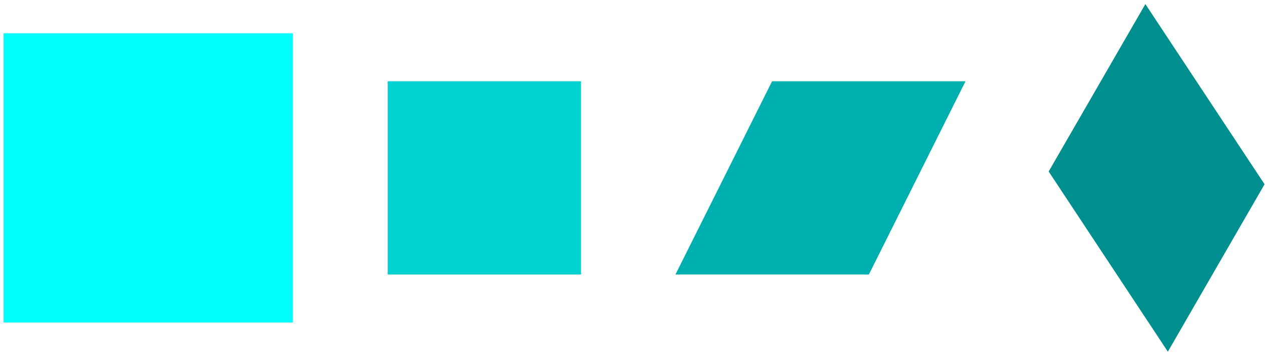
Geometrical Transforms



CSC 292:
Computer Graphics

Randal C. Nelson

2D Transformations

- To start, we represent points on plane as 2 component column vectors:

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$

- If we can describe the transformation of points, then we can describe the transformation of other primitives - which are generally collections of points.
- For now, assume a floating-point world.
- 2-D transformations of interest:
  - Translation.
  - Scaling (relative to a point, along different axes).
  - Rotation (about a point).

Translation

We can write the translation of a point (x, y) by writing

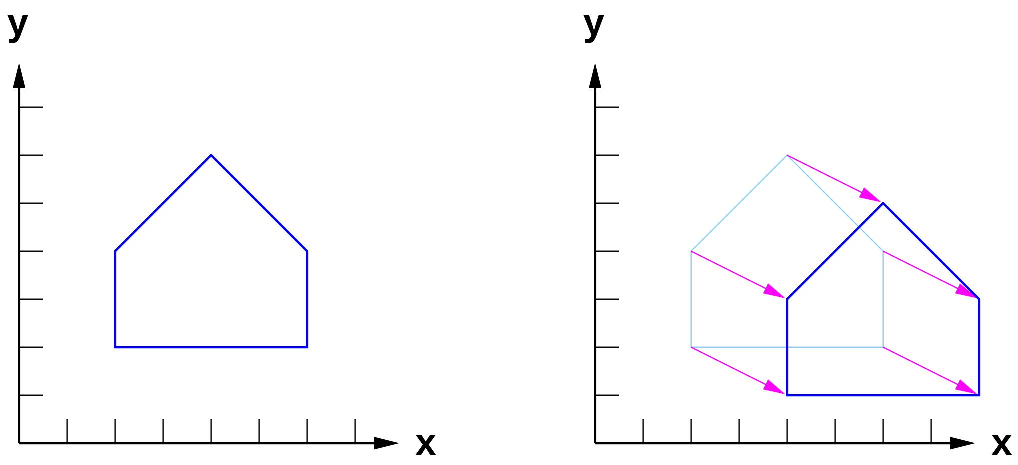
$$x' = x + dx; y' = y + dy.$$

More compactly, translations can be represented by vector addition,

$$P' = P + T$$

where we have defined the column vectors

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} dx \\ dy \end{bmatrix}.$$



- By translating every point in a primitive using the same equation, we translate the entire primitive.
- Note that the different points move along parallel lines.

