Computer Graphics (CS 4731)
Lecture 5 (Part 1)
Introduction to Transformations

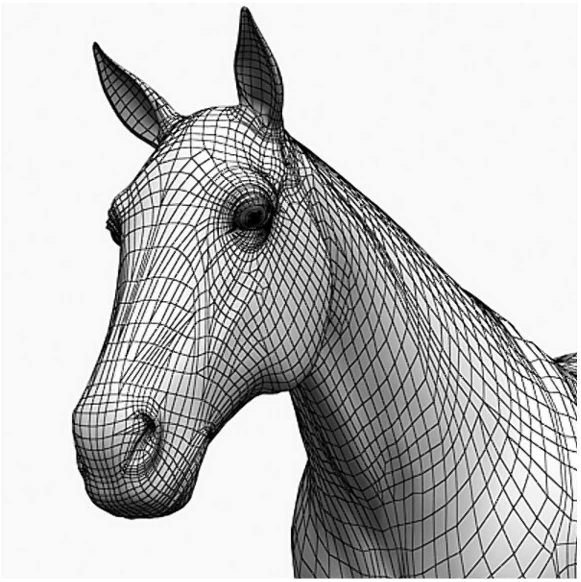
Prof Emmanuel Agu

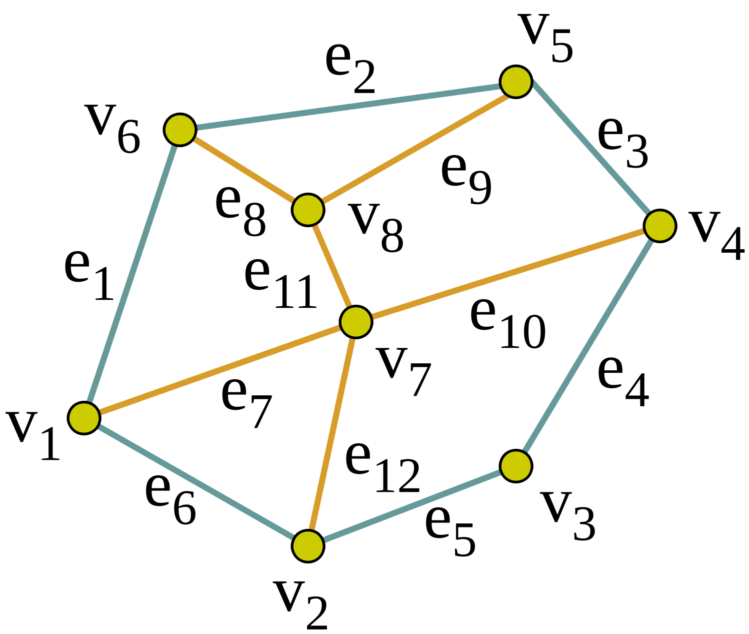
Computer Science Dept.

Worcester Polytechnic Institute (WPI)

