USING GITHUB WITH UNREAL ENGINE

BY: REBECCA ALOISIO

CONTEXT OF THIS PRESENTATION

- This presentation was created using:
  - Git version: 2.30.0
  - Git LFS version: 2.13.2
  - GitHub Desktop version: 2.6.3
  - Unreal Engine: 4.25.4
- It was also created on 2/12/2021
- Elements of this presentation may carry over to other versions of the software, but there are no guarantees.
- Any prices and/or data plans mentioned were valid on the date of creating this presentation.

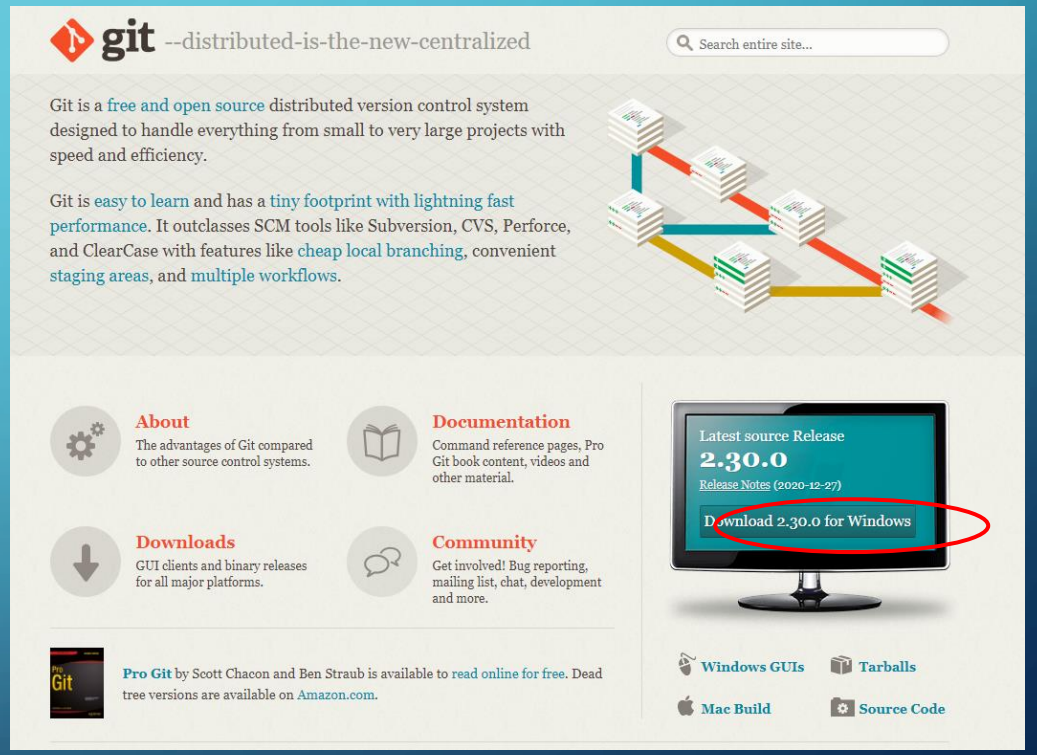
NEEDED SOFTWARE AND TOOLS

GITHUB ACCOUNT

- Seems obvious, but make sure you have one.
- If you are working in a group, make sure you get your teammates usernames. You'll need them later.
- Go to: https://github.com/

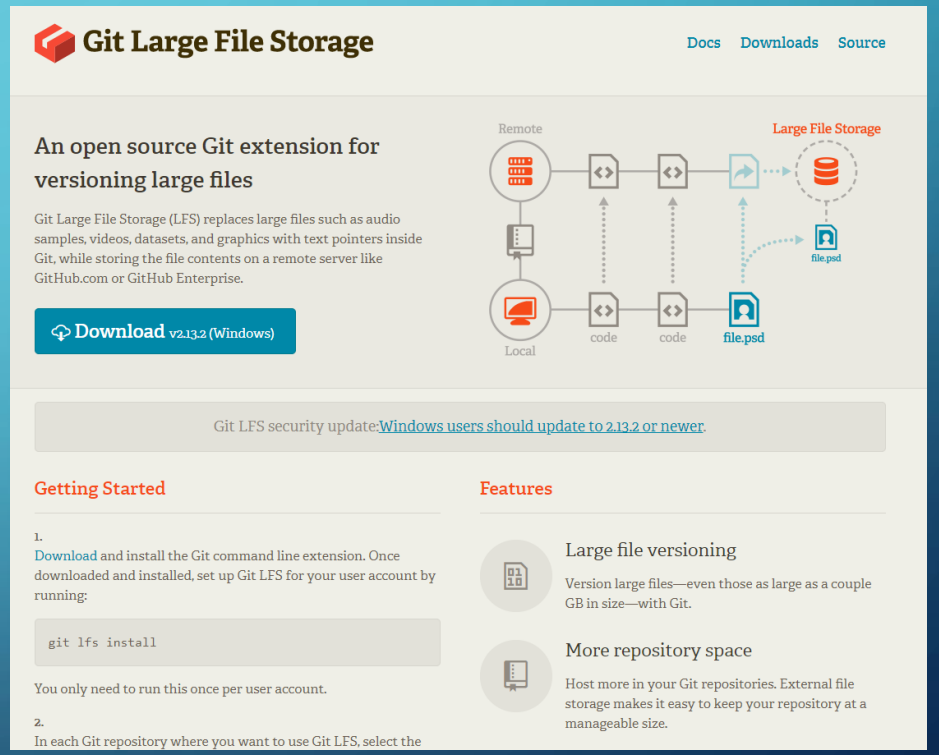
GIT

- Git is version control system.
- Useful for:
  - backing up your projects
  - Collaboration
- Download at: https://git-scm.com/
- Website should look like this->
- Circled is the download button.



GIT LFS

- Short for “Git Large File Storage”
- Needed for files that are too large for standard GitHub
  - GitHub’s limit is 100 MB for a file
- Download at: https://git-lfs.github.com/
- Website should look like this->



GIT LFS CONSIDERATIONS

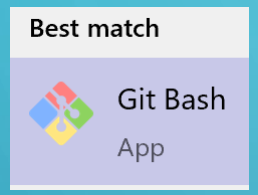
- You only get 1 GB of Git LFS storage and bandwidth for free with GitHub.
- After that, it costs $5 / month for every 50 GB of storage and bandwidth you need.
  - You can purchase this under “Billing & plans” in your GitHub Account settings.

WHAT ARE GIT LFS STORAGE AND BANDWIDTH AND HOW IS USE DETERMINED?

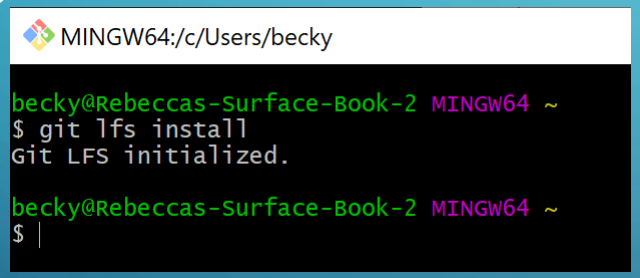
- **Storage** is used whenever a file is uploaded to the repository.
- **Bandwidth** is used whenever a file is downloaded.
- The amount used is based on the files size.
  - i.e. a file that is 500 MB will use that amount of your allotted storage and/or bandwidth.
- This applies for any small change, so USE WISELY!

