Lecture 13: Matrix Representations for Affine and Projective Transformations

thou shalt set apart unto the LORD all that openeth the matrix Exodus 13:12

1. Matrix Representations for Affine Transformations

In the previous lecture we derived formulas for affine and projective transformations using the techniques of vector geometry and vector algebra. Formulas expressed in vector algebra are effective for computing individual transformations, but these formulas are not efficient for composing transformations. Therefore in this lecture we are going to derive matrix representations for each of the classical transformations of Computer Graphics so that we will be able to compose these transformations by matrix multiplication.

To fix our notation, denote the origin by O and the three unit vectors along the x,y,z coordinate axes by i,j,k. Then in 3-dimensions for any vector v and any point P, there are constants (v_1, v_2, v_3) and (p_1, p_2, p_3) such that

$$v = v_1 i + v_2 j + v_3 k$$
$$P = O + p_1 i + p_2 j + p_3 k \ .$$

The constants (v_1, v_2, v_3) and (p_1, p_2, p_3) are the rectangular coordinates of v and P, and we often abuse notation and write $v = (v_1, v_2, v_3)$ and $P = (p_1, p_2, p_3)$.

Now let A be an affine transformation. Then since A preserves linear and affine combinations

$$A(v) = v_1 A(i) + v_2 A(j) + v_3 A(k)$$
$$A(P) = A(O) + p_1 A(i) + p_2 A(j) + p_3 A(k) \ .$$

Thus if we know the value of A on the three vectors i,j,k and on the single point O, then we know the value of A on every vector v and every point P -- that is, any affine transformation A is completely determined by the four values $A(O), A(i), A(j), A(k)$.

Since affine transformations map points to points and vectors to vectors, there are constants (a_{ij}) such that

$$A(i) = a_{11} i + a_{12} j + a_{13} k = (a_{11}, a_{12}, a_{13})$$
$$A(j) = a_{21} i + a_{22} j + a_{23} k = (a_{21}, a_{22}, a_{23})$$
$$A(k) = a_{31} i + a_{32} j + a_{33} k = (a_{31}, a_{32}, a_{33})$$
$$A(O) = O + a_{41} i + a_{42} j + a_{43} k = (a_{41}, a_{42}, a_{43})$$

Therefore the 12 numbers (a_{ij}) completely characterize the affine transformation A. It seems natural then that we should represent A by a 4×3 matrix and write

$$A = \begin{pmatrix} A(i) \\ A(j) \\ A(k) \\ A(O) \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix}.$$

But there is something not quite right about this 4×3 matrix. If we represent a transformation by a matrix, then we expect that if we multiply the coordinates of a point or a vector by the matrix we should get the result of the transformation applied to the point or vector. However, points and vectors in 3-dimensions have only 3 coordinates, whereas this matrix has 4 rows.

To resolve this inconsistency, we shall introduce affine coordinates for points and vectors, and 4×4 matrices to represent affine transformations. Recall that in 2-dimensions we insert a third coordinate, an affine coordinate, for points and vectors: the affine coordinate for points is 1, the affine coordinate for vectors is 0. This affine coordinate allows us to represent affine transformations in the plane by 3×3 matrices. We shall follow similar conventions in 3-space; in 3-dimensions we insert a fourth, affine coordinate for points and for vectors. Again the affine coordinate for points is 1, the affine coordinate for vectors is 0. Following this convention,

$$A = \begin{pmatrix} A(i) & 0 \\ A(j) & 0 \\ A(k) & 0 \\ A(O) & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{pmatrix} \tag{1.1}$$

.

Now for vectors we have

$$A(v) = v_1 A(i) + v_2 A(j) + v_3 A(k) = (v_1, v_2, v_3, 0) * \begin{pmatrix} A(i) & 0 \\ A(j) & 0 \\ A(k) & 0 \\ A(O) & 1 \end{pmatrix}$$

or equivalently

$$A(v) = (v_1, v_2, v_3, 0) * \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{pmatrix} = (v, 0) * A.$$

Similarly, for points

$$A(P) = A(O) + p_1 A(i) + p_2 A(j) + p_3 A(k) = (p_1, p_2, p_3, 1) * \begin{pmatrix} A(i) & 0 \\ A(j) & 0 \\ A(k) & 0 \\ A(O) & 1 \end{pmatrix}$$

or equivalently

$$A(P) = (p_1, p_2, p_3, 1) * \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{pmatrix} = (P,1) * A .$$

In this way affine transformations are represented by multiplying the affine coordinates of points and vectors by special 4×4 matrices, where the last column is always $(0,0,0,1)^T$.

2. Linear Transformation Matrices and Translation Vectors

To derive explicit 4×4 matrix representations for all the standard transformations of 3-dimensional Computer Graphics -- translation, rotation, mirror image, uniform and non-uniform scaling, and orthogonal and perspective projection -- we need to convert the corresponding formulas from vector algebra into matrix form.

To achieve this goal, notice that the 4×4 matrix representing an affine transformation A always has the following form:

$$A = \begin{pmatrix} L_A & 0 \\ w_A & 1 \end{pmatrix}, \tag{2.1}$$

where L_A is a 3×3 matrix and w_A is a 3-dimensional vector. Moreover, for any vector v,

$$A(v) = (v,0) * \begin{pmatrix} L_A & 0 \\ w_A & 1 \end{pmatrix} = (v * L_A, 0). \tag{2.2}$$

Thus the 3×3 matrix L_A represents a linear transformation on vectors. Similarly, for any point P,

$$A(P) = (P,1) * \begin{pmatrix} L_A & 0 \\ w_A & 1 \end{pmatrix} = (P * L_A + w_A, 1). \tag{2.3}$$

Thus the vector w_A represents a translation vector. Hence we can decompose A so that

$$A = \begin{pmatrix} L_A & 0 \\ w_A & 1 \end{pmatrix} = \begin{pmatrix} 3 \times 3 \ Linear Transformation \ Matrix & 0 \\ Translation \ Vector & 1 \end{pmatrix}. \tag{2.4}$$

Our goal is to find explicit expressions for the matrix L_A and the vector w_A for each of the standard affine transformations of 3-dimensional Computer Graphics.

