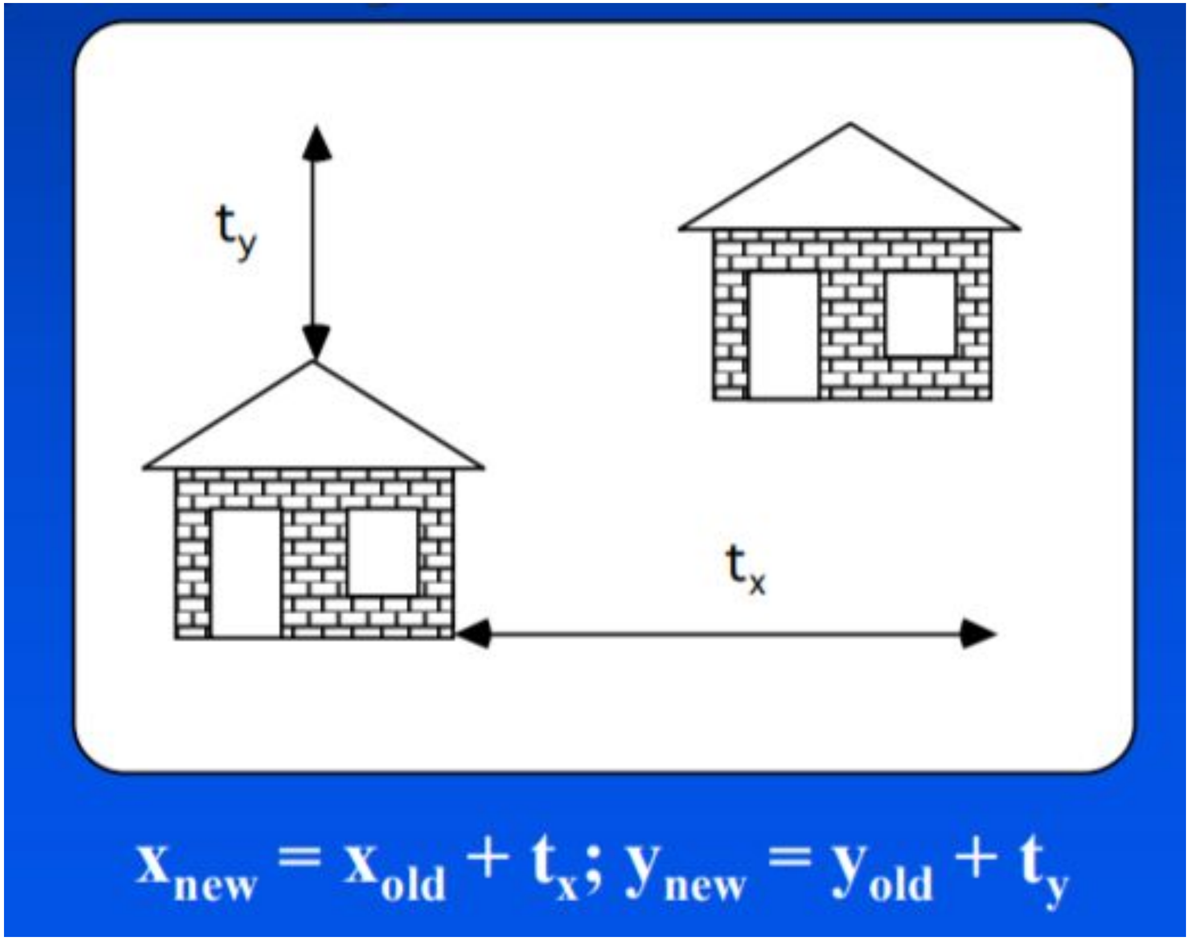
CS 174A
Discussion 1B-Week 3

01/24/2020

Slide credit: Bowman,Cao,Li,Terzopoulos

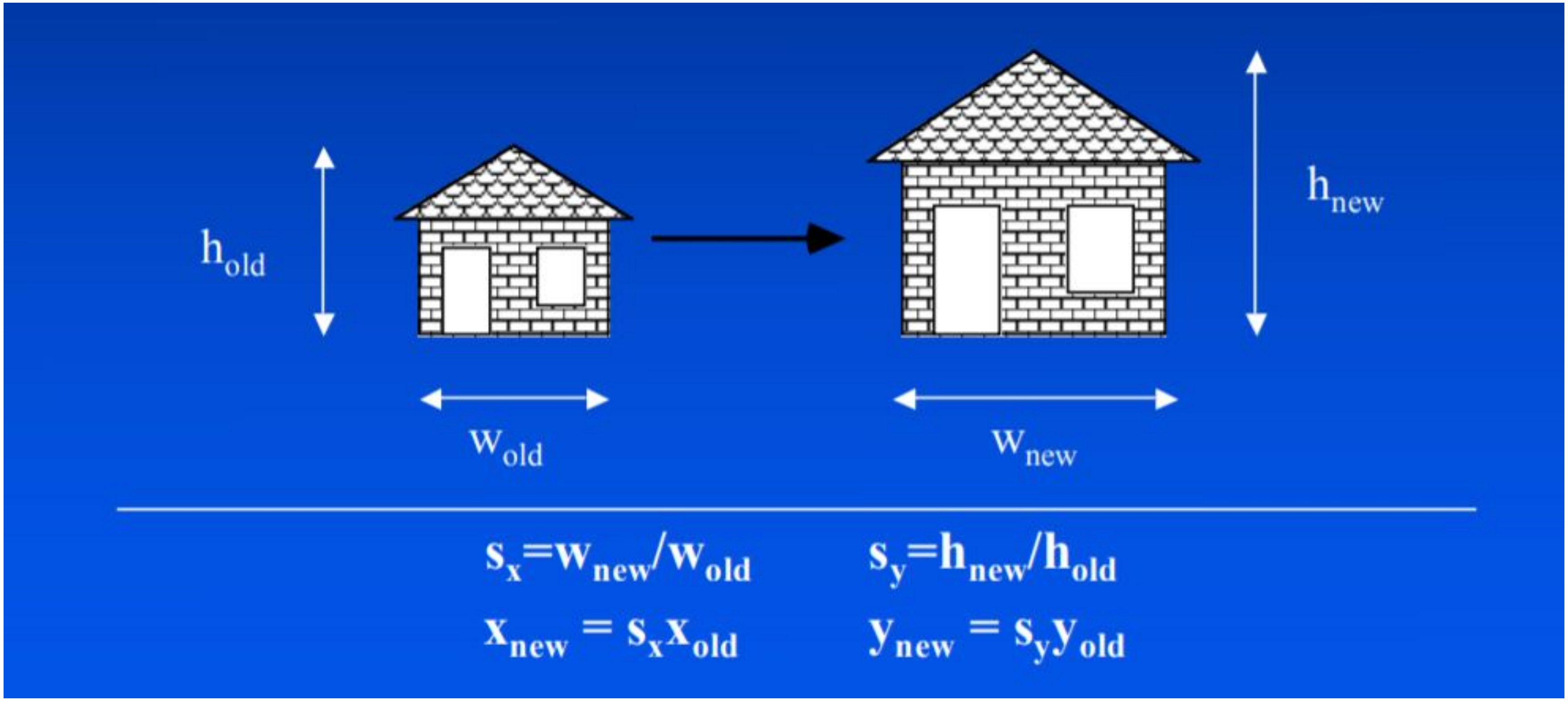
Translation

Shifting each vertex position



Scaling

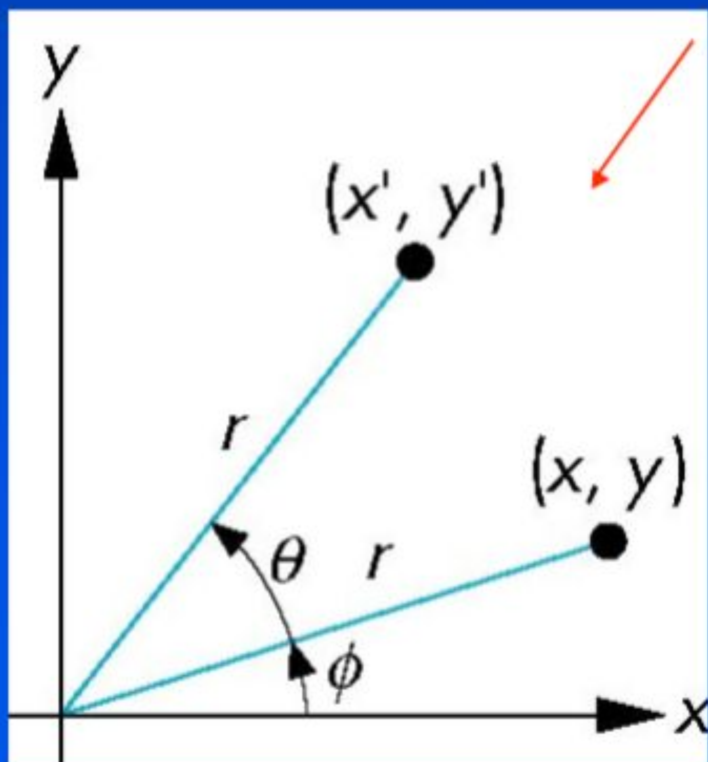
Changing the size of an object



Rotation

Rotation around origin

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



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

Rotation

Rotation around origin

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,

