(polygons)

7 – triangles and hidden surfaces

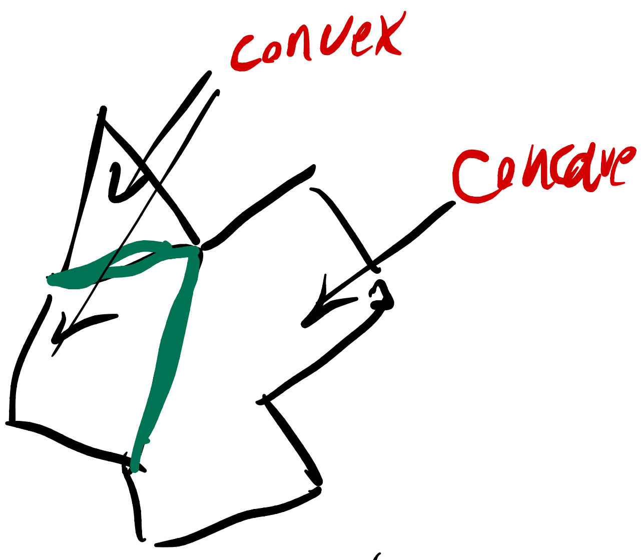
Rasterizing triangles

Hidden / visible surface algorithms

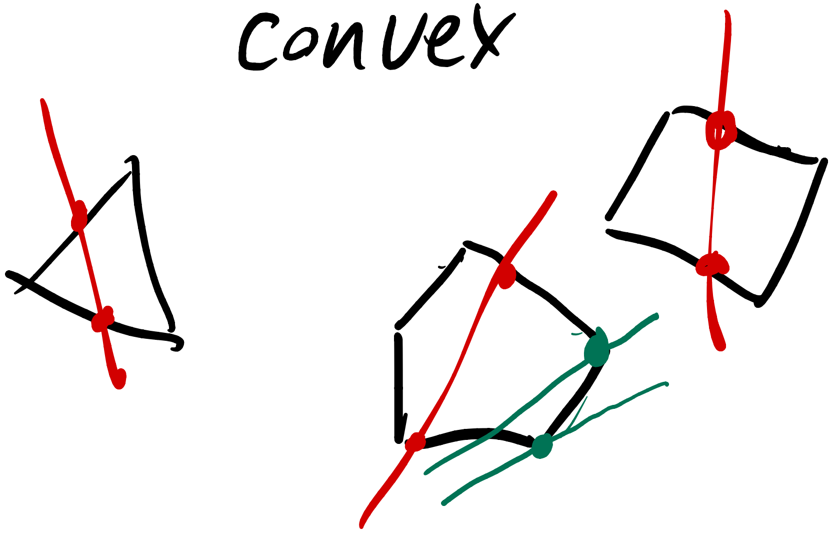
Polygons & Rasterization

Goal: no cracks between adjacents polys
no overlaps

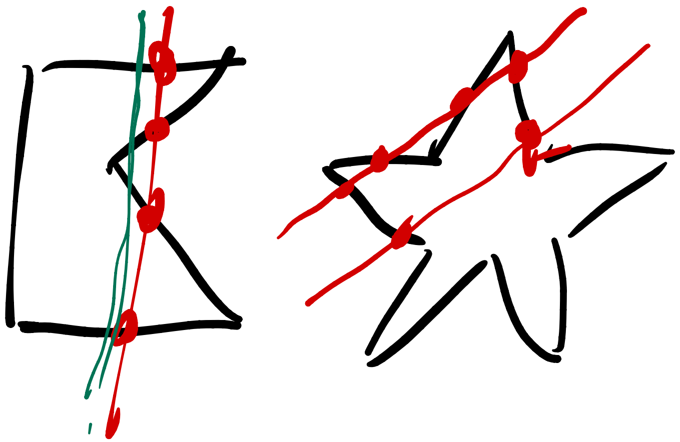
GPU's are optimized for polygons (triangles)