2.1 Linear Transformation Matrices. Let us begin with the 3×3 matrix L_A. Looking at the formulas for affine transformations from the previous lecture, we see that on vectors these formulas all have following form:

$$A(v) = v^{new} = c_1 v + c_2 (v \bullet u)\tilde{u} + c_3 \, u \times v ,$$

where $u, \tilde{u}$ are fixed unit direction vectors and c_1, c_2, c_3 are constants. To represent these linear transformation using matrices, we introduce the following three 3×3 matrices:

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 3 \times 3 \ \ Identity \ \ Matrix \tag{2.5}$$

$$u^T * \tilde{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} * \begin{pmatrix} \tilde{u}_1 & \tilde{u}_2 & \tilde{u}_3 \end{pmatrix} = \begin{pmatrix} u_1\tilde{u}_1 & u_1\tilde{u}_2 & u_1\tilde{u}_3 \\ u_2\tilde{u}_1 & u_2\tilde{u}_2 & u_2\tilde{u}_3 \\ u_3\tilde{u}_1 & u_3\tilde{u}_2 & u_3\tilde{u}_3 \end{pmatrix} \tag{2.6}$$

$$u \times \_ = \begin{pmatrix} 0 & u_3 & -u_2 \\ -u_3 & 0 & u_1 \\ u_2 & -u_1 & 0 \end{pmatrix}. \tag{2.7}$$

It follows easily by direct computation from Equations (2.5), (2.6), (2.7) that

$$v * I = v$$

$$v * (u^T * \tilde{u}) = (v * u^T) * \tilde{u} = (v \bullet u)\tilde{u}$$

$$v * (u \times \_) = u \times v .$$

Thus I represents the identity transformation; $u^T * \tilde{u}$ represents the dot product with u (or parallel projection onto u when $\tilde{u} = u$); and $u \times \_$ represents the cross product with u. Hence

$$c_1 v + c_2 (v \bullet u)\tilde{u} + c_3 u \times v = v * \left(c_1 I + c_2 u^T * \tilde{u} + c_3 u \times \_ \right). \tag{2.8}$$

Therefore using Equation (2,8), we can easily find the matrix representing the linear transformations L_A mechanically in the following manner:

$$\begin{array}{l} A(v) = c_1 v + c_2 (v \bullet u)\tilde{u} + c_3 u \times v \\ \quad \downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow \\ L_A = c_1 I + c_2 u^T * \tilde{u} + c_3 u \times \_ \end{array} \tag{2.9}$$

2.2 Translation Vectors. To find the translation vector w_A, observe that each of the affine transformations A from the previous lecture, except for translation, has a fixed point Q -- that is, a point Q for which

$$A(Q) = Q .$$

Now for any point P,

$$P = Q + (P - Q) .$$

Therefore since A is an affine transformation,

$$A(P) = A(Q) + A(P - Q) = Q + A(P - Q) . \tag{2.10}$$

Moreover, since $P - Q$ is a vector, we know from Section 2.1 that

$$A(P - Q) = (P - Q) * L_A = P * L_A - Q * L_A . \tag{2.11}$$

Hence by Equations (2.10) and (2.11),

$$A(P) = Q + P * L_A - Q * L_A = P * L_A + Q(I - L_A) . \tag{2.12}$$

But by Equation (2.3)

$$A(P) = P * L_A + w_A . \tag{2.13}$$

Therefore, we conclude from Equations (2.12) and (2.13) that

$$w_A = Q * (I - L_A), \tag{2.14}$$

where Q is any fixed point of the affine transformation A.

With Equations (2.9) and (2.14) for L_A and w_A in hand, we are now ready to derive matrix representations for each of the standard transformations of 3-dimensional Computer Graphics.

3. Rigid Motions

Rigid motions in 3-dimensions are composites of three basic transformations: translation, rotation, and mirror image.

3.1 Translation. Let w be a translation vector. For any point P and any vector v,

$$v^{new} = v = v * I$$

$$P^{new} = P + w = P * I + w$$

Therefore the matrix $Trans(w)$ representing translations is simply

$$Trans(w) = \begin{pmatrix} I & 0 \\ w & 1 \end{pmatrix}. \tag{3.1}$$

Example 3.1: The matrix representing translation parallel to the x-axis by the distance d is given by

$$Trans(d\,i) = \begin{pmatrix} I & 0 \\ d\,i & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d & 0 & 0 & 1 \end{pmatrix}.$$

Similarly, the matrices representing translation parallel to the y or z-axes by the distance d are given by

$$Trans(d\,j) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & d & 0 & 1 \end{pmatrix} \quad \text{and} \quad Trans(d\,k) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & d & 1 \end{pmatrix}.$$

3.2 Rotation. Let L be an axis of rotation defined by specifying a point Q on L and a unit direction vector u parallel to L, and let θ be the angle of rotation. Denote by $Rot(Q,u,\theta)$ the 4×4 matrix representing rotation of points around the line L through the angle θ, and denote by $Rot(u,\theta)$ upper 3×3 submatrix of $Rot(Q,u,\theta)$. Then by Equation (2.14) since Q remains fixed under rotation,

$$Rot(Q,u,\theta) = \begin{pmatrix} Rot(u,\theta) & 0 \\ Q*\big(I - Rot(u,\theta)\big) & 1 \end{pmatrix}. \tag{3.2}$$

Moreover by the Formula of Rodrigues (see Lecture 12, Section 6.2) for any vector v,

$$v^{new} = (\cos\theta)v + (1-\cos\theta)(v \bullet u)u + (\sin\theta)u \times v$$

Therefore we can find $Rot(u,\theta)$ by invoking Equation (2.9):

$$\begin{aligned} &v^{new} = (\cos\theta)v + (1-\cos\theta)(v \bullet u)u + (\sin\theta)(u \times v) \\ &\downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \\ &Rot(u,\theta) = (\cos\theta)\,I + (1-\cos\theta)(u^T * u) + (\sin\theta)(u \times \_)\,. \end{aligned} \tag{3.3}$$

Thus by Equations (2.5), (2.6), (2.7)

$$Rot(u,\theta) = \begin{pmatrix} \cos\theta + (1-\cos\theta)u_1^2 & (1-\cos\theta)u_1u_2 + (\sin\theta)u_3 & (1-\cos\theta)u_1u_3 - (\sin\theta)\,u_2 \\ (1-\cos\theta)u_1u_2 - (\sin\theta)u_3 & \cos\theta + (1-\cos\theta)\,u_2^2 & (1-\cos\theta)\,u_2u_3 + (\sin\theta)\,u_1 \\ (1-\cos\theta)\,u_1u_3 + (\sin\theta)u_2 & (1-\cos\theta)u_2u_3 - (\sin\theta)\,u_1 & \cos\theta + (1-\cos\theta)u_3^2 \end{pmatrix}. \tag{3.4}$$

Example 3.2: Suppose that the axis of rotation is the z-axis. Then we can choose $Q = Origin = (0,0,0)$ and $u = k = (0,0,1)$. Now by Equation (3.3)

$$Rot(k,\theta) = (\cos\theta)\,I + (1-\cos\theta)(k^T * k) + (\sin\theta)(k \times \_). \tag{3.5}$$

