Práctica 3. Uso de GLUT. Parte 2

Creando Interfaces de Usuario

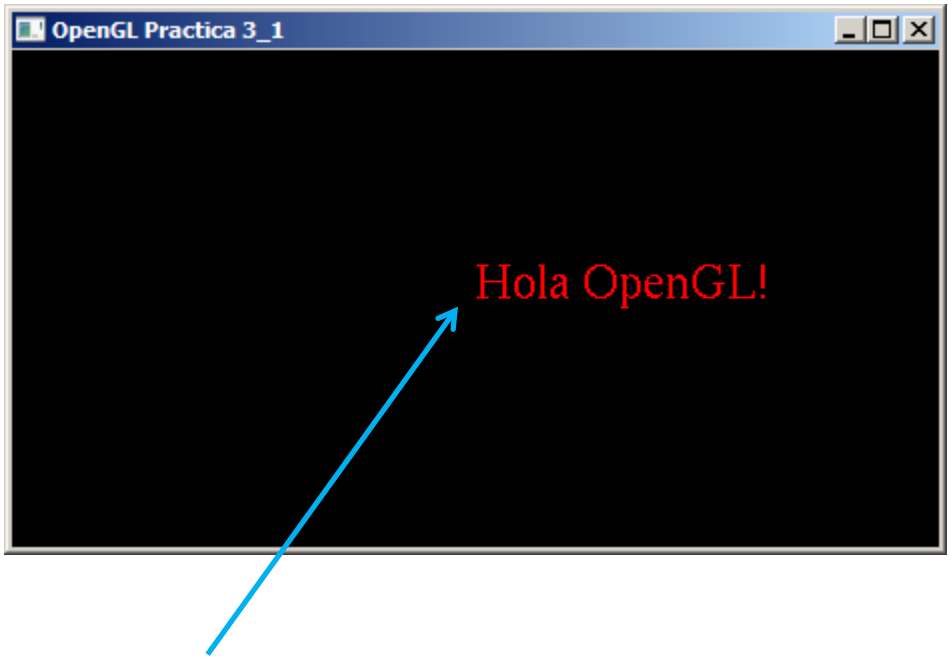
Grado en Ingeniería Informática. Mención Computación
Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Contenidos

- Dibujar/Render textos en la ventana OpenGL.
- Definir menús.
- Dibujar/Render objetos básicos GLUT.
- Construir animaciones sencillas.

Tarea 1. Dibujar textos.

Utilizar los textos bitmap, tal y como se indica en el Tema de Teoría.



El procedimiento requiere definir el punto de inserción del texto, y a partir de ese punto se insertan los caracteres de izquierda a derecha secuencialmente.

Se puede seleccionar el font solamente de entre los disponibles

El color es definible previamente al dibujo/render del string

Procedimiento

Construir una aplicación a partir del prototipo neutro que se les suministra a los alumnos en el Campus Virtual. Únicamente es necesario añadir una función de escritura del string y modificar la función de Display();

```
void writeBitmapString(float x,float y,void *font, char *string){

    char *c;
    glRasterPos2f(x, y);
    for (c = string; *c != '\0'; c++) glutBitmapCharacter(font, *c);

}

void writeStrokeString(float x, float y, void *font, char *string){

    char *c;

    glPushMatrix();

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(x, y, 0);
    glScalef(1.0/300.0,1.0/300,1.0/300.0); // los caracteres stroke son muy grandes, hay que reducirlos
    for (c = string; *c != '\0'; c++) glutStrokeCharacter(font, *c);

    glPopMatrix();
}
```

