Z-buffer

The Z-buffer or depth buffer algorithm [Catmull, 1974] is probably the simplest and most widely used.

Here is pseudocode for the Z-buffer hidden surface algorithm:

for each pixel \((i,j)\) do
    \[ Z\text{-buffer} [i,j] \leftarrow \text{FAR} \]
    \[ \text{Framebuffer}[i,j] \leftarrow \text{<background color>} \]
end for

for each polygon \(A\) do
    for each pixel in \(A\) do
        Compute depth \(z\) and shade \(s\) of \(A\) at \((i,j)\)
        if \(z > Z\text{-buffer} [i,j]\) then
            \[ Z\text{-buffer} [i,j] \leftarrow z \]
            \[ \text{Framebuffer}[i,j] \leftarrow s \]
        end if
    end for
end for

Q: What should FAR be set to? 

Far clipping plane depth
- big number
Z-buffer: Analysis

- Easy to implement?
- Easy to implement in hardware?
- Incremental drawing calculations (uses coherence)?
- Pre-processing required?
- On-line (doesn’t need all objects before drawing begins)?
- If objects move, does it take more work than normal to draw the frame?
- If the viewer moves, does it take more work than normal to draw the frame?
- Typically polygon-based?
- Efficient shading (doesn’t compute colors of hidden surfaces)?
- Handles transparency?
- Handles refraction?

\[
\alpha_A A + (1-\alpha_A) (\alpha_B B + (1-\alpha_B) C) \\
\neq \\
\alpha_B B + (1-\alpha_B) (\alpha_A A + (1-\alpha_A) C)
\]