checks if the
"TopLeft","MiddleLeft" and "BottomLeft" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopCenter = "O" And MiddleCenter = "O" And BottomCenter = "O" Then 'Checks if
the "TopCenter","MiddleCenter" and "BottomCenter" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "O" And MiddleRight = "O" And BottomRight = "O" Then 'Checks if the
"TopRight","MiddleRight" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "O" And MiddleCenter = "O" And BottomLeft = "O" Then 'Checks if the
"TopRight","MiddleCenter" and "BottomLeft" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        End If
        If WonRound = True Then 'Checks if the boolean variable "WonRound" to True
            ComputerPlayerTokenNo = ComputerPlayerTokenNo + 1 'Increments the
"ComputerPlayerTokenNo" integer variable
            TokenSoundStream() 'Calls the "TokenSoundStream" subroutine
            DisplayPlayerTwoWinner() 'Calls the "DisplayPlayerTwoWinner" subroutine
            GetTokens() 'Calls the "GetTokens" subroutine
        End If
        PlayerOne = True 'Sets the boolean variable "PlayerOne" to True
        Chosen = False 'Sets the boolean variable "Chosen" to False
        CheckDraw() 'Calls the "CheckDraw" subroutine
    End Sub
    Private Sub DisplayPlayerOneWinner() 'DisplayPlayerOneWinner Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picPlayerOneToken.Visible = True 'Shows the "picPlayerOneToken" picture box
        DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
    End Sub
    Private Sub DisplayPlayerTwoWinner() 'DisplayPlayerTwoWinner Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picComputerPlayerToken.Visible = True 'Shows the "picComputerPlayerToken" picture box
        DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
    End Sub
    Private Sub DisableButtons() 'DisableButtons Subroutine code
        picTopLeft.Enabled = False 'Disables the "btnTopLeft" button
        picMiddleLeft.Enabled = False 'Disables the "btnMiddleLeft" button
        picBottomLeft.Enabled = False 'Disables the "btnBottomLeft" button
        picTopCenter.Enabled = False 'Disables the "btnTopCenter" button
        picMiddleCenter.Enabled = False 'Disables the "btnMiddleCenter" button
        picBottomCenter.Enabled = False 'Disables the "btnBottomCenter" button
        picTopRight.Enabled = False 'Disables the "btnTopRight" button
        picMiddleRight.Enabled = False 'Disables the "btnMiddleRight" button
        picBottomRight.Enabled = False 'Disables the "btnBottomRight" button
    End Sub
    Private Sub EnableButtons() 'EnableButtons Subroutine code
        picTopLeft.Enabled = True 'Enables the "btnTopLeft" button
        picMiddleLeft.Enabled = True 'Enables the "btnMiddleLeft" button
        picBottomLeft.Enabled = True 'Enables the "btnBottomLeft" button
        picTopCenter.Enabled = True 'Enables the "btnTopCenter" button
```

```
        picMiddleCenter.Enabled = True 'Enables the "btnMiddleCenter" button
        picBottomCenter.Enabled = True 'Enables the "btnBottomCenter" button
        picTopRight.Enabled = True 'Enables the "btnTopRight" button
        picMiddleRight.Enabled = True 'Enables the "btnMiddleRight" button
        picBottomRight.Enabled = True 'Enables the "btnBottomRight" button
    End Sub
    Private Sub RefreshGame() 'RefreshGame Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        WriteList() 'Calls the "WriteList" subroutine
        ClearBoard() 'Calls the "ClearBoard" subroutine
        ClearBoardStrings() 'Calls the "ClearBoardStrings" subroutine
        EnableButtons() 'Calls the "EnableButtons" subroutine
        ResetBooleans() 'Calls the "ResetBooleans" subroutine
    End Sub
    Private Sub RestartGame() 'RestartGame Subroutine code
        WriteList() 'Calls the "WriteList" subroutine
        ClearBoard() 'Calls the "ClearBoard" subroutine
        ClearBoardStrings() 'Calls the "ClearBoardStrings" subroutine
        ClearTokens() 'Calls the "ClearTokens" subroutine
        ResetBooleans() 'Calls the "ResetBooleans" subroutine
        EnableButtons() 'Calls the "EnableButtons" subroutine
        PlayerOneTokenNo = 0 'Sets the integer variable "PlayerOneTokenNo" to 0
        ComputerPlayerTokenNo = 0 'Sets the integer variable "ComputerPlayerTokenNo" to 0
        lblPlayerOneTokens.Text = "Tokens: 0"
        lblComputerPlayerTokens.Text = "Tokens: 0"
        picTicTacToeWinner.Visible = False 'Hides the "picTicTacToeWinner" picture box
    End Sub
    Private Sub ResetBooleans() 'ResetBooleans Subroutine code
        Chosen = False 'Sets the boolean variable "Chosen" to False
        PlayerOne = True 'Sets the boolean variable "PlayerOne" to True
        Draw = False 'Sets the boolean variable "Draw" to False
        PlayerTwoWinner = False 'Sets the boolean variable "PlayerTwoWinner" to False
        PlayerOneWinner = False 'Sets the boolean variable "PlayerOneWinner" to False
    End Sub
    Private Sub ClearBoardStrings() 'ClearBoardStrings Subroutine code
        TopLeft = "" 'Clears the value of the string variable "Topleft"
        MiddleLeft = "" 'Clears the value of the string variable "MiddleLeft"
        BottomLeft = "" 'Clears the value of the string variable "BottomLeft"
        TopCenter = "" 'Clears the value of the string variable "TopCenter"
        MiddleCenter = "" 'Clears the value of the string variable "MiddleCenter"
        BottomCenter = "" 'Clears the value of the string variable "BottomCenter"
        TopRight = "" 'Clears the value of the string variable "TopRight"
        MiddleRight = "" 'Clears the value of the string variable "MiddleRight"
        BottomRight = "" 'Clears the value of the string variable "BottomRight"
    End Sub
    Private Sub ClearBoard() 'ClearBoard Subroutine code
        picTopLeft.BackgroundImage = Nothing 'Sets the "picTopLeft" picture box's background
image to Nothing
        picMiddleLeft.BackgroundImage = Nothing 'Sets the "picMiddleLeft" picture box's
background image to Nothing
        picBottomLeft.BackgroundImage = Nothing 'Sets the "picBottomLeft" picture box's
background image to Nothing
        picTopCenter.BackgroundImage = Nothing 'Sets the "picTopCenter" picture box's
background image to Nothing
        picMiddleCenter.BackgroundImage = Nothing 'Sets the "picMiddleCenter" picture box's
background image to Nothing
        picBottomCenter.BackgroundImage = Nothing 'Sets the "picBottomCenter" picture box's
background image to Nothing
        picTopRight.BackgroundImage = Nothing 'Sets the "picTopRight" picture box's
background image to Nothing
        picMiddleRight.BackgroundImage = Nothing 'Sets the "picMiddleRight" picture box's
background image to Nothing
        picBottomRight.BackgroundImage = Nothing 'Sets the "picBottomRight" picture box's
background image to Nothing
    End Sub
    Private Sub ClearTokens() 'ClearTokens Subroutine code
```

```
        picPlayer1_GoldToken1.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken1" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken2.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken2" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken3.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken3" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken4.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken4" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken5.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken5" picture box's background image to the "No_Token" image in resources
        picComputerPlayer_GoldToken1.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken1" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken2.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken2" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken3.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken3" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken4.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken4" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken5.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken5" picture box's background image to the "No_Token" image in
resources
    End Sub
    Private Sub WriteList() 'WriteList Subroutine code
        list.Clear() 'Clears the "list" list box
        For i = 1 To 10 'Executes a set of commands 10 times (1-10)
            list.Add(i) 'Adds the numbers (1-10) to the "list" list box's collection
        Next i
    End Sub
    Private Sub GetTokens() 'GetTokens Subroutine code
        If PlayerOneTokenNo > 0 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 0
            picPlayer1_GoldToken1.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken1" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 1 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 1
            picPlayer1_GoldToken2.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken2" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 2 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 2
            picPlayer1_GoldToken3.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken3" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 3 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 3
            picPlayer1_GoldToken4.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken4" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 4 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 4
            picPlayer1_GoldToken5.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken5" picture box's background image to the "Gold_Token" image in resources
        End If
        lblPlayerOneTokens.Text = "Tokens: " & PlayerOneTokenNo
        If ComputerPlayerTokenNo > 0 Then 'Checks if the integer variable
"ComputerPlayerTokenNo" is greater than 0
            picComputerPlayer_GoldToken1.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken1" picture box's background image to the "Gold_Token" image in
resources
        End If
```

```
        If ComputerPlayerTokenNo > 1 Then 'Checks if the integer variable
"ComputerPlayerTokenNo" is greater than 1
            picComputerPlayer_GoldToken2.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken2" picture box's background image to the "Gold_Token" image in
resources
        End If
        If ComputerPlayerTokenNo > 2 Then 'Checks if the integer variable
"ComputerPlayerTokenNo" is greater than 2
            picComputerPlayer_GoldToken3.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken3" picture box's background image to the "Gold_Token" image in
resources
        End If
        If ComputerPlayerTokenNo > 3 Then 'Checks if the integer variable
"ComputerPlayerTokenNo" is greater than 3
            picComputerPlayer_GoldToken4.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken4" picture box's background image to the "Gold_Token" image in
resources
        End If
        If ComputerPlayerTokenNo > 4 Then 'Checks if the integer variable
"ComputerPlayerTokenNo" is greater than 4
            picComputerPlayer_GoldToken5.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken5" picture box's background image to the "Gold_Token" image in
resources
        End If
        lblComputerPlayerTokens.Text = "Tokens: " & ComputerPlayerTokenNo
    End Sub
    Private Sub CheckDraw() 'CheckDraw Subroutine code
        If WonRound = False And picTopLeft.Enabled = False And picMiddleLeft.Enabled = False
And picBottomLeft.Enabled = False And picTopCenter.Enabled = False And
picMiddleCenter.Enabled = False And picBottomCenter.Enabled = False And picTopRight.Enabled =
False And picMiddleRight.Enabled = False And picBottomRight.Enabled = False Then 'Checks if
all the pictureboxes are disabled
            Draw = True 'Sets the boolean variable "Draw" to True
        End If
        CheckWinner() 'Calls the "CheckWinner" subroutine
    End Sub
    Private Sub CheckWinner() 'CheckWinner Subroutine code
        If PlayerOneTokenNo = 5 Then
            GetTokens() 'Calls the "GetTokens" subroutine
            PlayerOneTokenNo = 0
            ComputerPlayerTokenNo = 0
            Winner = True 'Sets the boolean variable "Winner" to True
            PlayerOneWinner = True 'Sets the boolean variable "PlayerOneWinner" to True
        ElseIf ComputerPlayerTokenNo = 5 Then
            GetTokens() 'Calls the "GetTokens" subroutine
            Winner = True 'Sets the boolean variable "Winner" to True
            PlayerTwoWinner = True 'Sets the boolean variable "PlayerTwoWinner" to True
        End If
        DisplayWinner() 'Calls the "DisplayWinner" subroutine
        If Draw = True Then 'Checks if the boolean variable "Draw" is set to True
            DisableButtons() 'Calls the "DisableButtons" subroutine
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            picTicTacToeWinner.BackgroundImage = My.Resources.Player_Draw 'Changes the
"picTicTacToeWinner" picture box's background image to "Player_Draw" in resources
            picTicTacToeWinner.Visible = True 'Shows the "picTicTacToeWinner" picture box
            DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
        End If
    End Sub
    Private Sub DisplayWinner() 'DisplayWinner Subroutine code
        If Winner = True Then 'Checks if a player has won the game
            DisplayTimer.Stop() 'Stops the "DisplayTimer" timer tick event
            picPlayerOneToken.Visible = False 'Hides the "picPlayerOneToken" picturebox
```

```
            picComputerPlayerToken.Visible = False 'Hides the "picComputerPlayerToken"
picturebox
            If PlayerOneWinner = True Then 'Checks if player one has won the game
                WinnerSoundTimer.Start() 'Starts the sound timer tick event
                picTicTacToeWinner.Visible = True 'Makes the display visible
                picTicTacToeWinner.BackgroundImage = My.Resources.Player_One_Wins 'Changes
the image to "Player one has won the game!"
            ElseIf PlayerTwoWinner = True Then 'Checks if player Two has won the game. In
this case, the computer player
                WinnerSoundTimer.Start() 'Starts the sound timer tick event
                picTicTacToeWinner.Visible = True 'Makes the display visible
                picTicTacToeWinner.BackgroundImage = My.Resources.Computer_Player_Wins
'Changes the image to "Computer player has won the game!"
            End If
            btnStart.Visible = True 'Shows the "btnStart" button
        End If
    End Sub
    Private Sub DisplayTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DisplayTimer.Tick 'DisplayTimer Timer tick code
        RefreshGame() 'Calls the "RefreshGame" subroutine
        Draw = False  'Sets the boolean variable "Draw" to True
        picTicTacToeWinner.Visible = False 'Hides the "picTicTacToeWinner" picturebox
        picPlayerOneToken.Visible = False 'Hides the "picPlayerOneToken" picturebox
        picComputerPlayerToken.Visible = False 'Hides the "picComputerPlayerToken" picturebox
        DisplayTimer.Stop() 'Stops the "DisplayTimer" timer tick event
    End Sub
    Private Sub WinnerSoundTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles WinnerSoundTimer.Tick 'WinnerSoundTimer Timer tick code
        If Winner = False Then  'Checks if there is a winner
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
                frmMainMenu.player.Play() 'Sets the soundplayer to the "FormMinimizing" WAV
file in the resources
            End If
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
            Winner = False  'Sets the boolean variable "Winner" to False
        Else
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_setback
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Winner = False 'Sets the boolean variable "Winner" to False
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
        End If
        If Draw = True Then 'Checks if there is a draw
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Draw = False 'Sets the boolean variable "Draw" to False
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
        End If
    End Sub
    Private Sub frmTicTacToeSinglePlayer_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmTicTacToeSingleplayer Form Load code
        If My.Settings.DisableCaptions = False Then
            MouseMoveTimer.Start()
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            lblMute.Text = "Mute Sounds" 'Sets the "lblMute" label to "Mute Sounds"
```

```
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        Else
            lblMute.Text = "UnMute Sounds" 'Sets the "lblMute" label to "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        End If
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picTicTacToeWinner.BackgroundImage = My.Resources.Press_Start 'Changes the image to
"Press Start"
        picTicTacToeWinner.Visible = True 'Shows the "picTicTacToeWinner" picture box
        list.Clear() 'Clears the "list" list box
        For i = 1 To 10 'Executes a set of commands 10 times (1-10)
            list.Add(i) 'Adds the numbers (1-10) to the "list" list box's collection
        Next i
    End Sub
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown 'btnMinimize Button
MouseDown code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed 'Changes the
background image of the "btnMinimize" button when the mouse is down
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter 'btnMinimize Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted 'Changes the
background image of the "btnMinimize" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave 'btnMinimize Button MouseLeave code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp 'btnMinimize Button MouseUp
code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized 'Minimizes the form by changing the form's
"FormWindowState" properties to "Minimized"
    End Sub
    'Exit
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed 'Changes the background
image of the "btnClose" button when the mouse is down
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
```

