# Computer Graphics

Lecture 6

View Transformation and Clipping

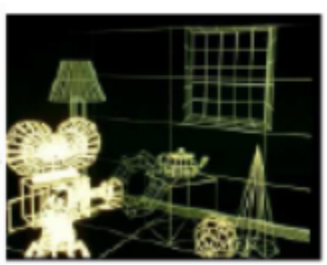
Taku Komura

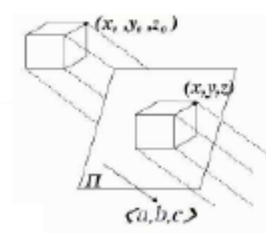
# Overview

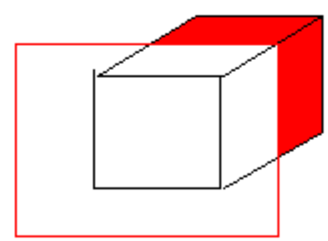
- View transformation
  - Recap of homogeneous transformation
  - Parallel projection
  - Perspective projection
  - Canonical view volume
- Clipping
  - Line / Polygon clipping

# Procedure

1. Transform into camera coordinates. (done in Lecture 3)
2. Perform projection into *view volume* or *screen coordinates*.
3. Clip geometry outside the *view volume*.

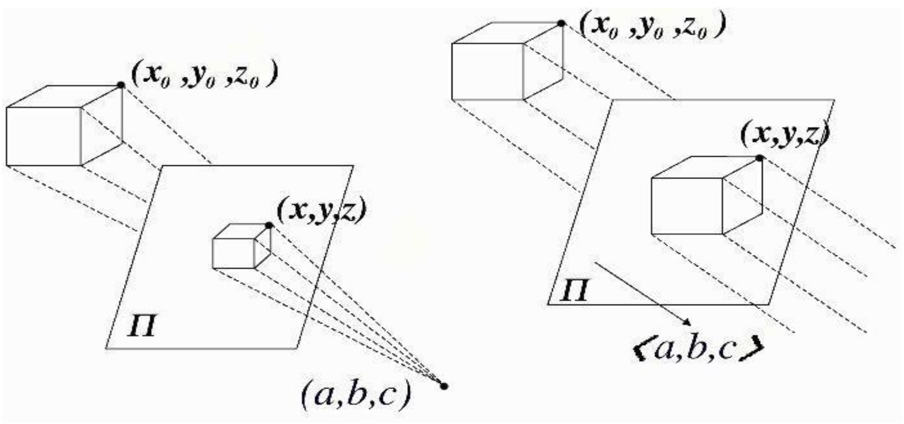


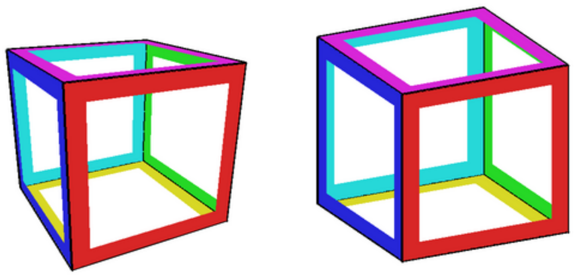


# View Projection : Topics

- Homogenous transformation
- Parallel projection
- Perspective projection
- Canonical view volume

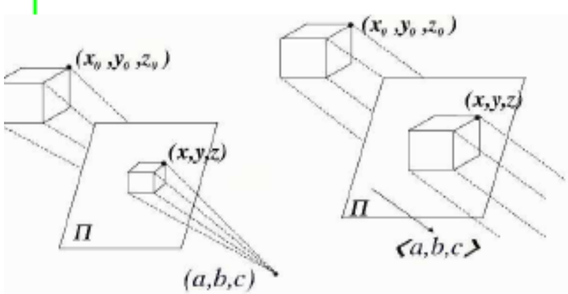


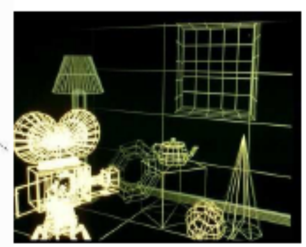



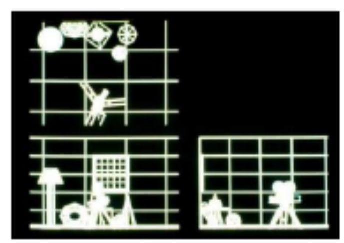
# Homogeneous Transformations

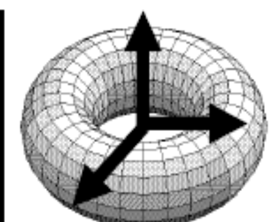
$$v = M_{proj}\, M_{c\leftarrow w}\, M_{w\leftarrow l}\, v_l$$

| Projection matrix | World to camera matrix | Local to world matrix | Local coordin ates |
|---|---|---|---|







The projection matiix should be 4x4 matrices

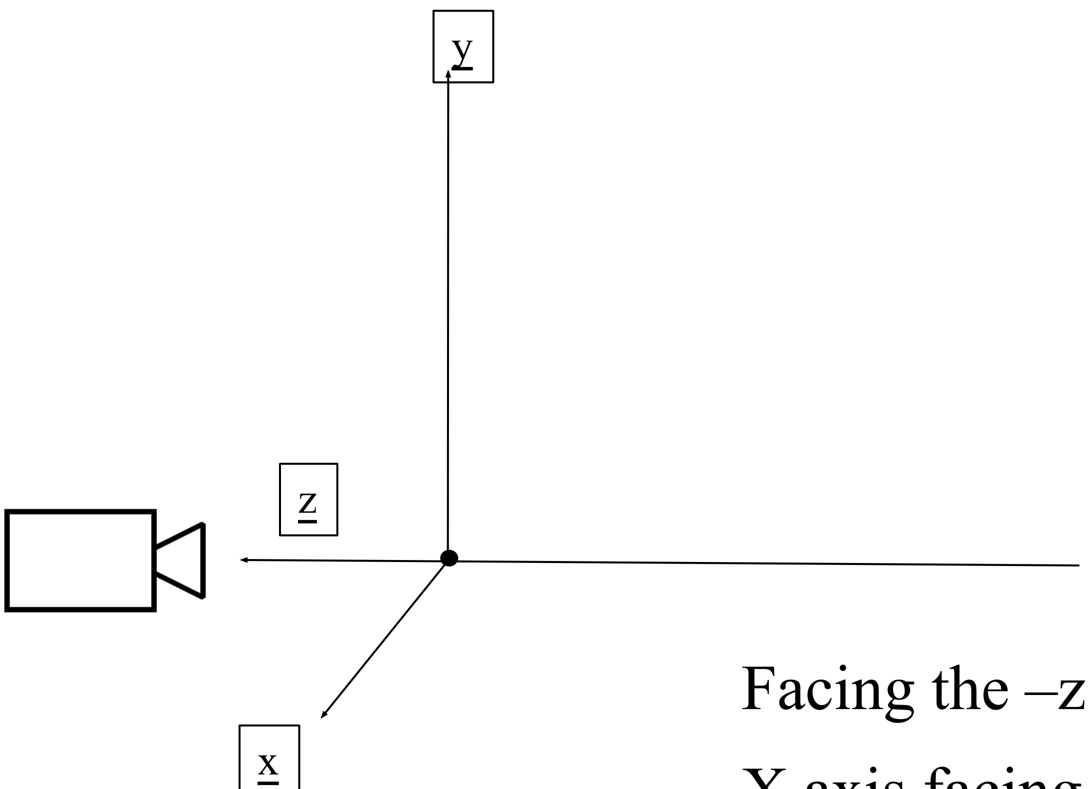
to allow general concatenation

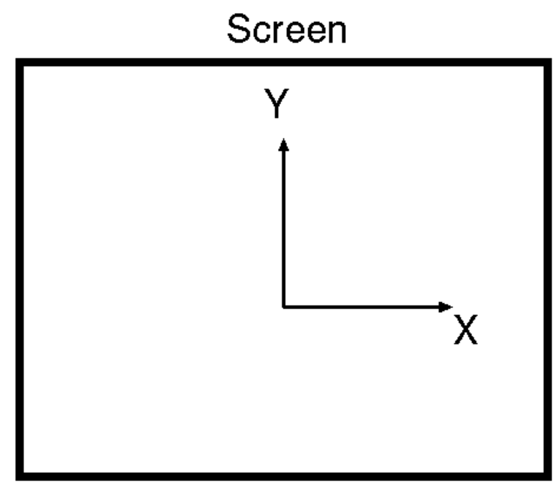
# Homogeneous Coordinates

- Introduced to represent various transformations by multiplications (in Lecture 2)
- In homogenous coordinates, (x,y,z,w) represent the same point when all elements are multiplied by the same factor
  - (2,0,1,1) and (4,0,2,2) are the same points
  - To bring back to Cartesian space, need to divide the other elements by the fourth element *w*
    - $(\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{w}) \rightarrow$ (x/w, y/w, z/w, 1)

# Camera Coordinate System we use

# (same as OpenGL)



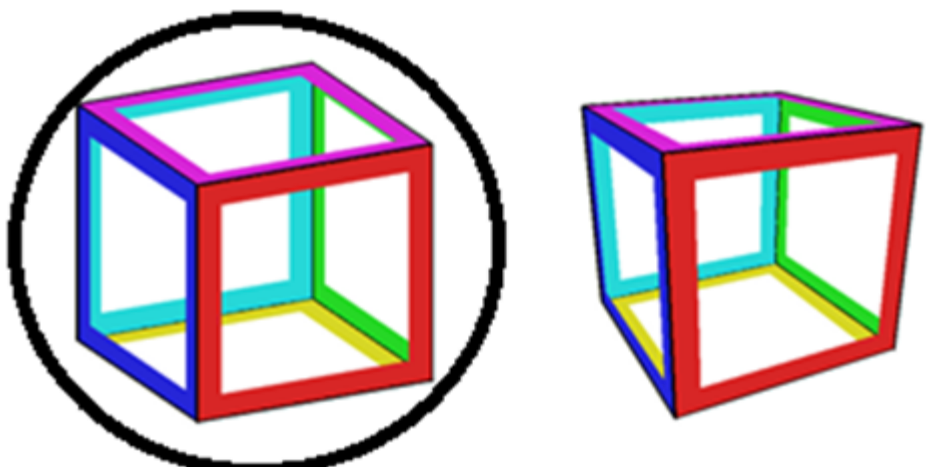




Facing the –z direction

X axis facing the right side

# Parallel projections (Orthographic projection)

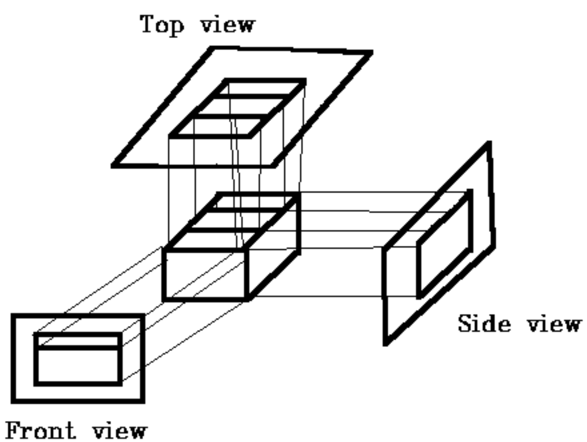
- Specified by a direction of projection, rather than a point.
- Objects of same size appear at the same size after the projection



