UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

CARRERA PROFESIONAL DE INGENIERIA INFORMATICA Y DE SISTEMAS

COMPUTACION GRAFICA 2

DOCENTE: M.SC. HECTOR E. UGARTE R.

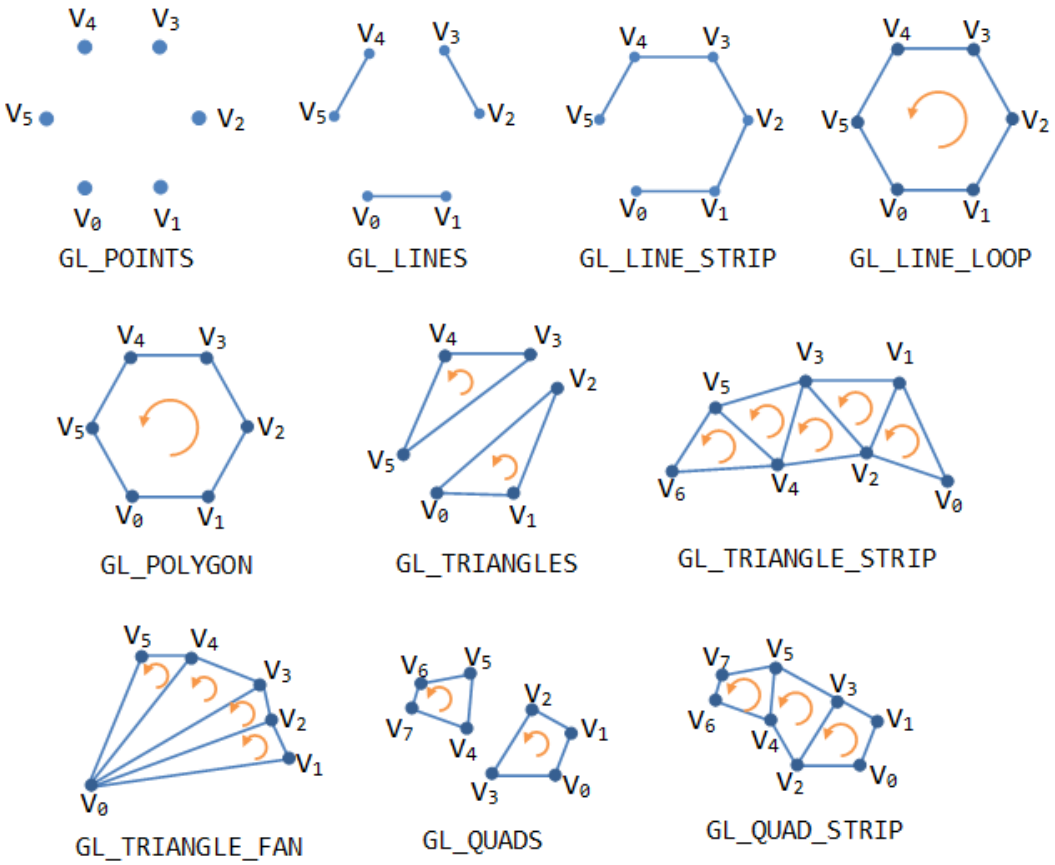
HOJA DE EJERCICIOS 1: INTRODUCCION

1. COMPETENCIAS

- El estudiante conoce y utiliza las primitivas en OpenGL.
- El estudiante es capaz de utilizar el modelo de colores RGB.

2. MARCO TEORICO

A. Color y primitivas



La computadora moderna tiene una unidad de procesamiento de gráficos (GPU) dedicada a producir imágenes para la pantalla, con su propia memoria de gráficos (o RAM de video o VRAM).

Todas las pantallas modernas están basadas en tramas. Un ráster (trama) es una cuadrícula rectangular 2D de píxeles (o elementos de imagen). Un píxel tiene dos propiedades: un color y una posición. El color se expresa en componentes RGB (Rojo-Verde-Azul), normalmente 8 bits por componente o 24 bits por píxel (o color verdadero).

Los valores de color de los píxeles se almacenan en una parte especial de la memoria gráfica llamada búfer de fotogramas. La GPU escribe el valor del color en el búfer de fotogramas. La

pantalla lee los valores de color del búfer de fotogramas fila por fila, de izquierda a derecha, de arriba a abajo, y pone cada uno de los valores en la pantalla. Esto se conoce como escaneo de trama. La pantalla se actualiza varias docenas de veces por segundo, generalmente a 60 Hz para monitores LCD y más para tubos CRT. Esto se conoce como frecuencia de actualización.

