Style Guide:

This may seem like it's unnecessary but I feel it would make a more uniform coding experience.

I will be posting a rough guide that we can alter and then push to github as our guide. Things like using lowerCamelCase for variable names, 2 (maybe 4 if people desire) spaces for indenting rather than tabs. Using get/set for getters etc. except when boolean, in which case use the word is. eg. someShip.isAlive() rather than someShip.getAlive(). Also please I beg you, use the preexisting code formatters in your IDE before committing, all IDE's have some keybind for it. It will space out/indent code correctly.

Code Review:

Everyone should feel welcome to critique each other code, if you see something that's named horridly im sure people wont mind a quick commit to remedy that.

How to commit:

Small commits, Don't commit 3 days work. commit small increments, at milestone points. Branch, do all your work on a branch and merge back into the master frequently, don't push straight to master, but don't leave a branch sitting for too long, or else you may end up well behind the master. Although everyone should be coding in separate areas there is a chance that multiple people will end up working in the same area to get things done.

What to commit:

DO NOT COMMIT LIBRARIES!! I've done it before personally and it's a nightmare, all of a sudden you have written 100,000 lines of code since last commit. You should only commit code relevant to the application. All library information should be kept in the maven & gulp build files, they will automatically retrieve the correct libraries on build from the internet, this includes SoGaCo files, we should still treat this as a black box, even though we have the source.

Don't commit Intellij IDEA/Netbeans/Eclipse files, keep them in .gitignore. Reason being I know Aidan is using Netbeans (Easy to work with Java servlets), I will most likely use a combination of ATOM and IDEA, and I know some of you may use Eclipse, Once we have sorted the build script all we will need to do is run some command on CLI like "make development" or similar and that will launch the server. everything else will be handled, you won't have to worry about main classes etc. (Thats the plan.) Which means there's no reason to copy project files across.