CPSC 3720 Code::Blocks Plugin Project

1 Overview

In this project your team will create a unique plugin for Code::Blocks.

2 Objectives

Some of the objectives of this project are:

- Work collaboratively as a team to build a useful real-world product.
- Gain real-world experience by developing for a plugin framework.
- Follow an agile methodology to develop a software product.
- Use good software engineering practices to develop software.

3 Getting Started

A mostly configured Code::Blocks plugin (Example) project is provided as a starting point for you.

1. Open Code::Blocks
2. Open `Settings -> Global Variables`
3. Define the global variable `cb` as `/usr/local/codeblocks-17.12/src/`
4. Open the Code::Blocks project
    a. If you open the project first, it will ask you to define `cb`, but it seems to work better if this is first defined.
5. Build the project.
    a. There shouldn't be any errors. If there are, please file a bug report on the project repository.
6. Open `Plugins -> Manage Plugins`…
7. Click `Install New`
8. Open the `Example.cbplugin` file to install the example plugin.
    a. There shouldn't be any errors. If there are, please file a bug report on the project repository.
9. Confirm that the plugin is listed and close the window.
10. Select `Plugins -> Example`
11. Find the `Code::Blocks log` (should be at the bottom of the screen with the `Build Log` and `Build Messages`). You should see a message from the plugin at the bottom of the log.

4 Project

Use the example plugin as a starting point for building your own Code::Blocks plugin. The project will be completed in a number of stages:

4.1 PROJECT PROPOSAL

Submit a proposal for the Code::Blocks plugin that your team is planning to build. A proposal template is provided on Moodle for you.

a. Ideas for Code::Blocks plugins
    i. Refactoring plugin. Allows a user to perform refactorings such as renaming, push up, push down, and extract method.
    ii. Code coverage plugin. Configure and run lcov and display the results.
    iii. Build status. Display the results of continuous integration from a repository on the CS department's GitLab instance.
    iv. Version Control. Integrate a version control system into Code::Blocks.
        1. There are already a few plugins that do this, but they are generally no longer maintained or are not very good quality. Do a better job.
b. If you are choosing to do a plugin inspired by another plugin (e.g. GitBlocks), acknowledge this in your proposal and explain how your team's plugin will be different.

4.2 SPRINTS

Complete three 2-week sprints. Following each sprint you will:

1. Submit an updated project proposal with a section that has detailed answers to the following questions:
    a. What was your sprint plan?
    b. How close did you come to achieving your plan?
    c. What problems did you encounter and how did you solve them?
    d. What is your plan for the next sprint, if there is one?
    e. What will you do differently in the next sprint, if there is one?

    Each sprint will add a new section so that the final sprint report will be your complete project report.

2. Give a demonstration of your plugin to the class (max. 7 minutes) as of the end of the sprint.

5 GRADING

Your project will be graded based on:

- Sprint reports [each 20% of project grade]
    - Quality of writing.
    - Detail of answers to Scrum questions and retrospective.
- Sprint presentations [each 10% of project grade]
    - Quality of presentation.
    - Presenting a working plugin.
- A working and useful plugin by the end of the course [5% of the project grade]
- Evidence of the use of good software engineering practices (e.g. version control, unit testing, static analysis, continuous integration) throughout the project [5% of the project grade]

APPENDIX: PROJECT PROPOSAL

The project proposal document will contain the following:

1. **Title Page** showing:
    a. the name of the game,
    b. team name and logo,
    c. the team letter on Moodle
    d. names of team members (only list those that contributed to the design of the project and the report)
    e. due date
2. **Table of Contents.** It should be on its own page.

3. **Introduction**. The introduction provides an overview of the entire document. A person should be able to get a clear idea of what the project is about from this section. At minimum, describe what the plugin will do and why it is needed or someone would care about the plugin.

    The introduction should end with a preview of the major sections that follow.

4. **Proposal**. Describe the plugin that your team is proposing to build. Give as much detail as possible, including:

    a. Prototype images of the interface (can be hand drawn).
    b. Use cases (i.e. product backlog items).
    c. Configuration options, if applicable.

5. **Project Management.** Provide a description of, and address any foreseeable problems. The section must start with an introductory paragraph summarizing the contents of the section and previewing any subsections. This section will have two subsections:
    a. ***Team Organization:*** Describe how the team will organize themselves (including team structure) to create and support the plugin. Suggested team roles are:
        i. *Team Lead* (keeps everyone on track)
        ii. *Quality Assurance Lead* (makes sure application follows good OOAD principles and that a testing plan is being followed)
        iii. *Documentation Lead* (oversees the creation and maintenance of the project report/user manual.)

        All team members are expected to fill the roles of *Software Developer* and *Software Tester,* and to contribute to the project documentation.

b. ***Risk Management:*** Describe how the team will address foreseeable risks that could prevent the team from completing the project. Examples of risks include, but are not limited to:
    i. Requirements/Design/Estimation
        1. The team planned a project that is too large (i.e. “eyes bigger than stomach”).
        2. The team underestimated how long parts of the project would take.
        3. Major changes to design are needed during implementation.
    ii. People
        1. Addition or loss of team member (i.e. someone dropped the course, a new person joins the team)
        2. Unproductive team member(s)
        3. Team member(s) lacking expected background
        4. Illness or unanticipated life events (e.g. death of family member)
    iii. Learning & Tools
        1. Inexperience with new tools
        2. Learning curve for tools steeper than expected
        3. Tools don’t work together in an integrated way

6. **Development Process.** Describe the process that the team will follow in developing the software. Such items should include:
    a. Coding conventions (give small examples).
    b. Code review process (e.g. how are pull requests handled?).
    c. Communication tools/channels (e.g. **Slack (recommended)?** Email? Text? Issue tracking? Skype?).
    d. Change management (how will the team deal with feature requests/bug reports? Does the team lead triage reports, or does everyone?)

7. **Appendices.** Any figures or tables that are more than half of a page should be put in the appendices and reference in the report text. Omit this section if there is no content.