CIS 227 Assignment 11

Assignment Details

Design your own extensions and function for Hangman

Use QDIR to set paths for files

Open a UI form to send an email to developers

* must use QDesktopServices

Special note

* I will be looking for proper data type usage
* Complete documentation of new functionality
* Memory Leak minimization

Team Roles

Lead Programmer – Caleb M

UX/UI Programmer –Caleb B

Functional Programmer – Amanuel F

Program – 200

UX/UI – 100

Function - 100

Documentation – 100

Total Possible Points – 300

**Version 0.0.0**

| REVISION HISTORY | | | |
| --- | --- | --- | --- |
| DATE | VERSION | DESCRIPTION | AUTHOR |
| 1/31/2021 | 1.0.0 | Original program for array printer | Group 3 |
| 2/8/2021 | 1.1.0 | Added utility for reading from text file | Group 3 |
| 3/8/2021 | 2.0.0 | Converted to guessing game with feedback/score | Group 3 |
| 3/17/2021 | 3.0.0 | Integrated QT GUI | Group 3 |
| 3/23/2021 | 3.1.0 | Splash screen/Menu/Gameplay update | Group 3 |
| 3/29/2021 | 3.2.0 | Visual Upgrade and score tracking | Group 3 |
| 4/6/2021 | 4.0.0 | Difficulty levels and multi-window functionality (High scores and quotes added) | Group 3 |
| 4/25/2021 | 5.0.0 | High Score importing (QDIR path) | Group 3 |

# INTRODUCTION

## PURPOSE

Identify and describe scope of product whose technical specifications are being documented and describe desired outcome.

1. Display splash screen with coding quote
2. Log user identity
3. Get word from array
4. Create game around guessing the identity of the word
5. Output win/loss guessing (UI implementation)
6. Instance games for user change
7. Options for displaying high score and quote list
8. Ability to import high scores from separate file directories

## DOCUMENT CONVENTIONS

Describe any naming or structural conventions employed throughout document and how they benefit reader.

* Consistent use of camelCase for variable naming conventions
* Assignment 2 class handles rules for user input and managing string population
* GameHandler class handles the progress of the game and its rules
* Point system governed by correct/incorrect guesses and usage of hints
* Menu options for editing the gamestate
* Each stage of the program is interacted with on its own page
* Error handling to prevent crash in case of .txt file deletion

## REFERENCES

List any referenced document names or links.

# DESCRIPTION

## FEATURES

List main features with brief description.

1. Splash Screen – Devs, version, quote
2. User ID – Request user input through text box and then set them through button click event
3. Array – Random word is chosen for use throughout the instance of the game
4. Word guessing game – After word is generated from array, the submit guess button will compare it to that string
5. Point System – Correct choices reward points, incorrect choices deduct points. Invoking the hint option deducts points.
6. Win/Lose – The game will terminate when word is correctly guessed or to many letters have been guessed incorrectly
7. UI Display – Points and timer based on current instance of the game
8. Menu Options – Options given for handling the game and its outputs
9. High Score – Show high scores and the player responsible for them
10. Quote List – Displays quotes and gives the options to add new ones
11. Customizable – Use a QDir to find alternate folder specified by user

## USER OVERVIEW

Define groups and describe user characteristics.

## ASSUMPTIONS / DEPENDENCIES

Detail all assumed factors (not known facts) that could potentially impact technical specifications set forth. Include external factors.

* .txt files should be stored in the build folder

# SYSTEM FEATURES

## SYSTEM FEATURE 1

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Hold a string from array[x] for comparison to letter guess |
| **STIMULUS / RESPONSE SEQUENCES** | Form Load |
| **FUNCTIONAL REQUIREMENTS** | iSecret = rand() % 12 + 1;  gamer.hide(true, *iSecret*); |

## SYSTEM FEATURE 2

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Set initials for the user playing the current game. |
| **STIMULUS / RESPONSE SEQUENCES** | Inputs must be obtained from user |
| **FUNCTIONAL REQUIREMENTS** | string init = this->ui->InitialsTextBox->toPlainText().toStdString();  gamer.setInitials(*init*); |

## SYSTEM FEATURE 3

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Search string array for letter and assign points based on its availability |
| **STIMULUS / RESPONSE SEQUENCES** | Compare user input to string |
| **FUNCTIONAL REQUIREMENTS** | bankStorage = wordArray[n - 1];  for (int i = 0; i < bankStorage.length(); i++) {  guessBank.push\_back(bankStorage[i]);  (find(guessBank.begin(), guessBank.end(), guess[0])!= guessBank.end())  vector<char> :: iterator gFinder = find(guessBank.begin(), guessBank.end(), guess[0]);  if (gFinder != guessBank.end()) guessBank.erase(gFinder); |

## SYSTEM FEATURE 4

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Control gamestate through menu commands |
| **STIMULUS / RESPONSE SEQUENCES** | Menu click events |
| **FUNCTIONAL REQUIREMENTS** | this->ui->GuessBox->clear();  this->ui->InitialsTextBox->clear();  this->ui->CurrentInitials->clear();  this->ui->WordIs->clear();  this->ui->scoreLabel->clear();  this->ui->GameWinOrLose->clear();  iValue = 0;  gamer.setCurrentScore(0);  gamer.clearBank();  gamer.clearScore(); |
| **DESCRIPTION AND PRIORITY** | Open File for reading writing for scores and quotes text file |
| **STIMULUS / RESPONSE SEQUENCES** | Menu click events |
| **FUNCTIONAL REQUIREMENTS** | QFile file("quotes.txt");  if (file.open(QIODevice::ReadOnly | QIODevice::Text)){  QTextStream stream(&file);  while (!stream.atEnd()){  strings += stream.readLine();  }  }  file.close(); |

## SYSTEM FEATURE 6

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Find substitute text file to open for high scores |
| **STIMULUS / RESPONSE SEQUENCES** | Text Entry by user |
| **FUNCTIONAL REQUIREMENTS** | qFileLocation = this->ui->fileLocationEdit->text();  sFileLocation = qFileLocation.toStdString();  for (char x : sFileLocation) {  if (x == '\\') {  correctedLocation.append("/");  }  else {  correctedLocation += x;  }  } |

# REQUIREMENTS OF EXTERNAL INTERFACE

## USER INTERFACES

Describe product / user interface characteristics, including standards, style guides, constraints, functionality, and sample screens if applicable.

# APPENDICES

## APPENDIX A: GLOSSARY OF TERMS

Define all terms and unique acronyms employed throughout document and specific to project.

## APPENDIX B: ANALYSIS DOCUMENTATION

List file / document names / provided links to all diagrams, models, additional findings pertinent to technical specification development.

## APPENDIX C: ISSUES

List all unresolved issues, TBDs, pending decisions, findings required, conflicts, etc.

| ISSUES | | |
| --- | --- | --- |
| ID | DESCRIPTION | PARTY RESPONSIBLE |
| 01 | Once guess is submitted, program loops through without accepting new input **(RESOLVED)** | Group 3 |
| 02 | Output for Win/Lose currently in test phase **(RESOLVED)** | Group 3 |
| 03 | Implementing the external text reader causes an undiagnosed crash, no output error **(RESOLVED)** | Group 3 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |