Information Visualization

W05: Computer Graphics

Graduation School of System Informatics

Department of Computational Science

Naohisa Sakamoto, Akira Kageyama

Apr. 24, 2018

Errata

- w04_ex01.html

```
<script src="http://mrdoob.github.io/three.js/build/three.min.js"></script>
<script src="main.js"></script>
```

→

```
<script src="https://threejs.org/build/three.min.js"></script>
<script src="main.js"></script>
```

- Task submission

1. Upload **w04_ex01.html** and **main.js** to your github repository.
2. Send the URL
   https://xxx.github.io/InfoVis2018/yyy/w04_ex01.html
   xxx: user ID
   yyy: directory name

Schedule

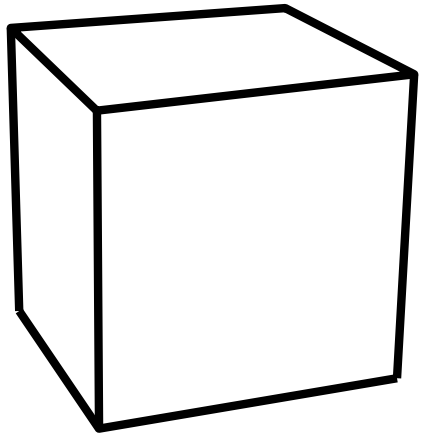
- W01 4/10 Guidance
- W02 4/11 Exercise (Setup)
- W03 4/17 Introduction to Data Visualization
- W04 4/18 Exercise (JavaScript Programming)
- W05 4/24 Computer Graphics
- W06 4/25 Exercise (Shader Programming)
- W07 5/01 Visualization Pipeline
- W08 5/02 Exercise (Data Model and Transfer Function)
- W09 5/08 Volume Visualization
- W10 5/09 Exercise (Isosurfaces and Volume Rendering)
- W11 5/22 Flow Visualization
- W12 5/23 Exercise (Streamlines and Line Integral Convolution)
- W13 5/29 Workshops 1
- W14 5/30 Workshops 2
- W15 6/05 Presentations

Table of Contents

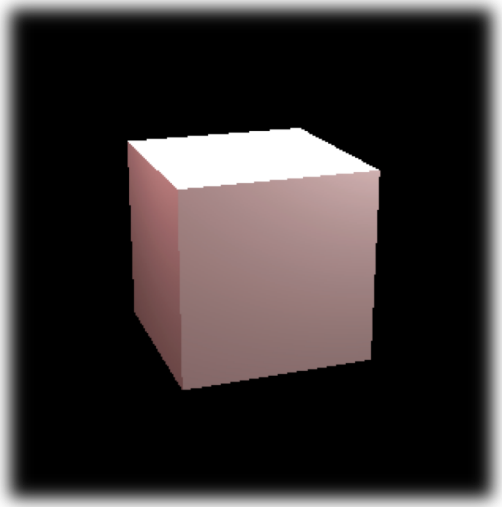
Table of Contents

Rendering Pipeline

- Rendering pipeline (Graphics pipeline)
  - Sequence of steps when rendering objects



3D model



Scene

Rendering Pipeline

- Processing steps on rendering pipeline

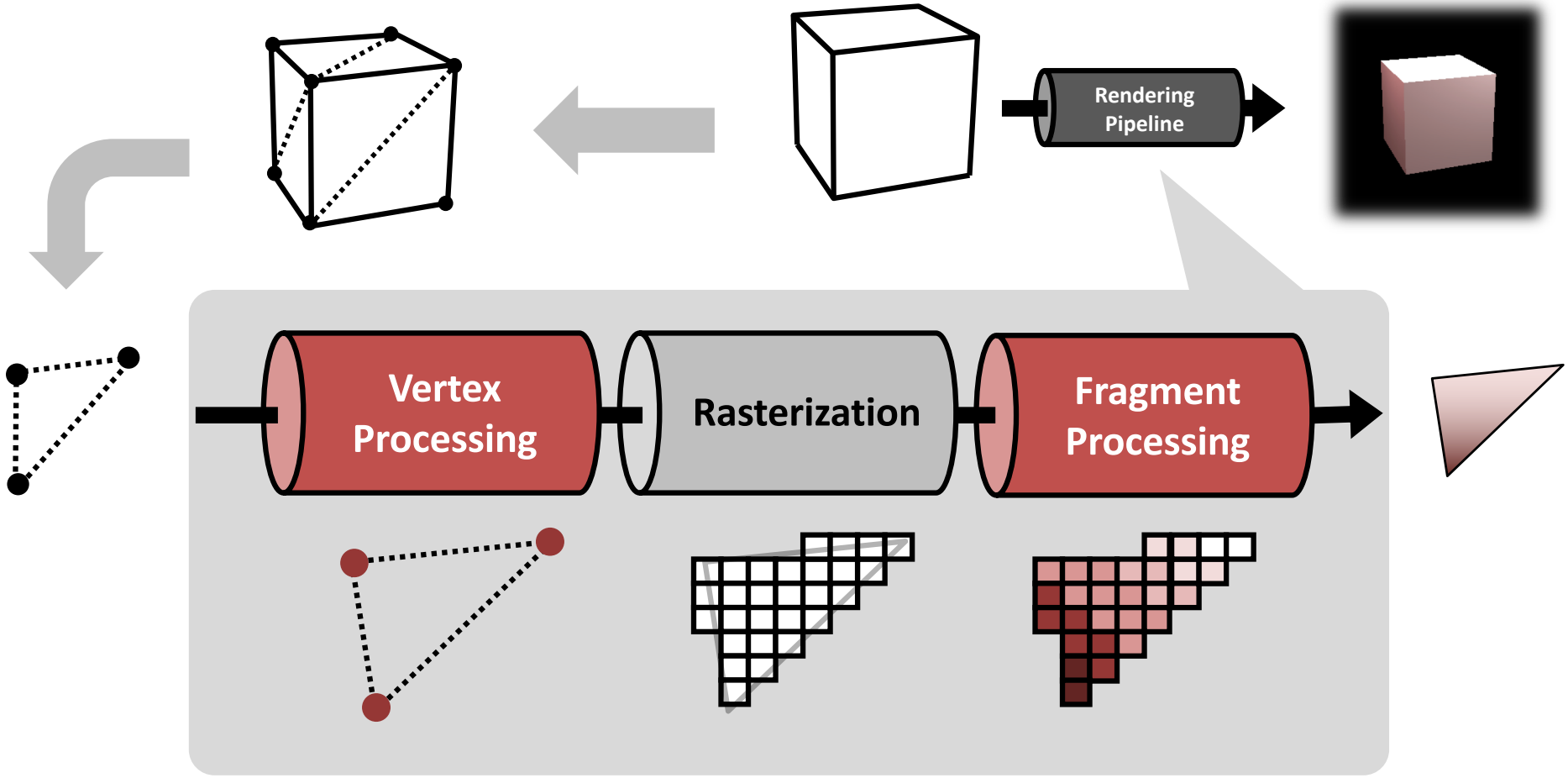


Table of Contents

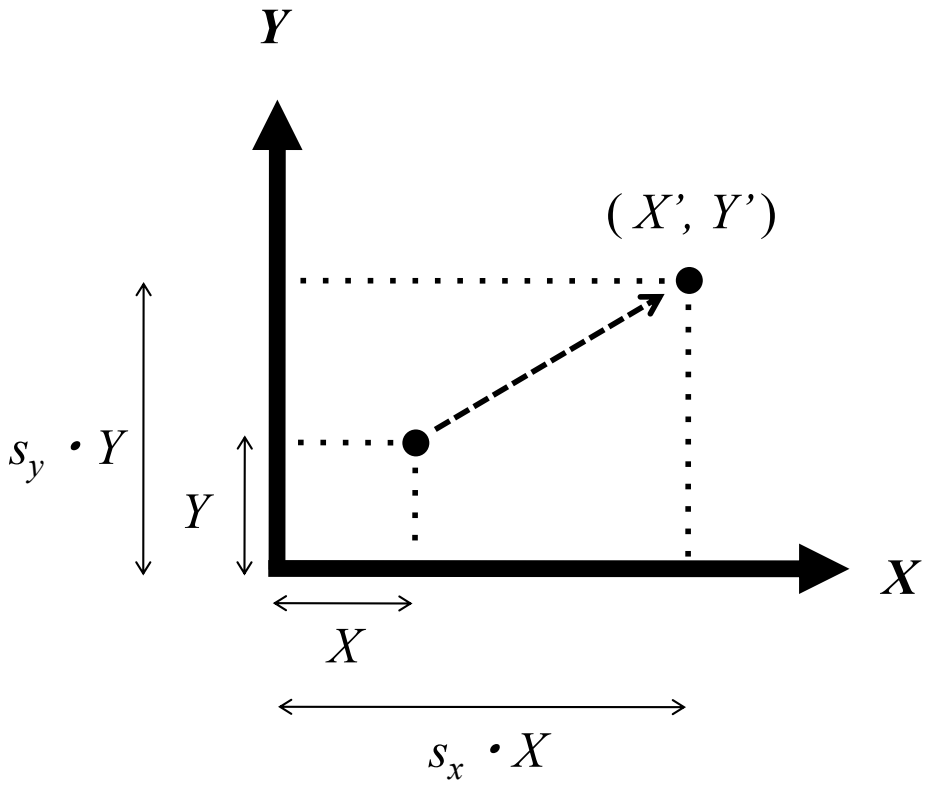
Geometric Transformations

- Geometric objects consist of a set of points and a point $\mathbf{P}$ is represented in Cartesian coordinates.
  - $\mathbf{P} = (X, Y)$
- Size, position, and orientation of objects can be changed by matrix operations.
  - Scaling
  - Translation
  - Rotation

Scaling

- Scaling an object from the point of origin by the factors sx and sy in x- and y-direction, respectively.

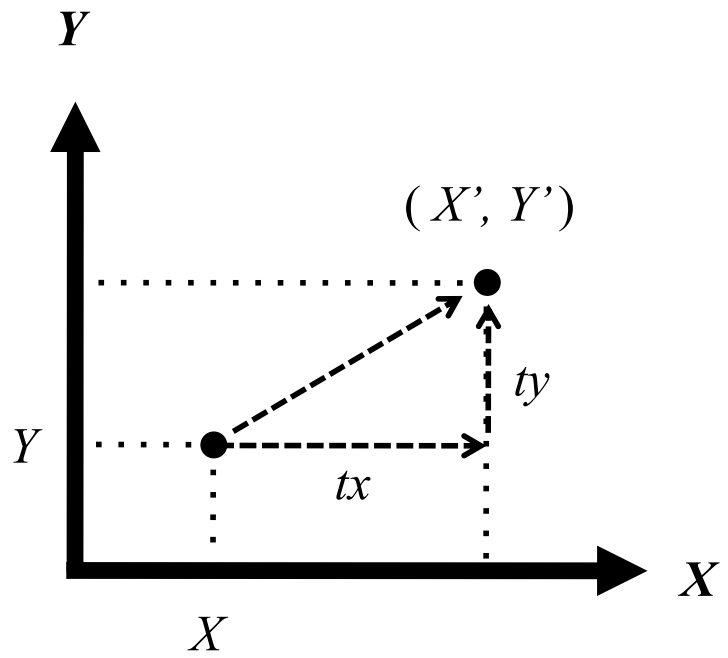
$$X' = sx \cdot X$$
$$Y' = sy \cdot Y$$



Translation

- Translation of a point $(X,\ Y)$ by the factors tx and ty in x- and y-direction.

$$X' = X + tx$$
$$Y' = Y + ty$$



Rotation

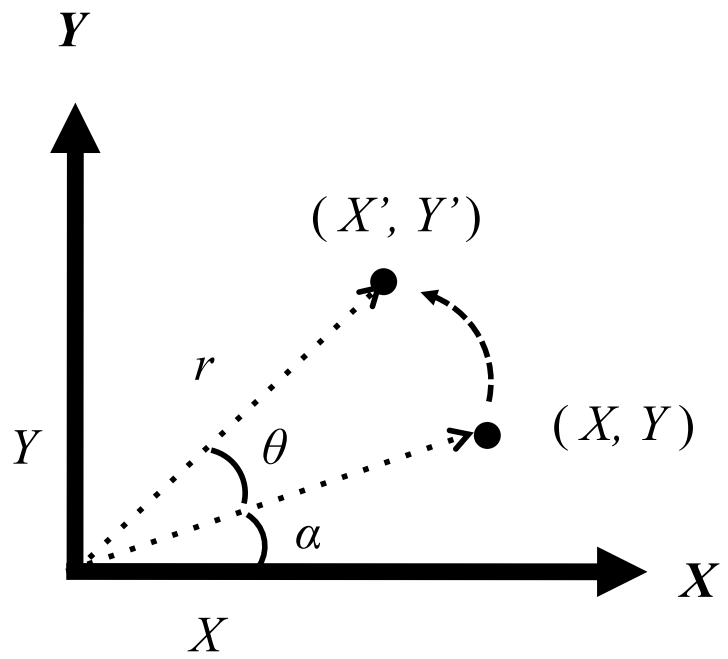
- Rotating a point $(X,\ Y)$ around the point of origin by an angle θ.

$$\begin{cases} X &= r\cos\alpha \\ Y &= r\sin\alpha \end{cases}$$

$$\begin{cases} X' &= r\cos(\alpha+\theta) \\ Y' &= r\sin(\alpha+\theta) \end{cases}$$

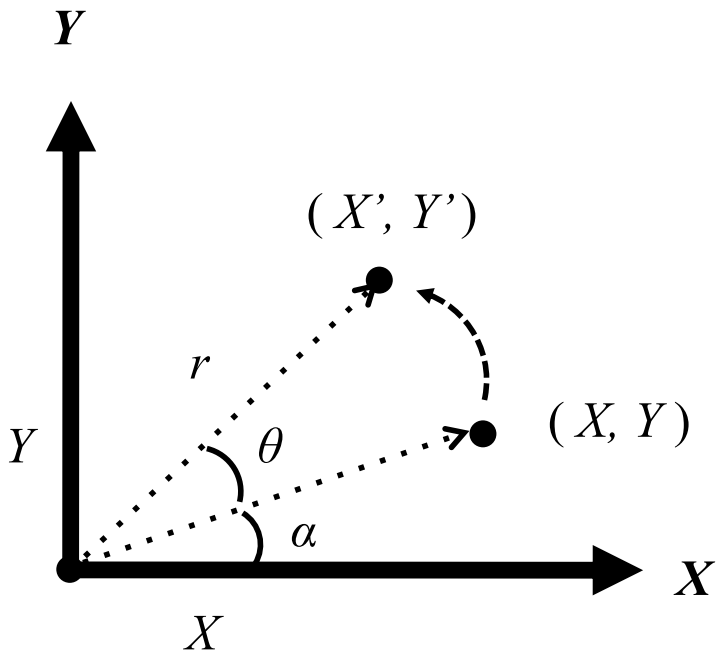
$$\Downarrow$$

$$\begin{aligned} X' &= \cos\theta\cdot X - \sin\theta\cdot Y \\ Y' &= \sin\theta\cdot X + \cos\theta\cdot Y \end{aligned}$$



