- Basic Concepts
- Two-Dimensional Affine Transformations
- Translations
- Scales
- Rotations
- Inverses of Basic Transformations
- Composite Transformations
- Other Transformations
- Simple Transformation Commands
- Window to Viewport Transformations
- Raster Methods for Transformations
- Three Dimensional Affine Transformations
- Graphics Libraries Transformation Commands

- Objects are defined by a set of vertices
- Objects are transformed by transforming their vertices
- Straight lines stay straight
- This implies that *linear* or *affine* transformations are used to transform objects
- Linear and affine transformations can be represented via matrices
- Later perspective projections will be introduced — these are non-linear transformations
- Non-linear maps are sometimes useful in graphics

- Translation (reposition)
- Scaling (reduce or enlarge)
- Rotation (re-orient)
- Reflection
- Shear
- These transformation are called *affine* and can be expressed as

$$\left[\begin{array}{cc} x' & y' \end{array}\right] = \left[\begin{array}{cc} ax+by+c & dx+ey+f \end{array}\right]$$

  where $[x'\ y']$ is the transformation of the point $[x\ y]$

- A *linear* map can be expresses as

$$\left[\begin{array}{cc} x' & y' \end{array}\right] = \left[\begin{array}{cc} ax+by & dx+ey \end{array}\right]$$

  where $[x'\ y']$ is the transformation of the point $[x\ y]$

- Straight line movement from one position to another
- Let $[x\ y]$ be input vertex, let T_x and T_y be shift in x and y directions, respectively, then

$$\begin{aligned} x' &= x + T_x \\ y' &= y + T_y \end{aligned}$$

is translated vertex

- Translate objects by adding the translation vector to the coordinates of each vertex
- A homogeneous coordinate is appended to each point

$$(x,\, y) \rightarrow (x,\, y,\, 1)$$

- Translation in matrix form

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

- Alter the size of an object
- Let $[x\,y]$ be input vertex, let S_x and S_y be scale in x and y directions, respectively, then

$$\begin{aligned} x' &= x \cdot S_x \\ y' &= y \cdot S_y \end{aligned}$$

  is the scaled vertex

- Scaling in matrix form

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

- Scale objects by multiplying the coordinates of each endpoint by the scaling factors

- Lengths and distances from the origin are scaled
- One point will remain fixed under a scaling
  - By default the *fixed point* is the origin $(0,\ 0)$
  - To select an arbitrary fixed point (x_F, y_F)
    1. Translate (x_F, y_F) to $(0,\ 0)$
    2. Scale by $(S_x,\ S_y)$
    3. Translate $(0,\ 0)$ to (x_F, y_F)
  - The result is

$$\begin{aligned} x' &= x_F + (x - x_F)S_x \\ y' &= y_F + (y - y_F)S_y \end{aligned}$$

- Transformation along circular paths
- Let $[x\ y]$ be the input vertex and let θ be rotation angle

$$
\begin{aligned}
x' &= x\cos\theta - y\sin\theta \\
y' &= y\cos\theta + x\sin\theta
\end{aligned}
$$

- Rotation in matrix form

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

- Rotate objects by rotating each endpoint
- By default the *pivot point* is the origin $(0,\ 0)$
- To select an arbitrary pivot point (x_R, y_R)
  1. Translate $(x_R,\ y_R)$ to $(0,\ 0)$
  2. Rotate by θ
  3. Translate $(0,\ 0)$ to $(x_R,\ y_R)$
- The result is

$$
\begin{aligned}
x' &= x_R + (x - x_R)\cos\theta - (y - y_R)\sin\theta \\
y' &= y_R + (y - y_R)\cos\theta + (x - x_R)\sin\theta
\end{aligned}
$$

- Translation

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

- Scaling

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

- Rotation

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

- A composite transformation matrix is the product of two or more individual transformation matrices
- Multiplying two matrices together is referred to as concatenation
- Two Successive Translations

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x_1} & T_{y_1} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x_2} & T_{y_2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x_1} + T_{x_2} & T_{y_1} + T_{y_2} & 1 \end{bmatrix}$$

- Scaling Relative to a Fixed Point

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_F & -y_F & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_F & y_F & 1 \end{bmatrix} =$$

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ (1-S_x)x_F & (1-S_y)y_F & 1 \end{bmatrix}$$

