BSP Tree

Based on division into clusters and observations, that if clusters can be separated by a plane, clusters on the same side of the plane as the viewpoint can obscure, but cannot be obscured by, clusters on the opposite side of the plane.

We can recursively divide scene into subregions by inserting planes and splitting polygons, then represent that subdivision with a binary tree, the BSP tree:

\[
\begin{array}{c}
\text{P1} \\
\text{P2} \\
\text{P3} \\
\text{Q} \\
\end{array}
\]

To use BSP, traverse inorder and at each internal node determine which side of dividing plane viewpoint lies on. If in front, recursively process all polygons behind plane, then any polygon on plane, then recursively all polygons in front of plane. If viewpoint behind plane, recursively process front cluster, then plane, then back cluster.
e.g. Suppose we use polygon 3, 2, 4 to define locations of planes to subdivide scene into BSP

Given viewpoint V1, we start at root and:

@3, V1 in front so do B, 3, F
@4, V1 in back so do F, 4, B
⇒ 4, 5b
⇒ 3
@2, V1 in back so do F, 2, B
⇒ 5a, 2, 1

So order is 4, 5b, 3, 5a, 2, 1
Suppose viewpoint moved to V2.

C 3, in back so F, 3, B
C 2, in back so F, 2, B
  ⇒ 5a, 2, 1
  ⇒ 3
C 4, in front so B, 4, F
  ⇒ 5b, 4

So order is 5a, 2, 1, 3, 5b, 4

BSP trades initial setup cost against fast depth ordering for arbitrary viewpoint, good if scene is static.