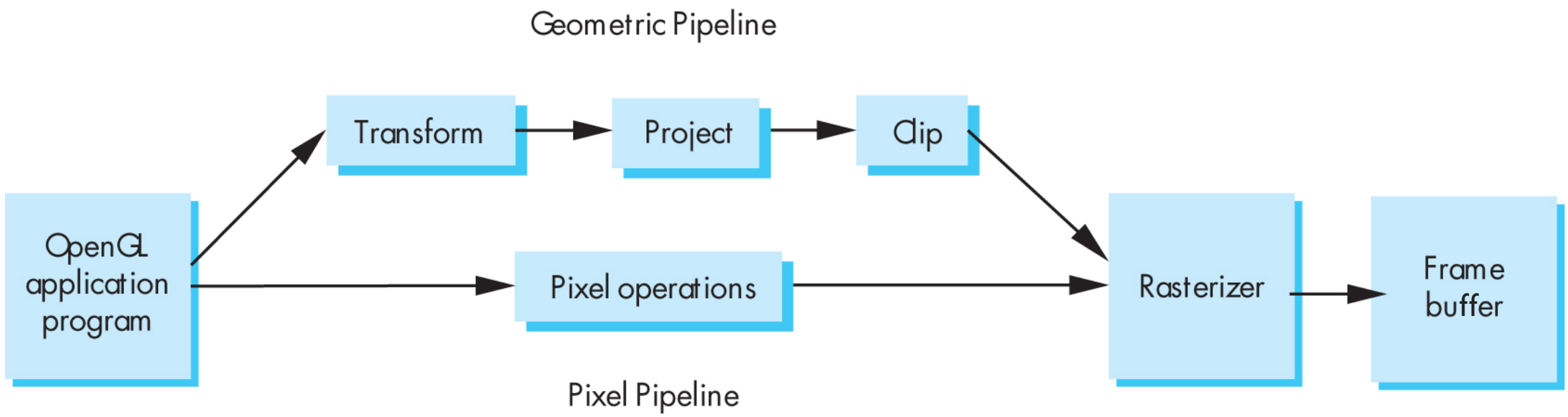
CS130 : Computer Graphics

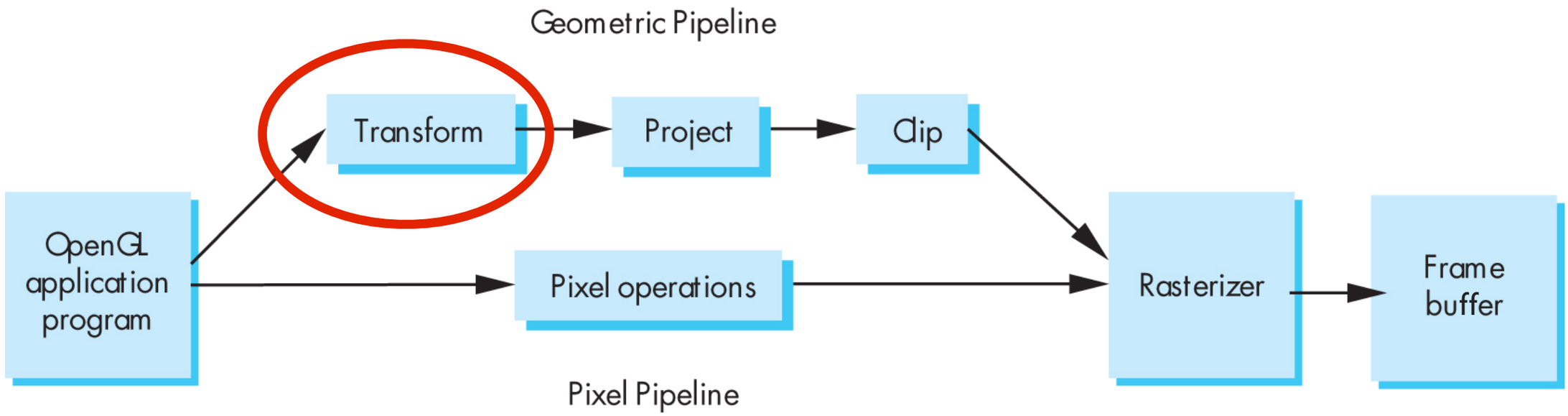
Lecture 6: Graphics Pipeline (cont.)

