Hochiminh city University of Technology
Faculty of Computer Science and Engineering

COMPUTER GRAPHICS

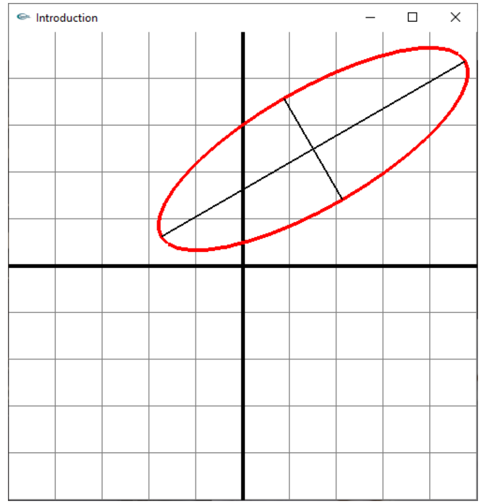
CHAPTER 06:

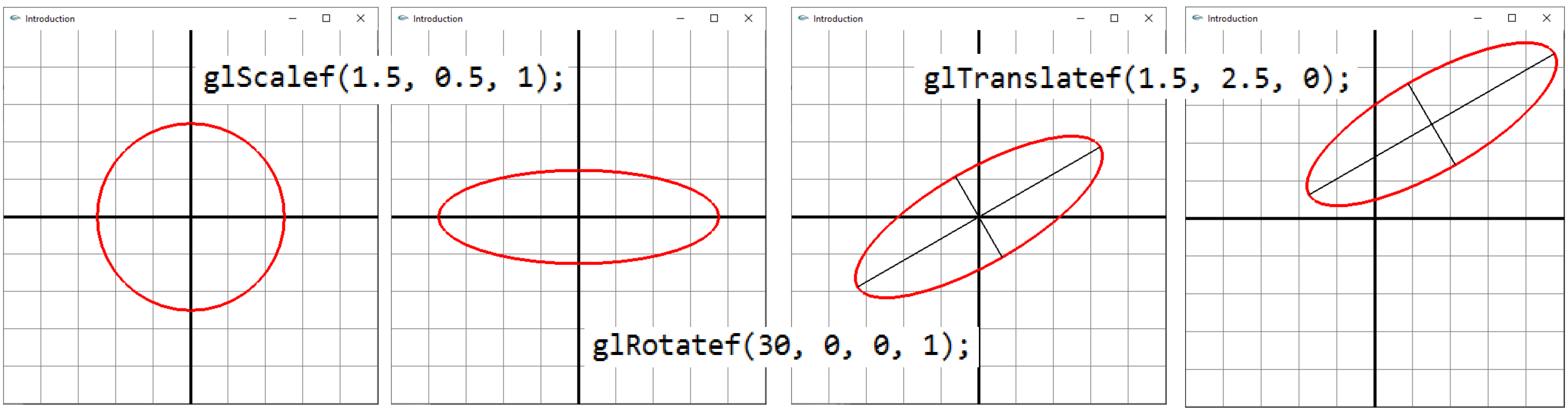
Transformations

OUTLINE

- Introduction
- Transformation in 2D
- Transformation in 3D
- Transformation in OpenGL

Introduction

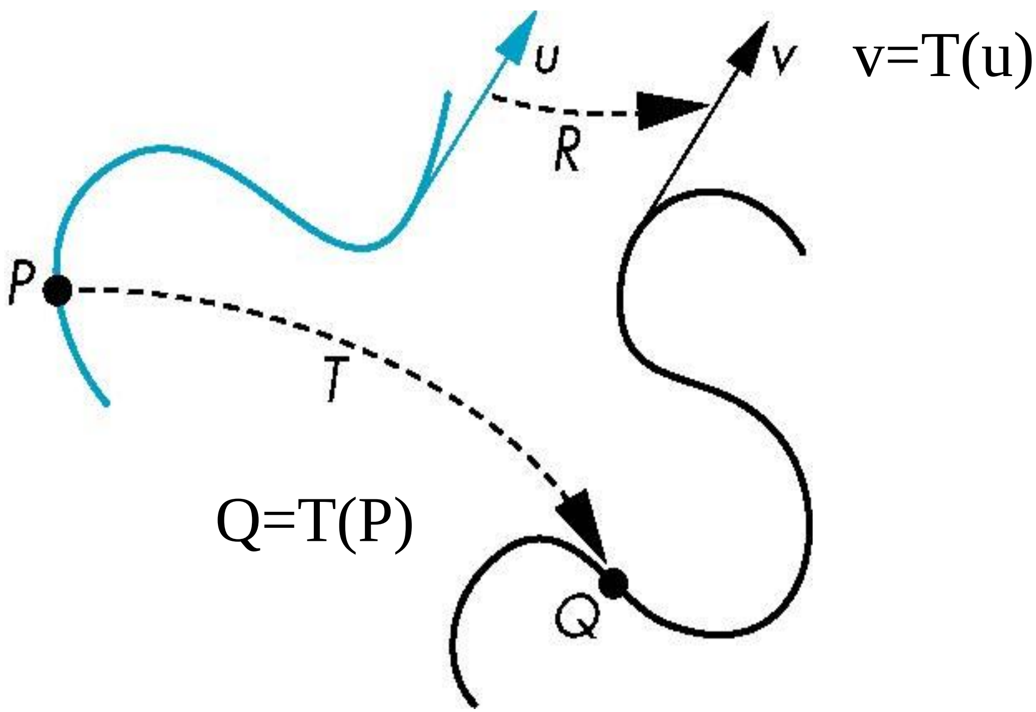




Introduction

- General Transformations

  A transformation maps points to other points and/or vectors to other vectors



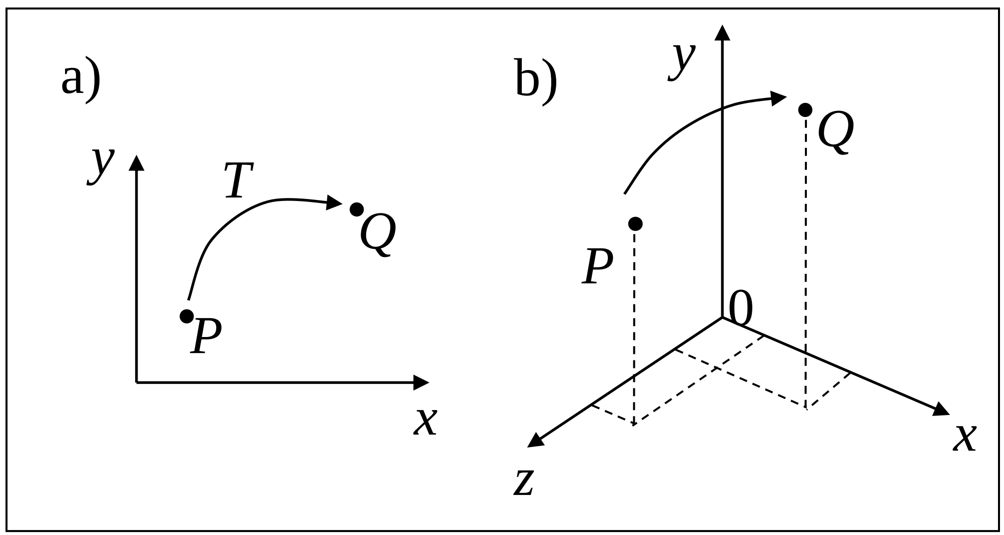
Introduction

General Transformations

P = (Px, Py, 1); Q = (Qx, Qy, 1) (**Q** - image)

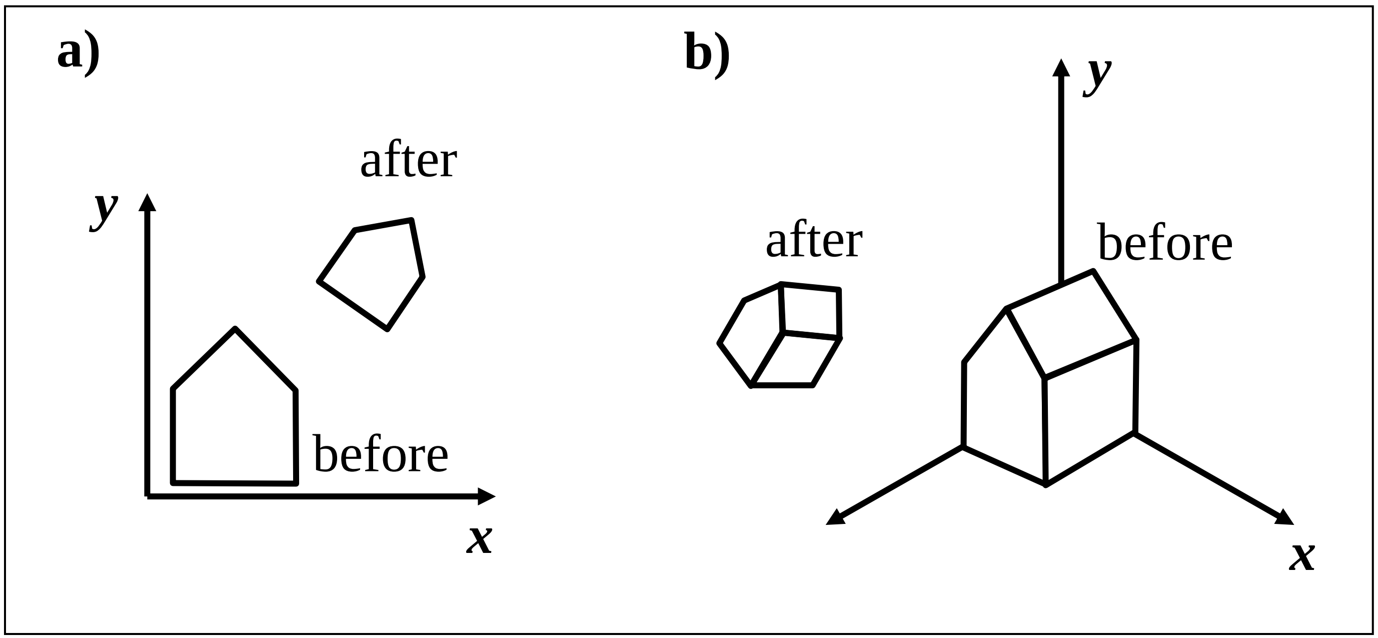
(Qx, Qy, 1) = T (Px, Py, 1) (**T** – transformation)

Q = T(P).



Introduction

❑ General Transformations



Introduction

Affine Transformations

$$Q_x = m_{11}P_x + m_{12}P_y + m_{13}$$

$$Q_y = m_{21}P_x + m_{22}P_y + m_{23}$$

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

T

always (0, 0, 1)

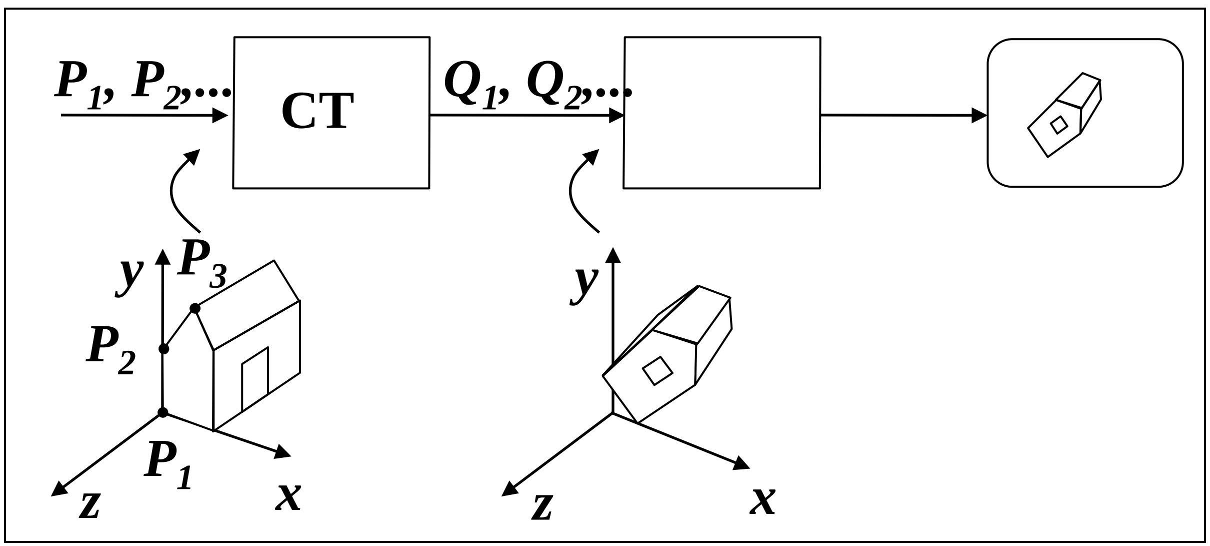
Introduction

- Affine Transformations
  - Line preserving
  - Characteristic of many physically important transformations
    - **Rigid body transformations: rotation, translation**
    - **Scaling, shear**
  - Importance in graphics is that we need only transform endpoints of line segments and let implementation draw line segment between the transformed endpoints

Introduction

- Pipeline Implementation

```
glBegin(GL_LINES);
    glVertex3f(. . .);
    glVertex3f(. . .);
    glVertex3f(. . .);
glEnd();
```



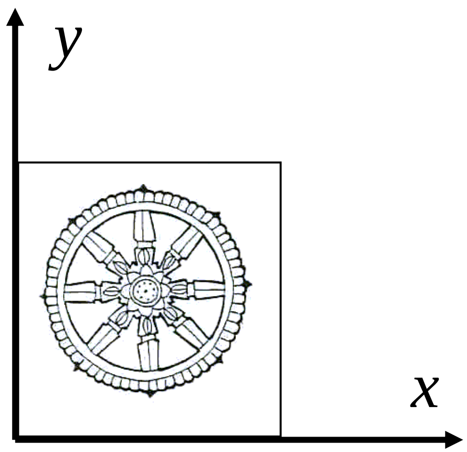
Transformations in 2D

- Translation

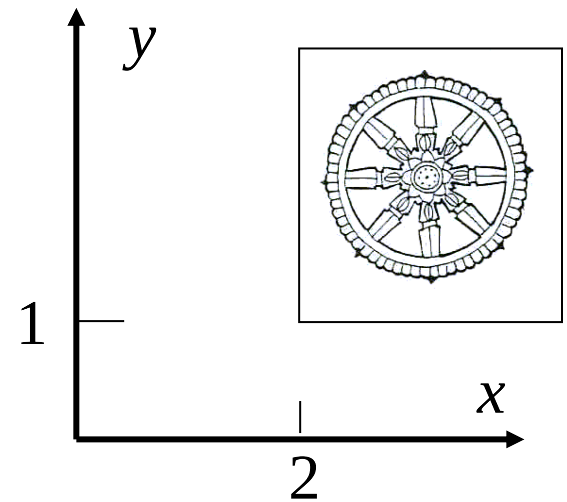
  Qx = Px + m13

  Qy = Py + m23

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & m_{13} \\ 0 & 1 & m_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$



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



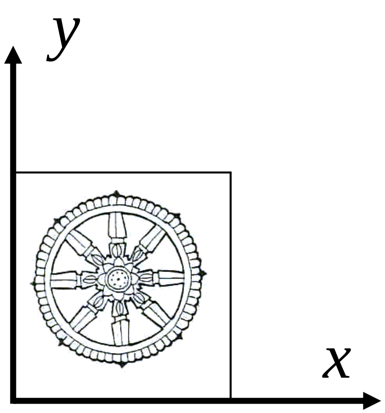
Transformations in 2D

- Scaling

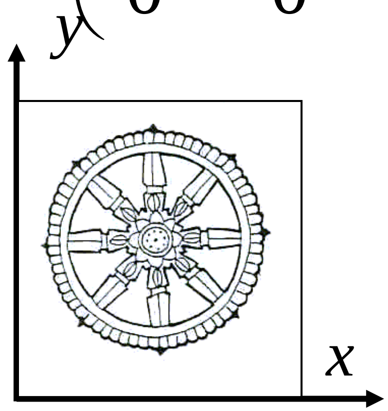
  Qx = Sx Px

  Qy = Sy Py

$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

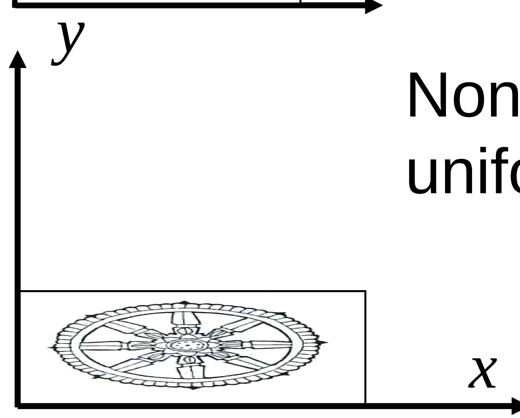


$$\begin{pmatrix} \boxed{1.3} & 0 & 0 \\ 0 & \boxed{1.3} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



uniform

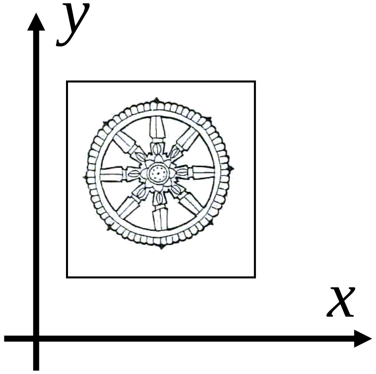
$$\begin{pmatrix} \boxed{1.5} & 0 & 0 \\ 0 & \boxed{0.5} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



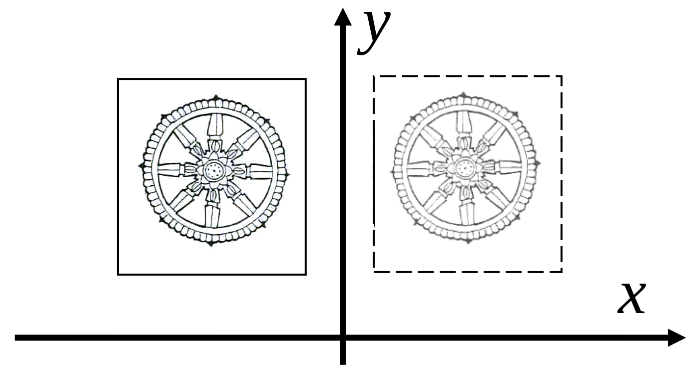
Non-uniform

Transformations in 2D

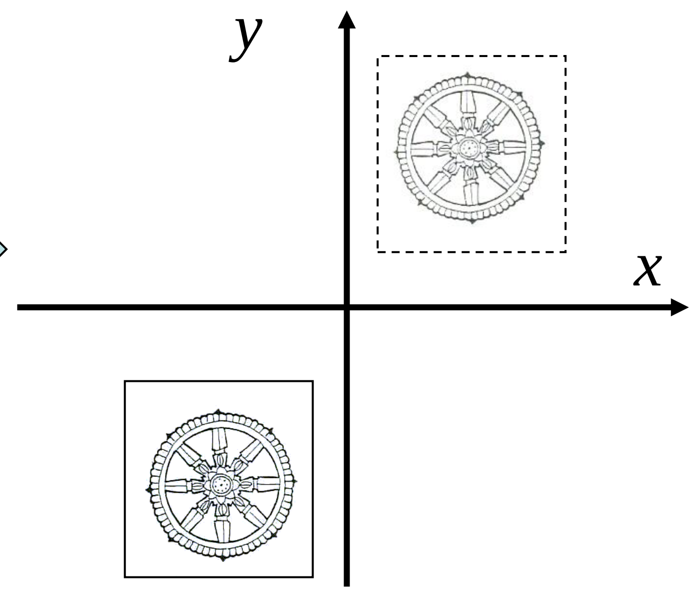
- Scaling (Reflection)



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

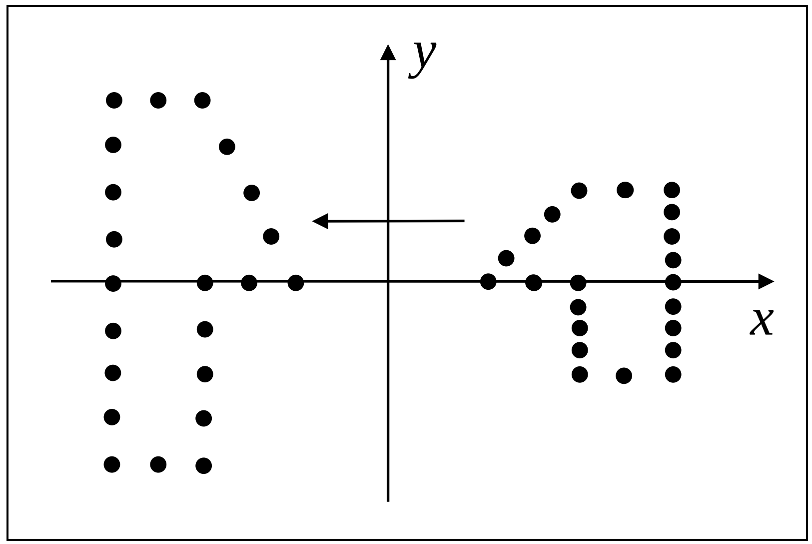


$$\begin{pmatrix} \boxed{-1} & 0 & 0 \\ 0 & \boxed{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Transformations in 2D

❑ Scaling (Reflection)



$$\begin{pmatrix} \boxed{-1} & 0 & 0 \\ 0 & \boxed{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformations in 2D

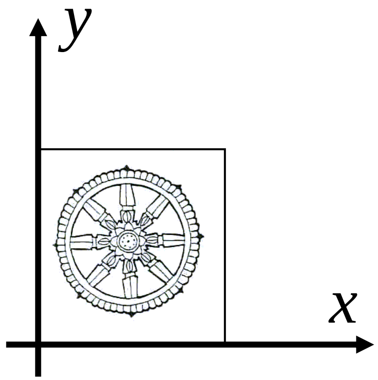
- Rotation

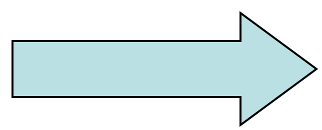
  $Q_x = P_x \cos(\theta) - P_y \sin(\theta)$

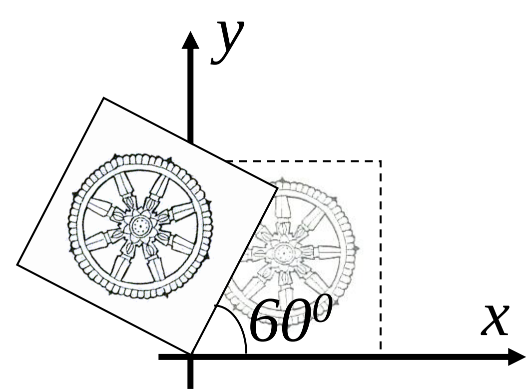
  $Q_y = P_x \sin(\theta) + P_y \cos(\theta)$

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

$$\begin{pmatrix} 0.5 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

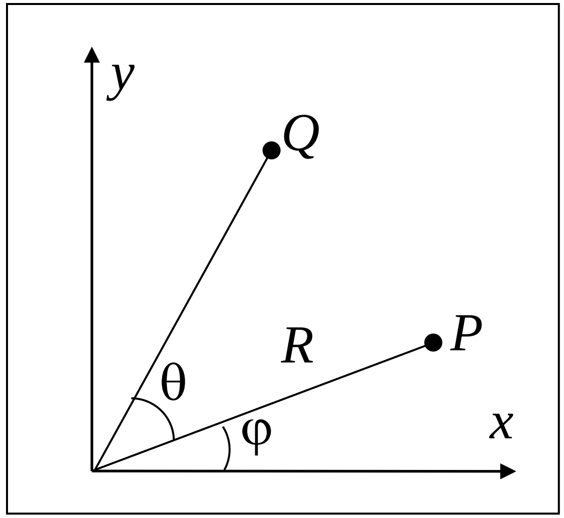






Transformations in 2D

❑ Rotation



$$Q_x = R\cos(\theta + \varphi)$$

$$Q_y = R\sin(\theta + \varphi)$$

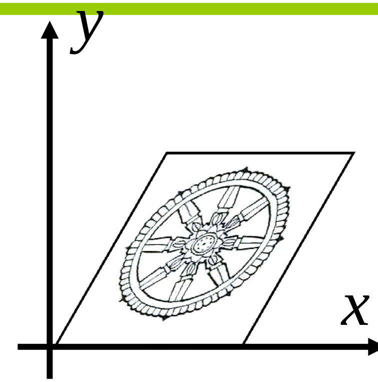
$$Q_x = R\cos\theta\cos\varphi - R\sin\theta\sin\varphi = P_x\cos\theta - P_y\sin\theta$$

