
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** 02

**NOMBRE COMPLETO:** Medrano Miranda Daniel Ulises

**Nº de Cuenta:** 318045351

**GRUPO DE LABORATORIO:** 03

**GRUPO DE TEORÍA:** 04

**SEMESTRE 2025-1**

**FECHA DE ENTREGA LÍMITE:** 24/Agosto/2024

**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.***

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

Para la realización de este ejercicio se tomaron las coordenadas de las letras del ejercicio anterior y únicamente se agregaron los colores, los elegidos fueron azul para la “D”, amarillo para la “U” y cian para la “M”.

```
106   void CrearLetrasyFiguras()
107   {
108       GLfloat vertices_letras[] = {
109               //X       Y      Z              R       G       B
110               //    ------ LETRA D ------
111               -0.6f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
112               -0.6f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
113               -0.5f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
114               -0.6f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
115               -0.5f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
116               -0.5f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
117               -0.5f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
118               -0.5f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
119               -0.4f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
120               -0.5f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
121               -0.4f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
122               -0.4f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
123               -0.4f,  -0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
124               -0.4f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
125               -0.3f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
126               -0.4f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
127               -0.4f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
128               -0.3f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
129               -0.4f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
130               -0.3f,  -0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
131               -0.3f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
132               -0.3f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
133               -0.4f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
134               -0.4f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
135               -0.5f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
136               -0.4f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
137               -0.4f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
138               -0.5f,   0.1f,  0.0f,          0.0f,   0.0f,   1.0f,
139               -0.5f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
140               -0.4f,   0.2f,  0.0f,          0.0f,   0.0f,   1.0f,
141
```

```
142             //    ------ LETRA U ------
143             -0.2f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
144             -0.1f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
145             -0.2f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
146             -0.2f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
147             -0.1f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
148             -0.1f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
149             -0.1f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
150             -0.1f,  -0.1f,  0.0f,          1.0f,   1.0f,   0.0f,
151              0.0f,  -0.1f,  0.0f,          1.0f,   1.0f,   0.0f,
152             -0.1f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
153              0.0f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
154              0.0f,  -0.1f,  0.0f,          1.0f,   1.0f,   0.0f,
155              0.0f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
156              0.0f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
157              0.1f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
158              0.0f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
159              0.1f,  -0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
160              0.1f,   0.2f,  0.0f,          1.0f,   1.0f,   0.0f,
161
162              //    ------ LETRA M ------
163              0.2f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
164              0.2f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
165              0.3f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
166              0.2f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
167              0.3f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
168              0.3f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
169              0.3f,   0.05f, 0.0f,          0.0f,   1.0f,   1.0f,
170              0.3f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
171              0.4f,   0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
172              0.3f,   0.05f, 0.0f,          0.0f,   1.0f,   1.0f,
173              0.4f,  -0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
174              0.4f,   0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
175              0.4f,   0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
176              0.4f,  -0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
177              0.5f,   0.05f, 0.0f,          0.0f,   1.0f,   1.0f,
178              0.4f,   0.1f,  0.0f,          0.0f,   1.0f,   1.0f,
179              0.5f,   0.05f, 0.0f,          0.0f,   1.0f,   1.0f,
180              0.5f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
181              0.5f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
182              0.5f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
183              0.6f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
184              0.5f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
185              0.6f,  -0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
186              0.6f,   0.2f,  0.0f,          0.0f,   1.0f,   1.0f,
187          };
188          MeshColor *letras = new MeshColor();
189          letras->CreateMeshColor(vertices_letras,432);
190          meshColorList.push_back(letras);
```

Después se utilizaron las instrucciones para la impresión en pantalla

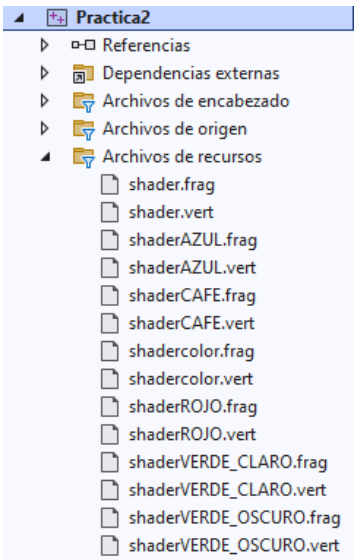
```
449     ///*               - - - EJERCICIOS DE LA PRÁCTICA 2 - - -
450     //Para las letras hay que usar el segundo set de shaders con índice 1 en ShaderList
451     shaderList[1].useShader();
452     uniformModel = shaderList[1].getModelLocation();
453     uniformProjection = shaderList[1].getProjectLocation();
454
455     //Inicializar matriz de dimensión 4x4 que servirá como matriz de modelo para almacenar las t
456     model = glm::mat4(1.0);
457     model = glm::translate(model, glm::vec3(-0.4f, 0.5f, -1.7f));
458     model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.0f));
459     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO
460     glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
461     meshColorList[0]->RenderMeshColor();
462
```