GLUT_STROKE_ROMAN

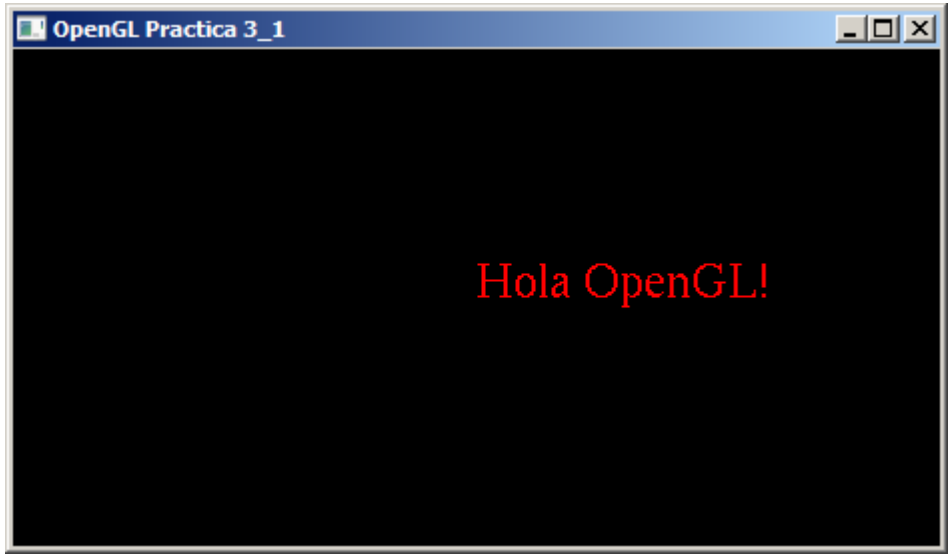
A proportionally spaced Roman Simplex font for ASCII characters 32 through 127. The maximum top character in the font is 119.05 units; the bottom descends 33.33 units.

Version bitmap

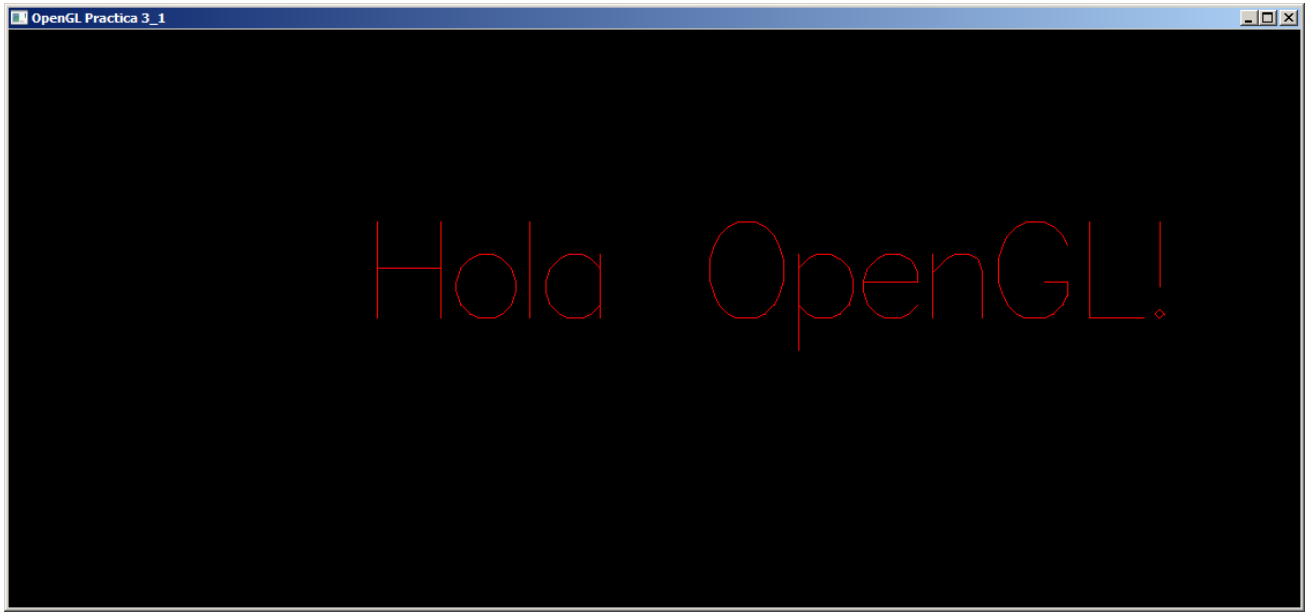
```
// función de gestion de ventana
void Display(){

    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer
    // TO DO

    glColor3f(1.0, 0.0, 0.0);
    writeBitmapString(0.0,0.0,GLUT_BITMAP_TIMES_ROMAN_24, "Hola OpenGL!");
    glFlush(); // actualiza el framebuffer
}
```



