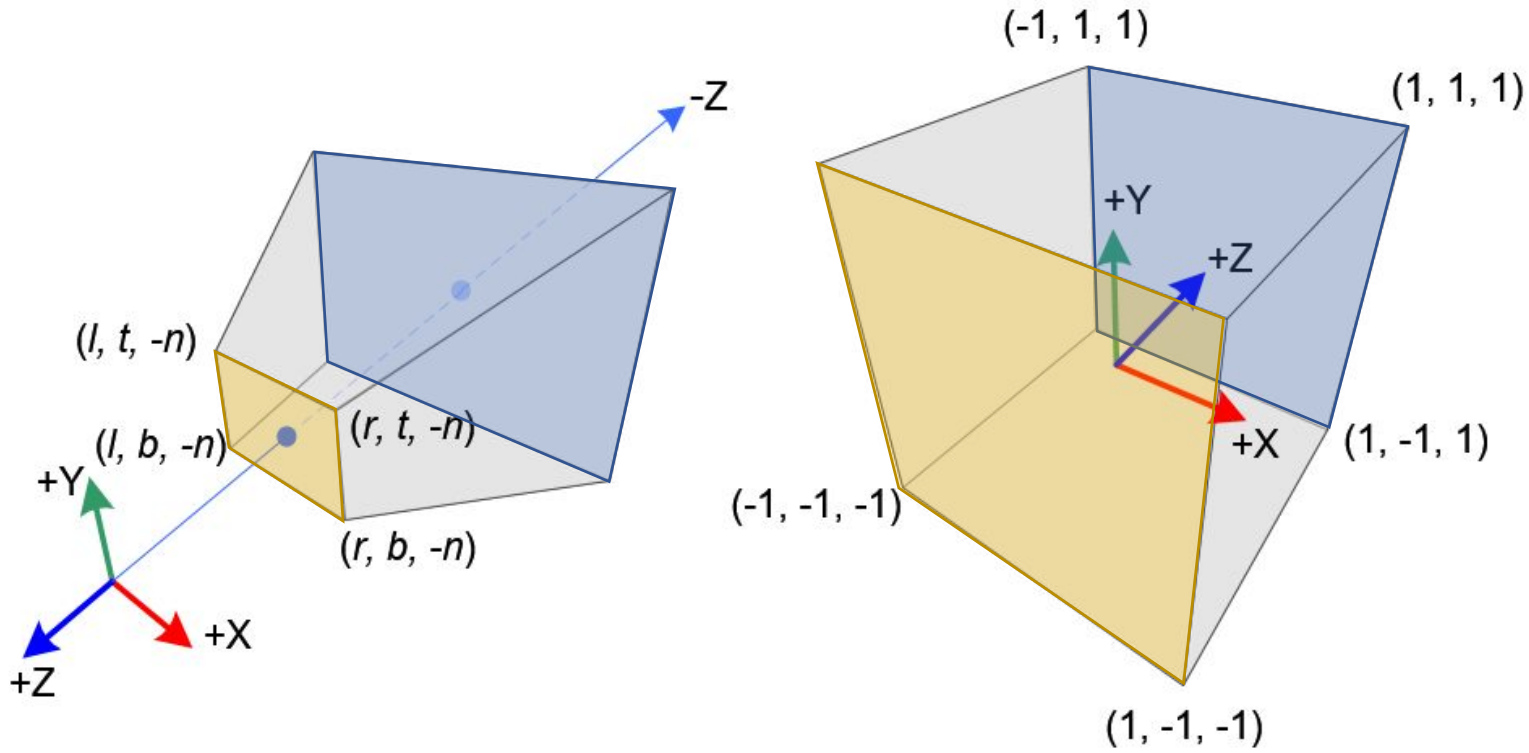
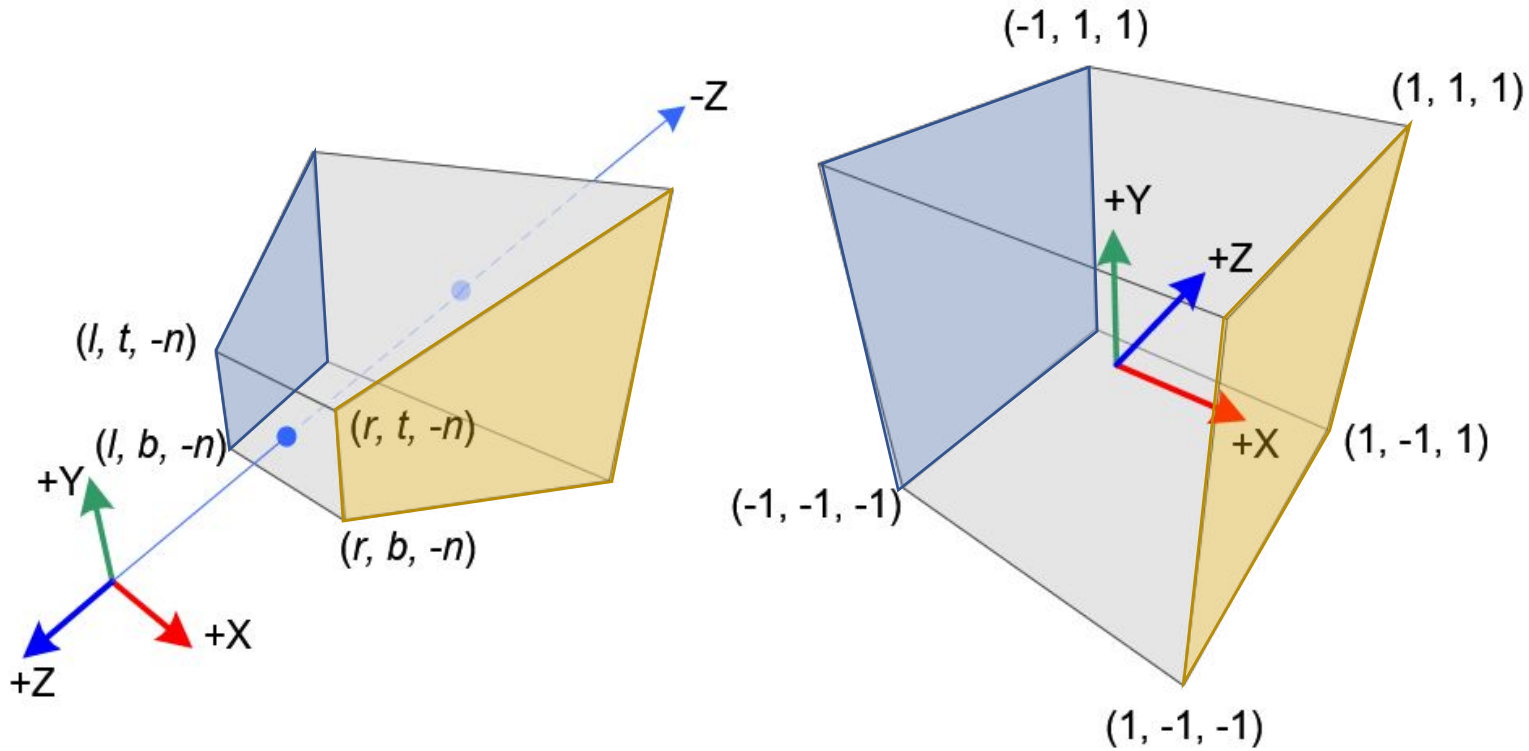
Z-Buffering

