A Descriptive Study of 2-D, 3-D Graphics And Transformations

Dr.S.V.B. Subrahmanyeswara Rao [1], Mrs.Yalamanchili.Anjani [2]
[1, 2] Assoc. Professor,
[1]Ramachandra College of Engineering, Eluru,A.P.India,
[2]VKR,VNB& AGK College of Engineering, Gudivada, A.P.India

Abstract- *Computer Graphics is transforming our world significantly and rapidly. Computer Graphics is itself a very broad term and describes everything on the computer which is not text or sound. It can be a video, image, animation or anything else. It can be anything but everything is made up of 'pixel'. Pixel is the smallest unit of computer graphics. Computer Graphics is an art of drawing line, object, pictures etc. with the help of computer programming. There are two types of computer graphics: Interactive Graphics and Non-Interactive Graphics. Interactive graphics are those which have a two-way communication and the operator can control the operations like playing a video game on a computer. Non-interactive graphics are those in which where the user doesn't have any kind of control over the graphics.*

Keywords- Transformations, Translation, Scaling, Rotation, Reflection, Shearing, Computer Graphics, Matrix representations.

I. INTRODUCTION

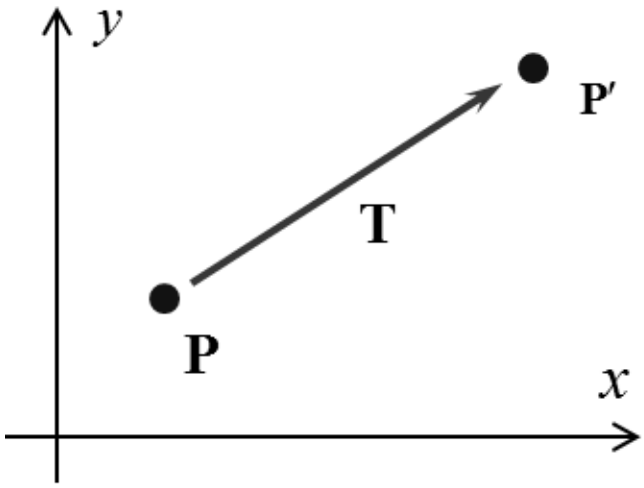
Computer Graphics definitely looks stunning and are often very complicated and takes a lot of time and effort to make. The techniques and hardware used today are very advanced and has automated most of the things. But this was not the case always. The development of Computer Graphics started in 1950's when it was designed using the vector graphics. Times changed and the methods also changed. Raster graphics comes into play and make things interesting and easy. But the problem was it doesn't look real and that's what stopped it being the mainstream technology. Nowadays, 3D graphics are used to design everything due to extra features of depth and originality. They are not very much different than 2D graphics apart from the fact that they have an extra dimension. 3D graphics rely on similar algorithms which are used to design 2D graphics. Even 2D graphics can also achieve 3D effects by using different techniques like lightening. The animation is one step ahead 3D graphics where moving objects or graphics are used. No matter how much complicated and advanced the graphic is, it will be using the same basic algorithms and transformation techniques to get the work done.

II. 2-D GEOMETRICAL TRANSFORMATIONS

Changes in orientation, size and shape are accomplished with Geometric transformations that change the coordinate descriptions of objects

Basic Transformations

We describe the general procedures for applying translation, rotation, and scaling parameters to reposition and resize two-dimensional objects



Translation

It is a process of changing position of an object along a straight-line path from one coordinate location to another.
We translate a two-dimensional point by adding translation distances, t_x, and t_y to the original coordinate position (x, y) to move the point to a new position (x', y')

$$x' = x + t_x$$
$$y' = y + t_y$$

The translation distance pair (t_x, t_y) is called a translation vector or shift vector.

Expressing the Translation equations as a single matrix equation by using column vectors to represent coordinate positions and the translation vector

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

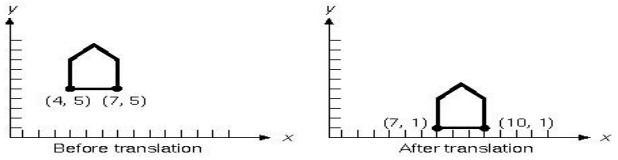
This allows us to write the two-dimensional translation equations in the matrix form:

$$P' = P + T$$

Note: Matrix-Transformation equations are expressed in terms of coordinate row vectors instead of column vectors.

$$P = [x_1 \quad x_2] \qquad P' = [x_1' \quad x_2'] \qquad T = [t_x \quad t_y]$$

Translation is a Rigid-body transformation that moves objects without deformation. That is, every point on the object is translated by the same amount



(a) (b)

Fig.2: Moving a polygon from Position(a) to Position(b) with translation vector (3,- 4)

Similar methods are used to translate curve objects

Rotation

A two-dimensional rotation is applied to an object by repositioning it along a circular path in the *xy* plane.

To generate a rotation, we specify a rotation angle θ and position of rotation point about which the object is to be rotated

Positive values for the rotation angle define counter clockwise rotations about the pivot point and negative values rotate objects in the clockwise direction.

Let us consider rotation of the object about the origin

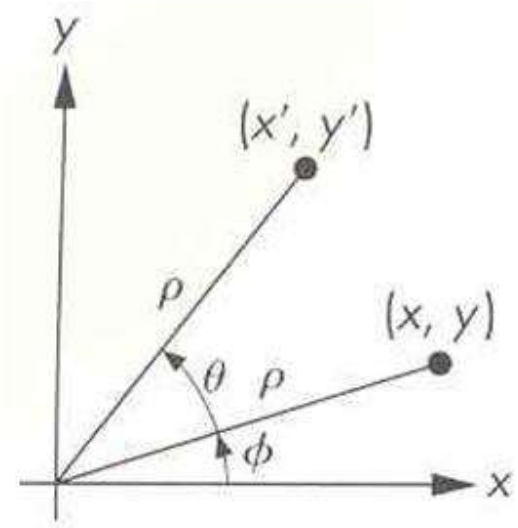


Fig.3: Rotation of a point from position (x, y) to position (x', y') through an angle θ relative to coordinate origin

r is the constant distance of the point from the origin,

Angle $\emptyset$ is the original angular position of the point from the horizontal, and θ is the rotation angle.

Using standard trigonometric identities in terms of angles

$$x' = r\cos(\emptyset + \theta) = r\cos\emptyset\cos\theta - r\sin\emptyset\sin\theta$$

$$y' = r\sin(\emptyset + \theta) = r\cos\emptyset\sin\theta + r\sin\emptyset\cos\theta$$

Original coordinates of the point in polar coordinates are

$$x = r\cos\emptyset, \quad y = r\sin\emptyset$$

Substituting expressions (2) into (1), we get the transformation equations for rotating a point at position (x, y) through an angle θ about the origin:

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

1. By using the column-vector representations for coordinate positions, we can write the rotation equations in the matrix form:

$$P' = R.P$$

where the rotation matrix is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2.When coordinate positions are represented as row vectors instead of column vectors, the matrix product in rotation equation is transposed so that the transformed row coordinate vector $[x' \quad y']$ is calculated as

$$P'^{T} = (R.P)^{T} = P^{T}.R^{T}$$

Where $P^{T} = [x \quad y]$

Transpose R^{T} of matrix R is obtained by interchanging rows and columns.

Note: For a rotation matrix, the transpose is obtained by simply changing the sign of the sine terms.

Rotation of a point about an arbitrary pivot position

By using trigonometric relationships, we can obtain transformations equations, for rotation of a point about any specified rotation point (x_r, y_r)

$$x' = x_r + (x - x_r)\cos\theta + (y - y_r)\sin\theta$$
$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

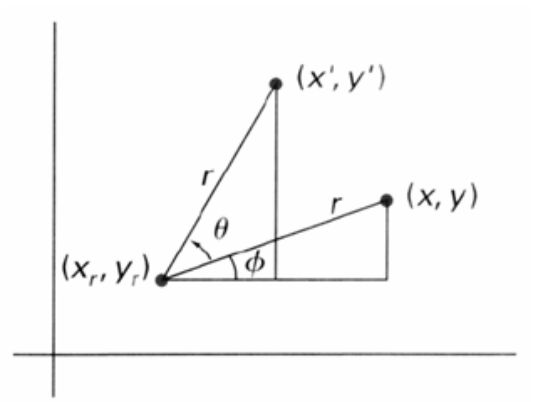


Fig.4: Rotation of a point from position (x, y) to position (x', y') through an angle θ about rotation point (x_r, y_r)

Rotations are rigid-body transformations that move objects without deformation. Every point on an object is rotated through the same angle.

