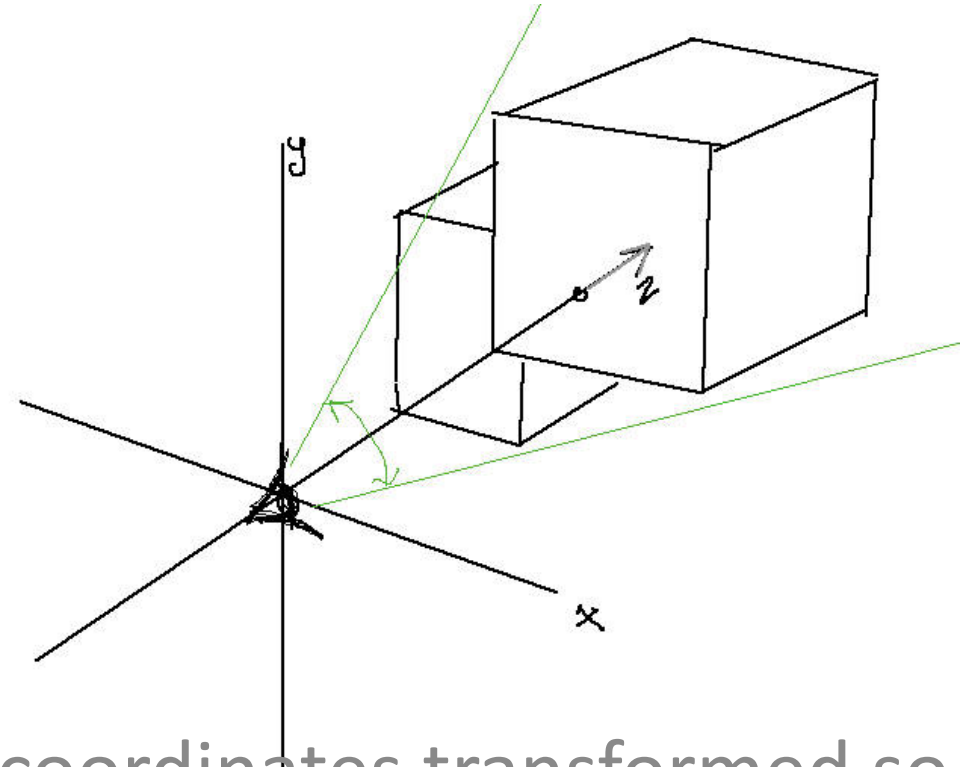
2: The Graphics Pipeline

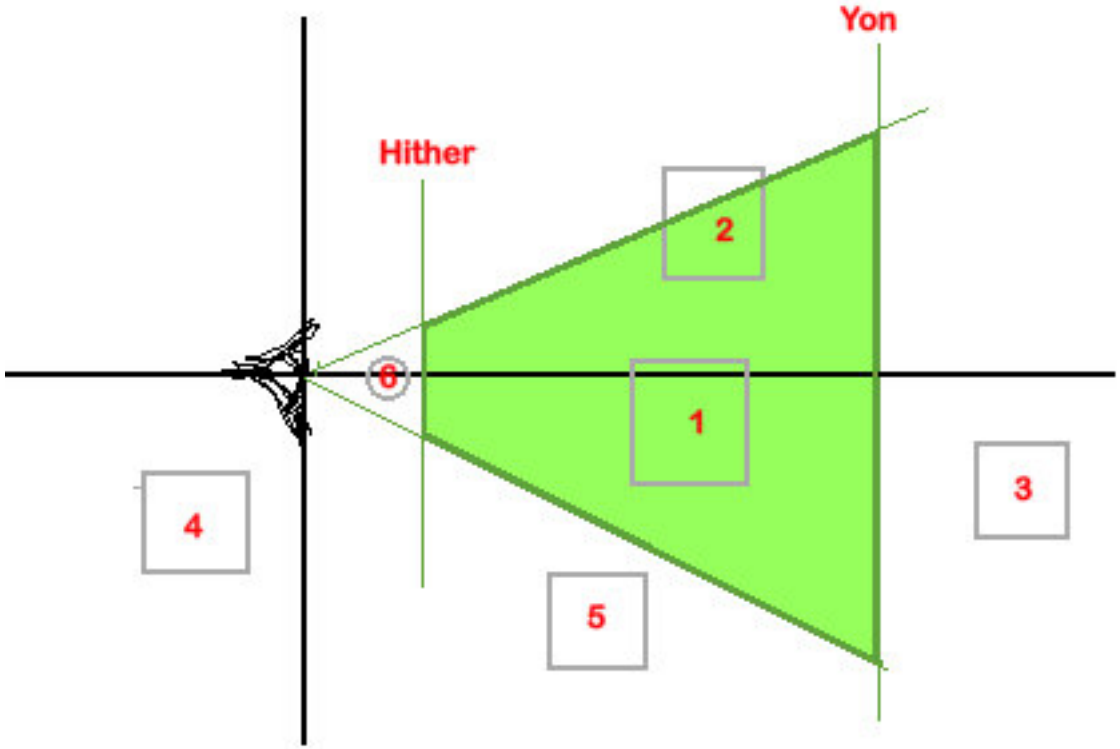
Arch 481

Eye-space transform



