Guide

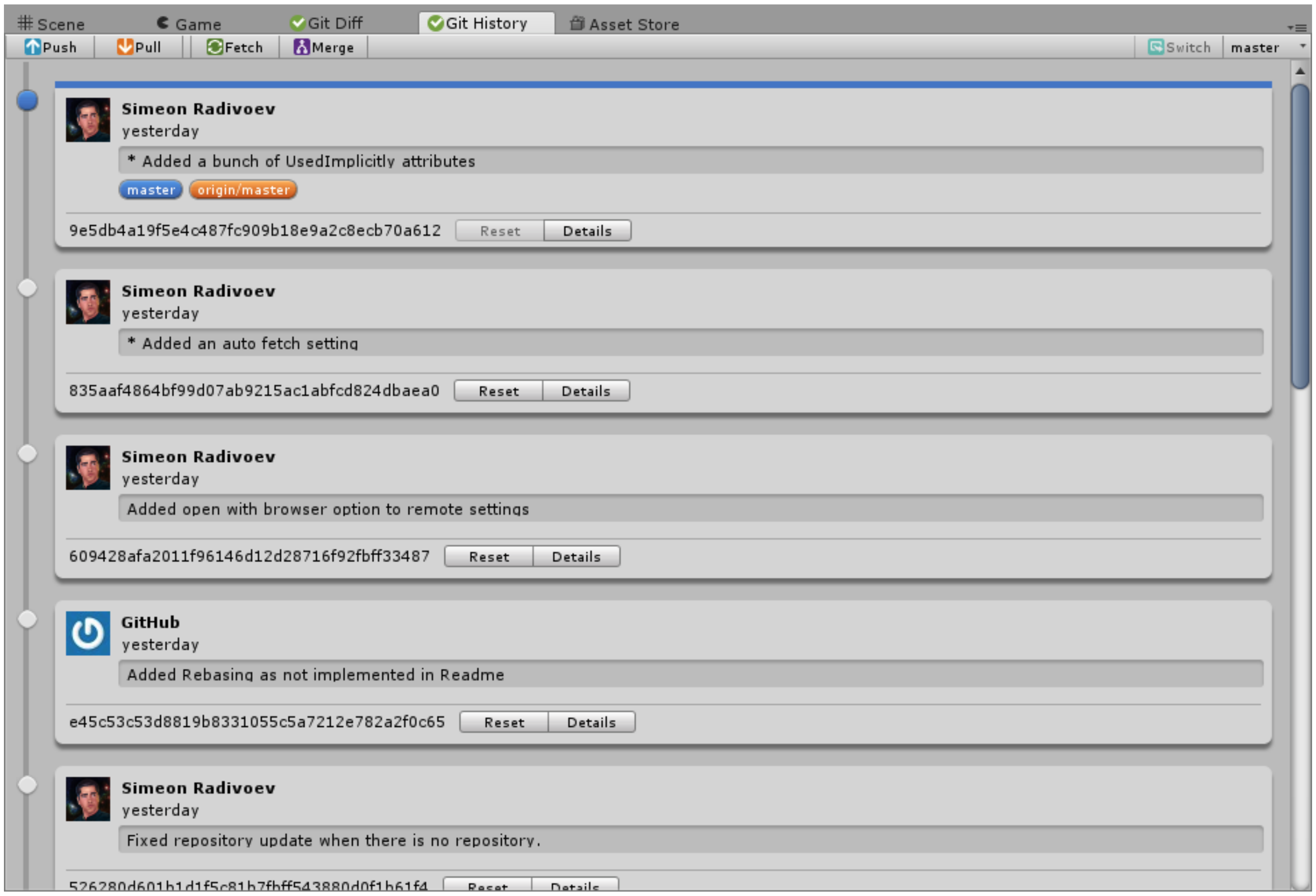
Table of Contents

Commit History....2
File Difference and Committing....3
File Difference....4
    Meta Changes....4
    Stage/Unstage....4
    Differences/Changes....4
    Reverting....4
Committing....4
Configuration....5
Remotes....5
Credentials....5
Git Settings....6
UniGit Settings....6
External Programs....6
Credential Managers....6
Git LFS....7

Features and Usage

Commit History

The commit history shows all the commits on a selected branch. The default selected branch once a repository is created is the 'master' branch. The shown branch can be selected form the dropdown button at the top right corner.



