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 02

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: 23 DE FEBRERO DE 2025

CALIFICACIÓN: _____

Actividades

1. Generar las figuras copiando los vértices del triángulo rojo y cuadrado verde: triángulo azul, triángulo verde (0,0.5,0), cuadrado rojo, cuadrado verde, cuadrado café (0.478, 0.255, 0.067).

Para esta actividad, se crearon dentro de la función *CrearLetrasyFiguras* los arreglos para los vértices de los triángulos y cuadrados solicitados, copiando los vértices que ya estaban establecidos, cambiando únicamente el color.

Triángulo azul:

```
124     GLfloat vertices_trianguloazul[] = {
125         //X         Y           Z           R       G       B
126         -1.0f,  -1.0f,      0.5f,           0.0f,   0.0f,   1.0f,
127         1.0f,   -1.0f,      0.5f,           0.0f,   0.0f,   1.0f,
128         0.0f,   1.0f,       0.5f,           0.0f,   0.0f,   1.0f,
129
130     };
131
132     MeshColor* trianguloazul = new MeshColor();
133     trianguloazul->CreateMeshColor(vertices_trianguloazul, 18);
134     meshColorList.push_back(trianguloazul);
```

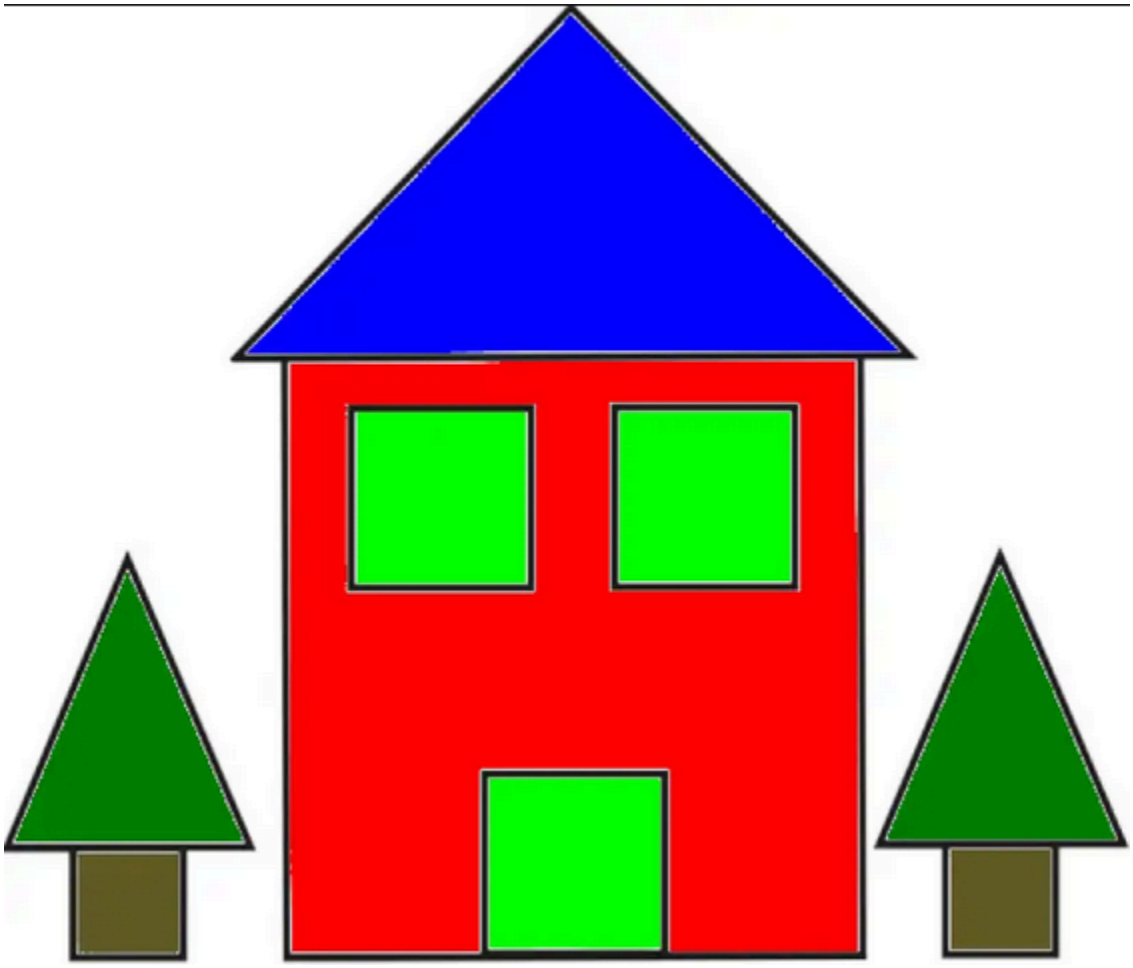
Cuadrado verde:

```
163     GLfloat vertices_cuadradoverde[] = {
164         //X         Y           Z           R       G       B
165         -0.5f,  -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
166         0.5f,   -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
167         0.5f,   0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
168         -0.5f,  -0.5f,      0.5f,           0.0f,   1.0f,   0.0f,
169         0.5f,   0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
170         -0.5f,  0.5f,       0.5f,           0.0f,   1.0f,   0.0f,
171
172     };
173
174     MeshColor* cuadradoverde = new MeshColor();
175     cuadradoverde->CreateMeshColor(vertices_cuadradoverde, 36);
176     meshColorList.push_back(cuadradoverde);
```

De esta misma forma se crearon el resto de los triángulos y cuadrados con el color solicitado.

