Imagen que contiene alimentos, taza

Descripción generada automáticamenteImagen que contiene dibujo, alimentos, taza

Descripción generada automáticamenteUNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

**REPORTE DE PRÁCTICA Nº** 02

**NOMBRE COMPLETO:** Montiel Aviles Axel Fernando

**Nº de Cuenta:** 422051042

**GRUPO DE LABORATORIO:** 03

**GRUPO DE TEORÍA:** 04

**SEMESTRE** 2025-2

**FECHA DE ENTREGA LÍMITE:** 22/feb/2025

**CALIFICACIÓN: \_\_\_\_\_\_\_\_**

REPORTE DE PRÁCTICA:

# 1.- Ejecución de los ejercicios que se dejaron.

comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

1.- Dibujar las iniciales de sus nombres, cada letra de un color diferente

En este caso mis iniciales corresponden a **F**ernando **M**ontiel **A**viles.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

En este caso se usaron los mismos vértices de la práctica pasada, con la única diferencia de que ahora estos tienen un color RGB.

Mostrando los vértices, con colores:

Calendario

El contenido generado por IA puede ser incorrecto.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Imagen que contiene tabla, grupo, monitor, computadora

El contenido generado por IA puede ser incorrecto.

Imagen que contiene electrónica, tabla, sostener, teclado

El contenido generado por IA puede ser incorrecto.

Pantalla de un computador

El contenido generado por IA puede ser incorrecto.

Un reloj digital en la pantalla

El contenido generado por IA puede ser incorrecto.

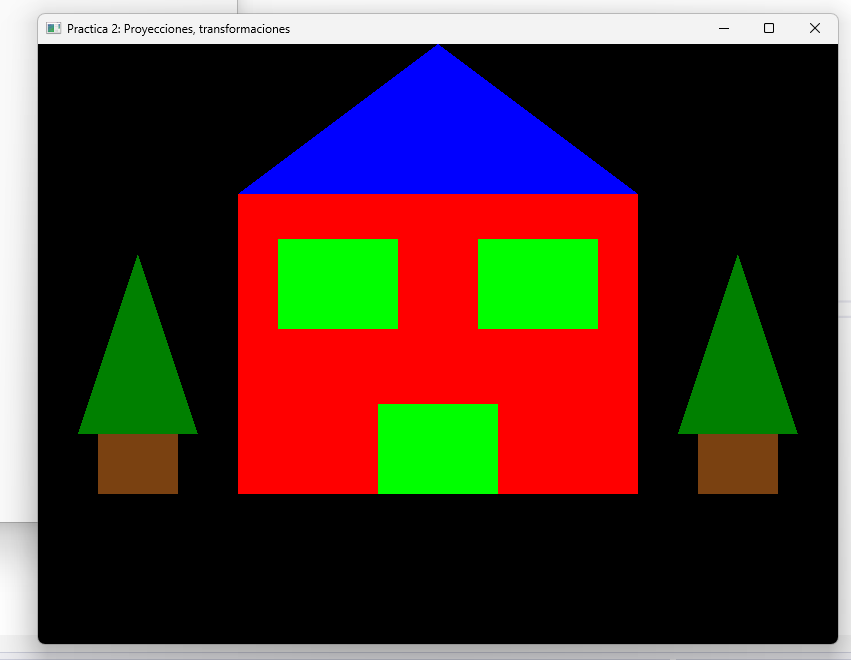
Ahora basta con pasar la longitud de nuestro arreglo de vértices a nuestra función y agregarla a la lista de meshColorList.

Texto

El contenido generado por IA puede ser incorrecto.

2.- Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando pirámides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp

Casa hecha con pirámides y cubos:



Shaders utilizados para dar color a las figuras:

Por cada color de shader se tendrá un par de archivos, en este caso el archivo .frag en todos los casos será el mismo:

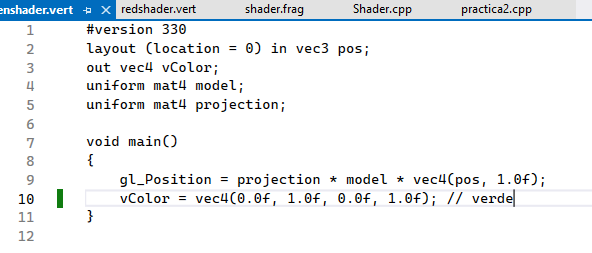
Interfaz de usuario gráfica

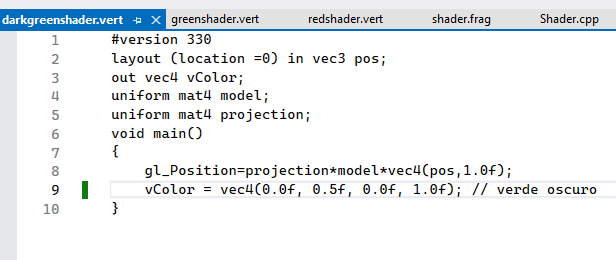
El contenido generado por IA puede ser incorrecto.