Scaling

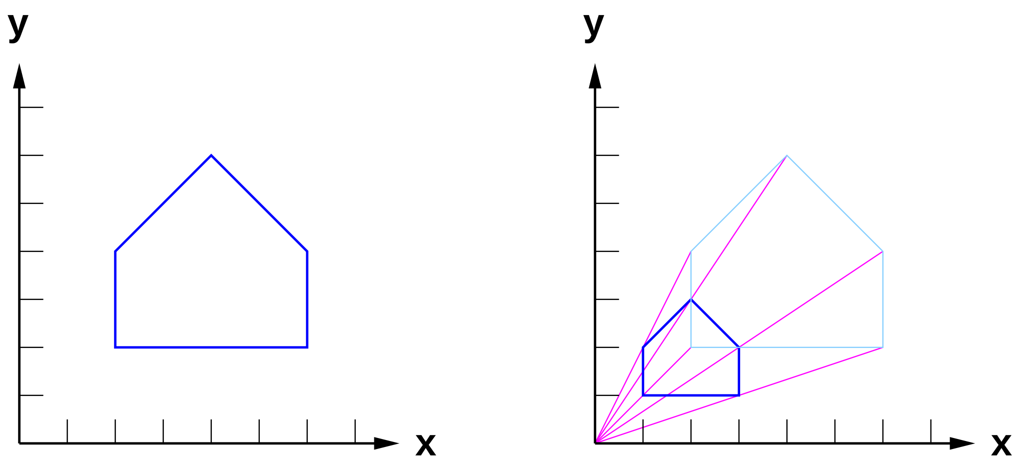
We can scale both the x and y coordinates of a point, independently, as follows.

$$x' = s_x \cdot x; y' = s_y \cdot y.$$

This can also be written in matrix form:

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

or $P' = S \cdot P$ where S is a *scaling matrix*.



- As before, we can scale and entire figure by scaling each point by the same equation.
- Note that if $s_x = s_y$ then all the points move directly towards the origin. In other cases the curves traced out are exponentials.

Rotation

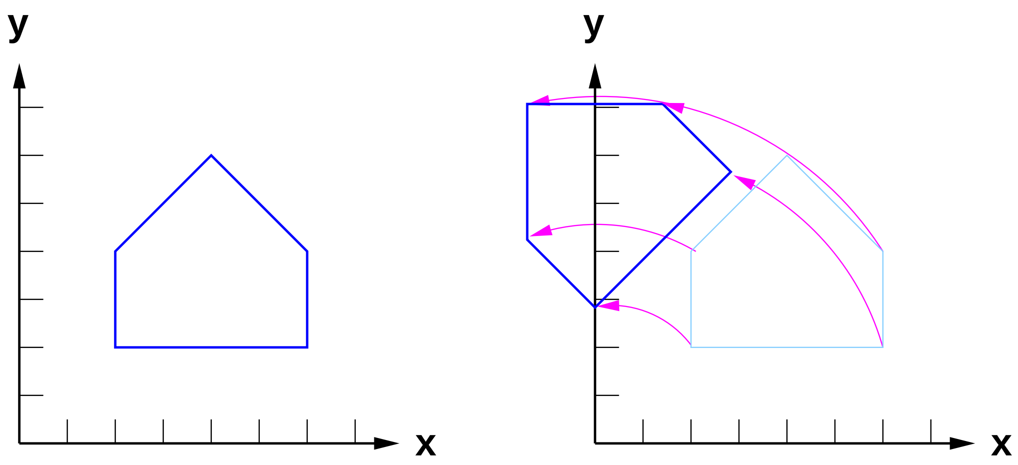
A point can be rotated through an angle θ about the origin using the equation:

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

In matrix form we have:

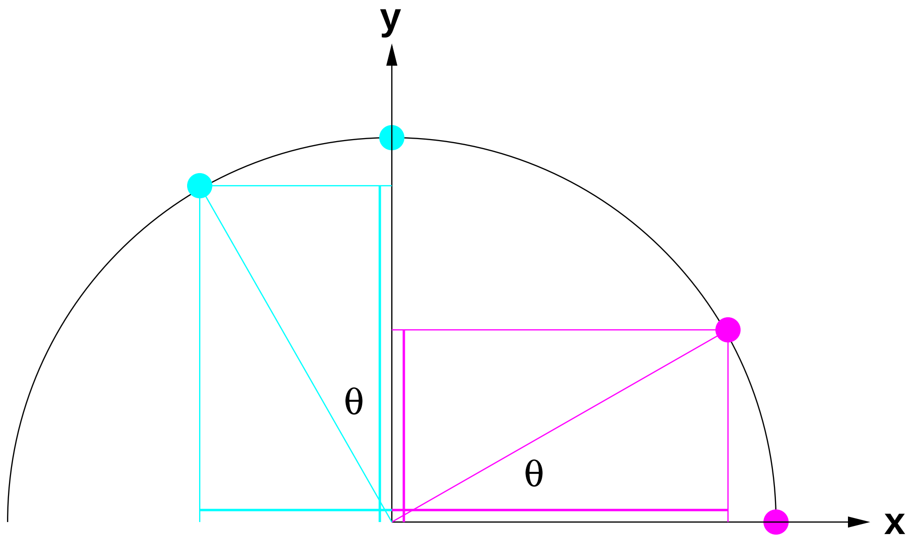
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

or $P' = R \cdot P$ where R is a *rotation matrix*.