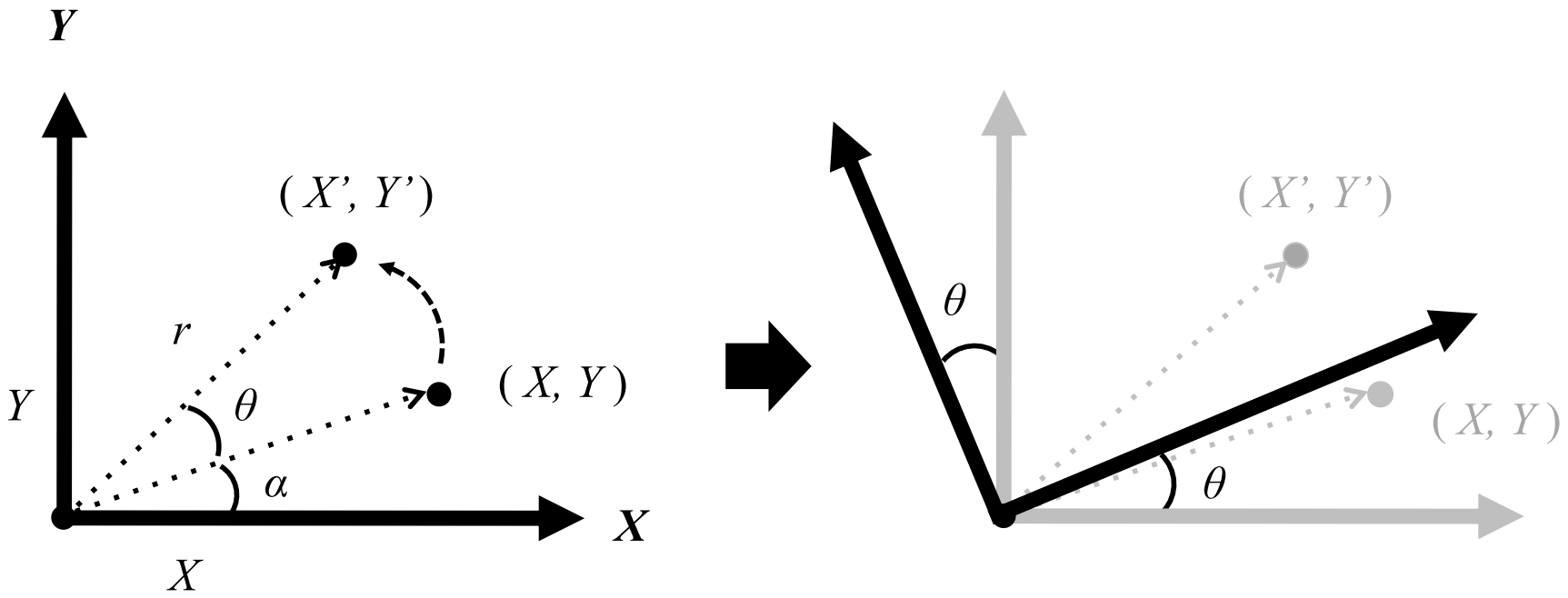
Rotation

- Rotation of X and Y axis.



Rotation

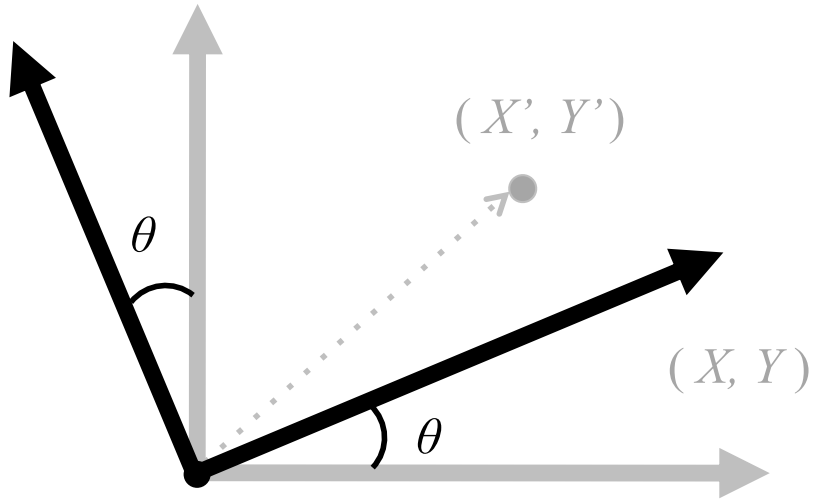
- Rotation of X and Y axis.



Rotation

- Rotation matrix
  - Column vectors represent unit direction vectors of the rotated axes.

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



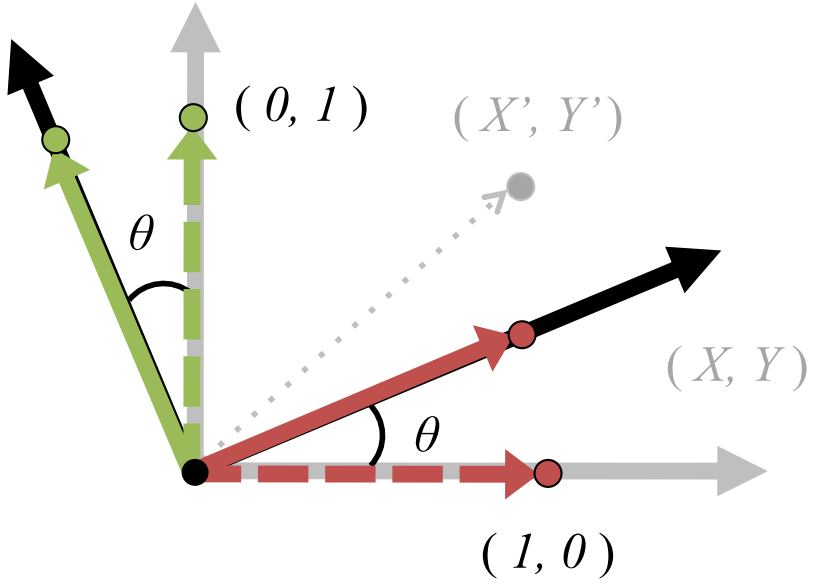
Rotation

- Rotation matrix
  - Column vectors represent unit direction vectors of the rotated axes.

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

Unit direction vector of the rotated x-axis

Unit direction vector of the rotated y-axis



Matrix Representations

- Scaling

$$\begin{aligned} X' &= sx \cdot X \\ Y' &= sy \cdot Y \end{aligned}$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- Translation

$$\begin{aligned} X' &= X + tx \\ Y' &= Y + ty \end{aligned}$$

?

- Rotation

$$\begin{aligned} X' &= \cos\theta \cdot X - \sin\theta \cdot Y \\ Y' &= \sin\theta \cdot X + \cos\theta \cdot Y \end{aligned}$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

Matrix Representations

- Scaling

$$\begin{aligned} X' &= sx \cdot X \\ Y' &= sy \cdot Y \end{aligned}$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

- Translation

$$\begin{aligned} X' &= X + tx \\ Y' &= Y + ty \end{aligned}$$

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- Rotation

$$\begin{aligned} X' &= \cos\theta \cdot X - \sin\theta \cdot Y \\ Y' &= \sin\theta \cdot X + \cos\theta \cdot Y \end{aligned}$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

Matrix Multiplication

- Transformations can be described as a matrix multiplication.
  - Homogeneous representation
  - Combined matrix $\boldsymbol{M}$ with scaling, translation and rotation matrices

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Homogeneous Representations

- Scaling

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- Translation

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- Rotation

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Ordering Transformations

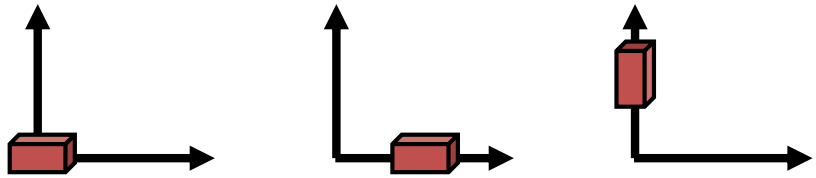
- Where is P'=(X',Y') transformed with a scaling matrix S, a translation matrix T and a rotation matrix R from P = (X,Y)?

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = STR \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Ordering Transformations

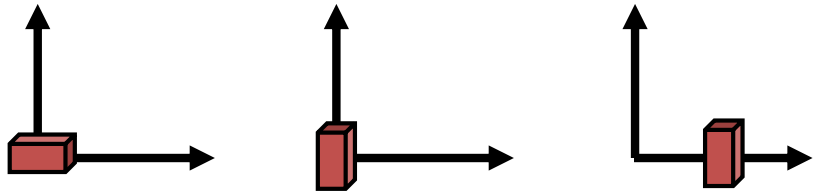
- Matrix multiplication is not cumulative.
  - Translation ($\boldsymbol{T}$) and then rotation ($\boldsymbol{R}$)

$$P' = RT\,P$$



  - Rotation ($\boldsymbol{R}$) and then translation ($\boldsymbol{T}$)

$$P' = TR\,P$$



Scaling

- Scaling a point (X, Y, Z) with (sx, sy, sz)

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

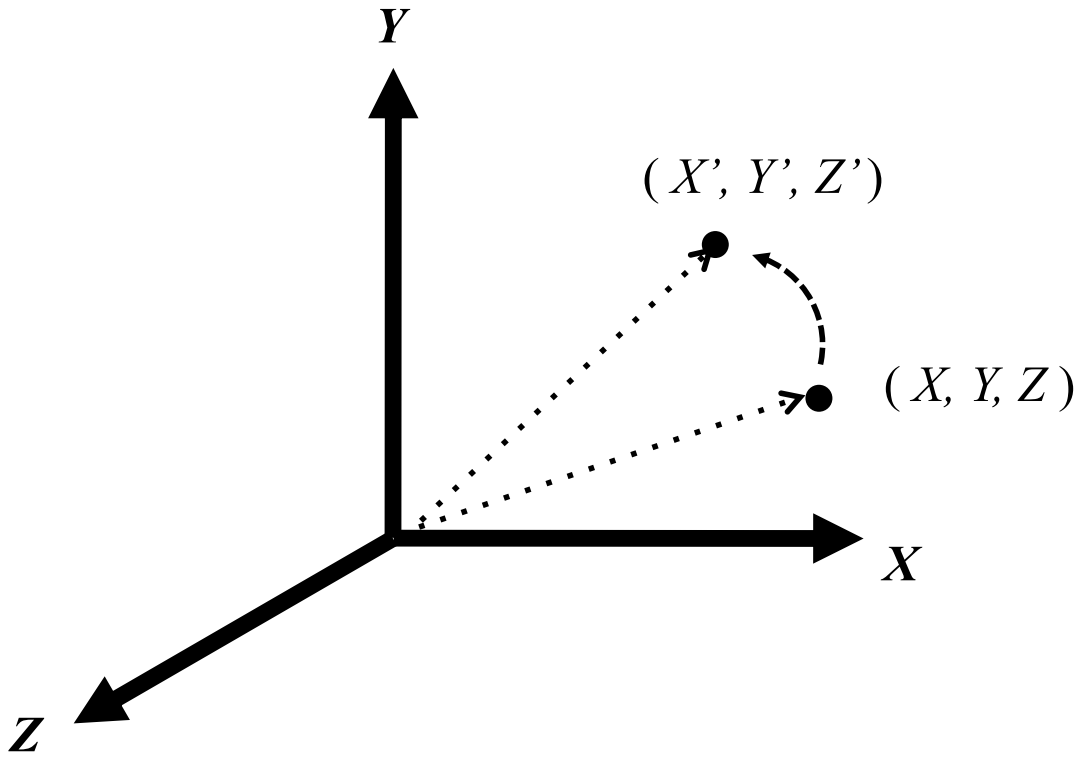
Translation

- Translation a point $(X,\ Y,\ Z)$ with $(tx,\ ty,\ tz)$

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

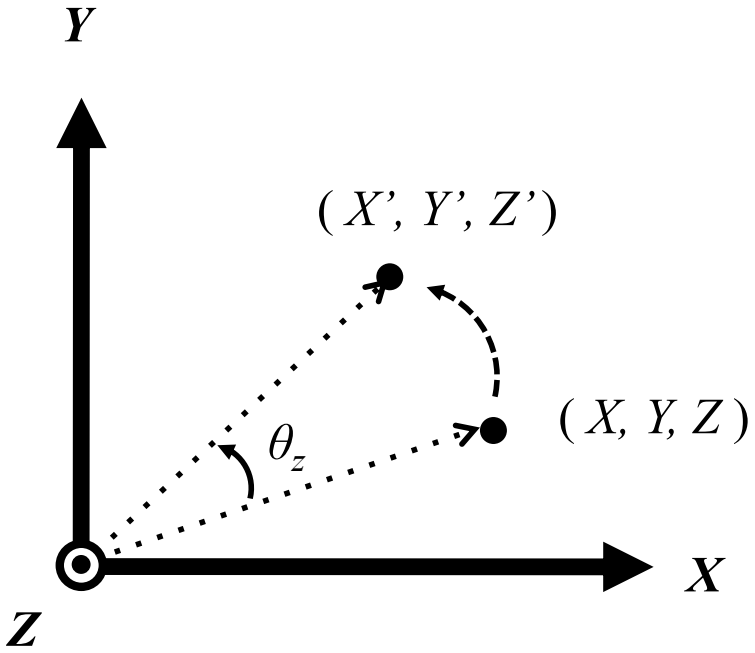
Rotation

- Rotation in 3D space



Rotation around z-axis

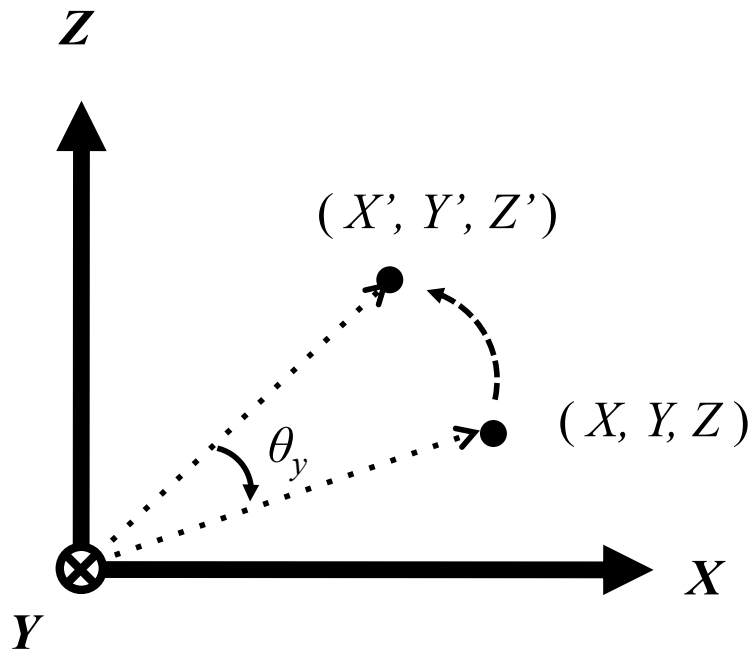
- Rotation around the z-axis by an angle θ_z



