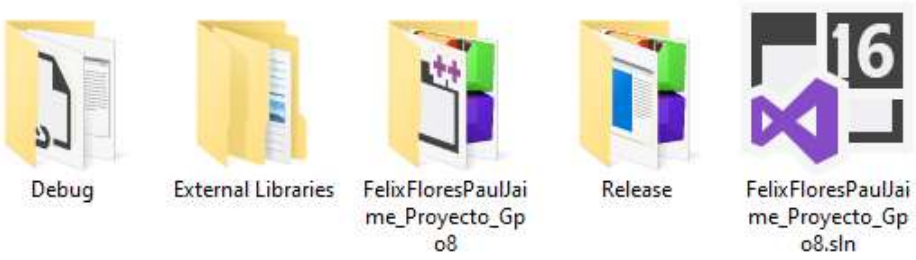
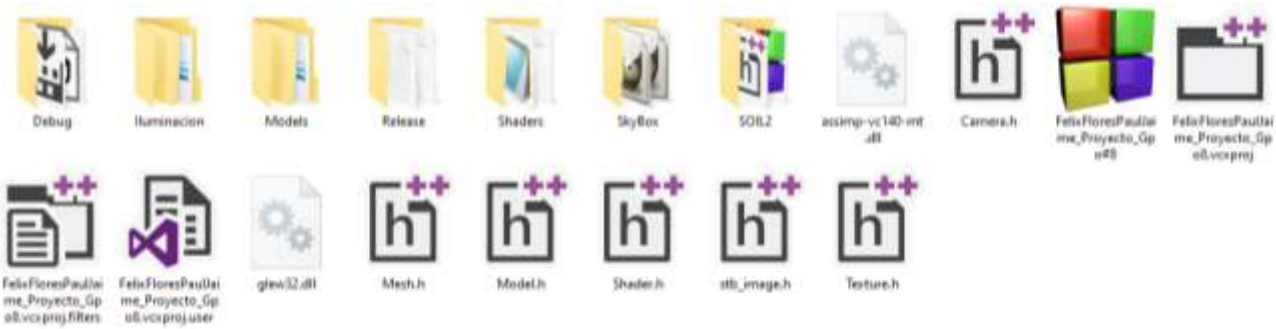
Ordenamiento de carpetas.

Para compilar el proyecto, una vez descargadas las carpetas de GitHub tenemos que tener este orden, es muy importante para que no cause problemas.

Dentro de la carpeta principal estarán estas carpetas, así como el ejecutable llamado Release.

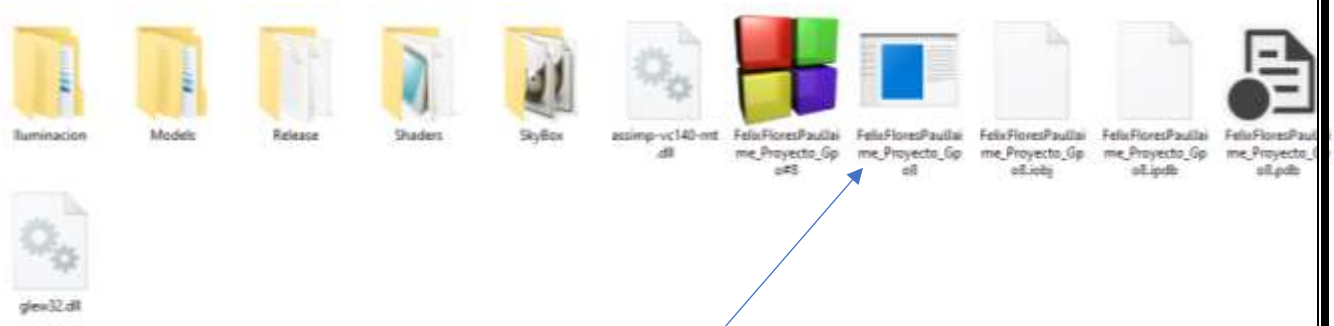


Dentro de la carpeta del proyecto 8 encontraremos todas estas carpetas a excepción de la carpeta Debut, que aparecerá después de compilarlo.



