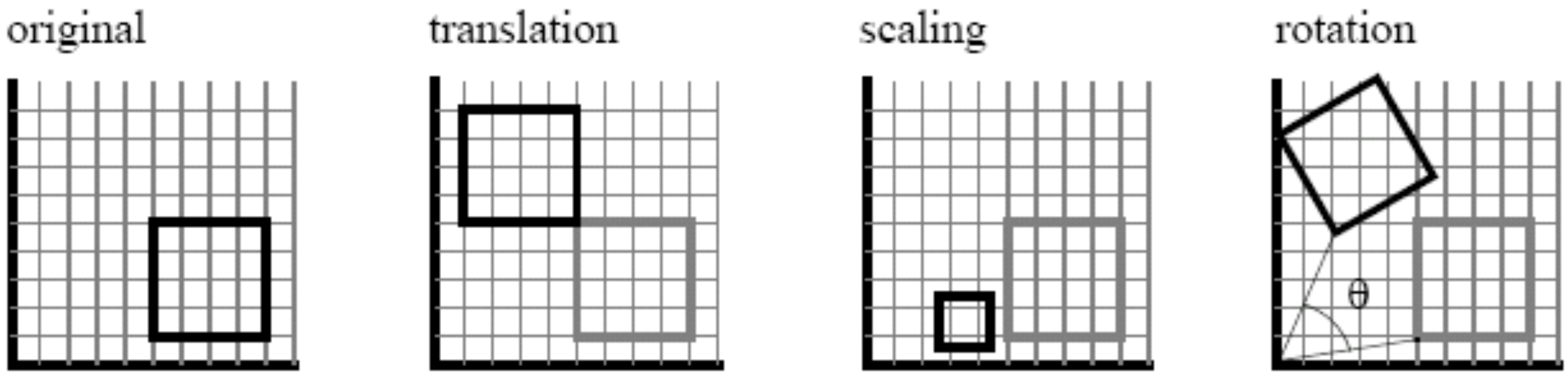


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

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

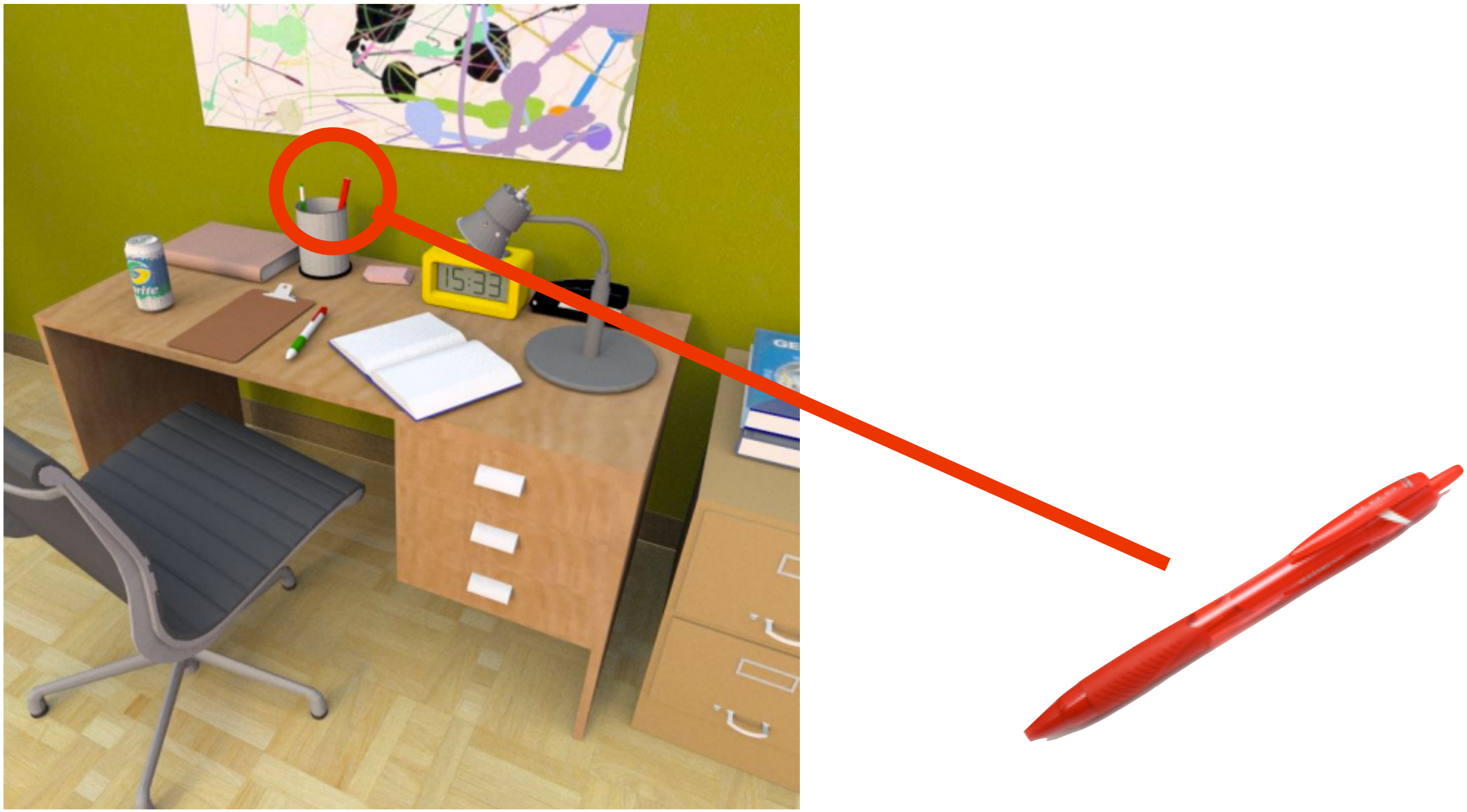
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformation

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

http://www.pling.org.uk/cs/cgv.html

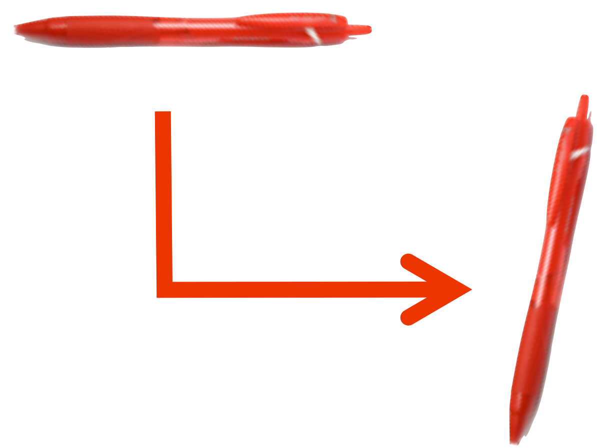
Placement of Objects



Fisher et al. (2012)

Placement of Objects

Oriented

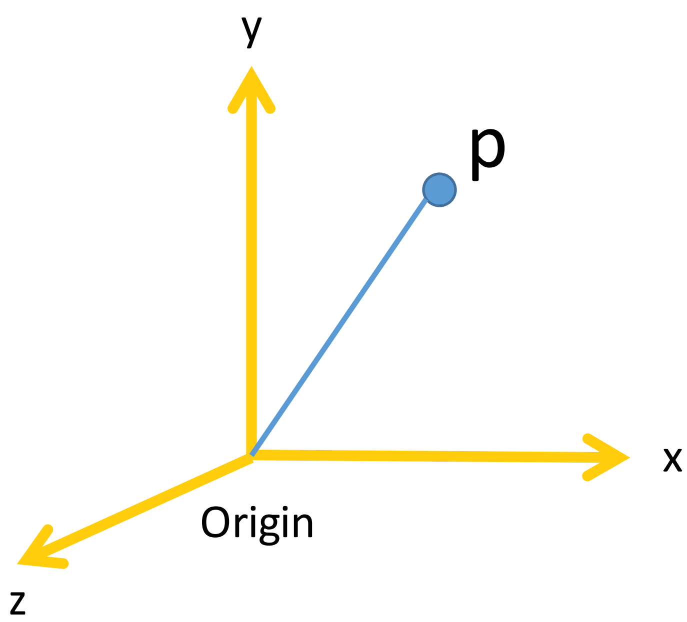


Fisher et al. (2012)

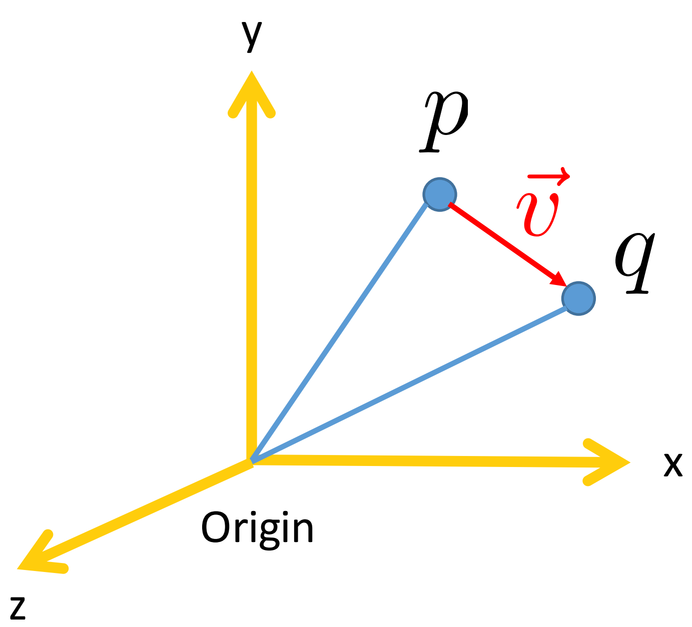
Placements of Objects

Fisher et al. (2012)

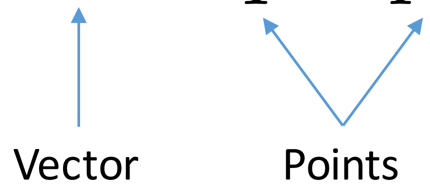
Points and Vectors



Points and Vectors



$$\vec{v} = q - p$$



Transformations

- What? Just functions acting on points:

$$(x', y', z') = T(x, y, z)$$

$$p' = T(p)$$

- Why?
  - Viewing:
    - Convert between coordinates systems
    - Virtual camera, e.g. perspective projections
  - Modeling:
    - Create objects in a convenient orientation
    - Use multiple instances of a given shape
    - Kinematics – characters/robots

Linear Transformation

Linear Transformation

$$(x', y', z') = T(x, y, z)$$

$$x' = Ax + By + Cz$$
$$y' = Dx + Ey + Fz$$
$$z' = Gx + Hy + Iz$$

Linear Transformation

$$(x', y', z') = T(x, y, z)$$

$$T = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Linear Transformation

$$(x', y', z') = T(x, y, z)$$

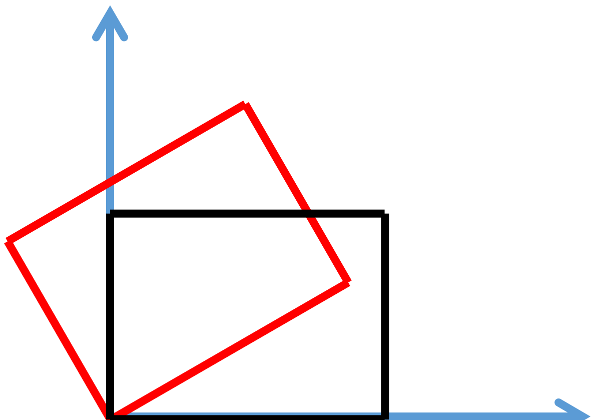
$$T = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Lines Map to Lines

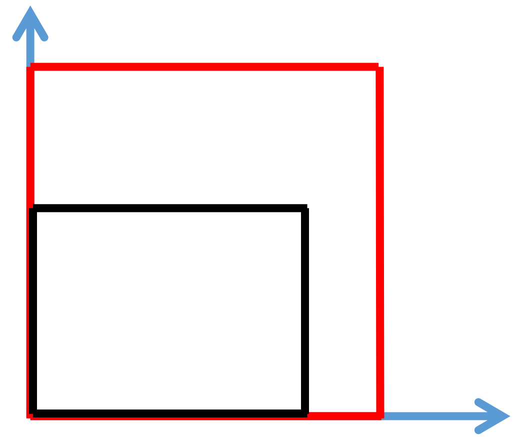
Common Transformation

Rotate



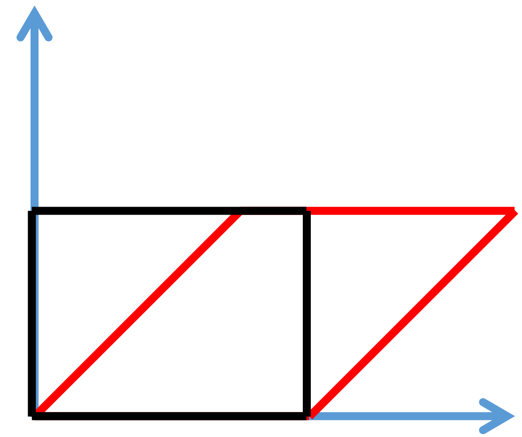
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & cos\theta \end{bmatrix}$$

Scale



$$\begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}$$

Shear



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

Archive
What If?
Blag
Store
About

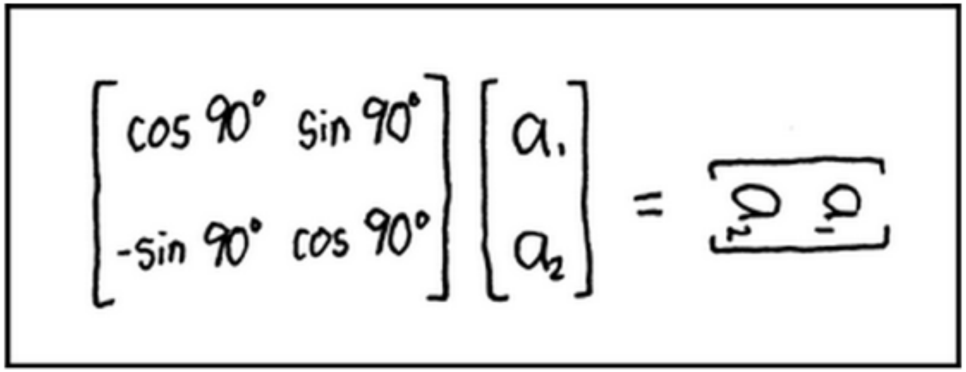
xkcd

A webcomic of romance, sarcasm, math, and language.

