CMP205: Computer Graphics

جامعة القاهرة

Lecture 5: Triangle Drawing and Hidden Surface Removal

Ahmed S. Kaseb

Fall 2018

Slides by: Dr. Mohamed Alaa El-Dien Aly

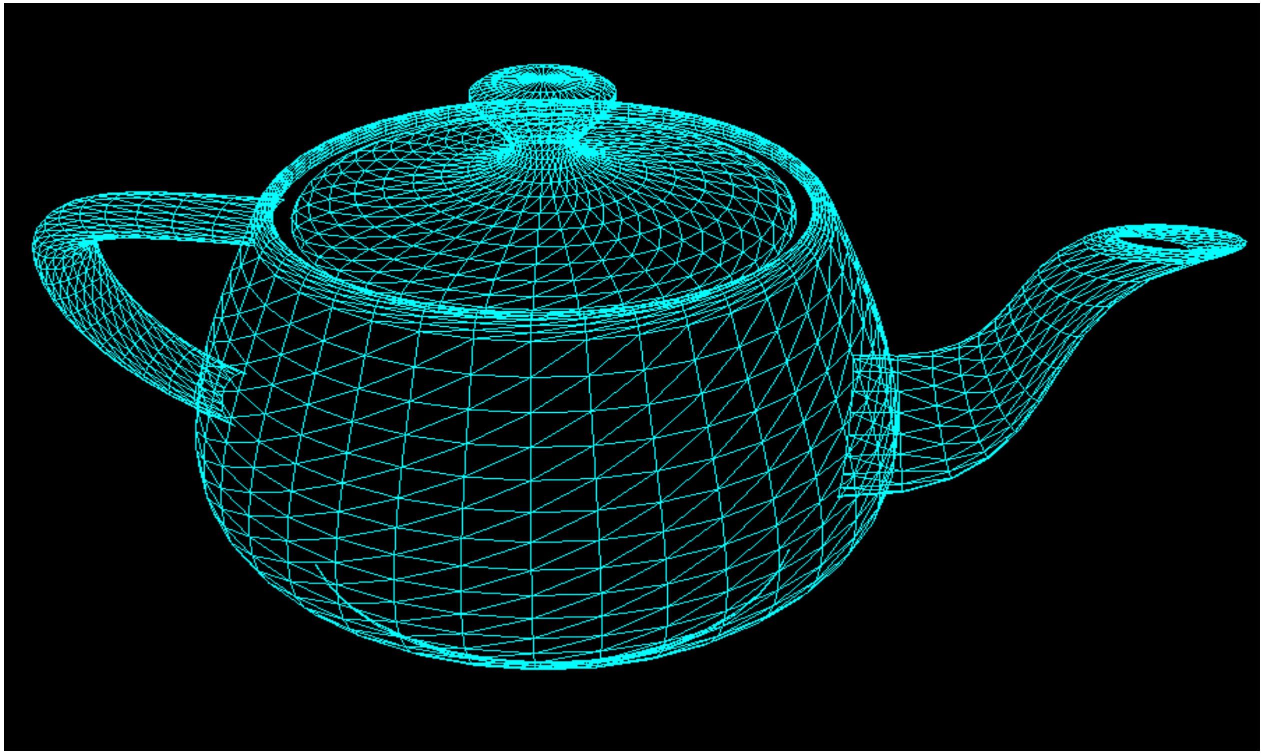
Agenda

- Triangle Drawing
- Anti-aliasing
- Z-Buffer

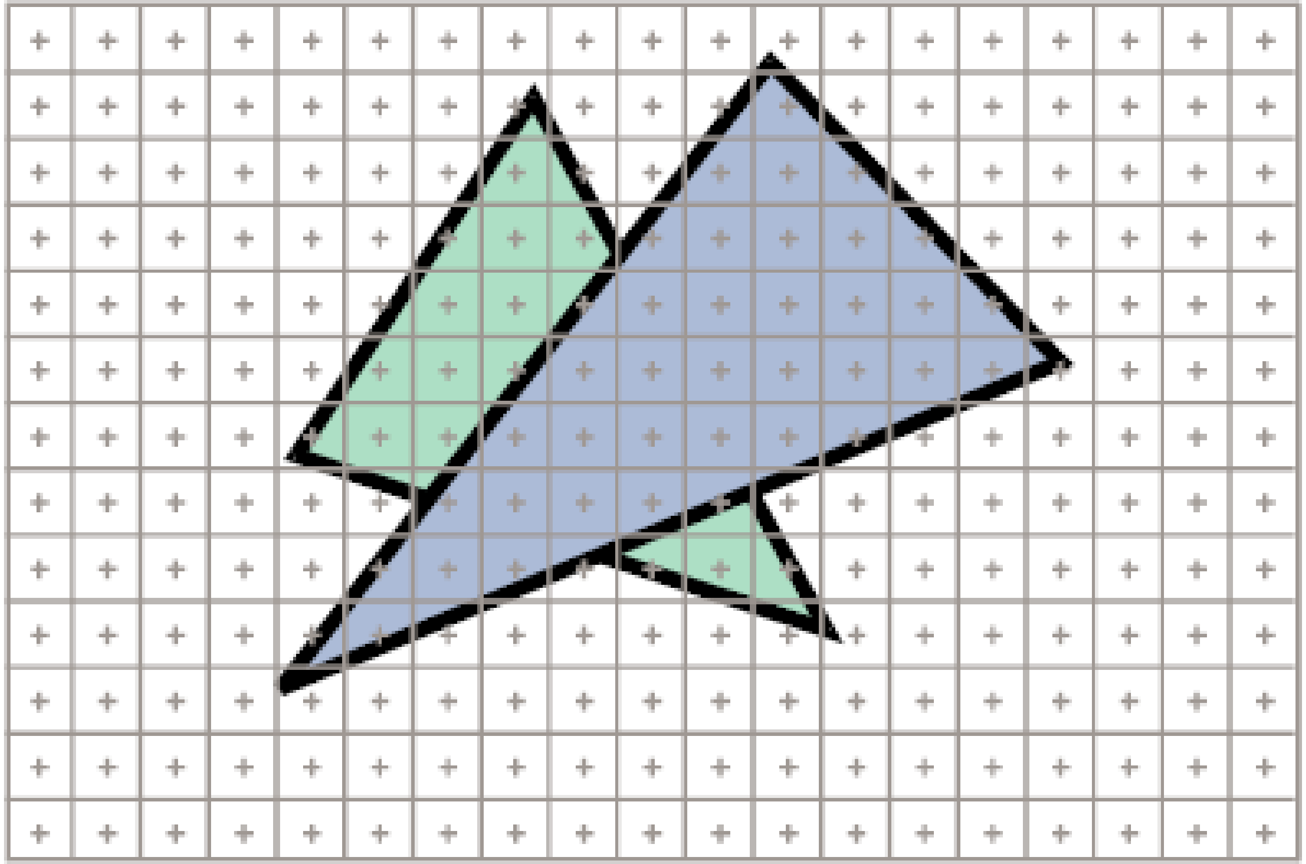
Acknowledgment: Some slides adapted from Steve Marschner and Maneesh Agrawala