Ejecutable.

Dentro de la carpeta Release tendremos el ejecutable.

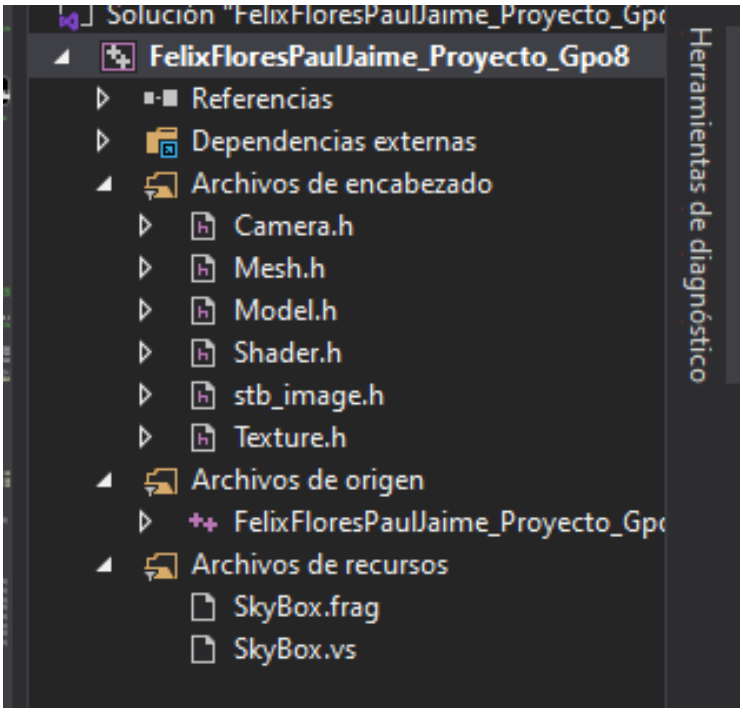


Al abrir este ejecutable el proyecto se nos mostrara

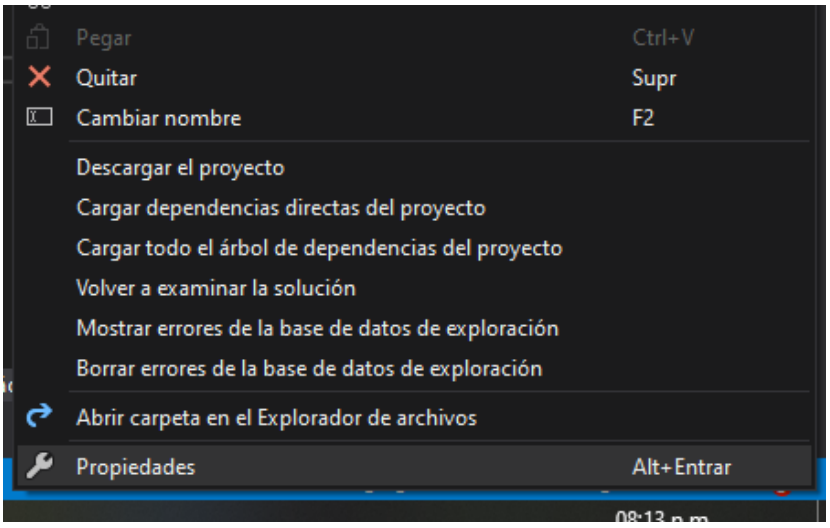
Configuración para compilar.

Para crear la carpeta proyecto ,tendremos en consideracion estas configuraciones.

