CSE4203: Computer Graphics

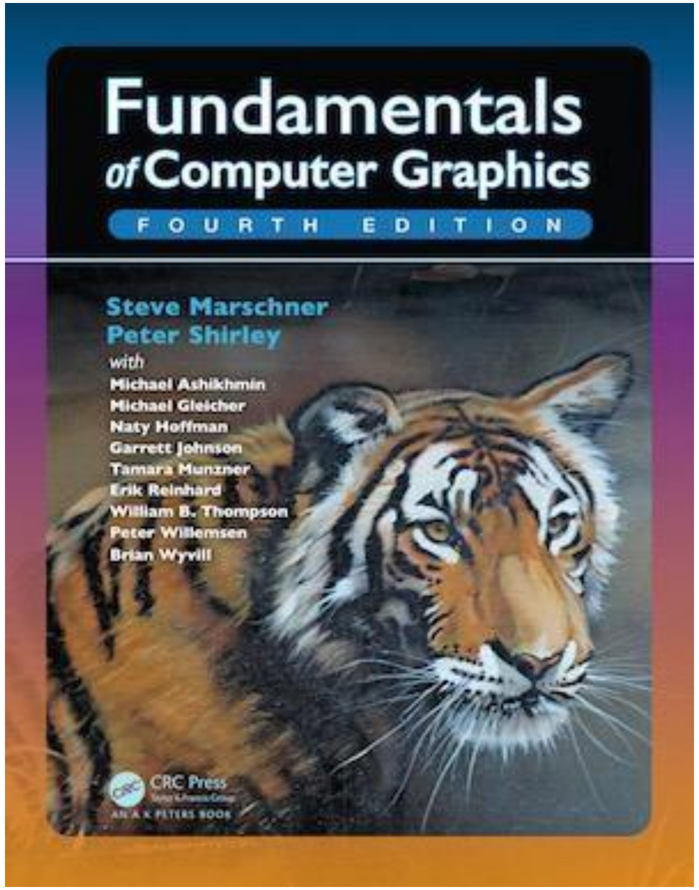
Chapter – 7 (part - A)

Viewing

Outline

– Image-order and object-render rendering
– Viewing transformation
– Viewport transformation
– Orthographic projection transformation

Credit



CS4620: Introduction to Computer Graphics

Cornell University

Instructor: Steve Marschner

http://www.cs.cornell.edu/courses/cs4620/2019fa/

Rendering Techniques (1/2)

- One of the basic tasks of computer graphics is rendering 3D objects:
  - taking a scene, or model, composed of many geometric objects arranged in 3D space
  - producing a 2D image that shows the objects as viewed
  - from a particular viewpoint.

Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Rendering Techniques (2/2)

1. Image-order rendering: iterate over the pixels in the image to be produced, rather than the elements in the scene to be rendered.
2. object-order rendering: that iterate over the elements in the scene to be rendered, rather than the pixels in the image to be produced.

Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Image-order Rendering (1/2)

- Image-order rendering:
  - ***Ray-tracing:***

    For each pixel is considered in turn,
    - All the objects that influence it are found
    - and the pixel value is computed.
      - in Chapter 4

Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Object-order Rendering (1/2)

- Object-order rendering:
  - ***Viewing Transformation:***

    For each object is considered in turn,

    All the pixels that it influences are found and updated

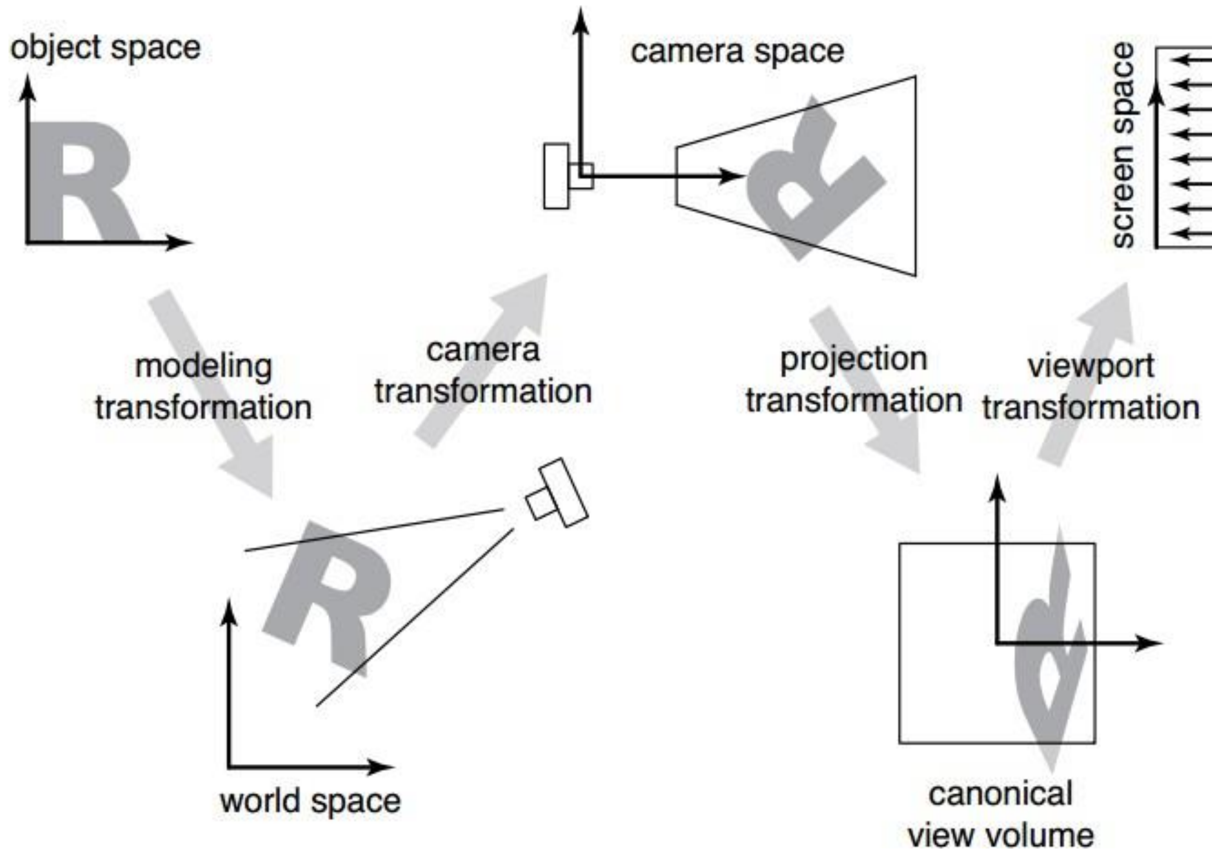
Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Object-order Rendering (2/2)

- **Viewing Transformation** (this chapter):
  - The inverse of the previous process.
  - How to use ***matrix transformations*** to express any parallel or perspective view.
  - These transformations:
    - Project 3D points in the scene (world space) to 2D points in the image (image space)

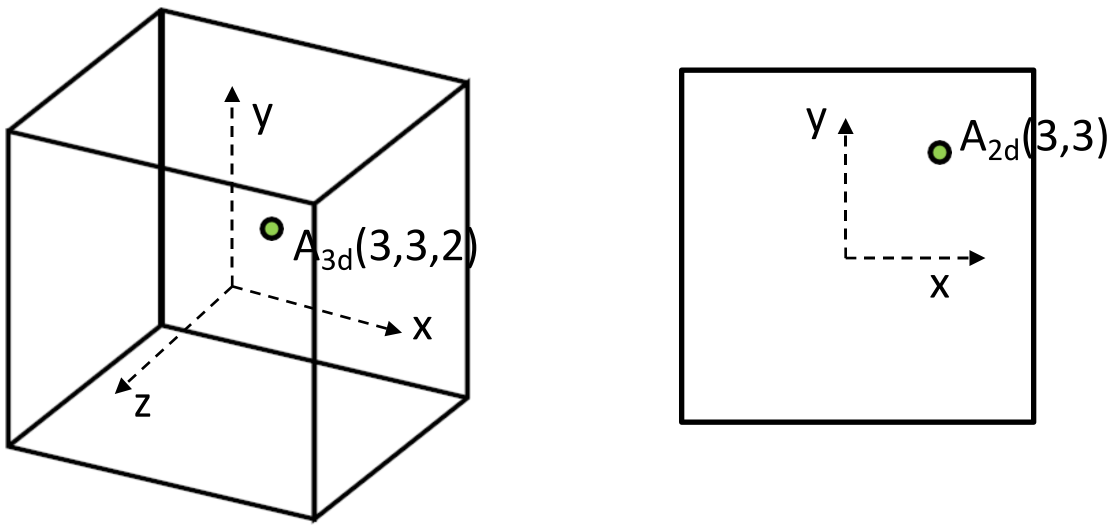
Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Viewing Transformation Sequences (1/1)



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

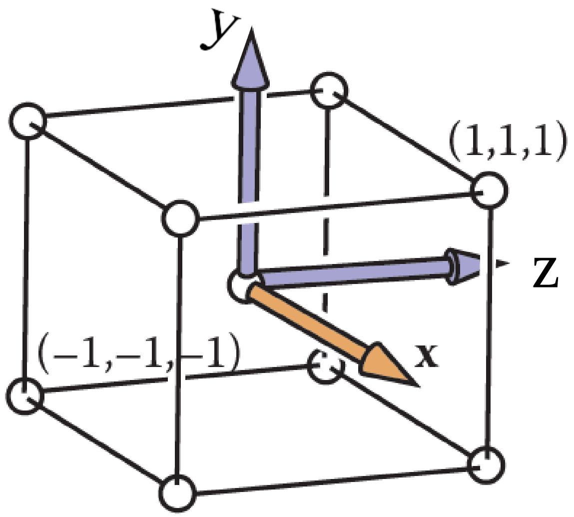
Viewport Transformation (1/19)



