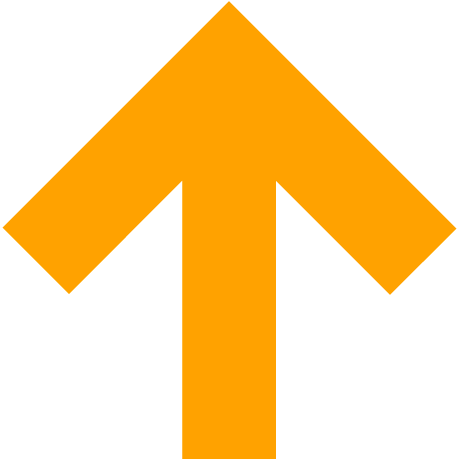
Febrero 2009

|  |
| --- |
| David Flores Barbero  Julio García Valdunciel |



|  |  |
| --- | --- |
| [Informática Gráfica – Ingeniería Informática] | [Tema 6  Manipulación del Espacio: Transformaciones y Proyecciones] |

Contenido

[1 Introducción 1](#_Toc497925887)

[2 Transformaciones 2](#_Toc497925888)

[2.1 Pila de matrices 2](#_Toc497925889)

[2.2 Sistema de referencia 3](#_Toc497925890)

[2.3 Tipos de transformaciones 5](#_Toc497925891)

[2.3.1 Rotación de elementos 5](#_Toc497925892)

[2.3.2 Título 3 10](#_Toc497925893)

[3 Titulo 1 10](#_Toc497925894)

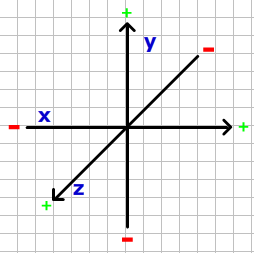
[3.1 Título 2 10](#_Toc497925895)

# Introducción

En este trabajo se pretende exponer la forma en que se puede realizar el dibujado de elementos gráficos tales como puntos, líneas y otros elementos así como realizar modificaciones en la escena resultante aplicando transformaciones. Estas transformaciones que vamos a realizar contarán con rotación, traslación de figuras o el escalado del propio escenario con todos los elementos que este compone.

Un primer concepto a tener en cuenta a la hora de realizar el dibujado de elementos sobre la pantalla es establecer la correspondencia entre las coordenadas de los elementos transformados y la pantalla, debido a que es necesario realizar una transformación en la vista para poder representar correctamente los objetos.

En este trabajo se mostrará cómo se van a colocar los elementos en un sistema de coordenadas tridimensional para su correcta visualización así como las transformaciones que pueden realizarse a dicho objeto en sus dimensiones, posición y orientación.



1. Representación de las coordenadas

# Transformaciones

Una transformación es la acción de asignar un punto/vector transformado a cada punto/vector original del espacio. Por ejemplo: el punto movido o girado de una determinada manera. Se utilizan, por ejemplo, para armar objetos complejos a partir de piezas simples, para posicionar objetos, las luces y la cámara en escena y también para proyectar la escena sobre el plano de la imagen.

Antes de aplicar las transformaciones es necesario saber que openGL actúa como una máquina de estado, es decir lo primero que se lee en el programa se aplica a todo lo que sigue a continuación, por lo tanto es la última acción que se realiza. Este concepto es muy importante tenerlo en cuenta en las transformaciones ya que el orden de realización de las mismas influirá en el resultado final en gran medida.

## Pila de matrices

OpenGL cuenta con tres matrices homogéneas para realizar las transformaciones. Dichas matrices cuentan con cuatro dimensiones. El propósito de las matrices es el siguiente:

* **Matriz 1:** Mapear el modelo al sistema visual.
* **Matriz 2:** Definir proporción del espacio y mapearlo en un cubo canónico. Posteriormente realizará un aplastamiento del mismo.
* **Matriz 3:** Aplicación de texturas.