Scaling

A scaling transformation changes the size of an object.

This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors s_x and s_y to produce the transformed coordinates (x', y')

$$x' = x.s_x$$
$$y' = y.s_y$$

Scaling factor s_x scales objects in the *x* direction, while s_y scales in the *y* direction.

The transformation equations can also be written in the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = S.P$$

Here S is 2 by 2 scaling matrix

Any positive numeric values can be assigned to scaling factors s_x and s_y. Scaling factors values greater than 1 produce an enlargement and move objects farther from origin

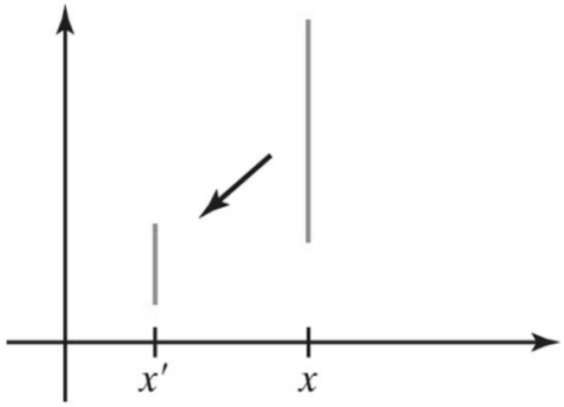


Fig.5:Line scaled using $s_x = s_y = 0.5$ 1.Reduce in size 2.Moved closer to coordinate origin

Scaling factors values less than 1 reduce size of objects closer to coordinate origin. Specifying a value of 1 for both s_x and s_y leaves size of objects unchanged

When s_x and s_y are assigned the same value, a Uniform Scaling is produced that maintains relative object proportions

Unequal values for s_x and s_y result in a differential scaling that is often used in design applications

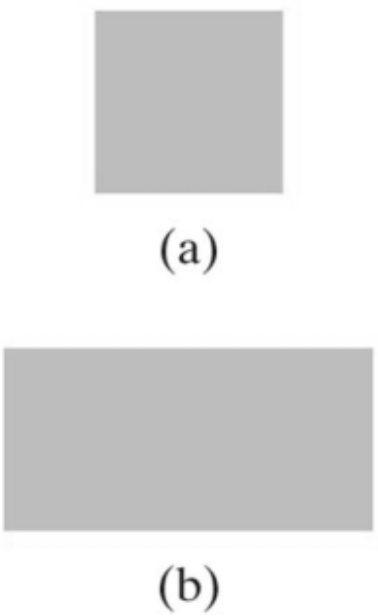


Fig.6: Turning square into rectangle $S_x = 2\ S_y = 2$

One can control the location of a scaled object by choosing a position, called the *Fixed Point*, that is to remain unchanged after the scaling transformation.

Coordinates for the fixed point (x_f, y_f) can be chosen as one of the vertices, the object centroid, or any other position

A polygon is then scaled relative to the fixed point by scaling the distance from each vertex to the fixed point.

For a vertex with coordinates (x, y), the scaled coordinates (x', y') are calculated as

$$x' = x_f + (x - x_f)s_x$$
$$y' = y_f + (y - y_f)s_y$$

Rewrite these scaling transformations to separate. the multiplicative and additive terms:

$$x' = x.s_x + x_f(1 - s_x)$$
$$y' = y.s_y + y_f(1 - s_y)$$

where the additive terms $x_f(1 - s_x)$ and $y_f(1 - s_y)$ are constant for all points in the object.

Polygons are scaled by applying above transformations to each vertex and then regenerating the polygon using the transformed vertices.

III. MATRIX REPRESENTATIONS AND HOMOGENEOUS COORDINATES

In Design and picture formation process, many times we must require to perform Translations, Rotations and Scaling to fit the picture components into their proper positions

In the previous section we have seen that each of the basic transformations can be expressed in general matrix form

$$P' = M_1.P + M_2$$

with coordinate positions P and P' represented as column vectors.

Matrix M_1 is a 2 by 2 array containing multiplicative factors, and M_2 is a two-element column matrix containing translational terms.

For translation, M_1 is the identity matrix.

For rotation or scaling, M_2 contains the translational terms associated with the pivot point or scaling fixed point.

To produce a sequence of transformations with these equations, such as scaling followed by rotation then translation, we must calculate the transformed coordinates one step at a time.

First, coordinate positions are scaled, then these scaled coordinates are rotated, and finally the rotated coordinates are translated.

A more efficient approach would be to combine the transformations so that the final coordinate positions are obtained directly from the initial coordinates. This eliminates the calculation of intermediate coordinate values.

To express any two-dimensional transformation as a matrix multiplication, we represent each Cartesian coordinate (x, y) with the homogeneous coordinate triple (x, y, h)

Where

$$x = \frac{x_h}{h} \qquad y = \frac{y_h}{h}$$

Thus, a general homogeneous coordinate representation can also be written as $(h.x, h.y, h)$.

For two-dimensional geometric transformations, we can choose the homogeneous parameter *h* to be any nonzero value.

Thus, there is an infinite number of equivalent homogeneous representations for each coordinate point (x, y).

Note: Set $h = 1$ is a convenient choice.

The term homogeneous coordinate is used in Mathematics

When a Cartesian point $(x, y$) is converted to a homogeneous representation (x_h, y_h, h)equations containing *x* and *y* such as

$f(x, y) = 0$

Above equation become homogeneous equations in the three parameters $x_h, y_h \ and \ h$.

Expressing positions in homogeneous coordinates allows us to represent all geometric transformation equations as matrix multiplications.

Coordinates are represented with three-element column vectors, and transformation operations are written as 3 bv 3 matrices.

For Translation, we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T(t_x, t_y).P$$

Inverse of the translation matrix is obtained by replacing the translation parameters with their negatives: $-t_x \ and - t_y$

For Rotation, we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = R(\theta).P$$

We get the Inverse Rotation Matrix when θ is replaced by $-\theta$

For Scaling, we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = S(s_x, s_y).P$$

$S(s_x, s_y)$ is 3 by 3 matrix with parameters $s_x \ and \ s_y$

Inverse of the scaling matrix is obtained by replacing the $s_x \ and \ s_y$ with their Multiplicative inverses: $\frac{1}{s_x} \ and \ \frac{1}{s_y}$

IV. COMPOSITE TRANSFORMATIONS

Setting up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transformations.

Forming products of transformation matrices is often referred to as a Concatenation, or Composition, of matrices.

Note: For column-matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left.

A. Translations

Two successive translation vectors (t_{x_1}, t_{y_1}) and (t_{x_2}, t_{y_2}) are applied to a coordinate position P, the final transformed location P' is calculated as

$$P' = T(t_{x_2}, t_{y_2}).\{T(t_{x_1}, t_{y_1}).P\}$$
$$= \{T(t_{x_2}, t_{y_2}).T(t_{x_1}, t_{y_1})\}.P$$

where P and P' are represented as homogeneous-coordinate column vectors.

By multiplying the two translation matrices, we can verify below

$$T(t_{x_2}, t_{y_2}).T(t_{x_1}, t_{y_1})$$
$$= \begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & t_{x_1} + t_{x_2} \\ 0 & 1 & t_{y_1} + t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$
$$= T(t_{x_1} + t_{x_2}, t_{y_1} + t_{y_2})$$

"Two successive translations are Additive"

Final translated coordinates can be calculated with composite translated matrix as

$$P' = T(t_{x_1} + t_{x_2}, t_{y_1} + t_{y_2}).P$$

B. Rotations:

Two successive rotation vectors $R(\theta_1)$ and $R(\theta_2)$ are applied to a coordinate position P, the final transformed location P' is calculated as

$$P' = R(\theta_2).\{R(\theta_1).P\}$$
$$= \{R(\theta_2).R(\theta_1)\}.P$$

where P and P' are represented as homogeneous-coordinate column vectors.

