Universidad Nacional

Autónoma de México

Practica No. 3

Alumno: Arroyo Llanes Miguel Alejandro.

No. Cuenta: 319218460

Materia: Lab. Computación Grafica e Interacción Humano-Computadora

Grupo Laboratorio: 3

Grupo teoría: 6

Fecha de Entrega: 01/03/2025

Nombre profesor: Ing. Jose Roque Roman Guadarrama

Comentario:

Para esta práctica se tuvo que entender como modelar en forma de 3d calculando y aproximando el espacio de la figura viendo el cómo serán y deberán ser ubicadas.

Se tuvieron que definir múltiples funciones de mini pirámides las cuales iban a ser los colores del pyraminx para formar el Rubik.

Se hizo uso de la rotación y translación para asegurar una correcta posición de las pirámides.

Desgraciadamente, no se logro satisfactoriamente toda la práctica, se logro la construcción de la mayor parte del pyraminx Rubik, sin embargo, las pirámides “volteadas” de las demás secciones no se lograron y solo se logro hacer en una cara de la figura siendo la verde, además de que se observó que a pesar de seguir las instrucciones y aparentemente los vértices estar bien colocados, por alguna razón una sección de las pirámides quedaban huecas.

Bloque de código:

Creación de las pirámides:

```
90  void PiramideMiniVolteada()
91  {
92      unsigned int indices_piramideMinivolt[] = {
93              0,1,2,
94              1,3,2,
95              3,0,2,
96              1,0,3
97
98      };
99      GLfloat vertices_piramideMinivolt[] = {
100         -0.3f, 0.3f,0.0f,   //0
101         0.3f,0.3f,0.0f, //1
102         0.0f,-0.3f, -0.3f,  //2
103         0.0f, 0.3f,-0.3f,   //3
104
105     };
106     Mesh* obj1 = new Mesh();
107     obj1->CreateMesh(vertices_piramideMinivolt, indices_piramideMinivolt, 12, 12);
108     meshList.push_back(obj1);
109
110 }
```

```cpp
void PiramideMiniVolteada2()
{
    unsigned int indices_piramideMinivolt2[] = {
            0,1,2,
            1,3,2,
            3,0,2,
            1,0,3

    };
    GLfloat vertices_piramideMinivolt2[] = {
        -0.3f, 0.3f,0.0f,   //0
        0.3f,0.3f,0.0f, //1
        0.0f,-0.3f, -0.f,   //2
        0.0f, 0.3f,-0.3f,   //3

    };
    Mesh* obj1 = new Mesh();
    obj1->CreateMesh(vertices_piramideMinivolt2, indices_piramideMinivolt2, 12, 12);
    meshList.push_back(obj1);

}
```

```cpp
void PiramideMini()
{
    unsigned int indices_piramideMini[] = {
            0,1,2,
            1,3,2,
            3,0,2,
            1,0,3

    };
    GLfloat vertices_piramideMini[] = {
        -0.3f, -0.3f,0.0f,  //0
        0.3f,-0.3f,0.0f,    //1
        0.0f,0.3f, -0.3f,   //2
        0.0f,-0.3f,-0.3f,   //3

    };
    Mesh* obj1 = new Mesh();
    obj1->CreateMesh(vertices_piramideMini, indices_piramideMini, 12, 12);
    meshList.push_back(obj1);

}
```

```
void PiramideMini2()
{
    unsigned int indices_piramideMini2[] = {
            0,1,2,
            1,3,2,
            3,0,2,
            1,0,3

    };
    GLfloat vertices_piramideMini2[] = {
        -0.3f, -0.3f,0.3f,  //0
        0.3f,-0.3f,0.0f,    //1
        0.0f,0.3f, -0.3f,   //2
        0.0f,-0.3f,-0.3f,   //3

    };
    Mesh* obj1 = new Mesh();
    obj1->CreateMesh(vertices_piramideMini2, indices_piramideMini2, 12, 12);
    meshList.push_back(obj1);

}
```

Llamar la creación de las pirámides:

```
int main()
{
    mainWindow = Window(1000, 800);
    mainWindow.Initialise();
    //Cilindro y cono reciben resolución (slices, rebanadas) y Radio de circunferencia de la base y tapa

    CrearCubo();//índice 0 en MeshList
    CrearPiramideTriangular();//índice 1 en MeshList
    CrearCilindro(5, 1.0f);//índice 2 en MeshList
    CrearCono(25, 2.0f);//índice 3 en MeshList
    CrearPiramideCuadrangular();//índice 4 en MeshList
    PiramideMini();//índice 5 en MeshList
    PiramideMiniVolteada();//índice 6 en MeshList
    PiramideMini2();//índice 7 en MeshList
    PiramideMiniVolteada2();//índice 8 en MeshList
    CreateShaders();
```

Dibujado de la primera cara de la pirámide:

```
////////////SECCION 1 DE LA CARA 1/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(-0.0f, -0.3f, -4.03f));
        model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
        glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
        meshList[5]->RenderMeshGeometry();


        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.3f, -0.3f, -4.03f));
        model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
        glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
        meshList[5]->RenderMeshGeometry();

        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(-0.3f, -0.3f, -4.03f));
        model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
        glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
        meshList[5]->RenderMeshGeometry();


        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(-0.15f, -0.24f, -4.05f));
        model = glm::rotate(model, glm::radians(305.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
        glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
        meshList[6]->RenderMeshGeometry();


        model = glm::mat4(1.0f);
        color = glm::vec3(0.0f, 1.0f, 0.0f);
        model = glm::translate(model, glm::vec3(0.15f, -0.24f, -4.05f));
        model = glm::rotate(model, glm::radians(305.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
        glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
        meshList[6]->RenderMeshGeometry();
```

```
////////////SECCION 2 DE LA CARA 1/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.15f, -0.0f, -4.18f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.15f, -0.0f, -4.18f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();


model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.0f, 0.055f, -4.2f));
model = glm::rotate(model, glm::radians(305.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[6]->RenderMeshGeometry();
```

```
////////////SECCION 3 DE LA CARA 1/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.0f, 0.3f, -4.33f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

Dibujado de la segunda cara de la pirámide:

```
////////////SECCION 1 DE LA CARA 2//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.32f, -0.3f, -4.05f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();


model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.17f, -0.3f, -4.2f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.02f, -0.3f, -4.35f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

```
////////////SECCION 2 DE LA CARA 2//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.02f, 0.0f, -4.35));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.17f, 0.0f, -4.2));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

```
////////////SECCION 3 DE LA CARA 2//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-0.025f, 0.3f, -4.35));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

Dibujado de la tercera cara de la pirámide:

```
////////////SECCION 1 DE LA CARA 3//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.33f, -0.3f, -4.05f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();


model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.17f, -0.3f, -4.2f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.02f, -0.3f, -4.35f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

```
////////////SECCION 2 DE LA CARA 3/////////////////////////////////////////////////////////////////////////////////////////////////////////////////
      model = glm::mat4(1.0f);
      color = glm::vec3(1.0f, 0.0f, 0.0f);
      model = glm::translate(model, glm::vec3(0.04f, 0.0f, -4.35));
      model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
      meshList[5]->RenderMeshGeometry();

      model = glm::mat4(1.0f);
      color = glm::vec3(1.0f, 0.0f, 0.0f);
      model = glm::translate(model, glm::vec3(0.17f, 0.0f, -4.2));
      model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
      meshList[5]->RenderMeshGeometry();
```

```
  ////////////SECCION 3 DE LA CARA 3/////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  model = glm::mat4(1.0f);
  color = glm::vec3(1.0f, 0.0f, 0.0f);
  model = glm::translate(model, glm::vec3(0.03f, 0.3f, -4.35));


  model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
  glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
  glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
  meshList[5]->RenderMeshGeometry();
```

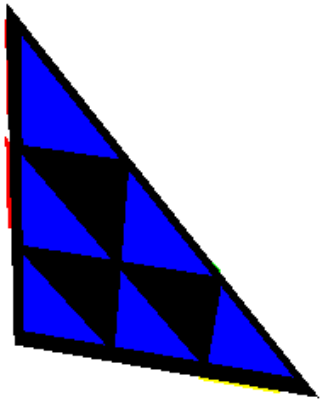
Dibujado de la cuarta cara de la pirámide:

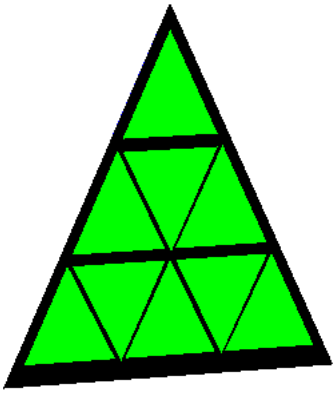
```
////////////SECCION 1 DE LA CARA 4/////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.0f, -0.39f, -4.05f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.3f, -0.39f, -4.05f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.3f, -0.39f, -4.05f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

```
////////////SECCION 2 DE LA CARA 4/////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.15f, -0.39f, -4.19f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();

model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.15f, -0.39f, -4.19f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

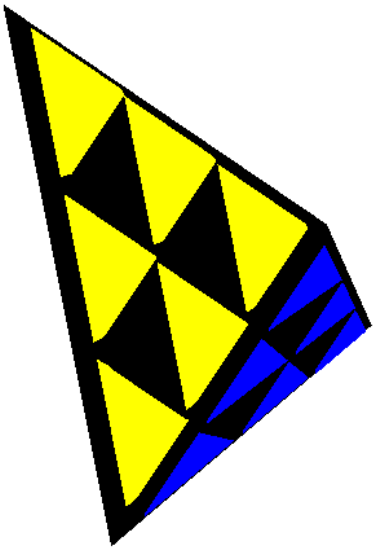
```
////////////SECCION 3 DE LA CARA 4/////////////////////////////////////////////////////////////////////////////////////////////////////////////////
model = glm::mat4(1.0f);
color = glm::vec3(1.0f, 1.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.0f, -0.39f, -4.33f));
model = glm::scale(model, glm::vec3(0.45f, 0.45f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[5]->RenderMeshGeometry();
```

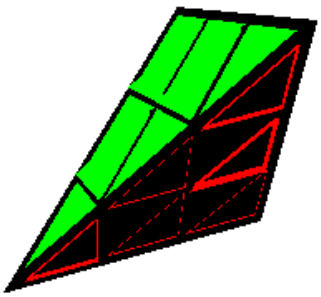
Ejecución:





Practica 03: Modelado Geom&#65533;trico y C&#65533;mara Sint&#65533;tica





Conclusión:

Esta practica fue en extremo difícil especialmente por el tiempo de entrega, se lograron entender y aprender finalmente las funciones más básicas de OpenGL, aunque aun se debe de encontrar una manera de eficientar el trabajo ya que con los conocimientos actuales sigue siendo muy complicado.

El manejo de la cámara, aunque novedoso se debe de encontrar una forma más controlada ya que, aunque se trató de bajar la sensibilidad en algunos intentos la realidad es que quizás por mi computadora, pero se trababa o al momento de probar el código era difícil de manejar.

A pesar de todo se ha entendido la construcción de una figura 3d en un espacio, así como su manipulación en el espacio, cambio de color y rotación, aun no se domina el tema pero cada vez se está más cerca de tener una mejor técnica en la practica de este tipo de problemas.

Bibliografía:

1. No se busco en ninguna fuente externa.