New Unity Project Setup

Rishav Banerjee, Omang Baheti

December 20, 2024

To follow the video guide version instead, click here.

Contents

1 Introduction **1**

2 Create a new project **1**
2.1 Unity Hub 2
2.2 Unity Editor 3
2.3 Remove Readme assets 5
2.4 Remove new input system 5

3 Connecting with Git **7**
3.1 Adding to GitHub Desktop 8
3.2 Pushing online to GitHub 9

4 Conclusion **10**

1. Introduction

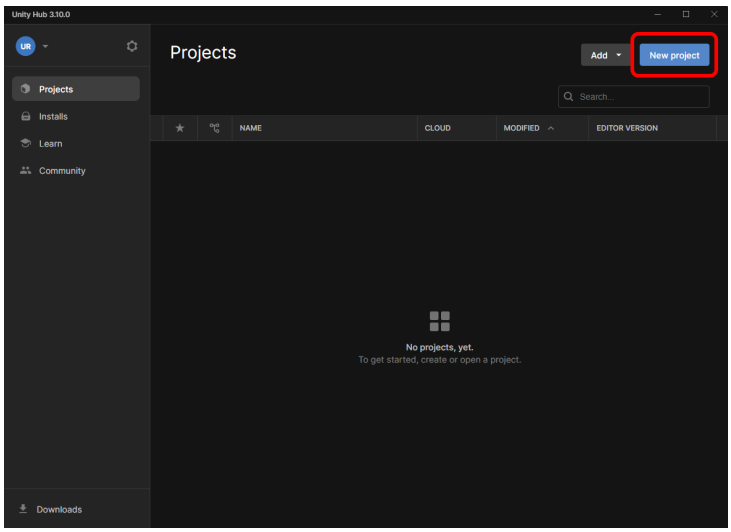
Welcome! This document will cover the steps to set up a new Unity project. You will need to set up multiple new projects throughout this course (one for each assignment at least), so please refer to this document if you are unsure or unable to remember any crucial step! We'll go over how to create a new Unity project, remove readme assets, set the active input handling method and connect it with git and GitHub to easily share your work.
This document assumes you have followed the previous guide to set up your Unity development environment. If you have not done so, please follow this guide first.

2. Create a new project

First we'll have to create our Unity project.

2.1. Unity Hub

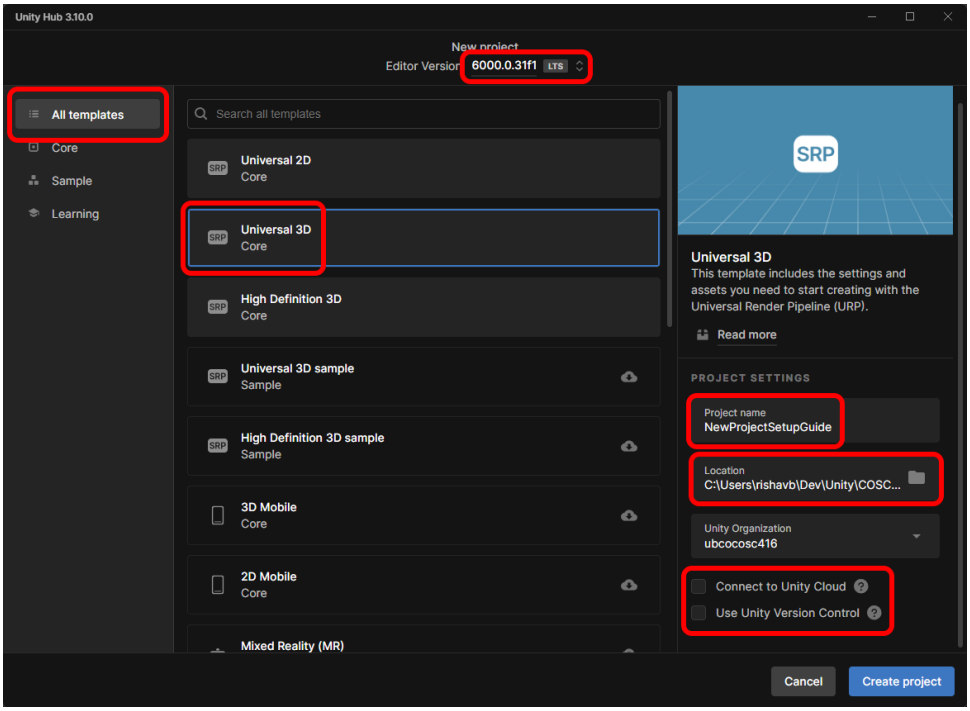
Open up the Unity Hub. In the top right corner, click on "New Project".



Next you should see the project creation screen. Now there are a few things you want to make sure here.

