UNIVERSIDAD 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 03

NOMBRE COMPLETO: Carballo Ramírez Hanny

Nº de Cuenta: 319236761

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 03

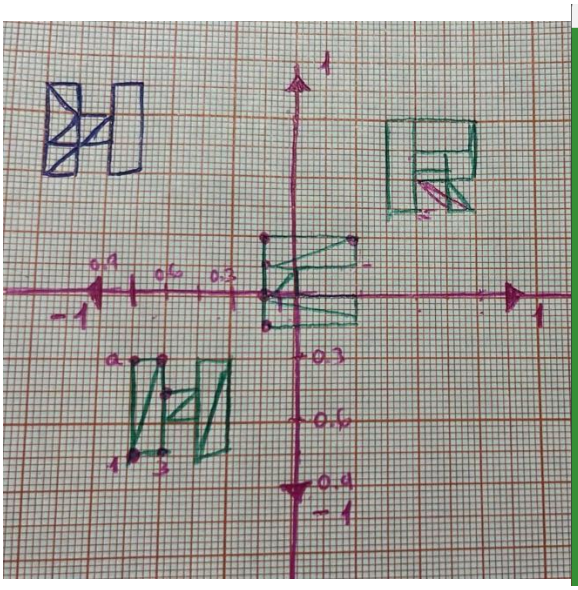
SEMESTRE 2026-2

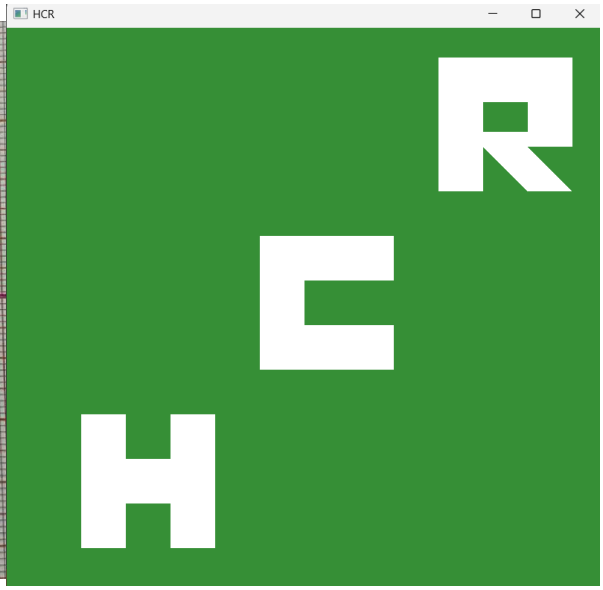
FECHA DE ENTREGA LÍMITE: 1 de marzo 2026

CALIFICACIÓN: _____

1. ACTIVIDADES REALIZADAS. UNA DESCRIPCIÓN DE LOS EJERCICIOS Y CAPTURAS DE PANTALLA DE BLOQUES DE CÓDIGO GENERADOS Y DE EJECUCIÓN DEL PROGRAMA

