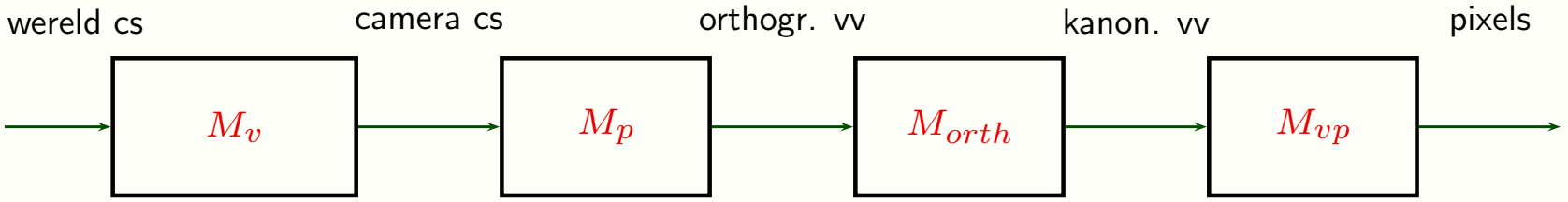
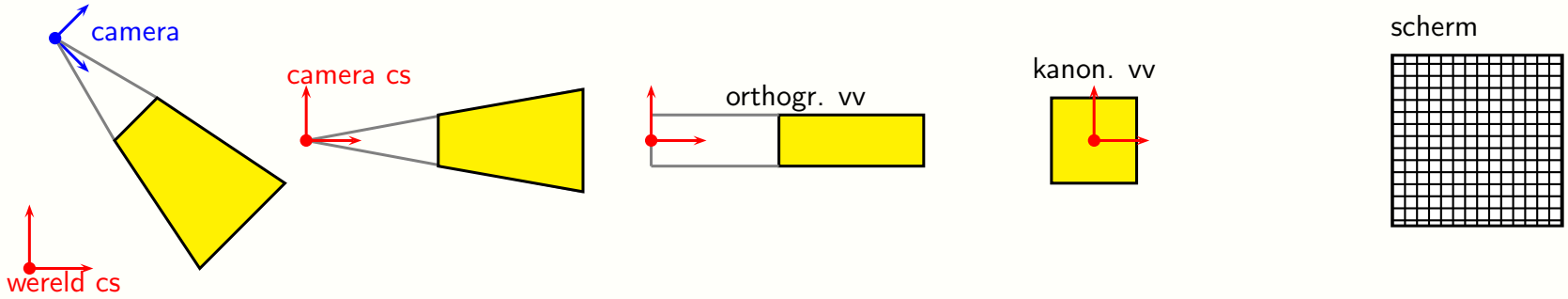
Grafische pijplijn





$$\begin{pmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

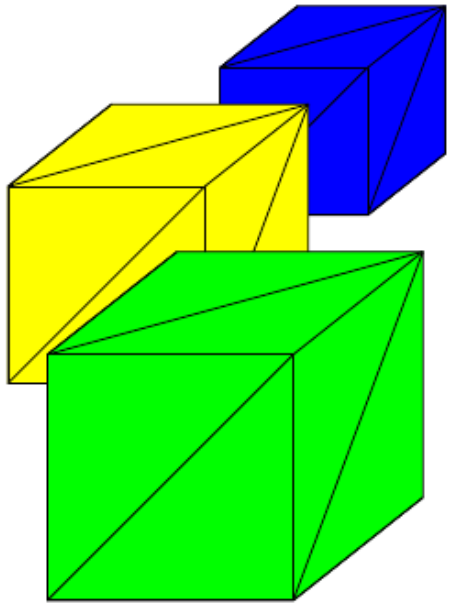
$$\begin{pmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

z-buffer, belichtingsmodellen en ray tracing

- z-buffer
- **Diffuse** reflectie en omgevingslicht
- Flat shading en Gouraud shading
- **Spiegel** reflectie
- Phong shading
- Ray tracing en recursieve ray tracing

