

Práctica 2. Uso de GLUT. Parte 1

Creando Interfaces de Usuario

Grado en Ingeniería Informática. Mención Computación

Escuela de Ingeniería Informática

Universidad de Las Palmas de Gran Canaria



Contenidos

- Practicar procedimientos elementales de GLUT.
- Gestión de Ventanas y eventos de teclado y del ratón.
- Cambio de tamaño de ventanas.

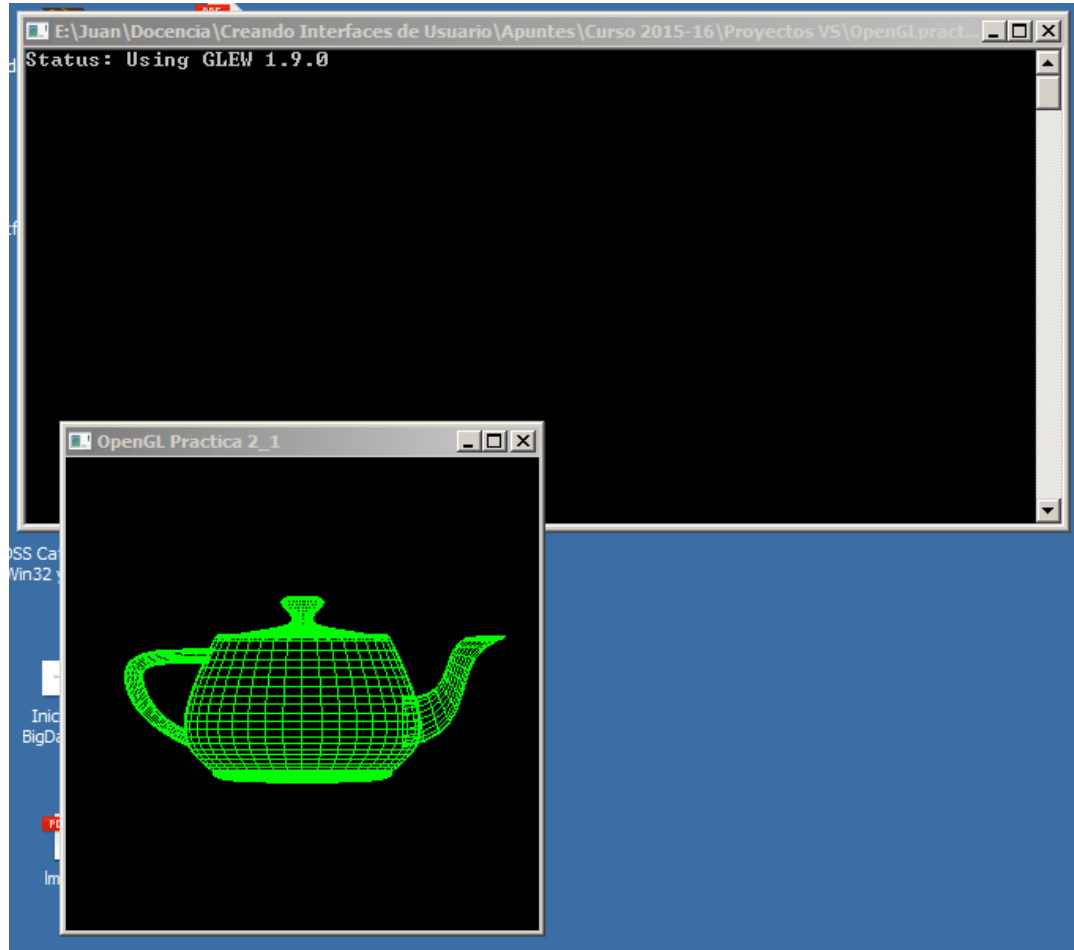


Configurar

1. Configurar el VS con el directorio de includes a **C:/OpenGL/include**
2. Configurar el VS con el directorio de libarías a **C:/OpenGL/lib**
3. Incluir en la Entrada del Vinculador las libarías **freeglut.lib** y **glew32.lib**



Tarea 1. Gestión de teclado



Dibujaremos una figura de modelo de alambre/wireframe y según la tecla que pulsemos se cambiará el color de la figura. Debemos incluir un gestor de teclado.

