Computing Normals w/out Polygonization or Cross-products

you'll need this come ray tracing time

Normal Vectors at Intersection Points (1/4)

Normal vector to implicit surfaces

If a surface bounds a solid whose interior is given by $f(x, y, z) < 0$

then can find normal vector at point (x, y, z) via *gradient* at that point:

$$\boldsymbol{n} = \nabla f(x, y, z)$$

Recall that the gradient is a vector with three components, the partial derivatives:

$$\nabla f(x, y, z) = \left(\frac{\partial f}{\partial x}(x, y, z), \frac{\partial f}{\partial y}(x, y, z), \frac{\partial f}{\partial z}(x, y, z) \right)$$

Normal Vectors at Intersection Points (2/4)

Sphere normal vector example

For the unit sphere, the implicit equation is

$$f(x,y,z) = x^2 + y^2 + z^2 - 1$$

The partial derivatives are

$$\frac{\partial f}{\partial x}(x, y, z) = 2x$$
$$\frac{\partial f}{\partial y}(x, y, z) = 2y$$
$$\frac{\partial f}{\partial z}(x, y, z) = 2z$$

So the gradient is

$$\boldsymbol{n} = \nabla f(x, y, z) = (2x, 2y, 2z)$$

Normalize ***n*** before using in dot products!

In some degenerate cases, the gradient may be zero, and this method fails…use nearby gradient as a cheap hack

Normal Vectors at Intersection Points (3/4)

Transforming back to world space

Have an object space normal vector

Want a world space normal vector for lighting equation

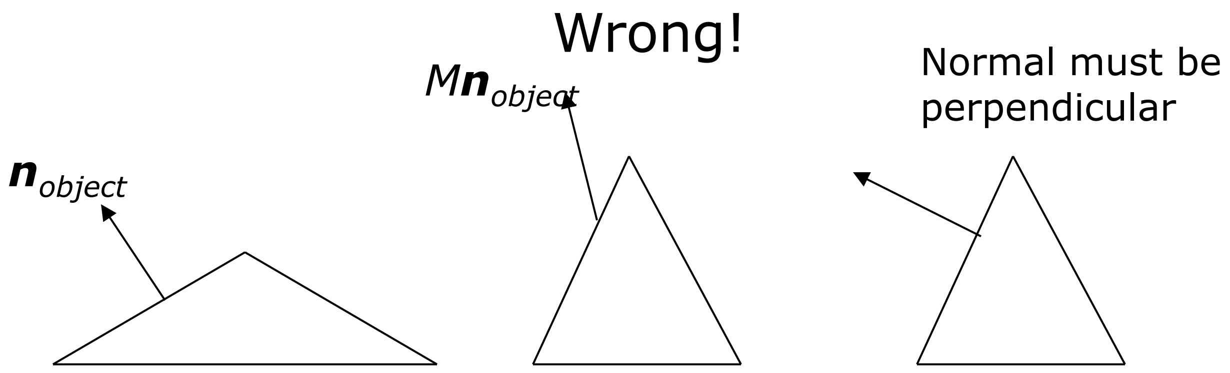
To transform object to world coordinates, we just multiplied its vertices by *M*, the object's CTM

Can we treat the normal vector the same way?

Answer: NO

$$\boldsymbol{n}_{world} \neq M\boldsymbol{n}_{object}$$

Example: say *M* scales in *x* by .5 and *y* by 2


Wrong!
$M\boldsymbol{n}_{object}$
Normal must be perpendicular
$\boldsymbol{n}_{object}$

See the normal scaling applets in *Applets → Lighting and Shading*

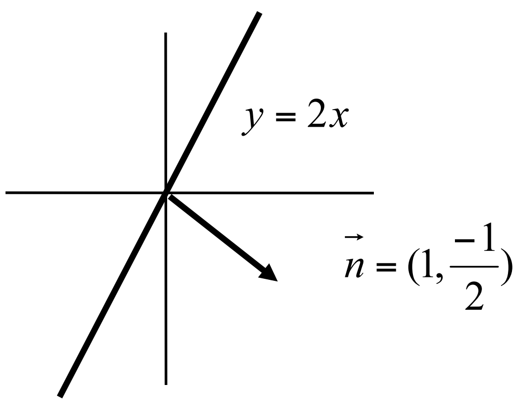
Normal Vectors at Intersection Points (4/4)

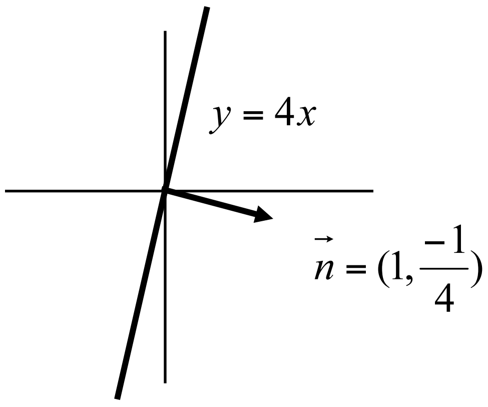
Why doesn't multiplying the normal work?

