MEMORIA DE LAS PRÁCTICAS 3-5 DE INFORMÁTICA GRÁFICA

Grado en Ingeniería Informática - Curso 22/23

Informática gráfica (IG) - Grupo C2

Marina Jun Carranza Sánchez

ÍNDICE

a) PRÁCTICA 3

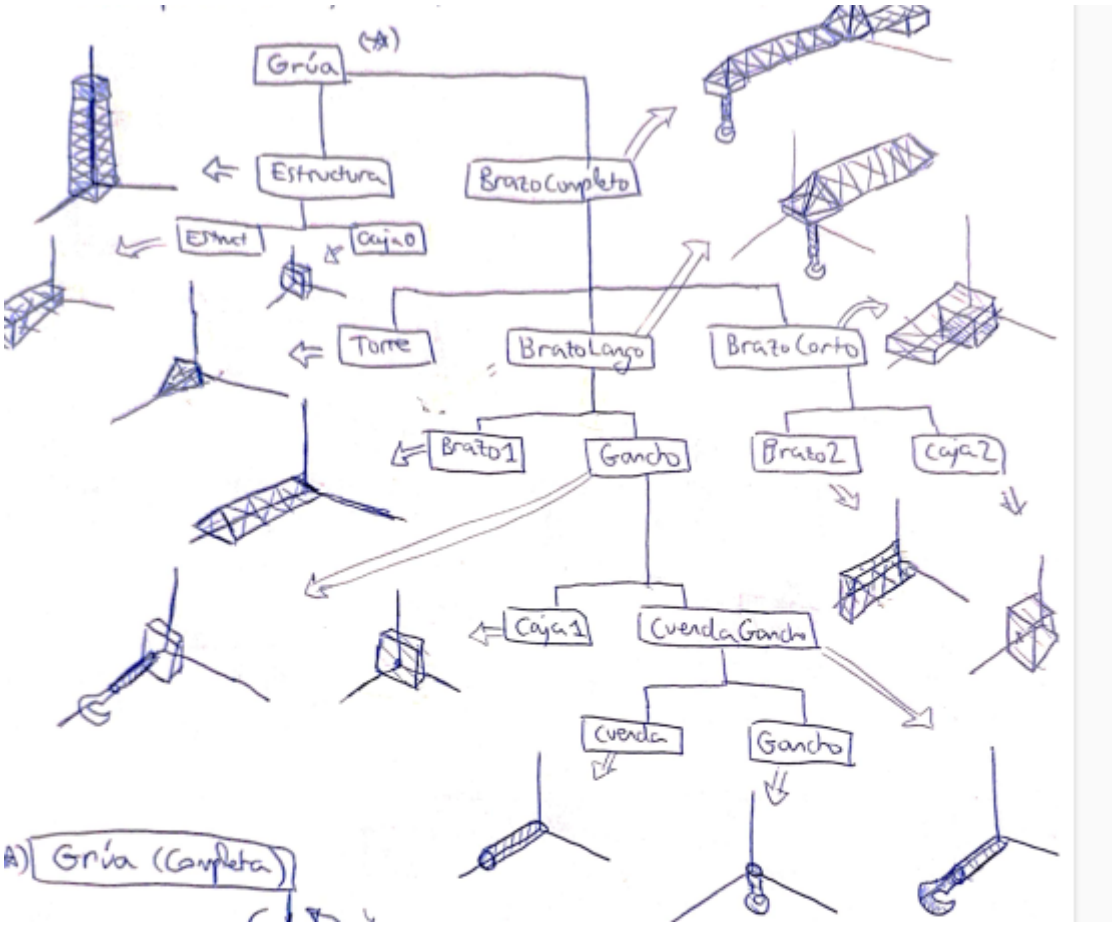
1) Explicación y ubicación de ficheros

En esta práctica se pedía elaborar un modelo jerárquico al que se le darían, como mínimo, tres grados de libertad distintos. En mi caso, desarrollé una grúa haciendo uso de las funciones que ya vienen implementadas en `estructura.c`. Los grados de libertad escogidos son: giro del brazo entero, giro del gancho y altura de la cuerda.

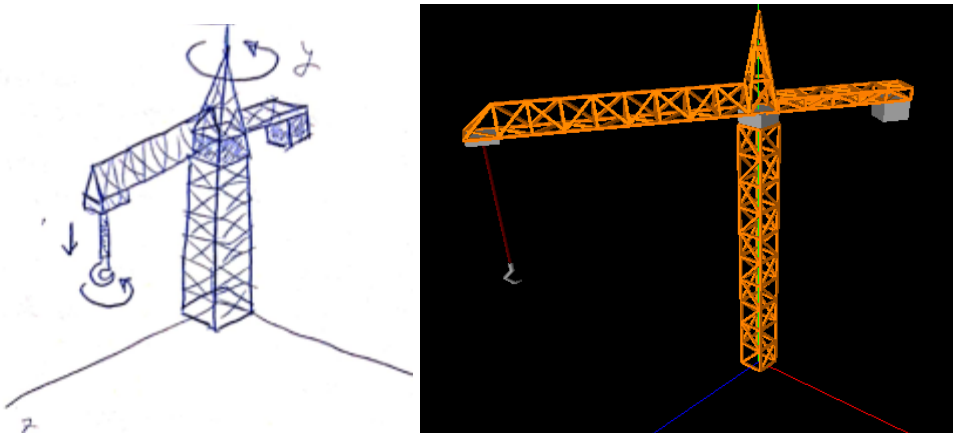
La clase `Grua` hereda de `Objeto3D`, su cabecera está ubicada en **`grua.h`** y sus métodos, en **`grua.c`**. A excepción del método constructor, el de dibujado y los dos de modificación de parámetros (`modParam()` y `setMove()`), el resto de métodos se corresponden a las distintas partes o conjunto de partes de la grúa, de modo que se encapsulan siguiendo una jerarquía. El programa main donde se llamará creará y dibujará una instancia de la grúa se encuentra en **`modelo.c:112`**.

Para un mayor entendimiento del modelo jerárquico seguido, se puede consultar el archivo **`IG-P3_grafo.pdf`** dentro de la misma carpeta `/Practica3`.

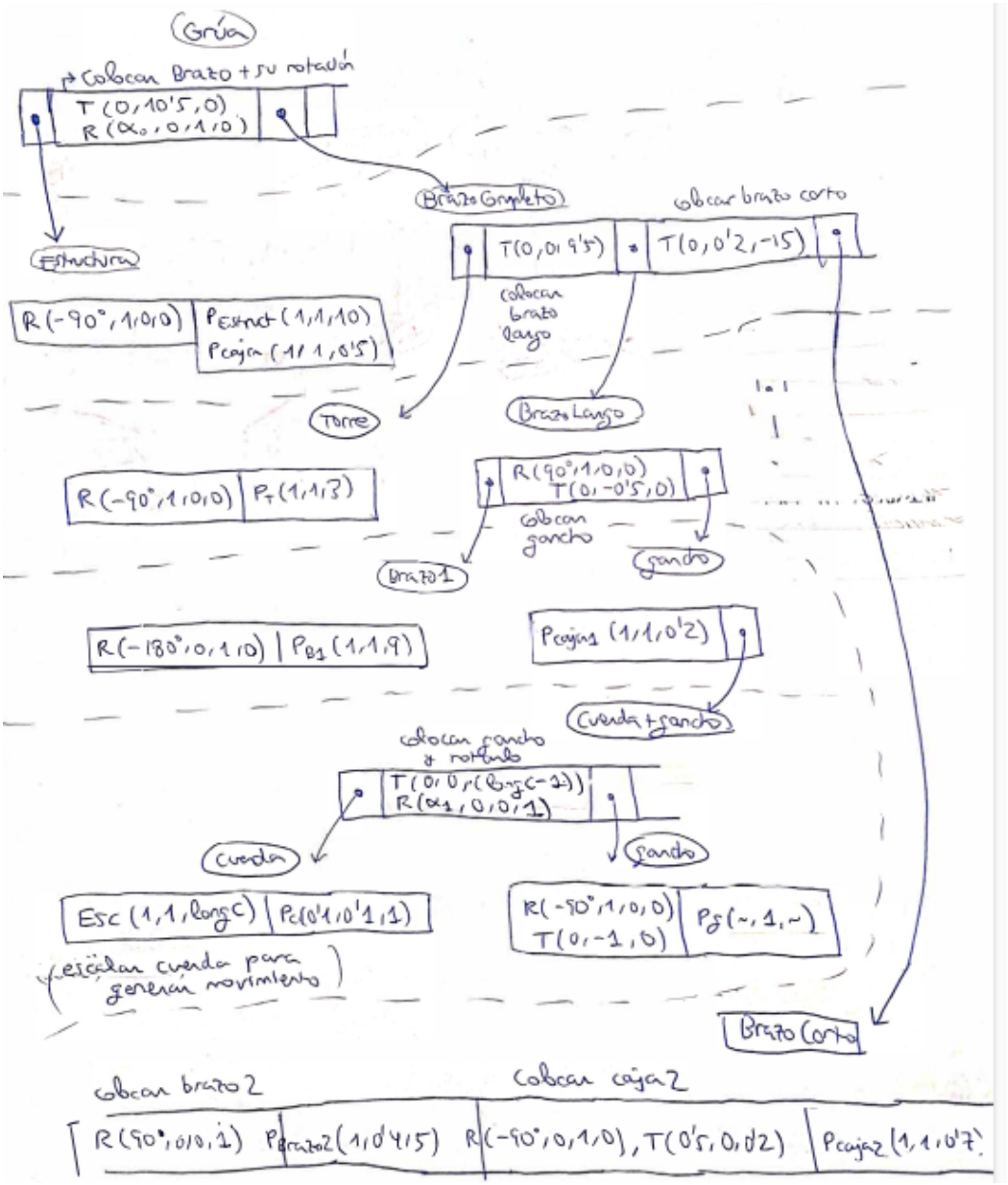
La descomposición de la grúa, ilustrada con bocetos de la escena (grafo), es la que sigue:



Y la grúa completa en su estado inicial, con sus tres grados de movimiento, sería así:



El esquema de la pila de transformaciones es el siguiente:



2) Lista de teclas utilizadas

Se encuentra en el fichero `entradaTeclado.c`, en la siguiente función:

```
void letra(unsigned char k, int x, int y);
```

- **'h','H':** llaman a la función printHelp() para imprimir ayuda.
- **'+':** acerca la cámara.
- **'-':** aleja la cámara.
- **'←','↑','→','↓'(flechas teclado):** giran la cámara hacia dicha dirección.
- **'p':** llama a `setModo(GL_POINT)` para visualizar los modelos como puntos.
- **'l':** llama a `setModo(GL_LINE)` para visualizar los modelos como líneas.
- **'f':** llama a `setModo(GL_FILL)` para visualizar los modelos como objetos sólidos y rellenos.
- **'i':** llama a `setIlum(ilum)` para encender/apagar la fuente de luz.
- **'A':** alterna el valor del booleano `move` mediante la función `setMovement(move)` para (des)activar el movimiento automático por defecto de la grúa.
- **'B'**: modifica el parámetro que determina el ángulo del brazo de la grúa para ir rotándolo en sentido antihorario a través del método `modParamGrua(1,0,0)`.
- **'b':** modifica el parámetro que determina el ángulo del brazo de la grúa para ir rotándolo en sentido horario a través del método `modParamGrua(-1,0,0)`.
- **'M':** modifica el parámetro que determina el ángulo del gancho de la grúa para ir rotándolo en sentido horario a través del método `modParamGrua(0,5,0)`.
- **'m':** modifica el parámetro que determina el ángulo del gancho de la grúa para ir rotándolo en sentido antihorario a través del método `modParamGrua(0,-5,0)`.
- **'C':** modifica el parámetro que determina la longitud de la cuerda de la grúa para ir alargándola a través del método `modParamGrua(0,0,1)`.
- **'c':** modifica el parámetro que determina la longitud de la cuerda de la grúa para ir acortándola a través del método `modParamGrua(0,0,-1)`.
- **'Esc'(27):** es la tecla Escape, se llama para cerrar la ventana de visualización OpenGL.

b) PRÁCTICA 4

1) Explicación y ubicación de ficheros

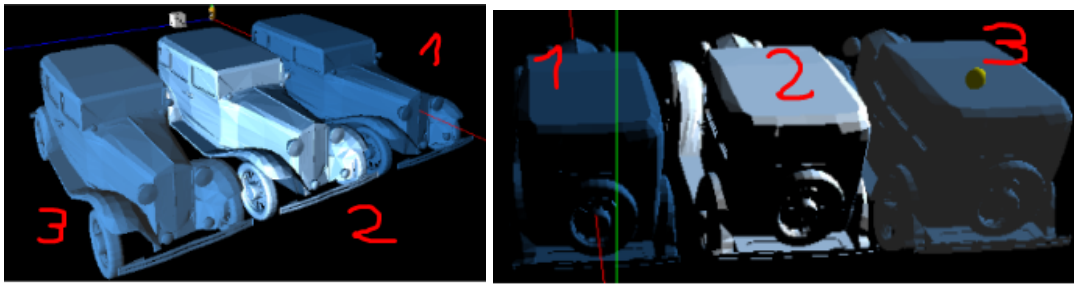
Todas las instancias de objetos se declaran como punteros y se inicializan en la función **texturas.c:120:**`initModel()`, dentro de la cual también deben cargarse las texturas antes de dibujar los modelos. Todas las llamadas a métodos de dichas instancias se realizan dentro de **texturas.c:184:**`Dibuja()`.