$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Rotation around y-axis

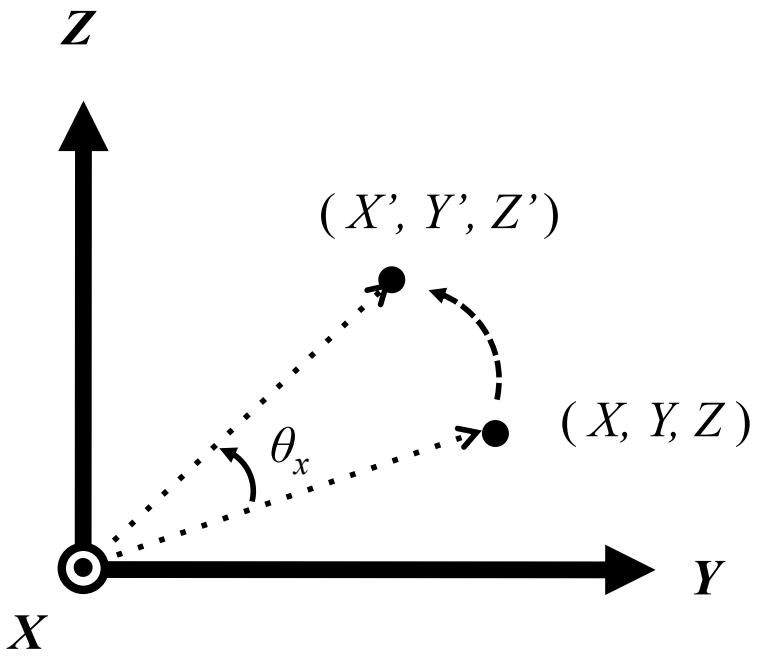
- Rotation around the y-axis by an angle θ_y



$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Rotation around x-axis

- Rotation around the x-axis by an angle θ_x



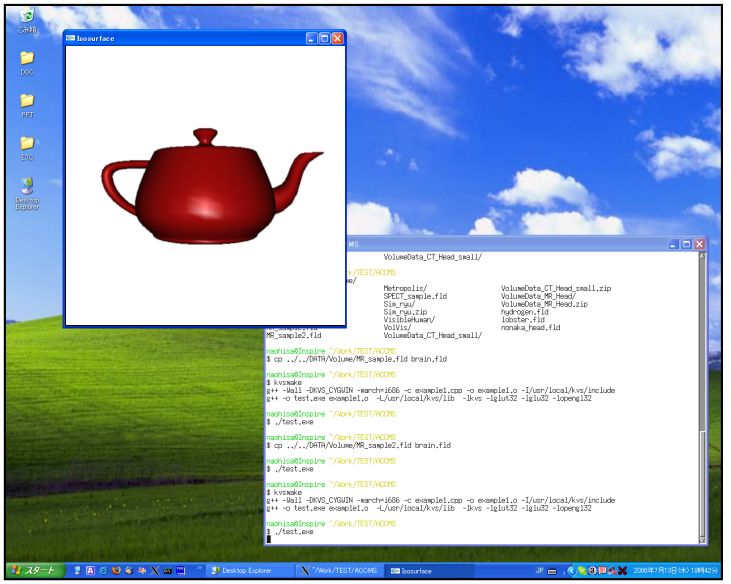
$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Table of Contents

Coordinate Transformations

- 3D rendering
  - Converting 3D objects into 2D images

Teapot (3D)



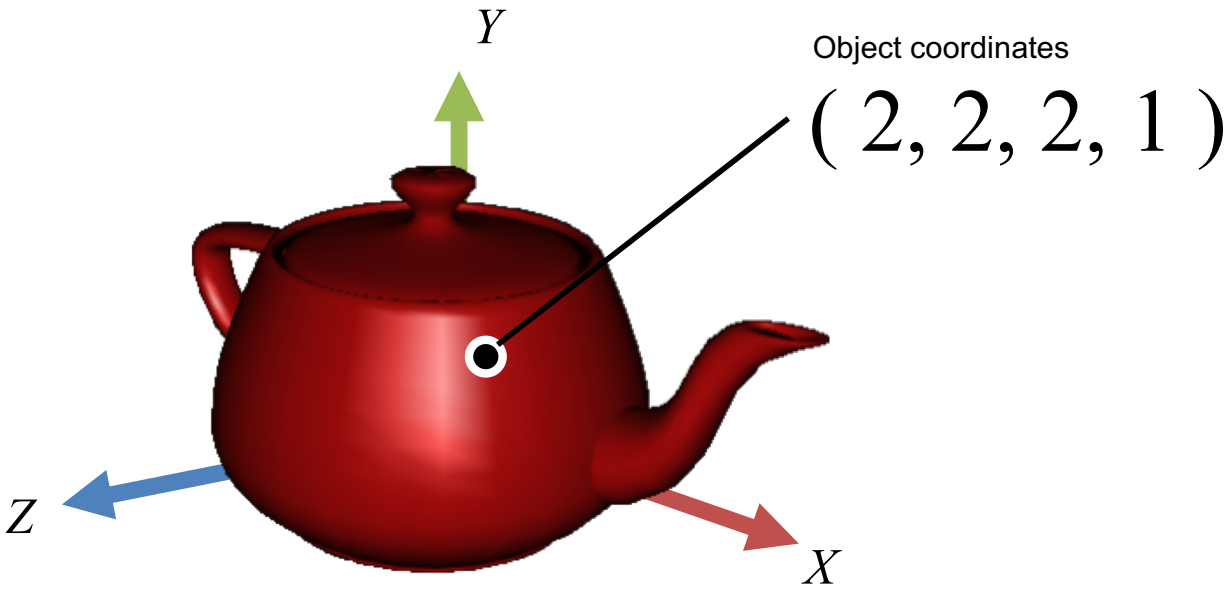
Desktop (2D)

Coordinate Systems

- In 3D computer graphics, objects are projected onto a image plane through several coordinate systems.
- Coordinate Systems
  1. Object Coordinate System
  2. World Coordinate System
  3. Camera Coordinate System
  4. Clip Coordinate System
  5. Normalized Device Coordinate System
  6. Window Coordinate System

Object Coordinate System

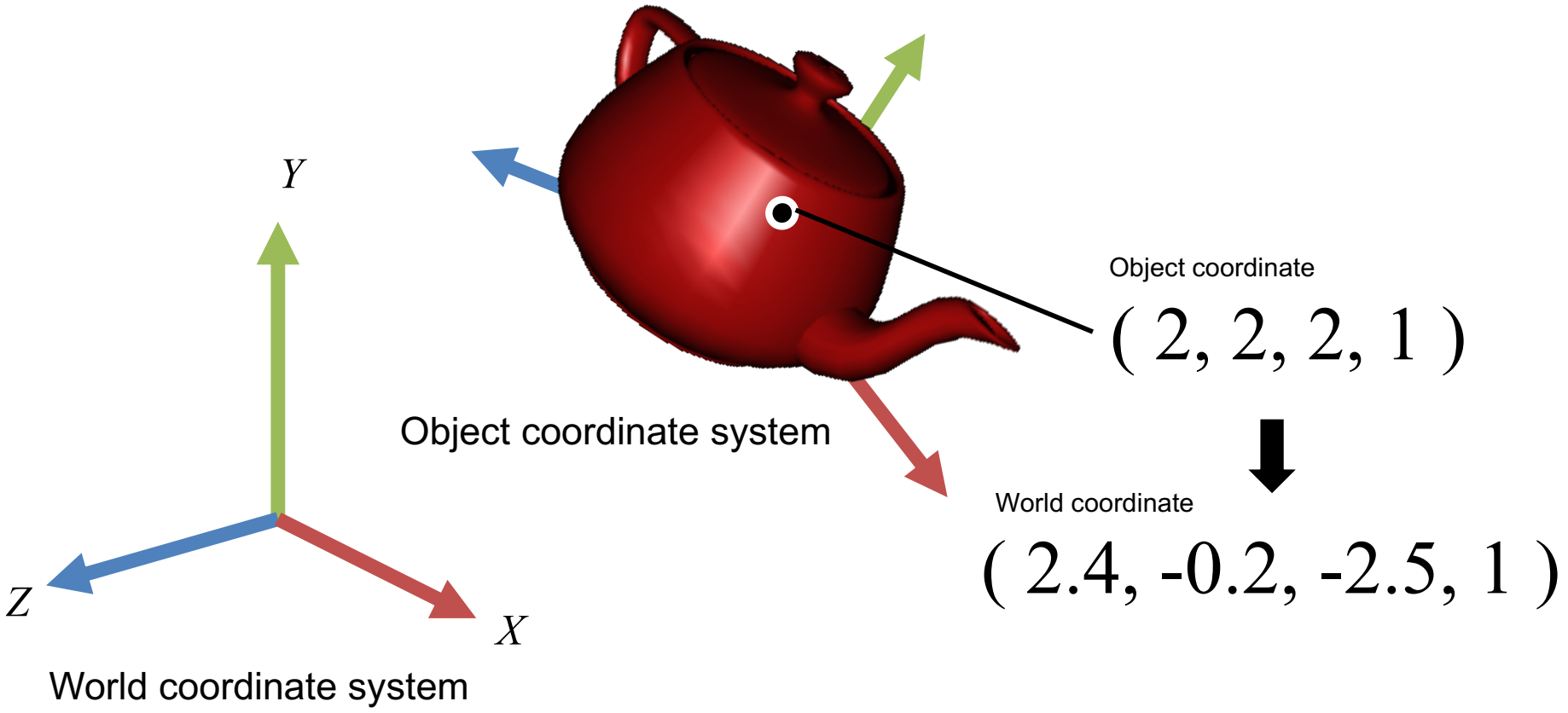
- 3D objects are often defined in an original local coordinate system.



Object coordinate system

World Coordinate System

- A basic reference coordinate system for all objects.



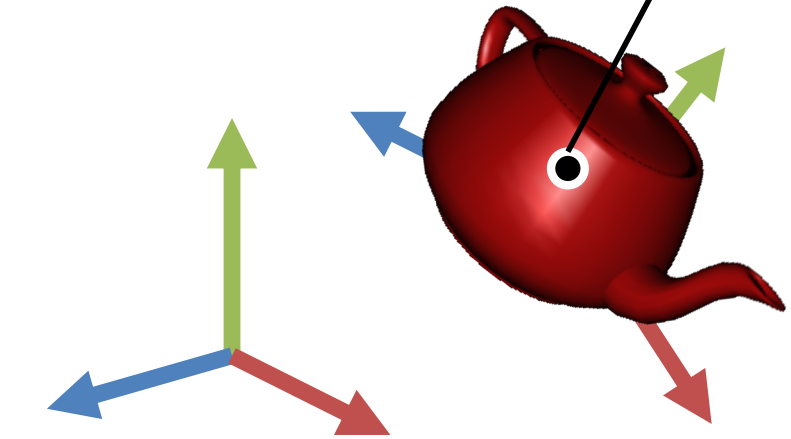
Camera Coordinate System

- A coordinate system on the basis of a camera for rendering the objects.

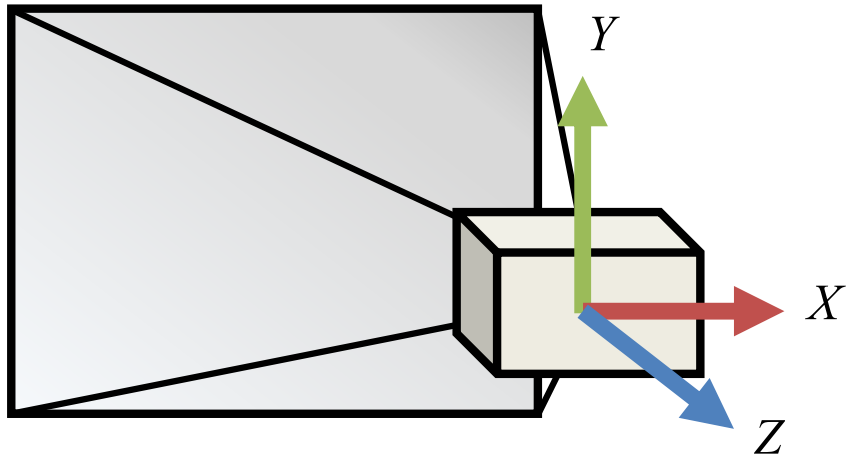
World coordinate system

(2.4, -0.2, -2.5, 1) ➡ (2.4, -0.2, -14.5, 1)

