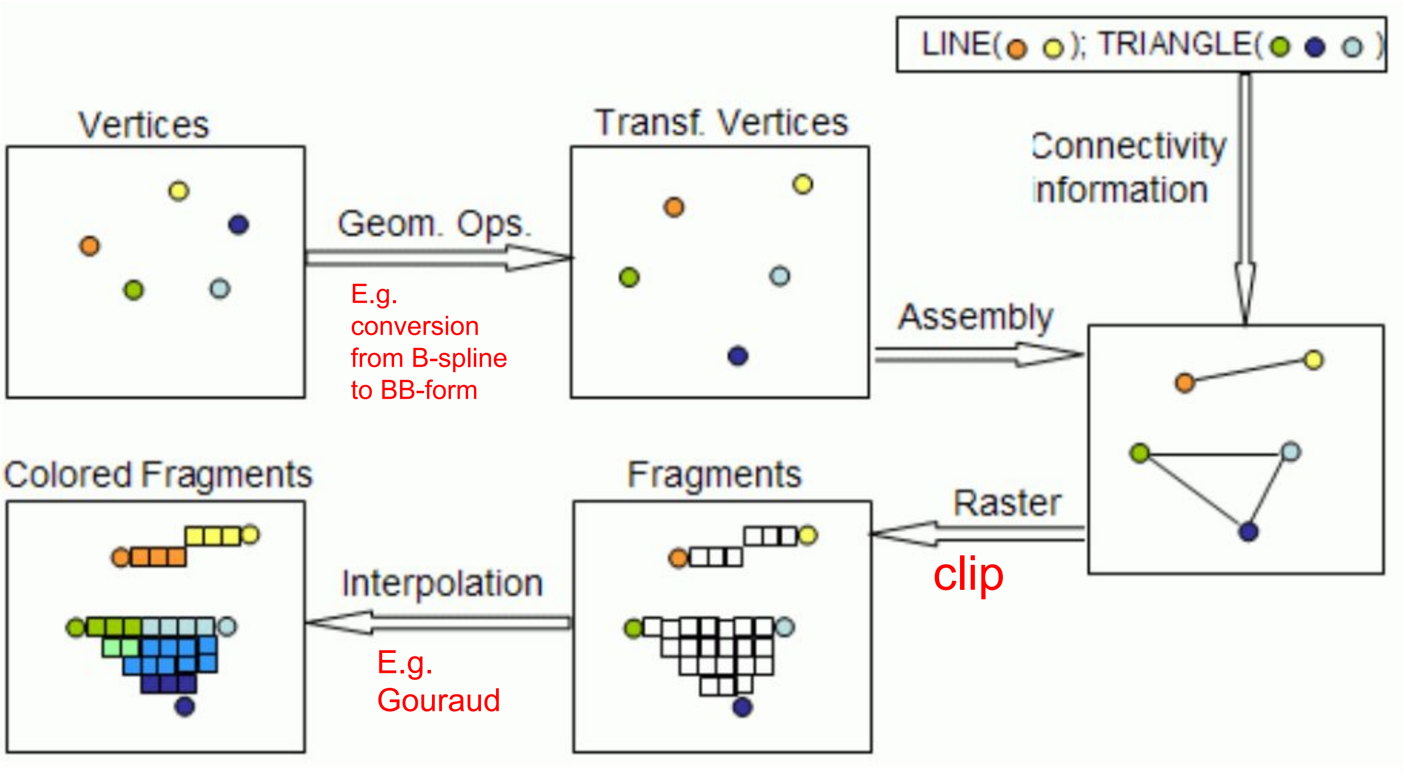
Discretization, Graphics pipeline

Discretization = Rendering to a Grid

Knowing the renderer details can

- ➢ Increase efficiency (better heuristics)
- ➢ Allow fine tuning and effects.

