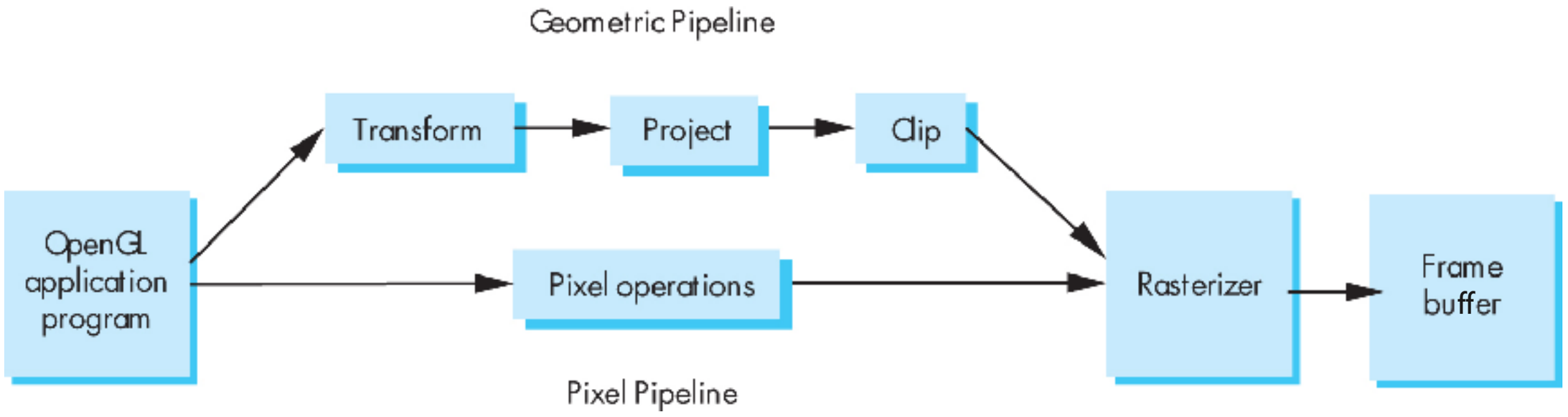
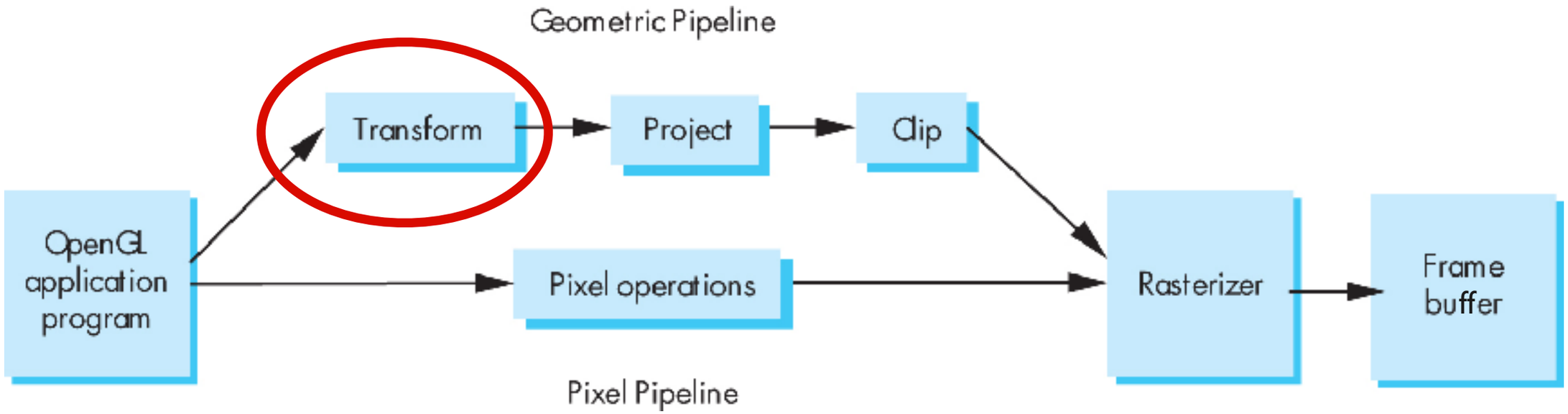
Graphics Pipeline (cont.)

