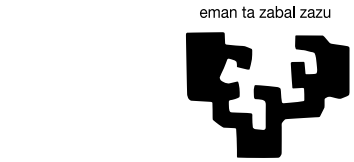

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Transformaciones Geométricas

Silvia Arenales Muñoz

October 31, 2022

Abstract

Este escrito describe la implementación de una aplicación implementada en lenguaje de programación C y ha sido documentada mediante Latex.

1 Objetivos de nuestra aplicación

El objetivo de esta primera parte de nuestra práctica final de la asignatura de Gráficos por Computador, sobre el tema dado en clase, transformaciones geométricas, es desarrollar una aplicación es cargar objetos 3D desde fichero y que se visualice en pantalla. Además que el usuario sea capaz de cargarlos y transformalos (rotar, transaladar o escalar) sobre los objetos en el escenario. Para la realización de este se realizará mediante la librería OpenGL de C.

2 Sobre este documento

Este documento explica el desarrollo de la aplicación implementada, el cual es la implementación de una aplicación que cargue, transforme y visualice objetos 3D. Asimismo, este especifica los problemas surgidos y conocimientos adquiridos a la hora de realizarlo.

3 Funciones de teclado

La interacción del usuario con la aplicación consistirá en presionar varias teclas que tendrán diversas funciones.
Primero tenemos las teclas con funciones principales:

- **?** Visualizar la ayuda
- **ESC** Finalizar la ejecución de la aplicación
- **F,f** Carga de objeto desde un fichero *.obj
- **TAB** Seleccionar siguiente objeto (de entre los cargados)
- **SUPR** Eliminar objeto seleccinado
- **Z** Deshacer
- **UP** Trasladar +Y; Escalar + Y; Rotar +X
- **DOWN** Trasladar -Y; Escalar - Y; Rotar -X
- **RIGHT** Trasladar +X; Escalar +X; Rotar +Y
- **LEFT** Trasladar -X; Escalar -X; Rotar -Y
- **AVPAG** Trasladar +Z; Escalar +Z; Rotar +Z

- **REPAG** Trasladar -Z; Escalar - Z; Rotar -Z
- **O,o** Modo Objeto
- **C,c** Modo Cámara

Luego tenemos las teclas correspondientes a las tranformaciones.
Teniendo en cuenta el tipo de transformación, presionaremos según lo siguiente:

- **T,t** Activar traslación (desactiva rotación y escalado)
- **R,r** Activar rotación (desactiva traslación y escalado)
- **E,e** Activar escalado (desactiva rotación y traslación)

Teniendo en cuenta el sistema de referencia, presionaremos según lo siguiente:

- **G,g** Activar transformaciones en el sistema de referencia del mundo (transformaciones globales)
- **L,l** Activar transformaciones en el sistema de referencia local del objeto (objeto 3D, cámara o luces)

Y por último, si estamos en **modo objeto**:

- + Escalar + en todos los ejes (solo objeto seleccionado)
- **-** Escalar - en todos los ejes (solo objeto seleccionado)

Por otro lado, si estamos en **modo cámara**:

- + Realiza zoom in de todos los objetos
- **-** Realiza zoom out de todos los objetos

4 Datos recibidos

Los objetos a dibujar son estructuras 3D formadas por diversas líneas que forman polígonos, dichos polígonos son las caras de nuestro objeto a moldear. En el fichero *.obj* se definen los puntos y polígonos que conforman el objeto mediante la siguiente sintaxis:

- **Vértices**: Las lineas del fichero que empiezan por el carácter **v** tienen información sobre los vértices del objeto. Seguidos del carácter **v** separados por un espacio están las coordenadas x,y,z del vértice en formato float.

  Ex: v -0.750000 0.750000 2.500000

- **Nº vértices**: Tras las lineas correspondientes a los vértices habrá una línea que comienze por el carácter , que contendrá el número total de vértices del objeto.

  Ex: 601 vertices

- **Polígonos**: Después de la línea que contienen el nº de vértices, separada por una línea vacía, estrán las líneas de los polígonos, que comienzan por el **f** y están seguidos por varios números, que representan el índice de los vértices que los forman.

  Ex: f 1 2 3 4

- **Nº polígonos**: En la última línea del texto contiene el número de polígonos por los que está formado el objeto, el cual comienza por el carácter .

  Ex: 535 elements

5 Implementación del código

Primeramente explicaremos el funcionamiento de nuestra aplicación.

Al iniciar nuestra aplicación se dibujará un escenario vacío. Para poder visualizar algún objeto, debemos de cargar un objeto pulsando la tecla F y escribiendo su ruta hasta su descriptor.

La aplicación cargará objetos en extensión **.obj**, para ello leerá la información del fichero y la guardará en una tabla.