But by Equations (2.6) and (2.7)

$$k^T * k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad k \times \_ = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Therefore

$$Rot(k,\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Hence the matrix $Rot(Origin,k,\theta)$ representing rotation around the z-axis by the angle θ is given by

$$Rot(Origin,k,\theta) = \begin{pmatrix} Rot(k,\theta) & 0 \\ Origin*\big(I - Rot(k,\theta)\big) & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly, the matrices representing rotation around the x and y-axes by the angle θ are given by

$$Rot(Origin,i,\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$Rot(Origin,j,\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.3 Mirror Image. Let S be a mirror plane specified by a point Q on S and a unit vector n normal to S. Denote by $Mir(Q,n)$ the 4×4 matrix representing the mirror image of points in the plane S, and denote by $Mir(n)$ upper 3×3 submatrix of $Mir(Q,n)$. Then by Equation (2.14) since Q remains fixed under mirror image,

$$Mir(Q,n) = \begin{pmatrix} Mir(n) & 0 \\ Q*\big(I - Mir(n)\big) & 1 \end{pmatrix}. \tag{3.6}$$

Moreover, for mirror image (see Lecture 12, Section 6.2)

$$v^{new} = v - 2(v\bullet n)n.$$

Therefore we can find $Mir(n)$ by invoking Equation (2.9)

$$\begin{array}{ccccc} v^{new} & = & v & - & 2(v\bullet n)n \\ \downarrow & & \downarrow & & \downarrow \\ Mir(n) & & I & - & 2n^T * n. \end{array} \tag{3.7}$$

Now notice that

$$I - Mir(n) = 2n^T * n,$$

so

$$Q*\big(I - Mir(n)\big) = Q*(2n^T * n) = 2(Q\bullet n)n.$$

Therefore by Equations (3.6) and (3.7)

$$Mir(Q,n) = \begin{pmatrix} I - 2n^T * n & 0 \\ 2(Q\bullet n)n & 1 \end{pmatrix}. \tag{3.8}$$

Example 3.3: Suppose that the mirror plane is the xy-plane. Then we can choose $Q = Origin = (0,0,0)$ and $n = k = (0,0,1)$. By Equations (2.6)

$$k^T * k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore

$$Mir(Origin,k) = \begin{pmatrix} I - 2k^T * k & 0 \\ 2(Origin \bullet n)n & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly, the matrices representing mirror images in the xz-plane and yz-plane are

$$Mir(Origin,j) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad Mir(Origin,i) = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

4 Scaling

We are interested in both uniform and non-uniform scaling.

4.1 Uniform Scaling. Uniform scaling is defined by specifying a point Q from which to scale distance and a scale factor s. For any point P and any vector v (see Lecture 12, Section 6.2),

$$v^{new} = sv = v * (sI)$$
$$P^{new} = sP + (1-s)Q = P * (sI) + (1-s)Q.$$

Therefore the matrix $Scale(Q,s)$ representing uniform scaling about the point Q by the scale factor s is given by

$$Scale(Q,s) = \begin{pmatrix} sI & 0 \\ (1-s)Q & 1 \end{pmatrix}. \qquad (4.1)$$

Example 4.1: To scale uniformly about the origin by the scale factor s, set $Q = Origin = (0,0,0)$. Then by Equation (4.1)

$$Scale(Origin,s) = \begin{pmatrix} sI & 0 \\ (1-s)Origin & 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

4.2 Non-Uniform Scaling. Non-uniform scaling is defined by designating a unit vector u specifying the scaling direction, as well as a point Q about which to scale and a scale factor s. Denote by $Scale(Q,u,s)$ the 4×4 matrix representing non-uniform scaling from the point Q in the direction u by the scale factor s, and denote by $Scale(u,s)$ upper 3×3 submatrix of $Scale(Q,u,s)$. Then by Equation (2.14) since Q remains fixed under this scaling,

$$Scale(Q,u,s) = \begin{pmatrix} Scale(u,s) & 0 \\ Q*\big(I - Scale(u,s)\big) & 1 \end{pmatrix}. \tag{4.2}$$

Moreover, for non-uniform scaling (see Lecture 12, Section 6.2),

$$v^{new} = v + (s-1)(v \bullet u)u\,.$$

Therefore we can find $Scale(u,s)$ by invoking Equation (2.9)

$$\begin{array}{ccccc} v^{new} & = & v & + & (s-1)(v \bullet u)u \\ \downarrow & & \downarrow & & \downarrow \\ Scale(u,s) & = & I & + & (s-1)u^T * u\,. \end{array} \tag{4.3}$$

Now notice that

$$I - Scale(u,s) = (1-s)u^T * u,$$

so

$$Q*\big(I - Scale(u,s)\big) = (1-s)Q*(u^T * u) = (1-s)(Q \bullet u)u\,.$$

Therefore by Equations (4.2) and (4.3)

$$Scale(Q,u,s) = \begin{pmatrix} I + (s-1)u^T * u & 0 \\ (1-s)(Q \bullet u)u & 1 \end{pmatrix}. \tag{4.4}$$

Example 4.2: To scale about the origin in the direction of the z-axis by the scale factor s, we can choose $Q = Origin = (0,0,0)$ and $u = k = (0,0,1)$. By Equations (2.6)

$$k^T * k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore by Equation (4.4)

$$Scale(Origin,k,s) = \begin{pmatrix} I + (s-1)k^T * k & 0 \\ (1-s)(Origin \bullet k)k & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly, the matrices representing scaling about the origin along the x and y-axes are

$$Scale(Origin,i,s) = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad Scale(Origin,j,s) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

5. Projections

Orthogonal projection is an affine transformation, but perspective projection is not affine. Therefore in this section we will be concerned only with orthogonal projections. Parallel projections -- projections into a fixed plane parallel to a fixed direction -- are also affine. These projections are covered in Exercise 2. Perspective projections will be investigated in detail in Section 6.

5.1 Orthogonal Projection. Let the plane S into which points are projected be defined by a point Q and a unit normal vector n, and denote by $Orth(Q,n)$ the 4×4 matrix representing orthogonal projection into the plane S. The formulas for orthogonal projection are (see Lecture 12, Section 6.2)

$$v^{new} = v - (v \bullet n)n$$
$$P^{new} = P - \big((P - Q) \bullet n\big)n.$$

Notice that these formulas are almost exactly the same as the formulas for mirror image, except for the missing factor of 2. Therefore using the same analysis we used for mirror image, we arrive at

$$Ortho(Q,n) = \begin{pmatrix} I - n^T * n & 0 \\ (Q \bullet n)n & 1 \end{pmatrix}. \tag{5.1}$$

which is the same matrix as $Mir(Q,n)$ in Equation (3.8), except for the missing factors of 2.

