Reporte de Práctica de Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Información de la práctica

Nombre del alumno:	Martínez Pérez Brian Erik		
Número de Cuenta:	319049792	Práctica número:	2
Título de la práctica:	Proyecciones y puertos de vista. Transformaciones geométricas		
Fecha de entrega:	1 de septiembre de 2025		

Introducción: (descripción inicial sobre la práctica)

Esta práctica tiene como objetivo aplicar proyecciones y transformaciones geométricas básicas en OpenGL para visualizar un cubo en 3D. Se aplicarán dos tipos de proyecciones: ortográfica y perspectiva, además se aplicarán las operaciones de transformaciones como traslación, rotación y escalamiento. El trabajo tiene como propósito comprender las características de los dos tipos de proyecciones y observar visualmente el cómo afectan la vista cada una de ellas.

Resumen Teórico: (introducción sobre los fundamentos teóricos en los que se basa la práctica y cuáles son los objetivos teóricos que persigue.)

Conocimiento de sistemas de proyección

Proyección ortogonal: Representación sin perspectiva donde los objetos mantienen su tamaño independientemente de la distancia.

Proyección en perspectiva: Simula la visión humana, donde los objetos lejanos aparecen más pequeños.

Transformaciones geométricas:

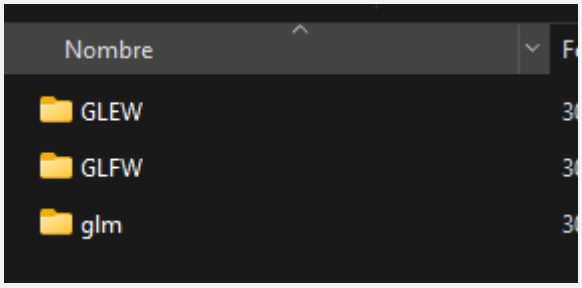
Traslación: Movimiento de objetos en el espacio 3D.

Rotación: Giro de objetos alrededor de un eje.

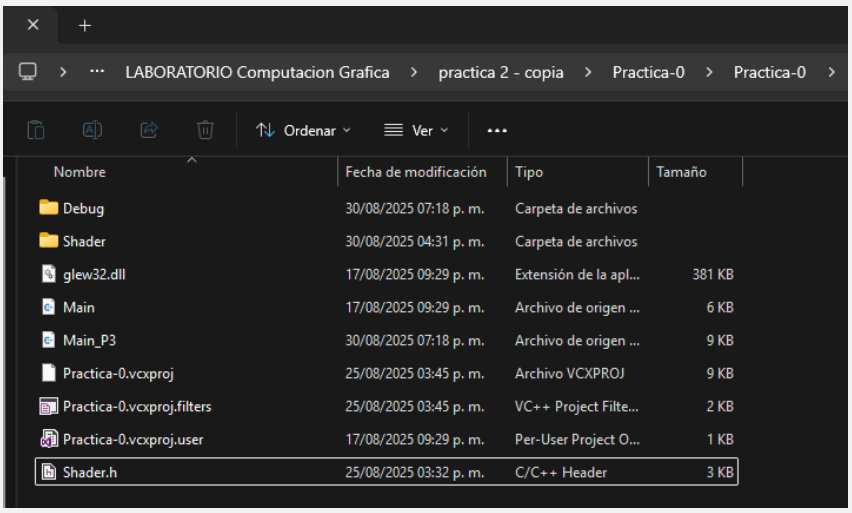
Escalamiento: Cambio de tamaño de los objetos.

Descripción de la práctica: (En esta sección se deben describir todos los pasos ejecutados durante la sesión práctica realizada en el laboratorio.)

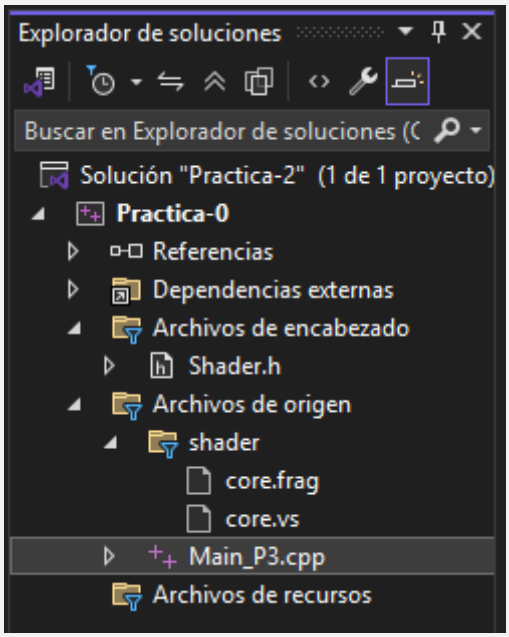
Agregamos la carpeta “glm” en la sección de eternal libraries, esta librería nos ayudara a crear las proyecciones y transformaciones



