CPSC 314
11 –CAMERA. PROJECTIONS

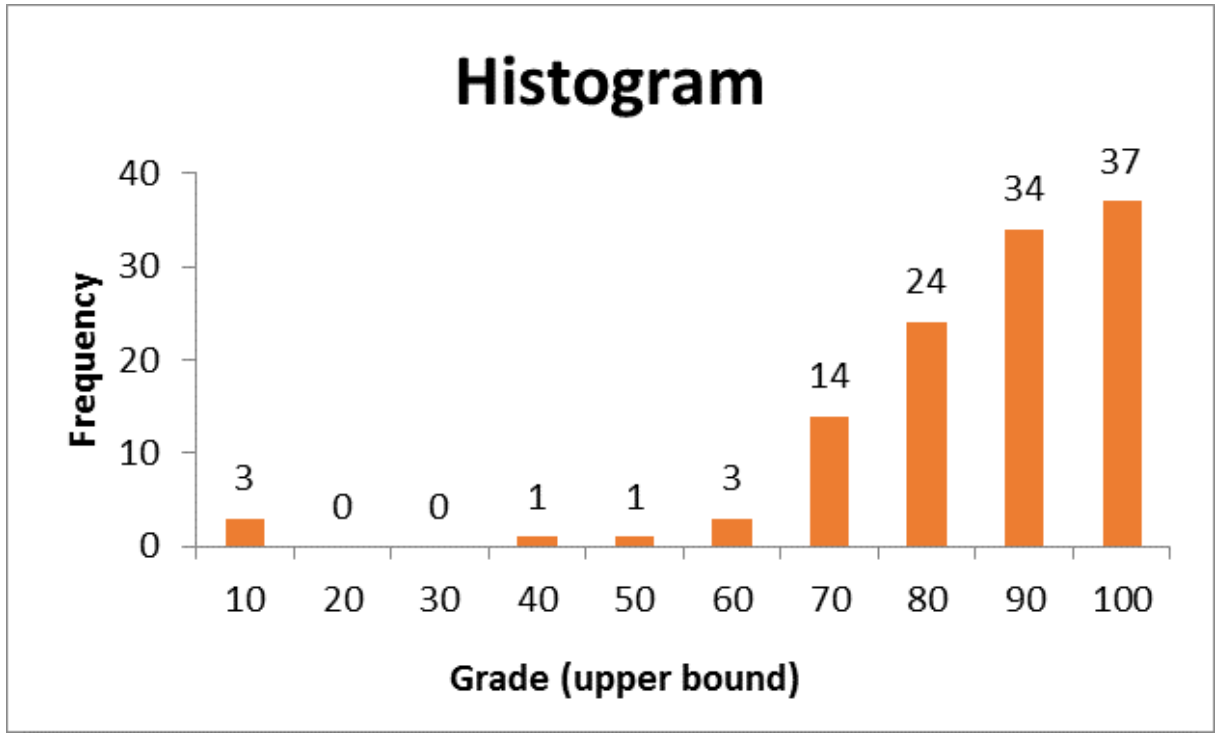
TEXTBOOK: 5.2, 10

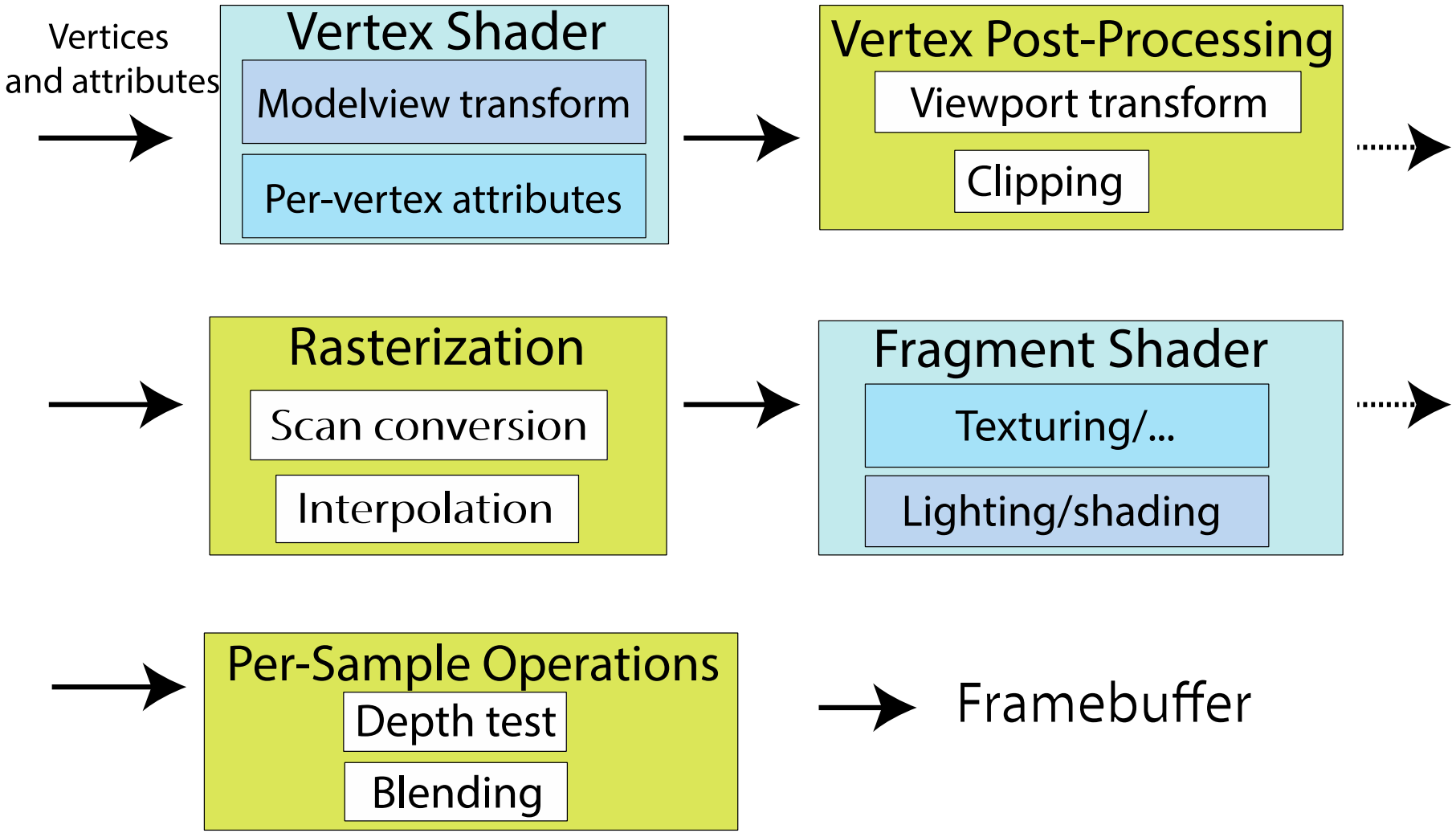
UGRAD.CS.UBC.CA/~CS314

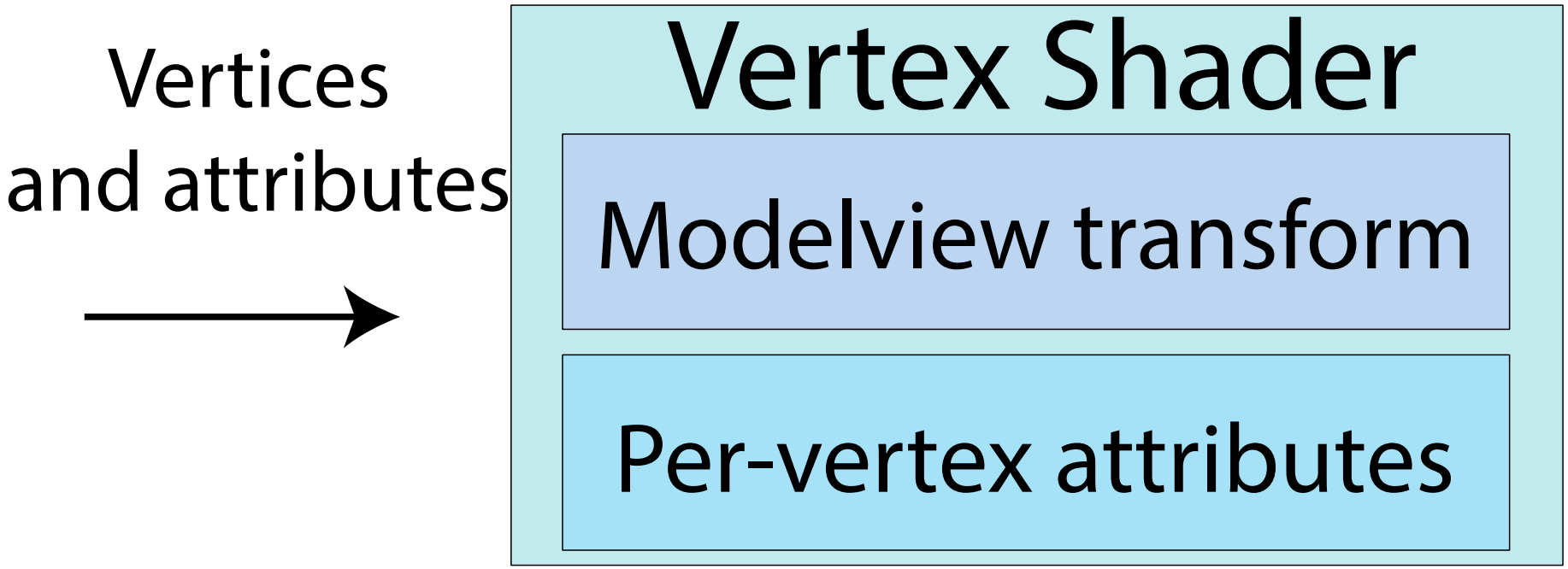
Alla Sheffer
2016

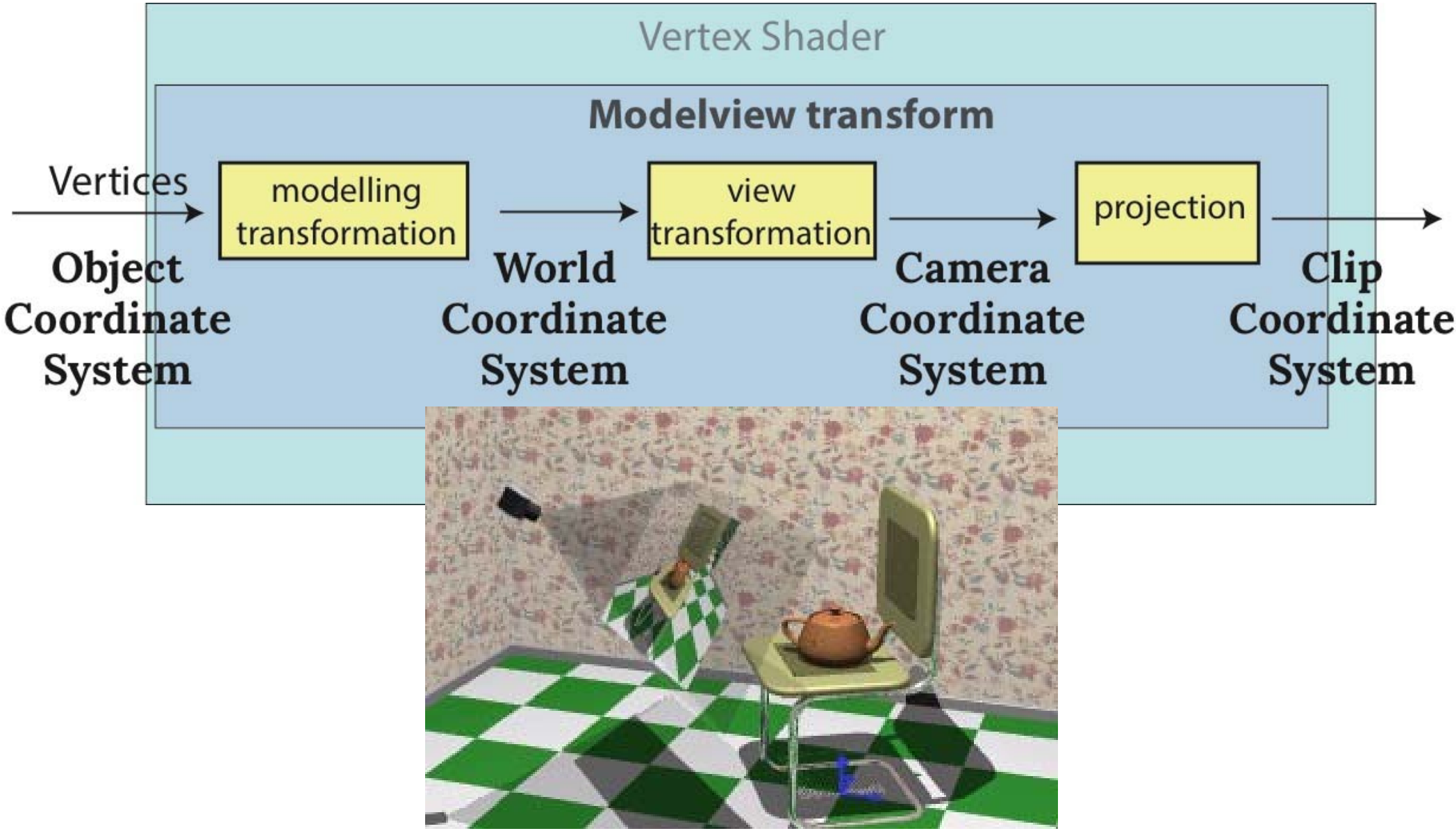
MIDTERM

- Midterm median=average = 84
- A1 average: 90

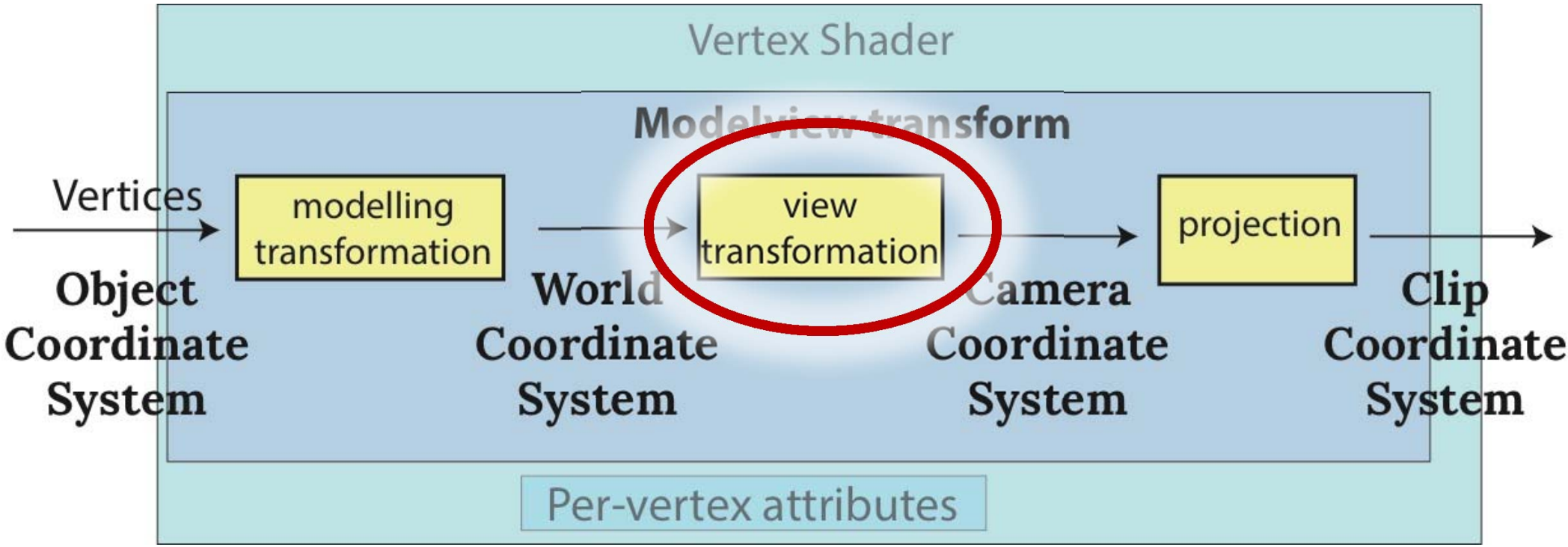



Vertices
and attributes
Vertex Shader
Modelview transform
Per-vertex attributes
Vertex Post-Processing
Viewport transform
