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.

To keep presentation time short, keep (a copy of) the original scene to generate the result shown below.

Note that some of the pictures in this exercise do not seem physically plausible. That is intentional and will become better, when we implement some more sophisticated surface shaders.

### 3.1 Quantities and Units (10 Points)

Identify the physical quantities *Power*, *Radiant intensity*, *Radiance*, *Luminous flux*, *Luminous intensity*, *Luminance*, *Illuminance* with the units lumen (lm), candela (cd), watt (W), nit, lux (lx), W/sr, W/m²/sr. Also match the radiometric quantities with their photometric counterparts and add missing counterparts, if necessary.

### 3.2 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 negative surface normal. Uncomment the ambient light source in *main.cpp*. If everything is implemented correctly, the image should look like this:

![Figure 1: Left: Lighting situation 1 without ambient light. Right: With ambient light.](image-url)
3.3 Spot Lights (30 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 \( \alpha = \arccos \left( \frac{\vec{x} - \vec{p} \cdot \vec{d} \parallel}{\| \vec{x} - \vec{p} \| \| \vec{d} \|} \right) \), 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 \texttt{SpotLight} class in \texttt{light/spotlight.cpp} and \texttt{light/spotlight.h}. Uncomment the lighting situation 2 part in \texttt{main.cpp}. If everything is implemented correctly, your image should look like this:

\[ \begin{array}{c}
\begin{array}{c}
\vec{x} \\
\vec{p}
\end{array}
\end{array} \]

\[ \begin{array}{c}
\begin{array}{c}
\alpha_{\text{max}} \\
\alpha_{\text{min}} \\
\alpha
\end{array}
\end{array} \]

\[ \begin{array}{c}
\begin{array}{c}
\vec{d} \\
\vec{x} - \vec{p}
\end{array}
\end{array} \]

Figure 2: Left: Schematic spotlight. Right: Spotlight lighting.
3.4 Goniometric Lights (50 Points)

A goniometer is used to precisely measure angles or rotate an object a specific amount of degrees. In combination with a photometer it can be used to measure the radiation characteristic of real light sources. To achieve this a photometer is placed at a certain known distance from the light source, which is affixed to a goniometer to allow it to rotate around its center. Based on the illuminance measurement of the photometer the luminous intensity of the light source for this specific polar angle can be computed. These measurements form the luminous intensity distribution (LID) and can be used for light simulations. Note that in this case the extent of the physical light source is neglected and thus only distant objects are lit correctly.

Write an importer for IES files, which contain such light measurements for real light sources. Refer to the document below for the IESNA91 specification. Have a look at light/goniolight.h and light/goniolight.cpp and implement the missing parts. Pay special attention to the comments, which point out lines that are not used in this implementation. Uncomment the lighting situation 3 in main.cpp.

If everything is implemented correctly, your image should look like this:

![Figure 3: Left: Goniometric measurement setup. Right: Scene with provided goniometric light source.](image)

3.5 Useful Links

- Goniometric measurements [https://tinyurl.com/y7b6f4yr](https://tinyurl.com/y7b6f4yr)