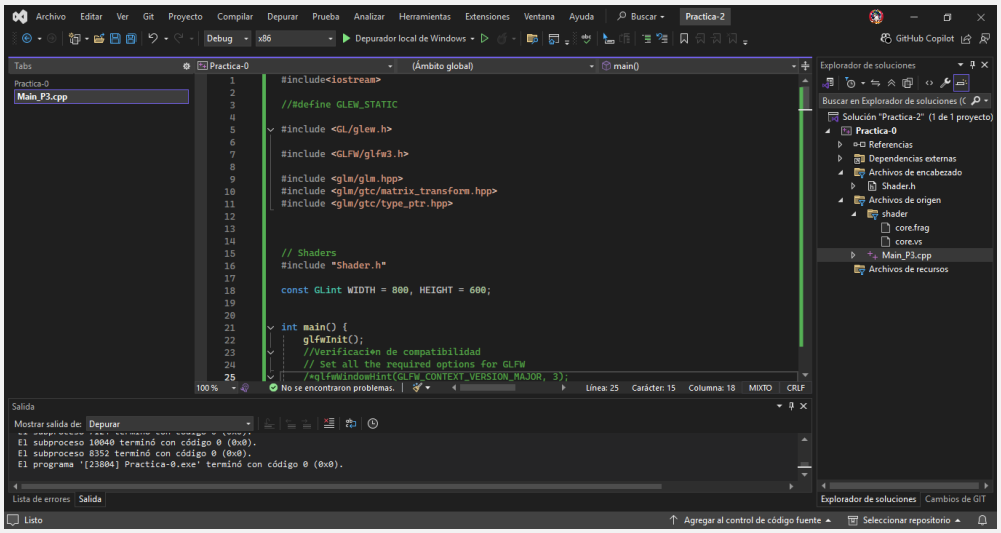
Además, agregamos los archivos “Main_P3”, “Shader.h” y la carpeta “shader” en la ruta que se visualiza en la imagen.



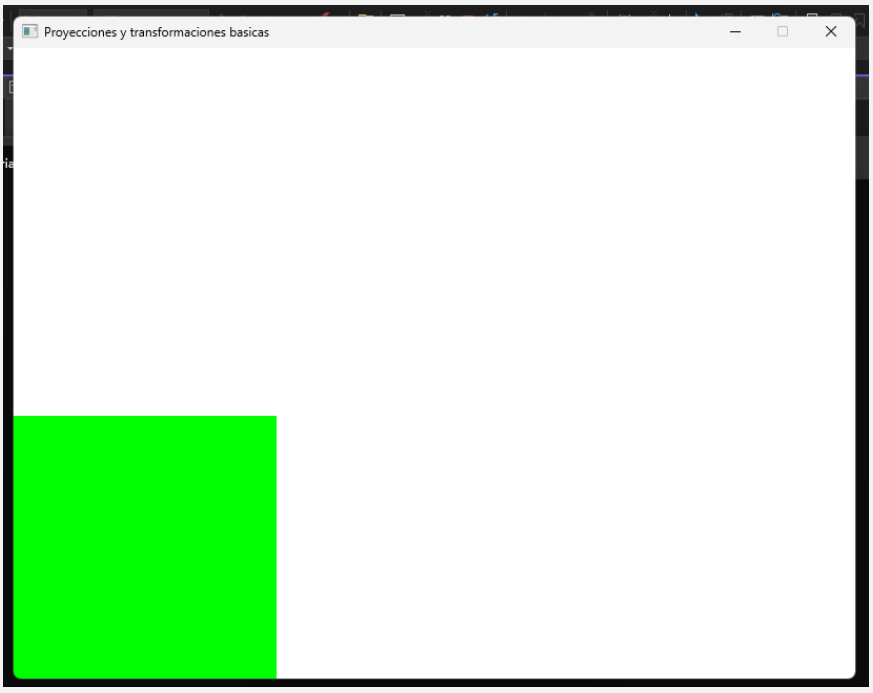
Por último, en la configuración de archivos, dejamos el archivo “Shader.h” en el apartado de Archivos de origen, y en archivos de origen colocamos la carpeta que contenía los archivos “core.frag”, “core.vs” y “Main_P3.cpp”.



