UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 03

NOMBRE COMPLETO: Hernández Vázquez Daniela

Nº de Cuenta: 318092867

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 02

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 28 de febrero de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

Para el ejercicio se solicita instanciar primitivas geométricas para recrear el dibujo de la práctica pasada (una cada) en 3D,se requiere que exista un piso; la casa tiene una ventana azul circular justo en medio de la pared trasera, 2 ventanas verdes en cada pared lateral iguales a las de la pared frontal y solo puerta en la pared frontal.

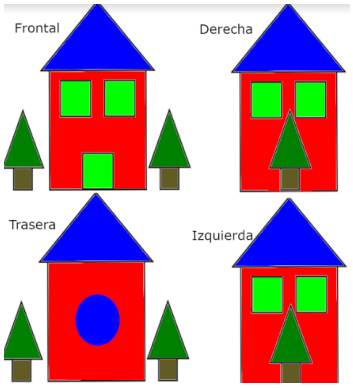

Figura 1. Figura de referencia

Durante la sesión se revisaron los archivos correspondientes a la practica 3. Se observó el uso de la cámara y algunas otras opciones referentes al uso de la misma. Si como ya habíamos trabajado en la practica 2 al instanciar figuran en 3D.
En este practica tenemos al igual que la anteriores los shaders como archivos eternos que deben de ser declarados para su uso.

Par instanciar la figura lo primero que se debe de hacer es modificar la linea del shader.vert activando la linea de *gl_Position* que considera la matriz view

shader.vert E03-318092867.cpp

```
1   #version 330
2   layout (location =0) in vec3 pos;
3   out vec4 vColor;
4   uniform mat4 model;
5   uniform mat4 projection;
6   uniform vec3 color;
7   uniform mat4 view;
8   void main()
9   {
10      gl_Position=projection*view*model*vec4(pos,1.0f);
11      //gl_Position=projection*model*vec4(pos,1.0f);
12      vColor=vec4(color,1.0f);
13
14  }
```

Figura 2. Figura de referencia

Modificamos para la figura del cilindro y cono con cuantos triángulos queremos que la figura sea formada, a partir de 20 triángulos la figura ya muestra una figura redondeada suficiente.

Finalmente instanciamos las figuras identificando cuales son instanciadas con el Create mesh (cubo y piramide) y con el create mesh geometry (cono y cilindro). Se inició por instanciar el cubo y con base a este se fueron dibujando los demás elementos de la imagen ajustándolo manualmente aplicando transformaciones de escalamiento y traslación.

```
375
376         // Cubo rojo
377         model = glm::mat4(1.0);
378         model = glm::translate(model, glm::vec3(0.0f, 0.5f, -4.0f));
379         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
380         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
381         //se programe cambio entre proyección ortogonal y perspectiva
382         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
383         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
384         color = glm::vec3(1.0f, 0.0f, 0.0f); // declarar color
385         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
386         meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
387
388         // Primamide azul
389         model = glm::mat4(1.0);
390         model = glm::translate(model, glm::vec3(0.0f, 1.525f, -4.0f));
391         model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
392         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
393         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
394         //se programe cambio entre proyección ortogonal y perspectiva
395         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
396         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
397         color = glm::vec3(0.0f, 0.0f, 1.0f); // declarar color
398         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
399         meshList[4]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
400
401         // Ventana verde 1 IZQ-DER
402         model = glm::mat4(1.0);
403         model = glm::translate(model, glm::vec3(0.0f, 0.7f, -4.25f));
```