By multiplying the two rotation matrices, we can verify below

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

"Two successive rotations are Additive"
Final rotated coordinates can be calculated with composite rotation matrix as

$$P' = R(\theta_1 + \theta_2).P$$

C. Scalings:

Two successive scaling vectors (s_{x_1}, s_{y_1}) and (s_{x_2}, s_{y_2}) are applied to a coordinate position P, the final transformed location P' is calculated as

$$P' = S(s_{x_2}, s_{y_2}).\{S(s_{x_1}, s_{y_1}).P\} = \{S(s_{x_2}, s_{y_2}).S(s_{x_1}, s_{y_1})\}.P$$

where P and P' are represented as homogeneous-coordinate column vectors.

By multiplying the two scaling matrices, we can verify below

$$S(s_{x_2}, s_{y_2}).S(s_{x_1}, s_{y_1})$$
$$= \begin{bmatrix} s_{x_2} & 0 & 0 \\ 0 & s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} s_{x_1} & 0 & 0 \\ 0 & s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} s_{x_1}.s_{x_2} & 0 & 0 \\ 0 & s_{y_1}.s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= S(s_{x_1}.s_{x_2}, s_{y_1}.s_{y_2})$$

"Two successive scalings are Multiplicative"
Final rotated coordinates can be calculated with composite rotation matrix as

$$P' = S(s_{x_1}.s_{x_2}, s_{y_1}.s_{y_2}).P$$

V. GENERAL PIVOT-POINT ROTATION

We can generate rotations about any selected pivot point (x_r, y_r) by performing the following sequence of Translate-Rotate-Translate operations:

1. Translate the object so that the pivot-point position is moved to the coordinate origin.
2. Rotate the object about the coordinate origin.
3. Translate the object so that the pivot point is returned to its original position.

This transformation sequence is illustrated in Fig.

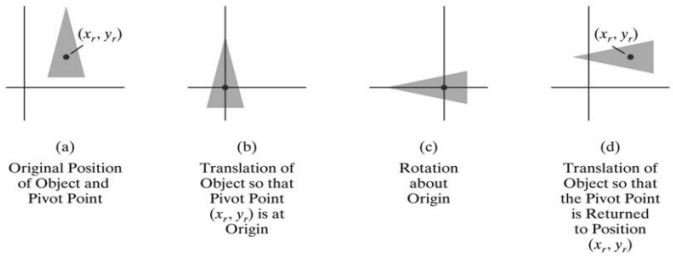


Fig.7: A transformation sequence for rotating an object about a specified pivot point using the rotation matrix $R(\theta)$ of transformation

Composite Transformation Matrix for this sequence is obtained with concatenation

$$T(x_r, y_r).R(\theta).T(-x_r, -y_r)$$
$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1 \quad \cos\theta) \quad x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$
$$= R(x_r, y_r, \theta)$$

Where $T(-x_r, -y_r) = T^{-1}(x_r, y_r)$

VI. GENERAL FIXED-POINT SCALING

We can generate Scalings about any selected fixed point (x_f, y_f) by performing following sequence of Translate-Scale-Translate operations

1. Translate object so that the fixed point coincides with the coordinate origin.
2. Scale the object with respect to the coordinate origin.
3. Use the inverse translation of step 1 to return the object to its original position.

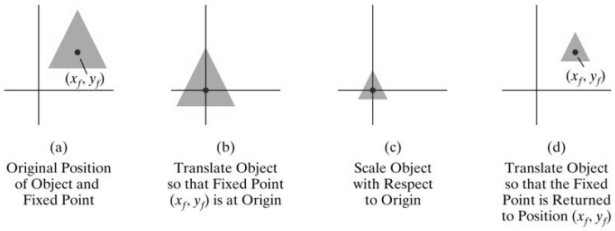


Fig.8: A transformation sequence for scaling an object with respect to a specified fixed point using the scaling matrix $S(s_x, s_y)$ of transformation

Composite Transformation Matrix for this sequence is obtained with concatenation

$$T(x_f, y_f).S(s_x, s_y).T(-x_f, -y_f)$$
$$= \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$
$$- \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$
$$= S(x_f, y_f, s_x, s_y)$$

VII. GENERAL SCALING DIRECTIONS

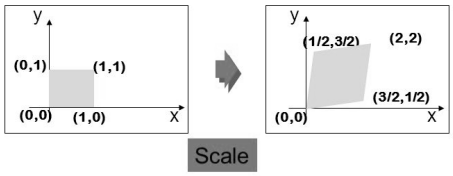
Parameters s_x *and* s_y scale objects along *x* and *y* directions

We can scale an object in other directions by rotating object to align desired scaling directions with coordinate axes before applying scaling transformations

Suppose we want to applying scaling factors with values specified by parameters s_1 and s_2 in directions shown

To accomplish the Scaling without changing the orientation

1.Perform a rotation so that directions for s_1 and s_2 coincide with *x* and *y* axes
2.Then the scaling transformation is applied



3.Opposite rotation to return points to their original orientation
Composite Matrix resulting from product of these three transformations is

$$R^{-1}(\theta).S(s_1, s_2).R(\theta)$$
$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} s_1 cos^2\theta + s_2 sin^2\theta & (s_2 - s_1)\cos\theta \sin\theta & 0 \\ (s_2 - s_1)\cos\theta \sin\theta & s_1 sin^2\theta + s_2 cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

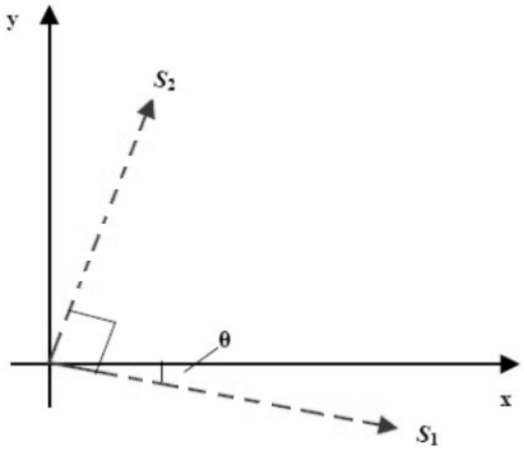


Fig: Scaling parameters s1 and s2 are to be applied in orthogonal directions defined by angular displacement θ

Ex: From General Scaling directions, Turn a unit square into parallelogram by stretching it along diagonal from $(0,0)\ and\ (1,1)$

Sol:

Rotate the diagonal onto Y-axis and double its length with transformation parameters

s1=1 and s2=2

Angular displacement $\theta = 45$

$$R^{-1}(\theta).S(s_1,s_2).R(\theta)$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\vartheta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.cos^2 45 + 2.sin^2 45 & (2-1)\cos 45 \sin 45 & 0 \\ (2-1)\cos 45 \sin 45 & 1.sin^2 45 + 2.cos^2 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^1/_2 + 2.{}^1/_2 & (2-1).{}^1/_{\sqrt{2}}.{}^1/_{\sqrt{2}} & 0 \\ (2-1)\,{}^1/_{\sqrt{2}}.{}^1/_{\sqrt{2}} & 1.{}^1/_2 + 2.{}^1/_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^3/_2 & {}^1/_2 & 0 \\ {}^1/_2 & {}^3/_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

VIII. CONCATENATION PROPERTIES

Matrix multiplication is associative.

For any three matrices, A, B, and C, the matrix product $A\,.B\,.C$ can be performed by first multiplying A and B or by first multiplying B and C:

$$A.B.C = (A.B).C = A.(B.C)$$

Therefore, we can evaluate matrix products using either a Left-To-Right or a Right-To-Left associative grouping.