```
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
background image of the close button to highlighted when the curser enters the button
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed 'Changes the background
image of the "btnInfo" button when the mouse is down
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        lblInfo.Visible = True 'Shows the "lblInfo" label when mouse enters the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted 'Changes the
background image of the info button to highlighted when the curser enters the button
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False 'Hides the "lblInfo" label when mouse leaves the picture box
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
```

```
    Private Sub btnSettings_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseDown 'btnSettings Button
MouseDown code
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Pushed 'Changes the
background image of the "btnSettings" button when the mouse is down
    End Sub
    Private Sub btnSettings_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseEnter 'btnSettings Button MouseEnter code
        lblSettings.Visible = True 'Shows the "lblSettings" label when mouse enters the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Highlighted 'Changes the
background image of the settings button to highlighted when the curser enters the button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnSettings_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseLeave 'btnSettings Button MouseLeave code
        lblSettings.Visible = False 'Hides the "lblSettings" label when mouse leaves the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnSettings_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseUp 'btnSettings Button MouseUp
code
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        frmSettings.Show() 'Shows the "frmSettings" form
    End Sub
    'MuteUnMute
    Private Sub btnMuteUnMute_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseUp 'btnMuteUnMute Button
MouseUp code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            My.Settings.Mute = True 'Sets the "Mute" variable in application settings to True
            lblMute.Text = "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        Else
            My.Settings.Mute = False 'Sets the "Mute" variable in application settings to
False
            lblMute.Text = "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseDown 'btnMuteUnMute Button
MouseDown code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Pushed
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Pushed
        End If
    End Sub
```

```
    Private Sub btnMuteUnMute_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseEnter 'btnMuteUnMute Button MouseEnter code
        lblMute.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Highlighted
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Highlighted
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseLeave 'btnMuteUnMute Button MouseLeave code
        lblMute.Visible = False
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed
    End Sub
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        lblLeaveGame.Visible = True
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblLeaveGame.Visible = False
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        If ExitFlag = False Then
            Dim MessageBoxResult As String
            MessageBoxResult = MsgBox("Are you sure you wish to exit the game?", vbYesNo,
"Exit Game")
            If MessageBoxResult = vbYes Then
                ExitFlag = True
            End If
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            frmTicTacToeModeMenu.lblMute.Text = "Mute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
```

```
            frmTicTacToeModeMenu.lblMute.Text = "UnMute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
        frmTicTacToeModeMenu.Location = New Point(Me.Location.X + 119, Me.Location.Y)
        frmTicTacToeModeMenu.Show() 'Shows the "frmTicTacToeModeMenu" form
        Me.Dispose() 'Closes the current form
    End Sub
    'Start
    Private Sub btnStart_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseDown 'btnStart Button MouseDown
code
        btnStart.BackgroundImage = My.Resources.Start_Button_Pushed 'Changes the background
image of the "btnStart" button when the mouse is down
    End Sub
    Private Sub btnStart_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseEnter 'btnStart Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.BackgroundImage = My.Resources.Start_Button_Highlighted 'Changes the
background image of the "btnStart" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnStart_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseLeave 'btnStart Button MouseLeave code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnStart_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseUp 'btnStart Button MouseUp code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        picTicTacToeWinner.Visible = False
        btnStart.Visible = False
        RestartGame()
    End Sub

    'Picture Boxes
    Private Sub btnTopLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopLeft.MouseUp  'btnpicTopLeftStart MouseUp code
        If PlayerOne = True Then
            picTopLeft.Enabled = False
            picTopLeft.BackgroundImage = My.Resources.TicTacToe_X1
            TopLeft = "X"
            list.Remove(1) 'Removes Number 1 from list
            CheckPlayerOneTurn()
        Else
            picTopLeft.Enabled = False
            picTopLeft.BackgroundImage = My.Resources.TicTacToe_O
            TopLeft = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnTopCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopCenter.MouseUp 'picTopCenter MouseUp code
        If PlayerOne = True Then
            picTopCenter.Enabled = False
            picTopCenter.BackgroundImage = My.Resources.TicTacToe_X1
            TopCenter = "X"
```

```
            list.Remove(2) 'Removes Number 2 from list
            CheckPlayerOneTurn()
        Else
            picTopCenter.Enabled = False
            picTopCenter.BackgroundImage = My.Resources.TicTacToe_O
            TopCenter = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnTopRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopRight.MouseUp 'picTopRight MouseUp code
        If PlayerOne = True Then
            picTopRight.Enabled = False
            picTopRight.BackgroundImage = My.Resources.TicTacToe_X1
            TopRight = "X"
            list.Remove(3) 'Removes Number 3 from list
            CheckPlayerOneTurn()
        Else
            picTopRight.Enabled = False
            picTopRight.BackgroundImage = My.Resources.TicTacToe_O
            TopRight = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnMiddleLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleLeft.MouseUp 'picMiddleLeft MouseUp code
        If PlayerOne = True Then
            picMiddleLeft.Enabled = False
            picMiddleLeft.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleLeft = "X"
            list.Remove(4) 'Removes Number 4 from list
            CheckPlayerOneTurn()
        Else
            picMiddleLeft.Enabled = False
            picMiddleLeft.BackgroundImage = My.Resources.TicTacToe_O
            MiddleLeft = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnMiddleCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseUp 'picMiddleCenter MouseUp code
        If PlayerOne = True Then
            picMiddleCenter.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleCenter = "X"
            list.Remove(5) 'Removes Number 5 from list
            CheckPlayerOneTurn()
            picMiddleCenter.Enabled = False
        Else
            picMiddleCenter.Enabled = False
            picMiddleCenter.BackgroundImage = My.Resources.TicTacToe_O
            MiddleCenter = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnMiddleRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseUp 'picMiddleCenter MouseUp code
        If PlayerOne = True Then
            picMiddleRight.Enabled = False
            picMiddleRight.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleRight = "X"
            list.Remove(6) 'Removes Number 6 from list
            CheckPlayerOneTurn()
        Else
            picMiddleRight.Enabled = False
            picMiddleRight.BackgroundImage = My.Resources.TicTacToe_O
            MiddleRight = "O"
```

```
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnBottomLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomLeft.MouseUp 'picBottomLeft MouseUp code
        If PlayerOne = True Then
            picBottomLeft.Enabled = False
            picBottomLeft.BackgroundImage = My.Resources.TicTacToe_X1
            BottomLeft = "X"
            list.Remove(7) 'Removes Number 7 from list
            CheckPlayerOneTurn()
        Else
            picBottomLeft.Enabled = False
            picBottomLeft.BackgroundImage = My.Resources.TicTacToe_O
            BottomLeft = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnBottomCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseUp 'picBottomCenter MouseUp code
        If PlayerOne = True Then
            picBottomCenter.Enabled = False
            picBottomCenter.BackgroundImage = My.Resources.TicTacToe_X1
            BottomCenter = "X"
            list.Remove(8) 'Removes Number 8 from list
            CheckPlayerOneTurn()
        Else
            picBottomCenter.Enabled = False
            picBottomCenter.BackgroundImage = My.Resources.TicTacToe_O
            BottomCenter = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub btnBottomRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseUp 'picBottomRight MouseUp code
        If PlayerOne = True Then
            picBottomRight.Enabled = False
            picBottomRight.BackgroundImage = My.Resources.TicTacToe_X1
            BottomRight = "X"
            list.Remove(9) 'Removes Number 9 from list
            CheckPlayerOneTurn()
        Else
            picBottomRight.Enabled = False
            picBottomRight.BackgroundImage = My.Resources.TicTacToe_O
            BottomRight = "O"
            CheckComputerPlayerTurn()
        End If
    End Sub
    Private Sub picTopLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopLeft.MouseEnter 'picTopLeft MouseEnter code
        If PlayerOne = True Then
            picTopLeft.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picTopLeft_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopLeft.MouseLeave 'picTopLeft MouseLeave code
        picTopLeft.Image = Nothing
    End Sub
    Private Sub picTopCenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopCenter.MouseEnter 'picBottomRight MouseUp code
        If PlayerOne = True Then
            picTopCenter.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picTopCenter_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopCenter.MouseLeave 'picTopCenter MouseLeave code
```

```
        picTopCenter.Image = Nothing
    End Sub
    'Top Right
    Private Sub picTopRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopRight.MouseEnter 'picTopRight MouseEnter code
        If PlayerOne = True Then
            picTopRight.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picTopRight_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picTopRight.MouseLeave 'picTopRight MouseLeave code
        picTopRight.Image = Nothing
    End Sub
    'Middle Left
    Private Sub picMiddleLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleLeft.MouseEnter 'picMiddleLeft MouseEnter code
        If PlayerOne = True Then
            picMiddleLeft.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picMiddleLeft_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picMiddleLeft.MouseLeave 'picMiddleLeft MouseLeave code
        picMiddleLeft.Image = Nothing
    End Sub
    'Middle Center
    Private Sub picMiddlecenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseEnter 'picMiddleLeft MouseLeave code
        If PlayerOne = True Then
            picMiddleCenter.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picMiddleCenter_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseLeave 'picMiddleLeft MouseLeave code
        picMiddleCenter.Image = Nothing
    End Sub
    'Middle Right
    Private Sub picMiddleRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseEnter 'picMiddleRight MouseEnter code
        If PlayerOne = True Then
            picMiddleRight.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picMiddleRight_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseLeave 'picMiddleRight MouseLeave code
        picMiddleRight.Image = Nothing
    End Sub
    'Bottom Left
    Private Sub picBottomLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomLeft.MouseEnter 'picBottomLeft MouseEnter code
        If PlayerOne = True Then
            picBottomLeft.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picBottomLeft_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picBottomLeft.MouseLeave 'picBottomLeft MouseLeave code
        picBottomLeft.Image = Nothing
    End Sub
    'Bottom Center
    Private Sub picBottomCenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseEnter 'picBottomCenter MouseEnter code
        If PlayerOne = True Then
            picBottomCenter.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picBottomCenter_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseLeave 'picBottomCenter MouseLeave code
```

```
        picBottomCenter.Image = Nothing
    End Sub
    'Bottom Right
    Private Sub picBottomRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseEnter 'picBottomRight MouseEnter code
        If PlayerOne = True Then
            picBottomRight.Image = My.Resources.Selected_Box_Green()
        End If
    End Sub
    Private Sub picBottomRight_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseLeave 'picBottomRight MouseLeave code
        picBottomRight.Image = Nothing
    End Sub

    Private Sub frmTicTacToeSinglePlayer_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles MyBase.KeyPress 'frmTicTacToeSinglePlayer
form KeyPress code
        If Asc(e.KeyChar) = 13 And btnStart.Visible = True Then
            btnStart_MouseUp(Nothing, Nothing)
        End If
    End Sub
    Dim CaptionCounter As Integer
    Dim CurrentMousePosition As String
    Dim OpacityCounter As Integer
    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblLeaveGame.Visible = False 'Hides the "lblLeaveGame" label
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblMute.Visible = False 'Hides the "lblMute" label
                lblSettings.Visible = False 'Hides the "lblSettings" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblLeaveGame.Visible = True 'Shows the "lblLeaveGame" label
            lblInfo.Visible = True 'Shows the "lblInfo" label
            lblSettings.Visible = True 'Shows the "lblSettings" label
            If btnMuteUnMute.Visible = True Then 'Checks if the "btnMuteUnMute" picturebox is
showing
                lblMute.Visible = True 'Shows the "lblMute" label
            End If
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1  'Increments "CaptionCounter" integer variable
    End Sub

End Class
```

8.9. Tic Tac Toe - Two Player:

```
Public Class frmTicTacToeTwoPlayer
```

```
    Dim TopLeft As String
    Dim TopRight As String
    Dim TopCenter As String
    Dim MiddleLeft As String
    Dim BottomLeft As String
    Dim PlayerTemp As String
    Dim MiddleRight As String
    Dim BottomRight As String
    Dim MiddleCenter As String
    Dim BottomCenter As String
    Dim WinningPlayer As String
    Dim Draw As Boolean = False
    Dim OpacityCounter As Integer
    Dim CaptionCounter As Integer
    Dim Winner As Boolean = False
    Dim ExitFlag As Boolean = True
    Dim PlayerOne As Boolean = True
    Dim Countdowntimer As Integer = 9
    Dim CurrentMousePosition As String
    Dim PlayerTwoTokenNo As Integer = 0
    Dim PlayerOneTokenNo As Integer = 0
    Dim WonRound As Boolean = False 'Checks if a player has won the game or not
    Dim PlayerTwoWinner As Boolean = False
    Dim PlayerOneWinner As Boolean = False
    Dim AppPath As String = Application.StartupPath
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub

    Private Sub TokenSoundStream() 'TokenSoundStream Subroutine code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
```

```
    Private Sub CheckPlayerOneTurn() 'CheckPlayerOneTurn Subroutine code
        ExitFlag = False 'Sets the boolean variable "ExitFlag" to False
        WonRound = False 'Sets the boolean variable "WonRound" to False
        Draw = False 'Sets the boolean variable "Draw" to False
        If TopLeft = "X" And MiddleCenter = "X" And BottomRight = "X" Then 'Checks if the
"TopLeft","MiddleCenter" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "X" And TopCenter = "X" And TopRight = "X" Then 'Checks if the
"TopLeft","TopCenter" and "TopRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf MiddleLeft = "X" And MiddleCenter = "X" And MiddleRight = "X" Then 'Checks if
the "MiddleLeft","MiddleCenter" and "MiddleRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf BottomLeft = "X" And BottomCenter = "X" And BottomRight = "X" Then 'Checks if
the "BottomLeft","BottomCenter" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "X" And MiddleLeft = "X" And BottomLeft = "X" Then 'Checks if the
"TopLeft","MiddleLeft" and "BottomLeft" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopCenter = "X" And MiddleCenter = "X" And BottomCenter = "X" Then 'Checks if
the "TopCenter","MiddleCenter" and "BottomCenter" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "X" And MiddleRight = "X" And BottomRight = "X" Then 'Checks if the
"TopRight","MiddleRight" and "BottomRight" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "X" And MiddleCenter = "X" And BottomLeft = "X" Then 'Checks if the
"TopRight","MiddleCenter" and "BottomLeft" strings are "X"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        End If
        If WonRound = True Then 'Checks if the boolean variable "WonRound" is set to True
            PlayerOneTokenNo = PlayerOneTokenNo + 1 'Increments the "PlayerOneTokenNo"
integer variable
            TokenSoundStream() 'Calls the "TokenSoundStream" subroutine
            DisplayPlayerOneWinner() 'Calls the "DisplayPlayerOneWinner" subroutine
            GetTokens() 'Calls the "GetTokens" subroutine
            CheckDraw() 'Calls the "CheckDraw" subroutine
            If Draw = True Then 'Checks if the boolean variable "Draw" is set to true
                Draw = False 'Sets the boolean variable "Draw" to False
            End If
        Else
            PlayerOne = False 'Sets the boolean variable "PlayerOne" to False
            CheckDraw() 'Calls the "CheckDraw" subroutine
        End If
    End Sub
    Private Sub CheckPlayerTwoTurn() 'CheckPlayerTwoTurn Subroutine code
        ExitFlag = False 'Sets the boolean variable "ExitFlag" to False
        WonRound = False 'Sets the boolean variable "WonRound" to False
        Draw = False 'Sets the boolean variable "Draw" to False
        If TopLeft = "O" And MiddleCenter = "O" And BottomRight = "O" Then 'Checks if the
"TopLeft","MiddleCenter" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "O" And TopCenter = "O" And TopRight = "O" Then 'Checks if the
"TopLeft","TopCenter" and "TopRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf MiddleLeft = "O" And MiddleCenter = "O" And MiddleRight = "O" Then 'Checks if
the "MiddleLeft","MiddleCenter" and "MiddleRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf BottomLeft = "O" And BottomCenter = "O" And BottomRight = "O" Then 'Checks if
the "BottomLeft","BottomCenter" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopLeft = "O" And MiddleLeft = "O" And BottomLeft = "O" Then 'Checks if the
"TopLeft","MiddleLeft" and "BottomLeft" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopCenter = "O" And MiddleCenter = "O" And BottomCenter = "O" Then 'Checks if
the "TopCenter","MiddleCenter" and "BottomCenter" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
```

