Source Control in Unreal Engine 4

By Keenan Gray

Outline

- Brief intro to source control
- Using SVN to manage Ue4 asset files
- Using SVN or Git to manage C++ code files
- Using Ue4's migration tool to manage Ue4 Asset files

Why use source control

- Maintain backups
- See changes
- Merge files

Problem: Game engine content

- Asset files work differently than code files
  - Models
  - Textures
- .UMAPs and .UASSETS
  - Maps
  - **Blueprints**
  - Materials
  - Etc.

Options

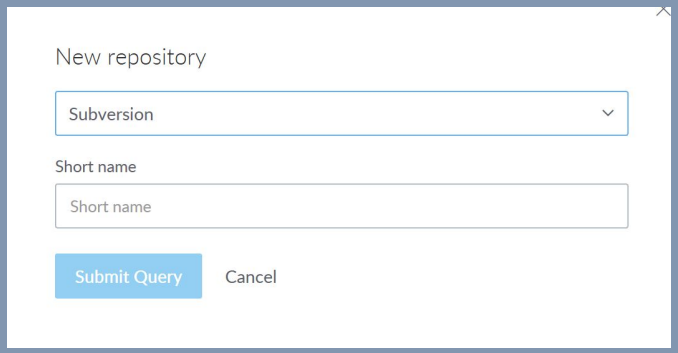
- Built-In
  - SVN – for managing Ue4 Assets
  - Have to use svn through the command line for code files

- Git/GitHub - Can be used with Git LFS, now available on lab computers (instructions for use can be found here https://git-lfs.github.com/ under “getting started”.
- Ue4 Migration tool can also be used for managing Ue4 Assets

Ue4 and SVN: Create an online Repo

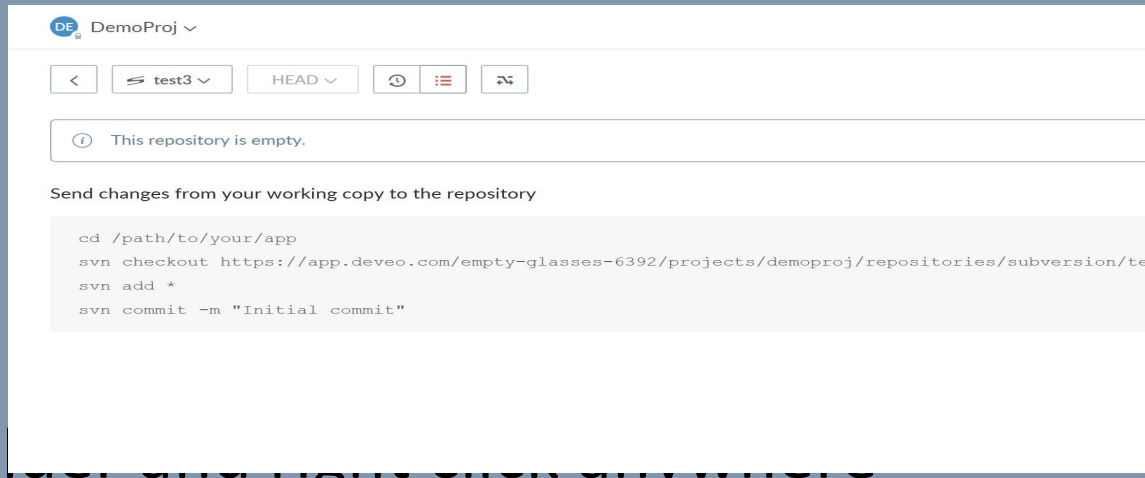
- I used Deveo to host the repository in the cloud.
- Create an account on Deveo.com if you don't already have on
- Add a new repo by clicking the plus.

- Select "Subversion" and name the repo the same name as the project directory (created when you create a new Ue4 project