Example 5.1: Suppose that the projection plane is the xy-plane. Then we can choose $Q = Origin = (0,0,0)$ and $n = k = (0,0,1)$. By Equations (2.6)

$$k^T * k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore

$$Ortho(Origin,k) = \begin{pmatrix} I - k^T * k & 0 \\ (Origin \bullet n)n & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Similarly, the matrices representing orthogonal projections into the xz-plane and yz-plane are

$$Ortho(Origin, j) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad Ortho(Origin, i) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

6. Perspective

Unlike all of the other transformations that we have investigated so far, perspective projection is not an affine transformation because perspective projection is not a well defined transformation at every point nor is perspective well defined for any vector. Nevertheless, we shall see shortly that we can still represent perspective projections by 4×4 matrices.

6.1 Projective Transformations and Homogeneous Coordinates. Perspective projection is defined by specifying an eye point $E = (e_1, e_2, e_3)$ and a plane S into which to project the image. The plane S is specified, as usual, by a point $Q = (q_1, q_2, q_3)$ on S and a unit vector $n = (n_1, n_2, n_3)$ normal to S.

The image of an arbitrary point $P = (x, y, z)$ under perspective projection from the eye point $E = (e_1, e_2, e_3)$ to the plane S is given by the expression (see Lecture 12, Section 6.2)

$$P^{new} = \frac{\big((E - Q) \bullet n\big)P + \big((Q - P) \bullet n\big)E}{(E - P) \bullet n}, \tag{6.1}$$

or in terms of coordinates

$$x^{new} = \frac{\big((e_1 - q_1)n_1 + (e_2 - q_2)n_2 + (e_3 - q_3)n_3\big)x + \big((q_1 - p_1)n_1 + (q_2 - p_2)n_2 + (q_3 - p_3)n_3\big)e_1}{-n_1x - n_2y - n_3z + e_1n_1 + e_2n_2 + e_3n_3}$$

$$y^{new} = \frac{\big((e_1 - q_1)n_1 + (e_2 - q_2)n_2 + (e_3 - q_3)n_3\big)y + \big((q_1 - p_1)n_1 + (q_2 - p_2)n_2 + (q_3 - p_3)n_3\big)e_2}{-n_1x - n_2y - n_3z + e_1n_1 + e_2n_2 + e_3n_3}$$

$$z^{new} = \frac{\big((e_1 - q_1)n_1 + (e_2 - q_2)n_2 + (e_3 - q_3)n_3\big)z + \big((q_1 - p_1)n_1 + (q_2 - p_2)n_2 + (q_3 - p_3)n_3\big)e_3}{-n_1x - n_2y - n_3z + e_1n_1 + e_2n_2 + e_3n_3}$$

Notice that since the point P appears in both the numerator and the denominator on the right hand side of Equation (6.1), the variables x, y, z appear in both the numerator and the denominator of the expressions for the coordinates $x^{new}, y^{new}, z^{new}$. Transformations where x, y, z appear in both the numerator and the denominator are called *projective transformations*. In order to represent perspective projection by matrix multiplication, we need to be able to represent projective transformations using matrix multiplication. So far we can only invoke matrices to represent linear and affine transformations.

Linear transformations are transformations of the form:

$$\begin{aligned} x^{new} &= a_{11}x + a_{21}y + a_{31}z \\ y^{new} &= a_{12}x + a_{22}y + a_{32}z \\ z^{new} &= a_{13}x + a_{23}y + a_{33}z \end{aligned} \quad \Leftrightarrow \quad (x \;\; y \;\; z) * \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

Linear transformations are readily represented by 3×3 matrices because matrix multiplication with rectangular coordinates generates 3 linear functions in the coordinates.

Affine transformations are transformations of the form:

$$\begin{aligned} x^{new} &= a_{11}x + a_{21}y + a_{31}z + a_{41} \\ y^{new} &= a_{12}x + a_{22}y + a_{32}z + a_{42} \\ z^{new} &= a_{13}x + a_{23}y + a_{33}z + a_{43} \end{aligned} \quad \Leftrightarrow \quad (x \;\; y \;\; z \;\; 1) * \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{2} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{pmatrix}.$$

Affine transformations cannot be represented by 3×3 matrices because matrix multiplication with rectangular coordinates cannot capture the constant (translation) terms. To represent affine transformations by matrices, we introduced affine coordinates for points and special 4×4 matrices, where the last column is $(0 \;\; 0 \;\; 0 \;\; 1)^T$, in order to capture the constant (translation) terms.

Projective transformations are transformations of the form:

$$x^{new} = \frac{a_{11}x + a_{21}y + a_{31}z + a_{41}}{a_{14}x + a_{24}y + a_{34}z + a_{41}}$$

$$y^{new} = \frac{a_{12}x + a_{22}y + a_{32}z + a_{42}}{a_{14}x + a_{24}y + a_{34}z + a_{41}}$$

$$z^{new} = \frac{a_{13}x + a_{23}y + a_{33}z + a_{43}}{a_{14}x + a_{24}y + a_{34}z + a_{41}}.$$

Projective transformations cannot be represented by either 3×3 or 4×4 matrices because matrix multiplication with either rectangular or affine coordinates cannot capture denominators. To represent projective transformations using matrices, we need to introduce a new device in order to express rational linear functions using matrix multiplication.

The device we shall adopt is called *homogeneous coordinates*, an extension of affine coordinates designed specifically to handle common denominators. So far we have permitted the fourth coordinate to be only 0 or 1; now we shall permit the fourth coordinate, the homogenous coordinate, to take on arbitrary values. By convention, if $w \neq 0$, then the four homogeneous coordinates (x, y, z, w) represent the point with rectangular coordinates $\left(x / w, y / w, z / w\right)$ -- that is, denominators are stored in the fourth coordinate.

We can now express projective transformations by 4 equations in the homogeneous coordinates (x,y,z,w):

$$\begin{aligned} x^{new} &= a_{11}x + a_{21}y + a_{31}z + a_{41} \\ y^{new} &= a_{12}x + a_{22}y + a_{32}z + a_{42} \\ z^{new} &= a_{13}x + a_{23}y + a_{33}z + a_{43} \\ w^{new} &= a_{14}x + a_{24}y + a_{34}z + a_{44} \end{aligned} \quad \Leftrightarrow \quad (x \;\; y \;\; z \;\; 1) * \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_2 & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Thus, using homogeneous coordinates, we can represent projective transformations by arbitrary 4×4 matrices, where the last column captures the denominator. Another advantage of homogeneous coordinates is that we can postpone division until we actually need a numerical result.

