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 02

NOMBRE COMPLETO: Hernández Vázquez Daniela

Nº de Cuenta: 318092867

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 02

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 17 de febrero de 2024

CALIFICACIÓN: _____

1.- ¿Como funciona la cámara sintética glm::LookAt?

Un modelo de cámara sintética debe tener información de:

- Posición — Campo de visión (ángulo de ancho, no mil)
- Orientación — Profundidad de campo (Plano cercano y lejano)
- Distancia focal — Desviación de planos vista/película (proyecciones)
- Desviación perspectiva o paralela (distancia de objetos)

glm::LookAt es una función de la biblioteca glm que se utiliza para ubicar la cámara construyendo una matriz de vista en OpenGL. La función recibe 3 parámetros:

- eye (posición de la cámara en el escenario)
- center/target (punto al cual mira la cámara)
- up (vector normal al espacio, dirección vertical de la cámara).

Con estos parámetros la función calcula y devuelve una matriz de vista que transforma los objetos desde su posición y orientación en el mundo al espacio de vista de la cámara.

2.- ¿Como funciona la matriz de vista en el shader? (Versión moderna de OpenGL)

La matriz de vista en el shader se utiliza para transformar los vértices de un objeto desde un sistema de coordenadas de la cámara. Esto simula el movimiento y la posición de la cámara en la escena.

La matriz de vista en el shader se utiliza en conjunto con otras matrices: la matriz de modelo (representa la posición y orientación de los objetos en el mundo) y la matriz de proyección (define la perspectiva de la cámara. Con esto se logra obtener la posición final de la pantalla.

$$\vec{V}_{final} = M_{proyeccion} \cdot M_{vista} \cdot M_{modelo} \cdot \vec{v}$$

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

Las variables uniformes (uniform) en GLSL son variables que tienen un valor común en todas las ejecuciones del shader.

Estas variables suelen ser utilizadas para almacenar matrices de transformación de coordenadas, valores de luz, texturas, etc. Son declaradas en los shaders y asignadas mediante comandos glUniform*(...)

Su declaración general es: Uniform tipo-de-dato Nombre;

Para asignar estos valores al shader con GLSL se usa la función:

glUniformX tipo-de-dato (parametros) // X = tipo de instrucción gl

4.- ¿Como funciona la variable extern?

La variable extern es declarada en otro archivo como el shader, con esto podemos acceder para modificar su valor en el archivo principal.

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

Aunque la salida del VertexShader debe de estar dentro en coordenadas homogéneas, lo normal es que la posición de los vértices se dentro en el sistema de coordenadas del modelo. Por tanto, el programa del VertexShader debe incorporar una transformación entre un sistema de coordenadas y otro (proyección)

▷ Proyección Ortográfica.

Consiste en transformar un volumen de forma rectangular en el volumen clip (normalizado) o cubo de escala en x, y, z.

La transformación es sencilla pero provoca que los objetos se vean igual tanto lejos como cerca

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

ejex ejey ejez

▷ Proyección Perspectiva

Transforma el espacio que ve el observador con un cierto angulo de visión (fov-field of vista) en el volumen clip.

Provoca que los objetos mas cercanos se vean demasiado mas grandes que los objetos lejanos (frustum).

ejex ejey ejez

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

glm::perspective (fovy, aspect, near, far)

ángulo de visión relación alto/ancho profundidad

6.- Matrices de transformación de Traslación, Rotación y Escala con glm.

Las transformaciones se realizan en dos pasos. En primer lugar las posiciones de todos los objetos se transforman a un sistema de coordenadas fijo del modelo. De esta forma las posiciones quedan expresadas en un mismo sistema de coordenadas.

Para transformar las coordenadas locales en coordenadas de modelo se utiliza una matriz de transformación conocida como Model. Transforma estas coordenadas de modelo al sistema de coordenadas del observador usa una matriz de transformación View. (Model - View - Projection)

- ▷ Traslación: Mover un objeto en el espacio
  `glm::translate(glm::mat4 M, glm::vec3 disp)`
- ▷ Rotación: Rotas un objeto al rededor de un punto de origen
  `glm::rotate(glm::mat4 M, float angle, glm::vec3 axis)`
- ▷ Escala: Cambia el tamaño en cada eje.
  `glm::scale(glm::mat4 M, glm::vec3 scale)`

Referencias:

Dpto. de Ciencias e Ingeniería de la Computación Universidad Nacional del Sur, Castro, S., & Urribarri, D. (2015). *Escenas 3D* [Diapositivas; Diapositivas web]. www.cs.uns.edu.ar. http://www.cs.uns.edu.ar/cg/clasespdf/3-Pipe3D.pdf

Kessenich, J. M., Sellers, G., & Shreiner, D. (2016). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V*. OpenGL.

Méndez Servín, M. (2022). *NOTAS PARA EL CURSO DE GRAFICACIÓN POR COMPUTADORA* [Libro electrónico]. https://prometeo.matem.unam.mx/recursos/VariosNiveles/iCartesiLibri/recursos/Notas_Graficacion_por_Computadora/index.html

Moreno V., J. (2023). *Tema 4 y Tema 5* [Diapositivas; Presentación electrónica]. www.uhu.es. https://www.uhu.es/francisco.moreno/gii_rv/

Sellers, G., Wright, R. S., & Haemel, N. (2015). *OpenGL superbible: Comprehensive Tutorial and Reference*. Addison-Wesley Professional.

Tutorial 3 : Matrices. (2014). https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/#matrices-modelo-vista-y-proyecci%C3%B3n