For translation and rotation, which are rigid body transformations, it actually does work

Scale, however, distorts the normal in exactly the opposite sense of the scale applied to the surface

- scaling y by 2 causes normal to scale by .5:





See this algebraically in the Appendix

Appendix: Transforming Normals

Transforming Normals (1/4)

Object-space to world-space

Take polygonal case, for example

Let's compute relationship between object-space normal $\boldsymbol{n}_{obj}$ to polygon H and world-space normal $\boldsymbol{n}_{world}$ to transformed version of H, called MH

For any vector $\boldsymbol{v}$ in world space that lies in polygon (e.g., one of edge vectors), normal is perpendicular to $\boldsymbol{v}$:

$$\boldsymbol{n}_{world} \bullet \boldsymbol{v}_{world} = 0$$

But $\boldsymbol{v}_{world}$ is just a transformed version of some vector in object space, $\boldsymbol{v}_{obj}$. So we could write

$$\boldsymbol{n}_{world} \bullet M\boldsymbol{v}_{obj} = 0$$

Recall that since vectors have no position they are unaffected by translations (they have $w = 0$) so to make things easier, we can consider only

$$M_3 = \text{upper left 3 x 3 of } M$$

$$\mathbf{n}_{world} \bullet M_3\, \mathbf{v}_{obj} = 0$$

(rotation/scale component)

Transforming Normals (2/4)

Object-space to world-space(cont.)

So we want a vector $\boldsymbol{n}_{\boldsymbol{world}}$ such that for any $\boldsymbol{v}_{\boldsymbol{obj}}$ in the plane of the polygon

$$\boldsymbol{n}_{world} \bullet M_3 \boldsymbol{v}_{obj} = 0$$

We will show on next slide that this equation can be rewritten as

$$M_3^t \boldsymbol{n}_{world} \bullet \boldsymbol{v}_{obj} = 0$$

We also already have

$$\boldsymbol{n}_{obj} \bullet \boldsymbol{v}_{obj} = 0$$

Therefore

$$M_3^t \boldsymbol{n}_{world} = \boldsymbol{n}_{obj}$$

Left-multiplying by $(M_3^t)^{-1}$,

$$\boldsymbol{n}_{world} = \left(M_3^t\right)^{1} \boldsymbol{n}_{obj}$$

Transforming Normals (3/4)

Object-space to world-space(cont.)

So how did we rewrite this:

$$\boldsymbol{n}_{world} \bullet M_3 \boldsymbol{v}_{obj} = 0$$

As this:

$$(M_3{}^t \boldsymbol{n}_{world}) \bullet \boldsymbol{v}_{obj} = 0$$

Recall that if we think of vector as *n* x 1 matrices, then switching notation,

$$\boldsymbol{a} \bullet \boldsymbol{b} = \boldsymbol{a}^t \boldsymbol{b}$$

Rewriting our original formula, we thus have

$$\boldsymbol{n}^t{}_{world} M_3 \boldsymbol{v}_{obj} = 0$$

Writing $M = M^{tt}$, we get

$$\boldsymbol{n}^t{}_{world} M_3{}^{tt} \boldsymbol{v}_{obj} = 0$$

Recalling that $(AB)^t = B^t A^t$, we can write this:

$$(M_3{}^t \boldsymbol{n}_{world})^t \boldsymbol{v}_{obj} = 0$$

Switching back to dot product notation, our result:

$$(M_3{}^t \boldsymbol{n}_{world}) \bullet \boldsymbol{v}_{obj} = 0$$

Transforming Normals (4/4)

Applying inverse-transpose of M to normals

So we ended up with

$$\boldsymbol{n}_{world} = (M_3^{\,t})^{-1}\boldsymbol{n}_{obj}$$

"Invert" and "transpose" can be swapped, to get our final form:

$$\boldsymbol{n}_{world} = (M_3^{\,-1})^{t}\boldsymbol{n}_{obj}$$

Why do we do this? It's easier!

- Instead of inverting composite matrix, accumulate composite of inverses which are easy to take for each individual transformation

A hand-waving interpretation of $(M_3^{-1})^t$

- M_3 is composition of rotations and scales, R and S (why no translates?). Therefore

  $$((RS\ldots)^{-1})^t = (\ldots S^{-1}R^{-1})^t = ((R^{-1})^t(S^{-1})^t\ldots)$$

- so we're applying transformed (inverted, transposed) versions of each individual matrix in original order
- for rotation matrix, transformed version equal to original rotation, i.e., normal rotates with object
  - $(R^{-1})^t = R$; inverse reverses rotation, and transpose reverses it back
- for scale matrix, inverse inverts scale, while transpose does nothing:
  - $(S(x,y,z)^{-1})^t = S(x,y,z)^{-1}=S(1/x,1/y,1/z)$