cse490j-2019au

Class Repository for CSE 490j - Special Topics in VR @ University of Washington

Using Git With Unity

When collaborating on large projects in Unity, having a method of version control is essential: this is why we use Git. Git allows us to save versions of our project as we work on it so that we still have the ability to go back to old versions if we need to.

Git: A Brief Overview

When we use Git, your project will live in a repository, or folder, on GitHub. Git allows you to create a local copy of this repository that is connected to the version on GitHub. You can **pull** a the current version from GitHub to your local machine so that you may make changes to it. When you edit the file in the repo on your local machine, you can safely edit them without changing the files on GitHub. When you have finished editing your files, you can safely **commit** the changes with a message explaining what you have changed. Once you have committed the files, you can then **push** them to GitHub where they will be stored. Git keeps track of all of the commits that you have made so that if you need to you can go back to earlier changes.

Installing Git

In order to use git we must install it first. If you already have it you may skip this step. Download the Installer for the system that you are using. - Mac OS: https://git-scm.com/download/mac - Windows: https://git-scm.com/download/win

This tutorial will cover using a command-line interface, or a GUI interface so you may choose whichever you are more comfortable with. If you wish to use a GUI you should download GitHub Desktop here: - https://desktop.github.com/

Because Git limits the size of the files you can save, you will also need to install **Git LFS** (Large File Storage).Download it here: https://git-lfs.github.com/

Then you will need to use the Command Line: - Mac: Open up a terminal window - Windows: Open up GitBash

Then run the command:

```
    git lfs install
```

Creating a Repo

First you'll have to create a repository to host your project. Start by going to https://github.com and creating an account. Once you have created an account you should see a big white button in the center of the page that says **Start a project**. You will need to give your project a name, and make sure that it is set to **public**. Check the box to **initialize with a README** as well. The last thing you will need to do is add a **gitignore**. This prevents git from tracking unecessary files which makes our repository take up less space. In the **.gitignore** dropdown, search for and select **Unity**. Finally hit **Create Repository**.

Cloning the Repo

Now we need to clone your repo onto your computer. First you'll need to copy the link to your repository.

If you are using GitHub Desktop you can clone it using**File -> Clone Repository**, and click the **URL** tab. Copy the URL from GitHub in the top right corner, and paste it into the URL link. You will have to choose where to store the repo on your machine in the **Local Path** field. Once everything is set hit**Clone**.

If you are using Command Line: - Mac: Open up a terminal window - Windows: Open up GitBash

You will need to navigate to the directory that you want to store your repo in. Then you can clone your repo using the following commands:

```
sh cd replace_this/with_path_to/your_directory git clone <link-to-your-repo>
```

Making Changes To Your Repo

Before working on any changes, you should always**pull** to make sure that your code is up to date. Once you have made your changes you need to **add** them to staged changes. You can also**remove** files that you don't wish to track. You can see which files have been added by checking the **status**. Once you have added, you need to **commit** and provide a summary of the changes. Once you have committed, you can**push**.

To use all of these commands:

Command	GitHub Desktop	Command Line
Pull	Repository -> Pull	git pull
Add	Done Automatically, check box on left side	git add file.filetype
Remove	Uncheck box on left side	git rm file.filetype
Status	Displayed on left	git status
Commit	Bottom Left Corner, add summary, hit commit	git commit -m "Summary"
Push	Repository -> Push	git push

Merge Conflicts

Sometimes your attempt to push new changes may fail. This is usually because you don't have the most up-to-date files, most-likely because another person has pushed their changes before you have pushed yours. This is usually an easy fix. If someone has added or changed different files than the ones you have been working on, you can usually simply just **pull** from GitHub.

If you were working on the same file as someone else, then you have a**Merge Conflict**. To resolve this, you will need to go into the file that is conflictied. Git will have highlighted the spots where differences exist. You should go through and choose the changes that you would like to keep.

Once you have made the changes, on desktop you can use **Branch -> Merge into current branch...**. Using command line:

```
    git merge origin/yourBranchName
```

Branching

If you need to work on something separately from somone else without worrying about merge-conflicts, you can create a **branch**. For this quarter, you will be creating your own branch to work in.

To do so in Desktop find the tab at the top labeled**Branch -> New Branch...**. Name your new branch with your name for the purpose of this class. Then hit **Create Branch**.

Using the command line, to create and switch to a new branch you can enter the following command:

```
    git checkout -b yourNameHere
```

From now on you should work in this branch only. Before editing your files and commiting, make sure you are working in your branch.

Updating From Master Branch

For your assignments we will add new files every week, which will require you to get these new files from the master branch. From your branch you can get files using GitHub Desktop by going to **Branch -> Update from default branch**.

Using command line you can use the command:

```
    git checkout yourBranchName
   git pull origin master
```

Turning In Your AssignmentS

When you finish an assignment, make a commit with the summary "**hw#-final**", with **#** being the number of the current assignment. For example, the first assignment should be "**hw1-final**". Please email the course staff with a link to your repo to turn in this first assignment.