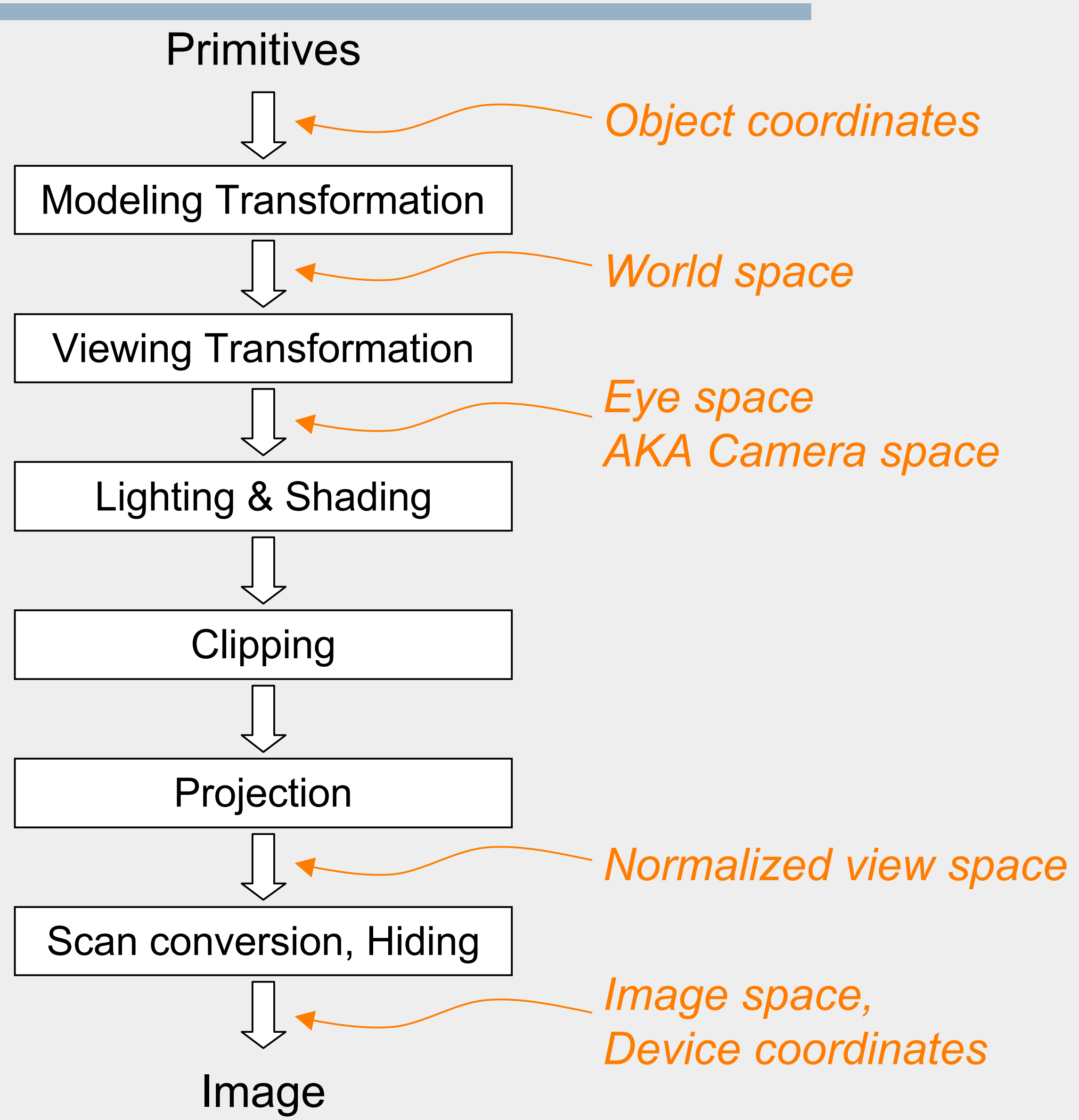
#6: Camera Perspecctive, Viewing, and Culling

