The Code and Movie Clips
for
"Medical Image Reconstuction with the FFT"

A chapter in the book: "GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation," Addison Wesley Professional

Thilaka Sumanaweera and Donald Liu
Siemens Medical Solutions USA, Inc.
1230 Shorebird Way
Mountain View, CA 94039, USA
Thilaka.Sumanaweera@siemens.com
12/15/2004

What is in this folder:

This folder contains the following items:

1. A bin folder containing three executables: FFTDemo256x256.exe, FFTDemo2048x64.exe and MRI256x256.exe and all the needed dlls. The first executable loads two 256x256 floating-point complex images containing a sync function into the GPU and performs the 2D FFT over and over again and prints the frame rate. The second does the same for two 2048x64 images. The last executable loads two separate 256x256 MRI data sets acquired in the Fourier domain: a beating mouse heart and a set of slices through a human head. It then loads the two data streams dynamically, performs the 2D FFT to reconstruct the data and displays one of the streams. You can press '1' to switch the stream that is displayed. In each case press 'h' to get a help line.
2. A library called FFT containing the code to perform 2D FFT in NVIDIA GPUs.
3. A library called Pbuffer for creating pbuffers.
4. Include and lib folders.
5. A workspace called FFTDemo that produced the executables, FFTDemo256x256.exe and FFTDemo2048x64, demonstrating the use of the FFT library above.
6. A workspace called MRI that produced the executable MRI256x256.exe, demonstrating MRI reconstruction of acquired Fourier-domain data using the FFT library.
7. A folder called Movies containing AVI files, *MouseOut.avi* and *HeadOut.avi*, showing the reconstructed MRI data.

Requirements:

1. This code has been successfully compiled using Visual Studio 7.0.
2. It assumes that you have installed Cg Release 1.2 or better.
3. The code is currently only implemented for NVIDIA (NV40 and above) GPUs. It is theoretically possible to run this code on other hardware with minor changes.
4. To get the best performance, you would need the Quadro boards (as of 12/15/2004). This allows you to use 8 draw buffers: GL_FRONT_LEFT, GL_BACK_LEFT, GL_FRONT_RIGHT, GL_BACK_RIGHT, GL_AUX0, GL_AUX1, GL_AUX2 and GL_AUX3, which is crucial for the fast performance.
5. If you don't have a Quadro board, it will also run slower.

Instructions:

1. If you are using a Quadro board, turn on the Stereo OpenGL mode from the Display Properties in Control Panel (Display Properties->Settings->Advanced)
2. Run each executable form the 'bin' folder.

Notes

1. The code in this folder was tested using NVIDIA driver 7.0.4.1 (beta), released on 10/9/2004 on a Quadro FX4000.
2. The FFT library queries the GPU to see if it is a Quadro board. If it is, the application will automatically use 1 pbuffer with 8 draw buffers. If not, it will use 2 pbuffers with 4 draw buffers, in which case, the performance will be lower due to context switching. You can also override the automatic selection process and specify whether you want to use 1 pbuffer or 2 pbuffers when you create the FFT class.
3. You can specify whether to do forward or backward FFT.
4. The code has two methods to compute the FFT:
    a. Method 1: Mostly loading the fragment processor
    b. Method 2: Loading the vertex processor, the rasterizer and fragment processor

    Method 1 produces faster frame rates for early butterfly stages. Method 2 produces faster frame rates for the later butterfly stages. This implementation picks the optimal transition point between method 1 and method 2 automatically by exhaustively searching all possible transition points at the beginning, thus balancing the load amongst all accessible processors in the GPU. A blue screen will be displayed during this process. Please wait until the optimization process is complete. You can also force your own transition point.