ITIC

Information Technology Innovation Center

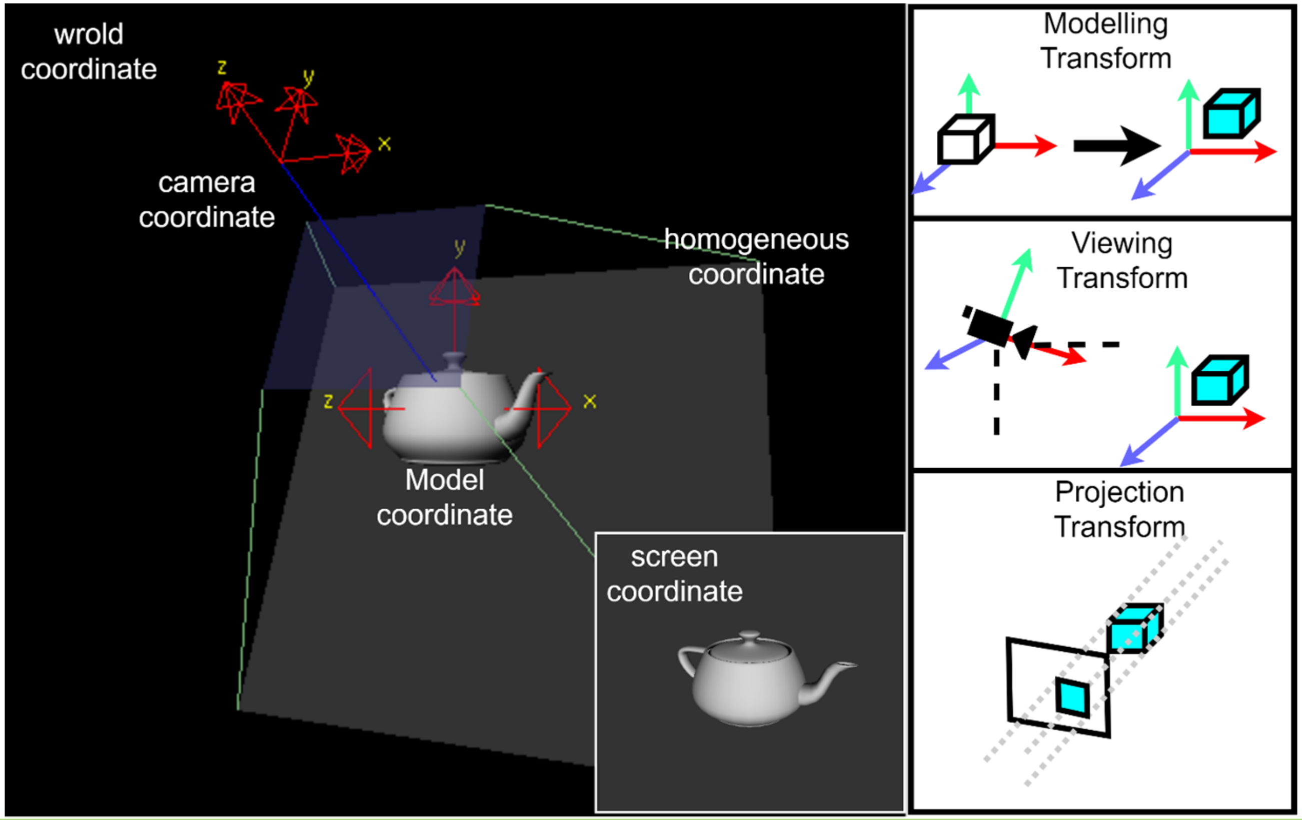
基礎電腦圖學

Ch03.基本數學相機與投影

Objective

- Vector & Matrix。
- Coordinate systems and Transformations。
- Camera system and projection。
- Coordinate system relevant To Rendering。

What You'll Learn in This Lecture



Vector & Matrix

$$\begin{bmatrix} \mathrm{a} & b \\ \mathrm{c} & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

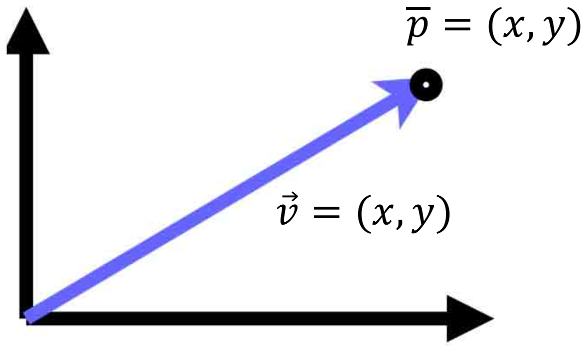
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

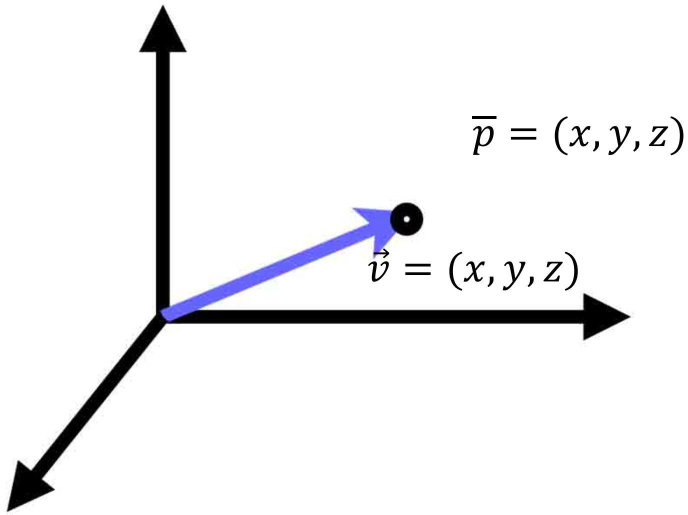
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$

Vector

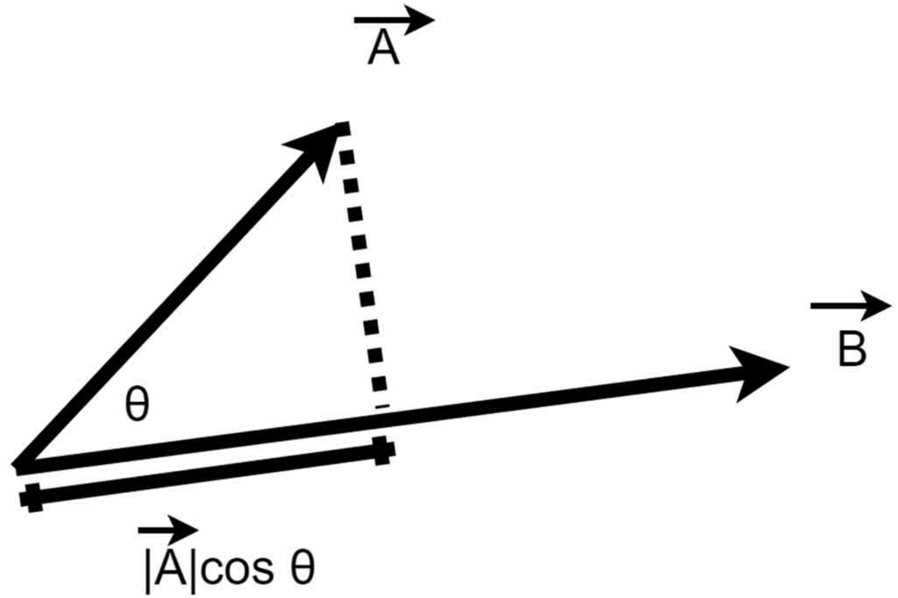
- A vector can be thought of a ***point*** in a space or a ***direction*** from the origin to the point





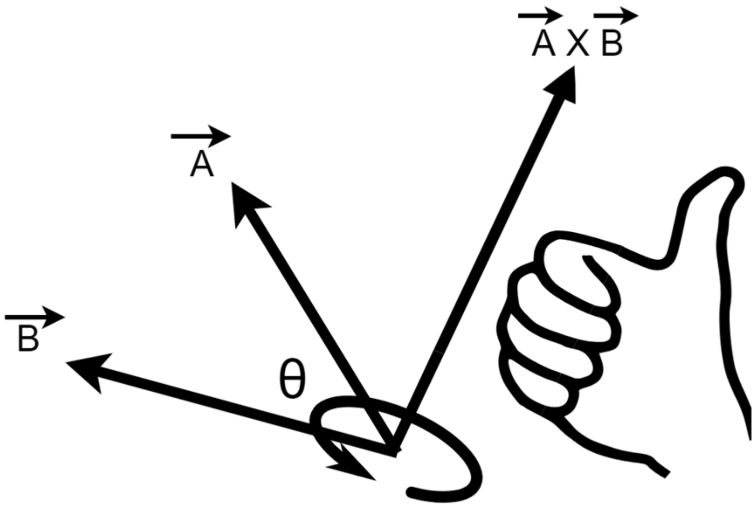
Vector Dot Product

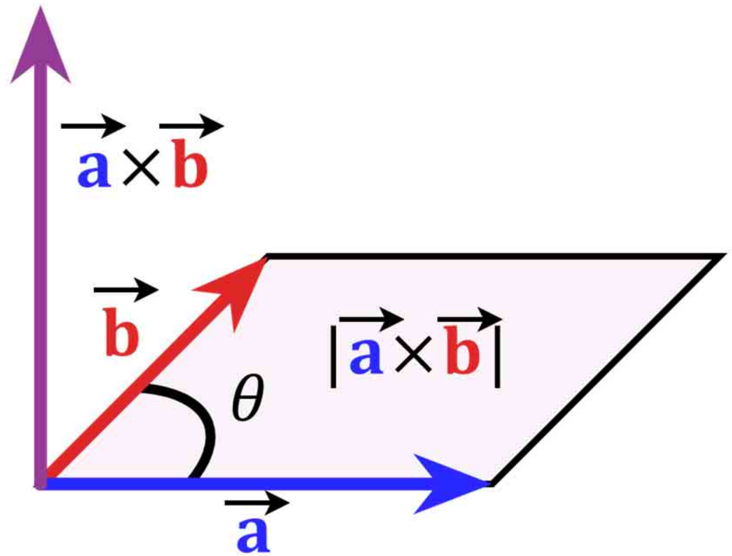
$$\vec{\mathrm{A}} \cdot \vec{\mathrm{B}} = \sum_{i}^{n} \vec{\mathrm{A}}_i \cdot \vec{\mathrm{B}}_i = \left|\vec{\mathrm{A}}\right| \left|\vec{\mathrm{B}}\right| \cos\theta$$



Vector Cross Product

- Rule of right hand $\vec{A} \times \vec{B} = |\vec{A}|\,|\vec{B}| \cdot \sin\theta\, \boldsymbol{n}$, where ***n* =** the length of unit vector





Matrix

- A **Matrix** is a rectangle array of number, symbols or expressions that arrange in **rows and columns**

m-by-n matrix

$a_{i,j}$ n columns j changes

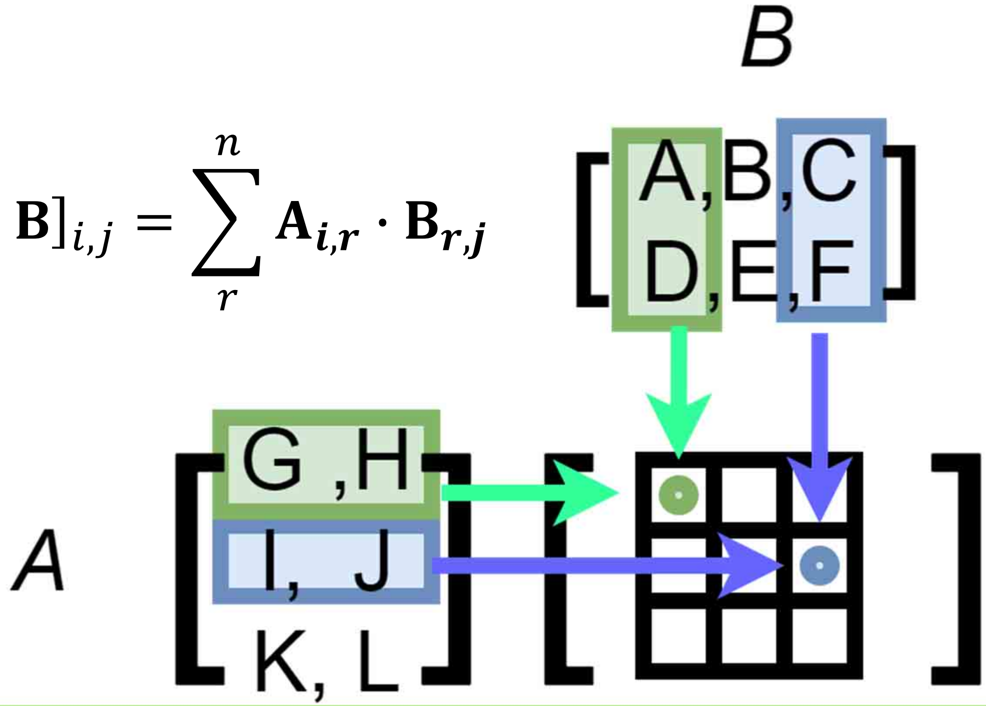
m rows i changes

$$
\begin{bmatrix}
a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\
a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\
a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
$$