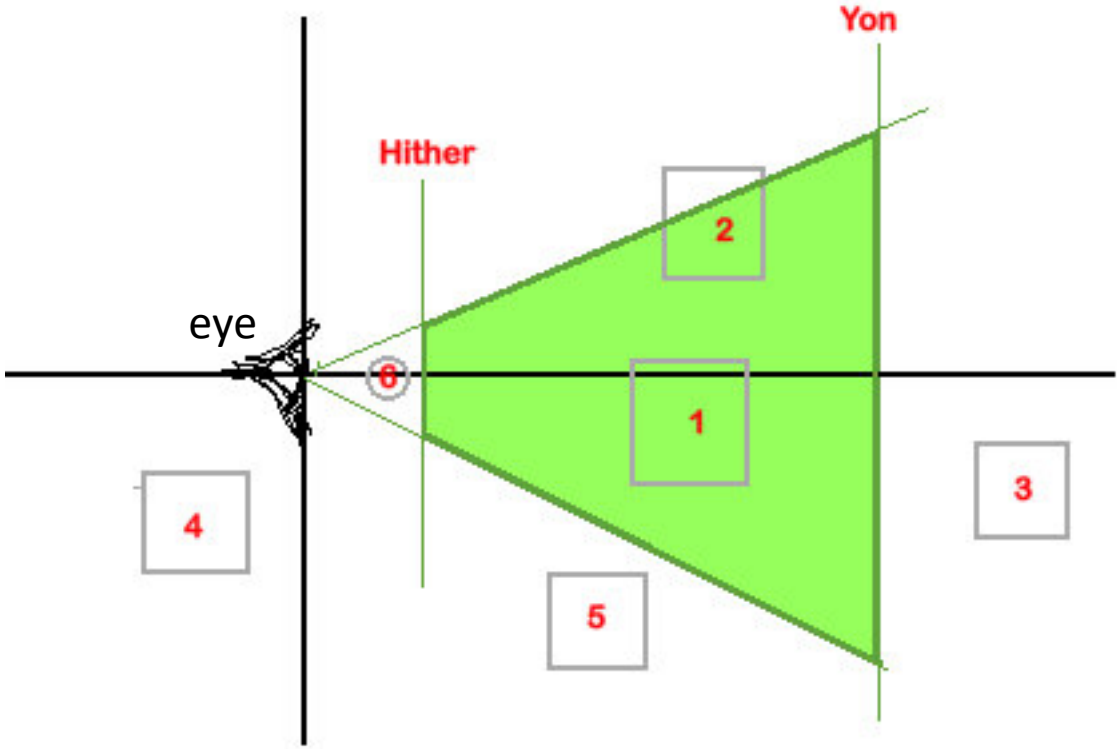
Model coordinates transformed so eye is at origin, with “image-x-axis” to the right and “image-y-axis” up (sometimes called u,v).

