Notes on Ray Tracing in One Weekend

Junyi Li

October 2022

1 Image Format

The book uses the PPM image format as a simple way of producing images from RGB values. This isn't that convenient in the long run, so I used SDL2 to create a window to view the image. There are three main entities in this process:

1. The SDL2 window and renderer
2. An *Image* which wraps a int buffer containing RGBA data
3. A *Scene* which contains objects in our world

The scene and its objects perform the actual ray tracing and write pixel colors to an image. That image buffer is then sent to SDL2 to be displayed. Everything is kept inside a main *App* class, with member variables *window_*, *renderer_*, *image_*, and *scene_*. The snippet below suggests how each component is initialized.

```
1 SDL_Init(SDL_INIT_EVERYTHING);
2 window_ = SDL_CreateWindow("Yet Another Ray Tracer", SDL_WINDOWPOS_CENTERED,
3                            SDL_WINDOWPOS_CENTERED, 1280, 720, 0);
4 renderer_ = SDL_CreateRenderer(window_, -1, 0);
5 image_.Init(1280, 720, renderer_);
```

On line 5, we are resizing the image's internal vector, and passing it a pointer to the SDL2 renderer. The latter is necessary because the Image class actually holds two buffers – a simple int vector for itself, and another *SDL2 texture*, which needs the renderer to be initalized. This texture is pretty much also an int vector, except we can't write to it directly. Note that the scene is not explicitly initialized here, as a default constructor suffices.

1.1 Pixel Data

Pixels are represented in either RGBA or ABGR format, depending on your platform. Each pixel is a 32-bit integer, with 8 bits per channel. Assuming we already have the values of each channel, we can write the pixel using the correct format like so:

```
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
    uint32_t pixelColor = (r << 24) + (g << 16) + (b << 8) + a;
#else
    uint32_t pixelColor = (a << 24) + (b << 16) + (g << 8) + r;
#endif
```

To send the pixel data to SDL2:

```
SDL_UpdateTexture(texture_, nullptr, pixels_.data(), w_ * sizeof(uint32_t));
SDL_RenderCopy(renderer_, texture_, nullptr, nullptr);
SDL_RenderPresent(renderer_);
```