Tamar Shinar
Computer Science & Engineering
UC Riverside

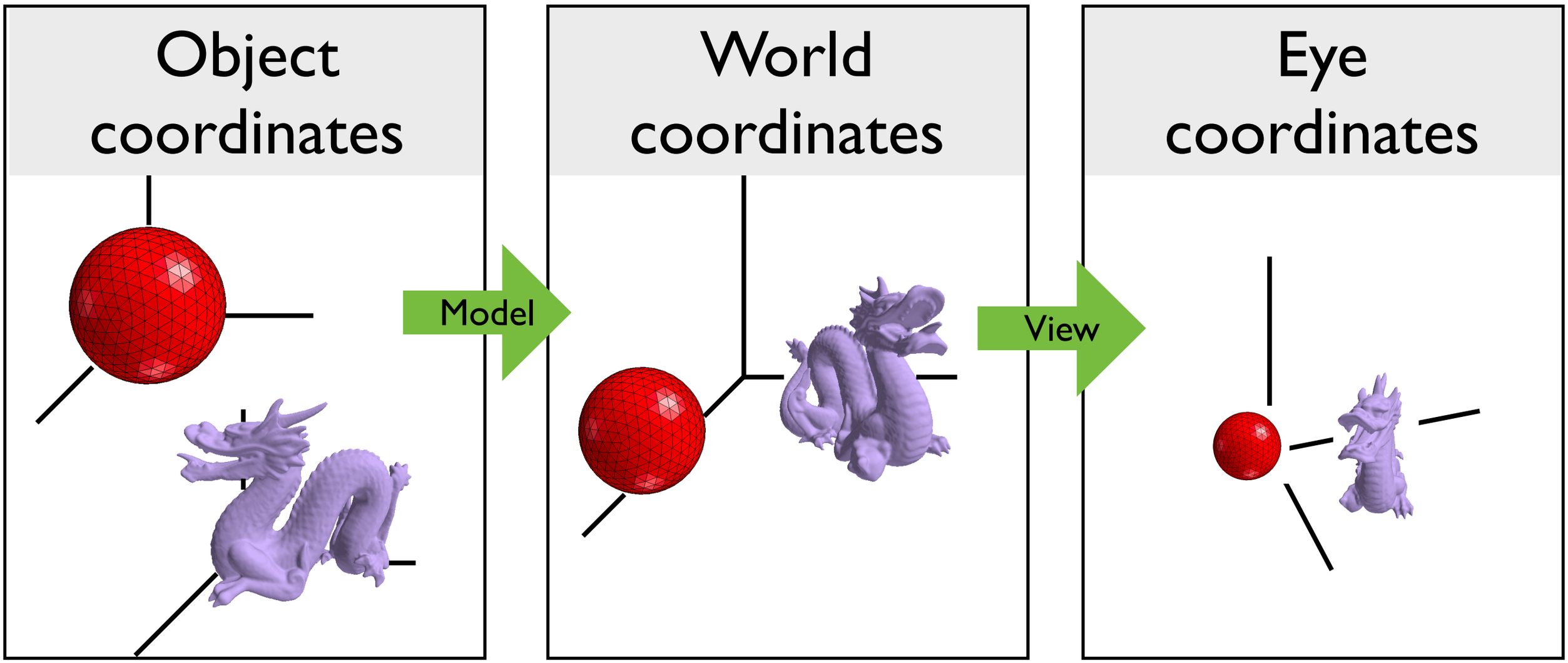
Graphics Pipeline



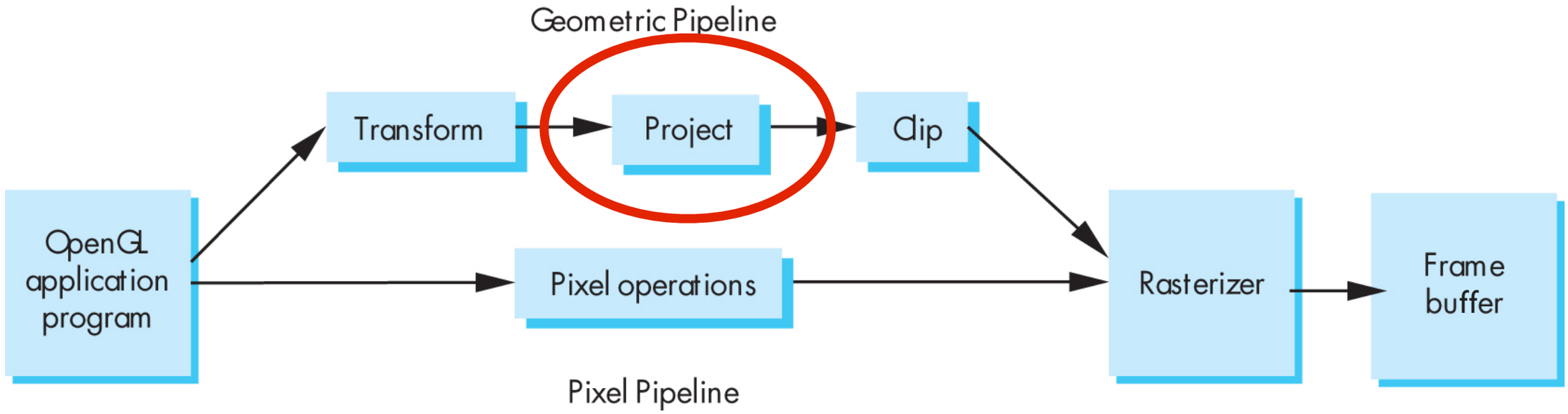
Transform


Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

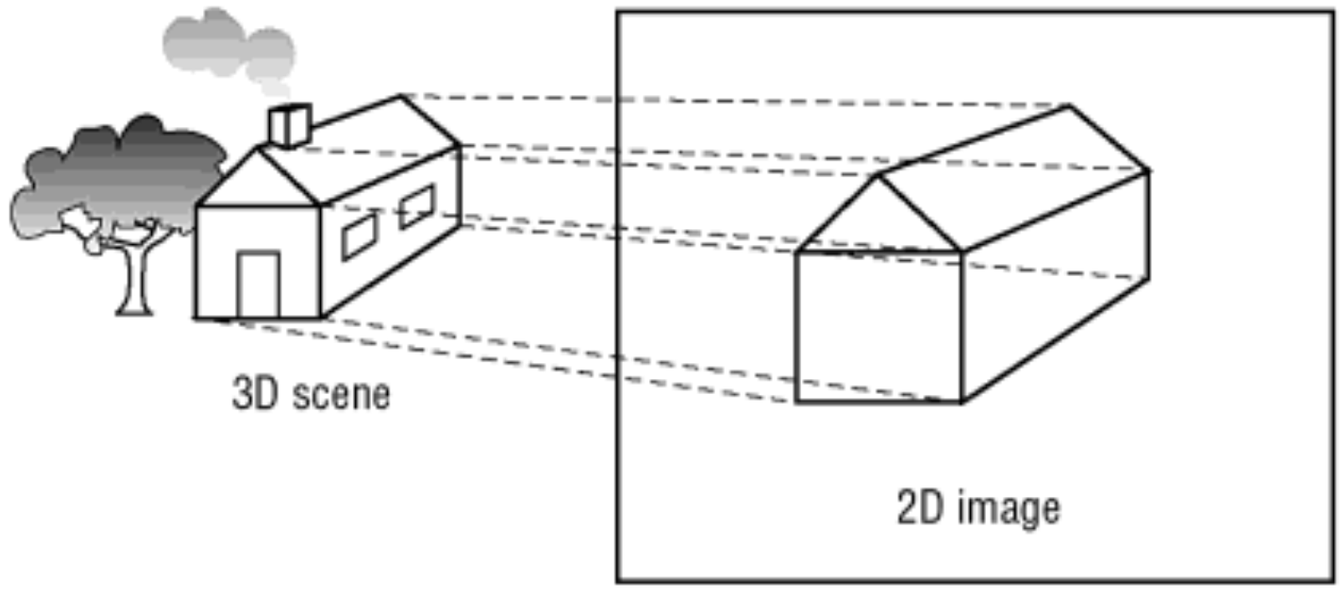
"Modelview" Transformation



Project

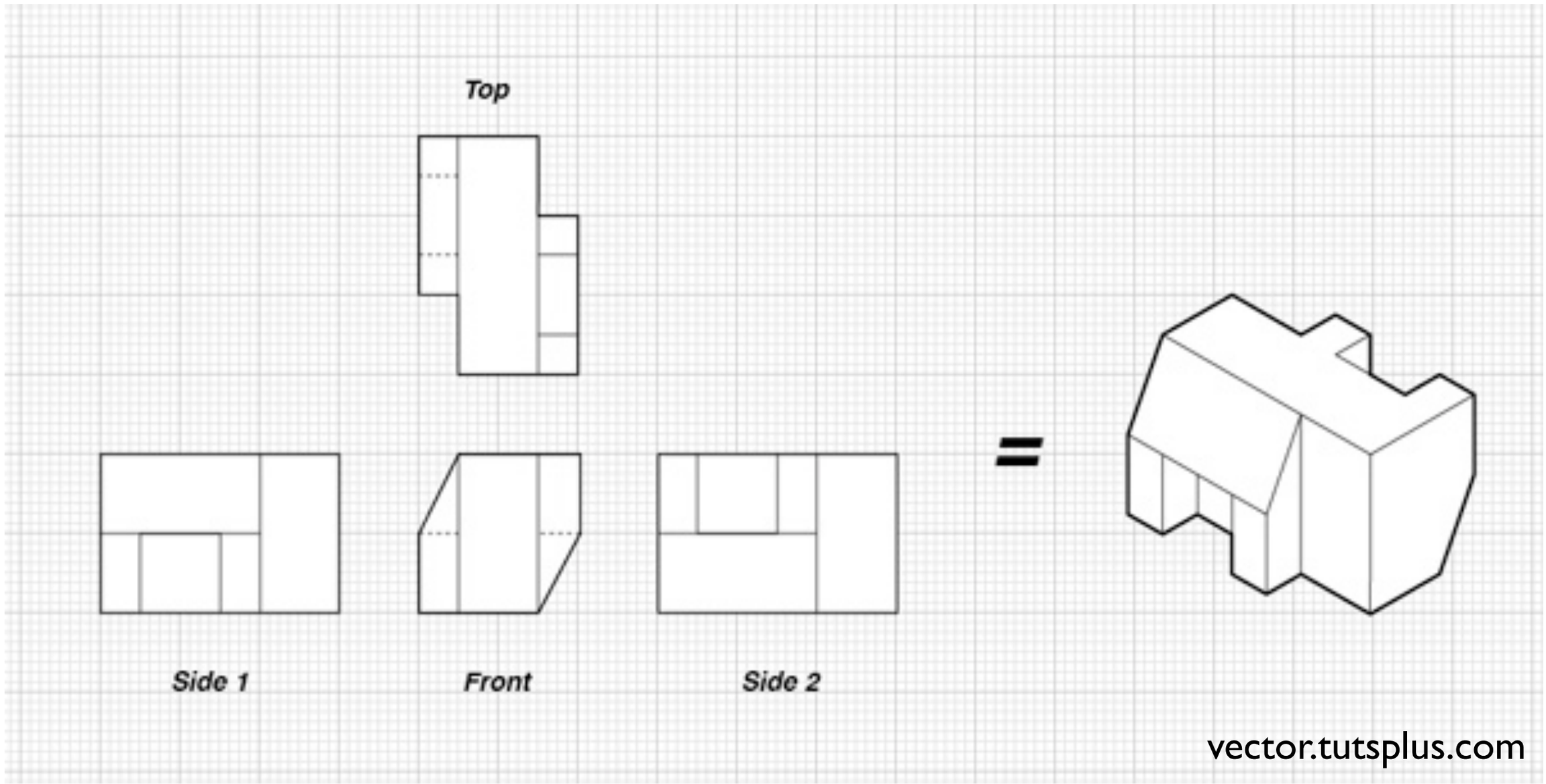

Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

Projection: map 3D scene to 2D image



OpenGL Super Bible, 5th Ed.

Orthographic projection

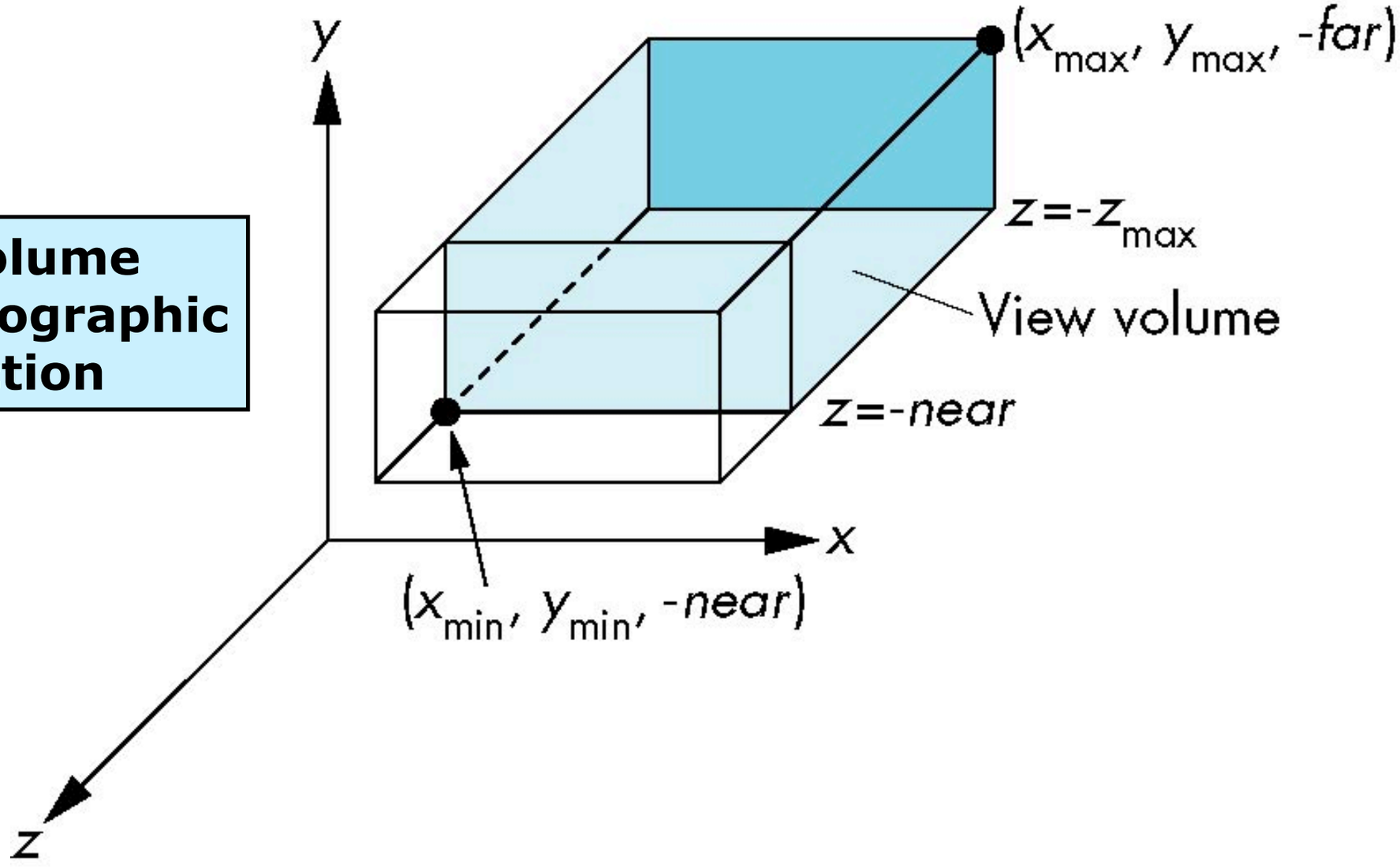


Orthographic, or parallel projection
- parallel lines appear parallel (unlike perspective proj.)
- equal length lines appear equal length (unlike perspective proj.)

