Spectra Final Project Page

Authors: Viktor Mooren, Ahsvin Verma, Curtis Hu, Toben Main

Abstract

In Spectra, we set out to model the iridescent effects seen in thin films and soap bubbles by simulating the physical behavior of light at the wavelength level. To achieve this, we extended our existing ray tracer from Homework 3 to support spectral rendering, enabling it to process and track light across a continuous range of wavelengths instead of traditional RGB values. Spectral ray tracing is more physically accurate because light in the real world is composed of a spectrum of wavelengths, not discrete red, green, and blue components. This allowed us to more easily capture wavelength-dependent interference patterns and color shifts that are characteristic of thin films. We implemented models for thin-film interference based on Fresnel equations and optical path differences, integrating them into the ray tracing pipeline. The final result is a physically informed renderer capable of producing realistic and visually rich simulations of phenomena like bubbles, oil slicks, and other interference-based effects.

Technical Approach

Part 1: Spectral Ray Tracing:

The SpectralBSDF is the core of the spectral extension. It moves beyond treating light as a single RGB triplet. The fundamental idea here is that material properties (reflectance, transmittance, refractive index) can vary with the wavelength of light. Therefore, when light interacts with a surface, the amount reflected or transmitted, and even the direction of transmission, can be different for each wavelength. So, our goal is to work with the spectral power distribution of light, instead of an RGB representation, to build a new ray tracer to accomplish the aforementioned tasks.

Since dealing with continuous spectral distributions is computationally expensive, a sampling approach is used. The sample_lambda function selects a discrete set of wavelengths to represent the incident light's spectrum. The current implementation uses a simplified method, picking a random wavelength within the visible range and then selecting a small band of nearby wavelengths. This is called "hero sampling" and is an attempt to capture some local spectral characteristics with a limited number of samples. While taking random samples of the wavelength every time would work, it requires a lot more samples to converge. Instead, we use hero sampling to decrease noise while remaining unbiased.

Because the final output of a typical display is in RGB, the sampled wavelengths need to be converted. We found data on the CIE XYZ color space to translate our wavelength into this needed conversion. Each sampled wavelength is converted to its corresponding XYZ (and then RGB, which we also found a matrix for online) value, representing how a human observer would perceive that single wavelength. By processing a set of sampled wavelengths and averaging their RGB contributions, we get an approximate RGB representation of the spectrally varying light interaction. In our new BSDF's f() function, we now introduce this converted wavelength value as our intensity of light, which is then scaled by reflectance or transmittance variables.