Se realizaron las modificaciones pertinentes a las letras (mis iniciales) de la práctica 1. Ahora quedaron de la siguiente forma, en donde se puede observar que ahora sí se ven correctamente y claras. También adjunto el boceto utilizado y el código modificado:






```
41      GLfloat vertices[] =
42      {
43          // ------------------------------
44          // LETRA H
45          // ------------------------------
46
47          -0.75f, -0.75f, 0.0f,
48          -0.6f, -0.3f, 0.0f,
49          -0.6f, -0.75f, 0.0f,
50
51          -0.75f, -0.75f, 0.0f,
52          -0.6f, -0.3f, 0.0f,
53          -0.75f, -0.3f, 0.0f,
54
55          -0.6f, -0.45f, 0.0f,
56          -0.6f, -0.6f, 0.0f,
57          -0.45f, -0.45f, 0.0f,
58
59          -0.6f, -0.6f, 0.0f,
60          -0.45f, -0.45f, 0.0f,
61          -0.45f, -0.6f, 0.0f,
62
63          -0.45f, -0.75f, 0.0f,
64          -0.3f, -0.3f, 0.0f,
65          -0.3f, -0.75f, 0.0f,
66
67          -0.45f, -0.75f, 0.0f,
68          -0.3f, -0.3f, 0.0f,
69          -0.45f, -0.3f, 0.0f,
70
```

```
71          // ---------------------------------
72          // LETRA C
73          // ---------------------------------
74          -0.15f, -0.15f, 0.0f,
75          0.3f, -0.15f, 0.0f,
76          -0.15f, 0.0f, 0.0f,
77
78          0.3f, -0.15f, 0.0f,
79          -0.15f, 0.0f, 0.0f,
80          0.3f, 0.0f, 0.0f,
81
82          -0.15f, 0.0f, 0.0f,
83          0.0f, 0.0f, 0.0f,
84          0.0f, 0.15f, 0.0f,
85
86          -0.15f, 0.15f, 0.0f,
87          -0.15f, 0.0f, 0.0f,
88          0.0f, 0.15f, 0.0f,
89
90          -0.15f, 0.3f, 0.0f,
91          0.3f, 0.3f, 0.0f,
92          -0.15f, 0.15f, 0.0f,
93
94          0.3f, 0.3f, 0.0f,
95          -0.15f, 0.15f, 0.0f,
96          0.3f, 0.15f, 0.0f,
97
```

```
99          // LETRA R
100         // ---------------------------
101         0.45f, 0.45f, 0.0f,
102         0.45f, 0.9f, 0.0f,
103         0.6f, 0.9f, 0.0f,
104
105         0.45f, 0.45f, 0.0f,
106         0.6f, 0.9f, 0.0f,
107         0.6f, 0.45f, 0.0f,
108
109         0.6f, 0.9f, 0.0f,
110         0.6f, 0.75f, 0.0f,
111         0.9f, 0.9f, 0.0f,
112
113         0.9f, 0.9f, 0.0f,
114         0.6f, 0.75f, 0.0f,
115         0.9f, 0.75f, 0.0f,
116
117         0.9f, 0.75f, 0.0f,
118         0.75f, 0.75f, 0.0f,
119         0.75f, 0.6f, 0.0f,
120
121         0.9f, 0.75f, 0.0f,
122         0.75f, 0.6f, 0.0f,
123         0.9f, 0.6f, 0.0f,
124
125         0.6f, 0.6f, 0.0f,
126         0.6f, 0.65f, 0.0f,
127         0.75f, 0.65f, 0.0f,
128
129         0.6f, 0.6f, 0.0f,
130         0.75f, 0.65f, 0.0f,
131         0.75f, 0.6f, 0.0f,
132
133         0.6f, 0.6f, 0.0f,
134         0.75f, 0.45f, 0.0f,
135         0.9f, 0.45f, 0.0f,
136
137         0.6f, 0.6f, 0.0f,
138         0.75f, 0.6f, 0.0f,
139         0.9f, 0.45f, 0.0f,
```

```
271             glDrawArrays(GL_TRIANGLES, 0, 66);
```

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

Para desarrollar este ejercicio se reutilizaron las coordenadas previamente definidas para las letras, y únicamente se incorporaron los colores correspondientes. En este caso, se asignó rojo a la “H”, verde a la “C” y azul a la “R”.

```
        //X    Y     Z            R      G      B
        //    ------ LETRA H ------
    -0.6f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.6f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.5f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.6f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.5f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.5f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,

    -0.3f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.3f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.2f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.3f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.2f, -0.2f, 0.0f,   1.0f, 0.0f, 0.0f,
    -0.2f,  0.2f, 0.0f,   1.0f, 0.0f, 0.0f,

    -0.5f,  0.05f, 0.0f,  1.0f, 0.0f, 0.0f,
    -0.5f, -0.05f, 0.0f,  1.0f, 0.0f, 0.0f,
    -0.3f,  0.05f, 0.0f,  1.0f, 0.0f, 0.0f,
    -0.5f, -0.05f, 0.0f,  1.0f, 0.0f, 0.0f,
    -0.3f, -0.05f, 0.0f,  1.0f, 0.0f, 0.0f,
    -0.3f,  0.05f, 0.0f,  1.0f, 0.0f, 0.0f,

    //    ------ LETRA C ------
    -0.1f,  0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f,  0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
    -0.1f,  0.1f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f,  0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f,  0.1f, 0.0f,   0.0f, 1.0f, 0.0f,
```