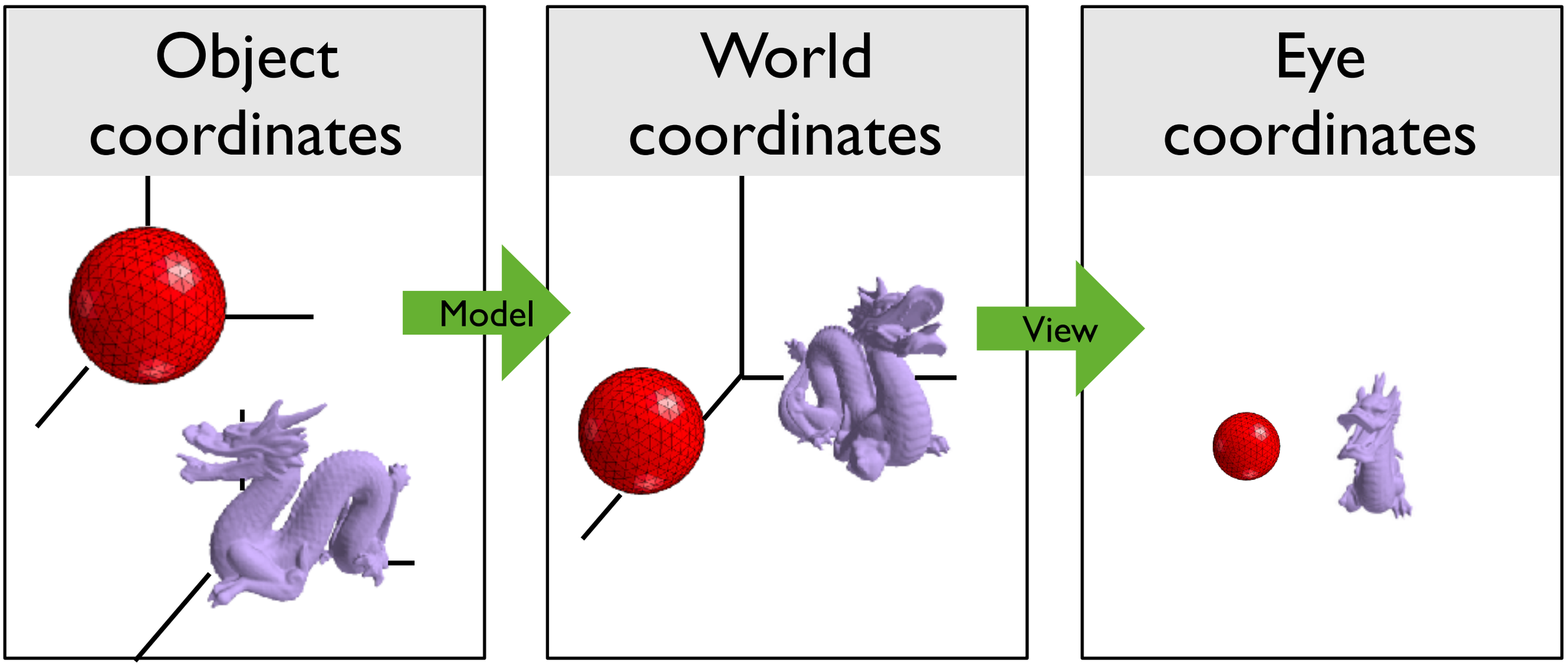
Graphics Pipeline



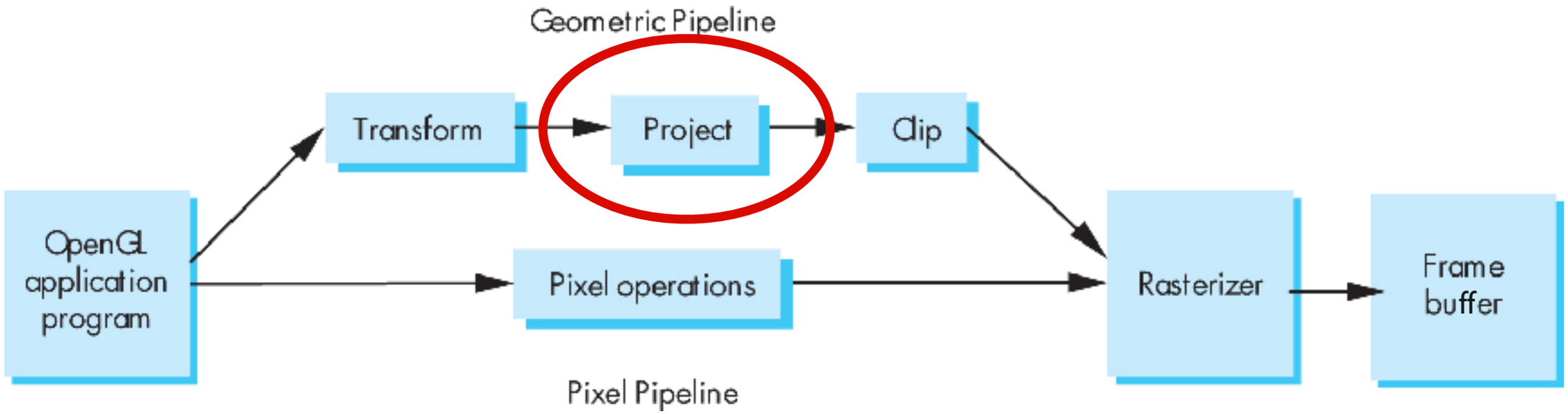
Transform


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

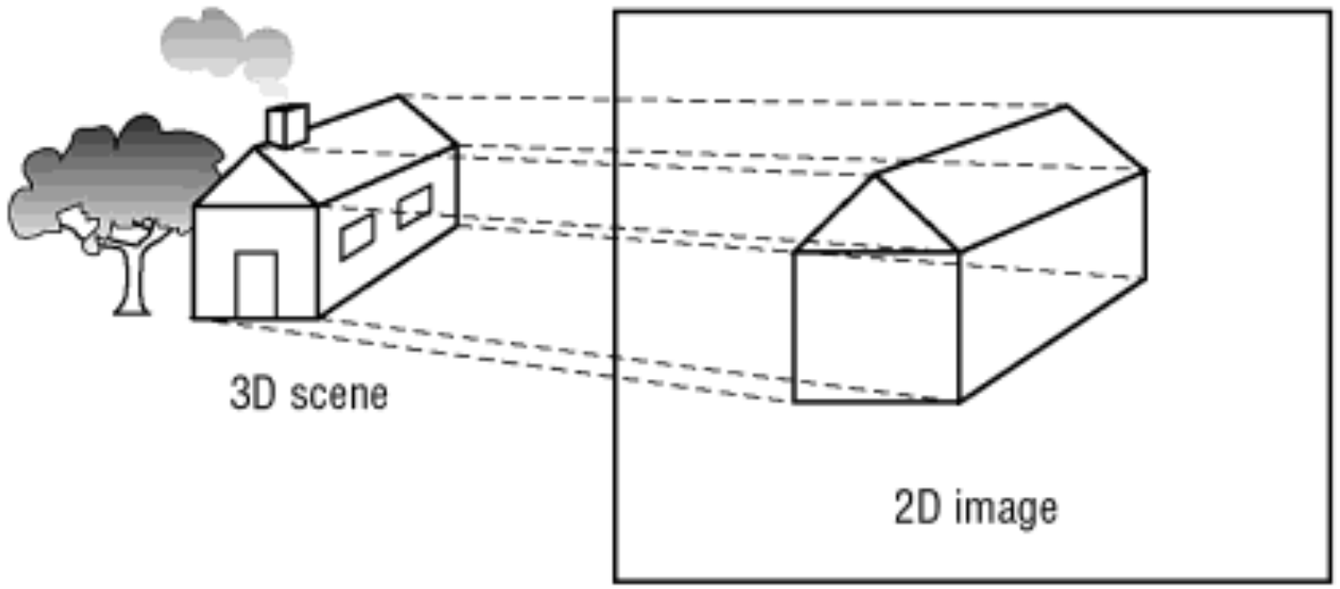
"Modelview" Transformation



Project

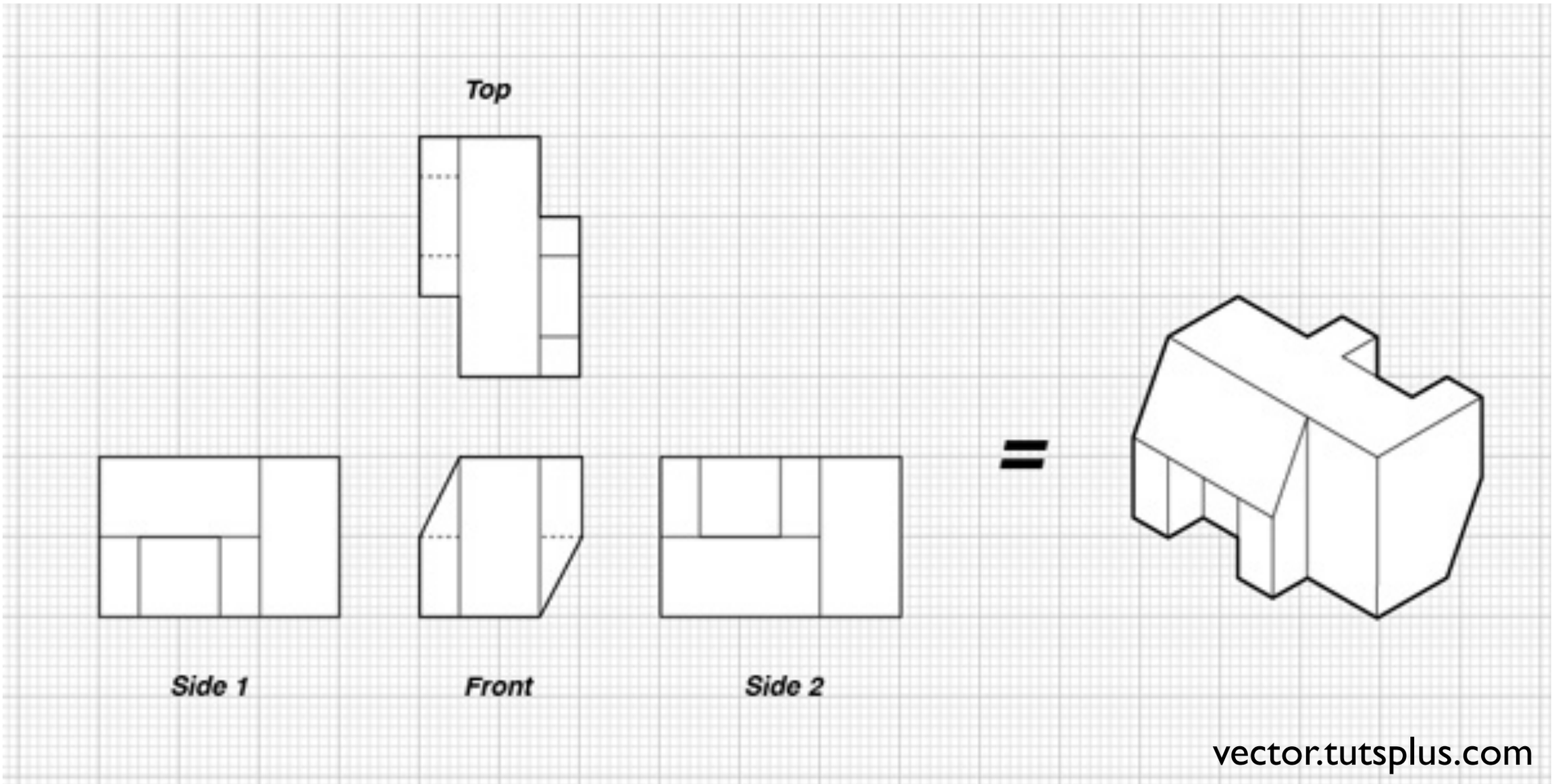

Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

Projection: map 3D scene to 2D image



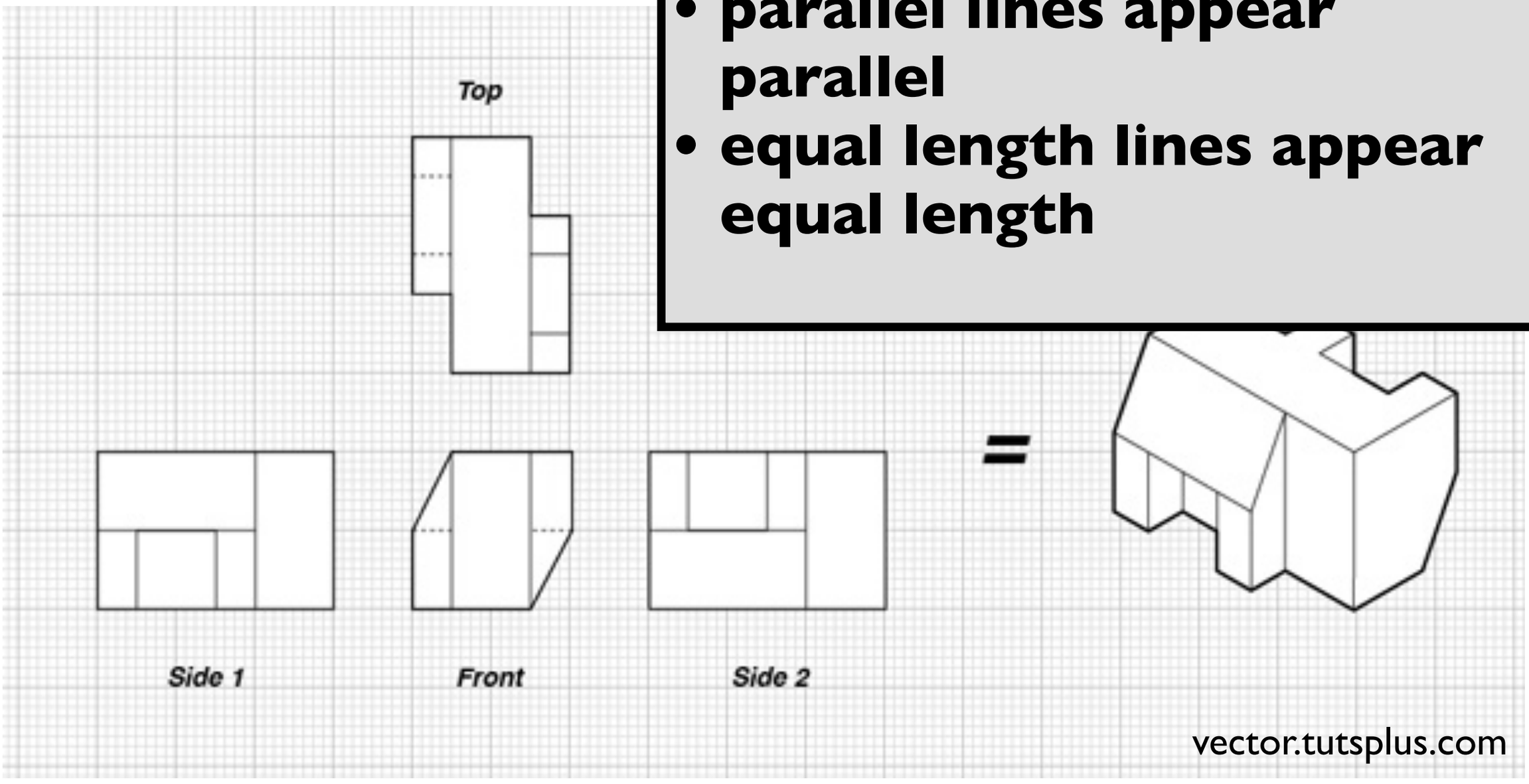
OpenGL Super Bible, 5th Ed.

Orthographic projection



Orthographic projection

- **parallel lines appear parallel**
- **equal length lines appear equal length**

