2. TRANSFORMACIONES: DANDO ÓRDENES A LA TORTUGA (FORWARD, RIGHT, LEFT, UP Y DOWN)

En el presente capítulo se introduce el concepto de transformación y de las matrices que las definen, tanto de transformaciones de modelado como de proyección. Se utilizarán posteriormente estos conceptos para implementar las primeras órdenes de Logo: FORWARD, RIGHT, LEFT, UP y DOWN.

2.1 TRANSFORMACIONES DE MODELADO Y DE PROYECCIÓN

La representación de las primitivas y de los objetos se realiza transformando las coordenadas originales. Estas transformaciones pueden originarse debido a cambios en el modelo o a las propiedades de la cámara. Las propiedades de la cámara se verán en el siguiente capítulo. En el presente capítulo veremos como afectan los cambios al modelo.

OpenGL dispone de tres matrices para realizar el proceso. Se especifican por los nombres:

GL_MODELVIEW: la matriz que contiene las transformaciones originadas por los cambios de modelado y posición de la cámara.

GL_PROJECTION: la matriz con las transformaciones que realizan la proyección de la escena de 3 a 2 dimensiones.

GL_TEXTURE: para transformaciones en las coordenadas de textura.

Por ello, antes de realizar una operación de transformación es necesario indicar sobre que matriz se va a realizar. Se especifica con la función **glMatrixMode**(Glenum mode) que tiene como argumento una de las tres constantes enumeradas. Se comporta como un estado, por tanto, hasta que se especifique un nuevo estado todas las transformaciones se realizan sobre la última matriz especificada.

En el código de la función **resaphe**() del capítulo anterior:

```
void reshape(int width, int height) {
  glViewport(0, 0, width, height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(60.0, (GLfloat)height / (GLfloat)width, 1.0, 128.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}
```

Se observa que se utiliza la función **glMatrixMode**() dos veces, la primera con *GL_PROJECTION* y la segunda con *GL_MODELVIEW* .

Despues de la primera llamada a **glMatrixMode**(), la matriz sobre la que se realizarán las transformaciones es *GL_PROJECTION*, la primera operación es inicializar la matriz con la función **glLoadIdentity**() que carga la matriz identidad y se define una proyección perspectiva con la función **gluPerspective**(). Esta función se explica en el siguiente capítulo.

Despues de la segunda llamada a **glMatrixMode**(), la matriz sobre la que se realizarán las transformaciones es *GL_MODELVIEW*, igualmente, la primera operación es inicializar la matriz con la función **glLoadIdentity**() y a continuación se establece la posición de la cámara con gluLookAt(). Esta función se explicará con más detalle en el siguiente capítulo.

Al comportarse OpenGL como una máquina de estados, las siguientes operaciones de transformación que se realicen en el código, que estarán fuera de la función **resaphe**(), se realizarán sobre la última matriz especificada, es decir sobre *GL_MODELVIEW*.

2.2 INTERPRETANDO LOS COMANDOS

La aplicación TecnunLogo lee comandos introducidos por el teclado y los interpreta para su ejecución. Los comandos que inicialmente interpreta son los correspondientes a avanzar, retroceder, girar a la derecha, girar a la izquierda, girar hacia arriba y girar hacia abajo. En la siguiente tabla se muestran estos comandos, con la descripción y la abreviatura que se utiliza en el interprete:

Comando	Abreviatura	Descripción	Argumento
FORWARD	fd	Avanza hacia adelante	Unidades de distancia
RIGHT	rt	Gira a la derecha	Grados (0 – 360º)
LEFT	lt	Gira a la izquierda	Grados (0 – 360º)
BACK	bk	Retrocede	Unidades de distancia
UPPITCH	up	Gira hacia arriba	Grados (0 – 360º)
DOWNPITCH	down	Gira hacia abajo	Grados (0 – 360º)
EXIT	exit	Sale del modo logo	-
HOME	home	Posicionarse en el inicio	-

La función parseCommand realiza la interpretación del comando y llama a la función de OpenGL correspondiente.

```
void parseCommand(char* strCommandParse) {
    char *strToken0;
    char *strToken1;
    double val;
    strToken0 = strtok(strCommandParse, " ");
    while ((strToken1 = strtok(NULL," ")) != NULL) {
        val = atof(strToken1);
        if (!strcmp("fd",strToken0)) {  // FORWARD
          glTranslatef(0.0, 0.0, val);
        } else if (!strcmp("bk",strToken0)) { // BACK
          glTranslatef(0.0, 0.0, -val);
        } else if (!strcmp("rt",strToken0)) { // RIGHT
          glRotatef(-val,0.,1.,0.);
        } else if (!strcmp("lt",strToken0)) { // LEFT
          glRotatef(val,0.,1.,0.);
        } else if (!strcmp("up",strToken0)) { // UP
          glRotatef(val,1.,0.,0.);
        } else if (!strcmp("dn",strToken0)) { // DOWN
          glRotatef(-val,1.,0.,0.);
        }
        strToken0 = strtok(NULL, " ");
        display();
    }
    // EXIT COMMAND MODE
    if (strToken0 != NULL && strncmp(strToken0, "exit", 4) == 0) {
        command = FALSE;
    // HOME
```

```
    } else if (strToken0 != NULL && !strcmp("home",strToken0)) {
        glLoadIdentity();
    }
}
```

