Determinación de la Superficie Visible

3 de noviembre de 2003

Dado un conjunto de objetos 3D y una especificación de una vista, se quiere determinar cuáles líneas o superficies de los objetos son visibles, ya sea donde el *centro de proyección* (para proyecciones en perspectiva) o a lo largo de la *dirección de proyección* (para proyecciones paralelas), de manera que se puedan desplegar sólo las líneas o superficies visibles. Este proceso se conoce como *determinación de la superficie visible* ó *eliminación de líneas ocultas* ó *superficies ocultas*. En la determinación de líneas visibles, las líneas se asumen que son bordes de superficies opacas que pueden obscurecer los bordes de otras superficies más lejanas del observador. Por lo tanto, al proceso general se refiere como *determinación de la superficie visible*.

Aunque la idea es simple, su realización requiere significante poder de procesamiento, y consecuentemente involucra grandes cantidades de tiempo de cómputo sobre máquinas convencionales.

La primera aproximación determina cuál de n objetos es visible en cada pixel de la imagen:

for(cada pixel en la imagen) {

Determine el objeto más cercano al observador que es atravesado por el proyector que pasa por el pixel;

Dibujar el pixel con el color apropiado;

}

Una aproximación de fuerza bruta para un pixel requiere examinar todos los n objetos para determinar cuál está más cercano al observador a lo largo del proyector que pasa a través del pixel. Para p pixels, el esfuerzo es proporcional a np, donde p es aproximadamente igual a 1 millón para displays de alta resolución.

La segunda aproximación es comparar los objetos directamente uno con otro, eliminando los objetos o porciones de ellos, que no son visibles:

for(cada objeto en el mundo) {

Determine aquellas partes del objeto cuyas vistas no están obstruidas por otras partes de él o cualquier otro objeto;

Dibujar aquellas partes con el color apropiado;

}

Para n objetos, el esfuerzo computacional es proporcional a n^2. Aunque es superior que el primer método si $n < p$, típicamente sus pasos individuales son más complejos y más consumidores de tiempo y es frecuentemente más lento y de más difícil realización.

Cada aproximación se le conoce como de *precisión en imagen* y *precisión en objetos*, respectivamente.

1 El algoritmo de buffer z

Este es uno de los algoritmos para determinar las superficie visible más fáciles de implantar, tanto en hardware como en software. Es un algoritmo de precisión en imagen. Este requiere, además del buffer de memoria para almacenar los colores de la imagen, otro buffer, llamado *zeta*, del mismo tamaño, donde se almacena el valor z de cada pixel. El buffer z se inicializa a cero, representando el valor z del plano de corte de atrás, y el buffer de color es inicializado al color del fondo. El valor más grande que puede almacenarse en el buffer z representa el valor de z del plano de corte frontal.

```
void zBuffer( void )
{
  int x, y;

  /** Limpia el buffer z **/
  for ( y=0; y<YMAX; y++ ) {
    for ( x=0; x<XMAX; x++ ) {
      WritePixel( x, y, BACKGROUND_VALUE);
      WriteZ( x, y, 0 );
    }
  }

  /** Dibuja los polígonos **/
  for ( cada polígono ) {
    for ( cada píxel en la proyección del
      polígono ) {

      double pz = valor z del polígono
        en las coordenadas (x, y);

      if ( pz >= ReadZ( x, y) ) {
        WriteZ( x, y, pz );
        WritePixel( x, y, color del
          polígono en las
          coordenadas (x, y ) );
      }
    }
  }
}
```