Discretization, Graphics pipeline

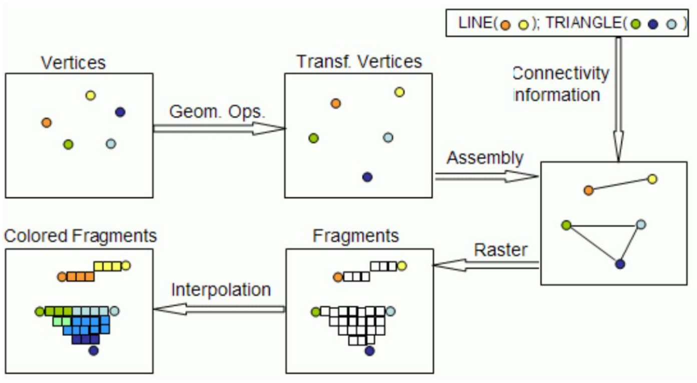


E.g. conversion from B-spline to BB-form

clip

E.g. Gouraud

Discretization, Graphics pipeline



Geometry processing: 3D, floating point discrete

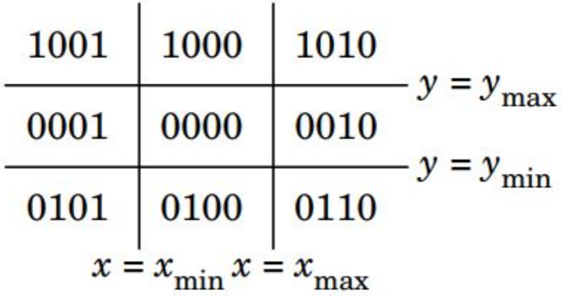
normalize, clip, hidden surface removal, shading…

Raster, scan convert: 2D, integer quantization

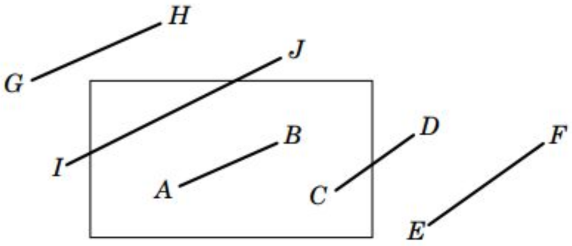
z-buffer, pixel manipulation, texturing,...

Clip

Cohen-Sutherland 2D Clipping



1001	1000	1010
0001	0000	0010
0101	0100	0110

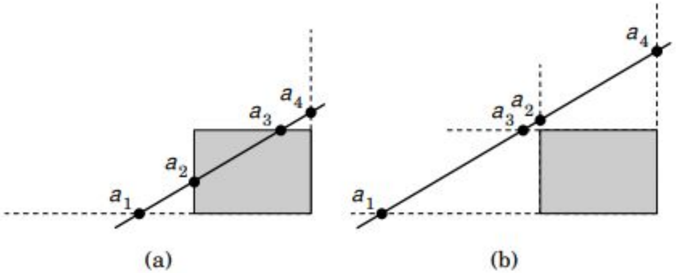


outcode o = (beyond ymax, ymin, xmax, xmin):

$o1 = o2 = 0$	take entire segment	AB
$o1 \neq 0, o2 = 0$	intersect, possibly twice	CD
$o1 \& o2 \neq 0$	discard	EF
$o1 \& o2 = 0$	intersect and test	GH, IJ

Liang-Barsky 2D Clipping

Jorg Peters



line segment $\begin{bmatrix} P_x \\ P_y \end{bmatrix}(1-t) + \begin{bmatrix} Q_x \\ Q_y \end{bmatrix} t$ intersects line $y = y_{\max}$ at t_3:

$$t_3(Q_y - P_y) = y_{\max} - P_y.$$

Can order intersection ts without floating point division:

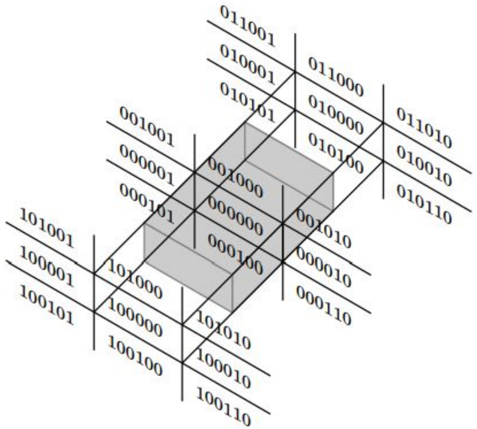
$$t_3(Q_x - P_x)(Q_y - P_y) = (Q_x - P_x)(y_{\max} - P_y),$$
$$t_2(Q_x - P_x)(Q_y - P_y) = (Q_y - P_y)(x_{\min} - P_x),$$

etc.