So Far: Representing a Mesh

- Learned how to read in and store graphics objects/meshes



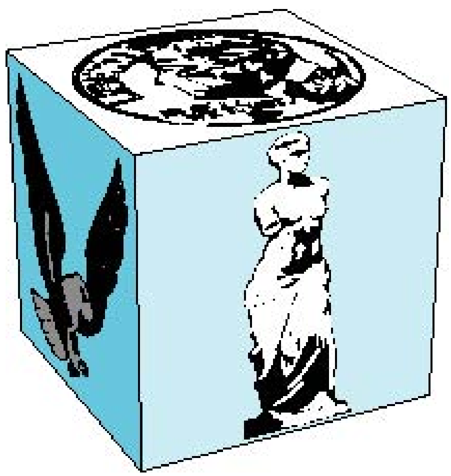


Introduction to Transformations

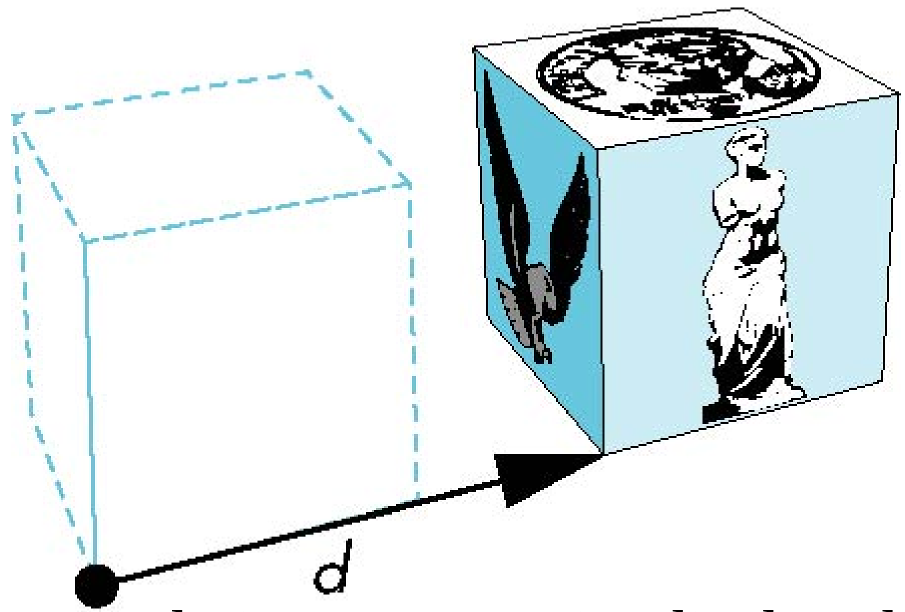
- May also want to transformation objects by changing its:
  - Position (translation)
  - Size (scaling)
  - Orientation (rotation)
  - Shapes (shear)

Translation

- Move each vertex by **same** distance $\mathbf{d} = (\mathbf{d_x}, \mathbf{d_y}, \mathbf{d_z})$



object



translation: every point displaced by same vector

Scaling

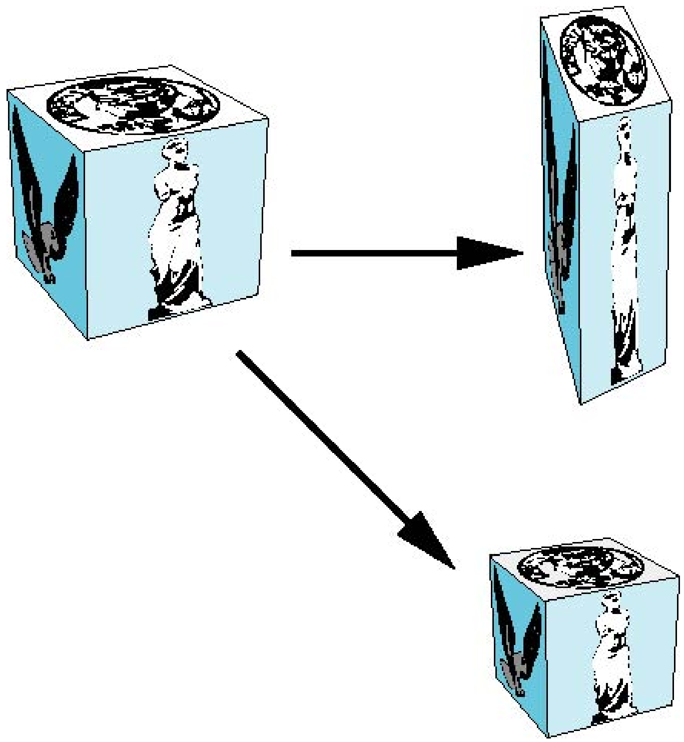
Expand or contract along each axis (fixed point of origin)

$x' = s_x x$
$y' = s_y y$
$z' = s_z z$

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

where

$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z)$$



Introduction to Transformations

- We can transform (translation, scaling, rotation, shearing, etc) object by applying matrix multiplications to object vertices

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

Transformed Vertex

Transform Matrix

Original Vertex

- Note: point (x,y,z) needs to be represented as (x,y,z,1), also called **Homogeneous coordinates**

Why Matrices?

- Multiple transform matrices can be pre-multiplied
- For example:

  transform 1

  transform 2 ….

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

Transformed Point

Transform Matrices can
Be pre-multiplied

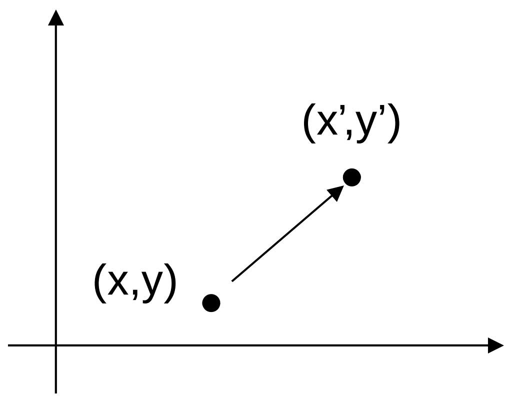
Original Point

Translation

- To reposition a point along a straight line
- Given point *(x,y)* and translation distance (t_x, t_y)
- The new point: *(x',y')*

