2. Transformations

R. Mukundan,
Department of Computer Science and Software Engineering
University of Canterbury.

2.1 Transformations in Graphics

One of the frequently performed operations in a graphics pipeline is a transformation. A point or a vector may undergo several transformations during the rendering of an object. The points that are transformed are primitive vertices and light source positions. The vectors that are transformed are normal vectors of primitives, light's direction vectors and other types of vectors used in the computation of angles. Following are the main stages where transformations are extensively used:

- Modelling: Models are created by creating multiple instances of smaller components (models or primitives) and transforming them relative to each other.
- Scene construction and animation: The construction of a three-dimensional scene and the animation of models within the scene require transformations of objects.
- Viewing: The transformation of a scene relative to the camera's point of view is performed using a special matrix known as the view matrix.
- Projection: The three-dimensional scene contained within the view frustum of the camera is projected on to the near-plane of the camera to obtain a two-dimensional view of the scene. This projection is implemented as a transformation from the camera space (eye coordinate space) to a canonical view volume known as the clip coordinate space.
- Viewport transformation: The two-dimensional coordinates obtained in the previous step are further transformed into pixel coordinates using the display view port's dimensions

2.2 Affine Transformations

A transformation of a point (or a vector) can be viewed as a change of coordinates from (x, y, z) to (x', y', z'). This transformation can be expressed a set of *linear* equations:

$$x' = a_1 x + b_1 y + c_1 z$$
$$y' = a_2 x + b_2 y + c_2 z$$
$$z' = a_3 x + b_3 y + c_3 z$$

where the a_i, b_i, c_i are all some constants. Every linear transformation preserves linearity of line segments, i.e., a set of collinear points will always get transformed to a set of collinear points. A *fixed point* of a transformation is a point, if it exists, that remains unchanged after the transformation. In the above example, the origin (0, 0, 0) is a fixed point. If we add translation components (d_1, d_2, d_3) to the above equation, we get an *affine* transformation:

$$x' = a_1 x + b_1 y + c_1 z + d_1$$
$$y' = a_2 x + b_2 y + c_2 z + d_2$$
$$z' = a_3 x + b_3 y + c_3 z + d_3$$

The affine transformation given above represents the most general form of a transformation. Translations, rotations, and scale transformations are all affine transforms. Let us represent the above equations in a matrix form:

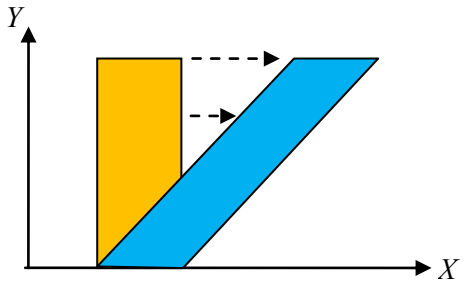
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

If we use homogeneous coordinates, the above general equation for a transformation can be represented as a matrix product, absorbing all transformation parameters in a single 4x4 matrix.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

OpenGL stores all transformation parameters as 4x4 matrices in the above form.

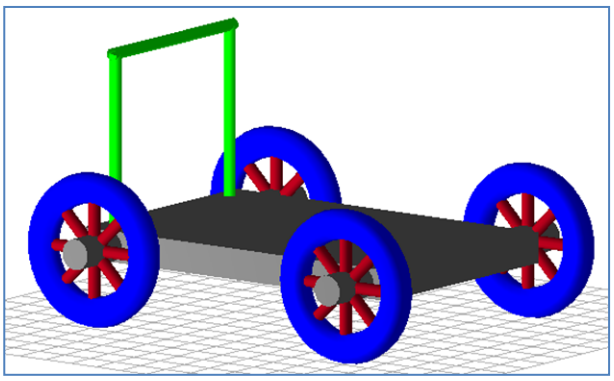
In addition to preserving linearity of line segments, an affine transformation also preserves parallelism between two lines (i.e., a rectangle will only get transformed into a parallelogram). A *shear* transformation is another example of an affine transform. It moves points along a particular axis direction proportional to the point's distance along another axis (eg. $x' = x + ky$). The following figure shows an example of a shear transformation on a 2D plane.



Scale and shear transformations can alter the length of line segments, and also the angle between two lines. Translation and rotation are called *rigid body* transformations as they preserve several attributes such as length of line segments, angle between two lines, area of polygons, and the volume of polyhedral objects. Animation sequences therefore use only translation and rotation, unless the animation involves a change of shape of the object. The scale transformation is usually used only in the modeling phase.

2.3 Hierarchical Transformations

The transformation stack in OpenGL is a very convenient data structure using which we can store and retrieve transformations at different stages of an application. The stack is particularly useful for modeling a composite object, where several component parts are first transformed relative to each other to construct the object, and the object is then transformed as a whole in a scene. Consider the "Cart" model shown in the following figure:



In the "Cart" object we can find parts of objects constructed using multiple copies of sub-parts. The cart itself has 4 wheels, and each wheel has 8 spokes. Such objects can be modeled using nested transformations (nested pushMatrix()-popMatrix() blocks). The hierarchical nature of transformations used in the construction of the cart model is shown below.