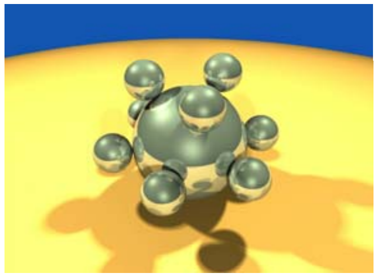
Computer Graphics using OpenGL, 3rd Edition

F. S. Hill, Jr. and S. Kelley



Chapter 5.1-2

Transformations of Objects

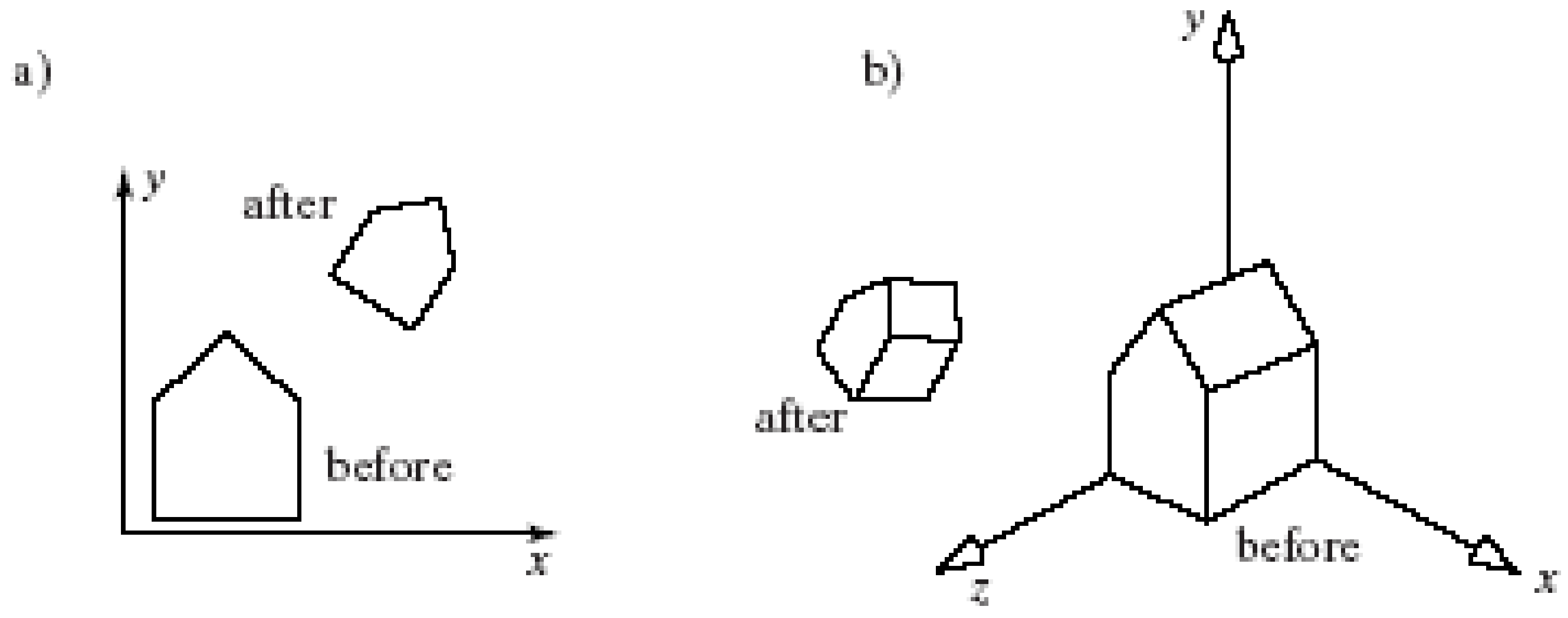
S. M. Lea
University of North Carolina at Greensboro

Transformations

- We used the window to viewport transformation to scale and translate objects in the world window to their size and position in the viewport.
- We want to build on this idea, and gain more flexible control over the size, orientation, and position of objects of interest.
- To do so, we will use the powerful **affine transformation**.

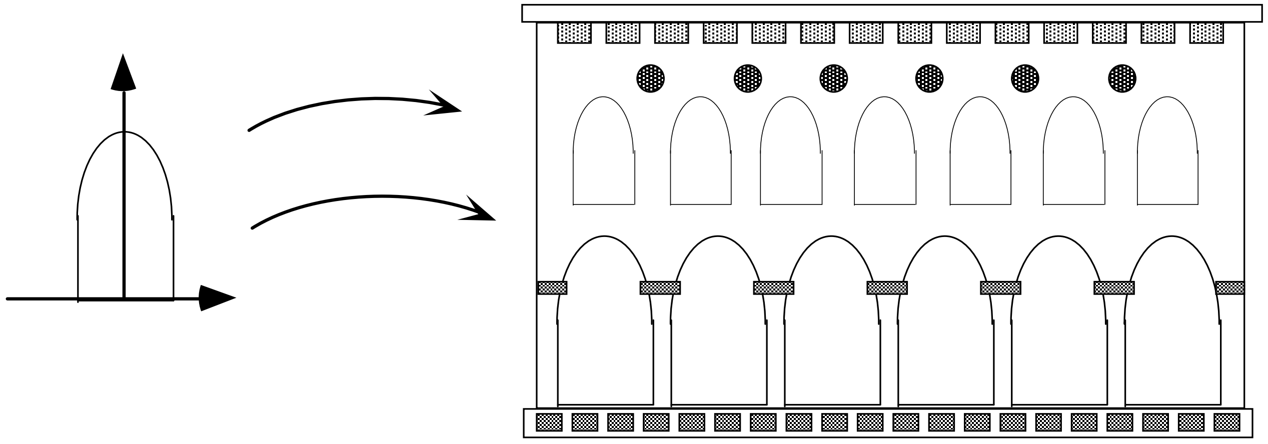
Example of Affine Transformations

- The house has been scaled, rotated and translated, in both 2D and 3D.