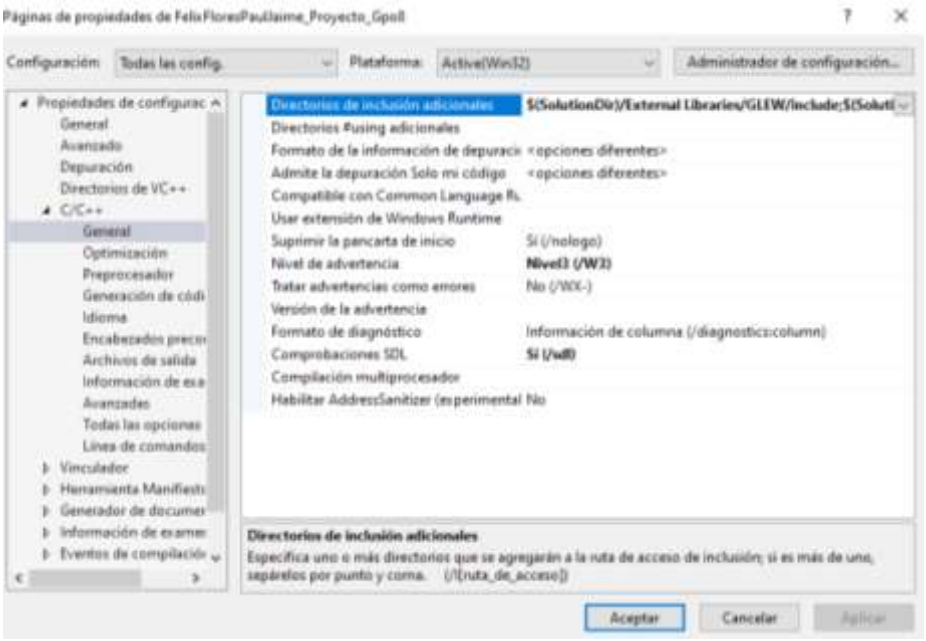
Una vez cargado todo esto:



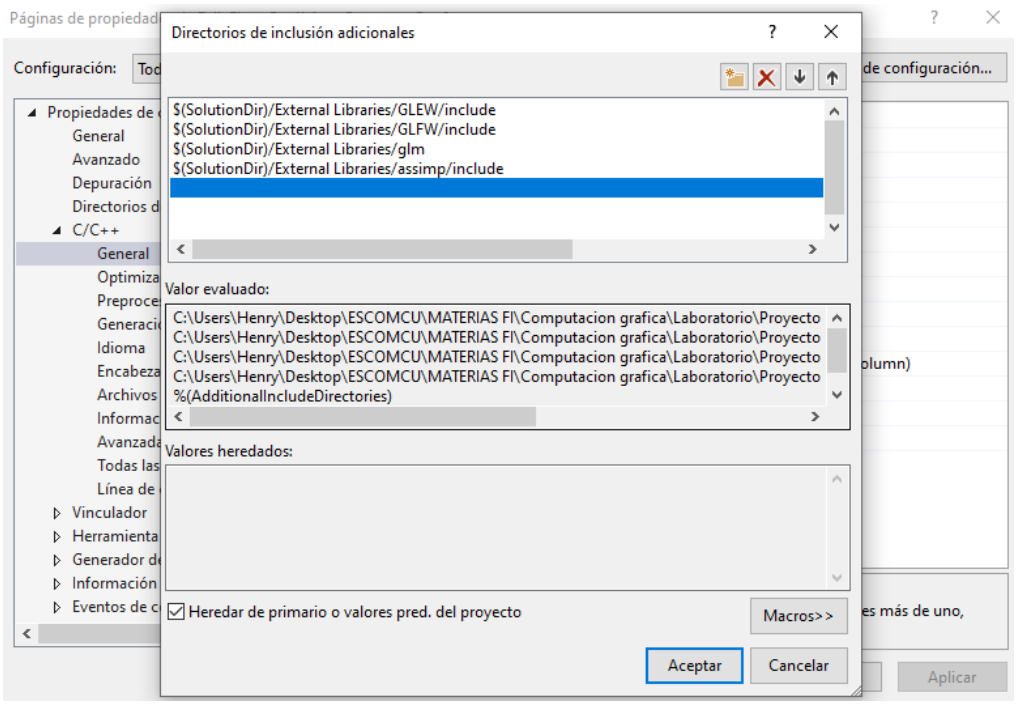
Haremos click derecho en el proyecto y configuraremos en propiedades.



