PARTICLE SYSTEM PRE-OPTIMIZATION ANALYSIS AND LOGS

CSC 461-Optimized C++, 2016

Charan Teja Allada,

1540260.

Introduction:

Given source code of particle system renders 40,000 tiny cubes which look like crystals of sand. This project uses OpenGL libraries to render these objects on the output window.

Objective:

Objective is to analyze and understand the construction of the source code for the existing project and make it run fast in terms of speed, optimize the given code.

Analysis Section:

- I understood the flow of the code and went through all the .cpp files in the given project.
- OpenGL stuff is irritating and hard to understand, but leaving it aside tried to understand the logic that is weaved with matrix multiplications.
- I tried to clean up the whole project, basically a first clean up: I put break points everywhere in the code at every condition.
- Debugged whole code with break points and deleted all the stuff to which the compiler has not visited.
- No surprises that this did not affect my speeds in both Debug and release modes.

- Even though I have not touched anything yet, I knew that I have to convert all these lazy code in SIMD in some time.

Location: Matrix.cpp and Vector.cpp

- Now I had put my time on refactoring Matrix.cpp and vector.cpp, I removed extra and temporary variables. Used simple optimizing techniques in constructors and operator overloads. Used RVO's at few places.
- (NO SIGN OF SIMD YET)
- I gave a try to see my values and there was no change in the timing.
- Now I implemented SIMD to all my Vector and Matrix methods, anticipated my timings will cut down to half promoting 50% efficiency. But the result was super slow particles moving around in the window.

Location: Particle.cpp and Particle Emitter.cpp

- At this time, I was uncertain what to do next and I switched SIMD code to normal and changed all the doubles to floats (read from PIAZZA post).
- This change has given a major change in the timings, Update and Draw fell around 980 and 730ms.
- I have also converted all doubles in OpenGL source to floats and it worked in favor of optimization.
- Removed few unnecessary matrix calculations in Draw method, actually the world matrix or the final matrix which to be rendered is of TRS form. More unnecessary matrices are declared and messed up with Draw method. These changes made the draw method few milliseconds faster.
- Made all possible variables as constants and it helped a little in the performance.

Location: Update

- First understood what was happening inside this method and removed temporary variables.
- Some of the operations are every time happening inside the loops which are not required, these statements were kept outside the loop.
- Removed the STL List draw buffer list and as the progress going on some of the methods are not called by compiler, those were cleaned up nicely.

- All the switch case are converted to **if** conditions, these did not improvise much on the performance.
- Alignment adjustments are made to the data of both the particle and emitter files.
- All the SIMD for Vector and Matrix are turned on now. I understood that SIMD is efficient when your code is clean and crisp. If your code is greasy and lazy SIMD makes it worse.
- Lastly compiler settings are made to optimize the existing project and these settings are same from the benchmark setting from PA4.
- Initially implemented proxy for the multiplication of matrices, but later removed it because the matrix multiplication is reduced to two matrices.

LOGS

1)

Initial Timings:

Debug:

Update - 1169.019834 milliseconds Draw - 722.452980 milliseconds

Total: 1891.47281 milliseconds

Release:

Update - 26.9461 milliseconds Draw - 79.1233 milliseconds

Total - 106.0694 milliseconds

2) After cleaning code

Debug:

Update - 1169.019834 milliseconds Draw - 722.452980 milliseconds

Total: 1891.47281 milliseconds

Release:

Update - 26.9461 milliseconds Draw - 79.1233 milliseconds

Total - 106.0694 milliseconds

(Almost similar)

3) Implemented SIMD before refactoring

Debug:

Update - 1221.000345 milliseconds Draw - 756.345654 milliseconds

Total: 1977.345999 milliseconds

Release:

Update - 33.4570 milliseconds Draw - 89.3850 milliseconds

Total – 122.842 milliseconds

4) Converting doubles to floats

Update - 980.555725 milliseconds Draw - 731.281921milliseconds

Total: 1711.837646milliseconds

Release:

Update - 17.156250milliseconds Draw - 71.034790milliseconds

Total – 88.191040 milliseconds

5) Draw buffer removal and temporary variables and loop clean ups.

Update - 454.673370 milliseconds Draw - 192.546600milliseconds

Total: 647.220032milliseconds

Release:

Update - 8.334164milliseconds Draw - 62.446560milliseconds

Total – 70.780724milliseconds

6) SIMD and Alignment

Update - 446.538910milliseconds Draw - 269.252167milliseconds

Total: 715.791077milliseconds

Release:

Update - 6.519125milliseconds Draw - 53.112968milliseconds

Total – 59.632092milliseconds

7) Switch statements to if conditions

Update - 446.538910milliseconds Draw - 269.252167milliseconds

Total: 715.791077milliseconds

Release:

Update - 6.519125milliseconds Draw - 53.112968milliseconds

Total – 59.632092milliseconds

(Similar)

8) Matrix multiplication resolve in draw and update temporaries

Debug:

Update - 156.538910milliseconds Draw - 219.252167milliseconds

Total: 375.791077milliseconds

Release:

Update - 4.69152milliseconds Draw - 48.213698milliseconds

Total – 52.905218milliseconds

9) Bench Marks

Update - 26.538910 milliseconds Draw - 203.192398 milliseconds

Total: 229.550873 milliseconds

Release:

Update - 1.775100 milliseconds Draw - 41.486021 milliseconds

Total – 43.261121milliseconds

REFLECTION:

Final result:

Update - 26.538910 milliseconds Draw - 203.192398 milliseconds

Total: 229.550873 milliseconds

Release:

Update - 1.775100 milliseconds Draw - 41.486021 milliseconds

Total – 43.261121milliseconds.

- My approach in understanding and cleaning the code is right.
- SIMD should not be implemented in very first phase of the project. It messes up with other things and makes more confusion to optimize.

- Once the code is cleaned and loos crisp, then make you alignment and next think about SIMD.
- Tune the compiler to yield fast results, you can tune according to your project.
- Converting types from double to float gives a lot difference in optimization.
- One should not only look about syntaxes and programming optimization, we should get the concept so that we can optimize more.(If you know about TRS matrices and world matrices, you would not need all matrices to finish the project)