- A figure can be rotated by transforming all the component points by the same matrix.
- Note that all the points move along concentric circles centered at the origin.
- Also note that a figure not at the origin is effectively translated as well as rotated.

Derivation of Rotation



To derive the rotation formula, consider what happens to the unit vectors $\hat{x}$ and $\hat{y}$ when they are rotated by an angle θ.

$$\hat{x} \Rightarrow \cos\theta\hat{x} + \sin\theta\hat{y}; \hat{y} \Rightarrow -\sin\theta\hat{x} + \cos\theta\hat{y}$$

If we recall that a point (x, y) represents the vector $x \cdot \hat{x} + y \cdot \hat{y}$, substitute in the rotated versions of $\hat{x}$ and $\hat{y}$ to get a representation for the rotated vector, and collect terms, we get

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

Homogeneous coordinates

- Note that scaling and rotation can be represented by matrix multiplication, but translation is a vector addition.
- It would be nice if all our basic transformations were matrix multiplications. Then we could put them together by multiplying matrices.
- It turns out we can do this if we represent 2-D points not by a pair of numbers (x, y) (represented as a 2-vector), but by a triple (X, Y, W) called a set of *homogeneous coordinates*.
- Two sets of homogeneous coordinates represent the same 2-D point if and only if they are multiples of each other.

- Each 2-D point thus has many representations in homogeneous coordinates, which if interpreted as 3-D points, form a line in 3-space passing through the origin, and intersecting the 2-D point at the $z = 1$ plane.

- To get a unique representation, we *homogenize* the point (X, Y, W) by dividing by W to get the form $(x, y, 1)$.

- The points with $W = 0$ are referred to as the *points at infinity*, and are important in *projective geometry*, but we will not see too much of them in this course.

Transformations in Homogeneous Coordinates

It turns out that translation, scaling and rotation can all be represented as 3×3 matrix multiplications in homogeneous coordinates.

Translation takes the form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Scaling has the form:

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

And rotation has the form

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

Forms of Matrices

Consider the rows of the 2×2 rotation matrix

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

- Each row is a unit vector.
- The rows are orthogonal (the dot product is 0).
- When rotated by θ the first vector ends up on the x axis, and the second on the y axis.
- The same properties hold for the columns, except that the first and second rows are the vectors into which the x and y unit vectors are rotated.
- A matrix with the first two properties is called *orthogonal*, and has the additional property that its inverse is its transpose.

A transformation matrix of the form

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the upper left 2×2 submatrix is orthogonal is called a *rigid-body* transformation.

- Preserves angles.
- Preserves lengths.
- Can perform an arbitrary translation and rotation.

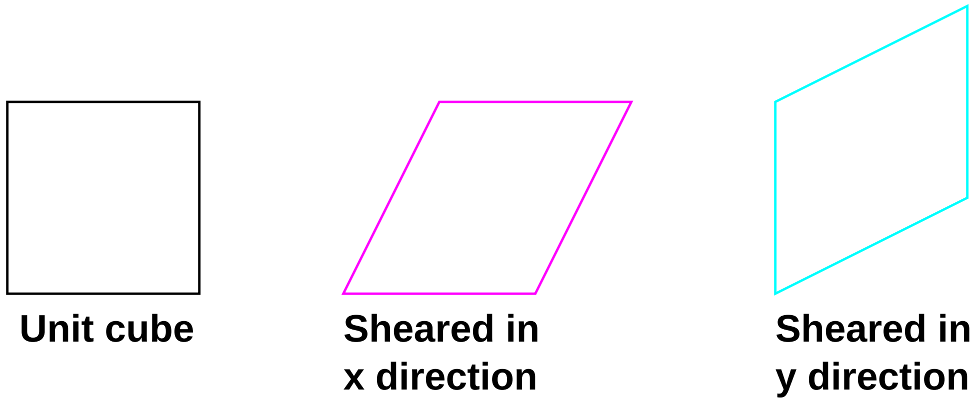
The product of an arbitrary sequence of translation, scaling, and rotation matrices, is referred to as an *affine* transformation. In our 3×3 representation, such transformations have the form:

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the upper 2×2 matrix is not necessarily orthogonal.

