Initial Steps:

0) Make a GitHub account

1) Make a repository for your game

- This should be done by *one* person. This is designated as the "**upstream**" repository that everyone else forks from. It has no special permissions! If that person drops out / goes away for the weekend / quits computers you do not have to change your git workflow!
- Upload your local repo if you've already been working locally, create a new one on GitHub if not.

2) Each team member:

https://help.github.com/articles/fork-a-repo/

- Fork that repository
- Pull it down to your own machine
- Add the first repo as a remote called upstream

At this point you should have:

- All the code on your machine
- All the code on your own fork on GitHub

Daily Workflow:

0) Make a *develop* branch off of *master*.

git checkout master (Switch over to master)
git checkout -b develop (Make develop and switch to it)

- You only need to do this once, that branch should stay for the lifetime of the project.

1) Make a *feature-MY_FEATURE* branch from *develop*.

git checkout develop (Switch over to develop)
git checkout -b feature-MY_FEATURE

- These feature branches are short lived, and can be deleted after they get merged back into develop

2) Write your code.

- The hard stuff. Good luck!

3) Commit your code.

git add THE_FILES I_CHANGED
git commit -m "feature is donezo"

- Be a good citizen, leave a more meaningful message summarizing the changeset.

Loop between 2-3 until you are ready to share your work. Below are several scenarios that you might find yourself in, loosely ordered in how common they should be.

I'm done working for the day and want to backup my work / move between computers:

i) Push your feature branch to GitHub

git push origin feature-MY_FEATURE

iI) Pull it back down on your new computer

git pull origin feature-MY_FEATURE

I've finished my feature, I've tested it locally and now I want to merge it:

i) Push your feature branch to GitHub

git push origin feature-MY_FEATURE

ii) Start a pull request, asking a teammate to review the code.

https://help.github.com/articles/creating-a-pull-request/

- Merges should be against *develop* not *master*
- Try to ask a different person each time. One person shouldn't become the "code reviewer" or the "git merger". These are basic tasks that *everyone* should be doing *every day*.

iii) That person pulls the code down locally, double checks your work and responds.

- GitHub would like you to make these comments on the pull request. If in person feedback works better for you, that's totally fine. One is traceable/trackable and one isn't. Be aware of what you want future employers / teammates / teachers to see.

iv) After it's all good, accept the pull request (which will merge it), then make a second pull request to whoever you have designated "upstream".

I'm working with a subset of the team on some feature.

i) Push your feature branch to GitHub

ii) Use pull requests amongst that subset to sync until the feature is complete.

iii) Start a pull request to merge that feature into someone else's develop

We finished a sprint / We have a demo / We have a playtest

i) Merge develop into master

git checkout master

git merge develop

Resolve any merge conflicts if necessary

ii) Push the updated master to GitHub

git push origin master

Resources:

https://guides.github.com/introduction/flow/
https://help.github.com/articles/github-flow/
http://nvie.com/posts/a-successful-git-branching-model/