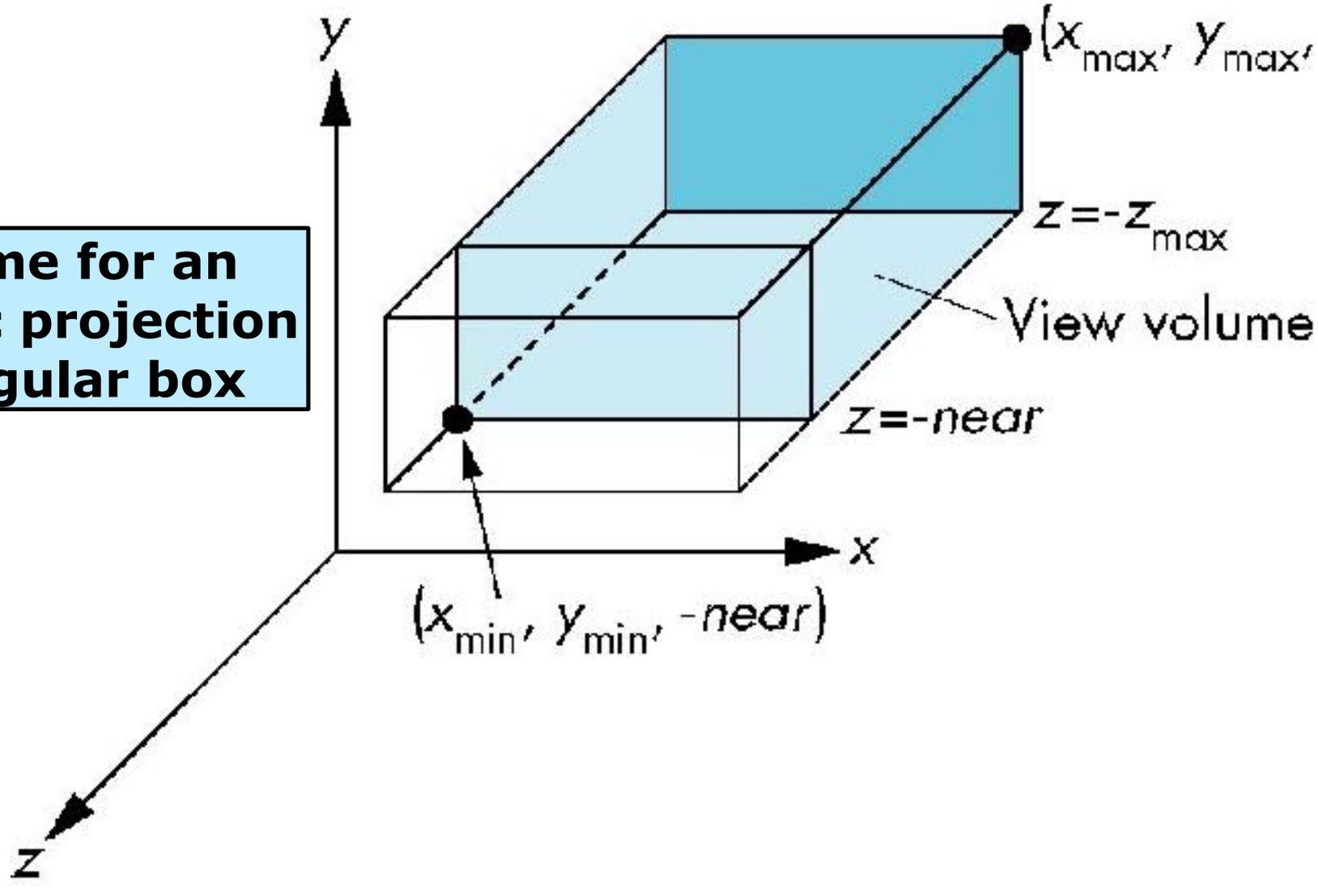


vector.tutsplus.com

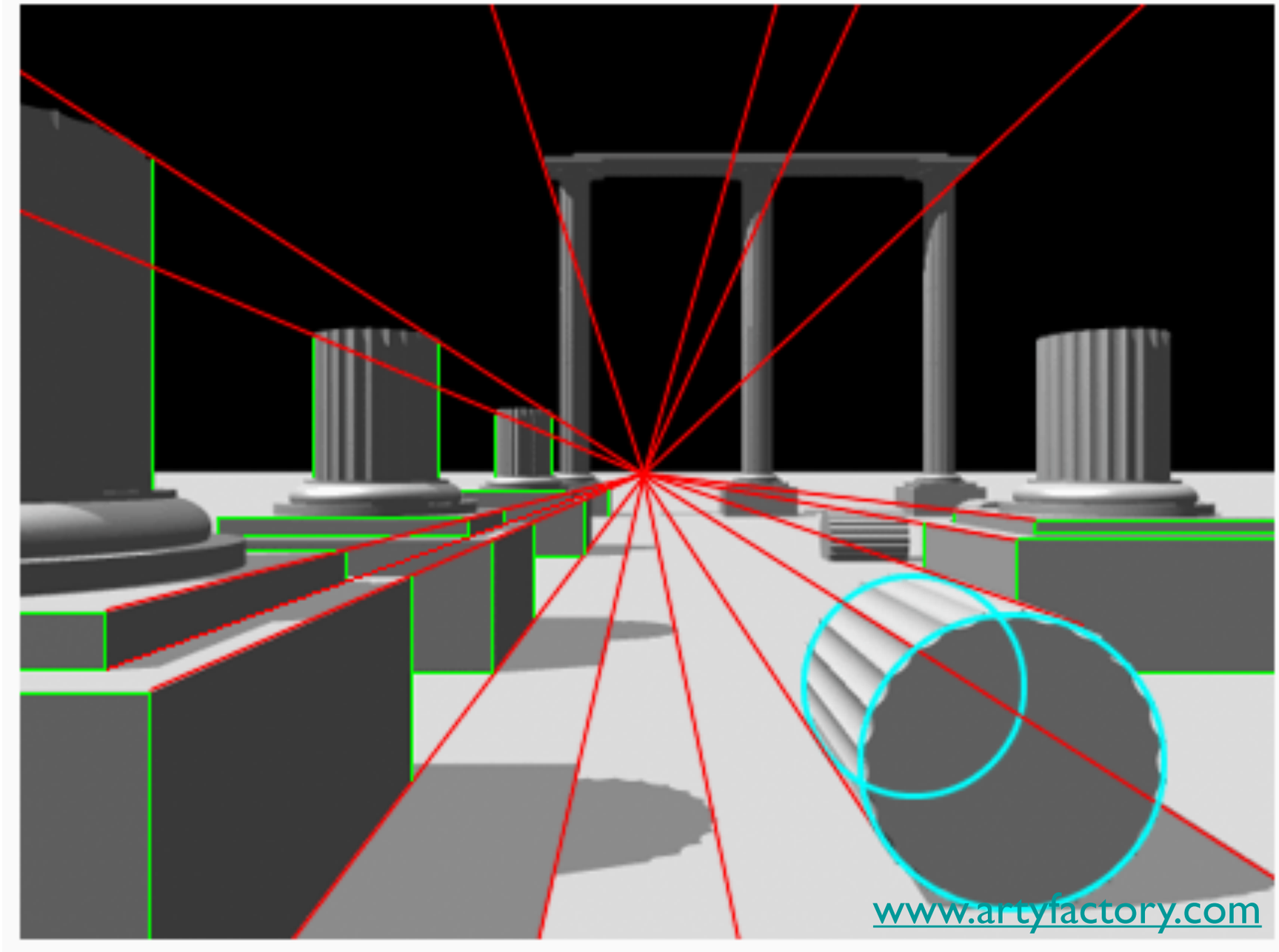
OpenGL Orthogonal Viewing

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

View volume for an orthographic projection is a rectangular box



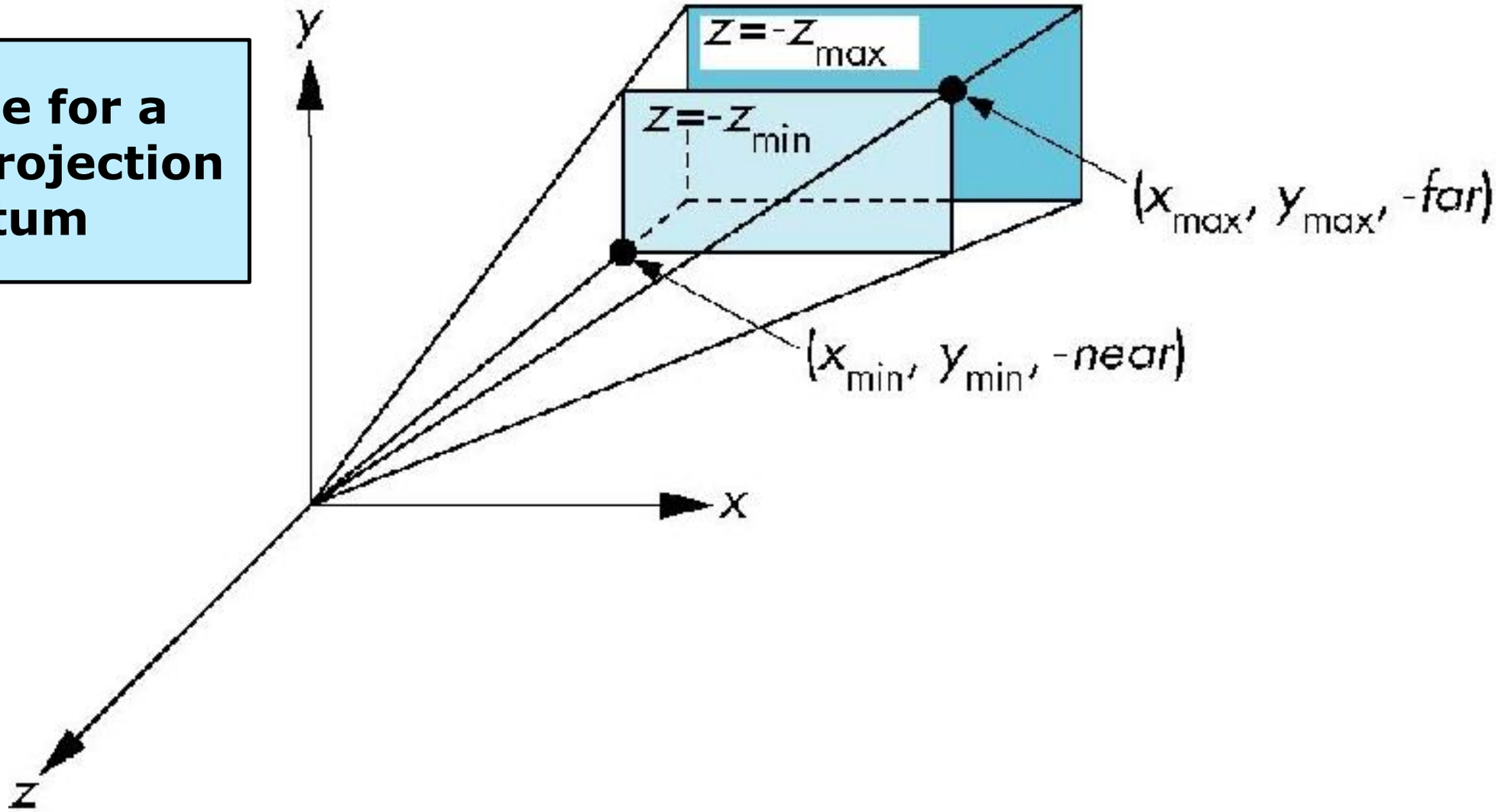
Perspective projection



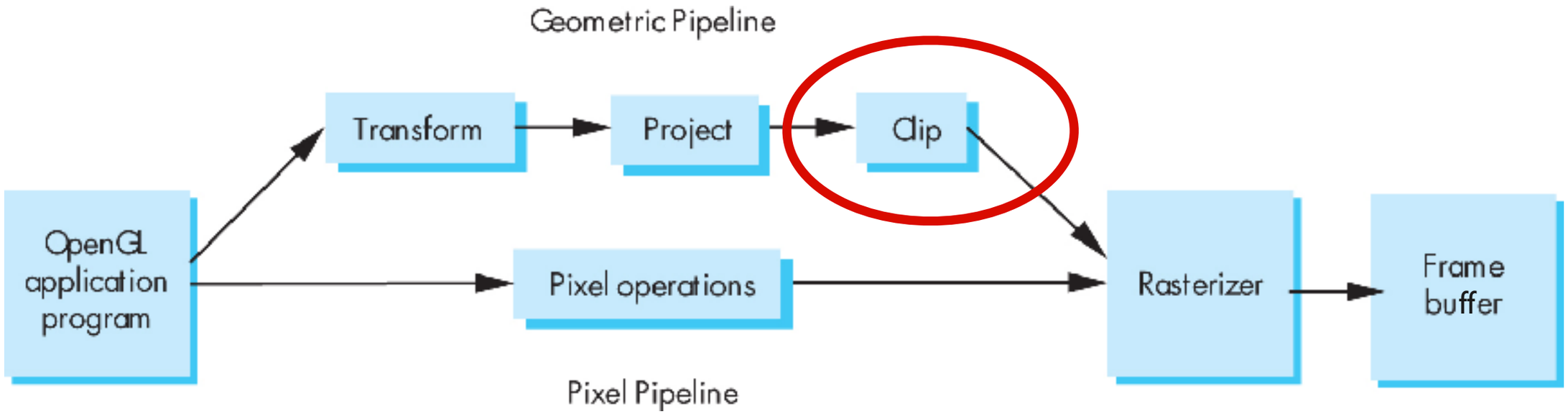
OpenGL Perspective Viewing

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

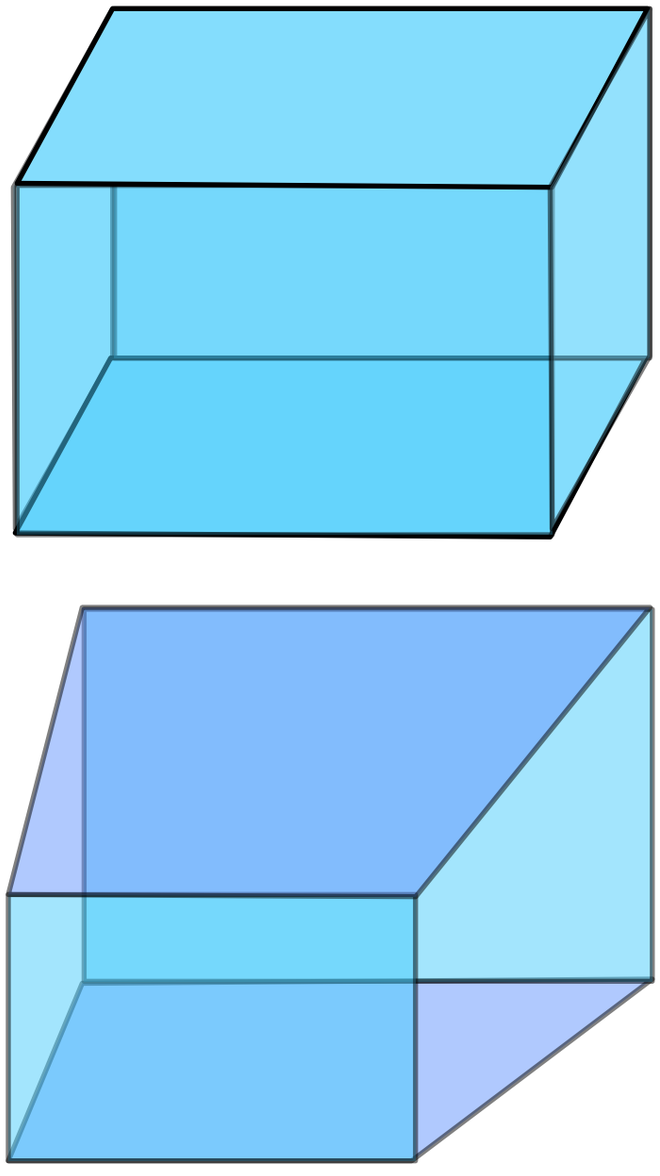
View volume for a perspective projection is a frustum