Camera coordinate system



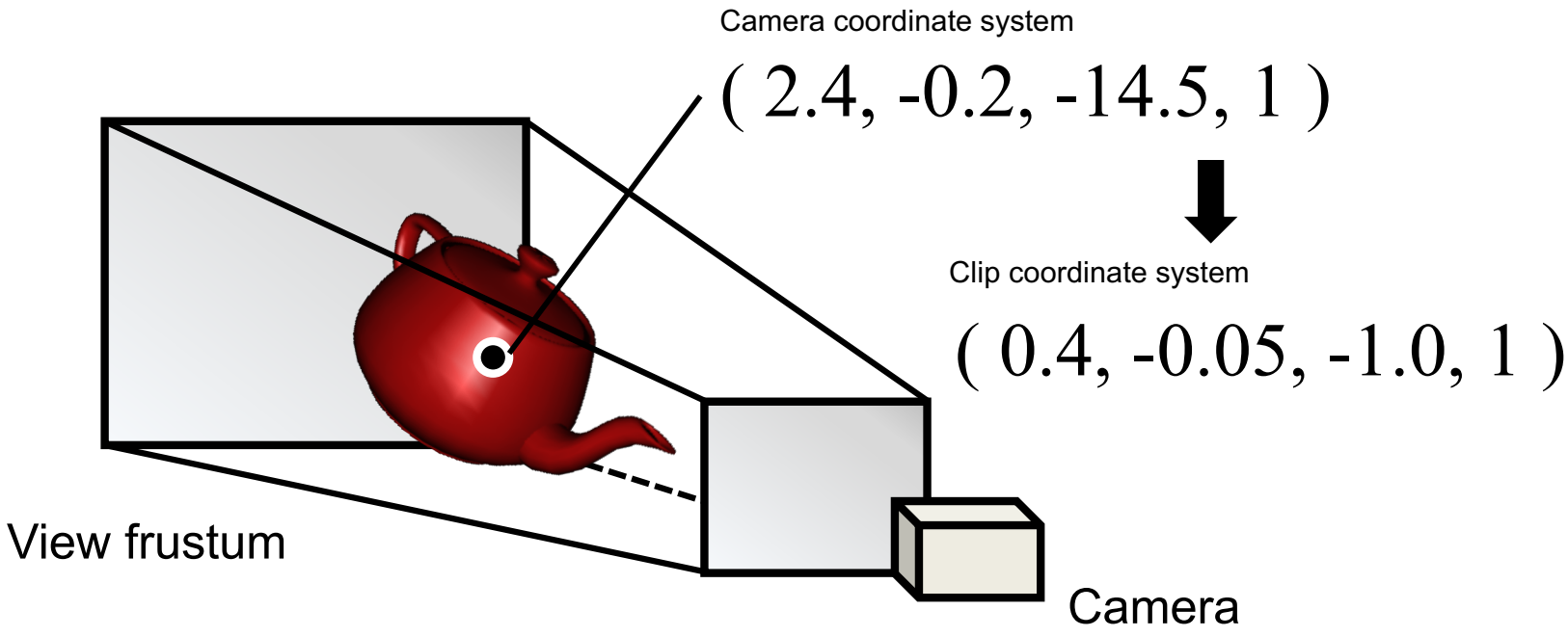
World coordinate system



Camera coordinate system

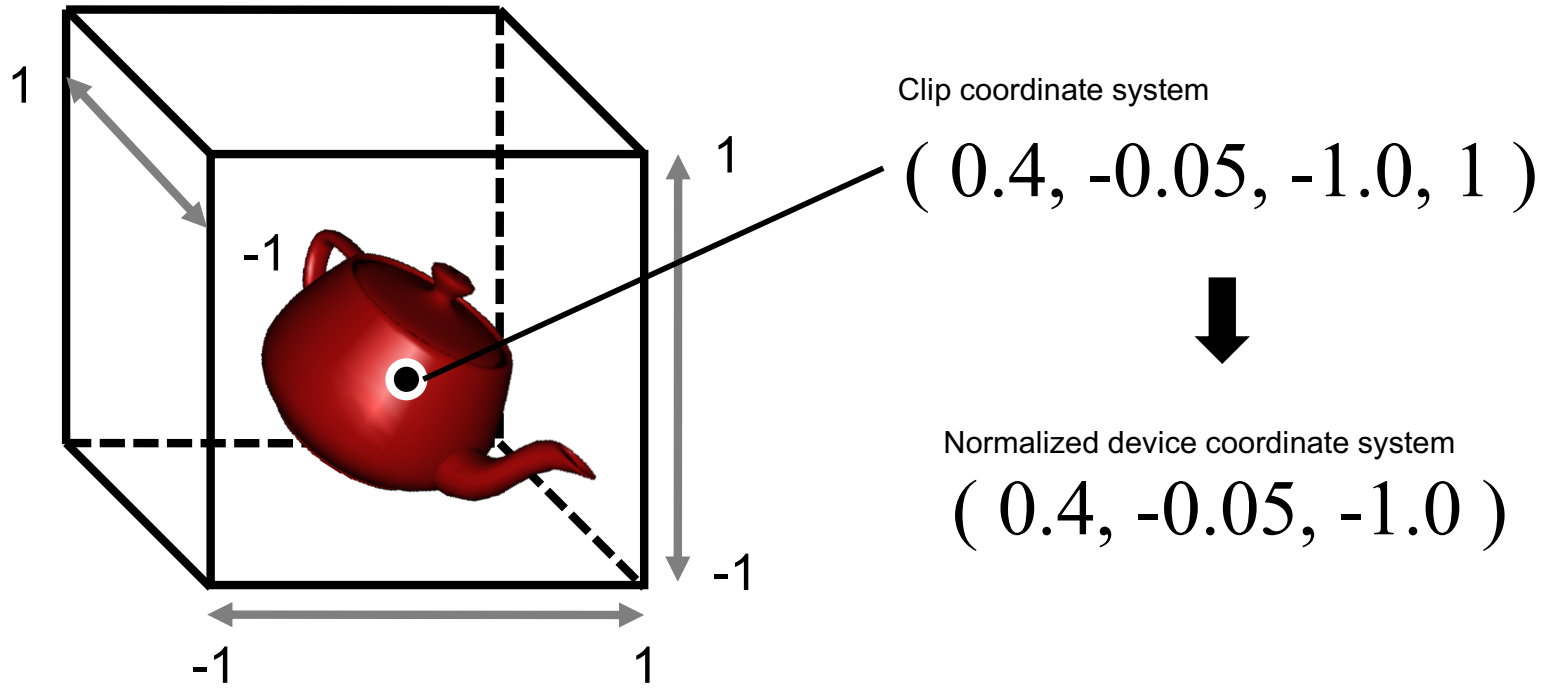
Clip Coordinate System

- A coordinate system to clip points in the camera coordinate system based on the view frustum.
  - Range: $-w \leqq x \leqq w$, $-w \leqq y \leqq w$, $-w \leqq z \leqq w$



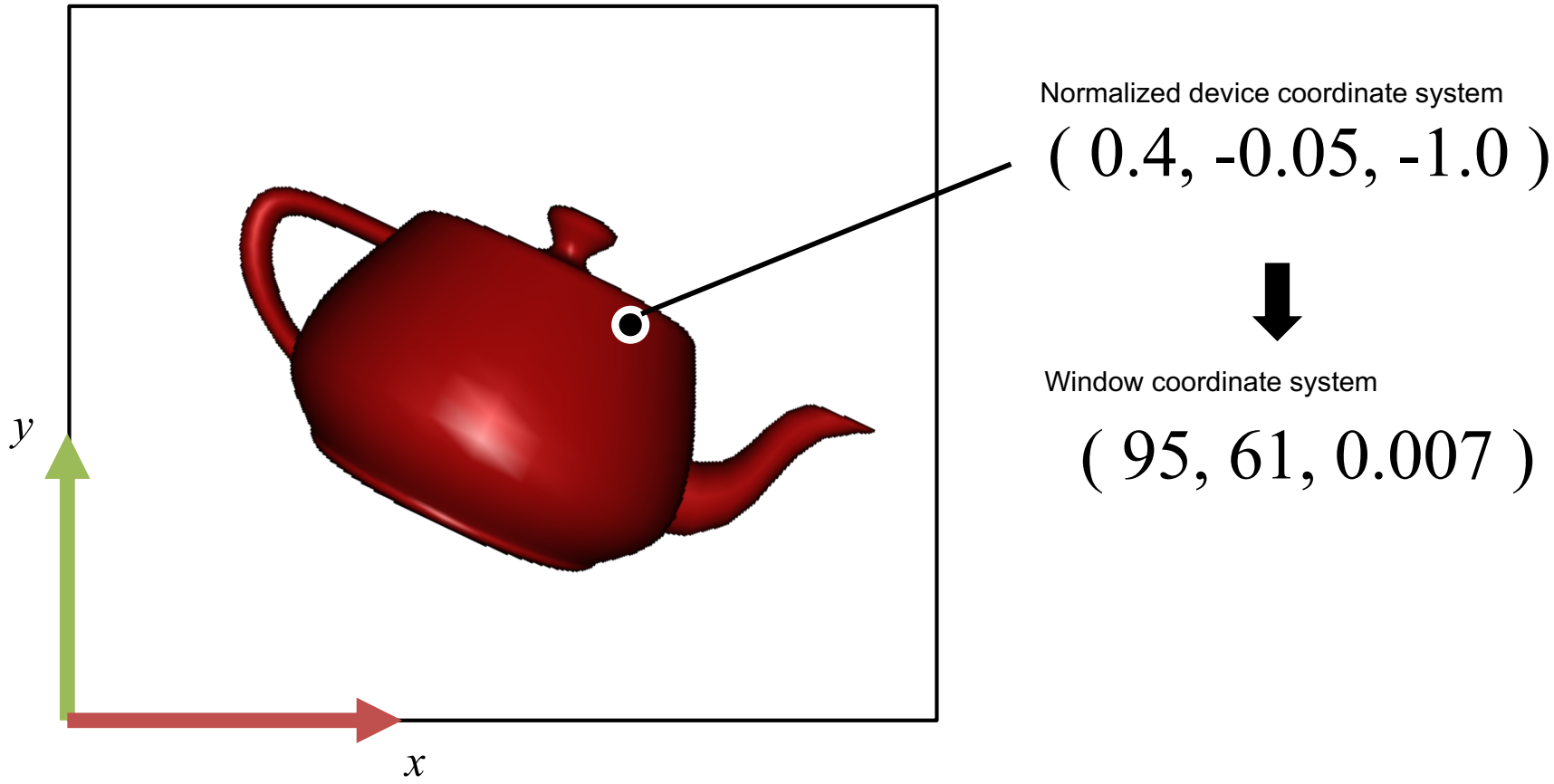
Normalized Device Coordinate System

- A coordinate system yielded by dividing the clip coordinates by w.
  - Range: $-1 \leqq x \leqq 1,\ -1 \leqq y \leqq 1,\ -1 \leqq z \leqq 1$



Window Coordinate System

- A 2D coordinate system defined on a projection plane.



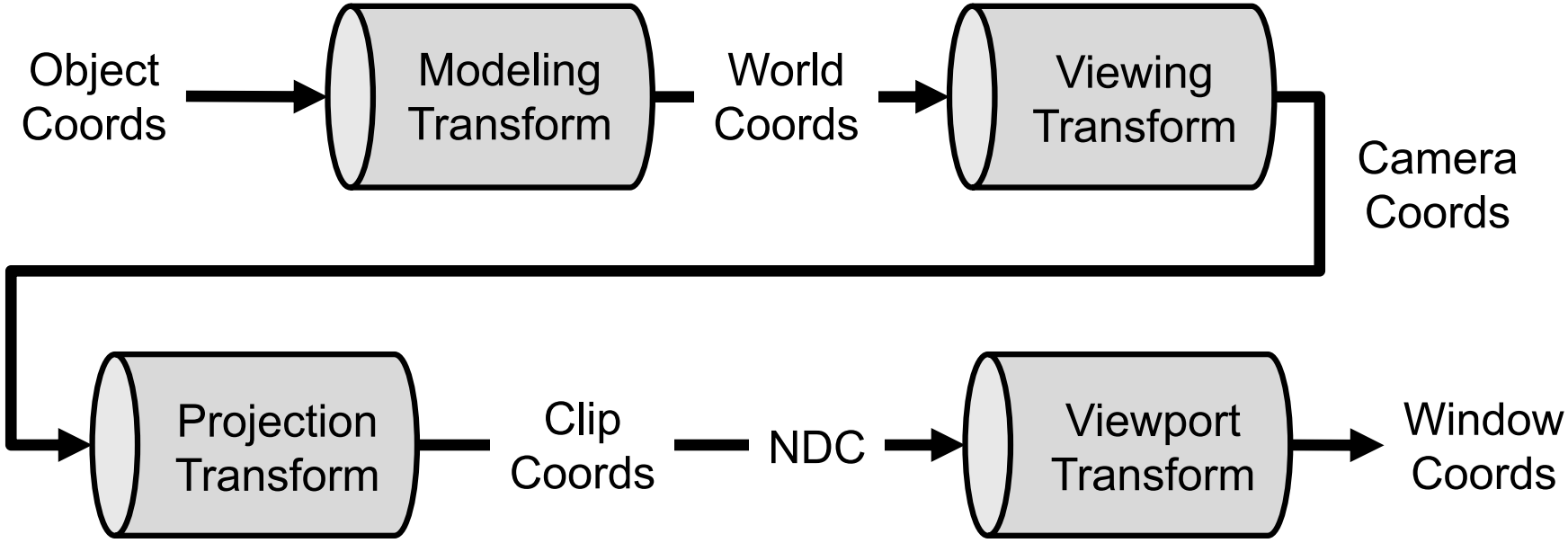
Coordinate Transformations

- 3D Objects are rendered through the following coordinate transformations

1. Modeling transformation
   - Object coordinates to world coordinates
2. Viewing transformation
   - World coordinates to camera coordinates
3. Projection transformation
   - Camera coordinates to clip coordinates (NDC)
4. Viewport transformation
   - NDC to window coordinates

Coordinate Transformations

- Transformation pipeline



Modeling Transformation

- Object coordinates to world coordinates
  - Scaling matrix $\boldsymbol{S}$
  - Translation matrix $\boldsymbol{T}$
  - Rotation matrix $\boldsymbol{R}$

} Modeling matrix (Combined matrix) $\boldsymbol{M}_{\text{model}}$

$$\begin{bmatrix} x_{\text{world}} \\ y_{\text{world}} \\ z_{\text{world}} \\ w_{\text{world}} \end{bmatrix} = M_{\text{model}} \begin{bmatrix} x_{\text{obj}} \\ y_{\text{obj}} \\ z_{\text{obj}} \\ w_{\text{obj}} \end{bmatrix}$$

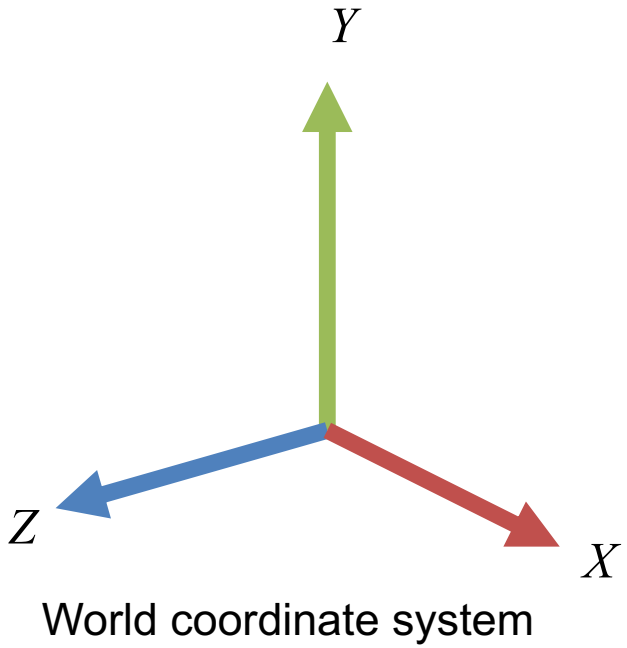
Viewing Transformation

- World coordinates to camera coordinates
  - Camera position and orientation in world coords

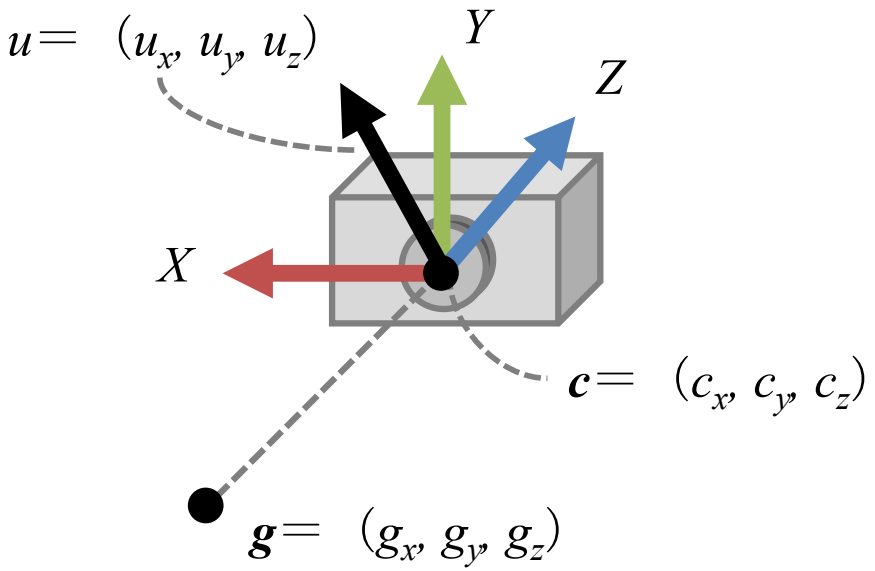
$$\begin{bmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \\ w_{\text{cam}} \end{bmatrix} = M_{\text{view}} \begin{bmatrix} x_{\text{world}} \\ y_{\text{world}} \\ z_{\text{world}} \\ w_{\text{world}} \end{bmatrix}$$

