NEAR EAST UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

Mr. Mehrdad Khaledi

Meltem İzgi
91397

MATRIX ALGEBRA AND ITS APPLICATION IN TWO-DIMENSIONAL TRANSFORMATIONS....... 1

Transformations Commands....... 12
Raster Method for transformations....... 13

MATRIX REPRESENTATION AND THREE-DIMENSIONAL GRAPHICS....... 14

Representing Graphic Object in Three Dimensional....... 15
Matrix Representation of Translation and Scaling....... 25
Rewiew of Vector Operations....... 28

ROTATIONS....... 31

Reflection and Sheares....... 36
Transformation of Coordinate System....... 37

MATRIX ALGEBRA AND IT'S APPLICATION IN TWO DIMENSIONAL TRANSFORMATION

The transformations of two-dimensional objects were introduced in section 3.5 When an object is described in terms of coordinates relative to hotspot,rotations about that hotspot are actually rotations about the origin.It is more difficult to compute the coordinates of a point rotated about an arbitrary point.This sections explains how complex transformations can be decompased in to a product of matrices, each representing a simpler transformation.
While matrices provide a way to investogate transformations,programs written to describe these transformations usually condense the result in code similar to that given in the last section.You will briefly review matrix multiplication before applying it.

The size of a matrix with m rows and n column is given as m*n.The matrix A,given by

$$A = \begin{bmatrix} 1\,2\,3 \\ 4\,5\,6 \end{bmatrix}$$

has 2 rows and 3 column and is therefore a 2*3 matrix,while the matrix B, given by

$$B = \begin{bmatrix} 1\,2 \\ 3\,4 \\ 5\,6 \end{bmatrix}$$

is a 3*2 matrix.A point with coordinates (x,y) may be represented as the 1*2 matrix [x,y].Two matrices are equal if and only if they are the same size and all the corresponding entires are equal.

The matrix C,given by

$$C = \begin{bmatrix} 1\,4 \\ 2\,5 \\ 3\,6 \end{bmatrix}$$

is obtained from the matrix.A described above by converting the i th row of A in to i th column of C.The matrix C is called the transpose of A,written A'.

The product XY of two matrices X and Y can only be computed if the number of columns of the first matrix.X,is the same as the number of rows of the second matrix,Y.The resulting matrix XY has the same number of rows as X and the same number of column as Y.The product of the two matrices of given above,AB,is a 2 * 2 matrix,while the product BA of the same matrices is a 3 * 3 matrix.Clearly AB=BA in this case.Even if the product XY of two matrices is defined,the product YX need not be defined because the number of columns of the first matrix must equal the number of rows of the second matrix.

To calculate the entry in the İth row and jth column of the product matrix,multioly corresponding entires in the ith row of the first matrix and the jth column of the second matrix and add the products.The product AB of mtrices A=[aij] and B=[bij] can be described formally as AB =[cij] where

$$C = \sum_{k=1}^{n} (ai,k\ bk,j)$$

FIGURE 3.6.1

Computing the entry in the first row,second column of the product of two matrices

Columns: 1 2 3, Rows: 1 2

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} ? & \boxed{?} \\ ? & ? \end{bmatrix}$$

The entry in the first row and second column of the product,AB,given in fig.3.6.1, is

$$(1*2)+(2*4)+(3*6) =2 + 8 + 18 = 28$$

The entry in the second row and second column of the product BA computed by multiplying corresponding members of the highlighted row from B and the highlighted column from A (figure 3.6.2).The value of the entry

$$(3*2) + (4*5) =26$$

One of the s,mplest applications of matrix multiplications is scaling a graph object.Figure 3.6.3 shows a rectangle with one of its corners at the origin.Corner has its coordinates changed to x'=xSx and y'=ySy, where Sx and Sy are the scaling factors.This set of equations can be given as the single matrix equation.

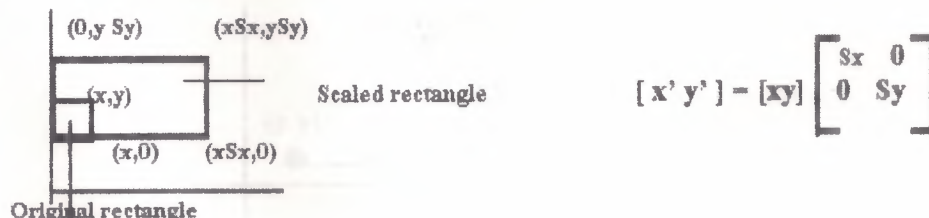
FIGURE 3.6.2

Compute the entry in the second row ,second column of a matrix.

$$\begin{bmatrix} 1 & 2 \\ \boxed{3 \quad 4} \\ 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & \boxed{2} & 3 \\ 4 & \boxed{5} & 6 \end{bmatrix} \quad \begin{bmatrix} ? & ? & ? \\ ? & \boxed{?} & ? \\ ? & ? & ? \end{bmatrix}$$

FIGURE 3.6.3

Rectangle with one corner at origin



Similar matrix equations can be used to represent reflections; for example, the reflection through the origin, x' = -x and y' = -y (figure 3.6.4), and the origin x' =-x and y' =-y figure(3.6.4) and the reflection through the line y =c,x' =y and y' =x.

Recallhat if a point (x,y) is rotated through an angle θ about the origin rectangle (figure 3.6.5), then the coordinates of its new position, (x',y'), can be calculated by the following equations:

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

Application of matrix multiplication shows that this transformation can also be repsented by the following matrix equation:

$$[x'y'] = [xy] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

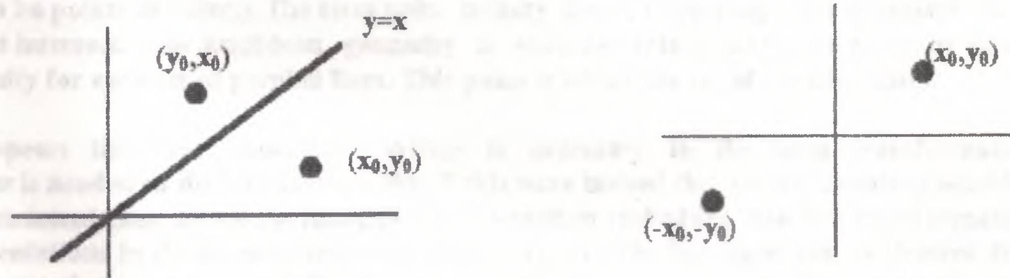
Here the matrices [x'y'] and [xy] represent the points (x',y') and (x,y) respectively, and the matrix

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\text{sm}\,\theta & \cos\theta \end{bmatrix}$$

is the representation of the rotation.

FIGURE 3.6.4

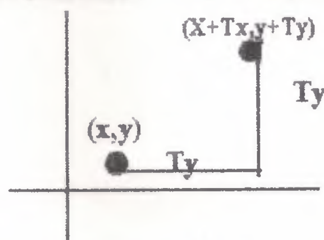
Reflection through a line and reflection through a point



Reflection across the line y=x

FIGURE 3.6.6

The translation of the point (x,y) to the point (x+Tx,y+Ty)



Another basic transformation is translation (figure 3.6.6). The equation this transformation are

$$x' = x + Tx$$
$$y' = y + Ty$$

These equations cannot be converted to a useful matrix representation within usual coordinate system.To facilitate the uniform representation of transformations as matrix equations,the standard coordinate system is extended system known as homogeneus coordinates. Apoint with coordinates has homogeneous coordinates [x''y''] where

$$xh = x\,w\ ,\ yh = y\,w$$

All of your applications in two-dimensions will use w=1,so the point has homogeneous coordinates [x y 1].As long as w is not zero,the point coordintes [x y w] can be normalized to [x/w y/w 1]. Translation can now represented with homogeneous coordinates as

$$[x'y'1] = [x\ y\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & I \end{bmatrix}$$

The result of this multiplication is the point with homogeneous coordinates

$$[\ x{+}Txy{+}Ty\ 1\ ]$$

No point with finite coordinates can have homogeneous coordinates [x y 1]. because if any translation is applied to such a point,it will remain fixed.Point wih coordinates [x y 0) are said to be points at infinity.The term point infinity comes from projective geometry ,where all lines must intersect. Now Euclidean geometry is embeded into a projective plane by adding a point infinity for each set of parallel lines. This point is where the set of parallel meet.

It appears that one coordinate system is necessary to do most transformations while another is needed to do translations. But if this were indeed the matrix notation would be more of a detriment than an aid.Fortunately transformation embedded into the transformations have representations in the homogeneous coordinate system.The homogen can be derived from matrix representations standard coordinates by embedding these matrices in the upper-left corner 3x 3 matrix given in figure 3.6.7.

The representation of the Rotation matrix in homogeneous coordinates

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

It is easy to verify that the following matrix equation gives the equation rotation about the origin.

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

FIGURE 3.6.7

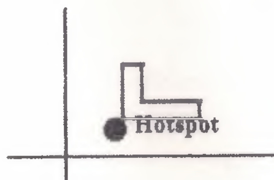
The matrix representation of a transformation embeded in to the matrix representing the transformation in homogeneous coordinates

$$\begin{bmatrix} \boxed{\begin{matrix}\text{Representation} \\ \text{in standard} \\ \text{coordinates}\end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

Matrices provide more than just a notational device for sets of equations: the Rotation of an object around a hotspot also give a simple way to compute the effect of a sequence of operations. If, for example, you want to rotate a triangle around a hotspot different from the origin, you can translate the hotspot to the origin, rotate, and then translate the hotspot back to its original location (figure 3.6.8). If the hotspot is located at the point (5,7), then the translation to the origin is represented by

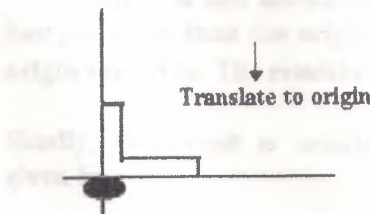
FIGURE 3.6.8

Rotation of an object around a hotspot realized as a composition of transformations



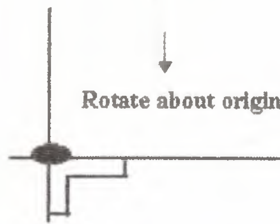
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -5 & -7 & 1 \end{bmatrix}$$

The matrix representing the translation back to the hotspot is the matrix.



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 7 & 1 \end{bmatrix}$$

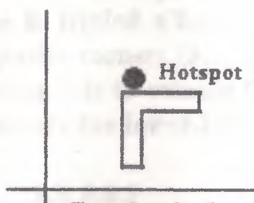
If these transformations are applied consecutively on a point,the resulting point will be at the same location.These transformations are inverses of each other.Notice that the producother. Notice that the product of the matrices representing the two transformations is the multiplicative identity matrix



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

Thus, the matrix representations of these inverse transformations are multipli cative inverses of each other.

Calculating the multiplicative inverse of a matrix when it exists is,in general, a lengthy procedure.For the transformations used in computer graphics,however,the geometric interpretations provide shortcuts for calculating these inverses. A rotation about the origin through an angle has the inverse transformation of a rotation through an angle of - ϕ .The equations



Translate back to original hotspot

$$\cos(\phi) = \cos\phi$$
$$\sin(\phi) = -\sin\phi$$

can be substituted into the matrix representation of the rotation through -ϕ to get the matrix representing that transformation:

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

Notice that the rows of the matrix representing a rotation are equal to the corresponding columns of the matrix representing the inverse of the rotation.Hotspot the multiplicative inverse of a rotation representation is its transpose. Computation of other inverse matrices is left for the reader.

In light of this discussion, the problem of rotating a graphic object around a hotspot other hotspot other than the origin can be reduced to multiplication by a matrix,T, to translate to the origin the ori in. The resultin oint is multi lied by a matrix, R, to rotate through angle ϕ

Finally, this result is multiplied byT^{-1}. The entry ansformation for a point p to point p is given by

$$p' = pTRT^{-1}$$

Matrix multiplication is associative;thus,the product matrix TRT represents the transformation,and the product needs to be calculated only one before the transformation is applied to the points of a graphic object. The product TRT^{-1} follows.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -T_x & -T_y & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ (1-\cos\phi)Tx + Ty\sin\phi & (1-\cos\phi)Ty + Tx\sin\phi & 1 \end{bmatrix}$$

In general,if a transformation can be decomposed into a sequence,transformation,then the oriãinal transformation is represented by the produced of the matrices representing the transformations in the sequence.Anothher application of this technique is the scaling of a geometric figure whith maintaining one of its corners in a fixed position.