- Projection: Ignoring the *z*-coordinate

Viewport Transformation (2/19)

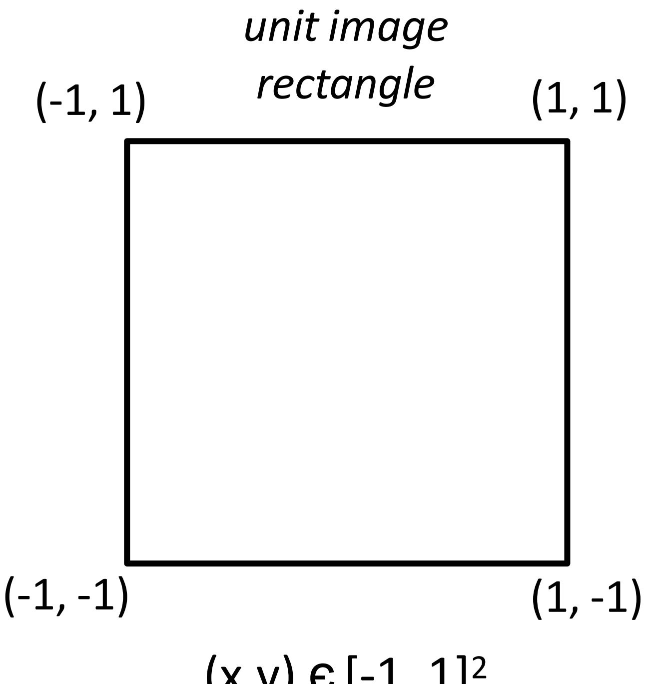
- Canonical View Volume: (x,y,z) є [-1, 1]^3
  - We will assume that the model to be drawn are completely inside the canonical view vol.

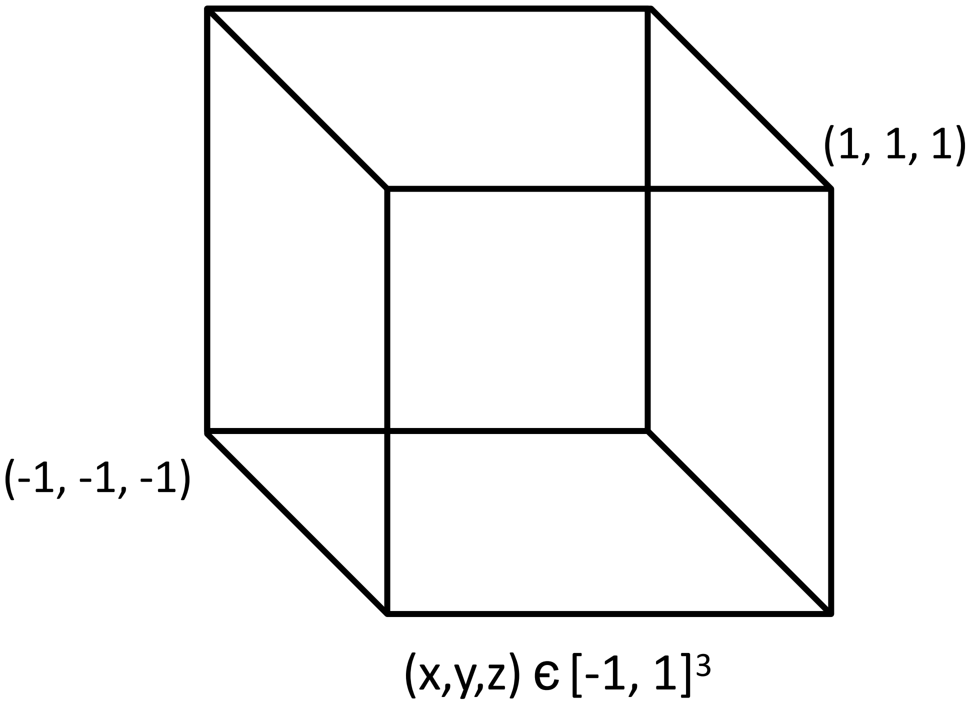


(x,y,z) є [-1, 1]^3

Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

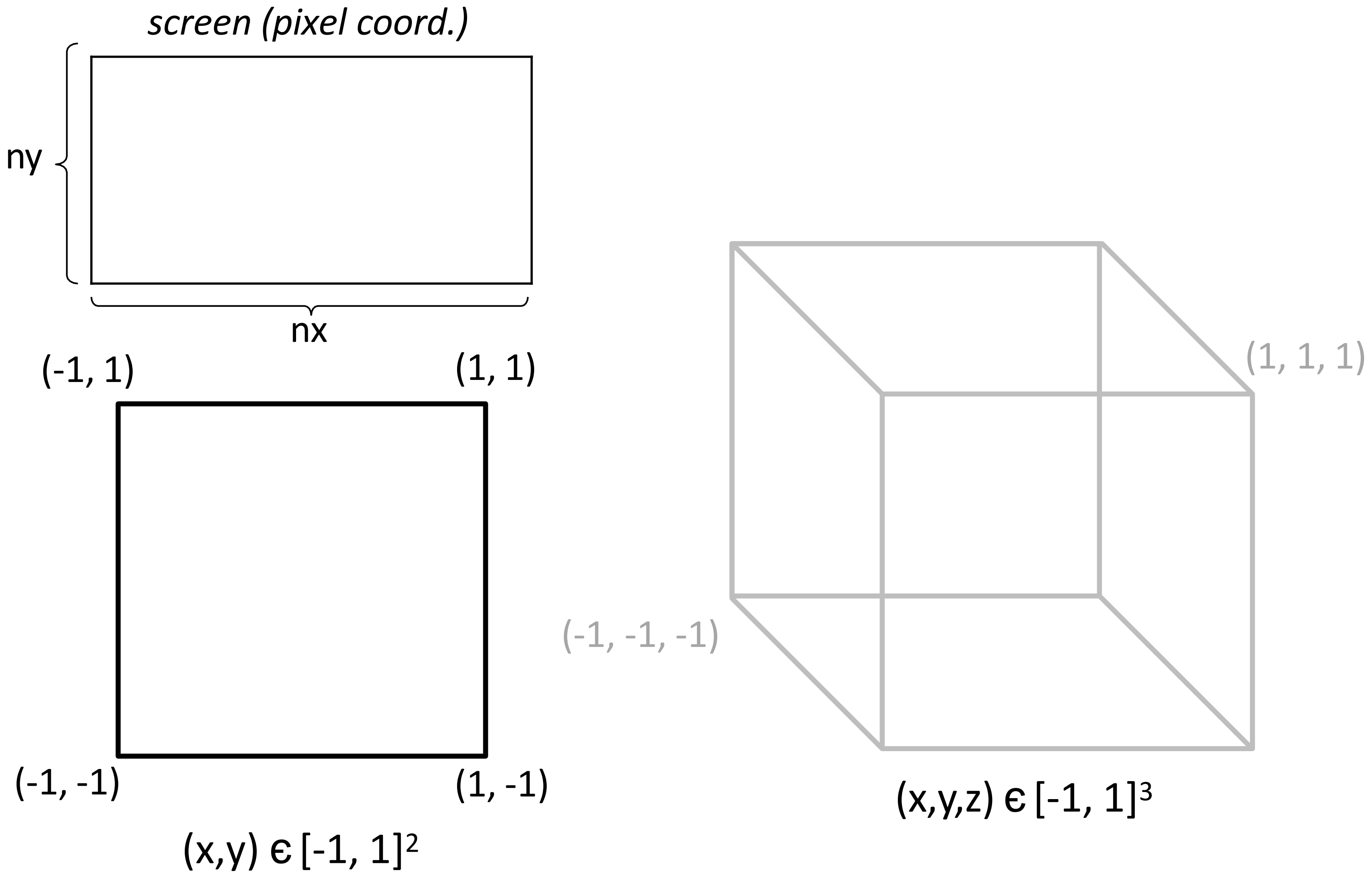
Viewport Transformation (4/19)