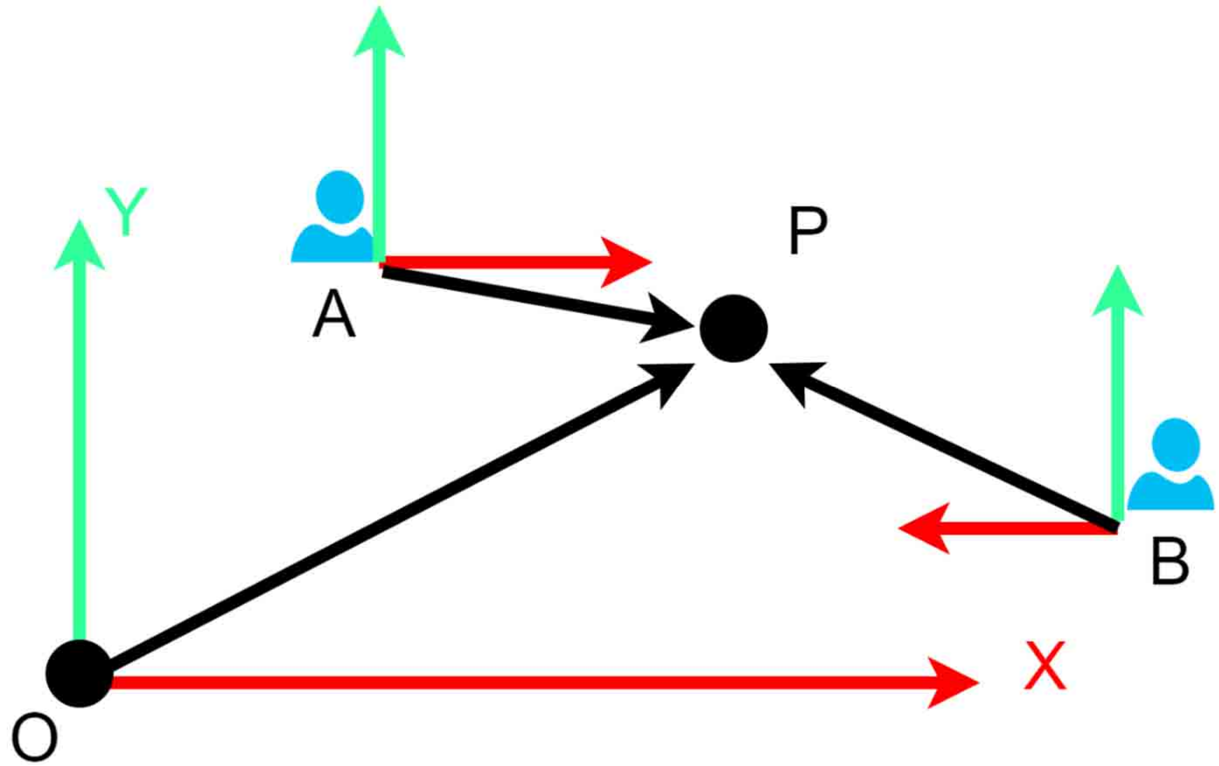
Matrix Product

- if $\mathbf{A}$ is m $\times$ n matrix & $\mathbf{B}$ is n $\times$ p matrx $\rightarrow$ $\mathbf{A} \cdot \mathbf{B}$ is m $\times$ p matrix

$$[\mathbf{A} \cdot \mathbf{B}]_{i,j} = \sum_{r}^{n} \mathbf{A}_{i,r} \cdot \mathbf{B}_{r,j}$$



Coordinate Systems and Transformations

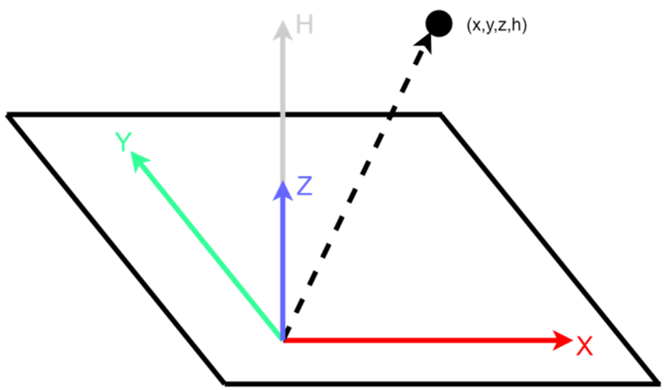


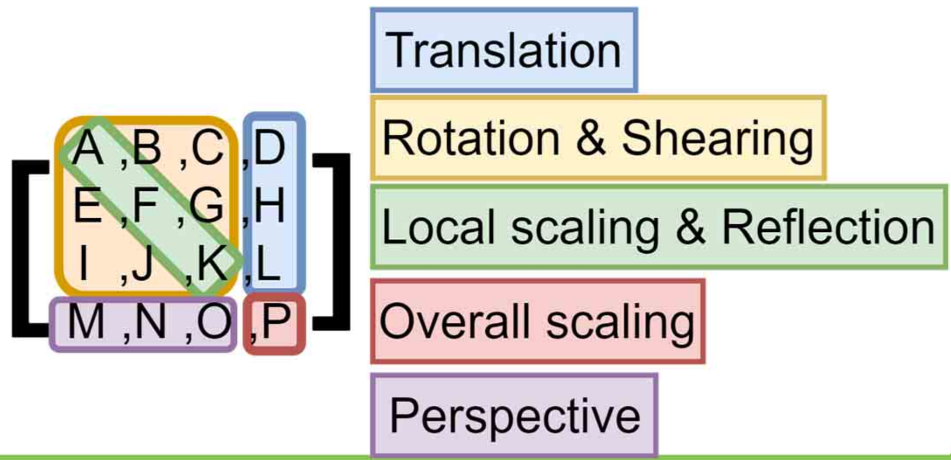
Why Different Coordinate

- Different coordinate in space has it’s own purpose and functionality.
  - The geometry date in object is more convenient to descried in object space.
  - The relation between objects is more convenient to descried in world space .
  - In Projection system sequence, object’s data needs to transfer to each relevant coordinate space to correctly rendering on the screen.

Transformation matrix & coordinate

- To generalize transformation, the object's coordinate will be transfer to 3D homogeneous coordinate.
- The transformation(translation, rotation…) can be stored into 4×4 transformation matrix.

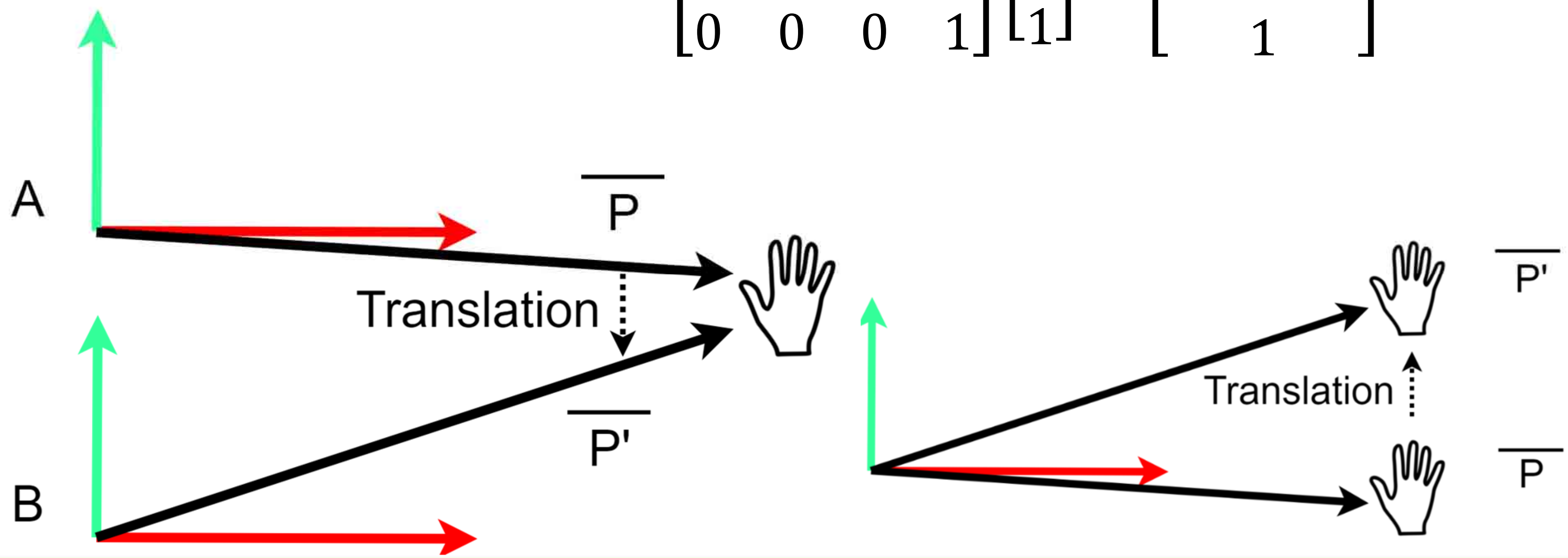




Translation Transformation

$$\overline{\mathrm{P}}(\mathrm{x}, \mathrm{y}, \mathrm{z}) \text{ translate } (\mathrm{a}, \mathrm{b}, \mathrm{c}) \text{in space to } \overline{\mathrm{P}'}$$

$$\mathbf{M}_{\mathrm{translation}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} 1 & 0 & 0 & \mathrm{a} \\ 0 & 1 & 0 & \mathrm{b} \\ 0 & 0 & 1 & \mathrm{c} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{x} \cdot 1 + \mathrm{a} \\ \mathrm{y} \cdot 1 + \mathrm{b} \\ \mathrm{z} \cdot 1 + \mathrm{c} \\ 1 \end{bmatrix}$$

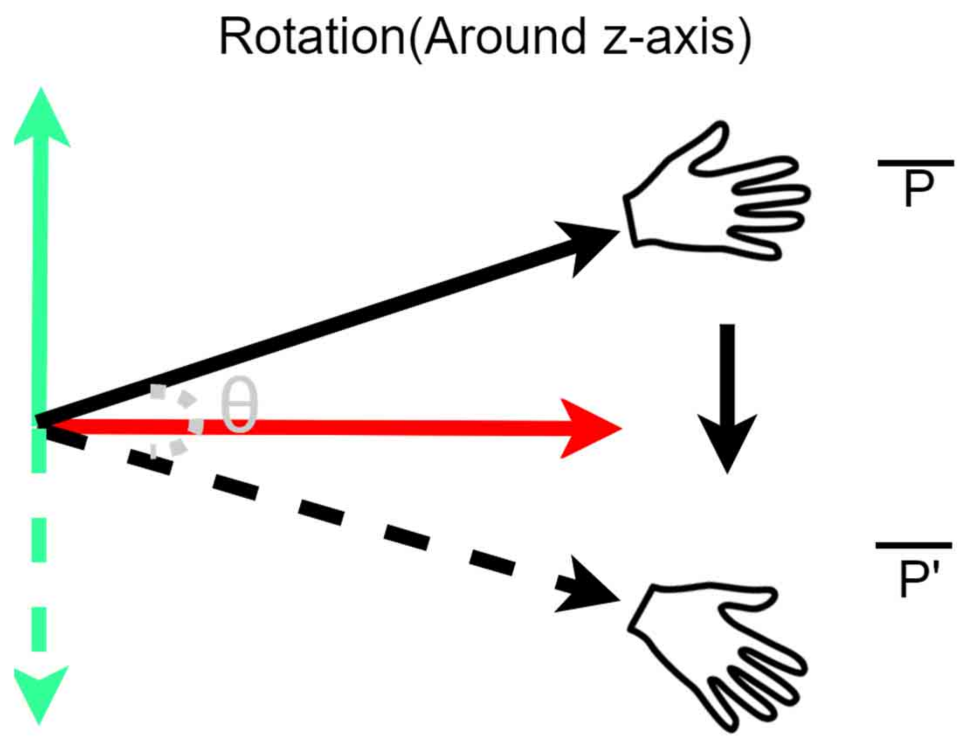


Thought of Transformation in Spaces

"*I understand how the engines work now. It came to me in a dream. The engines don't move the ship at all. The ship stays where it is and the engines move the universe around it.*"

— Futurama

Rotation Transformation



Rotation Transformation

$\overline{\mathrm{P}}(x, y, z)$ rotate around x_axis with θ degreed in space to $\overline{\mathrm{P}'}$

$$\mathbf{M}_{\mathrm{rotation}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{bmatrix}$$

$\overline{\mathrm{P}}(x, y, z)$ rotate around y_axis with θ degreed in space to $\overline{\mathrm{P}'}$