INITIALIZE GIT LFS

- Using Git Bash (Installed with Git):
  - Open Git Bash



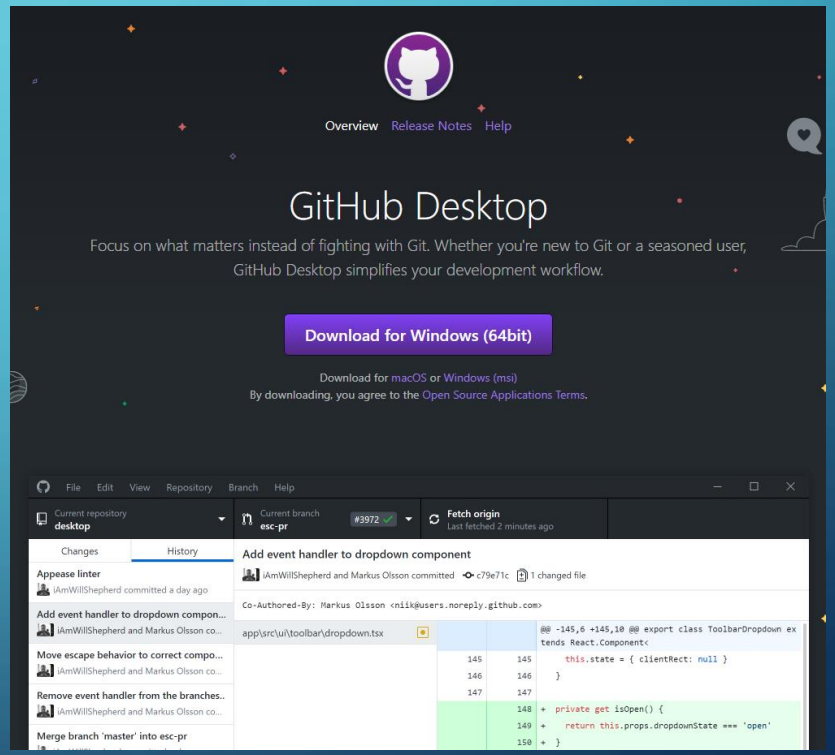
  - Type: git lfs install



- Note: If opening GitHub Desktop it may ask if you want to initialize Git LFS. Accept that as well.

GITHUB DESKTOP (OPTIONAL, BUT RECOMMENDED)

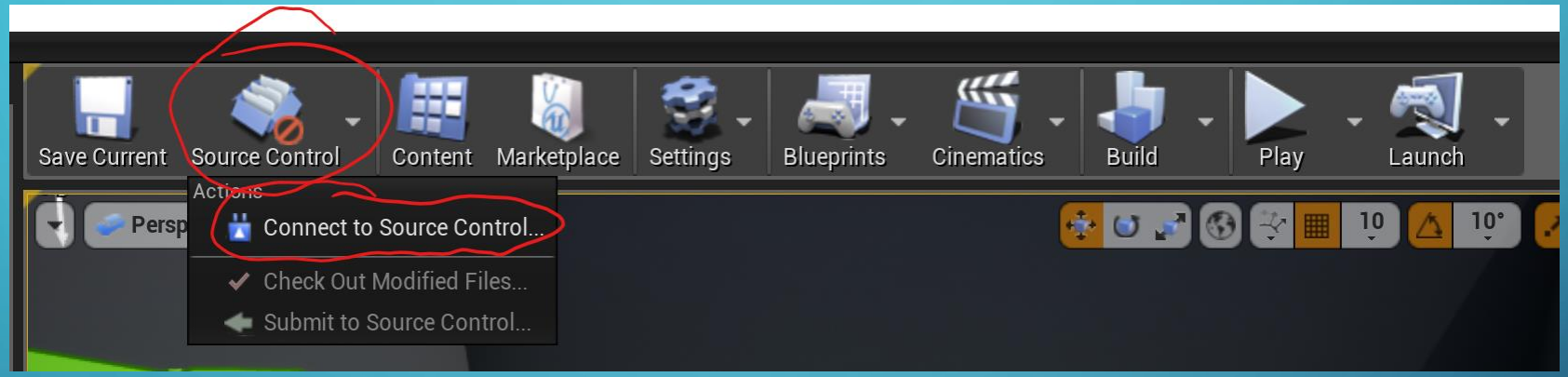
- If you don't want to mess with console commands, use this.
- Gives a GUI for git and quickly connecting projects with GitHub.
- Download at: https://desktop.github.com/
- Website should look like this->