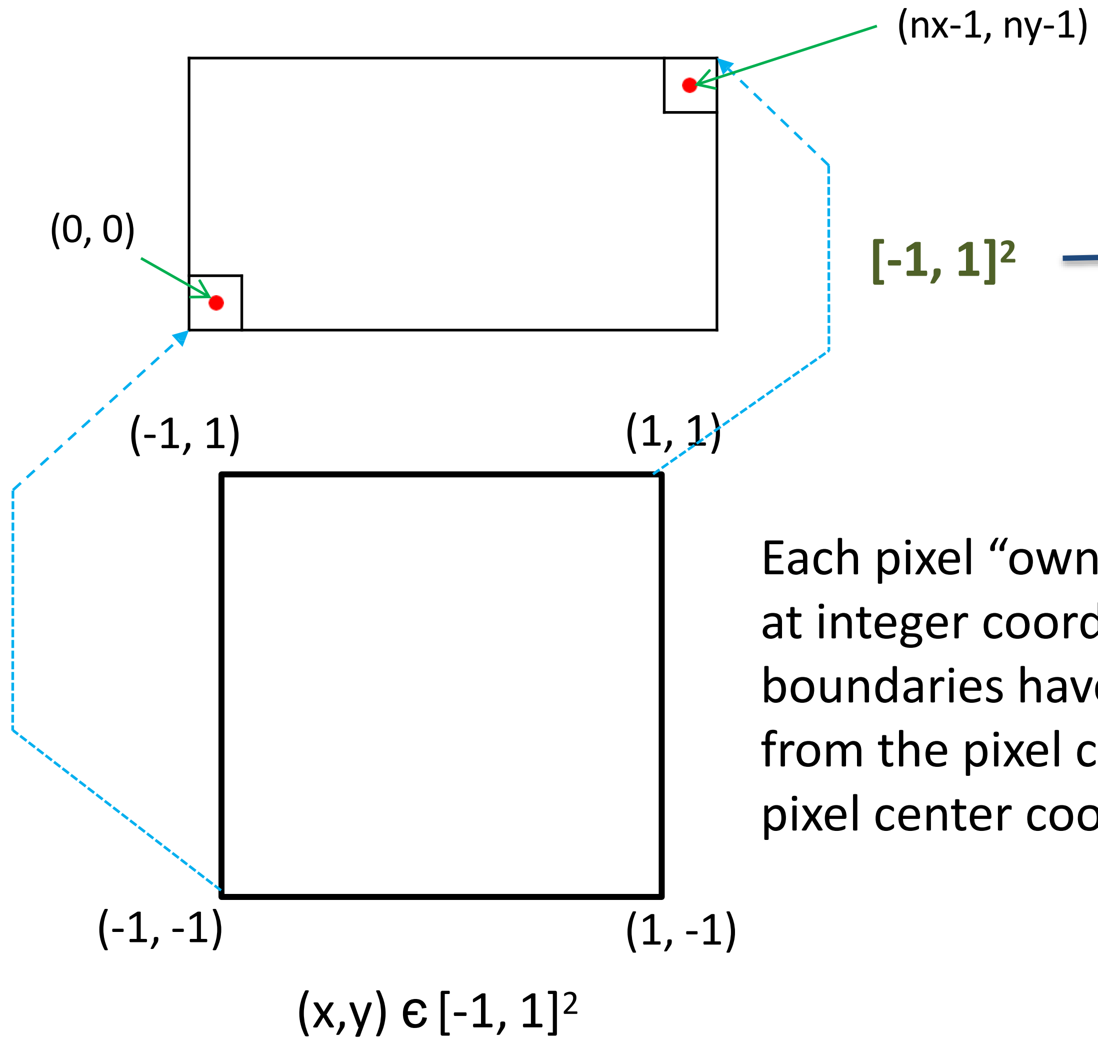
Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Viewport Transformation (5/19)



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Viewport Transformation (7/19)

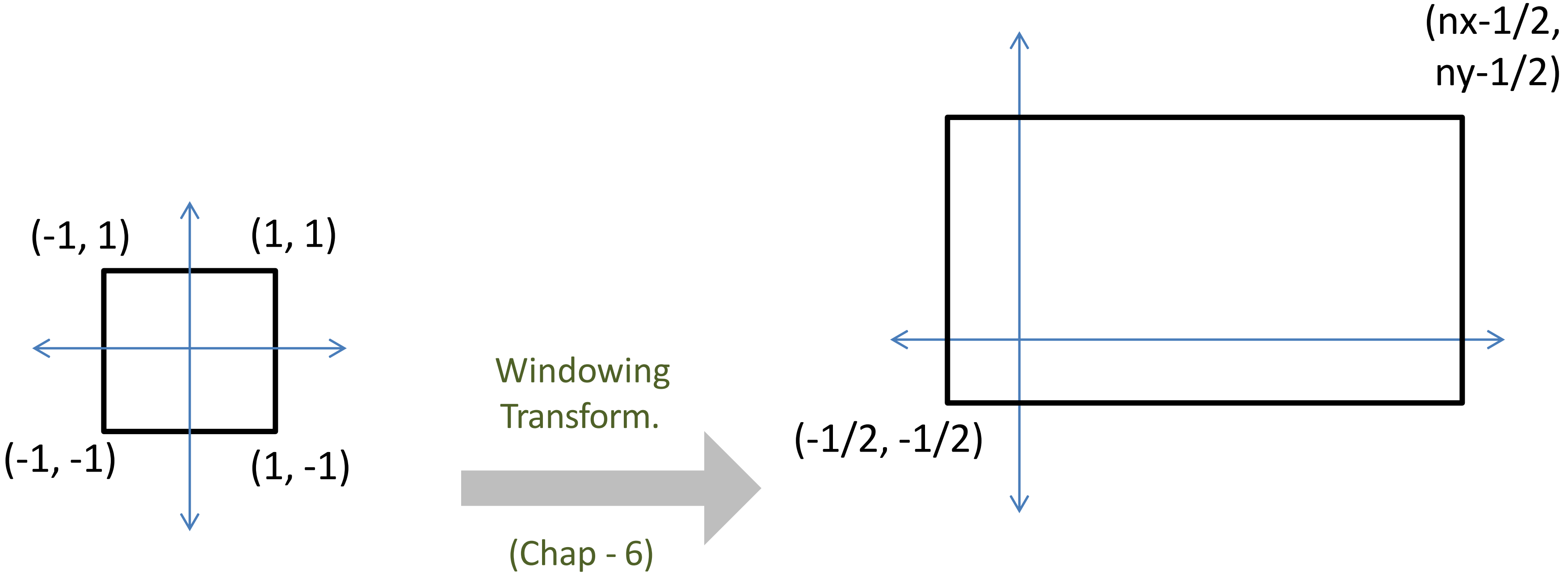


$[-1, 1]^2 \longrightarrow [-0.5, nx-0.5] \times [-0.5, ny-0.5]$

Each pixel "owns" a unit square centered at integer coordinates; the image boundaries have a half unit overshoot from the pixel centers; and the smallest pixel center coordinates are (0, 0).

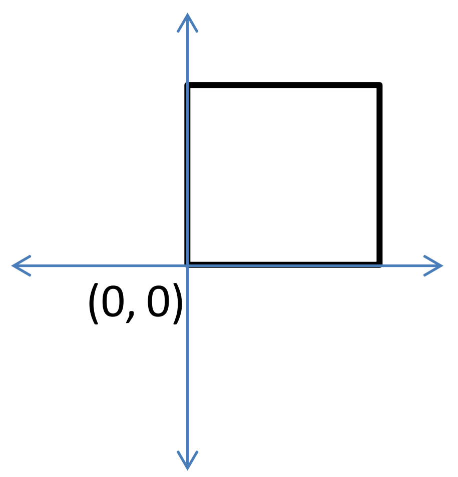
Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

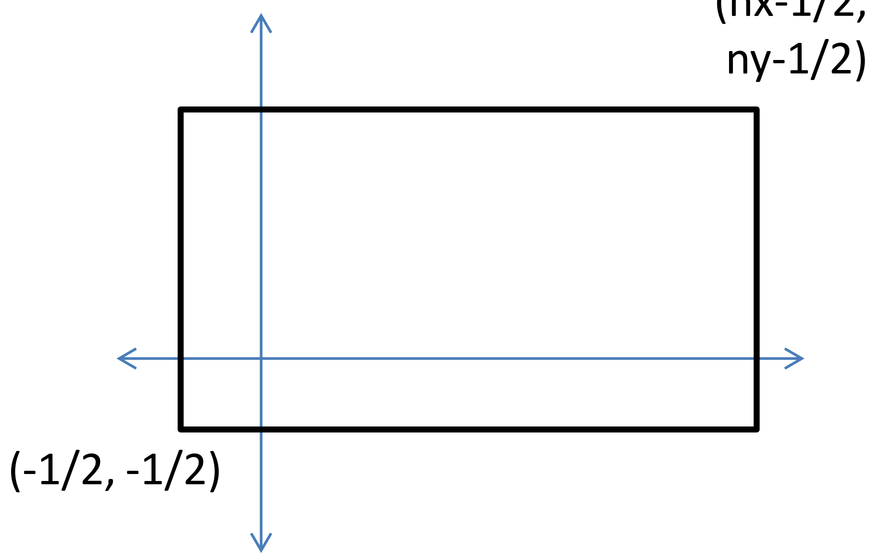
Viewport Transformation (8/19)



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

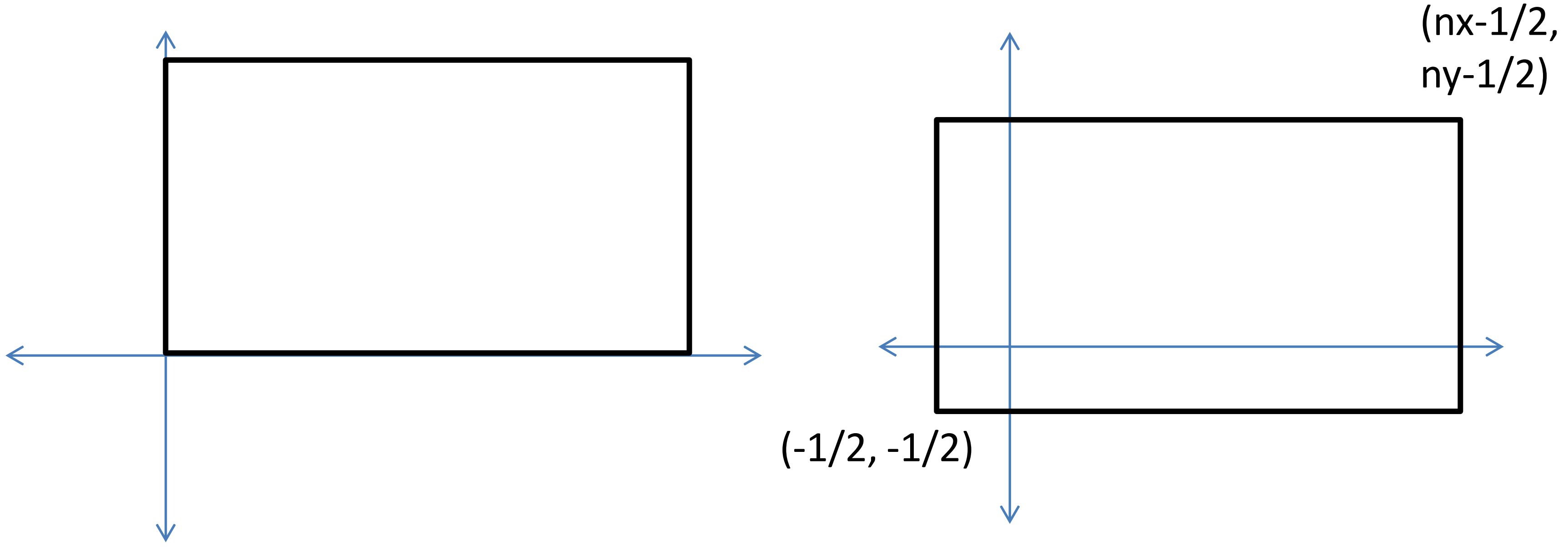
Viewport Transformation (9/19)