Un ejemplo de colores:

- 1: rojo
- 2: verde
- 3: azul
- 4: amarillo
- ...
- F1: blanco

Cambiaremos también el cursor



Estructura

```
/*
  Prototipo de main.c
  ULPGC, EII, Creando Interfaces de Usuario
*/
#include <stdio.h>
#include <GL\glew.h>
#include <GL\freeglut.h>

// Variables globales para comunicar funciones
float red=1.0, green=1.0, blue=1.0;

void InitGlew(){
    // para poder utilizar transparentemente todas las extensiones de OpenGL
    GLenum glew_init = glewInit();
    if (glew_init != GLEW_OK)
    {
        // Problem: glewInit failed, something is seriously wrong.
        fprintf(stderr, "Error: %s\n", glewGetErrorString(glew_init));
    }
    else
        fprintf(stdout, "Status: Using GLEW %s\n", glewGetString(GLEW_VERSION));
}
```

- Display()
- Init()
- InitGlew()
- main(int argc, char * argv[])
- Tedado1(unsigned char key, int x, int y)
- Tedado2(int key, int x, int y)
- blue
- green
- red



Inicialización y Display

```
void Init(){
    glClearColor(0.0, 0.0, 0.0, 0.0); // fondo negro
    // TO DO

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.0f, 1.0f, -1.0f, 1.0f, 1.0f, 0.0f); // espacio de trabajo
}

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

    glColor3f(red, green, blue); ← El color definido en las variables globales

    glutWireTeapot(0.5);
    glFlush(); // actualiza el framebuffer
}
```



Gestores de teclado

```
void Teclado1(unsigned char key, int x, int y){
```

```
    switch (key){  
    case '1':  
        red = 1.0;  
        green = 0.0;  
        blue = 0.0;  
        break;  
    case '2':  
        red = 0.0;  
        green = 1.0;  
        blue = 0.0;  
        break;  
    case '3':  
        red = 0.0;  
        green = 0.0;  
        blue = 1.0;  
        break;  
    case '4':  
        red = 1.0;  
        green = 1.0;  
        blue = 0.0;  
        break;  
    }
```

```
    glutPostRedisplay();  
}
```

```
void Teclado2(int key, int x, int y){
```

```
    if (key == GLUT_KEY_F1){  
        red = 1.0;  
        green = 1.0;  
        blue = 1.0;  
    }  
    glutPostRedisplay();  
}
```

