§4 Rendering and visibility

4.1 Overview

What is light?

- Visible part of the electro-magnetic spectrum from $380 - 780$ nm wave length or approx. 10^{15} Hz frequency.
- Characterized by:
  - light intensity (physical quantity),
  - brightness (perceived quantity).

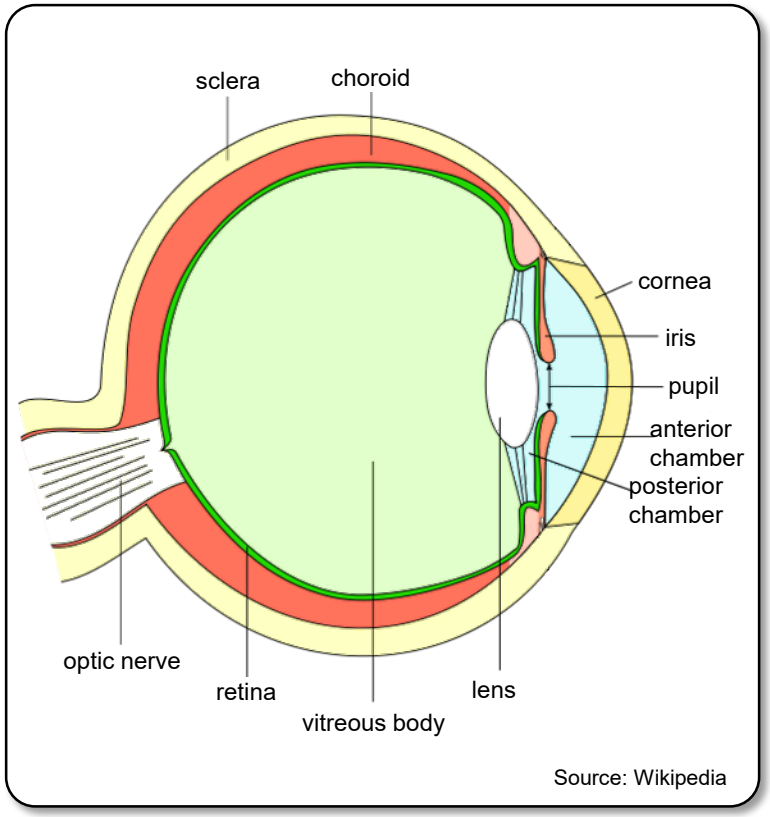
What is color?

- Perception of the spectral components of the light.
  - Which component of the light is of which wave length?

4.2 Physiology of the human visual system

4.2.1 **Perception of light** in two steps

1. Perception of the stimulus by two types of receptors on the retina:
   - Rods: Black-and-white vision, also for low intensities (ca. 120 Mio),
   - Cones: Color perception (ca. 6.5 Mio).
2. Processing of the stimulus in multiple steps:
   - Contrast enhancement at the outlet of the retina,
   - interpretation in the visual cortex of the brain.



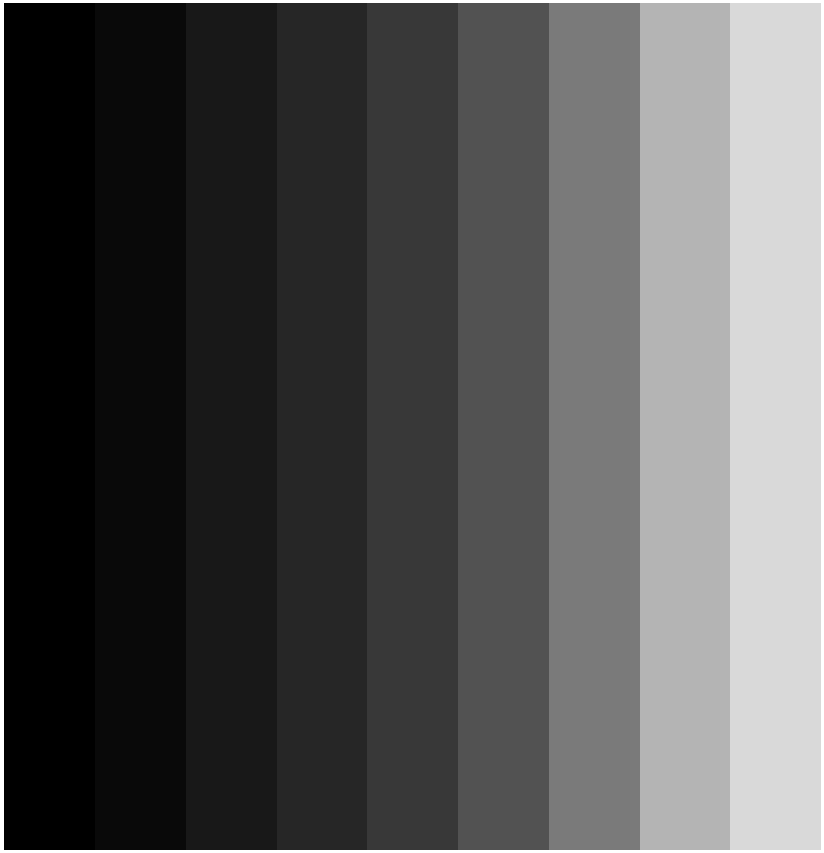
4.2 Physiology of the human visual system

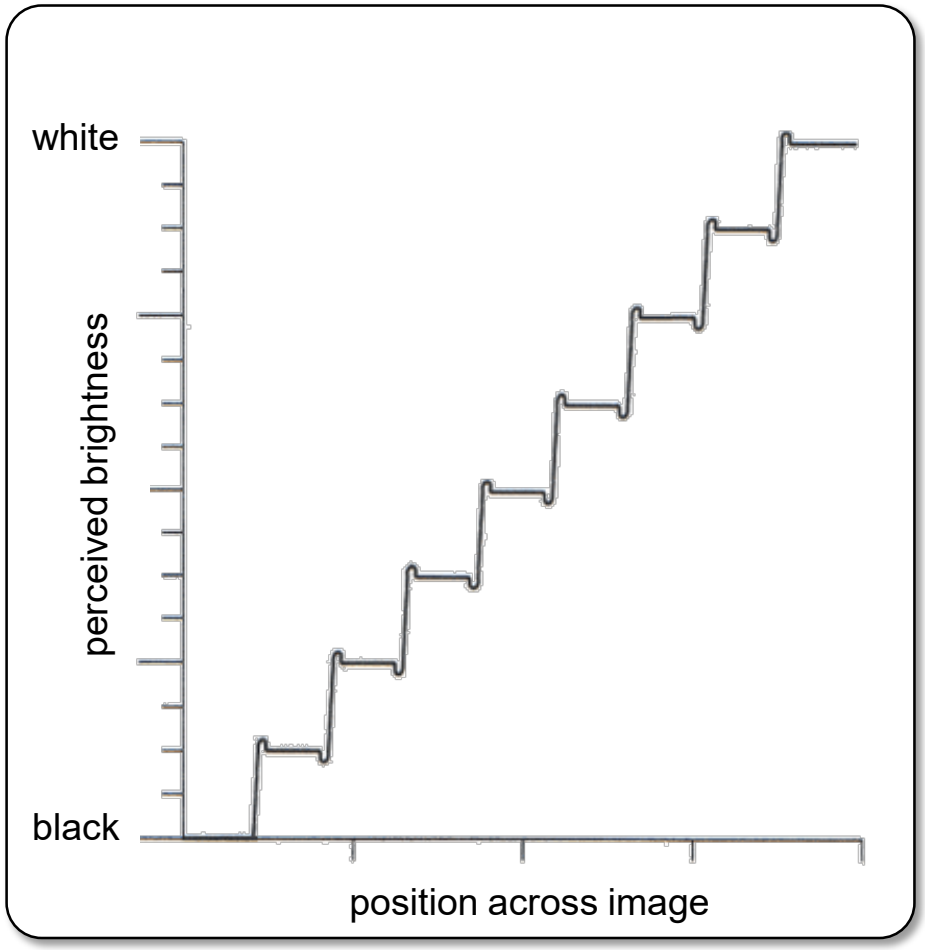
4.2.2 Mach-Band-Effect

- Interaction of light receptors in the eye amplifies „sharp“ intensity changes.
- For intensity changes in the incoming light signal into the eye, adjacent light receptors in the eye generate undershoots and overshoots in the perceived intensity to emphasis the detected intensity change.
- This automatic mechanism of edge detection at intensity changes improves the contrast and contour sensitivity of the human visual system.

4.2 Physiology of the human visual system

Mach-Band-Effect



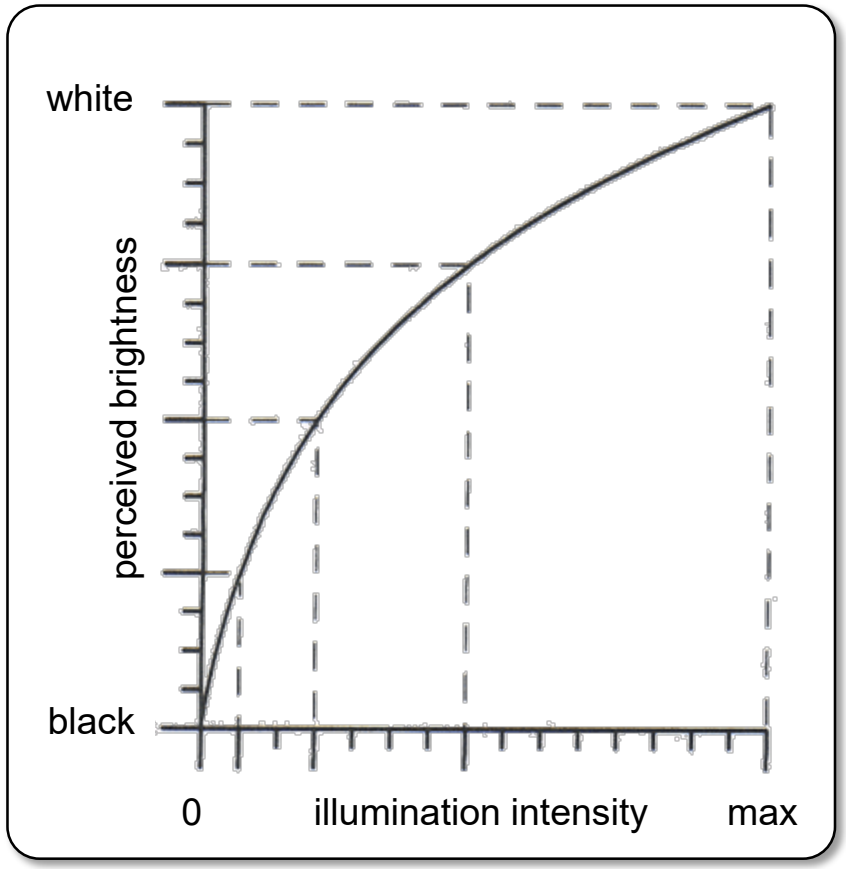


4.2 Physiology of the human visual system

4.2.2 Lechner's principle

The relation of the physical intensity of incoming light to the perceived intensity of light is non-linear, it is approximately logarithmic.

- Small light intensity differences in dark regions can be perceived better than the same light intensity difference in brighter regions.



4.2 Physiology of the human visual system

Lechner's principle

Increasing intensity in equidistant steps of 12,5% relative to the incoming intensity (from 0% to 100%)

- Intensity jump in dark regions is perceived more clearly than the same intensity jump in a lighter region.
- Perception of intensity differences depends on absolute intensity.

Increasing intensity in equidistant steps relative to the perceived intensity.

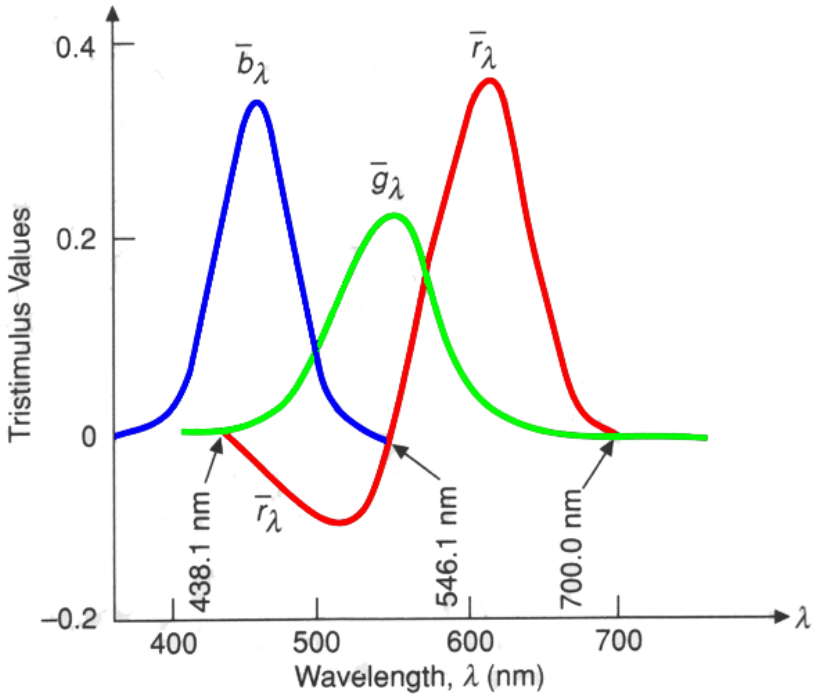
- Almost equidistant perception of intensity jumps.

4.2 Physiology of the human visual system

4.2.4 Color perception

- Cones for color perception:
  - blue-sensitive: 4% at 430 nm
  - green-sensitive: 32% at 530 nm
  - red-sensitive: 64% at 560 nm

- Every color can be generate as a (weighted) combination of three primary colors.



Color functions of the human eye to combine the complete human color spectrum.

Source: Foley et al.

4.2 Physiology of the human visual system

- **Objective color characteristic:**
  - **Dominant wave length:** The wave length in the spectrum, with the maximum radiation power.
  - **Excitation purity:** Ratio of dominant wave length to the white content, e.g. white/gray levels → 0% purity.
  - **Luminance:** Radiation energy [energy/surface area]

- **Subjective color characteristic:**
  - **Hue/tone:** Distinguishes between different color patterns and pure colors (e.g. red, yellow, green, blue, etc.)
  - **Saturation:** Distance of a color from a grey level of the same intensity, e.g. rose is less saturated than red.
  - **Lightness/brightness:** Intensity of the perceived total energy flow
    - **Lightness:** Light intensity of a reflecting object.
    - **Brightness:** Light intensity of a light emitting object.

Content

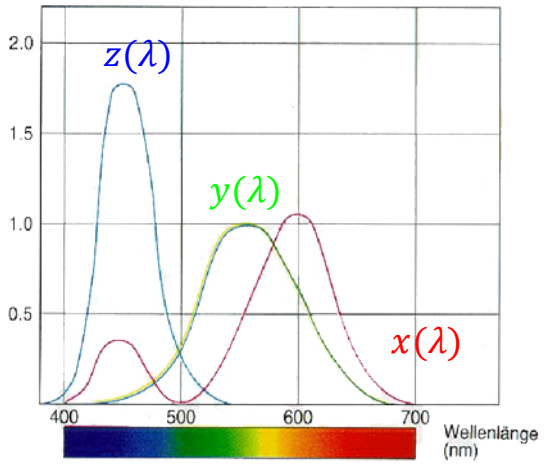
4.3 Color models

4.3.1 CIE color space

(Commission Internationale de l´Éclairage, 1931)

- International, device-independent standard for the specification of all colors perceivable by humans.
- Universal color space:
  - spectral energy $E(\lambda)$,
  - **artificial primary colors X**, **Y**, **Z** for additive mixture of colors (CIE XYZ-color space),
  - representation of a color **C** as

$$\mathbf{C} = X\mathbf{X} + Y\mathbf{Y} + Z\mathbf{Z} .$$



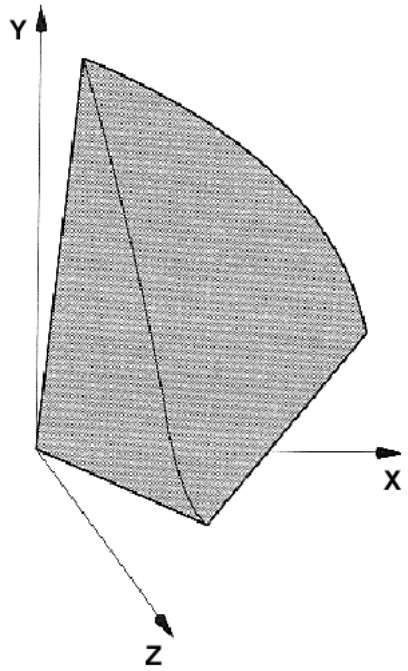
$$X = \int_{380}^{780} E(\lambda) \cdot x(\lambda)\, d\lambda$$

$$Y = \int_{380}^{780} E(\lambda) \cdot y(\lambda)\, d\lambda$$

$$Z = \int_{380}^{780} E(\lambda) \cdot z(\lambda)\, d\lambda$$

4.3 Color models

CIE color space



CIE XYZ-color gamut contains all perceivable colors.

4.3 Color models

CIE color space

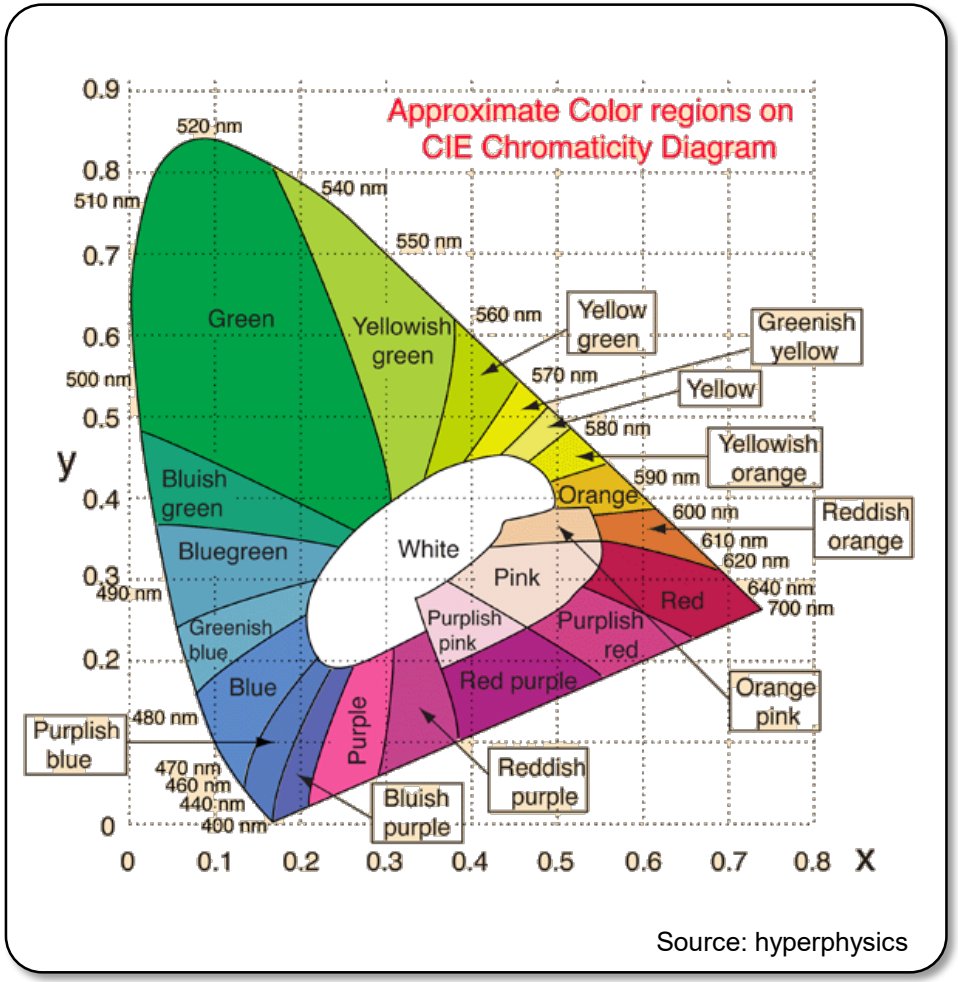
- Alternatively: Mapping of the CIE XYZ-color triple (X, Y, Z) to (x, y, Y) with

$$x = \frac{X}{X+Y+Z} \quad \text{and}$$

$$y = \frac{Y}{X+Y+Z}$$

yields CIE xyY color space.

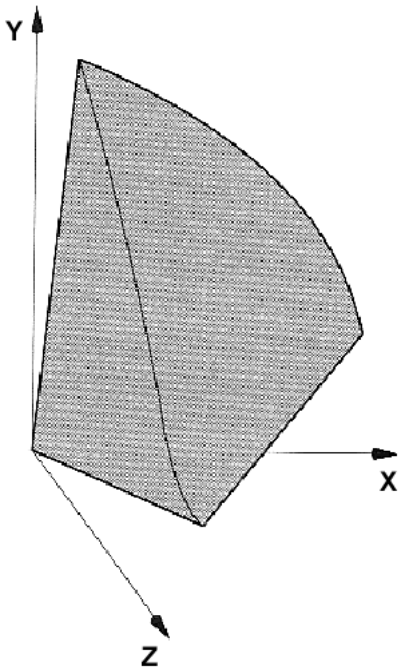
- (x, y)-diagram for all colors of the XYZ color gamut: horse-shoe-shaped CIE-diagram of chromaticity.



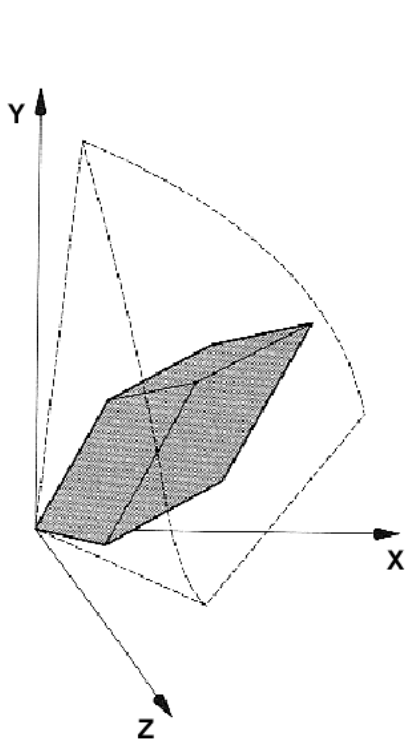
Source: hyperphysics

4.3 Color models

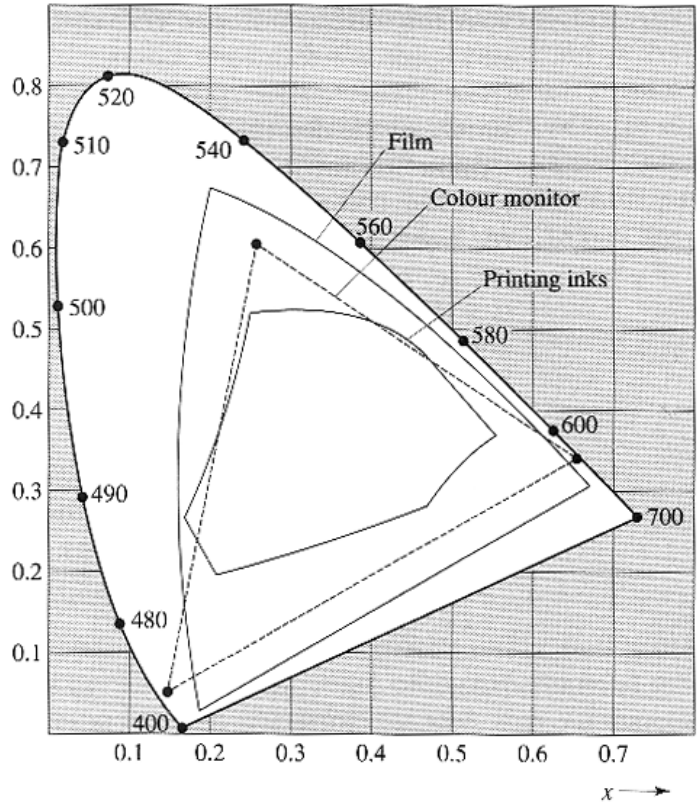
CIE-color space



CIE XYZ-color gamut
Contains all perceivable colors.



Color gamut of representable colors of a monitor.



CIE xyY-color space.

4.3 Color models

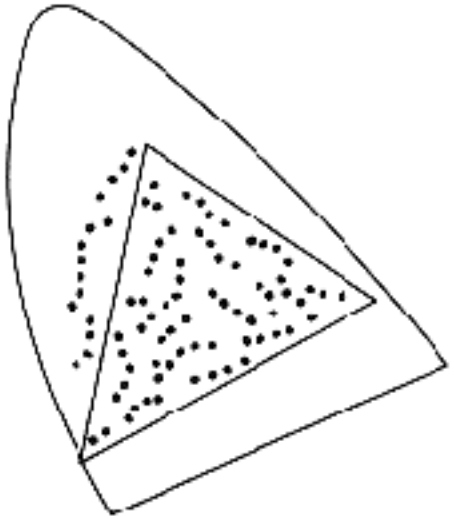
Hierarchies of known color sets

A: Set of colors **perceivable by humans**.

B: Set of colors **representable by an output device (e.g. monitor)**:
B is a sub-set of A.

C: Set of colors that **can be specified by a program**, limited by the graphics hardware (frame buffer): 24 bit/pixel → 16.777.216 colors
C is a sub-set of A but not of B.

Plane cut through 3d color space.
Each point represents a color.



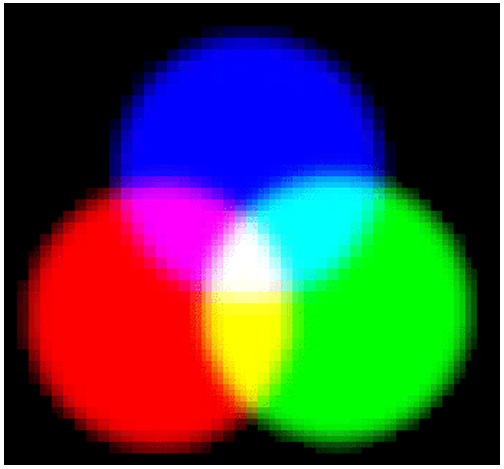
4.3 Color models

4.3.2 Classification of color models

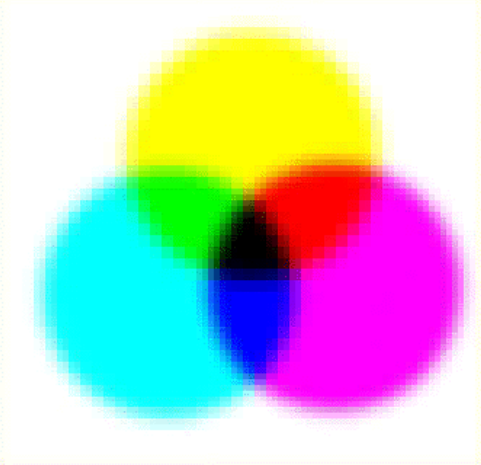
- Hardware-oriented color models:
  - Motivated by characteristics of the output device.
  - E.g.: RGB, CMY(K), YUV, YIQ, YIV, YPbPr, YCbCr, YCgCo, etc.
- User-oriented color models:
  - Equal distances in color space correspond to (approximately) equal distances in color perception,
  - E.g.: HSV, HSB, HLS, HVC.
- Transformations between these models.

4.3 Color models

- Additive color models:
  - Primary colors are weighted and added to yield a destined color.
  - Equally weighted combination of two primary colors yields the complementary color of the third primary color.
  - *„What is added to black?“*



- Subtractive color models:
  - From incoming light certain wave lengths are absorbed and the rest is reflected.
  - Combination of all three primary colors yields (theoretically) black.
  - *„What is subtracted from white?“*



4.3 Color models

4.3.3 RedGreenBlue color model

- Uses the primary colors red, green and blue for additive mixture of colors (e.g. used for screens).
- Representation of a color by a triple (r, g, b) of weights with $0 \leq r, g, b \leq 1$.

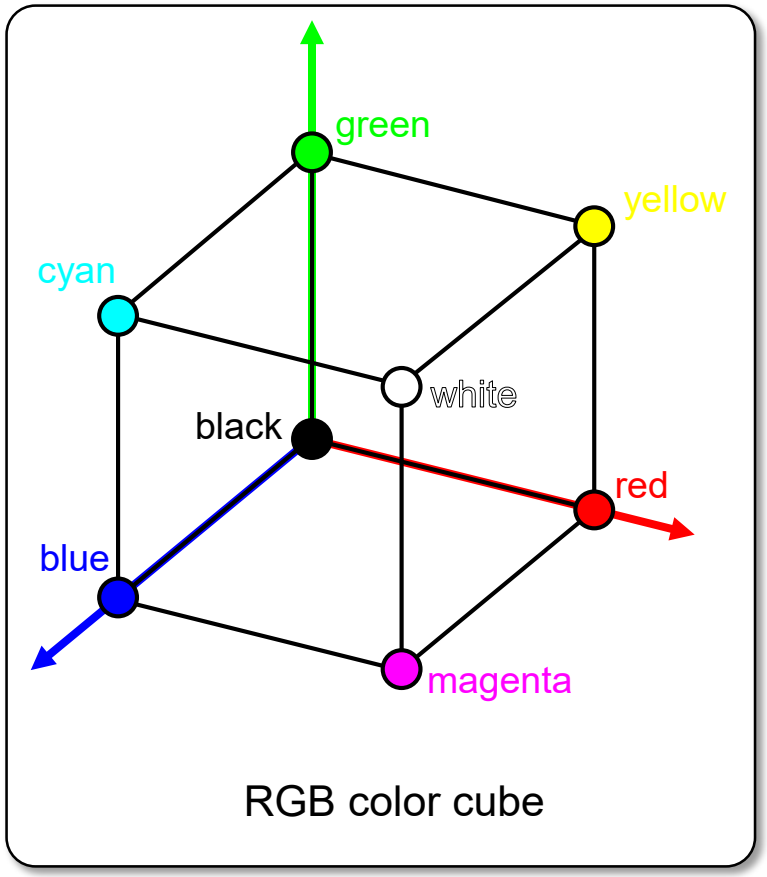
$(0,0,0) =$ Black $\quad (1,1,1) =$ White

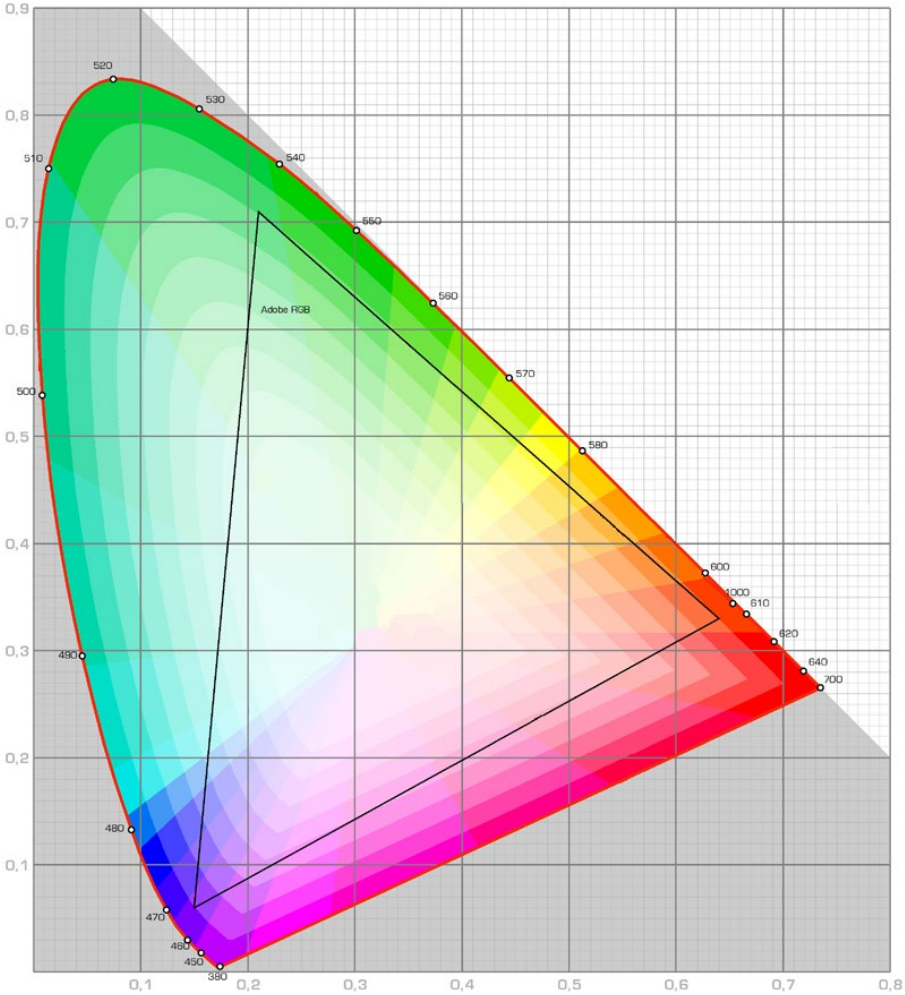
$(1,0,0) =$ Red $\quad (0,1,0) =$ Green $\quad (0,0,1) =$ Blue

$(0,1,1) =$ Cyan $\quad (1,0,1) =$ Magenta $\quad (1,1,0) =$ Yellow

- At computers e.g. 8 bit per primary color, i.e. $0 \leq r, g, b \leq 255$.
- The set of a representable colors in 3d-space is given by a cube (color cube).
  - It does not cover the complete set of perceivable colors.

