CSC290. Project Repository

The final deliverable for the CSC290 group project is the project repository. Ideally, this project repository will contain clean code and a well-written README that you are proud to show to potential employers. If done well, the repository shows not only your coding skills, but also your ability to work together in a team, and your ability to communicate with a technical audience.

Good repositories not only have clean code, but also a well-written README describing your project.

Here are some examples of projects (in the wild) with good READMEs:

- https://github.com/tasdikrahman/spaceShooter
- https://github.com/mbostock/polly-b-gone/wiki

Here is an example of a good CSC290 project README:

- https://github.com/kwpark23/Zeros_Matter

How to submit

Submit your repository link in a plain text file called `link.txt` on Markus. The submission will be **as a group**, so only one person from your group need to make the submission.

Please remember that grace tokens **cannot** be used for group work. No late work will be accepted. Markus submission time will be used, not your local computer time or any other screenshots that you provide. You can submit your work as many times as you want before the deadline, so please submit often and early.

Please do not make any changes to your repository after the deadline, until we are finished grading the projects.

Repository Requirements

We'll be looking at the following:

- Project README (30%)
- Individual Contribution (10%)
- Code Quality (20%)
- Commits History (20%)
- Grammar & Mechanics (20%)

Project README

The project README should have a set of instructions for how to install/play the game that is understandable by a general audience. The README should also have documentation that provides a high-level overview of your code, understandable by a computer science audience (e.g. your classmates).

The first part should contain instructions on how to install and run the game, and instructions on how to play your game. Include at least one screenshot.

You are also required to select a license for your repository. See https://help.github.com/en/articles/licensing-a-repository Your team has full authority over what license you choose, but you should choose one and communicate it clearly.

The documentation should describe your projects directory structure, and the major classes and methods. Your description should give readers an idea of where to look if they wanted to tweak a small thing about your game: for example if they wanted to change the game background colour.

To help others who might want to extend your code, provide at least one example of how to extend your game. That is, provide instructions on what to do if someone else wants to add an out-of-scope feature to your game.

Individual Contribution

In the README, include a section where each team member writes a one-paragraph summary describing their contribution to the project. This is like the addendum that you wrote for your project plan. These paragraph and their contents are graded individually, and **cannot** be written by anyone other than yourself.

The paragraph should contain a description of both your individual contribution to the *code*, and the contribution to the writing of the README.

Code Quality

We are looking for a repository that follows the style guidelines typical for the programming language that you choose, that is consistent, and that is easy to read and understand.

Specifically, we are looking to see if...

- Variable and function names are appropriate
- Function docstrings are descriptive and appropriate
- Code follows the standard guidelines for the language
- Functions are atomic and testable
- Code contains little/no duplication

See the "code commits" handout. We'll be looking for similar quality of code and comments.

Commit History

We will check your commit history, and are looking for commit messages that continue to follow the guidelines discussed in class and in the "code commits" handout. Like in the "code commits" assignment, we will follow a commonly used set of guidelines described here: https://chris.beams.io/posts/git-commit/

1. Separate the commit message subject from body with a blank line.
2. Limit the commit message subject line to 50 characters.
3. Capitalize the commit message subject line.
4. Do not end the commit message subject line with a period.
5. Use the imperative mood in the commit message subject line: i.e. describe what the commit message does (in present tense) if you apply the commit.
6. Wrap the commit message body at 72 characters.
7. Use the commit message body to explain what and why vs. how.

It is difficult to change a commit message after you have committed the code. Instead, you should think about the commit message as you are writing code, or even before writing your code. Think about the changes you want to make, then make those changes, and make your commit.

If you want to become a git superuser, you can look up "interactive rebase" and the `git commit --amend` functionality.

Grammar & Mechanics

We are looking for proper grammar and good sentence structure throughout your repository. This includes the repository README, the commit messages, and the comments in your code.