Assignment 2: Transformation and Viewing

15-462 Graphics I
Spring 2002
Frank Pfenning

Sample Solution
Based on the homework by
Kevin Milans `kgm@andrew.cmu.edu`

1 Three-Dimensional Homogeneous Coordinates (15 pts)

If we are interested only in two-dimensional graphics, we can use three-dimensional homogeneous coordinates by representing a point $\mathbf{P}$ by $[x\ y\ 1]^T$ and a vector $\mathbf{v}$ by $[\alpha\ \beta\ 0]^T$.

1. Find the matrix representation of a counter-clockwise rotation by θ degrees about the origin.

   The matrix for counter-clockwise rotation by θ is given by

   $$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

   As a check, consider the point $\mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

   When rotated by 45° *counter-clockwise, we'd expect* $\mathbf{p}$ *to be translated to* $\begin{bmatrix} 0 \\ \sqrt{2} \\ 1 \end{bmatrix}$.

   We have

   $$\begin{aligned} \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} &= \begin{bmatrix} \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \sqrt{2} \\ 1 \end{bmatrix}, \end{aligned}$$

   as expected.

2. Find the translation matrix for given displacement vector $[\delta_x\ \delta_y\ 0]^T$

The matrix for translation by displacement vector $\begin{bmatrix} \delta_x \\ \delta_y \\ 0 \end{bmatrix}$ *is given by*

$$\begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix}.$$

3. Find the scaling matrix for factors α_x and α_y.

   The scaling matrix for factors α_x *and* α_y *is given by*

$$\begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

4. Find the x-shear matrix for shear angle θ.

   The x*-shear matrix for shear angle* θ *is given by*

$$\begin{bmatrix} 1 & \cot\theta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

5. Derive the explicit transformation matrix for a reflection about the axis specified by a point $\mathbf{p_0} = [x_0\ y_0\ 1]^T$ and a unit vector $\mathbf{u} = [\alpha_x\ \alpha_y\ 0]^T$.

   To derive the reflection matrix, begin with the identity matrix, translate the point p_0 *to the origin, rotate until the line given by* u *coincides with the* x *axis, flip everything across the* x *axis with a scaling matrix, rotate back to the original orientation, and finally translate the origin to the location* p_0*. The final result is displayed below.*

$$\begin{bmatrix} \alpha_x^2 - \alpha_y^2 & 2\alpha_x\alpha_y & x_0(\alpha_y^2 - \alpha_x^2 + 1) - 2\alpha_y\alpha_x y_0 \\ 2\alpha_x\alpha_y & \alpha_y^2 - \alpha_x^2 & y_0(\alpha_x^2 - \alpha_y^2 + 1) - 2\alpha_y\alpha_x x_0 \\ 0 & 0 & 1 \end{bmatrix}$$

6. Show how the x-shear matrix can be represented as a composition of rotations, scalings, and translations.

   The x*-shear operation for a shearing angle* Ψ *reduces to rotations and scalings as follows:*

   (a) Rotate by $\theta = \frac{1}{2}\Psi$ *counter-clockwise. Call this matrix* M_1*.*
   (b) Scale with $\alpha_x = \sin\theta$ *and* $\alpha_y = \cos\theta$*. Call this matrix* M_2*.*
   (c) Rotate by $45°$ *clockwise. Call this matrix* M_3*.*
   (d) Scale with $\alpha_x = \frac{\sqrt{2}}{\sin\Psi}$ *and* $\alpha_y = \sqrt{2}$*. Call this matrix* M_4*.*

Intuitively, we want to rotate and then scale so we can "stretch" the world along an arbitrary orthogonal axis. Then, we want to rotate back so that the original x-axis line again coincides with the true x-axis. Finally, we have to scale the world to clean up scaling.

