

## Ejercicios

1. Dibujar las iniciales de sus nombres, cada letra de un color diferente.
2. Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando pirámides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp.

## Desarrollo

Para el desarrollo de esta práctica se realizaron varios cambios al programa principal. En el dibujo de las letras se utilizó la función `CrearLetrasYFiguras()`, en donde se crearon tres arreglos:

- `vertices_letraV`
- `vertices_letraA`
- `vertices_letraN`

```
//Asignando los vértices de la letra "V"
GLfloat vertices_letraV[] {
    //X      Y      Z      R      G      B
    0.3f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.5f,    0.3f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.5f,    0.5f,    0.0f,    0.5f,    0.2f,    0.5f,

    0.5f,    0.3f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.5f,    0.5f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.7f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,

    0.5f,    0.5f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.6f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.7f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,

    0.4f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.5f,    0.5f,    0.0f,    0.5f,    0.2f,    0.5f,
    0.3f,    0.7f,    0.0f,    0.5f,    0.2f,    0.5f,

};

MeshColor* letraV = new MeshColor();
letraV->CreateMeshColor(vertices_letraV, 72);
meshColorList.push_back(letraV);
```

El color decidido de las letras fue combinando los colores disponibles, para así obtener un resultado específico:

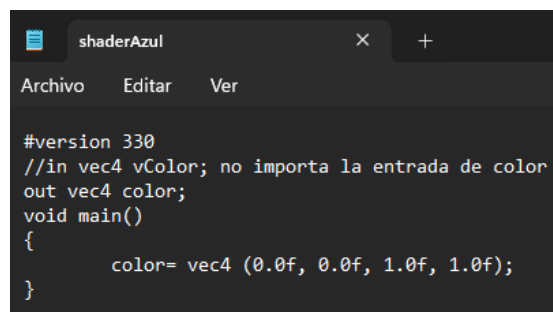
VAN

Donde cada uno de ellos almacena la información de los vértices correspondientes de cada letra que se iba a dibujar. Se agregaron a la lista `meshColorList`, la cual nos iba a permitir mostrar la letra más adelante.

```
// Letra V
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.7f, 1.3f, -4.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshColorList[0]->RenderMeshColor();
```

Después que se instanciaron pirámides y cubos en vez de cuadrados y triángulos para el dibujo de la casa, por lo que se tuvieron que utilizar shaders para lograr realizar la actividad.

Se crearon cinco Fragment Shader en dónde se define el color a través del código, por ejemplo:



```
shaderAzul
Archivo  Editar  Ver

#version 330
//in vec4 vColor; no importa la entrada de color
out vec4 color;
void main()
{
    color= vec4 (0.0f, 0.0f, 1.0f, 1.0f);
}
```

Lo siguiente fue definir las variables con la dirección de los archivos que se planean utilizar, seguido de esto se crearon objetos de la clase Shader para poder utilizarlos.

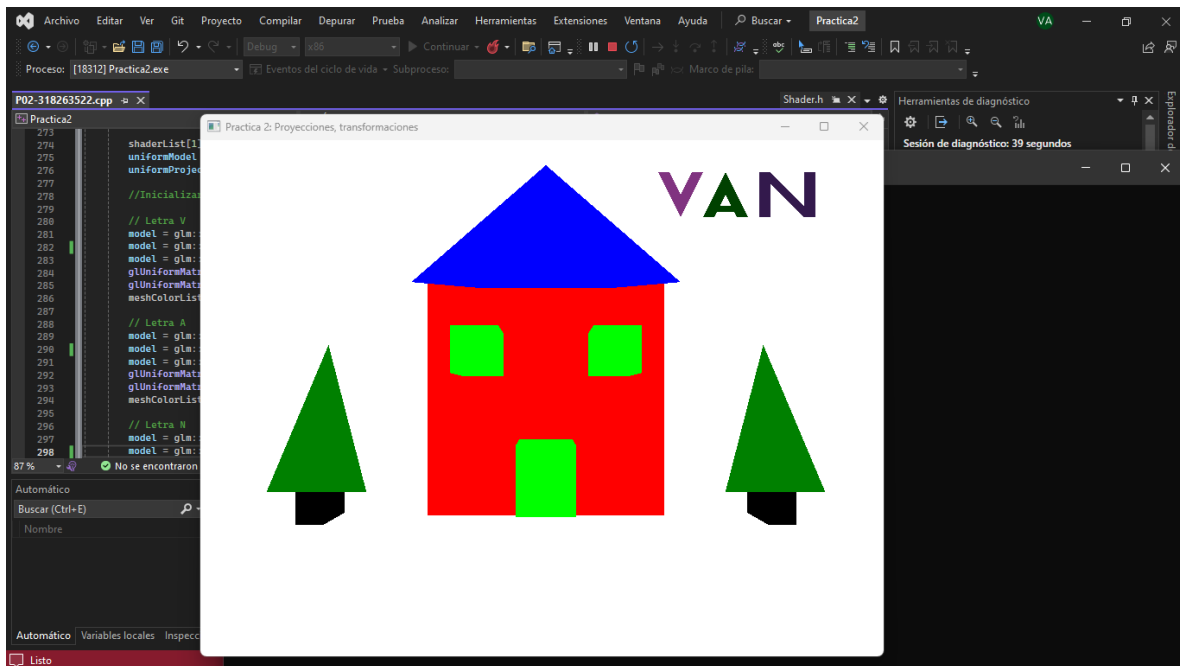
Finalmente, para dibujar se calcularon nuevamente las matrices de transformación como en el ejercicio de laboratorio previamente hecho.

```
//Triangulo azul
shaderList[4].useShader();
uniformModel = shaderList[4].getModelLocation();
uniformProjection = shaderList[4].getProjectLocation();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 1.8f, -4.5f));
model = glm::scale(model, glm::vec3(2.7f, 1.3f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
meshList[0]->RenderMesh();
```

## Resultados

Casa desde vista prespectiva.



### Obstáculos que se presentaron

Uno de los mayores obstáculos que se me presentaron fue el manejo de los shaders, ya que no comprendía como es que se utilizaba junto con las figuras geométricas. Sin embargo, logré comprender el funcionamiento de los shaders y después y la manera que se utilizan.

### Conclusiones

Los ejercicios que se dejaron como actividad, creo que fueron sencillos y claros, lo que nos permite tener una mejor comprensión y un mejor manejo de los shaders.

Además, creo que la aproximación que se brinda para la estructura del proyecto es mucho mejor que la que se proponía en la sesión pasada. De esta manera, ahora estamos segregando distintas responsabilidades en distintos archivos y de esta forma evitamos tener archivos muy largos o con mucha lógica.

### Bibliografía

De Vries, J. (2020). Learn OpenGL: Learn Modern OpenGL Graphics Programming in a Step-by-step Fashion