**Documenting of Framework Optimisation:  
Ryan Buxton – B020612e**

***Table of Figures***

[Figure 1: Linked List Timer Structure 1](file:///C:\Users\Maverick\Source\Repos\LowLevelProgramming_Assignment\FrameworkOptimisation_WriteUp.docx#_Toc498462578)

[Figure 2: Frametime in milliseconds as a result of the LinkedList Timer displayed using the RTA Visualiser. (183.249ms) 1](file:///C:\Users\Maverick\Source\Repos\LowLevelProgramming_Assignment\FrameworkOptimisation_WriteUp.docx#_Toc498462579)

[Figure 3: Frametime in milliseconds as a result of the std::map Timer displayed using the RTA Visualiser. (77.7902ms) 2](file:///C:\Users\Maverick\Source\Repos\LowLevelProgramming_Assignment\FrameworkOptimisation_WriteUp.docx#_Toc498462580)

***Data Structures***

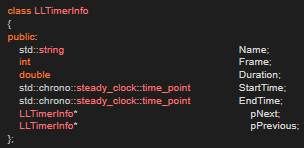
When I discovered that the application ran faster using an array of Spheres versus an std::vector<Sphere\*>, I decided that I would see if I would attempt to optimise the use of my Timers which featured std::vector<double> and std::map<std::string, TimerInfo> to use a LinkedList instead. I figured the reason the std::vectors would be slower is because it would have significant logic around its’ structure to handle such things as error checking, which the native C++ array lacks. I developed a Linked List structure which contains relevant storage for time-reporting information as well as pointers to the next and previous node (See left).

Figure 1: Linked List Timer Structure

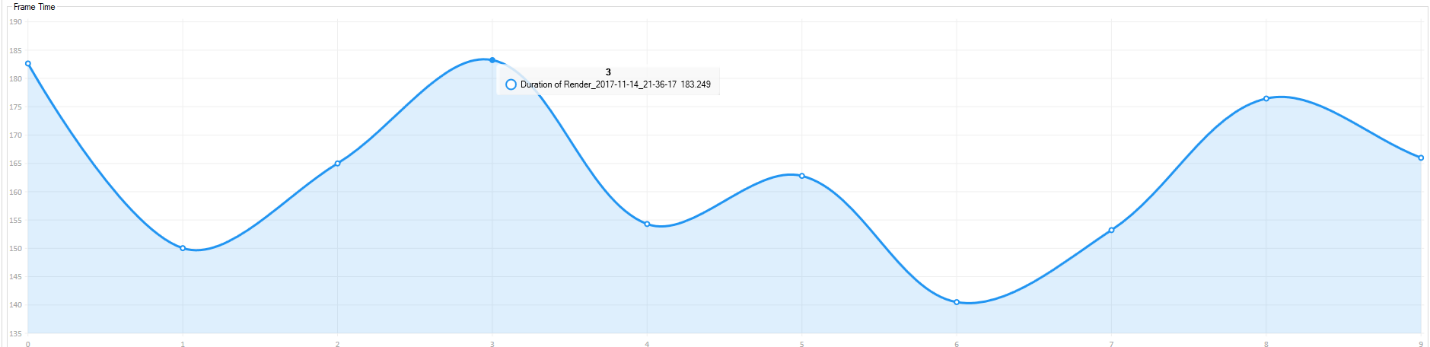
However, to my surprise this increased the time spent rendering each frame. I believe this is because of the recursive methods defined for a LinkedList Timer Class (of which there were three instances of measuring the running times of a frame in rendering, each thread created for the rendering (each frame) and the running times of File IO operations). So with this decrease in effective performance (which can be seen below) I decided to revert back to the old method of using std::maps and std::vectors to track the running times.

Figure 2: Frametime in milliseconds as a result of the LinkedList Timer displayed using the RTA Visualiser. (183.249ms)

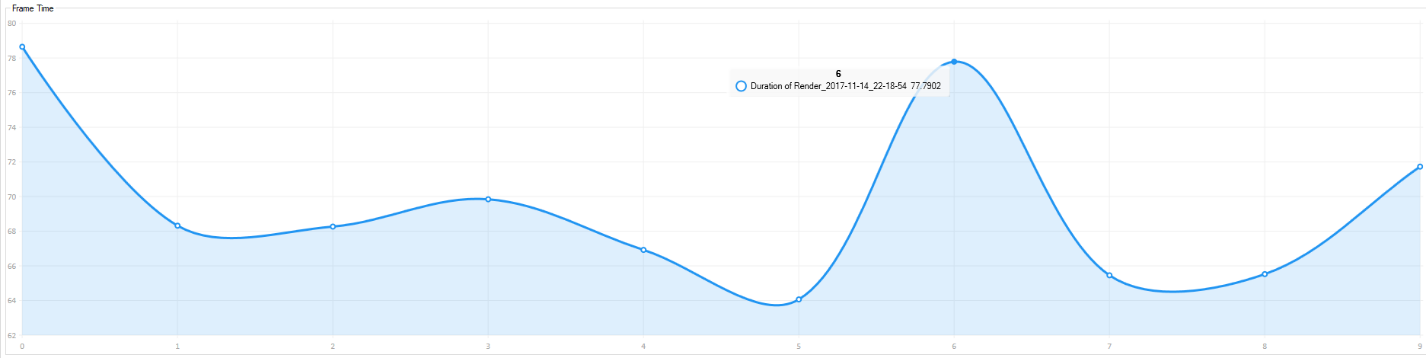


Figure 3: Frametime in milliseconds as a result of the std::map Timer displayed using the RTA Visualiser. (77.7902ms)

***Compiler Optimisations***