On the other hand, transformation products may not be commutative: The matrix product A. B is not equal to B - A, in general. This means that if we want to translate and rotate an object, we must be careful about the order in which the composite matrix is evaluated

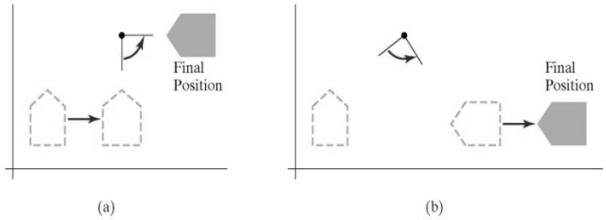


Fig.9: Reversing the order in which a sequence of transformation is performed may affect transformed position of an object

(a) An object is first translated, then rotated
(b) Object is rotated first, then translated

For some special cases, such as a sequence of transformations all of the same kind, the multiplication of transformation matrices is commutative.

Ex: Two successive rotations could be performed in either order and the final position would be the same.

This commutative property holds also for two successive translations or two successive scalings.
Another commutative pair of operations is rotation and uniform scaling $s_x = s_y$

IX. GENERAL COMPOSITE TRANSFORMATIONS AND COMPUTATIONAL EFFICIENCY

General two-dimensional transformation, representing a combination of translations, rotations, and scalings, can be expressed as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The four elements rs_{ij} are the multiplicative Rotation-Scaling terms in the transformation that involve only rotation angles and scaling factors.

Elements trs_x and trs_y are the translational terms containing combinations of translation distances,

pivot-point and fixed-point coordinates, and rotation angles and scaling parameters.

Ex; If an object is to be scaled and rotated about its centroid coordinates (x_c, y_c) and then translated, the values for the elements of the composite transformation matrix are

$$T(t_x, t_y).R(x_c, y_c, \theta).S(x_c, y_c, s_x, s_y)$$
$$= \begin{bmatrix} s_x \cos\theta & -s_y \sin\theta & x_c(1 - s_x \cos\theta) + y_c s_y \sin\theta + t_x \\ s_x \sin\theta & s_y \cos\theta & y_c(1 - s_y \cos\theta) - x_c s_x \sin\theta + t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Although matrix equation requires nine multiplications and six additions,

Explicit calculations for the transformed coordinates are

$$x' = x.rs_{xx} + y.rs_{xy} + trs_x$$
$$y' = x.rs_{yx} + y.rs_{yy} + trs_y$$

Thus, we actually only need to perform four multiplications and four additions to transform coordinate positions.

This is the maximum number of computations required for any transformation sequence, once the individual matrices have been concatenated and the elements of the composite matrix evaluated.

An efficient implementation for the transformation operations, therefore is to formulate transformationmatrices, concatenate any transformation sequence, and calculate transformed coordinates

Rigid-body Transformation Matrix:

Involving only translations and rotations
Expressed as

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the four elements r_{ij} are the multiplicative rotation terms, and elements tr_x and tr_y are the translational terms.

Rigid-Motion transformation: Rigid-body change in coordinate

All angles and distances between coordinate positions are unchanged by the transformation.

Above matrix has the property that its upper-left 2-by-2 submatrix is an orthogonal matrix. This means that if we consider each row of the submatrix as a vector, then the two vectors (r_{xx}, r_{xy}) and (r_{yx}, r_{yy}) form an orthogonal set of unit vectors:

Each vector has unit length

$$r_{xx}^2 + r_{xy}^2 = r_{yx}^2 + r_{yy}^2 = 1$$

and the vectors are perpendicular (their dot product is *0):*

$$r_{xx}.r_{yx} + r_{xy}.r_{yy} = 0$$

Therefore, if these unit vectors are transformed by the rotation submatrix, (r_{xx}, r_{xy}) is converted to a unit vector along the *x* axis and (r_{yx}, r_{yy}) is transformed into a Composite Transformations

unit vector along they axis of the coordinate system:

$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Ex: Following rigid-body transformation

a.Rotates an object through an angle θ about a pivot point (x_r, y_r) and b.then translates:

Sol

$$T(t_x, t_y).R(x_r, y_r, \theta)$$
$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1 - \cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1 \quad \cos\theta) \quad x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1 - \cos\theta) + y_r \sin\theta + t_x \\ \sin\theta & \cos\theta & y_r(1 - \cos\theta) - x_r \sin\theta + t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Here, orthogonal unit vectors in the upper-left 2-by-2 submatrix are $(\cos\theta, -\sin\theta)$ and $(\sin\theta, \cos\theta)$

Unit Vector along X-axis

$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

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

Unit Vector along Y-axis

$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

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

Orthogonal property of rotation matrices is useful for constructing a rotation matrix when we know the final orientation of an object rather than the amount of angular rotation necessary to put the object into that position.

Directions for the desired orientation of an object could be determined by the alignment of certain object in a scene or by selected positions in the scene.

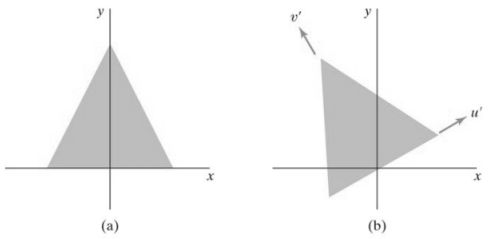


Fig.10: Rotation matrix for revolving an object from position (a) to position (b) can be constructed with the values of the unit orientation vectors

X. OTHER TRANSFORMATIONS

Some packages provide a few additional transformations that are useful in certain applications.

Two such transformations are Reflection and Shear.

A.Reflection:

A Reflection is a transformation that produces a mirror image of an object relative to an axis of the reflection

We can choose an axis of reflection in the *xy* plane or perpendicular to the *xy* plane.

Examples of common reflections:

1.Reflection about line $y = 0$, $x - axis$

Transformation Matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

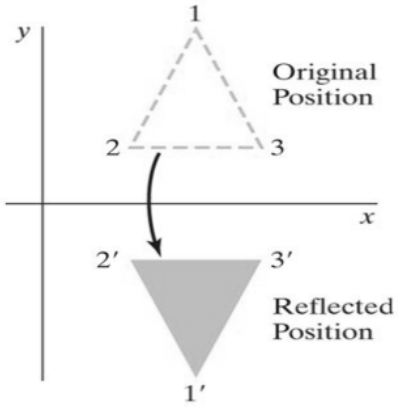
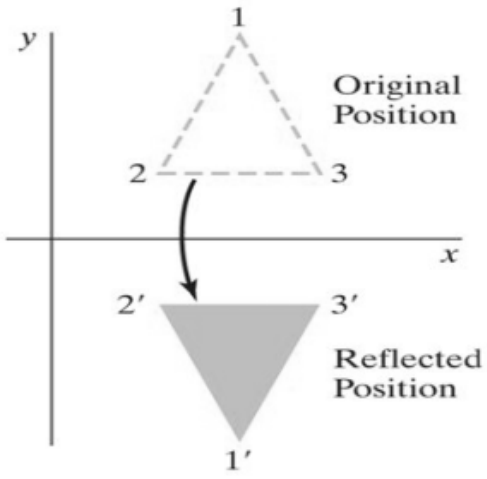


Fig.11

This transformation keeps *x* values same but flips *y* values of coordinates positions



2. Reflection about line $x = 0$, $y - axis$

Transformation Matrix $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

This transformation keeps *y* values same but flips *x* values of coordinates positions

3. Reflection about origin $x = 0$ $y = 0$

Transformation Matrix $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

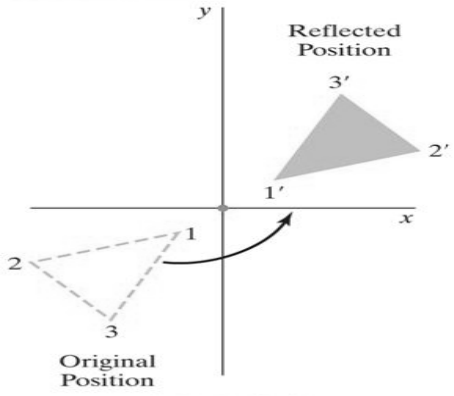


Fig.13

This transformation flips both *x* and *y* coordinates of a point by reflecting relative to an axis that is perpendicular to *xy* plane and that passes through coordinate origin

Rotation Matrix $R(\theta)$ with $\theta = 180^0$

Half a revolution about origin

4.Reflection axis as diagonal line $y = x$

Transformation Matrix $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

This transformation can be derived by concatenating a sequence of rotation and coordinate-axis reflection
Possible sequence is as follows:

a.Perform clockwise rotation through an 45^0 angle, which rotates line $y = x$ onto $x - axis$

b.Next we perform a reflection with respect to $x - axis$

c.

