CS 418: Homework #2

Solution

1. (20pt) You are given two quaternions, $\mathbf{q_1} = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\mathbf{k}$ and $\mathbf{q_2} = \mathbf{i} + \mathbf{j} + \mathbf{k}$.

   (a) What is the norm of $\mathbf{q_1}$?

   Solution:

   Recall that the norm of a quaternion $\mathbf{q} = a_1 + a_2\mathbf{i} + a_3\mathbf{j} + a_4\mathbf{k}$ is given by $||\mathbf{q}|| = \sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}$. Therefore $||\mathbf{q_1}|| = \sqrt{\left(\frac{\sqrt{2}}{2}\right)^2 + \left(\frac{\sqrt{2}}{2}\right)^2} = \sqrt{\frac{1}{2} + \frac{1}{2}} = 1$.

   (b) What is the inverse of $\mathbf{q_1}$?

   Solution:

$$\mathbf{q_1}^{-1} = \frac{\bar{\mathbf{q_1}}}{||\mathbf{q_1}||^2} = \frac{\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\mathbf{k}}{1} = \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\mathbf{k}$$

   Notice that the inverse of a unit quaternion is simply its conjugate.

   (c) Compute $\mathbf{q_1 q_2}$ and $\mathbf{q_2 q_1}$. Are they equal ?

   Solution:

   Rewrite the quaternions in the form $\mathbf{q} = (s, \mathbf{v})$, where $s = a_1$ and $\mathbf{v} = \begin{bmatrix} a_2 & a_3 & a_4 \end{bmatrix}$. In this form, quaternion multiplication is given by:

$$\mathbf{q_1 q_2} = (ss' - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

$$\mathbf{q_1 q_2} = \left(-\frac{\sqrt{2}}{2}, \begin{bmatrix} 0 & \sqrt{2} & \frac{\sqrt{2}}{2} \end{bmatrix}\right) = -\frac{\sqrt{2}}{2} + \sqrt{2}\,\mathbf{j} + \frac{\sqrt{2}}{2}\mathbf{k}$$

$$\mathbf{q_2 q_1} = \left(-\frac{\sqrt{2}}{2}, \begin{bmatrix} \sqrt{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}\right) = -\frac{\sqrt{2}}{2} + \sqrt{2}\,\mathbf{i} + \frac{\sqrt{2}}{2}\mathbf{k}$$

   Clearly $\mathbf{q_1 q_2} \neq \mathbf{q_2 q_1}$.

   (d) $\mathbf{q_1}$ represents a rotation. What are the rotation axis and angle ?

   Solution:

   First we find θ using the fact that $s = \cos(\theta/2)$.

$$\theta = 2\cos^{-1}\left(\frac{\sqrt{2}}{2}\right) = 90^\circ$$

   Now we can find the axis of rotation $\mathbf{u}$ by dividing $\mathbf{v}$ by $\sin(\theta/2)$.

$$\mathbf{u} = \frac{\mathbf{v}}{\sin(45^\circ)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

   Therefore $\mathbf{q_1}$ represents a rotation of 90° around the z-axis.

2. (10pt) Suppose you are given the 4×4 transformation matrix

$$\mathbf{M} = \begin{bmatrix} \cos\theta & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & \cos\theta \end{bmatrix}$$

For a given value of θ, describe the geometric effect of applying this transformation to a torus. (You may assume that $\cos\theta \neq 0$.)

Solution:

Multiplying the given transformation matrix with a point (x, y, z) on the torus, we get:

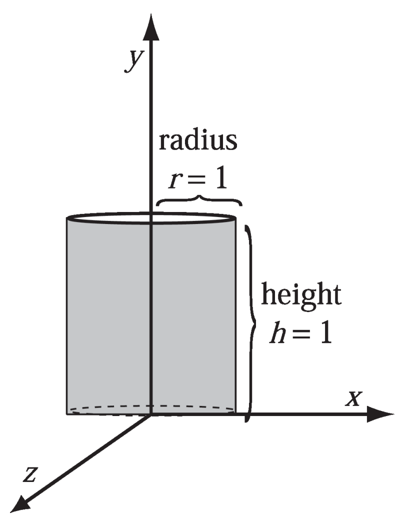
$$\begin{bmatrix} \cos\theta & & & \\ & \cos\theta & & \\ & & \cos\theta & \\ & & & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta \\ y\cos\theta \\ z\cos\theta \\ \cos\theta \end{bmatrix}$$

We need to project the resulting vector back into 3D space by dividing the first three components by the fourth:

$$\begin{bmatrix} x\cos\theta/\cos\theta \\ y\cos\theta/\cos\theta \\ z\cos\theta/\cos\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Which gives us the same original point. In general, a transformation matrix with the same non-zero coefficient across the diagonal does nothing.

