hello [GitHub] world

Here is a small workshop on how to setup a github repository for your unity projects as well as adding large files to our project! GitHub has a limit on how large one file can be on a commit [100mb].

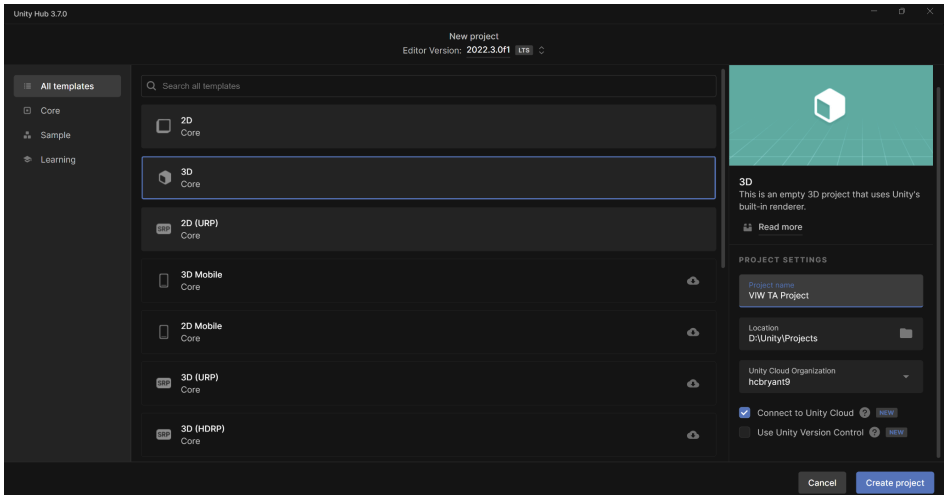
**prerequisites*: github desktop, unity v 2022.3.0f1 (very important your team all has the same version) , visual studio (yes this is different from vs code, you should be prompted to download the corresponding version when you download the correct verison of unity).

**note* that the lfs is only for models that break the 100mb limit which is probably not gonna happen with us. Also this is gonna take up like 10 gbs or so on your computer ;-;

Step 1: Creating a new Unity Project

We learned how to do this in Oscar Keyes' Tutorial but let's quickly recap. (It's important to note that only one person in your group needs to create a Unity project.)

First let's open up Unity Hub.
Next, let's hit New project



Remember to give your project a name, set an editor version, and assign a location on your computer for the project to live. We'll select 3D core for now, we will learn more about Unity's Render Pipelines later.

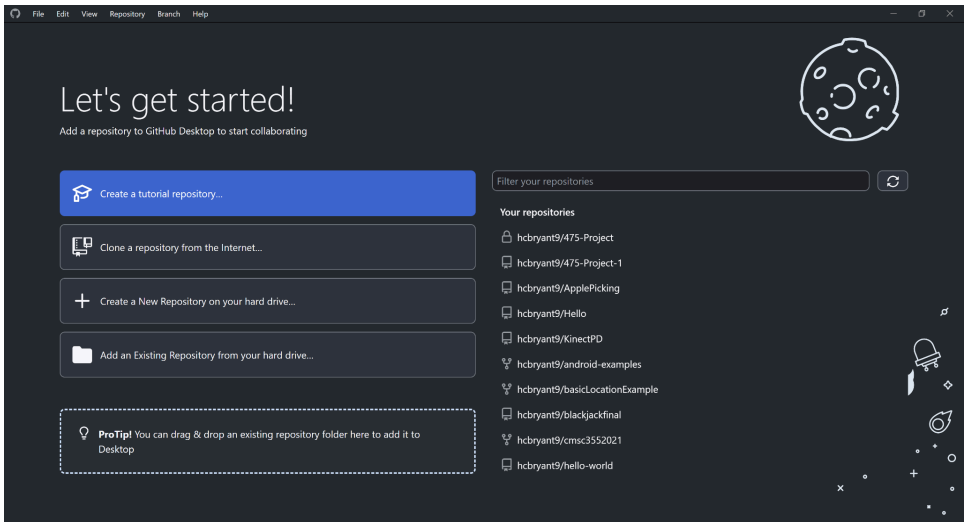
Next, Hit Create Project. It'll then take a few minutes to set up your project files and launch the Unity editor. It may prompt you to install Visual Studio, make sure to check that box as we will be utilizing it to write scripts in the future.