z-buffer of diepte-buffer

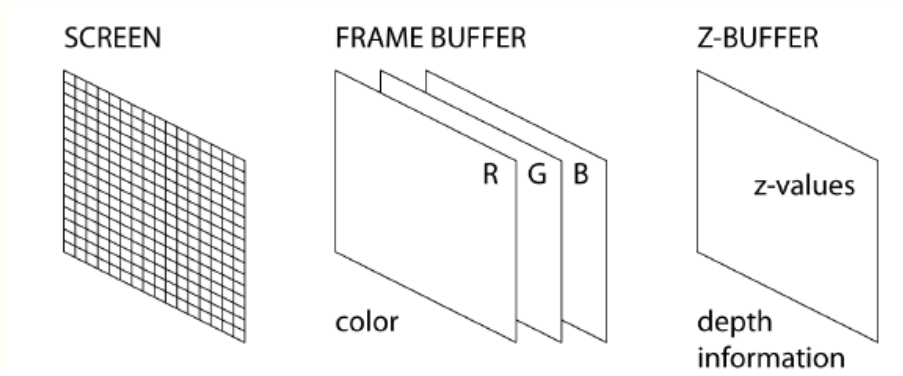
z-buffer vaak gebruikt voor hidden surface removal



Extra opslag van diepte informatie

z-buffer algoritme: basis idee

Naast frame buffer (die pixels van beeld bevat) ook een z-buffer van dezelfde grootte, om diepte informatie bij te houden



z-buffer algoritme

- z-buffer algoritme:

  1. zet achtergrond kleur in rgb-buffer en diepste z-waarde in z-buffer
  2. voor ieder polygoon: rasterizeer
  3. voor elk pixel:
     - bepaal of z-waarde (diepte) kleiner is dan opgeslagen z-waarde
     - als dat zo is, zet nieuwe kleur in rgb-buffer en nieuwe z-waarde in z-buffer

- z-waarden voor hoekpunten berekend, voor overige pixels geïnterpoleerd

z-buffer pros en contras

1. veel geheugen nodig om z-waarden te bewaren
2. kan in hardware geïmplementeerd worden
3. geen limiet aan aantal polygonen
4. implementatie heel simpel
5. kan geen transparante oppervlakken aan

z-buffer versus ray tracing

z-buffer

```
for each object
    for each pixel inside objects projection
        if depth is closer than z-buffer value
            update color buffer with objects shading
            update z-buffer
```

ray casting

```
for each pixel
    for each object
        if ray through pixel intersects object
        and is closer than saved depth
            compute shading
            update color
```

Belichtingsmodellen

Belichtingsmodel berekent lichteffecten van oppervlak gebruikmakend van **optische** eigenschappen van oppervlak, zoals:
- transparantie
- kleurweerspiegelingscoëfficiënten
- textuur-parameters

en is afhankelijk van **geometrie**:
- richting van lichtbron
- richting van oog waarnemer
- **normaal** van oppervlak

Belichtingsmodel berekent hoeveel licht verspreid of weerspiegeld wordt en ons oog bereikt vanaf elk deel van oppervlak

Lichtbronnen en oppervlak

Twee typen **lichtbronnen**:
- puntlichtbronnen
- omgevingslicht (ambient)

Licht schijnt op **oppervlak**
- deel geabsorbeerd (mat)
- deel weerspiegeld (glimmend)
- deel doorgelaten (transparant)

Deel licht dat weerspiegeld wordt en oog bereikt is wat we zien

Weerspiegelde licht

Licht van lichtbron kan weerspiegeld worden:

- **diffuse reflectie**: uniform verspreid in alle richtingen
- **spiegelreflectie**: vooral in één richting verspreid

Meeste oppervlakken combinatie van 2 typen van reflecties:
totale weerspiegelde licht is som van diffuse component en spiegelcomponent

Voor elk punt op oppervlak berekenen we de grootte van elke component die ons oog bereikt