Una vez que terminamos la configuración de archivos, podemos visualizar el archivo “Main_P3.cpp”, podemos ver el código que contiene el archivo.



Al ejecutar el código podemos ver un cuadrado verde, aunque realmente lo que realiza el código es un cubo, solo podemos una ver una cara de este ya que no tenemos una perspectiva activada en este punto de la ejecución, además cabe mencionar que la construcción del cubo completo se realizo con las primitivas de vértices que forman un triángulo, al ser un cubo debemos usar dos triángulos para generar una sola cara del cubo.

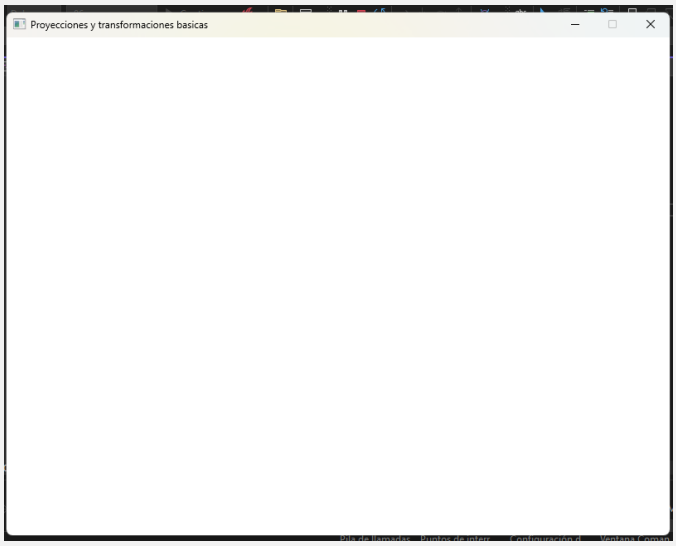


Este segmento de código es donde se definen los vértices que generan el cubo.

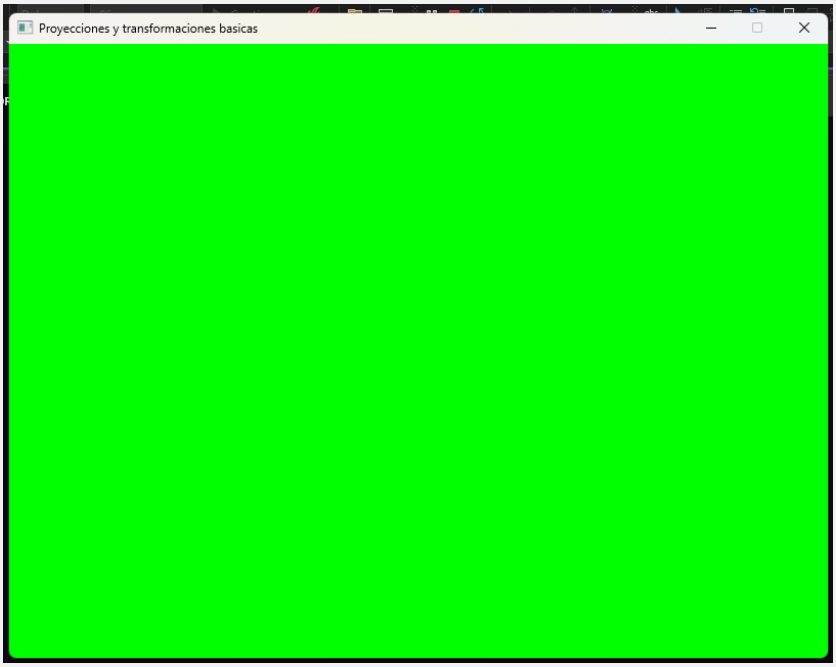
```
GLfloat vertices[] = {
   -0.5f * 500, -0.5f, 0.5f, 1.0f, 0.0f,0.0f,//Front
    0.5f * 500, -0.5f * 500, 0.5f * 500,  1.0f, 0.0f,0.0f,
    0.5f * 500,  0.5f * 500, 0.5f * 500,  1.0f, 0.0f,0.0f,
    0.5f * 500,  0.5f * 500, 0.5f * 500,  1.0f, 0.0f,0.0f,
   -0.5f * 500,  0.5f * 500, 0.5f * 500, 1.0f, 0.0f,0.0f,
   -0.5f * 500, -0.5f * 500, 0.5f * 500, 1.0f, 0.0f,0.0f,

   -0.5f * 500, -0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,//Back
    0.5f * 500, -0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,
    0.5f * 500,  0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,
    0.5f * 500,  0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,
   -0.5f * 500,  0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,
   -0.5f * 500, -0.5f * 500,-0.5f * 500, 0.0f, 1.0f,0.0f,

    0.5f * 500, -0.5f * 500,  0.5f * 500,  0.0f, 0.0f,1.0f,
    0.5f * 500, -0.5f * 500, -0.5f * 500,  0.0f, 0.0f,1.0f,
    0.5f * 500,  0.5f * 500, -0.5f * 500,  0.0f, 0.0f,1.0f,
    0.5f * 500,  0.5f * 500, -0.5f * 500,  0.0f, 0.0f,1.0f,
    0.5f * 500,  0.5f * 500,  0.5f * 500,  0.0f, 0.0f,1.0f,
    0.5f * 500,  -0.5f * 500, 0.5f * 500, 0.0f, 0.0f,1.0f,
```