- Rotation About a Pivot Point

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

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

Composite Transformations

- Arbitrary Scaling Directions
  - Rotate so that the scaling axes coincide with the x and y axis
  - Scale
  - Rotate scaling axes back to their original positions
- Concatenation Properties
  - Matrix multiplication is associative
  - Matrix multiplication is <u>not</u> commutative
- A rotation followed by a scale will usually give a different result than the scale followed by the rotation

- Reflections
  - Reflection about x-axis

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

  - Reflect about arbitrary line
    * Translate line so it passes through the origin
    * Rotate line so it aligns with x-axis
    * Reflect
    * Undo the rotation and the translation
- Shears
  - Shear along x-axis

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

  - Shear useful for mapping viewing pyramid to standard cube in NDC space

- `Tran (tx, ty)` – replace matrix C on top of stack with
$$C \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix} C$$
- `Scale (sx, sy)` – replace matrix C on top of stack with
$$C \leftarrow \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} C$$
- `Rot (a)` – replace matrix C on top of stack with
$$C \leftarrow \begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix} C$$
- `Draw Object` – draw the object applying the matrix on the top of the stack to the vertices of the object

- A *window* is a rectangle in world coordinates (it defines the region of the world we can see)
- A *viewport* is a rectangle in screen coordinates (it defines where on the monitor the image will occur)
- The window to viewport map is a composition of a scale and translation
- Let x_{min}, x_{max}, y_{min}, y_{max} define the window rectangle
- Let u_{min}, u_{max}, v_{min}, v_{max} define the viewport rectangle
- The map

  $$u = \frac{u_{max} - u_{min}}{x_{max} - x_{min}}(x - x_{min}) + u_{min}$$

  describes how horizontal edges line up
- The map

  $$v = \frac{v_{max} - v_{min}}{y_{max} - y_{min}}(y - y_{min}) + v_{min}$$

  describes how vertical edges line up

- The window to viewport map written as a matrix transformation is

$$\begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \\ t_x & t_y \end{bmatrix}$$

where

$$s_x = \frac{u_{max} - u_{min}}{x_{max} - x_{min}},$$

$$s_y = \frac{v_{max} - v_{min}}{y_{max} - y_{min}},$$

$$t_x = -\frac{u_{max} - u_{min}}{x_{max} - x_{min}} x_{min} + u_{min},$$

$$t_y = -\frac{v_{max} - v_{min}}{y_{max} - y_{min}} y_{min} + v_{min},$$

- Some simple transformations can be carried out by manipulating the frame buffer contents (raster ops)
- Translation — block transfers (bitblts or pixblts)
  - copy a block from one area of the frame buffer to another area
  - fill the old area with background color
  - begin with an overlapped corner (if any)
- Rotations by multiples of 90 degrees
- Scaling by integer multiples
- Often a logical operation is applied to the source and destination
  - src $\vee$ dst
  - src $\wedge$ dst
  - src $\oplus$ dst
- Exclusive OR's are useful for "rubber banding" of objects

- Translation — straight line movement from one position to another
- Given input vertex $[x\ y\ z]$ and shifts T_x, T_y, T_z

$$\begin{aligned} x' &= x + T_x \\ y' &= y + T_y \\ z' &= z + T_z \end{aligned}$$

- Translate objects by adding the translation vector to the coordinates of each endpoint
- A homogeneous coordinate is appended to each point

$$(x,\ y,\ z) \rightarrow (x,\ y,\ z,\ 1)$$

- Translation in matrix form

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

- Alter the size of an object
- Given input vertex $[x\ y\ z]$ and scales S_x, S_y, S_z

$$\begin{aligned} x' &= x \cdot S_x \\ y' &= y \cdot S_y \\ z' &= z \cdot S_z \end{aligned}$$

- Scale objects by multiplying the coordinates of each endpoint by the scaling factors
- Scaling in matrix form

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

- Lengths and distances from the origin are scaled
- One point (x_F, y_F, z_F) can remain fixed
- A scale with fixed point $(x_F,\ y_F,\ z_F)$ is given by

$$\begin{aligned} x' &= x_F + (x - x_F)S_x \\ y' &= y_F + (y - y_F)S_y \\ z' &= z_F + (z - z_F)S_z \end{aligned}$$