If the square with opposite corners (1,1) and (2,2) is to be scaled so that its horizontal side is tripled a'hile its vertical side is doubled, simple scaling create a new rectangle with opposite corners (3,2) and (6,4) (figure 3.6.9). Suppose the lower-left corner (1.l) of the original rectangle is to remain fixed under the scaling (figure 3.6.10). A simple way to achieve this is to translate the lower-left corner of the rectangle to the origin with the matrix.

FIGURE 3.6.9

A rectangle scaled without fixing one of its corners

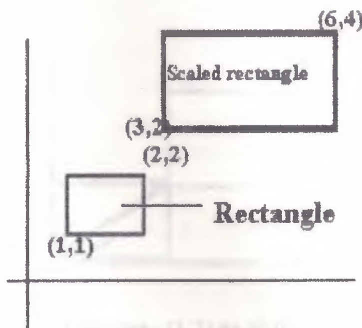
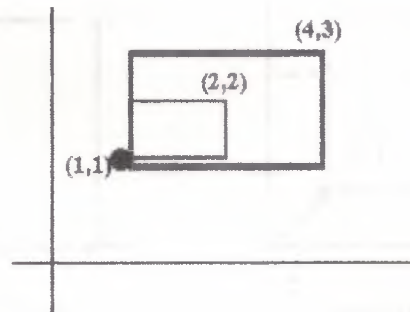


FIGURE 3.6.10

Scaling the rectangle (1,1) (2,2) by 4 in the x-direction and 3 in they y- direction; (1,1) is a fixed point.



Next,apply the scaling matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Finally, multiply that result by the inverse of the original translation matrix

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

This sequence of transformations is illustrated in figure 3.6.11.The scaling depicted in the last example can be generalized to scaling relative to an arbitrary fixed point (xo,yo). This general transformation is characterized by the matrix product

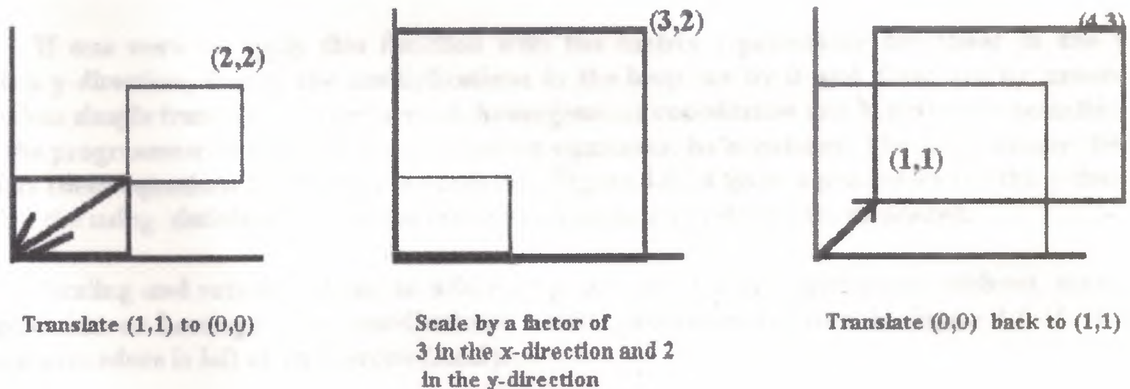
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Xo & -Yo & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ X_0 & Y_0 & 1 \end{bmatrix}$$

which equals

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ (1-S_x)x_0 & (1-S_y)y_0 & 1 \end{bmatrix}$$

FIGURE 3.6.11

The sequence of transformations needed to scale a rectangle and keep the point (1, I) fixed



Another transformation, shearing, alters the shape of an object by adding Shearing a rectangle in the x-direction multiple of one coordinate to the other coordinate of a point. The math representation for a shear with shear factor SHX in the x-direction is

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

FIGURE 3.6.12.

Sharing a rectangular in the x- direction by a factor of 3

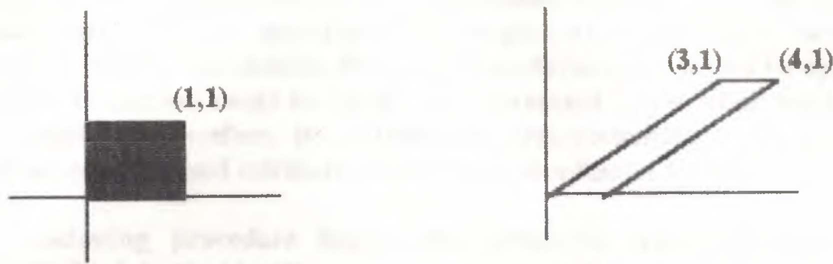


Figure 3.6.12 shows an x-direction shear with shear factor 3.Shear in the y-direction has the matrix representation

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

Once a transformation is expressed as a matrix equation of the form P' PM, it is possible to implement this transformation directly with the procedure in listed in figure 3.6.13.In this transformation the coordinates of a point P a stored in a matrix[P[1],P[2],P[3]],and the coordinates of the transformed point.

If one were to apply this function with the matrix representing the shear in the y-direction y-direction, five of the multiplications in the loop are by 0 and three are by needed. Even when simple translation is performed, homogeneous coordinates are Matrix representations allow the programmer to discover transformation equations. he equations. The programmer then transfers these equations to efficient procedures.. Figure 3.6.14 gives a procedure for the y-shear in which the using definition of a point is used and no matrix references are needed.

Scaling and rotation about an arbitrary point can also be implemented without, matrix multiplication or homogeneous coordinates.A scaling,procedure is given in figure 3:6.15. The rotation procedure is left as an exercise (3.6.9).

General Transformation Equations

Any general transformation, representing a combination of translations ings, and rotations, can be expressed as

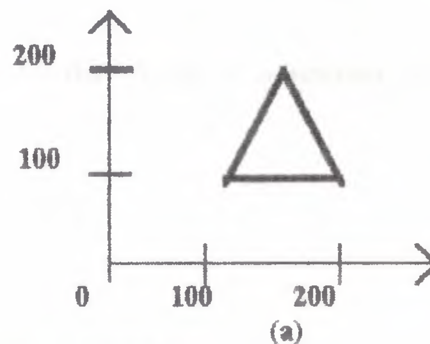
$$[x'\ y'\ 1] = [z\ y\ 1] \begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix}$$

Explicit equations for calculating the transformed coordinates are then

$$x' = ax + by + c \quad , \quad y' = dx + ey + f$$

These calculations involve four multiplications and four additions for each ordinate point in an object. This is the maximum number of computations request for the for the determination of a coordinate pair for any transformation sequence,the individual matrices have been concatenated. Without concatenation, the vidual transformations would be applied one at a time and the number of caltions could be significantly increased. An efficient implementation for the transation operations, therefore, is to formulate transformation matrices, concatention any transformation sequence, and calculate transformed coordinates by Eqs. 5

The following procedure implements composite transformations.A transformation matrix T is initialized to the identity matrix. As each individual transformation is specified, it is concatenated with the total transformation matrix T. When all transformations havc been specified, this composite transformation is applied to a given object. For this eYample, a polygon is scaled, rotated, and translated. Figure 5-13 shows the original and final positions of a polygon transformed by this sequence.



(a)

```
procedure transform_object;
 type
  matrix = array [1..3,1..3] of real;
  points = array [1..10) of real;
 var
  t : matrix;
  x, y : points;
  xc, yc : integer;

 procedure transform-points (n : integer; var x, y : points);
 var k : integer; tempx : real;
 begin
 for k : 1 to n do begin
    tempx :- x[k) * t[1,1] + y(k] * t[2,1) + t[3,1);
    y[k] :-x[k) * t[1,2] + y(k) * t[2,2] + t,[3,2);
    x[k] : tempx
   end
 end; {transform-points}

 procedure fill_area (n : integer; x, y : points);
 begin
    {remainder of fill_area steps}
 end; {fill_area}

 procedure get _vertices _and_center (n : integer; var x, y: points;
    var xc, yc : integer);.
 Begin
  {get vertices and center point}
 end;

 procedure make identity (var m : matrix);
 var r, c : integer;
 begin
  for r :    1 to 3 do
```

```
procedure  combine_transformations (var t : matrix; m : matrix);
var r, c : integer; temp : matrix;
begin
 for r :- 1 to 3 do
  for c: - 1 to 3 do
   temp[r,c] :- t(r,l)*m[l,c] + t[r,2]*m[2,c] + t[r,3]*m[3,c];
 for r : 1 to 3 do
  for c : 1 to 3 do
   t[ r,c ] : temp[ r,c )
end; {cosnbine_transformations}

procedure scale (sx, sy : real; xf, yf : integer);
 var m : matrix;
```

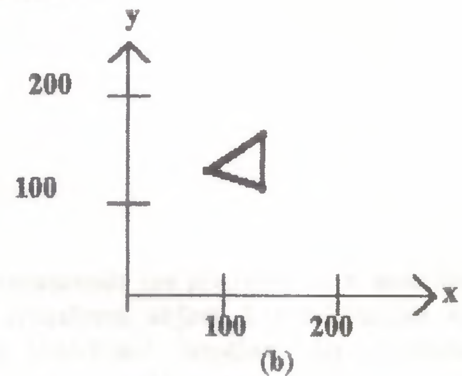


Figure 5-13 A polygon (a) transformed into (b) by the composite operations in procedure transform_object

```
  begin;
   make identity (m);
   m[ 1,1 ] :- sx; m[2,2] : sy;
   m[3,1] :-(1 - sx) * xf;
   m[ 3,2 ] :-( 1 - sy)*yf;
   combine transformations (t, m) {multiply m into t}
  end; {scale}
 procedure rotate (a : real; xr, yr : integer);
   var ca, sa : real; m : matrix;
   function radian_equivalent (a : real) : real;
    begin
     radian_equivalent : a * 3.14159 / 180
    end; {radian_equivalent}
   begin {rotate}
    make_identity (m);
    a : radian_equivalent (a);
   ca : cos(a); sa : sin(a);
   m[l,l] : ca; m(1,2) : sa;
   m[2,1] : -sa; m[2,2] : ca;
   m[3,1] : xr * (1 - ca) + yr * sa;
   m[3,2] :- yr * (1 - ca) - xr * sa;
   combine_transformations (t, m) {multiply m into t}
  end; {translate}
 begin; { transform_object }
   get vertices ánd_center (3, x, y, xc, yc);
   make _identity (t);
   set_ fill areá interior_ style (hollow);
   set_ fill_areá_ color_ index ( 1 );
   fill_area (3, x, y);
```

```
make _identity (t);
set_ fill areá interior_ style  (hollow);
set_ fill_areá_ color_ index ( 1 );
fill_area (3, x, y);
scale  (0.5, 0.5, xc, yc);
rotate  (90, xc, yc);
translate(-60, 20);
fill_area (3, x, y)
end;  {transform_object}
```

5-5 Transformation Commands

Graphics packages can be strutured so that separate commands are provided to a user for each of the transfòrmation operations, as in procedure transform_object..A combination of transformations is then performed by referencing each individual function. An alternate formulation is to use one transformations or a single operation.Sinemations are often performed in combination,a composite transformation can provide a more convenient method for applying transformations.

We introduce the following command to perform composite tranformation involving translation,scaling, nd rotation.

Create_transformation_matrix(xf,yf,sx,sy,xr,yr,a,tx,ty,matrix)

Paramaters in this command are the scaling fixed point (xf,yf) ,scaling sx and sy,rotation pivot point (xr,yr),rotation angle a,translation vector and the output matrix.We assume that this command evaluates the transformations sequence in the fixed order;first scale,than rotate,then translate.The composite transformations for this sequence is than stored in the paramater.A produce for implementing this command perform a concatanation matrices,using the values specified for the input paramaters.

A single transformation or a sequence of two or three trnasformations can be carried out with this transformations command.To tranformation object,a user sets sx=sy=1, a=0,and assigns translation values.The fixed-point and pivot point coordinates could be set to any value,dont effect the trnasformation calculations when no scaling or rotation place. Similarly, ,rotations are specified by setting sx=sy=1,and giving appropriate rotation angle and pivot-point values to paramater and yr.