99 % No se encontraron problemas. Línea: 443, Carácter: 9 TA

```

    -0.1f,  0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
    -0.1f, -0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.0f,   0.2f, 0.0f,  0.0f, 1.0f, 0.0f,
    -0.1f, -0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.0f,  -0.2f, 0.0f,  0.0f, 1.0f, 0.0f,
     0.0f,   0.2f, 0.0f,  0.0f, 1.0f, 0.0f,

    -0.1f, -0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f, -0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
    -0.1f, -0.1f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f, -0.2f, 0.0f,   0.0f, 1.0f, 0.0f,
     0.1f, -0.1f, 0.0f,   0.0f, 1.0f, 0.0f,
    -0.1f, -0.1f, 0.0f,   0.0f, 1.0f, 0.0f,

    //    ------ LETRA R ------
    0.2f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.2f, -0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.3f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.2f, -0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.3f, -0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.3f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,

    0.3f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.5f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.3f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.5f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.5f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,
    0.3f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,

    0.3f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
    0.5f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
```

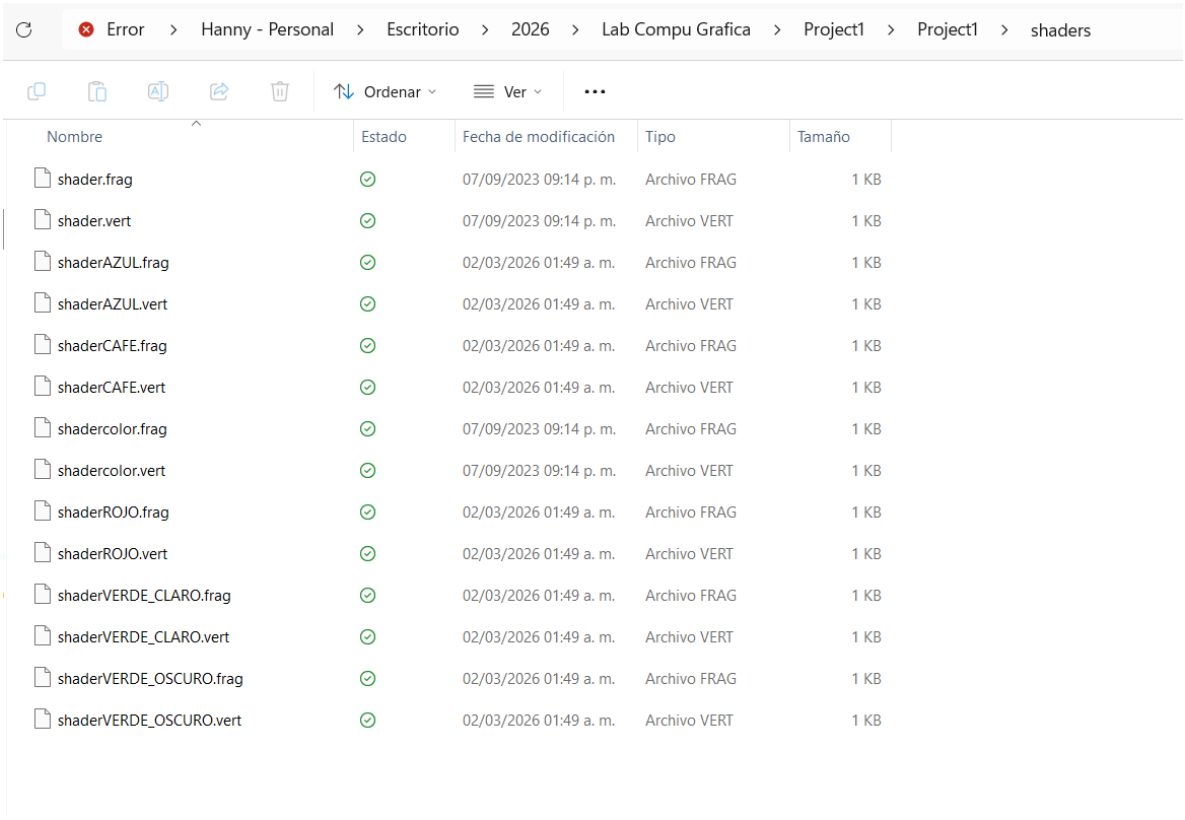
```
0.3f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
0.5f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
0.3f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,
0.5f,  0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
0.5f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,
0.3f,  0.1f, 0.0f,   0.0f, 0.0f, 1.0f,

0.3f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.5f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.3f, -0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.5f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.5f, -0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.3f, -0.05f, 0.0f,  0.0f, 0.0f, 1.0f,

0.3f,  0.05f, 0.0f,  0.0f, 0.0f, 1.0f,
0.5f, -0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
0.4f, -0.2f, 0.0f,   0.0f, 0.0f, 1.0f,
```