Step 2: Creating a GitHub Repository
https://desktop.github.com/

GitHub desktop is a GUI app to work with Git and GitHub where you can collab, manage repositories, commit changes, and restore old versions of your project.

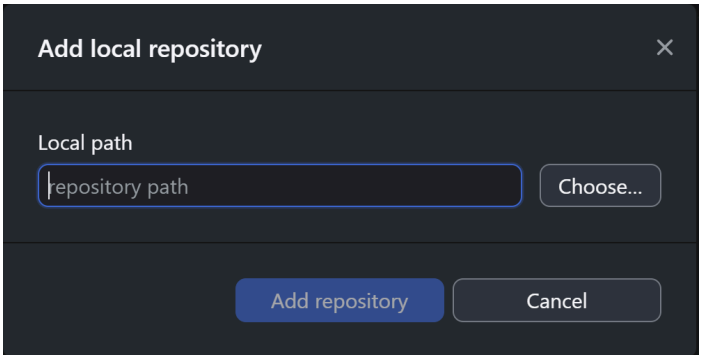
When you log in for the first time you will need to sign in with your GitHub account you created as well as a name and an email address that you want to associate with your commits. If you don't see this pop-up right away you can go to *File->Options->Accounts*.

We should have a screen that looks similar to this:

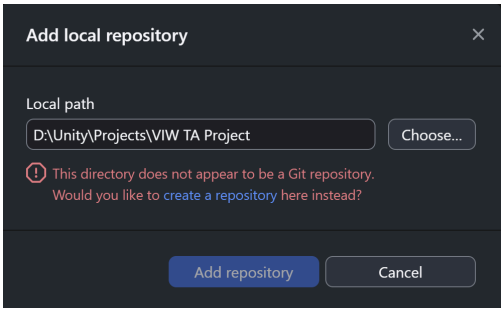


Now, we are going to hit Add an Exisiting Repository from your Hard Drive.
Or if you don't see this
File-> Add Local Repository

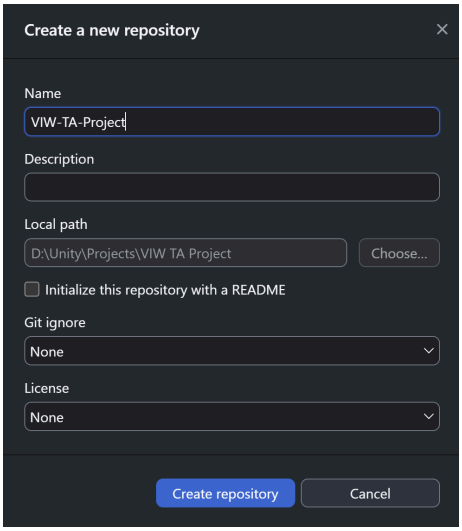
(Note this is still for setting up a repository for the first time we will get to joining your group member's repository in a second).



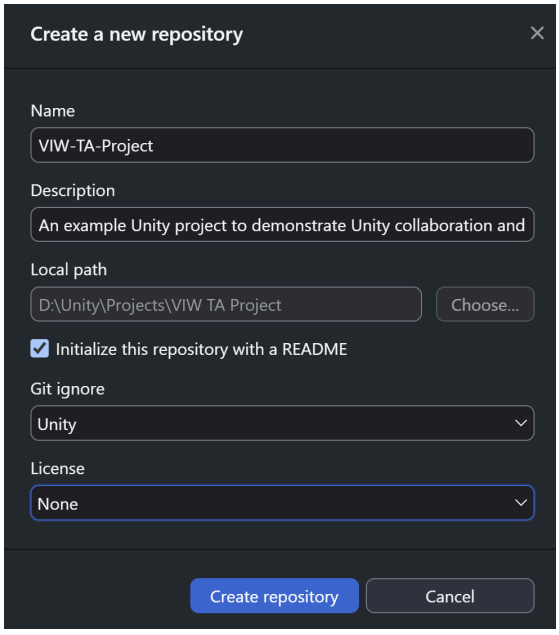
Now we'll see this window pop up and go ahead and navigate to the Unity project in file explorer and hit "Add Repository".



