Prof. Dr. F. Sadlo, M.Sc. Y. Agapov, M. Tabachnik *Heidelberg, November 12, 2019*

Exercise Sheet 4

Assignment 4.1 Ray Tracing Fundamentals [2 Points]

a) Explain the different kinds of rays: primary ray, secondary ray, shadow ray, reflection ray, and transmission ray. [*1.5 Points*]

b) Why is global illumination a recursive problem? How can a stopping criterion be derived from reflection/transmission coefficients? [*0.5 Points*]

Assignment 4.2 Analytic Geometry [4 Points]

To generate an image with ray tracing, we need so-called view rays. Assume the view ray $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$, with camera position $\mathbf{e} = (0, 0, 1)^T$, and view direction $\mathbf{d} = (1, 1, 0)^T$.

a) Calculate the intersection points between the view ray and a (opaque) sphere with radius $r = 3$ and center $M = (5, 5, 2)$. Which one of the two intersections is closer to the camera and therefore visible to the observer? [*2 Points*]

b) Calculate the intersection between the view ray and the triangle with vertices $A = (6, 0, 0)$, $B = (0, 6, 0)$, and $C = (0, 0, 6)$. First, calculate the intersection between the view ray and the plane that is spanned by A, B, and C. Afterwards, decide whether this intersection is located within the triangle. [*2 Points*]

Assignment 4.3 Ray Tracing [4 Points]

For this exercise, you should extend the code skeleton of a simple ray tracer, provided via Moodle. The ray tracer is already compiling, but only displaying a black image ("image.bmp"). Implement the intersection for rays with spheres in the methods

`Raytracer::Objects::Sphere::HitTest()` and
`Raytracer::Objects::Sphere::GetIntersection()`.

The source code located in `src/Raytracer/Objects/Sphere.cpp`. Hint: A correct solution should look like the image shown below, since this only uses ambient lighting. This framework will be extended in a later exercise.

Submission: November 19, 2019, 14:15 CEST, via Moodle