

## Geometric Transformations

Eric C. McCreath

School of Computer Science  
The Australian National University  
ACT 0200 Australia

[ericm@cs.anu.edu.au](mailto:ericm@cs.anu.edu.au)

- Basic Geometric Transformation
- Homogeneous Coordinates
- Raster Transformations
- Graphics2D – Affine Transform - Demo

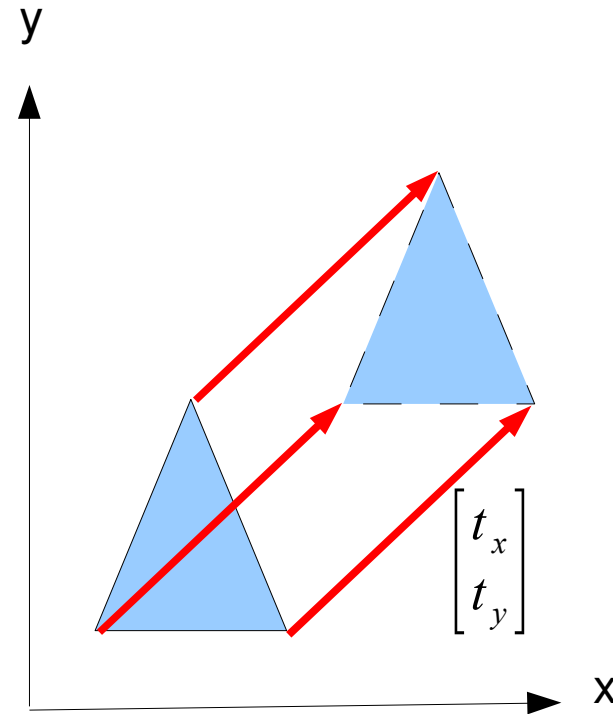
- Transformations are useful for modelling and viewing a scene.
  - Modelling transformations are used to construct a scene.
  - Geometric transformations are used for moving objects within the scene and controlling the viewing of those objects.
- Such transformations are useful and powerful tools for computer graphics applications, however, at times they can be tricky to get working properly.

- Translation provides a simple way of repositioning objects.

New point

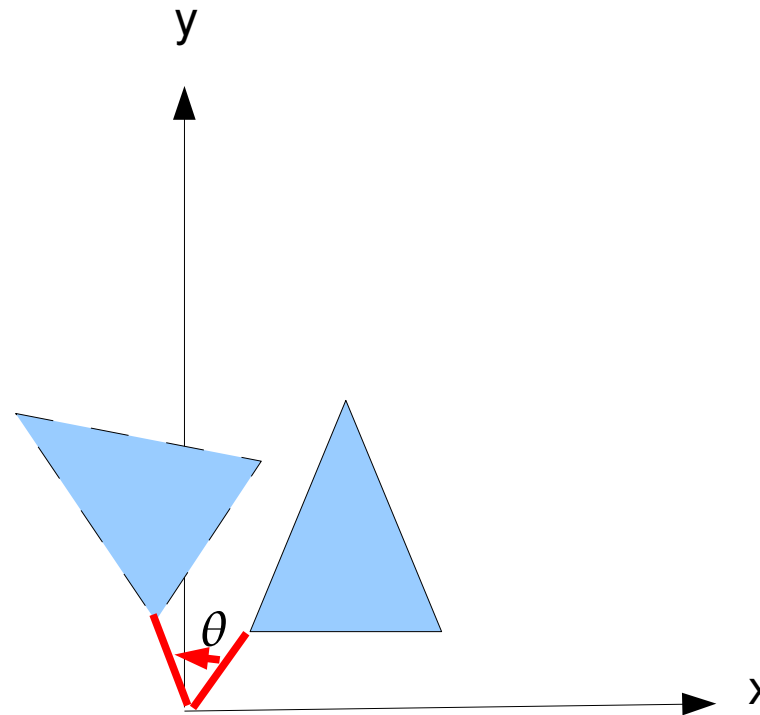
Original point

$$P_n = P_o + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



- Rotation about the origin.

$$P_n = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} P_o$$



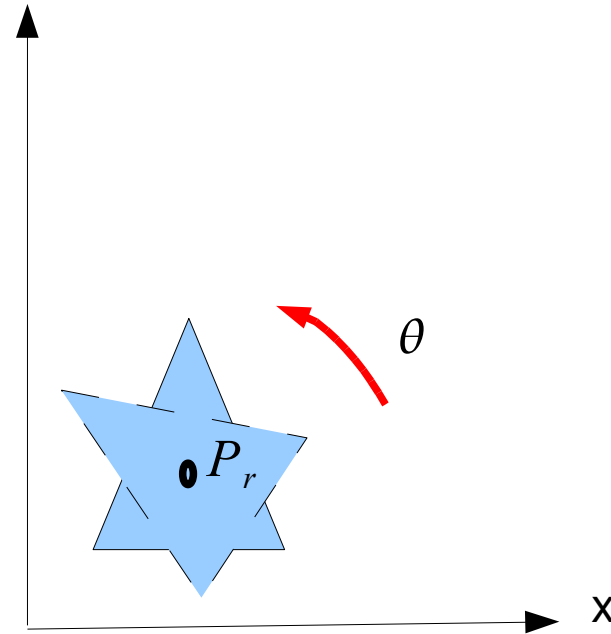
Note, for small theta one can use:

$$\sin \theta \approx \theta$$

$$\cos \theta \approx 1$$

- Rotation about a point  $P_r$ .

$$P_n = P_r + \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} (P_o - P_r)$$



