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 05

NOMBRE COMPLETO: Hernández Vázquez Daniela

Nº de Cuenta: 318092867

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 02

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 23 de marzo de 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Importar su modelo de coche propio dentro del escenario a una escala adecuada.

El modelo escogido es de un automóvil genérico sin textura obtenido de la página: https://free3d.com/es/modelo-3d/automobile-v1--84248.html, donde hay varios modelos gratuitos. En un primer momento abrimos el modelo con 3ds max y observamos que el modelo está volteado. Haciendo so de las herramientas del software es posible corregir esto y también podemos modificar directamente la escala del modelo para asi facilitar el manejo de las piezas que más adelante se van a exportar.

Para importar el modelo dentro del escenario, declaramos dentro del programa el modelo y posteriormente lo llamamos desde su ubicación en la carpeta Models.

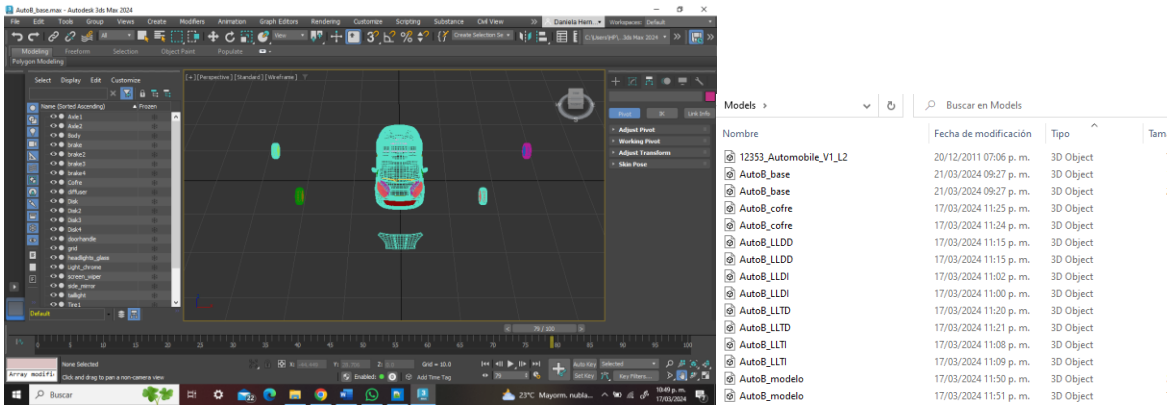
```
// Importar modelos de Auto
AutoB_modelo = Model();     // Modelo completo
AutoB_modelo.LoadModel("Models/AutoB_modelo.obj");
```

```
//Automovil
//Modelo completo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 0.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
AutoB_modelo.RenderModel();
```

Figura 1. Modelo base

2.- Importar sus 4 llantas y acomodarlas jerárquicamente, agregar el mismo valor de rotación a las llantas para que al presionar puedan rotar hacia adelante y hacia atrás.

Luego de manipular la geometría del modelo separamos el auto en las piezas que necesitamos en 3ds max y exportamos cada una de ellas; en este caso la base, las 4 llantas y el cofre. Posteriormente los modelos son declarados en el programa.




```
// Declaración de modelos de piezas de Automovil B
Model AutoB_modelo;     // Modelo completo
Model AutoB_base;       // base del auto
Model AutoB_cofre;      // cofre
Model AutoB_LLTI;       // Llanta 1
Model AutoB_LLDI;       // Llanta 2
Model AutoB_LLTD;       // Llanta 3
Model AutoB_LLDD;       // Llanta 4
```

```
// Importar modelos de Auto
AutoB_modelo = Model();     // Modelo completo
AutoB_modelo.LoadModel("Models/AutoB_modelo.obj");
AutoB_base = Model();
AutoB_base.LoadModel("Models/AutoB_base.obj");
AutoB_cofre = Model();
AutoB_cofre.LoadModel("Models/AutoB_cofre.obj");
AutoB_LLTI = Model();       // Llanta trasera izquierda
AutoB_LLTI.LoadModel("Models/AutoB_LLTI.obj");
AutoB_LLDI = Model();       // Llanta delatera izquierda
AutoB_LLDI.LoadModel("Models/AutoB_LLDI.obj");
AutoB_LLTD = Model();       // Llanta trasera derecha
AutoB_LLTD.LoadModel("Models/AutoB_LLTD.obj");
AutoB_LLDD = Model();       // Llanta delantera derecha
AutoB_LLDD.LoadModel("Models/AutoB_LLDD.obj");
```

