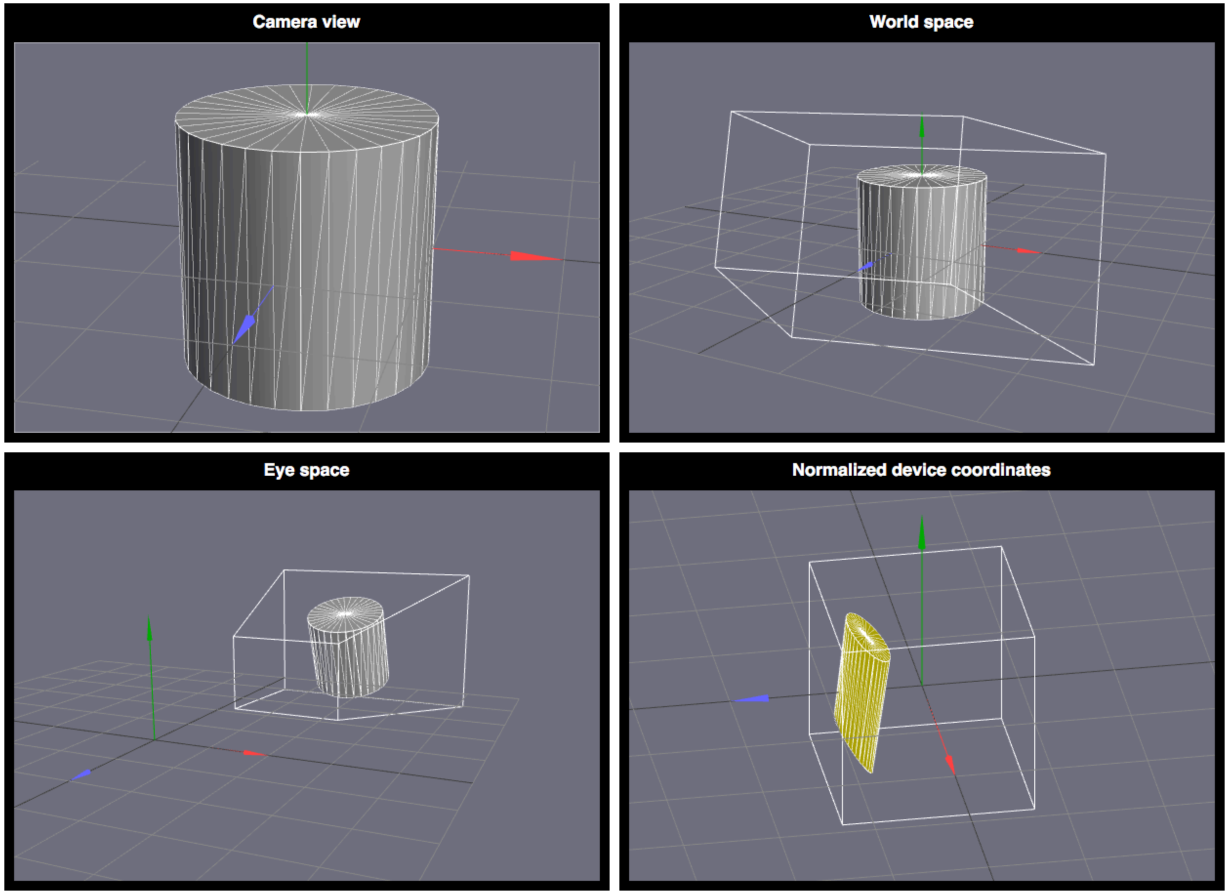


Computer Graphics

Lecture 18

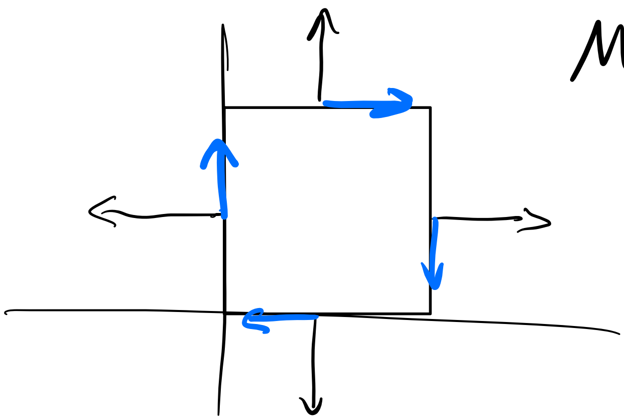
Object Order Rendering

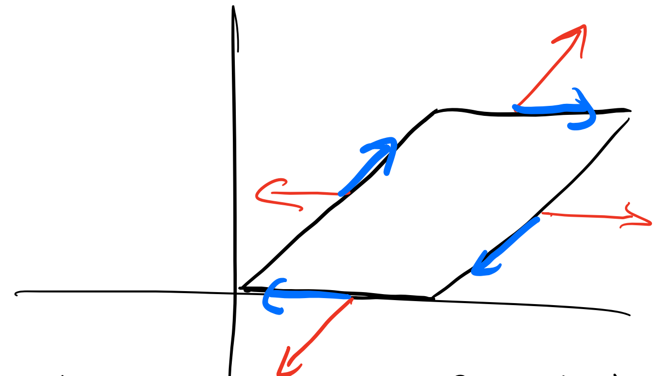
Viewing Transformations - 1

Announcements

Transformations and Normals

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





$$\vec{n} \cdot \vec{t} = 0$$

$$\overbrace{(N\vec{n})}^{\text{aspirationally transformed normal}}{}^{T} \overbrace{(M\vec{t})}^{\text{transformed tangent}} = 0$$