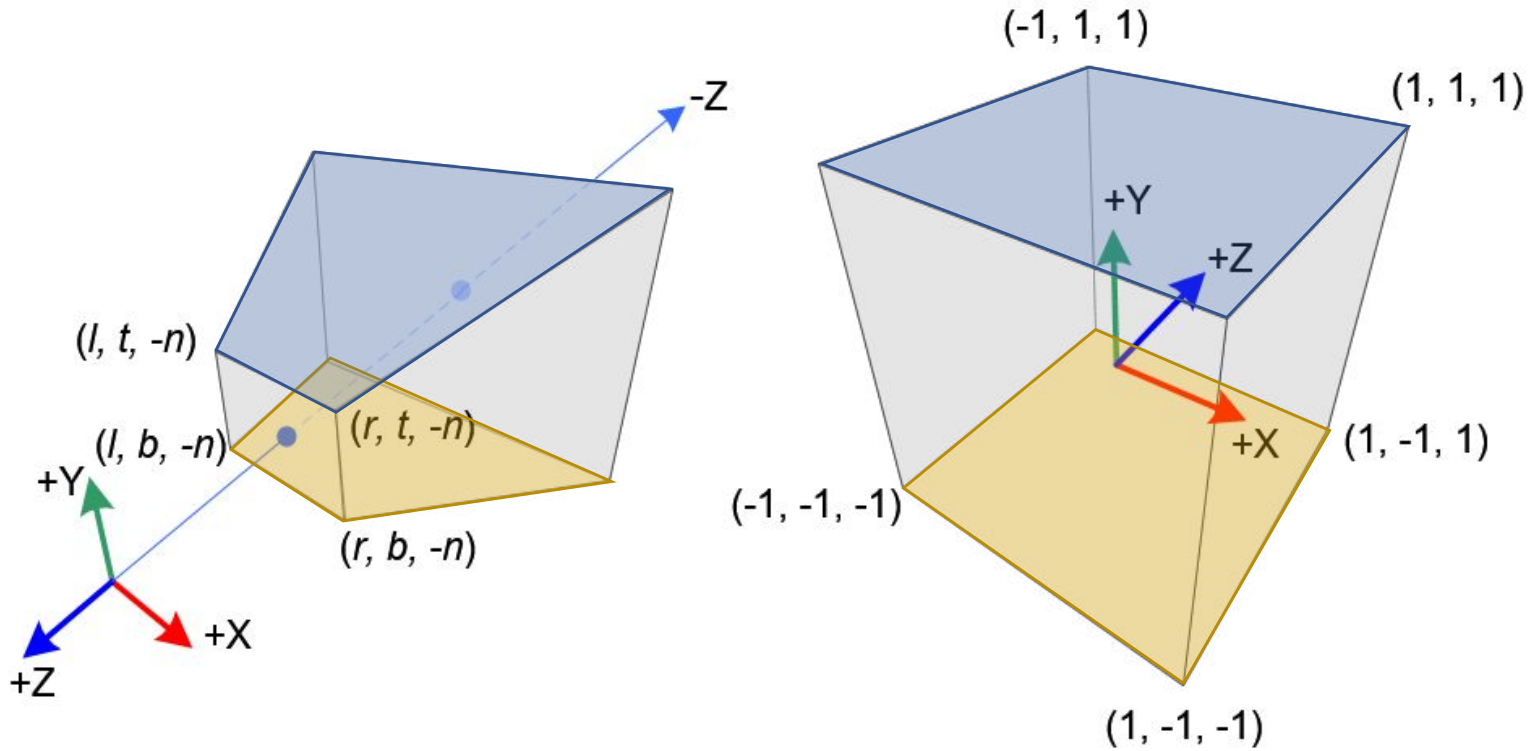
CSE 409 Computer Graphics

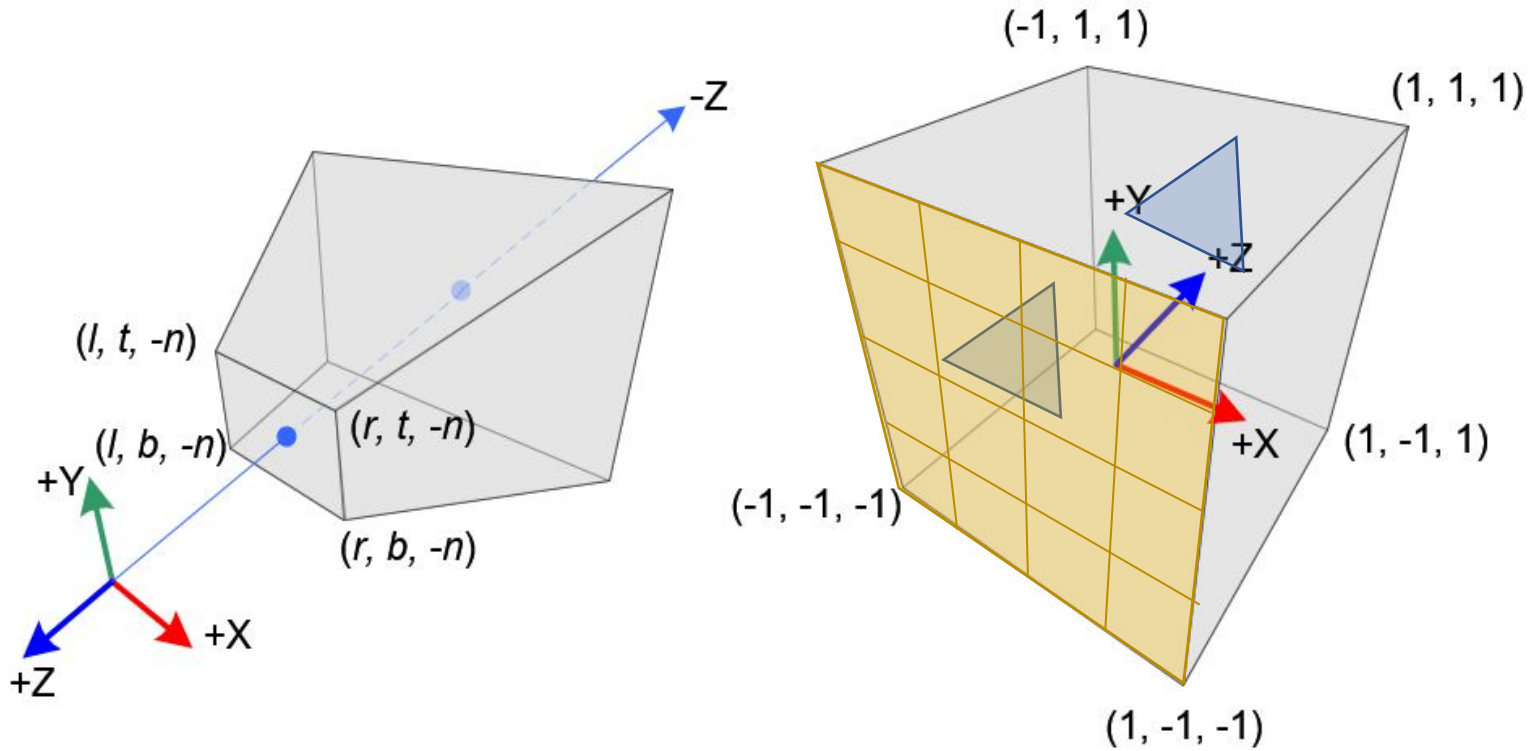
Md. Tareq Mahmood

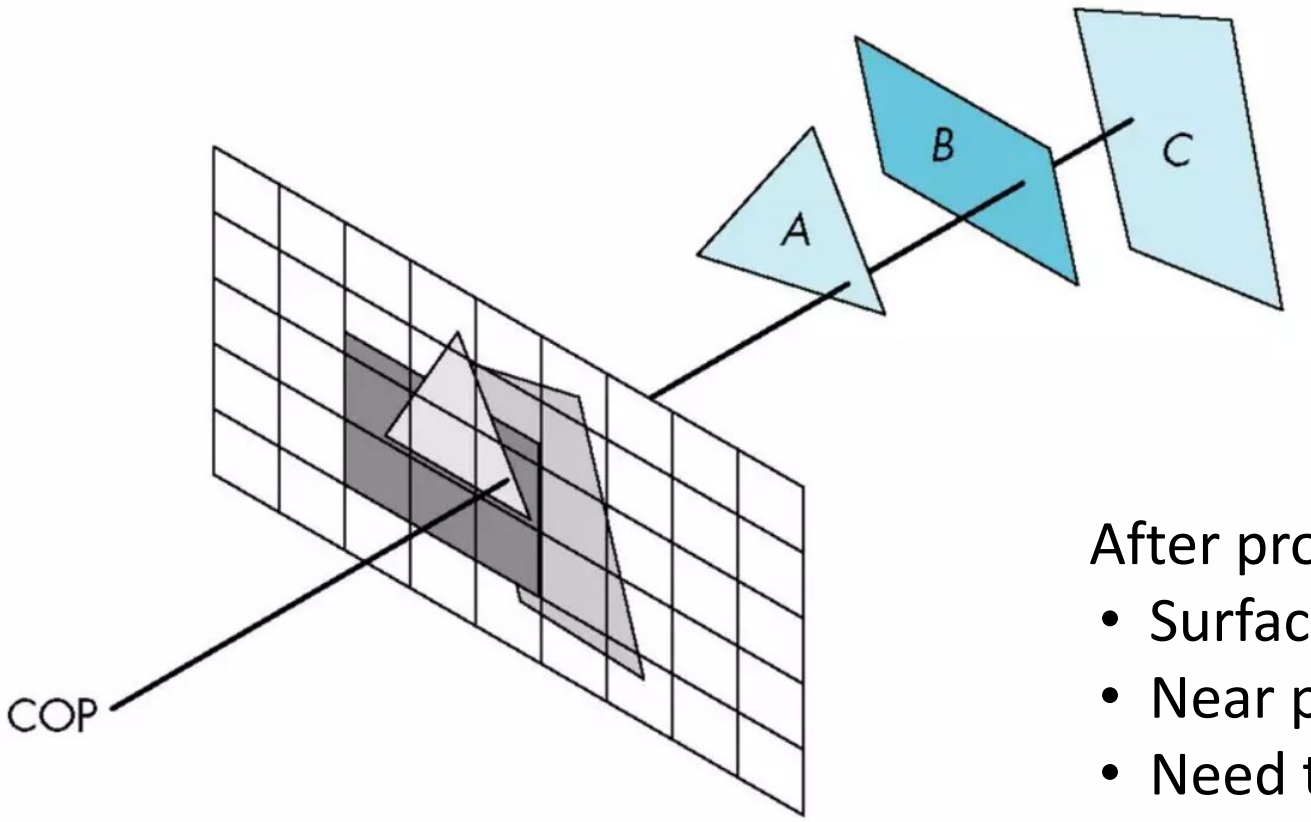
Department of CSE, BUET


-Z
(l, t, -n)
(l, b, -n)
(r, t, -n)
(r, b, -n)
+Y
+X
+Z
(-1, 1, 1)
(1, 1, 1)
+Y
+Z
+X
(1, -1, 1)
(-1, -1, -1)
(1, -1, -1)