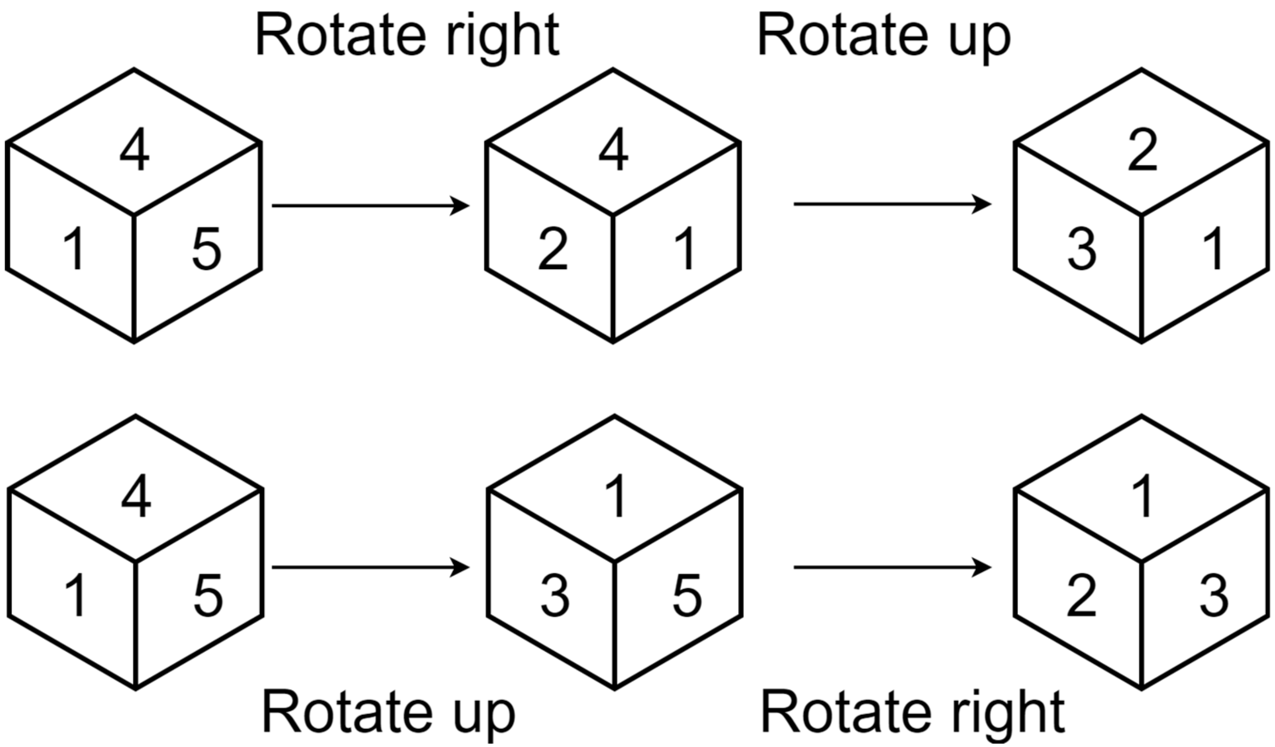
$$\mathbf{M}_{\mathrm{rotation}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{bmatrix}$$

Rotation Transformation(cont.)

$\overline{\mathrm{P}}(\mathrm{x},\mathrm{y},\mathrm{z})$ rotate around z_axis with θ degreed in space to $\overline{\mathrm{P}'}$

$$\mathbf{M}_{\mathrm{rotation}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta \cdot \mathrm{x} - \sin\theta \cdot \mathrm{y} \\ \sin\theta \cdot \mathrm{x} + \cos\theta \cdot \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix}$$

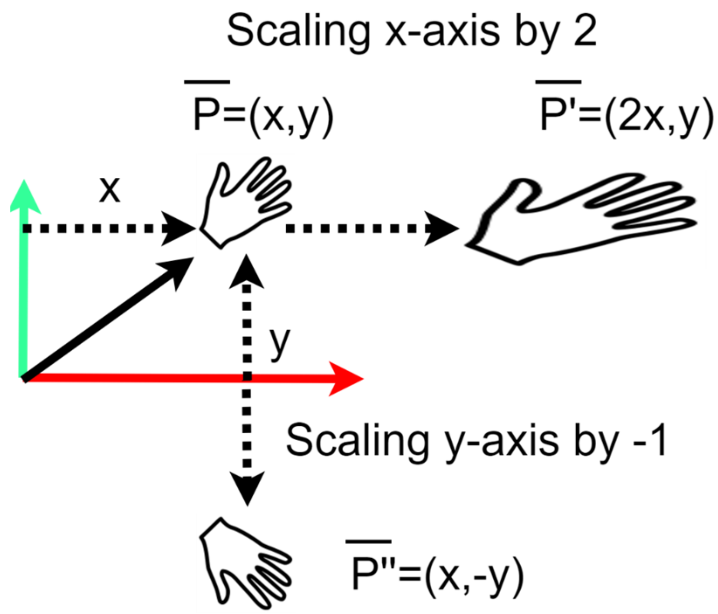
Different in Rotation Order



Scaling Transformation

$\overline{\mathrm{P}}(\mathrm{x}, \mathrm{y}, \mathrm{z})$ scaling with each axis by $(\mathrm{a}, \mathrm{b}, \mathrm{c})$ in space to $\overline{\mathrm{P}'}$

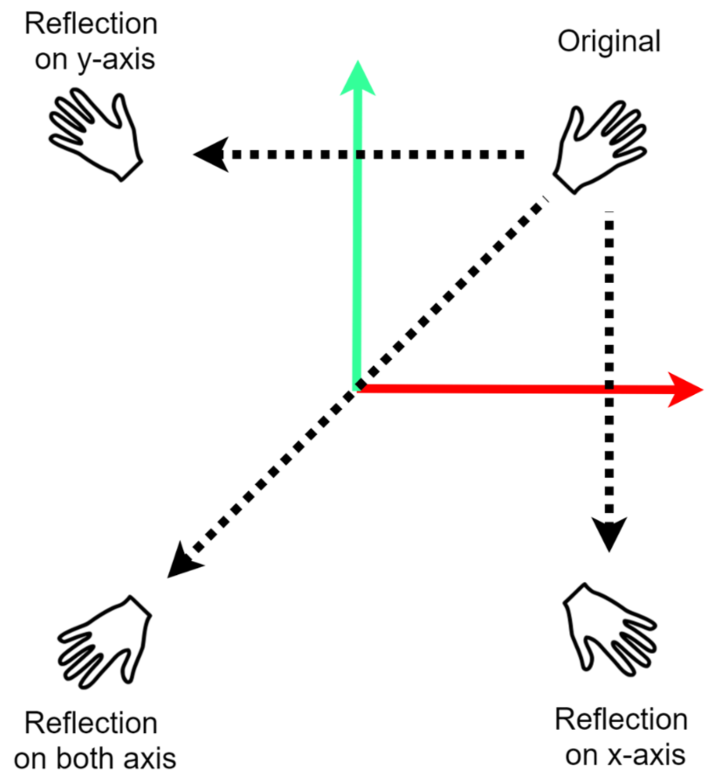
$$\mathbf{M}_{\text{scaling}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a \cdot x \\ b \cdot y \\ c \cdot z \\ 1 \end{bmatrix}$$



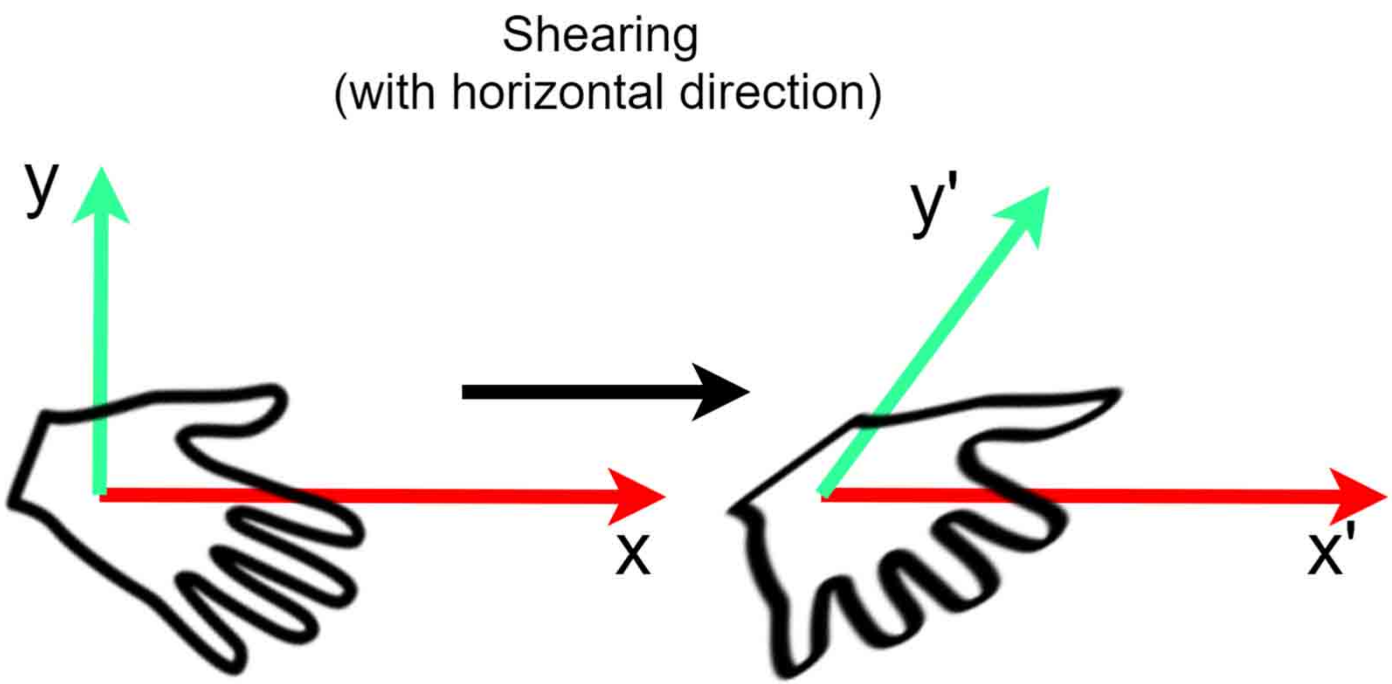
Reflection Transformation

$\overline{\mathrm{P}}(\mathrm{x},\mathrm{y},\mathrm{z})$ Reflection by x_axis in space to $\overline{\mathrm{P}'}$

$$\mathbf{M}_{\text{reflection(x)}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} -\mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix}$$



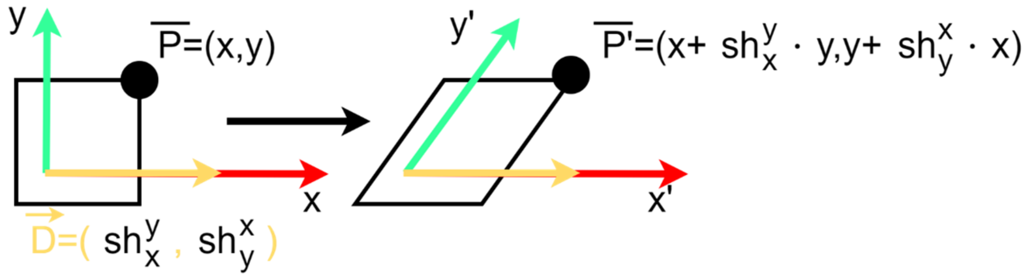
Shearing Transformation



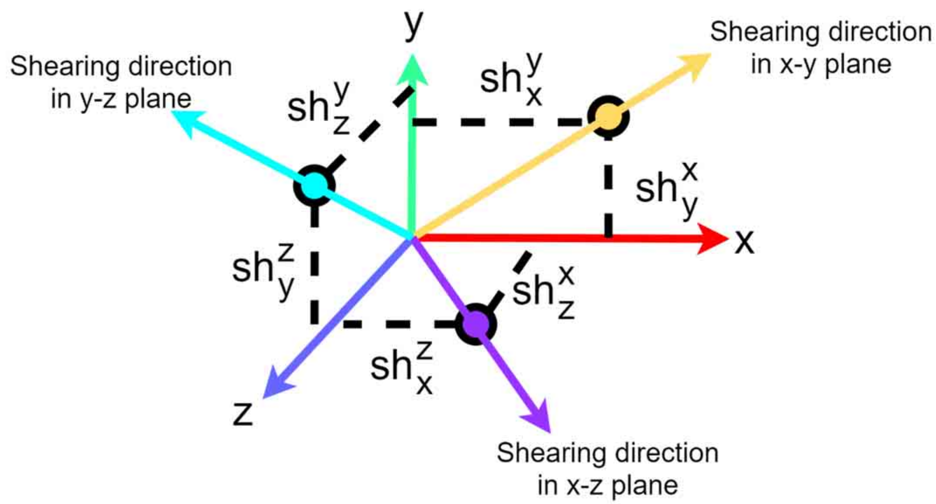
Shearing in 2D/3D Dimension

2D: one plane shearing

Shearing with direction D

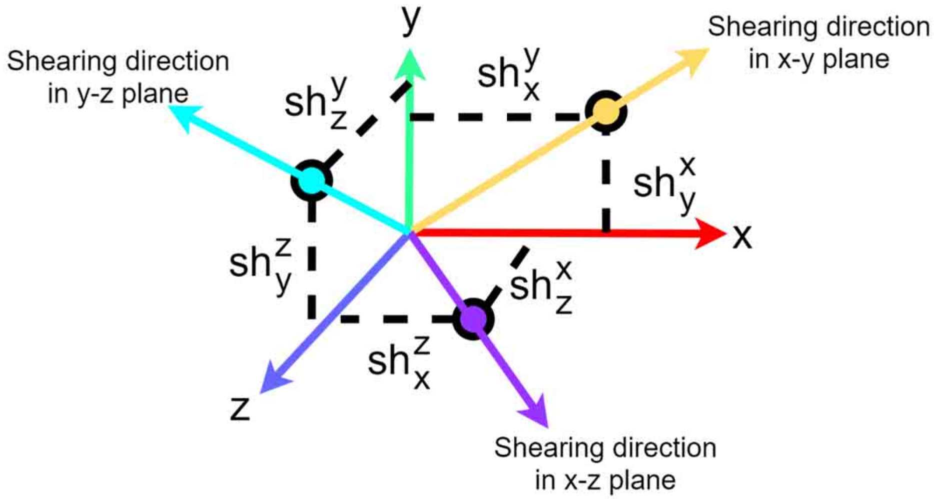


3D: three plane shearing



Shearing Transformation

$\overline{\mathrm{P}}(\mathrm{x}, \mathrm{y}, \mathrm{z})$ Shearing by each plane in space to $\overline{\mathrm{P}'}$