Viewing Transformation

- Camera position $\boldsymbol{c} = (c_x, c_y, c_z)$
- Up vector $\boldsymbol{u} = (u_x, u_y, u_z)$
- Look-at point $\boldsymbol{g} = (g_x, g_y, g_z)$



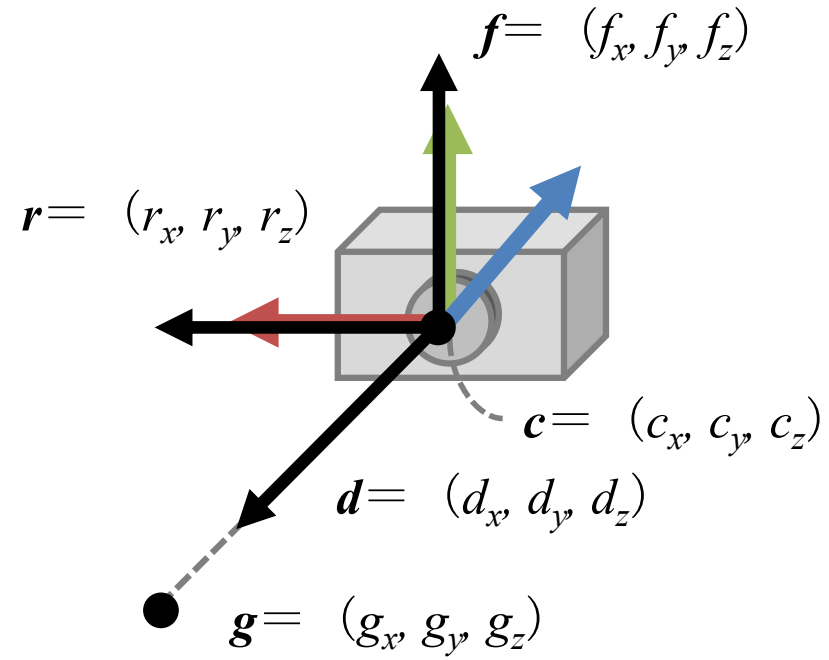
World coordinate system



Viewing Transformation

- Viewing matrix $\boldsymbol{M}_{\text{view}} = \boldsymbol{C}_{\text{dir}}\,\boldsymbol{C}_{\text{pos}}$
  - $\boldsymbol{C}_{\text{dir}}$: Camera direction matrix

$$C_{\text{dir}} = \begin{bmatrix} r_x & r_y & r_z & 0 \\ f_x & f_y & f_z & 0 \\ -d_x & -d_y & -d_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$d = \frac{g - c}{|g - c|} \quad f = \frac{r \times d}{|r \times d|} \quad r = \frac{d \times u}{|d \times u|}$$

Projection Transformation

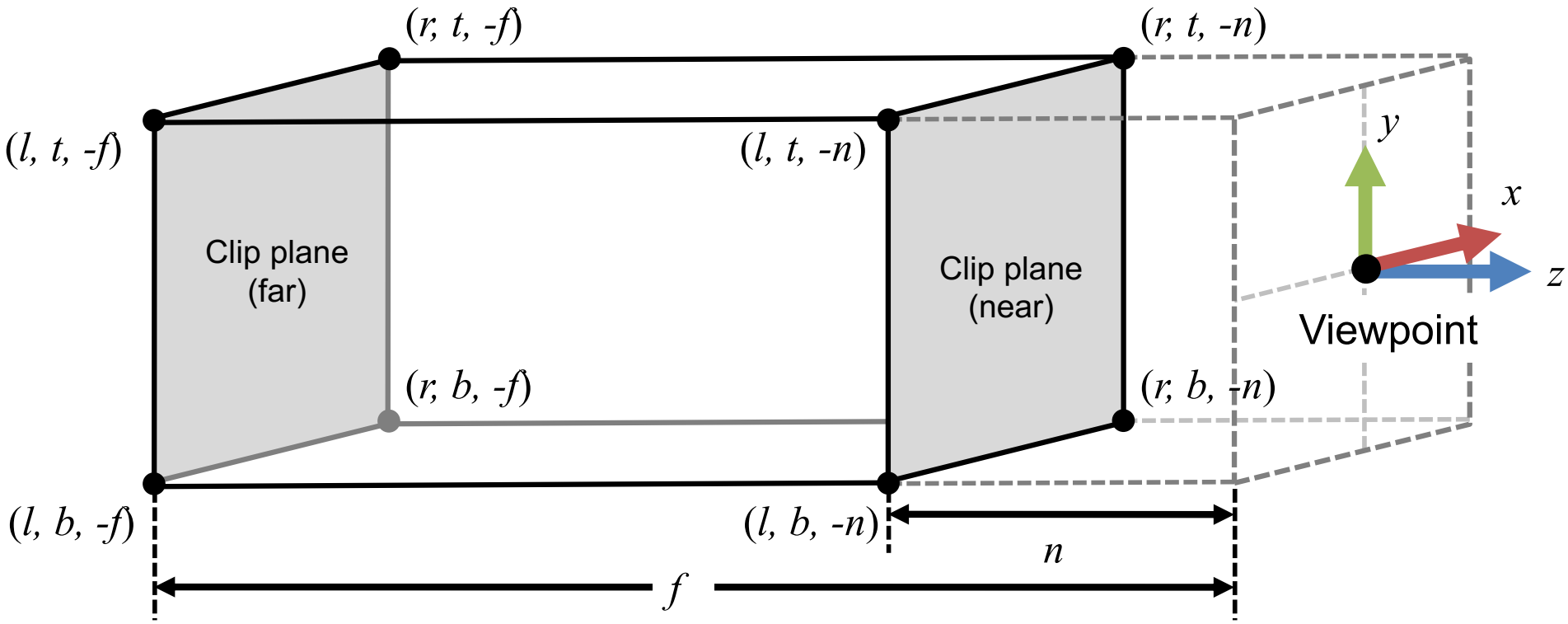
- Camera coordinates to clip coordinates (normalized device coordinates)
  - Orthogonal Projection
  - Perspective Projection

$$\begin{bmatrix} x_{\text{clip}} \\ y_{\text{clip}} \\ z_{\text{clip}} \\ w_{\text{clip}} \end{bmatrix} = M_{\text{proj}} \begin{bmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \\ w_{\text{cam}} \end{bmatrix}$$

$$\begin{bmatrix} x_{\text{NDC}} \\ y_{\text{NDC}} \\ z_{\text{NDC}} \end{bmatrix} = \begin{bmatrix} x_{\text{clip}} / w_{\text{clip}} \\ y_{\text{clip}} / w_{\text{clip}} \\ z_{\text{clip}} / w_{\text{clip}} \end{bmatrix}$$

Orthogonal Projection

- All projection lines are orthogonal to the projection plane.



Orthogonal Projection

- Orthogonal projection matrix $\boldsymbol{P}_{\text{orth}}$

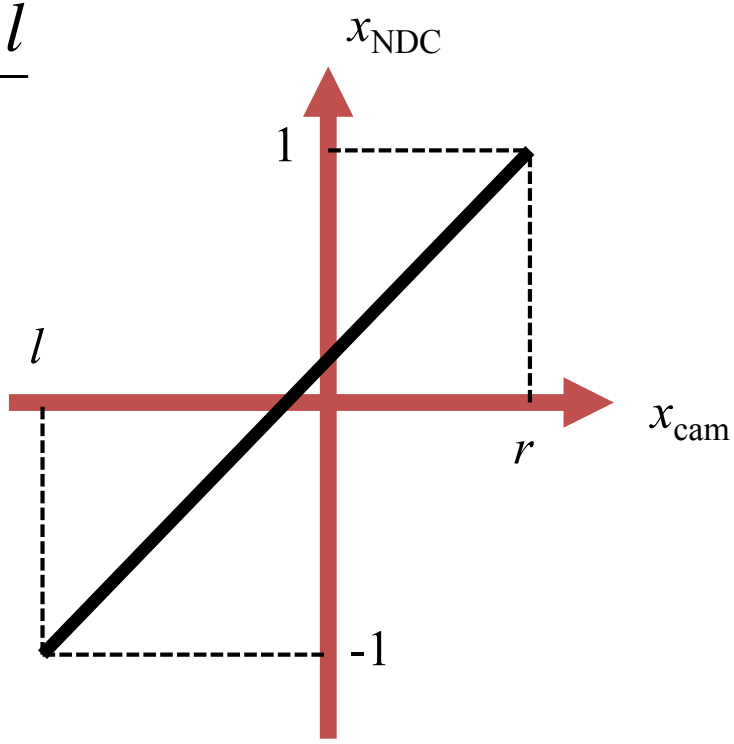
$$M_{\text{proj}} = P_{\text{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthogonal Projection Matrix

- Mapping from xcam to xNDC
  - $[l, r]$ to [-1, 1]

$$x_{\text{NDC}} = \frac{1-(-1)}{r-l}x_{\text{cam}} + \frac{r\cdot(-1)-1\cdot l}{r-l}$$

$$= \frac{2}{r-l}x_{\text{cam}} - \frac{r+l}{r-l}$$

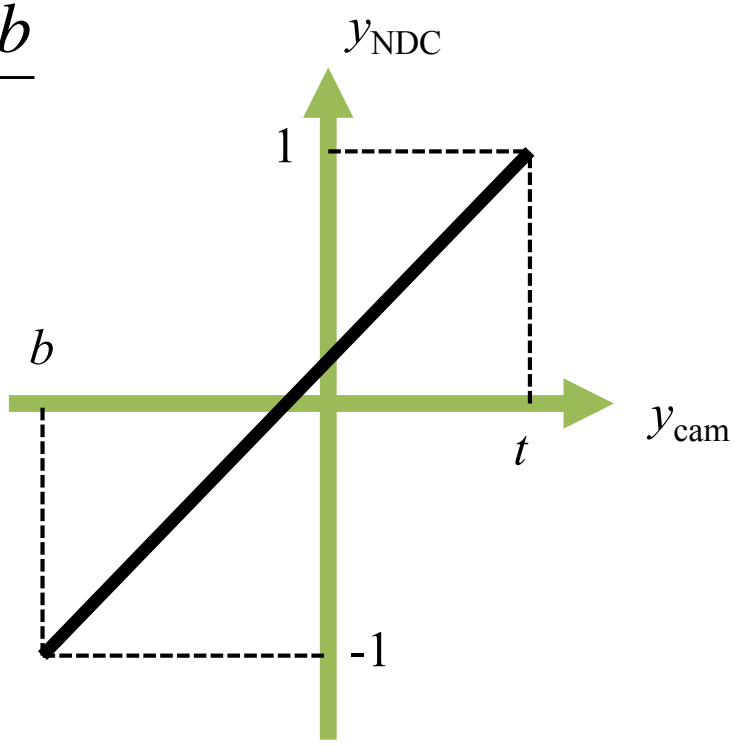


Orthogonal Projection Matrix

- Mapping from ycam to yNDC
  - $[b, t]$ to $[-1, 1]$

$$y_{\text{NDC}} = \frac{1-(-1)}{t-b} y_{\text{cam}} + \frac{t\cdot(-1)-1\cdot b}{t-b}$$

$$= \frac{2}{t-b} y_{\text{cam}} - \frac{t+b}{t-b}$$

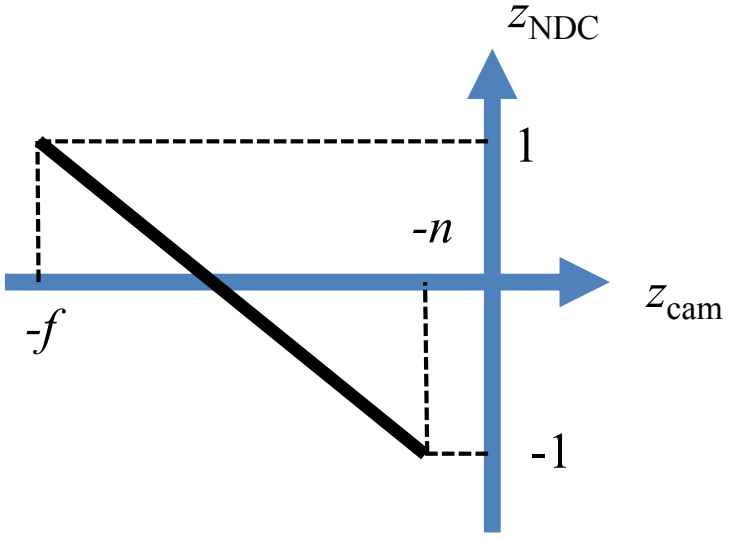


Orthogonal Projection Matrix

- Mapping from zcam to zNDC
  - [-*n*, -*f*] to [-1, 1]