-Z
(l, t, -n)
(l, b, -n)
(r, t, -n)
(r, b, -n)
+Y
+X
+Z
(-1, 1, 1)
(1, 1, 1)
+Y
+Z
+X
(1, -1, 1)
(-1, -1, -1)
(1, -1, -1)


-Z
(l, t, -n)
(l, b, -n)
(r, t, -n)
(r, b, -n)
+Y
+X
+Z
(-1, 1, 1)
(1, 1, 1)
+Y
+Z
+X
(1, -1, 1)
(-1, -1, -1)
(1, -1, -1)


-Z
(l, t, -n)
(l, b, -n)
(r, t, -n)
(r, b, -n)
+Y
+X
+Z
(-1, 1, 1)
(1, 1, 1)
+Y
+Z
+X
(1, -1, 1)
(-1, -1, -1)
(1, -1, -1)

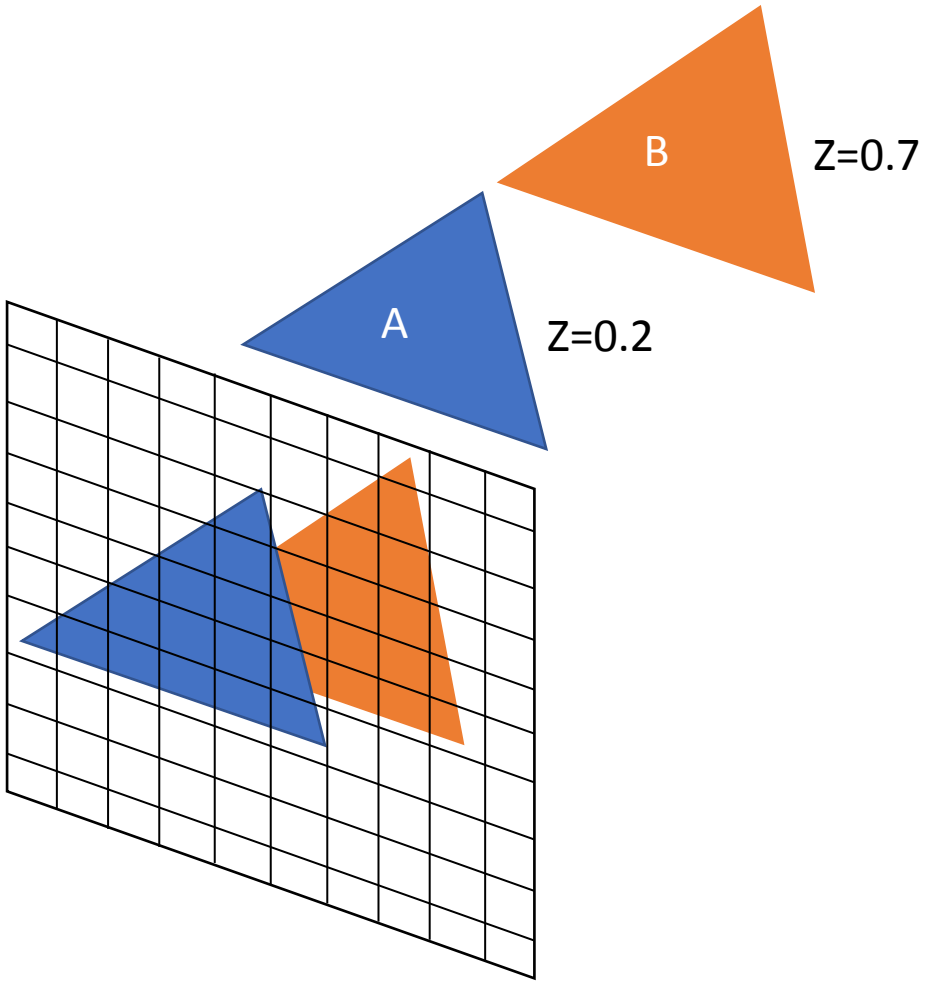


After projection transformation

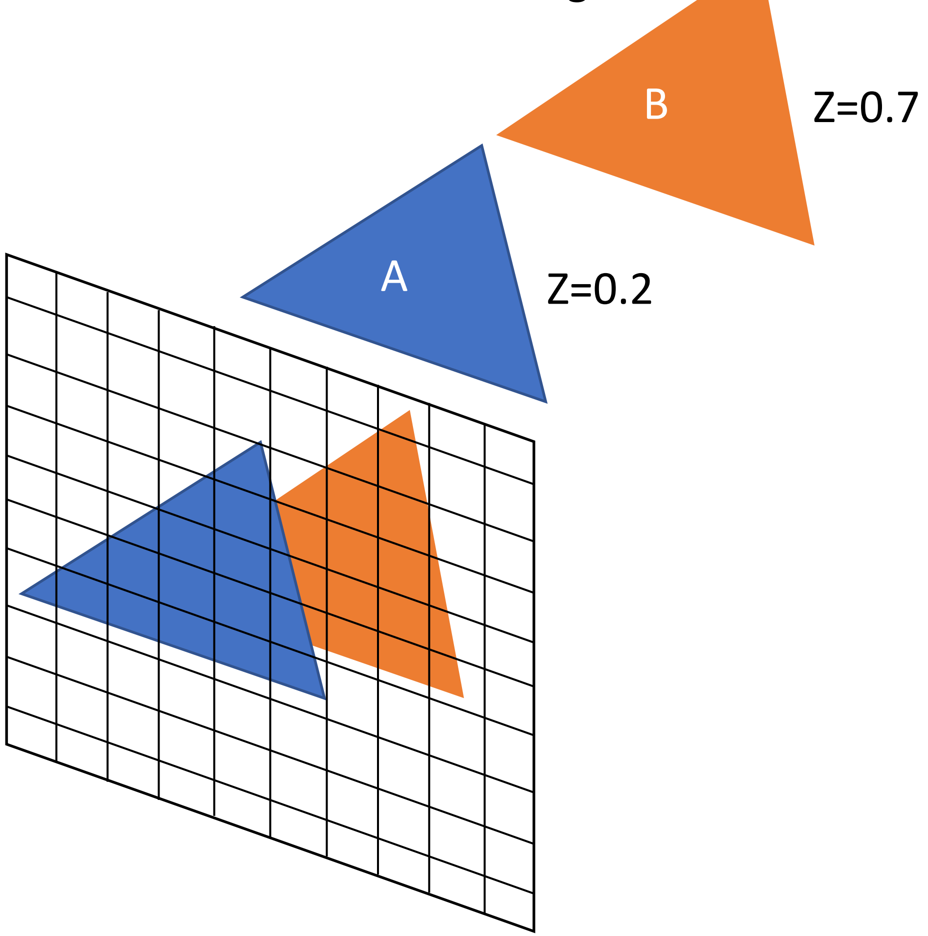
- Surfaces projected on the near plane
- Near plane divided into grids
- Need to decide the color of the grids

Z-Buffering

- A z-buffer, is a type of data buffer used in computer graphics to represent **depth information** of objects in 3D space **from a particular perspective**.

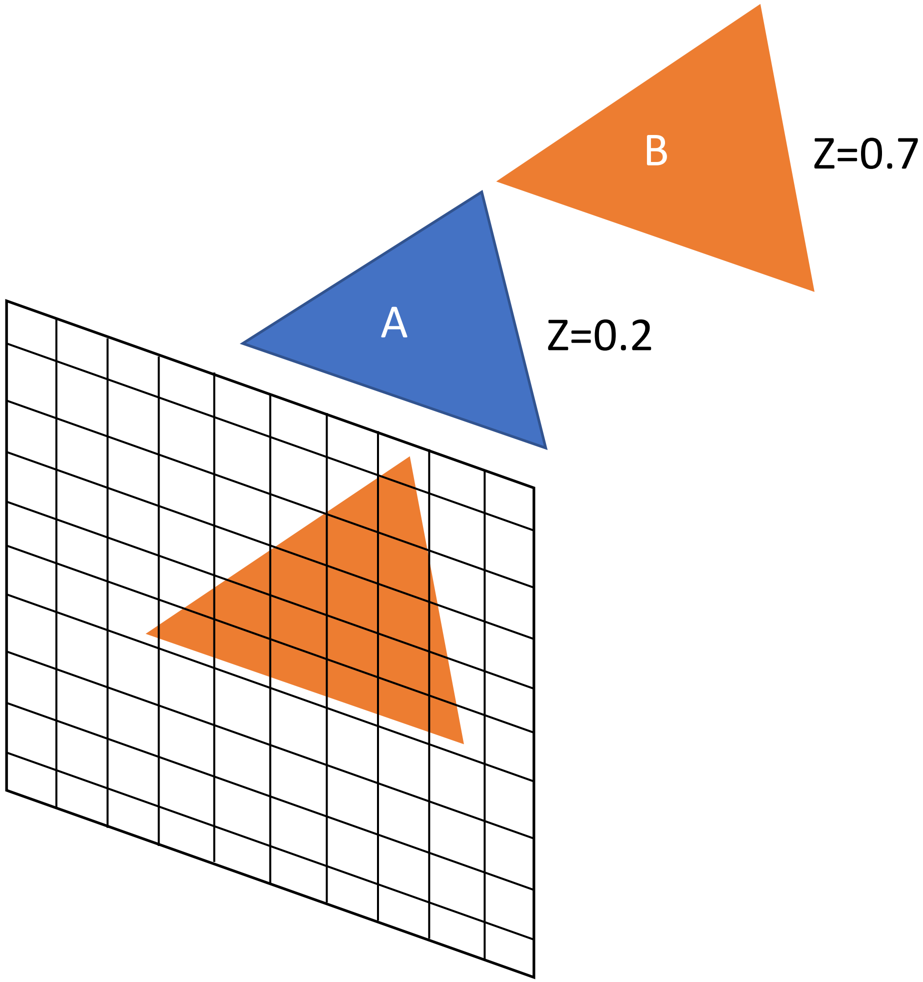

B
Z=0.7
A
Z=0.2

Let, z_max = 1
Initialize z-buffer with z_max
The color of all cells are background



1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Drawing B First

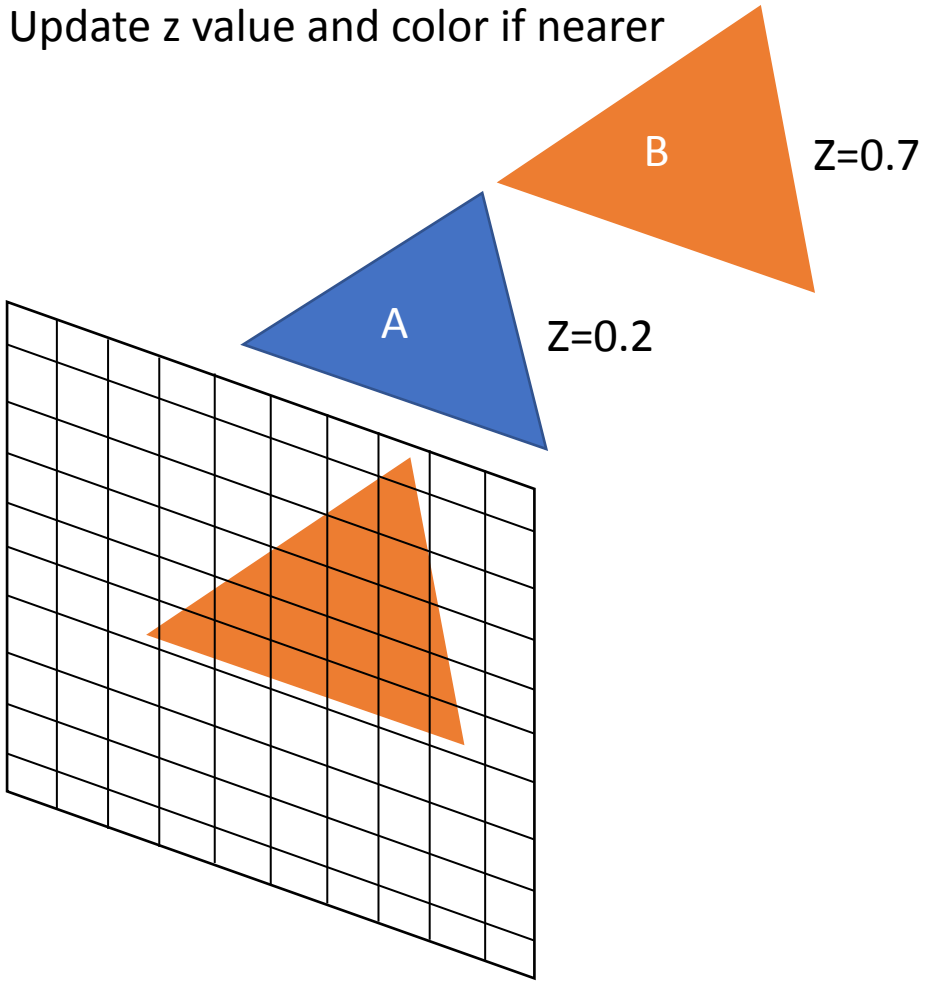


1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Drawing B First

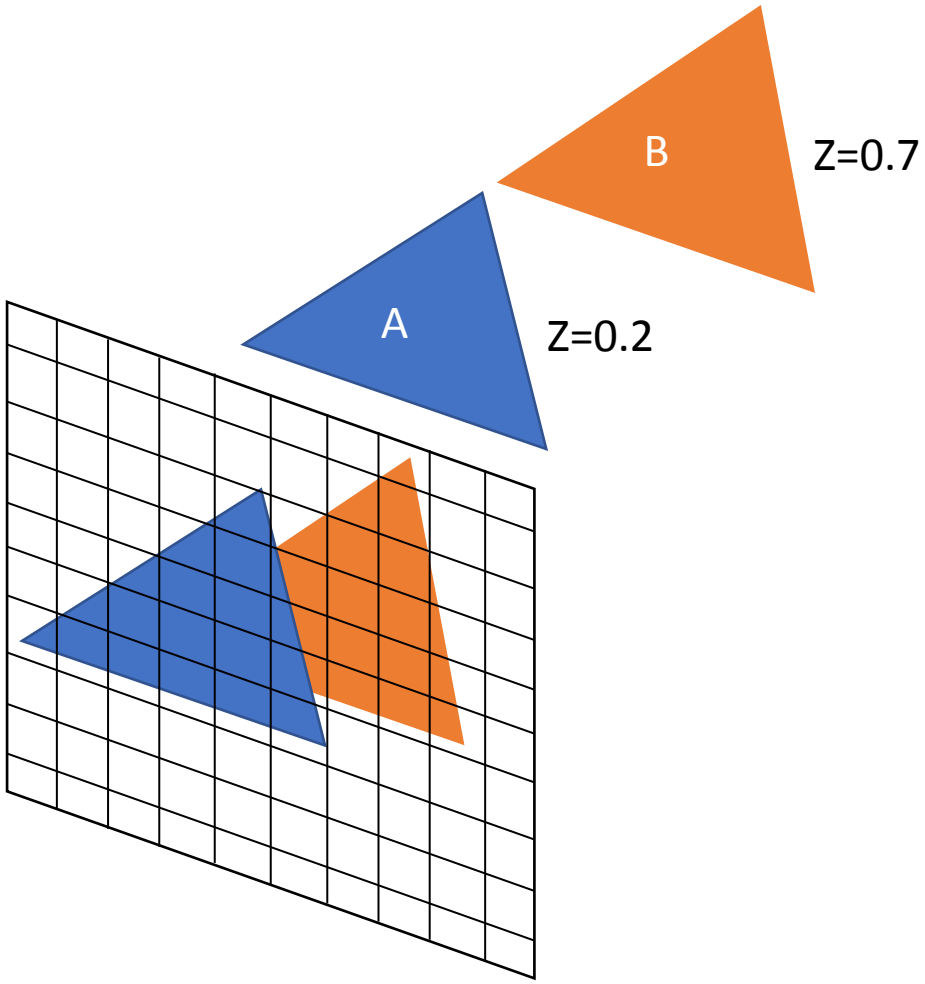
Find z value of all grids that is covered by B

Update z value and color if nearer



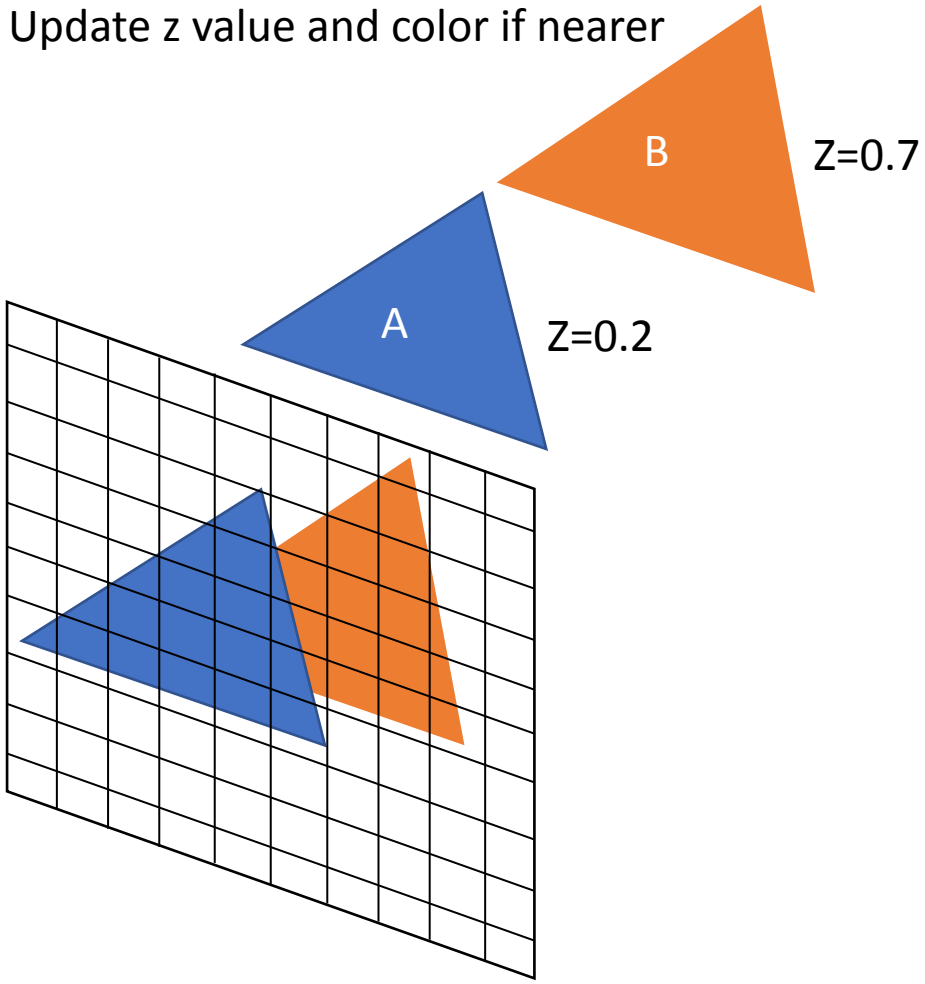
1	1	1	1	1	1	0.7	1	1	1
1	1	1	1	1	0.7	0.7	1	1	1
1	1	1	1	0.7	0.7	0.7	0.7	1	1
1	1	1	1	0.7	0.7	0.7	0.7	1	1
1	1	1	0.7	0.7	0.7	0.7	0.7	0.7	1
1	1	0.7	0.7	0.7	0.7	0.7	0.7	0.7	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Drawing A Second



