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 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: 16 de marzo de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

Actividad 1

Como primera actividad se importa por separado cada elemento del modelo de Goddard y se agrega jerarquía en cuerpo, cabeza, mandíbula inferior y cada una de las 4 patas como un solo modelo.

Para realizar lo anterior haciendo uso de 3ds max utilizamos el modelo base de Goddard para separar los elementos necesarios, en cada elemento individualmente es modificado el pivote tomando como referencia desde dónde el elemento se va a mover y luego colocando la figura con el pivote en el origen exportamos cada uno de ellos.

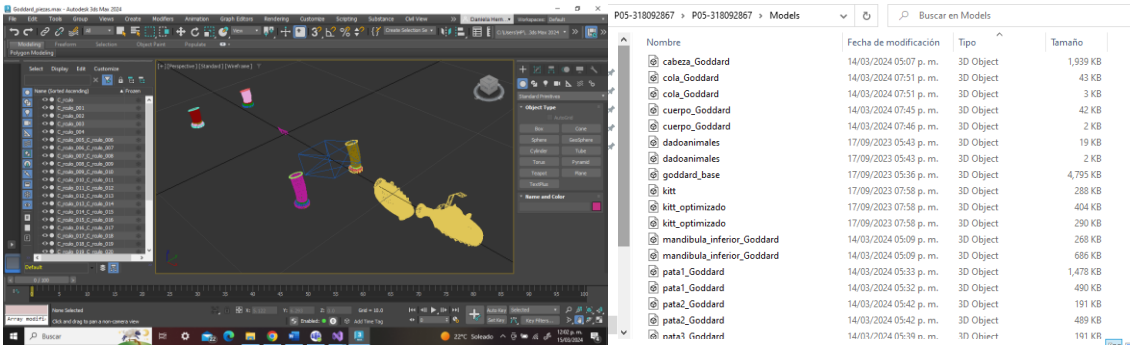

Figura 1. Creación y exportación de modelos

Posteriormente dentro del archivo de la practica se declaran los modelos que serán utilizados. Dentro del main son declarados de igual modo los modelos esta vez colocando la ubicación de cada modelo correspondiente dentro de la carpeta de Models.

```
36      Camera camera;
37
38      // Declaración de modelos de piezas de Goddard
39      Model cuerpo_Goddard;
40      Model cabeza_Goddard;
41      Model mandibula_inferior_Goddard;
42      Model cola_Goddard;
43      Model pata1_Goddard;     // Pata trasera izquierda
44      Model pata2_Goddard;     // Pata delantera izquierda
45      Model pata3_Goddard;     // Pata trasera derecha
46      Model pata4_Goddard;     // Pata delantera derecha
```

```
133     // Importar modelos de Goddard
134
135     cuerpo_Goddard = Model();
136     cuerpo_Goddard.LoadModel("Models/cuerpo_Goddard.obj");
137     cabeza_Goddard = Model();
138     cabeza_Goddard.LoadModel("Models/cabeza_Goddard.obj");
139     mandibula_inferior_Goddard = Model();
140     mandibula_inferior_Goddard.LoadModel("Models/mandibula_inferior_Goddard.obj");
141     cola_Goddard = Model();
142     cola_Goddard.LoadModel("Models/cola_Goddard.obj");
143
144     pata1_Goddard = Model();    // Pata trasera izquierda
145     pata1_Goddard.LoadModel("Models/pata1_Goddard.obj");
146     pata2_Goddard = Model();    // Pata delantera izquierda
147     pata2_Goddard.LoadModel("Models/pata2_Goddard.obj");
148     pata3_Goddard = Model();    // Pata trasera derecha
149     pata3_Goddard.LoadModel("Models/pata3_Goddard.obj");
150     pata4_Goddard = Model();    // Pata delantera derecha
151     pata4_Goddard.LoadModel("Models/pata4_Goddard.obj");
```

Figura 2. Se importan los modelos

Una vez importados los modelos procedemos a instanciar cada uno de ellos iniciando por el cuerpo y desde el comienza a jerarquizarse cada uno de os elementos del cuerpo de Goddard. En este caso solo se utiliza una matriz auxiliar para los demás elementos.

Actividad 2

Como segunda actividad se debe agregar rotaciones a las patas de forma independiente para que se muevan como si fuera a avanzar Goddard, limitar la rotación a 45° en Cada sentido.