1. Diffuse of Lambertian reflectie

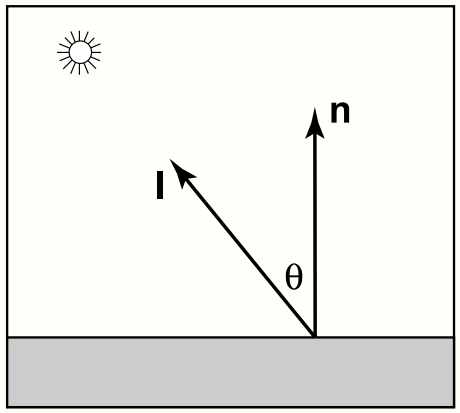
- licht van puntlichtbron valt op deel oppervlak
- deel van licht wordt weerspiegeld in alle richtingen **evenveel**
- deel daarvan bereikt ons oog met **kleur** c

Mat oppervlak ziet er vanaf alle richtingen even helder uit, omdat licht in alle richtingen evenveel gereflecteerd wordt

1. Diffuse of Lambertian reflectie

Lambertian cosine law:
kleur c van oppervlak is evenredig met hoek tussen **normaal** oppervlak $\mathbf{n}$ en **richting** lichtbron $\mathbf{l}$ (directional light)

$$c \sim \cos\theta \qquad \text{of} \qquad c \sim \mathbf{n}.\mathbf{l}$$



1. Diffuse of Lambertian reflectie

Voor **Lambertian oppervlak** is hoeveelheid licht gezien door viewer:
- onafhankelijk van view
- evenredig met $\cos\theta$

Diffuse belichtingsvergelijking:

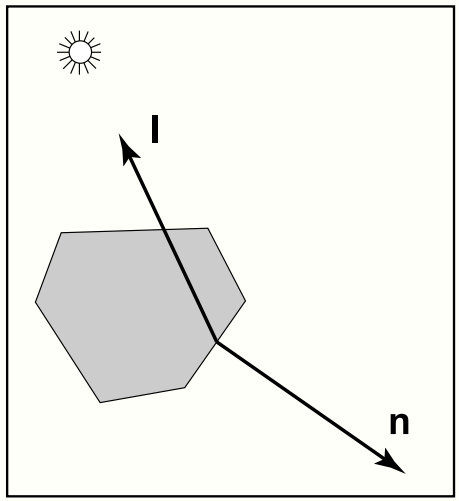
$$c = c_r c_l \mathbf{n}.\mathbf{l}$$

c is waargenomen kleur
c_r is diffuse-reflectie-coëfficiënt van oppervlak (tussen 0 en 1)
c_l is intensiteit van puntlichtbron (constant voor alle voorwerpen)
θ is hoek van inval (tussen $0°$ en $90°$)

Kleur c tussen 0 en 1

$$c = c_r c_l \mathbf{n}.\mathbf{l}$$

Wat als $\mathbf{n}.\mathbf{l}$ **negatief**?



Als **normaalvector** in punt in andere richting, dan $\mathbf{n}.\mathbf{l}$ negatief

In dit geval: $c = c_r c_l \max(0, \mathbf{n}.\mathbf{l})$

2. Omgevingslicht of ambient light

Probleem: punt met normaal weg van lichtbron is zwart

- diffuse reflectie: **zwarte schaduwen** die onrealistisch en hard zijn

Oplossing: voeg constante kleur toe aan elk object

- schaduwen verzachten met **omgevingslicht** of ambient light
- achtergrondlichtbron die in alle richtingen uniform licht verspreidt

c_a is intensiteit achtergrondlichtbron (constant voor alle voorwerpen)

2. Omgevingslicht of ambient light

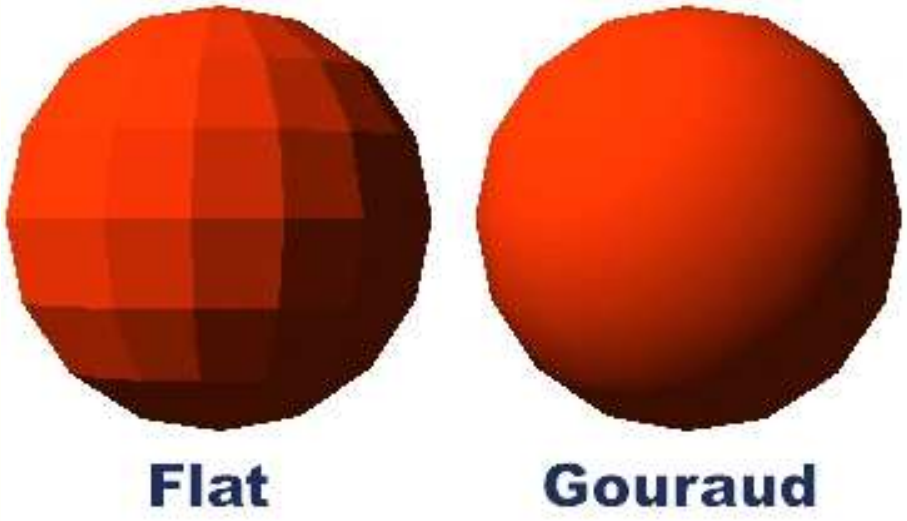
Ambiente component c_a toevoegen aan belichtingsvergelijking:

