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 2

NOMBRE COMPLETO: Resendiz Casas Axel Ariel

Nº de Cuenta: 316208516

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 4

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 24/08/2024

CALIFICACIÓN: _____

Reporte Previo 2

D 21 M 08 A 24

1.- ¿Cómo funciona la cámara sintética glm::LookAt)
- Un modelo de cámara sintética debe tener información de:
  - Posición
  - Orientación
  - Campo de visión (ángulo de ancho, normal...)
  - Profundidad de campo (Plano cercano y lejano)
  - Distancia focal
  - Desviación de los planos vista/película (si no es normal a la dirección de vista produce proyecciones oblícuas)
  - Proyección perspectiva o paralela (cámara cerca de los objetos o a distancia infinita).

Dicho esto, The glm::LookAt Function requires a position, target and up vector respectively.
La posición de la cámara (desde donde se está mirando) El vector LookAt (mirando hacia) especifica en que dirección apunta la cámara la orienbación de la cámara está determinada por el vector LookAt y el ángulo en el que está rotada la cámara con respecto a ese vector, es decir, la dirección del vector up.

2.- ¿Cómo funciona la matriz de vista en el shader)
- La matriz de vista define la posición y orientación de la cámara respecto al mundo virtual, Modificando sus valores, podemos alterar la posición o dirección de la cámara virtual (o lo que es lo mismo, alteramos el centro y el plano de proyección respecto al sistema de referencia del mundo).

Ejemplo: Si queremos ver una montaña desde otro ángulo, podemos mover la cámara ... o mover la montaña. No es práctico en la vida real, es todo que así es más fácil en computación gráfica.
Inicialmente la cámara está en el origen del espacio mundo.
Para mover el mundo, simplemente se introduce otra matriz
Digamos que queremos mover nuestra cámara 3 unidades a la

D 21 M 08 A 24

derecha (+x). Esto es equivalente a mover todo el mundo (y lo que contenga) 3 unidades a la izquierda.

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

"A uniform is a global shader variable declared with the uniform storage qualifier. These act as parameters that the user of a shader program can pass to that program. Their values are stored in a program object.

Uniforms are so named because they do not change from one shader invocation to the next within a particular rendering call thus their value from is uniform among all invocations. This make them unlike shader stage inputs and outputs, wich are often different for each invocation of a shader stage. Uniform variables must be defined in GLSL at global scope.

Uniforms can be of any type, or any aggregation of types. The following are all legal GLSL code:

```
struct TheStruct
{
    vec3 first;
    vec4 second;
    mat4x3 third;
};
uniform vec3 oneUniform;
uniform TheStruct aUniformOfArrayType;
Uniform mat4 matrixArrayUniform[25];
Uniform TheStruct uniformArrayOfStructs[10];
```

Uniforms are implicity constant, within the shader (though they are not Constant Expressions). Attempting to change them with shader code will result in a compiler error. Similary, you cannot pass a uniform as an out or inout parameter to Function.

PS BD IS

D 21 M 08 A 24

Uniforms are intended to be set by the user from OpenGL, rather than within the shader. However, you can initialize them to a default value using standard GLSL initializer syntax:

```
Uniform vec3 initialUniform = Vec3 (1.0, 0.0, 0.0);
```

4.- ¿Cómo funciona la variable extern?

La palabra clave extern se puede aplicar a una declaración de variable, función o plantilla global. Especifica que el símbolo tiene externa vinculación.

La palabra clave extern tiene cuatro significados en función del contexto:

- En una declaración de variable no global const, extern especifica que la variable o función se define en otra unidad de traducción, extern debe aplicarse en todos los archivos excepto en el que se define la variable.
- En una declaración de variable const, especifica que la variable o función se define tiene una vinculación extern. extern debe aplicarse a todas las declaraciones de todos los archivos (las variables globales const tienen vinculación interna de forma determinada)
- extern "C" especifica que la función se define en otro lugar y usa la convención de llamada del lenguaje C. El modificador extern "C" también se puede aplicar a varias declaraciones de función en un bloque.
- En una declaración de plantilla, extern especifica que ya se ha creado una instancia de la plantilla en otro lugar. extern indica al compilador que puede reutilizar la otra creación de instancias en lugar de crear una nueva en la ubicación actual.

D 21 M 08 A 24

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

- Proyección ortográfica Consiste en transformar un volumen de forma rectangular en el volumen clip. Esta supone un cambio de escala en las diferentes coordenadas (x, y, z) para adaptarse al sistema de coordenadas homogéneo. La transformación provoca que los objetos se vean del mismo tamaño tanto si están cerca como si están alejados

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

Si el clipping volume es simétrico ($r = -l$ y $t = b$) la matriz queda

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

Se puede generar la matriz de proyección utilizando la función de la librería GLM: glm::ortho(left, right, bottom up, top, near, far)

- Proyección en Perspectiva, consiste en transformar el espacio que se ve desde el observador con un cierto ángulo de visión (fov-field of vision) en el volumen clip. Esta transformación provoca que los objetos más cercanos se vean de mayor tamaño que los objetos más lejanos.

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

D 21 M 08 A 24

Si el volumen clip es simétrico (r = -l y t = -b)

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

Está matriz se puede generar con funciones de la librería GLM:

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

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

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

- Matriz de Traslación

```
#include <glm/gtx/transform.hpp>
glm::mat4 myMatrix = 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 = myMatrix * myVector;
```

- Matriz de Rotación

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

Conclusión

Me parece sumamente interesante ver la relación de la materia de Álgebra Lineal (de los primeros semestres) con este tipo de programas que estamos estudiando, sinceramente no tenía idea de todo lo que implica la creación de un juego, animación, etc. Me doy cuenta de que las matrices son muy importantes y de mucha ayuda para la realización de nuestros proyectos, ya que más adelante nos ayudarán a crear cámaras y replicar sus efectos de manera virtual.

Bibliografía

OpenGL. Learn OpenGL - Camera. Consultado el 20 de Agosto 2024. https://learnopengl.com/Getting-started/Camera

Castro S. & Urribarri D.(2015). Escenas 3D. Consultado el 20 de Agosto 2024. http://www.cs.uns.edu.ar/cg/clasespdf/3-Pipe3D.pdf

OpenGL. Tutorial 3: Matrices. Consultado el 20 de agosto de 2024. https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/

Universidad de Valencia. Práctica 5 Transformada de vista y transformada de proyección. Consultado el 20 de agosto de 2024. https://informatica.uv.es/iiguia/IG/prac5.pdf

OpenGL (2020). Uniform(GLSL). Consultado el 20 de agosto de 2024. https://www.khronos.org/opengl/wiki/Uniform_(GLSL)

Microsoft Learn (2023). extern (C++). Consultado el 20 de agosto de 2024. https://learn.microsoft.com/es-es/cpp/cpp/extern-cpp?view=msvc-170

Moreno F. (). Tema 5 Dibujando en el espacio. Consultado el 20 de agosto 2024. https://www.uhu.es/francisco.moreno/gii_rv/docs/Tema_5.pdf