➢ Representación y visualización de materiales

En el fichero **objetoPly.h:44**, se le ha añadido a la clase `ObjetoPly` (o clase mallas) variables para almacenar los parámetros de visualización (ambiente, difusa y especular) de los materiales, así como un método para modificar dichos valores (**objetoPly.c:152**: `materialSettings()`).

Ejemplo de visualización del mismo objeto con ajustes de material distintos:

1. Solo difusa
2. Solo especular
3. Ambiente y difusa

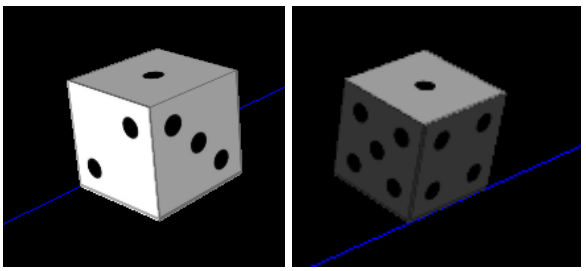


➢ Textura del cubo (Dado)

Antes se debe añadir una variable identificadora de la textura (`GLuint texId`) a la clase `Objeto3D` (**texturas.h:48**). También métodos que carguen y vinculen la textura (**texturas.c:39:**`setTexture()` y **texturas.c:45:**`loadTexture()`), cuando se le pasa como parámetro el nombre de un archivo `.jpg`.

Se ha creado una clase `Dado`, que hereda de `Objeto3D`, y cuyo código se encuentra en el archivo **miModelo.c:394**.

El método de dibujo del dado consiste en ir especificando manualmente las coordenadas de textura (dos flotantes) con `glTexCoord2f()` antes de llamar a `glVertex3f()` e ir trazando el modelo.



➢ Texturas en objetos de revolución (Lata)

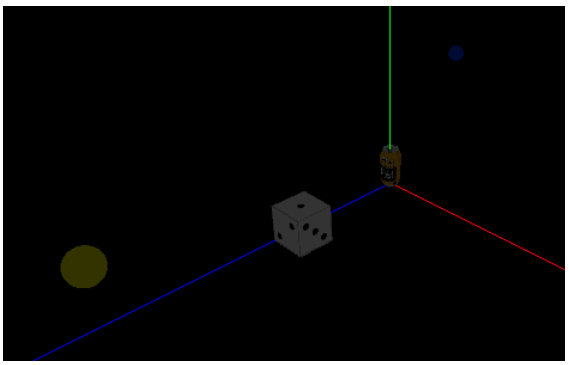
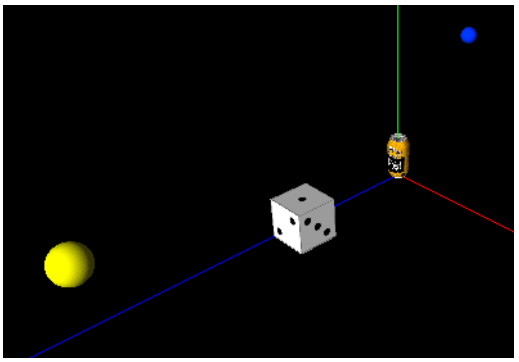
Previamente, se ha añadido a la clase `ObjetoPly` un vector que contiene todos los pares de coordenadas de la textura, cuyo tamaño es el número de vértices total del perfil original.

Para dibujar la lata con la textura que se haya seleccionado, se ha creado un nuevo método (**`objetoPly:292:`**`drawTexture()`) que sigue el procedimiento explicado en el guión para calcular el conjunto de pares de coordenadas de la textura y almacenarlo en el vector de la clase. Luego, halla las normales de vértices, habilita las texturas 2D y carga la que tenga asignada bajo el identificador `texId` y procede con la asignación de coordenadas de textura (que acaban de calcularse) a los vértices con `glTexCoord2f()`.

