Computer Graphics CS248

Ray Tracing

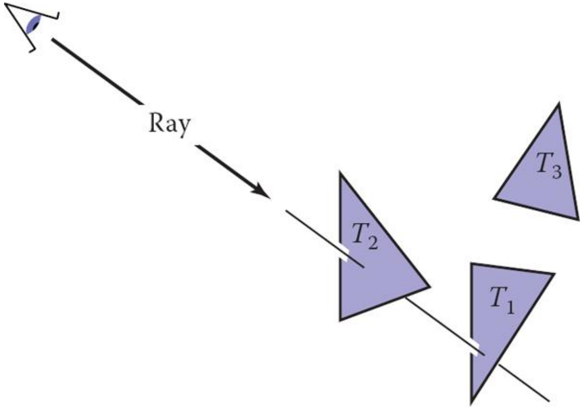
Ivan Viola

NANOVISUALIZATION GROUP

Ray Tracing

- Image-order rendering technique
- Inherently includes: perspective, shadowing, occlusion, reflection, global illumination, …

```
for each pixel {
    compute viewing ray
    for each object {
        find the intersection
    }
    select the closest object
    (cast secondary rays)
    perform shading
}
```

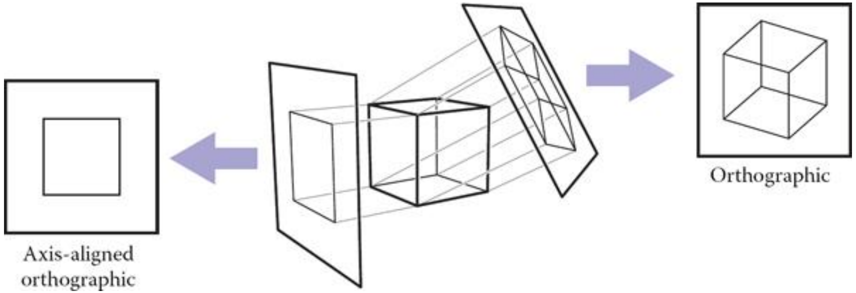


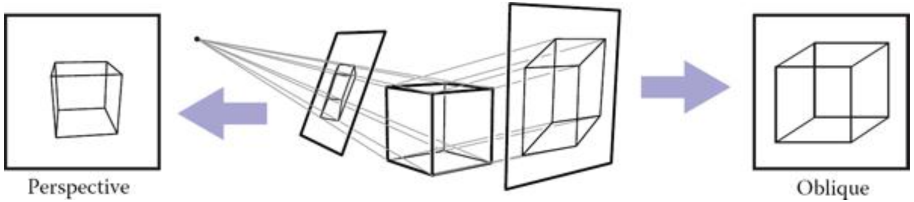
Projection

- Linear perspective
  - 3D object are projected onto an image plane
  - Straight lines in the scene become straight lines in the image

Projection (cont.)

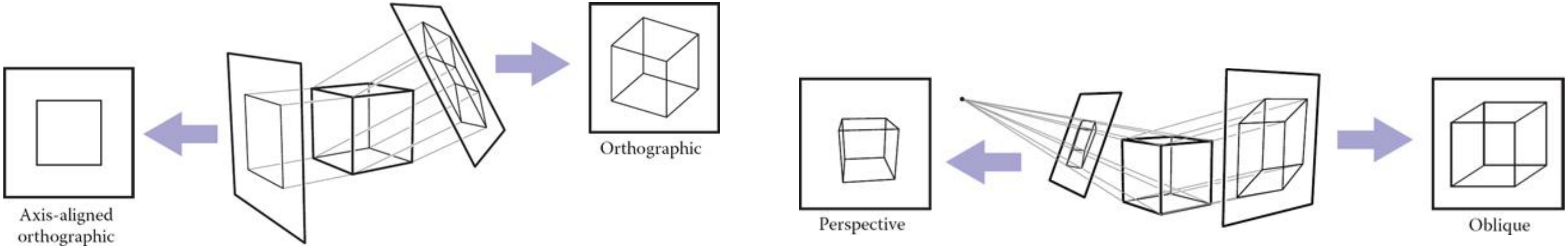
- Parallel projection: viewing rays are parallel lines to each other
- Perspective projection: viewing rays are cast from a shared viewpoint





Projection (cont.)

- Orthographic projection: central viewing ray is parallel to the normal of the viewing plane
- Oblique: otherwise

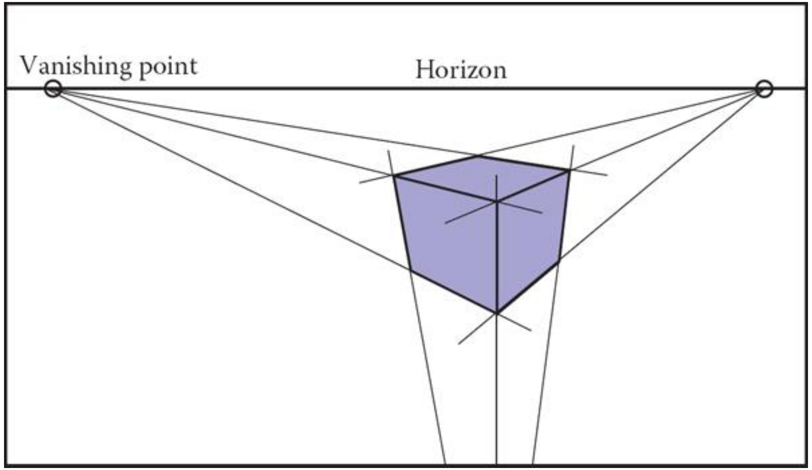


Parallel Projection

- Used in mechanical and architectural blueprints
- Parallel lines in object space correspond to parallel lines in image space
- Preserve the size and shape of planar objects that are parallel to the image plane