Simeon Radivoev
yesterday
* Added a bunch of UsedImplicitly attributes
master origin/master
9e5db4a19f5e4c487fc909b18e9a2c8ecb70a612 Reset Details

Simeon Radivoev
yesterday
* Added an auto fetch setting
835aaf4864bf99d07ab9215ac1abfcd824dbaea0 Reset Details

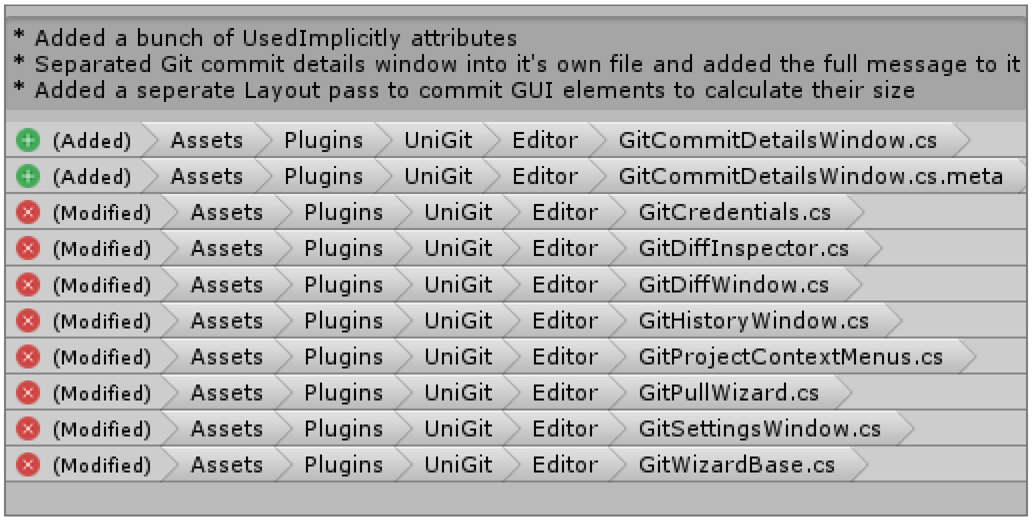
Simeon Radivoev
yesterday
Added open with browser option to remote settings
609428afa2011f96146d12d28716f92fbff33487 Reset Details

GitHub
yesterday
Added Rebasing as not implemented in Readme
e45c53c53d8819b8331055c5a7212e782a2f0c65 Reset Details

Simeon Radivoev
yesterday
Fixed repository update when there is no repository.
526280d601b1d1f5c81b7fbff543880d0f1b61f4 Reset Details

Every commit can be reverted by clicking on it's **reset** button. The full commit message and all the changes made can be viewed by clicking the **details** button.

The Details popup window shows all file changes a commit has made. Information on the type of change made is included as well as the path the each file. The path is divided by folders, and each section can be clicked and the file or folder will be selected in Unity's project browser window. The user can also right click on a file to see the difference between the previous commit or the working tree.


```
* Added a bunch of UsedImplicitly attributes
* Separated Git commit details window into it's own file and added the full message to it
* Added a seperate Layout pass to commit GUI elements to calculate their size
```