Lo primero que hice para esta actividad fue definir con que teclas se movería cada elemento del cuerpo de Goddard:

- Cabeza: F(arriba) - G(abajo)
- Mandíbula inferior: H(arriba) - J(abajo)
- Cola: I(derecha) - O(izquierda)
- Pata1: K(adelante) - L(atrás)
- Pata2: Z(adelante) - X(atrás)
- Pata3: C(adelante) - V(atrás)
- Pata4: B(adelante) - N(atrás)

Luego dentro del archivo Window.cpp asigno a cada tecla su “articulación” correspondiente para llamarlo dentro del main, solo que ahora se colocan condiciones para que cuando llegue a 45° o -45° deje de realizar movimiento. De igual modo son declaradas las variables en el archivo Window.h inicializaondo el ángulo en 0°

```
if (key == GLFW_KEY_F)
{
    if (theWindow->angulocabeza > 45.0)
    {
    }
    else
    {
        theWindow->angulocabeza += 10.0;
    }
}
if (key == GLFW_KEY_G)
{
    if (theWindow->angulocabeza < -45.0)
    {
    }
    else
    {
        theWindow->angulocabeza -= 10.0;
    }
}
```

```
if (key == GLFW_KEY_H)
{
    if (theWindow->angulomandibula > 5.0)
    {
    }
    else
    {
        theWindow->angulomandibula += 10.0;
    }
}
if (key == GLFW_KEY_J)
{
    if (theWindow->angulomandibula < -45.0)
    {
    }
    else
    {
        theWindow->angulomandibula -= 10.0;
    }
}
```

```
if (key == GLFW_KEY_K)
{
    if (theWindow->angulopata1 > 45.0)
    {
    }
    else
    {
        theWindow->angulopata1 += 10.0;
    }
}
if (key == GLFW_KEY_L)
{
    if (theWindow->angulopata1 < -45.0)
    {
    }
    else
    {
        theWindow->angulopata1 -= 10.0;
    }
}
```

```
if (key == GLFW_KEY_Z)
{
    if (theWindow->angulopata2 > 45.0)
    {
    }
    else
    {
        theWindow->angulopata2 += 10.0;
    }
}
if (key == GLFW_KEY_X)
{
    if (theWindow->angulopata2 < -45.0)
    {
    }
    else
    {
        theWindow->angulopata2 -= 10.0;
    }
}
```

```
class Window
{
public:
    Window();
    Window(GLint windowWidth, GLint windowHeight);
    int Initialise();
    GLfloat getBufferWidth() { return bufferWidth; }
    GLfloat getBufferHeight() { return bufferHeight; }
    bool getShouldClose() {
        return  glfwWindowShouldClose(mainWindow);
    }
    bool* getsKeys() { return keys; }
    GLfloat getXChange();
    GLfloat getYChange();
    void swapBuffers() { return glfwSwapBuffers(mainWindow); }
    GLfloat getmovex() { return movex; }
    GLfloat getangulocola() { return angulocola; }
    GLfloat getangulocabeza() { return angulocabeza; }
    GLfloat getangulomandibula() { return angulomandibula; }
    GLfloat getangulopata1() { return angulopata1; }
    GLfloat getangulopata2() { return angulopata2; }
    GLfloat getangulopata3() { return angulopata3; }
    GLfloat getangulopata4() { return angulopata4; }

    ~Window();
    ~Window();
private:
    GLFWwindow* mainWindow;
    GLint width, height;
    bool keys[1024];
    GLint bufferWidth, bufferHeight;
    void createCallbacks();
    GLfloat lastX;
    GLfloat lastY;
    GLfloat xChange;
    GLfloat yChange;
    GLfloat rotax, rotay, rotaz;
    GLfloat movex;
    GLfloat angulocola = 0.0f;
    GLfloat angulocabeza = 0.0f;
    GLfloat angulomandibula = 0.0f;
    GLfloat angulopata1 = 0.0f;
    GLfloat angulopata2 = 0.0f;
    GLfloat angulopata3 = 0.0f;
    GLfloat angulopata4 = 0.0f;
    bool mouseFirstMoved;
    static void ManejaTeclado(GLFWwindow* window, int key, int code, int action, int mode);
    static void ManejaMouse(GLFWwindow* window, double xPos, double yPos);
};
```

Figura 3. Se declaran las variables para las rotaciones y se limitan.

En el archivo de la practica se crea todo el modelo con jerarquía a partir del cuerpo de forma similar a lo hecho en la práctica pasada, solo que en lugar de crear una articulacion con una esfera extra directamente se agregó la rotación a cada modelo y este gira en el lugar donde se ubica el pivote del modelo.

```
//Goddard
color = glm::vec3(0.0f, 0.0f, 0.0f); //modelo de goddard de color negro

//cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.5f, 0.0f));
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));
cuerpo_Goddard.RenderModel();//modificar por el modelo sin las 4 patas y sin cola

//cola
modelaux = model;
model = glm::translate(model, glm::vec3(-1.4f, 0.0f, 0.2f));
model = glm::rotate(model, glm::radians(mainWindow.getangulocola()), glm::vec3(0.0f, 1.0f, 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));
cola_Goddard.RenderModel();//modificar por el modelo sin las 4 patas y sin cola
```

```
// cabeza
model = modelaux;
model = glm::translate(model, glm::vec3(1.5f, 1.0f, 0.2f));
model = glm::rotate(model, glm::radians(mainWindow.getangulocabeza()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
cabeza_Goddard.RenderModel();
// mandibula inferior
model = glm::translate(model, glm::vec3(0.86f, -0.05f, -0.1f));
model = glm::rotate(model, glm::radians(mainWindow.getangulomandibula()), glm::vec3(0.0f, 0.0f, 1.0f));
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));
mandibula_inferior_Goddard.RenderModel();

//pata trasera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-0.2f, -0.9f, -0.5f));
model = glm::rotate(model, glm::radians(mainWindow.getangulopata1()), glm::vec3(0.0f, 0.0f, 1.0f));
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));
pata1_Goddard.RenderModel();
```