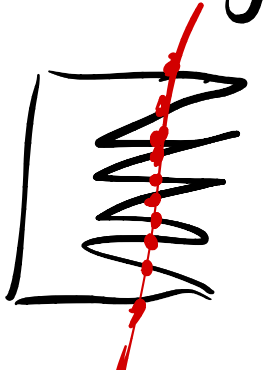
Convex



Concave



algs focus on drawing between 2 edges

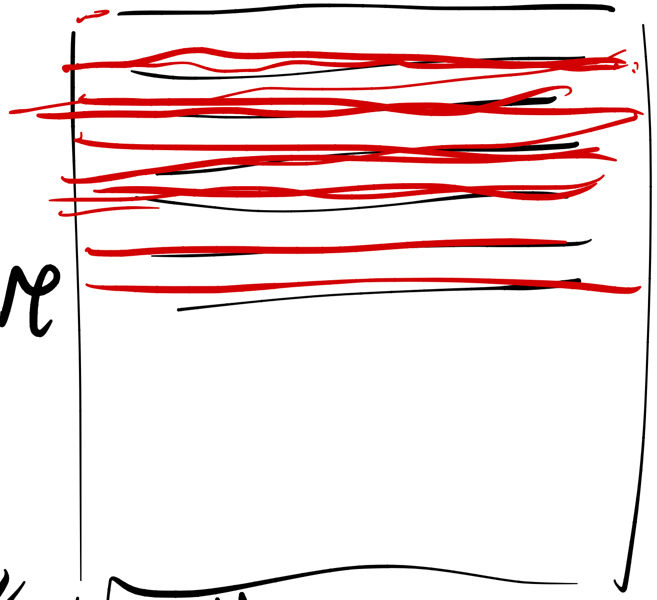


Rasterization ⇒ Scan Conversion

. fill a polygon

fill in the horizontal line

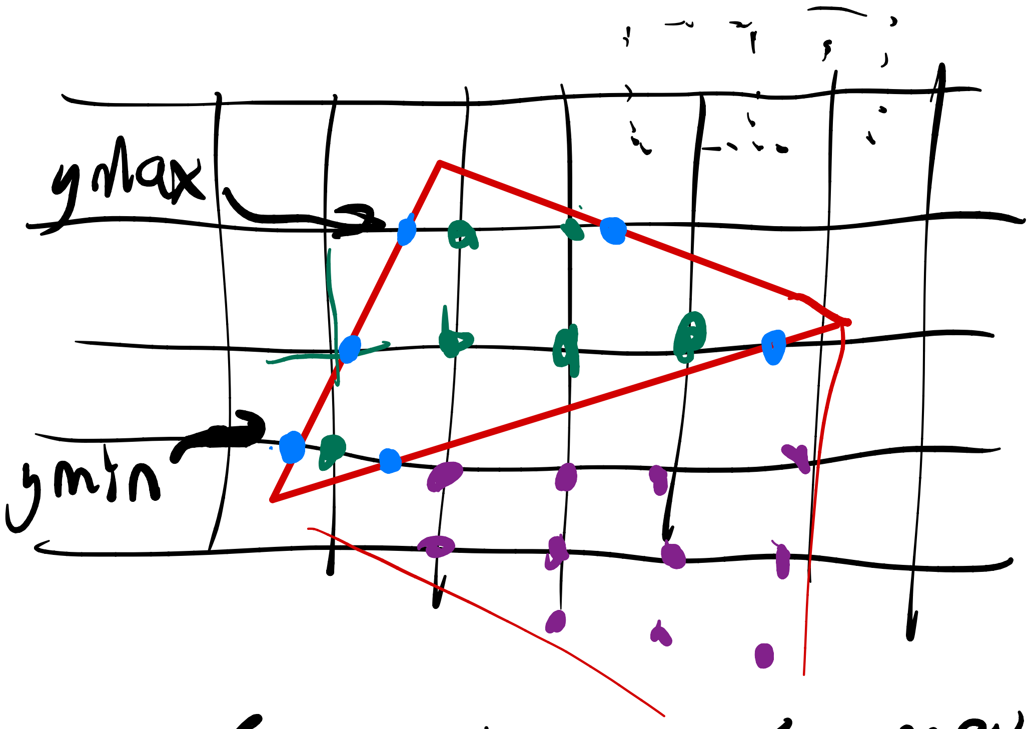
⇒ CPU's an "old" hardware

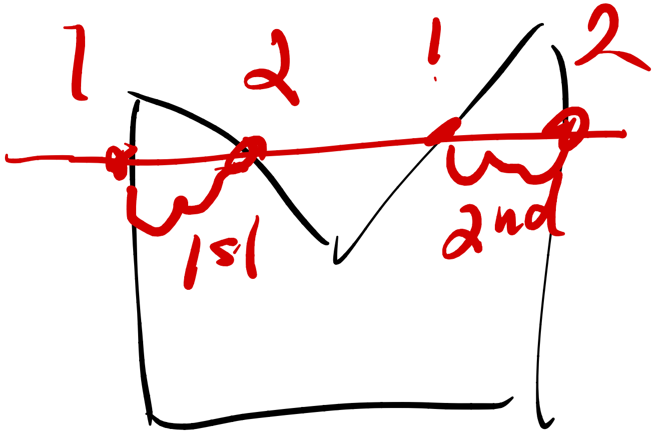


GPU's . rasterization is done by "pixel"

