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: Romero Dominguez Ricardo Damian

Nº de Cuenta: 319094493

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

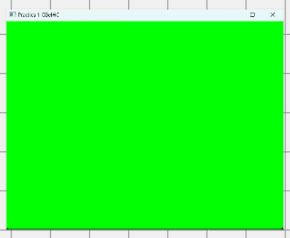
FECHA DE ENTREGA LÍMITE: 2/15/2025

CALIFICACIÓN: _____

1. Contenido requerido (A mano y digitalizado escaneado/fotografiado)
2. Conclusión (A mano y digitalizado escaneado/fotografiado)
3. Bibliografía en formato APA (digital)

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)



2.-¿Qué es un VAO?

Un Vertex Array Object (Objeto de matriz de vértices) tiene como objetivo almacenar la información de un objeto que ya fue renderizado guardando objetos de búffer de vertices

3.- ¿Qué es un VBO?

Un Vertex Buffer Object (Objeto de Buffer de vértices), como su nombre lo indica, es un búffer que almacena información sobre los vértices de un objeto

4.- ¿Qué parámetros recibe el comando glVertexAttribPointer?

- index: Especifica el índice del atributo genérico del vértice que será modificado
- size: Especifica el número de componentes por atributo genérico. Debe ser 1, 2, 3, 4. Adicionalmente la constante GL_BGRA es aceptada. El valor inicial es 4.
- type: Especifica el tipo de dato de cada componente en el arreglo. Las constantes simbólicas:
  - GL_BYTE
  - GL_UNSIGNED_BYTE
  - GL_UNSIGNED_SHORT
  - GL_SHORT
  - GL_INT
  - GL_UNSIGNED_INT
  - GL_HALF_FLOAT
  - GL_FLOAT
  - GL_DOUBLE
  - GL_FIXED
  - GL_INT_2_10_10_10_REV
  - GL_UNSIGNED_INT_2_10_10_10_REV
  - GL_UNSIGNED_INT_10F_11F_11F_REV

  son aceptadas. Adicionalmente, el valor inicial es GL_FLOAT
- normalized: Especifica si los valores de datos fixed-point deben normalizarse (GL_TRUE) o convertirse directamente en valores fixed-point (GL_FALSE) cuando son accesados
- stride: Especifica el byte de corrimiento entre atributos genéricos de vértice. Si es 0, los atributos genéricos de vértice están consecuentemente almacenados en el arreglo. Su valor inicial es 0
- pointer: Especifica un desplazamiento de la primera componente del atributo genérico del vértice en el arreglo almacenado en el búffer vinculado a GL_ARRAY_BUFFER

5.- ¿Qué información maneja Vertex Shader?

Se encarga del procesamiento de vértices individuales, por lo que es invocado en la fase de renderizado, recibiendo un único vértice del flujo de vértices, y generando un único vértice a la salida del flujo

6.- ¿Qué información maneja Fragment Shader?

Esta etapa procesa fragmentos generados por la rasterización y se genera una serie de colores junto a un único valor de profundidad

7.- ¿Qué parámetros recibe el comando glDrawArrays?

- mode: Especifica qué tipo de primitivas renderizar. Acepta las constantes
  - GL_POINTS
  - GL_LINE_STRIP
  - GL_LINE_LOOP
  - GL_LINES
  - GL_LINE_STRIP_ADJACENCY
  - GL_LINES_ADJACENCY
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP_ADJACENCY
  - GL_TRIANGLES_ADJACENCY
  - GL_PATCHES

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

Son variables locales de tipo Shader que se deben definir en GLSL, pudiendo ser de cualquier tipo, como por ejemplo:

```
Struct ejemplo{
   vec4 vector_1;
   vec3 vector_2;
};
Uniform vec3 vector_3;
Uniform ejemplo variable_1;
Uniform mat4x3 matriz_1;
```

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

ORTOGRÁFICA

$$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glm::ortho(left, right, bottom, top, near, far)

PERSPECTIVA

$$\begin{bmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f+n} & -\frac{2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

glm::frustum(left, right, bottom, top, near, far)

10. Matrices de transformación de Traslación, Rotación y Escala y con glm.

TRASLACION $\vec{J}=(10,10,10,1)$ 10 pts en X

```
#include <glm/gtx/transform.hpp>
glm::mat4 myMatriz = glm::translate(glm::mat4(), glm::vec3(10.0f, 0.0f, 0.0f));
glm::vec4 myVector(10.0f, 10.0f, 10.0f, 0.0f);
glm::vec4 transformedVector = myMatriz * myVector
```

ESCALA

```
glm::mat4 myScalingMatrix = glm::scale(2.0f, 2.0f, 2.0f);
```

ROTACIÓN

```
glm::vec3 myRotationAxis(??, ??, ??);
glm::rotate(angle_degrees, myRotationAxis);
```


Referencias

1. Tutorial de OpenGL (VAOs, VBOs, Shaders):
Khronos Group. (n.d.). Tutorial 2: VAOs, VBOs, Vertex and Fragment Shaders (C/SDL). OpenGL Wiki. Recuperado el 10 de mayo de 2024, de https://www.khronos.org/opengl/wiki/Tutorial2:_VAOs,_VBOs,_Vertex_and_Fragment_Shaders_(C_/_SDL)

2. Documentación de glVertexAttribPointer:
Khronos Group. (n.d.). glVertexAttribPointer. Registry.Khronos.org. Recuperado el 10 de mayo de 2024, de https://registry.khronos.org/OpenGL-Refpages/gl4/html/glVertexAttribPointer.xhtml

3. Vertex Shader (OpenGL Wiki):
Khronos Group. (n.d.). Vertex Shader. OpenGL Wiki. Recuperado el 10 de mayo de 2024, de https://www.khronos.org/opengl/wiki/Vertex_Shader

4. Rasterization (OpenGL Wiki):
Khronos Group. (n.d.). Rasterization. OpenGL Wiki. Recuperado el 10 de mayo de 2024, de https://www.khronos.org/opengl/wiki/Rasterization

5. Documentación de glDrawArrays:
Khronos Group. (n.d.). glDrawArrays. Registry.Khronos.org. Recuperado el 10 de mayo de 2024, de https://registry.khronos.org/OpenGL-Refpages/gl4/html/glDrawArrays.xhtml

6. Uniform (GLSL):
Khronos Group. (n.d.). Uniform (GLSL). OpenGL Wiki. Recuperado el 10 de mayo de 2024, de https://www.khronos.org/opengl/wiki/Uniform_(GLSL)

7. Apuntes de clase (Universidad de Huelva):
Moreno, F. (n.d.). Tema 5: Transformaciones geométricas y proyecciones. Universidad de Huelva. Recuperado el 10 de mayo de 2024, de https://www.uhu.es/francisco.moreno/gii_rv/docs/Tema_5.pdf

8. Tutorial de matrices en OpenGL:
OpenGL Tutorial. (n.d.). Tutorial 3: Matrices. OpenGL Tutorial. Recuperado el 10 de mayo de 2024, de https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/