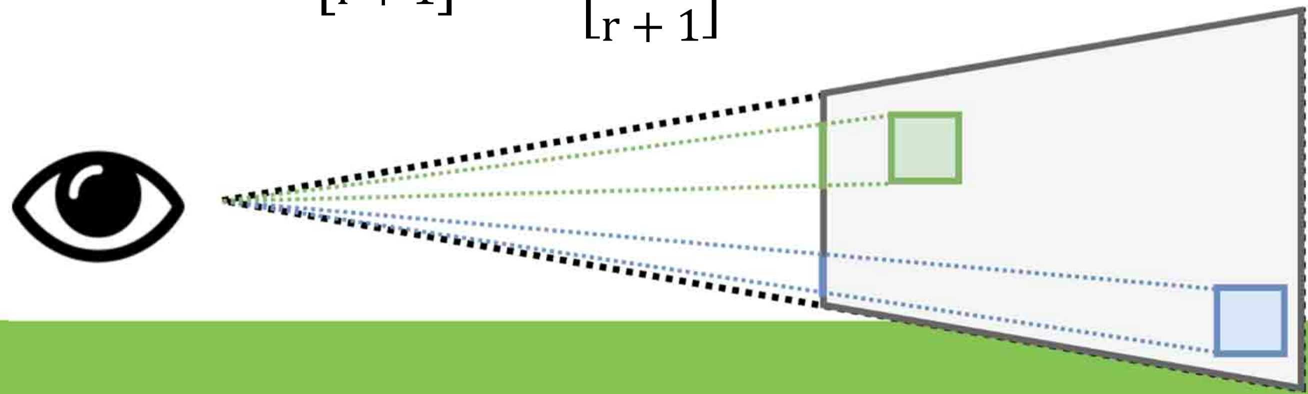
$$\mathbf{M}_{\text{shearing}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + sh_x^y \cdot y + sh_x^z \cdot z \\ sh_y^x \cdot x + y + sh_y^z \cdot z \\ sh_z^x \cdot x + sh_z^y \cdot y + z \\ 1 \end{bmatrix}$$

Perspective Transformation

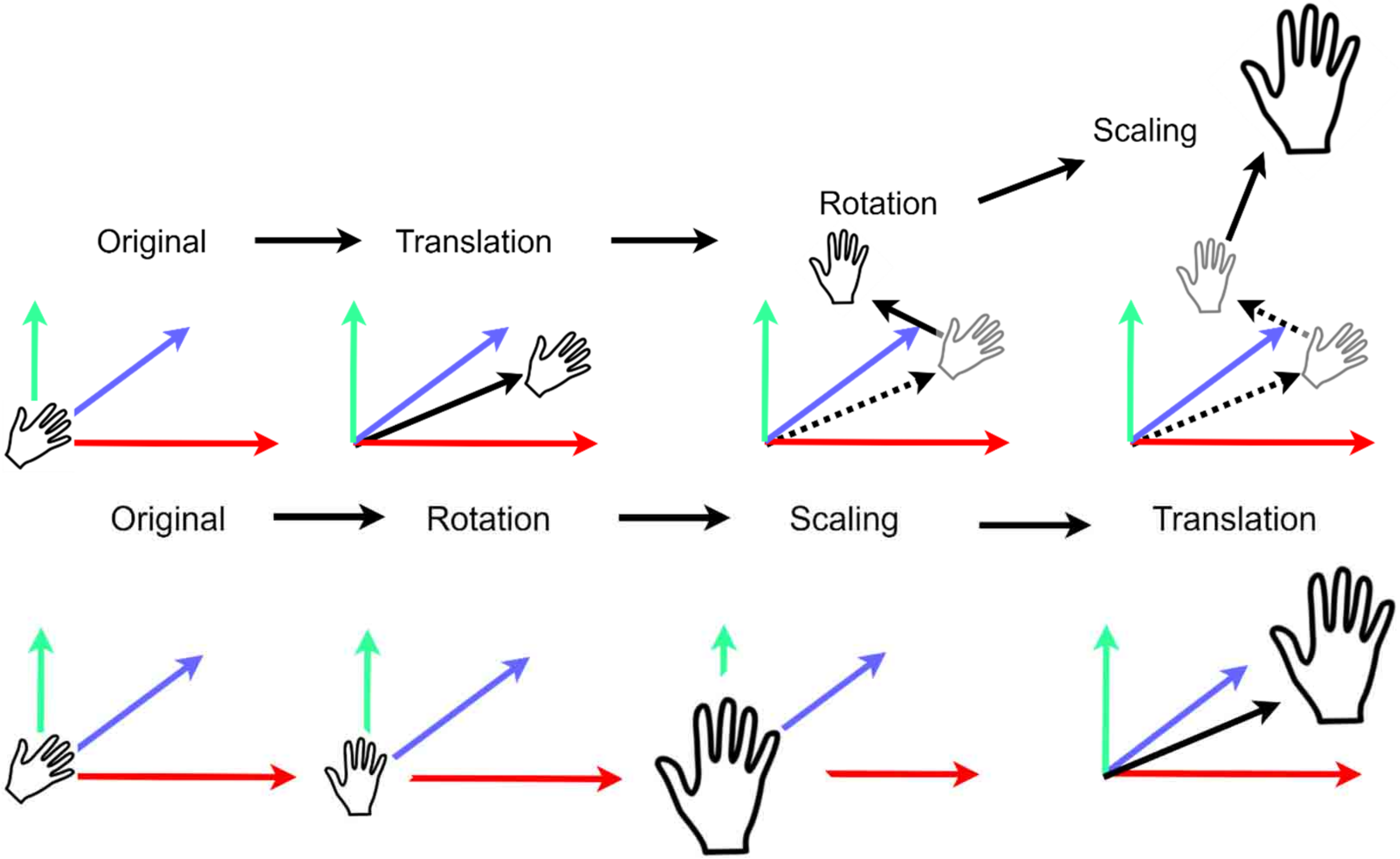
$\overline{P}(x, y, z)$ Perspective transform with each axis by (X, Y, Z) in space to $\overline{P^*}$

$$\mathbf{M}_{persepective} \cdot \overline{P} = \overline{P'} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X & Y & Z & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ X \cdot x + Y \cdot y + Z \cdot z + 1 \end{bmatrix}$$

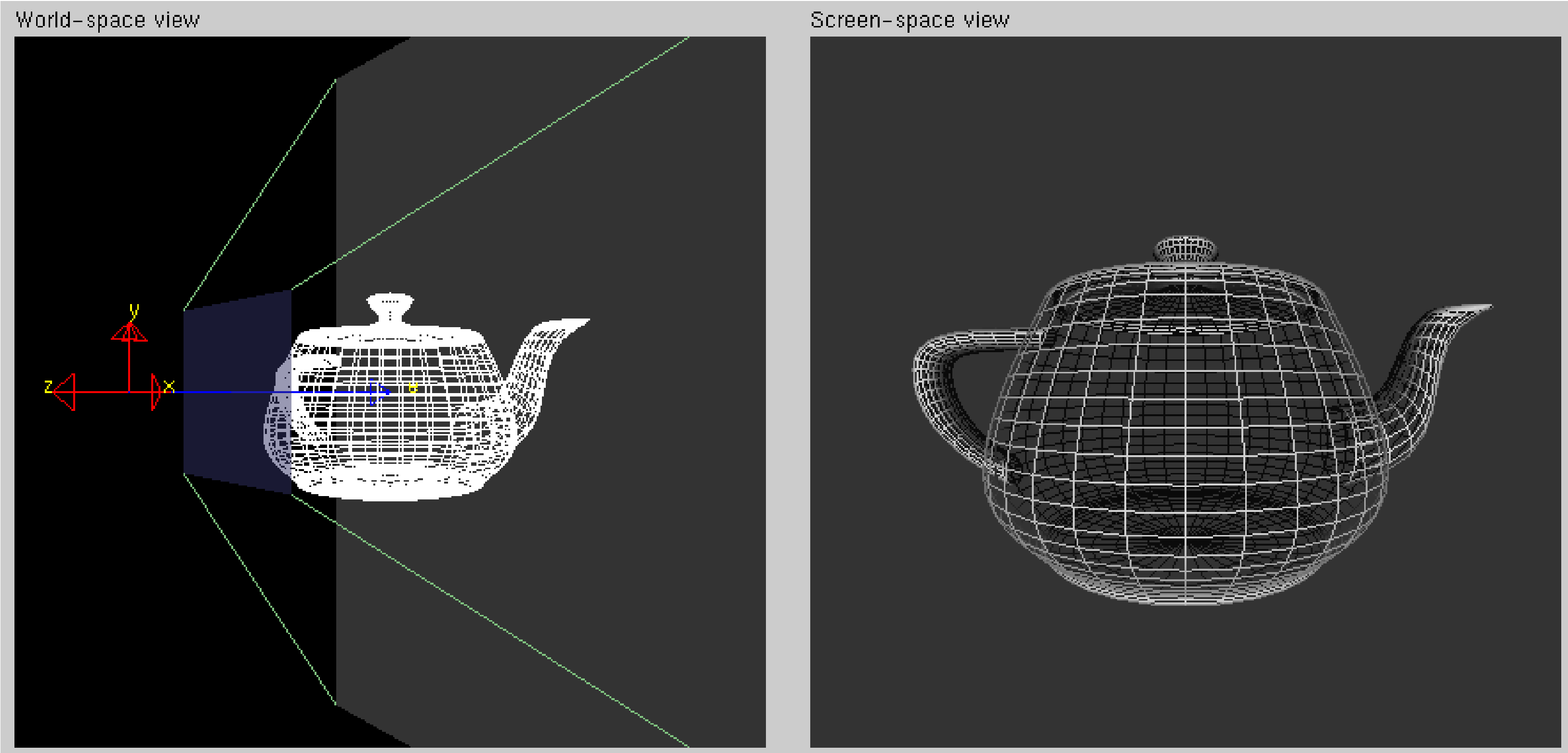
$$\text{Let } (X \cdot x + Y \cdot y + Z \cdot z) = r, \overline{P'}: \begin{bmatrix} x \\ y \\ z \\ r + 1 \end{bmatrix} = \overline{P^*}: \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \\ r + 1 \end{bmatrix}$$



