CO19-320212 Software Engineering
Spring 2019

Project Write-up

Introduction

Welcome to the Pencil Producer! You are now in charge of creating a system that would efficiently produce and sell pencils, with the ultimate goal of making sure everyone on the planet (or the galaxy) has enough pencils forever.

The end product of the project will be a program similar to the Universal Paperclip game (http://www.decisionproblem.com/paperclips/index2.html); with a visual interface implemented using Qt, a widely used cross-platform application framework and widget toolkit.

The project will be organised as a series of "sprints". The exact duration of each sprint is indicated in the sprint write-up. At the beginning of every sprint, you will be paired up with a new partner to work on the code base which will be assigned anew as well. You will not be given detailed instructions on what to do in each sprint, this is up to the team.

Grading of the progress in each sprint is based on any advances you make, and not only the feature implementation. You earn points for adding/improving features, adding/improving tests, adding/improving documentation, or even working on the build process. In line with the definition of "software" discussed in class, all these aspects contribute to the quality of the final software product; and would hence give you points.

As part of your learning process, you are expected to adopt the XP development process to complete the sprints. You will learn more about the XP development process in class.

Tools

The project will be implemented in C++ and Qt, supported by the NetBeans Integrated Development Environment (IDE), and CMake. For automated generation of code documentation, we will use Doxygen.

You need to install several tools constituting your programming environment, based on freely available components; frequently you can choose between installing the tools by source code for own compilation or prefabricated packages for installation through a package manager.

The following packages are required:

- GNU C++: https://gcc.gnu.org/install/
- Qt: https://www.qt.io/download
- NetBeans: https://netbeans.org/downloads/8.2/ (be sure to use version 8.x as version 9 and upwards no longer support C++)
- CMake: https://cmake.org/install
- Doxygen: https://sourceforge.net/projects/doxygen/

The operating system you use to code can be Linux or Windows. You may use other operating systems on your own responsibility and without your TAs' support.

For version control and code submission your TAs will set up the repository for your team to use on GitHub. You will be notified of this via email.

Primers:

- Qt: https://doc.qt.io/qt-5/qtexamplesandtutorials.html
- NetBeans: https://netbeans.apache.org/help/getting-started.html
- CMake: https://cmake.org/cmake-tutorial
- Doxygen: http://www.doxygen.nl/manual/starting.html

Testing

For unit testing, you may write your own framework (easier to start with, but could quickly become tedious as it does not scale and erodes quickly), QtTest (advantage: integrates with Qt), or Google Test (Google Test primer: https://github.com/janoszen/clion-project-stub/blob/master/gtests/googletest/docs/Primer.md).

Academic integrity

As always, academic integrity is critical throughout this project. As a rule of thumb, you must not copy-and-paste any code snippet that is not yours (apart from those in your repository).

If you seek help online or from a friend, and find a relevant code snippet you would like to use in your code, you must: understand the code snippet, close the code snippet, and type your code out yourself.

References file

Cite all the help you get in a references file called references.txt. This document must be included in your submissions in the root folder. Append to it as the project progresses.

Describe the contribution of each source as a JSON parseable text. Here is a sample format for the references file:

```
{
	"online-references": [{
			"url": "https://www.geeksforgeeks.org/bubble-sort/",
			"contribution": "Implementation for bubbleSort()",
			"file": "main.cc"
		},
		{
			"title": "https://www.geeksforgeeks.org/quick-sort",
			"contribution": "Implementation for quickSort()",
			"file": "main.cc"
		}
	],
	"colleague": [{
			"jacobs-id": "f.mcfreshface",
			"contribution": "Helped with bubbleSort() algorithm",
			"file": "main.cc"
		},
		{
			"jacobs-id": "f.mcfreshface",
			"contribution": "Helped with quickSort() algorithm",
			"file": "main.cc"
		}
```

```
        ]
}
```

You may use https://jsonlint.com to help format and validate your references file.

Your TAs will perform automated plagiarism checks on the code.

Do not make your code public

You must not make any code base used in the project available to the public (e.g. store it as a public GitHub repository). This is because the code base you work with has contributions made by multiple students; it is not owned by you alone.

Submission

The deadline for each sprint is indicated in the writeup. Your TAs will consider the last commit before the deadline when they retrieve from the repository.

You are encouraged to frequently push your code to the repository. This makes sure that you will have something close to your latest state available for grading even in case you miss the deadline for your last push.

You are also encouraged to make multiple submissions. Only your last submission will be looked into and used in the code review.

Important for space reasons: Do not include the build folder in your submission. Use a .gitignore file to achieve this.

Grading

Grading will be done after every sprint. It consists of two components, interview and code interview.

Normally a common grade will be issued. However, in cases where we feel it is more appropriate, grades may be individualized.

If for any reason you are unable to work effectively with your partner, write your TAs explaining the situation. Such a situation will be resolved accordingly, on a case-by-case basis.

Code interview

Your work will be assessed during oral code review sessions at the end of every sprint. Your TAs will organize the time and location of these sessions with your team via email. Each session will take approximately 10 minutes. You will have to demonstrate that you understand the work accomplished by your team during the sprint.

Both you and your partner are expected to attend these code reviews.

Code review

Your TAs will download your team's code from the repository after the deadline and will assess its quality. Assessment criteria are as follows:

Working Code

The code must compile, the compiled program must run (and do "something"), and all unit tests present must pass.

Properly formatted code is a grading criterion. Different styles are in use by programmers for formatting source code; your TAs will announce a C++ beautifier that may be used.

Incremental Improvements

This assesses the progress made towards the goal stated in the specification. Any and all improvements count — not just adding functionality. Among others, implementing these improvements would give you credits:

- adding functionality
- fixing bugs
- adding/improving documentation
- adding/improving tests
- code refactoring
- code optimization

Be sure to document the changes you implemented during the sprint, so that your TAs will not miss the improvements you have made.

Note that restarting from scratch or copying over from some other repository is not an option; it is always and invariably mandatory to work on the code base you get. If you find it really bad to work with, improve it!