```
        ElseIf TopRight = "O" And MiddleRight = "O" And BottomRight = "O" Then 'Checks if the
"TopRight","MiddleRight" and "BottomRight" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        ElseIf TopRight = "O" And MiddleCenter = "O" And BottomLeft = "O" Then 'Checks if the
"TopRight","MiddleCenter" and "BottomLeft" strings are "O"
            WonRound = True 'Sets the boolean variable "WonRound" to True
        End If
        If WonRound = True Then 'Checks if the boolean variable "WonRound" to True
            PlayerTwoTokenNo = PlayerTwoTokenNo + 1 'Increments the "ComputerPlayerTokenNo"
integer variable
            TokenSoundStream() 'Calls the "TokenSoundStream" subroutine
            DisplayPlayerTwoWinner() 'Calls the "DisplayPlayerTwoWinner" subroutine
            GetTokens() 'Calls the "GetTokens" subroutine
        End If
        PlayerOne = True 'Sets the boolean variable "PlayerOne" to True
        CheckDraw() 'Calls the "CheckDraw" subroutine
    End Sub
    Private Sub DisplayPlayerOneWinner() 'DisplayPlayerOneWinner Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picPlayerOneStatus.Visible = True 'Shows the "picPlayerOneToken" picture box
        DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
    End Sub
    Private Sub DisplayPlayerTwoWinner() 'DisplayPlayerTwoWinner Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picPlayerTwoStatus.Visible = True 'Shows the "picComputerPlayerToken" picture box
        DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
    End Sub
    Private Sub DisableButtons() 'DisableButtons Subroutine code
        picTopLeft.Enabled = False 'Disables the "btnTopLeft" button
        picMiddleLeft.Enabled = False 'Disables the "btnMiddleLeft" button
        picBottomLeft.Enabled = False 'Disables the "btnBottomLeft" button
        picTopCenter.Enabled = False 'Disables the "btnTopCenter" button
        picMiddleCenter.Enabled = False 'Disables the "btnMiddleCenter" button
        picBottomCenter.Enabled = False 'Disables the "btnBottomCenter" button
        picTopRight.Enabled = False 'Disables the "btnTopRight" button
        picMiddleRight.Enabled = False 'Disables the "btnMiddleRight" button
        picBottomRight.Enabled = False 'Disables the "btnBottomRight" button
    End Sub
    Private Sub EnableButtons() 'EnableButtons Subroutine code
        picTopLeft.Enabled = True 'Enables the "btnTopLeft" button
        picMiddleLeft.Enabled = True 'Enables the "btnMiddleLeft" button
        picBottomLeft.Enabled = True 'Enables the "btnBottomLeft" button
        picTopCenter.Enabled = True 'Enables the "btnTopCenter" button
        picMiddleCenter.Enabled = True 'Enables the "btnMiddleCenter" button
        picBottomCenter.Enabled = True 'Enables the "btnBottomCenter" button
        picTopRight.Enabled = True 'Enables the "btnTopRight" button
        picMiddleRight.Enabled = True 'Enables the "btnMiddleRight" button
        picBottomRight.Enabled = True 'Enables the "btnBottomRight" button
    End Sub
    Private Sub RefreshGame() 'RefreshGame Subroutine code
        DisableButtons() 'Calls the "DisableButtons" subroutine
        ClearBoard() 'Calls the "ClearBoard" subroutine
        ClearBoardStrings() 'Calls the "ClearBoardStrings" subroutine
        EnableButtons() 'Calls the "EnableButtons" subroutine
        ResetBooleans() 'Calls the "ResetBooleans" subroutine
    End Sub
    Private Sub RestartGame() 'RestartGame Subroutine code
        ClearBoard() 'Calls the "ClearBoard" subroutine
        ClearBoardStrings() 'Calls the "ClearBoardStrings" subroutine
        ClearTokens() 'Calls the "ClearTokens" subroutine
        ResetBooleans() 'Calls the "ResetBooleans" subroutine
        EnableButtons() 'Calls the "EnableButtons" subroutine
        PlayerOneTokenNo = 0 'Sets the integer variable "PlayerOneTokenNo" to 0
        PlayerTwoTokenNo = 0 'Sets the integer variable "ComputerPlayerTokenNo" to 0
        lblPlayerOneTokens.Text = "Tokens: 0"
        lblPlayerTwoTokens.Text = "Tokens: 0"
```

```
        picTicTacToeWinner.Visible = False 'Hides the "picTicTacToeWinner" picture box
    End Sub
    Private Sub ResetBooleans() 'ResetBooleans Subroutine code
        PlayerOne = True 'Sets the boolean variable "PlayerOne" to True
        Draw = False 'Sets the boolean variable "Draw" to False
        PlayerTwoWinner = False 'Sets the boolean variable "PlayerTwoWinner" to False
        PlayerOneWinner = False 'Sets the boolean variable "PlayerOneWinner" to False
    End Sub
    Private Sub ClearBoardStrings() 'ClearBoardStrings Subroutine code
        TopLeft = "" 'Clears the value of the string variable "Topleft"
        MiddleLeft = "" 'Clears the value of the string variable "MiddleLeft"
        BottomLeft = "" 'Clears the value of the string variable "BottomLeft"
        TopCenter = "" 'Clears the value of the string variable "TopCenter"
        MiddleCenter = "" 'Clears the value of the string variable "MiddleCenter"
        BottomCenter = "" 'Clears the value of the string variable "BottomCenter"
        TopRight = "" 'Clears the value of the string variable "TopRight"
        MiddleRight = "" 'Clears the value of the string variable "MiddleRight"
        BottomRight = "" 'Clears the value of the string variable "BottomRight"
    End Sub
    Private Sub ClearBoard() 'ClearBoard Subroutine code
        picTopLeft.BackgroundImage = Nothing 'Sets the "picTopLeft" picture box's background
image to Nothing
        picMiddleLeft.BackgroundImage = Nothing 'Sets the "picMiddleLeft" picture box's
background image to Nothing
        picBottomLeft.BackgroundImage = Nothing 'Sets the "picBottomLeft" picture box's
background image to Nothing
        picTopCenter.BackgroundImage = Nothing 'Sets the "picTopCenter" picture box's
background image to Nothing
        picMiddleCenter.BackgroundImage = Nothing 'Sets the "picMiddleCenter" picture box's
background image to Nothing
        picBottomCenter.BackgroundImage = Nothing 'Sets the "picBottomCenter" picture box's
background image to Nothing
        picTopRight.BackgroundImage = Nothing 'Sets the "picTopRight" picture box's
background image to Nothing
        picMiddleRight.BackgroundImage = Nothing 'Sets the "picMiddleRight" picture box's
background image to Nothing
        picBottomRight.BackgroundImage = Nothing 'Sets the "picBottomRight" picture box's
background image to Nothing
    End Sub
    Private Sub ClearTokens() 'ClearTokens Subroutine code
        picPlayer1_GoldToken1.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken1" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken2.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken2" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken3.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken3" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken4.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken4" picture box's background image to the "No_Token" image in resources
        picPlayer1_GoldToken5.BackgroundImage = My.Resources.No_Token 'Sets the
"picPlayer1_GoldToken5" picture box's background image to the "No_Token" image in resources
        picComputerPlayer_GoldToken1.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken1" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken2.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken2" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken3.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken3" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken4.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken4" picture box's background image to the "No_Token" image in
resources
        picComputerPlayer_GoldToken5.BackgroundImage = My.Resources.No_Token 'Sets the
"picComputerPlayer_GoldToken5" picture box's background image to the "No_Token" image in
resources
    End Sub
```

```
    Private Sub GetTokens() 'GetTokens Subroutine code
        If PlayerOneTokenNo > 0 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 0
            picPlayer1_GoldToken1.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken1" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 1 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 1
            picPlayer1_GoldToken2.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken2" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 2 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 2
            picPlayer1_GoldToken3.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken3" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 3 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 3
            picPlayer1_GoldToken4.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken4" picture box's background image to the "Gold_Token" image in resources
        End If
        If PlayerOneTokenNo > 4 Then 'Checks if the integer variable "PlayerOneTokenNo" is
greater than 4
            picPlayer1_GoldToken5.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picPlayer1_GoldToken5" picture box's background image to the "Gold_Token" image in resources
        End If
        lblPlayerOneTokens.Text = "Tokens: " & PlayerOneTokenNo
        If PlayerTwoTokenNo > 0 Then 'Checks if the integer variable "ComputerPlayerTokenNo"
is greater than 0
            picComputerPlayer_GoldToken1.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken1" picture box's background image to the "Gold_Token" image in
resources
        End If
        If PlayerTwoTokenNo > 1 Then 'Checks if the integer variable "ComputerPlayerTokenNo"
is greater than 1
            picComputerPlayer_GoldToken2.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken2" picture box's background image to the "Gold_Token" image in
resources
        End If
        If PlayerTwoTokenNo > 2 Then 'Checks if the integer variable "ComputerPlayerTokenNo"
is greater than 2
            picComputerPlayer_GoldToken3.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken3" picture box's background image to the "Gold_Token" image in
resources
        End If
        If PlayerTwoTokenNo > 3 Then 'Checks if the integer variable "ComputerPlayerTokenNo"
is greater than 3
            picComputerPlayer_GoldToken4.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken4" picture box's background image to the "Gold_Token" image in
resources
        End If
        If PlayerTwoTokenNo > 4 Then 'Checks if the integer variable "ComputerPlayerTokenNo"
is greater than 4
            picComputerPlayer_GoldToken5.BackgroundImage = My.Resources.Gold_Token 'Sets the
"picComputerPlayer_GoldToken5" picture box's background image to the "Gold_Token" image in
resources
        End If
        lblPlayerTwoTokens.Text = "Tokens: " & PlayerTwoTokenNo
    End Sub
    Private Sub CheckDraw() 'CheckDraw Subroutine code
        If WonRound = False And picTopLeft.Enabled = False And picMiddleLeft.Enabled = False
And picBottomLeft.Enabled = False And picTopCenter.Enabled = False And
picMiddleCenter.Enabled = False And picBottomCenter.Enabled = False And picTopRight.Enabled =
False And picMiddleRight.Enabled = False And picBottomRight.Enabled = False Then 'Checks if
all the pictureboxes are disabled
            Draw = True 'Sets the boolean variable "Draw" to True
```

```
        End If
        CheckWinner() 'Calls the "CheckWinner" subroutine
    End Sub
    Private Sub CheckWinner() 'CheckWinner Subroutine code
        If PlayerOneTokenNo = 5 Then
            GetTokens() 'Calls the "GetTokens" subroutine
            PlayerOneTokenNo = 0
            PlayerTwoTokenNo = 0
            Winner = True 'Sets the boolean variable "Winner" to True
            PlayerOneWinner = True 'Sets the boolean variable "PlayerOneWinner" to True
        ElseIf PlayerTwoTokenNo = 5 Then
            GetTokens() 'Calls the "GetTokens" subroutine
            Winner = True 'Sets the boolean variable "Winner" to True
            PlayerTwoWinner = True 'Sets the boolean variable "PlayerTwoWinner" to True
        End If
        DisplayWinner() 'Calls the "DisplayWinner" subroutine
        If Draw = True Then 'Checks if the boolean variable "Draw" is set to True
            DisableButtons() 'Calls the "DisableButtons" subroutine
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            picTicTacToeWinner.BackgroundImage = My.Resources.Player_Draw 'Changes the
"picTicTacToeWinner" picture box's background image to "Player_Draw" in resources
            picTicTacToeWinner.Visible = True 'Shows the "picTicTacToeWinner" picture box
            DisplayTimer.Start() 'Starts the "DisplayTimer" timer tick event
        End If
    End Sub
    Private Sub DisplayWinner() 'DisplayWinner Subroutine code
        If Winner = True Then 'Checks if a player has won the game
            DisplayTimer.Stop() 'Stops the "DisplayTimer" timer tick event
            picTicTacToeWinner.BackgroundImage = My.Resources.TicTacToe_GameOver
            picPlayerOneStatus.Visible = True 'Hides the "picPlayerOneToken" picturebox
            picPlayerTwoStatus.Visible = True 'Hides the "picComputerPlayerToken" picturebox
            If PlayerOneWinner = True Then 'Checks if player one has won the game
                WinnerSoundTimer.Start() 'Starts the sound timer tick event
                picTicTacToeWinner.Visible = True 'Makes the display visible
                picPlayerOneStatus.BackgroundImage = My.Resources.TicTacToe_PlayerOne_Winner
                picPlayerTwoStatus.BackgroundImage = My.Resources.TicTacToe_PlayerTwo_Loser
            ElseIf PlayerTwoWinner = True Then 'Checks if player Two has won the game. In
this case, the computer player
                WinnerSoundTimer.Start() 'Starts the sound timer tick event
                picTicTacToeWinner.Visible = True 'Makes the display visible
                picPlayerTwoStatus.BackgroundImage = My.Resources.TicTacToe_PlayerTwo_Winner
                picPlayerOneStatus.BackgroundImage = My.Resources.TicTacToe_PlayerOne_Loser
            End If
            btnStart.Visible = True 'Shows the "btnStart" button
        End If
    End Sub
    Private Sub DisplayTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DisplayTimer.Tick 'DisplayTimer Timer tick code
        RefreshGame() 'Calls the "RefreshGame" subroutine
        Draw = False  'Sets the boolean variable "Draw" to True
        picTicTacToeWinner.Visible = False 'Hides the "picTicTacToeWinner" picturebox
        picPlayerOneStatus.Visible = False 'Hides the "picPlayerOneToken" picturebox
        picPlayerTwoStatus.Visible = False 'Hides the "picComputerPlayerToken" picturebox
        DisplayTimer.Stop() 'Stops the "DisplayTimer" timer tick event
    End Sub
    Private Sub WinnerSoundTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles WinnerSoundTimer.Tick 'WinnerSoundTimer Timer tick code
        If Winner = False Then  'Checks if there is a winner
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
```

