UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 02

NOMBRE COMPLETO: Vargas López Miguel Adán

Nº de Cuenta: 421079522

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 4

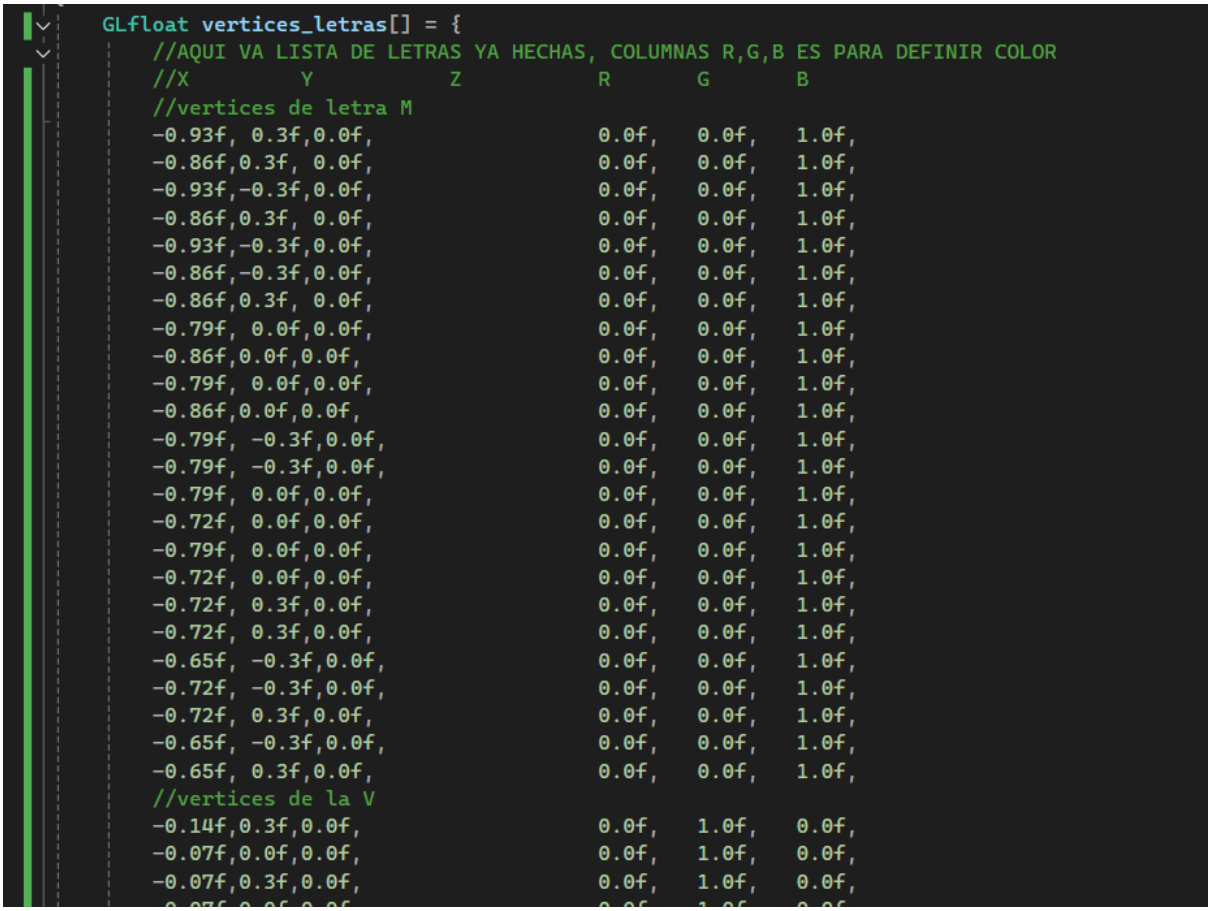
SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 24/08/2024

