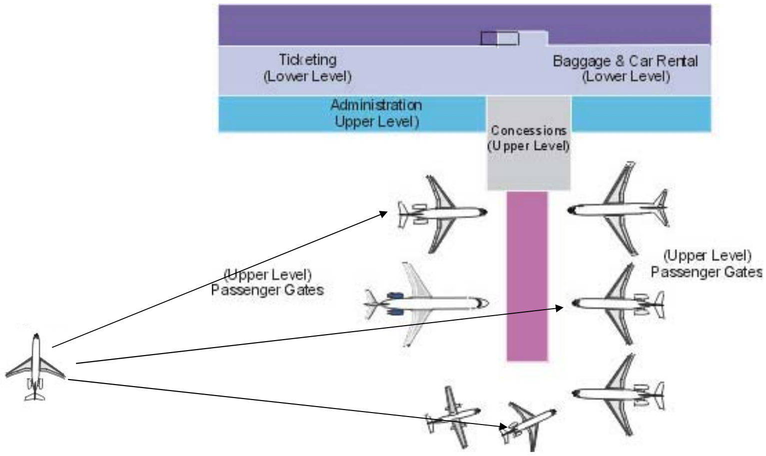
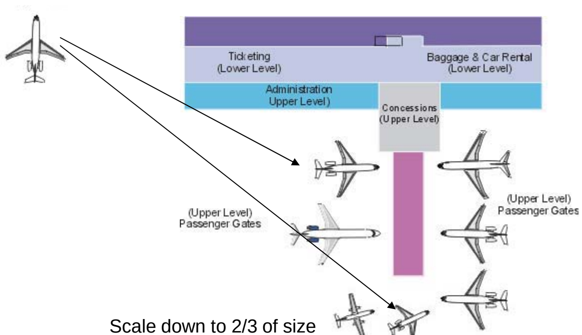
Three dimensional transformations

Object Manipulation

- Graphics primitives allows us to draw a variety of pictures, images and graphs
- In many application there is a also need to manipulate the displays
  - Rearranging the display
  - Changing the size of the different objects
- This is accomplished by using geometric transformations



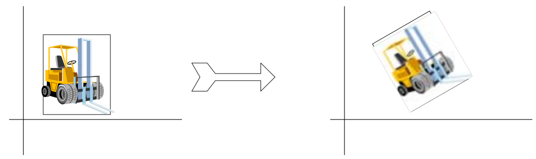
Rotate right by 90º
Translate to position



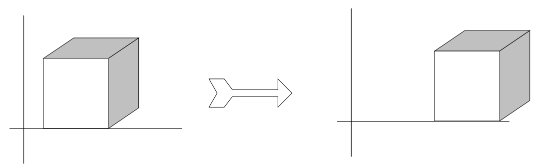
Scale down to 2/3 of size
Rotate right by 60º
Translate to position

Basic Geometric Transformation

- The basic geometric transformation are:
  - Translation
  - Scaling
  - Rotation

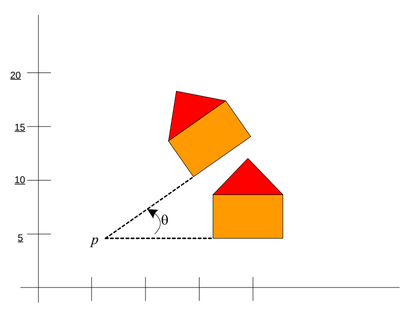


- Other
  - Reflection
  - Shear



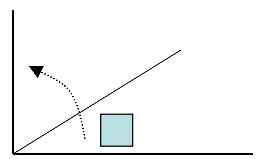
Rotation

- Rotation of an object is repositioning it along a circular path in the *xy* plane.
- A rotation is specified by a rotation angle θ and a pivot point (rotation point) p

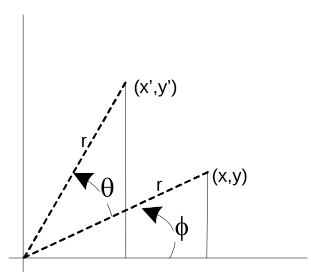


2D Rotation

- The angle is defined in a counter clockwise order around the point p (counter clockwise is positive angle and clockwise is negative angle)
- The rotation can also be viewed as rotation around an axis line perpendicular to the *xy* plane through the point *p*



