Chapter 5

Two-Dimensional Geometric Transformations

Methods for the implementation and application of both two-dimensional and three-dimensional translation, rotation, scaling, reflection, and shear operations are explored in the exercises for this chapter, in addition to programming projects involving the OpenGL geometric-transformation functions.

Exercises

5-1. The example rotation program in Section 5-1 can be completed using routines from previous example programs and some additional animation procedures. An animation sequence can be displayed with the processing steps: (1) display a given polygon at an initial position, (2) compute the coordinates for the next position of the polygon, (3) delete the previously displayed polygon, (4) display the polygon at the rotated position, (5) repeat steps (2) through (4) until an end condition is satisfied.

One method for erasing the polygon is to display it in the background color. The animation display can be ended after a fixed number of cycles, or the rotation can be ended when a particular key is pressed. (Other methods for performing and ending a computer-generated animation are given in Chapter 13.)

For rotation steps of 10 degrees (0.175 radians) or less, the trigonometric functions are accurately approximated as

$$\cos\theta \approx 1.0, \qquad \sin\theta \approx \theta$$

These approximations can be used for larger angular steps also (up to a maximum of about 20 to 30 degrees), but the object will get more distorted as the rotation step size increases. At the end of each 360° cycle, the polygon vertices should be reset to their original positions.

5-2. This equation can be validated by evaluating both sides using a matrix representation from either Section 5-2 or Section 5-11.

5-3. The translational terms for the rotation transformation with respect to a specified pivot point (Eqs. 5-9) are

$$tr_x = x_r(1 - \cos\theta) + y_r \sin\theta$$

and

$$tr_y = -x_r \sin\theta + y_r(1 - \cos\theta)$$

From Eqs. 5-9, we can also derive expressions for the inverse transformation. This yields the following matrix representation for the inverse rotation transformation.

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & -tr_x\cos\theta - tr_y\sin\theta \\ -\sin\theta & \cos\theta & tr_x\sin\theta - tr_y\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore, the translational terms (elements in the first two positions of the third column) for the modified scaling matrix of Eq. 5-39 are

$$x_f(1 - s_1\cos^2\theta - s_2\sin^2\theta) + y_f(s_1 - s_2)\cos\theta\sin\theta$$

and

$$x_f(s_1 - s_2)\cos\theta\sin\theta + y_f(1 - s_1\sin^2\theta - s_2\cos^2\theta)$$

where the fixed position $(x_f,\ y_f)$ is also the rotational pivot point for this transformation sequence. The elements in the first two columns and the third row of the scaling matrix in Eq. 5-39 are unchanged.

5-4. (a) Using the trigonometric identities for the sine and cosine of the sum of two angles, we can express the elements of the product matrix for two successive rotations in the xy plane about the coordinate origin as

$$\begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since the addition of angles is commutative, this composite transformation is commutative. The commutative property of two successive rotations can also be shown by comparing the elements of the composite matrix $\mathbf{R}_1\,\mathbf{R}_2$ with the elements of the composite matrix $\mathbf{R}_2\,\mathbf{R}_1$.

(b) The composite matrix for two successive translations is given by Eq. 5-28. And this matrix product is commutative because the addition of the translation parameters is commutative.

(c) The composite matrix for two successive scaling transformations is given by Eq. 5-33. This is also a commutative matrix product because the product of the scaling parameters is commutative.

5-5. The matrix representation for a two-dimensional transformation sequence consisting of scaling followed by rotation, relative to the coordinate origin, is

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

And the matrix representation for a rotation followed by scaling, relative to the coordinate origin, is

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

These two matrices are equivalent only when $s_x = s_y$.

5-6. Since matrix multiplication is associative, the elements in the composite matrix can be obtained by first multiplying the translate and rotate matrices or by first multiplying the rotate and scale matrices. (However, matrix multiplication is not commutative, in general, so the ordering in the transformation sequence cannot be altered.)

5-7. Values for the input parameters can be specified in a file, or they could be entered using keyboard input. The input parameters are the pivot-point coordinates, the scaling fixed-point coordinates, the translation distances, the rotation angle, and the scaling factors. The input statements replace the assignment statements for these parameters in procedure `displayFcn`.

5-8. An additional input for the preceding exercise is the vertex list for the polygon.

5-9. This exercise can be combined with the previous two exercises so that the program includes input parameters for specifying the transformation sequence order.

