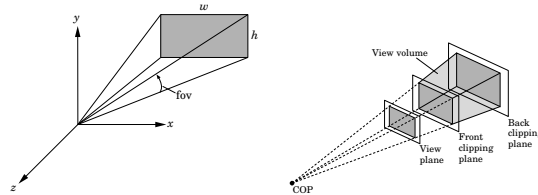


5. Projection and Clipping

Projections – Perspective

glPerspective(fovy, aspect, near, far)

Modelview → Projection → perspective division.

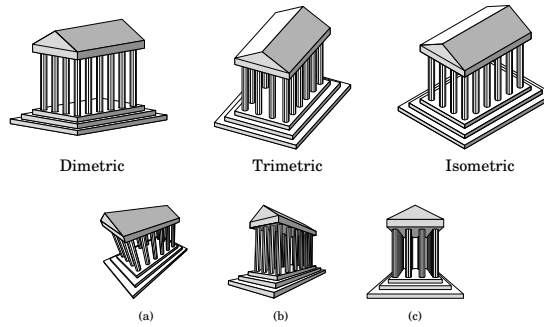


Nonuniform shortening: $y_p = d \frac{y}{z}$ where d is the distance of the view-plane along the $-z$ axis.

$$P_z := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \equiv \begin{bmatrix} dx/z \\ dy/z \\ d \\ 1 \end{bmatrix}$$

parallel projection

3-, 2-, and 1-point perspective (=number of vanishing points).



PseudoPerspective (advanced)

The decomposition of the original point \mathbf{p} into *projection* + *depth* (see earlier Figure)

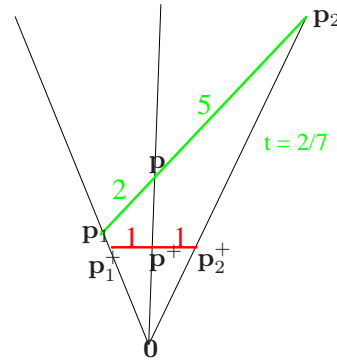
$$\begin{aligned} \mathbf{p}^\perp &:= \mathbf{p}^+ + \text{depth}(\mathbf{p})\mathbf{n} \\ &= P\mathbf{p} + (\mathbf{p} \cdot \hat{\mathbf{n}})\begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix} = (P + \text{outerprod}(\begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix}, \hat{\mathbf{n}}))\mathbf{p}. \end{aligned}$$

The depth is needed depth for hidden surface removal, Gouraud shading, etc.

PseudoPerspective + scan line (advanced)

Recovering ratios in \mathbf{R}^3 from ratios of the projection and the depth.

Linear averaging of intensity \mathbf{i} at \mathbf{p} between \mathbf{i}_1 at \mathbf{p}_1 and \mathbf{i}_2 at \mathbf{p}_2 should be computed from \mathbf{p}_j not \mathbf{p}_j^\perp since *projection does not preserve ratios!* Want (green) ratio 2:5, not (red) 1:1.



$$\mathbf{i} := (1 - t)\mathbf{i}_1 + t\mathbf{i}_2,$$

$$t := \frac{|\mathbf{p} - \mathbf{p}_1|}{|\mathbf{p}_2 - \mathbf{p}_1|} \neq \frac{|\mathbf{p}^+ - \mathbf{p}_1^+|}{|\mathbf{p}_2^+ - \mathbf{p}_1^+|} =: t^+.$$

How to recover t after pseudoperspective, i.e. after $\mathbf{p}_1, \mathbf{p}_2$ have been replaced by \mathbf{p}_1^+ and \mathbf{p}_2^+ ?

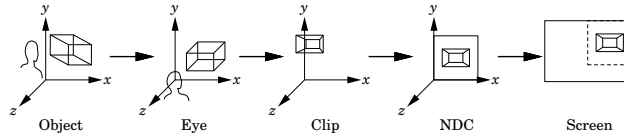
By Grassmann addition

$$\mathbf{p}^+ = \frac{(1 - t)w_1\mathbf{p}_1^+ + tw_2\mathbf{p}_2^+}{(1 - t)w_1 + tw_2} \text{ where } t = \frac{t^+w_1}{(1 - t^+)w_2 + t^+w_1}.$$

However, t is expensive to compute! Compute $t^+ = \frac{tw_2}{(1-t)w_1+tw_2}$ incrementally. Then use Blinn's method:

$$\mathbf{i} = \frac{(1 - t^+)\begin{bmatrix} \mathbf{i}_1 \\ w_1 \end{bmatrix} + t^+\begin{bmatrix} \mathbf{i}_2 \\ w_2 \end{bmatrix}}{(1 - t^+)\begin{bmatrix} 1 \\ w_1 \end{bmatrix} + t^+\begin{bmatrix} 1 \\ w_2 \end{bmatrix}} = (1 - t)\mathbf{i}_1 + t\mathbf{i}_2.$$