- Right-handed coordinate system
- Designate an axis of rotation and an angle of rotation
- Convention: Positive rotation is counterclockwise
- Rotation about the principal axes
- z-axis rotation

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}$$

- x-axis rotation

$$\begin{aligned} x' &= x \\ y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \end{aligned}$$

- y-axis rotation

$$\begin{aligned} x' &= x\cos\theta + z\sin\theta \\ y' &= y \\ z' &= -x\sin\theta + z\cos\theta \end{aligned}$$

- Rotation about z axis

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

- Rotation about x axis

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

- Rotation about y axis

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

- Translation

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

- Scaling

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

- Rotation

$$R^{-1} = R^T$$

Rotation about an Axis Parallel to a Principal Axis

- Translate object so the rotation axis coincides with a principal axis
- Rotate through specified angle
- Translate object so the rotation angle returns to its original position

For example, suppose axis is parallel to z axis and intersects xy at (x_1, y_1).

$$T(-x_1, -y_1, 0)R(z, \theta)T(x_1, y_1, 0)$$

- Translate object so the rotation axis passes through the origin
- Rotate object so the rotation axis coincides with a principal axis (Usually requires 2 rotations)
- Rotate through specified angle
- Return rotation axis to its original orientation
- Translate rotation axis back to its original position

$$T \cdot R(x, \alpha) \cdot R(y, \beta) \cdot R(z, \theta) \cdot R^T(y, \beta) \cdot R^T(x, \alpha) \cdot T^{-1}$$

- Reflections
  - Reflect about xy plane

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

  - Reflect about yz plane

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

  - Reflect about zx plane

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

- Shears
  - x – Shear
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
  - y – Shear
$$\begin{bmatrix} 1 & c & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
  - z – Shear
$$\begin{bmatrix} 1 & 0 & e & 0 \\ 0 & 1 & f & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
  - General Shear
$$\begin{bmatrix} 1 & c & e & 0 \\ a & 1 & f & 0 \\ b & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Shear useful for mapping viewing pyramid to standard cube in NDC space.

Simple Building of Transformation Commands

- `Tran (tx, ty, tz)` – replace matrix C on top of stack with

$$C \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ tx & ty & tz & 1 \end{bmatrix} C$$

- `Scale (sx, sy, sz)` – replace matrix C on top of stack with

$$C \leftarrow \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} C$$

- `Rot (a, 1)` – replace matrix C on top of stack with

$$C \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & \sin a & 0 \\ 0 & -\sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} C$$

- `Rot (a, 2)` – replace matrix C on top of stack with

$$C \leftarrow \begin{bmatrix} \cos a & 0 & -\sin a & 0 \\ 0 & 1 & 0 & 0 \\ \sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} C$$

- `Rot (a, 3)` – replace matrix C on top of stack with

$$C \leftarrow \begin{bmatrix} \cos a & \sin a & 0 & 0 \\ -\sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} C$$

- `Draw Object` – draw the object applying the matrix on the top of the stack to the vertices of the object

- Graphics Standards such as GKS and PHIGS are libraries of function calls that implement transformations and provide other functionality
- Below is the functional description of PHIGS transformation commands
- `SET LOCAL TRANSFORMATION 3 (M, type)`
  - Input parameter `M` is a 4×4 real matrix
  - Input parameter `type` is a flag `PRECONCATENATE, POSTCONCATENATE, REPLACE`
- `TRANSLATE 3 (Vec, err, M)`
  - Input parameter `Vec` is a 3D translation real vector
  - Output parameter `err` is an integer error flag
  - Output parameter `M` is a 4×4 real matrix

- `SCALE 3 (sx, sy, sz, err, M)`
  - Input parameters `sx`, `sy`, `sz` are 3 real scale factors
  - Output parameter `err` is an integer error flag
  - Output parameter `M` is a 4×4 real matrix
- `ROTATE X (a, err, M)`
- `ROTATE Y (a, err, M)`
- `ROTATE Z (a, err, M)`
  - Input parameter `a` is a real angle in radians
  - Output parameter `err` is an integer error flag
  - Output parameter `M` is a 4×4 real matrix