```
                frmMainMenu.player.Play() 'Sets the soundplayer to the "FormMinimizing" WAV
file in the resources
            End If
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
            Winner = False  'Sets the boolean variable "Winner" to False
        Else
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_setback 'Sets the player's
stream to "sound_setback" inresources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Winner = False 'Sets the boolean variable "Winner" to False
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
        End If
        If Draw = True Then 'Checks if there is a draw
            If My.Settings.Mute = False Then 'Checks if "Mute" variable in application
settings is set to false
                frmMainMenu.player.Stream = My.Resources.sound_rejected 'Sets the player's
stream to "sound_rejected" inresources
                frmMainMenu.player.Play() 'Plays the sound file
            End If
            Draw = False 'Sets the boolean variable "Draw" to False
            WinnerSoundTimer.Stop() 'Stops the "WinnerSoundTimer" timer tick event
        End If
    End Sub
    Private Sub frmTicTacToeSinglePlayer_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmTicTacToeSingleplayer Form Load code
        If My.Settings.DisableCaptions = False Then 'Checks if the application setting's
boolean variable "DisableCaptions" is set to false
            MouseMoveTimer.Start() 'Start's the "MouseMoveTimer" timer
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            lblMute.Text = "Mute Sounds" 'Sets the "lblMute" label to "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        Else
            lblMute.Text = "UnMute Sounds" 'Sets the "lblMute" label to "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button 'Sets the
"btnMuteUnmute" background image to specified file in resources
        End If
        lblPlayerOne.ForeColor = Color.Black 'Sets the "lblPlayerOne" label's forecolor to
black
        lblPlayerTwo.ForeColor = Color.Black 'Sets the "lblPlayerTwo" label's forecolor to
black
        DisableButtons() 'Calls the "DisableButtons" subroutine
        picTicTacToeWinner.BackgroundImage = My.Resources.Press_Start 'Changes the image to
"Press Start"
        picTicTacToeWinner.Visible = True 'Shows the "picTicTacToeWinner" picture box
    End Sub
    Private Sub btnMinimize_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseDown 'btnMinimize Button
MouseDown code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Pushed 'Changes the
background image of the "btnMinimize" button when the mouse is down
    End Sub
    Private Sub btnMinimize_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseEnter 'btnMinimize Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
```

```
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button_Highlighted 'Changes the
background image of the "btnMinimize" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnMinimize_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnMinimize.MouseLeave 'btnMinimize Button MouseLeave code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnMinimize_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMinimize.MouseUp 'btnMinimize Button MouseUp
code
        btnMinimize.BackgroundImage = My.Resources.Minimize_Button 'Changes the "btnMinimize"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormMinimizing 'Sets the
soundplayer to the "FormMinimizing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.WindowState = FormWindowState.Minimized 'Minimizes the form by changing the form's
"FormWindowState" properties to "Minimized"
    End Sub
    'Exit
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed 'Changes the background
image of the "btnClose" button when the mouse is down
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
background image of the close button to highlighted when the curser enters the button
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to close the program?",
vbInformation + vbYesNo, "Close") 'Prompts the user before closing the program
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            End 'Closes the application
        End If
    End Sub
    Private Sub btnInfo_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseDown 'btnInfo Button MouseDown code
```

```
        btnInfo.BackgroundImage = My.Resources.Info_Button_Pushed 'Changes the background
image of the "btnInfo" button when the mouse is down
    End Sub
    Private Sub btnInfo_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseEnter 'btnInfo Button MouseEnter code
        lblInfo.Visible = True 'Shows the "lblInfo" label when mouse enters the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "MouseScrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnInfo.BackgroundImage = My.Resources.Info_Button_Highlighted 'Changes the
background image of the info button to highlighted when the curser enters the button
    End Sub
    Private Sub btnInfo_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnInfo.MouseLeave 'btnInfo Button MouseLeave code
        lblInfo.Visible = False 'Hides the "lblInfo" label when mouse leaves the picture box
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnInfo_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnInfo.MouseUp
        btnInfo.BackgroundImage = My.Resources.Info_Button 'Changes the "btnInfo" button's
background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Dim ProcessDirectory As String = AppPath & "\Information.ppsx" 'Declares
"ProcessDirectory" as a string and sets the string value to the path of the
"Information.ppsx" file
        System.Diagnostics.Process.Start(ProcessDirectory) 'Starts the slide show
    End Sub
    Private Sub btnSettings_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseDown 'btnSettings Button
MouseDown code
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Pushed 'Changes the
background image of the "btnSettings" button when the mouse is down
    End Sub
    Private Sub btnSettings_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseEnter 'btnSettings Button MouseEnter code
        lblSettings.Visible = True 'Shows the "lblSettings" label when mouse enters the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button_Highlighted 'Changes the
background image of the settings button to highlighted when the curser enters the button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnSettings_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnSettings.MouseLeave 'btnSettings Button MouseLeave code
        lblSettings.Visible = False 'Hides the "lblSettings" label when mouse leaves the
picture box
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnSettings_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnSettings.MouseUp 'btnSettings Button MouseUp
code
```

```
        btnSettings.BackgroundImage = My.Resources.Settings_Button 'Changes the "btnSettings"
button's background image to the original image when the curser has left the picture box
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        frmSettings.Show() 'Shows the "frmSettings" form
    End Sub
    'MuteUnMute
    Private Sub btnMuteUnMute_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseUp 'btnMuteUnMute Button
MouseUp code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            My.Settings.Mute = True 'Sets the "Mute" variable in application settings to True
            lblMute.Text = "UnMute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        Else
            My.Settings.Mute = False 'Sets the "Mute" variable in application settings to
False
            lblMute.Text = "Mute Sounds"
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnMuteUnMute.MouseDown 'btnMuteUnMute Button
MouseDown code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Pushed
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Pushed
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseEnter 'btnMuteUnMute Button MouseEnter code
        lblMute.Visible = True
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button_Highlighted
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button_Highlighted
        End If
    End Sub
    Private Sub btnMuteUnMute_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMuteUnMute.MouseLeave 'btnMuteUnMute Button MouseLeave code
        lblMute.Visible = False
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
    End Sub
    'Return
    Private Sub btnReturn_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseDown 'btnReturn Button MouseDown
code
        btnReturn.BackgroundImage = My.Resources.Return_Button_Pushed
    End Sub
```

```
    Private Sub btnReturn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseEnter 'btnReturn Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        lblLeaveGame.Visible = True
        btnReturn.BackgroundImage = My.Resources.Return_Button_Highlighted
    End Sub
    Private Sub btnReturn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReturn.MouseLeave 'btnReturn Button MouseLeave code
        lblLeaveGame.Visible = False
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnReturn_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnReturn.MouseUp 'btnReturn Button MouseUp code
        btnReturn.BackgroundImage = My.Resources.Return_Button 'Changes the "btnReturn"
button's background image to the original image when the curser has left the picture box
        If ExitFlag = False Then
            Dim MessageBoxResult As String
            MessageBoxResult = MsgBox("Are you sure you wish to exit the game?", vbYesNo,
"Exit Game")
            If MessageBoxResult = vbYes Then
                ExitFlag = True
            End If
        End If
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormReturning 'Sets the
soundplayer to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
            frmTicTacToeModeMenu.lblMute.Text = "Mute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.Mute_Button
        Else
            frmTicTacToeModeMenu.lblMute.Text = "UnMute Sounds"
            frmTicTacToeModeMenu.btnMuteUnMute.BackgroundImage = My.Resources.UnMute_Button
        End If
        frmTicTacToeModeMenu.Location = New Point(Me.Location.X + 119, Me.Location.Y)
        frmTicTacToeModeMenu.Show() 'Shows the "frmTicTacToeModeMenu" form
        Me.Dispose() 'Closes the current form
    End Sub
    'Start
    Private Sub btnStart_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseDown 'btnStart Button MouseDown
code
        btnStart.BackgroundImage = My.Resources.Start_Button_Pushed 'Changes the background
image of the "btnStart" button when the mouse is down
    End Sub
    Private Sub btnStart_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseEnter 'btnStart Button MouseEnter code
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        btnStart.BackgroundImage = My.Resources.Start_Button_Highlighted 'Changes the
background image of the "btnStart" button to highlighted when the curser enters the button
    End Sub
    Private Sub btnStart_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnStart.MouseLeave 'btnStart Button MouseLeave code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
```

```
    End Sub
    Private Sub btnStart_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnStart.MouseUp 'btnStart Button MouseUp code
        btnStart.BackgroundImage = My.Resources.Start_Button 'Changes the "btnStart" button's
background image to the original image when the mouse is set to up
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormSelect
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        picPlayerOneStatus.Visible = False 'Hides the "picPlayerOneToken" picturebox
        picPlayerTwoStatus.Visible = False 'Hides the "picComputerPlayerToken" picturebox
        picPlayerOneStatus.BackgroundImage = My.Resources.Token_Player
        picPlayerTwoStatus.BackgroundImage = My.Resources.Token_Player
        picTicTacToeWinner.Visible = False
        btnStart.Visible = False
        RestartGame()
    End Sub

    'Picture Boxes
    Private Sub btnTopLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopLeft.MouseUp 'picTopLeft MouseUp code
        If PlayerOne = True Then
            picTopLeft.Enabled = False
            picTopLeft.BackgroundImage = My.Resources.TicTacToe_X1
            TopLeft = "X"
            CheckPlayerOneTurn()
        Else
            picTopLeft.Enabled = False
            picTopLeft.BackgroundImage = My.Resources.TicTacToe_O
            TopLeft = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnTopCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopCenter.MouseUp 'picTopCenter MouseUp code
        If PlayerOne = True Then
            picTopCenter.Enabled = False
            picTopCenter.BackgroundImage = My.Resources.TicTacToe_X1
            TopCenter = "X"
            CheckPlayerOneTurn()
        Else
            picTopCenter.Enabled = False
            picTopCenter.BackgroundImage = My.Resources.TicTacToe_O
            TopCenter = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnTopRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopRight.MouseUp 'picTopRight MouseUp code
        If PlayerOne = True Then
            picTopRight.Enabled = False
            picTopRight.BackgroundImage = My.Resources.TicTacToe_X1
            TopRight = "X"
            CheckPlayerOneTurn()
        Else
            picTopRight.Enabled = False
            picTopRight.BackgroundImage = My.Resources.TicTacToe_O
            TopRight = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnMiddleLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleLeft.MouseUp 'picMiddleLeft MouseUp code
        If PlayerOne = True Then
            picMiddleLeft.Enabled = False
```

```
            picMiddleLeft.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleLeft = "X"
            CheckPlayerOneTurn()
        Else
            picMiddleLeft.Enabled = False
            picMiddleLeft.BackgroundImage = My.Resources.TicTacToe_O
            MiddleLeft = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnMiddleCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseUp 'picMiddleCenter MouseUp code
        If PlayerOne = True Then
            picMiddleCenter.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleCenter = "X"
            CheckPlayerOneTurn()
            picMiddleCenter.Enabled = False
        Else
            picMiddleCenter.Enabled = False
            picMiddleCenter.BackgroundImage = My.Resources.TicTacToe_O
            MiddleCenter = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnMiddleRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseUp 'picMiddleRight MouseUp code
        If PlayerOne = True Then
            picMiddleRight.Enabled = False
            picMiddleRight.BackgroundImage = My.Resources.TicTacToe_X1
            MiddleRight = "X"
            CheckPlayerOneTurn()
        Else
            picMiddleRight.Enabled = False
            picMiddleRight.BackgroundImage = My.Resources.TicTacToe_O
            MiddleRight = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnBottomLeft_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomLeft.MouseUp 'picBottomLeft MouseUp code
        If PlayerOne = True Then
            picBottomLeft.Enabled = False
            picBottomLeft.BackgroundImage = My.Resources.TicTacToe_X1
            BottomLeft = "X"
            CheckPlayerOneTurn()
        Else
            picBottomLeft.Enabled = False
            picBottomLeft.BackgroundImage = My.Resources.TicTacToe_O
            BottomLeft = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    Private Sub btnBottomCenter_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseUp 'picBottomCenter MouseUp code
        If PlayerOne = True Then
            picBottomCenter.Enabled = False
            picBottomCenter.BackgroundImage = My.Resources.TicTacToe_X1
            BottomCenter = "X"
            CheckPlayerOneTurn()
        Else
            picBottomCenter.Enabled = False
            picBottomCenter.BackgroundImage = My.Resources.TicTacToe_O
            BottomCenter = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
```

```
    Private Sub btnBottomRight_MouseUp(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseUp 'picBottomRight MouseUp code
        If PlayerOne = True Then
            picBottomRight.Enabled = False
            picBottomRight.BackgroundImage = My.Resources.TicTacToe_X1
            BottomRight = "X"
            CheckPlayerOneTurn()
        Else
            picBottomRight.Enabled = False
            picBottomRight.BackgroundImage = My.Resources.TicTacToe_O
            BottomRight = "O"
            CheckPlayerTwoTurn()
        End If
    End Sub
    'Tic Tac Toe Board Animations
    'Top Left
    Private Sub btnTopLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopLeft.MouseEnter 'picTopLeft MouseEnter code
        If PlayerOne = True Then
            picTopLeft.Image = My.Resources.Selected_Box_Green
        Else
            picTopLeft.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnTopLeft_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picTopLeft.MouseLeave 'picTopLeft MouseLeave code
        picTopLeft.Image = Nothing
    End Sub
    'Top Center
    Private Sub btnTopCenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopCenter.MouseEnter 'picTopCenter MouseEnter code
        If PlayerOne = True Then
            picTopCenter.Image = My.Resources.Selected_Box_Green
        Else
            picTopCenter.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnTopCenter_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picTopCenter.MouseLeave 'picTopCenter MouseLeave code
        picTopCenter.Image = Nothing
    End Sub
    'Top Right
    Private Sub btnTopRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picTopRight.MouseEnter 'picTopRight MouseEnter code
        If PlayerOne = True Then
            picTopRight.Image = My.Resources.Selected_Box_Green
        Else
            picTopRight.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnTopRight_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picTopRight.MouseLeave 'picTopRight MouseLeave code
        picTopRight.Image = Nothing
    End Sub
    'Middle Left
    Private Sub btnMiddleLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleLeft.MouseEnter 'picMiddleLeft MouseEnter code
        If PlayerOne = True Then
            picMiddleLeft.Image = My.Resources.Selected_Box_Green
        Else
            picMiddleLeft.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnMiddleLeft_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picMiddleLeft.MouseLeave 'picMiddleLeft MouseLeave code
        picMiddleLeft.Image = Nothing
```

