Since this is a solo project, all the work was done by Dominic Nguyen.

**Classes That Were Used**

There are two classes in my Hangman program. The Game class and the Difficulty class.

The Game class holds most of the necessary information needed for the game. Firstly, there is a Difficulty object of the Difficulty class called difficulty. The game’s difficulty is determined by the values set on difficulty, which is determined by the player’s choice at the beginning of the game. Secondly, there is the string wordToGuess. This string holds the word that the player will be guessing. This word is determined by a random selection from a text file of words and matches the difficulty. Thirdly, there is the string progressWord, which is the string displayed to the player whenever they are guessing the word. For example, if the player is given the word “ape” and has only guessed the letter “a,” then progressWord would be “a--”, with dashes where the unknown letters are. Fourthly, there is a dynamic string array called availableSelections, which holds all the valid inputs during the guessing part of the game. These inputs can include letters a through z (lowercase only), the question mark (?), and the asterisk (\*). When a letter is entered, that letter is removed from the array. When the question mark is entered, another prompt to guess the entire word is provided. When the asterisk is entered, the state of the game is saved. Finally, there is an integer sizeOfAvailableSelections, which hold the array size of the mentioned dynamic string array, which is initially 28 (26 letters plus the question mark and asterisk). These variables are all set to private, so there are corresponding public setter and getter methods associated with these variables in the class. Also, note that through a SaveGame and LoadGame function inside main.cpp, the values of these variables can be saved to a text file that allows the game to be loaded in the menu. In addition, there are functions. Firstly, there is the UpdateProgressWord. It takes a letter that is a string and converts its corresponding dash(es) to that letter. For example, if the player is given the word “apple,” is guessing the letter “p,” and has progressWord that is equal to “a----,” then this function will take the letter “p” and set progressWord to “app--”. Secondly, there is a Print function that takes a string of the filename with the hangman Ascii art. This function prints the correct state of the hangman drawing by using the attemptsLeft variable. The default Game constructor sets the values to its default values. The deconstructor deallocates the availableSelections array.

The Difficulty class holds information specific to the difficulty. This includes three integer variables called minAmountOfLetters, maxAmountOfLetters, and numberOfAttempts, which determines the smallest amount of letters a word can have, the greatest amount of letters, and the number of attempts the player has before losing, respectively. Included, is these private variables’ corresponding public setter and getter methods. The Difficulty class also contains a public copy constructor and public operator overload of the = operator. These are used by the Game class’s SetDifficulty method to conveniently assign a difficulty to it. This class also has a default constructor which sets the integers to its default values. The class also has a constructor that takes in 3 integers that are then used to set the three integer variables to those integers.

**How the Program was Designed**

The program contains the following four text files: Drawing.txt which contains the picture of the hangman, engmix.txt which is a list of over 80,000 words, MenuScreen.txt which contains the text for the main menu, and SaveFile.txt which stores information about any saved game or is blank if it doesn’t. When the program is run, a pointer to a Game object called hangman is created that will be used to contain the state of the game. There are two sections to the main code which is the main menu code and the gameplay code.

The menu code shows the menu text and gives the user the option to start a new game based on the difficulty of choice, continue game, or quit the game. When a difficulty is selected, a corresponding Difficulty object is created, which becomes the Game object’s difficulty. Then, engmix.txt is read and a random word from it is selected; this will continue until the selected word meets the required length for the difficulty. The variables in the Game object are set to their corresponding values. If continue game is selected, then the SaveFile.txt will be open and values from it will be read and used to set the variables in the Game object based on the information in the text file. If quit game is selected, the program closes.

If the user chooses a difficulty or continue, the gameplay code will run. This code will print a drawing of the hangman which depends on the number of attempts left, the progress the player has made so far, and some text telling the user what they can enter. Based on the user’s input, a single-letter or whole-word guess can be made or the game’s current state will save. The Game object’s values are updated accordingly. Whenever the player makes a guess, the progress word is updated to show the guess if it the guess is correct or the attempts left is subtracted by one if the guess is wrong. Also, in the Game object there is an array which contains the letters the player has not chosen that will be updated. If the progress word is updated and there are no more letters to guess, then the winning code will run which tells the player that they won and redirects them to the main menu to play again. If the attempts left reaches zero, the losing code will run which tells the player they lost and the word and redirects them to the main menu to play again. If the player enters that they want to save, then the Save function will run which reads the current Game object’s values and stores them in SaveFile.txt. If the game has not ended yet, then the gameplay code will continue to run, using the values of the updated Game object.

**How to Operate and Play**

To play Hangman, open the program up by opening the project on Visual Studios and starting the program. The menu will provide 3 values you can enter that correspond to 3 different options (“New Game”, “Continue Game”, and “Quit Game”). Entering “1” selects the first option, entering “2” selects the second option, and entering “3” selects the third option (enter them without the quotation marks). If “New Game” is selected or “1” is entered, you can select the difficulty. Entering “1” selects the easy mode, entering “2” selects the medium mode, and entering “3” selects the hard mode (enter them without the quotation marks). Once the difficulty is selected or “Continue Game” is selected, the game of Hangman will begin. The dashes (“-”) indicate the letters in the word you have not guessed. The goal in the game is to reveal all of the dashes by guessing the right letters or by guessing the word. Enter a letter from a to z (lowercase) to guess a letter. Enter a question mark (“?”), then enter what you think the word is to guess the word. You can’t enter previously guessed letters again and every time you guess wrong you lose an attempt. Enter an asterisk (“\*”) to save the gave. Continue guessing until you guess the word or run out of attempts. If you guess the word then you win and if you run out of attempts then you lose. Once, you win or lose, enter any value to go back to the main menu.