CSE167: Computer Graphics

Instructor: Ronen Barzel

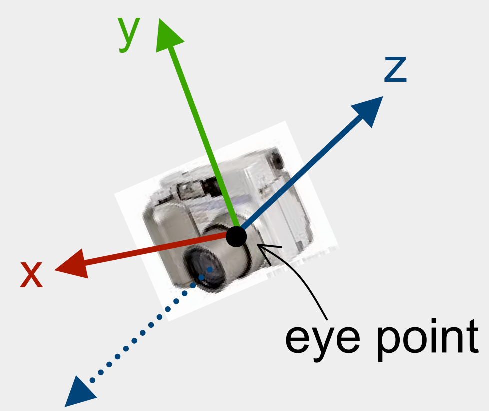
UCSD, Winter 2006

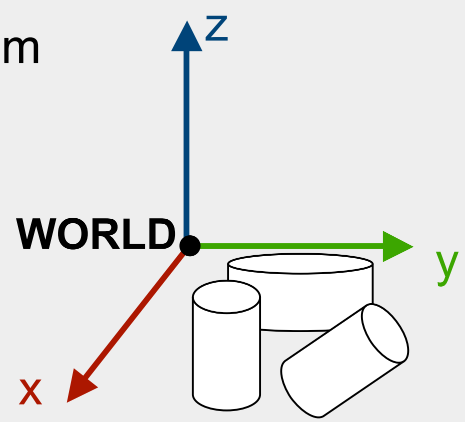
3-D Graphics Rendering Pipeline



Camera

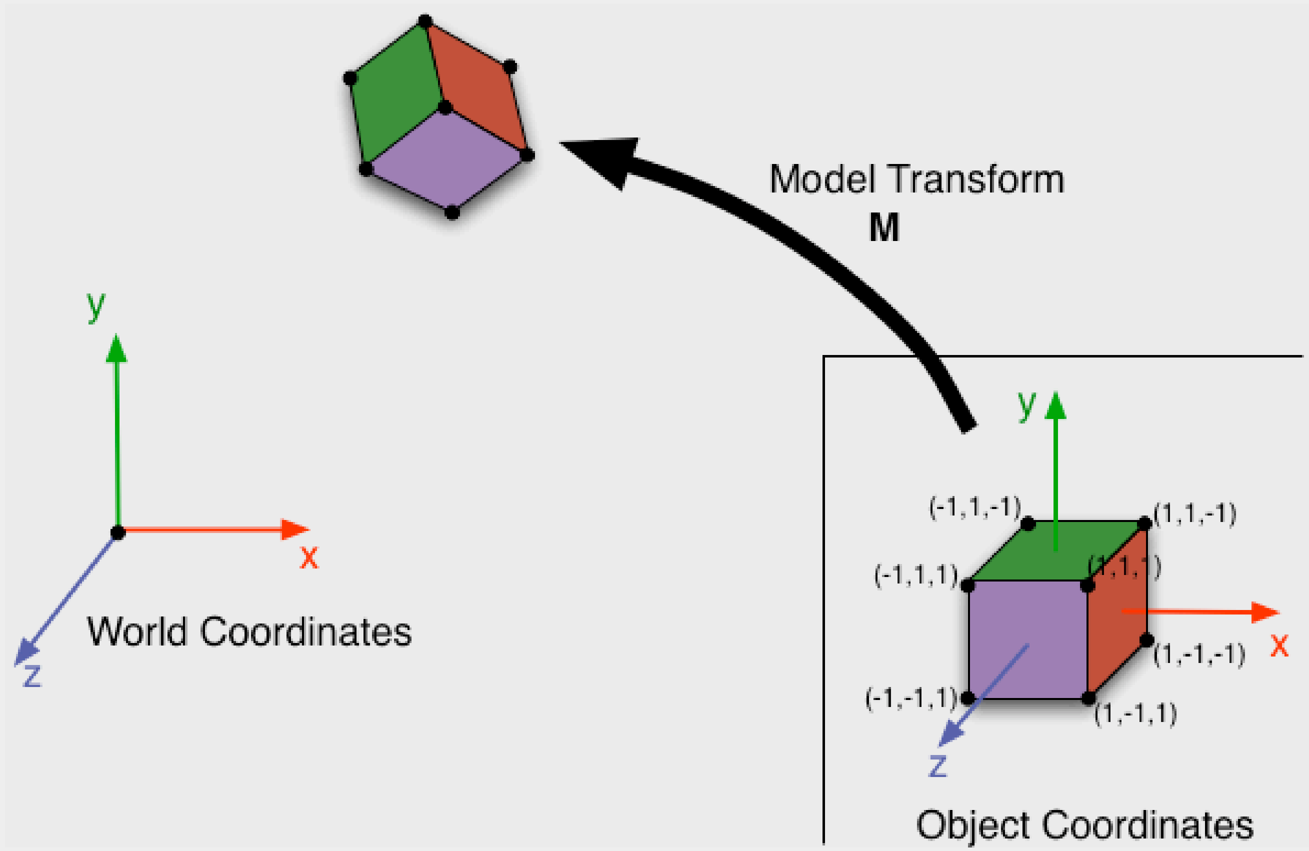
- Think of camera itself as a model
  - Place it in 3D space
- Camera's frame:
  - origin at *eye point*
  - -z points in the viewing direction
  - x,y define the *film plane*
    - x is to the right on the film
    - y is up on the film