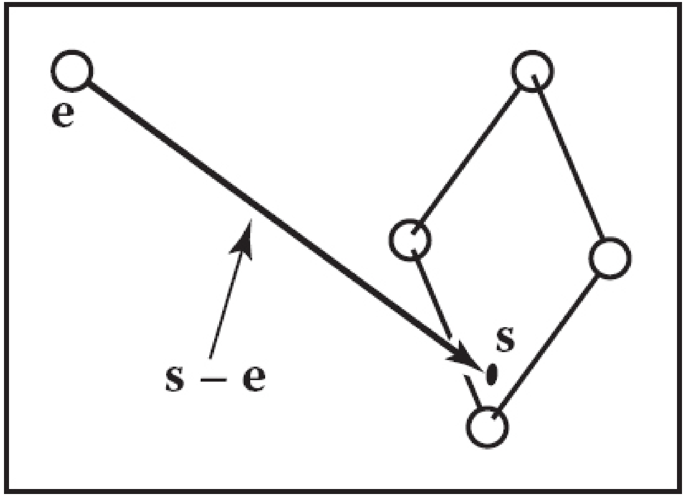
Perspective Projection

- Natural looking imagery, contains depth cues
- Three-point perspective with vanishing points and horizon line
- Vanishing point: where the parallel lines meet



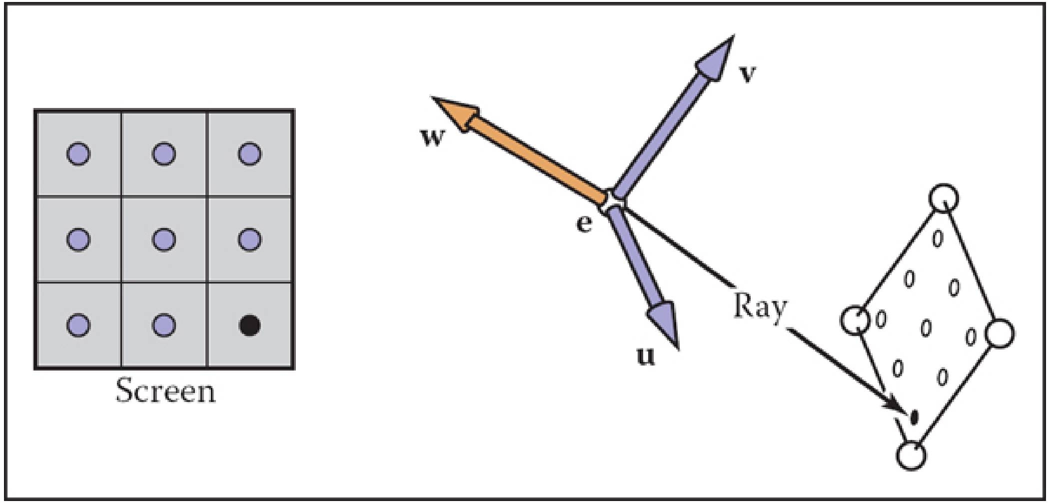
Computing Viewing Ray

- Ray is a parametric line $\boldsymbol{p}(t) = \boldsymbol{e} + t(\boldsymbol{s} - \boldsymbol{e})$
- $t_1 < t_2 \ \leftrightarrow |\boldsymbol{p}(t_1) - \boldsymbol{e}| < |\boldsymbol{p}(t_2) - \boldsymbol{e}|$
- $\boldsymbol{p}(0) = \boldsymbol{e}, \boldsymbol{p}(1) = \boldsymbol{s}$
- $t \in [0, \infty)$



