2 Jan, 2018

Robert Trzeciak

Program Manager

Northrop Grumman IS

468 Viking Drive

Virginia Beach, VA 23452

757-498-5544 work

757-635-2628 mobile

**Progress Report – 1 Dec 2017 – 31 Dec 2017**

Contract Number: HSHQDC-06-D-00022

Contract Number 7500097279

Order Number: HSCG23-07-J-TED150

Task Order – Performance Work Statement (PWS) 1.12

Attachments: (1) SAROPS subcontractor financial reports.

1. **Jira meetings, e.g. Sprint Planning, Sprint retrospective, etc. Lots of meetings. The last week, there were no meetings, and I didn’t take time off except for the 29th (happy anniversary to me) so that was an unusually productive week.**
2. **Worked on Sailboat voyages. More long discussions with Art Allen. In the process, I discovered and fixed two bugs that nobody else has seen. Times are stored as longs, but put into the particle file as ints (for space considerations). I have a routine that truncates the longs, but it was doing min(Integer.MAX\_VALUE, (int)timeToBeTruncated). That looks ok, but it actually creates a negative number; casting a big long to an int results in a negative number, which then “wins” the min. I fixed that with if (timeToBeTruncated > Integer.MAX\_VALUE) then return Integer.MAX\_VALUE.**
3. **Along these lines, I also found a “performance bug” when determining the “level” of a point. Sailboats are supposed to stay in the water and I had an assert statement that checked that they were in the water. That assert was taking too much time. It’s not relevant for the production code since “asserts” are not executed, but it was slowing me down during development, and is an error that I did not want to just leave waiting for it to bite me later when I’d forgotten about it.**
4. **For sailboats, I added to the particle file, an array that translates another array of ints to “leg type.” The leg types indicate whether the particle is tacking port or starboard, and whether the tacking is because it is going downwind or upwind. It would be much easier to do this if we were in netcdf4, but we’re stuck with netcdf3 for the time being; the latter does not support strings.**
5. **For transits, there are several steps. The first is to identify the blocks within a sortie (a block is what causes the start of a sequence of fluids, the fluids, and what causes the end). The second is step is to place individual blocks. The 3rd step is to modify the birds nest algorithm to push around Sortie-PatternVariables, and the 4th step is to make minor moves on Sortie-PatternVariables. There will be other steps, but these are the basic ones. The 3rd and 4th require me to adjust one Sortie-PatternVariable and figure out what to do with the other ones in that block. I think I have that code written. I’ve written the code that identifies the blocks, and I can place an individual block’s PatternVariables.**
6. **On Sailboats, it was decided that I should be the keeper of the polars. To that end, I put them into my distribution as resources within my jar file, but during the install, I exported them to the directory <installDir>/data/polars. Then it was decided that Judy would send me all of the polars in the xml so I wrote code to read the xml (and write it, so I could generate an example of what it should look like).**
7. **Later, it was decided that I would not be the keeper of the Polars. So I designed the xml Judy to pass to me. I then ran a case with my own gui, reading in from a spreadsheet, and wrote out a file with that xml, thus producing a concrete example for Judy to use. I then wrote the code to read that, and created another sample case to debug my own code. I discussed the sample with all concerned and it is now the agreed upon choice. We also cleared up some information about the use of the forbidden upwind (tacking) and downwind (gibing) zones; forbidden angle is the same as the first legal angle. Once we got that concept across, it was pretty easy.**
8. **Most bugs seem to be out of sailboat; two simple cases, one with changing winds and currents, seem to be almost working now; should finish this up early January.**
9. **Judy was producing xml that looks as if it’s for a real transit case, (not-standalone PATTERN\_VARIABLEs). We’re trying to just have standalones working, but that’s what was being produced for standalones. Fortunately, there are enough errors in that non-standalone xml so that I can recognize it as “not non-standalone.” Even better, this xml does have cst in the PATTERN\_VARIABLEs, so I can use them as standalones. I wrote code to recognize these as standalones instead of non-standalones with errors.**
10. **Along the way, I discovered a bug in my logging; my “error logging” was being logged to the out file. It’s supposed to, but it is also supposed to be logged to an error file. I then added a “warn” for the above situation; I put something to a “wrn” file that Judy won’t interpret as an error, the inconsistencies that allow me to interpret a non-standalone-looking snippet of xml, as a standalone.**
11. **Revamped sailboat to check for a turn when the environment expires (I check for new environment at a specified interval), when the boat, staying on its present course, starts to sail away from its target, when it arrives, when the current leg is close enough to check (another parameter in Sim.properties), and when it is close enough to its target to be classified as “arrived.” The latter two appear to be the same, but are actually quite different.**
12. **Rob Wilson got me a good Sailboat case; the winds and currents shift. I worked at getting that case to work, and think I’m close.**
13. **Re-defined the sail stages; they are SAIL\_PORT\_TACK, SAIL\_PORT\_DIRECT, SAIL\_PORT\_GIBE, similar for STRBRD, SAIL\_MOTOR, SAIL\_HOVE\_TO, SAIL\_PRJCTN, SAIL\_END\_STRETCH, and STUCK\_ON\_LAND. The last one can be for distress objects as well. These are the possible values that show up in the particle file for each particle/time-step combination. This list has grown and shrunk, but I’m pretty confident that it’s the right set now, or at least a lot better than some earlier versions.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Activity Worked** | **Hours Worked** | **Hourly Cost** | **Total Cost** |
| Kratzke | Coding/Doc/Travel | 179.25 | -- | -- |
|  |  |  |  |  |
| **Totals** |  | 179.25 |  |  |
|  |  |  |  |  |