$$(\vec{n}^T N^T)(M\vec{t}) = 0$$

$$\Rightarrow \vec{n}^T \underbrace{N^T M}_{I} \vec{t} = 0$$

$$N^T M = I$$

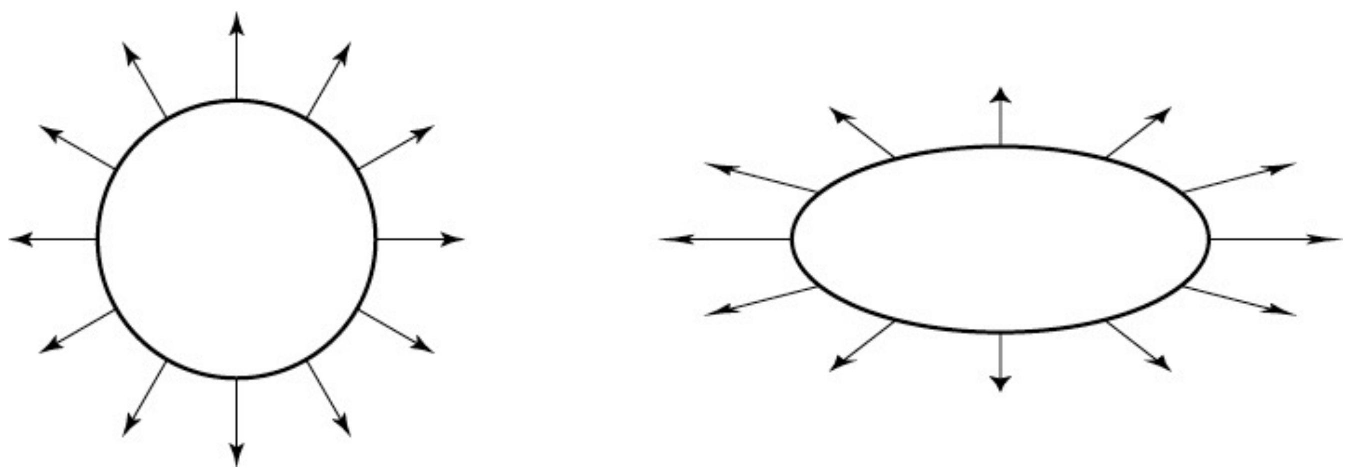
$$N^T \cancel{M M^{-1}} = M^{-1}$$

$$N^T = M^{-1}$$

$$\boxed{N = M^{-T}}$$

Transforming normal vectors

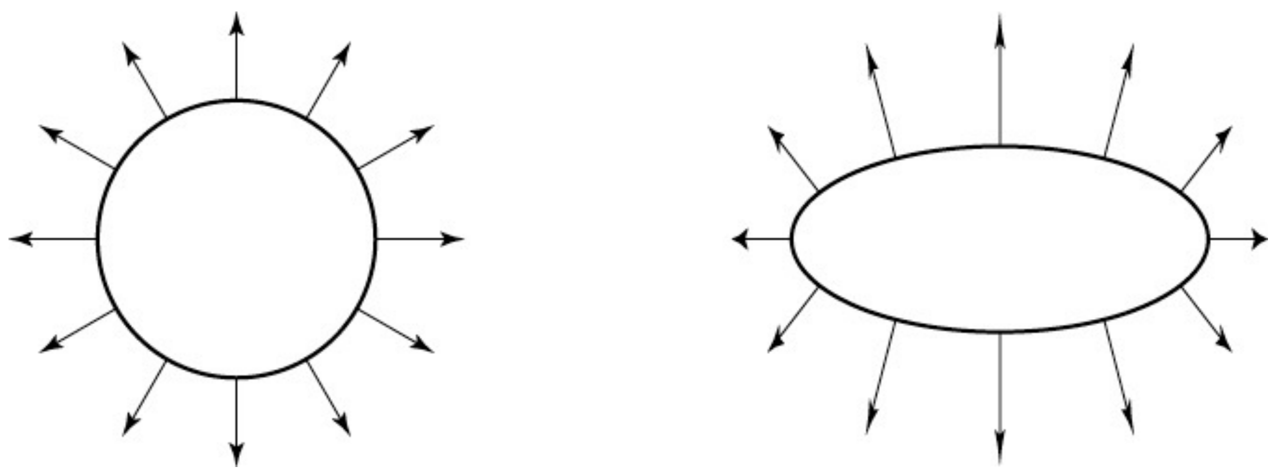
- Transforming surface normals
  - –differences of points (and therefore tangents) transform OK
  - –normals do not --> use inverse transpose matrix



have: $\mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$
want: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$
so set $X = (M^T)^{-1}$
then: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$

Transforming normal vectors

- Transforming surface normals
  - –differences of points (and therefore tangents) transform OK
  - –normals do not --> use inverse transpose matrix



have: $\mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$
want: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$
so set $X = (M^T)^{-1}$
then: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$

Object Order Rendering

```
for each object:
  for each pixel:
    if object affects pixel:
      update pixel's color
```

Object Order Rendering: The Secret Sauce

$$\mathbf{p}_{pixel} = \boxed{M}\mathbf{p}_{object}$$

What does this depend on?

- camera pose
- camera "intrinsics" (d, vp size, etc.)
- object ~~pose~~ pose, size

