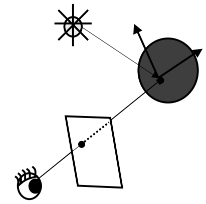
ray tracing

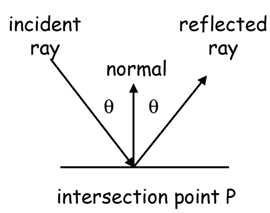
- simple ray casting
- **recursive ray tracing**
- modeling transforms
- cheap tricks
- optimizations

ray tracing



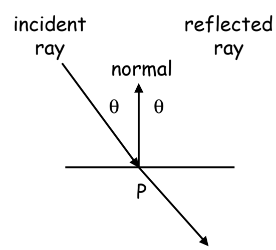
- cast ray through pixel into scene
- find closest intersection (if any)
- compute luminance at intersection
  - direct illumination
  - **reflections**
  - **refraction**

specular reflections



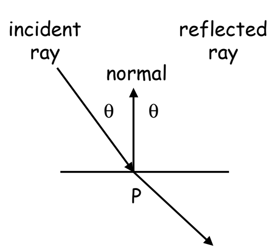
- cast ray reflected at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by msr[g,b] and add to luminance at P

transmission



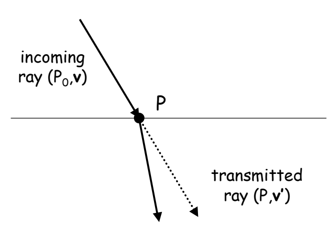
- cast ray transmitted at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by k_{trans} and add to luminance at P

transmission



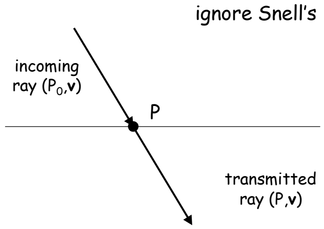
- **cast ray transmitted at P into scene**
- find closest intersection point P' (if any)
- compute luminance at P' point
- scale by k_{trans} and add to luminance at P

refraction – snell's law

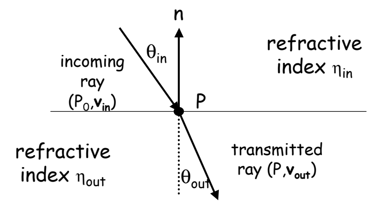


thin surface refraction

ignore Snell's law



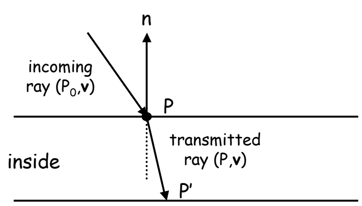
thick surface refraction



θ_{out} satisfies: $\eta_{out} \sin \theta_{out} = \eta_{in} \sin \theta_{in}$

$\mathbf{v}_{out} = [\beta\cos\theta_{in} - (1-\beta^2\sin^2\theta_{in})^{\frac{1}{2}}]\,\mathbf{n} + \beta\mathbf{v}_{in}$ where $\beta=\eta_{in}/\eta_{out}$

thick surface recursion



What is direct illumination at P'?

stopping conditions

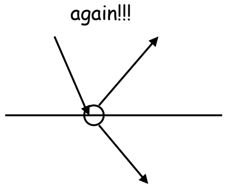
recurse until:

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 and add to luminance at P

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

stopping conditions

recurse until:

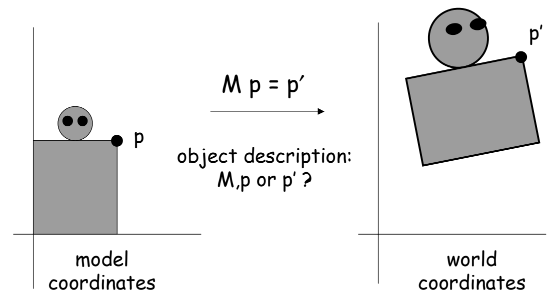
a) maximum recursive depth specified by user is reached
b) contribution to luminance is less than user specified bound

- 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

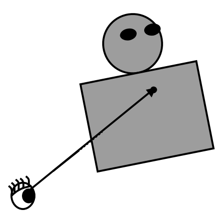
ray tracing

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

modeling transforms

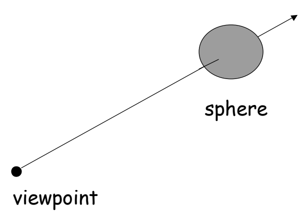


ray tracing

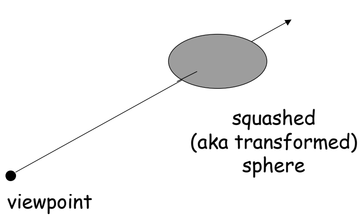


- cast ray into scene
- **find intersection point (if any) that is closest to eye**
- compute luminance at intersection

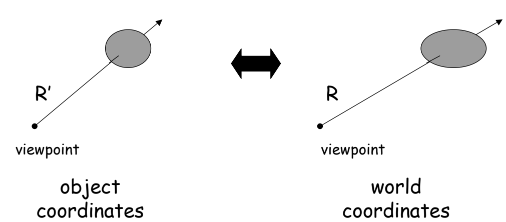
find intersection point



find intersection point



find intersection point



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?

