CS 3202 – Spring 2014
Programming Assignment 5 - 25 points
Due date: Friday, April 18th at 9a

Learning objectives

- Demonstrate the ability to use C++/CLI to create a .NET Windows Form application.
- Demonstrate the ability to apply good software design and development practices in the development of an application.
- Select appropriate data structures and algorithms.
- Use a version control system to manage a software project.

Overview

This assignment will implement a variation of a Text Twist game as a standalone GUI application created as a C++/CLI .NET GUI application. You may want to visit the following sites to play a variation of this game.

- https://games.yahoo.com/game/text-twist-2-flash.html
- https://games.yahoo.com/game/word-scramble-ii-flash.html
- http://www.wordplays.com/wordgames/text-twist

Getting started

If your team consists of two members then only one person from your team needs to do the following:

1. Create a private Mercurial repository within Bitbucket.

    a. Name your project and the repository: Team*X*TextTwist, where *X* is your team number.

    b. Set the language to be C++.

    c. Add a wiki and issue tracking for this repository.

    d. If applicable, share the repository with your team member and grant him/her appropriate access privileges.

    e. Share the repository with dyoder@westga.edu and just grant me read access only.

2. Create a C++ Windows Form application using Visual Studio 2012. You may not use Visual Studio 2010 to write this application. Name the project Team*X*TextTwist.

   a. The title displayed in the form must be Text Twist by *Team member name(s).*

   b. Create a local repository and then sync the initial project to Bitbucket.

      i. Make sure you edit the .hgignore file to ignore all the Visual Studio created files that do not need to be included in version control.

Requirements

1. The GUI program must have the following functionality:

   a. Display seven random letters from which the words will be formed. The letter set frequency from which the random letters are chosen are as follows:

```
11 e
 9 t
 8 o
 6 a,i,n,s
 5 h,r
 4 l
 3 d,u,w,y
 2 b,c,f,g,m,p,v
 1 j,k,q,x,z
```

   b. Allow the user to “enter” words. The words entered must meet the following criteria:

      i. The words must be formed from the random letters that are displayed.

      ii. Each letter can only be used once.

      iii. Once the user has entered a word, there must be a way for the user to submit the word so it can be evaluated to see if it is valid word.

         1. A player should not be able to submit a word if it is not at least three letters.

iv. Words entered to be checked for validity and if they are valid words then scored appropriately.

Word length	**Points**
3, 4	1
5	2
6	3
7	5
8+	11

1. A word is valid only if it is found in the provided dictionary, dictionary.txt with Moodle.

2. If an invalid word is entered, the score should be reduced by one (1) point.

v. Display all the valid words entered by the player and their score and the overall score.

c. Display a one (1) minute count-down timer in which the player can find words.

d. Allow the ability to clear a single letter or all the letters the player was entering for a word.

e. Allow the ability to start a new game and exit the application.

f. A high score board system to store the name and high scores.

i. The score board should be persistent through different invocations of the game and have the ability to be reset.

ii. Allow the ability to view the high score board.

g. Allow the following customizations to the game:

i. Set timer to be one, two, or three minutes.

1. The high score board should indicate time allotted.

ii. Toggle the ability to reuse the random letters.

h. Note: the implementation does not need to indicate the number of words that are available as is done in many online versions. However, this could be a WOW factor addition.

Bitbucket and version control requirements

1. There must be at least 10 commits of key contributions or milestones in the project. If your team consists of two members, each team member must make at last five commits.

2. For teams of two, within the repository there must be at least two merges of branches within the repository.

3. In the Wiki do the following:

    a. Provide a short summary of the project.

    b. Provide a screen capture of your game.

    c. Provide a description of how the list of valid words was stored and searched, i.e., describe the data structure(s) and algorithms used.

    d. Provide a description of any WOW factor items.

    e. Provide a list of any known bugs or missing functionality. If there are not any known bugs or missing functionality indicate so.

4. In Issue Tracking do the following:

    a. Setup issues for varies parts of the functionality that need to be added to the project.

    b. For teams of two, assign specific issues to each team member.

    c. Resolve issues within Issue Tracking as they are addressed.

5. Once you have completed the assignment tag the completed assignment changeset with the following label Assignment 5 submission.

Submission

In Moodle, for teams of two, each team member must specify an estimate of the percentage of work each team member contributed, e.g., John – 55%, Sally – 45%. Additionally, please specify the parts of the project you worked on, in particular, what items did you do the majority of the work. If you are worked on this individually, then you do not need to do specify this information within Moodle.

One member of the team must do the following:

1. Remember to tag the completed version as described above.

2. Within Visual Studio select Build → Clean solution to remove all files that can be regenerated upon building the project.

3. Close Visual Studio.

4. Navigate to the root folder for the project and zip the root folder and verify the zip file is name *TeamX*TextTwist.zip and submit the zip file to Moodle by the due date.

Note: I will actually clone the project from Bitbucket, but I want to make sure I have a version of the project.

Grading breakdown

Any program that does not compile will receive a 0. Partial credit is not possible for any program that does not compile.

All grading will be itemized for this assignment and be as follows:

- 12 pts. – application works correctly and meets basic requirements.
- 3 pts. – Clean code – this includes design and implementation of classes and methods; names of identifiers; well-written classes and methods, appropriate documentation, separation of concerns, etc.
- 2 pts. – Type of data structure and algorithm(s) used for storage and searching of the dictionary. This will include overall efficiency of the execution of the program during game play.
- 3 pts. – Design and usability of the GUI.
- 3 pts. – WOW factor – reward for special creativity, functionality, and/or thoroughness in doing the project.
- 2 pts. – Use of version control, wiki, and issue tracking and group contribution.
    - For teams with two members, the group contribution evaluation will be based on commits in Bitbucket and evaluation from the other team member.
        - **If you are on a team of two, and individually you do not contribute any work on this project, you will receive a 0 on this assignment regardless of what the rest of your team receives.**