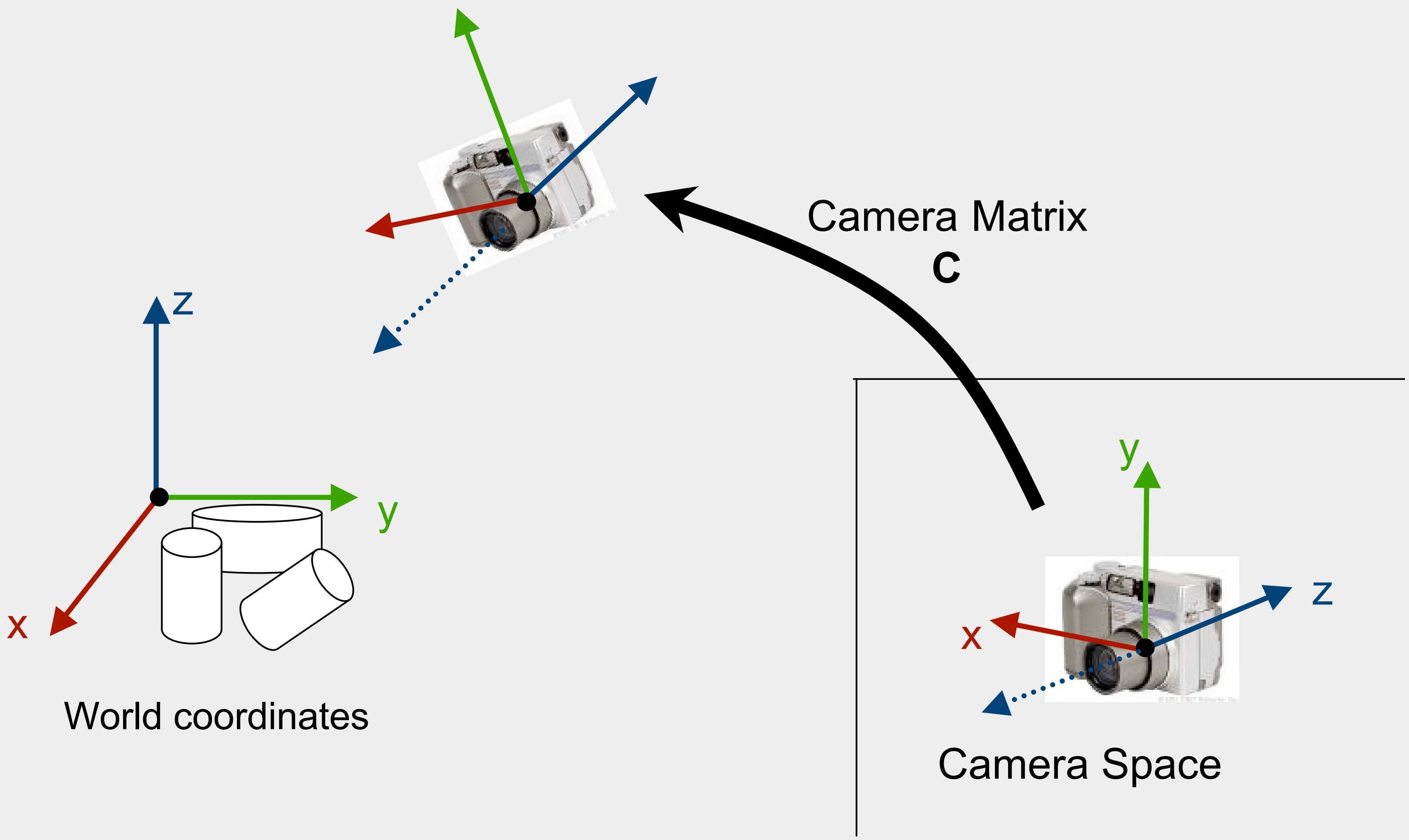
Remember…

- Local-to-world matrix, AKA Model Transform

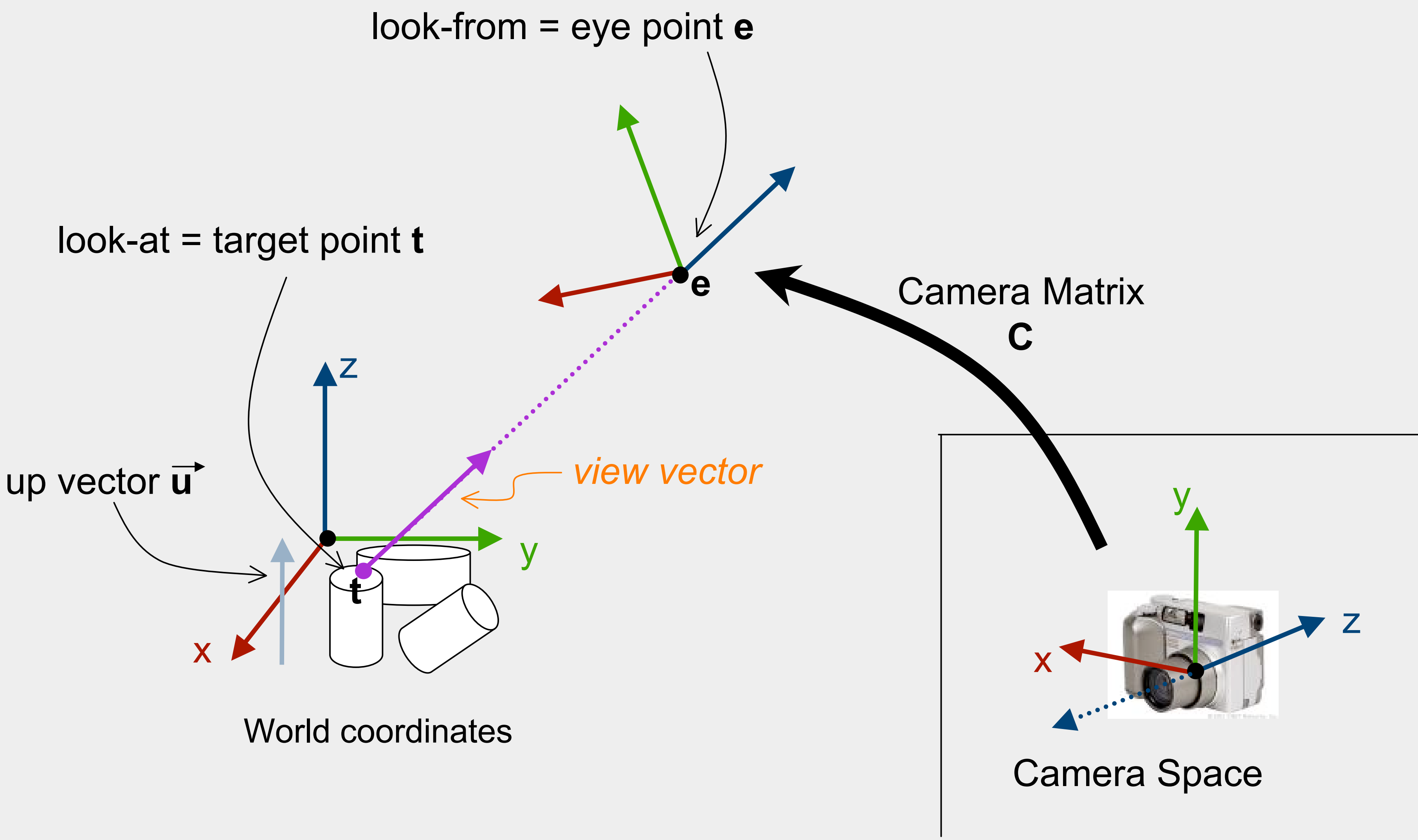


Camera Matrix

- The local-to-world matrix for the camera



Camera Look-At setup



"Look-at" Matrix calculation

- Given:
  - look-from: eye at position **e**
  - look-at: target at position **t**
  - up-vector: $\vec{\mathbf{u}}$

- Fill the **a,b,c,d** columns of the matrix with the world-space coordinates of the camera's frame:
  - **d** is position of frame origin, i.e. the eye point:

    $$\mathbf{d} = \mathbf{e}$$

  - **c** is the z axis of the frame, i.e. the view vector:

$$\vec{\mathbf{c}} = \frac{\mathbf{e} - \mathbf{t}}{|\mathbf{e} - \mathbf{t}|}$$

“Look-at” Matrix calculation

