University of Dhaka

Department of Computer Science and Engineering

Project Report:
Fundamentals of Programming Lab(CSE-1211)

Project Name:
Words

Team Members:
Group ID: 2021CSE1211022
Member -1:
Name: S. M. Imran Rahman
Roll: 40
Member-2:
Name: Nirjhar Singha
Roll: 46

Introduction:

The name of the project is ‘Words’. This project is a game of words. There are two major modules in the project. These modules are:

1. Guess the word

In this module, the player has to guess a random word with the help of some clues. At the beginning, there will be some blank spaces equal to the number of letters in the word. If the player enters any character that is present in the word, then that character will be displayed in the dashed place/places as a hint. The player has to guess the word in a particular number of attempts and every wrong answer will decrease his chance of winning the game.

2. Search the word

In this module, a square size puzzle (size varying from 10-15) of random characters will be displayed. In this module, there will be two major sections.

i.Word Find: From a group of ten words, the player has to find which word is present in the puzzle.
ii.Position Find: Given a random word, the player has to find the position of the word in the puzzle.

In both sections, the player has to find the correct answer within a particular time. Dual player options will also be available. In this case, the player who gives the correct answer within less time will be the winner.

Objectives:

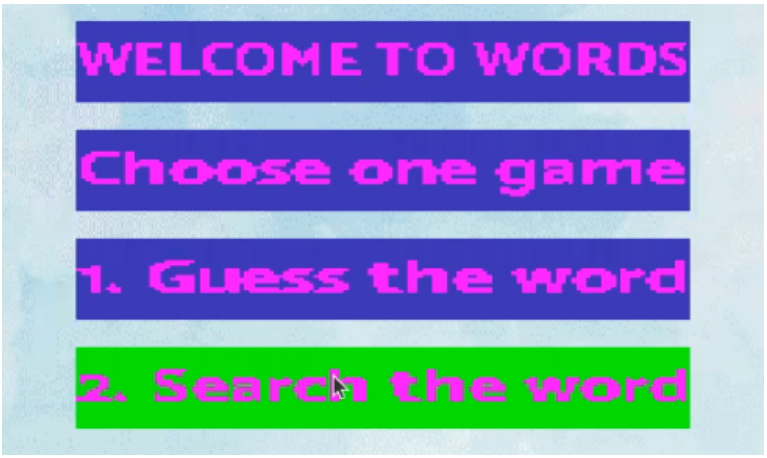
This project is mainly a game application. The concept of this game is to play with words. It becomes more interesting when the player needs to think creatively to crack the hints as well as solve the puzzle to win the game. The objective of this project is to get introduced with the field of game development. Our expectation

from the project was to build an end-to-end game application which will be enjoyable for the player to play.

Project features:

1. Display buttons:

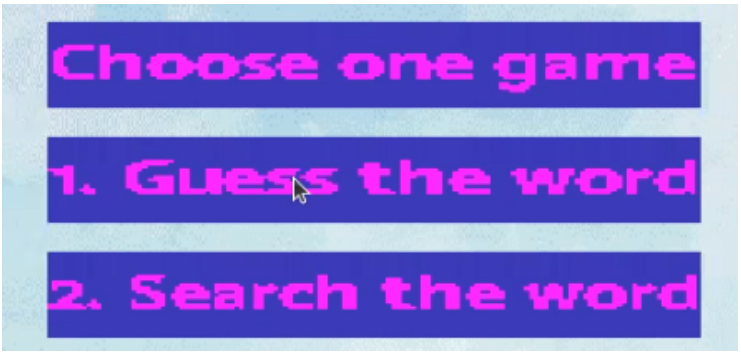
In order to improve the graphical interface as well as user experience, we displayed buttons to show among multiple options, which one is selected.