Clipping
Rasterization
Scan conversion
Interpolation
Fragment Shader
Texturing/...
Lighting/shading
Per-Sample Operations
Depth test
Blending
Framebuffer

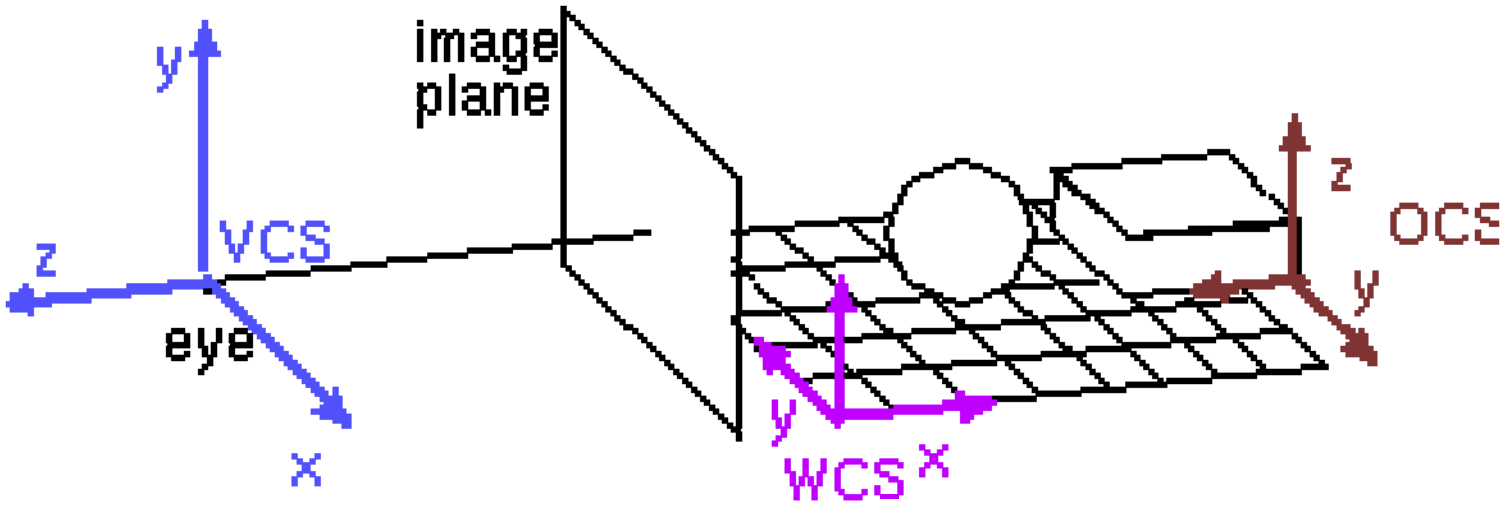

Vertices
and attributes
Vertex Shader
Modelview transform
Per-vertex attributes


Vertex Shader
Modelview transform
Vertices
modelling transformation
view transformation
projection
Object Coordinate System
World Coordinate System
Camera Coordinate System
Clip Coordinate System