No olvidar actualizar la imagen

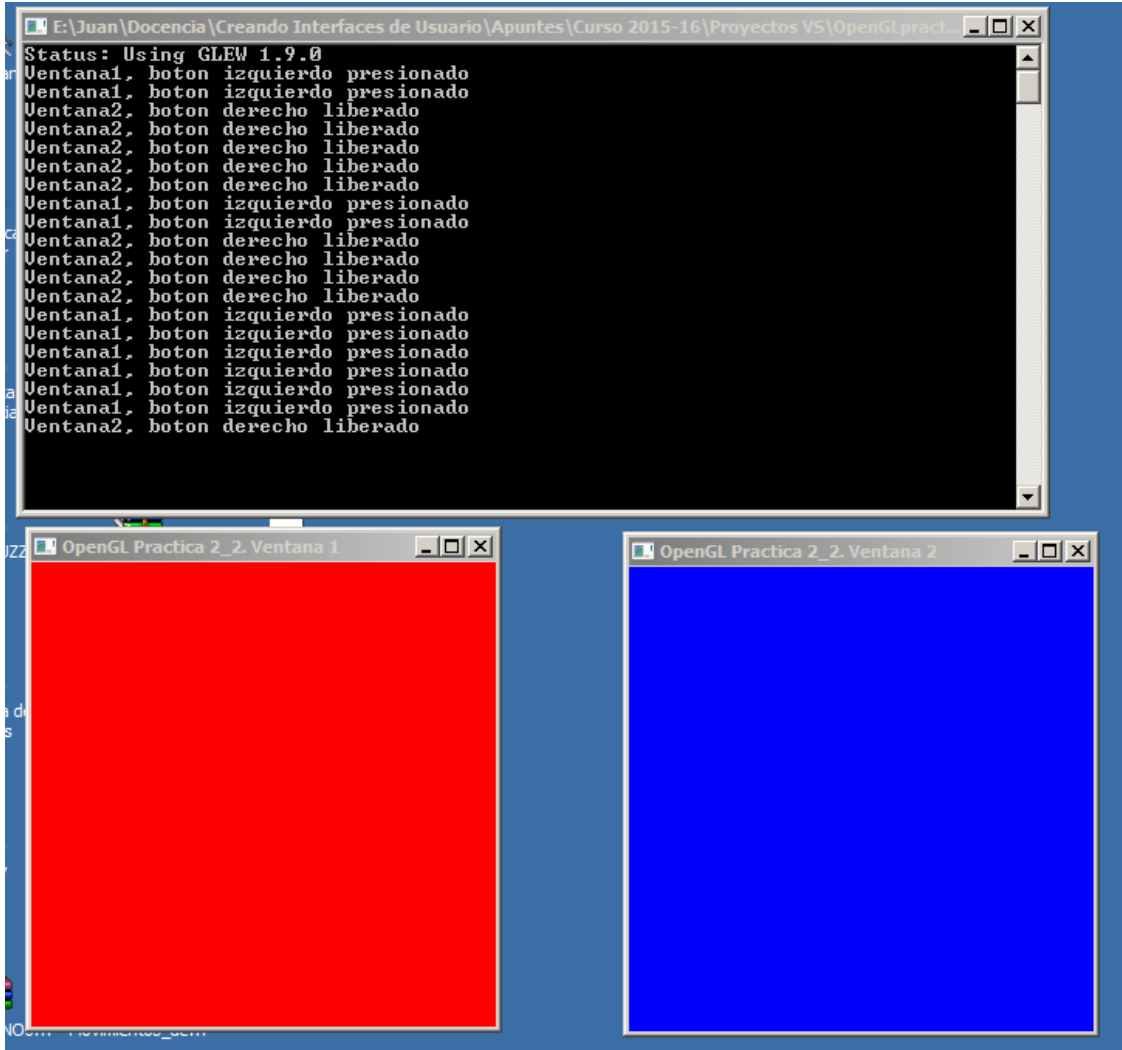


Main()

```
int main(int argc, char *argv[]){  
    glutInit(&argc, argv);  
  
    glutInitWindowPosition(100, 100);  
    glutInitWindowSize(300, 300);  
    glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);  
    glutCreateWindow("OpenGL Practica 2_1");  
    glutSetCursor(GLUT_CURSOR_WAIT); // cursor especial de la ventana  
    InitGlew();  
    Init();  
    glutDisplayFunc(Display); // define la función de rendering  
    glutKeyboardFunc(Teclado1);  
    glutSpecialFunc(Teclado2);  
  
    glutMainLoop(); // bucle principal  
  
    return 0;  
}
```



Tarea 2. Ventanas y Ratón



Crearemos dos ventanas inicializadas de forma diferente. En este caso solo con colores de fondo diferente.

Asociaremos a cada ventana un gestor de ratón diferente.

Una capturarán un presión del botón izquierdo y la otra la liberación del botón derecho.

Ambas imprimirán en la consola la ocurrencia del evento.



Estructura General

```
/*
  Prototipo de main.c
  UPGC, EII, Creando Interfaces de Usuario
*/
#include <stdio.h>


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

// Variables Globales
int win1, win2;

void InitGlew(){

    // para poder utilizar transparentemente todas las extensiones de OpenGL

    GLenum glew_init = glewInit();
    if (glew_init != GLEW_OK)
    {
        // Problem: glewInit failed, something is seriously wrong.
        fprintf(stderr, "Error: %s\n", glewGetErrorString(glew_init));
    }
    else
        fprintf(stdout, "Status: Using GLEW %s\n", glewGetString(GLEW_VERSION));
}
```



- Display10
- Display20
- Init10
- Init20
- InitGlew0
- main(int argc, char * argv[])
- MouseVentana1(int button, int state, int x, int y)
- MouseVentana2(int button, int state, int x, int y)
- win1
- win2

Explorador de soluciones Team Explorer Vista de clases

Inicialización de Ventanas

Color Rojo

```
void Init1(){  
    glClearColor(1.0, 0.0, 0.0, 0.0);  
    // TO DO  
  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    glOrtho(-1.0f, 1.0f, -1.0f, 1.0f, 1.0f, 0.0f); // espacio de trabajo  
}
```

Color Azul

```
void Init2(){  
    glClearColor(0.0, 0.0, 1.0, 0.0);  
    // TO DO  
  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    glOrtho(-1.0f, 1.0f, -1.0f, 1.0f, 1.0f, 0.0f); // espacio de trabajo  
}
```

```
void Display1(){  
    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer  
    // TO DO  
  
    glFlush(); // actualiza el framebuffer  
}  
  
void Display2(){  
    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer  
    // TO DO  
  
    glFlush(); // actualiza el framebuffer  
}
```