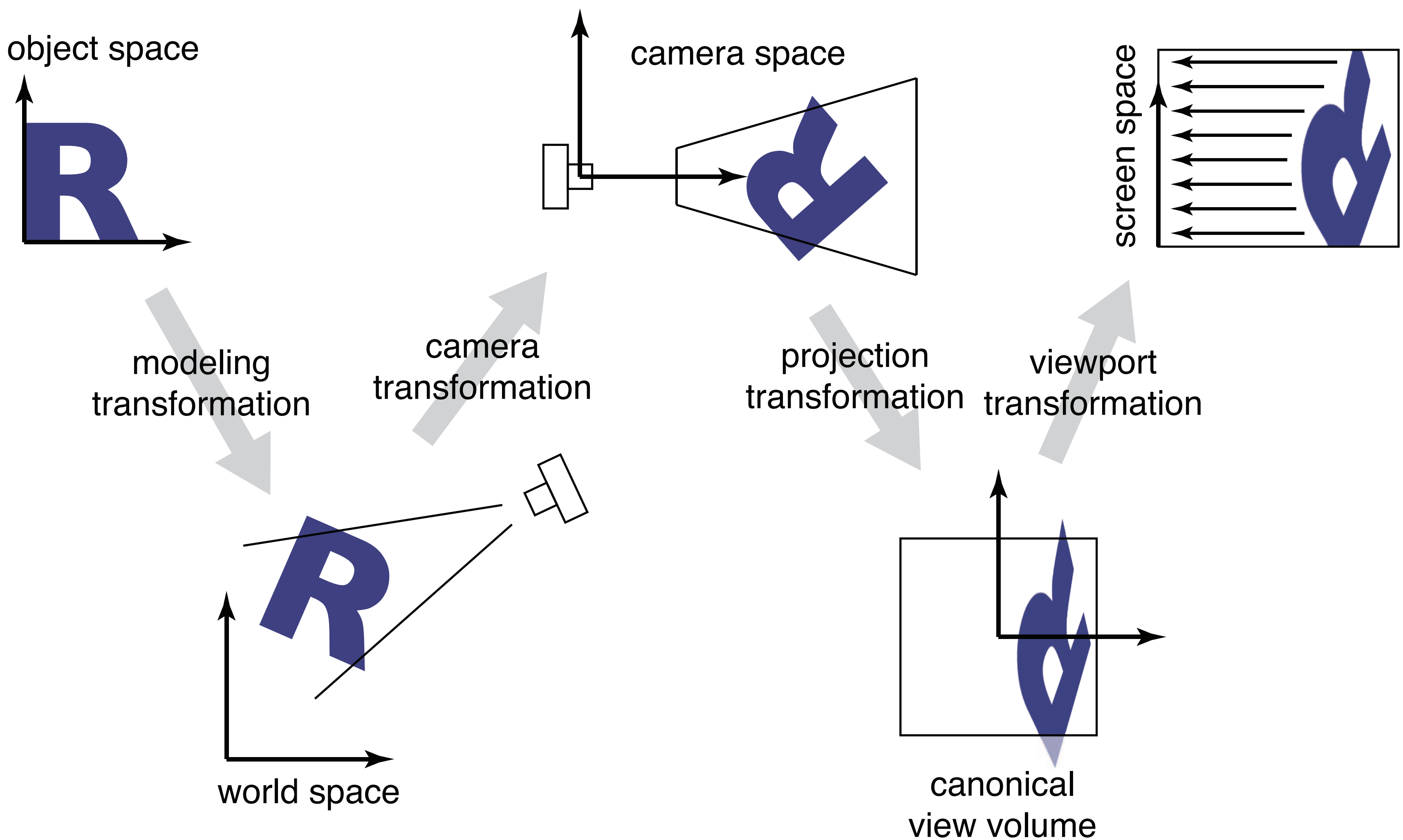
Viewing Transformations

A standard sequence of transforms to go from
object (model) space to **screen (image) space**

whiteboard!
see Scott attempt to draw in 3d!

Viewing Transformations

A standard sequence of transforms to go from
object (model) space to **screen (image) space**