News:
I'm publishing a *What If* book!

Matrix Transform

|< < Prev Random Next > >|

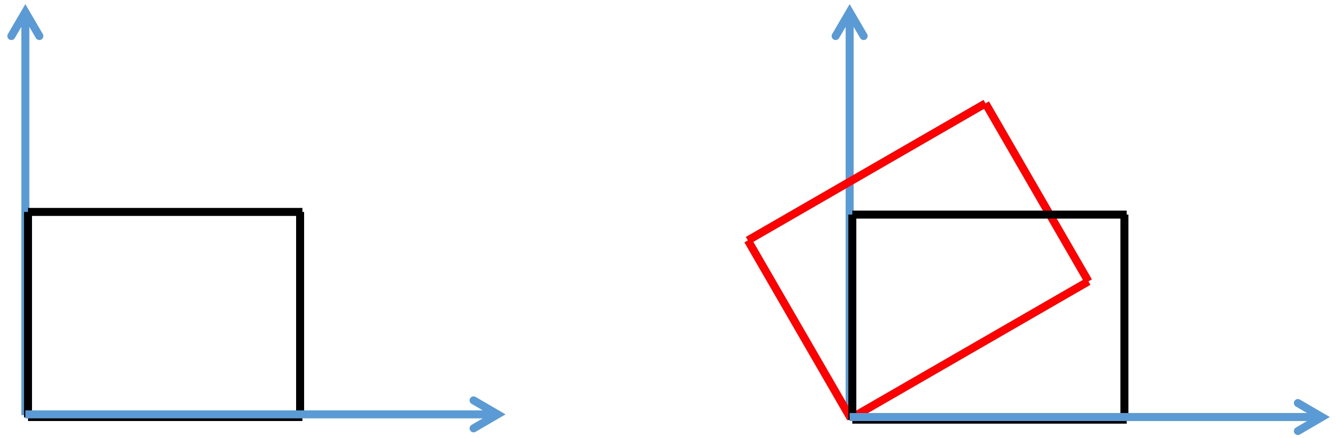


|< < Prev Random Next > >|

Permanent link to this comic: http://xkcd.com/184/
Image URL (for hotlinking/embedding): http://imgs.xkcd.com/comics/matrix_transform.png

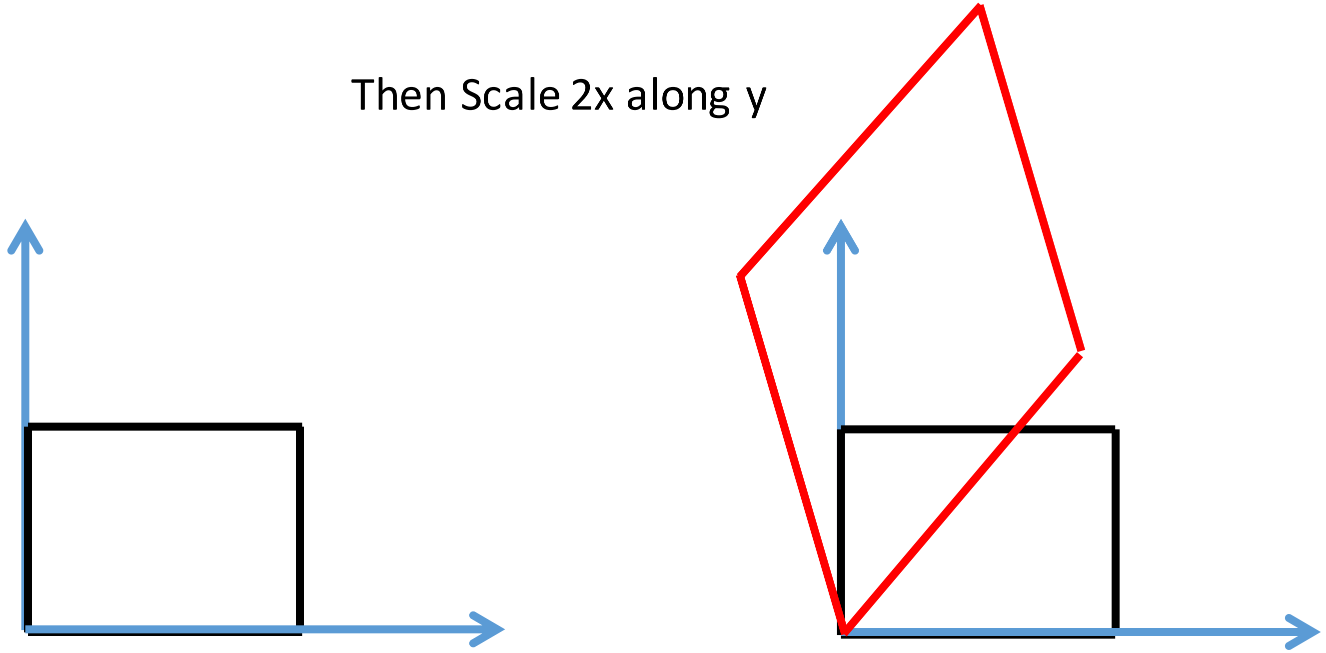
Composing Transformations

First Rotate 45°



Composing Transformations

Then Scale 2x along y



Composing Transformations

$$(x', y', z') = T_2\left(T_1(x, y, z)\right)$$
$$= T_2 T_1(x, y, z)$$

Scale Rotation

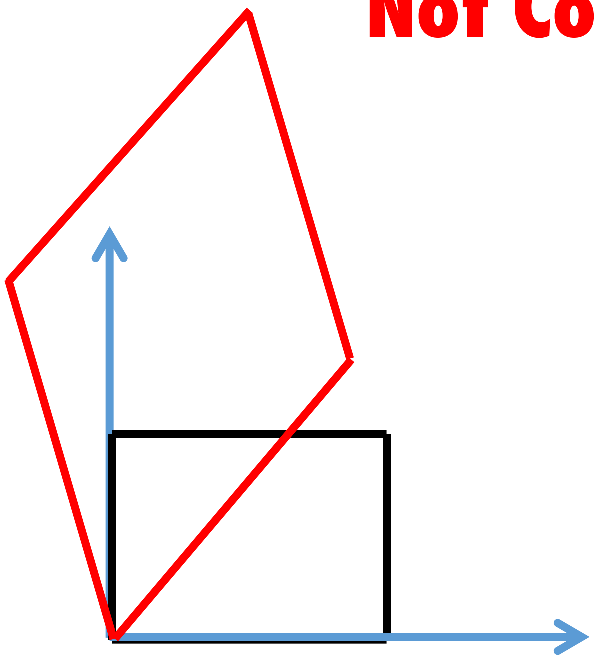
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Order of transformations

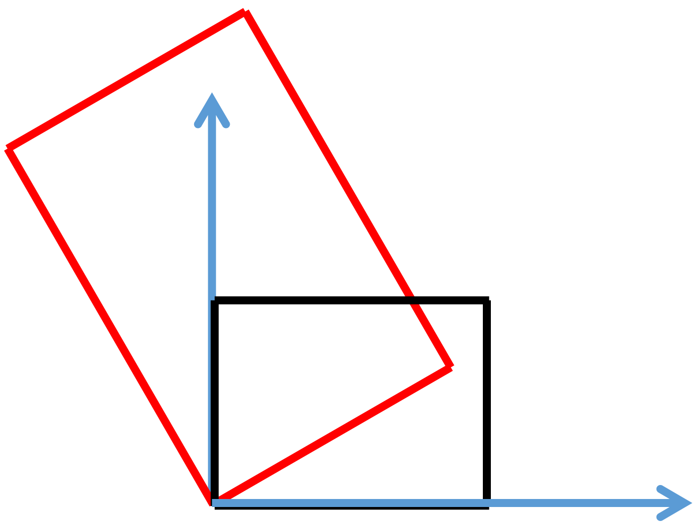
Composing Transformation

$$T_1 T_2 \neq T_2 T_1$$

Not Commutative



$\neq$



Rotate -> Scale

Scale -> Rotate

Translation

$$p' = Mp + b$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv c \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

For any non-zero c

Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv c \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

$$c = 1/w$$

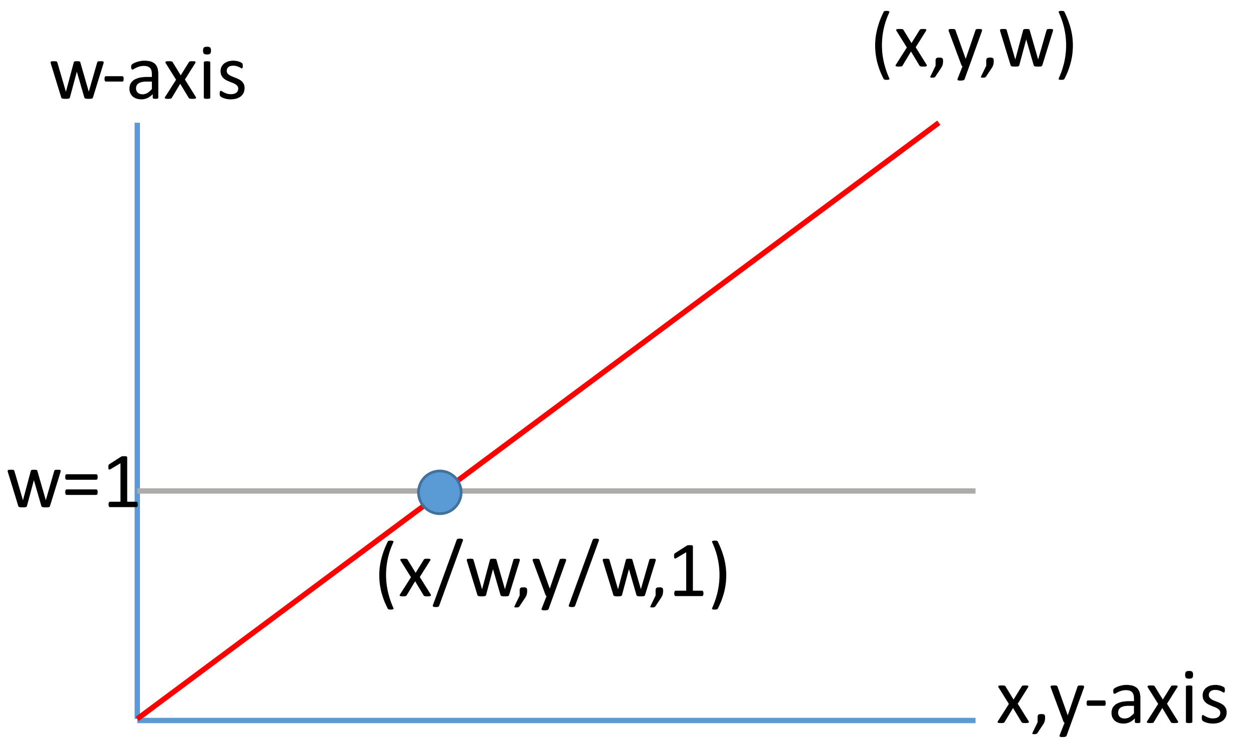
Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$$