4.3 Color models





4.3 Color models

- With respect to human color perception the RGB color model is non-linear:
  - At the usual color resolution of 8 bit per primary color (so-called true color), there are regions in the color gamut where adjacent points yield the same color perception for the human eye.
  - However, there are other regions where adjacent point yield a clear color difference for the human perception.
- For a user it might be very difficult
  - to mix the desired color (e.g. chestnut brown) determine the corresponding (r, g, b)-tuple or
  - to reduce the saturation of a color (requires non-uniform change of r, g and b).
- HSV color model.

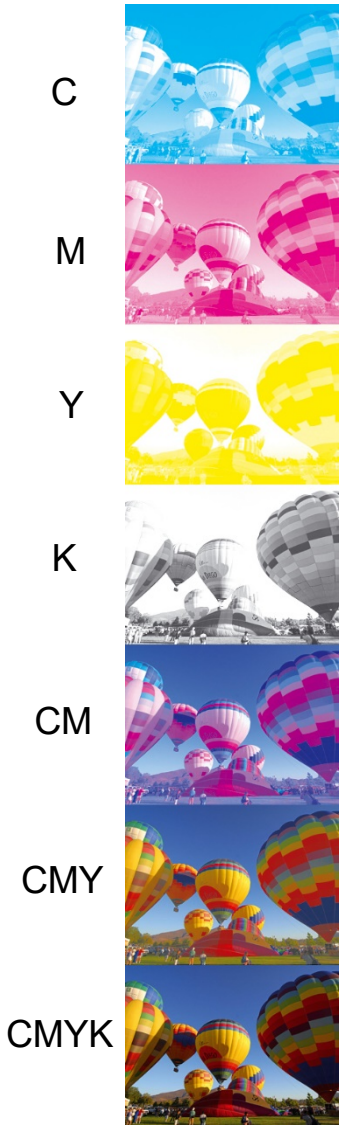
4.3 Color models

4.3.4 CyanMagentaYellow color model

- Use the primary colors cyan, magenta and yellow for subtractive mixture of colors (e.g. used for printing)
- Complementary colors to the des RGB color models, i.e.

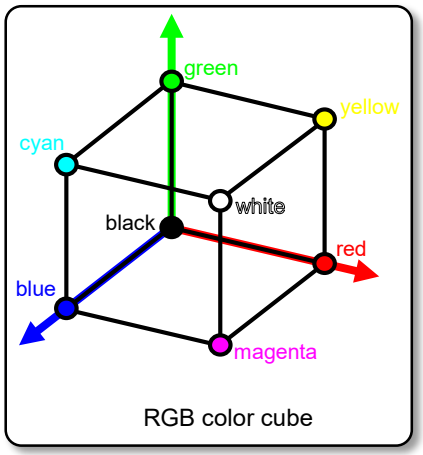
$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

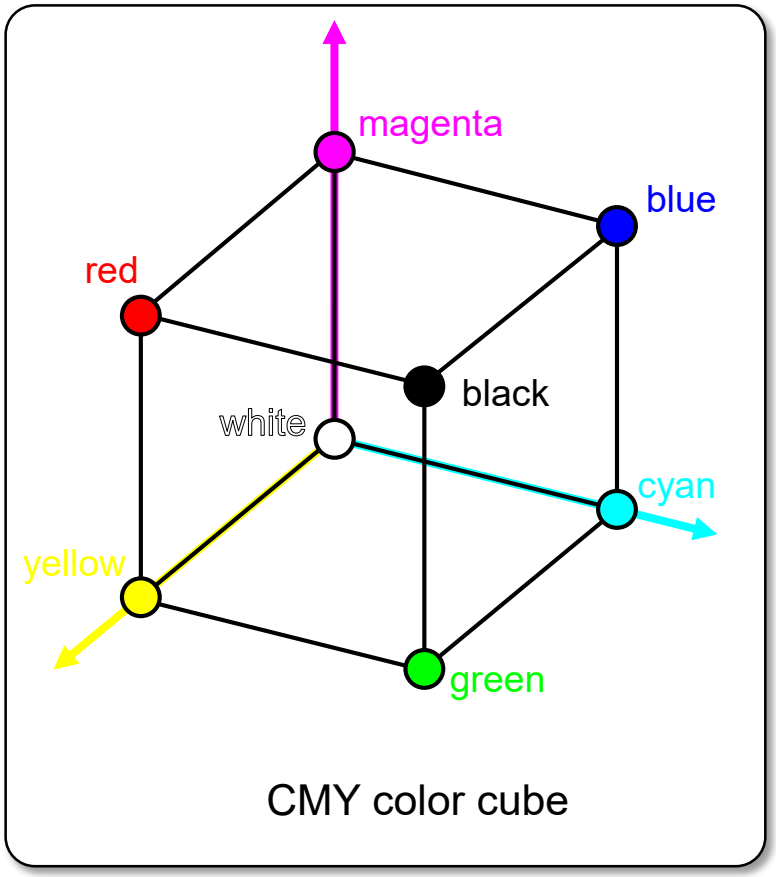
- Printers often use additional colors, e.g. black (CMYblacK color model).



Source: Wikipedia

4.3 Color models





4.3 Color models

YUV color model

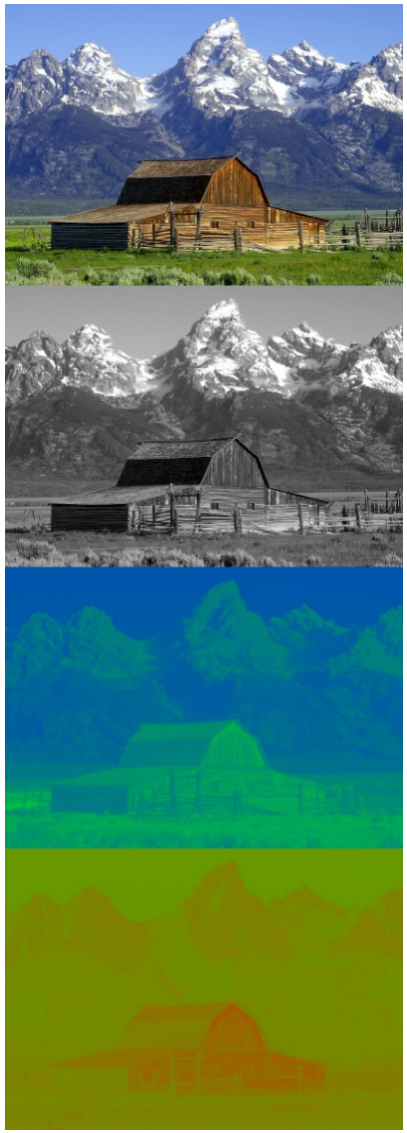
- Used for analog TV-transmissions to enhance the transmission efficiency and downward compatibility to black-white-TVs:
  - **Luminance signal** Y

    $$\mathrm{Y} = 0.299\,\mathrm{R} + 0.587\,\mathrm{G} + 0.114\,\mathrm{B}.$$

    Used in black-white-TVs.
  - **Chrominance signals** U and V (for analog PAL)

    $$\mathrm{U} = 0.493\,(\mathrm{B} - \mathrm{Y}),$$

    $$\mathrm{V} = 0.877\,(\mathrm{R} - \mathrm{Y}).$$
  - Similar: YIQ for analog NTSC,
    YPbPr for analog video transmission,
    YCbCr for digital PAL/NTSC.,
    YCgCo for mpeg/jpeg.



Source: Wikipedia

4.3 Color models

PGF-YUV color model (2)

RBG

R: ~30%

G: ~40%

B: ~30%

YUV

Y: ~90%

U: ~5%

V: ~5%

% of the total energy

Source: C. Stamm

4.3 Color models

YCbCr color model

- Used in the jpg format.
  - Luminance signal Y

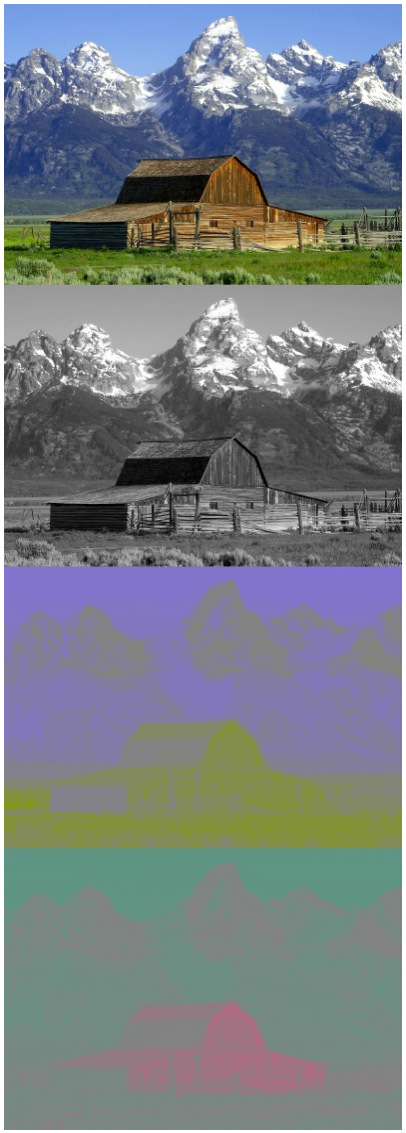
$$\mathrm{Y} = 0.299\,\mathrm{R} + 0.587\,\mathrm{G} + 0.114\,\mathrm{B}.$$

  - **Chrominance blue/red** Cb and Cr

$$\mathrm{Cb} = -0.1687\,\mathrm{R} - 0.3313\,\mathrm{G} + 0.5\,\mathrm{B},$$

$$\mathrm{Cr} = 0.5\,\mathrm{R} - 0.4187\,\mathrm{G} - 0.0813\,\mathrm{B}.$$

  - Advantages:
    - Better decorrelation of the chrominance components than for YUV and
    - smaller variance of the chrominance components.



Source: Wikipedia

4.3 Color models

YCgCo color model

- Used in the mpg-format.
  - Luminance signal Y

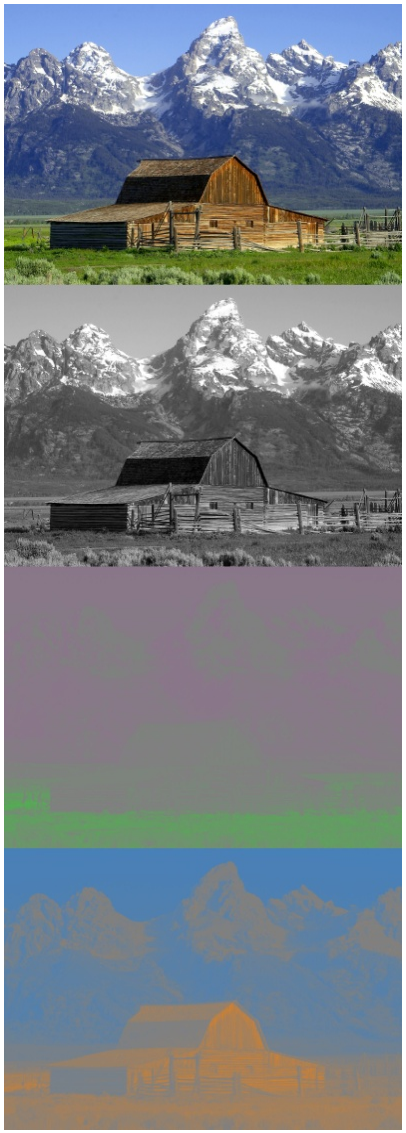
$$\mathrm{Y} = \frac{1}{4}\mathrm{R} + \frac{1}{2}\mathrm{G} + \frac{1}{4}\mathrm{B}.$$

  - **Chrominance green/orange** Cg and Co

$$\mathrm{Cg} = -\frac{1}{4}\mathrm{R} + \frac{1}{2}\mathrm{G} - \frac{1}{4}\mathrm{B},$$

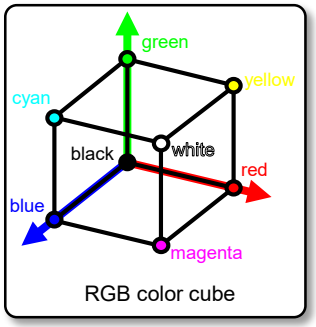
$$\mathrm{Co} = \frac{1}{2}\mathrm{R} - \frac{1}{2}\mathrm{B}.$$

  - Advantages:
    - Simpler color space transformations and
    - better decorrelation of the chrominance components than for any other Yxx color model.



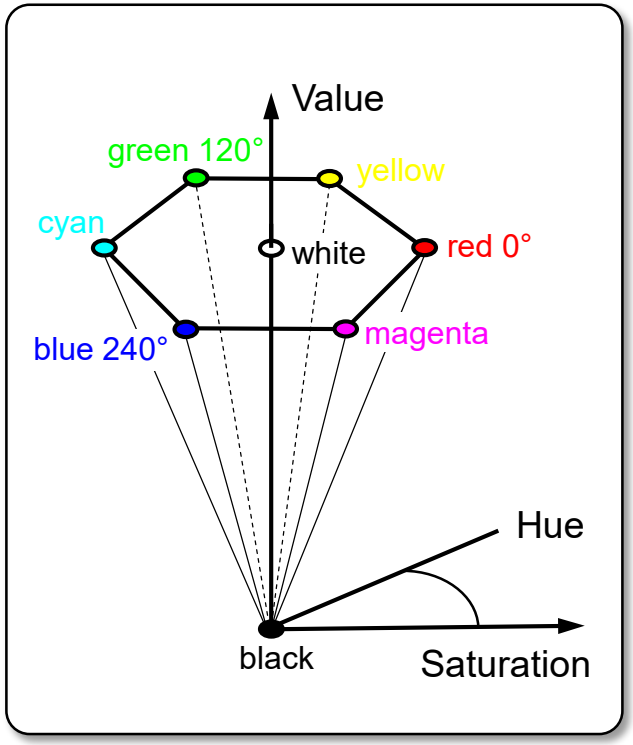
Source: Wikipedia

4.3 Color models



4.3.5 HSV color model

- Developed to support a more intuitive color mixture (perception oriented color model).
- The color “cube” in 3d color space is a six-sided pyramid.
- Uses polar coordinates:
  - **Hue:** color (color family) given as color angle in degrees: $0° \leq H \leq 360°$.
  - **Saturation:** (reduction adds white) $0 \leq S \leq 1$.
  - **Value:** brightness (reduction adds black) $0 \leq V \leq 1$.



4.3 Color models

Connection between HSV and RGB color model

- The hexagonal base of the HSV pyramid arises from the RGB color gamut by projection along the diagonal d from white to black onto a plane perpendicular to d.

- This yields the following correspondences:

RGB	Color	HSV
(1, 0, 0)	**Red**	(0, 1, 1)
(1, 1, 0)	**Yellow**	(60, 1, 1)
(0, 1, 0)	**Green**	(120, 1, 1)
(0, 1, 1)	**Cyan**	(180, 1, 1)
(0, 0, 1)	**Blue**	(240, 1, 1)
(1, 0, 1)	**Magenta**	(300, 1, 1)

- **Remark:** In the HSV model complementary colors have an angle difference of 180° in hue.

4.3 Color models

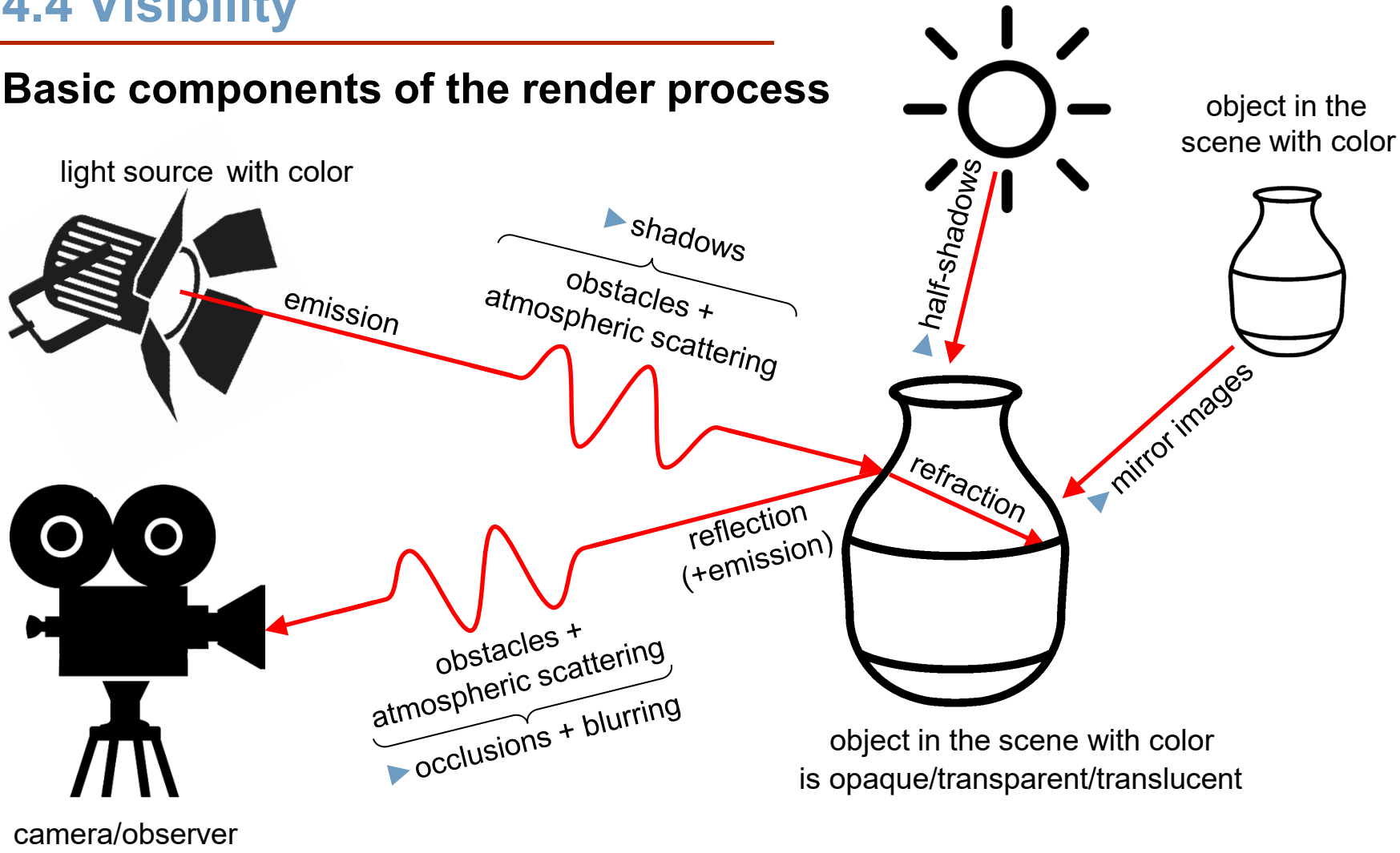
Connection between HSV and RGB color model (cont.)

- Every point P on the diagonal d from white to black in the RGB color cube, defines a sub-cube:
  - the diagonals of both cubes coincide,
  - one corner of the sub-cube is at black = $(0, 0, 0)$,
  - the opposite corner is at P,
  - the sub-cube is completely contained in the RGB color cube.
- Every sub-cube corresponds to a hexagon, which corresponds to an intersection through the HSV pyramid with $V = \text{const}$.
- The diagonal d of the RGB color cube corresponds to the V-axis of the HSV pyramid.

Content

4.4 Visibility

Basic components of the render process



Content

4.4 Visibility

Goal of visibility test

Determine all visible and invisible (occluded) parts of a scene

- from a given camera position
- as exact as possible,
- as fast as possible …

… to reduce scene complexity for subsequent steps of the rendering pipeline.

Desirable properties: high interaction rates

- Input of the user causes immediate change in the rendered image.
- Ideally in real time…

4.4 Visibility

- Occlusions occur for the projection of a 3d scene to the 2d image plane, if different 3d scene points are mapped the same 2d image point.
- Object points are visible, if they are closer to the eye of the observer than any other object point.
- Rendering requires
  - the (x, y)-coordinates in the image plane and also
  - depth information in the scene (z-coordinates).

4.4 Visibility

Coherence: Using local similarities

- **Object coherence:**

  If two objects do not intersect, their faces do not intersect.

- **Face coherence :**

  Properties of adjacent points on a face differ only slightly.

- **Depth coherence :**

  The depth $z(x, y)$ on a face can be computed incrementally.

4.4 Visibility

Classification of methods:

1. Object space methods:

- Occlusions are computed in 3d-object space,
- in principle independent on output device,
- accuracy of computation is machine accuracy.
- Example: back-face culling

```
for (all objects of the scene)
{
   1. Determine all parts of the object, that are not
      occluded by parts of the same object or parts of
      other objects.
   2. Draw these parts of the object.
}
```

4.4 Visibility

2. Image space methods:

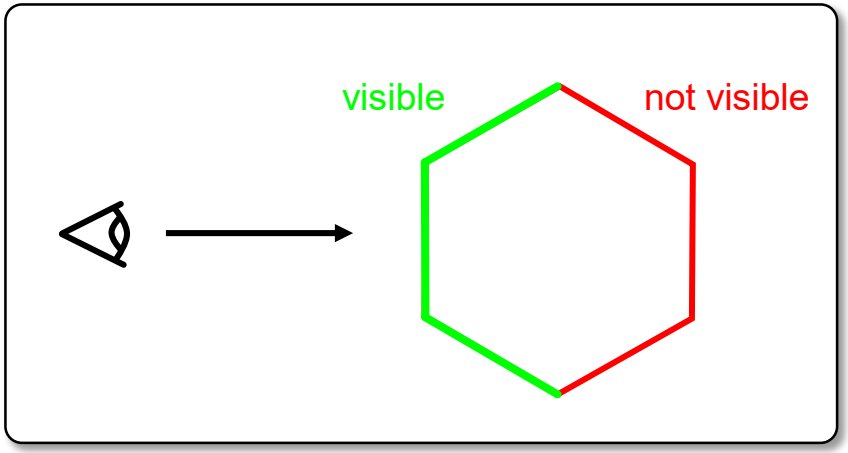
- Projection to image space and subsequent computation of intersections of the projected objects in 2d-image space,
- dependent on the output device,
- accuracy of computation is determined by the resolution of the output device.
- Example: z-buffer, α-buffer, ray-casting

```
for (all pixel of the image)
{
   1. Determine the object, whose intersections with
      the projection ray through the respective pixel
      is closest to the observer.
   2. Draw this pixel in the color of this object.
}
```

4.4 Visibility

4.4.1 Back-Face-Culling

- Removal of occluded faces can be computational expensive.
- Simple test, to reduce the complexity of the scene, before more complicated methods are applied.
- **Back-Face-Culling:**
  - Depending on the position of the camera all **back faces** of opaque objects are removed, since these will not be visible from the given perspective.

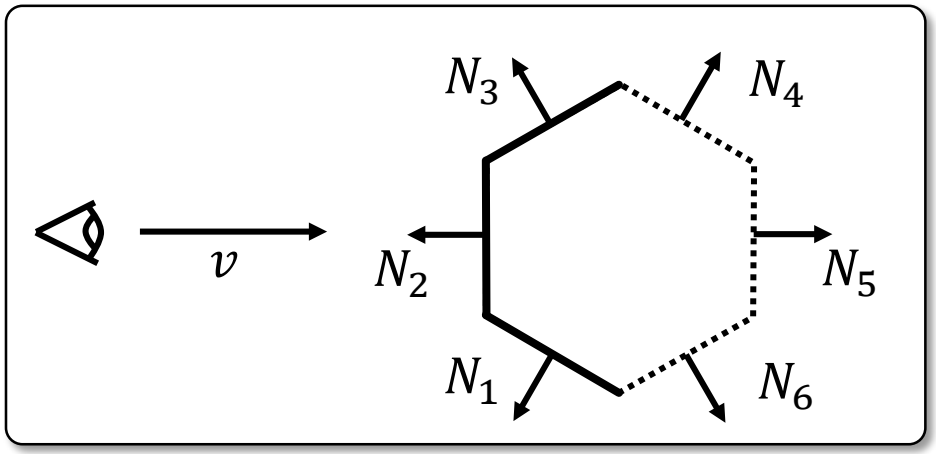


  - For polygonal objects, this reduces on average 50% of the scene complexity.

4.4 Visibility

Determination of back sides for back-face-culling

- Compute all face normals N_i in camera coordinates.
- For a back face the normal N_i has a component in view direction v.
- For the scalar product of view direction v and normal N_i this yields $v \cdot N_i > 0$.



visible	not visible
$v \cdot N_1 < 0,$	$v \cdot N_4 > 0,$
$v \cdot N_2 < 0,$	$v \cdot N_5 > 0,$
$v \cdot N_3 < 0.$	$v \cdot N_6 > 0.$

4.4 Visibility

Properties of back-face-culling

(1) Object space methods.

(2) The number of polygons in the rendering pipeline is on average reduced by a factor of two by the removal of back faces.

(3) The extra costs for the computation of the scalar products are small.

(4) If the scene contains only one convex polyhedron, back-face-culling solves the visibility problem.

- For concave polyhedrons or scenes with many objects, there might be self-occlusions or occlusions from different objects.
- Here more complex methods are required.

4.4 Visibility

Edwin Catmull. source: Wikipedia

4.4.3 Z-buffer-algorithm [Catmull, 1975]

- Determine visibility for image pixels,
- uses in image space display coordinates,
- applicable for image display on raster displays.

Principle:

- **Functional:** For each pixel in image space determine the polygon that
  a. overlaps the pixel and
  b. whose z-coordinate in camera coordinates is smallest, i.e. it is closest to the camera.
- **Realization:** Additional memory, the so-called **z-buffer**, that stores for each pixel the smallest z-value.

4.4 Visibility

Z-buffer-algorithm:

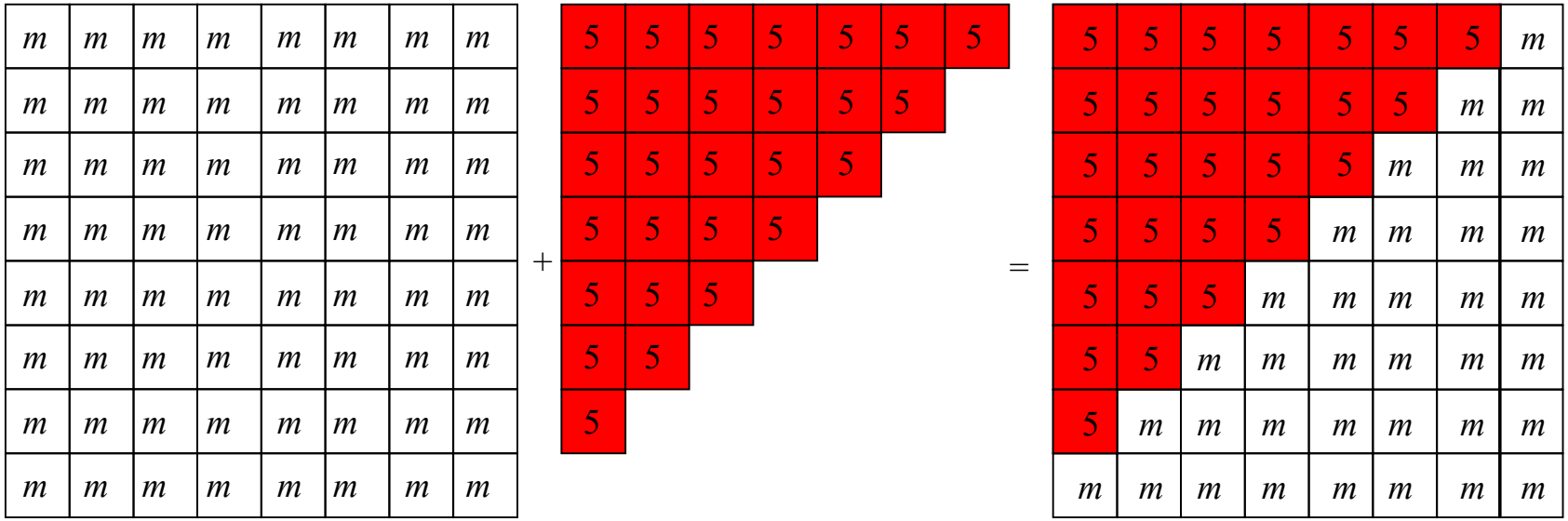
1. Initialize frame buffer with background color.
2. Initialize z-buffer with maximal z-value (back-clipping plane).
3. Scan-Conversion of all polygons in arbitrary order:
   a) Compute z-value $z(x, y)$ for each pixel (x, y) in polygon.
   b) If $z(x, y)$ is smaller than the value stored in the z-buffer at (x, y),
      i. store polygon attributes (color) in the frame buffer at (x, y) and
      ii. store in the z-buffer at (x, y) the z-value $z(x, y)$.

- At termination of the algorithm, the frame buffer holds the final image and the z-buffer its depth distribution.