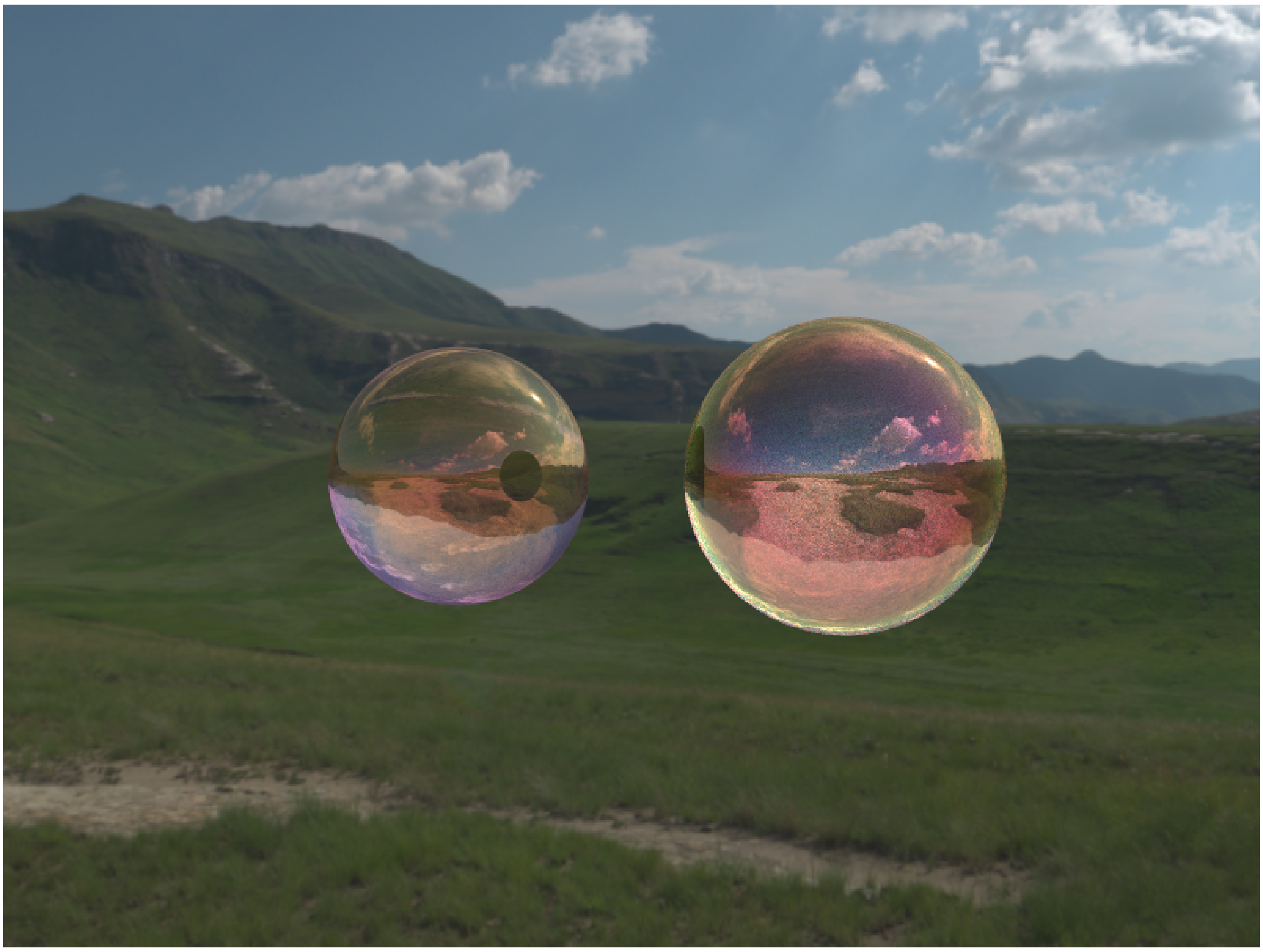
We should also note our decision to write our spectral segment as a BSDF instead of processing wavelengths throughout the rendering pipeline and doing an RGB conversion at the end. While this approach is viable and something that we saw in Dawson's paper on spectral ray tracing, we found a bit of difficulty trying to implement it. We figured we'd have to create a wavelength attribute for our rays and then find a way to propogate it throughout our system, but then we thought it would be difficult to work with these properties throughout our illuminance portions of the ray tracer. By calculating change in light at the surface of the object, we prevent having to rewrite the entire rendering pipeline to introduce spectral rendering, while also allowing us to maintain the properties of spectral ray tracing and additionally work within the realms of reflectance, transmittance, and refraction to produce thin films, bubbles, etc. Intuitively, we also liked the idea that the materials themselves carry the properties of having certain wavelengths of light associated with them, and they're also responsible for the way light interacts with them.

