UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 03

NOMBRE COMPLETO: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 01 DE MARZO DE 2025

CALIFICACIÓN: _____

Actividades

1. Instanciar primitivas geométricas para recrear el dibujo de la práctica pasada en 3D,se requiere que exista un piso; la casa tiene una ventana azul circular justo en medio de la pared trasera, 2 ventanas verdes en cada pared lateral iguales a las de la pared frontal y solo puerta en la pared frontal.

Para el desarrollo de este ejercicio se hizo uso de cada una de las figuras que se almacenaron dentro de la estructura *meshList* a excepción de la esfera.

Primero se instanció una pirámide cuadrada, la cual representa el techo de nuestra casa. Dicha pirámide se ubica en la posición cuatro de la estructura.

```
378         //Techo
379         model = glm::mat4(1.0f);
380         color=glm::vec3(0.0f,0.0f,1.0f);
381         //Opcional duplicar esta traslación inicial para posicionar en -Z a lo
382         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
383         model = glm::translate(model, glm::vec3(0.0f, -1.25f, -4.0f));
384         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f
385         model = glm::scale(model, glm::vec3(1.0f, 0.5f, 1.0f));
386         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));/
387         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
388         meshList[4]->RenderMeshGeometry();
```

También se hizo uso de las funciones *translate* y *scale* para modificar su posición y escala respectivamente. Dentro de la variable color definimos el color del objeto que se instancia.

Bajo el mismo procedimiento se instancian las demás figuras de nuestra casa a excepción de la esfera.

Cubos verdes

```
402         //Cubos verdes
403         model = glm::mat4(1.0f);
404         color = glm::vec3(0.0f, 1.0f, 0.0f);
405         //Opcional duplicar esta traslación inicial para posicionar en -Z a los c
406         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
407         model = glm::translate(model, glm::vec3(-0.2f, -1.8f, -3.45f));
408         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f))
409         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
410         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FA
411         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
412         meshList[0]->RenderMeshGeometry();
```

Cilindros

```
480         //Cilindros
481         model = glm::mat4(1.0f);
482         color = glm::vec3(0.478f, 0.255f, 0.067f);
483         //Opcional duplicar esta traslación inicial para posicionar en -Z a los
484         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
485         model = glm::translate(model, glm::vec3(-1.0f, -2.4f, -4.0f));
486         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f)
487         model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
488         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//
489         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
490         meshList[2]->RenderMeshGeometry();
```

Conos

```
503         //Cono
504         model = glm::mat4(1.0f);
505         color = glm::vec3(0.0f, 0.5f, 0.0f);
506         //Opcional duplicar esta traslación inicial para posicionar en -Z a los ob
507         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
508         model = glm::translate(model, glm::vec3(1.0f, -2.0f, -4.0f));
509         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
510         model = glm::scale(model, glm::vec3(0.2f, 0.5f, 0.2f));
511         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FAL
512         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
513         meshList[3]->RenderMeshGeometry();
```

Piso

```
526         //Piso
527         model = glm::mat4(1.0f);
528         color = glm::vec3(0.5f, 0.0f, 0.5f);
529         //Opcional duplicar esta traslación inicial para posicionar en -Z a los obj
530         //model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
531         model = glm::translate(model, glm::vec3(0.0f, -2.6f, -3.6f));
532         model = glm::scale(model, glm::vec3(3.2f, 0.1f, 3.2f));
533         //model = glm::rotate(model, 180*toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
534         model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
535         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALS
536         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
537         meshList[0]->RenderMeshGeometry();
```

Figura	Representación	Posición en MeshList
Cubos	Casa, ventanas, puertas y piso	0
Pirámide cuadrangular	Techo	4
Conos	Árboles	3
Cilindros	Troncos	2

Esfera	Ventana circular	-

Para la esfera, primero modificamos el radio desde la instancia del objeto.

```
41      Sphere sp = Sphere(0.2, 20, 20); //recibe radio, slices, stacks
```