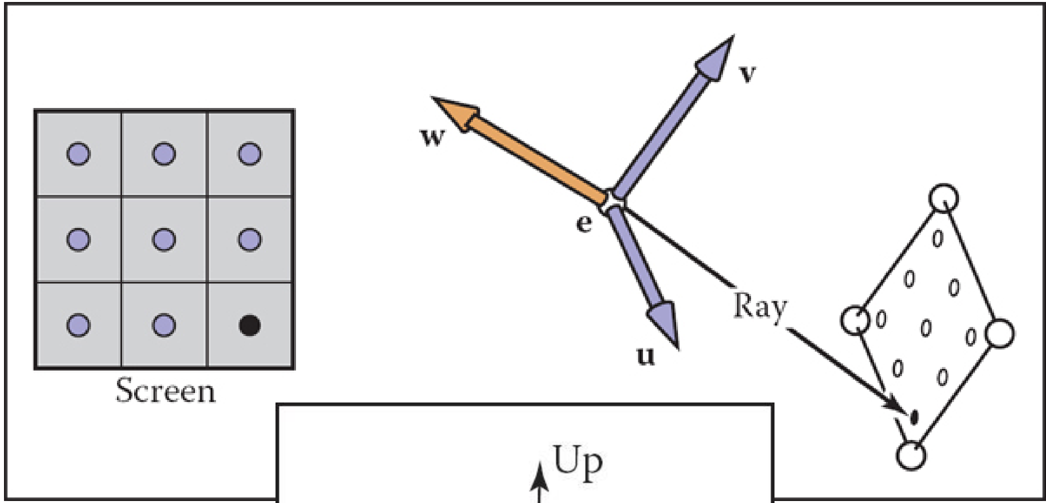
Camera Frame

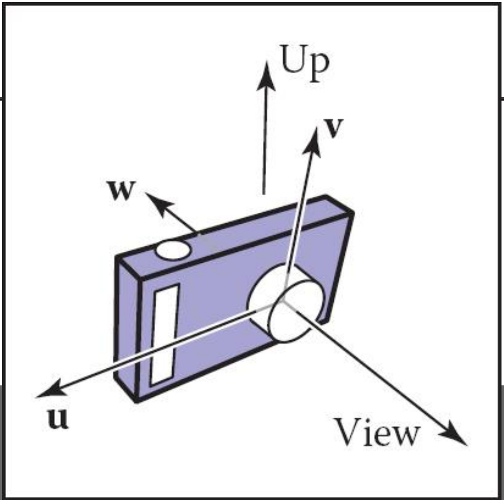
- Orthonormal coordinate frame
- $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ are basis vectors
- Right-handed coordinate system
- $\boldsymbol{y}$ is an *up-vector*



Camera Frame (cont.)

- $\boldsymbol{w} = -\boldsymbol{view}$
- $\boldsymbol{u} = \boldsymbol{y} \times \boldsymbol{w}$
- $\boldsymbol{v} = \boldsymbol{w} \times \boldsymbol{u}$



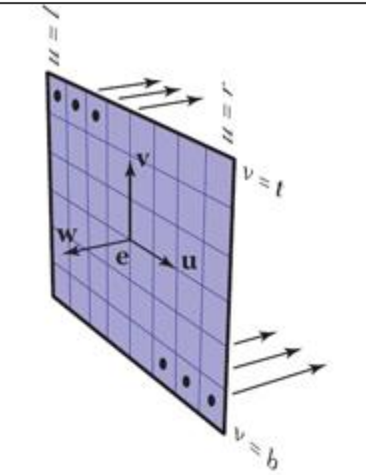


Computing Viewing Ray: Parallel

- Orthographic view direction $-\boldsymbol{w}$ is parallel to the viewing plane, $\boldsymbol{e}$ is viewport center
- Left, right, top, bottom are the edge parameters of the image

$$l < 0 < r, b < 0 < t$$

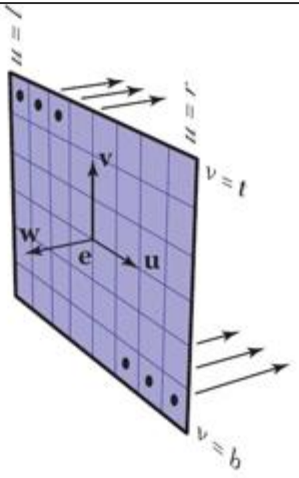
sometimes $l = -r, b = -t$



Parallel projection
same direction, different origins

Computing Viewing Ray: Parallel (cont.)

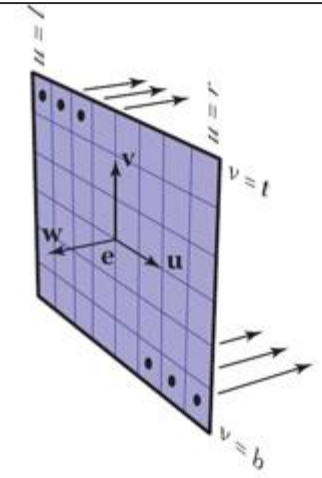
- Image contains $n_x \times n_y$ pixels
- Pixel spacing $s_x = \frac{(r-l)}{n_x}$, $s_y = \frac{(t-b)}{n_y}$
- $u = l + (r - l)\frac{i+0.5}{n_x}$
- $v = b + (t - b)\frac{j+0.5}{n_y}$



Parallel projection
same direction, different origins

Computing Viewing Ray: Parallel (cont.)