Para definir la matriz activa en cada momento del proceso utilizaremos la función **glMatrixMode(GLenum mode)** . Las constantes que se pueden utilizar para indicar el tipo de matriz que se utiliza en cada momento son las siguientes: **GL\_MODELVIEW, GL\_PROJECTION** o **GL\_TEXTURE** que se corresponde cada una con las matrices anteriormente descritas.

OpenGL posee además una pila para cada una de las matrices. Es posible guardar el estado de una matriz con push, para posteriormente poder recuperar dicho estado con pop.

**glPushMatrix() glPopMatrix()**

Al realizar las llamadas a dichas funciones la altura de la pila aumentará o disminuirá en función de si añadimos un nuevo estado o si estamos realizando una extracción de uno. Al tratarse de una pila tiene un comportamiento LIFO (Last in First out).

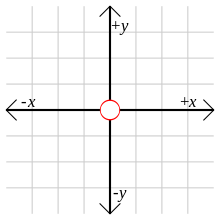
## Sistema de referencia

Para poder proceder a una correcta visualización de los objetos en la escena es necesario poseer un correcto sistema de visualización. Todas las coordenadas de los distintos objetos han de estar dadas en un sistema de referencia igual.

Cuando se aplican las transformaciones es necesario modificar el sistema de coordenadas, con lo que se tendrá un nuevo sistema de coordenadas transformado pero cuyas coordenadas serán las mismas.

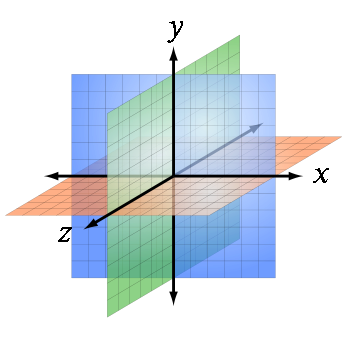
OpenGL utiliza un sistema de coordenadas para la representación de objetos en la pantalla, en el cual las coordenadas estás ubicadas de la siguiente manera:

* La dirección positiva del eje de las X es representada hacia la derecha.
* La dirección positiva del eje de las Y es representada hacia arriba.
* El eje Z corresponde con una línea que une al observador con el eje de coordenadas de manera que los puntos negativos corresponden las mayores distancias con respecto al observador y la dirección positiva con los puntos más cercanos.



1. Representación de las coordenadas del espacio

En la imagen 2 podemos observar como es el sistema de coordenadas si a esta no se le ha aplicado ningún tipo de transformación, lo que quiere decir que el eje Z es perpendicular a la vista del observador, con lo cual no es visible.



1. Representación de las coordenadas del espacio

En la imagen 3 podemos comprobar cómo se mostraría el espacio de coordenadas con un desfase con respecto al eje Y. Esto hace que se posible al observador ver la posición del eje Z.

Cabe destacar que aunque se realicen transformaciones en el espacio de coordenadas las coordenadas de los objetos siguen siendo las mismas ya que el sistema de representación no cambia.

## Tipos de transformaciones

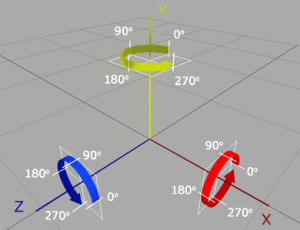
En este apartado vamos a pasar a explicar las distintas transformaciones que pueden realizarse a los objetos. Antes de nada es importante tener en cuenta las siguientes consideraciones:

* Es necesario realizar una llamada a la función **GlMatrixMode** con argumento **GL\_PROJECTION** o **GL\_MODELVIEW** para activar la matriz de transformación.
* Es necesario cargar la identidad con lo cual se realiza una llamada a la función **glLoadIdentity().**

Una vez hemos realizado los pasos anteriores, podemos hablar ya de las tres transformaciones principales que pueden realizarse, las cuales son: rotación, escalado y traslación.

