CAPITULO 3

Métodos de Detección de Superficies Visibles

3.1 Introducción

Los algoritmos de detección de superficies visibles se pueden clasificar en dos tipos:

1.- Métodos de espacio-objeto.
2.- Métodos de espacio-imagen.

En 1. se determina en la escena que objeto está más cerca de la cámara y por lo tanto será visible.

En 2. se trabaja con la coordenada z píxel a píxel en la imagen para determinar y clasificar la cercanía del objeto.

Ecuaciones de un plano y normal (preliminar)

La ecuación para un plano está dada por:

$$Ax + By + Cz + D = 0$$

Donde (x,y,z) es cualquier punto en el plano.
Los coeficientes *A, B, C, D* pueden determinarse de un conjunto de ecuaciones para puntos no-colineales en el plano.

Se pueden seleccionar tres vértices de un polígono y determinar:

$$\left(\frac{A}{D}\right) * x_k + \left(\frac{B}{D}\right) * y_k + \left(\frac{C}{D}\right) * z_k = -1 \qquad k = 1,2,3$$

La solución es por regla de Cramer:

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

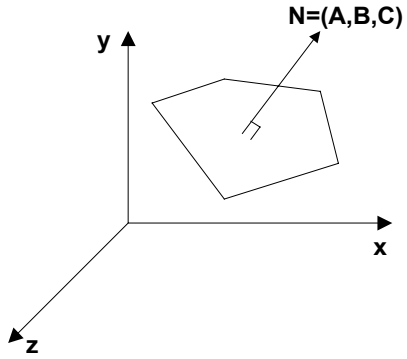
$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Expandiendo las determinantes se tiene:

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

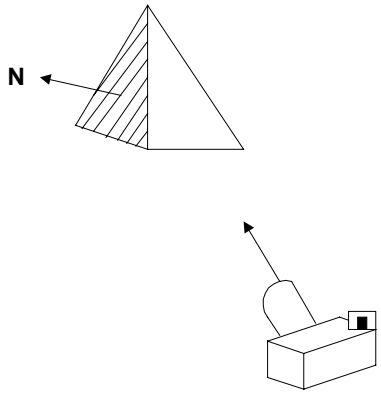
$$D = -x_1(y_2 * z_3 - y_3 * z_2) - x_2(y_3 * z_1 - y_1 * z_3) - x_3(y_1 * z_2 - y_2 * z_1)$$

La normal al polígono en cuestión tiene componentes (*A,B,C*) que son los coeficientes del plano recién encontrado.



3.2 Detección de cara de atrás (Back-Face detection)

Se considera la normal *N* y un vector *V* en la dirección que se mira al objeto, según figura:



Es visible si $V*N<90º$ (en radianes)
Esto es si $V*N>0º$
Es no visible si $V*N<0º$

Si *V* está en dirección del eje z_v, entonces $V=(0,0,V_z)$ y

$$V * N = V_z * C$$

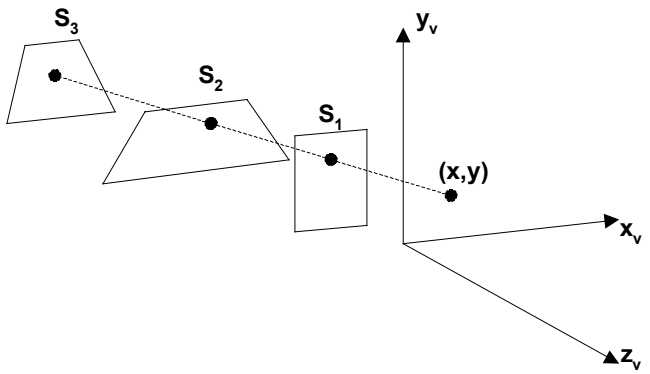
Por lo que sólo se necesita considerar el signo de *C* para determinar visibilidad.
Basta mirar la componente de z en la normal *N* y ver si:

$$C \leq 0$$

Para decir que no es visible.

3.3 Método Buffer de Profundidad

Se comparan los objetos por la distancia de su coordenada z como se muestra en la figura.