Clip

3D Clipping

Cohen-Sutherland, outcode

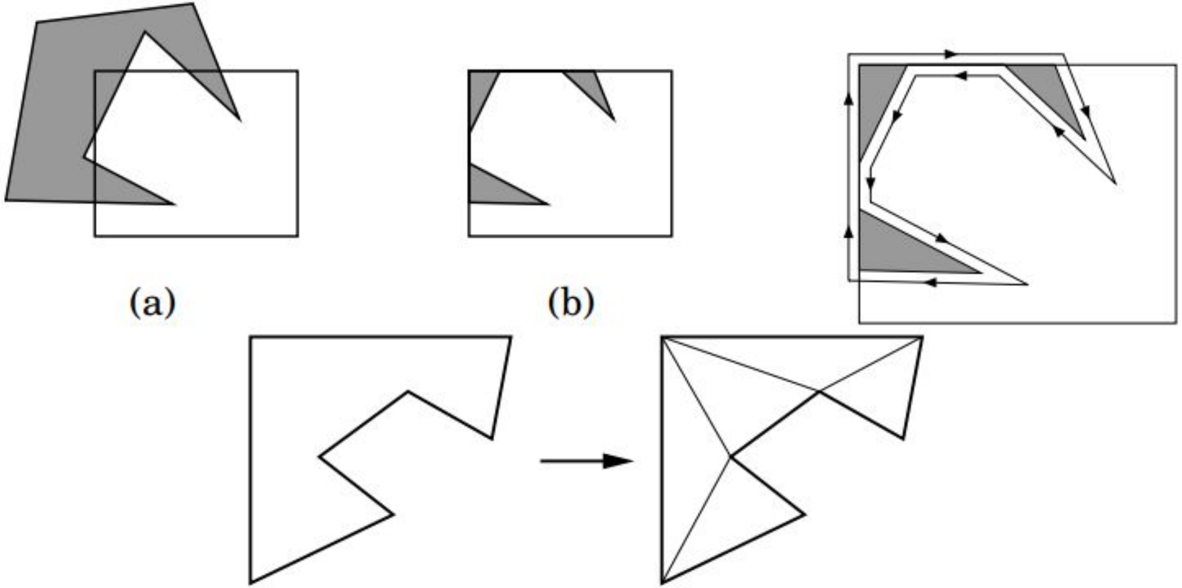


Liang-Barsky tests against a plane with normal N:

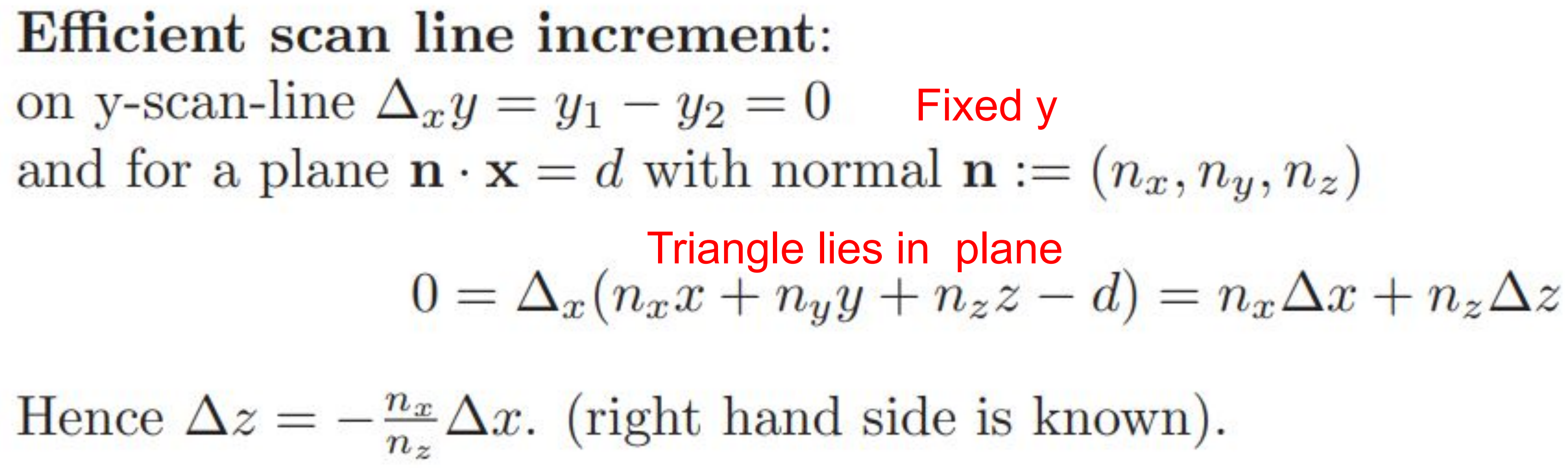
$$(P(1-t)+Qt-V)\cdot N=0$$

Why only triangles, not polygons?