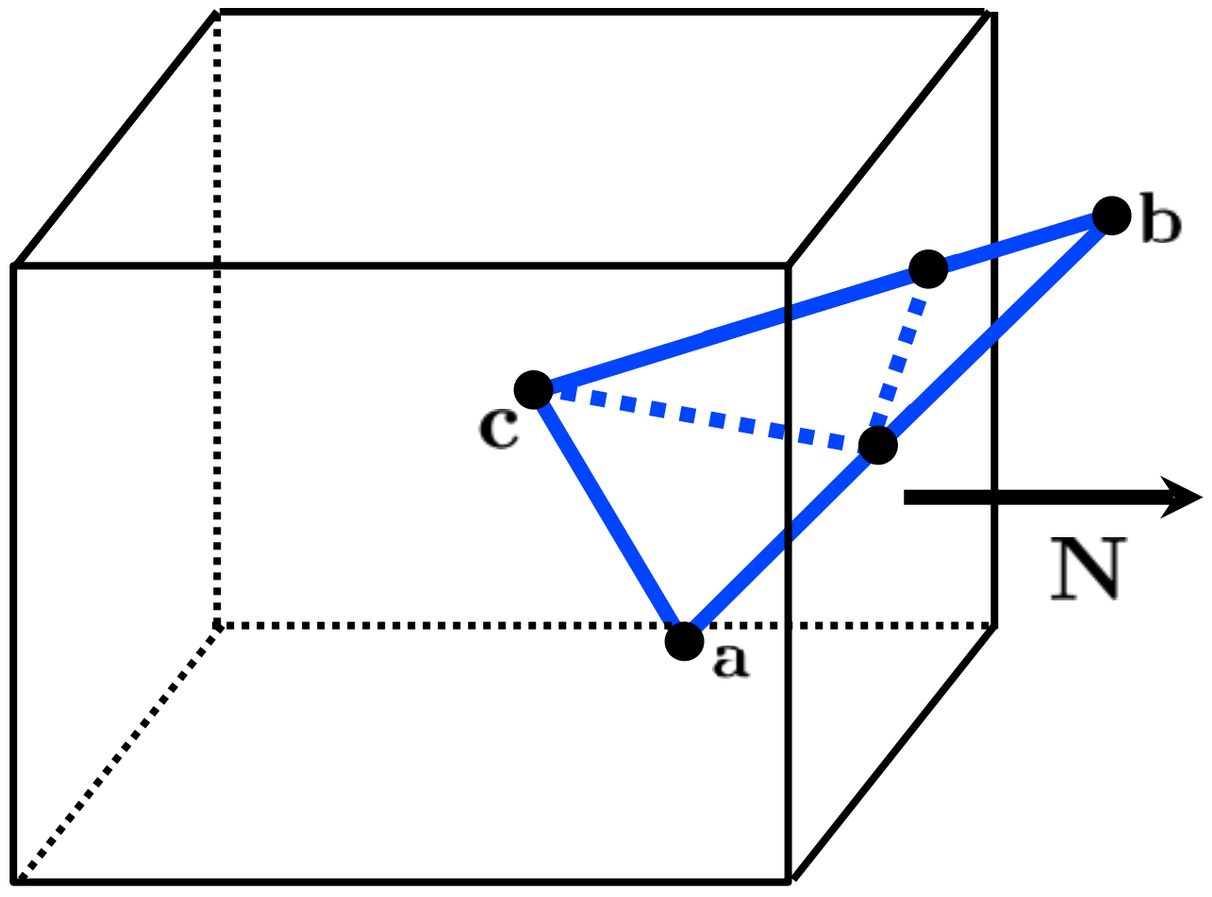


Clip


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

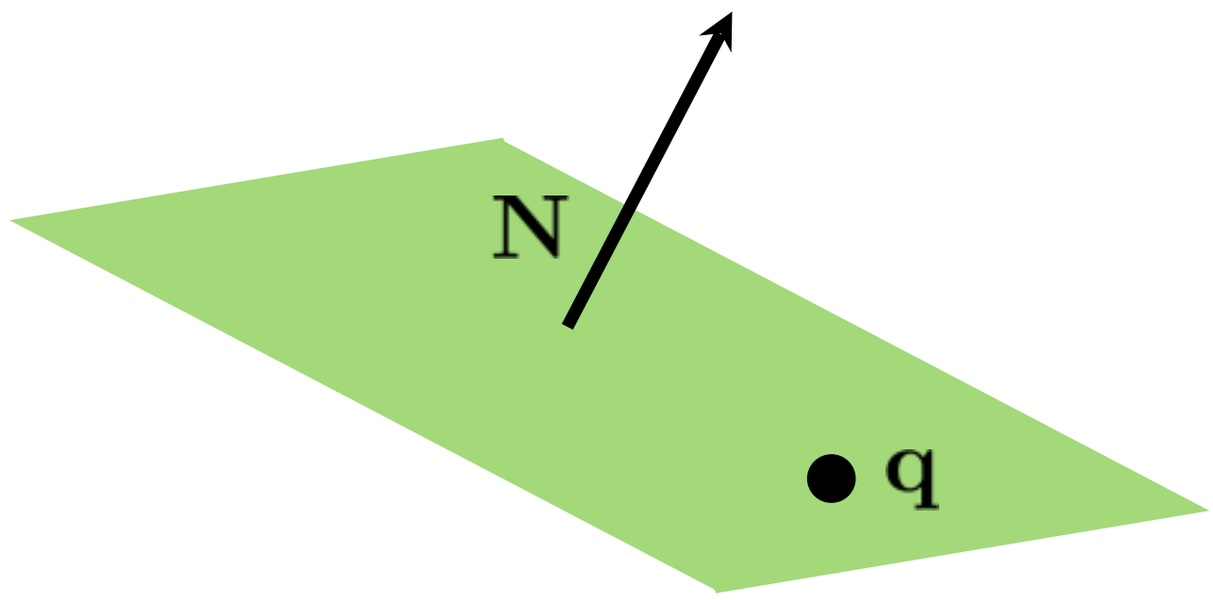
Clip against view volume

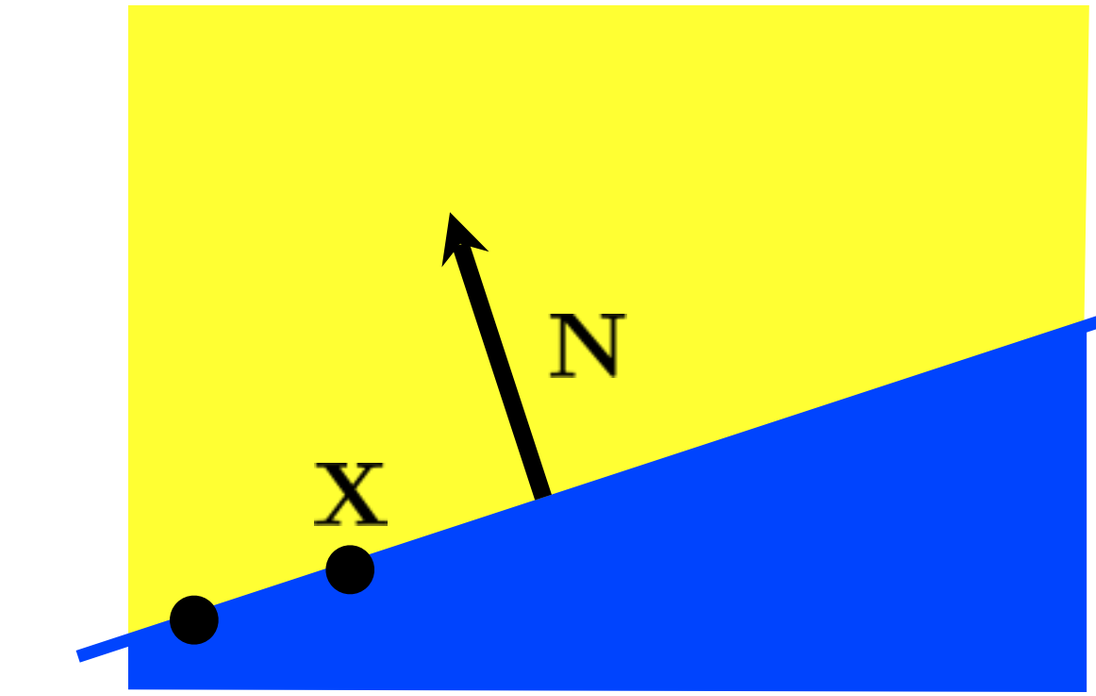




Clipping against a plane

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





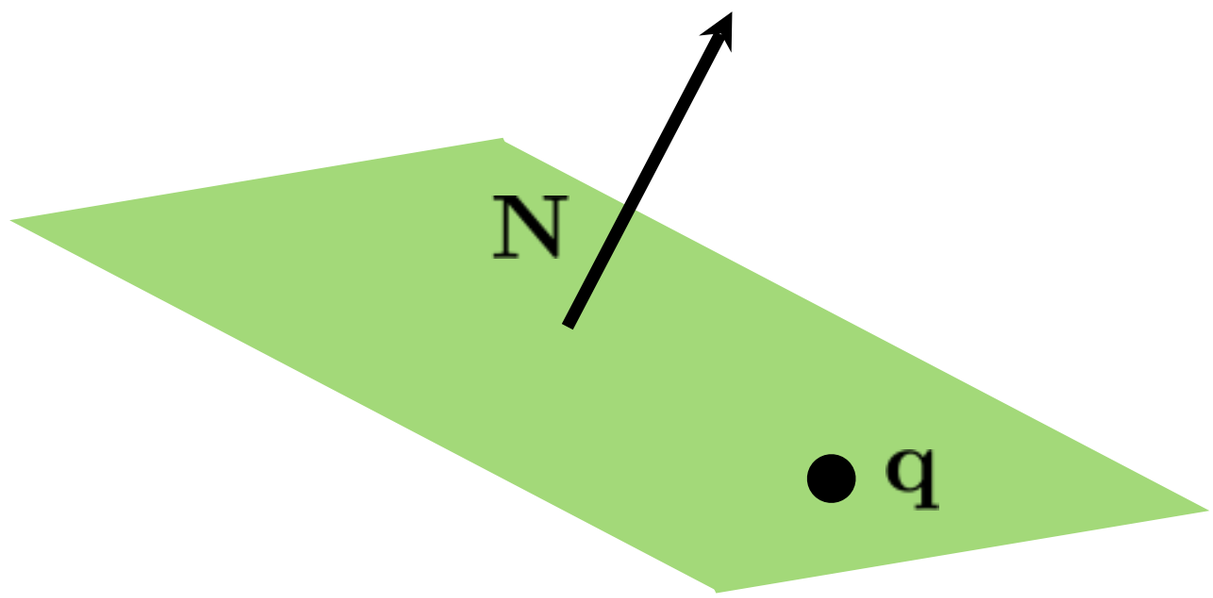
$\mathbf{X}_0 = (x_0, y_0)$

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**?

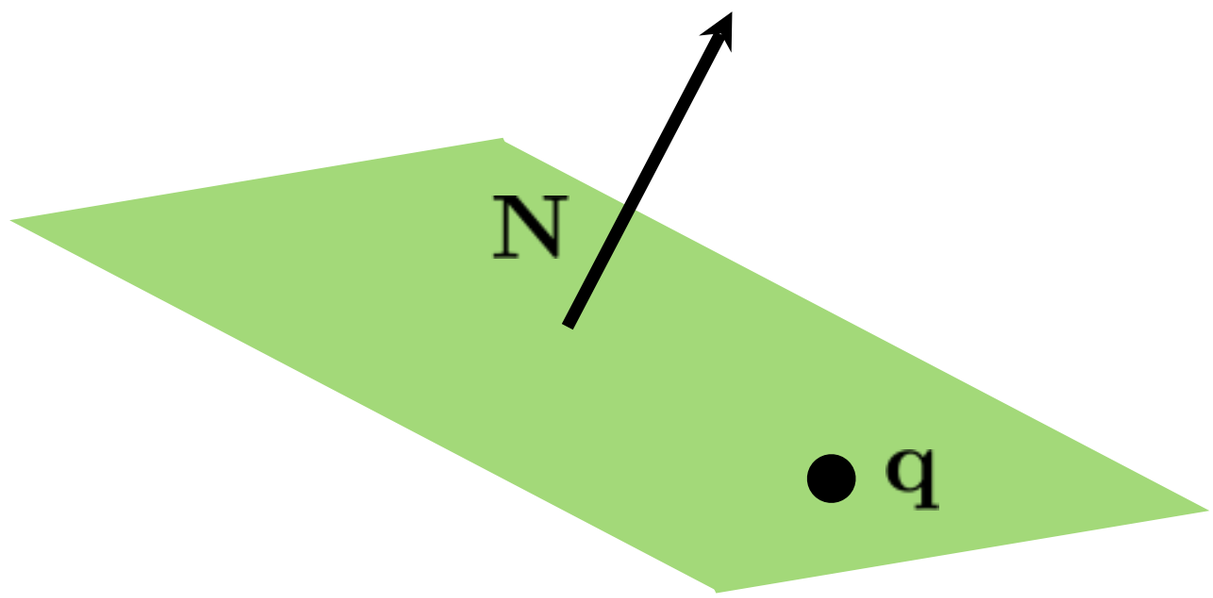


$$f(\mathbf{p}) =? = 0$$

<whiteboard>

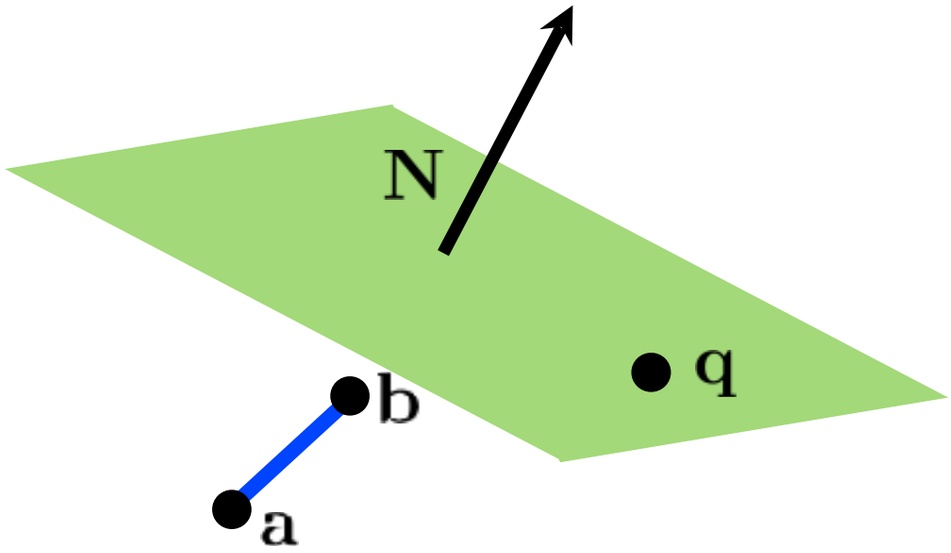
Clipping against a plane

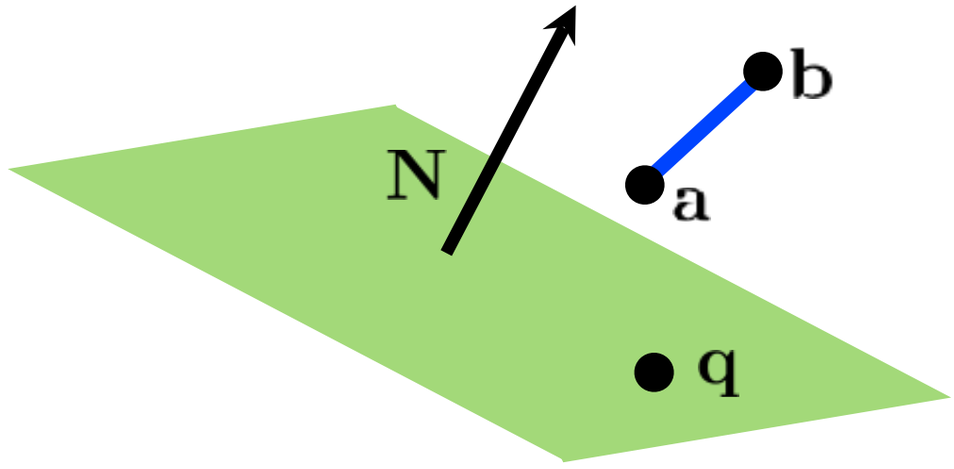
What's the equation for the plane through **q** with normal **N**?



