The exercises will take place in room G40 in Mühlenpfordtstrasse 23.

Throughout the course you will implement your own minimal raytracer. In each exercise you will extend your raytracer a little further. To make the task easier, you are provided with a basic raytracing framework so that you just have to fill in the missing core parts. You may use your own computer to solve the exercises, but your final program must run on the machines in the CIP pool.

Each week you must complete the assignments and hand in your commented source code for the practical tasks, as well as your solutions to the theoretical tasks (with drawings/formulas). Please use different colors in your drawings and also make sure that formulas are recognizable in your source code. Your group must present the completed assignments on each Friday, 9:45.

3.1 .obj Loader (20 Points)

Until now we have hard coded our scene descriptions in main.cpp. This is of course not practical. The OBJ file format is a simple data-format that represents 3D geometry. It consists of the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make up each polygon. To learn more about the format have a look at http://www.martinreddy.net/gfx/3d/OBJ.spec Make yourself familiar with the ObjModel class and have a look at the file cow.obj. Study how triangles are stored in the obj-format. Implement the missing parts of ObjModel::loadObj in primitive/objmodel.cpp. If your obj-importer works as expected you should see something like the image on the right:
3.2 Smooth Triangles (35 Points)

In a smooth triangle each vertex is also assigned a normal vector. When determining the normal for any point on the triangle, you interpolate between these normals using barycentric coordinates. Have a look at the revised Ray class and the Triangle::intersect(Ray * ray) function. In SmoothTriangle::getNormal(Ray const ray) use the \( u \) and \( v \) coordinates of the ray to interpolate between the vertex normals. Remember that interpolating between normalized vectors will not necessarily return a normalized vector!

If everything is implemented correctly the difference between the regular and the smooth triangles should look something like the image on the right:

![Image of smooth triangles comparison]

3.3 A simple speed up (20 Points)

You may have noticed that your raytracer has become significantly slower, due to the large number of primitives. In order to speed up computation a little, we want to introduce bounding boxes around our loaded objects. This way we can quickly check if a ray intersects the bounding box, and only then test it against all the primitives in the object. Extend the ObjModel::loadObj function to also determine the minBounds and maxBounds. Then take a look at ObjModel::intersect(Ray * ray) and revise it to only test those rays that intersect the bounding box.

3.4 Ambient Light (10 Points)

Implement a new AmbientLight class in light/ambientlight.cpp and light/ambientlight.h. Remember, this light source is always visible, regardless of the scene geometry. The illumination direction is just the inverted surface normal.

![Image of ambient light comparison]

Figure 1: Left: Without ambient light. Right: With ambient light.
3.5 Spot Lights (25 Points)

Spot lights behave much like point light sources (originating from a single point, using intensity and attenuation) except that they point into a certain direction. If a point $\vec{x}$ to be illuminated is seen from the position $\vec{p}$ of the light source under an angle $\alpha$ with the spotlights major direction $\vec{d}$, i.e.

$$\alpha = \cos \left( \frac{\vec{x} - \vec{p}}{\|\vec{x} - \vec{p}\|}, \frac{\vec{d}}{\|\vec{d}\|} \right),$$

this results in a cone of full illumination up to angle $\alpha_{\text{min}}$, no illumination above angle $\alpha_{\text{max}}$, and a falloff between those two angles. Here you can use a simple linear falloff.

Implement a new `SpotLight` class in `light/spotlight.cpp` and `light/spotlight.h`. If everything is implemented correctly, your image should look like this:

![Diagram of spot light and angles](image.png)