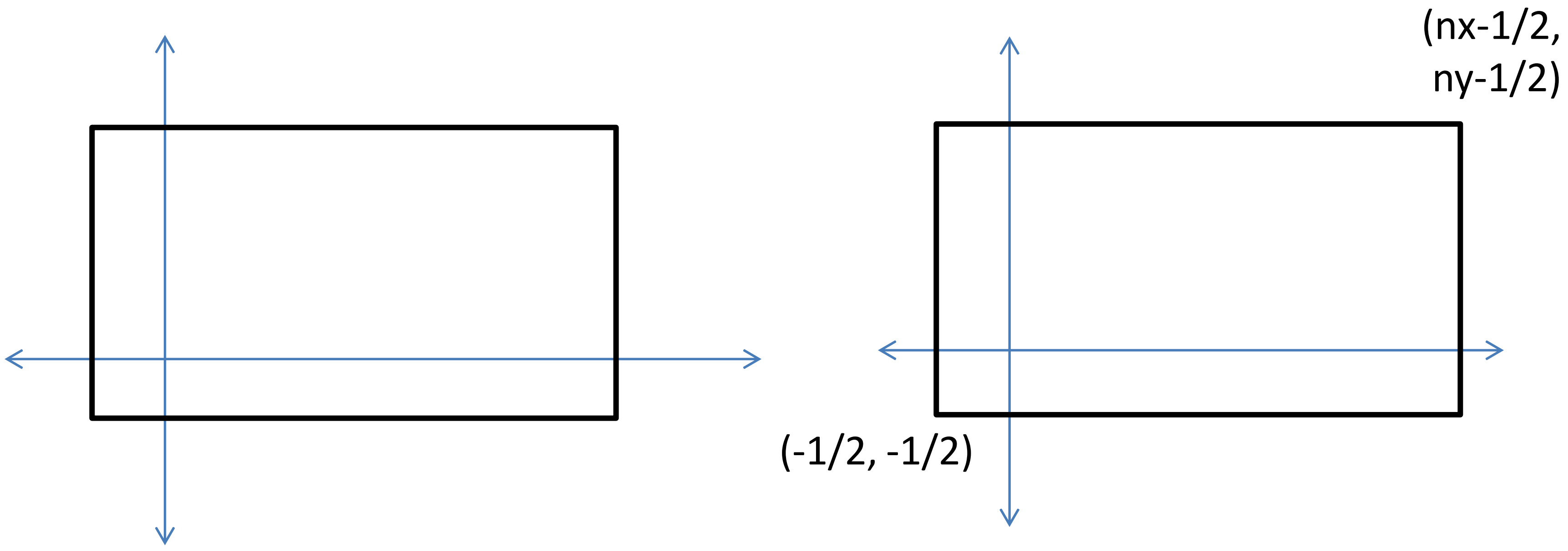
T(1, 1)

Viewport Transformation (10/19)



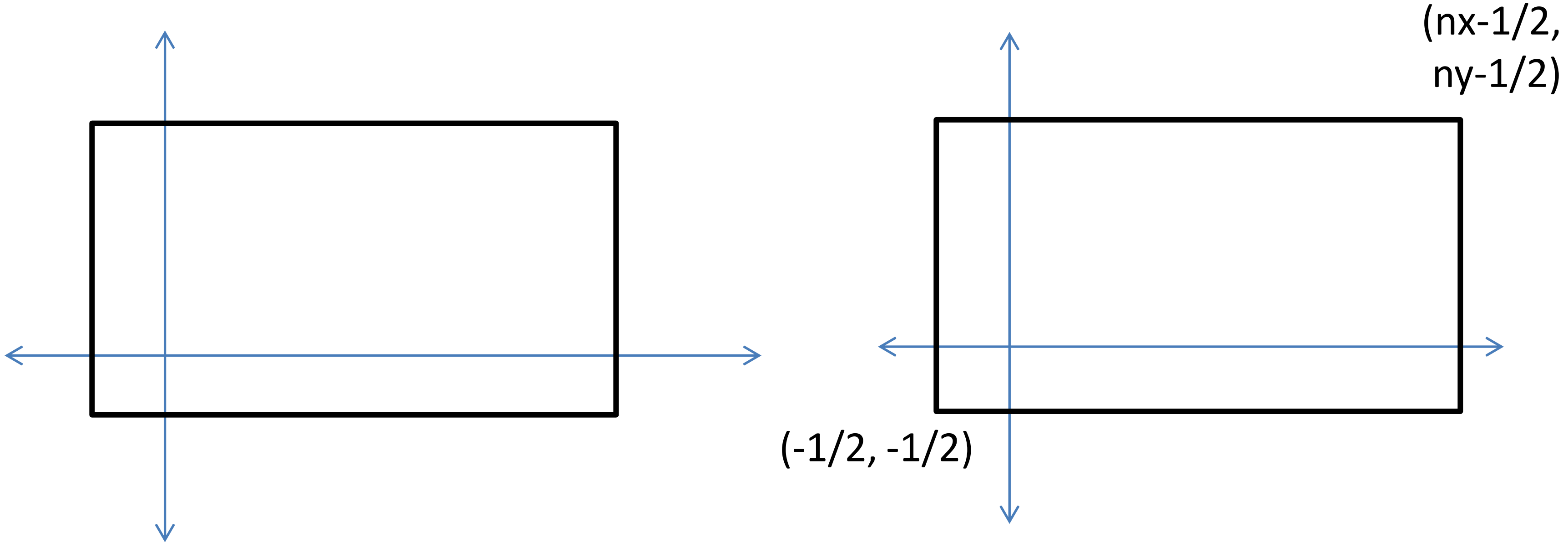
T(1, 1) ⟶ S(nx/2, ny/2)

Viewport Transformation (11/19)



T(1, 1) ⟶ S(nx/2, ny/2) ⟶ T(-1/2, -1/2)

Viewport Transformation (12/19)



$T(1, 1) \longrightarrow S(nx/2, ny/2) \longrightarrow T(-1/2, -1/2)$

$$M_{vp} = T(-1/2, -1/2) * S(nx/2, ny/2) * T(1,1)$$

Viewport Transformation (15/19)

M_{vp} = T(-1/2, -1/2) * S(nx/2, ny/2) * T(1,1)

$$\begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

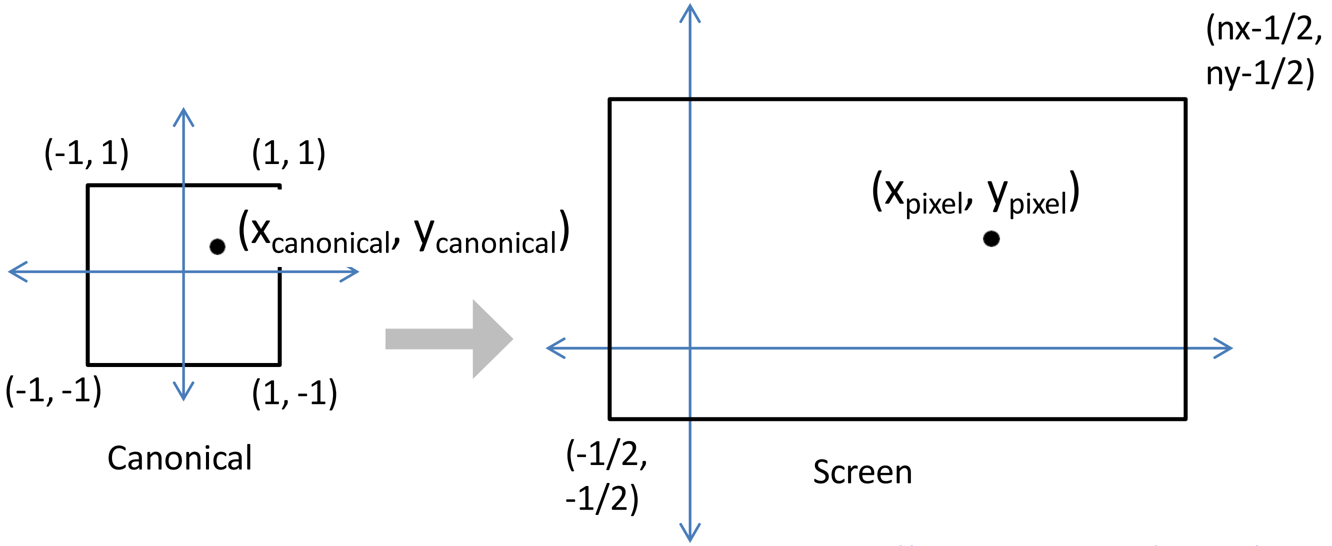
Q: Do matrix multiplication and check

Viewport Transformation (16/19)

$$M_{vp} = T(-1/2, -1/2) * S(nx/2, ny/2) * T(1, 1)$$

This is similar to windowing transform. (Chap - 6)

$$\begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y - 1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ 1 \end{bmatrix}$$



