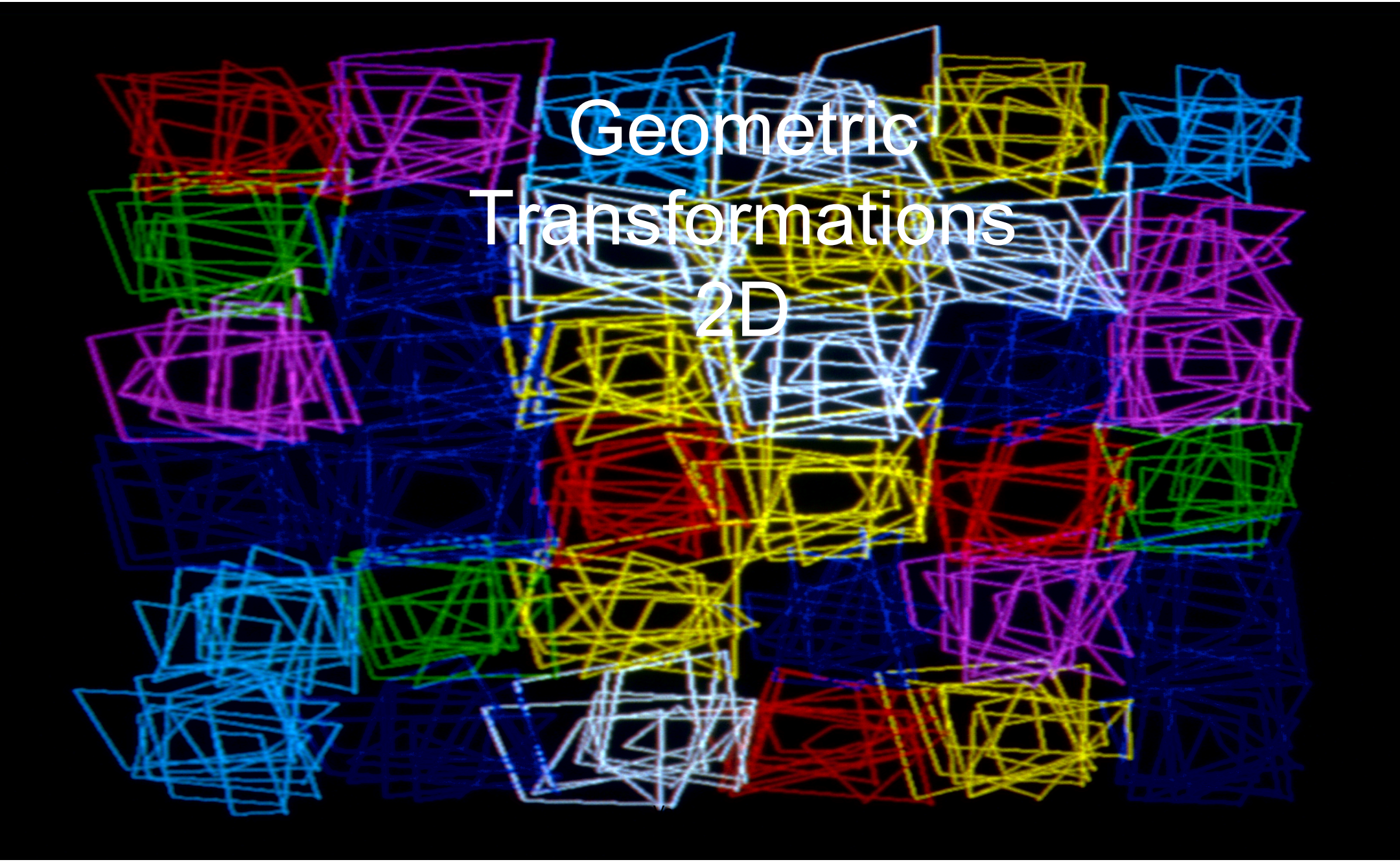
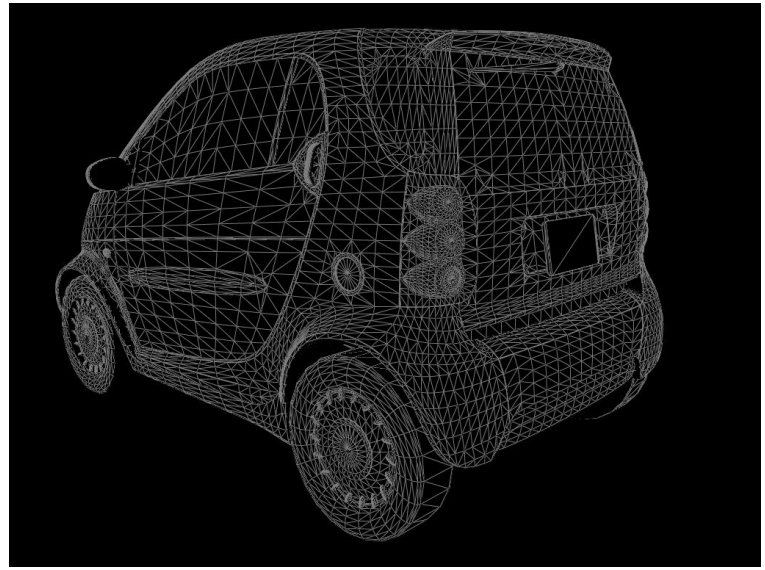

Geometric
Transformations
2D

# Geometric Transformations

- Operations performed on geometric objects to reposition, resize, etc.
- Modeling transformations used to “model” or assemble parts to make up a geometric object

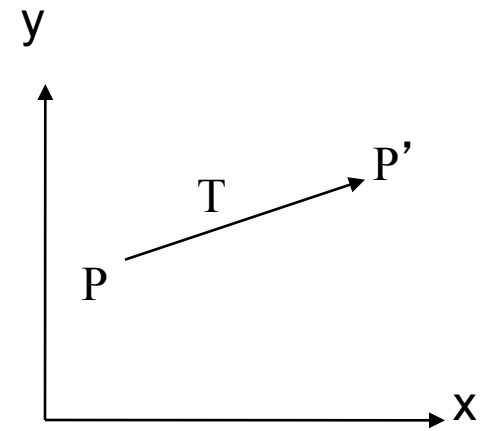


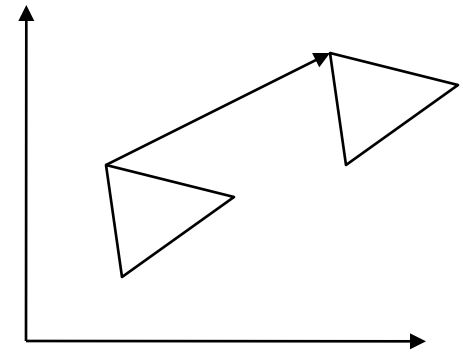
# 2D Translation

- Move a point (or set of points) from P to P'
  - $x' = x + t_x$
  - $y' = y + t_y$
- In vector form
  - $P' = P + T$