Triangle Rasterization

- Second primitive shape, after the line!
- Used for modeling and shading surfaces
- Assign properties to vertices and interpolate

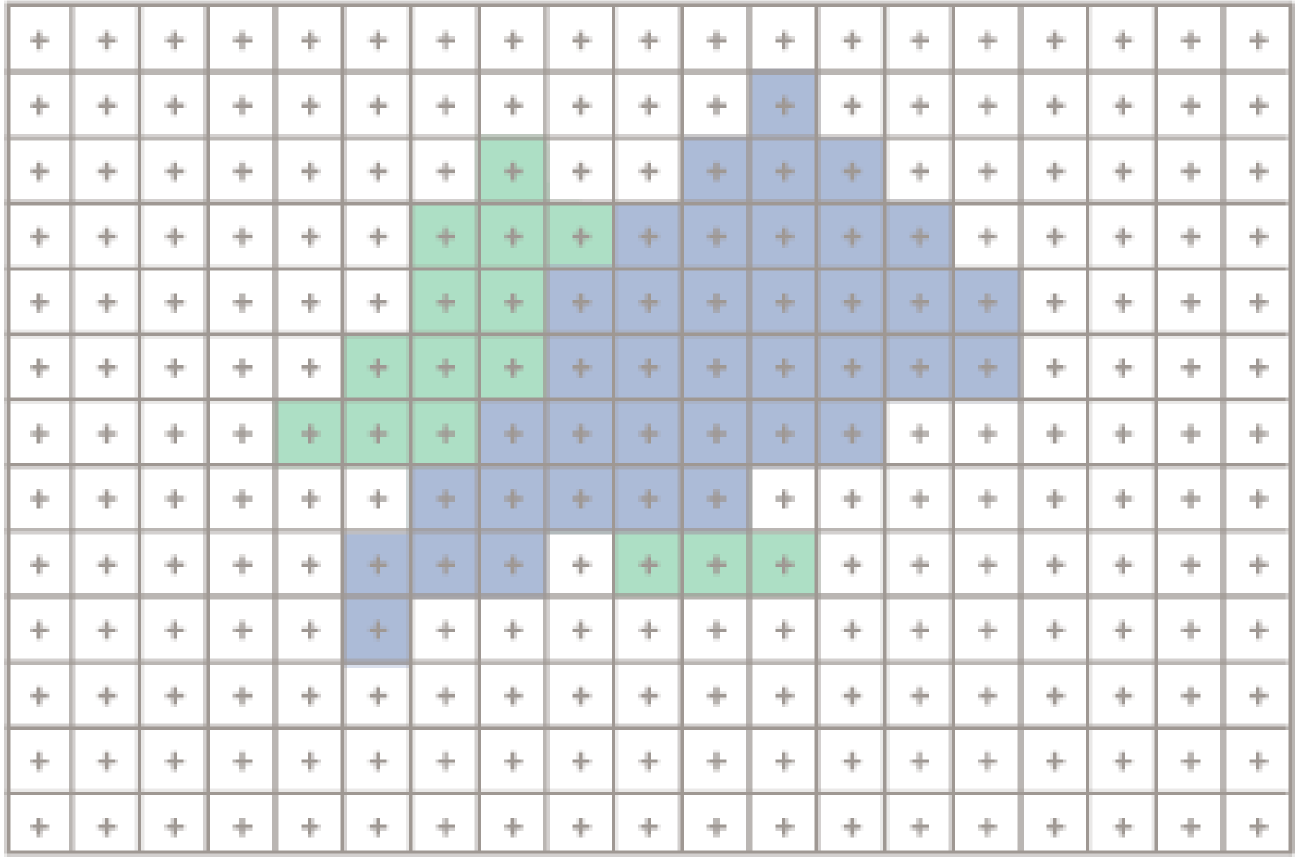


Triangle Rasterization



Given the three vertices of the triangle

Triangle Rasterization

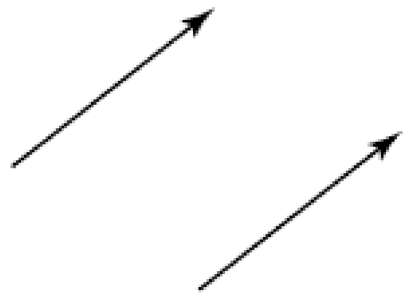


Find out which pixels belong to the triangle

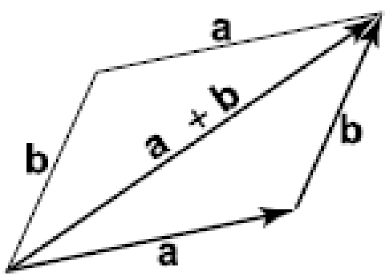
Vectors

A vector is a direction and a magnitude

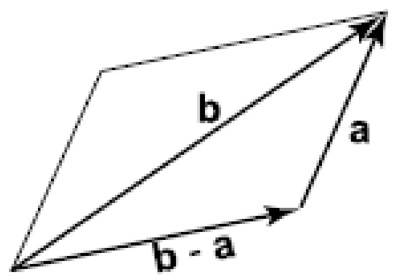
Two vectors are the same if they have the same direction and magnitude, even if they are at *different* places.



