

# COMP3201 – Computer Graphics

## 2.2 Modelling Transformations in 2D

### 2.2.1 Introduction

The need for modelling transformations arises in several situations. For example, when building a scene it may be necessary for an object to be redrawn in different locations. Sometimes the object itself is made up of the same shape drawn at different locations or at different angles.

Affine transformations are used frequently in Computer Graphics; under an affine transformation, parallel straight lines are transformed into other parallel straight lines. However non-affine transformations can be used for special effects.

The examples that follow are given in 2D as it is easier to visualise (and represent on paper/screen), and the extension to the 3D case is (mostly) straightforward.

Affine transformations are linear; this means that the coordinates of  $Q$  are linear combinations of the coordinates of  $P$ , under an affine transformation.

The four elementary transformations are: translation, scaling, shear and rotation.

Recall that, in order to use a matrix to represent the transformation (in particular translations), we must use homogeneous coordinates. So, in 2D, a point  $\begin{pmatrix} x \\ y \end{pmatrix}$  will be

represented as  $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ .

### 2.2.2 Translation in 2D

This transformation simply shifts each point the required distance in the  $x$ - and/or  $y$ -directions. Thus

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ 0 \end{pmatrix}$$

or, in matrix form,  $Q = TP$ , that is:

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

The OpenGL command for translation in 2D is **glTranslatef( $T_x$ ,  $T_y$ , 0.0f)** where  $T_x$  and  $T_y$  are given in floating point format.

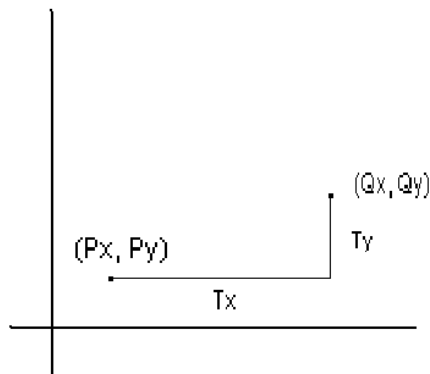


Figure 2.7

### 2.2.3 Rotation in 2D

Suppose that the point  $P = (P_x, P_y)$  is to be rotated through an angle  $\beta$  about the origin to get the new point  $Q = (Q_x, Q_y)$ . (See figure.)

Writing  $P$  in polar coordinates, we see that  $P_x = r \cos \alpha$ ,  $P_y = r \sin \alpha$ .

The new position has coordinates  $Q_x = r \cos(\alpha + \beta)$ ,  $Q_y = r \sin(\alpha + \beta)$ .

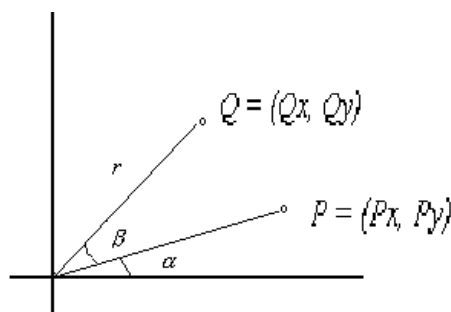


Figure 2.8

Using the trigonometric expansion

$$\cos(A + B) = \cos A \cos B - \sin A \sin B$$

$$\sin(A + B) = \sin A \cos B + \cos A \sin B$$

we can see that

$$\begin{aligned} \begin{pmatrix} Q_x \\ Q_y \end{pmatrix} &= \begin{pmatrix} r \cos \alpha \cos \beta - r \sin \alpha \sin \beta \\ r \sin \alpha \cos \beta + r \cos \alpha \sin \beta \end{pmatrix} \\ &= \begin{pmatrix} P_x \cos \beta - P_y \sin \beta \\ P_y \cos \beta + P_x \sin \beta \end{pmatrix} = \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} P_x \\ P_y \end{pmatrix}. \end{aligned}$$

Writing this in homogeneous coordinates, the Rotation matrix is given by  $Q = R P$ , or

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}.$$

This rotation is about a fixed point, in this case the origin. The 2D rotation can be interpreted in 3D by noting that it is equivalent to three-dimensional rotation about the  $z$ -axis.

