2D Transformations

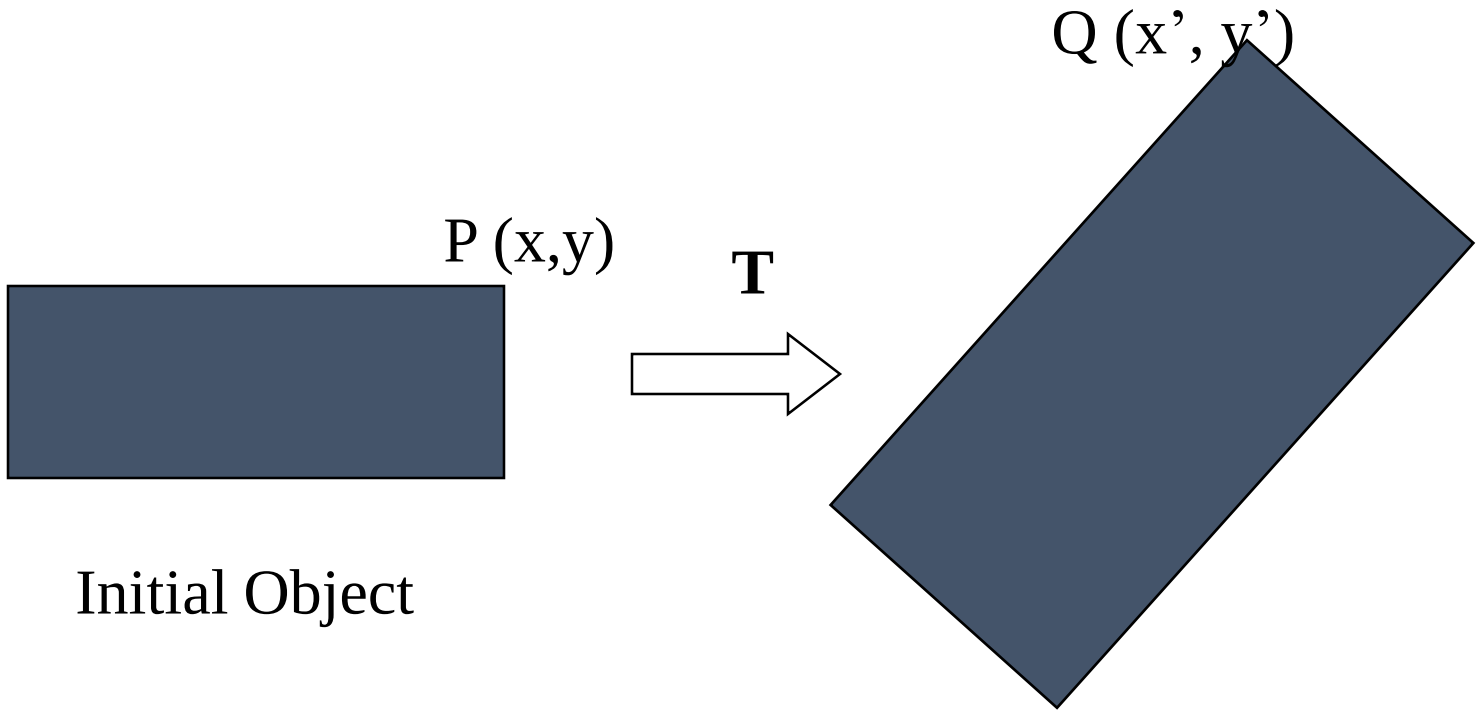
Transformation

- What are they?
  - Changing something to something else via rules.
  - Mathematics: mapping between values in a range set and domain set (function/ relation)
  - Geometric: translate, rotate, scale, shear…

- Why are they important to graphics?
  - Moving objects on screen/ in space
  - specifying the camera's view of a 3D scene
  - Mapping from model space to world space to camera space to screen space
  - Specifying parent/child relationships

Transformation

Transform every point on an object according to certain rule.



$x \rightarrow x'$
$y \rightarrow y'$

The point Q is the image of P under the transformation T.

Transformations

Basic 2D Transformations:

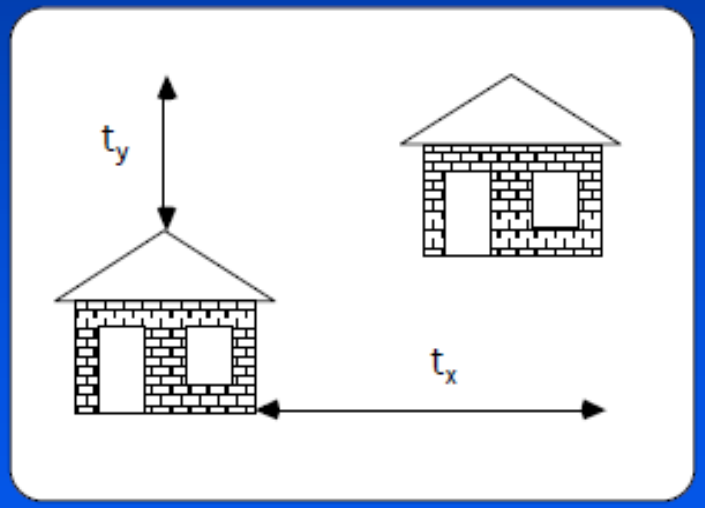
- **Translation**
- **Scaling**
- **Rotation**

Other 2D Transformations:

- **Reflection**
- **Shearing**

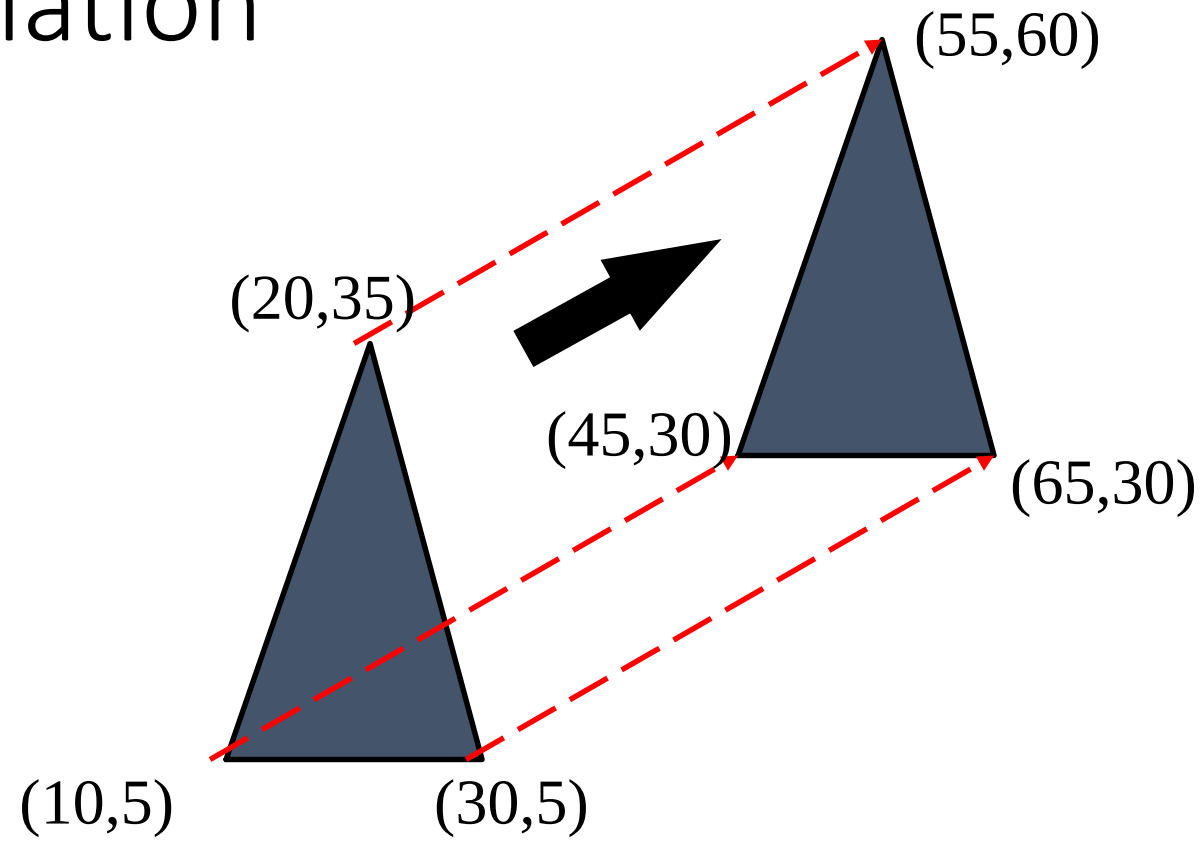
Translation

Moving an object is called a translation. We translate a point by adding to the x and y coordinates, respectively, the amount the point should be shifted in the x and y directions. We translate an object by translating each vertex in the object.



$$x_{new} = x_{old} + t_x; \; y_{new} = y_{old} + t_y$$

Translation



$$x' = x + t_x$$

$$y' = y + t_y$$

The vector (t_x, t_y) is called the *offset vector*.

Translation (OpenGL)

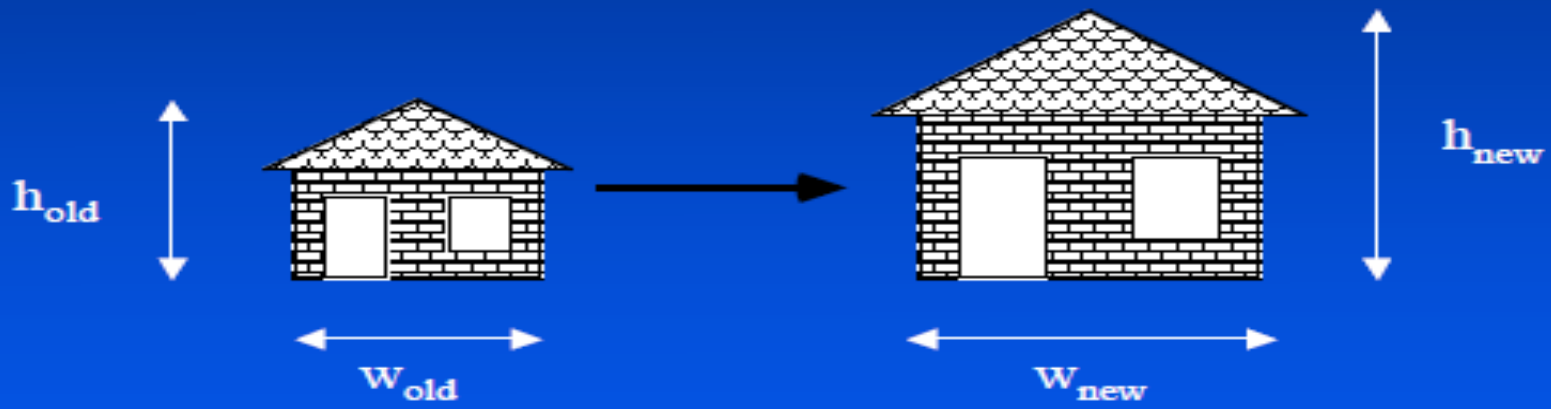
Specifying a 2D-Translation:

```
glTranslatef(tx, ty, 0.0);
```

(The z component is set to 0 for 2D translation).