OpenGL Orthogonal Viewing

`glOrtho(left,right,bottom,top,near,far)`

View volume for an orthographic projection

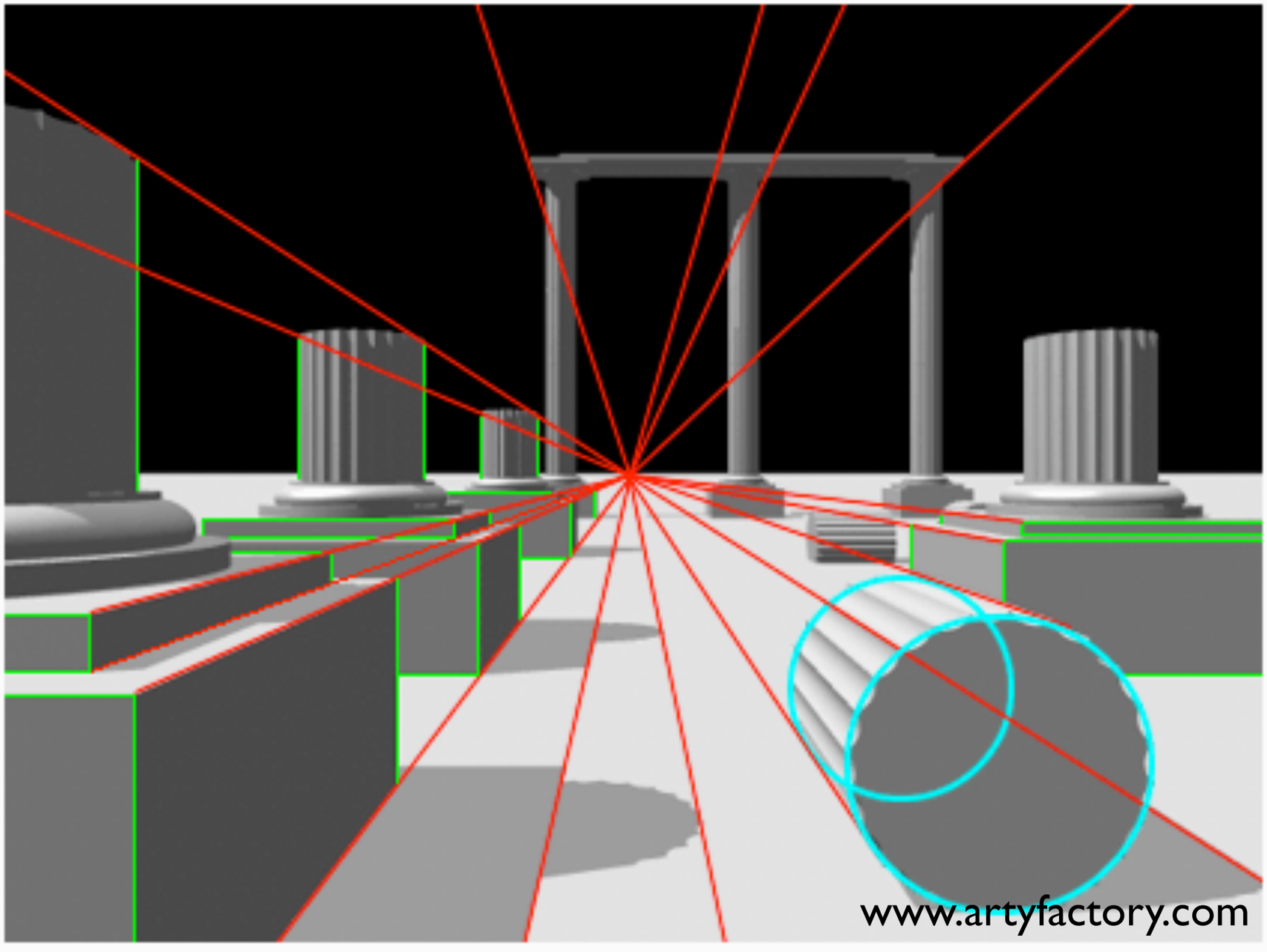


`near` and `far` measured from camera

Orthographic, or parallel projection

- square or rectangular viewing volume
- anything outside volume is not drawn

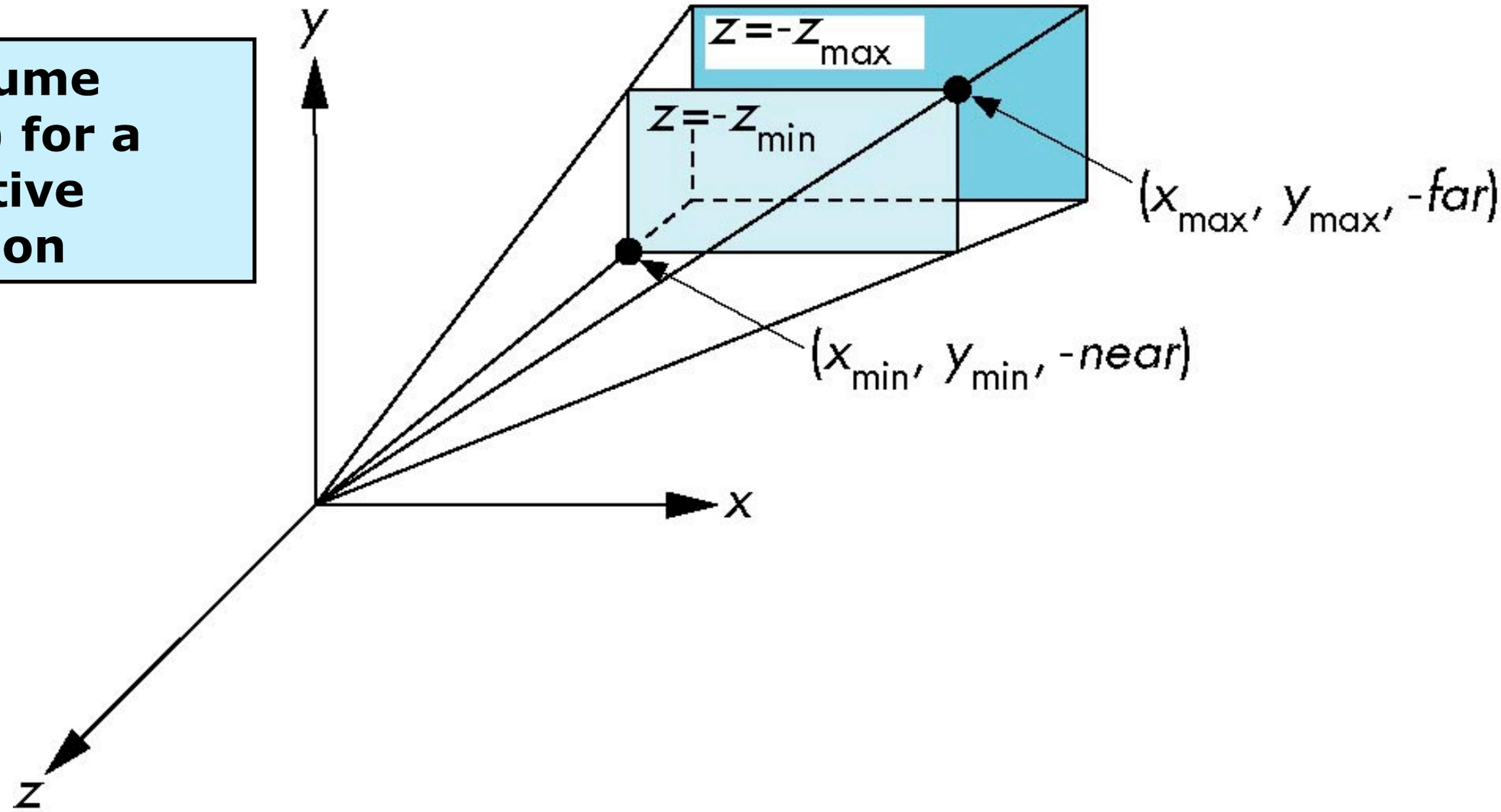
Perspective projection



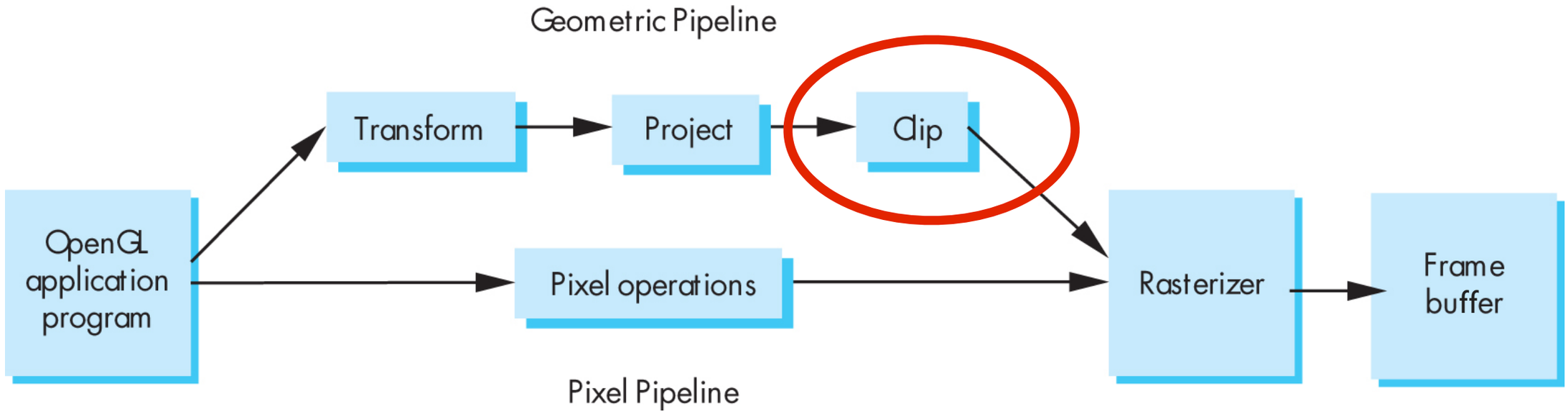
OpenGL Perspective Viewing

`glFrustum(xmin,xmax,ymin,ymax,near,far)`

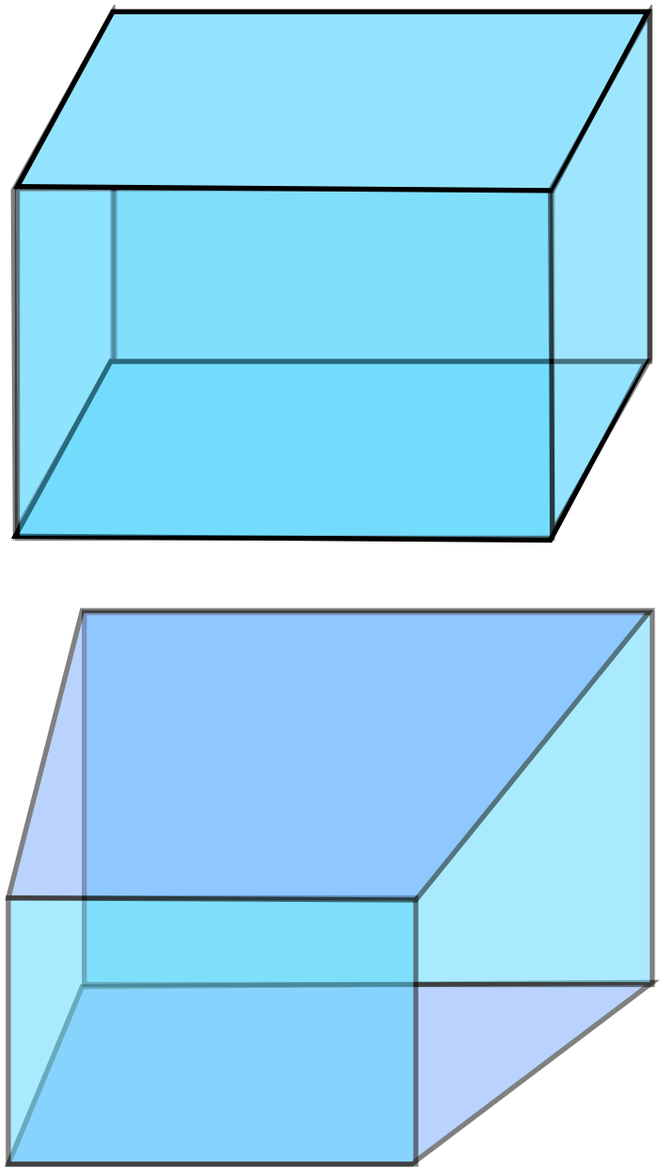
View volume (frustrum) for a perspective projection