Vector Addition



Vector Subtraction

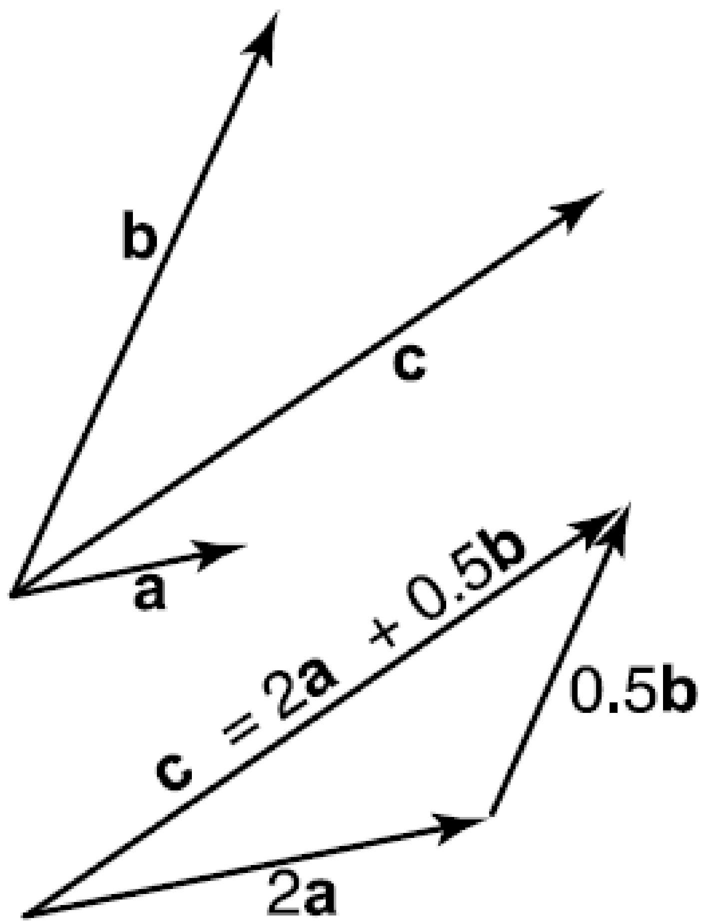


Vectors and Coordinates

A 2D vector can be written as the combination of any two non-parallel vectors. This is called linear independence.

These two vectors are called the basis vectors.

$$c = c_a a + c_b b$$



We can then represent a vector as an ordered pair of numbers:

$$c = \begin{bmatrix} c_a \\ c_b \end{bmatrix}$$

Barycentric Coordinates

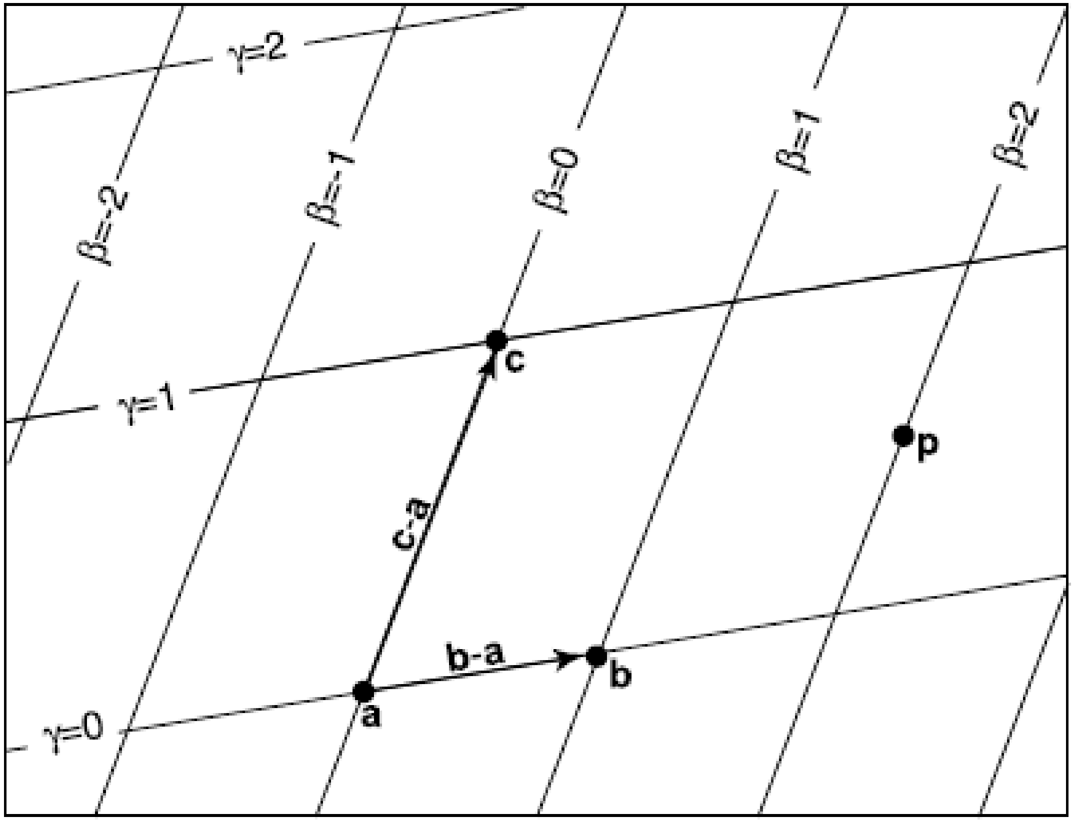
Any point p on the plane can be written as:

$$p = a + \beta(b-a) + \gamma(c-a)$$

$$p = (1-\beta-\gamma)a + \beta b + \gamma c$$

$$p(\alpha,\beta,\gamma) = \alpha a + \beta b + \gamma c$$

$$s.t. \quad \alpha + \beta + \gamma = 1$$



Color Interpolation:

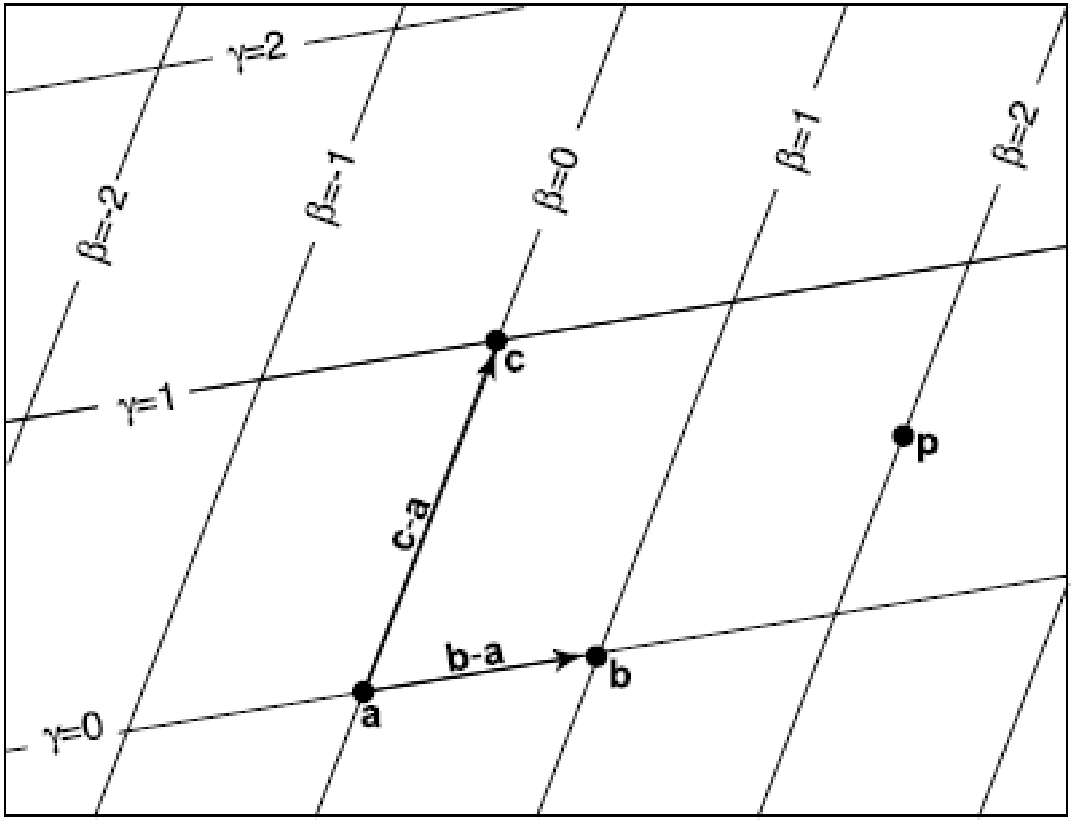
$$c_p = \alpha c_a + \beta c_b + \gamma c_c$$

