practica0.pdf

postdata9

Informática Gráfica

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

WUOLAH

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

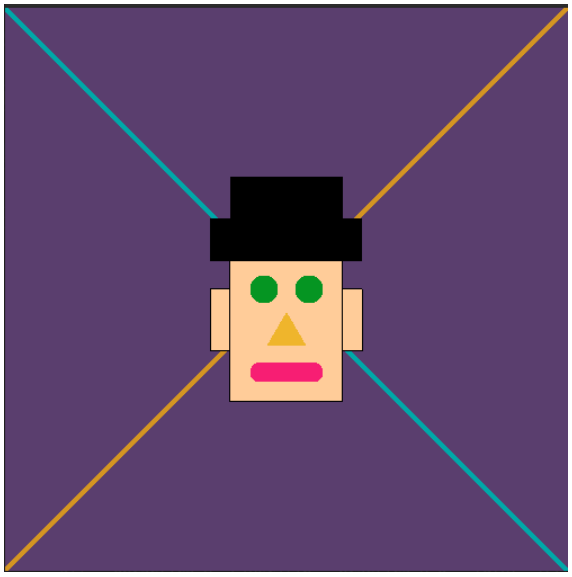
Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

Práctica 0:

Introducción a OpenGL



Informática Gráfica

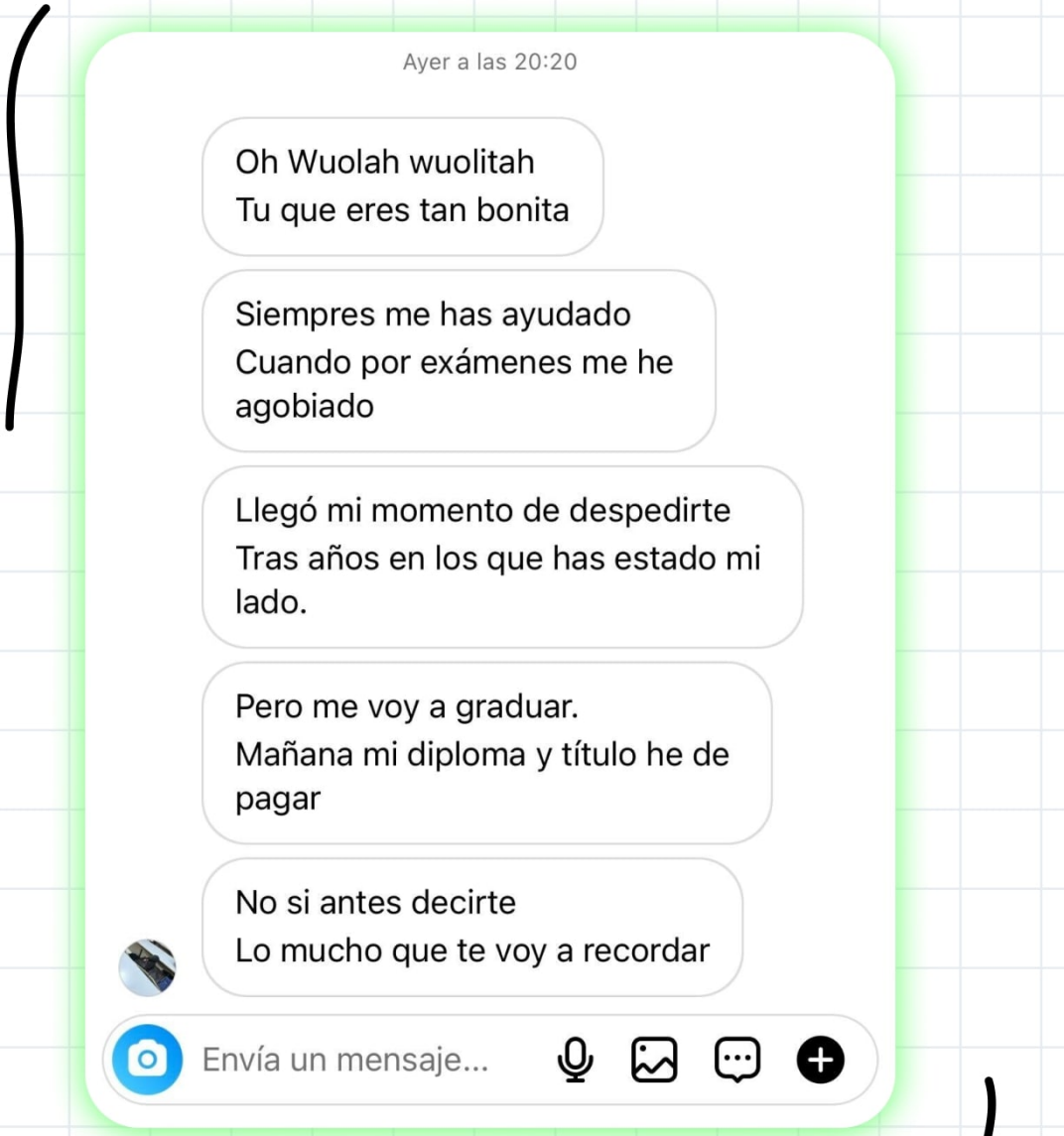
2020 – 2021

Índice:

1. Preámbulo
2. Sintaxis de las funciones
3. Primitivas gráficas
4. Monigote

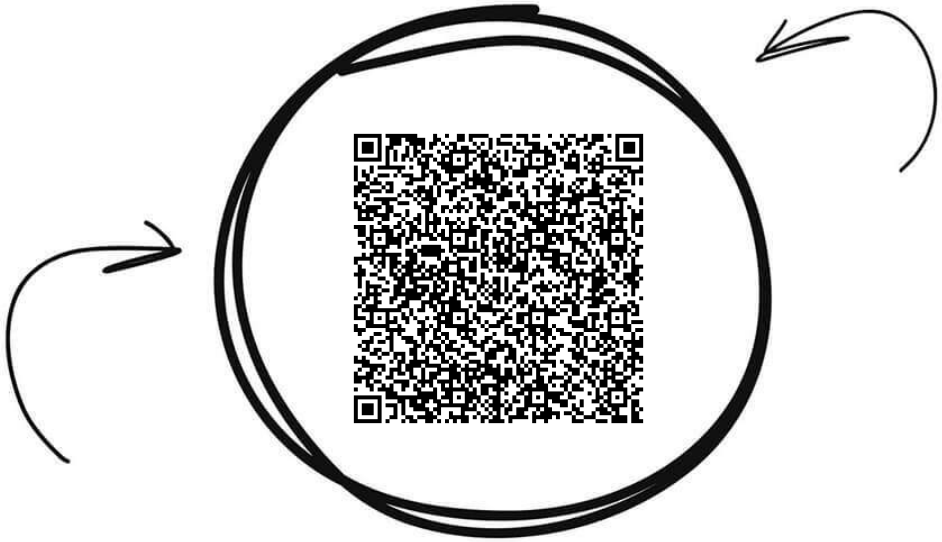
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Informática Gráfica



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

1 Preámbulo

OpenGL es una **biblioteca gráfica** que permite realizar aplicaciones gráficas interactivas tanto 2D como 3D.

- Las **funciones** de OpenGL ya están implementadas en los procesadores gráficos.
- Se puede utilizar con **distintos lenguajes**, nosotros usaremos C++.
- Se puede usar en **diversas plataforma** como Windows, Apple, Linux.
- Es **portable**, ya que no contiene funciones para entornos de ventanas y se puede cambiar de plataforma.
- Tiene una arquitectura cliente/servidor.
- Funciona como una máquina de estados.
- OpenGL permite, como mucho, hacer caras planas que es algo muy básico, por lo que usaremos **GLUT** para entorno de ventanas, en la que podremos utilizar funciones ya predefinidas como esferas, cilindros.
- Con **WebGL** podemos utilizar OpenGl dentro de un navegador web.
- La última es la 4.5, ya que paró en 2014 debido a que salió una nueva librería gráfica: **Vulkan**, aunque hoy día hay pocas tarjetas gráficas que lleven implementado Vulkan por lo que hay que seguir utilizando OpenGL.

2 Sintaxis de las funciones

- Las **funciones** en OpenGL siempre empiezan con el prefijo **gl**, seguido del nombre de la función que puede ser de una palabra o más. La primera de letra de cada palabra va en mayúsculas.