# Parallel projection.

Orthographic Projection onto a plane at z = 0.

$$x_p = x \ , \ y_p = y \ , \ z = 0.$$

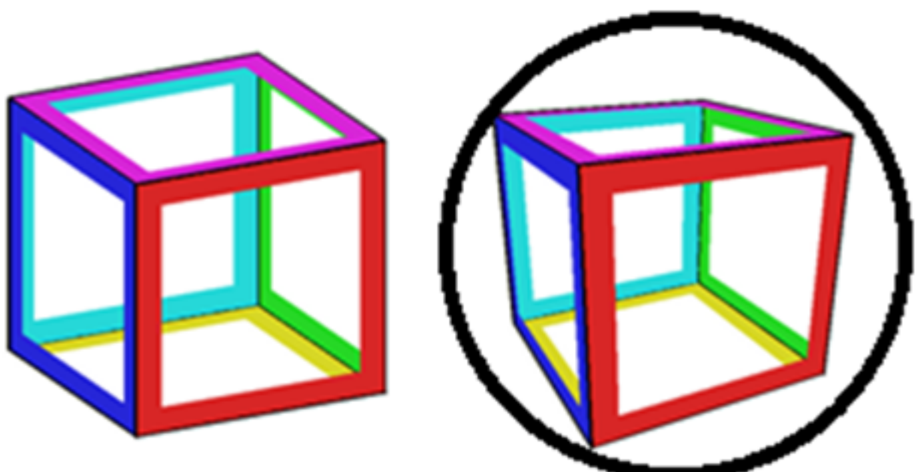



$$M_{orth} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

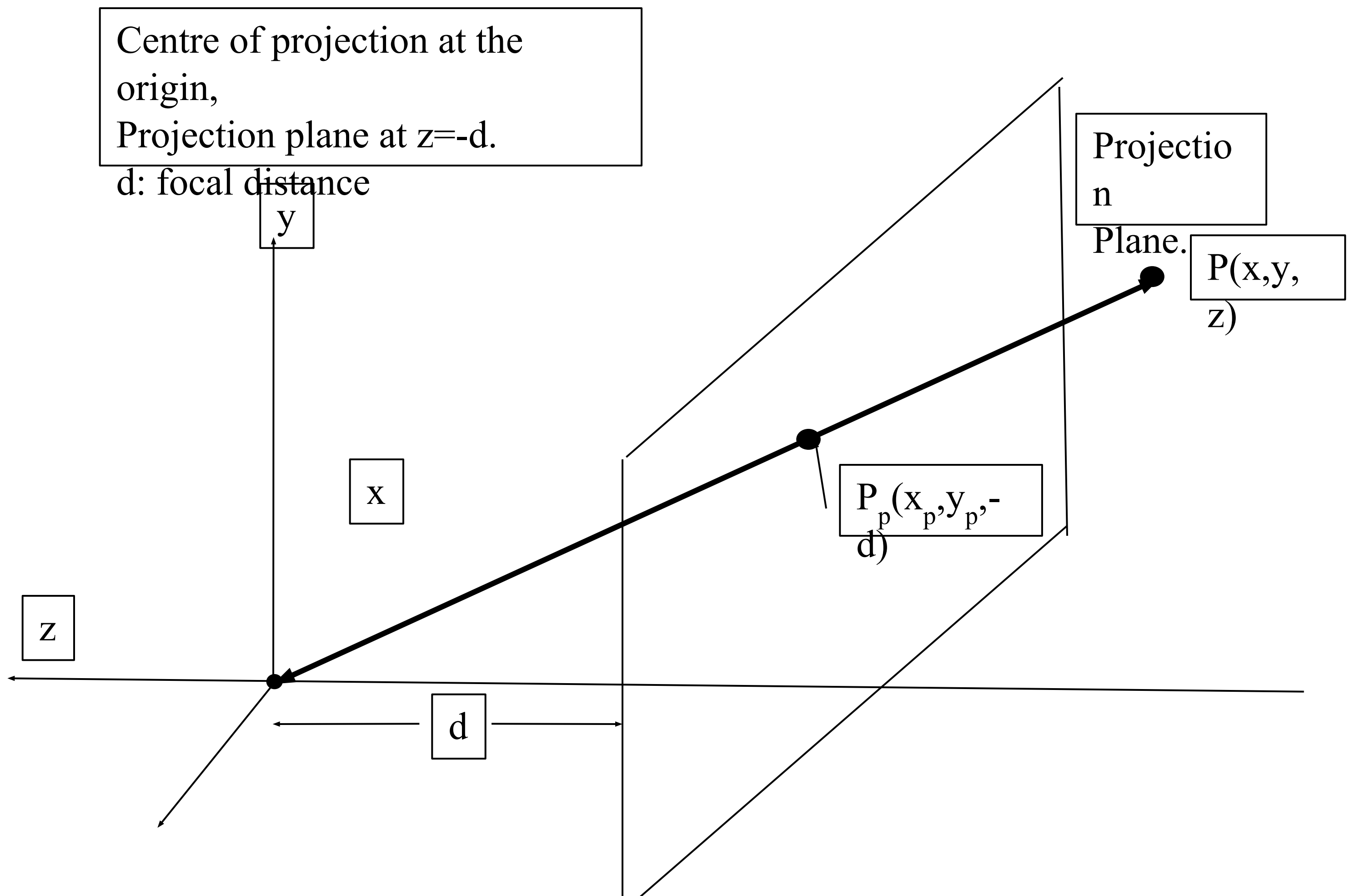
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

# Perspective Projection

- Objects far away appear smaller, closer objects appear bigger
- Specified by a center of projection and the focal distance (distance from the eye to the projection plane)



# Perspective projection

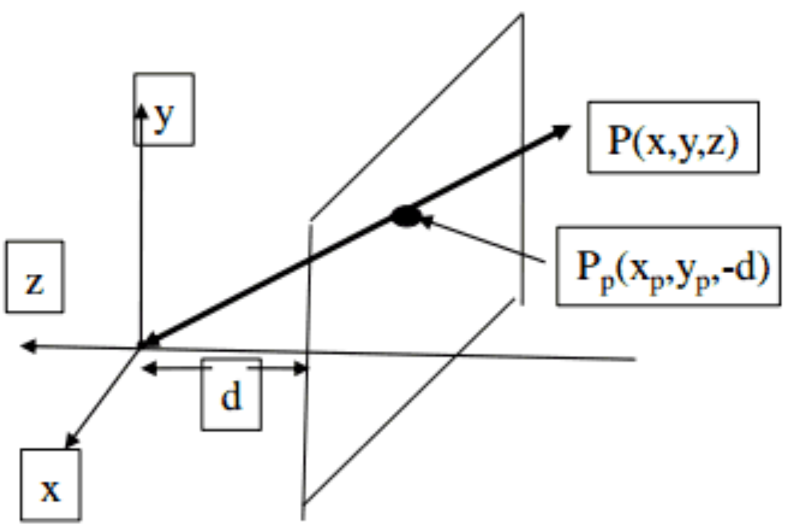


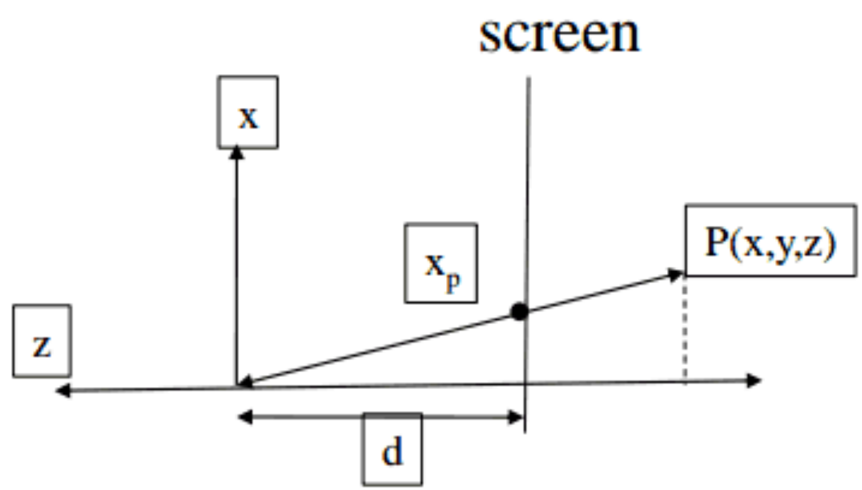
# Perspective projection – simplest case.

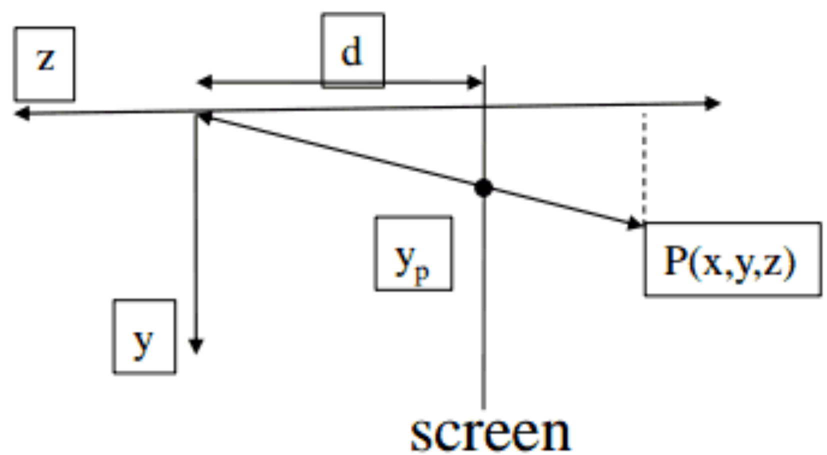
From similar triangles:

$$\frac{x_p}{d} = \frac{x}{-z}; \quad \frac{y_p}{d} = \frac{y}{-z}$$

$$x_p = \frac{d \cdot x}{-z} = \frac{x}{-z/d}; \quad y_p = \frac{d \cdot y}{-z} = \frac{y}{-z/d}$$



