aaUNIVERSIDAD 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: Vargas López Miguel Adán

Nº de Cuenta: 421079522

GRUPO DE LABORATORIO: 3

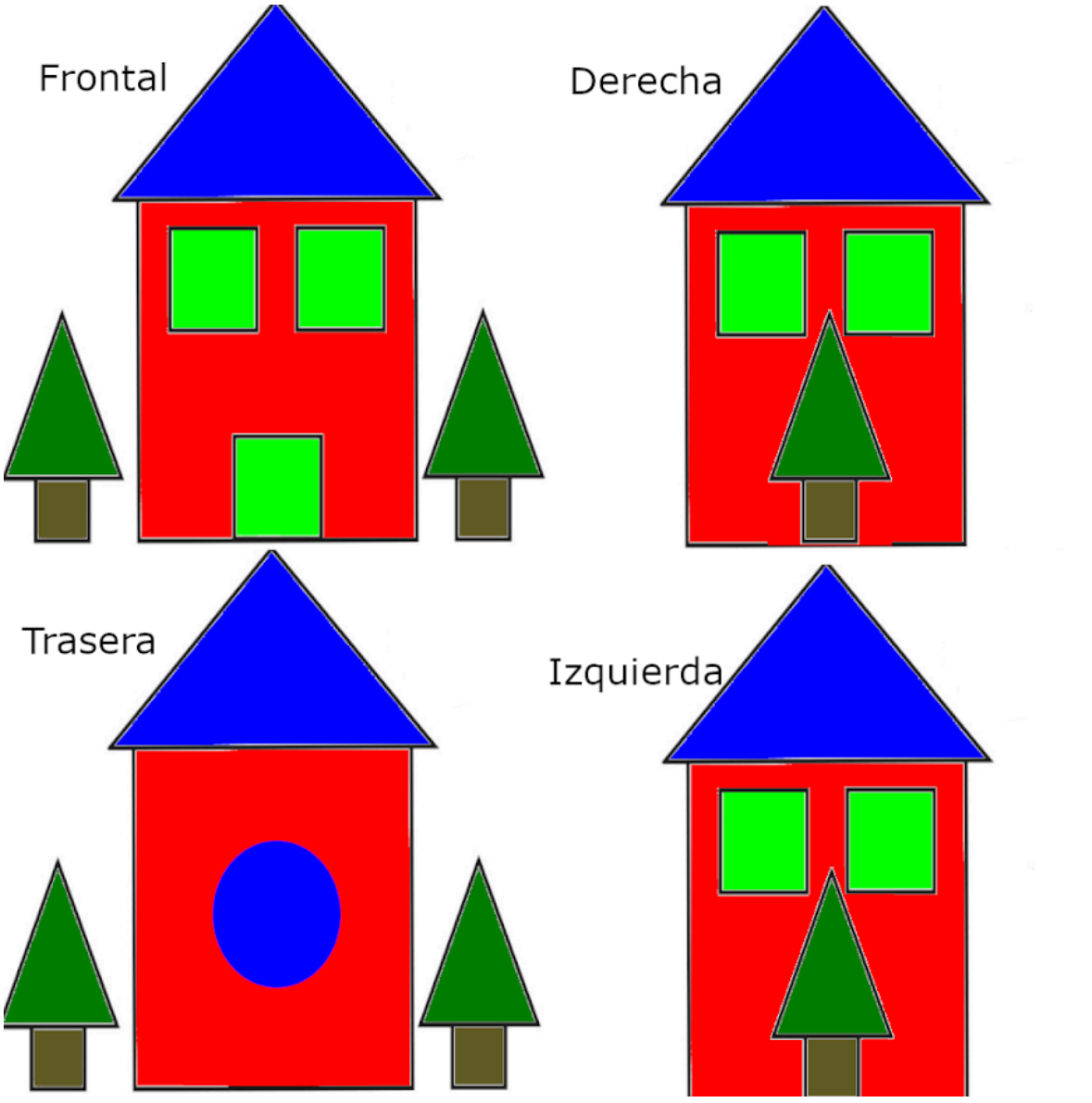
GRUPO DE TEORÍA: 4

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 27/08/2024

CALIFICACIÓN: _____

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 realizar este programa lo que tuve que modificar fue primero en el recurso shader.vert, comentar la línea 11 y descomentar la línea 10 para así habilitar la cámara en mi escena.