$$c = c_r(c_a + c_l \max(0, \mathbf{n}.\mathbf{l}))$$

c de resulterende intensiteit ten gevolge van ambiente en diffuse component

c_a door experimenten bepaald,
correspondeert **niet** met fysische eigenschap

Flat shading en Gouraud shading



Flat shading

Berekenen belichting van polygoon-oppervlakken

`Stap 1`:
bepaal **normaalvector** van elk polygoon

`Stap 2`:
bereken **kleur** van hele polygoon met belichtingsmodel

Gouraud shading

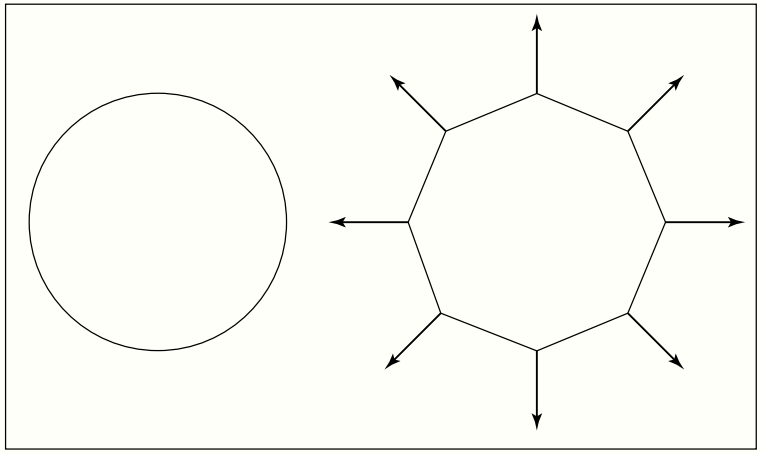
Berekenen belichting van polygoon-oppervlakken

`Stap 1`:
bepaal **normaalvector** in elk hoekpunt polygoon

`Stap 2`:
bereken **kleur** van elk hoekpunt met behulp van gevonden hoekpuntnormaalvectoren met belichtingsmodel

`Stap 3`:
interpoleer **hoekpuntkleuren** over hele gebied polygoon

Normaalvectoren bij Gouraud shading



3. Spiegelreflectie

Voorwerpen verspreiden licht **niet** uniform in alle richtingen, daarom spiegelcomponent toevoegen

Spiegelreflecties veroorzaken glansplekken (glimmende voorwerpen)

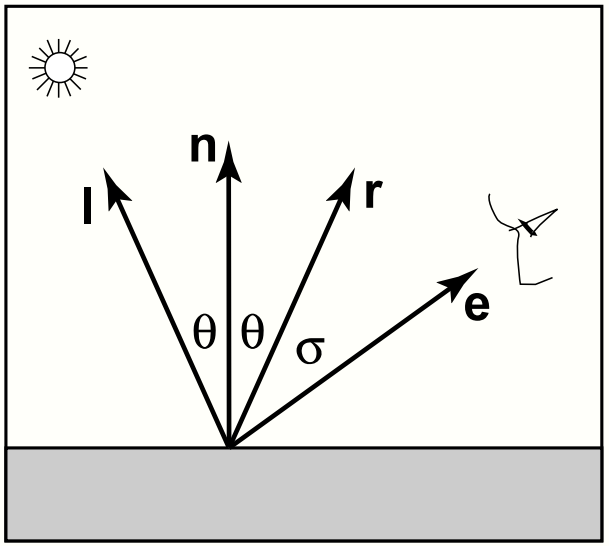
Verlicht appel met helder wit licht:
- glansplek door spiegelreflectie
- rest door diffuse reflectie