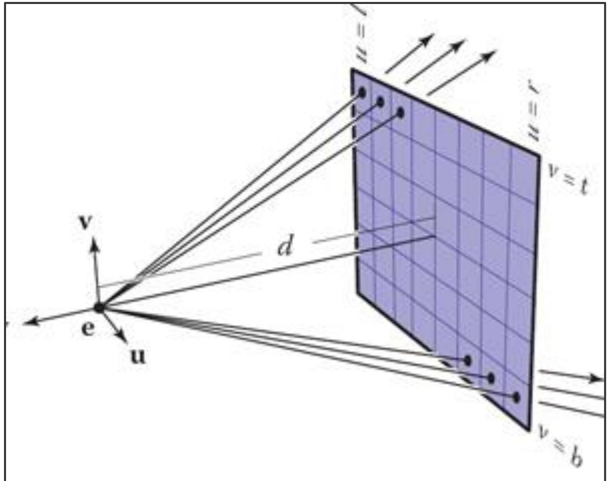
- $\boldsymbol{r}_d = -\boldsymbol{w}$
- $\boldsymbol{r}_o = \boldsymbol{e} + u\boldsymbol{u} + v\boldsymbol{v}$



Parallel projection
same direction, different origins

Computing Viewing Ray: Perspective

- Point $\boldsymbol{e}$ is eye position, distance d is image-plane distance, aka focal length
- u and v computed in the same way as in the parallel proj.
- $\boldsymbol{r_d} = -d\boldsymbol{w} + u\boldsymbol{u} + v\boldsymbol{v}$
- $\boldsymbol{r_o} = \boldsymbol{e}$



Perspective projection
same origin, different directions

Ray-Object Intersection: Sphere

- $\boldsymbol{p}(t) = \boldsymbol{e} + t\boldsymbol{d}, f(\boldsymbol{p}) = 0 \rightarrow f(\boldsymbol{e} + t\boldsymbol{d}) = 0$
- A sphere is defined through radius R and a center $\boldsymbol{c} = (x_c, y_c, z_c)$

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0$$
$$(\boldsymbol{p} - \boldsymbol{c}).(\boldsymbol{p} - \boldsymbol{c}) - R^2 = 0$$
$$(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}).(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}) - R^2 = 0$$

Ray-Object Intersection: Sphere

$$(\mathbf{d}.\mathbf{d})t^2 + 2\mathbf{d}.(\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}).(\mathbf{e} - \mathbf{c}) - R^2 = 0$$

$$t = \frac{-\mathbf{d}.(\mathbf{e} - \mathbf{c}) \pm \sqrt{(\mathbf{d}.(\mathbf{e} - \mathbf{c}))^2 - (\mathbf{d}.\mathbf{d}).((\mathbf{e} - \mathbf{c}).(\mathbf{e} - \mathbf{c}) - R^2)}}{(\mathbf{d}.\mathbf{d})}$$

$$n = \frac{\mathbf{p} - \mathbf{c}}{R}$$

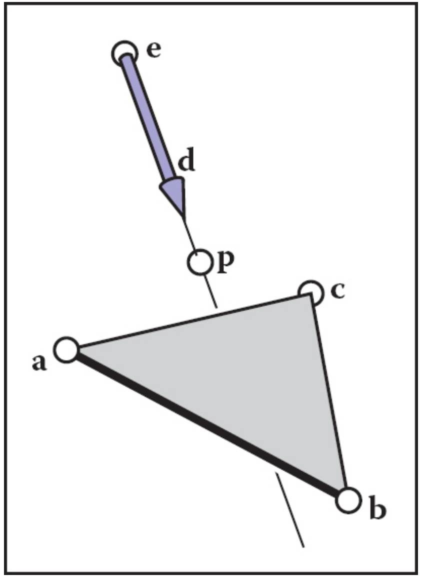
Ray-Object Intersection: Triangle

- Intersection of ray with a parametric plane

$$\boldsymbol{e} + t\boldsymbol{d} = \boldsymbol{a} + u\boldsymbol{u} + v\boldsymbol{v}$$

- Triangle: $= \boldsymbol{a} + \beta(\boldsymbol{b} - \boldsymbol{a}) + \gamma(\boldsymbol{c} - \boldsymbol{a})$
- t, β, γ are unknown values

  inside test: $\beta > 0, \gamma > 0, \beta + \gamma < 1$



Ray-Object Intersection: Triangle (cont.)

$$x_e + tx_d = x_a + \beta(x_b - x_a) + \gamma(x_c - x_a)$$
$$y_e + ty_d = y_a + \beta(y_b - y_a) + \gamma(y_c - y_a)$$
$$z_e + tz_d = z_a + \beta(z_b - z_a) + \gamma(z_c - z_a)$$

$$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}$$

Ray-Object Intersection: Triangle (cont.)

Cramer's rule: $x = \frac{D_x}{D}$

$$\beta = \frac{\begin{vmatrix} x_a - x_e & x_a - x_c & x_d \\ y_a - y_e & y_a - y_c & y_d \\ z_a - z_e & z_a - z_c & z_d \end{vmatrix}}{\begin{vmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{vmatrix}}$$

Ray-Object Intersection: Triangle (cont.)

```
bool raytri (ray r, vec3 a, vec3 b, vec3 c, int t0, int t1){
    compute t
    if (t < t0) or (t > t1) return false
    compute gamma
    if (gamma < 0) or (gamma > 1) return false
    compute beta
    if (beta < 0) or (beta > 1- gamma) return false
    return true
}
```