$$\boxed{\begin{aligned} x' &= x \cos\theta - y \sin\theta \\ y' &= x \sin\theta + y \cos\theta \end{aligned}}$$

Rotation-Scaling-Translation

Scale:

$x_{new} = s_x x_{old}$

$y_{new} = s_y y_{old}$

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

Rotation:

$x_2 = x_1 \cos\theta - y_1 \sin\theta$

$y_2 = x_1 \sin\theta + y_1 \cos\theta$

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

Translation:

$x_{new} = x_{old} + t_x$

$y_{new} = y_{old} + t_y$

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

Composite Transforms

[new]= [transform *n*] … [transform 2] [transform 1] [old]

$$P' = T_n \ldots T_2 T_1 P$$

– **Inefficient**

$$P' = T_n \ldots T_2 \left(T_1 P\right) \quad P' = T_n \left(\ldots \left(T_2 \left(T_1 P\right)\right)\right)$$

– **Efficient**

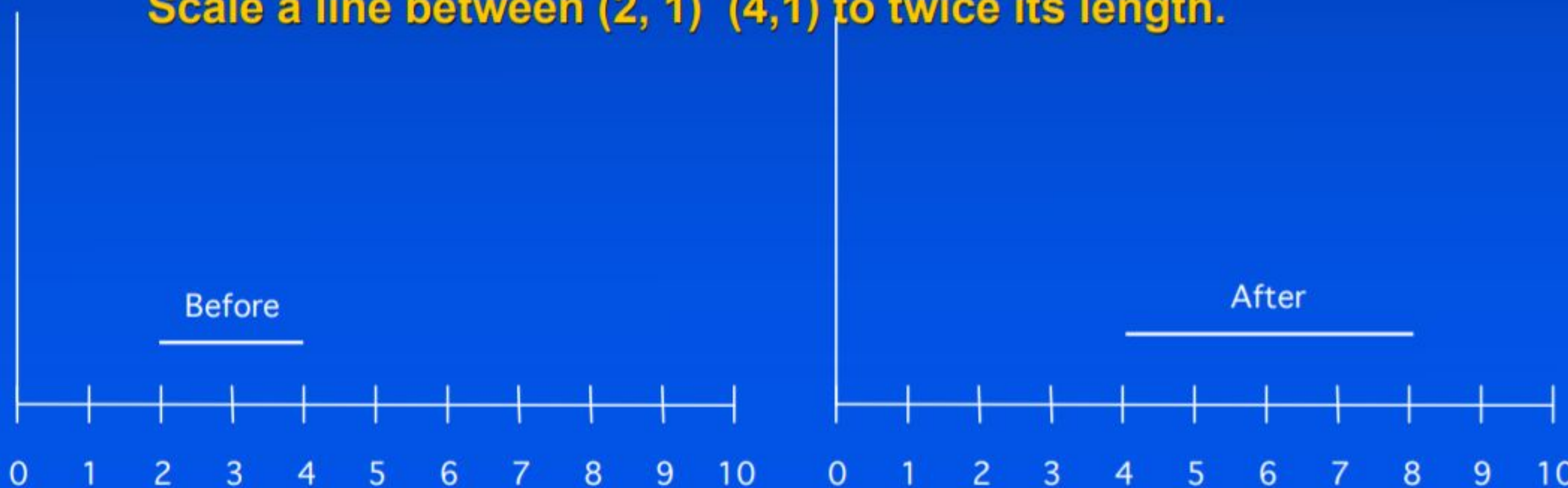
$$P' = (T_n \ldots T_2 T_1) P \qquad P' = \mathrm{T} P$$

Composite Transforms-Scaling

Given our three basic transformations we can create other transformations.