- On top, you should see Editor Version 6000.0.31f1. If you are unsure how to get that, follow the previous guide.
- You should be on the "Templates" menu. When creating a project for the first time, Unity Hub puts you on the "Samples" menu, which has preloaded assets and scripts. You want to create a blank new project instead.
- For this course, we'll mostly be using the Universal 3D Core setup. The 2D variant generally has a few settings tweaked around for 2d games, and High Definition 3D will be too graphically intensive for regular laptops to work with.
- Make sure you set a good project name! Generally a good policy to follow is to use `CamelCase`. For instance, since this is a new project setup guide, a good name would be `NewProjectSetupGuide`.
- You also want to make sure it's in the right folder for your organization requirements.
- Finally, make sure to DISABLE "Connect to Unity Cloud" or "Use Unity Version Control". We'll be using Git as explained in the previous guide for our version control purposes.

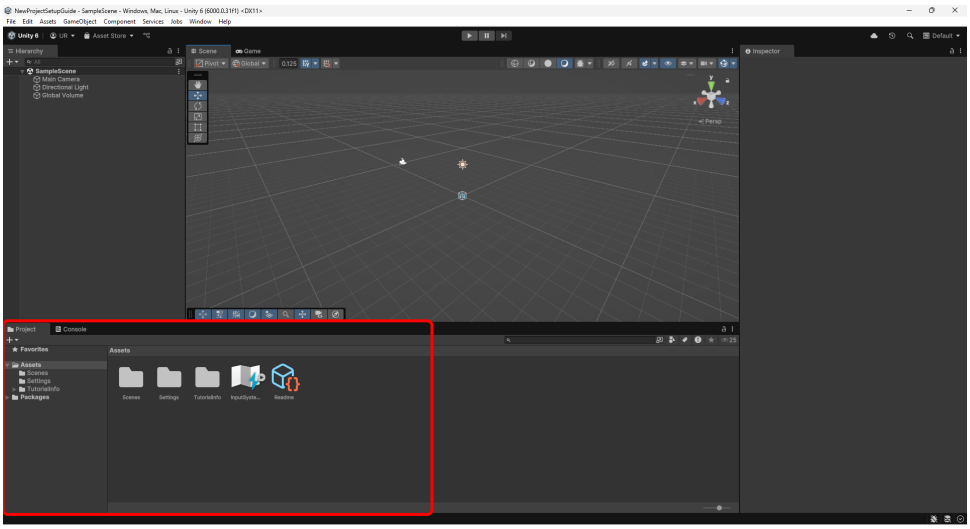
Your hub page should look something like this:



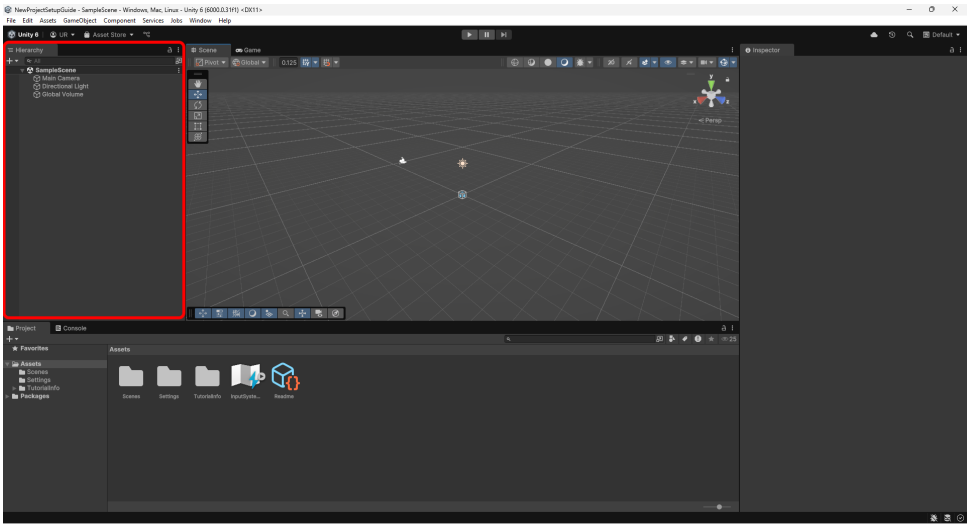
With this you can click on Create Project on the bottom right. Since this is the first time, it will take a long time to launch. Once the project is created, you should see the Unity Editor!