Ue4 and SVN: Checkout a working copy

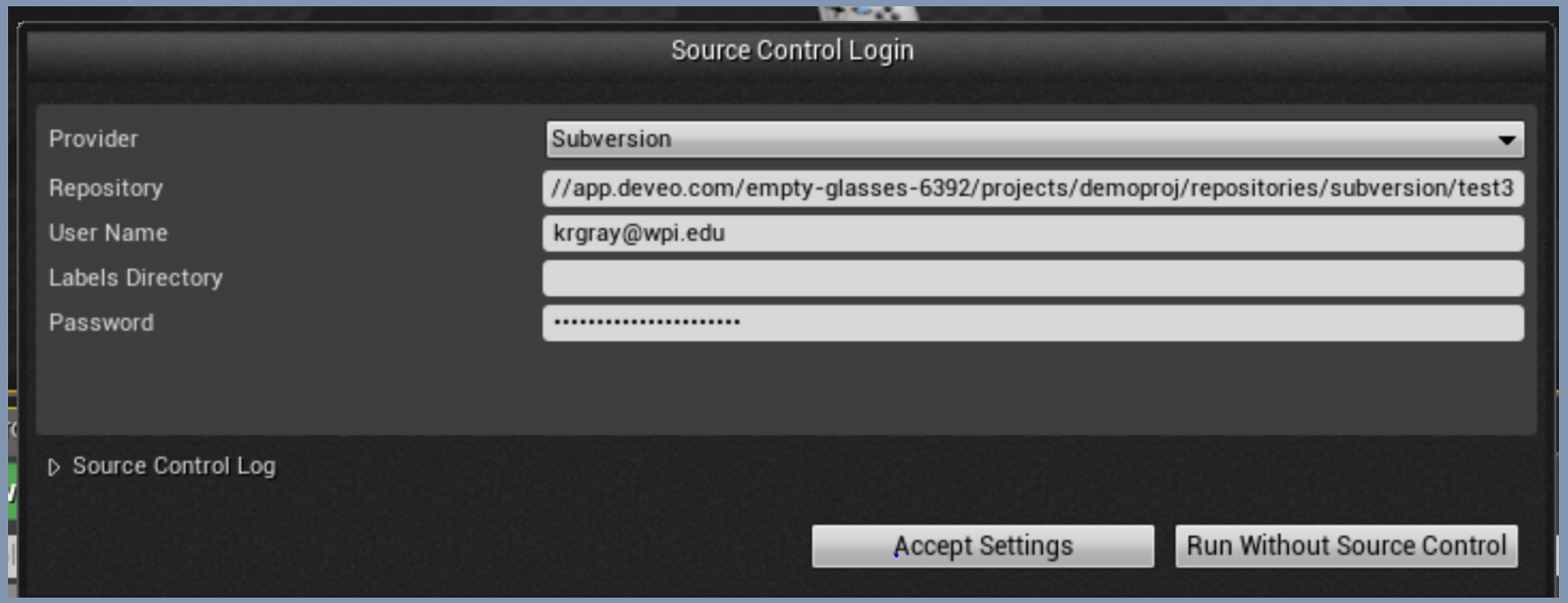
- Click on the new repo



- Open the ue4 project fo
- Select SVN Checkout from the menu.
  - Be sure the checkout directory is set to your project folder
- Enter the name of the repository press “Ok” and enter the use and password

Ue4 and SVN: Add repo to Unreal

- Back in the Unreal Editor, right click a UASSET file in the editor select “connect to source control”.
- Fill in the information and press accept



Ue4 and SVN: Finishing Up

- If you have connected successfully you can now manage all o
  asset files using SVN.
- Right click asset files and select “add to source control”
- After adding files you can click the “Source Control’ button ab
  editor window and commit all the changes.

Final notes on in Editor source control

- Open maps cannot be synced, you will have to open a differe
  before you can synchronize changes from one project to anot
- I confirmed that it works but have not delved into its use exte
  so please let me know if you have any trouble.

Managing Code Files using SVN

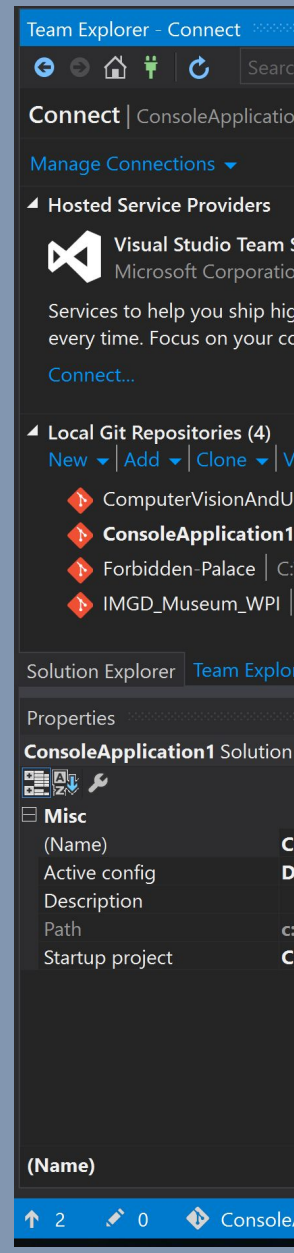
- You can use the command line or Visual Studio.
- In Visual Studio “VisualSVN” from the taskbar and select “Add solution to subversion”
- If you have already set up the Ue4 project with SVN you will to commit and push changes to the remote repository.