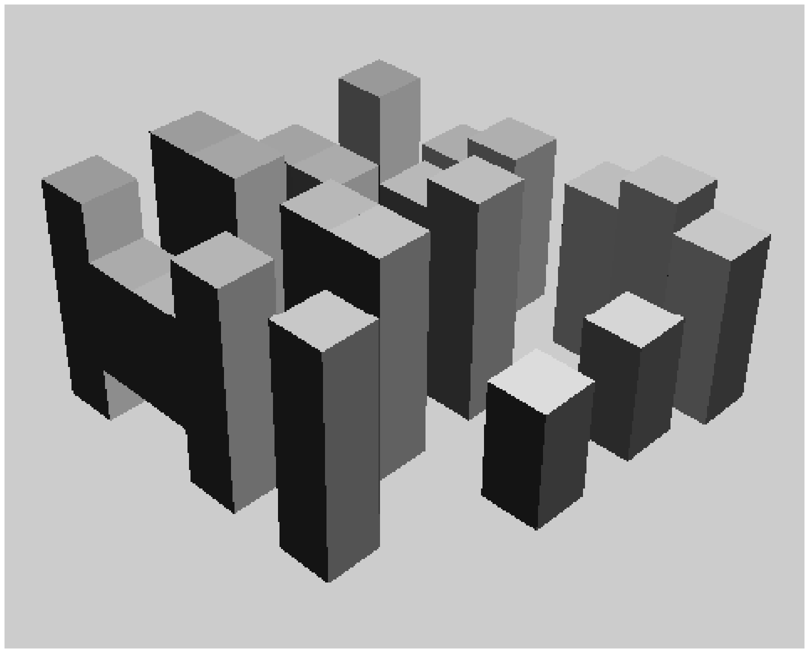
Using Transformations

- The arch is designed in its own coordinate system.
- The scene is drawn by placing a number of instances of the arch at different places and with different sizes.



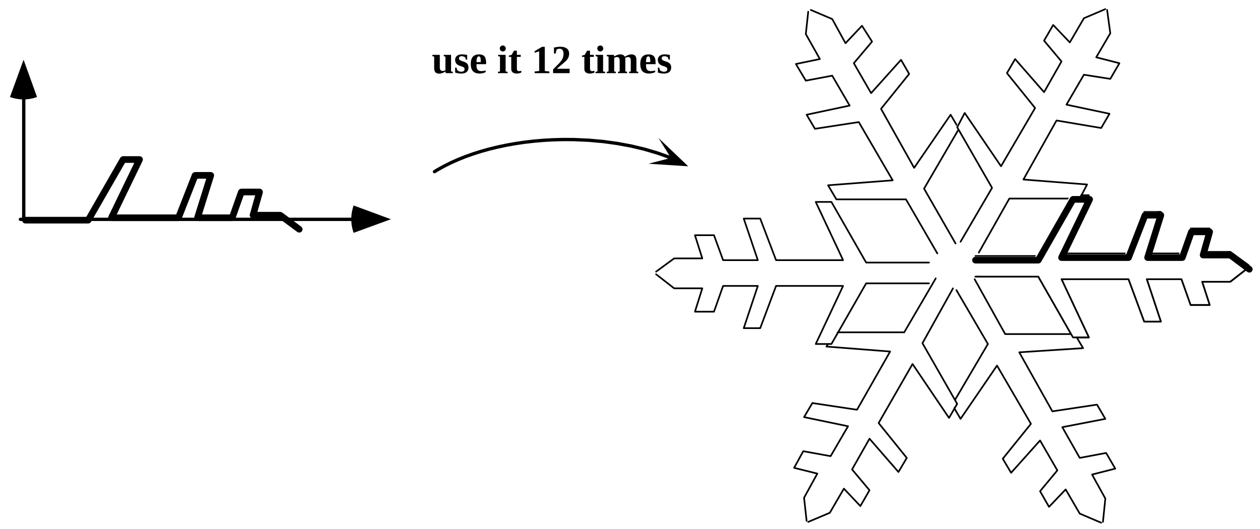
Using Transformations (2)

- In 3D, many cubes make a city.



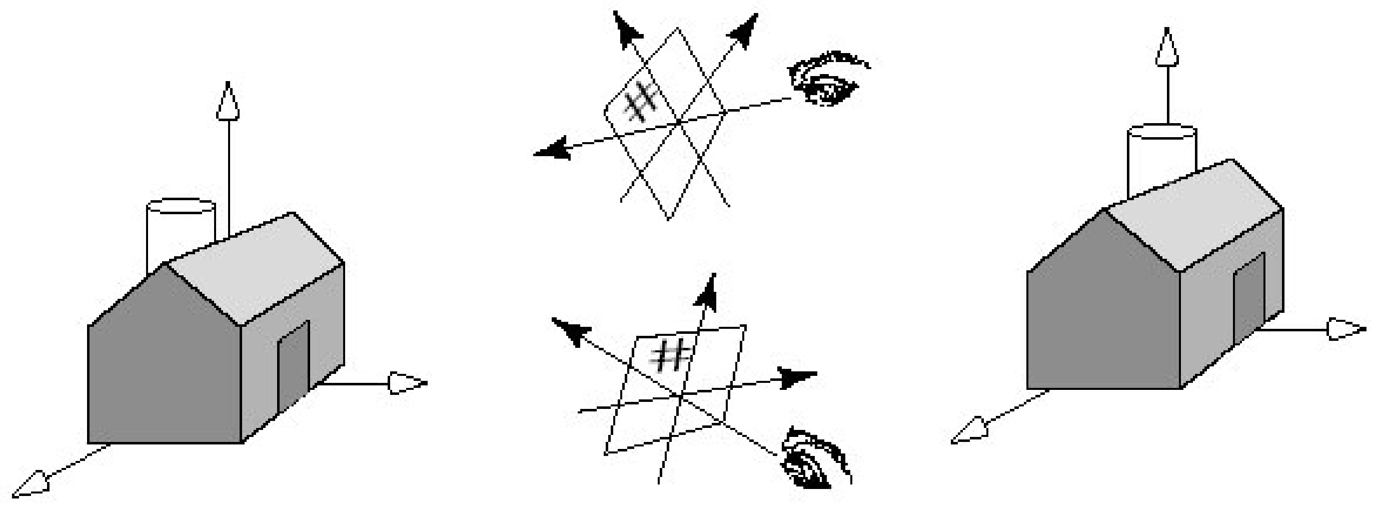
Using Transformations (3)

- The snowflake exhibits symmetries.
- We design a single **motif** and draw the whole shape using appropriate reflections, rotations, and translations of the motif.