2.2. Unity Editor

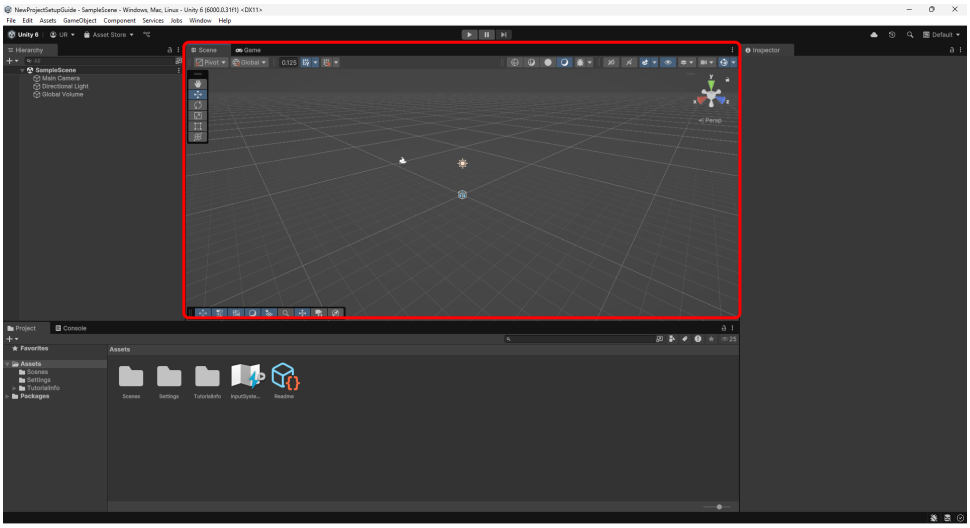
We'll quickly go over the different parts of the Unity Editor. This will be explained more in-depth during lectures and upcoming guides, but for now a brief overview should suffice. The Unity editor is composed of many windows, each with its own functionality. This default view has some of the most important windows laid out, however you can customize this and change this up easily.
The window at the bottom is the Project window. This is the same as your file explorer or finder, but rendered within unity, inside the Assets folder. This is where all your relevant game files, assets, code, etc. will reside.



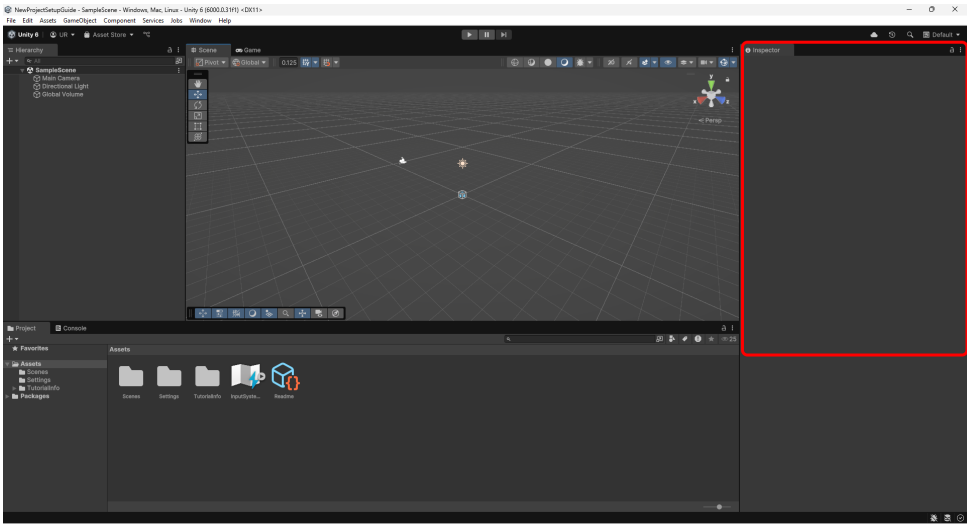
On the left we have the hierarchy. A unity game is composed of many scenes. Each scene has various GameObjects present in it. The hierarchy shows us the present GameObjects in the scene. Our current scene "Sample Scene" has a Main Camera for viewing the game, a Directional Light for lighting and a Global Volume for post-processing.



In the middle we have the 3d scene view of our current scene. This is currently empty now, but we can manipulate various objects, alter their properties, and perform many other actions in this scene view.