Gestores de ratón

```
void MouseVentana1(int button, int state, int x, int y){  
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)  
        printf("Ventana1, boton izquierdo presionado\n");  
}  
  
void MouseVentana2(int button, int state, int x, int y){  
    if (button == GLUT_RIGHT_BUTTON && state == GLUT_UP)  
        printf("Ventana2, boton derecho liberado\n");  
}
```

El gestor de ratón recibe cuatro argumentos. Los dos primeros sirven para identificar el botón presionado/liberado y el estado en que se encuentra. Debemos filtrar la combinación que deseamos capturar.



Main()

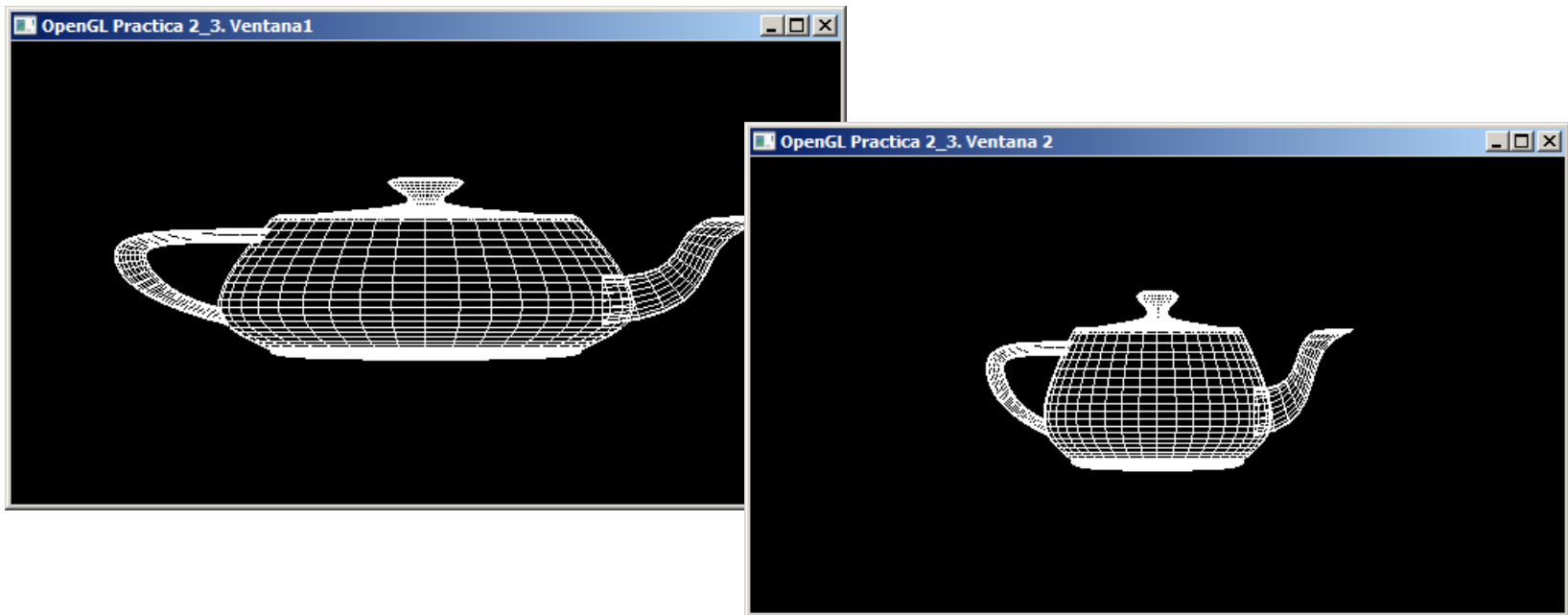
```
int main(int argc, char *argv){  
    glutInit(&argc, argv);  
    glutInitWindowPosition(100, 100);  
    glutInitWindowSize(300, 300);  
    glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);  
  
    win1=glutCreateWindow("OpenGL Practica 2_2. Ventana 1");  
    InitGlew(); // despues de crear la primera ventana  
    Init1();  
    glutDisplayFunc(Display1); // registra la funcion de rendering  
    glutMouseFunc(MouseVentana1);  
  
    win2 = glutCreateWindow("OpenGL Practica 2_2. Ventana 2");  
    Init2();  
    glutDisplayFunc(Display2); // registra la funcion de rendering  
    glutMouseFunc(MouseVentana2);  
  
    glutMainLoop(); // bucle principal  
  
    return 0;  
}
```



Tarea 3. Reshape de Ventanas .

