Chapter4: Ray Tracing

object rendering: each object is considered in turn
image-order rendering: each pixel is considered in turn

> Ray Tracing is an image-order algorithm for making renderings of 3D scenes

4.1 The Basic Ray-Tracing Algorithm

a basic ray tracer have 3 parts:

- **ray generation**
  which computes the origin and direction of each pixel's viwing ray based on the camera geometry
- **ray intersection**
  find the cloeset object intersecting the viewing ray
- **shading**
  computes the pixel color based on the results of ray intersection

```
for each pixel do
    computing viewing ray
    ray, surface norm n -> first object hit
    light, hit point, n -> pxiel color
```

4.2 Perspective

linear perspective

- **parallel projection**
  orthographic oblique
- **perspective projection**

4.3 Computing Viewing Rays

A ray is really just an origin point and a propogation direction,a 3D parametric line is ideal for this
$p(\mathbf{t}) = \mathbf{s} + t(\mathbf{e} - \mathbf{s})$

4.3.1 Orthographic Views

the pixel at position (i,j) in the raster images has the position
$u = l + (r - l)(i + 0.5)/n_x$
$v = b + (t - b)(j + 0.5)/n_y$
where (u, v) are the coordinates of the pixel's position on the image plane

4.3.2 Perspective Views

all the rays have same origin

compute $\mathbf{w}$ and $\mathbf{w}$
ray.direction $\leftarrow -d\mathbf{w} + u\mathbf{u} + v\mathbf{v}$
ray.origin $\leftarrow e$

4.4 Ray-Object Intersection

4.4.1 Ray-Sphere Intersection

$\mathbf{p} = \mathbf{e} + t\mathbf{d}$
$\mathbf{c} = (x_c, y_c, z_c)$
$(\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - R^2 = 0$
this is a classic quadratic equation in t
$discriminant < 0$ line and sphere do not intersect
$discriminant > 0$ line enter and leave the sphere
$discriminant = 0$ ray graze the sphere

Ray-Triangle Intersection

the vertices of the triangle is in 3D space
$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$
$\beta > 0 \quad \gamma > 0 \quad \beta + \gamma < 1$ intersection is inside the triangle, otherwise the ray hit the plane outside

of the triangle.
if no solution, either the traingle is degenerate or the ray is paralled to the plane containing the traingle

4.4.3 Ray-Polygon Intersection

4.4.4 Intersecting a Group of Objects

4.5 Shading

4.5.1 Lambertian Shading

a surface facing directly toward the light receives maximum illumination
a surface tangent to tht light direction recevies no illumination
$L = k_d I max(0, \mathbf{n} \cdot \mathbf{l})$
L : pixel color
k_d diffuse coefficient
I : intensity of the light source
$\mathbf{l}$: light direction
$\mathbf{v}$: view direction
this equation applies separately to the three color channels

4.5.2 Blinn-Phong Shading

Lambertian Shading is *view independent*
the color of a surface does not depend on the direction from which you
shininess, **producing highlights**, **specular reflections** appear to move around the view point changes
$\mathbf{h} = \dfrac{\mathbf{v} + \mathbf{l}}{\| \mathbf{v} + \mathbf{l} \|}$
$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^p$
k_s : *specular coefficient*
p: Phong exponent

4.5.3 Ambient Shading

$L = k_a I_a + k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^p$
k_a : surface's ambient coefficient/ ambient color

I_a : ambient light intensity

4.5.4 Multiple Point Lights

simple shading model can easily be extended to handle *N* light sources
$L = k_a I_a + \sum_{i=1}^{N} [k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^p]$

4.6 A Ray-Tracing

4.7 Shadows

4.8 Ideal Specular Reflection

mirror reflection
some energy is lost when the light reflects from the surface, and this loss can be different for different colors
$colorc = c + k_m raycolor(\mathbf{p} + sr, \epsilon, \infty)$
k_m : specular RGB color