Prácticas de Informática Gráfica

Grupos C y D (Curso 2024/2025)

Germán Arroyo y Juan Carlos Torres

2024-06-06

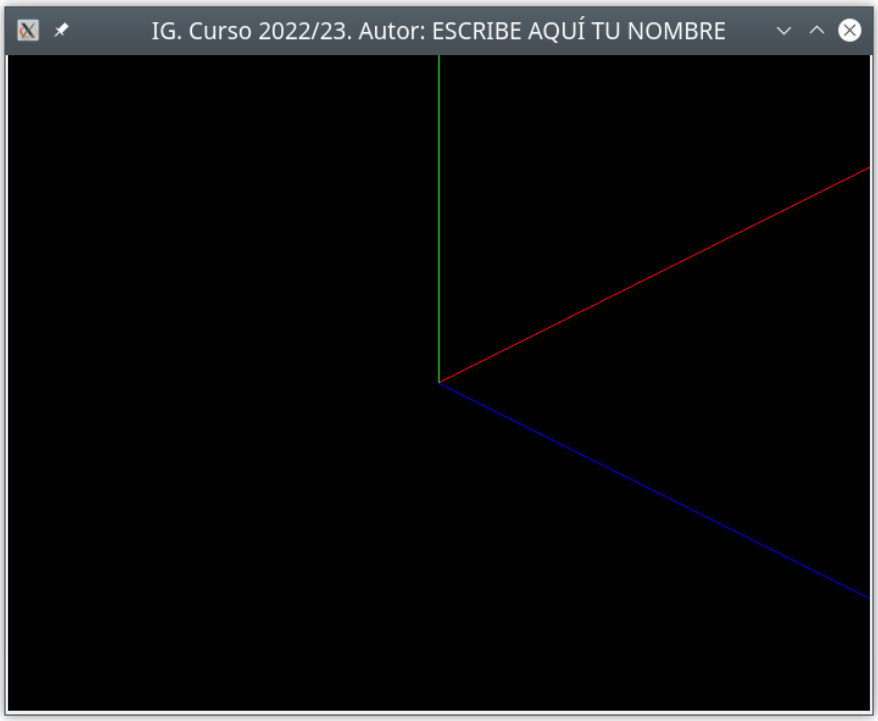
Práctica 1: Programación con una biblioteca de programación gráfica

Objetivos

- Saber crear programas que dibujen geometrías simples con OpenGL.
- Entender la estructura de programas sencillos usando OpenGL y glut.
- Aprender a utilizar las primitivas de dibujo de OpenGL.
- Distinguir entre la creación del modelo geométrico y su visualización.

Código inicial

Para el desarrollo de esta práctica se entrega el esqueleto de una aplicación gráfica programada en C++ sobre OpenGL y glut, formado por los siguientes módulos:



[r0.50]

- `practicasIG.c`: Programa principal. Inicializa OpenGL y glut, crea la ventana de dibujo, y activa los manejadores de eventos (Debes editarlo para escribir tu nombre en el título de la ventana X).
- `entradaTeclado.c`: Contiene las funciones que respoden a eventos de teclado (Todas las prácticas).
- `modelo.c`: Contiene la función que dibuja la escena, y las funciones de creación de objetos (Todas las prácticas).
- `mouse.c`: Contiene las funciones que responden a eventos de ratón (Práctica 5).
- `visual.c`: Contiene las funciones de proyección transformación de visualización, y el *callback* de cambio de tamaño de ventana (Práctica 4 y 5)
- `file_ply_stl.cc`: Funciones de lectura de archivos ply (práctica 2).
- `practicasIG.h`: Incluye los archivos de cabecera de todos los módulos.

Abre los diferentes archivos y mira su contenido, pero no te preocupes por las cosas que en este momento no entiendas.

En las primeras prácticas trabajaremos solamente con los archivo `modelo.c` y `entradaTeclado.c`.

El código base se puede compilar usando el fichero `Makefile` incluido. Si lo ejecutas debes ver los ejes del sistema de coordenadas, dibujados con una cámara

orbital que mira al origen de coordenadas. Puedes girar la cámara alrededor del origen (usando las teclas **X** e **Y** o bien las teclas de movimiento del cursor: ←, ↑, →, ↓) y alejarla o acercarla (usando **d** y **D** o las teclas de avance y retroceso de página: **Re Pág** y **Av Pág**).

Funcionalidad a desarrollar

Se deberá crear y visualizar una pirámide de base cuadrada y un cubo (ver Fig. 1.1), que se visualizarán con los siguientes modos:

- Puntos
- Alambre
- Sólido sin iluminación
- Sólido con iluminación

El cambio del modo de visualización se hace usando el teclado.

Se debe utilizar un color diferente para cada modelo, y los dos deben dibujarse en el escenario sin solaparse cerca del origen de coordenadas.