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Viewport Transformation (18/19)

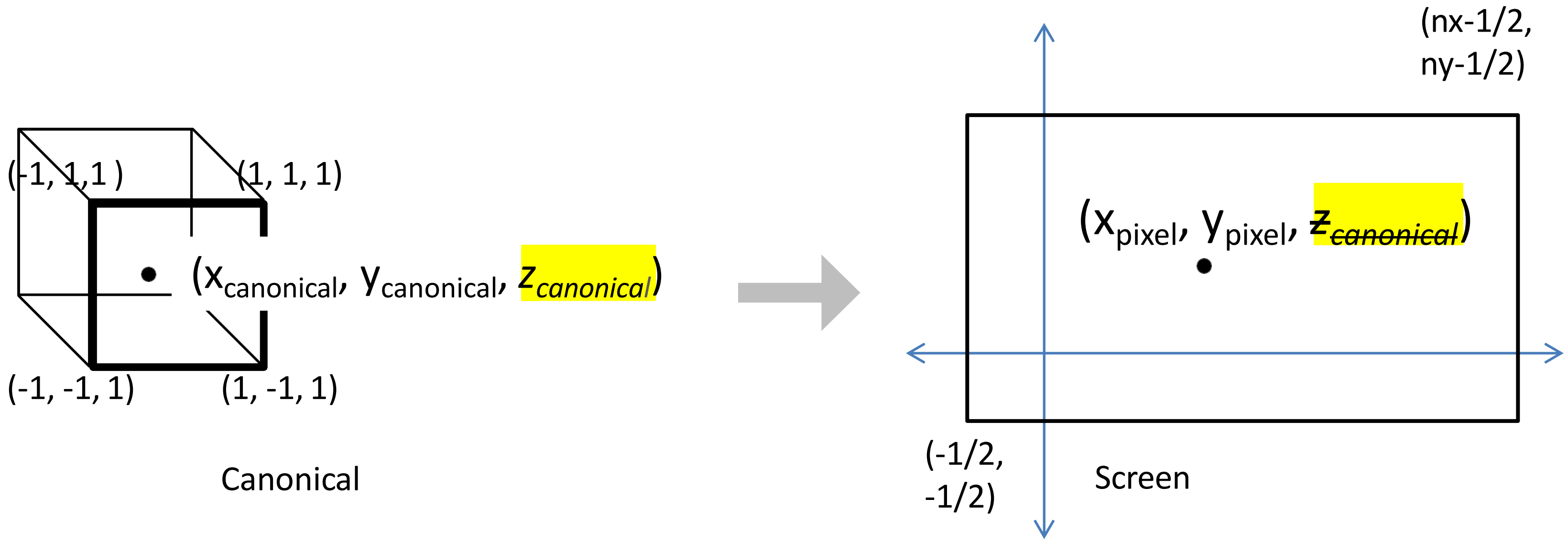
$$\cancel{\begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix}}$$

$$M_{\text{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewport Transformation (19/19)

[carry along the z-coordinate without changing it]

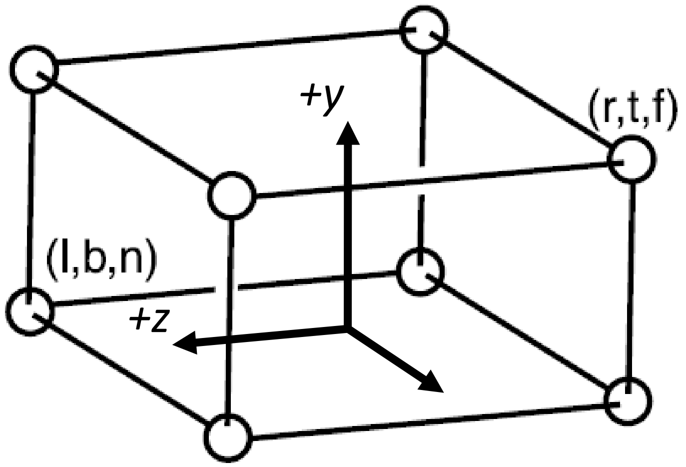
$$\begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \\ z_{\text{canonical}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ z_{\text{canonical}} \\ 1 \end{bmatrix}$$



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

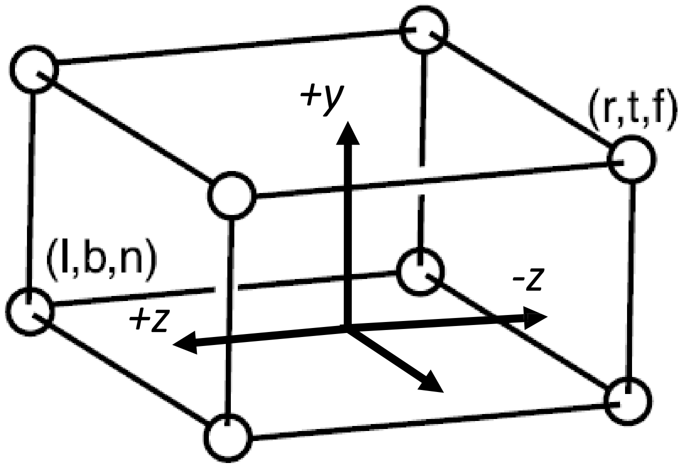
Orthographic Projection Transformation (1/1)

- What if we want to render geometry in some region other than canonical view vol.?



Orthographic View Volume (1/3)

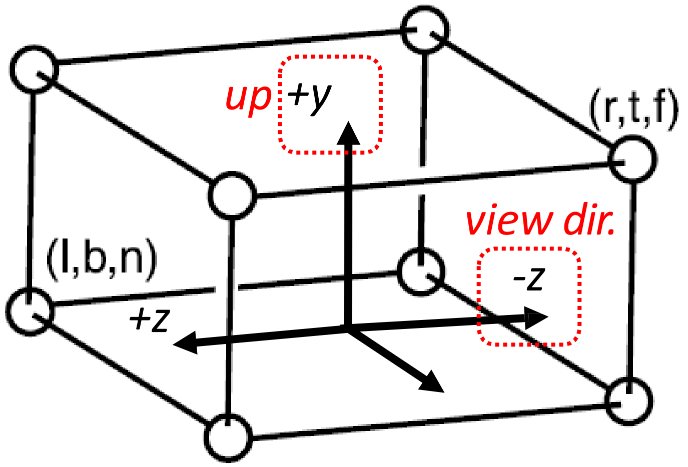
- We'll name the coordinates of its sides so that the view volume is ***[l, r] × [b, t] × [f, n]***
  - View direction: *looking along −z*
  - Orientation: *+y up*
    - $x = l \equiv$ *left plane,*
    - $x = r \equiv$ *right plane,*
    - $y = b \equiv$ *bottom plane,*
    - $y = t \equiv$ *top plane,*
    - $z = n \equiv$ *near plane,*
    - $z = f \equiv$ *far plane.*



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Orthographic View Volume (2/3)

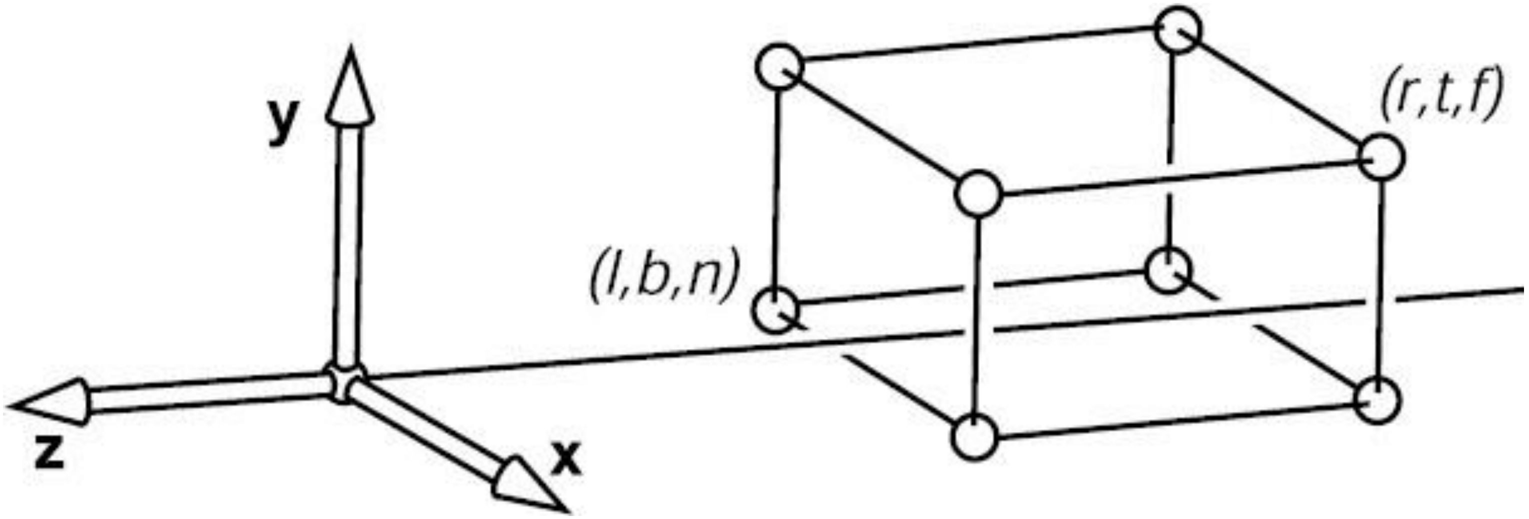
- Looking along the *minus z-axis* with his head pointing in the *positive y-direction.*
  - View direction: *looking along −z*
  - Orientation: *+y up*
- *But, this is unintuitive!*



