Games Engineering:

All work done was done in pair programing.

Things we tried:

* We tried using openmp critical. This allowed one thread to update a specific piece of code at a time. This made no real difference in our fps, so we removed it.
* We used #pragma omp parallel to create a work sharing region. Inside this region we used “#pragma omp for” for our for loops rather than “#pragma omp parallel for”. This increased our fps because while using “omp parallel” inside another “omp parallel” the second parallel statement created another 8 threads resulting in 16 threads. This slowed down our programme considerably.
* We tried unwrapping our inner if statement in our update loop to update 4 grids square for each loop. This gave us a slight increase in performance. We later removed this as we got better performance using different methods.
* We added two if statements to our update loop so our program will only enter the update function if there is a fish or shark present. This increased our fps at the beginning of our program. We believe this is due to reduced locking of the map array and thus less threads are made to wait.
* We tried adding a work sharing region to our opengl rendering code but this provided no extra performance so we removed it.

Conclusion:

The program runs slightly faster with our OpenMp code.

Our performance was increased greatest in the beginning of our program when run, this is because at the later stages in the program there are more fish than sharks.

The Average Frames Per Second Graph:

* The logged data on the graph is an average of three runs.
* From this graph we concluded that with openMp and more threads the program runs slightly better.
* The program runs faster with our improved openMp code, at the start of the program.
* We ran the program for one life cycle and that is where the graph information comes from.