```
    End Sub
    'Middle Center
    Private Sub btnMiddlecenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseEnter, picMiddleCenter.MouseEnter
        If PlayerOne = True Then
            picMiddleCenter.Image = My.Resources.Selected_Box_Green
        Else
            picMiddleCenter.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnMiddleCenter_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picMiddleCenter.MouseLeave
        picMiddleCenter.Image = Nothing
    End Sub
    'Middle Right
    Private Sub btnMiddleRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseEnter, picMiddleRight.MouseEnter
        If PlayerOne = True Then
            picMiddleRight.Image = My.Resources.Selected_Box_Green
        Else
            picMiddleRight.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnMiddleRight_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picMiddleRight.MouseLeave
        picMiddleRight.Image = Nothing
    End Sub
    'Bottom Left
    Private Sub btnBottomLeft_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomLeft.MouseEnter, picBottomLeft.MouseEnter
        If PlayerOne = True Then
            picBottomLeft.Image = My.Resources.Selected_Box_Green
        Else
            picBottomLeft.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnBottomLeft_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles picBottomLeft.MouseLeave
        picBottomLeft.Image = Nothing
    End Sub
    'Bottom Center
    Private Sub btnBottomCenter_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseEnter, picBottomCenter.MouseEnter
        If PlayerOne = True Then
            picBottomCenter.Image = My.Resources.Selected_Box_Green
        Else
            picBottomCenter.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnBottomCenter_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picBottomCenter.MouseLeave
        picBottomCenter.Image = Nothing
    End Sub
    'Bottom Right
    Private Sub btnBottomRight_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseEnter, picBottomRight.MouseEnter
        If PlayerOne = True Then
            picBottomRight.Image = My.Resources.Selected_Box_Green
        Else
            picBottomRight.Image = My.Resources.Selected_Box_Red
        End If
    End Sub
    Private Sub btnBottomRight_MouseLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles picBottomRight.MouseLeave
        picBottomRight.Image = Nothing
    End Sub
```

```
    Private Sub frmTicTacToeTwoPlayer_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles MyBase.KeyPress
        If Asc(e.KeyChar) = 13 And btnStart.Visible = True Then
            btnStart_MouseUp(Nothing, Nothing)
        End If
    End Sub
    Private Sub MouseMoveTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MouseMoveTimer.Tick 'MouseMoveTimer Timer Tick code
        If Me.PointToClient(Control.MousePosition).ToString() = CurrentMousePosition Then
'Checks if the mouse's location on the screen is the same as it was before using the string
variable "CurrentMousePosition
            CaptionTimer.Start() 'Starts "CaptionTimer" timer
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        Else
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            If CaptionCounter > 3 Then 'Checks if the "CaptionCounter" integer value is
greater than 3
                lblLeaveGame.Visible = False 'Hides the "lblLeaveGame" label
                lblInfo.Visible = False 'Hides the "lblInfo" label
                lblMute.Visible = False 'Hides the "lblMute" label
                lblSettings.Visible = False 'Hides the "lblSettings" label
            End If
            CaptionCounter = 0 'Sets the "CaptionCounter" integer value to 0
            CurrentMousePosition = Me.PointToClient(Control.MousePosition).ToString() 'Sets
the "CurrentMousePosition" string variable to the same number aas the location of the mouse
        End If
        If CaptionCounter > 5 Then 'Checks if the "CaptionCounter" integer value is greater
than 5
            CaptionTimer.Stop() 'Stops the "CaptionTimer" timer
            lblLeaveGame.Visible = True 'Shows the "lblLeaveGame" label
            lblInfo.Visible = True 'Shows the "lblInfo" label
            lblSettings.Visible = True 'Shows the "lblSettings" label
            If btnMuteUnMute.Visible = True Then 'Checks if the "btnMuteUnMute" picturebox is
showing
                lblMute.Visible = True 'Shows the "lblMute" label
            End If
        End If
    End Sub
    Private Sub CaptionTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CaptionTimer.Tick 'CaptionTimer Timer Tick code
        CaptionCounter = CaptionCounter + 1 'Increments "CaptionCounter" integer variable
    End Sub
End Class
```

8.10. Game Settings:

```
Imports System.IO 'Form makes reference to the "System.IO" namespace
Public Class frmGameSettings 'frmGameSettings form code
    Dim Word As String
    Dim Reader As IO.StreamReader
    Dim Valid As Boolean = True
    Dim SelectedIndex As Integer
    Dim DefaultArray(10000) As String
    Dim DefaultRandomArray(10) As String
    Dim NotValid As Boolean = False
    Dim ValidWord As Boolean = False
    Dim ClearList As Boolean = False
    Dim CancelExit As Boolean = False
    Dim Filenum As Integer = FreeFile() 'Declares "Filenum" as an integer variable obtaining
the next available file number using "FreeFile()"
    Dim txtwords As System.IO.StreamReader
    Dim FileWriter As System.IO.StreamWriter
    Dim appPath As String = Application.StartupPath() 'Gets the path of where the application
started from (file specified)
```

```
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Sub ValidTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ValidTimer.Tick 'ValidTimer Timer Tick code
        SelectedIndex = lstWordList.SelectedIndex 'Sets the SelectedIndex variable to the
selected item in the listbox
        If My.Settings.HangmanDefaultGameList = False Then 'Checks if the
"HangmanDefaultGameList" boolean variable in application settings is set to False
            If lblWordsAdded.Text <= 10 Then 'Checks if the label has the number "10" or
below is displayed.
                lblWordsAdded.ForeColor = Color.DarkRed 'Sets the label's text color to Dark
Red
                btnRemove.Enabled = False 'Disables remove button
                btnRemove.BackColor = Color.Gray 'Sets the remove button's background color
to gray
            Else 'Executes a set of commands if the list box contains more than 10 words
                lblWordsAdded.ForeColor = Color.Black 'Sets the label's text color to Black
                btnRemove.Enabled = True 'Enables the remove button
                btnRemove.BackColor = Color.Firebrick 'Sets the remove button's background
color to Firebrick
            End If
        End If
        If txtWord.Text.Length < 3 Then 'Checks if the user's entered word's length is less
than 3 characters
            picValid.BackgroundImage = Nothing 'The green or red tick is not shown if the
word's length is below 3 characters
            NotValid = True 'The "NotValid" boolean variable is set to true
        Else
            Valid = True 'Sets "Valid" boolean variable to True
            If txtWord.Text.Length > 21 Then 'Checks if the entered word length is greater
than 21 characters
                Valid = False  'Sets "Valid" boolean variable to False
            End If
            For Symbol = 1 To 27 'Repeates execution of commands 27 times
                If txtWord.Text.Contains(lstSymbol.Items(Symbol)) Then 'Checks the invisible
symbol listbox for characters to compare with the user's entered word
```

```
                    Valid = False 'Sets "Valid" boolean variable to false
                End If
                If Valid = False Then 'Checks if the "Valid" boolean variable is set to false
                    Exit For 'Exits the For statement immediately
                End If
            Next
            For Number = 0 To 9 'Executes set of commands 10 times (0-9)
                If txtWord.Text.Contains(Number) Then 'Checks if the user has entered numbers
into the textbox
                    Valid = False 'Sets "Valid" boolean variable to false
                End If
                If Valid = False Then 'Checks if the boolean variable has been set to false
                    Exit For 'Exits the For Next statement
                End If
            Next
            If Valid = False Then 'Checks if the boolean variable has been set to false
                picValid.BackgroundImage = My.Resources.Red_Cross 'Changes the picturebox's
background image to a red cross
                NotValid = True 'Sets boolean variable "Not Valid" to true
                Valid = True 'Sets boolean variable "Valid" to true so that the timer can
start whole execution every tick
            Else
                picValid.BackgroundImage = My.Resources.Green_Tick 'Changes the picturebox's
background image to a green tick
                NotValid = False 'Sets boolean variable "Not Valid" to false
                Valid = True 'Sets boolean variable "Valid" to true so that the timer can
start whole execution every tick
            End If
        End If
    End Sub
    Private Sub txtword_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtWord.KeyPress 'txtWord Textbox Keypress
code
        ValidTimer.Start() 'Starts ValidTimer
        If Asc(e.KeyChar) = 13 Then 'Checks if the enter key has been pressed
            Call btnInsert_Click(Nothing, Nothing) 'Calls the insert button's click event
handler
        End If
    End Sub
    Private Sub txtWord_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles txtWord.KeyDown 'txtWord Textbox KeyDown code
        If e.KeyCode = Keys.Up Then 'Checks if the up arrow key has been pressed whilst the
textbox is focused
            If lstWordList.SelectedIndex < 1 Then 'Checks if the selected word's index is
below 1
                Exit Sub 'Exits the sub if the above statement is true
            Else
                lstWordList.SelectedIndex = SelectedIndex - 1  'Changes selection to the next
word up in the listbox
            End If
        End If
        If e.KeyCode = Keys.Down Then 'Checks if the down arrow key has been pressed whilst
the textbox is focused
            If lstWordList.SelectedIndex > lstWordList.Items.Count - 2 Then 'Checks if the
selected word is the last one in the list
                Exit Sub 'Exits the sub if the above statement is true
            Else
                lstWordList.SelectedIndex = SelectedIndex + 1 'Changes selection to the next
word down in the listbox
            End If
        End If
    End Sub
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
```

```
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed 'Changes the background
image of the "btnClose" button when the mouse is down
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
background image of the "btnClose" button to highlighted when the curser enters the button
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button Mouseleave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the "btnClose" button's
background image to the original image when the mouse is set to up
        If My.Settings.Mute = False Then 'Checks if "Mute" variable in application settings
is set to false
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "FormClosing" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        Me.Hide() 'Hides the form once the "Close" button has been pressed
    End Sub
    Private Sub btnInsert_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnInsert.Click 'btnInsert Button Click code
        CancelExit = True 'The CancelExit flag is set to true so that the messageboxes
doesn't close the form upon deactivating
        If txtWord.Text = "" Then 'Checks if the textbox is empty
            MsgBox("You have not yet entered a word. Please enter a valid word into the text
box to proceed.", vbInformation, "Invalid Word") 'Prompts the user to enter a word into the
textbox
            CancelExit = False 'Sets the boolean variable back to False
            txtWord.Focus() 'Places the curser into the text box
        ElseIf NotValid = True Then 'Checks if the boolean variable "Not Valid" is set to
true
            MsgBox("The word you have entered is invalid. Valid words must consist of only
alphabetical characters. Do not add any symbols or numbered values. The length of the word
must be between 3 to 21 characters Please try again.", vbExclamation, "Invalid Word")
'Prompts the user to enter a valid word
            CancelExit = False 'Sets the boolean variable back to False
            txtWord.Focus() 'Places the curser into the text box
        Else
            SaveAnimation() 'Calls the "SaveAnimation" subroutine
            Word = StrConv(txtWord.Text, VbStrConv.ProperCase) 'Converts the word in the
textbox to lowercase except for the first character and places the word into the "Word"
string variable
            Word = Word.Trim() 'Removes white spaces from the "Word" string
            Do While Word.IndexOf("  ") <> -1 'Executes a set of commands until there are no
more double spaces in the "Word" string
                Word = Word.Replace("  ", " ") 'Replaces every double space is one space only
            Loop
            FileWriter = File.AppendText("CustomWordList.txt") 'FileWriter is told to append
text into the text document "CustomWordList"
            FileWriter.WriteLine(Word) 'FileWriter writes word in the "Word" string into the
text document
            FileWriter.Close() 'closes the File Writer
            GetCustomWordList() 'Calls the "GetCustomWordListtxt" subroutine
            txtWord.Text = "" 'Clears the textbox
```

```
            lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of
the amount of words that have been added to the list box
            CancelExit = False 'Sets the "CancelExit" boolean variable back to true
            txtWord.Focus() 'Places the curser into the text box
        End If
    End Sub
    Private Sub btnRemove_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnRemove.Click 'btnRemove Button Click code
        If lstWordList.SelectedIndex = -1 Then 'Checks if nothing has been selected in the
list box
            Exit Sub 'Exits the click event
        Else
            SaveAnimation() 'Calls the "SaveAnimation" subroutine
            lstWordList.Items.RemoveAt(lstWordList.SelectedIndex) 'Removes selected item from
the list
            File.WriteAllText(appPath & "\CustomWordList.txt", "") 'Clears the entire text
document
            FileWriter = File.AppendText("CustomWordList.txt") 'FileWriter is told to append
text into the text document "CustomWordList"
            For i = 0 To lstWordList.Items.Count - 1 'Executes commands several times until
the value of "i" reaches the maximum count of the words in the listbox
                FileWriter.WriteLine(lstWordList.Items(i)) 'Writes the new set of words in
the listbox to the text document
            Next
            FileWriter.Close() 'closes the File Writer
            lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of
the amount of words that have been added to the list box
            lstWordList.SelectedIndex = SelectedIndex - 1 'Sets the selected word in the
listbox to the one it was originally, other than moving to the next word down in the listbox
            txtWord.Focus() 'Places the curser into the text box
        End If
    End Sub
    Private Sub btnCleanList_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCleanList.Click 'btnCleanList Button Click code
        CancelExit = True 'The CancelExit flag is set to true so that the messageboxes
doesn't close the form upon deactivating
        Dim MessageBoxResult As String 'Declares "MessageboxResult" as a string
        MessageBoxResult = MsgBox("Are you sure you want to clean the list? Note: This will
add 10 random words from the default list to the custom list.", vbYesNo + vbInformation,
"Clear All") 'The "MessageBoxResult" variable gathers the user's decision from the message
box, whether it be yes/no
        txtWord.Focus() 'Places the curser into the text box
        If MessageBoxResult = vbYes Then 'Checks if the user's decision from the message box
was "Yes"
            SaveAnimation() 'Calls the "SaveAnimation" subroutine
            GetDefaultWords() 'Calls the "GetDefaultWords" subroutine
            lstWordList.Items.Clear() 'Clears all items in the listbox
            FileOpen(Filenum, appPath & "\CustomWordList.txt", OpenMode.Output) 'Opens the
next available file in the destination specified with "write" access
            FileClose() 'Closes the file immediately, erasing all of its contents
            For i = 0 To 9 'Executes a set of commands 10 times (0-9)
                FileWriter = File.AppendText("CustomWordList.txt") 'Sets "FileWriter"
variable to appendtext of the "CustomWordList" text document
                FileWriter.WriteLine(DefaultRandomArray(i)) 'Adds words line-by-line in the
text document through "DefaultArray" as well as the index of the word in the array
                FileWriter.Close() 'closes the File Writer
                lstWordList.Items.Add(DefaultRandomArray(i)) 'Adds the word with the index
"i" in the "DefaultArray" to the "Word" list box
            Next
            lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of
the amount of words that have been added to the list box
        End If
        CancelExit = False 'The CancelExit flag is set back to False
    End Sub
    Private Sub btnReplace_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnReplace.Click 'btnReplace Button Click code
```