  Ej.: glLineStipple

  - Los **sufijos con dos partes** en los nombres de la función significan:
    - el primero indica el número de argumentos,
    - la segunda el tipo de dato.

      Ej.: glColor3f(f1, f2, f3) → 3 argumentos de tipo float
  - En algunas funciones, puede aparecer una **v** al final que indica un **puntero** a un vector.

    Ej.: glColor3fv(array_color)
- Las **constantes** van precedidas por el prefijo **gl** y en **mayúsculas** con palabras separadas por _.

  Ej.: GL_POLYGON
- Los **tipos de datos** son los siguientes, que se corresponden de la siguiente forma:

Tipo de dato		Tipo en lenguaje C	Tipo definido en OpenGL
b	entero 8 b.	signed char	GLbyte
s	entero 16 b.	short	GLshort
i	entero 32 b.	long	GLint, Glsizei
f	real 32 b.	Float	GLfloat, Glclampf
d	real 64 b.	double	GLdouble, Glclampd
ub	ent. sin signo 8 b.	unsigned char	GLubyte, Glboolean
us	ent. sin signo 16 b.	unsigned short	GLushort
ui	ent. sin signo 32 b.	unsigned long	GLuint,GLenum, GLbitfield

3 Primitivas gráficas

Son aquellas **funciones** que exclusivamente se utilizan para **crear objetos gráficos**. OpenGL es un sistema gráfico a muy bajo nivel, cuyo objeto gráfico más sofisticado es una cara plana.

Para hacer una figura, se debe encerrar en una estructura, es la geometría. Esta **estructura**:

- empieza con **glBegin(modo)**, este modo indica la forma de la figura;
- termina con **glEnd()**,
- entre estas dos funciones, utilizaremos **glVertexYZ()**, que será el conjunto de vértices de la figura, donde Y será la dimensión del vértice y Z el tipo de dato.

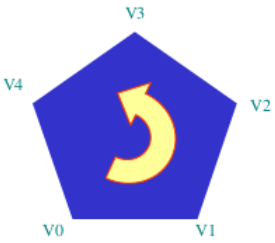
En el siguiente ejemplo, se dibuja una figura por puntos en el que el de la izquierda es de dos dimensiones (glVertex2f) y el de la derecha de 3 dimensiones (glVertex3f):

```
glBegin (GL_POINTS);
    glVertex2f (0.0, 0.0);
    glVertex2f (1.0, 0.0);
    glVertex2f (2.0, 3.0);
    ...
glEnd ( );
```

```
glBegin (GL_POINTS);
    glVertex3f (0.0, 0.0,1.0);
    glVertex3f (1.0, 0.0,0.0);
    glVertex3f (2.0, 3.0,0.7);
    ...
glEnd ( );
```

Es importante el **orden** en el que se tienen que expresar **los vértices**, tiene que ser en el **sentido antihorario**. Esto ayuda a diferenciar la parte frontal de una cara plana de la parte de atrás.

```
glBegin (GL_POLYGON);
    glVertex3f (V0_x, V0_y,0.0);
    glVertex3f (V1_x, V1_y,0.0);
    glVertex3f (V2_x, V2_y,0.0);
    glVertex3f (V3_x, V3_y,0.0);
    glVertex3f (V4_x, V4_y,0.0);
glEnd();
```

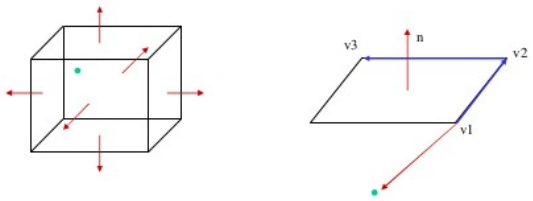


No importa el vértice por el que se empiece, siempre que los siguientes vértices, vengan dado en sentido antihorario. Una vez dado los vértices y el sentido antihorario, OpenGL calcula el vector normal a una cara dada (sólo le hace falta 3 puntos).