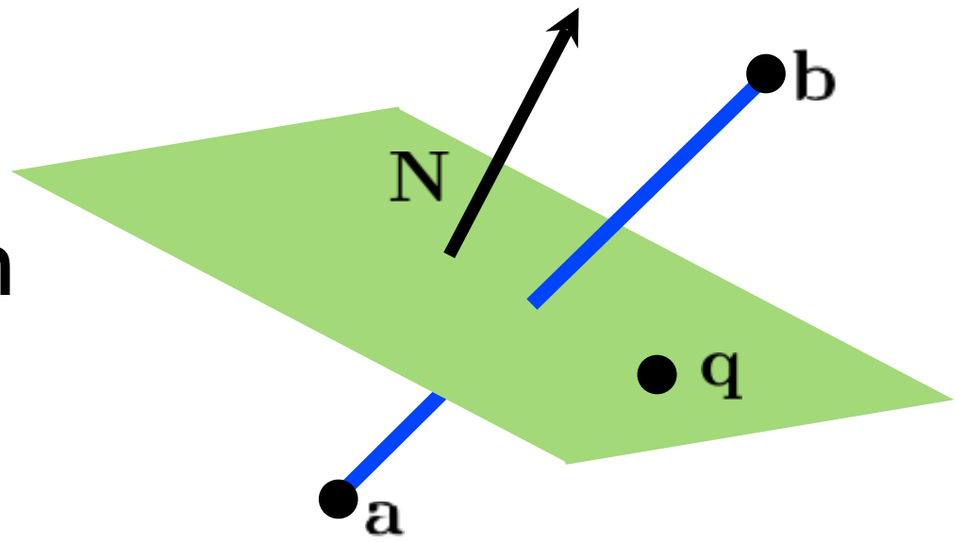
$$f(\mathbf{p}) = \mathbf{N} \cdot (\mathbf{p} - \mathbf{q}) = 0$$

Intersection of line and plane



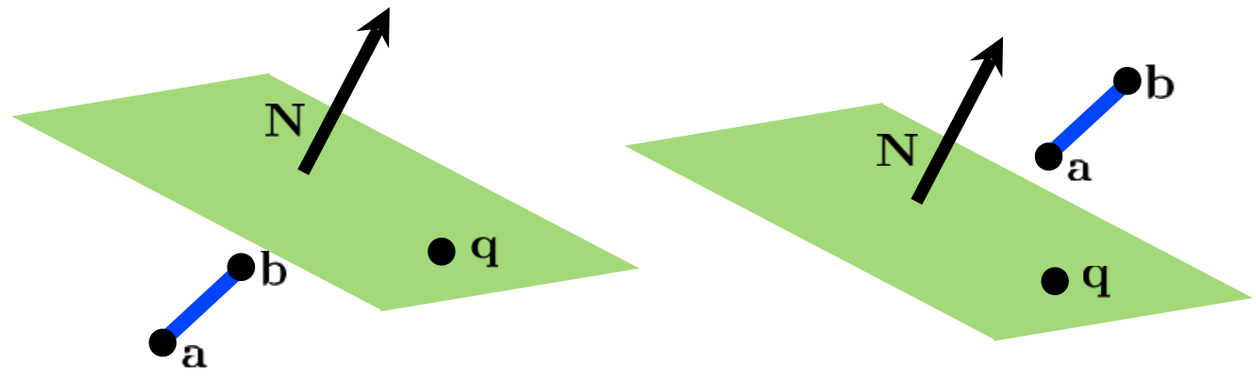


How can we distinguish between these cases?

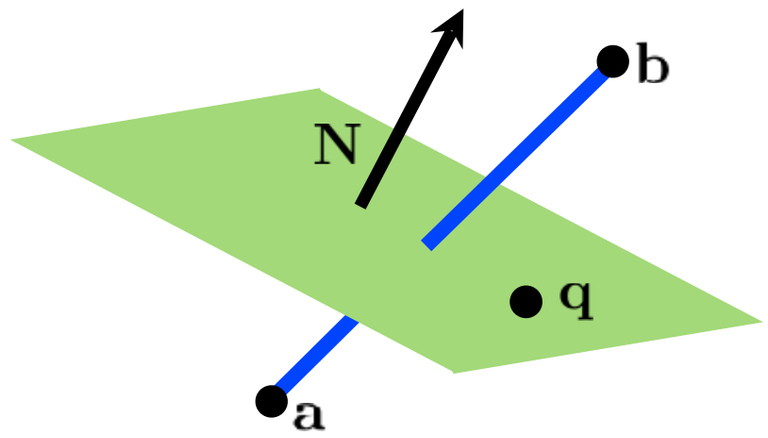


Intersection of line and plane

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

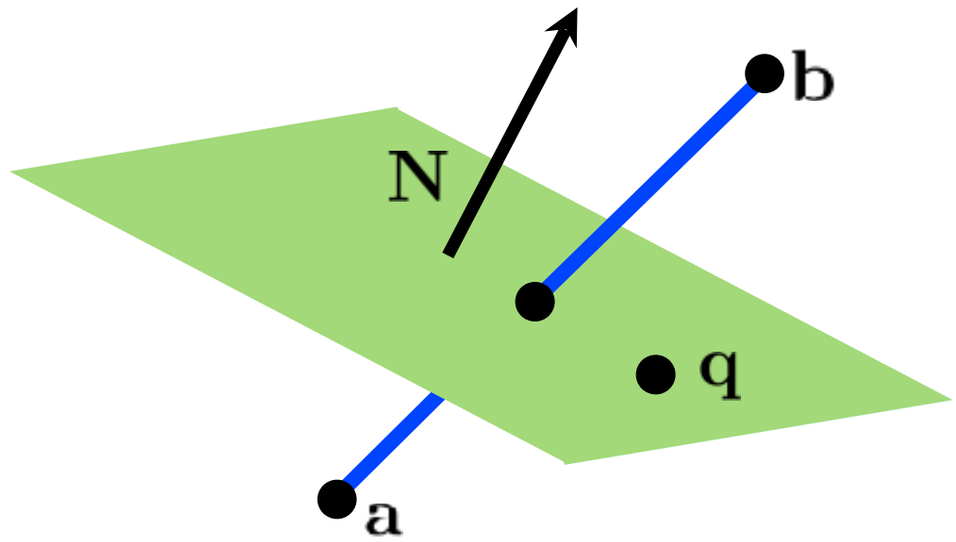


$$f(\mathbf{a})f(\mathbf{b}) < 0$$



Intersection of line and plane

How can we find the intersection point?



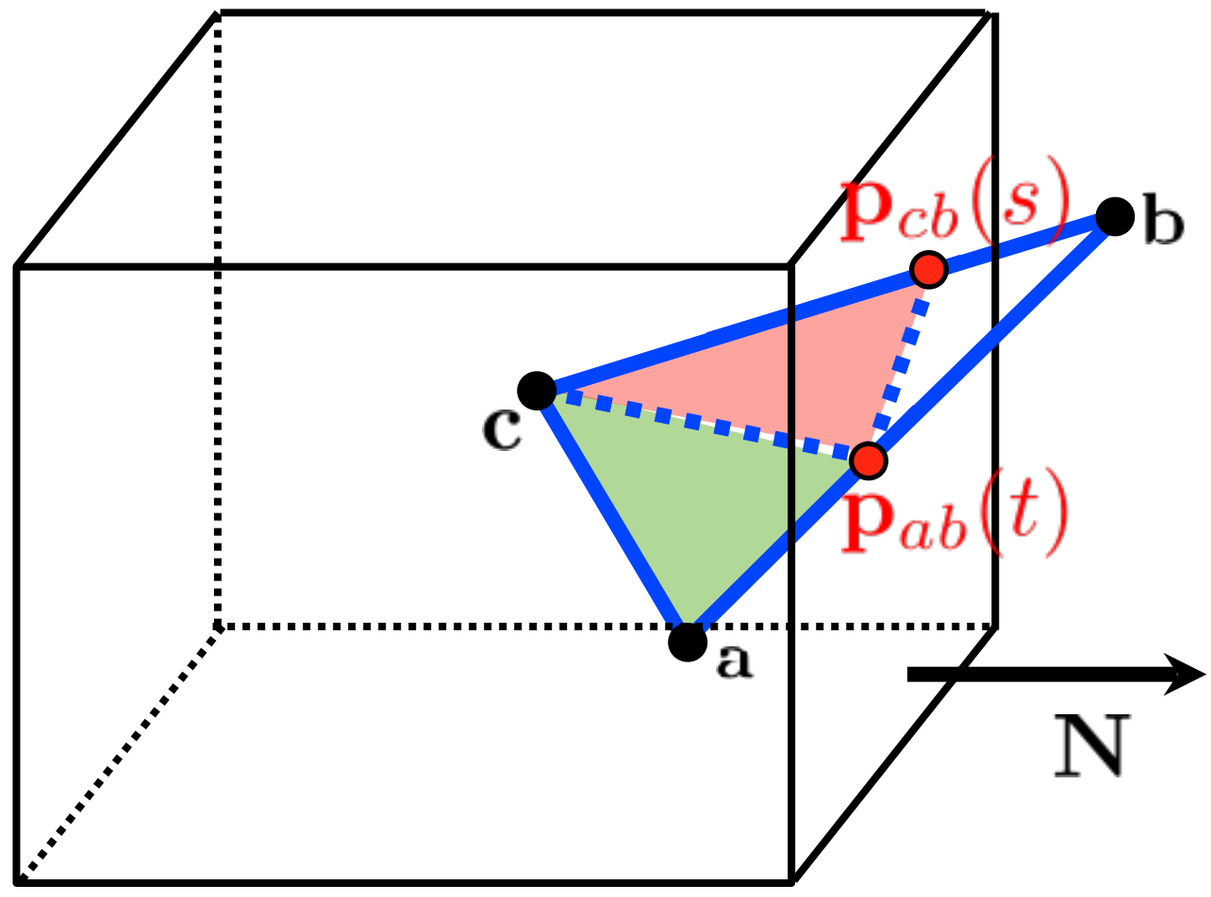
<whiteboard>

Clip against view volume

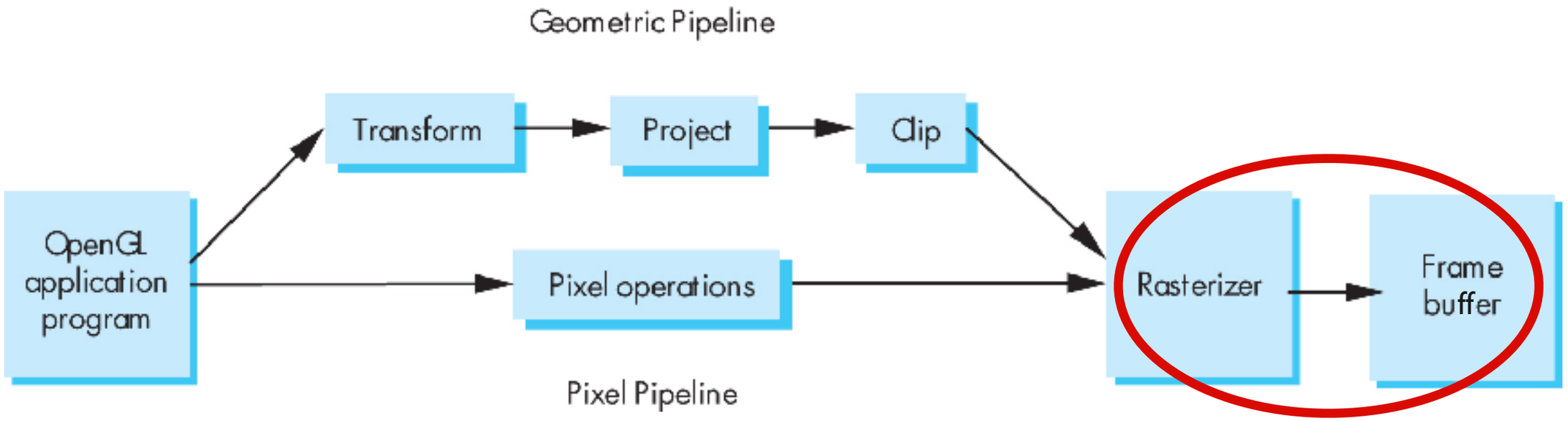
$$s = \frac{\mathbf{N} \cdot (\mathbf{q} - \mathbf{c})}{\mathbf{N} \cdot (\mathbf{b} - \mathbf{c})}$$

