UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

PREVIO Nº 02

NOMBRE COMPLETO: Pérez Uribe José Alberto

Nº de Cuenta: 318143213

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 07

SEMESTRE 2026-2

FECHA DE ENTREGA LÍMITE: 22/02/2026

CALIFICACIÓN: _____

1. ¿Cómo funciona glm::ortho?
Genera una matriz de proyección ortográfica. Esto significa que define un volumen rectangular dentro del cual se van a dibujar los objetos, y todo lo que quede fuera de ese rango simplemente no se ve. En este tipo de proyección no existe efecto de profundidad visual internamente, esta función construye una matriz que transforma las coordenadas del mundo al espacio normalizado que usa OpenGL. Para hacerlo necesita saber hasta dónde quieres ver en horizontal, en vertical y en profundidad.

Ejemplo
```
glm::ortho(-2.0f, 2.0f, -2.0f, 2.0f, -1.0f, 1.0f);
```
Aquí se dice que dibuje lo que esté entre -2 y 2 en X y Y y entre -1 y 1 en Z.

2. ¿Cómo funciona glm::perspective?
Crea una matriz de proyección con perspectiva, es decir, simula cómo ve el ojo humano: los objetos que están más lejos se ven más pequeños. A diferencia de la ortográfica, aquí el volumen visible tiene forma de pirámide truncada.
Para poder generarla necesita saber que tan abierta está la cámara (ángulo de visión) la relación entre el ancho y el alto de la ventana y hasta que distancia se debe dibujar

```
glm::perspective(glm::radians(45.0f), 1.0f, 0.1f, 100.0f);
```
Esto crea una cámara con un ángulo de 45 grados, que empieza a ver desde 0.1 unidades y deja de ver después de 100.

3. ¿Cómo funciona glViewport?
Le dice a OpenGL en qué parte de la ventana se va a dibujar. OpenGL trabaja internamente con coordenadas normalizadas que van de -1 a 1, pero la ventana tiene tamaño en pixeles. Esta función hace la conversión entre ese espacio interno y el tamaño real de la ventana. Cuando se llama, se indica desde qué punto de la ventana empieza el dibujo y qué tamaño tendrá.

```
glViewport(0, 0, 800, 800);
```
Eso significa que el área de dibujo empieza en la esquina inferior izquierda y ocupa toda la ventana de 800 por 800 píxeles

4. ¿Cómo funciona glm::translate?
Se usa para mover un objeto dentro del espacio. No cambia su forma ni su tamaño, solo modifica su posición. Lo que hace es generar una matriz de transformación que desplaza

todas las coordenadas del objeto la misma cantidad en X, Y o Z
glm::translate(model, glm::vec3(1.0f, 0.0f, 0.0f));
Esto mueve el objeto una unidad hacia la derecha

5. ¿Cómo funciona glm::rotate?
Sirve para girar un objeto alrededor de un eje. Para hacerlo necesita un ángulo (en radians) y un vector que indique el eje de rotación
glm::rotate(model, glm::radians(45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
Este rota el objeto 45 grados alrededor del eje Z.

6. ¿Cómo funciona glm::scale?
Cambia el tamaño del objeto multiplicando sus coordenadas por ciertos factores. Si el valor es mayor que 1, el objeto crece; si es menor que 1, se reduce.
glm::scale(model, glm::vec3(2.0f, 2.0f, 1.0f));
Esto hace que el objeto sea el doble de grande en X y en Y

7. ¿Qué son las variables Uniform en GLSL y cómo se usan?
Las variables uniform son variables del shader que reciben su valor desde el programa en C++. Se usan cuando un dato debe ser igual para todos los vértices o todos los fragmentos, como una matriz de transformación o un color general.
En el shader:
uniform mat4 model;
En C++:
glUniformMatrix4fv(glGetUniformLocation(shader, "model"), 1, GL_FALSE
glm::value_ptr(model));
Esto envía la matriz model al shader.

8. ¿Cómo funciona gl_Position?
Es una variable especial del Vertex Shader y es obligatoria. En ella se debe guardar la posición final del vértice después de aplicar todas las transformaciones. OpenGL usa ese valor para saber dónde colocar el vértice en la pantalla. Sino se asigna un valor a gl_Position, el objeto no se dibuja.
gl_Position = projection * view * model * vec4(aPos, 1.0);
Aquí se está aplicando la transformación completa antes de dibujar.

9. ¿Cómo funciona la variable extern?
Es una palabra reservada. Se usa cuando una variable fue definida en otro archivo del proyecto, pero queremos usarla sin volver a declararla como nueva.

Conclusión

En estas preguntas entendí mejor como funcionan las transformaciones y proyecciones dentro de OpenGL usando GLM.
Entendi que glm::ortho y glm::perspective sirven para definir como se va a ver la escena, ya sea sin profundidad o con efecto de perspectiva.
Así como las otras funciones para mover, girar y cambiar el tamaño de los objetos mediante matrices. Así como glViewport que adapta el dibujo al tamaño de la ventana y variables uniform.
En general estos comandos/codigo ayudan a controlar cómo se posicionan y se ven los objetos dentro de la escena.

Bibliografia

- LearnOpenGL - Sistemas de coordenadas. (n.d.). https://learnopengl-com.translate.goog/Getting-started/Coordinate-Systems?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc
- LearnOpenGL - Hello Triangle. (n.d.). https://learnopengl.com/Getting-started/Hello-Triangle
- *LearnOpenGL - Transformations*. (n.d.). https://learnopengl.com/Getting-started/Transformations
- *LearnOpenGL - Shaders*. (n.d.). https://learnopengl.com/Getting-started/Shaders
- GeeksforGeeks. (2025, July 23). *extern Keyword in C*. GeeksforGeeks. https://www.geeksforgeeks.org/c/understanding-extern-keyword-in-c/
- Khronos Group. (2023). *The OpenGL® Shading Language (Version 4.60.8)* (G. Leese, J. Kessenich, D. Baldwin & R. Rost). https://registry.khronos.org/OpenGL/specs/gl/GLSLangSpec.4.60.pdf