Barycentric Coordinates

Inside Triangle iff:
$0 < \alpha < 1$
$0 < \beta < 1$
$0 < \gamma < 1$

On edge ab if:
$0 \leqslant \alpha \leqslant 1$
$0 \leqslant \beta \leqslant 1$
$\gamma = 0$

On vertex a if:
$\alpha = 1$
$\beta = 0$
$\gamma = 0$



$$p(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$$

$$s.t. \quad \alpha + \beta + \gamma = 1$$

Computing Barycentric Coordinates

- Algebraic Solution
  - 2 equations in 2 unknowns

$$p = a + \beta(b-a) + \gamma(c-a)$$

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \end{bmatrix} + \beta \begin{bmatrix} x_b - x_a \\ y_b - y_a \end{bmatrix} + \gamma \begin{bmatrix} x_c - x_a \\ y_c - y_a \end{bmatrix}$$

$$\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x_p - x_a \\ y_p - y_a \end{bmatrix}$$

Computing Barycentric Coordinates

- Geometric Solution

They are signed scaled distances from triangle sides:

Implicit Function: $\beta = f_{ac}(x, y)$

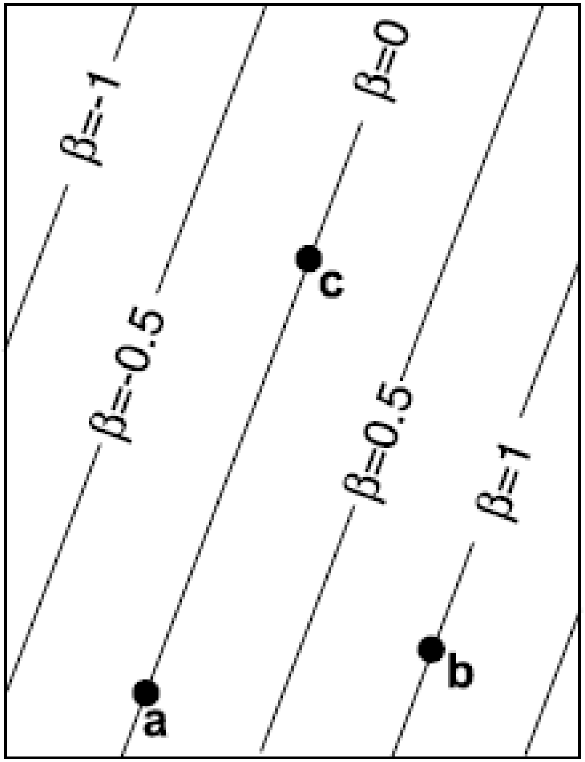
want:

$\beta = 0 \rightarrow$ on line ac

$\beta = 1 \rightarrow$ at point b

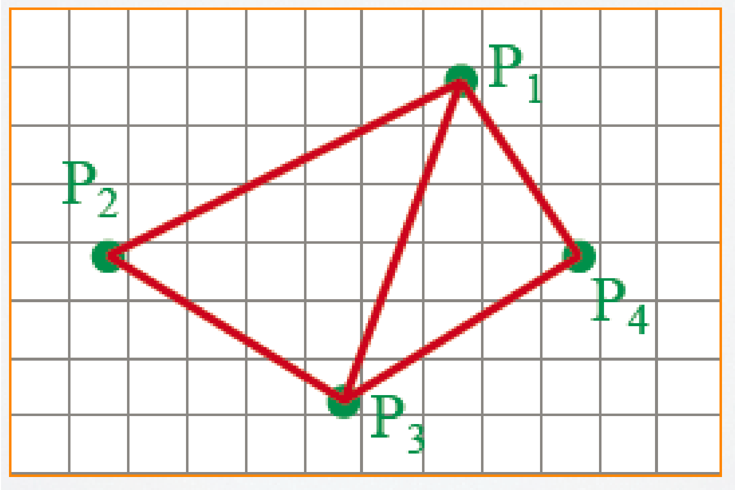
$$\beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)}$$

$$f_{ac}(x, y) = (y_a - y_c)x + (x_c - x_a)y + x_a y_c - x_c y_a$$



Triangle Rasterization

- We want it to be:
  - Fast
  - Accurate
  - No gaps
  - No order dependency



- Approach
  - Compute Barycentric Coordinates for every pixel
  - If inside triangle
    - Interpolate color
    - Draw
- What if pixels are *on* edges? Later.

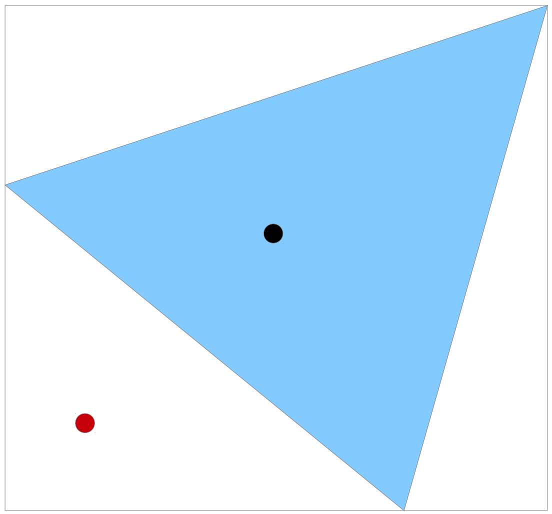
Triangle Rasterization

```
for all x do
  for all y do
    compute (alpha, beta, gamma) for (x,y)

    // Inside?
    if (alpha in (0,1) AND
        beta in (0,1) AND
        gamma in (0,1)) then
      c = alpha*c0 + beta*c1 + gamma*c2
      drawpixel(x,y) with color c
```

Optimizations?