Scaling

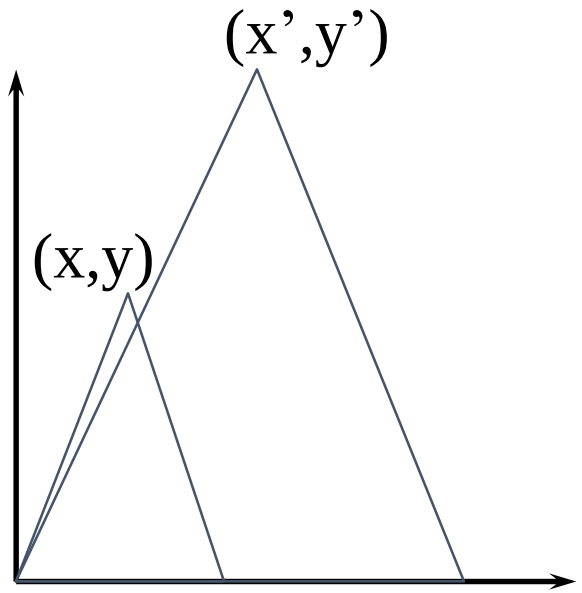
Changing the size of an object is called a scale. We scale an object by scaling the x and y coordinates of each vertex in the object.

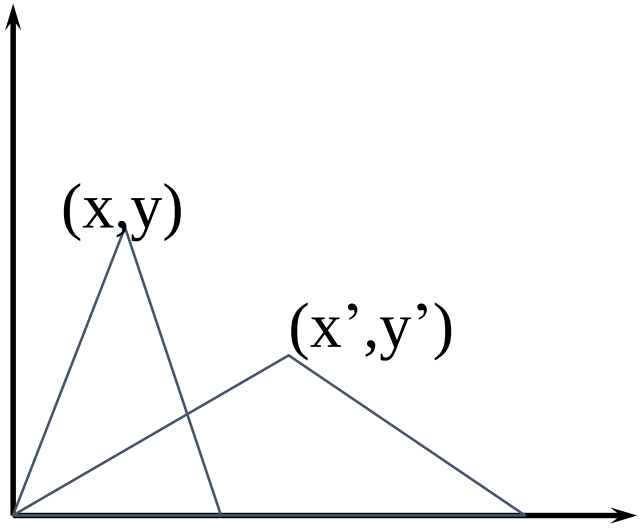


$$s_x = w_{new}/w_{old} \qquad s_y = h_{new}/h_{old}$$

$$x_{new} = s_x x_{old} \qquad y_{new} = s_y y_{old}$$

Scaling About the Origin





$$x' = x.s_x$$

$$y' = y.s_y$$

Uniform

$s_x = s_y$

Non-Uniform

$(s_x, s_y > 0)$

The parameters s_x, s_y are called *scale factors*.

$$s_x \neq s_y$$

Scaling About the Origin(OpenGL)

Specifying a 2D-Scaling with respect to the origin:

```
glScalef(sx, sy, 1.0);
```

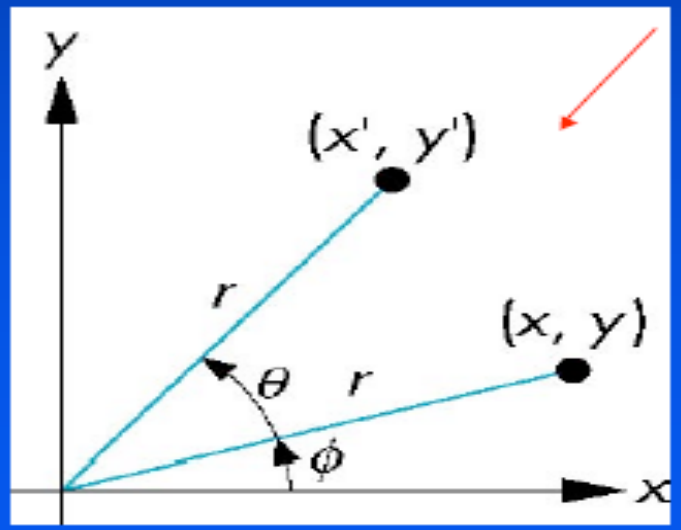
`sx, sy`: Scale factors along x, y.
For proper scaling `sx, sy` must be positive.
For 2D scaling, the third scale factor must be set to 1.0.

Rotation about the origin

Consider rotation about the origin by Θ degrees

- radius stays the same, angle increases by Θ

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



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

Rotation about the origin (cont.)

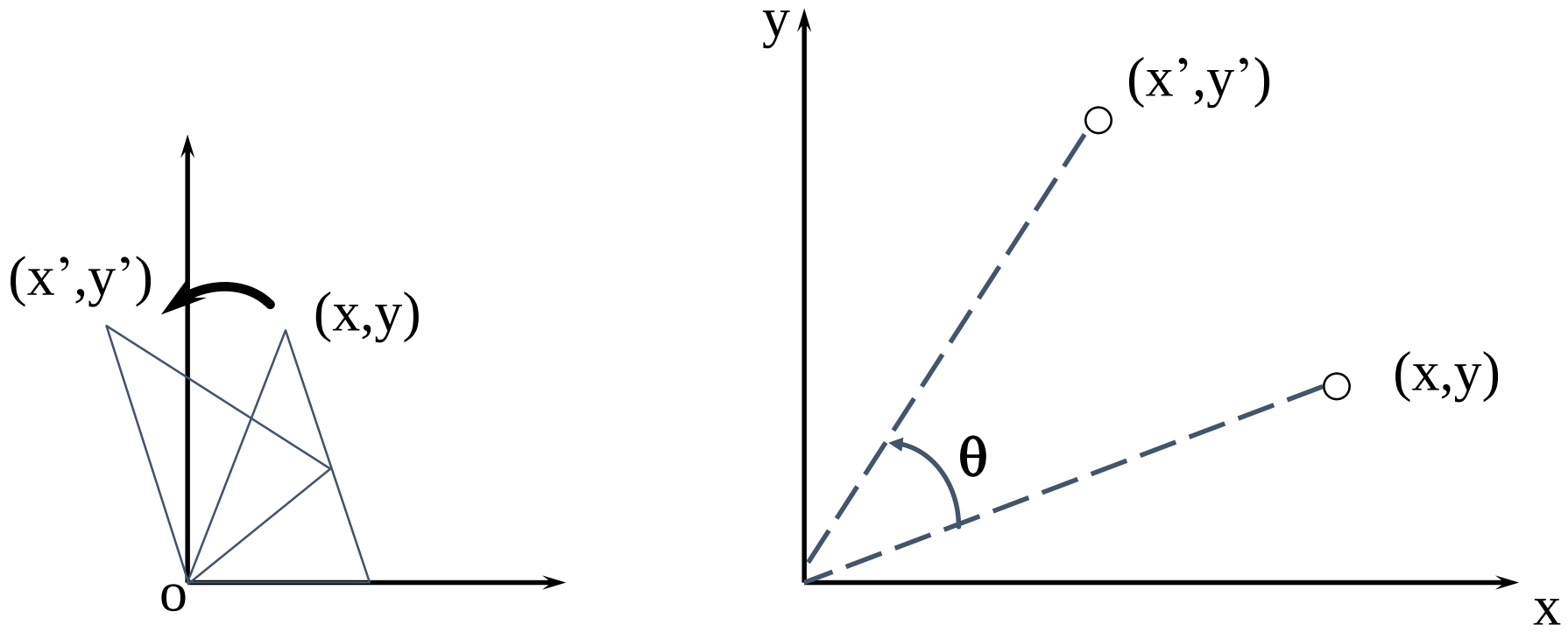
From the double angle formulas:

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

Thus,

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

Rotation About the Origin cont.



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

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

The above 2D rotation is actually a rotation about the z-axis (0,0,1) by an angle θ.

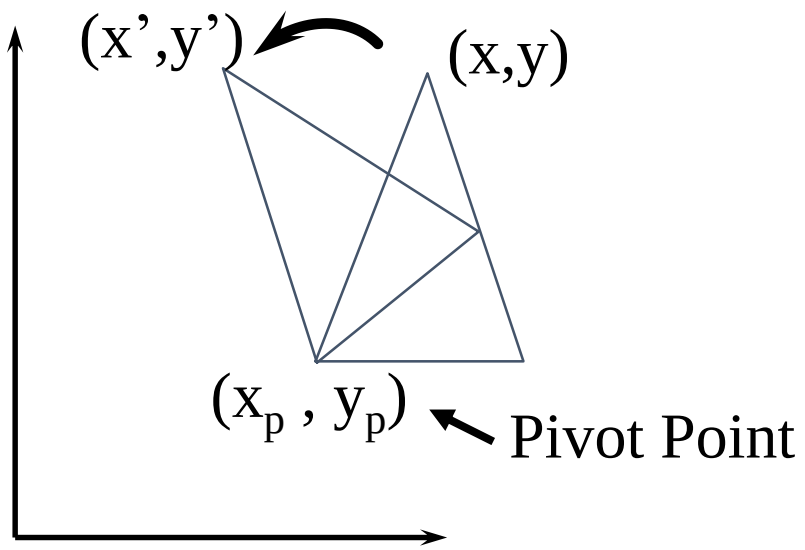
Rotation About the Origin (OpenGL)

Specifying a 2D-Rotation about the origin:

```
glRotatef(theta, 0.0, 0.0, 1.0);
```

`theta`: Angle of rotation in degrees.
The above function defines a rotation about the z-axis (0,0,1).

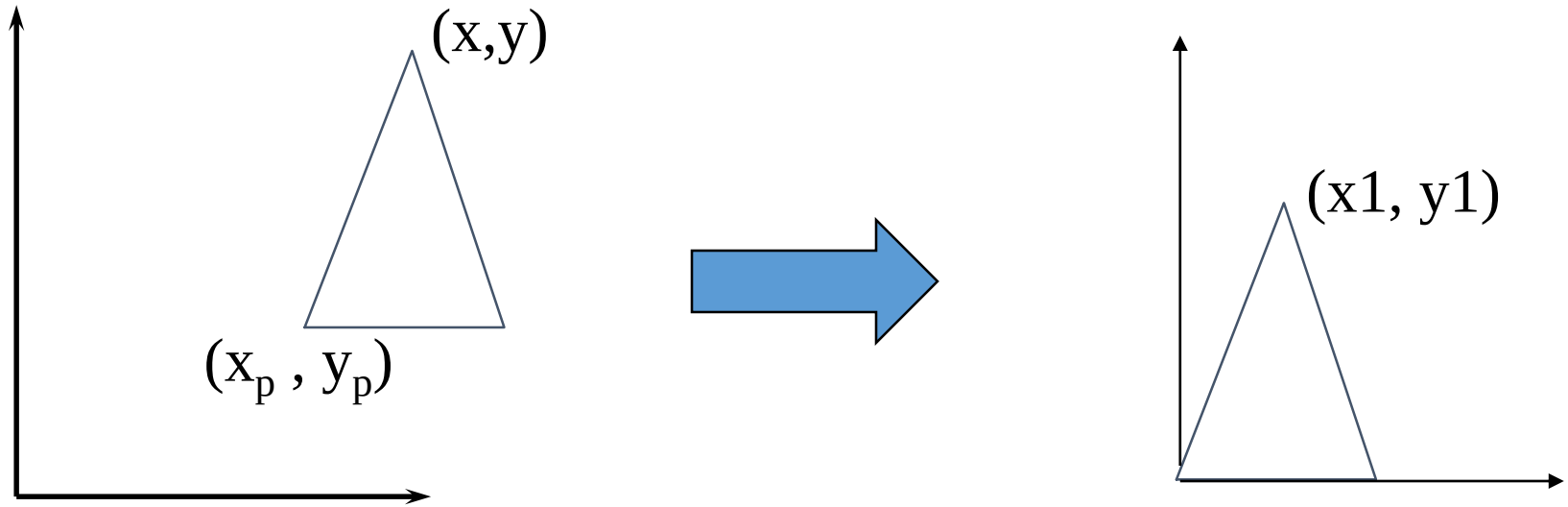
Rotation About a Pivot Point



- Pivot point is the point of rotation
- Pivot point need not necessarily be on the object