$x' = x + t_x$

$y' = y + t_y$



or

$P' = P + T$ where $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$ $P = \begin{pmatrix} x \\ y \end{pmatrix}$ $T = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$

2D Translation Matrix => 3x3 Matrix

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

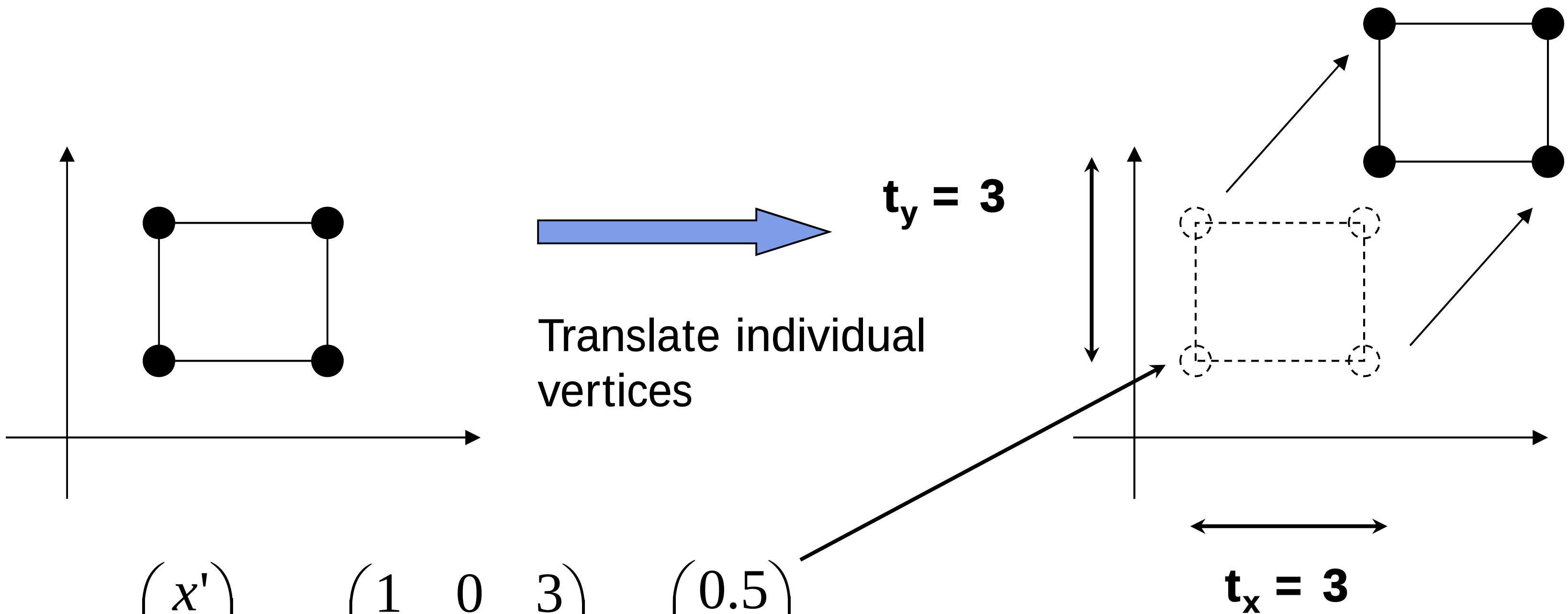
use 3x1 vector

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

- Note: it becomes a matrix-vector multiplication

Translation of Objects

- How to translate an object with multiple vertices?



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 0.5 \\ 0.5 \\ 1 \end{pmatrix}$$

Repeat multiplication
for all four vertices

3D Translation Matrix

- Now, 3D :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Translate(tx,ty,tz)

