Milestone 3

Chad Shafer

SNHU

The artifact that I chose to represent is once again the Hangman game. Even though it is the same artifact as the last milestone, I will go over the details again. The code is written in C++ and was created a year or so ago. The program is composed of 7 functions and a main function. The first function is called printMessage. This function sets up the borders to the game and it called when the game wants to print out certain words. The next function is one that I recently added in. This function is called DrawHangman. Its purpose is to print a piece of the hangman every time the user enters an incorrect guess. The third function is called PrintLetters. This function is set to print a list of letters ranging from one character to another, leaving off characters that the player has already guessed. The fourth function is called PrintAvailableLetters. This function prints the letters for the player to guess. The fifth function is PrintWordAndCheckWin. This function prints the word the player must guess. It shows the letter if the player has already guessed it and shows and underscore if the letter remains unguessed. The sixth functions is named LoadRandomWord. Just like the name suggests, this function loads a file containing a list of words and picks a random word to use for the player to guess. The seventh and final function before our main is called TriesLeft. This function determines how many incorrect guesses the player has made. Finally, is our main. This is where our program is run and calls to the other functions. When working correctly, the user has a fun little hangman game.

I used this artifact because I feel that this program, along with the enhancements, is a good representation of the range of skills needed to perform as a coder. I believe that this code demonstrates a combination of knowledge, aptitude, technical capabilities and problem-solving when it comes to developing a working code based of user’s expectations. While I have utilized different coding languages such as Java, Python, and SQL, I prefer to work with C++ as it seems most appropriate to the field of choice.

When it comes to algorithms and data structure, I feel this code is a great representation and example. Programming algorithms are the “recipe” or procedure for solving a problem. For instance, I wanted the function LoadRandomWord to do just that, load a random word. I needed the code in that function to be a specific sequence of actions that tell the program exactly what to do and receive an output. Data structures organizes data in the memory such as an array, or vector. The way that my code is set up, organized, and the way the functions are built are all great showcases of a good algorithm. For the data structure, one part of the code showcases this skill better than others. This part of the code is also part of my enhancement. In my function LoadRandomWord, I have the reader go into a save .txt file, load the file containing a list of words and save them into a vector. I have multiple data structures in play in this function such as a string, vector, and integers. Once the words are loaded into the vector, we choose a random word from the vector for the player to guess, using rand() and the size of the vector. The only problem is that the word is not too random. This is because that rand generates a range of integers by a mathematical algorithm. The numbers are given a starting number and if that starting number is the same it will always be the same sequence. To get around this I wanted to change the seed value. In my main I used srand which sets the starting value. I did not want to just give it an integer because the starting value will always be the same. To solve this problem I used the time() function. With srand(time(NULL)), I am amble to change the seed value based off of the time the user started the game, from the computer’s internal clock. That way this will always be different.

Enhancing this randomizer was a lot of fun. I played around with different equations and ideas on how I can randomly generate numbers. The hard part was knowing that there is no such thing as “random numbers” in computers, just pseudo-random numbers. Doing research on how to utilize this in my code was simple. I already had the base, I just needed to identify a value to seed and I needed that to change every time.