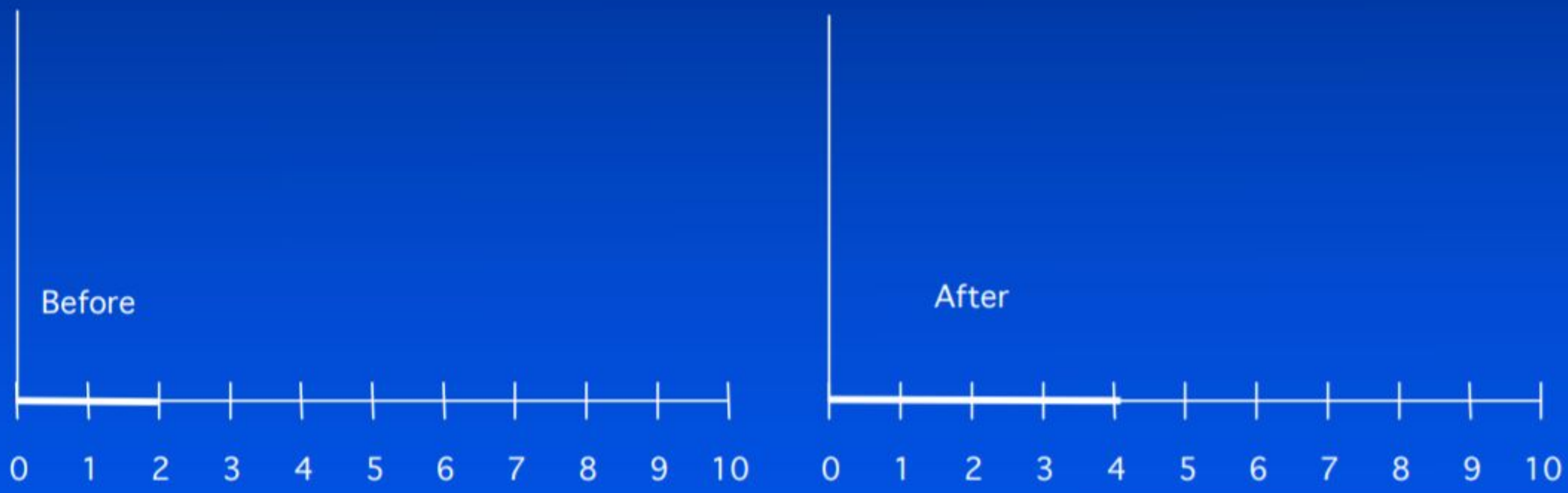
A problem with the scale transformation is that it also moves the object being scaled.

Scale a line between (2, 1) (4,1) to twice its length.



Composite Transforms-Scaling

If we scale a line between (0,0) & (2,0) to twice its length, the left-hand endpoint does not move.



(0,0) is known as a ***fixed point*** for the basic scaling transformation. We can use composite transformations to create a scale transformation with different fixed points.

Composite Transforms-Scaling

Scale by 2 with fixed point = (2,1)