You should now see the option to create a repository. Click that!



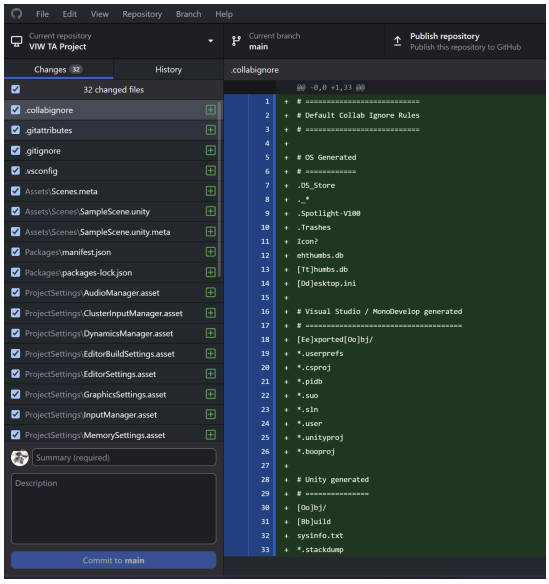
You'll then see this screen. Go ahead and fill out the items as shown below. Make sure it add Unity under Git ignore.



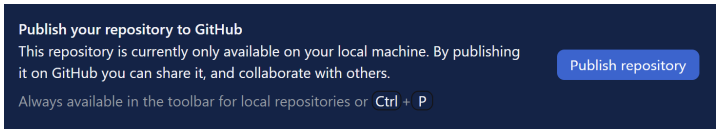
Now let’s hit Create Repository.

(possible *bugs* - If GitHub Desktop tells you that there are too many files, repeat the steps and it will let you create it).

Nice! Now we should have the following screen.



Let’s type in a summary for initial commit and commit to main. After this, we should see we have 0 changed files and a window like the following.

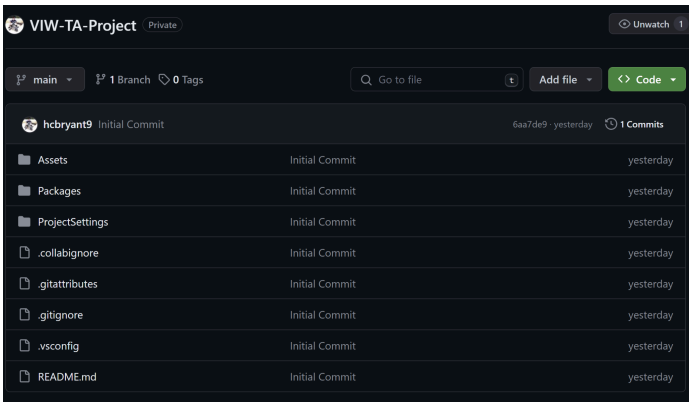


Hitting Publish repository will publish the repository to GitHub!

Great! We’ve now created a GitHub repository to share all our Unity project files, but how do we share the repo with other group mates?