Ahora sí, definimos los colores en los archivos .vert

Interfaz de usuario gráfica, Texto, Aplicación

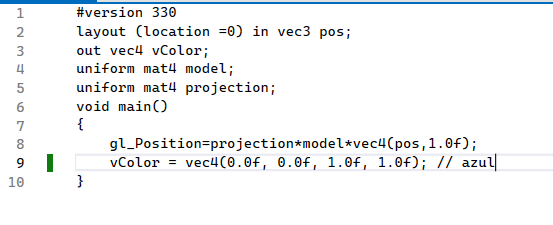
El contenido generado por IA puede ser incorrecto.



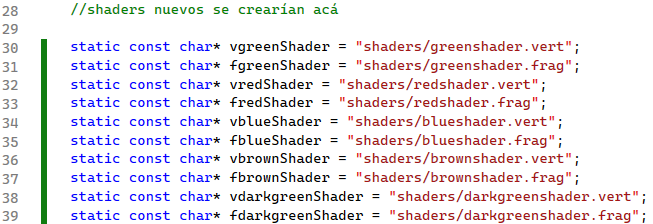


Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.



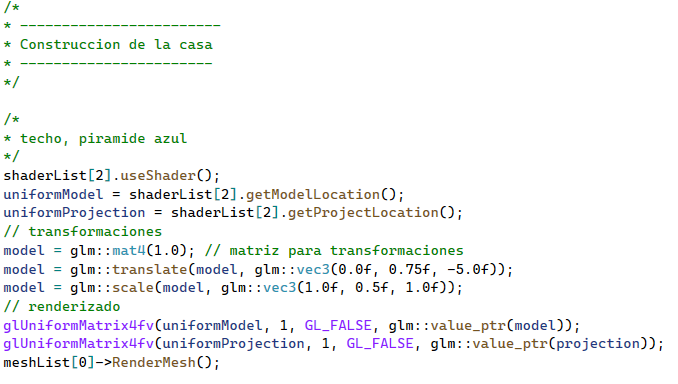
Una vez creado los shaders, ahora tenemos que poner su ruta en el código:



Después de poner las rutas de los shaders, ahora en nuestra función CreateShaders, agregamos cada uno a la lista de shaders, instanciando cada uno y agregándolo al vector de instancias de shaders.

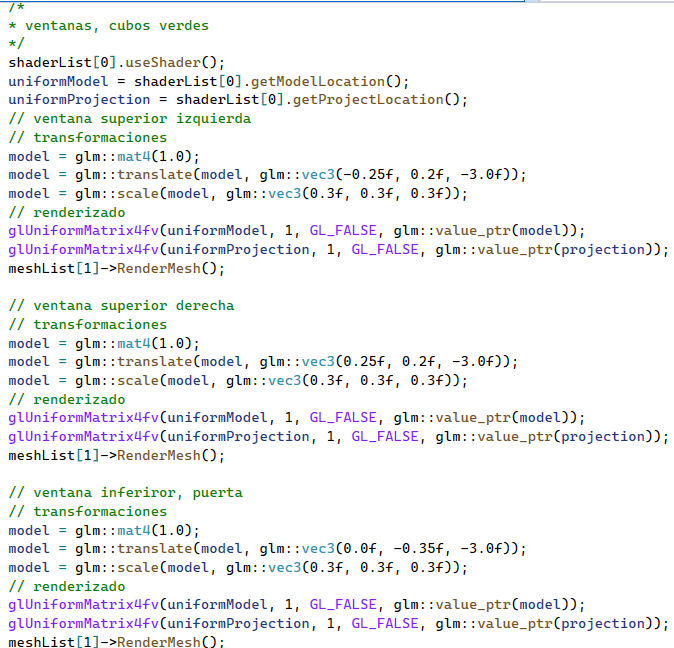


Una vez hecho está configuración, ya podemos empezar a instanciar cada polígono dentro de nuestro ciclo while de dibujado. El principal cambio que va a tener ahora nuestro código es que antes de renderizar cualquier polígono, cambiaremos el shader a usar, de esta forma podemos obtener figuras de distintos colores. Importante recordar igual los índices de la pirámide y el cubo.