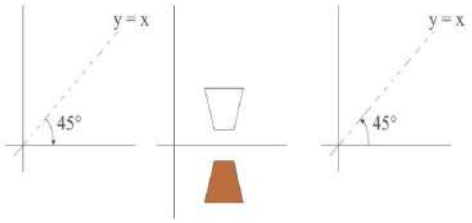
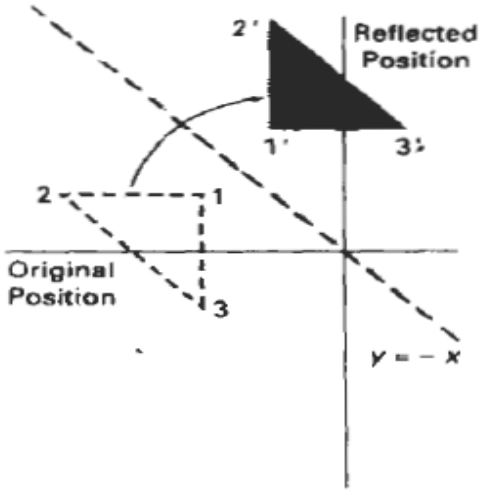


Fig.14:Sequence of transformations to produce reflection about line $y = x$



a. Clockwise rotation of 45^0

b, Reflection about $x - axis$

c. Counter Clockwise rotation of 45^0

5. Reflection about diagonal line $y = -x$

a. Clockwise rotation of 45^0

b, Reflection about $y - axis$ Fig.15

c. Counter Clockwise rotation of 45^0

Transformation Matrix $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Note: Reflection about line $y = mx + b$ in *xy* plane can be accomplished with a combination of Translate-Rotate-Reflect Transformations

a.Translate line so that it passes through origin
b.Rotate line onto one of coordinate axes
c.Reflect about that axis

Finally we restore line to its original position with inverse rotation and translation transformations

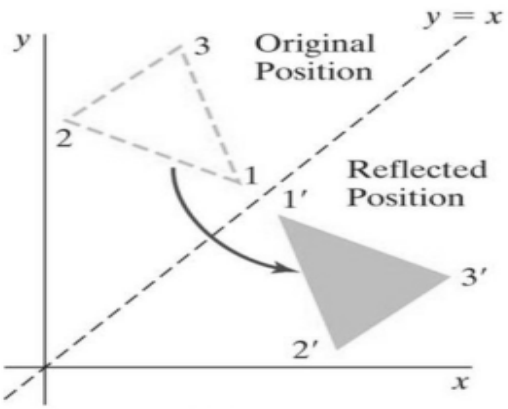


Fig.16

B.Shear

A transformation that distorts the shape of an object
Such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a *Shear*.

Two common shearing transformations:

a.Those that shift coordinate *x* values and
b.Those that shift coordinate *y* values.

a.An *x*-direction shear relative to the *x* axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which transforms coordinate positions as

$$x' = x + sh_x . y$$
$$y' = y$$

Any real number can be assigned to the shear parameter sh_x.
A coordinate position (x, y) is then shifted horizontally by an amount proportional to its distance *(y value)* from the *x axis* $(y = 0)$.

Ex: Setting sh_x to 2

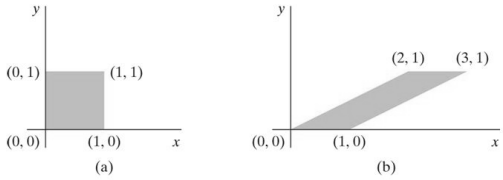


Fig.17: A Unit Square

a) is converted to a parallelogram

b) Using *x*-direction Shear Matrix with with $sh_x = 2$

We can generate *x*-direction shears relative to other reference lines with transformation matrix

$$\begin{bmatrix} 1 & sh_x & -sh_x . y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which transforms coordinate positions as

$$x' = x + sh_x . (y - y_{ref})$$
$$y' = y$$

Ex: For a shear parameter value of $1/2$ relative to line $y_{ref} = -1$

Shearing transformation is given by

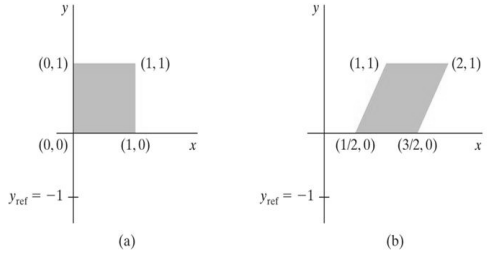


Fig.18: A Unit Square

a) is transformed to a shifted parallelogram

b) with $sh_x = 1/2$ and $y_{ref} = -1$ in shear matrix

b.A *y*-direction shear relative to the line *y* axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which transforms coordinate positions as
$x' = x$

$$y' = y + sh_y . x$$

Any real number can be assigned to the shear parameter sh_y. We can generate *y*-direction shears relative to other reference lines with transformation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y . x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

which transforms coordinate positions as

$$x' = x$$

$$y' = sh_y . (x - x_{ref}) + y$$

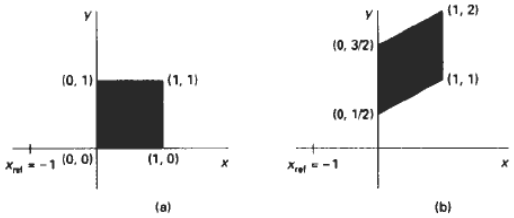


Fig.19: A Unit Square

a) is turned into a shifted parallelogram

b) with parameter values $sh_y = 1/2$ and $x_{ref} = -1$ in *y*-direction using shearing transformation

3-Dimensional Geometric Transformations

Methods for Geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the *z* coordinate

A.Translation

In a three-dimensional homogeneous coordinate representation, a point is translated from position $P = (x, y, z)$ to position $P' = (x', y', z')$ with the matrix operation

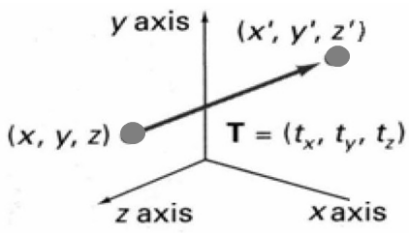


Fig.20: Translating a point with Translation Vector $T = (t_x, t_y, t_z)$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = T \cdot P$$

Parameters t_x ,t_y and t_z specifying translation distances for the coordinate directions *x,* y, and z, are these are assigned any real values.

Matrix representation

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

An object is translated in three dimensions by transforming each of the defining points of the object.

Note: Inverse of the translation matrix can be achieved by negating the translation distances t_x ,t_y and t_z.

This produces a translation in the opposite direction, and the product of a translation matrix and its inverse produces the identity matrix.

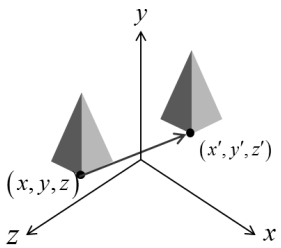


Fig.21*: Translating an object with translation vector T*

B.Rotation

To generate a rotation transformation for an object, we must designate an axis of rotation (about which the object is to be rotated) and the amount of angular rotation.

Also, we can use combinations of coordinate axis rotations (along with appropriate translations) to specify any general rotation.

Positive rotation angles produce counterclockwise rotations about a coordinate axis, if we are looking along the positive half of the axis toward the coordinate origin.