Using Transformations (4)

- A designer may want to view an object from different vantage points.
- Positioning and reorienting a camera can be carried out through the use of 3D affine transformations.

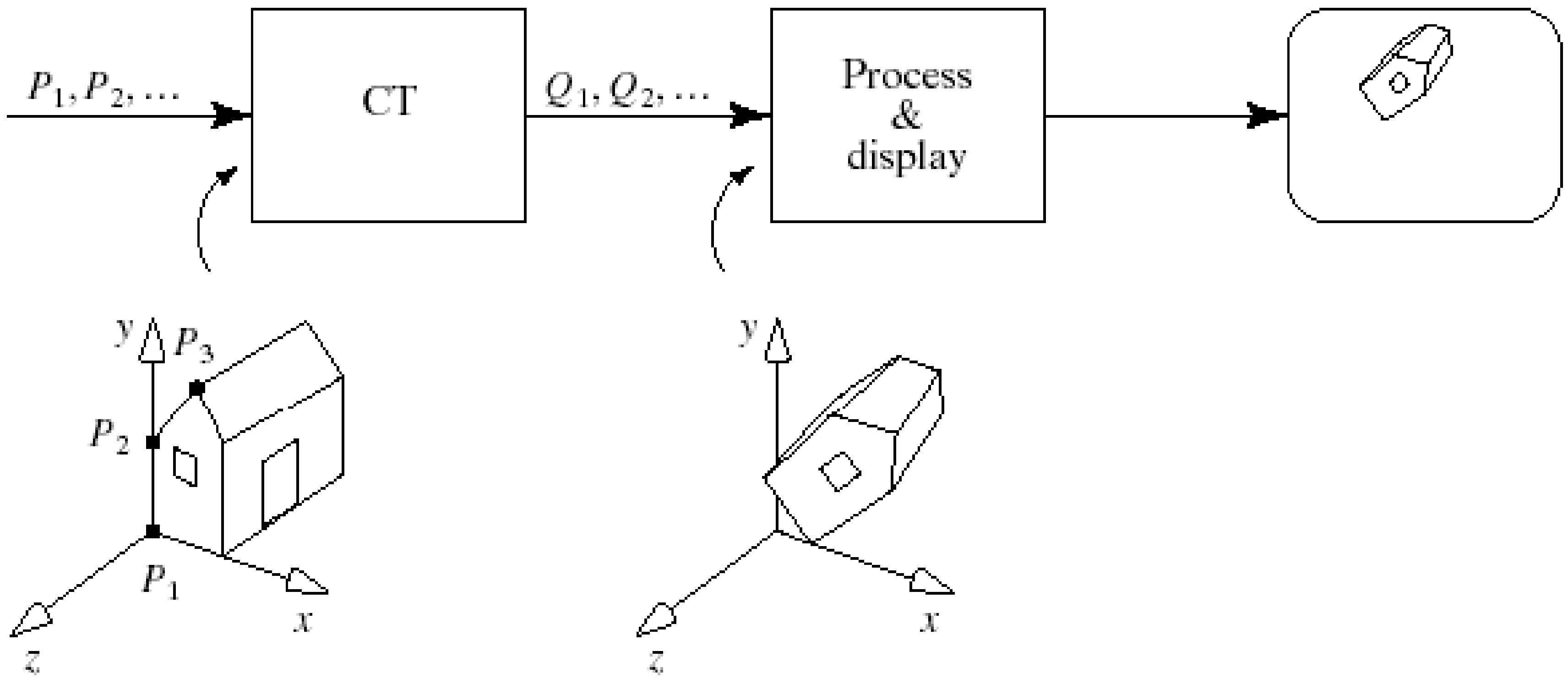


Using Transformations (5)

- In a computer animation, objects move.
- We make them move by translating and rotating their local coordinate systems as the animation proceeds.
- A number of graphics platforms, including OpenGL, provide a graphics pipeline: a sequence of operations which are applied to all points that are sent through it.
- A drawing is produced by processing each point.

The OpenGL Graphics Pipeline

- This version is simplified.



Graphics Pipeline (2)

- An application sends the pipeline a sequence of points P_1, P_2, ... using commands such as:

```
glBegin(GL_LINES);
   glVertex3f(...); // send P1 through the pipeline
   glVertex3f(...); // send P2 through the pipeline
   ...
glEnd();
```

- These points first encounter a transformation called **the current transformation** (CT), which alters their values into a different set of points, say Q_1, Q_2, Q_3.

Graphics Pipeline (3)

- Just as the original points P_i describe some geometric object, the points Q_i describe the transformed version of the same object.
- These points are then sent through additional steps, and ultimately are used to draw the final image on the display.