Since the transformation command can carry out only one fixed transformations we can provide for alternative sequences by defining an a command :

accumulate-transformtion_matrix (matrix_in,xf,yf,sx,sy,xr,yr,a,tx,ty, matrix_out)

Paramaters xf,yf,sx,sy,xr,yr,a,tx, and ty are the same as in the create_transformation_ matrix command.This accumulate operation will take any previously transformation matrix and concanate it with the transformation defined by the paramater list ,in the order matrix_in,scale,rotate the resulting transformation matrix is stored in matrix_out.

Using the accumulate_transformation_matrix command in conjunction create_transfor-

mation_matrix allows a user to perform transformations in any for instance,a translarion follawed by a rotation canot be carried out create command alone.But this transformation sequence could be accommunicate with the following program statament:

create transformation matrix (0, 0, 1, 1, 0, 0, 0, tx, ty, ml);
acoumulate-transformation matrix (ml, 0, 0, 1, 1, xr, yr,a,0,0,m2)

The composite matrix m2 is then applied to the poilnts defining the object translated and rotated .

Several transformation mlatrices could be cornstructed inan application gram.The particular matrix is to be applied to subsequent output could be selected with a function such as

set transformation (matrix)

Parameter mcitrix stores the matrix elements that are to be applied to all quent output primitive commands until the transformation is reset.A method for turning of the transformation operations is to set matriz to the identity matrix.

Another ilnplementation rnethod for applying a particular matrix to a deflned object is to group and label related output pirimitives (picture components). A transformation colmmand could then be structured so that it is applied to a selected object by referencing the label assigned to the grorlp of primitives defining the object. We return to this topic in Chapter- 7.

5-6 Raster Methods for Transformations

The particular capabilities of raster systems suggest an alternate way to approach some transformations. Raster systems store picture information by setting bits in the frame buffer.Some simple transformations can be carried out by manipulating the frame buffer contents directly.Few arithmatic operations are needed, so the transformations are particularly efficient.

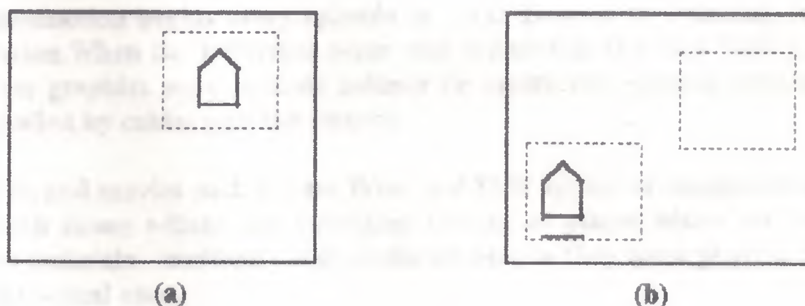


FIGURE 5-22
Bloc transfer of a raster area can be used to translate an object from one scareen position to another

Figure 5-22 illustrates a translation performed as a block transfer of a raster area.All bit settings in the rectangulal- area shown are copied as a block into another of the raster.This translation is accomplished by reading pixel intensities from a specified rectangular area of the r-

astel- into an array, then copying the array back into the raster at the new locatior.The original object could be elased by filling its rectangular area with the background intensity.

Two functions can be provided to a user for carrying out these translation operations. One function is used for reading a rectangular area of the raster into a specified array, and the other is used to copy the pixel values in the array back into the frame buffer. Parameters for a read function are the name of the array and the size and location of the rastel- area. Parameters for a copy function are the name of the array and the copy position within the raster.

Some implementations provide options for the copy function so that bit values in the array can be combined ,with the raster values in various ways. Depending on to another the mode selected, the copy flrnction could simply replace bit settings in the frame buffer witlh those in the array, or a Boolean or binary arithmetic operation could be applied.For- exarnple, bit settings to be- introduced into some area of the frame buffer might be combined vitlh existing contents of the buffer using an and or an or operation.The Boolean exclusive or can be particilularly useful.With the exclusive or mode, two succesive copies of a block to the same raster area restores the values that were originally present in that area.This technique can be used to move an object across a scence without destroying the bacground.

In addition to translation rotations in 90 degree increments could be done using block transfer.a 90 degree rotation is accomplished by copying each row of the block in to a column in the new frame buffer location.Reversing the order of bits within each row rotates the block 180 degree.

Bloc transfer of raster areas,sametimes reffered to as bit bloc transfers or bit-bit,are quick.These techniques form the basis for many animation implementations.Once an array of bit settings has been saved for an object,it can be repeatedly placed at different positions in the raster to simulate motion.

MATRIX REPRESENTATIONS AND THREE-DIMENSIONAL GRAPHICS

This introduction begins every episode of Star Trek as the starship enterpise ambarks on another mission.When the television series was induced in the mid 1960 s,satelite photography and computer graphics were in their infancy.To create the opening sequence the ship, a large model was pulled by cables past the camera.

Star Trek and movies such as Star Wars and 2010 appeal to imaginations of viewes who have grown up with moon adings and television images of places where no man has before.These movies must maintain credibility with audiance and so they must provide images of at least the quality as the actual ones.

Typical fly-by sequence shows a spaceship traversing a planets surface.In the filming of such sequences ships are still commonly superimposed on images of planets produced by computers. There are many different types of planetry surfaces.Same planets have cratered surface like that of the moon.Some have a mixture of cratered and desrt-canyon surfaces similar to that of Mars.The outer,gas-giant planets of the solar system are surrounded by bands of turbulent clouds,while some of their moons are covered by an almost mirorlike ice.The techniques used to create these surface textures are presented.

Computer graphics has progressed to the point where a fly-by sequence can now be generated entirely by computer.Newtonian physics dictates the motions of two or more bodies relative to each other and is the starting point for creating a realistic scence.Once an orbital system has been modeled,it must be rotated and translated to appear in anadvantageous position on the screen.In addition,physical models are traditionaly created in a righthand coordinate system,while graphics systems have a left-hand orientation.Therefore ,all points computed from physical models must be transformed into left-hand orientation,the rotated and translated to be appropriately aligned with the camera.Finally,the image must be projected in to screen coordinates.

8.1 Three-Dimensional Graphics

The move from two-dimensional to three-dimensional graphics creates two categories of problems. The first concerns the extension of the two-dimension tools you have developed to three-dimensional tools.The second class problems deals with the represnta-tion of three-dimensional objects in two dimensions.

In this chapter you will work only with the extension to three dimensions While coordinate systems may be extended by adding one coordinate,the orientation of positive axes is more complicated in three dimensions. If you assume the xy-plane is the graphics screen, you must decide whether the positive z-axis will point out of or into the sereen.

Polygons in two dimensions consist of vertices and edges.When you move to three dimensions,polygons also have faces.The representations of polygon surfaces and curved surfaces are presented in section 8.2.

Three-dimensional transformations are also more complicated than transformations mations in two dimensions.In addition to rotations about a point,you must no consider rotations consider rotations about a line.It is often necessary to describe a transformations as a sequenceof simpler transformations.Matrix algebra becomes indispensable tool for creating these composite transformations.The matrixforms of three dimensional transformations such as translation and rotation are given in this chapter.These solutions are applied to the problem of establishing the viewer's position in a graphics system.

The remaining chapters address the projection of t hree-dimensional images on a two dimensional surface, the hiding of surfaces and lines that are not visible from the viewer's position, and the creation of realistic images. The solutions to these problems rely heavily on vector operations.The vector products also provide an alternative approach to finding the matrix representation of some commonly used three-dimensional transformations. An example of this approach is given in section 8.7.

8.2 Representing Graphic Objects in Three-Dimensions

Representing three-dimensional objects is considerably more difficult than adding an extrac oordinate to the Cartesian system used in two-dimensional systems.Even if you systems.Even if you elect to retain the normal screen position of the x- and y-axes, the positive part of the z-axis can point either into or out of the screen. It is also possible to have the z-axis in the plane of the screen and one of the ther axes perpendicular to the screen. It is equally possible to have none of the axes in the plane of the screen. In this section you determine the orientation of the z-axis relative to the x- and y-axes. You also look at various methods used to represent objects in such a system. The more difficult problem of positioning the axes relative to the screen and viewer is covered later in the chapter.

The position of the positive z-axis points relative to the x-line determines whether the coordinates of a specific point will be (x,y,z) or op(x,y,-z). If the z-axis points out of the screen when the x- and y-axes are in their normal positions on the screen, the coordinate system is called a right-hand system. If you align your right hand so that your fingers curve from the x-axis to the y-axis, then your thumb points along the positive z-axis (figure 8.2.1). If however, your thumb points along the negative z-axis, then the system is a The right-hand referenee system left-hand system. In this system, if the fingers of your left hand are positioned so that they curl from the x-axis to the y-axis, then your left thumb points in the positive z-direction (figure 8.2.2).

Whether to use the right- or left-hand system often depends on the positions of the important objects in the application you are working wittr.For some applications, the solution is easier to derive with one orientation than with the other. In such cases, right- and left-handed systems may be used in the same problem (Blinn 1988).

Many problems are easier to solve within a coordinate system other than the Cartesian system.Both the cylindrical and spherical coordinate systems can provide simple descriptions of objects that would be difficult to derive in the z-axis Cartesian system.Nevertheless most graphics packages use the Cartesian system, which, as a result system, which, as a result, limits the usefulness of non-Cartesian systems in computer graphics because objects represented in these systems must be converted to the Cartesian
system before they can be displayed.

FIGURE 8.2.1
The right -hand refernce system

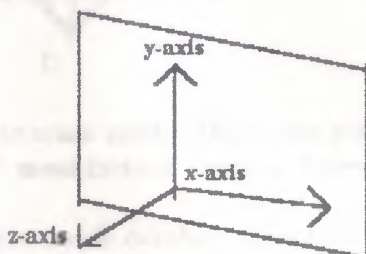
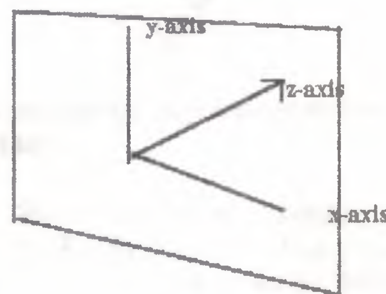


FIGURE 8.2.2
The left hand reference system



Once objects are represented in world coordinates, they must be converted The left-hand reference system into screen coordinates. Methods used to convert world coordinates to normalized screen coordinates in two-dimensional systems are also used in three-dimensional systems.Homogeneous coordinates are often used in three axis dimensions as they were in two dimensions to simplify certain matrix representations . The homogeneous coordinates of a point (x,y,z) are usually given as (x,y,z,l).

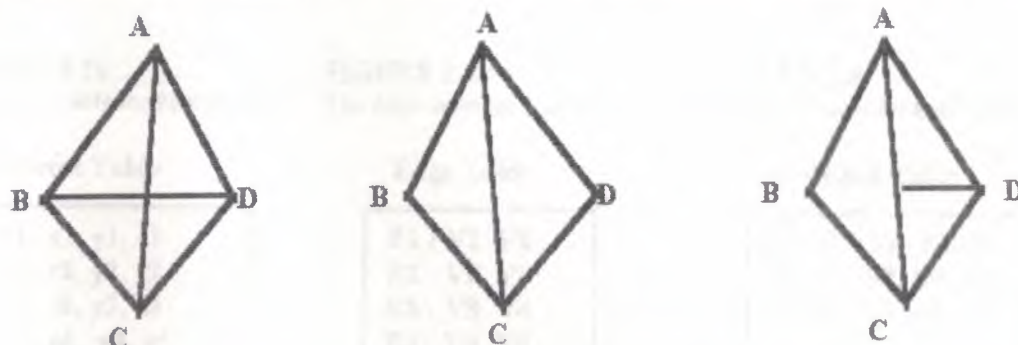
Images of solid objects can be created in many ways, with varying levels of complexity, depending on the type of information required by an application.When the surface appearance of an object is the only requirement, geometric or x-axis shape modeling is used to create the descriptions of the solid objects used in creating their images.This section presents two commonly used methods for performing geometric modeling of three-dimensional surfaces; polygonal meshes and spline surfaces.Objects with regular geometric surfaces, such as buildings and furniture, can be decomposed into a set of polygonal surfaces.This representation is called a polygon mesh.

