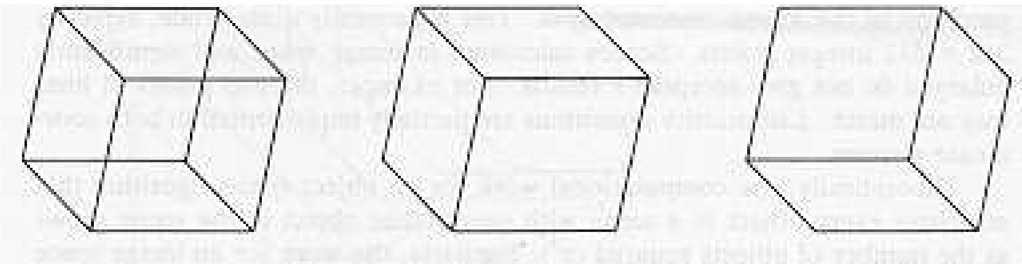
Visible Surface Determination

- Determine which lines, planes, etc. are visible from the center of projection (for perespective) or for a direction of projection (for parallel)



- **Methods**

  - Object precision (or object space) methods

  - Image precision (or image space) methods

- **Object precision methods**

  - Use object descriptions and are performed in the coordinate system in which the objects are described

  - Idea: compare objects directly with each other, eliminating portions of them that are not visible

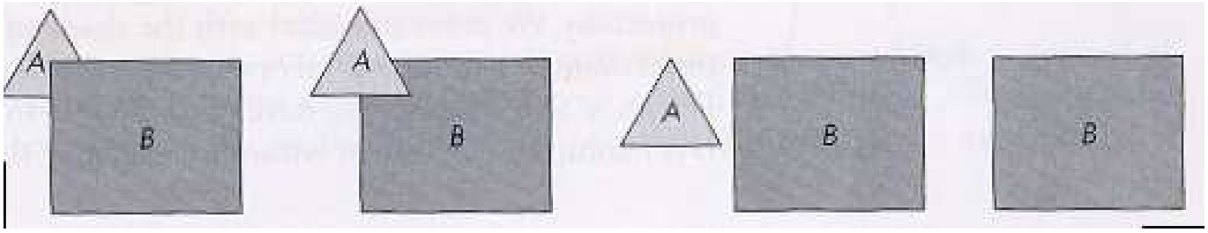
  - Generic object precision algorithm

for (each object in the world) {

determine those parts of the object whose view is unobstructed by other parts of of it or any other object

draw those parts in the appropriate color

}

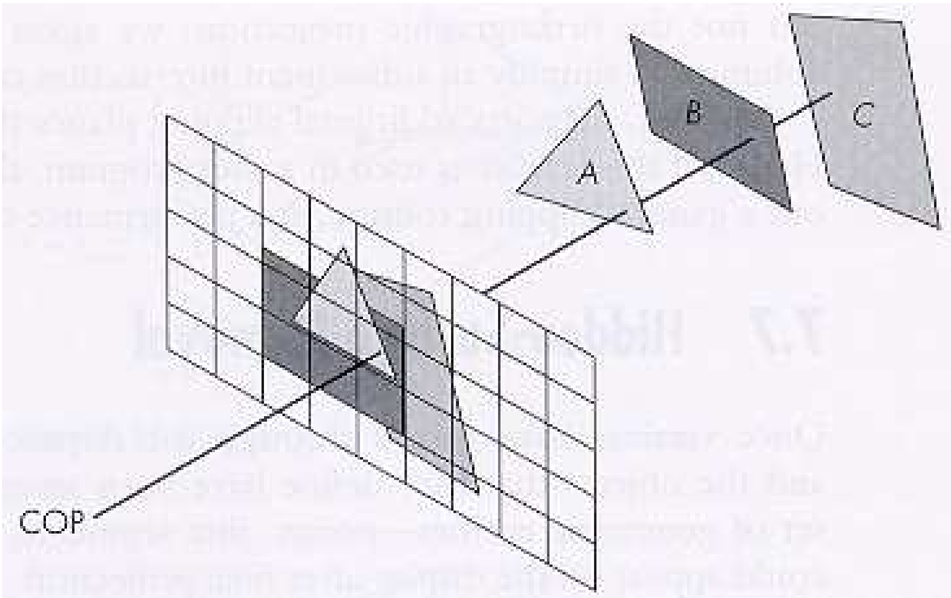


- Cost: $(n-1)+(n-2)+\ldots+1=n^2$, (n: number of objects)

- **Image precision methods**

  - Use projected images of objects and are performed at the resolution of the display device