Graphics Pipeline (4)

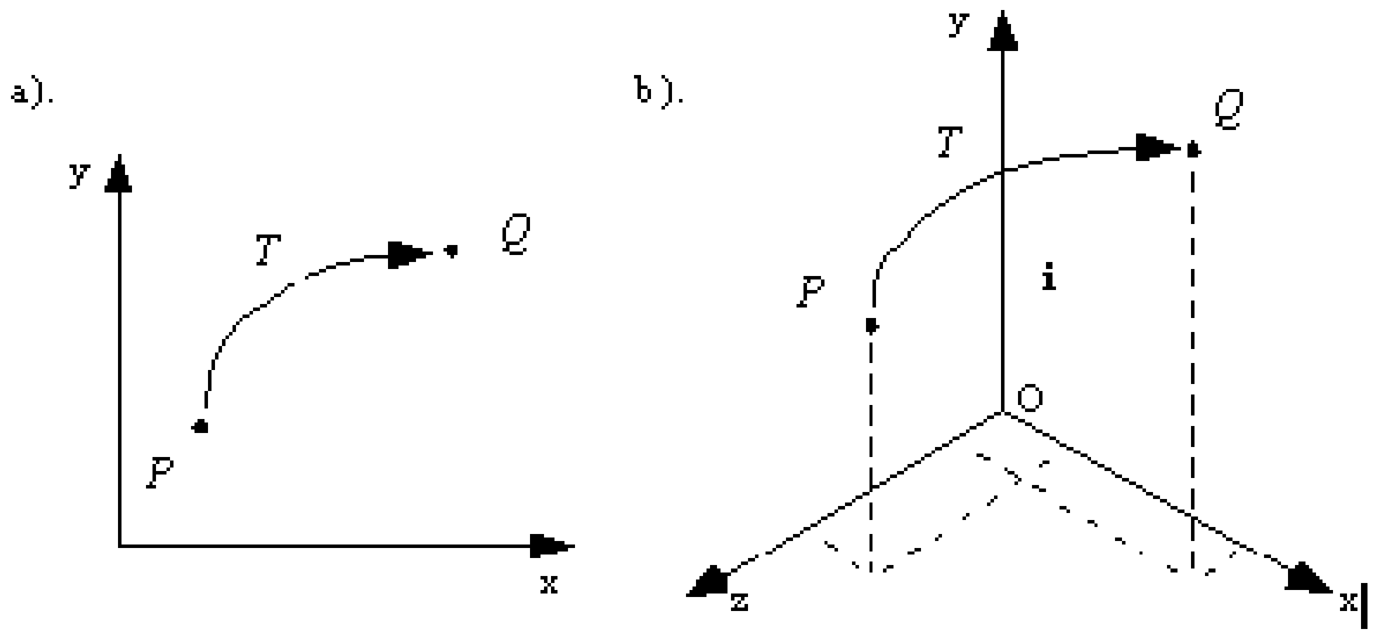
- Prior to OpenGL 2.0 the pipeline was of *fixed-functionality:* each stage had to perform a specific operation in a particular manner.
- With OpenGL 2.0 and the Shading Language (GLSL), the application programmer could not only change the order in which some operations were performed, but in addition could make the operations **programmable**.
- This allows hardware and software developers to take advantage of new algorithms and rendering techniques and still comply with OpenGL version 2.0.

Transformations

- Transformations change 2D or 3D points and vectors, or change coordinate systems.
  - An object transformation alters the coordinates of each point on the object according to the same rule, leaving the underlying coordinate system fixed.
  - A coordinate transformation defines a new coordinate system in terms of the old one, then represents all of the object’s points in this new system.
- Object transformations are easier to understand, so we will do them first.

Transformations (2)

- A (2D or 3D) transformation *T*( ) alters each point, *P* into a new point, *Q*, using a specific formula or algorithm: *Q*= *T*(*P*).



Transformations (3)

- An arbitrary point *P* in the plane is **mapped** to *Q*.
- *Q* is the **image** of *P* under the mapping *T*.
- We transform an object by transforming each of its points, using the *same* function *T*() for each point.
- The **image** of line *L* under *T*, for instance, consists of the images of *all* the individual points of L.

Transformations (4)

- Most mappings of interest are continuous, so the image of a straight line is still a connected curve of some shape, although it’s not necessarily a straight line.
- Affine transformations, however, *do* preserve lines: the image under *T* of a straight line is also a straight line.

Transformations (5)

- We use an explicit coordinate frame when performing transformations.
- A coordinate frame consists of a point $\mathcal{O}$, called the **origin**, and some mutually perpendicular vectors (called **i** and **j** in the 2D case; **i, j,** and **k** in the 3D case) that serve as the axes of the coordinate frame.
- In 2D,

$$\tilde{P} = \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}, \tilde{Q} = \begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix}$$

Transformations (6)

- Recall that this means that point $\mathscr{P}$ is at location $= \mathscr{P}_x\,\mathbf{i} + \mathscr{P}_y\,\mathbf{j} + \mathscr{O}$, and similarly for $\mathscr{Q}$.
- $\mathscr{P}_x$ and $\mathscr{P}_y$ are the coordinates of $\mathscr{P}$.
- To get from the origin to point $\mathscr{P}$, move amount $\mathscr{P}_x$ along axis **i** and amount $\mathscr{P}_y$ along axis **j**.

Transformations (7)

- Suppose that transformation *T* operates on any point $\mathscr{P}$ to produce point $\mathscr{Q}$:
- $\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = T(\begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix})$ or $\mathscr{Q} = T(\mathscr{P})$.
- T may be any transformation: e.g.,

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(P_x)e^{-P_x} \\ \dfrac{\ln(P_y)}{1+P_x^2} \\ 1 \end{pmatrix}$$

Transformations (8)

- To make **affine** transformations we restrict ourselves to much simpler families of functions, those that are *linear* in P_x and P_y.
- Affine transformations make it easy to scale, rotate, and reposition figures.
- Successive affine transformations can be combined into a single overall affine transformation.

Affine Transformations

- Affine transformations have a compact matrix representation.
- The matrix associated with an affine transformation operating on 2D vectors or points must be a three-by-three matrix.
  - This is a direct consequence of representing the vectors and points in homogeneous coordinates.