4D Homogenous Coordinate

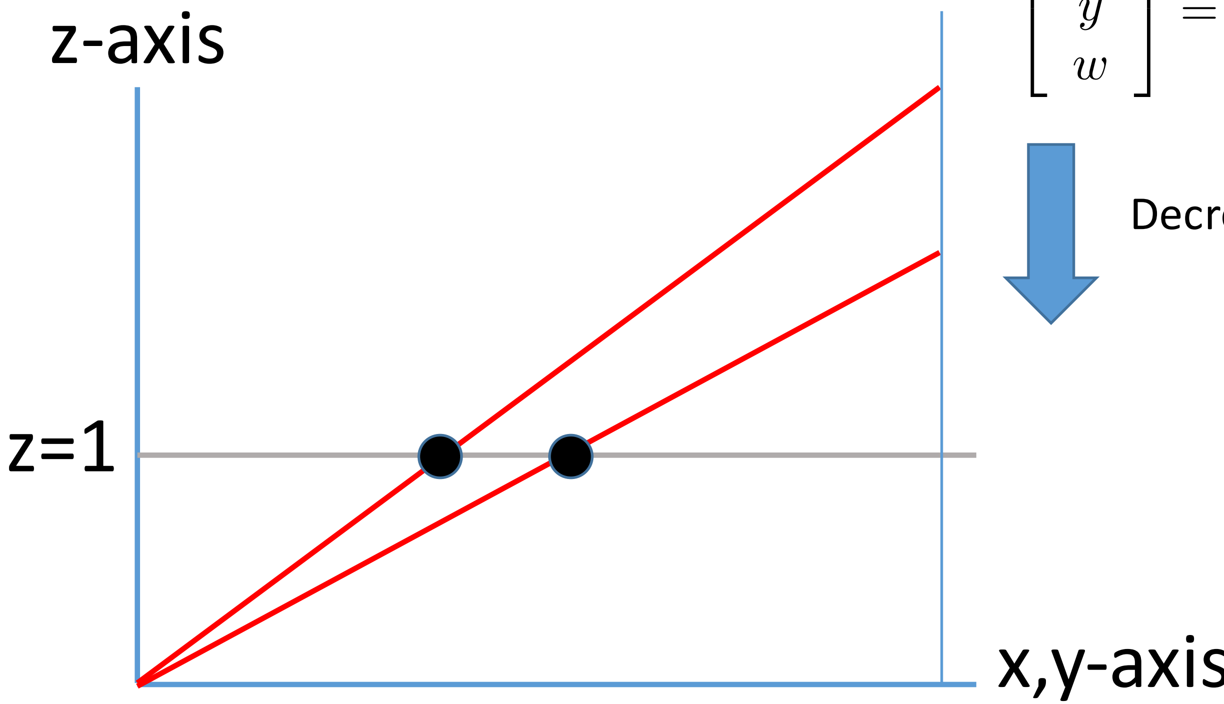
3D Cartesian Coordinate

Homogenous Coordinates



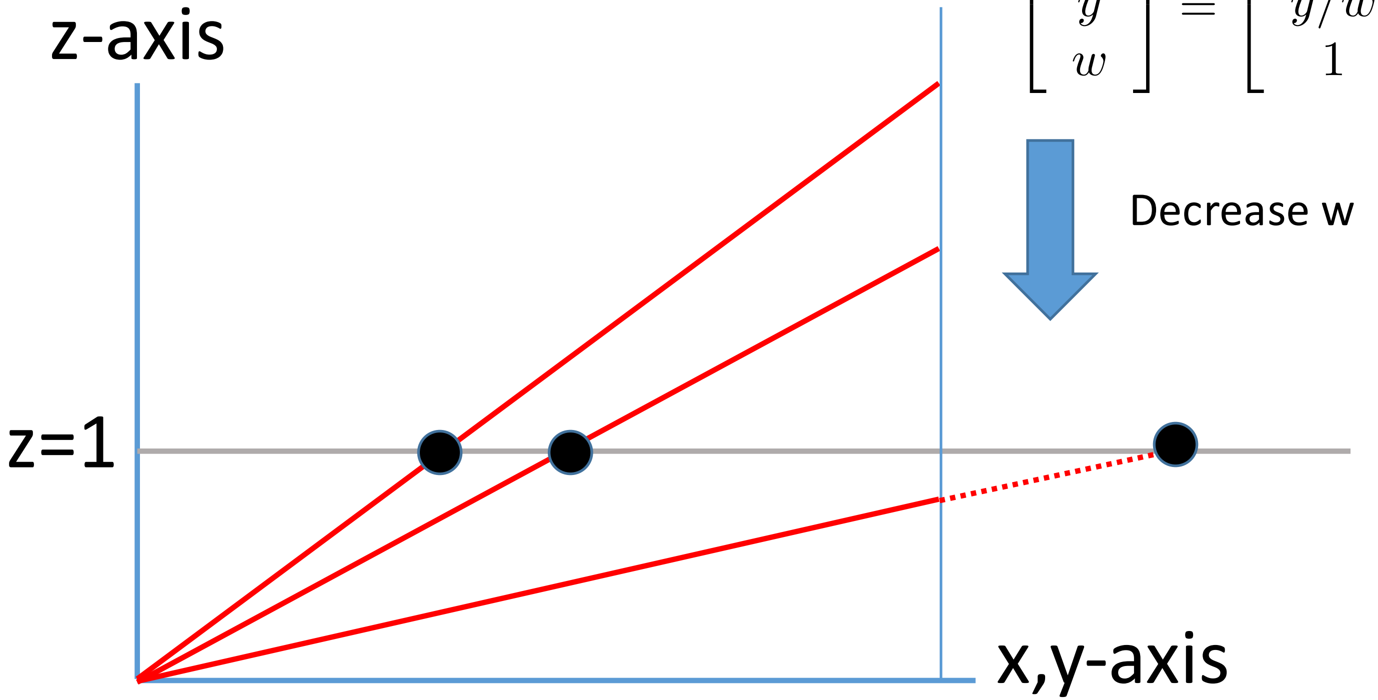
Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \equiv \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$



Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \equiv \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$



Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} ?$$

Homogenous Coordinates

$$\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} ?$$

Represents a Vector!

(Homogenous Coordinates express both Vectors and Points)

Back to Translation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & bx \\ D & E & F & by \\ G & H & I & bz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogenous Coordinate

Homogenous Coordinate

Some Exercise

Convert 3D Cartesian Coordinate to Homogenous Coordinate

(3,4,2) :

Some Exercise

Convert 3D Cartesian Coordinate to Homogenous Coordinate

(3,4,2) : (3,4,2,1)

Some Exercise

Convert 3D Cartesian Coordinate to Homogenous Coordinate

(3,4,2) : (3,4,2,1)

Convert Homogenous Coordinate to Cartesian Coordinate

(3,4,2,2) :

Some Exercise

Convert 3D Cartesian Coordinate to Homogenous Coordinate

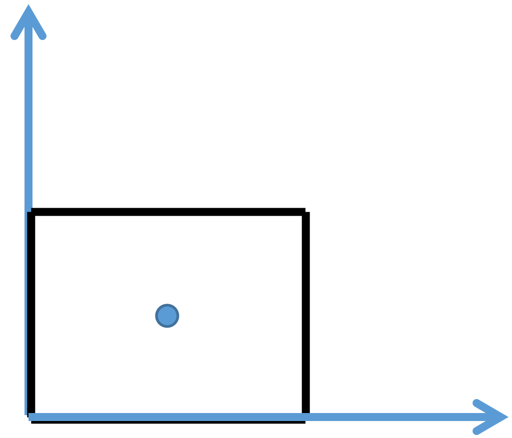
(3,4,2) : (3,4,2,1)

Convert Homogenous Coordinate to Cartesian Coordinate

(3,4,2,2) : (1.5,2,1)

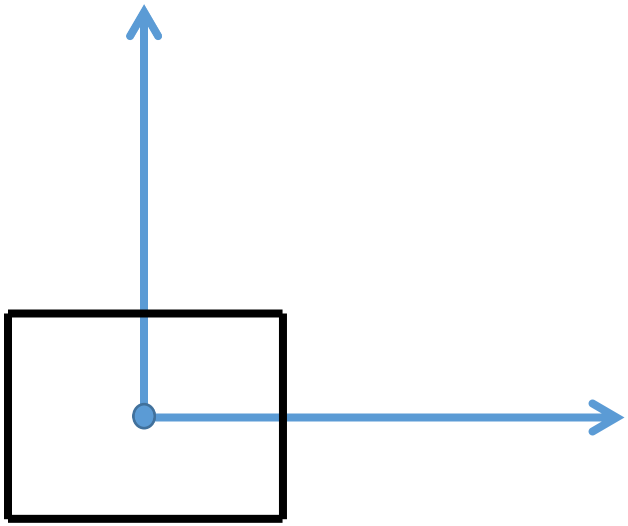
Some Exercise

Rotate about the center of the box



Some Exercise

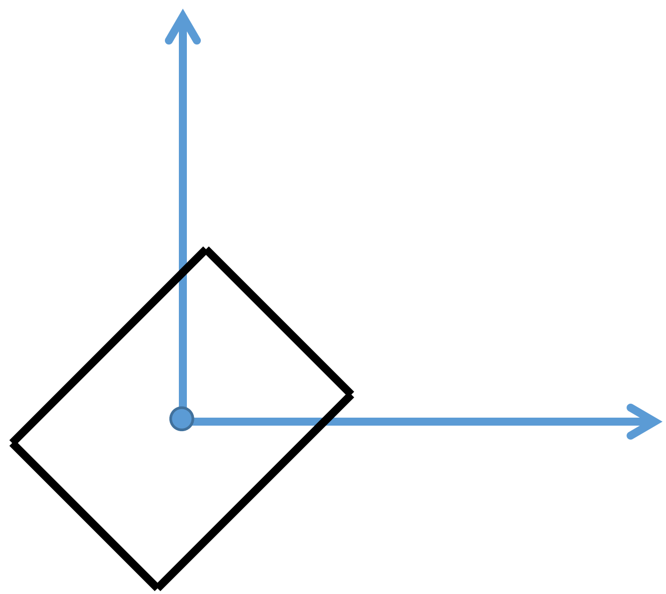
Translate to center



$$\begin{bmatrix} 1 & 0 & -bx \\ 0 & 1 & -by \\ 0 & 0 & 1 \end{bmatrix}$$

Some Exercise

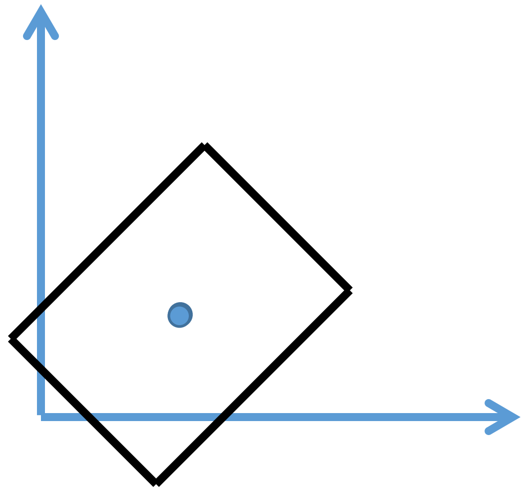
Rotate



$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -bx \\ 0 & 1 & -by \\ 0 & 0 & 1 \end{bmatrix}$$