1	1	1	1	1	1	0.7	1	1	1
1	1	1	1	1	0.7	0.7	1	1	1
1	1	1	1	0.7	0.7	0.7	0.7	1	1
1	1	1	1	0.7	0.7	0.7	0.7	1	1
1	1	1	0.7	0.7	0.7	0.7	0.7	0.7	1
1	1	0.7	0.7	0.7	0.7	0.7	0.7	0.7	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Drawing A Second
Find z value of all grids that is covered by A
Update z value and color if nearer



1	1	1	1	1	1	0.7	1	1	1
1	1	1	1	1	0.7	0.7	1	1	1
1	1	1	1	0.2	0.7	0.7	0.7	1	1
1	1	1	0.2	0.2	0.2	0.7	0.7	1	1
1	1	0.2	0.2	0.2	0.2	0.7	0.7	0.7	1
1	0.2	0.2	0.2	0.2	0.2	0.7	0.7	0.7	1
0.2	0.2	0.2	0.2	0.2	0.2	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Algorithmics [edit]

The following pseudocode demonstrates the process of z-buffering:

```
// First of all, initialize the depth of each pixel.
d(i, j) = infinite // Max length

// Initialize the color value for each pixel to the background color
c(i, j) = background color

// For each polygon, do the following steps :
for (each pixel in polygon's projection)
{
    // Find depth i.e, z of polygon
    //   at (x, y) corresponding to pixel (i, j)
    if (z < d(i, j))
    {
        d(i, j) = z;
        c(i, j) = color;
    }
}
```

We actually track 2 buffers

Z buffer (for depth)

Frame buffer (for color)

Algorithmics [edit]

The following pseudocode demonstrates the process of z-buffering:

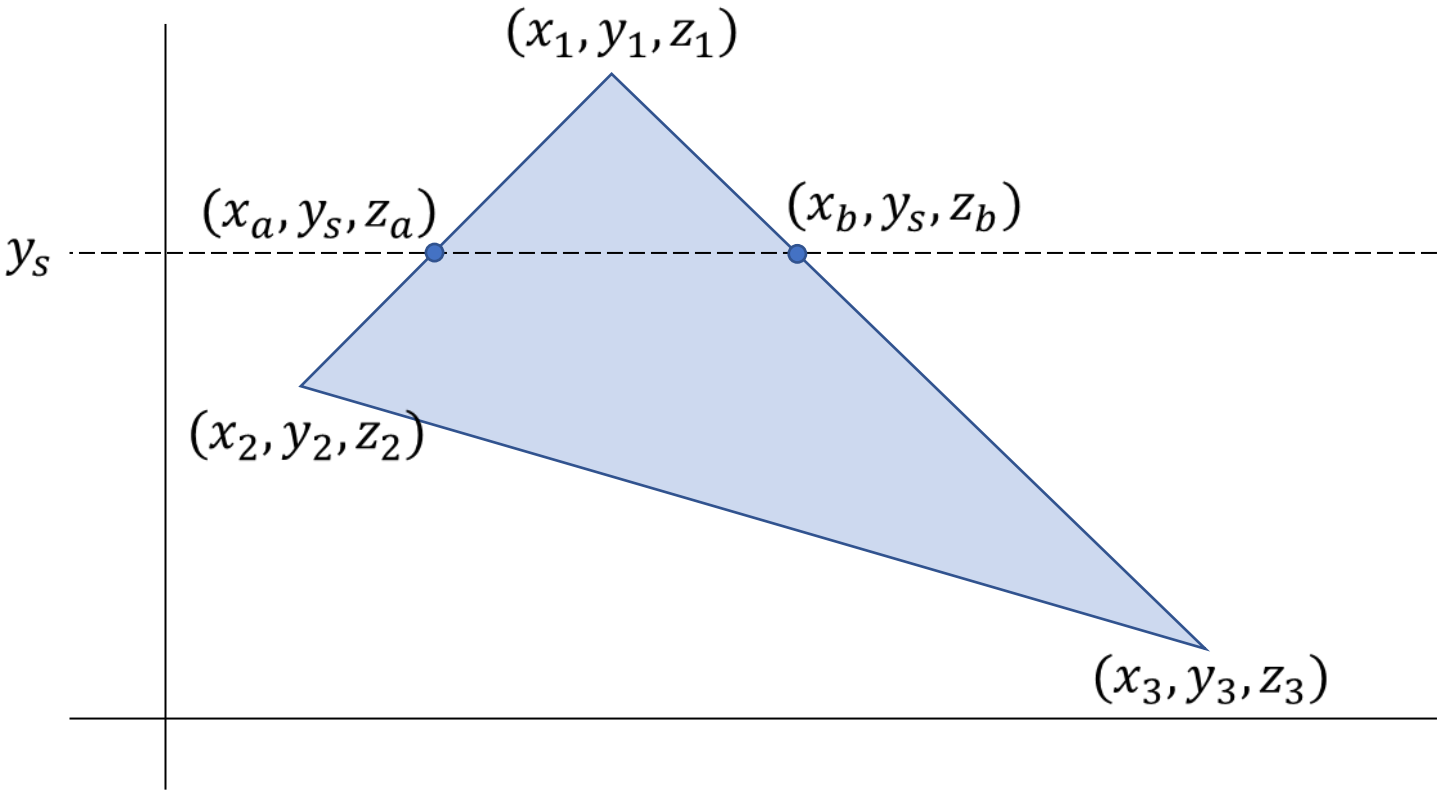
```
// First of all, initialize the depth of each pixel.
d(i, j) = infinite // Max length

// Initialize the color value for each pixel to the background color
c(i, j) = background color

// For each polygon, do the following steps :
for (each pixel in polygon's projection)
{
    // Find depth i.e, z of polygon
    //   at (x, y) corresponding to pixel (i, j)
    if (z < d(i, j))
    {
        d(i, j) = z;
        c(i, j) = color;
    }
}
```

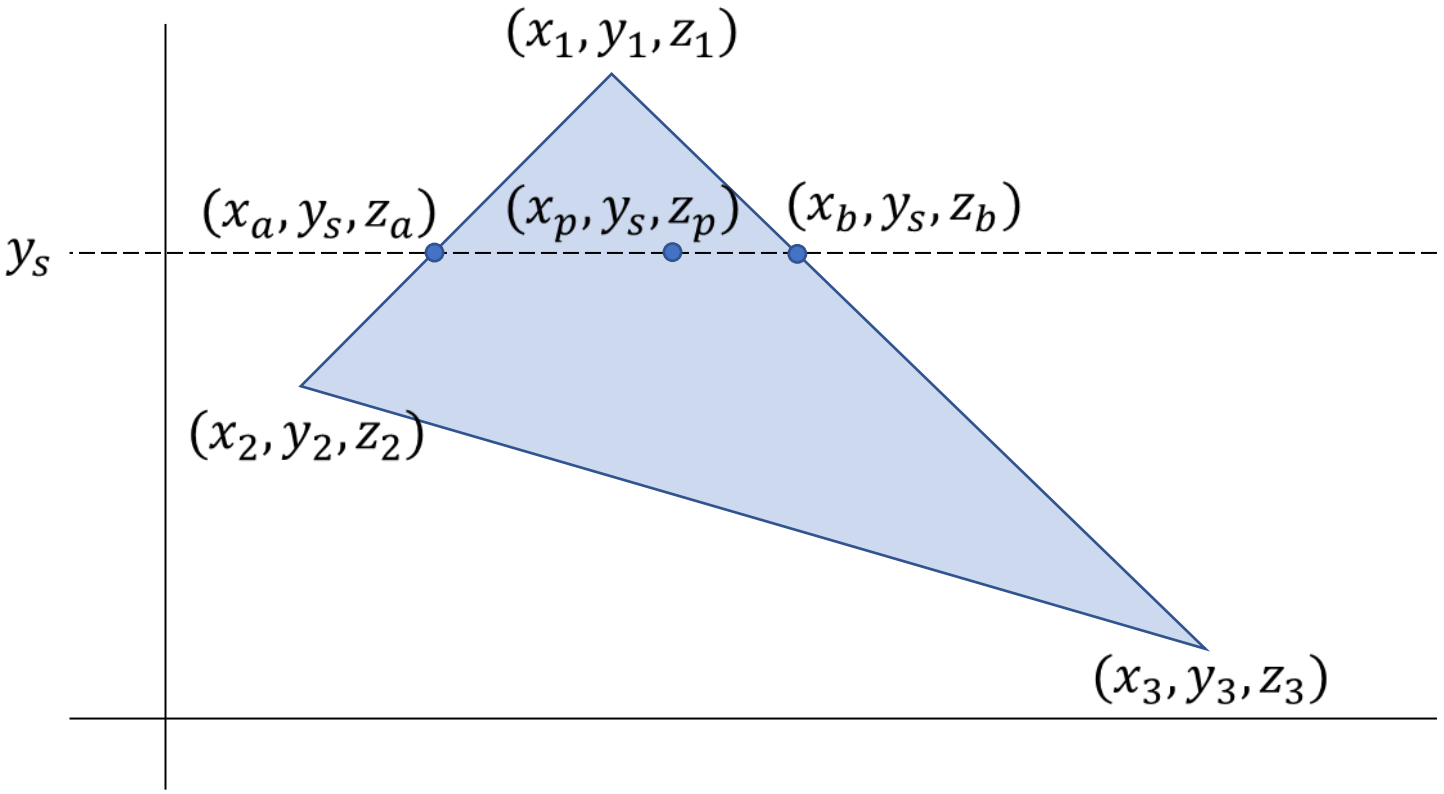
How can we find the z values of a pixel?