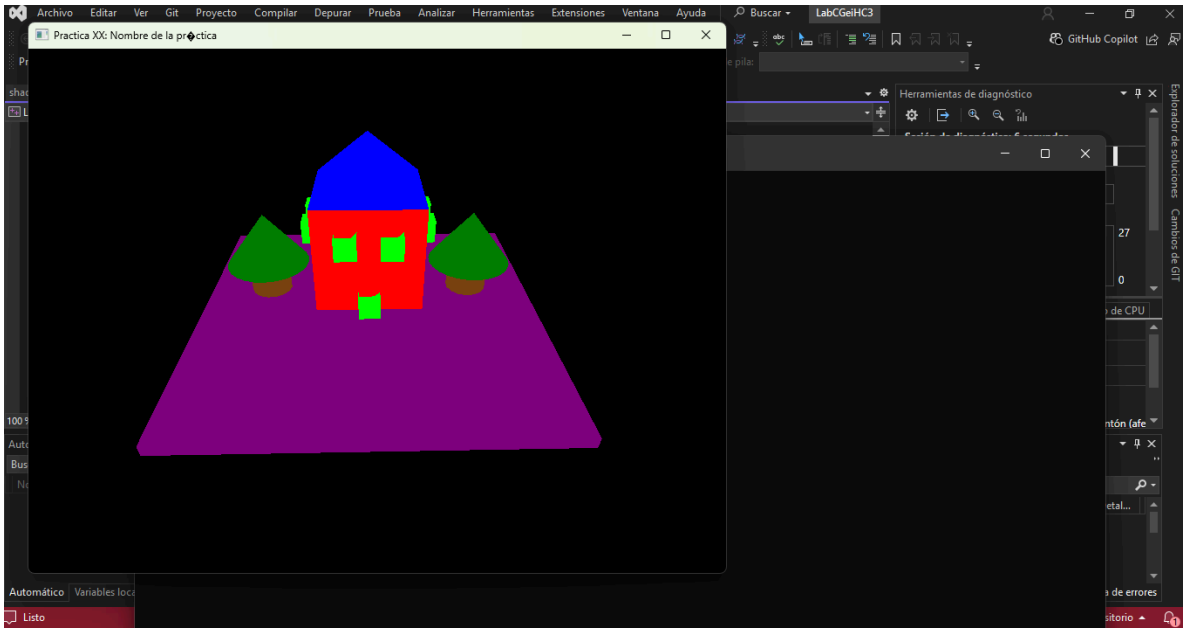
Usamos un radio de 0.2.

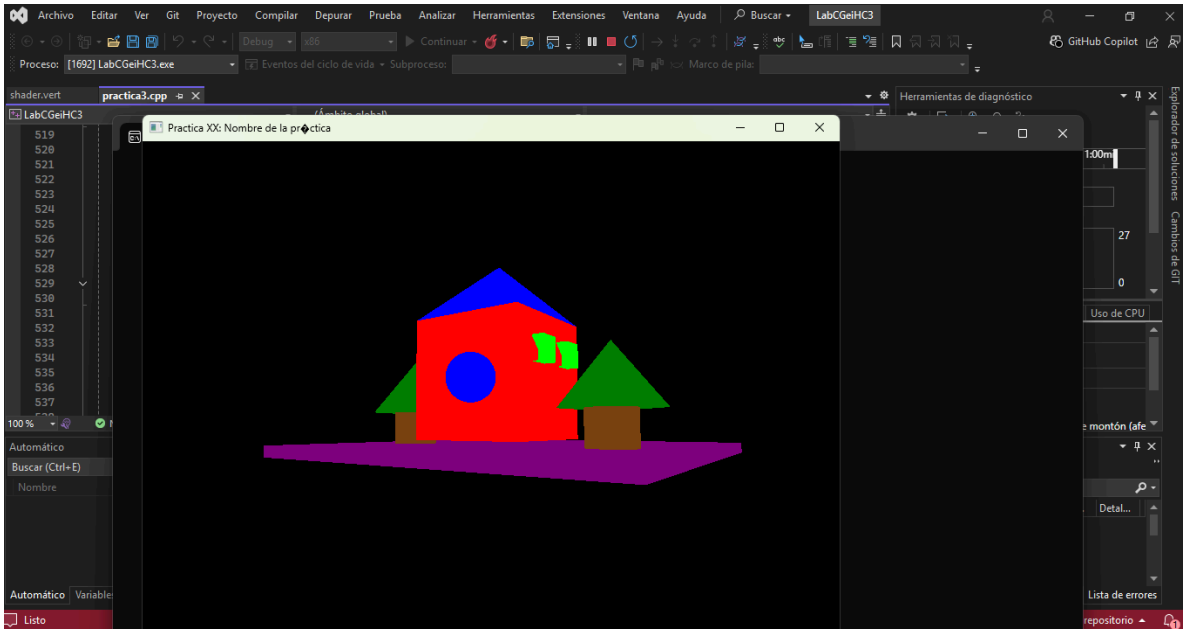
Luego, usamos las funciones igual que con las figuras anteriores, con la diferencia que no usaremos *MeshList* para usar alguna figura, sino que se usará la función correspondiente para dibujar la pirámide.