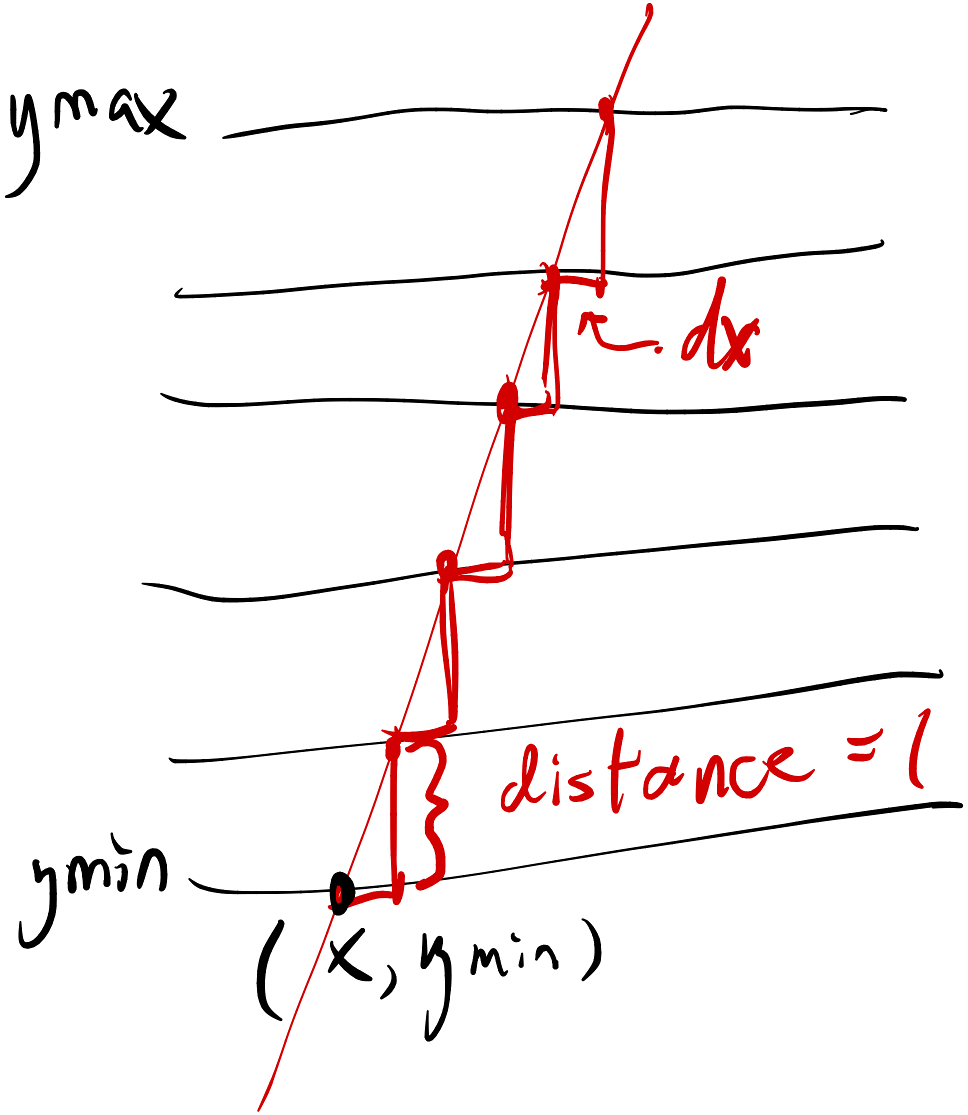
- seperately scans "current display"

polygon rast





```
for (y=ymin ; y<ymax ; y++)
    for x-intersection for scanline & edges
    sort the set of line segments by x-vales
    fill the pixels between intersections
```



