GRAPHICS TRANSFORMATIONS IN 2-D

Transformations are fundamental to computer graphics. They enable, for example, two-dimensional (2-D) perspective representation of three-dimensional (3-D) objects. 2-D transformations themselves have many less obvious but equally important applications. Rotation, scaling, and location of text, placement of symbols in charts, map projections, clipping objects within panes and windows, and managing the behavior of in-place editing tools all require the use of 2-D transformations somewhere in the coding of a graphical system. Furthermore, transformations of the plane have become especially important recently in the development of interfaces for navigating highly dense data configurations, networks, and tables. We will begin with an introduction to a basic hierarchy of 2-D transformation groups and then examine some general 2-D coordinate transformations. Useful references for applications are Rogers and Adams (1), Foley et al. (2), Glassner (3), Arvo (4), Hill (5), and Hearne and Baker (6).

TRANSFORMATIONS OF THE PLANE

A transformation is a function

$$T: S \rightarrow S$$

mapping S to itself. For example, if (u, v) and (x, y) are elements of two sets S_1 and S_2, respectively, then the set of equations

$$x = g(u, v)$$
$$y = h(u, v)$$

where g and h are functions, transforms (u, v) to (x, y). As with all functions, we call (x, y) the image of (u, v) and $T(u, v) = (g(u, v), h(u, v))$ a mapping of S_1 to S_2. The mathematical term *image* is particularly appropriate for understanding what we are doing with graphical transformations, because it suggests that we are producing a figure that is an image (reorientation, projection, distortion, etc.) of an original. All transformations are functions, but not all functions are transformations. Transformations are a subclass of functions because a transformation maps a set to itself: A transformation can be composed with itself by using its output as input. In other words, we can define a new transformation T^2 by the formula $T^2 = T(T(x))$. In computer graphics, this means that we can compose a sequence of operations (for example, repeated multiplications of a series of matrices) into a single, more efficient operation (multiplication by one composed matrix). For examples of how 2-D matrix transformations can be composed, see Rogers and Adams (1). Some 2-D transformations cannot be done by simple matrix multiplication, but they can nonetheless be composed. Some newer object-oriented graphics systems, for example Java, recognize this by allowing one to collect *chains* of transformation objects.

Figure 1 shows a class hierarchy of planar transformations. The "Class" column shows each class of transformation inheriting from its more general parent. For example, an isometry is a similarity, but a similarity is not an isometry. The "Transformation" column indicates the methods within the transformation class. The "Invariance" column indicates the feature of graphical objects unchanged after transforma-