Version Stroke



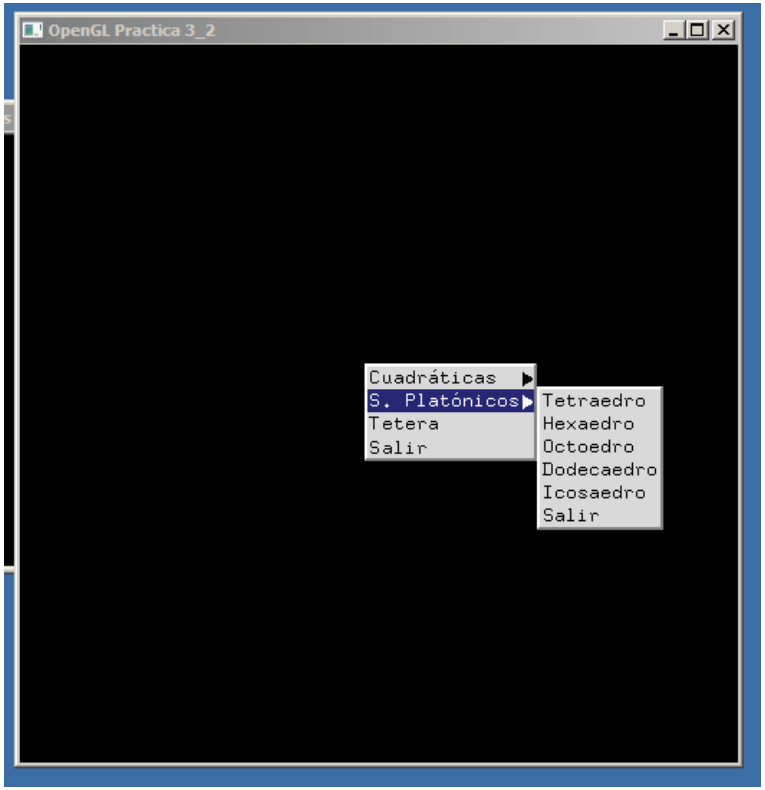

```
// función de gestion de ventana
void Display(){

    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer
    // TO DO

    glColor3f(1.0, 0.0, 0.0);
    writeStrokeString(-1.0f, 0.0f, GLUT_STROKE_ROMAN, "Hola OpenGL!");
    glFlush(); // actualiza el framebuffer
}
```

Tarea 2. Crear Jerarquía de menús

Relazaremos una tarea que crea una estructura de menús que nos permitirá visualizar las formas cuadráticas (esfera, cono, cilindro y toro), la tetera y los objetos geométricos de la famila "Edros" también denominados Sólidos Platónicos:
https://es.wikipedia.org/wiki/S%C3%B3lidos_plat%C3%B3nicos



La aplicación presenta algún defecto en Windows, no realiza la transición entre el menú y la figura de forma directa, si no tras un redisplay, por ejemplo con pique de ratón.

Para ilustrarlo hemos incluido un traceado de las llamada a reshaphe y display.

El menú raíz se construye al final, y se incluirá Salir en cada menú.

Hemos incluido el uso de doble buffer

Cabecera y gestor de menú

```
#include <stdio.h>

#include <GL\glew.h>
#include <GL\freeglut.h>

// Espacio para las variables globales de la ventana
float gl_ancho = 2.0, gl_alto= 2.0, gl_cerca=-1.0, gl_lejos=5.0;
int w_ancho=500, w_alto=500;

#define SALIR           -1
#define VER_NADA        0
#define VER_ESFERA      1
#define VER_TORO        2
#define VER_CONO        3
#define VER_CILINDRO    4

#define VER_T4          5
#define VER_T6          6
#define VER_T8          7
#define VER_T12         8
#define VER_T20         9
#define VER_TETERA      10

// Espacio para otras variables globales
int visualiza=VER_NADA;
int win1;
int menu1,menu2,menu3;
```

```
void gestor_menu(int value){

    if (value == SALIR){
        glutDestroyWindow(win1);
        exit(0);
    }
    visualiza = value;
    glutPostRedisplay();
}
```

La variable global visualiza es utilizada por Display()

Se ha implementado un gestor de menú único para todos con una variable para seleccionar el caso correspondiente

