**PROJECT**

**Self-Assessment**

****

**Prepared by: Sandali Oveena Widyaratne (A1882491)**

**OVERVIEW**

The link to my VIDEO: <https://youtu.be/usk30mZ0vm8>

**Please note that my final files are labelled mainGame.m and playHangman.m.**

For my MATLAB project, I have implemented the Hangman game because I believe it provides a good balance of complexity and simplicity, making it suitable for me to practice concepts such as loops, conditional execution, user input/output, arrays, and functions, which I have learned throughout my course.

My Hangman code is structured into multiple stages, beginning with the basic hangman logic and then gradually incorporating features such as user-interface setup, difficulty levels, and ASCII art hangman representation.

Here is my written self-assessment against the rubric criteria, where I identify the rubric level, I have reached and provide evidence:

**RUBRIC CRITERA ASSESSMENT**

**Criteria 1: Conceptual coverage- demonstrates correct use of MATLAB programming concepts.**

I believe I have made use of all programming concepts mentioned in the practical correctly:

* The input function is used to get user input for guessing letters in the hangman game. There is also a validation on the user input taken, to prevent errors. In my game, even if the user inputs uppercase or lowercase letters, they are considered equal. Repeated letters are filled automatically and displayed for user convenience.
* The ‘disp’ function is utilised to display messages, current status, and ASCII art of the hangman.
* The ‘questdlg’ function is used to create a customised dialog box for selecting difficulty level.
* I have made used of vectors to store guessed letters and represent the guessed word, moreover, matrices have been used to represent the hangman drawings of the ASCII art.
* The ‘while’ loop is employed to iterate through the game until the player either wins or loses. The looping constructs are utilised for creating typing effects and displaying hangman drawings based on the number of incorrect guesses.
* I have made use of functions to modularise and perform specific tasks and have used it throughout to enhance code reusability and readability when I came across executional errors. I have made use of a Callback function that was new to me in order to handle events triggered by user interactions with GUI elements.
* Conditional statements (‘if’, ‘switch’) are used to implement branching logic based on user input and game conditions. The difficulty level chose by the user determines which word list file to read from (wordsSmall.txt for Easy and wordsHuge.txt for Difficult).

Overall, the code demonstrates a comprehensive understanding of effective application of programming concepts including some new concepts that I have learned. I believe I have made use of modular functions, control structures, and appropriate data structures to enhance code organisation, readability, and maintainability.

**Criteria 2: Value-add- The amount of value that you added in your assignment. How much coding and conceptual effort is demonstrated by your code. Is the code that other sources contributed clear from your comments? Is the functionality of that code (as the program runs) substantial?**

In my code, I have made use of extensive coding efforts to provide excellent functionality and creativity in design of user interaction:

* I have made use of GUI implementation to enhance user experience by providing visually appealing and interactive way to play the game. I have designed this via CANVA (Appendix 1).
* The custom dialog box for selecting the difficulty level adds an extra layer of engagement and polish to the game as well (see MathWorks, 1; YouTube, 1).
* I have also made use of “typing effects” for the GUI set-up which I feel arouses the sense of suspense in the MATLAB and gives it game-like vibes when user is interacting. This makes the game more immersive to the user.
* I changed my text properties for appearance and behaviour of a Text object with the help of MathWorks (see MathWorks, 2).
* I have used this project as a chance to improve myself, therefore challenged myself to display some version of ASCII art as inspired by the course seminars that displayed creative use of ASCII art. I have used ASCII art for depicting the hangman’s progression clearly to convey the state of the hangman, contributing to the overall gameplay experience. The’ drawHangman’ function implemented in stage 4 of my model demonstrates this.
* Moreover, each of my functions encapsulates a specific aspect of the game logic, making it easier to understand and modify individual components.
* I have also made use of a Callback function to define a pointer for the name input box and the play button (MathWorks 3). I learned that a callback command is a command that can help execute a response for a set of predefined user actions when setting up GUI features.
* The code includes comments that provide explanations for the purpose and functionality of different sections, enhancing clarity and understanding as I progress. I believe that the functionality of the code contributed by other sources is clear from the comments and the integration into the overall game logic.
* The program offers a choice for the user by implementing a game with a difficulty level of the user’s choice adding more variation, gives a random word selection, user input handling, and win/lose conditions.

All of them together offer a complete and entertaining hangman game experience for the user.