```
        CancelExit = True 'The CancelExit flag is set to true so that the messageboxes
doesn't close the form upon deactivating
        If lstWordList.SelectedIndex = -1 Then 'Checks if nothing is selected in the listbox
            MsgBox("Please select the word you wish to replace.", vbExclamation + vbOKOnly,
"Word not selected") 'Displays messagebox prompting the user to select a word in the listbox
            CancelExit = False 'The CancelExit flag is set back to false
            txtWord.Focus() 'Places the curser into the text box
        ElseIf txtWord.Text = "" Then 'Checks if the textbox is empty
            MsgBox("You have not yet entered a word. Please enter a valid word into the text
box to proceed.", vbInformation, "Invalid Word") 'Prompts the user to enter a word into the
textbox
            CancelExit = False 'The CancelExit flag is set back to false
            txtWord.Focus() 'Places the curser into the text box
        ElseIf NotValid = True Then 'Checks if the "NotValid" variable has been set to True,
meaning that the word is not valid
            MsgBox("The word or letter you have entered is invalid. Please insert a valid
word to replace your selected word." & vbCrLf & vbCrLf & "Selected Word: " &
lstWordList.SelectedItem & vbCrLf & "Invalid word entered: " & txtWord.Text, vbExclamation +
vbOKOnly, "Invalid Word") 'Displays a messagebox prompting the user to enter a valid word.
This also displays the incorrect word that the user has entered
            CancelExit = False 'The CancelExit flag is set back to false
            txtWord.Focus() 'Places the curser into the text box
        Else
            SaveAnimation() 'Calls the "SaveAnimation" subroutine
            lstWordList.Items.RemoveAt(SelectedIndex) 'Removes the selected word in the
listbox
            Word = StrConv(txtWord.Text, VbStrConv.ProperCase) 'Converts all the characters
in the textbox to lowercase except for the first character and places it inside the "Word"
string variable
            lstWordList.Items.Insert(SelectedIndex, Word) 'Inserts "Word" string variable
into selected index of the listbox
            OverwriteCustomWordList() 'Calls the Overwrite CustomWordList subroutine
            lstWordList.SelectedIndex = SelectedIndex 'Sets the selected word in the listbox
to the one it was originally, other than moving to the next word down in the listbox
            txtWord.Text = "" 'Clears the textbox
            txtWord.Focus() 'Focuses on the textbox
            CancelExit = False 'The CancelExit flag is set back to false
        End If
        txtWord.Focus() 'Focuses on the textbox regardless if the above is true or not
    End Sub
    Private Sub GetDefaultWords() 'DefaultWords Subroutine code
        Reader = New IO.StreamReader(appPath & "\DefaultWordList.txt") 'Reader collects
information from the "DefaultWordList" text document
        Dim RandomNumber As New Random 'Declares "RandomNumber" as a random integer value
        Dim WordCounter As Integer = 0 'Declares "WordCounter" as an Integer value
        Dim WordIndex As Integer 'Declares "WordIndex" as an integer value
        While (Reader.Peek() > -1) 'Executes a set of commands until the "Reader" stream
reader has reached the end of the text document
            DefaultArray(WordCounter) = Reader.ReadLine 'Adds words to the index value of
"DefaultArray" from the Reader's line in the text document
            WordCounter = WordCounter + 1 'Increments "WordCounter"
        End While
        For Words = 0 To 9 'Executes a set of tommands 10 times (0-9)
            WordIndex = RandomNumber.Next(1, WordCounter) 'Sets the "WordIndex" integer to
the next random value from "1" to "WordCounter"
            DefaultRandomArray(Words) = DefaultArray(WordIndex) 'Sets the "Words" index of
"DefaultRandomArray" to the next random number in "DefaultArray" in a set of 10
        Next
        Reader.Close() 'Closes the streamreader
    End Sub
    Private Sub GetDefaultWordList() 'GetDefaultWordList Subroutine code
        lstWordList.Items.Clear() 'Clears the word list box
        Reader = New IO.StreamReader(appPath & "\DefaultWordList.txt") 'Reader collects
information from the "DefaultWordList" text document
        While (Reader.Peek() > -1) 'Reads the file until the end of the text document has
been reached
```

```
            lstWordList.Items.Add(Reader.ReadLine) 'Adds words to the listbox from the text
document line-by-line
        End While
        Reader.Close() 'Closes the streamreader
        lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of the
amount of words that have been added to the list box
    End Sub
    Private Sub DisableButtons() 'DisableButtons Subroutine code
        txtWord.Enabled = False 'Disables text box
        btnInsert.Enabled = False 'Disables insert button
        btnReplace.Enabled = False 'Disables replace button
        btnRemove.Enabled = False 'Disables remove button
        btnCleanList.Enabled = False 'Disables clean list button
        btnInsert.BackColor = Color.Gray 'Sets the background color of insert button to gray
        btnReplace.BackColor = Color.Gray 'Sets the background color of replace button to
gray
        btnRemove.BackColor = Color.Gray 'Sets the background color of remove button to gray
        btnCleanList.BackColor = Color.Gray 'Sets the background color of clean list button
to gray
    End Sub
    Private Sub EnableButtons() 'EnableButtons Subroutine code
        txtWord.Enabled = True 'Enables text box
        btnInsert.Enabled = True 'Enables insert button
        btnReplace.Enabled = True 'Enables replace button
        btnCleanList.Enabled = True 'Enables clean list button
        If Not lstWordList.Items.Count <= 10 Then 'Checks how many words are displayed in the
word listbox.
            btnRemove.Enabled = True 'Enabled the remove button if the above statement is
true.
            btnRemove.BackColor = Color.Firebrick 'The remove button's color is set to
firebrick, showing that it is enabled
        End If
        btnInsert.BackColor = Color.LimeGreen 'Sets the insert button's background color to
LimeGreen
        btnReplace.BackColor = Color.Orange 'Sets the replace button's background color to
orange
        btnCleanList.BackColor = Color.SteelBlue 'Sets Clean List button's color is set to
SteelBlue.
    End Sub
    Private Sub GetCustomWordList() 'GetCustomWordList Subroutine code
        lstWordList.Items.Clear() 'Clears the list box
        Reader = New IO.StreamReader(appPath & "\CustomWordList.txt") 'Reader set to read off
of the "CustomWordList" text document
        While (Reader.Peek() > -1) 'Executes commands until the reader has reached the end of
the file
            lstWordList.Items.Add(Reader.ReadLine) 'Adds words to the listbox that the reader
has read line-by-line
        End While
        Reader.Close() 'Closes the streamreader
        lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of the
amount of words that have been added to the list box
    End Sub
    Private Sub OverwriteCustomWordList() 'OverwriteCustomWordList Subroutine code
        FileOpen(Filenum, appPath & "\CustomWordList.txt", OpenMode.Output) 'Opens the next
available file in the destination specified with "write" access
        FileClose()  'Closes the file immediately, erasing all of its contents
        For i = 0 To lstWordList.Items.Count - 1 'Executes set of commands until the integer
"i" is equal to the amount of words in the list box
            FileWriter = File.AppendText("CustomWordList.txt") 'Sets "FileWriter" variable to
appendtext of the "CustomWordList" text document
            FileWriter.WriteLine(lstWordList.Items(i)) 'Adds words line-by-line in the text
document through "DefaultArray" as well as the index of the word in the array
            FileWriter.Close() 'closes the File Writer
        Next
    End Sub
```

```
    Private Sub frmGameSettings_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown 'frmSettings Form KeyDown code
        If e.KeyCode = 46 Then 'Traps the Delete "KeyCode" instead of using Ascii value 127
            If btnRemove.Enabled = False Then 'Checks if the "btnRemove" button is enabled
                CancelExit = True 'The CancelExit flag is set to true so that the messagebox
doesn't close the form upon deactivating
                If rbDefaultList.Checked = True Then 'Checks if the "rbDefaultList" radio
button is selected
                    MsgBox("You cannot delete any words from the default word list.",
vbExclamation + vbOKOnly, "Invalid Function") 'The Messagebox notifies the user that they
cannot delete words from the default word list if they press delete
                Else
                    MsgBox("You cannot delete anymore words. You are only allowed a minimum
of 10 words.", vbExclamation + vbOKOnly, "Invalid Function") 'The Messagebox notifies the
user that they cannot delete anymore words if the list only contains 10 words
                End If
                CancelExit = False 'The CancelExit flag is set back to false
            Else
                Call btnRemove_Click(Nothing, Nothing) 'Clicks the remove button
            End If
        End If
    End Sub
    Private Sub frmGameSettings_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmGameSettings Form Load code
        If My.Settings.HangmanDefaultGameList = True Then 'Checks the if the boolean variable
in application settings "HangmanDefaultGameList" is set as true
            DisableButtons() 'Calls the "DisableButtons" subroutine
            rbDefaultList.Checked = True 'Selects the "Default List" radio button if the
above statement is true
            lblDefaultWordListInfo.BringToFront() 'Displays the "DefaultWordList" label
showing information on this radio button's selection
            GetDefaultWordList() 'Calls the "GetDefaultWordListtxt" subroutine
        Else 'Executes a set of commands if the "If statement" is false
            EnableButtons() 'Calls the "EnableButtons" subroutine
            rbCustomList.Checked = True 'Selects the "Custom List" radio button
            lblCustomWordListInfo.BringToFront() 'Displays the "CustomWordList" label showing
information on this radio button's selection
            GetCustomWordList() 'Calls the "GetCustomWordListtxt" subroutine
        End If
        ValidTimer.Start() 'Starts "ValidTimer"
        GetDefaultWords() 'Calls the "DefaultWords" subroutine
    End Sub
    Private Sub frmSettings_Deactivate(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Deactivate 'frmSettings Form Deactivate code
        If CancelExit = False Then 'Checks if the "CancelExit" boolean variable is set to
false
            Me.Dispose() 'Closes the form
        End If
    End Sub
    Private Sub rbDefaultList_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbDefaultList.MouseUp 'rbDefaultList Radio
Button MouseDown code
        lblWordsAdded.ForeColor = Color.Black 'Sets the "lblWordsAdded" label's forecolor to
black
        ValidTimer.Stop() 'Stops the valid timer when the "DefaultList" radio button has been
checked
        lblDefaultWordListInfo.BringToFront()  'Displays the "lblDefaultWordListInfo" label
showing information on this radio button's selection
        GetDefaultWordList() 'Calls the "GetDefaultWordListtxt" subroutine
        DisableButtons() 'Calls the "DisableButtons" subroutine
        lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of the
amount of words that have been added to the list box
        SaveSettings() 'Calls the "SaveSettings" subroutine
    End Sub
```

```
    Private Sub rbCustomList_Mouseup(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbCustomList.MouseUp 'rbCustomList Radio Button
MouseDown code
        ValidTimer.Start() 'Starts the valid timer when the "CustomList" radio button has
been checked
        lblCustomWordListInfo.BringToFront() 'Displays the "lblCustomWordListInfo" label
showing information on this radio button's selection
        GetCustomWordList() 'Calls the "GetCustomWordListtxt" subroutine
        EnableButtons() 'Calls the "EnableButtons" subroutine
        SaveSettings() 'Calls the "SaveSettings" subroutine
        lblWordsAdded.Text = lstWordList.Items.Count 'Sets the label with the index of the
amount of words that have been added to the list box
    End Sub

    Private Sub SaveSettings() 'SaveSettings Subroutine code
        If rbDefaultList.Checked = True Then 'Checks if the "rbDefaultList" radio button is
checked
            My.Settings.HangmanDefaultGameList = True 'Sets the application settings variable
"HangmanDefaultGameList" to True
        ElseIf rbCustomList.Checked = True Then 'Checks if the "rbCustomList" radio button is
checked
            My.Settings.HangmanDefaultGameList = False 'Sets the application settings
variable "HangmanDefaultGameList" to False
        End If
        My.Settings.Save() 'Saves settings to application
    End Sub
    Private Sub SaveAnimation() 'SaveAnimation Subroutine code
        picSaveGif.Visible = True 'Shows the save picture box gif animation
        PicGreenTick.Visible = False 'Hides the green tick picture box from the form
        lblGameSettingsSaved.Text = "Saving Settings..." 'Sets the "lblGameSettingsSaved"
label's text to "Saving Settings..."
        With picSaveGif 'Executes a set of commands regarding the "picSaveGif" picture box
            .Image = My.Resources.LoadingScreen 'Sets the "picSaveGif" picture box's image to
the gif file in resources
            .SizeMode = PictureBoxSizeMode.CenterImage 'Changes the "SizeMode" of the image
module in the picture box
        End With
        SaveTimer.Start() 'Starts "SaveTimer" Timer
    End Sub
    Private Sub SaveTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles SaveTimer.Tick 'SaveTimer Tick Subroutine code
        PicGreenTick.Visible = True 'Shows the green tick "picGreenTick" picture box
        picSaveGif.Visible = False 'Hides the "picSaveGif" picture box gif animation
        lblGameSettingsSaved.Text = "Game Settings Saved." 'Sets the "lblGameSettingsSaved"
label's text to "Game Settings Saved."
        SaveTimer.Stop()  'Stops "SaveTimer" Timer
    End Sub
End Class
```

8.11. Settings:

```
Imports System.IO 'Form makes reference to the "System.IO" namespacePublic Class frmSettings
'frmSettings Form Class
Public Class frmSettings 'frmGameSettings Form code
    Dim MaxCounter As Integer
    Dim Files(10000) As String
    Dim Toggle As Boolean = False
    Dim ToggleCounter As Integer = 6
    Dim CancelExit As Boolean = False
    Dim AppPath As String = Application.StartupPath
    Dim fileNames = My.Computer.FileSystem.GetFiles(AppPath & "\Music",
FileIO.SearchOption.SearchAllSubDirectories)
    Const WM_NCLBUTTONDBLCLK As Integer = &HA3 'Declares constant variable
"WM_NCLBUTTONDBLCLK" as an integer and assigns it's message "&HA3"
```

```
    Const WM_NCHITTEST As Integer = &H84 'Declares constant variable "WM_NCHITTEST" as an
integer and assigns it's message "&H84"
    Const HTCLIENT As Integer = &H1 'Declares constant variable "HTCLIENT" as an integer and
assigns it's message "&H1"
    Const HTCAPTION As Integer = &H2 'Declares constant variable "HTCAPTION" as an integer
and assigns it's message "&H2"
    Protected Overrides Sub WndProc(ByRef Message As System.Windows.Forms.Message) 'The
"WndProc" Function (Processing Windows Messages) Protected Override subroutine
        If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for the
message (Message.Msg) is "WM_NCLBUTTONDBLCLK" which is posted when the user double-clicks the
left mouse button while the cursor is within the nonclient area of the window
        MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the subroutine for
location
        Select Case Message.Msg 'Selects the "Message.Msg" message for the Case Else
statement
            Case WM_NCHITTEST 'Checks if the message is sent to a window to determine which
aspect of the window corresponds to a specific screen coordinate
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
                If Message.Result = HTCLIENT Then Message.Result = HTCAPTION 'Checks if the
result of the "Message" function returns with "HTCLIENT" which is posted when the user's
curser enters the client area, then changes the result to "HTCAPTION" which posts the message
position to the title bar
                If Message.Msg = WM_NCLBUTTONDBLCLK Then Return 'Checks if the ID number for
the message (Message.Msg) is "WM_NCLBUTTONDBLCLK" then returns the message to the subroutine
            Case Else
                MyBase.WndProc(Message) 'Returns the "WndProc(Message)" message to the
subroutine for location
        End Select
    End Sub
    Private Sub frmsettings_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load 'frmSettings Form Load code
        LoadSettings() 'Calls the "LoadSettings" subroutine code
        MaxCounter = fileNames.count 'Sets "MaxCounter" integer variable to the maximum
number in the "fileNames" string
        For x = 0 To MaxCounter - 1 'Executes a set of commands until the integer value "x"
has reached the value of "MaxCounter"
            Files(x) = fileNames(x) 'Sets the "Files" array index value to "fileNames" index
value
            Files(x) = Files(x).Substring(Files(x).LastIndexOf("\") + 1) 'Erases the path
directory in the string line index value "x" of "Files" (e.g, "c:\Users\" etc...)
        Next
        For i = 0 To MaxCounter - 1 'Executes a set of commands until the integer value "i"
has reached the value of "MaxCounter"
            lstMusic.Items.Add(Files(i)) 'Adds the "Files" array index value "i" to the
"lstMusic" list box
        Next
    End Sub
    Private Sub frmSettings_Deactivate(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Deactivate 'frmSettings Form Deactivate code
        If CancelExit = False Then 'Checks if the boolean variable "CancelExit" is set to
False
            Me.Hide() 'Hides the form if the above statement is true
        End If
    End Sub
    Private Sub StopAllMouseMoveTimers() 'StopAllMouseMoveTimers subroutine code
        'Stops all MouseMoveTimers on all forms
        frmMainMenu.MouseMoveTimer.Stop()
        frmHangmanModeMenu.MouseMoveTimer.Stop()
        frmHangmanSinglePlayer.MouseMoveTimer.Stop()
        frmHangmanTwoPlayer.MouseMoveTimer.Stop()
        frmHangmanVersus.MouseMoveTimer.Stop()
        frmTicTacToeModeMenu.MouseMoveTimer.Stop()
        frmTicTacToeSinglePlayer.MouseMoveTimer.Stop()
        frmTicTacToeTwoPlayer.MouseMoveTimer.Stop()
    End Sub
```