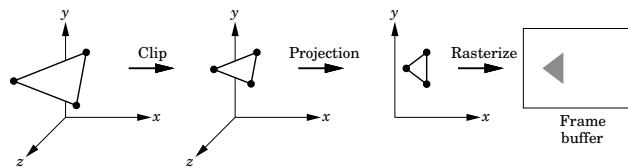
Transformations and coordinates



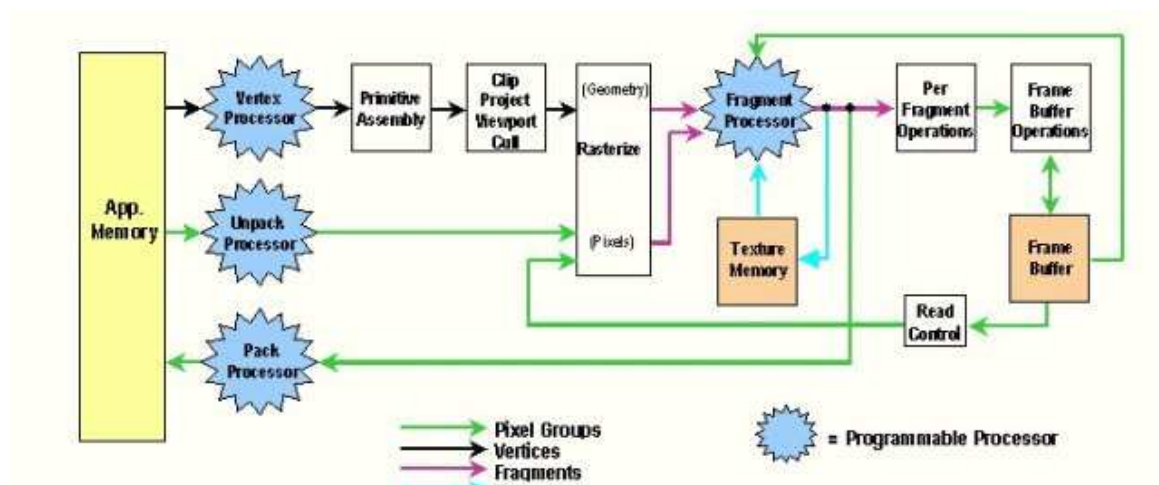
Coordinates:

- object (world)
- eye (camera)
- clip (2-unit cube)
- normalized device (3D after perspective division)
- screen (after viewport transformation)

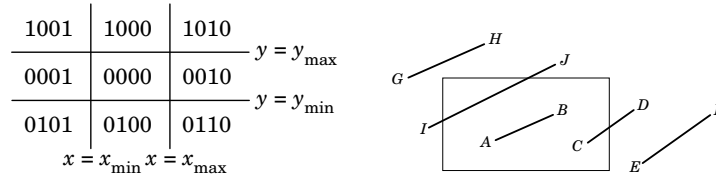
Graphics pipeline



- geometry processing *3D*, *floating point*: normalization, clipping, hidden-surface removal, shading.
- rasterization, scan conversion *2D*, *integer* pixel manipulation, quantization.



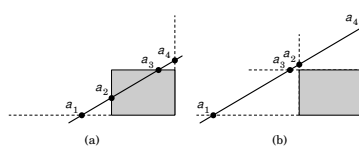
Cohen-Sutherland 2D Clipping



outcode $o = (\text{beyond } y_{\max}, \text{ymin}, x_{\max}, x_{\min})$:

$o1 = o2 = 0$	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



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 t s without floating point division:

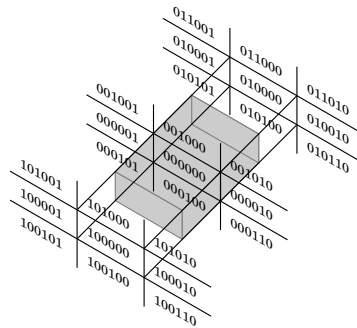
$$t_3(Q_x - P_x)(Q_y - P_y) = (Q_x - P_x)(y_{\max} - P_y),$$

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

etc.

3D Clipping

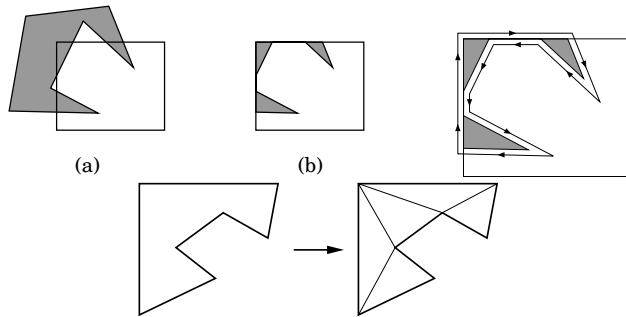
Cohen-Sutherland, outcode



Liang-Barsky tests against a plane with normal N :

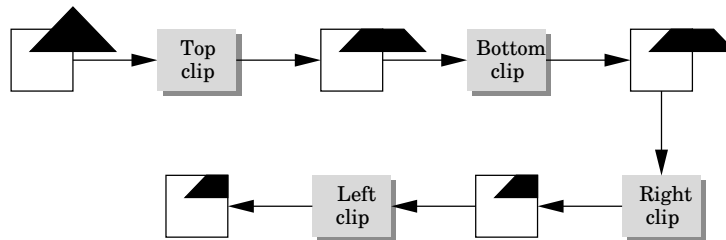
$$(P(1 - t) + Qt - V) \cdot N = 0$$

Polygon Clipping



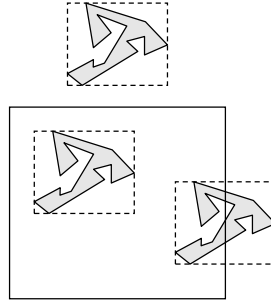
`gluTessellate()`

Sutherland-Hodgson pipeline clipping



Reduce GPU processing: Bounding boxes

curves, surfaces, text (but not raster text, that is in the frame buffer)



Reduce GPU processing: Hidden-surface removal – Culling

Do not draw a planar face if

$$(\text{Eye} - \text{Vertex}) \cdot \text{Normal} \leq 0$$

In window (or ND) coordinates:

if $\sum x_i y_{i+1} - y_i x_{i+1} > 0$ and `GL_CCW`
then `glCullFace(GL_FRONT)` will not remove the face.