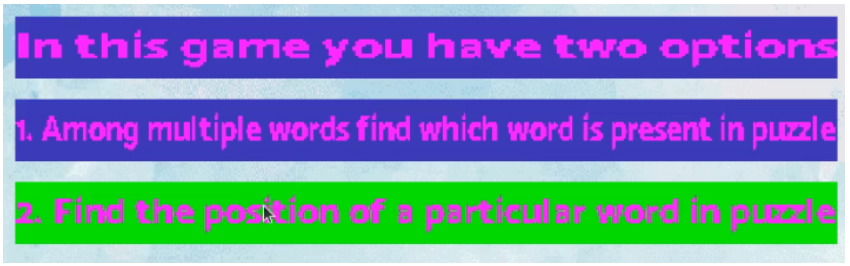


2. Multiple options in one game:

There are two major sections in the game- one is “Guess the word” and another is “Search the word”. Again in the “Search the word” section, we further have two options- “Word Find” and “Position

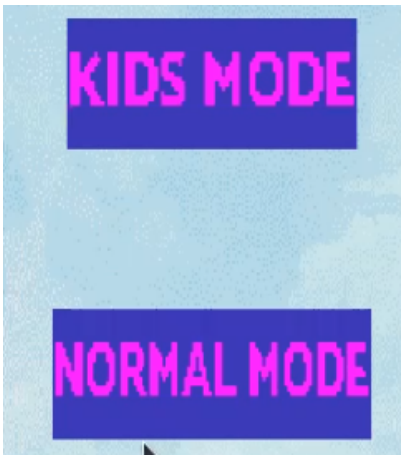
Find". All these options together make the game more interesting and give the user a better experience.

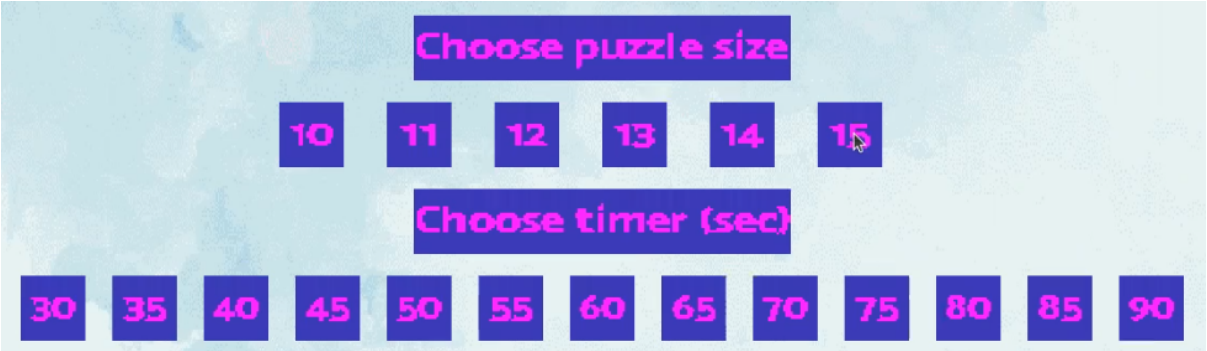




3. Different difficulty levels:

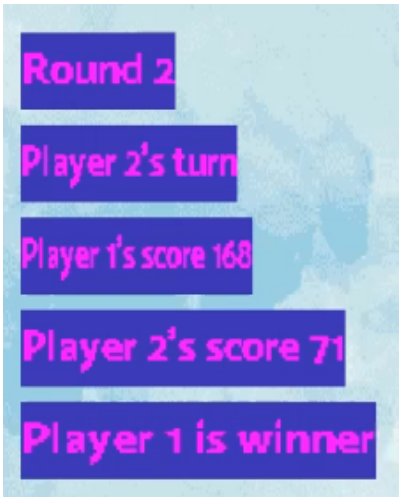
For better user experience, we introduced different difficulty levels in the game.





4. Dual Player:

The game has dual player mode also. Competing with an opponent makes the game more enjoyable.



5. Creative hints and clues:

In the “Guess the word” section, the player will be given interesting hints and clues which make the player think out of the box so that he can find the correct word in a minimum number of attempts.

6. Timer:

In the “Search the word” section, we introduced a timer to give a time constraint. It makes the game more exciting.

