MASS C++ Updates

Jennifer Kowalsky, Updated June 4, 2015

# Work Done

## Git Reflects

I went through the git commits to the Java version of the MASS library and reflected as many relevant commits as I could to the C++ version. To view the specific things I changed (and what remains to be changed), check out the issues in the Bitbucket repository for MASS C++ as wella s my commits to the master and develop branches.

## Documentation

Documentation for the MASS C++ project is now generated by Doxygen. The format for code comments is the Javadoc style. This was chosen so relevant comments can be easily copied and pasted between the two version of the library. Any new methods should follow the Javadoc comment style—it is very difficult to go back a year later and add comments to people’s work. The library has much better documentation coverage now, thanks to my efforts and particularly Zac Brownell’s efforts in documenting the tricky parts of the library.

Additionally the sample program included in the MASS library also follows Doxygen formatting now, and provides explanations of the various methods. It would be a good candidate for adding to the Bitbucket wiki code examples.

## Neighbors

The neighbor’s feature has been merged into MASS C++’s develop branch. I have left the branch Neighbors in the git repository though, since someone will have to port my changes over to the Java version. The changes are pretty minimal—design of the solution was the hard part, not implementation. You can performa diff of the Neighbors branch against the current master to what changed, or even go through the commits in the Neighbor’s branch. (I apologize for any unclear commits.) Check out my MASS Neighbors presentation to see where the major changes were as a guide.

## Performance Tests

I ran performance tests based on Jay Hennon’s test program located in the dslab account folder ~/jay. There are three scripts I've been using (Located in jay/Agents\_Baseline and jay/Baseline respectively):

**nighttests.sh** : this calls the next script several times to get a sample size of n = 20. That's double the sample size n = 10 I provided because about half of the tests experience a failure on at least one test, and that messes up the columns of data I end up grepping to form Excel sheets. The only thing you need to modify in here is the filename you want the data stored in, for example perftest\_256\_without09\_.

**runmanytests.sh** : This runs Jay Hennon's test program with all the variables we ended up collecting. The only variable you need to change here is size, which should be either 64 or 256 depending on which set of size data you need.

**perftests/grep.sh** : A simple grep that grabs all of the performance times and places them into a new file. The only variable you need to change is the filenames you want to grep.

The primary reason for this setup is that it provides a set of files in perftest with all the time data in order for all the variables, one file per column of data you need to get a sample size of n. Future work would be to add timeouts to these scripts to prevent hangups and timeouts from interfering with gathering test results.

# Future Work

## Merge Develop into Master

We left this alone until Zac Brownell was done running his performance tests, but develop is stable and ready to be merged into master with the new neighbors functionality.

## Add Documentation to the Bitbucket Wiki

The Bitbucket Wiki needs to be updated to provide most of the explanations currently in the MASS User Manual. Sample code should also be posted and explained there.

# Contact

If you have any questions about any of these things, particularly the documentation, neighbors, or the performance test scripts, don’t hesitate to contact me at jennykowalsky@gmail.com. I’d be happy to help anyone who needs information on this.