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

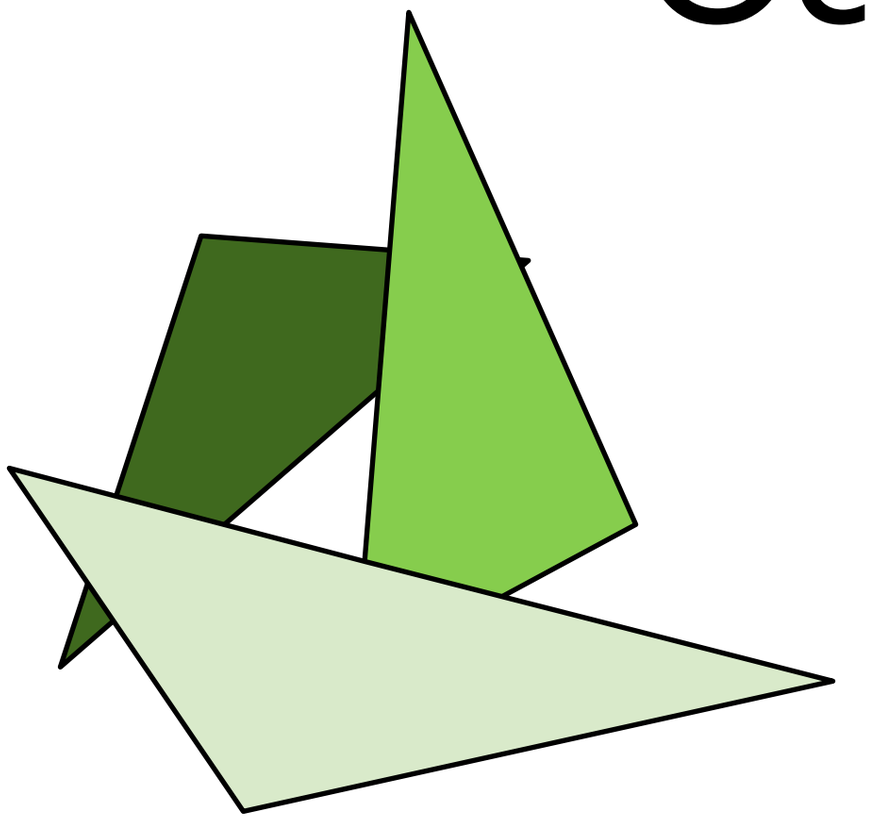
need to generate new triangles



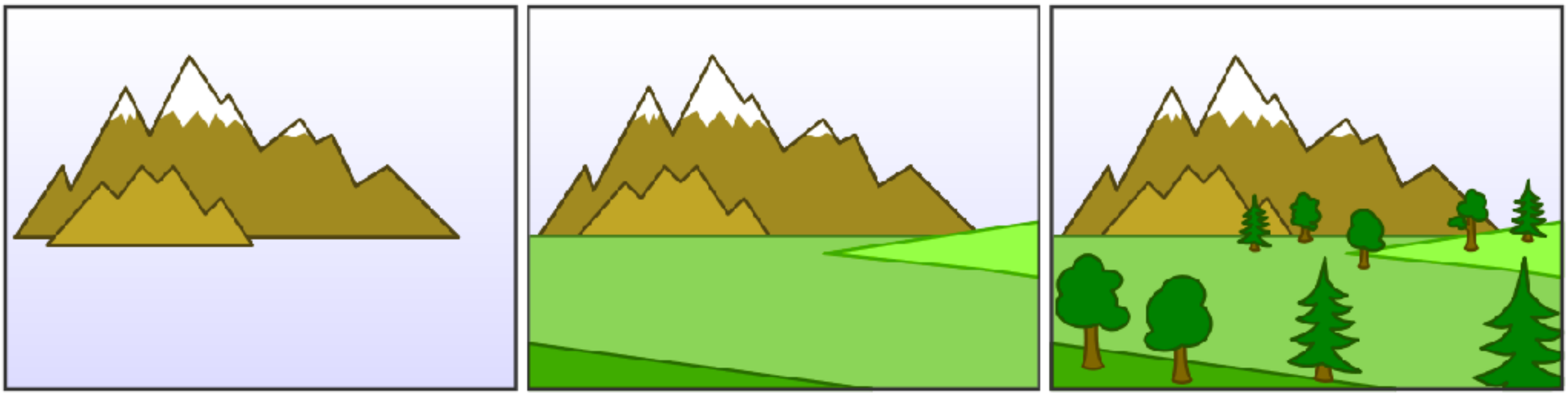
Hidden Surface Removal



Occlusion

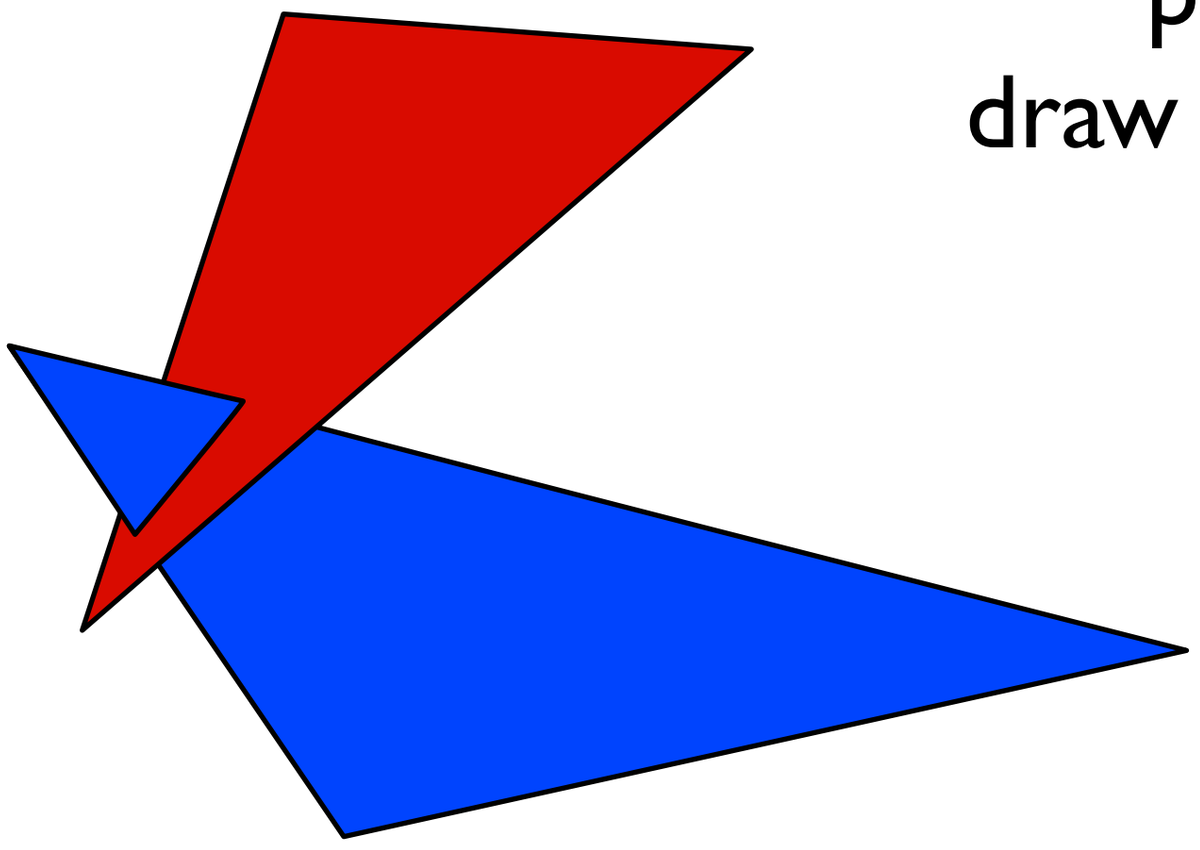


"painter's algorithm"
draw primitives in
back-to-front order



[Wikimedia Commons]