A Wireframe Rendering Algorithm

`Form matrices` $M_{vp}, M_{proj}, M_{cam}, M_{model}$

$M \leftarrow M_{vp} M_{proj} M_{cam} M_{model}$

`for each line segment` $\mathbf{a}_i, \mathbf{b}_i$`:`

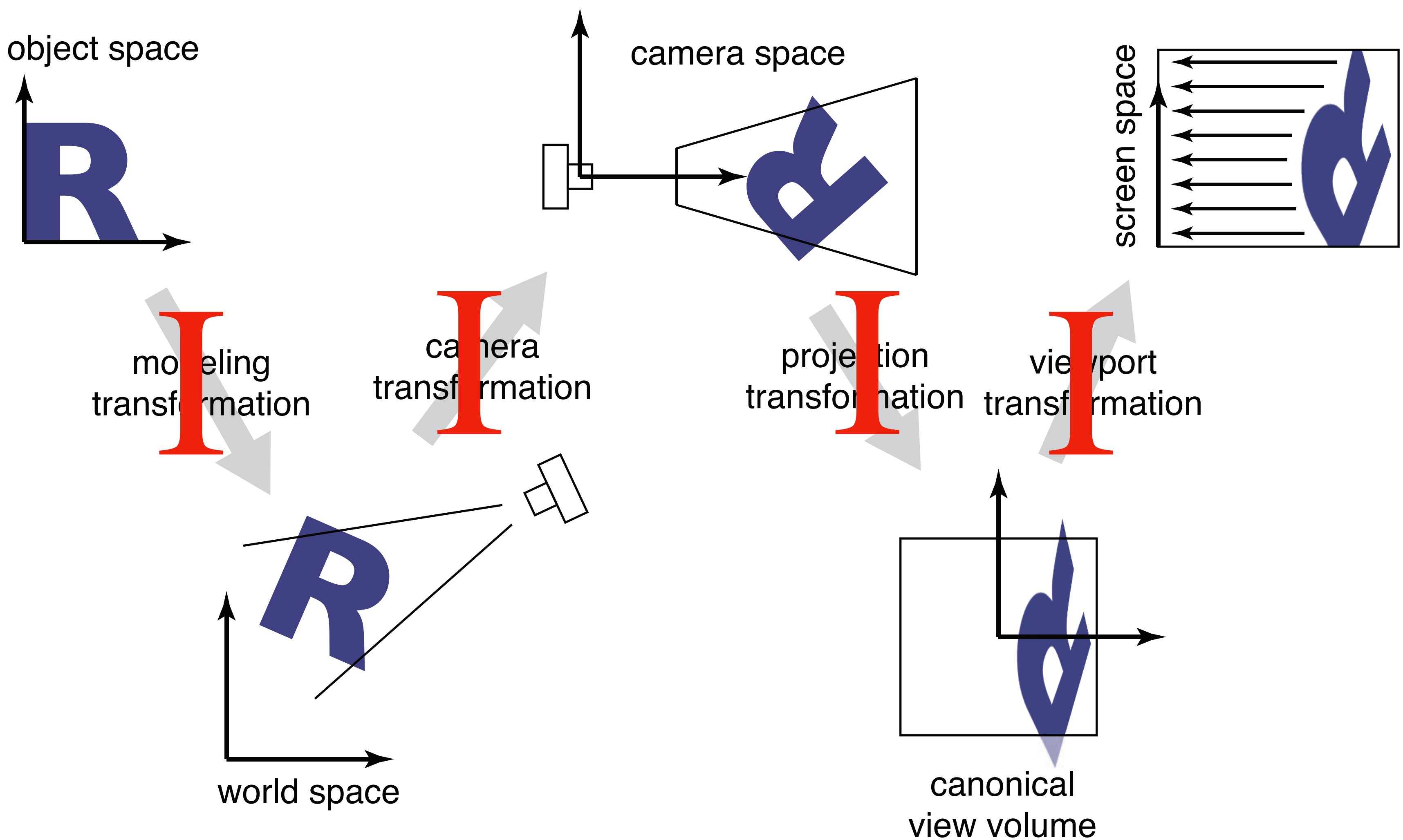
    $\mathbf{p} \leftarrow M\mathbf{a}_i$