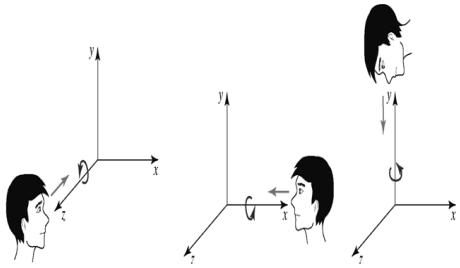


Fig.22:Positive rotations Directions about coordinate axes are counter-clockwise, when looking towards the origin from positive coordinate position on each axis

XI. COORDINATE-AXES ROTATIONS

1. Z-Axis Rotation equations in three dimensions

$$x' = xCos\theta - ySin\theta$$
$$y' = xSin\theta + yCos\theta$$
$$z' = z$$

Parameter θ specifies Rotation Angle

In Homogeneous coordinate form, the three-dimensional Z-Axis rotation equations

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Cos\theta & -Sin\theta & 0 & 0 \\ Sin\theta & Cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_z(\theta).P$$

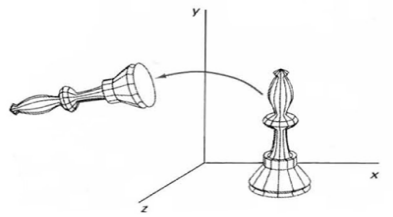


Fig.23: Rotation of an object about z axis

Note: Transformation equations for rotations about other two coordinate axes can be obtained with cyclic permutation of coordinate parameters *x,y* and *z*

$$x \rightarrow y \rightarrow z \rightarrow x$$

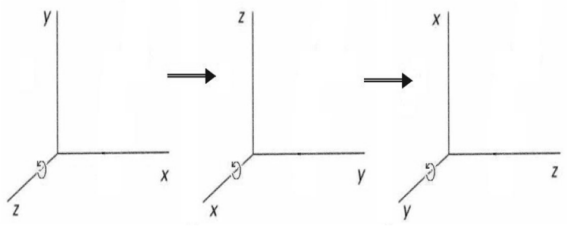


Fig.24*: Cyclic permutation of the Cartesian-Coordinate axes to produce the three sets of coordinate-axis rotation equations*

2.X- Axis Rotation equations in three dimensions

$$y' = yCos\theta - zSin\theta$$
$$z' = xSos\theta + zCin\theta$$
$$x' = x$$

Parameter θ specifies Rotation Angle

In Homogeneous coordinate form, the three-dimensional Z-Axis rotation equations

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\theta & -Sin\theta & 0 \\ 0 & Sin\theta & Cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_x(\theta).P$$

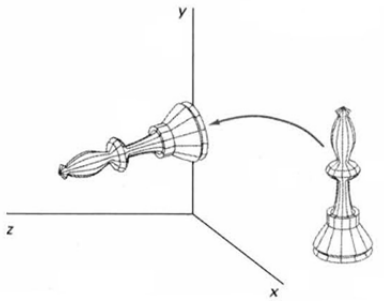


Fig.25*: Rotation of an object about x axis*

3.Y- Axis Rotation equations in three dimensions

$$z' = zCos\theta - xSin\theta$$
$$x' = zSin\theta + xCos\theta$$
$$y' = y$$

Parameter θ specifies Rotation Angle

In Homogeneous coordinate form, the three-dimensional Z-Axis rotation equations

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Cos\theta & 0 & Sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -Sin\theta & 0 & Cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_y(\theta).P$$

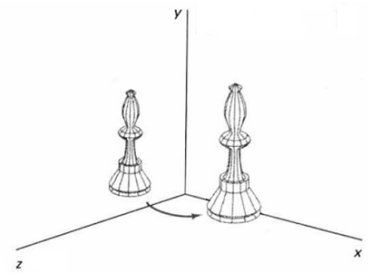


Fig.26*: Rotation of an object about y axis*

Note: Inverse Rotation Matrix is formed by replacing θ with $-\theta$

Negative Values for Rotation Angles generate rotation in clock-wise direction

Identity Matrix is obtained when Rotation Matrix is multiplied by its inverse

Inverse of a Rotation Matrix R by calculating its transpose $R^{-1} = R^T$

XII. GENERAL THREE-DIMENSIONAL ROTATIONS

A rotation matrix for any axis that does not coincide with a coordinate axis can be set up as a composite transformation involving combinations of translations and the coordinate-axes

Case 1:

An object is to be rotated about an axis that is parallel to one of the coordinate axes

Desired rotation can be made with following transformation sequence

1.Translate the object so that the rotation axis coincides with the parallel coordinate axis
2. Perform the specified rotation angle about that axis.
3.Translate the object so that the rotation axis is moved back to its original position

Steps in this sequence are illustrated in Fig.

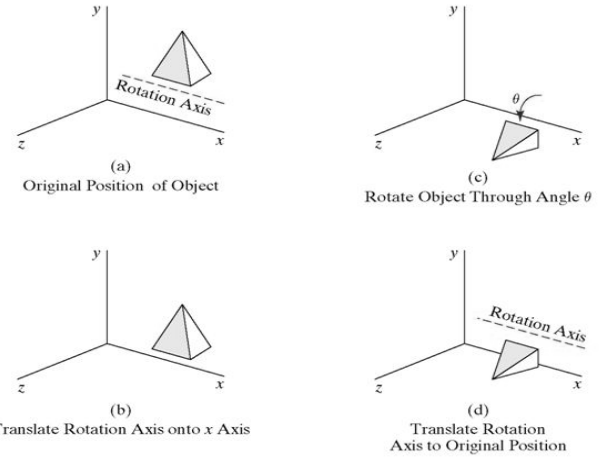


Fig.27*: Sequence of Transformations for rotating an object about an axis that is parallel to X-axis*

Any coordinate position P on the object in this figure is transformed with the sequence shown as

$$P' = T^{-1}.R_x(\theta).T.P$$

Composite Matrix for the transformation is

$$R(\theta) = T^{-1}.R_x(\theta).T$$

Case 2:

When an object is to be rotated about an axis that is not parallel to one of the coordinate axes, we need to perform some additional transformations.

In this case, we also need rotations lo align the axis with a selected coordinate axis and to bring the axis back to its original orientation.

Five Steps:

1 Translate the object so that the rotation axis passes through the coordinate origin.
2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes.
3. Perform the specified rotation about that coordinate axis.
4. Apply inverse rotations to bring the rotation axis back to its original orientation.
5. Apply the inverse translation to bring the rotation axis back to its original position.

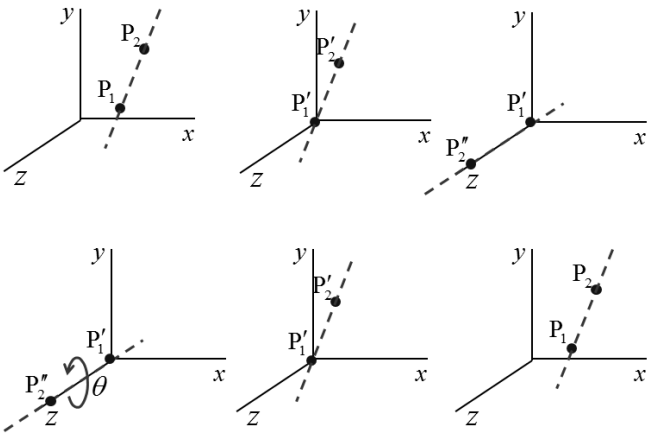


Fig.28:*Five transformation steps for obtaining a composite matrix for rotation about an arbitrary axis, with the rotation axis projected onto the z axis.*

A rotation axis can be defined with two coordinate positions with one coordinate point and direction angles (or direction cosines) between the rotation axis and two of the coordinate axes.

We will assume that the rotation axis is defined by two points, as illustrated, and that the direction of rotation is to be counterclockwise when looking along the axis from P_2 to P_1

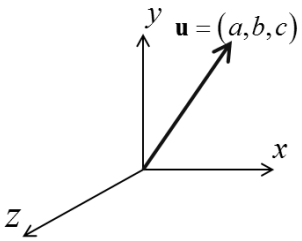


Fig.29: An axis of Rotation

An axis vector is then defined by the two points as

$$V = P_2 - P_1$$

A unit vector u is then defined along the rotation axis as

$$u = \frac{V}{|v|} = (a, b, c)$$

where the components *a*, *b* and *c* of unit vector *u* are the direction cosines for the rotation axis:

$$a = \frac{x_2 - x_1}{|v|}$$

$$b = \frac{y_2 - y_1}{|v|}$$

$$c = \frac{z_2 - z_1}{|v|}$$

If the rotation is to be in the opposite direction (clockwise when viewing from P_2 to P_1), then we would reverse axis vector V and unit vector u so that they point from P_2 to P_1

The first step in the transformation sequence for the desired rotation is to set up the translation matrix that repositions the rotation axis so that it passes through the coordinate origin.