Figura 2. Creación y exportación de modelos

Los modelos ya creados son llamados para ser acomodados jerárquicamente. Las 4 llantas están unidas al modelo de la base del carro utilizando la matriz auxiliar.

```
//base automovil
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 3.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getmueve()));
modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
AutoB_base.RenderModel();//modificar por el modelo sin las 4 patas y sin cola

//LlTI
modelaux = model;
model = glm::translate(model, glm::vec3(3.5f, -2.0f, -7.105f));
model = glm::rotate(model, glm::radians(mainWindow.getangulollantas()), glm::vec3(1.0f, 0.0f, 0.0f));
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));
AutoB_LLTI.RenderModel();
model = modelaux;
//LlDI
modelaux = model;
model = glm::translate(model, glm::vec3(3.5f, -2.0f, 7.305f));
model = glm::rotate(model, glm::radians(mainWindow.getangulollantas()), glm::vec3(1.0f, 0.0f, 0.0f));
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));
AutoB_LLDI.RenderModel();
model = modelaux;
```

```
//LlTD
modelaux = model;
model = glm::translate(model, glm::vec3(-4.0f, -2.0f, -7.105f));
model = glm::rotate(model, glm::radians(mainWindow.getangulollantas()), glm::vec3(1.0f, 0.0f, 0.0f));
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));
AutoB_LLTD.RenderModel();
model = modelaux;
//LlDD
modelaux = model;
model = glm::translate(model, glm::vec3(-4.0f, -2.0f, 7.305f));
model = glm::rotate(model, glm::radians(mainWindow.getangulollantas()), glm::vec3(1.0f, 0.0f, 0.0f));
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));
AutoB_LLDD.RenderModel();
model = modelaux;
```

Figura 3. Modelos de llantas

Para que puedan rotar hacia adelante y hacia atrás todas se manipulan con la variable *getangulollantas* y las teclas son declaradas en los archivos window.cpp y window.h asignando las teclas numéricas 1(giro hacia adelante) y 2(giro hacia atrás).

```
if (key == GLFW_KEY_1)
{
    theWindow->angulollantas += 10.0;
}
if (key == GLFW_KEY_2)
{
    theWindow->angulollantas -= 10.0;
}
```

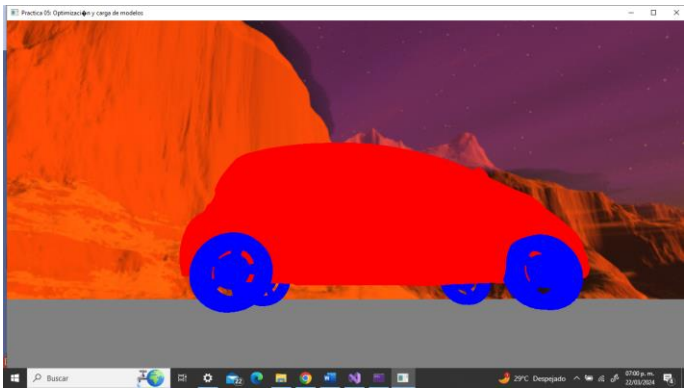
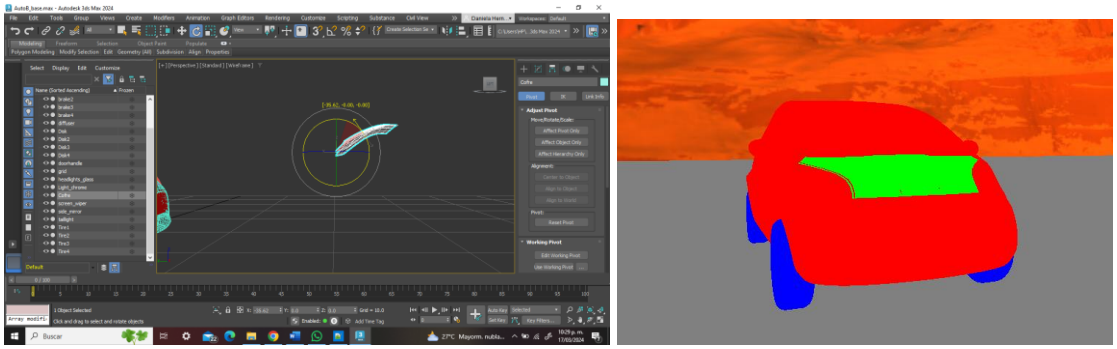


