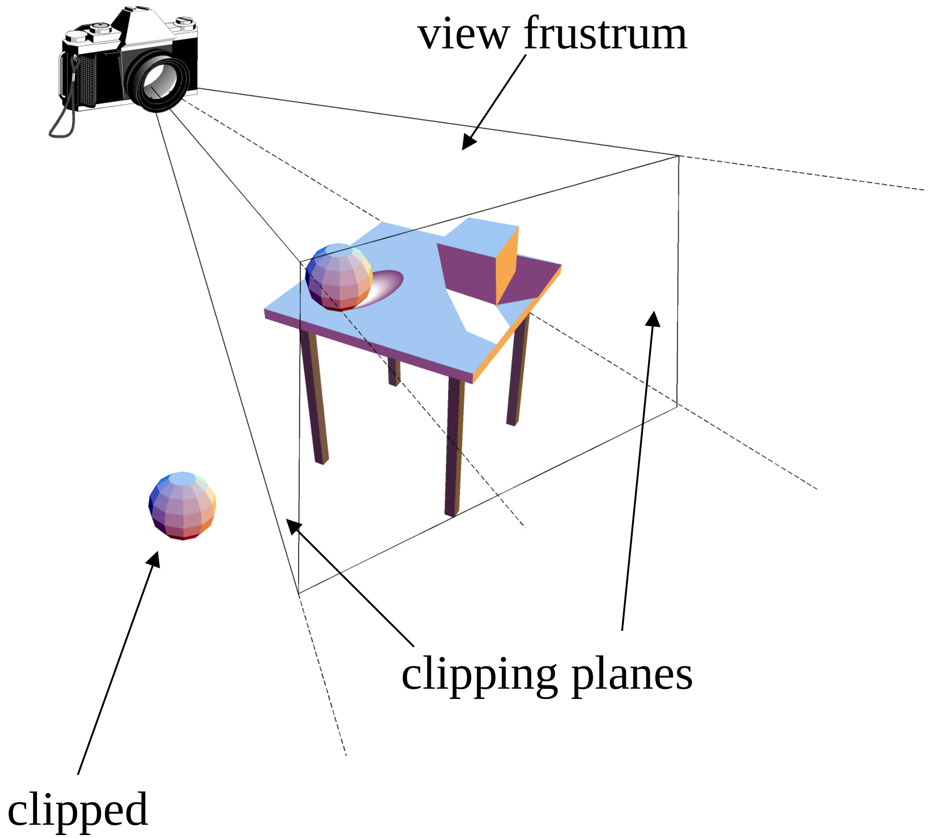
# 3D rezanje

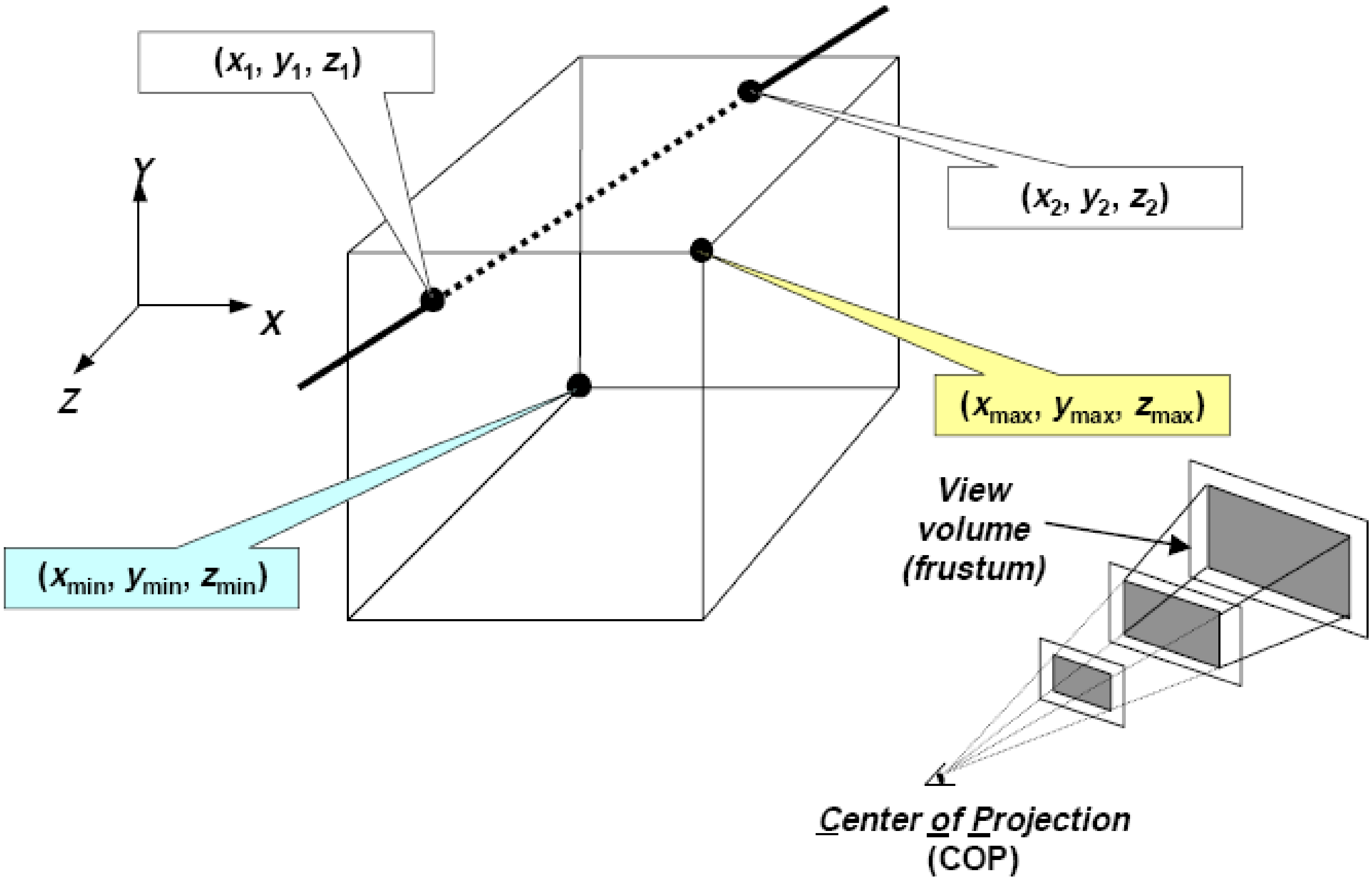
## 3D Clipping



# 3D Projections and Clipping

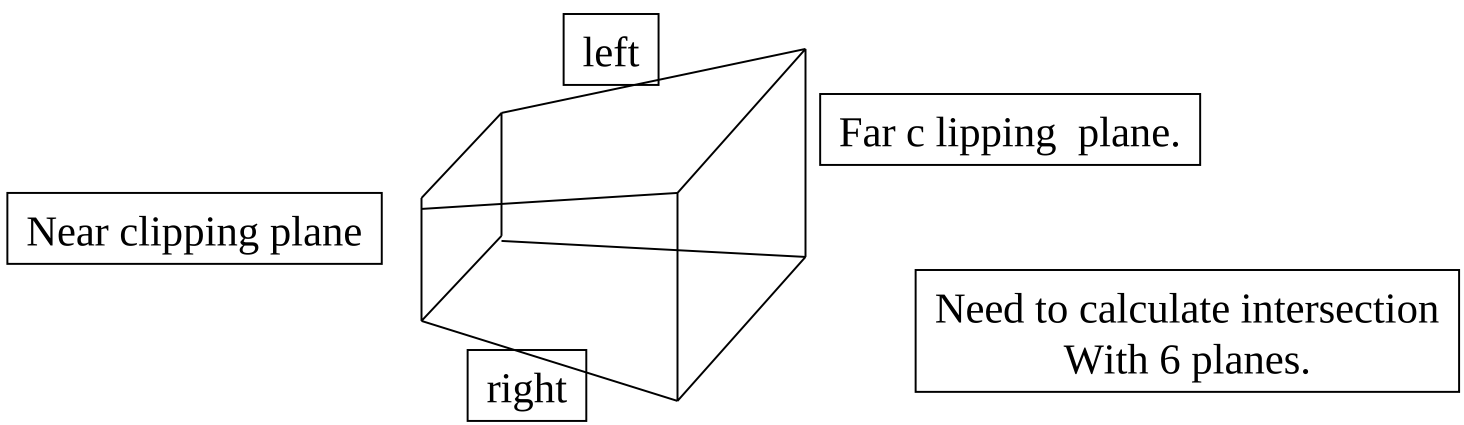
- **Projections (Concluded)**
  - Parallel projection: cuboid view volume
  - Perspective projection: truncated pyramidal view volume (frustum)
  - Problem: how to clip?
- **Clipping**
  - Given: coordinates for primitives (line segments, polygons, circles, ellipses, etc.)
  - Determine: *visible components* of primitives (e.g., line segments)
  - Methods
    - Solving simultaneous equations (quick rejection: testing endpoints)
    - Solving parametric equations
  - Objectives: efficiency (e.g., fewer floating point operations)
- **Clipping in 3D**
  - *Some* 2D algorithms extendible to 3D
  - Specification (and implementation) of view volumes needed