VERTEX SHADER: CLOSER LOOK



POSITIONING THE CAMERA

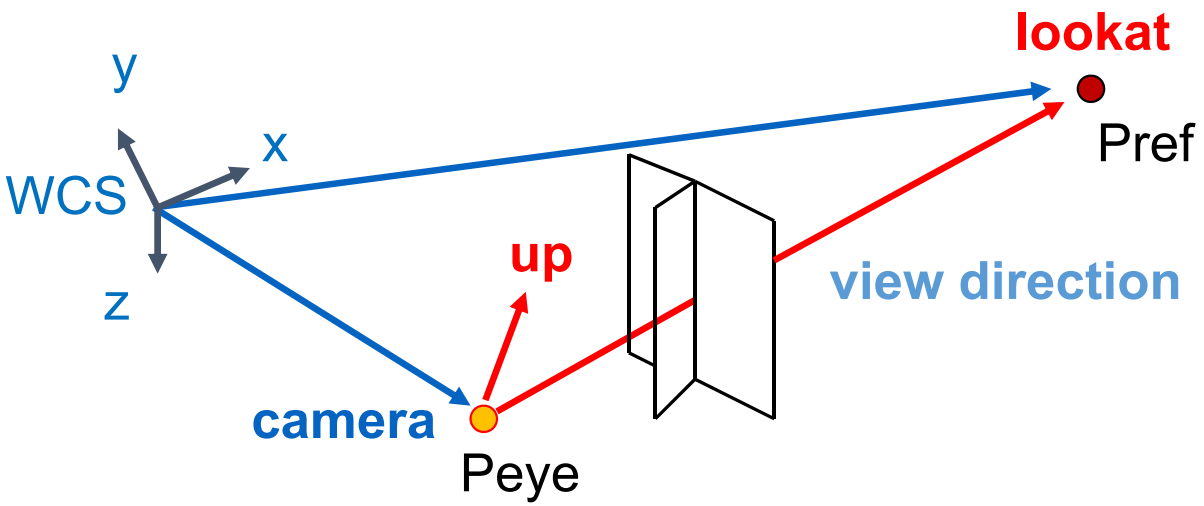


DEFINING CAMERA

- Eye point (*where is the camera?*)
- Reference point (*which point is it looking at?*)
- Up vector

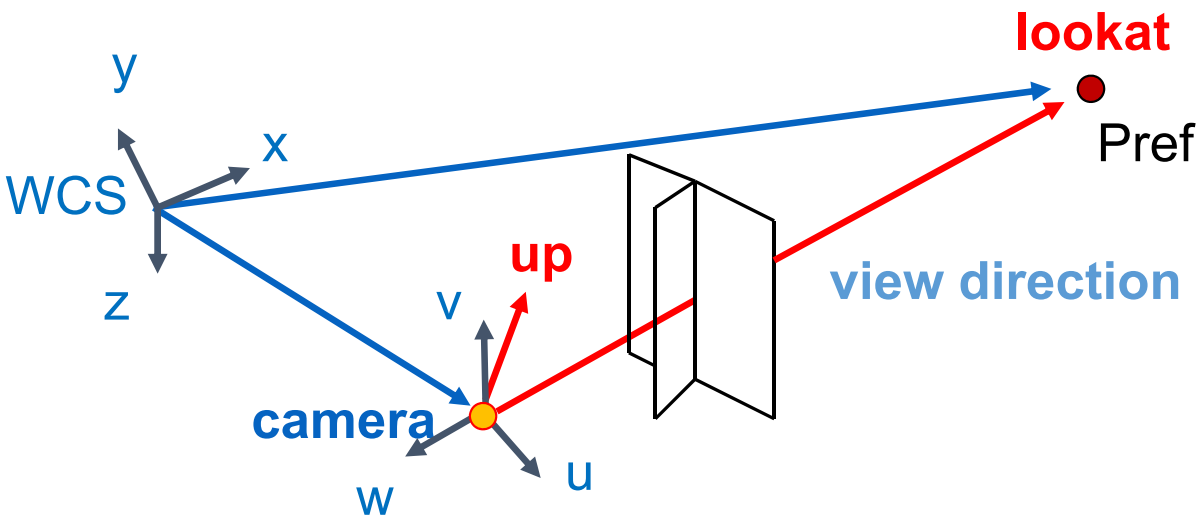
DEFINING CAMERA

- Eye point (*where is the camera?*)
- Reference/lookat point (*which point is it looking at?*)
- Up vector



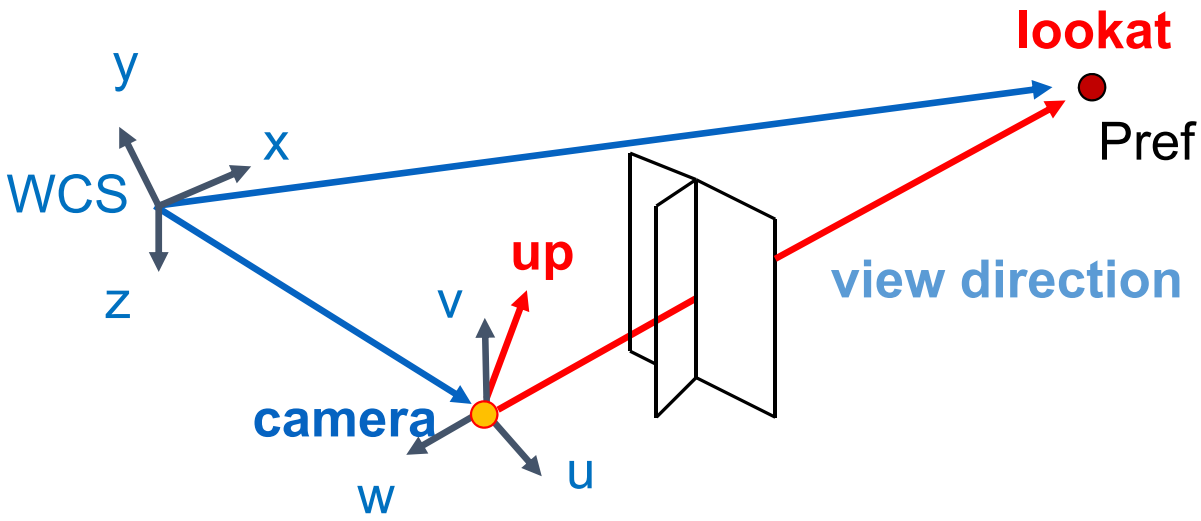
DEFINING CAMERA

- Eye point (*where is the camera?*)
- Reference/lookat point (*which point is it looking at?*)
- Up vector



CAMERA COORDINATE SYSTEM

- $w = \frac{P^{eye} - P^{ref}}{\|P^{eye} - P^{ref}\|}$
- $u = \frac{v_{up} \times w}{\|v_{up} \times w\|}$
- $v = w \times u$



CAMERA MATRIX

$$M_{cam} = \begin{bmatrix} & \\ & \end{bmatrix}$$

CAMERA MATRIX

$$M_{cam} = \begin{bmatrix} u_x & v_x & w_x & P_x^{eye} \\ u_y & v_y & w_y & P_y^{eye} \\ u_z & v_z & w_z & P_z^{eye} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CAMERA MATRIX

$$M_{cam} = \begin{bmatrix} u_x & v_x & w_x & P_x^{eye} \\ u_y & v_y & w_y & P_y^{eye} \\ u_z & v_z & w_z & P_z^{eye} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{view} = M_{cam}^{-1} = \cdots$$

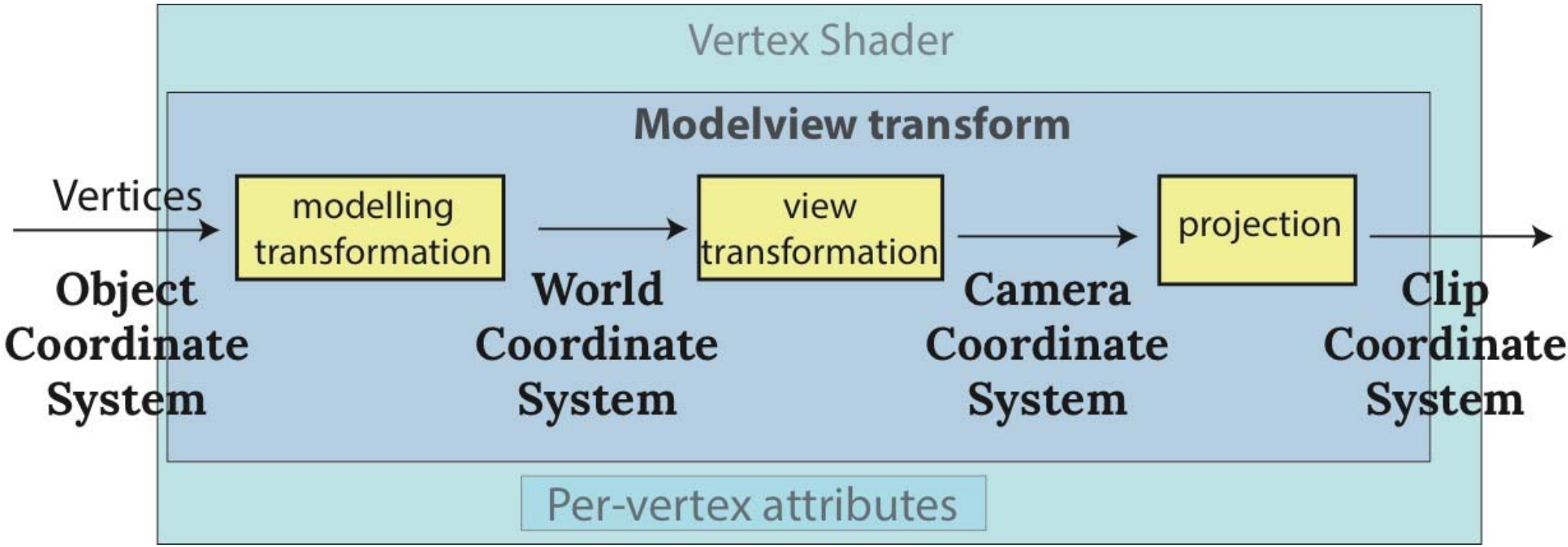
THREE.JS

```
camera = new THREE.OrthographicCamera(/*some parameters*/);
camera.position.set(30,0,0);
camera.up = new THREE.Vector3(0,0,1);
camera.lookAt(new THREE.Vector3(0,0,0));
```

