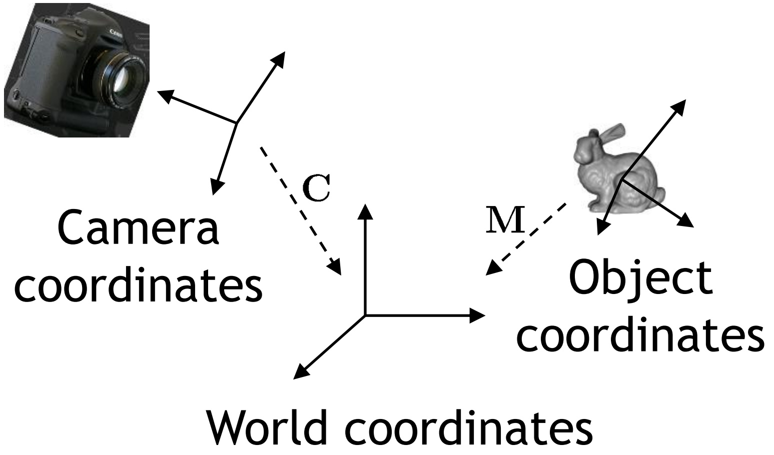
CMSC427
Transformations II: Frustrum

Credit: some slides from Dr. Zwicker

Viewing transformations: the virtual camera

Need to know

- Where is the camera?
  - CAMERA TRANSFORM
- What lens does it have?
  - PROJECTIVE TRANSFORM

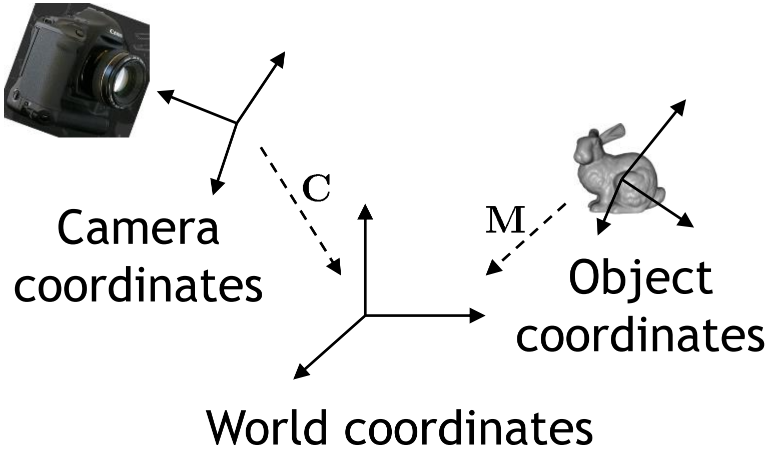


Today

- Rendering pipeline
- Projections
- View volumes, clipping
- Viewport transformation

Virtual camera routines in Processing

- Camera (where)
- [beginCamera()]
- [camera()]
- [endCamera()]



- Projective (length of lens)
- [frustum()]
- [ortho()]
- [perspective()]

- Tracing
- [printCamera()]
- [printProjection()]

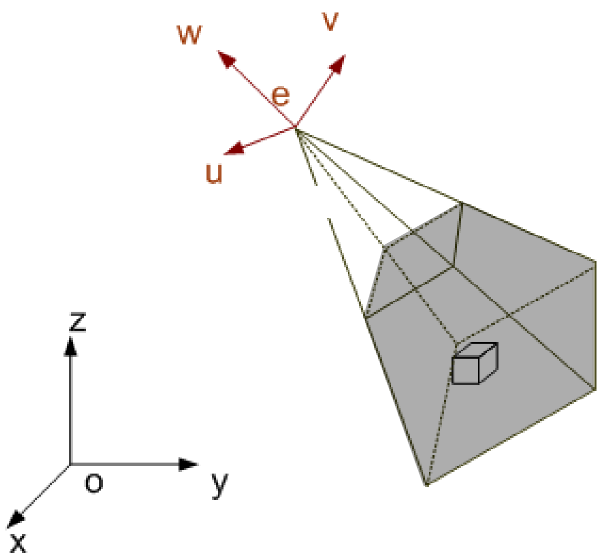
Camera routine in Processing

```
void setup() {
  size(640, 360, P3D);
  }

void draw() {
  background(0);

  camera(width/2, height/2, (height/2) / tan(PI/6),
              width/2, height/2, 0,    0, 1, 0);

  translate(width/2, height/2, -100);
  stroke(255);
  noFill();
  box(200);
}
```