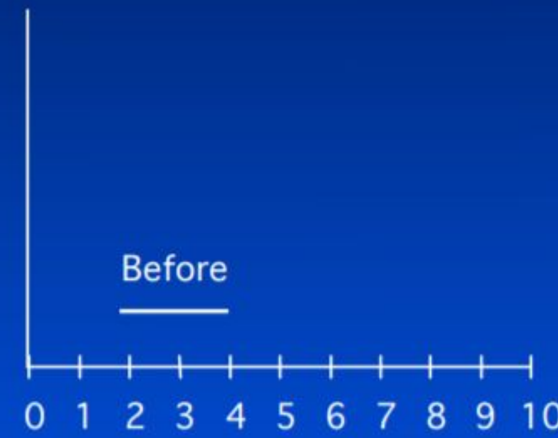
Translate the point (2,1) to the origin

Scale by 2

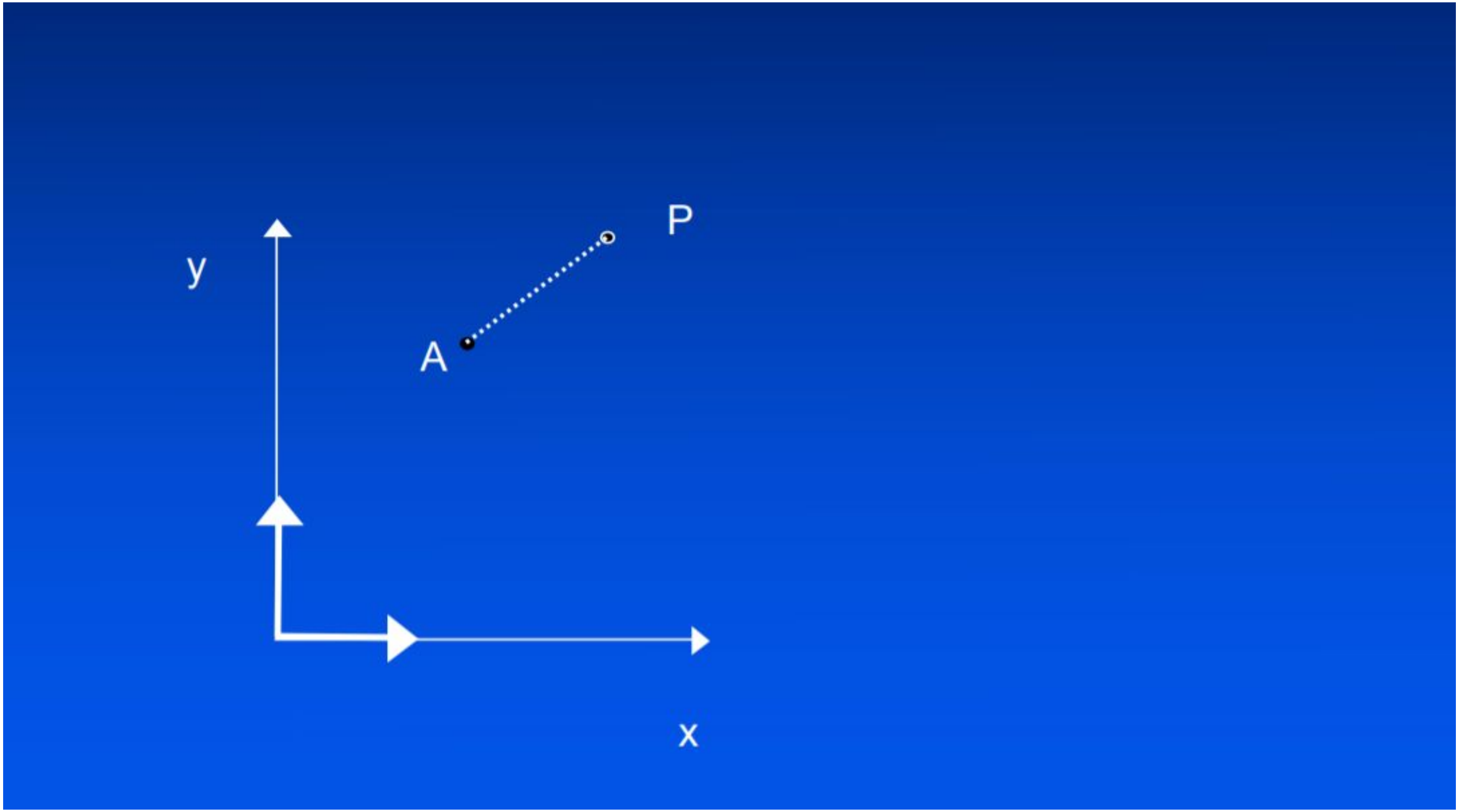
Translate origin to point (2,1)

$$\underbrace{\begin{pmatrix}1 & 0 & 2\\ 0 & 1 & 1\\ 0 & 0 & 1\end{pmatrix}}_{T_{2,1}}\underbrace{\begin{pmatrix}2 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}}_{S_{2,1}}\underbrace{\begin{pmatrix}1 & 0 & -2\\ 0 & 1 & -1\\ 0 & 0 & 1\end{pmatrix}}_{T_{-2,-1}}=\underbrace{\begin{pmatrix}2 & 0 & -2\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}}_{C}$$

$$\underbrace{\begin{pmatrix}2 & 0 & -2\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}}_{C}\begin{bmatrix}2\\ 1\\ 1\end{bmatrix}=\begin{bmatrix}2\\ 1\\ 1\end{bmatrix} \qquad \underbrace{\begin{pmatrix}2 & 0 & -2\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}}_{C}\begin{bmatrix}4\\ 1\\ 1\end{bmatrix}=\begin{bmatrix}6\\ 1\\ 1\end{bmatrix}$$