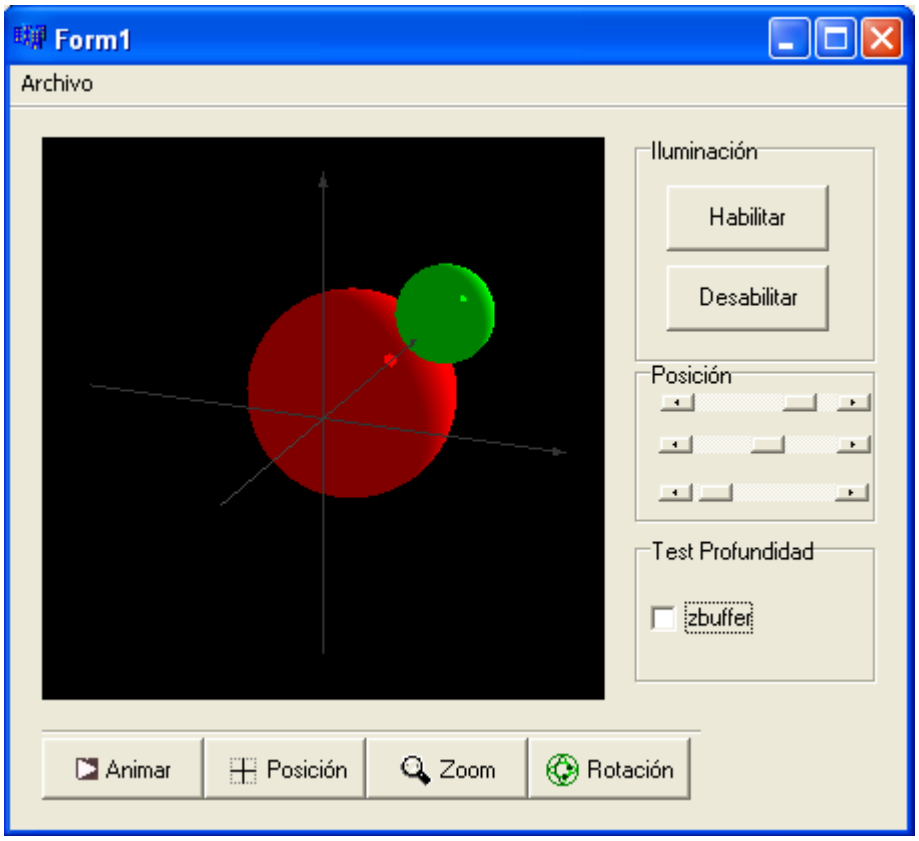


En este caso se guarda el punto S_1.
Donde:

$$z = \frac{-Ax - By - D}{C}$$ para un plano.

Ejemplo Utilizando OpenGl

Opengl incorpora el método del Z-Buffer para poder discriminar que objetos están detrás de otros, sin embargo, esta propiedad es necesario habilitarla. Normalmente si creamos una escena con varias imágenes, la visualización no reflejara si un objeto esta detrás de otro. En la siguiente figura se han dibujado dos esferas y no se ha especificado la condición de test de profundidad a OpenGL (z-Buffer), el resultado es una imagen que carece de la sensación de profundidad .