Finding Z



- Given y_s
- Find x_a, z_a
- Find x_b, z_b

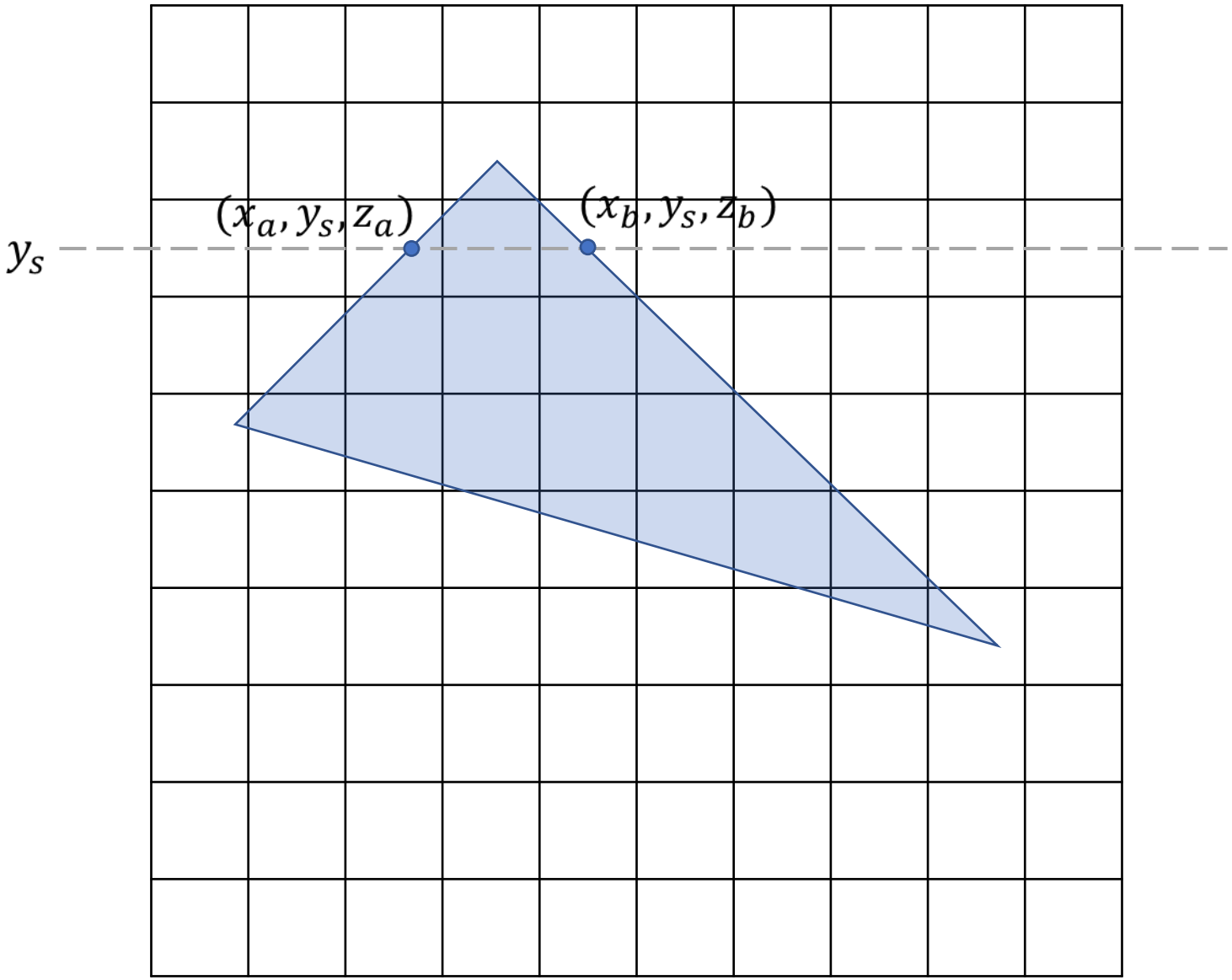
Finding Z



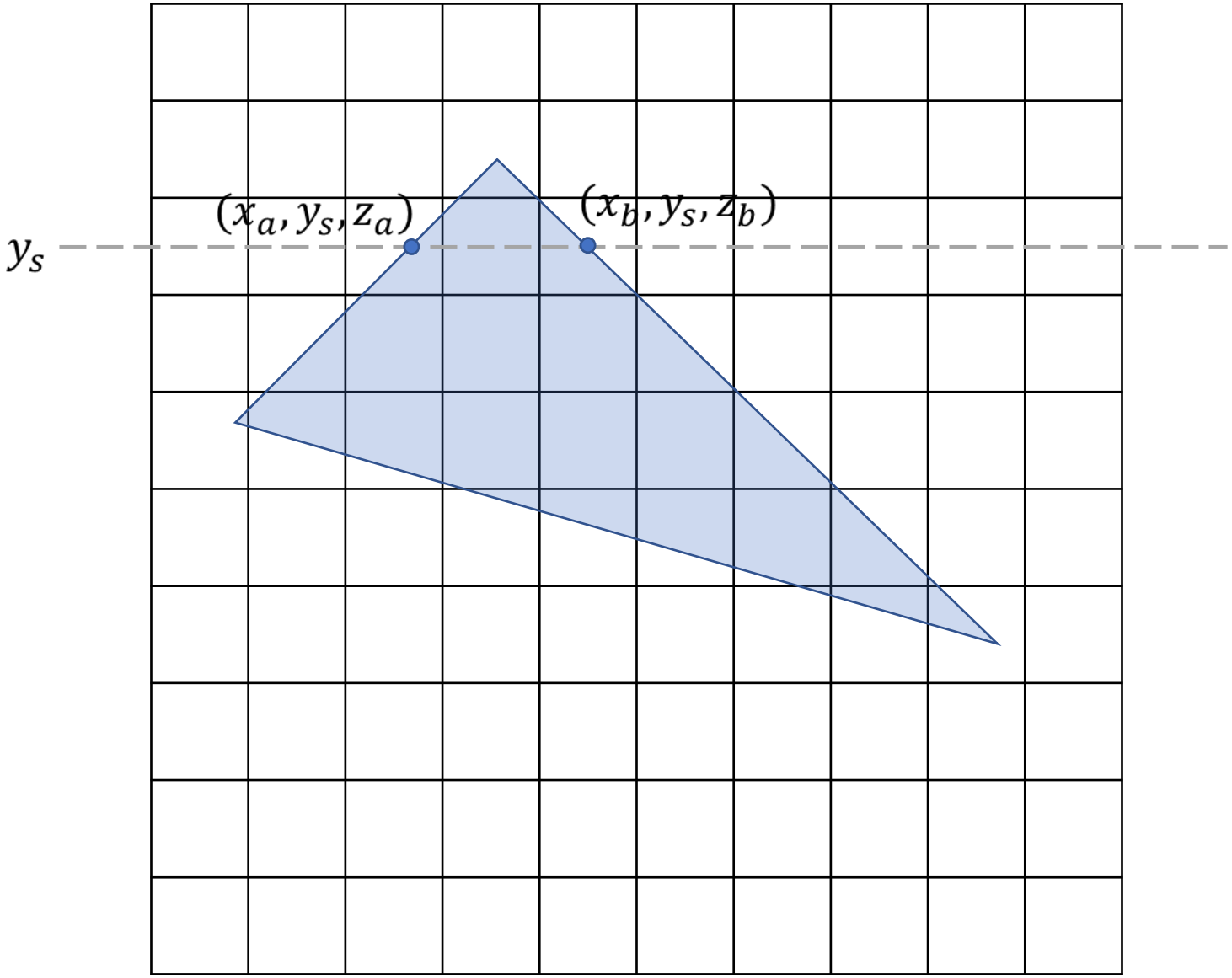
- Given x_p
- Find z_p

- XY Coordinate of cell = XY Coordinate of the midpoint of the cell

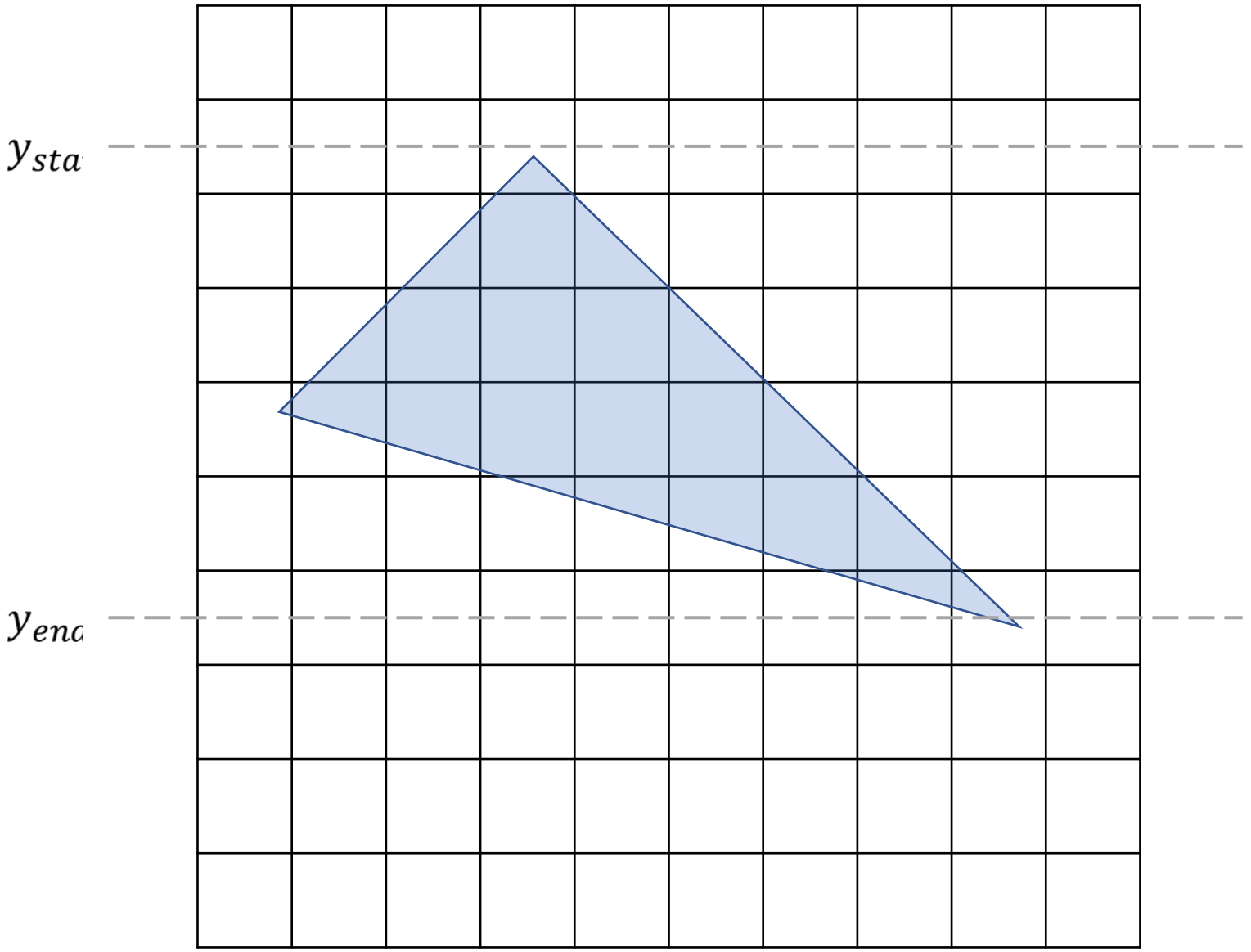
- Choose a row
- Draw scan line y_s along the middle of the row
- Find two intersecting points



- Find the cells covered by the triangle on that row
- Find the left intersecting column and the right intersecting column
- For each covered cell, find depth (z)