Figura 4. Jerarquía de modelo.

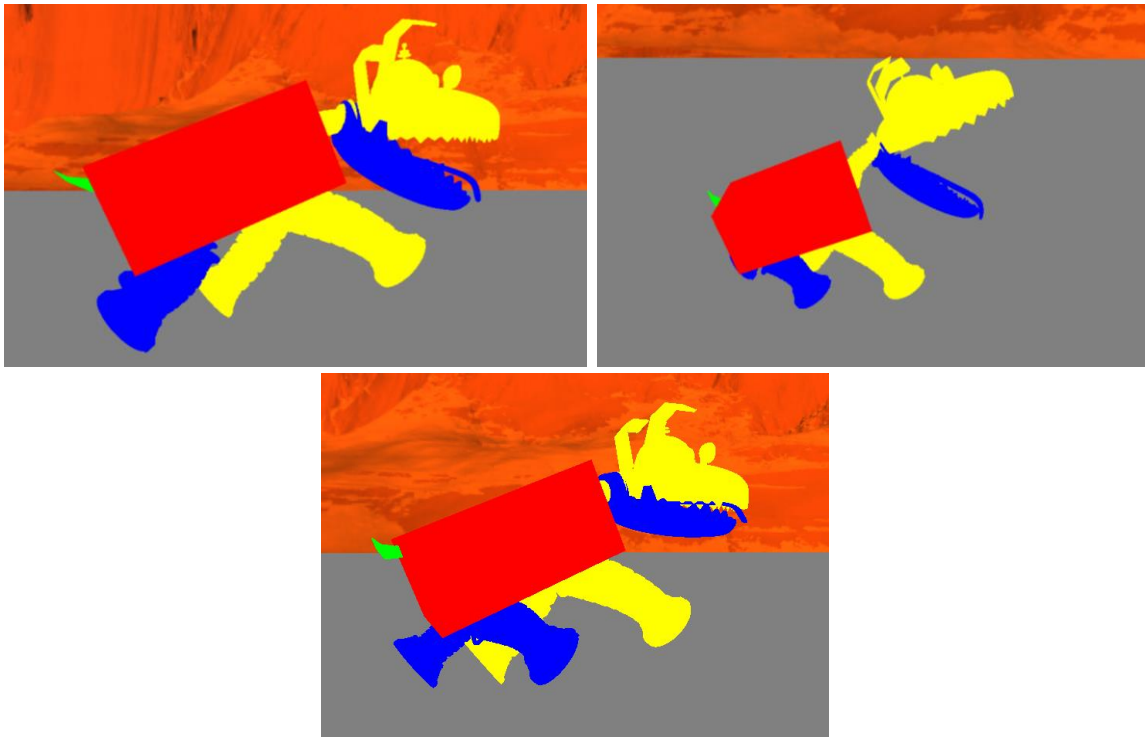

Figura 5. Resultado de ejecución

Problemas:

Para ir uniendo las piezas yo consideré ver el modelo desde la parte de atrás para decidir las direcciones por ello el cómo consideré las atas puede parecer algo extraño. El ejercicio creo que fue más fácil En este ejercicio solo la mayor complejidad fue hacer que los atributos se heredaran en la mandíbula. Para comprobar que la jerarquía estuviera bien esta vez, se modificaba la posición del cuerpo para asegurarse de que todo estuviera unido a él de forma correcta.

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

Conclusión:

Creo que los ejercicios están aumentando poco a poco su dificultad, pero 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 es realizar los ajustes a la geometría. Me ayudó el tener el video de cómo realizar las configuraciones y también que tuve que volver a revisar como realizar las jerarquías con el video de la practica 4, pero también lo que realmente ayudó mucho fue tener los modelos ya exportados solo para llamarlos en lugar de instanciarlos uno por uno con triángulos. Siento que en la clase va algo rápido porque si el programa no compila de inmediato nos atrasamos para realizar las demás configuraciones. A pesar de ello creo que la explicación estuvo bien para realizar el ejercicio.