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

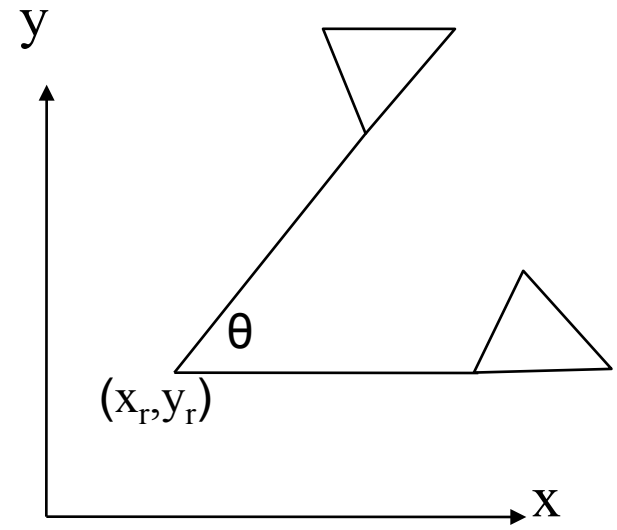
- No deformation in shape: example of Rigid body transformation



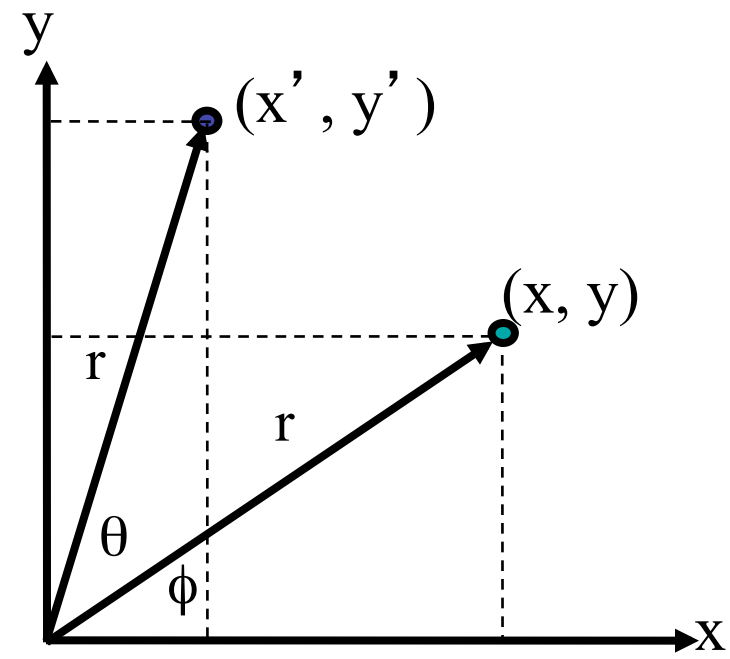
# 2D Rotation

- Rotation by an angle about an axis at a rotation point (pivot point)
- In 2D, rotation axis is the z-axis and positive rotation is counterclockwise
- We will first discuss rotation about the origin



# Rotate (x,y) by the angle θ

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




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

– Substitute x and y from first 2 equations into above 2 equations:
$x' = x \cos(\theta) - y \sin(\theta)$
$y' = x \sin(\theta) + y \cos(\theta)$

x’ = x cos(θ) - y sin(θ)
y’ = x sin(θ) + y cos(θ)

– In matrix form:

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

– Rotation is also a rigid body transformations
– Rotation matrix can be extended for rotations about an arbitrary point (x, y)

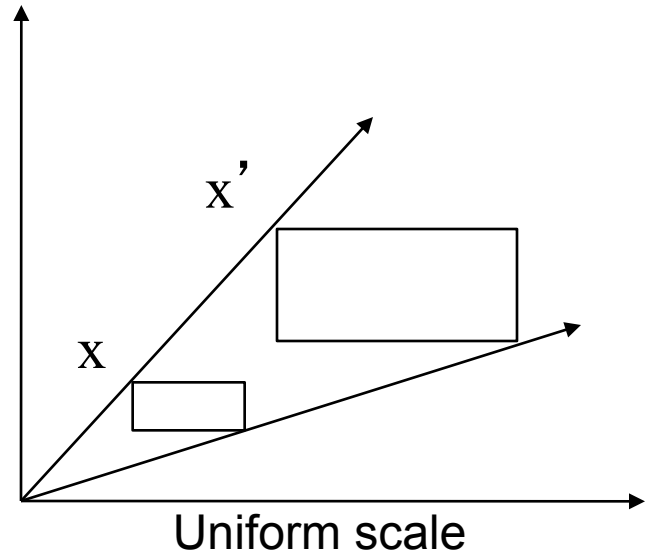
# 2D Scaling

– $x' = x * s_x$

– $y' = y * s_y$

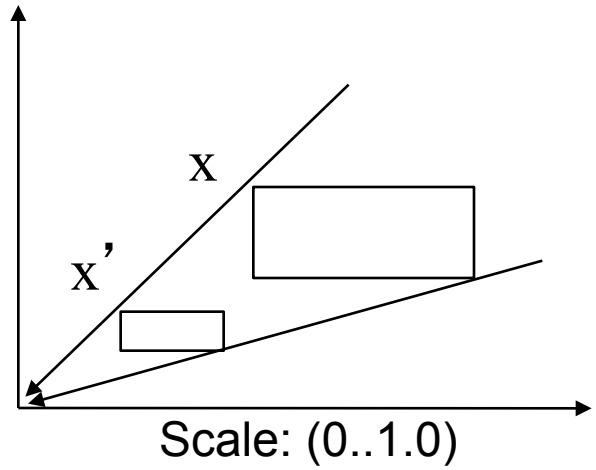
– In matrix form:

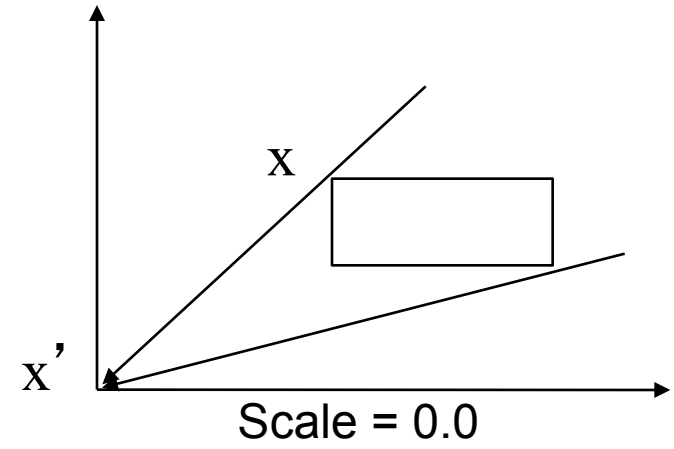
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

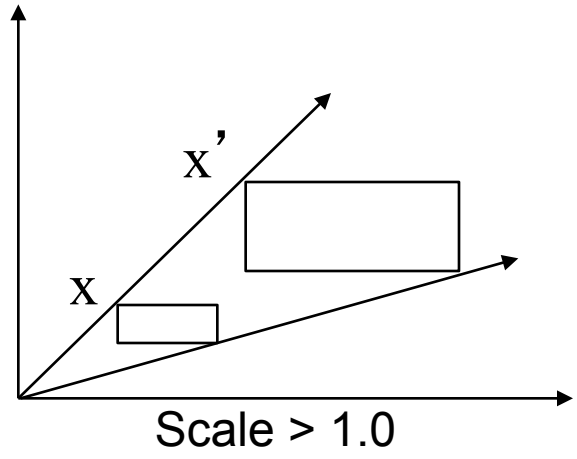


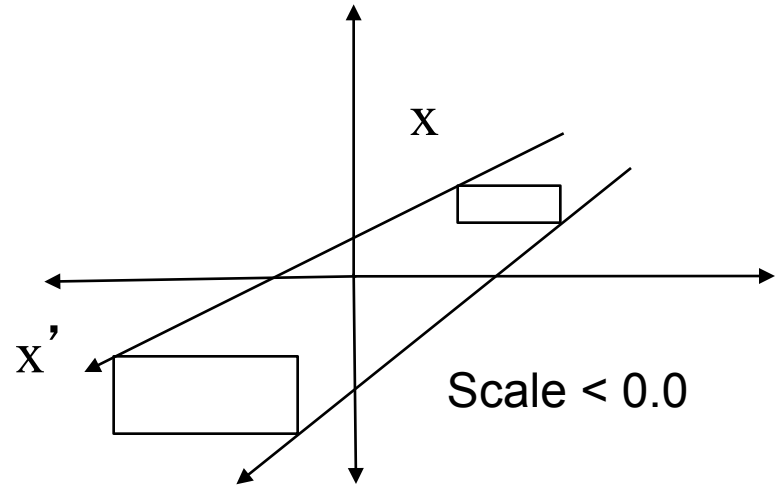

– If $s_x = s_y$ then uniform scale

# Uniform Scaling Factors



# Concatenating transformation

- A series of transformations can be performed on a vector
  - E.g. $x' = S_1 x, \quad x'' = R x', \quad x''' = S_2 x''$
- Alternatively, the transformations can all be concatenated and then applied to the vector
  - This is possible because rotations and scale are represented as matrices
  - E.g. $x''' = (S_2 R S_1) x$
  - Note the order in which the transformations are concatenated

– Rotation:

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

– Scaling:

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

– Translation (oops)

P’ = P + T

– Prevents us from concatenating an arbitrary series of transformations into a single matrix.

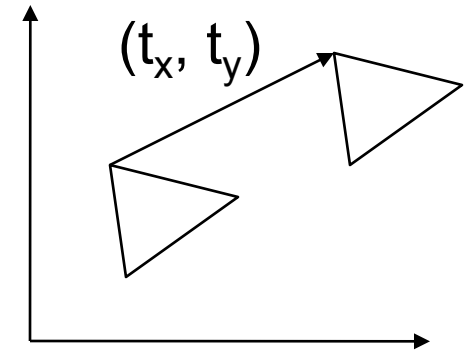
# Homogeneous coordinate systems

- In 3D, we can express 2D translations in matrix form
  - Homogeneous coordinate: $(hx, hy, h)$
  - $(x, y) \Rightarrow (hx, hy, h)$
  - $(x_h, y_h, h) \Rightarrow (x_h/h, y_h/h)$
  - For our purposes, we can set $h = 1$
- In 2D, use 3D homogeneous coordinate to do transformations, then divide by $h$ to get back to 2D
- In 3D, use 4D homogeneous coordinate to do transformations then divide by $h$ to get back go 3D

# 2D Translation using homogeneous coordinate

- The last column is used to represent the translation
- h = 1

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



# 2D Rotation using homogeneous coordinate

– Upper left 2 x 2 sub-matrix gives rotation

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

# 2D Scale about the origin

– A lower right component of $s$ can be used to specify a uniform scale by $1/s$

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

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

(x’ , y’ , h) = (x, y, s)
(x’ , y’ ) = (x/s,y/s)

# Inverse transformations

– Translation: negate the translation

$$T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

– Rotation: negate the θ

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

– Only the sine function is affected by the change:

$$R^{-1} = R^T$$

– Scale Inverse

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

– A transformation multiplied by its inverse gives the identity

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

# Composite transformations

- Transformations can be multiplied to generate a composite (concatenation) transformation matrix

- $P' = (T_n (T_{n-1} \ldots (T_2 (T_1 P)))) = (T_n T_{n-1} \ldots T_2 T_1) P = T P$

- More efficient since the same series of transformations are usually applied to many different points (e.g. to transform each vertex of a polygonal object)

