AS5 - CS 411

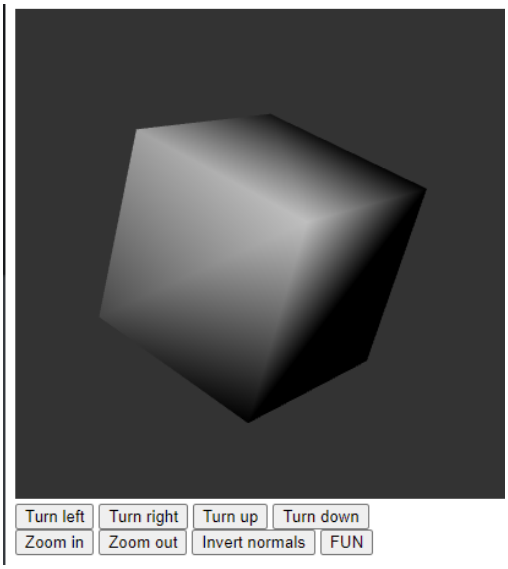
Reflection

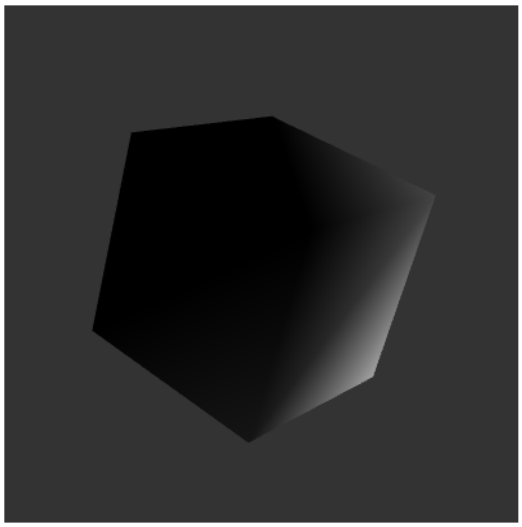
This assignment was very challenging, especially reading through the write up the first few times. After watching the lecture recording again, I realized that the key change I need to make is to translate the algorithm from the last bullet point into code.

After my first attempt, I realized that my normals were most likely incorrect. I calculated them out by hand, and realized that my code was not updating the correct vertices. After that, things still looked weird, so I implemented code to traverse the entire array of normals and normalize them. I didn't think I had to do this, because I had used the calcNormal code from the provided lib, which included normalization on each normal, but this was actually wrong, and I removed it. Adding rotation/zooms was very easy, I simply updated the matrix in drawScene.

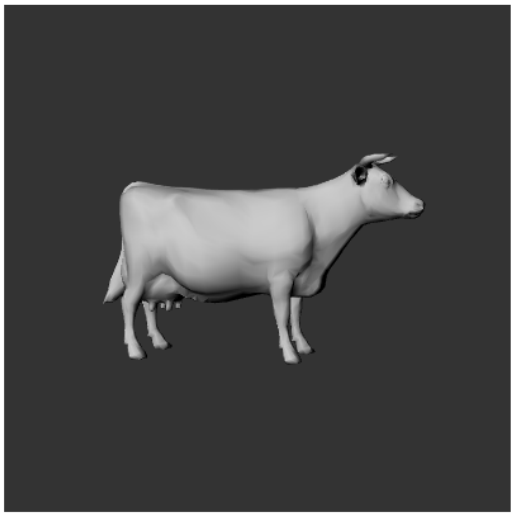
Demo

The fun button does nothing. Removed from final code.





Turn left | Turn right | Turn up | Turn down
Zoom in | Zoom out | Invert normals | FUN



Turn left | Turn right | Turn up | Turn down
Zoom in | Zoom out | Invert normals | FUN

1.1 Explain what are the parameters used to specify a general rotation in 3d. Explain why we use 4 x 4 matrices to specify 3d rotations and explain the advantages of using homogeneous coordinates.