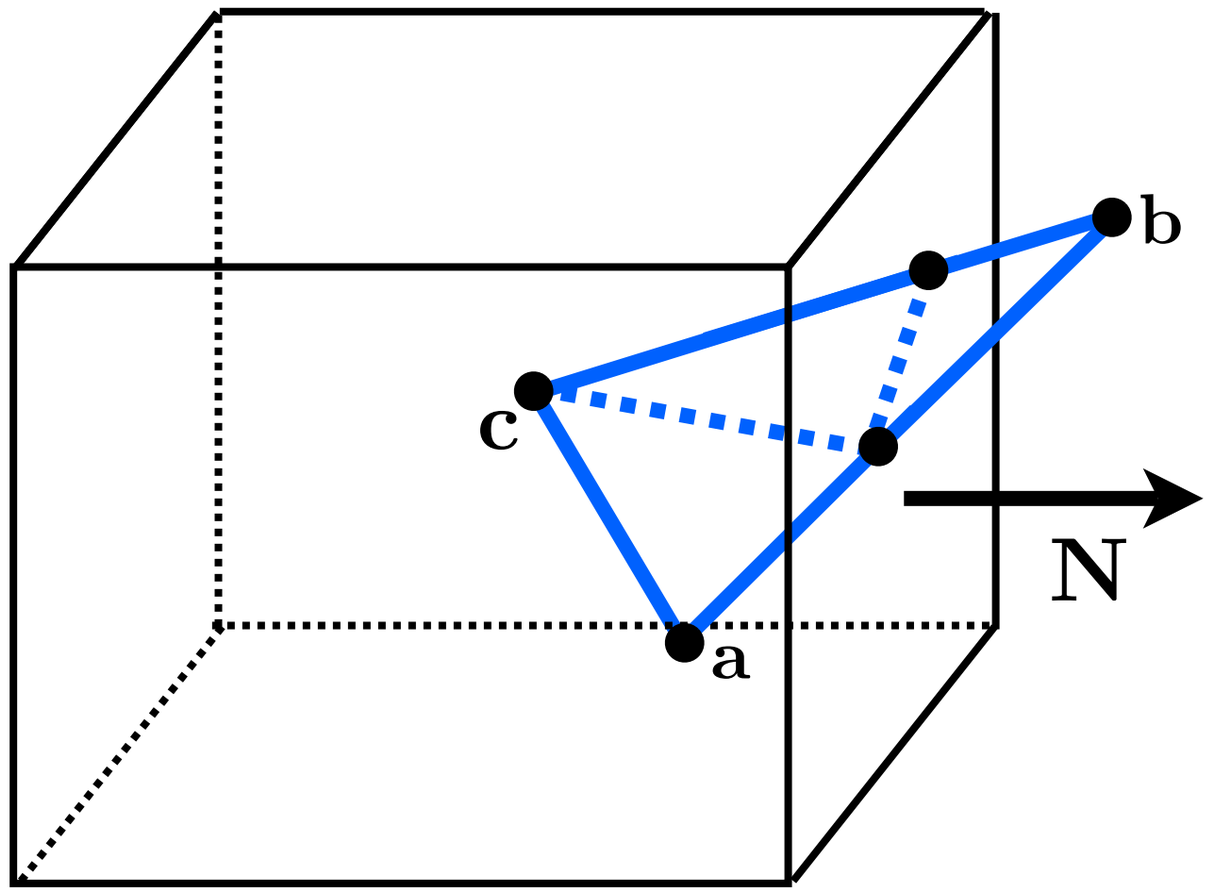


Clip


Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

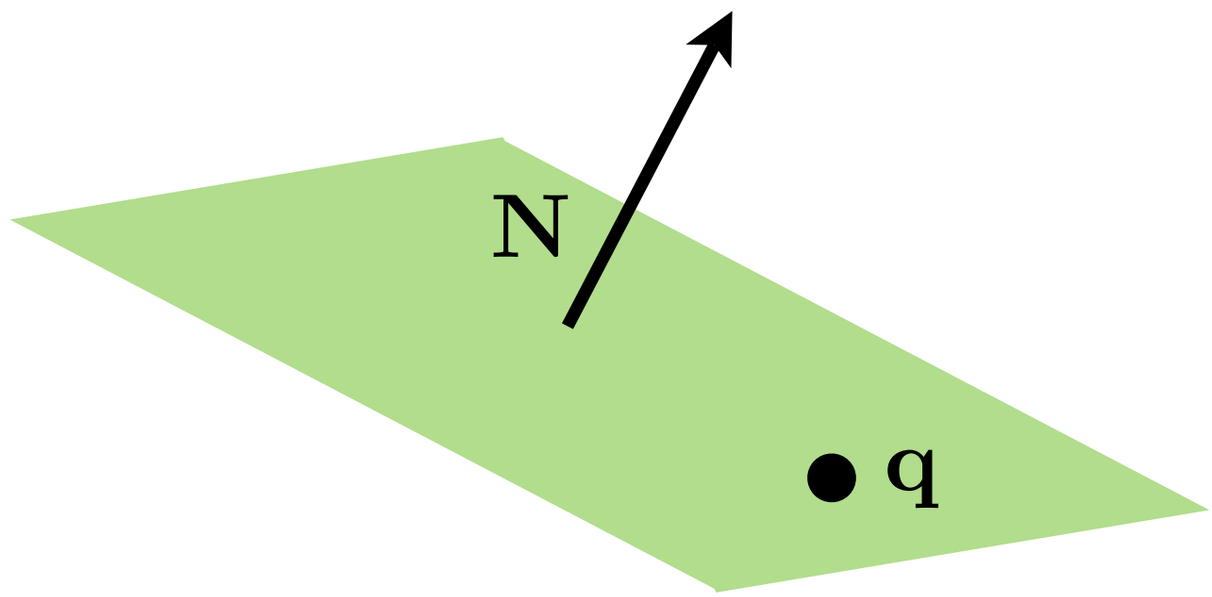
Clip against view volume

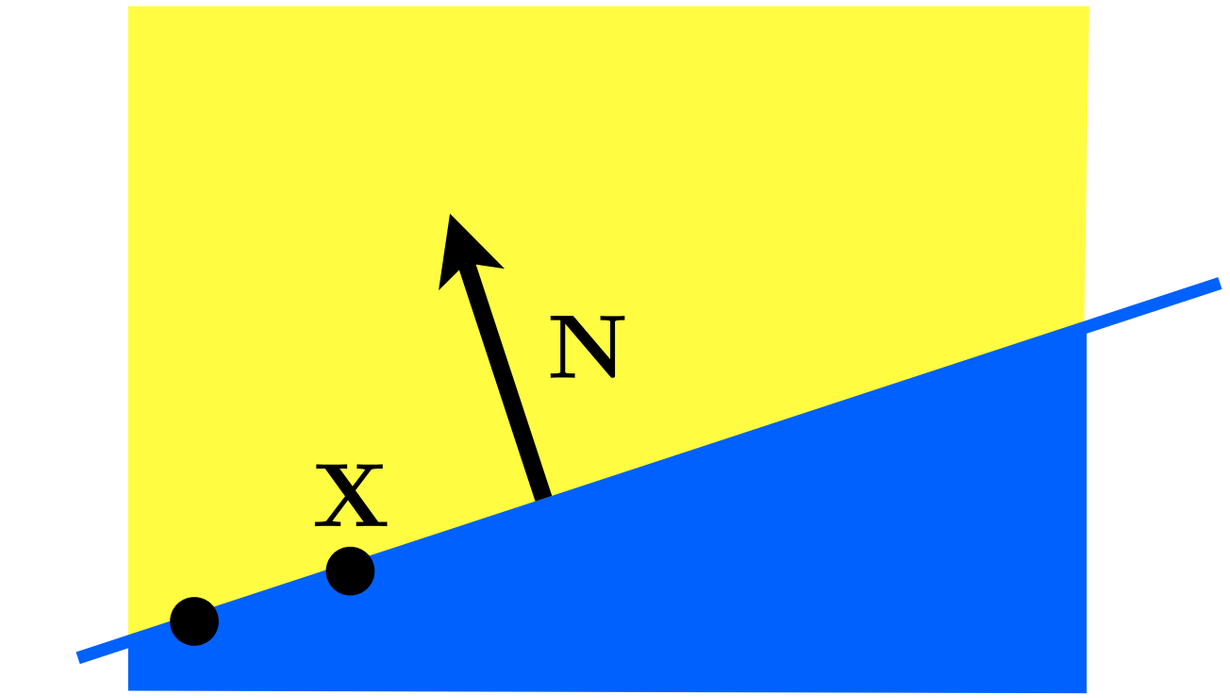




Clipping against a plane

What's the equation for the plane through **q** with normal **N**?





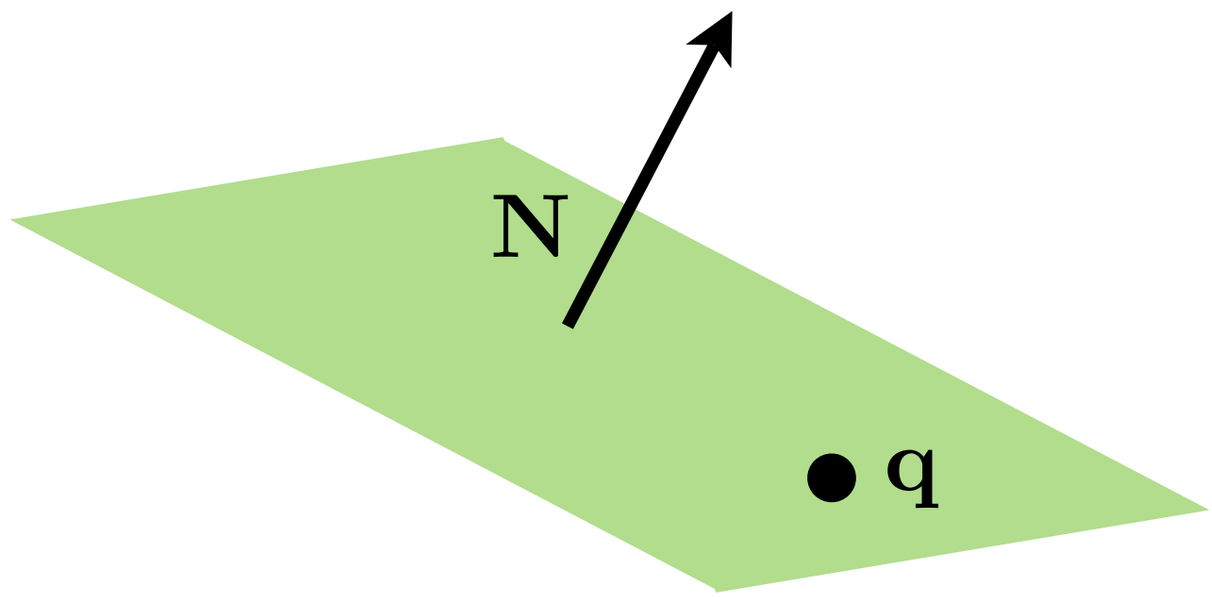
$\mathbf{X}_0 = (x_0, y_0)$

implicit line equation:

$$f(\mathbf{X}) = \mathbf{N} \cdot (\mathbf{X} - \mathbf{X}_0) = 0$$

Clipping against a plane

What's the equation for the plane through **q** with normal **N**?

