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: Miranda González José Francisco

Nº de Cuenta: 318222327

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 24/08/24

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

Reporte de práctica 2: Proyecciones y puertos de vista. Transformaciones Geométricas

Instrucciones:

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

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.

Actividades realizadas

Actividad 1:

Para completar la actividad 1, realice lo siguiente:

En la función CrearLetrasyFiguras() coloqué los vértices de la Practica01 (los cuales construían las iniciales de mi nombre) y solo tuve que identificar a cuáles cambiarle el RGB para obtener las tres letras de diferentes colores.

```
106     //PARA LAS LETRAS
107     void CrearLetrasyFiguras()
108     {
109         // 0 meshColorList
110         // Iniciales de mi nombre: MGF
111         GLfloat vertices_letras[] = {
112
113                 //X     Y       Z       R       G       B
114
115                 -0.90f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
116                 -0.90f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
117                 -0.80f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
118
119                 -0.90f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
120                 -0.80f,0.50f, 0.0f,     1.0f,   0.0f,   0.0f,
121                 -0.80f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
122
123                 -0.80f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
124                 -0.80f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
125                 -0.55f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
```

```
127             -0.80f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
128             -0.55f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
129             -0.60f,-0.20f,0.0f,     1.0f,   0.0f,   0.0f,
130
131             -0.55f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
132             -0.60f,-0.20f,0.0f,     1.0f,   0.0f,   0.0f,
133             -0.50f,-0.20f,0.0f,     1.0f,   0.0f,   0.0f,
134
135             -0.55f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
136             -0.50f,-0.20f,0.0f,     1.0f,   0.0f,   0.0f,
137             -0.30f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
138
139             -0.55f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
140             -0.30f,0.10f,0.0f,      1.0f,   0.0f,   0.0f,
141             -0.30f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
142
143             -0.30f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
144             -0.30f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
145             -0.20f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
146
147             -0.20f,0.50f,0.0f,      1.0f,   0.0f,   0.0f,
148             -0.30f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
149             -0.20f,-0.50f,0.0f,     1.0f,   0.0f,   0.0f,
150
151             -0.10f,-0.50f,0.0f,     0.0f,   1.0f,   0.0f,
152             -0.10f,0.40f,0.0f,      0.0f,   1.0f,   0.0f,
153             0.0f,0.40f,0.0f,        0.0f,   1.0f,   0.0f,
154
155             0.0f,0.40f,0.0f,        0.0f,   1.0f,   0.0f,
156             -0.10f,-0.50f,0.0f,     0.0f,   1.0f,   0.0f,
157             0.0f,-0.50f,0.0f,       0.0f,   1.0f,   0.0f,
158
159             -0.10f,0.40f,0.0f,      0.0f,   1.0f,   0.0f,
160             -0.10f,0.50f,0.0f,      0.0f,   1.0f,   0.0f,
161             0.40f,0.50f,0.0f,       0.0f,   1.0f,   0.0f,
162
163             0.40f,0.50f,0.0f,       0.0f,   1.0f,   0.0f,
164             -0.10f,0.40f,0.0f,      0.0f,   1.0f,   0.0f,
165             0.40f,0.40f,0.0f,       0.0f,   1.0f,   0.0f,
```

```
167             0.0f,-0.50f,0.0f,       0.0f,   1.0f,   0.0f,
168             0.40f,-0.40f,0.0f,      0.0f,   1.0f,   0.0f,
169             0.0f,-0.40f,0.0f,       0.0f,   1.0f,   0.0f,
170
171             0.0f,-0.50f,0.0f,       0.0f,   1.0f,   0.0f,
172             0.40f,-0.40f,0.0f,      0.0f,   1.0f,   0.0f,
173             0.40f,-0.50f,0.0f,      0.0f,   1.0f,   0.0f,
```

```
175             0.40f,-0.40f,0.0f,      0.0f,   1.0f,   0.0f,
176             0.40f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
177             0.30f,-0.40f,0.0f,      0.0f,   1.0f,   0.0f,
178
179             0.40f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
180             0.30f,-0.40f,0.0f,      0.0f,   1.0f,   0.0f,
181             0.30f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
182
183             0.40f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
184             0.40f,0.0f,0.0f,        0.0f,   1.0f,   0.0f,
185             0.10f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
186
187             0.10f,-0.10f,0.0f,      0.0f,   1.0f,   0.0f,
188             0.40f,0.0f,0.0f,        0.0f,   1.0f,   0.0f,
189             0.10f,0.0f,0.0f,        0.0f,   1.0f,   0.0f,
190
191             0.50f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
192             0.50f,-0.50f,0.0f,      0.0f,   0.0f,   1.0f,
193             0.60f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
194
195             0.60f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
196             0.50f,-0.50f,0.0f,      0.0f,   0.0f,   1.0f,
197             0.60f,-0.50f,0.0f,      0.0f,   0.0f,   1.0f,
198
199             0.60f,0.0f,0.0f,        0.0f,   0.0f,   1.0f,
200             0.70f,0.0f,0.0f,        0.0f,   0.0f,   1.0f,
201             0.70f,0.10f,0.0f,       0.0f,   0.0f,   1.0f,
202
203             0.70f,0.10f,0.0f,       0.0f,   0.0f,   1.0f,
204             0.60f,0.0f,0.0f,        0.0f,   0.0f,   1.0f,
205             0.60f,0.10f,0.0f,       0.0f,   0.0f,   1.0f,
206
207             0.60f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
208             0.90f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
209             0.60f,0.40f,0.0f,       0.0f,   0.0f,   1.0f,
210
211             0.60f,0.40f,0.0f,       0.0f,   0.0f,   1.0f,
212             0.90f,0.50f,0.0f,       0.0f,   0.0f,   1.0f,
213             0.90f,0.40f,0.0f,       0.0f,   0.0f,   1.0f,
214
215         };
216         MeshColor *letras = new MeshColor();
217         letras->CreateMeshColor(vertices_letras,450);
218         meshColorList.push_back(letras);
219
220     }
```