- Top scanline (row)
- Bottom scanline (row)



Algorithm

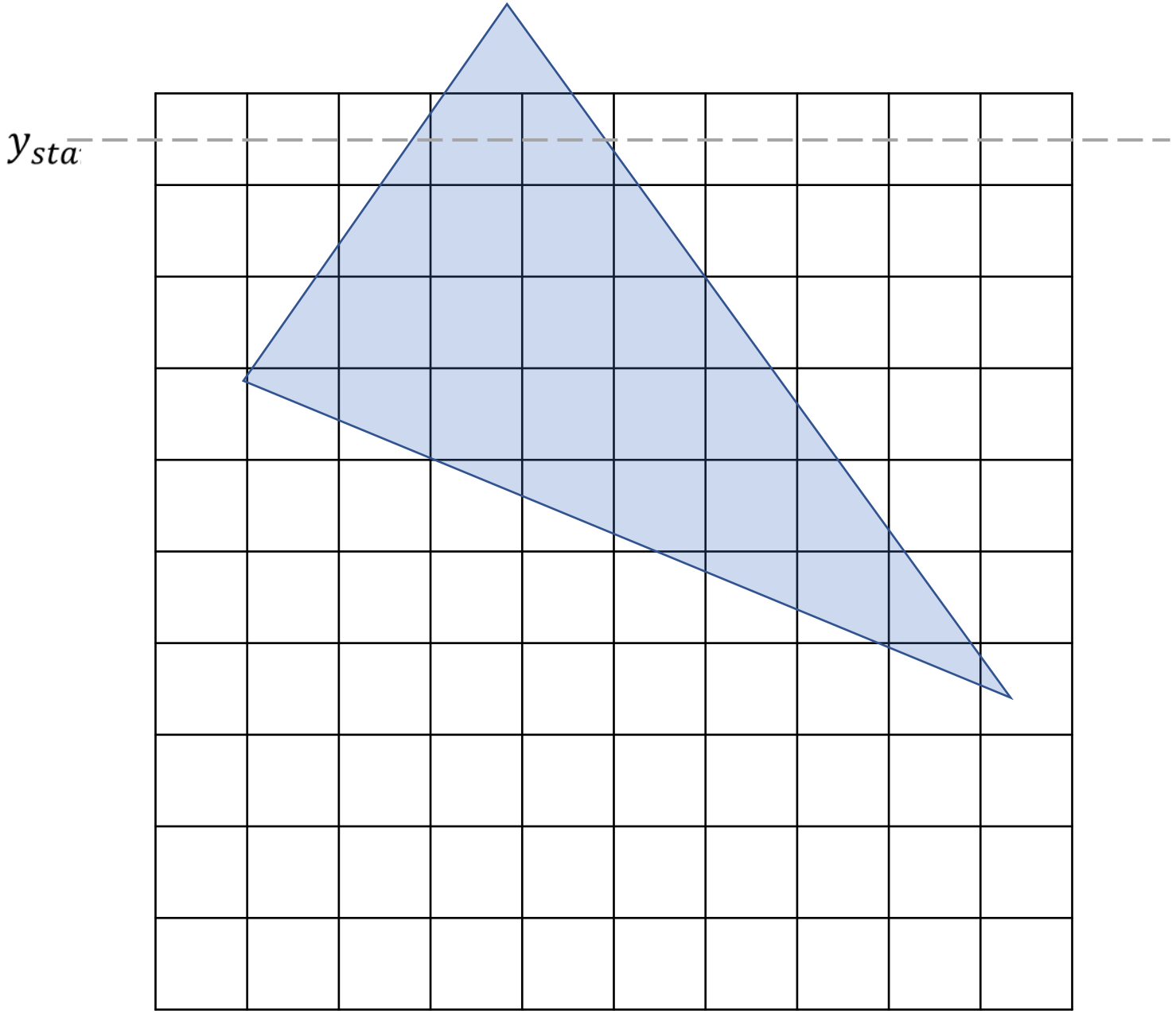
```
d(i, j) = z_max          // z-buffer
c(i, j) = background color    // frame buffer

For each triangle T:
    Find the top and bottom rows
    For each row (i) in [top row, bottom row]:
        Find the left and right intersecting columns
        For each column (j) in [left column, right column]:
            Find z value at pixel (i, j)
            If z < d(i, j):
                d(i, j) = z
                c(i, j) = T.color
```