Una vez hecho esto, haremos lo siguiente:



1.-Primer nos iremos a General y pondremos en directorios de inclusión adicionales y agregaremos los siguientes directorios.



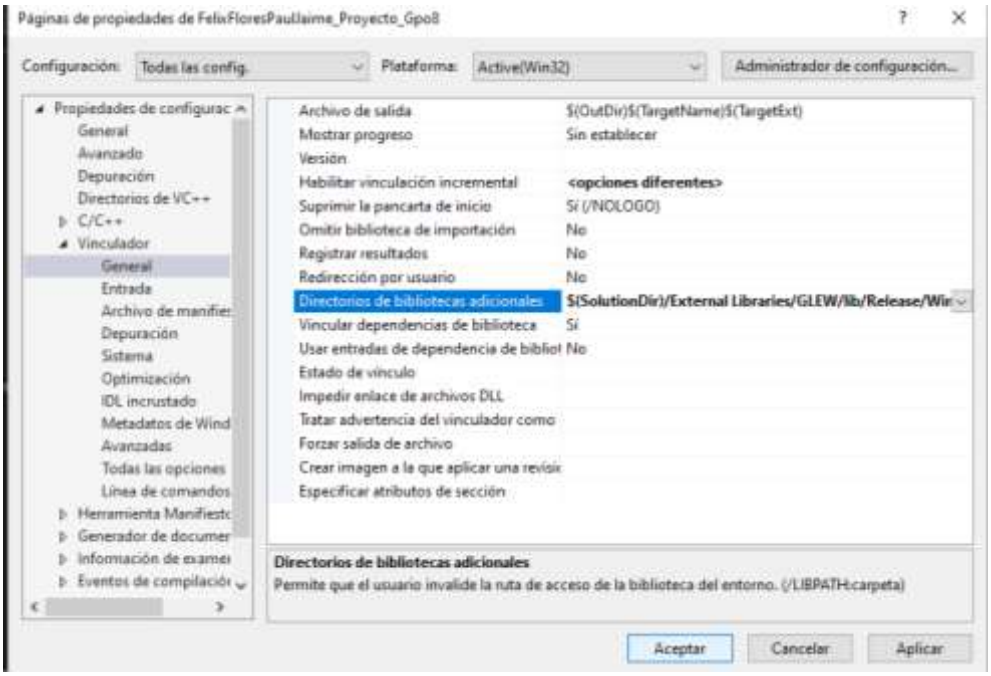
$(SolutionDir)/External Libraries/GLEW/include

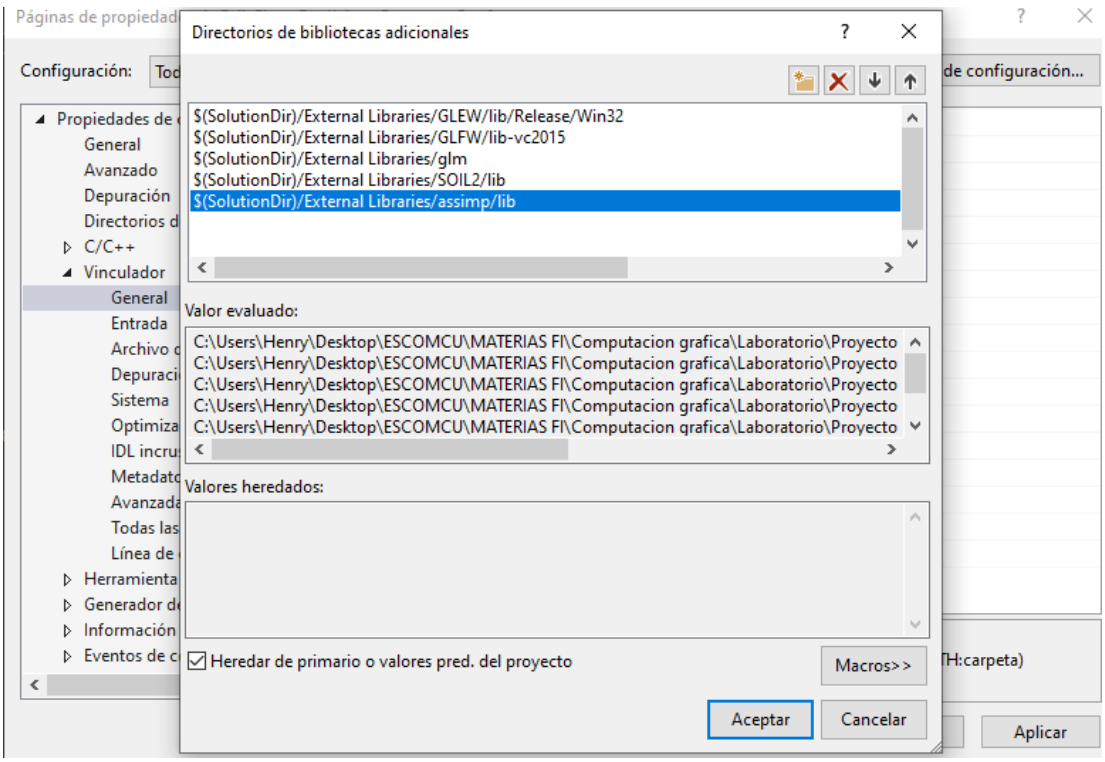
$(SolutionDir)/External Libraries/GLFW/include