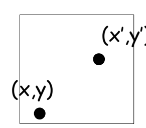
For $s \neq 0$, scale by 1/s in the x-direction.

Any problem?

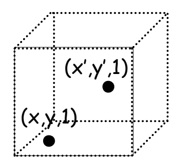
We are not alone!

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

$$\begin{pmatrix} s^{-1} & 0 & 0 & 0 \\ 0 & t^{-1} & 0 & 0 \\ 0 & 0 & u^{-1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = ?$$

Conceptually: rotate

What operation inverts a rotate by θ about the x-axis?

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

rotate about z axis

$$\begin{pmatrix} \cos -\phi & -\sin -\phi & 0 & 0 \\ \sin -\phi & \cos -\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \phi & -\sin \phi & 0 & 0 \\ \sin \phi & \cos \phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =?$$

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

rotate about z axis

$$\begin{pmatrix} \cos \phi & \sin \phi & 0 & 0 \\ -\sin \phi & \cos \phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \phi & -\sin \phi & 0 & 0 \\ \sin \phi & \cos \phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =?$$

remember $\cos^2 \phi + \sin^2 \phi = 1$

Conceptually: translate

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

Translate by -dx in the x-direction.

translate

$$\begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =?$$

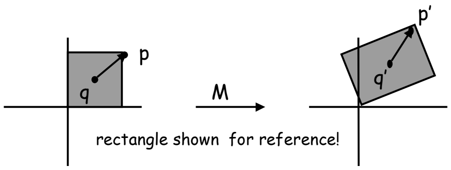
does this make sense?

- 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

tranforms: vector



v=p-q and T(**v**)=T(p-q)=T(p)-T(q) = Mp-Mq

Warnings:

- because of translation we can't ignore q=(0,0,0)
- re-unitize unit vectors

transforms

- Points Done!!
- Vectors Done!!
- **Rays**

transforms: ray

- Points
- Vectors
- **Rays: r=(p,v)**

Transform point
and
transform/unitize
vector!

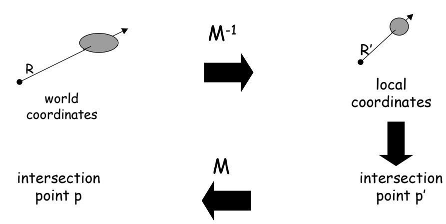
does this make sense?

- 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



M and M^{-1}

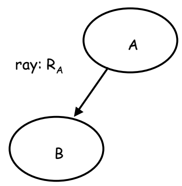
single transform

M	M^{-1}
– scale by s	– scale by 1/s
– rotate by θ	– rotate by $-\theta$
– translate by Δ	– translate by $-\Delta$

composite transform

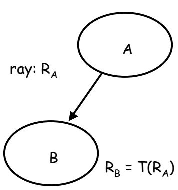
$(M_1M_2 \ldots M_k)^{-1}$ $\quad$ $M_k^{-1} \ldots M_2^{-1}M_1^{-1}$

scene graph traversal



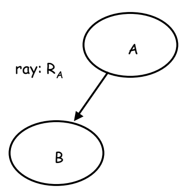
A sends the ray (represented relative to A's coordinate system) to B.

scene graph traversal



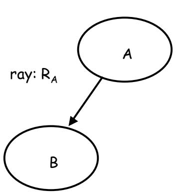
B converts the ray into its own coordinate system

scene graph traversal



B computes the intersections of R_B with its objects

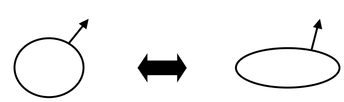
scene graph traversal



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

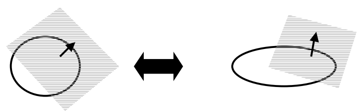
surface normal

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 $N^T \bullet (p-q) = 0$.

The right way …

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

Assume N is normal to the tangent plane and QN is normal to the tangent plane transformed by M.

Q must satisfy the following for any points p and q on the tangent plane:

$$N^T(p-q)=0 \text{ iff } (QN)^T(M(p-q))=0$$

$$\Updownarrow$$

$$N^T(p-q)=0 \text{ iff } N^T(Q^T M)(p-q)=0$$

Thus $Q=(M^{-1})^T$

ray tracing

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