A polygon mesh can also approximate curved surfaces (figure 8.2.3). When this technique is applied,picture quality can be improved by using a finer mesh in the curved areas , but the result

will not be realistic. To achieve realistic curved surfaces, the spline techniques discussed in chapter 7 can be extended to three dimensions.The parametric equations utilized in this extension have two parameters rather than one and are normally cubic polynomials. Because of these characteristics, they are often called bicubic patches.

Engineers construct mechanical devices by combining pieces such as blocks, pyramids, and cylinders well as spline surfaces.This type of constructive representation is called solid modeling .It is more complicated than geometric modeling and will be covered.You have displayed polygons in two dimensions by connecting successive vertices with line segments.If you attempt the same type of construction with three-dimensional objects, one set of lines may represent many different polygonal surfaces. In figure 8.2.4 (left), the polygon mesh and vertices A, B,C, and D, can be interpreted in different ways.One interpretation (figure 8.2.4 [center]) is the solid with four triangular surfaces, while another (figure 8.2.4)

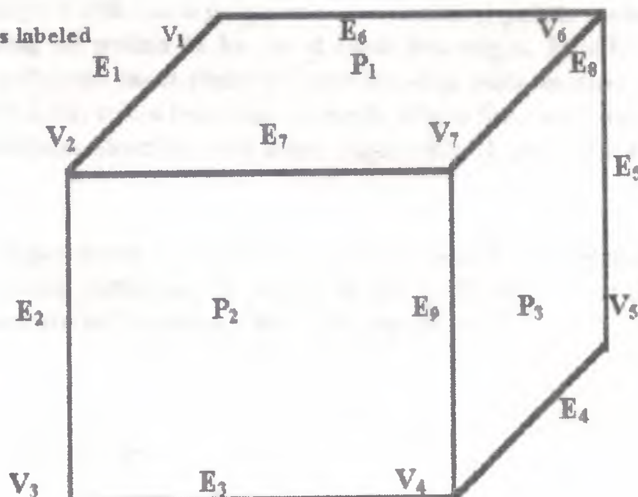
FIGURE 8.2.4
The polygon mesh with vertices A, B, C and D (left) can be interpreted in different ways. One way is as a solid with four triangular sides (center); another (right) is as the same solid with the side ACD missing.



[right]) is the same solid without the side ACD. To avoid this ambiguity, any representation of such a solid must include a way to determine its polygonal surfaces.

A polygon mesh consists of vertices, edges, and polygons. Figure 8.2.5 is an example of a polygon mesh with vetrtices {V_1 , V_2 V_3 V_4 V_5,V_6,V_7}edges,{ E_1,E_3, E_4, E_5,E_6 E_7, E_8,E_9 } and polygons { P_1,P_2,P_3 } The vertices are defined by their coordinates. The polygons can be defined by listing their vertices in clockwise or counterclockwise order. All polygons should have their

FIGURE 8.2.5
A cube with vertices, edges, and polygons labeled



vertices listed in the same order. The polygon P, in figure 8.2.5 could be described by ((xl, yl, zl), (x2, y2, z2), (x7, y7, z7), (x6, y6, z6)), where the vert V; is assumed to have coordinates (xi, yj, zl).

For a single polygon,this type of representation is relatively efficient. Yet even in the comparative small polygon mesh given in figure 8.2.5, the vertex V_7 has its coordinates listed for each polygon. Other vertices, such as Vz, also appear in more than one vertex list. To save room, a vertex table listing the coordinates of a vertex is created, and references to a given vertex are made through its subscript in the table (figure 8.2.6). Representing polygons as the list of vertex subscripts improves storage efficiency but does not provide explicit methods for determining shared edges and vertices.

To expedite the retrieval of relationships between parts of the polygon mesh, a list of edges and their end end points is kept in an edge table, and the polygons are described in terms of are described in terms of the subscripts of the edges in the table. The edge table for figure 8.2.5 is given in figure 8.2.7, and the polygon table is given in figure 8.2.8.

FIGURE 8.26
The vertex table for figure 8.2.5

Vertex Table

Vertex Table
V1: x1, y1, z1
V2: x2, y2, z2
V3: x3, y3, z3
V4: x4, y4, z4
V5: x5, y5, z5
V6: x6, y6, z6
V7: x7, y7, z7

FIGURE 8.2.7
The edge table for figure 8.2.5

Edge Table

Edge Table
E1 : V1 V2
E2 : V2 V3
E3 : V3 V4
E4 : V4 V5
E5 : V5 V6
E6 : V6 V1
E7 : V7 V2
E8 : V6 V7
E9 : V7 V4

FIGURE 8.2.8
The polygon table for figure 8.2.5

Polygon Table

Polygon Table
P1 : E1, E6 ,E8, E7
P2: E2, E7, E9, E3
P3: E4, E9, E8, E5

Not all lists of vertices, edges, and surfaces describe real objects. Tables the describe real objects called consistent. Figure 8.2.9 is an example of vertex not on any edge. If an edge is drawn from D to A (figure 8.2.10), the resulting image is still not a polygon mesh. Isolated points such as A in figure 8.2.9 can be avoided by requiring all points to be on at least two edges. Finally even if each polygon is properly formed, all polygons must share at least on edge with another polygon or the surface is not connected (figure 8.2.11) takes less time to check tables for consistency if the edge table is extended to include the surfaces abutting each edge. Figure 8.2.12 gives the extended edge table for figure 8.2.5.

Some implementations of the polygon mesh require that each polygon lie one plane. Recall that Recall that three non-collinear points determine a plane. If the polygon has more three vertices, you must check that all vertices are in the same plane. The equation of
a plane has the form

$$Ax + By + Cz + D = 0$$

FIGURE 8.2.9
The vertex A is not on any edge; therefore,A cannot be listed as part of this polygon.

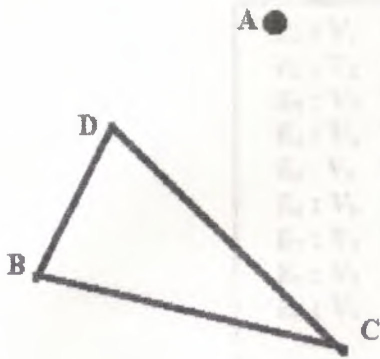
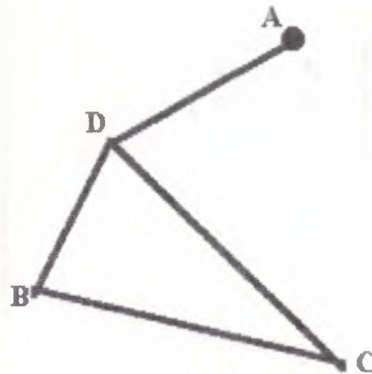


FIGURE 8.2.10
This is not a polygon mesh because A is an isolated point.



The coordinates (x,y,z) of any point in a plane will satisfy the e uation of the plane.

This is not a polygon mesh bec·ause A is Any three non-collinear points determine a plane.By choosing any three an isolated point. non-collinear vertices of a polygon $(x_1,y_1,z_1,)$, (x_2,y_2,z_2), and (x_3,y_3,z_3), you can substitute each into the equation of the plane and normalize to get the three equations

$$(A/D)\, x_1 + (B/D)\, y_1 + (C/D)\, z_1 = -1$$
$$i = 1,2,3$$

in the three unknowns A' = A/D, B' = B/D, and C' = C/D; or

$$A'\, x_1 + B'\, y_1 + C'\, z_1 = -1$$
$$A'\, x_2 + B'\, y_2 + C'\, z_2 = -1$$
$$A'\, x_3 + B'\, y_3 + C'\, z_3 = -1$$

These equations can be solved using Cramer's rule, provided the determinant of the coefficients

FIGURE 8.2.11
This figure does not represent a polygon mesh because the two polygons do not share a common edge.

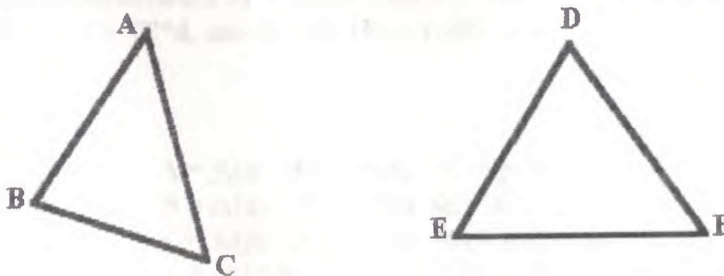


FIGURE 8.2.12
The extended edge table for the polygon mesh in figure 8.2.5

Edge Table

$E_1 : V_1 \ V_2 \ P_1$
$E_2 : V_2 \ V_3 \ P_2$
$E_3 : V_3 \ V_4 \ P_2$
$E_4 : V_4 \ V_5 \ P_3$
$E_5 : V_5 \ V_6 \ P_3$
$E_6 : V_6 \ V_1 \ P_1$
$E_7 : V_1 \ V_2 \ P_1 \ P_2$
$E_8 : V_6 \ V_7 \ P_1 \ P_3$
$E_9 : V_4 \ V_7 \ P_2 \ P_3$

$$\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

is not zero. This will always be true if the vertices are chosen so that they are not collinear, and every non-degenerate polygon has at least three non-collinear vertices. (Why?)

If the determinant of the coefficients is d, then the solutions are given as

$$A' = \begin{vmatrix} -1 & y_1 & z_1 \\ -1 & y_2 & z_2 \\ -1 & y_3 & z_3 \end{vmatrix} /D \qquad B' = \begin{vmatrix} x_1 & -1 & z_1 \\ x_2 & -1 & z_1 \\ x_3 & -1 & z_3 \end{vmatrix} /D \qquad C' = \begin{vmatrix} x_1 & y_1 & -1 \\ x_2 & y_2 & -1 \\ x_3 & y_3 & -1 \end{vmatrix} /D$$

FIGURE 8.2.13
The vector (A,B,CJ normal to a plane can be computed from the equations of the plane.

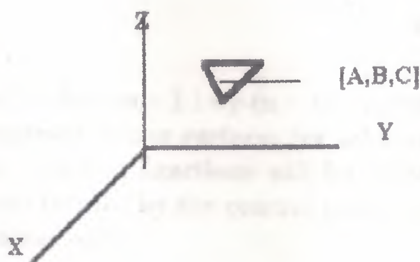
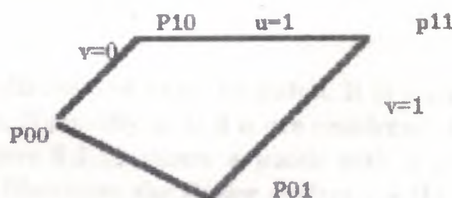


FIGURE 8.2.14
The boundary values for the paramaters in the parametric representation of a polygon



To avoid repeated division by d, apply simple algebra to get an alternative set of solutions: A = A'*d, B = B'*d, C = C'*d, and D = d. These reduce to

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

The vector (A, B, C) is normal to the plane (figure 8.2.13). Hence, for fixed A , B, and C, different D values give planes that are parallel.

When a polygon mesh does not produce a smooth enough representation of an object, it is replaced by one of the bicubic models. These models are the three-dimensional counterparts of the spline curves introduced in chapter 7. A major difference between spline curves and spline surfaces is that parametric equations of a surface have two variables. Linear interpolation is a simple example of a parametric representation of a surface.If. for example, you want to develop linear interpolation of the surface bounded by the four points Po,P1o,Po1,P11, (figure 8.2.14), then the parametric equation for this surtace is

$$P(u,v) = P_{00}*(I - u)*(1 - v) + P_{10}*u*(1 - v) + P_{01}*v*(1 - u) + P_{11}u*v$$

where u and v range from 0 to l.

The region bounded by the four points is called a patch.

A smoother, more adjustable surface can be achieved by extending the Bézier curves to three dimensions. Recall that two-dimensional Bézier curves are defined by

$$P(t) = \sum_{k=0}^{n} Bk,n(t)Pk$$

where the blending functions Bk,n are given as

$$Bk,n(t) = \left|\begin{matrix} n \\ k \end{matrix}\right|^{k} (1 - t)^{n-k}$$

The parametric form of Bézier surfaces is given by

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} Bj,m(u)Bk,n(V)pj,k$$

where the (m + 1)-by-(n + 1) control points pj,k are distributed over the patch. It is possible to construct Bézier surfaces for arbitrarily large m and n. Normally m and n are restricted to 3 so the blending functions will be cubic polynomials.Figure 8.2.15 shows a patch with a polygon mesh formed by the control points.while figure 8.2.16 illustrates the Bézier surface for this set of control points.

