UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE INGENIERÍA.

LABORATORIO DE COMPUTACIÓN GRÁFICA.

GRUPO: 08

PROYECTO.

FELIX FLORES PAUL JAIME.

Carlos Aldair Roman Balbuena.

22 /01/2021.

ÍNDICE.

Descripción del proyecto.

Trate de recrear el taller de santa y ser lo mas fiel posible al entorno que dije que haría.

Los siete elementos para el proyecto:

-Casa.

-Adornos navideños.

-Trineo.

-Juguetes.

-Árbol.

-Mesas.

-Chimenea.

En mi proyecto hice eso e hice un poco más, me fui dejando ir por mi imaginación para crear las animaciones y ponerlas en los lugares correspondientes.

Aldea al ejecutar el proyecto podremos ver esto, a simple vista observamos todo lo que veremos en este proyecto

Como podemos observar en la primera imagen, vemos el usado del Skybox así como el uso de cámara.

Dividí mi proyecto en 12 cuadrantes:

El lugar numero 3 y *, están vacíos porque cómo mi proyecto es compartido con teoría, en eso cuadrantes va el material de otra compañera, por eso se dejaron vacíos.

En la aldea navideña podemos ver dos tipos de casas, una casa y a lado de ella el pequeño taller.

Estas animaciones obj., las descargue de https://www.turbosquid.com/ y una vez ejecutadas en Maya uní los elementos.

Taller de santa.

El taller de santa, está conformado por 4 niveles, esta casa esta hecha por mi desde cero en Maya.

En la planta baja vemos a los ayudantes haciendo juguetes, podemos ver pelotas, un barco y un avión.

En la segunda planta observamos el baño.

Tercera planta:

Observamos el juguetero.

Cámara.

Se trabaja en una manera perspectiva y esta hecha de manera muy fácil para moverse dentro de nuestro escenario.

```
//Una vez hecho eso se lo mando a la biblioteca para que lo texturice.
GLuint cubemapTexture = TextureLoading::LoadCubemap(faces);

glm::mat4 projection = glm::perspective(camera.GetZoom(), (GLfloat)SCREEN_WIDTH / (GLfloat)SCREEN_HEIGHT, 0.1f, 1000.0f);

// ----------------------------------------------------------------Game loop
while (!glfwWindowShouldClose(window))
{
    // Calculate deltatime of current frame
```

Animaciones básicas.

En esta imagen vemos a muñeco de nieve que gira en rotación , el girara de manera automática en circulos en la pista de hielo.

```
//muneco de nieva girando

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(-40.0f, 2.5f, 0));
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(0.0f, 0.01f, 0.0f));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Sn.Draw(lightingShader);

//Nieve10
```

Las estrellas rotan en una dirección, pero si oprimimos la tecla 1, estas cambiarán de dirección ya que se les aplica una siguiente rotación, simulando la trayectoria de las estrellas que siempre tienen una dirección diferente.

```
Casax.Draw(lightingShader);

//-----------------------------------------Estrellas
//Estrellas0

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(1.0f, 0.0f, 5.0f));
model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::translate(model, glm::vec3(-20.0f, 10.5f, 0));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Star.Draw(lightingShader);

//Estrella-1

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(1.0f, 0.0f, 5.0f));
model = glm::rotate(model, glm::radians(rot), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::translate(model, glm::vec3(-20.0f, 10.5f, 23));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Star.Draw(lightingShader);

//Estrella-4
```

La Luna que cambiara con las teclas “z” y “x”, usa un movimiento de rotación y esta va cambiando de colores como se vayan apretando las teclas, inspirada en la práctica de los peces.

```
// Draw the loaded model
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-60.0f, 90.0f, 00));
model = glm::rotate(model, (float)glfwGetTime(), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
ourModel.Draw(lightingShader);
//ourModel.Draw(shader);
if (activeA) {

    if (avanza || retrocede) {
        switch (a) {
        case 0:
            a++;
            avanza = false;
            retrocede = false;
            break;
        case 1:
            a++;
            avanza = false;
            retrocede = false;
            break;

        case 2:
            a++;
            avanza = false;
            retrocede = false;
            break;
        case 3:
            a++;
            avanza = false;
            retrocede = false;
            break;
        }
```

Animaciones complejas.