# Paralelni kuboid in View volume



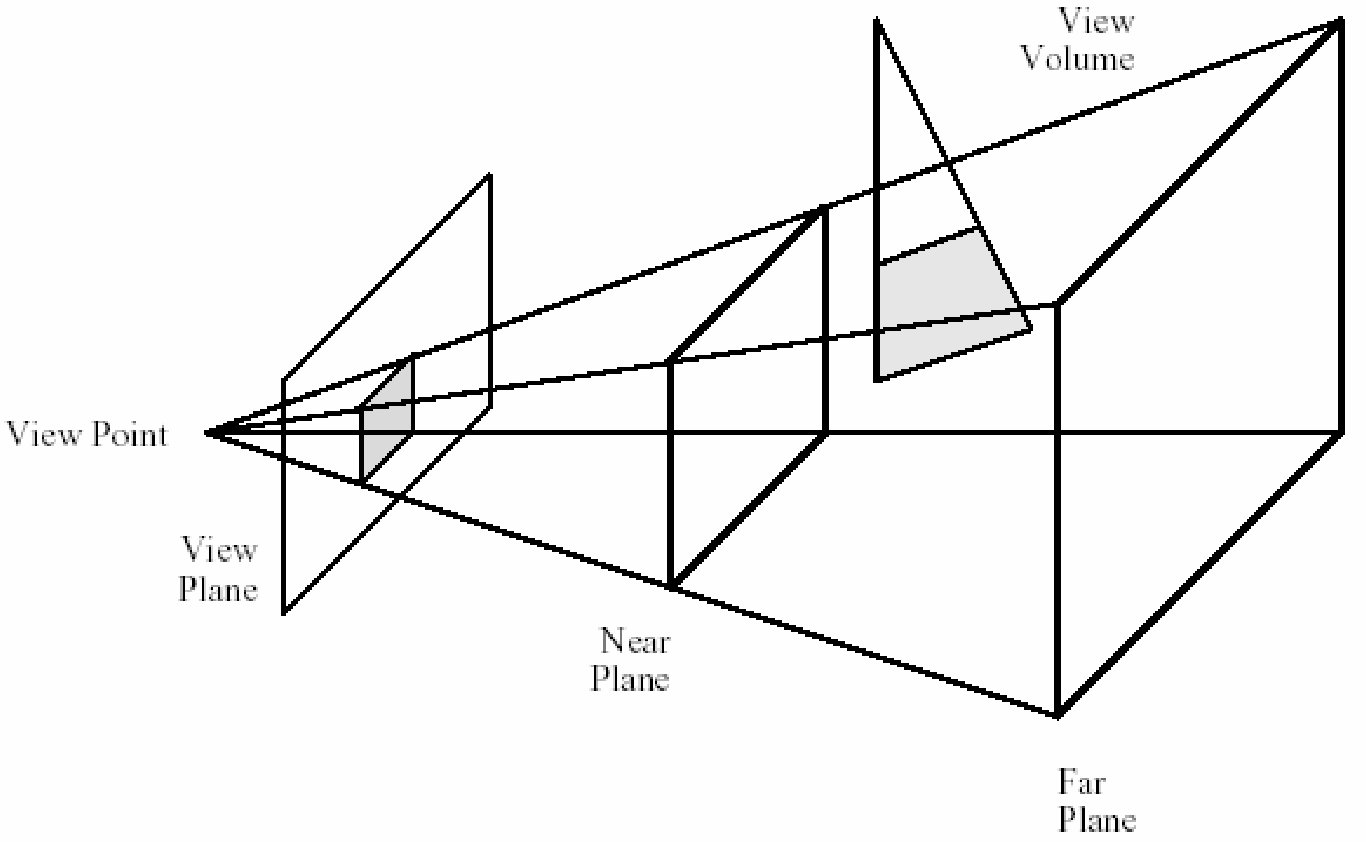
# 3D Clipping

- For orthographic projection, view volume is a box.
- For perspective projection, view volume is a *frustrum*.

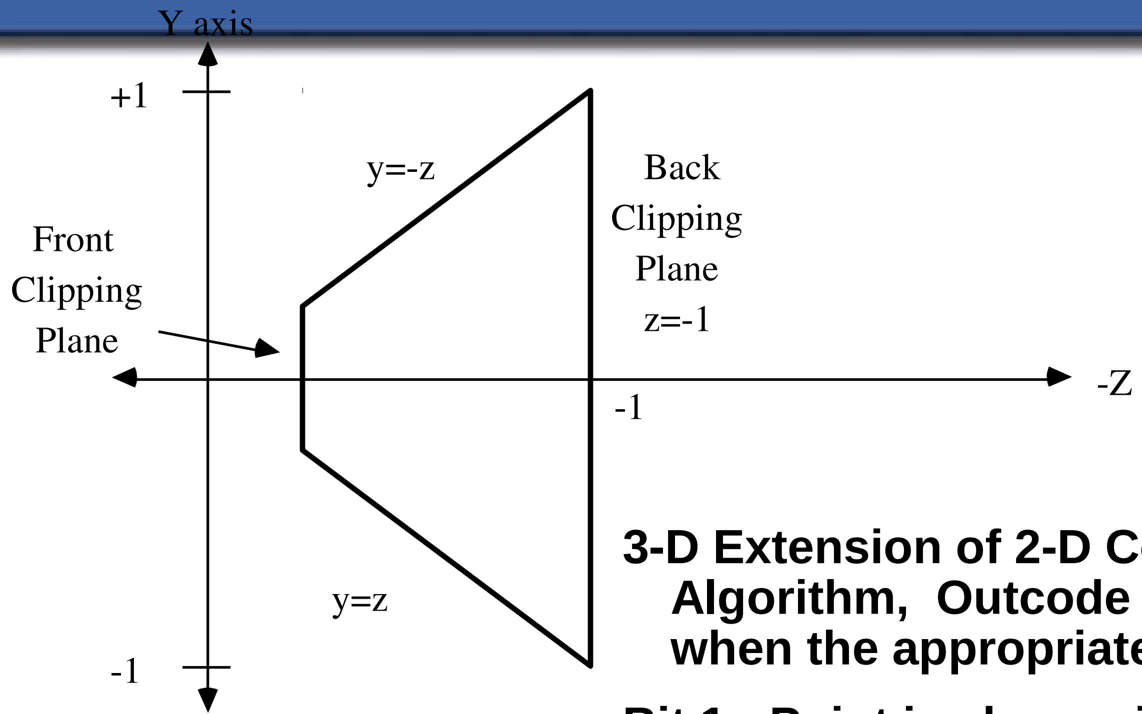


# 3D obrezovanje poligonov

It is sufficient to clip each polygon against the view pyramid.



# Canonical View Volume




**3-D Extension of 2-D Cohen-Sutherland Algorithm, Outcode of six bits. A bit is true (1) when the appropriate condition is satisfied**