- View volume is 3D volume seen by camera

Perspective view volume

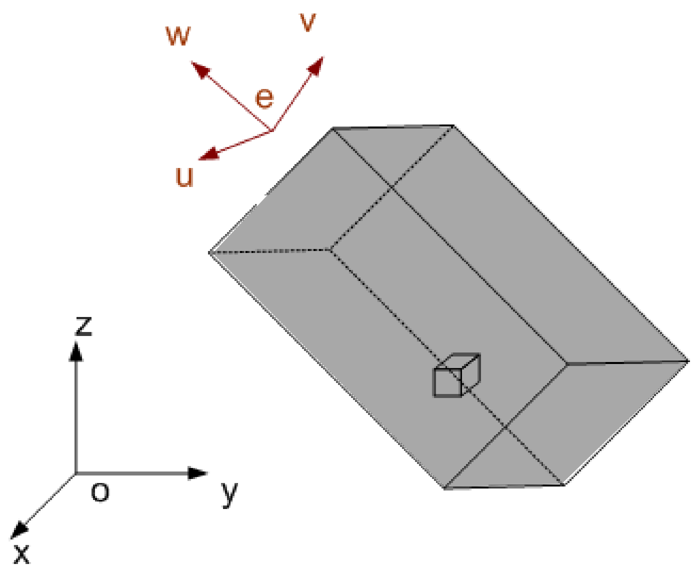
Camera coordinates



World coordinates

Orthographic view volume

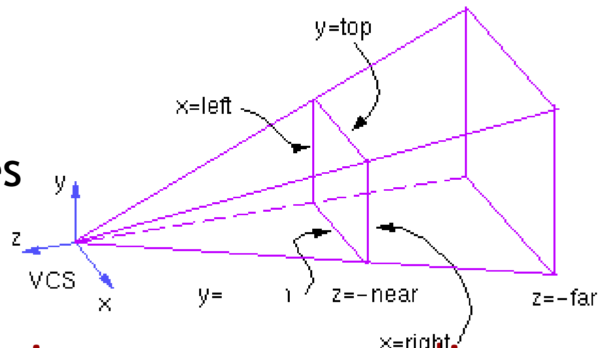
Camera coordinates



World coordinates

General view volume

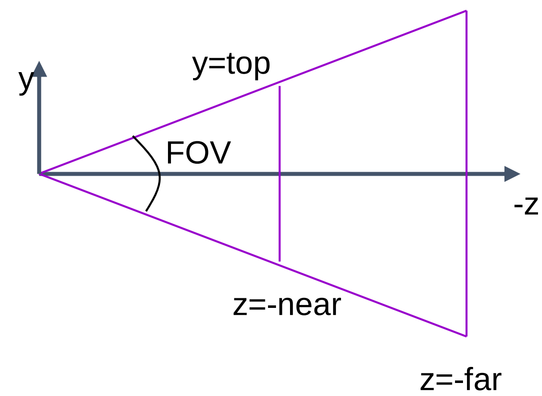
Camera coordinates



- Defined by 6 parameters, in camera coordinates
  - Left, right, top, bottom boundaries
  - Near, far clipping planes
- Clipping planes to avoid numerical problems
  - Divide by zero
  - Low precision for distant objects
- Often symmetric, i.e., left=-right, top=-bottom

Perspective view volume

Symmetric view volume

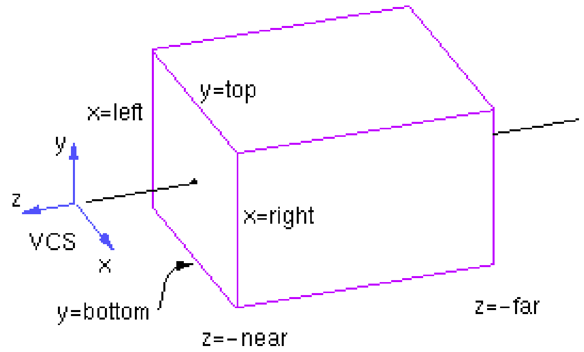


- Only 4 parameters
  - Vertical field of view (FOV)
  - Image aspect ratio (width/height)
  - Near, far clipping planes