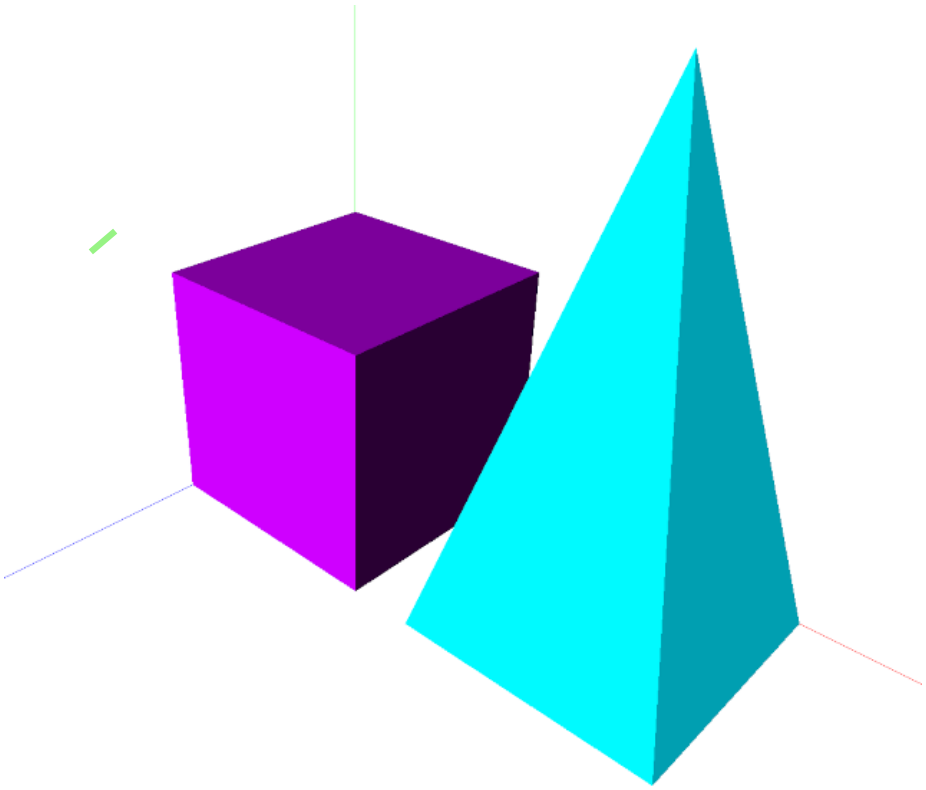


Figura 1: Escena creada en la práctica 1

Procedimiento

Antes de nada abre los archivos `practicasIG.c`, `entradaTeclado.c` y `modelo.c`, familiarízate con el código.

Observa la clase objeto3D (definida en `modelo.h`):

```
class Objeto3D
{
  public:

  virtual void draw( ) = 0; // Dibuja el objeto
} ;
```

y la definición de `Ejes` como una clase derivada de `Objeto3D`. Los objetos que crees en la práctica los debes crear igualmente como clases derivadas de `Objeto3D`.

A continuación añade tu nombre a la ventana X. Para ello edita la llamada a `glutCreateWindow` en `practicasIG.c`.

Comienza dibujando el cubo. Para ello crea una clase `Cubo`, con un constructor al que le puedas pasar el tamaño `Cubo(float lado)` e implementa su método `draw`. Puedes utilizar triángulos o cuadriláteros para dibujarlo. Crea un cubo y llama a su método `draw` desde la función `Dibuja` para que se dibuje en cada fotograma (*frame*). Coloca la llamada al final del dibujo (antes del `glPopMatrix`).

Para que se calcule la iluminación debes asignar a cada cara su normal (puedes hacer un dibujo para ver que normal tiene cada cara), y ten en cuenta que los vértices se deben dar en sentido antihorario mirando el modelo desde fuera:

```
glBegin( GL_QUADS );
    glNormal3f( -1.0, 0.0, 0.0 );
    glVertex3f( x, 0, 0 );
    glVertex3f( x, y, 0 );
    glVertex3f( x, y, z );
    glVertex3f( x, 0, z );
    ...
```

Ahora crea la pirámide siguiendo el mismo procedimiento, utilizando dos parámetros en el constructor (`lado` y `alto`), y haz que se dibuje junto al cubo, para ello puedes aplicarle una traslación. No es difícil calcular geométricamente las normales de las caras, pero no hace falta que te preocupes de esto ahora, puedes asignarlas de forma aproximada (veremos como calcularlas mas adelante). Para que se dibuje de otro color crea otra variable para representar el nuevo color y aplícalo antes de dibujar la pirámide:

```
Cubo.draw();
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, color2 );
glTranslatef(5,0,0);
Piramide.draw();
```

Para cambiar el modo de visualización puedes usar la función glPolygonMode para indicarle a OpenGL que debe dibujar de cada primitiva (`GL_POINT`, `GL_LINE` o `GL_FILL`):

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

Para modificar interactivamente el modo de dibujo crea una variable en `modelo.c` para almacenar el modo actual, e inicialízala a `GL_FILL`:

```
int modo = GL_FILL;
```

Crea una función para cambiar el modo (p.e. `void setModo(int M)`) que asigne el valor de `M` al modo. Tendrás que añadir su cabecera a `modelo.h`.

Añade casos en el `switch` de la función `letra` en `entradaTeclado` para responder a las pulsaciones de las letras **P**, **L**, **F**. Haz que en cada una se llame a `setModo` con el modo que corresponda.

Para activar y desactivar el cálculo de iluminación se puede seguir un procedimiento parecido, usando la tecla **I**: Crea una variable para indicar si se activa la iluminación y una función para modificar su valor. Haz que se cambie su estado pulsando la letra **I**. Por último, haz que se active o desactive el cálculo de iluminación en la función `Dibuja` llamando a

```
glDisable (GL_LIGHTING);
```

o

```
glEnable (GL_LIGHTING);
```

Ten en cuenta que la función que dibuja los ejes activa el modelo de iluminación en el código de partida. Tendrá que hacerlo solamente cuando se esté en el modo *Sólido con iluminación*.

Temporización

Se recomienda realizar esta práctica en una única sesión. Para ello, debe trabajarse previamente en casa.