



UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e
INTERACCIÓN HUMANO COMPUTADORA



PREVIO N° 1

NOMBRE COMPLETO: Gómez Enríquez Agustín

N° de Cuenta: 317031405

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 5

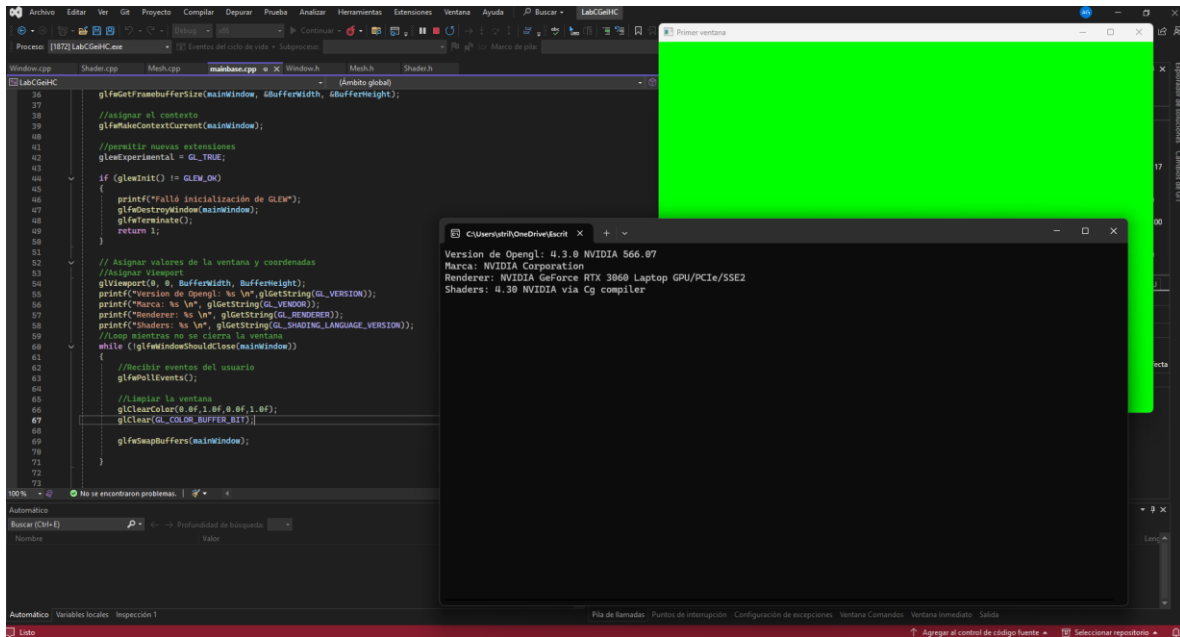
SEMESTRE 2026 - 1

FECHA DE ENTREGA LÍMITE: 24 de agosto del 2025

CALIFICACIÓN: _____

Respuestas previo 1:

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 (no es necesario que se imprima para la entrega del previo a mano)



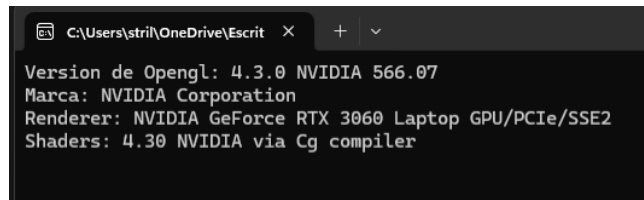
Texto:

Version de Opengl: 4.3.0 NVIDIA 566.07

Marca: NVIDIA Corporation

*Renderer: NVIDIA GeForce RTX 3060
Laptop GPU/PCIe/SSE2*

Shaders: 4.30 NVIDIA via Cg compiler



2. ¿Que es un VAO?

R. Bajo el contexto de Open GL, definimos a VAO (Vertex Array Object), que es un objeto encargado de guardar la configuración de los datos contenidos en los vertices desde los buffers. Almacena datos como: Posición, Normales, coordenadas, color, (Etc)

3. Que es un VBO?

R. Vertex Buffer object es un buffer en la memoria de la GPU donde se guardan los datos de los vertices.

4. Que parametros recibe el comando `glVertexAttribPointer`?

R. Interpreta los datos de un atributo de vertice dentro de un buffer de vertices y recibe los siguientes Parametros: `index`, `size`, `type`, `normalized`, `stride` y `Pointer`.

• `Size (GLint)`: Número de componentes por atributo de vertice (1-4)

• `Type (GLenum)`: Tipo de dato de cada componente del atributo.

• `Normalized (GLboolean)`: Valor Booleano que indica si los datos fijos deben ser normalizados o no.

• `Stride (GLsizei)`: Desplazamiento en bytes entre atributos de vertice consecutivo.