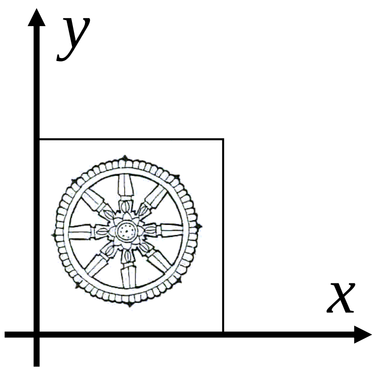
$$Q_y = R\sin\theta\cos\varphi + R\cos\theta\sin\varphi = P_x\sin\theta + P_y\cos\theta$$

Transformations in 2D

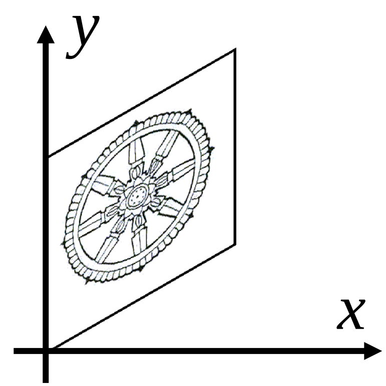
- Shear

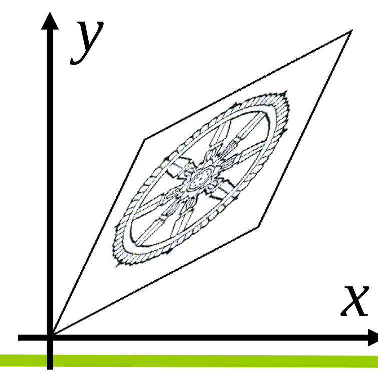
$$\begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$





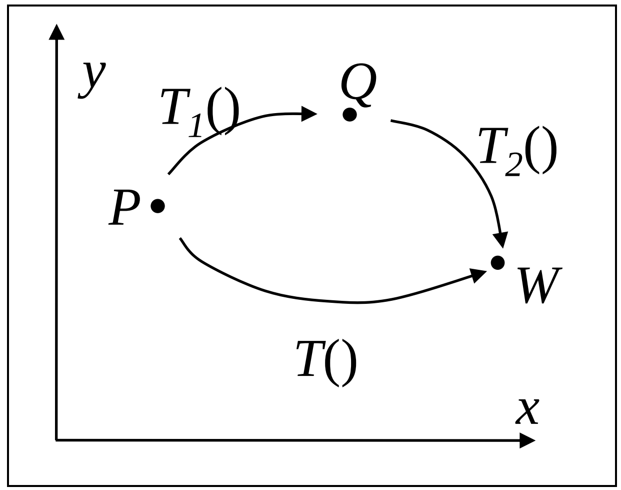
$$\begin{pmatrix} 1 & 0 & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$





Transformations in 2D

- Concatenation

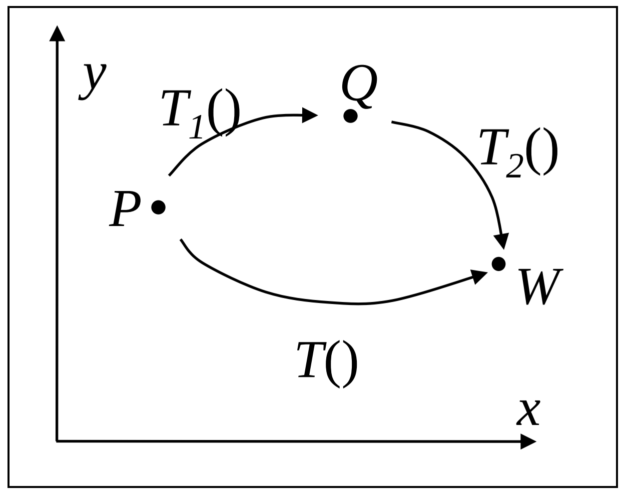


Transformations in 2D

- Concatenation
  - We can form arbitrary affine transformation matrices by multiplying together rotation, translation, and scaling matrices
  - Because the same transformation is applied to many vertices, the cost of forming a matrix $\mathbf{M}=\mathbf{ABCD}$ is not significant compared to the cost of computing $\mathbf{Mp}$ for many vertices $\mathbf{p}$

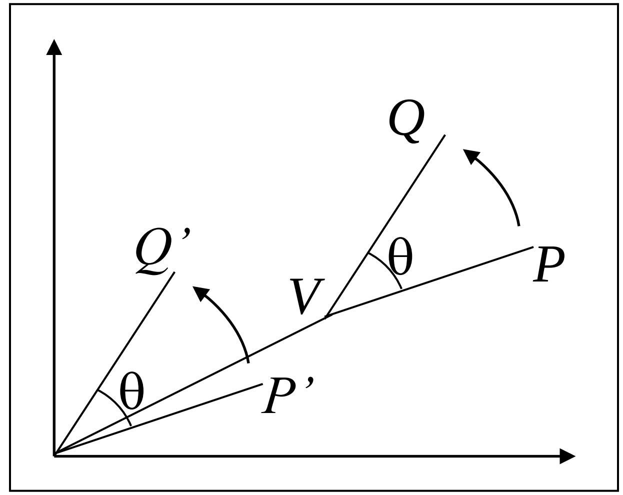
Transformations in 2D

- Concatenation
  - Note that matrix on the right is the first applied
  - Mathematically, the following are equivalent
    - $\mathbf{W = T_2\, Q = T_2\, (T_1\, P) = (T_2\, T_1)P}$



Transformations in 2D

- Concatenation
  - Move fixed point to origin
  - Rotate
  - Move fixed point back
  - **M** = **T**(p_f) **R**(θ) **T**(-p_f)

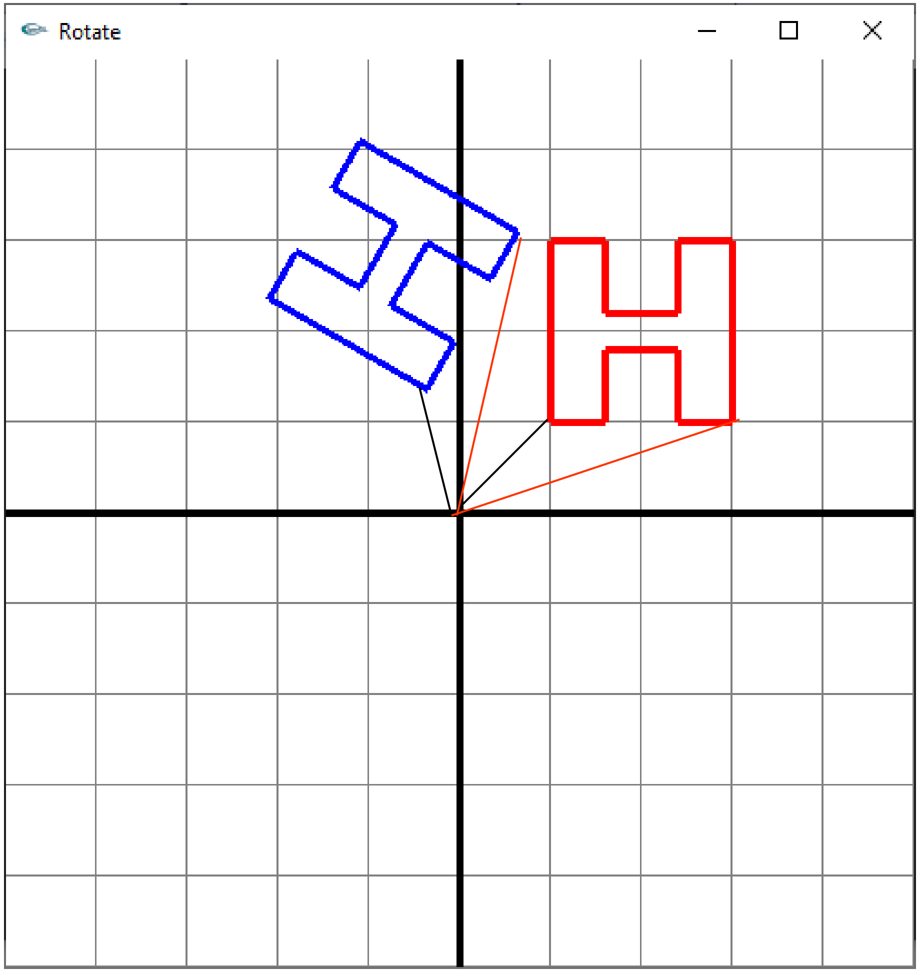


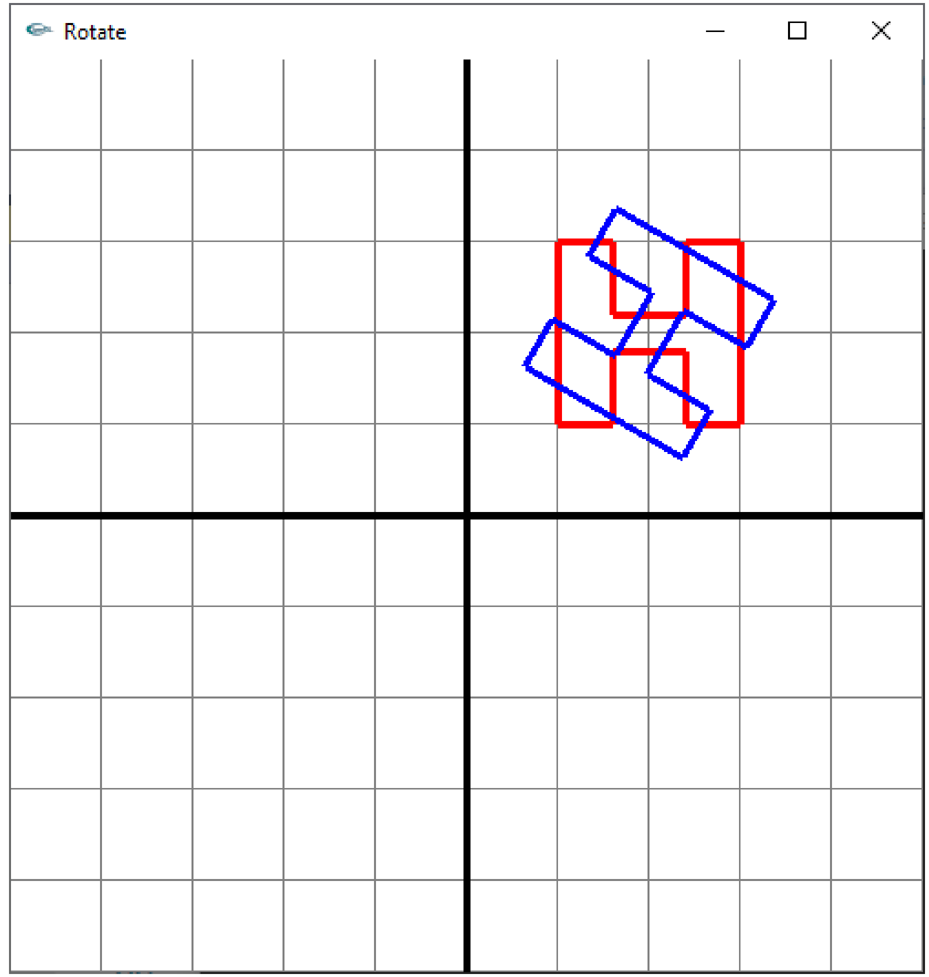
$$\begin{pmatrix} 1 & 0 & V_x \\ 0 & 1 & V_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -V_x \\ 0 & 1 & -V_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos(\theta) & -\sin(\theta) & d_x \\ \sin(\theta) & \cos(\theta) & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$d_x = -V_x \cos(\theta) + V_y \sin(\theta) + V_x$$
$$d_y = -V_x \sin(\theta) - V_y \cos(\theta) + V_y$$

Transformations in 2D





Transformations in 3D

- General Formula

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

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = M \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

