Worksheet 1

This first exercise is primarily about getting a ray tracer up and running. We will also look into the commonly used material appearance property called the bihemispherical reflectance (ρ_d). To solve the assignments in this course, you need a ray tracer with path tracing capabilities and physically based shaders. If you have not previously implemented a ray tracer (including an intersection acceleration data structure) that you would like to use, several other rendering frameworks meet these requirements. In particular, a rendering framework has been developed for the course and has been implemented to suit the exercises. If you choose this option, every assignment can be solved by filling in a C++ code snippet based on hints inserted as comments in the framework code. Another option is to use a fully functional renderer. The framework accompanying the text book (PBRT, https://pbrt.org/) is an option. Mitsuba is another option (https://www.mitsuba-renderer.org/). If you decide to use a framework other than the one developed for the course, you can probably solve some of the exercises more easily, but you would also have to use the selected renderer rather independently.

Learning Objectives

- Get a ray tracer up and running which loads Wavefront objects and renders them with Lambertian materials using directional lights and area lights.
- Be able to change the material of an object (by modifying the OBJ and MTL files, for example).
- Render the Stanford bunny and the Cornell box with direct illumination.
- Explain edge aliasing and do jitter sampling to get anti-aliased edges.

Geometry and Materials

Throughout the course we expect you to be able to load triangle meshes defined by Wavefront .obj-files. This includes the .mtl-files which describe the materials attached to the different triangles of the meshes.

- Familiarise yourself with the Wavefront OBJ file format and especially with the Wavefront MTL file format. Make sure you know how to create a new material (in the .mtl-file) and how to change the material attached to an object (in the .obj-file). Investigate the material parameters that you can set in the MTL format. Links to file format descriptions are in the section "Reading Material" below.
- Find out how you load .obj-files into your renderer. (In the course framework, you simply provide them as command line arguments to the executables. In Visual Studio, command line arguments are set in the project properties under Debugging.)

The models that you need for the exercises are available in the `models` folder of the course framework.

Rendering Framework

Take some time to choose which rendering framework you would like to use. The renderer should support implementation of the following features:

- Loading of triangle meshes (preferably an ability to load Wavefront OBJ files).
- Path tracing (including full global illumination).
- Environment mapping (preferably including a procedural sky model).
- Shading with specular and diffuse materials.
- Shading with a rough surface (microfacet model).
- Rendering of caustics (preferably including dispersion due to spectral optical properties).
- Rendering of light scattering volumes (participating media).
- Rendering of subsurface scattering using a BSSRDF.

Ray Tracing

Once you have made your choice, do the following.

- Load the Stanford bunny (`bunny.obj`) into your renderer. Preferably, this should result in some simple pre-visualization of the bunny. If your rendering framework uses the command line as its interface for loading scene files, I recommend that you maintain a text file (`scenes.txt`) with the command line arguments used for each scene, so that you can more easily switch between scenes. In addition, take some time to learn the runtime user interface of the rendering framework.
  (CPU framework: inspect the mouse and keyboard callback functions in `RenderEngine.cpp`. GPU framework: inspect the mouse and keyboard callback functions in `render.cpp`.)
- Illuminate the bunny by a directional light source[1] emitting the radiance $L_e = \pi$ in the direction $\vec{\omega}_e = (-1, -1, -1)/\sqrt{3}$ (this is the default light in the course framework). Make sure that you shade the bunny correctly using the Lambertian BRDF.
  (CPU framework: implement the `sample` function in `Directional.cpp` and the `shade` function in `Lambertian.cpp`. GPU framework: implement `__closesthit__directional` in `shaders.cu`.)
- Compose a new material in the file `bunny.mtl` and attach it to the bunny.
- Cast shadow rays and use several samples per pixel to get a nice anti-aliased bunny. Save the resulting image (take screenshots or use the keyboard interface to save an image file).
- Now load the Cornell box (`CornellBox.obj`) and the blocks inside it (`CornellBlocks.obj`). The Cornell box has an area light source. Render this scene with direct illumination and save the result.
  (CPU framework: implement the `sample` function in `AreaLight.cpp`. GPU framework: implement `__closesthit__arealight` in `shaders.cu` and `sample_center` or `sample` in `AreaLight.h`.)

Worksheet 1 Deliverables

Bunny images (e.g. with and without modified material), Cornell box image. Include relevant code and/or render settings. Take note of number of triangles, number of samples, and rendering time. Please copy everything into your lab journal.

Reading Material

The curriculum for Worksheet 1 is

P Sections 1–1.2 and 1.7. *Photorealistic Rendering and the Ray-Tracing Algorithm*.
P Chapter 5.4–5.6. *Basic Radiometry*.
P Intro. to Chapter 8 and Section 8.3. *Reflection models* and *Lambertian reflection* as a special case.
P Sections 12–12.5 except 12.2 and 12.3.1–12.3.3. *Light Sources*.

Alternative literature available online or uploaded to DTU Inside:

- Glassner, A. S. An Overview of Ray Tracing. In *An Introduction to Ray Tracing*, Chapter 1, Morgan Kaufmann, 1989.
- Hanrahan, P. Rendering Concepts. In *Radiosity and Realistic Image Synthesis*, Chapter 2, Morgan Kaufmann, 1993.

Additional resources:

- The Wavefront OBJ file format specification: http://paulbourke.net/dataformats/obj/
- The Wavefront MTL file format specification: http://paulbourke.net/dataformats/mtl/
- The Cornell box (data and history): https://www.graphics.cornell.edu/online/box/

JERF 2021

[1] Technically, a directional light is an infinitely distant collimated light source of infinite area. Very theoretical idea, but, from a human perspective, it somehow resembles the sun.