shader.vert | Shader.cpp | Sphere.h | Practica3.cpp

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
uniform vec3 color;
uniform mat4 view;
void main()
{
    gl_Position=projection*view*model*vec4(pos,1.0f);
    //gl_Position=projection*model*vec4(pos,1.0f);
    vColor=vec4(color,1.0f);

}
```

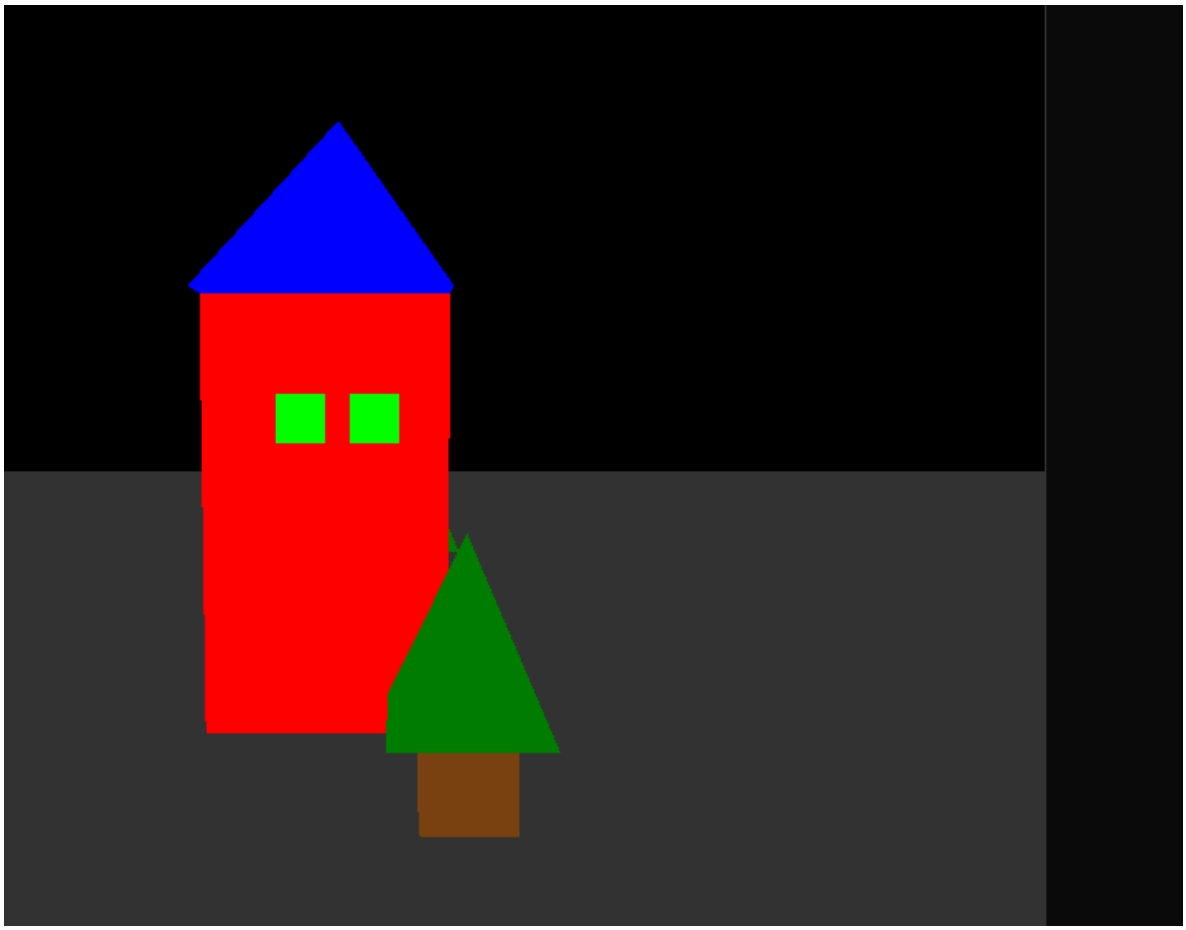
Seguido de instanciar cada objeto en la escena como los pinos, las ventanas, el techo, la casa dentro de la escena. Además de agregar un piso para la vista.Aquí algunos ejemplo de la instanciación de estos objetos:

```
//piso
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));  //al presionar l
model = glm::translate(model, glm::vec3(0.0f, -1.8f, -3.0f));
model = glm::scale(model, glm::vec3(30.0f, 0.01f,30.0f));
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();

//cubo puerta central
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));  //al presionar l
model = glm::translate(model, glm::vec3(0.0f, -1.5f, -2.5f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.01f));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
//glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[0]->RenderMesh();
```

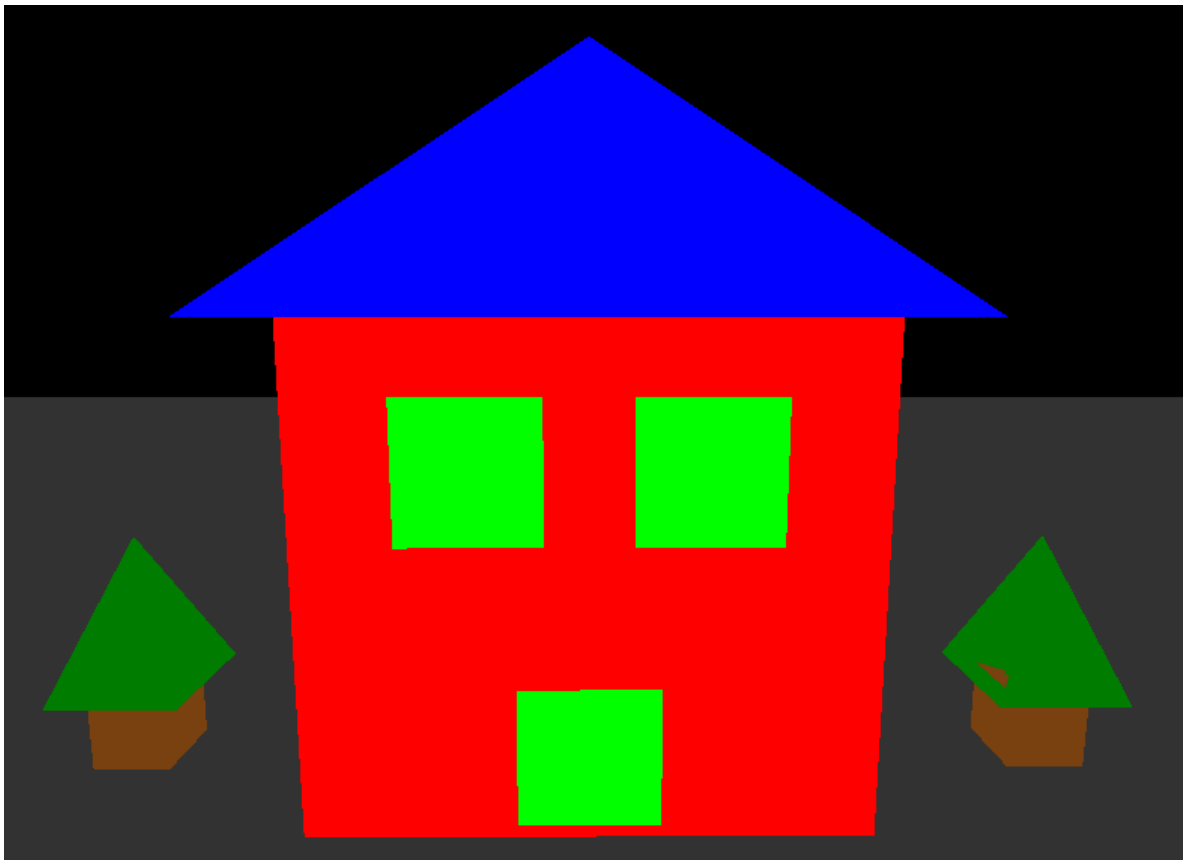
```
//cubo tronco izquierda
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));  //al presionar la tecl
model = glm::translate(model, glm::vec3(-1.8f, -1.6f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
color = glm::vec3(0.478, 0.255, 0.067);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
//glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh();

