

Workshop exercise: Clone, commit, pull, push

Getting started

Step 1: Get yourself on Github! Go to <https://github.com/> and make yourself an account, if you don't already have one.

Step 2: Click **HERE** and leave a comment on the issue (include your name so I know who you are, please). This is just so I know your Github username and I can add you as a collaborator. This is important - because even though you can see all the code right now, you won't be able to push any of your changes until you've been added.

Setting up the git user

Before we can start using git, we have to tell it our name and email address. Weirdly, this isn't actually related to your Github account - it's just an identifier that it will put next to your commits (you know, so your friends know who to blame when the project breaks!).

Open Git Bash if you haven't already. This will give us a command line interface - it's a text-based way of navigating our folders and running programs. To run a command, you type it and then press enter.

To set the user name, use this command:

```
git config --global user.name "Replace this text with your name"
```

Don't forget to replace the text in quotes with your actual name! For example, I would put:

```
git config --global user.name "Delia Hamwood"
```

Then set the email address. This can be any address you use, it doesn't have to match your Github account:

```
git config --global user.email replacethis@example.com
```

Replacing the fake email with your real one. Mine would look like this:

```
git config --global user.email d.hamwood@lcc.arts.ac.uk
```

Cloning the project

Now git knows who we are, we need to navigate to the folder we want to put the documents in. Let's put this one in our Documents folder. Type this and press enter:

```
cd Documents
```

Now you should be in the Documents folder - we're ready to to clone the project. Here's the command:

```
git clone https://github.com/creativetechnologylab/LCCUnityGitWorkshop.git
```

Wondering where I got that URL? There's a button on the project page for "clone or download" which gives you the link - you can get the clone URL for any project that's publicly available on Github, even if you aren't a collaborator!

A screenshot of a GitHub repository page. At the top, it says "A test project for students to work in to learn how to use git with unity" with an "Edit" button. Below this is a "Manage topics" link. The repository statistics show 3 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. There are buttons for "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". A dropdown menu is open for "Clone or download", showing options for "Clone with HTTPS" (with a "Use SSH" link), "Use Git or checkout with SVN using the web URL.", the URL "https://github.com/creativetechnology", "Open in Desktop", and "Download ZIP". Below the dropdown, there is a table of files: "Assets" (added some objects to the scene), "Packages" (initial commit), "ProjectSettings" (added some objects to the scene), and ".gitignore" (Created ignore). At the bottom, there is a section "Help people interested in this repository understand your project by adding a README." with an "Add a README" button.

Making changes and committing them to the local repository

Now you can open the project like normal in Unity and make some edits! It's pretty empty at the moment - maybe create some scripts or assets. But remember, you should only work on the Main scene file if you have the merge token.

Once you've changed some files, you can use these commands.

`git status` will show you all the changes you've made since the last commit. Filenames in green will be saved as a version in the local repository if you make a commit. Files in red have changed, but won't be sent to the repository unless you add them. If you're happy with your changes, the command `git add -A` will "stage" them all, ready to be committed to your local repository.

Then you're ready to commit!

```
git commit -m "Replace this text with a descriptive commit message that explains what files you have changed"
```

You always have to put -m followed by a message in quotes. Git won't let you commit without sending a message. Try to make it something sensible that explains what work you've done. This helps your collaborators, but most of all it helps future you.

Syncing up with the remote repository

You can do this any time, but remember with lots of people working it could get messy. If you've only added your own files (and not edited anyone else's), you can go ahead and do this. Or if you've changed the Main scene and it's your turn with the merge token, go ahead.

You need to pull down other changes first. If somebody else has made a change, git won't let you push yours until you're up to date.

`git pull` will fetch and merge anyone else's changes that are already online. If you get a message about a merge conflict, ask for help! Otherwise it should be safe to do this:

`git push`

And this will send your changes up to the remote repository.

You can have a look at the project commit history here to see all the changes we've made:

<https://github.com/creativetechologylab/LCCUnityGitWorkshop/commits/master>

Additional resources:

- Tutorials and cheat sheets from Github: **<https://try.github.io/>**
- Download and install git for your laptop or home PC: **[Git downloads page](#)**
- GUI applications for git:
 - **[Fork](#)**
 - **[Github Desktop](#)**

Revision #14

Created 15 February 2018 17:49:49

Updated 20 January 2020 10:59:17