Source Control for Game Development Teams

Introduction

Source control allows a team of programmers to work on the same codebase from a central repository. The “true” version of the code is stored on a server (sometimes called the repository). Members of the team download the code files from the server, and make changes locally. Once those changes are tested, they are uploaded back to the server, and the latest version on the server is updated accordingly.

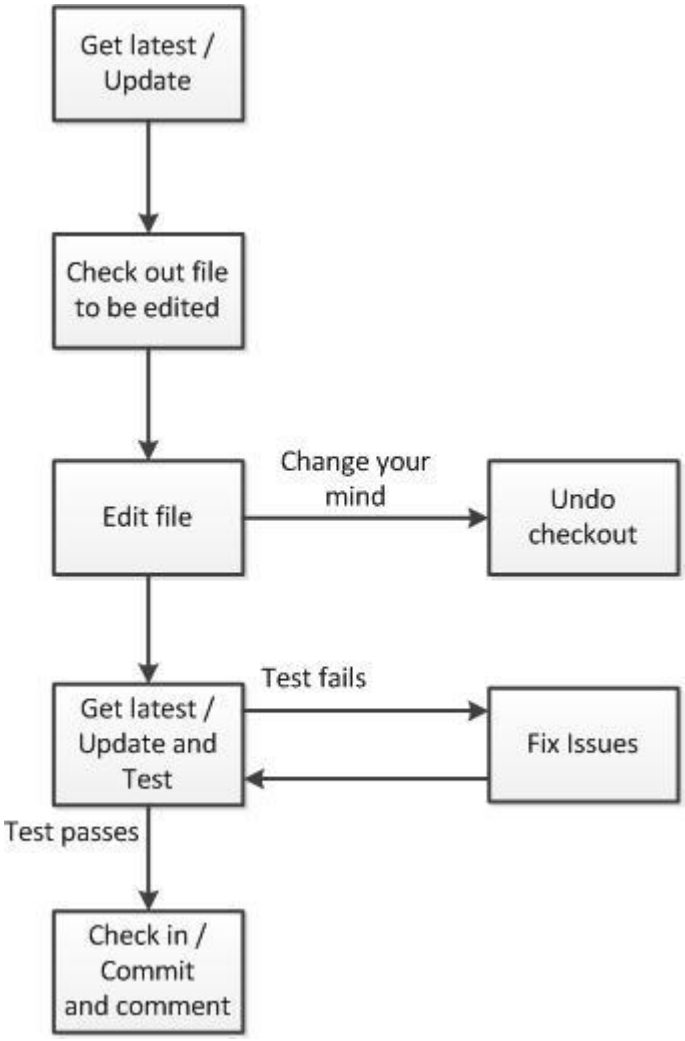
Features

Typical features of a source control system are:

- Download the latest version of the code from the server onto a local PC. This is usually called either “Updating”, or “Getting Latest”.
- Upload changes made to particular code files on a local PC back to the server. This is known as “Committing”, or “Checking In”, and it allows other team members to access your latest work as part of the project.
- Mark a file on the server as being unavailable to be edited by other team members while one (or more) team member works on it. This is known as “Checking Out”.
- Provide differencing information between the local file(s) and the server file(s). This is an extremely useful tool for comparing the changes you have made locally with what is on the server.
- Provide a history of changes made to each file on the server. Every time a file is checked in to the server, a new version of the file is stored and dated. This history can then be searched and compared using the differencing tool.
- Rollback the server, or the local PC, to older versions of files. This is useful if something has gone wrong with the latest version, so you need to roll back the whole project, or specific files, to a point when it worked.
- Attach comments to files as they are uploaded. This is very useful when searching the history for when a particular change was made.
- Attach labels to particular versions of the files on the server. This is used for identifying important versions of the project (for example a milestone deliverable build, or a demo).

Teamwork

Before committing code, you should test it against the latest files on the server. The reason for this is, while you have been working on some code, somebody else may have made changes and checked in code which is more recent than that which you have been using. Testing should involve both checking that your new code compiles with the latest server code, and that the resulting executable does what you expect.



Most source control packages include integration with Visual Studio, so that the functions are available from within Visual Studio. Some prevent editing of a file unless it has been checked out from the repository.

Source control is usually used for both code files and other assets (i.e. artwork, animation files, design docs, audio files, script files, etc.). Depending on the nature of the asset, some features of source control tend to be turned off (for example, keeping a full history of art assets can be very costly on disk space, and not especially useful).