4.4 Visibility

Example

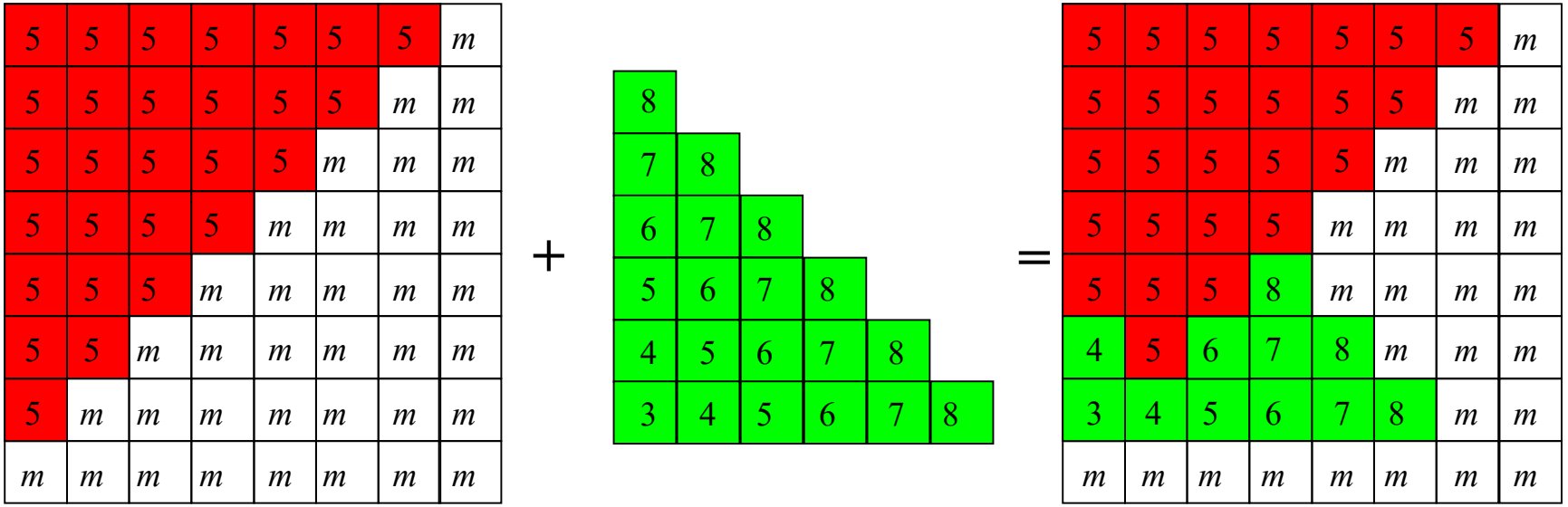
1. z-values coded by integers: smaller number ➡ closer to camera
2. Initialize z-buffer with maximal z-value m.
3. Add a first polygon with constant z-value.



4.4 Visibility

Example

4. Add a second skew polygon, that intersects the first polygon.



4.4 Visibility

Ad 3a): Computation of z for polygons

- For the computation of $z(x, y)$ for **planar polygons** (e.g. triangles) along a scan-line:
  - Plane equation: $a \cdot x + b \cdot y + c \cdot z + d = 0$
    - $z(x, y) = (-d - ax - by) / c.$
  - Use depth coherence:
    - $z(x + d_x, y) = z(x, y) - d_x \cdot a/c.$
    - This requires only one subtraction, because
      - a/c is constant and
      - $d_x = 1.$

4.4 Visibility

Properties of the z-buffer-algorithm – advantages

(1) Very simple to implement.

(2) Independent on the representation of the objects.

(3) Every point of the object surfaces must allow the computation of its z-value.

(4) No limit on the complexity of the scene.

(5) No special order or sorting required.

4.4 Visibility

Properties of the z-buffer-algorithm – disadvantages

(6) Resolution of the z-buffers limits discretization of image depth, e.g. using 20 bit exactly 2^{20} depth values are defined.

- Problematic for far away objects with small details.

(7) Extra memory is required.

Example: $1600 \times 1400 \times 16\text{Bit} = 4{,}2$ MBytes

- Subdivision into sub-images or stripes possible.
- Scan-line z-buffer-algorithm

(8) For transparent objects (α-buffering) and anti-aliasing, complex modifications are required.

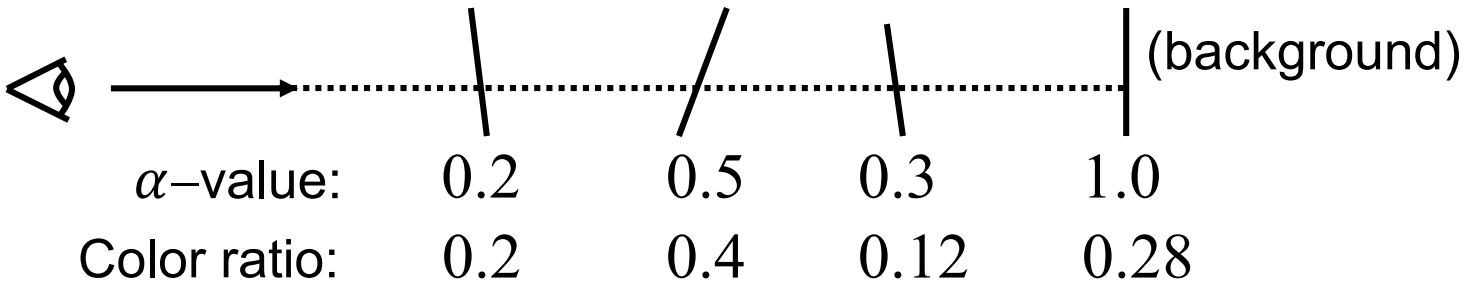
4.4 Visibility

4.4.4 α-buffer-algorithm

- Transparent surfaces have besides their color attributes an additional local value for their opacity:

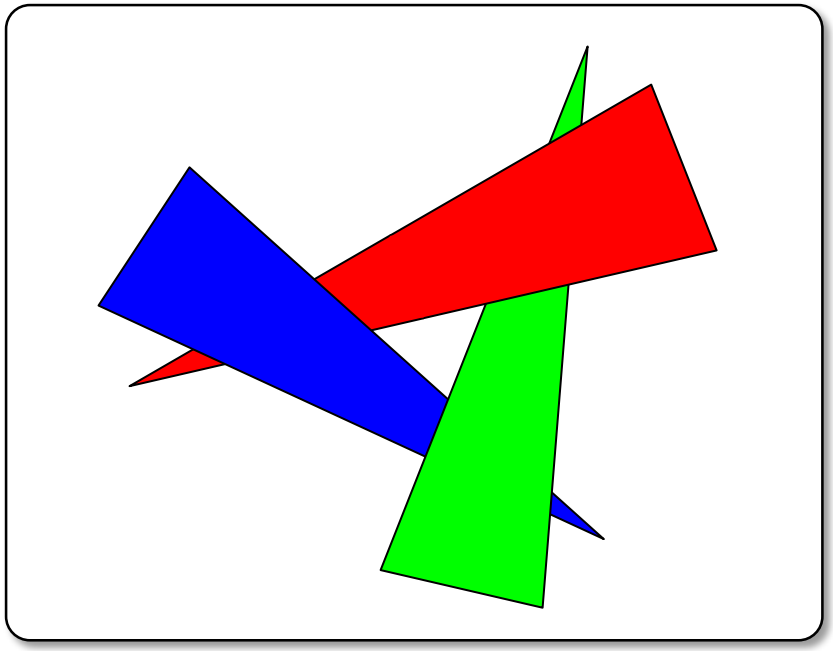
$$\alpha \in [0,1],\ 0 = \text{transparent},\ 1 = \text{opaque}.$$

- The color of a pixels consists of an α percentage of the polygon color and an $(1-\alpha)$ percentage of the color objects behind.



4.4 Visibility

- α-buffering works analog to z-buffering, with the difference that the scene is constructed from back to front.
- A sorting of the polygons is necessary, which might be difficult…



4.4 Visibility

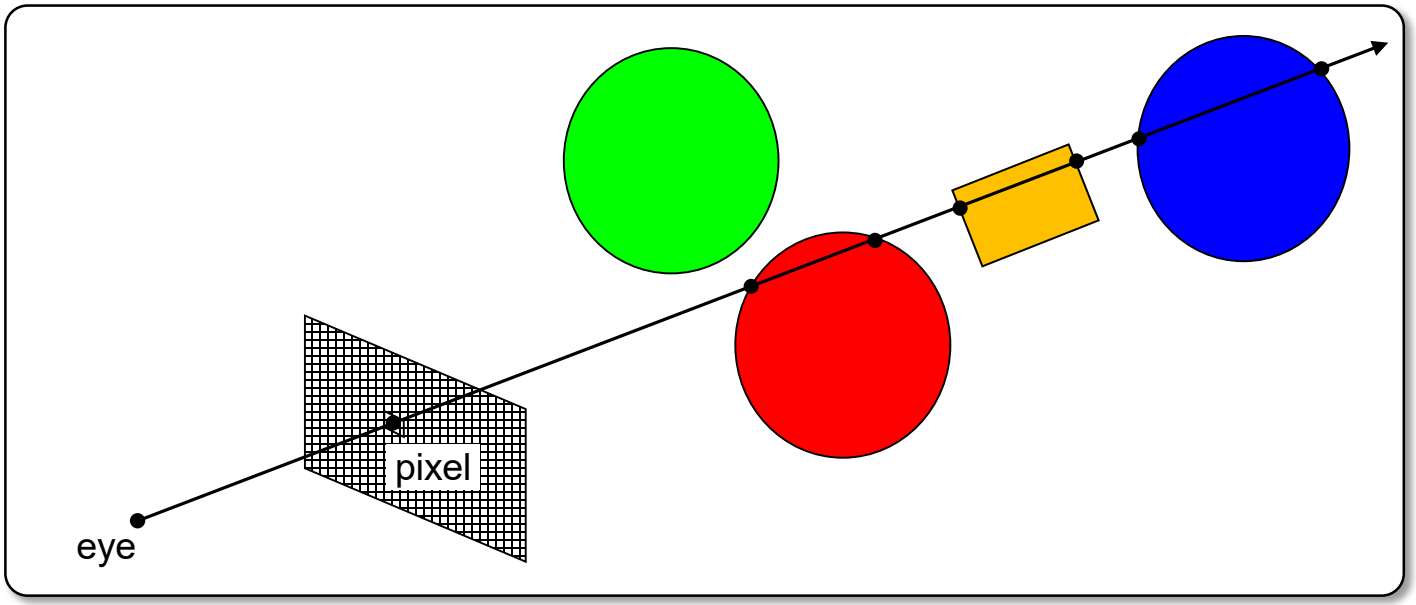
4.4.5 Ray casting

- Ray casting: Solves the visibility problem.
- Ray tracing:
  - Ray casting and
  - subsequent tracing of reflected and refracted rays.
  - ▶ Global illumination model (see §4.5.3)
- Ray tracing is often used synonymously fort both.
- Image space method, because it is applied for each pixel.

4.4 Visibility

Ray Casting

1. Cast a ray from the eye point through all pixels of the image plane into the scene.
2. Compute intersections with all objects of the scene.
3. The object with the intersection closest to the eye point is visible in the respective pixel.



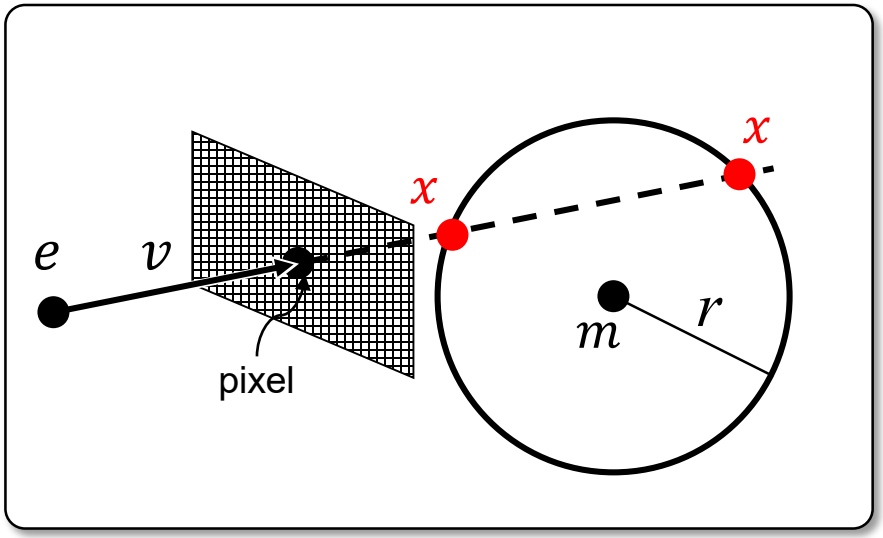
4.4 Visibility

Ad 2.:

Intersection with a ray

$$s(t) = e + t\,v$$

e eye point
v view direction: v = pixel $-e$
t ray parameter



Example 1: Intersection with an implicit sphere

$$\|x - m\|^2 - r^2 = 0$$

Substitution of the ray $s(t)$ into the sphere equation yields for x:

4.4 Visibility

- $\|e + t\,v - m\|^2 - r^2 = 0$
- $(e + t\,v - m)\cdot(e + t\,v - m) - r^2 = 0$
- $\big((t\,v) + (e - m)\big)\cdot\big((t\,v) + (e - m)\big) - r^2 = 0$
- $q(t) := t^2\,v\cdot v + 2\,t\,v\cdot(e - m) + (e - m)\cdot(e - m) - r^2 = 0$

- Solving this quadratic equation $q(t)$ for t yields at most two parameters of the intersection points s_1 and s_2:

$$s_{1,2} = r\big(t_{1,2}\big) = e + t_{1,2}v.$$

- The intersection with smallest $t > 0$ is closest to the eye point.

4.4 Visibility

Example 2: Intersection with a plane

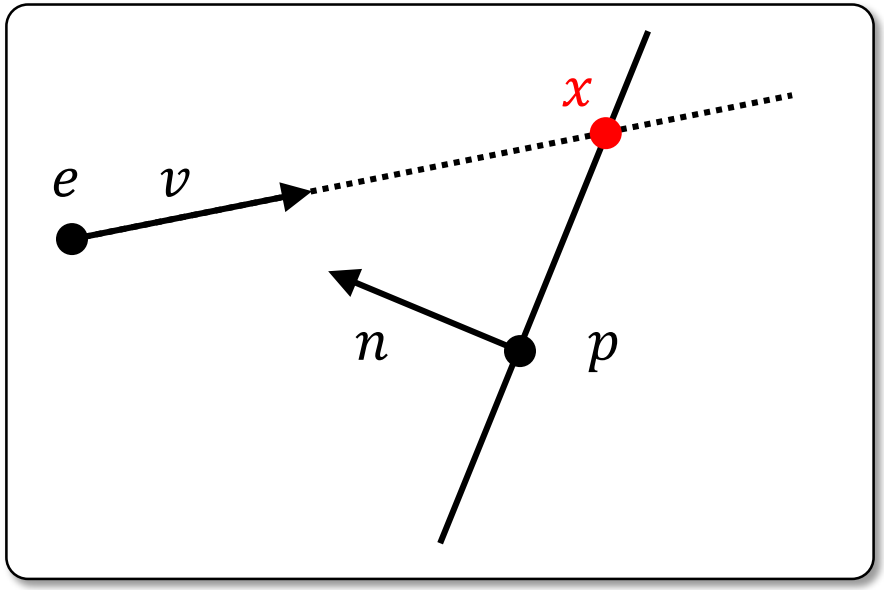
p point in the plane

n normal vector

Substitution of the ray $s(t)$ into the normal form

$$(x - p) \cdot n = 0$$

of the plane yields:

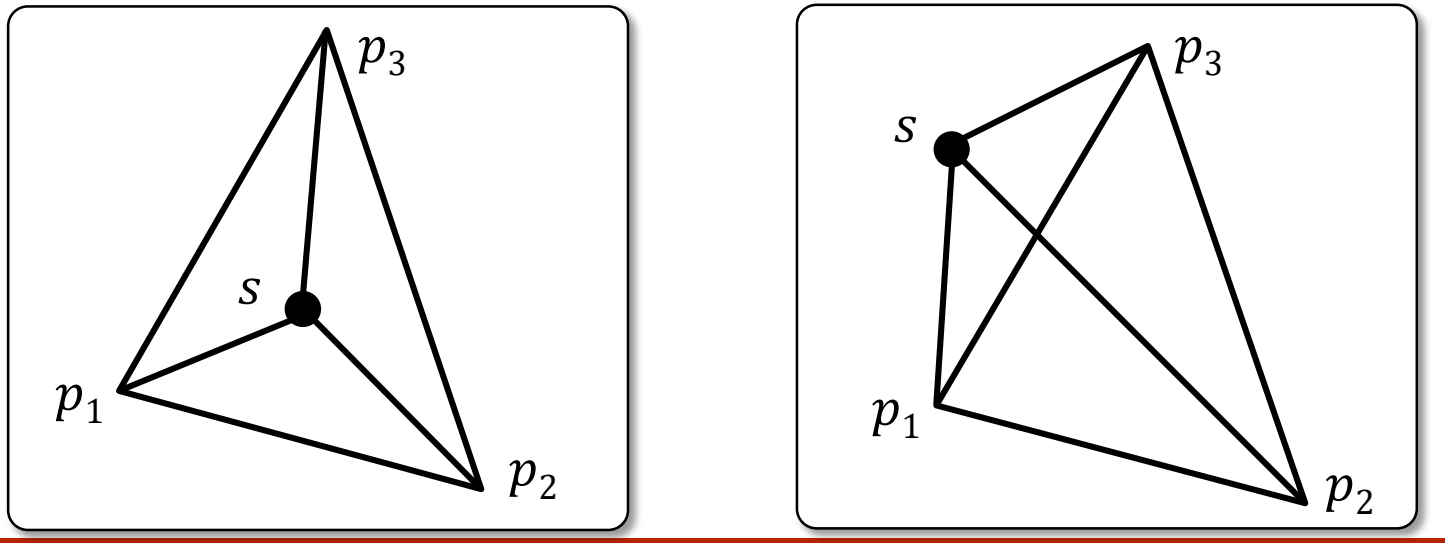


- $(e + t\,v - p) \cdot n = 0$
- $t\ v \cdot n + (e - p) \cdot n = 0$
- $t = (p - e) \cdot n\,/(v \cdot n),$ if $v \cdot n \neq 0.$
- Intersection: $s = r(t) = e + t\,v,$ if $t > 0.$

4.4 Visibility

Example 3:

- For intersections with polygons (triangles) the validity of the intersection s must be tested (is the intersection inside?)
  1. Compute the sum of the (un-signed) areas of sub-triangles.
  2. If this sum is larger than the area of the original triangle, the point s is outside.



4.4 Visibility

Ray Casting: Disadvantages

(1) For each ray, every object in the scene must be tested for intersections.

(2) At a resolution of 1024×1024 with 100 objects in the scene, 100 million intersections must be tested and computed!

- Use back-face-culling.

(3) For typical scenes, up to 95% of computation time is spend for the computation of intersections.

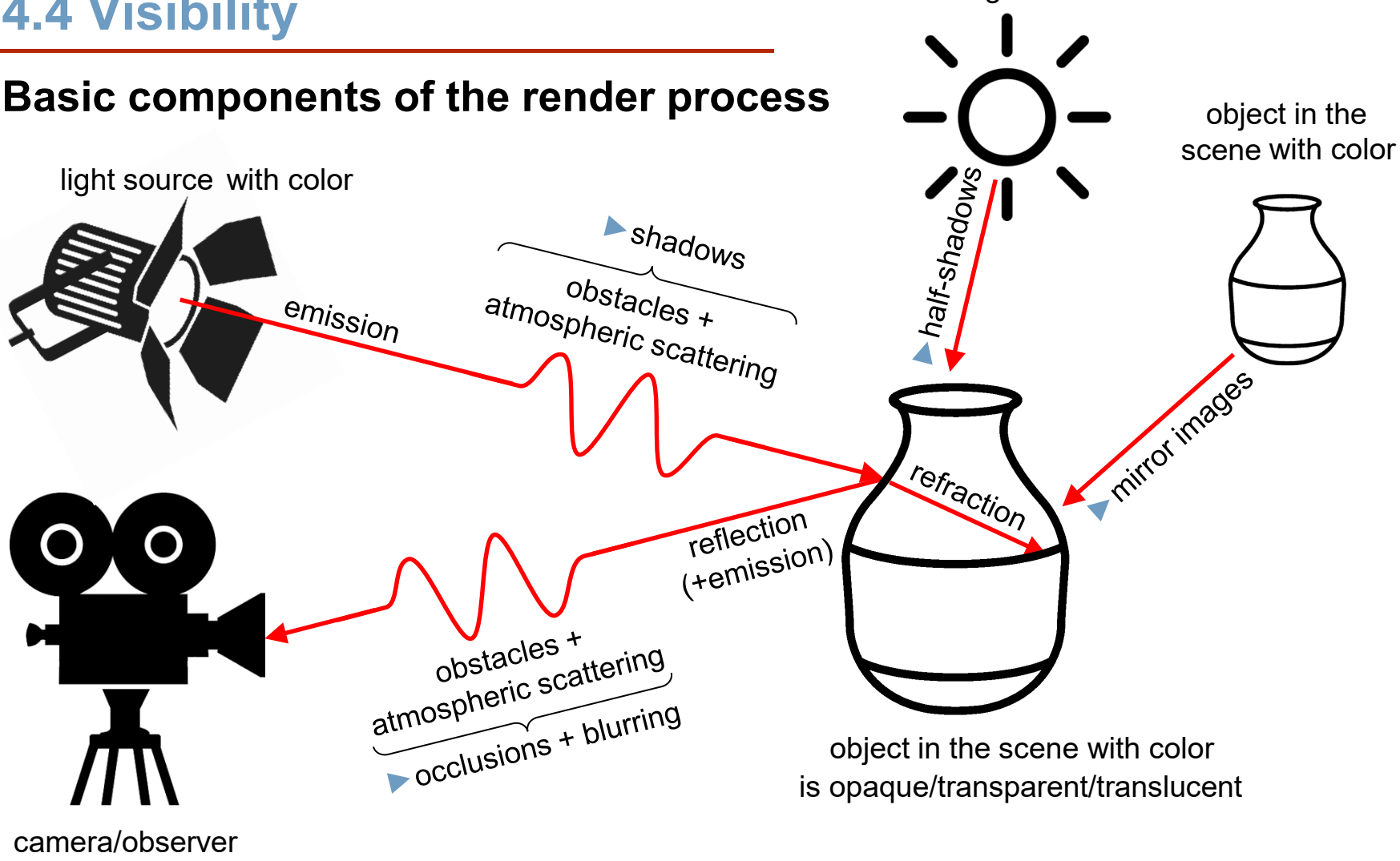
4.4 Visibility

Ray Casting: Acceleration methods

1. Transform the rays onto the z-axis.
   - In this local view coordinate system the intersection with objects is always at $x = y = 0$.
2. **Bounding Boxes:** Complex objects are enclosed in axis aligned bounding boxes.
   - If the ray has no intersection with the bounding box, the enclosed objects do not need to be intersected with the ray.
3. Avoid unnecessary computations of intersections using **hierarchical data structures** and **space partitioning**, such as octrees or bsp-trees.
   - Problem: The generation of suitable hierarchies is difficult.

4.4 Visibility

Basic components of the render process



Content

4.5 Illumination and shading

4.5.2 Illumination and light sources

- **Point light:** The light is emitted from a point in the scene uniformly in all directions into the scene.
- **Direction light:** The light is emitted from a point at infinity in one direction into the scene.
- **Spot light:** The light is emitted within a cone from the apex of a cone.
- **Surface light source:**
  - Soft illumination,
  - technically realized by a plane, cone, or cylinder with “many” light sources.

4.5 Illumination and shading

4.5.3 Illumination model (lighting model, reflection model)

- Illumination- and reflection-algorithms and -models:

  How do you compute the intensity (color) of the pixels, which correspond to an projected object (e.g. polygon) of the scene?

4.5.4 Shading model

- Basic structure, into which an illumination model is embedded:

  When and **where** do you apply the illumination model?

4.5 Illumination and shading

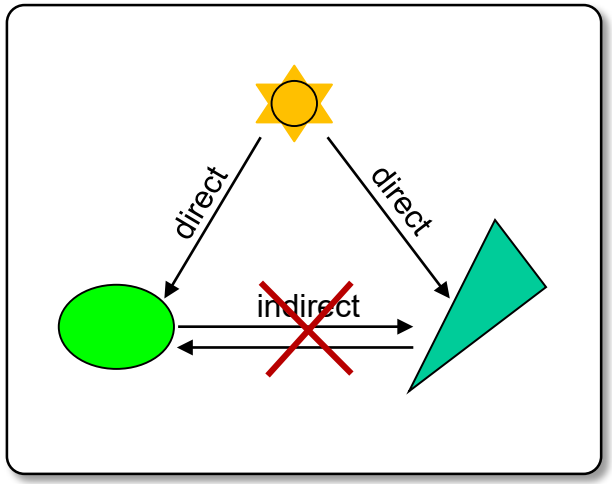
4.5.3 Illumination model (lighting model, reflection model)

- Illumination- and reflection-algorithms and -models:

  Computation of intensities (color) of those pixels an object (e.g. polygon) in the scene is projected onto.

- **Local illumination model:**
  - Computation of the intensity (color) of a point dependent on **direct light** from one light source:

    only direct illumination.
  - e.g. Phong local reflection model, physically based models.

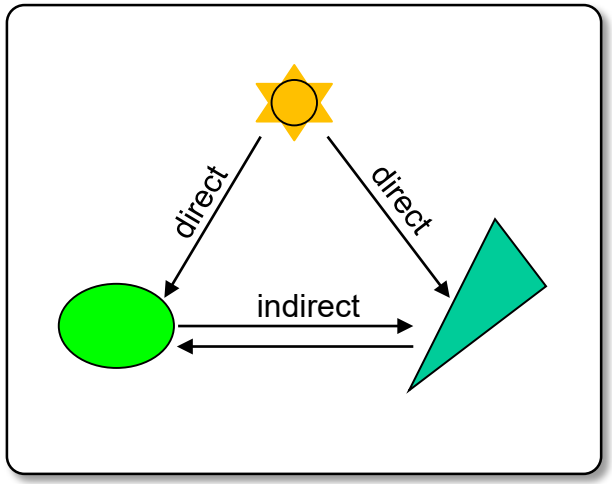


4.5 Illumination and shading

- **Global illumination model:**
  - Computation of the intensity (color) of an object point dependent on
    - **direct light** from one light source and
    - **indirect light**, i.e. from reflections at or transmissions through its own and other surfaces:

      direct and indirect illumination.

  - e.g. ray-tracing (§4.8.1), radiosity (§4.8.2).
  - Global illumination models use often local illumination models or extend/adapt these.



4.5 Illumination and shading

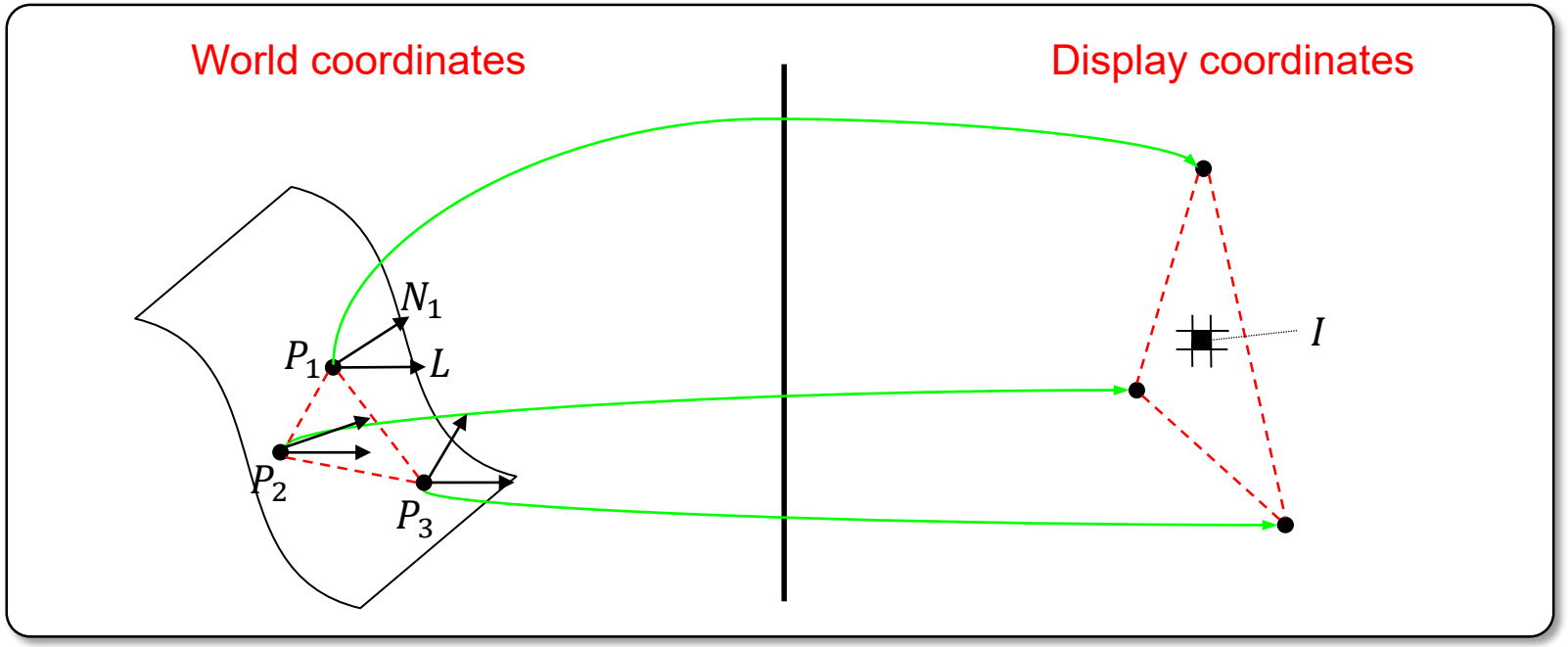
4.5.4 Shading model

- Basic structure, into which an illumination model is embedded.
- A shading model determines, **when** and **where** to apply an illumination model, e.g.:
  - evaluation of an illumination model for each pixel,
  - ► e.g. used for ray-tracing-methods,

  versa
  - evaluation of an illumination model for certain pixels and interpolation of the color for the intermediate pixels,
  - ► *interpolatory shading techniques,* e.g. flat shading, Gouraud shading, Phong shading.

4.5 Illumination and shading

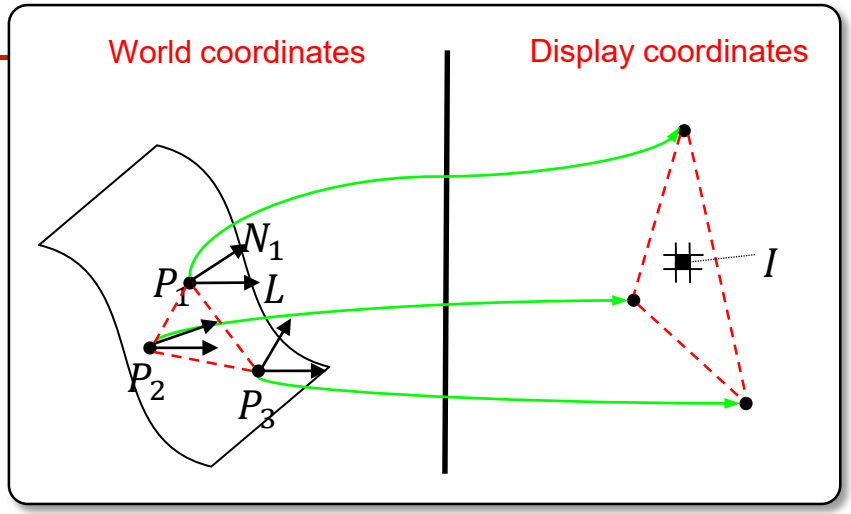
Where to apply an illumination model? Usual practice…



Local reflection model:
Compute light intensity for each P on the surface of an object.

Interpolatory shading algorithm:
Interpolate pixel intensities I from light intensities computed in polygon vertices.

4.5 Illumination and shading



Problem?