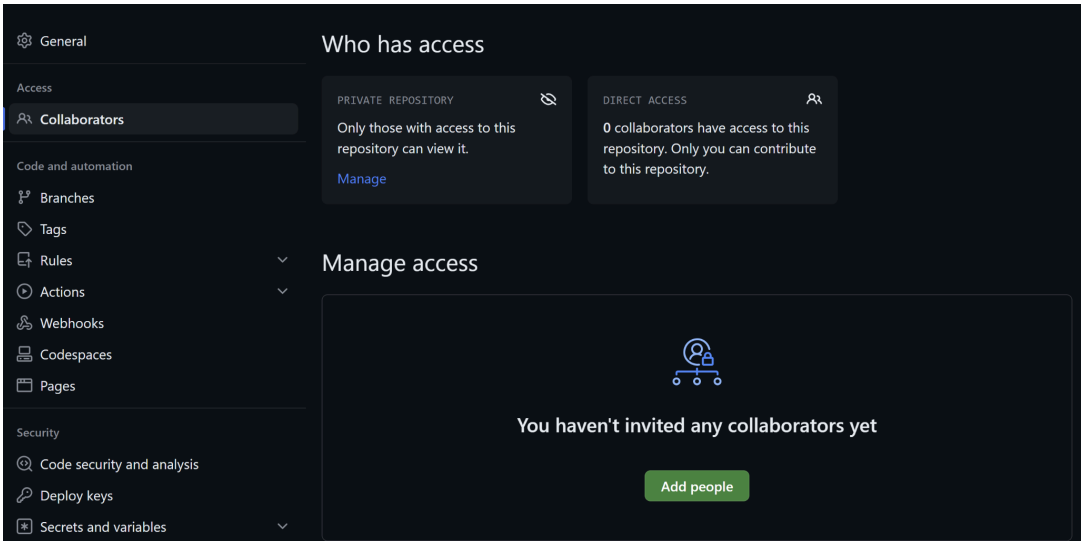
Step 3: Sharing
https://github.com/

Now that we've created the repository & published it we can now view it on GitHub. It should be at the address github.com/USERNAME/REPO_NAME.

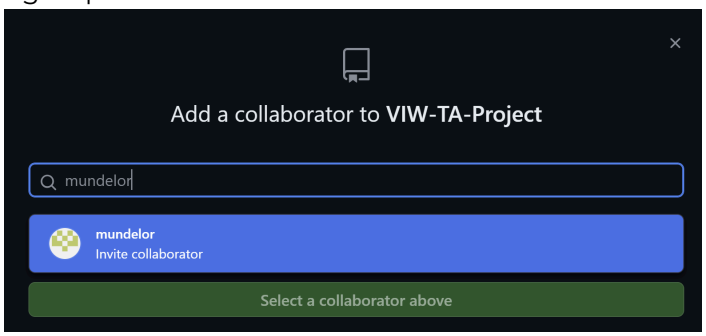


Cool, now let's add our group members to the project so they can clone the project and work concurrently with us on the project.

Let's go Settings -> Collaborators -> Add People



Next, type in your group member's GitHub username and add them to the repository!



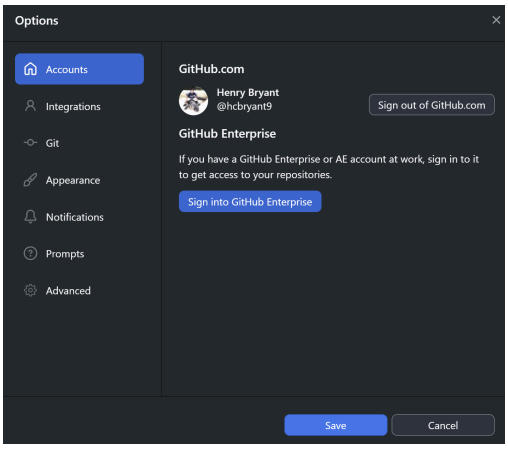
Nice! Now your friend will see an invitation in their email associated with their GitHub account to accept the invite.

Great, now our group members are on the project. Next, we'll go into how your friends can copy or clone the repository to their machine.

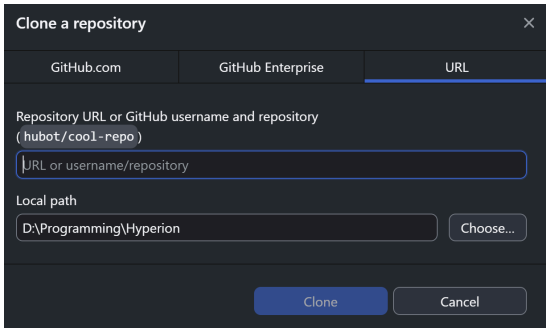
Step 4: Cloning

So, you've just joined your groups repository on GitHub and you want to start adding things in the scene. Open up GitHub Desktop!