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Orthographic View Volume (3/3)

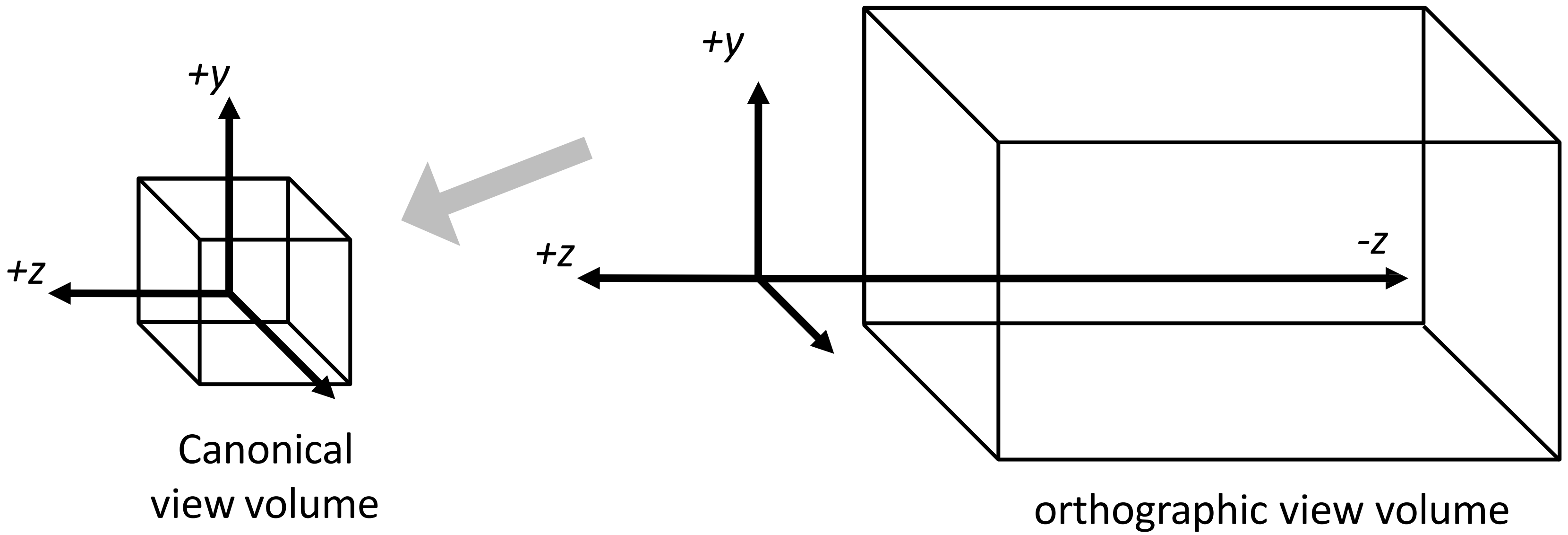
- If entire orthographic view volume has **negative *z*** then $n > f$.
  - *$z = n$ plane is closer*



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Orthographic to Canonical View Volume (1/3)

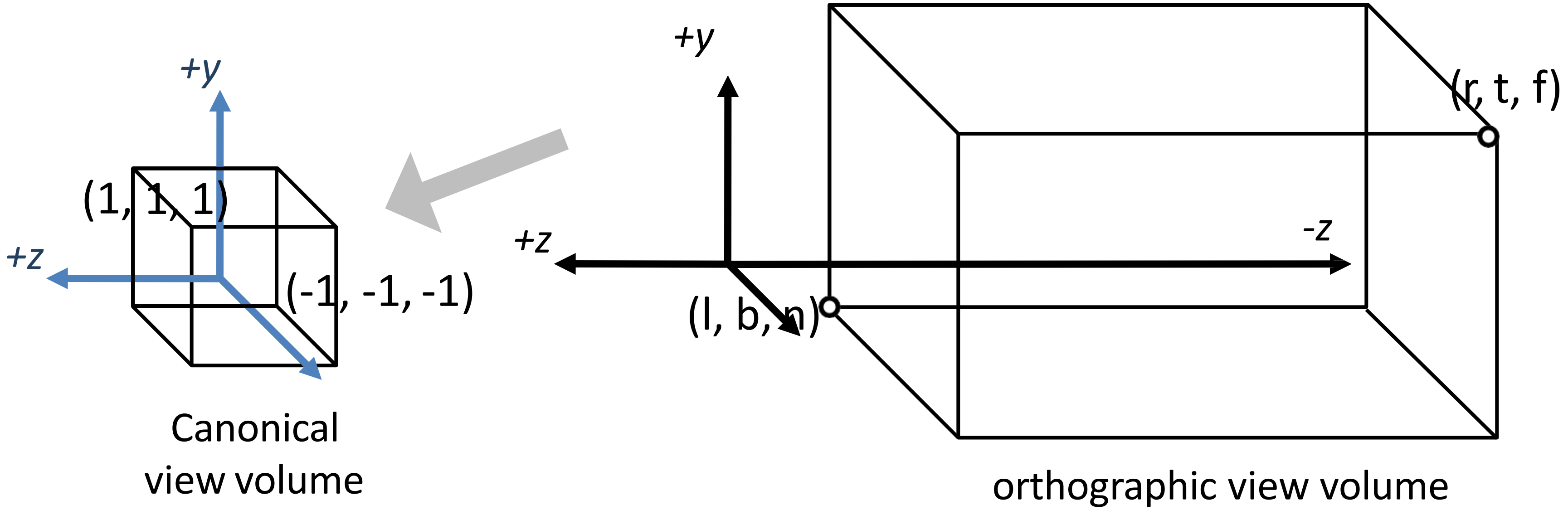
- Transform from orthographic view volume to the canonical view volume
  - We need to apply *windowing transformation* (just like before!)



Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Orthographic to Canonical View Volume (2/3)

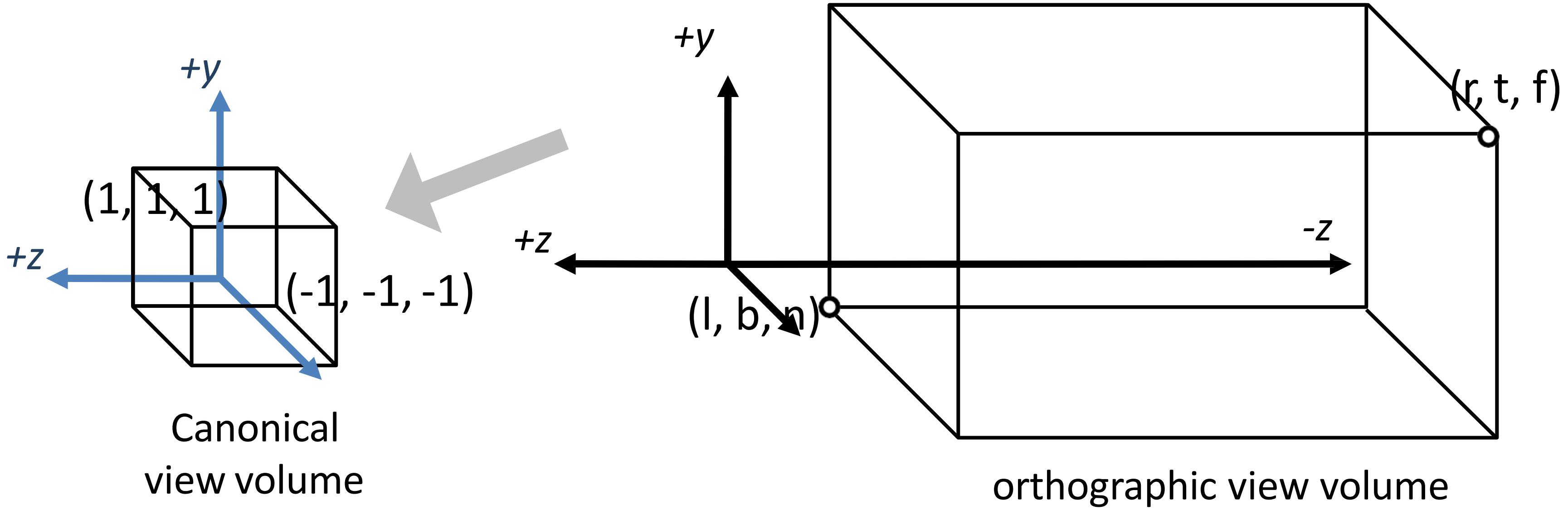
$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Orthographic to Canonical View Volume (3/3)

