



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 N° 2

NOMBRE COMPLETO: Gómez Enríquez Agustín

Nº de Cuenta: 317031405

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 5

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 31 de agosto del 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.

ACTIVIDAD 1

```
33     void agregarCuadradoRGB(GLfloat*& vertices, int& numFloats, float x, float y, float r, float g, float b, float tam) {
34         const int floatsPorVert = 6; // x y z r g b
35         const int nuevosVerts = 6; // 2 triángulos
36         const int totalNuevos = nuevosVerts * floatsPorVert;
37
38         GLfloat* temp = new GLfloat[numFloats + totalNuevos];
39         for (int i = 0; i < numFloats; i++) temp[i] = vertices ? vertices[i] : 0.0f;
40
41         float m = tam / 2.0f;
42         float x0 = x - m, x1 = x + m;
43         float y0 = y - m, y1 = y + m;
44         float z = 0.0f;
45         int idx = numFloats;
46
47         // tri 1
48         temp[idx++] = x0; temp[idx++] = y0; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
49         temp[idx++] = x1; temp[idx++] = y0; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
50         temp[idx++] = x1; temp[idx++] = y1; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
51         // tri 2
52         temp[idx++] = x0; temp[idx++] = y0; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
53         temp[idx++] = x1; temp[idx++] = y1; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
54         temp[idx++] = x0; temp[idx++] = y1; temp[idx++] = z; temp[idx++] = r; temp[idx++] = g; temp[idx++] = b;
55
56         if (vertices) delete[] vertices;
57         vertices = temp;
58         numFloats += totalNuevos;
59     }
```

A diferencia de la primera práctica, esta vez vamos a trabajar la construcción de las letras por medio de cuadrados de color, donde realmente se están usando dos triángulos unidos por su lado mas largo. Formando un cuadrado que vamos a ir replicando hasta formar nuestras iniciales “AGE”

```
77     // ----- letras A G E con MeshColor -----
78     void CrearInicialesAGE() {
79         GLfloat* V = nullptr; int N = 0;
80         const float tam = 0.07f; // más grande
81         const float oy = -0.15f;
82         const float oxA = -0.86f, oxE = -0.20f, oxE = 0.40f;
83         const float Ar = 1.0f, Ag = 0.2f, Ab = 0.2f;
84         const float Gr = 0.2f, Gg = 1.0f, Gb = 0.2f;
85         const float Er = 0.2f, Eg = 0.5f, Eb = 1.0f;
86
87         // A: columnas + barra superior + barra media (rejilla 7x9)
88         for (int y = 0; y <= 8; ++y) { addBlock(V, N, oxA, oy, tam, 0, y, Ar, Ag, Ab); addBlock(V, N, oxE, oy, tam, 6, y, Ar, Ag, Ab); }
89         for (int x = 0; x <= 6; ++x) { addBlock(V, N, oxA, oy, tam, x, 8, Ar, Ag, Ab); }
90         for (int x = 1; x <= 5; ++x) { addBlock(V, N, oxE, oy, tam, x, 4, Ar, Ag, Ab); }
91
92         // G: marco C + diente
93         for (int x = 0; x <= 6; ++x) { addBlock(V, N, oxG, oy, tam, x, 8, Gr, Gg, Gb); addBlock(V, N, oxG, oy, tam, x, 0, Gr, Gg, Gb); }
94         for (int y = 0; y <= 8; ++y) { addBlock(V, N, oxG, oy, tam, 0, y, Gr, Gg, Gb); }
95         for (int y = 0; y <= 4; ++y) { addBlock(V, N, oxG, oy, tam, 6, y, Gr, Gg, Gb); }
96         for (int x = 3; x <= 6; ++x) { addBlock(V, N, oxG, oy, tam, x, 4, Gr, Gg, Gb); }
97
98         // E: columna izq + barras
99         for (int y = 0; y <= 8; ++y) { addBlock(V, N, oxE, oy, tam, 0, y, Er, Eg, Eb); }
100        for (int x = 0; x <= 6; ++x) { addBlock(V, N, oxE, oy, tam, x, 8, Er, Eg, Eb); addBlock(V, N, oxE, oy, tam, x, 0, Er, Eg, Eb); }
101        for (int x = 0; x <= 4; ++x) { addBlock(V, N, oxE, oy, tam, x, 4, Er, Eg, Eb); }
102
103        MeshColor* letras = new MeshColor();
104        letras->CreateMeshColor(V, N); // pos+color intercalados (6 floats/vert) :contentReference[0a0c1e:2]{index=2}
105        meshColorList.push_back(letras);
106        if (V) { delete[] V; V = nullptr; }
107    }
```