# Perspective projection.

$$\begin{bmatrix} x_p & y_p & -d & 1 \end{bmatrix}^T = \begin{bmatrix} -d.x/z & -d.y/z & -d & 1 \end{bmatrix}^T = \begin{bmatrix} x & y & z & -z/d \end{bmatrix}^T$$

Using homogeneous transformation, perspective projection can be represented as a 4x4 matrix multiplication :

$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix}$$

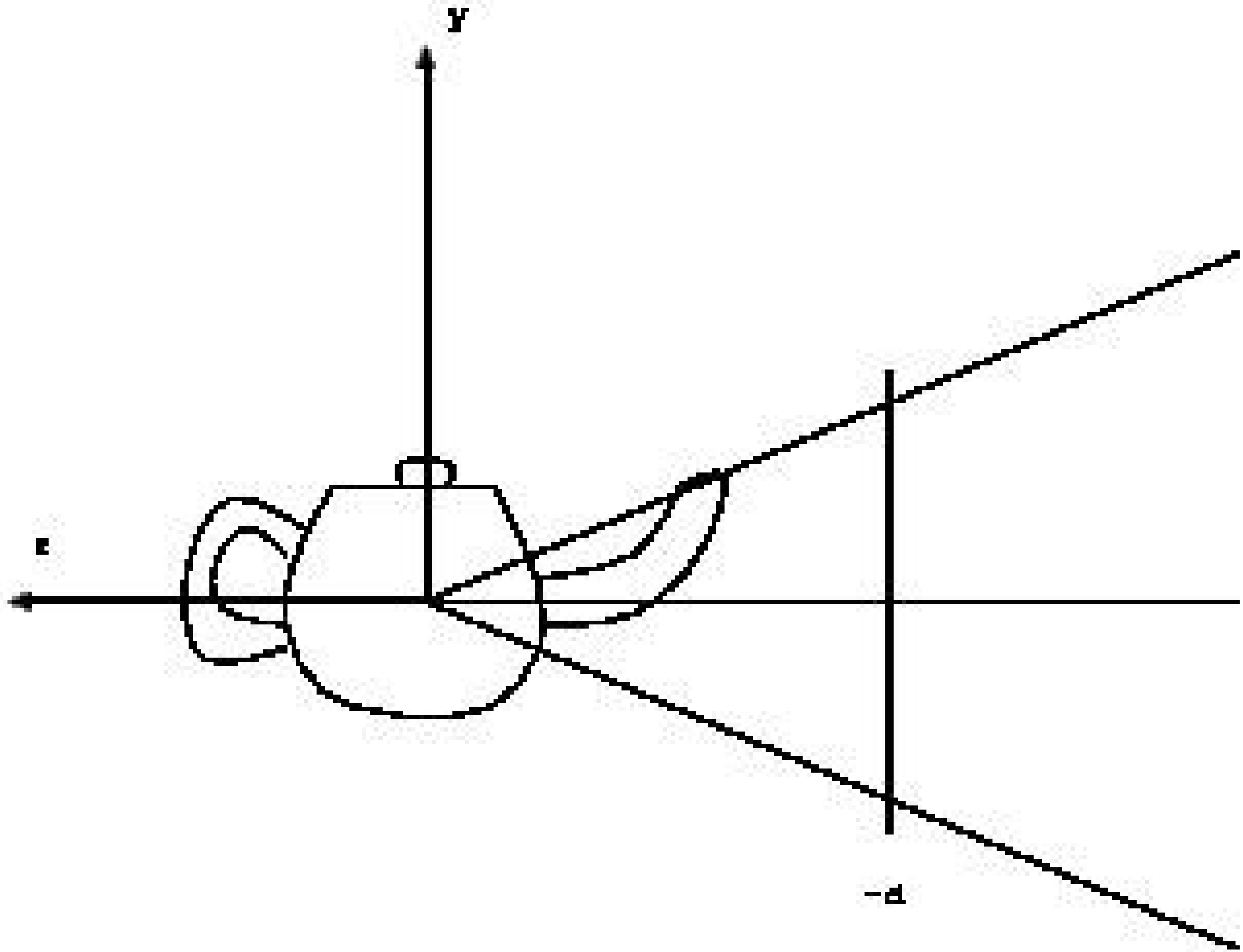
# Perspective projection.

Projected point : $P_p = \begin{bmatrix} X & Y & Z & W \end{bmatrix}^T$

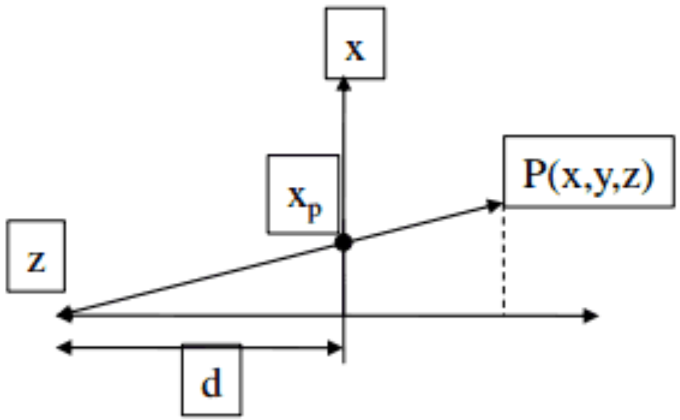
$$P_p = M_{per} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

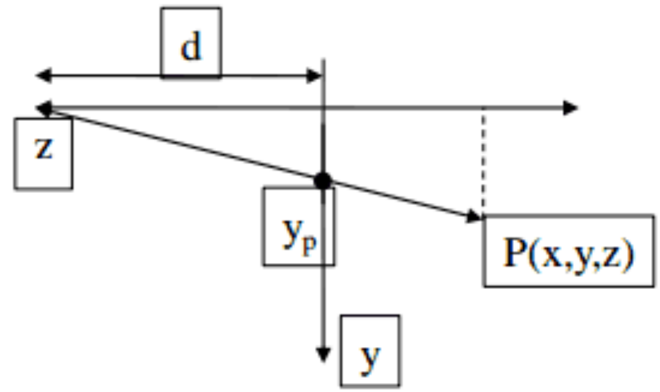
$$= \begin{bmatrix} X & Y & Z & W \end{bmatrix}^T = \begin{bmatrix} x & y & z & -z/d \end{bmatrix}^T$$

$$\left( \frac{X}{W}, \ \frac{Y}{W}, \ \frac{Z}{W} \right) = \left( \frac{x}{-z/d}, \ \frac{y}{-z/d}, \ -d \right)$$


y
z
-d

# Alternative formulation.



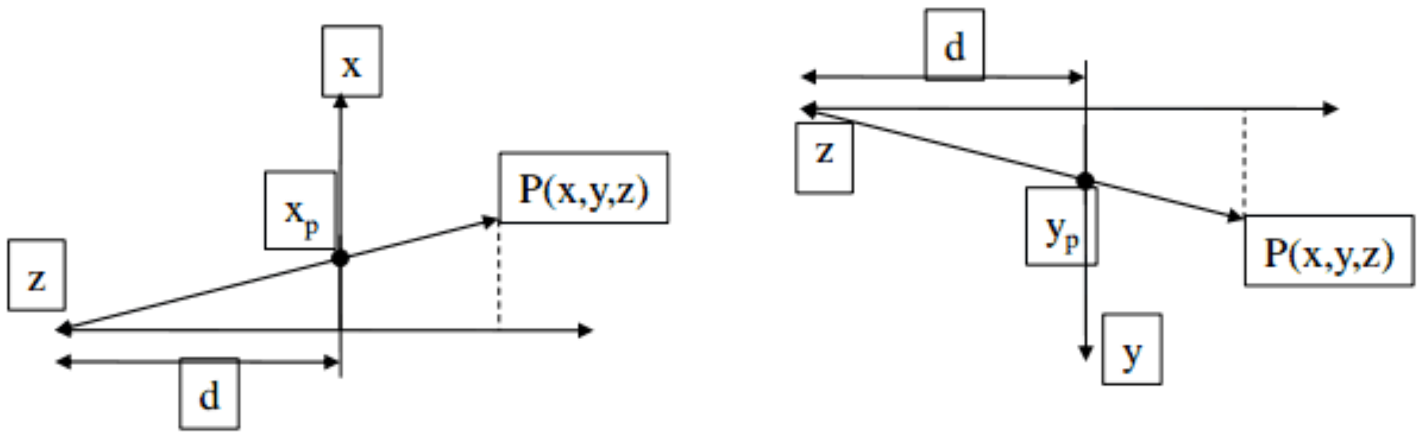

Projection plane at z = 0
Centre of projection at
z = d

$$\frac{x_p}{d} = \frac{x}{-z+d}; \quad \frac{y_p}{d} = \frac{y}{-z+d}$$

Multiply by d

$$x_p = \frac{d \cdot x}{-z+d} = \frac{x}{(-z/d)+1}; \quad y_p = \frac{d \cdot y}{-z+d} = \frac{y}{(-z/d)+1}$$