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

Para la realización de este ejercicio fue necesario primero elaborar los documentos ".vert" y ".frag" de los diferentes colores, una vez realizados se subieron al apartado de "archivos de recursos" del proyecto.

```
1   #version 330
2   layout (location =0) in vec3 pos;
3   out vec4 vColor;
4   uniform mat4 model;
5   uniform mat4 projection;
6   void main()
7   {
8       gl_Position=projection*model*vec4(pos,1.0f);
9       //vColor=vec4(color,1.0f);
10      vColor=vec4(0.0f,0.0f,1.0f,1.0f);
11  }
```

Ejemplo del archivo shaderAZUL.vert



Una vez se tuvieron los archivos se crearon los shaders y se aregaron a la lista de shaders para poder utilizarlos

```
275  void CreateShaders()
276  {
277
278      Shader *shader1 = new Shader(); //shader para usar índices: objetos: cubo y  pirámide
279      shader1->CreateFromFiles(vShader, fShader);
280      shaderList.push_back(*shader1);
281
282      Shader *shader2 = new Shader();//shader para usar color como parte del VAO: letras
283      shader2->CreateFromFiles(vShaderColor, fShaderColor);
284      shaderList.push_back(*shader2);
285
286      Shader* shader3 = new Shader(); //shader AZUL
287      shader3->CreateFromFiles(vShaderAZUL, fShaderAZUL);
288      shaderList.push_back(*shader3);
289
290      Shader* shader4 = new Shader(); //shader CAFE
291      shader4->CreateFromFiles(vShaderCAFE, fShaderCAFE);
292      shaderList.push_back(*shader4);
293
294      Shader* shader5 = new Shader(); //shader ROJO
295      shader5->CreateFromFiles(vShaderROJO, fShaderROJO);
296      shaderList.push_back(*shader5);
297
298      Shader* shader6 = new Shader(); //shader VERDE_CLARO
299      shader6->CreateFromFiles(vShaderVERDE_CLARO, fShaderVERDE_CLARO);
300      shaderList.push_back(*shader6);
301
302      Shader* shader7 = new Shader(); //shader VERDE_OSCURO
303      shader7->CreateFromFiles(vShaderVERDE_OSCURO, fShaderVERDE_OSCURO);
304      shaderList.push_back(*shader7);
305  }
```