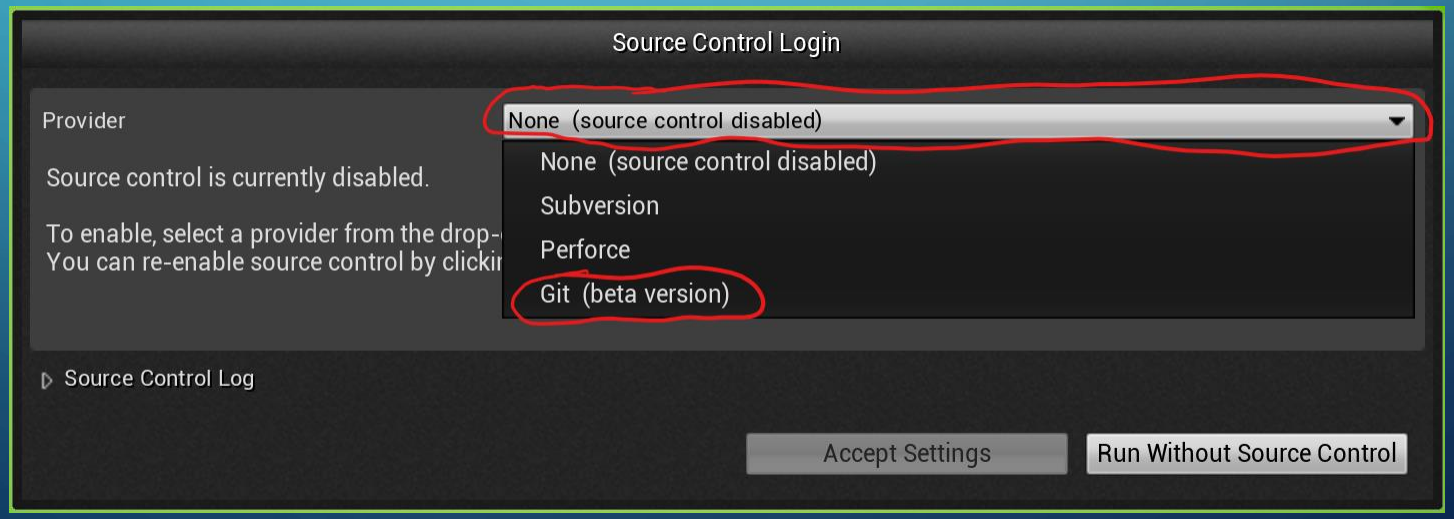


CONNECTING AN UNREAL PROJECT TO SOURCE CONTROL

FIRST TIME SET UP – PART 1

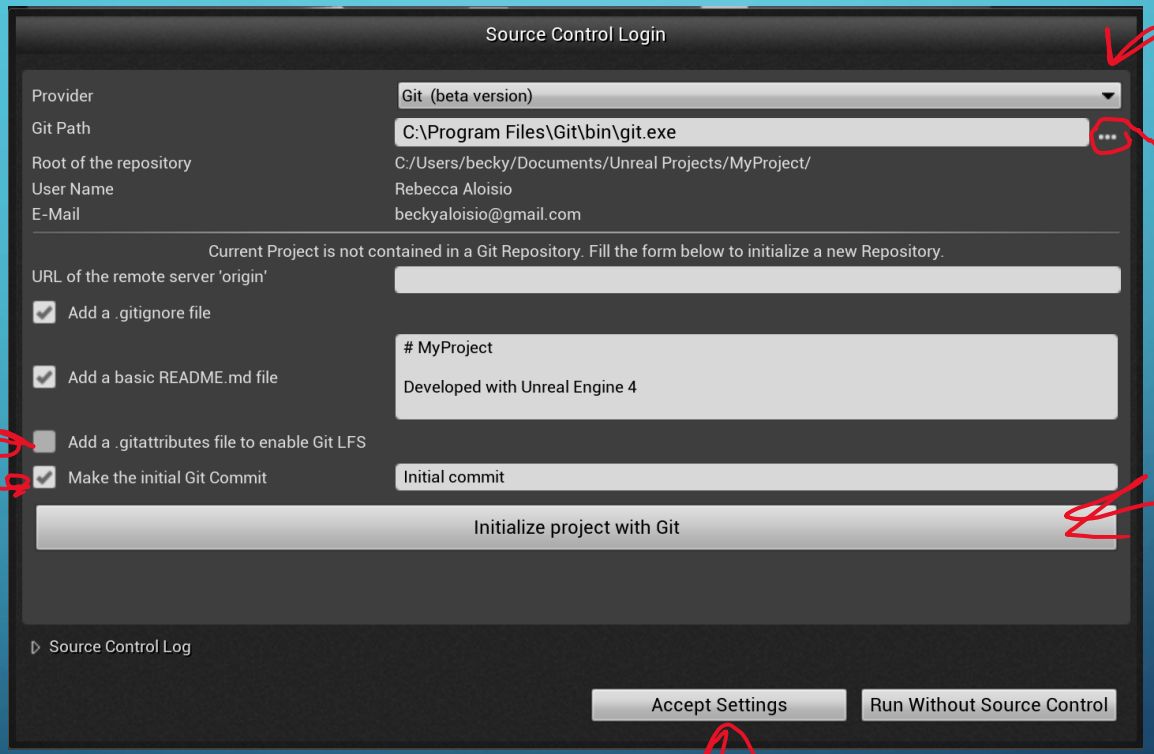
- For any project, fresh or established, open the project and:





FIRST TIME SET UP – PART 2

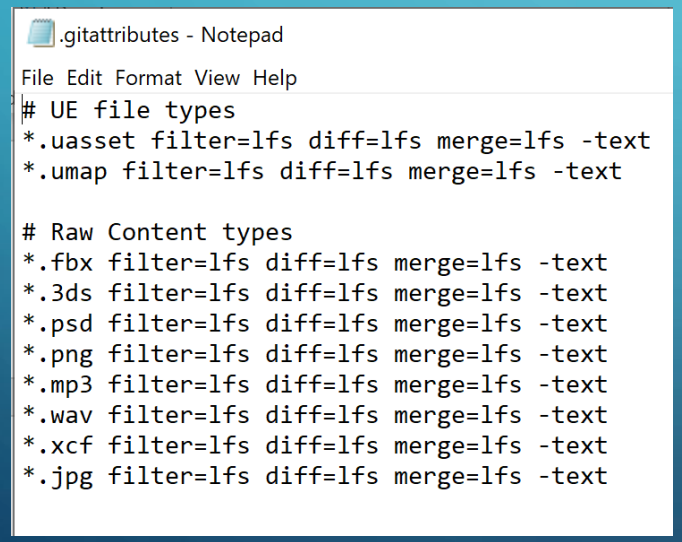
- Make sure this is clicked.
- Uncheck this.



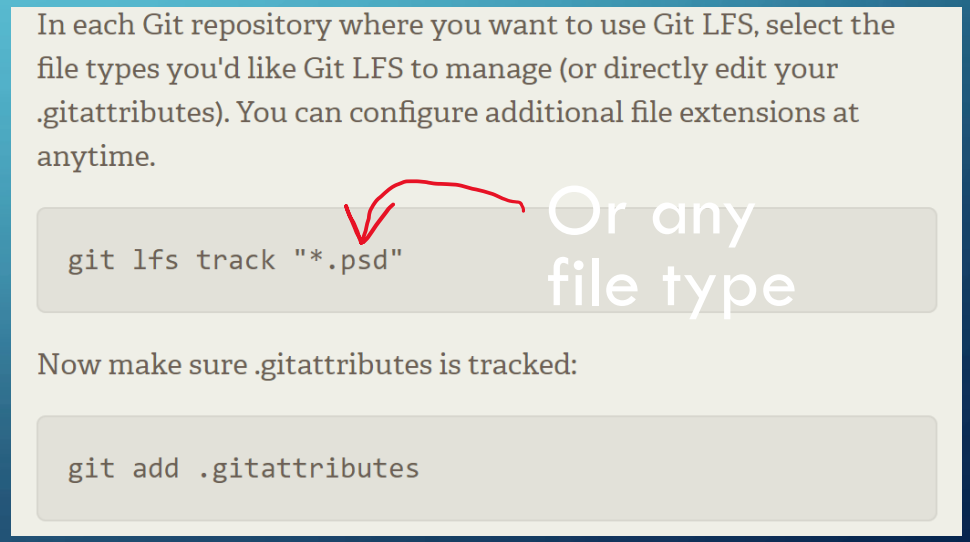
- If this is not filled, you need to navigate to your installation of git.
- Click this
- Then this when done.

FIRST TIME SET UP – PART 3

- The ".gitattributes" file contains the settings for Git LFS for a particular project.
- If you want the simplest set-up, track the files in the text editor example.
- There are two ways to modify the file:
- Edit .gitattributes with a text editor.



- Using console commands:

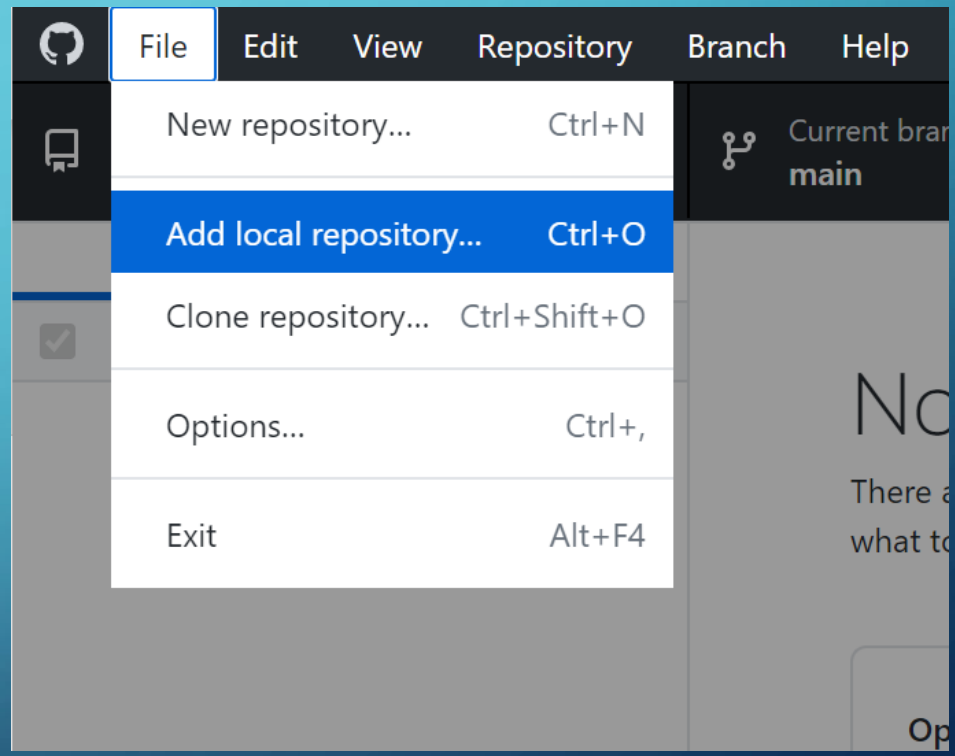


WHY MODIFY?

