Worksheet 3

The subject of this third worksheet is primarily reflection and refraction/transmission. The fact that light is electromagnetic waves is surprisingly important in this context. The complex index of refraction provides a way of accurately capturing the appearance of metals (conductors) and transparent objects such as glass (dielectrics). Fresnel's equations for reflectance tell us how much light is reflected and how much it is refracted into the material. The imaginary part of the refractive index also tells us how much light is absorbed once inside the material. In the following, we will use these properties of the complex index of refraction to accurately render metals and transparent objects. So, the Fresnel equations and the recursive nature of reflection and transmission is what you will be working with in this set of exercises.

Learning Objectives

- Implement recursive ray tracing.
- Explain the trade-off in splitting versus Russian roulette.
- Apply splitting and Russian roulette to do efficient, unbiased rendering.
- Use the complex index of refraction and the Fresnel equations to simulate the appearance of dielectrics and conductors in their pure form.

Ray Tracing

After introducing Lambertian (perfectly diffuse) reflection in Worksheet 1, your ray tracer now needs to include the ability to render perfectly specular objects.

- Load the Cornell box (`CornellBox.obj`) and two spheres (`CornellLeftSphere.obj` and `CornellRightSphere.obj`) into your ray tracer. The materials of the two spheres are mirror and glass respectively, but they will appear black as long as shaders for mirror objects and transparent objects have not been implemented.
- Implement a shader for mirror objects. Use the number of reflections (the trace depth) to stop the recursive ray tracing. (In the `pathtrace` project of the framework, implement the `shade` function in `Mirror.cpp`.)
- Change the material of the right sphere such that both spheres are perfect mirrors. Ray trace an image and save the result.
- Change the material of the right sphere back to glass and implement a shader for transparent objects. Again, stop the recursion after a specific trace depth. (In the framework, you need to implement the `split_shade` function in `Transparent.cpp`.)
- Use the Fresnel equations for reflectance to get the correct amount of reflection and transmission at the glass surface. (In the framework, implement the first four functions in `fresnel.h` and use the `RayTracer`'s `trace_refracted` function with an extra argument R to retrieve the reflectance in you shader.) Ray trace the scene and save the result.

Russian Roulette

We can use Russian roulette to eliminate the error (bias) that eventually appears when you stop tracing rays at a certain trace depth (after a pre-determined number of light bounces). Russian roulette also prevents a combinatorial explosion in the number of rays that need to be traced. The trade-off is an increase of high-frequency noise in the image. Therefore one should always consider splitting rays at the first (or first few) bounces before switching to Russian roulette.

- Most real-world material parameters are available as a spectrum (per wavelength). To translate a spectrum into RGB, we need the CIE colour matching functions. An XML file format and some code has been developed to do this translation for you. The file format is called MPML (Material Properties Markup language) and the loader is implemented in the `load_mpml` and `Medium` files of the framework. Take a look at the file `media.mpml` which is distributed with the framework. The idea is to build a database of material properties where you can add new materials whenever you find measured or computed properties in the literature. The format is simple, but still undocumented. Try to figure out how it works. Pick a metal from `media.mpml` (feel free to add a new one) and apply it to the Stanford bunny model (`bunny.obj`). (In the framework, names of the wavefront OBJ materials will be matched to names in `media.mpml`. This means that you can load the optical properties for your materials by ensuring that these names match.)
- The imaginary part of the refractive index for metals is very large. This means that everything that refracts into a metal is almost immediately absorbed. Consequently, we can use an RGB Fresnel reflectance computation with the complex index of refraction to capture the appearance of metals. Implement a metal shader. Use Russian roulette to stop the recursion after a random number of light bounces. Use splitting for the first light bounce to reduce noise. (In the `pathtrace` project of the framework, implement the function `fresnel_R` in `fresnel.h` which computes the Fresnel reflectance for complex indices of refraction. And implement the `shade` function in `Metal.cpp`. The framework uses the metal shader for wavefront OBJ materials set to `illum 11`.)
- Render the (metal) bunny inside the Cornell box (`CornellBox.obj`).
- Load the elephant model (`justElephant.obj`). Make sure the elephant has only perfectly transparent glass material (use `illum 4` for all three wavefront OBJ materials). Render the elephant three times with maximum trace depth 1 (reflection only), 2, and 10 respectively. Save the three images. Replace the maximum trace depth by an implementation of Russian roulette and splitting using Fresnel reflectance. Your implementation works when the elephant disappears. Load the elephant together with the (metal) bunny into the Cornell box and render the scene. Save the resulting image. (In the framework, revise the `shade` and `split_shade` functions in `Transparent.cpp`.)

Worksheet 3 Deliverables

Any new materials that you might have added to `media.mpml`. Glass elephant images with maximum trace depth of 1, 2, and 10, respectively. Cornell box images (two mirror balls, one mirror ball and one glass ball, metal bunny and glass elephant). Include relevant code. Please copy everything into your lab journal.

Reading Material

The curriculum for Worksheet 3 is

- Frisvad, J. R. Electromagnetic Radiation. In *Light, Matter, and Geometry: The Cornerstones of Appearance Modelling*, Chapter 4, VDM, December 2008.

P Section 8.2. *Specular Reflection and Transmission*.

P Sections 13.7. *Russian Roulette and Splitting*.

Alternative literature available online or uploaded to CampusNet:

- Dutré, P. *Global Illumination Compendium*. Lecture Notes, Katholieke Universiteit Leuven, September 2003. http://www.cs.kuleuven.ac.be/~phil/GI/
  (See Section VII: Optics.)
- Arvo, J., and Kirk, D. Particle Transport and Image Synthesis. *Computer Graphics (Proceedings of ACM SIGGRAPH 90) 24*(4), pp. 63-66, August 1990.