Bounding Rectangles

Compute only for pixels likely to be inside the triangle



Every point inside the triangle *is* inside the bounding rectangle

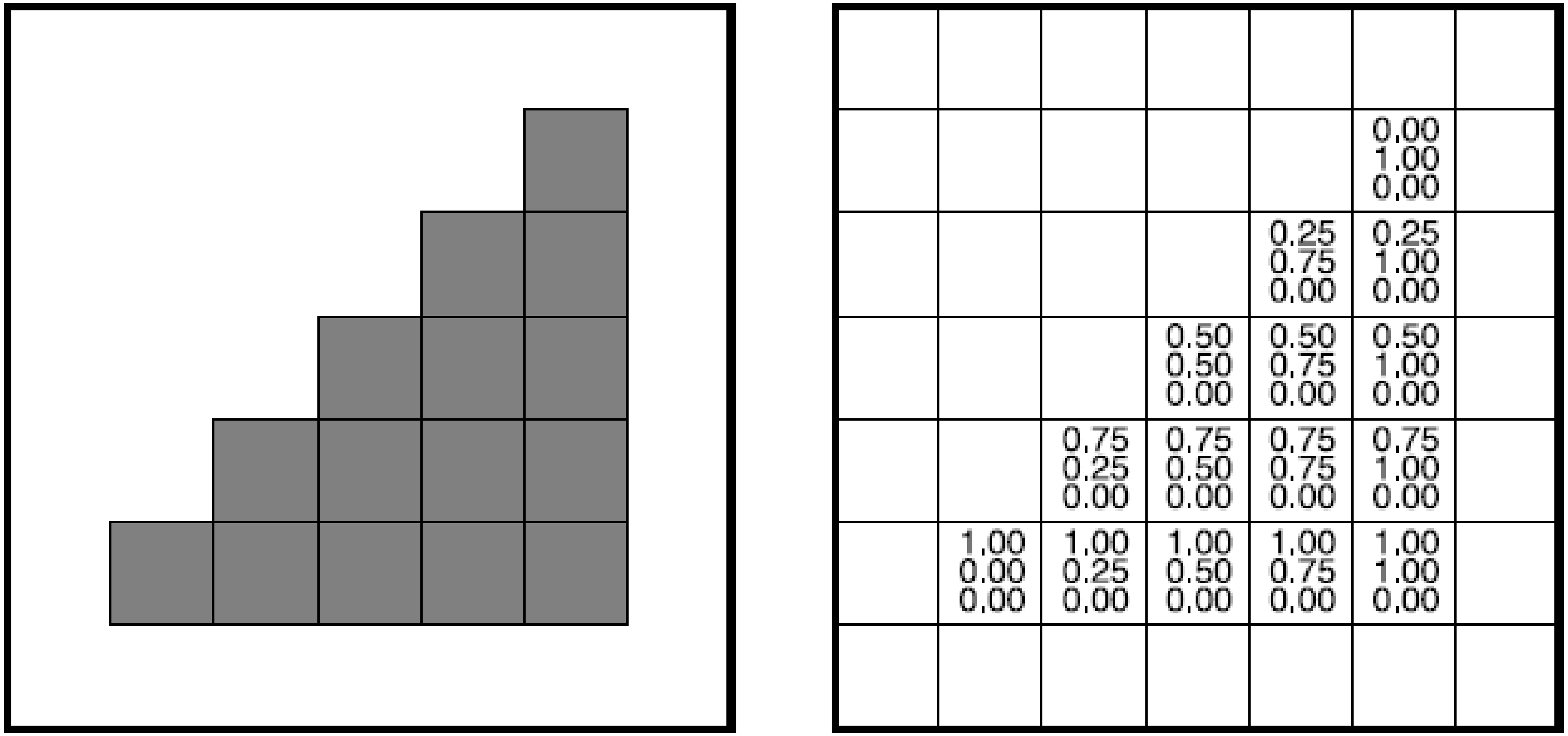
Not every point inside the bounding rectangle is inside the triangle

Triangle Rasterization

```
// Bounding rectangle
(xmin, xmax) = minmax(x0, x1, x2)
(ymin, ymax) = minmax(y0, y1, y2)

for x=xmin:xmax do
  for y=ymin:ymax do
    α = f12(x,y) / f12(x0,y0)
    β = f20(x,y) / f20(x1,y1)
    γ = f01(x,y) / f01(x2,y2)

    // Inside?
    if (α > 0 AND β>0 AND γ>0)
      c = α*c0 + β*c1 + γ*c2
      drawpixel(x,y) with color c
```

More Optimizations?

