Transformación de las primitivas

I. Remolar

Planteamiento

Esta ficha analiza las funciones disponibles en OpenGL para especificar transformaciones en 3D: traslación, giro y escalado. Estas funciones serán de utilidad tanto para aplicar transformaciones a los distintos objetos de la escena, como para definir las transformaciones asociadas al posicionamiento de dicha escena en el volumen de la vista.

Solución

OpenGL utiliza distintas matrices durante el proceso de visualización. Una de ellas es la matriz de transformación del modelo. Así, en primer lugar, se debe especificar qué se desea trabajar con dicha matriz:

- `void glMatrixMode(GL_MODELVIEW);`

Para inicializarla a la matriz identidad se usa la orden:

- `void glLoadIdentity(void);`

Las transformaciones realizadas, irán multiplicando esta matriz de forma acumulativa. Existen tres rutinas en OpenGL para realizar transformaciones:

- Rutina de traslación, `void glTranslate{f d}(TIPO x, TIPO y, TIPO z);` multiplica la matriz actual por una matriz con desplazamientos en los ejes indicados en los parámetros `x, y, z`.
- Rutina de rotación, `void glRotate{f d}(TIPO angulo, TIPO x, TIPO y, TIPO z);` la matriz actual es multiplicada por otra que realiza un giro en el sentido contrario a las agujas del reloj, del valor indicado por el parámetro `angulo`. El ángulo se define en grados. El giro se realiza alrededor del eje que pasa por los puntos `(0, 0, 0)` y `(x, y, z)`.

- Rutina de escalado, `void glScale{f d}(TIPO x, TIPO y, TIPO z);` multiplica la matriz por otra que amplía o reduce el objeto con respecto al objeto indicado. Valores mayores de 1.0 lo amplían, y entre 0 y 1 lo reducen.

Las transformaciones siempre se aplican en orden inverso al que han sido especificadas en el programa. Por ejemplo, si se desea trasladar y rotar un objeto, la codificación es la siguiente:

```
glRotatef(...);
glTranslatef(...);
dibuja_objeto();
```

Como las transformaciones se almacenan como matrices, OpenGL utiliza una pila de matrices para recordar situaciones anteriores a las transformaciones aplicadas. La matriz actual es siempre la que está en la posición más elevada de la pila. Existen dos rutinas para gestionar esta pila:

- `void glPushMatrix(void);` esta rutina almacena en la pila una copia de la matriz actual.
- `void glPopMatrix(void);` elimina de la pila la matriz que está en la posición más elevada. La matriz actual será la que se encontraba en segundo lugar desde arriba antes de la operación. En el caso de que antes de ejecutar esta rutina sólo existiera una matriz en la pila, se produciría un error.

Ejemplo

Mediante el siguiente código (ver *transf.c*), se dibuja un cono y un toro separado en 22.0 unidades a lo largo del eje X (ver figura 3.11 izquierda).

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glutSolidCone (9.0, 20.0, 16, 16);
glTranslatef(22.0,0.0,0.0);
glutSolidTorus(3.0,7.0, 16,16);
```

Se añade una rotación, -90 grados alrededor del eje X, que produce el giro de ambas primitivas (ver figura 3.11 centro).

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef (-90.0, 1.0, 0.0, 0.0);
glutSolidCone (9.0, 20.0, 16, 16);
glTranslatef(22.0,0.0,0.0);
glutSolidTorus(3.0,7.0, 16,16);
```

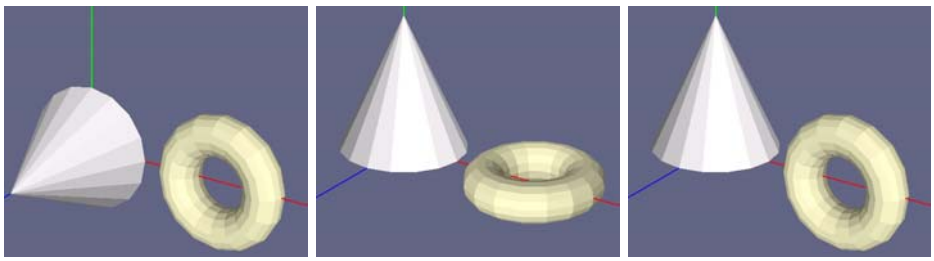

Figura 3.11: Ejemplos de escenas construidas con funciones de transformación.

Mediante el uso de las rutinas que controlan la pila de matrices se establece que el giro se aplique exclusivamente al cono, recuperando la matriz anterior antes de especificar la traslación del toro (ver figura 3.11 derecha).

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
  glRotatef (-90.0, 1.0, 0.0, 0.0);
  glutSolidCone (9.0, 20.0, 16, 16);
glPopMatrix();
glTranslatef(22.0,0.0,0.0);
glutSolidTorus(3.0,7.0, 16,16);
```

Fichas relacionadas

- «La cámara», página 59.

Bibliografía

- Capítulo 3 de Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, *OpenGL programming guide: the official guide to learning OpenGL, (Version 1.2)*, 1999.