Al ejecutar el otro segmento de código comentado obtenemos una ventana blanca ya que los vértices en el código anterior tenían un factor multiplicado por 500 que realizaban un escalamiento.

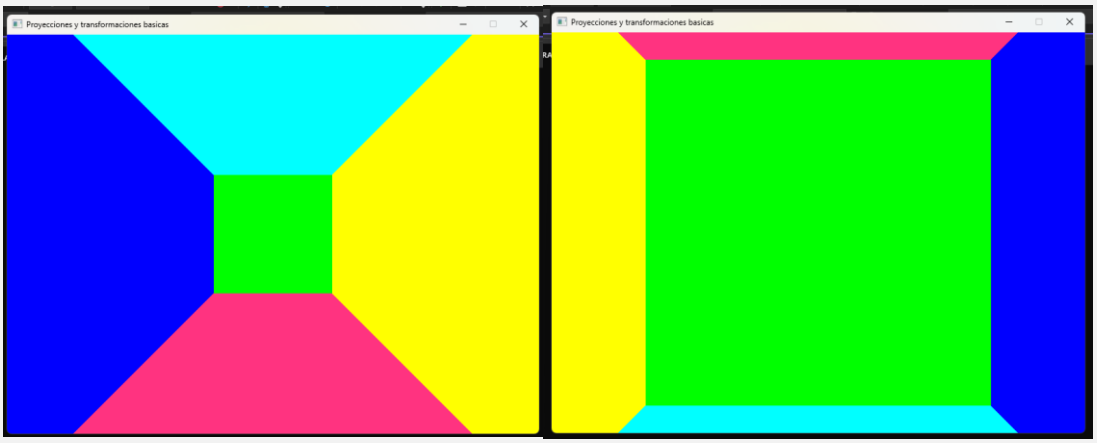
```
// use with Perspective Projection
float vertices[] = {
    -0.5f, -0.5f, 0.5f, 1.0f, 0.0f,0.0f,//Front
    0.5f, -0.5f, 0.5f,  1.0f, 0.0f,0.0f,
    0.5f,  0.5f, 0.5f,  1.0f, 0.0f,0.0f,
    0.5f,  0.5f, 0.5f,  1.0f, 0.0f,0.0f,
    -0.5f,  0.5f, 0.5f, 1.0f, 0.0f,0.0f,
    -0.5f, -0.5f, 0.5f, 1.0f, 0.0f,0.0f,

    -0.5f, -0.5f,-0.5f, 0.0f, 1.0f,0.0f,//Back
     0.5f, -0.5f,-0.5f, 0.0f, 1.0f,0.0f,
     0.5f,  0.5f,-0.5f, 0.0f, 1.0f,0.0f,
     0.5f,  0.5f,-0.5f, 0.0f, 1.0f,0.0f,
    -0.5f,  0.5f,-0.5f, 0.0f, 1.0f,0.0f,
    -0.5f, -0.5f,-0.5f, 0.0f, 1.0f,0.0f,
```