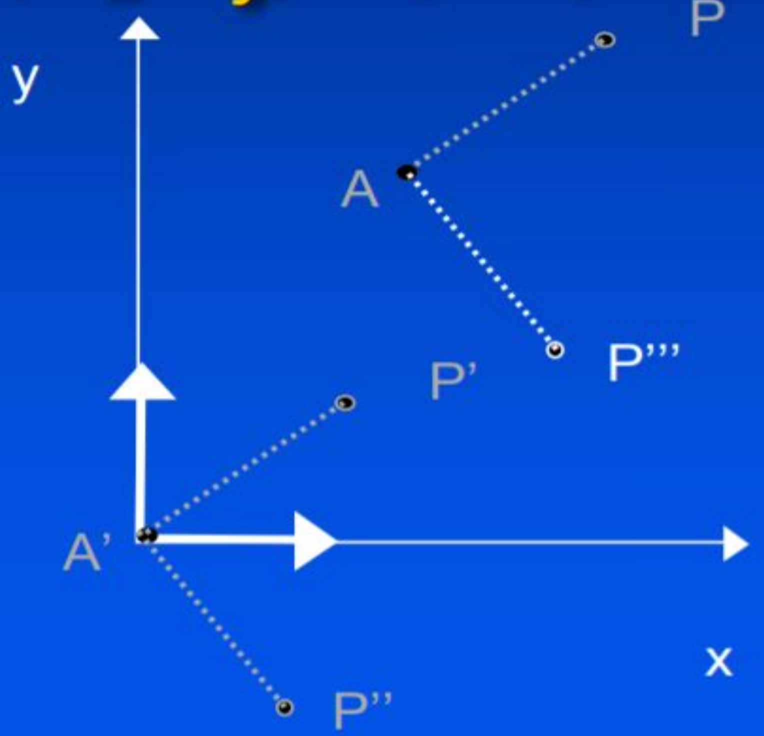
Rotation About an Arbitrary Point



Rotation About an Arbitrary Point

IMPORTANT!: Order

$$M = T(A_x, A_y)\, R(-90)\, T(-A_x, -A_y)$$



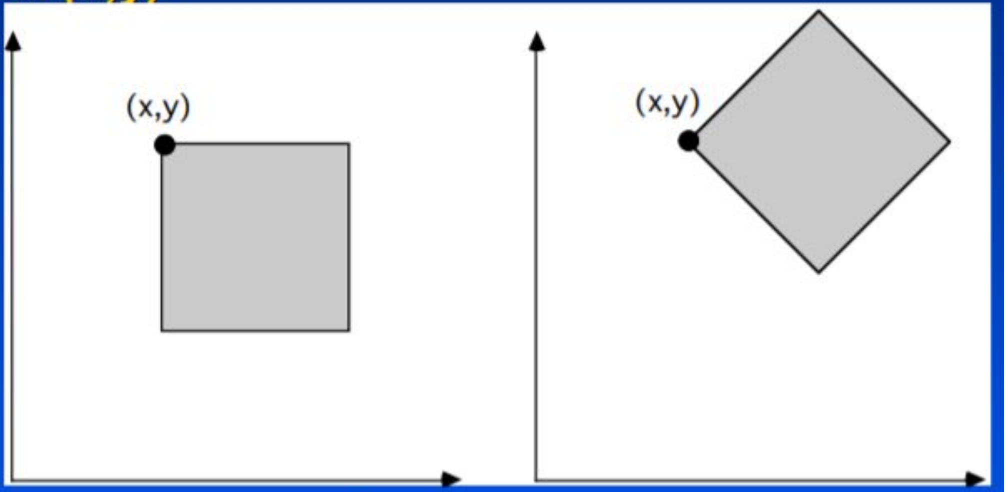
Rotation About an Arbitrary Point

Rotation Of θ Degrees About Point (x,y)