Creamos las iniciales asignando un color por medio del mesh color, ya que se estará trabajando por columnas vamos a establecer la estructura de cada letra por medio de addBlock, con esto aseguramos su color y su posición en la ventana al hacer ejecución.

```

139     // ---- LETRAS ----
140     shaderList[SH LETTERS].useShader();
141     {
142         GLuint uModel = shaderList[SH LETTERS].getModelLocation();
143         GLuint uProj = shaderList[SH LETTERS].getProjectLocation();
144         glm::mat4 M(1.0f);
145         glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(M));
146         glUniformMatrix4fv(uProj, 1, GL_FALSE, glm::value_ptr(projection));
147         meshColorList[0]->RenderMeshColor(); // usa aPos(0) y aColor(1) :contentReference[oaicite:3]{index=3}

```

Dentro del main y con el uso de los shaders mandamos a llamar nuestras letras para visualizarlas en grande, centradas y con un color diferente cada una.

EJECUCION



ACTIVIDAD 2

```

int main() {
    mainWindow = Window(800, 600);
    mainWindow.Initialise();
    glEnable(GL_DEPTH_TEST);

    CreaPiramide();      // meshList[0]
    CrearCubo();         // meshList[1]
    CrearInicialesAGE(); // meshColorList[0]
    CreateShaders();
}

```

Para la segunda actividad debemos realizar una casa como la que se trabajó en clase. Para esto hacemos uso de nuestras bases que son la pirámide y el cubo, que declaramos en nuestro main.

```

// ---- CASA (mismo shader, color uniforme con 'objectColor') ----
shaderList[SH_GEOM].useShader();
{
    GLuint uModel = shaderList[SH_GEOM].getModelLocation();
    GLuint uProj = shaderList[SH_GEOM].getProjectLocation();
    GLuint uCol = shaderList[SH_GEOM].getUniformLocation(); // "objectColor" en shader.frag
    glUniformMatrix4fv(uProj, 1, GL_FALSE, glm::value_ptr(projection));

    // Paleta similar a tu referencia
    const glm::vec3 COLOR_RED_BODY = { 0.93f, 0.18f, 0.18f }; // muro
    const glm::vec3 COLOR_BLUE_ROOF = { 0.00f, 0.25f, 1.00f }; // techo azul intenso
    const glm::vec3 COLOR_LIME = { 0.61f, 0.98f, 0.28f }; // ventanas y puerta (verde-lima)
    const glm::vec3 COLOR_DARK_GREEN = { 0.10f, 0.45f, 0.10f }; // copas
    const glm::vec3 COLOR_BROWN = { 0.35f, 0.27f, 0.17f }; // troncos
    const glm::vec3 COLOR_GRASS = { 0.00f, 0.70f, 0.15f }; // césped

    glm::mat4 M;

    // CÉSPED (fino y ancho, con margen)
    M = glm::mat4(1.0f);
    M = glm::translate(M, glm::vec3(0.0f, -0.92f, -3.1f));
    M = glm::scale(M, glm::vec3(2.0f, 0.06f, 0.6f));
    glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(M));
    glUniform3fv(uCol, 1, &COLOR_GRASS[0]);
    meshList[1]->RenderMesh();
}

```

Vamos a crear nuestra paleta de referencia para cada elemento de la casa que es la pared (rectángulo), el techo (pirámide), ventanas y puerta que igualmente son rectángulos y nuestros arboles y césped. Aquí mismo hacemos la asignación de colores que vamos a trabajar por RGB.

shader.frag	Mesh.h	Mesh.cpp	Shader.h	Shader.cpp	practica2.cpp
1 #version 430 core 2 out vec4 FragColor; 3 uniform vec3 objectColor; 4 void main() { FragColor = vec4(objectColor, 1.0); } 5					

