



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: Hernández Vázquez Daniela

N° de Cuenta: 318092867

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 02

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 20 de febrero de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

En esta ocasión durante la sesión fueron revisados los diferentes archivos del proyecto correspondientes a la practica 2. En este proyecto podemos observar como a diferencia de la practica 1 donde el VertexShader y el FragmentShader estaban embebidos en esta ocasión se presentan como archivos externos los cuales pueden ser invocados desde el programa principal. Lo primero que podemos notar es que en esta ocasión se tienen 2 tipos de shader, los archivos shader (tanto .frag como .vert) reciben como parámetros únicamente vértices, mientras que los ShaderColor reciben datos tanto de vértices como de color. Cada uno de estos se encarga de dar una forma específica.

Actividad 1:

Como primer ejercicio se solicitó generar las figuras copiando los vértices de triángulo rojo y cuadrado verde: triángulo azul, triángulo verde (0,0.5,0), cuadrado rojo, cuadrado verde, cuadrado café (0.478, 0.255, 0.067).

Para realizar esto, modificamos la sección de *CrearLetrasyFiguras*, con esto podremos instanciar más adelante las figuras que necesitemos, debemos de tomar en cuenta que el lugar donde creemos cada figura será el tipo de forma que se creará. Dado que ya tenemos 2 tipos de figuras formadas (triangulorojo y cuadradoverde) podemos simplemente copiar estas estructuras, cambiar el nombre y modificar los parámetros de color de cada vértice para crear trianguloazul, trianguloverde, cuadradorojo y cuadradocafe.

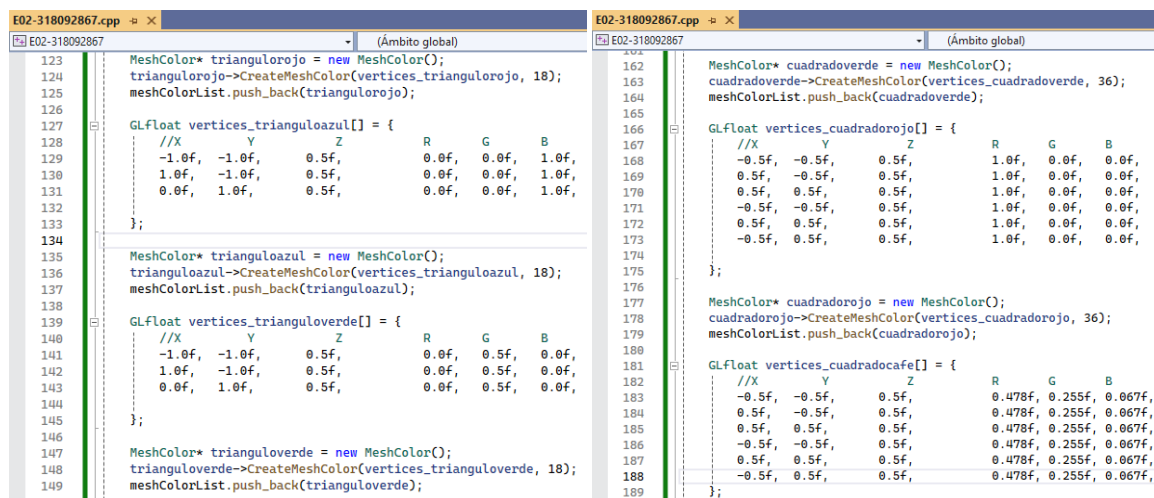


Figura 1. Variables agregadas

Procedemos a inicializar las matrices modelos, 9 en total, una para cada figura generada, y posteriormente procedemos a instanciar las figuras. Cuando instanciamos las figuras y en ellas se tiene solo una traslación en el eje z la figura

que esté más al frente será la que se vea, en este caso al instanciarlas la figura que se encuentra más adelante es la café.

```

246 //Inicializar matriz de dimensión 4x4 que servirá como matriz de modelo para almacenar
247 glm::mat4 model(1.0);
248 glm::mat4 model1(1.0);
249 glm::mat4 model2(1.0);
250 glm::mat4 model3(1.0);
251 glm::mat4 model4(1.0);
252 glm::mat4 model5(1.0);
253 glm::mat4 model6(1.0);
254 glm::mat4 model7(1.0);
255 glm::mat4 model8(1.0);
256
257 // Imprimimos los elementos, el orden importa
258 // como todos se crean con respecto al origen cada uno debe de trasladarse y escalarse

260 // Casa
261 //cuadrado rojo
262 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -6.0f));
263 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
264 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
265 meshColorList[5] -> RenderMeshColor();
266 //triángulo azul
267 model1 = glm::translate(model1, glm::vec3(0.0f, 0.0f, -5.0f));
268 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model1)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
269 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
270 meshColorList[2] -> RenderMeshColor();
271 //cuadrado verde
272 model2 = glm::translate(model2, glm::vec3(0.0f, 0.0f, -4.0f));
273 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model2)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
274 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
275 meshColorList[4] -> RenderMeshColor();

```

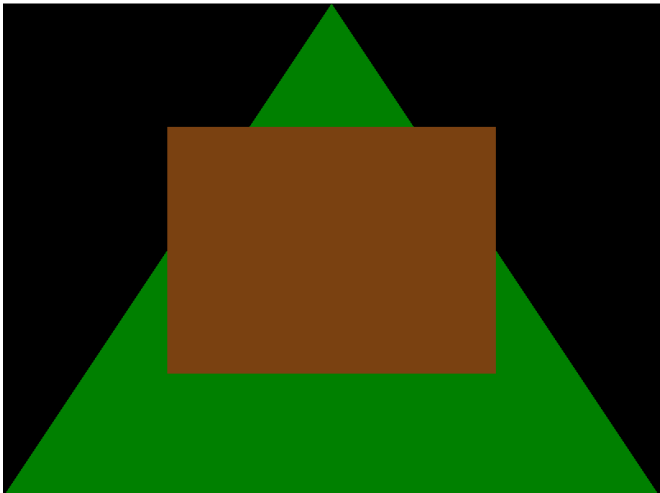


Figura 2. Figuras instanciadas

Actividad 2:

Como segunda actividad se utiliza la proyección ortogonal y se busca generar la figura de una casa con las formas anteriormente realizadas, recordando que todas las figuras se forman en el origen y con un tamaño determinado procedemos a aplicar los escalamientos y traslaciones necesarias siguiendo la imagen de referencia dada por el profesor.