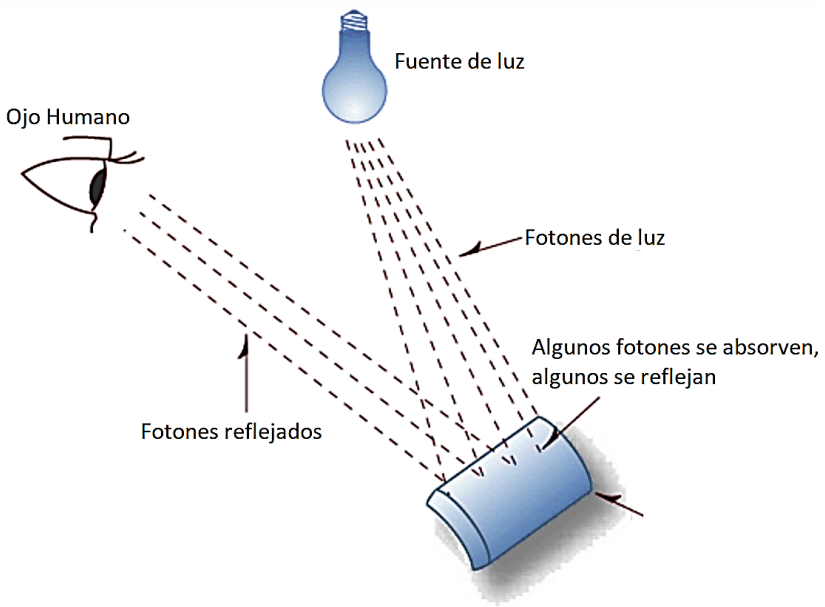
Una imagen de pantalla completa se denomina frame (cuadro).

OpenGL admite tres clases de primitivas geométricas: puntos, segmentos de línea y polígonos cerrados. Se especifican mediante vértices. Cada vértice está asociado con sus atributos como la posición, el color, la normalidad y la textura. OpenGL proporciona 10 primitivos como se muestra. Esfera, caja 3D y pirámide no son primitivas. Por lo general, se ensamblan utilizando un triángulo primitivo o quad.

El color es simplemente una longitud de onda de luz que es visible para el ojo humano. Un objeto blanco refleja uniformemente todas las longitudes de onda de los colores, y un objeto negro absorbe todas las longitudes de onda uniformemente.

Considerando la luz como una partícula, cualquier objeto dado cuando es iluminado por una fuente de luz es golpeado por fotones.

El reflejo de los fotones de un objeto depende de los tipos de átomos, el número de cada tipo y la disposición de los átomos (y sus electrones) en el objeto.

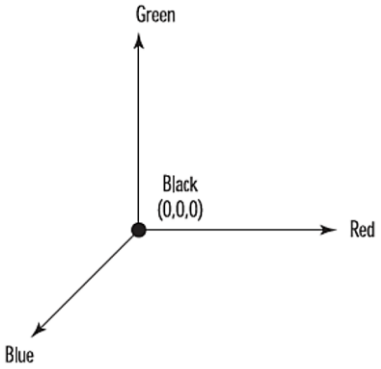


El color se especifica mediante tres valores de color positivos, se puede modelar como un volumen llamado espacio de color RGB

Las coordenadas roja, verde y azul se especifican igual que las coordenadas x, y i z.

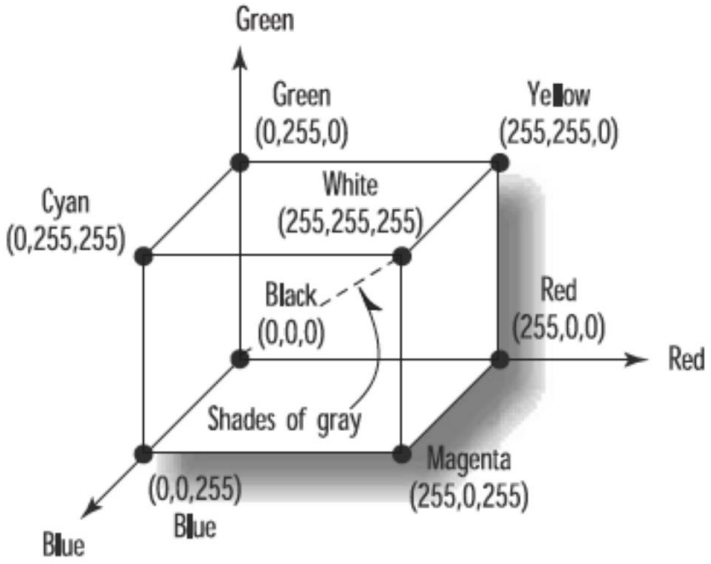
En el origen (0,0,0), la intensidad relativa de cada componente es cero y el color resultante es negro.

Con 8 bits para cada componente, 255 a lo largo del eje representa la saturación total de ese componente.



El “cubo de color” contiene todos los colores posibles, ya sea en la superficie del cubo o dentro del interior del cubo.

Todos los posibles tonos de gris entre el blanco y el negro se encuentran internamente en la línea diagonal entre la esquina en (0,0,0) y la esquina en (255,255,255).



B. La función glColor

```
void glColorNT(red, green, blue, alpha);
```

N = número de parámetros

• 3 RGB
• 4 RGBA (alfa)
• T = Tipo: b, d, f, i, s, ub, ui o us para byte, doble, flotante, entero, short, byte sin signo, entero sin signo y short sin signo
• Otra versión de la función tiene una v adjunta: hasta el final; esta versión toma una matriz que contiene los argumentos (la v significa vectorizado)

La mayoría de los programas OpenGL que verá usan glColor3f y especifican la intensidad de cada componente como 0.0 para ninguno o 1.0 para la intensidad total. Internamente, OpenGL representa los valores de color como valores de coma flotante.

A medida que los búferes de color de punto flotante de mayor resolución evolucionen, el hardware de color representará más fielmente el uso de flotadores.

3. PRACTICA

A. Dibujemos un triángulo usando la primitiva **GL_TRIANGLES**
Digita y entiende el siguiente código, verifica el correcto funcionamiento.

```
#include <GLFW/glfw3.h>

int main(int argc, const char* argv[]) {
    GLFWwindow* win;
    if (!glfwInit()) {
        return -1;
    }
    win = glfwCreateWindow(640, 480, "Ejemplo 1", NULL, NULL);
    if (!win)
    {
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(win);
    while (!glfwWindowShouldClose(win)) {
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_TRIANGLES);
        {
            glColor3f(1.0, 0.0, 0.0);
            glVertex2f(0, .5);
            glColor3f(0.0, 1.0, 0.0);
            glVertex2f(-.5, -.5);
            glColor3f(0.0, 0.0, 1.0);
            glVertex2f(.5, -.5);
        }
        glEnd();
        glfwSwapBuffers(win);
        glfwPollEvents();
    }
    glfwTerminate();
    return 0;
}
```

Salida:

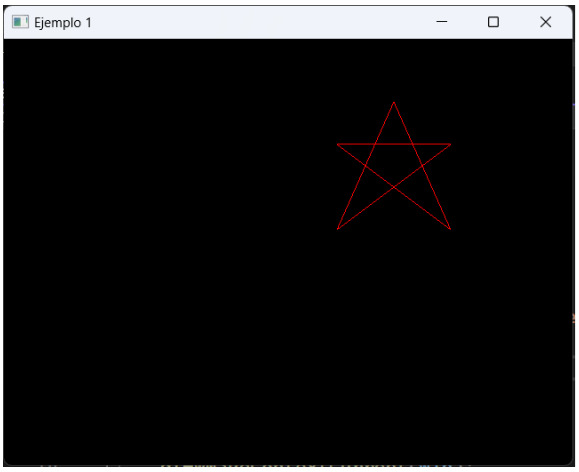
B. De manera similar, utilizando la primitiva **GL_QUADS**, escribe un programa que dibuje el cubo RGB, similar a:

C. Probemos otra primitiva, generemos una estrella básica con **GL_LINE_LOOP**

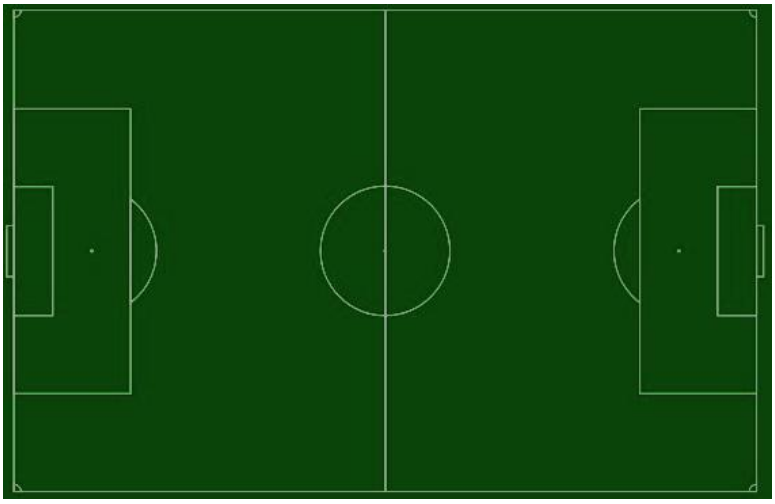
```
#include <GLFW/glfw3.h>

int main(int argc, const char* argv[]) {
    GLFWwindow* win;
    if (!glfwInit()) {
        return -1;
    }
    win = glfwCreateWindow(640, 480, "Ejemplo 2", NULL, NULL);
    if (!win)
    {
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(win);
    while (!glfwWindowShouldClose(win)) {
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_LINE_LOOP);
        {
            glColor3f(1.0, 0.0, 0.0);
            glVertex2f(0.5, 0.5);
            glVertex2f(0.1, 0.1);
            glVertex2f(0.3, 0.7);
            glVertex2f(0.5, 0.1);
            glVertex2f(0.1, 0.5);
        }
        glEnd();
        glfwSwapBuffers(win);
        glfwPollEvents();
    }
    glfwTerminate();
    return 0;
}
```

Salida:



D. Usando las primitivas que consideres pertinente, dibuja las líneas de un campo de futbol, similar a:



E. Investigue y explique para que sirve el parámetro alfa en glColor. De un ejemplo de su uso.