COMP 3300 – Fall 2024
Project 2, 100 points
Due: Sunday, Nov 24th 11:55PM

Learning Objectives

- Apply development principles learned in this course and prerequisite courses to implement a small software application in C#.

Items to Note

- Since this is a project that applies what has been covered in this course semester as well as in the prerequisite courses, specific implementation details will not be provided as the goal of a project is to demonstrate that you can apply what you have previously learned.
- If there are concepts that have not been covered in C#, but were covered in the prerequisite courses, you may need to leverage the Microsoft documentation to find how to implement it within C#.
- You are free to use any elements we covered this term – every topic may not be utilized, but everything we covered this term is available to pull from when completing this project.
- The entire project must be done individually without consultation with any other individual other than the instructor. If you have a question regarding this project, please feel free to contact the instructor. Keep in mind, only high-level clarification questions will be answered, no implementation questions or inquiries if the code is ok.

Tips

- <u>Don't wait till last minute to start.</u> You will have to decide what route to take with some of these steps, some methods make more sense (and are easier to implement) than others. Since this is the last project of term and the culmination of the material we've learned, there will be more emphasis put on getting things correct, especially the little things.
- All best practices we've covered this term are applicable here – comments/git/etc. Not properly implementing these are an easy way to lose points.
- Along the same lines, you should be writing code defensively, meaning expect your user to do something to break things. Even if not explicitly listed, you should be implementing validation checks on elements like parameters/etc. Carefully consider what the code section is trying to do and pick checks that seem to make sense. For example – if the user can enter letters, should the also be allowed to enter numbers? Does it make sense within the context of the code/what's happening to allow numbers?

Overview

This assignment will implement a variation of a Text Twist game as a standalone GUI application created as a C# Win Forms (Windows Forms) application.

Visit the following site to play a variation of this game:

- http://www.wordplays.com/wordgames/text-twist
- You can use this example to extract a general idea of how your version should work.

Getting started

Read through all the requirements before you begin. **Any submitted project that does not compile will receive a zero.**

1. Start Visual Studio and create a C# Windows Forms App with the name: `*FirstnameLastname*Project2`.

2. Change the form title bar to be `Text Twist by *Lastname*`.

Requirements

The following requirements will not be as detailed as what we've covered to this point. The last half of the class we've mostly been working toward this, so it should not be a significant change, but it is one that you should keep in mind.

1. The 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
    ```

    Suggestions: All the letters for the game can be put into a bag (list) and then seven random letters can be drawn. The `Random` class can be used to get a random number.

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

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

ii. Each displayed letter can only be used once.

   1. It should be clear to the user that a letter was already used this attempt ***hint* Remember controls have properties that can be utilized to convey different states/or behaviors.**

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.

   2. When the word is entered it must indicate if it was a valid word or not and the points earned for a valid word.

iv. Words entered are to be checked for validity and if they are valid words then scored appropriately. The scoring mechanism is given below.

Word length	**Points**
3	90 (30 pts. per letter)
4	160 (40 pts. per letter)
5	250 (50 pts. per letter)
6	360 (60 pts. per letter)
7	490 (70 pts. per letter)

   1. A word is valid only if it is found in the provided dictionary, *dictionary.json* (found in Moodle)

   2. A valid word entered two or more times is not scored.

v. Words entered by the user should be tracked.

   1. Valid words entered should be stored in such a way that the word entered, number of points earned for the word, and time the word was entered, game time, can easily be discerned (more info on time is later in these instructions).

   2. Invalid words need the word, time (game time), and a reason it was not valid (i.e., not found in dictionary, etc.) to be tracked.

hint Consider carefully what is being asked to be tracked here, are there some common items between the two and is there some method we covered that could be applied in this situation?

3. Each “round” should be tracked. Meaning if the user played 4 “rounds” then there should be details stored for all 4 of those rounds (both valid and invalid words).

vi. At the end of the round, display all the valid words entered by the player and their score and the overall score for the round just played.

c. Display a count-down timer in which the player can find words. This represents the “game time”.

   i. Via a menu selection allow the user to set the timer to be one (60 seconds), two (120 seconds), or three (180 seconds) minutes.

   ii. The default selection should be one minute.

d. Allow the user to be able to twist the letters during the game. Twisting is taking the existing letters for the current game and displaying them in a different order.

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

f. Add functionality to display a high score board system to show the name, score, and time for the game.

   i. The high score board should be persistent between various invocations of the game. This is inclusive of closing out the application and then starting it again.

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

   iii. The user should be able to sort by overall score and a two-level sort which is by time and score.

   iv. Allow the ability to reset the high score board.

g. Add functionality that will allow the user the ability to “export” their stats.

   i. If there have been no rounds played previously (i.e., just starting the game for the first time) this ability should not be available.
   ii. The export should produce a file that contains <u>all the details</u> for both valid and invalid words, for every attempt.
   iii. You may choose the file types from any of the methods we reviewed this term. CSV, JSON, TXT.

This concludes the main portion of the project. Even though there are not a lot of steps listed here, there really are several things that must be sorted out and coded within them. As mentioned before, do not wait until last minute to start this. There are a lot of little details that will arise as you work through this project.

Optional: Extra Credit

This portion is completely optional. You are not required to complete it, but if you choose to do so then there will be extra points at the end of it, should you complete it successfully. The conditions are as follows:

- You must have completed the main portion of the project. If there are major portions missing of the main project requirements, the extra credit will not be counted.
- <u>If you attempt this section, there should be a .txt file included within your submission that states that you are attempting it. If there is no .txt file included, the extra credit will not be counted, even if completed.</u>
- This section is more of a challenge, so the instructions will be extremely basic/vague in addition to the extra credit is all or nothing. There will be no partial points for extra credit.
- **If you complete this extra credit successfully (following best practices we covered this term), your lowest assignment or quiz grade will be replaced with a 100. Note this is for a single assignment OR quiz grade and it does NOT extend to a project or exam grade. Should you complete this, do not need to message me or note what grade to replace, I will simply pick the lowest grade and apply it.**

1. Implement a method to allow a user to "continue" adding to their stats from a previous session. Or in other words, allow a user to load in a previous session. Here are the requirements:

    a. You are free to choose how to do this, although it must involve the data that was exported previously from the program.
    b. In addition, "continuing" like this will also set the countdown timer to the one last set by the user.
        - For example, if a user exported and the time was 120 sec, when "continuing" the countdown timer would also be set to 120 sec.
    c. If something goes wrong with the loading, inform the user, and treat the game as if it was a brand new one. In this case the default timer would be 1 min.

Submission

Make sure you clean your solution (Build ➔ Clean Solution) and then zip up your project which needs to include the solution file into a ZIP file called *YourName*Project2.zip and submit the assignment by the due date.

Grading breakdown

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

- 70 pts. – Application works correctly and meets requirements.
- 15 pts. – Implementation – this includes design and implementation of classes and methods; names of identifiers; well-written classes and methods, appropriate documentation, separation of concerns, git, etc.
- 15 pts. – Design and usability of the GUI.