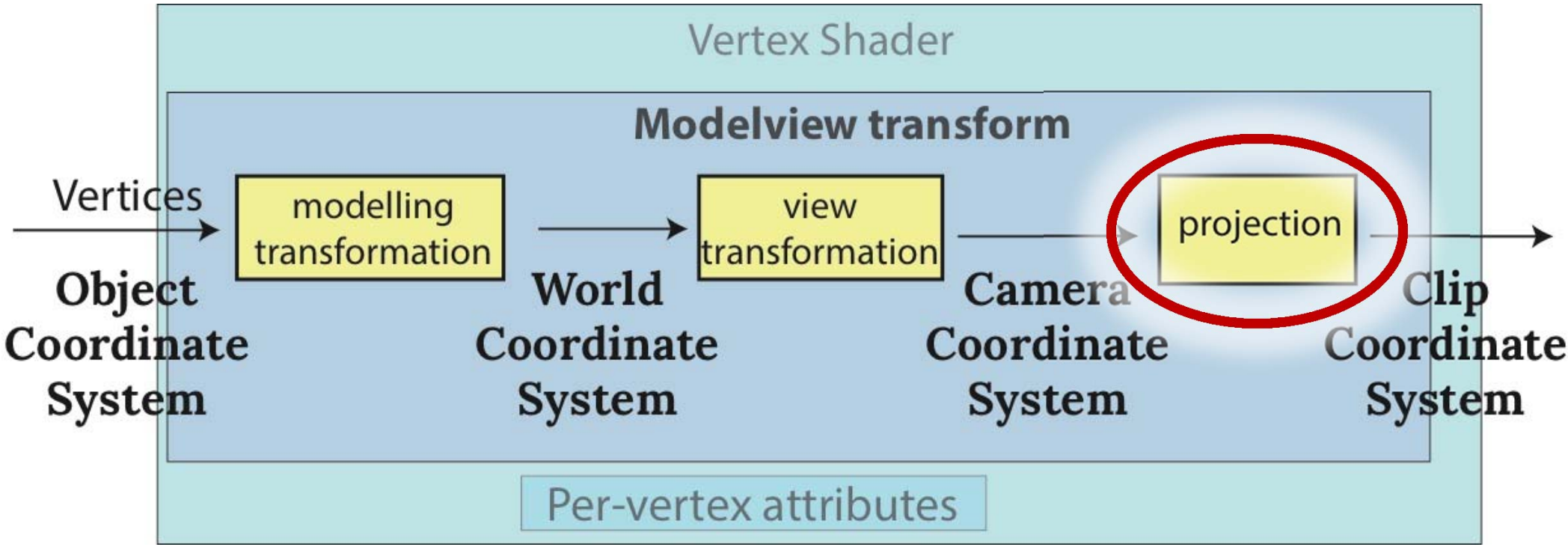
MOVING THE CAMERA OR THE WORLD?

- two equivalent operations
  - move camera one way vs. move world other way
- example
  - initial OpenGL camera: at origin, looking along -z axis
  - create a unit square parallel to camera at z = -10
  - translate in z by 3 possible in two ways
    - camera moves to z = -3
      - Note OpenGL models viewing in left-hand coordinates
    - camera stays put, but square moves to -7
  - resulting image same either way
    - possible difference: are lights specified in world or view coordinates?

VERTEX SHADER: CLOSER LOOK



VERTEX SHADER: CLOSER LOOK



PROJECTIONS MATTER.

PROJECTION

HOW TO CONVERT 3D INTO 2D?

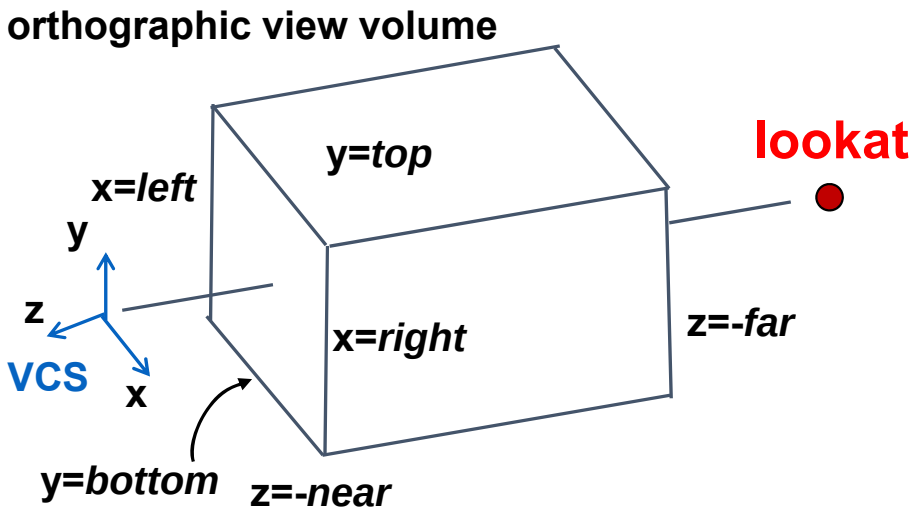
- Now the whole scene is in camera coordinates (homogeneous)
- How should we get 2D homogeneous coordinates?
- One way: just get rid of Z

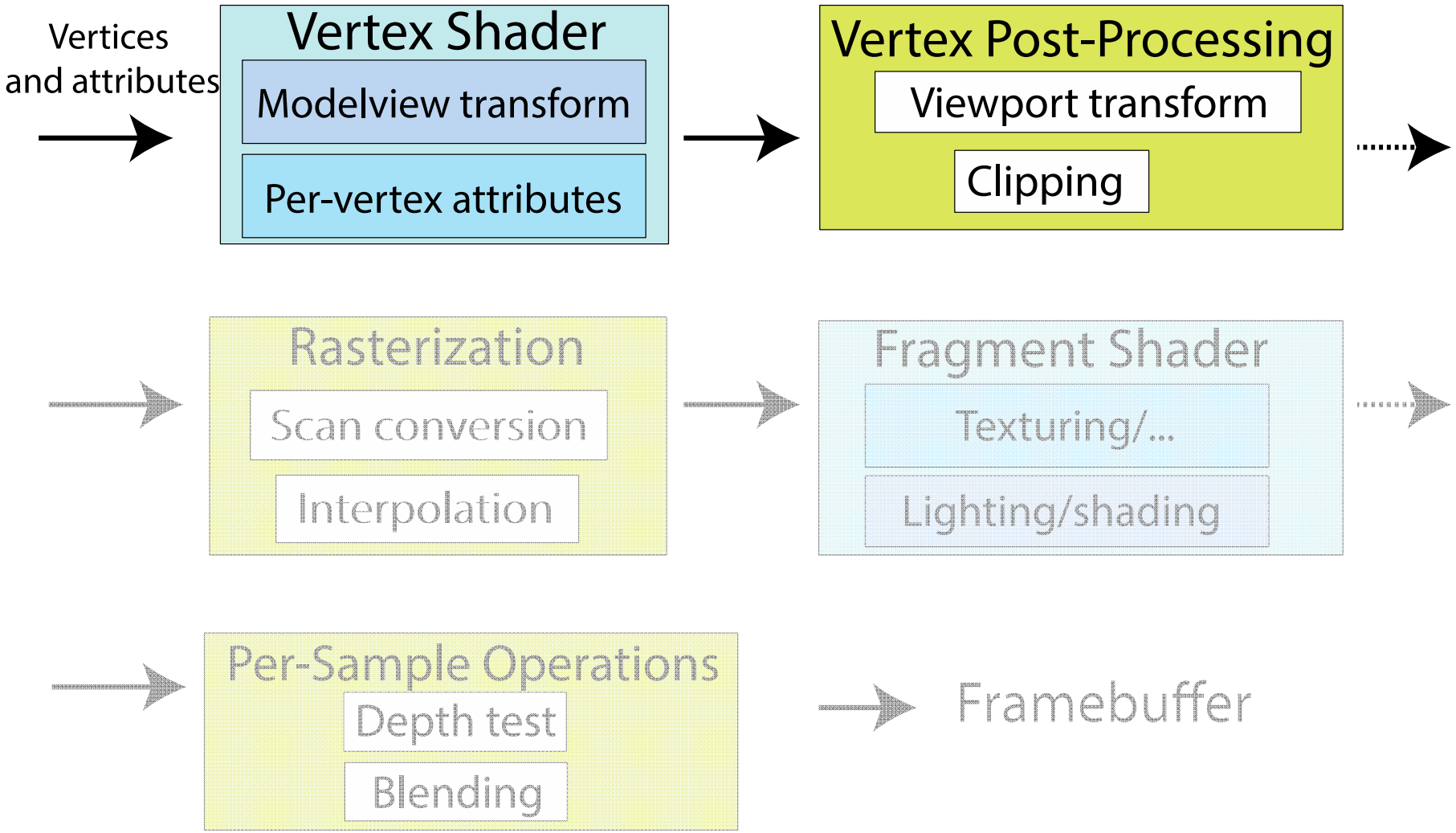
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

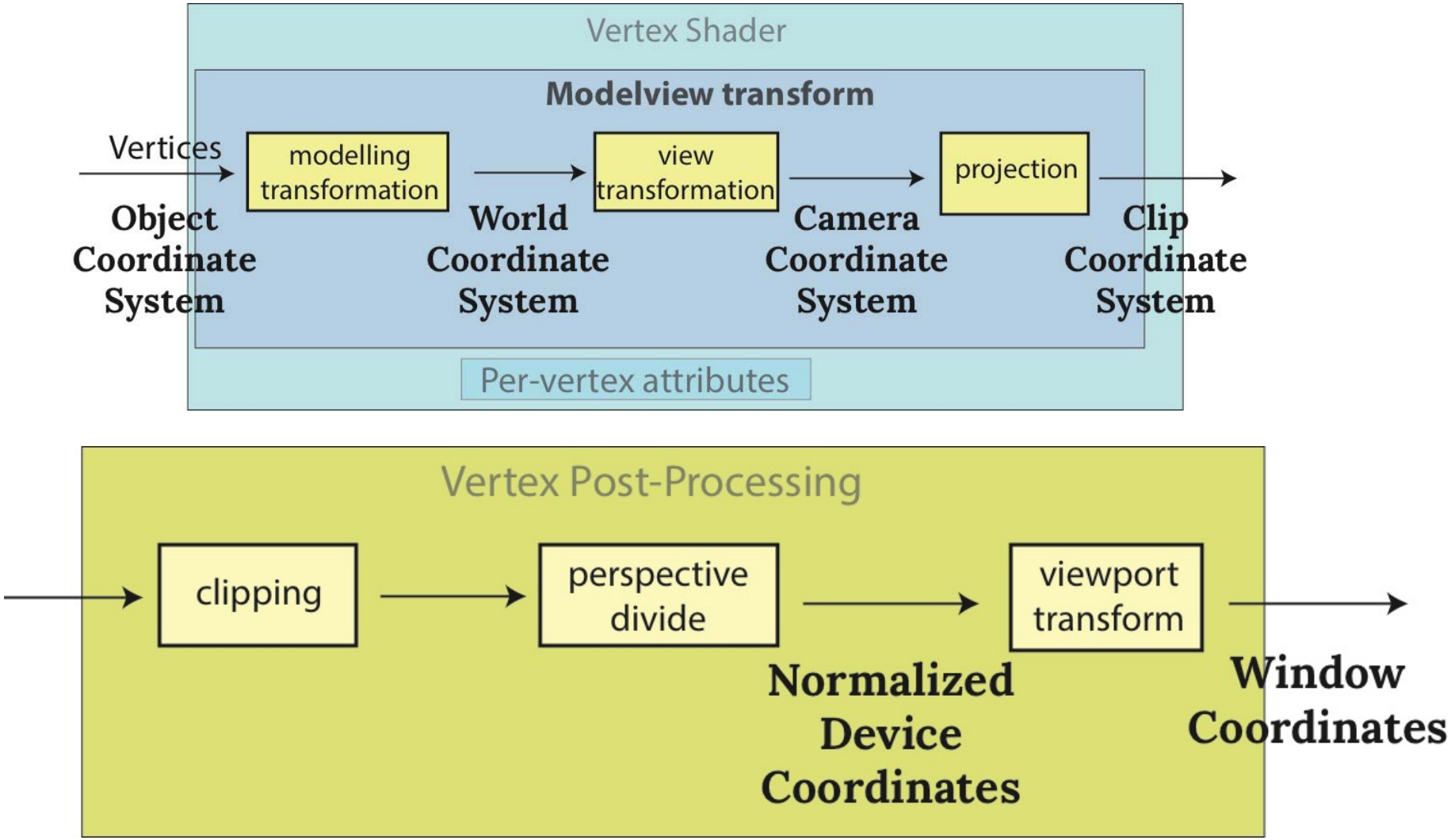
Issues?