    $\mathbf{q} \leftarrow M\mathbf{b}_i$

    `draw_line(`$\mathbf{p}, \mathbf{q}$`)`

Viewing Transformations: Minimalist Edition

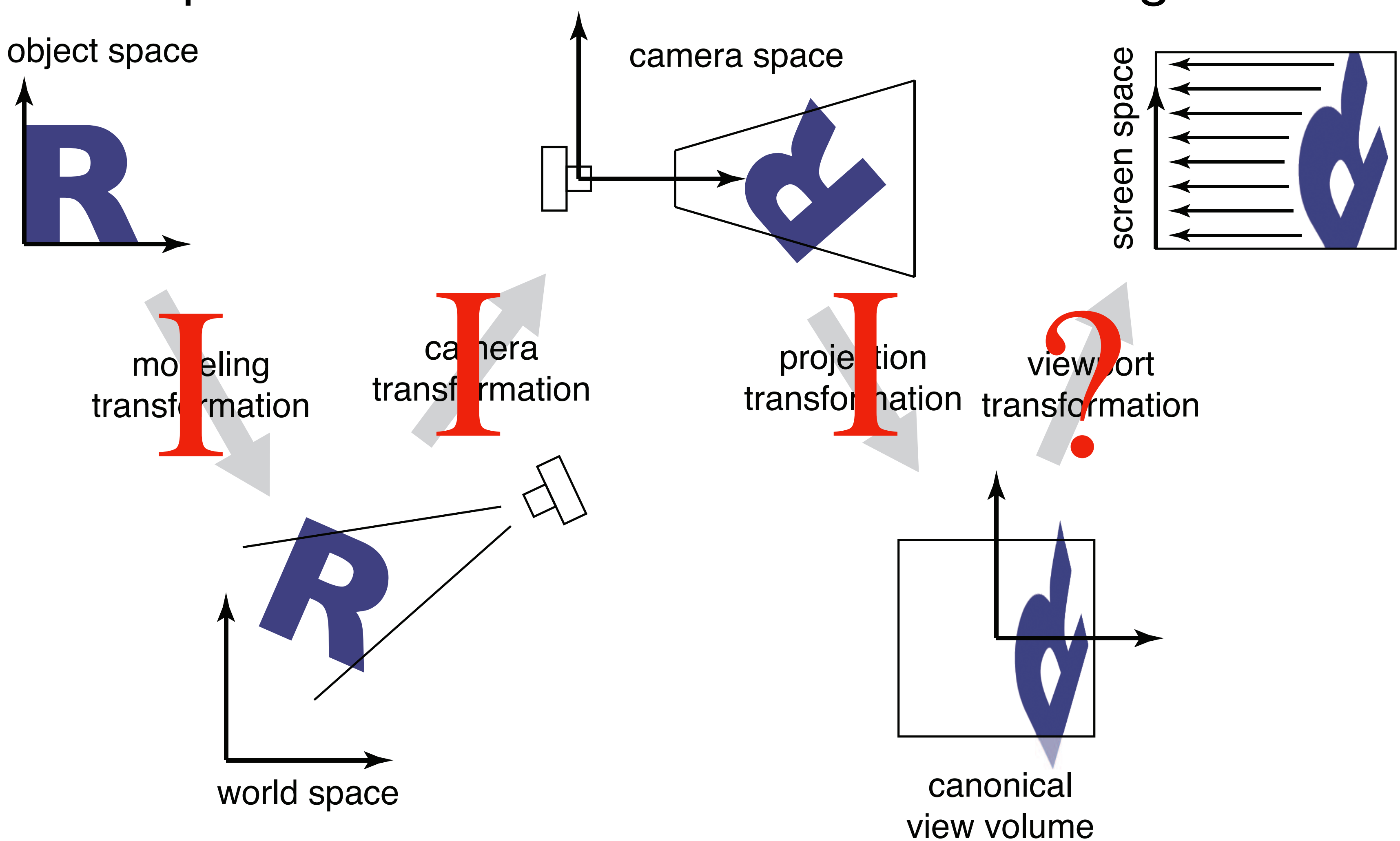
Let's do nothing and see how this works out...