Cuando se modifica el tamaño de la ventana, se modifica la relación de aspecto de la imagen (relación de dimensiones entre los ejes x e y)

Registraremos una función de Reshape para redefinir las dimensiones del dibujo al nivel de funciones OpenGL para que se preserve la relación de aspecto.



La ventana de la izquierda no gestiona el cambio de dimensiones, la de la derecha si



Inicializar

```
#include <stdio.h>

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

// Variables Globales
int win1, win2;

void InitGlew(){

    // para poder utilizar transparentemente todas las extensiones de OpenGL

    GLenum glew_init = glewInit();
    if (glew_init != GLEW_OK)
    {
        // Problem: glewInit failed, something is seriously wrong.
        fprintf(stderr, "Error: %s\n", glewGetErrorString(glew_init));
    }
    else
        fprintf(stdout, "Status: Using GLEW %s\n", glewGetString(GLEW_VERSION));
}

void Init(){
    glClearColor(0.0, 0.0, 0.0, 0.0);
    // TO DO

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.0f, 1.0f, -1.0f, 1.0f, 1.0f, 0.0f); // espacio de trabajo
}
```

- ⊗ Dimensiones(int ancho, int alto)
- ⊗ Display1()
- ⊗ Display2()
- ⊗ Init()
- ⊗ InitGlew()
- ⊗ main(int argc, char *argv[])
- ⊗ win1
- ⊗ win2



Algoritmo

La ventana inicialmente tiene unas dimensiones físicas de 300x300 que proyecta unas dimensiones lógicas de 2x2. Luego la relación de aspecto entre los ejes x e y son correctos.

Cuando cambian las dimensiones tendrá unas dimensiones físicas de ancho x alto, que para mantener la correcta proporción deberían implicar unas dimensiones lógicas de dX dY .

Si ancho > alto: $dY = 2$ y $dX = 2 * \text{ancho} / \text{alto}$

Si alto > ancho: $dX = 2$ y $dY = 2 * \text{alto} / \text{ancho}$

El espacio ortogonal se debe situar centrado, desde $-dx/2$ hasta $dx/2$ e igual en el eje Y.

Con este algoritmo, la figura se mantiene en relación de aspecto correcto, pero el tamaño puede cambiar.




```

void Display1(){
    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer
    // TO DO

    glutWireTeapot(0.5);
    glFlush(); // actualiza el framebuffer
}

void Display2(){
    glClear(GL_COLOR_BUFFER_BIT); // borra todo lo existente en el framebuffer
    // TO DO

    glutWireTeapot(0.5);
    glFlush(); // actualiza el framebuffer
}

void Dimensiones(int ancho, int alto){
    float dx = 2.0;
    float dy = 2.0;

    if (ancho > alto){
        dx = 2.0*(float)ancho / (float)alto;
    }
    if (alto > ancho){
        dy = 2.0*(float)alto / (float)ancho;
    }

    glViewport(0, 0, ancho, alto);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-dx/2.0, dx/2.0, -dy/2.0, dy/2.0, 1.0f, 0.0f); // espacio de trabajo
    glutPostRedisplay();
}

```

Pintar la tetera en ambos casos

Función de gestión del cambio de dimensiones



Main()

```
int main(int argc, char *argv[]){  
  
    glutInit(&argc, argv);  
    glutInitWindowPosition(100, 100);  
    glutInitWindowSize(300, 300);  
    glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);  
  
    win1 = glutCreateWindow("OpenGL Practica 2_3. Ventana1");  
    InitGlew(); // despues de crear la primera ventana  
    Init();  
    glutDisplayFunc(Display1); // registra la funcion de rendering  
  
    win2 = glutCreateWindow("OpenGL Practica 2_3. Ventana 2");  
    Init();  
    glutDisplayFunc(Display2); // registra la funcion de rendering  
    glutReshapeFunc(Dimensiones);  
  
    glutMainLoop(); // bucle principal  
  
    return 0;  
}
```

Registramos la función de reshape a la ventana 2, pero no a la 1



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 autenticar 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>