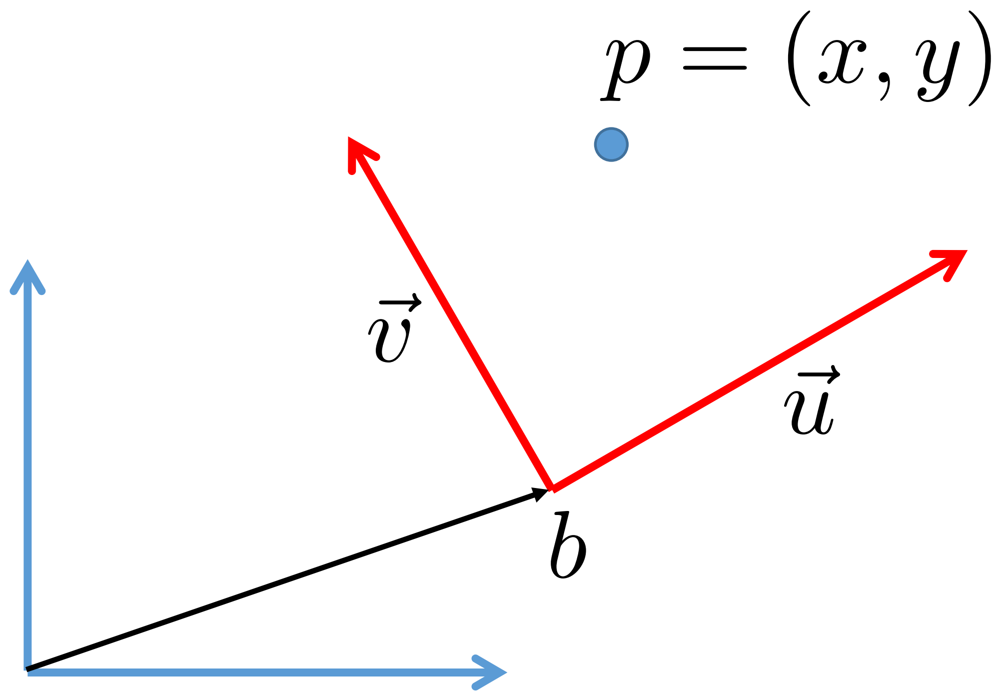
Some Exercise

Translate back

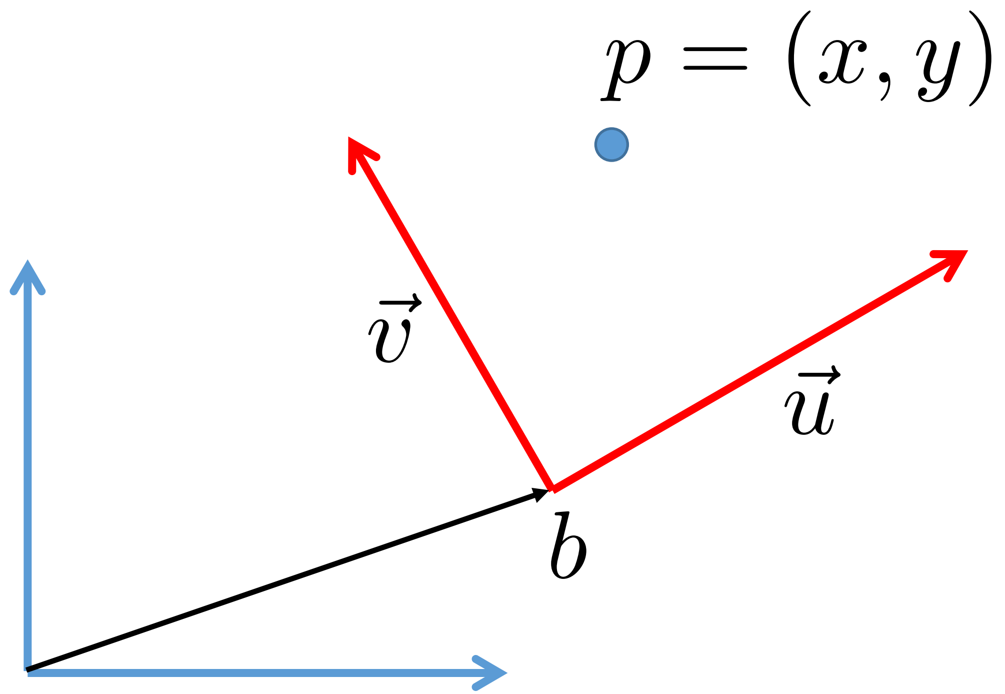


$$\begin{bmatrix} 1 & 0 & bx \\ 0 & 1 & by \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -bx \\ 0 & 1 & -by \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate System

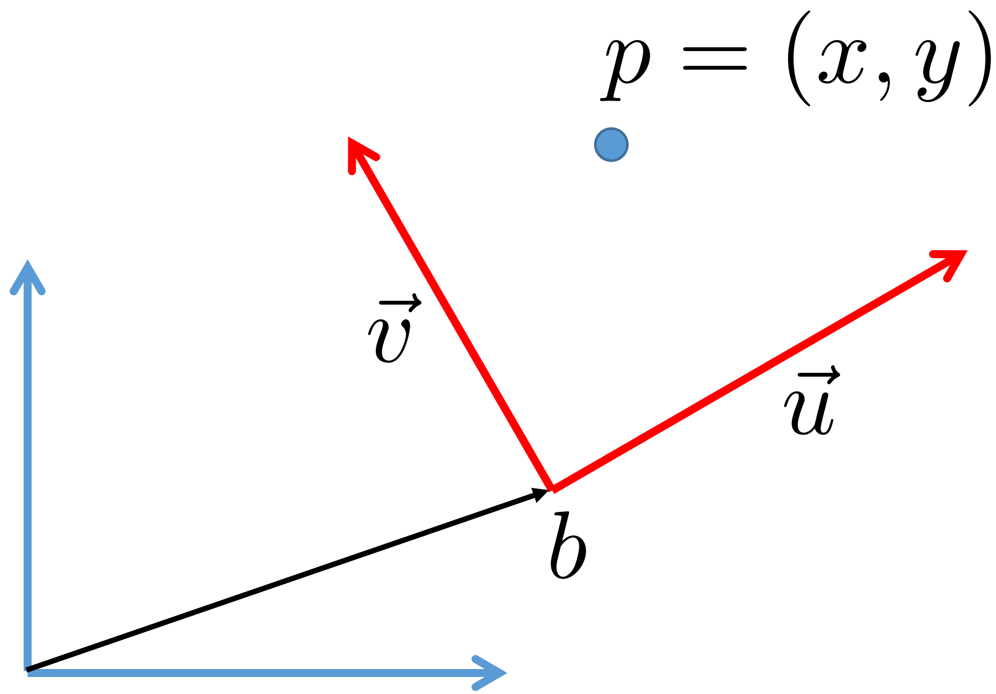


Coordinate System



$$p_g = x\vec{u} + y\vec{v} + b$$

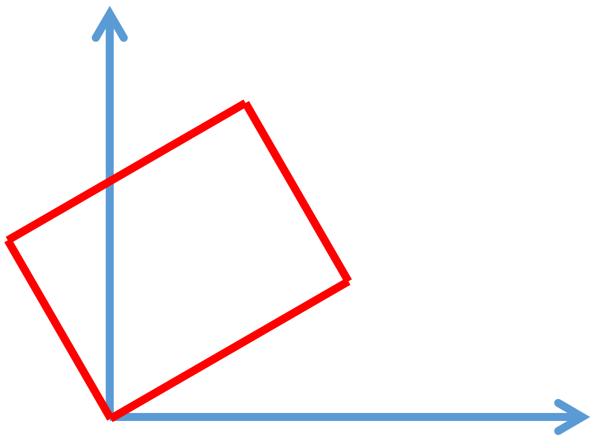
Coordinate System



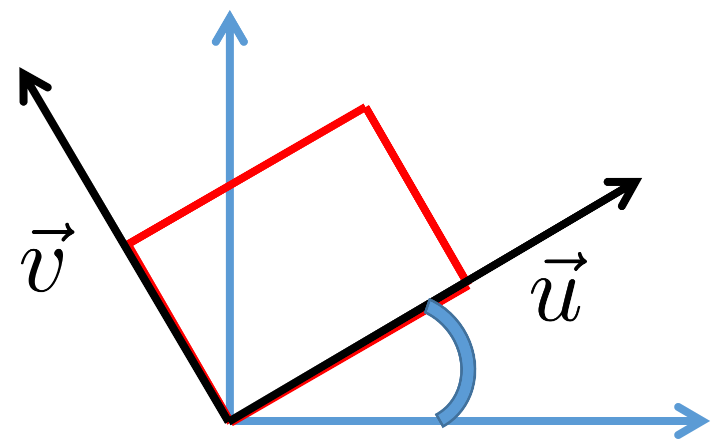
$$p_g = x\vec{u} + y\vec{v} + b$$

$$p_g = \begin{bmatrix} u_1 & v_1 & b_1 \\ u_2 & v_2 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Coordinate System: Rotation

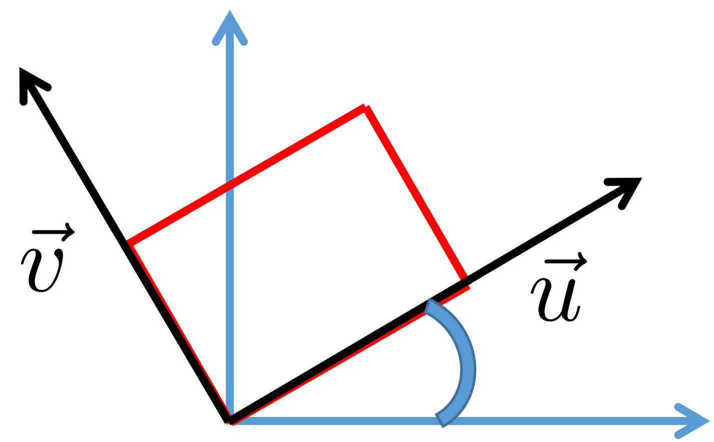


Coordinate System: Rotation



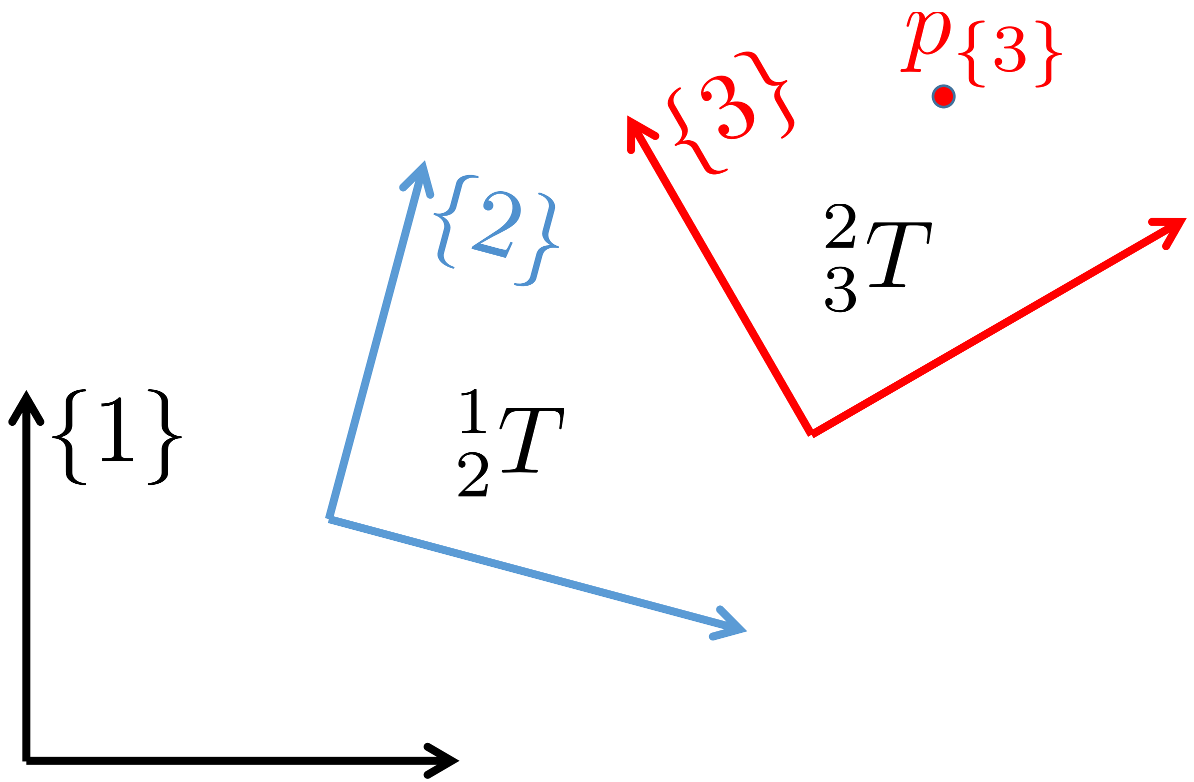
$$\vec{u} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \quad \vec{v} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

Coordinate System: Rotation

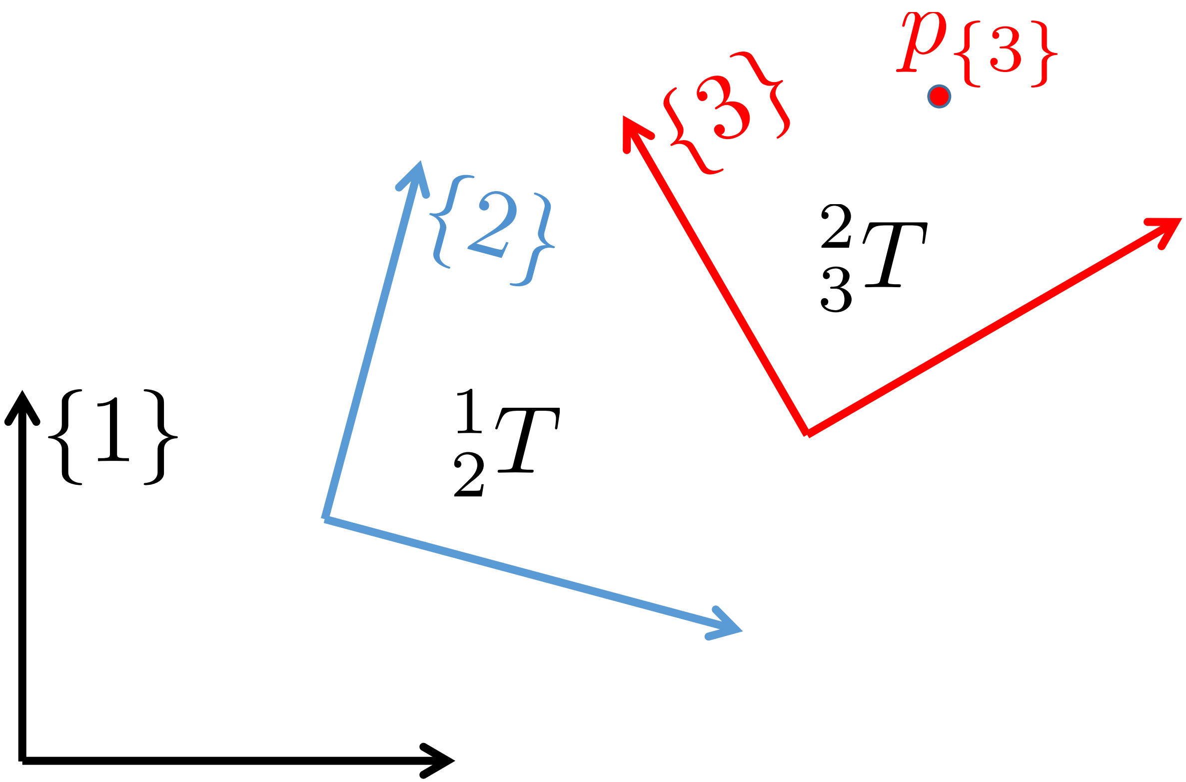


$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & cos\theta \end{bmatrix}$$