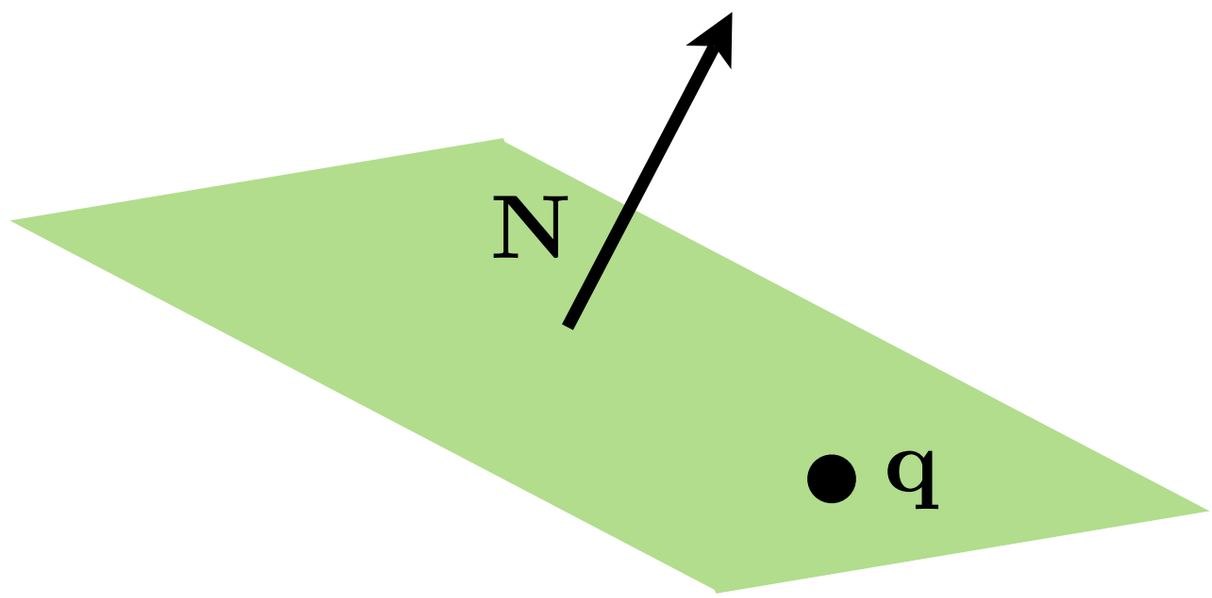


$$f(\mathbf{p}) =? = 0$$

<whiteboard>

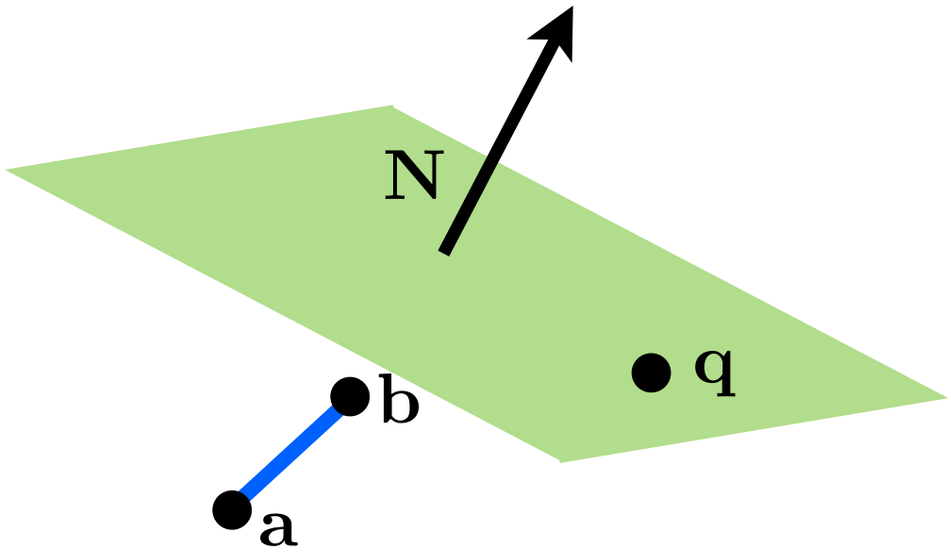
Clipping against a plane

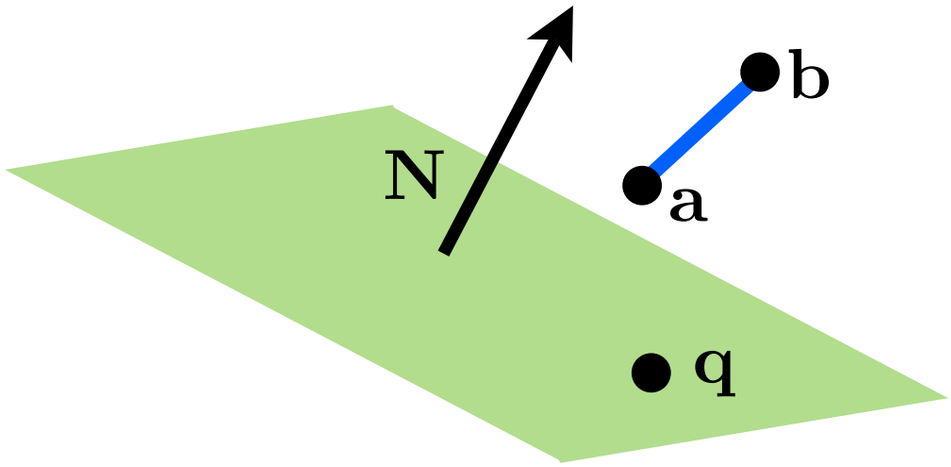
What's the equation for the plane through **q** with normal **N**?



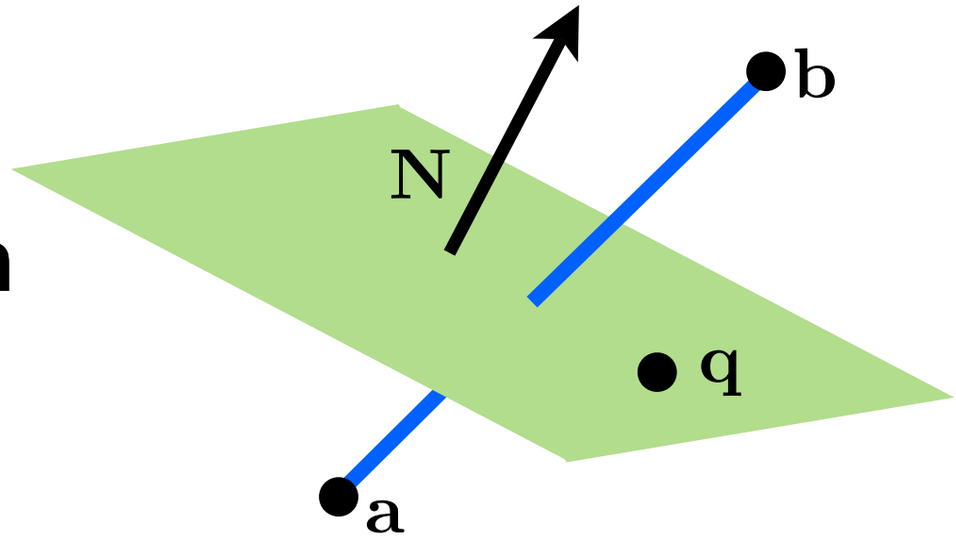
$$f(\mathbf{p}) = \mathbf{N} \cdot (\mathbf{p} - \mathbf{q}) = 0$$

Intersection of line and plane



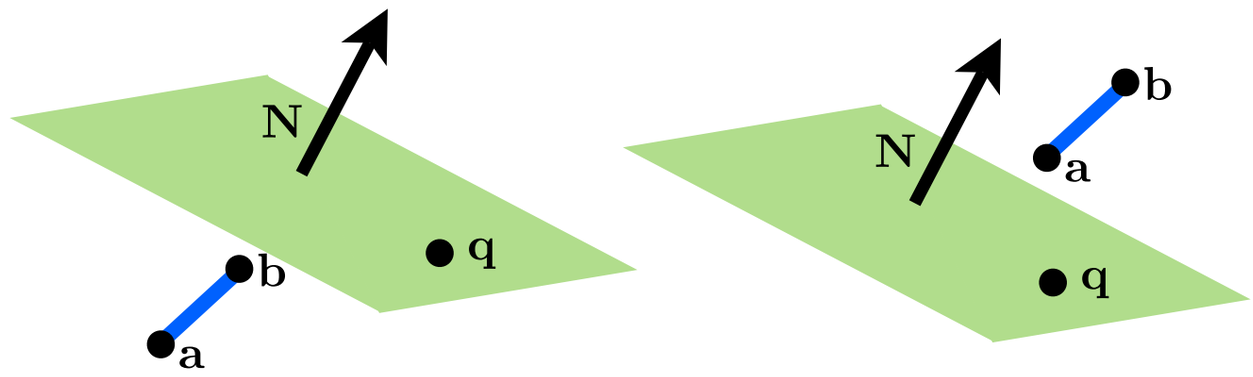


How can we distinguish between these cases?

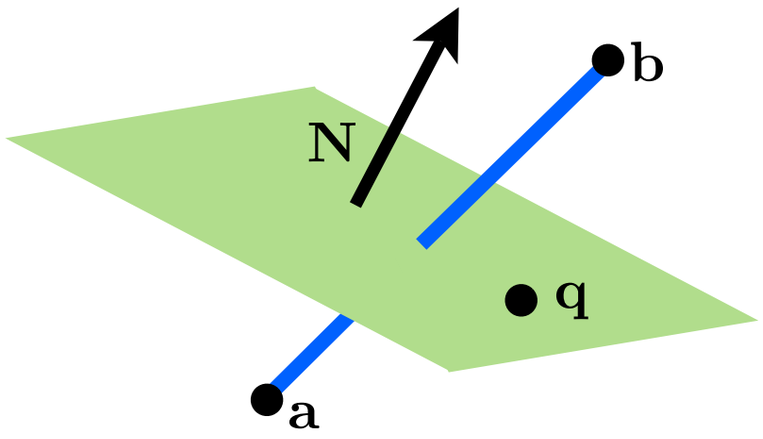


Intersection of line and plane

$$f(\mathbf{a})f(\mathbf{b}) \geq 0$$

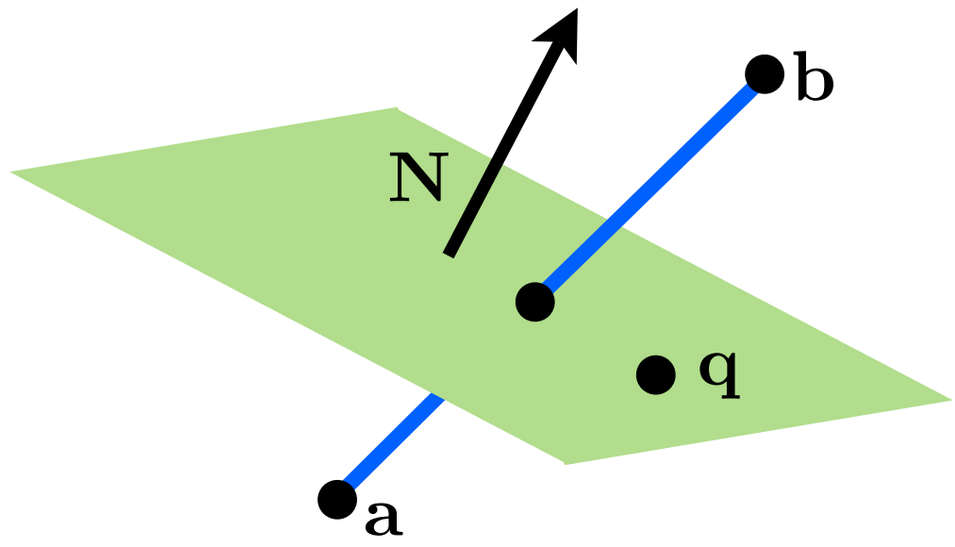


$$f(\mathbf{a})f(\mathbf{b}) < 0$$



Intersection of line and plane

How can we find the intersection point?