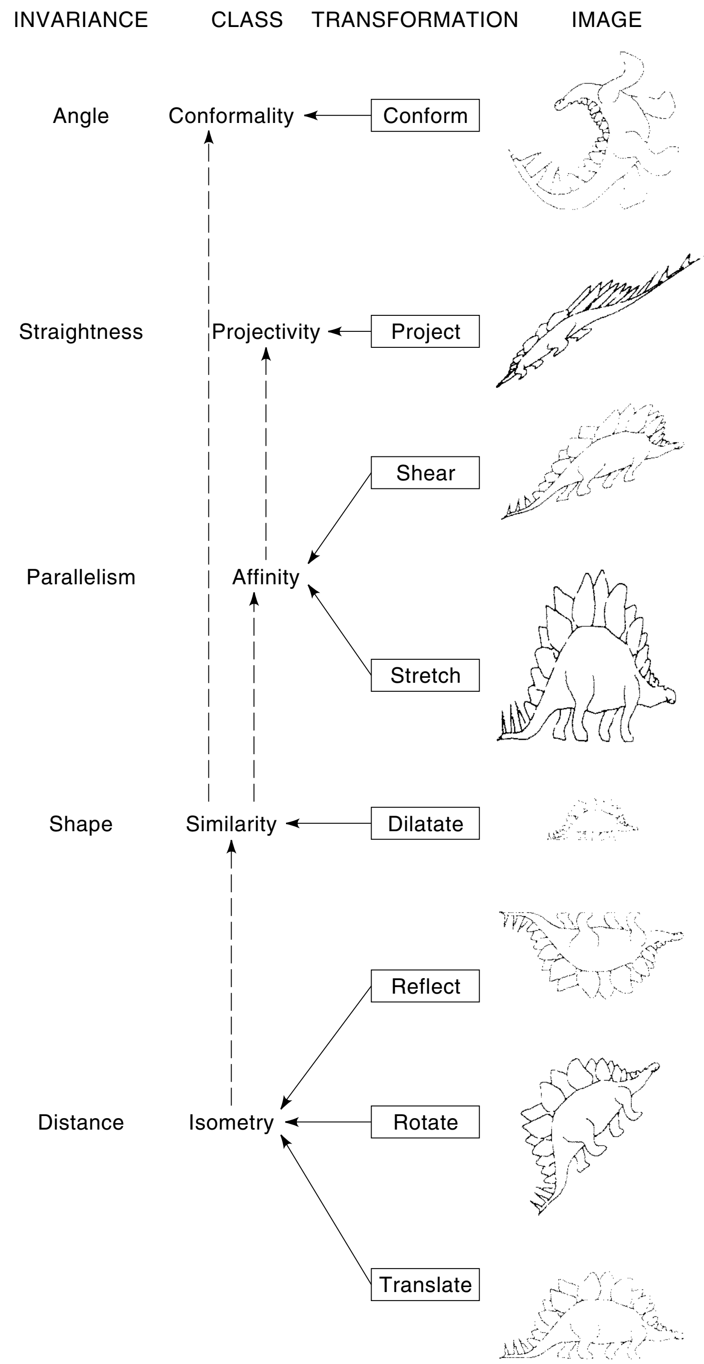

Figure 1. Hierarchy of planar transformations.

tion. Finally, the "Image" column shows the effect of each tranformation on a dinosaur sketch. This figure does not cover all possible transformations of the plane to itself, nor does it constitute a strict hierarchy. The reasons for this will become more apparent in the sections to follow.

The isometry group is the set of transformations that preserve distance between points. These operations obey the axioms of Euclidean geometry. The three isometric transformations are the rigid transformations: translation, rotation, and reflection. Translation moves an object vertically or horizontally without changing its shape, size, or orientation. Rotation rotates an object around a point (usually its center) without changing shape or size. Reflection inverts an object horizontally or vertically without changing its size or shape, like looking in a mirror.

The similarity group is the set of transformations that change the size of an object. The method name, dilatate, suggests enlargement, but it includes both shrinking and enlargement.

The affinity group is the set of transformations that cause a dimension to stretch or shrink independently of the other. It also includes a shear, which is like turning Roman into italic letters. Other shearings resemble a flexible object squeezed between the blades of a pair of scissors.

The projectivity group is the set of transformations that is most easily visualized by thinking of a light source shining on shapes drawn on a transparent plane and projecting a shadow onto another plane. This transformation preserves straight lines but can modify angles considerably.

The conformality class covers conformal mappings. Conformal mappings preserve local angles, but may distort global shape considerably. The conformality class, like the affinity class, is a parent of the similarity class.

Isometric Transformations

A metric space is a set S together with a function

$$\delta : S \times S \rightarrow [0, \infty)$$

where

$$\delta(x, y) = 0 \Leftrightarrow x = y$$
$$\delta(x, y) = \delta(y, x)$$
$$\delta(x, z) \leq \delta(x, y) + \delta(y, z)$$

Although this definition is general enough to be applied to objects other than real numbers, we will assume that δ is a distance measure and x, y, z are points in a space defined on the real numbers. Thus, (1) zero distance between two points implies that the points are the same, and if two points are the same, the distance between them is zero, (2) the distance between the point x and the point y is the same as the distance between the point y and the point x, and (3) a triangle inequality among distances exists for any three points x, y, and z.

An instance of a metric space is the n-dimensional Euclidean space consisting of n-tuples $(x_1, x_2, \ldots, x_n)$ of real numbers x_i, $i = 1, \ldots, n$, with distance metric

$$\delta = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$$

If S_1 and S_2 are metric spaces with distance functions δ_1 and δ_2, then a function $g: S_1 \rightarrow S_2$ is an isometry transformation if and only if

$$\delta_2((g(x)), g(y)) = \delta_1(x, y) \qquad \text{for all } x, y \in S_1$$

Isometries on the plane involve translation, rotation, and reflection. All of these preserve distance. While there are formal proofs for this assertion, the simplest thing is to look at pictures.

Translation. Translation sends the coordinates (x, y) to $(x + a, y + b)$. Translation moves a graphic right or left, up or down, or a combination of both, without changing its orientation. The dinosaur in Fig. 1 was translated from somewhere off the page.

Rotation. Rotation sends the polar coordinates (r, θ) to $(r, \theta + c)$. This is equivalent to sending (x, y) to $(\cos(\theta)x - \sin(\theta)y, \sin(\theta)x + \cos(\theta)y)$.The dinosaur transformation in Fig. 1 is the rotation $(\cos(\theta)x - \sin(\theta)y, \sin(\theta)x + \cos(\theta)y)$, where θ is 45°.