If greater numbers of control points are needed for a desired effect, the surface can be divided surface can be divided into more than one patch.To maintain continuity along the common successive patches, certain properties must be maintained.In figure 8.2.17, the triples of control points (P1,Q1,R1), (P2,Q2,R2), (P3,Q3,R3), and (P4,Q4,R4) must be collinear. In addition, the ratios

$$\frac{\text{length } (P_iQ_i)}{\text{length } (Q_iR_i)}$$

must be the same constant for i = 1, 2, 3, 4.

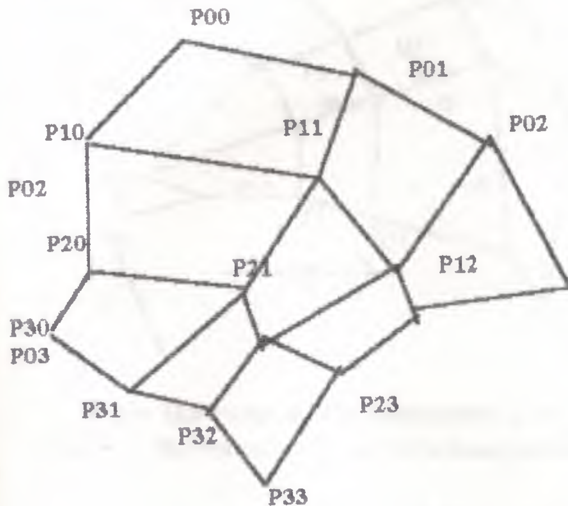


FIGURE 8.2.15
A patch with the polygon mesh formed by the control points.

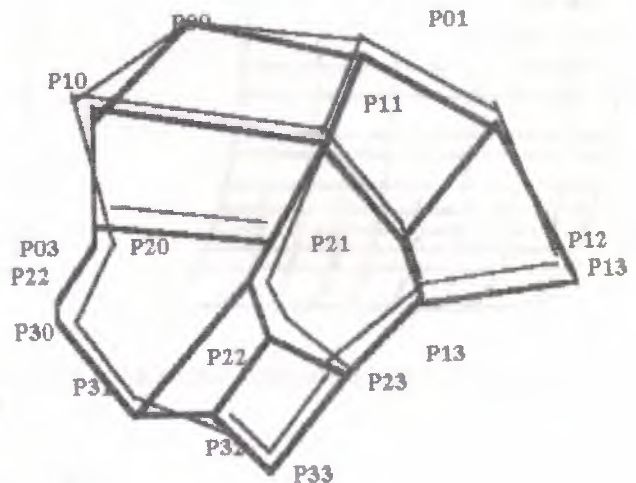


FIGURE 8.2.16
The bezier surface constructed from e Set of control points in figure 8.2.15

B-splines can be generalized to three dimensions in much the same way that Béizer curves are generalized.The parametric equation for B-spline surfaces is

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} N_{j,s}(u) N_{k,t}(v)$$

where the blending functions $N_{j,s}$ and $N_{k,t}$ are of degree s - 1 and t - 1 respectively. The control points $p_{j,k}$ define the patch, but are generally not on the patch.

For any type of spline construction, the set of patches depends heavily on the shape of the object being modeled.The density of control points in an area is proportional to the curvature of the object. Planar areas can be treated with 1 patch and 16 control points; areas of high curvature require more control points and, hence, a large number of patches (figure 8.2.18). Most systems allow the user to specify the smoothness of the surface but reserve the actual subdivision into patches for the program. Subdivision is usually done recursively.

Computing the values for a parametric bicubic surface requires numerous multiplications. The number of multiplications can be reduced by applying Horner's rule for factoring:

$$f(u)=au^3+bu^2+cu+d=[(au+b)u+c]u+d$$

But all the multiplications are real multiplications.The values can be computed more efficiently by using differences in function values between consecutive points.

FIGURE 8.2.17
Two successive patches with common edge Q1,Q2,Q3,Q4

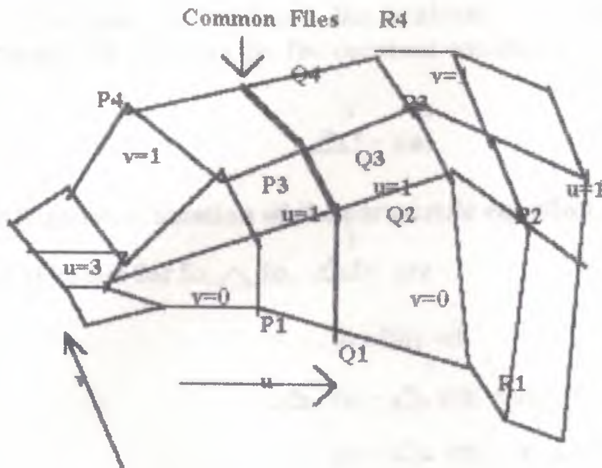
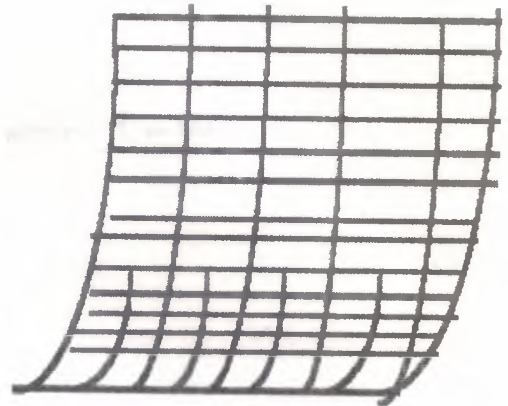


FIGURE 8.2.18
Areas with high curvature, such as the bottom section of this surface, require more patches for a realistic renderin R.



When the value of the parameter u is increased by a small amount, â > 0 , the amount of change in the function value Of is computed by

$$Of(u) = f(u + \hat{a}) - f(u)$$

This difference is called the forward difference. The value of f(u + â) can be computed directly from f(u): $\triangle$

$$f(u + \hat{a}) = f(u) + Of(u)$$

If the values of f are computed iteratively, then f; is associated with f(u), and fl+, with f(u + 0). With these substitutions the equation becomes

$$fl+1 = fl + \triangle fl$$

The forward difference for the cubic polynomial can be computed directly as follows:

$$\triangle f(u) = f(u + \hat{a}) - f(u) = 3au\hat{a}^2 + u(3a\hat{a}^2 + 2b\hat{a}) + a\hat{a}^2 + b\hat{a}^2 - + s$$

Unfortunately, this formula is still quadratic. To reduce the degree of Of, you find its forward difference, $\triangle^2 f$:

$$\triangle^2 f(u) = \triangle f(u + \hat{a}) - \triangle f(u)$$

In an iterated process:

$$\triangle^2 fi = \triangle fi{+}i - \triangle fi$$

Evaluating $\triangle^2 f(u)$, you get

$$\Delta^2 f(u) = 6a\hat{a}^2u + 6a\hat{a}^2 + 2b\hat{a}^2$$

You now have reduced the problem to a linear equation, but if you find the forward difference for Ozf you get the constant equation

$$\Delta^3 f = 6a\hat{a}^3$$

The iterative evaluation of the parametric equation begins with u = 0, so the initial values for fo, Δ fo, Δ^2fo are

$$f0 = f(0) = d$$

$$\Delta fo = \Delta f(0) = aa^3 + b\hat{a}^2 + c\hat{a}$$

$$\Delta^2 fo = \Delta^2 f(0) = 6a\hat{a}^3 + 2b\hat{a}^2$$

At the ith iteration, fi, Δfi and Δ^2fi are known. Their $(i+1)^{th}$ counterparts are computed with the following sequence of equations:

$$fi + 1 = fi + \Delta fi$$

$$\Delta fi{+}i = \Delta fi + \Delta^2 fi$$

$$\Delta^2 fi{+}1 = \Delta^2 fi + \Delta^3 fo$$

Figure 8.2.19 is an example of forward differences applied to the parametric function f(u) = 2u3 + u2 - 2u + 1 for u in [0,1 J and S = 0.1.

For spline surfaces, there are separate parametric equations for each of the coordinate directions that is, x(u), y(u). and z(u).

FIGURE 8.2.19

Forward differences applied to the function f(u) = 2u3 + u2 - 2u + I.

i	U	f(u)		Δ fi		Δ^2 fi		Δ^3 fi
0	0.00	1.0000	+	-0.1880	+	0.0320	+	0.0120
1	0.10	0.8120	+	-0.1560	+	0.0440	+	0.0120
2	0.20	0.6560	+	-0.1120	+	0.0560	+	0.0120
3	0.30	0.5440	+	-0.0560	+	0. 0680	+	0. 0120
4	0.40	0.4880	+	0.0120	+	0.0800	+	0. 0120
5	0.50	0.5000	+	0.0920	+	0.0920	+	0.0120

6 0.60	0.5920 +	0.1840 +	0.1040 +	0.0120
7 0.70	0.7760 +	0.2880 +	0.1160 +	0.0120
8 0.80	1.0640 +	0.4040 +	0.1280 +	0.0120
9 0.90	1.4680 +	0.5320 +	0.1400 +	0.0120
0 1.00	2.000 +	0.6720 +	0.1520 +	0.0120

8.3 Matrix Representation of Translation and Scaling

Each two dimensional matrix representation introduced in section 3.6 has its hree-dimensional counterpart. Translation and scaling are realized in three dimensions by simply adding one row and column to their matrix representations to account for the transformation of the third coordinate.Homogeneous coordinates must again be used to represent points for translations, and as in two dimensions, the amount of translation in each direction is entered in the last row of the matrix. If the amounts of translation in each direction are given as Tx, Ty, and Tz then the representation matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

It is often necessary to translate a point to the origin (figure 8.3.1). If the homogeneous coordinates of the point are (a,b,c, 1), then the translation matrix is given by

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

To invert a translation, points must be moved an equal distance in the opposite direction. What is the matrix representation of this translation?

Scaling is done by multiplying the homogeneous coordinates of a point by a matnx with the scaling factors on the mam diagonal. The following is an example of such multiplication:

$$[x'y'z'1] = [xyz1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here SX, Sy, Sz are the scaling factors. Recall that all vertices have their coordinates multiplied by scaling factors, so all vertices of the scaled object are moved unless one of the vertices is the origin (figure 8.3.2).

To scale a polygonal object and fix one of its vertices, say (xo,yo,zo, I) (figure 8.3.3), you first translate the object so that the vertex (xo,yo,zo,l) moves to the origin (figure 8.3.4). Thescaling takes place (figure 8.3.5) and is followed by the inverse of the original translation
inverse of the original translation (figure 8.3.6).

FIGURE 8.3.1
Translation of the point (a,b,c,I) to the origin

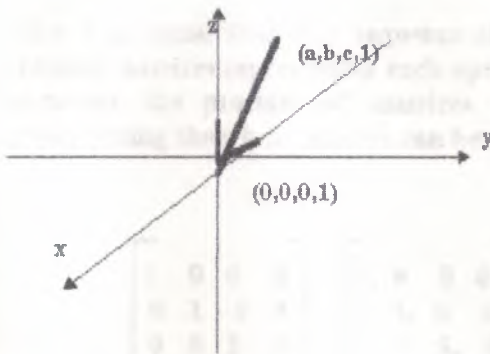


FIGURE 8.3.2
All vertices of a polygon are moved by scaling unless one of the vertices is the origin.

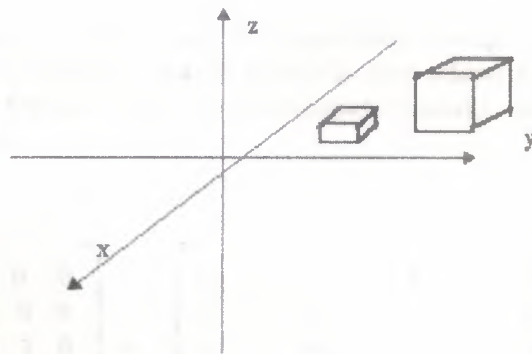


FIGURE 8.3.3
It is possible to scale a polygon and fix a vertex, but it must be Done through a sequence of transformations.