- Assuming rotation around the origin (0,0)



$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

Since

$x = r\cos\phi$ and $y = r\sin\phi$

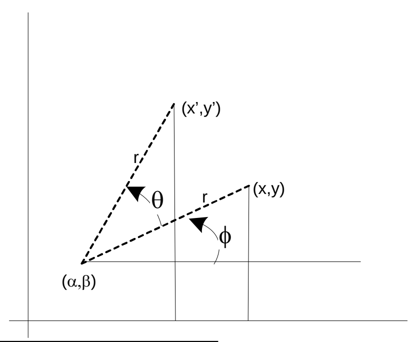
we obtain

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

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

Rotation around point *p*

$$x = \alpha + r\cos\phi \quad \Rightarrow r\cos\phi = x\text{-}\alpha$$
$$y = \beta + r\sin\phi \quad \Rightarrow r\sin\phi = y\text{-}\beta$$
$$x' = \alpha + r\cos(\phi+\theta)$$
$$= \alpha + r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$= \alpha + (x\text{-}\alpha)\cos\theta - (y\text{-}\beta)\phi\sin\theta$$
$$y' = \beta + (x\text{-}\alpha)\sin\theta + (y\text{-}\beta)\cos\theta$$



$$\boxed{(x', y') = (x, y)\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} + (\alpha, \beta)}$$

Matrix Transforms

- For "simplicity", we want to represent our transforms as matrix operations
- Why is this simple?
  - single type of operation for all transforms
  - IMPORTANT: collection of transforms can be expressed as a single matrix
    - efficiency

Homogenous Coordinates Motivation

- In many cases, e.g., animation, we need to combine a number of basic transformations such as a rotation followed by a translation

$$P' = RP + T$$

- R is a 3x3 matrix while T is a 3x1 matrix
- The rotation operation is multiplication operation while the translation operation is an addition

Homogenous Computations

- How to speed up the process and to avoid long processing (recall that there can be thousands of vertices in an image)?
- Idea:
  - use a single matrix operation for all types of transformations (rotation, scaling, translation)

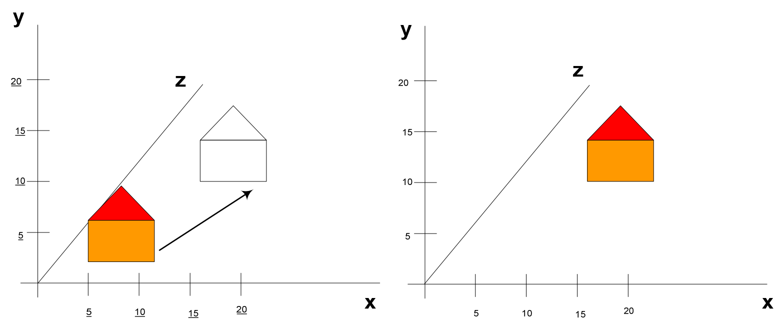
- Instead of using triplet of coordinates (x, y, z) for each vertex use a (x_w, y_w, z_w, w) where

$x = \frac{x_w}{w}, \quad y = \frac{y_w}{w}, z = \frac{z_w}{w}$ which leads to $\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} xw \\ yw \\ zw \\ w \end{bmatrix}$

- There are many choices for w (normally w=1)

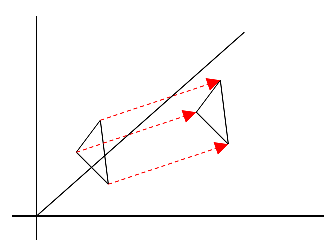
Translation

- Purpose: to move an object along a line from one location to another location



Translation Matrix

$$(x', y', z', 1) = (x, y, z, 1)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

