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 09

NOMBRE COMPLETO: Tapia Ledesma Angel Hazel

Nº de Cuenta: 320070358

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 02

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 02-11-25

CALIFICACIÓN: _____

Actividades realizadas

1.- Separar del arco la parte del letrero y las puertas o rejas
2.-Hacer que en el arco que crearon se muestre la palabra: PROYECTO CGEIHC desplazándose las letras de derecha a izquierda como si fuera letrero LCD/LED de forma cíclica
3.- Hacer que al presionar una tecla una de las puertas se abra por medio de una rotación y la otra puerta se abra por medio de un deslizamiento.

Descripción de las actividades

Para la actividad 1 y 2, lo que se hizo fue trabajar primero el modelo en blender, en esta parte, se separó el cartel del arco, las puertas y en si la estructura de las columnas, obteniendo así 3 objetos separados. Una vez realizado esto, se importó a Open GL los modelos y se realizaron las transformaciones correspondientes para acomodar cada uno a su respectivo lugar. Una vez que colocamos todo correctamente, se procedió a la lógica para hacer que el cartel recorriera las letras cómo una especie de pantalla “LED”, para esto lo que se hizo fue importar una imagen con la textura del cartel, en este caso de “PROYECTO CGEIHC” con la tipografía del universo, una vez hecho esto, creamos un objeto de forma cuadrada, y en el modificamos los UVs de sus vertices, para adaptarlo a nuestra textura, una vez realizado esto, adaptamos esta figura a las dimensiones de nuestro cartel, para “sobreponerlo”, una vez hecho esto ya se mostraba el texto, pero aún faltaba recorrerlo, entonces lo que se realizó fue crear 2 variables, para controlar el offset en u y v para las coordenadas UV, “toffsetCartelu” y “toffsetCartelv, ya con esto teníamos casi todo, pues en el while lo que se hizo fue acumular en el offsetu el movimiento del cartel, ya que solo se iba desplazando en este eje, de igual forma agregamos un control para evitar el desbordamiento de la variable, cuando llegara a 1, se reiniciaba a 0.

Para la actividad 3 lo que se hizo fue, como se mencionó al inicio, acomodar en su lugar a las rejas, en este caso, tuvimos 2 casos, el de la rotación y el de la traslación, lo primero que se hizo fue en Window.cpp y Window.h poner las variables para manejar la apertura de la puerta derecha, puerta izquierda, también las banderas de inicio de animación, si la puerta está en la animación de abrir o cerrar, si ya acabó y también variables para controlar el tiempo de espera en que la puerta se mantiene abierta.
Una vez realizado esto en window.cpp, lo que hicimos fue trabajar la lógica en el main, la primer parte se basa en un “toggle” el cual sencillamente indica cuando se presionó el boton para iniciar la animación, dentro de este if, lo que se hace es sencillamente

actualizar las variables para indicar que se ha entrado al estado en que se están abriendo las puertas, después de esto tenemos otro if, el cual ahora se basa en “puertasAbriendo” este if es el que revisa que no se pueda volver a ejecutar la animación si ya está siendo ejecutada, ya que si no existiera, al presionar varias veces la tecla de la animación, esta no terminaría, se interrumpiría y se volvería a ejecutar, así que esta parte del código es útil para eso, de igual forma es útil, porque dentro de ella es donde se realiza la lógica de apertura de las puertas, tanto por rotación cómo por traslación, para esto, lo que se hace es acumular el offset de movimiento y rotación para las respectivas puertas, pero esto viene acompañado después de una validación de condición, el cual es un “tope” o actúa para que tanto la rotación cómo la traslación no se pasen del límite, y en caso de llegar al límite, actualizar las variables a este límite. Después viene otro if, el cual valida que ahora estas 2 cosas han sucedido, que ambas rejas están en su límite, en caso de que esto se cumpla, lo único que hace es actualizar las variables de puertasAbiertas y puertasAbriendo para que se pueda pasar al siguiente if, el cual es el responsable de tenerlas abiertas un tiempo. Una vez que ha acabado la etapa de “abrir” las puertas, ahora se va a un if, el cual valida que estén abiertas y en este if, se da la lógica para esperar el tiempo que se decidió, dentro de este if hay otro if, el cual, una vez que se ha cumplido el tiempo de espera, lo que hace es actualizar las banderas de “puertasAbiertas” y “puertasCerrando” para proceder a la siguiente parte, donde las puertas inician el proceso de cierre.

Pasamos al último bloque de código para la animación de las puertas, el cual es también un if, el cuál con la bandera “puertasCerrando” valida que estemos en esta etapa, dentro tenemos ifs anidados, los cuales siguen la misma lógica que en abriendo, pero aqui a la inversa, partiendo de los límites máximos a llegar al final al estado inicial de las puertas, es decir, cuando sus valores eran 0, de igual forma también se incluye un if para validar que ambas rejas hayan “cerrado” y se actualiza el estado de las banderas, llevándolas al estado “inicial” para volver a ejecutar la animación si se toca la tecla.

Código generado:
P09-320070358.cpp

```
// practica
float toffsetCartelu = 0.0f;
float toffsetCartelv = 0.0f;

// para puertas (lo del 60 es por como definimos el delta time)
const GLfloat VELOCIDAD_ROTACION_PUERTA = 90.0f / 60.0f;
const GLfloat VELOCIDAD_TRASLACION_PUERTA = 11.0f / 60.0f;
const GLfloat MAX_ROTACION_REJA_IZQUIERDA = -90.0f;
const GLfloat MAX_TRASLACION_REJA_DERECHA = 11.0f;
const GLfloat TIEMPO_ESPERA_ABIERTA = 1.0f*60;

// Variables para manejar el tiempo
float acumuladorTiempoNumero = 0.0f;
const float tiempoPorNumero = 60.0f;
```

```
Texture CartelTexture;

Model Arco_M;
Model Reja_M;
Model Cartel_M;
```

```
// Rectangulo
unsigned int rectanguloIndices[] = {
    0, 1, 2,
    0, 2, 3,
};

GLfloat rectanguloVertices[] = {
    -0.5f, 0.0f, 0.5f,      0.0f, 0.0f,     0.0f, -1.0f, 0.0f,
    0.5f, 0.0f, 0.5f,       1.0f, 0.0f,     0.0f, -1.0f, 0.0f,
    0.5f, 0.0f, -0.5f,      1.0f, 1.0f,     0.0f, -1.0f, 0.0f,
    -0.5f, 0.0f, -0.5f,     0.0f, 1.0f,     0.0f, -1.0f, 0.0f,

};
```

```
Mesh* obj8 = new Mesh();
obj8->CreateMesh(rectanguloVertices, rectanguloIndices, 32, 6);
meshList.push_back(obj8);
```

```
CartelTexture = Texture("Textures/LetreroTexto.png");
CartelTexture.LoadTextureA();

Model Arco_M = Model();
Arco_M.LoadModel("Models/arco.obj");
Model Reja_M = Model();
Reja_M.LoadModel("Models/reja.obj");
Model Cartel_M = Model();
Cartel_M.LoadModel("Models/letrero.obj");
```

```
// Movimiento del cartel como si fuera un cartel LED
toffsetCartelu += 0.001 * deltaTime;
toffsetCartelv = 0;
if (toffsetCartelu > 1.0f)
{
    toffsetCartelu = 0.0f;
}
toffset = glm::vec2(toffsetCartelu, toffsetCartelv);
```

```
// Practica 9
// Arco
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-5.0f, 5.5f, 0.0f));
modelaux = model;
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));
Arco_M.RenderModel();

if (mainWindow.puertaToggle) // Si se activó la tecla para abrir
{
    mainWindow.puertasAbriendo = true;
    mainWindow.puertasCerrando = false;
    mainWindow.puertasAbiertas = false;
    mainWindow.tiempoPuertasAbiertas = 0.0f; // Reiniciar el contador
    mainWindow.puertaToggle = false; // Desactivar el toggle para que no se repita
}

if (mainWindow.puertasAbriendo)
{
    // Reja izquierda (la que rota)
