

GPP Assignment Technical Report

Individual Architecture Specific Coding Optimizations Assignment

LA TROBE UNIVERSITY

2009

Authored by: Jason Thompson 15310338

GPP Assignment Technical Report

Individual Architecture Specific Coding Optimizations Assignment

AIMS

The aim of the assignment is to:

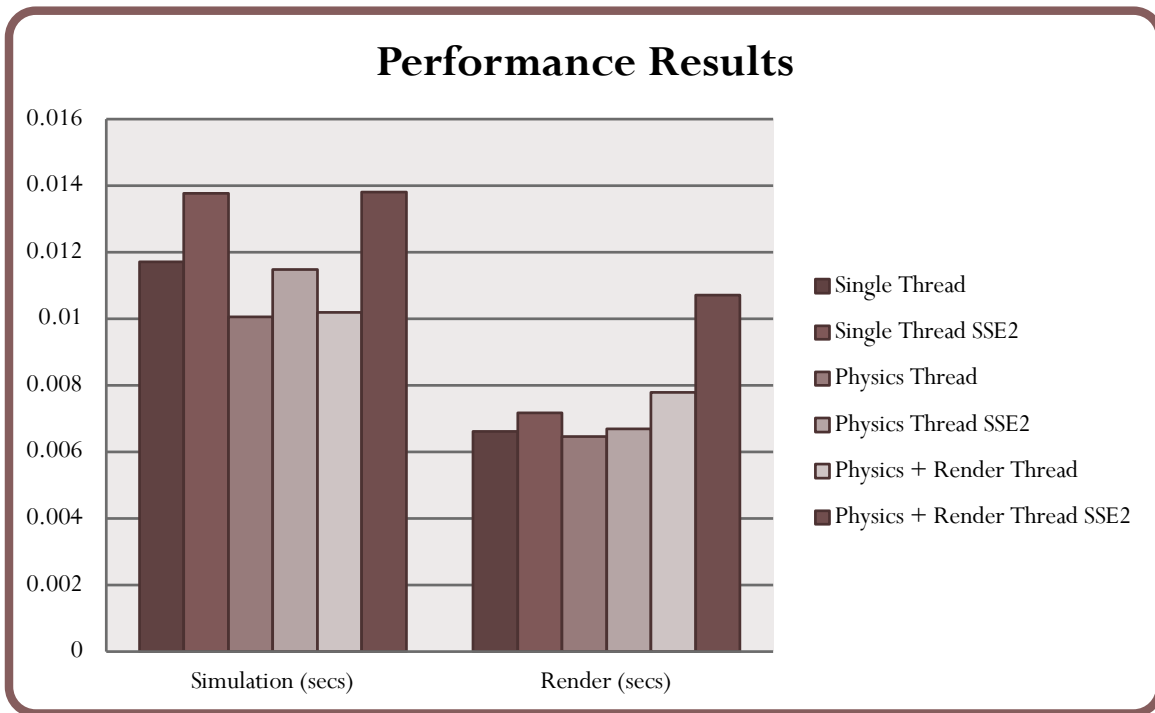
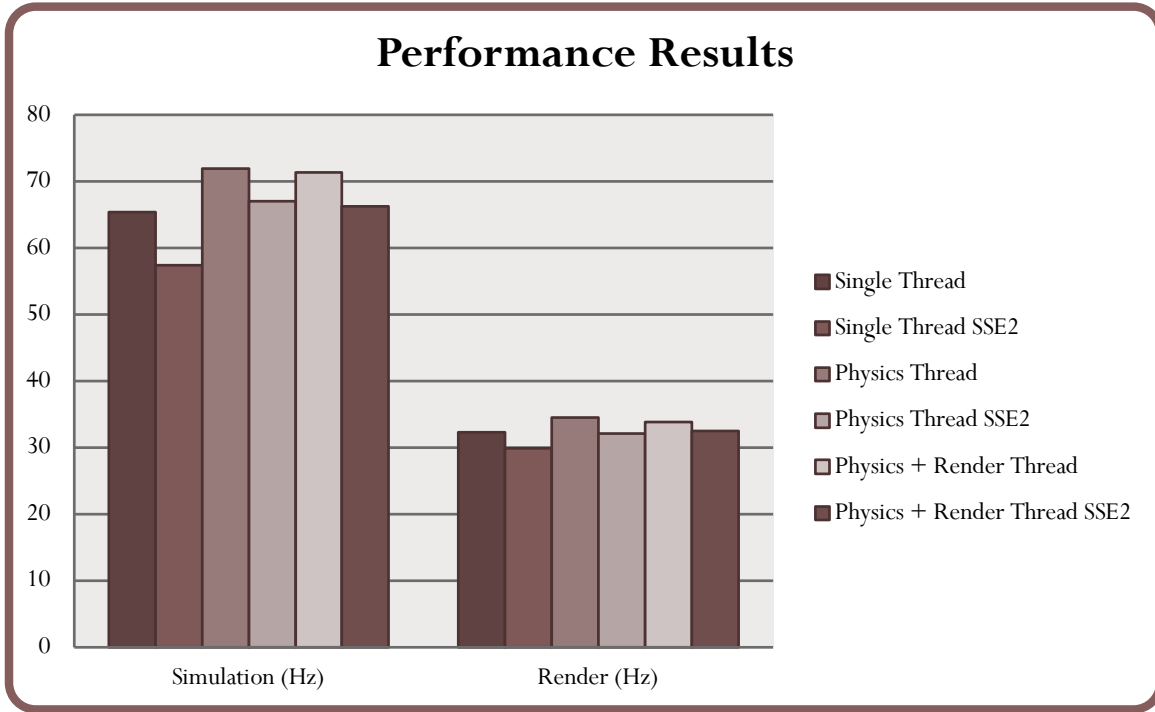
1. Implement a physics simulation that without optimisations does not produce useable results.
2. Implement a real-time graphics representation of the physics simulation.
3. Optimize the implementation using multi-threading (a) and SIMD instructions (b).
4. Optionally implement other optimisation such as more advanced threading techniques like thread pools.

