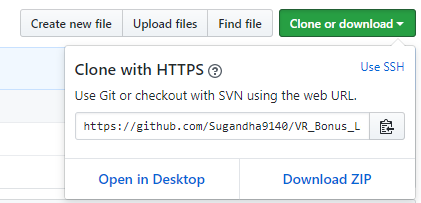
**Basic Guide to Version Controlling this Repository**

**(For Windows)**

**Getting Started**

1. Create an account on Github
2. Download Git. Git is different from Github. You can also download Github as it is has its uses, but what you need most importantly, is Git.
3. Get access to this repository from the creator
4. Go to the homepage of the repository and click on the green button that says "Clone or download" and copy the HTTP url given there. Make sure that you are not copying the SSH url instead by checking if the option on the upper right hand corner of the dialog box says "Use SSH" or "Use HTTP".



1. Open your Command Prompt and check into the directory where you want to clone this repository by typing ***cd*** followed by the path of the location. For example if you want this repository to exist on the Desktop, you would type the command line **cd Desktop** and press enter.
2. Once you are into the directory where you want your repository to live, type the command **git clone** followed by the HTTP url you just copied, so in this case, you would type **git clone *https://github.com/Sugandha9140/VR\_Bonus\_Level.git*** and press enter.
3. You now have the most recent updated version of this repository, with all its contents and you can use this local repository to work on the project, modify code, add files etc and then push it on the master repository existing on github.

**Before Proceeding**

1. Make sure that your cloned repository has a file called ‘unity.gitignore’ and NEVER delete or untrack this file. This file ensures that all sorts of extra files that are generated every time you work on this project are not tracked, added or committed by git and is EXTREMELY IMPORTANT while working with Unity especially since Unity generates all sorts of unwanted heavy files. If you have time, it might help to read and understand this gitignore file so that you are aware of what folder and files are not to be tracked and thus should not appear on your cloned repository.
2. When making major changes in a scene, save a copy of the scene and make changes in that instead of modifying the original scene, unless you know how to work on and merge different branches on git effectively, in which case you don’t need to read anything in this document!
3. All scenes and scripts with ‘VR’ in their name have been created for virtual reality and would not work without the VR setup.

**Adding New Files/Folders and Modifying existing ones**

1. When you add any new file or folder in your local repository, you can add it in the master repository by doing the following:

**git pull origin master** (to ensure that you make your commit to the most updated version merged from all working branches and to minimize any merge conflicts)

**cd Name\_Of\_Folder\_Within\_The\_Repository**

**git add NameOfFolder** or **git add NameOfFile.extension**

**git commit -m “Name(s) of things uploaded”**

**git push**

1. If you have a lot of files/folders to upload within a common folder in the repository, you can also do **git add .** and then do **git commit** and **git push**. The ‘**.**’in any git command refers to all files and folders.
2. If you have staged something to be added by doing **git add** but don’t want to add it anymore, you can do **git reset filename.extenstion** to remove it from the stage before doing committing the changes. You can do **git reset .** to reset all the changes you have made after your last commit.
3. If you are only modifying existing files, then you can skip **git add** and simply commit. However, make sure that you are on the correct directory within your repository. Also, it doesn’t hurt to do **git add** again, just because often the files/folders that you are trying to commit might be untracked by git (because of the gitignore file), so they won’t commit.