Clipping



Projection flattens ALL of 3D space onto the image plane, whether data is behind you (4) or off to the side (5). Clipping reduces it …

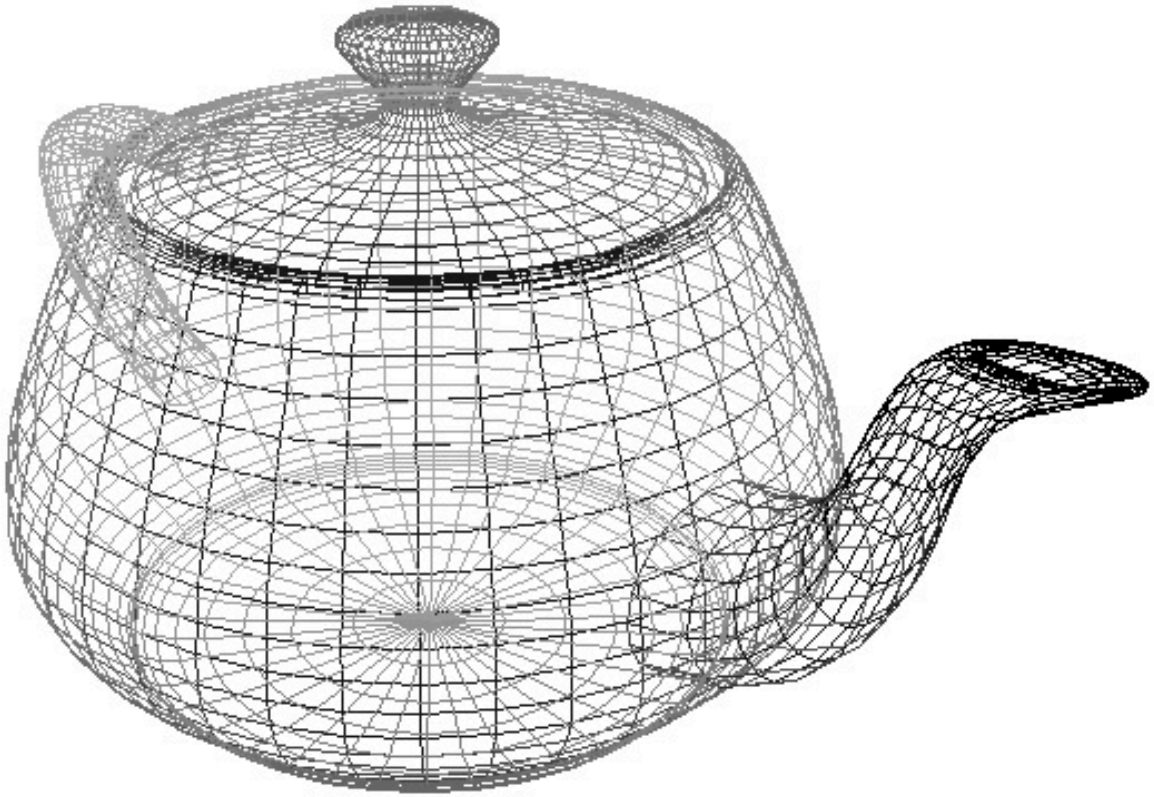
Clipping



Regular clipping will remove items 4 and 5 and trim 2. "Hither and Yon" clipping, if available, will not draw items 6 or 3 either.

Hidden Surfaces

This image uses "atmospheric perspective" (far lines are gray, near lines are black) to help viewers understand the model.



The default is for EVERYTHING to be visible (wireframe). "Hidden" is a decision the program must *make*.

Hidden Surfaces

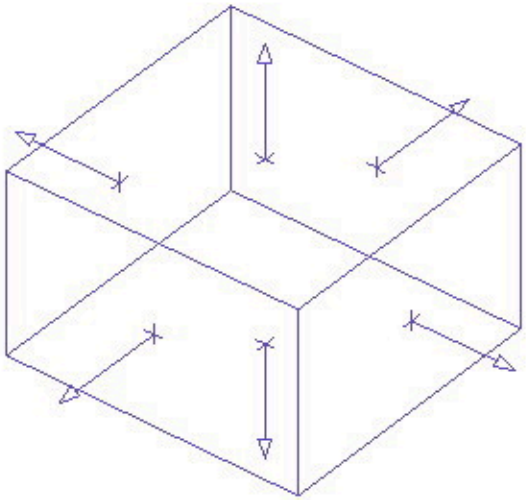
Multiple strategies have been developed to address this problem. Different renderers use different schemes or combinations of:

- Culling
- Depth sorting
- Z-buffer

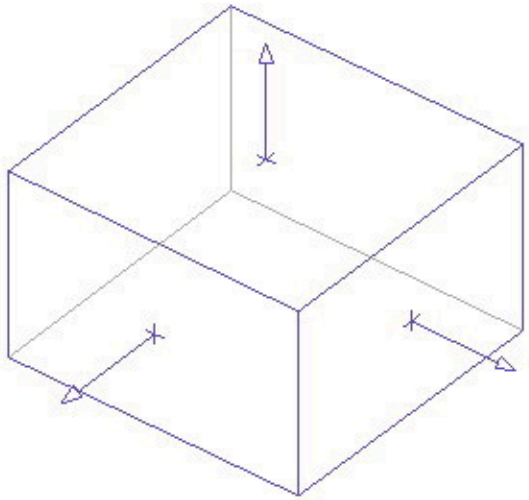
#1: "Culling" Hidden Surfaces

Surfaces have orientation (surface normals)

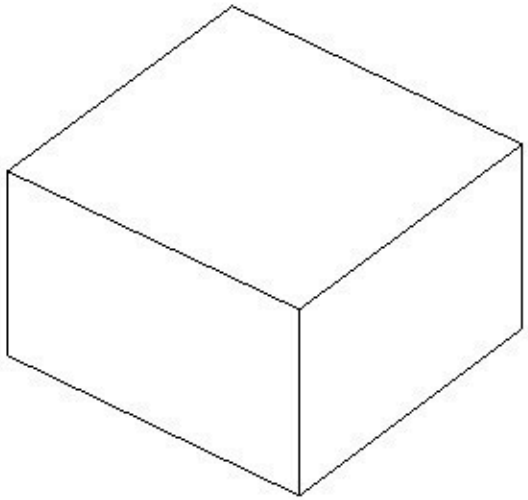
- **Normal**: a vector perpendicular to something
- You won't see what doesn't face you



