2D Transformations

- There are 3 types of transformations that are essential.
  - Translation
    - Moving an object to a new position by adding to the object x and y coordinates.
  - Scaling
    - Changing the size of the object. This can be uniform where both dimensions are resized by the same factor, or non-uniform.
  - Rotation
    - Object is rotated around the origin by a specified angle.

2D Translation – Approach A

- We want to move a point *p* at (x,y) to a new position p' at (x', y') where

  $x' = x + d_x$ and $y' = y + d_y$

- We can represent the points and the translation as column vectors.

$$p = \begin{bmatrix} x \\ y \end{bmatrix}, p' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- The translation can be expressed as.

$$p' = p + T$$

- A line or shape can be translated by translating its vertices and redrawing the line or shape.

2D Scaling – Approach A

- Scaling refers to rescaling along an axis. A point can be scaled along the x-axis or the y-axis, or both.

  $x' = s_x.x \qquad y' = s_y.y$

- In matrix form this is:

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

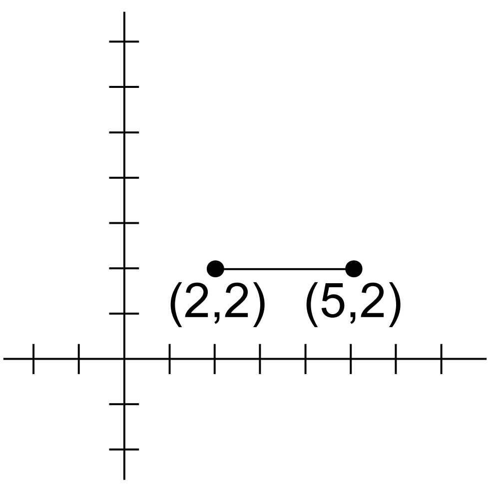
- or

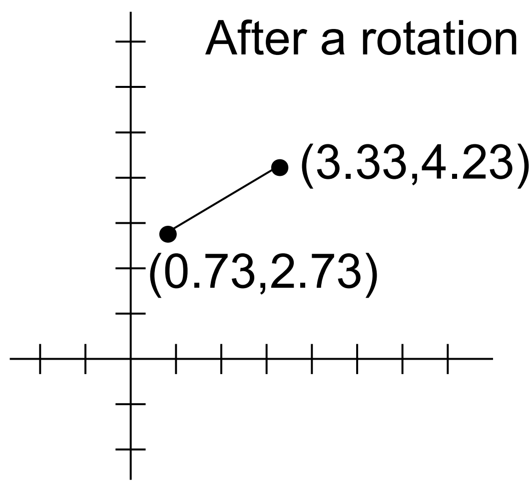
  $p' = S.p$

  where S is the scaling matrix

2D Rotation – Approach A

- Points are rotated around the origin by an angle θ.





$$x' = x.cos(\theta) - y.sin(\theta),\ y' = x.sin(\theta) + y.cos(\theta)$$

- In matrix form:

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

or

$$p' = R.p$$

Homogeneous Coordinates

- The problem with Approach A is that different transformations are handled differently.
  - $p' = p + T$
  - $p' = S.p$
  - $p' = R.p$
- The solution is to use *homogeneous coordinates*.
- Each point (x, y) is represent as a triple (x, y, W).
  - Two points are the same if one is a multiple of the other.
    - (1,2,3) and (3,6,9) represent the same point.
  - Each 2D point is now a line in a 3D space. The W = 1 plane is our 2D space and the intersection of the line with this plane gives us the point.
- In a homogeneous coordinate system translation, scaling and rotation are matrix multiplications.

Homogeneous Transformations

- Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Scaling

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

- Rotation

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

Homogeneous Transforms (contd.)

- The general form the 2D transformation matrix is:

$$\begin{bmatrix} r_{11} & r_{12} & d_x \\ r_{21} & r_{22} & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

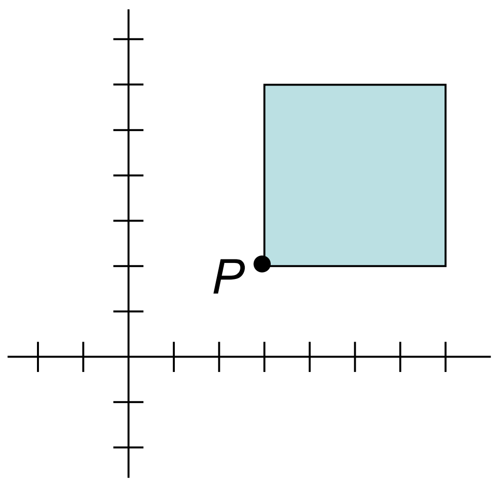
- Multiplying a set of transformation matrices is equivalent to applying a sequence of transformations one after another.
  - The order is **very important**.
  - If you want to apply transformation A then B and finally C, you can pre-calculate the combined effect as