Se emplea la función char* **strtok**(s, ct) de la librería <string.h> que busca elementos en la cadena "s", separados por los carácteres "ct". En este caso los caracteres separadores es el espacio en blanco.

El primer elemento "strToken0" es la instrucción Logo y el segundo elemento "strToken1" es el argumento. A continuación se compara el texto de la instrucción con las distintas posibilidades (fd, rt, lt, ...) y se ejectua la acción correspondiente. En este caso, el cuerpo de la acción muy reducido, se inserta a continuación en lugar de realizar una función separada.

Para introducir las instrucciones Logo, se debe pasar al modo interprete. Esto se realiza pulsando la tecla "i", con lo que los caracteres introducidos a continuación se almacenan en una cadena hasta que se pulsa la tecla retorno (ASCII: 13).

Se añade la librería stdio.h y las variables globales command y strCommand que indican respectivamente si está en modo comando y la cadena de texto introducida hasta el momento:

```
#include <stdio.h>

boolean  command = FALSE; /* command mode */
char strCommand[256];
```

La función **keyboard**() posee el siguiente código:

```
void keyboard(char key, int x, int y) {

  if (command) {
    if (key == 13) {
      strcat(strCommand, " ");
      if (strlen(strCommand) == 1) command = FALSE;
      parseCommand(strCommand);
      strcpy(strCommand, "");
    } else {
      char strKey[2] = " ";
      strKey[0] = key;
      printf(strKey);
      strcat(strCommand, strKey);
    }
  } else {  // not in command mode
    switch (key) {
      case 'h':
        ...
      case 'i':
        command = TRUE;
        break;
        ...
      case 27:
        exit(0);
        break;
    }
  }
  glutPostRedisplay();
}
```

Se emplea la variable booleana “command” para determinar si está en modo interprete o no. Al pulsar la tecla “i” se activa el modo interprete.

Primeramente se comprueba si está en modo interprete:

```
case 'i':
  command = TRUE;
```

Si se está en modo interprete, las teclas pulsadas se controlan en el bloque “if (command)”, en el cual se detemina si se ha pulsado la tecla retorno, en cuyo caso se procede a ejecutar el comando o si no, se añade el carácter a la cadena de caracteres que va almacenando la instrucción. Si se pulsa la tecla retorno, sin haber ninguna instrucción, se vuelve al modo interactivo, en el cual determinadas teclas tiene asociadas distintas funciones.

Las distintas instrucciones que reconoce el interprete se corresponden con las sentencias de OpenGL para variar la matriz *GL_MODELVIEW*. Las tres funciones de OpenGL para realizar transformaciones de modelado son **glTranslate**(), **glRotate**() y **glScale**(). Cada una de ellas puede tener argumentos en precisión simple (float) o doble (double), como por ejemplo glTranslatef() y glTranslated()

2.3 TRASLACIÓN

Realiza una traslación a las coordenadas x, y, z. Se realiza con la función glTraslate() que tiene como argumentos estos tres valores. La función realiza la multiplicación de la matriz actual por la matriz de traslación que se compone con estos tres valores.

En la aplicación de TecnunLogo se utiliza en los comandos FORWARD Y BACK

```
        if (!strcmp("fd",strToken0)) {  // FORWARD
          glTranslatef(0.0, 0.0, val);
} else if (!strcmp("bk",strToken0)) { // BACK
          glTranslatef(0.0, 0.0, -val);
```

Se realiza un desplazamiento según el eje Z, que es el eje de visualización en sentido positivo para FORWARD o negativo para BACK. En la figura 1 se muestra la tortuga en su posición inicial y en la figura 2 una vez trasladada, en esta figura se ha conservado también la posición inicial.

