Analysis paper

My optimization plan for the game particle system including using floats instead of double, doing RVO optimization, define big 4 and overload for the necessity, adding const to function signature, using std::vector instead of std::list, using SIMD for math operation, optimization for cache friendly data structure and working on alignment of private member variable to shrunk to data structure, general optimization to function body such as optimization for variable declaration based on their lifetime scope, invariants in the loop, and finally removing unnecessary code including assignment, math operation and member variable.

Since double is 8bytes but float is 4bytes, though might lose some precision but it significantly improves the performance by replacing all double into float. In addition, I changed all assignment to double into float by adding f at the end of double to prevent unnecessary implicit conversion. The openGL also provides function for float input parameter, I also apply them to the original code.

Then by using std::vector instead of the std::list because list is not contiguous as std::vector and std::vector is way much better in the performance in this case we don’t add or remove the element in the middle of the container. This is also a huge improvement to performance.

Defining big4 to default in header file allows compiler to do the optimization and we no more need the same big4 as default written in cpp files. Then I use initializer list in constructor and copy constructor, in this way it has a better performance than initialize member variable in function body. It is a slightly improvement on performance.

Adding const to the function signature is also a friendly behavior to compiler. It helps compilers to recognize and understand the function behavior and their inputs behavior. Key word const guarantee the safety and slightly better performance.

Doing the return value optimization allows to reduce the unnecessary temporary variables created in the function body. It is a middle level of performance improvement. Most of this technique I apply them to the Vect4D math operation.

By using the SIMD, it allows multiple data to do the math operation in single instruction. It is a relevant big performance improvement. I apply this technique in Matrix operation. For using this I have to ensure the data is align16.

Therefore, I work on the data structure of those 4 cpp files. The Matrix and Vect4D don’t need too many modifications because they only have float as member variables. I create a union for the better use in SIMD operation. For the Particle and ParticleEmitter, I have to work on the data structure because they are too bloated. I remove the member variables which never change to shrunk the size of data structure and then reorder the sequence of member variable based on their calling by the function and ensure they are cache friendly and have a good alignment. This should be a middle lower level of performance improvement.

Finally, I read the function body and remove the unnecessary and unused variables and code such as identmatrix multiply other matrix. Remove identical assignment to member variable. Move the invariants outside the loop. All for reducing the unnecessary code for program to run. It is a slightly improvement on performance.