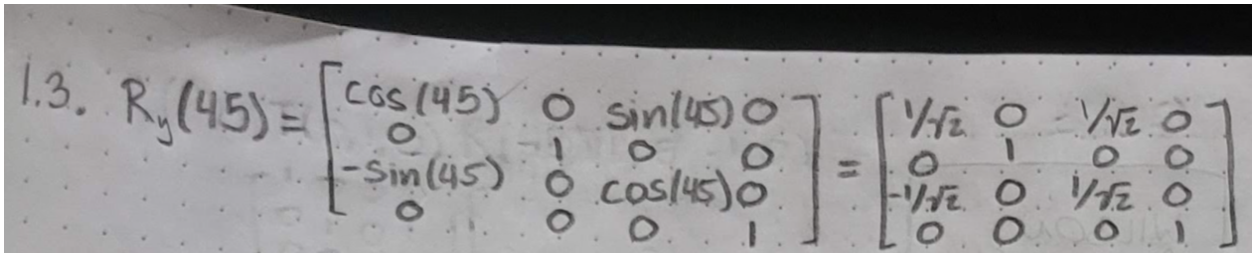
To do a general rotation, you need a vector to rotate about, and an angle theta. One approach is to model Ru(theta) by using a combination of Rx, Ry, and Rz. We use 4x4 matrices to specify 3d rotations because it allows us to use homogeneous coordinates, which again in turn allow us to use matrix operations to do transformations.

1.2 Write the 3d transformation matrix that is necessary for scaling by sx,sy,sz about a point p.

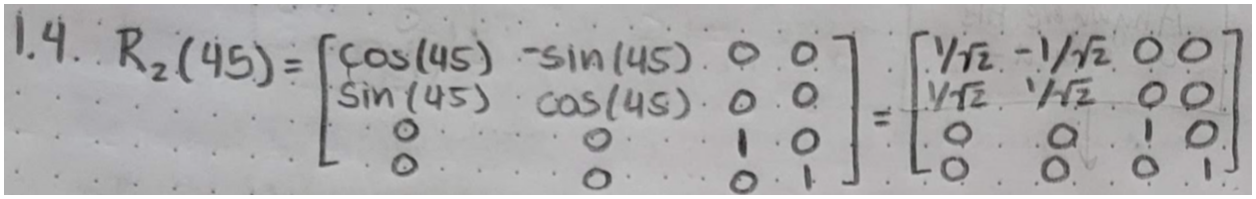
To scale about a point p, you must first translate the object so that p will center about the origin, then scale the object, and translate the object back. If the starting matrix is A, and the resultant matrix is M, the transformation matrix looks like:

M = T(-px,-py,-pz)S(sx,sy,sz)T(px,py,pz)

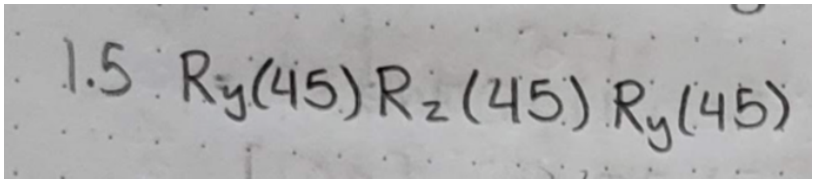
1.3 Write the transformation matrix for rotating a 3D point in homogeneous coordinates by 45° about a rotation axis given by (0, 1,0).



1.4 Write the transformation matrix for rotating the vector (1, 1, 0) to become aligned with the y-axis (0,1, 0) in homogeneous coordinates.



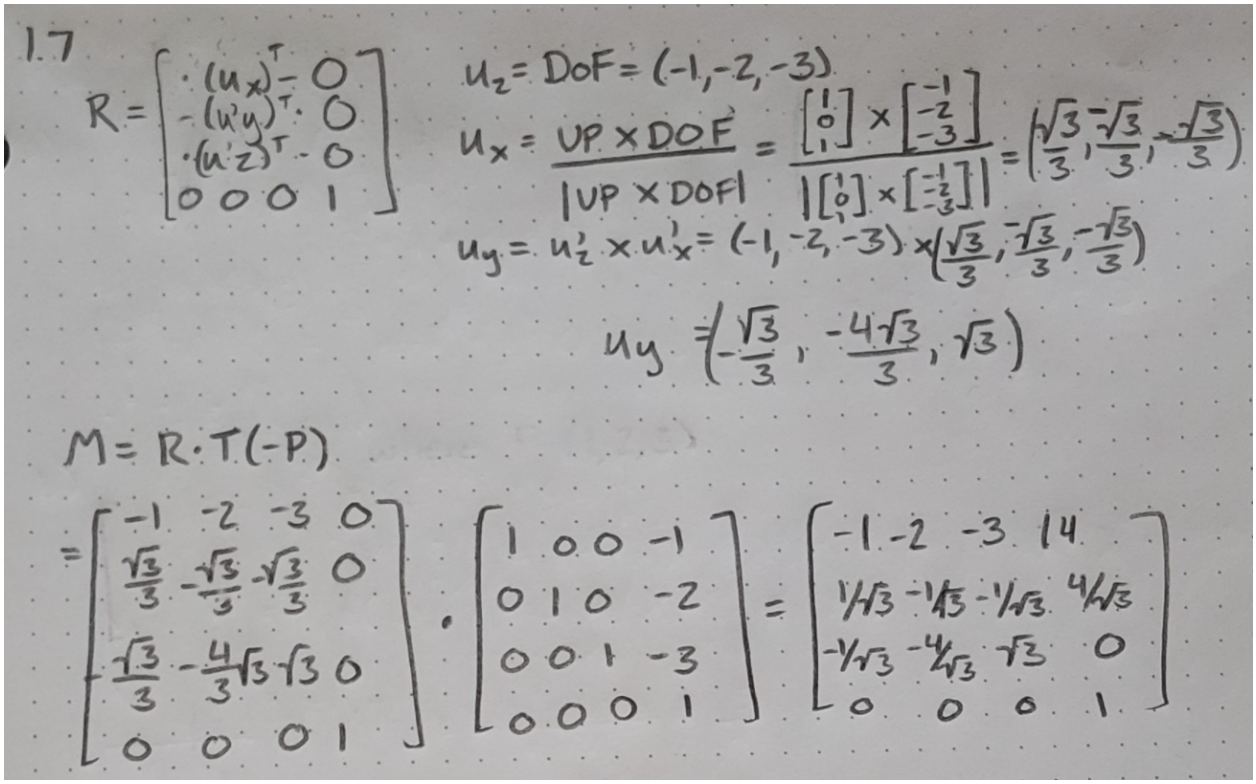
1.5 Write the transformation matrix for rotating a 3D point in homogeneous coordinates by 45° about a rotation axis given by (1, 1, 0). Express the transformation matrix you write in terms of the transformation matrices in the previous two questions.