Affine Transformations (2)

- Affine transformations have a simple form.
- Because the coordinates of $\mathcal{Q}$ are *linear* combinations of those of $\mathcal{P}$, the transformed point may be written in the form:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11}P_x + m_{12}P_y + m_{13} \\ m_{21}P_x + m_{22}P_y + m_{23} \\ 1 \end{pmatrix}$$

Affine Transformations (3)

- There are six given constants: m_{11}, m_{12}, etc.
- The coordinate Q_x consists of portions of both P_x and P_y, and so does Q_y.
- This *combination* between the *x*- and *y*-components also gives rise to rotations and shears.

Affine Transformations (4)

- Matrix form of the affine transformation in 2D:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

- For a 2D affine transformation the third row of the matrix is always (0, 0, 1).

Affine Transformations (5)

- Some people prefer to use row matrices to represent points and vectors rather than column matrices: e.g., $P = (P_x, P_y, 1)$
- In this case, the *P* vector must *pre-multiply* the matrix, and the transpose of the matrix must be used: $Q = P\,M^T$.

$$M^T = \begin{pmatrix} m_{11} & m_{21} & 0 \\ m_{12} & m_{22} & 0 \\ m_{13} & m_{23} & 1 \end{pmatrix}$$

Affine Transformations (6)

- Vectors can be transformed as well as points.
- If a 2D vector **v** has coordinates V_x and V_y then its coordinate frame representation is a column vector with third component 0.

Affine Transformations (7)

- When vector **V** is transformed by the same affine transformation as point P, the result is $$\begin{pmatrix} W_x \\ W_y \\ 0 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ 0 \end{pmatrix}$$
- Important: to transform a point *P* into a point *Q*, *post-multiply M* by *P*: Q = M P.

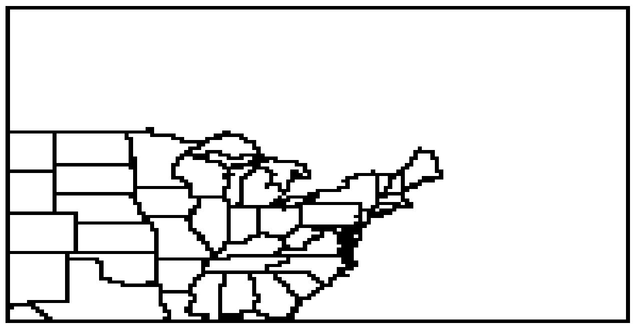
Affine Transformations (8)

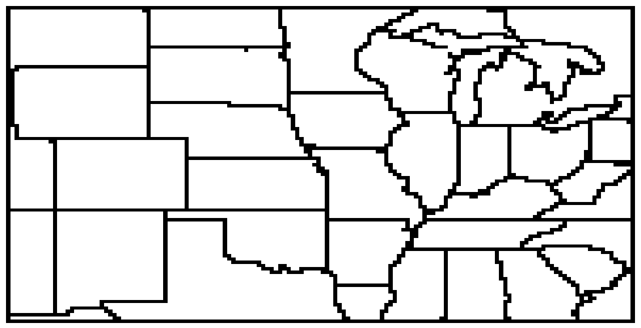
- Example: find the image Q of point P = (1, 2, 1) using the affine transformation

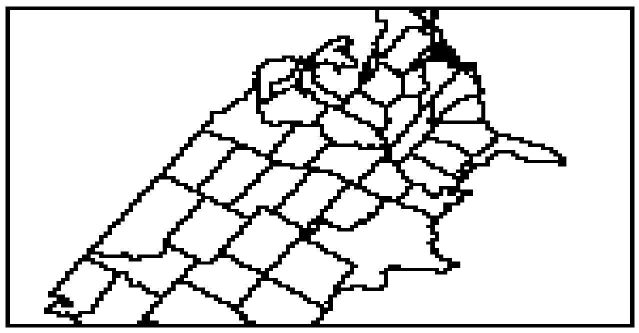
$$M = \begin{pmatrix} 3 & 0 & 5 \\ -2 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}; Q = \begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 5 \\ -2 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

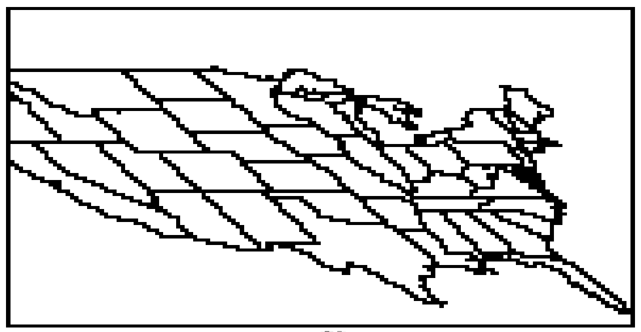
Geometric Effects of Affine Transformations

- Combinations of four elementary transformations: (a) a translation, (b) a scaling, (c) a rotation, and (d) a shear (all shown below).


a)


b)


c)


d)

Translations

- The amount *P* is translated does not depend on P’s position.
- It is meaningless to translate vectors.
- To translate a point P by a in the x direction and b in the y direction use the matrix:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix} = \begin{pmatrix} Q_x + a \\ Q_y + b \\ 1 \end{pmatrix}$$

- Only using homogeneous coordinates allow us to include translation as an affine transformation.