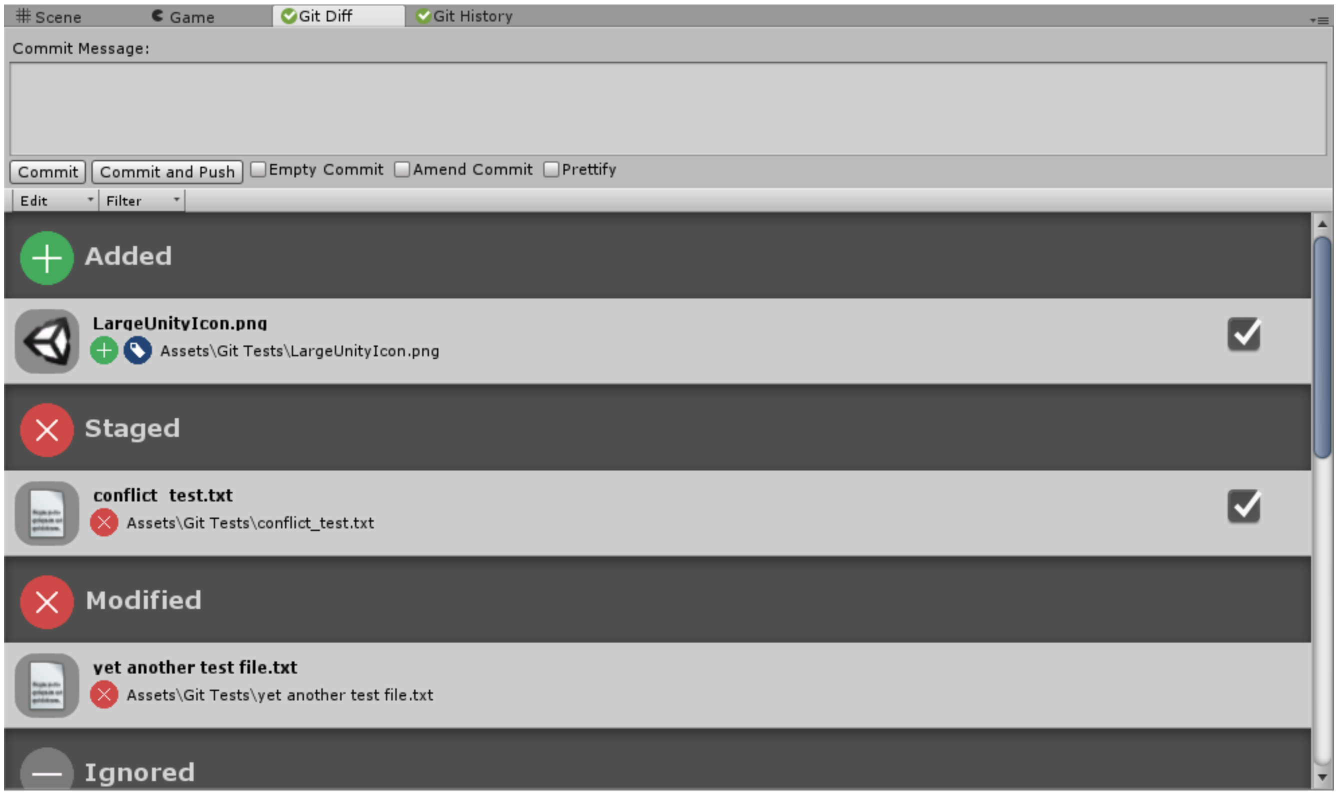
(Added)	Assets	Plugins	UniGit	Editor	GitCommitDetailsWindow.cs
(Added)	Assets	Plugins	UniGit	Editor	GitCommitDetailsWindow.cs.meta
(Modified)	Assets	Plugins	UniGit	Editor	GitCredentials.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitDiffInspector.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitDiffWindow.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitHistoryWindow.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitProjectContextMenus.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitPullWizard.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitSettingsWindow.cs
(Modified)	Assets	Plugins	UniGit	Editor	GitWizardBase.cs

All Git options like pull, push, merge and fetch can be found in the toolbar. They open their respective wizard windows.

File Difference and Committing

All changes to files can be seen in detail in the Git Diff window. The Diff window separates files into change type categories like added, modified, staged, removed and so on.

Each category can be expanded by clicking on it. They can also be hidden entirely by adjusting the filter option in the toolbar.



LargeUnityIcon.png
Assets\Git Tests\LargeUnityIcon.png ✓

conflict_test.txt
Assets\Git Tests\conflict_test.txt ✓

yet another test file.txt
Assets\Git Tests\yet another test file.txt

You can quickly jump to the affected object by double clicking it, or you can open more options by right clicking.

File Difference

Meta Changes

Each change can affect a file or it's meta file. UniGit unifies changes to the file meta and show the change on the file with an icon this indicates that the meta file has changed.

If the asset and it's meta file are changed then both the meta icon and asset icon will appear.

Stage/Unstage

Each change has 2 distinct types of change. Changes made that are either staged or unstaged. This means that when you commit changes, only files that are "staged" will be committed. UniGit indicates such files with the tick as shown above.