Polygon Clipping



Discretization, Graphics pipeline



Efficient scan line increment:

on y-scan-line $\Delta_x y = y_1 - y_2 = 0$

Fixed y

and for a plane $\mathbf{n} \cdot \mathbf{x} = d$ with normal $\mathbf{n} := (n_x, n_y, n_z)$

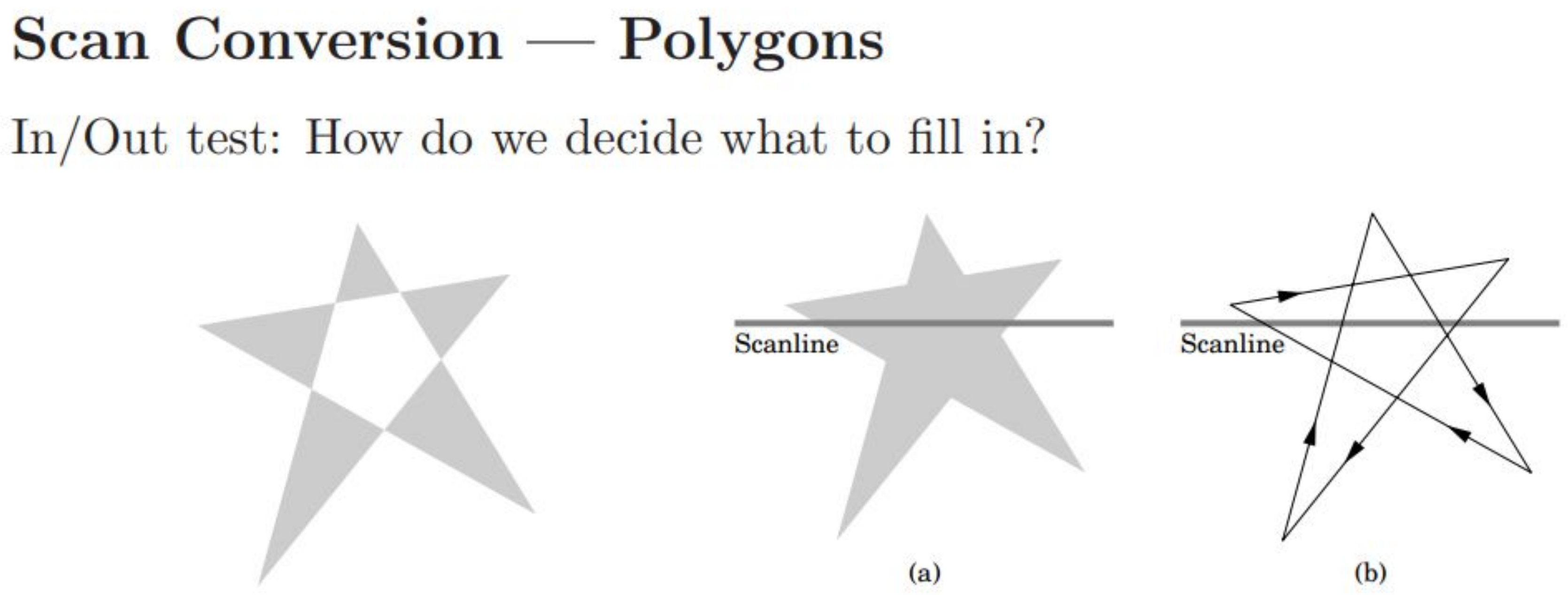
Triangle lies in plane

$$0 = \Delta_x (n_x x + n_y y + n_z z - d) = n_x \Delta x + n_z \Delta z$$

Hence $\Delta z = -\frac{n_x}{n_z} \Delta x$. (right hand side is known).

z increment if move to the right by x

To fill or not to fill ?