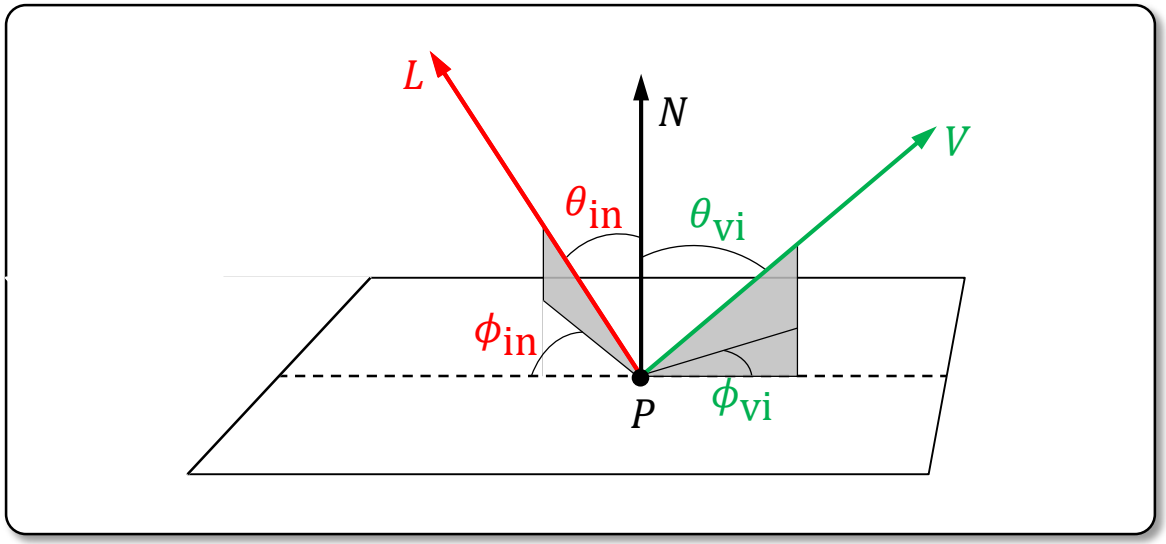
- Illumination and viewing of the scene is in world coordinates.
- Interpolation of intensity values in done in display coordinates.

- But: Central projections are in general not affine!
  - For the interpolation (e.g. linear interpolation) „wrong“ ratios with respect to the world coordinate system are used!
- Although this approach is mathematically wrong, this combination yields fast and visually acceptable results.
  - This combination is used in practice.

Content

4.6 Local illumination models

4.6.1 Geometrical setting



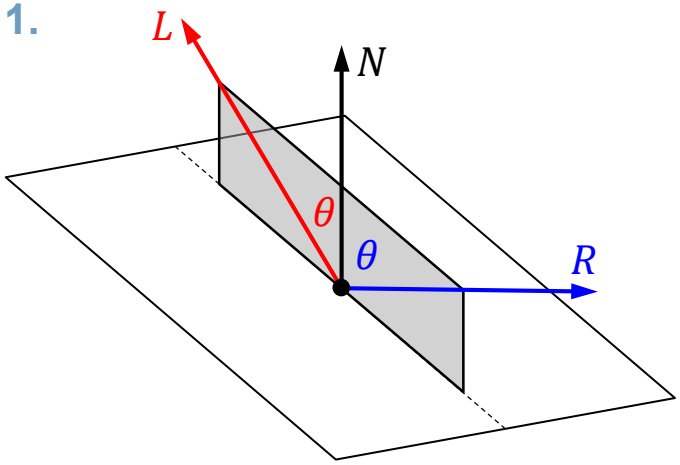
P	point on the surface,
N	surface normal vector in P, normalized,
L	vector from P to a point light source, normalized,
V	vector from P to the eye point (view), normalized,
ϕ_i, θ_i	(local) spherical coordinates (of L and V).

4.6 Local illumination models

4.6.2 Reflection, (perfect) specular reflection

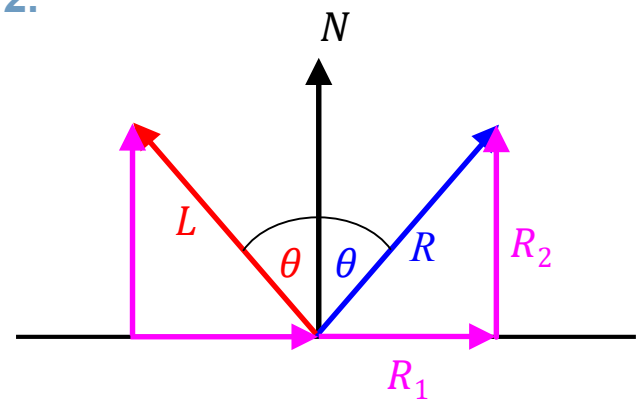
R is vector of reflected ray, normalized.

1.



L and R are co-planar and $\theta = \theta_{\text{in}} = \theta_{\text{ref}}$.

2.



$$R = R_2 + R_1$$
$$= 2R_2 - L$$
$$= 2(L \cdot N)N - L$$

4.6 Local illumination models

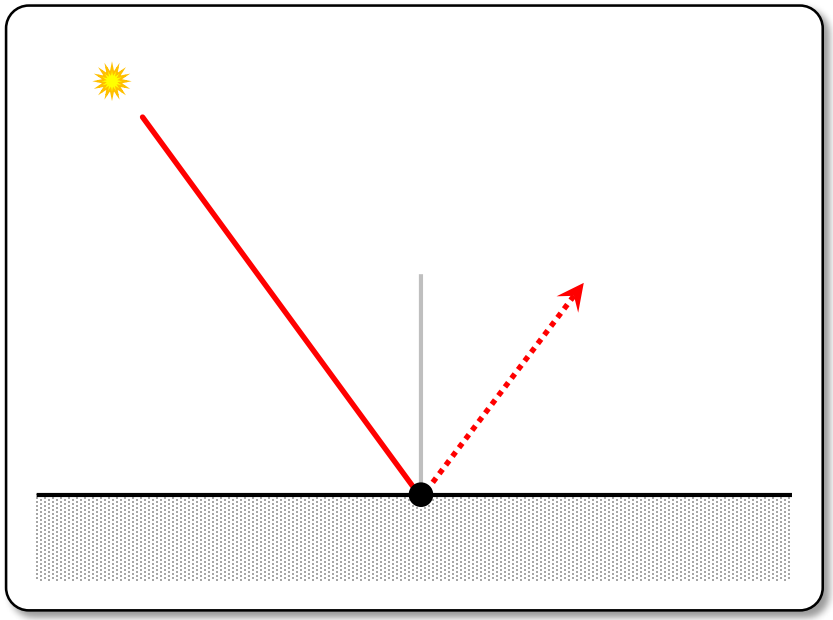
4.6.3 Illumination model of Phong (1975)

Empirical model without physically correct basis,
but yields good practical results!

The model simulates the following physical phenomena:

a) Perfect specular reflection.

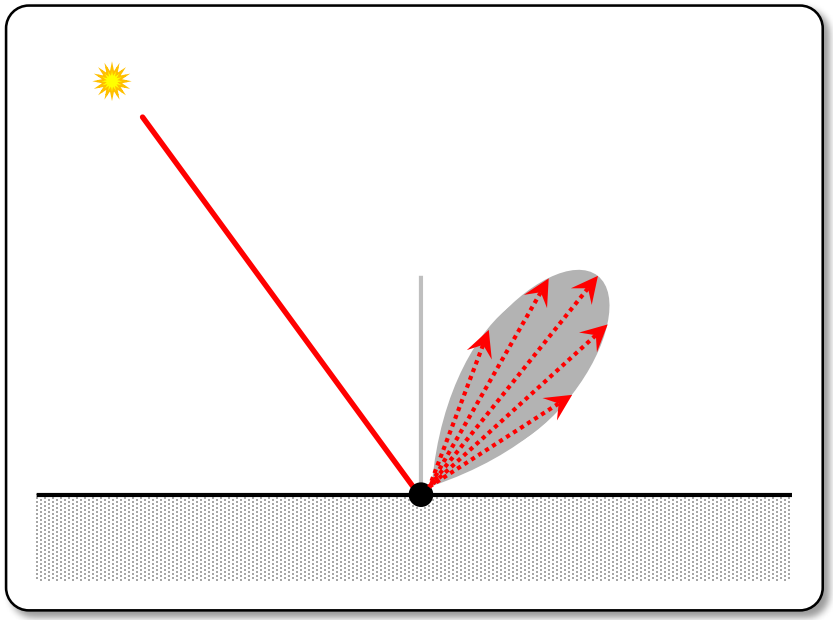
- A light ray is perfectly reflected without scattering at the surface, following the law of reflection.
- Surface: ideal mirror.
- Does not exist in reality.



4.6 Local illumination models

b) Imperfect specular reflection.

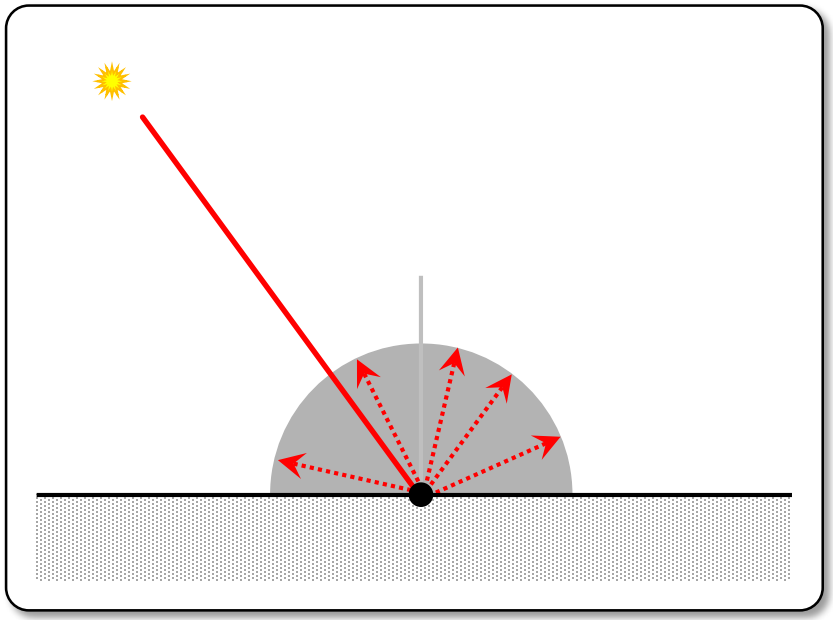
- A light ray is scattered at the surface point, i.e. the is a cone of reflected rays around the perfect reflection direction.
- Surface: imperfect mirror, rough surface.
  - On the microscopic level the surface consists of many tiny mirrors with slightly different normals.



4.6 Local illumination models

c) Perfect diffuse reflection.

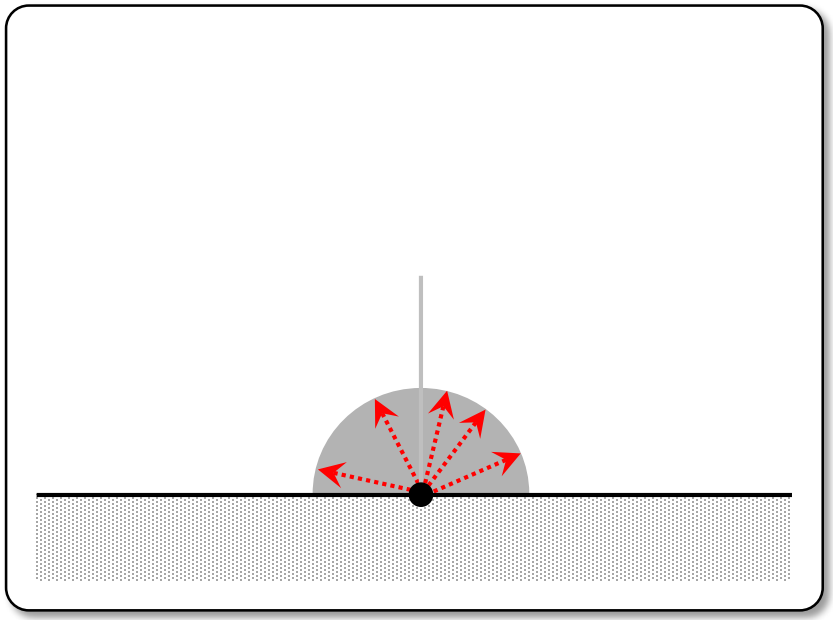
- A light ray is perfectly scattered at the surface, i.e. it is reflected in all directions with the same intensity.
- Intensity depends on incidence angle.
  - The larger the angle to the normal the smaller the intensity.
- Surface: ideal matt surface, does not exist in reality.
  - Approximately: thin layer of powder.



4.6 Local illumination models

d) Ambient light.

- Just an artificial, auxiliary construction to simulate indirect illumination.
  - Often constant and simulates global effects of indirect illumination:
    - Some objects that are not visible by the light source(s), would be rendered in this model as black objects, although in reality these objects are indirectly illuminated.
  - Simple adding of a constant intensity replaces complex global illumination computations.



4.6 Local illumination models

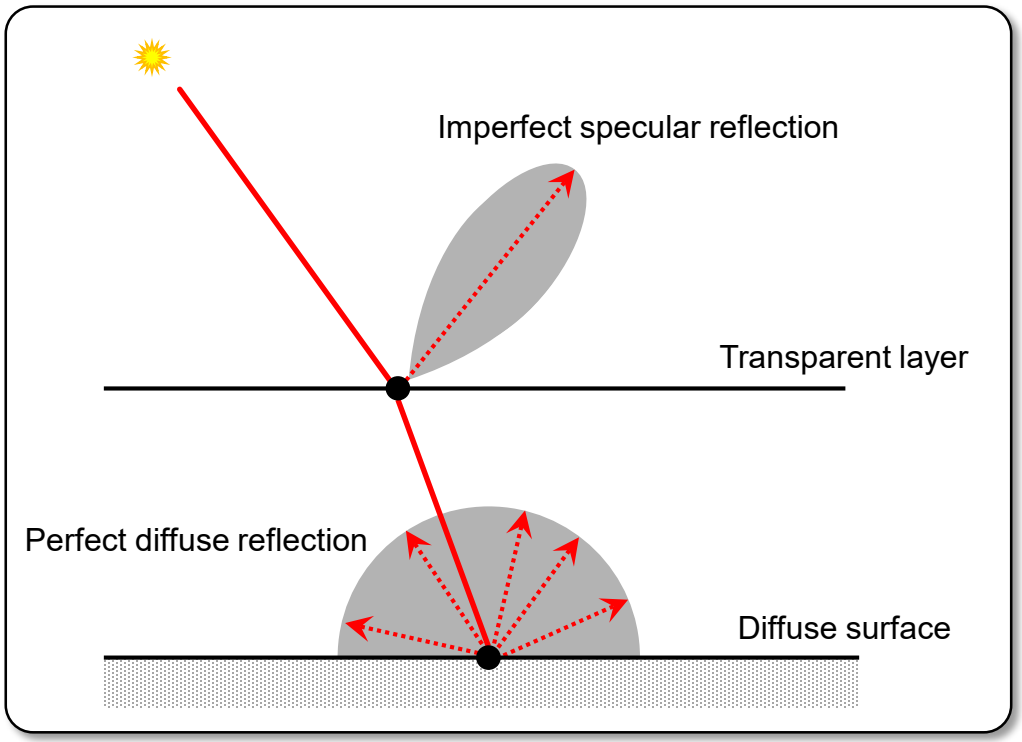
- In the Phong illumination model the reflected light of a surface point is linearly combined from three components:

reflected light = imperfect specular component („highlight“)
+ perfect diffuse component („scattered light“)
+ ambient light („indirect light“)

4.6 Local illumination models

What types of surfaces are described in the model?

- The linear combination of diffuse and specular reflection corresponds to the physics of polished surfaces.
  - E.g. polished wood:
    - a transparent layer: specular,
    - surface: diffuse.



4.6 Local illumination models

The mathematical model (without color)

$$I = k_d \cdot I_d + k_s \cdot I_s + k_a \cdot I_a.$$

- The physical properties of a surface are modeled by the ratio of the individual components.
  - Diffuse component k_d: color of object.
  - Specular component k_s: color of light source.
  - Ambient component k_a: color of environment or ambient scene.
- The weights have to sum to one $k_d + k_s + k_a = 1$,
  - because there is no light energy generated or lost in the scene.

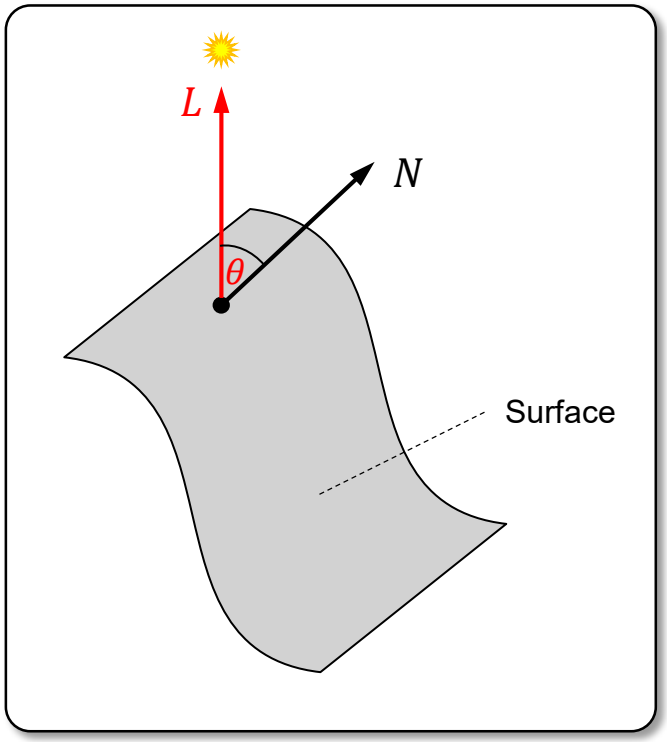
4.6 Local illumination models

Diffuse Reflection (the term $k_d \cdot I_d$)

$$I_d = I_i \cos(\theta) = I_i \ (L \cdot N)$$

I_i Intensity of incoming light

θ Angle between point normal N and light vector L.



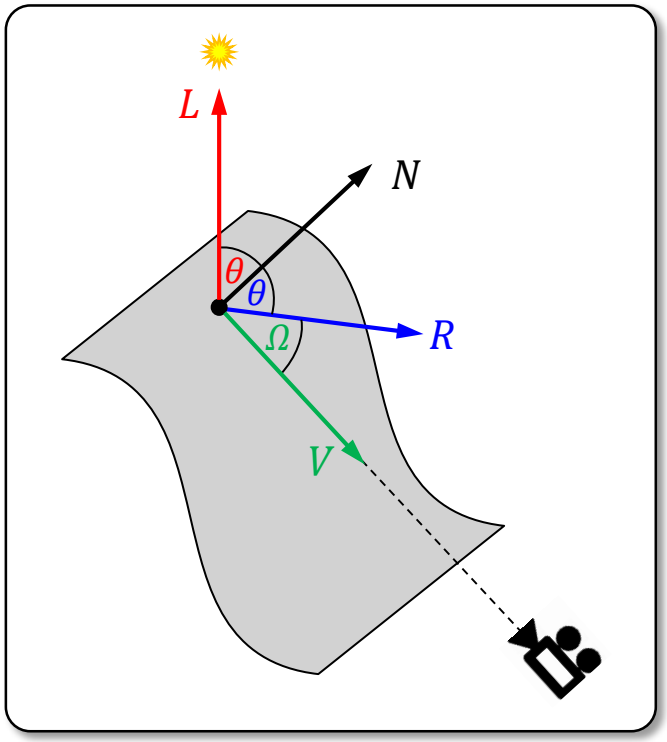
- The diffuse component of the Phong model simulates the **cosine law of Lambert**:

For ideal diffuse surfaces, the intensity (in all directions equal, i.e. **isotropic**) of the reflected light is the cosine of the angle between the surface normal and the light direction.

4.6 Local illumination models

Specular reflection (the term $k_s \cdot I_s$)

- Physically a specular reflection is the mirror image of the light source, smeared over an area of the surface
  - So-called highlight.
- A highlight can only be seen from a direction V that is close to the reflected direction R.



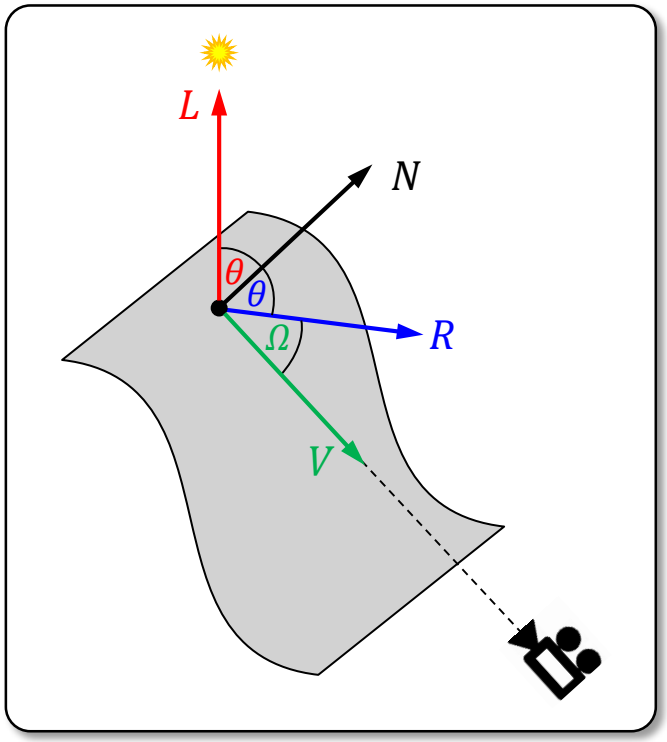
4.6 Local illumination models

- This is simulated via

$$I_s = I_i \cos^n(\Omega) = I_i\,(R \cdot V)^n$$

  Ω angle between V and R

  n simulates degree of perfection of the surface (**shininess**)



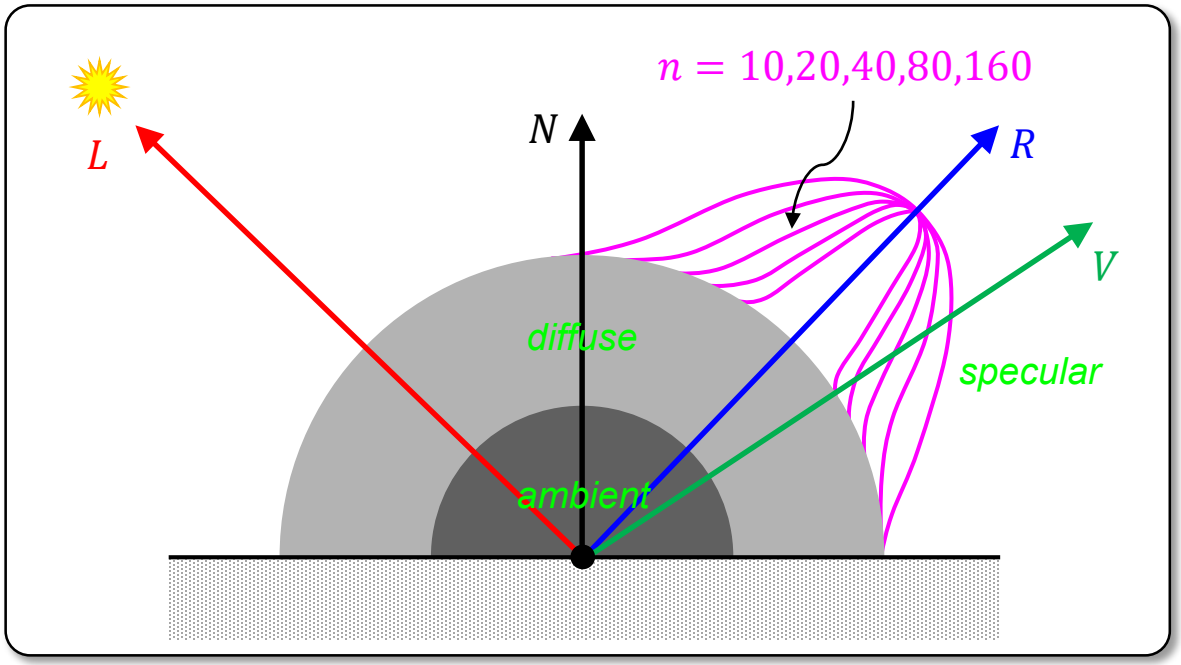
- $n \to \infty$ is a perfect mirror:
  - Light is only reflected in direction R.
  - Different L yield (up to alignment around R) always the same reflection-intensity cone.
  - This is not the physically correct dependence between the reflection and the light direction! Major drawback!

4.6 Local illumination models

Combined model

$$I = k_d \cdot I_d + k_s \cdot I_s + k_a \cdot I_a$$

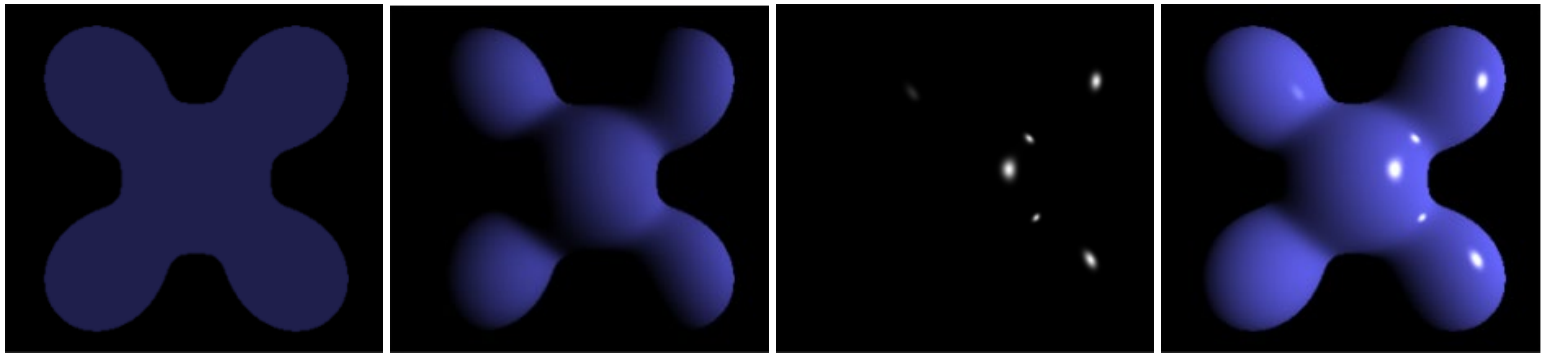
$$= I_i \cdot (\underbrace{k_d \max(0, L \cdot N)}_{\text{diffuse}} + \underbrace{k_s \max(0, R \cdot V)^n}_{\text{specular}}) + \underbrace{k_a \cdot I_a}_{\text{ambient}}$$



4.6 Local illumination models

Complete model

- Effect of the individual components:



source: wikipedia.org

4.6 Local illumination models

Example:

k_a constant

Source: Watt

4.6 Local illumination models

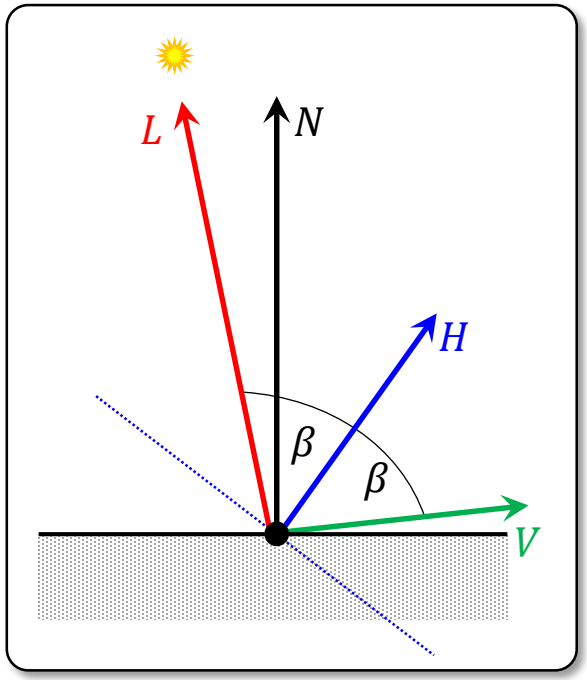
Blinn-illumination model

- If the camera and the light source are infinitely (or sufficiently) far away, the reflection vector R can be replaced by a constant vector H (half-way vector):

$$H = (L + V) / ||L + V||$$

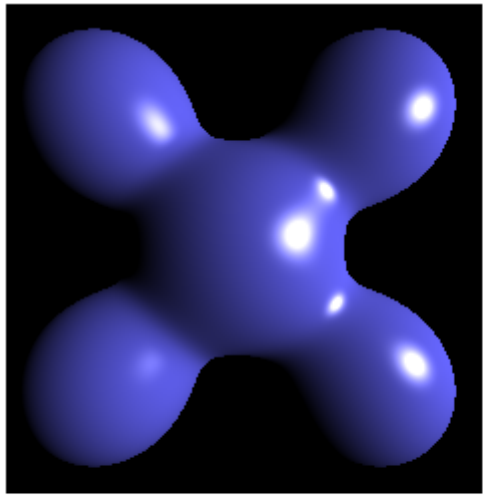
- Instead of $R \cdot V$ use $N \cdot H$, which is different from $R \cdot V$, but behaves similarly.
- This yields

$$I = I_i(k_d(L \cdot N) + k_s(N \cdot H)^n) + k_a I_a$$

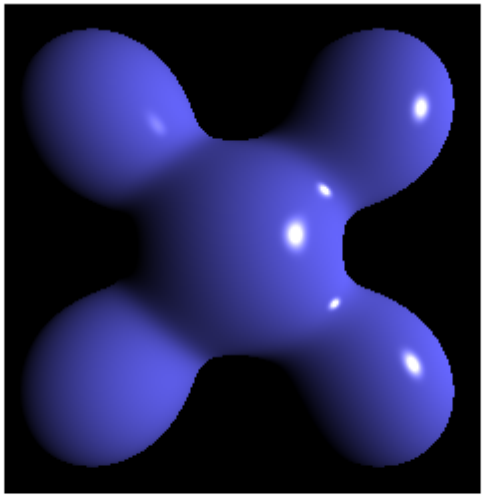


4.6 Local illumination models

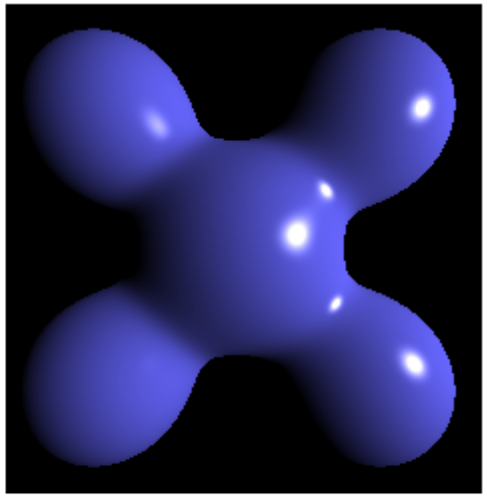
Blinn-illumination model



Blinn illumination



Phong illumination



Blinn illumination with enlarged n

source: wikipedia.org

4.6 Local illumination models

The complete mathematical model (with color)

- For colored objects and light sources the model is applied to each color component I_r, I_g, I_b separately:

$$I_r = I_i(\, k_{dr}(\max(0, L \cdot N)) + k_{sr}(\max(0, N \cdot H))^n) + k_{ar} I_a$$

$$I_g = I_i(\, k_{dg}(\max(0, L \cdot N)) + k_{sg}(\max(0, N \cdot H))^n) + k_{ag} I_a$$

$$I_b = I_i(\, k_{db}(\max(0, L \cdot N)) + k_{sb}(\max(0, N \cdot H))^n) + k_{ab} I_a$$

  - k_{dr}, k_{dg}, k_{db} models the color of the object.
  - k_{sr}, k_{sg}, k_{sb} models the color of the light source.
  - k_{ar}, k_{ag}, k_{ab} models the color of the ambient light.