6.2 Matrices for Perspective Projections. Let us return now to perspective projection. Recall that the formula for perspective is

$$P^{new} = \frac{\big((E-Q)\bullet n\big)P + \big((Q-P)\bullet n\big)E}{(E-P)\bullet n}$$

where E is the eye point and S is the projection plane, specified by a point Q on S and a unit vector n normal to S. Using homogenous coordinates, we can place the denominator in the fourth coordinate and write

$$P^{new} = \Big(\big((E-Q)\bullet n\big)P + \big((Q-P)\bullet n\big)E,\ (E-P)\bullet n\Big)\,. \tag{6.2}$$

Expanding the terms on the right hand side of Equation (6.2) gives

$$P^{new} = \Big(\big((E-Q)\bullet n\big)P - (P\bullet n)E + (Q\bullet n)E,\ -P\bullet n + E\bullet n\Big)\,, \tag{6.3}$$

Now observe that

$$\begin{aligned} &\big((E-Q)\bullet n\big)P = P * \big((E-Q)\bullet n\big)I \\ &(P\bullet n)E \;\; = \;\; (P * n^T) * E \;\; = \;\; P * (n^T * E)\,. \end{aligned} \tag{6.4}$$

Therefore, by Equations (6.3) and (6.4), the numerator of P^{new} is

$$\big((E-Q)\bullet n\big)P - (P\bullet n)E + (Q\bullet n)E \;\; = P * \Big(\big((E-Q)\bullet n\big)I - (n^T * E)\Big) + (Q\bullet n)E\,.$$

Moreover, the denominator of P^{new} is

$$-P\bullet n + E\bullet n = P * (-n^T) + E\bullet n\,.$$

Therefore it follows that

$$(P^{new},1) \;\; = \;\; (P,1) * \begin{pmatrix} \big((E-Q)\bullet n\big)I - n^T * E & -n^T \\ (Q\bullet n)E & E\bullet n \end{pmatrix}.$$

Thus, with the convention that the fourth coordinate represents the denominator, the 4×4 matrix $Persp(E,Q,n)$ representing perspective projection from the eye point E to the plane S is given by

$$Persp(E,Q,n) = \begin{pmatrix} ((E-Q)\bullet n)I - n^T * E & -n^T \\ (Q\bullet n)E & E\bullet n \end{pmatrix}. \qquad (6.5)$$

Example 6.1: Suppose that the eye is located along the z-axis at $E = (0,0,1)$ and perspective plane is the xy-plane. Then we can choose $Q = Origin = (0,0,0)$ and $n = k = (0,0,1)$. By Equations (2.6)

$$k^T * E = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore by Equation (6.5)

$$Persp(E,Origin,k) = \begin{pmatrix} ((E-Origin)\bullet k)I - k^T * E & -k^T \\ (Origin\bullet k)E & E\bullet k \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Thus the projective transformation representing perspective projection into the xy-plane is

$$x^{new} = x,\ y^{new} = y,\ z^{new} = 0,\ w^{new} = 1-z \ \Leftrightarrow\ x^{new} = x/1-z,\ y^{new} = y/1-z,\ z^{new} = 0.$$

7. Summary

Below we summarize the main themes of this lecture. Also listed below for your convenience are the matrix representations for each of the transformations that we have studied in this lecture.

7.1 Matrix Representations for Affine and Projective Transformations. The core content of this lecture are 4×4 matrix representations for each of the basic affine and projective transformations -- translation, rotation, mirror image, uniform and non-uniform scaling, and orthogonal and perspective projection -- of 3-dimensional Computer Graphics.

Every affine transformation A is completely determined by the four values $A(O), A(i), A(j), A(k)$, and these values form the rows of the matrix representation for A -- that is

$$A = \begin{pmatrix} A(i) & 0 \\ A(j) & 0 \\ A(k) & 0 \\ A(O) & 1 \end{pmatrix}.$$

Thus, matrix representations for affine transformations all have the form:

$$A = \begin{pmatrix} L_A & 0 \\ w_A & 1 \end{pmatrix} = \begin{pmatrix} 3\times 3\ Linear\ Transformation\ Matrix & 0 \\ Translation\ Vector & 1 \end{pmatrix}.$$

The upper 3×3 matrix

$$L_A = \begin{pmatrix} A(i) & 0 \\ A(j) & 0 \\ A(k) & 0 \end{pmatrix}$$

can be constructed from three basic 3×3 matrices: I, $u^T * \tilde{u}$, $u\times\_$, and the vector w_A can typically be expressed in terms of a fixed point Q of the transformation A. In particular,

$$v^{new} = c_1 v + c_2 (v \bullet u)\tilde{u} + c_3 u \times v \;\Rightarrow\; L_A = c_1 I + c_2 u^T * \tilde{u} + c_3 u\times\_$$

and

$$w_A = Q*(I - L_A)\ .$$

Perspective projection is a projective transformation, not an affine transformation. Affine transformations have the linear form

$$\begin{aligned} x^{new} &= a_{11}x + a_{21}y + a_{31}z + a_{41} \\ y^{new} &= a_{12}x + a_{22}y + a_{32}z + a_{42} \\ z^{new} &= a_{13}x + a_{23}y + a_{33}z + a_{43} \end{aligned} \quad \Leftrightarrow \quad (x\ \ y\ \ z\ \ 1)*\begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_2 & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{pmatrix},$$

whereas projective transformations have the rational linear form

$$\begin{aligned} x^{new} &= \frac{a_{11}x + a_{21}y + a_{31}z + a_{41}}{a_{14}x + a_{24}y + a_{34}z + a_{41}} \\ y^{new} &= \frac{a_{12}x + a_{22}y + a_{32}z + a_{42}}{a_{14}x + a_{24}y + a_{34}z + a_{41}} \\ z^{new} &= \frac{a_{13}x + a_{23}y + a_{33}z + a_{43}}{a_{14}x + a_{24}y + a_{34}z + a_{41}} \end{aligned} \quad \Leftrightarrow \quad (x\ \ y\ \ z\ \ 1)*\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_2 & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}.$$

Perspective projections can still be represented by 4×4 matrices, but the last column is no longer $(0,0,0,1)^T$; instead the last column represents the common denominator To realize this representation, we adopted a new convention called *homogeneous coordinates* for representing points in 3-dimensions. By convention, if $w \neq 0$, then

$$(x,y,z,w) \Leftrightarrow \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1\right).$$

Thus the fourth coordinate, the homogeneous coordinate, represents a common denominator. In the next lecture we shall look more closely at the intrinsic geometric meaning behind this formal algebraic convention.