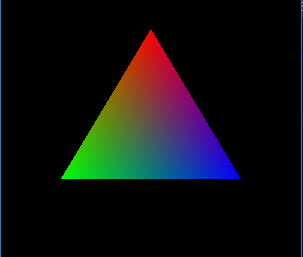
### Rotación de elementos

El fenómeno conocido como rotación se podría describir como el giro de un determinado objeto o conjunto de estos con respecto a un eje. Así podemos definir tres tipos de rotaciones básicas respecto a los ejes principales X, Y, Z. A partir de estos tipos básicos de rotaciones es posible definir combinaciones de los mismos de modo que podemos rotar objetos en torno a varios ejes al mismo tiempo.



1. Representación de las posibles rotaciones que puede realizar un objeto

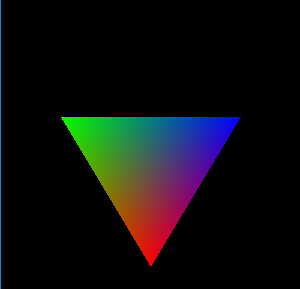
Vamos a realizar un sencillo ejemplo en que aplicaremos distintas rotaciones sobre un objeto. El objeto que vamos a representar será un triángulo cuyas coordenadas serán (0.0,0.8), (-0.6, -0.2), (0.6, -0.2).



1. Figura a representar sin aplicar transformaciones

**EJE X**

Ahora vamos a proceder a realizar una serie de rotaciones de la siguiente figura, empezaremos con una rotación de 180 grados con respecto al eje X:

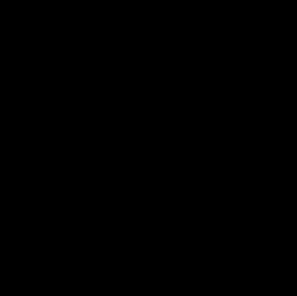


1. Figura a representar con rotación en el eje X

Como podemos comprobar la imagen ha realizado una rotación con respecto al eje de las X de 180 grados lo que nos deja ver el triángulo dado la vuelta ya que se representa media vuelta. La función que hemos utilizado para realizar esta rotación es:

***glRotated(180, 1, 0, 0);***

Hay que tener en cuenta la orientación del sistema de coordenadas ya que si realizamos un giro de 90 grados el resultado es el siguiente:



1. Figura a representar con rotación en el eje X

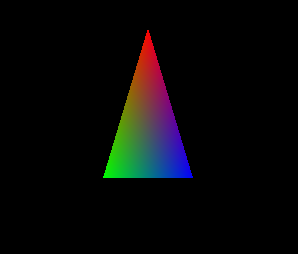
Como se puede observar no se representa nada esto es porque al representar una figura en 2D y realizar una rotación de manera que la misma quede perpendicular al observador este no es capaz de visualizarla.

***glRotated(90, 1, 0, 0);***

**EJE Y**

En esta ocasión vamos a realizar rotaciones con respecto al eje de las Y con la salvedad de que excluimos el caso del giro de 90 grados ya que como vimos en el apartado anterior esto resultará en un modelo perpendicular a la visión del espectador y cuya visibilidad no será posible.

Vamos a probar el caso en el que establecemos una rotación de 60 grados con respecto al eje de las Ys.

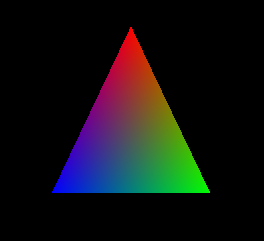


1. Figura a representar con rotación en eje Y

En este caso podemos comprobar que la realización del giro de 60 grados deja la figura deformada como es apreciable por la distancia de separación de los vértices inferiores del mismo.

***glRotated(90, 0, 1, 0);***

Si probamos a realizar un giro de 120 grados en el eje de las Y podemos comprobar como el triángulo queda orientado de la misma forma que el original pero con los vértices inferiores cambiados (colores verde y azul).