**Criteria 3: Incremental Development- are there intermediate MATLAB and test files that indicate a clear path to development of the project?**

It has been a challenge to focus on code logic while keeping records of substantial evidence over the course of weeks I have developed the game. I have developed my code in the following incremental stages:

* **Stage 0- Basic Hangman game logic**: in this stage, I tried to focus on implanting the core logic of the hangman game without any complications. It includes the basic functionality such as word selection, user input handing and the hangman display. I wanted the game to begin by selecting random words from a defined list. This was achieved by generating a random index within the range of the word corresponding to that index. I sketched out that the game should progress through a main loop where the player makes a guess until either the word is guessed correctly, or the maximum number of incorrect guesses is reached. And, after each guess, the game checks whether the guessed letter is present in the word. If the guess is correct, the corresponding underscores in the guessed word are replaced within the guessed letter. If the guess is incorrect, a count is kept until max incorrect guesses are reached. Finally, the game determines whether the player has won or lost based on the outcome of the main loop.
* **Stage 1- User Interface Setup**: in this stage, the focus shifts to setting up the GUI for the game. Overall, the initial setup provided an intuitive user interface that guided the player along the game. I wanted to implement the entire game in GUI, however, was not able to under time constraints. Regardless, to show my understanding of the concepts and the ‘can-do’ possibility I have added predefined MATLAB functions to create and add pop-up windows for the first part of code. Here, it was in this stage that I struggled to learn and understand the use of new concepts to gather information to make my GUIs presentable. For my dialog box, I have made use of Cyan(#00FFFF) and Luminous Pink(#F535AA) colours as this corresponds to my designs for the GUI. I have advanced these features by adding colour properties to match with my design, changed font styles and even given user the option to opt for their level of difficulty using a dialog box in my latest development.
* **Stage 2- Hangman Game Initialisation**: Here, the game’s initialisation process is refined to include prompting for the difficulty level and reading word lists from files based on the selected difficulty level. The code for choosing a random word and initialising game variables were implemented. Depending on the selected level, the scrip reads a word list from a file for easy and difficult modes. Finally, once a word is chosen, the script initialises several variables crucial for the game (‘guessedWord’ is a string of underscores representing each letter of word to be guessed; ‘incorrectGuesses is a counter to keep track of the number of incorrect guesses made by the player; ‘maxIncorrectGuesses’ is the maximum number of incorrect guesses made by the player; ‘guessedLetters’ is an array to store the letters that the player has guessed so far. This helps prevent the player from guessing the same letter multiple times.
* **Stage 3- Play Hangman Testing**: although not explicitly labelled as a separate stage, the playHangmanTesting() function serves as a testing stage where the core hangman game is integrated and tested. Here, I have modified the code so that user input is validates, script updates and checks for guessed letters and keeps count of them. More importantly, this stage was created to check if the main game loop is working and returns arguments pf code correctly.
* **Stage 4- Hangman Drawing in ASCII art**: In this section, I have made use of a game function specifically dedicated to drawing the hangman using ASCII art. By isolating the drawing logic into its own function, I was able to use the function across different stages of the game development process. I believe this was the best example of my incremental development stage because I was able to easily locate and modify the drawing logic without affecting other parts of the codebase. This simplified troubleshooting and debugging process when adding visual elements to the hangman drawing.

An important point in the incremental stage was that o have organised my code into separate functions, making it easier to manage and maintain during development. Hence, it was not very ideal for me to backup all code as I only had to change specific features in the function when errors occurred. I have attached these files for your reference.

**Criteria 4: Testing Strategy- evidence of testing through test files and intermediate versions of MATLAB code**

To satisfy testing criteria, I have made use of test files that have been created to verify the functionality of individual components of code or functions. As attached in my submission, the test files contain test cases designed to assess whether each function behaves as expected under various conditions. As the hangman game code was developed overtime, I have also attached different versions of MATLAB code at specific points in time showcasing the progression of the project. Each intermediate version likely includes additions, modifications, or improvements made to the project since the previous version. Once completed, I have also been able to showcase my code to several of my friends to gather feedback and this has also helped me identify any usability issues, bugs, or areas of improvement before final release. Kindly find the attached files for your validation.

A screenshot of a computer

Description automatically generated

The above code displays one of the earliest versions to the mainGame function, whereas the picture below is a snapshot of the latest:

A screenshot of a computer

Description automatically generated

**Criteria 5: Comments and style- consistent use if indenting. Consistent and sensible use of variable names. Commenting throughout all versions of program.**

I believe I have made use of excellent code consistently to exhibit all elements of good code stye through all versions. You can find comments through all versions on compilation errors, why there is an error, why some codes had to be improved or any other commits I had to make in order to arrive at the final project code. By formatting my code consistently in such a manner, I was able to evaluate my code functionality. Indenting my code consistently allowed me to see if there are any deviations and I made sure to use meaningful variable names so that I wouldn’t get confused of each variables purpose in each version of my code. My functions too followed a certain variable naming pattern. For instance, function playHangman42() would be the second version of code in my 4th attempt.

**SOURCES USED**

**MathWorks**

1. Custom dialog box Yes or NO <https://au.mathworks.com/help/matlab/ref/questdlg.html>
2. Text Properties <https://au.mathworks.com/help/matlab/ref/matlab.graphics.primitive.text-properties.html>
3. Callback Functions <https://au.mathworks.com/help/matlab/creating_plots/create-callbacks-for-graphics-objects.html>

**YouTube**

1. <https://www.youtube.com/watch?v=zqnA1uijFVs> <https://www.youtube.com/watch?v=vRC6556jKJw>

Appendix:

1. Design <https://www.canva.com/design/DAGCv8v9NIQ/5dj1gRJuf0lw-Zy3h-GpPg/edit?utm_content=DAGCv8v9NIQ&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>