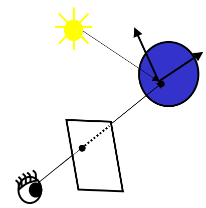
ray tracing

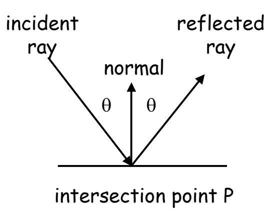
- simple ray casting
- **recursive ray tracing**
- modeling transforms
- cheap tricks
- optimizations

ray tracing



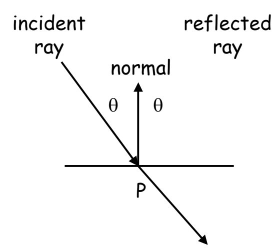
- cast ray through pixel into scene
- find closest intersection (if any)
- compute luminance at intersection
  - direct illumination
  - **reflections**
  - **refraction**

specular reflections



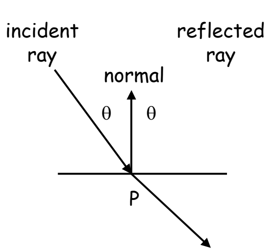
- cast ray reflected at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by msr[g,b] and add to luminance at P

transmission



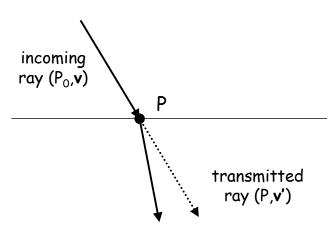
- cast ray transmitted at P into scene
- find closest intersection point P' (if any)
- compute luminance at P'
- scale by k_{trans} and add to luminance at P

transmission



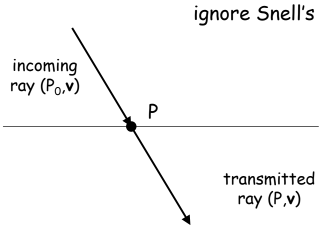
- **cast ray transmitted at P into scene**
- find closest intersection point P' (if any)
- compute luminance at P' point
- scale by k_{trans} and add to luminance at P

refraction – snell's law

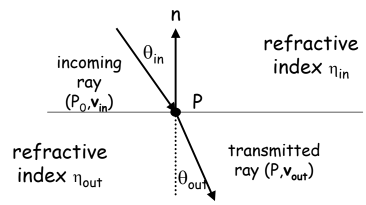


thin surface refraction

ignore Snell's law



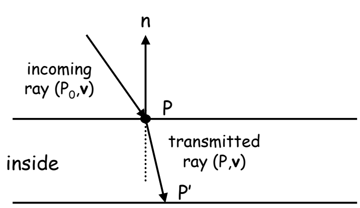
thick surface refraction



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

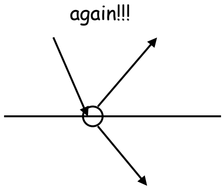
recurse until:

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:

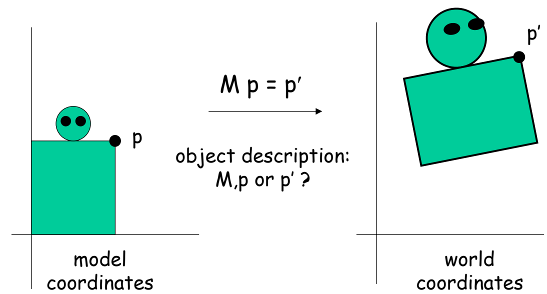
a) maximum recursive depth specified by user is reached
b) contribution to luminance is less than user specified bound

- 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

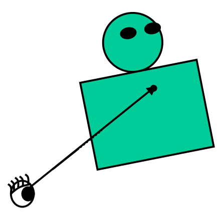
ray tracing

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

modeling transforms

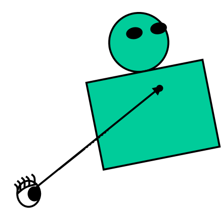


ray tracing



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

intersection



1. find intersection with transformed primitive Hard!

