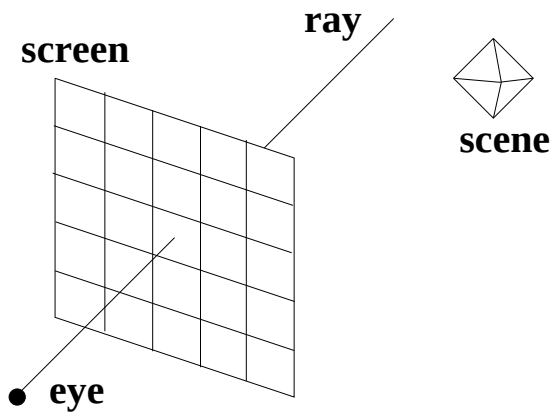
Global Illumination: Ray Tracing

Basic Definitions

Ray Tracing/Casting:

- Setting: *eyepoint*, *virtual screen* (an array of *virtual pixels*, and *scene* are organized in convenient coordinate frames (e.g., all in view or world frames)
- Ray: a half line determined by the eyepoint and a point associated with a chosen pixel
- Interpretations:
  - Ray is the path of photons that successfully reach the eye
    (we simulate selected photon transport through the scene)
  - Ray is a sampling probe that gathers color/visibility information



Ray Tracing: Recursive

- Eye-screen ray is the *primary* ray
- Backward tracking of photons that could have arrived along primary
- Intersect with objects
- Determine nearest object
- Generate secondary rays
  - to light sources
  - in reflection-associated directions
  - in refraction-associated directions
- Continue recursively for each secondary ray
- Terminate after suitably many levels
- Accumulate suitably averaged information for primary ray
- Deposit information in pixel

See raytrace psuedo-code document http://www.cs.utexas.edu/users/bajaj/graphics24/cs354/raytrace

Ray Casting: Nonrecursive

- As above for ray tracing, but stop before generating secondary rays
- Apply illumination model at nearest object intersection with no regard to light occlusion
- Ray becomes a sampling probe that just gathers information on
  - visibility
  - color

Intersection Computations

General Issues:

- Ray: express in parametric form

$$E + t(P - E)$$

  where E is the eyepoint and P is the pixel point
- Scene Object: direct implicit form
  - express as $f(Q) = 0$ when Q is a surface point, where f is a given formula
  - intersection computation is an equation to solve: find t such that $f(E + t(P - E)) = 0$
- Scene Object: procedural implicit form
  - f is not a given formula
  - f is only defined procedurally
  - $\mathcal{A}(f(E + t(P - E)) = 0)$ yields t, where $\mathcal{A}$ is a root finding method (secant, Newton, bisection, etc.)

Quadric Surfaces:

- Surface given by

$$Ax^2 + Bxy + Cxy + Dy^2 + Eyz + Fz^2 + Gx + Hy + Jz + K = 0$$

- Ray given by

$$\begin{aligned} x &= x_E + t(x_P - x_E) \\ y &= y_E + t(y_P - y_E) \\ z &= z_E + t(z_P - z_E) \end{aligned}$$

- Substitute ray x, y, z into surface formula
  - quadratic equation results for t
  - organize expression terms for numerical accuracy; i.e., to avoid
    - cancellation
    - combinations of numbers with widely different magnitudes

Polygons:

- The plane of the polygon should be known

$$Ax + By + Cz + D = 0$$

  - $(A, B, C, 0)$ is the normal vector
  - pick three successive vertices

$$\begin{aligned} v_{i-1} &= (x_{i-1}, y_{i-1}, z_{i-1}) \\ v_i &= (x_i, y_i, z_i) \\ v_{i+1} &= (x_{i+1}, y_{i+1}, z_{i+1}) \end{aligned}$$

  - should subtend a "reasonable" angle (bounded away from 0 or 180 degrees)
  - normal vector is the cross product $v_{i-1} - v_i \times v_{i+1} - v_i$
  - $D = -(Ax + By + Cz)$ for any vertex $(x, y, z, 1)$ of the polygon
- Substitute ray x, y, z into surface formula
  - linear equation results for t
- Solution provides planar point $(\overline{x}, \overline{y}, \overline{z})$

– is this inside or outside the polygon?

Planar Coordinates:

- Take origin point and two independent vectors on the plane

$$\begin{aligned} \mathcal{O} &= (x_{\mathcal{O}}, y_{\mathcal{O}}, z_{\mathcal{O}}, 1) \\ \vec{b}_0 &= (u_0, v_0, w_0, 0) \\ \vec{b}_1 &= (u_1, v_1, w_1, 0) \end{aligned}$$

- Express any point in the plane as

$$P - \mathcal{O} = \alpha_0 \vec{b}_0 + \alpha_1 \vec{b}_1$$

  - intersection point is $(\vec{\alpha}_0, \vec{\alpha}_1, 1)$
  - clipping algorithm against polygon edges in these α coordinates

Alternatives: Must this all be done in world coordinates?