Op glansplek lijkt appel niet rood, maar wit (Kleur invallende licht)

Model voor spiegelreflectie: **Phong model**

3. Spiegelreflectie: Phong model

Glimmende voorwerpen weerspiegelen licht vooral in **één** richting $\mathbf{r}$



Als viewer $(\mathbf{e})$ in de buurt van teruggekaatste richting (σ is klein), dan **highlight** te zien

Phong model

Gegeven terugkaatsrichting $\mathbf{r}$, welke functie levert grote waarde bij $\mathbf{r}$ en kleine waarden verder daarvandaan?

$$c = c_l(\mathbf{e}.\mathbf{r})$$

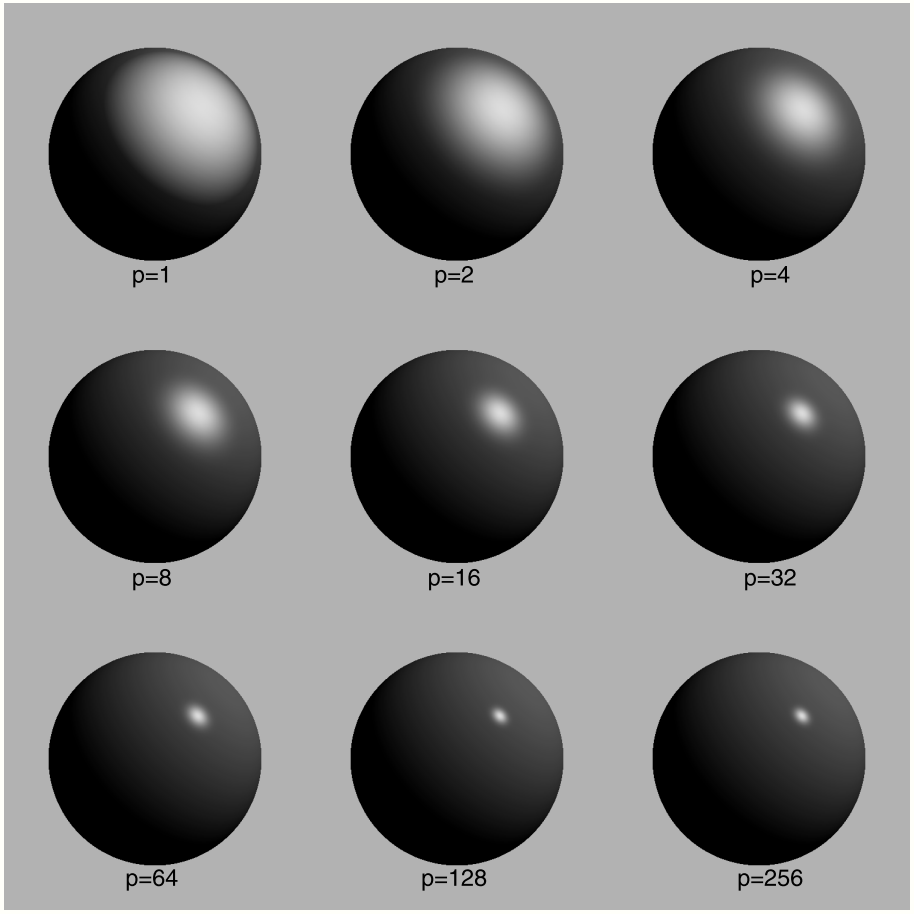
Twee problemen: inproduct kan negatief zijn en highlight te breed

Oplossing:

$$c = c_l \max(0, \mathbf{e}.\mathbf{r})^p$$

met p **Phong exponent**

Effect Phong exponent



Phong belichtingsmodel

Kleur c van oppervlak berekend met:

$$c = c_r(c_a + c_l \max(0, \mathbf{n}.\mathbf{l})) + c_l \max(0, \mathbf{e}.\mathbf{r})^p$$