Rotation About a Pivot Point

STEP-1: Translate the pivot point to the origin

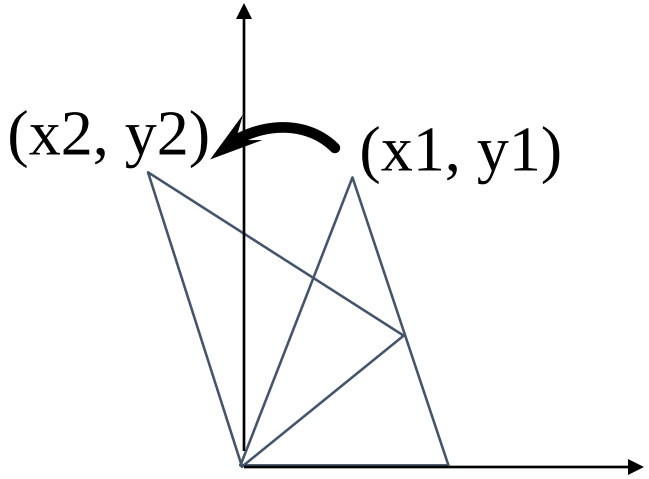


$$x1 = x - x_p$$

$$y1 = y - y_p$$

Rotation About a Pivot Point

STEP-2: **Rotate about the origin**

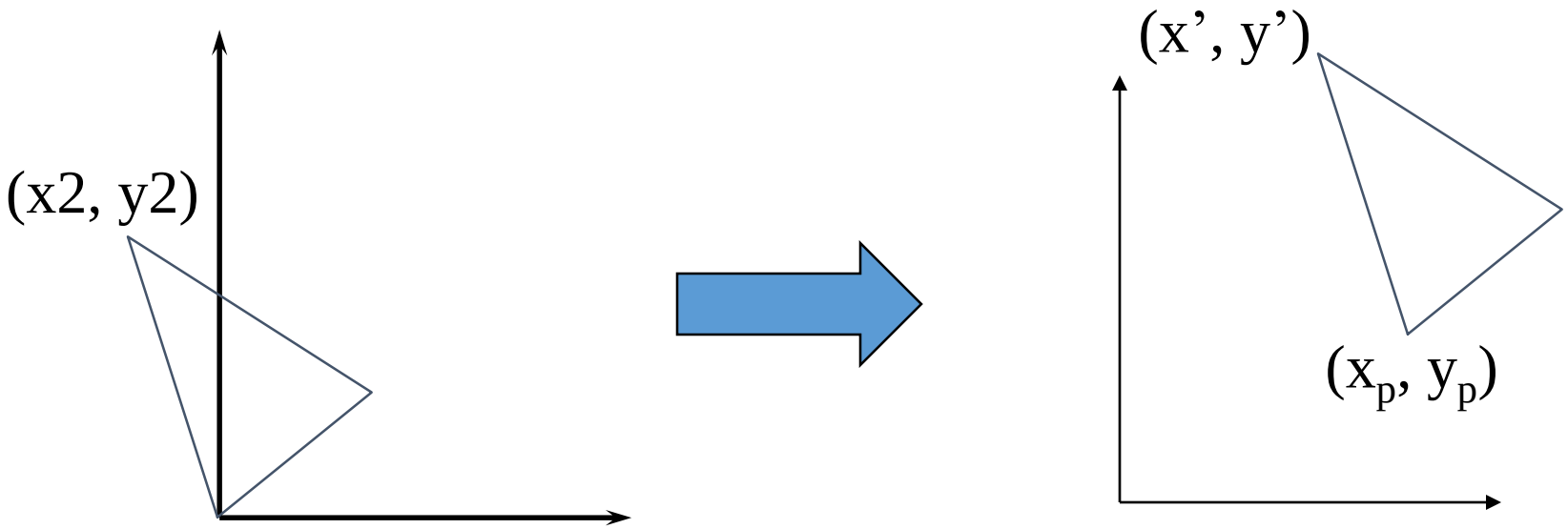


$$x2 = x1\cos\theta - y1\sin\theta$$

$$y2 = x1\sin\theta + y1\cos\theta$$

Rotation About a Pivot Point

STEP-3: **Translate the pivot point to original position**



$$x' = x2 + x_p$$

$$y' = y2 + y_p$$

Rotation About a Pivot Point

$$x' = (x - x_p)\cos\theta - (y - y_p)\sin\theta + x_p$$

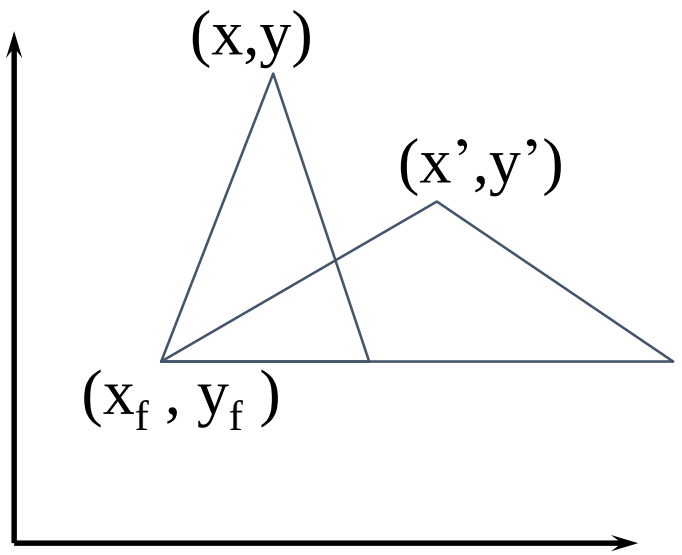
$$y' = (x - x_p)\sin\theta + (y - y_p)\cos\theta + y_p$$

Specifying a 2D-Rotation about a pivot point (xp,yp):

```
glTranslatef(xp, yp, 0);
glRotatef(theta, 0, 0, 1.0);
glTranslatef(-xp, -yp, 0);
```

Note the OpenGL specification of the sequence of transformations in the reverse order !

Scaling About a Fixed Point



- Translate the fixed point to origin
- Scale with respect to the origin
- Translate the fixed point to its original position.