To "stage" or "unstage" files you can right click on a file and do so. This menu is also available in the Project View Window.

Differences/Changes

Differences between changed files and previous version can be show by clicking the **Difference** and **Difference with previous version** button in the context menu of each file. These options are also available in the Project View Window.

Reverting

Files can also be reverted to before they were changed. To do so just click the **Revert** button in the context menu for each file. This option is also available in the Project View Window.

Committing

Committing changes can be done by hitting the **commit** button. It will prompt you to choose **commit** or **commit and push**. Note that if you have selected to commit trough an external program you will be taken to that program once you hit the commit button and the **commit and push** button will be grayed out.

Once the changes are committed you can push them to your remote repository.

Setup

Configuration

If you put UniGit in an empty project, or a project that is not a Git repository, a notification will be shown on every UniGit window. The notification will allow you to Initialize a git repository. Note that UniGit copies a built in. .gitignore file that contains all the necessary Unity3D filters. The ignore file also ignores UniGit itself

If you want to clone an existing repository, you will have to do so with an external tool. Unfortunately this plugin requires Unity and changes to project setting or other files outside unity may not be allowed.

Remotes

Once you initialize a repository a **master** branch will be created. If you want to assign a remote to this branch, you must create one first.

In the settings menu open the **remotes** tab, there you can add a new repository and assign its URL and Name. Once the repository is created you can head over to the **Branches** tab in the settings window. There you can assign a remote to your new **master** branch, or any other branch in your repository.

*Once you push/fetch changes to/from your remote repository a new branch will appear, Called **(remote name)/(branch name)** - substitute **(remote name)** with your remote name and **(branch name)** with your branch name.*

Credentials

UniGit has it's own Credentials manager. This manager is entirely optional, but if used will allow you to automatically fetch changes from remotes. It uses Windows Data Protection API (DPAPI) to encrypt passwords.

The Git Credentials Manager only works in Windows

To add a credential go to the **Security** tab in the settings window. Once you create a credential you can name it, assign a URL (usually the path to your remote repository), add a username and/or change a password.

GitHub tokens are also supported.

It is important to not include the **Git-Credentials.asset** file to your repository, especially if you use a GitHub token. This is the file that your encrypted passwords are used. If you do include this file, the Windows Data Protection API should help protect the encrypted password as it can only be decrypted by your Windows user.

If you don't want you use the credentials manager you can always type your password manually when you push or fetch changes.

Git Settings

Git has a lot of settings. Documentation no all Git settings can be found here. UniGit includes some of the basic settings, like:

- **User.Name** The Name of the user
- **User.Email** The email of the user

You can find these settings in the **General** tab in the settings window. You can also manually edit the local config file .git/config.

UniGit Settings

- **AutoStage** this options allows for automatic staging of changed files. Staging the changes ready for committing.
- **AutoFetch** allows for automatic fetching of changes from remote repository. This will notify you of any changes made to the remote repository. Note that if the changes are quite big, the fetching process might take some time, as fetching download all changes but does not merge them.

External Programs

UniGit provides the use for External Programs for tasks like pushing,pulling,fetching,merging,committing and others. You can choose your external program and when to use it in the Externals tab in Git Settings.

Programs that work with UniGit:

- TrotoiseGit
- Git Extensions

Credential Managers

Many operations that connection to a remote repositories, like pulling, pushing and fetching require some form of credentials. UniGit allows for direct input of username and password each time you do an operation.

UniGit has it own credential manager, that uses Windows Data Protection API (DPAPI) to encrypt passwords, but also allows for external Credential Managers. Managers that work with UniGit include:

- Windows Credentials Manager (Windows 7/8/10)

To choose your credentials manager just head over to the Security tab in the settings window. Note that if you change credential managers your old passwords will be lost, even if you used the inbuilt credential manager.

Git LFS

Note that Git LFS with inbuilt tools is still in beta. It is recommended you use External tools when pushing, pulling, fetching or merging.

Git Large File Storage will work out of the box while using external tools. If using inbuilt tools then you need to setup a credential manager like Windows Credentials Manager os that Git LFS has access to your credentials when pulling and push