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: Pérez Uribe José Alberto

Nº de Cuenta: 318143213

GRUPO DE LABORATORIO: 02

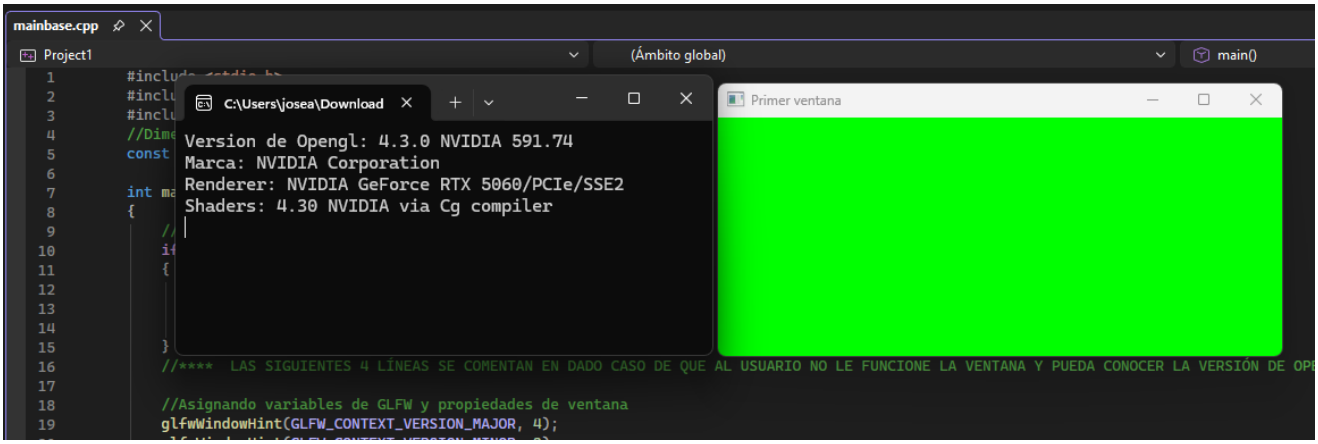
GRUPO DE TEORÍA: 07

SEMESTRE 2026-2

FECHA DE ENTREGA LÍMITE: 14/02/2026

CALIFICACIÓN: _____

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)



2. ¿Qué es un VAO?
Un Vertex Array Object es un objeto que almacena la configuración del estado de los atributos de los vértices.
En OpenGL no se puede dibujar nada sin un VAO activo
Un VAO almacena:
- Qué VBO está asociado a cada atributo
- Cómo se interpretan los datos del VBO
- Qué atributos están habilitados

* El VAO no guarda los datos, guarda la forma en que OpenGL debe leerlos

3. ¿Qué es un VBO?
Un Vertex Buffer Object) es un buffer en memoria de la GPU que almacena los datos de los vértices.
* Antes de los VBO los datos se enviaban cada frame desde CPU-GPU
Con VBO los datos se suben una vez y se reutilizan.
¿Que puede almacenar?
- Posiciones (vec3)
- Colores (vec3 o 4)
- Coordenadas de textura (vec2)
- Normales (vec3)
- Cualquier atributo personalizado

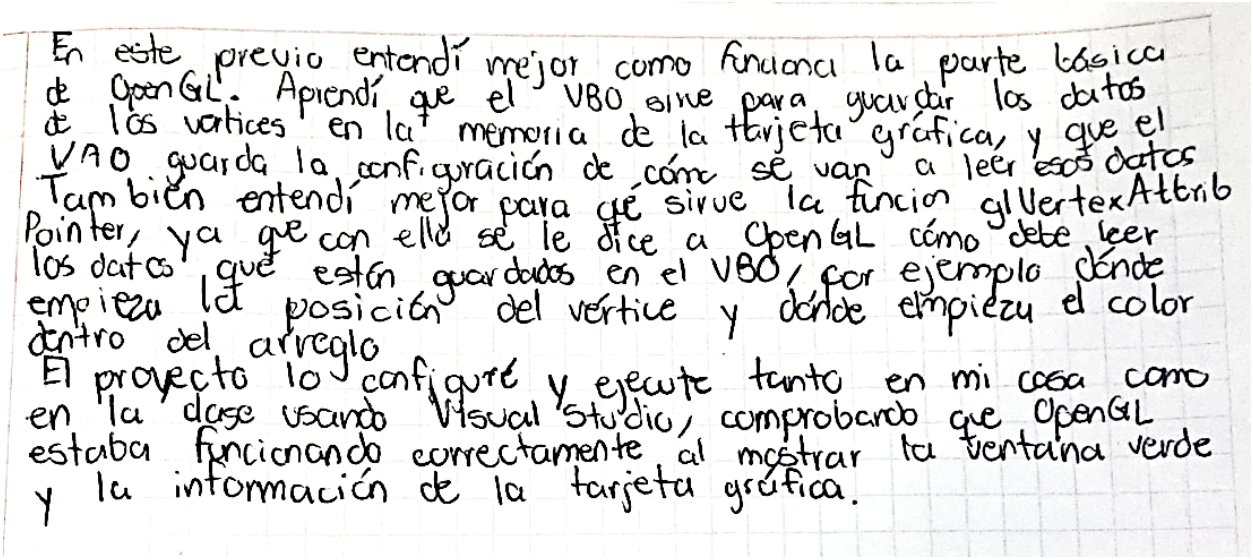
1. ¿Qué parámetros recibe el comando glVertexAttribPointer?
Recibe seis parametros para definir cómo interpretar los datos de vértices almacenados en un buffer (VBO)
- GLuint index: El indice del atributo de vertice que se va configurar
- GLint size: El número de componentes por atributo. Puede ser 1, 2, 3 o 4 (Ej. 3 para coordenadas XYZ)
- GLenum type: El tipo de datos de cada componente, como GL_FLOAT, GL_INT, GL_UNSIGNED_BYTE, etc..
- GLboolean normalized: Indica si los datos de punto fijo deben normalizarse (GL_true) o convertirse directamente a valores de punto flotante (GL_False).
- GLsizei stride: El desplazamiento en bytes entre el inicio de un atributo de vértice y el siguiente. Si es 0, se asume que los atributos están estrechamente empaquetados.
- const void *pointer: El desplazamiento (offset) inicial del primer componente del primer atributo en el buffer, expresado como un puntero o valor entero. Comunmente 0 si los datos inician al principio del buffer.