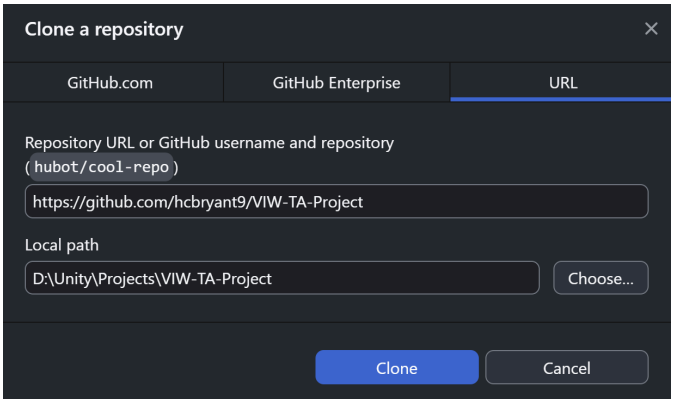
Make sure your GitHub is connected to your GitHub Desktop by going to File -> Options



If you are all logged in, go to file -> clone repository

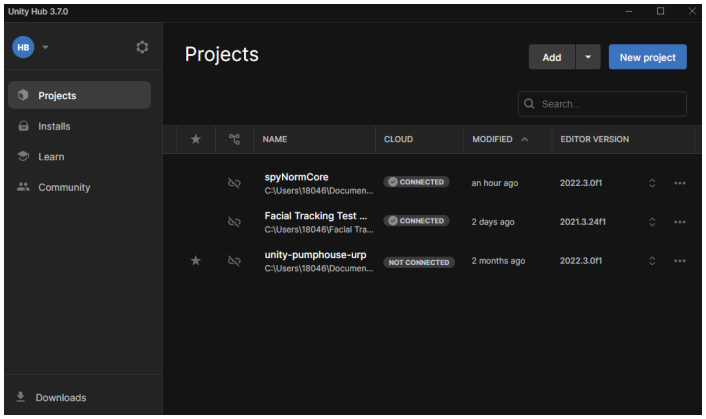


There are several ways to clone a repository from GitHub, but I'm going to choose via URL. Just copy and paste the Repository's web address into box, like so.



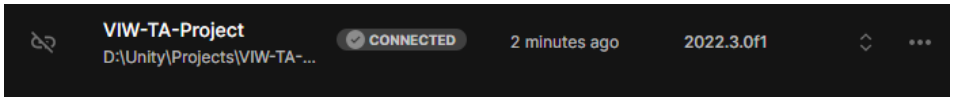
Now, hit Clone!

Success! We've cloned the repository to our computer, now we'll open it in Unity Hub.



Let's hit Add-> Add from Disk
Then navigate to your project folder and hit 'Add Project'

Great, now we can see our project in Unity Hub



If you see an Icon where the ‘Editor Version’ option is , you need to install the same version that your group member created the project in. Unity will usually tell you which version that is. In this case it is v2022.3.0f1.

Now, let’s look at how to use GitHub to collaborate.

Step 5: Collaboration & Use

GitHub & GitHub desktop can be very overwhelming at first and they are indeed comprehensive (tech companies like Netflix and AirBnB use GitHub in their tech stacks), but repositories are only as comprehensive as you configure them. Lucky for most you have a Computer Science student who is experienced with GitHub committing and pulling or pushing techniques if you get confused. There are also several resources for correct etiquette.

Brackeyes tutorial for GitHub & Unity

There are far too many features of GitHub to cover within this condensed tutorial, but let’s go over the basics.

1) .gitignore

If you followed this tutorial you will have generated a .gitignore file custom for Unity projects.