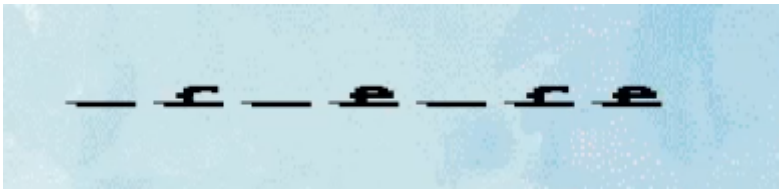
7. Handling edge cases:

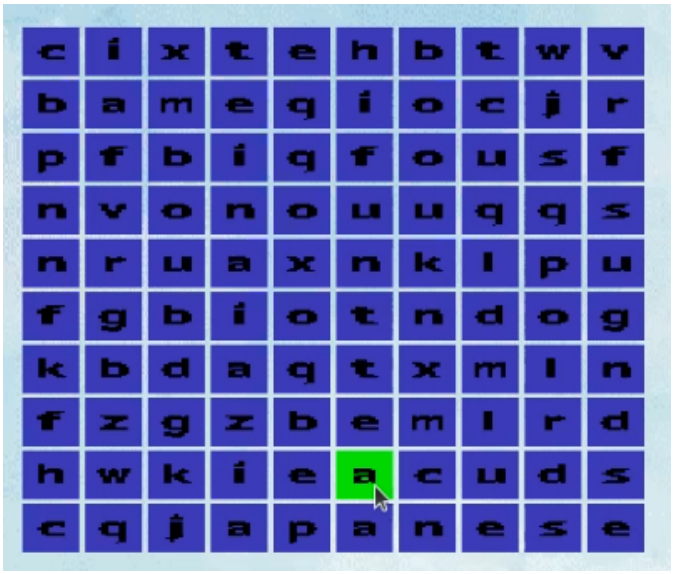
In the “Search the word” section, we are creating a puzzle with random characters. So there is a slight probability that in the “word find” module there might be more than one word present in the puzzle from the given group of ten words or in the “position find” module the given word might be present multiple times in the puzzle. Though the possibility of such cases is very rare, it has been handled in the game. In such cases, from all the possible answers anyone will be considered as the correct answer.

The header files “wordFind.h” and “posFind.h” can be referred to in this regard. The functions for cross-checking the result are defined in these header files.

8. Keyboard and mouse interaction:

Both keyboard and mouse interact with the game fluently.





9. Background music:

In order to improve the user experience we included background music in the game.

Project Modules:

The project is divided into two major modules-

1. “Guess the word”
2. “Search the word”

Some header files are related with only one of these two modules, some for both the modules. A brief description of all the header files is given below.

General purpose header files:

These header files are related with both the modules.

initializer.h :

Function in this header file:

initializeSDL(): This function will create an SDL window, initialise renderer, open TTF font to render text and start background music. All these things will be destroyed at the end of the game.

startGame.h:
Function in this header file:

gameInit(): This function will initialise the game loop.

chooseGame.h:
Function in this header file:

chooseGame(): This function will bring the page from where the player will choose whether he wants to play “Guess the word” or “Search the word”. According to the player’s choice the related functions will be called.

newFrame(): Whenever there is any change on the display, this function will clear the renderer and load the frame again.

From here the role of the header files that are related to only one of the two modules starts.

Header files related to module 1- Guess the word:
These header files are related to only module 1- Guess the word.

guessTheWord.h:
This header file will call the module-1. Function in this header file:

guessTheWord(): This function will call all other functions related to this module. The game loop that runs the module “Guess The Word” is defined here.

constants.h:
The constants, textures and some other global variables are declared in this header file.

textures.h:
Function in this header file:

texture_initialization(): The textures used in this module are created here.

main_render(): This function will load the background image when called.

mode.h:
Function in this header file:

mode(): Here the player will choose whether he will play “Kids mode” or “Normal mode”.
In “kids mode” some letters of the word will be given along with a clue statement. So, this is a mode of less difficulty.
On the other hand, in “Normal mode” only the clue statement will be given but no letter would be visible. So this is a comparatively difficult level.

mode_screen(): This function will load all the mode (“Kids mode” or “Normal mode”) options on the window.

buttonShow(): This function will display the buttons. In this case, if the cursor comes to an option, the colour of that option will be changed. If the cursor is moved away, then the colour is changed again and gets its previous look.