2.- Generar el dibujo de la casa de la clase, pero en lugar de instanciar triangulos y cuadrados será instanciando piramides 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

Se realizaron los archivos .vert y .frag de los diferentes colores solicitados



Después se crearon los shaders y se agregaron a la lista para poder ser utilizados

```
void CreateShaders()
{

    Shader* shader1 = new Shader(); //shader para usar índices: objetos: cubo y  pirámide
    shader1->CreateFromFiles(vShader, fShader);
    shaderList.push_back(*shader1);

    Shader* shader2 = new Shader();//shader para usar color como parte del VAO: letras
    shader2->CreateFromFiles(vShaderColor, fShaderColor);
    shaderList.push_back(*shader2);

    Shader* shader3 = new Shader(); //shader AZUL
    shader3->CreateFromFiles(vShaderAZUL, fShaderAZUL);
    shaderList.push_back(*shader3);

    Shader* shader4 = new Shader(); //shader CAFE
    shader4->CreateFromFiles(vShaderCAFE, fShaderCAFE);
    shaderList.push_back(*shader4);

    Shader* shader5 = new Shader(); //shader ROJO
    shader5->CreateFromFiles(vShaderROJO, fShaderROJO);
    shaderList.push_back(*shader5);

    Shader* shader6 = new Shader(); //shader VERDE_CLARO
    shader6->CreateFromFiles(vShaderVERDE_CLARO, fShaderVERDE_CLARO);
    shaderList.push_back(*shader6);

    Shader* shader7 = new Shader(); //shader VERDE_OSCURO
    shader7->CreateFromFiles(vShaderVERDE_OSCURO, fShaderVERDE_OSCURO);
    shaderList.push_back(*shader7);
}
```

Por último, tomando como referencia el código trabajado en clase, se realizó la visualización en pantalla del dibujo de la casa, sustituyendo las figuras planas por representaciones en 3D.

```
        //3D
        shaderList[4].useShader();
        uniformModel = shaderList[4].getModelLocation();
        uniformProjection = shaderList[4].getProjectLocation();

        //Pared Casa (Cubo Rojo)
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(0.0f, -0.58f, -2.0f));
        model = glm::scale(model, glm::vec3(1.0f, -1.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
        meshList[1]->RenderMesh();

        shaderList[5].useShader();
        uniformModel = shaderList[5].getModelLocation();
        uniformProjection = shaderList[5].getProjectLocation();
        //Ventana Izquierda (Cubo Verde)
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(-0.25f, -0.3f, -2.0f));
        model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
        meshList[1]->RenderMesh();

        shaderList[5].useShader();
        uniformModel = shaderList[5].getModelLocation();
        uniformProjection = shaderList[5].getProjectLocation();
        //Ventana Derecha (Cubo Verde)
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(0.25f, -0.3f, -2.0f));
        model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
        meshList[1]->RenderMesh();

        shaderList[5].useShader();
        uniformModel = shaderList[5].getModelLocation();
        uniformProjection = shaderList[5].getProjectLocation();
        //Puerta Casa (Cubo Verde)
        model = glm::mat4(1.0);
```

```
Project1                    (Ámbito global)                    main()
491         shaderList[5].useShader();
492         uniformModel = shaderList[5].getModelLocation();
493         uniformProjection = shaderList[5].getProjectLocation();
494         //Puerta Casa (Cubo Verde)
495         model = glm::mat4(1.0);
496         model = glm::translate(model, glm::vec3(0.0f, -0.85f, -2.0f));
497         model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
498         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
499         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
500         meshList[1]->RenderMesh();
501
502         shaderList[2].useShader();
503         uniformModel = shaderList[2].getModelLocation();
504         uniformProjection = shaderList[2].getProjectLocation();
505         //Techo (Piramide Azul)
506         model = glm::mat4(1.0);
507         model = glm::translate(model, glm::vec3(0.0f, 0.15f, -1.9f));
508         model = glm::scale(model, glm::vec3(1.0f, 0.5f, 1.0f));
509         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
510         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
511         meshList[0]->RenderMesh();
512
513         shaderList[3].useShader();
514         uniformModel = shaderList[3].getModelLocation();
515         uniformProjection = shaderList[3].getProjectLocation();
516         //Tronco Izquierdo (Cubo Café)
517         model = glm::mat4(1.0);
518         model = glm::translate(model, glm::vec3(-0.75f, -0.9f, -2.0f));
519         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
520         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
521         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
522         meshList[1]->RenderMesh();
523
524         shaderList[3].useShader();
525         uniformModel = shaderList[3].getModelLocation();
526         uniformProjection = shaderList[3].getProjectLocation();
527         //Tronco Derecho (Cubo Café)
528         model = glm::mat4(1.0);
529         model = glm::translate(model, glm::vec3(0.75f, -0.9f, -2.0f));
530         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
531         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
532         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
533         meshList[1]->RenderMesh();
534
535         shaderList[6].useShader();
536         uniformModel = shaderList[6].getModelLocation();
537         uniformProjection = shaderList[6].getProjectLocation();
538         //Hojas Pino Izquierdo (Pramide Verde Oscuro)
539         model = glm::mat4(1.0);
540         model = glm::translate(model, glm::vec3(-0.75f, -0.6f, -1.9f));
541         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.5f));
542         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
543         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
544         meshList[0]->RenderMesh();
545
546         shaderList[6].useShader();
547         uniformModel = shaderList[6].getModelLocation();
548         uniformProjection = shaderList[6].getProjectLocation();
549         //Hojas Pino Derecho (Piramide Verde Oscuro)
550         model = glm::mat4(1.0);
551         model = glm::translate(model, glm::vec3(0.75f, -0.6f, -1.9f));
552         model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.5f));
553         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
554         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
555         meshList[0]->RenderMesh();
556         //*/
557
558         glUseProgram(0);
559         mainWindow.swapBuffers();
560
561     }
562     return 0;
```

