Ray Tracing

Ray Tracing/Casting:

- Basic definitions of terms
- Ray tracing and casting

Intersection Computations:

- In general
- With quadric surfaces
- With polygons

Surface Information:

- Normals to quadric surfaces
- Normals to polygons

Improvements – Speed:

- Spatial subdivision
- Bounding volumes
- Visibility maps

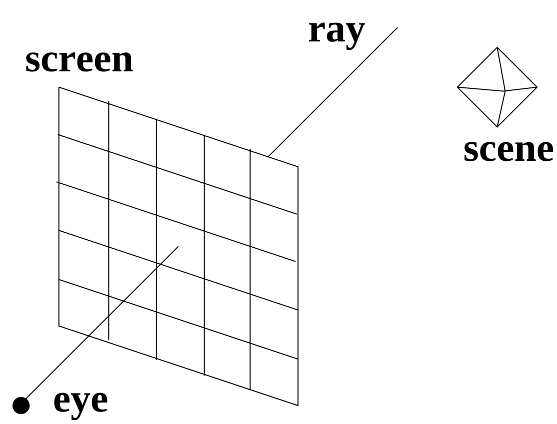
Improvements – Quality:

- Importance-based tracing
- Bidirectional tracing
- Distributed tracing
- Beam and cone tracing

Basic Definitions

Ray Tracing/Casting: Recursive photon tracking vs. visibility probing

- 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

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
    - cancelation
    - 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

Normal Transformations: How do affine transformations affect surface normals?

- Let P_a and P_b be any two points on an object
  - arbitrarily close
  - $\vec{n} \cdot (P_b - P_a) = 0$
  - using transpose notation: $(\vec{n})^T (P_b - P_a) = 0$
- After an affine transformation on P_a, P_b:

$$\mathbf{M}(P_b - P_a) = \mathbf{M}P_b - \mathbf{M}P_a$$

we want $(\mathbf{N}\vec{n})$ to be a normal for some transformation **N**:

$$\begin{aligned} & (\mathbf{N}\vec{n})^T \mathbf{M}(P_b - P_a) = 0 \\ \Longrightarrow \quad & \vec{n}^T \mathbf{N}^T \mathbf{M}(P_b - P_a) \end{aligned}$$

and this certainly holds if $\mathbf{N} = (\mathbf{M}^{-1})^T$

- Only the upper 3-by-3 portion of **M** is pertinent for vectors

- Translation:

$$(\mathbf{M}^{-1})^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\Delta x & -\Delta x & -\Delta x & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix} = \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}$$

$\Longrightarrow$ no change to the normal

- Rotation (example):

$$(\mathbf{M}^{-1})^T = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) & 0 & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) & 0 & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\longrightarrow \quad \begin{bmatrix} \cos(-\theta) & \sin(-\theta) & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\Longrightarrow$ rotation applied unchanged to normal

- Scale:

$$(\mathbf{M}^{-1})^T = \begin{bmatrix} \frac{1}{sx} & 0 & 0 & 0 \\ 0 & \frac{1}{sy} & 0 & 0 \\ 0 & 0 & \frac{1}{sz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{1}{sx} & 0 & 0 \\ 0 & \frac{1}{sy} & 0 \\ 0 & 0 & \frac{1}{sz} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{sx} & 0 & 0 \\ 0 & \frac{1}{sy} & 0 \\ 0 & 0 & \frac{1}{sz} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{nx}{sx} \\ \frac{ny}{sy} \\ \frac{nz}{sz} \\ 0 \end{bmatrix}$$

$\Longrightarrow$ reciprocal scale applied to normal

Surface Information

Surface Normals:

- Illumination models require:
  - surface normal vectors at intersection points
  - ray-surface intersection computation must also yield a normal
  - light-source directions must be established at intersection
  - shadow information determined by light-ray intersections with other objects
- Normals to polygons:
  - provided by planar normal
  - provided by cross product of adjacent edges
- Normals to any implicit surface (e.g., quadrics)
  - move from (x, y, z) to $(x + \Delta x, y + \Delta y, z + \Delta z)$ which is *maximally* far from the surface
  - direction of greatest increase to $f(x, y, z)$

- Taylor series:

$$f(x+\Delta x, y+\Delta y, z+\Delta z) = f(x,y,z) + [\Delta x, \Delta y, \Delta z]\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} + \cdots$$

  - maximality for the *gradient vector* of f
  - not normalized

- Normal to a quadric surface

$$\begin{bmatrix} 2Ax + By + Cz + G \\ Bx + 2Dy + Ez + H \\ Cx + Ey + 2Fz + J \\ 0 \end{bmatrix}$$