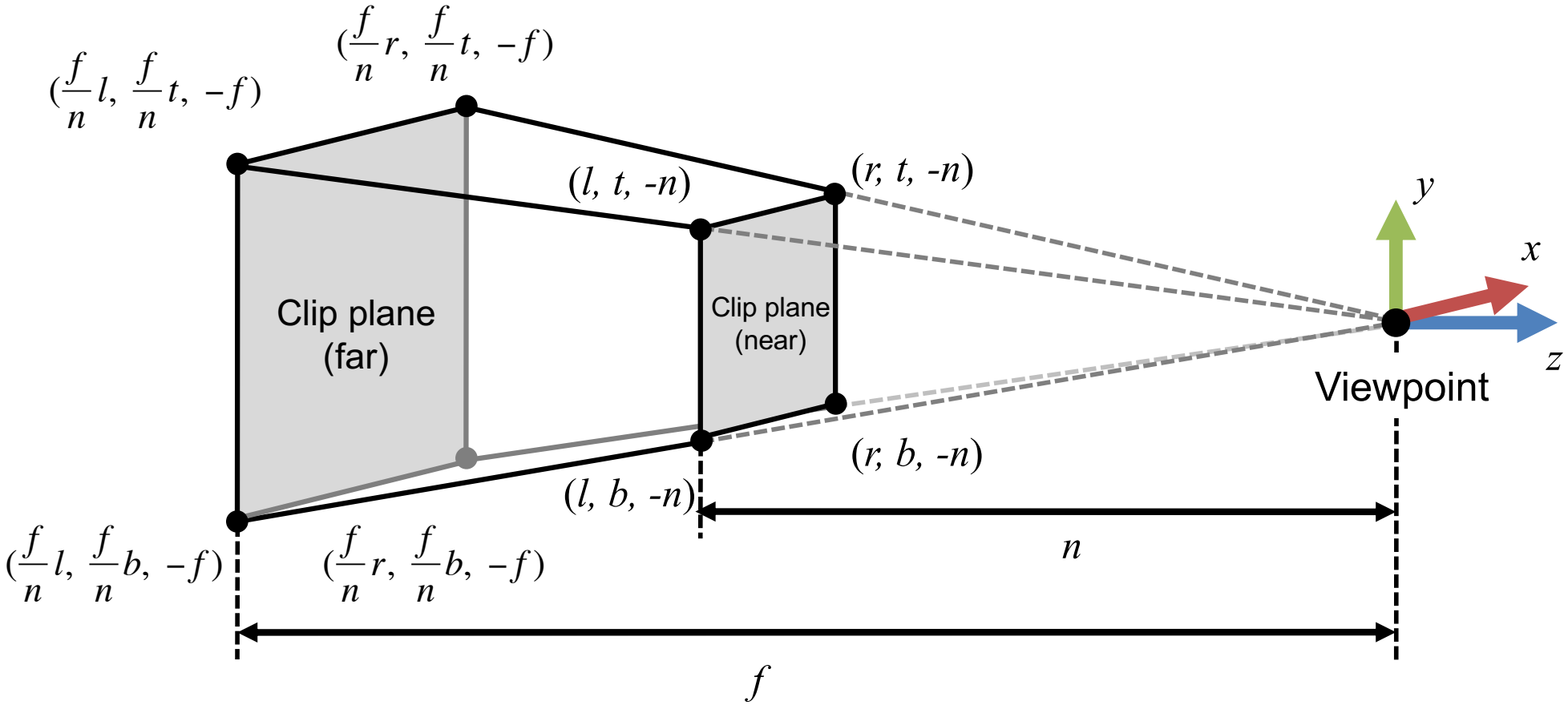
$$z_{\text{NDC}} = \frac{1-(-1)}{-f-(-n)} z_{\text{cam}} + \frac{(-f)\cdot(-1)-1\cdot(-n)}{-f-(-n)}$$

$$= -\frac{2}{f-n} z_{\text{cam}} - \frac{f+n}{f-n}$$



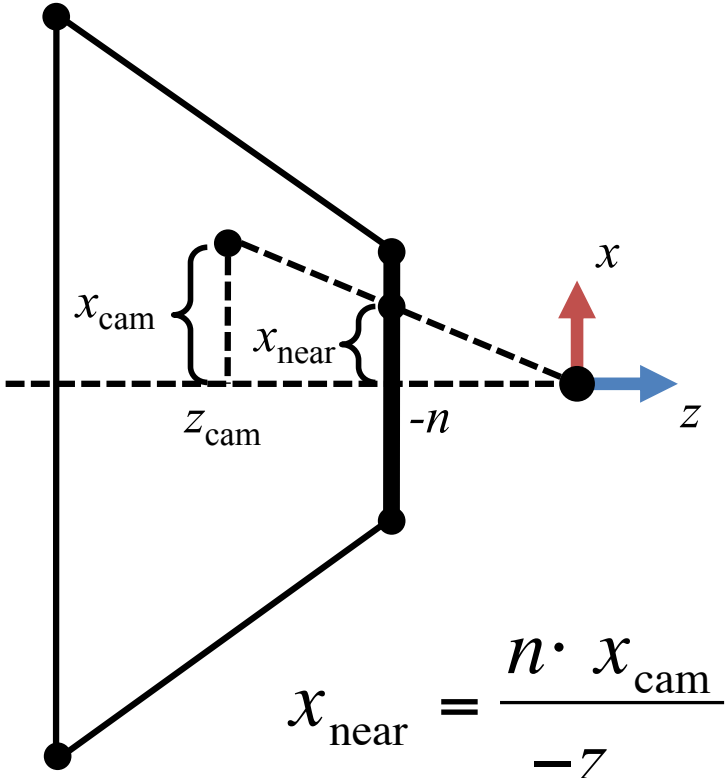
Perspective Projection

- Perspective projection has a center of projection (viewpoint)

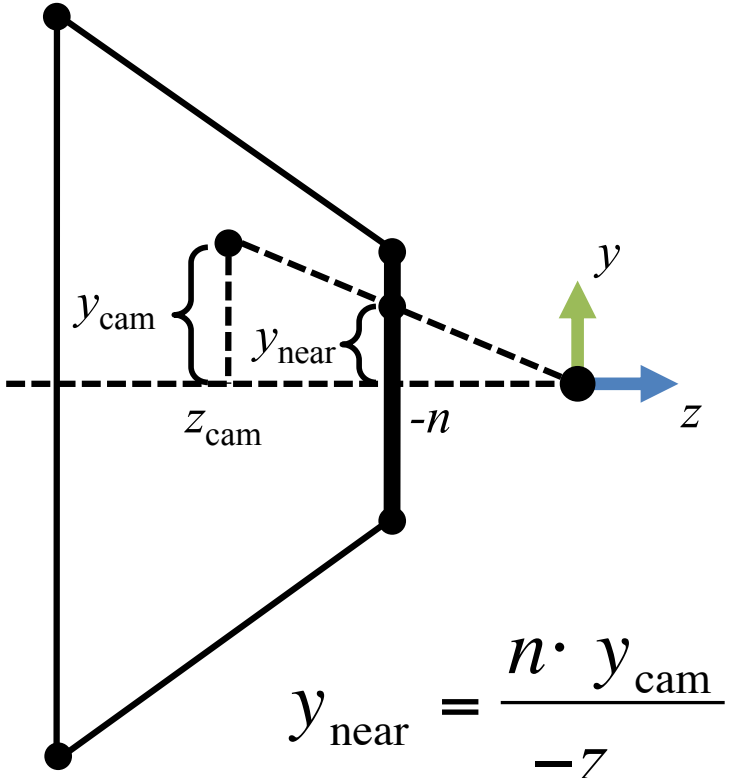


Perspective Projection Matrix

- Projected point $(x_{\text{near}}, y_{\text{near}})$ onto the near clipping plane



$$x_{\text{near}} = \frac{n \cdot x_{\text{cam}}}{-z_{\text{cam}}}$$



$$y_{\text{near}} = \frac{n \cdot y_{\text{cam}}}{-z_{\text{cam}}}$$

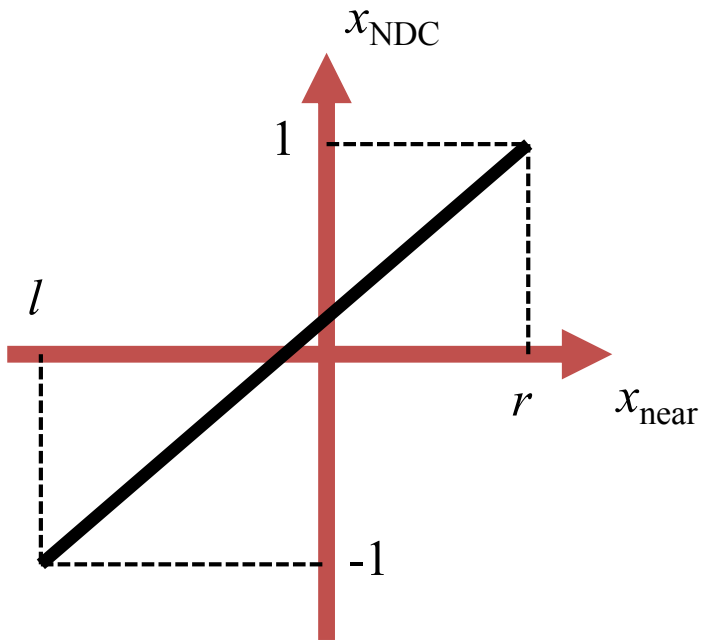
Perspective Projection

- Perspective projection matrix $\boldsymbol{P}_{\text{pers}}$

$$M_{\text{proj}} = P_{\text{pers}} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective Projection Matrix

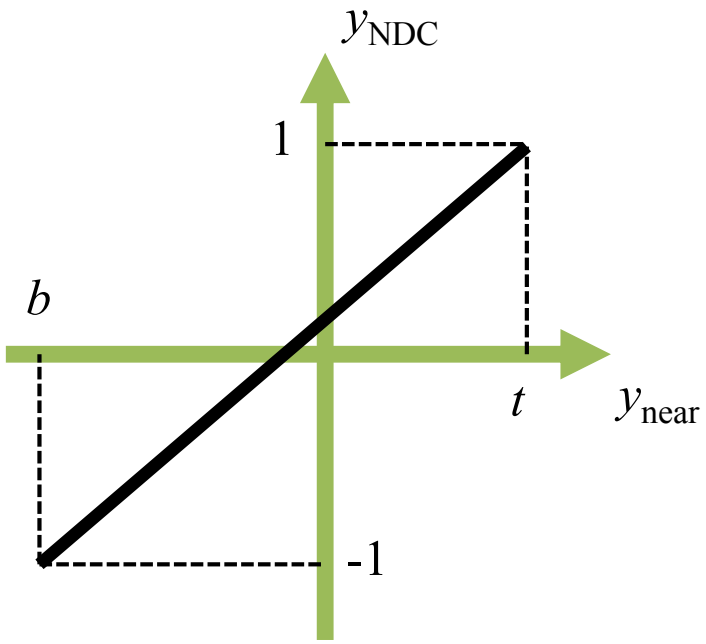
- Mapping from xnear to xNDC
  - $[l, r]$ to [-1,1]



$$
\begin{aligned}
x_{\text{NDC}} &= \frac{2}{r-l} x_{\text{near}} - \frac{r+l}{r-l} \\
&= \frac{2}{r-l} \cdot \frac{n \cdot x_{\text{cam}}}{-z_{\text{cam}}} - \frac{r+l}{r-l} \\
&= \left(\frac{2n}{r-l} x_{\text{cam}} + \frac{r+l}{r-l} z_{\text{cam}} \right) \Big/ -z_{\text{cam}}
\end{aligned}
$$

Perspective Projection Matrix

- Mapping from ynear to yNDC
  - $[b, t]$ to [-1, 1]



$$
\begin{aligned}
y_{\text{NDC}} &= \frac{2}{t-b} y_{\text{near}} - \frac{t+b}{t-b} \\
&= \frac{2}{t-b} \cdot \frac{n \cdot y_{\text{cam}}}{-z_{\text{cam}}} - \frac{t+b}{t-b} \\
&= \left(\frac{2n}{t-b} x_{\text{cam}} + \frac{t+b}{t-b} z_{\text{cam}} \right) \Big/ -z_{\text{cam}}
\end{aligned}
$$

Perspective Projection Matrix

- Mapping from zcam to zNDC
  - $[-n, -f]$ to $[-1, 1]$
  - The projection does not depend on xcam and ycam.
  - Solve for A and B in the following matrix.

$$\begin{bmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ w_{cam} \end{bmatrix} \Rightarrow \begin{aligned} z_{clip} &= Az_{cam} + Bw_{cam} \\ w_{clip} &= -z_{cam} \\ \therefore z_{NDC} &= \frac{z_{clip}}{w_{clip}} = \frac{Az_{cam} + Bw_{cam}}{-z_{cam}} \end{aligned}$$

Perspective Projection Matrix

- $w_{\text{cam}}=1$
- $(z_{\text{cam}}, z_{\text{NDC}})=(-n, -1), (-f, 1)$

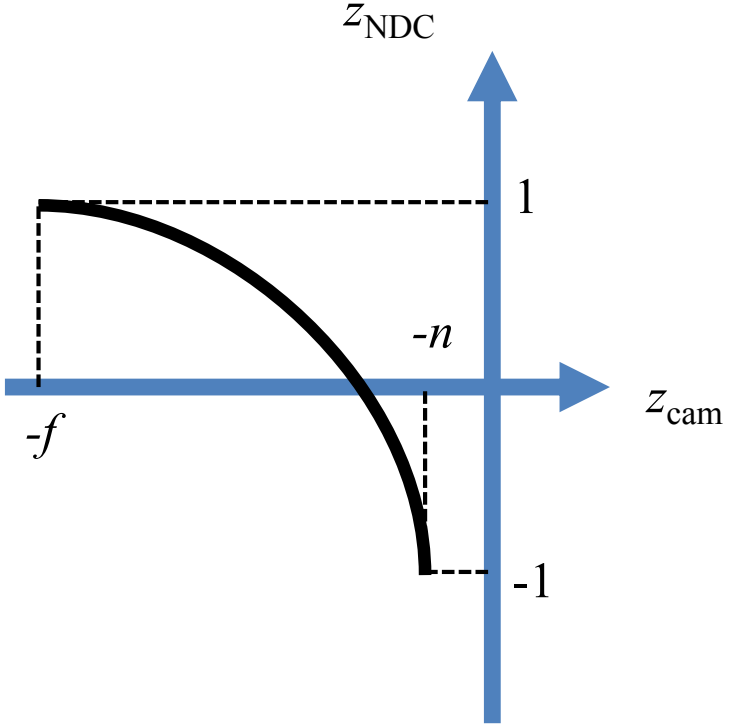
$$\begin{cases} \dfrac{-An+B}{n} = -1 \\ \dfrac{-Af+B}{f} = 1 \end{cases} \Rightarrow \begin{cases} A = -\dfrac{f+n}{f-n} \\ B = -\dfrac{2fn}{f-n} \end{cases}$$

$$\therefore z_{NDC} = \frac{Az_{cam}+Bw_{cam}}{-z_{cam}} = \frac{-\dfrac{f+n}{f-n}z_{cam} - \dfrac{2fn}{f-n}}{-z_{cam}}$$

z_{NDC} and z_{cam}

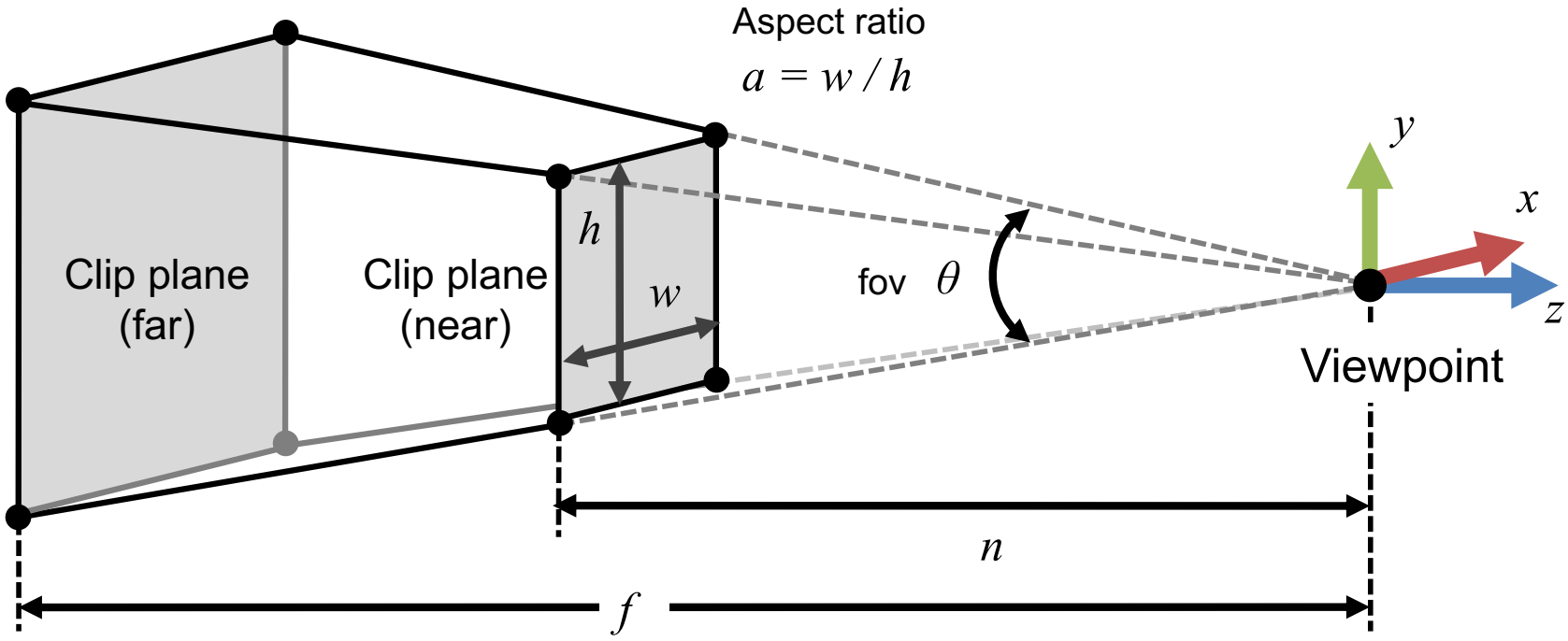
- Nonlinear mapping of z_{cam} to z_{NDC}

$$z_{\text{NDC}} = \frac{Az_{\text{cam}} + Bw_{\text{cam}}}{-z_{\text{cam}}} = \frac{-\frac{f+n}{f-n} z_{\text{cam}} - \frac{2fn}{f-n}}{-z_{\text{cam}}} = \frac{2fn}{f-n}\frac{1}{z_{\text{cam}}} + \frac{f+n}{f-n}$$