Resultado:

2. PROBLEMAS PRESENTADOS. LISTAR SI SURGIERON PROBLEMAS A LA HORA DE EJECUTAR EL CÓDIGO

Durante el desarrollo surgieron algunos inconvenientes, tales como:

- Errores de compilación al no enlazar correctamente algunos shaders.
- Problemas al olvidar reinicializar la matriz modelo antes de cada transformación.
- Dificultades al escalar la pirámide para que coincidiera correctamente con el tamaño del cubo base.

Todos los problemas fueron solucionados revisando el orden correcto de transformaciones y asegurando que cada objeto tuviera su propio shader activo antes de renderizarse

3. CONCLUSIÓN:

A. LOS EJERCICIOS DE LA CLASE: COMPLEJIDAD, EXPLICACIÓN

B. COMENTARIOS GENERALES: FALTÓ EXPLICAR A DETALLE, IR MÁS LENTO EN ALGUNA EXPLICACIÓN, OTROS COMENTARIOS Y SUGERENCIAS.

A. Sobre los ejercicios

La práctica presentó un nivel de complejidad mayor respecto a la anterior, ya que implicó trabajar con múltiples shaders, matrices de transformación y proyecciones. Se reforzó el entendimiento del pipeline gráfico moderno y el uso de transformaciones homogéneas.

B. Comentarios generales

La explicación fue adecuada, aunque algunos conceptos como el orden de multiplicación de matrices podrían explicarse con mayor detalle. Sería útil profundizar más en el manejo del depth buffer y en la diferencia visual entre proyección ortogonal y perspectiva.

Conclusión:

Al concluir esta práctica, noto que los cambios realizados hasta el momento no han sido especialmente difíciles. No obstante, sí resulta algo laborioso que para lograr un solo resultado sea necesario escribir una gran cantidad de líneas de código. Por ejemplo, en la construcción de la casa con figuras en 3D se requirieron varios bloques previamente definidos, además de agregar las instrucciones correspondientes para aplicar los shaders a cada objeto. Pienso que, conforme adquiera más experiencia, podré desarrollar este tipo de actividades con mayor facilidad y eficiencia.

Fuentes de consulta:

- Angel, E., & Shreiner, D. (2015). *Gráficos por computadora con OpenGL* (7.ª ed.). Pearson Educación.
- Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2014). *Guía oficial de programación de OpenGL, versión 4.3* (8.ª ed.). Addison-Wesley Professional.