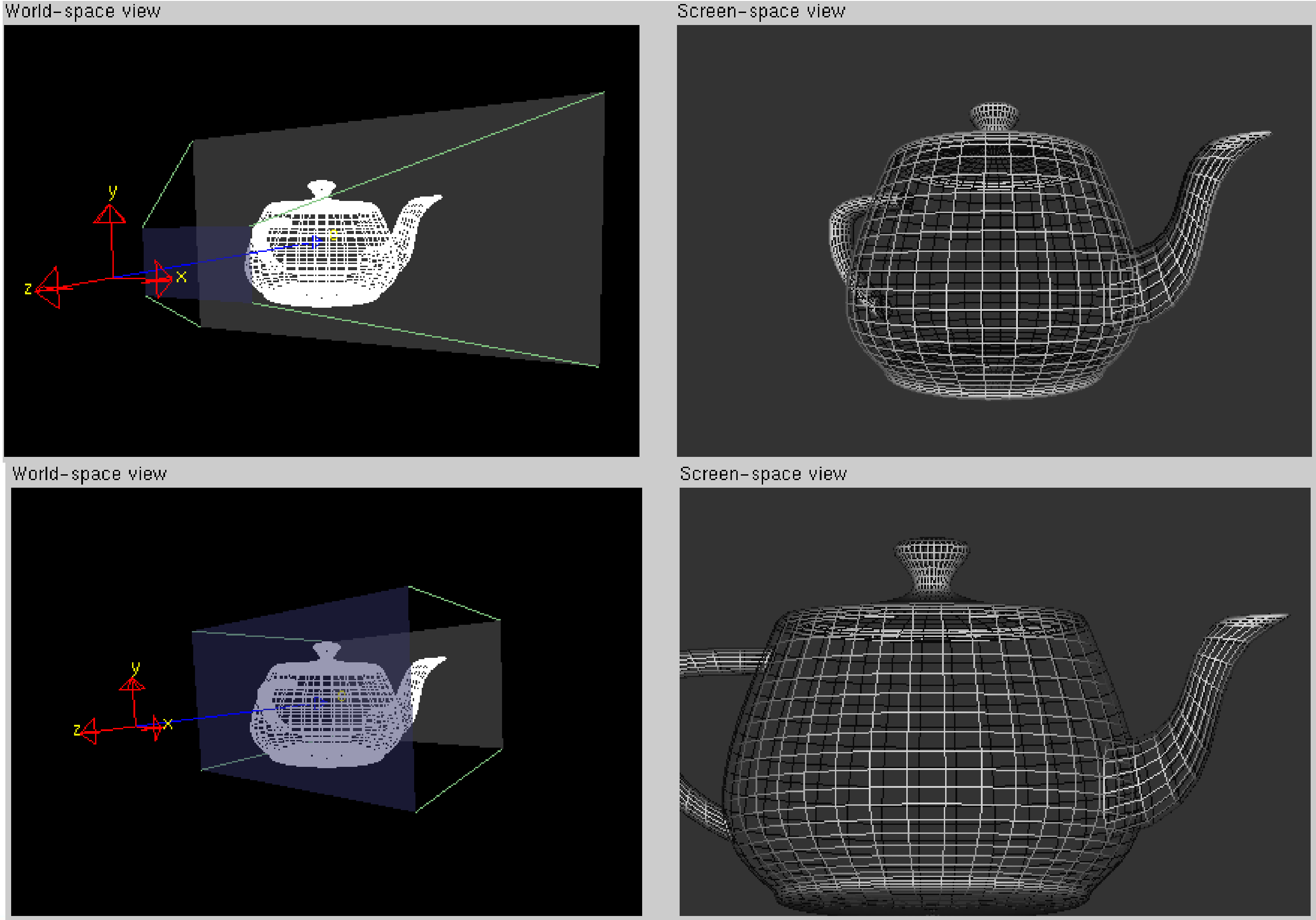
Different order in Transformation



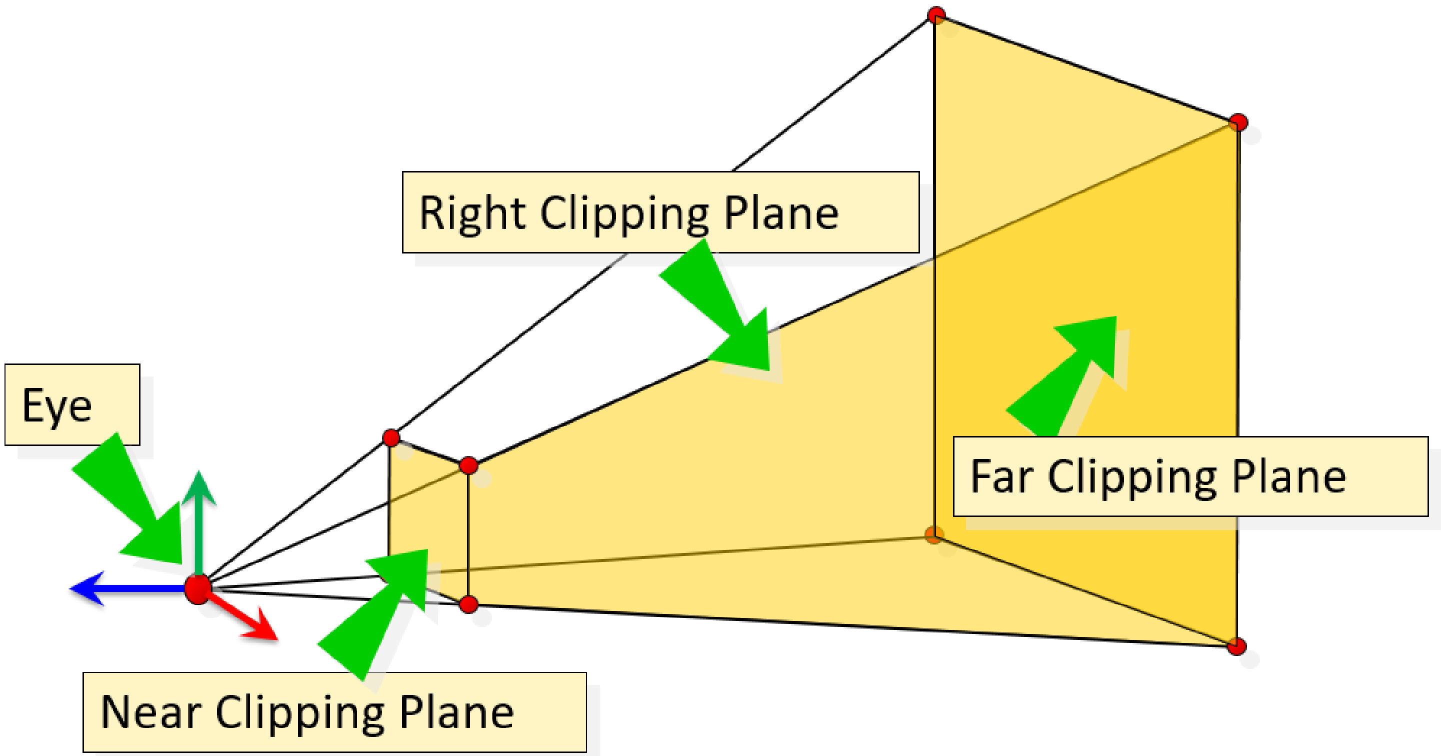
Camera System And Projection



Camera system

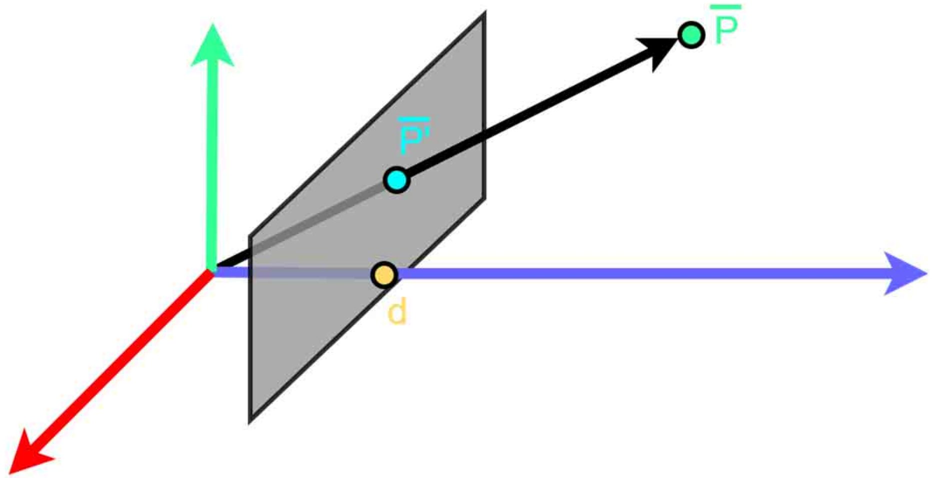


Perspective Projection



Perspective Projection

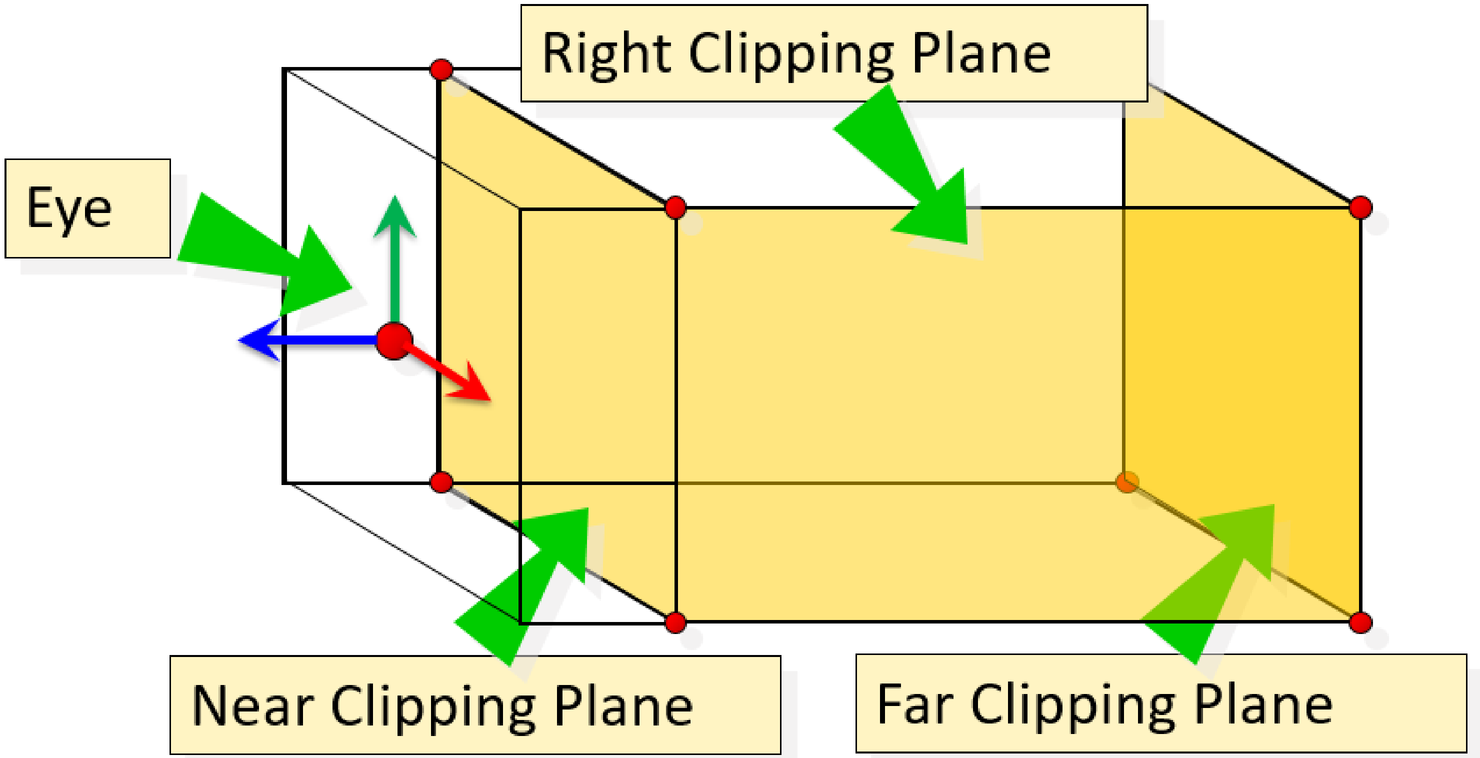
$\overline{\mathrm{P}}(\mathrm{x,y,z})$ Project to the plane$(\mathrm{z=d})$ to $\overline{\mathrm{P'}}$



$$\mathbf{M}_{\mathrm{persepective}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

$$\mathbf{M}_{\mathrm{persepective}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P'}} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} \frac{x \cdot d}{z} \\ \frac{y \cdot d}{z} \\ d \\ 1 \end{bmatrix}$$

