Worksheet 2

Local illumination, like the result from the first worksheet, is rendered more quickly and in equal quality using rasterization techniques. However, the moment we start looking at global illumination effects (shadows, reflections, refractions, etc.), ray tracing has a number of advantages. Rasterization techniques for global illumination effects work well for some special cases only,[1] whereas ray tracing is simpler to implement and works well in general.

Learning Objectives

- Implement ray tracing.
- Render hard shadows by tracing shadow rays to a point light.
- Render reflections and refractions by tracing rays recursively.
- Compute shading of surfaces using the Phong illumination model.

Ray Tracing

Continuing the exercises of the first week, we will in the following extend our ray tracer to also support shadows and specular surfaces.

- Split the colour of each object into three components: ambient, diffuse, and specular (change the `HitInfo` struct to reflect this). Ambient is added as a constant to the diffuse part of the shader results. Diffuse (ρ_d) is the matte colour of the object used for Lambertian shading. Specular (ρ_s) is the colour and intensity of the highlights. Change the scene to use 10% of the object colour for ambient and 90% for the diffuse component. Use $\rho_s = (0.1, 0.1, 0.1)$ for the sphere, no specular for the other objects.

1. Render hard shadows by creating a shadow ray and checking for intersections between the hit point and the point light in your shader for diffuse objects. To this end, it is convenient to include the distance to the light source in the `Light` struct used for returning information from a sampled light source. If an intersection is found, only ambient should be returned and no diffuse component.

- Ensure that you have one `shade` function to be called for all materials. The `shade` function should take a pointer to a `Ray` struct and a pointer to a `HitInfo` struct as arguments and return a three-vector result (`vec3f`). Let the `HitInfo` struct contain an unsigned integer `shader` data field to be used in a `switch` statement of the `shade` function so that each object can use a specific shader.

2. Implement a shader that performs mirror reflection and render the sphere in the default scene as a mirror ball. This requires a ray tracing loop to let the ray continue along a new path if the shader sets a flag in the `HitInfo` struct asking for continuation of the path.

- Create an HTML selection menu that enables switching between different shaders for the glass material of the sphere and another one that enables switching between different shaders for the matt materials in the scene. Include a shader that simply returns the base colour of the material (diffuse plus ambient from the `HitInfo` struct).

3. Implement refraction and render the sphere in the default scene as a refractive object with index of refraction (IoR) $n = 1.5$. Refraction requires that you keep track of the relative index of refraction for each intersection with the refractive object. Use the sign of the dot product between the surface normal and the ray direction to find out whether a ray hit the surface from the inside or from the outside and store the corresponding relative index of refraction in the `HitInfo` struct.

[1] As an example, shadow mapping works well if you do not zoom in too closely, since the map has a limited resolution.

Phong Reflection

You may have noticed that the point light in the default scene does not reflect in the glass sphere. The problem is that a point has no physical extent, and, thus, the reflected rays cannot hit the light source. The Phong illumination model provides a way of imitating the reflection of light sources in the surfaces of glossy (specular and/or diffuse) objects. To include this effect, we will add Phong reflection to the ray tracer.

4. Implement the Phong illumination model in a new shader option for the sphere in the default scene. Use a specular reflectance of $\rho_s = (0.1, 0.1, 0.1)$ and a shininess (Phong exponent) of $s = 42$ for the sphere object. Your selection menu should now include the following shader options: base colour, Lambertian, Phong, mirror, and refractive.

5. Create another shader option for the sphere in the default scene called glossy. Combine the Phong illumination model with the refractive object shader in this glossy shader. Ensure that the scene description (object and material settings) is fully contained in one `intersect_scene` function. This information should all be passed to the intersection and shade functions by means of function arguments (we want the intersection and shade functions to work in general and not just for one particular object).

Reading Material

The curriculum for Worksheet 2 is (7 pages)

B Sections 4.5.3-4.5.4 and 5.2.2: *Shadows*, *Mirror Reflection*, and *Specular Reflection*.

B Section 14.3. *Smooth Dielectrics*.