- **a** is the camera frame’s x axis. we want it to be perpendicular to the view vector, and also perpendicular to the up vector:

$$\vec{\mathbf{a}} = \frac{\vec{\mathbf{u}} \times \vec{\mathbf{c}}}{\left|\vec{\mathbf{u}} \times \vec{\mathbf{c}}\right|}$$

- **b** is the camera frame’s y axis. it must be perpendicular to **a** and **c.**

$$\vec{\mathbf{b}} = \vec{\mathbf{c}} \times \vec{\mathbf{a}}$$

- Notes:
  - cross product order is important to make sure the frame is right-handed
  - since **a** and **c** are unit length and perpendicular to each other, we don’t need to normalize **b**.

“Look-at” Matrix calculation, summary

Given: eye point $\mathbf{e}$, target point $\mathbf{t}$, and up vector $\vec{\mathbf{u}}$

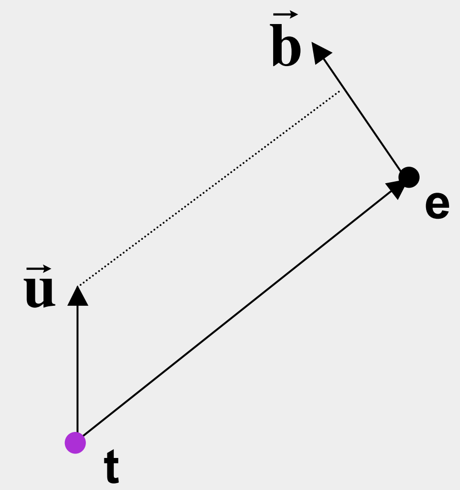
Construct: columns of camera matrix $\mathbf{C}$

$$\mathbf{d} = \mathbf{e}$$

$$\vec{\mathbf{c}} = \frac{\mathbf{e}-\mathbf{t}}{|\mathbf{e}-\mathbf{t}|}$$

$$\vec{\mathbf{a}} = \frac{\vec{\mathbf{u}} \times \vec{\mathbf{c}}}{|\vec{\mathbf{u}} \times \vec{\mathbf{c}}|}$$

$$\vec{\mathbf{b}} = \vec{\mathbf{c}} \times \vec{\mathbf{a}}$$



- Note: The up vector may not end up parallel to the camera y axis
  - The projection of the up vector onto the film plane lines up with camera y
- If the up vector is parallel to the view vector, the result is undefined!
  - the up vector will project to nothing in the image
  - no matter how you spin the camera, there’s no thing to line up with the camera y
  - it’s a user error!