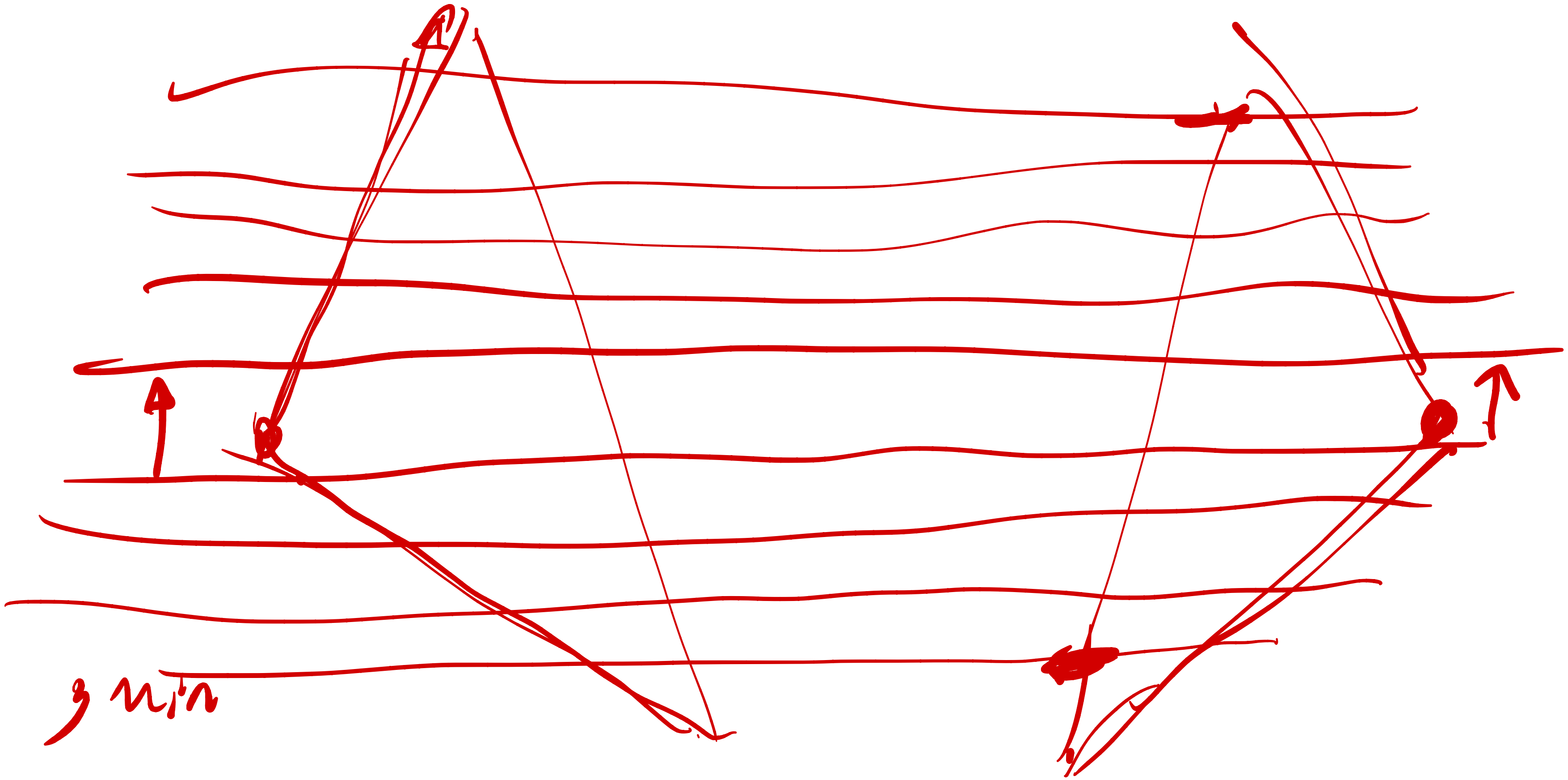
ymin (integer)

edge: { x, dx }

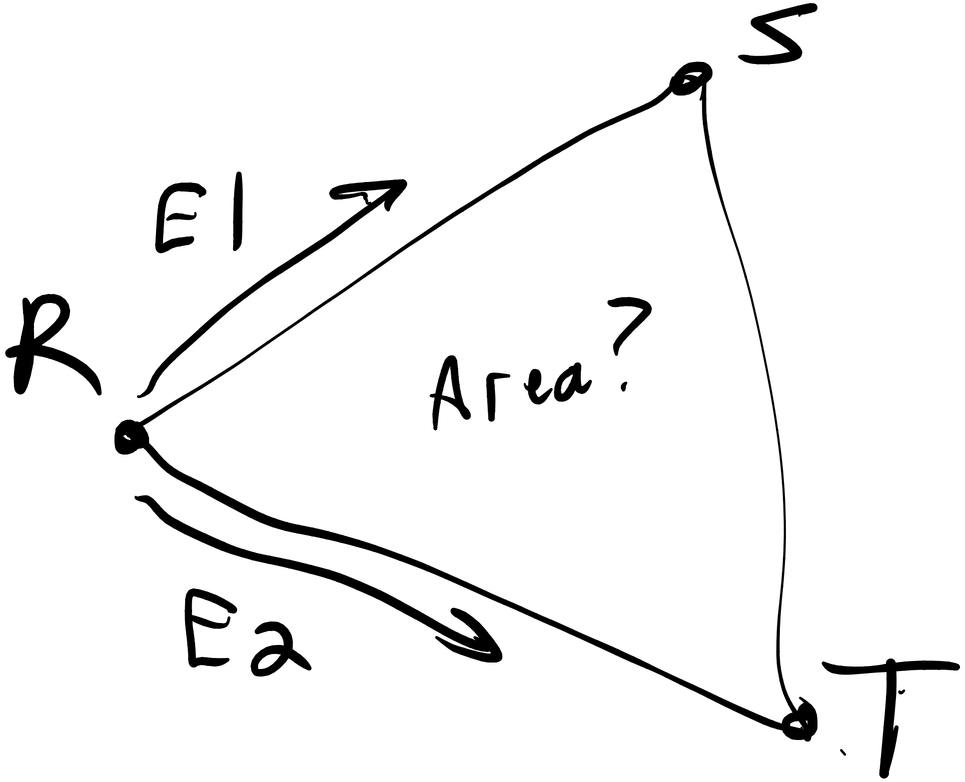
intersection of edge with y-min

```
find ymin, ymax
find xleft, xright, dxleft, dxright
for ( y = ymin , y ≤ yMax ; y++ ) {
    for ( x = ceil(xleft) ; x < xright ; x++ )
        writePixel( x, y, r, g, b )
    maybe_switch();
    xleft += dxleft
    xright += dxright
}
```