- **Important:** Bound and crop the intensities to $[0,1]$!

4.6 Local illumination models

Remark

- The illumination model of Phong does not try to simulate optical phenomena physically exact, e.g. no energy conservation!

  The model is purely empirical!

- The local illumination can be computed fast, the images look good:
  - Except for the normal, no further geometrical information is required!
  - Diffuse and specular component are computed locally.
- The color of the specular component is determined by the color of the light source, i.e. by the corresponding constants k_{sr}, k_{sg}, k_{sb}.

4.6 Local illumination models

Major disadvantage of the model

- Mutual reflections and mirror images of surfaces are described only by the constant ambient term to an unsatisfactory level.
- Object surfaces look like plastic, e.g. shiny metal surfaces cannot be modeled.
- Physically based local illumination models, that simulate the **Bi-directional Reflection Distribution Function (BRDF)** physically correct

  or

- completely different approaches, e.g. mapping-techniques (§6).

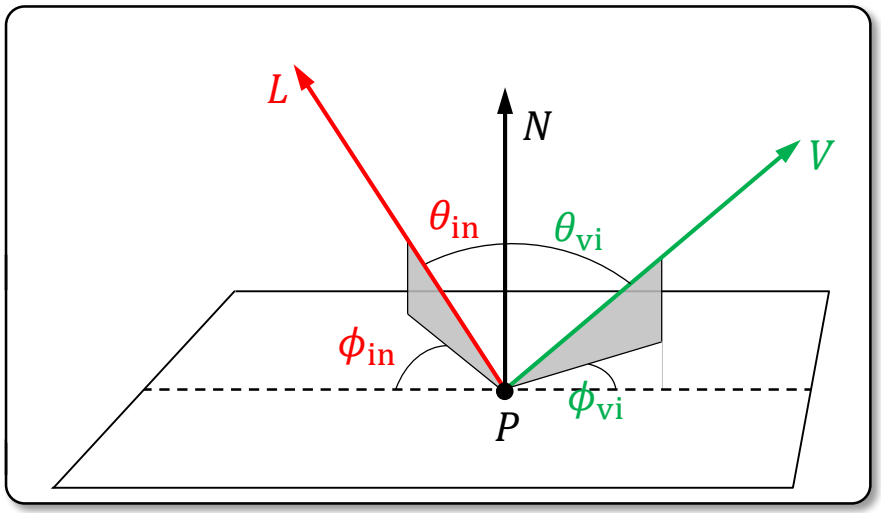
4.6 Local illumination models

4.6.4 BRDF (bi-directional reflection distribution function)

- The light that is reflected from a single surface point is described by a BRDF.
- This function describes the relation of the light that is reflected in an arbitrary direction V in dependence to the direction of the incoming light L.
- For given directions L and V, the relation between the corresponding intensities is given by a 4-variate function

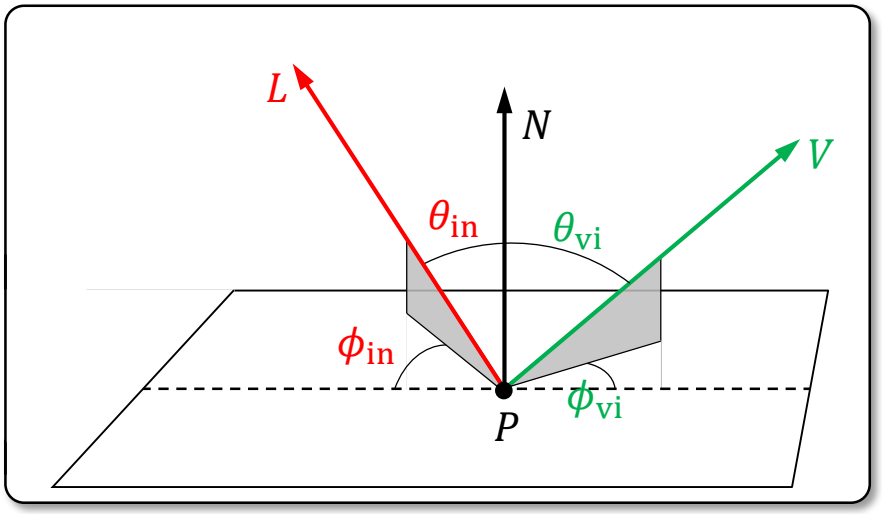
  $\text{BRDF} = f(\theta_{\text{in}}, \phi_{\text{in}}, \theta_{\text{vi}}, \phi_{\text{vi}})$

  (possibly dependent on the wavelength).



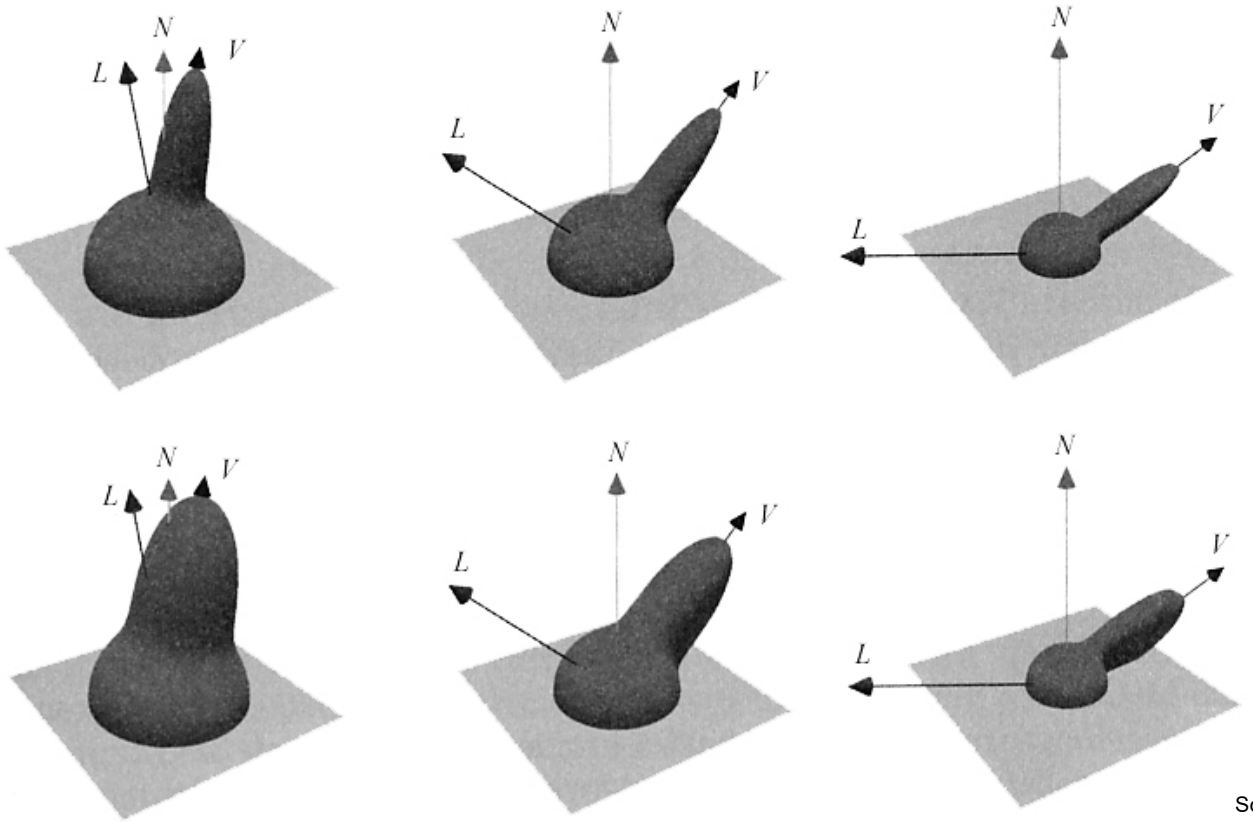
4.6 Local illumination models

- In practice, the incoming light at a single surface point comes from more than one direction.
- The total resulting reflected light is determined by integration over the hemi-sphere.
- Questions:
  - How do you determine a BRDF?
    - e.g. measurements, models.
  - In which resolution do you represent a BRDF?
    - Heuristics, if not given in closed form.
  - How do you store a BRDF efficiently?
    - e.g. matrices, tables.



4.6 Local illumination models

BRDFs for different directions od incoming light: (4d function: L fixed → intensity dependent on V)



Source: Bender/Brill

4.6 Local illumination models

4.6.5 Disadvantage of purely local illumination models

- Models a only single object illuminated by a single point light source in the scene.
- Uses only direct lighting (except for auxiliary workarounds).
- Interaction with other objects are not described in the model, i.e.
  - no indirect light,
  - no transparency,
  - no shadows.

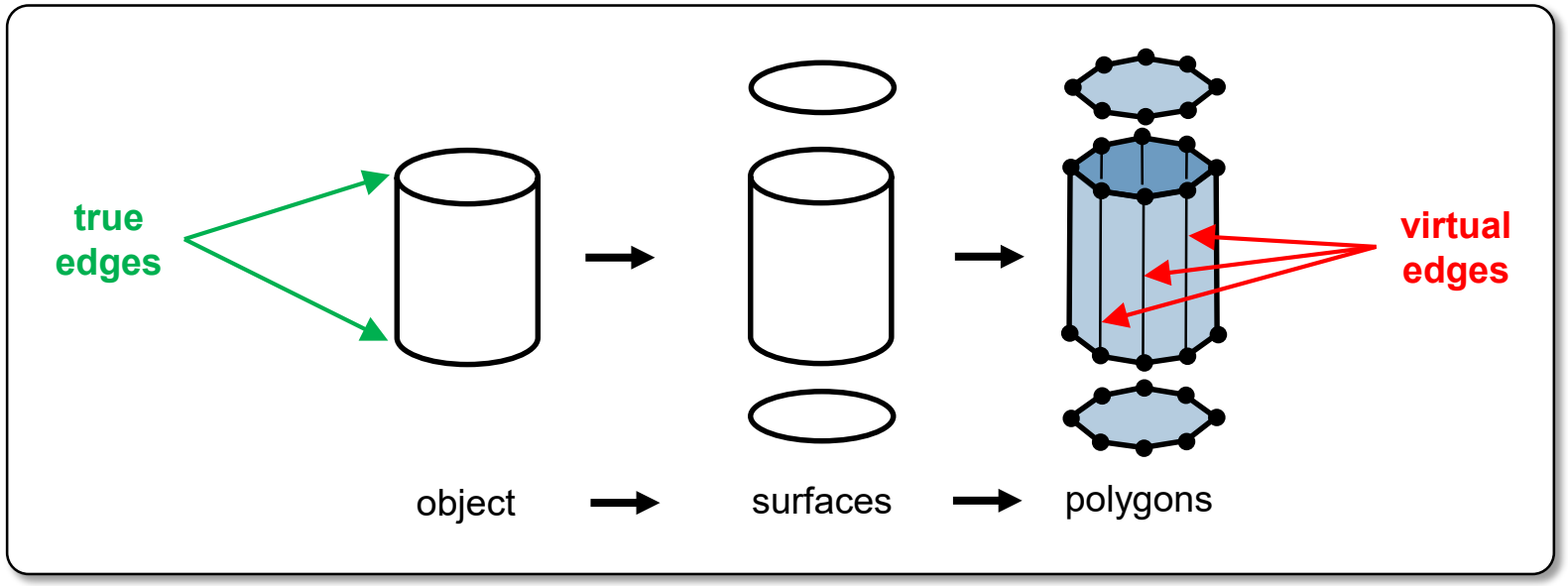
- ▶ Global illumination models (§4.8)

Content

4.7 Interpolatory shading models

How do you apply an illumination model to an object to determine the light intensities on its surfaces?

- We assume, the object is facetted into simple polygons.
- This yields **true** and **virtual** edges.

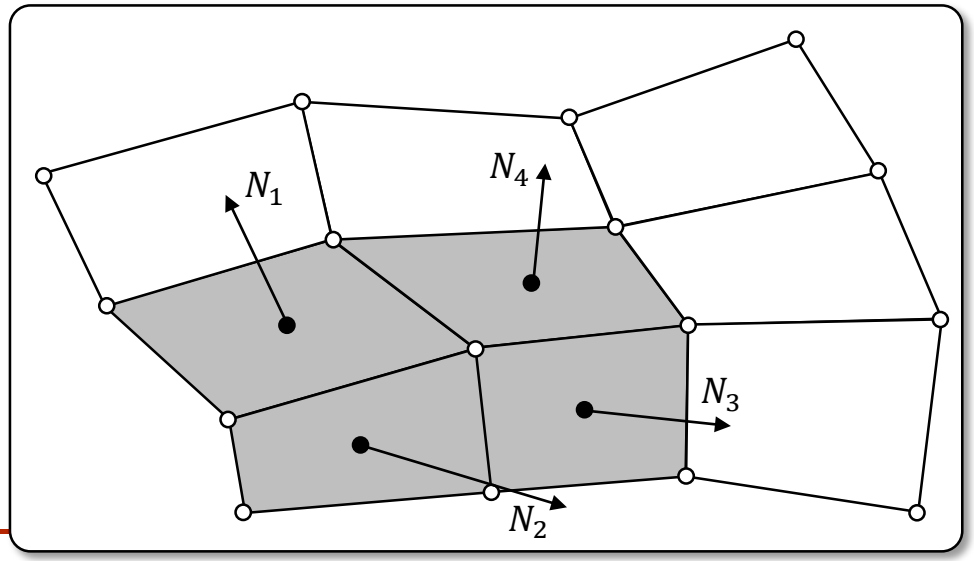


4.7 Interpolatory shading models

4.7.1 Flat Shading

- Per polygon/facet the light intensity is computed
  - exactly once in one particular surface point and
  - all other points in the polygon get the same intensity

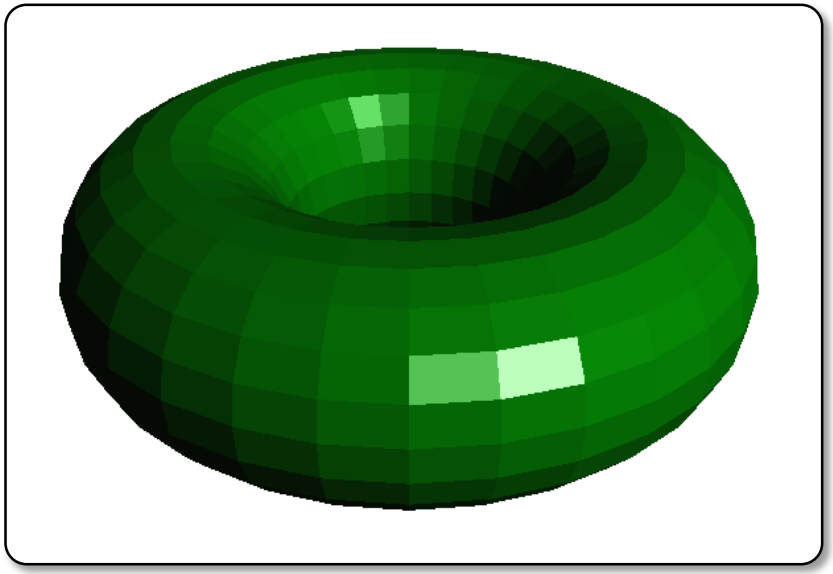
  using the respective illumination model.
- These computations use polygon-/surface-normals in object space, here e.g. N_1, N_2, N_3, N_4.
- The surface points, where the illumination model is evaluated, are e.g. polygon barycenters or polygon corners.



4.7 Interpolatory shading models

- Simple and efficient method, no interpolation necessary (i.e. constant interpolation).
- Edges in the polygon mesh are visible in the image:
  - Objects look facetted,
  - discontinuous intensity across polygon edges.

- „Round“ objects require an extremely large number of polygons!
- Application for pre-views, draft views, visualization of polygonal resolution, high accuracy visualizations, etc.



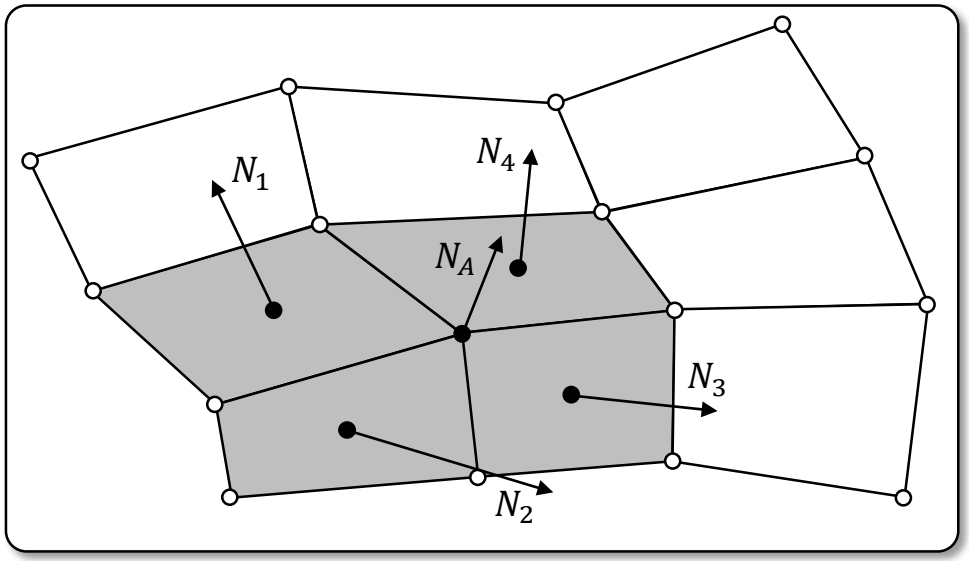
4.7 Interpolatory shading models

4.7.2 Gouraud and Phong Shading

- Methods which use interpolation to smoothen or smear out the (virtual) edges between polygon facets.

  (Polygon mesh is approximation to a curved, smooth surfaces.)

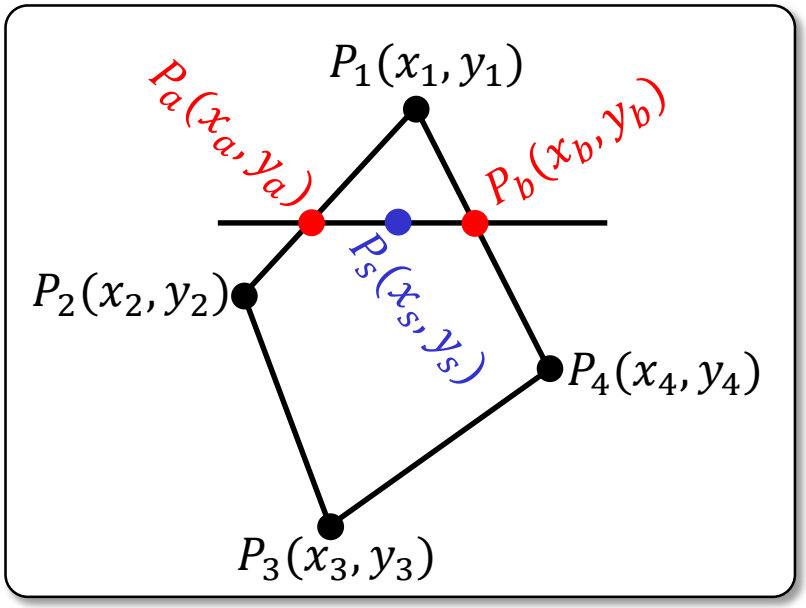
- Basis for the computation are vertex normal in shared polygon vertices.
  (here e.g. N_A, ...)



- Vertex normal are determined by an (weighted) average of polygon normal of abutting polygons.
  Do not forget to normalize!

4.7 Interpolatory shading models

- Bi-linear interpolation in image space:
  - Values of a certain quantity (here e.g. intensity) in the inside and on the boundary of a polygon are computed from the values of this quantity in the vertices using double linear interpolation (usually in image space).
  - Efficient implementations use an incremental scan line approach.



4.7 Interpolatory shading models

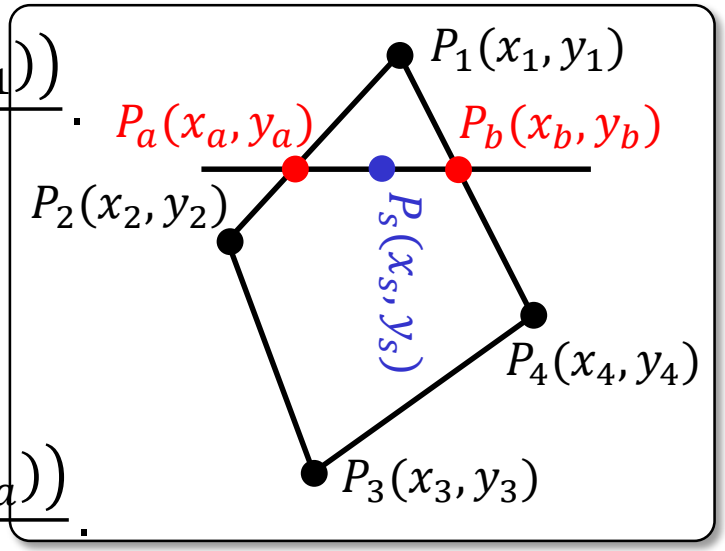
1. Step: Determine values $W(P_1), W(P_2), W(P_3), W(P_4)$.

2. Step: Determine intersections of polygon edges with scan-line P_a, P_b.

3. Step: Determine values $W(P_a), W(P_b)$:

$$W(P_a) = \frac{\big(W(P_1)(y_2 - y_s) + W(P_2)(y_s - y_1)\big)}{y_2 - y_1},$$

$$W(P_b) = \frac{\big(W(P_1)(y_4 - y_s) + W(P_4)(y_s - y_1)\big)}{y_4 - y_1}.$$



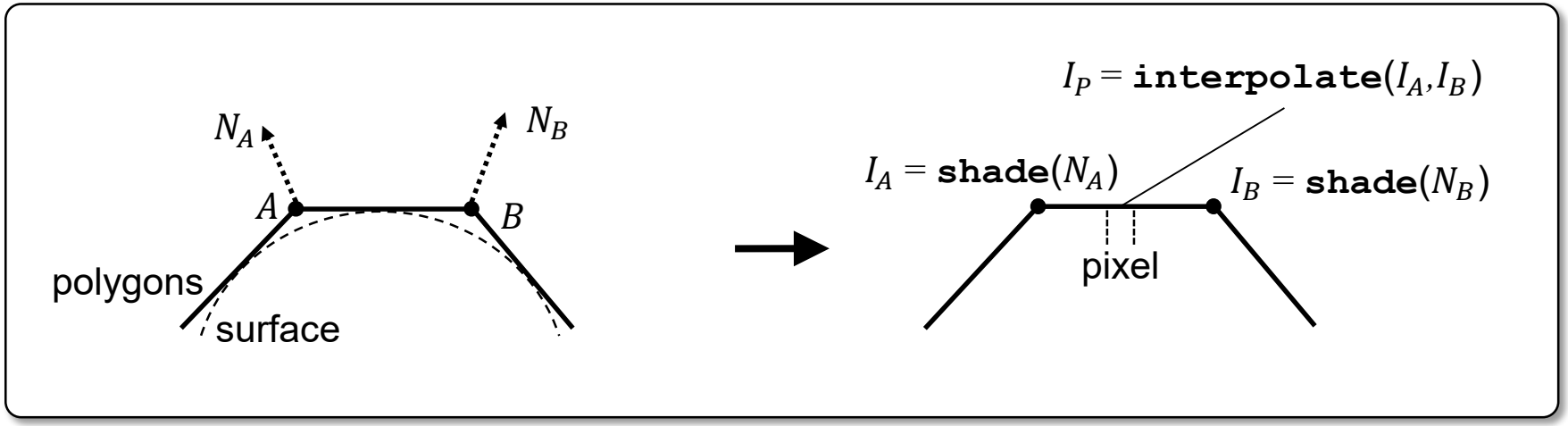
4. Step: Determine value $W(P_s)$.

$$W(P_s) = \frac{\big(W(P_a)(x_b - x_s) + W(P_b)(x_s - x_a)\big)}{x_b - x_a}.$$

4.7 Interpolatory shading models

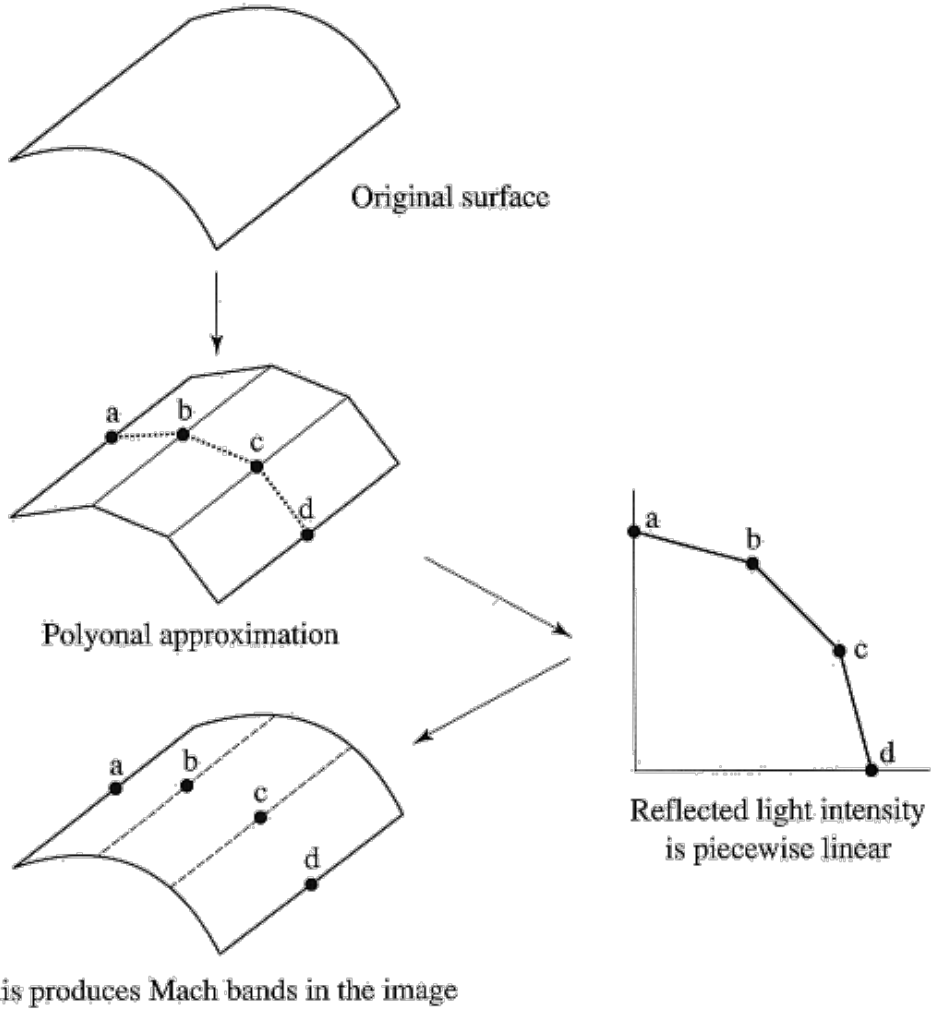
4.7.3 Gouraud Shading

- The illumination model (**`shade`**) is evaluated only in the polygon vertices (A, B) using vertex normal (N_A, N_B).
  - The intensities I_P at projected, inner polygon points are computed via interpolation (**`interpolate`**) of the intensities in the polygon vertices.



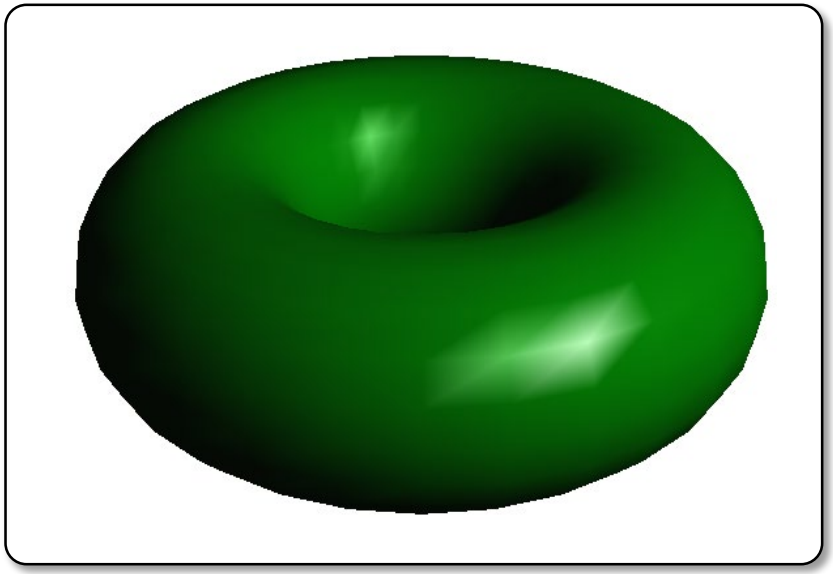
4.7 Interpolatory shading models

- Edges in polygon meshes are smoothed:
  - The intensity across polygon edges is continuous, but
  - not differentiable.
- ➡ Prone to Mach bands effects.



4.7 Interpolatory shading models

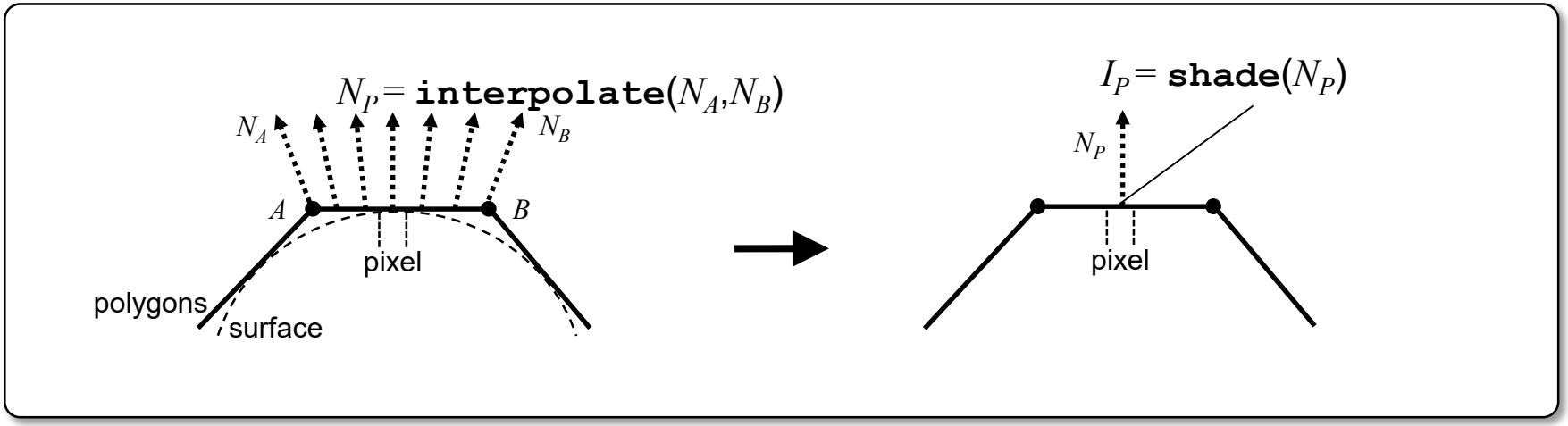
- Highlights are not represented adequately:
  - They occur only if the view direction is close to the reflection direction;
  - but the illumination model is evaluated only in the polygon vertices, which might be outside the highlight.
- High lights can be missed due to sampling errors or occur polygon-shaped (instead of round), see page §5-101.