  - Idea: visibility is determined pixel by pixel on the projection plane



  - Generic image precision algorithm

for (each pixel in the image) {

determine the object closest to the viewer that is pierced by the projector through the pixel

draw the pixel in the appropriate color
}

  - Cost: np, (n: number of objects, p: number of pixels, $p >> n$), worst-case

- **Comparison between object and image precision algorithms**

Image Precision	*Object Precision*
performed at the resolution of the display device	performed at the precision with which each object is defined
if the size of the image changes, visible-surface calculations have to be repeated	if the size of the image changes, visible-surface calculations do not have to be repeated
subject to aliasing (jaggies)	no aliasing problems
initially written for raster devices	initially written for vector devices

- **Improving the efficiency of visible surface determination algorithms**

- Using coherence (i.e., local similarities)

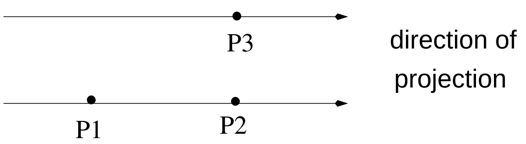
* Usually, images include objects whose properties vary smoothly from one part to another

* Coherence is exploited by reusing calculations made for one part of the picture to other nearby parts (e.g., use an incremental approach)

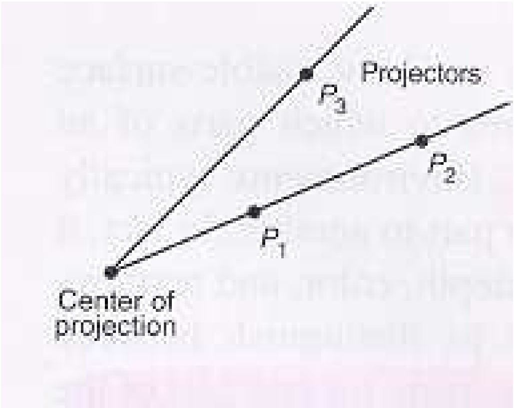
* Examples of coherence: object coherence, face coherence, edge coherence, depth coherence etc.

- Transforming the perspective projection view volume to the parallel projection view volume

Paralell Projection



Perspective Projection

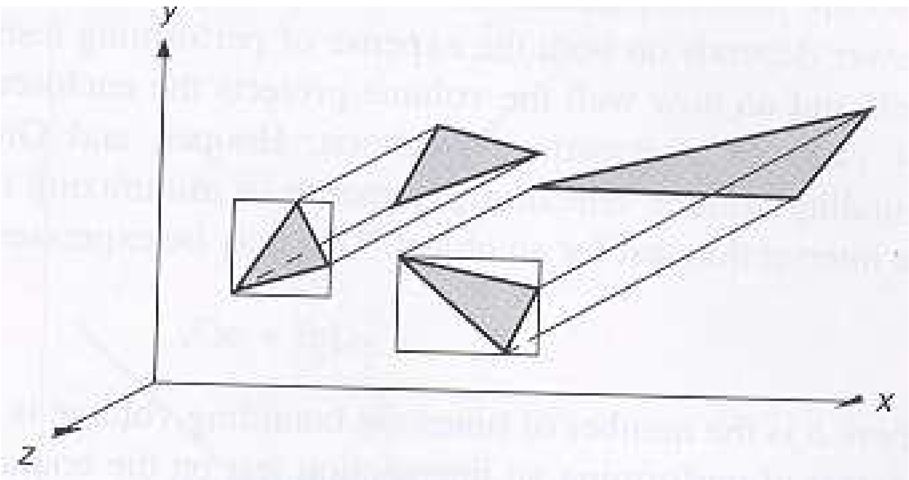


P_1 and P_2 lie on the same projector if x_1=x_2 and y_1=y_2

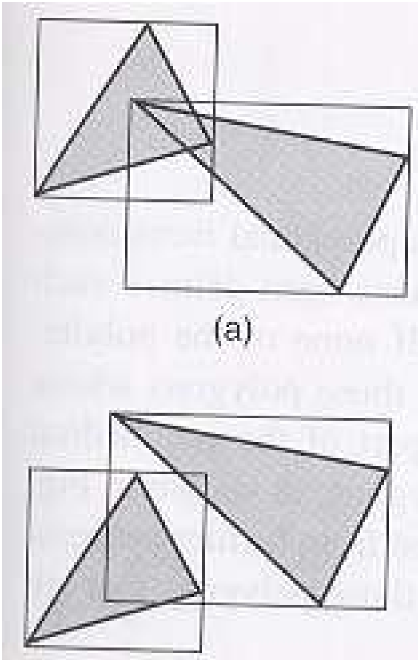
P_1 and P_2 lie on the same projector if x_1/z_1=x_2/z_2 and y_1/z_1=y_2/z_2