A Wireframe Rendering Algorithm: Code

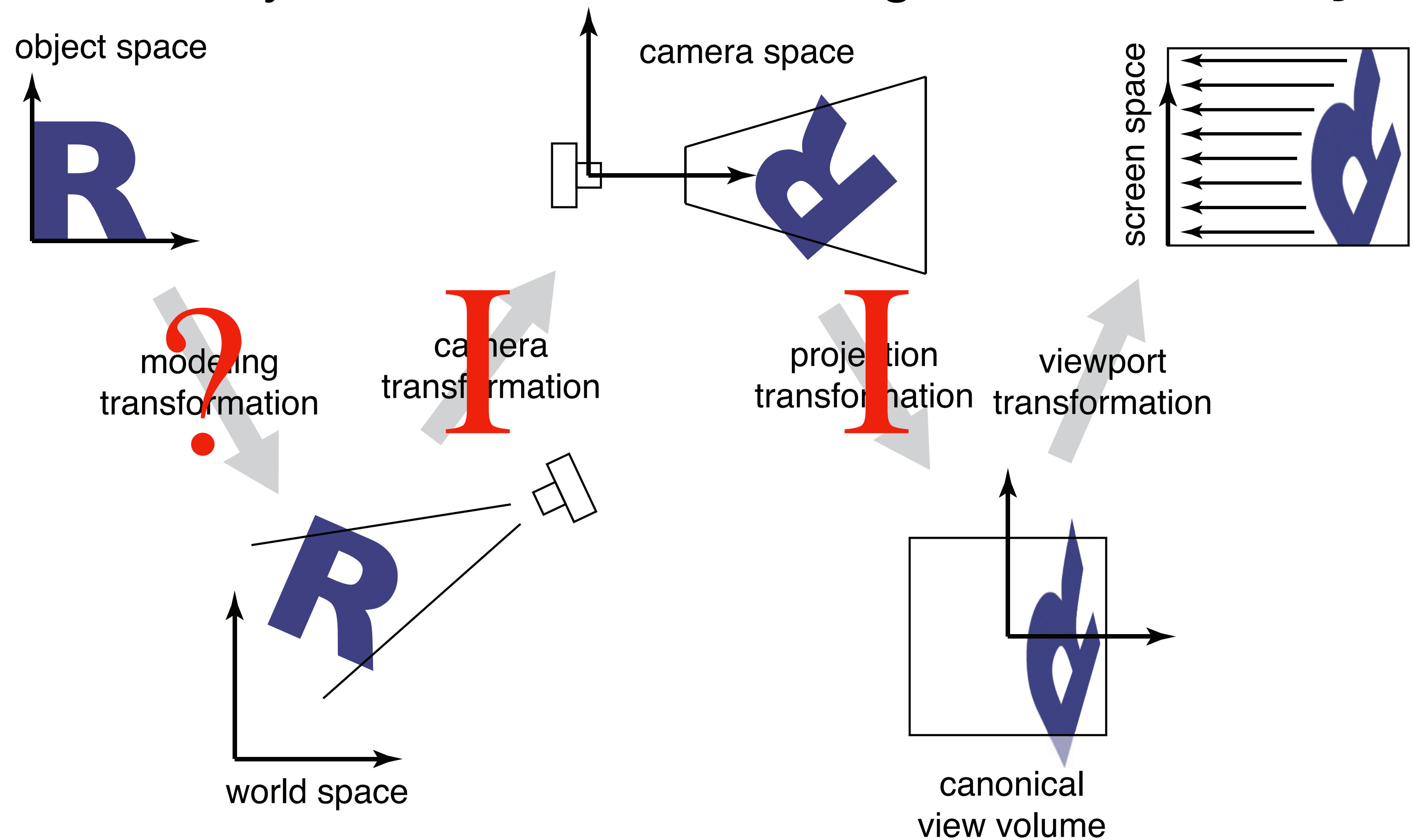
Viewing Transformations: Minimalist Edition

Task 1: Find a **viewport transformation** that puts the cube in the center of the image.



Viewing Transformations: Minimalist Edition

Task 2: Build a **model transformation** that centers a 40x40 cube at x=0, y=1, z=-4, rotated 30 degrees around the **y** axis.



Viewing Transformations: Minimalist Edition

Task 3: Move the camera 20 units in the +y direction (i.e., the new eye should be at (0, 20, 0).