Los **usos** que tiene calcular la normal a una cara dada:

- se puede saber la **posición de un punto** con respecto a una cara, si está **dentro** o **fuera** de una cara. Si hacemos el producto escalar de un vector y su normal:
  - si sale **negativo** el punto se encuentra **encima** del plano (**cara frontal**);
  - si sale **positivo** el punto se encuentra por **debajo** del plano (**cara trasera**);

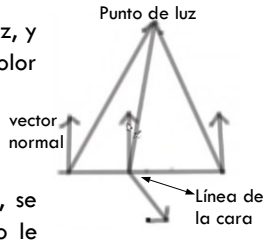
  Por tanto, si un punto se encuentra en la parte interior (detrás) de todas las caras, el punto se encuentra dentro de la figura sólida.



- para ver si una figura **choca con otra,** se puede realizar un corte;

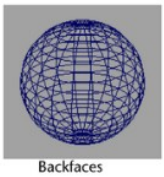
- para la **iluminación** de un polígono, el color que deben tener los objetos:

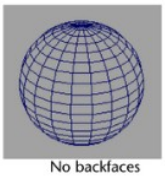
  al tener un **punto de luz**, trazamos un vector desde un punto de la cara a la luz, y el ángulo que forma el vector normal y ese vector hacia la luz, indica el color que hay que pintar en ese punto.



  - cuanto mayor sea el ángulo, más oscuro es el color (más sombra tiene);
  - cuando el ángulo supere los 90ª (el producto escalar saldrá negativo), se pondrá negro, ya que el punto se encontraría en la cara trasera y no le daría la luz.

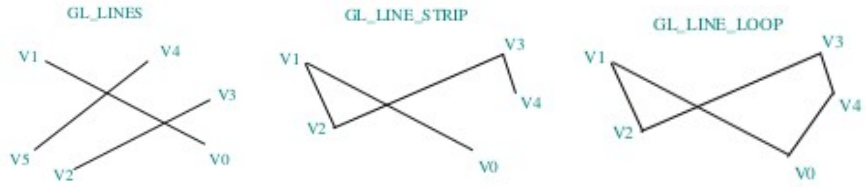
- para optimización mediante eliminación de la cara trasera en la visualización (**BackFace Culling**). Es decir, cuando tenemos una figura como una esfera, las caras que están frente al observador tapan a las traseras, por lo que podemos eliminar dichas caras a la hora de dibujar el objeto. De esta forma será más rápido dibujar el objeto ya que se tardará la mitad de tiempo.


Backfaces


No backfaces

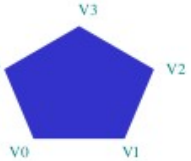
Argumentos de glBegin

- **GL_POINTS**: puntos individuales;
- **GL_LINES**: segmentos de líneas individuales sin conexión. Si queremos dibujar 1 línea, se necesitarán 2 vértices; si queremos dibujar 3 líneas serán 6 vértices;
- **GL_LINE_STRIP**: dibuja una polilínea;
- **GL_LINE_LOOP**: dibuja una polilínea cerrada, hace una conexión entre el primer y último vértice;

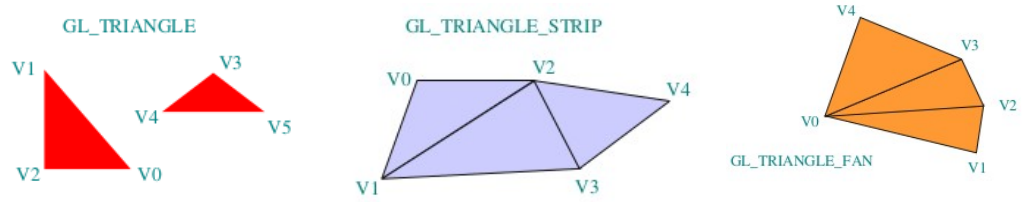


- **GL_POLYGON**: polígono convexo, una cara plana. Las caras planas pueden ser un polígono cóncavo (como una estrella) o convexo (un pentágono).
  - En un **polígono cóncavo**, el ángulo que se forma en un punto interior es **mayor** que **180ª**, todos los segmentos que se hagan entre los vértices pueden quedar fuera de la figura. Son **difíciles** de implementar en OpenGL, no puede trabajar con este tipo de polígonos. La estrella se haría: con 5 triángulos para los lados y un pentágono para el interior.