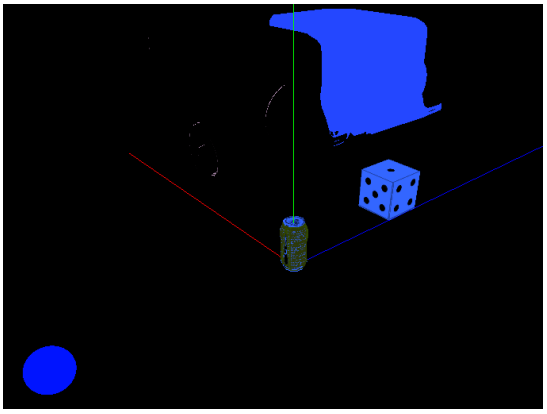
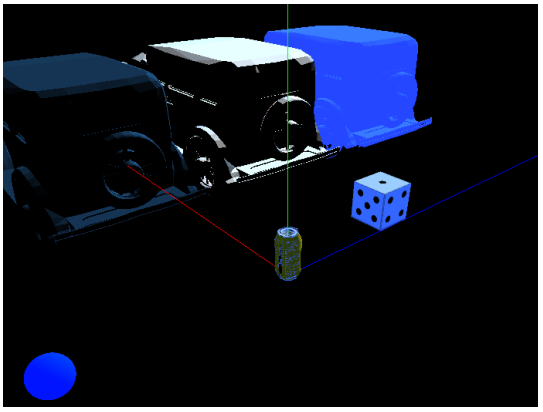
➢ Fuentes de luz

Se han creado dos fuentes de luz distintas, que se encienden y apagan pulsando la tecla 1 y 2 respectivamente.

La primera (`GL_LIGHT0`), representada con una esfera amarilla, funciona como un foco que ilumina las figuras de la siguiente forma:

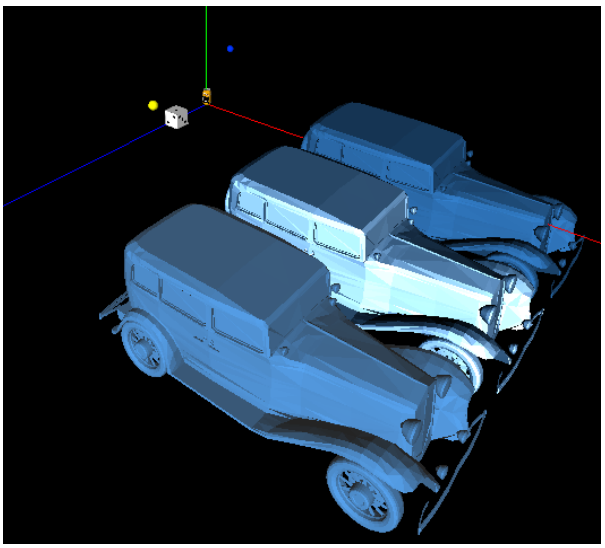



La segunda (`GL_LIGHT1`), cuya posición viene representada por la esfera azul, hace que todos los objetos parezcan azules. En la imagen derecha de abajo se han encendido ambas luces a la vez.

➢ Composición de la escena

Una visión general de la escena final, la cual se ha ido construyendo progresivamente durante el desarrollo de la práctica.



2) Lista de teclas utilizadas

Se encuentra en el fichero `entradaTeclado.c`, en la siguiente función:

```
void letra(unsigned char k, int x, int y);
```

- **'h','H':** llaman a la función printHelp() para imprimir ayuda.
- **'+':** acerca la cámara.
- **'-':** aleja la cámara.
- **'←','↑','→','↓'(flechas teclado):** giran la cámara hacia dicha dirección.
- **'p':** llama a `setModo(GL_POINT)` para visualizar los modelos como puntos.
- **'l':** llama a `setModo(GL_LINE)` para visualizar los modelos como líneas.
- **'f':** llama a `setModo(GL_FILL)` para visualizar los modelos como objetos sólidos y rellenos.
- **'1':** llama a `setIlum2(ilum,1)` para encender/apagar la luz 1 (`GL_LIGHT0`).
- **'2':** llama a `setIlum2(ilum,2)` para encender/apagar la luz 2 (`GL_LIGHT1`).
- **'Esc'(27):** es la tecla Escape, se llama para cerrar la ventana de visualización OpenGL.

