SAGAZAN 278239 baptiste.sagazan@student.tut.fi

Which version of the code did you use? The original 1.0 before 15.3.2018 or version 1.1 released in 15.3.2018? The different versions will result in different performance, and version 1.0 may easily result in stuck satellites if running too fast due limited precision of 32-bit floats.

We used version 1.1.

Which of the loops are allowed to be parallelized to multiple threads?

All the 4 loops seem possible to parallelize.

Are there loops which are allowed to be parallelized to multiple threads, but which do not benefit from parallelization to multiple threads? If yes, which and why?

The graphics satellite loop could be parallize, but this loop must determine which of the satellite is the closest, to determine the colour of the pixel. So even if it’s possible to parallize it there is no point to do so because the programs need to compare the distance between all the satellites.

Can you do some code transformation (change of code which still causes same results to be calculated) which either allows parallelization of a loop which originally was not parallelizable, or makes a loop which originally was not beneficial to parallelize beneficial to parallelize? If yes, explain your code transformation?

So, with the previous question we have noted that the graphic satellite loop could be parallize but it’s not worth the work. An idea could be to parallize the loop and store the data in vectors and after the loop do the comparison to determine which satellite is the closest to the pixel.