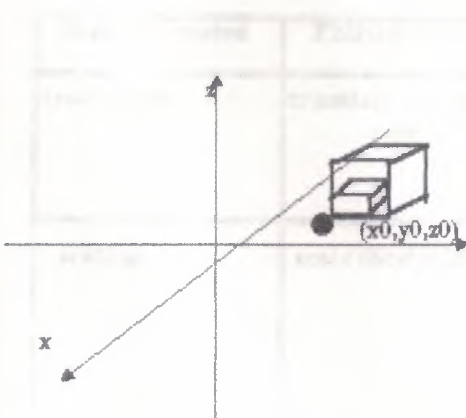


FIGURE 8.3.4
The first transformation in scaling with a fixed point is the translation of the fixed point to the origin.

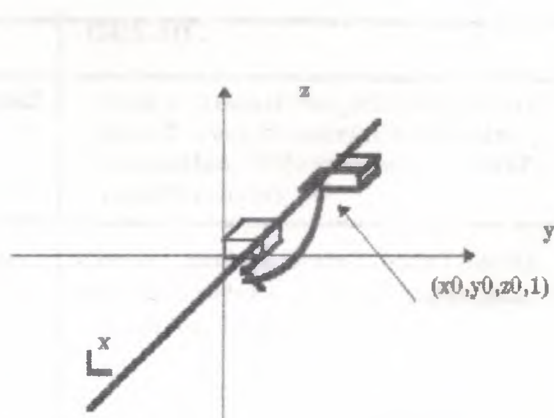


FIGURE 8.3.5
Next, the polygon is scaled by the desired factors.

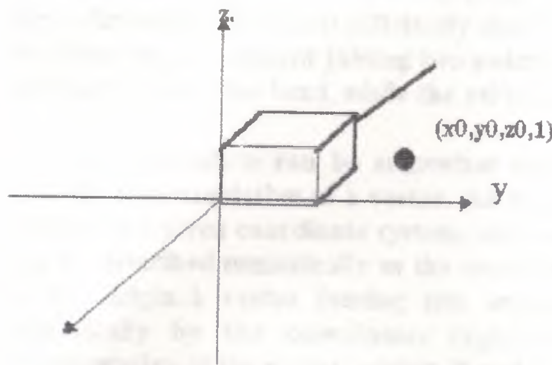
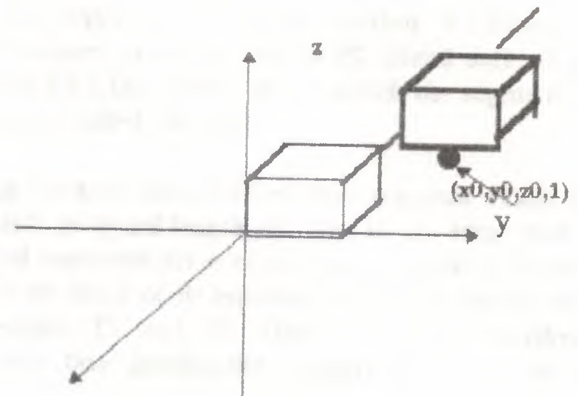


FIGURE 8.3.6
Finally, the polygon is translated so that the origin is backto the fixed point



You may recall that this sequence of operations can be realized by consecutive multipli-cations matrices representing each operation. Associaivity of matrix multiplication allows you to compute the product of matrices before performing the transformation.Thus,the matrix representing the whole process can be computed as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ (1-Sx)x_0 & (1-Sy)y_0 & (1-Sz)z_0 & 1 \end{bmatrix}$$

Both PHIGS and GKS-3D have functions for creating matrices representing translations and scaling (figure 8.3.7).

FIGURE 8.3.7
Functions used to create translation and scaling matrices in PHIGS and GKS-3D.

Matrix Created	PHIGS	GKS-3D
translation	translate (DelateX,DelatY,DeltaZ: real)	Create _Translation_3(DeltaX,DelataY, DeltaZ :real;M:matrix);Accumulate_ Translation _3(DeltaX,DeltaY,DeltaZ: real;M:matrix);
scaling	scale (ScaleX,ScaleY,ScaleZ:Real)	Create ;_Scale_3(ScaleX,ScaleY,ScaleZ real;M:matrix);e_Scale_3(ScaleX,ScaleY, ScaleZ: real;M:matrix

8.4 Review of Vector Operations

Just as a plane is best described by the equation given in section 8.2,lines in three dimensions are most efficiently described by vectors. A vector can be P2 (Head defined as the directed line segment joining two points (figure 8.4.1).The point toward which the segment is directed is called the head, while the other end point is called the tail.

This definition can be somewhat misleading because the directed line segment (Tail) is actually a epresentative of a vector. All line segments of equal length directed in the same way , relative to a given coordinate system, are considered representatives of the same vector. A vector can be described numerically as the coordinates of the head of its representative that has its tail at the origin.A vector joining two arbitrary points, Pl and P2 (figure 8.4.2),is described numerically by the coordinates (x,y,z),where the line joining,the origin to (x,y,z) is a representative of the vector joining Pl and P .

Addition of two vectors can be defined in much the same way as addition of two matrices is defined . If V1 = (x1,y1,z1) and V2 = (x2,y2,z2) are vectors, then the sum is

$$V_1 + V_2 = (X_1 + X_2, Y_1 + Y_2, Z_1 + Z_2)$$

The sum has a geometric interpretation.The head of the representative of V2 with tail at the head V, is the head of the sum vector (figure 8.4.3).

While vector addition is important in many applications, you will find vector subtraction even more useful. Vector subtraction is defined by

$$V_1 - V_2 = (X_1 - X_2, Y_1 - Y_2, Z_1 - Z_2)$$

FIGURE 8.4.1
The vector joining the point P1 to the point P2

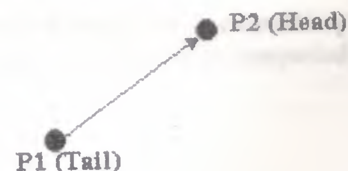


FIGURE 8.4.2
The vector joining P, and PZ can be written as the triple (x,y,z)

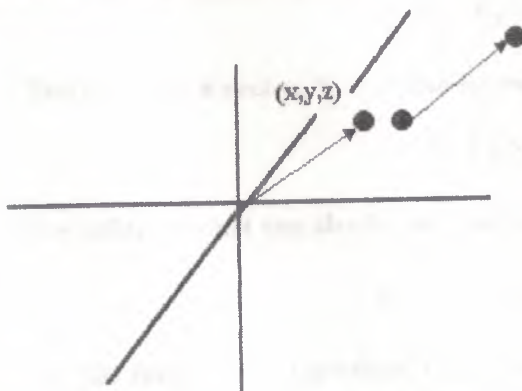


FIGURE 8.4.3
The sum of vectors б + c can be constructed by connecting the tail of b with the head of the representative of c with the tail at the head of b

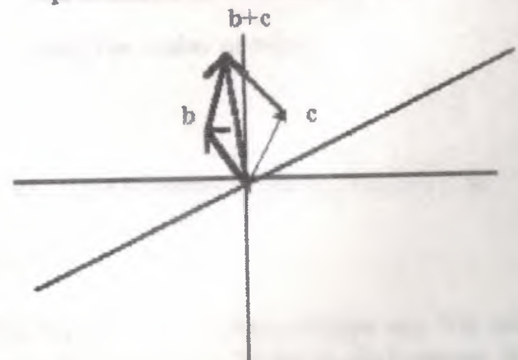


FIGURE 8.4.5
The difference c - b can be seen as the sum c + -b.

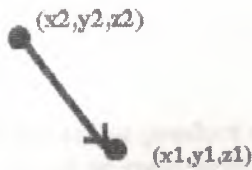
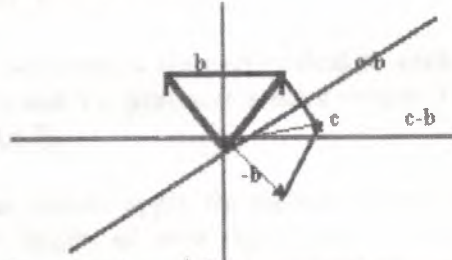


FIGURE 8.4.4
To compute the numerical representation of the vector joining the point with coordinates (x2,y2,z2) and (x1,y1,z1) you need only subtract the vector (x2,y2,z2) from the vector (x1,y1,z1) to get (x1-x2,y1-y2,z1-z2).



If $-V = (-x,-y,-z)$ when $V = (x,y,z)$, then $V_1 - V_2 = V_1 + (-V_2)$ and the geometric interpretation of a vector computed by subtraction is just the vector joining the head of V_2 to the head of V I (figure 8.4.4). Thus, the vector representing the line joining two points (x_1,y_1,z_1) and $(x_2,y_2,z_2$)(figure 8.4.5) is computed by

$$(x_1 - x_2,\ y_1 - y_2,\ z_1 - z_2)$$

There are three different types of multiplication associated with vectors: need only multiplication by a real number, the scalar or dot product, and the cross or vector (x2,y2,z2) vector product. A vector $V = (x,y,z)$ is multiplied by a real number r when all its coordinates are multiplied by r:

$$rV = (rx,ry,rz)$$

The length of a vector, $V = (x,y,z)$, is computed by

$$|V| = x^2 + y^2 + Z^2$$

When the direction of a vector is more important than its length, it is convenient to use the unit vector in that direction. The unit vector u in the direction of vector V is computed from V by multiplying by real number 1/ V that is,

$$u = (1/|V|)V$$

The scalar or dot product of two vectors,$V_1=(x_1,y_1,z_1)$ and $V_2=(x_2,y_2,z_2)$ is the real number

$$V_1 . V_2 = x_1x_2 + y_1y_2 + z_1z_2$$

The length of a vector, V, can also be computed by using the scalar product

$$|V| = \sqrt{V \cdot V}$$

The scalar product can also be defined in terms of the angle 8 between vectors V1 and V2;

$$V_1 \cdot V_2 = |V_1|\,|V_2|\cos\theta$$

In this form, the dot product is the product of the length of one of the vector say Vz, and the length of the projection of the other vector, V1, onto it (fig, 8.4.6). If the angle between the

two vectors is 90°, the vectors are said to orthogonal, or perpendicular, to each other.Since cos90 = 0, the scaler product of two orthogonal vectors is 0. This formula also can be used compute the angle between two vectors because

$$\cos \theta = \frac{V_1 . V_2}{|V_1| . |V_2|}$$

While the scalar product can be used to tell if two vectors are perpendicul to each other, the vector product or cross product, of two vectors,V_1 and V_2, produces a third vector, $V_1 * V_2$, which is perpendicular to each of the vectors (figure 8.4.7).

To determine the direction of the perpendicular vector, apply the right-hand rule-that is, position the fingers of your rule-that is, position the fingers of your right hand so they point vecctor in the product toward the second vector.The direction our thumb point is the direction of the vector product.To compute the vector product of vector $V_1 = (x_1,y_1,z_1)$ and $V_2 = (x_2,y_2,z_2)$ use the following formula:

$$V_1 * V_2 = (y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2)$$

This form of the vector product is difficult to remember, so the following determinant form sed as a memory aid:

$$V_1 * V_2 = \left[\begin{vmatrix} 1 & 0 & 0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \begin{vmatrix} 0 & 1 & 0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \begin{vmatrix} 0 & 0 & 1 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \right]$$

FIGURE 8.4.6

The projection of V, onto V2 has length □ V, □ cos8.

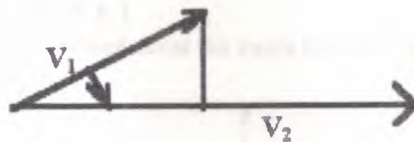
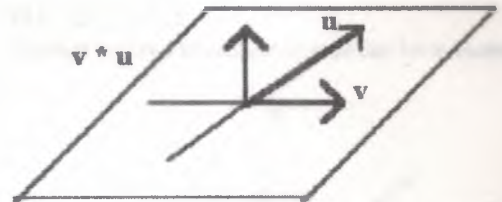


FIGURE 8.4.7

The vector product of two vectors v x u is a vector v * u is a vector perpendicular to the plane of v and ,v u and v x u forming a right-hand system.