Reflection. Reflection sends (x, y) to $(-x, y)$ or to $(x, -y)$. This operation reverses the vertical or horizontal orientation of a graphic. The dinosaur is upended in Fig. 1 by negating the second coordinate.

Similarity Transformations

A transformation g is a similarity if and only if there is a positive number r such that

$$\delta_2(g(x), g(y)) = r\delta_1(x, y) \qquad \text{for all } x, y \in S_1$$

Similarities on the plane involve isometries as well as dilatation.

Dilatation. Dilatation sends polar coordinates (ρ, θ) to $(c\rho, \theta)$ or rectangular coordinates (x, y) to (cx, cy). Dilatation works like a photo magnifier or reducer. The dinosaur is transformed in Fig. 1 by $c = 0.5$.

Affine Transformations

The n-tuple coordinate for a point in a space can be represented by the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Vector notation allows us to express simply the affine class of transformations:

$$\boldsymbol{x}^* = \boldsymbol{x}\boldsymbol{T} + \boldsymbol{c}$$

where $\boldsymbol{x}^*$, $\boldsymbol{x}$, and $\boldsymbol{c}$ are row vectors and $\boldsymbol{T}$ is an n by n transformation matrix. In this notation, $\boldsymbol{xT}$ is the image of $\boldsymbol{x}$. If $\boldsymbol{c} = \boldsymbol{0}$, we call this a linear mapping. The linear subset of the affine class includes rotation, reflection, and dilatation, as well as stretch and shear. If $\boldsymbol{c} \neq \boldsymbol{0}$, we call it an affine mapping. This adds translation to these operations.

Let's first review the matrix form of the isometric and similarity transformations we have seen so far. Beginning with $\boldsymbol{T}$, we can see that an identity transformation results from making $\boldsymbol{T}$ an identity matrix:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Rotation involves the more general matrix

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

where θ is the angle of rotation.

Reflection involves an identity matrix with one or more diagonal elements signed negative—for example,

$$\boldsymbol{T} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Any negative diagonal element will reflect the corresponding dimension of $\boldsymbol{x}$. This particular $\boldsymbol{T}$ matrix reflects both dimensions.

Dilatation involves a matrix of the form

$$\boldsymbol{T} = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$$

where a is a real number.

Finally, translation involves a row vector of the form

$$\boldsymbol{c} = [u \quad v]$$

where u and v are real numbers.

The affine class permits $\boldsymbol{T}$ to be a real matrix of the form

$$\boldsymbol{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where a, b, c, and d are real numbers. Stretch and shear are produced by two types of this matrix.

Stretch. Stretch is the transformation that sends (x, y) to (ax, dy), so

$$\boldsymbol{T} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$

The stretch transformation varies the aspect ratio of a graphic. This is the ratio of the physical height to the physical width of the frame graphic.

Shear. Shear is the transformation that sends (x, y) to $((ax + cy), (bx + dy))$, so

$$\boldsymbol{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The sheared dinosaur in Fig. 1 was produced by the matrix whose elements are $a = 0.96$, $b = 0.3$, $c = 0.3$, and $d = 0.96$.

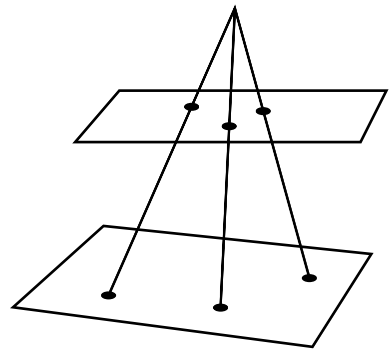

Figure 2. Planar projection.

Planar Projections

A planar projection is the mapping of one plane to another by perspective projection from any point not lying on either. Figure 2 illustrates this mapping spatially. As the figure suggests, we also may use a perspective projection to create 2-D perspective views of 3-D objects in computer graphics. Planar projections are more restrictive than 3-D to 2-D projections, however. They share the *composition* behavior of other planar transformations. We can, in other words, project a projection and stay within the projectivity class.