A positive value of  $\beta$  corresponds to counter-clockwise rotation when you look down the positive  $z$ -axis.

The OpenGL command for rotation in 2D is **glRotatef(angle, 0.0f, 0.0f, 1.0f);**

#### **2.2.4 Scaling in 2D**

With this transformation, the point  $P$  is mapped onto  $Q$  by applying the corresponding scale factor for each of the  $x$  and  $y$ -direction:

$$Q_x = S_x P_x \quad Q_y = S_y P_y.$$

If the scale factor is greater than 1, then the object's size increases, while if the factor is between 0 and 1, the object shrinks. If the scale factor is less than zero, then the scaling becomes a reflection as well as an adjustment of size.

Writing the points in homogeneous coordinates, the scaling matrix is given by  $Q = SP$ , or

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

#### **2.2.5 Shearing in 2D**

Shearing in the  $x$ -direction is a transformation that shifts each  $x$ -coordinate an amount linearly proportional to the corresponding  $y$ -coordinate. The shear-in- $x$  matrix would be

$$\begin{pmatrix} 1 & S_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

for some value  $S_x$ . Similarly for shearing in the  $y$ -direction, where the shear-in- $y$  matrix is

$$\begin{pmatrix} 1 & 0 & 0 \\ S_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

for some value  $S_y$ . It results in the object being slanted (for example, italic writing).

#### **2.2.6 Inverse Affine Transformations (2D case)**

It is often necessary to undo the effect of a transformation and for this we need to be able to apply the inverse of the transformation.

Recall that the inverse of a matrix  $M$  can be formed if  $\det(M) \neq 0$ ; also recall the matrices  $T$ ,  $R$  and  $S$  from above, for the Translation, Rotation and Scaling transformations.

Then  $\det(T) = 1$ ,  $\det(R) = 1$ ,  $\det(S) = S_x S_y$ .

The inverse Translation transformation is

$$T^{-1} = \begin{pmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{pmatrix},$$

namely the same translation but in the opposite direction.

**Exercise:** Verify that  $TT^{-1} = T^{-1}T = I$ .

The inverse of a scale transformation requires the scale factors to be inverted and reapplied. So the inverse scale transformation matrix is

$$S^{-1} = \begin{pmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

**Exercise:** Verify that  $SS^{-1} = S^{-1}S = I$ .

To undo the rotation transformation, we need to rotate in the opposite direction (that is, rotate for an angle  $-\beta$ ). Consequently, the inverse rotation transformation matrix is

$$R^{-1} = \begin{pmatrix} \cos(-\beta) & -\sin(-\beta) & 0 \\ \sin(-\beta) & \cos(-\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The properties of  $\cos$  and  $\sin$  can be applied here:

$$\cos(-\beta) = \cos \beta, \quad \sin(-\beta) = -\sin \beta$$

and so we can write

$$R^{-1} = \begin{pmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

**Exercise:** Verify that  $RR^{-1} = R^{-1}R = I$ .

### 2.2.7 Composition of Affine Transformations

Often there are a combination of transformations to be applied, and this can be achieved by multiplying together all the transformation matrices and then applying the product matrix.

Note how the order of transformation matrices must be applied. For example, suppose we wish to map  $P$  to  $Q$  using translation matrix  $T_1$ , and then map  $Q$  to  $U$  using translation matrix  $T_2$ .