Main

```
void gestor_menu(int value);

//main
int main(int argc, char *argv[]){

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(w_ancho, w_alto);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);


    win1=glutCreateWindow("OpenGL Practica 3_2");
    InitGlew(); // despues de crear la primera ventana
    InitGL();
    glutDisplayFunc(Display); // registra la funcion de rendering
    glutReshapeFunc(ReshapeSize);

    menu1 = glutCreateMenu(gestor_menu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutAddMenuEntry("Esfera", VER_ESFERA);
    glutAddMenuEntry("Toro", VER_TORO);
    glutAddMenuEntry("Cono", VER_CONO);
    glutAddMenuEntry("Cilindro", VER_CILINDRO);
    glutAddMenuEntry("Salir", SALIR);

    menu2 = glutCreateMenu(gestor_menu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutAddMenuEntry("Tetraedro", VER_T4);
    glutAddMenuEntry("Hexaedro", VER_T6);
    glutAddMenuEntry("Octoedro", VER_T8);
    glutAddMenuEntry("Dodecaedro", VER_T12);
    glutAddMenuEntry("Icosaedro", VER_T20);
    glutAddMenuEntry("Salir", SALIR);

    menu3 = glutCreateMenu(gestor_menu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutAddSubMenu("Cuadráticas", menu1);
    glutAddSubMenu("S. Platónicos", menu2);
    glutAddMenuEntry("Tetera", VER_TETERA);
    glutAddMenuEntry("Salir", SALIR);


    glutMainLoop(); // bucle principal

    return 0;
}
```

Se crean los menús para poder ser utilizados independientemente. Solo el último es el activo.

Se crean primeramente los menús “hojas”

Se crea el menú de mayor jerarquía incluyendo sub-menús o entradas

```
// función de gestion de ventana
void Display(){
    // borra todo lo existente en el framebuffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // TO DO

    switch (visualiza){
    case VER_ESFERA:
        glutWireSphere(0.5, 20, 20);
        break;
    case VER_TORO:
        glutWireTorus(0.2, 0.6, 20, 20);
        break;
    case VER_CONO:
        glutWireCone(0.4,1.0,20,20);
        break;
    case VER_CILINDRO:
        glutWireCylinder(0.4,1.0,20,10);
        break;

    case VER_T4:
        glutWireTetrahedron();
        break;
    case VER_T6:
        glutWireCube(0.5);
        break;
    case VER_T8:
        glutWireOctahedron();
        break;
    case VER_T12:
        glutWireDodecahedron();
        break;
    case VER_T20:
        glutWireIcosahedron();
        break;

    case VER_TETERA:
        glutWireTeapot(0.5);
        break;

    default:;
    }
    glutSwapBuffers(); // actualiza el doble buffer
    //glFlush(); // actualiza el framebuffer
    printf("Display\n"); // <- para comprobar que se ejecuta Display
}
```

Display

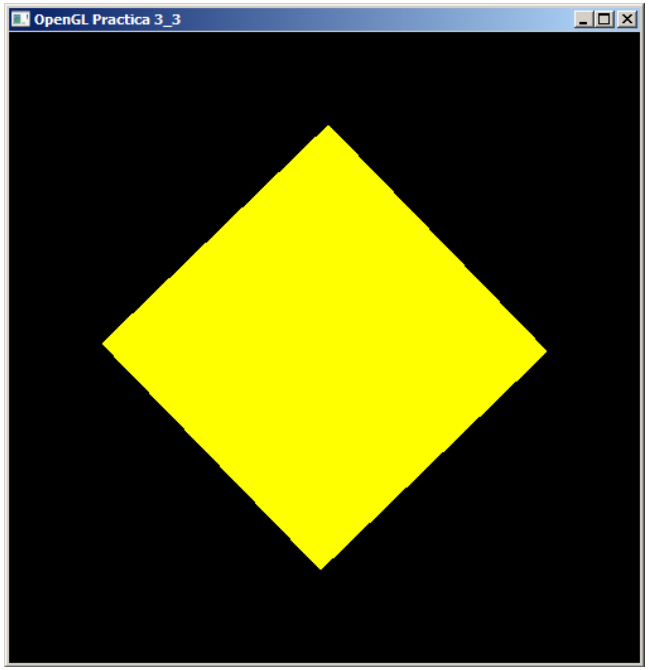
Utiliza la variable global visualiza en un switch

La visualización de cada figura no es perfecta, por diversas razones: no existe perspectiva sino proyección ortográfica y las dimensiones y orientación de los ejes de cada figura es la de defecto.

Con los conocimientos de los próximos temas podríamos mejorar estos aspectos. De momento solo nos interesa saber utilizar menús.

Tarea 3. Animación, ratón y Timer.

Incluiremos un ejemplo de animación, consistente en un cuadrado que rota automáticamente y que podemos parar/rotar con un click de ratón. Es un ejemplo inspirado en el demo Spinning Square del tutorial:
http://cs.lmu.edu/~ray/notes/openglexamples/



Hemos cambiado un poco el demo e introducido dos sentidos de giro seleccionables con los botones del ratón. La animación utilizar un Timer y doble buffer.