To notate projections, it is helpful to adopt homogeneous coordinates. We combine the $\boldsymbol{T}$ and $\boldsymbol{c}$ matrices into one general square matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} a & b & p \\ c & d & q \\ u & v & s \end{bmatrix}$$

The elements a, b, c, and d are from the $\boldsymbol{T}$ matrix, and u and v are from the $\boldsymbol{c}$ vector that we used for affine transformations. The elements p, q, and s are for projection. To make this system work, we need to express $\boldsymbol{x}$ in homogeneous coordinates by augmenting our coordinate vector by one element:

$$\boldsymbol{x} = (x, y, h)$$

If $h = 1$, then our Cartesian coordinates are simply $x = x/h$ and $y = y/h$. This reparameterization makes the general projective transformation

$$\boldsymbol{x}^* = \boldsymbol{x}\boldsymbol{A}$$

This matrix equation produces the following homogeneous coordinates:

$$\boldsymbol{x}^* = ((ax + cy + u), (bx + dy + v), (px + qy + s))$$

If we renormalize after the transformation so that the third coordinate is unity, we can retrieve (x^*, y^*) as the Cartesian coordinates from the projection. To see what projection adds to the affine class, we should notice that the third column of $\boldsymbol{A}$ produces a different scaling of x and y, depending on their values. And because all the transformations are linear, the straightness of lines is preserved in the class.

Project. The projected dinosaur in Fig. 1 was produced by the coordinate transformation

$$(x, y) \to (1/x, y/x)$$

The projection matrix for expressing this transformation is

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

This takes $\boldsymbol{x}$ in homogeneous coordinates to $\boldsymbol{x}^* = (1, y, x)$, which produces the result we want in Cartesian coordinates after dividing through by x.

Conformal Mappings. We need to generalize our coordinates once more in order to move to the next level of the planar transformation hierarchy. By working on the complex plane, we can define functions that would be messy or difficult to understand in the real domain. A complex number $z = x + iy$ may be represented by a vector $\boldsymbol{z}$ on the complex plane whose coordinates are $\mathrm{Re}(z) = x$ and $\mathrm{Im}(z) = y$. Coordinate transformations on (x, y) can then be expressed in the form

$$w = f(z) = u(z) + iv(z)$$

where $u(z)$ and $v(z)$ are real functions of z and where w is the image point of z under f.

Similarity transformations can be expressed in the complex formula

$$w = az + b$$

where w, a, b, and z are all complex. We can see this by noting that

$$(a_1 + ia_2)(x + iy) = (a_1x - a_2y) + i(a_2x + a_1y) + (b_1 + ib_2)$$

which is the same set of operations involved in the similarity subclass of the projective transformation

$$\boldsymbol{xA} = (x, y, 1) \cdot \begin{bmatrix} a_1 & a_2 & 0 \\ -a_2 & a_1 & 0 \\ b_1 & b_2 & 1 \end{bmatrix}$$

The projective matrix notation tells us that the complex constant b is involved in translation and that the complex constant a is involved in rotation and dilatation of the plane represented in z, since the submatrix

$$\begin{bmatrix} a_1 & a_2 \\ -a_2 & a_1 \end{bmatrix} = r \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

where r is a real number. There is another way to show the rotational role of the complex constant a. Euler's formula

$$e^{i\theta} = \cos\theta + i\sin\theta$$

locates a point on the unit circle at angle θ on the complex plane. It tells us that any complex number can be expressed as

$$z = re^{i\theta}$$

We can thus reexpress the complex constant a and define a similarity transformation as

$$w = re^{i\theta}z + b$$

which rotates z by θ and dilatates it by r.

A conformal mapping adds a peculiar geometric characteristic to a similarity transformation: Local angles (at the intersection of two curves) are preserved, but straight lines may become curves. A planar mapping is conformal if every point on the plane is transformed so that all possible infinitesmal vectors emanating from that point are rotated and dilatated by the same amount in the image. This local rotation and dilatation means that very small squares remain squares in the image, but large squares can be distorted considerably. The paradoxical beauty of this transformation is that locally it looks like a similarity but globally it looks like a nonlinear warping.

Of particular interest is the Möbius transformation

$$w = \frac{az + b}{cz + d}$$

where all the constants and variables are complex. This transformation has inspired a variety of basic applications in physics, fluid dynamics, electromagnetic fields, and other areas. Needham (7) offers a glimpse into this world from a geometric perspective and illustrates its application to vector flows and other graphics in physics. The conformal dinosaur in Fig. 1 was produced by the transformation

$$w = \frac{1 - z}{1 + z}$$

GENERAL 2-D COORDINATE TRANSFORMATIONS

The classes of transformations presented in Fig. 1 cover the most common, but by no means the majority, of transformations used in 2-D computer applications. Almost any continuous warping of the plane could be seen to have an application in image processing and geometic modeling. This section will mention only two—polar coordinates and fisheye views—that are widely used in scientific graphics and user–interface design.