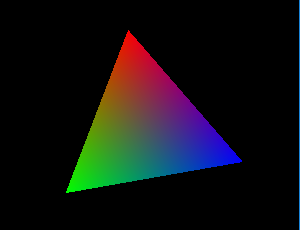


1. Figura a representar con rotación en eje Y

***glRotated(120, 0, 1, 0);***

**EJE Z**

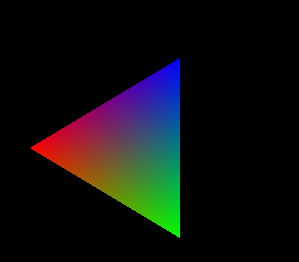
Las rotaciones con respecto al eje de la Z en condiciones normales son las más visibles al observador debido a que al estar en perpendicular al espectador este puede apreciar de manera más efectiva las rotaciones.

****

1. Figura a representar con rotación en el eje Z

La anterior imagen representa una rotación de la figura de 10 grados con respecto al eje Z y como se ha comentado anteriormente estas rotaciones son más visibles que las de los otros ejes a ojos del observador.

***glRotated(10, 0, 0, 1);***



1. Figura a representar con rotación en eje Z

Y por último, la imagen anterior representa un triángulo con una rotación de 90 grados lo que hace que se encuentre en una posición fuera del equilibrio, ya que para completar un giro completo el triángulo necesitaría de 120 grados (resultado de la división de 360 grados entre el número de vértices de la figura en cuestión).

***glRotated(90, 0, 0, 1);***

<http://www.cimec.org.ar/~ncalvo/transformaciones_practica_doc.pdf>

<https://www.fing.edu.uy/inco/cursos/compgraf/Clases/2012/05-Transformaciones%20Geometricas%5bopenGL%5d.pdf>

<http://albertojaspe.net/uploads/articulos/tutorial-opengl.pdf>

<http://castroarellano.blogspot.com.es/2013/02/piramide-triangular-opengl-jesus-castro.html>

### Título 3

#### Titulo 4

##### Título 5

# Titulo 1

## Título 2

[Primer párrafo] El primer párrafo debe estar sin tabular siguiendo el formato [Primer párrafo].



1. Texto de la figura

[normal] Las figuras hay que ponerlas como imagen centrada para que no se separe el texto de la figura de la imagen. Después del primer párrafo usar el estilo normal. Las figuras que pongáis en el texto deben de ser propias en la medida de lo posible o al menos que no tengan derechos de autor y no deben incluir texto en inglés.

Las fórmulas hay que crearlas como fórmulas no las pongáis como imágenes.

Los ejemplos de código que se añadan tienen que ser originales (no copiar ejemplos de libros), estar suficientemente documentados y tienen estar referenciados con el fichero en el que se encuentran. En los docs, se deberá introducir la parte importante de los ejemplos. Los ejemplos hay que almacenarlos en un directorio con el nombre del tema y en subdirectorios con el nombre del ejemplo. Para referenciarlos poner entre corchetes la ruta relativa (sin poner el nombre del directorio del tema) y el nombre del fichero fuente como primera línea del código. Ejemplo.

[figura/figura.cpp]

class Figura

{

Protected:

Public:

void funcion(int param1, int param2, int param3, int param4, int param5, int param6, int param7)

};

Para las tablas seguir el siguiente formato [Tabla encabezado] para la cabecera y [Tabla] para el contenido.

|  |  |  |
| --- | --- | --- |
|  | Encabezado | **Encabezado** |
| Fila 1 | Fila 1 | Fila 1 |
| Fila 2 | Fila 2 | Fila 2 |

1. Nombre de la tabla

Hay que referenciar todas las tablas y figuras que se pongan en el texto, además de referenciar hay que explicarlas. Para referenciar los elementos hacerlo con referencias cruzadas para evitar problemas.

Finalmente, tenéis que actualizar la información de los Power Points con los ejemplos, imágenes y demás cosas que hayáis hecho.

Importante: No modificar los estilos ni introducir nuevos para que todos los trabajos queden similares.