La tabla tiene la siguiente estructura:

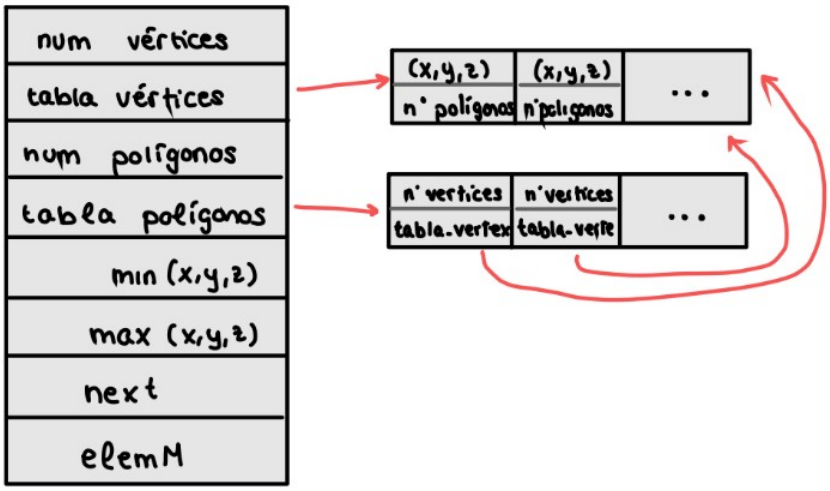

Figure 1: Estructura

Los objetos están compuestos por puntos, líneas y polígonos. Y para poder dibujarlos necesiatamos sabes sus polígonos. Cada polígono es diferente y necesitamos el número de vértices para ello.

Podemos cargas tantos objetos como queramos, creando así una lista de objetos.

Con el objeto en nuestro mundo ya es posible transformarlo. Usaremos las teclas R, T, E, AvPag y RePag para seleccionar el tipo de transformación y usaremos las flechas de direccionamiento y las teclas UP, DOWN, LEFT y RIGHT para aplicar la transformación. Las transformaciones se pueden realizar bien en el sistema de referencia local correspondiente al objeto o en el sistema global del mundo, para cambiar entre estas dos opciones se usan

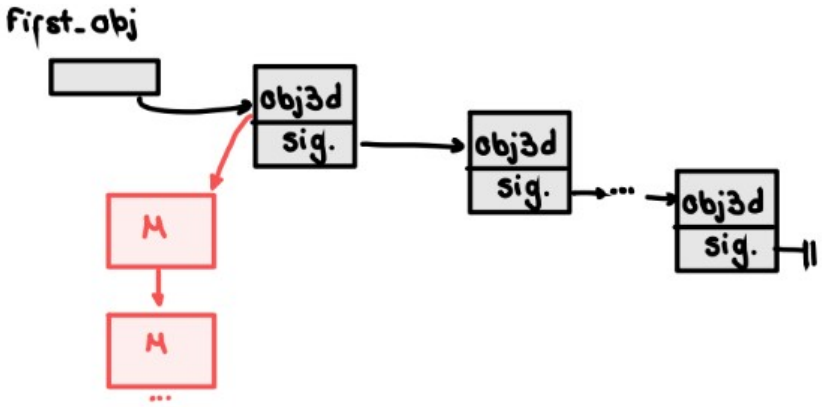

Figure 2: Escena, lista de objetos

las teclas G y L.

Para navegar entre los objetos cargados, usaremos TAB, así podremos diferenciar la seleccionada. Además, si queremos deshacer alguna podemos hacer uso SUPR para borrarla.

5.1 Transformaciones a los objetos

Sabemos que los objetos se cargan desde un fichero y se guardan en una estructura en la memoria del programa. Dicha estructura contiene la información de los vértices y los polígonos que lo conforman, pero esa información solo es correcta cuando el objeto es cargado, en el momento que comenzamos a realizar transformaciones.

Para poder realizar transformaciones manteniendo los datos iniciales de los objetos se ha hecho uso de matrices de transformación y para esto hacemos uso de las funciones propias de **OpenGL**. Cada objeto tendrá asociada una lista de matrices que representarán las transformaciones que ha sufrido. Al cargar el objeto se inicializará su matriz como la matriz identidad, la cual lacargamos con la función **glLoadIdentity()**

Para facilitar las operaciones entre matrices se utilizan los métodos integrados en OpenGL **glRotate(angulo, x, y, z)**, **glTranslate(x, y, z)** y **glScale(x, y, z)**.

OpenGL trabaja con dos matrices principales, **GL_PROJECTION** y **GL_MODELVIEW**. Estas funciones aplicarán la operacion a la matriz con

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & m \\ 0 & 1 & 0 & n \\ 0 & 0 & 1 & o \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

(a) Matriz de translación.

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} p & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

(b) Matriz de escalado.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

(c) Rotación respecto al eje Z.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

(d) Rotación respecto al eje X.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

(e) Rotación respecto al eje Y.