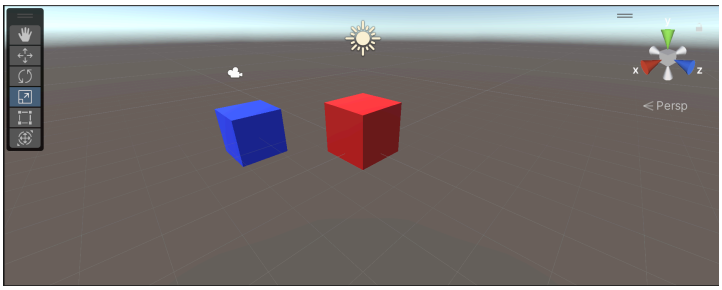
```
# This .gitignore file should be placed at the root of your Unity project directory
#
# Get latest from https://github.com/github/gitignore/blob/main/Unity.gitignore
#
/[Ll]ibrary/
/[Tt]emp/
/[Oo]bj/
/[Bb]uild/
/[Bb]uilds/
/[Ll]ogs/
/[Uu]ser[Ss]ettings/
```

This file does not let you send out any files in these folders since they’re not necessary to share among people in the repository. It will also discard any .meta files that Unity generates for your assets.

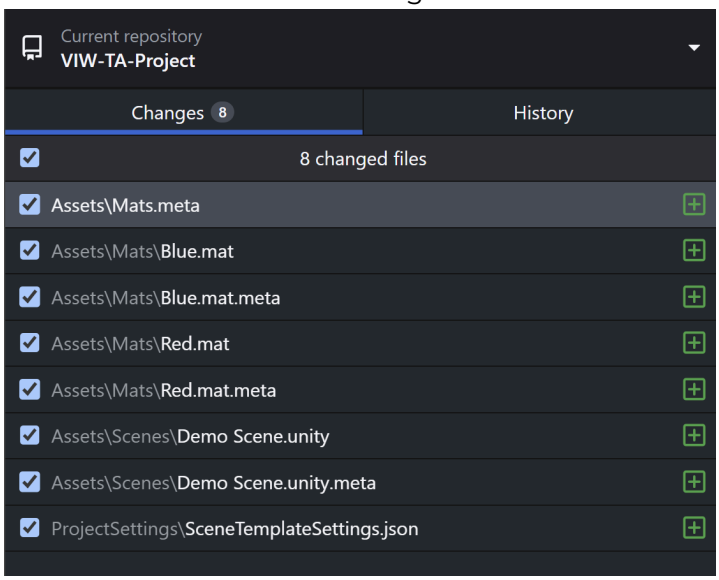
2) Pushing

Making changes and sending them to the GitHub repository is known as pushing. Whenever you’re making changes in the Unity Project, be sure to push commits when you reach a stopping point or implement a feature. Let’s look at an example.

Say I cloned my group’s repository and opened it in Unity. I’m want to add 2 cubes to the scene and color them red and blue.



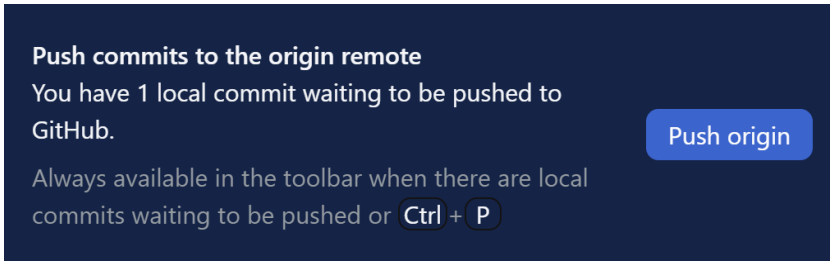
After I create them I'll hit save and go back to GitHub Desktop.



It'll show me everything I changed: Creating a new scene, a new folder for mats, and changing the unity scene.

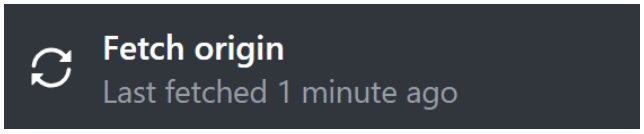
Next, write a summary for what I changed. In this case I'll write something like "Adding 2 cubes: red and blue"

Next, hit Commit to Main and then Finally hit Push Origin.