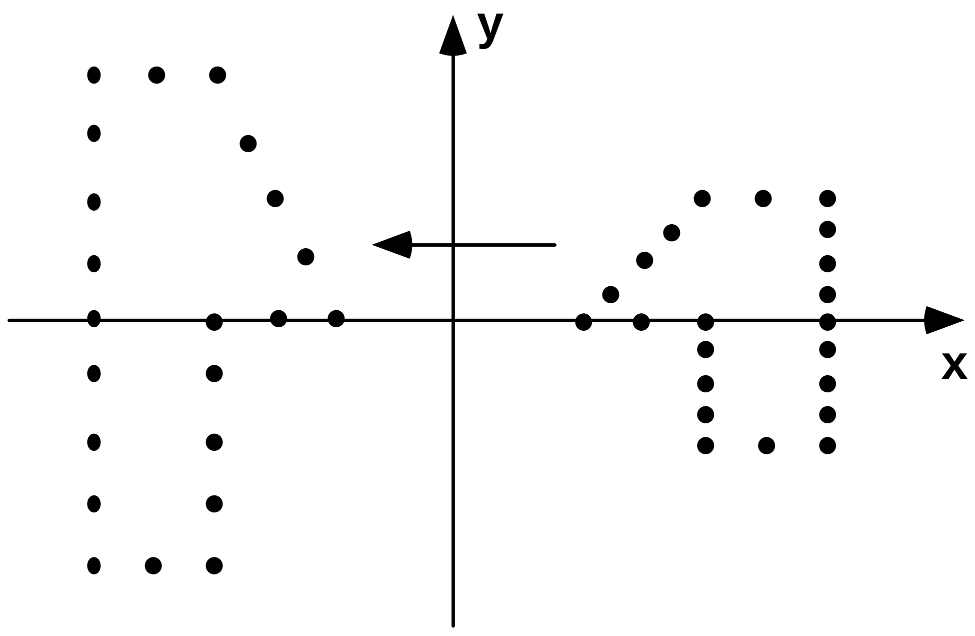
Scaling

- Scaling is about the origin. If $S_x = S_y$ the scaling is uniform; otherwise it distorts the image.
- If S_x or $S_y < 0$, the image is reflected across the x or y axis.
- The matrix form is

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

Example of Scaling

- The scaling (*S*x, *S*y) = (-1, 2) is applied to a collection of points. Each point is both reflected about the *y*-axis and scaled by 2 in the *y*-direction.



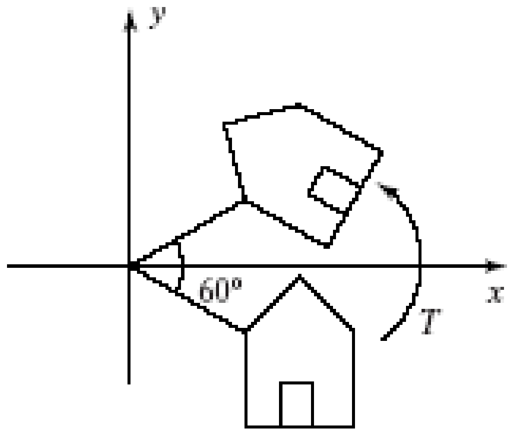
Types of Scaling

- Pure reflections, for which each of the scale factors is +1 or -1.
- A **uniform scaling**, or a magnification about the origin: $S_x = S_y$, magnification $|S|$.
  - Reflection also occurs if S_x or S_y is negative.
  - If $|S| < 1$, the points will be moved closer to the origin, producing a reduced image.
- If the scale factors are not the same, the scaling is called a **differential scaling**.

Rotation

- Counterclockwise around origin by angle θ:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$



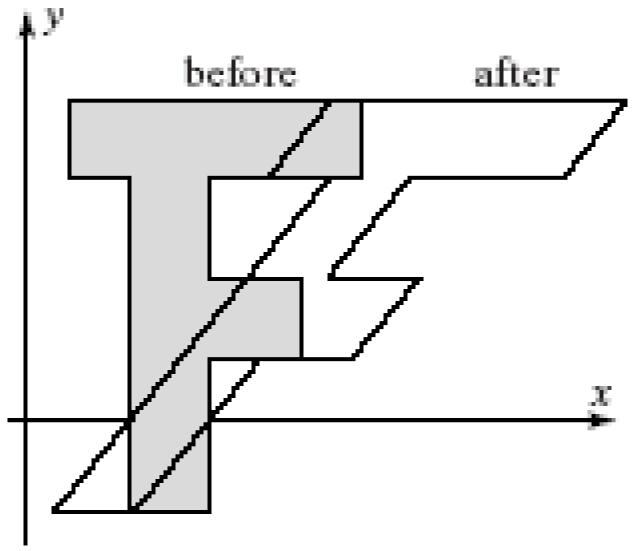
Deriving the Rotation Matrix

- P is at distance R from the origin, at angle Φ; then $P = (R\ cos(\Phi), R\ sin(\Phi))$.
- Q must be at the same distance as P, and at angle $\theta + \Phi$: $Q = (\mathrm{R} \cos(\theta + \Phi), \mathrm{R} \sin(\theta + \Phi))$.
- $cos(\theta + \Phi) = cos(\theta)\ cos(\Phi) - sin(\theta)\ sin(\Phi)$; $sin(\theta + \Phi) = sin(\theta)\ cos(\Phi) + cos(\theta)\ sin(\Phi)$.
- Use $P_x = R\ cos(\Phi)$ and $P_y = R\ sin(\Phi)$.

Shear

- Shear H about origin: x depends linearly on y in the figure.
- Shear along x: $h \neq 0$, and P_x depends on P_y (for example, *italic* letters).
- Shear along y: $g \neq 0$, and P_y depends on P_x.

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & h & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$



Inverses of Affine Transformations

- det(M) = m_{11}*m_{22} - m_{21}*m_{12}⑤ 0 means that the inverse of a transformation exists.
  - That is, the transformation can be "undone“.
- $M\,M^{-1} = M^{-1}M = I$, the identity matrix (ones down the major diagonal and zeroes elsewhere).

Inverse Translation and Scaling

- Inverse of translation T^{-1}:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

- Inverse of scaling S^{-1}:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

Inverse Rotation and Shear

- Inverse of rotation $R^{-1} = R(-\theta)$:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

- Inverse of shear H^{-1}: generally h=0 or g=0.