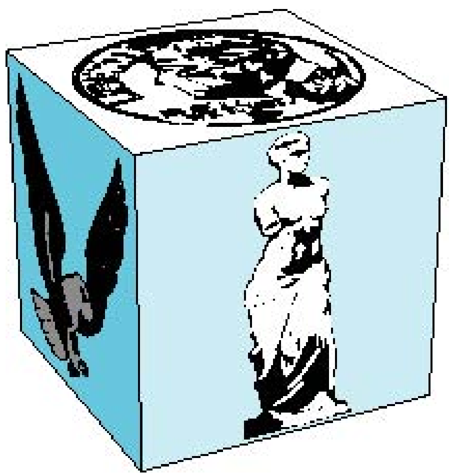
⇓

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

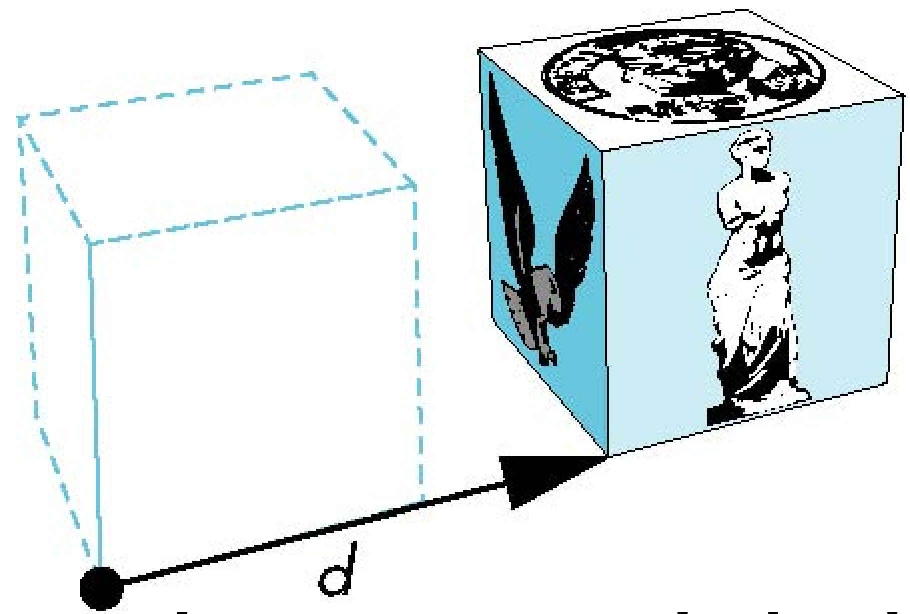
- Where: $x' = x.1 + y.0 + z.0 + tx.1 = x + tx$, … etc

3D Translation

- Move each vertex by same distance **d = (d_x, d_y, d_z)**



object



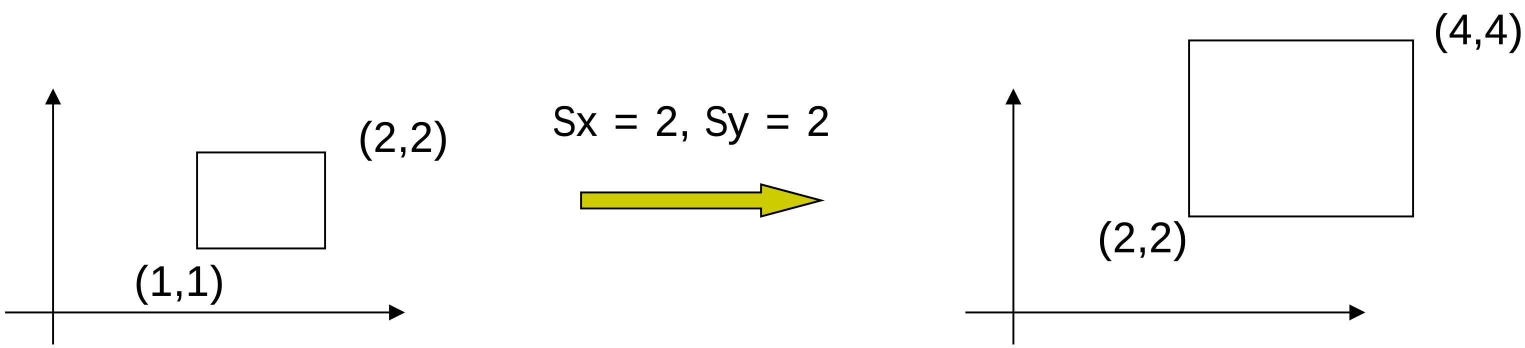
translation: every point displaced by same vector

2D Scaling

- Scale: Alter object size by scaling factor (s_x, s_y). **about origin**

$$x' = x \cdot Sx$$
$$y' = y \cdot Sy$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



2D Scaling Matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

4x4 3D Scaling Matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Example:
- If $Sx = Sy = Sz = 0.5$
- Can scale:
- big cube (sides = 1) to small cube (sides = 0.5)
- 2D: square, 3D cube