Además, cambie el valor de: letras->CreateMeshColor(vertices_letras,450);

Pues ahora se ocupaban más vértices (75 * 6 = 450).

```
217     letras->CreateMeshColor(vertices_letras,450);
```

Dentro de main(), ajuste el tamaño de la ventana para apreciar mejor las letras.

```
262     //DIFERENTE TAMAÑO DE VENTANA PARA LAS LETRAS
263     mainWindow = Window(800, 600);
```

En el while de main(), ocupé una parte del código visto en clase y solo tuve que modificar los índices y el eje Z en translate para que las letras se mostraran en la ventana de manera correcta.

```
294     //LO COMENTAMOS PARA NO MOSTAR LAS LETRAS
295     //Para las letras hay que usar el segundo set de shaders con índice 1 en ShaderList
296     shaderList[1].useShader();
297     uniformModel = shaderList[1].getModelLocation();
298     uniformProjection = shaderList[1].getProjectLocation();
299
300     // LO COMETAMOS PARA NO MOSTRAR LAS LETRAS
301     // Iniciales de mi nombre: MGF
302     model = glm::mat4(1.0);
303     model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
304     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
305     glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
306     meshColorList[0]->RenderMeshColor();
```

Ejecución del programa:

Actividad 2:

Para completar la actividad 2, realice lo siguiente:

Lo primero que hice fue crear los shaders con diferentes colores y agregarlos a la carpeta shaders:

Cubo Rojo:

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    //vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);
    vColor=vec4(1.0, 0.0, 0.0, 1.0); //Rojo Cubo
}
```

Pirámide Azul:

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    //vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);
    vColor=vec4(0.0, 0.0, 1.0, 1.0); // Azul Piramide
}
```

Pirámide Verde:

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    //vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);
    vColor=vec4(0.0, 0.5, 0.0, 1.0); // Verde Piramide
}
```

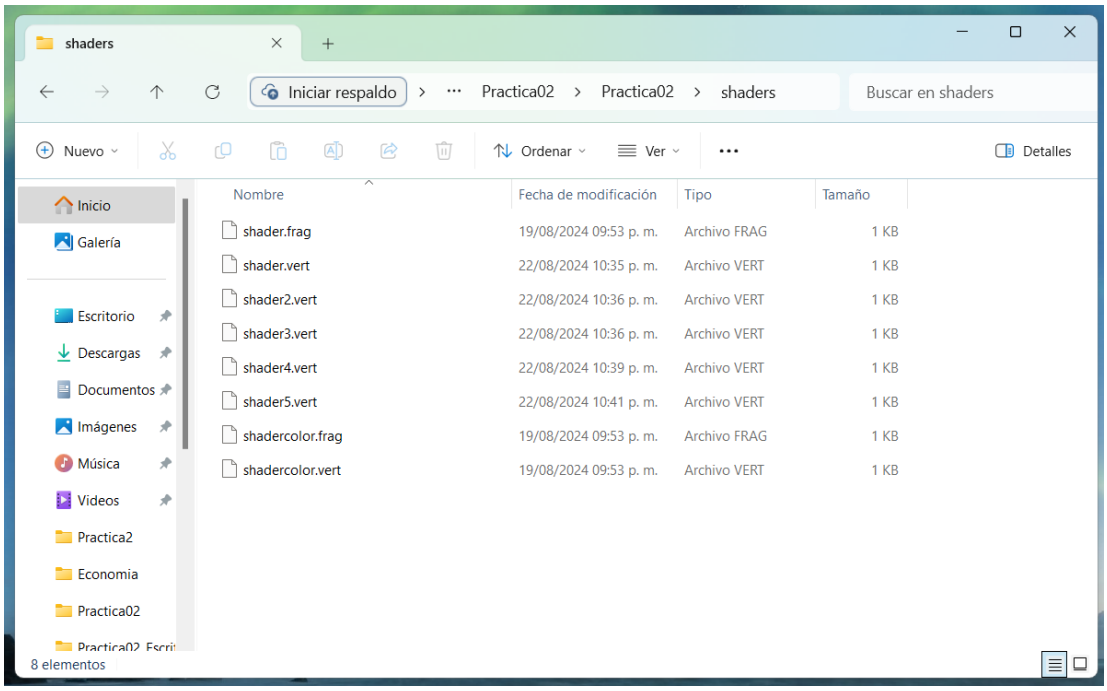
Cubo Verde:

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    //vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);
    vColor=vec4(0.0, 1.0, 0.0, 1.0); // Verde Cubo
}
```