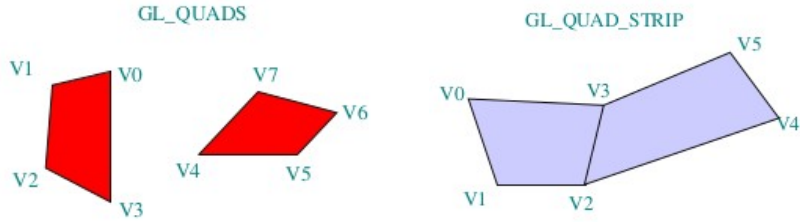
  - En un **polígono convexo**, el ángulo que se forma en un punto interior es **menor** que **180ª**, todos los segmentos que se hagan entre los vértices quedarán dentro de la figura.



- **GL_TRIANGLES:** dibuja triángulos no conectados;
- **GL_TRIANGLE_STRIP:** dibuja triángulo encadenados en la misma orientación de modo que formen parte de una superficie;
- **GL_TRIANGLE_FAN:** enlaza triángulos en abanico (sirve para hacer conos o bases de cilindros);



- **GL_QUADS**: para cara plana de 4 puntos separados;
- **GL_QUAD_STRIP:** para enlazar objetos de 4 puntos;

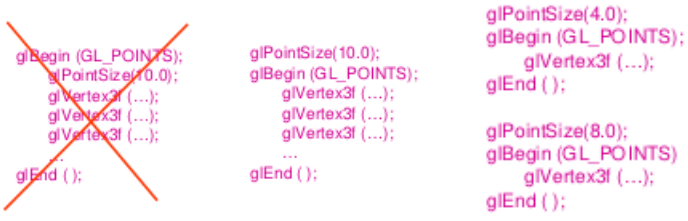


Atributos

Los atributos de las figuras son: **forma, tamaño** y **color**.

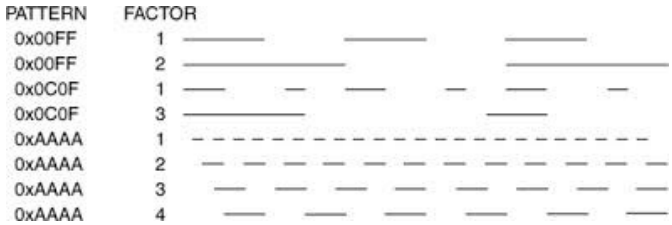
Atributo de punto

- **Forma:** Para un punto no hay forma, tiene una forma cuadrada predeterminada y no se puede modificar.
- **Tamaño:** se indica en píxeles con la función **glPointSize(GLfloat size)** (esta función debe ir encima del glBegin(), foto de la izquierda). Para darle un tamaño distinto a varios puntos, hay que hacerlo como se indica en la segunda foto:



Atributo de línea

- **Forma:** hay distintas formas para una línea. La función que se usa es **glLineStipple(GLint factor, Glushort pattern)**. El factor es el espaciado entre líneas, mientras que el pattern es el patrón de la línea. Abajo hay una tabla con las diferentes formas de una línea:



  - Para que la línea se pinte con un patrón de arriba, hay que habilitarlo antes del glBegin() y desactivarlo después de glEnd(). Para ello están las funciones **glEnable(GL_LINE_STIPPLE)** y **glDisable(GL_LINE_STIPPLE),** después del glEnable se escribe el patrón a seguir.
- **Tamaño:** especifica el ancho en píxeles de una línea con la función **glLineWidth(GLfloat width)**.
- Ejemplo:
  - Se establece el color
  - Se activa la forma de la línea seguido del patrón que queremos, en este caso discontinuo
  - La estructura de línea con dos vértices
  - Se desactiva el patrón de línea. Al actuar como una máquina de estados, si no se desactiva el patrón, va a permanecer así siempre.

```
glColor3f(1.0, 1.0, 1.0);

glEnable (GL_LINE_STIPPLE);
glLineStipple (1, 0x00FF);
  glBegin( GL_LINES );
        glVertex2f( -1.0, 0.0,0.0 );
        glVertex2f( 0.5, 0.0,0.0 );
  glEnd();
glDisable (GL_LINE_STIPPLE );
```

Atributo de polígono

- **Forma:** para establecer el estilo de un polígono se usa **glPolygonMode(GLenum face, Glenum mode)**, donde:
  - face: puede ser **GL_BACK, GL_FRONT_AND_BACK** o **GL_FRONT**, que indica si se aplica a la cara trasera o delantera;
  - mode: **GL_POINT, GL_LINE** y **GL_FILL**, con puntos, con líneas o un sólido.

