1 Feb, 2017

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 Jan 2017 – 31 Jan 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. **The BuildSimLand “sausage maker” seems to be complete; there were a few glitches, but I think I caught them before the testers did. I’ve released my third version of the updated files and they seem to be pretty solid. This cleared all of cgtrk that was related to shoreline and then I suggested that someone should generate more files. The current version has Potomac, Shuylkill, Hudson, Hudson islands, Delaware, Delaware islands, and a stray Columbia island.**
2. **Organizing the input in a tractable form was possibly the most time-consuming part of running the program. I do not expect data to come to me with the following form, and it’s easy for me to put them there. It was not that easy to arrive at the following scheme:  
   The input files:  
   a. Need a ‘?’ stripped from the first line (NotePad++ makes this easy)  
   b. The xml files need to be renamed so that they end with “-adj.xml.” This is a requirement for me that makes it easy for BuildSimLand to collect all of the relevant files.  
   c. Those that are “ADD\_POLYGON” directives need to be changed from “SUBSTITUTE\_PATH.” Otherwise, BuildSimLand would try to attach the polyline to an existing polygon.  
   d. Any relevant xml file needs to go in a directory that starts with xx.xx-, where each x is a digit, and xx.xx indicates the order in which the files should be processed. For example, I put all of the “old” adjustment files in a folder called “00.00-00Adjs,” and labeled each folder of this current batch of adjustments either 01.00 or 01.01; I’ve enclosed two screenshots of the the main input directory and the Carib subdirectory  
   e. If it is not true that land-is-to-the-left, I need to put a ‘Reverse= “true”’ into the xml (so that I don’t have to reverse the order of points within the xml.  
   f. Add the “Delete” directives for the islands that are to be deleted (in this case, there were 2 in the Caribbean).  
   I wrote some code that pieced together shoreline sections; the Deleware, Potomac, and Hudson all came in multiple sections, and my program does not deal with a “section at a time.” It was easier to write code that produced a complete file than to modify the current code, although I could have gone that route.**
3. **All of #1 is a side-effect of my completing the overhaul of the geometry. This:  
   a. Improved performance (most noticeable on Slippery Shore cases with many land edges). The most time-consuming step in a Sim case now seems to be writing out the particle file (but we haven’t been working on completed searches lately). The point is, it’s not slippery shore.  
   b. Reduced memory usage.  
   c. (Most important of all from my point of view) SIMPLIFIED the code. I keep talking about “fundamental” problems, and a slightly higher-level one, as I mentioned before, is merging loops. In the old code, I would partition the set of loops and then apply the merging algorithm to each set, and then combine the sets. Because of the better use of memory, I don’t have to do that, and it’s considerably simpler. There’s one routine MakeLoopsFromLoops that does this merging. This comes up in the sausage maker, and planner (and the same code evaluates completed searches within Sim).**
4. **There were 4 approaches that I tried for splitting up a set of edges to make finding crossings faster. The first and oldest required some “tuning parameters” that I never could get to my satisfaction; I either burned up memory or performance took a hit, and different applications (e.g. USGS polygons vs GSHHS polygons) required different settings. That’s all gone now. There is still one parameter, but I’ve tried it set to 100 and again with it set to 250, and the program works fine either way, through all of the sets of polygons. Approach 2 (splitting the world up into longitudinal strips) and 3 (subdividing the sets of edges into 2 sets, then each of those into 2 sets, …) also had some “arbitrary parameters.” As an aside, the third approach was HORRIBLE to debug; the stack trace would be 15 deep in no time because of the recursive nature of the subdividing. The 4th one works very nicely, uses less memory, is simpler, and the stack trace is manageable.**
5. **A slippery shore bug popped up, but I tracked it down and solved it pretty quickly. In fact, it was reassuring to see that my code really will recognize when a 20-minute particle path “T-Bones” a land edge and doesn’t cross it, even at a very tight angle and close to the endpoint. In floating point arithmetic, that of course cannot happen, but we want to recognize when it “does.” I did invent a test for this and it caught this one, but there was another related problem. The bug was for such a case, two “dot products were being checked for a less than relationship, and the one that should have been smaller started with 8 9’s followed by 654, and the one that was supposed to be larger started with 8 9’s followed by 642. Fortunately, I was just trying to minimize the number of things that I had to check. The fix was “ok, check it if a < b or a and b agree to 7 digits.”**
6. **Rivers, which use the geometry, was broken, but I fixed that pretty quickly. Rivers have 3 strips of edges, and I have to find which edge in the center strip is closest to a given point. Then I construct a small great circle arc and figure out if it crosses one of the side edges. These are 2 of the 3 fundamental problems in the geometry; “given a point and a collection of edges, find the closest edge,” and “given an edge *E* and a collection of edges, find the first edge in the collection that *E* crosses, or determine that there is none.”Again, the code is shorter and simpler, and everything it running through the new geometry code. The test cases seem to run faster.**
7. **Worked on the TRACE problem; Using Curl, I verified that TRACE gets rejected (as it should), but I’m still getting complaints. I found another way of suppressing TRACE, that throws a 405 error instead of a 403 error. Am awaiting word on whether or not that passes; here’s more that I can try if that doesn’t work.**
8. **Voyage problems are now fixed. Interesting dead reckon case goes over Florida and drops particles in the Gulf, in *Lake Okeechobee,* and in the Atlantic. Also concluded that the “not enough particles are getting towards the end” is not a bug. I put in code that eliminated the draw for the time of distress, and with that draw eliminated, everything gets to the end.**
9. **Changed the installer so that it now considers the target machine when setting up its memory parameters.**
10. **There were other minor bugs and/or red herrings, but they seem to be clear now (knock on wood); the critical ones for 2.1 in cgtrk are done now.**

**Will be off 2/9 through 2/19. Travel on 2/22 for 2/23 IPR.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Activity Worked** | **Hours Worked** | **Hourly Cost** | **Total Cost** |
| Kratzke | Coding/Doc/Travel | 207 | -- | -- |
|  |  |  |  |  |
| **Totals** |  | 207 |  |  |
|  |  |  |  |  |