//pino izquierda
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));  //al presionar la tecl
model = glm::translate(model, glm::vec3(-1.8f, -1.15f, -3.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.6f, 0.5f));
color = glm::vec3(0.0f, 0.5f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
//glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[4]->RenderMeshGeometry();
```

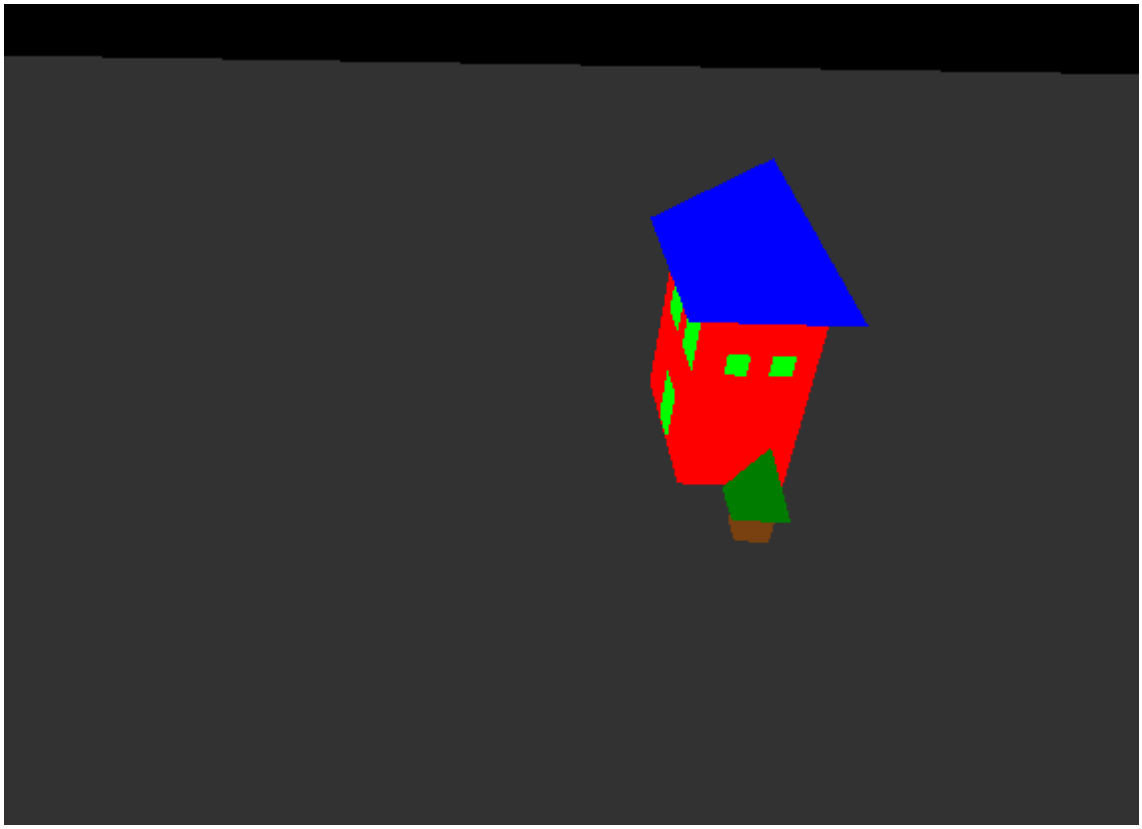
Complicaciones:

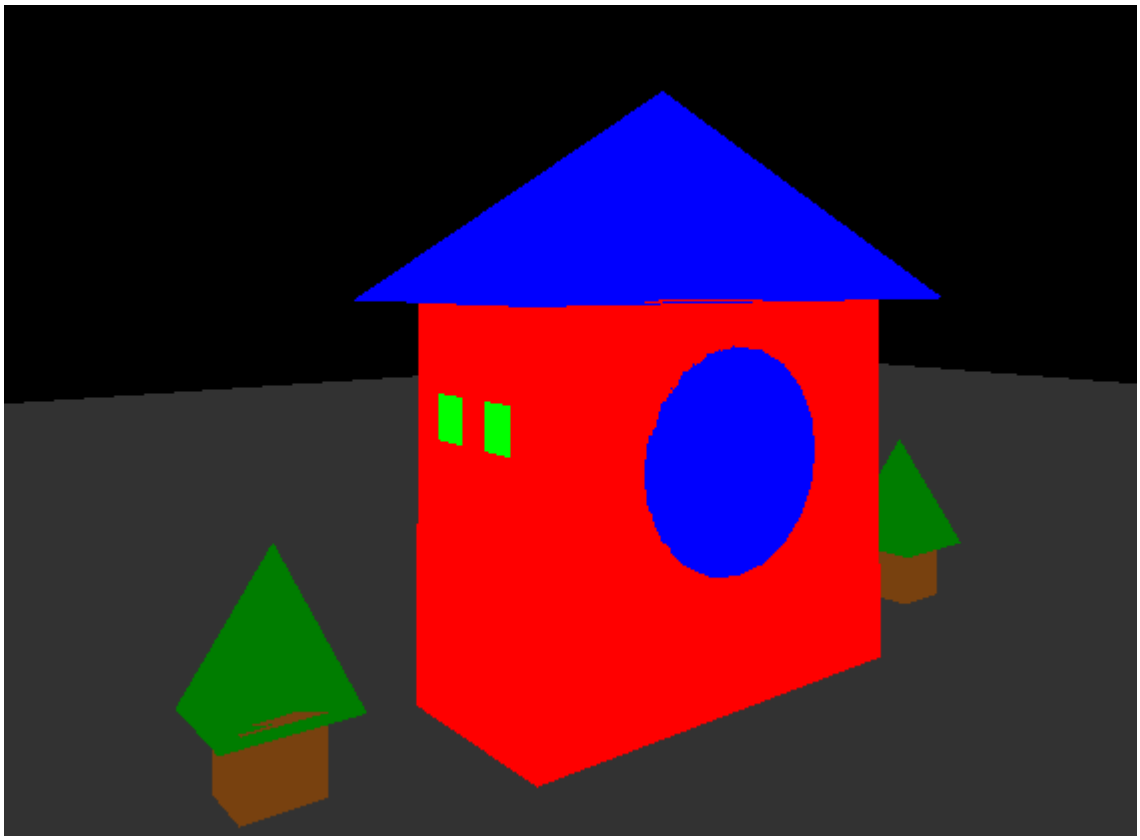
Dentro de este ejercicio las complicaciones tuvieron que ver con el acomodo de las figuras ya que al principio las había instanciado sin conciencia de la cámara, al acceder a la cámara pude ver que había objetos desacomodados o que en cierta vista parecía concordar con el dibujo cuando no lo era.



Pero al acomodar cada uno en su lugar y escalando los objetos de una forma adecuada podía tener vistas como las siguientes:







Conclusión:

Para esta práctica tuvimos que trabajar sobre todo con los métodos de traslación, escalado y de rotación, además de implementar una cámara para la escena que nos permitió tener un mejor panorama de nuestra escena. Además, dentro de todas las instancias de los objetos que declaramos en el código, no solo se pudo ver la importancia de el orden en el que se renderizan, sus profundidades o sus colores, sino también ver que incluso el orden en el que se hacen estas modificaciones al objeto puede ser importante, ya que no es lo mismo escalar un objeto para rotarlo y viceversa, en el caso del cilindro azúl, vi que era importante trasladar, luego rotar y por último escalar el objeto.