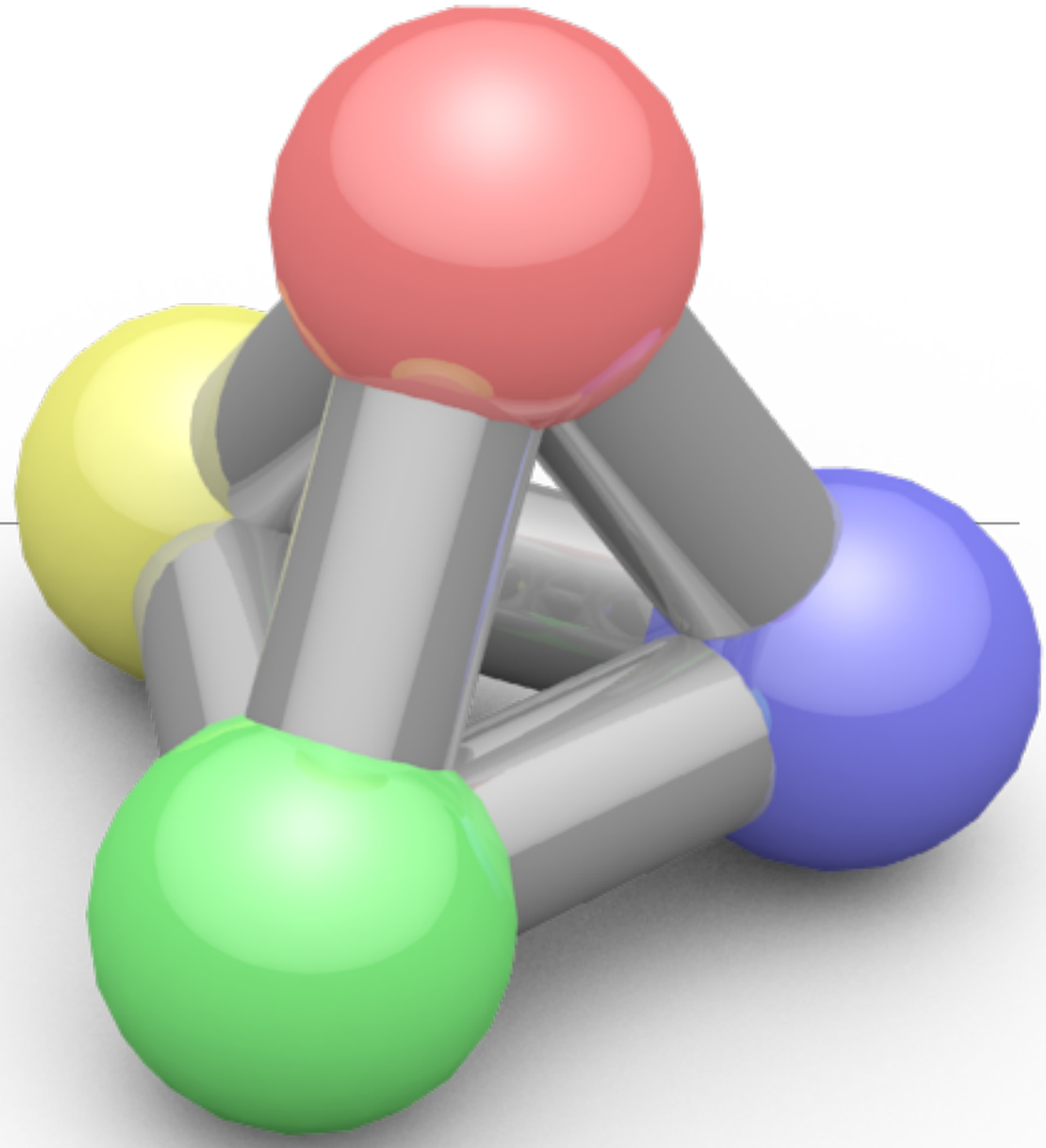
Hidden Surface Removal



CPSC 453 – Fall 2016
Sonny Chan

Perspective Divide

Objects at twice the distance appear at half the height:

$$x_p = \frac{x}{-z}$$

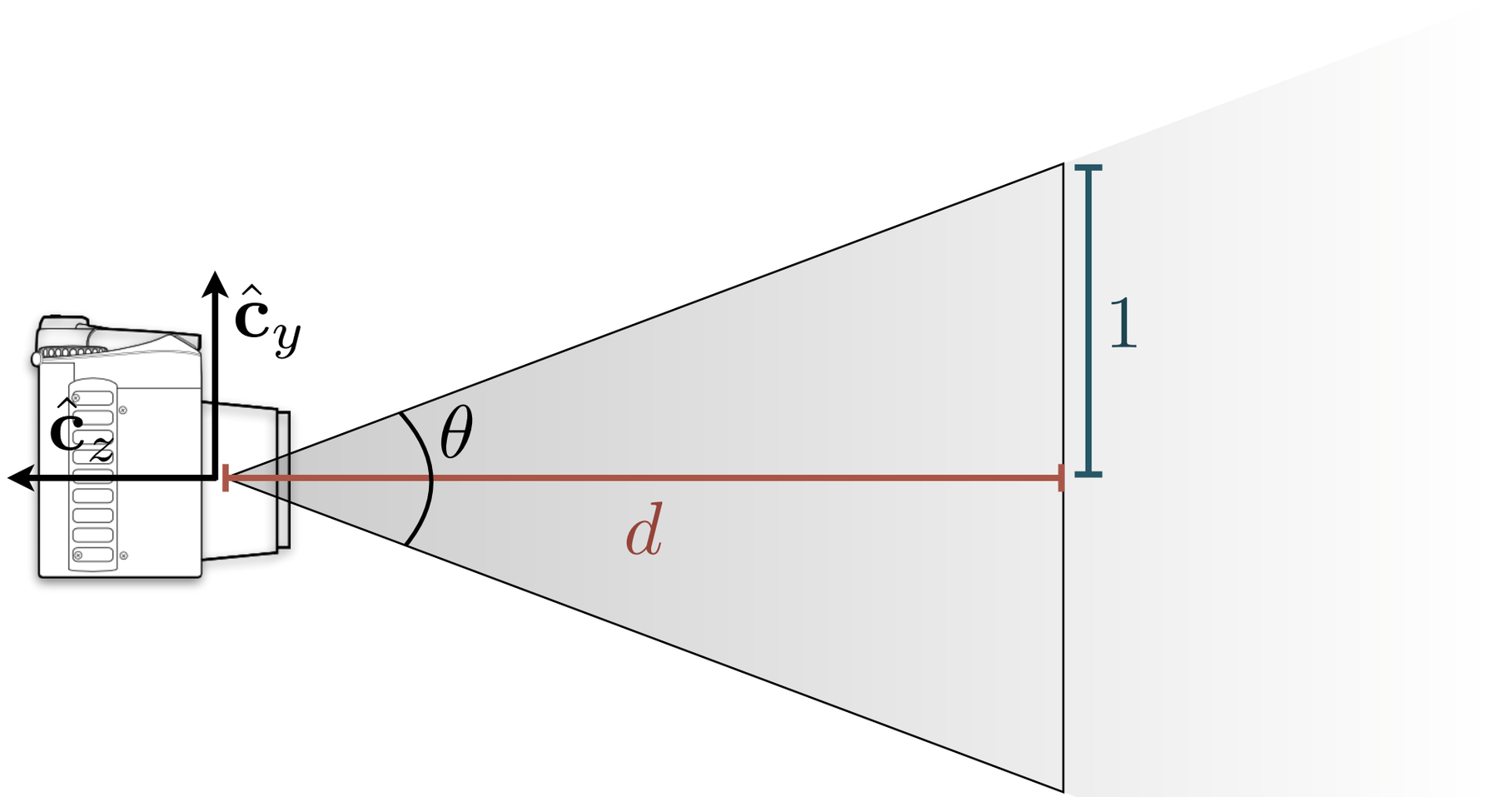
$$y_p = \frac{y}{-z}$$

[photo by Ricky Leong]

Focal Distance: Field of View

$$x_p = \frac{dx}{-z}$$

$$y_p = \frac{dy}{-z}$$



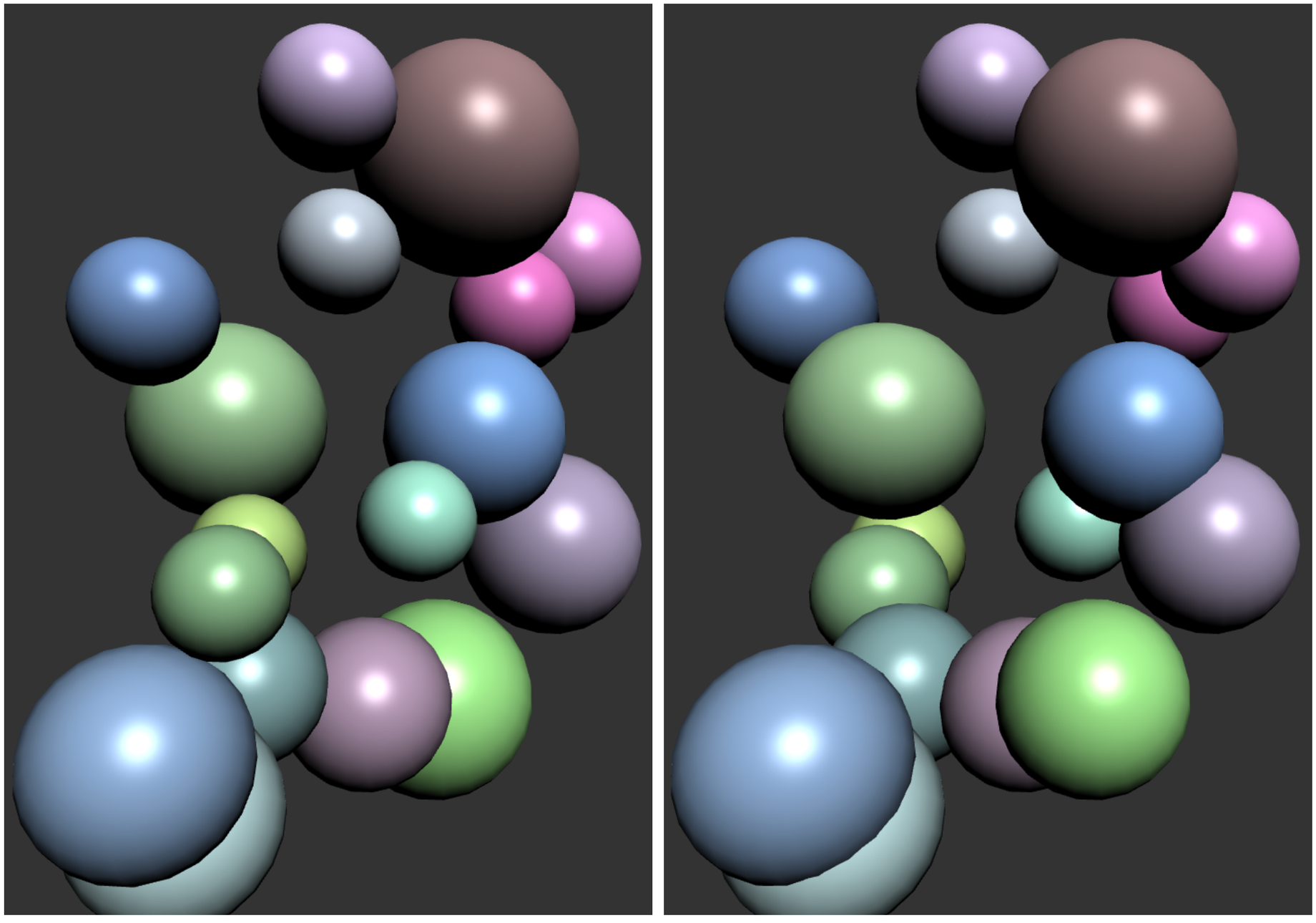
$$\tan\frac{\theta}{2} = \frac{1}{d}$$

Perspective Projection

$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ 0 \\ -z \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{dx}{-z} \\ \frac{dy}{-z} \\ 0 \\ 0 \end{bmatrix}$$

Can you think of some limitations or problems?

Occlusion Ordering

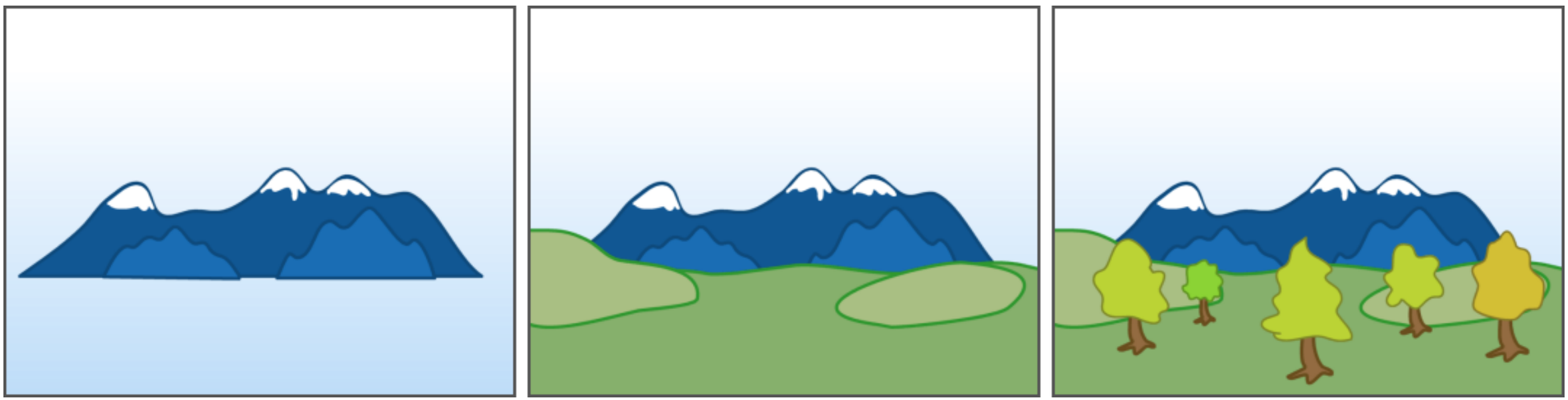


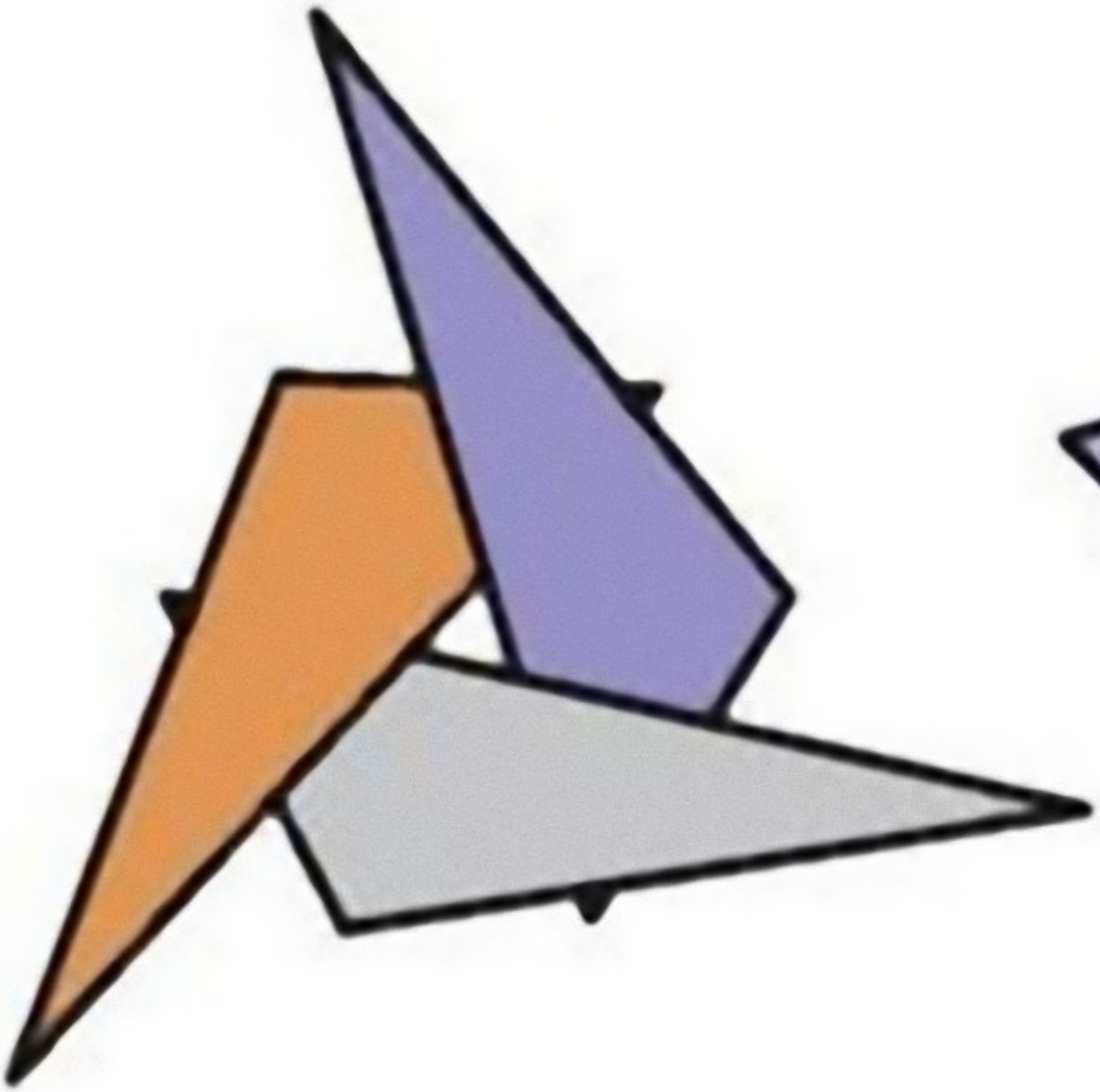
The Painter’s Algorithm

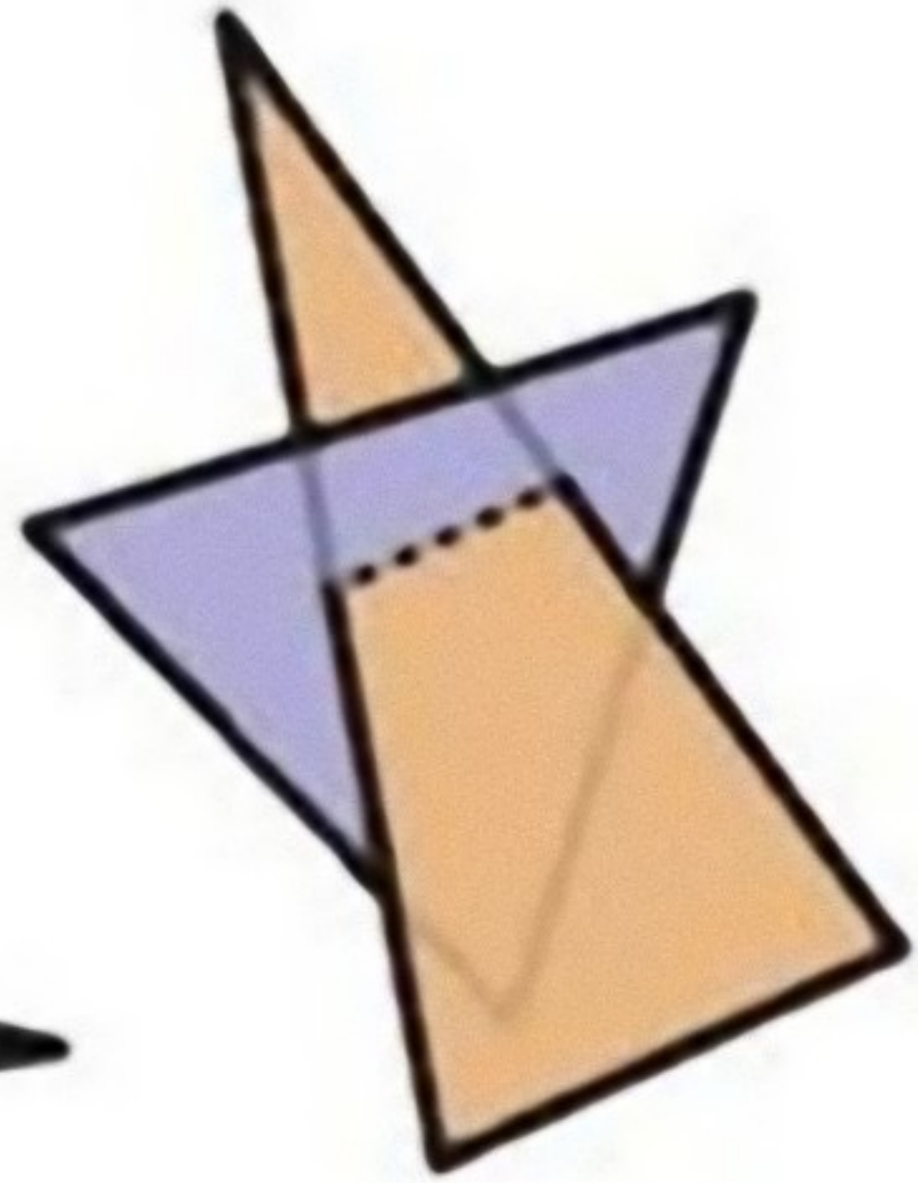
Painting by Julie Pollard

The Painter's Algorithm

- Sort objects or primitives in order from back to front
- Render primitives in sorted order, *over* the existing scene
- Does it always work?







The Z-Buffer Algorithm

Associates with each fragment a colour *and* a depth value

[images from fabiensanglard.net]

[images from fabiensanglard.net]

Z-Buffer Implementation

Can we rig our projection matrix to help us do Z-buffering?

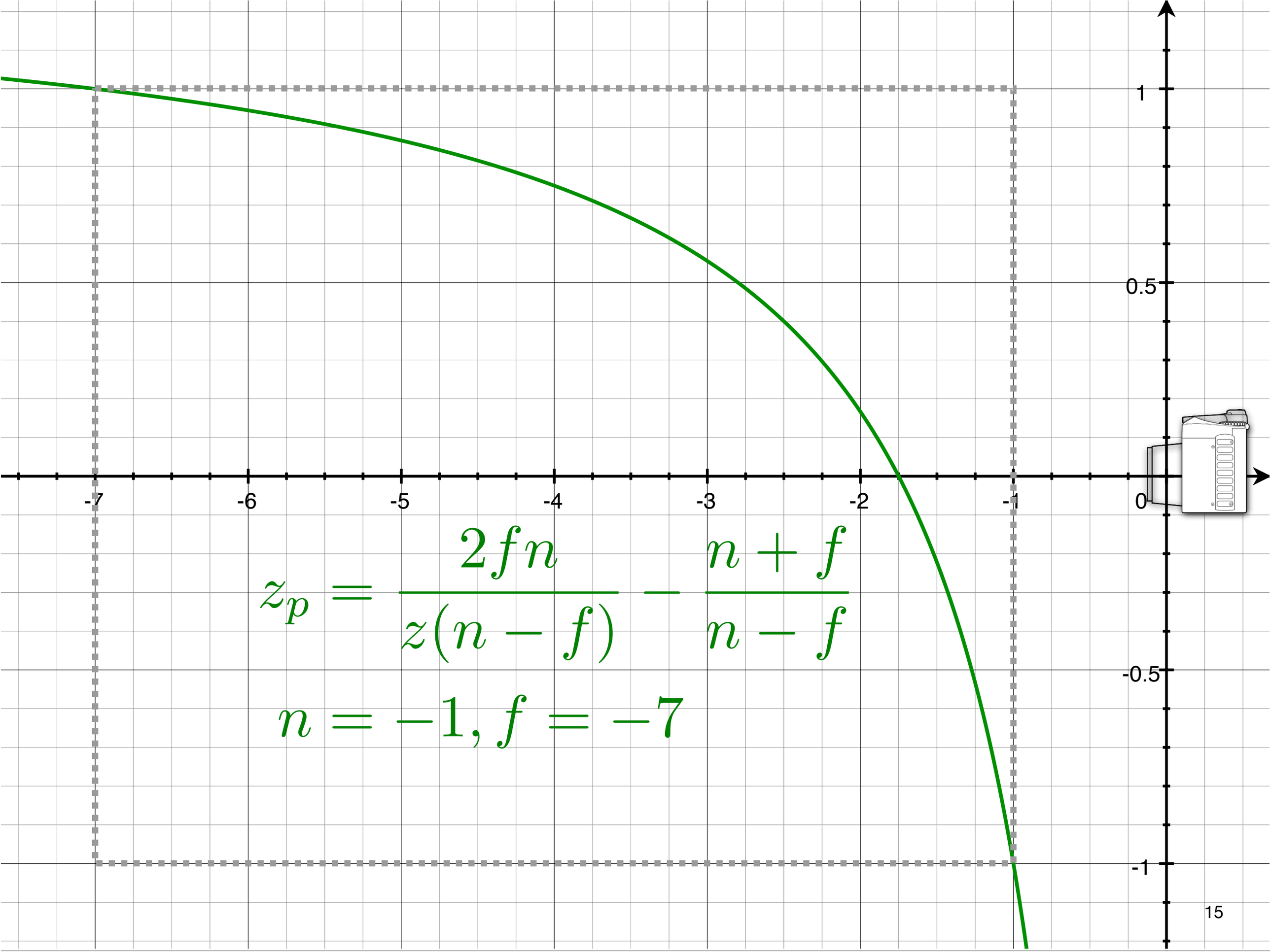
$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$z_p = \frac{1}{-z}$$

Our Final Perspective Projection Matrix

Where a is the aspect ratio:
width/height of the viewport

$$\begin{bmatrix} \frac{d}{a} & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{-2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



$$z_p = \frac{2fn}{z(n-f)} - \frac{n+f}{n-f}$$

$$n = -1, f = -7$$

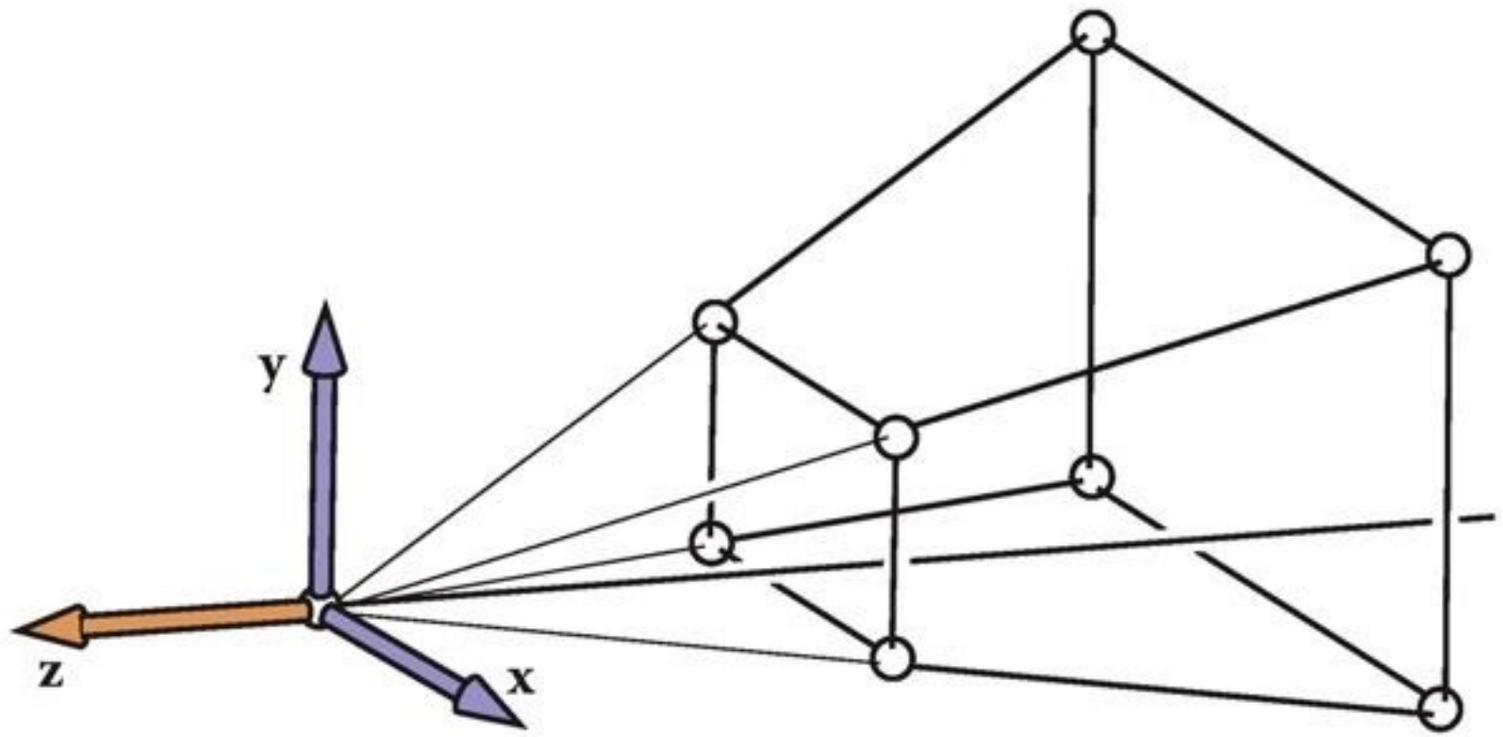
Z-Buffer Algorithm

- Compute a depth value for each fragment using the projection matrix
- If the fragment's depth is less (greater) than the existing depth stored at (x_p, y_p), write the fragment colour to the frame buffer
- Otherwise, discard the fragment