3. (10pt) Harvey has written some code to draw the following unit cylinder:



His drawing code places vertices along the upper and lower rims of the cylinder, connecting them together with triangles which stretch the entire length of the cylinder. He draws the cylinder using a shiny material. Given the position of the light, he expects to see a specular highlight in the middle of the cylinder.

(a) When Harvey executes his OpenGL code (using Gouraud shading) no highlight appears in the middle of the cylinder. However, when he renders it with RenderMan (using Phong shading) the highlight appears. Explain why this is.

Solution:

Gouraud shading evaluates the normals and colors at the vertices of each polygon, and interpolates the vertex colors to obtain the colors throughout the rest of the polygon. Phong shading interpolates the normal of the vertices, but still evaluates the color at each pixel using a normal. As a result, a hightlight that falls onto the middle of polygon will show in Phong shading, but not be picked up by Gouraud shading. Since the cylinder is made of triangles with vertices only at the bases, hightlights in the middle of the cylinder do not show in Gouraud shading.

(b) How could the OpenGL code be altered (still using Gouraud shading) so that the highlight would appear?

Solution:

Compose the cylinder with more, smaller triangles in the **vertical direction**, so that there are sufficient vertices in the middle of the cylinder to show hightlights.

4. (20pt) Suppose we use a pinhole camera to take a picture of a real rectangular carpet, and the picture covers the entire carpet. Note that the carpet may become a quadrilateral in the picture because of perspective projection. The resolution of the picture is 640x480. The pixel locations of the four corners of the carpet in the picture are $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$. Later we produce a 3D virtual model of the carpet as a rectangle in the world coordinate system and render it using texture mapping in OpenGL. We would like to use the picture as the texture image.

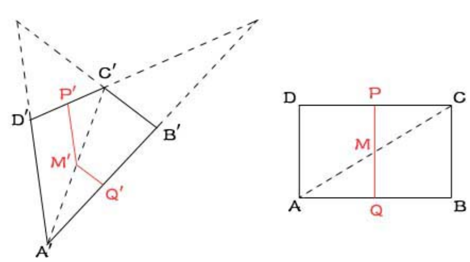
(a) How can we obtain the texture coordinates for the corners of the carpet so that the texture-mapped carpet looks like the original one ?

Solution:

Since the texture coordinates are in range $[0, 1]$, we need to convert the pixel location (x_i, y_i) into texture coordinates $(\frac{x_i}{640}, \frac{y_i}{480}), (i = 1, 2, 3, 4)$.

(b) Is it possible that the edges of the carpet become curved in the picture ? Why ?

Solution:



The carpet in the texture image(left figure) is acquired through perspective projection. Perspective projection maps straight lines to straight lines, but is more general than affine transformation in that parallel lines AB and CD can become non-parallel lines $A'B'$ and $C'D'$.

When you specify the texture coordinates of polygon $ABCD$ to be $A'B'C'D'$ in OpenGL, the way it setup mapping between the two polygons is it first decomposes polygon $ABCD$ into two triangles ABC and ACD, then map points within each triangle using barycentric coordinates. So triangle ABC is mapped to $A'B'C'$ using barycentric mapping, and so on for the other triangles. Because the edges $A'B', B'C', C'D'$ and $D'A'$ are all straight lines in the texture, we know that they **cannot** be curved when mapped onto the polygon.

What's worth mentioning is that there maybe a discontinuity in the direction (not color) of the texture across edge AC. Since in triangle ACD, line MP will get mapped to $M'P'$, and in triangle ABC, line MQ will get mapped to $M'Q'$, but M', P', Q' may not be colinear in the texture.

5. (20pt) In class we derived the basis matrices for cubic Bézier curves and Hermite curves. Cubic B-splines are another type of piecewise smooth curves. Each segment of a cubic B-spline curve is defined by four consecutive control points, $\mathbf{p}_{i-3}, \mathbf{p}_{i-2}, \mathbf{p}_{i-1}$, and, $\mathbf{p}_i$, as follows,

$$\mathbf{p}(u) = b_1(u)\mathbf{p}_{i-3} + b_2(u)\mathbf{p}_{i-2} + b_3(u)\mathbf{p}_{i-1} + b_4(u)\mathbf{p}_i,$$

where $b_1(u) = \frac{1}{6}(1-u)^3$, $b_2(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$, $b_3(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$, and $b_4(u) = \frac{1}{6}u^3$. Derive the basis matrix for the B-spline formulation.