# Matrix concatenation properties

- Matrices are associative
  - $M_3\ M_2\ M_1 = (M_3\ M_2)\ M_1 = M_3\ (M_2\ M_1)$
- Matrices are not commutative
  - The order in which the matrix operations are performed for a vector is important
  - $M_2\ M_1 \neq M_1\ M_2$
  - In general: $M_2\ M_1\ V \neq M_1\ M_2\ V$

# Composite translations

- Multiply matrices to get composite transformations
- Two consecutive translations:

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

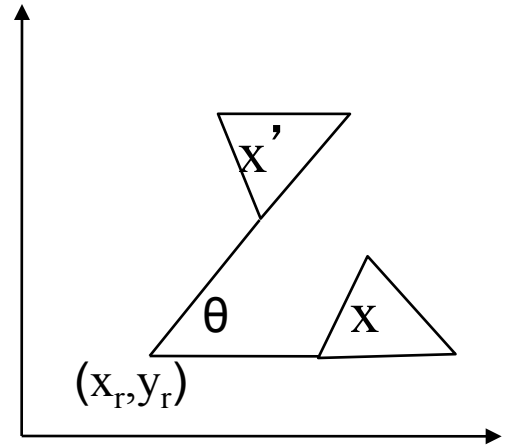
- Two consecutive rotations (you can verify):

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

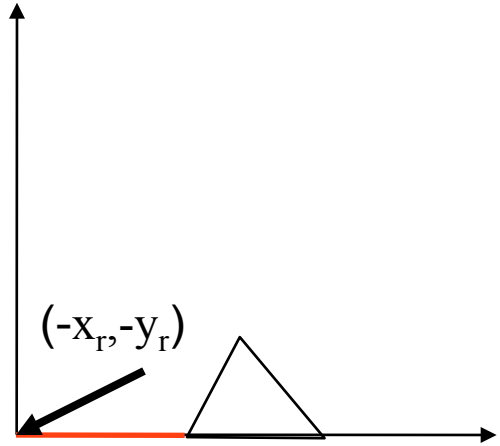
– Two consecutive scales multiplies the scale factors:

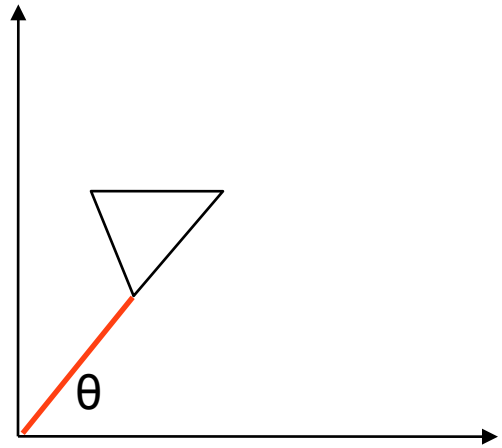
$$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1x}s_{2x} & 0 & 0 \\ 0 & s_{1y}s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

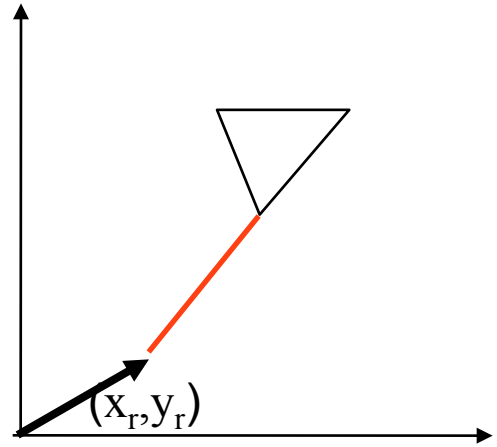
# Rotation about an arbitrary point



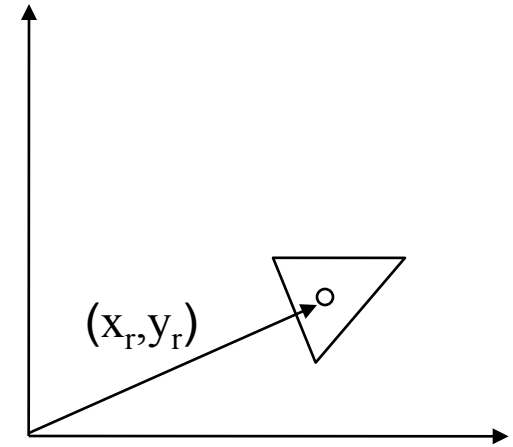

- Translate $(-x_r, -y_r)$: $T(-x_r, -y_r)$
- Rotate θ about the origin: $R(\theta)$
- Translate $(x_r, y_r)$ back: $T(x_r, y_r)$
- Concatenated transformation: $T(x_r, y_r)\ T(\theta)\ T(-x_r, -y_r)$



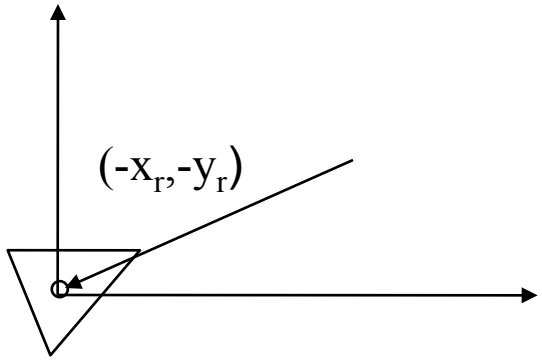



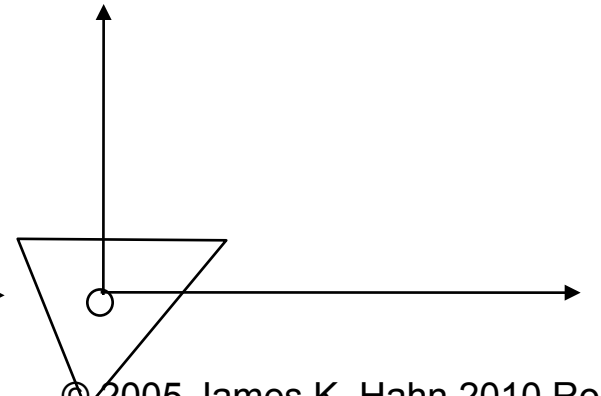



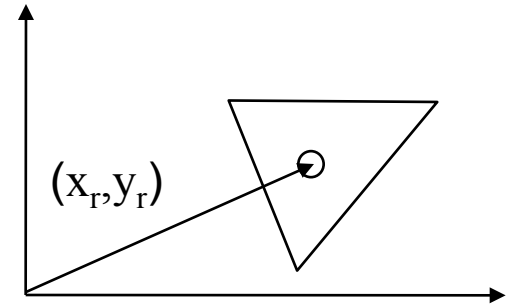
# Scale about an arbitrary point




- Translate $(-x_r, -y_r)$ to the origin: $T(-x_r, -y_r)$
- Scale about the origin: $S(s_x, s_y)$
- Translate $(x_r, y_r)$ back: $T(x_r, y_r)$
- Concatenated transformation: $T(x_r, y_r)\ S(s_x, s_y)\ T(-x_r, -y_r)$