Cubo Café:

```
#version 330
layout (location =0) in vec3 pos;
out vec4 vColor;
uniform mat4 model;
uniform mat4 projection;
void main()
{
    gl_Position=projection*model*vec4(pos,1.0f);
    //vColor=vec4(color,1.0f);
    //vColor=vec4(clamp(pos,0.0f,1.0f),1.0f);
    vColor=vec4(0.478, 0.255, 0.067, 1.0); // Café Cubo
}
```

Carpeta shaders:



Después, dentro del código definí las variables para almacenar las rutas a los shaders creados.

```
23      //Vertex Shader
24      static const char* vShader = "shaders/shader.vert";
25      //PARA LA CASA
26      static const char* vShader2 = "shaders/shader2.vert";
27      static const char* vShader3 = "shaders/shader3.vert";
28      static const char* vShader4 = "shaders/shader4.vert";
29      static const char* vShader5 = "shaders/shader5.vert";
```

En la función CreateShaders(), también tuve que colocar los shaders creados.

```
223 void CreateShaders()
224 {
225     // 0 shaderList
226     Shader *shader1 = new Shader(); //shader para usar índices: objetos: cubo y  pirámide
227     shader1->CreateFromFiles(vShader, fShader);
228     shaderList.push_back(*shader1);
229
230     // 1 shaderList
231     Shader *shader2 = new Shader();//shader para usar color como parte del VAO: letras
232     shader2->CreateFromFiles(vShaderColor, fShaderColor);
233     shaderList.push_back(*shader2);
234
235     //PARA LOS DEMAS COLORES DE LA CASA
236
237     // 2 shaderList
238     Shader* shader3 = new Shader();
239     shader3->CreateFromFiles(vShader2, fShader);
240     shaderList.push_back(*shader3);
241
242     // 3 shaderList
243     Shader* shader4 = new Shader();
244     shader4->CreateFromFiles(vShader4, fShader);
245     shaderList.push_back(*shader4);
246
247     // 4 shaderList
248     Shader* shader5 = new Shader();
249     shader5->CreateFromFiles(vShader5, fShader);
250     shaderList.push_back(*shader5);
251
252     // 5 shaderList
253     Shader* shader6 = new Shader();
254     shader6->CreateFromFiles(vShader3, fShader);
255     shaderList.push_back(*shader6);
256 }
```

Dentro de main(), ajuste el tamaño de la ventana ahora para la casa

```
265     //TAMAÑO DE VENTANA PARA LA CASA
266     mainWindow = Window(760, 760);
```