VIEW VOLUME
ORTHOGRAPHIC PROJECTION

- specifies field-of-view, used for clipping
- restricts domain of z stored for visibility test

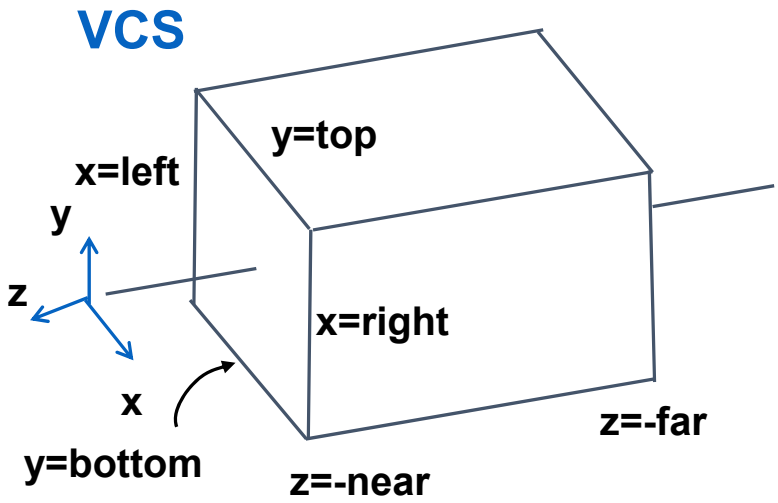


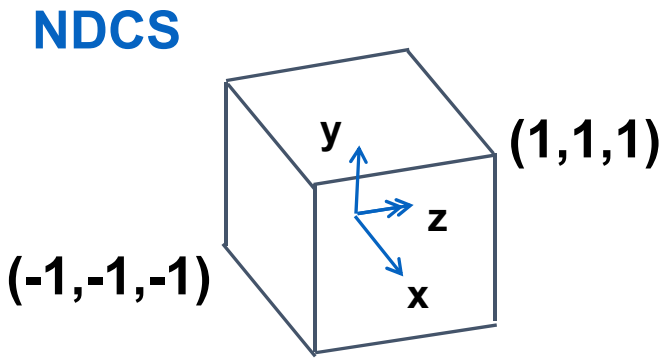

Vertices
and attributes
Vertex Shader
Modelview transform
Per-vertex attributes
Vertex Post-Processing
Viewport transform
Clipping
Rasterization
Scan conversion
Interpolation
Fragment Shader
Texturing/...
Lighting/shading
Per-Sample Operations
Depth test
Blending
Framebuffer


Vertex Shader
Modelview transform
Vertices
modelling transformation
view transformation
projection
Object Coordinate System
World Coordinate System
Camera Coordinate System
Clip Coordinate System
Per-vertex attributes
Vertex Post-Processing
clipping
perspective divide
viewport transform
Normalized Device Coordinates
Window Coordinates

Z-AXIS

- z axis flip changes coord system handedness
  - RHS before projection (eye/view coords)
  - LHS after projection (clip, norm device coords)





UNDERSTANDING Z

- why near and far plane?
  - near plane:
    - avoid singularity for perspective (division by zero, or very small numbers)
  - far plane:
    - store depth in fixed-point representation (integer), thus have to have fixed range of values (0...1)
    - avoid/reduce numerical precision artifacts for distant objects

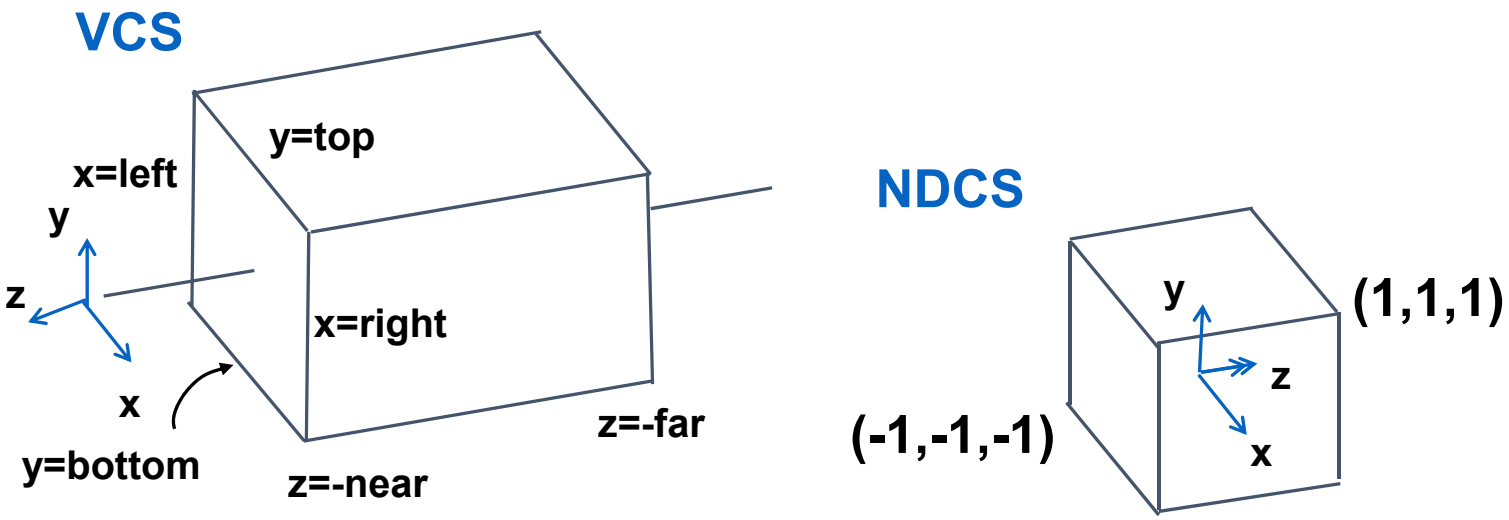
ORTHOGRAPHIC DERIVATION

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b$$

$$top \rightarrow 1$$

$$bottom \rightarrow -1$$



ORTHOGRAPHIC DERIVATION

• scale, translate, reflect for new coord sys

$$y' = a \cdot y + b$$

$$top \rightarrow 1$$

$$bottom \rightarrow -1$$

•Solve two eq.