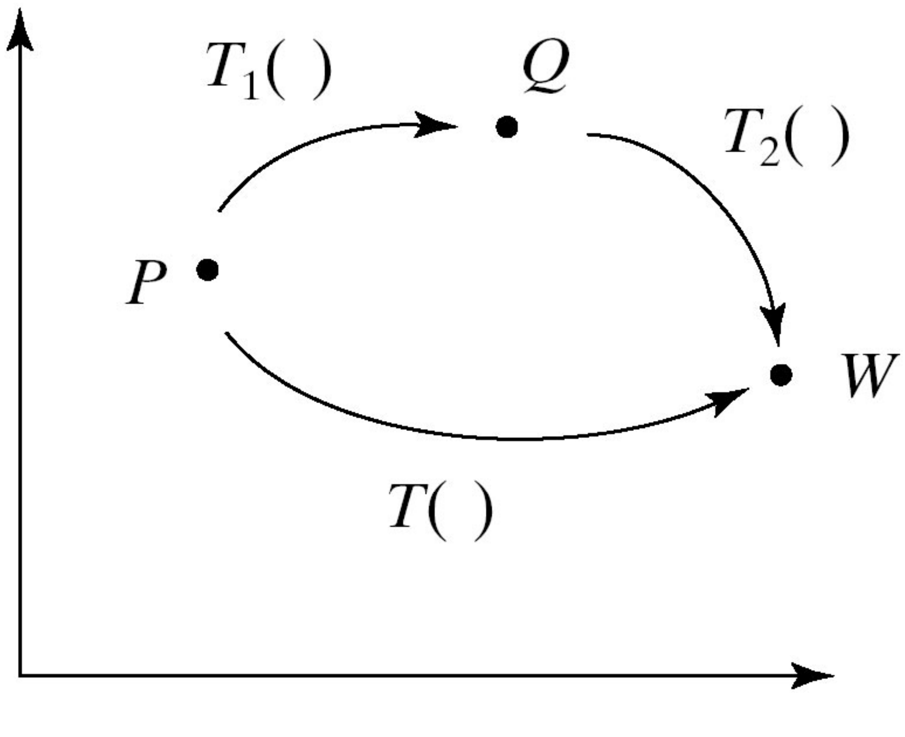
$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -h & 0 \\ -g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix} \frac{1}{1-gh}$$

Composing Affine Transformations

- Usually, we want to apply several affine transformations in a particular order to the figures in a scene: for example,
  - translate by (3, - 4)
  - then rotate by 30^o
  - then scale by (2, - 1) and so on.
- Applying successive affine transformations is called **composing** affine transformations.

Composing Affine Transformations (2)

- $T_1(\ )$ maps P into Q, and $T_2(\ )$ maps Q into point W. Is $W = T_2(Q) = T_2(T_1(P))$affine?
- Let $T_1=M_1$ and $T_2=M_2$, where M_1 and M_2 are the appropriate matrices.
- $W = M_2(M_1P)) = (M_2M_1)P = MP$ by associativity.
- So $M = M_2M_1$, the product of 2 matrices (in reverse order of application), which is affine.



Composing Affine Transformations: Examples

- To rotate around an arbitrary point: translate P to the origin, rotate, translate P back to original position. $Q = T_P R T_{-P} P$
- Shear around an arbitrary point:
  $Q = T_P H T_{-P} P$
- Scale about an arbitrary point:

  $Q = T_P S T_{-P} P$

Composing Affine Transformations (Examples)

- Reflect across an arbitrary line through the origin $\mathcal{O}$: Q = R(θ) S R(-θ) P
- The rotation transforms the axis to the x-axis, the reflection is a scaling, and the last rotation transforms back to the original axis.
- Window-viewport: Translate by -w.l, -w.b, scale by A, B, translate by v.l, v.b.

Properties of 2D and 3D Affine Transformations

- Affine transformations *preserve* affine combinations of points.
  - $W = a_1P_1 + a_2P_2$ is an affine combination.
  - $MW = a_1MP_1 + a_2MP_2$
- Affine transformations preserve lines and planes.
  - A line through A and B is $L(t) = (1-t)A + tB$, an affine combination of points.
  - A plane can also be written as an affine combination of points: $P(s, a) = sA + tB +(1 – s – t)C$.

Properties of Transformations (2)

- Parallelism of lines and planes is preserved.
  - Line A + **b***t* having direction **b** transforms to the line given in homogeneous coordinates by M(A + **b**t) **=** MA **+** M**b**t, which has direction vector M**b***.*
  - M**b** does *not* depend on point A. Thus two different lines A_1+ **b***t* and A_2 + **b***t* that have the same direction will transform into two lines both having the direction, so they *are* parallel.
- An important consequence of this property is that *parallelograms map into other parallelograms*.

Properties of Transformations (3)

- The direction vectors for a plane also transform into new direction vectors independent of the location of the plane.
- As a consequence, parallelepipeds map into other parallelepipeds.

Properties of Transformations (4)

