2017.1 Human Media Multicore Computing

Project 3: CUDA/OpenMP Ray-Tracing (Team Project)

(Deadline : June 8th 11:59pm)

Summision method : eClass

- All students need to submit a zip file containing (i) final report, (ii) source codes (openmp_ray.c, cuda_ray.cu), (iii) executable codes, (iv) README.txt

Step 1. make a team

- team size : 1~2 members

Step 2. Implementing two versions of Ray-Tracing (openmp_ray.c, cuda_ray.cu) that utilizes CUDA and OpenMP, respectively.

- Look at the code [raytracing.cpp] for ray-tracing of random spheres, which is available on our class webpage, and modify the code for your purpose.
- You may assume the simplest form of Ray tracing that renders a scene with only spheres.
- Program input :
  (i) openmp_ray.c: [number of threads] [output filename]
  (ii) cuda_ray.cu: [output filename]
- Program output :
  (i) print ray-tracing processing time of your program using OpenMP or CUDA
  (ii) generate image file (format: .ppm or .bmp, image size: 2048X2048) that shows the rendering result

Execution example of cuda_ray.cu) > cuda_ray.exe result.ppm
CUDA ray tracing: 0.15 sec
[result.ppm] was generated.

Execution example of openmp_ray.c) > openmp_ray.exe 8 result.ppm
OpenMP (8 threads) ray tracing: 0.41 sec
[result.ppm] was generated.

Step 3. write a final report (pdf) that includes

- project title, member list (name and student id)
- execution : describe (i) execution environment (OS type, CPU type, grahics card/GPU type, memory size) (ii) how to compile, (iii) how to execute
- your group's contribution (describe exactly what your group actually did for this project)
- entire source code and detailed explanation on the openmp_ray.c and cuda_ray.cu codes
- other implementation issues (describe how you implemented)
- program output results including screen capture pictures. (Insert sufficient number of screen captures showing the various output results)
- experimental results : measuring the performance (execution time) of your OpenMP/CUDA implementation and your single threaded CPU implementation. Show the performance results with respect to various number of threads. Include screen captures of output results.
- conclusion : summarize your project result

Step 4. submission (to eClass)

- final report (should include a list of team members with student ID#)
- source code files
- executable files
- README.txt file (describe (i) execution envionment (OS type, CPU type, GPU type, memory size) (ii) how to compile, (iii) how to execute)