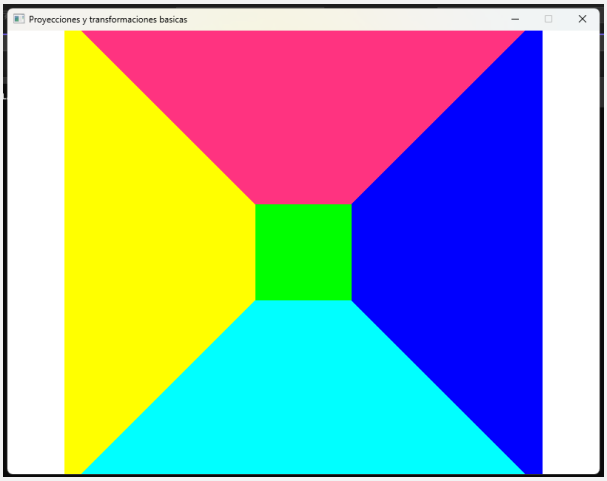
Con la instrucción `projection = glm::perspective(45.0f, (GLfloat)screenWidth /(GLfloat)screenHeight, 0.1f, 100.0f);` activamos la proyección en perspectiva, y lo que visualizamos es la ventana de color verde, ya que la proyección hace que la vista sea demasiado cercana a una de las caras del cubo.



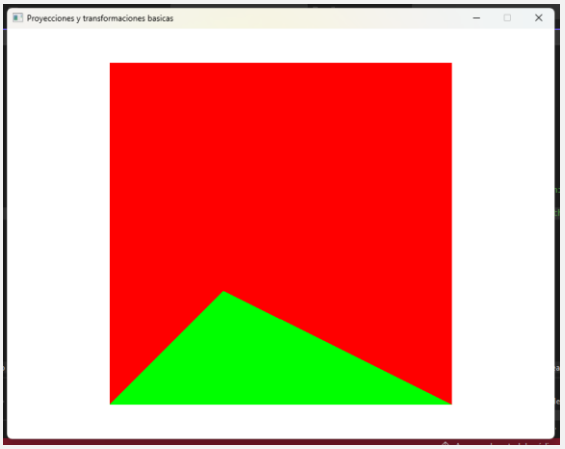
Al cambiar el parámetro de ángulo de visión en la instrucción `projection = glm::perspective(10.0f, (GLfloat)screenWidth / (GLfloat)screenHeight, 0.1f, 100.0f`, cambiamos la proyección, y ahora observamos la vista desde adentro del cubo y podemos 5 de las caras del cubo. También anexo capturas para distintos valores del ángulo.

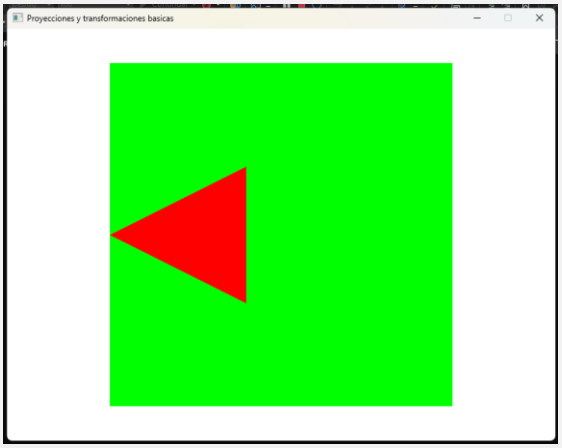


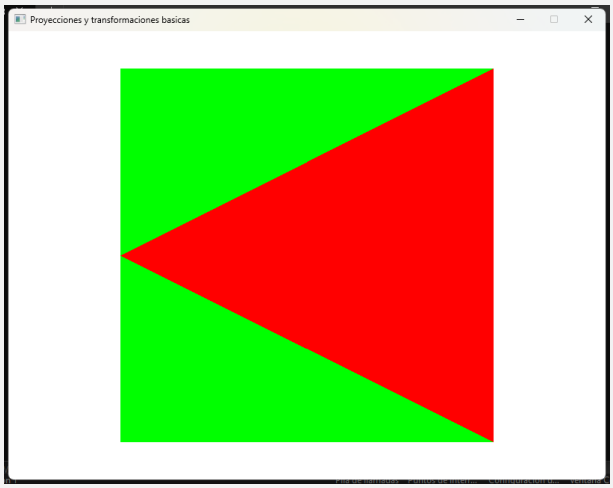
Reporte de Práctica de Laboratorio de Computación Gráfica e Interacción Humano-Computadora