Figure 3: Matrices de Transformación

la se este trabajando en ese momento, por lo que habrá que seguir los siguientes pasos para conseguir la matriz resultado:

1. Se indica que se va a trabajar con las matrices de OpenGL.
2. Se carga la matriz previa del objeto en la matriz MODELVIEW.
3. Se llama a una de las funciones de transformación, que se ejecutará sobre la matriz que está en MODELVIEW.
4. Se obtiene la matriz resultado desde el MODELVIEW.
5. Se guarda la matriz resultado en una estructura de matriz, que tendrá un puntero que apunte a la matriz anterior del objeto.
6. Una vez obtenida la matriz del objeto transformado, deberemos cargar en la matriz MODELVIEW antes de dibujarlo, y de esta manera se aplicarán al objeto las transformaciones correspondientes a la matriz.

6 Resultados

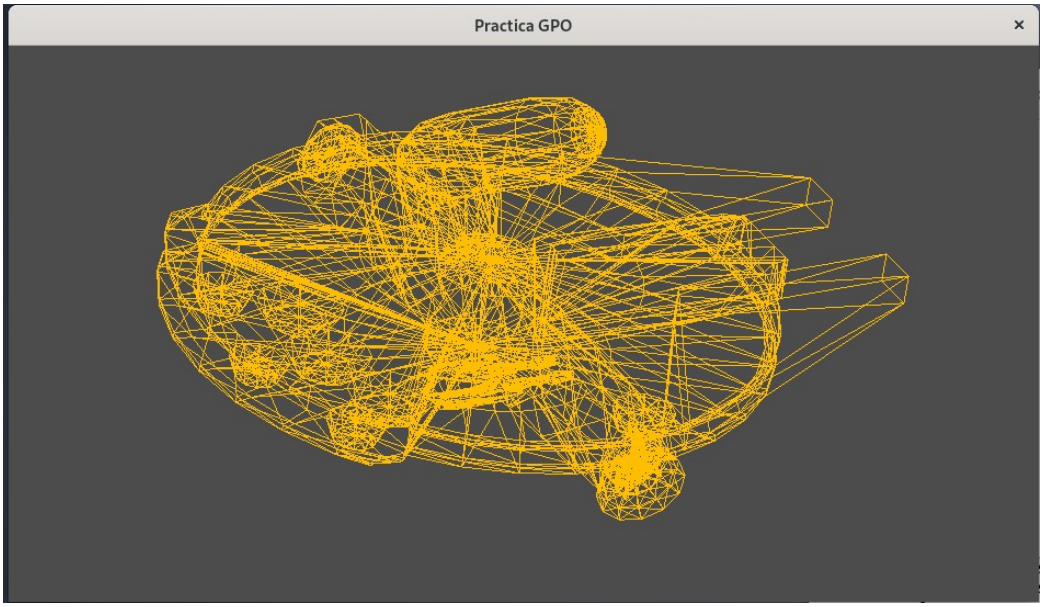

Figure 4: Objeto mirua.obj

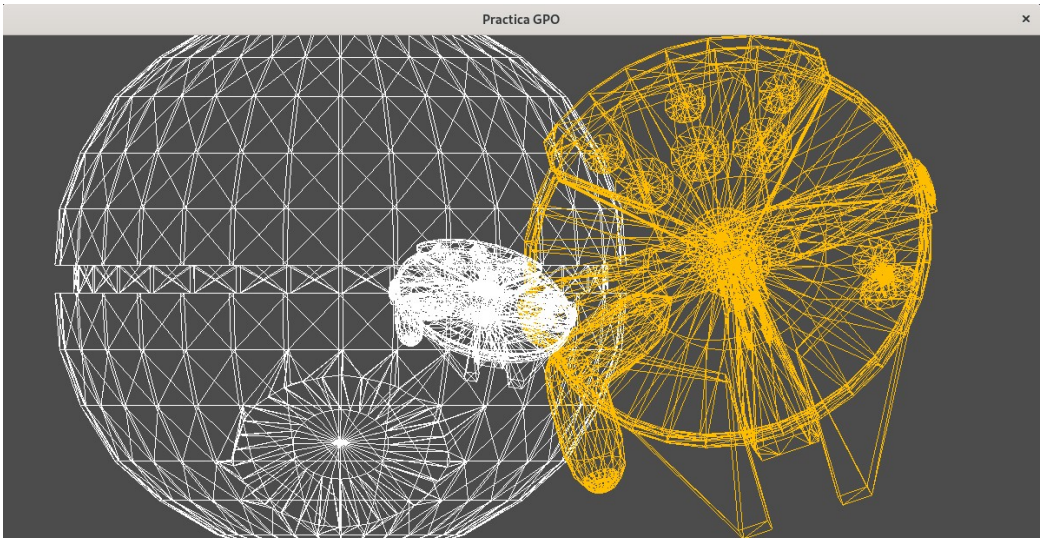

Figure 5: Varios objetos

7 Trabajo futuro

En esta primera parte del proyecto hemos podido ver otra funcionalidad de OpenGL para manipular objetos en un espacio, desde dos puntos de vista, desde el objeto y desde la cámara.

Además, para su realización ha hecho falta comprender conceptos matemáticos sobre espacios geomátricos dados en clase.

References

[1] Transparencias de Egela de la asignatura Gráficos por Computador, *Transparencias-discretización*

[2] OpenGL Programming Guide Eighth Edition - Addison Wesley Pearson