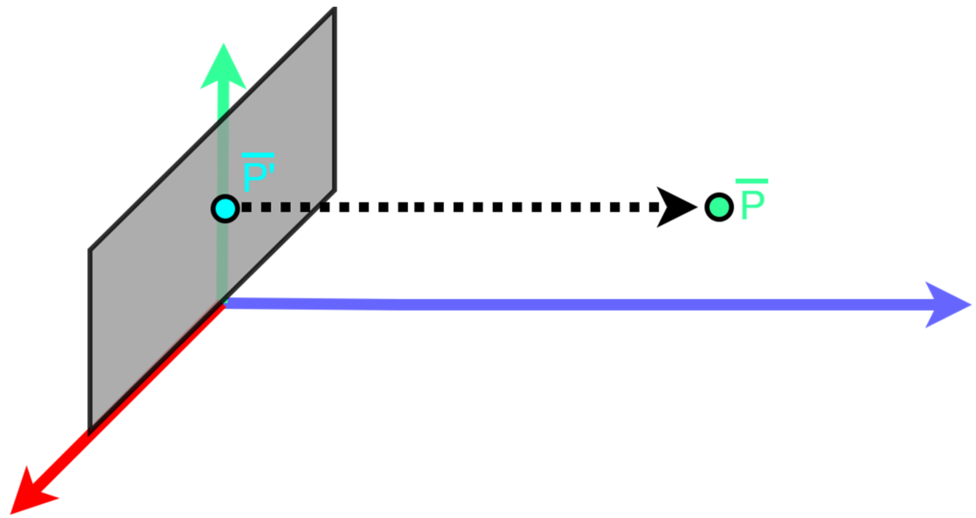
Orthographic Projection



Orthographic Projection

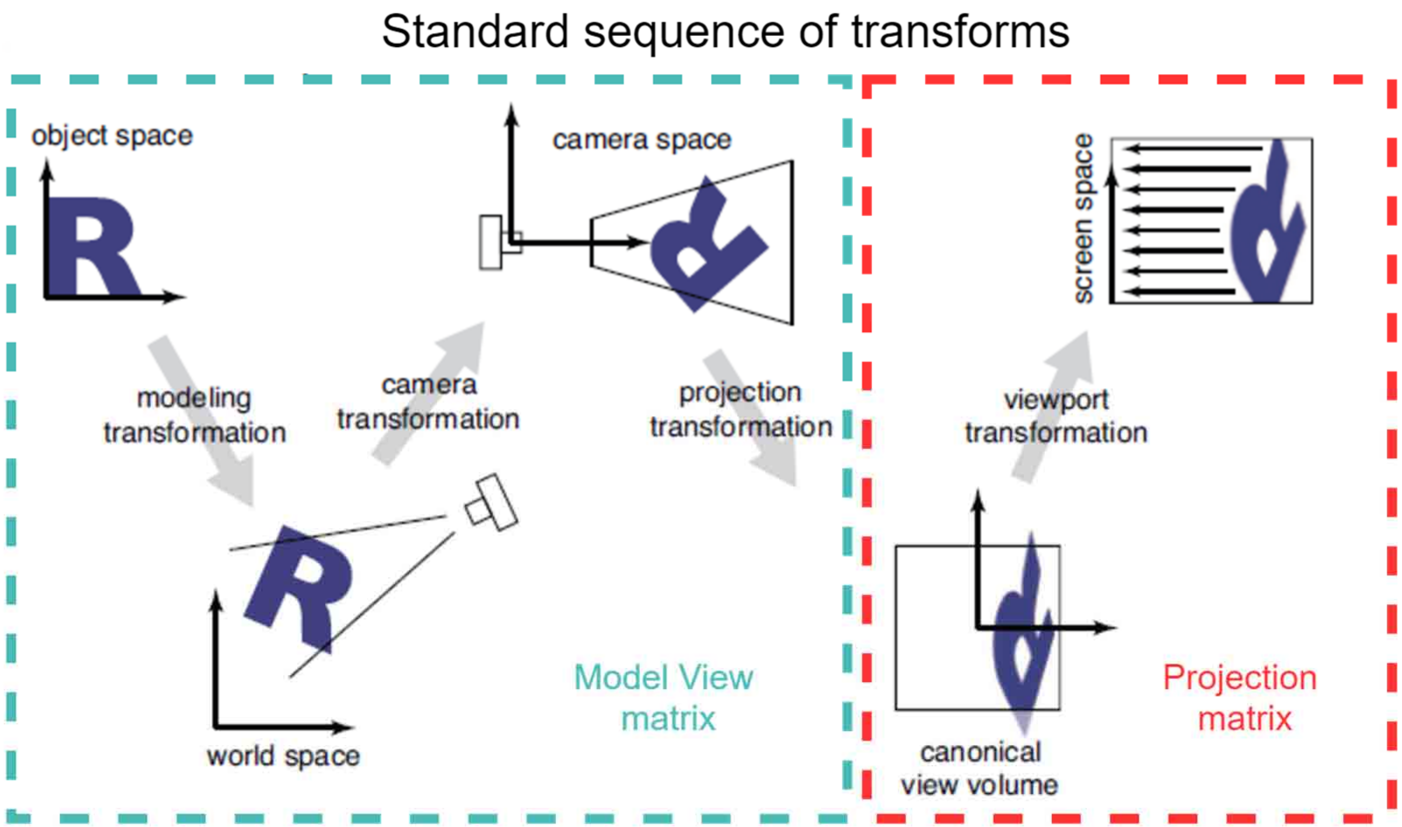
$\overline{\mathrm{P}}(\mathrm{x}, \mathrm{y}, \mathrm{z})$ Project to the plane$(\mathrm{z} = 0)$ to $\overline{\mathrm{P}'}$

$$\mathbf{M}_{\text{orthographic}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

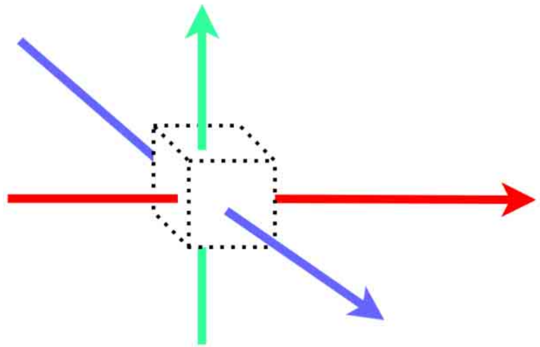


$$\mathbf{M}_{\text{orthographic}} \cdot \overline{\mathrm{P}} = \overline{\mathrm{P}'} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