- Extents

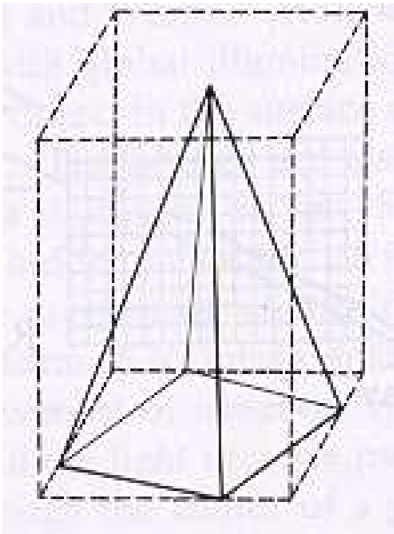
* If the extents of the projections <u>do not overalp,</u> no comparisons need to be made



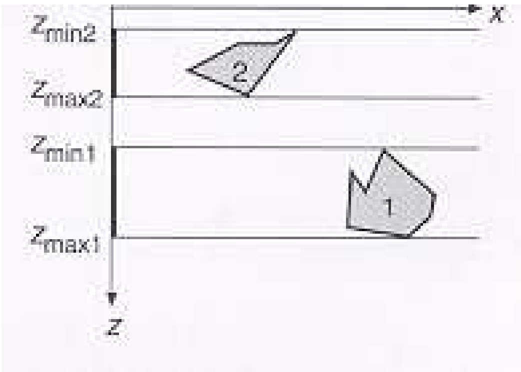
* If the extents overlap, more testing is needed



- Bounding volumes



- <u>Minmax testing:</u> bounding each dimension seperately (*minmax testing*)

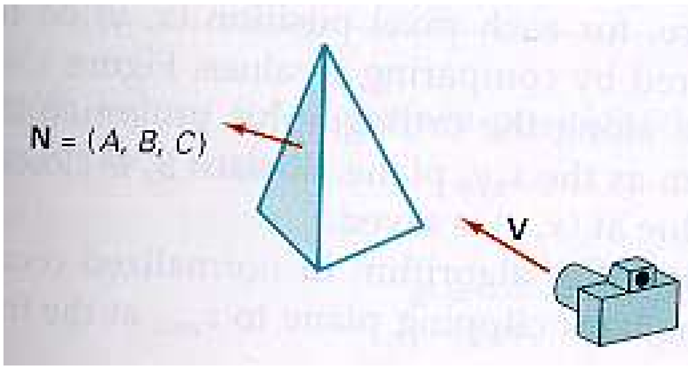


* No overlap if: $z_{max2} < z_{min1}$ or $z_{max1} < z_{min2}$

- Back-face culling

  * Eliminate back-facing polygons

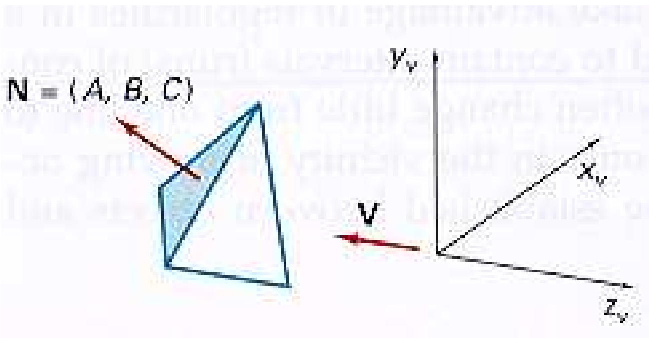
  * Surface normals of back-facing polygons point away from the observer



* <u>Test:</u> backface if $V . N \geq 0$

* Test after normalizing transformation:

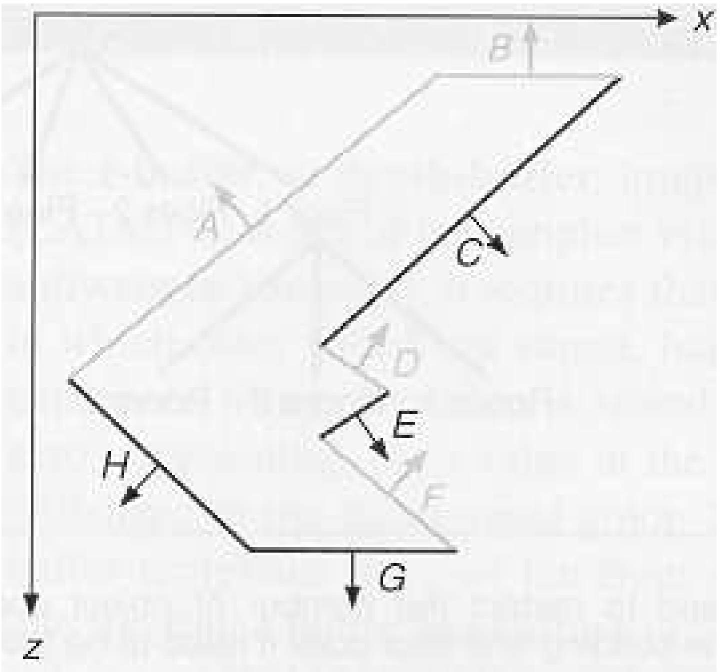
$$V.N=(0,0,-1)(A,B,C)=-C, \text{ thus, if } C<0, \text{ then } V.C>0$$