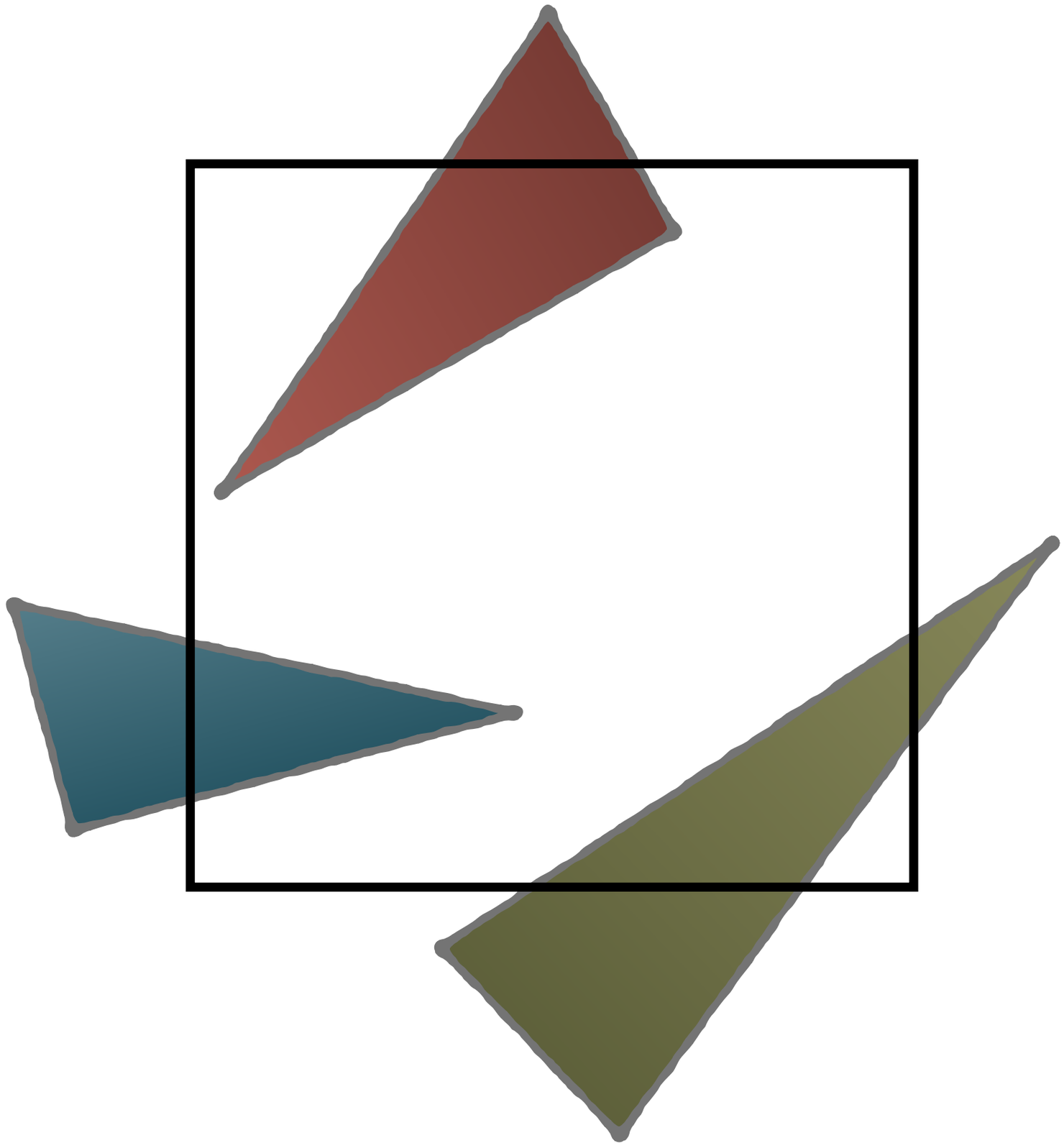
View Frustum

- The camera position, field of view, aspect ratio, near plane, and far plane together define a **view frustum**

- A truncated 3D pyramid shape that encloses everything we can see in our 3D scene