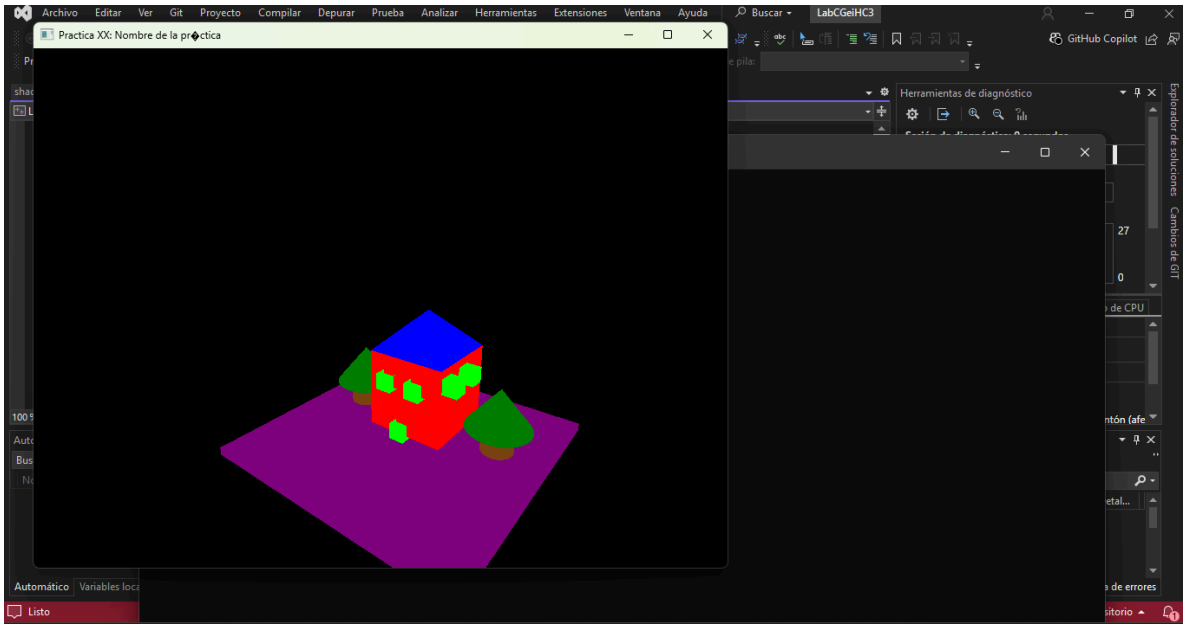
```
354         model = glm::mat4(1.0);
355         //Traslación inicial para posicionar en -Z a los objetos
356         model = glm::translate(model, glm::vec3(0.0f, -2.0f, -4.4f));
357         //otras transformaciones para el objeto
358         //model = glm::scale(model, glm::vec3(0.5f,0.5f,0.5f));
359         model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));
360         model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
361         model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));
362         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
363         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
364         //se programe cambio entre proyección ortogonal y perspectiva
365         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
366         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
367         color = glm::vec3(0.0f, 0.0f, 1.0f);
368         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
369         //meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
370         //meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide
371         sp.render(); //dibuja esfera
```

Igualmente usamos *translate* para mover la esfera y esta vez no usaremos *scale* porque el tamaño de la esfera ya fue modificado al cambiar su radio.

La ejecución quedó de la siguiente manera:


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
Buscar
LabCGeiHC3
GitHub Copilot
Practica XX: Nombre de la pr&#65533;ctica
Herramientas de diagnóstico
Automático
Listo


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
Buscar
LabCGeiHC3
GitHub Copilot
Proceso: [1692] LabCGeiHC3.exe
shader.vert
practica3.cpp
Practica XX: Nombre de la pr&#65533;ctica
Herramientas de diagnóstico
Automático
Buscar (Ctrl+E)
Nombre
Lista de errores
Listo



Problemas presentados

El problema que se presentó al elaborar el ejercicio fue la implementación del piso, ya que primero se pensó en instanciar un cuadrado en 2D, sin embargo, iba a resultar más sencillo instanciar un cubo y modificar su escala principalmente en el eje Y para que pueda simular un plano.

Otro problema que se presentó fue la incrustación entre las figuras, a veces provocan que se distorsionan un poco, esto se corrigió en cierta medida modificando la traslación en el eje Z.

Conclusión

Este ejercicio se realizó con éxito, sin tantas problemáticas presentadas, esto gracias a la explicación dada en la clase de laboratorio donde se mencionaron partes importantes del código como las diferentes figuras, como es que se instancia la esfera y cómo modificar el color de cada una de las formas. Además del manejo de la cámara tanto con mouse como con teclado, mediante el uso de shaders. El único problema presentado es en la visualización de las figuras pero las funciones *translate* y scale ayudan a una mejor visualización.