Polar Coordinates

If (u, v) represents the polar coordinates (ρ, θ) of a point, then a polar coordinate function $P(u, v)$ corresponds to the case $u = \rho$, $v = \theta$, where $x = \rho\cos\theta$ and $y = \rho\sin\theta$. There are numerous applications of the polar transformation in engineering and business graphics. The most common is the ordinary pie chart. Engineers and scientists often plot periodic

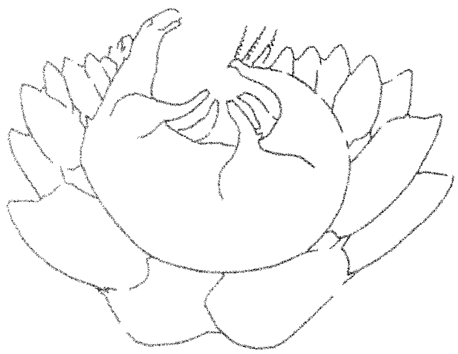

Figure 3. Polar dinosaur.

functions and data in the polar domain. Figure 3 shows our dinosaur in polar form, assuming the length coordinates of the body rougly span the interval $(0, 2\pi)$ and the height coordinates are positive.

We have two choices if the polar domain extends beyond 2π radians. One is to let the image overlap itself, thus superimposing each element of the set $\{. . ., (-2\pi, 0[, (0, 2\pi[, (2\pi, 4\pi[, . . .\}$ in a single circle. This method is often used for plotting periodic functions. The other approach is to set aside a separate circle for each element and plot the image segmented in adjacent circles.

Lensing

The fisheye transformation expands a graph away from an arbitrary locus, usually the center of the frame or viewing area. This class of transformations has received a lot of attention by computer interface designers because of the need to make the best use of limited screen "real estate" when navigating through dense networks and graphical browsers (8–10). A broad class of smooth functions will serve these purposes. For real-time applications, computational speed is critical, so a function like

$$fish : x \to 2^x/(1 + 2^x)$$

will do well in integer arithmetic.

The fisheye transformation can be used on either coordinate (for vertical or horizontal lensing of tables of objects) or on both (for lensing uniformly dense displays). Figure 4 shows a fisheye transformed dinosaur with the center of focus at the middle of the body. This reveals most detail in the center. Lensing can be used on almost any graphical object, including text fields, spreadsheets, and tables.

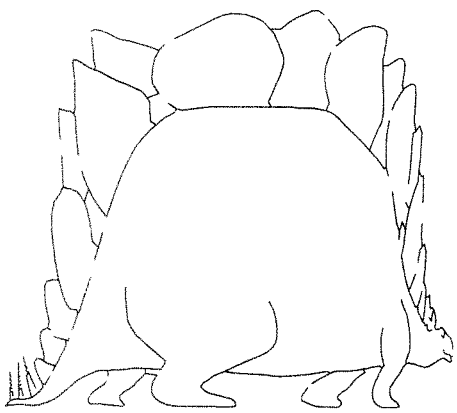

Figure 4. Fisheye transformed dinosaur.

BIBLIOGRAPHY

1. D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics,* New York: McGraw-Hill, 1976.
2. J. D. Foley et al., *Introduction to Computer Graphics,* 2nd ed, Reading, MA: Addison-Wesley, 1993.
3. A. S. Glassner (ed.), *Graphics Gems,* New York: Academic Press, 1990.
4. J. Arvo (ed.), *Graphics Gems II,* New York: Academic Press, 1991.
5. F. S. Hill, Jr., *Computer Graphics,* New York: Macmillan, 1990.
6. D. Hearne and M. P. Baker, Computer Graphics, C Version, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1997.
7. T. Needham, *Visual Complex Analysis,* Oxford: Clarendon Press, 1997.
8. G. W. Furnas, Generalized Fisheye Views, *Human Factors Comput. Syst.: CHI 86 Conf. Proc.,* New York: ACM Press, 1986, pp. 16–23.
9. M. Sarkar and M. H. Brown, Graphical fisheye views. *Commun. ACM,* **37**: 73–83, 1994.
10. Y. K. Leung and M. D. Apperly, A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. CHI,* **1**: 126–160, 1994.

Reading List

T. F. Banchoff, *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions,* New York: Freeman, 1996.

D. V. Ahuja and S. A. Coons, Geometry for construction and display, *IBM Syst. J.* **7** (3–4): 188–205, 1968.

LELAND WILKINSON
SPSS Inc.