CS154 ABSTRACTIONS AND PARADIGMS PROJECT

PROJECT : WORD GRID

TEAM MEMBERS:
Ch.Sai Charith(160050083)
G.Nikhil Samrat(160050088)
T.Nikhil(160050096)

Description of the Project:

1)TO A GENERATE A WORD-GRID PUZZLE:
A random grid puzzle, consisting of hidden words is to be generated along with the words printed sideways, so that the player solves the puzzle by finding all the words.

2)TO CREATE A WORD-GRID PUZZLE, WITH GIVEN SET OF WORDS:

Given a set of words as an input, an N x N grid consisting of random letters is generated with the given words hidden in it, and the words printed sideways.

Design Outline:

1)**Forming word-lists** :-

-A super-list consisting of 10000 dictionary words, is provided to the main program.

-Horizontal, Vertical, Diagonal-down, Diagonal-up word-lists are created from the super-list, and the number of words in each list varies with the difficulty level(1-8) of the game.

-In case of creating a word-grid with given words, super-list is replaced with the given list of words.

2)**Grid generation** :-

-Grid is formed using a 2d-vector nxn.

-First, all the place-able positions, based on length of the word, are identified.

-Out of the possible positions, one is chosen and further checked that, the word does not intersect with any word in a wrong letter and also doesn't overlap(be a part of) with any word.[Choosing this one position may require more than one and many random checks,and here we used a little **memoization** so that two random checks are not repeated].

-Same orientation words are placed all at once.

-This way all the words are set into the grid by passing the generated word-lists, into the function.

-Also, for the sake of user-interface another two 2d-vectors are generated,

(i)An extra nxn vector is created, to mark positions of the initial and final letters of all the words. (with a characteristic-number (range : 0-n), that has been allotted to each word)

(ii)A nx3(word-details) vector is created, which stores the word, starting-letter coordinates and last-letter coordinates. (index is set equal to the characteristic number of the word)

-Brief Idea of extra vectors :-

(i)When the player finds the word in the grid and clicks on the end letters of the word, the user-interface program checks for the presence of characteristic number in the end-letters(both) and shades the identified word.

(ii)Using the nx3 grid, words are printed beside the grid and also with the help of stored end-letter coordinates, solved grid could be displayed at will, with a keyboard-command.

-In this procedure of grid-forming, after setting the grid with certain number of words, there will be cases of no-place for next word in the grid, after this identification the program automatically resets the grid and starts forming the grid freshly.

- With this procedure, there are just good number of chances that the grid will be formed, in a reasonable time. To claim that the grid cannot be formed is a tedious task(which is not the interest).

3)**Display of grid**

-Letters in the grid are displayed from the 2d-vector grid on the canvas. Besides that all the hidden words are displayed to the left of the grid in two columns.

4)**Adding text-fields for input words and game-levels**

-User has to choose between two options:

i)*To solve a computer-generated grid* : For this user has to choose "computer-generated-grid" button, after typing a difficulty-level(1-8) in the text-field called "level".

ii)*To make his own grid* : For this user has to type in all the words(separated by non-alphabetical character) into the text field and then press "My-grid" button. Then grid is generated by replacing the super-list by this list of words.

5)**User-Interface with the grid**

-All the mouse events within the grid are captured. When user selects a word(by clicking on the ends of word) it gets highlighted in yellow.Now the program looks into the 2d-vector called word-check and if same number is present at both locations then selected word is considered hidden and it turns green and the word in the list is striked off.

-Also when user types in "solve" or "erase" entire solution is displayed or clears the solved part respectively.

-The method solve uses the third vector "word-details" from which positions of each word in the grid is retrieved and the solution is displayed.

Abstractions & Higher order functions:

-A similarity is made among placing the words in vertical,horizontal and diagonal orientations,by modifying the 2d-vector definitions.

Eg :Setting "sym" as V' makes the grid-set function to place all the words in "wlist" in Vertical orientation.

-General Higher- order functions like map, foldr and iterators are used extensively.

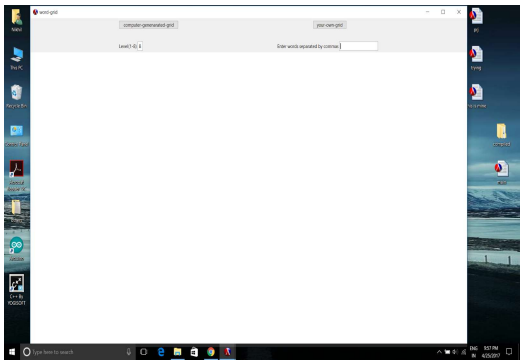
Limitations:

-May take a lot of time if the number of possibilities of forming grids are too less.

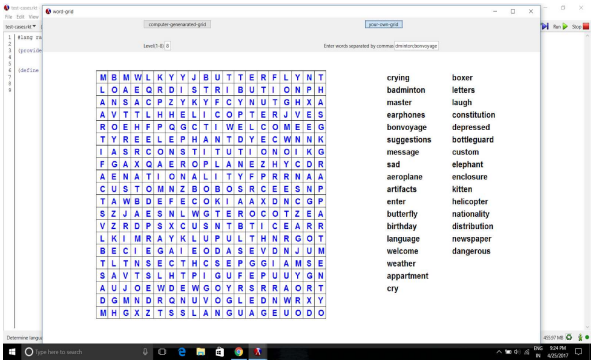
-Non-existence of a grid cannot be determined by this method.

Sample Input and Output:

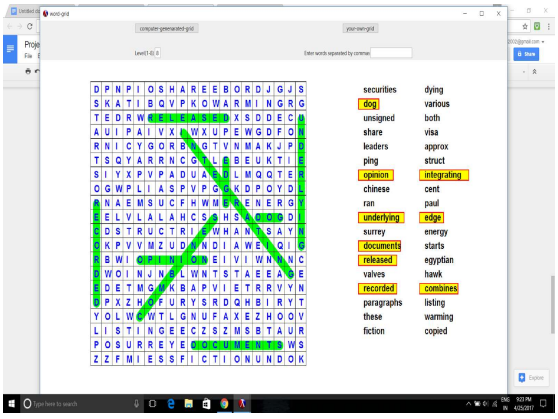
After running the program



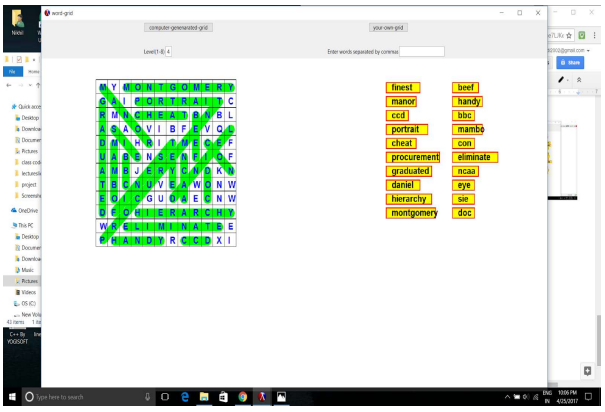
when user inputs in text-field



solving grid



complete solution also different level



Interesting aspects:

1) The system of grid coordinates (gvectors) helped us to place words in four orientations using a single function instead of four functions.
2) Use of memoization to check already placed words while forming grid.

Finally, our sincere thanks to Prof. Amitabha Sanyal for making the course enjoying and fun to do.

Also our thanks to all TA 's for their help and support.