4.7 Interpolatory shading models

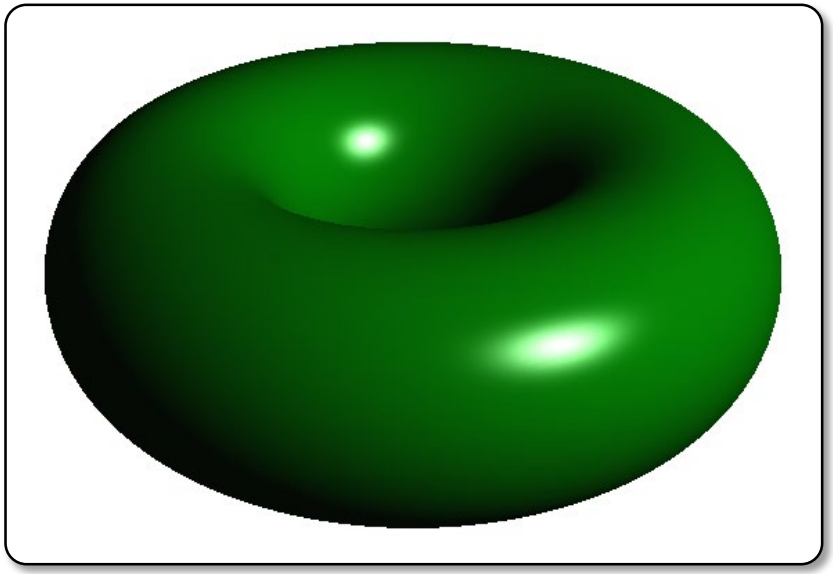
4.7.4 Phong Shading

- The illumination model (**`shade`**) is evaluated for each projected surface point of a polygon:
  - The surface normals N_P at the polygon points are computed via interpolation (**`interpolate`**) from the vertex normal (N_A, N_B).
  - Normalize!



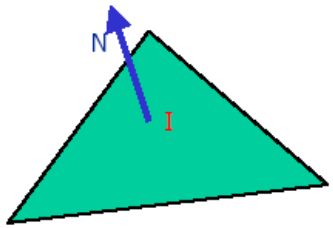
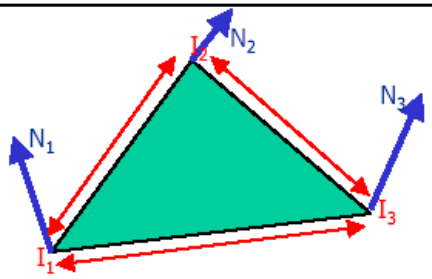
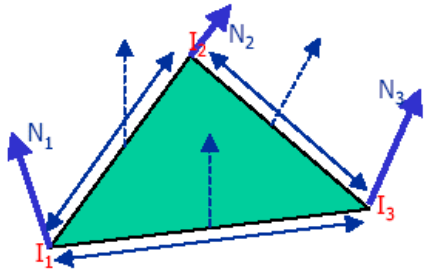
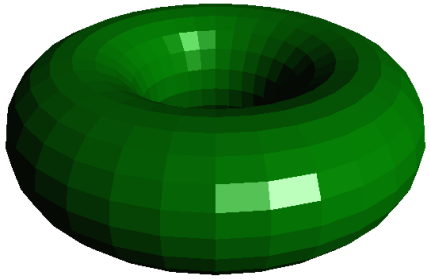
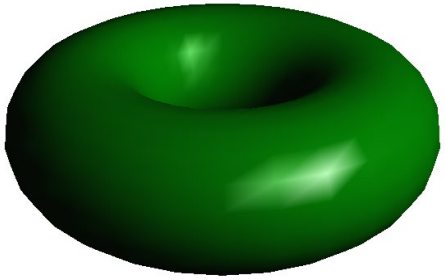
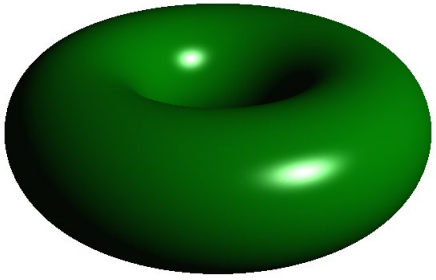
4.7 Interpolatory shading models

- The intensities across edges change continuously and smooth; the interpolation of normal yields the impression of a truly curved surface.
- High computation costs!
- Highlights are represented adequate.
- Phong shading is supported by today's high-end GPUs.
  (i.e. vector interpolation, evaluation of the illumination model in hardware)



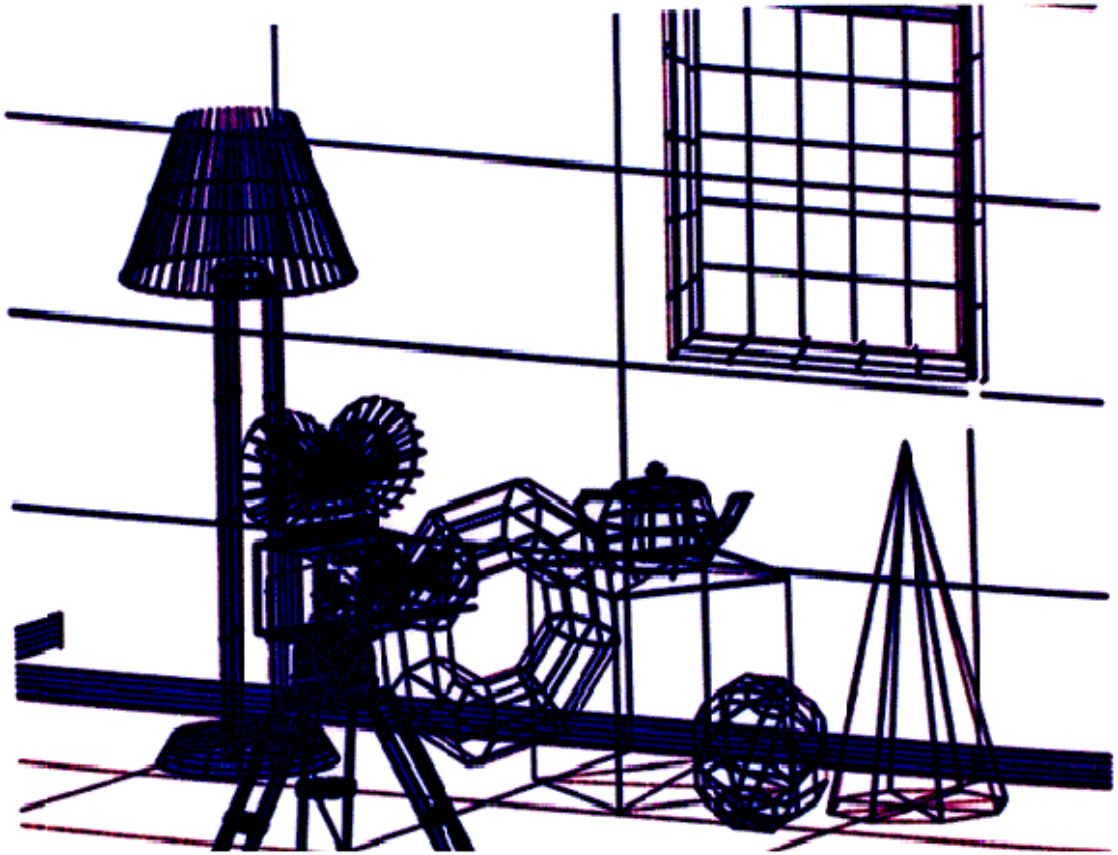
4.7 Interpolatory shading models

4.7.5 Comparison of Flat, Gouraud, and Phong Shading

Flat-Shading	Gouraud-Shading	Phong-Shading
		
One intensity for the complete polygon	Interpolation of intensities at the vertices to the inside of the polygon	Interpolation of normals at the vertices and computation of the intensities at every normal
		

4.7 Interpolatory shading models

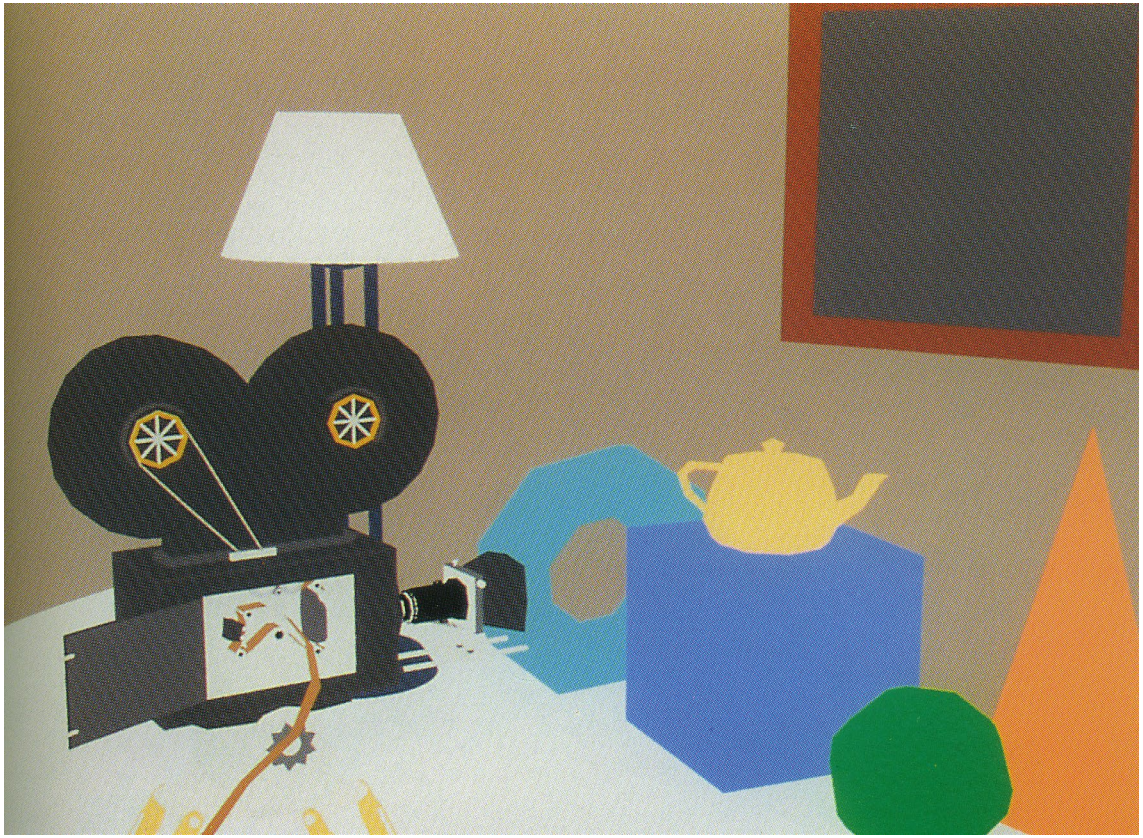
Wire-frame representation



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

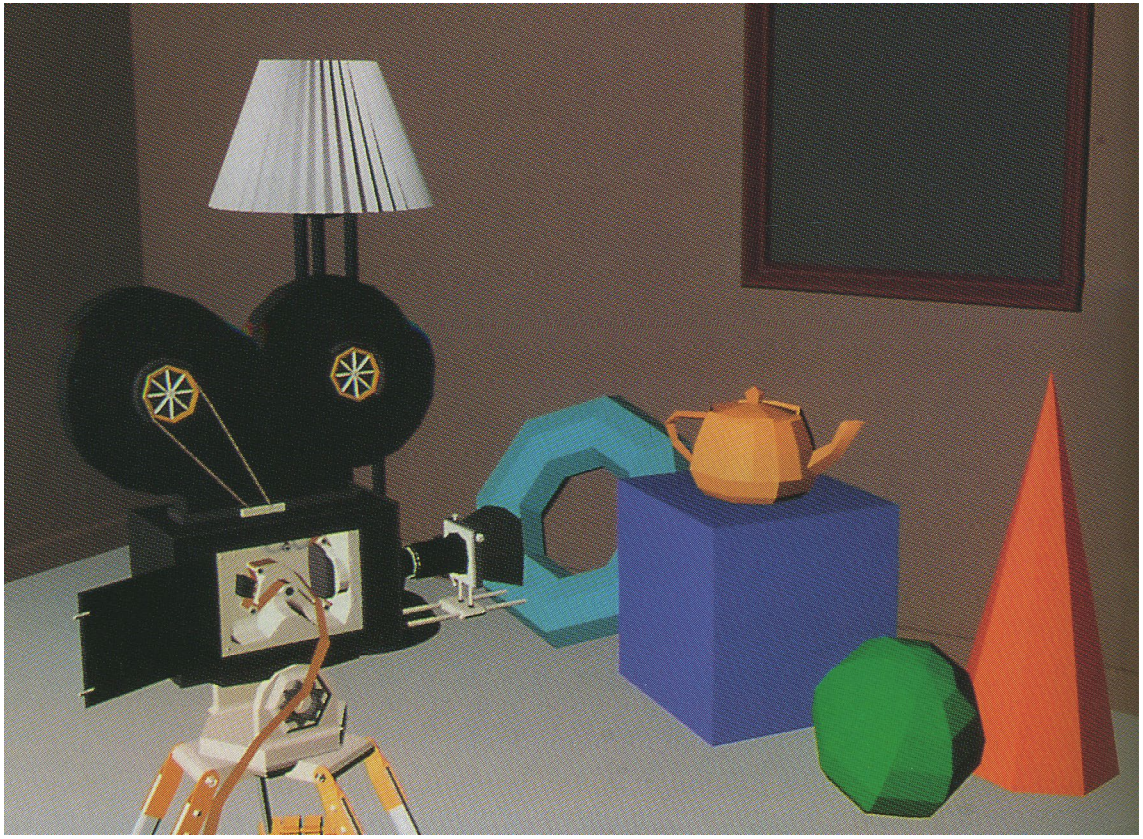
Only ambient illumination



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

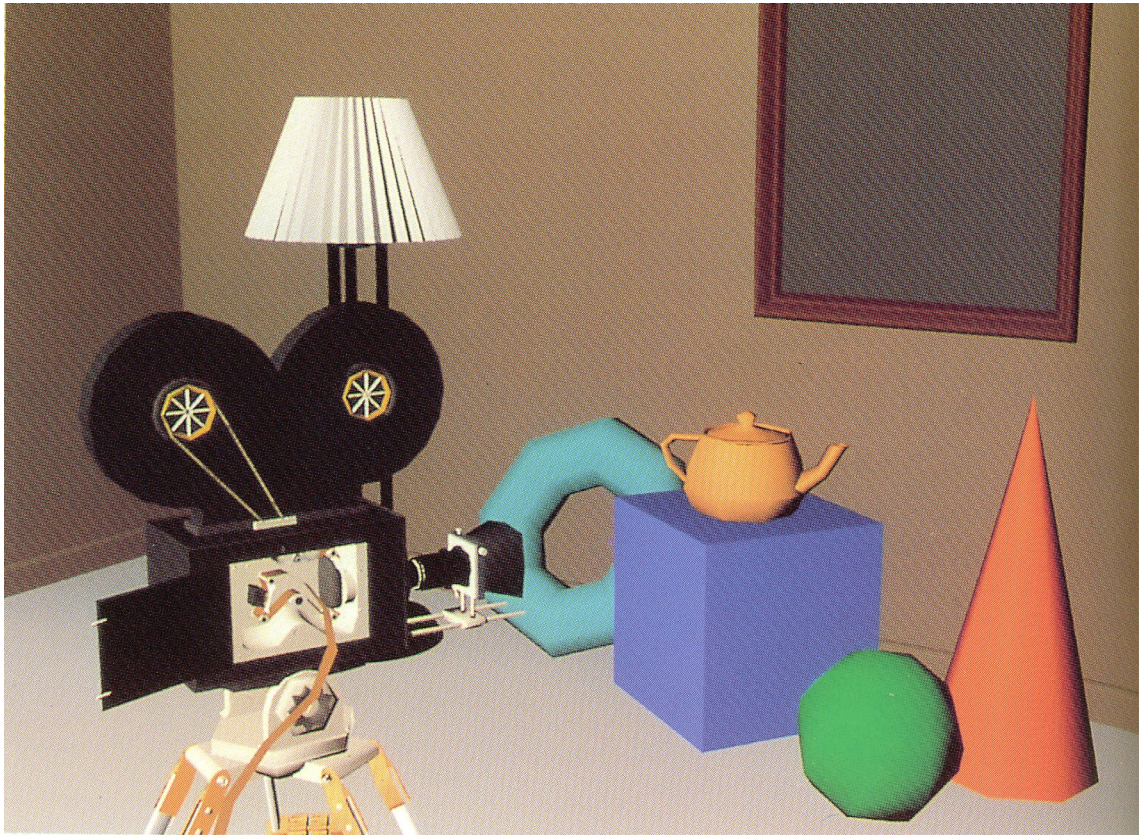
Flat Shading



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

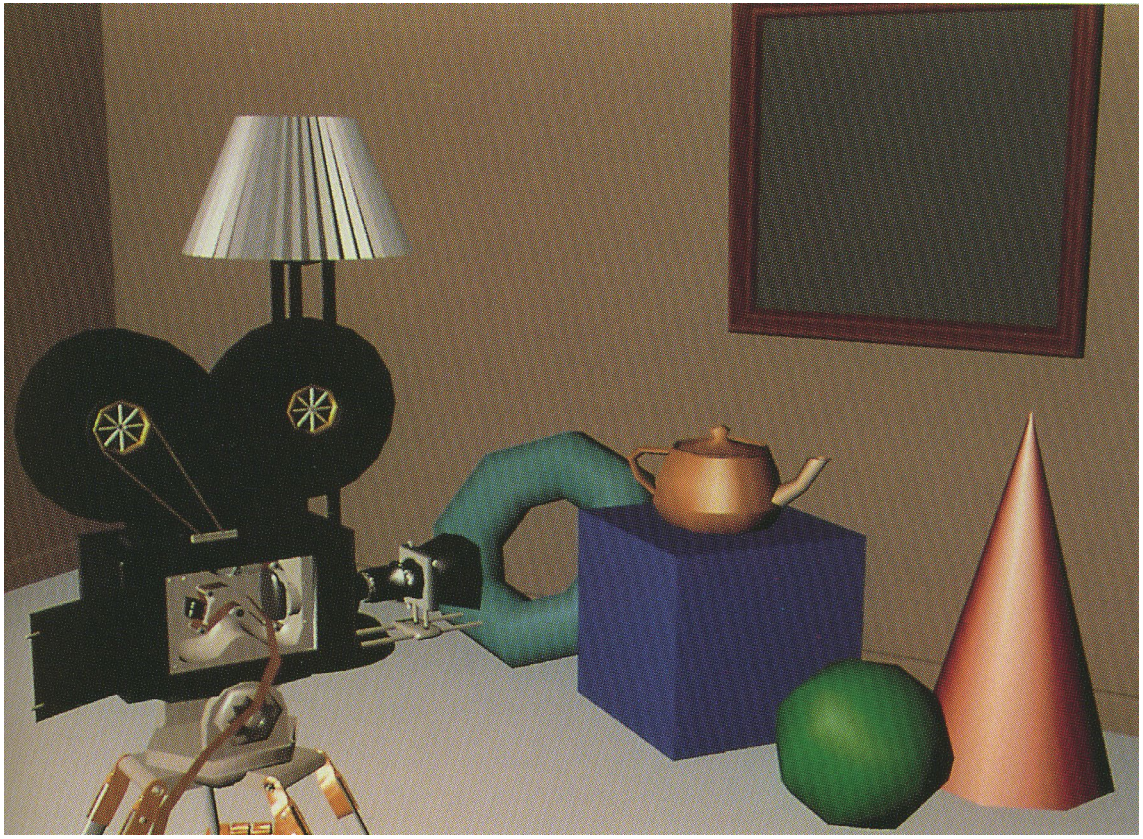
Gouraud Shading (ambient, diffuse illumination)



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

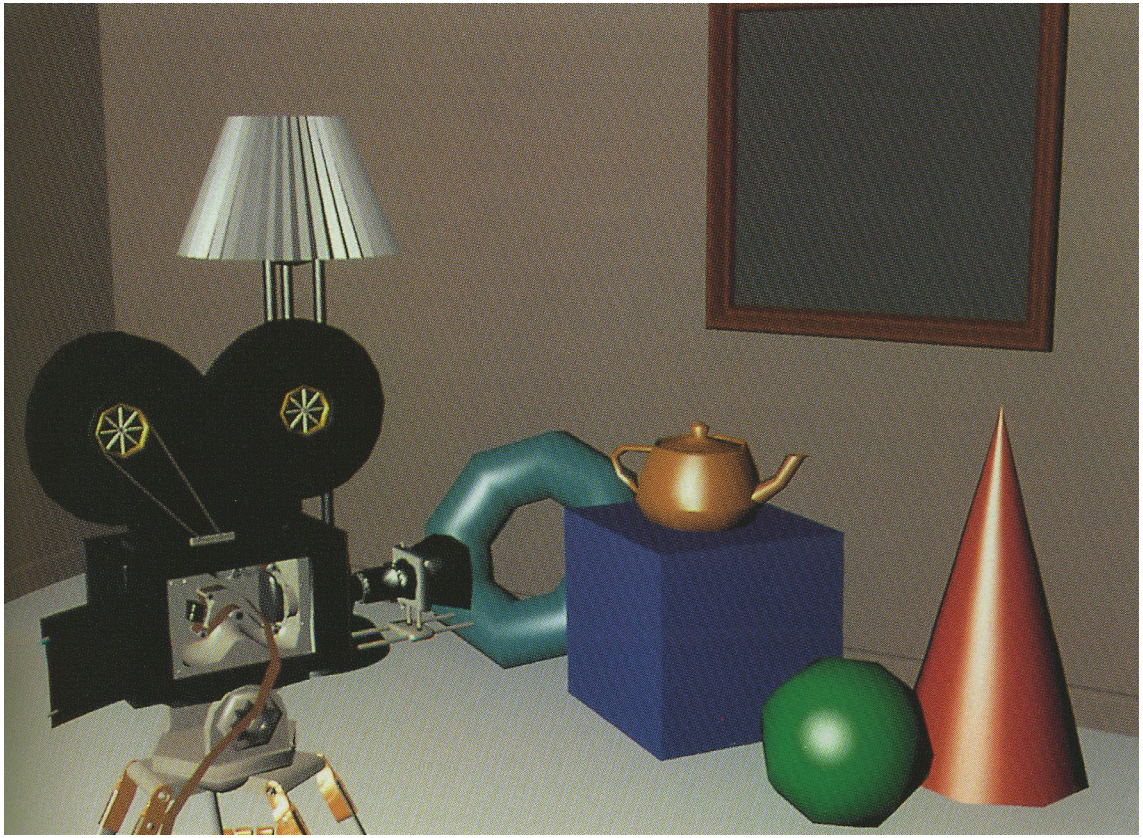
Gouraud Shading (ambient, diffuse, specular illumination)



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

Phong Shading (ambient, diffuse, specular illumination)



source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

Phong Shading of curved surfaces

source: Foley/van Dam/Feiner/Hughes

4.7 Interpolatory shading models

How do you render **feature lines**, polygon edges that should not be smoothed and should be visible in the rendered image, using Gouraud or Phong Shading?

- Polygon vertices, that belong to feature lines, need separate normal vectors for each adjacent polygon.
- Close connection and dependence of shading method and polygonization rsp. triangulation method of the geometric object. (Feature Recognition).

- This causes usually massive problems in practice, when transferring data between different visualization systems!

Content

4.8 Global illumination models

Review: illumination models

- A **local illumination model** considers only the directly incoming light of a single light source.
- **Phong model:** Only local illumination **plus** a constant ambient term, to mimic reflected and transmitted light from other objects.

Better: **Global illumination**

- Directly incoming light as well as reflected and transmitted light in considered in every point of the scene.

4.8 Global illumination models

Two approaches:

4.8.1 Ray tracing:

1. Ray Casting plus
2. tracing of reflected and refracted rays plus
3. computation of shadows.

- Dependent on position of eye point.

4.8.2 Radiosity:

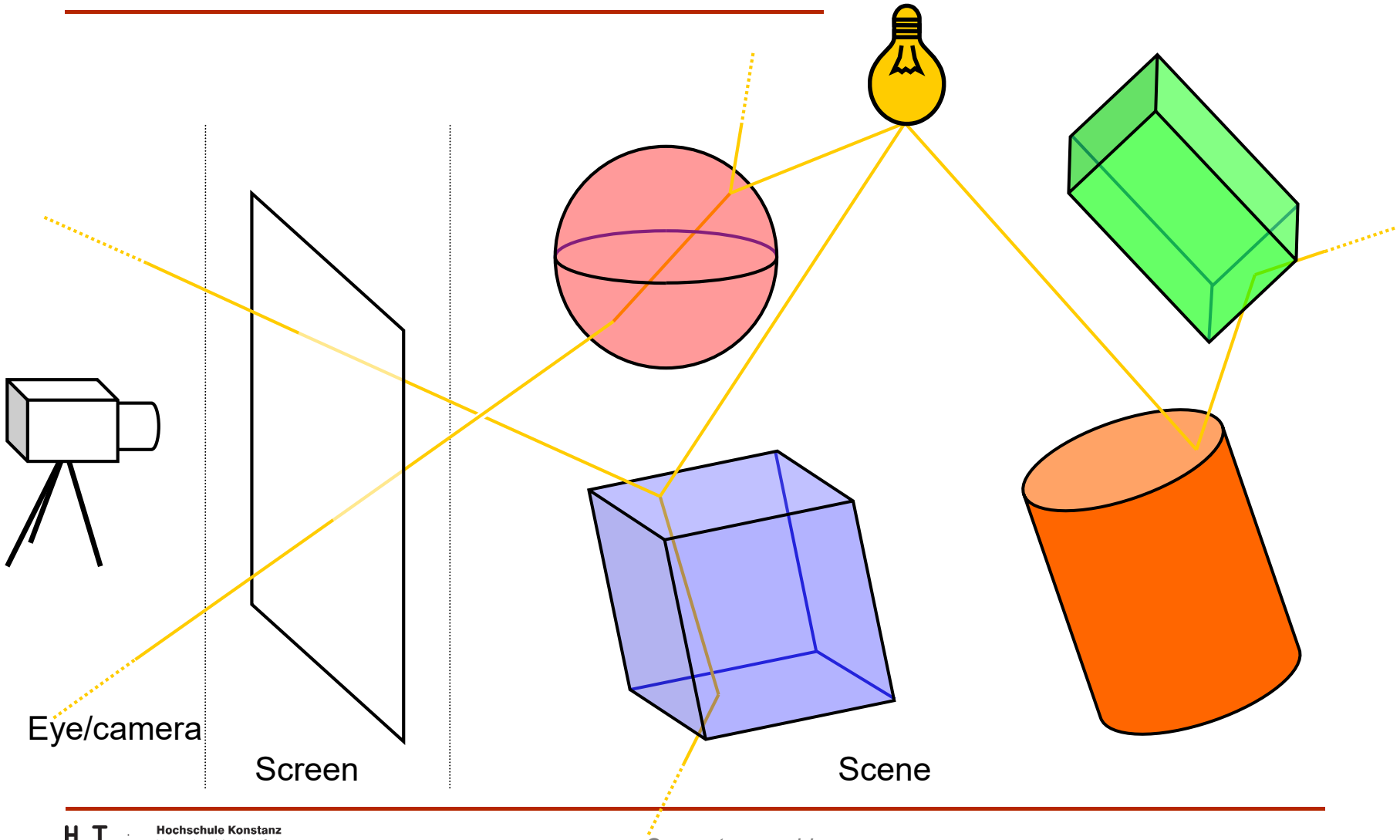
- Separation of visibility tests from intensity computations.
- All interactions of the light with objects of the scene are pre-computed.
- Independent on position of eye point.

4.8 Global illumination models

4.8.1 Ray-Tracing

- Ray Tracing is suitable particularly for the modeling of **directed light** (many specular / transparent objects).