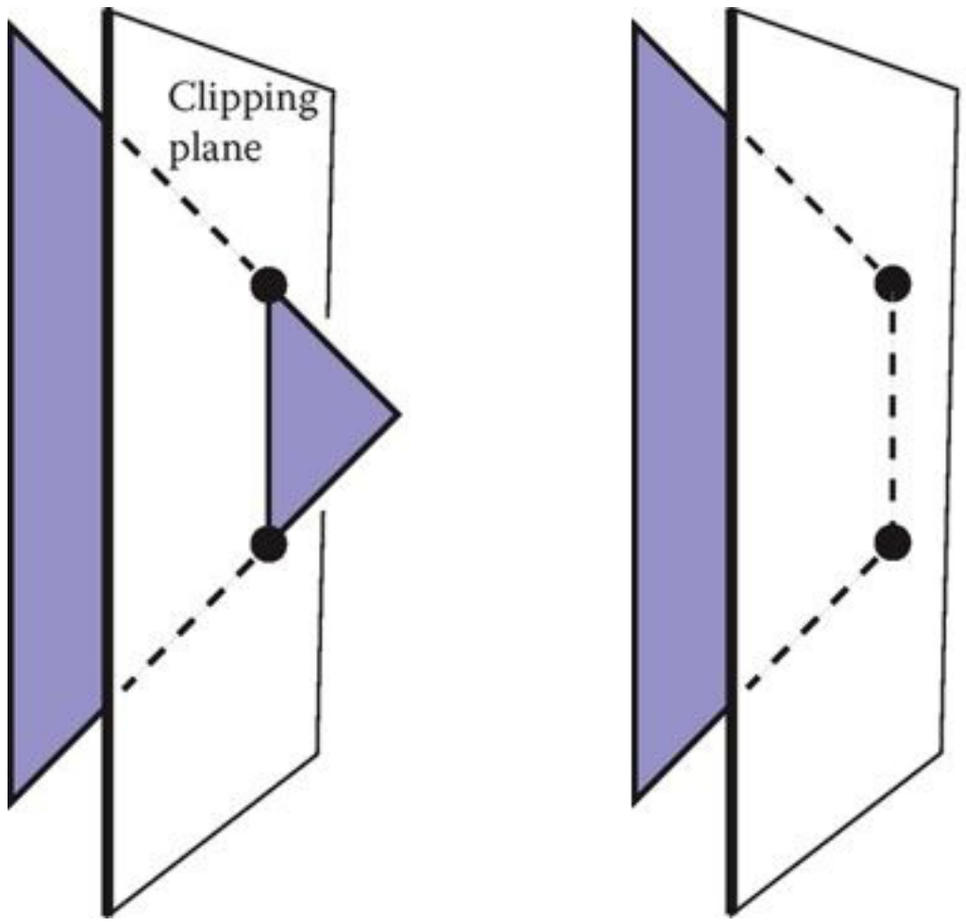


What about these triangles?

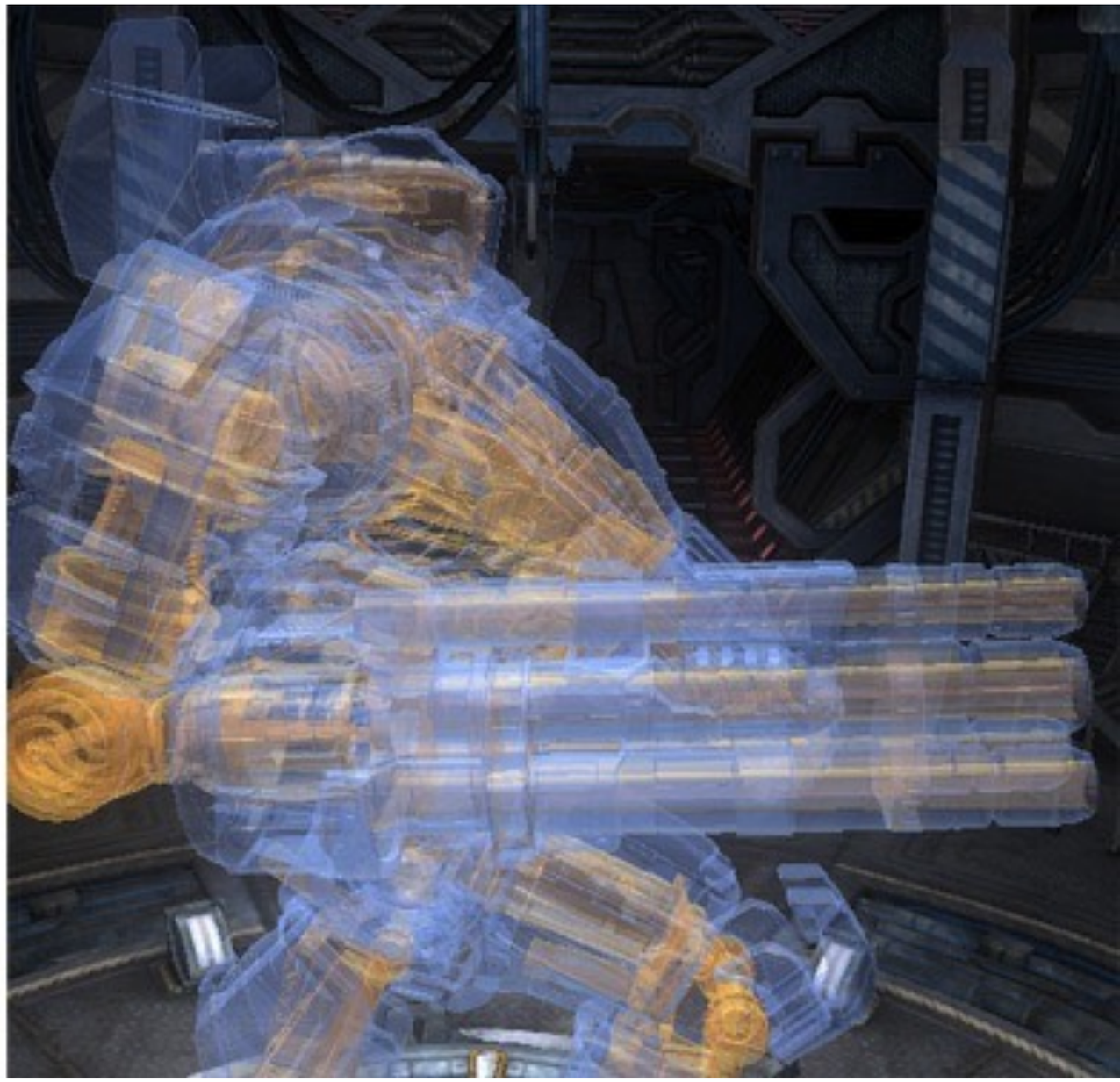


Primitive Clipping

One of the really, really helpful things that OpenGL does for us!



Does perspective projection with z-buffer always work?



Transparency!

From ATI Mecha Demo

Things to Remember

- Homogeneous coordinates allow us to compute the perspective divide with a projection matrix

- Projective rendering creates occlusion ordering problem
  - the z-buffer algorithm as a simple, fast, and effective solution

- The view frustum, defined by the projection matrix, encapsulates everything visible in our 3D scene
  - partially visible primitives are clipped by the OpenGL rasterizer