$(SolutionDir)/External Libraries/glm

$(SolutionDir)/External Libraries/assimp/include

2.-Iremos a la parte de vinculador ,iremos ahora a directorios de bibliotecas adicionales y pondremos lo siguiente.





$(SolutionDir)/External Libraries/GLEW/lib/Release/Win32

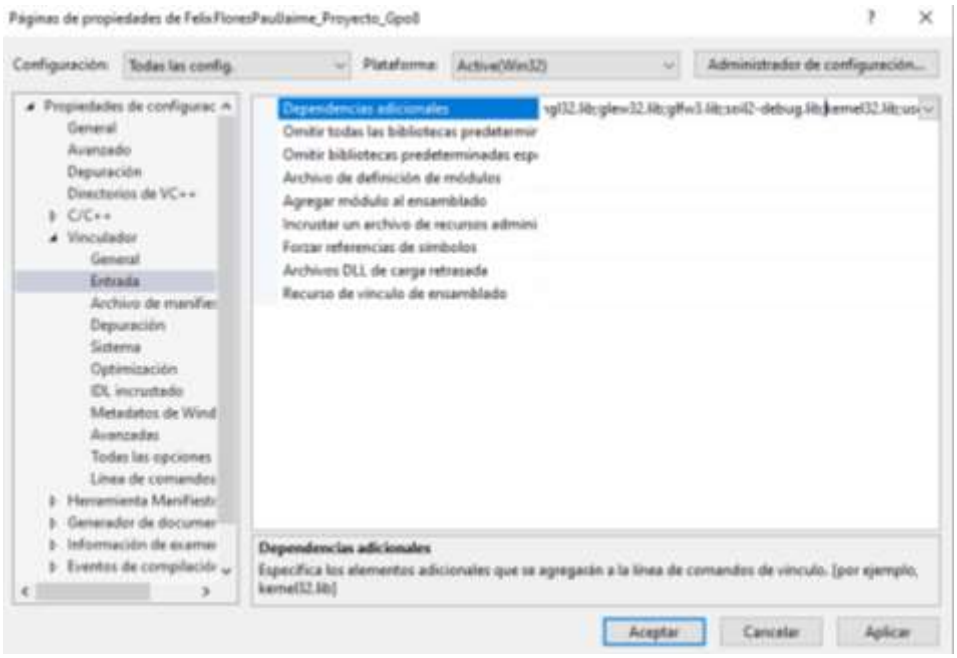
$(SolutionDir)/External Libraries/GLFW/lib-vc2015

$(SolutionDir)/External Libraries/glm

$(SolutionDir)/External Libraries/SOIL2/lib

$(SolutionDir)/External Libraries/assimp/lib

3.-Por último, iremos a entrada e iremos a Dependencias adicionales y agregaremos estas dependencias:



assimp-vc140-mt.lib;opengl32.lib;glew32.lib;glfw3.lib;soil2-debug.lib;

Cámara.