A general rule of thumb is to check in, or commit, work frequently. This means that the history consists of many smaller changes, rather than just a few major changes. If something goes wrong with a project, it is much easier to track down the issue if submissions are small and frequent. Also, it is more likely that your code will work with the latest version if you have been checking in and updating frequently. I generally recommend that coders check work in to source control at least once a day.

Structuring your code well is vital to a successful team project. The code files should be arranged in such a way that the whole team can work at once without constantly treading on each other's toes. An obviously extreme bad example would be to put every line of code into a single file – main.cpp –

then everyone on the team would be trying to edit it at the same time. Split your project up into sensible areas (AI, graphics, physics, input, etc) and structure and name your files accordingly.

SVN for Team project

There are many different packages providing source control that are used in the games development industry, including Microsoft Sourcesafe, AlienBrain, Perforce and SVN. For the team project you will be using SVN.

The university will set up a SVN project for each team. The software you should use for interfacing with SVN is:

AnkhSVN

This provides the integration with Visual Studio, and should be used for all your code updates.

http://ankhsvn.open.collab.net/downloads

Tortoise

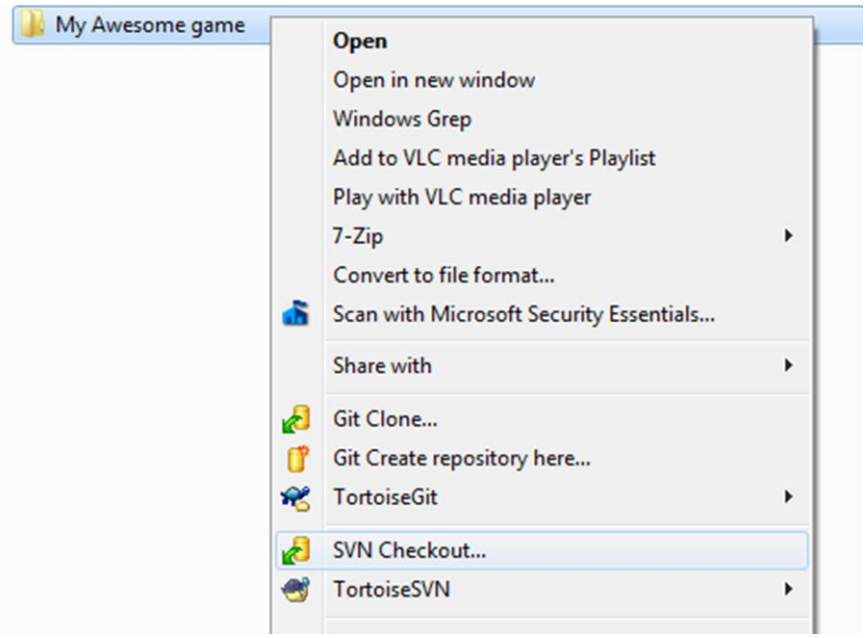
This provides an interface within Windows Explorer for all assets; it should be used for the artwork, audio files, etc.