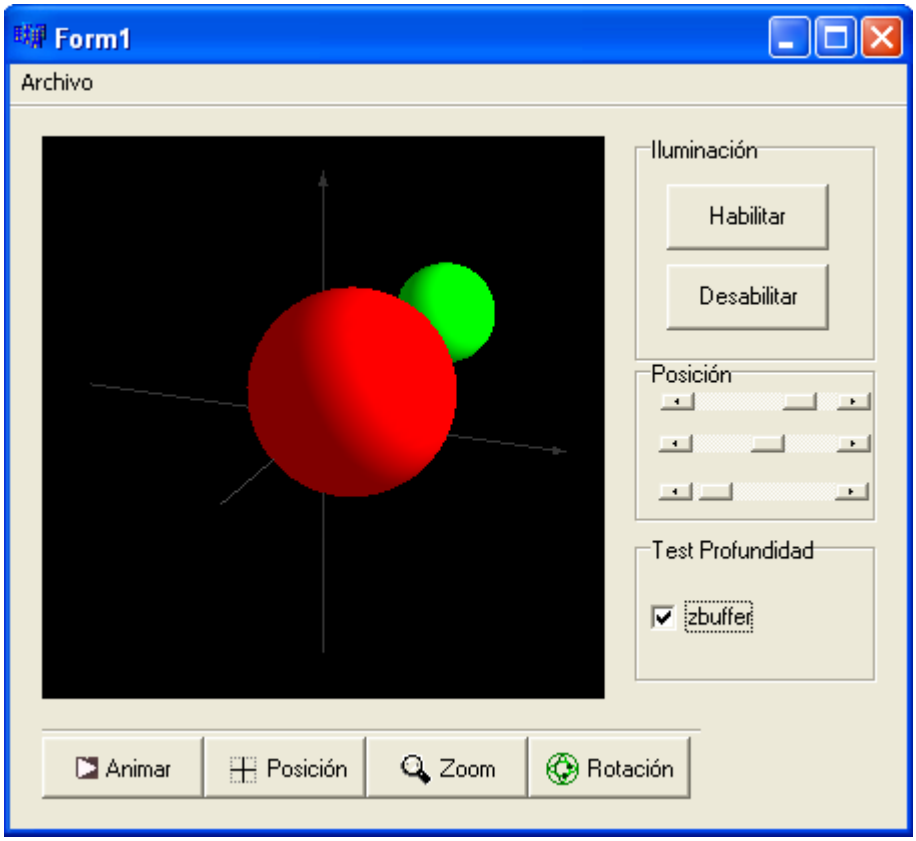


Para indicar el test de profundidad utilizamos la siguiente instrucción:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

GL_DEPTH_BUFFER_BIT indica a OpenGL que almacene las distancia del eje Z en al dirección del observador para poder dibujar apropiadamente los objetos. El resultado lo podemos apreciar en la siguiente figura. En ella

observamos como la sensación de profundidad y apariencia cambia radicalmente.



El código principal del programa anterior es el siguiente:

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::OglWindowOglResize(TObject *Sender)
{
        Camera.OnSetViewport((GLshort)OglWindow->Width,
(GLshort)OglWindow->Height);
        glViewport(0,0,OglWindow->Width,OglWindow->Height);
        glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
        Camera.OglPerspectiveTransformation();
        glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::OglWindowOglCreate(TObject *Sender)
{
     glShadeModel(GL_SMOOTH);
     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
     glClearDepth(1.0f);
     glEnable(GL_DEPTH_TEST);
     glDepthFunc(GL_LEQUAL);
     glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
      Sphere = gluNewQuadric();
      spot = gluNewQuadric();
      x->Position=35;
      y->Position=35;
      z->Position=35;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::OglWindowOglPaint(TObject *Sender)
{
     glPushMatrix();
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      if(!zbuffer->Checked) glDisable(GL_DEPTH_TEST);
      if(zbuffer->Checked) glEnable(GL_DEPTH_TEST);
      Camera.OglViewTransformation();
      iluminacion();
      glRotatef(angulo,1,1,0);
     //    ESFERA  1
       glPushMatrix();
              glTranslatef(1.0f,1.0f,1.0f);
              glColor3f(1.0f, 0.0f, 0.0f);
              gluQuadricDrawStyle(Sphere, GLU_FILL);
              gluQuadricNormals(Sphere, GLU_SMOOTH);
              gluSphere(Sphere, 2, 50, 50);
       glPopMatrix();
      //    ESFERA  2
       glPushMatrix();
              glTranslatef(1.0f,1.0f,-4.0f);
              glColor3f(0.0f, 1.0f, 0.0f);
              gluQuadricDrawStyle(Sphere, GLU_FILL);
              gluQuadricNormals(Sphere, GLU_SMOOTH);
              gluSphere(Sphere, 1, 50, 50);
       glPopMatrix();
      ejes();
       if(l==1) puntos();
     glPopMatrix();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  OglCamera::LoadCursors();
  Camera.OglWindow = OglWindow;
  Camera.SetTarget( 0.0, 0.0, 0.0);
```

```
  Camera.SceneRadius = 1.5;
  Camera.Twist = 0.0;
  Camera.ZoomMin = 0.1; Camera.ZoomMax = 5;
  Camera.Zoom  = 0.3;
}
//--------------------------------------------------------------------------
```