- Alternative: Intersections in model space
  - form normals and planar coordinates in model space and store with model

  - backtransform the ray into model space using inverse modeling transformations
  - perform the intersection and illumination calculations
- Alternative: Intersections in world space
  - form normals and planar coordinates in model space and store
  - forward transform using modeling transformations

Ray-Polyhedron Intersection

(This algorithm is a variation/extension of the Liang-Barsky line segment clipping algorithm)

The convex polyhedron is defined as the intersection of a collection of halfspaces in 3-space. Represent the ray parametrically as $P(t) = P + t\vec{u}$, for scalar $t > 0$, point P and a unit vector $\vec{(}u)$. Let H_1, H_2, . . . , H_k denote the halfspaces defining the polyhedron. Compute the intersection of the ray with each halfspace in turn. The final result will be the intersection of the ray with the entire polyhedron.

An important property of convex bodies (of any variety) is that a line intersects a convex body in at most one line segment. Thus the resulting intersection of the ray with the polyhedron can be specified entirely by an interval of scalars $[t_0, t_1]$. The parametric representation of the intersection is defined by the line segment Initially, let this interval be $[0, \infty]$.

(For a line-segment intersection with a polyhedron, the only change is that an initial value of t_1 is set as the endpoint of the segment rather than ∞).

Suppose we have already performed the intersection with some number of the halfspaces. It might be that the intersection is already empty. This will be reflected by the fact that

$t_0 > t_1$. When this is so, we may terminate the algorithm at any time. Otherwise, let $h = (a, b, c, d)$ be the coefficients of the current halfspace.

We want to know the value of t (if any) at which the ray intersects the plane. Plugging in the representation of the ray into the halfspace inequality we have

$$a(p_x + t\vec{u}_x) + b(p_y + \vec{u}_y) + c(p_z + t\vec{u}_z) + d \leq 0,$$

which after some manipulations is

$$t(a\vec{u}_x + b\vec{u}_y + c\vec{u}_z) \leq -(ap_x + bp_y + cp_z + d)$$

If P and $\vec{u}$ are given in homogeneous coordinates, this can be written as

$$t(H \cdot \vec{u}) \leq -(H \cdot P)$$

This is not really a legitimate geometric expression (since dot product should only be applied between vectors). Actually the halfspace H should be thought of as a special geometric object, a sort of *generalized normal vector*. For example, when transformations are applied, normal vectors should be multiplied by the inverse transpose matrix to maintain orthogonality.

We consider three cases.

$(H \cdot \vec{u}) > 0$: in this case we have the constraint

$$t \leq \frac{-(H \cdot P)}{(H \cdot \vec{u})}.$$

Let t^* denote the right-hand side of this inequality. We trim the high-end of the intersection interval to $[t_0, \min(t_1, t^*)]$.

$(H \cdot \vec{u}) < 0$ in this case we have

$$t \geq \frac{-(H \cdot P)}{(H \cdot \vec{u})}.$$

Let t^* denote the right hand side of the inequality. In this case, we trim the low-end of the intersection interval to $[t_0, \min(t_1, t^*)]$.

$(H \cdot \vec{u}) = 0$: In this case the ray is parallel to the plane, either entirely above or below or on. We check the origin. If $(H \cdot P) <= 0$ then the origin lies in (or on the boundary

of) the halfspace, and so we leave the current interval unchanged. Otherwise, the origin lies outside the halfspace, and the intersection is empty. To model this we can set t_1 to any negative value, e.g., -1.

After we repeat this on each face of the polyhedron, we have the following possibilities:

$t_1 < t_0$: In this case the ray does not intersect the polyhedron.

$0 = t_0 = t_1$: In this case, the origin is within the polyhedron. If $t_1 = \infty$, then the polyhedron must be ungrounded (e.g. like a cone) and there is no intersection. Otherwise, the first intersection point is the point $P + t_1\vec{u}$.

$0 < t_0 \leq t_1$: In this case, the origin is outside the polyhedron, and the first intersection is at $P + t_0\vec{u}$.

Reading Assignment and News

Chapter 12 pages 625 - 638, of Recommended Text.

Please also track the News section of the Course Web Pages for the most recent Announcements related to this course. (http://www.cs.utexas.edu/users/bajaj/graphics25/cs354/)