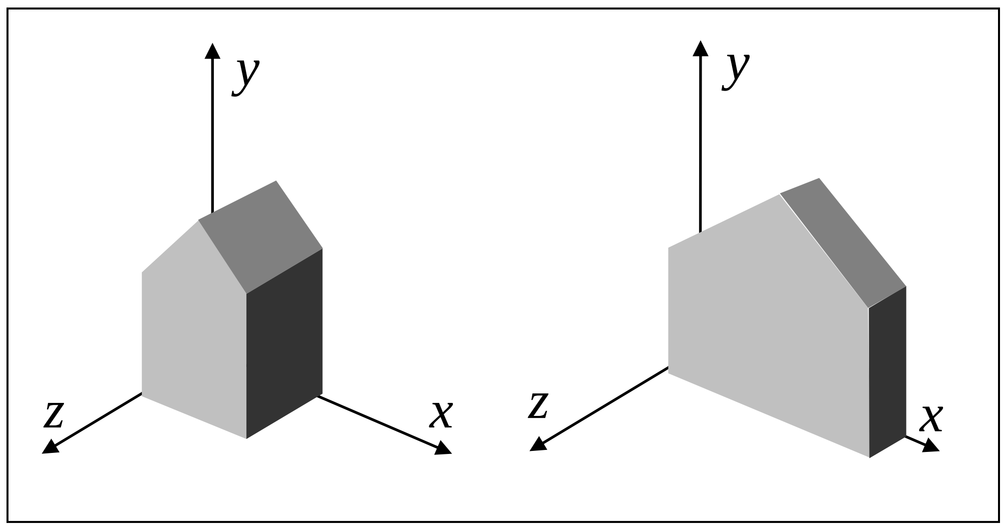
Transformations in 3D

- Translation

$$\begin{pmatrix} 1 & 0 & 0 & m_{14} \\ 0 & 1 & 0 & m_{24} \\ 0 & 0 & 1 & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Scaling

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



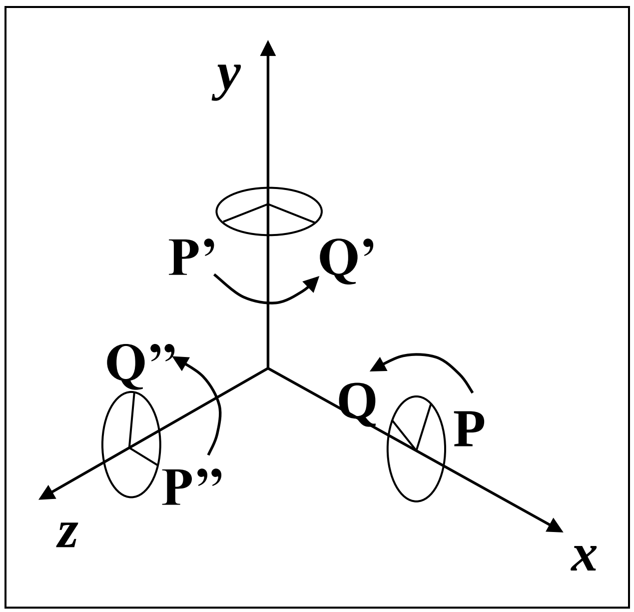
Transformations in 3D

- **Shear**

$$\begin{pmatrix} 1 & \boxed{0} & \boxed{0} & 0 \\ \boxed{f} & 1 & \boxed{0} & 0 \\ \boxed{0} & \boxed{0} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$Q = (P_x, fP_x + P_y, P_z)$

- **Rotation**



- ✓ x-roll, y-roll, z-roll
- ✓ when angle = 90^0:

z-roll: x→y

x-roll: y→z

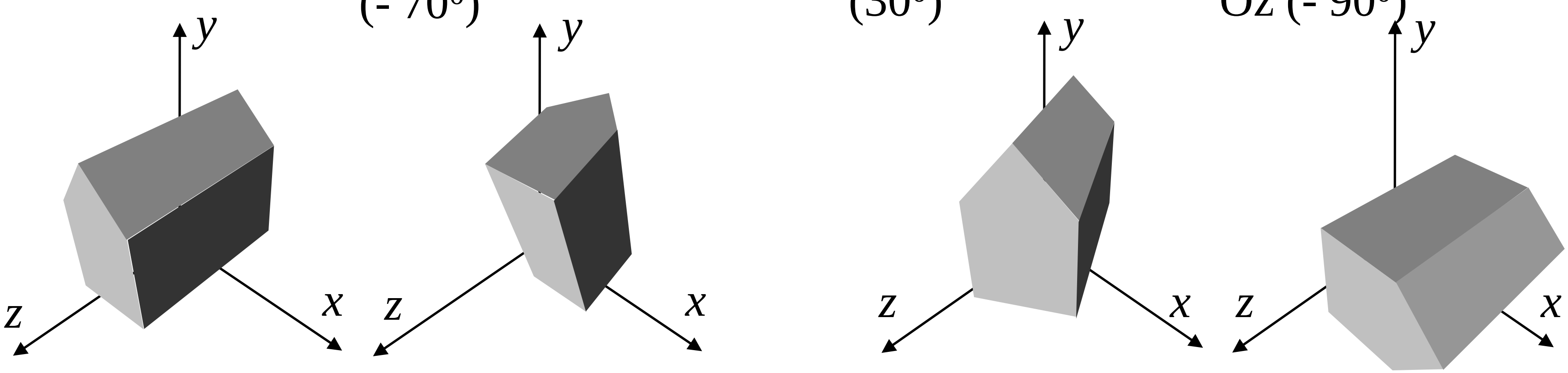
y-roll: z→x

Transformations in 3D

Rotation

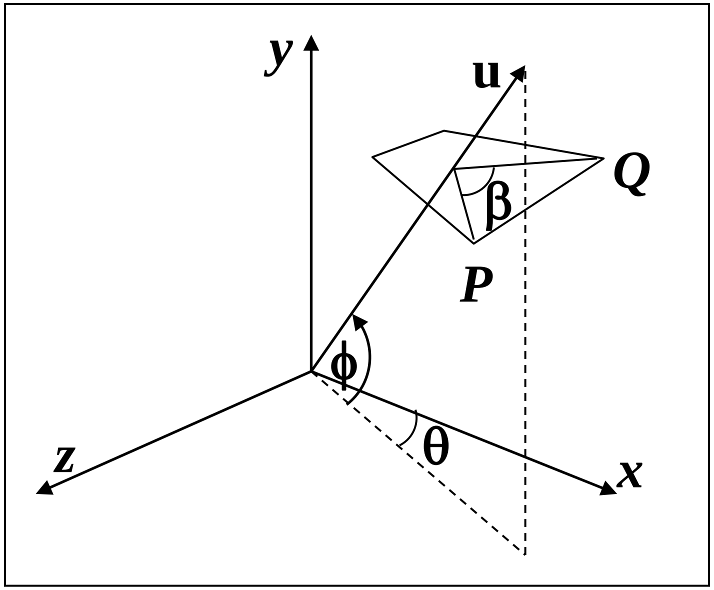
$$R_x(\beta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y(\beta) = \begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_z(\beta) = \begin{pmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a) object | b) Rotate about Ox (- 70^0) | c) Rotate about Oy (30^0) | d) Rotate aboute Oz (- 90^0)



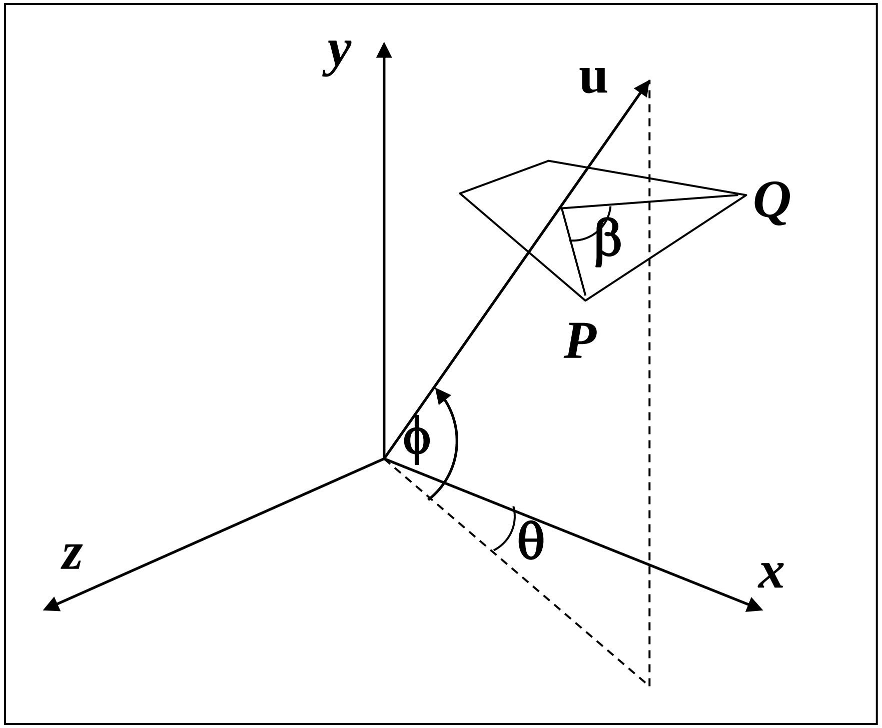
Transformations in 3D

❑ Rotation about an Arbitrary Axis



- ✓ u → x.
- ✓ Rotate about x axis with angle β.
- ✓ recover u.

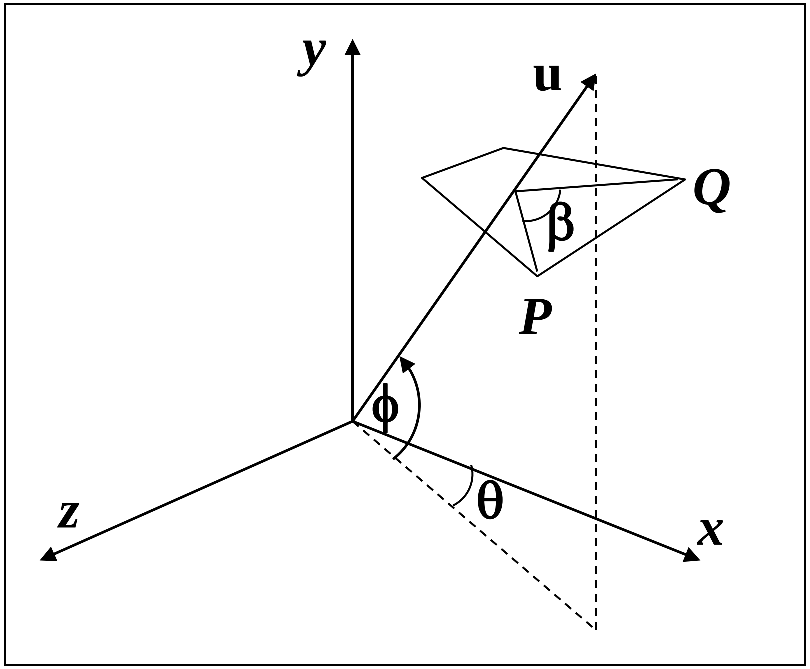
Transformations in 3D



- u → x
  - Rotate u around y (+θ)
  - Then rotate around z (-φ)
- Rotate around x (+β)
- Recover u
  - Rotate around z (+φ)
  - Rotate around y (-θ)

$$R_u(\beta)=R_y(-\theta)R_z(\phi)R_x(\beta)R_z(-\phi)R_y(\theta)$$

Transformations in 3D



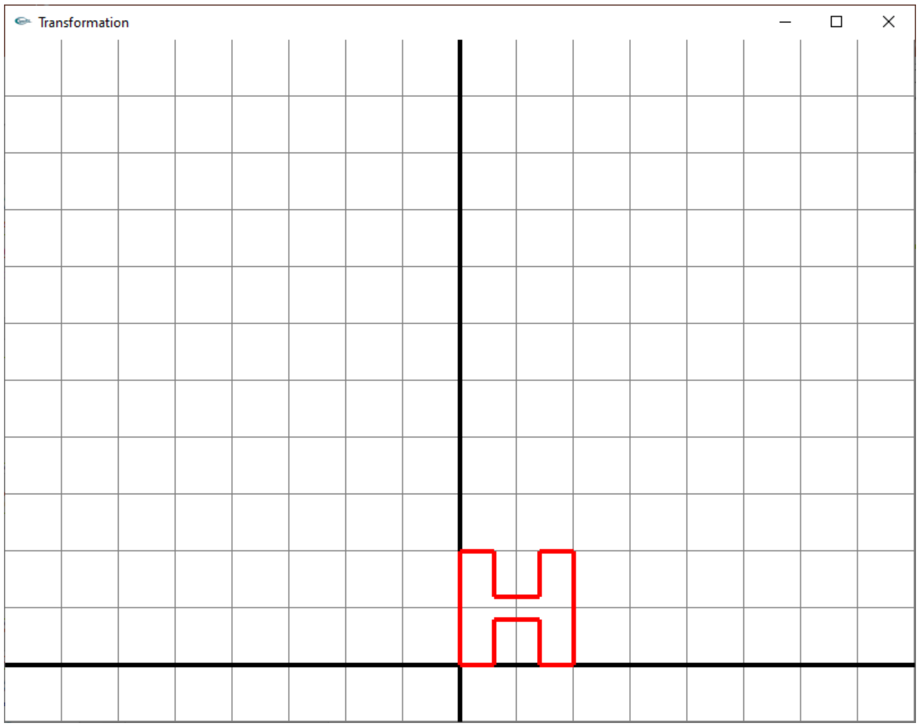
$$R_u(\beta)=R_y(-\theta)R_z(\phi)R_x(\beta)R_z(-\phi)R_y(\theta)$$

$$\begin{pmatrix} c+(1-c)u_x^2 & (1-c)u_yu_x - su_z & (1-c)u_zu_x + su_y & 0 \\ (1-c)u_xu_y + su_z & c+(1-c)u_y^2 & (1-c)u_zu_y - su_x & 0 \\ (1-c)u_xu_z - su_y & (1-c)u_yu_z + su_x & c+(1-c)u_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformations in OpenGL