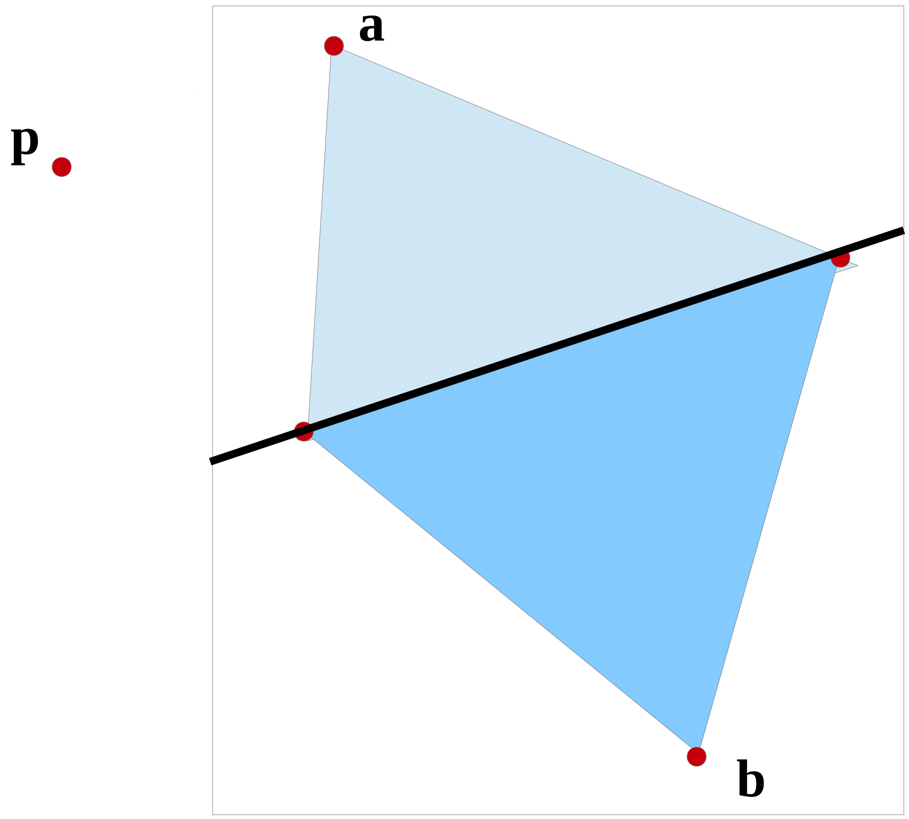
Triangle Rasterization



Triangle Rasterization

What to do with pixels on a shared edge? Which color should they get?

Pick a point *p* outside the screen e.g. (-1, -1)



If the pixel on the shared edge belongs to the triangle whose opposite vertex is on the same side as *p*, then draw it

$$f(a) \times f(p) > 0\,?$$

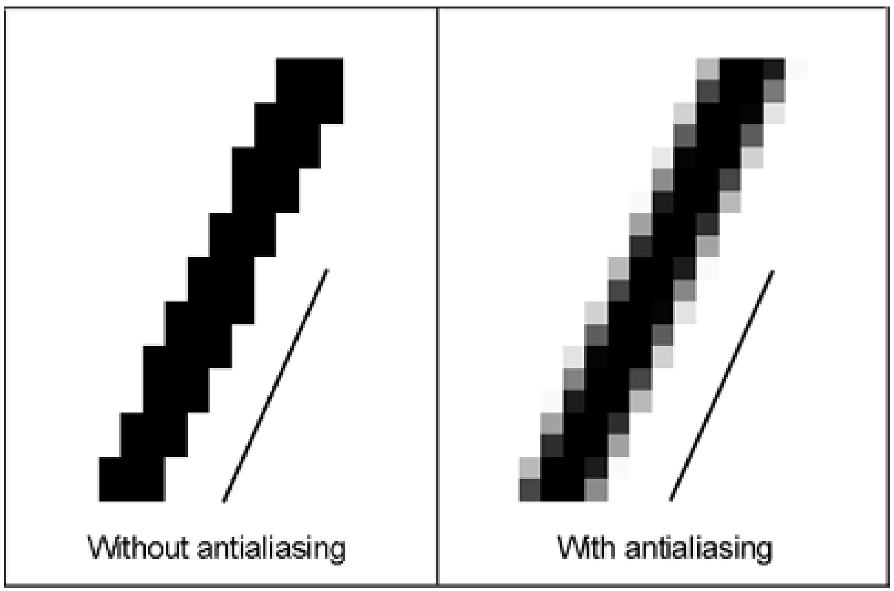
Triangle Rasterization

```
// Bounding rectangle
(xmin, xmax) = minmax(x0, x1, x2)
(ymin, ymax) = minmax(y0, y1, y2)

fα = f12(x0,y0)
fβ = f20(x1,y1)
fγ = f01(x2,y2)

for x=xmin:xmax do
  for y=ymin:ymax do
    α = f12(x,y) / fα
    β = f20(x,y) / fβ
    γ = f01(x,y) / fγ

    // Inside or on edge?
    if (α>=0 AND β>=0 AND γ>=0) then
      if (α>0 OR fα * f12(-1,-1) > 0) AND
         (β>0 OR fβ * f20(-1,-1) > 0) AND
         (γ>0 OR fγ * f01(-1,-1) > 0)) then
        c = α*c0 + β*c1 + γ*c2
        drawpixel(x,y) with color c
```

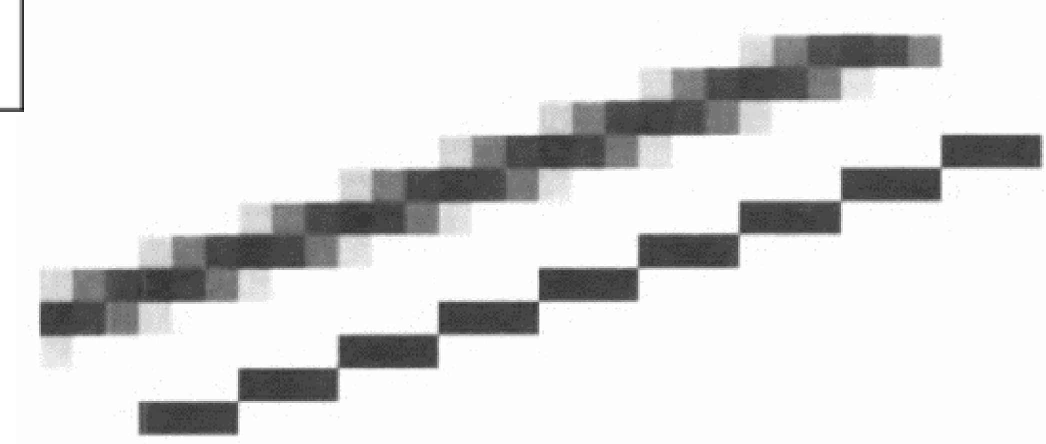
Inside the triangle or on the edge
and belongs to the right triangle!

Anti-aliasing

Technique used to reduce the artifacts of sub-sampling and make the output more visually pleasing

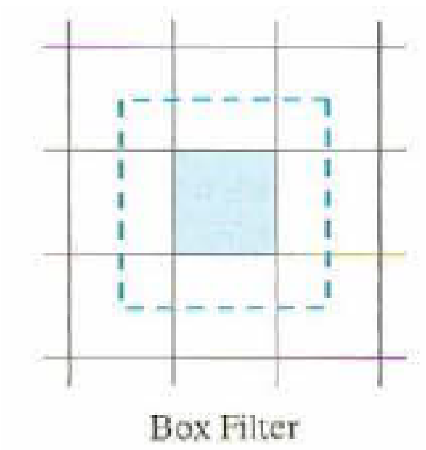


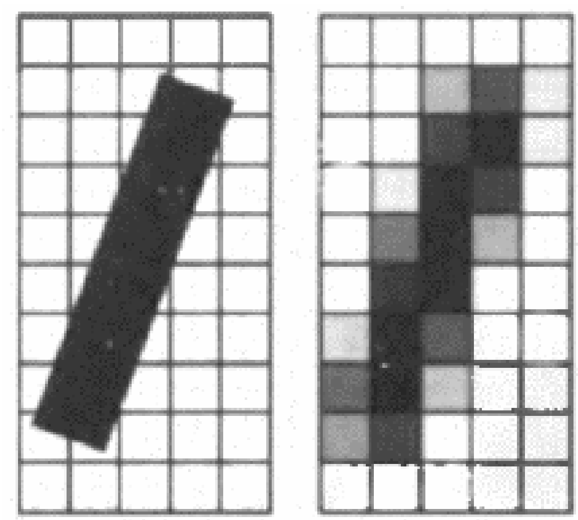
The transition from black to white is smooth