En el while de main(), al igual que en la actividad pasada, ocupé una parte del código visto en clase. A este solo le modifique los índices y los valores en translate y scale para mostrar el dibujo en la ventana de manera correcta.

```
// CUBO ROJO
//Para el cubo y la pirámide se usa el primer set de shaders con índice 0 en ShaderList
shaderList[0].useShader();
uniformModel = shaderList[0].getModelLocation();
uniformProjection = shaderList[0].getProjectLocation();
angulo += 0.01;
//Inicializar matriz de dimensión 4x4 que servirá como matriz de modelo para almacenar las transformaciones geométricas
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.3f, -4.0f));
model = glm::scale(model, glm::vec3(0.9f, 0.9f, 0.5f));
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));
meshList[1]->RenderMesh();
```

```
// PIRAMIDE AZUL
shaderList[2].useShader();
uniformModel = shaderList[2].getModelLocation();
uniformProjection = shaderList[2].getProjectLocation();
angulo += 0.01;
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.4f, -4.0f));
model = glm::scale(model, glm::vec3(1.0f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh();

// CUBO VERDE 1
shaderList[3].useShader();
uniformModel = shaderList[3].getModelLocation();
uniformProjection = shaderList[3].getProjectLocation();
angulo += 0.01;
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-0.2f, -0.05f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();

// CUBO VERDE 2
shaderList[3].useShader();
uniformModel = shaderList[3].getModelLocation();
uniformProjection = shaderList[3].getProjectLocation();
angulo += 0.01;
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.2f, -0.05f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();
```

```
// CUBO VERDE 3
shaderList[3].useShader();
uniformModel = shaderList[3].getModelLocation();
uniformProjection = shaderList[3].getProjectLocation();
angulo += 0.01;
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -0.6f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[1]->RenderMesh();
```

```
371         // CUBO CAFE 1
372         shaderList[4].useShader();
373         uniformModel = shaderList[4].getModelLocation();
374         uniformProjection = shaderList[4].getProjectLocation();
375         angulo += 0.01;
376         model = glm::mat4(1.0);
377         model = glm::translate(model, glm::vec3(0.75f, -0.65f, -2.0f));
378         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.5f));
379         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
380         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
381         meshList[1]->RenderMesh();
382
383         // CUBO CAFE 2
384         shaderList[4].useShader();
385         uniformModel = shaderList[4].getModelLocation();
386         uniformProjection = shaderList[4].getProjectLocation();
387         angulo += 0.01;
388         model = glm::mat4(1.0);
389         model = glm::translate(model, glm::vec3(-0.75f, -0.65f, -2.0f));
390         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.5f));
391         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
392         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
393         meshList[1]->RenderMesh();
394
395         // PIRAMIDE VERDE 1
396         shaderList[5].useShader();
397         uniformModel = shaderList[5].getModelLocation();
398         uniformProjection = shaderList[5].getProjectLocation();
399         angulo += 0.01;
400         model = glm::mat4(1.0);
401         model = glm::translate(model, glm::vec3(-0.75f, -0.35f, -2.0f));
402         model = glm::scale(model, glm::vec3(0.3f, 0.4f, 0.5f));
403         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
404         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
405         meshList[0]->RenderMesh();
```

```
407         // PIRAMIDE VERDE 2
408         shaderList[5].useShader();
409         uniformModel = shaderList[5].getModelLocation();
410         uniformProjection = shaderList[5].getProjectLocation();
411         angulo += 0.01;
412         model = glm::mat4(1.0);
413         model = glm::translate(model, glm::vec3(0.75f, -0.35f, -2.0f));
414         model = glm::scale(model, glm::vec3(0.3f, 0.4f, 0.5f));
415         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
416         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
417         meshList[0]->RenderMesh();
```

Ejecución del programa:

Problemas presentados

1. Translate:
   Al realizar la primera actividad y ejecutar el programa solo mostraba una pantalla negra sin ninguna letra.
   Investigando un poco comprendí que el error era el eje Z de translate, pues estaba definido como: model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
   Al estar en 0.0f no permitía ver las letras, así que solo cambie el valor a -4.0f.

2. Vértices:
   Al principio no me aparecían todos los triángulos en la ventana y esto era porque no cambie el valor en letras->CreateMeshColor(vertices_letras,18);
   Dejé el valor de 18 que era para el ejemplo, así que lo tuve que cambiar a 450 para mostrar todos los triángulos.

Conclusión

Realizar la primera actividad no fue muy difícil, solo era copiar los vértices que ya había creado en la práctica pasada y cambiar el RGB en cada letra. Los únicos "problemas" que presente fueron los siguientes:

Había olvidado cambiar el valor de los vértices y en un principio no se mostraban todos los triángulos.

```
letras->CreateMeshColor(vertices_letras,450);
```

Tenia el valor de Z en 0.0f y solo me aparecía una pantalla negra.

```
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
```

En el caso de la segunda actividad, lo que se me complico fue saber como crear los archivos .vert, ya que no sabía si basarme en shader.vert o shadercolor.vert.

Revisando el código me di cuenta de que shader.vert era el que se ocupaba para las pirámides y cubos, así que a partir de ese archivo cree los demás.

Después de eso ya no presente más problemas, pues solo con ir siguiendo el código me daba cuenta que es lo que tenía que modificar o agregar.

De el ejercicio de clase a esta práctica comprendí el funcionamiento de los índices para meshList, meshColorList y shaderList. Entonces ya sabía cuándo ocuparlos.

Al finalizar la practica pienso que las actividades solicitadas se cumplieron de manera correcta.

Bibliografía

Khronos Registry. (s.f). clamp - OpenGL 4 Reference Pages. Consultado el 24 de agosto del 2024 de https://registry.khronos.org/OpenGL-Refpages/gl4/html/clamp.xhtml