605.617.81 FA21
Introduction to GPU Programming

Project Report

Seo, Stephen

December 13th, 2021

Contents

1 Abstract: What the Project is About

The project can be explained here (project proposal), but the end result is a little different than in the project proposal.

The project aims to implement dithering images via the usage of blue-noise images, and can be applied to dither images or video with the option of doing so in color or in grayscale. The project proposal mentioned dithering images in real time, but I ended up choosing to add the capability to dither input video clips instead.

1.1 Blue-Noise

Blue-noise is basically uniformly distributed white-noise. Before the course had started, I wanted to make a program that dithered images for me, and set up a private repo. Though I never ended up making dithered images in the end, only a program that generated blue-noise for the use of dithering, which can be found here. I felt that since I started that project before the course started, I couldn't use it as the course project. Though I felt it important to state here where the blue-noise images came from. Note that the blue-noise algorithm is somewhat explained in the link provided in the project proposal, and a reference implementation is also available at the same place. I also must note that I finished my blue-noise project after the course had started, which was good timing for me to generate my own blue-noise to use in my project.

1.2 How the Blue-Noise is Used

Take for example grayscale dithering. First, an offset into blue-noise is randomly generated (blue-noise is supposed to be tileable, so any offset should be fine, as repeatedly tiling blue-noise should not produce repeating artifacts (if you try different zoomed views of dithered images, you do see artifacts, so it's possible that my blue-noise generator has a flaw, or that's just how blue-noise is)).

The entirety of the image is then "compared" with the blue-noise by applying the blue-noise onto the image with tiling. The algorithm that determines whether or not a grayscale image's pixel is black or white is the following:

```
1 uint8_t pixel = input_image[image_idx] > blue_noise[blue_noise_idx] ? 255 : 0;
```

Listing 1: `Algorithm that decides if a grayscale pixel is black or white`

Since blue-noise ranges from 0 to 255 uniformly, the result is that images will have a "smoother" dithered appearance than applying just plain old random white-noise.

2 Project Result

Project Source Repository

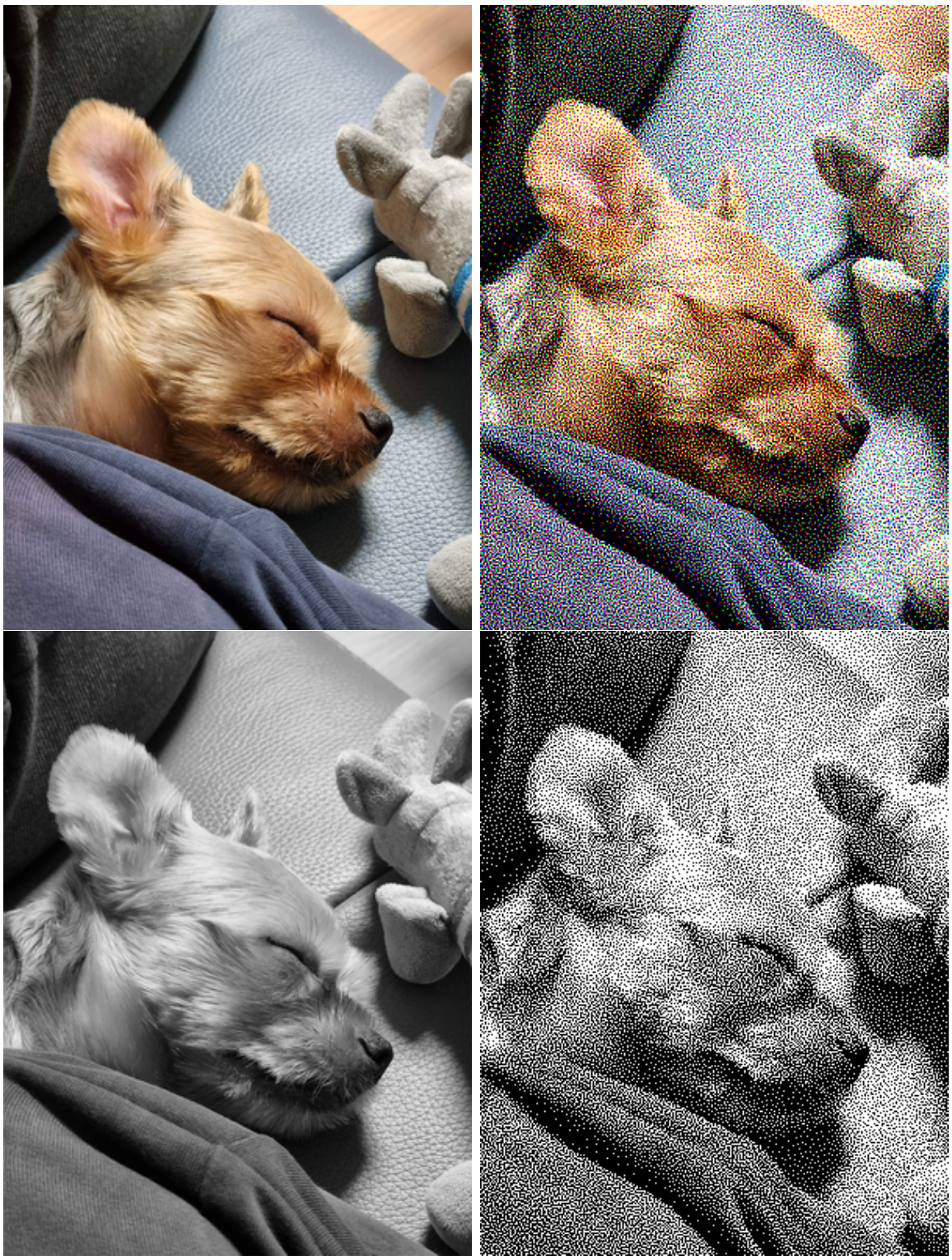

Figure 1: Original compared with blue-noise dithering of my pet dog Louie

I was able to implement dithering for images and video in grayscale or color. The caveat for color is that 8 possible colors are used: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta. Grayscale uses: Black and White. Image output to PNG is optimized to use a palette for color, and grayscale mode for grayscale (this is possible through the use of libpng, though it wasn't easy to figure out as libpng provides a guide pdf but not an API documentation page). When outputting to video, I merely set the quality settings high enough to preserve quality for Colored Dithered output. The result is that the video size may use a significant amount of disk-space. I'm not sure how to use a palette for H264 encoding, and I'm not even sure if that's possible, but for now I've just set it up such that video encodes with "good-enough" quality.

2.1 Video to PNGs

Note that the project also provides the ability to take video input and output each output frame as individual PNG files (which can be muxed together as described here with whatever settings one wants). Some reasons why I added this implementation is that I've had problems with different video sources outputing dithered video with the wrong FPS (frames-per-second), or bad quality (worse than generating each PNG and muxing them together).

3 Project Presentation

The project presentation can be viewed here.

4 Conclusion

The creation of a program that can dither images and video using OpenCL was mostly a success (different zoomed views of the dithered images may show tiling artifacts sometimes, and I haven't been able to fix this, but it isn't noticable if the image is viewed at its original dimensions), though I must say that working on my blue-noise generator on the side was helpful in figuring out OpenCL too. I leave both of these programs open-source in the hopes that they may be useful, as I personally think dithered images is a cool aesthetic to use.