5-10. The product of matrix 5-52 and matrix $\mathbf{R}(\pi/2)$ (Eq. 5-19) yields matrix 5-55.

5-11. The product of matrix 5-53 and $\mathbf{R}(\pi/2)$ (Eq. 5-19) yields matrix 5-56.

5-12. Two successive reflections about a single axis yields the identity matrix; i.e., the object is returned to its original position. A reflection about one axis followed by a reflection about the other axis is equivalent to a rotation of 180°, assuming that the reflection parameters are either 1 or -1.

5-13. A transformation sequence for obtaining the composite matrix for reflection about a given line is:

(1) Translate the line so that it passes through the coordinate origin. Since parameter b is the value for the y-axis intersection, the translation parameters are $t_x = 0$ and $t_y = -b$.

(2) Rotate the line onto the x axis. A horizontal line is already on the x axis, and a vertical line is perpendicular to the x axis (a 90° angle). Otherwise the angle between the reflection line and the positive x axis is calculated from the slope of the line:

$$\theta = \tan^{-1} m$$

If this angle is positive ($m > 0$), perform a clockwise rotation with respect to the coordinate origin. If the angle is negative ($m < 0$), perform a counterclockwise rotation with respect to the coordinate origin. Thus, in either case, set $\theta = -\theta$ and use the transformation matrix for a counterclockwise rotation.

(3) Perform the reflection about the x axis using transformation matrix 5-52.

(4) Carry out the inverse of steps (2) and (1) to return the line to its original position.

Multiplying these five transformation matrices and simplifying the trigonometric expressions using double-angle formulas, we obtain the transformation matrix for reflection about the line $y = m\,x + b$ as

$$\begin{bmatrix} \cos 2\theta & -\sin 2\theta & b\sin 2\theta \\ -\sin 2\theta & -\cos 2\theta & b(1+\cos 2\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

In this matrix, angle θ is the negative of the line angle with respect to the x axis, as calculated in step (2).

5-14. A reflection about any line in the xy plane can be accomplished using the transformation matrix in Exercise 5-13. Two successive reflections produce the identity transformation, which is equivalent to a rotation of 360° about the coordinate origin, providing the reflection parameters are the same in both reflections.

5-15. Vertex coordinates for a polygon can be sheared using a modified transformation matrix 5-39, which scales each vertex and returns it to its original y position. Shearing transformation 5-57 is also equivalent to an x-direction translation with t_x as a function of y.

5-16. As in the preceding exercise, vertex coordinates for a polygon can be sheared using a modified transformation matrix 5-39. In this case, vertices are scaled relative to their distance from the reference line and returned to their original x positions. Shearing transformation 5-61 is also equivalent to a y-direction translation with t_y as a function of x.

5-17. Given a set of character definitions and a required shearing amount, we can apply matrix 5-59 to each character to shear it relative to its baseline before displaying the character.

5-18. The transformation equations can be obtained by projecting the x and y coordinates of point **P** onto the x' and y' axes of the rotated system and noting the positions of these projections. For the x' axis, the projection of x is $x\cos\theta$ and the projection of y is $y\sin\theta$. The x' coordinate in the rotated system is the sum of these two projections. Similarly, the y' coordinate in the rotated system is the projection of y onto the y' axis, which is $y \cos\theta$, minus the projection of x onto the y' axis, which is $x \sin\theta$. These transformation equations can also be obtained using matrix 5-64, since the coordinate origins for the two systems are at the same position.

5-19. Section 5-8 describes the algorithm for constructing the coordinate-transformation matrix using the y' direction vector. Point $\mathbf{P}_0$ is first translated to the position of the origin for the first system. This translation matrix is then multiplied by rotation matrix 5-68, whose elements are obtained from the normalized expression for $\mathbf{V}$.

5-20. Transfer the block of pixels to an array, then load the array back into the frame buffer, transferring the lower left element of the array to the specified frame-buffer position, as shown in Fig. 5-26. Sections 3-19 and 5-7 discuss the functions `glReadPixels`, `glDrawPixels`, and other OpenGL routines for performing these operations.

5-21. Construct a table of binary results for the logical operations *and*, *or*, and *exclusive or*. Two successive *and* operations combining an input value of 0 with an original frame-buffer value of 1 changes the frame-buffer value to 0, while other original frame-buffer bit values are unchanged. Two successive *or* operations combining an input value of 1 with an original frame-buffer value of 0 changes the frame-buffer value to 1, while other original frame-buffer bit values are unchanged. Two successive *exclusive or* operations restores all original frame-buffer bit values.