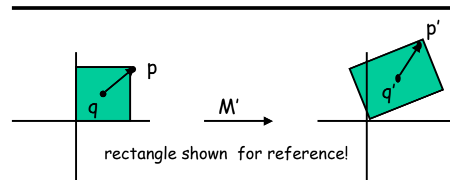
OR

2. convert ray to local coordinate system & test for intersection with (non-transformed) primitive

transform

- Points Done!!
- Vectors
- Rays

tranforms: vector



v=p-q and M'**v**=M'(p-q)=M'p-M'q

Warnings:

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

transforms

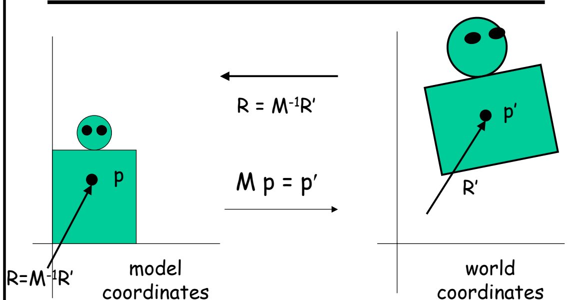
- Points
- Vectors
- **Rays**

transforms: ray

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

Transform point and transform/unitize vector!

modeling transforms



M and M^{-1}

single transform

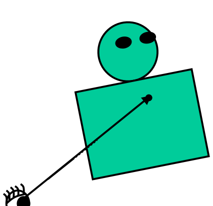
M
- scale by s
- rotate by θ
- translate by Δ

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

composite transform

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

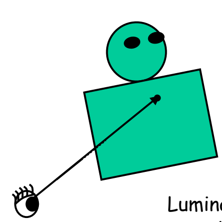
ray tracing



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

compute distance to eye in world coordinates

ray tracing



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

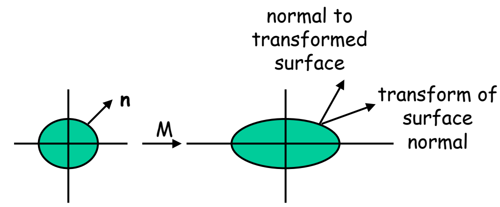
Luminance depends on geometry so compute in world coordinates. Alas, need intersection point and surface normal!

transforms

- Points
- Vectors
- Rays
- Surface normal ←

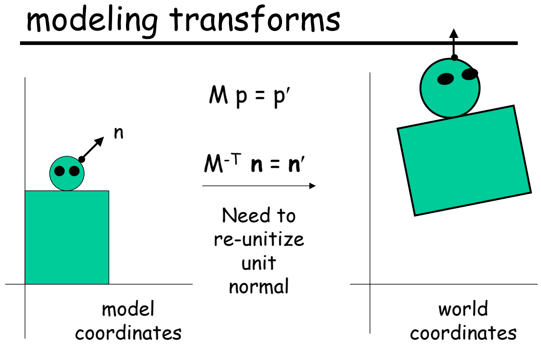
Normal of a transformed surface not the transformed surface normal!

transforms: surface normal



WHY? Because it works!

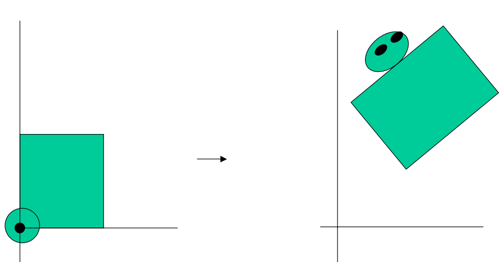
modeling transforms



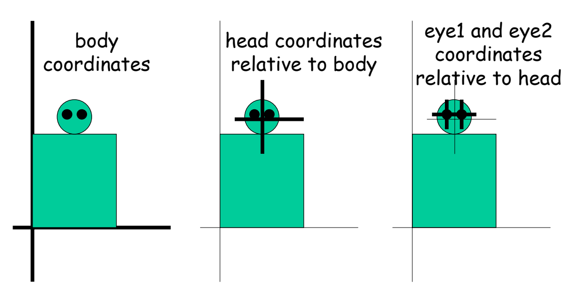
Object specification

- World coordinates
- Object coordinates and modeling transform
- Hierarchical coordinates

transform composition



hierarchical coordinates



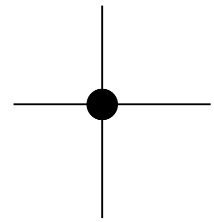
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        eye1 translate wrt head
        eye1 scale
        eye1 description

order of operations

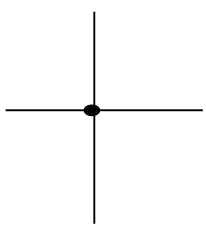
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        {eye1 translate wrt head
        eye1 scale
        eye1 description}
        {eye2 translate wrt head
        eye2 scale
        eye2 description}



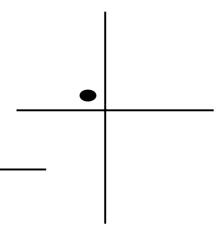
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        {eye1 translate wrt head
        eye1 scale
        eye1 description}
        {eye2 translate wrt head
        eye2 scale
        eye2 description}



