Question 1  Write the viewing transformation, as a product of matrices, that brings a point $p$ from world to the canonical viewing volume. Which of these matrices would be affected by changing the height angle of the camera? How about changing the near and far clip planes?

\[ DS_{xy}^{-1}S_{xyz}RTp \]

Changing the height angle of the camera will affect the matrix used to scale the view volume in the $x$ and $y$ directions, $S_{xy}$. Changing the near and far clip planes will affect the uniform scaling matrix $S_{xyz}$ and the perspective transformation $D$.

Question 2  Call the inverse of the world to camera matrix the camera to world matrix. You will need to keep track of this matrix for later assignments. Because it is costly (and numerically less accurate) to use a general matrix inversion algorithm, it is suggested that you compute the camera to world matrix directly using entries in the world to camera matrix. Given the rotation, translation, and scale matrices in the world to camera transform, how will you compute the camera to world matrix? Suppose that the eye point is $P = \{Px, Py, Pz\}$, height angle is $\theta_h$, aspect ratio is $\alpha$, and near and far plane distances are $d_n, d_f$.

The camera to world matrix is $T^{-1}R^{-1}S_{xy}^{-1}S_{xyz}^{-1}$, i.e., invert every component matrix and multiply them in reverse order. The inversions of the individual matrices are:

\[
T^{-1} = T(P_x, P_y, P_z) \\
R^{-1} = R^T \\
S_{xy}^{-1} = S(\alpha \tan(\theta_h/2), \tan(\theta_h/2), 1) \\
S_{xyz}^{-1} = S(d_f, d_f, d_f)
\]

Question 3  If you were given a camera to world matrix, how would you reconstruct the eye point $P$, the up vector $U$, and the look vector $L$?

We know that in camera space the eye point lies at $(0, 0, 0)$, the look lies at $(0, 0, -1)$, and the up vector lies at $(0, 1, 0)$, regardless of rotations and translations that are applied to the camera. So, to convert the camera space eye point, look vector, or up vector to world space simply multiply them by the camera to world matrix.
1 Programming hints

- Store the camera to world, world to camera, and perspective matrices. Store the translation, rotation, and scale matrices as well. Write a routine which multiplies them together to produce the camera to world matrix and the world to camera matrix. Make sure you multiply the matrices together in the correct order.
- Write a check routine that checks the above invariants. Call it every time the matrices are changed.
- Make sure your clipping planes are set (0.01, 1000000 are good values).
- Write the orient routine first, explicitly setting the above matrices. The axis routines should then call the orient routine. Once you’re sure this is working, move on to rotating/ translating the camera.

Some common problems:

- *My cubes are missing pieces or look self-skewered.* Check your clipping planes (too far apart and you can get depth buffer errors).
- *I don’t see anything, but my matrices are set to the canonical location.* Check your clipping planes and remember that you can’t see any of a cube if your camera is inside the cube.
- *When I re-size the window the cubes changes shape, or my cubes don’t look like cubes.* Check your aspect ratio.
- *My matrix passes the above checks but I still don’t see anything, or see it wrong.* Make sure the bottom right value in your perspective matrix is 0.