**GitHub On-boarding**

When I first used Git, I felt for a while I was walking on egg shells. It took me a long time to have any sort of confidence about what I was doing. Please don’t be afraid to say you don’t understand something or ask me for help; I will do my best and I completely sympathise. However, the idea of Git is that you an repair almost any mistake, so don’t be too afraid of it.

**Create an account:**

* Create an account with GitHub if you don’t already have one. I would advise against using your uni email address as you may lose access to the account after your graduate. It’s better to use another email (I used an iCloud throwaway) as your primary one.

**Accessing Education Programme (free Pro for 2 years)**

* If you don’t want to access the free Pro student programme, then you can skip the rest and go to “**Joining the Repository**”. I have gone through the verification process, so you will be able to utilise the benefits through our repository anyway (but not through others unless you get on the programme).
* A note on the application process: at some point during the application, I had to allow GitHub to access my current location to verify it. If you are not comfortable with this step then an option would be to complete the application on one of the uni computers instead.
* Once you are logged into the GitHub website; click profile photo. Go to **Settings** > **Billing and Licensing** > **Payment Information** (you won’t need to input any card details). Fill out your full name and address as per the university records. Leave the card details blank and then **Save**.
* Click profile photo. Go to **Settings** > **Emails** and then scroll down to “**Add email address**”. Input your university email here and it will add it as a secondary email address. Go to your university email inbox and follow the link GitHub emailed to verify it.
* The Education programme require 2FA to be enabled. You can’t use OTP via SMS as they consider it insecure. The other options are GitHub mobile app, or an Authenticator. Since you have to use an Authenticator to then enable 2FA through GitHub mobile app, I suggest to just use the Authenticator app (message me if you need help with one).
* Click profile photo. Go to **Settings** > **Password and Authentication**. Scroll down to “**Two-factor authentication**”. Click “**Enable two-factor authentication**”. Under “**Scan the QR code**” either scan it using your Authenticator or if you can’t, click “setup key” link to get a code to manually input into the Authenticator. Copy the TOTP it generates and navigate back to the GitHub page and type the code in the field under “**Verify the code from the app**”.
* Under “**Save your recovery codes**”, click **Download** and save them somewhere safe (not your phone – they’re necessary if you lose your phone to regain access). Click “**I have saved my recovery codes**”.
* Click profile photo. Go to **Edit profile.** Add some name, details, bio, picture, whatever you like. This I believe is some kind of extra layer of verification to ensure you’re not a bot. Either way, it’s required to access the programme.
* Navigate to Bournemouth University website and log in to myHub.
* Go down to **My Documents** and click “**Student status letter**”. Type the address you used on GitHub exactly, and then **Generate letter.**
* Take a screenshot of the .pdf. I can’t access the GitHub application page anymore because I am already approved. Some of this part is therefore from memory but message me if you get stuck anywhere. That said, I can’t remember the exact dimensions this screenshot needs to be, but GitHub was very fussy with me. In the end I had to rotate it 90’deg and save it landscape, and then it accepted it. If you get issues uploading later, try that.
* Make sure your browser can access your location details (at least for now, you can turn it off later). Temporarily disable any VPN. It’s necessary to prove your location to GitHub if asked. If you aren’t comfortable with this step then an option would be to complete the application on one of the uni computers.
* Go to **GitHub.com/education**. Make sure you are logged in. Scroll down to “**Students**” on the main page and click “**Learn more**”. On the next page, click “**Join GitHub Education”** button.
* Under **“Select your role in education”**, select **Student**.
* In “**Application”** dialogue, enter “**Bournemouth University**”.
* If required, select or add your uni email address.
* Click **Continue**, and you will be prompted to upload proof of your academic status. Find the screenshot taken earlier, and upload it.
* Verify the details you’ve given and then click “**Process my application**”. Don’t navigate away until it confirms your application was processed.
* Mine took 2 days to authenticate, but apparently this can take up to a week. You can use GitHub in the meantime anyway while you wait for authentication (just minus Pro benefits). Please note that although it said you’ll get an email, I didn’t; I only noticed on checking back. It is also worth mentioning that once approved, it still takes a few days longer for the benefits to kick in.

**Please message me your email/username**

* Once you have created an account, please message me your account email and username so I can add you as a collaborator on the repository.
* I’ll let you know when I’ve done it, and you should have received an email to the inbox you signed up using, with an invite link to join.
* Once you have joined, please go to **Trello** > **Weekly Meeting 1** > “**On board the team members to the GitHub repository”** and then find your name in the checklist and check it off.

**Git/GitHub**

* **This is an important introduction. If you haven’t got time, skip this section. But please do read it at some point.**
* Git is a version control system that allows developers to track changes during development. It helps in coordinating work among multiple people, when everyone’s working on the same files.
* GitHub, on the other hand, is a web-based platform built on top of Git (and it’s not the only one, but it’s the most popular). **Git is not GitHub**. GitHub provides an interface, a place to track and report issues and bugs, and assign people to fix them, and kee[ track of who contributed what.
* GitHub allows developers to host their Git repositories online (a repository can be thought of like a Dropbox folder that you share with others and everyone can make changes), making it easier to collaborate with others.
* Since Git is more of a “language”, it works through the command line. But GitHub Desktop provides an application interface to manage that for you (or if you really prefer you can manage a repository using solely the command line with an emulation like GitBash, ask me if you want help with that).
* The last thing to know is that every historical version of every file is stored on the repository, not just the latest version. This is fine for 1kb code files, but can balloon the storage massively when editing heavy art assets. **10Gb is all we get**, so unfortunately that means we have to implement an LFS (“large file storage”) server to store the large files outside our repository. GitHub provides a free one with **1Gb** storage, but the Pro version (which we will be able to utilise through the Education prorgramme) allows for a further **10Gb storage** with **10Gb** monthly bandwidth. The LFS stores tiny pointers to the separate server instead of hosting the large files itself, and downloads large assets on the fly when required by our repository. **It is set up and works out-the-box so you don’t need to do anything special to make it manage large files properly.** However the reason I am going to pains to explain it is because using Git LFS is pretty much mandatory for Unity projects and will be very helpful knowledge for the future.