$$\text{aspect ratio} = \frac{right - left}{top - bottom} = \frac{right}{top}$$

$$\tan(FOV/2) = \frac{top}{near}$$

Orthographic view volume

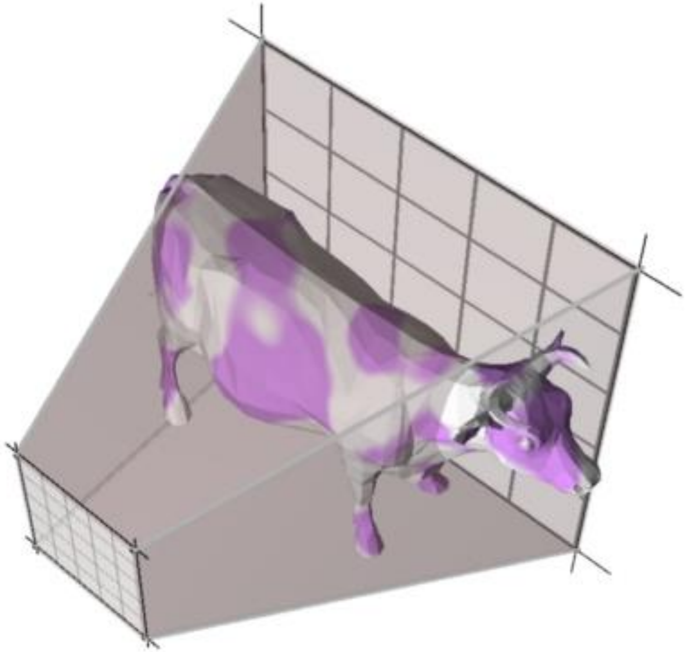


Camera
coordinates

- Parametrized by 6 parameters
  - Right, left, top, bottom, near, far
- If symmetric
  - Width, height, near, far

Clipping

- Need to identify objects outside view volume
  - Avoid division by zero
  - Efficiency, don't draw objects outside view volume
- Performed by OpenGL rendering pipeline
- Clipping always to canonic view volume
  - Cube [-1..1]x[-1..1]x[-1..1] cent



- Need to transform desired view frustum to canonic view frustum

Canonic view volume

- Projection matrix is set such that
  - User defined view volume is transformed into canonic view volume, i.e., unit cube [-1,1]x[-1,1]x[-1,1]

    “Multiplying vertices of view volume by projection matrix and performing homogeneous divide yields canonic view volume, i.e., cube [-1,1]x[-1,1]x[-1,1]“

