Transformaciones geométricas en OpenGL

Inicialización de las transformaciones geométricas

- **`GlMatrixMode(...);`**

  /* Para activar la matiz de transformación */

  - **`GL_PROJECTION`**
  - **`GL_MODELVIEW`**

- **`glLoadIdentity();`**

  /* Para cargar la identidad y así poder hacer todas las transformaciones necesarias */

Tres transformaciones principales:

- **`glScalef(GLfloat sx, GLfloat sy, GLfloat sz);`**

  /* Escalar según sean los factores sx, sy y sz*/

- **`glTranslatef(GLfloat tx, GLfloat ty, GLfloat tz);`**

  /* Trasladar según los factores tx, ty y tz */

- **`glRotatef(GLfloat angulo, GLfloat vx, GLFloat vy, GLfloat vz);`**

  /* Rotar "angulo" según el eje que define el vector (vx,vy,vz) */

Composición de transformaciones

¡¡ATENCIÓN!!

- En la siguiente secuencia de código:

```
glScalef(...);
glRotatef(...);
glTranslatef(...);
```

- Primero se ejecuta la traslación, luego la rotación y por último el escalamiento.

Almacenamiento de las matrices de transformación

- Pueden existir varias matrices de transformación.
- OpenGL permite guardar el estado de la matriz para luego volver a utilizarlo de la siguiente manera:

- **`glPushMatrix( );`**

  /* Salva el estado actual de la matriz */

- **`glPopMatrix( );`**

  /* Recupera el estado de la matriz */

- Funciona como una pila clásica de programación.

Ejemplo de almacenamiento de las matrices de transformación

```
/*Transformaciones que aplican a toda la geometría*/
glRotatef(...);
glTranslatef(...);

glPushMatrix( ); /*Guardo el estado de la matriz*/

/*Transformaciones que aplican a la geometría específica*/
glTranslatef(...);
glScalef(...);

dibujo_geometría_específica( ); /* Render de la geometría que
                                   pasará por 4 transformaciones */

glPopMatrix( );                 /* Recupero el estado de la matriz anterior */

dibujo_el_resto( );             /* Render de la geometría que pasará por las
                                   2 primeras transformaciones */
```

Transformación geométrica a partir de una matriz dada

- Dada una matriz se puede aplicar directamente la transformación asociada.
- Se tienen dos primitivas:

```
glLoadMatrixf(puntero_a_matriz);
```

/* Setea la matriz que se le pasa como parámetro */

```
glMultMatrixf(puntero_a_matriz);
```

/* Multiplica la matriz actual por la pasada como parámetro */

Punteros a matriz

- El puntero a una matriz es una variable de uno de los siguientes tipos:

```
GLfloat M[16];
```

o

```
GLfloat M[4][4];
```

Representación de matrices

- Es importante destacar que las matrices utilizadas por openGL están ordenadas en forma de columnas.

- Para la siguiente matriz

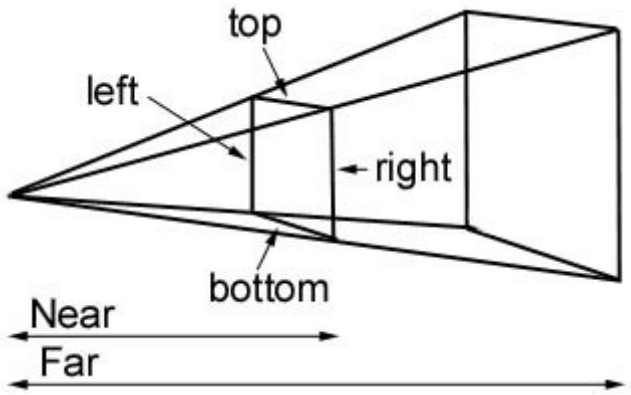
```
|a00 a04 a08  a12|
|a01 a05 a09  a13|
|a02 a06 a10  a14|
|a03 a07 a11  a15|
```

- El vector utilizado para su representación es:

```
[a00, a01, a02, ... a14, a15]
```

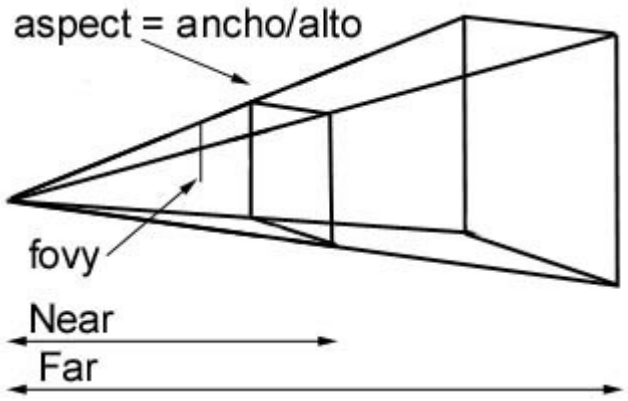
Proyección perspectiva

- Para establecer la proyección perspectiva lo que tenemos que hacer es
- glMatrixMode(GL_PROJECTION); // Empezamos a mostrar como se va a realizar la visualización
- glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);



Proyección perspectiva

- Otra forma de especificar lo mismo es con la siguiente función:
- void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);
- //fovy es el ángulo en el plano xz y varía en el siguiente intervalo: [0.0,180.0]



Proyección Ortogonal

- Otra manera de visualizar la geometría es proyectando ortogonalmente, para esto se dispone de la siguiente primitiva.
- void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble far);