By rows
(CPU)


3 min

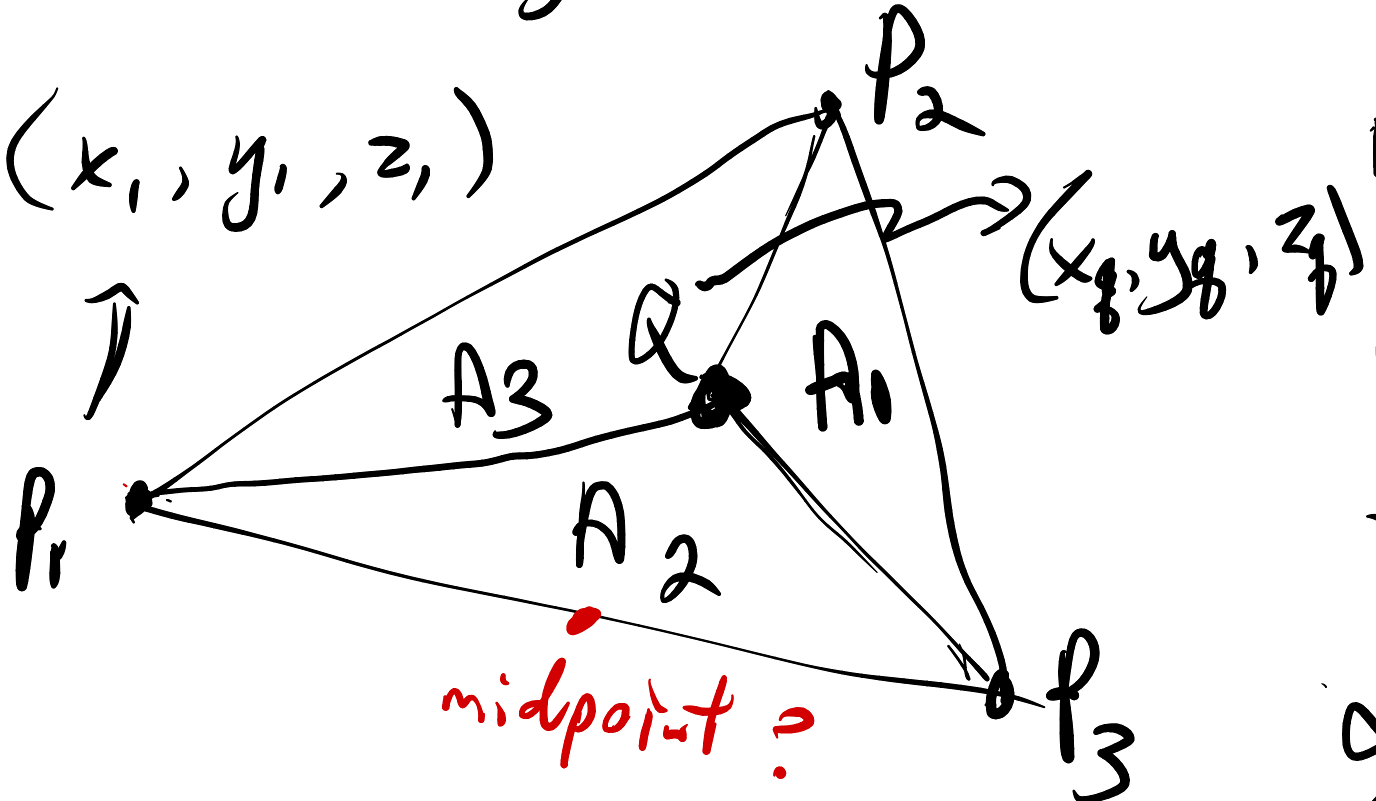
Triangle Area.