Coordinate System: Hierarchy

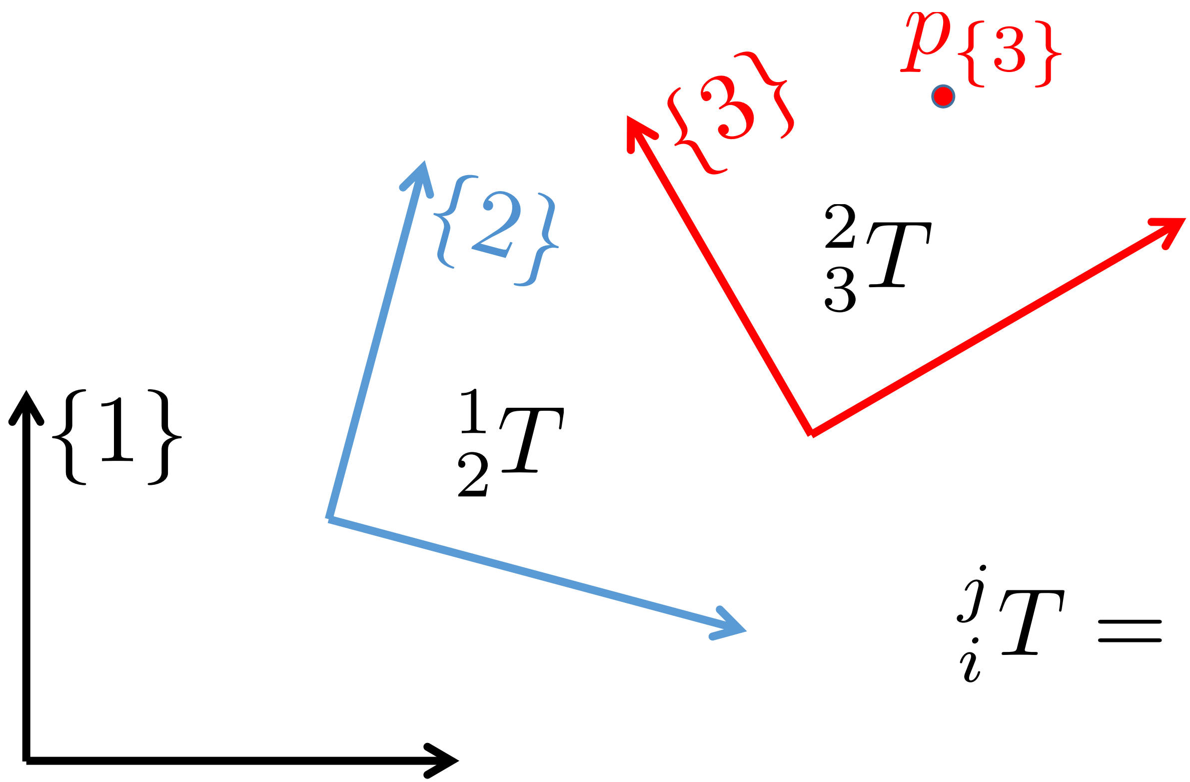


Coordinate System: Hierarchy



$$p_{\{1\}} = {}^{1}_{2}T \; {}^{2}_{3}T \; p_{\{3\}}$$

Coordinate System: Hierarchy



$${}^{j}_{i}T = {}^{i}_{j}T^{-1}$$

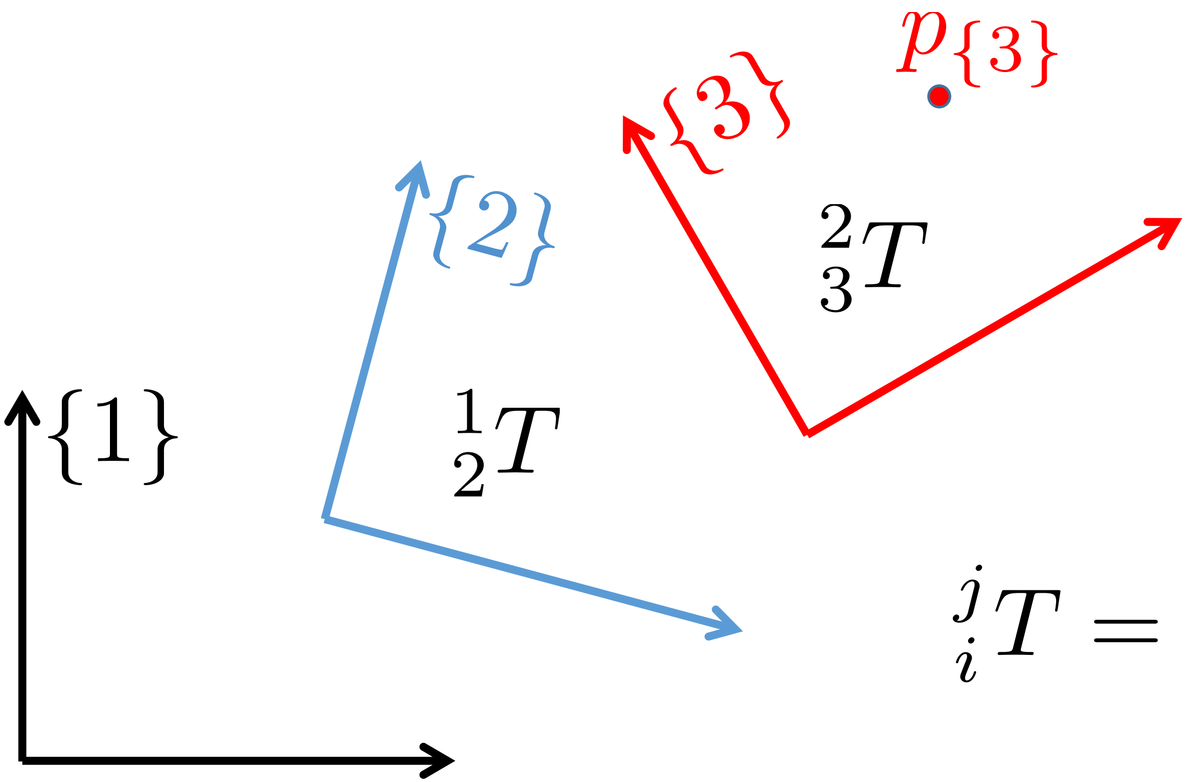
$$p_{\{1\}} = {}^{1}_{2}T \; {}^{2}_{3}T \; p_{\{3\}}$$

Interpreting Transformations

Two Interpretations

- **With respect to Global Frame**
- **With respect to Local Frame**

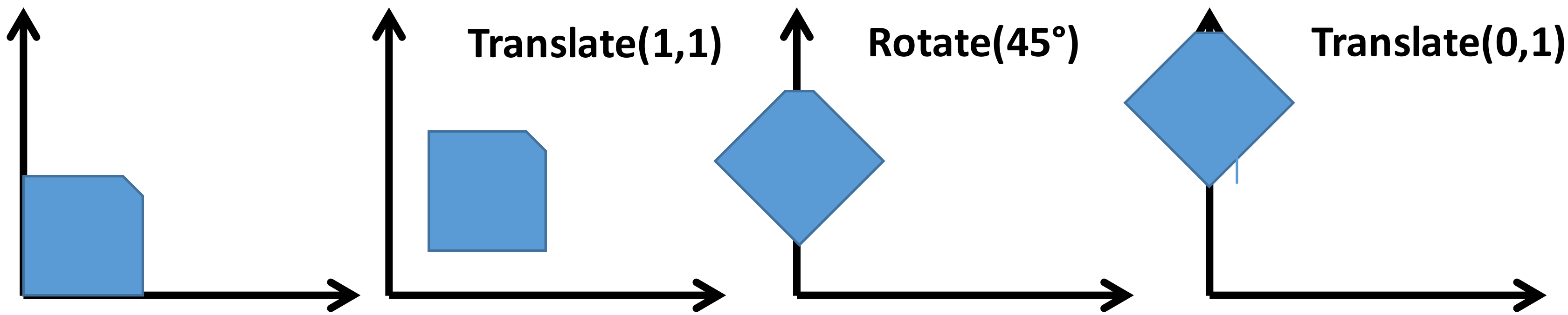
Coordinate System: Hierarchy



$$ {}^{j}_{i}T = {}^{i}_{j}T^{-1} $$

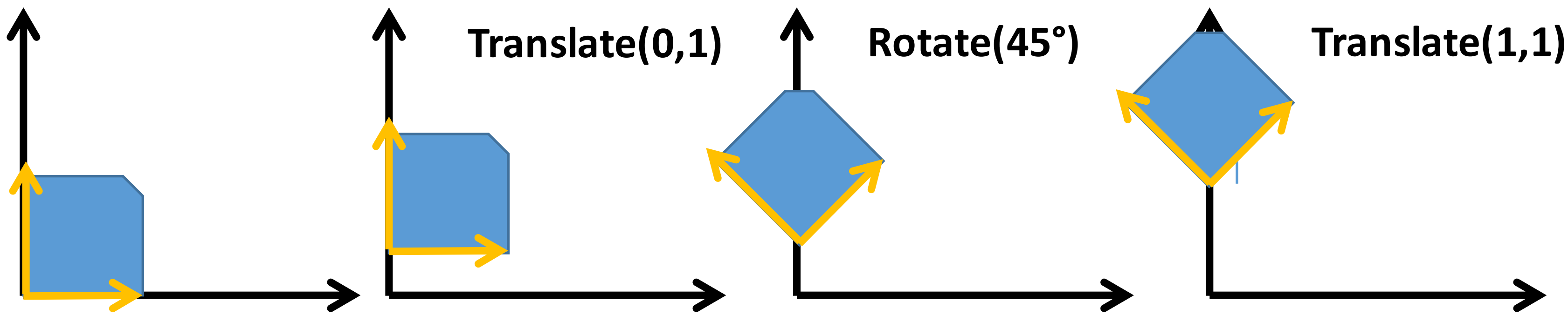
$$ p_{\{1\}} = {}^{1}_{2}T \; {}^{2}_{3}T \; p_{\{3\}} $$

w.r.t Global Frame



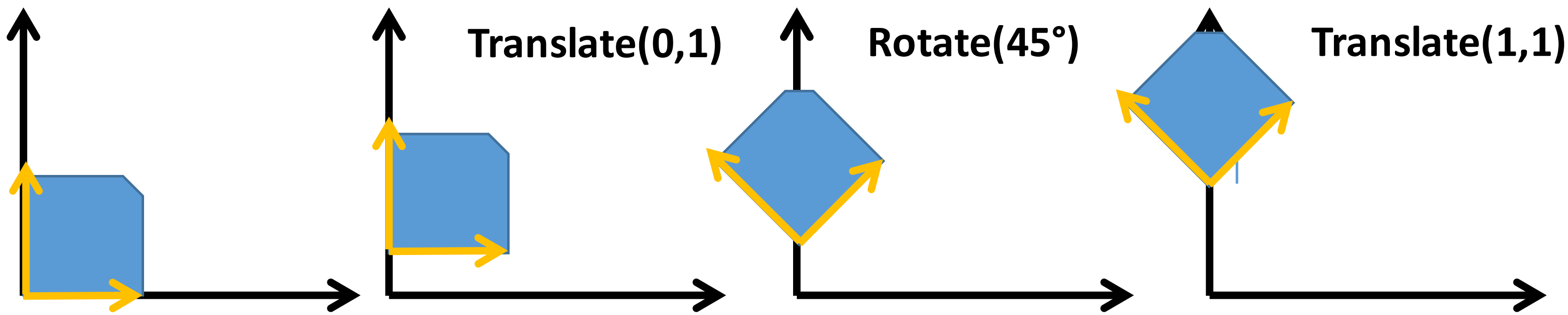
Combined = Translate(0,1) Rotate(45°) Translate(1,1)

w.r.t Local Frame



Combined = Translate(0,1) Rotate(45°) Translate(1,1)