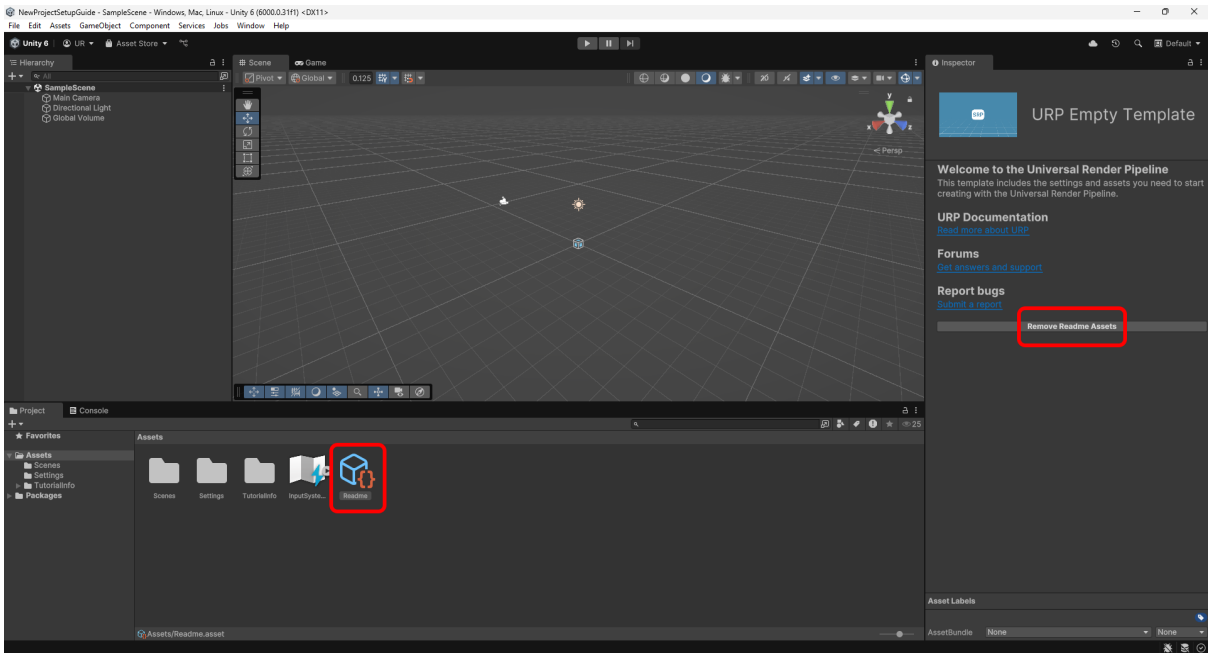
Finally, to the right we have the Inspector window. This shows us contextual information for anything we select from either our Project window or our Hierarchy window. When selecting assets from our Project window it gives us options to modify those assets as required. When selecting GameObjects from our Hierarchy window, it shows us the relevant components attached to that GameObjects that define its behavior.



This should suffice as a quick overview for now. We'll now see some of this used for the following sections.

2.3. Remove Readme assets

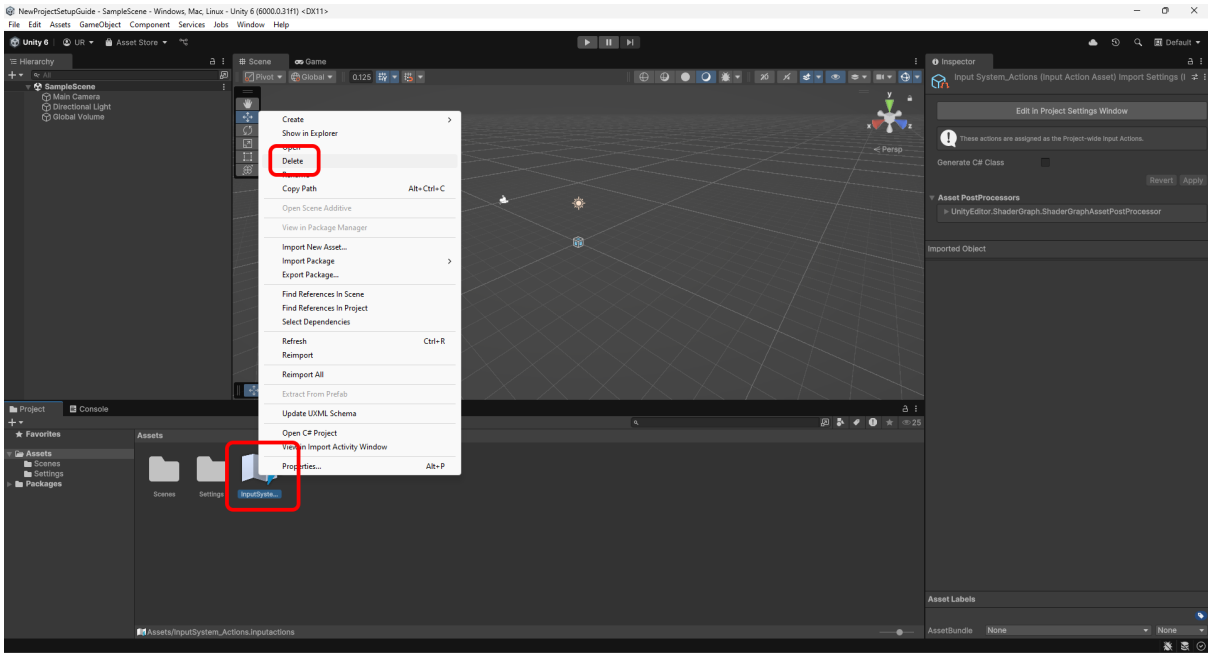
A blank Unity project comes with a readme asset for further reading. We'll be removing that now. In the project window, click on the Readme file. In the inspector, there should be an option to "Remove Readme Assets" now. Click on that to remove the readme assets. And that's it!