$$1 = a \cdot top + b$$

$$-1 = a \cdot bottom + b$$

$$a = \frac{2}{top - bottom}$$

$$b = \frac{-top - bottom}{top - bottom}$$

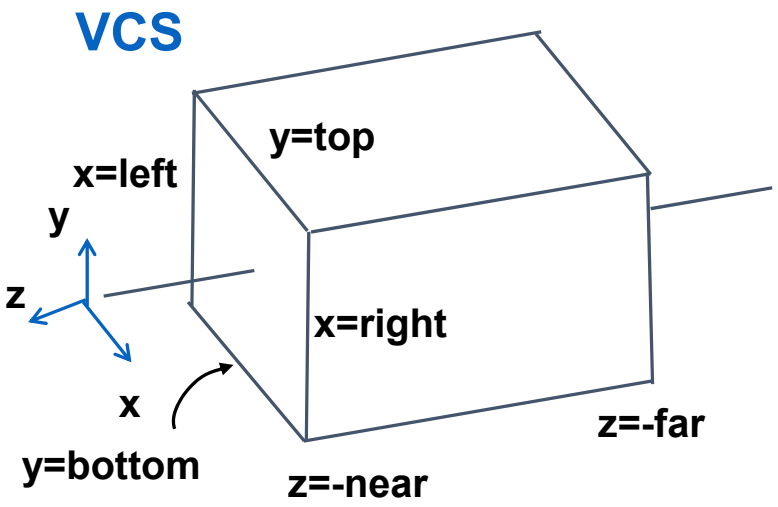
ORTHOGRAPHIC DERIVATION

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b$$

$$y = top \rightarrow y' = 1$$

$$y = bot \rightarrow y' = -1$$



$$a = \frac{2}{top - bot}$$

$$b = -\frac{top + bot}{top - bot}$$

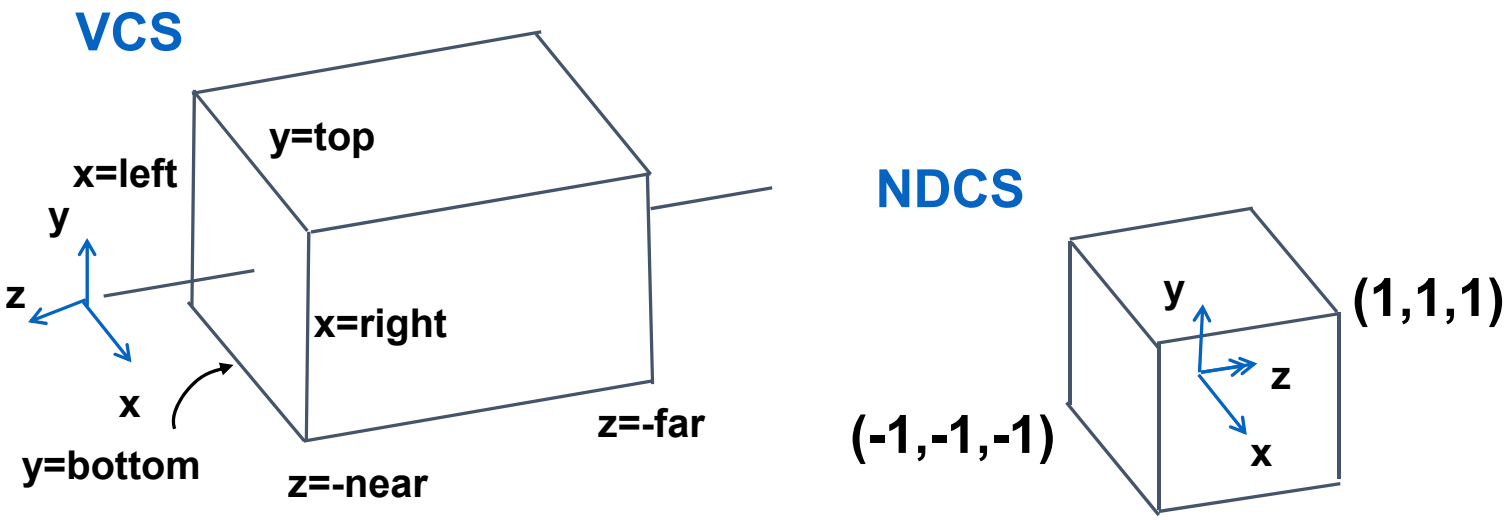
same idea for right/left, far/near

ORTHOGRAPHIC DERIVATION

- scale, translate, reflect for new coord sys

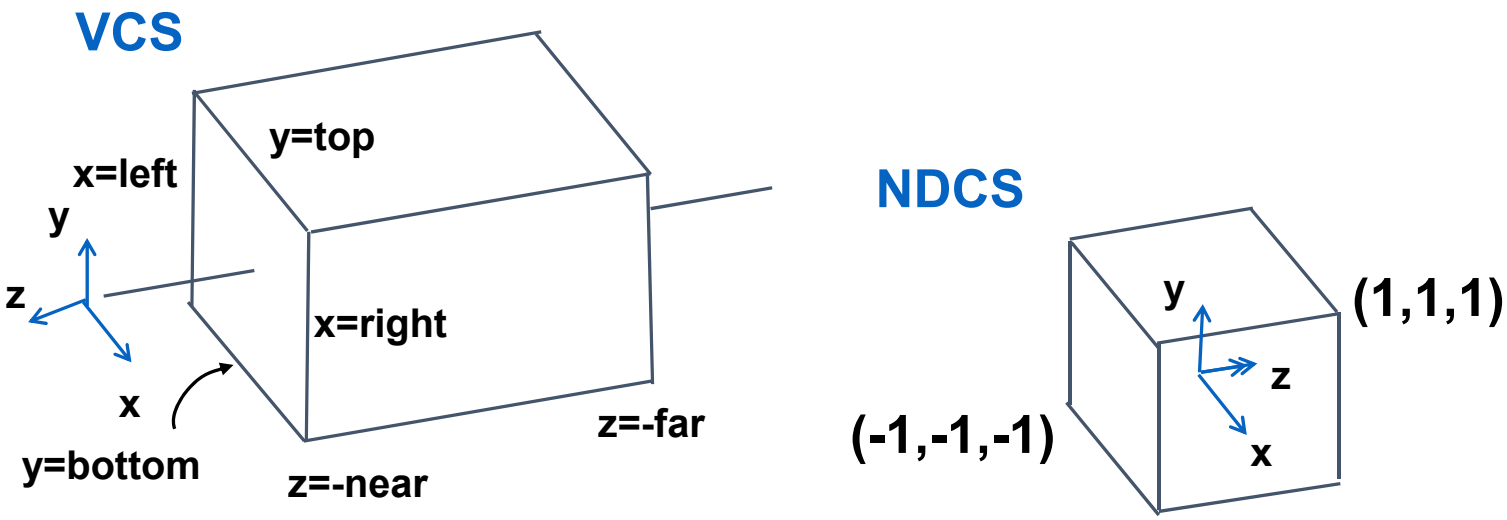
$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

… OR A BETTER WAY



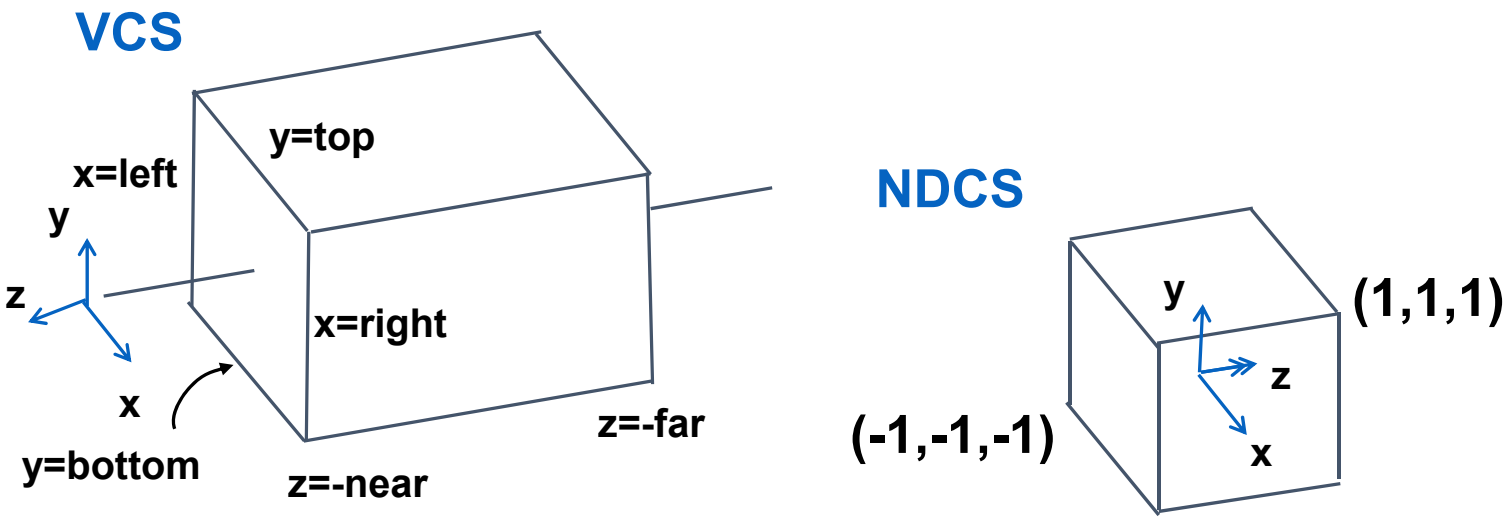
… OR A BETTER WAY

- Vertices should be first translated, then scaled!



… OR A BETTER WAY

- $M = Scale\left(\frac{2}{r-l}, \frac{2}{t-b}, \frac{-2}{f-n}\right) \cdot Tr(-\frac{r+l}{2}, -\frac{t+b}{2}, \frac{f+n}{2})$



ORTHOGRAPHIC DERIVATION

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

ORTHOGRAPHIC DERIVATION

- scale, translate, reflect for new coord sys

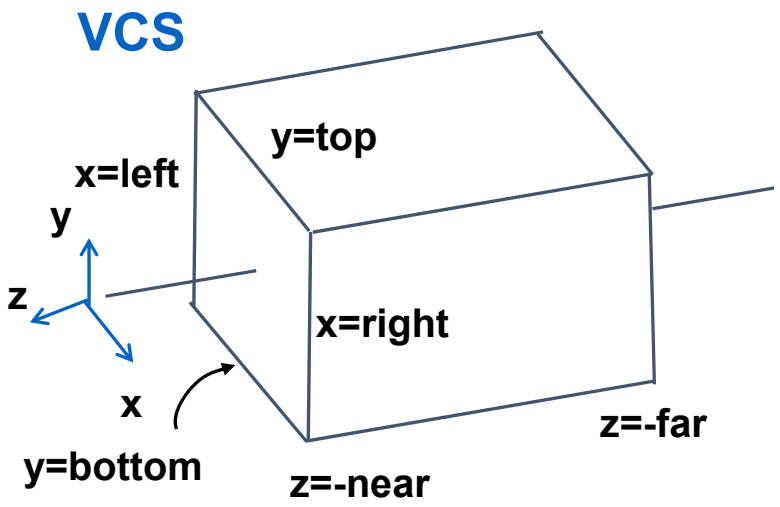
$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