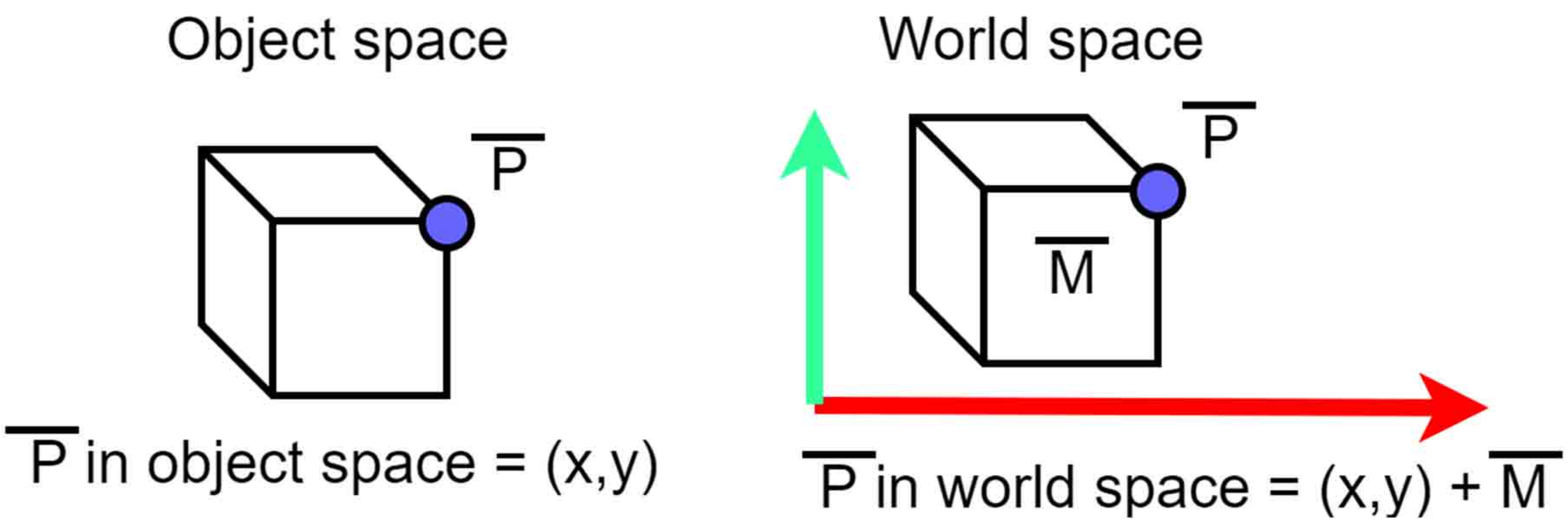
Coordinate System Relevant To Rendering



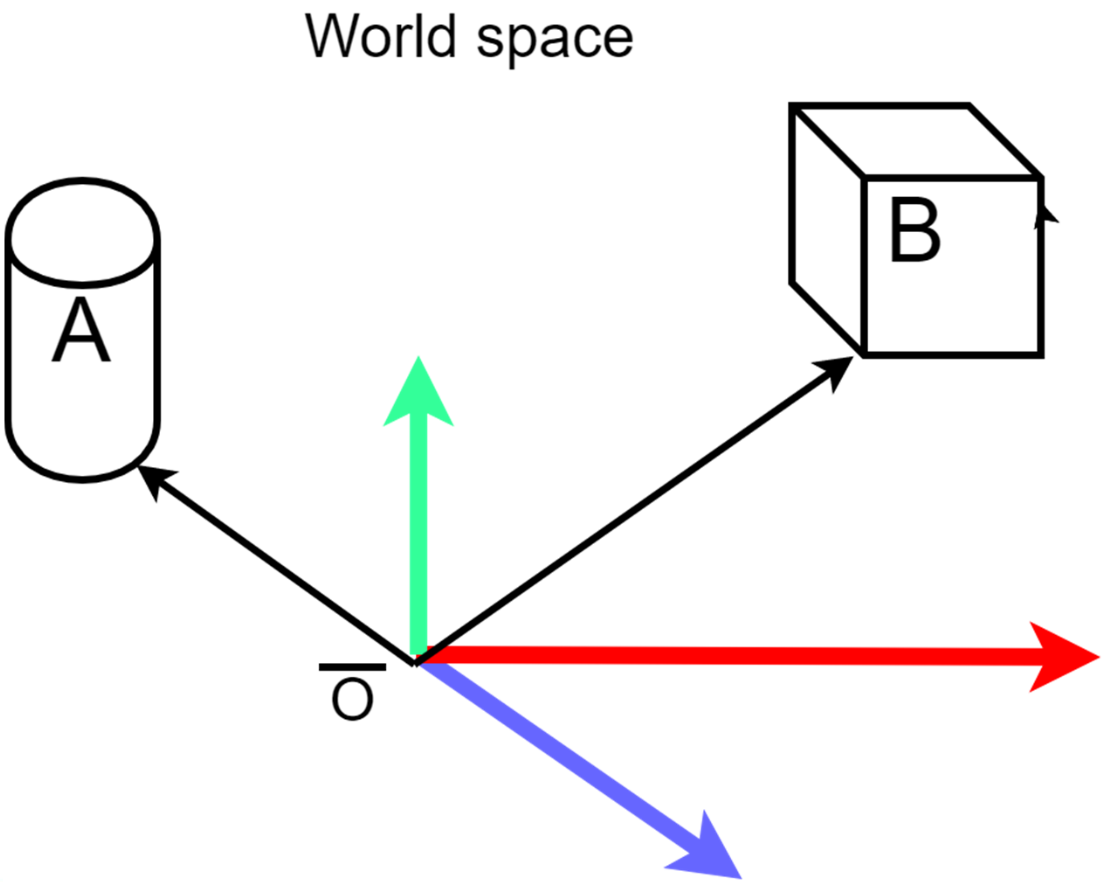
Object Space



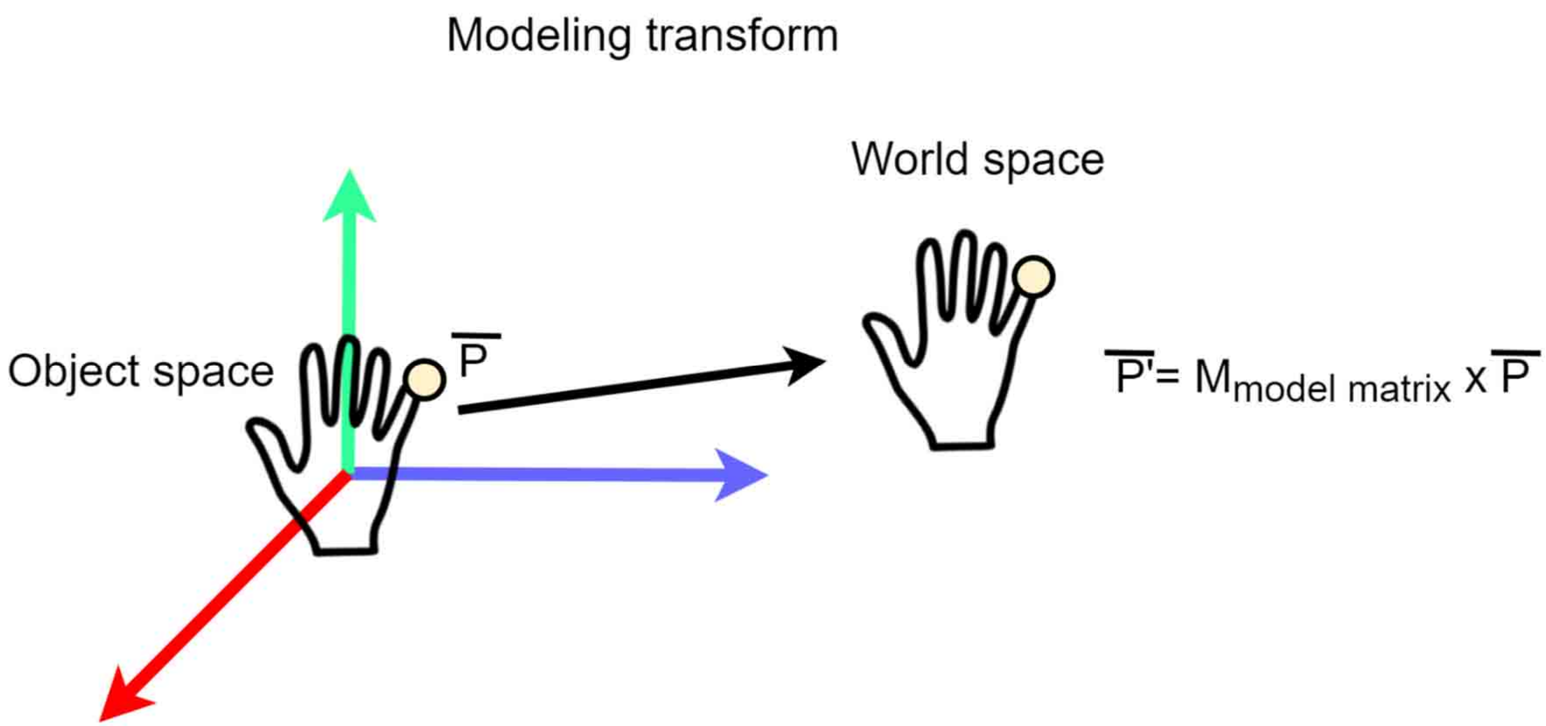
Object space



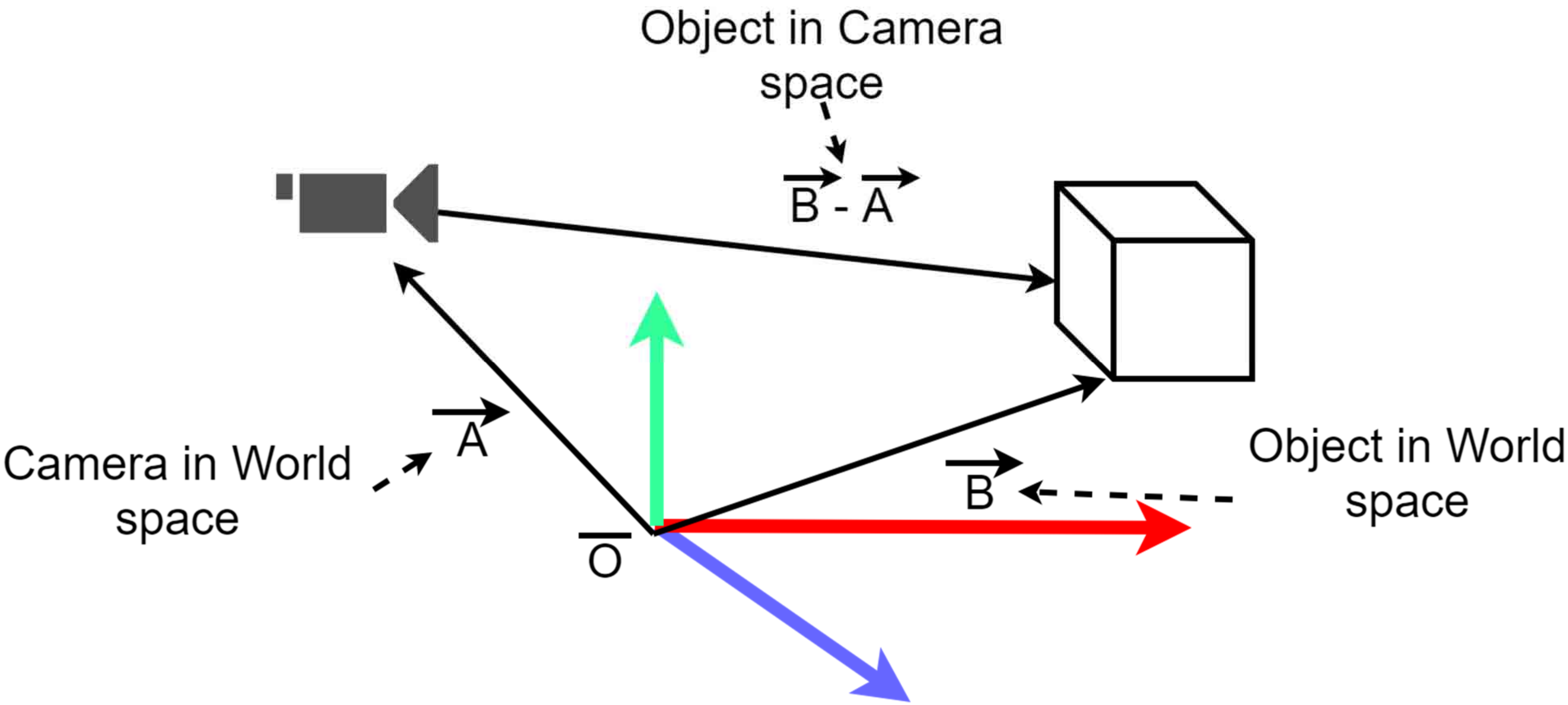
World Space



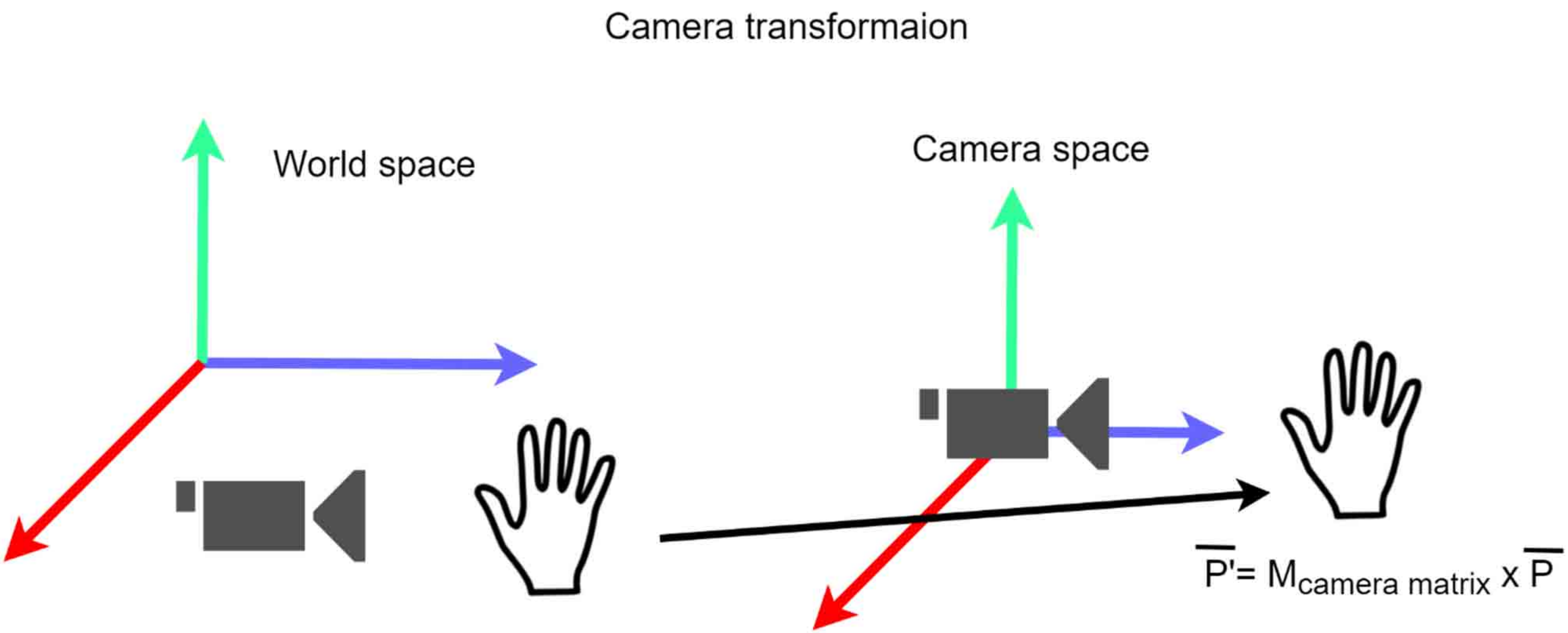
Object space to World space



Camera Space

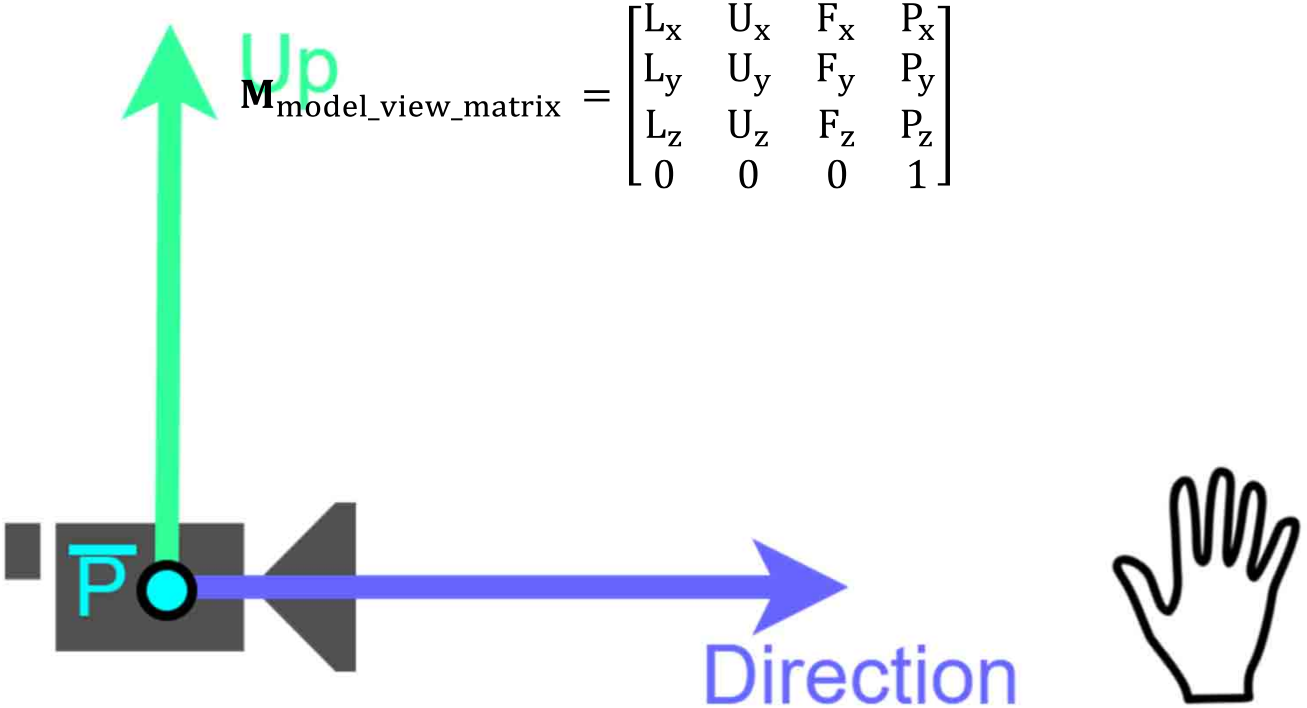


World space to Camera space

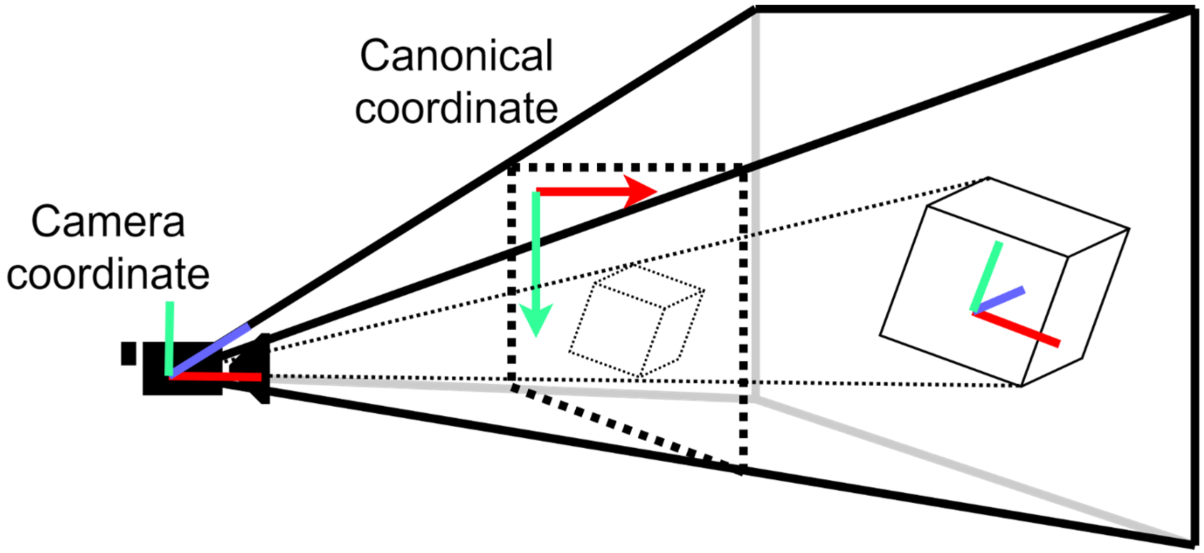


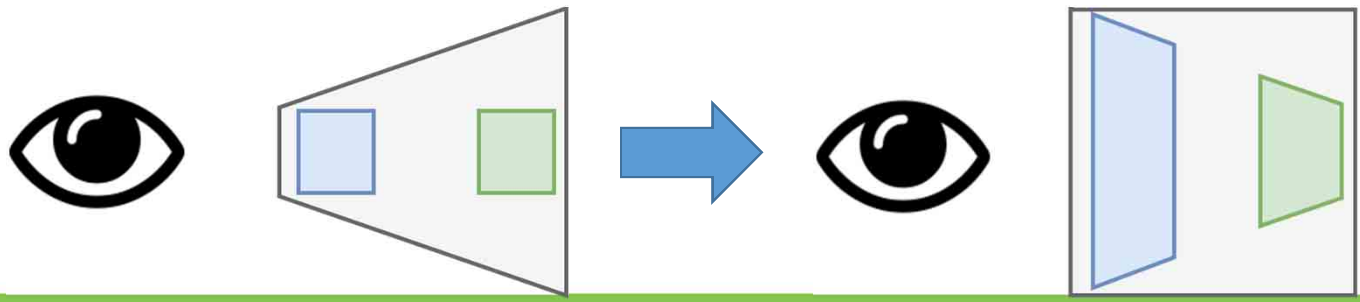
Camera matrix setting

$$\mathbf{M}_{\text{model\_view\_matrix}} = \begin{bmatrix} L_x & U_x & F_x & P_x \\ L_y & U_y & F_y & P_y \\ L_z & U_z & F_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