$$E1 = S - R$$
$$E2 = T - R$$

$$\text{Area}(R,S,T) = \frac{1}{2}\overbrace{\|E1 \times E2\|}^{\text{magnitude}}$$

Barycentric Coordinates



A_1 = area of sub-triangle opposite P_1

(A_2, A_3)...

$$A = A_1 + A_2 + A_3$$

$$\alpha = A_1/A$$

$$\beta = A_2/A$$

$$\gamma = A_3/A$$

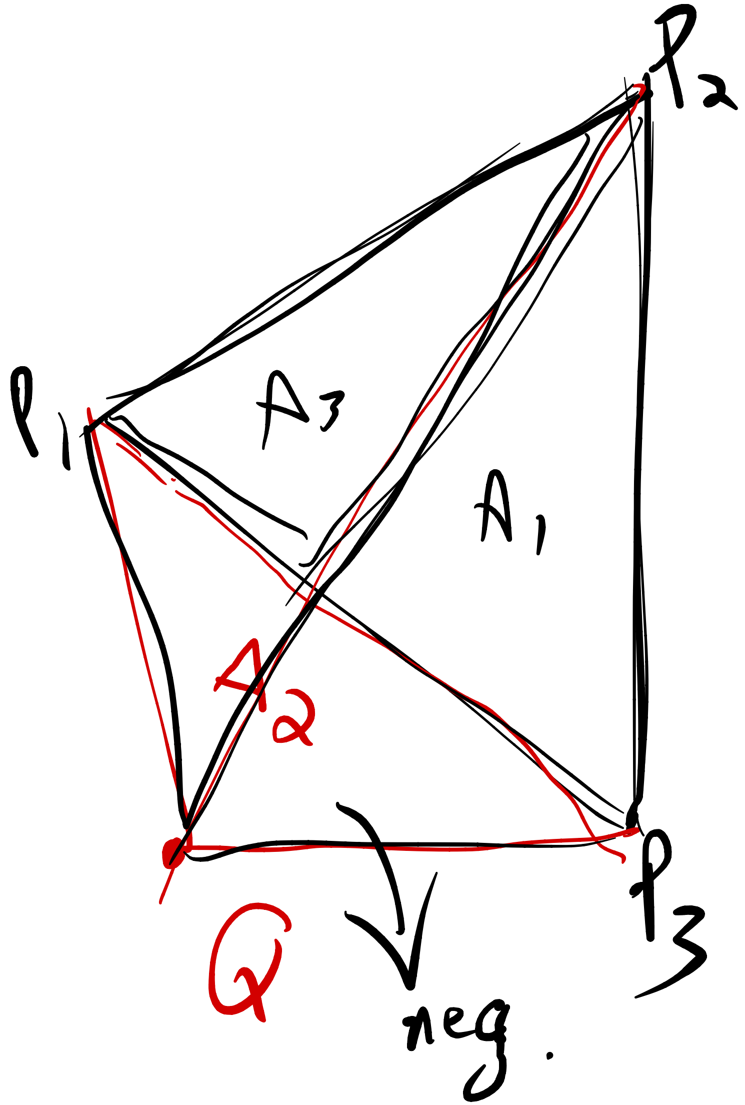
$$\alpha + \beta + \gamma = 1$$

$$Q = \alpha P1 + \beta P2 + \gamma P3$$

