**Project Report (HangMan)**

**Credits For:**

1. **Abanoub Amir (Team Leader);;**
2. **Khaled Ashraf;;**
3. **Ahmed Mohamed Reda;;**
4. **Omar Mohamed Abd Al Razik;;**
5. **Omar Khaled Tharwat;;**

**Functions Explanation:**

**Action Buttons:**

1. GoToX (char StartMenu[ ][40], int& i):

-This function takes the Currently Displayed Menu and the Value of i as where the Cursor's place is;

2. GoToY\_1 (char StartMenu[ ] [40], int& i):

-Currently Displayed Start Menu and the value of i which is the ’>’ position;

3. GoToY\_2 (char StartMenu [ ] [40], int& i):

-This function was made to navigate the second column of the Category menu, other than that it's the same as the previous function except for the Column value of course;

4. EnterKey (int& i):

-Receives the i's value and checks if user pressed Enter or not and Returns the i if user pressed Enter;

**Data Base:**

1. **read\_file (int Category, int Difficulty):**

-Read the files based on the category and difficulty chosen;

1. **get\_word ():**

-Randomize a word from the array of string words, and check if the word randomized have been used before or not;

1. **dashes (string word):**

-Takes the word and changes it into dashes;

1. **letter\_appear (char letter):**

-Takes the pressed letter and changes the dash into this letter if it exists in the word;

1. **hint ():**

-Randomize a number and then check if the letter in the index randomized is equal to dash then let it appear;

1. **check\_word\_right (string dash[ ]):**

-Checks if the word is completely guessed correctly or not;

**Game Drawings and Sound Effects:**

1. **Intro ():**

-Displays “Welcome to HangMan” and plays the intro music;

1. **ErrorMessage** **():**

-If the player entered a wrong letter, this function displays red colour;

1. **PlayMusic** **():**

-Plays the background music during all the game;

1. **DisplayGameOver** **():**

-When the player loses, it displays the game over figure;

1. **DisplayHangManStick )int streak):**

-Displays the Hangman figure according to the number of the wrong letters the player have entered;

**Game Keyboard:**

1. **Display\_Letters (char Scoped\_Letters[3][10], int** **Number\_Of\_Wrongs):**

-Displays the Keyboard;

1. **Check\_Letters (char Char\_Selected, char** **Scoped\_Letters[3][10]):**

-Receives the pressed letter and changes the pressed letter into ‘#’ in the keyboard in order not to be pressed again, and displays the keyboard after the change;

1. **GetPressedLetters ():**

-Detect the pressed character either a letter or shift or backspace;

1. **check\_Hash\_Letter (char Char\_Selected, char** **Scoped\_Letters[3][10]):**

-Takes the selected character and checks if it has been pressed before or not;

**Menus:**

1. **DisplayStartMenu ():**

-Displays the start menu and the cursor moves inside it;

1. **DisplaySingleOrMulti ():**

-Displays the single and multiplayer menu;

1. **DisplayPlayMenu ():**

-Displays the categories of the game;

1. **DisplayDifficultyMenu ():**

-Displays the difficulty (Easy or Hard);

**Game Start Function:**

1. **StartGame (int category, int difficulty)**

-Starts the game with the chosen category and difficulty;

**Code Explanation:**

At the beginning, **Khaled\_reading** function is called to read the text file that the leaderboard is saved in it, after that **Intro** is called to play the music of the intro and a thread is activated to enable the background music and the game function to multi thread at the same time.

The start menu appears and then the player choose any of the choices and the function **DisplayStartMenu** returns the value of the chosen option.

If the player chose Start then the function returns (1) and Displays the next menu by the function **DisplaySingleOrMulti** and returns the value of the chosen option;

If the player chose leaderboard, the leaderboard appears by the function **DisplayLeaderBoard**.

If the player chose Exit, the game ends.

In the first case when the single or multiplayer menu appears, if the player clicked backspace or escape the game returns to the previous menu.