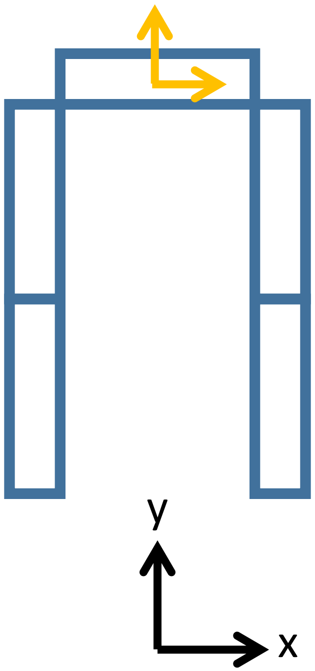
w.r.t Local Frame



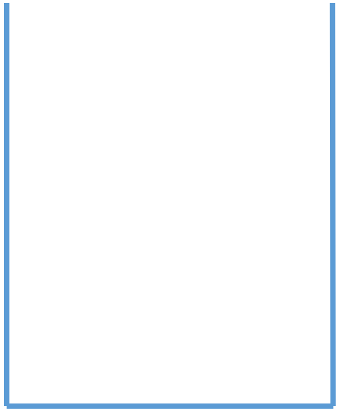
Combined = Translate(0,1) Rotate(45°) Translate(1,1)

Both interpretations are equivalent

Hierarchical Modeling



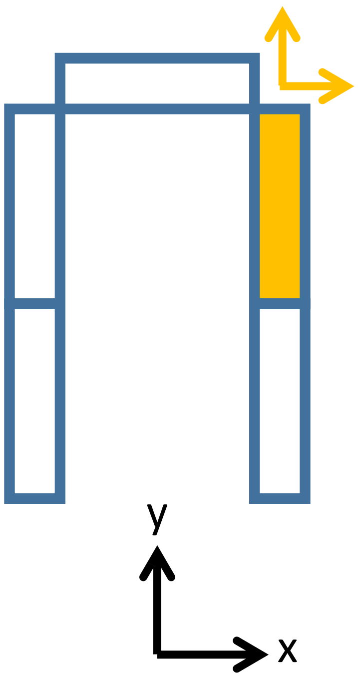

```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()
         ...
    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```



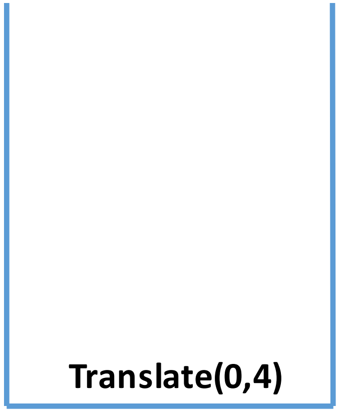
Matrix Stack

CurrentMatrix = Translate(0,4)

Hierarchical Modeling



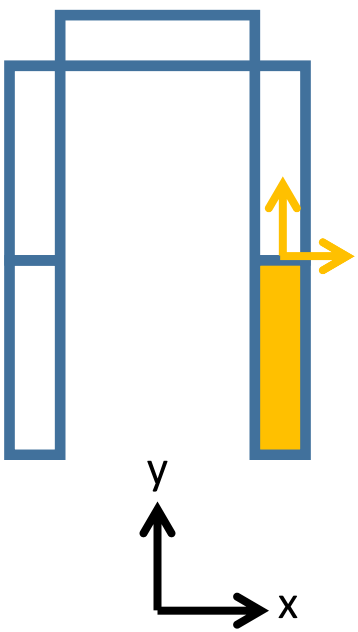

```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()
         ...
    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```



Matrix Stack

CurrentMatrix = Translate(0,4)

Hierarchical Modeling




```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()
         ...
    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```

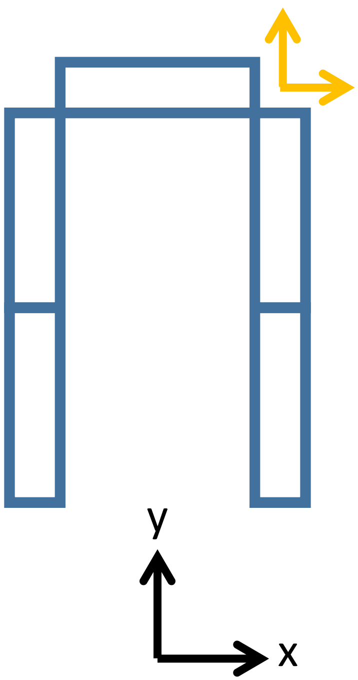
translate(0,4) translate(1.5,0) rotateX(leftHipRotate)

translate(0,4)

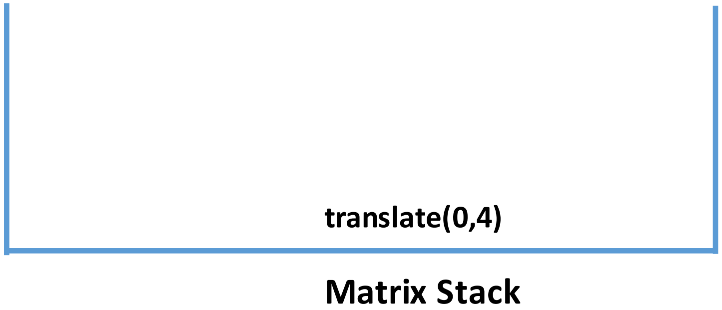
Matrix Stack

CurrentMatrix = translate(0,4) translate(1.5,0) rotateX(leftHipRotate) translate(0,-2) rotate(leftKneeRotate)

Hierarchical Modeling

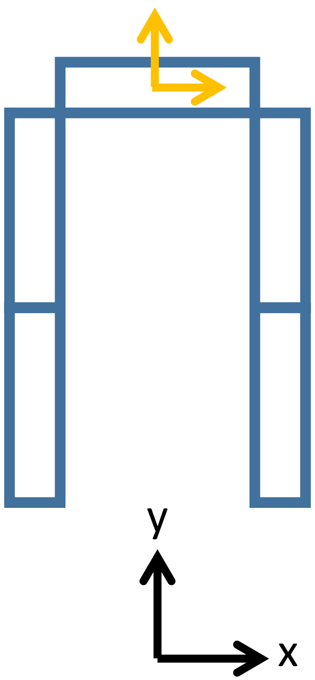



```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()

    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```



CurrentMatrix = translate(0,4) translate(1.5,0) rotateX(leftHipRotate)

Hierarchical Modeling

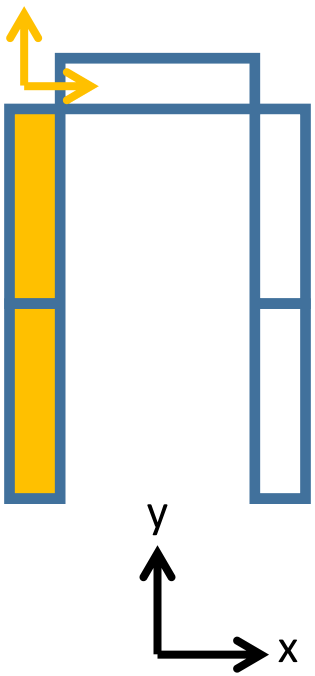



```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()
         ...
    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```

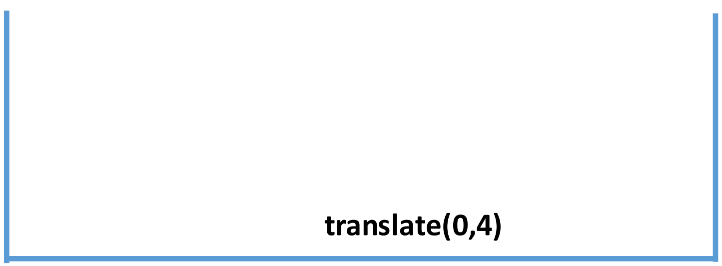
Matrix Stack

CurrentMatrix = translate(0,4)

Hierarchical Modeling



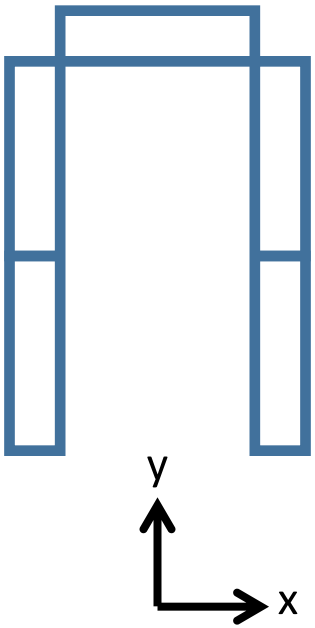