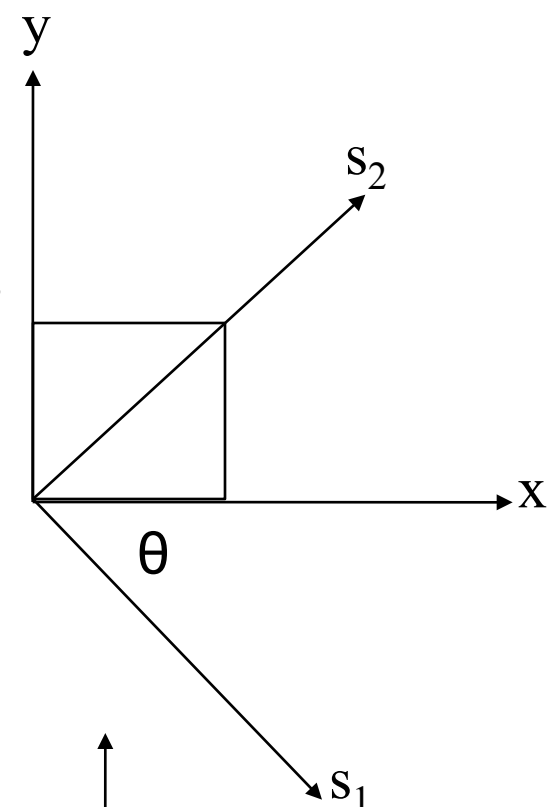


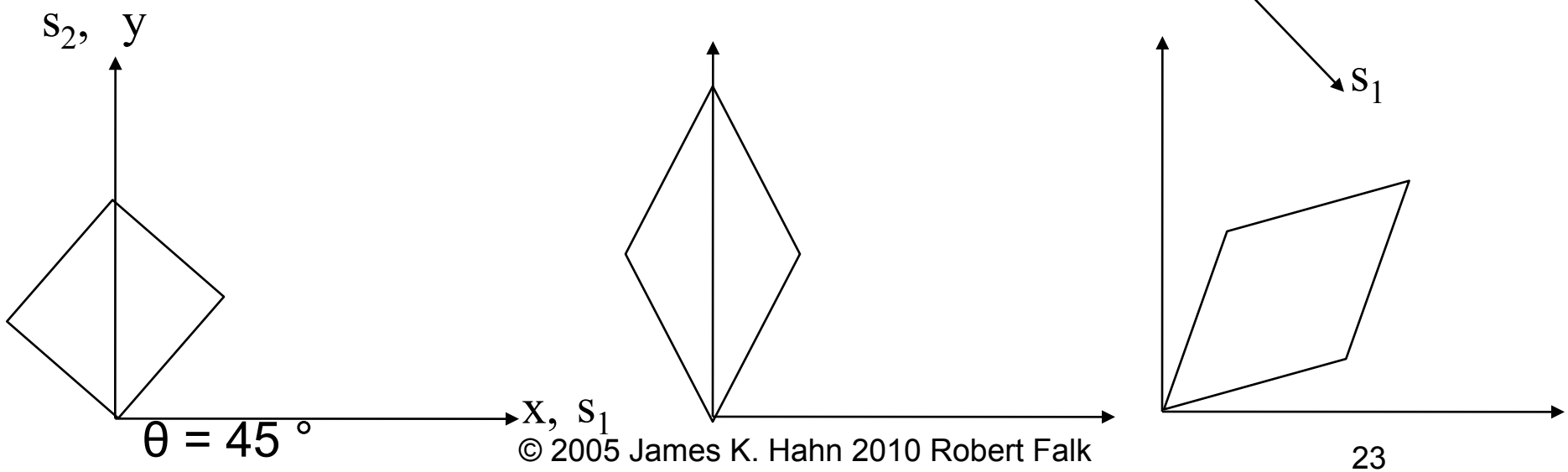





# Scale about Arbitrary direction

- Rotate so that the scaling direction aligns with the x and y axes
- Scale
- Rotate back



# 2D Rigid Body Transformation

- Composed of a rotation and translation
- The shape is preserved
  - All distance and angles between points are preserved

$$\begin{bmatrix} r_{xx} & r_{xy} & t_x \\ r_{yx} & r_{yy} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Upper 2x2 matrix (the rotational part) is an orthogonal matrix
  - Each row is orthogonal to other rows, and each column is orthogonal to other columns (dot products == 0)
  - Rows/ columns are of unit length (orthonormal)

$$r_{xx}r_{yx} + r_{xy}r_{yy} = 0$$

$$r_{xx}^2 + r_{xy}^2 = r_{yx}^2 + r_{yy}^2 = 1$$

- The product of the rows and columns of the rotation matrix are both 0
- The length of the rows and columns are both 1

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

$$r_{xx}r_{yx} + r_{xy}r_{yy} = \cos(\theta)\sin(\theta) - \sin(\theta)\cos(\theta) = 0$$