Ray-Object Intersection: Polygon

Planar polygon defined through vertices $\boldsymbol{p}_1 \ldots \boldsymbol{p}_m$

$$(\boldsymbol{p} - \boldsymbol{p}_1) \cdot \boldsymbol{n} = 0, \boldsymbol{p} = \boldsymbol{e} + t\boldsymbol{d}$$

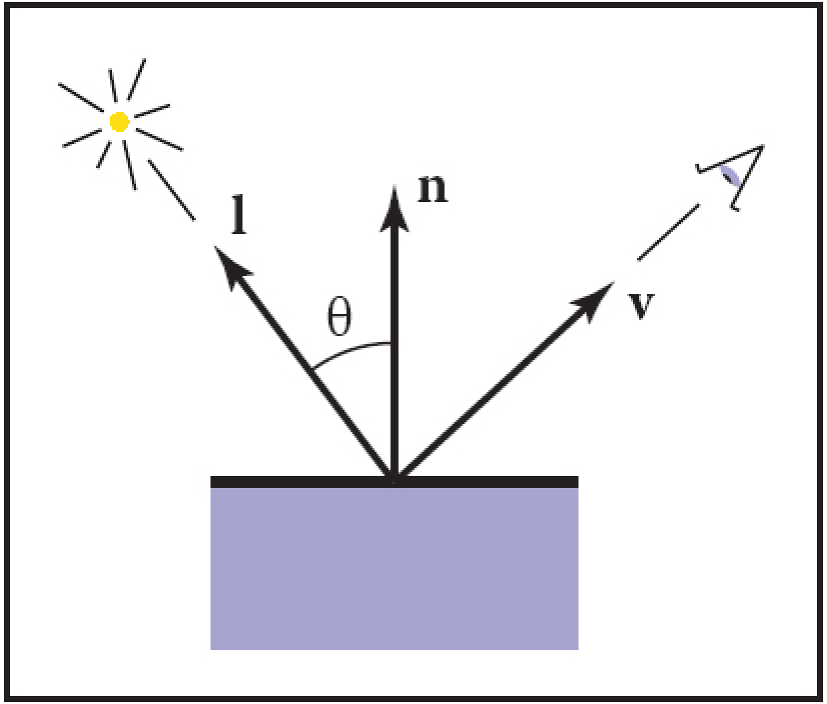
$$t = \frac{(\boldsymbol{p}_1 - \boldsymbol{e}).\boldsymbol{n}}{\boldsymbol{d}.\boldsymbol{n}}$$

- Project polygon and $\boldsymbol{p}$ onto $\boldsymbol{xy}$-plane
- Cast ray from $\boldsymbol{p}$ within plane and calculate number of intersections with edges of polygon
- Odd number of intersections, $\boldsymbol{p}$ is inside

Shading: Lambertian Illumination

- Assuming point light
- Proportional to cosine of light $\boldsymbol{l}$ and normal $\boldsymbol{n}$ vectors

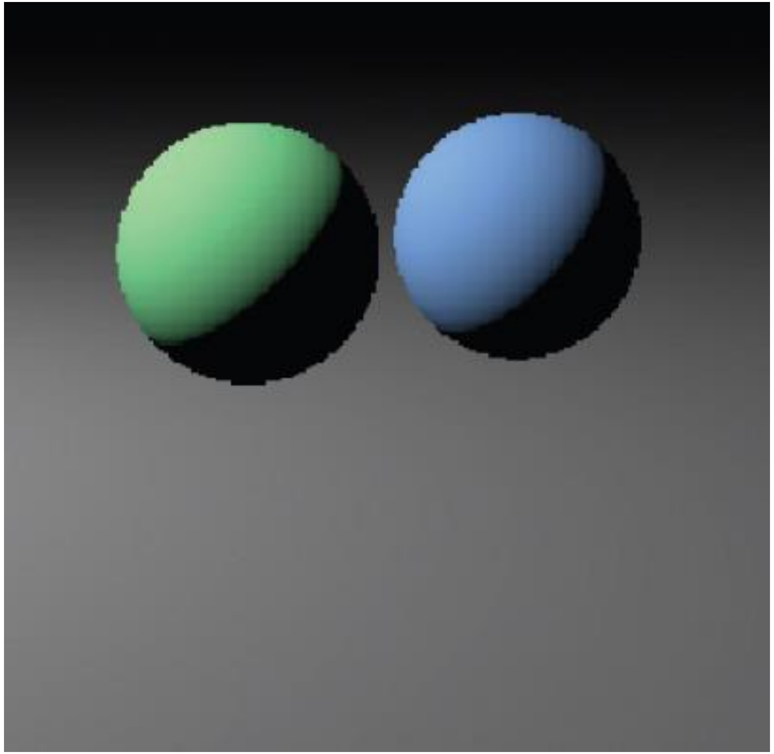
$$L = k_d . I \max(0, \boldsymbol{n}.\boldsymbol{l})$$



Shading: Lambertian Illumination (cont.)

