GPGPU Assignment #2

Author: Yu Sheng Lin Instructor: Wei Chao Chen

April 7, 2018

1 Goals

Make a video with CUDA in this assignment. Be creative – this is an open-ended assignment.

2 Requirements

The requirements are very loose. You just need to render a series of frames with CUDA, with as little input data as possible. For example, you can try to render Perlin Noise, Simplex Noise, or Stable Fluid. Be more artistic and try to render fireworks, ocean or even a short movie. Google "Demoscene" and see some extreme examples about what people can do with very small executables.

2.1 Video Format

In this lab we will first call `void get_info(Lab1VideoInfo &info)` to get the height H, width W, FPS N/D [1] and number of frames N_f of your video. Then we will call your `void Generate(uint8_t *yuv)` N_f times to get all the frames for your video.

Instead of storing RGB values directly, please use the YUV color space, which is the commonly used color space for video codecs.

$$\begin{cases} Y = +0.299R + 0.587G + 0.114B \\ U = -0.169R - 0.331G + 0.500B + 128 \\ V = +0.500R - 0.419G - 0.081B + 128 \end{cases} \tag{1}$$

We also subsample U and V channels both horizontally and vertically, that is, $\frac{W}{2} * \frac{H}{2}$[2]. The YUV channels are stored sequentially so the total size of a frame is $1.5WH$.

2.2 Sample code

The sample code (Listing 1.) generates a grayscale video which is black initially and gradually becomes lighter. "Grayscale" means $R = G = B$, so the YUV is:

$$\begin{cases} Y = +0.299R + 0.587R + 0.114R = R \\ U = -0.169R - 0.331R + 0.500R + 128 = 128 \\ V = +0.500R - 0.419R - 0.081R + 128 = 128 \end{cases} \tag{2}$$

[1] We use a ratio to represent the FPS because we are using the y4m format. If you want 24 FPS, you shell returns $N = 24, D = 1$.

[2] W and H must both be even.

```
void Lab2VideoGenerator::Generate(uint8_t *yuv) {
    cudaMemset(yuv, (impl->t)*255/NFRAME, W*H);
    cudaMemset(yuv+W*H, 128, W*H/2);
    ++(impl->t);
}
```

Listing 1: Sample code explanation

```
avconv -i output.y4m output.mkv
```

Listing 2: Sample code explanation

The Y channel is $W * H$, so we first calculate the brightness according to the frame index then fill it into the device pointer.

The U and V channels are $\frac{W}{2} * \frac{H}{2}$ each, so we fill 128 to the following memory space. Finally, we increase the frame index by one.

The output file is of the y4m raw video format (it can be very large!) and you can use software such as FFmpeg or Avconv to convert it to other compressed formats (Listing 2.).

Hints: We do not allow you to modify the header, and if you don't know how to achieve that, please read Pimpl Idiom or Opaque Pointer.

3 Submission

- The submission deadline is 2018/4/10 23:59.
- Apart from submitting your code in time, you must also post the link on the course Facebook group AND fill the Facebook link in this form. As there can be delays with Facebook posts, we won't be super strict about the deadline, but please do make sure to complete the spreadsheet on time.
- We won't run your code but you still have to submit it in time. TA will still clone your code and look into your code if it becomes necessary. (We expelled a few students owing to plagiarism in last year.)
- Do not add ANY video in your Git HISTORY, or you will suffer 20% penalty. We recommend you to use the `.gitignore` file.
- Your grade for this assignment is determined by both "techniques" and "aesthetics". You may also promote your work on Facebook and we will consider giving you bonus points based on your post popularity.
- For those who don't know how to get the link for a video, just right click on the video.
- You can add music or text to the video, and we do not allow post-processing of any other form, otherwise you will suffer from a 10% penalty.