HW3: Mandelbulb Set

Due: Fri., 2025/10/31 23:59

Problem Description

In this assignment, you are asked to implement the simplest Ray Marching algorithm to render the "Classic" Mandelbulb and accelerate your program using cuda and GPU. We hope this assignment helps you to learn the following concepts:

- How to write a cuda program
- How to distribute workloads efficiently on the GPU
- The importance of Load Balancing.
- The importance of parallel algorithms and libraries.

Please note that you are not allowed to use mathematical techniques to bypass or reduce the computational complexity and the iteration process of calculating Mandelbulb sets (i.e. do not modify the parameters in the sample code).

Provide Materials

- A sequential version of Mandelbulb rendering is provided at `/work/b10502010/pp25/hw3/hw3_cpu.cpp`
- The default makefile is provided at`/work/b10502010/pp25/hw3/Makefile`
- The public test cases are provided at `/work/b10502010/pp25/hw3/testcases/`

Environment Setup and Sequential Code

1. Copy the folder with the provided materials with the following command:

```
cp -r /work/b10502010/pp25/hw3/ ~/hw3
```

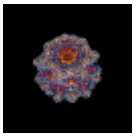
2. Load the required modules for the program:

```
module load cuda
```

3. Compile and execute the example code:

```
cd ~/hw3
make cpu
srun -N 1 -n 1 --gpus-per-node 1 -A ACD114118 -t 3 ./hw3_cpu -0.522
2.874 1.340 0 0 0 64 64 00.png
```

4. The output image will be saved in 00.png as follows:



It may take a few hours to run large cases using sequential code. Please start with small cases first and remember to set a time limit.

Program Execution and I/O Specifications

The program is executed with the following command:

```
./executable [x1] [y1] [z1] [x2] [y2] [z2] [width] [height] [filename]
```

The types of the arguments and their meanings are listed in the table below:

Argument	Type	Explanation
$x1	double	Camera position x
$y1	double	Camera position y
$z1	double	Camera position z
$x2	double	Camera target position x
$y2	double	Camera target position y
$z2	double	Camera target position z
$width	unsigned int	Width of the image
$height	unsigned int	Height of the image
$filename	string	File name of the output PNG image

Please note that the output image should be a 32-bit PNG image with RGBA channels.

Parameter For Mandelbulb Rendering

The table below lists the parameters, along with their default values and explanations, in the sequential version.

Parameter	Default Value	Explanation
power	8.0	Power of the equation
md_iter	24	The mandelbulb's maximum iteration count
ray_step	10000	The maximum step count of ray marching
shadow_step	1500	The maximum step count of shadow casting
step_limiter	0.2	The limit length of each step when casting a shadow
ray_multiplier	0.1	A multiplier for ray marching to prevent over-shooting
bailout	2.0	The escape radius
eps	0.0005	The precision of the rendering calculation
FOV	1.5	Field of view
far_plane	100	The maximum depth of the scene

Library Used

The following libraries are the ones used for the sequential version. Please refer to TA's sample code and Makefile. You are free to implement your own computation logic in hw3.cu if you wish.

- lodepng - loading/saving PNG images.
- GLM - vector/matrix arithmetic.

Output Verification

You can use `hw3-judge` to check the correctness of your program on all test cases and submit your result to the scoreboard http://140.112.187.55:7771/scoreboard/hw3/.

Please note that your homework is graded based on the code submitted to NTU Cool, the statistics on the scoreboard will not be considered, and also remember to load the modules mentioned above before using the auto-judge.

Report

Your report must contain the following contents, and you can add more as you like. You can write in either English or Traditional Chinese:

1. Explain your implementation, especially in the following aspects:
    - How did you implement your program using cuda?
    - How did you partition the tasks among GPU threads and blocks?
    - Other optimization skills in your program.
2. Analysis:
    - Measure the GPU kernel execution time using nvprof.
    - Profile your program with Nsight Compute (ncu). Design and present your own plots or figures to show the performance difference under different GPU kernel configurations (e.g., block size, grid size).
    - Include any additional analysis or observations you find meaningful.
3. Conclusion:
    - What difficulty did you encounter in this assignment?
    - Any feedback or suggestions for this assignment or spec.

Grading

1. (30%) Correctness.
    - 8 public cases [01-08] + 2 private cases
    - Proportional to the number of test cases solved.
    - If the output PNG file of your program fails to match **97%** of the ground truth, you will get 0 points for that test case.
2. (40%) Performance.

    Note that you are required to produce the right answer to get the performance points.
3. (30%) Report.

Submission

Please place the following files into a directory named <studentID>, and archive the directory as <studentID>.tar:

- `hw3.cu` – the source code of your implementation.
- `report.pdf` – your report.

- `Makefile` – optional. Submit this file if you want to change the build command.

Please note that the first character of your student ID in the file name MUST be lowercase, and you should run the archiving command on the directory (also in lowercase), not the files themselves. Any submission violation would lead to a point deduction.

For instance, the correct way to archive your files is as follows:

- `mkdir b10000000`
- `cp <your files> b10000000/`
- `tar cvf b10000000.tar b10000000`
- Submit the file `b10000000.tar`
- Your directory should look like this:

```
b10000000/
├── hw3.cu
├── Makefile
└── report.pdf
```