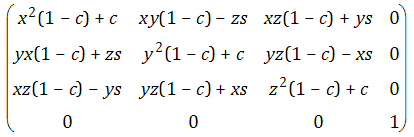
**CS 525G Programming Assignment 1. Due before class on Wednesday Oct 12**

**Overview**

This assignment requires you to implement CPU and GPU versions of a function that calculates the elements of a 4x4 rotation matrix and prints out their values.

**Background**

The rotation 4x4 matrix below rotates a 3D point (Px, Py, Pz) by **angle** degrees about the arbitrary axis **(x, y, z)**.



****

**The Assignment**

1. Create a new Visual Studio 2008 solution to implement the assignment. You will submit this solution and associated files in a zip file.
2. Write a C function **bae\_rotation( )**, which takes as input the **angle** and the arbitrary **axis** **(x, y, z)** then calculates and returns the 16 elements of the 4x4 rotation matrix. Note that the axis (x,y,z) should be normalized (magnitude of (x,y,z) should be 1) before using it in all calculations. If (x,y,z) is not normalized, do so before starting your calculations.
3. Write a **C main( )** function that calls your **bae\_rotation( )** function above on the CPU and prints out the actual matrices generated by **bae\_rotation( )** for the following values of **(angle, x, y, z):**

(10, 0, 1, 0)

( 34, 0, 0, 1)

( 64, 1, 0, 0)

(164, 0, 0, 1)

(94, 0, 1,0,1)

(252, 0, 0.25, 1.4)

(45, 0.6, 0.8, 1.45)

(198, 0.2, 0.65, 1.75)

(204, 0.5, 0.2, 1.56)

(213, 1.06, 0.6, 1.68)

(346, 0.78, 0.34, 1.6)

(68, 0.45, 0.7, 1.82)

(192, 0.64, 2.68, 3.96 )

( 75, 0.4, 0.35, 1.89)

(29, 1.78, 0.65, 1.92 )

(43, 0.45, 0.88, 0.38 )

(144, 1.29, 9.68, 0.69)

(189, 8.58, 0.46, 0.63)

(249, 9.78, 0.44, 0.56)

(56, 0.78, 5.46, 3.6)

1. Write a GPU kernel **gpu\_bae\_rotation1( )*,*** that does the same calculations but fills out the 4x4 rotation matrix on the GPU. Set things up so that each of the 16 elements of the matrix is calculated by a different thread. In this version, copy one set of inputs to the GPU, calculate the matrix elements and copy back the resulting matrix before going on to the next set of inputs. i.e. fill out the matrices one at a time. i.e. fill out matrix one, then fill out matrix 2, etc.. up till matrix 20 above.
2. Write a second version of your GPU kernel **gpu\_bae\_rotation2( )**, that spawns enough threads to calculate all 20 rotation matrices in parallel. i.e. copy all 20 arguments to the GPU in a single CPU-GPU data transfer, spawn enough threads to calculate all 20 matrices in parallel and copy back results of all resulting matrices in a single GPU-CPU data transfer. Use multiple blocks of threads for this version such that each block of threads has at most 32 threads. Each of the 16 elements of all 20 matrices is calculated by a different thread.
3. Finally, set up your main function to call the CPU rotation function as well as the 2 GPU functions. Print out the set 3 rotation matrices (1 CPU + 2 GPU) for each of the inputs above. For each input print out the function inputs **(angle,x,y,z)** followed by the corresponding matrices for the CPU and 2 GPU versions of the rotation implementation. After each set of results is printed, wait for the user to hit <ENTER> before displaying the next result.

**Checking your answer:** Check your program’s output on two levels:

1. Using any of the input arguments, calculate the elements in the output matrix using a simple hand calculator. Check that your program generates the expected results for all 16 elements.
2. Compare the output of the CPU and 2 GPU implementations. Make sure that the generated rotation matrices are the same.

**Submitting**

Test your program rigorously, clean up and comment your code properly. Write a simple README file in either ASCII text or in Microsoft Word. Your documentation doesn’t have to be long, but it should describe the main files generated and functions you implemented. Zip up your entire solution, files and README file and email it to me ([emmanuel@cs.wpi.edu](mailto:emmanuel@cs.wpi.edu)) by the due date. Name your zip file as your FirstName\_lastName\_cs525g\_prog1.zip