Tutorial on writing a Raytracer in Common Lisp

Part 2.1 ...continued

Alexander Lehmann <lehmanna@in.tum.de>

So far...

- continued with the implementation of
  - the camera,
  - the scene and
  - both the object- as well as the sphere-class,
- intersection of a ray and a sphere,
- rendered the first (and quite ugly) image.

Yet to do...

- small corrections regarding the camera ☺,
- discuss and implement
  - intersections of a ray and a cube,
  - Cramer's rule,
  - materials and also
  - the Blinn-Phong shading model.

Ray-Plane-Intersection

All points on a plane can be expressed via a linear combination of

$$\begin{aligned} & \underline{\boldsymbol{p_0}} + u \cdot \underline{\boldsymbol{p_1}} + v \cdot \underline{\boldsymbol{p_2}} \\ = \quad & \underline{\boldsymbol{p_0}} + [\underline{\boldsymbol{p_1}} \quad \underline{\boldsymbol{p_2}}] \cdot \begin{pmatrix} u \\ v \end{pmatrix}. \end{aligned}$$

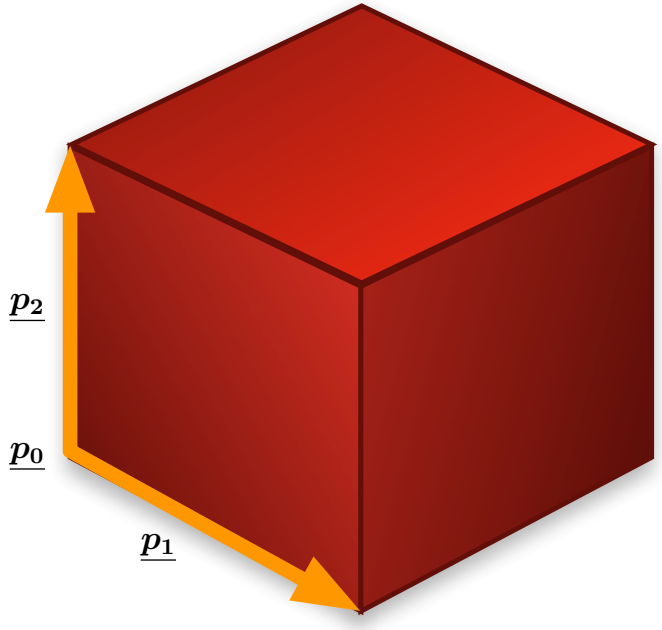
Setting the plane's equation equal to the ray's equation gives

$$\underline{\boldsymbol{p_0}} + [\underline{\boldsymbol{p_1}} \quad \underline{\boldsymbol{p_2}}] \cdot \begin{pmatrix} u \\ v \end{pmatrix} = \underline{\boldsymbol{r_o}} + t \cdot \underline{\boldsymbol{r_d}}$$

$$\Leftrightarrow [\underline{\boldsymbol{p_1}} \quad \underline{\boldsymbol{p_2}} \quad -\underline{\boldsymbol{r_d}}] \cdot \begin{pmatrix} u \\ v \\ t \end{pmatrix} = \underline{\boldsymbol{r_o}} - \underline{\boldsymbol{p_0}}.$$

Multiplying by the inverse of the matrix yields

$$\begin{pmatrix} u \\ v \\ t \end{pmatrix} = [\underline{\boldsymbol{p_1}} \quad \underline{\boldsymbol{p_2}} \quad -\underline{\boldsymbol{r_d}}]^{-1} \cdot (\underline{\boldsymbol{r_o}} - \underline{\boldsymbol{p_0}}).$$



...suitable for (planar) rectangular and triangular faces...

Cramer's Rule

According to Cramer's rule, the inverse of a given matrix A can be determined as follows:

$$A^{-1} = \frac{\text{adj}(A)}{\det(A)}$$

Let $A \in \mathbb{R}^{3x3}$ be a non-singular matrix

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}.$$

It's cofactor matrix C computes to

$$C = \begin{pmatrix} ei - fh & fg - di & dh - eg \\ ch - bi & ai - cg & bg - ah \\ bf - ce & cd - af & ae - bd \end{pmatrix},$$

hence:

$$\text{adj}(A) = C^T = \begin{pmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{pmatrix}$$

The Blinn-Phong Shading Model

According to the *Phong* reflection model:

$$\begin{aligned} I_{out} =& I_{amb} + I_{diff} + I_{spec} \\ =& k_{amb} I_{amb} + \sum_{lights} (k_{diff}(\underline{\boldsymbol{L}} \cdot \underline{\boldsymbol{N}}) I_{diff} + k_{spec}(\underline{\boldsymbol{R}} \cdot \underline{\boldsymbol{V}})^{\alpha} I_{spec}) \end{aligned} \tag{1}$$

where k_{amb}, k_{diff} and k_{spec} are empirically chosen constants subject to the condition

$$k_{amb} + k_{diff} + k_{spec} = 1,$$

and α describes the surface's *roughness*.