```
    Private Sub StartAllMouseMoveTimers() 'StartAllMouseMoveTimers subroutine code
        'Starts all MouseMoveTimers on all forms
        frmMainMenu.MouseMoveTimer.Start()
        frmHangmanModeMenu.MouseMoveTimer.Start()
        frmHangmanSinglePlayer.MouseMoveTimer.Start()
        frmHangmanTwoPlayer.MouseMoveTimer.Start()
        frmHangmanVersus.MouseMoveTimer.Start()
        frmTicTacToeModeMenu.MouseMoveTimer.Start()
        frmTicTacToeSinglePlayer.MouseMoveTimer.Start()
        frmTicTacToeTwoPlayer.MouseMoveTimer.Start()
    End Sub
    'Listbox
    Private Sub lstMusic_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstMusic.SelectedIndexChanged 'lstMusic List box
SelectedIndexChanged code
        btnPlay.BackColor = Color.Lime
        btnPlay.Enabled = True 'Enables the "btnPlay" button
        btnRemove.BackColor = Color.Firebrick 'Sets the "btnRemove" background color to
Firebrick
        btnRemove.Enabled = True 'Enables the "btnRemove" button
    End Sub

    'Radio buttons
    Private Sub rbDefaultSettings_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbDefaultSettings.MouseUp 'rbDefaultSettings
Radio Button MouseUp code
        My.Settings.DefaultSettings = True 'Sets the "DefaultSettings" boolean variable in
the application settings to True
        My.Settings.Save() 'Saves settings to application
        lblDefaultSettingsInfo.BringToFront() 'Displays the "DefaultSettingsInfo" label
showing information on this radio button's selection
        rbDefaultSettings.Checked = True
        rbMainMenu.Checked = True
        rbHangMan.Checked = False
        rbTicTacToe.Checked = False
        rbMusicAndSounds.Checked = True
        rbSoundsOnly.Checked = False
        rbMusicOnly.Checked = False
        cbDisableCaptions.Checked = False
        DisableAllButtons() 'Calls the "DisableRadioButtons" subroutine
    End Sub
    Private Sub rbCustomSettings_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbCustomSettings.MouseUp 'rCustomSettings Radio
Button MouseUp code
        My.Settings.DefaultSettings = False 'Sets the "DefaultSettings" boolean variable in
the application settings to False
        My.Settings.Save() 'Saves settings to application
        lblCustomSettingsInfo.BringToFront() 'Displays the "CustomSettingsInfo" label showing
information on this radio button's selection
        EnableAllButtons() 'Calls the "EnableRadioButtons" subroutine
        LoadSettings() 'Calls the "LoadSettings" subroutine
    End Sub
    Private Sub rbMusicAndSounds_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbMusicAndSounds.MouseUp 'rbMusicAndSounds Radio
Button MouseUp code
        bxSelectMusic.Enabled = True 'Enables the "bxSelectMusic" group box
        My.Settings.Mute = False 'Sets the boolean variable "Mute" in the application
settings to True
        frmHangmanModeMenu.btnMuteUnmute.Visible = True 'Shows the "btnMuteUnmute" button in
the "frmHangManModeMenu" form
        frmMainMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button in the
"frmMainMenu" form
        frmTicTacToeModeMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button
in the "frmTicTacToeModeMenu" form
        frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button in the "frmTicTacToeSinglePlayer" form
```

```
        frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button
in the "frmTicTacToeTwoPlayer" form
        OCXMusicPlayer.Ctlcontrols.play() 'Plays the song that's inside the "OCXMusicPlayer"
OCX's URL
        OCXMusicPlayer.Ctlenabled = True  'Enables the "OCXMusicPlayer" Windows Media Player
OCX (ActiveX Control)
        SaveAnimation() 'Calls the "SaveAnimation" subroutine
    End Sub
    Private Sub rbSoundsOnly_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbSoundsOnly.MouseUp 'rbSoundsOnly Radio Button
MouseUp code
        bxSelectMusic.Enabled = False 'Disables the "bxSelectMusic" group box
        My.Settings.Mute = False 'Sets the boolean variable "Mute" in the application
settings to True
        frmMainMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button in the
"frmMainMenu" form
        frmHangmanModeMenu.btnMuteUnmute.Visible = True 'Shows the "btnMuteUnmute" button in
the "frmHangManModeMenu" form
        frmTicTacToeModeMenu.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button
in the "frmTicTacToeModeMenu" form
        frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute"
button in the "frmTicTacToeSinglePlayer" form
        frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = True 'Shows the "btnMuteUnmute" button
in the "frmTicTacToeTwoPlayer" form
        OCXMusicPlayer.Ctlcontrols.pause() 'Pauses the song that's inside the
"OCXMusicPlayer" OCX's URL
        OCXMusicPlayer.Ctlenabled = False 'Disables the "OCXMusicPlayer" Windows Media Player
OCX (ActiveX Control)
        SaveAnimation() 'Calls the "SaveAnimation" subroutine
    End Sub
    Private Sub rbMusicOnly_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbMusicOnly.MouseUp 'rbMusicOnly Radio Button
MouseUp code
        bxSelectMusic.Enabled = True 'Enables the "bxSelectMusic" group box
        frmMainMenu.btnMuteUnMute.Visible = False 'Hides the "btnMuteUnmute" button in the
"frmMainMenu" form
        frmHangmanModeMenu.btnMuteUnmute.Visible = False 'Hides the "btnMuteUnmute" button in
the "frmHangManModeMenu" form
        frmTicTacToeModeMenu.btnMuteUnMute.Visible = False 'Hides the "btnMuteUnmute" button
in the "frmTicTacToeModeMenu" form
        frmTicTacToeSinglePlayer.btnMuteUnMute.Visible = False 'Hides the "btnMuteUnmute"
button in the "frmTicTacToeSinglePlayer" form
        frmTicTacToeTwoPlayer.btnMuteUnMute.Visible = False 'Hides the "btnMuteUnmute" button
in the "frmTicTacToeTwoPlayer" form
        My.Settings.Mute = True 'Sets the boolean variable "Mute" in the application settings
to True
        OCXMusicPlayer.Ctlcontrols.play() 'Plays the song that's inside the "OCXMusicPlayer"
OCX's URL
        OCXMusicPlayer.Ctlenabled = True 'Enables the "OCXMusicPlayer" Windows Media Player
OCX (ActiveX Control)
        SaveAnimation() 'Calls the "SaveAnimation" subroutine
    End Sub
    Private Sub cbDisableCaptions_MouseClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles cbDisableCaptions.MouseClick 'cbDisableCaptions
MouseClick code
        If cbDisableCaptions.Checked = True Then
            HideAllLabels()
            StopAllMouseMoveTimers()
        Else
            StartAllMouseMoveTimers()
            frmMainMenu.MouseMoveTimer.Start() 'Starts the "MouseMoveTimer" only on the main
menu because the main menu is the only form that doesnt refresh the "Load" event
        End If
        SaveAnimation() 'Calls the "SaveAnimation" subroutine
    End Sub
```

```
    Private Sub linklblWhatsThis_MouseClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles linklblWhatsThis.MouseClick 'linklblWhatsThis
MouseClick code
        CancelExit = True 'Sets the "CancelExit" boolean variable to True
        frmMainMenu.player.Stream = My.Resources.Windows_Information
        frmMainMenu.player.Play()
        frmDisableCaptionsDialog.ShowDialog()
        CancelExit = False 'Sets the "CancelExit" boolean variable to False
    End Sub
    Private Sub rbMainMenu_MouseUp(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles rbTicTacToe.MouseUp, rbMainMenu.MouseUp,
rbHangMan.MouseUp 'Mouse Up (all) RadioButton events code
        SaveAnimation() 'Calls the "SaveAnimation" subroutine
    End Sub

    'Buttons
    Private Sub btnPlay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnPlay.Click 'btnPlay Button Click code
        Dim FullDirectory As String = AppPath & "\Music\" 'Declares "FullDirectory" as a
private string containing a location to the Music Folder
        OCXMusicPlayer.URL = FullDirectory & lstMusic.SelectedItem 'Places the
"FullDirectory" string in the Media Player OCX's URL with the selected list item
        OCXMusicPlayer.Ctlcontrols.play() 'Plays the URL item
    End Sub
    Private Sub btnRemove_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnRemove.Click 'btnRemove Button Click code
        CancelExit = True 'Sets the "CancelExit" boolean variable to True
        Dim MessageBoxResult As String 'Declares MessageBoxResult as a string
        MessageBoxResult = MsgBox("Are you sure you want to delete this media file from the
list?", vbInformation + vbYesNo, "Remove Song") 'Sets the "MessageBoxResult" string as the
user's decision to the msgbox
        If MessageBoxResult = vbYes Then 'Checks if the user's decision in the msgbox was
"Yes"
            Dim FullDirectory As String = AppPath & "\Music\" & lstMusic.SelectedItem
'Declares "FullDirectory" as a string with the "Music" file's directory as well as the user's
selected List box item in "lstMusic"
            If lstMusic.SelectedItem = "Terrathede Games Theme Song - Adrenaline.mp3" Then
'Checks if the user tries to remove the selected item "Addrenaline.mp3" from the "lstMusic"
list box
                MsgBox("You cannot delete this media file because it's properties are set to
default") 'Tells the user that they cannot remove this specific file because it's a default
list item
                CancelExit = False 'Sets the "CancelExit" boolean variable to False
                Exit Sub 'Exits the subroutine
            End If
            If File.Exists(FullDirectory) Then 'Checks if the path entered in the
"FullDirectory" exists
                File.Delete(FullDirectory) 'Deletes the selected item using the full
directory (FullDirectory string path)
                lstMusic.Items.Remove(lstMusic.SelectedItem) 'Removes the same item from the
"lstMusic" list box
            End If
        End If
        CancelExit = False 'Sets the "CancelExit" boolean variable to False
    End Sub
    Private Sub btnImport_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnImport.Click 'btnImport Button Click code
        CancelExit = True 'Sets the "CancelExit" boolean variable to True
        Dim Open_File As New OpenFileDialog 'Declares "Open_File" as an OpenFileDialog box
        With Open_File 'Executes a set of commands regarding "Open_File"
            .Filter = "Media Files (*.mp3)|*.mp3" 'Sets the "Filter" to the appropriate file
extension
            .FileName = "" 'Clears the "FileName" text box in the OpenFileDialog box
            .CheckFileExists = True 'Gets a value indicating whether the OpenFileDialog box
displays a warning that the specified file does not exist
        End With
```

```
        If Open_File.ShowDialog() = Windows.Forms.DialogResult.OK Then 'Checks if the user
has finished selecting the file
            Dim Path As String 'Declares "Path" as a string variable
            Path = Open_File.FileName 'Writes the "FileName" from the "Open_File"
OpenFileDialog to the "Path" string`
            Dim FullDirectory As String = AppPath & "\Music\" & Open_File.SafeFileName
'Declares "FullDirectory" as a string with the "Music" file's directory as well as the user's
selected List box item in "lstMusic"
            If File.Exists(Path) And Not File.Exists(FullDirectory) Then 'Checks if the
"Open_File" OpenFileDialog's "FileName" path exists and the "FullDirectory" (with the
selected list item) does not exist in the same file path.
                File.Copy(Path, FullDirectory) 'Copy's the "FullDirectory" file to the file
"Path"
                lstMusic.Items.Add(Open_File.SafeFileName) 'Adds the new media file (.mp3) to
the "lstMusic" list box
            Else
                MsgBox("You have already added this media file to the list.") 'Tells the user
that the media file already exists if the user has tried to enter the same media file to the
list
            End If
        End If
        CancelExit = False 'Sets the "CancelExit" boolean variable to False
    End Sub
    Private Sub btnClose_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseDown 'btnClose Button MouseDown
code
        btnClose.BackgroundImage = My.Resources.Close_Button_Pushed 'Changes the background
image of the close button when the mouse is down
    End Sub
    Private Sub btnClose_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseEnter 'btnClose Button MouseEnter code
        btnClose.BackgroundImage = My.Resources.Close_Button_Highlighted 'Changes the
background image of the close button to highlighted when the curser enters the button
        If My.Settings.Mute = False Then 'Checks if the user wants the sounds muted on all
forms (using the mute button)
            frmMainMenu.player.Stream = My.Resources.sound_MouseScrollover 'Sets the
soundplayer to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
    End Sub
    Private Sub btnClose_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnClose.MouseLeave 'btnClose Button MouseLeave code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the close button's
background image to the original image when the curser has left the picture box
    End Sub
    Private Sub btnClose_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles btnClose.MouseUp 'btnClose Button MouseUp code
        btnClose.BackgroundImage = My.Resources.Close_Button 'Changes the close button's
background image to the original image when the mouse is set to up
        If My.Settings.Mute = False Then  'Checks if the user wants the sounds muted on all
forms (using the mute button)
            frmMainMenu.player.Stream = My.Resources.sound_FormClosing 'Sets the soundplayer
to the "Scrollover" WAV file in the resources
            frmMainMenu.player.Play() 'Plays the sound file
        End If
        LoadSettings() 'Calls the "LoadSettings" subroutine
        If cbDisableCaptions.Checked = True Then
            StopAllMouseMoveTimers()
        Else
            frmMainMenu.MouseMoveTimer.Start() 'Starts the "MouseMoveTimer" only on the main
menu because the main menu is the only form that doesnt refresh the "Load" event
        End If
        Me.Hide() 'Hides the form
    End Sub