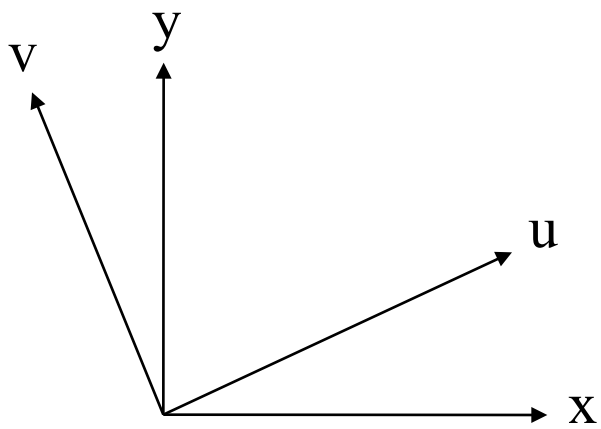
$$r_{xx}r_{xy} + r_{yx}r_{yy} = -\cos(\theta)\sin(\theta) + \sin(\theta)\cos(\theta) = 0$$

$$\cos^2(\theta) + (-\sin^2(\theta) = 1$$

$$\sin^2(\theta) + \cos^2(\theta) = 1$$

# Constructing rotation matrices

– If we want a rotation matrix that will take the x axis into a vector u and y axis into an orthogonal vector v



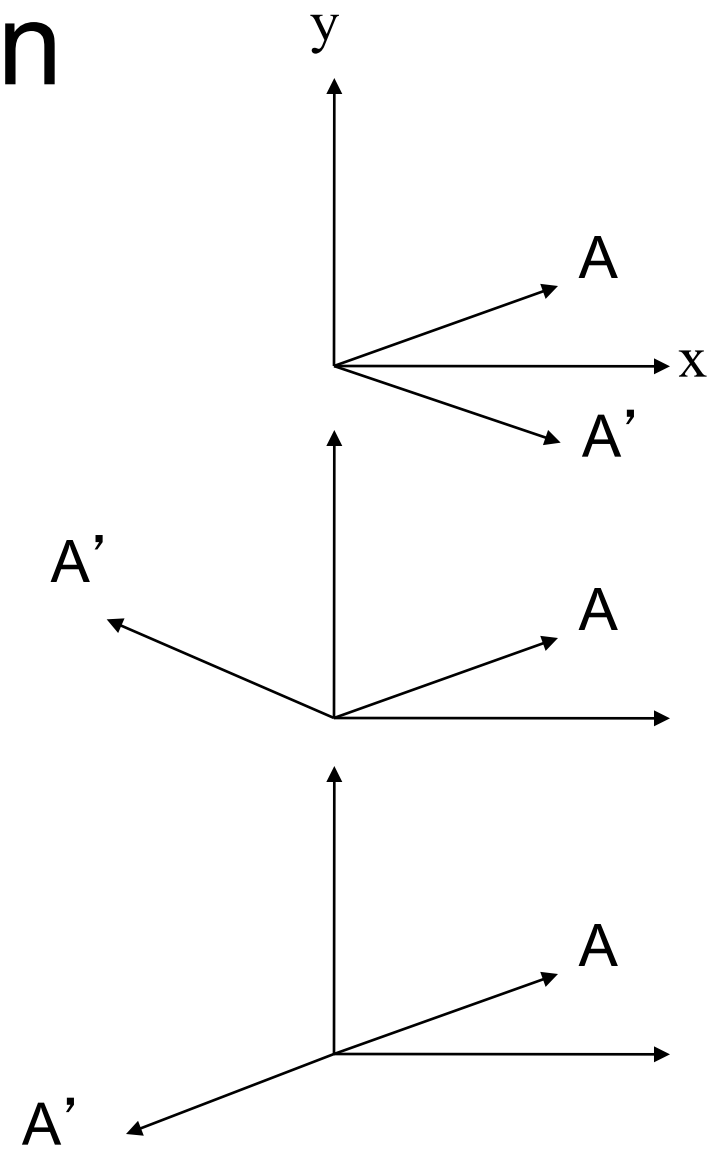

$$\begin{bmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix}$$

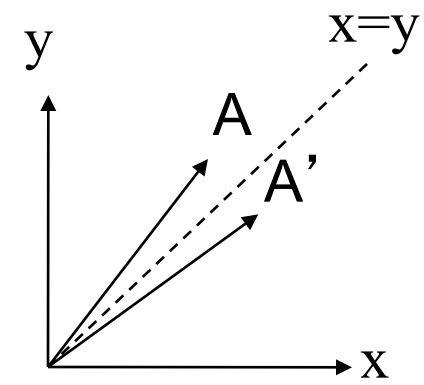
– Useful for determining rotation matrix to get objects into certain final configuration given as directions of x, y, (and z) axes

# Reflection

- About the x axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- About the y axis $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- About the origin $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



- About the x=y axis
  - First rotate +/-45° so that x=y is on the x (or y) axis
  - Reflect about the x (or y) axis
  - Rotate back
- Alternatively
  - First reflect about x-axis
  - Rotate counterclockwise 90°

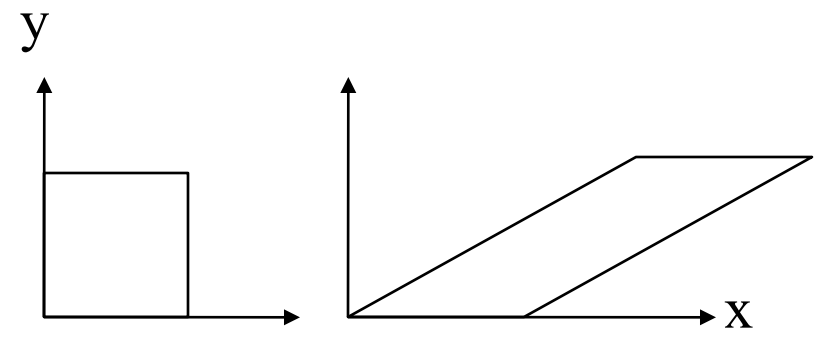


# Shear

– x-direction shear

$x' = x + sh\ y,\ \ y' = y$

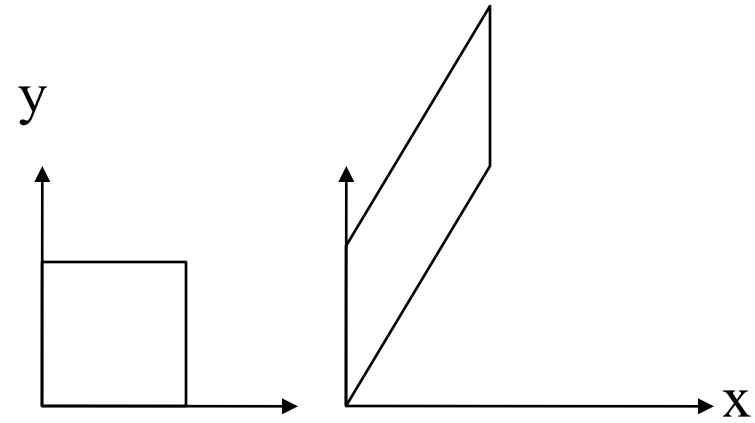
$$\begin{bmatrix} 1 & sh & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