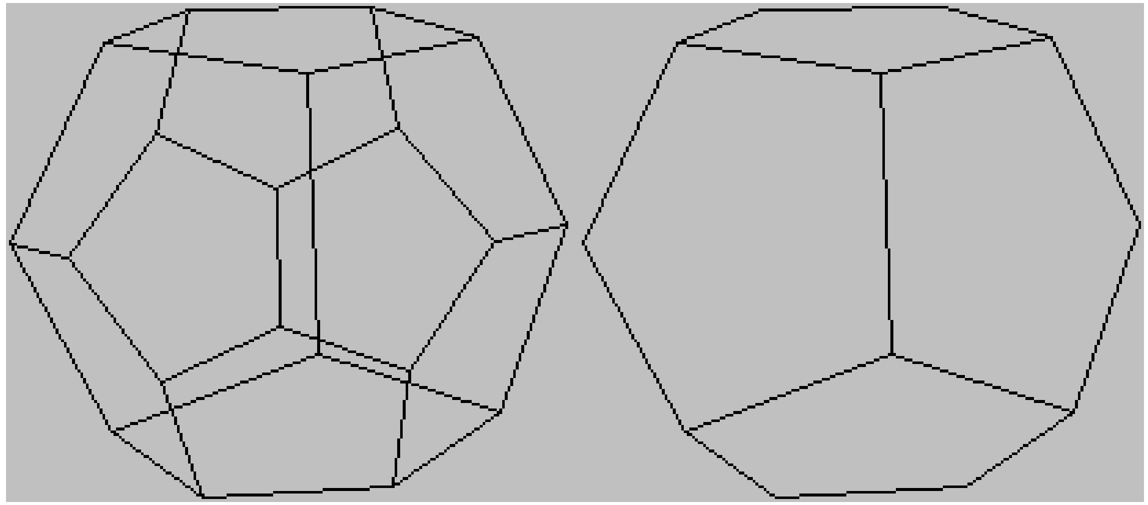
Antialiasing

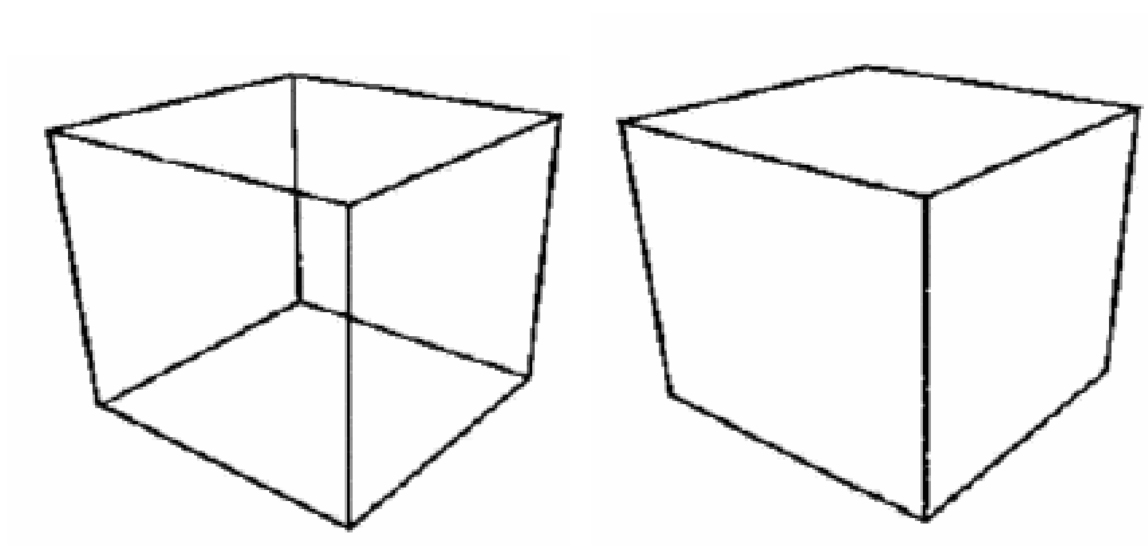
- Draw line at higher resolution
  - If we want a line on a raster of 256x256 pixels
  - Rasterize a line on a raster of 1024x1024
- “Box filter” to subsample
  - Replace every pixel with the average of its 16 neighbors