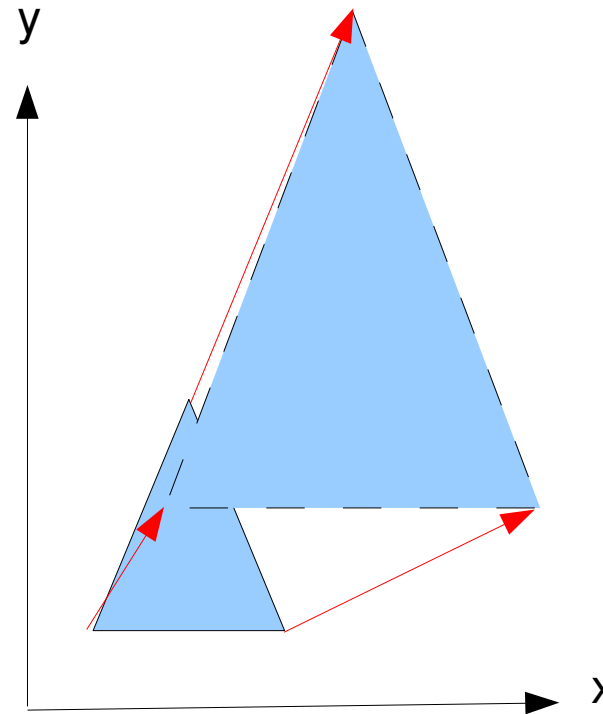
- Scaling from the origin. If scaling values are the same then it is called uniform scaling.

$$P_n = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} P_o$$

- Scaling from a fixed point.

$$P_n = \begin{bmatrix} x_f \\ y_f \end{bmatrix} + \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} (P_o - \begin{bmatrix} x_f \\ y_f \end{bmatrix})$$

$$P_n = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} P_o + \begin{bmatrix} x_f(1-s_x) \\ y_f(1-s_y) \end{bmatrix}$$



Notice how all these transformations (translation, rotation, and scaling) are of the form:

$$P_n = M_1 P_o + M_2$$

- By expanding to a 3x3 matrix we can combine all these transformations into a single matrix multiplication.
- The Cartesian point  $(x,y)$  is represented by the homogeneous coordinate  $(hx, hy, h)$ , where  $h$  is non zero. Often  $h$  is set to 1 so  $(x,y)$  is represented by  $(x,y,1)$ .
- Also the homogeneous coordinate  $(a, b, c)$  represents the Cartesian point  $(a/c, b/c)$ . A single point in Cartesian space is represented by a line in homogeneous space.
- All of the transformations we have looked at can be represented by a single matrix:

Translation

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

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

Scaling

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

- A sequence of transformations can be combined into a single composite transformation matrix.
  - Suppose we wish to conduct the sequence:  $T(5,4)$ ,  $R(0.4)$ ,  $T(-5,-4)$  of transformations on a point  $P_o$ . We could calculate:

$$P_n = T(-5, -4)(R(0.4)(T(5,4)P_o))$$

- this is the same as:

$$P_n = (T(-5, -4)(R(0.4)T(5,4)))P_o$$

- Composite transformations combine as expected:

$$T(a, b)T(c, d) = T(a + c, b + d)$$

$$R(\theta)R(\phi) = R(\theta + \phi)$$

$$S(a, b)S(c, d) = S(ac, bd)$$

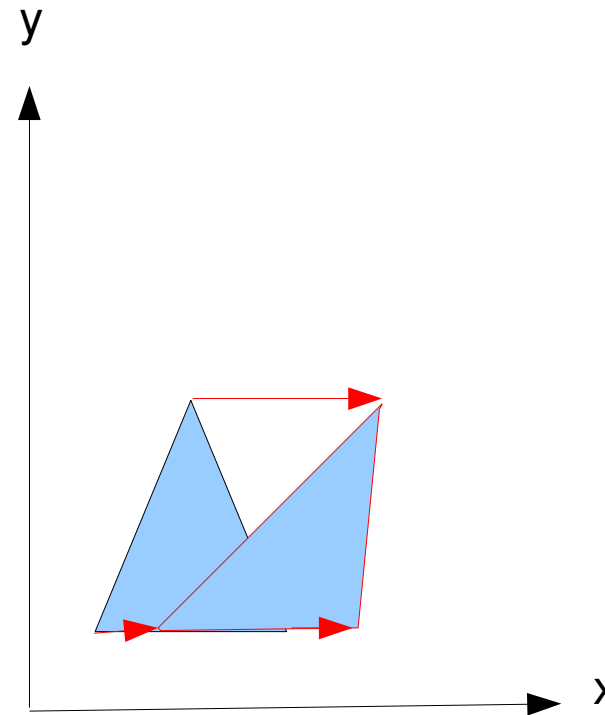
Order is generally important for matrix multiplication!

- Reflection about the x axis:  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Reflection about the y axis:  $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Reflection about both (same as rotation by  $180^\circ$ ) :  $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- How would you reflect about an arbitrary axis??

- A shear will distort the shape of an object.

This is shear along the x-axis:

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



- Geometric transformations can take place directly in the frame buffer. These can be done efficiently by manipulating the array of pixel values.
- 2D block translation can be done by copying blocks of memory (bitblt, block transfer).
- Rotation of a 2D block by  $90^\circ$ ,  $180^\circ$ ,  $270^\circ$  can be done by copying pixels.
- Rotation by other angles can be done by sampling pixel colours.
- Scaling can be also done as a raster operation, this may also require sampling pixel colours.