

Proposal: Implementing and Accelerating 3D Geometry Transformations with MMX™ Technology

Pei Qi and Yang Wang Electrical&Computer Engineering, University of Wisconsin-Madison

Motivation

3D objects are represented by thousands of polygons. Each vertex of polygons is represented by a 4-element vector. When a 3D object performs a geometry transformation, such as shift, rotation, expansion and so on, we have to recalculate polygon shape which is actually matrix multiplication. This operation will definitely incur a computation overhead since each vertex pixel requires 16 multiplies and 12 additions. However, there is also a lot of inherent parallelism which could be exploited to speedup the recalculation process.

The Intel Architecture MMX Technology provides a significant increase in processing power, through the use of its new single-instruction multiple-data (SIMD) extensions. MMX instruction set features 57 SIMD-type instructions and eight 64-bit wide MMX registers. In addition to that, 4 new data types are introduced into MMX instruction set to accelerate the processing by calculating in parallel. Thus, two 32-bit, four 16-bit, or eight 8-bit integer values can be operated on at once. The pixel of a digitalized image is typically represented as 8-bit integer, this enables us to take advantage of the characteristics of MMX instruction and thereby accelerate 3D geometry transformation processing.

Approach

First of all, we create an appropriate test case that is a straightforward MMX routine and should be easy to understand and debug. This enables us to examine the inside details regarding to MMX instruction execution. At the end of execution, we will record the execution time to be compared with the result achieved from optimized code in next step.

We will be attempting to optimize that MMX routine in successive steps. Like other assembly code routines, MMX can also be manipulated to achieve significant speedup. Firstly, we will try

to re-order instructions to gain benefit from pipelining. Pipeline technique enables more than one instruction to execute simultaneously as long as there are no data dependencies existed between those instructions. Thus, the execution time could be reduced significantly. In addition, we may try unrolling loops if necessary, because that will be able to reduce the average cost-per-loop of jump instructions. At last, we record the execution time after doing optimization.

Expectation

So far, we do not know exactly how much we can accelerate the 3D geometry transformation when implementing MMX instruction and optimizing the MMX routine. But we expect that the optimized MMX code could run at least two times faster than un-optimized code.

Task planning

Our group consists of Pei Qi and Yang Wang. Each person will be assigned equal amount of workload.

References

- [1] W. Ma, C. Yang, "*Using Intel SIMD Extension for 3D Geometry Processing*", 2000
- L. Gwennap, "*Intel's MMX Speeds Multimedia*", Vol 10(3), 1996
- [2] C. Yang, B. Sano, A. R. Lebeck, "*Exploiting Parallelism in Geometry Processing with General Purpose Processors and Floating-Point SIMD Instructions*", Vol 49(9), September 2000
- [3] K. Akelay, T. Jermoluk, "*Hight-Performance Polygon Rendering*", In *Computer Graphics*, pages 239-246, 1988
- [4] H. Nguyen, L. K. John, "*Exploiting SIMD parallelism in DSP and Multimedia Algorithm Using AltiVec Technology*", In ICS99, 1999