All faces and normals



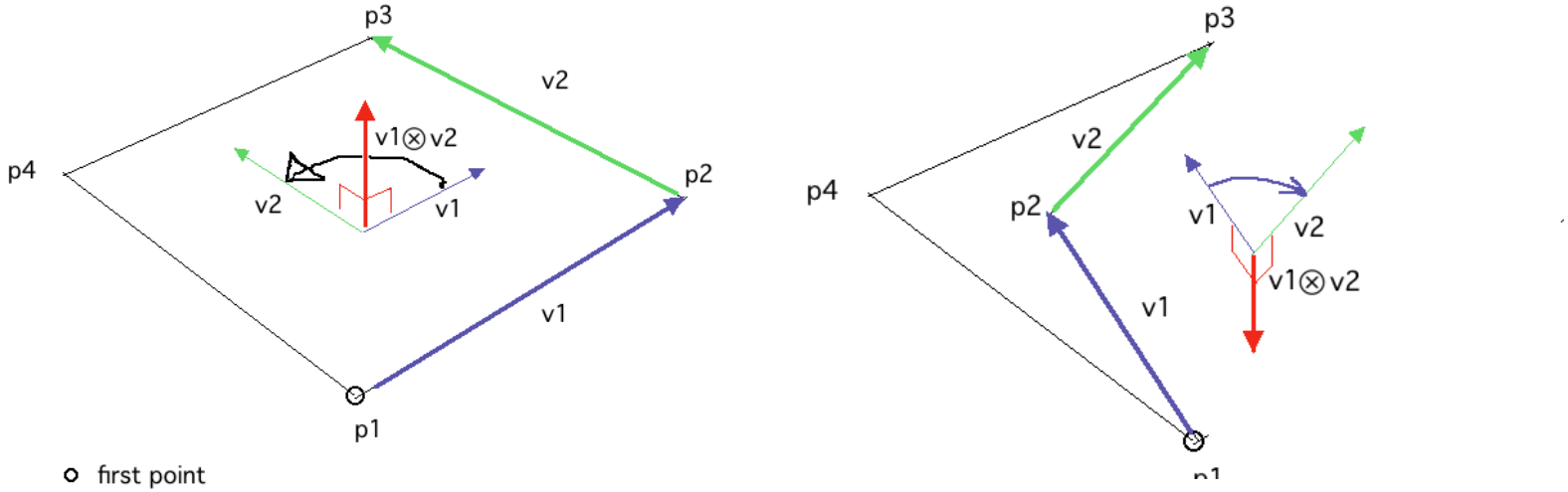
Faces facing the Camera



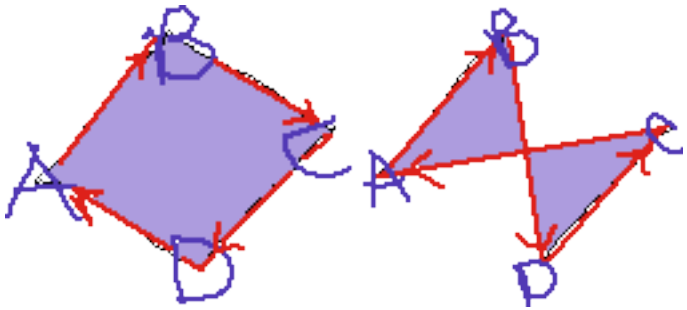
Finished Image

What can go wrong?

Orientation is computed from edges:

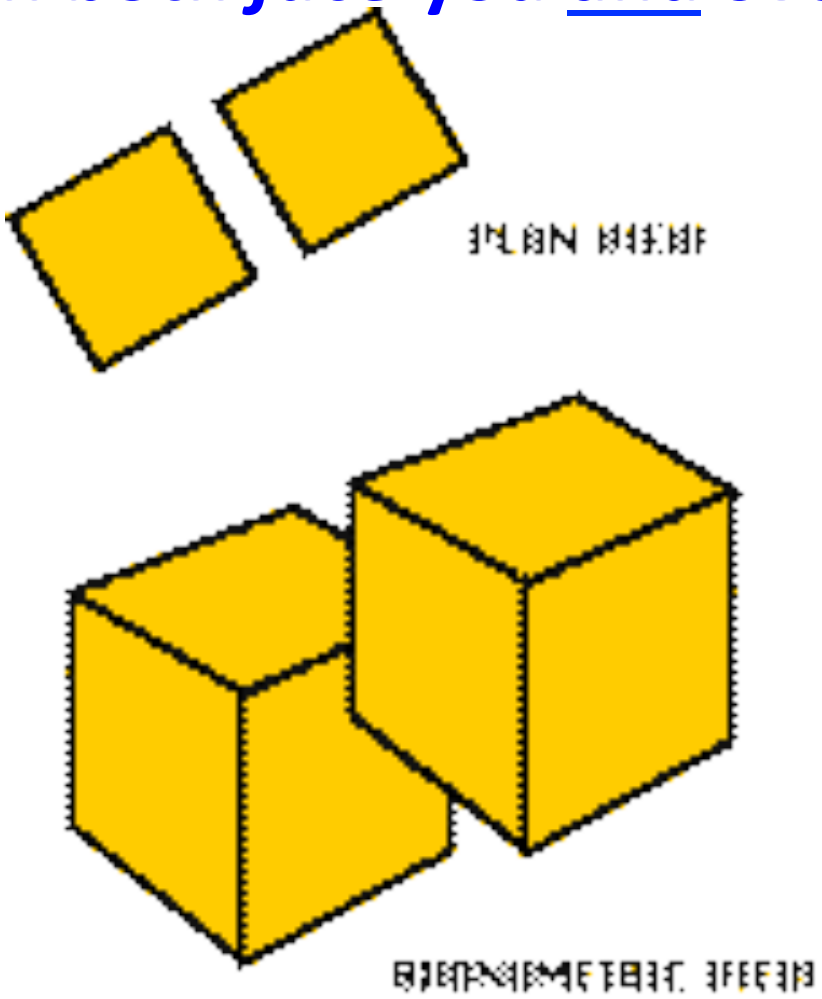


Polygons can get twisted or warped



What else can go wrong?

Polygons can both *face* you and *overlap*:



#2: “Depth Sorting” Surfaces

Eye space

- first step in graphics pipeline
- “+x” to the right, “+y” is up
- “+z” is distance away from the eye