logic.h:
Function in this header file:

strlen(): This function will find the length of a string.
sdltocharacter(): This function will poll SDL events from the keyboard and convert the input into a character type variable.

validity_check(): When the player gives any input character, this function will check whether the character is present in the word or

not. If the word doesn't contain the character, then this will be considered as a wrong attempt and the player will lose one of his lives.

next_step(): This function will check whether the player wants to play another round or he wants to go to the "Menu" option.

logic_for_adult(): This function will be executed if the player chooses "Normal mode". Here no character of the word will be visible at first. This function will choose a random word and its clue from files and load the textures on the window. It will take input from the user, check the result and print the result accordingly.

logic_for_kids(): This function will be executed if the player chooses "Kids mode". Here some characters of the word will be visible at first. This function will choose a random word and its clue from files and load the textures on the window. It will take input from the user, check the result and print the result accordingly.

Header files related to module 2- Search the word:
These header files are related to only module 2- Search the word.

wordSearch.h:
This header file will call the module-2. Function in this header file:

searchTheWord(): Here the user will give input whether he will play "wordFind" or "positionFind".

loadFrame(): Whenever there is any change on the display in the event loop, this function will clear the renderer and load the frame again.

chooseOptions.h:
Function in this header file:

chooseOptions(): This function will call other functions to take input from the user for number of players, number of rounds, size of puzzle and total time.

choosePlayer(): To select number of players.
chooseRound(): To select how many rounds the player wants to play.
choosePuzzle(): To select the size of the puzzle.
chooseTimer(): To select total time to find the answer.

frameLoad(): Whenever there is any change on the display in the event loop, this function will clear the renderer and load the frame again.

showButton(): This function will display the buttons. In this case, if the cursor comes to an option, the colour of that option will be changed. If the cursor is moved away, then the colour is changed again and gets its previous look.

wordFind.h:
The functions that are related to the “wordFind” section are defined here.

findWord(): This function will call the function to create a puzzle. It will randomly choose one group of ten words and one random word from the selected group. Then it will call the function editPuzzle() and displayPuzzle() to overwrite the selected word in a random position of the puzzle and display the puzzle. It will also call the functions to find and cross-check the result and print the result.

findResult(): This function will check whether the result is right or wrong.

crossCheckResult(): In some cases, from the group of ten words, multiple words might be present in the puzzle. In such a case any of

the possible answers will be considered as correct. This function will handle this with the help of isPresent() function. isPresent() function will check whether the input word is one of the given ten words or not. If yes, then the crossCheckResult() function will check whether the input word is present in the puzzle or not and print the result accordingly.

isPresent(): This function will check whether the input word is one of the given ten words or not.

copyWord(): Copy one string to another.

posFind.h:
The functions that are related to the “positionFind” section are defined here.

findPosition(): This function will call the function to create a puzzle. It will randomly choose one word from the word file. Then it will call the function editPuzzle() and displayPuzzle() to overwrite the selected word in a random position of the puzzle and display the puzzle. It will also call the functions to find and cross-check the result and print the result.

result(): This function will check whether the result is right or wrong.

crossCheck(): In some cases, there might be multiple occurrences of the given word in the puzzle. In such a case any of the possible answers will be considered as correct. This function will handle this to print the result accordingly.

wordCopy(): Copy one string to another.

gameEvents.h:
The following functions are defined in this header file.

createPuzzle(): This function will create the puzzle with random characters.

editPuzzle(): This function will overwrite the randomly selected word in a random position of the puzzle.

displayPuzzle(): This function will display the puzzle.

printResult(): This function will print the result of the game.

holdWindow(): Hold the frame until the user gives any input.

sdlTextureDestroy(): To destroy the textures.

sdlTextInput.h:
This header file contains functions that are needed for taking text input from the user in the “wordFind” section.

sdlTextInput(): In the “wordFind” section, this function will run an event loop to take which word is present in the puzzle as input from the user using the keyboard.

loadWindow(): When sdlTextInput() function runs an event loop to take input from the user, the timer updates its value every second. The loadWindow() function will display the updated time as well as the input given by the user.