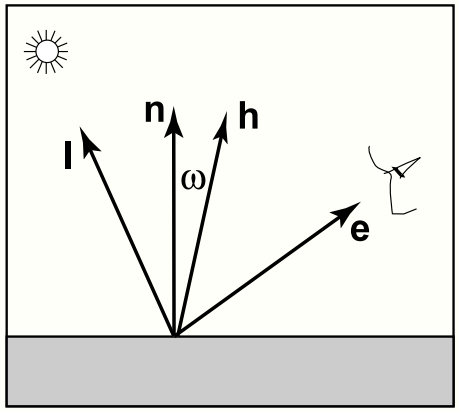
Het heet **Phong's** belichtingsmodel, omdat de spiegelcomponent van Phong

Berekenen van $\mathbf{r}$ en $\mathbf{e}.\mathbf{r}$ rekenintensief, vervangen door $\mathbf{h}.\mathbf{n}$ (altijd positief)
Halfway vector $\mathbf{h}$

$$\mathbf{h} = \frac{\mathbf{e} + \mathbf{l}}{|\mathbf{e} + \mathbf{l}|}$$

Halfway vector $\mathbf{h}$ tussen $\mathbf{l}$ en $\mathbf{e}$

Highlight als $\mathbf{h}$ in de buurt van $\mathbf{n}$, dus als $\cos\omega = \mathbf{h}.\mathbf{n}$ dichtbij 1



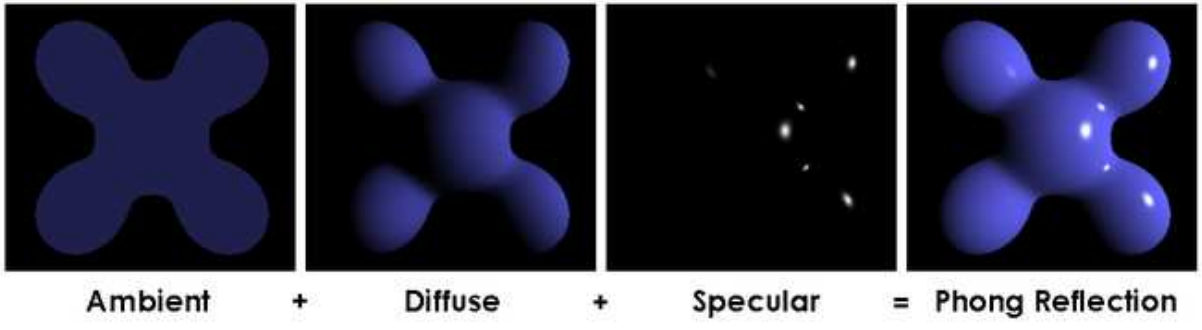
Kleur c van oppervlak berekend met:

$$c = c_r(c_a + c_l \max(0, \mathbf{n}.\mathbf{l})) + c_l \max(0, \mathbf{e}.\mathbf{r})^p \Rightarrow$$

$$c = c_r(c_a + c_l \max(0, \mathbf{n}.\mathbf{l})) + c_l(\mathbf{h}.\mathbf{n})^p$$

Phong belichtingsmodel

Phong belichtingsmodel is empirisch model voor locale belichting



Uit wikipedia

Visual illustration of Phong equation: here light is white, ambient and diffuse colors are both blue, and specular color is white, reflecting almost all of the light hitting the surface, but only in very narrow highlights. The intensity of the diffuse component varies with the direction of the surface, and ambient component is uniform (independent of direction).

Phong shading (normaalvector $\mathbf{n}$ interpoleren)

Berekenen belichting van polygoon-oppervlakken

`Stap 1`:
bepaal **normaalvector** in elk hoekpunt polygoon

`Stap 2`:
interpoleer **hoekpuntnormaalvectoren** over hele gebied polygoon
(Voor driehoek barycentrische coördinaten gebruiken: $\mathbf{n} = \alpha\mathbf{n}_0 + \beta\mathbf{n}_1 + \gamma\mathbf{n}_2$)

`Stap 3`:
bereken **kleur** voor elk punt met behulp van geïnterpoleerde normaalvector met belichtingsmodel