# Alternative formulation.

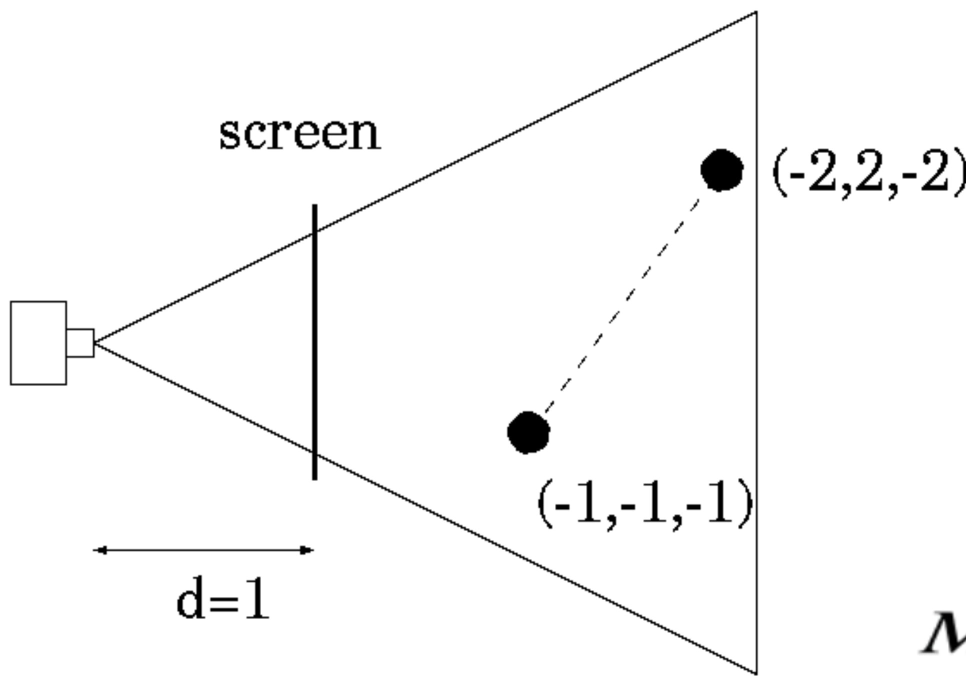



Projection plane at z = 0,
Centre of projection at
z = d

Now we can allow $d \rightarrow \infty$

$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix}$$

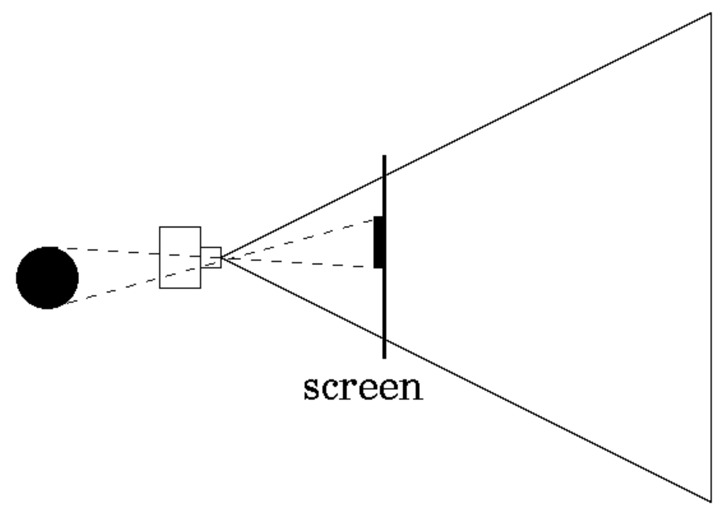
# Exercise: where will the two points be projected onto?




$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix}$$

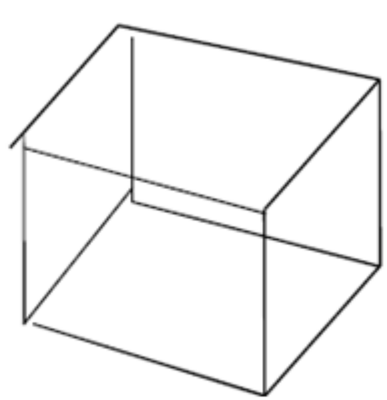
# Problems

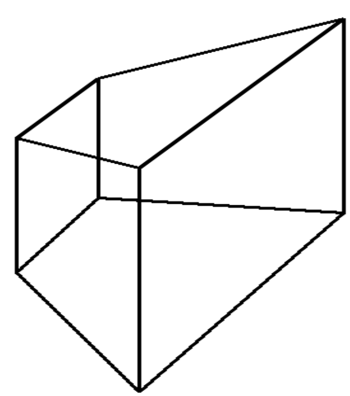
- After projection, the depth information is lost
  - We need to preserve the depth information for hidden surface removal
- Objects behind the camera are projected to the front of the camera



# 3D View Volume

- The volume in which the visible objects exist
  - For parallel projection, view volume is a box.
  - For perspective projection, view volume is a *frustum*.
- The surfaces outside the view volume must be clipped

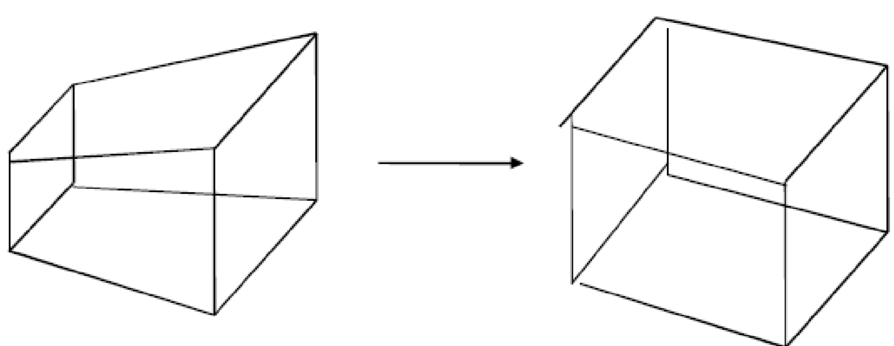




# Canonical View Volume

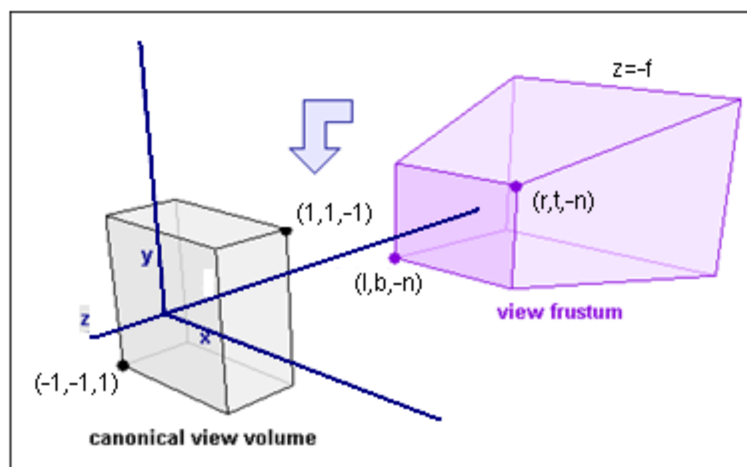
- Checking if a point is within a frustum is costly
- We can transform the frustum view volume into a normalized canonical view volume
- By using the idea of perspective transformation

- Much easier to clip surfaces and calculate hidden surfaces



# Transforming the View Frustum

- Let us define parameters (l,r,b,t,n,f) that determines the shape of the frustum
- The view frustum starts at z=-n and ends at z=-f, with $0<n<f$
- The rectangle at z=-n has the minimum corner at (l,b,-n) and the maximum corner at (r,t,-n)



# Transforming View Frustum into a Canonical view-volume

The perspective canonical view-volume can be transformed to the parallel canonical view-volume with the following matrix:

$$If\ z \in [-n, -f](0 < n < f)\ then$$

$$P_p = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

# Final step.

- Divide by w to get the 3-D Cartesian coordinates
- 3D Clipping
  - The Canonical view volume is defined by:

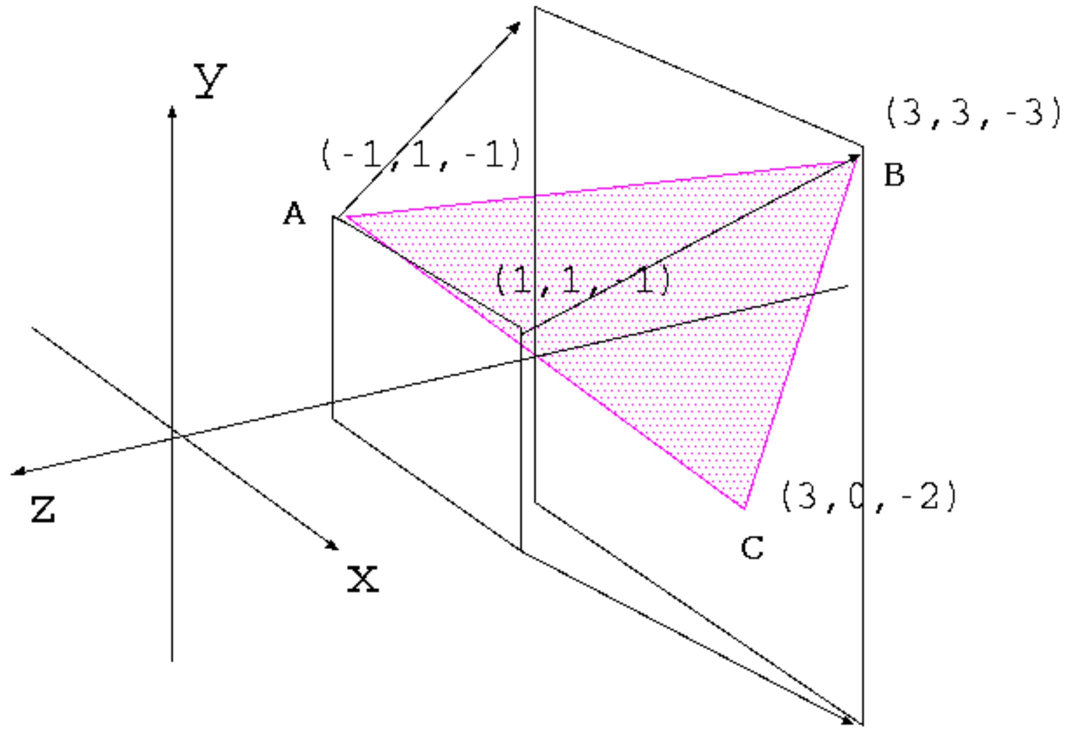
$$-1 \leq x \leq 1,\ -1 \leq y \leq 1\ ,\ -1 \leq z \leq 1$$

  - Simply need to check the (x,y,z) coordinates and see if they are within the canonical view volume

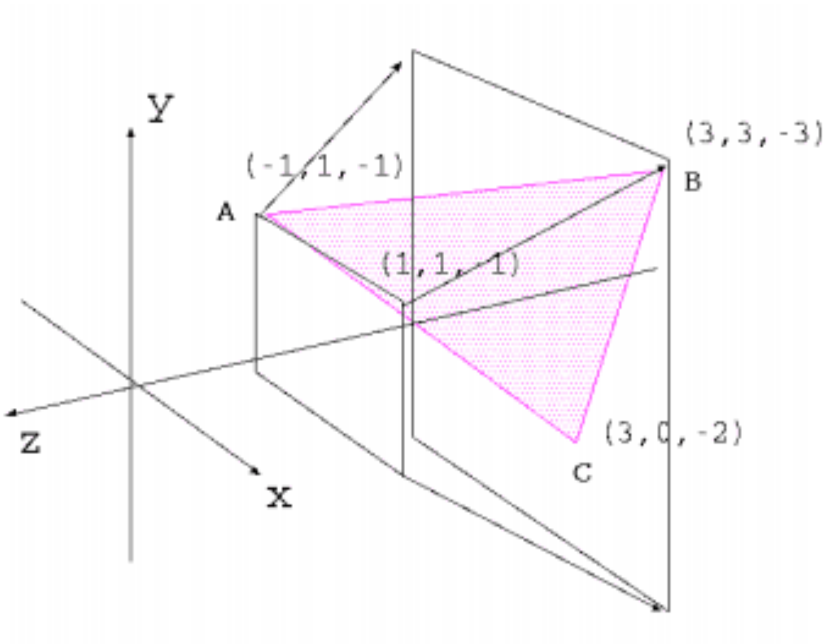
# Exercise

$If\ z \in [-n, -f] (0 < n < f)\ then$

$$P_p = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$




- How does ABC look like after the projection?