To see that this works, we compute the matrix given by these transformations.

$$
\begin{aligned}
M_4M_3M_2M_1 &= M_4M_3 \begin{bmatrix} \sin\theta & & \\ & \cos\theta & \\ & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & \\ \sin\theta & \cos\theta & \\ & & 1 \end{bmatrix} \\
&= M_4M_3 \begin{bmatrix} \sin\theta\cos\theta & -\sin^2\theta & \\ \sin\theta\cos\theta & \cos^2\theta & \\ & & 1 \end{bmatrix} \\
&= \frac{\sqrt{2}}{2} M_4 \begin{bmatrix} 1 & 1 & \\ -1 & 1 & \\ & & \sqrt{2} \end{bmatrix} \begin{bmatrix} \sin\theta\cos\theta & -\sin^2\theta & \\ \sin\theta\cos\theta & \cos^2\theta & \\ & & 1 \end{bmatrix} \\
&= \frac{\sqrt{2}}{2} M_4 \begin{bmatrix} 2\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta & \\ 0 & \sin^2\theta + \cos^2\theta & \\ & & \sqrt{2} \end{bmatrix} \\
&= \frac{\sqrt{2}}{2} M_4 \begin{bmatrix} \sin(2\theta) & \cos(2\theta) & \\ 0 & 1 & \\ & & \sqrt{2} \end{bmatrix} \\
&= \frac{\sqrt{2}}{2} \begin{bmatrix} \frac{\sqrt{2}}{\sin\Psi} & & \\ & \sqrt{2} & \\ & & 1 \end{bmatrix} \begin{bmatrix} \sin(2\theta) & \cos(2\theta) & \\ 0 & 1 & \\ & & \sqrt{2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sin\Psi} & & \\ & 1 & \\ & & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \sin(2\theta) & \cos(2\theta) & \\ 0 & 1 & \\ & & \sqrt{2} \end{bmatrix} \\
&= \begin{bmatrix} 1 & \frac{\cos\Psi}{\sin\Psi} & \\ & 1 & \\ & & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & \cot\Psi & \\ & 1 & \\ & & 1 \end{bmatrix}.
\end{aligned}
$$

2 Rigid Body Transformations (20 pts)

A *rigid body transformation* may rotate and move, but not reflect, re-scale, or otherwise distort an object. We first investigate these in two dimensions (see Problem 1) and then generalize to three dimensions. Your tests should avoid trigonometric functions or their inverses.

1. Devise a test whether a given 3×3 transformation matrix in homogeneous coordinates is a rigid body transformation in 2 dimensions.

   We can test whether a given matrix $\mathbf{M}$ *is a rigid body transformation in 2 dimensions by observing the action on the basis vectors and the origin. First, the resulting vectors must be of unit length, and second, they must be orthogonal, and third they must be "right-handed". Finally, the origin should be translated to a new origin.*

$$\begin{aligned}
\mathbf{M}\mathbf{u}_x &= \mathbf{M}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \mathbf{x} \\
\mathbf{M}\mathbf{u}_y &= \mathbf{M}\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} = \mathbf{y} \\
\mathbf{M}P_0 &= \mathbf{M}\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}q_1\\q_2\\q_3\end{bmatrix} = Q_0
\end{aligned}$$

   Then we check

   (a) $\mathbf{x}$ *is a unit vector:* $x_3 = 0$ *and* $|\mathbf{x}| = 1$ *(that is,* $x_1^2 + x_2^2 = 1$*)*
   (b) $\mathbf{y}$ *is a unit vector:* $y_3 = 0$ *and* $|\mathbf{y}| = 1$ *(that is,* $y_1^2 + y_2^2 = 1$*)*
   (c) Q_0 *is a point:* $q_3 \neq 0$
   (d) $\mathbf{x}$ *and* $\mathbf{y}$ *are orthogonal:* $\mathbf{x} \cdot \mathbf{y} = 0$ *(that is,* $x_1y_1 + x_2y_2 = 0$*)*
   (e) $\mathbf{x}$ *and* $\mathbf{y}$ *must be "right-handed":*

$$\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1y_2 - y_1x_2 = 1$$

2. Generalize your test to check if a given 4×4 transformation matrix in homogeneous coordinates is a rigid body transformation in 3 dimensions.