  Se puede indicar si las aristas se dibujan o no estableciendo el argumento de la función **glEdgeFlag(GLboolean flag)** a **GL_TRUE** (por defecto) o **GL_FALSE** (no se dibujan).

  Aquí también hay que activar y desactivar el patrón con **glEnable(GL_POLYGON_STIPPLE)** y **glDisable(GL_POLYGON_STIPPLE)**.

Atributo color

- **glColor3x(TYPE red, TYPE green, TYPE blue)**: para establecer el color de cualquier polígono. x puede ser {b s i f d ub us ui}. Ejemplo: glColor3f(1.0, 0.0, 0.0) color rojo puro.
- **glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)**: para el color de fondo.
- **glClear(GLbitfield {GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT})**: para limpiar el color de fondo. Se suele usar junto a la de arriba.

4 Monigote

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/freeglut.h>

//pinta un círculo, con esta función dibujamos los ojos y las comisuras de los labios
void Circulo (GLfloat radio, GLfloat cx, GLfloat cy, GLint n, GLenum modo){
	int i;
	if (modo == GL_LINE) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
	else if (modo == GL_FILL) glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
	else glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);

	glBegin( GL_POLYGON );
		for (i = 0; i<n; i++)
			glVertex2f( cx+radio*cos(2.0*M_PI*i/n), cy+radio*sin(2.0*M_PI*i/n));
	glEnd();
}

//para pintar los ejes en forma de cruz
void Ejes (int width){

	//cambia la anchura de la línea a la que le pasemos
	glLineWidth(width);
	glBegin(GL_LINES);   //dibujamos líneas

		//eje 1, va de la esquina superior izquierda a la esquina inferior derecha
		glColor3ub(0, 168, 168);       //color azul
		glVertex3f(-1.0, 1.0, 0.0);
		glVertex3f(1.0, -1.0, 0.0);

		//eje 2, va de la esquina inferior izquierda a la esquina superior derecha
		glColor3ub(212, 149, 34);      //color marrón
		glVertex3f(-1.0, -1.0, 0.0);
		glVertex3f(1.0, 1.0, 0.0);
	glEnd();
}

//dibuja la cara
void Monigote (){

	// cara
	glLineWidth(1);
	glColor3f(1.0, 0.8, 0.6);              //color carne
	//un polígono que rellenamos entero tanto por delante como por detrás
	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

