Worksheet 9

Light that goes through one or several reflections and refractions before reaching a diffuse surface is referred to as caustics. The smaller the light source, the harder it is to find caustics by chance in unidirectional path tracing. If we have idealized perfectly specular materials and use an idealized singular source, such as a point light or a directional light, the chance of finding caustics is zero. In this worksheet, we will use photon mapping to include caustics in our renderings.

Learning Objectives

- Implement photon mapping.
- Use rejection sampling to emit photons from a point light in random directions.
- Transform irradiance to reflected radiance.
- Render caustics.

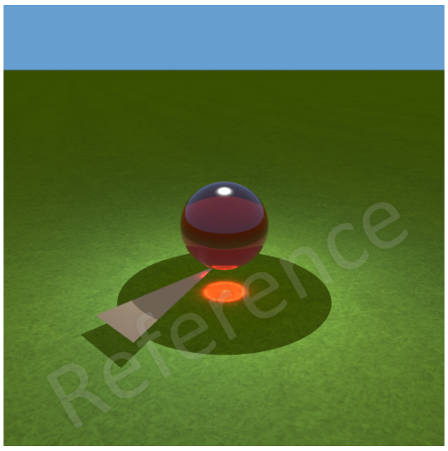
Photon Mapping

The photon mapping algorithm roughly consists of five steps: photon emission, tracing, storing, scaling, and radiance estimation. The photons are stored in a spatial data structure called a *k*d-tree. Code implementing this data structure was published by Henrik Wann Jensen in his book about photon mapping (see the reference below). This code is included in the render framework (see `PhotonMap.h`). Your job is to emit photons from a point light, trace them to the first diffuse surface, store them in a photon map, and write a shader that transforms an irradiance estimate to reflected radiance. This enables you to do a so-called direct visualisation of the photon map, which is useful for rendering caustics.

- Emit photons from a point light. Do this by implementing the function `emit` in the file `PointLight.cpp`. Use the simple von Neumann rejection sampling (the inner do-while loop) described in Figure 2 of **P** to sample the direction of the emitted photon. When setting the photon power `Phi`, do not divide by the number of photons, as this is done elsewhere in the code. Also, remember that a photon carries radiant flux (not intensity). After completing the `emit` function, go to the function `trace_particle` of the file `ParticleTracer.cpp` and call `light->emit` to shoot a particle from the sampled source.
- Trace photons to the first diffuse surface. Do this by implementing the forwarding from mirror and transparent surfaces in the function `trace_particle` of the file `ParticleTracer.cpp`. Use Russian roulette to choose either reflection or refraction when forwarding from a transparent surface (**B**: Sec. 14.5.2). Trace new rays using the functions `trace_reflected` and `trace_refracted` which were implemented in a previous set of exercises. Exit the function by calling `return` if a traced ray does not hit anything. Whenever a new surface is reached by a traced ray, you can overwrite the original ray `r` and its hit info `hit` by the new result.
- Store photons in a caustics photon map when they reach the first diffuse surface. Photons that "survive" the while-loop, which forwards from all specular surfaces, have reached their first diffuse surface and must be stored if they hit one or more specular surfaces along the way. Store a photon in the caustics photon map by calling the function `caustics.store`. Follow the convention of storing the *direction to the photon source* with the photon.
- Return to the default scene and render the stored photons as white dots in space. Run the program and press '2' on the keyboard. This chooses the shader implemented in the file `PhotonCaustics.cpp` for the ray tracing, and the preview is a render of the photons as white dots. Ensure that the photon distribution looks right and take a screenshot.
- Write a shader for diffuse surfaces that visualises the caustics photon map. Do this by implementing the function `shade` in the file `PhotonCaustics.cpp`. Remember that you get irradiance from the photon

map, but the shader must return radiance. Render the default scene with caustics only and with both caustics and other illumination. Store the results.

- Implement absorption in the photon tracing. Do this by copying your implementation of the `get-_transmittance` function to the function of the same name in the file `ParticleTracer.cpp`. Then use it in the function `trace_particle` to modify the power of a photon that has travelled through an absorbing medium (`illum` > 10). Render a coloured glass ball floating over a lawn beside a cardboard triangle as the final result.



Worksheet 9 Deliverables

Renderings of the default scene (e.g. caustics illumination only and the complete result with and without absorption) and a rendering of the photons in the caustics photon map as dots. Include relevant code snippets and render settings: number of photons in the map, number of photons in each radiance estimate, and number of samples per pixel. Please insert all this into your lab journal.

Reading Material

The curriculum for Worksheet 9 is

P Pages 11–38 of Jensen, H. W., and Christensen, N. J. A Practical Guide to Global Illumination Using Photon Maps. ACM SIGGRAPH 2000 Course Notes, Course 8, 2000.

B Section 14.5. *A Brute Force Photon Tracer*.

The SIGGRAPH course notes **P** were published in more complete form as the textbook

- Jensen, H. W. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, 2001.