- Perspective and orthographic projection are treated exactly the same way

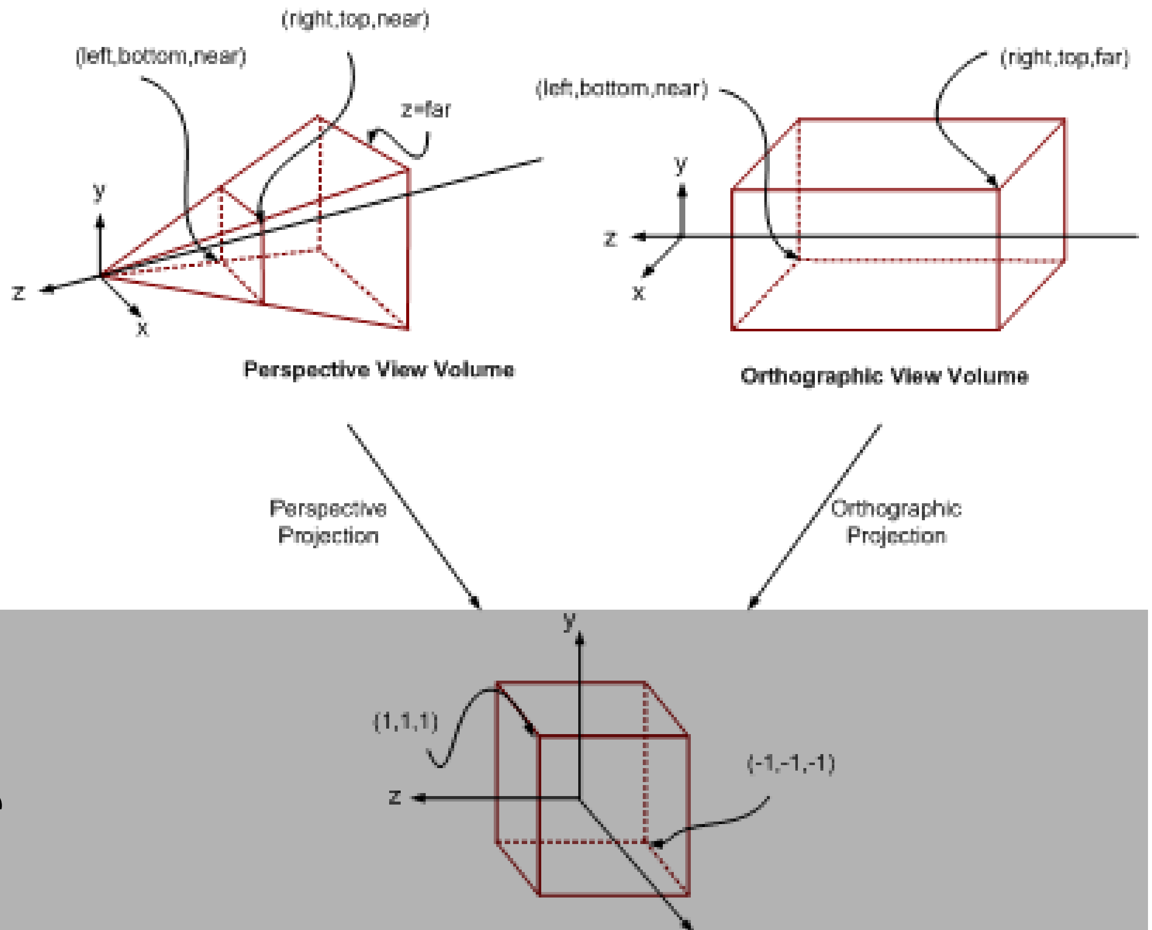
Projection matrix

Camera coordinates

Projection matrix

Canonic view volume

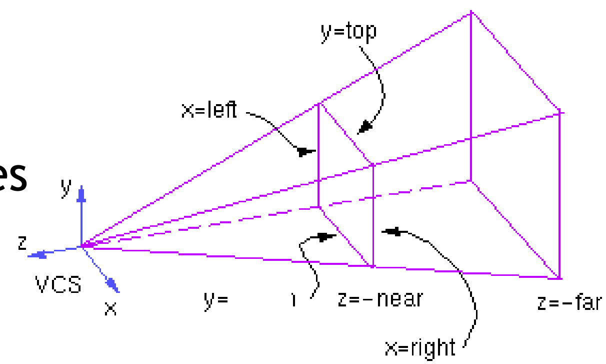
Viewport transformation (later)



Perspective projection matrix

- General view frustum

Camera coordinates



$$\mathbf{P}_{persp}(left, right, top, bottom, near, far) =$$

$$\begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \frac{-(far+near)}{far-near} & \frac{-2far \cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective projection matrix

- Compare to simple projection matrix from before

Simple projection

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

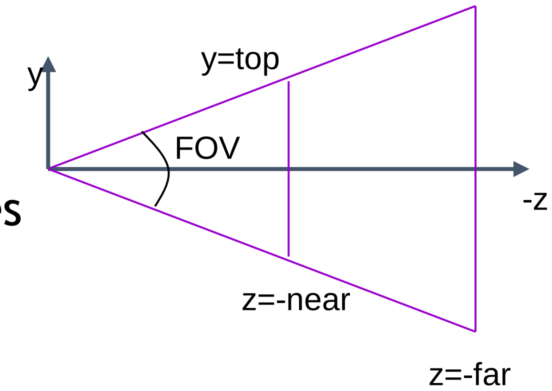
General view frustum

$$\begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \frac{-(far+near)}{far-near} & \frac{-2far \cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective projection matrix

- Symmetric view frustum with field of view, aspect ratio, near and far clip planes

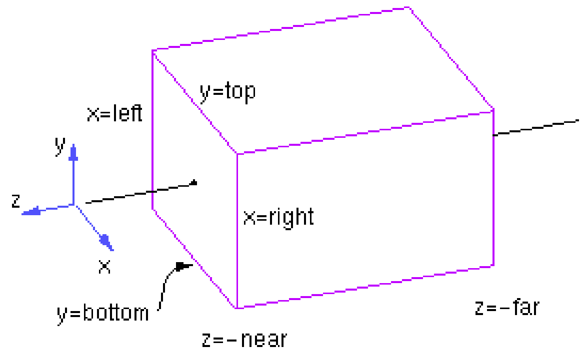
Camera coordinates



$$\mathbf{P}_{persp}(FOV, aspect, near, far) = \begin{bmatrix} \frac{1}{aspect \cdot \tan(FOV/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(FOV/2)} & 0 & 0 \\ 0 & 0 & \frac{near+far}{near-far} & \frac{2 \cdot near \cdot far}{near-far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Orthographic projection matrix

Camera coordinates



$$\mathbf{P}_{ortho}(right, left, top, bottom, near, far) = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_{ortho}(width, height, near, far) = \begin{bmatrix} \frac{2}{width} & 0 & 0 & 0 \\ 0 & \frac{2}{height} & 0 & 0 \\ 0 & 0 & \frac{2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$w = 1$ after mult. with orthographic projection matrix