- Assuming point light
- Proportional to cosine of light $\boldsymbol{l}$ and normal $\boldsymbol{n}$ vectors

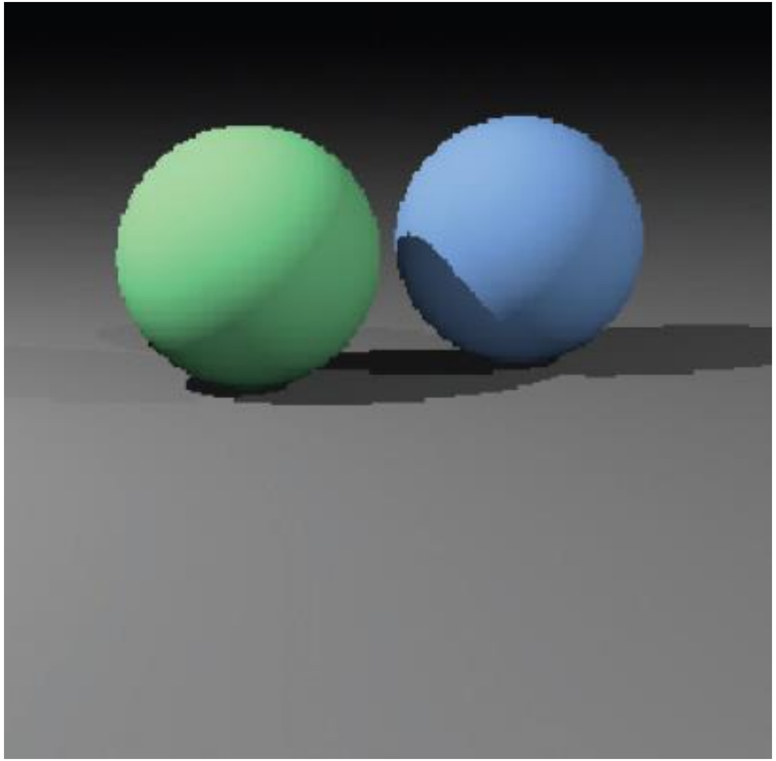
$$L = k_d . I \max(0, \boldsymbol{n} . \boldsymbol{l})$$



Shading: Lambertian Illumination (cont.)

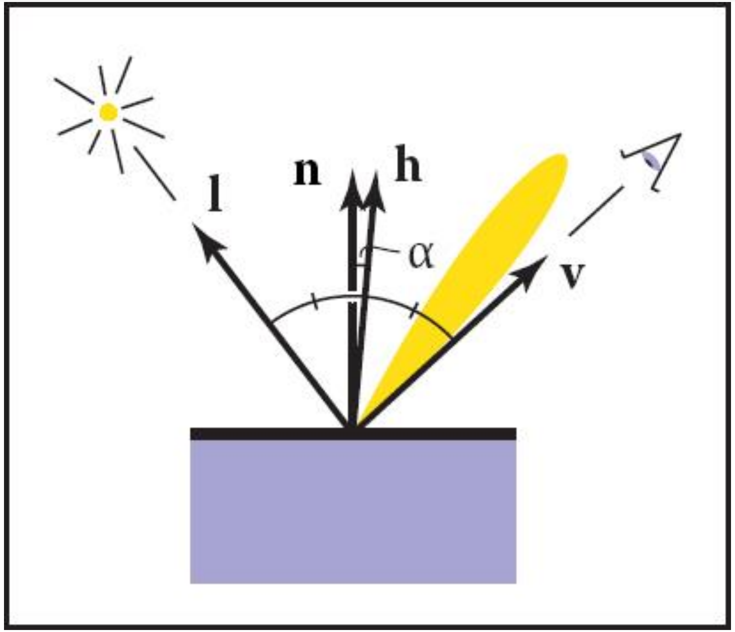
- Assuming point light
- Proportional to cosine of light $\boldsymbol{l}$ and normal $\boldsymbol{n}$ vectors

$$L = k_d . I \max(0, \boldsymbol{n} . \boldsymbol{l})$$



Shading: Blinn-Phong Illumination

- Adding view-dependent specular component
- If view $\boldsymbol{v}$ and light $\boldsymbol{l}$ vectors are symmetric wrt. normal $\boldsymbol{n}$ vectors

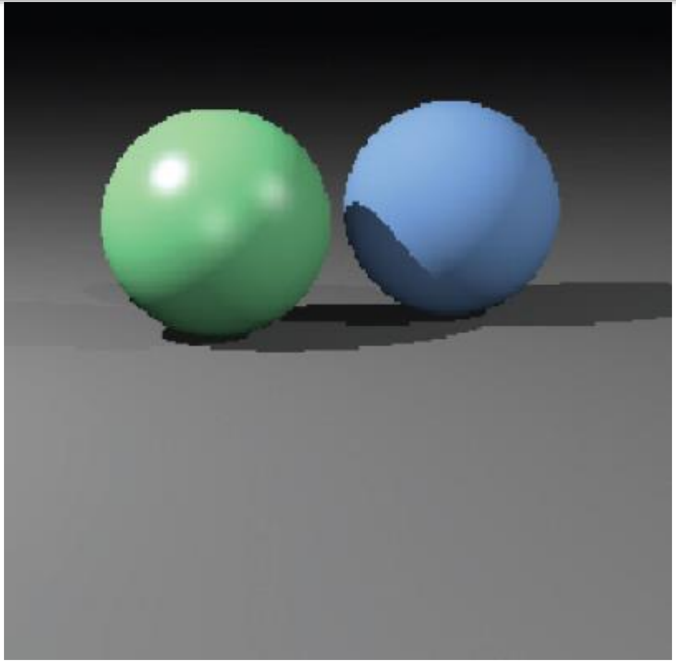


$$\boldsymbol{h} = \frac{(\boldsymbol{v}+\boldsymbol{l})}{\|\boldsymbol{v}+\boldsymbol{l}\|}$$

$$L = k_d . I \max(0, \boldsymbol{n}.\boldsymbol{l}) + k_s . I \max(0, \boldsymbol{n}.\boldsymbol{h})^p$$