If the player chose Multiplayer, the function **MultiplayerMode** is called.

If the player chose single player, the next menu appears which is the categories menu by the function **DisplayDifficultyMenu**, if the player clicked on backspace the game returns to the previous menu, else the game starts with the chosen category and difficulty.

The Single Player Mode:

At the beginning, all the inputs is cleared, then the player is asked to enter his/her name. **read\_file** function is called to open the chosen category file which contains the words of this category. Then the keyboard appears at the bottom of the console window.

If the player guessed correctly at least one word, the vector of the score and the hint points will be accessed, otherwise the score and the hint points are zero.

The function **get\_word** is called to get a word by random from the text file and put it in a string variable. Then the function **dashes** is called to turn the characters of the word into dashes ‘\_’.

If the player clicked on the shift button but he/she must have guessed at least one word correctly, the hint system will work and a random letter appear in its correct place in the word.

The function **Letter\_appear** is called to check if the letter selected exists in the word or not, if the letter doesn’t exist and if TIMEPRESSED>1, and the function **check\_Hash\_Letter** is called to check if the selected letter have been pressed before or not and if not and if all these conditions are true, the number of wrong answers increase and the function **ErrorMessage** is called to display the red colour when a wrong letter is selected.

If the number of wrong answers reaches 9, the word appears without any dashes to the player and then he has lost the game.

The function **DisplayHangManStick** is called and the figure of hangman appears according to the number of wrong answers the player has entered.

The function **Check\_word\_right** is called and if the player has guessed the whole word correctly within the limit of the wrong answers, the function **CorrectMessage** is called to display some cheerful colors and a thread is activated to play small track of music during the displaying of the colors.

The correct words increases and passes the current level, and the score and the hint points increases and reset the keyboard to its original state without any letter being changed into ‘#’.

If the number of wrong answers is less than 9 which is the limit of wrong answers when the player reaches it he/she loses, ………

If the player clicked on the escape button during the game, he/she is asked if he/she is sure to exit the game. If the answer was ‘y’, the game exits, if the answer was ‘n’, the game returns to its last state.

The Multiplayer Mode:

It is very similar to the single player mode but there is small difference which is the two players choose a maximum score which is if one of the players scored this score before the other player he wins the game.

The first player enters a word and It is turned into dashes and the second player guesses the letters of the word, the second player tries to get the highest score he can get or the maximum score they chose before the game starts and when he loses and can’t guess the word correctly and his/her score hasn’t reached the maximum score they chose before the game starts, the turn is turned to the first player to guess a word which is entered by the second player, if the first player got score more than the second player he won the game even if he hasn’t scored the maximum score, and if the first player didn’t reach the second player score, the second player wins the game.

The function **Khaled\_Reading** is called to read the previous names and scores that have been played before and writes them in a vector.

When the player guesses a word correctly, the function **Khaled\_Fun** is called to check if the player name is already in the vector, then it increases his/her score and hint points by 10 points, if the name isn’t found, the vector is pushed back and the name and the score and the hint point will be written in the vector and the score and the hint points increases.

After that, the function **Khaled\_Sort** is called to check if the name exists in the original vector which is called Scores or not, if it exists the score and the hint points and the name are sorted, if it doesn’t exist the name will be copied from the current vector to the original vector and then it will be sorted.

At the end, the function **Khaled\_Writing** is called to save all the vector which contains the name and the score and the hint points in the leaderboard text file.

**Role of the members**

**Abanoub Amir:**

The Team Leader, distributed the tasks on all of the members.

Made the game drawings and sound effects, made the action buttons.

**Omar Khaled:**

Responsible for the Multiplayer mode, and put some of the sound effects, wrote the project report.

**Ahmed Mohamed Reda:**

Designed the game logic and part of the database.

**Khaled Ashraf:**

Responsible for the leaderboard and the score and hint system.

**Omar Mohamed Abd Al Razik:**

Responsible for the keyboard display and part of the database, made the top-down-design.