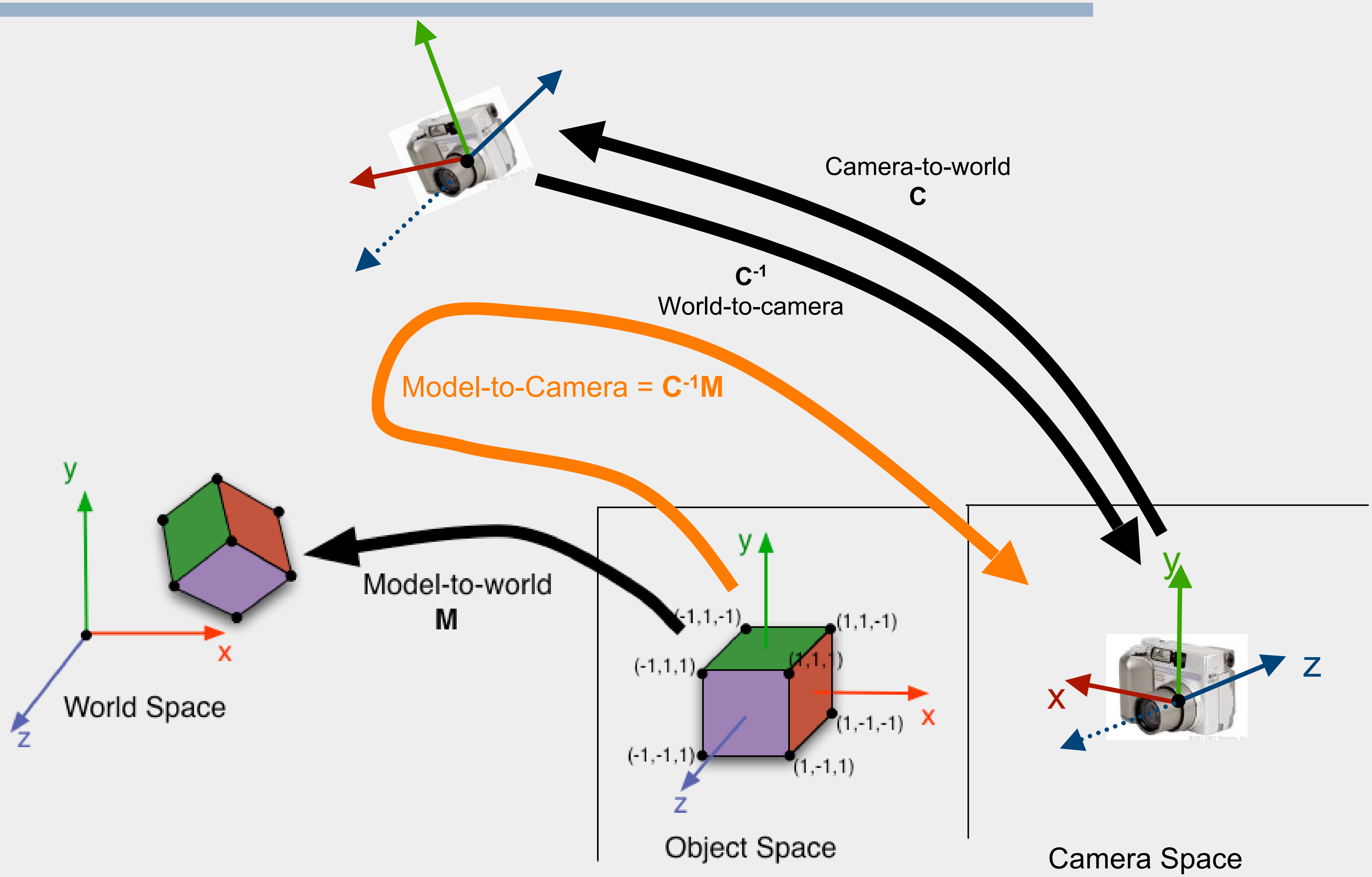
Camera Space

- For rendering, we want to consider all objects in camera space
  - We have matrix **C** that transforms from camera space into world space
  - View an object that was placed into world space using matrix **M**

  - To go from object space to camera space:
    - First go from object to world via **M**
    - Then go *backwards* from world to camera, using the inverse of **C**
    - Compose these into a single matrix:

Object-to-camera = $\mathbf{C}^{-1}\mathbf{M}$

Model-to-Camera transform



In camera space

- We have things lined up the way we like them on screen:
  - X to the right
  - Y up
  - -Z going into the screen
  - Objects to look at are in front of us, i.e. have negative z values
- But the objects are still in 3D.
  - Now let’s look at how to project them into 2D to get them on screen