MÁSTER EN INFORMÁTICA GRÁFICA, JUEGOS Y REALIDAD VIRTUAL
MEMORIA DE LA PRÁCTICA 3 DE GRÁFICOS 3D

José Zerpa

Objetivo:

- Crear dos modelos 3D de una pirámide y un cubo que estén rotando continuamente creando una animación. Se deben cumplir los siguientes requisitos:
    - El cubo debe tener coloreada cada una de sus caras de un color diferente, con sombreado plano.
    - La pirámide debe tener diferentes colores asociados a cada vértice, y el color de cada cara debe variar mediante un sombreado suave.
    - Ambas figuras deben realizar un movimiento de rotación continuo.
    - Para que la escena final tenga el aspecto adecuado, es necesario trabajar con ocultación de caras ocultas en función de la profundidad

Realización:

Para la realización de esta práctica se utilizaron muchos de los principios de la práctica 2. Lo unico es que ya no hay dependencia del teclado para la animación por lo cual nos tenemos que valer de otra función callback asignada al evento idle de glut mediante la función glutIdleFunc.

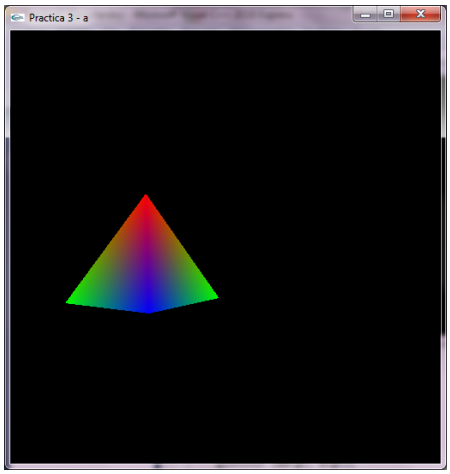
Función de dibujado:

Lo primero es dibujar la pirámide. Para ello se definen todos sus puntos y se traslada una posición hacia la derecha. Luego aplicamos una rotación en y con un ángulo definido en la variable angle. Este ángulo será incrementado en la función idle lo cual permitirá la rotación continua. Por último a cada vértice se le asigna un color diferente para que OpenGL se encargue de mezclarlos.

```
glTranslatef(-1.5f,0.0f,-6.0f);
        glRotatef(angle/5, 0.0,1.0,0.0);

        //Se define la piramide
        glBegin(GL_TRIANGLES);
                glColor3f(1.0f,0.0f,0.0f);
                glVertex3f( 0.0f, 1.0f, 0.0f);
                glColor3f(0.0f,1.0f,0.0f);
                glVertex3f(-1.0f,-1.0f, 1.0f);
                glColor3f(0.0f,0.0f,1.0f);
                glVertex3f( 1.0f,-1.0f, 1.0f);
                glColor3f(1.0f,0.0f,0.0f);
                glVertex3f( 0.0f, 1.0f, 0.0f);
                glColor3f(0.0f,0.0f,1.0f);
                glVertex3f( 1.0f,-1.0f, 1.0f);
                glColor3f(0.0f,1.0f,0.0f);
                glVertex3f( 1.0f,-1.0f, -1.0f);
                glColor3f(1.0f,0.0f,0.0f);
```

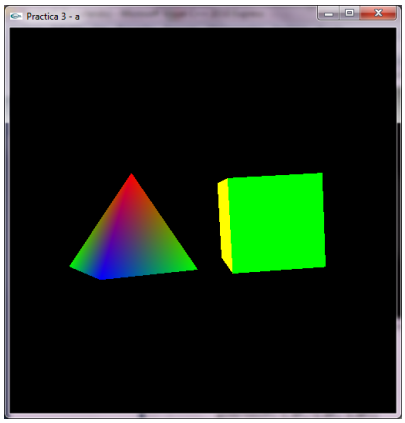
```
            glVertex3f( 0.0f, 1.0f, 0.0f);
            glColor3f(0.0f,1.0f,0.0f);
            glVertex3f( 1.0f,-1.0f, -1.0f);
            glColor3f(0.0f,0.0f,1.0f);
            glVertex3f(-1.0f,-1.0f, -1.0f);
            glColor3f(1.0f,0.0f,0.0f);
            glVertex3f( 0.0f, 1.0f, 0.0f);
            glColor3f(0.0f,0.0f,1.0f);
            glVertex3f(-1.0f,-1.0f,-1.0f);
            glColor3f(0.0f,1.0f,0.0f);
            glVertex3f(-1.0f,-1.0f, 1.0f);
    glEnd();
```



Lo siguiente es dibujar el cubo, para ello al igual que con la pirámide se definen la posición de sus vértices, se traslada a la derecha una posición y luego se rota en y con el mismo valor del ángulo de la pirámide para roten a una misma velocidad. Para que el cubo mantuviese un color sólido en cada cara se define un color por cada cara en vez de cada vértice, al ser opengl una máquina de estados al definir un color este se mantiene hasta que se defina otro.

```
glTranslatef(1.5f,0.0f,-7.0f);
glRotatef(angle/5,1.0f,1.0f,1.0f);
//Se define el cubo
glBegin(GL_QUADS);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glColor3f(1.0f,0.5f,0.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
```

```
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glColor3f(1.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glColor3f(1.0f,0.0f,1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);
glEnd();
```



Manejo de Eventos:

Para lograr la rotación continua nos valemos del evento Idle, que se llama cuando la CPU está desocupada. Glut permite asignar una función a este evento mediante la función glutIdleFunc. En nuestro caso utilizamos la misma función display así cada vez que dibuja aumenta el ángulo para en el próximo frame aparezca la figura rotada.

```
void display ()
{
    glClearDepth(1);
    glClearColor(0,0,0,1);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();
    dibujar_figuras();
    glutSwapBuffers();
    angle++;
}
```