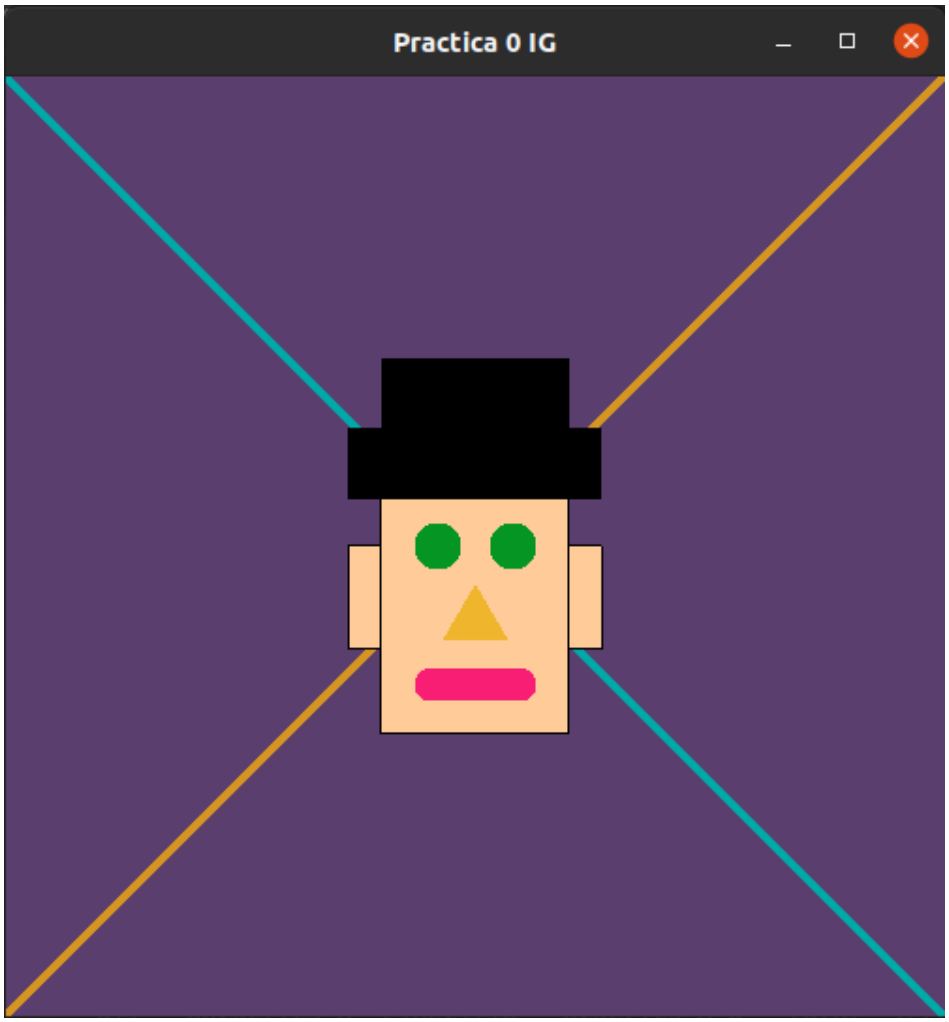
Oh Wuolah wuolitah
Tu que eres tan bonita

```
//cara del monigote, un rectángulo
glBegin(GL_POLYGON);
        glVertex3f(-0.2, -0.4, 0.0);     //vértice inferior izquierdo
        glVertex3f(0.2, -0.4, 0.0);      //vértice inferior derecho
        glVertex3f(0.2, 0.15, 0.0);      //vértice superior derecho
        glVertex3f(-0.2, 0.15, 0.0);     //vértice superior izquierdo
glEnd();

//las orejas
glColor3f(1.0, 0.8, 0.6);       //color carne
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(GL_POLYGON);
        glVertex3f(-0.27, 0.0, 0.0);     //vértice superior izquierdo
        glVertex3f(-0.27, -0.22, 0.0);   //vértice inferior izquierdo
        glVertex3f(0.27, -0.22, 0.0);    //vértice inferior derecho
        glVertex3f(0.27, 0.0, 0.0);      //vértice superior derecho
glEnd();

//los ojos
glColor3ub(5, 149, 34);
Circulo(0.05, -0.08, 0.0, 20, GL_FILL);         //ojo izquierdo
Circulo(0.05, 0.08, 0.0, 20, GL_FILL);          //ojo derecho

//la nariz
glColor3ub(239, 181, 44);
glBegin(GL_TRIANGLES);                  //forma de triángulo
        glVertex3f(0.07, -0.2, 0.0);     //vértice inferior derecho
        glVertex3f(0.0, -0.08, 0.0);     //vértice superior
        glVertex3f(-0.07, -0.2, 0.0);    //vértice inferior izquierdo
glEnd();

//las comisuras de la boca
glColor3ub(247, 30, 115);
Circulo(0.035, -0.095, -0.295, 20, GL_FILL); //comisura izquierda
Circulo(0.035, 0.095, -0.295, 20, GL_FILL);  //comisura derecha

//la boca
glColor3ub(247, 30, 115);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(GL_POLYGON);
        glVertex3f(0.095, -0.33, 0.0);          //vértice inferior derecho
        glVertex3f(0.095, -0.26, 0.0);          //vértice superior derecho
        glVertex3f(-0.095, -0.26, 0.0);         //vértice superior izquierdo
        glVertex3f(-0.095, -0.33, 0.0);         //vértice inferior izquierdo
glEnd();

//la copa del sombrero
glColor3ub(0, 0, 0);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(GL_POLYGON);
```

```
            glVertex3f(0.2, 0.4, 0.0);      //vértice superior derecho
            glVertex3f(-0.2, 0.4, 0.0);     //vértice superior izquierdo
            glVertex3f(-0.2, 0.25, 0.0);    //vértice inferior izquierdo
            glVertex3f(0.2, 0.25, 0.0);     //vértice inferior derecho
        glEnd();

        //el ala del sombrero
        glColor3ub(0, 0, 0);
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        glBegin(GL_POLYGON);
            glVertex3f(0.27, 0.25, 0.0);    //vértice superior derecho
            glVertex3f(-0.27, 0.25, 0.0);   //vértice superior izquierdo
            glVertex3f(-0.27, 0.1, 0.0);    //vértice inferior izquierdo
            glVertex3f(0.27, 0.1, 0.0);     //vértice inferior derecho