ORTHOGRAPHIC DERIVATION

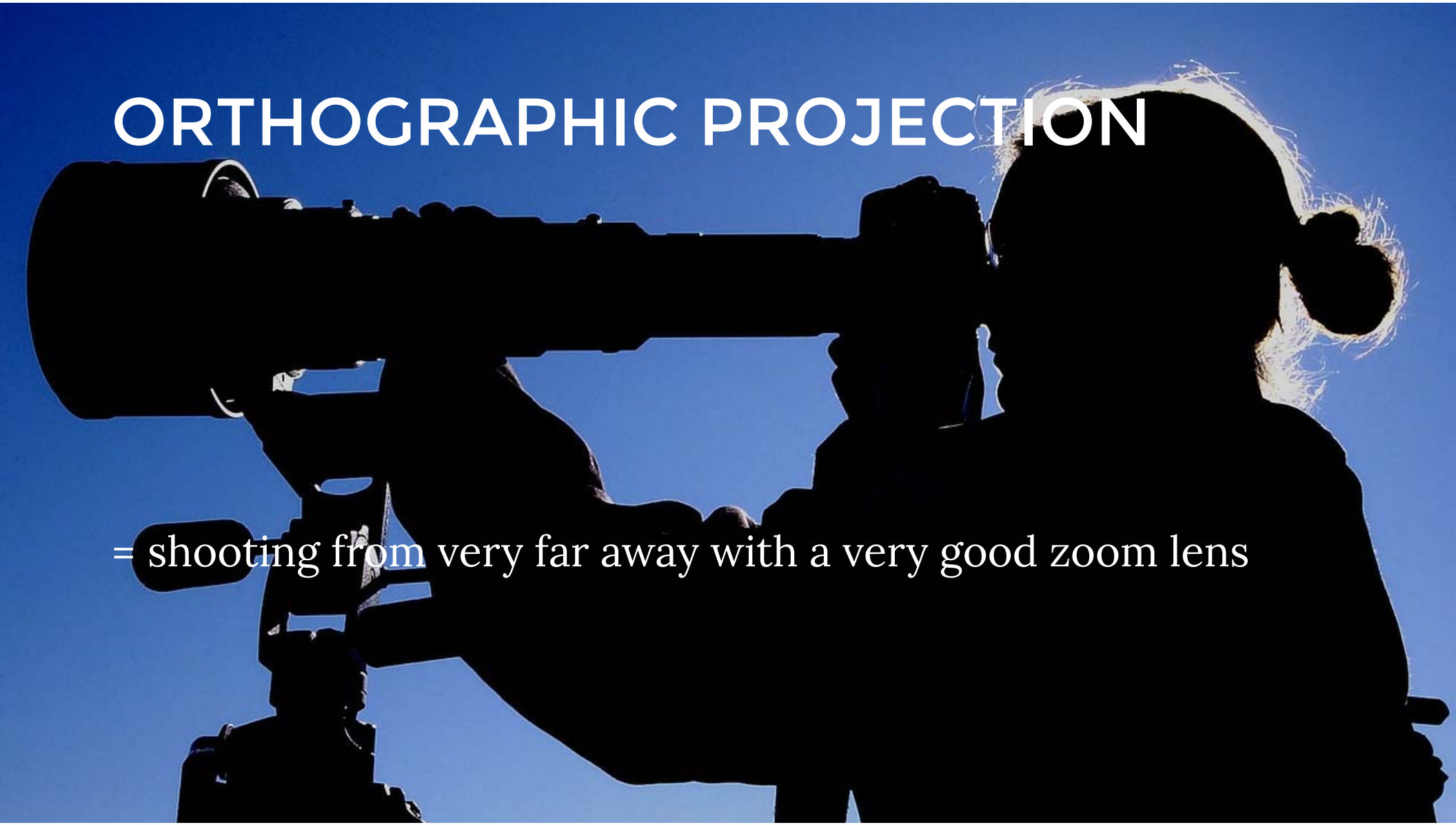
- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

WHAT HAPPENS IF CAMERA MOVES FURTHER BACK?



ORTHOGRAPHIC PROJECTION

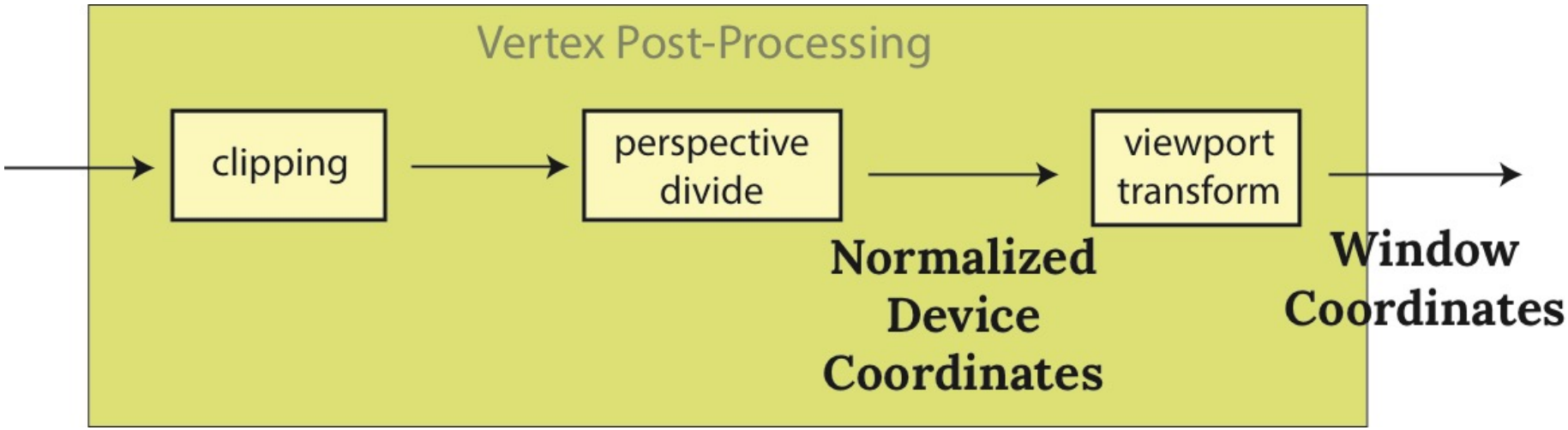


= shooting from very far away with a very good zoom lens

THREE.JS

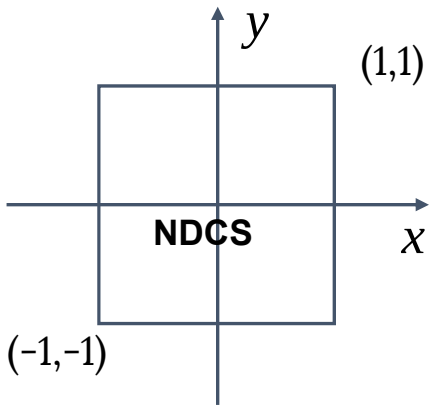
```
var camera = new THREE.OrthographicCamera( width / - 2,
width / 2, height / 2, height / - 2, 1, 1000 );

scene.add( camera );
```

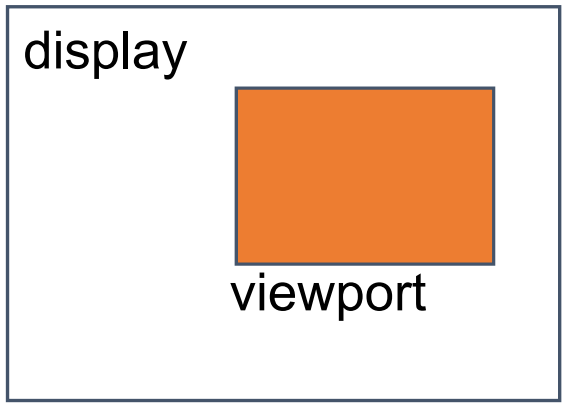
VIEWPORT TRANSFORM

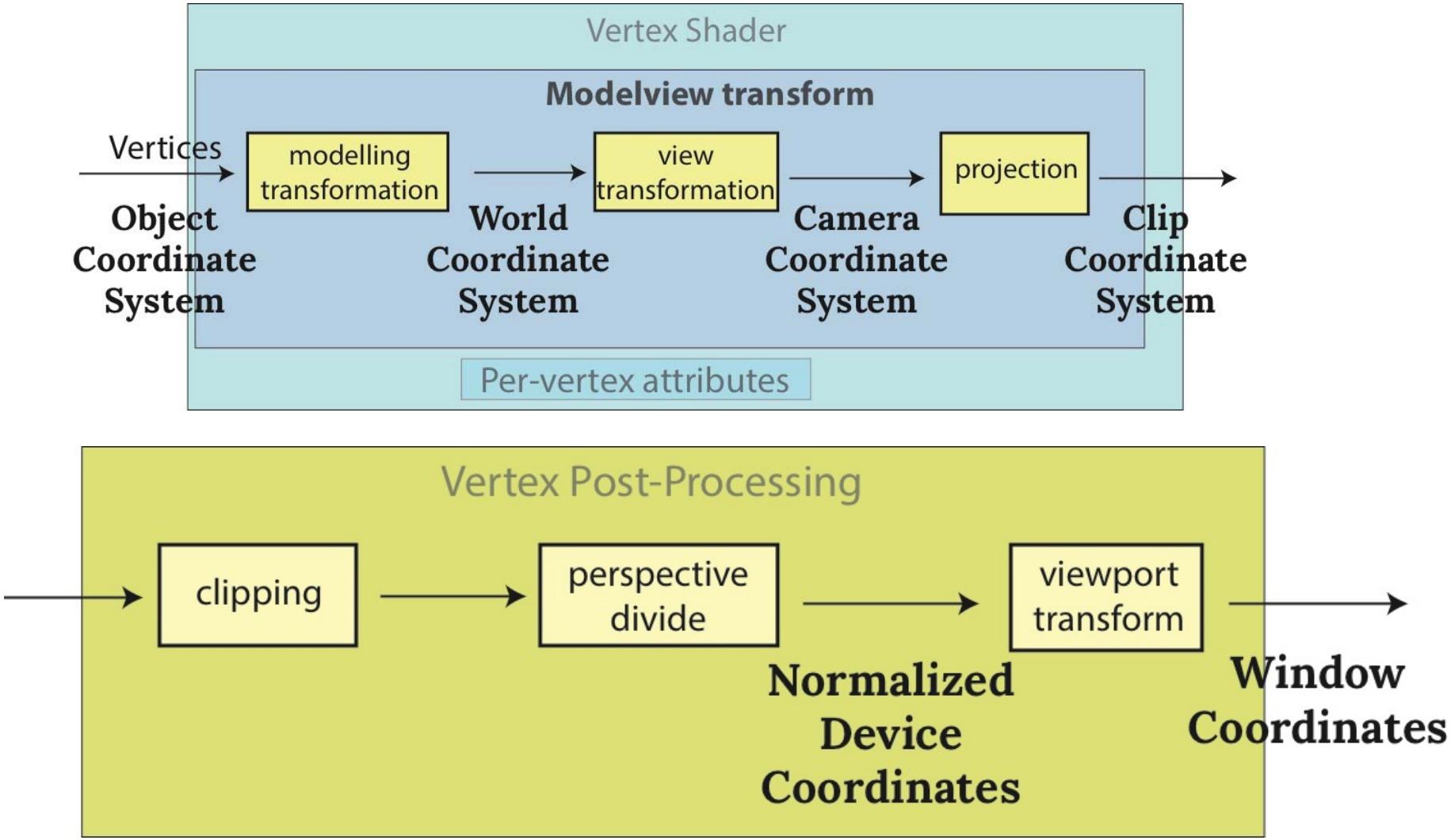


(AUTO) VIEWPORT TRANSFORM

- Converts Normalized Device Coordinates into Window coordinates
- I.e. this:

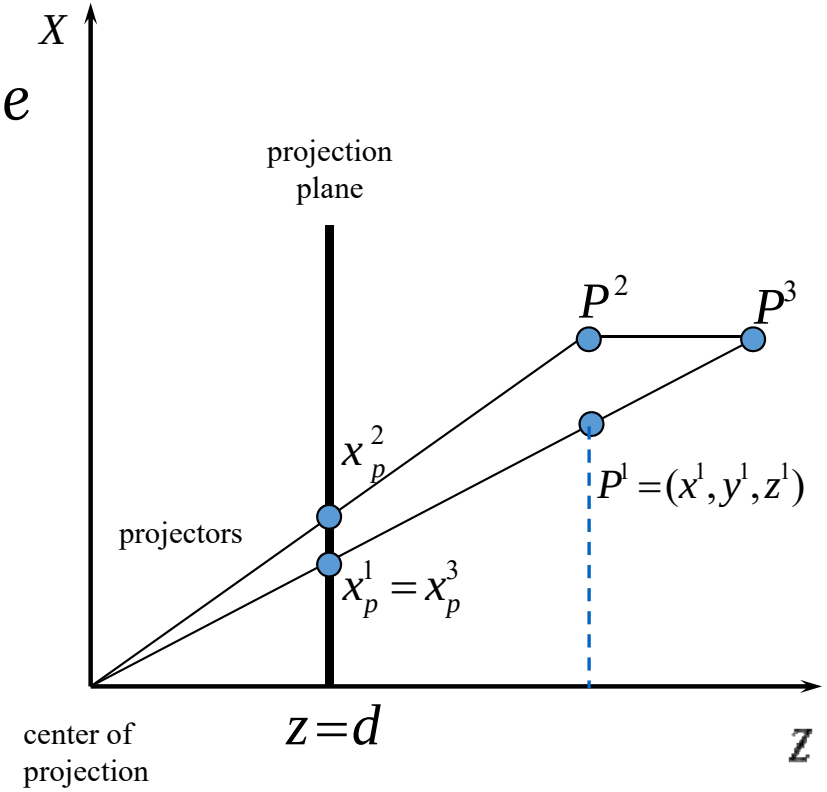





Vertex Shader
Modelview transform
Vertices
modelling transformation
view transformation
projection
Object Coordinate System
World Coordinate System
Camera Coordinate System
Clip Coordinate System
Per-vertex attributes
Vertex Post-Processing
clipping
perspective divide
viewport transform
Normalized Device Coordinates
Window Coordinates

PINHOLE CAMERA (PERSPECTIVE)

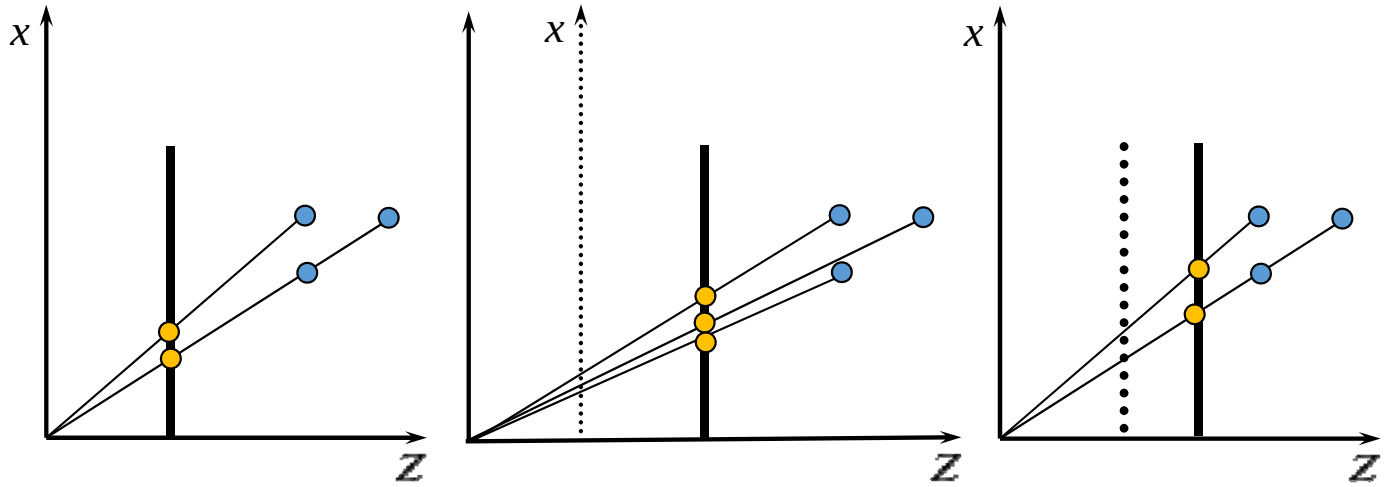
- Viewing from *point at finite distance*
- Without loss of generality:
  - Viewpoint at origin
  - Viewing plane is $z=d$
- Given P=(x,y,z) triangle similarity gives:



$$\frac{x}{z}=\frac{x_p}{d} \text{ and } \frac{y}{z}=\frac{y_p}{d} \Rightarrow x_p=\frac{xd}{z} \text{ and } y_p=\frac{yd}{z}$$

PERSPECTIVE PROJECTION (CONT'D)

- What is (if any) is the difference between:
  - Moving projection plane
  - Moving viewpoint (center of projection)?



PERSPECTIVE PROJECTION

- In matrix notation with homogeneous coordinates:

$$P(x, y, z, 1) = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} xd \\ yd \\ z \\ z \end{pmatrix}$$

- In Euclidean coordinates:

$$\left(\frac{xd}{z}, \frac{yd}{z}, \frac{z}{z} \right) = \left(\frac{xd}{z}, \frac{yd}{z}, 1 \right) = (x_p, y_p, 1).$$

- P singular: $det(P)=0$

SINGULAR PROJECTION MATRIX

- We can't invert it!
- We can't do depth test
  - All the z's are the same now
- Similar to that first orthographic projection matrix that we had:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

PERSPECTIVE PROJECTION

$$P(x,y,z,1) = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} xd \\ yd \\ z \\ z \end{pmatrix}$$

$$\left(\frac{xd}{z}, \frac{yd}{z}, \frac{z}{z}\right) = (z_p, y_p, 1).$$

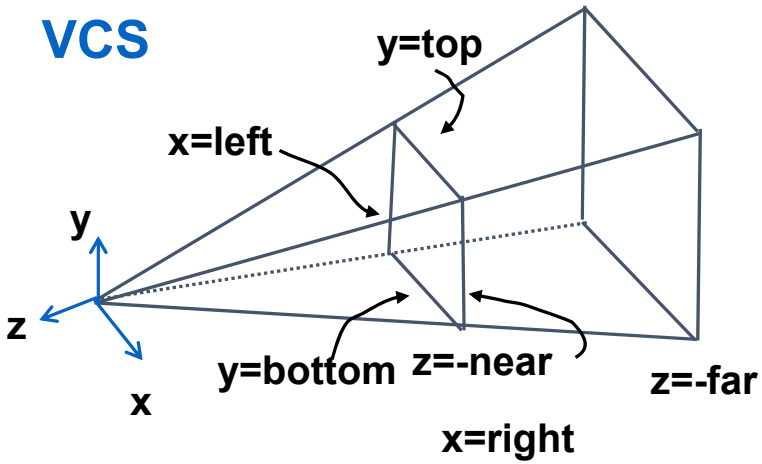
- Keeping track of z:

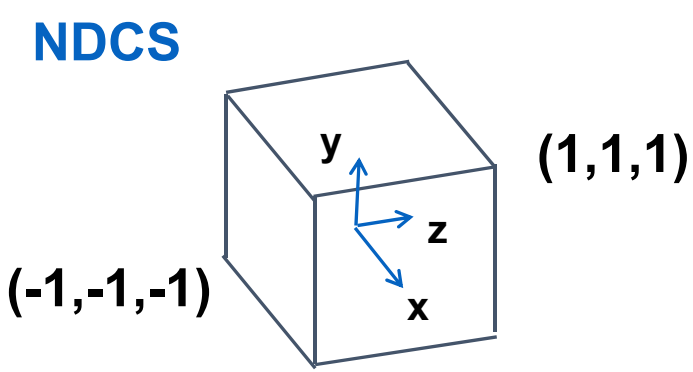
$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ z-1 \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} dx/z \\ dy/z \\ 1-1/z \end{bmatrix}$$

  - NEW Z is monotonic increasing function of old

OPENGL PERSPECTIVE DERIVATION

Map FRUSTUM to the NDCS cube.
Now we need to scale/translate/shear -> generic transform





(REMINDER) ORTHOGRAPHIC PROJECTION

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

PERSPECTIVE PROJECTION

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

DERIVATION AS LINEAR SYSTEM

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$x' = Ex + Az$$
$$y' = Fy + Bz$$
$$z' = Cz + D$$
$$w' = -z$$

$$x = left \rightarrow x'/w' = 1$$
$$x = right \rightarrow x'/w' = -1$$
$$y = top \rightarrow y'/w' = 1$$
$$y = bottom \rightarrow y'/w' = -1$$
$$z = -near \rightarrow z'/w' = 1$$
$$z = -far \rightarrow z'/w' = -1$$

$$y' = Fy + Bz, \quad \frac{y'}{w'} = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{-z},$$

$$1 = F\frac{y}{-z} + B\frac{z}{-z}, \quad 1 = F\frac{y}{-z} - B, \quad 1 = F\frac{top}{-(-near)} - B,$$

$$1 = F\frac{top}{near} - B$$

PERSPECTIVE EXAMPLE

view volume
- left = -1, right = 1
- bot = -1, top = 1
- near = 1, far = 4

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5/3 & -8/3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

PERSPECTIVE EXAMPLE

$$\begin{bmatrix} 1 \\ -1 \\ -5z_{VCS}/3 - 8/3 \\ -z_{VCS} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -5/3 & -8/3 \\ & & -1 & \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ z_{VCS} \\ 1 \end{bmatrix}$$

/ w

$$x_{NDCS} = -1/z_{VCS}$$

$$y_{NDCS} = 1/z_{VCS}$$

$$z_{NDCS} = \frac{5}{3} + \frac{8}{3z_{VCS}}$$