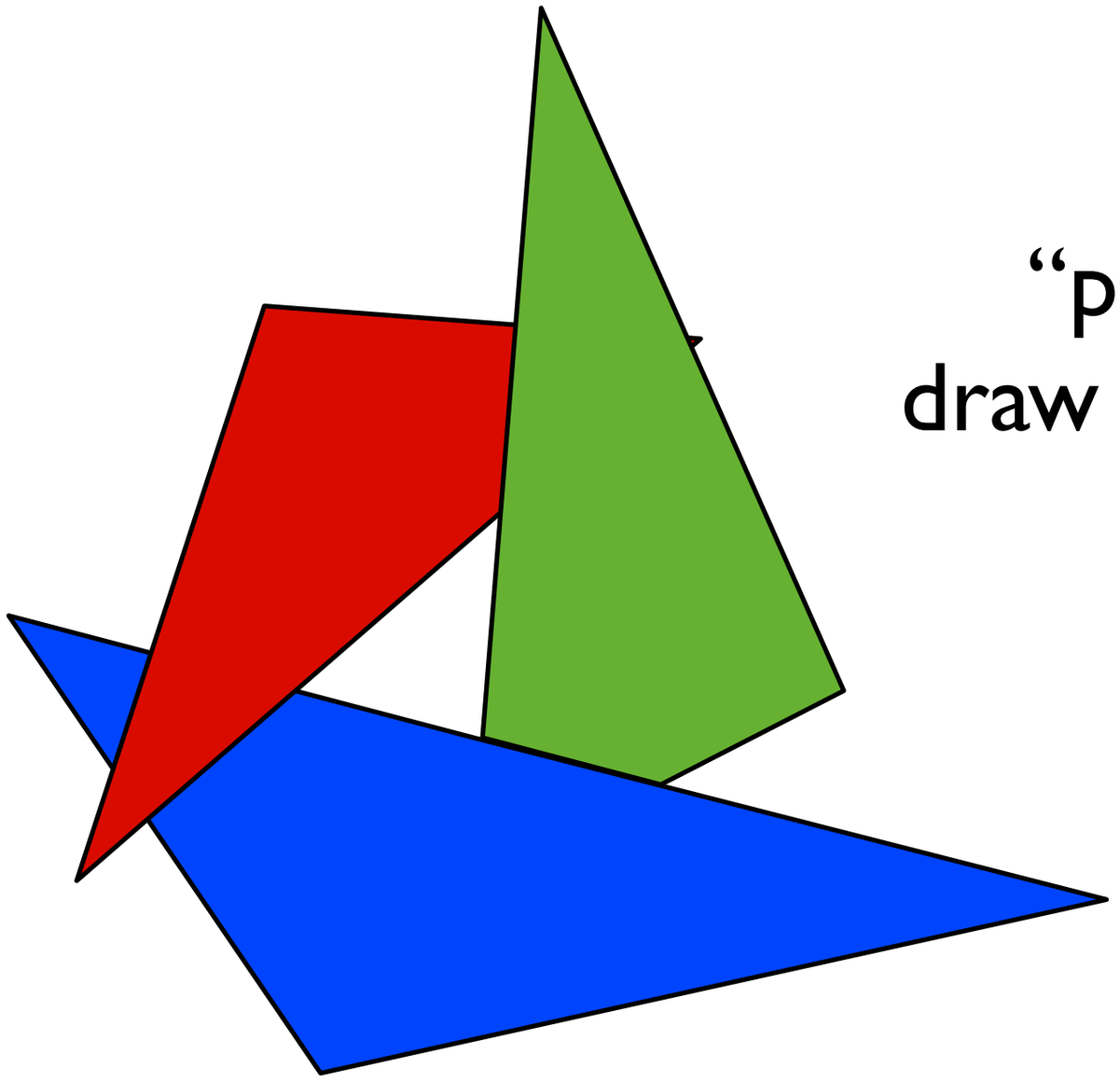
Occlusion



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

problem:
triangle intersection

Occlusion



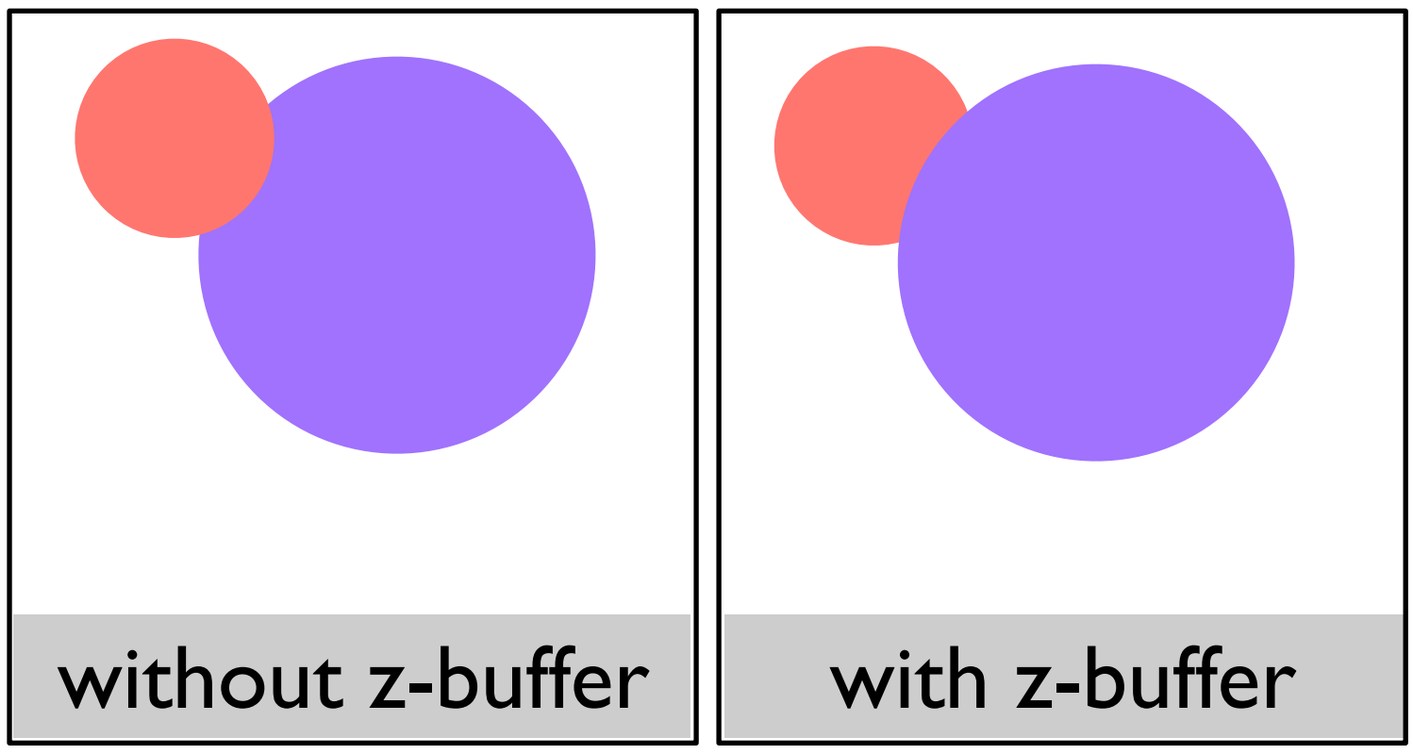
"painter's algorithm"
draw primitives in back-to-front order