If vectors V_1 and V_2 have the angle θ from V_1 to V_2 then the vector product can be computed with

$$V_1 * V_2 = U |V1| |V2| \sin\theta$$

Where U is the unit vector perpendicular to the plane containing V1 and V2 and for which VI, V2, and U, in that order, form a right-handed system.

8.5 ROTATIONS

In three-dimensional spaces, objects may be rotated about a line called an axis Rotations about the x-, y-,gr z-axes are relatively simple, corresponding closed to a rotation about the in in a plane. Rotations about an arbitrar axis an implemented as a sequence of simpler transformations.

When an object is rotated about the z-axis, the z-coordinate of every point is maintained while the x- and y coordinates are changed (figure 8.5.1). The new coordinates are computed as they were for rotations about the origin in two dimensions.

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

where θ is the angle of rotation. This set of equations can be re resented by the matrix equation in homogeneous coordinates

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

A rotation about the x-axis will fix all x-coordinates y and z are recalculated (figure 8.5.2) .To maintain the proper relotionship between the three axes,the three coordinates are transformed by the cyclic permutation;x replaces z,y replaces x, and z replaces y.This permutation results in the following rotation equations:

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

FIGURE 8.5.1
A. point rotated about the z-axis has its z-coordinate fixed.

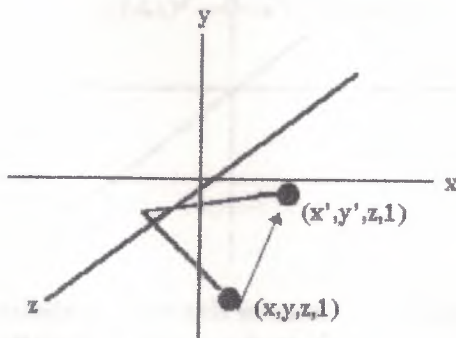
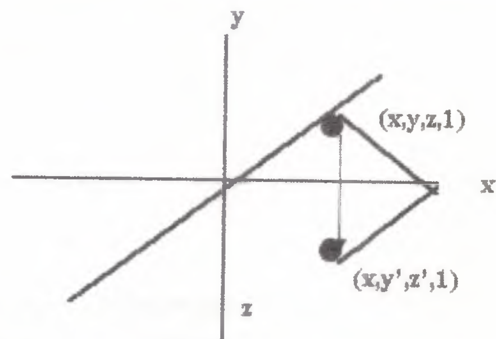


FIGURE 8.5.2
A point rotated about the x- axis has its x-coordinated



The matrix equation that represents this set of equations is

$$[x'\ y'\ z'\ 1] = (x\ y\ z\ 1)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The equations for a rotation about the y-axis (figure 8.5.3) are obtained by applying the permutation z replaces x, x replaces y, and y replaces z to the equations for a rotation about the z-axis:

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

The matrix equation for the rotation about the y-axis is

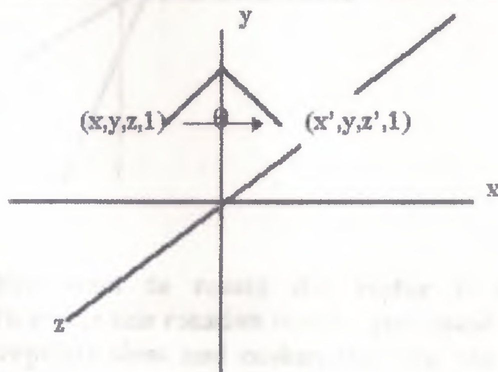
$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each of the rotations about one of the coordinate axes has the property that its inverse function is represented by the transpose of the matrix representing it.

Rotation about an arbitrary axis would be difficult if it were not broken down into a series of transformations.First, the axis of rotation is transformed to the z-axis.Next, the rotation is performed as a rotation about the z-axis. Last, the inverse of the transformation on the axis of rotation restores objects to their final rotated positions. To summarize, the transformation of the axis of rotation involves three steps:

FIGURE 8.5.3

A point rotated about the y-axis has its y-coordinate fixed.



1. **Translation of the axis of rotation so that it passes through the origin.**
2. **Rotation about the x-axis so that the axis of rotation is in the xz-plane.**
3. **Rotation about the y-axis so that the axis of rotation coincides with the z-axis**

The axis of rotation may be specified as the line passing through two points, say (x1, y1, z1) and (x2,y2,z2) (figure 5.4). The transformation represented by the matrix

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

translate the axis so that the point (x, , y, , z,) is transformed to the origin(figure 8.5.4)

The original points also define the vector along the axis of rotation

$$V = (x_2 , x_1 , y_2 - y_1 , z_2 - z_1) = (x ,y, z)$$

Following the translation, the axis of rotation passes through the origin and the point (x,y,z).Here you use the scalar and vector products to compute the sines and cosines in the rotation matrices, so it is useful to replace V with its unit vector u in the same direction.

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

FIGURE 8.5.4
The axis of rotation may be specified as the line passing through two points .

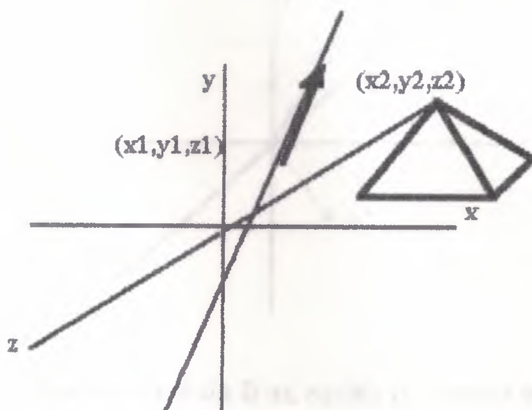
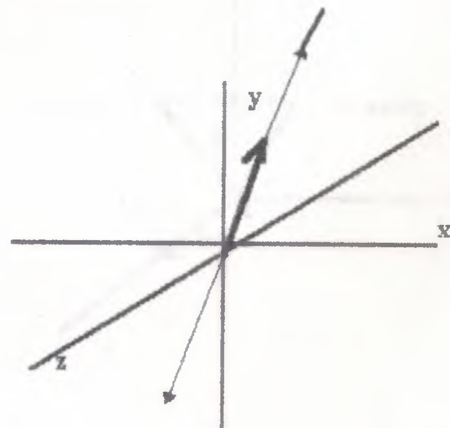


FIGURE 8.5.5
The first step in a rotation about an arbitrary axis is to translate the axis of rotation so it passes through the origin.



You want to rotate the vector u around the x-axis into the xz-plane (figure 8.5.6). To create this rotation matrix, you could compute the angle of rotation and then compute the appropriate sines and cosines.But you can use vector operations to compute the sine and cosine values directly. First, consider u' = (O,b,c), the projection of u into the yz-plane (figure 8.5.7). The rotation that takes u into the xz-plane also takes u' to the z-axis. Hence, if you could calculate the sine and cosine of the angles between u' and the z-axis, you could easily create the desired rotation matrix.

Let uX = (1,0,0), uy = (0,1,0), and u L = (0,0,1) be the unit vectors along the coordinate axes. Then the formula for the scalar product gives you the following formula for the cosine of the angle, θ , between u' and uz:

$$\cos\theta = \frac{u' \cdot uz}{|u'|\,|uz|} = \frac{c}{d}$$

where d = b2 + c2 is the length of u' .

The vector product gives you a formula for sin θ. The vector product of u' and u2 is

$$u' * u_z = u_x \; u' \; u_z \; \sin\theta$$
$$= u_x \, d \sin\theta$$

The vector uz is a unit vector, so its length is 1. The vector u' has length d; thus,using the determinate form of the vector product, you get

$$u' * u_z = (b,0,0) = b \, u_x$$

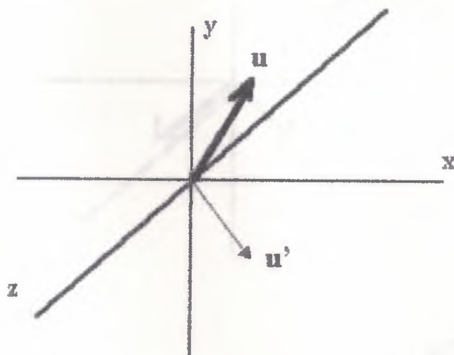


FIGURE 8.5.6
The second step in the general rotation is to rotate the axis of rotation about the x-axis until it is in the xz-plane.

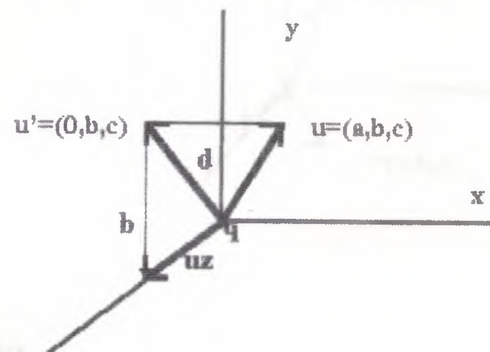


FIGURE 8.5.7
The entries in the rotation matrix may be from the coordinates' unit vectors.

The vector d sin θ ux equals the vector b ux, so

$$\sin\theta = b/d$$

You can substitute these values for sine and cosine directly into the matrix formulation for a rotation about the x-axis to get

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & b/d & 0 \\ 0 & -b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let u'' be the rotation of u onto the xz-plane (figure 8.5.8). The projection of u'' onto the z-axis is also the rotation of u' onto the z-axis. Because rotation does not change the length of a vector, the rotated form of u' is (O,O,d). The x-component of u is not altered by the rotation, so u'' = (a,0,d).

To rotate u around the y-axis until it coincides with the z-axis (figure 8.5.9), you repeat the operations used earlier to find the sine and cosine of the angle ϕ between u'' and the z-axis.Because the vector u'' is already in the xz-plane, it is unnecessary to use a projection of it in the computations.

First, use the scalar product to compute cos ϕ:

$$\cos\phi = \frac{u'' \cdot uz}{|u''|\,|uz|} = d$$

where the value d is computed directly because both u'' and uz are unit vectors.

The sine is computed by combining the two formulas for the vector product

$$u''*u_z = uy\,|u|\,|u_z|\,\sin\phi$$

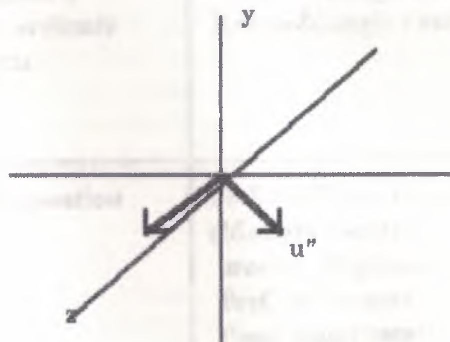


FIGURE 8.5.8
the vector u'' is the rotation of vector u into the xz-plane.

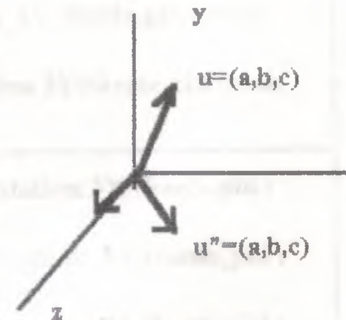


FIGURE 8.5.9
The vector u'' is rotated about the y-axis unit it coincides

and

$$u'' * uz = (-a)\ uy$$

As both u'' and uz have length 1,

$$\sin\Phi = -a$$

These values for the sine and cosine are substituted into the general matrix representation for a rotation about the y-axis to get

$$R_y = \begin{bmatrix} d & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, if the angle of the original rotation is ϕ, you rotate through this angle about the z-axis. This transformation is represented by

$$R\phi=\begin{bmatrix}\cos\phi & \sin\phi & 0 & 0\\ -\sin\phi & \cos\phi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

To get the matrix representation of rotation R(ϕ) about the arbitrary axis,you compute the matrix product

$$R(\phi) = T\, R_x\, R_y\, R\phi\, Ry^{-1} Rx^{-1}\, T^{-1}$$

PHIGS and GKS-3D provide functions for creating matrices to represent rotations about coordinate axes and the matrix products used to represent general rotations (figure 8.5.10).