7.2 Matrices for Affine and Projective Transformations.

Translation -- by the vector w

$$Trans(w) = \begin{pmatrix} I & 0 \\ w & 1 \end{pmatrix}$$

Rotation -- around the line L determined by the point Q and the unit direction vector u by the angle θ

$$Rot(Q,u,\theta) = \begin{pmatrix} Rot(u,\theta) & 0 \\ Q*\left(I - Rot(u,\theta)\right) & 1 \end{pmatrix}$$

:
$$Rot(u,\theta) = (\cos\theta)\, I + (1-\cos\theta)(u^T * u) + (\sin\theta)(u \times \_)$$

Mirror Image -- in the plane S determined by the point Q and the unit normal n

$$Mir(Q,n) = \begin{pmatrix} I - 2\, n^T * n & 0 \\ 2(Q \bullet n)n & 1 \end{pmatrix}$$

Uniform Scaling -- around the point Q by the scale factor s

$$Scale(Q,s) = \begin{pmatrix} sI & 0 \\ (1-s)Q & 1 \end{pmatrix}$$

Non-Uniform Scaling -- around the point Q by the scale factor s in the direction u

$$Scale(Q,u,s) = \begin{pmatrix} I + (s-1)\, u^T * u & 0 \\ (1-s)(Q \bullet u)u & 1 \end{pmatrix}$$

Orthogonal Projection -- into the plane S determined by the point Q and the unit normal n

$$Ortho(Q,n) = \begin{pmatrix} I - n^T * n & 0 \\ (Q \bullet n)n & 1 \end{pmatrix}$$

Perspective Projection -- from the eye point E into the plane S determined by the point Q and the unit normal vector n

‘
$$Persp(E,Q,n) = \begin{pmatrix} \left((E-Q) \bullet n\right)I - n^T * E & -n^T \\ (Q \bullet n)E & E \bullet n \end{pmatrix}$$

Exercises:

1. Consider the shear transformation introduced in Lecture 12, Exercise 2. Let

   S = Shearing plane

   n = Unit vector perpendicular to S

   Q = Point on S

   u = Unit vector in S (i.e. unit vector perpendicular to n)

   ϕ = Shear angle

   Recall that the formulas for shear are

   $$v^{new} = v + \tan(\phi)(v \bullet n)u$$

   $$P^{new} = P + \tan(\phi)\big((P - Q) \bullet n\big)u$$

   a. Show that the 4×4 matrix representing shear is given by

   $$Shear(Q,n,u,\phi) = \begin{pmatrix} I + (tan\phi)(n^T * u) & 0 \\ -tan\phi(Q \bullet n)u & 1 \end{pmatrix}.$$

   b. Suppose that

      i. S is the xy-plane,

      ii. u is the unit vector along the x-axis,

      iii. $\phi = 45^\circ$

      Write down the explicit entries of the 4×4 matrix $Shear(Q,n,u,\phi)$.

2. Recall from Lecture 12, Exercise 3 that parallel projection is projection onto a plane S along a direction u. As usual the plane S is defined by a point Q and a unit normal vector n Recall that the formulas for parallel projection are

   $$v^{new} = v - \frac{v \bullet n}{u \bullet n}u$$

   $$P^{new} = P - \frac{(P - Q) \bullet n}{u \bullet n}u$$

   a. Show that the 4×4 matrix representing parallel projection is given by

   $$PProj(Q,n,u) = \begin{pmatrix} I - \dfrac{n^T * u}{u \bullet n} & 0 \\ \dfrac{Q \bullet n}{u \bullet n}u & 1 \end{pmatrix}.$$

   b. Suppose that

      i. S is the *xy-plane*,

      ii. u is the unit vector parallel to the vector $(1,1,1)$,

      Write down the explicit entries of the 4×4 matrix $PProj(Q,n,u)$.

3. Verify that:
   a. $Rot(u,\theta)^{-1} = Rot(u,-\theta) = Rot(u,\theta)^T$
   b. $Mir(n)^{-1} = Mir(n) = Mir(n)^T$

4. Verify that:
   a. $Det\big(Rot(u,\theta)\big) = 1$
   b. $Det\big(Mir(n)\big) = -1$

5. Verify that:
   a. u is an eigenvector of $Rot(u,\theta)$ corresponding the the eigenvalue 1.
   b. n is an eigenvector of $Mir(n)$ corresponding the the eigenvalue -1.

   Interpret the results in parts *a,b* geometrically.

6. The trace of a matrix is the sum of the diagonal entries. Thus, for a 3×3 matrix M:

   $Trace(M) = m_{11} + m_{22} + m_{33}$
   a. Show that:
      i. $Trace(M+N) = Trace(M) + Trace(N)$
      ii. $Trace(c\,M) = c\,Trace(M)$
   b. Using part a and Equation (3.3), conclude that

   $$\cos(\theta) = \frac{Trace\big(Rot(u,\theta)\big) - 1}{2}.$$

7. Consider a rotation matrix $Rot(u,\theta)$.
   a. Using Equation (3.3) or (3.4), show that we can recover the unit vector u parallel to the axis of rotation and the sine of the angle of rotation from the rotation matrix $Rot(u,\theta)$ by the formula

   $$2sin(\theta)\,u = \big(Rot(u,\theta)_{2,3} - Rot(u,\theta)_{3,2},\ Rot(u,\theta)_{3,1} - Rot(u,\theta)_{1,3},\ Rot(u,\theta)_{1,2} - Rot(u,\theta)_{2,1}\big)$$
   b. Explain why we cannot solve independently for the axis of rotation u and the angle of rotation θ.

8. Let M be a 3×3 matrix, and let v,w be arbitrary vectors.
   a. Show that

   $(v * M) \bullet w = v \bullet (w * M^T)$
   b. Conclude from part a and Exercise 3 that
      i. $\big(v * Rot(u,\theta)\big) \bullet \big(w * Rot(u,\theta)\big) = v \bullet w$
      ii. $\big(v * Mir(n)\big) \bullet \big(w * Mir(n)\big) = v \bullet w$
   c. Interpret the results in part *b* geometrically.

9. Let M be a 3×3 matrix, and let v,w be arbitrary vectors.
   a. Show that
   $$(u * M) \times (v * M) = Det(M)(u \times v) * M^{-T}$$
   b. Conclude from part *a* and Exercises 3,4 that
      i. $\big(v * Rot(u,\theta)\big) \times \big(w * Rot(u,\theta)\big) = v \times w$
      ii. $\big(v * Mir(n)\big) \times \big(w * Mir(n)\big) = -v \times w$
   c. Interpret the results in part *b* geometrically.

10. Let v_1,v_2 be unit vectors, and let $u = v_1 \times v_2$.
   a. Show that the matrix
   $$Rot(u) = (v_1 \bullet v_2)I + \frac{u^T * u}{1 + v_1 \bullet v_2} + u \times \_,$$
   rotates the vector v_1 into the vector v_2.
   b. Interpret this result geometrically.

11. Show that $Mir(Q,n) = Scale(Q,n,-1)$. Interpret this result geometrically.

12. Show that $Mir(n) = -Rot(n,\pi)$. Interpret this result geometrically.

13. Let $I_{4\times 4}$ be the 4×4 identity matrix. Show that $Ortho(Q,n) = I_{4\times 4} + Mir(Q,n)$.

14. Let *A* be a nonsingular affine transformation. Show that
   a. *A* maps straight lines to straight lines.
   b. *A* preserves ratios of distances along straight lines -- that is, if P,Q,R are collinear, then
   $$\frac{|A(R) - A(P)|}{|A(Q) - A(P)|} = \frac{|R - P|}{|Q - P|}.$$

15. Let *T* be a nonsingular projective transformation. Show that
   a. *T* maps straight lines to straight lines.
   b. *T* does not necessarily preserve ratios of distances along straight lines. That is, give an example to show that if P,Q,R are collinear, then
   $$\frac{|T(R) - T(P)|}{|T(Q) - T(P)|} \neq \frac{|R - P|}{|Q - P|}.$$

16. The cross ratio of four collinear points P,Q,R,S is defined by
   $$cr(P,Q,R,S) = \frac{|R - P|\ |S - Q|}{|S - P|\ |R - Q|}.$$

Show that if T is a projective transformation, then T preserves cross ratios -- that is, show that $cr\big(T(P),T(Q),T(R),T(S)\big) = cr(P,Q,R,S)$.