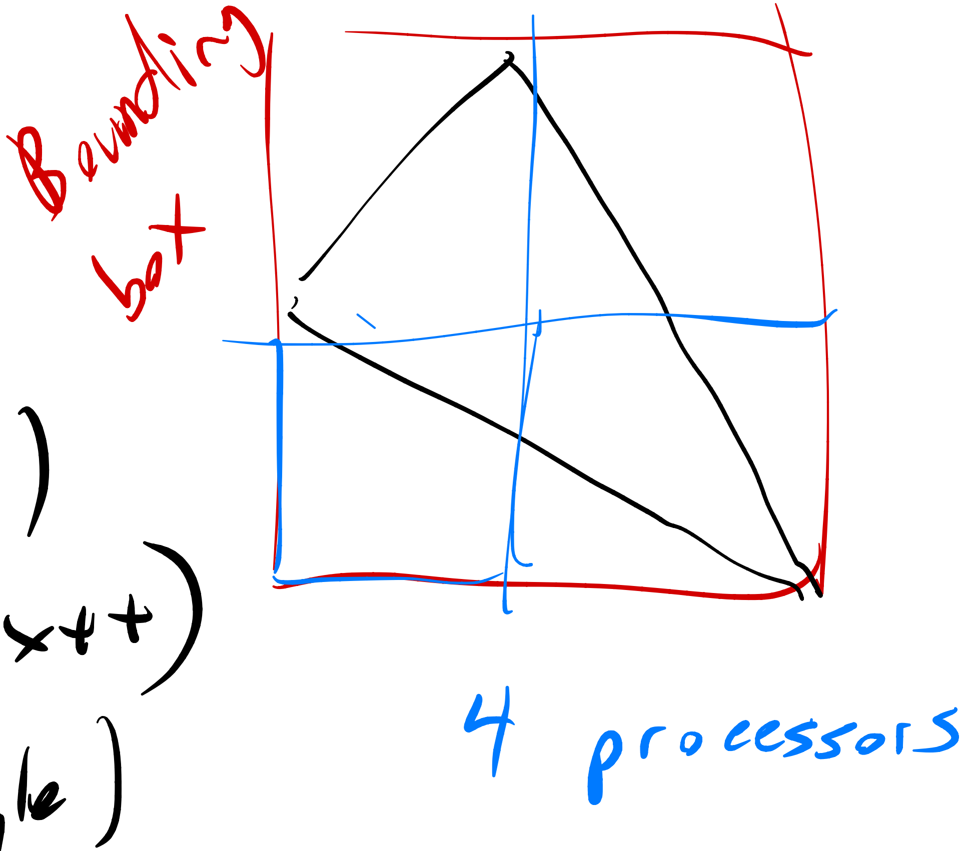
$$Q = \alpha P1 + \beta B_2 + \gamma P3$$

α, β, γ are positive inside tri

one or more negative if point is outside



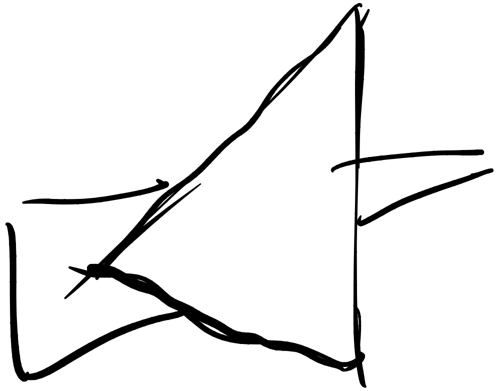
```
find xmin, ymin, xmax, ymax
for (y = ymin, y < ymax, y++)
    for (x = xmin, x < xmax, x++)
        if (x, y inside triangle)
            draw pixel()
```



