

Hidden Line Rendering with a Modified A-Buffer

Mayur Patel
3/27/00

G-Buffer Rendering (Saito et al SIGGRAPH 1990)

Saito et al describe a technique whereby depth maps and maps of surface normals are produced from an existing renderer. Edge detection is performed on these maps and the resulting line image can be used in non-photorealistic rendering applications.

The technique suffers from aliasing:

A 3x3 edge detection kernel produces 2-pixel lines. In order to produce properly anti-aliased output, the pipeline must over-sample and filter down.

The technique suffers from missing information:

Use of maps gives incentive to limit the scope of the information available to the line detector and to limit the precision of the information. The line detector typically lacks the resources to cope with special cases, such as flat surfaces orthogonal to the view vector or surfaces of high curvature. Functional G-buffer line detectors are either very complicated or require extensive user intervention.

Using A-Buffer Coverage Masks for Line Detection

- First derivatives and other data

are available to the line detector, not just P and N.

- Anti-aliasing can be performed on-line at low cost.
- Line shading architectures can be implemented which access traditional shading parameters.
- Reasonably easy to implement relative to analytical solutions.
- One-pass rendering of surfaces and lines can be executed.

Caveats:

- Efficient implementation requires use of extended coverage masks.
- As the precision of the extended coverage mask increases the effective anti-aliasing improves; however, a 16b extended mask is equivalent to a 4b standard mask.

Using 16b extended coverage masks:

```
unsigned short
GetHalo( unsigned short mask16b )
{
    unsigned short m;
    m = mask16b;
    m |= (mask16b & 0x7777) << 1;
    m |= (mask16b & 0xEEEE) >> 1;
    m |= mask16b << 4;
    m |= mask16b >> 4;
    return ( m ^ mask16b );
}

unsigned short
GetEdgeBetween(
    unsigned short mFrontMask,
    unsigned short mBackMask
) {
    unsigned short mHFront;
    unsigned short mHBack;
    mBackMask &= ~mFrontMask;
    mHFront = GetHalo( mFrontMask );
    mHBack = GetHalo( mBackMask );
    mHFront &= mBackMask;
    mHBack &= mFrontMask;
    return( mHFront | mHBack );
}
```