

# Ejercicios de exámenes de IG

## 1 Septiembre 2012-2013

**Ejercicio 3.** Describe brevemente para que sirven, en un programa OpenGL/glut, cada una de estas cuatro funciones:

**a) glutDisplayFunc(function);**

Esta función se llamará cada vez que se dibuje la ventana.

**b) glutReshapeFunc(function);**

Función de control del cambio de tamaño de la ventana de visualización.

**c) glutKeyboardFunc(function);**

Función de control de eventos con el teclado.

**d) glutSpecialFunc(function);**

Función de control de eventos con el teclado para cuando se ha pulsado una tecla especial.

## 2 Septiembre 2014

**Ejercicio 1.** Considera una malla de  $n$  triángulos almacenada en memoria con un vector *caras* (con  $n$  entradas), de forma que *caras[i][j]* es un entero, en concreto el índice del vértice número  $j$  de la cara número  $i$  (con  $0 \leq i \leq n$  y  $0 \leq j < 3$ ).

1. Con esta definición, escribe el código de una función con esta declaración:

**bool comparten\_vertice(int c1, int c2);**

**que devuelve true cuando las caras número c1 y c2 comparten un vértice (devuelve false si esto no es así).**

```
bool comparten_vertice(int c1, int c2) {
    const int NUMERO_VERTICES = 3;
    bool comparten_vertice = false;

    for (int j = 0; j < NUMERO_VERTICES && !comparten_vertice; ++j) {
        for (int k = 0; k < NUMERO_VERTICES && !comparten_vertice; ++k) {
            if (caras[c1][j] == caras[c2][k])
                comparten_vertice = true;
        }
    }

    return comparten_vertice;
}
```

**2. Escribe el código de otra función:**

**bool comparten\_aristas(int c1, int c2);**

**que devuelve true cuando las caras número c1 y c2 comparten una arista (devuelve false si esto no es así).**

```
bool comparten_arista(int c1, int c2) {
    const int NUMERO_VERTICES = 3;
    bool comparten_arista = false;
    bool comparten_vertice = comparten_vertice(c1, c2);
    int vertice_a;
    int vertice_b;

    vertice_a = -1;
    vertice_b = -1;

    if (comparten_vertice) {
        for (int j = 0; j < NUMERO_VERTICES && !comparten_arista; ++j) {
            for (int k = 0; k < NUMERO_VERTICES && !comparten_arista; ++k) {
                if (caras[c1][j] == caras[c2][k] && vertice_a == vertice_b)
                    vertice_a = caras[c1][j];
                else if (caras[c1][j] == caras[c2][k] && vertice_a != vertice_b)
                    comparten_arista = true;
            }
        }
    }

    return comparten_arista;
}
```