Perspective Projection with Field-of-View

- The parameters in the perspective projection matrix can be reduced by representing it with an angle θ called as field-of-view (fov).



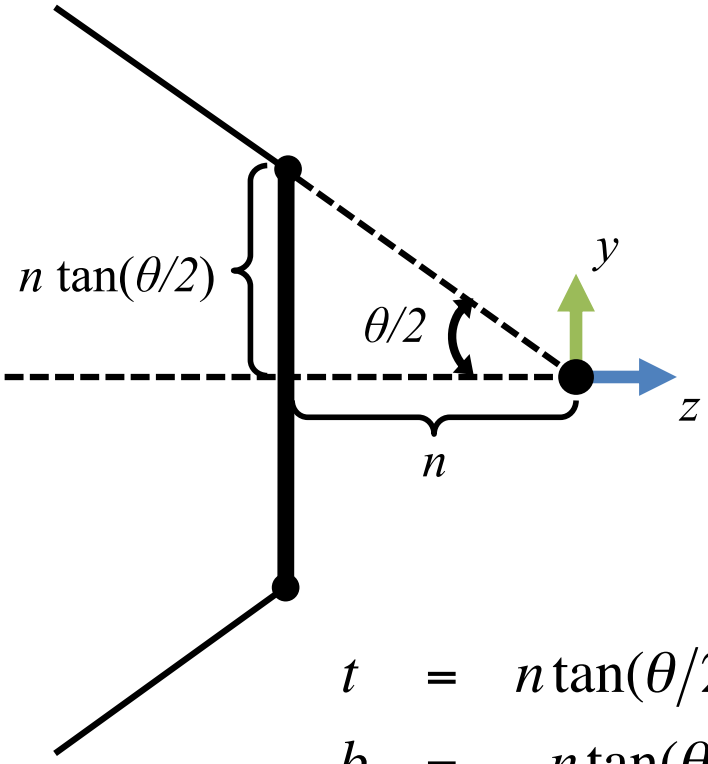
Perspective Projection with Filed-of-View

- Perspective projection matrix $\boldsymbol{P}_{\text{pers}}$

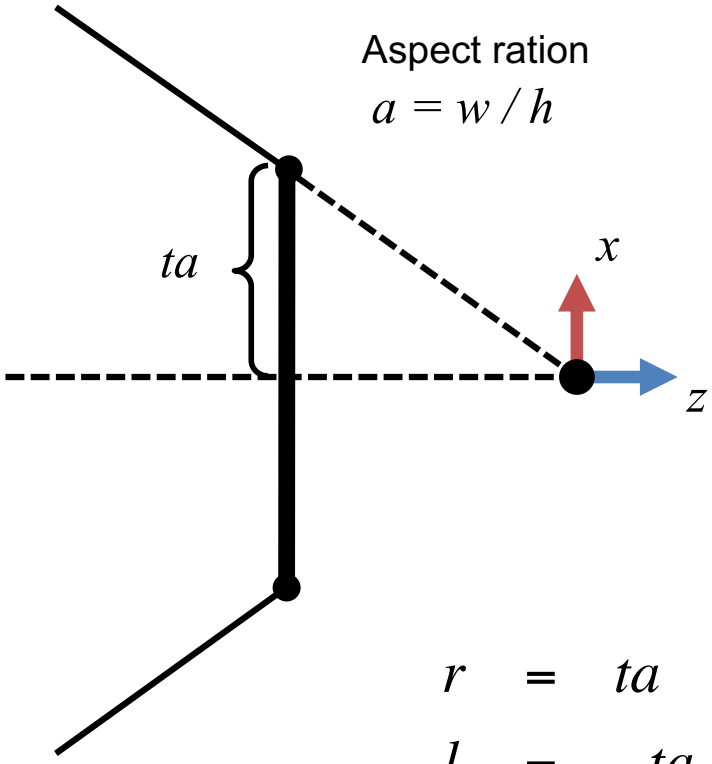
$$M_{\text{proj}} = P_{\text{pers}} = \begin{bmatrix} \frac{1}{a\tan(\theta/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\theta/2)} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective Projection Matrix with Field-of-View

- The near clipping plane represented with (r, l, t, b) is calculated as follows:



$$t = n\tan(\theta/2)$$
$$b = -n\tan(\theta/2)$$



$$r = ta$$
$$l = -ta$$

Viewport Transformation

- Normalized device coordinates (NDC) to window coordinates

$$\begin{bmatrix} x_{\text{win}} \\ y_{\text{win}} \\ z_{\text{win}} \\ 1 \end{bmatrix} = M_{\text{viewport}} \begin{bmatrix} x_{\text{NDC}} \\ y_{\text{NDC}} \\ z_{\text{NDC}} \\ 1 \end{bmatrix}$$

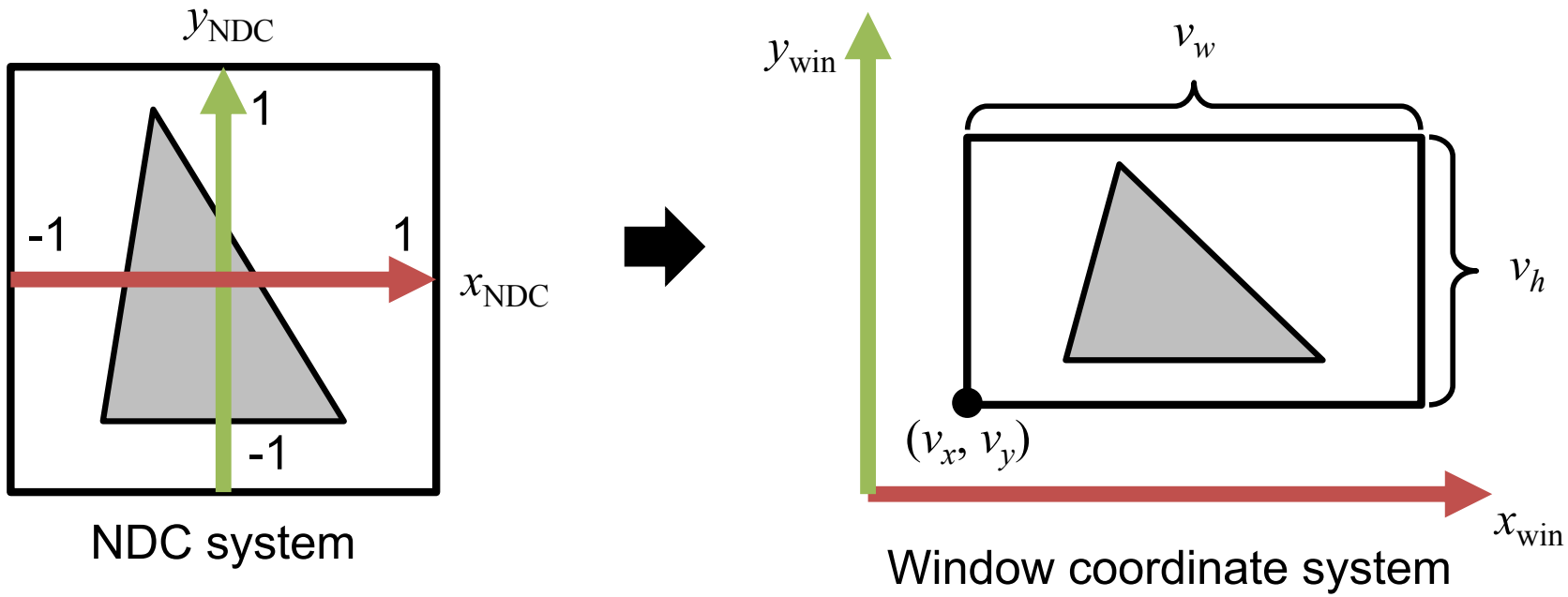
Viewport Transformation

- Viewport matrix M_{viewport}

$$M_{\text{viewport}} = \begin{bmatrix} \frac{v_w}{2} & 0 & 0 & v_x + \frac{v_w}{2} \\ 0 & \frac{v_h}{2} & 0 & v_y + \frac{v_h}{2} \\ 0 & 0 & \frac{d_f - d_n}{2} & \frac{d_f + d_n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewport Transformation

- Viewport (v_x, v_y, v_w, v_h)
- Range of depth value $[d_n, d_f]$

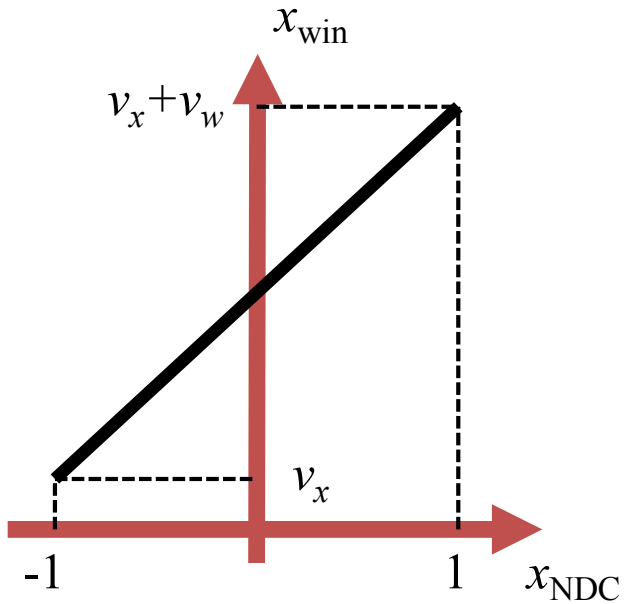


Viewport Matrix

- Mapping from xNDC to xwin
  - [-1, 1] to $[v_x, v_x+v_w]$

$$x_{\text{win}} = \frac{(v_x + v_w) - v_x}{1-(-1)} x_{\text{NDC}} + \frac{1\cdot v_x - (-1)\cdot (v_x + v_w)}{1-(-1)}$$

$$= \frac{v_w}{2} x_{\text{NDC}} + \left(v_x + \frac{v_w}{2} \right)$$

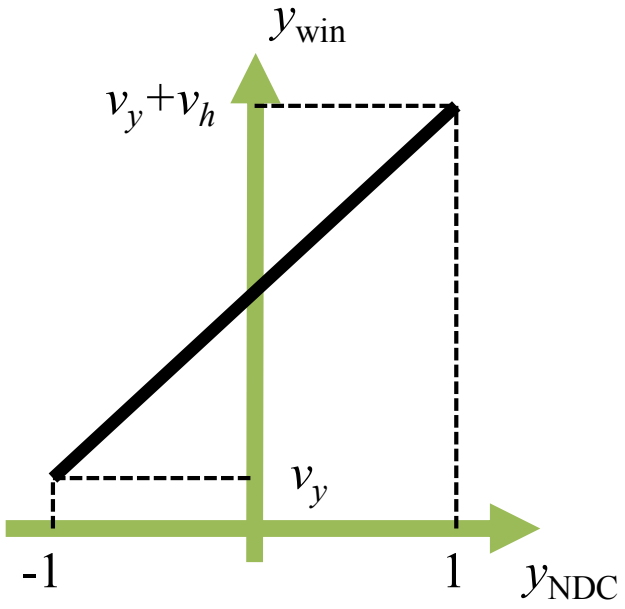


Viewport Matrix

- Mapping from yNDC to ywin
  - [-1, 1] to $[v_y, v_y+v_h]$

$$y_{\text{win}} = \frac{(v_y + v_h) - v_y}{1-(-1)} y_{\text{NDC}} + \frac{1 \cdot v_y - (-1) \cdot (v_y + v_h)}{1-(-1)}$$

