USING GITHUB

BY MATTHEW SUNTUP

FOR SRSSS AERO2711/AERO3711/AERO4711 PROJECT

SETTING UP LOCALLY

STEP 1 - DOWNLOAD GIT

Download the terminal Git from this link:
https://git-scm.com/downloads

Hint: Most of the options that it will ask during install really come down to personal preferences and won't affect overall function. If you're unsure, just leave options as default.

STEP 2 - AUTHENTICATE GIT

Open command prompt for Windows or terminal for Mac/Linux, and use the this command to configure your username:
>> git config --global user.name "Your Name Here"

Configure your email (this must match your GitHub email):
>> git config --global user.email "your_name@domain.com"

STEP 3 - CLONE THE REPO

In command prompt for Windows or terminal for Mac/Linux, navigate to the folder where you want to save your copy of the repo.

Clone the repo with this command:
>> git clone https://github.com/MatthewSuntup/SRSSS

Hint: If you haven't used command prompt/terminal in a while, use "dir" on windows or "ls" on mac/linux to list the directories in your current location. Use "cd <relative path>" to navigate deeper into a directory, and use "cd .." to go to the parent directory.

Step 4 - Branching

When you start out you'll be placed in master branch, but we want to be doing most of our work in the dev branch.

First, navigate inside the repo using the command:
>> cd SRSSS

Then switch branches using the command:
>> git checkout dev

Hint: You can use ">> git status" to check which branch you're currently on (as well as get other useful info about staging), and you can use ">> git branch" to list all available branches.

Using Git Version Control

Writing Files

When you make a new piece of code, you don't need to do anything different to normal, just save it somewhere inside your remote repository (this is the folder where you stored your GitHub repository).

IMPORTANT: When you add files to these folders, they're not automatically shared with anyone (and don't have to be), they'll only become a part of the repo if you choose to stage and commit them.

Committing Changes

Committing a change is the act of locking it in at a local level. This is a two-step process; staging and committing. For convenience, check git status before committing. It will highlight what is staged and what is not.

To stage a commit, navigate to the folder with the changed file(s) and enter:
>> git add <relativeFilePath.m>

At this point, you may want to check git status again to see you have staged the correct files.

To commit these changes, enter:
>> git commit -m "<A brief message about the changes>"

Note, at the moment these changes are only local.

Syncing - Push to Online Repository

In order to update the online repo we have to "push" these changes. To do this, enter:

>> git push origin <branch name>

NOTE: its important to use the correct branch name, so changes don't overwrite on a different branch. Generally, you'll always to to be pushing to the dev branch:

>> git push origin dev

Hint: You may get an error message at this point, likely because someone else has pushed a change to the repo since your last pull. Don't worry, Git is great at merging changes! Use a git pull (described below) before retrying your push.

Syncing - Pull from Online Repository

When using multiple machines often the repo on one machine will have code in a different state than the other. In order to bring a local branch up-to-date with its remote version, while also updating your other remote-tracking branches, enter:

>> git pull origin <branch name>.

NOTE: Again, you'll generally only want to pull from the dev branch:

>> git pull origin dev

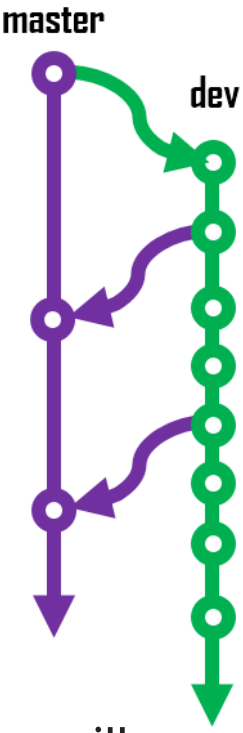
SRSSS GitHub Workflow

We'll have 2 branches in our repository, the master branch and the dev branch.

Master Branch

Contains only stable versions of complete software. When the dev branch contains stable code which overall achieves some level of tasks successfully, it can be merged into master. The code in the master branch should always be executable and capable of cleanly solving the optimisation task to some extent (even if it is missing some features which will be included in final product).

Merges to the master branch will be handled by the code architecture section.



Dev Branch

The dev branch is used for actual work, this is what you will clone on your local machines and what everyone will pull and push from.

CONGRATULATIONS!

Those are the basics. Just remember, the more often you pull (and push) the less likely you are to experience merge conflicts!