c) PRÁCTICA 5

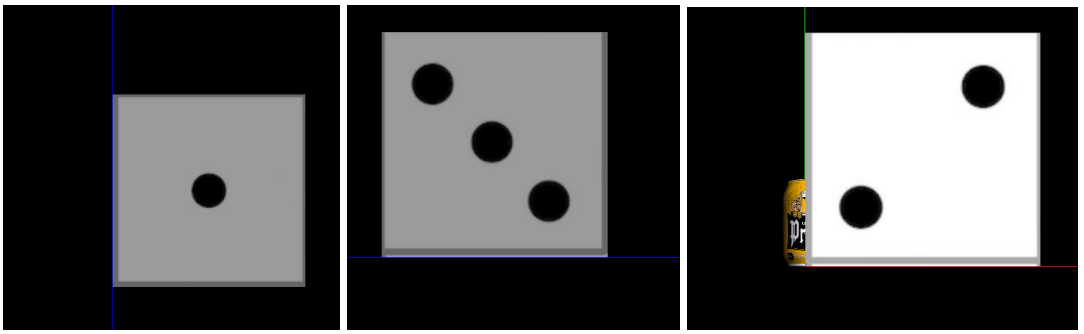
1) Explicación y ubicación de ficheros

El programa main, que contiene la función de dibujo de la escena, se encuentra en el fichero **`interaccion.c`**.

➢ Añadir vistas con proyección en planta, alzado y perfil.

Ficheros: `entradaTeclado.c, visual.c`

En planta | Alzado | Perfil



➢ Mover la cámara usando el ratón.

Ficheros: `entradaTeclado.c, mouse.c`

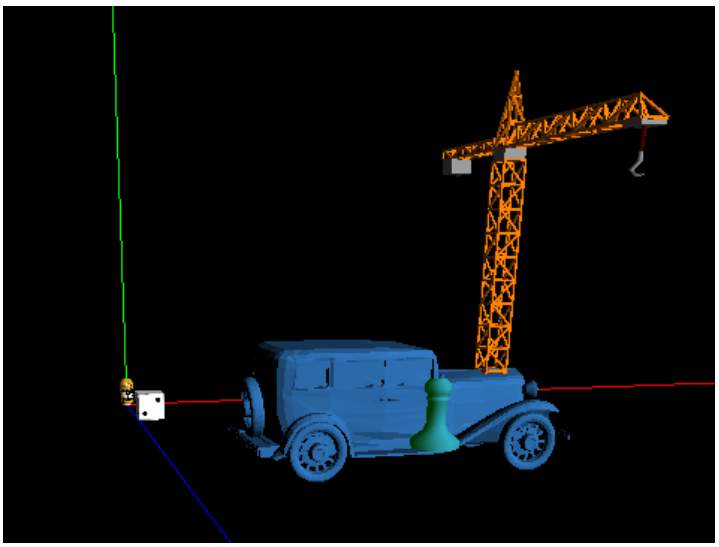
Funciones `clickRaton()` y `RatonMovido()` para mover y girar la cámara con las teclas indicadas en la práctica (a,w,d,s, botón central ratón).

➢ Seleccionar objetos de la escena con el ratón.

Ficheros: `interaccion.c`

Se ha creado la función `pick()` y una nueva función `dibujoEscena()`, que contiene el código de `Dibuja()` a excepción de la llamada a `SwapBuffer()`.

Escena con los objetos de las prácticas 2, 3 y 4:



2) Lista de teclas utilizadas

Se encuentra en el fichero `entradaTeclado.c`, en la siguiente función:

```
void letra(unsigned char k, int x, int y);
```

- **'h','H':** llaman a la función printHelp() para imprimir ayuda.
- **'+':** acerca la cámara.
- **'-':** aleja la cámara.
- **'←','↑','→','↓'(flechas teclado):** giran la cámara hacia dicha dirección.
- **'p':** llama a `setModo(GL_POINT)` para visualizar los modelos como puntos.
- **'l':** llama a `setModo(GL_LINE)` para visualizar los modelos como líneas.
- **'f':** llama a `setModo(GL_FILL)` para visualizar los modelos como objetos sólidos y rellenos.
- **'i':** llama a `setIlum(ilum)` para encender/apagar la fuente de luz.
- **'a','w','d','s'(a modo de flechas de movimiento):** mueve la cámara hacia la izquierda, arriba, la derecha y abajo, respectivamente, manteniendo la dirección hacia la que enfoca.
- **'r':** reinicia la posición de la cámara.
- **Botón central del ratón:** mantener pulsado y mover el cursor para rotar la cámara.
- **Botón derecho del ratón**: mantener pulsado para entrar en modo selección.
- **'F1':** visualización en planta.
- **'F2':** visualización alzada.
- **'F3':** visualización perfil.
- **'Esc'(27):** es la tecla Escape, se llama para cerrar la ventana de visualización OpenGL.