Texto

El contenido generado por IA puede ser incorrecto.



Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

# 2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios

Si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

# 3.- Conclusión los ejercicios del reporte

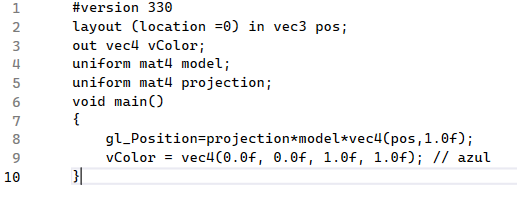
**La primera parte** fue la más sencilla, aunque un poco laborioso. Basto con copiar todos los vértices de la práctica anterior, con la diferencia de poner el color del que van a ser. Siguiendo el siguiente formato:

Interfaz de usuario gráfica, Texto, Aplicación

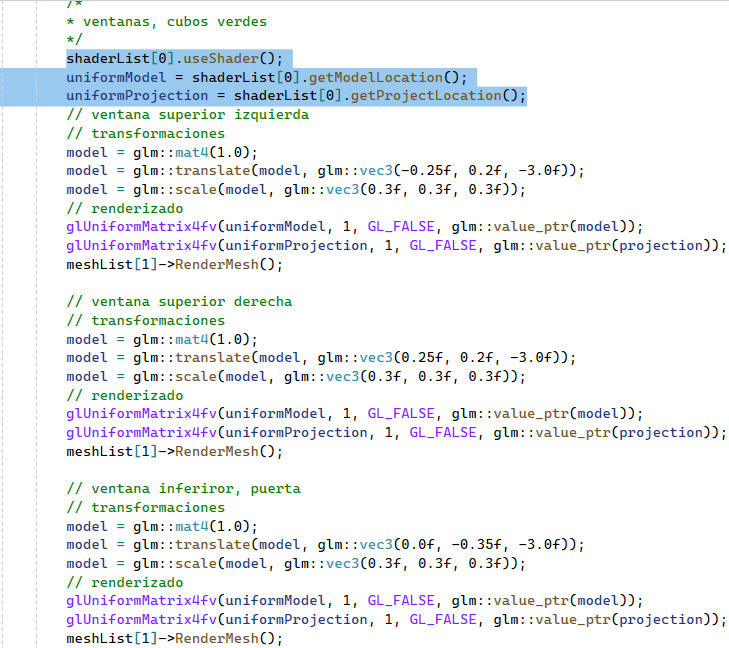
El contenido generado por IA puede ser incorrecto.

Siguiendo este formato es sencillo asignar los vértices, así como su color. Algo que cambió respecto a el código proporcionado en clase fue la profundidad, pues en el eje Z había un valor que no permitía ver la figura en pantalla, por lo que puse un valor más adecuado.

**Para la segunda parte,** se uso el concepto de shaders, estos shaders nos ayudan a dar color a figuras en 3 dimensiones, en este caso el cubo y la pirámide. Su estructura es simple:



En este caso lo que más nos interesaba era cambiar el color de nuestro polígono, esto lo logramos gracias a este archivo shader, en este caso se crearon 5 archivos de shader nuevos y cada uno se uso para distintas partes de la casa. En este caso de la casa, un ejemplo de la reutilización de los shader fue en esta parte del código:



Como podemos ver solo se usa una vez el archivo de shader en la parte resaltada, ya que las siguientes 3 figuras a renderizar usan el mismo shader, en este caso, el shader de color verde. Dándonos una idea de cómo podemos optimizar nuestro código sin reescribir esas instrucciones para cada una de las figuras.

Comentarios generales: Me gustaría haber hecho que rotara la casa, para lograr apreciar el 3D.

# Conclusión general

En esta práctica se vio el uso de más funciones en OpenGL, específicamente sobre el uso de colores en la computación gráfica. En está práctica se logro ver el funcionamiento de los shaders y los colores, cómo es que estos interactúan con nuestros modelos, en este caso fueron modelos muy sencillos, el cubo y la pirámide. Está práctica nos sirve para darnos una idea de cómo será trabajar más adelante en el curso con estos archivos shader, practicando y que nos sea más familiar al trabajar con modelos más complejos, considero que fue una buena introducción a este concepto, abriendo las posibilidades de lo que queramos representar más adelante. También nos ayudó a mejorar nuestra manera de hacer transformaciones geométricas para mover a nuestros modelos desde el origen hacia dónde queremos que estén.

## Bibliografía en formato APA

* *LearnOpenGL - Shaders*. (n.d.). <https://learnopengl.com/Getting-started/Shaders>