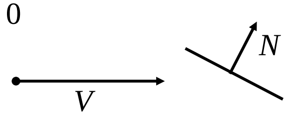
Visible-Surface Algorithms

- Given a set of 3D objects and a viewing specification, determine which lines or surfaces of the objects should be visible
- *Image-precision* algorithms: determine what is visible at each pixel
- *Object-precision* algorithms: determine which parts of each object are visible

Back-Face Culling

- If objects are represented by closed surfaces, polygons facing away from the viewer are always hidden and can be eliminated.
- Back-face test: $V \cdot N > 0$



- Back-face culling solves the hidden surface removal problem for a certain class of objects. What is this class?

Ray Casting

- For each pixel, generate the line of sight (ray) from the center of projection passing through the pixel.
- To find the surface visible through the pixel:
  - Intersect ray with all surfaces in the scene
  - Return intersection closest to the center of projection

The Z-Buffer Algorithm (Catmull 1974)

- In addition to the frame buffer, keep a Z-buffer of the same dimensions containing the depth value of each pixel.
- Initialize frame buffer to background color, and the Z-buffer to the depth of the far clipping plane.
- Scan-converted all polygons in an arbitrary order:
  - For each pixel (x,y) covered by the polygon, incrementally compute its color *C*, as well as its depth Z.
  - If Z < Z-buffer(x,y) then FrameBuffer(x,y) := *C*;
    Z-buffer(x,y) := Z

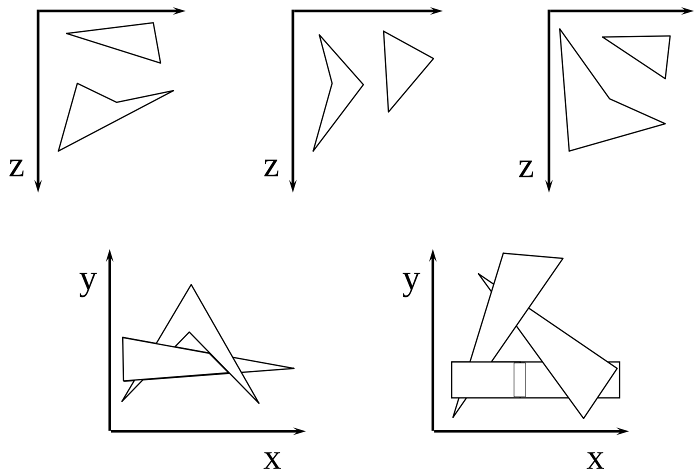
The Z-Buffer Algorithm

- Advantages:
  - Simple and easy to implement both in software and in hardware
  - Separately rendered images can be composited using their Z-buffers
- Disadvantages:
  - Requires a lot of memory (not so much of a problem anymore)
  - Finite depth precision can cause problems
  - Might spend a lot of time rendering polygons that are not visible in the image

List-Priority Algorithms

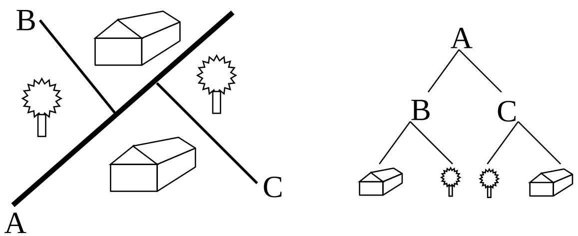
- Determine an ordering for objects ensuring that a correct picture results if objects are drawn in that order.
- Example: *painter's algorithm*. If all of the polygons in the scene are sorted by their depth, drawing them ____ to ____ will give the correct result.
- Question: does a depth ordering always exist?

Depth Ordering



The BSP tree

- BSP = Binary Space Partitioning
- Interior nodes correspond to partitioning planes:



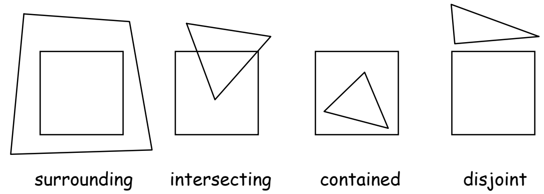
- Leaf nodes correspond to convex regions of space.

The BSP-tree algorithm

- Construct a BSP tree:
  - Pick a polygon, let its supporting plane be the root of the tree.
  - Create two lists of polygons: those in front, and those behind (splitting polygons as necessary)
  - recurse on the two lists to create the two sub-trees.
- Display:
  - Traverse the BSP tree back to front, drawing polygons in the order they are encountered in the traversal.

Warnock's (1969) Area Subdivision Algorithm

- Subdivide screen area recursively, until visible surfaces are easy to determine.
- Each polygon has one of four relationships to the area of interest:

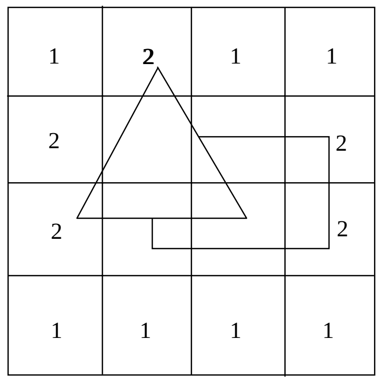


Warnock's Algorithm

- If all polygons are disjoint from the area, fill area with background color.
- Only one intersecting or contained polygon: first fill area with background color, then scan convert polygon.
- Only one surrounding polygon: fill area with polygon's color.
- More than one polygon is surrounding, intersecting, or contained, but one surrounding polygon is in front of the rest: fill area with polygon's color.
- If none of the above cases occurs, subdivide area into four, and recurse.

Warnock's Algorithm

- When the resolution of the image is reached polygons are sorted by their Z-values at the center of the pixel, and the color of the closest polygon is used.



Weiler-Atherton (1977) Area Subdivision Algorithm

- Subdivides screen area along polygon boundaries instead of along rectangular boundaries.
- Sort polygons by their nearest Z value.
- Clip all polygons into two lists: "inside" and "outside" the clip polygon.
- "Inside" polygons behind the clip polygon are invisible.

Coherence

- Most visible-surface algorithms attempt to utilize *coherence* - the degree to which parts of a scene exhibit local similarities.
- Possible kinds of coherence are:
  - object coherence
  - face coherence
  - edge coherence
  - scan-line coherence
  - area coherence
  - depth coherence
  - frame coherence