Nota: hice una modificación en el shader frag para poder trabajar los colores diferentes a los que estaban por defecto.

```

177      // MURO (más compacto)
178      // centro y=-0.20, alto=0.90 ? top ? 0.25 (deja espacio al techo)
179      M = glm::mat4(1.0f);
180      M = glm::translate(M, glm::vec3(0.0f, -0.20f, -3.0f));
181      M = glm::scale(M, glm::vec3(1.10f, 0.90f, 0.12f));
182      glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(M));
183      glUniform3fv(uCol, 1, &COLOR_RED_BODY[0]);
184      meshList[1]->RenderMesh();

185      // TECHO (pirámide) ajustado
186      // base = top del muro (? 0.25); scale_y=0.60 ? apex ? 0.25 + 0.25
187      M = glm::mat4(1.0f);
188      M = glm::translate(M, glm::vec3(0.0f, 0.55f, -2.98f)); // un poco más alto
189      M = glm::scale(M, glm::vec3(1.40f, 0.60f, 0.20f));
190      glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(M));
191      glUniform3fv(uCol, 1, &COLOR_BLUE_ROOF[0]);
192      meshList[0]->RenderMesh();

193      // PUERTA (centrada y más pequeña)
194      M = glm::mat4(1.0f);
195      M = glm::translate(M, glm::vec3(0.0f, -0.55f, -2.95f));
196      M = glm::scale(M, glm::vec3(0.28f, 0.32f, 0.06f));
197      glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(M));
198      glUniform3fv(uCol, 1, &COLOR_LIME[0]);
199      meshList[1]->RenderMesh();
200
201

```

Básicamente vamos a generar cada elemento con nuestra variable M, cargar la matriz de la figura que se estará usando y vamos a trabajar con dos elementos que es la posición con translate, la escala con scale y debemos recordar de renderizar cada uno con render mesh.

```

203     // VENTANAS (simétricas)
204     auto drawWindow = [&](float x) {
205         glm::mat4 W(1.0f);
206         W = glm::translate(W, glm::vec3(x, 0.05f, -2.95f));
207         W = glm::scale(W, glm::vec3(0.22f, 0.22f, 0.06f));
208         glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(W));
209         glUniform3fv(uCol, 1, &COLOR_LIME[0]);
210         meshList[1]->RenderMesh();
211     };
212     drawWindow(-0.35f);
213     drawWindow(0.35f);
214
215     // ÁRBOLES (compactos y dentro del marco)
216     auto drawTree = [&](float side) {
217         // Tronco
218         glm::mat4 T(1.0f);
219         T = glm::translate(T, glm::vec3(0.90f * side, -0.78f, -2.95f));
220         T = glm::scale(T, glm::vec3(0.12f, 0.18f, 0.06f));
221         glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(T));
222         glUniform3fv(uCol, 1, &COLOR_BROWN[0]);
223         meshList[1]->RenderMesh();
224
225         // Copia (base justo encima del tronco)
226         glm::mat4 C(1.0f);
227         C = glm::translate(C, glm::vec3(0.90f * side, -0.42f, -3.0f));
228         C = glm::scale(C, glm::vec3(0.45f, 0.48f, 0.20f));
229         glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(C));
230         glUniform3fv(uCol, 1, &COLOR_DARK_GREEN[0]);
231         meshList[0]->RenderMesh();
232     };
233     drawTree(-1.0f); // izq
234     drawTree(1.0f); // der
235 }
```

Para la última parte de los árboles, podemos trabajar como una variable de modelo, para que solo se defina sus valores de color, escala y posición y fuera del auto draw, podemos instanciar en los costados los arboles y que no bloquen la vista de la casa.

EJECUCION



2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

Ya que realice la practica posteriormente y no tenía noción de los comandos, me base en la practica 3 que nos da una imagen de referencia de la casa a trabajar. así como el como trabajar los glm para los tamaños. Porque al inicio la casa abarcaba toda la ventana de ejecución. Y tenia la duda de porque no se visualizaba la puerta hasta que note que se trabaja por jerarquía de elementos según su orden dentro del loop.

3.- Conclusión:

Fue una práctica completa de realizar, pero me sirvió para conocer de cierta forma el como se trabajan las primitivas y el uso de triángulos, de esta manera me fue fácil crear mis iniciales y solo hacer uso de shaders para cambio de colores, así como para el diseño de la casa.

Bibliografía en formato APA

Swiftless. (2010, March 25). 7. OpenGL Rotation and Translation (Version 2.0) – Swiftless Tutorials - Various older tutorials and newer programming musings. Swiftless.com.
<https://www.swiftless.com/tutorials/opengl/rotation.html>

Tutorial 3: Matrices. (n.d.). Opengl-tutorial.org, from <https://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/>

(N.d.). Umich.Mx, from
https://lc.fie.umich.mx/~rochoa/Materias/LABORATORIOS/GRAFICACION/GRAF_02.pdf

Interactivas y Computación Gráfica, T. [@ArturoVMS]. (n.d.). Proyecciones, Transformaciones y Shaders en OpenGL [Video]. Youtube, from
<https://www.youtube.com/watch?v=2hnqyXRpURQ>