Scan Conversion — Polygons

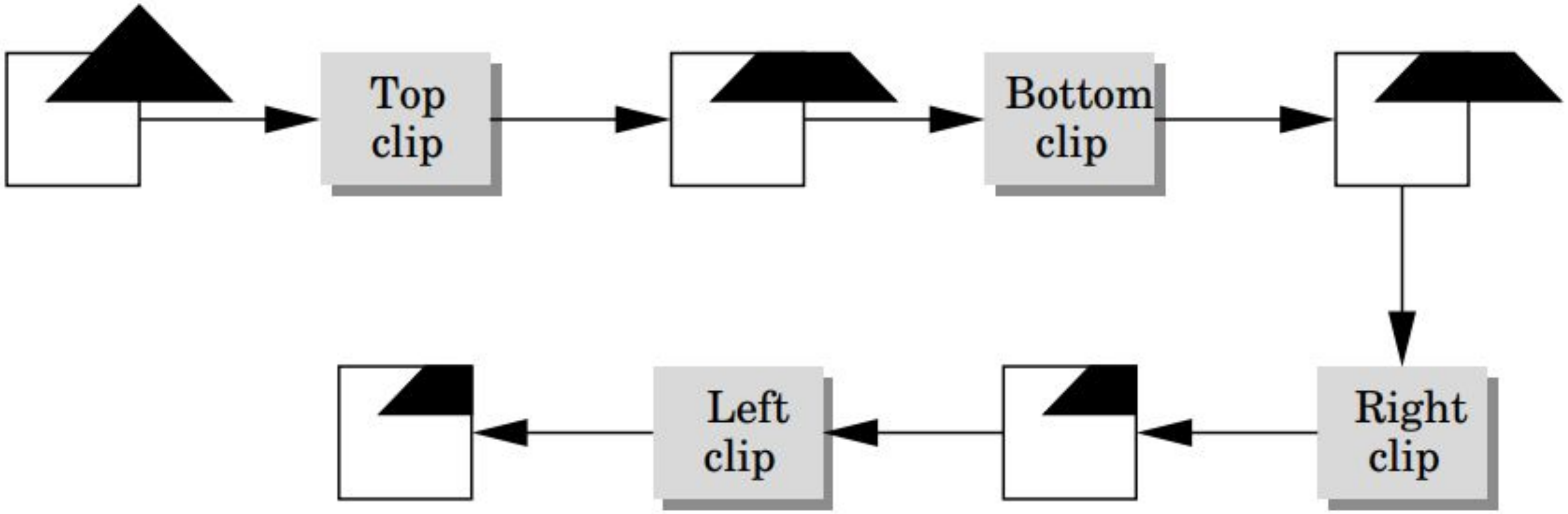
In/Out test: How do we decide what to fill in?

(a)

(b)

Clipping after mapping

Sutherland-Hodgson pipeline clipping

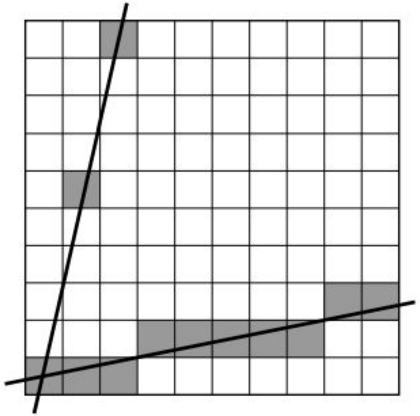


Drawing lines

Digital Differential Analyzer

Assumption: $0 < \text{slope} = \frac{\Delta y}{\Delta x} < 1$ (get other 7 cases by symmetry)

```
for (ix = x_start; ix < x_end; ix = ix+1)
    y = y + slope;
    write_pix(x, round(y), color);
```