FIGURE 8.5.10
PHIGS and GKS-3D provide a means to create matrices representing rotations about the coordinate axes, and a methot for composing these matrices with other transformation matrices to create the matrices representing general rotations.

Matrix Created	PHIGS	GKS-3D
rotations about a coordinate axis	RotateX(Angle : real); RotateY(Angle : real); RotateZ(Angle : real);	Create_Rotation_3X(theata,phi:real; m : matrix); Create Rotation_3Y(theata,phi : real; m : matrix); Create_Rotation 3Y(theata,phi :real; m : matrix);
compostion	SetLocalTransformation (Maxtrix : matrix 4by4; mode : (Replace, PreConcatenate PostConatenate))	Accumulate_Rotation 3X(theata,phi : real; m : matrix); Accumulate_Rotation_3Y(theata,phi : real; m : matrix); Accumulate_Rotation 3Z(theata,phi : real;m : matrix);

8.6 Reflections and Shears

Two simple transformations that have manyapplications are reflections and sheares.Conversion from a right-hand coordinate system toa left-hand coordinate system is a reflection through the xy-plane.A point with coordinates (x,y,z,1) in a right-hand system has coordinates (x,y,-z,I) in the corresponding left-hand system. To convert from the right-hand system to the left-hand system, one need only apply the reflection represented by

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

Similar matrices represent reflections through the other coordinate planes Reflections through an arbitrary plane are realized by applying a sequence of rotations and reflections Shears in three dimensions are similar to those in two dimensions because one of the coordinates fixed while the other coordinates have multiples of the fixed coordinate added to them. The following matrix represents a y-axis shear.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this transformation, the y-coordinate of a point is held constant while constants are added constants are added to the x- and z-coordinates.

8.7 Transformation of Coordinate Systems

Recall that for right-hand coordinate systems, one standard orientation uses the graphics screen as the xy-plane.This orientation results in the positive z-axis pointing out of the screen (figure8.7.I).If the left-hand system has the xy-plane coinciding with the screen then the positive z-axis points into the screen (figure 8.7.2). In either of these orientations, the viewer is looking directly down the z-axis.

While this orientation simplifies the display of an object, it does not always present the desired view.This can be a particularly vexing problem with animation. An example of this type of situation is camera (viewer) placement in space movies (Blinn 1988).A typical sequence in such movies shows a spaceship at point f orbiting planet at point a.Both the planet and ship are moving along different curve paths at different rates of speed. To create

FIGURE 8.7.1
The right-hand orientation of a coordinate system results in the positive z-axi.s pointing out of the screen.

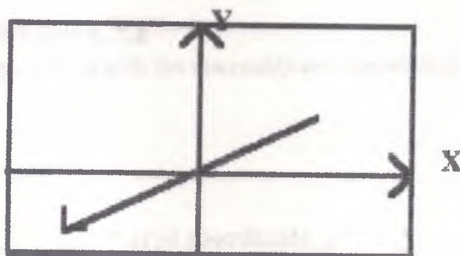
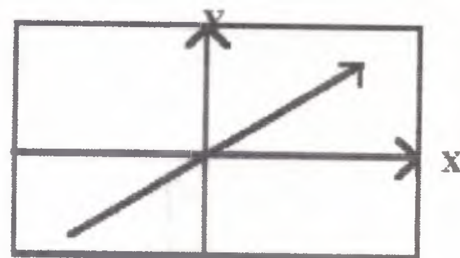


FIGURE 8.7.2
The left-hand orientation of a coordinate positive z-axis pointing into the screen.



realistic motion, the paths of both ship and planet are computed in world coordinates by means of Kepler's methods for computing orbital motion (figure 8.7.3). The camera is normally placed at point e, so both the object being tracked by the camera and the camera are on the z-axis. To facilitate such camera placement, all positions are converted to a new coordinate system called view coordinates.

Other considerations in this problem include the separation of the ship and planet on the screen , the distance from the camera to the object being tracked, and the need to track one or the other object.This brief discussion considers only the change of coordinates.

Suppose the camera is to track the planet. You then need to create a new coordinate system with the camera and the planet on the z-axis. In addition, you want to choose the coordinate system so that the ship will appear right side up.The up direction is defined by the vector U. To maintain these conditions as well as position the camera and image plane in the desired place, you translate the coordinate system so that a is at the origin. Then the vector T = (a - e), is rotated to lie along the z-axis (figures 8.7.4 and 8.7.5).

When this transformation is applied to U, the resulting vector should also point as close to up as possible;that is, the image of U should have x-component 0 (figure 8.7.6). If you

assume initially that both T and U are unit vectors and that M is the matrix representing the rotation, you have the following matrix equations:

$$TM = (0,0,1)$$

FIGURE 8.7.3

In order to create an image from a desired viewpoint, you must convert points given in the world coordinate system that are used to compute the motion of the spaceship and planets to coordinates determined by the new z-axis and the up vector u.

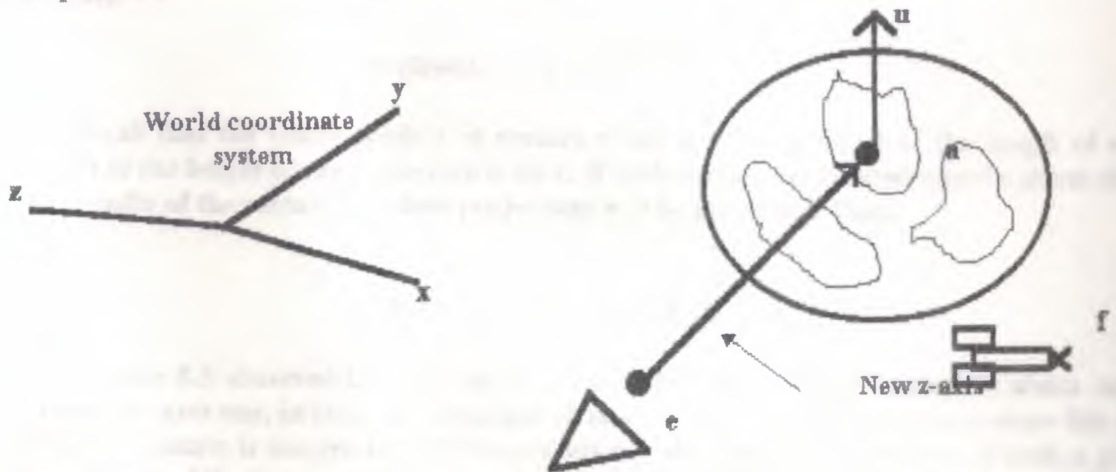


FIGURE 8.7.4

Figure 8.7.3 with the spaceship and planet removed

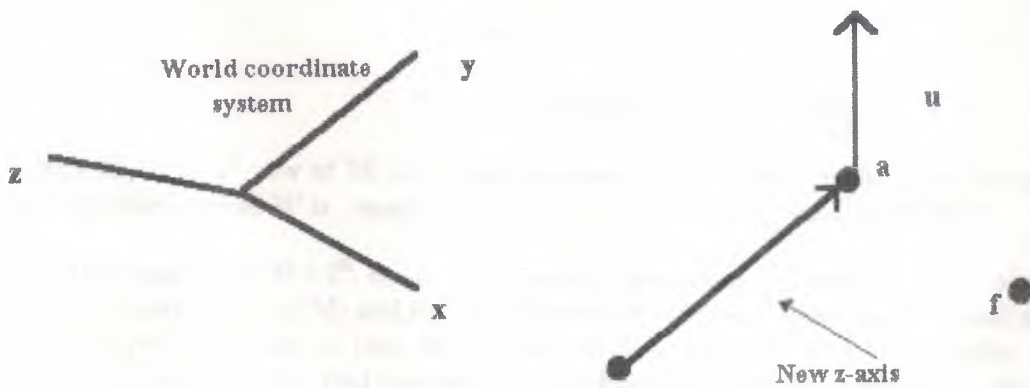


FIGURE 8.7.5
The first step in the transformation is to translate the point a to the origin

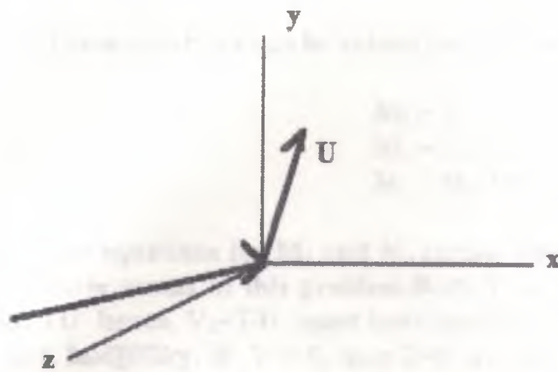
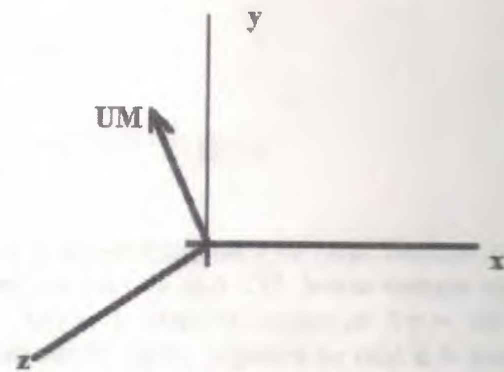


FIGURE 8.7.6
The vector T is rotated by a transformation with it coincides with the z-axis.



and

$$UM = (O, V_y, V_z)$$

The matrix M may be calculated by breaking the rotation process into a series of rotations about the x- and y-axes. Blinn (1988) applies the following vector and matrix arithmetic to compute M directly. Note that if unit vectors are rotated the resulting vectors are also unit vectors and both TM and UM have length l.It follows that V z = 1 - VIz, and because the vector U points in the positive

$$\text{y-direction } Vy = \sqrt{1 - V^2}$$

Recall that the scalar product of vectors v and w is the product of the length of v and the length of the length of the projection w on v. If both vectors are rotated equally about the origin, the lengths of the vectors and their projections will be preserved. Thus,

$$(0,0,1) \cdot (O, Vy, Vz) = T \cdot U$$

Section 8.5 observed that the inverse of a matrix representing a rotation about any of the coordinate axes was, in fact, the transpose of that matrix. It is easy to demonstrate this principle because a matrix is the product of three rotations about axes. The inverse of such a product is the product of the inverses in reverse order ($(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$)).The transpose of a product of matrices is the product of the transposes in reverse order ($C^tB^tA^t = (ABC)^t$).

Solving the original transformation equations using $M^{-1} = M^t$ you get

$$T = (0,0,I)M^t = M_3$$

$$U = (0, Vy, Vz)M^t = VyM_2 + VzM_3$$

where M_i is the ith row of M^t and therefore also the ith column of M. The first equation shows that the third row of M^t is equation shows that the third row of M^t is simply T.

The equation $M_tM = i^{th}$, the identity matrix, shows that the scalar product of the second row of M^t(second column of M) and the first column of M is zero. Thus, the first and second columns of M are perpendicular. In fact, each column of M is perpendicular to the other columns of M. This, along with the fact that you want a right-handed system, is sufficient to show that thé first

This, along with the fact that you want a right-handed system, is sufficient to show that thé first column of M, M1 is the cross-product of the second (M_2) and third (M_3) columns of M,that is, $M_1 = M_2 * M_3$.

These equations can be solved for the columns of M to give

$$M_3 = T$$
$$M_2 = (1/Vy)\ U - (Vz/Vy)\ T = [U - (U \cdot T)T]/Vy$$
$$M_1 = M_2 * M_3 = [U * T]/V_y$$

The equations for M_2 and M_1 cannot be used if Vy is either imaginary or zero. Neither of these cases accurs in this problem.Both T and U are unit vectors so that TU hence vectors so that TU hence, V_Z=T·U must have length less than or ual to 1. Thus, the value of $Vy =- Vz^2$ is not imaginary. If V = 0, then T=U .In most real problems,U canbe adjusted so that it is not T.You can accomplish the conversion from right to left handed coordinates by multiplying the z coordinate of each vector by -1 before transforming it in to view coordinates.

While placement of a movie camera may not be a common application of transformations solution to this problem is the first step in creating realistic.

REFERENCE

Intesrated Computer Graphics .... Bruce Mielke
Computer Graphics .... Donald Hearn / M.Pouline Baker