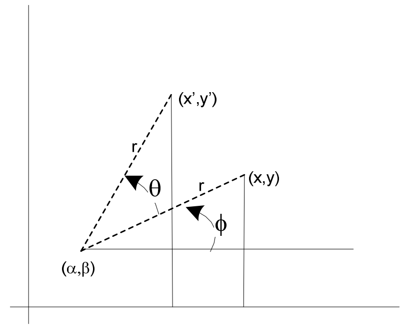


$$\Delta' = \{ p + t \mid p \in \Delta \}$$

- This is a rigid transformation - objects are not deformed.
- Enough to translate the vertices (end points)

Rotation around point *p*
2D Homogenous Coordinates

$$
\begin{aligned}
x &= \alpha + r\cos\phi \quad \Rightarrow r\cos\phi = x\text{-}\alpha \\
y &= \beta + r\sin\phi \quad \Rightarrow r\sin\phi = y\text{-}\beta \\
x' &= \alpha + r\cos(\phi+\theta) \\
&= \alpha + r\cos\phi\cos\theta - r\sin\phi\sin\theta \\
&= \alpha + (x\text{-}\alpha)\cos\theta - (y\text{-}\beta)\phi\sin\theta \\
y' &= \beta + (x\text{-}\alpha)\sin\theta + (y\text{-}\beta)\cos\theta
\end{aligned}
$$



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

3D Rotation Matrices

Rotation around **Z** axis

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

Rotation around **Y** axis

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

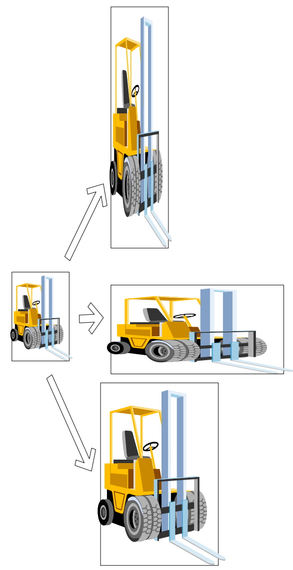
Rotation around **X** axis

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

Scaling

- A scaling transformation alters the size of an object.
- The operation is accomplished by multiplying each coordinate by scaling factors s_x, s_y s_z :
- When s_x (s_y, s_z) > 1 we enlarge the object
- When s_x (s_y, s_z) < 1 we reduce the object

$$(x', y', 1) = (x, y, 1)\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D Matrices

- Pay attention to:
  - Concatenating matrices (e.g., rotation, translation) to which side you should concatenate.
  - Note Direct3D multiplies the vector on the left side (V^T*M) vs. open GL which multiplies the vector on the right hand side similar to what is done in mathematics (M*V), where V is a 4x1 vector and M is a 4x4 matrix.
  - The difference in the sign before the sin(θ) when the rotation is around the y-axis vs. rotation around the x-axis and z-axis

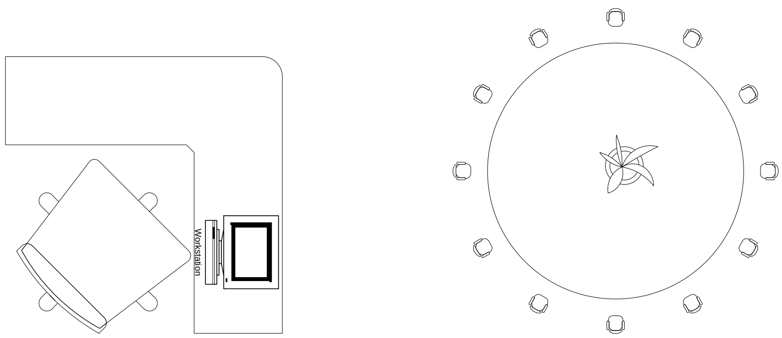
Compound Objects Concatenation

- Done on the board in class

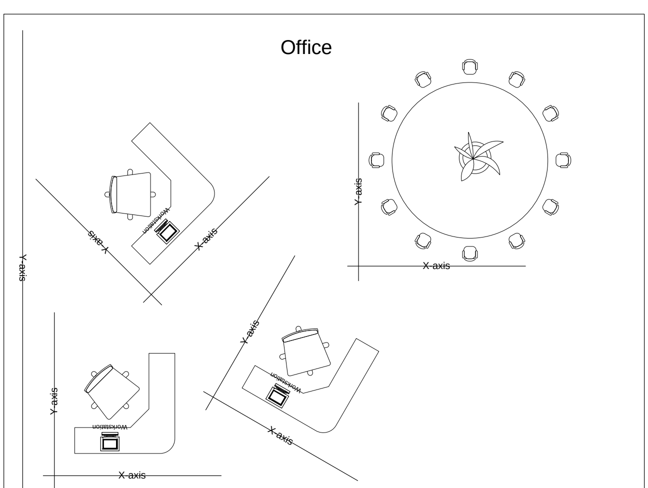
Transformation between coordinate systems