- `BUILD TRANSFORMATION MATRIX 3 (F, V, ax, ay, az, sx, sy, sz, err, M)`
  - Input parameter `F` is a real 3D fixed point
  - Input parameter `V` is a real 3D shift vector
  - Input parameter `ax` is a real angle in radians
  - Input parameter `ay` is a real angle in radians
  - Input parameter `az` is a real angle in radians
  - Input parameters `sx`, `sy`, `sz` are 3 real scale factors
  - Output parameter `err` is an integer error flag
  - Output parameter `M` is a 4×4 real matrix
  - The composite matrix `M` $= T_{fp}^{-1} S R_x R_y R_z T_{fp} T_{sh}$ where
    * T_{sh} is the translation by the shift vector
    * T_{fp} is the translation by the fixed point
    * T_{fp}^{-1} is the inverse translation by the fixed point
    * R_x, R_y, R_z are rotations about x, y, z
    * S is the scale matrix

- `COMPOSE MATRIX 3 (M1, M2, err, M)`
  - Input parameter `M1` is a 4×4 real matrix
  - Input parameter `M2` is a 4×4 real matrix
  - Output parameter `err` is an integer error flag
  - Output parameter `M = M2 M1` is a 4×4 real matrix
- `COMPOSE TRANSFORMATION MATRIX 3 (M1, F, V, ax, ay, az, sx, sy, sz, err, M)`
  - Input parameter `M1` is a 4×4 real matrix
  - Input parameter `F` is a real 3D fixed point
  - Input parameter `V` is a real 3D shift vector
  - Input parameter `ax` is a real angle in radians
  - Input parameter `ay` is a real angle in radians
  - Input parameter `az` is a real angle in radians
  - Input parameters `sx`, `sy`, `sz` are 3 real scale factors
  - Output parameter `err` is an integer error flag
  - Output parameter `M` is a 4×4 real matrix
- *Bindings* to C, Ada, Fortran, etc., define how the calls are made in various programming languages

Problems

1. Define a trapezoid R, square S and triangle T:

$$\begin{aligned} R &= [(-1, 0), (1, 0), (2, 1), (-2, 1)] \\ S &= [(0, 0), (1, 0), (1, 1), (0, 1)] \\ T &= [(-1, 0), (1, 0), (0, 1)] \end{aligned}$$

   1. Translate the each object (R, S, T) by 3 in x and 5 in y.
   2. Scale the each object (R, S, T) by 2 in x and 4 in y.
   3. Rotate each object (R, S, T) by $45°$.
   4. Rotate each object (R, S, T) by $-30°$.
   5. Reflect each object (R, S, T) about the line $x = 1$.
   6. Scale the each object (R, S, T) by 2 in x and 4 in y with fixed point $(1, 1)$.
   7. Rotate each object (R, S, T) by $45°$ with pivot $(1, 1)$.

2. Represent a rectangle with vertices

$$A = (2, 4),\ B = (8, 4),\ C = (8, 6),\ D = (2, 6)$$

   as a 3×4 matrix.

   1. Translate the rectangle above 1 by 5 and -3 in the x and y directions, respectively. Express your answer as a 3×4 matrix, and show the translation operator used in the transformation.
   2. Scale the original rectangle by 2 and 3 in the x and y directions, respectively. Express your answer as a 3×4 matrix, and show the scaling operator used in the transformation.
   3. Rotate the original rectangle around the center point $(5, 5)$ by 30 degrees. Express your answer as a 3×4 matrix, and show the transformation operations used.

3. A window is a rectangular region in world coordinates that maps to a rectangular viewport of the screen of a display device. Determine a matrix that maps a window defined by

$$-100 \leq x_w \leq 200, \quad 0 \leq y_w \leq 50$$

onto a viewport defined by

$$0.0 \leq x_v \leq 0.6, \quad 0.1 \leq y_v \leq 0.5.$$

4. What is the inverse of the matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}?$$

5. What is the inverse of the matrix

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}?$$

6. What is the inverse of the matrix

$$\begin{bmatrix} 0.5 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}?$$

7. Find a matrix that transforms triangle T into the "right" triangle

$$T' = [(0,\, 0),\, (1,\, 0),\, (1,\, 1)]$$

8. Show how to scale an object by S_x, S_y in x, y about the line $x + y = 0$.
9. Show that translations and scales do not commute.
10. Show that the (non-linear) shear

$$\begin{bmatrix} \frac{1}{1+y} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

maps the trapezoid R into a square.