3) Pulling

Your friend has just messaged you and said "Hey I got those 2 cubes in the scene we talked about". So you'll open up Unity Desktop and Hit "Fetch origin".



This tool let's you see if there's any changes to the repository you don't have. You'll see that your one commit behind the most recent so you need to Pull the changes. Now open the project and you'll see that the cubes are in the new scene.

4) Conflicts

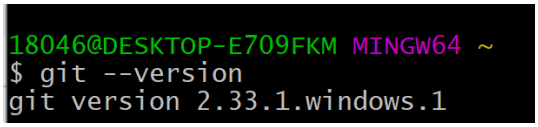
Conflicts are when you Pull from the repository and one of the files overwrites your own. For example, say I had made changes to the same scene as my friend put the cubes in. I won't be able to combine the files since the scene is only 1 file. This is where

coordination with your team mates comes in. If one person is transforming objects in the viewport, another can be working on a certain C# script and another can be working in another scene. Make sure that you guys aren't crossing files. (This can be difficult! You can try experimenting with branches but that is out of the scope of this tutorial. Ultimately, I recommend letting each other know when you will be working.)

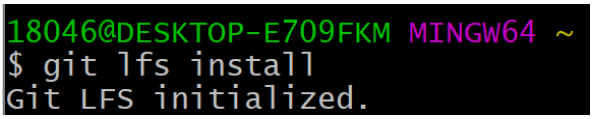
Extra: GitHub LFS

Now we have a good understanding of GitHub and version control we can dive into our next issue which is that GiHub has a limit on the size of a file can be: 100mb. The only files that will go over this size in a unity projects are large 3d models like FBX files. But we can work around this with GitHub LFS.

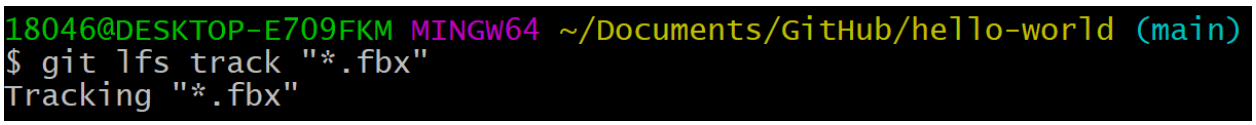
The requirements for this part is having Git installed on your computer. Run the following command "git –version"



Next run "git lfs install"



Now change directories to your existing repository of your project folder and use the following command "git lfs track "*.fbx"



This will generate a .gitattributes file or amends the existing one. Note that any file type can be added to this list that is exceeding 100mb not just fbx. In my case, I will also add .obj as an extension I want to use LFS with.

It is recommended that you commit this change before adding an fbx file and committing it.

Now we are going to add the .obj to our local project.
Then we use the path to the file to add it to our repository.
Finally, we commit and push!

```
18046@DESKTOP-E709FKM MINGW64 ~/Documents/GitHub/hello-world (main)
$ git add /c/Users/18046/Documents/GitHub/hello-world/largeObject.obj

18046@DESKTOP-E709FKM MINGW64 ~/Documents/GitHub/hello-world (main)
$ git commit -m "adding largeObject.obj"
[main efbcf65] adding largeObject.obj
 1 file changed, 3 insertions(+)
 create mode 100644 largeObject.obj

18046@DESKTOP-E709FKM MINGW64 ~/Documents/GitHub/hello-world (main)
$ git push
Uploading LFS objects: 100% (1/1), 112 MB | 18 MB/s, done.
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 388 bytes | 388.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/hcbryant9/hello-world.git
   6eec762..efbcf65  main -> main

18046@DESKTOP-E709FKM MINGW64 ~/Documents/GitHub/hello-world (main)
$
```

If you have any questions/issues email me at bryanthc@vcu.edu or refer to the LFS documentation