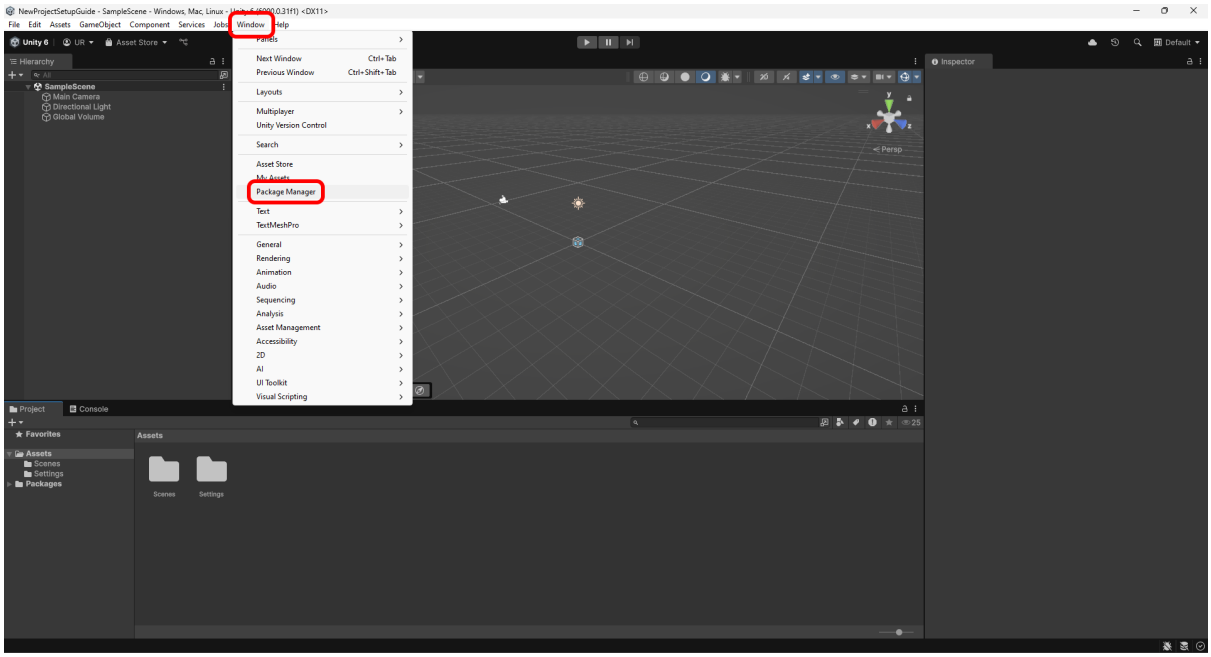
2.4. Remove new input system

Unity currently has 2 Input Systems. The older one works purely through code, and is easy, but not very scalable. The new input system is very scalable and well organized for different platforms, but requires a lot of setup. For this course, we will be using the old

input system for simplicity's sake. You are welcome to use the new input system, but you will be mostly on your own to debug it.
To remove the new input system, we'll be deleting the asset in our project first. In the project window, right click (or two finger tap on a MacBook) on the `InputSystem_Actions` asset.