5-22. A frame-buffer bit value is unchanged after two successive binary additions or subtractions, but higher-order bits are affected for an input bit value of 1 in a low-order position.

5-23. This routine can be implemented using the OpenGL functions that were introduced in Sections 3-19 and 5-7.

5-24. As discussed in Section 5-6, a 90° rotation is performed by interchanging rows and columns of the pixel block. The OpenGL routines from Section 5-7 can be used to retrieve a pixel array, rotate it, and then transfer it to another frame-buffer location. An input pattern can be given for this problem, or the pattern can be created by the program and then rotated.

5-25. This exercise is an extension of the preceding exercise. Arbitrary rotations require antialiasing or approximation methods to transfer pixel values, as illustrated in Fig. 5-28. Section 5-7 contains OpenGL routines for reading and writing pixel values to the frame buffer.

5-26. The `glPixelZoom` function (Section 5-7) can be used to scale the pixel values, or the scaling can be implemented using explicit replication operations. Other OpenGL routines in Section 5-7 can be used to save a block of pixels in an array and to place the scaled pixel values back into the frame buffer.

5-27. Unit vector **u**, the axis vector, is defined in Eq. 5-82, and the product of rotation matrices 5-90 and 5-95 is

$$\begin{bmatrix} d & -a\,b/d & -a\,c/d & 0 \\ 0 & c/d & -b/d & 0 \\ a & b & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first three rows of this matrix contain the elements of a mutually perpendicular set of vectors that satisfy the equations in 5-100, where parameter d is defined in Eq. 5-86.

5-28. This expression is evaluated using the rules for vector dot and cross products (Appendix A) and the vector elements $\mathbf{p} = (x,\, y,\, z)$, $\mathbf{p}' = (x',\, y',\, z')$, and $\mathbf{v} = (a,\, b,\, c)$. The solution for $\mathbf{p}'$ can then be written in matrix form as

$$\mathbf{p}' = \mathbf{M}_R(\theta)\,\mathbf{p}$$

5-29. In this case, $\mathbf{u} = (0,\, 0,\, 1)$, and parameters s and $\mathbf{v}$ are evaluated from Eqs. 5-103, where

$$\mathbf{v} = (a,\, b,\, c) = (0,\, 0,\, \sin\frac{\theta}{2})$$

Substituting these values for s and $\mathbf{v}$ into Eq. 5-107 and simplifying the expressions using the trigonometric half-angle formulas, we obtain the rotation matrix in Eq. 5-74.

5-30. Concatenating the rotation matrices in Eq. 5-97 produces matrix 5-108. That is,

$$\mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) = \mathbf{M}_R(\theta)$$

5-31. The components of vector $\mathbf{v}$ are calculated from Eqs. 5-103:

$$\mathbf{v} = (a,\, b,\, c) = (u_x,\, u_y,\, u_z)\sin\frac{\theta}{2}$$

Substituting these values into matrix 5-107 and simplifying using the trigonometric half-angle formulas, we obtain the matrix in Eq. 5-108.

5-32. This problem is a three-dimensional extension of the methods discussed in the solution to Exercise 5-1. A procedure for rotating the object about a specified axis can be implemented using one of the methods given in Section 5-11. This procedure could then be used in an animation program after the viewing methods in Chapter 7 have been discussed.

5-33. The direction cosines define the unit vector for the required scaling direction; that is,

$$\mathbf{u} = (a, b, c) = (\cos\alpha,\ \cos\beta,\ \cos\gamma)$$

Assume that the scaling is to be performed relative to the coordinate origin. Unit vector **u** can then be transformed onto the z axis using one of the rotation procedures discussed in Section 5-11. The rotation matrix for this transformation is

$$\mathbf{R} = \begin{bmatrix} d & -\frac{a\,b}{d} & -\frac{a\,c}{d} & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ a & b & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $d = \sqrt{b^2 + c^2}$. The inverse rotation transformation is obtained as the transpose of **R**:

$$\mathbf{R}^{-1} = \begin{bmatrix} d & 0 & a & 0 \\ -\frac{a\,b}{d} & \frac{c}{d} & b & 0 \\ -\frac{a\,c}{d} & -\frac{b}{d} & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After rotating the unit direction vector onto the z axis, we apply scaling transformation 5-110 with $s_x = 1$, $s_y = 1$, and $s_z = s$. Then objects are transformed so that the unit vector is returned to its original orientation. The matrix for the composite transformation is