Scale(Sx,Sy,Sz)

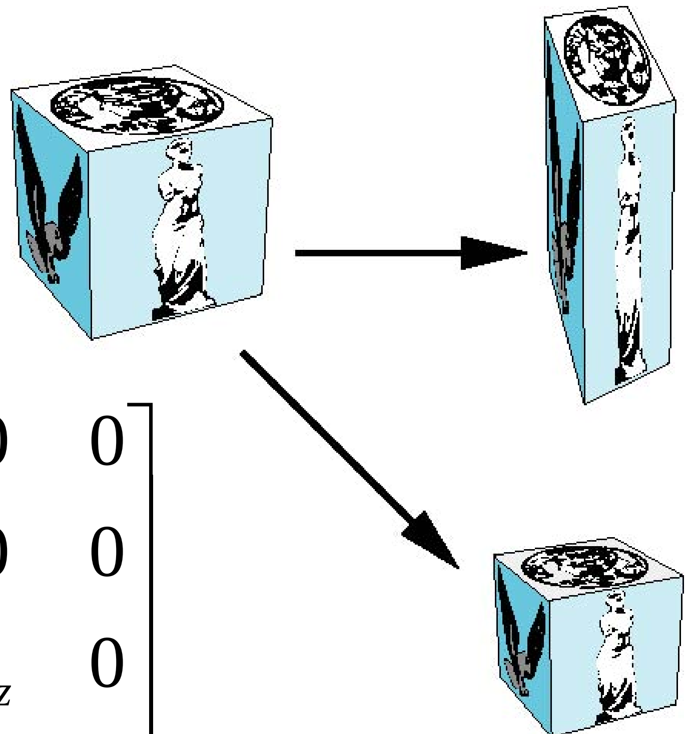
Scaling

Expand or contract along each axis (fixed point of origin)

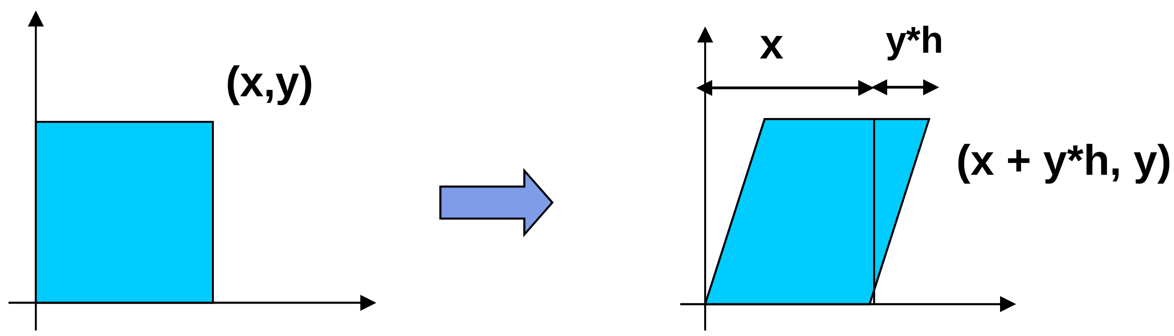
$x' = s_x x$
$y' = s_y y$
$z' = s_z z$

$\mathbf{p}' = \mathbf{S}\mathbf{p}$

$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Shearing

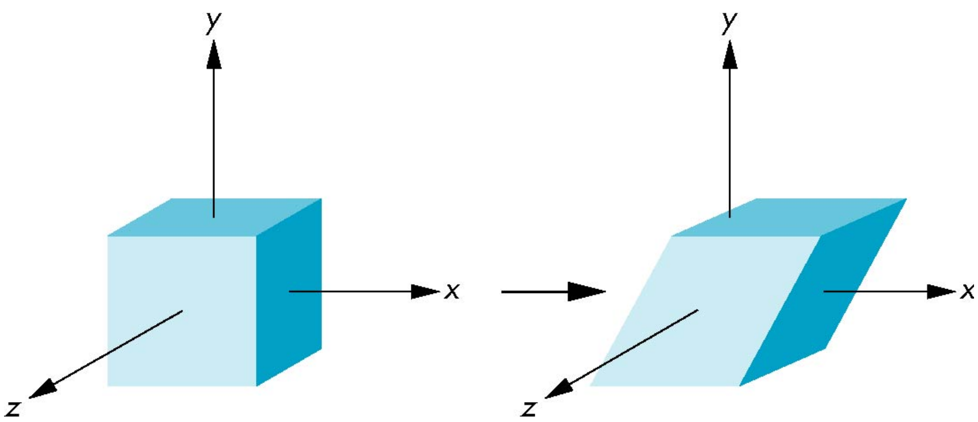


- Y coordinates are unaffected, but x cordinates are translated linearly with y
- That is:
  - **y' = y**
  - **x' = x + y * h**