```
translate(0,4)
drawTorso()
pushMatrix()
    translate(1.5,0)
    rotateX(leftHipRotate)
    drawThigh()
    pushMatrix()
        translate(0,-2)
        rotateX(leftKneeRotate)
        drawLeg()
         ...
    popMatrix()
popMatrix()
pushMatrix()
    translate(-1.5,0)
    rotateX(rightHipRotate)
    // Draw the right side
    ...
...
```

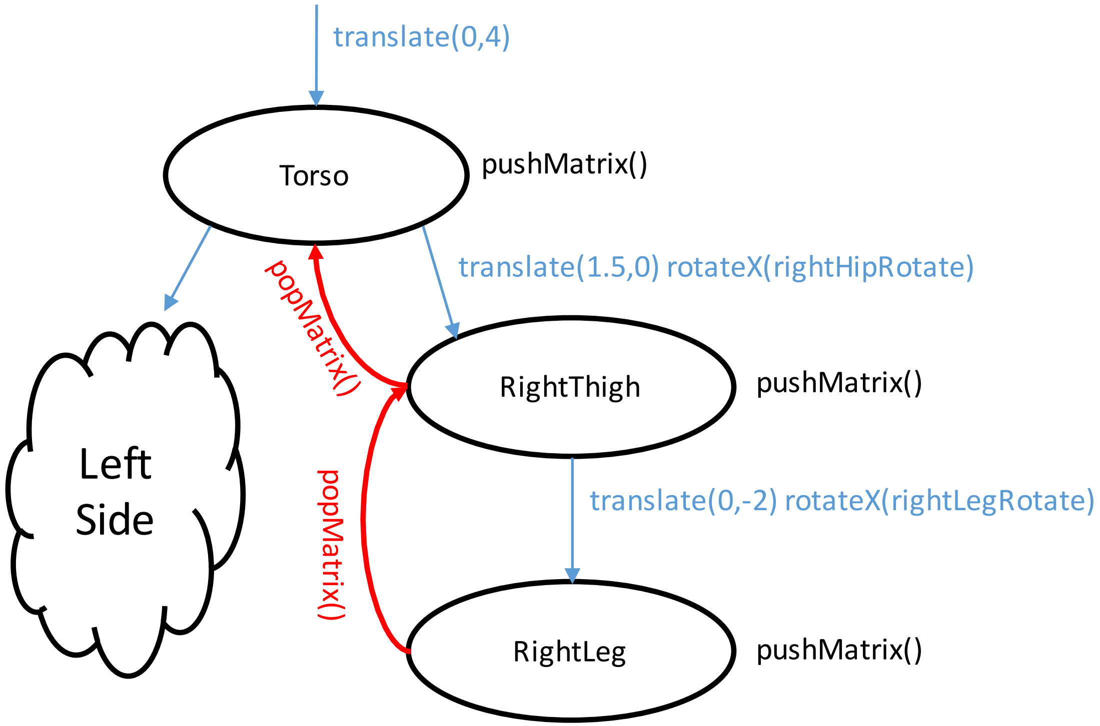


Matrix Stack

CurrentMatrix = translate(0,4) translate(-1.5,0) rotateX(rightHipRotate)

Hierarchical Modeling



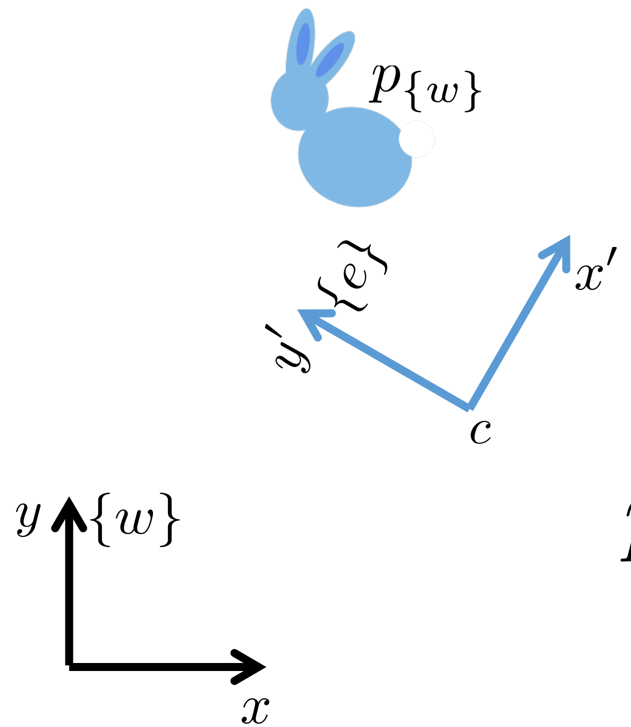


Camera and Projection Matrices

3D to 2D Conversion

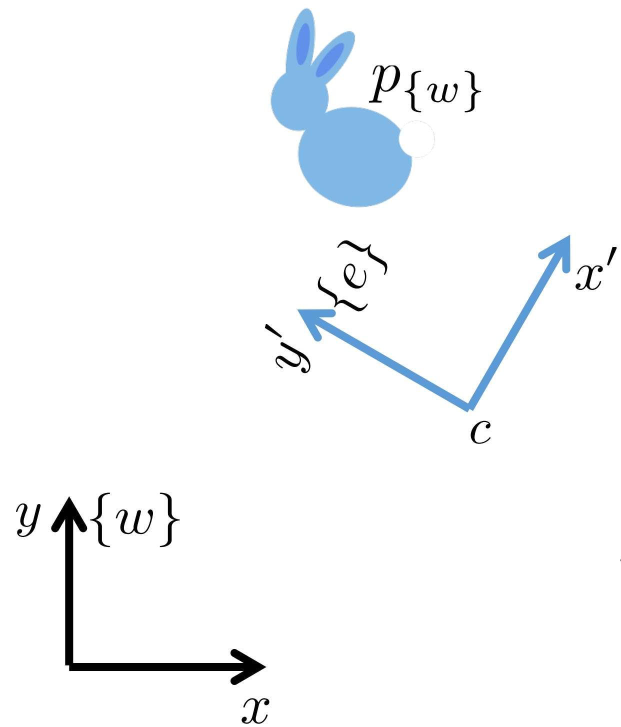
- The world is in 3D
- But our Screen is in 2D
- Imagine our screen is a camera looking out into the world.
- Tasks
    1. Convert 3D world coordinates to Camera Coordinates
    2. Project the Camera Coordinate on 2D screen

Camera Matrix



$$p_{\{e\}} = M p_{\{w\}}$$

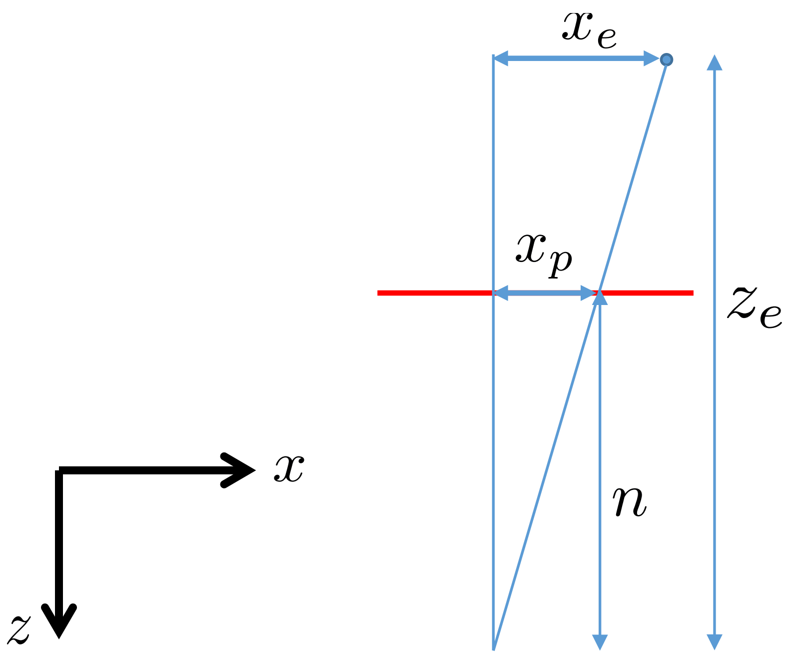
Camera Matrix



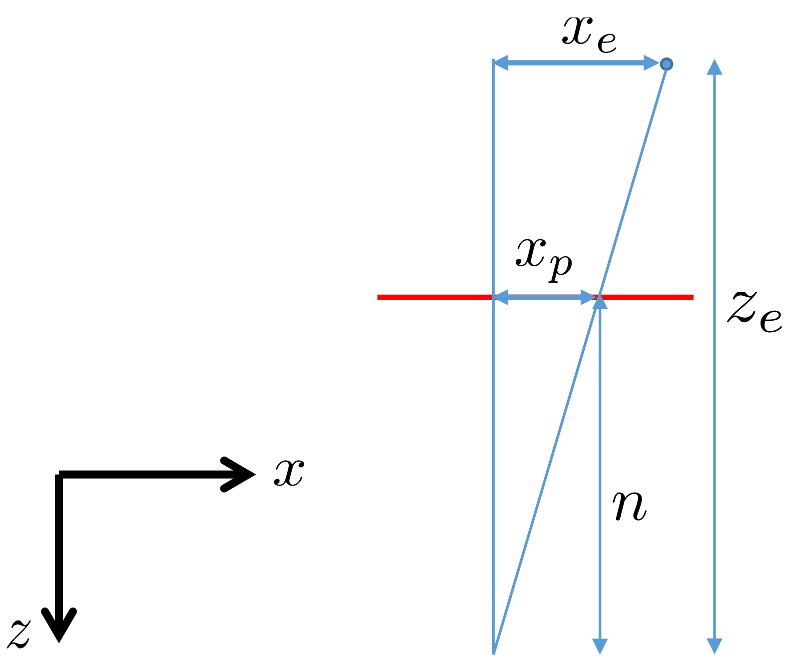
$$p_{\{e\}} = M p_{\{w\}}$$

Homework!

Projection Matrix (Basic)



Projection Matrix (Basic)

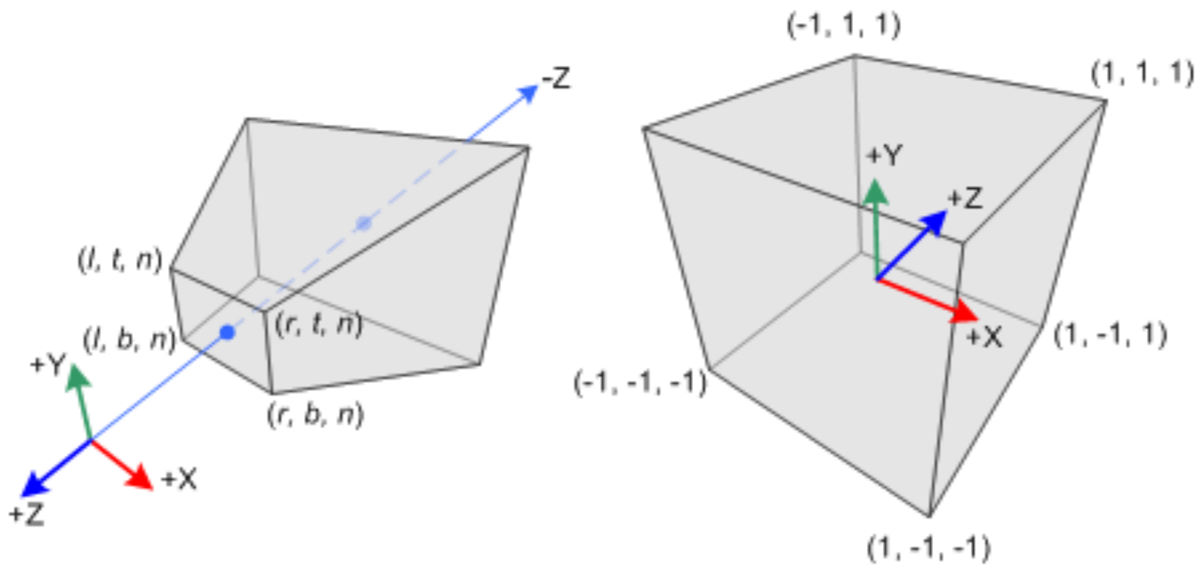


$$\frac{x_p}{x_e} = \frac{n}{z_e}$$

$$x_p = \frac{x_e n}{z_e}$$

$$\begin{bmatrix} nx_e/z_e \\ ny_e/z_e \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} nx_e \\ ny_e \\ nz_e \\ z_e \end{bmatrix} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

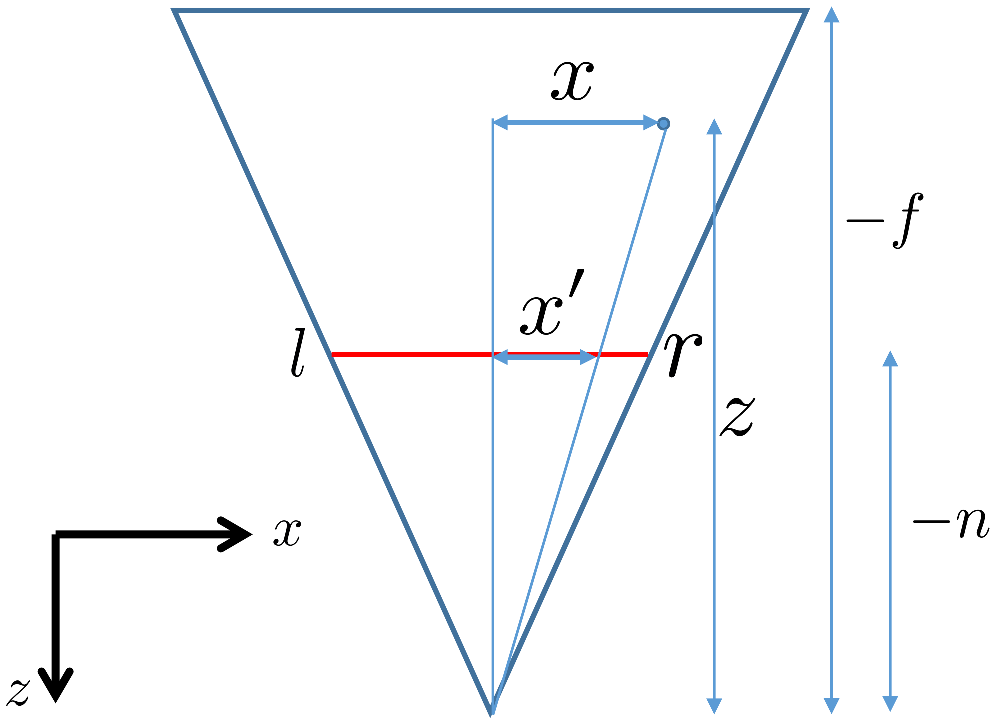
Normalized Device Coordinates



NDC
(Normalized Device Coordinate)

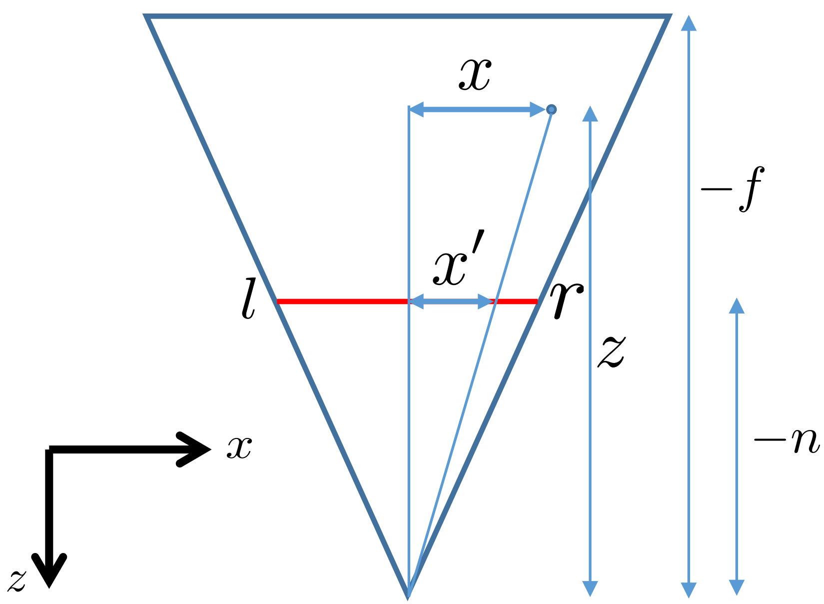
http://www.songho.ca/opengl/gl_projectionmatrix.html

OpenGL Projection Matrix



Note the –n and –f (to be consistent with OpenGL)

OpenGL Projection Matrix

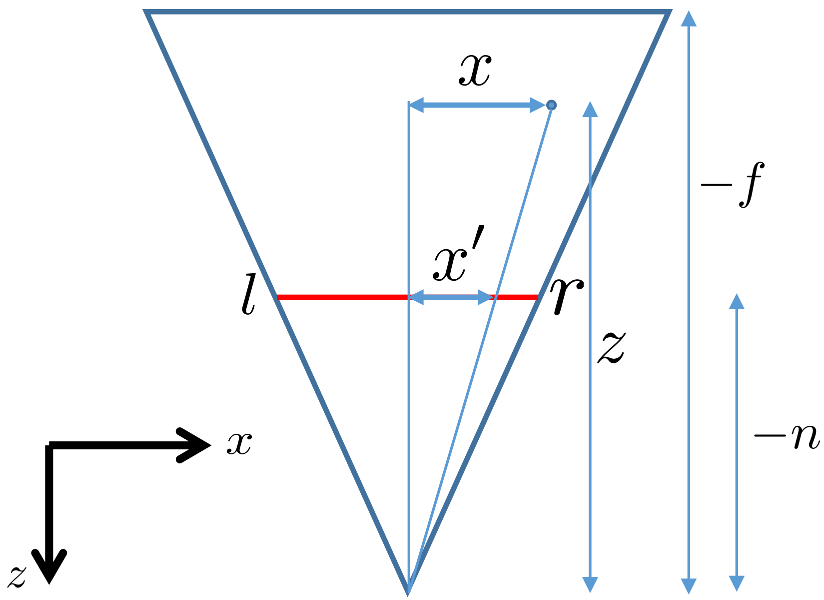


$$x' = \frac{1}{-z}\left[\left(\frac{2n}{r-l}\right)x + \left(\frac{r+l}{r-l}\right)z\right]$$

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$x' = \frac{\hat{x}}{\hat{w}}$$

OpenGL Projection Matrix



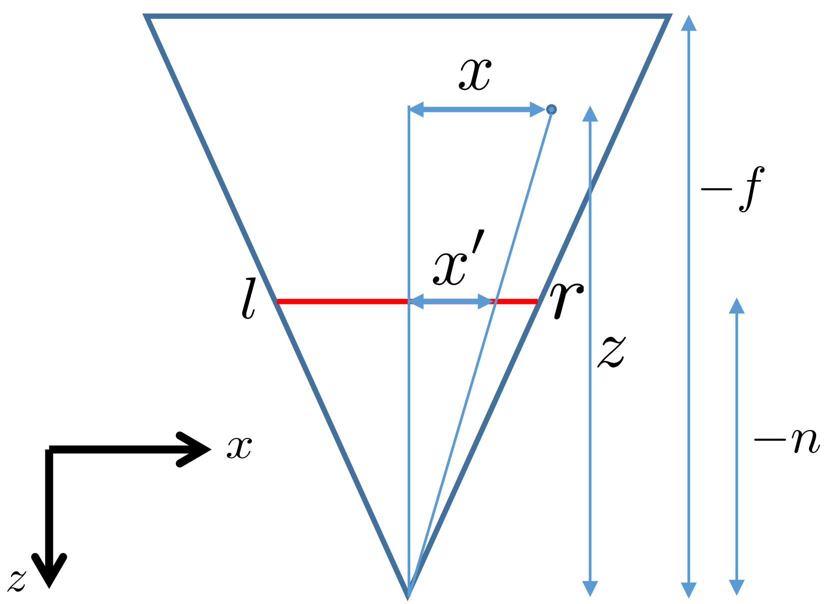
$$x' = \frac{1}{-z}\left[\left(\frac{2n}{r-l}\right)x + \left(\frac{r+l}{r-l}\right)z\right]$$

Similarly for y

$$\begin{bmatrix}\hat{x}\\ \hat{y}\\ \hat{z}\\ \hat{w}\end{bmatrix} = \begin{bmatrix}\frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0\\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & -1 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

$$x' = \frac{\hat{x}}{\hat{w}} \qquad y' = \frac{\hat{y}}{\hat{w}}$$

OpenGL Projection Matrix



z' is a little tricky

$$z' = \frac{Az + B}{-z}$$

$$\text{if } z = -n \text{ then } z' = -1$$

$$\implies -nA + B = -n$$

$$\text{if } z = -f \text{ then } z' = 1$$

$$\implies -fA + B = f$$

Solving for A and B

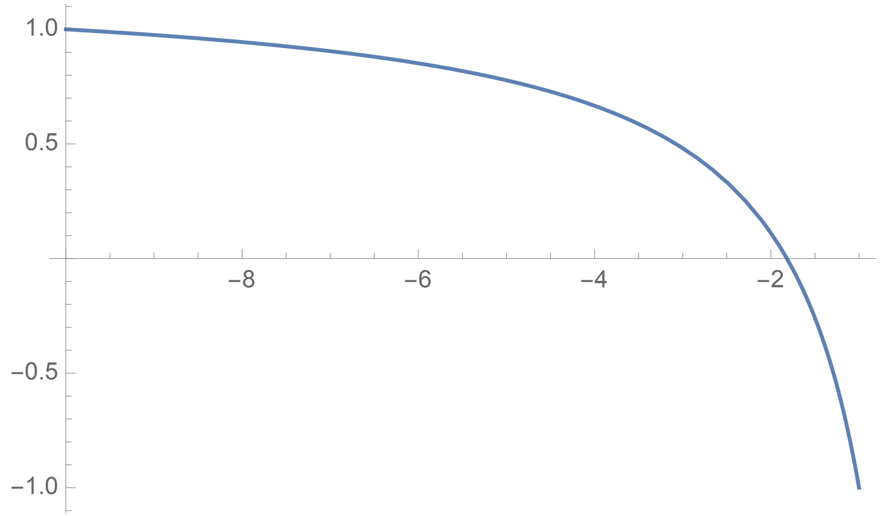
$$A = \frac{f+n}{n-f}, \quad B = \frac{2fn}{n-f}$$

OpenGL Projection Matrix

$$
\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

Non-Linearity in Z

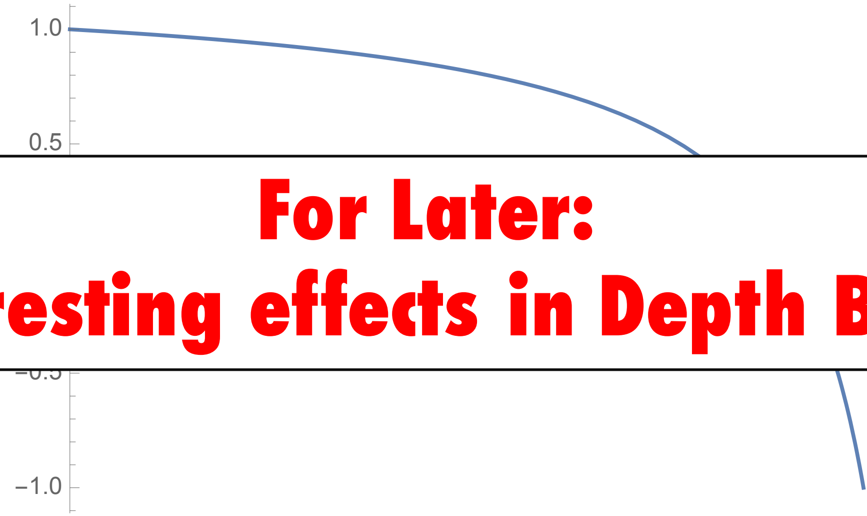
$$z' = -\frac{f+n}{n-f} - \left(\frac{2fn}{n-f}\right)\frac{1}{z}$$



1. **Monotonic: values keeps increasing as z goes in –ve direction**
2. **Resolution decreases as z decrease (along –ve) or depth increases**

Non-Linearity in Z

$$z' = -\frac{f+n}{n-f} - \left(\frac{2fn}{n-f}\right)\frac{1}{z}$$

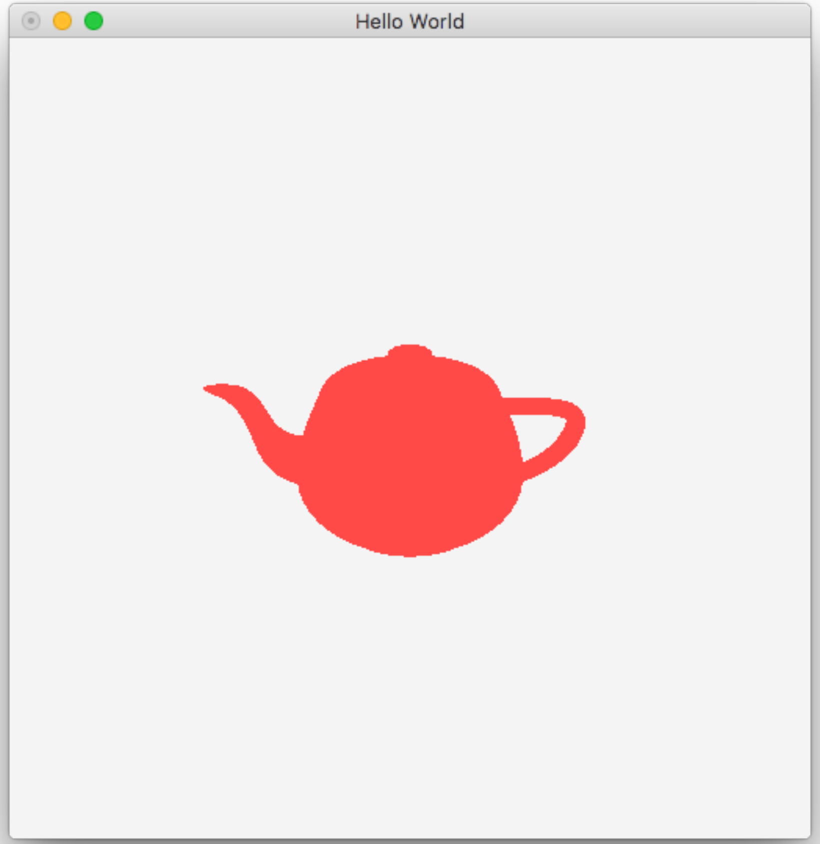


For Later:
Has interesting effects in Depth Buffering

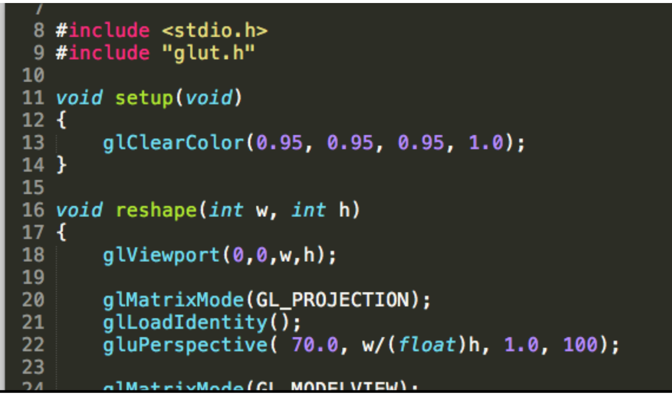
1. **Monotonic: values keeps increasing as z goes in –ve direction**
2. **Resolution decreases as z decrease (along –ve) or depth increases**

Start Learning OpenGL Now