Figura 4. Importación y giro de llantas

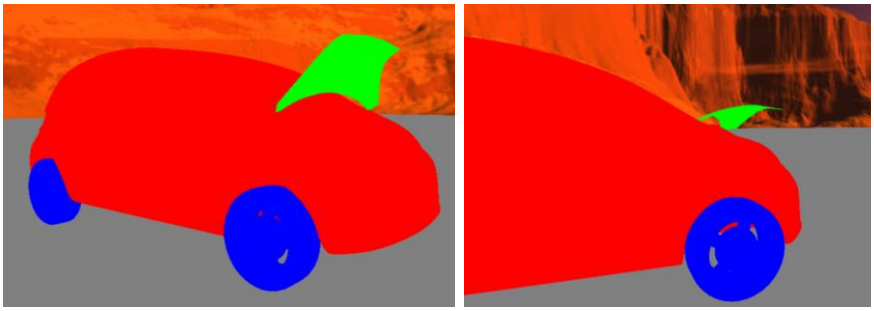
3.- Importar el cofre del coche, acomodarlo jerárquicamente y agregar la rotación para poder abrir y cerrar.

En el caso el modelo no tenia un cofre por lo que se tuvo que seleccionar parte por parte dentro del modelo para crear un cofre propio y unirlo utilizando detach.

El cofre se manipula con la variable *getangulocofre* y las teclas son declaradas en los archivos window.cpp y window.h asignando las teclas numéricas P(abrir) y M(cerrar).



```
if (key == GLFW_KEY_P)
{
    if (theWindow->angulocofre < -75.0)
    {
    }
    else
    {
        theWindow->angulocofre -= 10.0;
    }
}
if (key == GLFW_KEY_M)
{
    if (theWindow->angulocofre > -0.17)
    {
    }
    else
    {
        theWindow->angulocofre += 10.0;
    }
}
```



```
//cofre
modelaux = model;
model = glm::translate(model, glm::vec3(-0.19f, 2.255f, 7.005f));
model = glm::rotate(model, 17 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getangulocofre()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
AutoB_cofre.RenderModel();
```

Figura 4. Modelo de cofre

4.- Agregar traslación con teclado para que pueda avanzar y retroceder de forma independiente

Para que el automóvil pueda avanzar o retroceder se declara una nueva variable llamada mueve en window.h y window.cpp, esta se declara dentro del apartado de la ventana y se le asignan las variables Y(retrocede) y U(avanza), como se mencionó anteriormente para declararlas en el programa se declara como atributo heredado desde la base del automóvil y se declara que se trasladara en Z en este caso ya que es hacia donde se encuentra ubicado el carro.

```
GLfloat getmueve() { return mueve; }
```

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    mueve = 2.0f;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
```

```
// para avanzar o retroceder
if (key == GLFW_KEY_U)
{
    theWindow->mueve += 1.0;
}
if (key == GLFW_KEY_Y)
{
    theWindow->mueve -= 1.0;
}
```