    'Private Subs
```

```
    Private Sub LoadSettings() 'LoadSettings Subroutine code
        If My.Settings.DefaultSettings = True Then  'Checks if the boolean variable
"DefaultSettings" in application settings is set to True
            rbDefaultSettings.Checked = True 'Checks the "rbDefaultSettings" RadioButton
            rbMainMenu.Checked = True 'Checks the "rbMainMenu" RadioButton
            rbHangMan.Checked = False 'De-selects the "rbHangMan" RadioButton
            rbTicTacToe.Checked = False 'De-selects the "rbTicTacToe" RadioButton
            rbMusicAndSounds.Checked = True 'Checks the "rbMusicAndSounds" RadioButton
            rbSoundsOnly.Checked = False 'De-selects the "rbSoundsOnly" RadioButton
            rbMusicOnly.Checked = False 'De-selects the "rbMusicOnly" RadioButton
            DisableAllButtons()
            lblDefaultSettingsInfo.BringToFront() 'Displays the "DefaultWordList" label
showing information on this radio button's selection
            Exit Sub 'Exits the subroutine if the above If statement returns true
        Else
            rbCustomSettings.Checked = True 'Checks the "rbCustomSettings" RadioButton if the
above If statement returns false
            lblCustomSettingsInfo.BringToFront() 'Displays the "CustomWordList" label showing
information on this radio button's selection
        End If
        If My.Settings.MainMenu = True Then 'Checks if the boolean variable "MainMenu" in
application settings is set to True
            rbMainMenu.Checked = True 'Checks the "rbMainMenu" RadioButton
        Else
            rbMainMenu.Checked = False 'De-selects the "rbMainMenu" RadioButton
        End If
        If My.Settings.HangMan = True Then 'Checks if the boolean variable "HangMan" in
application settings is set to True
            rbHangMan.Checked = True 'Checks the "rbHangMan" RadioButton
        Else
            rbHangMan.Checked = False 'De-selects the "rbHangMan" RadioButton
        End If
        If My.Settings.TicTacToe = True Then 'Checks if the boolean variable "TicTacToe" in
application settings is set to True
            rbTicTacToe.Checked = True 'Checks the "rbTicTacToe" RadioButton
        Else
            rbTicTacToe.Checked = False 'De-selects the "rbTicTacToe" RadioButton
        End If
        If My.Settings.MusicAndSounds = True Then 'Checks if the boolean variable
"MusicAndSounds" in application settings is set to True
            rbMusicAndSounds.Checked = True 'Checks the "rbMusicAndSounds" RadioButton
        Else
            rbMusicAndSounds.Checked = False 'De-selects the "rbMusicAndSounds" RadioButton
        End If
        If My.Settings.SoundsOnly = True Then 'Checks if the boolean variable "SoundsOnly" in
application settings is set to True
            rbSoundsOnly.Checked = True 'Checks the "rbSoundOnly" RadioButton
            bxSelectMusic.Enabled = False 'Disables the "bxSelectMusic" group box
            OCXMusicPlayer.URL = Nothing 'Clears the "OCXMusicPlayer" Windows Media Player
OCX (ActiveX control)
            OCXMusicPlayer.Ctlenabled = False 'Disables the "OCXMusicPlayer" Windows Media
Player OCX (ActiveX control)
        Else
            rbSoundsOnly.Checked = False 'De-selects the "rbMainMenu" RadioButton
        End If
        If My.Settings.MusicOnly = True Then 'Checks if the boolean variable "MusicOnly" in
application settings is set to True
            rbMusicOnly.Checked = True 'Checks the "rbMusicOnly" RadioButton
        Else
            rbMusicOnly.Checked = False 'De-selects the "rbMusicOnly" RadioButton
        End If
        If My.Settings.DisableCaptions = True Then 'Checks if the boolean variable
"DisableCaptions" in application settings is set to True
            cbDisableCaptions.Checked = True 'Checks the "cbDisableCaptions" Checkbox
        Else
            cbDisableCaptions.Checked = False 'Checks the "cbDisableCaptions" Checkbox
```

```
        End If
    End Sub
    Private Sub SaveSettings() 'SaveSettings Subroutine code
        If rbMainMenu.Checked = True Then 'Checks if the "rbMainMenu" radio button is checked
            My.Settings.MainMenu = True 'Sets "MainMenu" boolean variable in application
settings to True
        Else
            My.Settings.MainMenu = False 'Sets "MainMenu" boolean variable in application
settings to False
        End If
        If rbHangMan.Checked = True Then 'Checks if the "rbHangMan" radio button is checked
            My.Settings.HangMan = True 'Sets "HangMan" boolean variable in application
settings to True
        Else
            My.Settings.HangMan = False 'Sets "HangMan" boolean variable in application
settings to False
        End If
        If rbTicTacToe.Checked = True Then 'Checks if the "rbTicTacToe" radio button is
checked
            My.Settings.TicTacToe = True 'Sets "TicTacToe" boolean variable in application
settings to True
        Else
            My.Settings.TicTacToe = False 'Sets "TicTacToe" boolean variable in application
settings to False
        End If
        If rbMusicAndSounds.Checked = True Then 'Checks if the "rbMusicAndSounds" radio
button is checked
            My.Settings.MusicAndSounds = True 'Sets "MusicAndSounds" boolean variable in
application settings to True
        Else
            My.Settings.MusicAndSounds = False 'Sets "MusicAndSounds" boolean variable in
application settings to False
        End If
        If rbSoundsOnly.Checked = True Then 'Checks if the "rbSoundsOnly" radio button is
checked
            My.Settings.SoundsOnly = True 'Sets "SoundsOnly" boolean variable in application
settings to True
        Else
            My.Settings.SoundsOnly = False 'Sets "SoundsOnly" boolean variable in application
settings to False
        End If
        If rbMusicOnly.Checked = True Then 'Checks if the "rbMusicOnly" radio button is
checked
            My.Settings.MusicOnly = True 'Sets "MusicOnly" boolean variable in application
settings to True
        Else
            My.Settings.MusicOnly = False 'Sets "MusicOnly" boolean variable in application
settings to False
        End If
        If cbDisableCaptions.Checked = True Then 'Checks if the "cbDisableCaptions" radio
button is checked
            My.Settings.DisableCaptions = True 'Sets "DisableCaptions" boolean variable in
application settings to True
        Else
            My.Settings.DisableCaptions = False 'Sets "DisableCaptions" boolean variable in
application settings to False
        End If
        My.Settings.Save() 'Saves the application's settings
    End Sub
    Private Sub SaveAnimation() 'SaveAnimation Subroutine code
        SaveSettings() 'Calls
        picSaveGif.Visible = True 'Shows the save picture box gif animation
        picGreenTick.Visible = False 'Hides the green tick picture box from the form
        lblSettingsSaved.Text = "Saving Settings..." 'Sets the "lblSettingsSaved" label's
text to "Saving Settings..."
        With picSaveGif 'Executes a set of commands regarding the "picSaveGif" picture box
```

```
            .Image = My.Resources.LoadingScreen 'Sets the "picSaveGif" picture box's image to
the gif file in resources
            .SizeMode = PictureBoxSizeMode.CenterImage 'Changes the "SizeMode" of the image
module in the picture box
        End With
        SaveTimer.Start() 'Starts "SaveTimer" Timer
    End Sub
    Private Sub DisableAllButtons() 'DisableSettings Subroutine code
        rbMainMenu.Enabled = False 'Disables the "rbMainMenu" RadioButton
        rbHangMan.Enabled = False 'Disables the "rbHangMan" RadioButton
        rbTicTacToe.Enabled = False 'Disables the "rbTicTacToe" RadioButton
        rbMusicAndSounds.Enabled = False 'Disables the "rbMusicAndSounds" RadioButton
        rbSoundsOnly.Enabled = False 'Disables the "rbSoundsOnly" RadioButton
        rbMusicOnly.Enabled = False 'Disables the "rbMusicOnly" RadioButton
        cbDisableCaptions.Enabled = False 'Disables the "cbDisableCaptions" CheckBox
    End Sub
    Private Sub EnableAllButtons() 'EnableSettings Subroutine code
        rbMainMenu.Enabled = True 'Enables the "rbMainMenu" RadioButton
        rbHangMan.Enabled = True 'Enables the "rbHangMan" RadioButton
        rbTicTacToe.Enabled = True 'Enables the "rbTicTacToe" RadioButton
        rbMusicAndSounds.Enabled = True 'Enables the "rbMusicAndSounds" RadioButton
        rbSoundsOnly.Enabled = True 'Enables the "rbSoundsOnly" RadioButton
        rbMusicOnly.Enabled = True 'Enables the "rbMusicOnly" RadioButton
        cbDisableCaptions.Enabled = True 'Enables the "cbDisableCaptions" CheckBox
    End Sub
    Private Sub SaveTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles SaveTimer.Tick 'SaveTimer Timer Tick code
        picGreenTick.Visible = True 'Shows the green tick "picGreenTick" picture box
        picSaveGif.Visible = False 'Shows the "picSaveGif" picture box gif animation
        lblSettingsSaved.Text = "Settings Saved." 'Sets the "lblSettingsSaved" label's text
to "Game Settings Saved."
        SaveTimer.Stop()  'Stops "SaveTimer" Timer
    End Sub
    Private Sub lstMusic_MouseDoubleClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles lstMusic.MouseDoubleClick  'lstMusic list box
MouseDoubleClick code
        Call btnPlay_Click(Nothing, Nothing) 'Calls the "btnPlay" click event
    End Sub

    Private Sub HideAllLabels() 'SaveSettiHideAllLabelsngs Subroutine code
        'Hides all labels on all forms that are next to a button
        frmMainMenu.lblInfo.Visible = False
        frmMainMenu.lblMute.Visible = False
        frmMainMenu.lblSettings.Visible = False
        frmHangmanModeMenu.lblMainMenu.Visible = False
        frmHangmanModeMenu.lblInfo.Visible = False
        frmHangmanModeMenu.lblMute.Visible = False
        frmHangmanModeMenu.lblGameSettings.Visible = False
        frmTicTacToeModeMenu.lblMainMenu.Visible = False
        frmTicTacToeModeMenu.lblInfo.Visible = False
        frmTicTacToeModeMenu.lblMute.Visible = False
        frmTicTacToeModeMenu.lblSettings.Visible = False
        frmTicTacToeSinglePlayer.lblLeaveGame.Visible = False
        frmTicTacToeSinglePlayer.lblInfo.Visible = False
        frmTicTacToeSinglePlayer.lblMute.Visible = False
        frmTicTacToeSinglePlayer.lblSettings.Visible = False
        frmTicTacToeTwoPlayer.lblLeaveGame.Visible = False
        frmTicTacToeTwoPlayer.lblInfo.Visible = False
        frmTicTacToeTwoPlayer.lblMute.Visible = False
        frmTicTacToeTwoPlayer.lblSettings.Visible = False
    End Sub
End Class
```

8.12. DisableCaptionDialog:

```
Public Class frmDisableCaptionsDialog 'frmDisableCaptionsDialog code
    Dim RandomFlip As Boolean = False
    Private Sub frmDisableCaptionsDialog_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'frmDisableCaptionsDialog form Load code
        AcceptButton = btnOK 'Sets the ok button as the focused button
        FlipImageTimer.Start() 'Starts the flipimagetimer
    End Sub

    Private Sub FlipImageTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FlipImageTimer.Tick 'FlipImageTimer Tick code
        If RandomFlip = True Then 'Checks if the boolean variable is set to true
            picCaptions.BackgroundImage = My.Resources.TerrathedeGamesMenu_Captions
            RandomFlip = False 'Sets RandomFlip to False
        Else
            picCaptions.BackgroundImage = My.Resources.TerrathedeGamesMenu_NoCaptions
            RandomFlip = True 'Sets RandomFlip to True
        End If
    End Sub

    Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnOK.Click 'btnOK Click code
        Me.Dispose() 'Closes the current form
        frmSettings.Show() 'Shows the "frmSettings" form
    End Sub
End Class
```

9. Evaluation Statement:

9.1. Hangman

Checklist / Objectives	Success?
Single player format where the progam generates a random word in Single Player hangman.	Successful
The player can choose a default list (of 1600+ words) or input their own list (in format according to the player guide.)	Successful
Option to delete individual words from the custom list	Successful
Two player format where each player can input words for the other player to guess.	Successful
A score system (gold tokens) of who wins each round and the overall winner at the end of each rounds	Successful
A system of lives where as a life is lost and image is presented which is stepped in regards to how many lives the player has left.	Successful
All information is shown when the game is finished (or players turn in 2 player) such as round information, number of attempts, winner of the round, score.	Successful
Include an information slide show be accessed while playing mid game.	Successful

Problems:

Throughout the making of the software, some problems arose such as:

- The variables were conmflicting with each other (especially temporary Boolean variables) that were hard initially to distinct them from each other.
- The initial screen design had to be re-designed as the buttons were all over the place and when tested by end-users (beta testing) they found it difficult to follow what to do.
- There was a problem with the radio button on the editing word list form; the confirmation (message box) the popped up conflicted with the changing of radio buttons and had to be taken out. In response to this a warning label was placed next to the radio buttons.

9.2. Tic Tac Toe

Checklist / Objectives	Success?
A grid referenced board (using numbers to indicate the grid reference).	Successful
Single Player tic tac toe computer player generates random markers to place on the grid	Successful
Tokens are incremented properly	Successful
Two Player works well in regards to markers and the differenciation between players	Successful
Two Player token increment is mended	Successful

Problems:

- Throughout the making of the software, some problems arose such as:
- The computer player's difficulty is set to recruit.
- Initially, the two player form's layout was too hard to figure out and withdrawn from consistency.

9.3. Future Enhancements

9.3.1. Hangman

Some future enhancement that could be implemented on Hangman could be:

- A hint system; where the computer gives a hint if the player is struggling to guess a word
- Introduct a better point system. E.g, lives in single player
- Single Player shouldn't be endless.

9.3.2. Tic Tac Toe

Some future enhancement that could be implemented on Tic Tac Toe could be:

- Computer Player in Two Player Tic Tac Toe is more difficult.
- Switching markers between rounds.
- Strategic gameplay (like hangman)
- Multiplayer system (i.e. using an online database)

9.4. Concluding evaluation

During the creation of the software solution was successful. However, in saying that there were a few faults that were ran into by myself and the company Terrathede Games. I can't clarify whether or not I had followed everything to plan and completed it with full integrity. However I tried my best to correct any mistakes that arrised to the occasion as well as the appropriate feedback by my beta testers that led to a gread success in finding and mending possible errors, screen element placement and the durability of the program itself. The Gantt Chart was not followed accurately the whole way through the creation process, I failed to allocate more time for tasks when I had test and assignments to study for/due as well as my art major and I had to make up for it later on. I had under estimated how long it would take to design, Photoshop CS6 and code the software application, this was okay as I left catch up time in the Gantt Chart which helped a lot. As far as features go, I was initially going to add the "Multi-player" tic tac toe using an online database with "MySQL", though, like I mentioned earlier, I under estimated my time for the duration of the creation process. The program is successful and it possibly had a few bugs/errors, but I have used a patcher with my program and updated it regularly during the beta testing process using facebook which amazingly hit over 1,000 likes.