17. Let P be a point, and let v_1,v_2,v_3 be three linearly independent vectors in 3-space.
    a. Show that in 3-dimensions there is a unique affine transformation A that maps P,v_1,v_2,v_3 to Q,w_1,w_2,w_3.
    b. Show that if u is an arbitrary vector, then
    $$A(u) = \frac{Det\begin{pmatrix} u & v_2 & v_3 \end{pmatrix} w_1 + Det\begin{pmatrix} v_1 & u & v_3 \end{pmatrix} w_2 + Det\begin{pmatrix} v_1 & v_2 & u \end{pmatrix} w_3}{Det\begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}}.$$
    c. Verify that the matrix representing this affine transformation A is given by
    $$A = \begin{pmatrix} v_1 & 0 \\ v_2 & 0 \\ v_3 & 0 \\ P & 1 \end{pmatrix}^{-1} * \begin{pmatrix} w_1 & 0 \\ w_2 & 0 \\ w_3 & 0 \\ Q & 1 \end{pmatrix}.$$
    d. Show that Equation (1.1) is a special case of the matrix in part *c*.

18. Let P_1,P_2,P_3,P_4 be four non-collinear points in 3-space.
    a. Show that in 3-dimensions there is a unique affine transformation A that maps four non-collinear points P_1,P_2,P_3,P_4 to four arbitrary points Q_1,Q_2,Q_3,Q_4.
    b. Show that if R is an arbitrary point, then
    $$A(R) = \frac{Det\begin{pmatrix} R & 1 \\ P_2 & 1 \\ P_3 & 1 \\ P_4 & 1 \end{pmatrix} Q_1 + Det\begin{pmatrix} P_1 & 1 \\ R & 1 \\ P_3 & 1 \\ P_4 & 1 \end{pmatrix} Q_2 + Det\begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ R & 1 \\ P_4 & 1 \end{pmatrix} Q_3 + Det\begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \\ R & 1 \end{pmatrix} Q_4}{Det\begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \\ P_4 & 1 \end{pmatrix}}.$$
    c. Verify that the matrix representing this affine transformation A is given by
    $$A = \begin{pmatrix} P_1 & 1 \\ P_2 & 1 \\ P_3 & 1 \\ P_4 & 1 \end{pmatrix}^{-1} * \begin{pmatrix} Q_1 & 1 \\ Q_2 & 1 \\ Q_3 & 1 \\ Q_4 & 1 \end{pmatrix}.$$

Programming Projects:

1. *The Turtle in 3-Dimensions*

In two dimensions the turtle's state consists of a point P specifying her location and a vector u specifying both her heading and her step size. To navigate in 3-dimensions, the turtle carries around not just one vector u, but three mutually orthogonal vectors u,v,w specifying her heading u, her left hand v, and her up direction w. Thus in 3-dimensions the turtle carries around her own 3-dimensional local coordinate system. The FORWARD and MOVE commands work much as before, but in place of the TURN command, there are three new turtle commands: YAW, PITCH, and ROLL. YAW represent a rotation in the $uv-plane$, the plane perpendicular to the up direction w; PITCH represents a rotation in the $wu-plane$, the plane perpendicular to the left hand vector v; and PITCH represents a rotation in the $vw-plane$, the plane perpendicular to the forward direction u. Also the RESIZE command is redefined to apply simultaneously to all three turtle vectors instead of just one turtle vector. Thus in 3-dimensions, LOGO has the following commands:

- *FORWARD D and MOVE D -- Translation*
  $P_{new} = P + D\,u$
- *YAW A -- Rotation* in $uv-plane$
  $u_{new} = u\cos(A) + v\sin(A)$
  $v_{new} = -u\sin(A) + v\cos(A)$
  $\Leftrightarrow \quad \begin{pmatrix} u_{new} \\ v_{new} \end{pmatrix} = \begin{pmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{pmatrix} * \begin{pmatrix} u \\ v \end{pmatrix}$
- *PITCH A -- Rotation* in $wu-plane$
  $w_{new} = w\cos(A) + u\sin(A)$
  $u_{new} = -w\sin(A) + u\cos(A)$
  $\Leftrightarrow \quad \begin{pmatrix} w_{new} \\ u_{new} \end{pmatrix} = \begin{pmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{pmatrix} * \begin{pmatrix} w \\ u \end{pmatrix}$
- *ROLL A -- Rotation* in $vw-plane$
  $v_{new} = v\cos(A) + w\sin(A)$
  $w_{new} = -v\sin(A) + w\cos(A)$
  $\Leftrightarrow \quad \begin{pmatrix} v_{new} \\ w_{new} \end{pmatrix} = \begin{pmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{pmatrix} * \begin{pmatrix} v \\ w \end{pmatrix}$
- *RESIZE s -- Uniform Scaling*
  $u_{new} = s\,u$
  $v_{new} = s\,v$
  $w_{new} = s\,w$
  $\Leftrightarrow \quad \begin{pmatrix} u_{new} \\ v_{new} \\ w_{new} \end{pmatrix} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix} * \begin{pmatrix} u \\ v \\ w \end{pmatrix}$

a. Implement the 3-dimensional turtle in your favorite programming language using your favorite API.
b. Write LOGO programs to generate the edges of simple polyhedra in 3-dimensions.
c. Write recursive LOGO programs to create your own novel fractals in 3-dimensions.