$$\text{If } z \in [-n, -f]\,(0 < n < f) \text{ then}$$

$$P_p = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

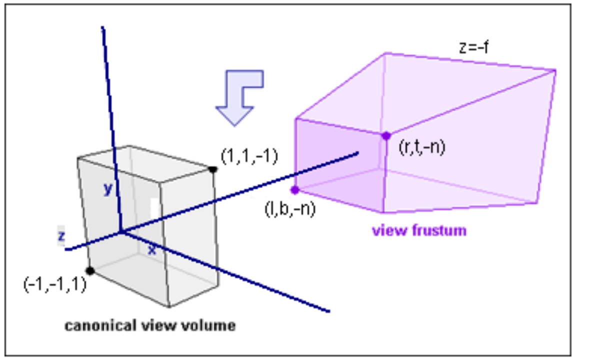
$$n = 1, f = 3, r = 1, l = -1, t = 1, b = -1$$

$$P_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & -3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & -3 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{array}{c} \begin{array}{ccc} A & B & C \\ \downarrow & \downarrow & \downarrow \end{array} \\ \begin{bmatrix} -1 & 3 & 3 \\ 1 & 3 & 0 \\ -1 & -3 & -2 \\ 1 & 1 & 1 \end{bmatrix} \end{array} = \begin{bmatrix} -1 & 3 & 3 \\ 1 & 3 & 0 \\ -1 & 3 & 1 \\ 1 & 3 & 2 \end{bmatrix} = \begin{array}{c} \text{projected} \\ \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} -1 & 1 & \frac{3}{2} \\ 1 & 1 & 0 \\ -1 & 1 & \frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

# Summary of Projection

- Two kind of projections:
  - parallel and perspective
- We can project points onto the screen by using projection matrices
- Canonical view volume is useful for telling if the point is within the view volume
  - parts outside must be clipped

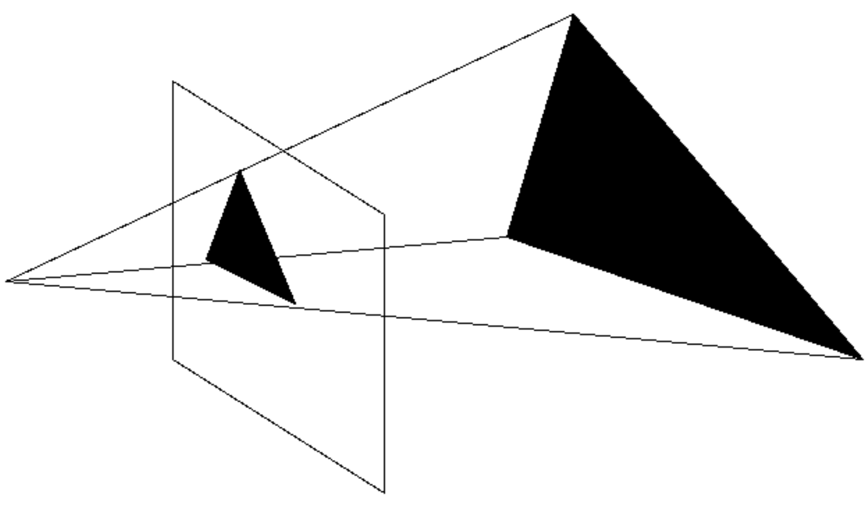


# Overview

- View transformation
  - Parallel projection
  - Perspective projection
  - Canonical view volume
- **Clipping**
  - **Line / Polygon clipping**

# Projecting polygons and lines

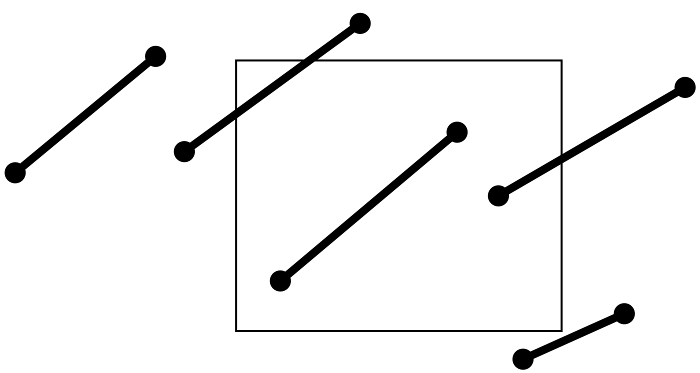
- After projection, a line in 3D space becomes a line in 2D space
- A polygon in 3D space becomes a polygon in 2D space

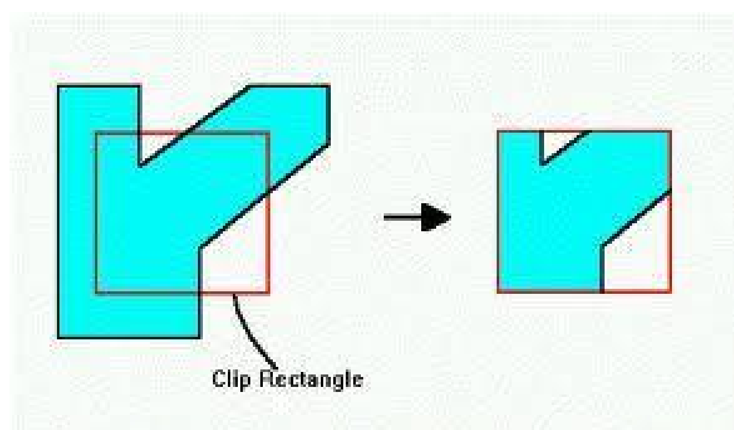


# Clipping

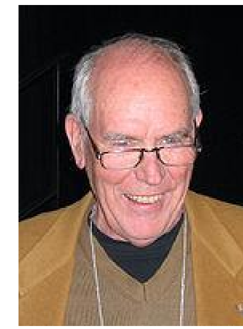
- We need to clip objects outside the canonical view volume

- Clipping lines (Cohen-Sutherland algorithm)
- Clipping polygons (Sutherland-Hodgman algorithm)





# Cohen-Sutherland algorithm

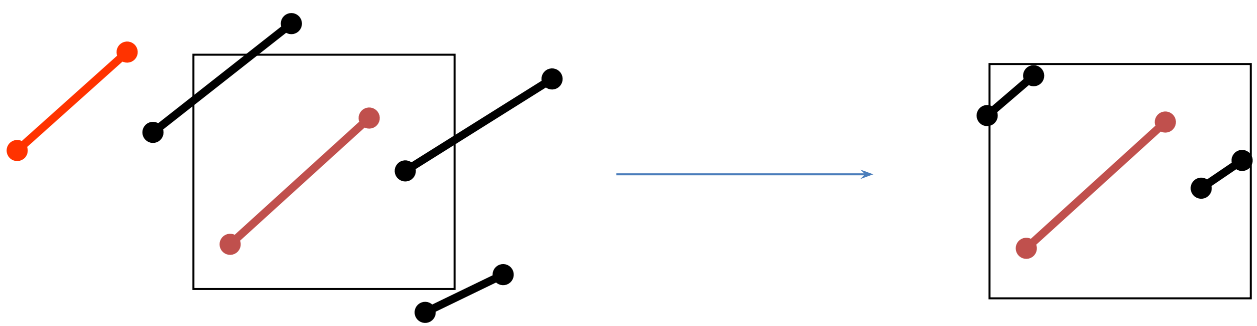


A systematic approach to clip lines

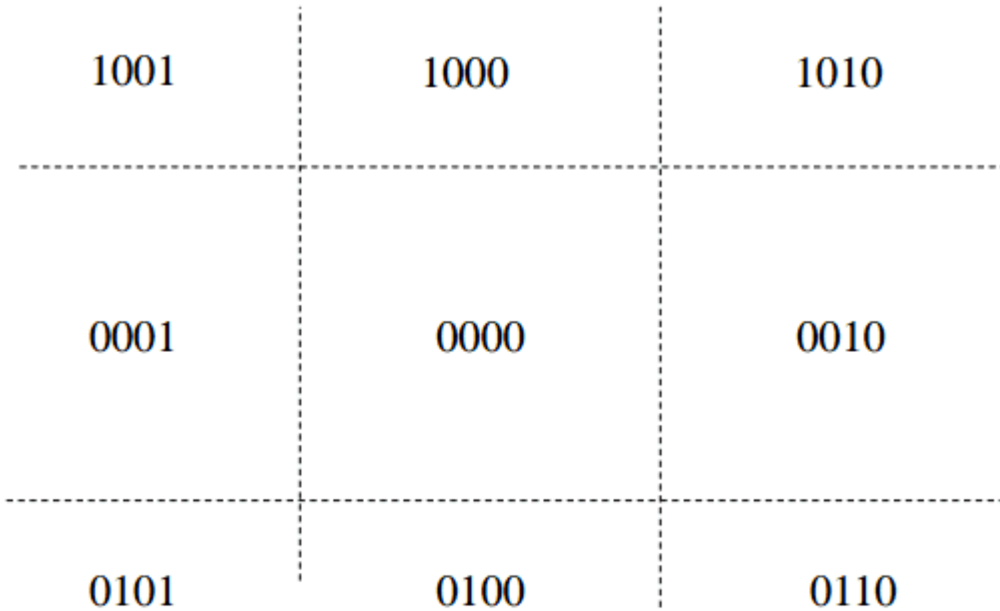
Input: The screen and a 2D line segment

(let's start with 2D first)

Output: Clipped line segment

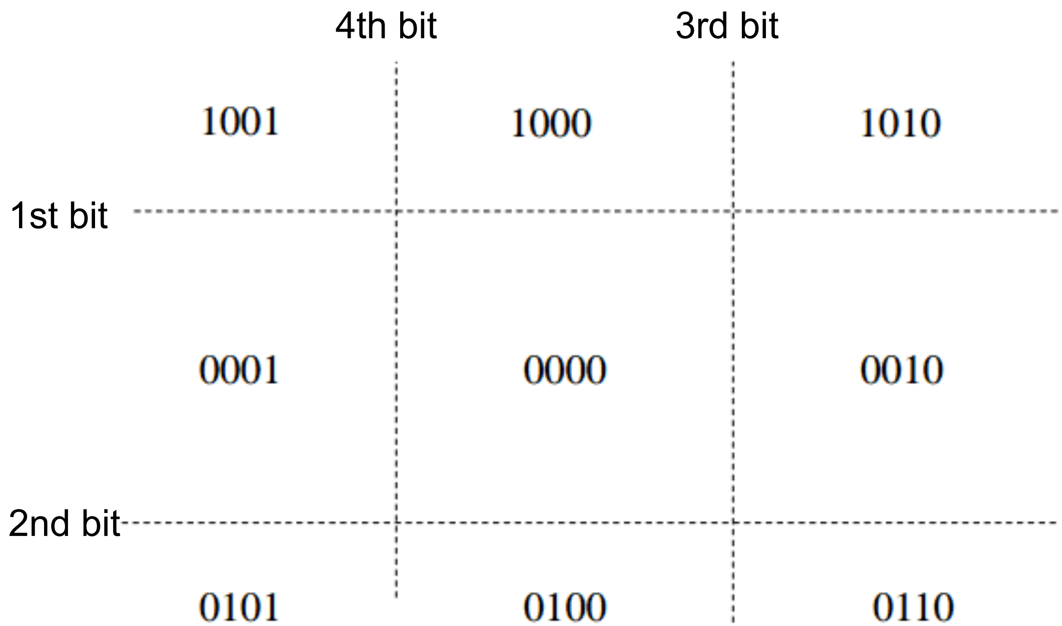


# Cohen-Sutherland 2D outcodes




- The whole space is split into 9 regions
- Only the center region is visible
- Each region is encoded by four bits