- How to draw this picture (the red letter H)

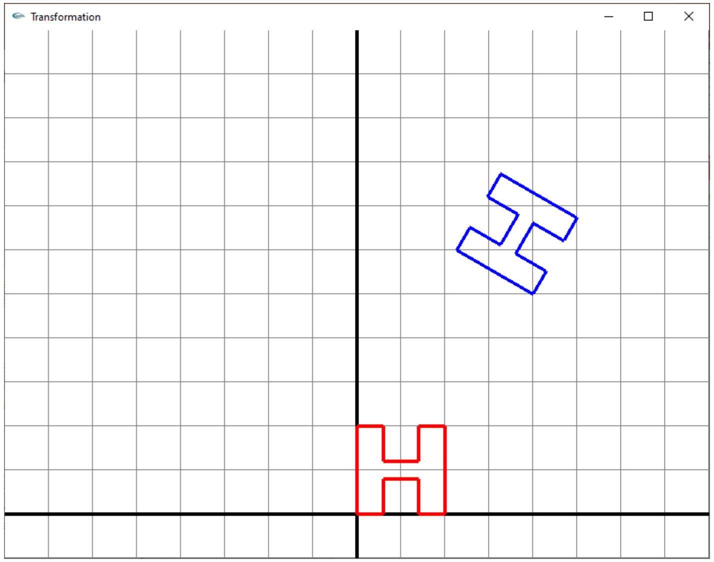
```
float   v0[12][2] = {   {0, 0}, {0, 2}, {0.6, 2},
                        {0.6, 1.2}, {1.4, 1.2}, {1.4, 2},
                        {2, 2}, {2, 0}, {1.4, 0},
                        {1.4, 0.8}, {0.6, 0.8}, {0.6, 0} };
```




```
void drawFigure0()
{
    glColor3f(1, 0, 0);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < 12; i++)
        glVertex2fv(v0[i]);
    glEnd();
}
```

Transformations in OpenGL

- How to draw the blue H



$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(60) & -\sin(60) & 0 \\ \sin(60) & \cos(60) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(60) & -\sin(60) & 4 \\ \sin(60) & \cos(60) & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformations in OpenGL

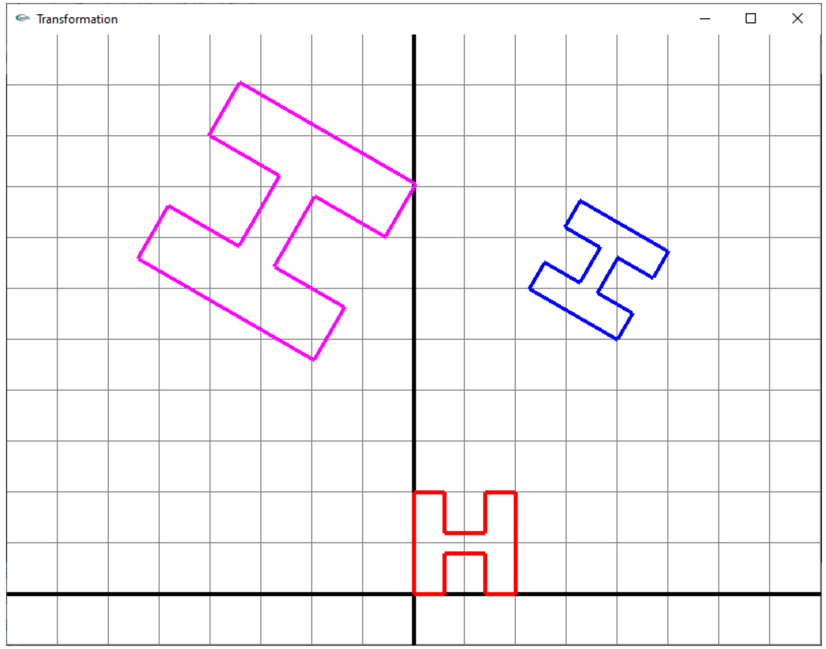
- How to draw the blue H

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos(60) & -\sin(60) & 0 \\ \sin(60) & \cos(60) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(60) & -\sin(60) & 4 \\ \sin(60) & \cos(60) & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

```
void drawFigure1()
{
    float v1[12][2];
    for (int i = 0; i < 12; i++)
    {
        v1[i][0] = v0[i][0] * cos(PI / 3) - v0[i][1] * sin(PI / 3) + 4;
        v1[i][1] = v0[i][0] * sin(PI / 3) + v0[i][1] * cos(PI / 3) + 5;
    }
    glColor3f(0, 0, 1);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < 12; i++)
        glVertex2fv(v1[i]);
    glEnd();
}
```

Transformations in OpenGL

- How to draw the purple H



$$\begin{pmatrix} \cos(60) & -\sin(60) & 0 \\ \sin(60) & \cos(60) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2\cos(60) & -2\sin(60) & 3\cos(60) - 4\sin(60) \\ 2\sin(60) & 2\cos(60) & 3\sin(60) + 4\cos(60) \\ 0 & 0 & 1 \end{pmatrix}$$

Transformations in OpenGL

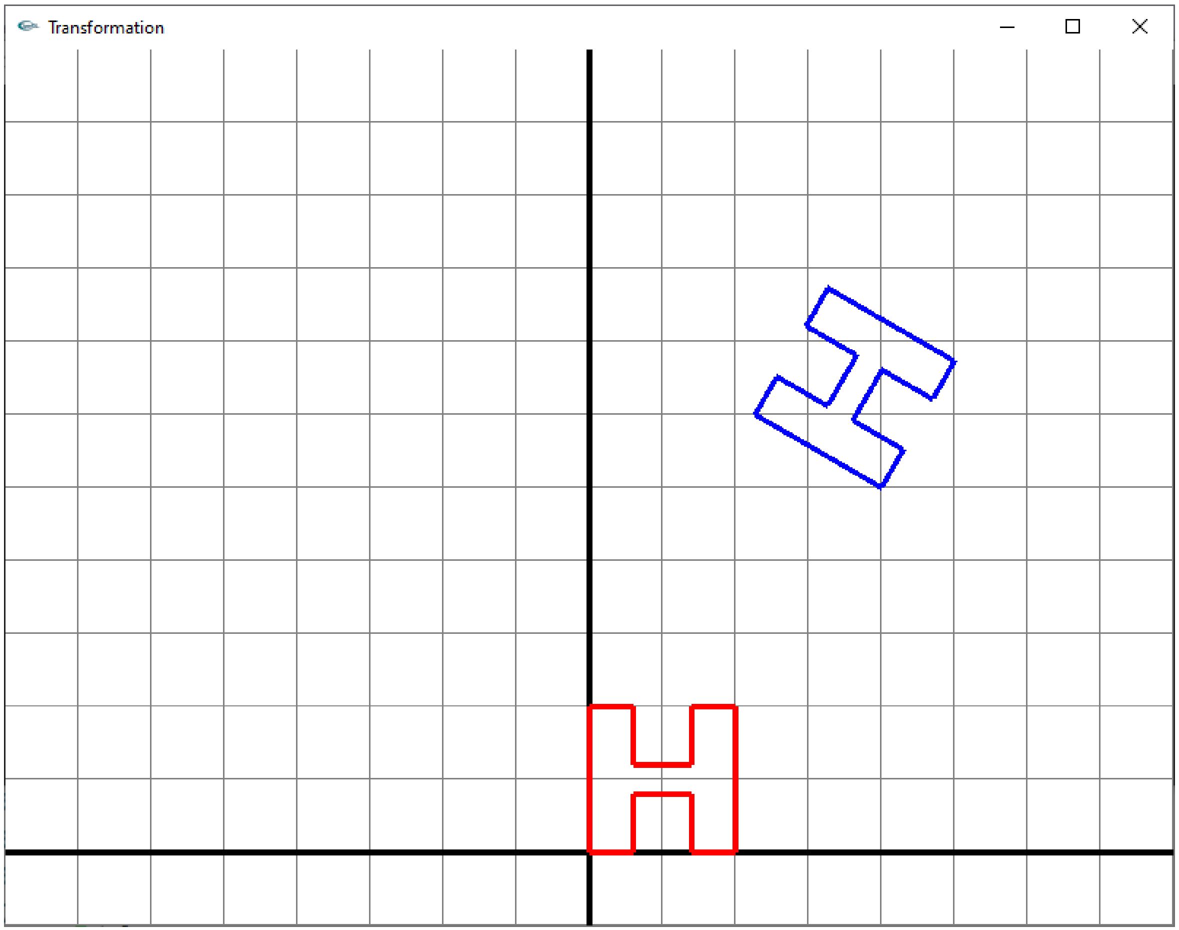
- How to draw the purple H

$$\begin{pmatrix} \cos(60) & -\sin(60) & 0 \\ \sin(60) & \cos(60) & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2\cos(60) & -2\sin(60) & 3\cos(60)-4\sin(60) \\ 2\sin(60) & 2\cos(60) & 3\sin(60)+4\cos(60) \\ 0 & 0 & 1 \end{pmatrix}$$

```
void drawFigure2(){
    float v2[12][2];
    for (int i = 0; i < 12; i++){
        v2[i][0] = v0[i][0] * 2*cos(PI / 3) - v0[i][1] * 2*sin(PI / 3) + 3*cos(PI/3)-4*sin(PI/3);
        v2[i][1] = v0[i][0] * 2*sin(PI / 3) + v0[i][1] * 2*cos(PI / 3) + 3*sin(PI/3)+4*cos(PI/3);
    }
    glColor3f(1, 0, 1);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < 12; i++)
        glVertex2fv(v2[i]);
    glEnd();
}
```