   We proceed as in part 1, except that tests for orthogonality and right-handedness is

slightly more complicated.

$$\begin{aligned}
\mathbf{M}\mathbf{u}_x &= \mathbf{M}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix} = \mathbf{x}\\
\mathbf{M}\mathbf{u}_y &= \mathbf{M}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix} = \mathbf{y}\\
\mathbf{M}\mathbf{u}_z &= \mathbf{M}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}z_1\\z_2\\z_3\\z_4\end{bmatrix} = \mathbf{z}\\
\mathbf{M}P_0 &= \mathbf{M}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}q_1\\q_2\\q_3\\q_4\end{bmatrix} = Q_0
\end{aligned}$$

Then we check

(a) $\mathbf{x}$ *is a unit vector:* $x_4 = 0$ *and* $|\mathbf{x}| = 1$

(b) $\mathbf{y}$ *is a unit vector:* $y_4 = 0$ *and* $|\mathbf{y}| = 1$

(c) $\mathbf{z}$ *is a unit vector:* $z_4 = 0$ *and* $|\mathbf{z}| = 1$

(d) Q_0 *is a point:* $q_4 \neq 0$

(e) $\mathbf{x}$ *and* $\mathbf{y}$ *are orthogonal:* $\mathbf{x} \cdot \mathbf{y} = 0$

(f) $\mathbf{y}$ *and* $\mathbf{z}$ *are orthogonal:* $\mathbf{y} \cdot \mathbf{z} = 0$

(g) $\mathbf{x}$ *and* $\mathbf{z}$ *are orthogonal:* $\mathbf{z} \cdot \mathbf{x} = 0$

(h) The basis vectors form a right-handed frame: $\mathbf{x} \times \mathbf{y} = \mathbf{z}$

We do not write out here the standard ways to compute $|u|$, $u \cdot v$, *and* $u \times v$ *similar to part 1.*

3 Viewing Transformations (15 pts)

Assume the function `void earth ();` draws a three dimensional model of the earth with the south pole at the origin, the north pole at the point $(0, 1, 0)$, and the Greenwich meridian (0^o longitude) pointing in the z-direction. We are interested in drawing the earth as seen from a point in space with a given *longitude* and *latitude* (specified in degrees) and given *distance* from the surface of the earth. We want to be looking down into the direction of the earth's center and have a square viewport that should cover a field of vision of 30^o degrees. We are assuming the earth is a perfect sphere.

1. Does the specification above uniquely determine the perspective viewing transformation? Explain if there are additional degrees of freedom.

   The specification above does not uniquely determine the perspective viewing transformation. First of all, after we position the camera, we can choose which direction

is up. We also have some freedom to determine the aspect ratio of the frustum as well as where we'll place the near and far clipping planes.

2. Give code for a function

   `void viewEarth (float` *`longitude`*`, float` *`latitude`*`, float` *`distance`*`);`

   and carefully explain the reasoning behind your solution. If there are additional degrees of freedom, set them to some reasonable values. Your function should call `earth ();` to draw the earth.

```
void viewEarth (float longitude, float latitude, float distance)
  {
        /* first, reset the projection matrix */
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        /*
         * 30 degree field of view, 1.0 aspect ratio.
         * We know we're 'distance' away from the surface of the
         * earth, and that the earth has a diameter of 1.0, so we
         * can set the near and far clipping planes accordingly
         */
        gluPerspective(30.0, 1.0, distance, distance + 1.0);
        glMatrixMode(GL_MODELVIEW);

        /*
         * reset the modelview matrix
         */
        glLoadIdentity();

        /*
         * look at the earth from far away.  be sure to remember that
         * there is 0.5 distance between the surface of the earth and the
         * origin
         */
        gluLookAt(0.0, 0.0, distance + 0.5, 0.0, 0.0, -1.0, 0.0, 1.0, 0.0);

        /*
         * rotate the earth to the correct latitude and longitude
         */
        glRotatef(latitude, 1.0, 0.0, 0.0);
        glRotatef(-longitude, 0.0, 1.0, 0.0);

        /*
         * translate the earth so it's center is at the origin
         */
        glTranslatef(0.0, -0.5, 0.0);

        earth();

        glutSwapBuffers();
  }
```