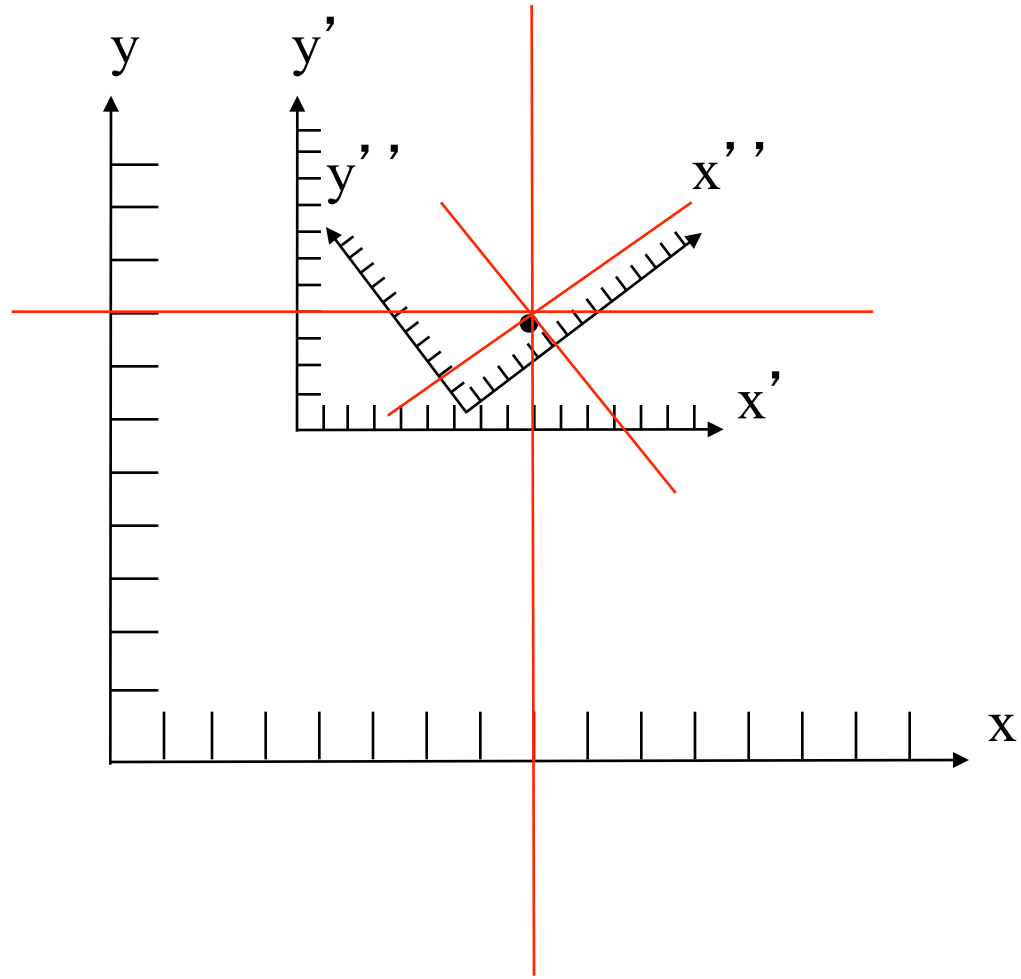

– y-direction shear

$x' = x,\ \ y' = y + sh\ x$

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



# Change of Coordinate Systems



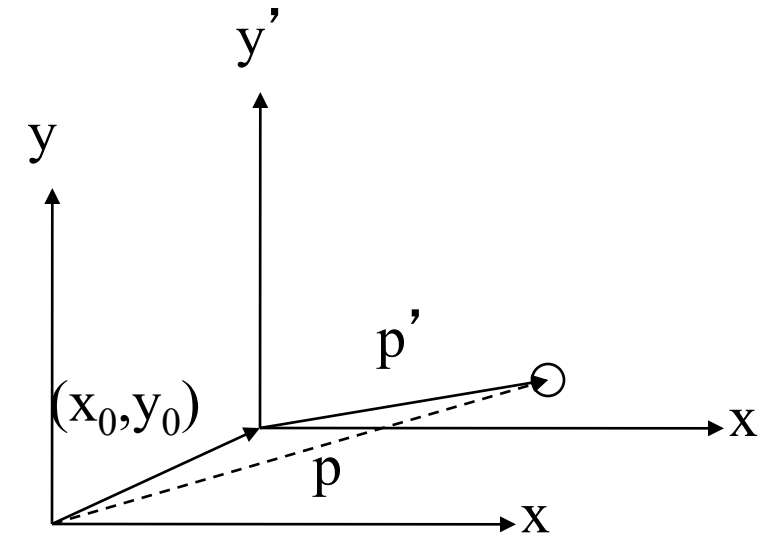

p=(8,8)

p'=(9,4)

p''=(6,2)

# Transformations between 2D coordinate systems

- A point can be described both in the unprimed coordinate system (p) and the primed coordinate system (p')
- $p=(x_0, y_0) + p'$



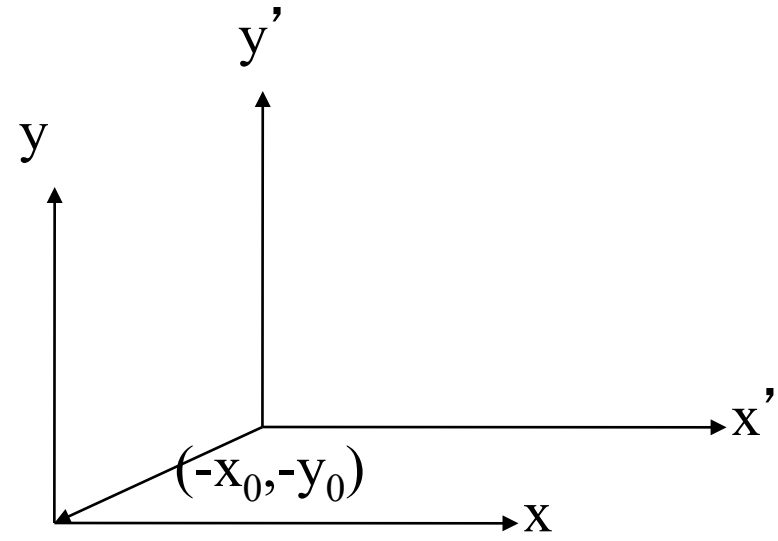

- The transformation to take a point from the unprimed to the primed coordinate system is the transformation needed to move/rotate the primed coordinate system so that it is coincident with the unprimed coordinate system