    mainWindow.rejaIzquierdaRotation -= VELOCIDAD_ROTACION_PUERTA * deltaTime;
    if (mainWindow.rejaIzquierdaRotation < MAX_ROTACION_REJA_IZQUIERDA) {
        mainWindow.rejaIzquierdaRotation = MAX_ROTACION_REJA_IZQUIERDA;
    }

    // Reja derecha (la que trasla)
    mainWindow.rejaDerechaTranslation += VELOCIDAD_TRASLACION_PUERTA * deltaTime;
    if (mainWindow.rejaDerechaTranslation > MAX_TRASLACION_REJA_DERECHA) {
        mainWindow.rejaDerechaTranslation = MAX_TRASLACION_REJA_DERECHA;
    }

    // Solo actualiza las variables para lo que está abajo
    if (mainWindow.rejaIzquierdaRotation <= MAX_ROTACION_REJA_IZQUIERDA &&
        mainWindow.rejaDerechaTranslation >= MAX_TRASLACION_REJA_DERECHA)
    {
        mainWindow.puertasAbriendo = false;
        mainWindow.puertasAbiertas = true; // Ahora están abiertas
    }
}
```

```
if (mainWindow.puertasAbiertas)
{
    // Tiempo en el que se conservan abiertas
    mainWindow.tiempoPuertasAbiertas += deltaTime;
    if (mainWindow.tiempoPuertasAbiertas >= TIEMPO_ESPERA_ABIERTA)
    {
        mainWindow.puertasAbiertas = false;
        mainWindow.puertasCerrando = true; // Iniciar el cierre
    }
}

if (mainWindow.puertasCerrando)
{
    // Cerrar reja izquierda (rotación)
    mainWindow.rejaIzquierdaRotation += VELOCIDAD_ROTACION_PUERTA * deltaTime;
    if (mainWindow.rejaIzquierdaRotation > 0.0f) {
        mainWindow.rejaIzquierdaRotation = 0.0f;
    }

    // Cerrar reja derecha (traslación)
    mainWindow.rejaDerechaTranslation -= VELOCIDAD_TRASLACION_PUERTA * deltaTime;
    if (mainWindow.rejaDerechaTranslation < 0.0f) {
        mainWindow.rejaDerechaTranslation = 0.0f;
    }

    // Si ambas están completamente cerradas
    if (mainWindow.rejaIzquierdaRotation >= 0.0f &&
        mainWindow.rejaDerechaTranslation <= 0.0f)
    {
        mainWindow.puertasCerrando = false;
    }
}

// RejaIzquierda (rota)
model = modelaux;
model = glm::translate(model, glm::vec3(-1.75f, 0.0f, -11.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrejaIzquierdaRotacion()), 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));
Reja_M.RenderModel();
```

```
// Reja derecha (Se mueve)
model = modelaux;
model = glm::translate(model, glm::vec3(-2.0f, 0.0f, 11.0f+mainWindow.getrejaDerechaTranslation()));
model = glm::rotate(model, 180 * toRadians, 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));
Reja_M.RenderModel();

// Cartel
model = modelaux;
model = glm::translate(model, glm::vec3(-1.75f, 11.0f, 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));
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
Cartel_M.RenderModel();

model = modelaux;
model = glm::translate(model, glm::vec3(-2.4f, 11.0f, 0.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(23.0f, 1.2f, 4.0f));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
CartelTexture.UseTexture();
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[7]->RenderMesh();
```

