1 Axis Aligned Rotation

[1 point] Write out the matrix that rotates a point about the X axis by $\theta$ radians.

Solution:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos\theta & -\sin\theta & 0 \\
0 & \sin\theta & \cos\theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

2 Arbitrary Rotation

Performing a rotation about an arbitrary axis can be seen as a composition of rotations around the bases. If you can rotate all of space around the bases until you align the arbitrary axis with the x axis, then you can treat the rotation about the arbitrary axis as a rotation around x. Once you have transformed space with that rotation around x, then you need to rotate space back so that the arbitrary axis is in its original orientation. We can do this because rotation is a rigid-body transformation.

To rotate a point $p$, this series of rotations looks like...

\[
p' = M_1^{-1} \cdot M_2^{-1} \cdot M_3 \cdot M_2 \cdot M_1 \cdot p
\]

Where $M_1$ rotates the arbitrary axis about the y axis, $M_2$ rotates the arbitrary axis to lie on the x axis, and $M_3$ rotates the desired amount about the x axis.

2.1 Rotation About an Arbitrary Point

[1 point] The equation above rotates a point $p$ about the origin. How can you make this operation rotate about an arbitrary point $h$?

Solution: Translate back to the origin from the point $h$, perform all rotations by applying the transformations listed above, and then translate back to $h$. (It was not necessary to write out the translation matrices to receive full credit for this problem; an explanation was sufficient.)

3 Camera Transformation

To transform a point $p$ from world space to screen space we use the normalizing transformation. The normalizing transformation is composed of four matrices, as shown here:

\[
p' = M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot p
\]

Each $M_i$ corresponds to a step described in lecture. Here, $p$ is a point in world space, and we would like to construct a point $p'$ relative to the camera’s coordinate system, so that $p'$ is its resulting position on the screen (with its z coordinate holding the depth buffer information). You can assume that the camera is positioned at $x,y,z,1$, it has look vector $\text{look}$ and up vector $\text{up}$, height angle $\theta_h$, width angle $\theta_w$, near plane $\text{near}$ and far plane $\text{far}$.

[3 pt. each] Briefly write out what each matrix is responsible for doing. Then write what values they have. Make sure to get the order correct (that is, matrix $M_2$ corresponds to only one of the steps described in lecture).

Solution:

- $M_1$: Does the unhinging; the perspective transformation.
  
  \[
  \hat{M}_1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{1}{c+1} & \frac{c}{c+1} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

- $M_2$: Shrinks back the clipping plane to lie at $z=-1$ and scales the corners of the back clipping plane to the canonical view volume.
  
  \[
  \hat{M}_2 = \begin{bmatrix}
\frac{\cot(\frac{\theta_w}{2})}{\text{far}} & 0 & 0 & 0 \\
0 & \frac{\cot(\frac{\theta_h}{2})}{\text{far}} & 0 & 0 \\
0 & 0 & \frac{1}{\text{far}} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- $M_3$: Rotates the view so that it is aligned with the world coordinate system.
  
  For $M_3$, let $w = -\frac{\text{look}}{\|\text{look}\|}$, $v = \frac{\text{up} - (\text{up} \cdot w)w}{\|\text{up} - (\text{up} \cdot w)w\|}$, $u = v \times w$
  
  \[
  \begin{bmatrix}
u_x & u_y & u_z & 0 \\
v_x & v_y & v_z & 0 \\
w_x & w_y & w_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

- $M_4$: Translates to the origin.
  
  \[
  \begin{bmatrix}1 & 0 & 0 & -x \\
0 & 1 & 0 & -y \\
0 & 0 & 1 & -z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
We accepted your solution if you made this substitution. You should pre-compute c in your implementation to avoid repeated computation. Note that we did NOT give you u, v, and w in this problem! You were expected to write out the definition via cross products in order to receive credit for M3. We were also unforgiving if you missed the negative sign on the translation matrix.

[3 pt. each] Recall that in class, we defined a camera’s virtual coordinate system as u, v, w axes. How would you translate the camera’s eye point E in its u, v, w coordinate system relative to the world x, y, z coordinate system… (find the new eye point E’)

1. One unit left? **Solution:** \[ E' = E - u \]
2. One unit down? **Solution:** \[ E' = E - v \]
3. One unit backward? **Solution:** \[ E' = E + w \]

[3 pt. each] How (mathematically) will you use the original u, v, and w vectors (call them \( u_0, v_0, w_0 \)) to get new u, v, and w vectors when...

1. Adjusting the “spin” in a clockwise direction by \( \theta \) radians?
2. Adjusting (rotating) the “pitch” to face downwards by \( \theta \) radians?
3. Adjusting the “yaw” to face right by \( \theta \) radians?

**Solution:**

1. \[ u = v_0 \ast \sin(\theta) + u_0 \ast \cos(\theta); v = v_0 \ast \cos(\theta) - u_0 \ast \sin(\theta); w = w_0 \]
2. \[ u = u_0; v = v_0 \ast \cos(\theta) - w_0 \ast \sin(\theta); w = v_0 \ast \sin(\theta) + w_0 \ast \cos(\theta) \]
3. \[ u = u_0 \ast \cos(\theta) - w_0 \ast \sin(\theta); v = v_0; w = u_0 \ast \sin(\theta) + w_0 \ast \cos(\theta) \]