**A Quick Privacy Note**

* Your contributions to any repository are signed with your primary email. Many repositories are public so, anyone in theory can see it. **If you don’t care about that, skip this section.**
* I will keep our repository **private for the duration of our development cycle**. However, after the semester is over, I will make it a public repository. Some people will surely want to use the repository as part of their portfolio, and potential employers wouldn’t be able to see the project/contributions otherwise.
* I am aware this might infringe your privacy if you prefer your email not to be associated with some particular project in public. If you are concerned, please go to GitHub website. Click profile photo. Go to **Settings > Emails.** Go to **Keep my email address private**. Check the box. It will sign it with your username slapped with a bunch of numbers, instead of your email address. If your username contains your name and you’re not comfortable your name on public repos, you can always change that too by editing your profile.
* Message me if you have a concern about it and I’ll help.

**Download GitHub Desktop**

* With all that said, download the Desktop application from GitHub website and install it. Please try to use **version 3.1.2.** Please also try to use **Windows OS** (sorry to any Mac users). If you absolutely have **no other choice but to use MacOS,** please let me know so that I can do my best to sort out the repo to handle it. It will make a lot of work and might introduce unforeseen bugs, so I’d rather you didn’t *choose MacOS* if you have other options available to you. (MacOS itself works with Git; however using different OSs to edit the repo can cause painful problems due to differing encoding formats used by them. The Labs PCs are Windows so I went that direction).
* Sign in using your credentials.

**Clone the repository**

* Go to **File > Clone Repository …** and then you should see our repo, “GSP-Unity-Project”. Select that one.
* You should clone the repository somewhere completely out of the scope of any other version control or cloud storage. OneDrive, GoogleDrive, Dropbox etc all can cause serious headaches because sometimes the way they handle syncing files can cause Git to think that changes have been made when they haven’t, and can lead to painful merges later down the line. **A “go to” place people often use is C: drive or D: drive.** Directly on the drive you can greate a folder called “Git Repositories”. You can store any/all repos there on your local machine.
* You should have something like this (and if so, you can **Clone**):

A screenshot of a computer

AI-generated content may be incorrect.

**Configure UnityYAMLMerge**

* Now, navigate to the repo’ you just cloned. We need to configure Git to use Unity’s in-built YAML merge for better handling of Unity projects by Git (it’s quick and simple). Go to the folder file path, and type the highlighted below onto the end:

A screenshot of a computer

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

* This should open up a hidden folder. Open the file “**config**” in a text editor (right click, **Open with…,** scroll to Notepad).
* Go back to the Github channel on Discord and download the little text file, “**addto--config--replace path**”. Now copy the text in there, and paste it in the “**config**” file. You should have something like this:

A screenshot of a computer

AI-generated content may be incorrect.

* Navigate to your Unity installation. You’re looking for the UnityYAMLMerge.exe. It should be somewhere around here, in the Tools folder:

**A screenshot of a computer

AI-generated content may be incorrect.**

* Once you’ve done that, copy the file path (you’ll have to right click instead of Ctrl+C). Go to the “**config**” file we opened in the text editor. Highlight the below path exactly, and paste over it exactly with your own path.

**A screenshot of a computer program

AI-generated content may be incorrect.**

* **Then, you must make sure to replace the back-slashes in the text you pasted (from “C:” to “Tools”) with forward-slashes again. Ignore the back-slashes later in this line after the “-p”, they are supposed to be this way.** Save the file.

**A screen shot of a computer

AI-generated content may be incorrect.**

**Cloning the Repo and got error: “unable to Initialise Git LFS” (or similar)**

* Initialisation should happen automatically, please message me if you have issues (that ChatGPT can’t solve 😉)
* Should never be an issue on the Lab PCs; Git LFS is preconfigured on them

**Done for now, but really important note about home/Lab PCs:**

* The repo is now available locally and configured to push your changes. Please leave everything else in the folder alone for now. 😊 You are safe to close GitHub Desktop and there’s nothing more to do for now.
* If you cloned this on your home machine, you’re set. If in future you clone it on a lab PC (as I know some programs are very resource intensive and people work better there), then there is an extra step to take: when you have **cloned** the repo, made some changes (changed art assets for example), **committed** your changes, **pushed** them to the remote repo, then when you are finished, you really should go to **Repository > Remove…** in GitHub Desktop, then select our repo(s), and remove them, and **please** **tick the box** that says “**Send to Recycle Bin”**). This is because: after you push, the remote repo has all the necessary work you did (i.e., you stored it on Git’s server). You can just **pull** those changes to your local machine when you are home. You can check on the GitHub website and look at the file list of our repo if you are unsure before you delete the repo from the uni Lab PC. Some of the terminology might not be familiar right now, but hopefully you can grasp the meaning.
* Last and final note: please check you are using the right version of any programs we’re using to share work. This will avoid some serious headaches. 😊