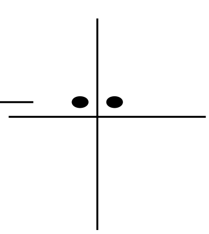
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        {eye1 translate wrt head
        eye1 scale
        eye1 description}
        {eye2 translate wrt head
        eye2 scale
        eye2 description}



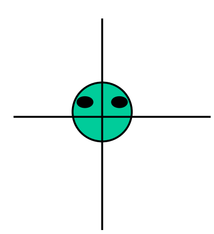
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        {eye1 translate wrt head
        eye1 scale
        eye1 description}
        {eye2 translate wrt head
        eye2 scale
        eye2 description}



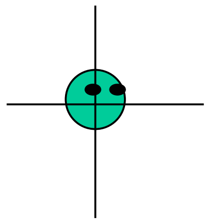
hierarchical coordinates

body xfm
body description
    head translate wrt body
    head rotate
    head description
        {eye1 translate wrt head
        eye1 scale
        eye1 description}
        {eye2 translate wrt head
        eye2 scale
        eye2 description}



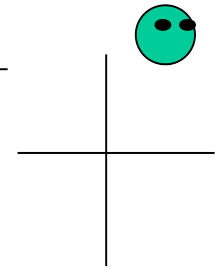
hierarchical coordinates

body xfm
body description
- head translate wrt body
- head rotate
- head description
  - {eye1 translate wrt head
  - eye1 scale
  - eye1 description}
  - {eye2 translate wrt head
  - eye2 scale
  - eye2 description}



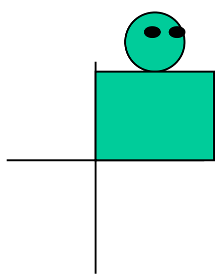
hierarchical coordinates

body xfm
body description
- head translate wrt body
- head rotate
- head description
  - {eye1 translate wrt head
  - eye1 scale
  - eye1 description}
  - {eye2 translate wrt head
  - eye2 scale
  - eye2 description}



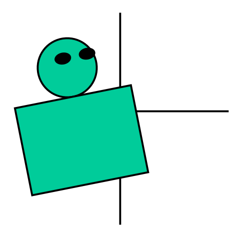
hierarchical coordinates

body xfm
body description
- head translate wrt body
- head rotate
- head description
  - {eye1 translate wrt head
  - eye1 scale
  - eye1 description}
  - {eye2 translate wrt head
  - eye2 scale
  - eye2 description}

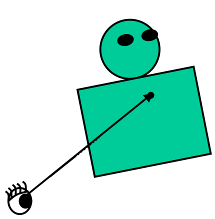


hierarchical coordinates

body xfm
body description
- head translate wrt body
- head rotate
- head description
  - {eye1 translate wrt head
  - eye1 scale
  - eye1 description}
  - {eye2 translate wrt head
  - eye2 scale
  - eye2 description}

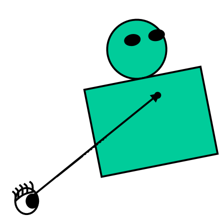


ray tracing



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

intersection



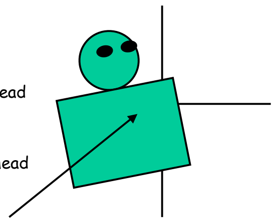
1. find intersection with transformed primitive

OR

2. convert ray to local coordinate system & test for intersection with (non-transformed) primitive

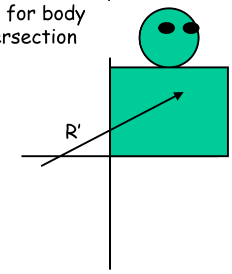
intersection

body xfm
body description
head translate wrt body
head rotate
head description
{eye1 translate wrt head
eye1 scale
eye1 description}
{eye2 translate wrt head
eye2 scale
eye2 description}



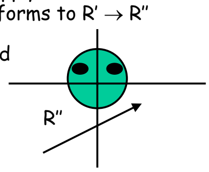
intersection

body xfm ← Apply inverse transform to ray
body description ← Test for body intersection
head translate wrt body
head rotate
head description
{eye1 translate wrt head
eye1 scale
eye1 description}
{eye2 translate wrt head
eye2 scale
eye2 description}



hierarchical coordinates

body xfm Apply inverse transform to R → R'
body description
head translate wrt body
head rotate
} Apply inverse transforms to R' → R''
head description ← Test head
{eye1 translate wrt head
eye1 scale
eye1 description}
{eye2 translate wrt head
eye2 scale
eye2 description}



ray tracing

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