- Motivation:
  – objects are often defined within their own local coordinate systems which must be converted into a scene coordinate system.
  – Often occurs when one buys 3D models from a modeller
- Example
  – For example a cubical consists of a desk, a workstation and a chair is defined in a local coordinate system. When an office is designed the layout consists of multiple cubicles which are positioned in the office. The location of the desks, chairs and workstations must be defined using with respect to the office

Office Elements

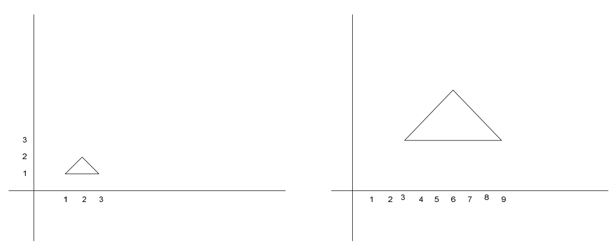


Office Layout



- We want to know what will be the coordinates of p=(x,y,z) in the new coordinate system

- In order to accomplish this we need to
  - translate the new coordinate system to the origin of the current coordinate system
  - Rotate the new coordinate system so that the axis overlap
  - Use the two transformation matrices to transform the object to the new coordinate system



Scaling using $s_x=s_y=3$

- Uniform scaling $s_x=s_y$
- Differential scaling $s_x \neq s_y$

- Transformed objects using the matrix *S* are not only scaled but also change their position

- Show example

Composite Transformation

- What happens when we combine several transformations (concatenation/composition)
  - compute the product of two or more matrices to obtain the final transformation matrix

$$
\begin{aligned}
P' &= (((PM_1)M_2)M_3) \\
&= P((M_1M_2)M_3) \\
&= P(M_{12}M_3) \\
&= PM_{1-3}
\end{aligned}
$$

Concatenation of Similar Transformations

- Translation

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

Hence,

$$T(t_{x2}, t_{y2})T(t_{x1}, t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$$

- Rotation

$$\begin{aligned} P' &= R(\theta_2)\{R(\theta_1)P\} \\ &= \{R(\theta_2)R(\theta_1)\}P \end{aligned}$$

Show a picture of two rotations

Hence,

$$R(\theta_2)R(\theta_1) = R(\theta_2+\theta_1)$$

- Scaling

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x2}s_{x1} & 0 & 0 \\ 0 & s_{y2}s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