# Cohen-Sutherland 2D outcodes




– 4-bit code called: *Outcode*
– First bit : above top of window, $y > ymax$
– Second bit : below bottom, $y < ymin$
– Third bit : to right of right edge, $x > xmax$
– Fourth bit : to left of left edge, $x < xmin$

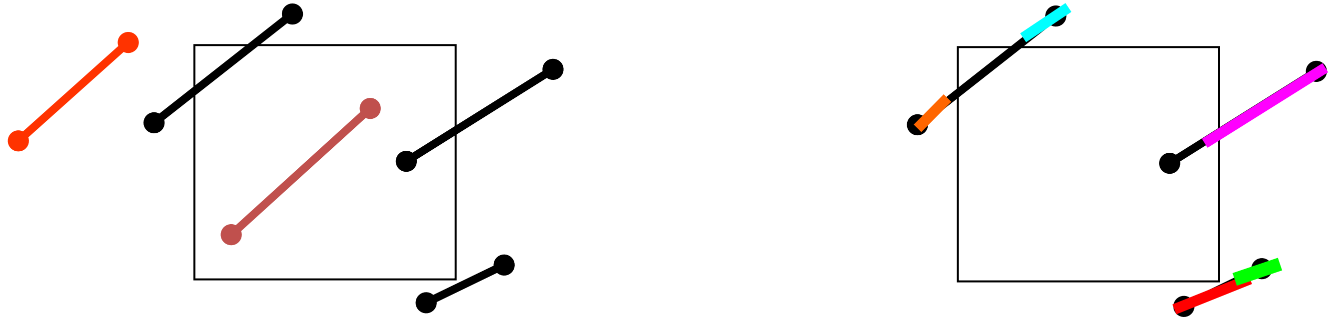
# Cohen-Sutherland algorithm

While (true) {

1. Check if the line segment is trivial accept/reject

2. Otherwise clip the edge and shorten

}



# Recap of AND/OR operators

0 OR 0 = 0

0 OR 1 = 1 (if either is true, true)

1 OR 1 = 1

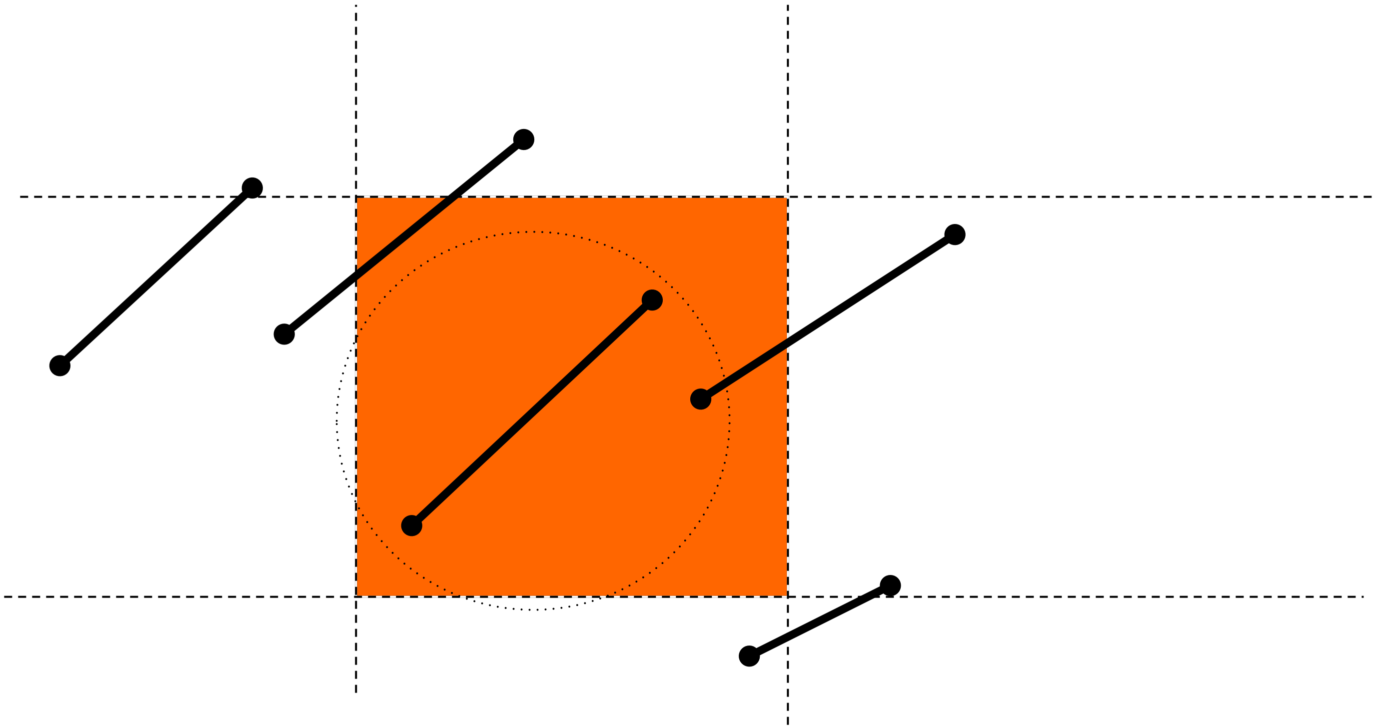
0 AND 0 = 0

1 AND 0 = 0 (if both are true, true)

1 AND 1 = 1

# What is a trivial accept?

- All line vertices lie inside box $\longrightarrow$ accept.
  - Apply an 'OR' operation to the outcodes of two endpoints



