Assignment 2: Roll Your Own

So we've been talking about a few concepts a bit this quarter, and—while I'm not sure I've done enough to emphasize this point enough—the whole point has been designing a library that is easy to use for your average developer, but powerful and fast enough that your more advanced developer can do whatever they want with it. To that end, we talked about policies (and exceptions) and now some template metaprogramming. Soon, we'll be talking about using operator overloading to provide "domain specific" interfaces, another feature of C++ that sets it apart from pretty much every other language. You're getting to the point now where you can begin to design your own libraries on an expert level.

So, the objective for this assignment is to get you to actually choose and design a small library, focusing on how the user will use it, and what points of customization you might want to use. Don't worry so much about the implementation; I like implementation, but a good design can always have its implementation improved. A bad design will always be a bad design, since refactoring a library's interface tends to anger current users.

Thus, think about some kind of data structure, group of algorithms, or systems library (or something else) that you would like to have, or would like to see improved. If you want to do the former, how would like to use it? Does it fit into a larger system of concepts? (e.g. Should it be STL-compatible?) If you want to undertake the latter, what's wrong with current libraries? (e.g. IOStreams is really quite terrible: not atomic, slow, exception unsafe.)

Now, for the details of the assignment spelled out quite nicely in a bulleted form:

- By small, I'm not really sure what I mean. STL container sized seems about right.
- Again, teams are fine. I don't care how large your teams are. Just make sure everyone does something.
- **Due date: June 17**. Grades are due June 20. I start my summer job June 20 as well, so I don't want to be looking over things that day. Thus, I really really really mean no later than June 19.
- Again, I don't really care about implementation details. Those are boring, and you can always improve them. Some kind of mockup would be nice. (Some output that says "element added" would be sufficient.)
- You may use any external libraries you see fit, as long as it's within reason. The Myths now have Boost installed. (They were quite friendly about this.)

Ideas!

- IOStreams is really terrible. I have trouble getting over how terrible it is. printf,on the other hand, is very elegant and atomic, and fast too! However, printf is not safe for C++ objects. (You can't use ...'s with C++ objects.) Figure out how to get around that.
- I was working on a project and I decided I really wanted a trie. Look on wikipedia for information about tries, but I want a trie that works for any STL sequence container.
- I also want an immutable string that does really nice string sharing. Something like Java's String class. (gasp! Something in Java that David likes! Never!)
- How about the basics of a drawing library? C++ doesn't have a very good drawing library that's cross platform. Fix that. (Again, here's a place where I really don't care about implementation.

Convenient printf's would suffice.)

- Pick a design pattern. Give it a really sexy policy-based interface. Don't do Singletons or Factories, or there may be death involved.
- GUI. Yeah, this one is sort of a joke. But maybe a window or something. (Again, interface first. Here, the real question is if you should have a declarative [XUL] or imperative [Swing] interface.)
- A database access layer would be nice, especially if it plays nice with STL algorithms. Or if it had a SQL-like syntax, that would be awesome too.
- Flesh out the unit system we created. Allow for easy arbitrary creation of new units. Consider using boost::mpl::map or something.
- Anything that involves metaprogramming, especially if you emulate something else or calculate something impressive. This doesn't really need to be generally applicable, since I enjoy seeing things.
- Anything on this list, especially if you apply...

http://www.crystalclearsoftware.com/cgi-bin/boost_wiki/wiki.pl?Google_Summer_Of_Code_2006