Shading: Blinn-Phong Illumination (cont.)

- Adding view-dependent specular component
- If view $\boldsymbol{v}$ and light $\boldsymbol{l}$ vectors are symmetric wrt. normal $\boldsymbol{n}$ vectors



$$\boldsymbol{h} = \frac{(\boldsymbol{v} + \boldsymbol{l})}{\|\boldsymbol{v} + \boldsymbol{l}\|}$$

$$L = k_d . I \max(0, \boldsymbol{n}.\boldsymbol{l}) + k_s . I \max(0, \boldsymbol{n}.\boldsymbol{h})^p$$

Shading: Ambient Illumination

- Adding constant minimal illumination from the environment

$$L = k_a . I_a + k_d . I \max(0, \boldsymbol{n} . \boldsymbol{l}) + k_s . I \max(0, \boldsymbol{n} . \boldsymbol{h})^p$$

$$k_a + k_d + k_s = 1$$

- Ambient occlusion: I_a illumination dependent on the concavity or convexity of the area near surface point

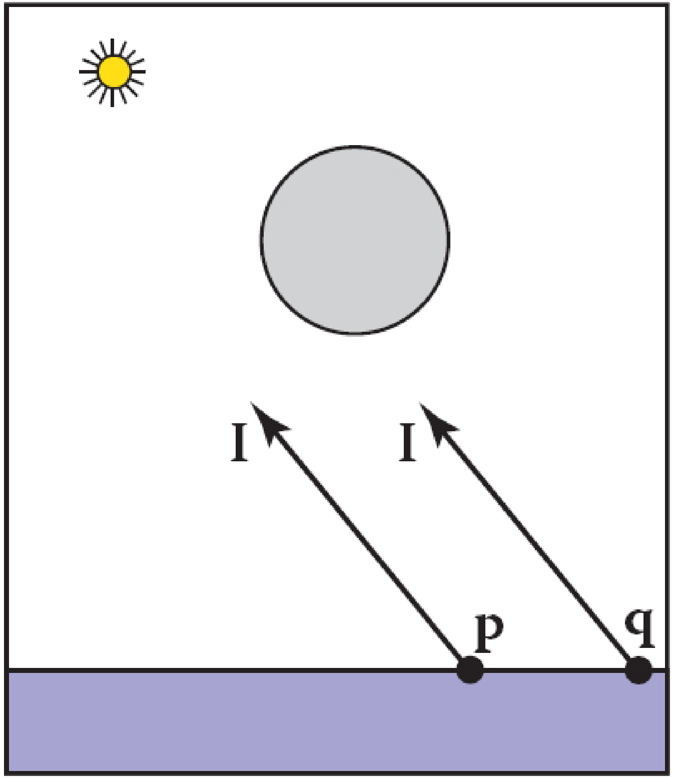
Shading: Multiple Light Sources

- Lights can be superpositioned through their additive property

$$L = k_a . I_a + \sum_{i=1}^{N} [k_d . I_i \max(0, \boldsymbol{n} . \boldsymbol{l}_i) + k_s . I_i \max(0, \boldsymbol{n} . \boldsymbol{h}_i)^p]$$

Shadows

- Shadow ray $\boldsymbol{l}_i$ from point $\boldsymbol{p}$ might intersect another surface geometry. If so, point $\boldsymbol{p}$ is in shadow and should not be illuminated by the i-th light source
- Intersection geometry vs. $\boldsymbol{p} + t\boldsymbol{l}_i,\ t \in [\varepsilon, d_i)$