Because of the fairly high costs of computing $\underline{\boldsymbol{R}} \cdot \underline{\boldsymbol{V}}$, the *Blinn-Phong* reflection model instead uses the *halfway* vector

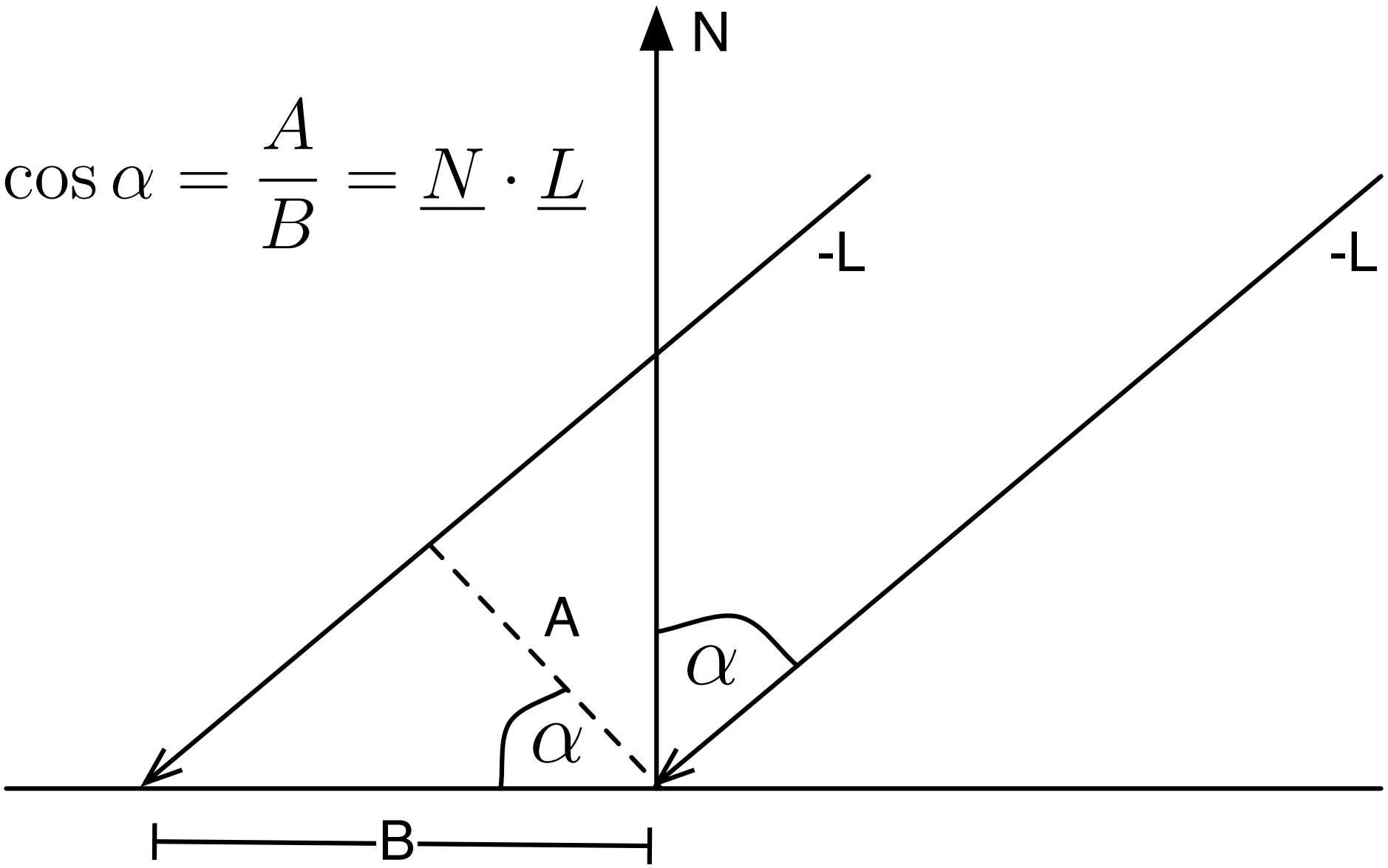
$$\underline{\boldsymbol{H}} = \frac{\underline{\boldsymbol{L}} + \underline{\boldsymbol{V}}}{|\underline{\boldsymbol{L}} + \underline{\boldsymbol{V}}|}$$

in conjunction with the surface normal $\underline{\boldsymbol{N}}$. Hence eq. (1) becomes