        glEnd();

        //borde de la cara
        glColor3ub(0, 0, 0);
        glBegin(GL_LINE_LOOP);
            glVertex3f(0.2, 0.15, 0.0);
            glVertex3f(-0.2, 0.15, 0.0);
            glVertex3f(-0.2, -0.4, 0.0);
            glVertex3f(0.2, -0.4, 0.0);
        glEnd();

        //borde de las orejas
        glColor3ub(0, 0, 0);
        glBegin(GL_LINE_STRIP);
            glVertex3f(-0.2, 0.0, 0);
            glVertex3f(-0.27, 0.0, 0);
            glVertex3f(-0.27, -0.22, 0);
            glVertex3f(-0.2, -0.22, 0);
        glEnd();

        //borde de las orejas
        glColor3ub(0, 0, 0);
        glBegin(GL_LINE_STRIP);
            glVertex3f(0.2, -0.22, 0);
            glVertex3f(0.27, -0.22, 0);
            glVertex3f(0.27, 0.0, 0);
            glVertex3f(0.2, 0.0, 0);
        glEnd();
}


//función de OpenGL
static void Init( ){

        //una cara se va a pintar en modo planp (GL_FLAT)
        glShadeModel( GL_FLAT);
```

```
        /*si ponemos GL_SMOOTH tiene en cuenta todos los distintos colores de la figura, no solo uno
        de ellos; aparece degradado de un color a otro*/
}

//para ajustar la ventana, se le pasan el tamaño actual de la ventana
static void Reshape( int width, int height ){
   glViewport(0, 0, (GLint)width, (GLint)height);
   glOrtho (-1.0, 1.0,-1.0, 1.0, -10, 10.0);
}

//funcion para pintar en la ventana
static void Display( ){

        //para pintar el fondo de morado
        glClearColor(90.0f/255.0f, 62.0f/255.0f, 110.0f/255.0f, 0);
        glClear(GL_COLOR_BUFFER_BIT);

        Ejes(6);            //pinta los ejes
        Monigote();         //pinta el monigote

        glFlush();
}

static void Keyboard(unsigned char key, int x, int y ){
        if (key==27)
        exit(0);
}

int main( int argc, char **argv ){

        //para crear la ventana
        //con RGB indicamos que trabajamos con un color real
        glutInit(&argc,argv);
        glutInitDisplayMode( GLUT_RGB );

        //punto inicial de la ventana, tamaño y título
        //el punto (0,0) está arriba a la izquierda
        glutInitWindowPosition( 20, 100 );
        glutInitWindowSize(500, 500 );
        glutCreateWindow("Practica 0 IG");

        Init();  //función de OpenGL

        /*funciones para barajar eventos
                1. Reshape: para el tamaño, para ajustar lo que se está dibujando
                al tamaño de la ventana si la agrando o la hago más pequeña
                2. Display: para decir a OpenGL lo que va a pintar
        */
        glutReshapeFunc(Reshape);
        glutDisplayFunc(Display);
```

```
        glutKeyboardFunc(Keyboard);

        glutMainLoop( );

        return 0;
}
```



Página para los colores: https://htmlcolorcodes.com/es/