Prof. G. Zachmann
D. Mohr

University of Bremen
School of Computer Science
CGVR Group
July 2, 2013

Summer Semester 2013

Assignment on Massively Parallel Algorithms - Sheet 5

Due Date 29. 05. 2013

Exercise 1 (Self-assignment, *10 Credits*)

Think about a small CUDA project that is both interesting and challenging with respect to the massively parallel programming paradigm. Present your idea in the next tutorial (1-2 slide). Your presentation should include a motivation for your exercise, a rough idea on how to parallelize it, and a short time schedule for the implementation. Also, please state clearly what is the output and required input data of your program.

Every week, you have to show at least some substantial progress with your CUDA program, which should at least compile and run without crashing when you show it in the tutorial.

Due date for the submission and presentation of the full implementation is 03.07.2013.

After presentation, you have to send the *well documented* source code including a readme file to your tutor (Daniel Mohr, email: mohr@informatik.uni-bremen.de).

The readme file includes at least the following:

1. Which problem does your program solve
2. Inputs and outputs of the program
3. How to use the program (e.g. parameters, command line, usage during runtime, hot keys, necessary hardware)
4. Additional library dependencies
5. Screenshot and caption

We plan to put the best projects (including your names if you want) on the lecture webpage, if you give us the permission.

The full implementation and functionality will be graded with 60 credit points.

If you really have no idea about an exercise, you can implement one of the following tasks:

a) (+++) Write a face recognition application using classification algorithms from machine learning e.g. Boosting, Support Vector Machine (SVM) or Random Forests.
b) (+) Write a high quality tutorial on how to efficiently develop Cuda applications on MacOS, including syntax highlighting, code auto completion, debugging and profiling.
c) (++) Write an OpenGL program to render an arbitrary object (you can use an OBJ-loader to load meshes). This object should then be textured using procedural textures. Generate the

texture in an Cuda program. For efficient transfer between Cuda and OpenGL, use the OpenGL-Cuda-interoperability functions that allows to share device memory between OpenGL and Cuda. Do *not* copy the texture back to the main memory on the host at any time!

Analog to the procedural texture, generate a procedural mesh (e.g. Steinbach screw). Texture the mesh with the texture!

d) **(++)** Write a procedural 3D Texture i.e. volume generation. For visualization, you can put a teapot into the volume and texture the teapot with the intersection area (this is a 2D subset of the 3D volume).

e) **(++)** Develop a raytracer that uses multiple rays (i.e. reflection and refraction). Note that the recursion in cuda is the main difficulty.

Note that this are just outlines of projects, you have to discuss the details with your superviser.