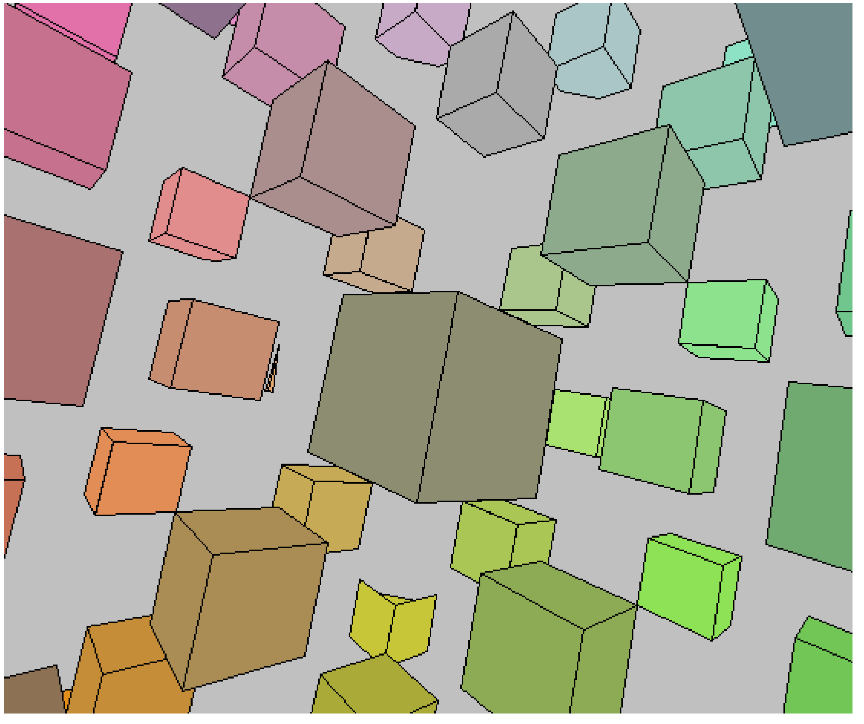
- **Bit 1 - Point is above view volume** $y > -z$
- **Bit 2 - Point is below view volume** $y < z$
- **Bit 3 - Point is right of view volume** $x > -z$
- **Bit 4 - Point is left of view volume** $x < z$
- **Bit 5 - Point is behind view volume** $z < -1$
- **Bit 6 - Point is in front of view volume** $z > zmin$

# 3D Clipping

- Can use Cohen-Sutherland algorithm.
  - Now 6-bit outcode.
  - Trivial acceptance where both endpoint outcodes are all zero.
  - Perform logical AND, reject if non-zero.
  - Find intersect with a bounding plane and add the two new lines to the line queue.
  - Line-primitive algorithm.

# 3D Polygon Clipping

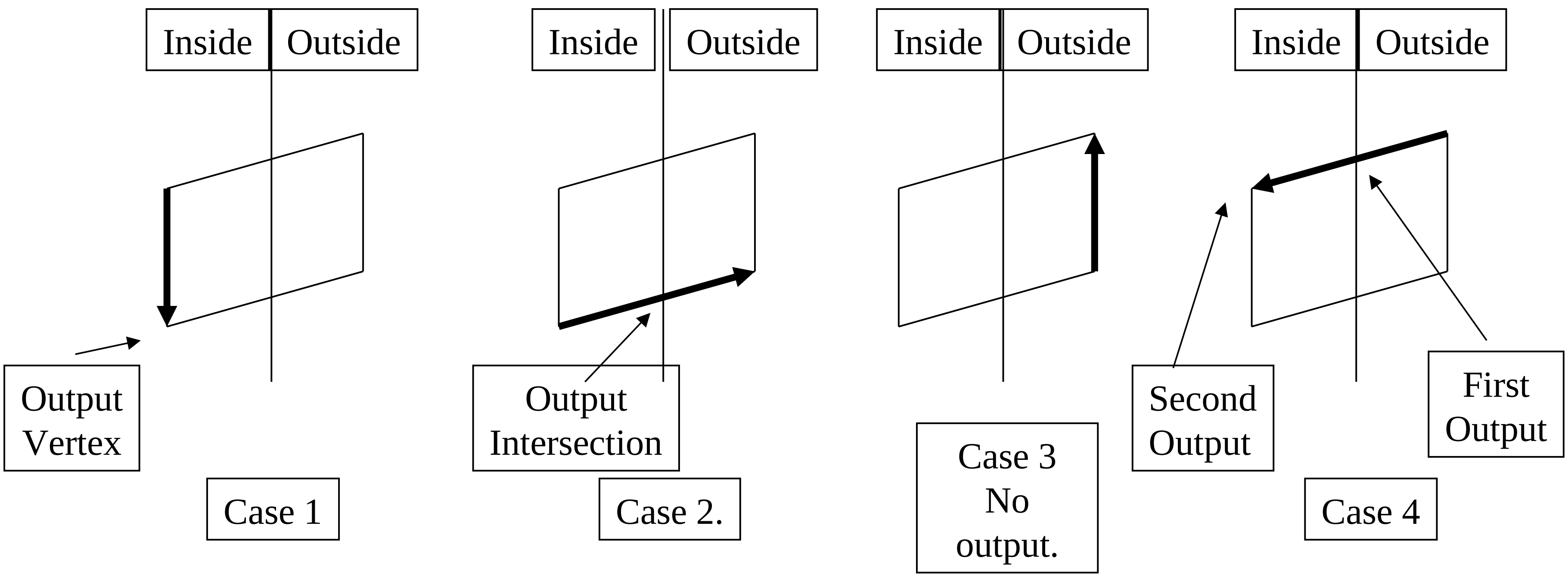
- Sutherland-Hodgman extends easily to 3D.
- Call ‘CLIP’ procedure 6 times rather than 4
- Polygon-primitive algorithm.