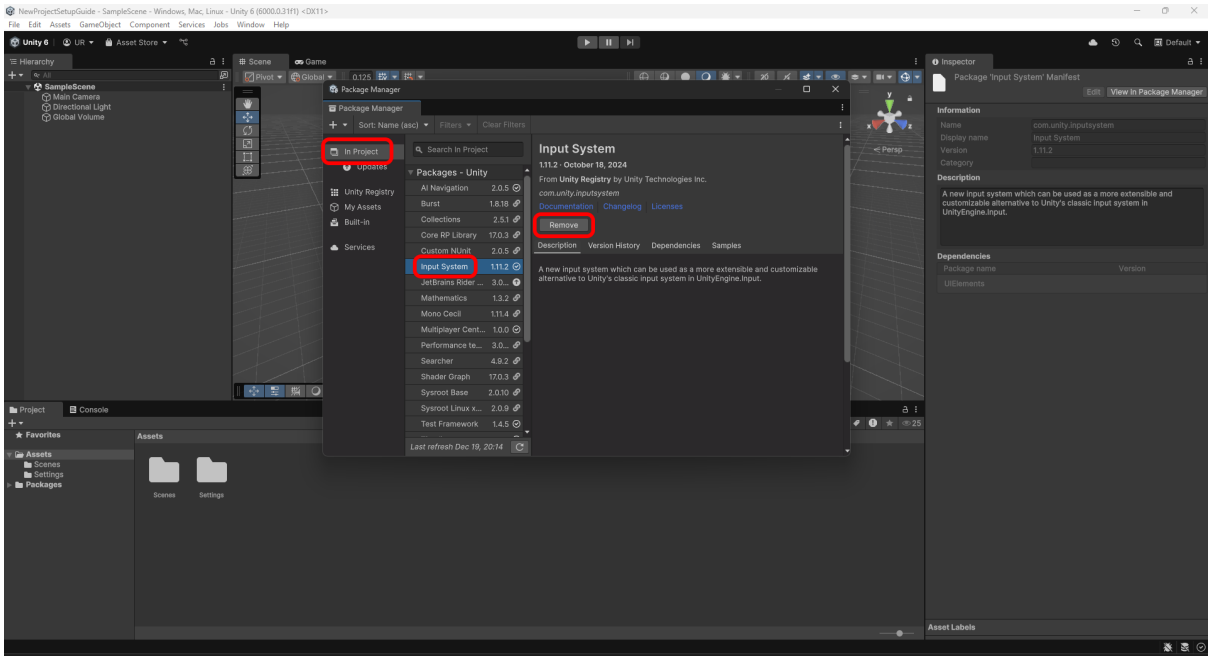


And next, we'll be removing the package associated with the new input system as well, so that Unity will default to the older framework. On the top menu bar, click on Window. This will bring up all the possible windows in the Unity Engine. Open Package Manager.



This window shows all the Packages that the current Unity project has. A blank Unity project is loaded with a lot of helpful packages from Unity. New packages can be added

from the Unity Asset Store, or from GitHub as well. Make sure you are in the "In Project" tab on the left, and in the "Packages-Unity" list, click on "Input System" and "Remove".



And that's it! Your Unity project is ready for development. But before we get started developing, we should connect it with Git and GitHub.

3. Connecting with Git

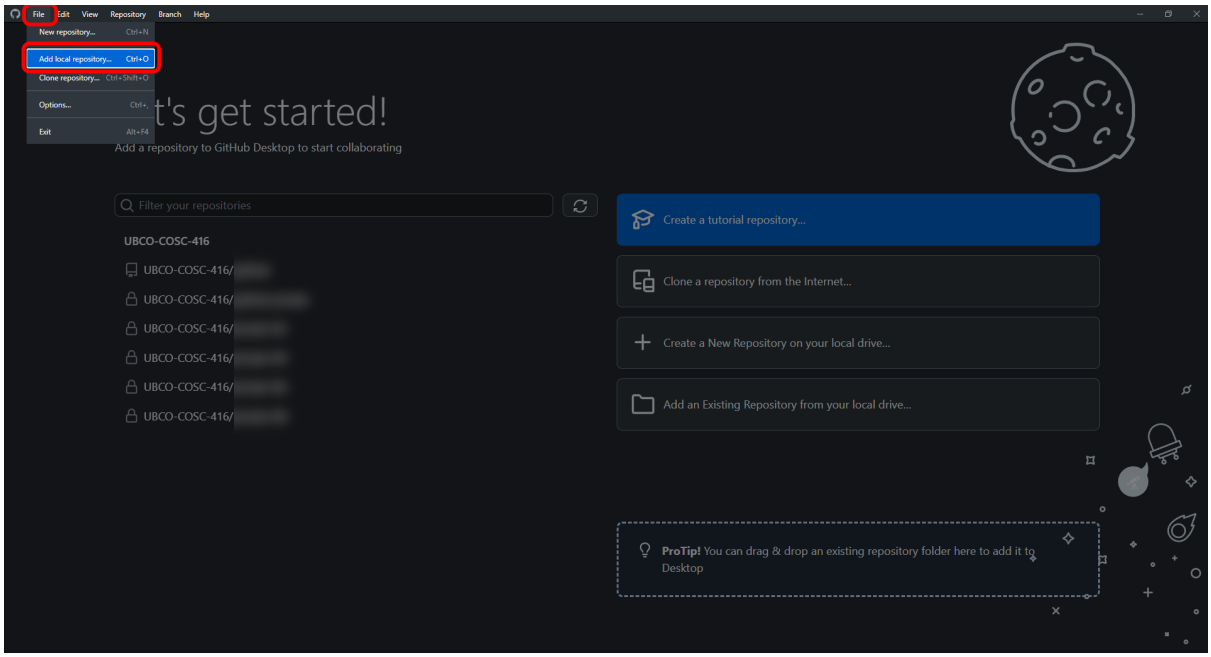
> **Note**
>
> If you are already comfortable using git, feel free to skip this section. Just make sure to set up a unity `.gitignore` from gitignore.io, or select Unity as the gitignore option for GitHub Desktop in your root folder. Your root folder is the one with Assets, Packages, ProjectSettings, etc. and should be the root folder of the final repo as well. Also make sure to have your repo set to public.