– A translation by $(-x_0,-y_0)$ will make the two coordinate systems coincident

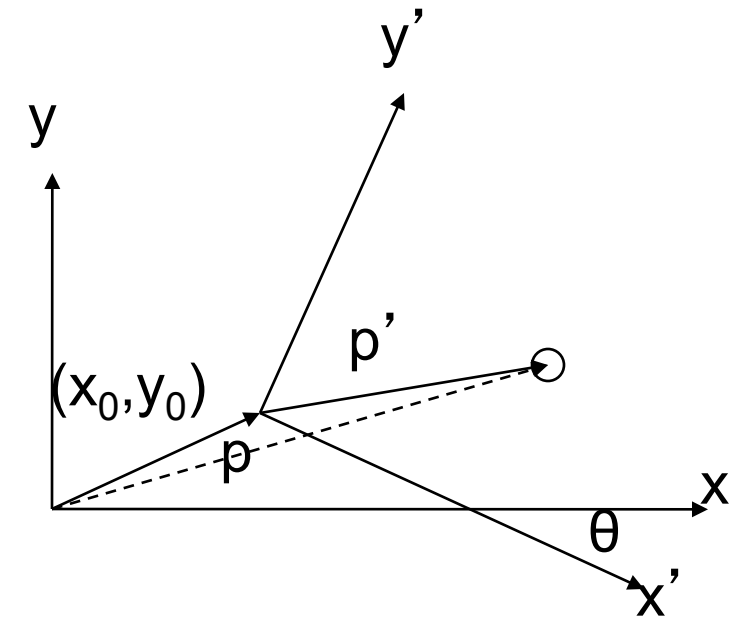
$$\begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

– The transformation from the unprimed to the primed coordinate system is

$$p' = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} p$$



– A translation followed by a rotation creates a general rigid body transformation




- Origin Translation:

$$T = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Axes Rotation:

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

– Concatenate the two transformations to go from unprimed to the primed coordinate system

  $M = R\,T$

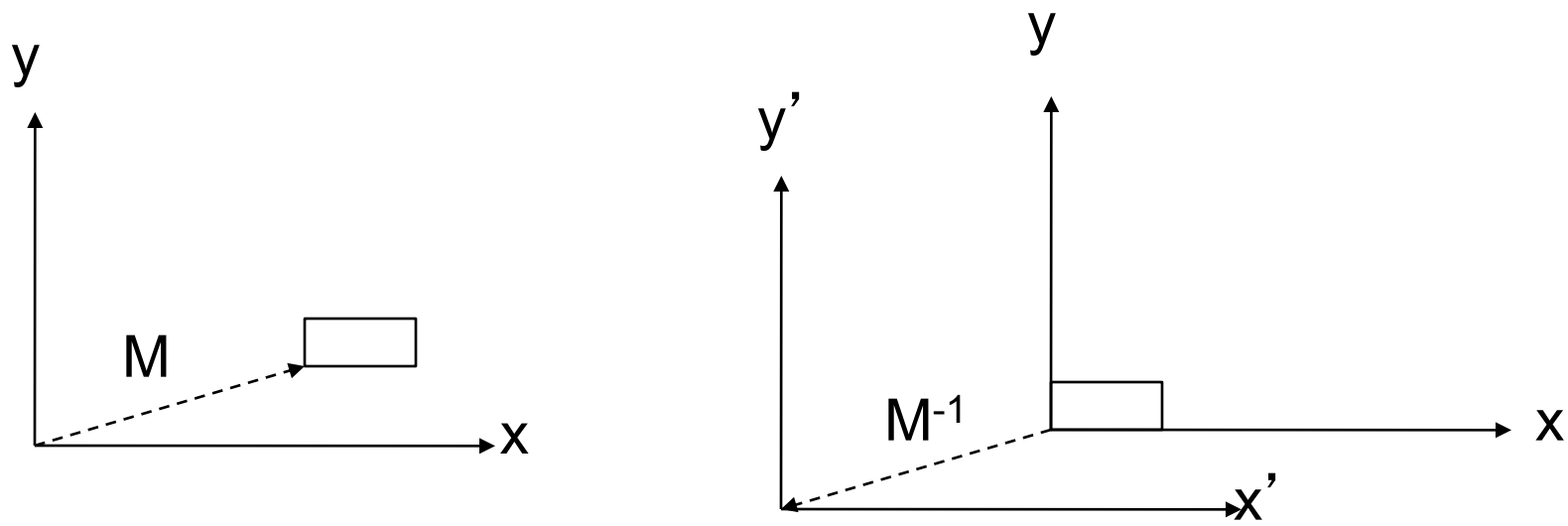
  $p' = M\,p$

– R could have been derived alternatively if we know the unprimed coordinates of the axes of the primed coordinate system $(u_x, u_y)$ and $(v_x, v_y)$

$$R = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Transformation of object and transformation of coordinate systems

– Transforming an object (vector) by a matrix M is equivalent to transforming the coordinate system by $M^{-1}$ and leaving the object unmoved



– To set geometric transformation mode

```
glMatrixMode( GL_MODELVIEW );
```

– This sets a 4x4 modelview matrix as the current matrix
– Any subsequent transformations are multiplied by the current matrix
– To assign an identity matrix as the current matrix

```
glLoadIdentity( );
```

# Constructing transformation matrix in OpenGL

– translation matrix (4x4)

```
glTranslate*( tx, ty, tz );
```

- *: float or double

– rotation matrix

```
glRotate*( theta_deg, vx, vy, vz );
```

– scaling matrix

```
glScale*( sx, sy, sz );
```

# Next: 2D Viewing