Chapter 4 - Ray Tracing

Summarized from "Fundamentals of Computer graphics".

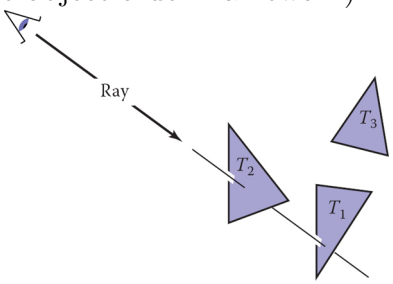
Def of **rendering**: taking a scene, or model, composed of many geometric objects arranged in 3D space and producing 2D image that shows the objects as view from a particular view point.

It can be organized in two general ways: *object-order* rendering and *image-order* rendering.

object-order rendering: each object is considered in turn, and for each object all pixels that it influences are found and updated.

image-order rendering: each pixel is considered in turn, and for each pixel all the objects that influence it are found and pixel value is computed.

Broadly speaking, image-order rendering is simpler to get working and more flexible in the effects that can be produced, and usually takes much more execution time produce a comparable image. (In a ray tracer, it is easy to computer accurate shadows and reflections, which are awkward in the object-order framework).



Ray tracing is an image-order algorithm for making renderings of 3D scenes.

4.1 The Basic Ray-Tracing Algorithm

Computing one pixel at a time, and for each pixel the basic task is to find the object that is seen at that pixel's position in the image. Any *object* that is seen by a pixel must intersect the *viewing ray: a line that emanates from the viewpoint in the direction that pixel is looking*. Once that object is found, a ***shading*** computation uses the intersection point, surface normal, and other info to determine the color of the pixel.

A basic ray tracing therefore has three parts:

1. **ray generation**: computes the origin and direction each pixel's viewing ray based on the camera geometry.
2. **ray intersection**: finds the closest object intersecting the viewing ray.
3. **shading**: computes the pixel color based on the result of ray intersection.

The struct of the basic ray algo:

```
for each pixel do:
    computing viewing ray;
    find first object hit by ray and its surface normal n;
    set pixel color to value computed from hit point, light, and n;
```

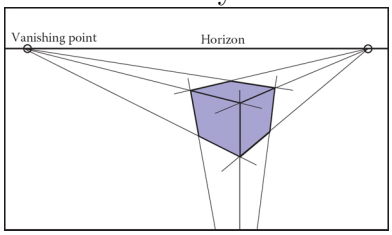
4.2 Perspective

linear perspective: 3D objects are projected onto an *image plane* in such a way that straight lines in the scene become straight lines in the image.(An image taking with a fisheye lens is not a linear perspective image)

The simplest type of projection is **parallel projection**, in which 3D points are mapped to 2D by moving them along a *projection direction* until they hit the image plane. If the image plane is perpendicular to the view direction, the projection is called **orthographic**; otherwise it is called **oblique**.

Parallel projections keep parallel lines parallel and they preserve the size and shape of planar objects that are parallel to the image plane.

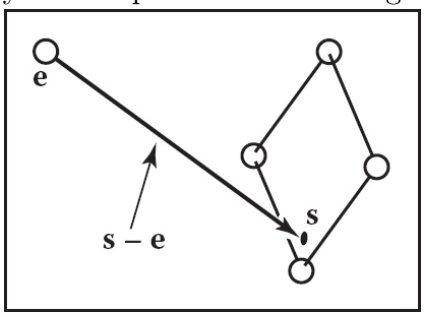
Producing natural-looking views using **perspective projection**: we simply project along line that pass through a single **point**, the **viewpoint**, rather than along parallel lines. A perspective view is determined by the choice of viewpoint and image plane.



Parallel horizontal lines will meet at a point on the horizon. Every set of parallel lines has its own vanishing points. Objects are projected directly toward the **eye**, and they are drawn where they meet a view plane in front of the eye.

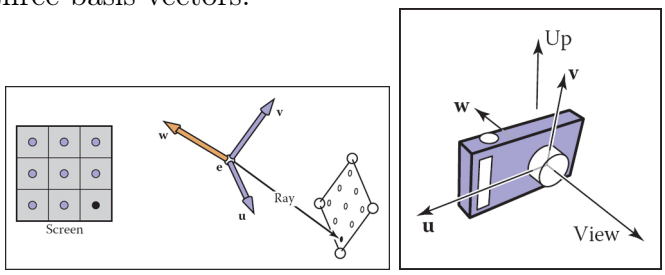
4.3 Computing Viewing Rays

A ray is really just an origin point and a propagation direction. The 3D parametric line from the eye e to a point s on the image plane is given by $p(t) = e + t(s - e)$.