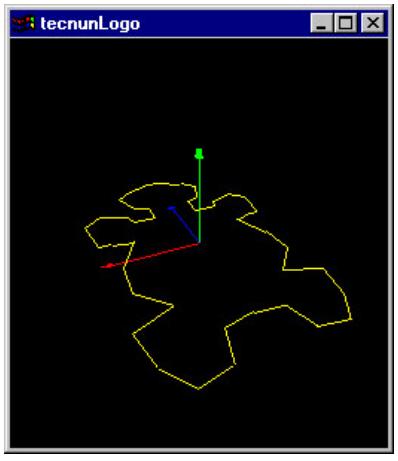

Figura 1 Posición inicial

Figura 2 Posición inicial y trasladada

2.4 ROTACIÓN

La función de OpenGL para aplicar una rotación es glRotate(). Tiene cuatro argumentos: el primero es el ángulo que se desea girar en grados sexagesimales en sentido

contrario a las agujas del reloj; los siguientes tres argumentos definen las coordendas x, y, z del eje alrededor del cual se realiza el giro. El eje pasa por el origen de coordendas, por lo tanto, si el objeto está alejado de este eje, se producirá un desplazamiento causado por la rotación, es decir, no solo girará. Si de desea evitar este desplazamiento será necesario primeramente trasladar el objeto al origen, realizar la rotación y realizar la traslación inversa.

La función **glRotate**() multiplica la matriz actual por la matriz de rotación creada con estos cuatro argumentos.

En la aplicación de TecnunLogo se utiliza en los comandos RIGTH, LEFT, UP Y DOWN:

```
} else if (!strcmp("rt",strToken0)) { // RIGTH
  glRotatef(-val,0.,1.,0.);
} else if (!strcmp("lt",strToken0)) { // LEFT
  glRotatef(val,0.,1.,0.);
} else if (!strcmp("up",strToken0)) { // UP
  glRotatef(val,1.,0.,0.);
} else if (!strcmp("dn",strToken0)) { // DOWN
  glRotatef(-val,1.,0.,0.);
```

En las instrucciones RIGTH y LEFT se realiza el giro alrededor del eje verical "Y" y en las instrucciones UP Y DOWN alrededor del eje horizontal "X". En la figura 3 se muestra el efecto de un giro a la derecha 2rt 90".

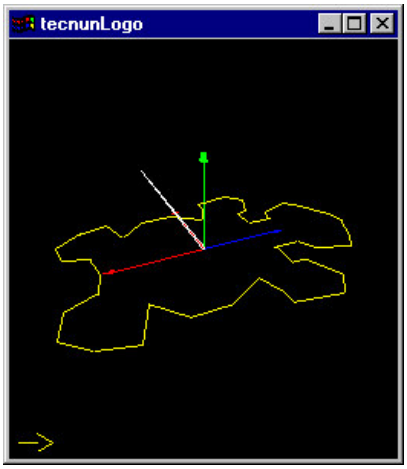

Figura 3 Giro a la derecha

Implementar las instrucciones RIGHTROLL y LEFTROLL con las abreviaturas rr y lr, que realizan un giro alrededor del eje "Z".

2.5 ESCALADO

Además de poder situar el objeto en distintas posiciones mediante traslaciones y rotaciones, OpenGL dispone de la función **glScalef**() para cambiar el tamaño de los objetos.

Posee 3 argumentos que son los factores de escala para cada uno de los tres ejes, de manera que si estos valores no son iguales, además de cambiar el tamaño, se realiza una deformación del objeto. Los factores mayores de 1 aumentan las coordenadas y los menores de 1 las disminuyen.

La función **glScalef**() multiplica la matriz actual por la matriz de escala generada.

El escalado se realiza a partir del origen, por lo que las nuevas coordenadas se calculan a partir de este sistema de referencia. Por ejemplo si hay un cubo definido en el eje X entre las coordendas 3 y 4, al aplicarle una escala de factor 2 según este eje, el cubo se encontrará ahora situado entre las coordenadas 4 y 6. Si se quiere evitar este desplazamiento, al igual que se ha indicado en la rotación, será necesario primero realizar una traslación, aplicar el cambio de escala deseado y finalmente realizar la traslación inversa.

Incluir en la aplicación las instrucciones para cambiar el tamaño del objeto: SCALEX, SCALEY y SCALEZ, con las abreviaturas sx, sy, sz. El argumento que se proporciona es el factor de escala. Para disminuir el tamaño basta emplear un factor menor a la unidad.

Los nuevos comandos introducidos son:

Comando	Abreviatura	Descripción	Argumento
RIGHTROLL	rr	Rota hacia la derecha según el eje de avance	Grados (0 – 360º)
LEFTROLL	lr	Rota hacia la izquierda según el eje de avance	Grados (0 – 360º)
SCALEX	sx	Multiplica las coordenadas x por el factor de escala	Factor de escala
SCALEY	sy	Multiplica las coordenadas y por el factor de escala	Factor de escala
SCALEZ	sz	Multiplica las coordenadas z por el factor de escala	Factor de escala

2.6 ORDEN DE LAS TRANSFORMACIONES

Las transformaciones de coordenadas no disponen de la propiedad conmutativa (si la disponene si sólo se realizan traslaciones o sólo cambios de escala). Se puede comprobar al ver el efecto de una traslación seguida de una rotación el efecto del proceso en el orden inverso: primero la rotación y después la traslación.