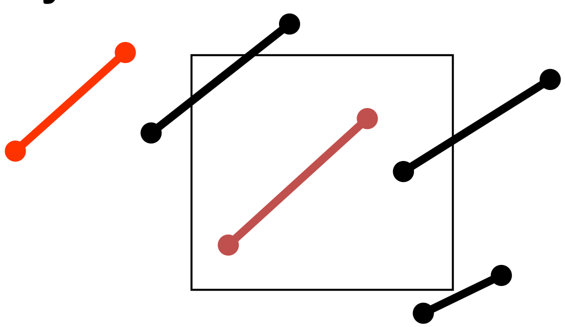
# Cohen-Sutherland algorithm

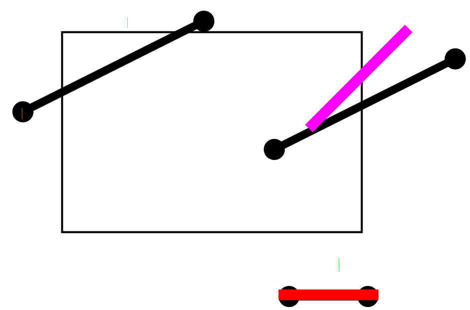
While (true) {

1. Check if the line segment is trivial accept/**reject**

2. Otherwise clip the edge and shorten

}

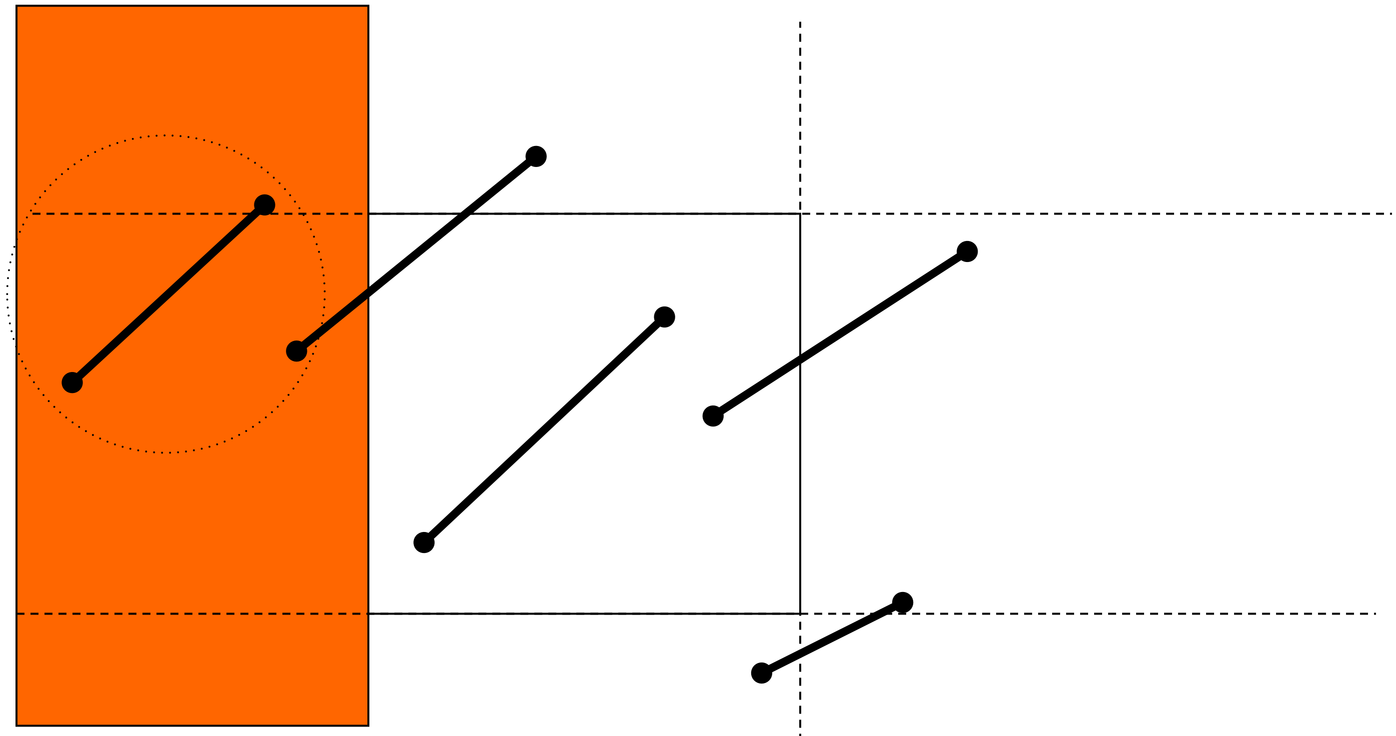




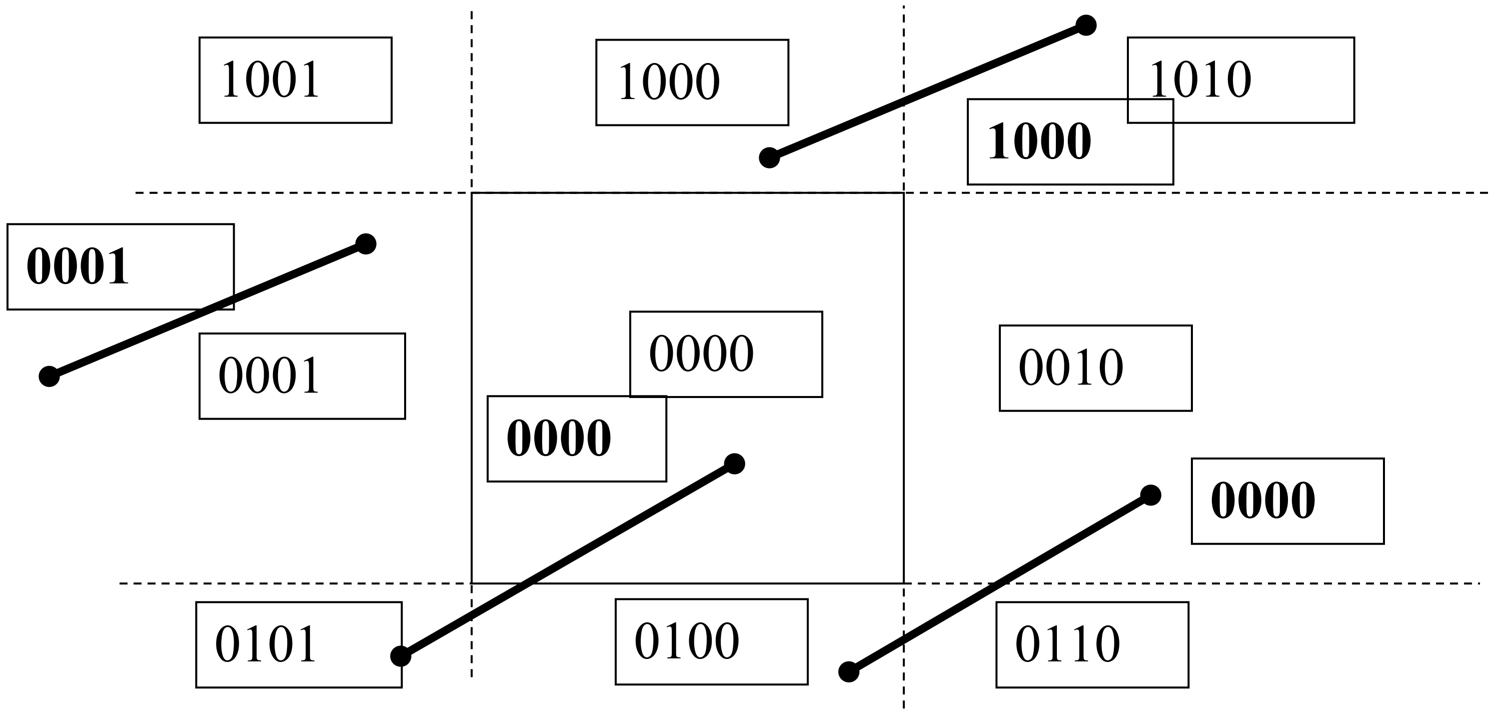
# What is a trivial reject?

All line vertices lie outside and on same side → reject.

- Apply an 'AND' operation to the two endpoints
- If not '0000', then reject

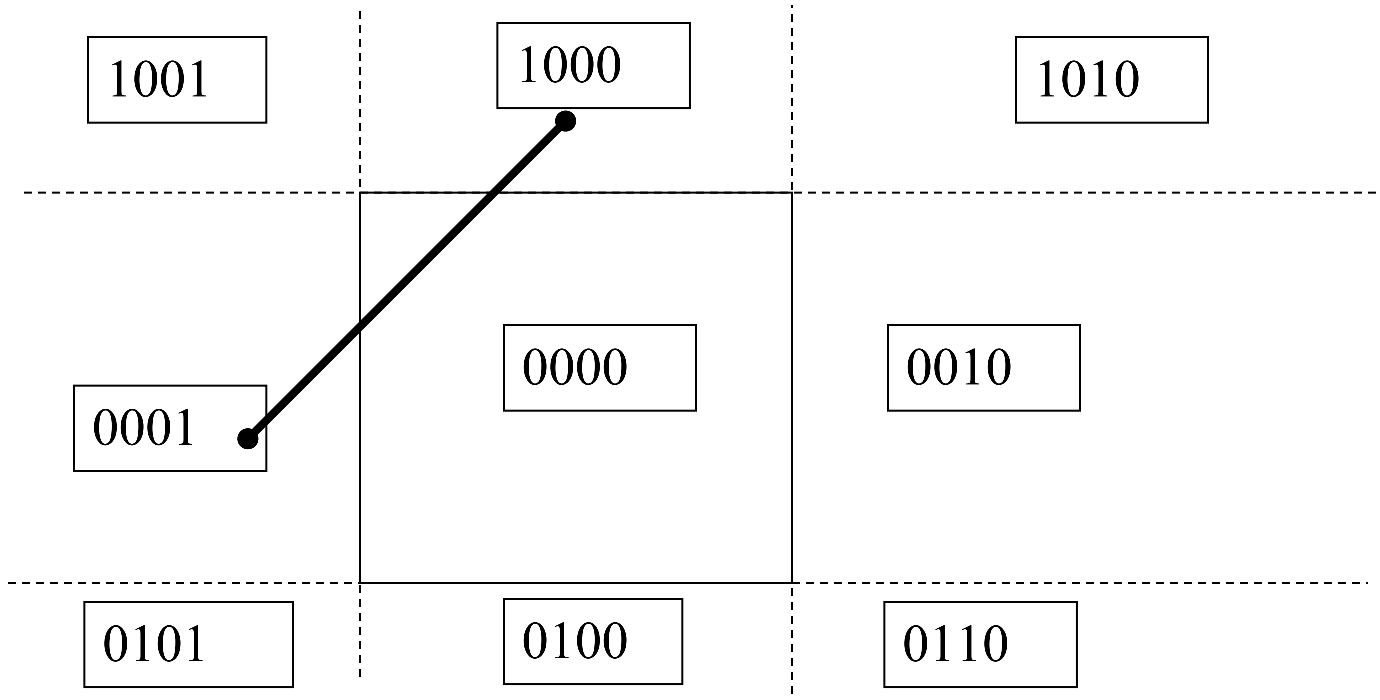


# Cohen-Sutherland 2D outcodes




Logical AND between codes for 2 endpoints,
Reject line if non-zero – trivial rejection.

# What about this one?




Logical AND between codes for 2 endpoints,
Reject line if non-zero – trivial rejection.

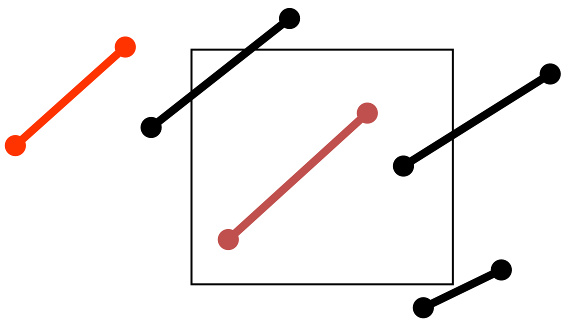
# Cohen-Sutherland algorithm

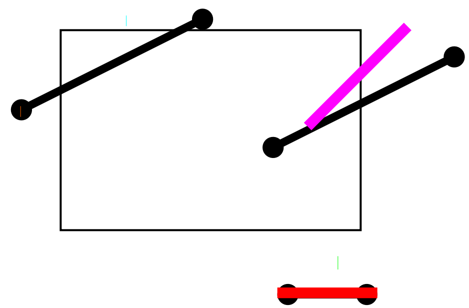
While (true) {

1. Check if the line segment is trivial accept/**reject**

2. **Otherwise clip the edge and shorten**

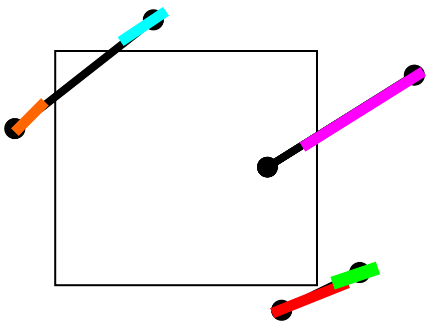
}





# Line Intersection.

- Clip the line by edges of the rectangle
- Select a clip edge based on the outcode, split and feed the  new segment on the side of the rectangle back into algorithm
- Need to perform 4 intersection checks for each line.

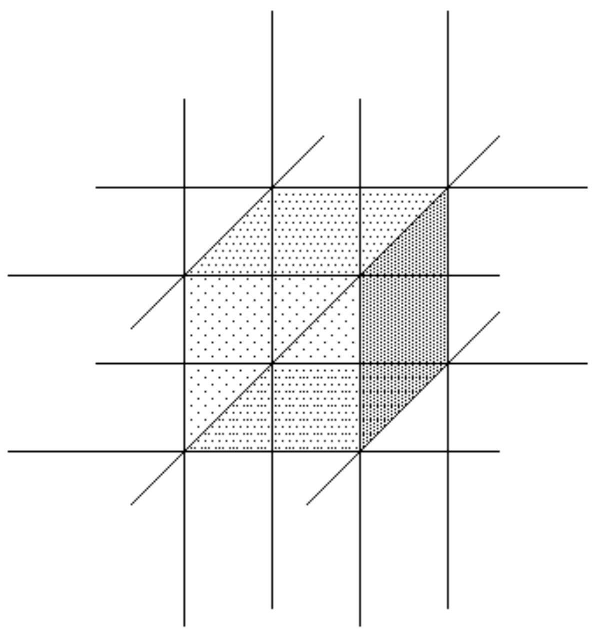


# Cohen-Sutherland algorithm

- How to extend to 3D?
  - Also clipping the lines using front / back planes
- How many bits needed for the outcode?