```
//base automovil
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 3.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getmueve()));
modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
AutoB_base.RenderModel();
```

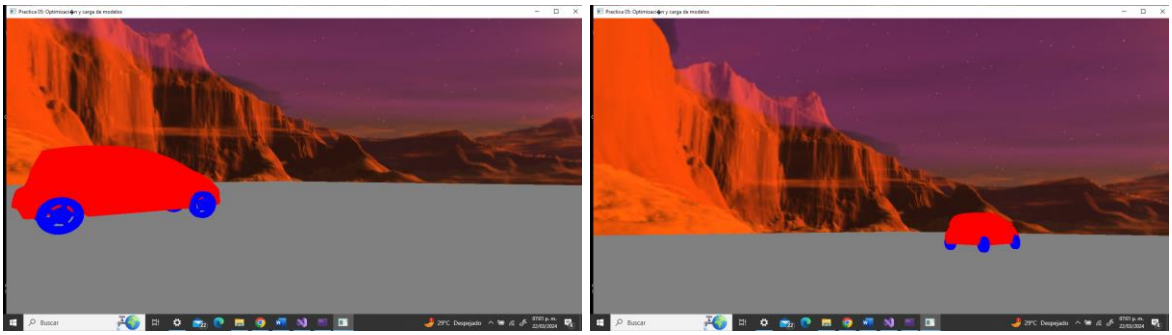


Figura 5. Avanzar y retroceder modelo

Problemas:

El primer problema fue que, dado que el modelo de mi coche no tenía cofre, tuve que separlo directamente, para esto utilicé detach. Una vez tuve todas las piezas y las exporté me di cuenta que el modelo quedaba con el frente del coche sobre Z lo que modificó el como tenia que trabajar las traslaciones y rotaciones, aunque para esto solo se tuvo que considerar la posición de los ejes para avanzar y realizar las transformaciones.

Para unir las piezas consideré ver el modelo desde la parte de atrás para decidir las direcciones de las llantas. Se importaron las 4 llantas, aunque hubiera también fácilmente creado una sola llanta y rotara en diferentes posiciones para crear las 4 y solo declarar un modelo. Las llantas no giran perfectamente porque el pivote al exportar el modelo no se pudo colocar exactamente en el centro de la rueda, a pesar de ello el movimiento se ve bien.

Lo más complicado del ejercicio nuevamente fue hacer que los atributos se heredaran adecuadamente y que la traslación o movimiento con las teclas también fuera el correcto. Para comprobar que la jerarquía estuviera bien, se modificaba la posición del cuerpo para asegurarse de que todo estuviera unido a él de forma correcta y se comprueba al desplazar el modelo completo.

En cuestión de exportar y modificar el modelo en 3dmax no tuve complicaciones más que para modificar el pivote de cada figura e intentar que este quedara en el centro para exportarlo.

Conclusión:

Creo que el ejercicio fue un poco más complicado debido a que se debe de considerar como importamos los objetos, ayuda el hecho de que al modelo al tener un pivote predefinido ahorramos código al no tener que crear elementos innecesarios. Lo más tardado de la actividad como siempre es realizar los ajustes a la geometría. Cada vez utilizamos mas variables para manipular el comportamiento de los elementos de lso modelos y, por tanto, hacemos uso de mas teclas.
Siento que en la clase va algo rápido porque si el programa no compila de inmediato nos atrasamos para realizar las demás configuraciones, además no podemos avanzar mucho a los ejercicios en la clase.

Referencias:

Kessenich, J. M., Sellers, G., & Shreiner, D. (2016). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V*. OpenGL.

Méndez Servín, M. (2022). *NOTAS PARA EL CURSO DE GRAFICACIÓN POR COMPUTADORA* [Libro electrónico]. https://prometeo.matem.unam.mx/recursos/VariosNiveles/iCartesiLibri/recursos/Notas_Graficacion_por_Computadora/index.html

Modelo: https://free3d.com/es/modelo-3d/automobile-v1--84248.html