SIT323 Practical Software Development, Trimester 2, 2016

Assessment Task 1 - Crozzle Validation

Due Date

Monday 9:00 am, August 29, 2016

Introduction

Assessment Tasks 1 and 2 comprise parts of the one project that must be developed using C# and MS Visual Studio. These require you to design and develop software related to the word game called crozzle (see class notes of week 1). In brief:

1. the first assessment task focuses on:
   - loading data from files related to a completed crozzle,
   - validating/invalidating the input data files and crozzle,
   - computing the score of a crozzle,
   - displaying that crozzle and its score,
   - coding to conventions and standards, and
   - implementing menus, file dialogs, web browser(s) and regular expressions;

2. the second assessment task focuses on:
   - loading data files related to creating a new crozzle such that its score is valid and is the highest possible that you can achieve,
   - ensuring that this new crozzle is valid,
   - saving this new crozzle to a file,
   - optimisation techniques,
   - appropriate scoping of code elements,
   - method characteristics such as cohesion, coupling, pre and post conditions,
   - the usage of return statements, and
   - implementing exception handling

Software Requirements

Your software solution for Assessment Task 1:

1. requires a GUI interface. As part of your GUI, you must use the WebBrowser control to display program generated data such as the crozzle, crozzle's score, and errors detected during validation. For example, it's possible to transform that program generated data to HTML, then use this WebBrowser to render this HTML.
2. requires the following data from a crozzle TXT file to be read by your program during runtime (see "Test 1 – crozzle.txt" as an example):
   a) The first line of this file contains the file header consisting of the following six kinds of values, which are separated by commas:
      i. a difficulty level: either EASY, MEDIUM or HARD as the only valid values
      ii. a positive integer representing the actual number of words in the list of words: the number of words in a list ranges from 10 to 1000, inclusive
      iii. a positive integer representing the actual number of rows in the crozzle: this number is within 4 to 400, inclusive
      iv. a positive integer representing the actual number of columns in the crozzle: this number is within 8 to 800, inclusive

v. a non-negative integer representing the number of horizontally oriented words within the crozzle
vi. a non-negative integer representing the number of vertically oriented words within the crozzle

b) The second line of this file contains a list of words, which are separated by commas. This list does not contain duplicates.
c) Data associated with each word of the crozzle is placed immediately after this list of words. For each word displayed in a crozzle (as depicted in Figure 1 and derived from the file "Test 1 – crozzle.txt") there is a line of data that includes the following four kinds of values, which are separated by commas:
   - word orientation: either HORIZONTAL or VERTICAL as the only valid orientation values,
   - the row location of the first letter of that word: 1 is used to represent the first row,
   - the column location of the first letter of that word: 1 is used to represent the first column, and
   - the actual word

	R	O	B	E	R	T								
			I					O	S	C	A	R		W
	J	I	L	L						H				E
	E		L					H		A				N
	S							A		R				D
	S		M	A	R	Y		R		L	A	R	R	Y
	I		A		O			R		E				
	C		R		G	A	R	Y		S				
J	A	C	K		E									
					R									

Figure 1.

3. requires the following data from a configuration TXT file to be read by your program during runtime (see "Configuration EASY.txt" as an example):
   a) The 1st line of a configuration file specifies the maximum number of disconnected word groups within the crozzle. A value of 1 indicates that all words in the crozzle are within 1 connected group of words. For example, the crozzle in Figure 1 shows 2 disconnected groups of words; one group is shaded in yellow, the other group is shaded in orange.
   b) The 2nd line specifies the points that can be scored for each word in the crozzle.
   c) The next 26 lines specifies the number of points that can be scored for each intersecting letter. For example, the letters M, R, L and Y are the intersecting letters on the 6th row in Figure 1.
   d) The next 26 lines specifies the number of points that can be scored for each non-intersecting letter. For example, the letters S, A, Y, R, A, R and R are the non-intersecting letters on the 6th row in Figure 1.
4. requires the TXT input data files to be validated. Aspects of these files which are found to be invalid must be written to a log file and displayed on your GUI.
5. requires the crozzle to be validated, only if the input data files are valid. Aspects of this crozzle that are invalid must be written to a log file and displayed on your GUI.
6. requires the valid or invalid crozzle to be displayed on your GUI.
7. requires the score of a valid crozzle to be computed and displayed on your GUI, but display 0 for an invalid crozzle.
8. requires constraints to be considered when validating a crozzle and determining a crozzle score (see below). There are constraints for 3 difficulty levels (EASY, MEDIUM and HARD); some constraints are common to all levels.

Constraints

Constraints common to all crozzles

1. Each sequence of two or more horizontal (vertical) characters delimited by spaces or the crozzle edge must form a word that can be found in the wordlist.
2. A word cannot be inserted in the crozzle more that once.
3. A horizontal word can only run from left to right.
4. A vertical word can only run from high to low.
5. Diagonal sequences of characters, which can be formed by horizontal and vertical words, are not pertinent to scoring or validity.

Constraints for the EASY difficulty level (in addition to common constraints)

1. A horizontal word is limited to intersecting at least 1 and at most 2 other vertical words.
2. A vertical word is limited to intersecting at least 1 and at most 2 other horizontal words.
3. A horizontal word cannot touch any other horizontal word. That is, there must be at least one grid space between a horizontal word and any other horizontal word. For example, no letter from any other horizontal word can be placed into the yellow region in the following diagram.
4. A vertical word cannot touch any other vertical word. That is, there must be at least one grid space between a vertical word and any other vertical word. For example, no letter from any other vertical word can be placed into the green region in the diagram of Figure 2.

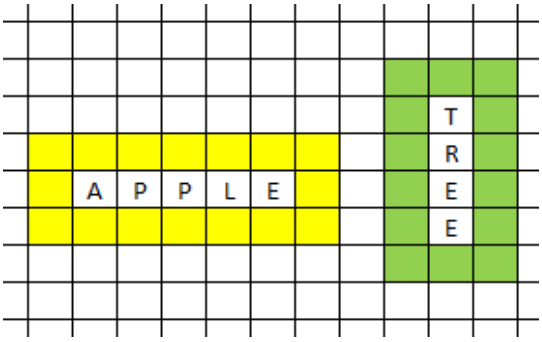

Figure 2.

Constraints for the MEDIUM difficulty level (in addition to those common constraints)

1. A horizontal word is limited to intersecting at least 1 and at most 3 other vertical words.
2. A vertical word is limited to intersecting at least 1 and at most 3 other horizontal words.
3. A horizontal word can touch another horizontal word, and a vertical word can touch another vertical word. However, you must adhere to the 1st common constraint.

Constraints for the HARD difficulty level (in addition to those common constraints)

1. A horizontal word must intersect 1 or more vertical words.
2. A vertical word must intersect 1 or more horizontal words.