    C.B.A

  - The resulting transformation is an *affine* transformation.
    - Preserves parallel lines, but not lengths or angles

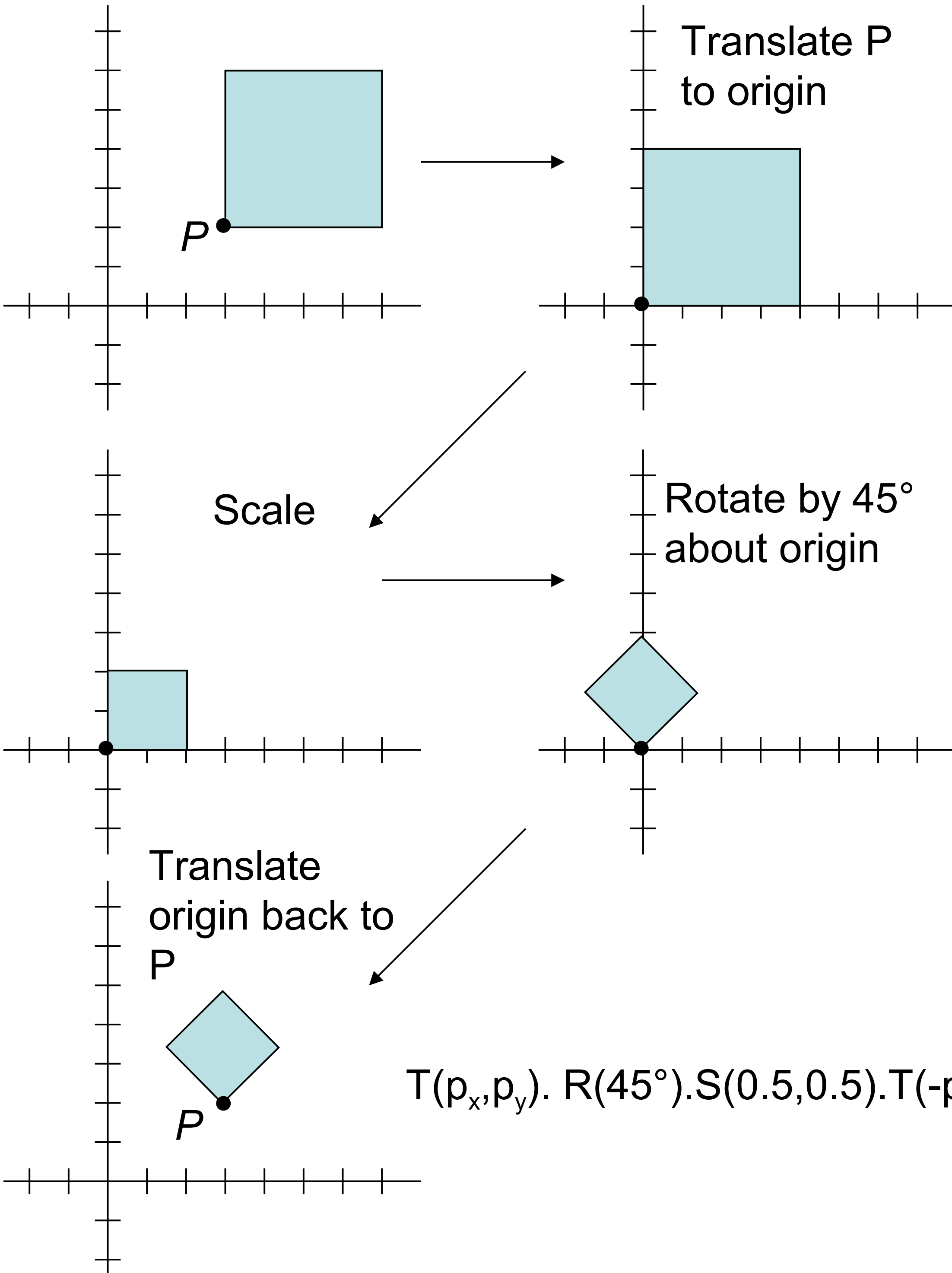
An Example

- Suppose we wish to reduce the square in the following image to half its size and rotate it by 45° about point *P*.



- We need to remember that that scaling and rotation takes place with respect to the origin.
  - Translate square so that the point around which the rotation is to occur is at the origin.
  - Scale
  - Rotate
  - Translate origin back to position *P*.

An Example (contd.)



$T(p_x,p_y).\ R(45°).S(0.5,0.5).T(-p_x,-p_y)$

Efficiency

- Because of the particular structure of the last row of the 2D homogeneous transformation matrix the number of operations can be reduced from 9 multiplications and 6 additions to 4 multiplications and 4 additions.