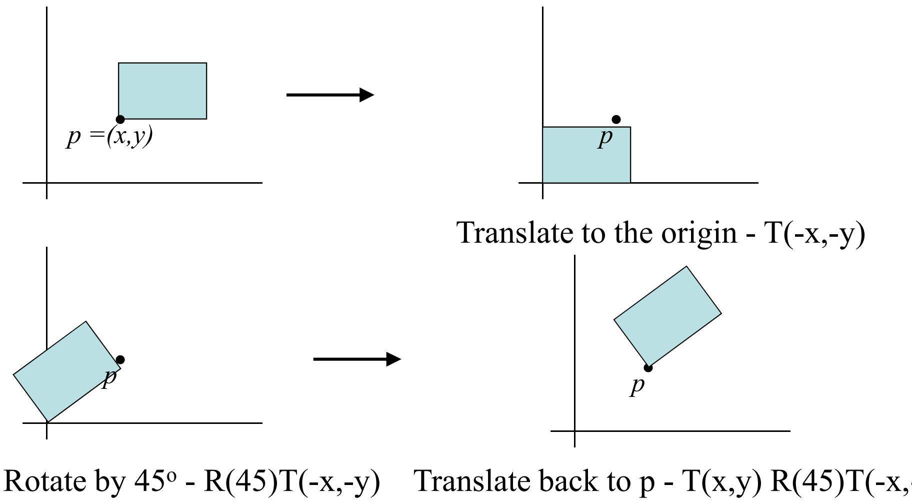
Hence,

$$S(s_{x2}, s_{y2})S(s_{x1}, s_{y1}) = S(s_{x1}s_{x2}, s_{y1}s_{y2})$$

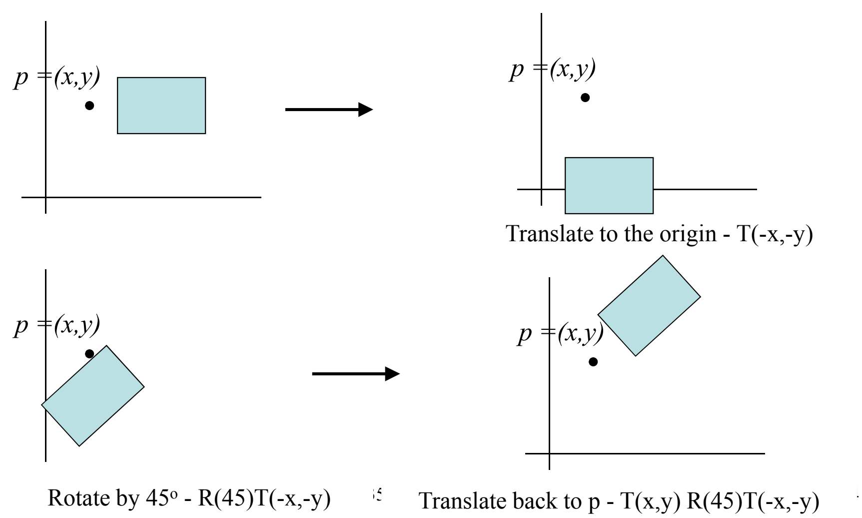
Pivot Point Rotation

- So far all the transformations assumed that they are done around the origin (0,0)
- This can be generalized to any arbitrary point as follows
  - Translate the object to the origin such that the pivot point is at the origin
  - Transform the object around the origin (e.g., rotation)
  - Use inverse translation used in step 1 to move the object back to its original position

Example Rotate by 45^o around *p*



Example Rotate by 45^o around *p*



2D example

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

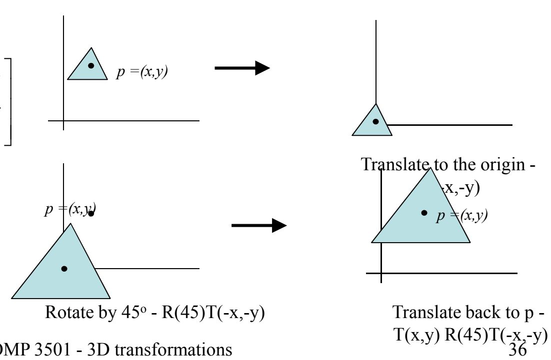
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling with respect to a fixed point

- Similar procedure is used to scale with respect to a fixed point

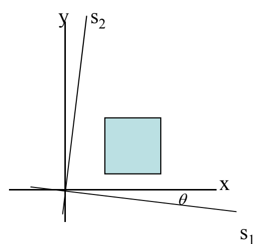
$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & x_r(1-S_x) \\ 0 & S_x & y_r(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$



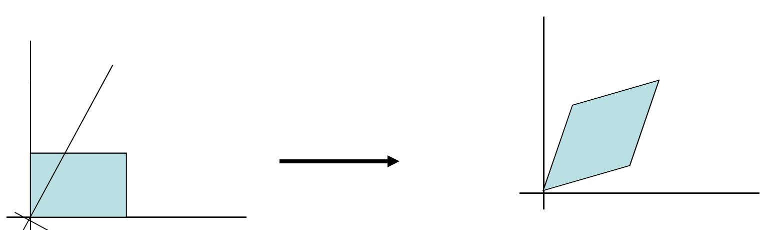
General Scaling - scaling with respect to a line

- The scaling matrix allows only scaling along the x-axis and the y-axis.
- How to scale with respect to any line?