- Preserves parallelism of lines.
- Does not necessarily preserve angles or lengths.
- Most general transformation possible with our representation.
- Can perform arbitrary translations, scalings, and rotations, and also *shear transformations*

Shear transformations have a simple geometric interpretation as a skewing of the coordinate system.



They are produced by matrices of the form:

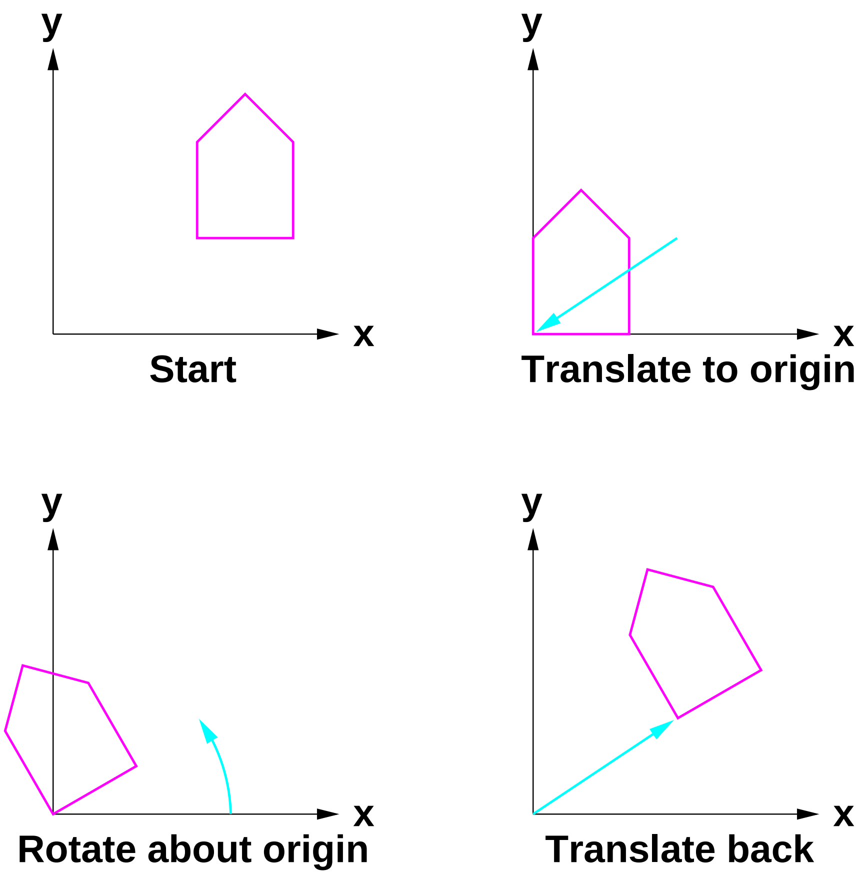
$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- Can be produced by a combination of rotation and non-uniform scaling.
- Hence do not represent a new independent class of transformations.

Composition of Transformations

- Suppose we want to rotate a figure about some point $P_1 = (x_1, y_1)$ not the origin.
- We don't have a primitive to do this directly.
- However, we can carry out the operation performing a sequence of fundamental transformations.

1. Translate so that P_1 is at the origin.
2. Rotate about the origin.
3. Translate back so that the origin returns to P_1.



A single matrix performing the transformation can be obtained by multiplying the individual matrices together (in reverse order).

$$A_{tot} = T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1) =$$

$$\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & Y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta) + y_1 \sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta) - x_1 \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

- Other non-primitive transformations (e.g. scaling about a point) can be performed in a similar manner.
- Recall that, in general matrix multiplication does not *commute*, so that order is important.
- Of the six combinations of primitive operations, the following pairs commute.
  - Translation with translation.
  - Scaling with scaling.
  - Rotation with rotation.
  - Rotation with uniform scaling ($s_x = s_y$).
- The following do not
  - Translation with scaling.
  - Translation with rotation.
  - Rotation with non-uniform scaling.