- By default, Unreal will set the project to add everything in “Content” to Git LFS.
- That means unnecessary files are using precious Git LFS space.
  - For greater control of tracking (and storage use), set it to track individual files rather than files of a specific type.

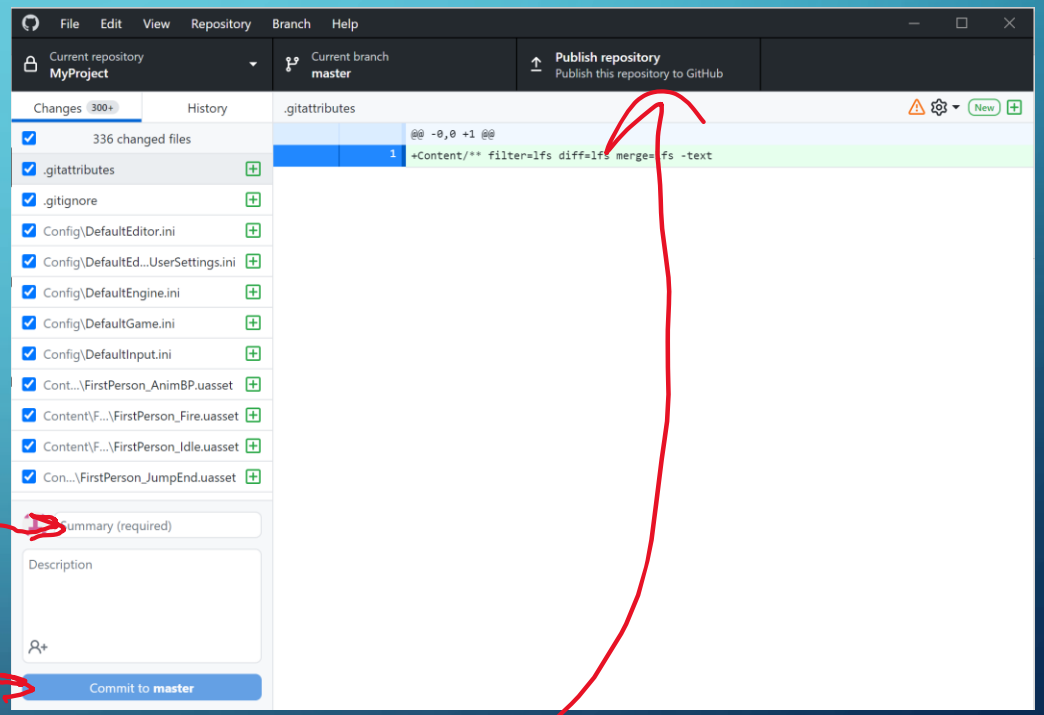
FIRST TIME SET UP – PART 4

- Save and close out of the project.
- Open GitHub Desktop
- Click “File->Add a local repository”
- Navigate to the game project folder



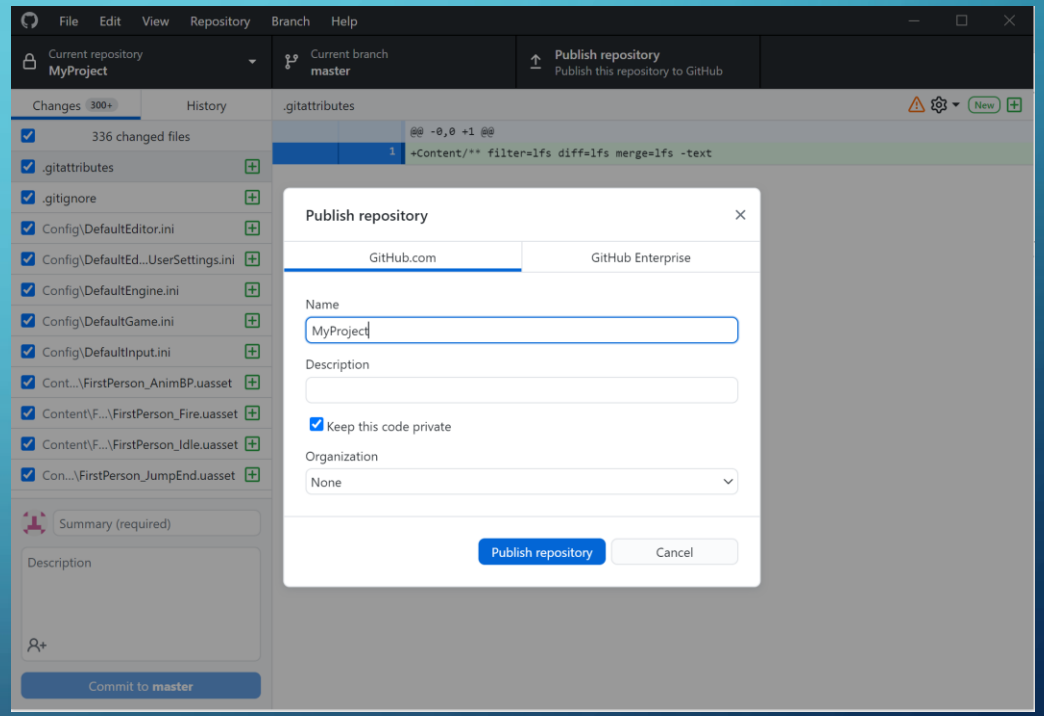
FIRST TIME SET UP – PART 5

- Uploading your project to GitHub.
- You should see something like the side image.
- Traditionally, the first commit is called “Initial commit” so type that here
- Then click the “Commit to…” button.
- Then click the “Publish repository” button.