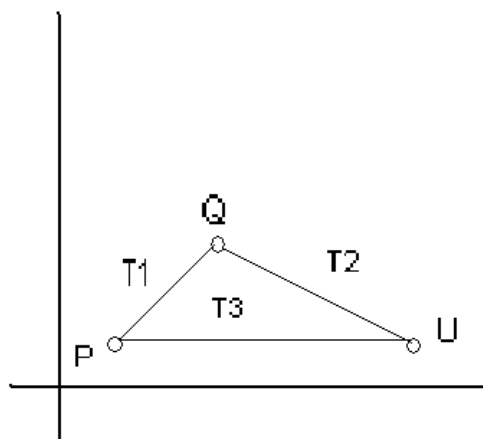


Figure 2.9

Then  $U = T_2 Q = T_2 (T_1 P)$ , so we can map  $P$  directly to  $U$  using the concatenation (or composition) of  $T_1$  and  $T_2$ , namely  $T_3 = T_2 T_1$ .

Note that the composition of affine transformations is also an affine transformation.

We can compose any combination of transformations. For example, suppose we have a triangle with vertices at  $\{(0.0, 1.0), (-1.0, 0.0), (1.0, 0.0)\}$ , and we wish to rotate the triangle 60 degrees, then translate it 2 units in the  $x$ -direction and  $-1$  in the  $y$ -direction, and finally scale it by a factor of 3 in the  $x$ -direction. For this example,

$$R = \begin{pmatrix} \cos 60 & -\sin 60 & 0 \\ \sin 60 & \cos 60 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.5 & -0.866 & 0 \\ 0.866 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$T = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}, \quad S = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Figure 2.10 shows the effect of each transformation. Indeed the final triangle positions can be calculated by applying the matrix product  $S T R$  to the original triangle vertices:

$$STR = \begin{pmatrix} 2 & 0 & 4 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & -0.866 & 0 \\ 0.866 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1.732 & 4 \\ 0.866 & 0.5 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

and so the transformed triangle vertices are given by the following matrix product

$$\begin{pmatrix} 1 & -1.732 & 4 \\ 0.866 & 0.5 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2.268 & 3 & 5 \\ -0.5 & -1.866 & -0.134 \\ 1 & 1 & 1 \end{pmatrix}.$$

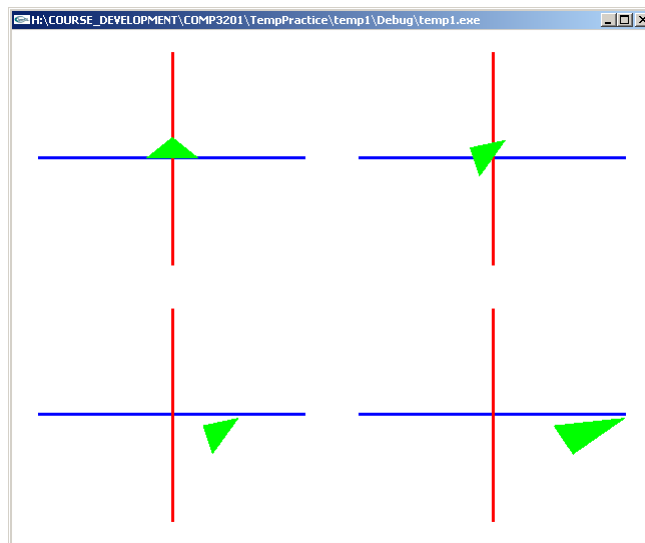


Figure 2.10

### Rotation about an arbitrary point

This is an example where composition of transformations is required. If the rotation is to be about an arbitrary point  $U$  rather than the origin, then the following steps are required: (see the figure)

- apply a translation  $-U_x, -U_y$  in the  $x$ - and  $y$ -directions respectively
- apply a rotation through angle  $A$
- apply the inverse translation  $U_x, U_y$  in the  $x$ - and  $y$ -directions.

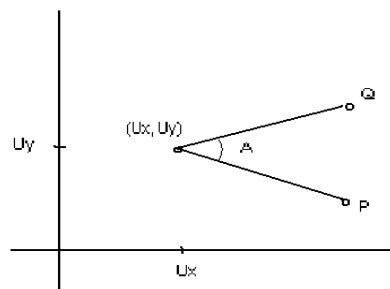


Figure 2.11