http://tortoisesvn.net/downloads.html

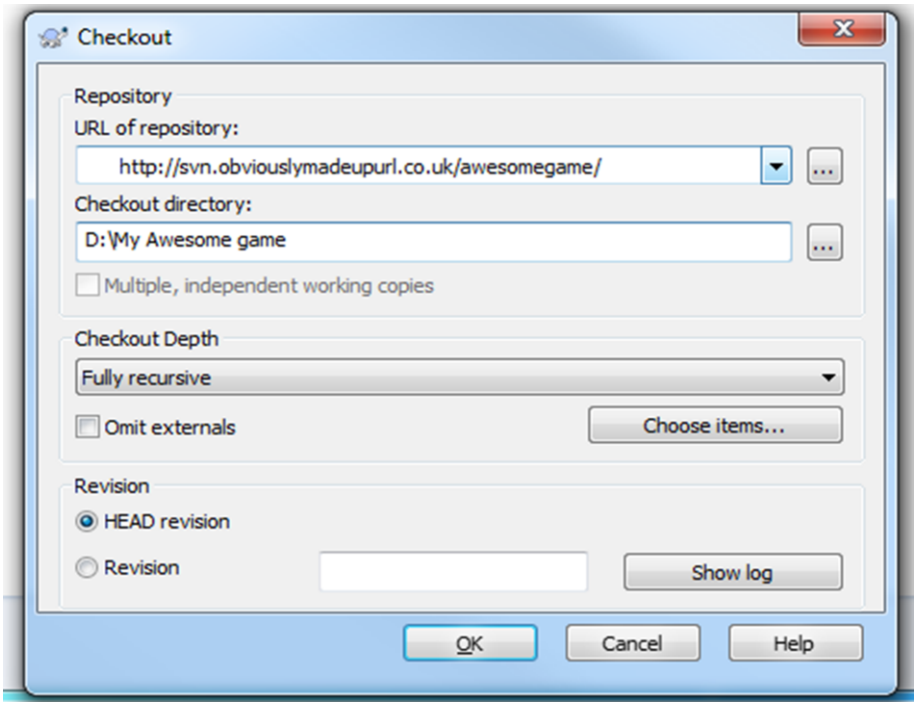
Getting Started

- Everyone should install the two pieces of software from the links above.
- Appoint one person on the team to set up the initial project in SVN.
- That person should commit the first version of the project – the codebase that you will use as the framework – to the repository using AnkhSVN.
- Once it is committed, everyone on the team should use AnkhSVN to update from the repository, to download the base framework onto your local PCs.
- Each team member should use AnkhSVN to check out a code file, edit it by adding a comment "//Testing Source Control YourName", and commit the file with a comment in SVN.
- Look at the history of the files you have edited, to check everybody has successfully committed a change to the project.
- Repeat this process for assets using Tortoise
- One person set up the initial project structure for the assets and add a dummy texture file using Tortoise
- Everyone use Tortoise to update from the repository, getting the file structure and the dummy file
- Everyone add an extra dummy texture file, or some other file, and commit it
- Check that all the dummy files have been successfully added.
- One team member should try to change a dummy file, and then commit it
- Another team member should try and revert the file change
- Delete these dummy assets from the repository if they are not going to be used in the project.

Checking out a repository using TortoiseSVN

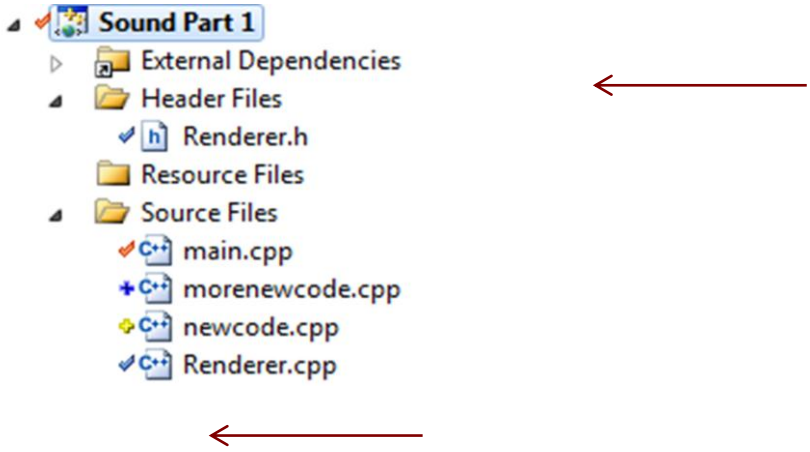


Step 1: Decide where to checkout the SVN to. With TortoiseSVN installed, click the destination folder, and select the new 'SVN Checkout' option