Para movernos dentro del escenario, nos podremos mover con la tecla w par acercarse a las animaciones y la tecla s para alejarnos y nos podremos mover con el raton.

Al mismo tiempo podeos desplazar las teclas, up,down,right,y left para movernos dentro del escenario.

Implementación

En el proyecto podemos observar la aldea navideña, fachadas de casa y distintas animaciones.

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

La casa tiene la siguiente estructura por dentro. Cuenta con una cocina, sala, comedor, televisión, un librero y las dos recamaras

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.

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
```

Estrellas.

Las estrellas rotan en una dirección, pero si oprimimos la tecla 1, estas cambiarán de dirección.

```
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
```

El taller de santa, esta conformado por 4 niveles, en la parte de arriba se encuentran las siguientes animaciones, este hecho desde cero.

En esta imagen podemos ver tres animaciones, la luna que cambiara con las teclas “z” y “x”, z para avanzar hacia adelante y con x hacia atrás. Esta animación juega un papel muy importante con la siguiente animación que es el general “Cascanueces”, en esta el General cascanueces solo aparecerá en luna roja.

```
// 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;
        }
```

El cascanueces moverá las articulaciones con las teclas:

2: Pierna derecha hacia delante.

3: Pierna derecho hacia atrás.

4: Pierna izquierda hacia delante.

5: Pierna izquierda hacia atrás.

6: Brazo derecho hacia delante.

7: Brazo derecha hacia atrás.

8: Brazo izquierdo hacia delante.

9: Brazo izquierdo hacia atrás.

Con la tecla k, salvaremos los movimientos que haga, y con la tecla L, podremos ejecutar todo el movimiento.

Ojo: solo podremos ver 9 movimientos guardados.

```
//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;
```

En esta imagen podemos ver el trineo, este se mueve con la tecla I y para con la tecla O, hace un recorrido en infinito, pero no trata de ser predictivo porque es un trineo mágico de acuerdo a la temática que estoy manejando.

```
            }

        }

        //  //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)
```

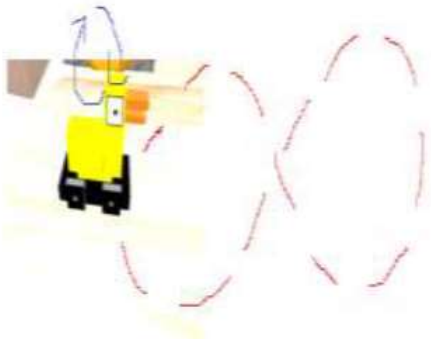
Calabaza

Esta animación hace un recorrido de un lado al otro y gira cuando llega a un extremo, además va dado “saltos”.
La animación de la calabaza usa las teclas
B para iniciar la animación
M para terminar la animación

```
//Animacion Calabaza
if (anim2) {
    if (rotcalabaza) {
        if (dircalabaza) {
            movcy += 0.005;
            movcz -= 0.003;
            if (movcy > 2.0)
                movcy = 0.0;
        }
        else {
            movcy += 0.005;
            movcz += 0.003;
            if (movcy > 2.0)
                movcy = 0.0;
        }