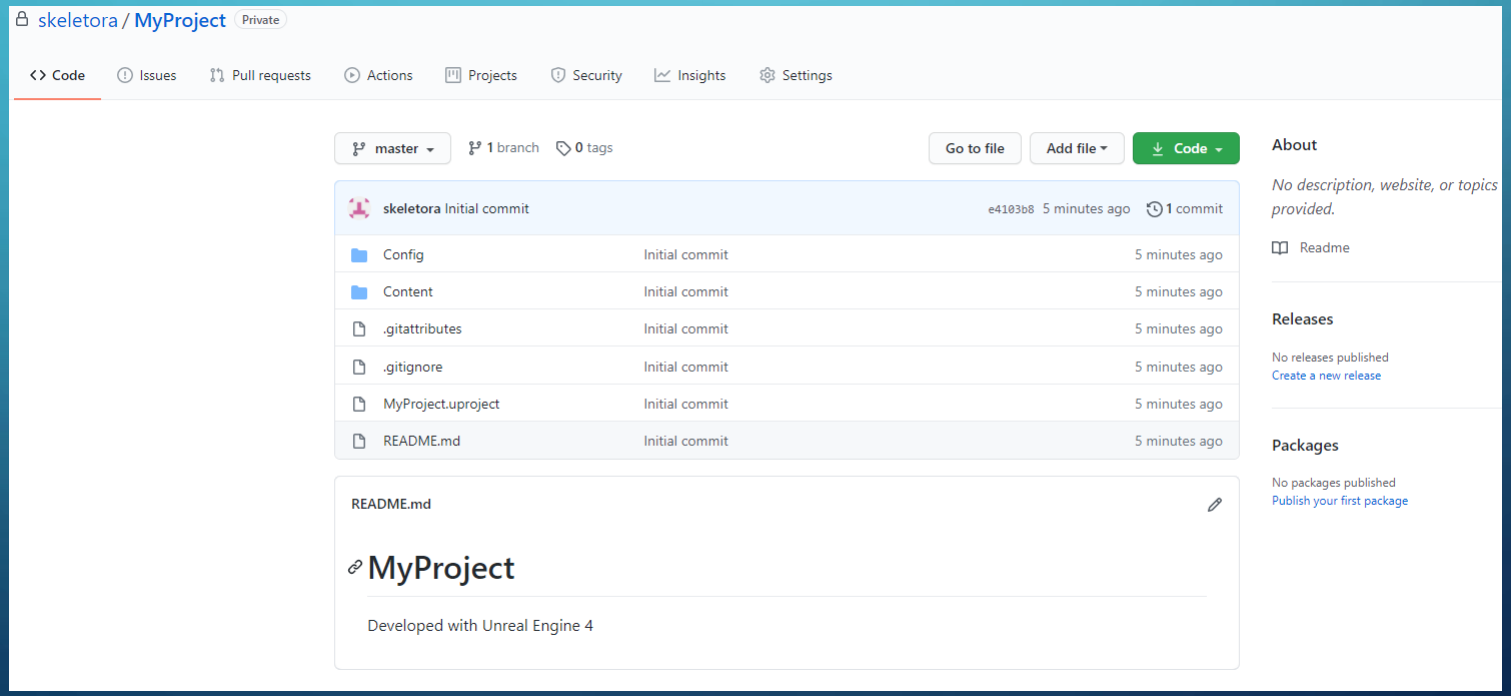
FIRST TIME SET UP – PART 6

- You'll get this pop-up.
- Leave the "Name" section alone.
- "Description" is what will show up in the "About" page of the repository.
- "Keep this code private" will determine if you have a public or private repository.
  - Look at GitHub's policy on private repositories before using them.
- Click "Publish repository"



FIRST TIME SET UP - FINAL

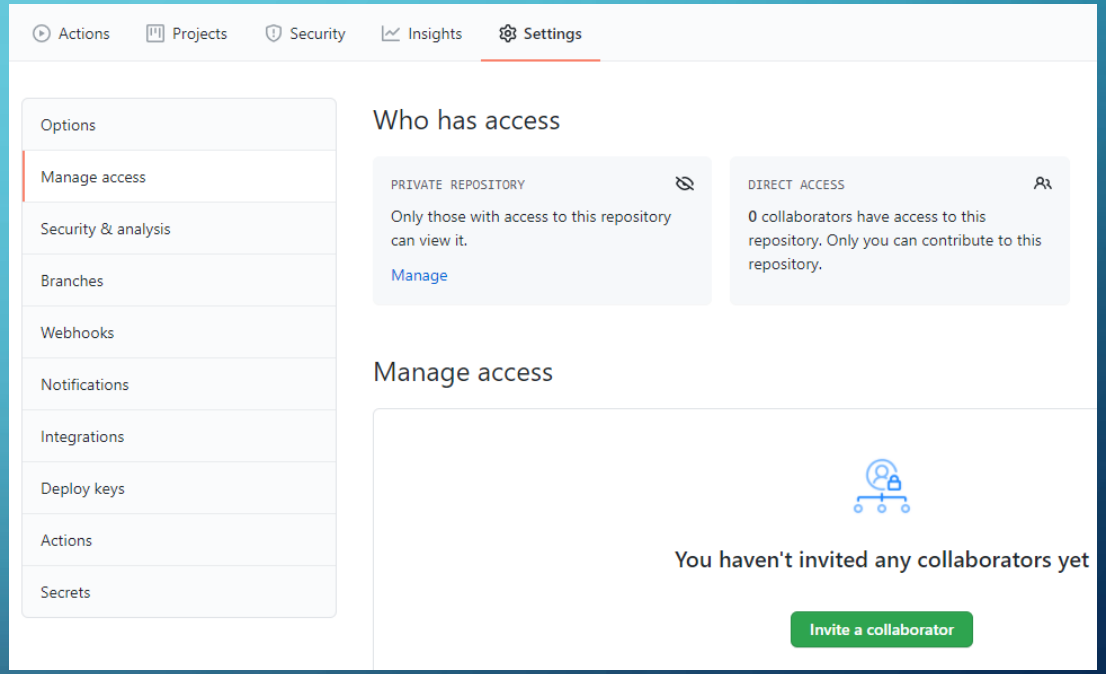
- You're done!
- Once done uploading, go to https://github.com/ to check that it uploaded okay.



OTHERS ACCESSING THE PROJECT

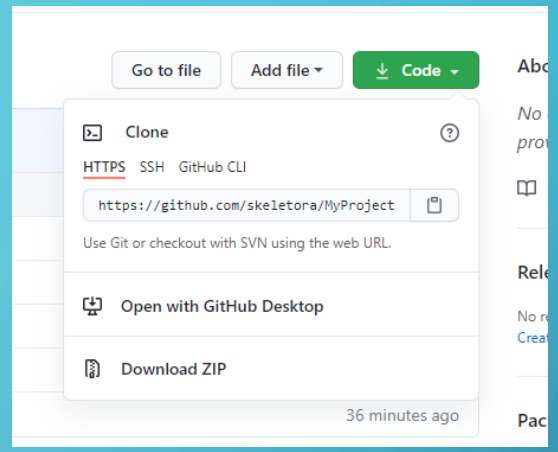
ADD TEAMMATES AS COLLABORATORS

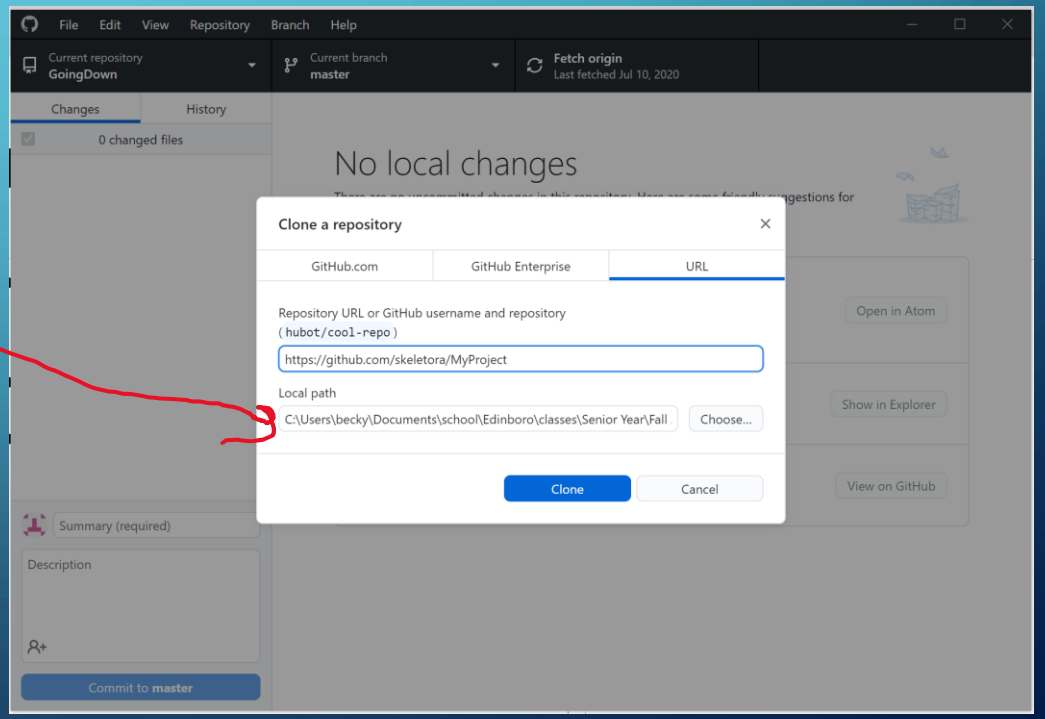
- Only collaborators can modify a project.
- Open the project repository.
- Go to “Settings”
- Click “Manage access”
- Click “Invite a collaborator”
- Enter their username.
- They will receive an email invite.