2. Usando la proyección ortogonal generar el siguiente dibujo a partir de instancias de las figuras anteriormente creadas , recordar que todos se dibujan en el origen y por transformaciones geométricas se desplazan

---



Para esta actividad, en la función *main* dentro del ciclo *while* se usaron las figuras previamente creadas que se almacenaron dentro de la estructura *meshColorList*, donde se utilizaron cortas funciones de traslación y escala para poder modificar las figuras tanto de su tamaño como de su ubicación.

Cuadrado rojo:

```
//Agregamos cuadrado rojo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.3f, -5.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES P
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4]->RenderMeshColor();
```

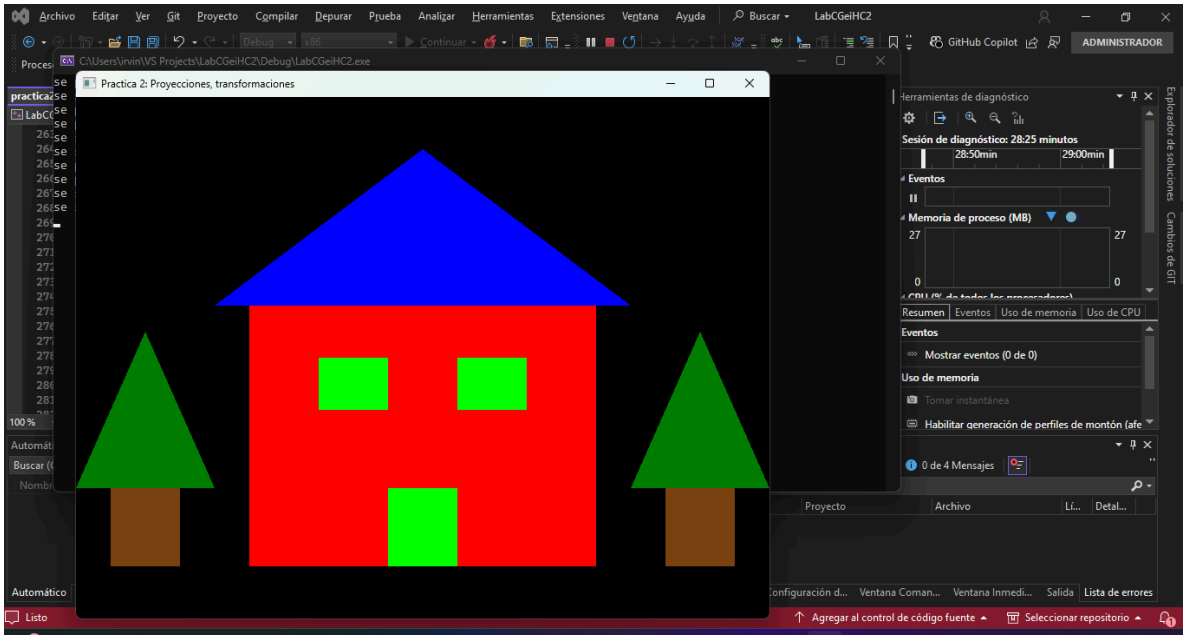
Triángulo azul:

```
273         //Triangulo azul
274         model = glm::mat4(1.0);
275         model = glm::translate(model, glm::vec3(0.0f, 0.5f, -4.0f));
276         model = glm::scale(model, glm::vec3(0.6f, 0.3f, 0.5f));
277         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PAR
278         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
279         meshColorList[2]->RenderMeshColor();
```

Es importante tener en cuenta el orden en el que se fueron agregando las figuras en la estructura para poder acceder a ellas de acuerdo a su posición y modificar sus formas.

Agregamos cada una de las figuras y usamos la función *translade* para mover la figura tanto en el eje *x, y* o *z* y así poder colocarlas en el lugar correspondiente. La función *scale* nos permite modificar las dimensiones de las figuras, ya sea alargarla o reducirla dentro de cualquiera de los ejes.

Finalmente, la ejecución queda de la siguiente forma:



Problemas presentados

La problemática que se presentó durante el desarrollo del ejercicio fue encontrar la forma en la que se le dé prioridad a las figuras para que se presenten por encima de otras. Siendo más puntual, se mostraba el cuadrado verde detrás del rojo, donde la primer propuesta para solucionar el problema fue cambiar el orden en la que se agregaron las figuras en la estructura, y dentro del main llamar primero el cuadro rojo, sin embargo esto no soluciona el problema, por lo que se modificaron los argumentos dentro de la función *scale* del cuadro rojo, y se le asignó el valor de -5 en el eje *z*, esto con el fin de alejar la figura y estuviera separada del cuadro

verde una unidad, ya que el valor en *z* de la función *scale* para el cuadrado verde es de -4. Esto soluciona el problema presentado y se pudo continuar el ejercicio con normalidad.

Conclusiones

Durante el desarrollo de este ejercicio, se logró comprender el uso tanto de proyecciones como de traslaciones geométricas, usando *shader* con el propósito de no repetir figuras con ayuda de los VBO y VAO. La explicación dada en el laboratorio fue de utilidad para realizar los ajustes correspondientes y poder agregar nuevos elementos con diferentes propiedades, ajustar tamaños y coordenadas dentro del dibujo. Además que se entendió el funcionamiento del programa brindado, evitando así realizar ajustes que no son necesarios evitando modificaciones que afecten tanto la compilación como ejecución del programa. El ritmo de explicación me parece adecuado y bastante claro, reduciendo el tiempo de trabajo en el laboratorio, pero eso evita potenciales errores dentro de nuestros trabajos y ejercicios.