Transformations in OpenGL

- How to draw the blue H



Transformations in OpenGL

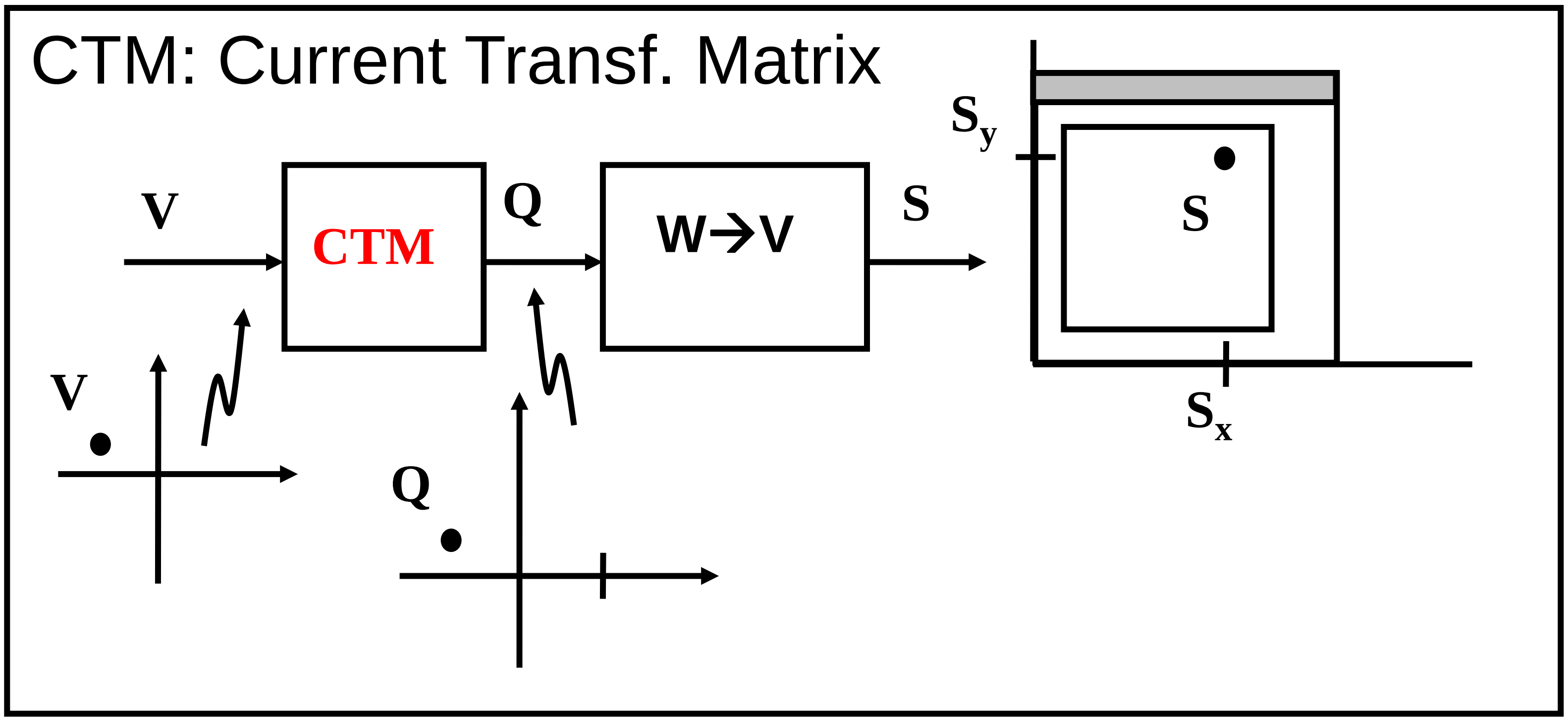
```
void mydisplay() {
    glClear(GL_COLOR_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    drawGrid();

    glColor3f(1, 0, 0);//The red H
    drawFigure0();

    glColor3f(0, 0, 1); //The blue H
    glTranslatef(4, 5, 0);
    glRotatef(60, 0, 0, 1);
    drawFigure0();

    glFlush();
}
```

Transformations in OpenGL

glMatrixMode(GL_MODELVIEW);

glLoadIdentity(); CTM = I

drawFigure0();



Transformations in OpenGL

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

CTM = I

```
glColor3f(1, 0, 0);
drawFigure0();
```

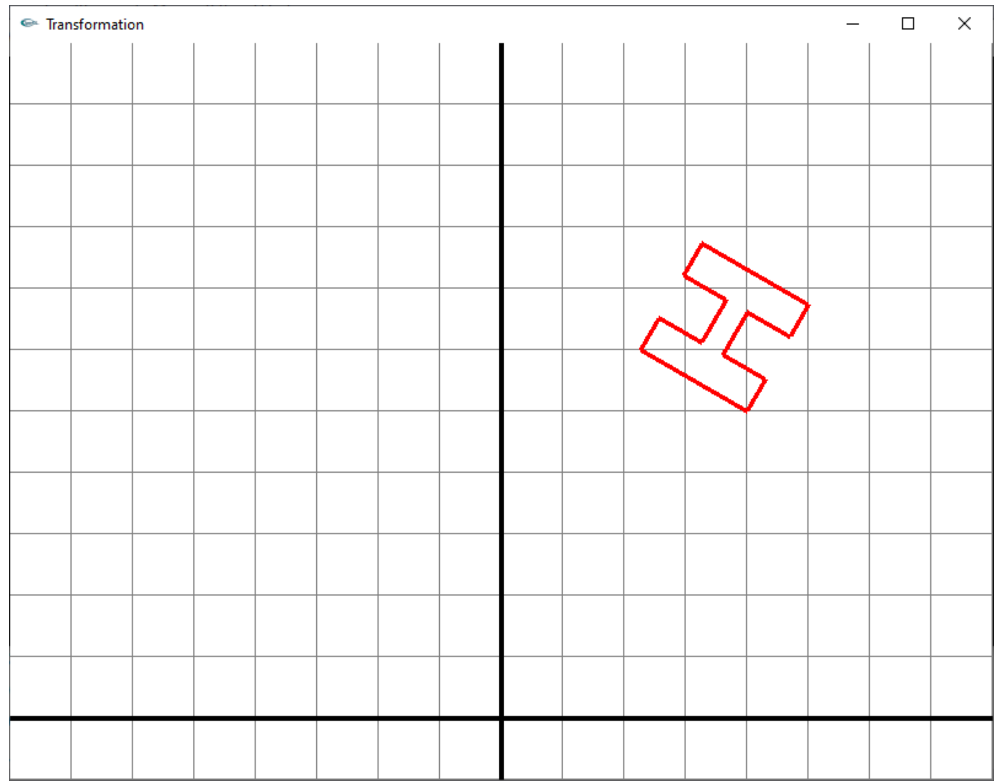
$$CTM = CTM * \begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glColor3f(0, 0, 1);
glTranslatef(4, 5, 0);
glRotatef(60, 0, 0, 1);
drawFigure0();
```

$$CTM = CTM * \begin{pmatrix} \cos(60) & -\sin(60) & 0 & 0 \\ \sin(60) & \cos(60) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(60) & -\sin(60) & 0 & 4 \\ \sin(60) & \cos(60) & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformations in OpenGL

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

drawGrid();

glColor3f(0, 0, 1);
glTranslatef(4, 5, 0);
glRotatef(60, 0, 0, 1);
drawFigure0();

glColor3f(1, 0, 0);
drawFigure0();
```



Transformations in OpenGL

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

CTM = I

```
glColor3f(0, 0, 1);
glTranslatef(4, 5, 0);
glRotatef(60, 0, 0, 1);
drawFigure0();
```

$$CTM = \begin{pmatrix} \cos(60) & -\sin(60) & 0 & 4 \\ \sin(60) & \cos(60) & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glColor3f(1, 0, 0);
drawFigure0();
```

Transformations in OpenGL

glMatrixMode(GL_MODELVIEW);

glLoadIdentity(); CTM = I

glPushMatrix();

glColor3f(0, 0, 1);//The blue H

glTranslatef(4, 5, 0); CTM = $\begin{pmatrix} \cos(60) & -\sin(60) & 0 & 4 \\ \sin(60) & \cos(60) & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

glRotatef(60, 0, 0, 1);

drawFigure0();

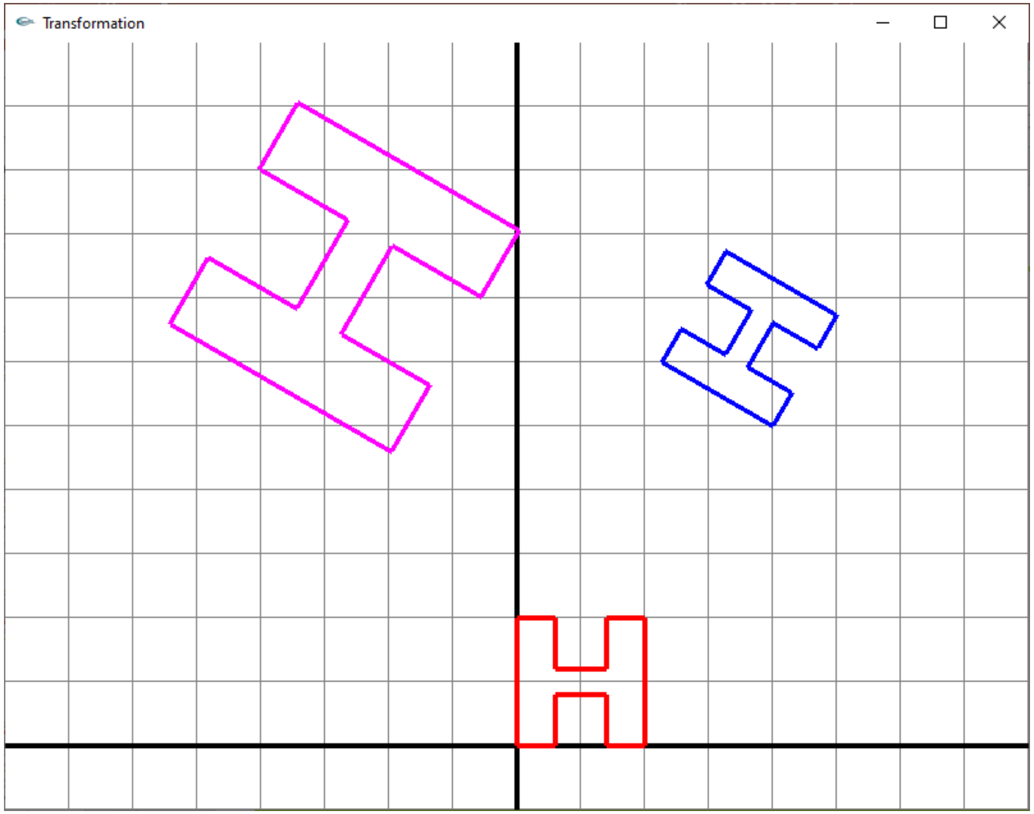
glPopMatrix();

glColor3f(1, 0, 0);//The red H CTM = I

drawFigure0();

Transformations in OpenGL

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
drawGrid();

glPushMatrix();
glColor3f(0, 0, 1); //The blue H
glTranslatef(4, 5, 0);
glRotatef(60, 0, 0, 1);
drawFigure0();
glPopMatrix();

glPushMatrix();
glColor3f(1, 0, 1);//The purple H
glRotatef(60, 0, 0, 1);
glTranslatef(3, 4, 0);
glScalef(2, 2, 1);
drawFigure0();
glPopMatrix();

glColor3f(1, 0, 0);//The red H
drawFigure0();
```



Transformations in OpenGL

- Draw the blue H

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos(60) & -\sin(60) & 0 \\ \sin(60) & \cos(60) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(60) & -\sin(60) & 4 \\ \sin(60) & \cos(60) & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