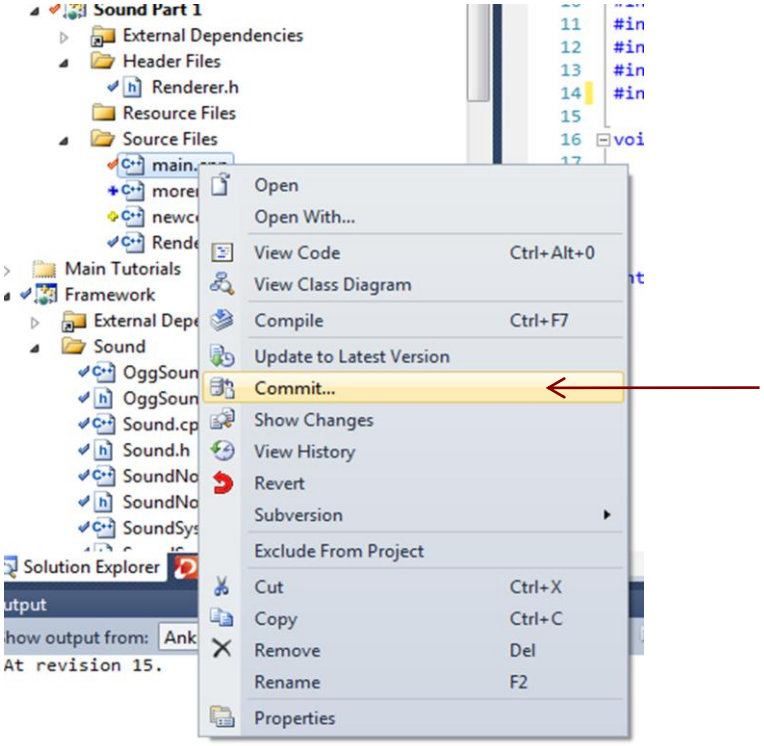


Step 2: Input the provided SVN repository URL. Select 'HEAD revision' if you want the latest version of the code, or select 'Revision' for a specific version. Press OK to download the repository to the selected folder.

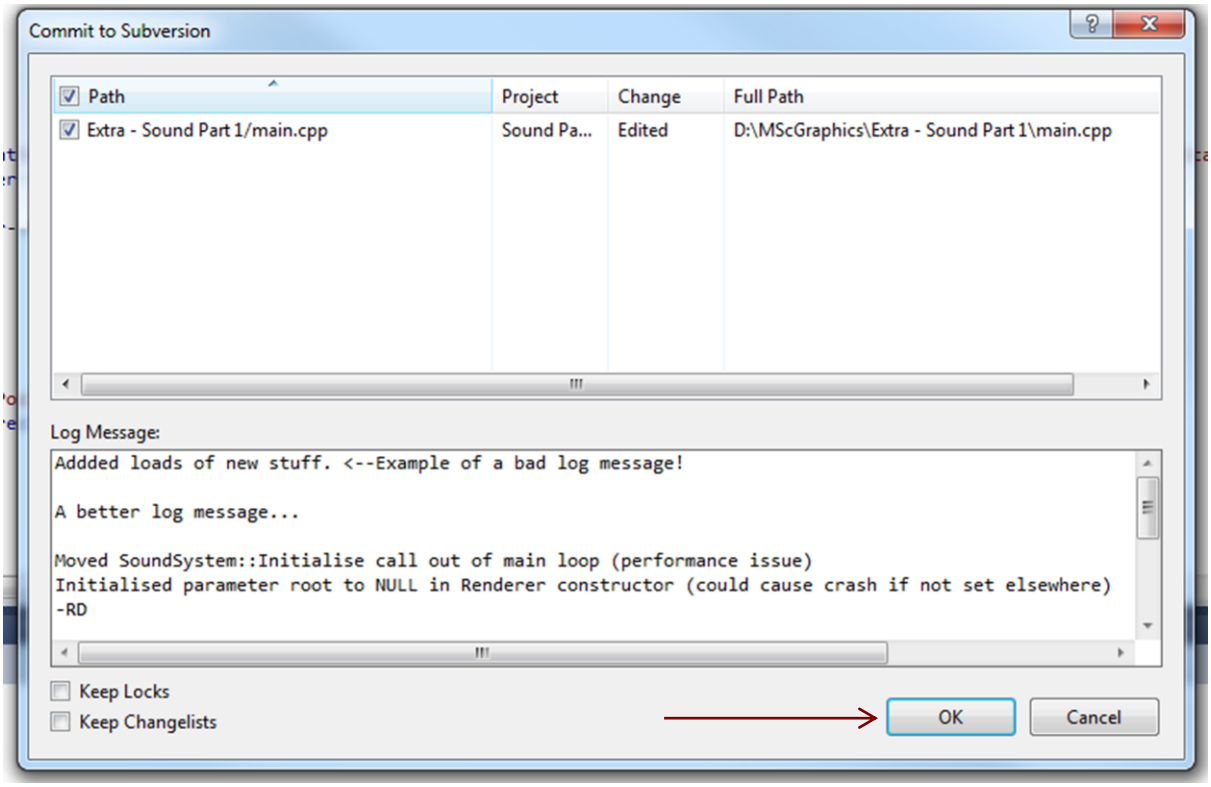
Commiting code changes using AnkhSVN



AnkhSVN uses a number of symbols and colours to represent the SVN status of your files: Ticks represent files handled by SVN, while a plus represents a file to be added to the repository. Blue means 'Up to date', yellow 'Just added to SVN', while red means 'Changes need committing'



You can either commit a single file, or your entire solution, by right clicking on the appropriate item in Visual Studio, and clicking on 'Commit'



Add a log message to describe the changes you have made to your codebase in the latest commit, and press OK. Remember to make the log message descriptive! It will help you if you need to revert any changes or track down any new bugs that occur