        if (dircalabaza && movcz < -3.0) {
            dircalabaza = false;
            rotcalabaza = false;
            movcz = -3.0;
        }if (!dircalabaza && movcz > 3.0f) {
            dircalabaza = true;
            rotcalabaza = false;
            movcz = 3.0;
        }
    }
    else {
        rotc += 0.2;
        if (!dircalabaza) {
            if (rotc > 180.0f) {
                rotcalabaza = true;
```

ANIMACIÓN Pato

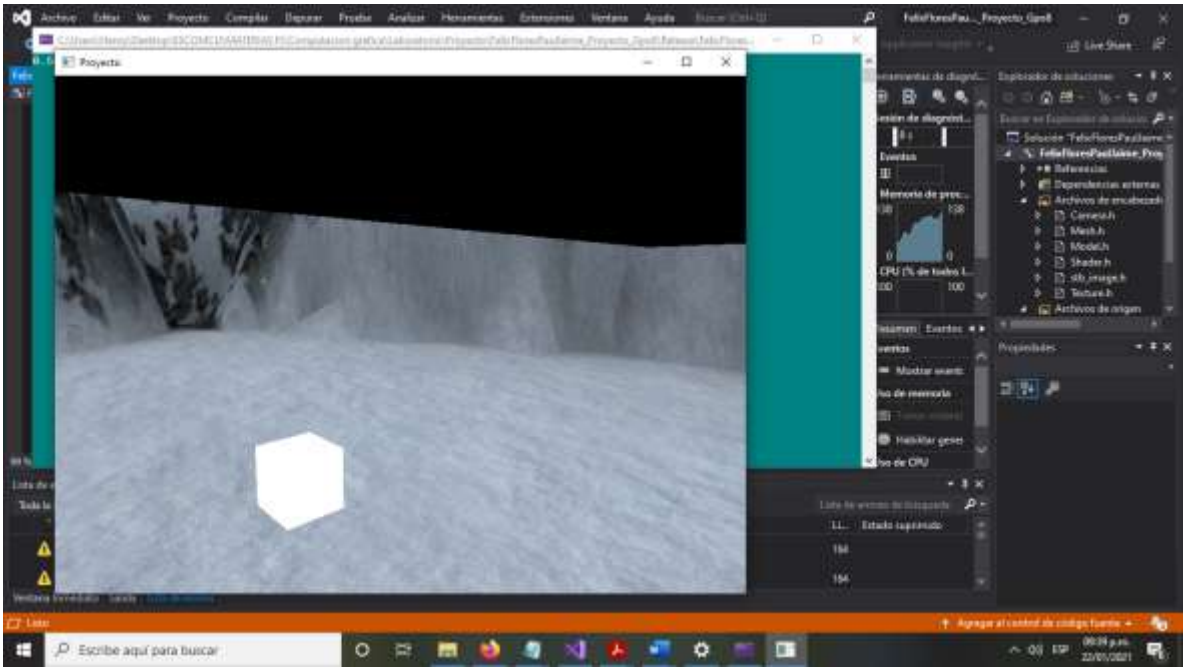
Para la animacion se hizo uso de las teclas
Z para iniciar la animación
C para terminar la animación
Esta animación simula la forma de un infinito, el pato también va girando.



```
//Animación Pato
if (activeP == true) {

/*if (activeP==false) {
    rotp = 0.0f;

}*/


if (a) {
    xpato = 20 + 4 * sin(2 * tpato) / 2;
    ypato = 22 + 4 * sin(tpato);
    tpato += 0.001;
    rotpato += 0.06;
}
}
```

Iluminación.

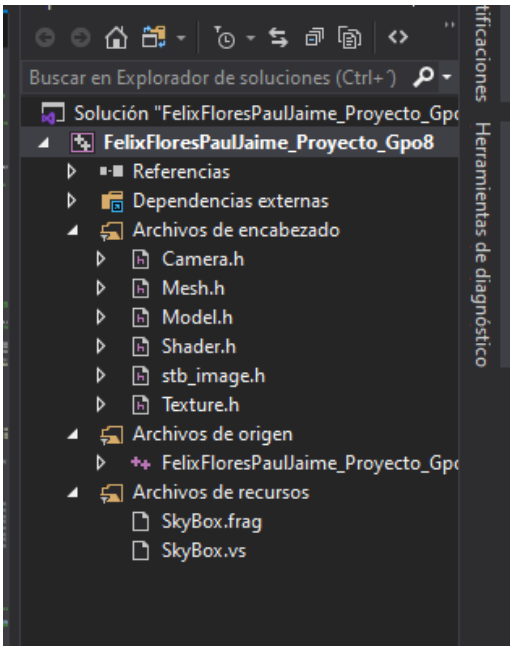
Está ubicada en la parte de debajo del plano.



```
// == ============================
// 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.



Hemos comentado el código en el proyecto para ayudarnos 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