Window to Viewport Transformations

- It is often convenient to specify primitives in a meaningful *world-coordinate* system (miles, microns, meters etc.)
- In general, this will not match up with the *screen coordinate* system, either in origin or units (sometimes the rotation is different too)
- We need to specify the relationship between the world and screen systems.
- One way is for the programmer to provide a transformation matrix explicitly (not very intuitive).
- Could also specify matching rectangular windows - a *world-coordinate window* and a screen coordinate *viewport*, and determine matrix from there (more natural).

The final transformation matrix can be determined by composing the matrices for the following transformations.

1. Translate the world-coordinate window to the origin.
2. (Rotate the window to match the orientation of the screen window.)
3. Scale the window to match the size of the viewport.
4. Translate the window to the screen coordinates.

Efficiency

Recall that the most general transformation has the form:

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- 3×3 matrices are nice symbolically, but inefficient computationally.
- 9 multiplications and 6 additions needed per point.

- If we use the structure of the matrix, we note that

$$\begin{aligned} x' &= x \cdot r_{11} + y \cdot r_{12} + t_x \\ y' &= x \cdot r_{21} + y \cdot r_{22} + t_y. \end{aligned}$$

- This uses 4 multiplies and 4 adds.
- Can be even more efficient for small angles if approximation is ok.

3D Transformations

- Just as 2D transformations can be represented using 3×3 matrices and homogeneous coordinates, 3D transformations can be represented using 4×4 matrices and a homogeneous representation for a 3-vector.
- Same basic transformations, but in 3D:
  - 3D translation (3 values).
  - 3D scaling (3 values, along the three axes).
  - 3D rotation (3 values, about the three axes).

- For rotation, *handedness* of coordinate system matters.
- For right-handed system, a 90 degree positive (counterclockwise) rotation about the z axis looking from $+z$ towards the origin, takes $+x$ into $+y$. Other relations are found by permuting around the cycle $x \to y \to z \to x$.
- Also whether we use *fixed angles* or *Euler angles*.

Matrix Reps. for 3D Transforms

- The homogeneous representation for the point (x, y, z) is $(x, y, z, 1)$.
- Translation is represented by the matrix:

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Scaling has the form:

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- There are three rotation matrices, one for rotation about each axis. They could be multiplied together, but the result is order dependent and quite complicated.
- For rotation about the z-axis, the matrix is just an extended version of the 2D rotation matrix:

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

- The x-axis rotation matrix is:

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

- And the y-axis rotation matrix is:

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

- The upper left 3×3 submatrices of all the rotation matrices is orthogonal, and the same holds for any product of rotation matrices (which is just a complex rotation).

The general *affine* form obtained by multiplying a sequence of translation, scaling, and rotation matrices together is:

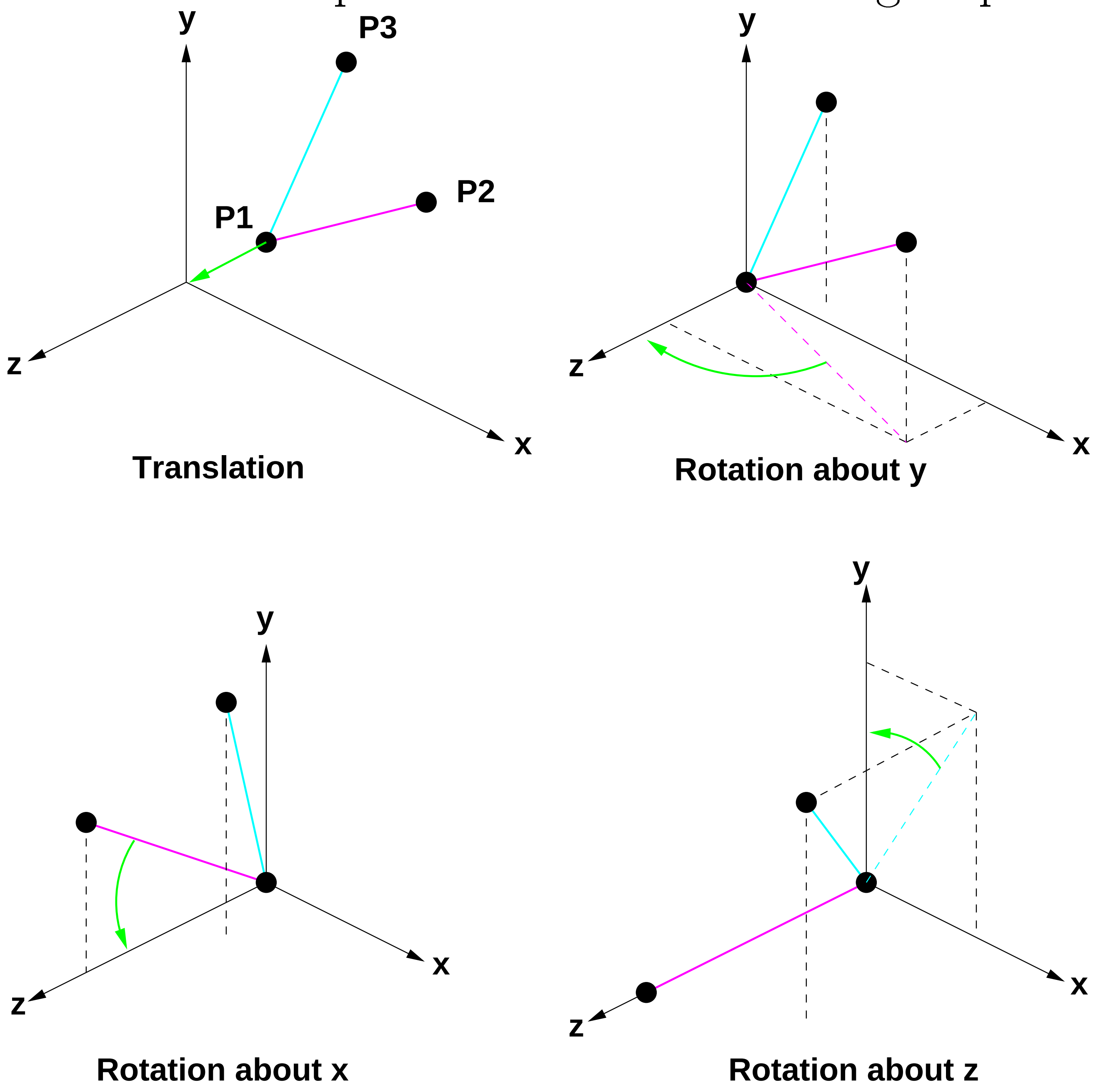
$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The r_{ij} entries represent a combined rotation and scaling matrix, and the t_i entries are a combined translation.
- Such matrices are generally invertable (unless a scale to 0 on some axis or *projection* is performed).
- As before, a *rigid transformation* is composed from translation and rotation, but no scaling operations, and preserves angles and lengths.

Composition of 3D transforms

- 3D transformations can be composed in the same way as the 2D transformations.
- Translation, scaling, and rotation about a single axis commute with themselves, as do uniform scaling and any complex rotation.
- Other combinations, including rotations about different axes do not commute.
- 3D transformations can be determined in various ways.

For example, If the goal is to transform the following figure so that P1 is at the origin, segment (P1, P2) lies along the z axis, and segment (P1, P3) is in the $y - z$ plane, then we can accomplish it with the following steps.



Transformations as Changes of Coordinate System

- So far, we have thought of transformations as moving objects within a coordinate system.
- An equivalent point of view is to consider transformations as changing the coordinate system that an object is described in.
- Transforming a coordinate system means that all the points in the space get re-labeled (as opposed to relabeling the object points).

- For example, the translation $x' = x + dx$ can either be viewed as moving an object along the x-axis by dx, or as placing the object in a new coordinate system, whose origin is displaced by an amount -dx along the x axis.
- General rule is that transforming an object by Q is equivalent to transforming the coordinate system by Q^{-1}.

- This is useful because it is often convenient to describe objects within their own "personal" (local) coordinate system.
- These objects may then be placed, at various locations, scales, and orientations, in a global coordinate system.
- The relationship can be described by a transformation $M_{L \leftarrow W}$ that converts values in the world coordinate system into values in the local coordinate system. The inverse transformation $M_{W \leftarrow L}$ is also useful.
- If Q is a transformation that produces some change in an object in the local coordinate system, then $M_{W \leftarrow L} \cdot Q \cdot M_{L \leftarrow W}$ produces the same change in the world coordinate system. This is known as a *similarity transformation*.