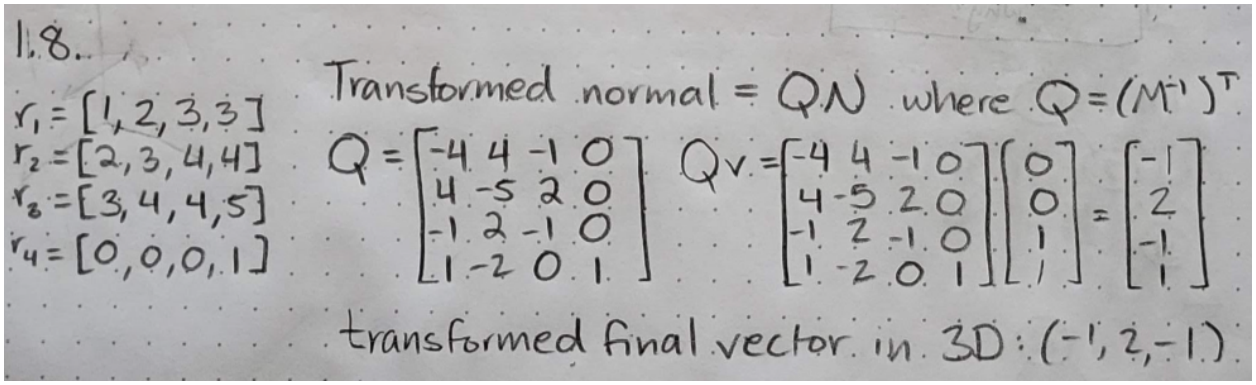
1.6 Given a camera coordinate system specified by 3 orthogonal axes u1,u2,u3 and a position c (all in world coordinates) write the transformation matrix that will transform world coordinates to camera coordinates.

The transformation matrix that will transform world coordinates to camera coordinates is R^T T(-(cx,cy,cz)) = M. This is notable, because this transformation will rotate and transform u1, u2 and u3 about the world coordinate system of x y z where c is the point of the camera. This can be found in the slides.

1.7 let p = (1, 2, 3) be the desired location of an airplane. Let v = (-1, -2, -3) be the desired direction of flight. Let u = (1, 0,1) be an up vector. Compute the transformation matrix that will position and orient an airplane according to the preceding specifications. Assume that initially the airplane is located at the origin and aligned with the x-axis.



1.8 Let r1 = [1,2, 3, 3], r2 = [2,3, 4, 4], r3 = [3,4,4, 5], r4 = [0,0, 0, 1] be the 4 rows of the transformation matrix M that is applied to the vertices of a 3D model. Let v = (0,0,1) be a vector that is normal to one of the surfaces in the model. Compute the coordinates of the normal vector after the object is transformed by M. Recall that the normal is transformed differently from vertices and that the homogeneous coordinate should be ignored.