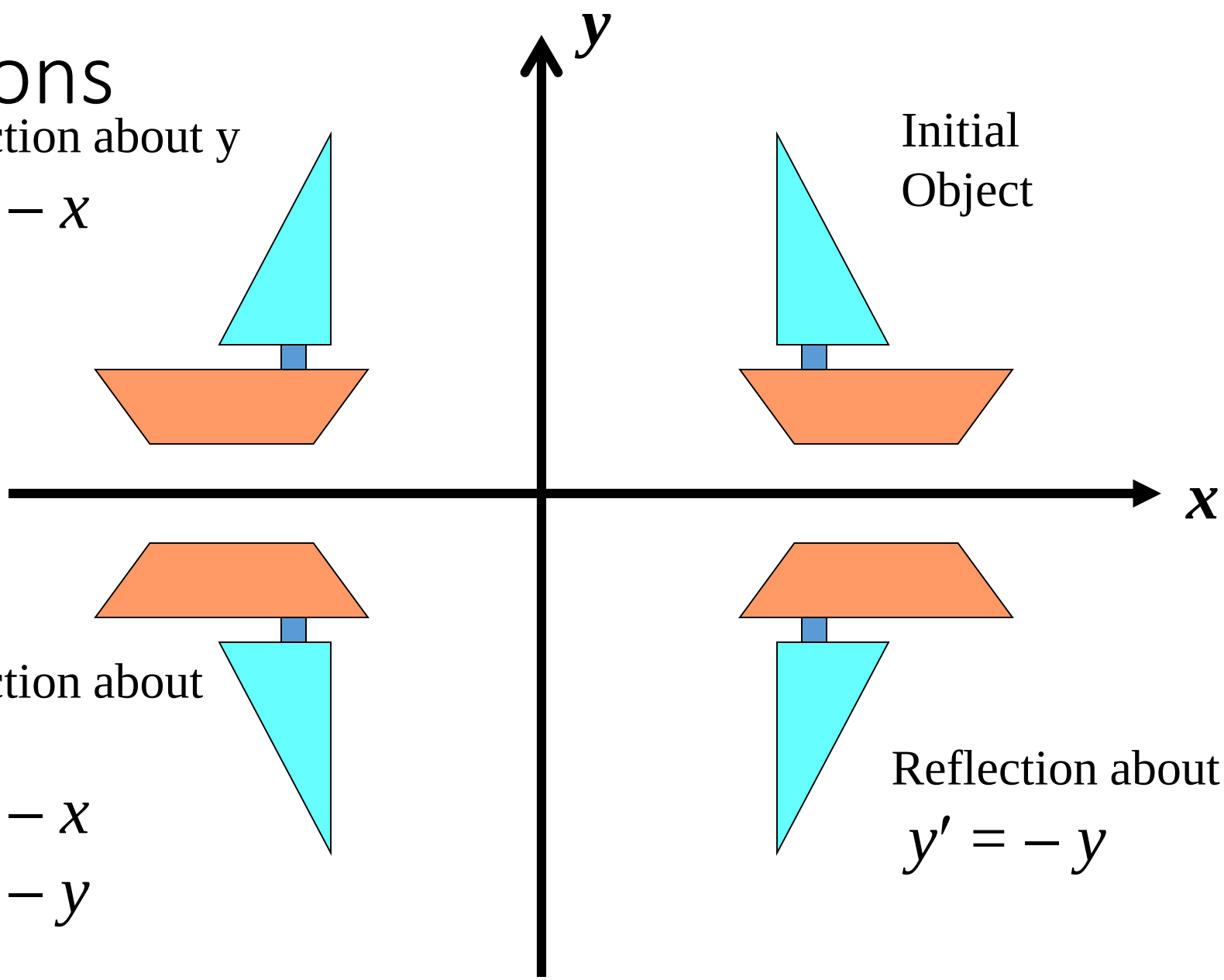
$$x' = (x - x_f).s_x + x_f$$

$$y' = (y - y_f).s_y + y_f$$

Reflections

Reflection about y

$$x' = -x$$



Reflection about origin

$$x' = -x$$

$$y' = -y$$

Reflection about x

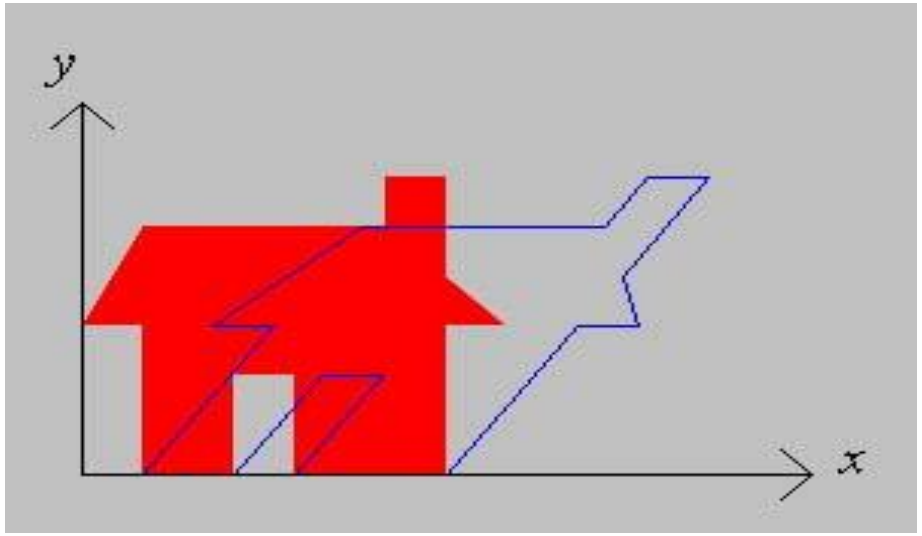
$$y' = -y$$

Reflections(OpenGL)

Reflection about x:	`glScalef(1, -1, 1);`
Reflection about y:	`glScalef(-1, 1, 1);`
Reflection about origin:	`glScalef(-1, -1, 1);`