Solution:

$$b_1(u) = \frac{1}{6}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\begin{bmatrix} 1 \\ -3 \\ 3 \\ -1 \end{bmatrix} \qquad b_2(u) = \frac{1}{6}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\begin{bmatrix} 4 \\ 0 \\ -6 \\ 3 \end{bmatrix}$$

$$b_3(u) = \frac{1}{6}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \\ 3 \\ -3 \end{bmatrix} \qquad b_4(u) = \frac{1}{6}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

So

$$\begin{bmatrix} b_1(u) & b_2(u) & b_3(u) & b_4(u) \end{bmatrix} = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

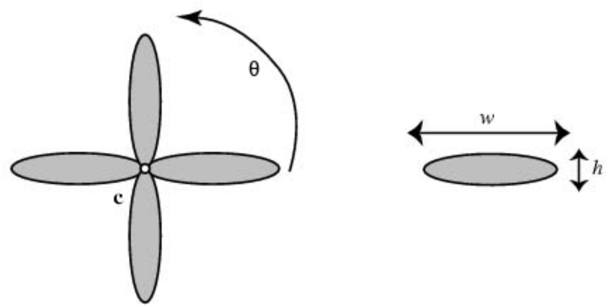
Thus

$$\mathbf{p}(u) = \begin{bmatrix} b_1(u) & b_2(u) & b_3(u) & b_4(u) \end{bmatrix}\begin{bmatrix} \mathbf{p}_{i-3} \\ \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \end{bmatrix} = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}\begin{bmatrix} \mathbf{p}_{i-3} \\ \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \end{bmatrix}$$

We know that the basis matrix for B-spline is

$$\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

6. (30pt) Suppose that we are building a model of a propellor-driven airplane that we can use in an exciting new animation. Each propellor will be modelled as a *planar* object made out of ellipses, as shown below:

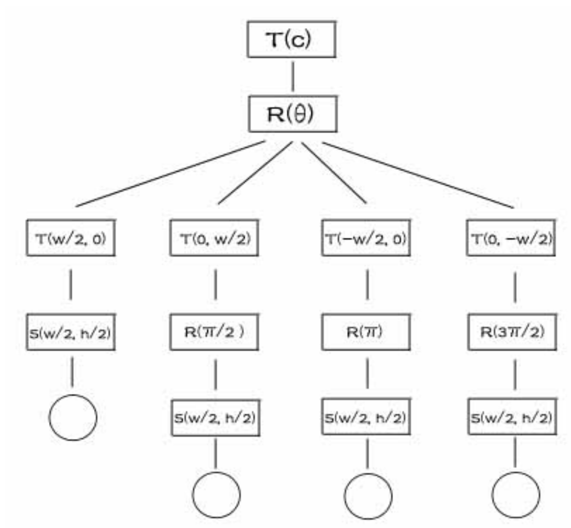


You have decided to provide two “control knobs” to the animator using this model: the center of the propellor (**c**) and the angle by which the blades are rotated about this center (θ). Each of the four blades is an ellipse, as shown on the right.

(a) Construct a transformation hierarchy for the object pictured above. The only geometric primitive you may use is a circle of radius 1 centered about the origin. Your hierarchy may only contain transformation nodes and geometry nodes that draw unit circles. For all transformation nodes, clearly indicate what transformation is being performed. [Hint: All your geometry nodes must be at the leaves of the tree.]

Solution:

There are many different ways to draw the propeller in the specified configuration. Here're we provide one of them. T means translation, R means rotation, S means scaling, and the circle at the bottom means "draw unit circle".



(b) Suppose the propellor *linearly* accelerates from 0 to N rotations/second in one second. We draw the propellor at n frames/second. That means we need to draw n frames during the acceleration. Please describe how to obtain the rotation angle for the i-th frame, $1 \leq i \leq n$. You can assume the first frame is drawn at time $\frac{1}{n}$.

Solution:

The propeller linearly acceralates from $v_0 = 0$ degrees/second to $v_t = 2\pi N$ degrees/second. Since $v_t = v_0 + at$, where a is the acceleration constant, we know $a = \frac{v_t - v_0}{t} = \frac{2\pi N}{1} = 2\pi N$.

The total angle of rotation θ at time t is given by formula $\theta = v_0 t + \frac{1}{2}at^2$, and the i-th frame corresponds to time $t = \frac{i}{n}$. Plugging in $v_0 = 0, a = 2\pi N, t = \frac{i}{n}$, we get the total angle of rotation θ at the i-th frame $\theta = \pi N(\frac{i}{n})^2$.