Unity and Subversion

1 Introduction

Unity offers an Asset Server add-on product for easy integrated versioning of your projects. If you for some reason are not able use the Unity Asset Server, it is possible to store your project in any other version control system, such as Subversion. This requires some initial manual setup of your project.

Before checking your project in, you have to tell Unity to modify the project structure slightly to make it compatible with storing assets in an external version control system. This is done by selecting Edit > Project Settings > Editor in the application menu and enabling External Version Control support by selecting Metafiles in the dropdown for Version Control. This will create a text file for every asset in the Assets directory containing the necessary bookkeeping information required by Unity. The files will have a .meta file extension with the first part being the full file name of the asset it is associated with. Moving and renaming assets within Unity should also update the relevant .meta files. However, if you move or rename assets from an external tool, make sure to syncronize the relevant .meta files as well.

When checking the project into a version control system, you should add the Assets and the ProjectSettings directories to the system. The Library directory should be completely ignored – when using external version control, it's only a local cache of imported assets.

When creating new assets, make sure both the asset itself and the associated .meta file is added to version control.

2 Setting Up

2.1 Getting the Necessary Software

Download TortoiseSVN from http://tortoisesvn.net/downloads.html and install it. It adds a few items into the Windows Explorer's context menu which we need for working with the version control system's repository.

2.2 Creating a Repository in GitHub

Go to https://github.com and click "New repository". Give it a name and an optional description and select if it should be public or private (the latter requires a payment plan). The repository is then created under https://github.com/username/reponame, i.e. https://github.com/CaptGrumpy/UnityTest.

2.3 Import an Existing Project

Follow these steps to create the initial import in the system:

1. Enable Meta files in Edit->Project Settings->Editor.
2. Quit Unity (this ensures that all the files are saved).
3. Delete the Library and Temp folders inside your project directory.
4. Import the project directory into Subversion by right-clicking on the root folder, selecting TortoiseSVN > Import… and provide the repository URL.

If successful, the project should now be imported into subversion and you can delete the project directory if you wish.

2.4 Checking Out

Check out an existing repository by right-clicking on a newly created (i.e. empty) project folder and select “SVN Checkout…” from the pop-up menu that appears.

Specify the URL of the Subversion repository and the directory where you want to create the working copy (this should be the root of your Unity project folder) and press OK to perform the checkout operation.

You should be presented with a confirmation dialog stating that the checkout operation completed.

Open the checked out project with Unity by launching it while holding down the Option or the left Alt key. Opening the project will recreate the Library directory.

You could also checkout an (empty) repository over an existing Unity project and then add all relevant files and folders manually (see next chapter), but importing is the better alternative here. If you do it nevertheless, the TortoiseSVN client should display a warning dialog box stating that the directory is not empty. Confirm that you want to checkout to this folder by clicking the Yes button.

3 Working with Subversion

3.1 Add Your Stuff

Only two folder (and all of their contents) should be added to the repository: Assets and ProjectSettings. The following folders should NOT be added to version control: Library and Temp. Also, all of the files that Unity generates in the root folder of the Unity project folder should not be added to version control (because these can be regenerated by selecting Assets > Sync MonoDevelop Project from the main menu in the Unity editor).

If you imported an existing project or checked out one from the repository, this directory structure should already exist, so you only need to take care of adding new files or folders.

You can do this by right-clicking on the item you want to add and select TortoiseSVN > Add…

It is highly recommended that you tell subversion to always ignore the files and directories that you do not need to add to version control. To do this, right-click on the files/folders that you do not want to add to version control and select TortoiseSVN > Add to ignore list > Ignore items by name (or ignore by extension if you want to ignore all files with a particular extension). Once a file or folder has been ignored, it won't show up in the commit dialog anymore.

3.2 Commit Your Stuff

Once the necessary files have been added to the working copy and the Library, Temp, and other files have been marked as ignore, we need to commit the changes to the SVN repository.

Do do that, right-click on the root folder of the Unity project and select SVN Commit… from the pop-up menu that appears.

You should be presented with a commit dialog box. Enter a message that describes the changes you are committing and confirm that all of the files and folders that should be added to the repository are checked (including the .meta files!)

Click OK to confirm you want to commit the changes to the repository. If everything was okay, you should a dialog box telling you that the commit has been completed.

3.3 Update From Repository

Right click on the project's root folder and select SVN Update to get the latest changes from the repository, i.e. what your team mates committed.

4 Working With Unity

It is important to understand the need for Meta Files when working with a Version Control System. When you reference a game asset in a scene, Unity does not use the name and location and name of an asset internally, instead Unity uses a Globally Unique Identifier (GUID) to uniquely refer to an asset in your project. This GUID is stored in a meta file that is created for each folder and each asset in your Unity project's Asset folder.

The use of GUIDs has its advantages but it also has its disadvantages. The advantage of using GUIDs is that you can move or rename or modify the contents of an asset and the asset can still be referenced by its GUID (as long as the GUID stays the same, the asset can be referenced). The disadvantage is that you must be aware of the presence of the meta file that is associated to a particular asset. If you delete the meta file that is associated with a particular asset, Unity will think the original asset has been deleted a new GUID is generated and associated to that asset. This is the most common reason why references to assets will suddenly break in your scene file.

In addition to the GUID, the meta files store information about the asset that is used to re-import it. For example, texture assets can be configured to be imported as a standard texture, a normal map, a GUI texture, a cookie, texture, or a lightmap texture. These import settings are also stored in the meta file.

If you have read the standard Unity documentation, you may have noticed that the guys at Unity recommend that you never use Windows Explorer to move or rename files that are in your Unity project's Asset folder. They recommend that you only use the Project View in the Unity editor. This is true if you are using the Unity Asset Server or not using version control at all. However, if you are using a version control system (like Subversion) then you MUST use the SVN client (in the case of Windows, this is usually TortoiseSVN) to move or rename files that are part of an SVN working copy. The most important thing to remember is that if you move a file or folder that you also move/rename the corresponding .meta file that is associated with that folder or asset.

When moving asset files or folders in a Unity project folder, always move/rename the corresponding .meta file! Not doing this will break the references to those assets in all of your scene files!

If you don't move/rename the corresponding .meta file for a folder or an asset in a Unity project's Asset folder you will break the reference to that asset in every scene file it is used.

BE VERY CAREFUL WHEN MOVING/RENAMING FILES IN YOUR UNITY PROJECT!

If you accidently move/rename files in the Project View in the Unity editor, you will break the link to the file as known by the Subversion system. To resolve this, move/rename the file/folder back to it's original name in the Project View and perform the move/rename using the SVN client instead. Make sure you also move/rename the corresponding .meta file that is associated with the file/folder you are moving/renaming.

If you want to make a copy an asset, do not copy the .meta file with it. The .meta file contains the GUID for that asset and if you make a copy of the .meta file, Unity will not longer know which is the correct asset for the GUID.

5 References

http://docs.unity3d.com/Documentation/Manual/ExternalVersionControlSystemSupport.html
http://3dgep.com/?p=5105
http://tortoisesvn.net/downloads.html