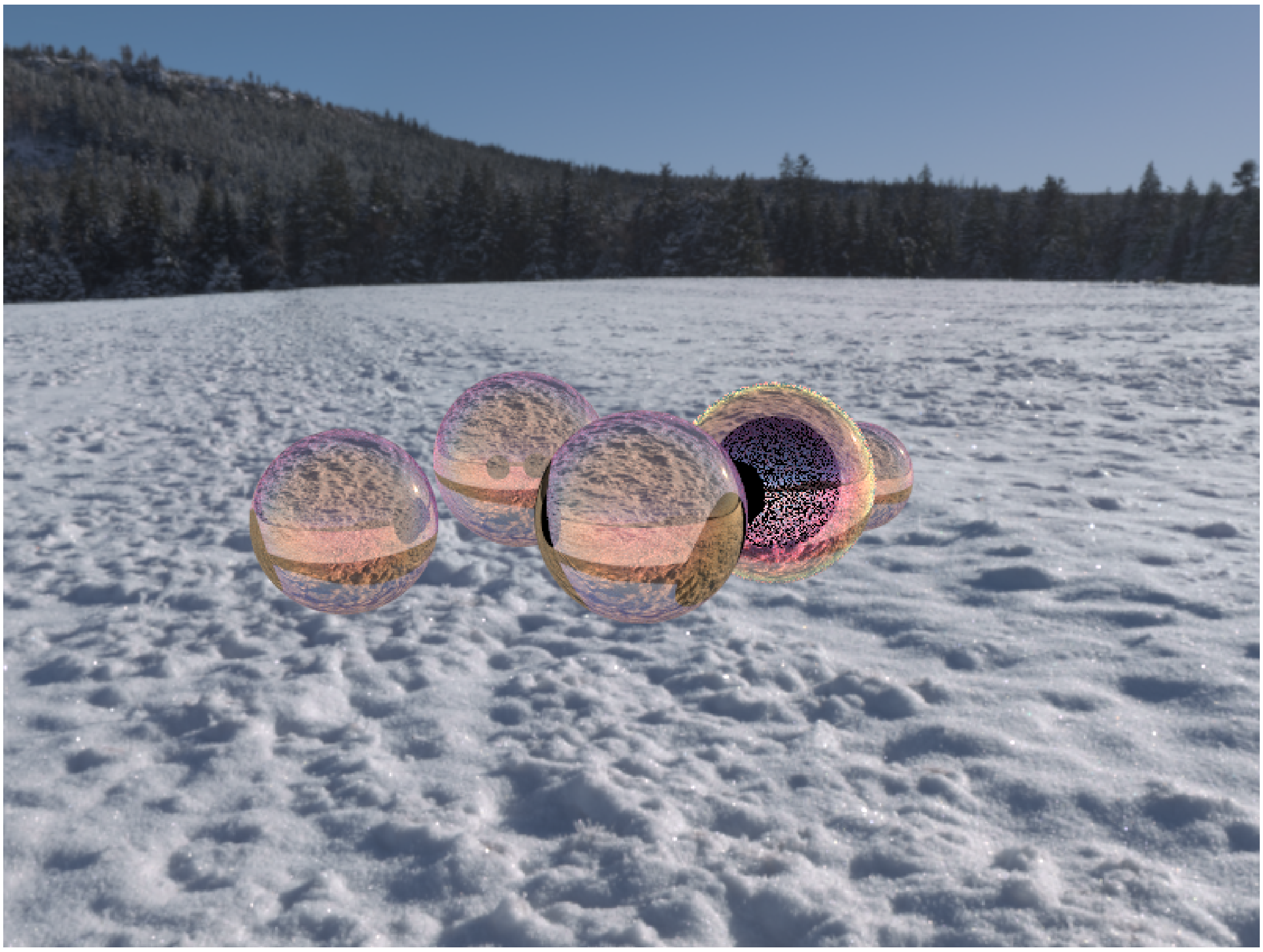
Part 2: Thin Films:

When doing this project, we learned not to invent the wheel. Initially, we considered either learning a new toolkit (PBRT) or craft a new ray tracer to create bubble/thin film rendering, but then we realized we had a good portion of the work already laid out for us in our homework 3. Utilizing resources and information other people have already tested and tried also came in handy. We were able to find various sources on XYZ and RGB conversions, as well as intricate details like the value of hero sampling, and implemented a new RNG for our hero sampling. We also used the same ray tracer as we did for our homework 3, so we were able to reuse the code for our ray tracer and our ray-object intersection functions.

There were a lot of issues, but we were able to get a working spectral implementation, albeit with higher general reflection and smaller transmission than we would've liked.

Results





Video

References

Contributions

Curtis Hu

Documentation

Viktor Mooren

Putting video together

Ashvin Verma

Thin film rendering, images

Toby Main

Spectral Primary Decomposition for Rendering with sRGB Reflectance Ian Mallett1 and Cem Yuksel

https://graphics.geometrian.com/research/spectral-primaries.html

Hero Wavelength Spectral Sampling A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, J. Hanika

https://cgg.mff.cuni.cz/publications/hero-wavelength-spectral-sampling/?utm_source=chatgpt.com

http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html