Translate (x,y) to origin

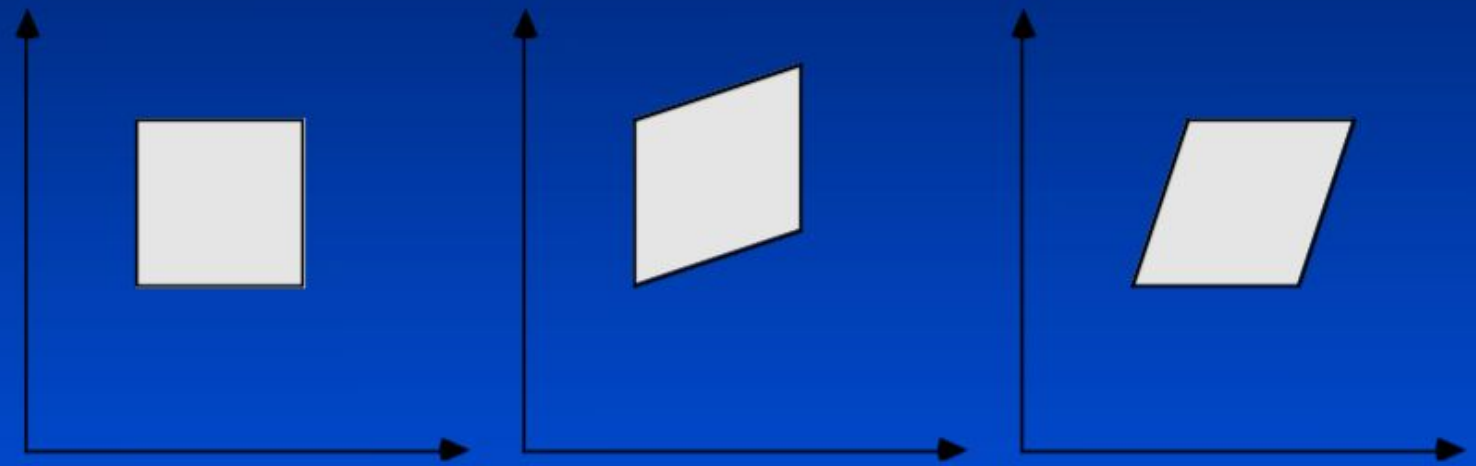
Rotate

Translate origin to (x,y)



$$C = \underbrace{\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}}_{T_{x,y}} \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_\theta} \underbrace{\begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{bmatrix}}_{T_{-x,-y}}$$

Shear



Original Data

y Shear

x Shear

$$\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear

$$x' = x + \mathrm{Sh}_x^y y + \mathrm{Sh}_x^z z$$

$$y' = \mathrm{Sh}_y^x x + y + \mathrm{Sh}_y^z z$$

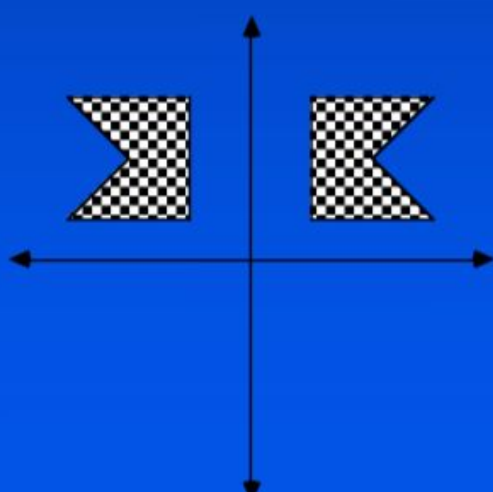
$$z' = \mathrm{Sh}_z^x x + \mathrm{Sh}_z^y y + z$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & \mathrm{Sh}_x^y & \mathrm{Sh}_x^z & 0 \\ \mathrm{Sh}_y^x & 1 & \mathrm{Sh}_y^z & 0 \\ \mathrm{Sh}_z^x & \mathrm{Sh}_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Reflection

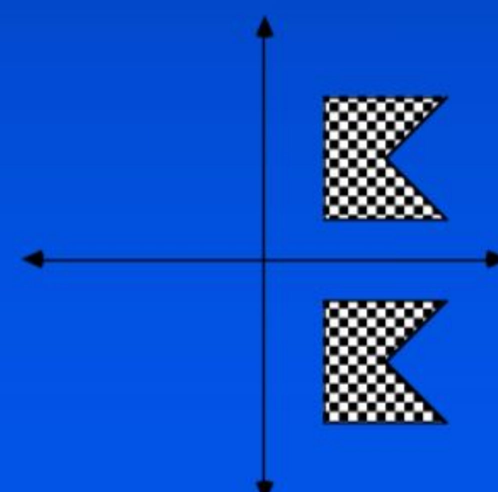
Reflection about the y-axis

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



Reflection about the x-axis

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



Reflection

To reflect a point through a plane $ax + by + cz = 0$ (which goes through the origin), if the L2 norm of a, b and c is unity, the transformation matrix can be expressed as:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1-2a^2 & -2ab & -2ac & 0 \\ -2ab & 1-2b^2 & -2bc & 0 \\ -2ac & -2bc & 1-2c^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Let $n = (a, b, c)$, and $v = \frac{n}{\|n\|}$ be the plane's normal unit vector, $x = (i, j, k)$ a given vector; then we need to subtract its projection onto v twice to reflect it in the plane: $x - 2v(x \cdot v)$. In coordinates: $(I - 2vv^T)x$