Figura 3. Figuras instanciadas con transformaciones

```
459         // Cilindro cian
460         model = glm::mat4(1.0);
461         model = glm::translate(model, glm::vec3(0.0f, 0.6f, -4.55f));
462         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.05f));
463         model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
464         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
465         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
466         color = glm::vec3(0.0f, 0.0f, 1.0f); // declarar color
467         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
468         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
469
470
471         // Arboles
472
473         // Cilindro café
474         model = glm::mat4(1.0);
475         model = glm::translate(model, glm::vec3(2.0f, 0.3f, -4.25f));
476         model = glm::scale(model, glm::vec3(0.15f, 0.5f, 0.15f));
477         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
478         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
479         color = glm::vec3(0.478f, 0.255f, 0.067f); // declarar color
480         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
481         meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
482
483         // Cilindro café
484         model = glm::mat4(1.0);
485         model = glm::translate(model, glm::vec3(-2.0f, 0.3f, -4.25f));
```

Figura 4. Figuras instanciadas con transformaciones

Todo esto dentro de la sección main, en este caso se utiliza una proyección de perspectiva y la cámara se controla mediante las teclas A(Izquierda), S(Alejar), D(Derecha), W(Acercar). Al ejecutar el programa obtenemos los siguientes resultados.

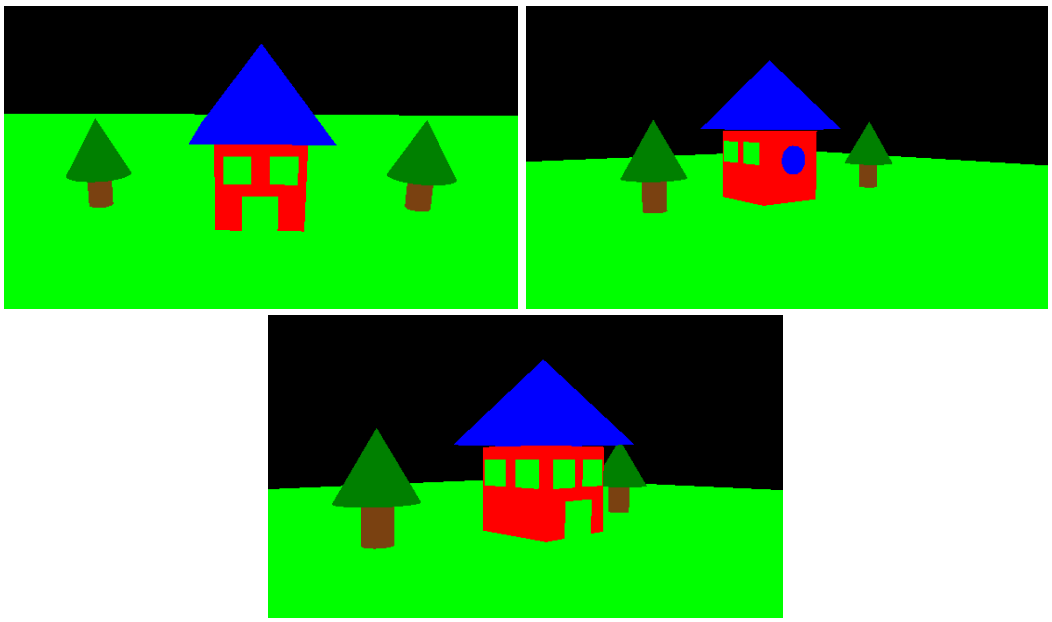

Figura 5. Resultado de ejecución

Problemas:

No hubo gran problema al instanciar las figuras más que tener un orden respecto a la utilización de los mesh y resultó bastante tardado modelar las 4 caras de la figura sin que los objetos se interpusieran unos con otros o que la escala fuera incorrecta, por lo que una vez generada una figura medianamente parecida se debía de reajustar con flotantes tomando un objeto base de referencia. Asi mismo cuando no se declaraban correctamente las líneas de color o modelo las figuras directamente no se instanciaban, pero se logró la actividad correctamente.

Conclusión:

Con respecto a los ejercicios de clase considero que tienen un buen nivel, lo más tardado de la actividad es realizar los ajustes. Me hubiera gustado que el uso de la cámara se hubiera explicado mejor y además de eso fue de gran ayuda tener un video de como realizar las configuraciones pues si algo no quedaba claro teníamos una referencia con la cual comparar.