Window.h

```
public:
    Window();
    Window(GLint windowWidth, GLint windowHeight);
    int Initialise();
    GLfloat getBufferWidth() { return bufferWidth; }
    GLfloat getBufferHeight() { return bufferHeight; }
    GLfloat getXChange();
    GLfloat getYChange();
    GLfloat getmuevez() { return muevez; }
    GLfloat getmuevexhelicoptero() { return muevexhelicoptero; }
    GLfloat getrejaIzquierdaRotacion() { return rejaIzquierdaRotation; }
    GLfloat getrejaDerechaTranslation() { return rejaDerechaTranslation; }
    bool getShouldClose() {
        return  glfwWindowShouldClose(mainWindow);}
    bool* getsKeys() { return keys; }
    void swapBuffers() { return glfwSwapBuffers(mainWindow); }
    bool getprendida() { return prendida; }
    GLfloat getlucespuntuales() { return lucesPuntuales; }
    bool getDireccion() { return direccion; }
    bool puertaToggle;
    bool puertasAbriendo;
    bool puertasCerrando;
    bool puertasAbiertas;
    GLfloat tiempoPuertasAbiertas;
    GLfloat rejaIzquierdaRotation;
    GLfloat rejaDerechaTranslation;
    ~Window();
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    bool keys[1024];
    GLint bufferWidth, bufferHeight;
    void createCallbacks();
    GLfloat lastX;
    GLfloat lastY;
    GLfloat xChange;
    GLfloat yChange;
    GLfloat muevez;
    GLfloat muevexhelicoptero;
    bool mouseFirstMoved;
    bool prendida;
    bool direccion;
    GLfloat lucesPuntuales;
    GLfloat articulacion1;
```

Window.cpp

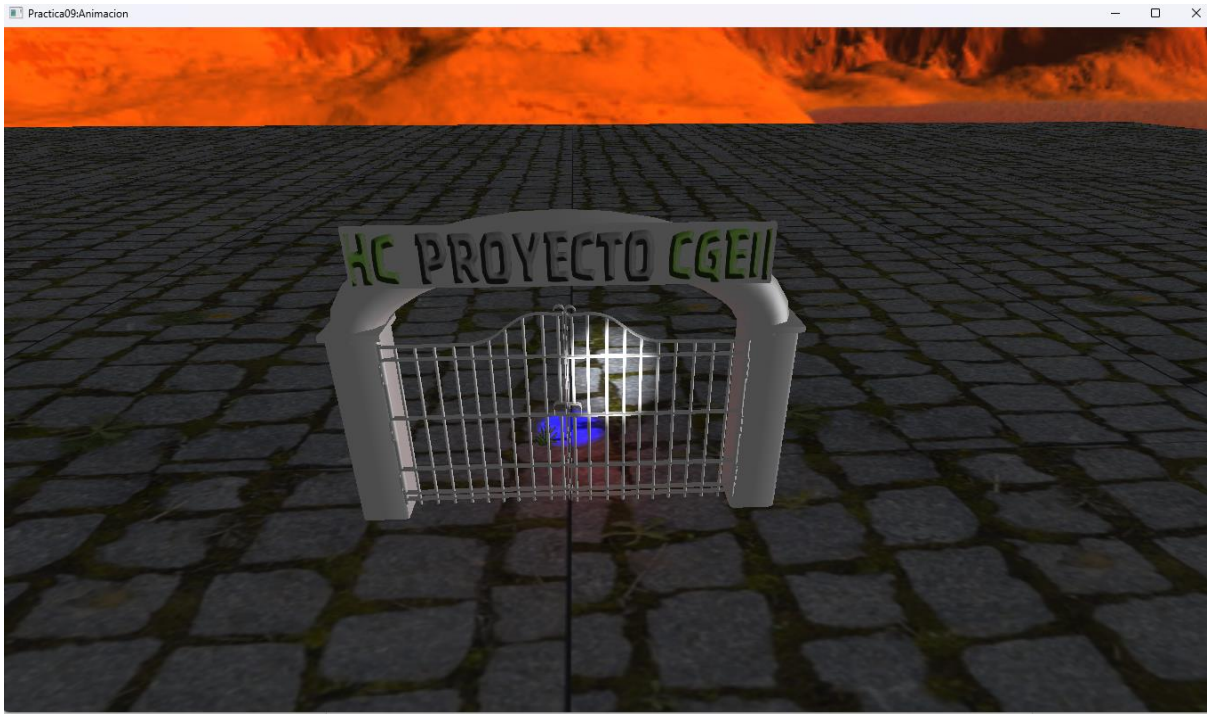
```
Window::Window()
{
    width = 800;
    height = 600;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
    prendida = true;
    lucesPuntuales = 0.0f;
    direccion = false;
    puertaToggle = false;
    puertasAbriendo = false;
    puertasCerrando = false;
    puertasAbiertas = false;
    tiempoPuertasAbiertas = 0.0f;
    rejaIzquierdaRotation = 0.0f;
    rejaDerechaTranslation = 0.0f;
}
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    muevez = 0.0f;
    muevexhelicoptero = 2.0f;
    prendida = true;
    lucesPuntuales = 0.0f;
    articulacion1 = 0.0f;
    direccion = false;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
    puertaToggle = false;
    puertasAbriendo = false;
    puertasCerrando = false;
    puertasAbiertas = false;
    tiempoPuertasAbiertas = 0.0f;
    rejaIzquierdaRotation = 0.0f;
    rejaDerechaTranslation = 0.0f;

}
```

```
if (key == GLFW_KEY_H && action == GLFW_PRESS)  {
    if (!theWindow->puertasAbriendo && !theWindow->puertasCerrando)
    {
        theWindow->puertaToggle = true;          }
}
```

