



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 N° 02

NOMBRE COMPLETO: Medrano Miranda Daniel Ulises

N° de Cuenta: 318045351

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 20/Agosto/2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa

1.- generar las figuras copiando los vértices de triangulorojo y cuadradoverde:

- triángulo azul
- triángulo verde (0,0.5,0)
- cuadrado rojo
- cuadrado verde
- cuadrado café (0.478, 0.255, 0.067)

```
124  GLfloat vertices_trianguloverde[] = {
125      //X      Y      Z      R      G      B
126      -1.0f,  -1.0f,   0.5f,   0.0f,  0.5f,  0.0f,
127      1.0f,   -1.0f,   0.5f,   0.0f,  0.5f,  0.0f,
128      0.0f,   1.0f,   0.5f,   0.0f,  0.5f,  0.0f,
129
130  };
131
132  MeshColor* trianguloverde = new MeshColor();
133  trianguloverde->CreateMeshColor(vertices_trianguloverde, 18);
134  meshColorList.push_back(trianguloverde);
135
136  GLfloat vertices_trianguloazul[] = {
137      //X      Y      Z      R      G      B
138      -1.0f,  -1.0f,   0.5f,   0.0f,  0.0f,  1.0f,
139      1.0f,   -1.0f,   0.5f,   0.0f,  0.0f,  1.0f,
140      0.0f,   1.0f,   0.5f,   0.0f,  0.0f,  1.0f,
141
142  };
143
144  MeshColor* trianguloazul = new MeshColor();
145  trianguloazul->CreateMeshColor(vertices_trianguloazul, 18);
146  meshColorList.push_back(trianguloazul);
147
148  GLfloat vertices_cuadradoverde[] = {
149      //X      Y      Z      R      G      B
150      -0.5f,  -0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
151      0.5f,   -0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
152      0.5f,   0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
153      -0.5f,  -0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
154      0.5f,   0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
155      -0.5f,  0.5f,   0.5f,   0.0f,  1.0f,  0.0f,
156
157  };
158
159  MeshColor* cuadradoverde = new MeshColor();
160  cuadradoverde->CreateMeshColor(vertices_cuadradoverde, 36);
161  meshColorList.push_back(cuadradoverde);
```

```

163 GLfloat vertices_cuadradorojo[] = {
164     //X      Y      Z      R      G      B
165     -0.5f,  -0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
166     0.5f,   -0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
167     0.5f,   0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
168     -0.5f,  -0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
169     0.5f,   0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
170     -0.5f,  0.5f,   0.5f,   1.0f,  0.0f,  0.0f,
171 };
172
173 MeshColor* cuadradorojo = new MeshColor();
174 cuadradorojo->CreateMeshColor(vertices_cuadradorojo, 36);
175 meshColorList.push_back(cuadradorojo);
176
177 GLfloat vertices_cuadradoafe[] = {
178     //X      Y      Z      R      G      B
179     -0.5f,  -0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
180     0.5f,   -0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
181     0.5f,   0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
182     -0.5f,  -0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
183     0.5f,   0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
184     -0.5f,  0.5f,   0.5f,   0.478f, 0.255f, 0.067f,
185 };
186
187 MeshColor* cuadradoafe = new MeshColor();
188 cuadradoafe->CreateMeshColor(vertices_cuadradoafe, 36);
189 meshColorList.push_back(cuadradoafe);
190
191

```

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

```

269 //EJERCICIOS DE CLASE (PROYECCION)
270 //Para las letras hay que usar el segundo set de shaders con índice 1 en ShaderList
271 shaderList[1].useShader();
272 uniformModel = shaderList[1].getModelLocation();
273 uniformProjection = shaderList[1].getProjectLocation();
274
275 //Pared Casa (Rojo)
276 model = glm::mat4(1.0);
277 model = glm::translate(model, glm::vec3(0.0f, -0.58f, -2.0f));
278 model = glm::scale(model, glm::vec3(1.0f, -1.0f, 0.0f));
279 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
280 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
281 meshColorList[5]->RenderMeshColor();
282
283 //Ventana Izquierda (Verde)
284 model = glm::mat4(1.0);
285 model = glm::translate(model, glm::vec3(-0.25f, -0.3f, -2.0f));
286 model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
287 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
288 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
289 meshColorList[4]->RenderMeshColor();
290
291 //Ventana Derecha (Verde)
292 model = glm::mat4(1.0);
293 model = glm::translate(model, glm::vec3(0.25f, -0.3f, -2.0f));
294 model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
295 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
296 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
297 meshColorList[4]->RenderMeshColor();

```

```

299 //Puerta Casa (Verde)
300 model = glm::mat4(1.0);
301 model = glm::translate(model, glm::vec3(0.0f, -0.85f, -2.0f));
302 model = glm::scale(model, glm::vec3(0.3f, -0.3f, 0.3f));
303 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
304 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
305 meshColorList[4] -> RenderMeshColor();
306
307 //Techo (Azul)
308 model = glm::mat4(1.0);
309 model = glm::translate(model, glm::vec3(0.0f, 0.2f, -2.0f));
310 model = glm::scale(model, glm::vec3(0.6f, 0.3f, 0.5f));
311 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
312 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
313 meshColorList[3] -> RenderMeshColor();
314
315 //Tronco Izquierdo (Café)
316 model = glm::mat4(1.0);
317 model = glm::translate(model, glm::vec3(-0.75f, -0.9f, -2.0f));
318 model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
319 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
320 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
321 meshColorList[6] -> RenderMeshColor();
322
323 //Tronco Derecho (Café)
324 model = glm::mat4(1.0);
325 model = glm::translate(model, glm::vec3(0.75f, -0.9f, -2.0f));
326 model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
327 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
328 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
329 meshColorList[6] -> RenderMeshColor();
330
331 //Hojas Pino Izquierdo (Verde)
332 model = glm::mat4(1.0);
333 model = glm::translate(model, glm::vec3(-0.75f, -0.6f, -2.0f));
334 model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
335 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
336 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
337 meshColorList[2] -> RenderMeshColor();
338
339 //Hojas Pino Derecho (Verde)
340 model = glm::mat4(1.0);
341 model = glm::translate(model, glm::vec3(0.75f, -0.6f, -2.0f));
342 model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.3f));
343 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO
344 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
345 meshColorList[2] -> RenderMeshColor();

```



2. Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código

A la hora de ejecutar el código no surgió ningún problema, lo más complejo se trató de realizar las traslaciones y escalamientos correctamente, en esta ocasión se tuvo que hacer “a prueba y error” para poder encontrar los valores en los que los ejes X, Y e Z se tenían que mover, sin embargo, considero que con más práctica se puede realizar de manera más fácil.

3. Conclusión:

a. Los ejercicios de la clase: Complejidad, explicación

La complejidad del ejercicio, en una escala del 1 al 5 yo la colocaría en 2, esto debido que se basó en copiar y pegar funciones, sin embargo, si fue de pensar un poco las coordenadas de las figuras para poder acomodarlas correctamente y que la imagen resultante fuera la misma que la pedida por el profesor.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.

Considero que nos faltó que se explicara en clase algún ejemplo de la función scale, esto porque pudimos observar correctamente cómo es que se trasladan las figuras pero no como se escalan, fuera de eso, la explicación en el laboratorio estuvo muy bien.