$$= \frac{v_h}{2} y_{\text{NDC}} + \left(v_y + \frac{v_h}{2} \right)$$

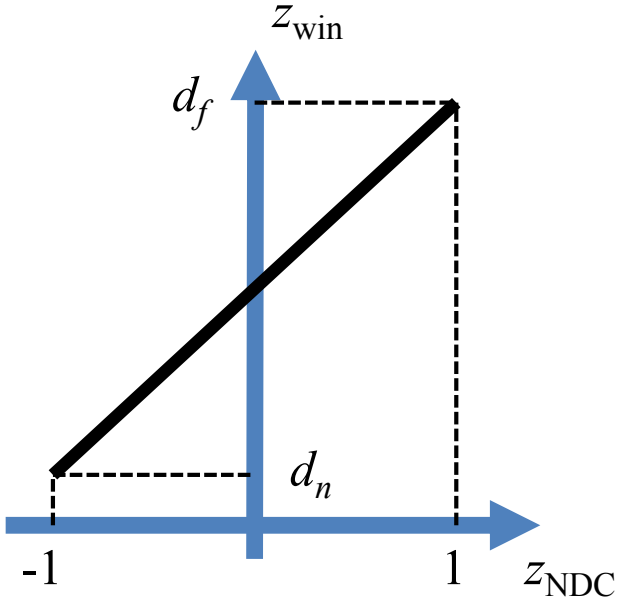


Viewport Matrix

- Mapping from zNDC to zwin
  - [-1, 1] to $[d_n, d_f]$

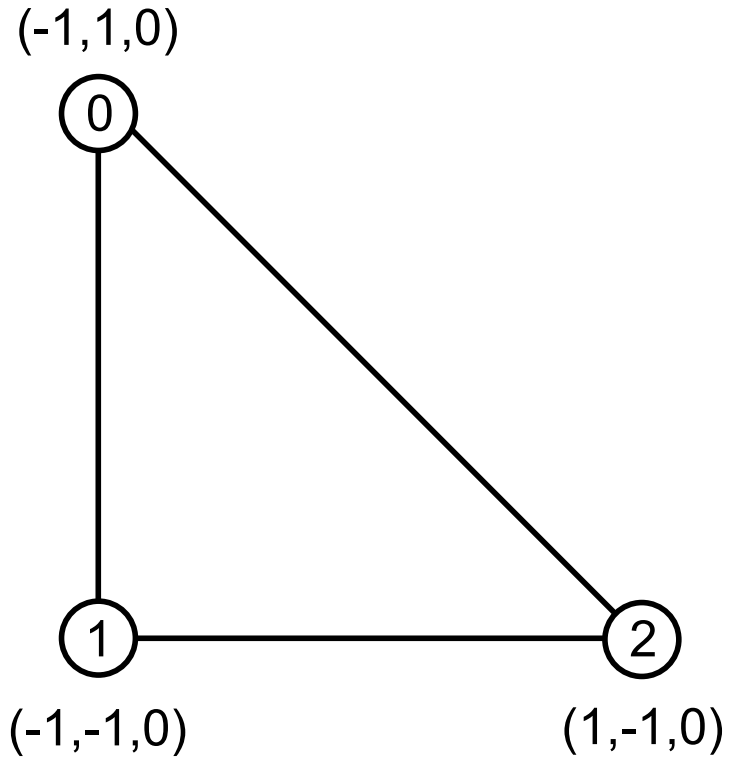
$$z_{\text{win}} = \frac{d_f - d_n}{1-(-1)} z_{\text{NDC}} + \frac{1\cdot d_n - (-1)\cdot d_f}{1-(-1)}$$

$$= \frac{d_f - d_n}{2} z_{\text{NDC}} + \frac{d_f + d_n}{2}$$



Drawing a triangle geometry

- A triangle geometry has three vertices and a connectivity list.




```
var vertices = [
    [-1, 1, 0], // v0
    [-1,-1, 0], // v1
    [ 1,-1, 0]  // v2
];

var faces = [
    [0,1,2] // f0: v0-v1-v2
];
```

Drawing a triangle geometry

- THREE.Geometry

```
var v0 = new THREE.Vector3().fromArray( vertices[0] );
var v1 = new THREE.Vector3().fromArray( vertices[1] );
var v2 = new THREE.Vector3().fromArray( vertices[2] );
var id = faces[0];
var f0 = new THREE.Face3( id[0], id[1], id[2] );

var geometry = new THREE.Geometry();
geometry.vertices.push( v0 );
geometry.vertices.push( v1 );
geometry.vertices.push( v2 );
geometry.faces.push( f0 );
```

Drawing a triangle geometry

- Assign a color to each face.

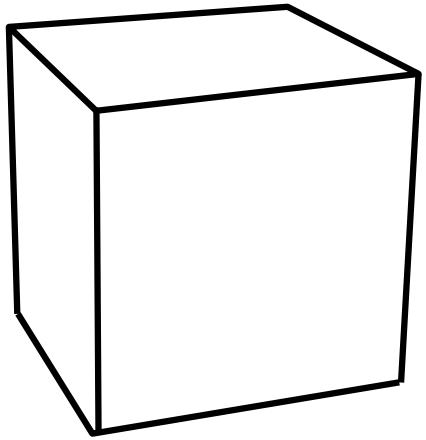
```
var material = new THREE.MeshBasicMaterial();
material.vertexColors = THREE.FaceColors;
geometry.faces[0].color = new THREE.Color( 1, 0, 0 );
```

- Assign a color to each vertex

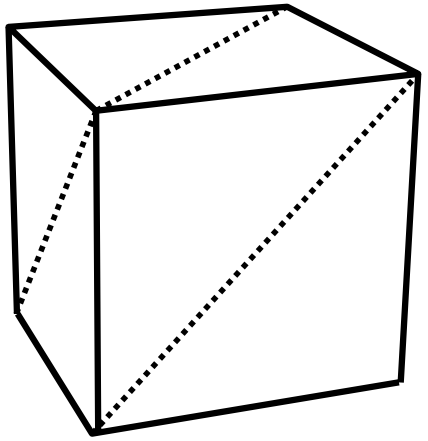
```
var material = new THREE.MeshBasicMaterial();
material.vertexColors = THREE.VertexColors;
geometry.faces[0].vertexColors.push(new THREE.Color(1,0,0));
geometry.faces[0].vertexColors.push(new THREE.Color(0,1,0));
geometry.faces[0].vertexColors.push(new THREE.Color(0,0,1));
```

Task 1

- Draw a cube by using THREE.Geometry



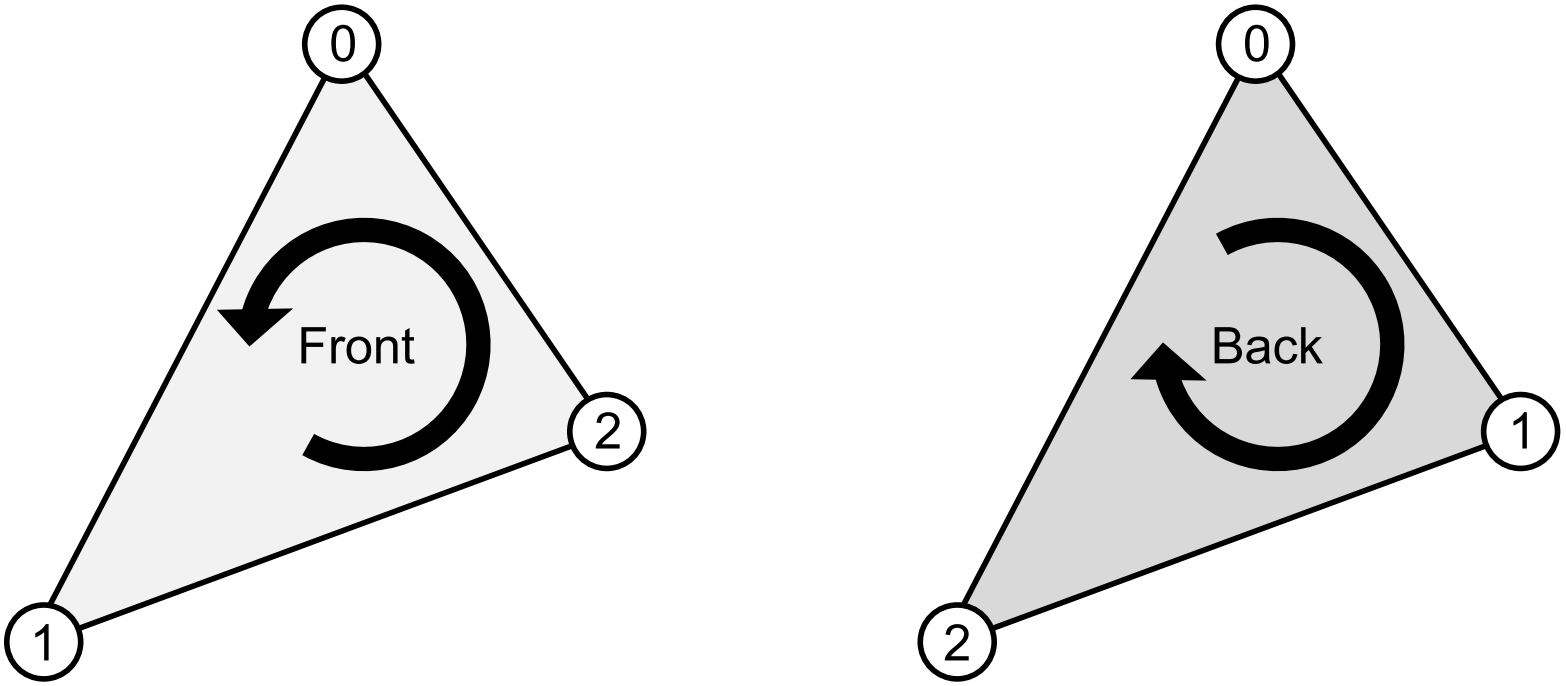
THREE.BoxGeometry



THREE.Geometry

Note

- Front and back faces
  - Depend on the order of the vertices in a triangle



Hint

- Normal Vector
  - Required for lighting (shading)

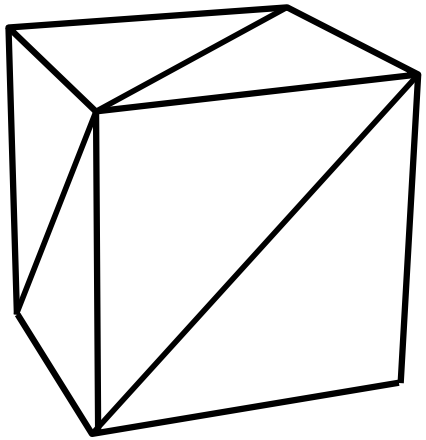
```
// Normal vectors for each face are automatically computed.
geometry.computeFaceNormals();
```

  - Both side rendering

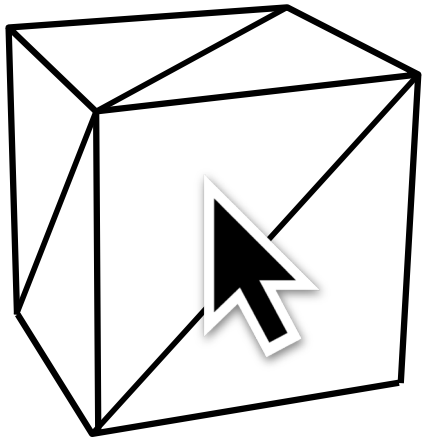
```
// Front: THREE.FrontSide (default)
// Back: THREE.BackSide
// Both: THREE.DoubleSide
metrial.side = THREE.DoubleSide
```

Task 2

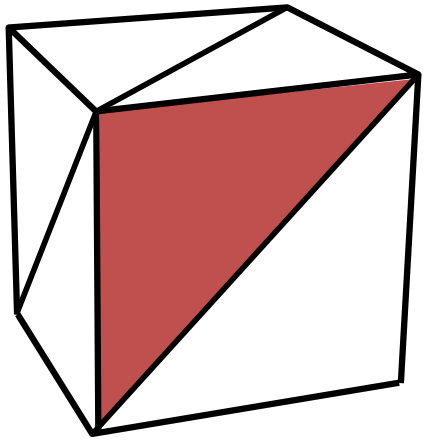
- Implement mouse picking for the triangle faces on the rotating cube.



1. Draw a rotating cube



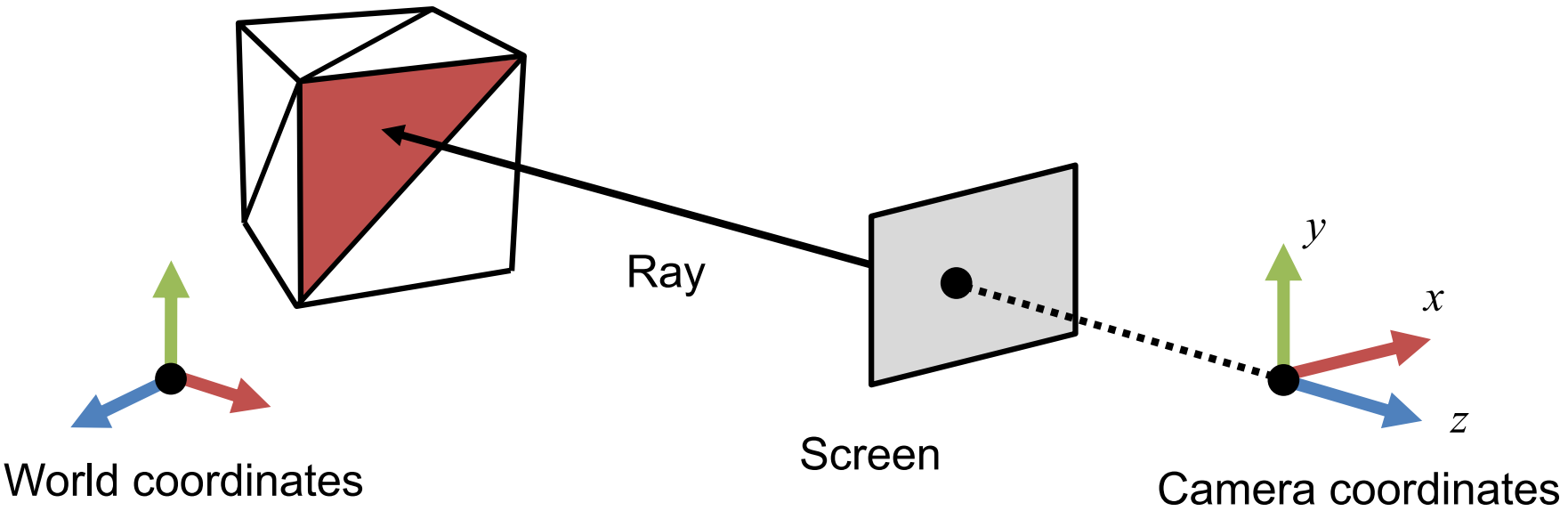
2. Click on a face



3. Change the color of the face

Hint

- Mouse picking with ray casting
  - Estimate a ray in world coordinate system from the clicked point on the screen.
  - Check ray intersections with the triangle faces.



Hint

- First of all, add a mouse down event.

```
document.addEventListener( 'mousedown', mouse_down_event );
function mouse_down_event( event )
{
    // Mouse picking
    // ...
}
```

Hint

- THREE.Raycaster

```
var raycaster = new THREE.Raycaster( origin, direction );
var intersects = raycaster.intersectObject( triangle );
if ( intersects.length > 0 )
{
    intersects[0].face.color.setRGB( 1, 0, 0 );
    intersects[0].object.geometry.colorsNeedUpdate = true;
}
```

Hint

- Clicked point in window coordinates

```
var x_win = event.clientX;
var y_win = event.clientY;
```

- Window coordinates to NDC

```
var vx = renderer.domElement.offsetLeft;
var vy = renderer.domElement.offsetTop;
var vw = renderer.domElement.width;
var vh = renderer.domElement.height;

var x_NDC = 2 * ( x_win - vx ) / vw - 1;
var y_NDC = -( 2 * ( y_win - vy ) / vh - 1 );
```

Hint

- NDC to world coordinates

```
var p_NDC = new THREE.Vector3( x_NDC, y_NDC, 1 );
var p_wld = p_NDC.unproject( camera );
```

- Origin and direction of the ray

```
var origin = ...;
var direction = ...;
```

Polling

- Take the poll
  - Student ID Number
  - Name
  - URL to Task 1
  - URL to Task 2