Ejecución

Actividad 1,2

Actividad 3 (Animación con tecla “H”)



Video: https://youtu.be/9M4Sv_8onf8

Problemas presentados

Hubo 2 problemas principales, el primero fue con respecto a la actividad 1 y 2, ya que al aplicar la textura al carte, pasaba algo, y era que se aplicaba a todas las caras del cartel, incluyendo las laterales y superiores, no solo la frontal, lo que se hizo para solucionar esto, ya que es un modelo y no un objeto al cual pudiéramos modificarle los UVS en open GL, ya que es un objeto de blender, así que lo que se hizo fue crear un cuadrado en Open GL con los UVS adaptados a la textura y después escalarlo para adaptar la forma del cartel, y "sobreponer" esta figura para que se viera sobre el cartel, el otro problema fue en el punto 3 de la práctica, ya que en la animación surgieron varios imprevistos, lo primero fue que al inicio solo se aplicó la rotación y traslación cómo tal, sin banderas de control, pero lo que pasaba era que podía interrumpirse la animación y daba la impresión de que se "teletransportaban" las rejas o se reiniciaban al estado inicial de forma abrupta, por lo que para resolver esto, se implementaron las banderas para indicar cuando las puertas estuvieran en estas etapas de "apertura", "Abiertas al máximo", "espera" y "cierre", de forma que ya no pudieran ser interrumpidas aunque se presionara muchas veces el botón de la animación.

Conclusión:

La práctica fue muy entretenida, en general creo que se aprendió sobre animación, más específicamente el manejo de las texturas con movimiento, a través del offset en este caso, y entender un poco más cómo es que funciona y porque es importante el mapeo de las UVs. También ayudó a que repasáramos todo lo anterior que sabíamos cómo transformaciones y texturizado.

En cuanto a la complejidad, creo que fue intermedia, en el caso de las texturas, creo que el hecho de haberlo hecho con la imagen del alfabeto hubiera sido muchísimo más complejo que la manera por la cual se optó, y creo que realmente no hubiera aportado mucho, si acaso, solo practicar un poco más sobre el offset aplicado a la textura, creo que el hecho de usar otra textura con el texto que debía tener el cartel fue una solución mucho más práctica y sencilla de implementar, de igual forma se logró usar y entender el offset. En cuanto a la animación de las rejas, creo que no fue tan complicado, más que nada lo difícil era imaginar los posibles casos que podían pasar, como el error que tuve de no considerar que el usuario podía interrumpir la animación, pero fuera de eso, no hubo mucha complejidad, ya que eran ifs con banderas o controles.

Cómo comentarios solo sugerirían el de explicar un poco más sobre el offset en las texturas, y en conjunto con los ejemplos que había en código se hubiera entendido mucho mejor, otra sugerencia es quizá explicar cómo modificar las UVS de un modelo importado de blender en Open GL para hacer la adaptación de la textura directamente sobre este objeto.