Clipping Y

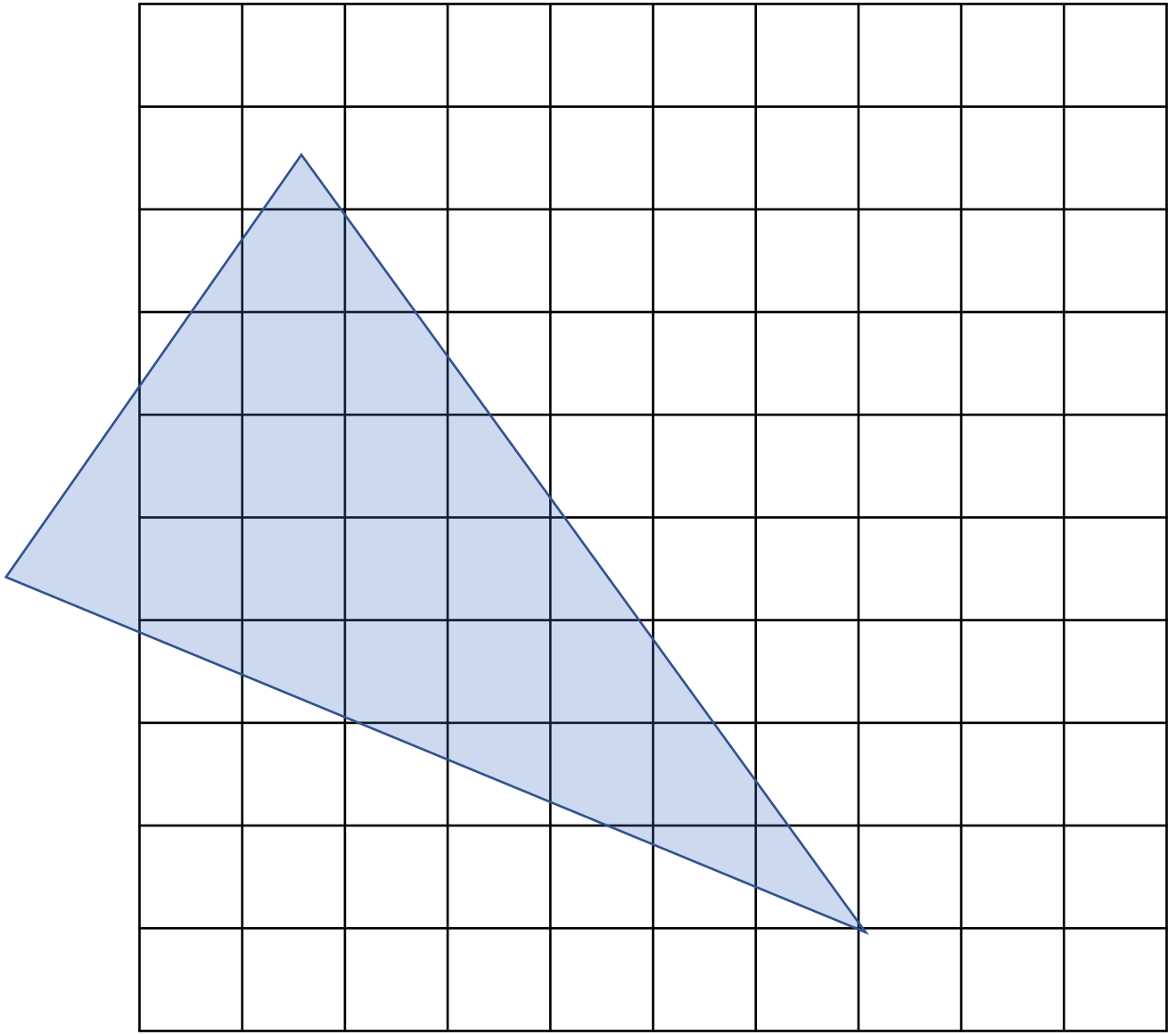
- If the triangle start before the topmost row, start scanning from the topmost row.

- If the triangle ends after the bottommost row, stop scanning at the bottommost row.



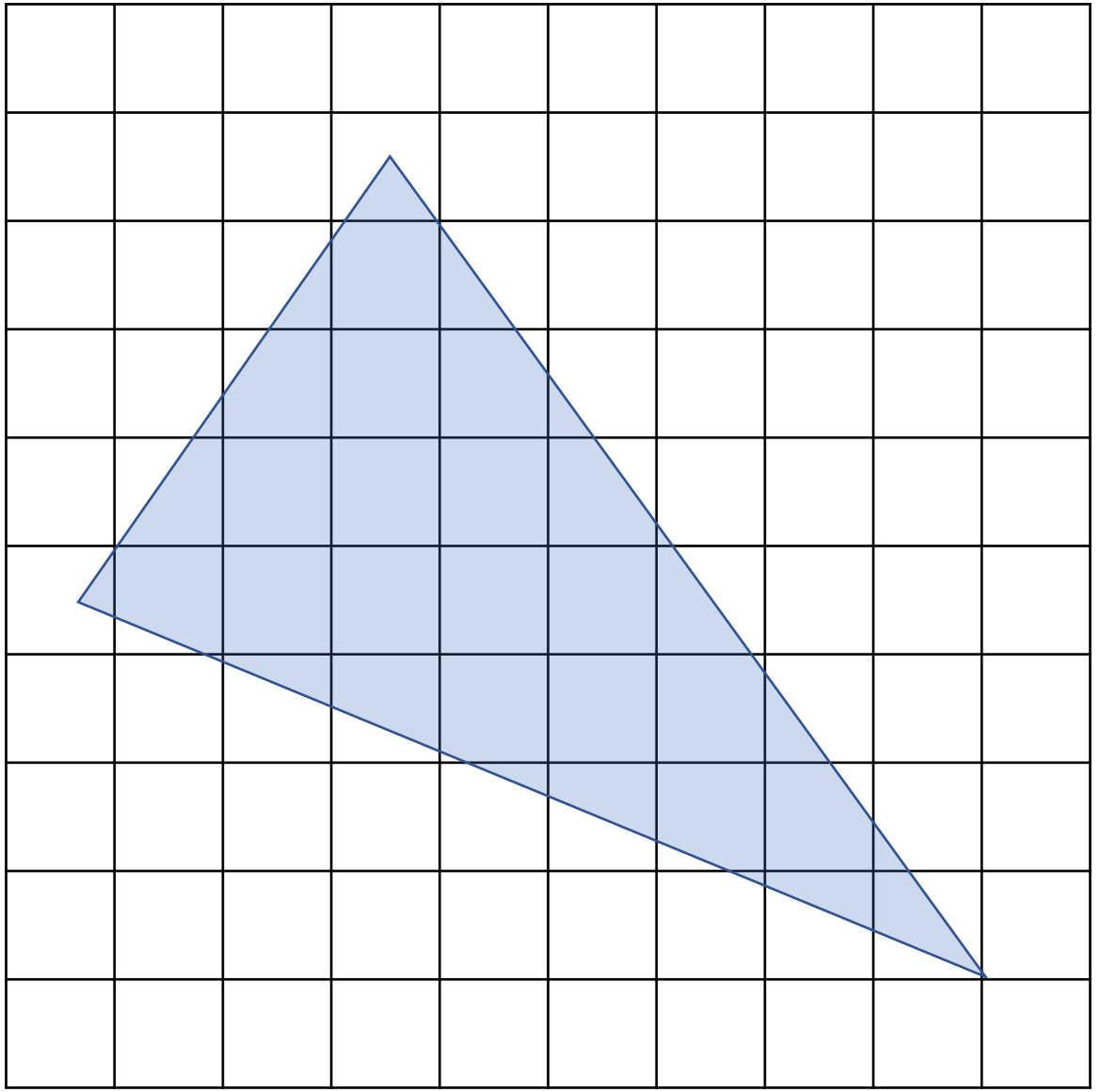
Clipping X

- Do it yourself



Clipping Z

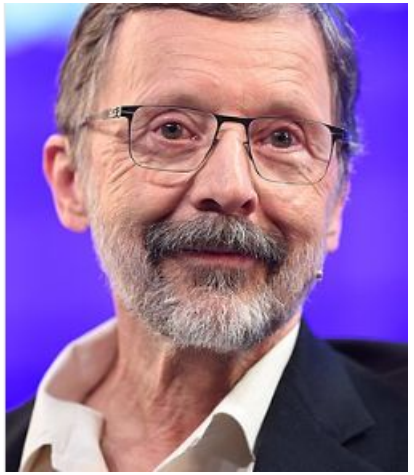
- If z value is not within [-zmin, zmax] range, ignore it.
- Do not update z or color



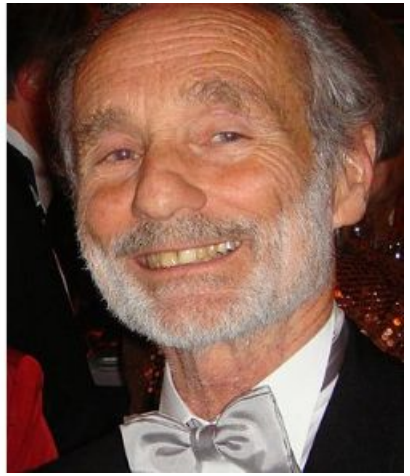
History

- Z-buffer concept is most often attributed to Edwin Catmull.

Edwin Earl "Ed" Catmull is an American computer scientist who is the co-founder of Pixar and was the President of Walt Disney Animation Studios. He has been honored for his contributions to 3D computer graphics, including the 2019 ACM Turing Award.



Wolfgang Straßer was a German computer scientist. In his dissertation in 1974, he described the process for the first time, which Edwin Catmull later gave the name Z-Buffer. He continued to make significant contributions in the areas of anti-aliasing and free-form modeling of curves and



- Although Wolfgang Straßer described this idea in his 1974 Ph.D. thesis months before Catmull's invention.

Thanks to…

- https://en.wikipedia.org/wiki/Z-buffering