B

Liam Baron

EDUV4813733@VOSSIE.NET

Abstract

I have attempted to create a hangman application

Basic c# project

Liam Baron – EDUV4813733

PROJECT SPECIFICATIONS:

* Visual Studio Code 2019
* Windows 10

HOW TO RUN THE APPLICATION:

* First unzip the folder that was submitted.
* Open Visual Studio Code.
* Click on open Project Solutions and select the location of the project.
* Finally just press start once everything is loaded up or press F5 on your keyboard.

HOW TO USE HANGMAN APPLICATION AND WHAT TO EXPECT:

* First run the application then you will be met with the main menu.
* You have the ability to quit the application with the button on the top right or the ability to create a new game. Click new game
* You will then be designated to a new form when this happens the game has already started.
* All you have to do now is guess the word by typing a letter on the keyboard there isn’t any prompt or box to put in the input so hitting enter won’t confirm it once you hit the key it will process the letter.
* When this happens you will either get it right then the letter will appear in the hidden word.
* If you get unlucky then your guesses left will decrease.
* You can go to the main menu mid game as well. This won’t affect your score.
* Once you succeed or fail you will receive a message box and the form will close that will be the end of the game and you will be able to see your score and the hidden word that was selected.

PROGRAM DESIGN:

* First I will create the main menu with two buttons one that will exit the application and one that will create the new game in another form it will also have a scoreboard that will only be visible once you have attempted a game.
* Once the form is created there will only be one button that will return you to the main menu and the form will then close.
* Now the program will be waiting for input from the user, with this input the program will check if the letter is contained in the word if its not then the guesses left will go down and the user will be able to see the incorrect letters.
* When the game is done the form will close and a message box will be displayed stating the word that was hidden and the score will then pop up in the scoreboard list box.

MainMenu.cs:

/\*

\* Filename: MainMenu.cs

\* Name: Liam Baron

\* Created: 08/12/2023

\* Operating System: Windows 10

\* Version: Visual Studio 2019

\* Description: This file will handle the main menu interactions and it also creates

\* the game form where the game takes place

\*/

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hangman

{

/\*

\* Date: 08/12/2023

\* Author: Liam Baron

\* Purpose: This is the Main Menu of the game, and this holds the scoreboard information and the ability to exit the

\* application

\*/

public partial class MainMenu : Form

{

// This will give us the ability to create a form

Game secondForm;

// This is being used in the second form meaning that the scores will be available until the appllicaiton is closed

public SortedList score = new SortedList();

public MainMenu()

{

InitializeComponent();

}

private void fileSystemWatcher1\_Changed(object sender, System.IO.FileSystemEventArgs e)

{

}

private void exitGame\_Click(object sender, EventArgs e)

{

// The application will exit with this method

Application.Exit();

}

private void newGame(object sender, EventArgs e)

{

// Creating a new form that will be able to play the game

secondForm = new Game(this);

secondForm.Show();

}

private void MainMenu\_Load(object sender, EventArgs e)

{

// This makes sure the scoreboard is not visible until a game has been played

scoreBoardPanel.Visible = false;

}

}

}

Game.cs:

/\*

\* Filename: Game.cs

\* Name: Liam Baron

\* Created: 08/12/2023

\* Operating System: Windows 10

\* Version: Visual Studio 2019

\* Description: This file will handle the game interactions

\*/

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hangman

{

/\*

\* Date: 08/12/2023

\* Author: Liam Baron

\* Purpose: This form will handle the game interactions like accepting only characters

\*/

public partial class Game : Form

{

// This variable will be used for creating a game instance

public NewGame x;

// This is a flag that will monitor the game state

bool stasis;

// This will help us with getting information from the first form. I will be using this to control

// the visibility of the panel in the main menu form

MainMenu firstForm;

public Game(MainMenu firstForm)

{

this.firstForm = firstForm;

x = new NewGame();

// Setting the game status to playing

stasis = false;

// Creating the special characters to hide the word

x.DisplayWord = String.Concat(Enumerable.Repeat("\*", x.word.Length));

InitializeComponent();

// I am now displaying the hidden word in the label

HiddenWord.Text = x.DisplayWord;

}

private void MainMenu\_Click(object sender, EventArgs e)

{

// This closes the form

this.Close();

}

private void Game\_KeyPress(object sender, KeyPressEventArgs e)

{

// This checks if the game is still being played

if (!stasis)

{

// And if it is we are only allowing letters to be used there is why there

// is another if check

char guess = char.ToLower(e.KeyChar);

if (char.IsLetter(guess))

{

// I am checking if the letter is valid through the checkGuess method

// The if else check here is for the end game state where it checks if this person has

// succeeded in the game or if the person has failed

HiddenWord.Text = x.checkGuess(guess);

// Here we are checking if the text does not have an asterisk in it if it doesn't it means they have won

if (!x.DisplayWord.Contains('\*'))