DOWNLOADING THE PROJECT

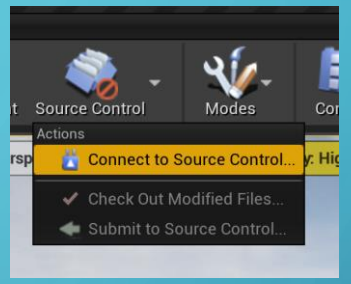
- Navigate to the project repository on GitHub.
- Click the green “Code” button.
- Click “Open with GitHub Desktop”
  - Give the browser permission.
- Set the file location to download the project to here
  - It will create a project folder there.
- Click “Clone”
- Agree to initialize Git LFS.

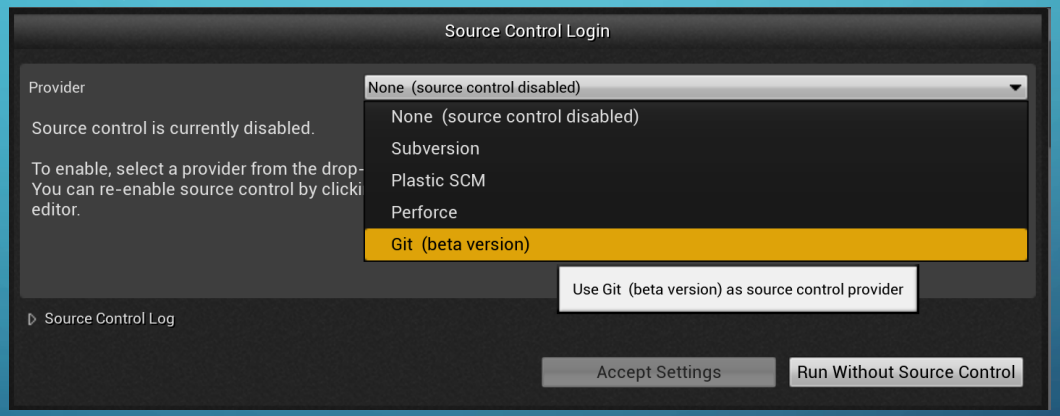




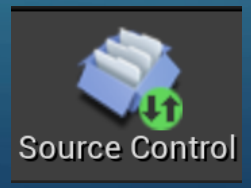
CONNECT TO SOURCE CONTROL

- Open the downloaded project in Unreal Engine.
- Click “Source Control->Connect to Source Control…”
- Select “Git (beta version)” under “Provider”





- Make sure it has detected your git installation (just like in the set-up section) and click “Accept Settings”



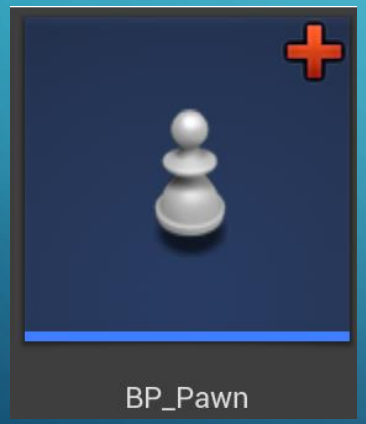
WHAT NOW?

SHARING CHANGES

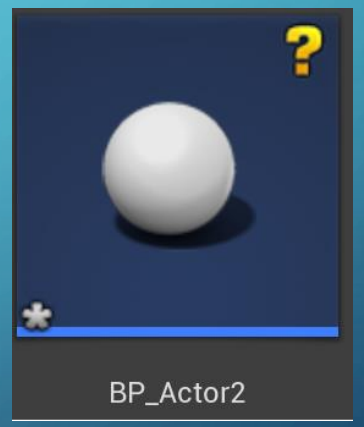
- Project modifications on your machine are not automatically shared with team.
- When ready to upload to GitHub, you have two options:
  - Unreal Engine
  - GitHub Desktop
- I suggest using the Unreal Engine method if the project is still open.

SHARING CHANGES – UNREAL ENGINE PT. 1

- Changed items should have one of the following symbols in the top right corner:



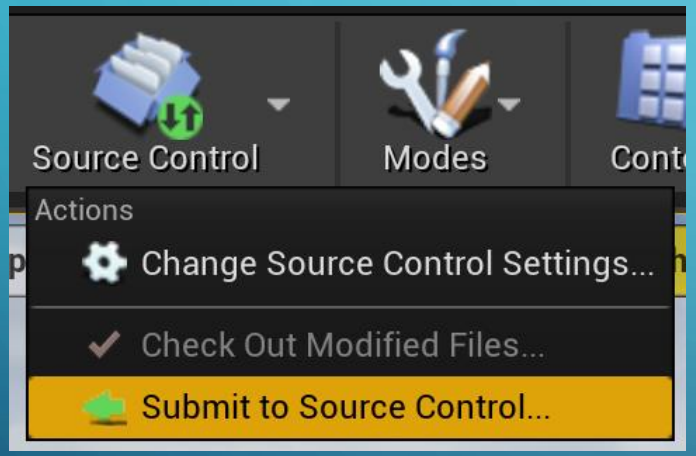
- If they don't, they won't be uploaded. They might have this:



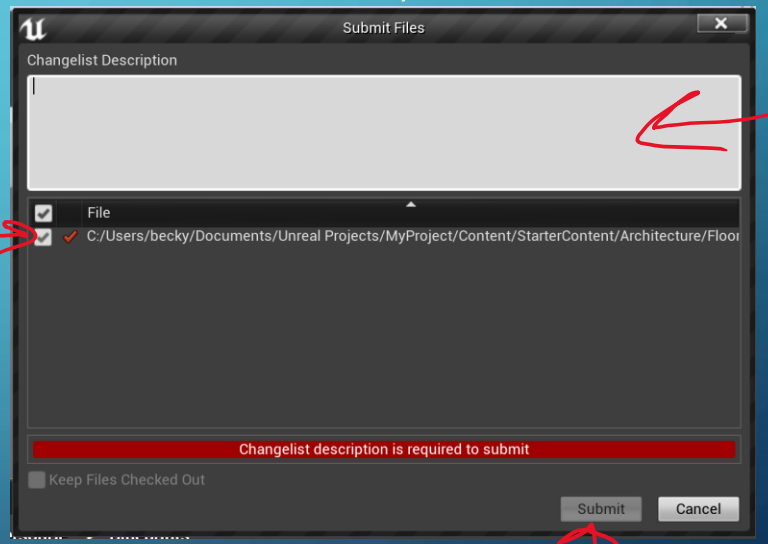
- If that is the case, then save it.

