1 Making Shapes

We have the Sphere class. You can tell the constructor how many lines of latitude you want and how many lines of longitude. As I showed in class and the current assignment, if you set these to 2 and 4 respectively, you get a cube-shaped (or at least box-shaped), object. If you set them to 2 and, say, 16, you get a cylinder. So the Sphere class can do cubes and cylinders too!

Furthermore, you can apply a transformation to the Sphere. In the homework, you scale each coordinate equally and get a smaller (or larger) cube. Suppose you scale x by more than y and z. Then you can get a long box or cylinder or a oval shape. Scaling means making model2object a scale matrix like Mat.scale(new PV(1, 2, 3, false)).

2 Transforming Normals

However, when you apply a general transformation, you have to be careful about what happens to the normals. If you don't compute the normals correctly, you won't get the right shading.

Suppose we apply model2world transformation M. A face has vertex a and contains point p and has normal vector n. Points a and p transform by M:

$$a' = Ma, \qquad p' = Mp.$$

Point p lies in the face if and only if vector $p - a$ is perpendicular to n. So the plane constraint is

$$n \cdot (p - a) = 0$$

where $\cdot$ is the dot product. The dot product is zero if and only if the vectors are perpendicular.

But we can do the dot product using matrix multiplication:

$$\begin{aligned} n \cdot (p - a) &= n_x(p_x - a_x) + n_y(p_y - a_y) + n_z(p_z - a_z), \\ &= \begin{bmatrix} n_x & n_y & n_z \end{bmatrix} \begin{bmatrix} p_x - a_x \\ p_y - a_y \\ p_y - a_y \end{bmatrix}, \\ &= n^T(p - a). \end{aligned}$$

In the world coordinate system,

$$\begin{aligned} n'^T(p' - a') &= 0, \\ n'^T(Mp - Ma) &= 0, \\ n'^T M(p - a) &= 0. \end{aligned}$$

But $(AB)^T = B^T A^T$, so

$$n'^T M = n'^T M^{T^T} = (M^T n')^T.$$

So the equation of the plane is

$$(M^T n')^T (p - a).$$

But plane equations are unique, up to a constant, so

$$\begin{aligned} M^T n' &= n, \\ n' &= M^{T^{-1}} n, \\ n' &= M^{-1^T} n. \end{aligned}$$

So instead of multiplying n by M, you have to multiply it by the transpose of the inverse.

What about the current assignment? I am having you multiply it by M. Well, in this case M is a rotation (the translation doesn't affect a vector), and the transpose of the inverse of a rotation is the same rotation back again! We are also scaling, but since we normalize n, equal scaling of all coordinates doesn't matter. But when we scale x, y, and z by different amounts, we have to use the correct formula.

3 Intersecting a ray with a plane

In the next assignment, we are going to want to be able to click on individual parts of a robot and drag them. How will we know which part we are clicking?

Given a point q and vector v, how can we tell if the ray $q + sv$, $s > 0$, intersects a face?

The face has vertex a and normal n, which we now know how to calculate in world coordinates. The intersection satisfies,

$$n \cdot (q + sv - a) = 0, \qquad sn \cdot v + n \cdot (q - a) = 0, \qquad s = -\frac{n \cdot (q - a)}{n \cdot v}.$$

If $s > 0$, then the ray intersects the plane of the face at $p = q + sv$ for that value of s.

Does this lie on face itself? If p lies inside the face, then $v_i - p$ to $v_{i+1} - p$ is counterclockwise for each i. So $(v_i - p) \times (v_{i+1} - p)$ points in the same direction as n (not opposite) by the right hand rule. So,

$$n \cdot ((v_i - p) \times (v_{i+1} - p)) > 0.$$

However, if this is negative for any i, then p is outside. So just check to make sure that it is positive for each i. By the way, when I say $i + 1$, I mean $i + 1$ mod m where m is the number of vertices of the face. So if the face has five vertices, $i, i + 1$ is $0, 1,\ 1, 2,\ 2, 3,\ 3, 4,\ 4, 0$.

But what if the ray hits multiple faces? Which one should we drag? Answer: the closest one. Which one is closest? Answer: the one with the smallest s.