Git is a version control and collaboration tool that allows you to keep track of your progress and changes in your project. Using git, you can collaborate with others, set save points for your projects so that if you make some destructive changes, you can roll back to a previous save point, and easily back your entire project to the cloud with GitHub. In this section, we'll give a brief overview of how git works, how to connect your project with Git.

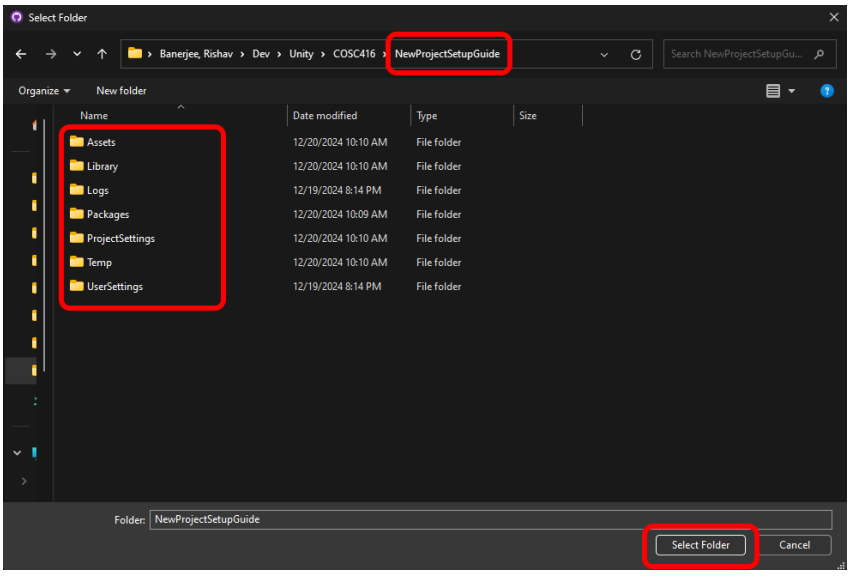
At its very core, Git simply tracks all changed files in your project. The project folder, once tracked by git, is referred to as a Git Repository. Every time you've made a feature or some changes, you can create a save point (called a "Commit"), where git will track all the changes from your previous commit. In future guides we'll be showing how to record these commits as you are developing your features, but for now, we'll show how to set up a new repository with Git and push it to GitHub.

3.1. Adding to GitHub Desktop

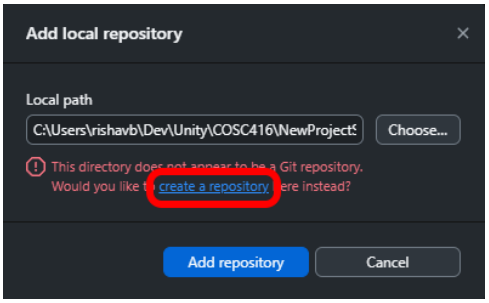
Open up GitHub Desktop. If you don't have it, please refer to the previous guide. You should already have your GitHub account logged in if you followed the guide properly. Now on the top menu bar, click on "File", and then "Add Local Repository".



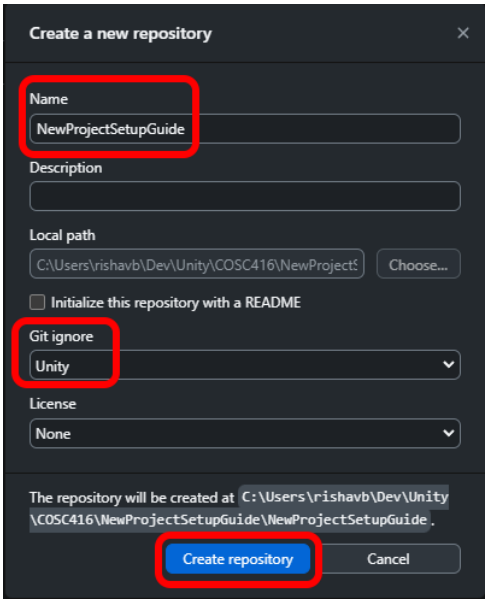
In the new pop up, to select your project folder's path, click on "Choose..." and then navigate to your project folder. Your project folder should be the one where there are folders for Assets, Library, Packages, ProjectSettings, etc.



Select this folder, and click on "Add Repository". When you do so, you will see a prompt saying "This directory does not appear to be a Git repository. Would you like to create a repository here instead?" Click on "create a repository".