Demo

# Sutherland-Hodgman Algorithm

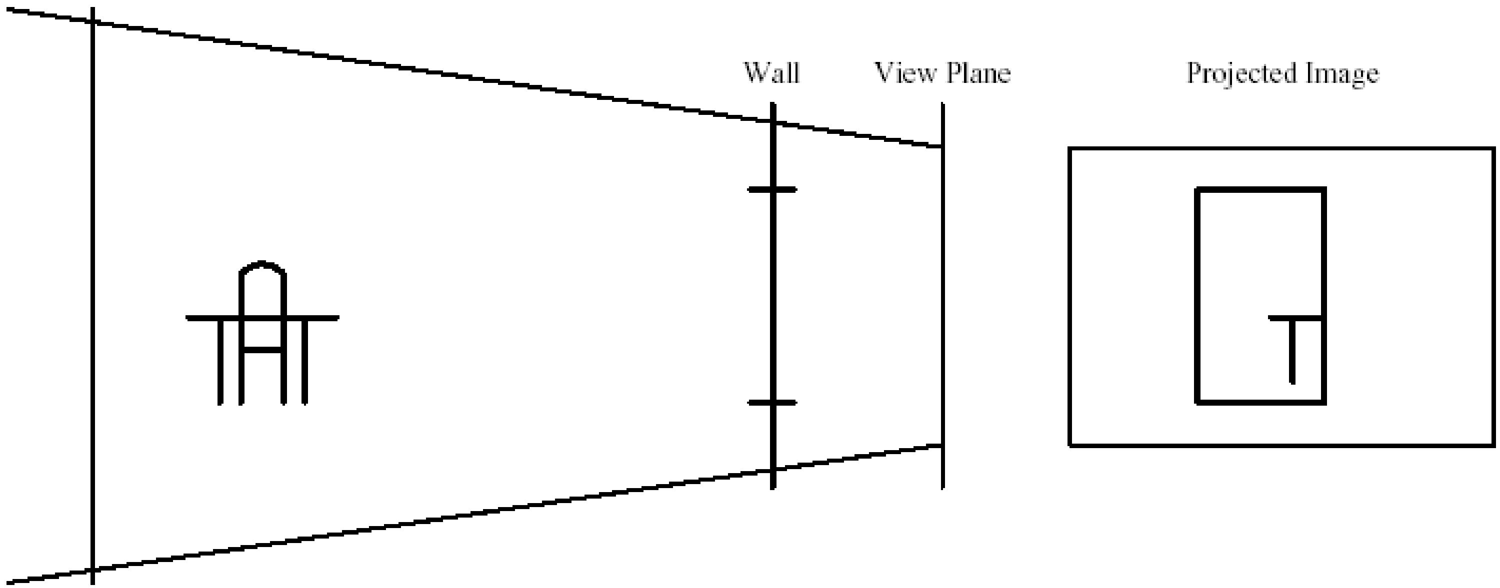
Four cases of polygon clipping :

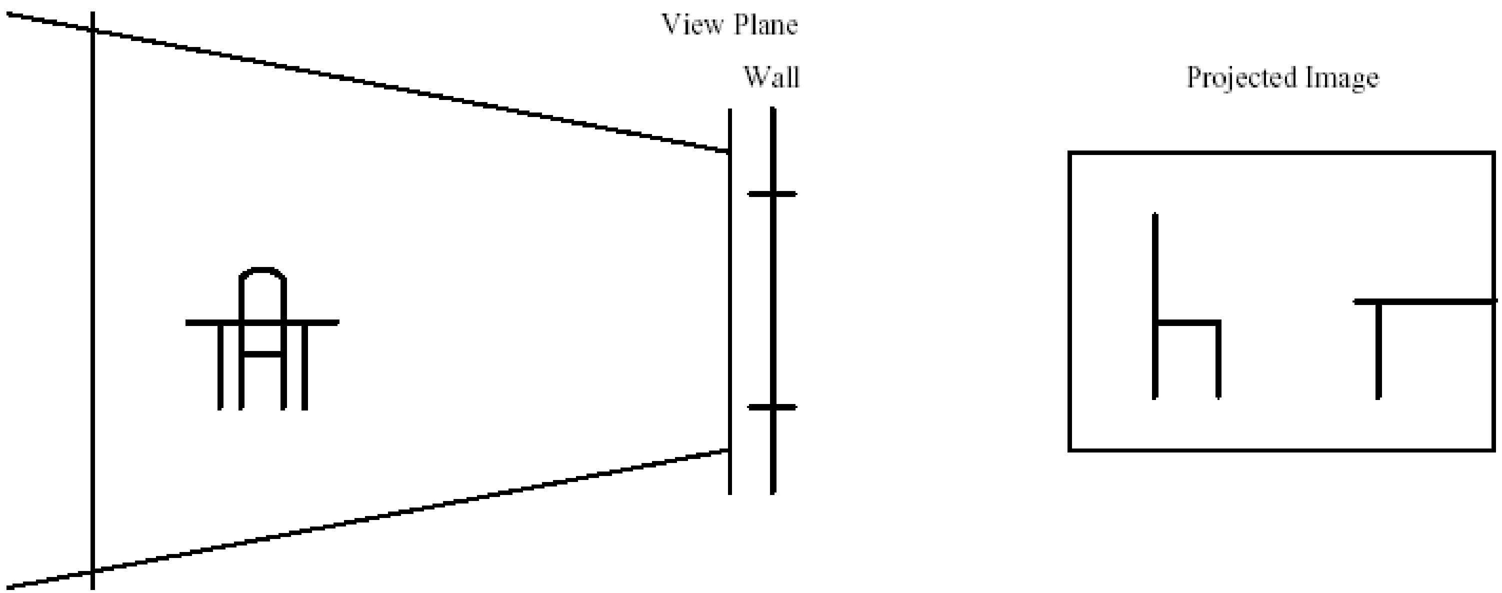


# Clipping and Homogeneous Coordinates

- Efficient to transform frustrum into perspective canonical view volume – unit slope planes.
- Even better to transform to parallel canonical view volume
  - Clipping must be done in homogeneous coordinates.
- Points can appear with –ve W and cannot be clipped properly in 3D.