Shear

Shear in x-direction



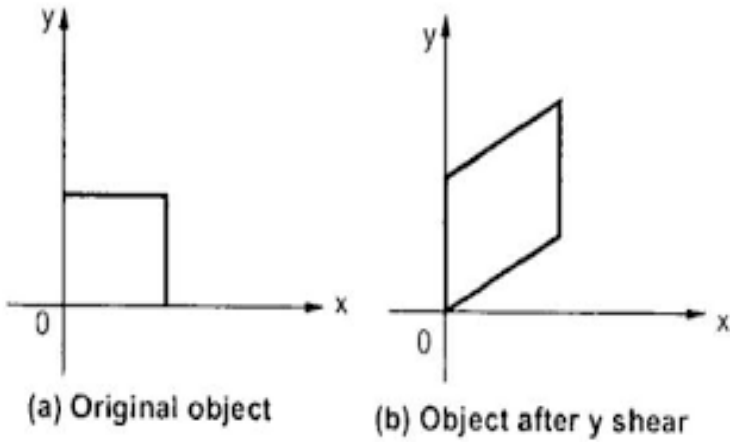
$$x' = x + Sh_x y$$

$$y' = y$$

•A shear transformation in the x-direction (along x) shifts the points in the x-direction proportional to the y-coordinate.
•The y-coordinate of each point is unaffected.

Shear

Shear in y direction



$$x' = x$$

$$y' = y + Sh_y x$$

- A shear transformation in the y-direction (along y) shifts the points in the y-direction proportional to the x-coordinate.
- The x-coordinate of each point is unaffected.

Matrix Representation

$$x' = x + t_x$$

$$y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

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

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

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

$$x' = x.S_x = S_x.x + 0.y$$

$$y' = y.Sy = 0.x + Sy.y$$

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

Matrix Representations

Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Rotation [Origin]

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

Scaling [Origin]

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

Matrix Representations

Reflection about x

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

Reflection about y

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

Reflection about the Origin

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

Matrix Representations

Shear along x

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

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

Shear along y

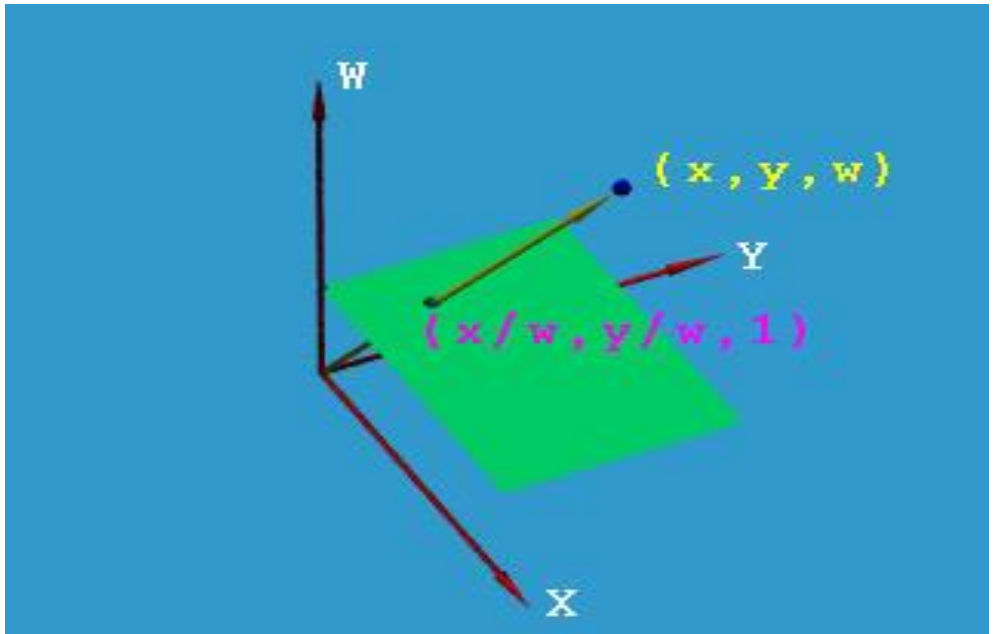
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ h_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

Homogeneous Coordinates

To obtain square matrices an additional row was added to the matrix and an additional coordinate, the *w*-coordinate, was added to the vector for a point. In this way a point in 2D space is expressed in three-dimensional homogeneous coordinates.

This technique of representing a point in a space whose dimension is one greater than that of the point is called homogeneous representation. It provides a consistent, uniform way of handling affine transformations.



Homogeneous Coordinates

•If we use homogeneous coordinates, the geometric transformations given above can be represented using only a matrix pre-multiplication.

• A composite transformation can then be represented by a product of the corresponding matrices.

Cartesian Homogeneous

$$(x, y) \longrightarrow (xh, yh, h), h \neq 0$$

$$\left(\frac{a}{c}, \frac{b}{c}\right) \longleftarrow (a, b, c), c \neq 0$$

Examples: (5, 8) → (15, 24, 3)
(x, y) → (x, y, 1)

Homogeneous Coordinates

Matrix Representation

$$x' = x + t_x = 1.x + 0.y + t_x.1$$
$$y' = y + t_y = 0.x + 1.y + t_y.1$$
$$1 = 0.x + 0.y + 1.1$$
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

Basic Transformations

Translation
P'=TP

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

Rotation [O]
P'=RP

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

Scaling [O]
P'=SP

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

Inverse of Transformations

If, $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [T] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ then, $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = [T]^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

Examples: $T^{-1}(t_x, t_y) = T(-t_x, -t_y)$

$$R^{-1}(\theta) = R(-\theta)$$

$$S^{-1}(s_x, s_y) = S\left(\frac{1}{s_x}, \frac{1}{s_y}\right)$$

$$M_x^{-1} = M_x \, and \, M_y^{-1} = M_y$$

Transformation Matrices

Additional Properties:

$$T(t_x, t_y)T(u_x, u_y) = T(t_x + u_x,\ t_y + u_y)$$

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

$$\left|R(\theta)\right| = 1$$

$$S(s_x, s_y)S(w_x, w_y) = S(s_x w_x,\ s_y w_y)$$

Composite Transformations

Transformation T followed by
Transformation Q followed by
Transformation R:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [R][Q][T] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: (Scaling with respect to a fixed point)

