Cory Zimmerman

(Q1)
TODO

(Q3: 11.2)
Notice that this is Q3. I'm doing them in book order, not pset order:
Let the projection matrix be defined as

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

With eye coordinates x_e, y_e, z_e, this yields clip coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_3 \\ 1 \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \\ 1 \\ -z_3 \end{bmatrix}$$

This produces device coordinates

$$\begin{bmatrix} \frac{x_e}{-z_e} \\ \frac{y_e}{-z_e} \\ \frac{1}{-z_e} \\ 1 \end{bmatrix}$$

Using the projection matrix PQ instead generates

$$PQ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -3 & 0 \end{bmatrix}, PQ \cdot \begin{bmatrix} x_e \\ y_e \\ z_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 3x_e \\ 3y_e \\ 1 \\ -3z_e \end{bmatrix}$$

This produces device coordinates

$$\begin{bmatrix} \frac{3x_e}{-3z_e} \\ \frac{3y_e}{-3z_e} \\ \frac{1}{-3z_e} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_e}{-z_e} \\ \frac{y_e}{-z_e} \\ \frac{1}{-3z_e} \\ 1 \end{bmatrix}$$

Mathematically, it appears that the coordinates projected by PQ have identical x and y values but are squished closer towards the origin in the z direction compared to coordinates projected by just P. I beliee this has the effect of pushing a wider range of z values into the range of what will be rendered, producing the effect of seeing further.

I made a test example available here: `https://github.com/cfzimmerman/S24-CS175/blob/main/assignment-5-cory/mtx_proj.py`. With eye coordinates $(0,0,-1,1)$, $(0.5,0,-0.5,1)$, $(0,0.5,0.5,1)$, the device coordinates projected by just P are $(0,0,1,1)$, $(1,0,2,1)$, $(0,-1,-2,1)$, while the device coordinates projected by PQ are $(0,0,0.33,1)$, $(1,0,0.67,1)$, $(0,-1,-0.67,1)$.

(Q2: 11.3)
Abbreviating some of the steps from the question above, consider the projection by PS:

$$PQ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & -3 & 0 \end{bmatrix}, PQ \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} 3x_e \\ 3y_3 \\ 3 \\ -3z_e \end{bmatrix}$$

This produces device coordinates

$$\begin{bmatrix} \frac{3x_e}{-3z_e} \\ \frac{3y_e}{-3z_e} \\ \frac{3}{-3z_e} \\ \frac{-3z_e}{-3z_e} \end{bmatrix} = \begin{bmatrix} \frac{x_e}{-z_e} \\ \frac{y_e}{-z_e} \\ \frac{1}{-z_e} \\ 1 \end{bmatrix}$$

The resulting device coordinates are identical to those after projection by just P. This shows up mathematically because all entries in the output are scaled by 3, and then dividing by w_c eliminates that scale from all entries. Visually, I believe this has the effect of expanding evenly expanding and then condensing our view of the world by the same constant, producing the same picture. I added code for this to the script linked above as well, but, because the outputs are the same for both, they're not very interesting.

(Q4: 11.4)
Again, model mathematically:

$$QP = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} 3x_e \\ 3y_e \\ 3 \\ -z_e \end{bmatrix}$$

This produces device coordinates

$$\begin{bmatrix} \frac{3x_e}{-z_e} \\ \frac{3y_e}{-z_e} \\ \frac{3}{-z_e} \\ 1 \end{bmatrix}$$

Under this transformation, the x, y, and x coordinates are all scaled by a factor of 3. With these expanded dimensions, fewer points will fit within the clipped viewing region, creating a zooming effect on the scene.

Again using the same script as before, the same eye coordinates $(0, 0, -1, 1), (0.5, 0, -0.5, 1), (0, 0.5, 0.5, 1)$ which map under just P to $(0, 0, 1, 1), (1, 0, 2, 1), (0, -1, -2, 1)$ now map under QP to $(0, 0, 3, 1)$, $(3, 0, 6, 1), (0, 3, 6, 1)$.

(Q5: 12.3)
todo