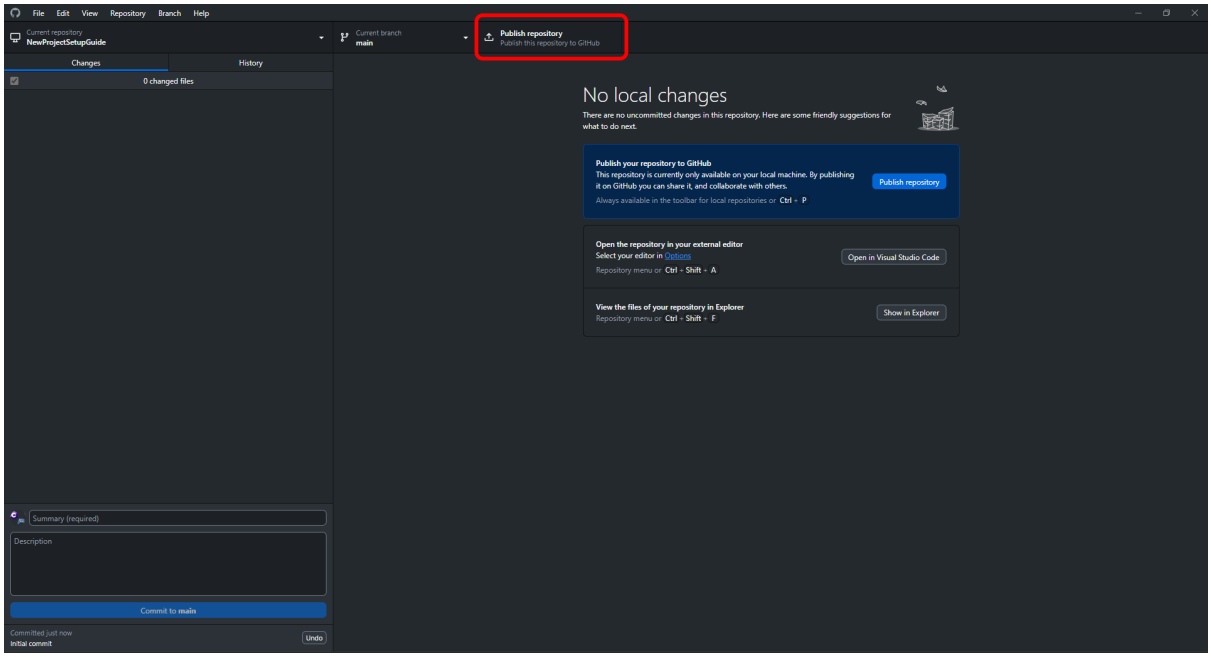
Then, in the next prompt, you'll have to select some basic configuration options for your repository. The main thing we want to make sure here is the name of the project is what you want (normally the same name as that of the Unity project), and that you select the "Unity" gitignore from the dropdown of options. This option makes it so that Git doesn't track all the files that can be regenerated by the Unity engine easily, and only tracks the project specific folders, such as Assets and Project Settings. Then finally, click on Create Repository.



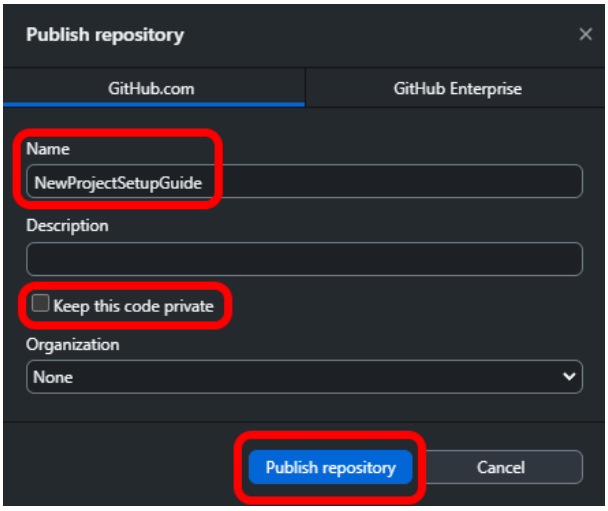
Now your Git Repository has been created! Next we'll push it online to GitHub for cloud backup, sharing your project and collaboration purposes.

3.2. Pushing online to GitHub

To publish your repository, simply click on the "Publish Repository" option on the top center of the window.



You will have a few more configuration options to decide. The main thing that matters is making sure the project name is once again, the same as the Unity Project name, and switching OFF "Keep this code private". This is so that you can give a link to anyone for your project, and they can access it without necessarily being an invited collaborator.



Click on Publish Repository, and you are done! Now if you go to your GitHub account, you should be able to see your new Unity repository as a project under the Repositories tab.

4. Conclusion

Now you can begin developing your game! Remember to always do this process before starting a new assignment for all future assignments. That will be it for now. Thank you for reading, and see you next time!