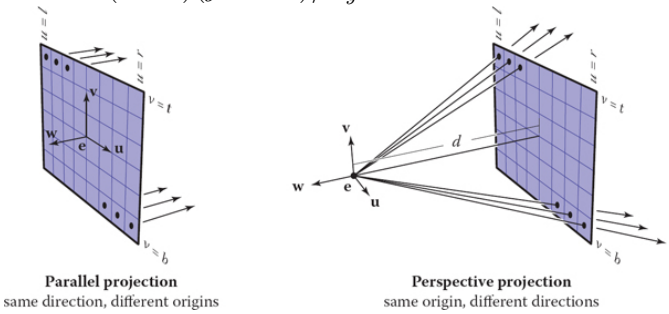
"We advance from e along the vector $(s-e)$ a fractional distance t to find the point p." So given t, we can determine a point p. The point e is the **ray's origin**, and $s - e$ is the **ray's direction**. Finding s may seem difficult.

Camera frame, which we denote by e, for the eye point, or view point, and u, v, and w for the three basis vectors.



4.3.1 Orthographic Views

The pixel at position (i, j) in the raster image has the position: $u = l + (r - l)(i + 0 - .5)/n_x$ and $v = b + (t - b)(j + 0.5)/n_y$.



Ray generation using the camera frame. Left: In an orthographic view, the rays start at the pixel's location on the image plane, and all share the same direction, which is equal to the view direction. Right: In a perspective view, the rays start at the viewpoint, and each ray's direction is defined by the line through the viewpoint e, and the pixel's location on the image plane.

```
compute u and v using 4.3.1
ray direction: -w
ray.origin: e + u*U(basis) + v*V(basis)
```

4.3.2 Perspective Views

For a perspective view, all the rays have the same origin, at the viewpoint. The image plane is no longer positioned at e, but rather some distance d in front of e; this distance is the **image plane distance**, or *focal length*. The direction of each ray is defined by the viewpoint and the position of the pixel on the image plane.

```
compute u and v using 4.3.1
ray direction: -dw + u*U(basis) + v*V(basis)
ray.origin: e
```

4.4 Ray-Object Intersection

We've generated a ray $e + td$, next we need to find the first intersection with any object where $t > 0$

4.4.1 Ray-Sphere Intersection

Given a ray $p(t) = e + td$ and a implicit surface $f(p) = 0$. Intersection points occur when points on the ray satisfy the implicit equation, so the values of t we seek are those that solve the equation $f(p(t)) = 0$ or $f(e + td) = 0$. A sphere with center $c = (xc, yc, zc)$ and radius R can be represented by the implicit equation $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0$. Can write this same equation in vector form: $(p - c)(p - c) - R^2 = 0$. Therefore, $(e + td - c) * (e + td - c) - R^2 = 0$. Rearranging terms yieldsThe normal vector at point p is given by the gradient $n = 2(p - c)$, and the unit normal is $(p - c)/R$.

$(d * d)t^2 + 2d(e - c)t + (e - c)(e - c) - R^2 = 0 => At^2 + Bt + C = 0$.

The normal vector at point p is given by the gradient $n = 2(p - c)$, and the unit normal is $(p - c)/R$.

4.5 Shading

Shading models are designed to capture the process of light reflection, whereby surface are illuminated by light source and reflect part of the light to the camera. Simple shading models are defined in terms of illumination from a point light source. The important variables in light reflection are:

1. The light direction l, which is a unit vector pointing toward the light source.
2. The view direction v, which is a unit vector pointing toward the eye or camera.
3. The surface normal n, which is a unit vector perpendicular to the surface at the point
4. The characteristics of the surface - color, shininess, or others.

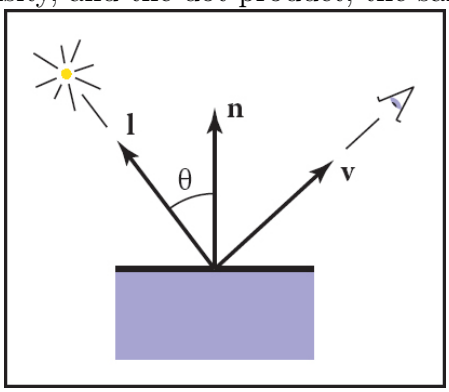
4.5.1 Lambertian Shading

Simplest shading model: the amount of energy from a light source that falls on an area of surface

depends on the angle of the surface to the light. The illumination is proportional to the cosine of the angle θ between the surface normal and the light source. This leads to the *Lambertian shading model*:

$$L = k_d \; I \, max(0, \mathbf{n} \cdot \mathbf{l})$$

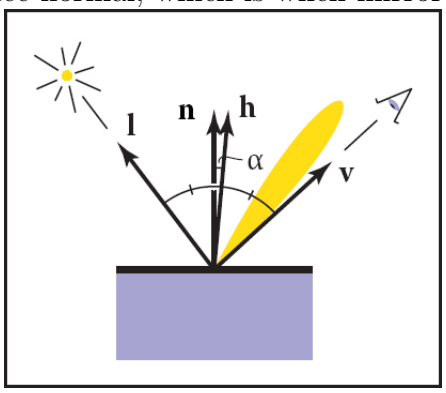
where L is the pixel color; k_d is the *diffuse coefficient*, or the surface color? and I is the intensity of light source. Since $\mathbf{n}$ and $\mathbf{l}$ are unit vectors, we can use $\mathbf{n} \cdot \mathbf{l}$ as a convenient shorthand for $cos(\theta)$. The red component of the pixel value is the product of the red diffuse component, the red light source intensity, and the dot product; the same holds for green and blue.



4.5.2 Blinn-Phong Shading

Lambertian shading is *view independent*: the color of a surface does not depend on the direction from which we look. However, many real surfaces show some degree of shininess, producing highlights, or specular reflections, that appear to move around as the viewpoint changes. Many shading models add a *specualr component* to Lambertian shading; the Lambertian part is then the *diffuse component*.

A very simple and widely used model for specular highlights was proposed. The idea is to produce reflection that is at its brightest when v and l are symmetrically positioned across the surface normal, which is when mirror reflection would occur.



The Blinn-Phong shading model is as follows:

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{||\mathbf{v} + \mathbf{l}||}, \;\; L = k_d \; I \, max(0, \mathbf{n} \cdot \mathbf{l}) + k_s \; I \; max(0, \mathbf{n} \cdot \mathbf{h})^p$$

where k_s is the **specular coefficient**, or the specular color of the surface

4.5.3 Ambient Shading

A crude but useful heuristic to avoid black shadows is to add a constant component to the shading model, one whose contribution to the pixel color depends only on the object hit, with no dependence on the surface geometry at all. Together with the rest of the Blinn-Phong model, ambient shading completes the full version of a simple and useful shading model:

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

where k_a is the surface's ambient coefficient, or "ambient color", and I_a is the ambient light intensity.

4.5.3 Multiple Point Lights

A very useful property of light is *superposition*, then simple shading model can easily be extended to handle N light sources

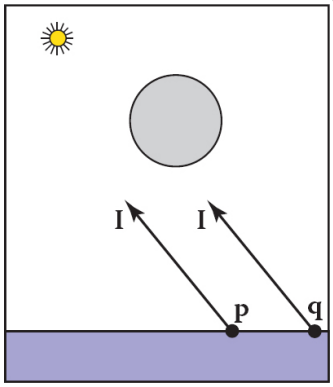
$$L = k_a\ I_a + \sum_{i=1}^{N} [k_d\ I_i\ max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s\ I_i\ max(0, \mathbf{n} \cdot \mathbf{h_i})^p]$$

where I_i, $\mathbf{l_i}$, and $\mathbf{h_i}$ are the intensity, direction, and half vector of the i^{th} light source.

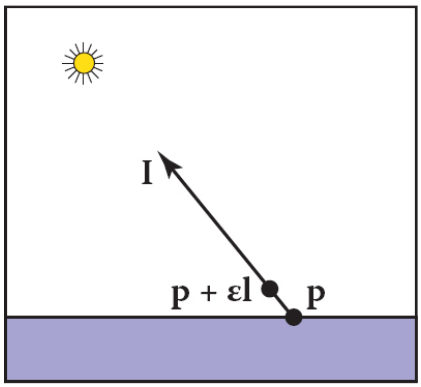
4.6 A Ray-Tracing Program

4.7 Shadow

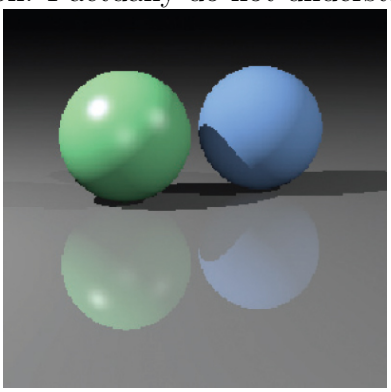
Once we have a basic ray tracing program, shadows can be added very easily



The ray $\mathbf{p} + t\mathbf{l}$ does not hit any objects and is thus not in shadow. The point $\mathbf{q}$ is in shadow because the ray $\mathbf{q} + t\mathbf{l}$ does hit an object. The vector $\mathbf{l}$ is the same for both points because the light is "far" away. The ray that determine in or out of shadow are called **shadow rays** to distinguish them from viewing rays.



By testing the interval starting at ϵ, we avoid numerical imprecision causing the ray to hit the surface p is on. I actually do not understand here.



A simple scene rendered with diffuse and Blinn-Phong shading from three light sources, and specular reflection from the floor.