El cascanueces este hecho con el tipo de animaciones de la práctica 11, usando KeyFrame's.

```
//Movimiento del personaje

if (play) // Ya esta activa la animacion ?
{
    //current ,reviso en que parte de la animacion me encuentro
    if (i_curr_steps >= i_max_steps) //end of animation between frames?
    {
        playIndex++;
        if (playIndex>FrameIndex - 2)   //end of total animation?
        {
            printf("termina anim\n");
            playIndex = 0;
            play = false;
        }
        else //Next frame interpolations
        {
            i_curr_steps = 0; //Reset counter
                              //Interpolation
            interpolation(); //Sino se ha terminado sigo calculando las interpolaciones..funcion interpolacion
        }
    }
    else
    {
        //Si ya culmino la animacion voy guardando el paso e incremento la rotacion o la posicion
        //Draw animation
        posX += KeyFrame[playIndex].incX;
        posY += KeyFrame[playIndex].incY;
        posZ += KeyFrame[playIndex].incZ;

        rotRodIzq += KeyFrame[playIndex].rotInc;
        rotRodDer += KeyFrame[playIndex].rotInc2;
```

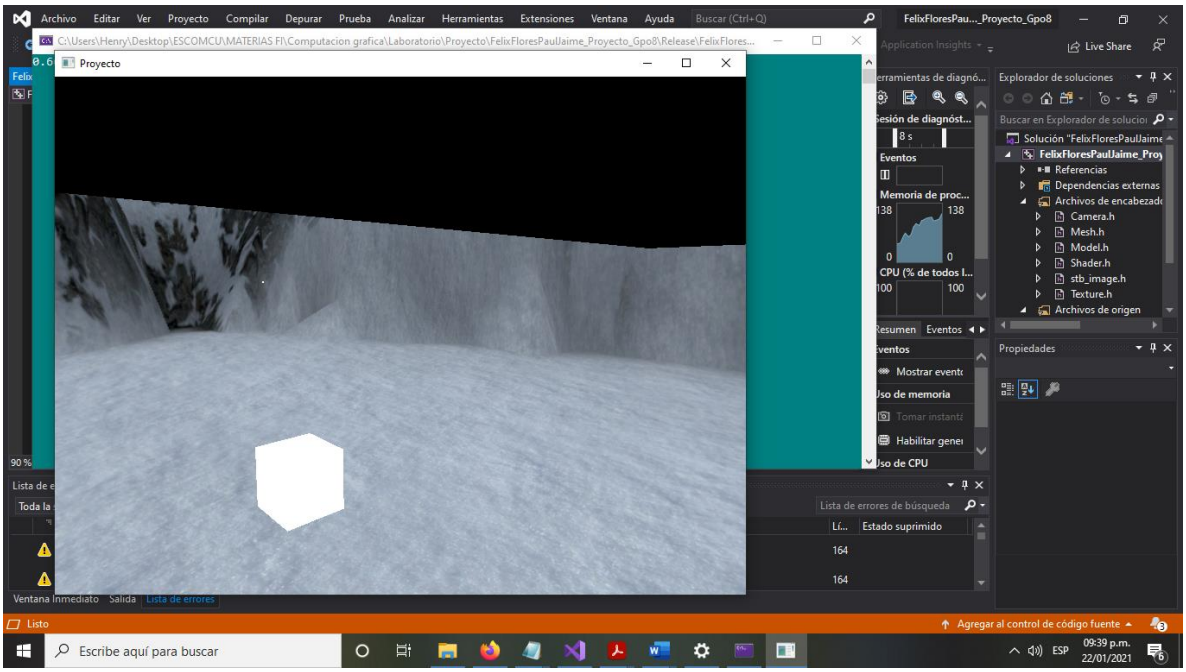
El trineo.