$$\mathbf{R}^{-1}\,\mathbf{S}\,\mathbf{R} = \begin{bmatrix} d^2 + s\,a^2 & a\,b(s-1) & a\,c(s-1) & 0 \\ a\,b(s-1) & \frac{c^2+b^2(a^2+c\,d^2)}{d^2} & \frac{b\,c(a^2+s\,d^2-1)}{d^2} & 0 \\ a\,c(s-1) & \frac{b\,c(a^2+s\,d^2-1)}{d^2} & \frac{b^2+c^2(a^2+s\,d^2)}{d^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5-34. Input to the routine can be a set of plane parameters, A, B, C, and D, or three coordinate positions in the plane. A reflection relative to the plane can be accomplished by first performing a transformation sequence that brings the plane into coincidence with a coordinate plane such as the xy plane. Then, the object is reflected (Section 5-4), and the inverse transformation sequence is applied to return the plane to its original position.

From the plane equation, we know that the coordinate position $(0,\ 0,\ -\frac{D}{C})$ is on the reflection plane. Then translate this point to the coordinate origin and rotate the plane so that it coincides with the xy plane. The translation distances are $t_x = 0$, $t_y = 0$, and $t_z = \frac{D}{C}$, and the rotation can be performed using matrix 5-102, which rotates the reflection-plane normal vector, $\mathbf{N} = (A,\ B,\ C)$, so that is aligned with the z axis. This brings the reflection plane into coincidence with the xy plane. The final steps are to reflect the object relative to the xy plane and perform the inverse of the translate-rotate operations.

5-35. Input to this procedure can include the shearing values, the shearing reference point, and a set of polyhedron vertices. The equations for the x and y shearing transformations are given in Section 5-5, and the z-axis transformation equations are given in Sections 5-14.

5-36. A Cartesian reference frame is defined by specifying the coordinate position for its origin and the vector directions for two of its three axes. One method for accomplishing this is to input three points: one point gives the position for the origin, and the other two points specify positions on two of the coordinate axes, relative to the first input point. Alternatively, the coordinate axes could be specified with two vectors that are defined relative to the origin of the original reference frame. In either case, the third axis vector is constructed along a line perpendicular to the first two axes, so that the three axes form a right-handed Cartesian system.

Since the two specified coordinate axes must be perpendicular, the procedure for this exercise could test the two input vectors. If their dot product is not 0, then the vectors are not perpendicular. In this case, the procedure could either reject the input or adjust one of the vectors so that it is perpendicular to the other input vector.

To adjust an input vector, first compute a vector (in the proper direction) that is perpendicular to the two input vectors. Then calculate the cross product of the computed vector with one of the input vectors. This produces a set of three mutually perpendicular vectors. As an example, suppose the two input vectors are $\mathbf{V}_y$ and $\mathbf{V}_z$, which define the y and z directions for the new system, and suppose the y axis vector is to be adjusted so that it is perpendicular to $\mathbf{V}_z$. A set of mutually perpendicular unit axis vectors is then obtained with the calculations

$$\begin{aligned}
\mathbf{u}'_z &= \frac{\mathbf{V}_z}{|\mathbf{V}_z|} \\
\mathbf{u}'_x &= \frac{\mathbf{V}_y \times \mathbf{u}'_z}{|\mathbf{V}_y|} \\
\mathbf{u}'_y &= \mathbf{u}'_z \times \mathbf{u}'_x
\end{aligned}$$

The composite matrix for the coordinate transformation is obtained by first translating the coordinate origin of the new system to the origin of the original system, then rotating the new axis vectors so that they are aligned with the axes of the original system. Equation 5-116 gives the rotation-matrix elements for this transformation.

5-37. For this exercise, the OpenGL transformation routines from the example program in Section 5-17 can be substituted into the example program in Section 5-4 to replace the explicit matrix-construction and transformation operations. The z coordinate for all triangle vertices is set to 0, and the triangle is to be rotated as indicated in Fig. 5-15.

5-38. The preceding program is modified by providing an input set of polygon vertices to replace the explicit vertex coordinate list. This set of coordinates can be specified in a data file.

5-39. The preceding program is modified by providing additional input parameters that specify the order in which the geometric transformations are to be applied.

5-40. The preceding program is modified by providing yet another input list, which specifies the values for the translation, rotation, and scale parameters. This input list replaces the list of explicit values for the transformation parameters in the example program of Section 5-4.