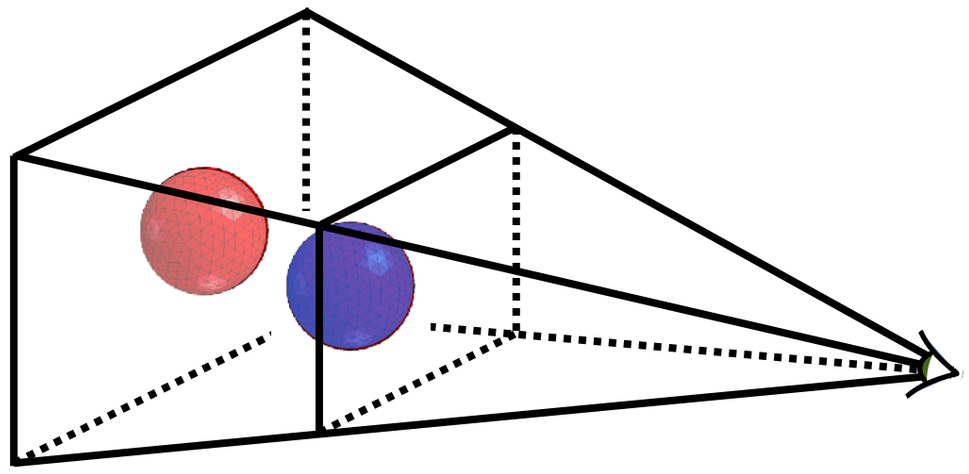
problem:
occlusion cycle

Use a *z-buffer* for hidden surface removal

test depth on a pixel by pixel basis

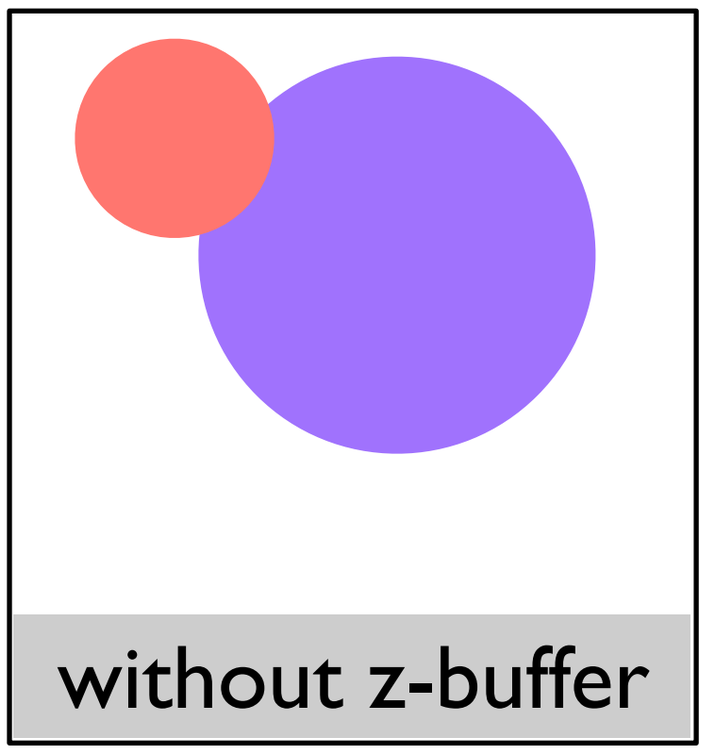
red drawn last

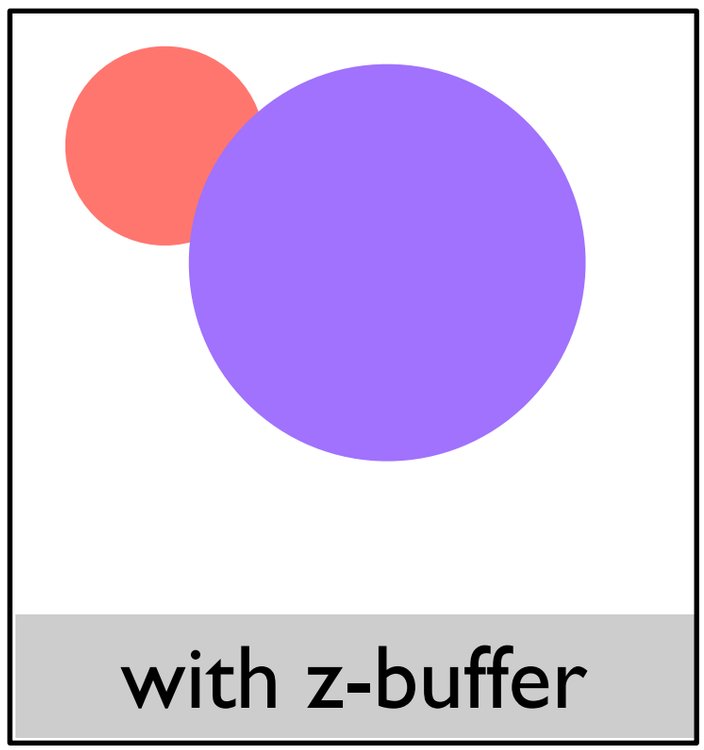


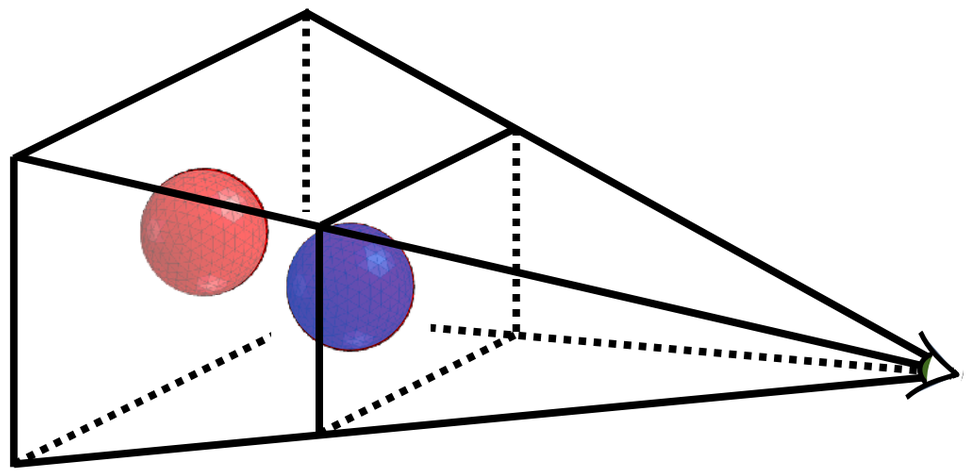


Use a *z-buffer* for hidden surface removal

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

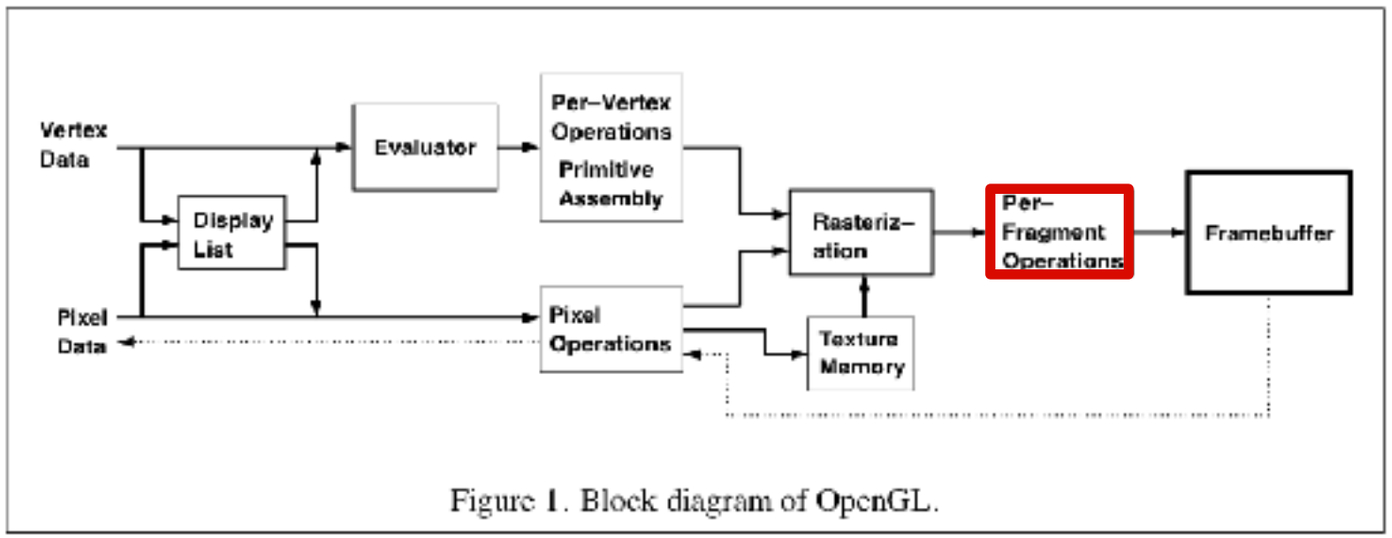
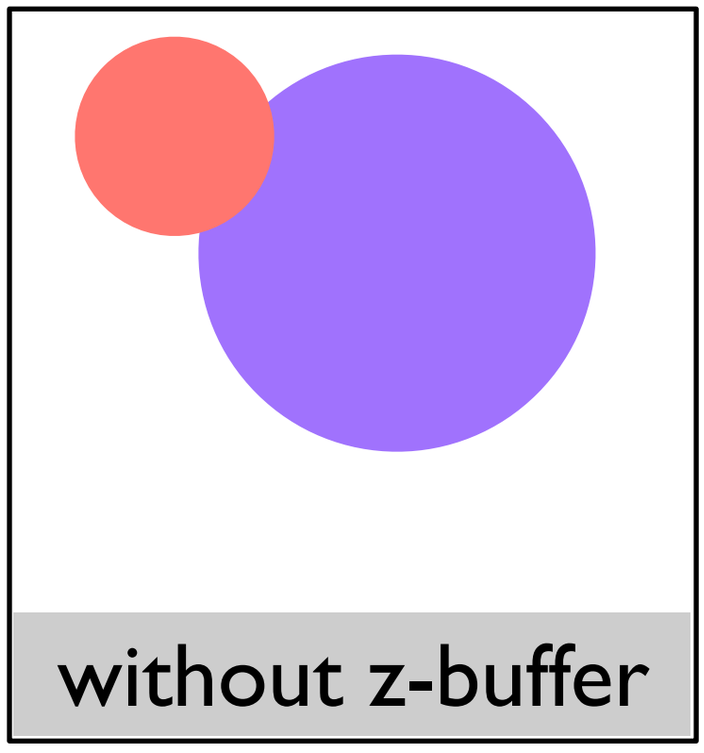
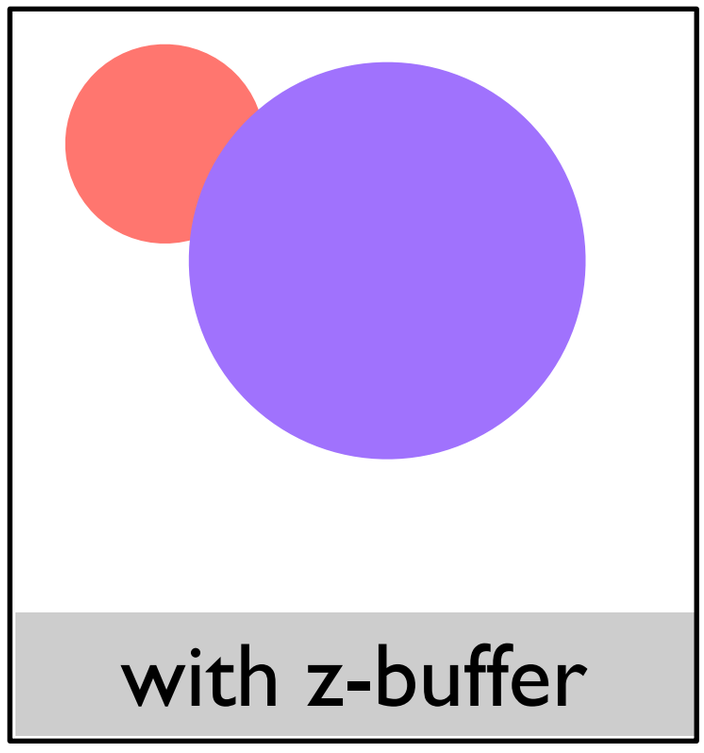
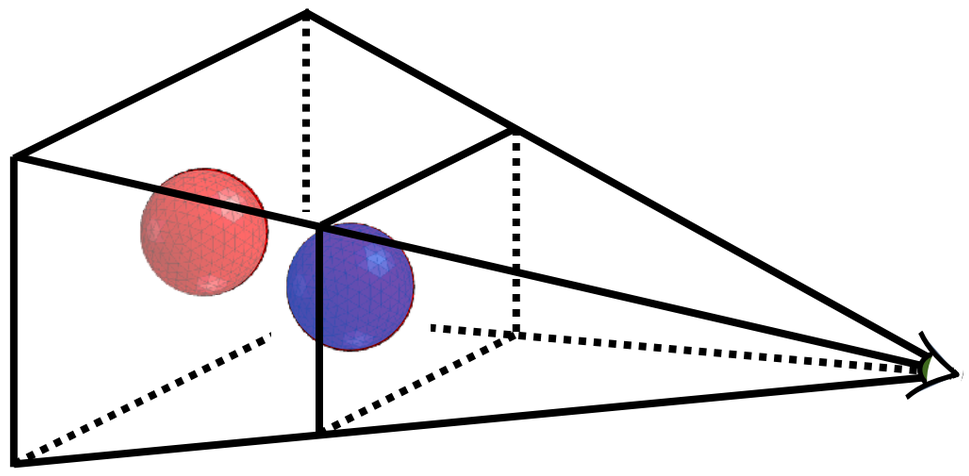


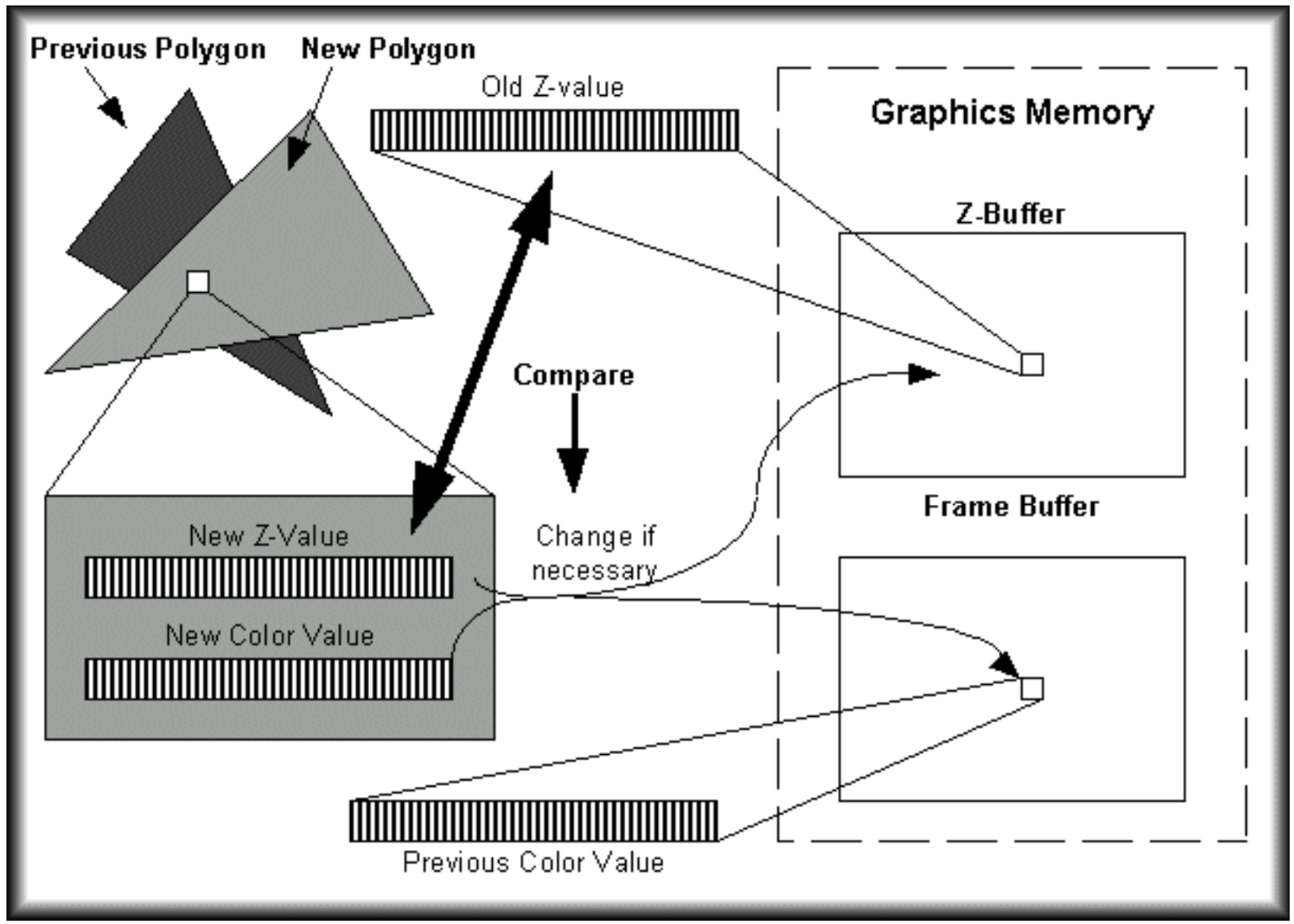
Figure 1. Block diagram of OpenGL.





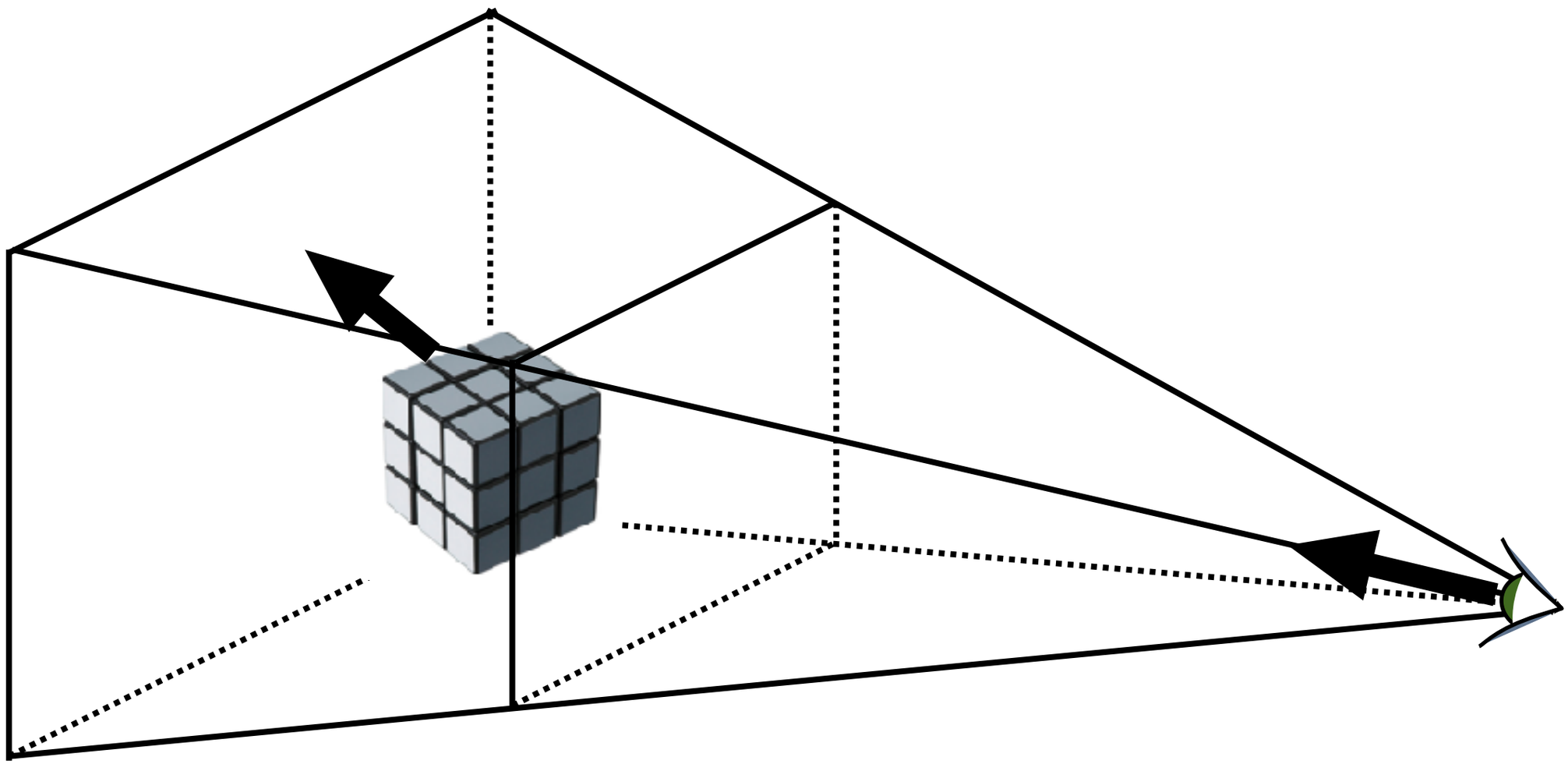


Use a *z-buffer* for hidden surface removal



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

Backface culling: another way to eliminate hidden geometry



Hidden Surface Removal in OpenGL

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

glEnable(GL_DEPTH_TEST);

glEnable(GL_CULL_FACE);
```

For a perspective transformation, there is more precision in the depth buffer for z-values closer to the near plane