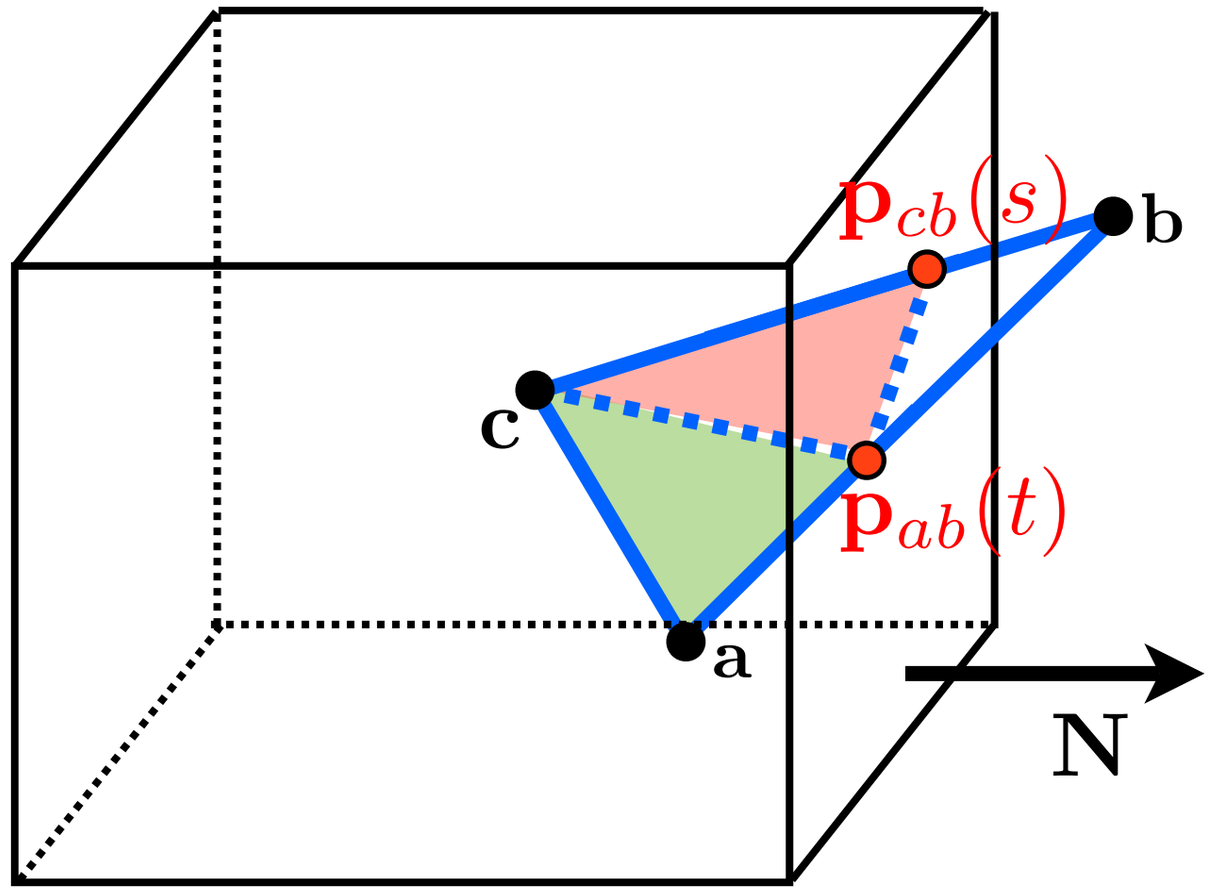
<whiteboard>

Clip against view volume

$$s = \frac{\mathbf{N} \cdot (\mathbf{q} - \mathbf{c})}{\mathbf{N} \cdot (\mathbf{b} - \mathbf{c})}$$

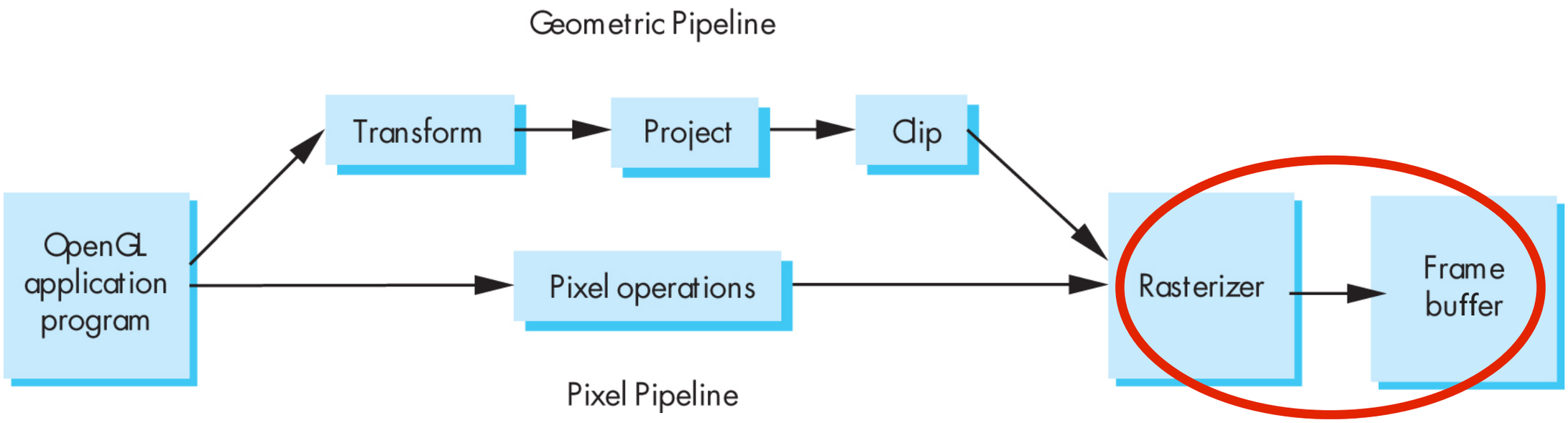
$$t = \frac{\mathbf{N} \cdot (\mathbf{q} - \mathbf{a})}{\mathbf{N} \cdot (\mathbf{b} - \mathbf{a})}$$

need to generate new triangles

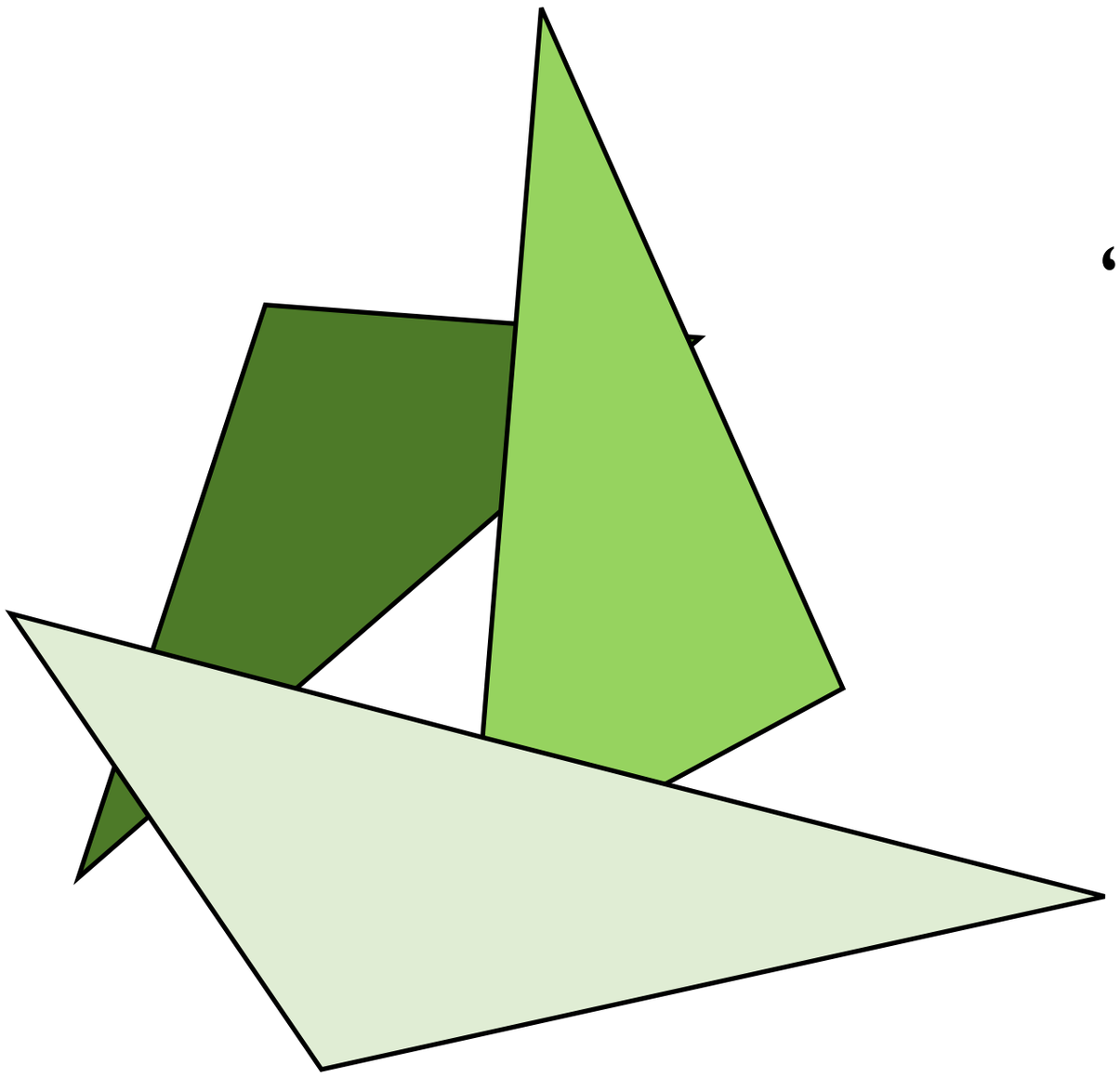


We write down the line equations p_cb(s) and p_ab(t) and find the s and t where they intersect the plane.