Git Plugin for Visual Studio

- Got to “Tools-->Options-->Source Control” and select Git as
  source control plugin.
- Right click the solution and select “add project to source con

Git Plugin for Visual Studio

- The menu in the bottom right corner can be used to add a remote repository, create branches, push and pull.



Managing Code Files Rebuilding the Proj

- Every time you **add** code files, by pulling or using "svn update new version, the project will need to be rebuilt.
  If you are only changing something in a file, you should not to rebuild – just compile.
- Close the Ue4 editor.
- Open the project directory and delete the Binaries and Intern folders.
- Re-open the Uproject and select "Rebuild" when prompted.

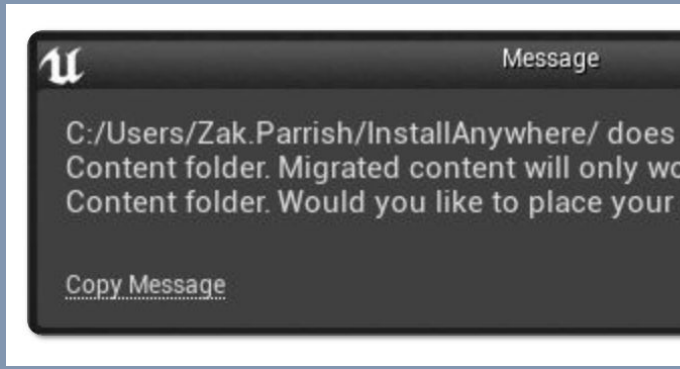
Unreal Asset Migration

- https://docs.unrealengine.com/latest/INT/Engine/Content/Br
  UserGuide/Migrate/index.html

- Useful for moving all assets between projects and maintainin
  between files (such as textures + materials)

Unreal Asset Migration

- The important takeaway is that you can migrate all assets (or updated assets) into a local folder.
- To migrate right click a file and select “migrate” or “asset actions -> migrate” click “Ok” then select a folder to put the output into.
- You will get this error when migrating the asset to any non-content folder, but given the reason we are using this you will want to say yes.



Unreal Asset Migration

- While you work on assets you can keep versions up to date by managing your zipped up content folders on google drive or d
- To add the new files to another project, simply unzip the folde are using and copy any new assets into the “Content” folder o project.
- Asset migration in this way is a manual way of moving assets b projects but it should be sufficient for this course.
- I offer this here as an alternative to SVN.

Summary

The Three options for managing your unreal projects are as foll

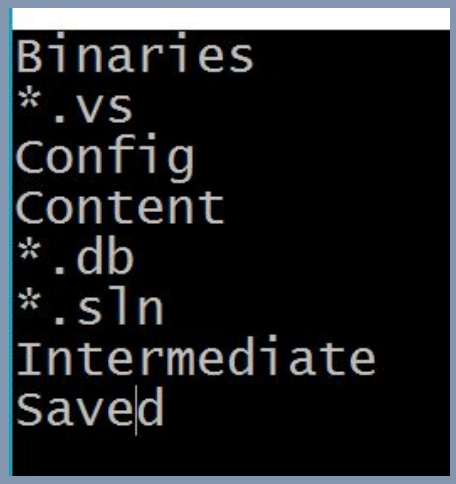
1. Use SVN in editor to manage binary assets + SVN in Visual Studio t
   code – This is what I recommend. I can help teams become more f
   with SVN it is relatively straightforward.
2. Using Git and Git LFS
3. Use Ue4 migration to manage binary assets + Git or SVN to manag
   this case you are also storing things in multiple repos but it may be
   sufficient for this course).

Finished

Notes on using the command line

Managing Code Files: Ignore Binary information

- Step one: Setting up an ignore file.
- Git: create a file called .gitignore
- SVN; create a file called .svnignore
- Add these lines



- For the .gitignore you may need to add /* to the directory names

Managing Code Files: Ignore Continued

- SVN: run the command “svn propset svn:ignore -F .svnignore
- The “.” at the end is important.
- This will set the ignore property of all the directories/files nar the .svnignore file
- Run “svn propget svn:ignore” to see the currently ignored file

```
krgra@WINDOWS-4HF1UFM MINGW64
$ svn propget svn:ignore
Binaries
*.vs
Config
Content
*.db
*.sln
Intermediate
Saved
```

Managing Code Files using SVN or Git

- Now you can add and commit code files using Git or SVN.
- I can provide individual help to teams that haven't used either these before.
- The basic structure is to first add the associated files, commit with a message, and then push them to the repo.
- **Caution:** In SVN do not use the "svn add *", this will still add ig files to the commit. Instead use "svn add . --force".
  - This is how it works, I have no idea why.