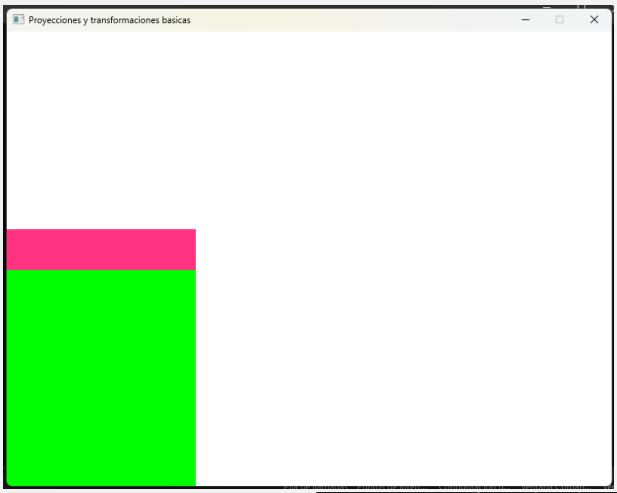
Utlizando la proyeccion ortografica podemos observar y la linea: `view = glm::translate( view, glm::vec3( screenWidth / 2, screenHeight / 2,-700.0f ) ); // use with orthographic projection,` el “-700.0f” es vista de la camara en el eje Z, ademas se anexan capturas para distintos valores de la vista del eje z.

