Using Git with Unity

ECS7016P — Interactive Agents & Procedural Generation
Dr Jeremy Gow — jeremy.gow@qmul.ac.uk

https://gameai.eecs.qmul.ac.uk
Queen Mary University of London

Using Git with Unity

- There's **no requirement** to use Git for the coursework, but version control is incredibly useful for any code project.
- These notes are for those **who already know** Git and Unity separately, and would like them to play nicely together.
    - They don't always play nice…
    - But I'd currently recommend Git over other solutions (PlasticSCM or Collaborate)

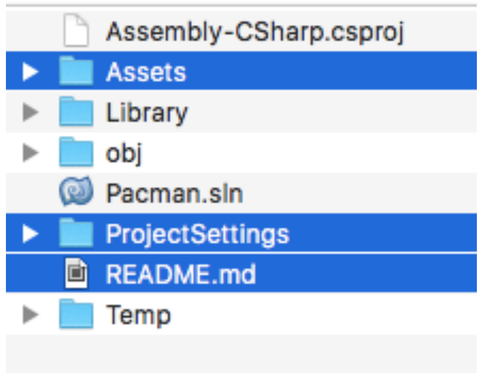
Configuring the Unity Editor

- Configure Unity to use Git-friendly text format
- Edit > Project Settings > Editor
- Set “Version Control Mode” to “Visible Meta Files"
- Set “Asset Serialization Mode” to “Force Text”

https://thoughtbot.com/blog/how-to-git-with-unity

Use a Good .gitignore

- Only Assets and ProjectSettings should be in repo (and the gitignore file)



- Download a good gitignore example from https://github.com/github/gitignore/blob/master/Unity.gitignore

```
mv gitignore .gitignore
```

Filename starts with a DOT

Install Git LFS

- Git Large File Storage
    - Stores **undiffable** large binary files separately
    - https://git-lfs.github.com/
- Add a **.gitattributes** file to your repository (see blog below)

https://thoughtbot.com/blog/how-to-git-with-unity

Steps to Create a Unity Repository

- Create new Unity project
- Create and clone a new hosted repo
- Copy in and add project files
- Add a README.md
- Add a .gitignore
- Commit then push

You Can Merge Scenes

- Scenes and Prefabs are stored in UnityYAML
- Unity provides a Smart Merge Tool for these
    - https://docs.unity3d.com/2020.3/Documentation/Manual/SmartMerge.html
- You should modify .gitconfig as follows…

```
[merge]
tool = unityyamlmerge

[mergetool "unityyamlmerge"]
trustExitCode = false
cmd = '<path to UnityYAMLMerge>' merge -p "$BASE" "$REMOTE" "$LOCAL" "$MERGED"
```

But You Shouldn’t Merge Scenes

(if you can avoid it)

- Smart Merge is useful but still complex to use
- Unity developers tend to avoid the need to **merge Scenes and Prefabs**
    - Can work on new features in separate sandbox scene
    - Later integrate those new features into core scenes
    - In teams, need to coordinate who’s working on what
- This goes against “the Git way” of doing things, but it saves a lot of pain with complex merges.