11. Why is the above transformation called *non-linear*?

12. B. L. Coder implemented nested local transformations in a graphics package by simply placing all transformations on a stack as they were defined. When the `Draw` command was issued, B. L.'s code applied each matrix on the stack, one at a time to the object's vertices. What do you think of B. L.'s practice?

13. A particular graphics library supports a function `set_transformation(sx, sy, a, tx, ty)` that scales (by `sx, sy` about $(0, 0)$), rotates (by angle `a` about $(0, 0)$) and translates (by `tx, ty`) *in order*. Show how to define these parameters to scale and rotate about a point (x, y) other than the origin.

14. Suppose you had a display device with a resolution of 1152×900 and you wanted to map the unit square in NDC space onto a rectangular portion of this display starting at lower left pixel $(100, 200)$ and extending to upper right pixel $(650, 800)$. Carefully describe the workstation transformation which performs this map from NDC space to DC space.

15. Define a trapezoid R, square S and triangle T:

$$\begin{aligned} R &= [(-1,\, 0,\, 0),\, (1,\, 0,\, 0),\, (2,\, 1,\, 0),\, (-2,\, 1,\, 0)] \\ S &= [(0,\, 0,\, 1),\, (1,\, 0,\, 1),\, (1,\, 1,\, 1),\, (0,\, 1,\, 1)] \\ T &= [(-1,\, 0,\, -1),\, (1,\, 0,\, -1),\, (0,\, 1,\, -1)] \end{aligned}$$

    1. Translate the each object $(R,\, S,\, T)$ by 3 in x, 5 in y and -4 in z.
    2. Scale the each object $(R,\, S,\, T)$ by 2 in x, 4 in y and 3 in z.
    3. Rotate each object $(R,\, S,\, T)$ by $45°$ about the x, y, and z axes.
    4. Rotate each object $(R,\, S,\, T)$ by $-30°$ about the x, y, and z axes.
    5. Reflect each object $(R,\, S,\, T)$ about the plane $z = 1$.
    6. Scale the each object $(R,\, S,\, T)$ by 2 in x, 4 in y and 3 in z with fixed point $(1,\, 1)$.
    7. Rotate each object $(R,\, S,\, T)$ by $45°$ about the line $x = t,\, y = t,\, z = t, \quad -\infty < t < \infty$

16. Suppose you wanted to scale a 3D object about the fixed point $(3,\, -2,\, 5)$ by scale factors $5,\, 7,\, 9$ in x, y, and z respectively. What 4×4 matrix would perform this transformation?

Problems

17. Suppose you want to rotate an object about the y axis (in a 3D coordinate system) by $\pi/6$ radians. Find the transformation matrix.

18. Express the line passing through $(3,\ -4,\ 5)$ in the direction of the vector $\langle 2,\ 6,\ -7\rangle$ in parametric form.

19. What matrix will rotate an object by $60°$ about the line $x = t,\ y = t,\ z = t,\ -\infty < t < \infty$?

20. What is the inverse of the matrix

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

21. What is the inverse of the matrix

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}?$$

22. What is the inverse of the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & -\sqrt{3}/2 & 0 \\ 0 & \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}?$$

23. Find a non-linear shear that maps the frustrum with front face

$$[(-1,\ -1,\ 0),\ (1,\ -1,\ 0),\ (1,\ 1,\ 0),\ (-1,\ 1,\ 0)]$$

and back face

$$[(-2,\ -2,\ 1),\ (2,\ -2,\ 1),\ (2,\ 2,\ 1),\ (-2,\ 2,\ 1)]$$

into the unit cube with front face

$$[(-1,\ -1,\ 0),\ (1,\ -1,\ 0),\ (1,\ 1,\ 0),\ (-1,\ 1,\ 0)]$$

and back face

$$[(-1,\ -1,\ 1),\ (1,\ -1,\ 1),\ (1,\ 1,\ 1),\ (-1,\ 1,\ 1)]$$

24. Show how to construct a mapping that (1) moves the origin to $(1, 1, 1)$, (2) aligns the z axis with the vector $\langle 1, 1, 1\rangle$, and (3) aligns the y axis with the vector $\langle 1, 0, -1\rangle$.