Para esto lo que hice fue dividir la imagen de referencia en cuadrantes y en base a ello sacar una escala aproximada para todos los valores tomando como referencia el centro siempre.

Ya que tenemos inicializadas las matrices de los modelos procedemos a asignar un elemento a cada matriz. Para formar la casa tomamos como base el triángulo azul y en base a ello escalamos los demás elementos, el caso de los árboles resulta un poco más sencillo pues una vez ajustada la escala simplemente reflejamos su posición con respecto al origen.

```
// Casa
//cuadrado rojo
model = glm::translate(model, glm::vec3(0.0f, -0.5f, -6.0f));
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
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));
meshColorList[5] -> RenderMeshColor();
//triángulo azul
model1 = glm::translate(model1, glm::vec3(0.0f, 0.75f, -5.0f));
model1 = glm::scale(model1, glm::vec3(1.0f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model1)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[2] -> RenderMeshColor();
//cuadrado verde
model2 = glm::translate(model2, glm::vec3(-0.4f, -0.2f, -4.0f)); //ventana
model2 = glm::scale(model2, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model2)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4] -> RenderMeshColor();
//cuadrado verde
model3 = glm::translate(model3, glm::vec3(0.4f, -0.2f, -4.0f)); //ventana
model3 = glm::scale(model3, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model3)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4] -> RenderMeshColor();
//cuadrado verde // puerta
model4 = glm::translate(model4, glm::vec3(0.0f, -1.0f, -4.0f));
model4 = glm::scale(model4, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model4)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[4] -> RenderMeshColor();
```

Figura 3. Figuras instanciadas con transformaciones

```
// Arbol
//triángulo verde
model5 = glm::translate(model5, glm::vec3(-1.3f, -0.6f, -2.0f));
model5 = glm::scale(model5, glm::vec3(0.25f, 0.35f, 0.35f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model5)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[3] -> RenderMeshColor();
//cuadrado cafe
model6 = glm::translate(model6, glm::vec3(-1.3f, -1.075f, -3.0f));
model6 = glm::scale(model6, glm::vec3(0.2f, 0.35f, 0.35f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model6)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[6] -> RenderMeshColor();

// Arbol
//triángulo verde
model7 = glm::translate(model7, glm::vec3(1.3f, -0.6f, -2.0f));
model7 = glm::scale(model7, glm::vec3(0.25f, 0.35f, 0.35f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model7)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[3] -> RenderMeshColor();
//cuadrado cafe
model8 = glm::translate(model8, glm::vec3(1.3f, -1.075f, -3.0f));
model8 = glm::scale(model8, glm::vec3(0.2f, 0.35f, 0.35f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model8)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[6] -> RenderMeshColor();
```

Figura 4. Figuras instanciadas con transformaciones

En la sección main, en caso de no haberlo hecho, modificamos el tipo de perspectiva que estamos utilizando, en este caso es una perspectiva ortogonal con valores arbitrarios, para finalmente ver la figura completa.

```
//Projection: Matriz de Dimensión 4x4 para indicar si vemos en 2D( orthogonal) o en 3D) perspectiva
glm::mat4 projection = glm::ortho(-2.0f, 2.0f, -1.5f, 1.5f, 0.1f, 100.0f); // proyección ortogonal, (L,R,B,T,N,F)(x,y,z)
//glm::mat4 projection = glm::perspective(glm::radians(60.0f), mainWindow.getBufferWidth() / mainWindow.getBufferHeight())
```

Figura 5. Perspectiva ortogonal



Figura 6. Izquierda, figura de referencia – Derecha, figura final

Problemas:

La parte de la primera actividad no hubo mayor problema más que el no confundirse entre las líneas del código para generar las figuras y modificar sus colores. El mayor problema se dio en la segunda actividad ya que como no se dio un sistema de referencia más que el centro, no se podía escalar de forma exacta por lo que una vez generada una figura medianamente parecida se debía de reajustar con flotantes cada vez más grandes.

Cuando una variable se repetía en nombre provocaba que: no se marcara la figura en la pantalla o que marcara un error y abriera la ventana de Windows indicando un problema en la ejecución directamente. Algunas cosas que pude notar es que al momento de escalar una figura si esta esca a es 0 el objeto desaparecía o si por el contrario era muy grande, al igual que si tenían una figura delante, las figuras se superponían. A pesar de ello el ejercicio se logró de forma adecuada.

Conclusión:

Con respecto a los ejercicios de clase considero que tienen un buen nivel, no son demasiado difíciles de hacer, aunque si son tardados por lo que no es posible realizarlos en el tiempo sobrante de la clase. Otra cuestión es que durante la clase no pude desarrollar la practica ni siquiera pues la computadora marcaba error al momento de instanciar algunas figuras, afortunadamente esto no ocurrió con mi computadora personal, pero siento que eso limitó un poco mi entendimiento del tema.