# Why Clip Against Near and Far?

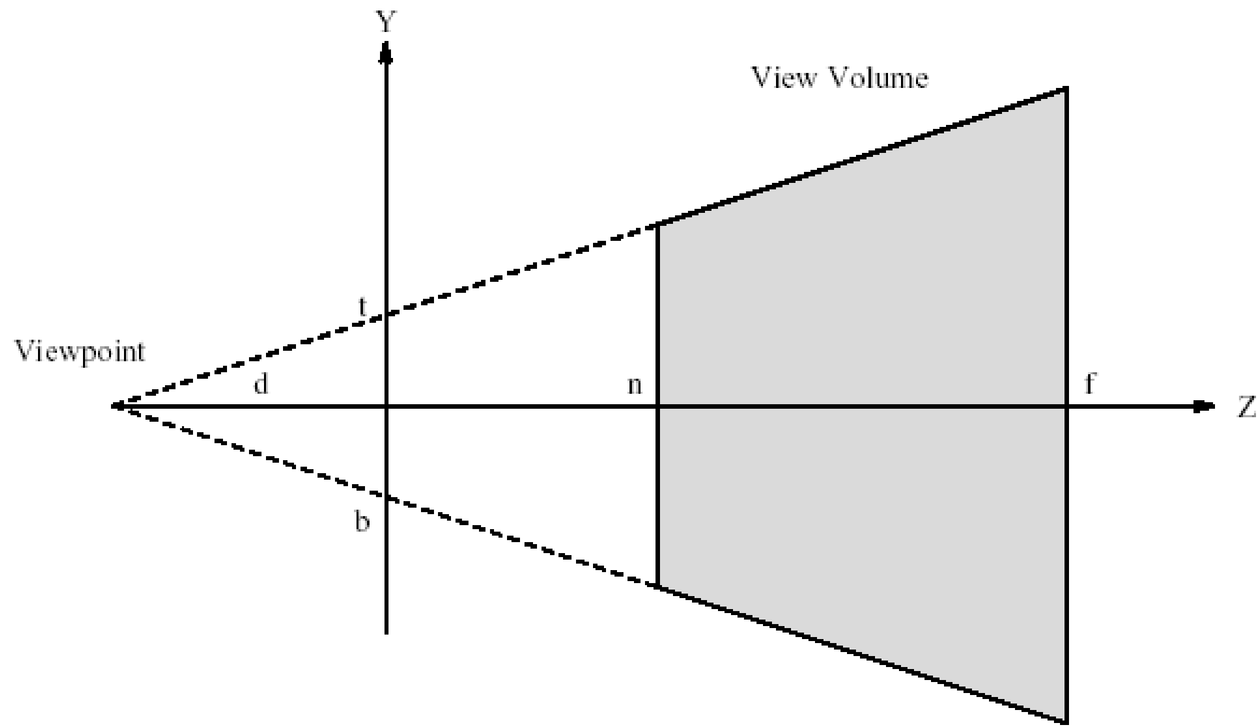


# Clipping Against Pyramid Sides

Let $l$, $r$, $b$, $t$ be the points where the sides of the view pyramid intersect the view plane. Let $d$ be the distance from the origin to the view point. Then

| | |
|---|---|
| slope of the left plane: | $s_L = -1/2(r-l)/d$ |
| slope of the right plane: | $s_R = 1/2(r-l)/d$ |
| slope of the bottom plane: | $s_B = -1/2(t-b)/d$ |
| slope of the top plane: | $s_T = 1/2(t-b)/d$ |



# Equations of the Sides of the View Pyramid

$$\begin{aligned}
L: &\quad x = l + s_L z \\
R: &\quad x = r + s_R z \\
B: &\quad y = b + s_B z \\
T: &\quad y = t + s_T z \\
N: &\quad z = n \\
F: &\quad z = f
\end{aligned}$$

A line from $(x_1, y_1, z_1)$ to $(x_2, y_2, z_2)$ intersects the top plane at $u$ value