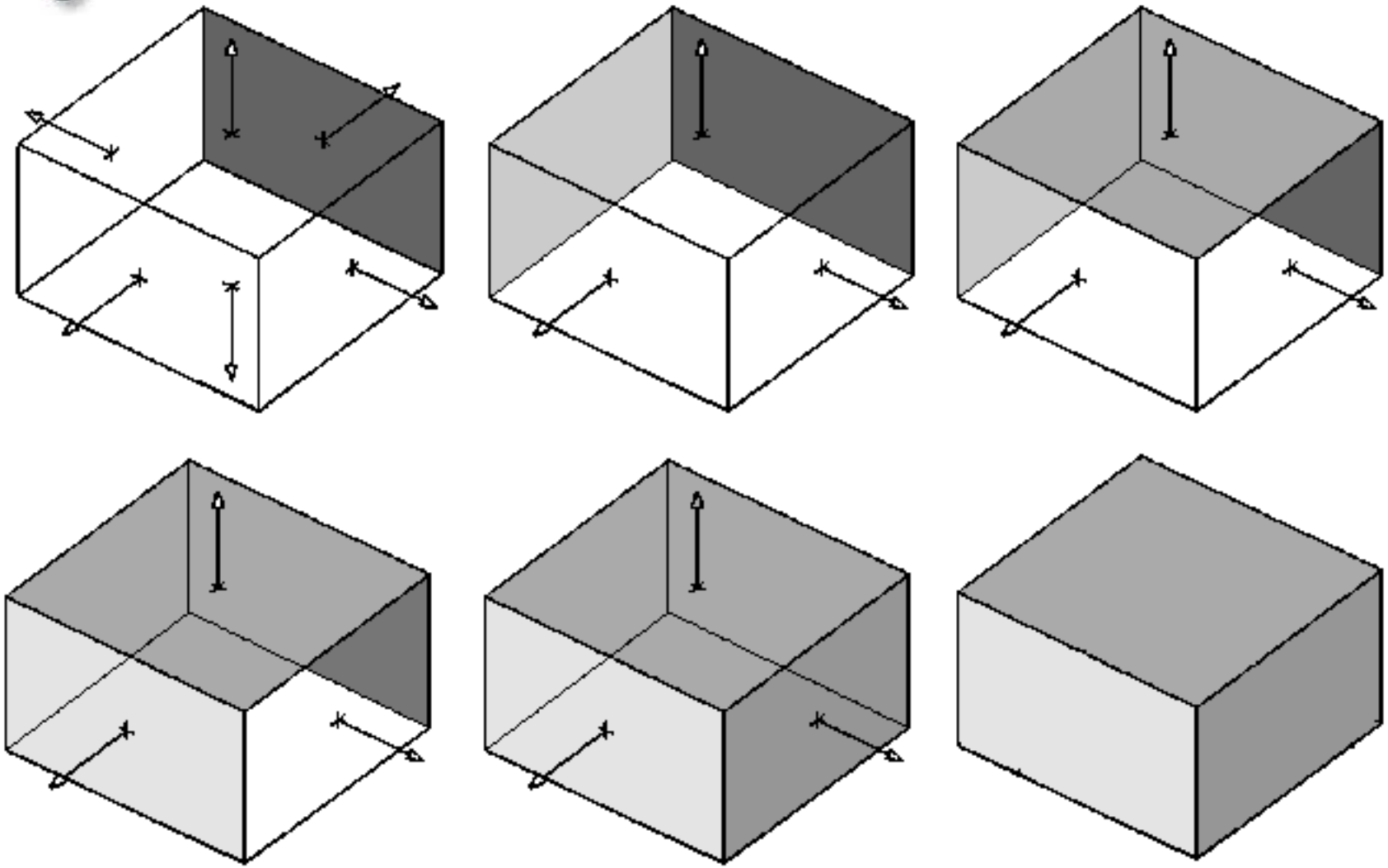
Raster screens support “destructive write”

- New stuff, drawn over old stuff, completely replaces it.

→ Draw from furthest away to nearest!

“Depth Sorting” Surfaces

Draw the most distant face first, based on view point, then the closer ones.

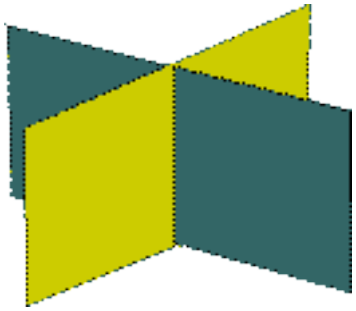


“Depth Sorting” errors…

Because it works with whole polygons…

Intersecting polygons like this …

(top view)



… that should look like this …

(axo view)

… will look like this instead!

#3: *Z-buffering (next week)*

Works with pixels, not polygons …

… works in “screen space” not “model space”

Screen space is NOT infinite (yay!)

Raster screen determines needed accuracy

Most uses will be raster.