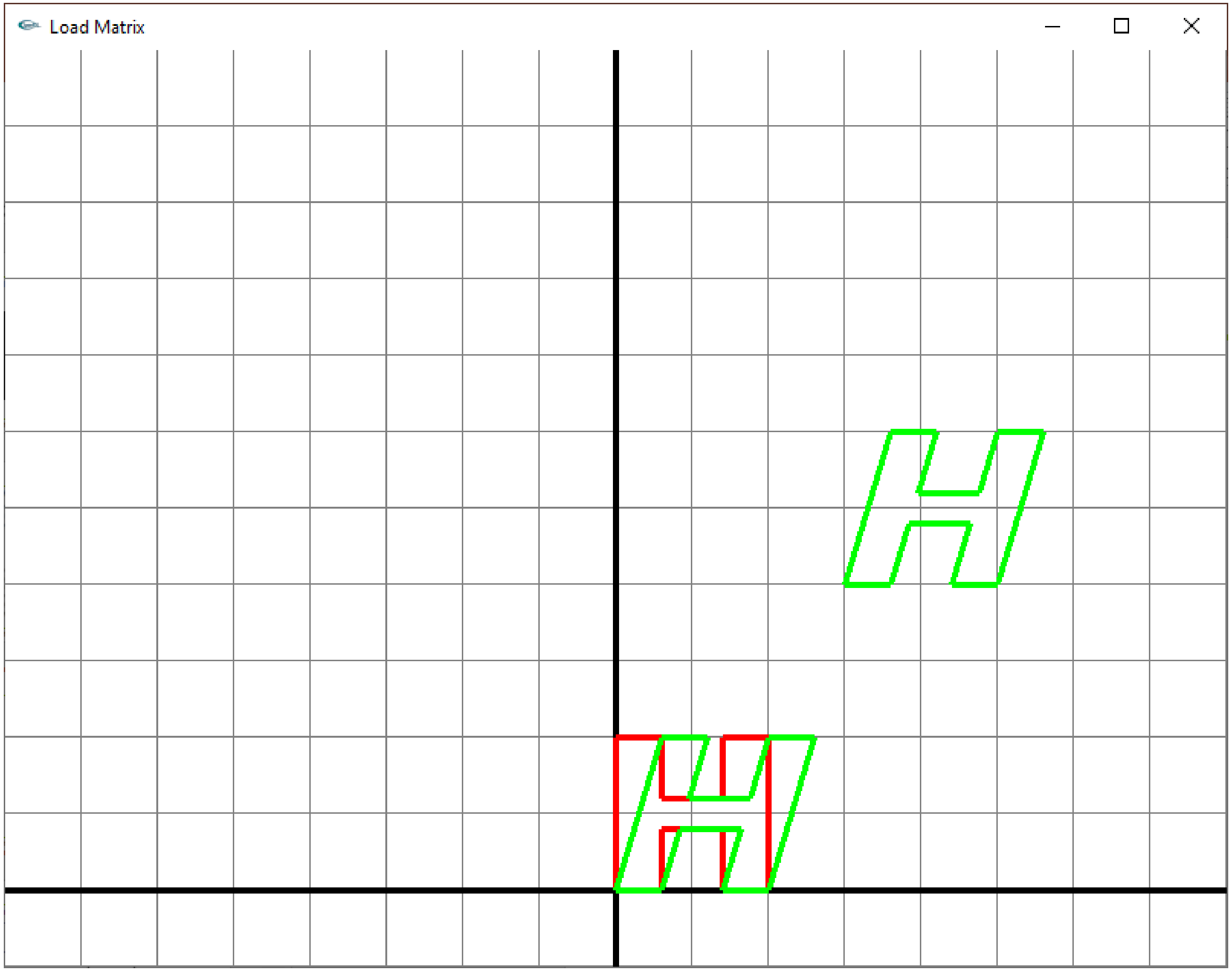
$$\begin{pmatrix} \cos(60) & -\sin(60) & 0 & 4 \\ \sin(60) & \cos(60) & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
float m[16] = { cos(PI / 3), sin(PI / 3), 0, 0,
                -sin(PI / 3), cos(PI / 3), 0, 0,
                0, 0, 1, 0,
                4, 5, 0, 1 };
```

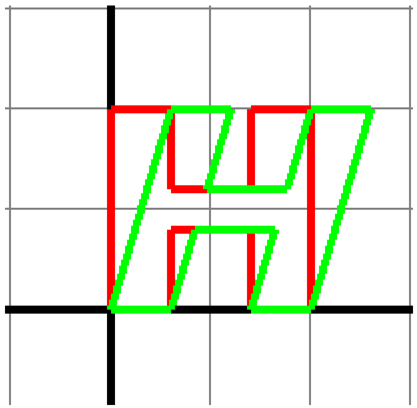
```
glPushMatrix();
glColor3f(0, 0, 1);
glTranslatef(4, 5, 0);
glRotatef(60, 0, 0, 1);
drawFigure0();
glPopMatrix();
```

```
glPushMatrix();
glLoadMatrixf(m);
glColor3f(0, 0, 1);
drawFigure0();
glPopMatrix();
```

Transformations in OpenGL



Transformations in OpenGL



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

- Qx = Px + hPy (Qx, Qy) = (0.6, 2.0); (Px, Py) = (0, 2.0)

  0.6 = 0 + 2h → h = 0.3

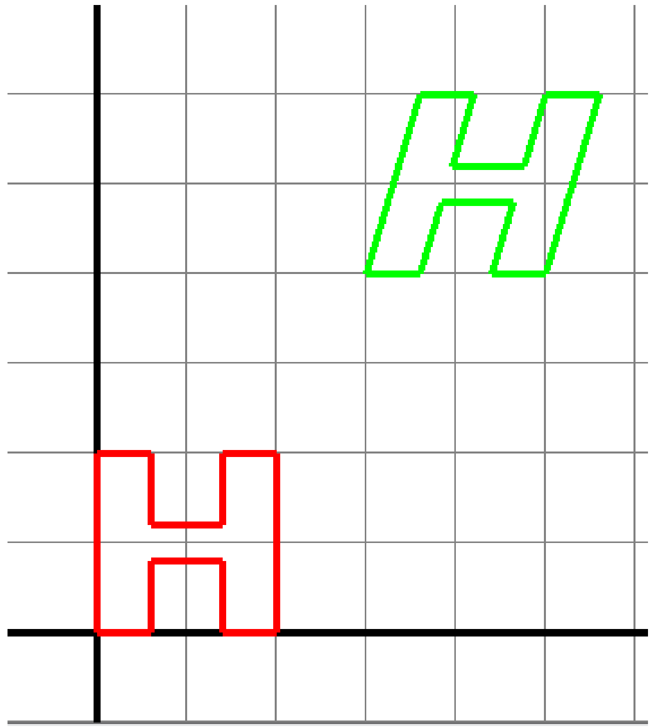
$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0.3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0.3 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformations in OpenGL

$$\begin{pmatrix} 1 & 0.3 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

```
float ShearNTranslate[16] = { 1, 0, 0, 0,
                              0.3, 1, 0, 0,
                              0, 0, 1, 0,
                              3, 4, 0, 1 };
```

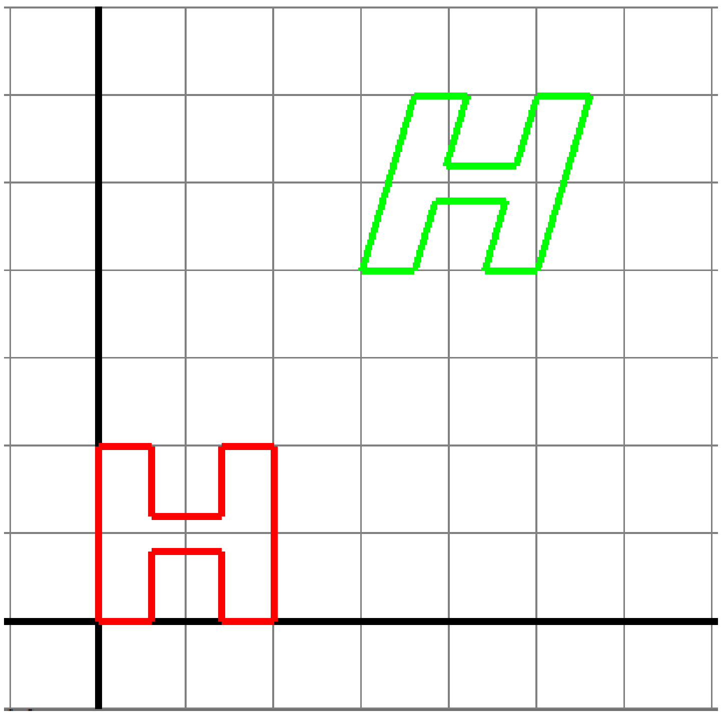
```
glPushMatrix();
glLoadMatrixf(ShearNTranslate);
glColor3f(0, 1, 0);
drawFigure0();
glPopMatrix();
```



Transformations in OpenGL

- glMultMatrixf

```
float Shear[16] = { 1, 0, 0, 0,
                    0.3, 1, 0, 0,
                    0, 0, 1, 0,
                    0, 0, 0, 1 };
```