Drawing lines

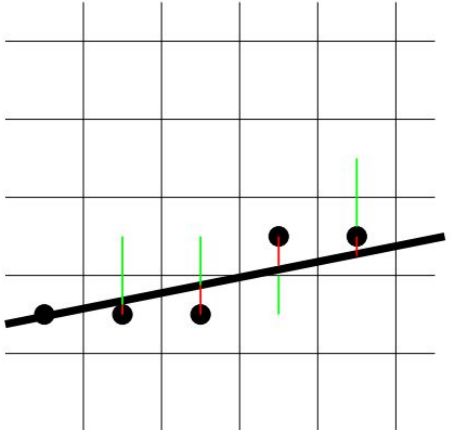
Bresenham's Algorithm

Slope expressed as quotient of integers: $\Delta y / \Delta x$

d_k := *(dist to upper candidate pixel) - (dist to lower candidate pixel)*

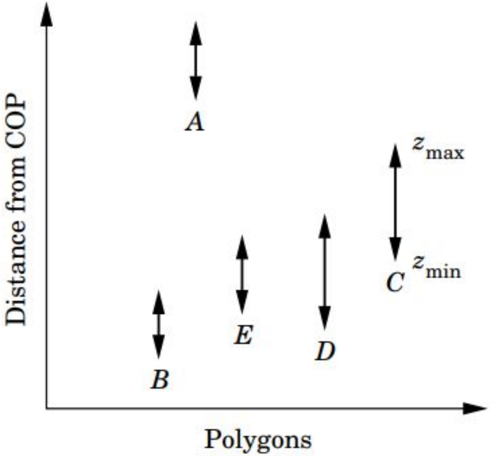
$$d_{k+1} = d_k - 2 \begin{cases} \Delta y & \text{if } d_k > 0 \quad \text{prev right} \\ \Delta y - \Delta x & \text{if } d_k \leq 0 \quad \text{prev right and up} \end{cases}$$

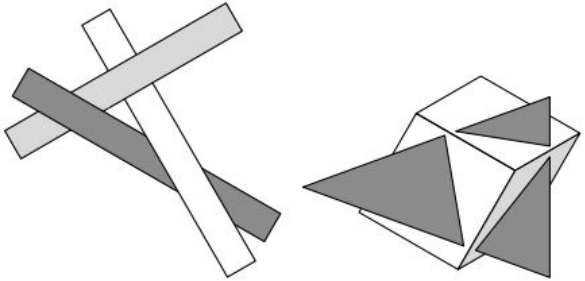
(if we move Δx to the right we gain Δy; if we move up we subtract pixel width).



Depth Sort

Depth Sort (painter's algorithm)
If z of polygon is larger than all others', paint;
if zs overlap but x or y do not, paint;
else (cycle or piercing, see below) divide (and conquer).





Buffers

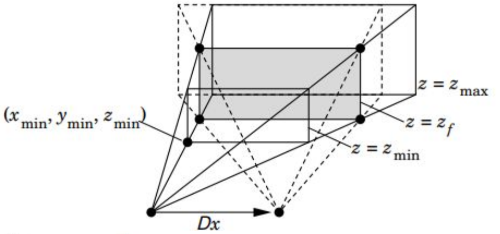
Buffers :

- ➢ color (rgba)
- ➢ depth (z)
- ➢ accumulation
- ➢ Stencil
- ➢ feedback

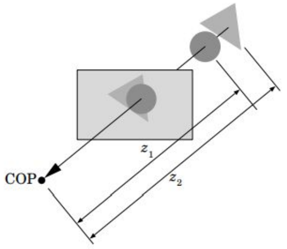
Buffer exchange (ingenious)
S = S xor M
M = S xor M
S = S xor M

jittering

motion blur



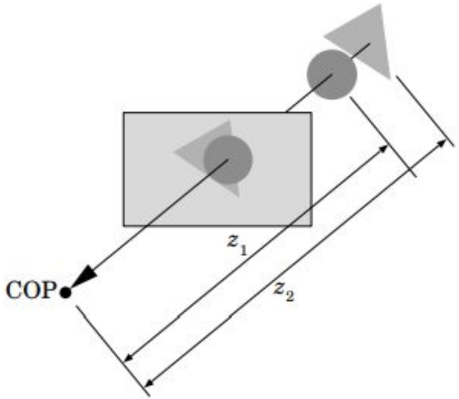
z-buffer contains closest object so far:
If z-buffer value is less than new vertex z-value do not render new



z-buffer

Hidden-surface removal

z-buffer contains closest object so far:
If z-buffer value is less than new vertex z-value do not render new



Digital filtering

Aliasing

$$\text{averaging} \begin{bmatrix} . & 1 & . \\ 1 & 1 & 1 \\ . & 1 & . \end{bmatrix} \text{ or}$$

$$\text{differencing} \begin{bmatrix} . & -1 & . \\ -1 & 4 & -1 \\ . & -1 & . \end{bmatrix}$$