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: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: jueves 15 de febrero de 2024

CALIFICACIÓN: _____

Cuestionario Previo 2ª. 13/Feb/24

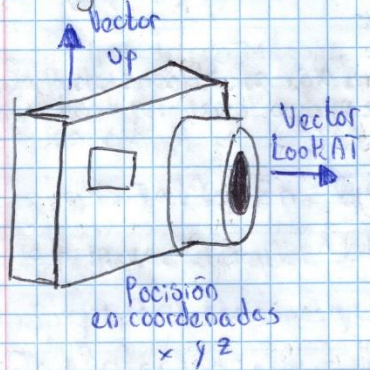
1-. ¿Cómo funciona la cámara sintética glm::LookAt?

Un modelo de cámara sintética debe tener la sig. info:

- Posición - Orientación - Campo de visión
- Profundidad de campo - Distancia focal - Desviación de los planos vista/película
- Proyección perspectiva o paralela

Y para definir el modelo debemos de tener:

* La posición de la cámara (desde donde mira)
* El vector LookAt, que especifica en que dirección apunta la cámara
* La orientación de la cámara, definida por el vector LookAt, y el ángulo de su rotación, definida por el vector Up



- glm::LookAt() se encarga de especificar la ubicación de la cámara, el viewing

2-. ¿Cómo funciona la matriz de vista en el shader (OpenGL 3.3 en adelante)?

Es una matriz que permite mover todos los elementos de un espacio mundo hacia otra posición, para que la cámara apunte a estos elementos y pueda visualizarlos

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

Son variables que tienen un valor común en todas las ejecuciones del shader. Suelen utilizarse para almacenar matrices de transformación de coordenadas

Para declararlos se usa la sintaxis:

uniform tipodedato nombredelavariable;

OpenGL tiene varias funciones para obtener los valores de variables, por ejemplo:

- glGetUniformLocation ("identificador del programa", "nombre de la variable uniform);

4-. ¿Cómo funciona la variable Extern?

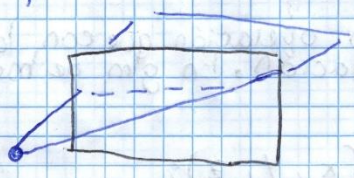
Son variables con almacenamiento permanente. Todas las funciones y bloques declarados después de una variable externa podrán acceder a ella. Con estas variables se puede transferir información entre funciones sin pasar argumentos
Para definirlas, se hace de la misma forma que una variable común, haciendo uso de la palabla extern para el tipo de almacenamiento

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

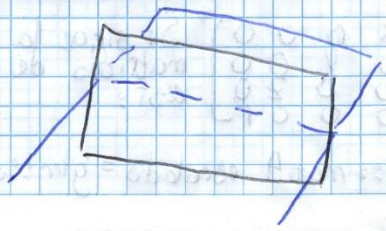
Una proyección transforma puntos de un sistema de coordenadas de dimensión N, a uno de dimensión menor que N.
Hay 2 principales proyecciones geométricas planas:

* Perspectiva: Determinada por el Centro de Proyección (COP)
* Paralela: Determinada por la dirección de proyección (DDP)

Perspectiva



Paralela



Para las proyecciones de perspectiva se utiliza una transformación lineal y una no lineal. La transformación lineal se realiza con una Matriz de Proyección, que se calcula con las ecuaciones de la transformación no lineal. Para esto, tenemos la sig. función:

glm::mat4 perspective (ángulo, aspecto, n, f)
(f → far, n → near)

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

- De Traslación: Tiene la sig. apariencia:

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde x, y, z son la posición a la que trasladas un objeto.

Se multiplica esta matriz por el vector de posición de nuestro objeto, y el resultado es otro vector con las nuevas coordenadas de posición. En OpenGL se ve así:

```
glm::mat4 mym = glm::translate(glm::mat4, glm::vec3(x,y,z));
glm::vec4 myV(x, y, z, 0 o 1 dependiendo_si_es_un_vector_o_dirección);
glm::vec4 transformedVector = mym * myV;
```

- De Rotación:

Se utiliza la sig. función

```
glm::vec3 rotacion(x, y, z);
glm::rotate(angulo_en_grados, rotacion);
```

- De escalado Tienen la sig. forma:

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Se sigue la misma operación que con las matrices de traslación. En glm se hace así:

```
glm::mat4 escalado = glm::scale(x, y, z);
```

Conclusión

Aprendí muchísimo con mi investigación, realmente no se me ocurrió nunca que muchas de las acciones que podemos realizar con un objeto se hicieran con matrices, o lo complicado que es definir una simple cámara en un entorno 3D.
Además, comprendí mejor la sintaxis de las líneas de código que utilizan glm, los tipos de datos y declaraciones.

Referencias

- Dpto. de Ciencias e Ingeniería de la Computación, “Computación Gráfica”. Universidad Nacional del Sur. Recuperado el 14 de febrero de 2024 de http://www.cs.uns.edu.ar/cg/clasespdf/3-Pipe3D.pdf
- Opengl-tutorial, “Tutorial 3 : Matrices”. Recuperado el 14 de febrero de 2024 de https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/
- Departamento de Tecnologías de la Información, “Tema 4. El lenguaje GLSL”. Universidad de Huelva. Recuperado el 14 de febrero de 2024 de https://www.uhu.es/francisco.moreno/gii_rv/docs/Tema_4.pdf
- “GLSL Tutorial – Uniform Variables”. Recuperado el 14 de febrero de 2024 de https://www.lighthouse3d.com/tutorials/glsl-tutorial/uniform-variables/
- “Capítulo 2: Léxico de C. Tipos básicos de datos , visibilidad y almacenamiento”. Universidad de Granada. Recuperado el 15 de febrero de 2024 de https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap2/f_cap25.htm