Hidden Surface Elimination

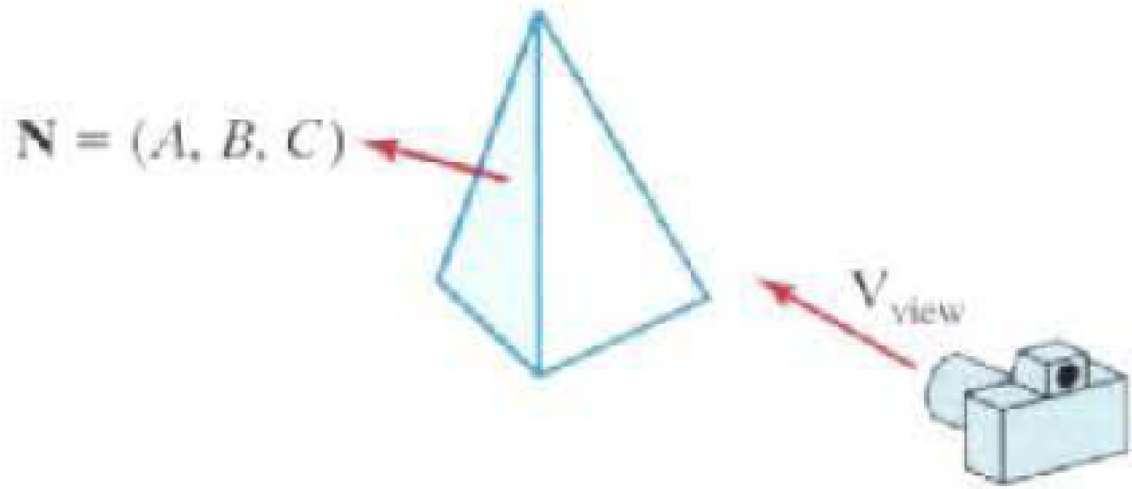




Back Face Elimination

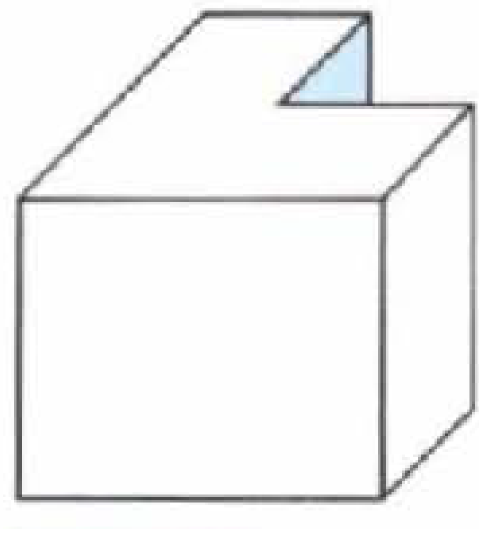
- Object Space
  - Works in the scene and compares objects
  - e.g. Back Face Culling
- Image Space
  - Works on the projected pixels
  - e.g. Z-Buffer and BSP Trees

Back Face Culling



If $\boldsymbol{N} \cdot \boldsymbol{V}_{view} > 0$ → Back Face

Is this enough?



Z-Buffer

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = M_{vp} M_{orth} P M_{cam} M_m \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

From the transformations pipeline, we get z_c

∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

Keep a z value for every pixel on the screen !

Z-Buffer

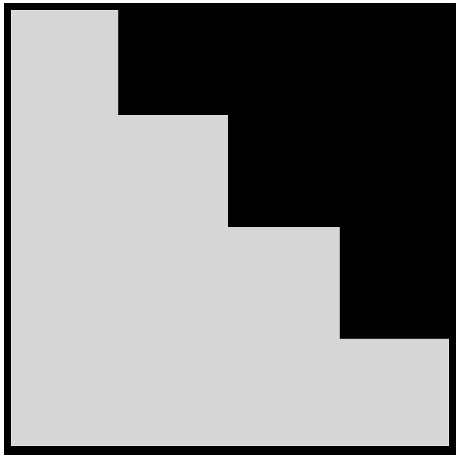
```
function SetPixel(i, j, color, z)
  if (z < z_buffer(i,j)) then
    z_buffer(i,j) = z
    screen(i,j) = color
```

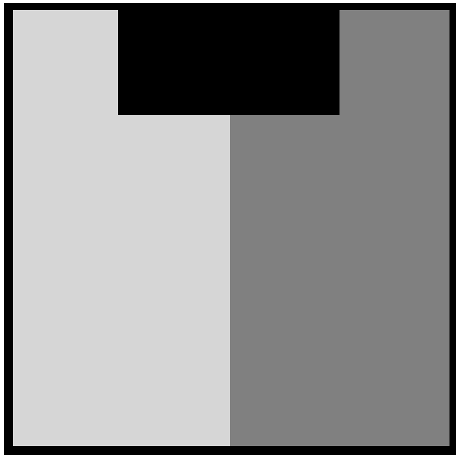
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

Z-Buffer

1	∞	∞	∞
1	3	∞	∞
1	3	5	∞
1	3	5	7

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

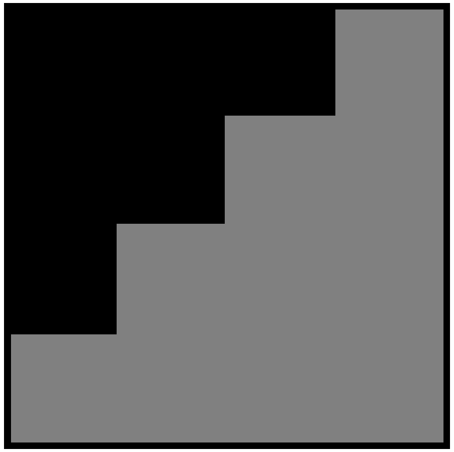


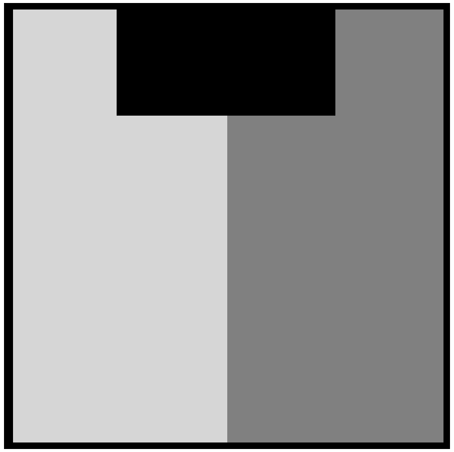


Left then right triangle

∞	∞	∞	1
∞	∞	3	1
∞	5	3	1
7	5	3	1

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1



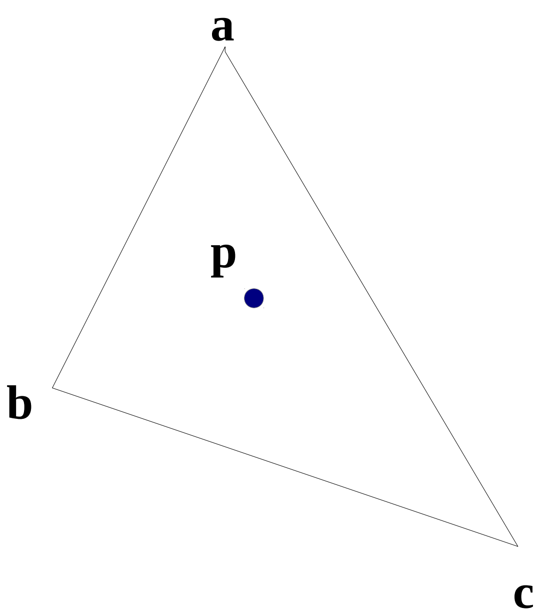


Right then left triangle

Z-Buffer independent of rendering order !

Z-Buffer

How do we get z values for pixels on a triangle ?!



Barycentric Coordinates !

$$p(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$$

$$s.t. \quad \alpha + \beta + \gamma = 1$$

Recap

- Triangle Drawing
- Anti-aliasing
- Z-Buffer