GPU

Hidden Surfaces / Visible Surfaces

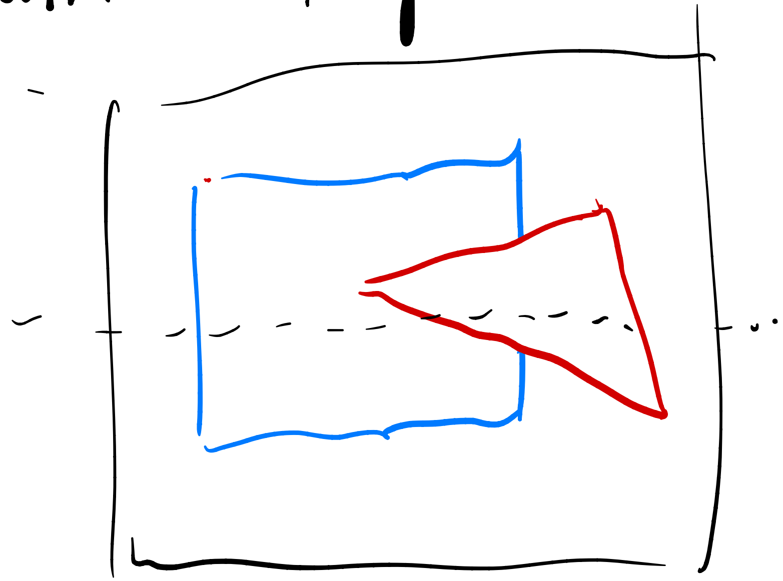
used to be a big problem

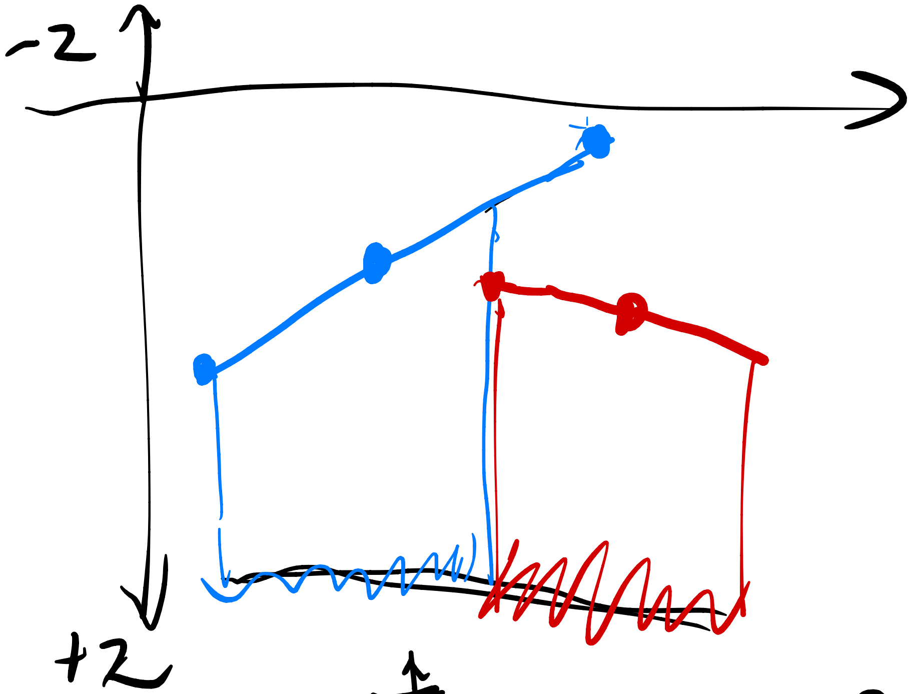


4 common Hidden Surface Algoriths

- Painter's Algorithm
- BSP - trees ← (Binary Space Partition)
- Z - buffer ← GPUs (fast)
- Ray tracing ← films, offline techniques (slower)

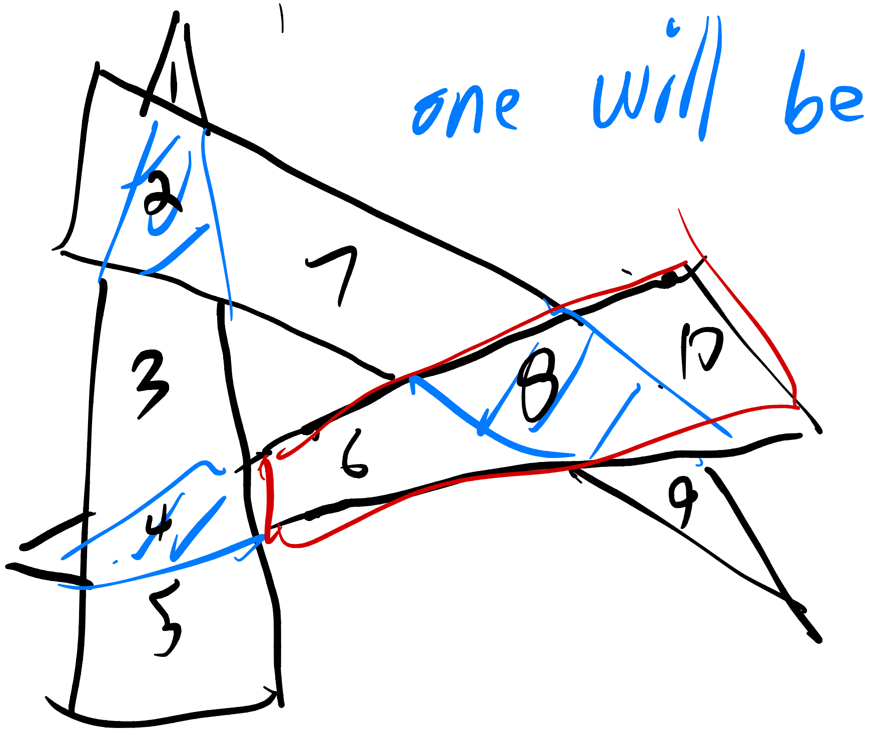
Painter's Alg





first sort polygons z. (z midpoint)

draw back to front in z



one will be wrong

BSP try to fix

Z-buffering

framebuffer with r,g,b values per pixel (0-255 / 8-bits per r,g,b)

add z ("seperate "z-buffer")

(add more data into frame buffer) · 32-bits per pixel

take rasterization alg's for lines & triangle

we had DrawPixel(x,y, r,g,b)

now → DrawPixel((x,y,z, r,g,b)

```
let pz = z-value at (x,y) in z-buffer
if (pz < passed in z value)
    WritePixel()
    Write z()
```

No matter what order I draw, the closest pixel (on some triangle) will be display.

Built into GPU's

transparency is BIG Problem

→ draw opaque objects first

→ sort trans objects (Painter)

→ draw back to front