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 01

NOMBRE COMPLETO: Aguilar Pérez José Ramón

Nº de Cuenta: 317515048

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

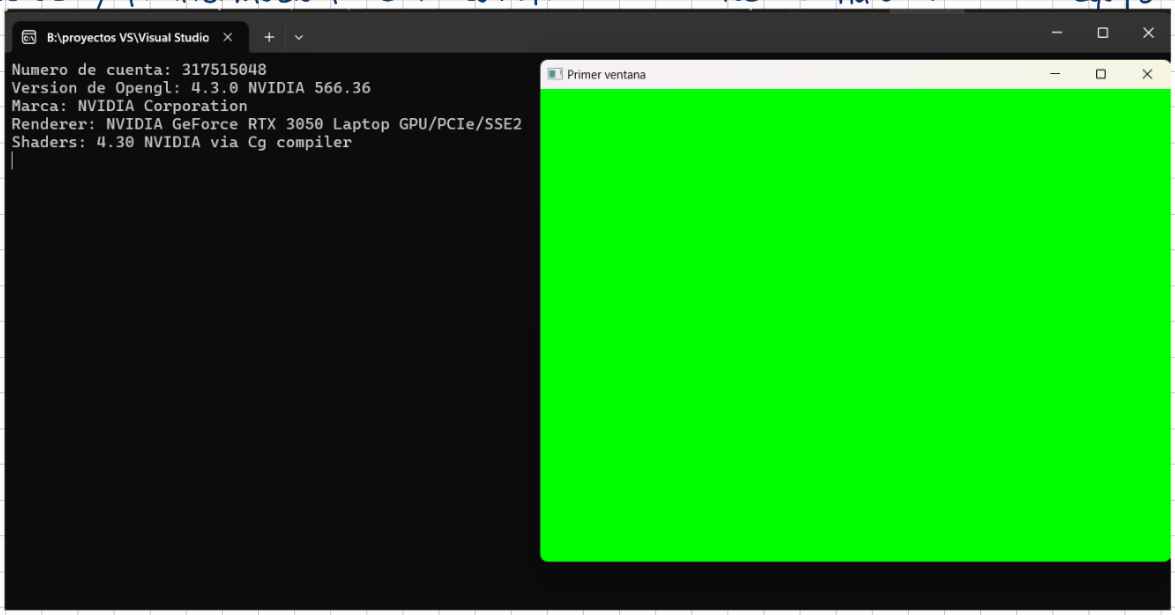
SEMESTRE: 2025-2

FECHA DE ENTREGA LÍMITE: 15/febrero/2025

CALIFICACIÓN: _____

1. Contenido requerido

1. Captura de pantalla como la del manual de configuración en la cual se muestra la ventana de fondo verde y la información de la consola con los datos de hardware de su equipo de cómputo.



2. ¿Qué es un VAO?
   Un Vertex Array Object es un objeto de OpenGL que permite almacenar el estado de los atributos de los vértices de una malla 3D. En otras palabras, almacena la información sobre como dibujar nuestro objeto.

3. ¿Qué es un VBO?
   Un Vertex Buffer Object es un tipo de buffer que almacena los datos de los vértices en la memoria del GPU. Los VBO funcionan dentro de los VAO, permitiendo así que OpenGL pueda saber como extraer la data del buffer, permitiendo así dibujar la figura.

4. ¿Qué parámetros recibe el comando glVertexAttribPointer?
   1. Indice de atributo: Variable que almacena la posición de los vértices usados
   2. Número de posiciones/vértices que se leerán del buffer
   3. Tipo de data con el cual están representados los datos (Ej: float → GL_FLOAT).
   4. Normalización: Indica si los valores deben estar normalicados o no.
   5. Especifica si se tiene más información que solamente los vertices en el buffer.
   6. Indica el inicio donde se debe leer el buffer.

5. ¿Qué información maneja Vertex Shader?
   Maneja la posición (coordenadas) de los vértices de la figura.

6. ¿Qué información maneja Fragment Shader?
   Maneja los colores en formato RGB que tendrán los pixeles de la figura.

7. ¿Qué parámetros recibe el comando glDrawArrays?
   1. Indica la primitiva a dibujar (forma, ej: Triángulos → GL_TRIANGLES, Lineas → GL_LINES, etc)
   2. Posición inicial de lectura.
   3. Número de vértices que se necesitan leer para hacer el dibujo.

8. ¿Qué son los variables Uniform dentro de GLSL y cómo se declaran y se mandan desde OpenGL a GLSL?
   Las variables de tipo Uniform permiten la comunicación entre la CPU y la GPU, estas mantienen su valor durante la ejecución del shader. Para la declaración y reclamiento por OpenGL es necesario:
   1. Declarar una variable de tipo Uniform: uniform [tipo de dato] [nombre]
   2. Obtener la ubicación de la variable anterior con glGetUniformLocation([shader], [var. uniform]).

3. Se establece un arreglo Uniform con ayuda de la función glUniform{1,2,3,4}{f,i,ui}v([ubicación de var. uniform], [tamaño del arreglo], [puntero del arreglo]), que indica un puntero que contiene los datos a establecer. También puede ser una matriz con glUniformMatrix{1,2,3,4}{f,i,ui}v.
Las variables uniform conservan su valor hasta que cambien o se use otro shader.

9. Proyecciones planares por medio de glm. (Matriz y línea de código).
Una proyección transforma puntos de un sistema de coordenadas de dimensión n a uno de dimensión menor que n. Existen tres tipos de proyecciones planares: paralela, oblicua y ortográfica. En computación gráfica se utilizan matrices de transformación 4X4:

```
glm::mat4 myMatrix;
glm::vec4 myVector;  → glm::vec4 transformedVector = myMatrix * myVector
```

10. Matrices de tranformación de Traslación, Rotación y Escala y con glm.
- Matriz de traslación: mueve un objeto en el espacio sin cambiar su tamaño o rotación.
  `glm::mat4 myMatrix = glm::translate(glm::mat4(n), glm::vec3(n, n, n));  n->float`
- Matriz de rotación: gira un objeto en dirección a un eje específico.
  `glm::vec3 myRotatingAxis(n, n, n); → glm::rotate(ángulo, myRotatingAxis);`
- Matriz de escala: cambia el tamaño del objeto en cada eje.
  `glm::mat4 myScalingMatrix = glm::scale(n, n, n);`

2. Conclusión
Gracias a la investigación realizada para esta práctica, pude comprender que uno de los elementos a la hora de trabajar con OpenGL son los vectores, ya que estos permiten diseñar los distintos dibujos que se pueden crear con estos. Además, cuando se aplican estos elementos como en matrices, se pueden crear proyecciones 3D en nuestros dibujos, las cuales se le pueden aplicar distintas operaciones como rotación, traslación y escalamiento, permitiendo así muchas posibilidades para los dibujos futuros.

3. Bibliografía

- De programación, T. (2016, septiembre 25). GLSL Variables Uniform. Blogspot.com. http://acodigo.blogspot.com/2016/09/glsl-variables-uniform.html
- Rogrp, E. P. (2020, mayo 21). OpenGL – Básico (C++). Home.blog. https://regor.home.blog/2020/05/21/opengl-basico-c/
- Saturnoz, S. (2017, noviembre 7). 07 - usando shaders. Medium. https://medium.com/@saturnozmarte/07-usando-shaders-fed8517eec90
- Tutorial 3 : Matrices. (s/f). Opengl-tutorial.org. Recuperado el 14 de febrero de 2025, de http://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/