{

// The game is now complete so we change the status of the stasis variable to true

stasis = true;

Console.WriteLine("Game won!");

// This if else will allow us to check if the word that was attempted was previously attempted and

// if so it will just update the score in the SortedList but not in the scoreboard listbox

if (firstForm.score.ContainsKey(x.word))

{

firstForm.score.SetByIndex(firstForm.score.IndexOfKey(x.word), x.guessesCount);

}

else

{

firstForm.score.Add(x.word, x.guessesCount);

}

// I am now adding the word with the amount of guesses left as the score

firstForm.ScoreBoard.Items.Add($"{x.word} : {x.guessesCount}");

}

// This checks if the player has run out of guesses

else if (x.guessesCount == 0)

{

// I am going to end the game by changing its status

stasis = true;

// This if else will allow us to check if the word that was attempted was previously attempted and

// if so it will just update the score in the SortedList but not in the scoreboard listbox

if (firstForm.score.ContainsKey(x.word))

{

firstForm.score.SetByIndex(firstForm.score.IndexOfKey(x.word), x.guessesCount);

}

else

{

firstForm.score.Add(x.word, x.guessesCount);

}

firstForm.ScoreBoard.Items.Add($"{x.word} : {x.guessesCount}");

}

// This will check if the game state has changed and if it does the scoreboard panel will now be

// visible with a score and a message box will be displayed that will show you the hidden word

if (stasis)

{

firstForm.scoreBoardPanel.Visible = true;

this.Close();

MessageBox.Show($"The word was {x.word}\n", "Result");

}

// This displays the amount of guesses left

guessesLeft.Text = $"You have {x.guessesCount} gueses left";

incorrectLetters.Text = x.WrongGuesses;

}

}

else

{

Console.WriteLine("Game done");

}

}

}

/\*

\* Date: 08/12/2023

\* Author: Liam Baron

\* Purpose: This will handle the letters input to give us the guesses left, incorrect guesses, formatting

\* and calculating the score board

\*/

public class NewGame

{

// The arraylists for incorrect and correct

public ArrayList correctGuesses = new ArrayList();

public ArrayList wrongGuesses = new ArrayList();

// This will help with formatting the label so that it is easy to see the

// incorrect letters in this case I am adding spaces between each letter

public string WrongGuesses

{

get

{

StringBuilder sb = new StringBuilder();

if (wrongGuesses.Count != 0)

{

// I am using the wrongGuesses arraylist to display wrong letters

sb.Append("Wrong guesses : ");

sb.Append(wrongGuesses[0]);

for (int i = 1; i < wrongGuesses.Count; i++)

{

sb.Append(" " + wrongGuesses[i]);

}

}

return sb.ToString();

}

}

public int guessesCount = 10;

private string displayWord;

// Again I am formatting the hidden word this time around so that there is

// spaces between each letter

public string DisplayWord

{

get

{

StringBuilder sb = new StringBuilder();

sb.Append(displayWord[0]);

for (int i = 1; i < displayWord.Length; i++)

{

sb.Append(" " +displayWord[i]) ;

}

return sb.ToString();

}

set {

displayWord = value; }

}

public string word = "";

// Everytime we create an instance for the NewGame Class it creates the words and puts them into an

// array then we get a random word for the game

public NewGame()

{

string[] words = new string[4];

words[0] = "apple";

words[1] = "english";

words[2] = "afrikaans";

words[3] = "cathedral";

randomWord(words);

}

// This just returns the random word that was picked up by the random class

public void randomWord(string[] words)

{

Random rnd = new Random();

int num = rnd.Next(0, 4);

word = words[num];

}

// This method will just subtract the amount of guesses from the guessesCount variable

public int guessesLeft()

{

return --guessesCount;

}

// So this method will format the hidden word to only add the characters that was

// guessed correctly and make that letter pop up in the hidden word

// using the correct letter from the correctGuesses made

public void labelConversion(char letter)

{

char[] hold = displayWord.ToCharArray();

for (int i = 0; i < word.Length; i++)

{

if (word.ToCharArray()[i] == letter)

{

hold[i] = letter;

}

}

DisplayWord = new string(hold);

}

// This method will help us check to see if the letter inputted is correct or incorrect

public string checkGuess(char letter)

{

// This if check checks if the letter is contained in the word and if it is not

// in the correctGuesses arraylist already

if (word.Contains(letter) && !correctGuesses.Contains(letter))

{

// Adds that correct letter to the correctGuesses arraylist and then formats it so

// that it is ready to be displayed

correctGuesses.Add(letter);

// This changes the DisplayWord variable

labelConversion(letter);

}

// if it is wrong we will then check if that letter is already present in the

// wrongGuesses arraylist and if not add it and substract the guesses left

else if (!wrongGuesses.Contains(letter))

{

wrongGuesses.Add(letter);

guessesLeft();

}

// when the if check was false the hidden displayed word will not change as that is the incorrect

// letter

return DisplayWord;

}

}

}