Affine Transformations in 3D

General form

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

Or:

$$Q = \mathbf{M}P$$

General Form

Rotation / Scaling / Shearing

Translation

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Elementary 3D Affine Transformations

Translation

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Scaling Around the Origin

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Shear Around the Origin

Along x-axis

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Transformations in OpenGL

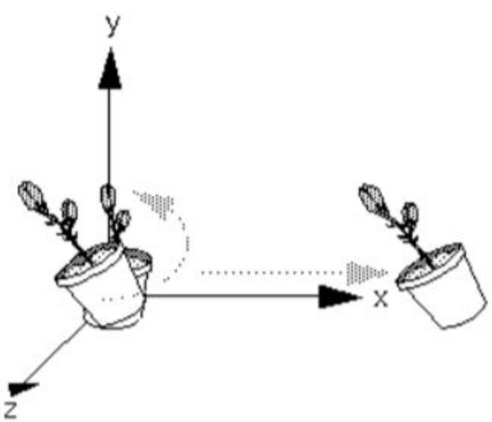
- Call order is the reverse of the order the transforms are applied.
- Different call orders result in different transforms!

```
// Example 1
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0,0.0,-6.0);
glRotatef(45.0,0.0,1.0,0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

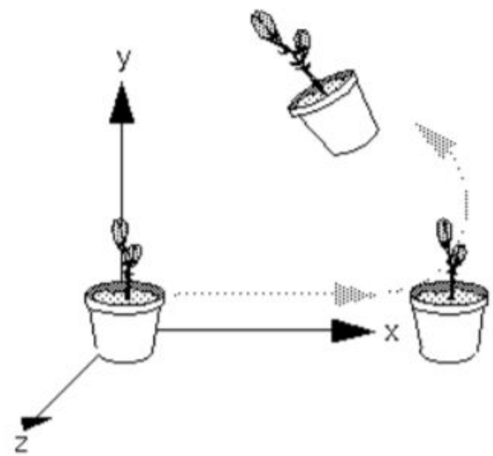
```
// Example II
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(45.0,0.0,1.0,0.0);
glTranslatef(0.0,0.0,-6.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

Transformations in OpenGL

- Each transform multiplies the object by a matrix that does the corresponding transformation.
- The transform closest to the object gets multiplied first.



Rotation first



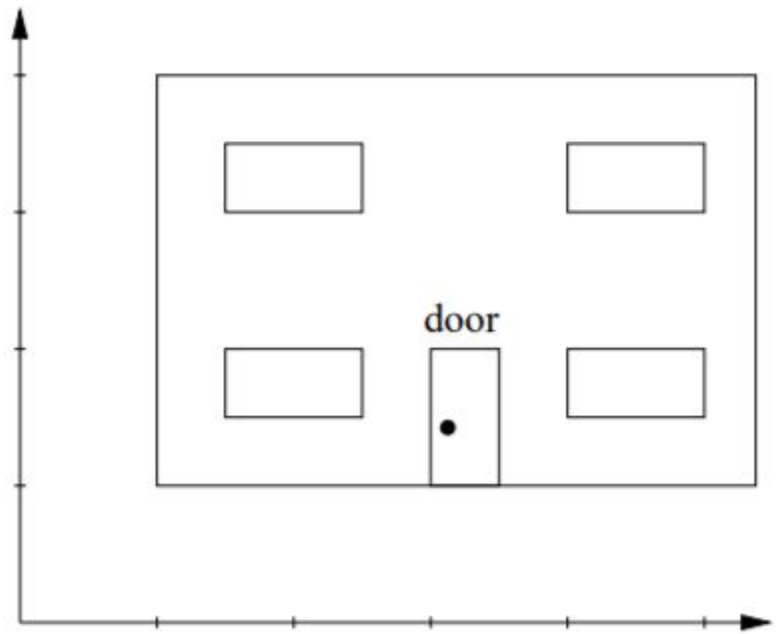
Translation first

Transformations in OpenGL

- Let
  - glTranslate = Mat Trans
  - glRotate = Mat Rot
  - glScale = Mat Scale
  - DrawCube = v
- Modelview matrix:
  - Identity -> Trans -> Trans*Rot -> Trans*Rot*Scale -> Trans*Rot*Scale*v
  - Or, Trans(Rot(Scale*v))).
  - So Scale is applied first, then Rot, then Trans

```
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -6.0);
glRotatef(45.0,0.0,1.0, 0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

Instancing



$$\mathrm{Trans}(3,1) \circ \mathrm{Scale}(0.5,1) = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$