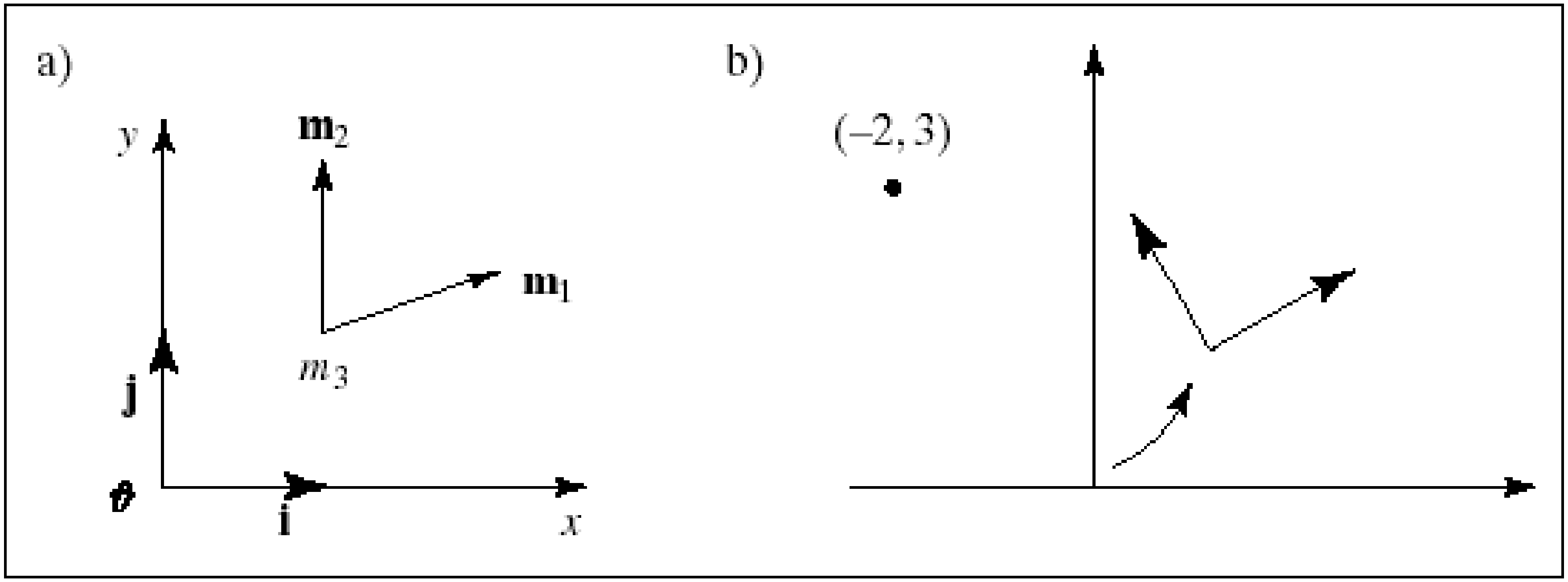
- The columns of the matrix reveal the transformed coordinate frame:
  - Vector **i** transforms into column m$_1$, vector **j** into column m$_2$, and the origin $\mathcal{O}$ into point m$_3$.
  - The coordinate frame (**i**, **j**, $\mathcal{O}$) transforms into the coordinate frame ($\mathbf{m}_1$, $\mathbf{m}_2$, m_3), and these new objects are precisely the columns of the matrix.

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} = \left(m_1 \mid m_2 \mid m_3\right)$$

Properties of Transformations (5)

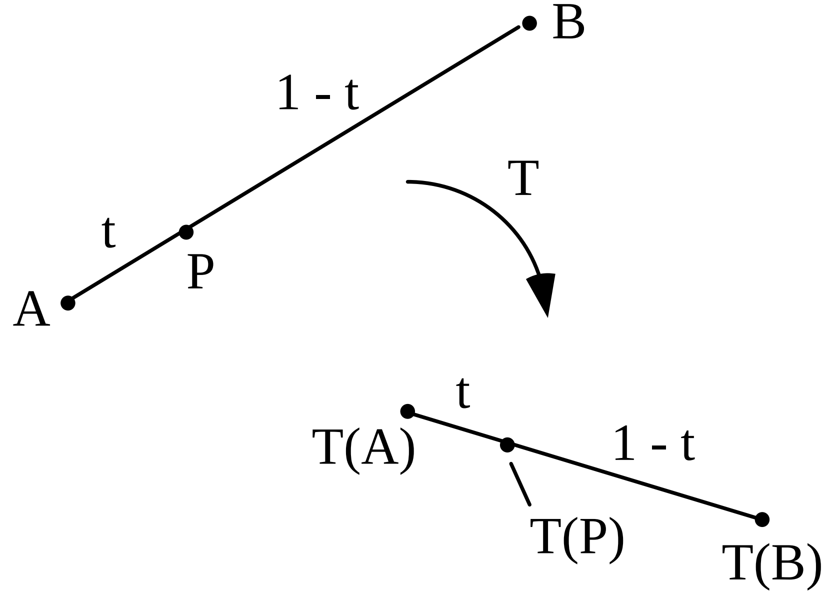
- The axes of the new coordinate frame are not necessarily perpendicular, nor must they be unit length.
  - They are still perpendicular if the transformation involves only rotations and uniform scalings.
- Any point $P = P_x\mathbf{i} + P_y\mathbf{j} + \mathcal{O}$ transforms into $Q = P_x\mathbf{m}_1 + P_y\mathbf{m}_2 + m_3$.

Properties of Transformations (6)



Properties of Transformations (7)

- Relative ratios are preserved: consider point *P* lying a fraction *t* of the way between two given points, *A* and *B* (see figure).
- Apply affine transformation *T*( ) to *A* , *B*, and *P*.
- The transformed point, *T*(*P*), lies the *same* fraction *t* of the way between images *T*(*A*) and *T*(*B*).



Properties of Transformations (8)

- How is the area of a figure affected by an affine transformation?
- It is clear that neither translations nor rotations have any effect on the area of a figure, but scalings certainly do, and shearing might.
- The result is simple: When the 2D transformation with matrix *M* is applied to an object, its area is multiplied by the *magnitude of the determinant* of *M*:

$$\frac{area\,after\,transformation}{area\,before\,transformation} = \left|\det M\right|$$

Properties of Transformations (9)

- In 2D the determinant of the matrix *M* is $(m_{11}m_{22} - m_{12}m_{21})$.
- For a pure scaling, the new area is S_xS_y times the original area, whereas for a shear along <u>one</u> axis the new area is the same as the original area.
- In 3D similar arguments apply, and we can conclude that the volume of a 3D object is scaled by $|\det M|$ when the object is transformed by the 3D transformation based on matrix *M*.

Properties of Transformations (10)

- Every affine transformation is composed of elementary operations.
- A matrix may be factored into a product of elementary matrices in various ways. One particular way of factoring the matrix associated with a 2D affine transformation yields

  M = (shear)(scaling)(rotation)(translation)
- That is, any 3 x 3 matrix that represents a 2D affine transformation can be written as the product of (reading right to left) a translation matrix, a rotation matrix, a scaling matrix, and a shear matrix.