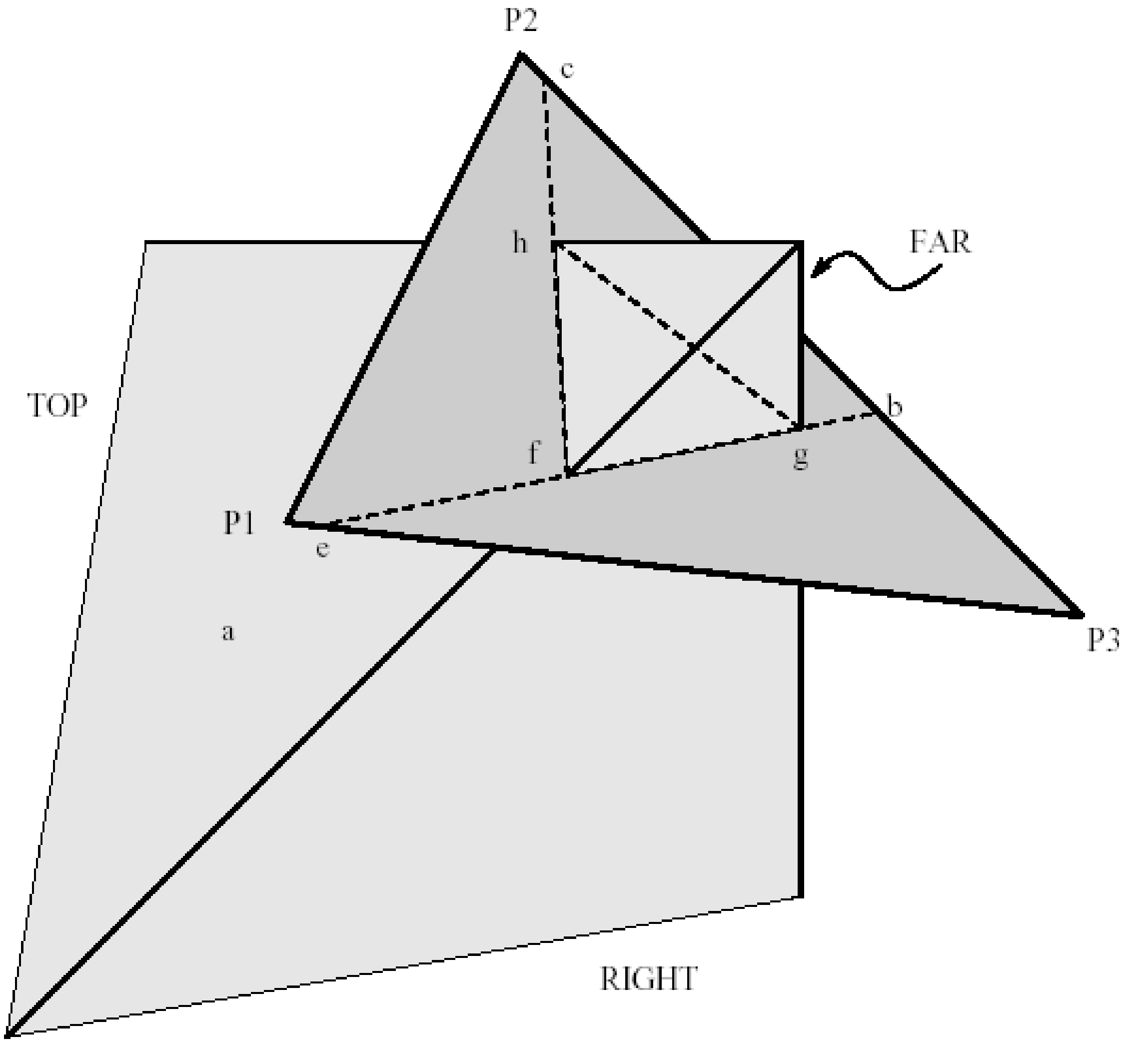
$$u_T = \frac{y_1 - t - s_T z_1}{y_1 - y_2 + s_T(z_2 - z_1)}$$

so we compute the $(x, y, z)$ point of intersection as

$$x = x_1 + u_T(x_2 - x_1) \qquad y = y_1 + u_T(y_2 - y_1) \qquad z = z_1 + u_T(z_2 - z_1)$$

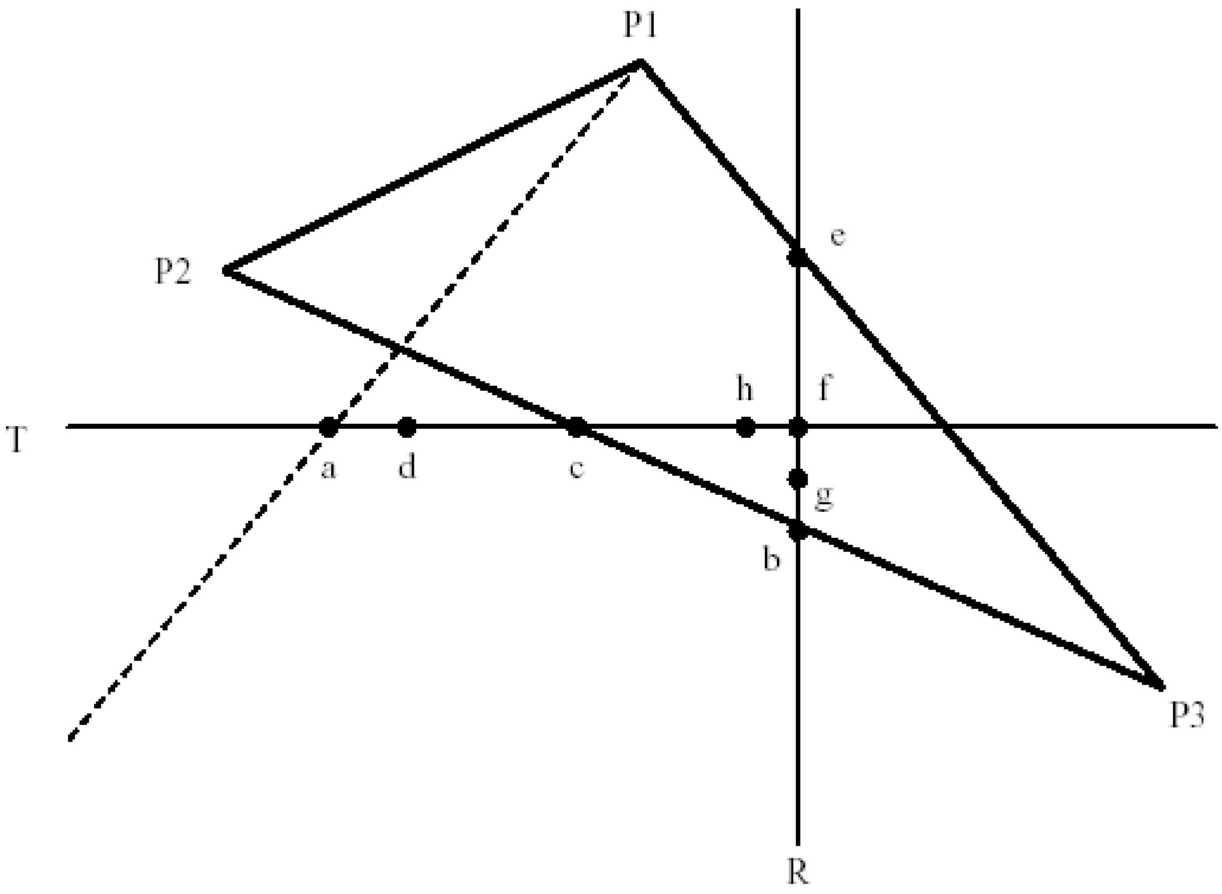
## 3D Clipping Pipeline

The code is exactly analogous to the 2D pipeline with two more clippers: $N$ (near) and $F$ (far).