Y finalmente, con base en el código de los ejercicios de clase se procedió a imprimir en pantalla el dibujo de la casa pero en lugar de usar formas planas usando las figuras 3D.

```
463          //Para las figuras 3D usar el set de shaders con índice 2:AZUL, 3:CAFE, 4:ROJO, 5:VERDE_CLARO, 6:VERDE_OSCURO
464          shaderList[4].useShader();
465          uniformModel = shaderList[4].getModelLocation();
466          uniformProjection = shaderList[4].getProjectLocation();
467
468          //Pared Casa (Cubo Rojo)
469          model = glm::mat4(1.0);
470          model = glm::translate(model, glm::vec3(0.0f, -0.58f, -2.0f));
471          model = glm::scale(model, glm::vec3(1.0f, -1.0f, 0.0f));
472          glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
473          glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
474          meshList[1]->RenderMesh();
475
476          shaderList[5].useShader();
477          uniformModel = shaderList[5].getModelLocation();
478          uniformProjection = shaderList[5].getProjectLocation();
479          //Ventana Izquierda (Cubo Verde)
480          model = glm::mat4(1.0);
481          model = glm::translate(model, glm::vec3(-0.25f, -0.3f, -2.0f));
482          model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
483          glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
484          glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
485          meshList[1]->RenderMesh();
486
```

```
487             shaderList[5].useShader();
488             uniformModel = shaderList[5].getModelLocation();
489             uniformProjection = shaderList[5].getProjectLocation();
490             //Ventana Derecha (Cubo Verde)
491             model = glm::mat4(1.0);
492             model = glm::translate(model, glm::vec3(0.25f, -0.3f, -2.0f));
493             model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
494             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
495             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
496             meshList[1]->RenderMesh();
497
498             shaderList[5].useShader();
499             uniformModel = shaderList[5].getModelLocation();
500             uniformProjection = shaderList[5].getProjectLocation();
501             //Puerta Casa (Cubo Verde)
502             model = glm::mat4(1.0);
503             model = glm::translate(model, glm::vec3(0.0f, -0.85f, -2.0f));
504             model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
505             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
506             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
507             meshList[1]->RenderMesh();
508
509             shaderList[2].useShader();
510             uniformModel = shaderList[2].getModelLocation();
511             uniformProjection = shaderList[2].getProjectLocation();
512             //Techo (Piramide Azul)
513             model = glm::mat4(1.0);
514             model = glm::translate(model, glm::vec3(0.0f, 0.15f, -1.9f));
515             model = glm::scale(model, glm::vec3(1.0f, 0.5f, 1.0f));
516             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
517             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
518             meshList[0]->RenderMesh();
519
520             shaderList[3].useShader();
521             uniformModel = shaderList[3].getModelLocation();
522             uniformProjection = shaderList[3].getProjectLocation();
523             //Tronco Izquierdo (Cubo Café)
524             model = glm::mat4(1.0);
525             model = glm::translate(model, glm::vec3(-0.75f, -0.9f, -2.0f));
526             model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
527             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
528             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
529             meshList[1]->RenderMesh();
530
531             shaderList[3].useShader();
532             uniformModel = shaderList[3].getModelLocation();
533             uniformProjection = shaderList[3].getProjectLocation();
534             //Tronco Derecho (Cubo Café)
535             model = glm::mat4(1.0);
536             model = glm::translate(model, glm::vec3(0.75f, -0.9f, -2.0f));
537             model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
538             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
539             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
540             meshList[1]->RenderMesh();
541
542             shaderList[6].useShader();
543             uniformModel = shaderList[6].getModelLocation();
544             uniformProjection = shaderList[6].getProjectLocation();
545             //Hojas Pino Izquierdo (Pramide Verde Oscuro)
546             model = glm::mat4(1.0);
547             model = glm::translate(model, glm::vec3(-0.75f, -0.6f, -1.9f));
548             model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.5f));
549             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
550             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
551             meshList[0]->RenderMesh();
552
553             shaderList[6].useShader();
554             uniformModel = shaderList[6].getModelLocation();
555             uniformProjection = shaderList[6].getProjectLocation();
556             //Hojas Pino Derecho (Piramide Verde Oscuro)
557             model = glm::mat4(1.0);
558             model = glm::translate(model, glm::vec3(0.75f, -0.6f, -1.9f));
559             model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.5f));
560             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA y
561             glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
562             meshList[0]->RenderMesh();
```

Captura de pantalla de la ejecución:


***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***

Considero que la parte más fácil de esta práctica fue colocar las letras de colores, esto debido a que ya se tenían las coordenadas y únicamente se le añadió el color. Por otro lado, la parte de la creación de los shaders se me dificultó un poco debido a que no sabía si debía crear otro archivo o el mismo código lo crearía, después de analizar por un buen rato el código descubrí que únicamente lee los archivos así que procedí a crearlos, después intenté usarlos pero el código "crasheaba" debido a que no los estaba creando en la función createShaders, así que copié las funciones y le cambié los nombres a los de los archivos que cree.

***3.- Conclusión:***

***a. Los ejercicios del reporte: Complejidad, Explicación.***

En esta ocasión la complejidad ya fue mayor debido a la creación de los shaders pero considero que ahora puedo crear shaders de cualquier color sin ningún problema.

***b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica***

Me parece que la explicación del profesor en el laboratorio fue la adecuada, estuvimos viendo cómo es que funciona el código y qué es lo que debemos modificar.

### *c. Conclusión*

Al finalizar esta práctica me doy cuenta que, hasta ahora, las modificaciones que se deben de hacer no son tan complejas, sin embargo, si es un poco tedioso que para hacer algo se necesiten de muchas líneas de código, por ejemplo, para la creación de la casa con figuras 3D fueron varios bloques de código que ya se tenían más las líneas del uso de los shaders para cada figura. Considero que con más práctica podré realizar estas tareas de manera más sencilla.