El cuadrado no rota, es el sistema de coordenadas del modelo el que lo hace.

Cabecera

```
#include <stdio.h>

#include <GL\glew.h>
#include <GL\freeglut.h>

// Espacio para las variables globales de la ventana
float gl_ancho = 2.0, gl_alto= 2.0, gl_cerca=-1.0, gl_lejos=5.0;
int w_ancho=500, w_alto=500;


// Espacio para otras variables globales
bool rotando = false;
float sentido = +1; // sentido de rotacion
int pasos_segundo = 50;
int angulo_paso = 1;
GLfloat angulo = 0;
```

Definimos un conjunto de variables para caracterizar nuestra aplicación.

Una booleana para indicar si está rotando o parado.

Un para indicar el sentido de giro:
+1 de derecha a izquierda, sentido anti-horario
-1 de izquierda a derecha, sentido horario

El ángulo actual

Los ángulos por paso y el numero de pasos por segundo.

Main

```
void Timer(int v);
void Raton(int boton, int estado, int x, int y);

//main
int main(int argc, char *argv[]){

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(w_ancho, w_alto);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);

    glutCreateWindow("OpenGL Practica 3_3");
    InitGlew(); // despues de crear la primera ventana
    InitGL();
    glutDisplayFunc(Display); // registra la funcion de rendering
    glutReshapeFunc(ReshapeSize);
    glutTimerFunc(100, Timer, 0); // El primer disparo a los 0.1 segundos
    glutMouseFunc(Raton);

    glutMainLoop(); // bucle principal

    return 0;
}
```

Definimos uso de doble buffer

El primer disparo del Timer ocurre a los 100 milisegundo.

Display

```
// función de gestion de ventana
void Display(){

    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer
    // TO DO

    glMatrixMode(GL_MODELVIEW); // sistema de coordenadas del modelo
    glLoadIdentity();
    glRotatef(angulo, 0.0, 0.0, 1.0); // roracion
    glColor3f(1.0, 1.0, 0.0); // quieres ponerle color?
    glRectf(-0.5, -0.5, 0.5, 0.5); // dibujo del cuadrado 1.0x1.0
    glFlush(); // flush del buffer
    glutSwapBuffers(); // intercambio de buffers
}
```

Uso de coordenadas del modelo y rotación. Aspectos a exponer en el futuro.

Timer y ratón

```
void Timer(int v){

    if (rotando){
        angulo += angulo_paso*sentido;
        if (angulo > 360.0) angulo -= 360.0;
        if (angulo < -360.0) angulo += 360.0;
        glutPostRedisplay();
    }
    glutTimerFunc(1000 / pasos_segundo, Timer, v); // se arma de nuevo
}

void Raton(int boton, int estado, int x, int y){

    if (boton == GLUT_LEFT_BUTTON && estado == GLUT_DOWN) {
        rotando = true;
        sentido = +1;
    }
    if (boton == GLUT_RIGHT_BUTTON && estado == GLUT_DOWN) {
        rotando = true;
        sentido = -1;
    }
    if (boton == GLUT_MIDDLE_BUTTON && estado == GLUT_DOWN) {
        rotando = false;
    }
}
```

Si el estado es rotando, se incrementa/decrementa el ángulo acorde al sentido.

Se ajusta al valores dentro de 360.0

Se lanza para la próxima captura del evento timer

La función del ratón, simplemente actualiza el estado de las variables booleana de rotación y el sentido.

Que debe entregar el alumno?

- Cada alumno entregará en el Campus Virtual una memoria en PDF en la que estará contenida una descripción del trabajo realizado, incluyendo descripción, el listado C/C++ de la actividad realizada y la captura de pantalla de las gráficas o imágenes generadas.
- Para autentificar las imágenes cuando sea posible el alumno incluirá su nombre en cada ventana en el título.
- En principio la tarea quedará abierta para su entrega hasta cierta fecha que se indicará.
- Se puede trabajar en grupo en el Laboratorio, pero la memoria elaborada y entregado será individual.

Bibliografía

Reference Card: https://www.khronos.org/files/opengl-quick-reference-card.pdf

Colección Canónica de OpenGL en: https://www.opengl.org/documentation/books/

Computer Graphics through OpenGL:
http://www.sumantaguha.com/files/materials/Experimenter.pdf