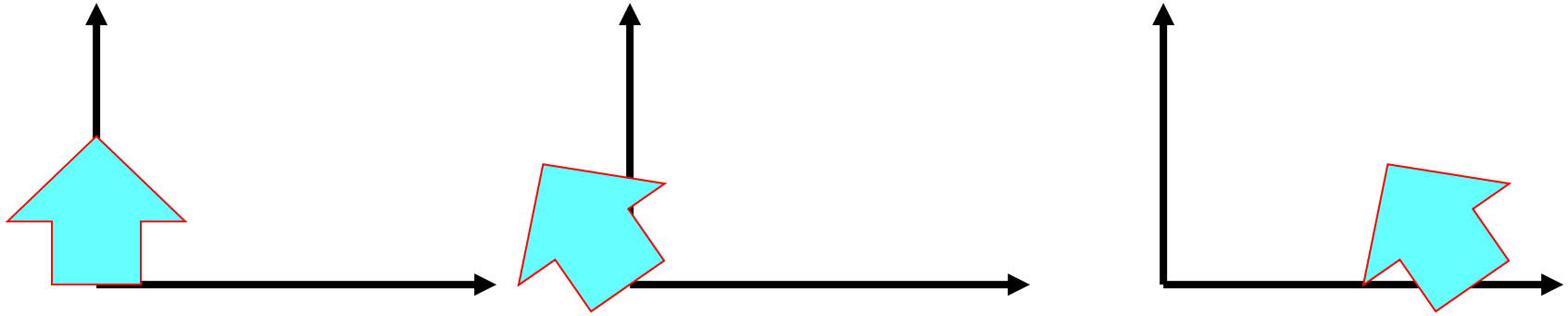
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Order of Transformations

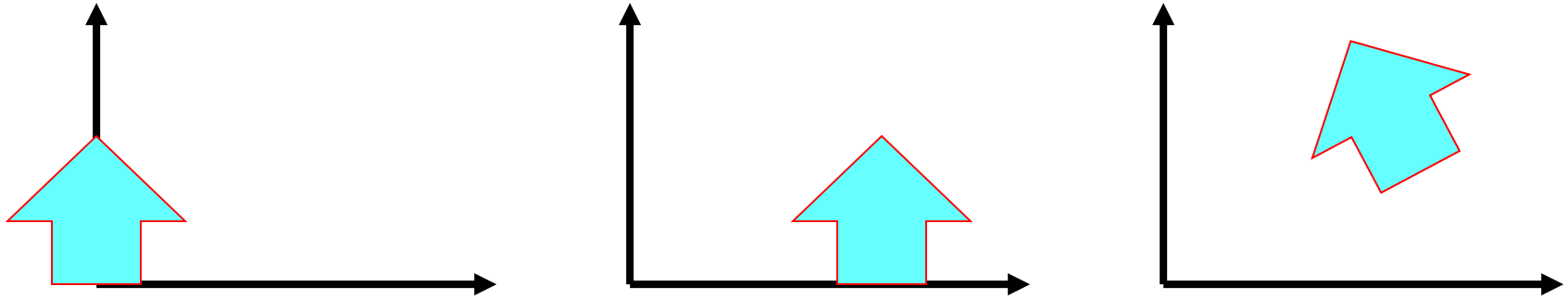
Order of Transformations

In composite transformations, the order of transformations is very important.

Rotation followed by Translation:



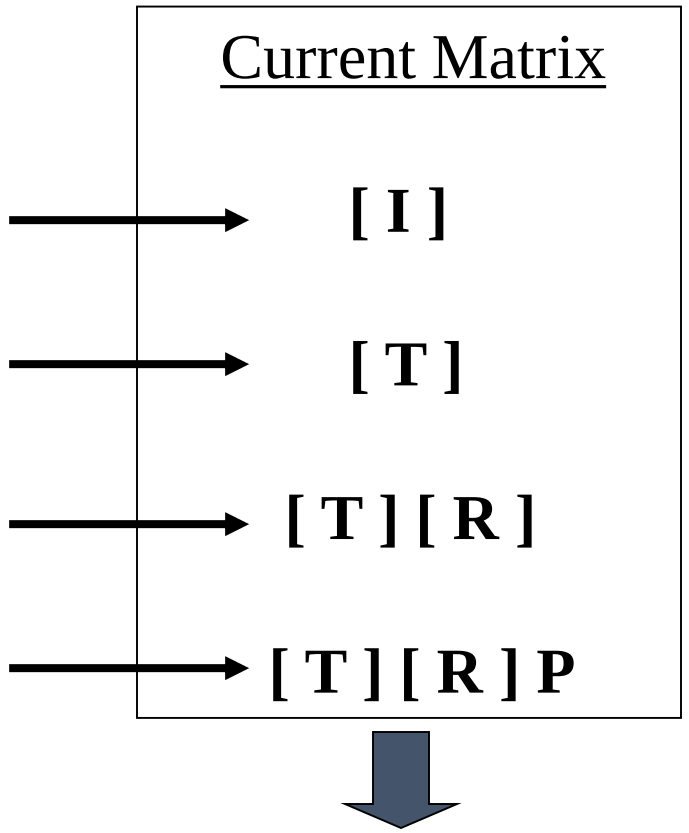
Translation followed by Rotation:



Order of Transformations (OpenGL)

OpenGL ***postmultiplies*** the current matrix with the new transformation matrix

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(tx, ty, 0);
glRotatef(theta, 0, 0, 1.0);
glVertex2f(x,y);
```



Rotation followed by Translation !!

Affine Transformation

A general invertible, linear, transformation.

$$x' = a_1 x + b_1 y + c_1$$
$$y' = a_2 x + b_2 y + c_2$$
$$(a_1 b_2 - a_2 b_1 \neq 0)$$

Transformation Matrix:

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformation: Properties

- Product of affine transformations is affine.
- Affine transformations preserve linearity of segments.
- Affine transformations preserve parallelism between lines.
- Affine transformations are invertible.

Thank You