4.8 Global illumination models

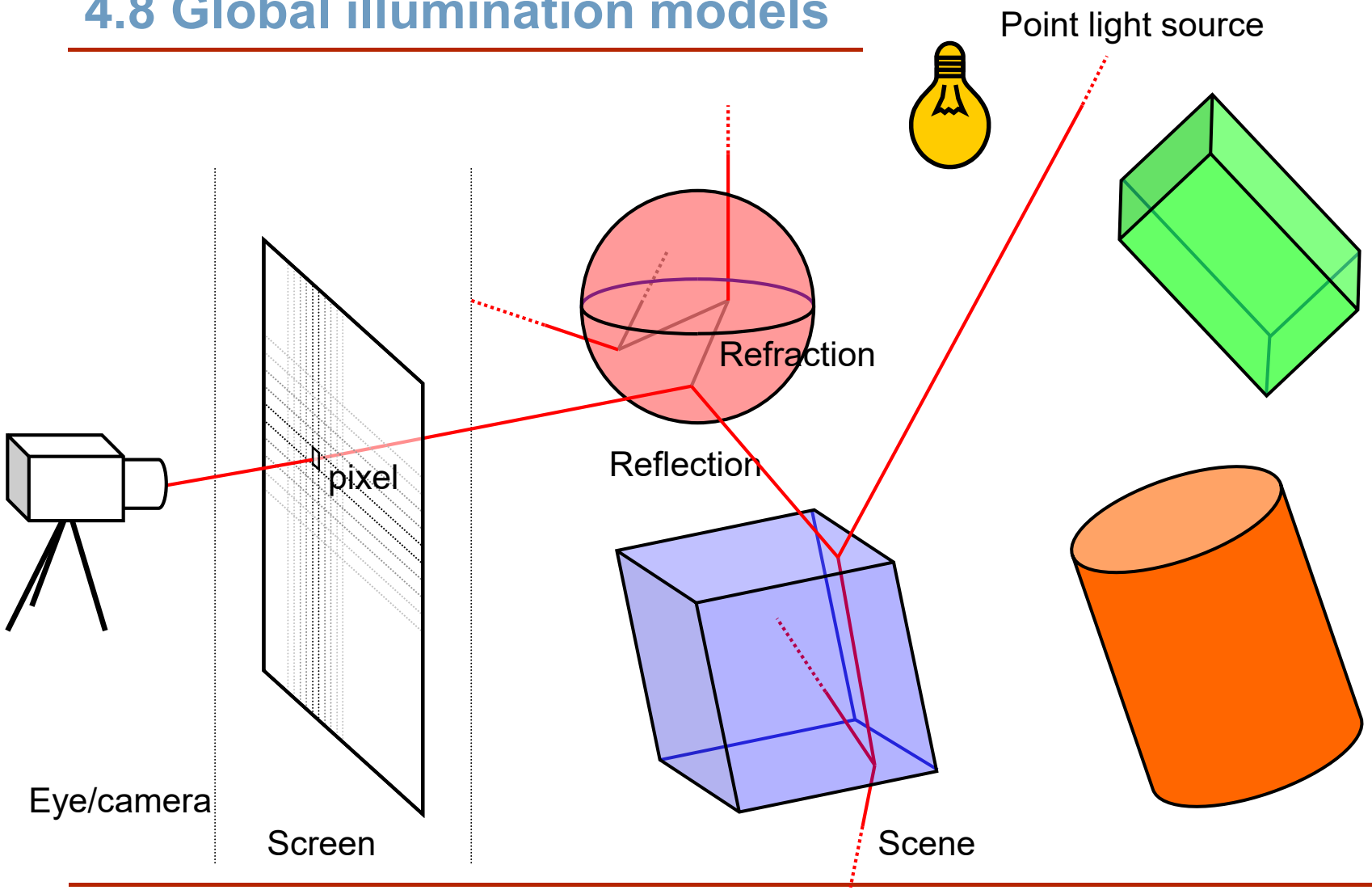


4.8 Global illumination models

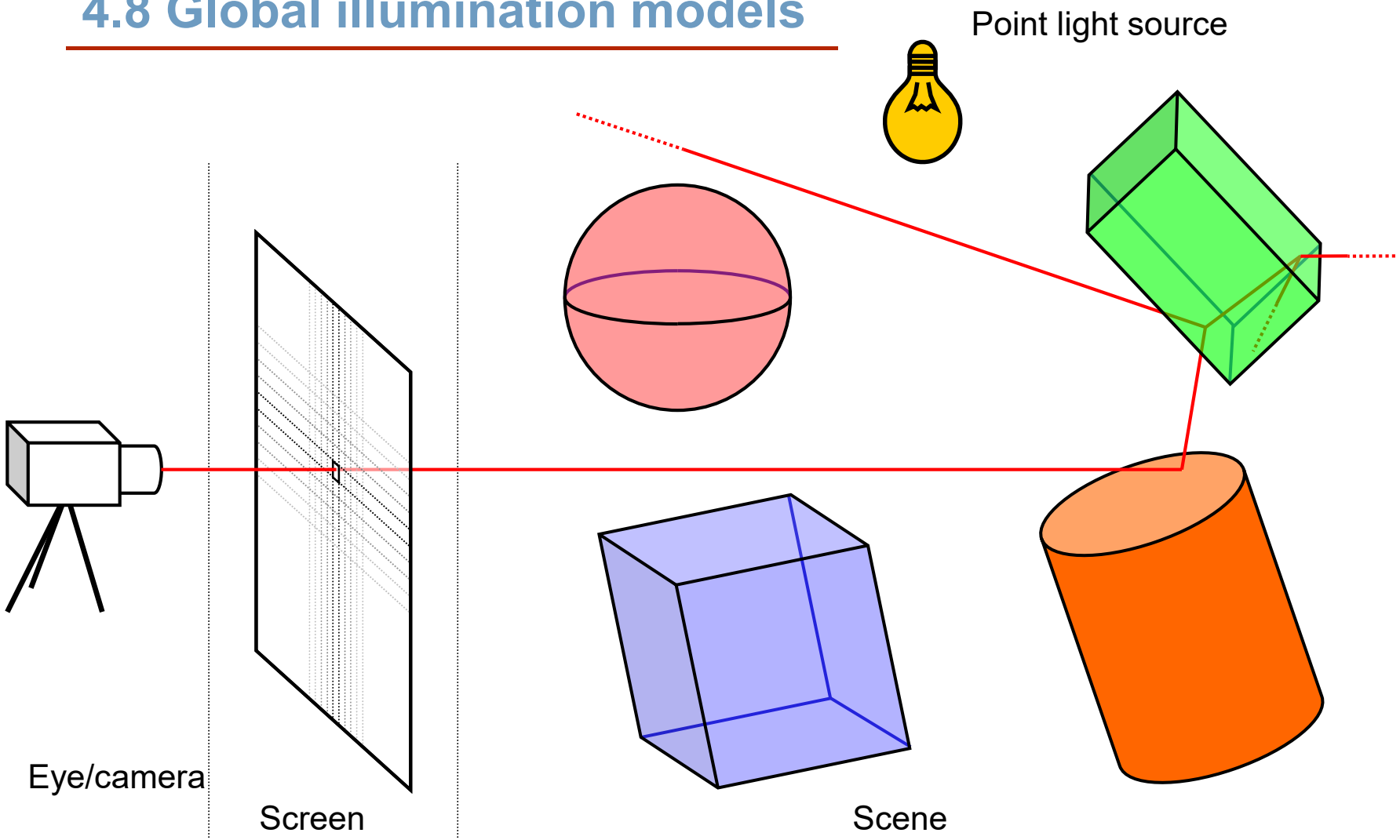
4.8.1 Ray-Tracing

- Ray Tracing is suitable particularly for the modeling of **directed light** (many specular / transparent objects).
- **backward ray-tracing**:
  Because most of the rays from the light source do not hit the eye/camera, the rays are traced backwards from the eye to the first surface and subsequently toe the light sources and further surfaces.
  a. The rays are traced through every pixel into the scene and
  b. for every intersection with an object the direct light components (local illumination model)
  c. and reflected and refracted light components are determined.
  ➨ This ray branching implies a tree structure for the ray-data-structure.

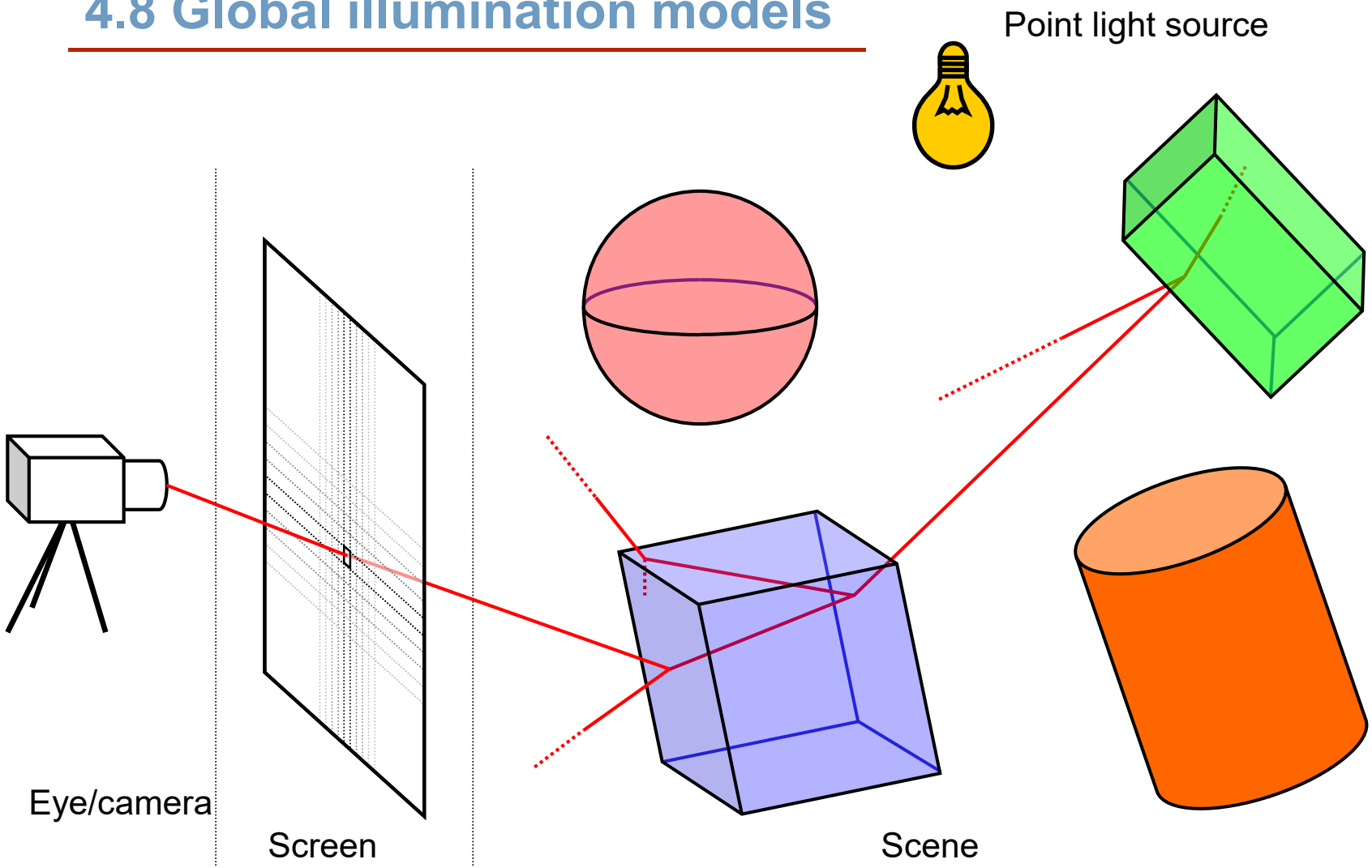
4.8 Global illumination models



4.8 Global illumination models



4.8 Global illumination models

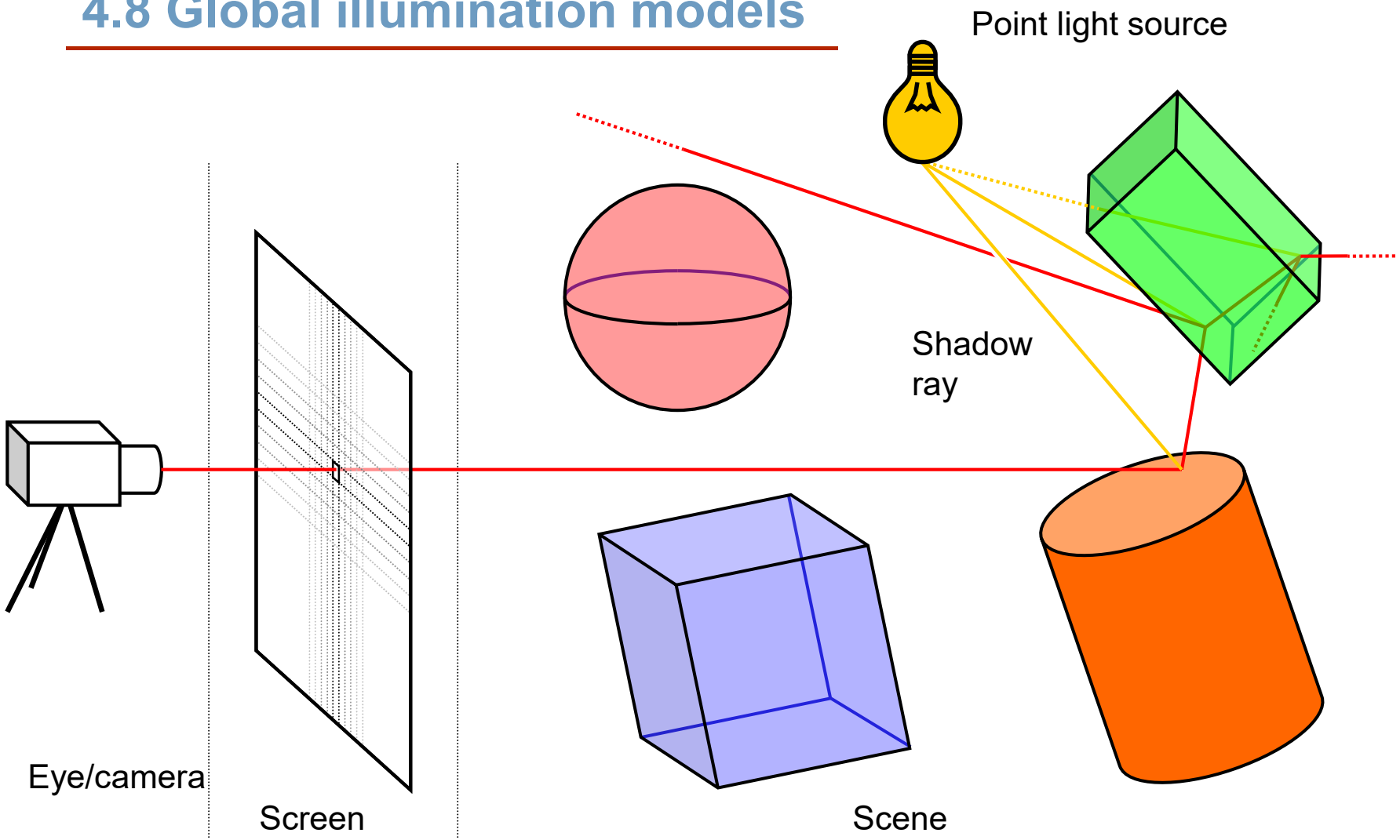


4.8 Global illumination models

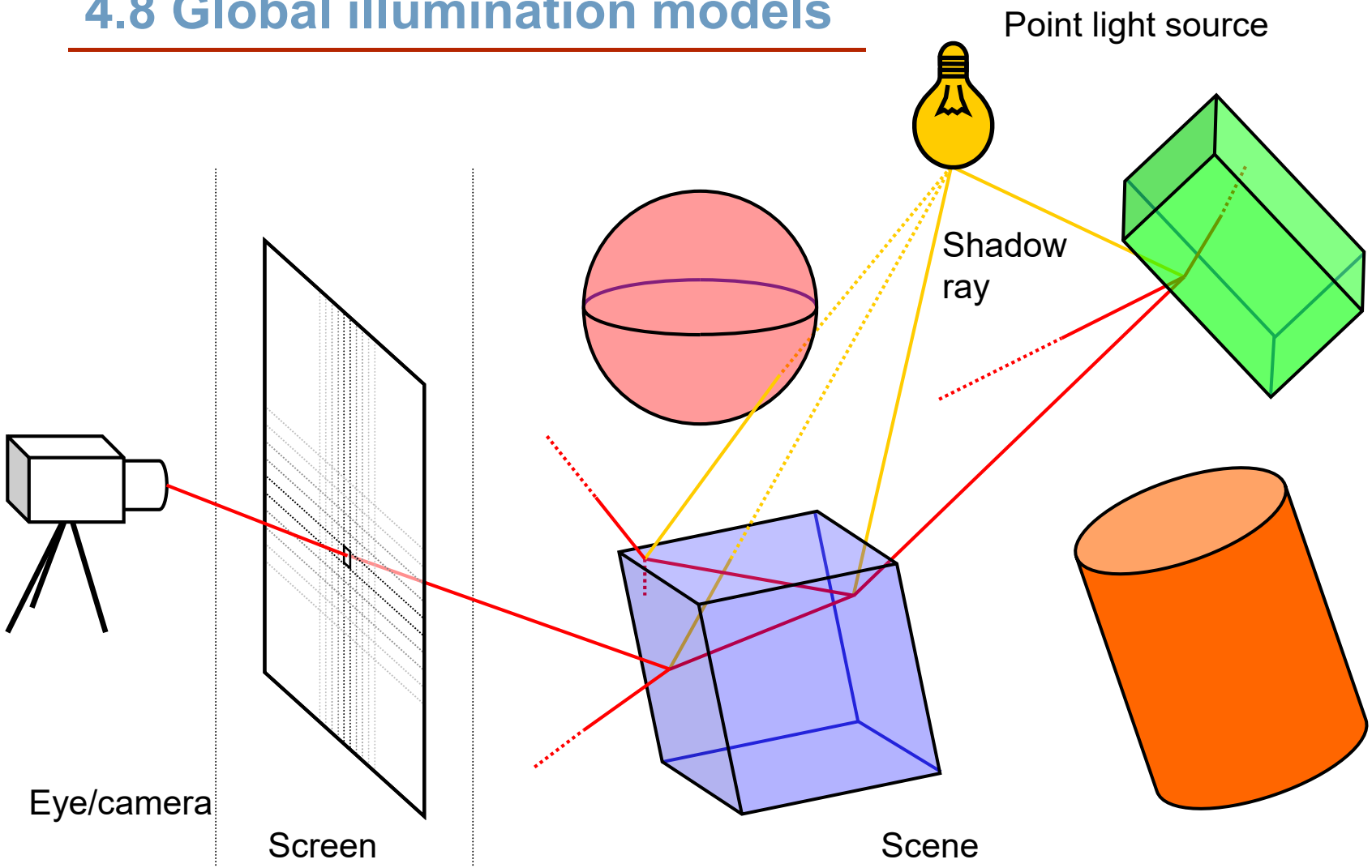
- The rays are traced from the camera through every pixel into the scene.
- For every intersection with an object
  - the reflected light components,
  - the refracted light components, and
  - the direct light components (locale illumination model)

  are determined.

4.8 Global illumination models



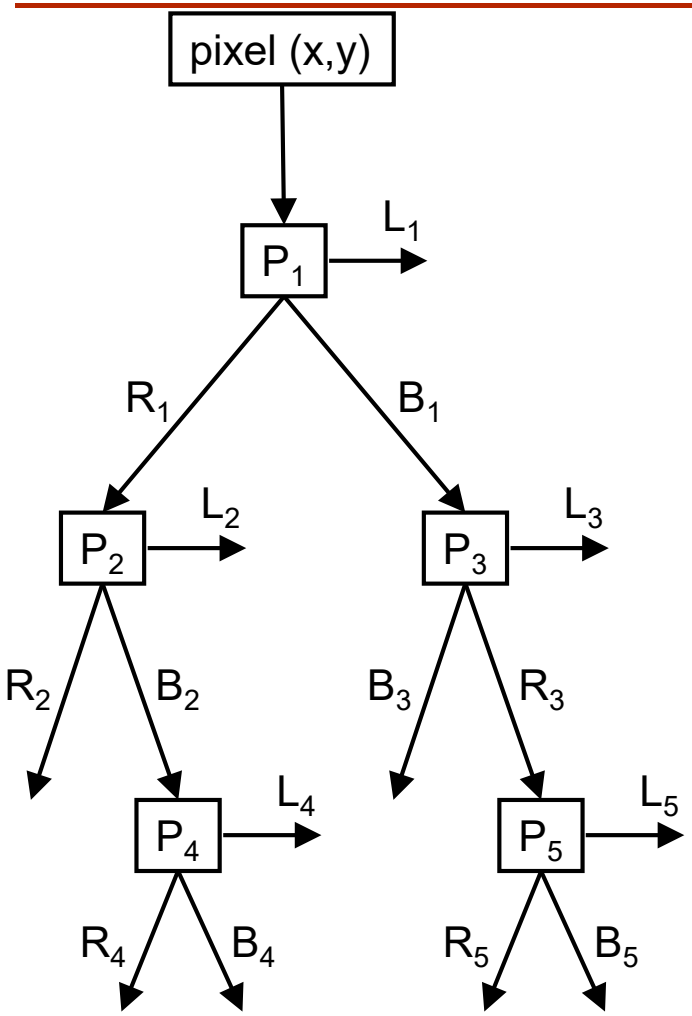
4.8 Global illumination models

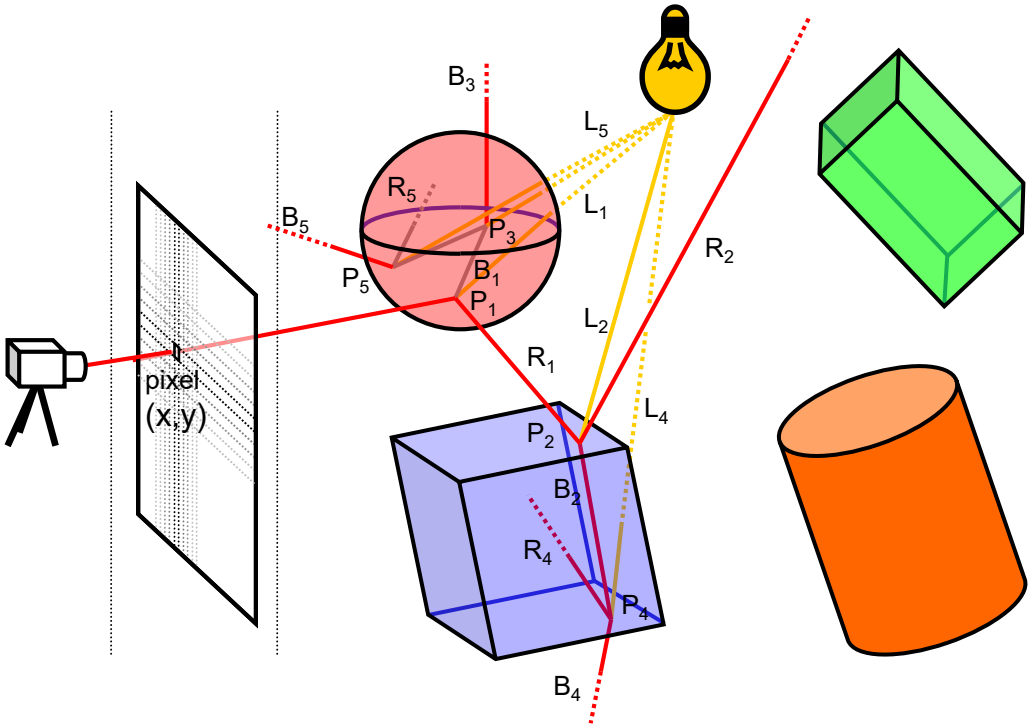


4.8 Global illumination models

- Shadow rays (light rays):
  - From every intersection trace a ray to every light source.
  - If this ray intersects an object, the respective intersection is in the shadow with respect to this light source.

4.8 Global illumination models





4.8 Global illumination models

1. Termination of the recursive ray tracing, if
   - the reflected or refracted rays do not intersect any further object,
   - the color component at the pixel color of a traced rays is too small, or
   - a predefined tree depth is reached.

2. Computation costs depend highly on the complexity and structure of the scene:
   - Only using space partition methods (e.g. octrees, bsp-trees, kd-trees) yield a sufficient runtime of ray tracing in practice.

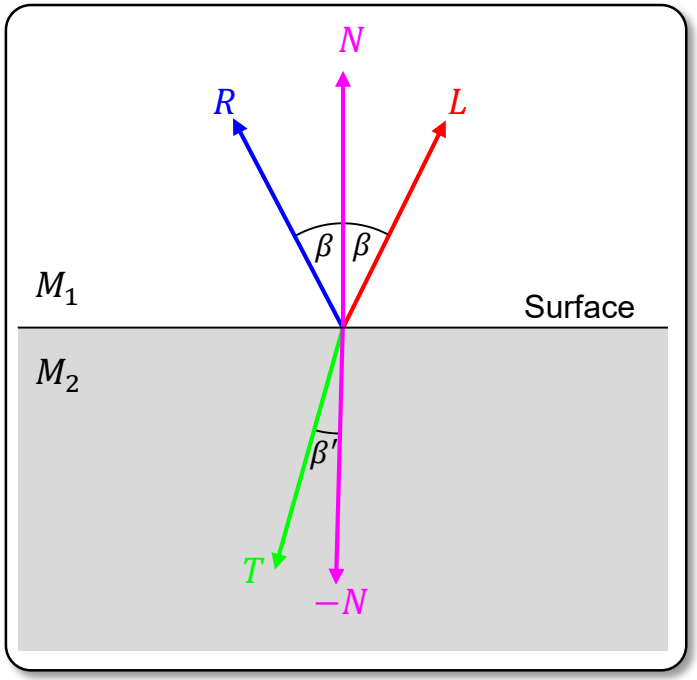
4.8 Global illumination models

Ray-Tracing: Snell's law of refraction

- Refraction happens in nature at bounding surfaces of two materials M_1 and M_2 with different densities, e.g. air and water.
- The refraction angle β' between $-N$ and T is proportional to the incidence angle β between N and R:

$$\frac{\sin\beta}{\sin\beta'} = \frac{n_2}{n_1}.$$

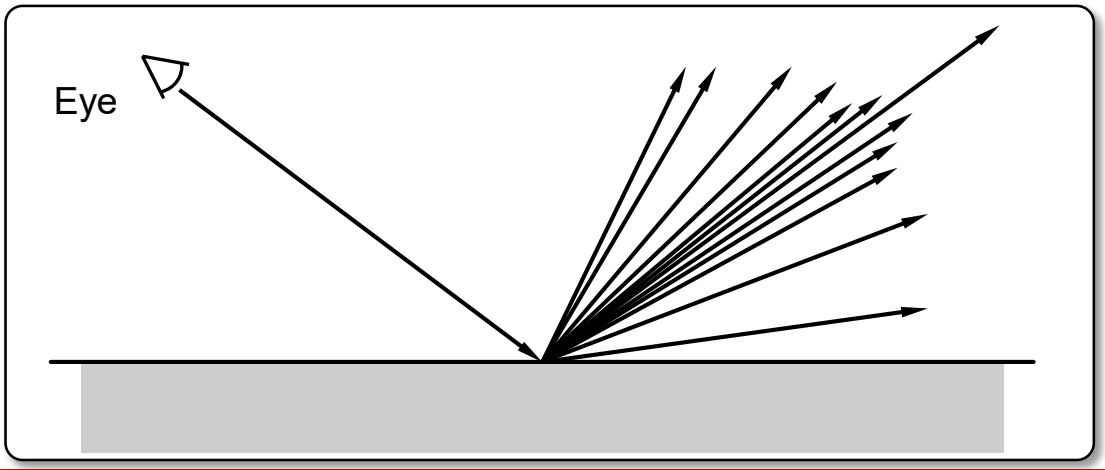
  - If the ray enters a denser material, this angle becomes smaller.
  - Surface can reflect and refract light at the same time in a certain ratio.
  - If the refraction angle β' becomes larger than $90°$, the ray is totally reflected.



4.8 Global illumination models

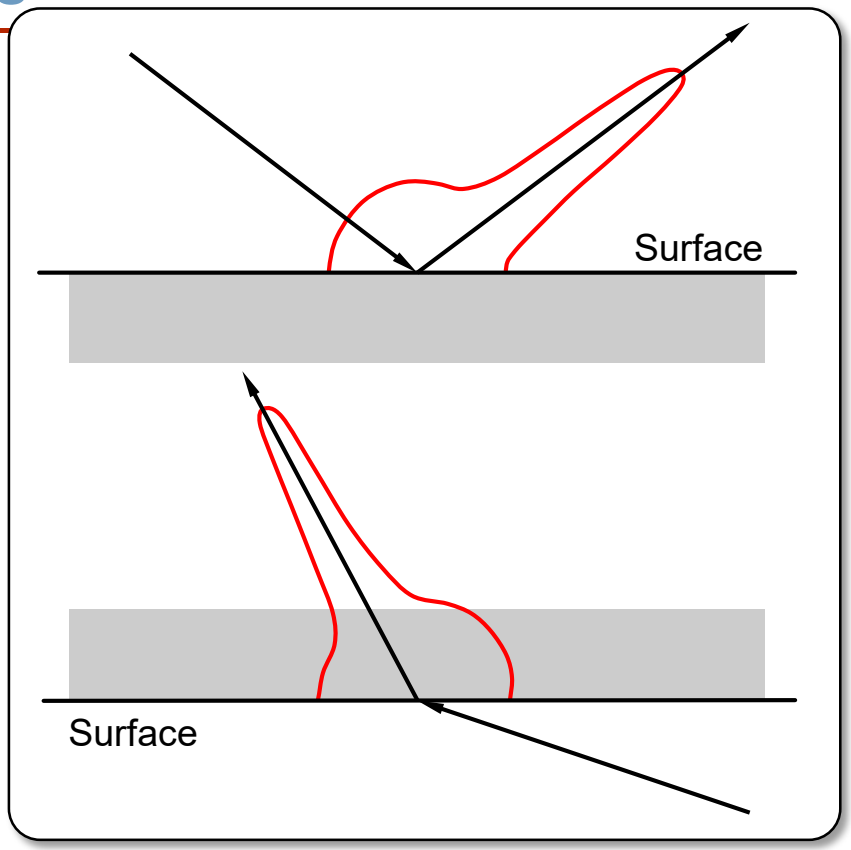
Distributed Ray-Tracing (randomly distributed)

- Usually specular reflections are imperfect, because there is no perfectly planar mirror reflecting 100% of the light.
- Distributed ray-tracing allows to generate more realistic fuzzy effects for ray-tracing.
- In stead of one reflected ray, multiple sub-rays are traced and their color values are averaged.



4.8 Global illumination models

- Many rays are reflected along the perfect reflection direction, and some diverge from this direction.
- The ray distribution is pear-shaped.
- The same holds for refracted rays

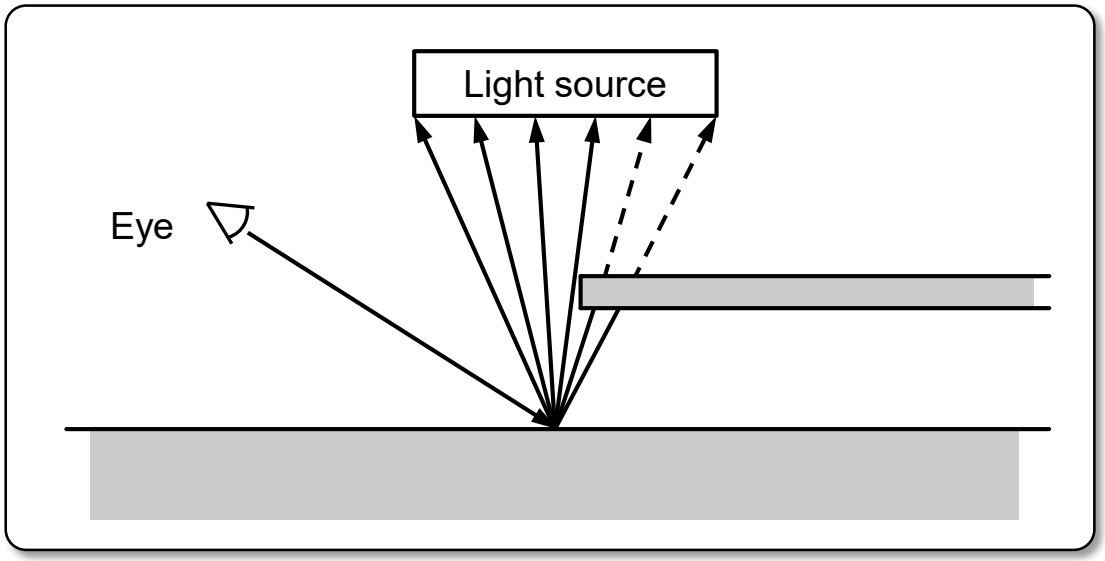


- Using a **stochastic distribution** for all possible reflection- and refraction-directions and subsequent **averaging** a more realistic simulation is achieved.

4.8 Global illumination models

Distributed ray-tracing − 2d-light sources

- Another increase in realism can be achieved using 2d-light sources, simulated e.g. by a **multitude of point light sources**.

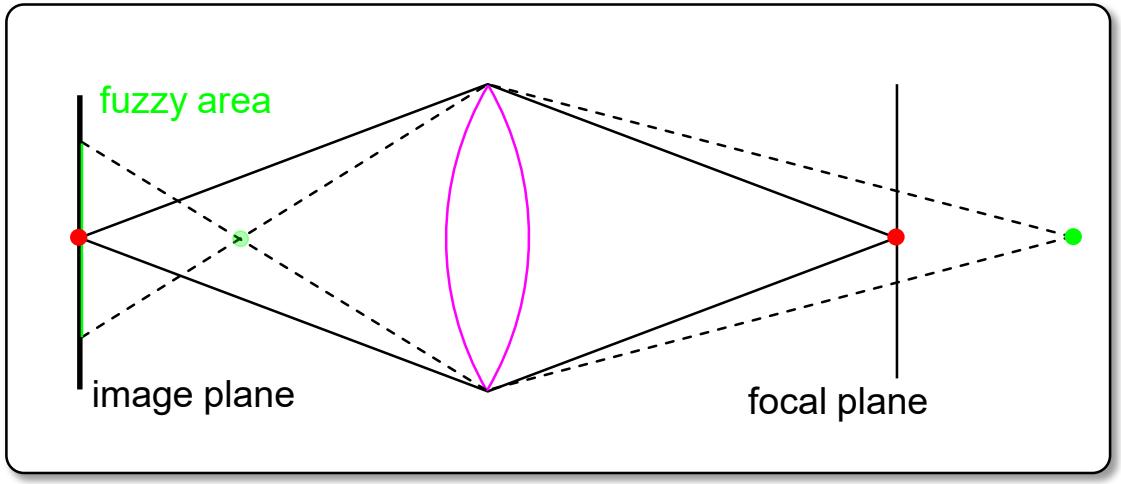


- Using a suitable stochastic distribution for the light rays, realistic half shades can be simulated.

4.8 Global illumination models

Distributed ray-tracing − apertures

- Photo realistic images can be generated **simulating the aperture** of the camera.

