

Algorithm 2: Camera/Transform

Due: Tuesday, September 29, 5:00pm

1 Transform : Translation

From the course notes, you know what a translation matrix looks like and how it works. Given a translation

matrix, $T = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$ what is T^{-1} ?

2 Transform : Scale

From the course notes, you know what a scaling matrix looks like and how it works. Given a scaling matrix,

$S = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$ what is S^{-1} ?

3 Transform : 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 \vec{x} axis, then you can treat the rotation about the arbitrary axis as a rotation around \vec{x} . Once you have transformed space with that rotation around \vec{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.

3.1

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

$$p' = A^{-1} \cdot B^{-1} \cdot C \cdot B \cdot A \cdot p$$

where:

- A rotates the arbitrary axis about the \vec{y} axis
- B rotates the arbitrary axis to lie on the \vec{x} axis
- C rotates the desired amount about the \vec{x} axis

Assuming we want to rotate $p = (p_x, p_y, p_z, 1)$ about vector $a = \langle a_x, a_y, a_z, 0 \rangle$ by λ radians, how would you calculate A , B , and C **in terms of the global functions** `getRotXMat`, `getRotYMat`, `getRotZMat`. (*Hint: This is tough! You can and should compute angles to use along the way. This can be accomplished by using `arctan` or `arccos`, for instance. Approach this step-by-step. This isn't extremely math heavy. For example you should only need one `sqrt` call.*) See the accompanying Camtrans assignment handout for details on the semantics of these functions.

3.2

The above equation rotates point p about the origin, how would you make this operation rotate about an arbitrary point h ?

4 Camera : Normalizing Transformation

4.1

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

$$p' = A \cdot B \cdot C \cdot D \cdot E \cdot p$$

$A, B, C, D,$ and E are matrices and correspond to the steps described in lecture. p is a point in world-space, and we would like to construct a 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).

BRIEFLY write out what the matrices $A, B, C, D,$ and E are responsible for doing. Then write what values they have. Make sure to get the order correct (that is, matrix E only corresponds to one of the steps described in lecture). Assume the following about the camera:

- it has position $(x, y, z, 1)$
- it has look vector $look$ and up vector up
- it has height angle θ_h and width angle θ_w .
- it has near and far clip planes $near$ and far , respectively.

4.2

For the assignment you need to perform rotation operations on the camera in its own virtual uvw coordinate system, e.g., spinning the camera about its v -axis. Additionally, you will need to perform translation operations on the camera in world space.

- Translation: How (mathematically) will you translate the camera's eye point...
 - one unit right?
 - one unit down?
 - one unit forwards?
- Rotation: How (mathematically) will you use the u , v , and w vectors, in conjunction with a rotation angle θ , to get new u , v , and w vectors when:
 - adjusting the “spin” in a clockwise direction by θ radians?
 - adjusting (rotating) the “pitch” to face upwards by θ radians?
 - adjusting the “yaw” to face right by θ radians?

You can either move in and out of the camera coordinate space to perform these transformations, or you can do arbitrary rotations in world space. Both answers are acceptable.