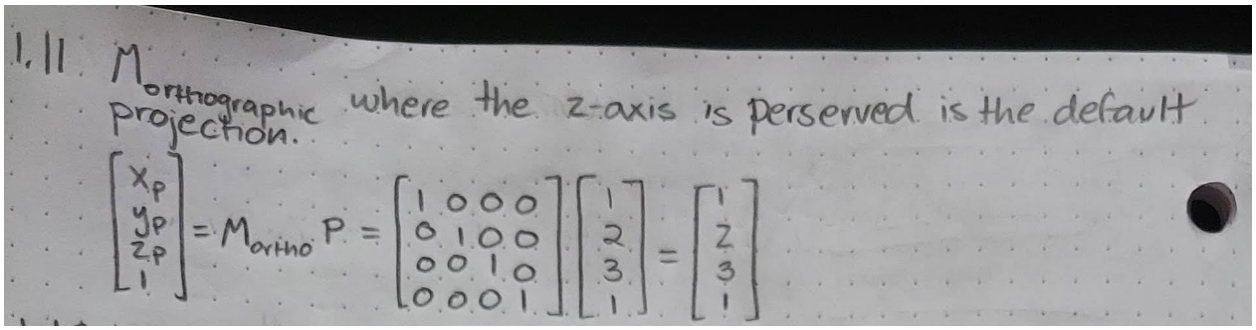
1.9 Explain the difference between perspective and orthographic projection. What are the advantages of each projection method?

Perspective projection makes sense for humans, and looks like what we would expect. It does not preserve length or angles though. Orthographic projection is great for measurements, as it preserves length and angles. Often used in CAD software.

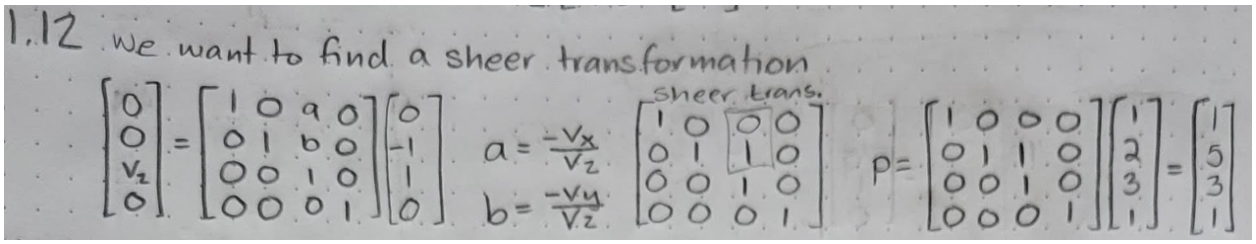
1.10 Explain the effect of oblique parallel projection and perspective projection on the view volume.

Perspective projection does not preserve length or angles, and in oblique parallel projection lines of sight are not perpendicular to the projection plane like in orthographic projection, but they are not realistic to real life either. There is no foreshortening in parallel projection.

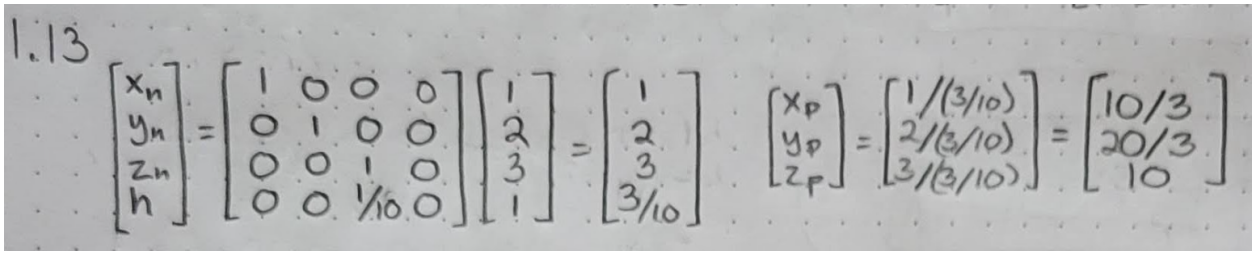
1.11 Let p (1, 2, 3) be a 3D point in camera coordinates. Let M_Othographic be an orthographic projection matrix that preserves the z-coordinate. Compute the coordinates of the point p after projection.