2. *The Turtle on the Surface of a Sphere*

The classical turtle walks on the surface of an infinite plane. But we could imagine that the turtle, like us, is actually walking on the surface of a large sphere. In this environment, when the turtle executes the command FORWARD *D*, the turtle draws the arc of a great circle on the sphere. The FORWARD and TURN commands correspond to different rotations of the sphere beneath the turtle or equivalently different rotations of the turtle in 3-space. Since the turtle never leaves the surface of the sphere, her position can be stored as a vector *u* of a fixed length from the center of the sphere. Hence the turtle's state consist of two orthogonal vectors: a position vector *u* of a fixed length and a direction vector *v*, perpendicular to the position vector *u*, tangent to the surface of the sphere. Thus for the turtle on the surface of a sphere, LOGO consists of the following commands:

- *FORWARD D and MOVE D -- Translation along the Sphere = Rotation in the uv-Plane*

$$u = |u|\left(\cos(D|v|)\frac{u}{|u|} + \sin(D|v|)\frac{v}{|v|}\right)$$
$$v = |v|\left(-\sin(D|v|)\frac{u}{|u|} + \cos(D|v|)\frac{v}{|v|}\right) \quad \Leftrightarrow \quad \begin{pmatrix} u_{new} \\ v_{new} \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} * rot\left(\frac{u \times v}{|u \times v|}, D|v|\right)$$

- *TURN A -- Rotation Around the Vector u*

$$v = |v|\left(\cos(A)\frac{v}{|v|} + \sin(A)\frac{u \times v}{|u \times v|}\right) \quad \Leftrightarrow \quad v_{new} = v * rot\left(\frac{u}{|u|}, A\right)$$

- RESIZE *S*

$$v_{new} = S\,v$$

a. Implement LOGO on the surface of a sphere in your favorite programming language using your favorite API.
b. Write turtle programs to generate simple curves like polygons, stars, and spirals on the surface of a sphere.
c. Write recursive turtle programs to generate fractals on the surface of a sphere.

3. *CODO in 3-Dimensions*

a. Implement CODO in 3-dimensions in your favorite programming language using your favorite API, incorporating the 3-dimensional affine transformations discussed in the text and the exercises of this lecture.
b. Write CODO programs to generate the edges of simple polyhedra in 3-dimensions.
c. Develop iterated function systems to create fractals in 3-dimensions with and without condensation sets. In particular, develop iterated function systems to generate:
   i. 3-dimensional gaskets
   ii. 3-dimensional snowflakes
   iii. 3-dimensional trees
   iv. 3-dimensional fractals of your own novel design

4. *Parser*

a. *Let* $S = \begin{pmatrix} u & v & w & P \\ 0 & 0 & 0 & 1 \end{pmatrix}^T$ denote the state of the 3-dimensional turtle (see Project 1). Show that the LOGO commands for the 3-dimensional turtle can be implemented in the following fashion:

Command	*Exocentric*	*Turtle-Centric*
FORWARD D	$S_{new} = S * Trans(D\,u)$	$S_{new} = Trans(D\,i) * S$
MOVE D	$S_{new} = S * Trans(D\,u)$	$S_{new} = Trans(D\,i) * S$
YAW A	$S_{new} = S * Rot(P, w/\lvert w \rvert, A)$	$S_{new} = Rot(Origin, k, A) * S$
PITCH A	$S_{new} = S * Rot(P, v/\lvert v \rvert, A)$	$S_{new} = Rot(Origin, j, A) * S$
ROLL A	$S_{new} = S * Rot(P, u/\lvert u \rvert, A)$	$S_{new} = Rot(Origin, i, A) * S$
RESIZE s	$S_{new} = S * Scale(P, s)$	$S_{new} = Scale(Origin, s) * S$

b. Write a parser to convert recursive programs for the 3-dimensional turtle into iterated function systems that generate the same fractal.

c. Give examples to illustrate the results of your parser.

5. *The 3-Dimensional Turtle on a Bounded Domain*

The 3-dimensional turtle lives in an infinite space. Suppose, however, that the 3-dimensional turtle is restricted to a finite domain bounded by walls. When the turtle hits a wall, she bounces off the wall so that her angle of incidence with the normal vector to the wall is equal to her angle of reflection, and then she continues on her way (see Lecture 2, Project 3: *The Classical Turtle on a Bounded Domain)*. To implement the 3-dimensional turtle on a bounded domain, the FORWARD command must be replaced by

NEWFORWARD *D*

D_1 = *Distance from Turtle to Wall in Direction of Turtle Heading*

IF $D < D_1$, FORWARD *D*

OTHERWISE

FORWARD D_1

ROTATE $A = \begin{pmatrix} u & v & w \\ 0 & 0 & 0 \end{pmatrix}^T * Rot\left(\frac{n \times u}{\lvert n \times u \rvert}, A\right)$ { A = *angle of reflection* }

NEWFORWARD $D - D_1$

Here n is the normal to the wall at the point of impact, and $\cos(A) = \frac{u \bullet n}{\lvert u \rvert \lvert n \rvert}$.

a. Implement LOGO on a bounded domain, where the walls form:
   i. a rectangular box
   ii. a sphere
   iii. an ellipsoid
b. Consider the turtle program consisting of the single command
   NEWFORWARD *D*
   Investigate the curves generated by this program for different shape walls, different values of *D,* and different initial positions and headings for the turtle.
c. Investigate how curves drawn by the walled in turtle differ from curves drawn by the turtle in infinite space using the same turtle programs, where FORWARD is replaced by NEWFORWARD.

6. *The 3-Dimensional Turtle in a 3-Dimensional Manifold.*

The 3-dimensional turtle lives in a flat 3-dimensional space. Here we shall investigate turtles that live on curved 3-dimensional manifolds. These manifolds can be modeled by rectangular boxes, where pairs of opposite sides -- front and back, left and right, top and bottom -- are glued together, possibly with a twist (see Lecture 2, Project 5: *The Turtle on 2-Dimensional Manifolds,* for the analogous 2-manifolds modeled by rectangles). When the turtle encounters a wall, she does not bounce off the wall like the turtle on a bounded domain (Project 5); rather she emerges on the opposite wall of the box heading in the same direction relative to the new wall in which she hit the opposing wall.

a. Implement LOGO on one or more different 3-manifolds.
   Note: For each of these 3-manifolds, you will need to define a command MFORWARD to replace of the command FORWARD in standard LOGO (see Project 5).
b. Consider the turtle program consisting of the single command
   MFORWARD *D*
   Investigate the curves generated by this program for different 3-manifolds, different values of *D,* and different initial positions and headings for the turtle.
c. Investigate how curves drawn by turtles on these 3-dimensional manifolds differ from curves drawn by the turtle in an infinite space using the same turtle programs, where FORWARD is replaced by MFORWARD.

7. *Linear Algebra Package for Computer Graphics*

Implement a basic linear algebra package for Computer Graphics. Your package should include the following:

a. Algebra of Points and Vectors
   Coordinate implementations for addition, subtraction, scalar multiplication, dot product, cross product, and determinant.
b. Affine and Projective Transformations

Matrix implementations for translation, rotation, mirror image, uniform and non-uniform scaling, shear, perspective projection, and composite transformations.

c. Distance Formulas
   Vector algebra implementations for the distance between two points, a point and a line, a point and a plane, and two parallel or two skew lines.
d. Intersection Computations
   Vector algebra implementations for the intersections of two lines, two planes, or three planes.

8. *Animation*
   a. Build a collection of 3-dimensional polyhedral models.
   b. Animate the models using the transformations developed in this lecture.