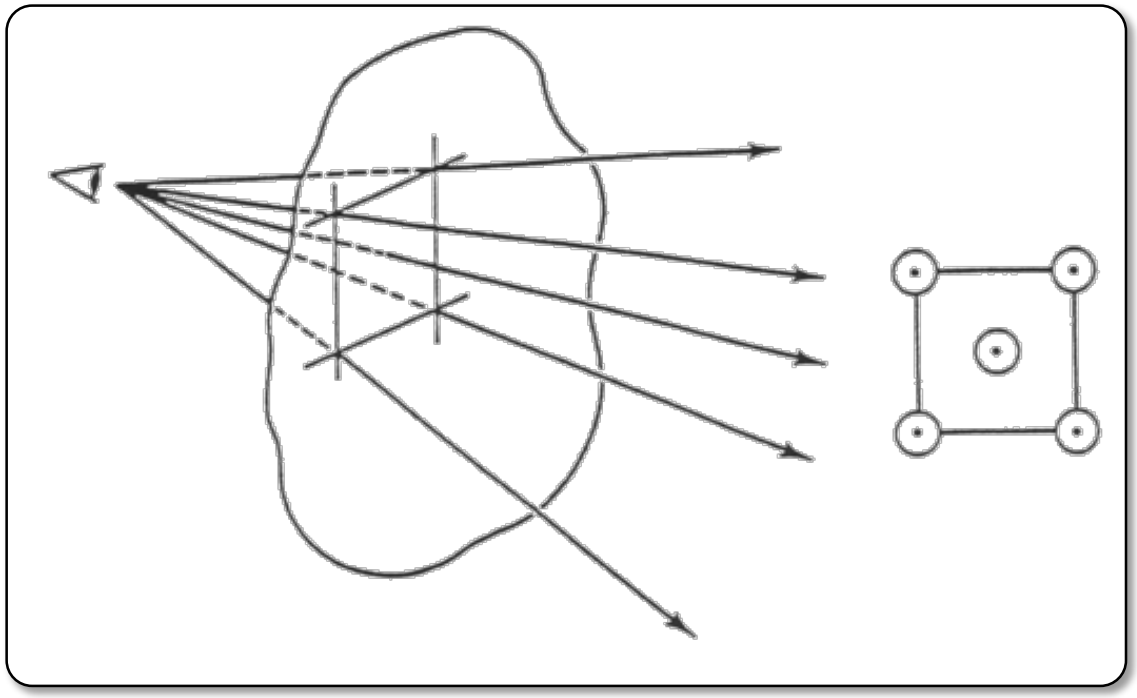


- An object outside the focal plane yields a fuzzy/blurred image.
- This is achieved by the correct computation of the lens-refraction with a stochastic distribution of rays over the lens surface.

4.8 Global illumination models

Ray-tracing − adaptive super-sampling

- To reduce aliasing-effects, multiple rays per pixels can be traced and their resulting color values are averaged.

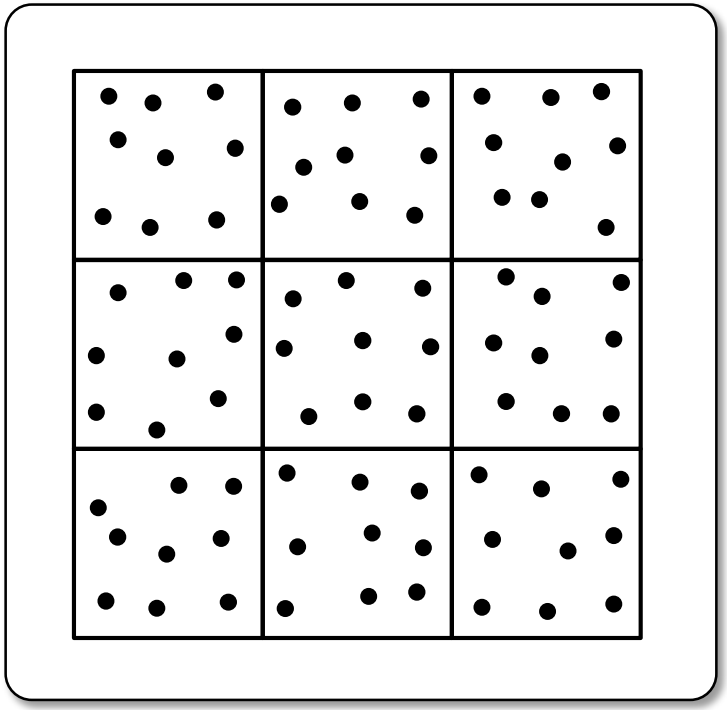


Example: Four rays through the corners and one ray through the center of a pixel.

4.8 Global illumination models

Stochastic ray-tracing

- Instead of tracing through a fixed pixel or sub-pixel grid, trace through a stochastic grid, e.g. for super-sampling:



4.8 Global illumination models

Ray-tracing – Properties

Advantages:

(1) The physical phenomena of illumination using geometric optics are very effectively simulated.

(2) Very good for specular reflections.

(3) Visibility is solved automatically.

(4) High realism for geometric optics.

Disadvantages:

(5) Not suitable for diffuse reflections.

(6) Large computation costs.

(7) Intersection computations are very inefficient.

(8) Prone to numerical problems.

4.8 Global illumination models

A. Hofmann, V. Reiberger, N. Ivlev, Computergrafik Praktikum, TU KL

4.8 Global illumination models

4.8.2 Radiosity

- The radiosity method computes the **energy transfer** caused by **diffuse radiation** between individual surface components in a scene.
- Diffuse reflections cause a highly varying light incidence of ambient (indirect) light.
  - Important e.g. in interior design.
- The physically exact relation is given by an integral equation, whose solution is approximated using finite elements.
- The emitted radiation energy (aka **radiosity**) of the surfaces in the scene is the input to this rendering approach.
- To model also directed light, radiosity can be combined with Ray-Tracing.

4.8 Global illumination models

- Considers propagation of light in terms of energy conservation and equilibrium in a closed system (aka scene).
- For every surface/object, all emitted and reflected light intensities (aka outgoing intensities) must be computed and transmitted to all other surfaces/objects in the scene.
- The computation of the incoming light intensity at a given surface point, requires
  - complete **geometric information about the relative position and pose** of all light emitting, reflecting and transmitting objects in the scene, and
  - complete **photometric characteristics** of all objects in the scene.

4.8 Global illumination models

Input: A 3d scene consists of surface segments (e.g. polygons)

$$S = \{S_i\}$$

and the emitted light energy per surface unit

$$E(x) \quad [\text{Watt m}^{-2}]$$

in every point x of S.

- Surface segments with $E(x) \neq 0$ are the light sources.

Goal: The **radiosity** function

$$B(x) \quad [\text{Watt m}^{-2}],$$

which determines the (diffuse) reflected power per surfaces for all segments of the scene.

- This is used to determine the color for the rendering of the scene.

4.8 Global illumination models

Radiosity equation

$$B(x) = E(x) + \rho(x) \int_S F(x,y) B(y) dy$$

$\rho(x) \in [0,1]$, reflectivity of the surface in x

with

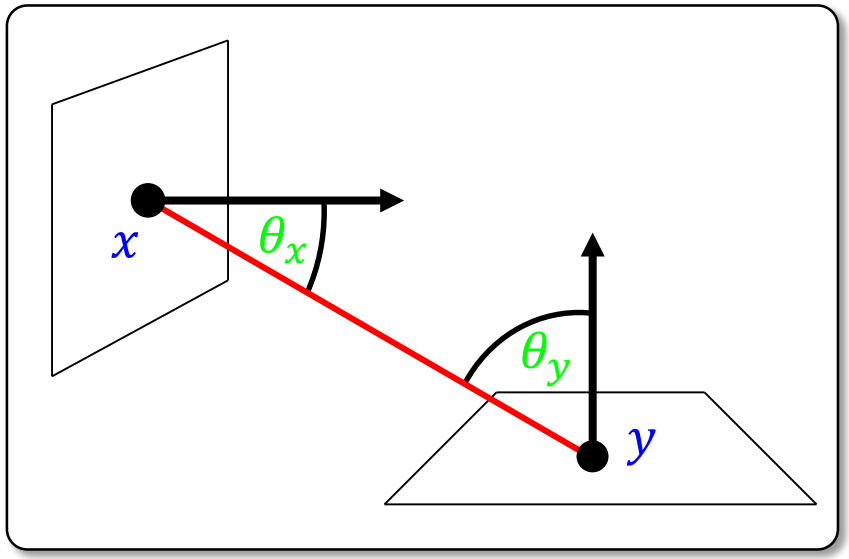
$$F(x,y) = V(x,y) \frac{\cos\theta_x \cdot \cos\theta_y}{\pi \cdot \|x-y\|^2}$$

visibility: 0 or 1

normalization

damping

incidence and emergent angle



4.8 Global illumination models

- Discretizing the radiosity equation e.g. using a spline-representation for the functions $E(x)$ and $B(x)$ using the spline-functions $\{\varphi_i\}$

$$E(x) = \sum_i e_i \varphi_i(x) \quad \text{and} \quad B(x) = \sum_i b_i \varphi_i(x)$$

yields for the integral equation

$$B(x) = E(x) + \rho(x) \int_S F(x,y) \cdot B(y) dy$$

a linear system of equations

$$b_i = e_i + \rho_i \sum_{j=1}^{n} f_{ij} b_j \,.$$

4.8 Global illumination models

- The coefficients e_i of the emission $E(x)$ and the reflectivity-coefficients ρ_i of the surface patches S_i are known.
- The ρ_i are material properties, that determine the ratio of the perpendicular light that is reflected.

- The **form factors** f_{ij} determine the ratio of energy of S_j transported to S_i and depend only on the geometry of the scene.
- The form factors are computed via numerical integration, depending on the size, orientation and mutual visibility of the surface patches S_j and S_i.

4.8 Global illumination models

- The form factor between two differential areas dA_j and dA_i of the surface patches S_j and S_i is given by

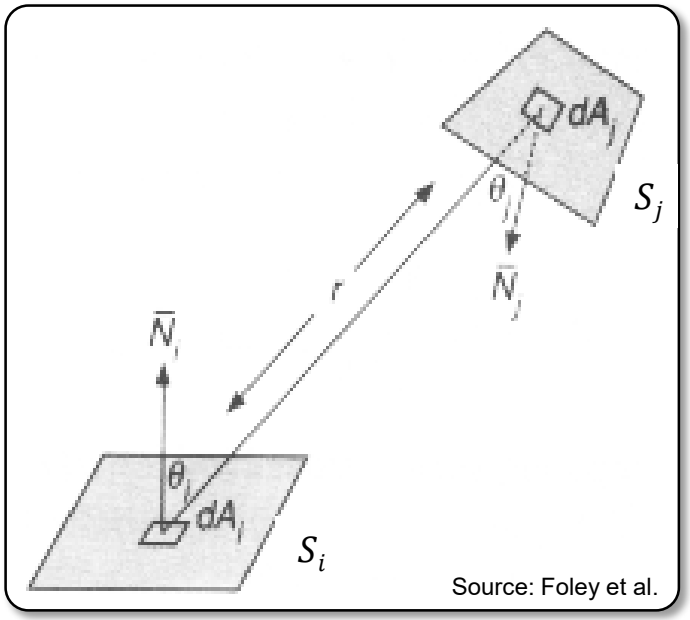
$$dF_{ij} = v_{ij} \frac{\cos\theta_i \cos\theta_j}{\pi \cdot r^2} dA_j$$

$$v_{ij} = \begin{cases} 1, & \text{if } dA_j \text{ is visible from } dA_i \text{ and } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

- The form factor from S_j to S_i is

$$f_{ij} = \frac{1}{|S_i|} \int_{S_i} \int_{S_j} v_{ij} \frac{\cos\theta_i \cos\theta_j}{\pi \cdot r^2} dA_j \, dA_i$$

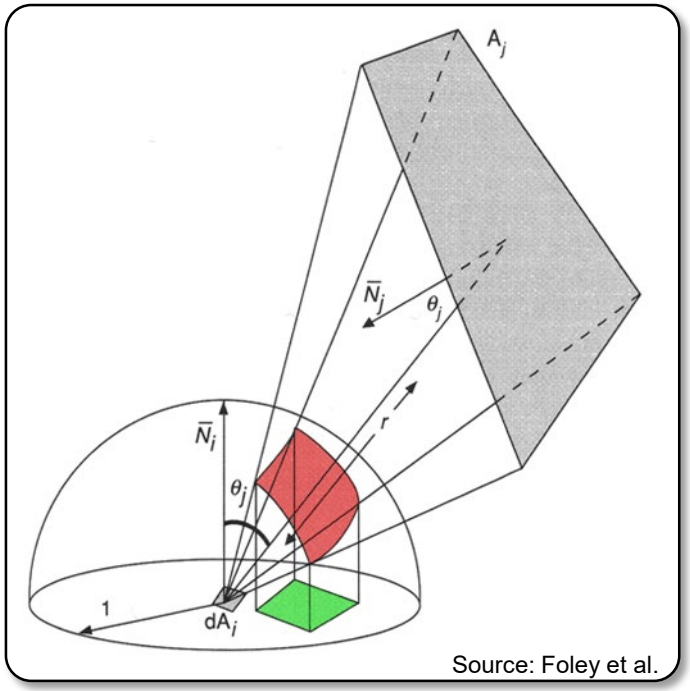
- Computation using numerical integration.



Source: Foley et al.

4.8 Global illumination models

- **Simplifying assumption:** The midpoint of a patches S_i is typical for the complete patch.
  - f_{ij} can be approximated by $\tilde{f}_{ij} = \int_{S_j} v_{ij} \frac{\cos\theta_i \cos\theta_j}{\pi \cdot r^2} dA_j$.
- Computation of $\tilde{f}_{ij}$ [Nusselt ´81]:
  1. Projection of those parts of S_j onto unit sphere at dA_i, which are visible from dA_i.
     - $\widehat{=} \cos(\theta_j)/r^2$
  2. Orthogonal projection onto unit circle.
     - $\widehat{=} \cos(\theta_i)$
  3. Division by circle area π.

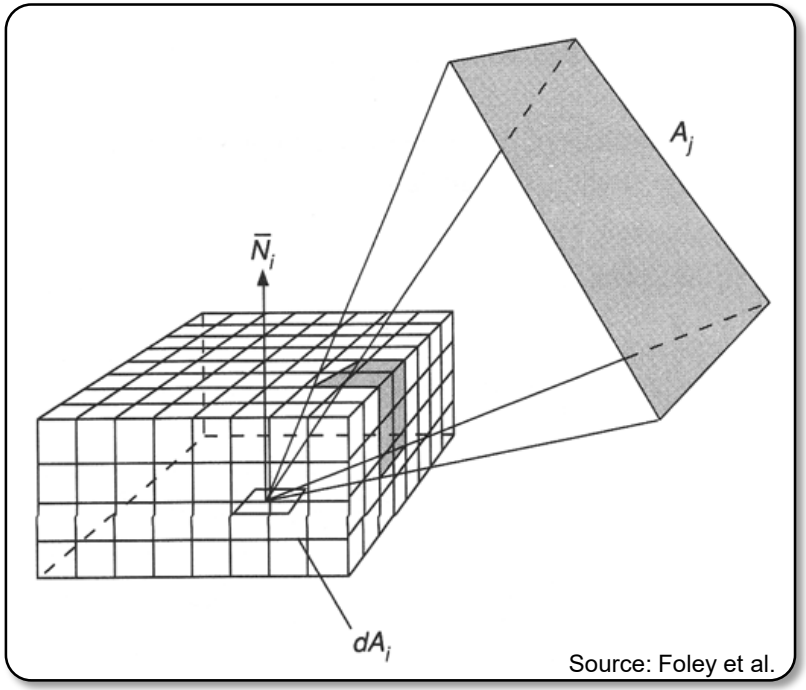


Source: Foley et al.

4.8 Global illumination models

- **Further simplification:** Projection onto unit cube centered at dA_i with top face parallel to dA_i [Cohen ´85]:

1. Subdivide unit cube into cells.
2. Projection onto the faces of the cube.
3. Each cell has a pre-computed form factor.
4. Summation of form factors of those sub-cells that are covered by the projected surface.



Source: Foley et al.

4.8 Global illumination models

- Finally this yields the system of linear equations

$$b_i - \rho_i \sum_j f_{ij} b_j = e_i \,.$$

- …or in matrix form

$$\begin{bmatrix} 1-\rho_1 f_{11} & -\rho_1 f_{12} & \cdots & -\rho_1 f_{1n} \\ -\rho_2 f_{21} & 1-\rho_2 f_{22} & \cdots & -\rho_2 f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n f_{n1} & -\rho_n f_{n2} & \cdots & 1-\rho_n f_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}.$$

- This system is solved for each frequency of light (e.g. RGB).

4.8 Global illumination models

Radiosity: Representation of a scene

1. Computation of radiosity value b_i for all patches S_i.
2. Mapping of scene and determination of its visible parts.
3. Computation of color for each pixel.

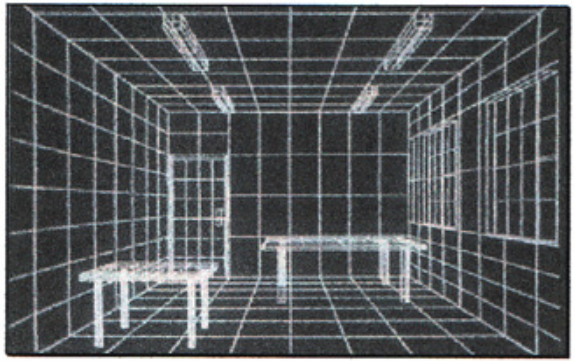
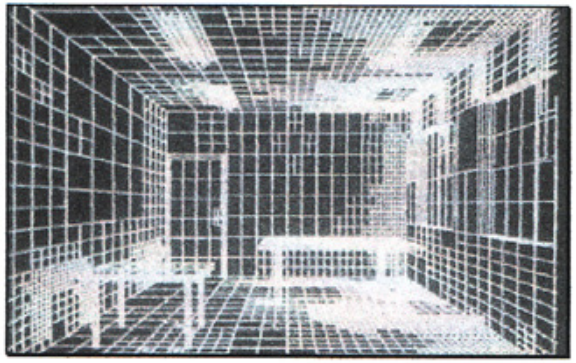
Remarks:

- For step 1. the form factors f_{ij} have to be determined, before the system of linear equations can be solved.
- For different perspectives only steps 2. and 3. have to be re-computed.
- Step 3. can be accelerated using linear interpolation along a scan line.

4.8 Global illumination models

Radiosity: Subdivision of the scene

- The larger the resolution of the scene (n patches), the better the quality of the resulting illumination.
- Number of form factors grow with $O(n^2)$ and the size of the system of equations with $O(n)$.
- Discretization if only exact for constant radiosity per patch.
- Subdivision and multi-scale-methods for critical regions.


Content

4.9 Rendering-Pipeline

The computer graphics-pipeline / rendering-pipeline

- The process of image synthesis,
  - i.e. mapping of the geometric model, the object(s), or the scene to an image on the display (output device),

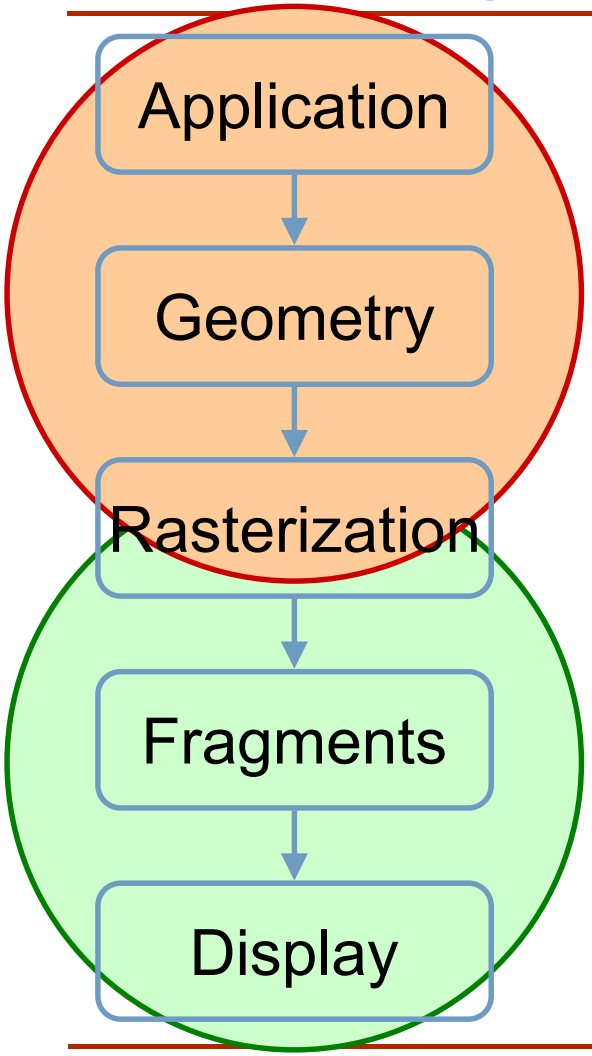
  is called ***rendering***.
- A concrete implementation of this process in soft- and/or hardware is called the ***rendering-pipeline***.
  - The individual stages of this pipeline are realized by the basic algorithms of computer graphics.
  - The individual stages can be implemented in soft- and/or hardware!
  - The structure of the rendering-pipeline can vary drastically depending on the type and realization of the rendering.

4.9 Rendering-Pipeline

Rendering Pipeline

- Subdivide the rendering into simple standard stages.
- Dependent on
  - Hardware of output device (screen, graphic card, etc.),
  - Algorithm for image synthesis (illumination, shading, etc.), etc.
- Some stages can be missing in a concrete realization or occur in a different order.
- De-facto standard: OpenGL Rendering Pipeline.

4.9 Rendering-Pipeline



Manipulation of geometric objects and primitives (vertex).

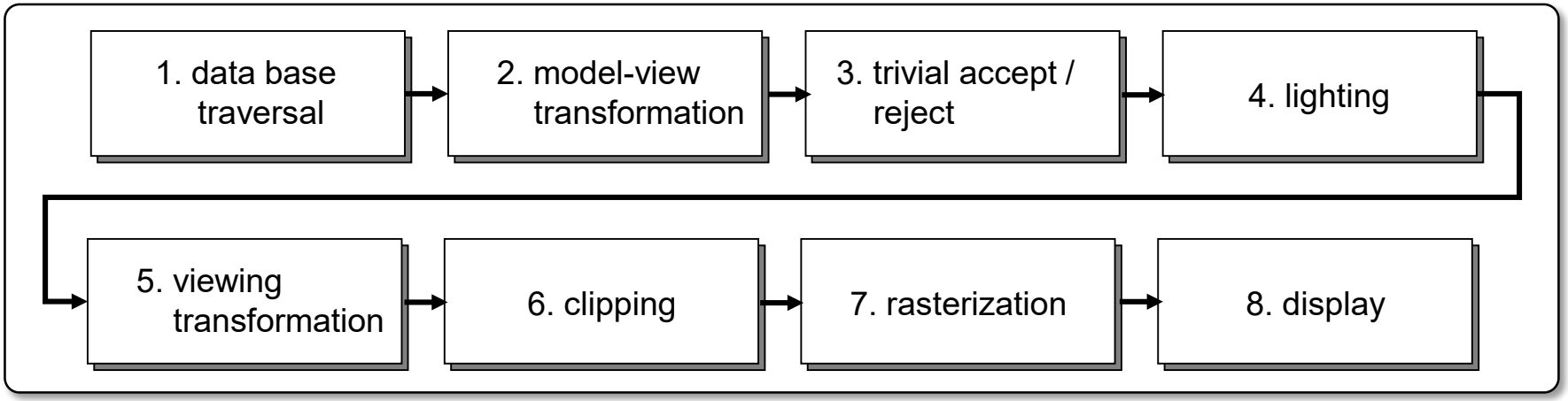
Manipulation of images and image points (fragment/pixel).

4.9 Rendering-Pipeline

Example: Local illumination with Gouraud shading with Z-buffer

- Shading in world coordinates for correct angles to the light sources.

1. See page §1/11
2. Model- & camera transformations (OpenGL Modelview-Matrix)
3. Back-face culling
4. Phong illumination
5. Transformation to view-coordinates (with additional z-value)
6. Clipping
7. Z-buffer and Gouraud shading
8. Display

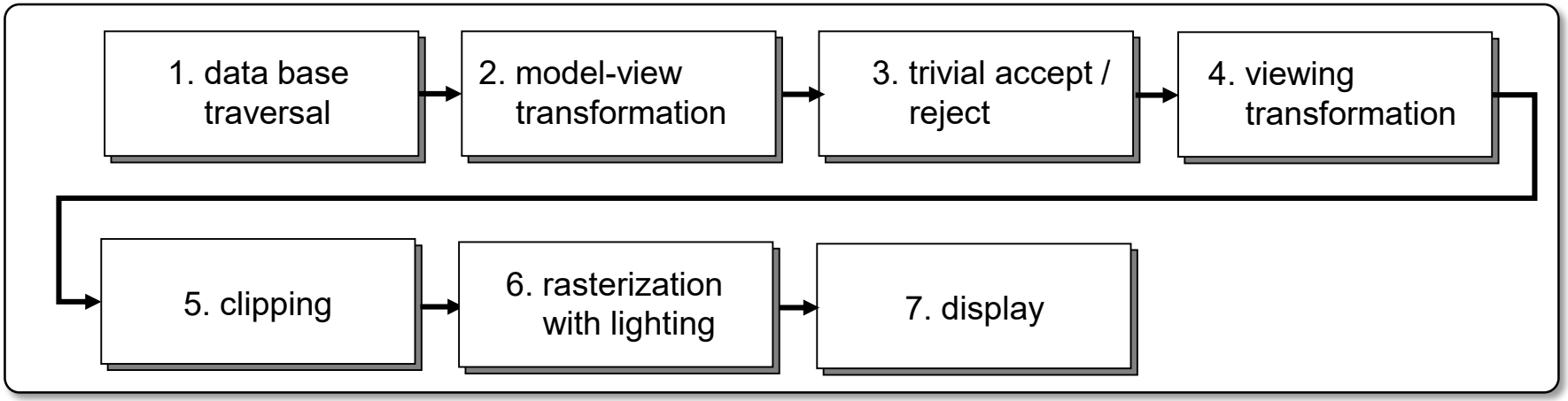


4.9 Rendering-Pipeline

Example: Local illumination with Phong shading and Z-buffer

- Shading after Z-buffer, because normals are interpolated.

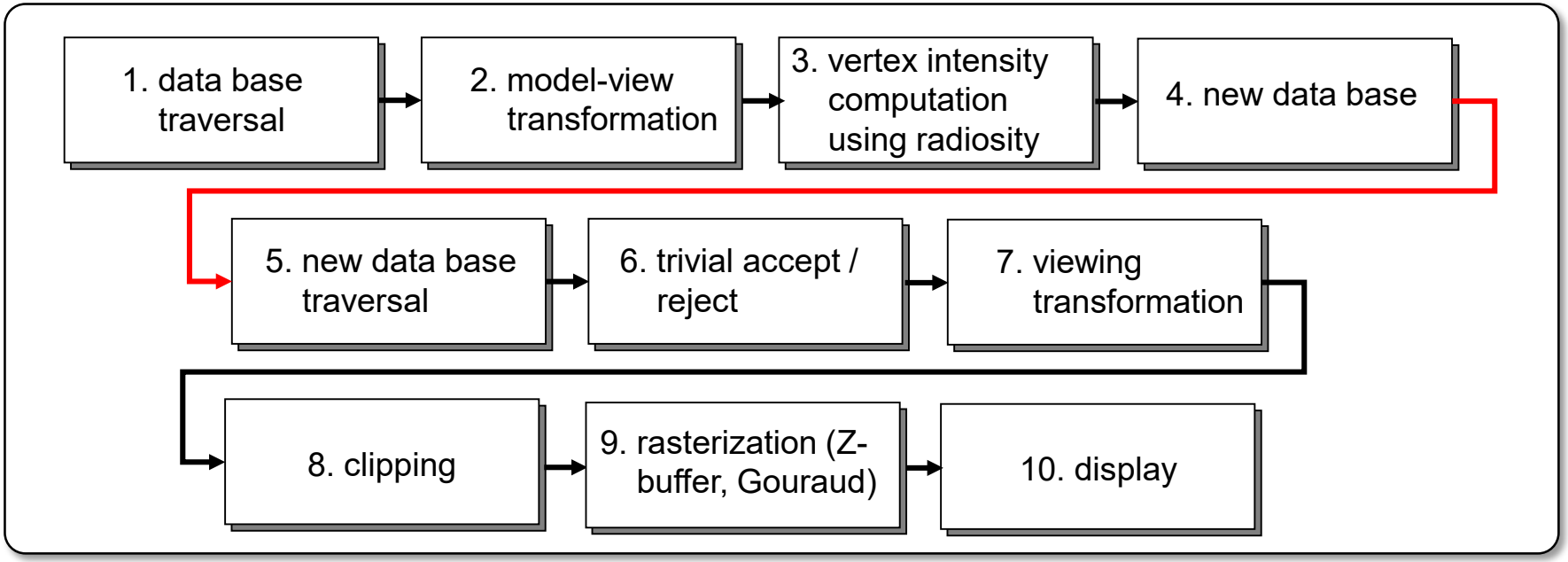
1. See page §1/11
2. See page §5-150
3. Back-face culling
4. Transformation to view-coordinates (with additional z-value)
5. Clipping
6. Z-buffer and Phong illumination/shading in world coordinates (back-mapping)
7. Display



4.9 Rendering-Pipeline

Example: Radiosity with Gouraud shading and Z-buffer

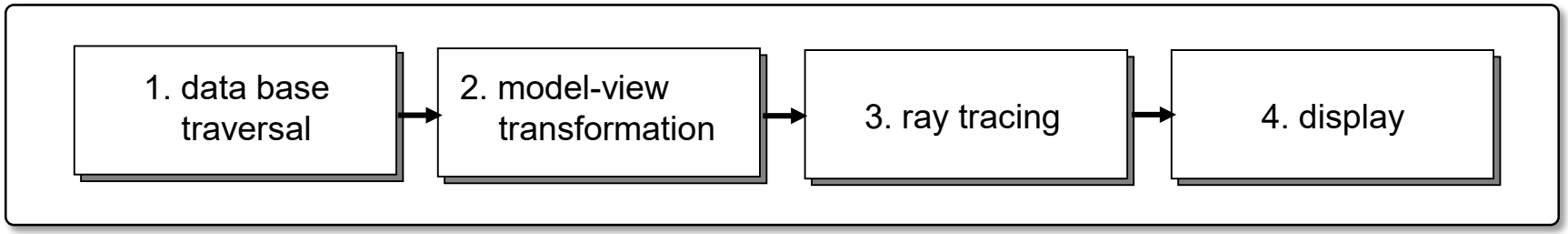
- Stages 1. – 5. in software, stages 6. – 9. in hardware.
- Global illumination: Intensities are independent on the camera position.
- No „lighting“-stage as in example at page §5-150.



4.9 Rendering-Pipeline

Example: Ray-Tracing

- All stages in software.
- The ray-trace performs all tasks of the last stages of the rendering-pipeline.



Attention: **In all these examples texture is missing!**

- **Chapter 5!**

Goals

- What is luminance?
- What is the CIE-color space?
- What is the RGB-, CMY-, HSV-color model?
- What is back-face culling?
- How do the Z- and α-Buffer-algorithm work?
- What is ray casting and how can it be accelerated?
- What is the difference between illumination and shading?
- What is the difference between local and global illumination models?
- What is the Phong illumination model?
- How do Gouraud and Phong shading work?
- What are the pros and cons of flat, Gouraud, and Phong shading?
- Describe ray tracing?
- What are the radiosity-equations (and how can they be solved)?
- What are the components of a rendering-pipeline?