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 Nº 04

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: sábado 2 de marzo de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

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

1. **Terminar de Construir la grúa con: brazo de 3 partes, 4 articulaciones, 1 canasta y con un cuerpo de la grúa (prisma rectangular)**

Primero, decidí empezar el dibujo desde la base de la torre:

```
//BASE
model = glm::mat4(1.0);
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 3.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

Para esta parte, almacene en la variable modelaux la rotación alrededor del eje Y que todo el cuerpo debe presentar. Decidí empezar el dibujo de la figura desde su cuerpo para poder pasar más fácil las transformaciones (rotaciones) que iré haciendo.

Posteriormente, dibuje las llantas de la figura, aunque estas no rotan de momento (como si estuvieran avanzando)

```
//DIBUJO DE LAS LLANTITAS
model = modelaux;
model = glm::translate(model, glm::vec3(-2.0f, -1.0f, -1.75f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 0.5f, 1.0f));
color = glm::vec3(0.0f, 0.5f, 0.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

model = glm::translate(model, glm::vec3(0.0f, +7.05f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry();

model = glm::translate(model, glm::vec3(4.0f, +0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry();

model = glm::translate(model, glm::vec3(0.0f, -7.05f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry();
```

Luego, comencé con el dibujo de las articulaciones y los brazos. Primero dibuje las articulaciones y luego su correspondiente brazo de la siguiente manera:

- Primero guardo las transformaciones almacenadas en la variable modelaux en la variable model.
- Luego, realizo las transformaciones necesarias para el brazo y su articulación en particular: como la traslación para la articulación, el Angulo en que estará rotado

el brazo y la rotación que presentara al presionar alguna de las teclas (todos los brazos giran de la misma manera)

- Luego, guardo estas transformaciones en la variable modelaux para irlas acumulando, y así las articulaciones y brazos superiores giren cuando una articulación y brazo inferior a ellos también gire
- Luego, escalo y dibujo la esfera, y luego dibujo su brazo correspondiente.

Este proceso lo repetí 4 veces, para los 3 brazos y para la cabina en lo mas alto de los brazos

```
// SE EMPIEZA EL DIBUJO DE LOS BRAZOS
//articulación 1
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 1.25f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(0.0f, 1.0f, 1.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

//brazo1
model = modelaux;
model = glm::translate(model, glm::vec3(+3.0f, -0.3f, 0.0f));
model = glm::scale(model, glm::vec3(6.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

```
//Articulacion2
model = modelaux;
model = glm::translate(model, glm::vec3(6.0f, -0.25f, 0.0f));
model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

//brazo2
model = modelaux;
model = glm::translate(model, glm::vec3(+3.0f, -0.0f, 0.0f));
model = glm::scale(model, glm::vec3(6.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

```
//Articulacion3
model = modelaux;
model = glm::translate(model, glm::vec3(6.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

//brazo3
model = modelaux;
model = glm::translate(model, glm::vec3(+3.0f, -0.0f, 0.0f));
model = glm::scale(model, glm::vec3(6.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

```
//CABINA

//Articulacion4
model = modelaux;
model = glm::translate(model, glm::vec3(6.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

//cabezita
model = glm::translate(model, glm::vec3(-0.5f, 2.5f, 0.0f));
model = glm::scale(model, glm::vec3(3.0f, 5.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();
```

Por último, dibuje el suelo donde se sienta la figura, aunque este fragmento aparece primero antes que todas las demás figuras en el código

```
//SUELO
model = glm::translate(model, glm::vec3(-2.0f, -2.0f, -2.0f));
model = glm::scale(model, glm::vec3(500.0f, 0.0f, 500.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución se progra
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

El funcionamiento del código se encuentra en el video anexado

- **Problemas Presentados**

  Durante la clase de laboratorio, me pareció entender que debíamos realizar la maquina de arriba hacia abajo, empezando desde la cabina y bajando por los brazos hasta el cuerpo y las llantas. Intenté hacer de esta manera el código, pero no encontré una forma de pasar las transformaciones necesarias a la siguiente articulación para hacer que todas giraran como debían sin complicar mucho el código y creas más variables. Finalmente desistí, y decidí hacerlo desde abajo hacia arriba, empezando con la base de toda la figura.

- **Conclusión:**

  Gracias a la metodología que use para crear la figura, al final no se me complico tanto crearla de abajo hacia arriba, además de que entendí como es que se acumulan las transformaciones que usas en una matriz. Aprendí a crear figuras con transformaciones correspondientes a otras figuras y como pasar estas transformaciones a otras hasta llegar a un punto.