For the desired direction of rotation, we accomplish this by moving P_1 to origin(If the rotation direction had been specified in the opposite direction, we would move P_2, to origin)

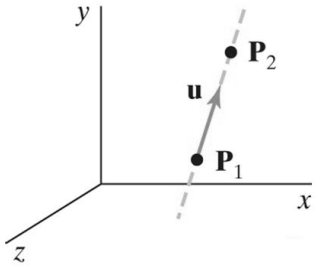


Fig.30: An axis of rotation(dashed line) with points P_1 and P_2. The direction for the unit axis vector u is determined by the specified rotation direction

This translation matrix is

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which repositions the rotation axis and the object as shown

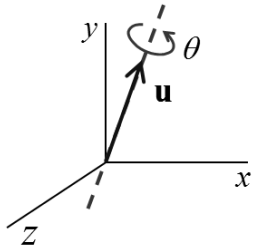


Fig.31: Translation of the rotation axis to the coordinate origin

Now we need the transformations that will put the rotation axis on the *z* axis

We can use the coordinate axis rotations to accomplish this alignment in two steps

There a number of ways to perform two steps

1.Rotate about the *x* axis to transform vector *u* into the *xy* plane
2.Then we swing *u* around to the *z* axis using a *y*-axis rotation
These two rotations are illustrated in fig. for one possible orientation of vector *u*

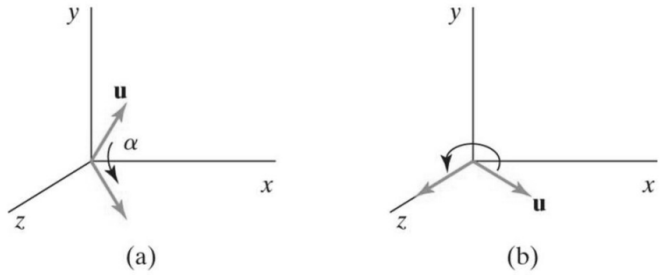


Fig.32: Unit vector u is rotated about the x axis to bring it into the xz plane

a) Then it is rotated around the *y* axis to align it with *z* axis
b) We establish the transformation matrix for rotation around the *x* axis by determining the values for the sine and cosine of the rotation angle necessary to get *u* into the *xz* plane

This rotation angle is angle between the projection of *u* in the *yz* plane and the positive *z*-axis

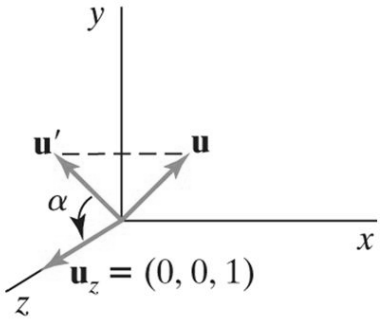


Fig.33: Rotation of u around the x axis into xz plane is accomplished by rotating u (the projection of u in the yz plane) through angle α onto z axis

If we assign the projection of *u* in the *yz* plane as the vector $u' = (0, b, c)$ then the cosine of rotation angle α can be determined from dot product of u' and the unit vector u_z along *z* axis

$$\cos \propto = \frac{u'.u_z}{|u'|.|u_z|} = \frac{c}{d}$$

Where *d* is the magnitude *of* u'

$$d = \sqrt{b^2 + c^2}$$

Similarly, we can determine the sine of $\propto$ from the cross product of u' and u_z

Coordinate-independent form of this cross product is

$$u' \times u_z = u_x|u'|.|u_z| \sin \propto$$

And the Cartesian form for cross product gives us

$$u' \times u_z = u_x.b$$

Equating above two equations and noting that $|u_z| = 1$ and $|u'| = d$
We have

$$d \sin \propto = b$$

$$\sin \alpha = \frac{b}{d}$$

We obtained the values for $\cos \propto$ and $\sin \alpha$ in terms of vector *u*, we can set up matrix for rotation of *u* about *x* axis

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix rotates unit vector *u* about the *x* axis into the *xz* plane counterclockwise around the *y* axis onto the positive *z* axis

Orientation of the Unit vector in the *xz* plane(After Rotation about the *x* axis) is shown in Fig

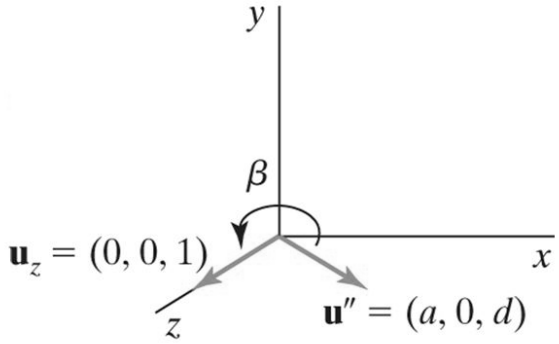


Fig.34: Rotation of unit vector u" (Vector u after rotation into xz plane)about the y axis. Positive rotation angle β aligns u" with vector u_z

This vector labeled u'' has the value *a* for its *x* component unchanged

Its *z* component is *d* (Magnitude of u'), because vector u' has been rotated onto *z* axis

And *y* component of u'' is zero, because it now lies in *xz* plane

We can determine cosine of rotation angle β from expressions for dot product of unit vector u'' and u_z

$$\cos\beta = \frac{u''.u_z}{|u''|.|u_z|} = d$$

Since $|u_z| = |u''| = 1$

Where *d* is the magnitude *of* u'

$$d = \sqrt{b^2 + c^2}$$

Comparing the coordinate-independent form of cross product

$$u'' \times u_z = u_y |u''| \, |u_z| \sin\beta$$

Where the Cartesian form

$$u'' \times u_z = u_y.(-a)$$

We find that

$$\sin\beta = -a$$

Thus, the transformation matrix for rotation of u'' about the *y* axis is

$$\mathbf{R}_y(\beta) = \begin{bmatrix} d & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Specified rotation angle θ can now be applied as a rotation about *z* axis

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix for rotation about an arbitrary axis can be expressed as composition of above seven individual transformations

$$R(\theta) = T^{-1}.R_x^{-1}(\alpha).R_y^{-1}(\beta).R_z(\theta).R_y(\beta).R_x(\alpha).T$$

Scaling

The matrix expression tor the scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as

$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{S} \cdot \mathbf{P}$$

where scaling parameters S_x, S_y and S_z are assigned any positive values.

Explicit expressions for the coordinate transformations for scaling relative to the origin are

$$x' = x.S_x$$
$$y' = y.S_y$$
$$z' = z.S_z$$

Scaling an object with transformation changes the size of the object and repositions the object relative to the coordinate origin.

Also, if the transformation parameters are not all equal, relative dimensions in the object are changed

Note: We preserve the original shape of an object with a uniform scaling

$$S_x = S_y = S_z$$

The result of scaling an object uniformly with each scaling parameter set to 2 is shown in Fig.

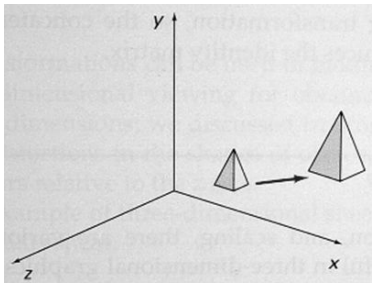


Fig.35: Doubling the size of an object with transformation also moves the object farther from origin

General Fixed-Point Scaling

Scaling with respect to a selected fixed position (x, y, z) can be represented with the following transformation sequence:

1. Translate the fixed point to the origin.
2. Scale the object relative to the coordinate origin
3. Translate the fixed point back to its original position.

This sequence of transformations is demonstrated in Fig.