METHOD

This assignment will meet the aims by:

- Aim (1) is met by implementing a simple fluid physics simulation, similar to “*Globular Dynamics: A Connected Particle System for Animating Viscous Fluids (1989)*” Gavin Miller and Andrew Pearce. In this case the fluid is modelled by particles which exhibit a close range repulsion force, medium range attraction force, no forces at a long range and a medium range velocity dependant drag force. The algorithm is $O(n^2)$ where n is the number of particles. As a result of the algorithms computational complexity its un-optimised performance should be very low for large numbers of particles.
- Aim (2) is met by using *OpenGL* to render the particles in the as spheres.
- Aim (3a) is met by using *WinThreads* to multi-thread the algorithm. In order to do this, two copies of particles are kept. One is used by the physics threads the other is used by the render thread. Each physics thread solves a portion of the total particles. The render thread renders its copy of the particles. Once the render thread and physics threads finish their respective tasks they swap their respective copies of the particles and continue to execute. Aim (3b) is solved by using the SSE2 instruction set to re-implement the vector operations used in the algorithm.
- Aim (4) is met by implementing a thread group module which groups together a number of threads that execute the same function. It provides functionality to execute threads and wait for them to synchronise.

RESULTS



DISCUSSION

The results are for 640 particles running on the following system:

```
Time of this report: 5/29/2009, 06:26:01

Machine name: CS-BG115-34

Operating System: Windows XP Professional (5.1, Build 2600) Service Pack 3
(2600.xpsp_sp3_gdr.090206-1234)

Language: English (Regional Setting: English)

System Manufacturer: INTEL_

System Model: D945GTP_

BIOS: Default System BIOS

Processor: Intel(R) Pentium(R) 4 CPU 3.20GHz (2 CPUs)

Memory: 1022MB RAM

Page File: 804MB used, 1654MB available

DirectX Version: DirectX 9.0c (4.09.0000.0904)
```

It shows that the overhead of running threads has a negative impact on the performance of the simulation. The cost of rendering a frame of the simulation remains constant. This is consistent with expectations as rendering can only occur from one thread. The SSE2 instructions produce a noticeable increase in processing time of the physics. This is possibly because of hardware limitations. SSE2 might have a large memory latency on this system i.e. takes a long time to load from main memory to registers. The performance hit could also have been caused by poor optimization of the intrinsics by the compiler i.e. excessive unnecessary transfers of data to and from registers and main memory.

CONCLUSIONS

On the lab computers there was no performance increases. However, when the application was run on a quad-core computer it showed a significant performances. Unfortunately no direct access to a quad-core was available, it had to be sent via email to another computer for some else to test, hence the lack of solid results.