  $x' = x.r_{11} + y.r_{12} + t_x$

  $y' = x.r_{21} + y.r_{22} + t_y$

- If a model is being incrementally rotated by a small angle θ, then by noting that cosθ is very close to 1 for small values of θ then

  $x' = x - y.\sin\theta$

  $y' = x.\sin\theta + y$

  just 2 multiplications and 2 additions.
  - This is just an approximation and as the errors accumulate the image will become unrecognisable.
  - A better approximations is

    $x' = x - y.\sin\theta$

    $y' = x'.\sin\theta + y = (x - y.\sin\theta).\sin\theta + y$

    $= x.\sin\theta + y(1-\sin^2\theta)$

    - The corresponding 2x2 matrix has a determinant of 1 and hence preserves areas.

3D Homogeneous Transformations

- Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Scaling

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

- Rotation about z axis

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

3D Homogeneous Transformations (contd.)

- Rotation about x axis

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

- Rotation about y axis

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

- General form of a composition of transformations

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

Composing 3D Homogeneous Transforms

- Same approach as when composing 2D homogeneous transformations.
- Rather than composing each rotation around the x, y, and z axis separately we can sometimes make use of the vector cross product and the following feature of a composite matrix.
- The composite rotation matrix applied to the unit vector along the x, y and z axes are:

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

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

Composing 3D Rotations

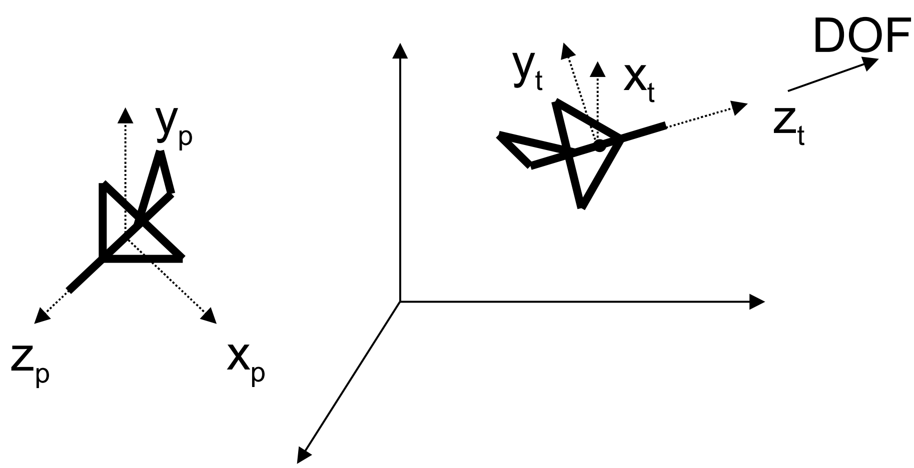
- The first, second and third columns of the upper-left 3x3 submatrix are the rotated x-axis, rotated y-axis and rotated z-axis respectively.

$$\begin{bmatrix} R_x & R_y & R_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Two non colinear vectors define a plane.
- The vector cross product v_1 x v_2 is a vector at right angle to the plane defined by v_1 and v_2.
- If either R_x, R_y or R_z is known and some relationship between an element of the rotated object and the x, y or z axis, then the computation of the composed rotation matrix can be simplified.

Example

- We want to position the following plane at location *P* with a specified direction of flight (DOF) and no bank angle (wings parallel x-z plane).



- z_p is transformed to z_t = DOF.
- $x_t = y \times z_t$ - no bank angle
- $y_t = z_t \times x_t$ = DOF $\times$ ($y \times$ DOF)

Example (contd.)

$$R = \begin{bmatrix} (y \times \vec{DOF}) & (\vec{DOF} \times (y \times \vec{DOF})) & \vec{DOF} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This general approach has limitations. In this case if the DOF is colinear with the *y* axis, the (*y* **x** DOF*)* is zero.
- This general approach can be used when instantiating models (specified in a model coordinate system) in our world coordinate system.

The inverse of a Rotation Matrix

- Suppose we wish to undo a rotation
- e.g. suppose we wish to rotate the aeroplane in the previous example back to its original orientation along the x,y and z axes
- We need to find the inverse of the previous rotation matrix

Inverse of a Rotation Matrix

- The inverse of a rotation matrix is just its transpose
- You get the transpose by flipping the matrix about the leading diagonal
- In other words you interchange the rows and columns

Summary of Rotation Matrices

- If you want to rotate an object from axes x,y,z to orientation u,v,n you use the matrix on the right
- If you want to rotate an object from orientation u,v,n back to axes x,y,z you use the matrix on the left

$$\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$