CS1566 Assignment 3 (part 1)
Transformer Algorithm

Out: Thu 9/26
Due: Thu 10/3 5:00pm

Grading:

Problem	Points
1	3
2	2
3	2
4	2
5	1
6	1
7	15
8	4

1. For Assignment 3, we ask you to move in space, along with the object, the local coordinate axes associated with the object. For this task, you will need to keep track and transform, along with the object vertices, the origin-point of the object's local coordinate system (i.e., originally, the same as the origin of the world coordinate system), **and** the unit vectors pointing along the object's local axes (originally, the same as the X, Y, Z unit vectors). Please recall from the Linear Algebra recitation (slide 4, "Vectors and Vector Space (1/2)") that vectors do not have a starting point; they only have direction and magnitude – where magnitude is 1 for unit vectors. In this exercise you will need to think about whether the transformation matrices used to rotate, scale, and translate vectors are the same or different (and if so, different how) from the transformation matrices used to perform the same operations on vertices.

1a. If the object vertices undergo a Rz(θ) rotation, where Rz is the rotation around Z matrix from your class notes, what transformation do you need to apply to:

I) the center point of the object coordinate system (cx, cy, cz, 1)
II) the unit vector (3/5, 0, 4/5, 0)

in order to have the unit vector rotate along with the object? You'll probably want to spell out the updated (if necessary) matrices here.

1b. If the object vertices undergo scaling by a factor of 2 along the X-axis, what transformation do you need to apply to:

I) the center point of the object coordinate system (cx, cy, cz, 1)
II) the unit vector (3/5, 0, 4/5, 0)

in order to have the unit vector scale along with the object? You'll probably want to spell out the updated (if necessary) matrices here.

1c. Finally, if the object vertices undergo translation by 5 units along the Y-axis, what transformation do you need to apply to:

I) the center point of the object coordinate system (cx, cy, cz, 1)
II) the unit vector (3/5, 0, 4/5, 0)

in order to have the unit vector translate along with the object? You'll probably want to spell out the updated (if necessary) matrices here.

Note: Keep in mind that rotations, translations and scaling may not work the same way for unit vectors as for unit normals (unit normals was the particular case briefly discussed in class).

2. Consider an object centered at the origin. What is the correct transformation order, in order to make sure that the object's vertices (let's call one of these vertices v) spin around the object's vertical axis, while the object is being translated in space along X:

T(x)*R(y)*v

or

R(y)*T(x)*v?

3. Given a vector [2,3,5], compute its length.

4. Given a vector [2,3,5], compute the unit vector pointing along the same direction.

5. Given a vertex in homogeneous coordinates
GLfloat my_vertex[4]
write the procedure (in pseudocode):

```
void real_translation(float tx, float ty, float tz);
```

that computes the effect of the translation tx, ty, tz on the coordinates of my_vertex. Store the updated coordinates in my_vertex[4]. You are not allowed to make any OpenGL calls. Use matrix multiplication in your pseudocode.

6.Given a vertex in homogeneous coordinates
GLfloat my_vertex[4]
write the procedure (in pseudocode):

```
voidreal_rotation(float deg, int x, int y, int z);
```

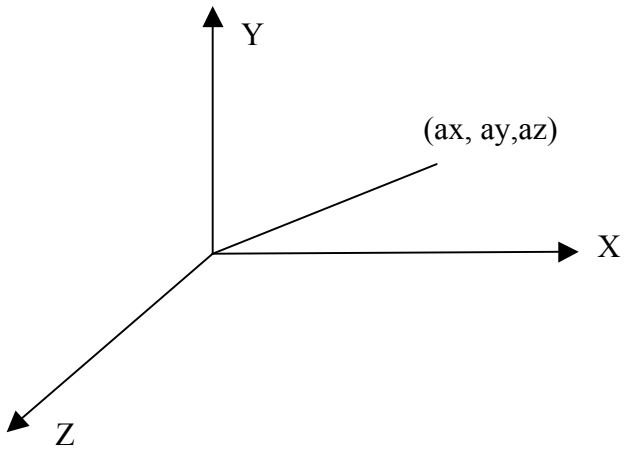
that, when
x = 1 computes the effect of a rotation by deg degrees around the X axis on the coordinates of my_vertex.
y = 1 computes the effect of a rotation by deg degrees around the Y axis on the coordinates of my_vertex.
z = 1 computes the effect of a rotation by deg degrees around the Z axis on the coordinates of my_vertex.

Store the updated coordinates in my_vertex[4]. You are not allowed to use any OpenGL calls. Use matrix multiplication in your pseudocode.

7(a).
Using the real_rotation you wrote for problem 6, you can rotate a point around the x or y or z axis. However, rotations are not always around these principal axes. An object can rotate around any arbitrary axis. Let's say that now you want to rotate a point around an arbitrary axis.

Given the axis (ax, ay, az), derive the transformation matrix for a rotation of Θ degrees around it. **Note:** to get any credit for this problem, you **must** derive the transformation matrix via the coordinate system alignment approach (u,v,w to x,y,z) discussed in the camera lectures. No other "high-school-style approach" (e.g., by breaking the alignment into two separate rotations, each along a separate axis) will be accepted.

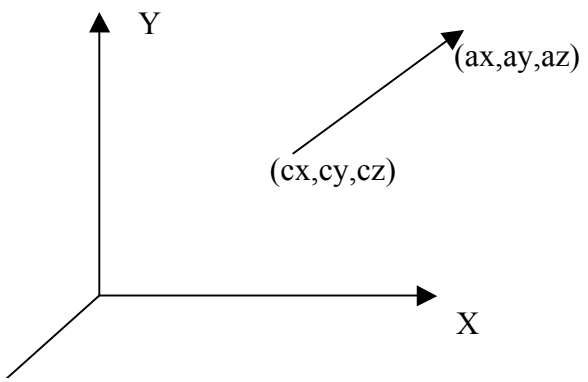
(Hint: This is somewhat similar to the procedure for aligning the camera coordinate system with the canonical volume coordinate system that we discussed in class. Basically if you can rotate the space until you align the arbitrary axis with the x axis, then you can treat the rotation about the arbitrary axis 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.
You do not have to pick especially the x axis for the alignment. You can align the arbitrary axis with any of the x, y or z axes. Keep in mind that you'll need normalized vectors).



7(b). Give the pseudocode for the arbitrary axis rotation. You may use the real_rotation function you wrote for problem 6.

```
void arbitrary_rotation(float deg, float ax, float ay,
float az);
```

8. In the previous question, you solved how to rotate a point about an arbitrary axis passing through the origin. In other words, you solved how to rotate around a line passing through (0,0,0) and (ax, ay, az). Now, we want to generalize the problem a bit more. Give the transformation matrix for rotation around a line passing through (cx,cy,cz) and (ax,ay,az). You can think of the problem as rotation around an arbitrary axis (ax,ay,az) located at point C(cx,cy,cz).



8(b). Give the pseudocode for the arbitrary axis, arbitrary point rotation. You may use any of the functions you wrote for the earlier problems.

```
void arbitrary_rotation_point(float deg, float cx, float
cy, float cz, float ax, float ay, float az);
```