Se mueve con MovKit, como la práctica 10 del carro, sigue una lógica similar, aunque con un recorrido muy peculiar a un trineo navideño de acuerdo a la temática propuesta.

```
            }

        }

        //  //Movimiento del trineo
        //Verifico que el circuito este activo
        if (circuito)
        {
            if (recorrido1)
            {
                rotKit = 90;
                movKitX += 1.1f; // El reocrrido se incrementa de .1 en.1
                if (movKitX > 90) //Hasta ser mayor a 90
                {
                    recorrido1 = false; // El recorrido 1 termina
                    recorrido2 = true; // Entonces empieza el recorrido 2
                }
            }
            if (recorrido2)
            {
                rotKit = -45;
                movKitZ += 1.1f;
                movKitX -= 1.1f;
                if (movKitZ > 90)
                {
                    recorrido2 = false;
                    recorrido3 = true;

                }

            }

            if (recorrido3)
```

Iluminación.

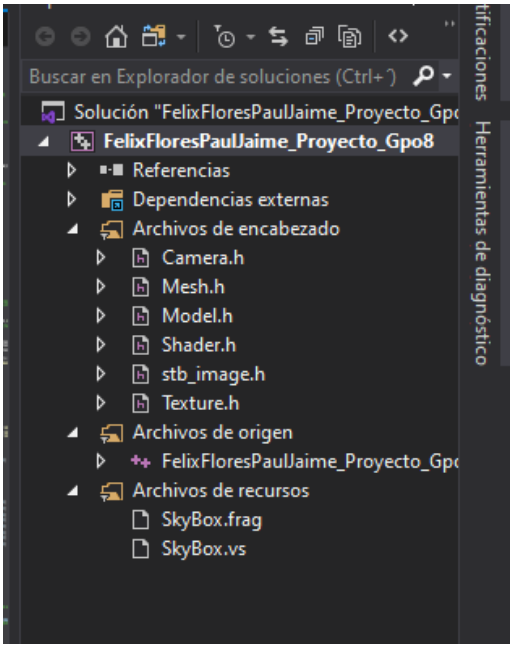
Está ubicada en la parte de debajo del plano.




```
// == ==========================
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 0.3f, 0.3f, 0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.1f, 0.1f, 0.1f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"), 0.4f, 0.4f, 0.4f);
// Point light 1
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLig
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), LightP1.x, LightP1.y, LightP1.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), LightP1.x, LightP1.y, LightP1.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 0.032f);
// Point light 2
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].position"), pointLightPositions[1].x, pointLig
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].diffuse"), 0.5f, 0.5f, 0.5f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].specular"), 1.0f, 1.0f, 0.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].quadratic"), 0.032f);
// Point light 3
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].position"), pointLightPositions[2].x, pointLig
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].diffuse"), 0.0f, 0.5f, 0.5f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].specular"), 0.0f, 1.0f, 1.0f);
```

Notas.

Podemos varias configuraciones que se hicieron en las practicas pasadas, pero trabajando de manera junta, estas configuraciones quedaron igual a la práctica 11, ya que todo se englobaba aquí, por lo cual no fue necesario agregar más cosas.



He comentado mi código en mi proyecto para ayudarme a reforzar los conocimientos de las practicas hechas.

La mayoría de las animaciones se sacaron de la siguiente página, aunque venían separadas.

https://www.turbosquid.com/

En cuanto a imágenes para renderizar pues de Google.

Use GIMP, así como Geogebra para auxiliarme en la elaboración del proyecto.

Conclusión:

He aprendido mucho en este proyecto ya que puse en marcha todas las prácticas, pero con modificaciones de acuerdo a la propuesta de proyecto. Me he divertido trabajando, nunca me estrese en este proyecto por lo visual que llega hacer, y al ser una persona muy visual me imaginaba ciertos escenarios y me gustaba crearlos, así que deje volar mi imaginación y pues así fue como quedo, al tener conocimiento con las prácticas echas en el semestre, pues fue un proyecto fácil en el sentido de no serlo tan pesado en cuanto a la programación, pero si largo por el hecho de hacer que todo coexista en uno solo.