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

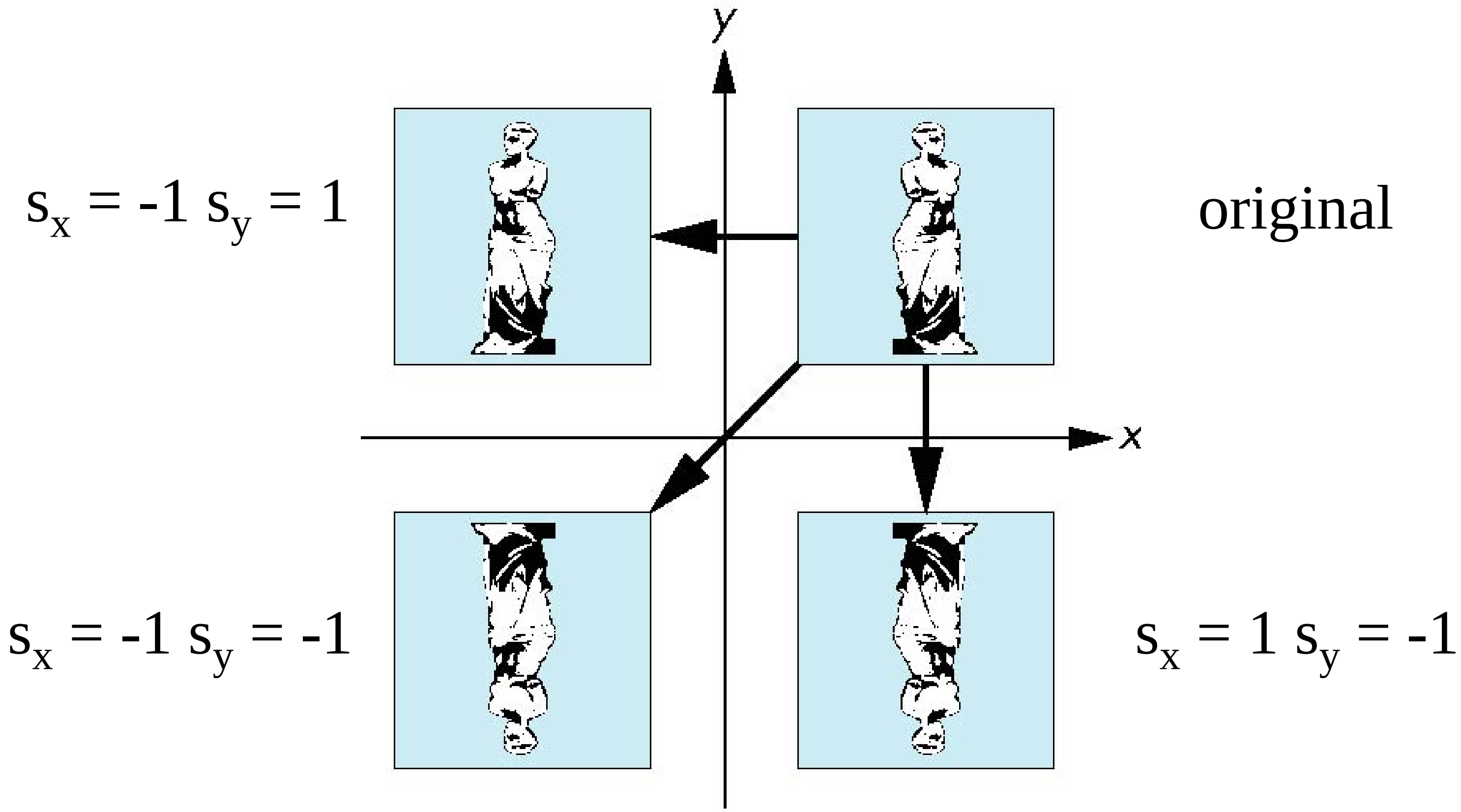
- h is fraction of y to be added to x

3D Shear



Reflection

corresponds to negative scale factors



References

- Angel and Shreiner, Interactive Computer Graphics, 6th edition, Chapter 3
- Hill and Kelley, Computer Graphics using OpenGL, 3rd edition