# Cohen-Sutherland algorithm

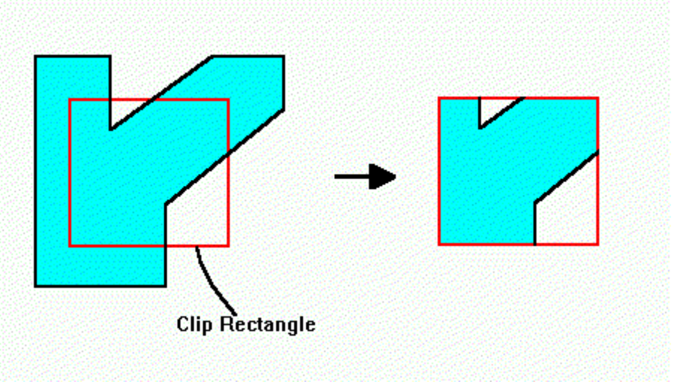
- How to extend to 3D?
  - Also clipping the lines using front / back planes
- How many bits needed for the outcode?



# Polygon Clipping: Sutherland-Hodgman's algorithm

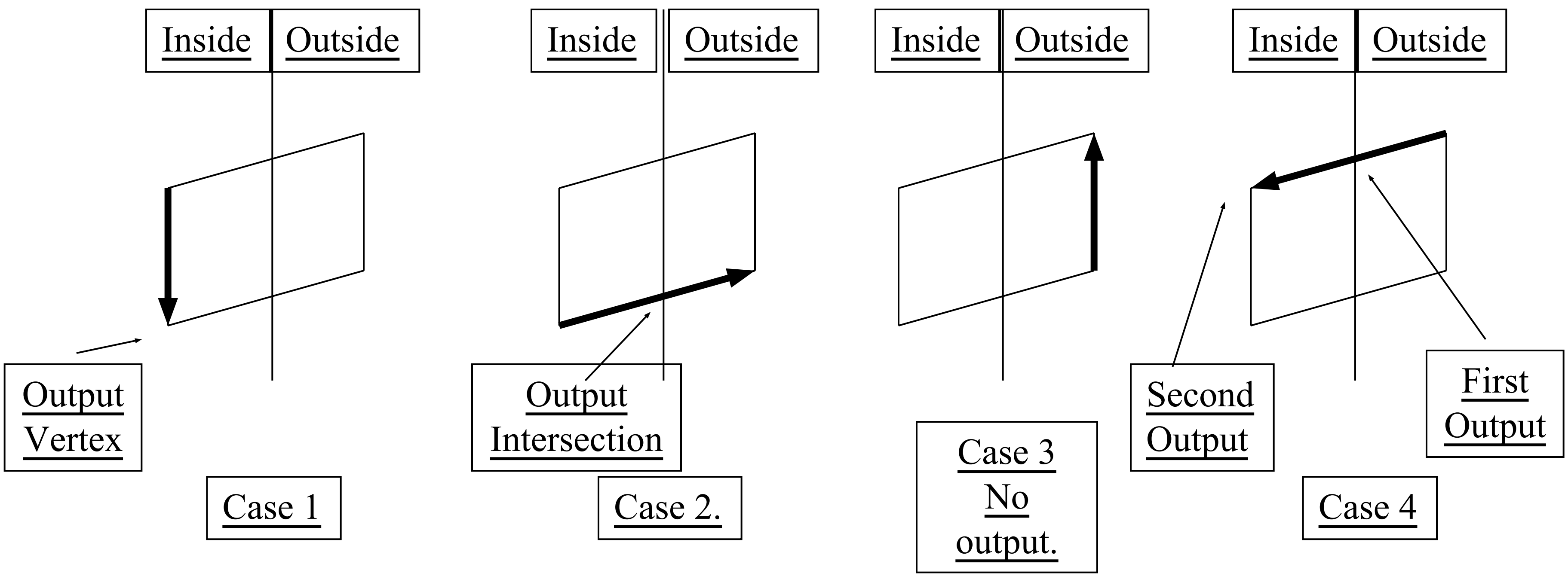
- A systematic approach to clip polygons
- Input : A 2D polygon
- Output : a list of vertices of the clipped polygon

Polygons are clipped at each edge of the window while traversing the polygon



# Sutherland-Hodgman's algorithm

- The edges of the polygon are traversed
- The edges can be divided into four types

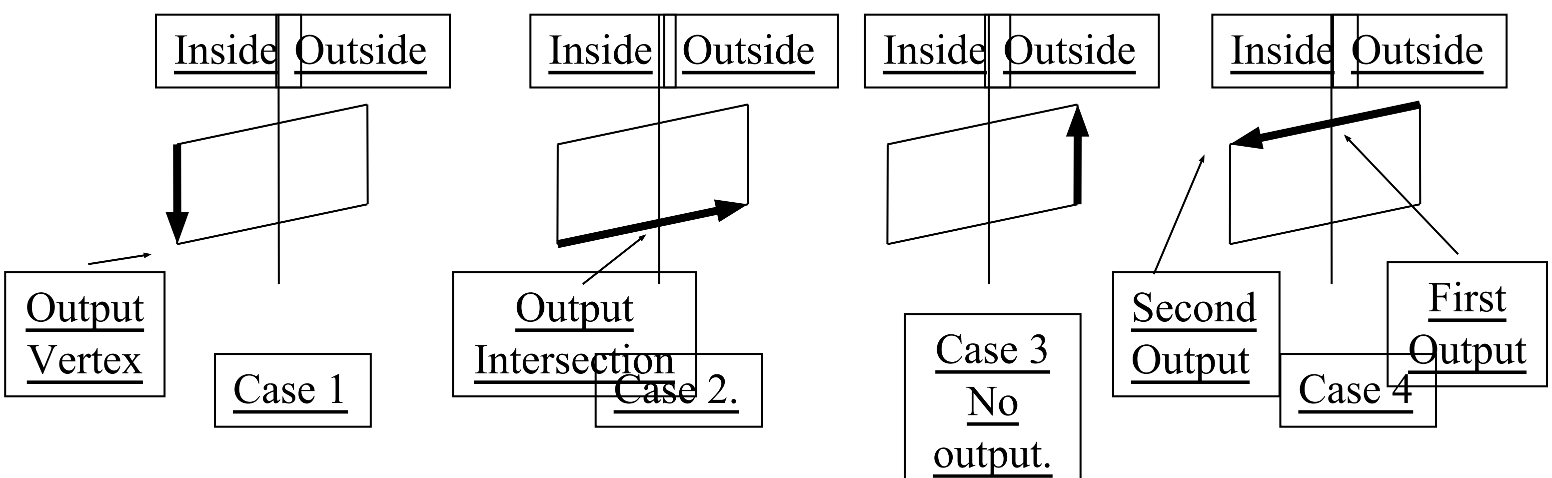


# Sutherland-Hodgman's algorithm

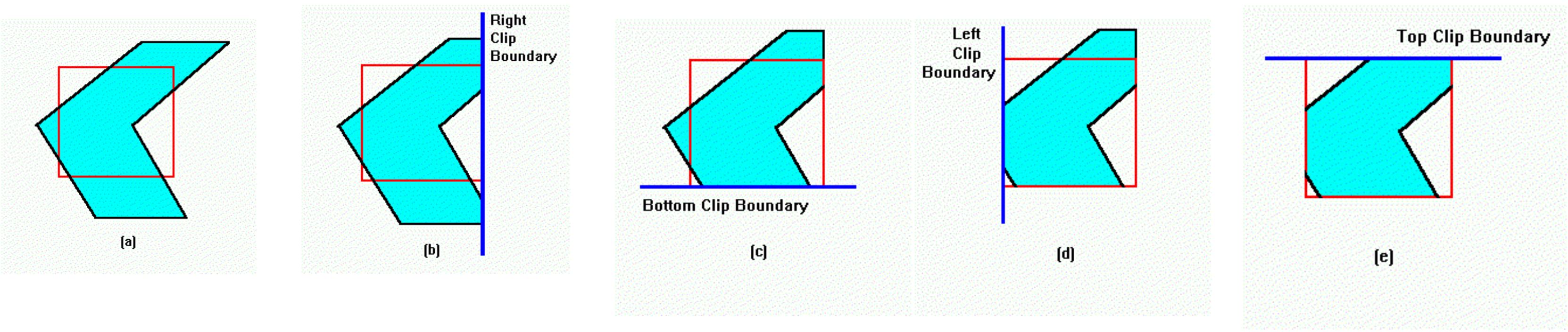
For each edge of the clipping rectangle

For each edge of the polygon (connecting pi, pi+1)

- If case 1  add p+1 to the output
- If case 2 add interaction to output
- If case 4 add intersection and p+1 to output



# Example




(a) (b) (c) (d) (e)

# Sutherland-Hodgman algorithm

- How to extend to 3D?

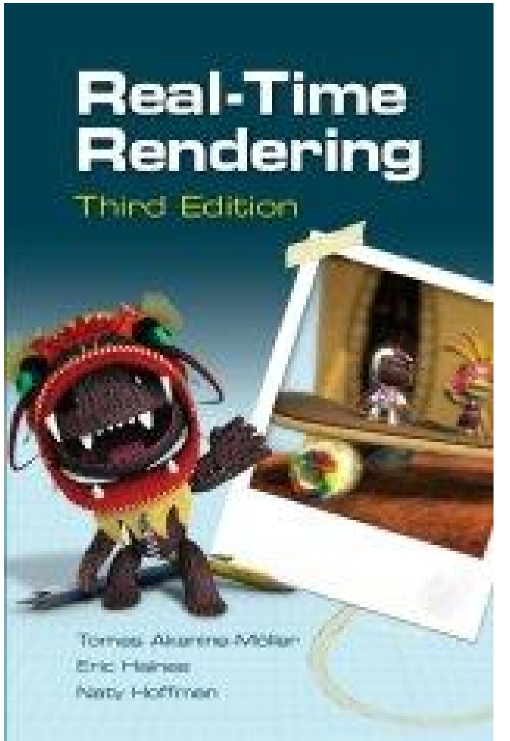
# Summary

- Projection
  - Perspective, parallel (orthographic) projection
  - Canonical view volume
- Clipping
  - Cohen-Sutherland's algorithm
  - Sutherland-Hodgmans's algorithm

# Another Good Modern Textbook

http://www.realtimerendering.com/

Akenine-Moller



# Readings

- Foley et al. Chapter 6 – all of it,
  - Particularly section 6.5
- Introductory text, Chapter 6 – all of it,
  - Particularly section 6.6
- Akenine-Moller, Real-time Rendering Chapter 3.5
- Clipping lines, polygons
  - Foley et al. Chapter 3.12, 3.14
  - http://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/mm.html