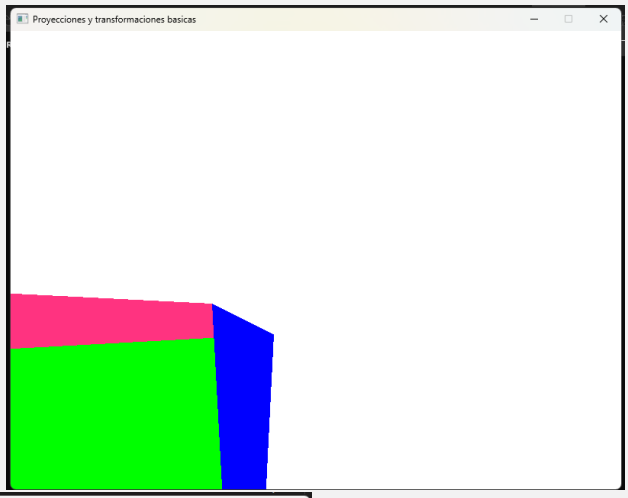


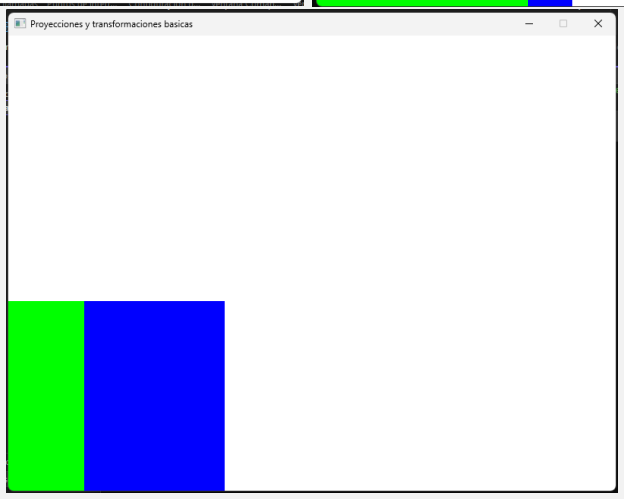




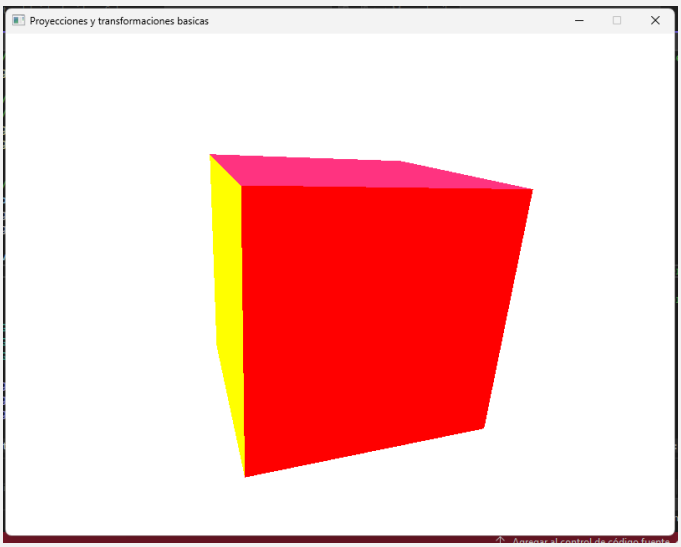
Ahora usan la línea `model = glm::rotate( model, -0.5f, glm::vec3( 1.0f, 2.0f, 0.0f ) );` aplicamos una rotación en los distintos ejes del cubo, dependiendo el cambio en los parámetros, visualizaremos las distintas caras del cubo, este caso anexamos imágenes donde podemos ver hasta 3 caras del cubo, eso si al ver las tres caras vemos deformes todas.

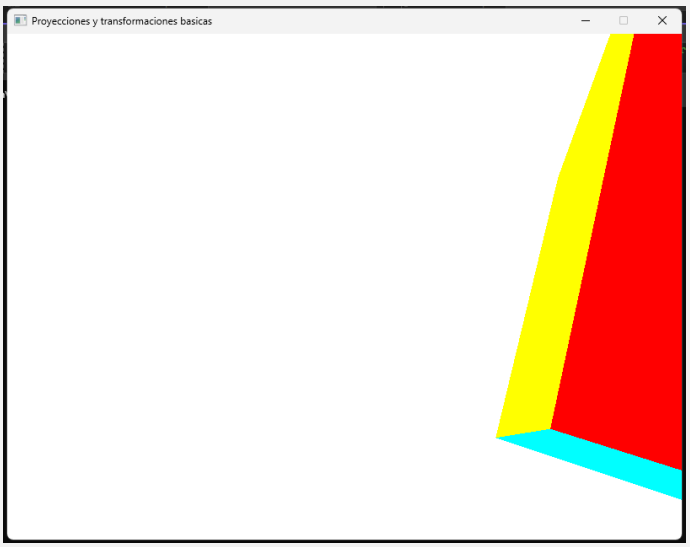






Por ultimo para poder ver las 3 caras del cubo, regresamos al arreglo de vertices de proyeccion en perpectiva para ello ademas aplicamos la operaciones de traslacion y rotacion para poder ver como afectan estas a la simulacion.





Reporte de Práctica de Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Resultados: (explicar lo que se logró o aquello que no lograron realizar o comprender)

Logramos aplicar y observar cada los dos tipos de proyecciones, ver como como en la proyección ortogonal, no podemos observar la componente “Z” de forma natural, que es la forma de profundidad, y observamos el objeto en solo 2 dimensiones. Para la proyección en perspectiva vimos cómo se cambia el tamaño al mover la cámara que es la que genera la un cambio en el tamaño del cubo, además en esta proyección observamos las 3 caras distintas del cubo.

Por último, aplicamos también las 3 transformaciones geométricas sobre el cubo, la primera la traslación dentro de la ventana blanca que generamos, la segunda la rotación, al girar alguno de los ejes del cubo y observar las caras que este contiene, y por ultimo el escalamiento, que genera un cambio en el tamaño de las caras del cubo, podíamos hacerlo pequeño o gigante.

Conclusiones:

En esta práctica logre comprender las diferencias entre los 2 tipos de proyecciones que existen, ninguna es mejor que la otra ya que una tiene diferentes funciones a las otras, y dependiendo el efecto que deseemos aplicar, será la proyección que debamos usar en nuestros modelos. La mejor forma de entenderlas fue aplicar al cubo y ver como afectaban a la vista del espectador. Además, las proyecciones las acompañamos de operaciones geométricas, las cuales cambiaban la forma de ver el cubo ya sea girándolo, cambiando de posición o modifican el tamaño, todas las acciones las aplicamos en el código y se verificaban al ejecutar el código y ver la ventana de salida.

Bibliografía consultada:

GLFW. (2023). GLFW Documentation. https://www.glfw.org/docs/latest/

GLM. (2023). OpenGL Mathematics (GLM). https://github.com/g-truc/glm