Hidden Surface Removal

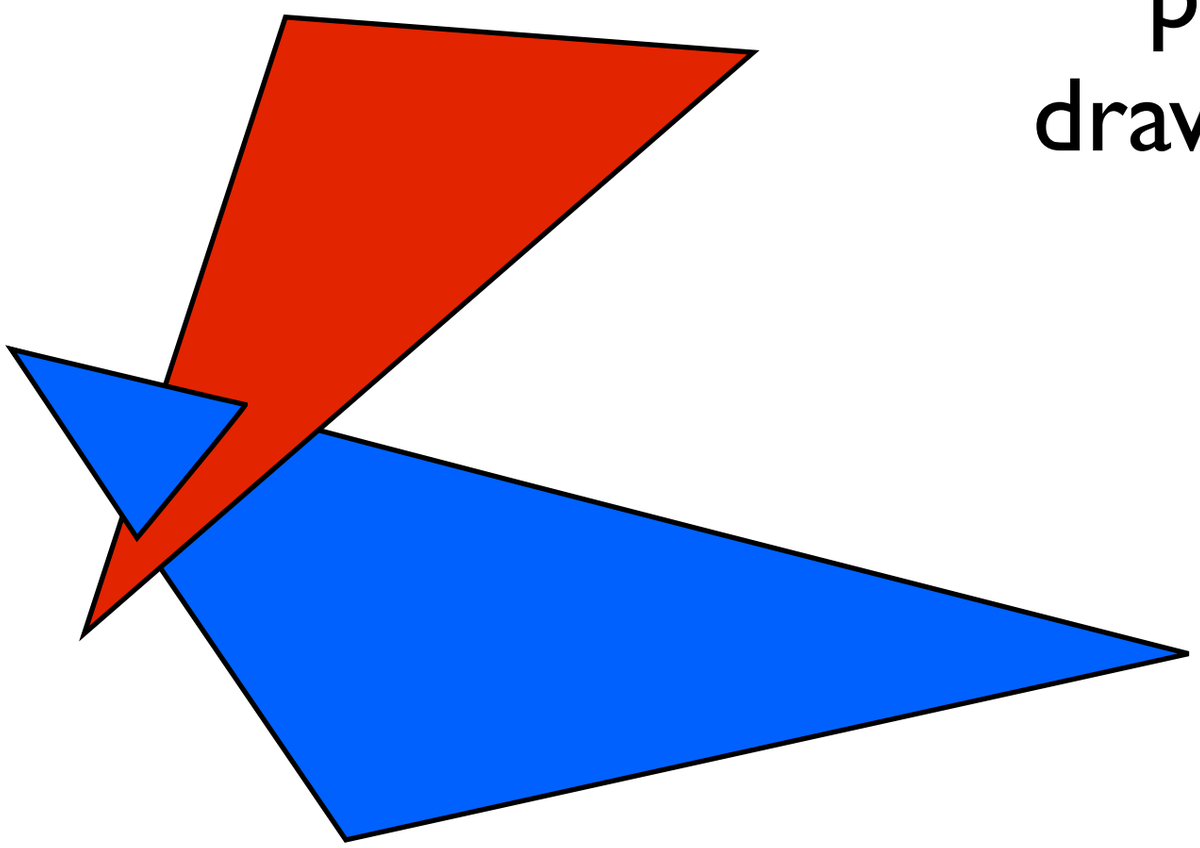


Occlusion



"painter's algorithm"
draw primitives in
back-to-front order

Occlusion

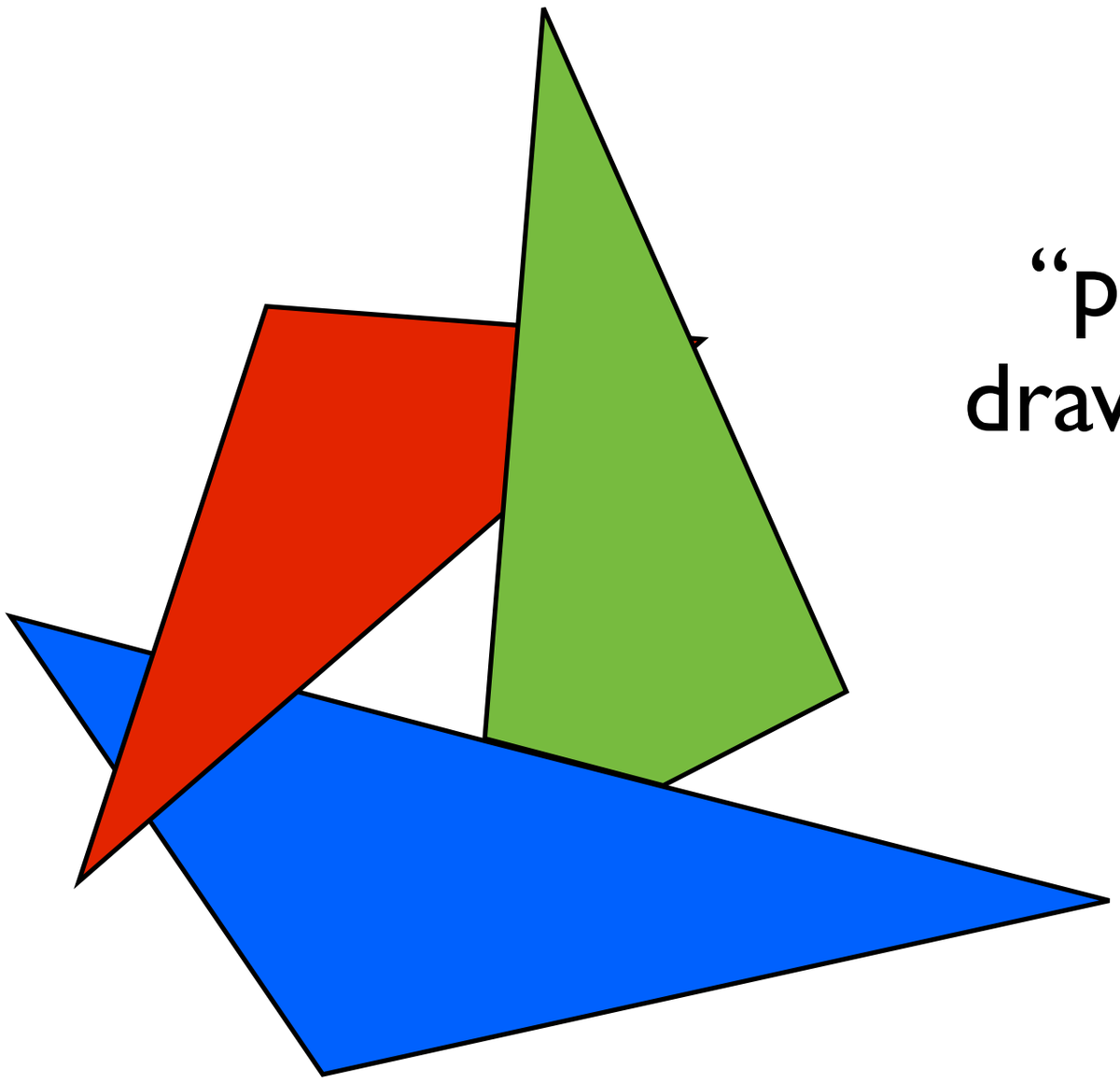


"painter's algorithm"
draw primitives in back-to-front order

problem:
triangle intersection

who's in front of whom?

Occlusion



"painter's algorithm"
draw primitives in back-to-front order

problem:
occlusion cycle

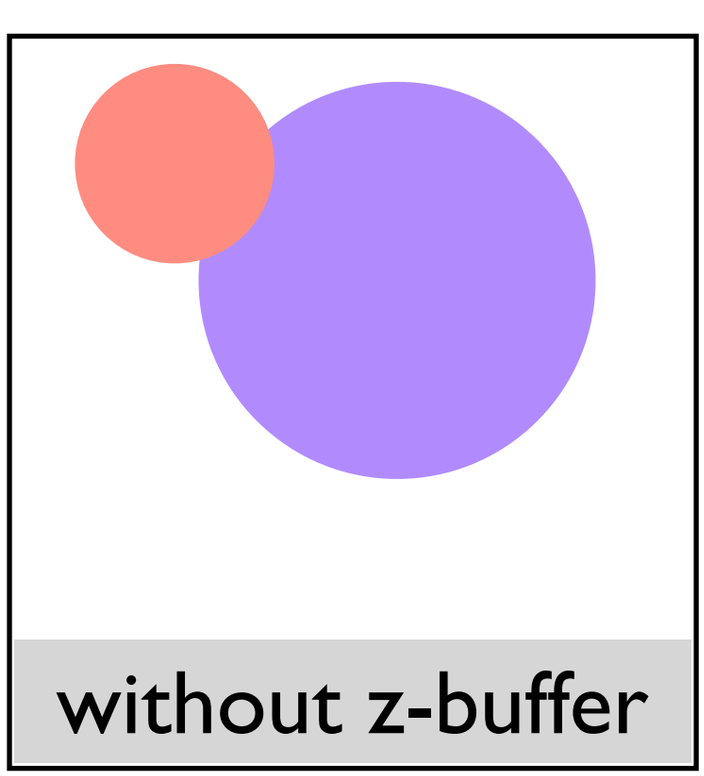
also, sorting primitives by depth is **slow**

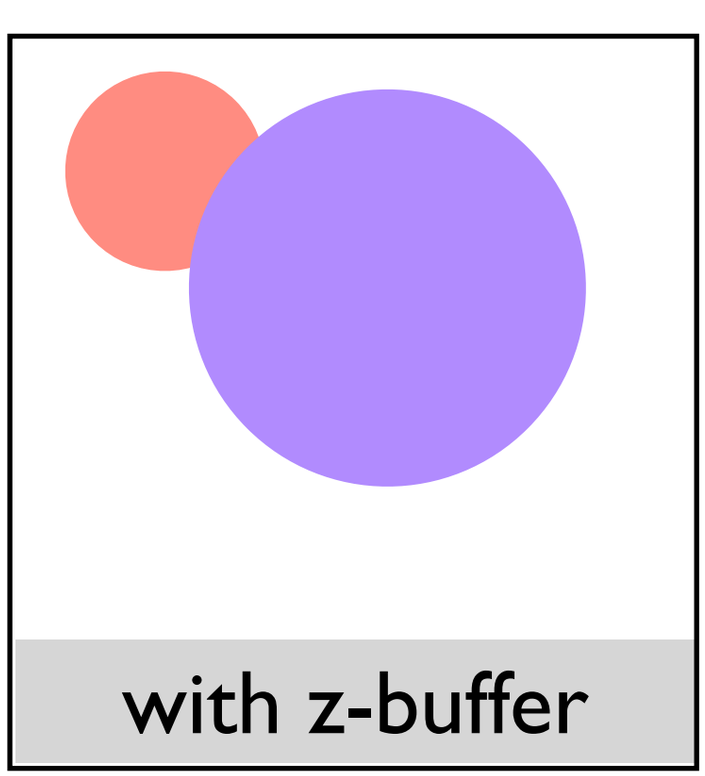
Use a *z-buffer* for hidden surface removal

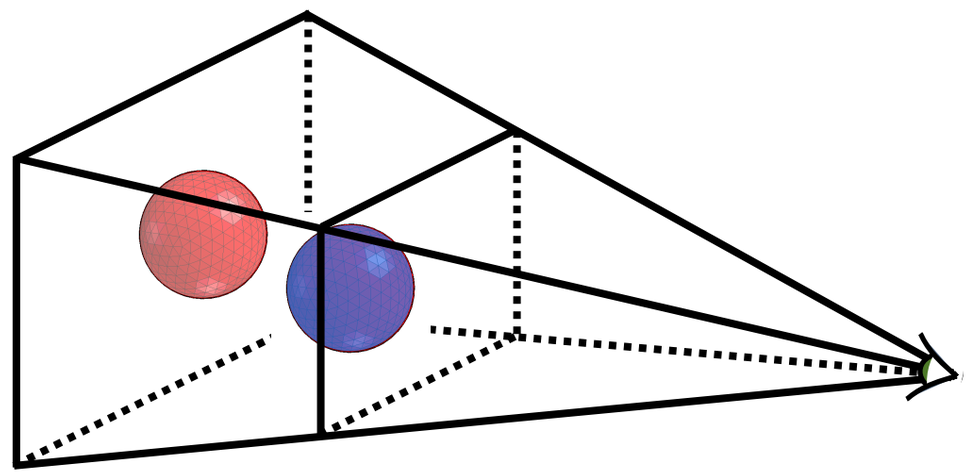
at each pixel, record distance to the closest object that has been drawn in a *depth* buffer

Use a *z-buffer* for hidden surface removal

at each pixel, record distance to the closest object that has been drawn in a *depth* buffer