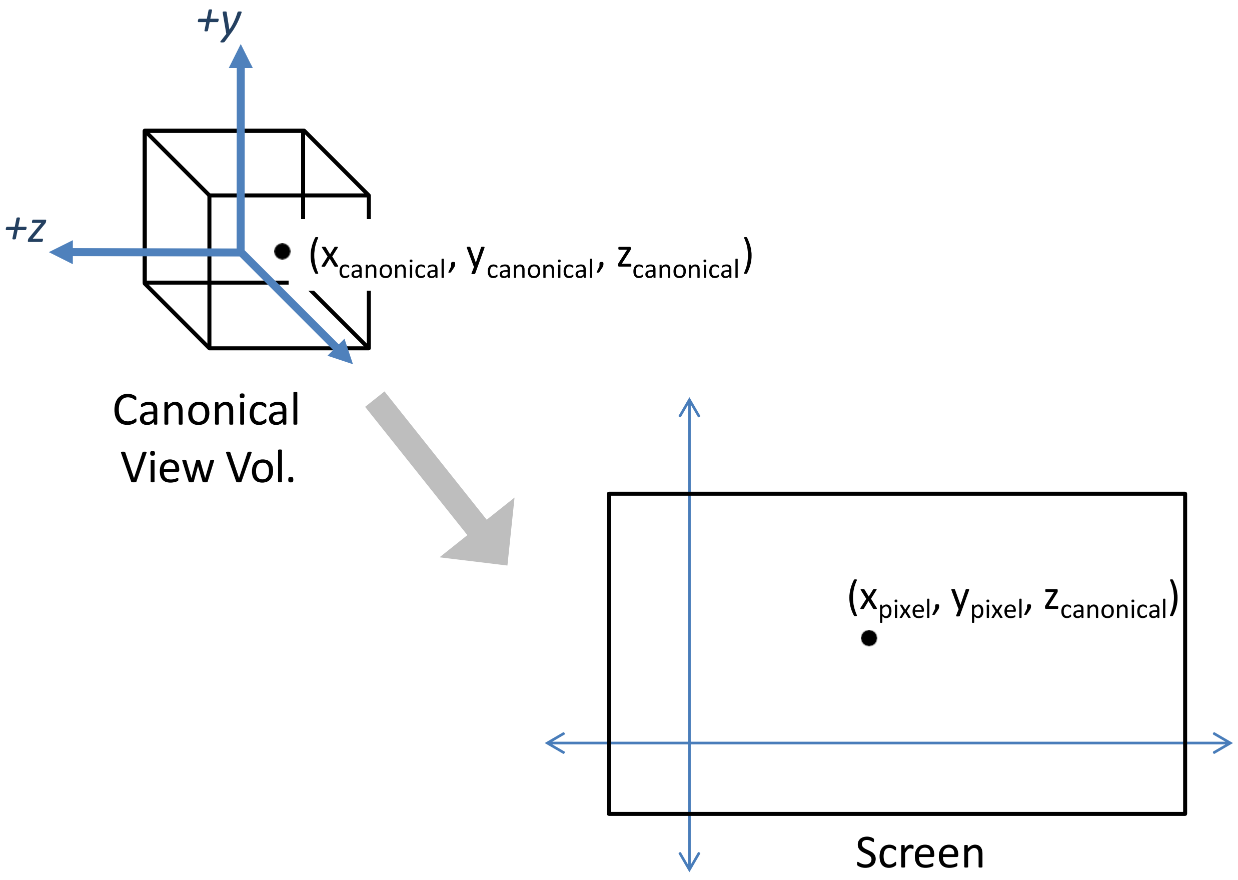
$$\mathrm{M}_{\mathrm{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q: How can we get this matrix?
Help: Chap 6 (Windowing Transformation) and M_{vp}

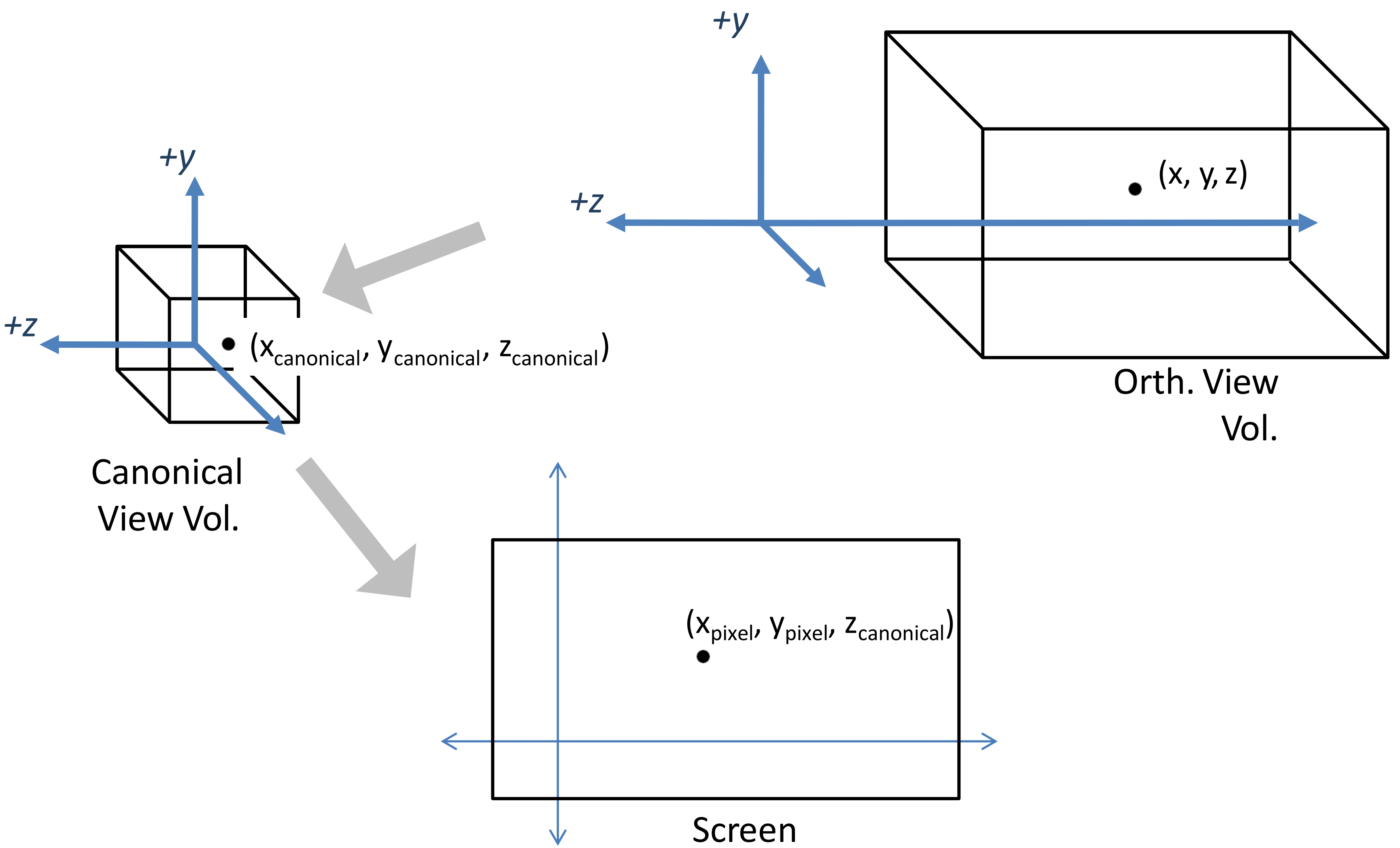


Credit: Fundamentals of Computer Graphics 3rd Edition by Peter Shirley, Steve Marschner | http://www.cs.cornell.edu/courses/cs4620/2019fa/

Orthographic → Canonical → Screen (1/5)