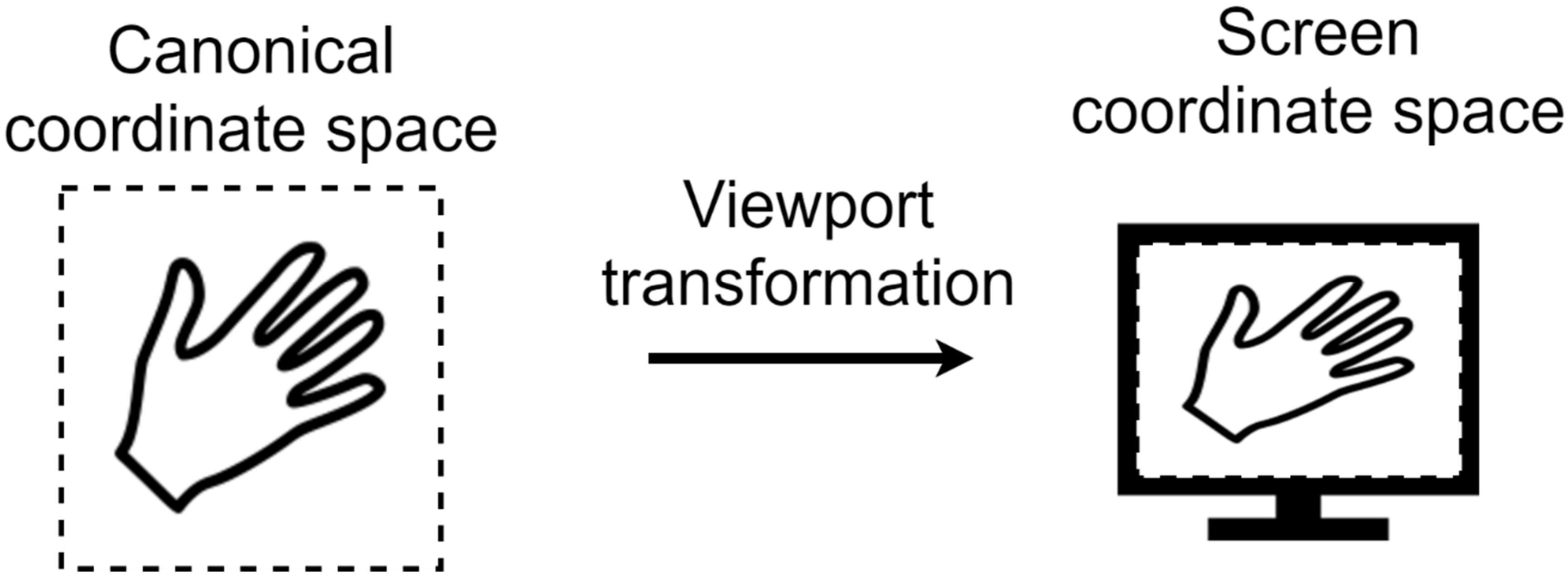


Canonical Space

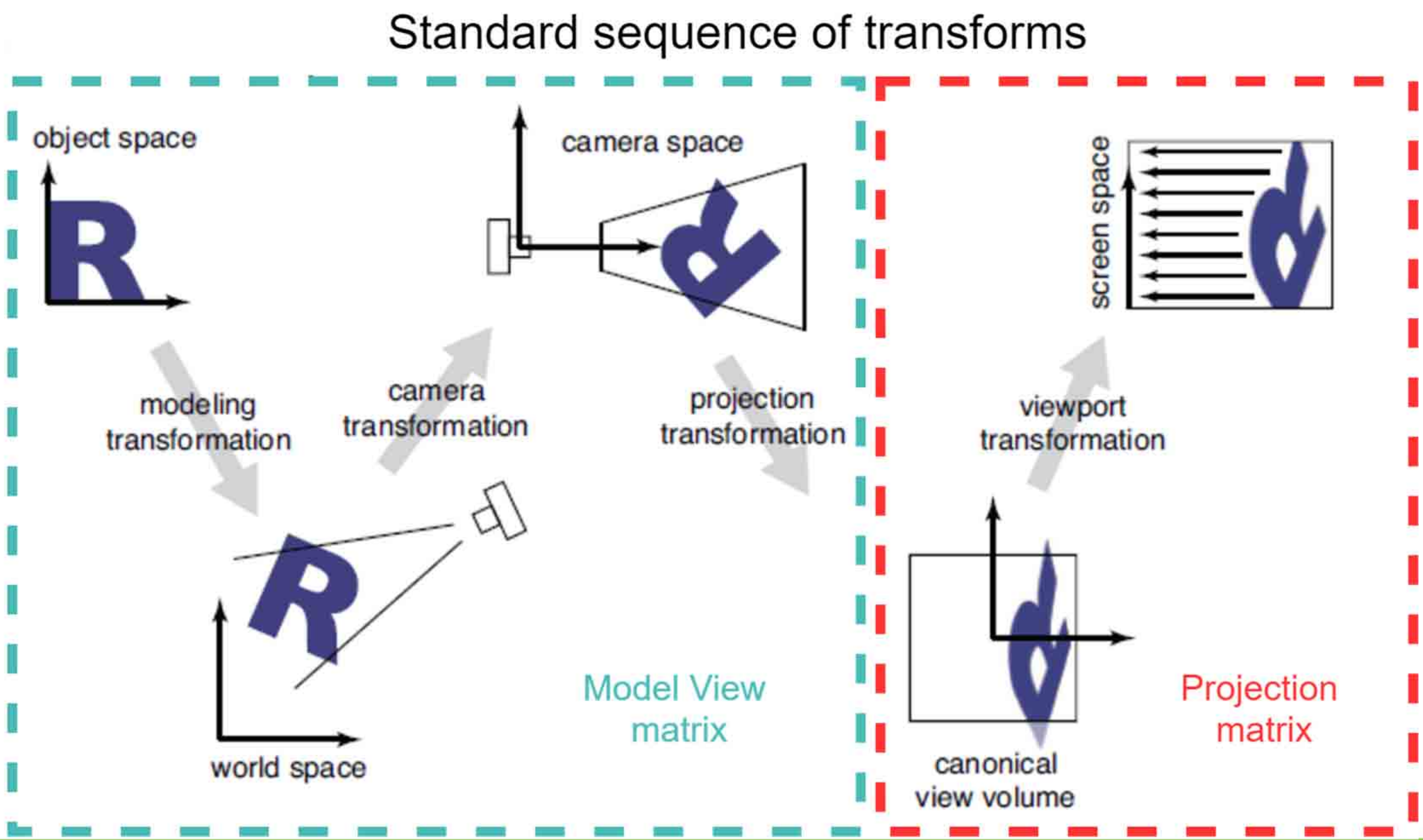




Screen Space



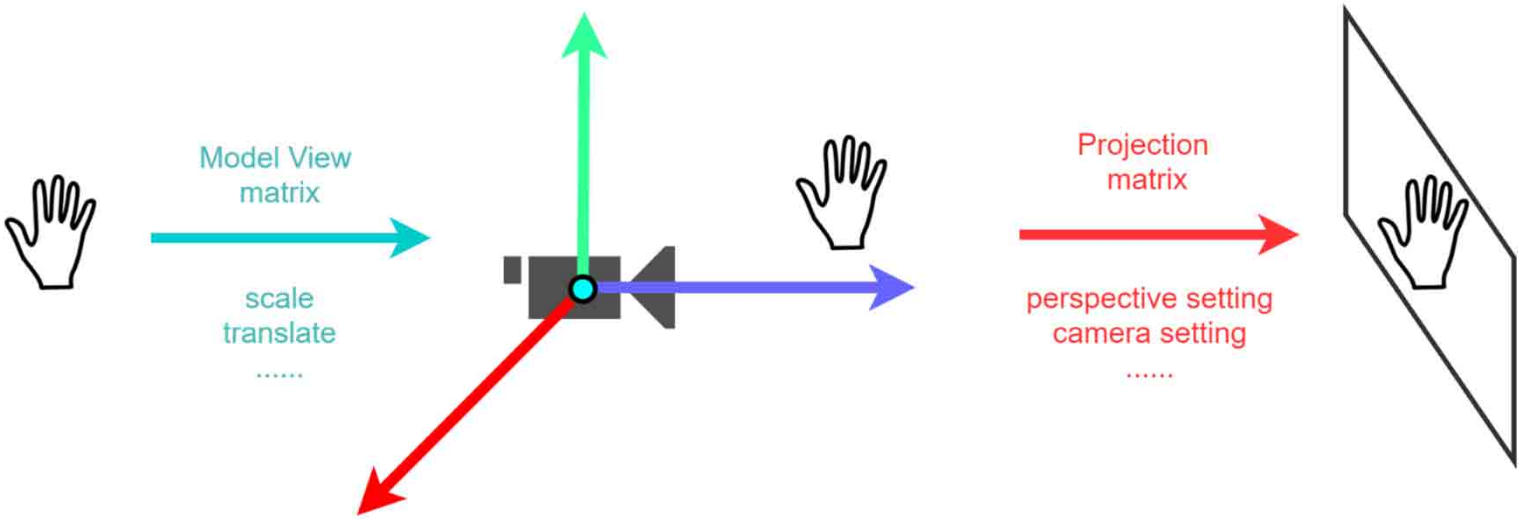
Sequence of Transforms



Model view matrix & Projection matrix

$$\mathbf{M}_{\mathrm{GL}_{\mathrm{ModelVIEW}}} = \mathbf{M}_{\mathrm{modeling}} \cdot \mathbf{M}_{\mathrm{transformation}}$$

$$\mathbf{M}_{\mathrm{GL\_PROJECTION}} = \mathbf{M}_{\mathrm{projection}} \cdot \mathbf{M}_{\mathrm{viewport}}$$



glMatrixMode

```
void glMatrixMode(GLenum mode);
```

- **`function`**: Specify which matrix is the current matrix.
- **`mode`**: Specifies which matrix stack is the target for subsequent matrix operations. There can accept three different values.
  - GL_MODELVIEW
  - GL_PROJECTION
  - GL_TEXTURE

glLoadIdentity

```
void glLoadIdentity(void);
```

- **function**: Replace the current matrix with the identity matrix.

glTranslatef

```
void glTranslatef(GLfloat x,  GLfloat y,  GLfloat z);
```

- **`function:`** Multiply the current matrix by a translation matrix.
- **`x, y, z`**: Specify the *x*, *y*, and *z* coordinates of a translation vector.

glRotatef

```
void glRotatef(Glfloat angle, GLfloat x,  GLfloat y,  GLfloat z);
```

- **`function`**: Multiply the current matrix by a rotation matrix.
- **`angle`**: Specifies the angle of rotation, in degrees.
- **`x, y, z`**: Specify the *x*, *y*, and *z* coordinates of a vector, respectively.

glViewport

```
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
```

- **`function`**: Set the viewport.
- **`x, y`**: Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0).
- **`width, height`**: Specify the width and height of the viewport. When a GL context is first attached to a window, width and height are set to the dimensions of that window.

gluPerspective

```
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear,
GLdouble zFar);
```

- **`function`**: Set up a perspective projection matrix.
- **`fovy`**: Specifies the field of view angle, in degrees, in the *y* direction.
- **`aspect`**: Specifies the aspect ratio that determines the field of view in the *x* direction. The aspect ratio is the ratio of *x* (width) to *y* (height).
- **`znear`**: Specifies the distance from the viewer to the near clipping plane (always positive).
- **`zfar`**: Specifies the distance from the viewer to the far clipping plane (always positive).

glGetDoublev

```
void glGetDoublev(GLenum pname, GLfloat* params);
```

- **function**: return the value or values of a selected parameter.
- **pname**: Specifies the parameter value to be returned. The symbolic constants in the list below are accepted.
- **params**: Returns the value or values of the specified parameter.

Camera Example

GL_MODELVIEW Example

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -5.0);
glRotatef(-45.0, 0.0, 1.0, 0.0);
```

GL_PROJECTION Example

```
aspect = width * 1.0f / height;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glViewport(0, 0, width, height);
gluPerspective(60.0f, aspect, 0.1f, 10.0f);
glGetDoublev(GL_PROJECTION_MATRIX, projection);
```

Program: Projection control