5. ¿Qué información maneja el Vertex Shader?
procesa los atributos de cada vertice individualmente, transformando coordenadas de posición del espacio local al espacio de recorte (clip space) usando matrices (MVP) y manejando datos como normales, coordenadas de textura (UVs) y colores de vértices.
Su función principal es transformar el vértice a coordenadas de pantalla.
Es la primera etapa programable en la GPU, especial para posicionar figuras, aplicar efectos de iluminación básicos y preparar datos para la resterización.

6. ¿Qué información maneja Fragment Shader?
El sombreador de fragmentos es la etapa programable de la GPU encargada de determinada el color final y la profundidad de cada fragmento que compone una primitiva
La información que maneja se divide principalmente en datos de entrada, texturas y la generación de salidas de color/profundidad.
Se ejecuta para cada fragmento generado después del rasterizado, determinando el color final del pixel

7. ¿Qué parámetros recibe el comando glDrawArrays?
La función es actuar como el disparador final en el flujo de renderizado de OpenGL. Su propósito es tomar los datos almacenados en el VertexBuffer Object (VBO) actualmente activo y transformarlo en imágenes en pantalla, a diferencia de otras funciones de dibujo que utilizan índices para reutilizar vértices glDrawArrays asume que la información está organizada de forma secuencial, por lo que recorre los arreglos desde el índice inicial especificado hasta cubrir el número total de vértices indicados, aplicando la topología definida en el modo de primitiva

Conclusión


En este previo entendí mejor como funciona la parte básica de OpenGL. Aprendí que el VBO sirve para guardar los datos de los vértices en la memoria de la tarjeta gráfica, y que el VAO guarda la configuración de cómo se van a leer esos datos. También entendí mejor para qué sirve la función glVertexAttribPointer, ya que con ella se le dice a OpenGL cómo debe leer los datos que están guardados en el VBO, por ejemplo dónde empieza la posición del vértice y dónde empieza el color dentro del arreglo.
El proyecto lo configuré y ejecuté tanto en mi casa como en la dase usando Visual Studio, comprobando que OpenGL estaba funcionando correctamente al mostrar la ventana verde y la información de la tarjeta gráfica.

Bibliografía

Stevewhims. (n.d.). *Función glVertexPointer (Gl.h) - Win32 apps*. Microsoft Learn. https://learn.microsoft.com/es-es/windows/win32/opengl/glvertexpointer

Guía Definitiva de VBOs, VAOs y EBOs en OpenGL. (n.d.). https://www.q2bstudio.com/nuestro-blog/21151/guia-definitiva-de-vbos-vaos-y-ebos-en-opengl?scriptscookies=1

Rogrp. (2020, June 3). *OpenGL – básico (C++)*. https://regor.home.blog/2020/05/21/opengl-basico-c/

Shader Basics - Vertex Shader | GPU Shader Tutorial. (n.d.). https://shader-tutorial.dev/basics/vertex-shader/#2-what-is-a-vertex-shader

Fragment Shader - OpenGL Wiki. (n.d.). https://wikis.khronos.org/opengl/Fragment_Shader

GlDrawArrays - OpenGL 4 - docs.gl. (n.d.). https://docs.gl/gl4/glDrawArrays