```
glPushMatrix();
glTranslatef(3, 4, 0);
glMultMatrixf(Shear);
glColor3f(0, 1, 0);
drawFigure0();
glPopMatrix();
```

Transformations in OpenGL

- Load an identity matrix:
  - **glLoadIdentity()**
- Rotation, Translation, Scaling
  - **glRotatef(theta, vx, vy, vz)**
    - *theta in degrees, (vx, vy, vz) define axis of rotation*
  - **glTranslatef(dx, dy, dz)**
  - **glScalef(sx, sy, sz)**

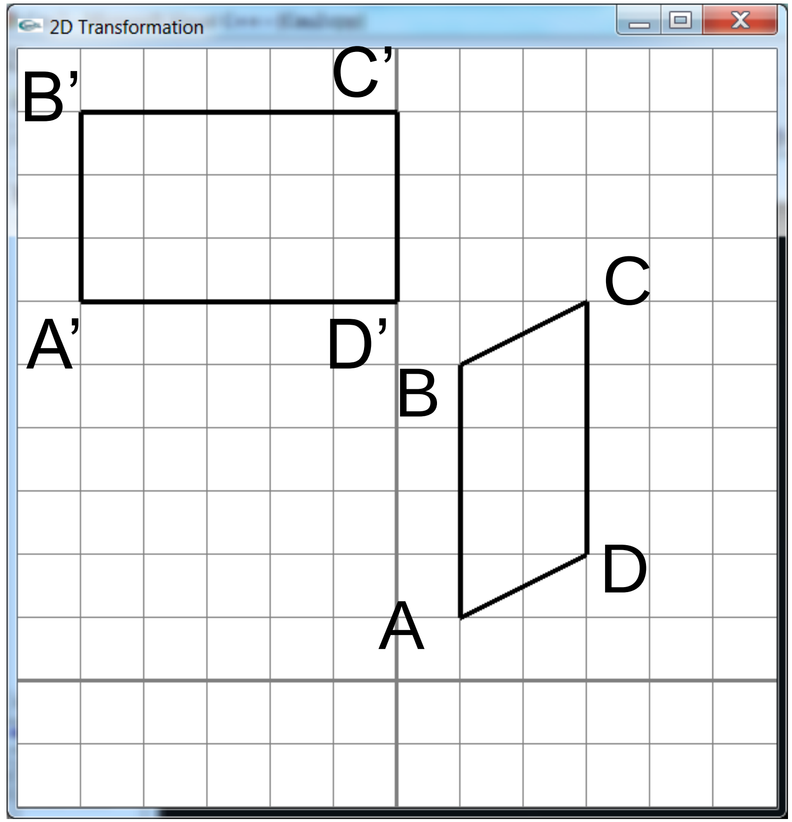
Transformations in OpenGL

- Can load and multiply by matrices defined in the application program
  - **glLoadMatrixf(m)**
  - **glMultMatrixf(m)**
- The matrix **m is a one dimension array of 16** elements which are the components of the desired 4 x 4 matrix stored by columns
- In **glMultMatrixf, m multiplies the existing** matrix on the right

Transformations in OpenGL

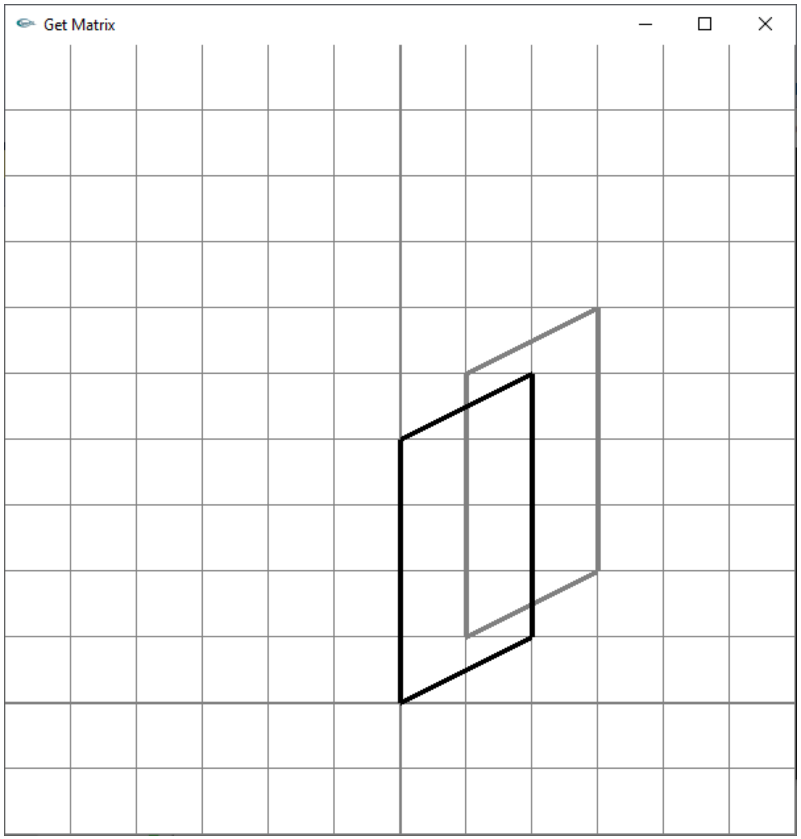
- Reading Back Matrices
  - Can also access matrices (and other parts of the state) by *query functions*
    - **glGetIntegerv**
    - **glGetFloatv**
    - **glGetBooleanv**
    - **glGetDoublev**
    - **glIsEnabled**
  - For matrices, we use as
    - **float m[16];**
    - **glGetFloatv(GL_MODELVIEW_MATRIX, m);**

Transformations in OpenGL



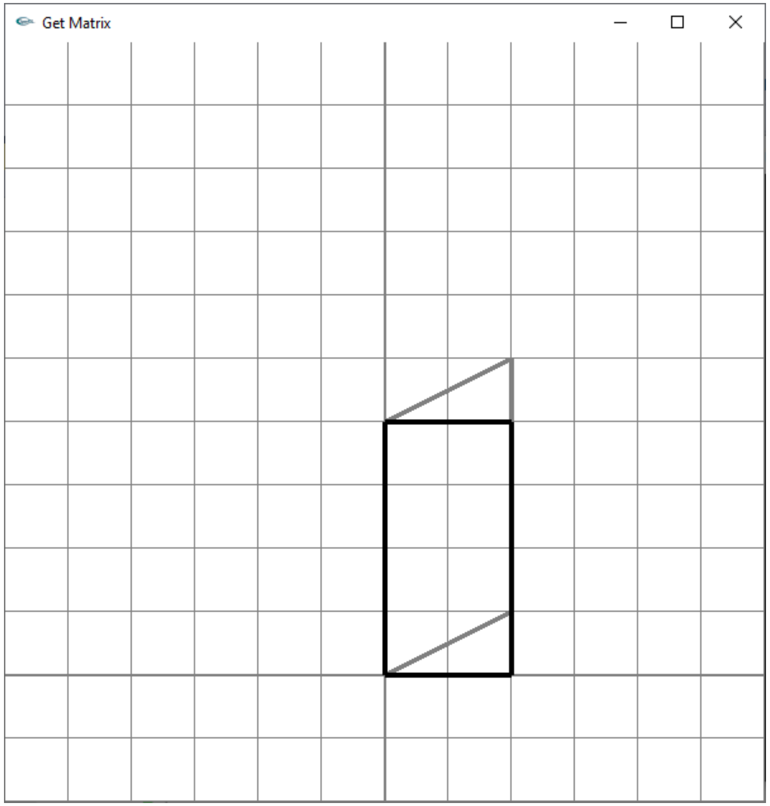
Tìm ma trận biến đổi hình bình hành thành hình chữ nhật

Transformations in OpenGL



$$M1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL

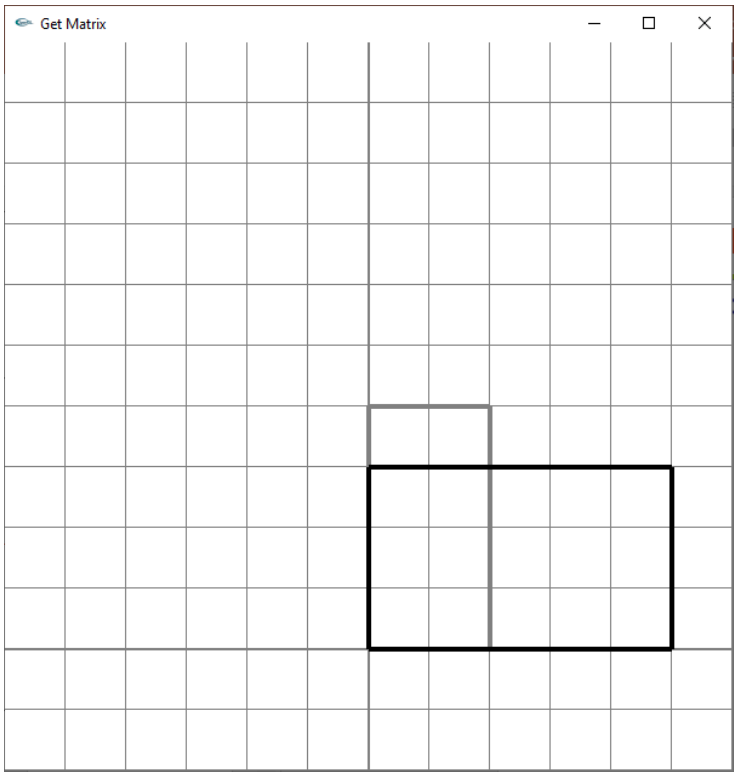


$$\begin{pmatrix} 1 & 0 & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Qy = gPx + Py
- (2, 1) → (2, 0)
- 0 = 2g + 1 → g = -0.5

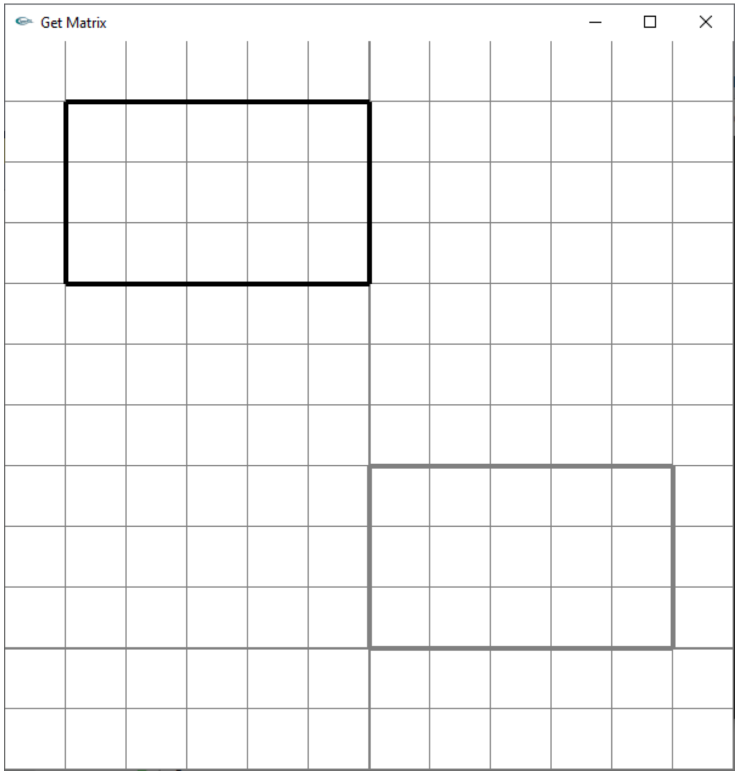
$$M2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL



$$M3 = \begin{bmatrix} 5/2 & 0 & 0 & 0 \\ 0 & 3/4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL



$$M4 = \begin{bmatrix} 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL

- Tịnh tiến (M1)
- Trượt theo trục y (M2)
- Tỷ lệ (M3)
- Tịnh tiến (M4)

Transformations in OpenGL

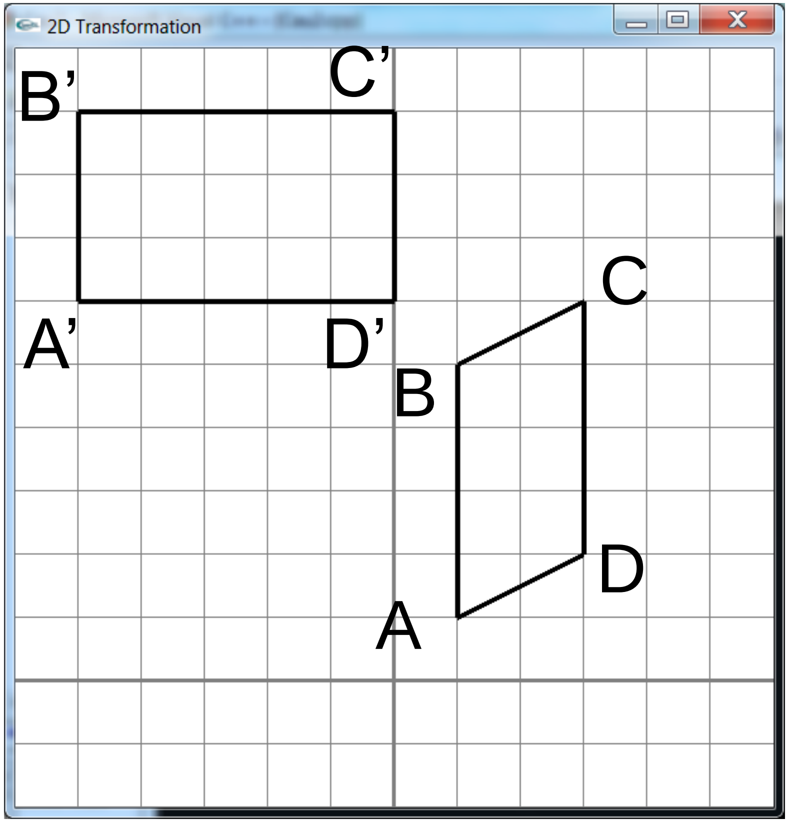
$$M1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M3 = \begin{bmatrix} 5/2 & 0 & 0 & 0 \\ 0 & 3/4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad M4 = \begin{bmatrix} 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

M = M4*M3*M2*M1

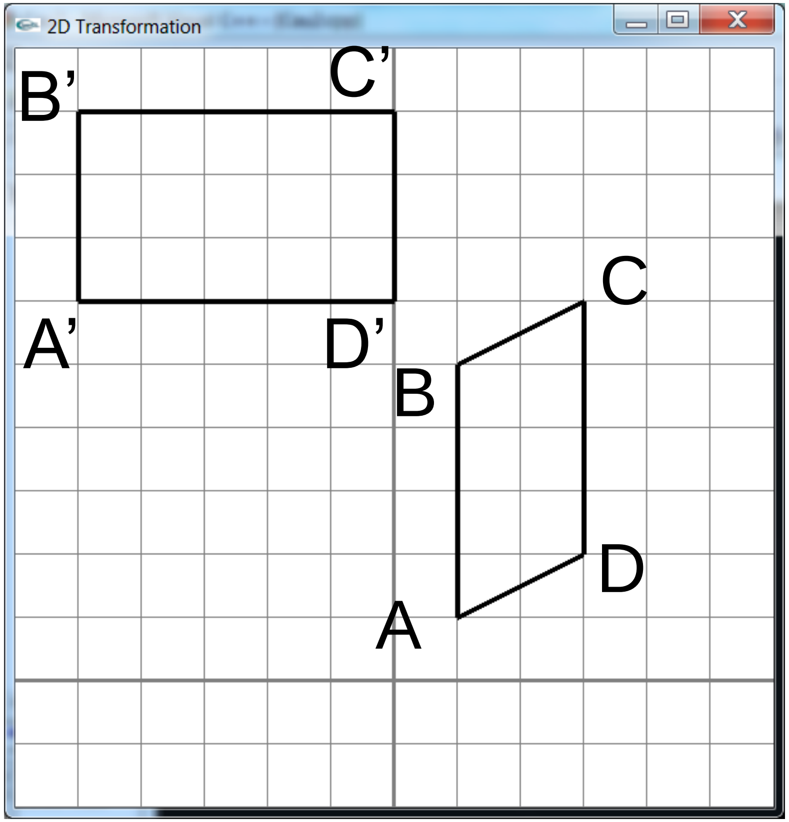
Transformations in OpenGL

M = M4*M3*M2*M1



$$M = \begin{bmatrix} 5/2 & 0 & 0 & -15/2 \\ -3/8 & 3/4 & 0 & 45/8 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL



$$Q_x = m_{11}P_x + m_{12}P_y + m_{13}$$
$$Q_y = m_{21}P_x + m_{22}P_y + m_{23}$$

-5 = m11 + m12 + m13

6 = m21 + m22 + m23

0 = 3m11+ 6m12 + m13

9 = 3m21 + 6m22 + m23

0 = 3m11 + 2m12 + m13

6 = 3m21 + 2m22 + m23

m11 = 5/2, m12 = 0, m13 = -15/2

m21 = -3/8, m22= 3/4,m23 = 45/8

Transformations in OpenGL

```
float Shear[16] = { 1, -0.5, 0, 0,
                    0, 1, 0, 0,
                    0, 0, 1, 0,
                    0, 0, 0, 1 };
```

```
float modelviewMatrix[16];
glColor3f(0, 0, 0);
glPushMatrix();
glTranslatef(-5, 6, 0);
glScalef(5 / 2.0, 3 / 4.0, 1);
glMultMatrixf(Shear);
glTranslatef(-1, -1, 0);
glGetFloatv(GL_MODELVIEW_MATRIX, modelviewMatrix);
printMatrix(modelviewMatrix);
glPopMatrix();
```

```
void printMatrix(float m[16]){
    for (int i = 0; i < 4; i++){
        printf("%8.4f  %8.4f  %8.4f  %8.4f", m[i], m[i + 4], m[i + 8], m[i + 12]);
        printf("\n");
    }
}
```