```c
#include <stdio.h>
#include "glut.h"

void setup(void)
{
    glClearColor(0.95, 0.95, 0.95, 1.0);
}

void reshape(int w, int h)
{
    glViewport(0,0,w,h);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective( 70.0, w/(float)h, 1.0, 100);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    gluLookAt(0,0,-5,0,0,0,0,1,0);

    glRotate3f(-30,1,0,0);
    glColor3f(1.0, 0.25, 0.25);
    glutSolidTeapot(1);
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);

    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(512,512);
    glutInitWindowPosition(50,50);
    glutCreateWindow("Hello World");

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    setup();

    glutMainLoop();
}
```



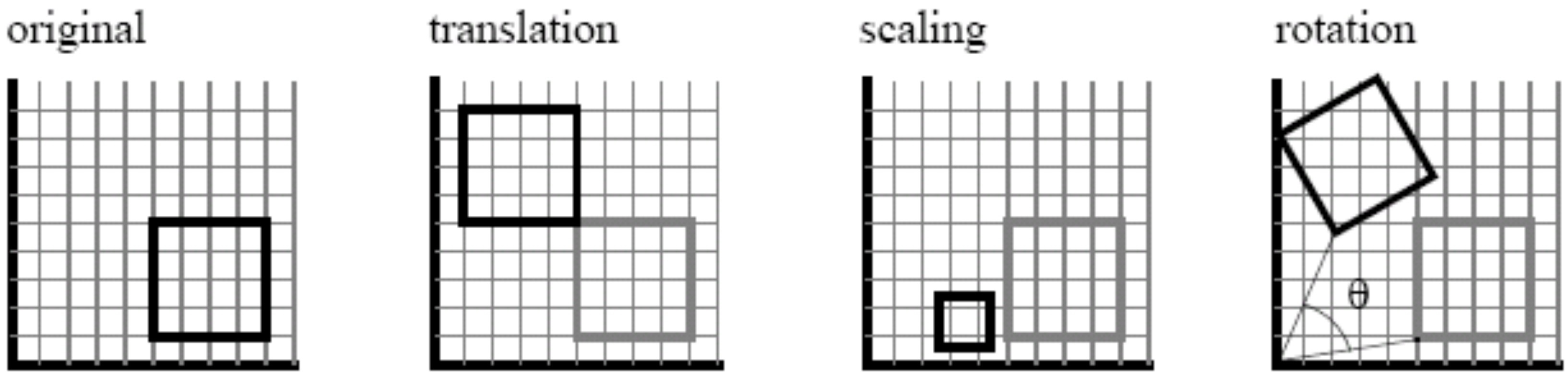
Start Learning OpenGL Now



Suggestions Only

For OpenGL 3.3: http://learnopengl.com

For OpenGL < 2.0: Any GLUT or FreeGLUT tutorial



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

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

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

Transformation

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

http://www.pling.org.uk/cs/cgv.html