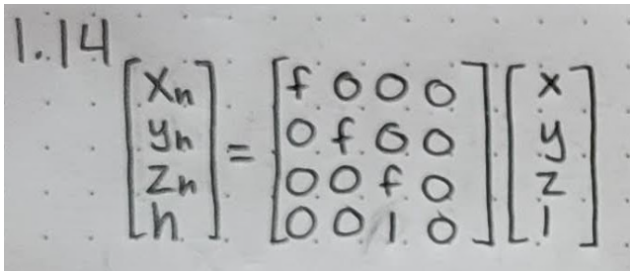
1.12 Let p (1, 2, 3) be a 3D point in camera coordinates. Let M_Parallel be an oblique parallel projection matrix that preserves the z-coordinate, whereas it is assumed that the direction of projection is given by v = (0, -1, 1) and the image plane is positioned at the origin. Compute the coordinates of the point p after projection.



1.13 Let p (1, 2, 3) be a 3D point in camera coordinates. Let M_Parallel be a perspective projection matrix, whereas it is assumed that the focal length is given by f=10, and that the image plane is perpendicular to the z-axis. Compute the coordinates of the point p after projection.



1.14 Given a perspective projection matrix M find another perspective projection matrix that will perform the exact same projection.



1.15 Given a view volume defined by a box with a minimum corner at $(x^w_{min}, y^w_{min}, z^w_{min})$ and a maximum corner at $(x^w_{max}, y^w_{max}, z^w_{max})$ and a viewport defined by a box with a minimum corner at $(x^v_{min}, y^v_{min}, z^v_{min})$ and a maximum corner at $(x^v_{max}, y^v_{max}, z^v_{max})$ find the window to viewport transformation that will transform the window box to the viewport box.

From the class slides this exact situation is referenced:

$$M_{\text{normalization}} = T_3 S_2 T_1$$

where:

$$T_3 \equiv T(XV_{\min}, YV_{\min}, ZV_{\min})$$

$$S_2 \equiv S\left(\frac{XV_{\max}-XV_{\min}}{XW_{\max}-XW_{\min}}, \frac{YV_{\max}-YV_{\min}}{YW_{\max}-YW_{\min}}, \frac{ZV_{\max}-ZV_{\min}}{ZW_{\max}-ZW_{\min}}\right)$$

$$T_1 \equiv T(-XW_{\min}, -YW_{\min}, -ZW_{\min})$$

1.16 Given a clipping plane defined by a unit normal n with a distance from the origin of $-d$ (i.e. all the points p on this plane satisfy $n \cdot p + d = 0$) write the condition that needs to be satisfied for a point p to be outside this clipping plane (i.e. on the side which is pointed by the normal). Specifically, given a plane with $n = (1, 2, 3)$ and $d = -4$ determine if the point $p = (2, 2, 2)$is inside or outside with respect to this clipping plane.

To determine whether or not a point is within a clipping plane, the equation $n \cdot p + d \leq 0$must be satisfied. Using that knowledge, we fill in the equation with the provided information$(2, 2, 2) \cdot (1, 2, 3) + (-4) = 8$ because 8 is greater than 0, this point does not lie in the clipping plane.

1.17 Given a face with a normal $n = (1, 2, 3)$ and a viewing direction of $v = (4, 5, 6)$ determine whether the front of this face needs to be rendered or not.

If $(v \cdot n) < 0$then the surface is visible, (1,2,3) dot (4,5,6) = 32 there are still 2 more checks, $c \cdot v_z < 0$and $c < 0$are both also not true and thus the surface is not visible

1.18 Explain the painter's algorithm and z-buffer algorithm for visible surface determination. Write the type (object space or image space) of each algorithm.

The painter's algorithm is an object space algorithm type that breaks down objects so that they could be separated without overlap, sorts based on depth, then renders distant objects first. The z buffer algorithm or depth buffering is an image space algorithm that checks if the z of an object is smaller than the z in the z buffer, if it is then the object is closer to the camera and should be rendered.

1.19 Explain the difference between oct-trees, k-d trees, and BSP trees for spatial sorting.

Octtrees split a view volume cube into 8 equal octants, until each octant has no more than k vertices in it.
K-d trees partition a plane along a predetermined axis, and just like octrees is based on hierarchical subdivision of space.
BSP trees are similar to k-d trees, however they do not worry about the axis that the plane is partitioned, meaning that there can be more efficient partitions however they will cost more computation power.

1.20 Explain the advantage of using vertex+face tables over using vertex tables alone. Describe a method for efficiently determining (after initial processing) what are all the faces that intersect at a given vertex.

Vertex tables alone repeat some vertices over and over (up to 3 times) for each face, and you have to scan every face and it is very difficult to update the vertex table. Having a vertex + face table makes it much more efficient, because each face holds 3 vertices that point then to the vertex table. An efficient strategy for determining what are all the faces that intersect at a given vertex is the winged edge table, which adds another concept of an edge, where the face table points to 3 edges, and the edge table references 2 vertices in the vertex table.