Revision Control: What it is & why I should care.

A. Buskov

For as long as there have been programmers and computer code, there has been some sort of version control. Without version control our code would soon grow to the point that further development, much less the fixing of bugs, would overwhelm any one developer. But what exactly is version control, and how will using it make me, or you for that reason, a better programmer.

Version control is really nothing more than a simple folder structure with a very elaborate way of indexing specific files. In short, version control allows you to keep track of a multitude of things related to an individual file. To fully understand version control though, I must explain a bit about coding in general and some specific definitions. When a programmer creates a file there are almost certainly going to be bugs somewhere; if you didn't find any bugs... you didn't test hard enough. As a single developer, working with a small set of files, finding these bugs and furthering development may not be too much of a problem. As a single developer, you are free to open, edit, move, and change your files as you see fit. However, problems arise quickly when two simple things change: multiple people work on development, and an increase in the number of files. With these two changes managing your work, allowing others to work on your code, and tracking changes become nearly impossible.

Enter version control. Because of the way version control is designed, it easily allows a developer to work on a piece of code without causing issues with the stable code, or other developers. All code starts out in what's called a trunk; the place where all stable (or master) code will live. Each trunk can have multiple branches. These branches will be an exact mirror of the main trunk, but will serve as working copies for developer to use so as not to disturb the main trunk. It is because of these branches that multiple developers can work on the same file at the same time. Without these branches when multiple

developers worked on the same file the developer who saved the file last would simply overwrite all changes by the first developer.

Because there are multiple developers, who each make multiple changes, version control automatically takes care of tracking those changes, allowing the developers to see what work has changed with each revision of a file, and who made those changes. This helps hold each member of a development team accountable for their code while making it easy to revert code should a feature be dropped or buggy. Branches are really where all the work takes place, and like a tree there can be multiple branches of development going on at any one time. When a piece of a branch is finished, it's code is merged into the master trunk for distribution.

Version control also allows for quicker and easier development by distributing the code easily to multiple machines. This allows a developer who has a tablet, laptop, desktop, an maybe even a server to edit his code on each machine yet easily transfer that code to other devices when necessary. This is something I do quite often. As all code is stored in a repository, the developer simply need to commit all code changes to the time line, and push those changes to their central repository. A repository is where the code is hosted, and there can be many repositories for the same code base depending on the system that is used. Each repository will hold an index of all the changes that have happened to each branch of the code that each and every developer has altered. In short, it's essentially a very detailed list of who changed what file, when they changed it, and what was changed.

Some of the more common version control structures out there are git & svn. Both of these do a very good job at tracking changes to code, but each does it in a specific way. Git is setup in such a way so that each developer has a complete set of all code in a specific project. Svn is setup so that a developer can check out a specific branch to work on without checking the entire project. Depending on the type

of development, the number of team members, and the decision of project managers each of these may be used with any specific project. In the end, it really boils down to a matter of preference.

With all this in mind, it should be easy to see how the use of version control will make you a better programmer; allowing you to work on multiple aspects of development at any one time without messing up other areas that you might be working on. It also allows you not only to track what changes you've made but to save your work in one location, yet work on it in many. Adding additional developers is relatively easy. And finally, tracking changes is a breeze due to the way version control handles changes.