Camtrans Lab Algorithm

October 18, 2020

Introduction to Computer Graphics, Fall 2020

Due Date: 6:00 PM EST on Oct 19, 2020.

Instructions: You may discuss this assignment with other students, but you must abide by the rules stated in the collaboration policy (no notes from any discussions, your handin must be completely written up by you and contain only your own work, 15 minute delay between discussions and writing down any notes of your own) and write the logins of the student(s) you collaborated with next to each problem. Hand in the assignment on Gradescope as a PDF (please make sure the document is anonymous) no later than 6:00pm EST on the due date. *Late hand-ins are not accepted under any circumstances.*

This assignment is worth **7%** of your Sceneview grade.

0 Axis Aligned Rotation

[**1 point**] Write out the matrix that rotates a point about the x-axis (or better, in the yz-plane) by θ radians.

1 Arbitrary Rotation

Performing a rotation about an arbitrary axis v can be done by composing rotations around the standard unit-vector directions (i.e., the x, y, and z axes), but in general this is very difficult.

If you can somehow rotate all of space using the three standard rotations so that your arbitrary axis v ends up being transformed to the x axis (let's represent this sequence of three rotations with a single matrix G, then you can rotate about x, and then *undo* the first rotation.

The end result will be a rotation (any product of rotations is always a rotation, amazingly enough!), and it'll be a rotation that leaves v fixed, i.e., it'll be a rotation about v.

When we apply this composite rotation to a point p, this series of rotations looks like...

$p' = G^{-1} \cdot R_x \cdot G \cdot p$

Where G rotates v to the x-axis, and R_x rotates the desired amount about the x-axis, and G^{-1} sends the x-axis back to v.

1.1 Rotation About an Arbitrary Point

[**1 point**] The equation above rotates a point p about the line through the origin, in direction v. How would you modify things so that you could rotate p around a line through an arbitrary point h, in direction v?

2 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 $(x, y, z, 1)$ and that it has look vector *look*, up vector *up*, height angle θ_h, width angle θ_w, near plane *near* and far plane *far*.

[**½ 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). We expect you to define any variables that are not among the camera parameters defined above!

[**½ pt. each**] Recall that in class, we described a camera's virtual coordinate system using u, v, w axes. To clarify, u, v, and w are orthogonal unit vectors that define the camera coordinate system. Given the camera's world space eye position E, what is the new eye position *E'* after translating the camera in its own coordinate system (i.e. camera space):

1. One unit right?
2. One unit up?
3. One unit forward?

[**½ pt. each**] Mathematically, how will you use the original u, v, and w vectors (let's call them u_0, v_0, w_0) to get new u, v, and w vectors for each of the following operations (in camera space):

1. "Rolling" the camera in a counterclockwise direction by θ radians?
2. "Pitching" the camera upwards by θ radians?
3. "Yawing" the camera to the right by θ radians?

If you're confused about what roll, pitch, and yaw are, look at (pun intended) the Viewing lectures.