A-BUFFER METHOD

An extension of the ideas in the depth-buffer method is the A-buffer method (at the other end of the alphabet from "z-buffer", where ***z*** represents depth). The **A** buffer method represents an antialiased, ***area-averaged, accumulation-buffer*** method developed by Lucasfilm for implementation in the surface-rendering system called REYES (an acronym for "Renders Everything You Ever Saw").

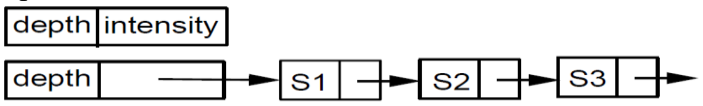
A drawback of the depth-buffer method is that it can only find one visible surface at each pixel position. In other words, it deals only with opaque surfaces and cannot accumulate intensity values for more than one surface, as is necessary if transparent surfaces are to be displayed**.** The A-buffer method expands the depth buffer so that each position in the buffer can reference a linked list of surfaces. Thus, more than one surface intensity can be taken into consideration at each pixel position, and object edges can be antialiased.

Each position in the A-buffer has two fields:

- depth field
    - stores a positive or negative real number
- intensity field
    - stores surface-intensity information or a pointer value.

If the depth field is positive, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RCB components of the surface color at that point and the percent of pixel coverage.

If the depth field is negative, this indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data



Data for each surface in the linked list includes:

- RGB intensity components
- opacity parameter (percent of transparency)
- depth
- percent of area coverage
- surface identifier
- other surface-rendering parameters
- pointer to next surface

The A-buffer can be constructed using methods similar to those in the depth-buffer algorithm. Scan lines are processed to determine surface overlaps of pixels across the individual scanlines. Surfaces are subdivided into a polygon mesh and clipped against the pixel boundaries. Using the opacity factors and percent of surface overlaps, we can calculate the intensity of each pixel as an average of the contributions from the overlapping surfaces.