Si se realizan las pruebas con la aplicación desarrollada hasta ahora, por ejemplo "rt 90 fd 2", esta se comporta como uno espera que se comporte la tortuga de Logo, es decir, si se realizado un giro a la derecha y a continuación se da la orden de avanzar, se realiza la traslación según la dirección en la que está mirando actualmente la tortuga. Sin embargo, si pensamos en las funciones OpenGL a las que llama el interprete, se están dando las órdenes:

```
glRotatef(-90,0.,1.,0.);
glTranslatef(0.0, 0.0, 2);
```

Con lo cual, si realizamos mentalemente estas transformaciones, la primera gira el objeto en eje "Y", en concordancia con lo que realiza la aplicación, pero la segunda transformación está dando la orden de trasladar según el eje Z, es decir que la tortuga se desplazaría derrapando en dirección Z. ¿Porqué la tortuga entonces avanza según la nueva dirección? La respuesta es que en OpenGL las transformaciones se producen en el orden inverso en el que se han definido. Al ver el efecto de las nuevas transformaciones, puede parecer más sorprendente, porque ahora lo que ocurre es que primero se produce la traslación y despues de la rotación. Sin embargo, si se realizan los dos movimientos se puede comprobar que la posición final es la misma que se espera obtener pensando como la tortuga: se realiza una tranlación en sentido Z de dos unidades, se gira alrededor del eje Y –90 grados, con lo que el objeto situado en el eje Z pasa a estar situado en el eje X. Se puede comprobar que se produce el mismo resultado independiente del número y tipo de las transformaciones.

2.7 EJEMPLO DE BUCLE REPEAT

Una instrucción que añade bastante potencia a Logo es REPEAT, que permite realizar bucles. La sintaxis es:

REPEAT n [instrucciones]

Este comando realiza n veces las instrucciones escritas entre corchetes. Admite anidaciones de cualquier nivel como en todos los lenguajes de programación.

En la aplicación tecnunLogo se implementa este comando con una inclusión en la función **parseCommand**() y la función **insideRepeat**():

```
char * insideRepeat(char* strCommandInside) {
      char *ini, *fin;
      ini = strchr(strCommandInside,'[');
      if (ini == NULL) return NULL;
      ini++;
      fin = strrchr(strCommandInside,']');
      if (fin == NULL) return NULL;
      strCommandInside[fin-strCommandInside]=0;
      return ini;
}
```

En **parseCommand**() se incluye antes del if de comprobación de las instrucciones existentes, la comprobación de "repeat".

```
    if (!strcmp("repeat",strToken0)) {
      repeatCommand = insideRepeat(strToken1 + strlen(strToken1) + 1);
      if (repeatCommand == NULL) return;
      nextCommand = repeatCommand + strlen(repeatCommand) + 1;
      for (i = 0; i < val; i++) {
        strcpy (parseCommandInit, repeatCommand);
        parseCommand(parseCommandInit);
      }
      strToken0 = strtok(nextCommand, " ");
      if (strToken0 == NULL) continue;
      continue;
    }
```

Se incluyen tambien las declaraciones correspondientes:

```
    char *repeatCommand;
    char *nextCommand;
    char parseCommandInit[256];
    int i;
```

2.8 PUNTOS A REALIZAR

Dibujar unos ejes del mundo y otros en el sistema de la tortuga como los que se muestran en la figura 2.

Utilizando los comandos de logo dar las instrucciones para que la tortuga se desplaze en una circunferencia.

Leer comandos desde un fichero, la instrucción se define "LOAD nombreFichero", el fichero contiene 1 o varias líneas de sentencias de logo.

Definir el sistema de medida de ángulos, de forma que en vez de utilizar el sistema sexagesimal de 0 a 360º, se pueda utilizar cualquier otro; como radianes de 0 a 2π; grados centesimales, de 0 a 400 o cualquier otro como de 0 a 12 o 0 a 60 como las horas o minutos de un reloj. Utilizar para ello el comando SETCIRCLE nn, siendo nn el número correspondiente a la medida de un giro completo. Por defecto el valor es 360. La abreviatura del comando es sc.

Implementar los comandos HIDETURTLE y SHOWTURTLE, con las abreviaturas ht y st que muestran u ocultan la torguga.