Implementation Issues

More from Interface point of view

World Coordinate System (WCS)

Viewing Coordinate System (VCS)
View Coordinate System (VCS)

Viewing coordinate system
- Position and orientation of the view plane
- Extent of the view plane (window)
- Position of the eye

View Plane
- View Reference Point (VRP): the origin of VCS specified as \((r_x, r_y, r_z)\) in WCS: center of the scene
- Normal to the view plane \((n_x, n_y, n_z)\)
View Coordinate System (VCS)

View Plane

- Normal Direction (View Plane Normal VPN) \( n(n_x, n_y, n_z) \)

User may provide normalized vector

e.g.

\[
\begin{align*}
  n_x &= \sin \phi \cos \theta \\
  n_y &= \sin \phi \sin \theta \\
  n_z &= \cos \phi
\end{align*}
\]
View Coordinate System (VCS)

View Plane

- Direction $v$
  
  $v$ is a unit vector intuitively corresponding to “up” vector
  “up” vector is specified by the user in WCS

  $$up' = up - (up.n)n$$

  $$v = \frac{up'}{|up'|}$$

- Direction $u$

  $$u = n \times v \text{ (Left Handed)}$$
Window and Eye

- Window: left, right, bottom, top \((w_l, w_r, w_b, w_t)\) generally is centered at VRP (origin)

- Eye: \(e = (e_u, e_v, e_n)\) Typically \(e = (0, 0, -E)\)
Transformation from WCS to VCS

\[(x, y) = (a \ b) \begin{pmatrix} u \\ v \end{pmatrix} + r = (a \ b)M + r\]
Transformation from WCS to VCS

Point object is represented as
• \((a, b, c)\) in VCS
• \((x, y, z)\) in WCS

\[
M = \begin{bmatrix}
u \\
v \\
n
\end{bmatrix} = \begin{bmatrix}
u_x & u_y & u_z \\
v_x & v_y & v_z \\
n_x & n_y & n_z
\end{bmatrix}
\]
3D Viewing

Transformation from WCS to VCS

Conversion from one coordinate system to another

\[
p = [x \ y \ z] = [a \ b \ c]M + r
\]
\[
[a \ b \ c] = (p - r)M^{-1}
\]
\[
= (p - r)M^T
\]

Therefore \(a = (p - r).u\), \(b = (p - r).v\), \(c = (p - r).n\)
Set up

3D Viewing Interface (Revisit)

Conversion from one coordinate system to another

- \((a,b,c)\) in VCS
- \((x,y,z)\) in WCS

\[
p = \begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix} M + r
\]

\[
\begin{bmatrix} a & b & c \end{bmatrix} = (p - r) M^{-1}
\]

\[
= (p - r) M^T
\]

Where

\[
M = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix}
\]
Set up

Steps
1. Define the view reference point (VRP) = \( r \)
2. Obtain \( M \) defining \( u, v, n \)
3. Define the position of eye (in VCS)
   \[ e = ( e_u, e_v, e_n ) \text{ typically } (0, 0, -E) \]
   corresponding point in WCS will be
   \[ \text{eye} = (0, 0, -E) M + r \]
4. Define the window and the pixel location.
Set up

(rows : 0 to MAXROW
cols : 0 to MAXCOL)

\((i,j)^{th}\) pixel: \((u_i, v_j, 0)\)

\[ u_i = W_l + i \Delta u \]
\[ v_j = W_t - j \Delta v \]

\[ \Delta u = \frac{(W_r - W_l)}{MAXCOL} \]
\[ \Delta v = \frac{(W_t - W_b)}{MAXROW} \]
Set up

5. Parametric equation of ray

\[ P_{ij}(t) = \text{eye} + \text{dir}_{ij} t \]
\[ \text{eye} = (0, 0, -E) M + r \]
\[ \text{dir}_{ij} = ((u_i, v_j, 0) - e) M = (u_i, v_j, E) M \]
\[ P_{ij}(t) = \text{eye} + \text{dir}_{ij} t \]
\[ R(t) = R_o + R_d t \]
Ray Tracing

Set up

- Find intersection with object \( (r_i) \)
- Find the normal at \( r_i \) as \( r_n \)
- Find intensity at \( I \) at the point
  (Illumination model)
- Find the pixel-color
Review

Basic ray tracing (one level): ray casting

Algorithm:
For each pixel shoot a ray from eye (COP)
Compute ray-object(s) intersection
Obtain the closest intersection point \( p \)
Compute normal at \( p \)
Compute illumination (intensity)

Set up:
Obtain ray in WCS for intersection using transformations

Transformation of objects:
Equivalent transformation for the ray
Recursive Ray Tracing
Recursive Ray Tracing
Recursive Ray Tracing

Different Rays

Eye ray (primary ray)
Reflected ray
Transmitted ray
Shadow ray

(secondary rays)
Recursive Ray Tracing

Reflected Ray

Recall Reflection Vector

\[ R = 2(L \cdot N)N - L \]
Recursive Ray Tracing

Refracted Ray

Snell’s Law

\[
\frac{\sin \theta_i}{\sin \theta_t} = \frac{\eta_t}{\eta_i}
\]
Ray Tracing

Recursive Ray Tracing

Refracted Ray

Snell’s Law

\[ M = \frac{I + (\cos \theta_i)N}{\sin \theta_i} \]

\[ T = (\sin \theta_t)M - (\cos \theta_t)N \]

\[ T = \frac{\sin \theta_t(I + (\cos \theta_i)N)}{\sin \theta_i} - (\cos \theta_t)N \]

\[ T = \frac{\eta_i(I + (\cos \theta_i)N)}{\eta_t} - (\cos \theta_t)N \]
Recursive Ray Tracing
Recursive Ray Tracing
Recursive Ray Tracing

Ray Tracing
Recursive Ray Tracing

When to stop?

When ray leaves the scene
When the contribution to the overall intensity is small
Recursive Ray Tracing

Phong Illumination Model

\[ I_{\text{total}} = \text{ambient reflection} + \text{diffuse reflection} + \text{specular reflection} \]
\[ = k_a I_a + k_d I_i \cos \theta + k_s I_i \cos^n \alpha \]
\[ = k_a I_a + k_d I_i (L \cdot N) + k_s I_i (R \cdot V)^n \]
\[ = k_a I_a + \sum_{i=1}^{m} k_d I_i (L_i \cdot N) + k_s I_i (R_i \cdot V)^n \]
Recursive Ray Tracing

Illumination

When in shadow (single light source)

\[ I_{total} = \text{ambient reflection} = k_a I_a \]
Recursive Ray Tracing

Illumination

With reflection and transmission rays

\[ I_{\text{total}}(P) = I_{\text{local}}(P) + k_{rg} I(P_r) + k_{tg} (P_t) \]

Global Illumination
Recursive Ray Tracing

T Whitted 1980
Recursive Ray Tracing

Other Example
Recursive Ray Tracing

Other features

• Ray tracing is an image based method (pixelization)
• Sampling
  “aliasing”
  Jagginess
  Moire patterns
• Anti-aliasing
Recursive Ray Tracing

Anti-alising

- Supersampling
- More number of rays per pixel
- Average the result