Ray tracing

- simple ray casting
- recursive ray tracing
- modeling transforms
- cheap tricks
- optimizations

Ray tracing

- cast ray through pixel into scene
- find closest intersection (if any)
- compute luminance at intersection
  - direct illumination
  - reflections
  - refraction

Specular reflections

- cast ray reflected at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by msr[g,b] and add to luminance at P

Transmission

- cast ray transmitted at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by ktrans and add to luminance at P

Transmission

- cast ray transmitted at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by ktrans and add to luminance at P

Refraction - Snell's Law

- incoming ray (P_0, v)
- transmitted ray (P, v')
thin surface refraction

ignore Snell’s law

incoming ray \((P_0, v)\)

\(P\)

transmitted ray \((P, v)\)

2/23/2003  90

thick surface refraction

\(\theta_{\text{in}}\) satisfies: \(n_{\text{out}} \sin \theta_{\text{out}} = n_{\text{in}} \sin \theta_{\text{in}}\)

\(v_{\text{out}} = (\cos \theta_{\text{in}} - 1 - \beta \sin^2 \theta_{\text{in}}) n + \beta v_{\text{in}}\) where \(\beta = n_{\text{in}} / n_{\text{out}}\)

2/23/2003  91

thick surface recursion

What is direct illumination at \(P'\)?

2/23/2003  92

stopping conditions

recurse until:

a) maximum recursive depth specified by user is reached

b) contribution to luminance is less than user specified bound

2/23/2003  93

implementation issues

offset new ray slightly to make sure you don’t find \(P\) again!!!

• cast new ray from \(P\) into scene

• find closest intersection point \(P'\) (if any)

• compute luminance at \(P'\)

• scale and add to luminance at \(P\)

2/23/2003  94

stopping conditions

recurse until:

a) maximum recursive depth specified by user is reached

b) contribution to luminance is less than user specified bound

• cast new ray from \(P\) into scene

• find closest intersection point \(P'\) (if any)

• compute luminance at \(P'\)

• scale by msr/g/b or ktrans and add to luminance at \(P\)

2/23/2003  95
Ray Tracing

- simple ray casting
- recursive ray tracing
- modeling transforms
- cheap tricks
- optimizations

Modeling Transforms

- \[ M p = p' \]
- Object description: \( M, p \) or \( p' \)?

Ray Tracing

- Cast ray into scene
- Find intersection point (if any) that is closest to eye
- Compute luminance at intersection

Find Intersection Point

- Sphere
- Viewpoint
- Squashed (aka transformed) sphere

Find Intersection Point

- \( R' \)
- Viewpoint
- Object coordinates

- \( R \)
- Viewpoint
- World coordinates
does this make sense?

- is there an inverse transform?
- how do we apply a transform to a ray?
- is a ray in world coordinates a ray in object coordinates?

Conceptually: scale

What operation inverts a scale by $s$ in the $x$-direction?

For $s \neq 0$, scale by $1/s$ in the $x$-direction.

Any problem?

We are not alone...

The parallel universe view of homogenous coordinates

We live in this universe. It's not the only one, but it is the only one we can experience!

scale

$$
\begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & t^* & 0 & 0 \\
0 & 0 & u^* & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
u \\
u \\
\end{bmatrix}
= ?
$$

Conceptually: rotate

What operation inverts a rotate by $\theta$ about the $x$-axis?

Rotate by $-\theta$ about the $x$-axis.
rotate about z axis

\[
\begin{bmatrix}
\cos \phi & -\sin \phi & 0 & 0 \\
\sin \phi & \cos \phi & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \Rightarrow \\
\begin{bmatrix}
\cos \phi & -\sin \phi & 0 & 0 \\
\sin \phi & \cos \phi & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

remember \( \cos(-\phi) = \cos(\phi) \) and \( \sin(-\phi) = -\sin(\phi) \)

Conceptually: translate

What operation inverts a translate by \( dx \) in the x-direction?

Translate by \(-dx\) in the x-direction.

\[
\begin{bmatrix}
1 & 0 & 0 & -x_0 \\
0 & 1 & 0 & -y_0 \\
0 & 0 & 1 & -z_0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix} \Rightarrow \\
\begin{bmatrix}
1 & 0 & 0 & x_0 \\
0 & 1 & 0 & y_0 \\
0 & 0 & 1 & z_0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

does this make sense?

- is there an inverse transform?
- how do we apply a transform to a ray?
- is a ray in world coordinates a ray in object coordinates?

transform

- Points  Done!!
- Vectors
- Rays
transforms: vector

\[ \mathbf{v} = \mathbf{p} - \mathbf{q} \quad \text{and} \quad T(\mathbf{v}) = T(\mathbf{p} - \mathbf{q}) = T(\mathbf{p}) - T(\mathbf{q}) = M\mathbf{p} - M\mathbf{q} \]

Warnings:
- because of translation we can’t ignore \( \mathbf{q} = (0,0,0) \)
- re-unitize unit vectors

transforms

- Points: Done!!
- Vectors: Done!!
- Rays

transforms: ray

- Points
- Vectors
- Rays: \( \mathbf{r} = (\mathbf{p}, \mathbf{v}) \)

Transform point and transform/unitize vector!

does this make sense?

- is there an inverse transform?
- how do we apply a transform to a ray?
- is a ray in world coordinates a ray in object coordinates?

Linear transforms

Linear transforms preserve lines!

find intersection point

Linear transforms preserve lines!
**M and M**⁻¹**1**

**single transform**

\[ M \]
- scale by \( s \)
- rotate by \( \theta \)
- translate by \( \Delta \)

\[ M^{-1} \]
- scale by \( 1/s \)
- rotate by \(-\theta\)
- translate by \(-\Delta\)

**composite transform**

\[ (M_kM_{k-1}...M_2M_1)^{-1} \]
\[ M_k^{-1}...M_2^{-1}M_1^{-1} \]

---

**scene graph traversal**

A sends the ray (represented relative to A's coordinate system) to B.

**scene graph traversal**

A sends the ray \( R_A \) to B.

B converts the ray into its own coordinate system

\[ R_B = T(R_A) \]

**scene graph traversal**

B computes the intersections of \( R_B \) with its objects

**scene graph traversal**

- B sends \( R_B \) to its descendants
- Each returns intersection information (represented in B's coordinate system)

**surface normal**

is the normal to a transformed surface the transformed normal?
Surface normal

Is the tangent plane to a transformed surface the transformed tangent plane?

The right way ...

N is normal to the tangent plane iff for any points p and q on the tangent plane N \cdot (p-q) = 0.

\[ \text{Assume N is normal to the tangent plane and QN is normal to the tangent plane transformed by M.} \]
\[ \text{Q must satisfy the following for any points p and q on the tangent plane:} \]
\[ N^T(p-q)=0 \iff (QN)^T(M(p-q))=0 \]
\[ N^T(p-q)=0 \iff N^T(Q^T M(p-q))=0 \]
\[ \text{Thus Q = (M^{-1})^T} \]

Ray tracing

- simple ray casting
- recursive ray tracing
- modeling transforms
- cheap tricks
- optimizations