Week 2

The subject for the second week is primarily reflection and refraction/transmission. The fact that light is electromagnetic waves is surprisingly important in this context. And, as you learn more about materials, you will find that it is necessary to take the wave properties of light into account to realistically capture the appearance of most materials. Fresnel's equations for reflectance are indispensable when we consider the scattering of light at a surface. So, the Fresnel equations and the recursive nature of reflection and transmission is what you will be working with in the following exercises.

Ray Tracing

After introducing Lambertian (perfectly diffuse) reflection last week, 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.

Real-Time Shading

It is hard to do exact reflection and transmission in real-time, but you can do reasonable approximations. In the following, the purpose is, again, to compute images which are as close to the ray traced versions as possible.

- Load the Cornell box (`CornellBox.obj`) and two spheres (`CornellLeftSphere.obj` and `CornellRightSphere.obj`) into your real-time shading program. The materials of the two spheres are mirror and glass respectively, but they will not look right as long as shaders for mirror objects and transparent objects have not been implemented. (Now, using the `realtime` project, the spheres will be colored by their normals.)
- Use a cube environment map to implement a shader for mirror objects. (In the framework, implement the fragment shader in `Mirror.cpp`.)
- Change the material of the right sphere such that both spheres are perfect mirrors. Render the scene using your mirror shader and the omnidirectional shadow mapping from Week 1. Save the result.

- Change the material of the right sphere back to glass and use a cube environment map to implement a single refraction shader for transparent objects. (In the framework, implement the fragment shader in `Transparent.cpp`.)
- Use the Fresnel equations for reflectance to get the correct amount of reflection and transmission at the glass surface. (Do this in the fragment shader in `Transparent.cpp`.) Render the scene and save the result.
- **(optional)**
  Improve your shader for mirror objects by doing approximate parallax correction of the reflections. There is a simple technique for doing this described in Section 6.1.1 of the lecture notes by Szirmay-Kalos *et al.* 2008 (see the reference below). This brings your real-time renderings of mirror objects closer to the ray traced reference images. (In the framework, all your fragment shaders would have to return the distance to the fragment in the alpha channel. Then you can implement the parallax correction in the fragment shader in `Mirror.cpp`.)

Week 2 Deliverables

Two Cornell box images with two mirror balls (one ray traced, one real-time shaded), two Cornell box images with one mirror ball and one glass ball (one ray traced, one real-time shaded). Discuss the differences between off-line and real-time rendering. Both differences in implementation difficulty, output quality, and generality. Include frame rates for the real-time results. Answer the following questions:

Why are the real-time images different from the ray traced images? What are the simplifying assumptions?

How come you get multiple reflections in the real-time images?

Reading Material

The curriculum for Week 2 is

P Section 9.2. *Specular Reflection and Transmission*.
P Section 13.5. *Infinite Area Lights*. (Environment mapping.)

Alternative literature available online or uploaded to CampusNet:

- Philip Dutré. *Global Illumination Compendium*. Lecture Notes, Katholieke Universiteit Leuven, September 2003. http://www.cs.kuleuven.ac.be/~phil/GI/
  (See Section VII: Optics.)

Additional resources:

- Randima Fernando and Mark J. Kilgard. Environment Mapping Techniques. In *The Cg Tutorial: The Definitive guide to Programmable Real-Time Graphics*, Chapter 7, Addison-Wesley, 2003.
  http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter07.html
- László Szirmay-Kalos, László Szécsi, and Mateu Sbert. Specular effects with rasterization. In *GPU-Based Techniques for Global Illumination Effects*, Sections 6–6.1.1, Morgan & Claypool, 2008.
  http://www.morganclaypool.com.globalproxy.cvt.dk/doi/abs/10.2200/S00107ED1V01Y200801CGR004

Background material:

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