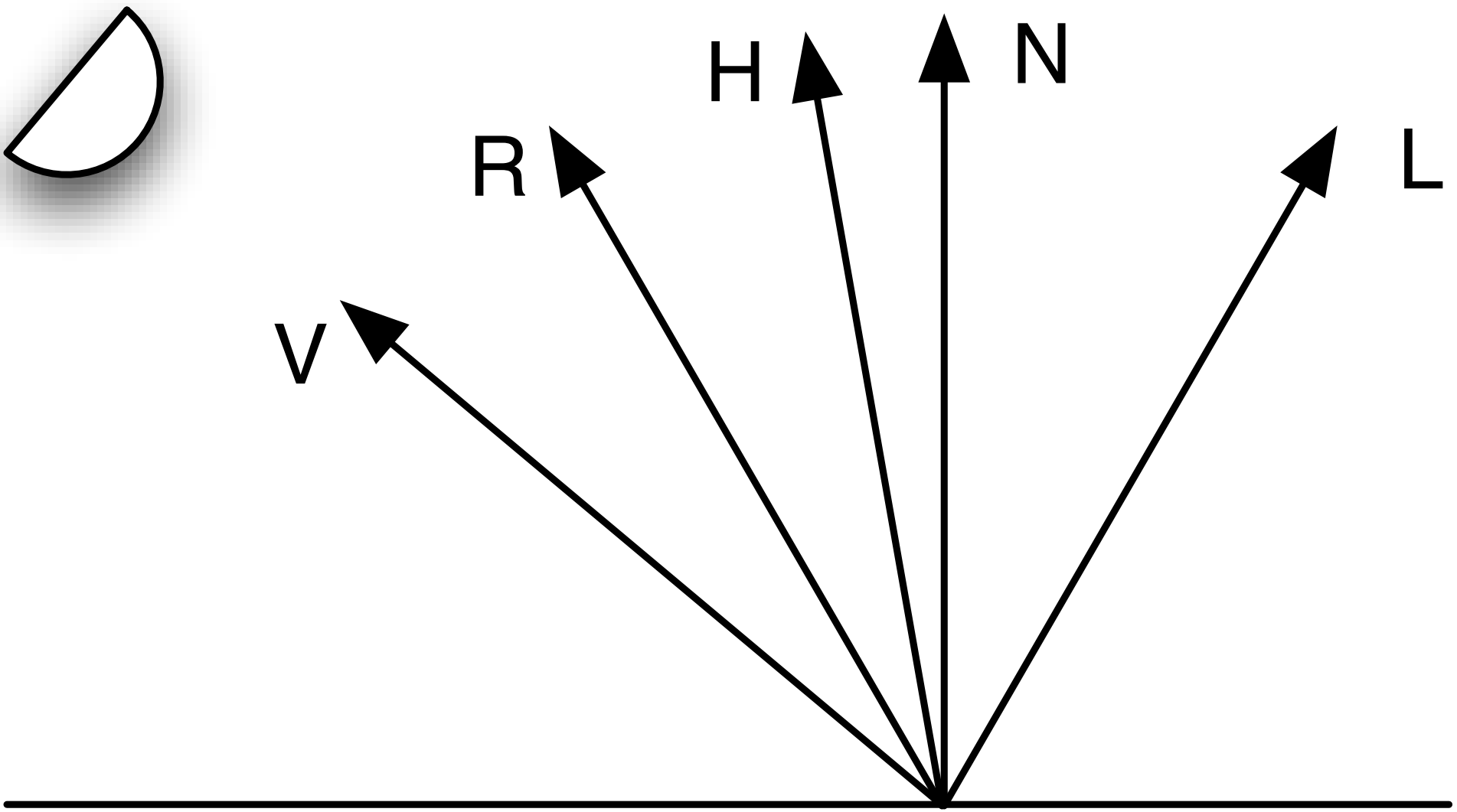
$$I_{out} = k_{amb} I_{amb} + \sum_{lights} (k_{diff}(\underline{\boldsymbol{L}} \cdot \underline{\boldsymbol{N}}) I_{diff} + k_{spec}(\underline{\boldsymbol{H}} \cdot \underline{\boldsymbol{N}})^{\alpha} I_{spec}) \,.$$

Lambertian Reflectance

$$\cos\alpha = \frac{A}{B} = \underline{N} \cdot \underline{L}$$



Phong vs. Blinn-Phong



Preview of Part 2.2

- transformations (rotation, scaling, translation),
- axis-aligned bounding-boxes,
- recursive raytracing.