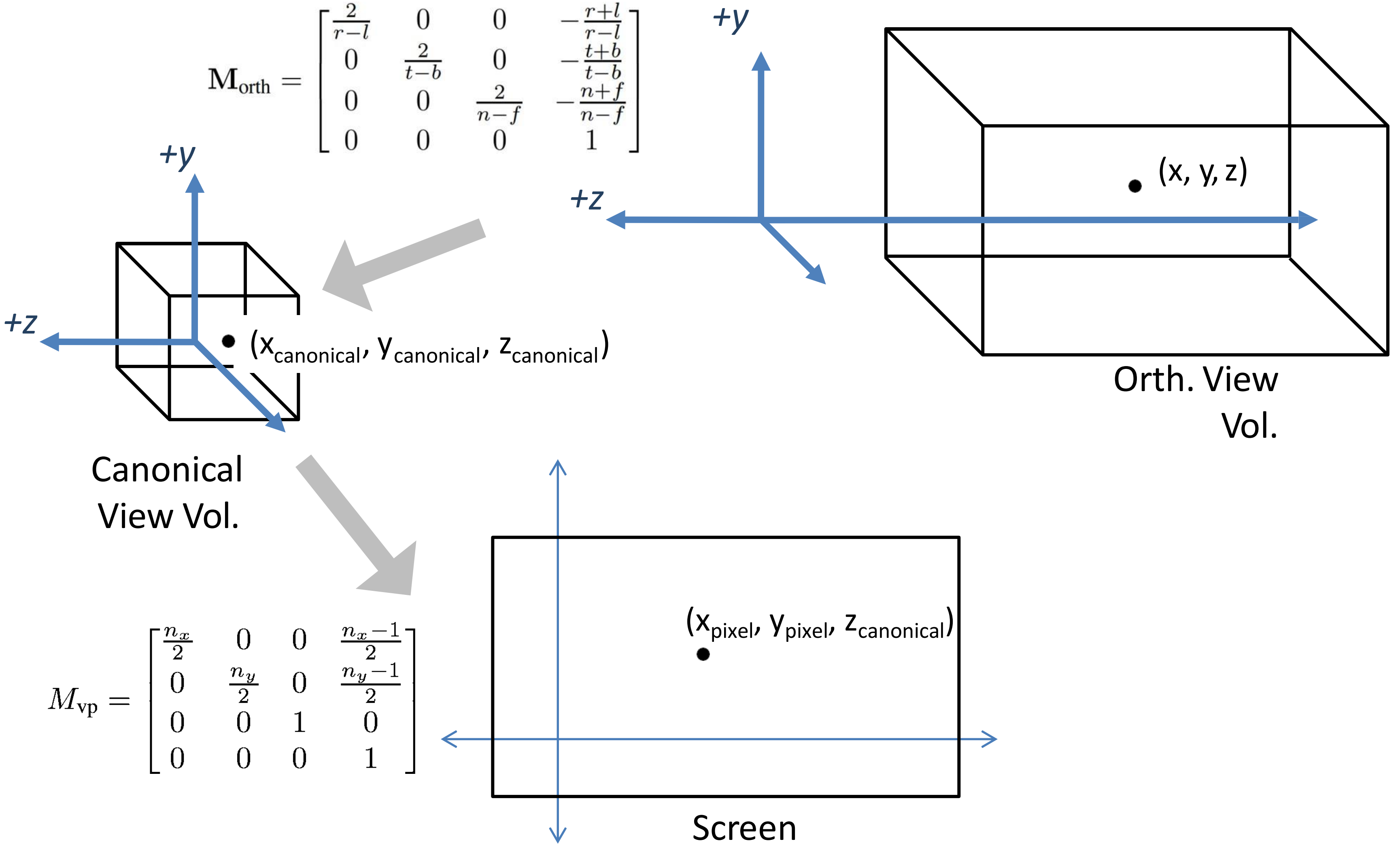


Orthographic → Canonical → Screen (2/5)



Orthographic → Canonical → Screen (3/5)

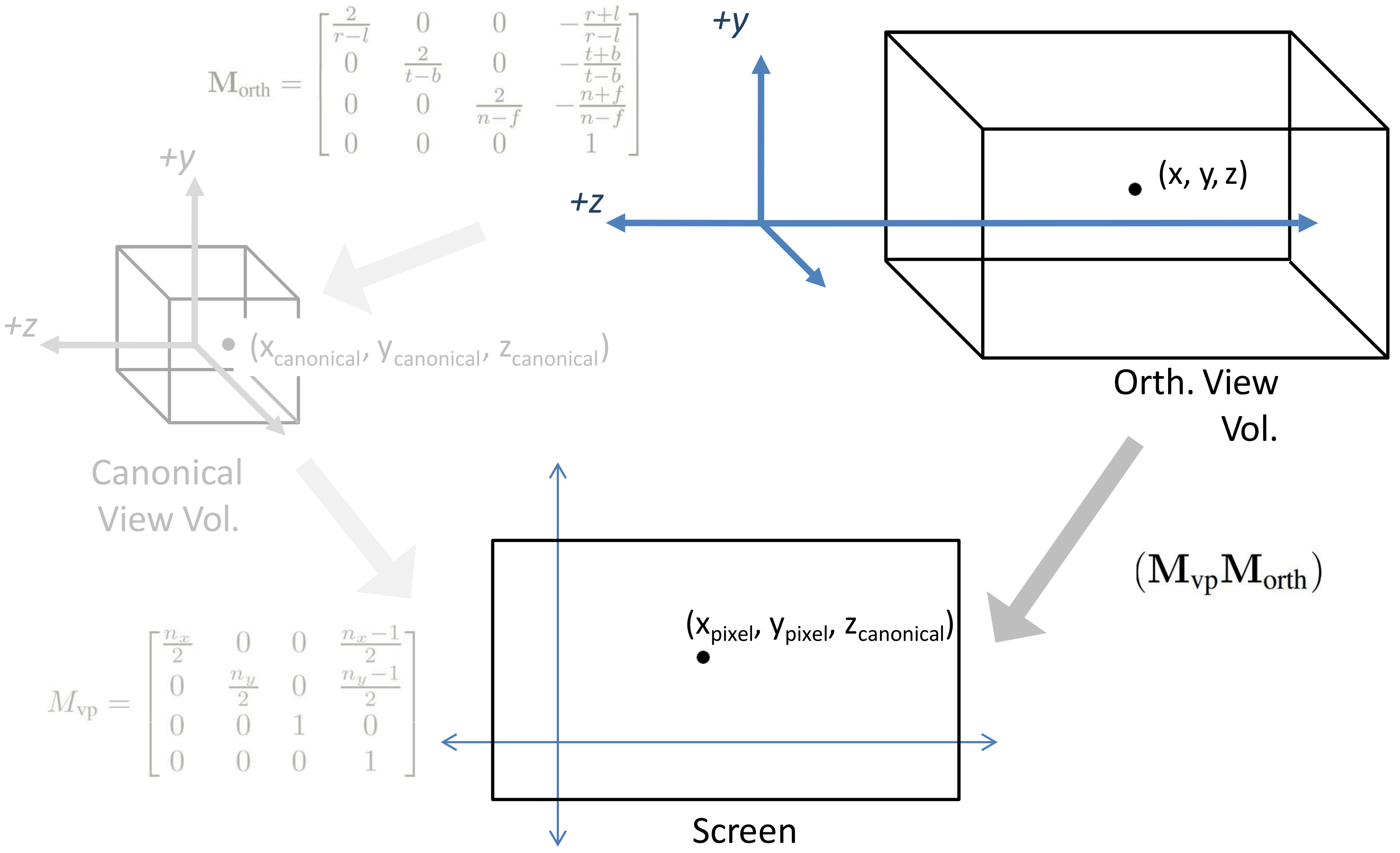
$$M_{\text{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$M_{\text{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic → Canonical → Screen (4/5)

$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

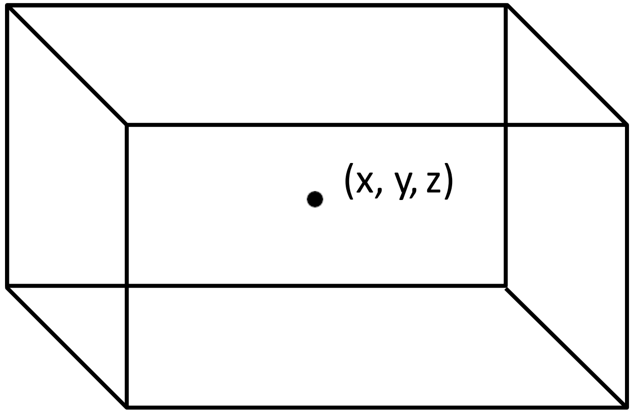


$(\mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}})$

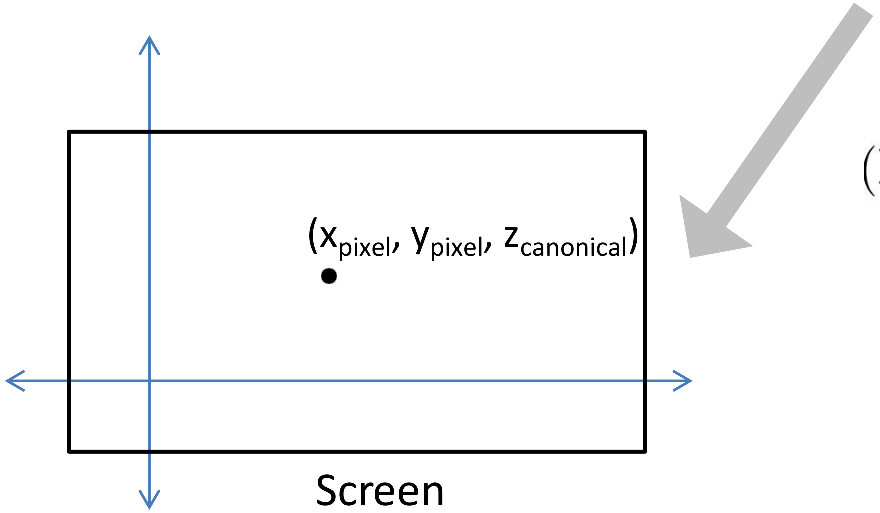
$$M_{\text{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic → Canonical → Screen (5/5)

$$\begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \\ z_{\text{canonical}} \\ 1 \end{bmatrix} = (\mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Orth. View Vol.



Screen

$(\mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}})$

Code: *Orthographic to Screen* (1/1)

Drawing many 3D lines with endpoints a_i and b_i:

```
Construct M_vp
Construct M_orth
M = M_vp*M_orth

for each line segment (a_i, b_i) do:
      p = M*a_i
      q = M*b_i
      drawline (x_p, y_p, x_q, y_q)
```

Practice Problem - 1

Transform a 3D line AB from an *orthographic view volume* to a *viewport* of size 128 x 96. Vertices of the line are A(-1, -3, -5) and B(2, 4, -6). The orthographic view volume has the following setup:

$l = -4, r = 4, b = -4, t = 4, n = -4, f = -8$

You must -

a. Determine the transformation matrix *M*.
b. Multiply *M* with the vertices of the line and determine the position of vertices on viewport.

Practice Problem – 1 (Sol.)

$$M_{\mathrm{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

nx = 128
ny = 96

$$\mathbf{M}_{\mathrm{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

l = - 4, r = 4,
b = - 4, t = 4,
n = - 4, f = - 8

Practice Problem – 1 (Sol.)

M = M_{vp}*M_{orth}

$$\mathbf{M} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A’ = M A

B’ = M B

Additional Reading

- Wireframe renderings
- Derive M_{orth}