• `Pointer (const void)`: Puntero o desplazamiento del primer componente del primer atributo vertice

5. ¿Que información maneja Vertex Shader?

R. Maneja y transforma los datos de los vertices de una malla 3D para calcular su posición final en pantalla, definir sus atributos (como color y texturas) y prepararlos para ser etapas posteriores como el pipeline grafico. Recibe información de cada vertice como coordenadas (posición), texturas (UV), colores y vectores normales.

6. ¿Que información maneja Fragment Shader?

R. Maneja información sobre la geometría y propiedades de cada pixel a través de datos interpolados desde la etapa de vertices, como su posición en pantalla y las texturas asignadas. Su función principal es calcular el color y el valor de profundidad de ese pixel, basándose en la iluminación, la configuración del material y las texturas.

7. ¿Que parametros recibe el comando `glDrawArrays`?

R. Recibe tres parametros:

• `Mode (GLenum)`: Determina el tipo de primitiva geometrica que va a construir como líneas, puntos y triángulos.

• `first (GLint)`: Es el índice del primer elemento en los arrays de vertices para construir primitivas

• `Count (GLsizei)`: Especifica el número de elementos de los arrays que se utilizaron para crear primitivas.

8. ¿Qué son las variables Uniform dentro de GLSL y como se declaran y se mandan desde OpenGL a GLSL?

R. Son parámetros globales de un programa shader que permanecen constantes durante una pasada de renderizado, pasándose desde la aplicación (CPU) al shader (GPU) para controlar aspectos como el color, la transformación o el tiempo. Se declaran en el shader con la palabra clave `uniform` seguida del tipo de dato. Y se pasan desde OpenGL obteniendo su ubicación usando `glGetUniformLocation` y luego asignando un valor con funciones como `glUniform4f`. Después de haber ligado el programa.

9. Proyecciones planares por medio de glm. (matriz y línea de código)

R. Se refiere al uso de técnicas de modelado lineal general (GLM), como regresión lineal para predecir coordenadas en un plano a partir de puntos bidimensionales, similar a como se proyectan las vistas en el dibujo técnico.

En OpenGL lo podemos definir como:

```
glMatrixMode(GL_Projection);  
glLoadIdentity();  
glOrtho(x_min, x_max, y_min, y_max);
```

10. Matrices de transformación de traslación, rotación y escala y con glm.

1. Traslación. `glm::mat4 traslación = glm::translate(glm::mat4(1.0f), glm::vec3(1.0f, 2.0f, 3.0f));`
Es una matriz de traslación de 4x4 que desplaza un objeto. Para unido tx, ty, tz

2. Rotación: Se puede realizar alrededor de los ejes x, y o z mediante un vector arbitrario.

```
glm::mat4 rotación = glm::rotate(glm::mat4(1.0f), glm::radians(45.0f), glm::vec3(0.0f, 1.0f, 0.0f));
```

rotación de 45° en y

3. Escala modifica el tamaño de un objeto (uniformemente o factores de cada eje)

```
glm::mat4 escala = glm::scale(glm::mat4(1.0f), glm::vec3(2.0f, 0.5f, 1.0f));
```

agrandar el objeto 2 veces en x , 0.5 en y y 1 en z .

Referencias

- Computer graphics stack exchange. (n.d.). Computer Graphics Stack Exchange. Retrieved Agosto 25, 2025, from <https://computergraphics.stackexchange.com/questions/10332/understanding-vao-and-vbo>
- de la Computación e Inteligencia Artificial, C. (n.d.). Dibujando en el espacio. Uhu.Es. Retrieved Agosto 25, 2025, from https://www.uhu.es/francisco.moreno/gii_rv/docs/Tema_5.pdf
- Función `glDrawArrays`. (n.d.). Microsoft.com. Retrieved Agosto 25, 2025, from <https://learn.microsoft.com/es-es/windows/win32/opengl/glDrawArrays>

- *Shader basics - vertex shader*. (n.d.). Shader-tutorial.dev. Retrieved Agosto 25, 2025, from <https://shader-tutorial.dev/basics/vertex-shader/>
- Unity Technologies. (n.d.). *Ejemplos del Vertex y Fragment Shader*. Unity3d.com. Retrieved Agosto 25, 2025, from <https://docs.unity3d.com/es/2018.4/Manual/SL-VertexFragmentShaderExamples.html>
- *Vertex Buffer Objects*. (n.d.). Antongerdelan.net. Retrieved Agosto 25, 2025, from <https://antongerdelan.net/opengl/vertexbuffers.html>
- (N.d.). Umich.Mx. Retrieved Agosto 25, 2025, from https://lc.fie.umich.mx/~rochoa/Materias/LABORATORIOS/GRAFICACION/GRAF_02.pdf