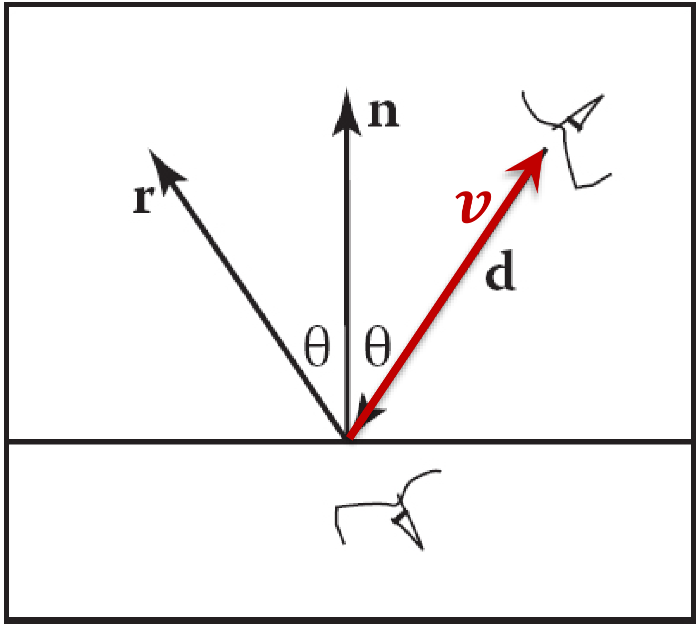
Reflection

- Ideal specular (mirror) reflection casts secondary ray $\boldsymbol{r}$

$$\boldsymbol{r} = 2(\boldsymbol{v}.\boldsymbol{n})\boldsymbol{n} - \boldsymbol{v}$$

- Illumination of surface affected by reflection and illumination of the secondary ray

$$L = L + k_r L_r(\mathbf{p}, \mathbf{r})$$



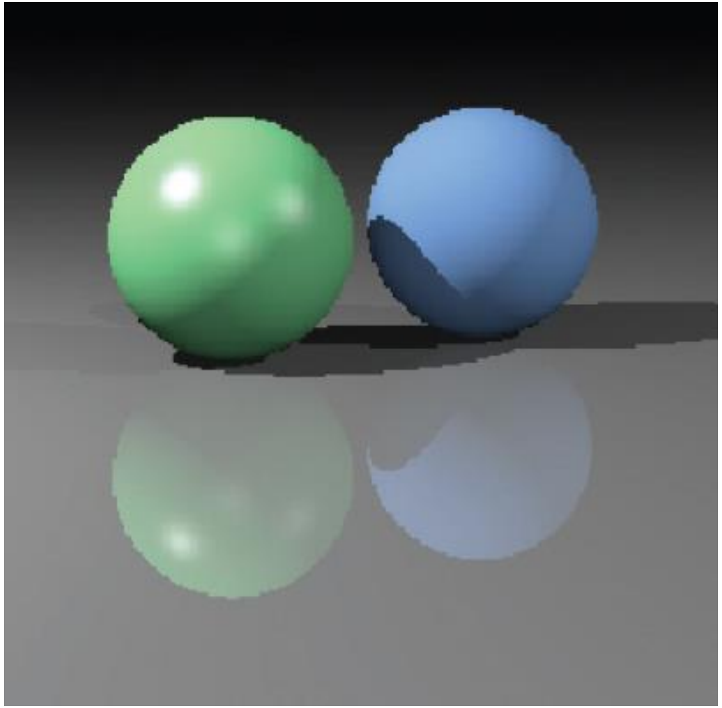
Reflection

- Ideal specular (mirror) reflection casts secondary ray $\boldsymbol{r}$

$$\boldsymbol{r} = 2(\boldsymbol{v}.\boldsymbol{n})\boldsymbol{n} - \boldsymbol{v}$$

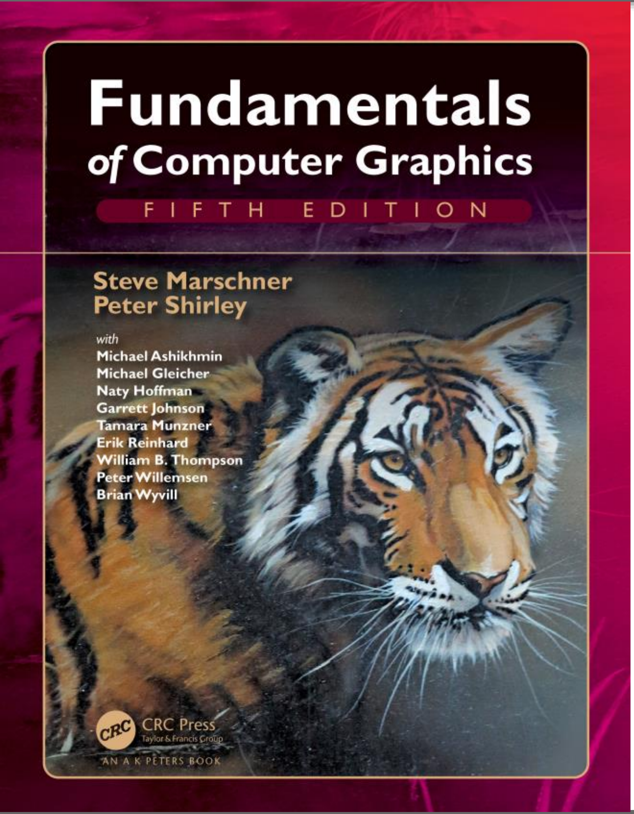
- Illumination of surface affected by reflection and illumination of the secondary ray

$$L = L + k_r L_r(\boldsymbol{p}, \boldsymbol{r})$$



Ray Tracing in Hardware

Credits



Fundamentals of Computer Graphics, 5th Edition

by Peter Shirley, Steve Marschner

Publisher: A K Peters/CRC Press

Release Date: September 30, 2021

ISBN: 9781000426359

https://learning.oreilly.com/library/view/fundamentals-of-computer/9781000426359/