* Convex polyhedra: no further tests are required

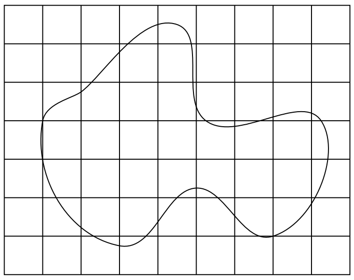
* Concave polyhedra: front faces may be partially obscured

* Back-face culling halves, on the average, the number of polygons to be considered by the visible surface determination algorithm

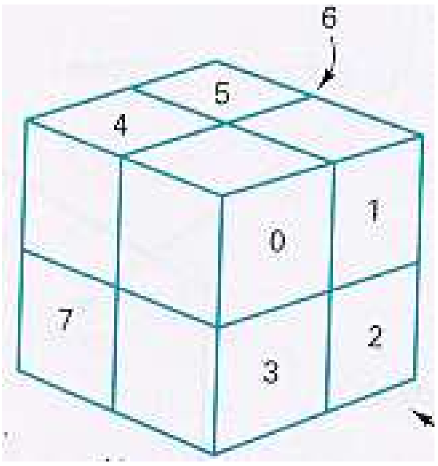


- Spatial partitioning

  * Divide the projection plane with a coarse, regular 2D rectangular grid and determine in which grid spaces each object's projection lies



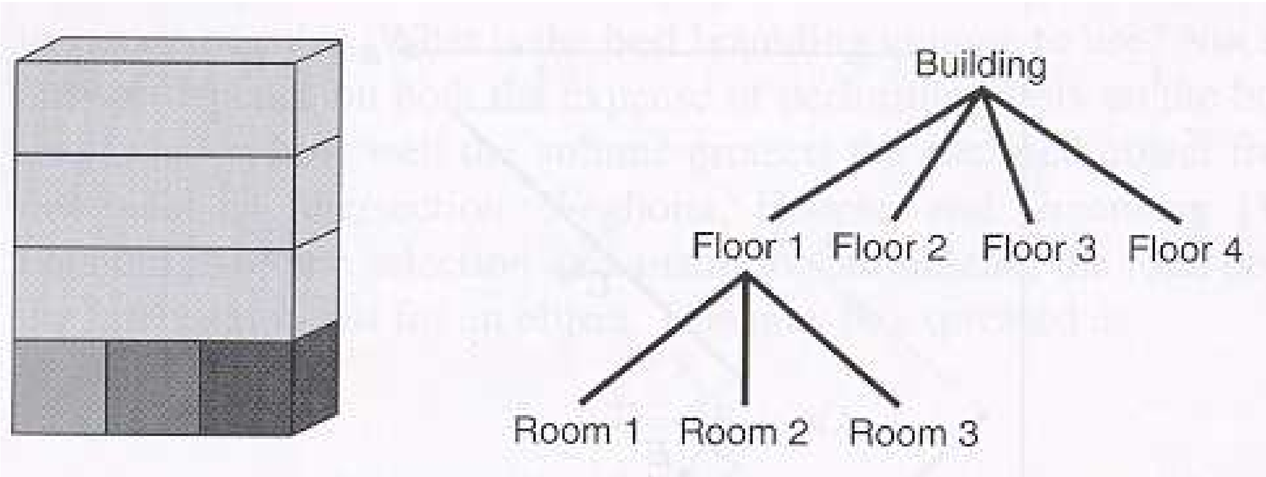
  * Impose a regular 3D grid on the objects in the environment



- Hierarchy

* An object on one level of the hierarchy can serve as an extent for its children if they are entirely contained within it

* Example: test for intersections with rooms 1, 2, and 3, only if a projector intersects the building amd floor 1



- **Algorithms to be discussed**

  - Z-buffer algorithm

  - Scan-line algorithm

  - Ray-tracing algorithm