sdlMouseInput.h:
This header file contains functions that are needed for taking input from the user using in the “positionFind” section.

sdlMouseInput(): In the “positionFind” section, this function will run an event loop to take the position of the word in the puzzle as input from the user using the mouse.

loadWindow(): When sdlMouseInput() function runs an event loop to take input from the user, the timer updates its value every second. The loadWindow() function will display the updated time as well as the input given by the user.

number.h:
There is one function in this header file.

randomNumber(): This function is used to generate random numbers. These random numbers are used to generate random characters to create the puzzle.

stringLib.h:
The functions that are related to strings are defined in this header file.

stringLength(): This function will find the length of a string.

stringCmp(): This function will compare two strings whether they are equal or not.

dualPlayer.h:
The function related to dual player mode is defined here.

doublePlayer(): This function will enable the dual player mode, calculate the score of each player and print the result accordingly.

From here the role of the header files that are related to only one of the two modules ends.
The role of the general purpose header files starts again.

closeGame.h:
The function that will close the game is defined here.

closeGame(): This function will display the page where the player gets two options- “New game” and “Game over”. If the player chooses “New game” then the game loop will keep running. If he chooses the “Game over” option, then the game loop will be terminated.

Team member responsibilities:

The game is divided into two major modules.

1. Guess the word
2. Search the word

Member-1:

S. M. Imran Rahman (Roll - 40)

Responsibility: Header files and functions related to module-1(Guess the word)

Member-2:

Nirjhar Singha(Roll - 46)

Responsibility: Header files and functions related to module-2(Search the word)

Platform, Library & Tools:

Programming language: C/C++

Graphics library: SDL2

SDL_Image library: To render image

SDL_TTF library: To render text

SDL_Mixer library: For background music

Limitations:

We tried to include as many features as possible in the game. But still some features can be added or improved. Such as:

1. The game is not responsive. We used fixed screen width and height in the game. If we run the game on a monitor equal to or greater than that size then this might not be a problem. But if we try to run the game on a small screen then this may cause some errors.

2. We didn't use any databases in the game. But in future if we want to scale up the project and we need to handle a huge amount of data then without databases it will be difficult.

3. To run the game the player must install some prerequisite libraries separately which might be a headache for a player.

4. The word files used in the game can be made more resourceful.

5. The overall graphics quality can be improved as well as complexity of the code may be optimised more.

Conclusions:

From the project, we learned the following things:

1. We learned how to divide a large code base into different functions and modules so that the code is easy to understand, update and upscale.

2. Another important learning is how to debug errors in a large code base.

3. We have learned how to combine a graphics library and other utilities with a programming language to handle the front-end with back-end.

4. We learned how to do team work. As the full project was divided between two team members, both of us had to be careful enough

so that one’s code doesn’t contradict other's code while merging. In this regard, modular programming helped us a lot.

5. The project has introduced us to the field of game development.

Our expectations at the beginning:

At the beginning the idea of the project was quite simple. We planned to make a game where the player will guess a word according to some clues. After that we tried to improve our project as much as possible. So we introduced two modules- “Guess the word” and “Search the word”. Further we tried to add other different features like difficulty levels, timer, dual player, buttons one by one most of which were not planned at the beginning. Our expectation was to build the game with as many features as we can implement and learn something new in this regard.

Difficulties that we faced in the journey of the project:

1. As the SDL library was quite new for us, we faced most of the difficulties in the graphical part of the project.

2. Some features like the timer, buttons were a little bit hard to implement.

3. Another big difficulty was to debug the code. As the code was of thousands of lines, sometimes it became very difficult to debug even a simple error in the code.

Future plan:

At first we will try to cover up the limitations of the project. We will also try to add some more features in the game if possible. If we become successful to solve the limitations, then we may host the game on a platform and keep maintaining the game in future.

Repositories:

Github repository:

https://github.com/imran-rahman543/Words

Youtube video:

https://youtu.be/OuePEQ4xwLw

References:

https://youtube.com/playlist?list=PLhfAbcv9cehhkG7ZQK0nfIGJC_C-wSLrx

https://lazyfoo.net/tutorials/SDL/