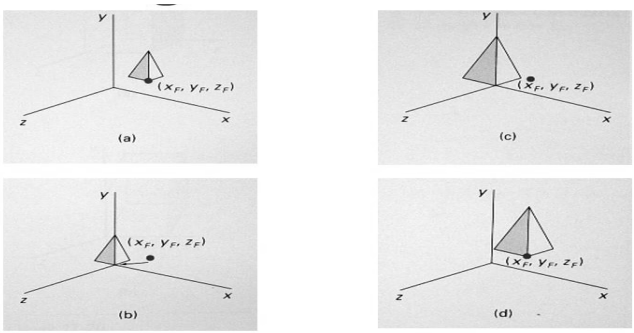


Fig.36: Scaling an object is relative to a selected fixed point is equivalent to the sequence of transformations shown

Matrix representation for an arbitrary fixed-point scaling can then be expressed as the concatenation of these *Translate-Scale-Translate* transformations as

$$T(x_f, y_f, z_f).S(s_x, s_y, s_z).T(-x_f, -y_f, -z_f)$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

We form the inverse scaling matrix by re- equivalent to the sequence of lacing the scaling parameters S_x, S_y and S_z with their reciprocals.

Inverse matrix generates an opposite scaling transformation, so the concatenation of any

Three-Dimensional Geometric scaling matrix and its inverse produces the identity matrix.

XIII. OTHER TRANSFORMATIONS IN THREE-DIMENSIONAL TRANSFORMATIONS

Various additional transformations that are often useful in three-dimensional graphics applications.

Two of these are

- Reflection And
- Shear.

A.Reflections

A three-dimensional reflection can be performed relative to a selected *reflection axis* or with respect to a selected *reflection plane.*

Reflections relative to a given axis are equivalent to 180^0 rotations about that axis.

Reflections with respect to a plane are equivalent to 180^0 rotations in four-dimensional space.

When the reflection plane is a coordinate plane (either xy, *xz, or* yz), we can make the transformation as a conversion between Left-handed and right-handed systems.

An example of a reflection that converts coordinate specifications from a right-handed system to a left-handed system (or vice versa) is shown in Fig.

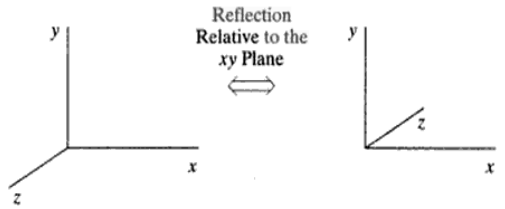


Fig.37*:* Conversion of coordinate specifications from a right-handed to a left-handed system can be carried out with the reflection transformation

This transformation changes the sign of the *z* coordinates, leaving the *x* and *y*-coordinate values unchanged.

Matrix representation for this reflection of points relative to the *xy* plane is

$$\begin{bmatrix} x \\ y \\ -z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformation matrices for inverting *x* and *y* values are defined similarly, as reflections relative to the *yz* plane and *xz* plane, respectively

B.Shears

Shearing transformations can he used to modify object shapes.

They are also useful in three-dimensional viewing for obtaining general projection transformations

.

In three dimensions, we can also generate shears relative to the *z* axis.

Example of three-dimensional shearing, the following transformation produces a z-axis shear:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Parameters *a* and *b* can be assigned any real values.

The effect of this transformation matrix is to alter *x*- and *y*-coordinate values by an amount that is proportional to the 2 value, while leaving the *z* coordinate unchanged.

Boundaries of planes that are perpendicular to the *z* axis are thus shifted by an amount proportional to *z.*

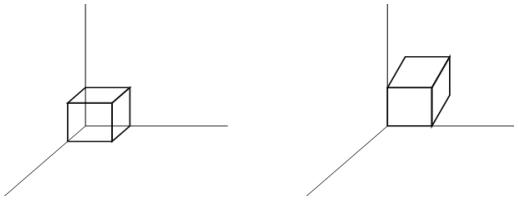


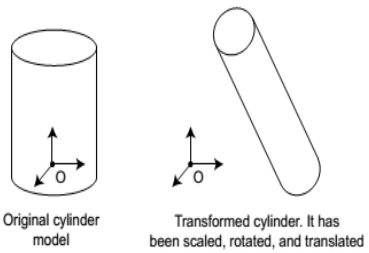
Fig.38: A Unit Cube a) is sheared

b) by transformation matrix $a = b = 1$

XIV.APPLICATIONS OF GEOMETRIC TRANSFORMATIONS

One could imagine a computer graphics system that requires the user to construct everything directly into a single scene. But, one can also immediately see that this would be an extremely limiting approach. In the real world, things come from various places and are arranged together to create a scene. Further, many of these things are themselves collections of smaller parts that are assembled together. We may wish to define one object relative to another – for example we may want to place a hand at the end of an arm. Also, it is often the case that parts of an object are similar, like the tires on a car. And, even things that are built on scene, like a house for example, are designed elsewhere, at a scale that is usually many times smaller than the house as it is built. Even more to the point, we will often want to animate the objects in a scene, requiring the ability to move them around relative to each other. For animation we will want to be able to move not only the objects, but also the camera, as we render a sequence of images as time advances to create an illusion of motion. We need good mechanisms within a computer graphics system to provide the flexibility implied by all of the issues raised above.

The figure below shows an example of what we mean.



On the left, a cylinder has been built in a convenient place, and to a convenient size. Because of the requirements of a scene, it is first scaled to be longer and thinner than its original design, rotated to a desired orientation in space, and then moved to a desired position (i.e. translated). The set of operations providing for all such transformations, are known as the affine transforms. The affines include translations and all linear transformations, like scale, rotate, and shear. Original cylinder model Transformed cylinder. It has been scaled, rotated, and translated

CONCLUSION AND FUTURE WORK

In this paper, we have reviewed the 2D transformations in computer graphics and how these transformations are done. We have discussed the difference between these transformations and how they are applied. These transformations are the basis of computer graphics and also used in 3D graphics, modeling, and animation. We have discussed the methods and techniques of implementing these transformations in an efficient way. In future, other transformations which are not much used will be discussed. There is always a scope of finding a better method of doing things and we will try to find a more efficient and faster method to solve these transformations with the use of least resources than the existing ones.

ACKNOWLEDGMENTS

Dr.S.V.B.Subrahmanyeswara Rao has 16 years of teaching experience. He is presently working in Rama Chandra College of Engineering, Eluru in the Department of Mathematics.

His areas of interest are Commutative Algebra and Cryptography.The author like to thank the Management of RCE for the support.

Mrs.Y.Anjani has 10 years of teaching experience. She is presently working in V.K.R,V.N.B & A.G.K College of Engineering, Gudivada in the Department of Computer Science & Engineering. Her areas of interest are Cryptography and Network Security, Mobile Computing. The author like to thank the Management of VKR,VNB& AGK College of Engineering for the support.

REFERENCES

[1] Durand Cutler, 'Transformations', Massachusetts Institute of Technology, September 2008.
[2] John Pile Jr., '2D Graphics Programming for Games', New York, NY: CRC Press. ISBN 1466501898, May 2013.
[3] John E. Howland, 'Representing 2D Transformations as Matrices', Department of Computer Science, Trinity University, 2001.
[4] Wayne Carlson, 'A Critical History of Computer Graphics and Animation, Ohio State University, 2003.
[5] John Peddie, 'The History of Visual Magic in Computers', 'How Beautiful Images Are Made in CAD, 3D, VR, and AR,Springer, ISBN 978-1447149316, pp. 101, 2013.
[6] Marc Levoy, '2D transformations', Introduction to Computer Graphics, Stanford University, 1996.
[7] B. Stenseth, '2D transformations', Department of Information Technology, Ostfold University College, 1998.
[8] David Blythe, 'Advanced Graphics Programming Techniques Using OpenGL', Siggraph 1999.
[9] David Rogers, 'Procedural Elements for Computer Graphics', McGraw Hill, ISBN-13: 978-0070535480, 1998.
[10] Donald Hearn and M. Pauline Baker, 'Computer Graphics', Prentice-Hall, ISBN 978-81-775-8765-4, 1994.
[11] Anon., 'What is Computer Graphics?', Cornell University Program of Computer Graphics, April 1998.