SHARING CHANGES – UNREAL ENGINE PT. 2

- Click “Source Control->Submit to Source Control…”



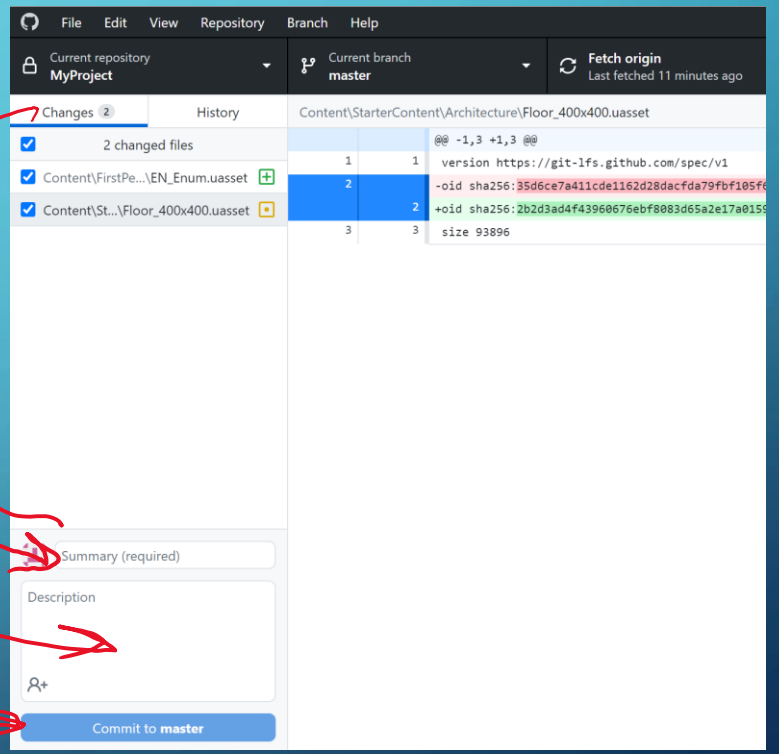
- Enter a description for the commit in “Changelist Description”
- Check the files included. Uncheck if you don’t want to include in commit.



- Click “Submit” when done.

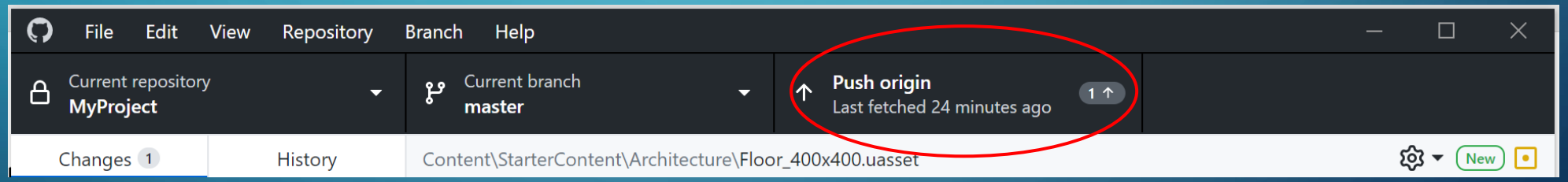
SHARING CHANGES – GITHUB DESKTOP

- Everything that has been changed since the last commit is listed here
  - Make sure only the things you want to include are checked.
- Type a short descriptor for the commit here
- You can give a more detailed description here
- Click “Commit to…” when done.



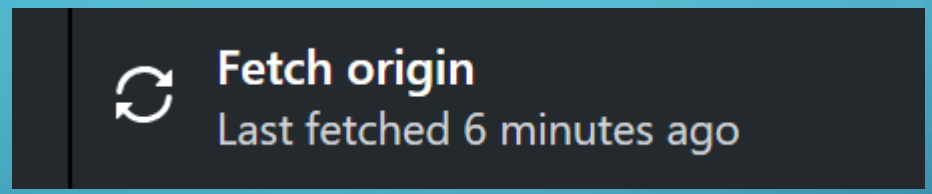
UPLOADING THOSE CHANGES

- You need to upload the changes to GitHub.
- Open GitHub Desktop (previous method doesn't matter)
- Click "Push to Origin"



CHECKING FOR & DOWNLOADING UPDATES

- To check for changes, open GitHub Desktop.
- Click “Fetch origin”



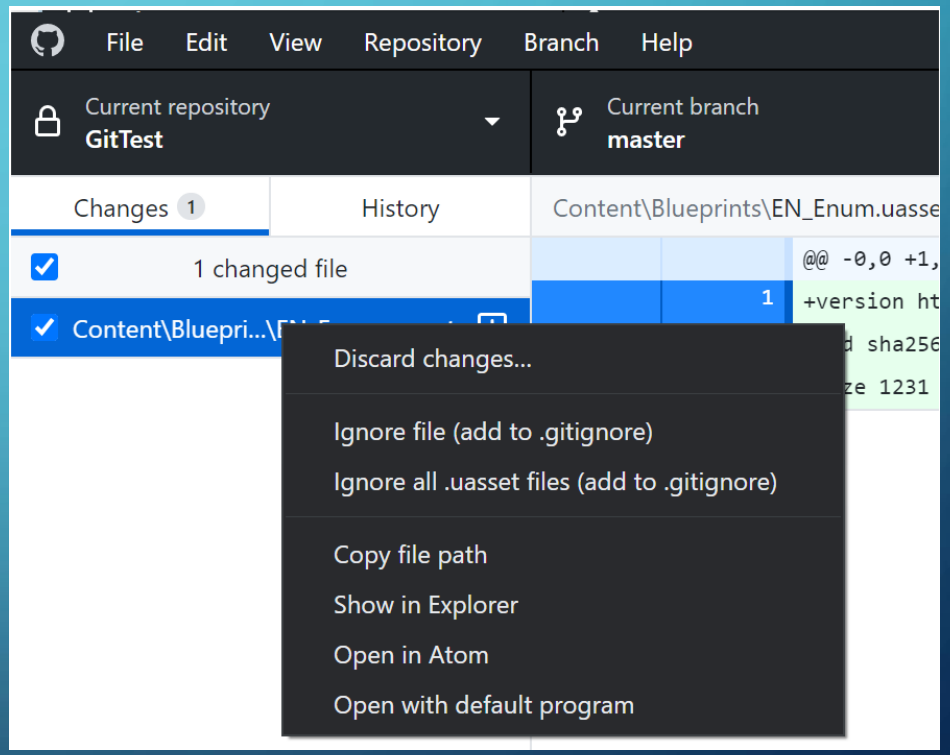
- If there is a change, the button will change to say “Pull origin”
- Click the button to download it.

A WORD OF CAUTION

- Carefully delegate work.
  - If two people are modifying the same portion of the project, you may run into a conflict that can't be merged. Someone will have to try and resolve that before a merge can go through.
- Levels are a pain
  - Levels have the greatest tendency to show up in your changes (and therefore commits) when all you may have done is move the camera. Be careful to not include them if you didn't want them or you may run into a conflict.

A FIX THAT SOMETIMES WORKS

- If changes are popping up that you didn't do, there is a way to discard them.
- CLOSE OUT OF THE PROJECT.
- Open GitHub Desktop
- Right-click on the offending object
- Click "Discard changes…"
- This can prevent future conflicts.

