



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

**Nº de Cuenta:** 317031405

**GRUPO DE LABORATORIO:** 3

**GRUPO DE TEORÍA:** 5

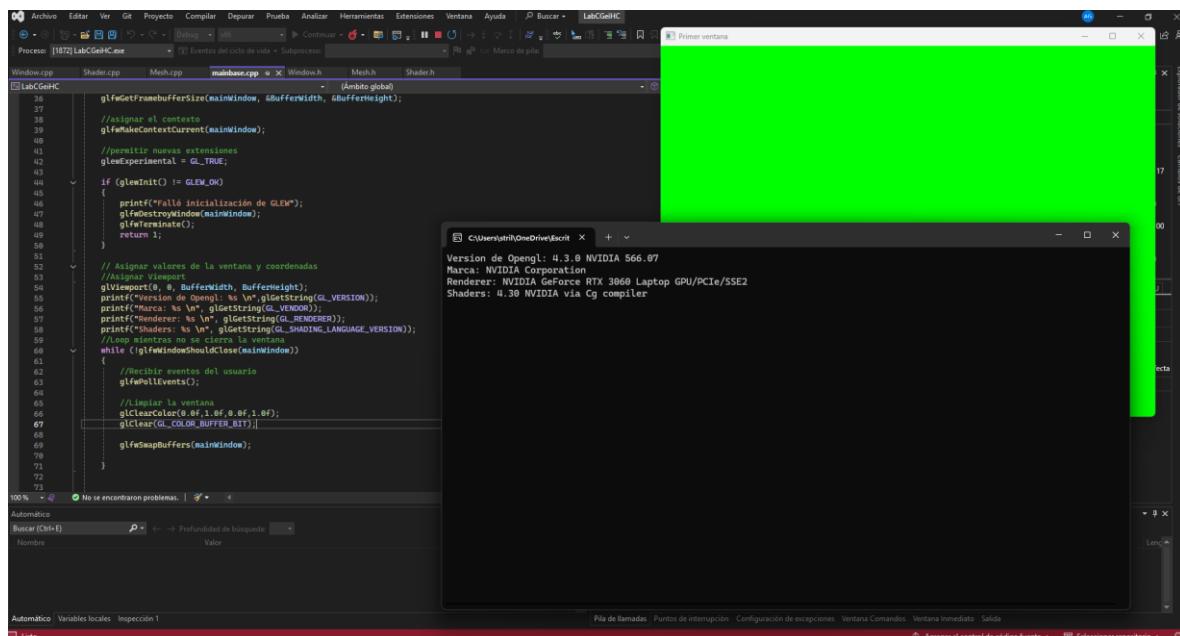
**SEMESTRE 2026 - 1**

**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)



The screenshot shows a Windows-based IDE with multiple tabs open. The main code editor tab contains C++ code for initializing OpenGL and setting up a window. A terminal window to the right displays the output of OpenGL version information. A green rectangular window is visible in the background.

```
36     glfwGetFramebufferSize(mainWindow, &bufferWidth, &bufferHeight);
37
38     // Asignar el contexto
39     glfwMakeContextCurrent(mainWindow);
40
41     // Permitir nuevas extensiones
42     glewExperimental = GL_TRUE;
43
44     if (glewInit() != GLEW_OK)
45     {
46         printf("Falló inicialización de GLEW");
47         glfwDestroyWindow(mainWindow);
48         glfwTerminate();
49         return 1;
50     }
51
52     // Asignar valores de la ventana y coordenadas
53     // Significado de los argumentos:
54     glfwSetWindowPos(0, BufferWidth, BufferHeight);
55     printf("Version de OpenGl: %s\n", glGetString(GL_VERSION));
56     printf("Marca: %s\n", glGetString(GL_VENDOR));
57     printf("Renderer: %s\n", glGetString(GL_RENDERER));
58     printf("Shaders: %s\n", glGetString(GL_SHADING_LANGUAGE_VERSION));
59
60     // Loop mientras no se cierra la ventana
61     while (!glfwWindowShouldClose(mainWindow))
62     {
63         // Recibir eventos del usuario
64         glfwPollEvents();
65
66         // Actualizar la ventana
67         glClearColor(0.8f, 0.8f, 0.8f, 1.0f);
68         glClear(GL_COLOR_BUFFER_BIT);
69
70         glfwSwapBuffers(mainWindow);
71     }
72 }
```

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

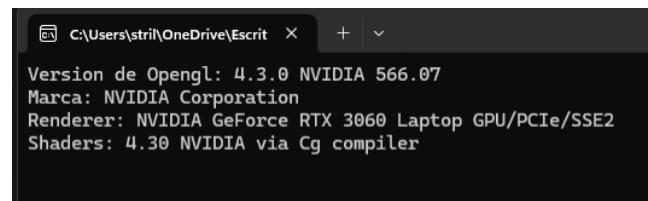
### Texto:

*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*



A terminal window showing the same OpenGL configuration details as the previous screenshot, including the version, vendor, renderer, and shaders.

```
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. ¿Qué es un VAO?

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

## 3. ¿Qué es un VBO?

R: Vertex Buffer Object es un buffer en la memoria de la GPU donde se guardan los datos de los vértices.

## 4. ¿Qué parámetros recibe el comando glVertexAttribPointer?

R: Interpreta los datos de un atributo de vértice dentro de un buffer de vértices y recibe los siguientes parámetros: index, size, type, normalized, stride y pointer.

• Size (GLint): Número de componentes por atributo de vértice (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 vértice consecutivos.

• Pointer (const void\*): Puntero o desplazamiento del primer componente del primer vértice

## 5. ¿Qué información maneja Vertex Shader?

R: Maneja y transforma los datos de los vértices 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 gráfico. Recibe información de cada vértice como coordenadas (posición), texturas (UV), colores y vectores normales.

## 6. ¿Qué información maneja Fragment Shader?

R: Maneja información sobre la geometría y propiedades de cada pixel a través de datos interpolados desde la etapa de vértices, 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. ¿Qué parámetros recibe el comando glDrawArrays?

R: Recibe tres parámetros:

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

• first (GLint): Es el índice del primer elemento en los arrays de vértices para constituir primitivas.

• Count (GLsizei): Especifica el número de elementos de los arrays que se utilizarán 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 glUniformNPF, después de haber ligado el programa.

9. Proyecciones planares por medio de glm (matriz y linea 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 tridimensionales, 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:  $\text{glm}::\text{mat4 traslación} = \text{glm}::\text{translate}(\text{glm}::\text{mat4}(4.0f), \text{glm}::\text{vec3}(1.0f, 2.0f, 3.0f))$ , Es una matriz de traslación de  $4 \times 4$  que desplaza un objeto. Poco visto 4X, 2Y, 3Z

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  $45^\circ$  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](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](https://lc.fie.umich.mx/~rochoa/Materias/LABORATORIOS/GRAFICACION/GRAF_02.pdf)