- assume both spheres of the same size, red drawn last

Use a *z-buffer* for hidden surface removal

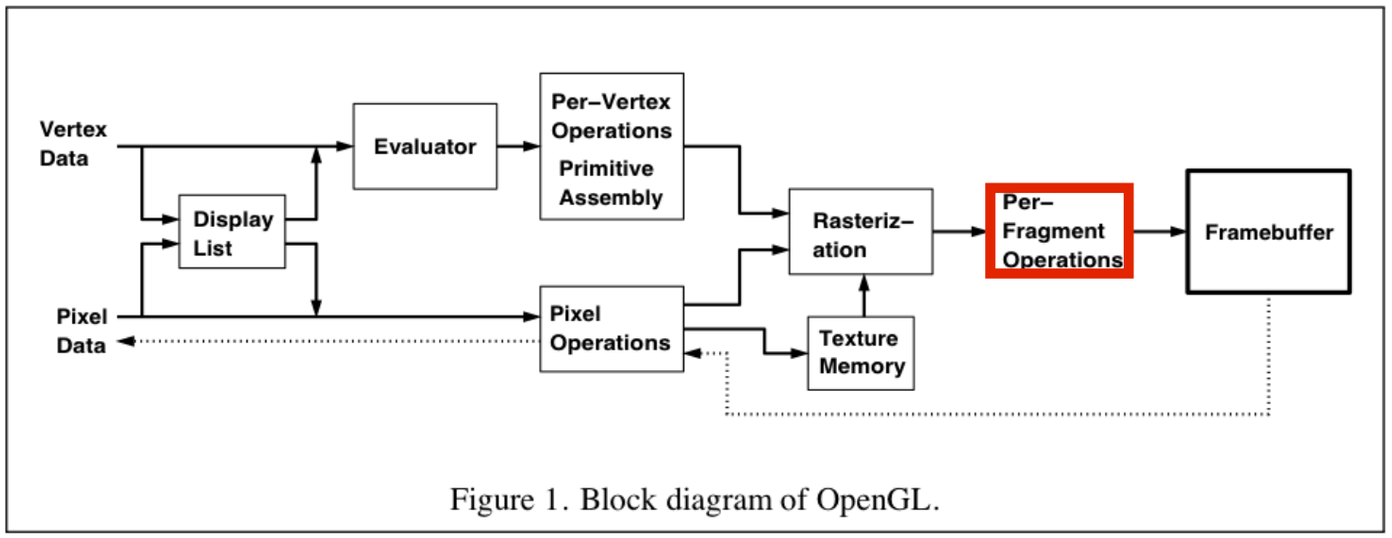
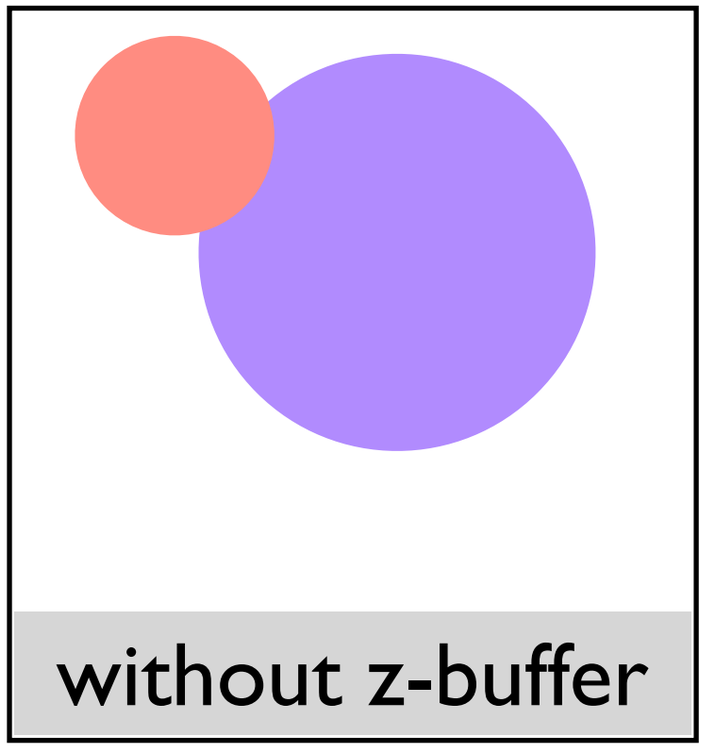
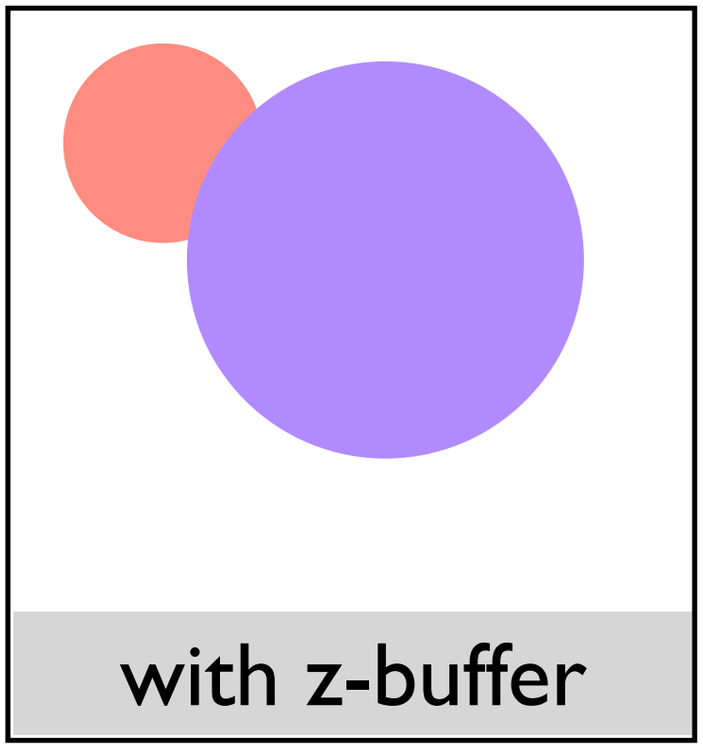
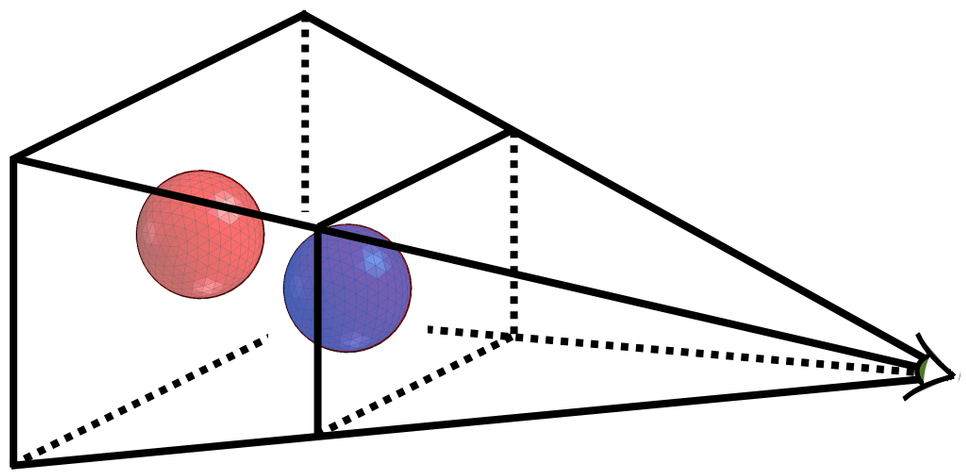


Figure 1. Block diagram of OpenGL.

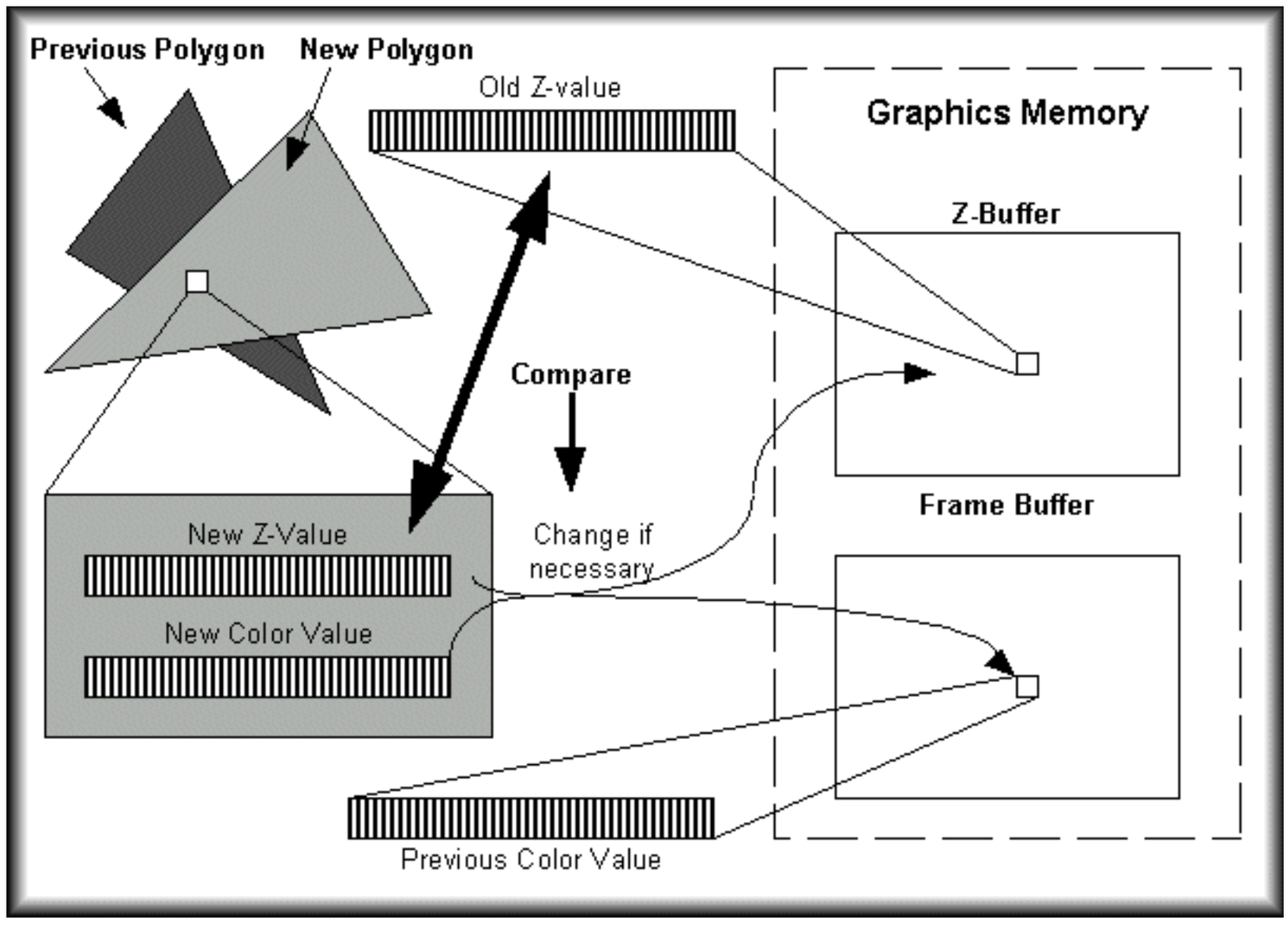






done in the **fragment blending** phase
- each fragment must carry a depth

Use a *z-buffer* for hidden surface removal



http://www.beyond3d.com/content/articles/41/