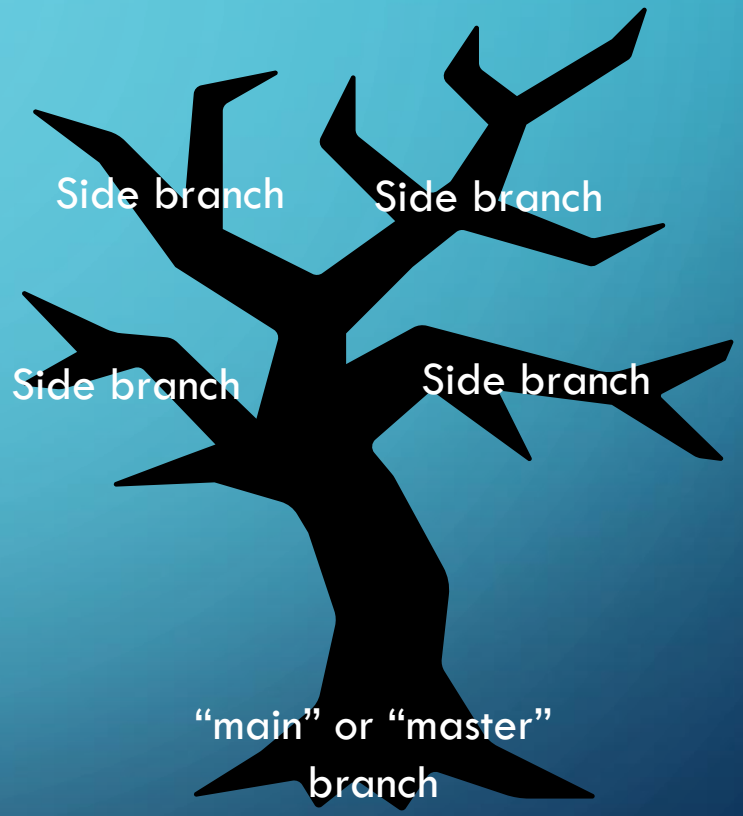


ADVANCED TOPICS

PRIMARILY RELATING TO GIT AND GITHUB

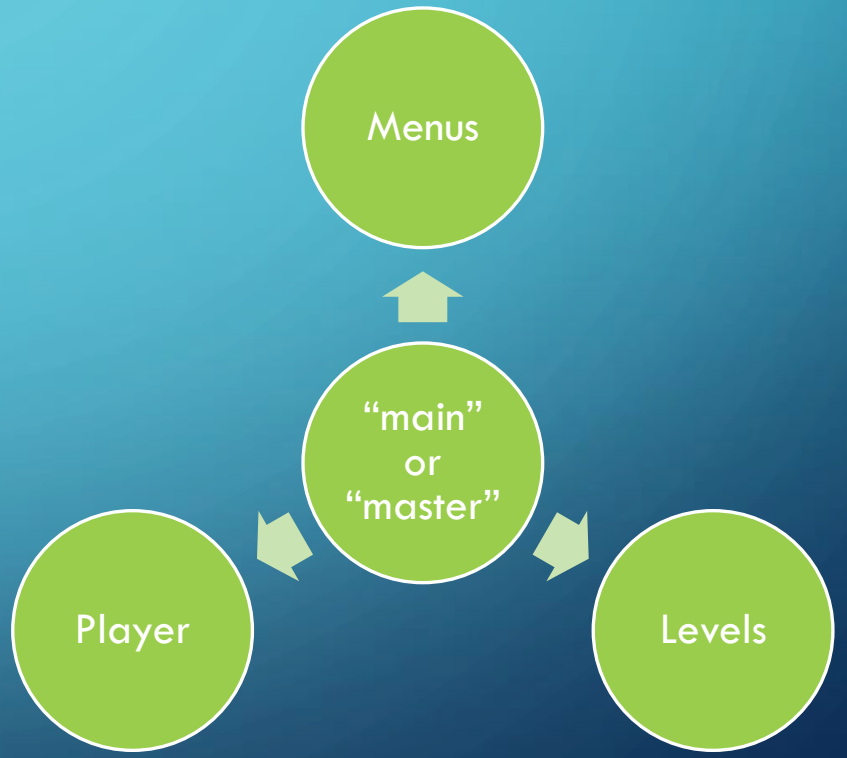
BRANCHES – PT. 1 – WHAT ARE THEY?

- Branches are a way divide a project up to safely test and implement features.
- You have your “main” or “master” branch.
  - This is the tree trunk of the project that all other branches shoot off from.
- Then you have custom side branches.



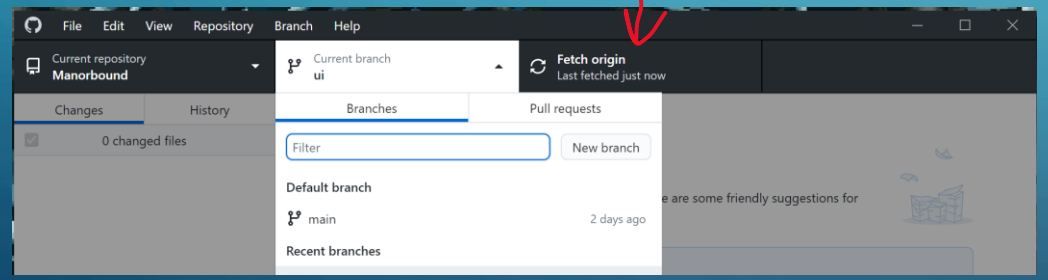
BRANCHES – PT. 2 – HOW THEY CONNECT

- Functionally, branches are like separate copies of your project.
- Creating a branch from another branch (i.e. “main”) makes the new branch a copy of the previous.
- Branches cannot access anything exclusive to another branch.
  - i.e. branch “menus” has the main menu. The branch “player” doesn’t so it cannot access the main menu.

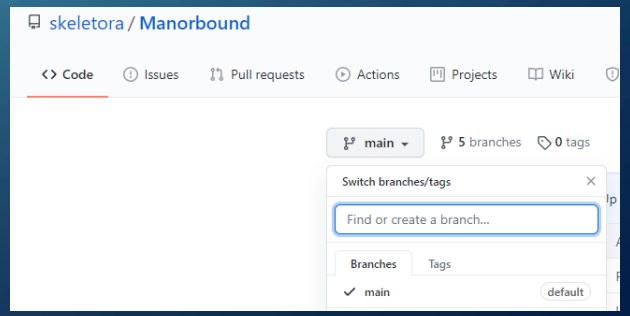


BRANCHES – PT. 3 – MAKING A BRANCH

- GitHub Desktop
  - Click “Current branch”
  - Type the name in “Filter”
  - Click “New branch”
  - Click “Publish branch” (it will change)



- GitHub
  - Under “Code” click “main” or “master”
    - Note: this may have a different branch name. Make sure it says “main” or “master”
  - Type the name of the branch
  - It should now be listed on GitHub Desktop



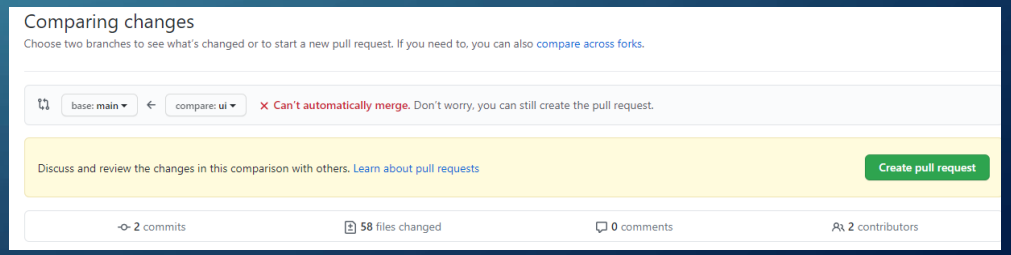
BRANCHES – PT. 4 – THEIR USE

- The “main” or “master” branch should be treated as the current working version of the project.
- All other branches are used for implementing and testing new features before adding them to “main” or “master.”
  - Ex. Testing out a double jump for the player class before adding it to the “main” or “master” branch.
- What do you do when you are done implementing and testing on a branch? Afterall, branches can’t access each other.
  - You submit a “pull request.”
- This can be done with GitHub or GitHub Desktop.

BRANCHES – FINAL - MERGING

- GitHub Desktop
  - Switch to the branch
    - Click “Current branch” and select the branch
  - Click “Create Pull Request”
  - You’ll be taken to github.com
  - Fill out info
  - Click “Create pull request”
  - Someone will then need to go in an apply the pull request

- GitHub
  - Click the “Pull Requests” tab
  - Click “New pull request”
  - The “compare: <branch>” tab is the branch that is being merged into “base: <branch>”
  - You want a green “Able to merge”
    - If you don’t, you have a conflict.
  - Click “Create pull request” when done.



QUESTIONS?