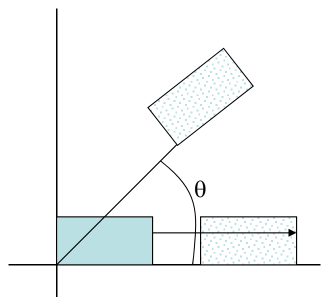
General Scaling

- We use a similar technique to use any scaling in any direction.
  - Rotate the object such that the scaling direction coincide with the x-y axis.
  - Scale the object
  - rotate back

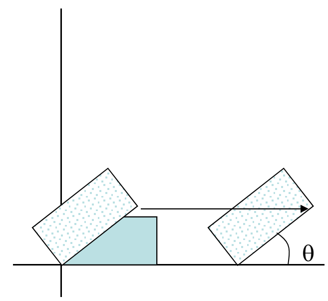


Scale at θ = 45° and s_x=1 s_y=2

Transformation are not commutative



Translate and then rotate - R(θ)T(x,y)
e.g. R(45)T(5,0)



Rotate and then Translate - T(x,y) R(θ)
e.g. T(5,0) R(45)

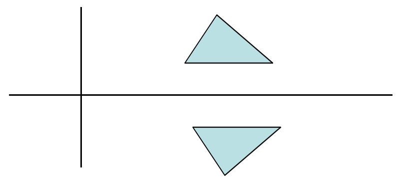
Other Transformations

- Reflection
  - A transformation that produces a mirror image of an object
  - it is generated by rotating the object 180^{o} around the axis of reflection

Reflection about the axis (x-axis or y-axis)

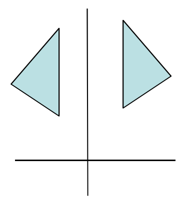
- a reflection about the x-axis

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



- a reflection about the y-axis

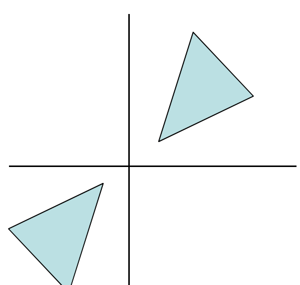
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Reflection about the origin (x-axis and the y-axis)

- Show a picture of reflection about the x-axis and y-axis

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



- This transformation can also be viewed as a rotation where $\theta = 180^o$

Reflection about lines

- Line y=x

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

- It can be viewed as
  - a rotation by -45^o (CCW 45^o)
  - Reflection about the x-axis
  - a rotation by 45^o (CW 45^o)

Reflection about a general line

- Reflection about a general line y=ax+ b is achieved as:
  - translate the object and the line such that the line passes through the origin - T(0,-b)
  - rotate the object and the line such that it coincide with one of the axis
  - apply the reflection transformation of that axis
  - rotate the object and the line back using the inverse rotation
  - translate the object and the line back using inverse transformation

Shear

- Shear transformation distorts the shape of the object.
- The effect causes the object to be pushed to one side as if it was constructed of layers that slide on top of each other

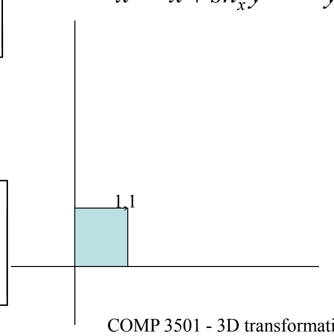
- (show picture)

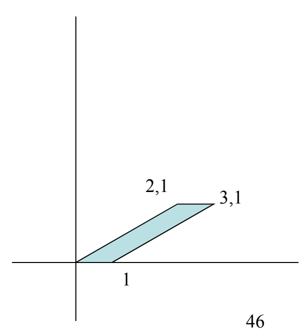
- Shearing transformation along the x-axis

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad x' = x + sh_x y \qquad y' = y$$

- example

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





- Shearing transformation along the x-axis using a reference line

$$\begin{bmatrix} 1 & sh_x & -sh_x y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad x' = x + sh_x(y - y_{ref}) \qquad y' = y$$

- example using a shearing along the x-axis of ½ and a reference line y=-1

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$