CALIFICACIÓN: _____

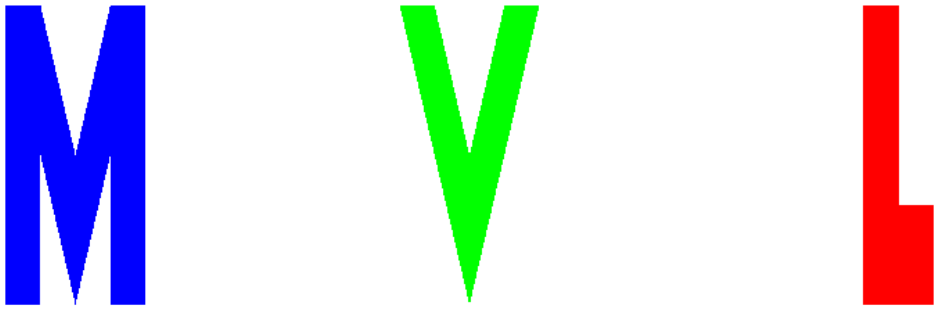
1. Ejercicio - Dibujar las iniciales de sus nombres, cada letra de un color diferente

   Para este ejercicio lo que se modificó fue el método vertices_letras donde asignamos los vértices para dibujar mis iniciales ‘M V L’, asignando a cada grupo de cada letra un color distinto.



```
GLfloat vertices_letras[] = {
    //AQUI VA LISTA DE LETRAS YA HECHAS, COLUMNAS R,G,B ES PARA DEFINIR COLOR
    //X         Y           Z           R       G       B
    //vertices de letra M
    -0.93f, 0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,0.3f, 0.0f,                  0.0f,   0.0f,   1.0f,
    -0.93f,-0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,0.3f, 0.0f,                  0.0f,   0.0f,   1.0f,
    -0.93f,-0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,-0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,0.3f, 0.0f,                  0.0f,   0.0f,   1.0f,
    -0.79f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,0.0f,0.0f,                   0.0f,   0.0f,   1.0f,
    -0.79f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.86f,0.0f,0.0f,                   0.0f,   0.0f,   1.0f,
    -0.79f, -0.3f,0.0f,                 0.0f,   0.0f,   1.0f,
    -0.79f, -0.3f,0.0f,                 0.0f,   0.0f,   1.0f,
    -0.79f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.72f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.79f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.72f, 0.0f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.72f, 0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.72f, 0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.65f, -0.3f,0.0f,                 0.0f,   0.0f,   1.0f,
    -0.72f, -0.3f,0.0f,                 0.0f,   0.0f,   1.0f,
    -0.72f, 0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    -0.65f, -0.3f,0.0f,                 0.0f,   0.0f,   1.0f,
    -0.65f, 0.3f,0.0f,                  0.0f,   0.0f,   1.0f,
    //vertices de la V
    -0.14f,0.3f,0.0f,                   0.0f,   1.0f,   0.0f,
    -0.07f,0.0f,0.0f,                   0.0f,   1.0f,   0.0f,
    -0.07f,0.3f,0.0f,                   0.0f,   1.0f,   0.0f,
```

   Teniendo el siguiente resultado:

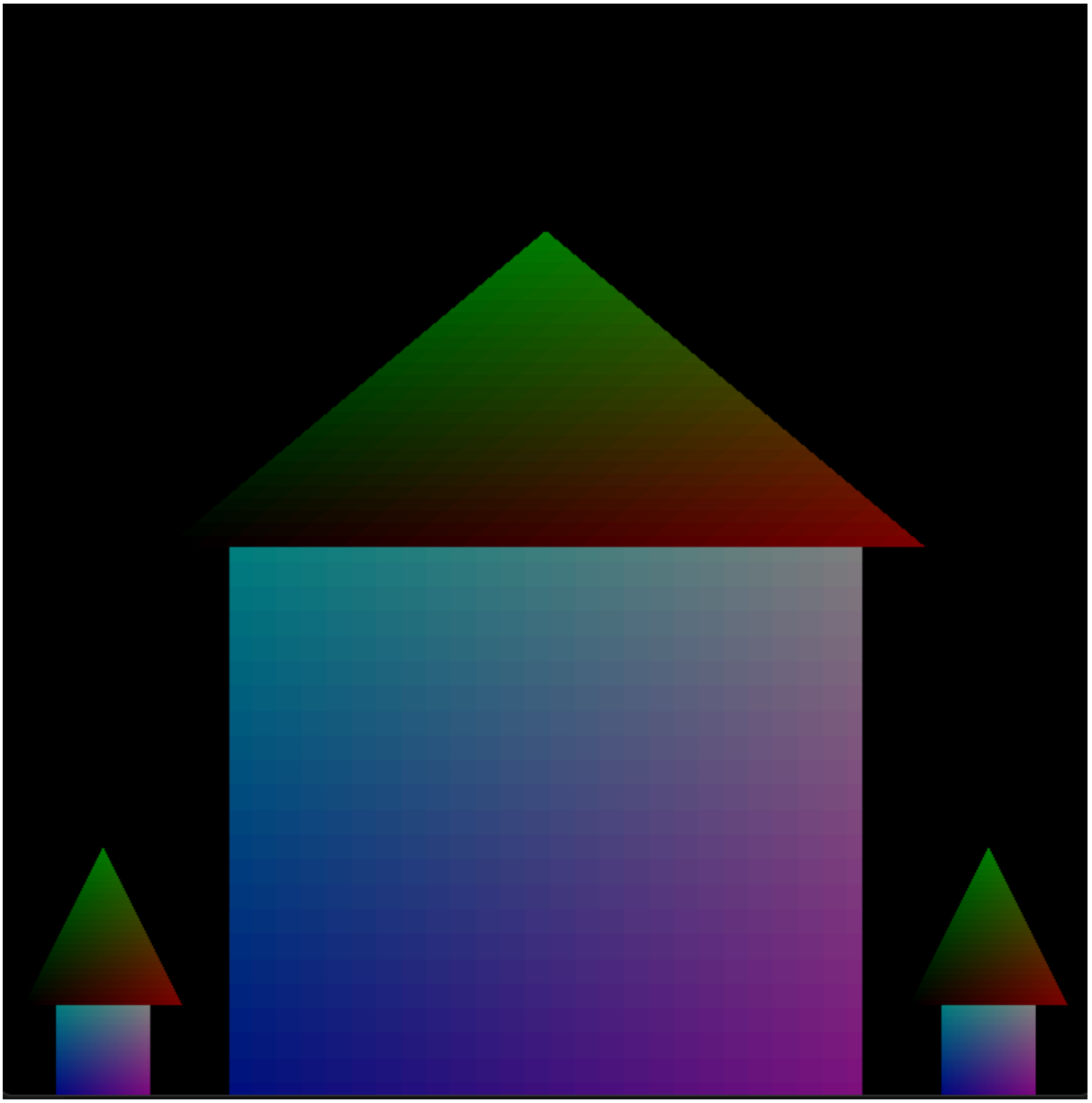


2.Ejercicio - Generar el dibujo de la casa de la clase, pero en lugar de instanciar triángulos y cuadrados será instanciando pirámides y cubos, para esto se requiere crear shaders diferentes de los colores: rojo, verde, azul, café y verde oscuro en lugar de usar el shader con el color clamp

Primero para esta visualización instanciamos los cubos y las pirámides necesarias escalandolas en el tamaño necesario para lograr el dibujo y así como trasladando las figuras a su posición necesaria.

```
//cubo tronco izquierda
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.4f, -1.6f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.0f));
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));

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[1]->RenderMesh();

//pino izquierda
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.4f, -1.2f, -3.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.0f));
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));

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();

//cubo tronco derecha
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.4f, -1.6f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.0f));
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));

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[1]->RenderMesh();

//pino derecha
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.4f, -1.2f, -3.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.0f));
//model = glm::rotate(model, glm::radians(angulo), glm::vec3(0.0f, 1.0f, 0.0f));
```

En las primeras ejecuciones tendríamos un resultado como el siguiente:



Lo siguiente sería declarar los shaders para cada color necesario para lograr la ilustración

Conclusiones

Para esta práctica no tuve complicaciones para el primer ejercicio, sin embargo en el segundo ejercicio los tuve para manejar los shaders, no tanto para instanciar, escalar y trasladar figuras, pero sí para crear los shaders y asignarlos a cada objeto, pero alcanzo a apreciar el potencial de futuros programas a realizar con opengl al empezar a manejar espacios en 3D, así como la habilidad para manejar objetos primitivos para obtener visualizaciones más complejas a través de estos mismos objetos como cubos y pirámides para construir visualmente una casa o un pino.

Bibliografía

***Utilizar los shaders para aplicar color en WebGL - Referencia de la API Web* | *MDN*. (2023, November 6). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/API/WebGL_API/Tutorial/Using_shaders_to_apply_color_in_WebGL**