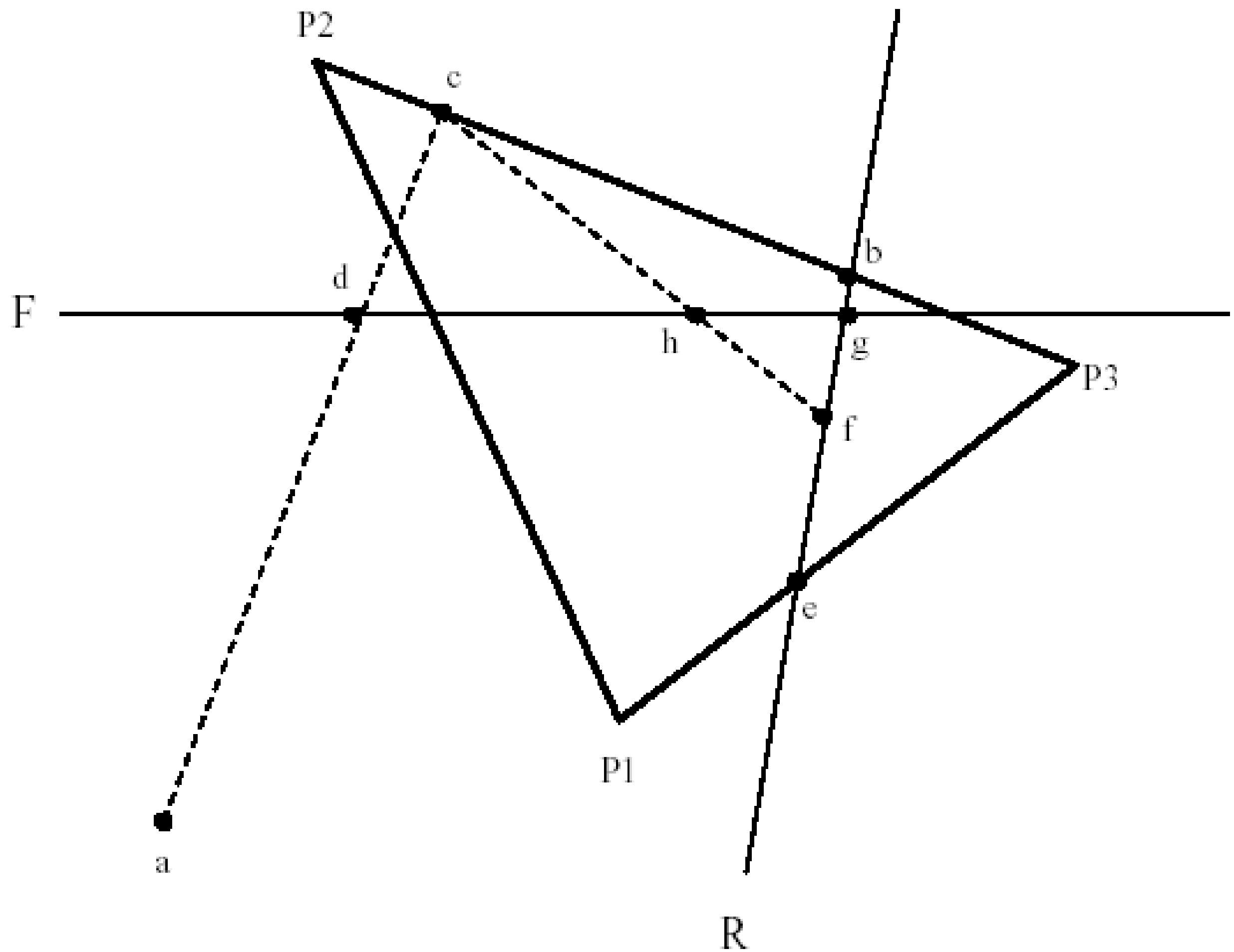
# 3D Clipping Pipeline (2)

Front View



# 3D Clipping Pipeline (3)

Top View



# 3D Clipping Pipeline (4)

First iteration:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $O$ | | $O$ | | $O$ | | $O$ | | $O$ | | $O$ | | |
| $P_1$ | $L$ | $P_1$ | $R$ | $P_1$ | $B$ | $P_1$ | $T$ | $a$ | $N$ | $a$ | $F$ | $a$ |
| $P_2$ | $L$ | $P_2$ | $R$ | $P_2$ | $B$ | $P_2$ | $T$ | | | | | |
| $P_3$ | $L$ | $P_3$ | $R$ | $b$ | $B$ | $b$ | $T$ | $c$ | $N$ | $c$ | $F$ | $d$ |
| | | | | | | | | $b$ | $N$ | $b$ | $F$ | |

Second Iteration:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_3$ | | $P_3$ | | $b$ | | $b$ | | $b$ | | $b$ | | |
| $P_1$ | $L$ | $P_1$ | $R$ | $e$ | $B$ | $e$ | $T$ | $f$ | $N$ | $f$ | $F$ | $g$ |
| | | | | | | | | | | | | $f$ |
| | | | | $P_1$ | $B$ | $P_1$ | $T$ | | | | | |
| $P_2$ | $L$ | $P_2$ | $R$ | $P_2$ | $B$ | $P_2$ | $T$ | | | | | |
| $P_3$ | $L$ | $P_3$ | $R$ | $b$ | $B$ | $b$ | $T$ | $c$ | $N$ | $c$ | $F$ | $h$ |
| | | | | | | | | $b$ | $N$ | $b$ | $F$ | |