ORDEN :

void MyGLWidget::initializeGL() {

initializeOpenGLFunctions(); == SE TIENE QUE PONER SIEMPRE PARA PONER LAS VARIABLES EN OPENGL

DECLARACION DE TODAS LAS VARIABLES

glClearColor(cosas); // LIMPIAMOS EL CODIGO

glEnable(cosas); //SIRVE PARA QUE SE VEAN LAS COSAS SEGUN PROFUNDAD Y NO ORDEN DE PINTADO

carregaShaders(); //FUNCION QUE ESTA MAS ABAJO \*\*\*\*2

createBuffers();

ini\_camera();

}

\*\*\*\*2

SHADERS == INDICAS COMO PINTAR LAS COSAS (LAMBERT Y PHONG, MANERAS DE ILUMINAR)

vertshad.vert ==> EN ESTE PROGRAMA PONERMOS EL ORDEN EN EL MAIN, EL ORDEN ES SIEMPRE

VOID MAIN() {

fcolor = matdif;

gl\_Position = proj \* view \* TG \* vec4 (vertex, 1.0];

}

vec4 ==> funcion de open gl que te crea un vector de 4 posiciones, vec3 te la crea de 3 y asi sucesivamente.

mat4 ==> crea una matriz de 4x4, mat3 crea una matriz de 3x3 y asi sucesivamente.

¡¡¡ATENCION!!! NO EXISTE NI vec2, NI vec1, LO MAS BAJO ES vec3.

CON UN vec3 Y UN NUMERO SUELTO, PUEDO CREAR UN vec4, MIRAR TEXTO EJEMPLO DEL VOID MAIN DE ARRIBA.

CUANDO EN UNA VARIABLE LE PONES DELANTE in tipo\_variable nombre\_variable ESTAS INDICANDO QUE ESTA VARIABLE TE VIENE DE FUERA DEL PROGRAMA, EN CAMBIO, CUANDO PONES out tipo\_variable nombre\_variable SIGNIFICA QUE ESTA VARIABLE LAS VAS A USAR EN OTRO CODIGO FUERA DE ESTE PROGRAMA.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LAS VARIABLES SE APUNTAN EN EL .h

FUNCION CARREGASHADERS():::::

GLuint ESTAS VARIABLES QUE DECLARAMOS EN EL .h LAS MODIFICAREMOS ASI:

NOMBRE\_DE\_LA\_VARIABLE\_EN\_EL\_.h = glGetAttribLocation(program->programId(), "nombre\_que\_usare\_en\_el\_programa\_vertshad.vert");

((SE TIENE QUE REPETIR LA LINIA DE CODIGO ANTERIOR UNA VEZ PARA CADA VARIABLE)).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCION INI\_CAMERA()::::: FUNCION QUE SIRVE PARA PREPARAR EL VIEWTRANSFORM

projectTtransform();

OBS = glm::vec3(X,Y,Z); // POSICIONES XYZ DE LA CAMARA (PERSONA QUE MIRA LA FIGURA)

VRP = glm::vec3(X,Y,Z); // OBJETO QUE ES MIRADO POR LA CAMARA

UP = glm::vec3(X,Y,Z); // VERTICAL QUE SE SITUA SOBRE LA CABEZA (EJEMPLO DEL CUERNO EN TU CABEZA)

viewTransform(); // HACE EL MOVIMIENTO DE LA CAMARA EN FUNCION DE LAS VARIABLES ANTERIORES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCION VIEWTRANSFORM():::::

glm::mat4 View = glm::lookAt (OBS, VRP, UP); // CON LOOKAT NOS TELETRANSPORTAMOS A LA POSICION QUE INDICAMOS CON LAS VARIABLES SELECCIONADAS. SI EN VEZ DE USAR LOOKAT USAMOS TRANSLATE LO QUE HACE ES DESPLAZARTE DESDE TU POSICION LE SUMAMOS LAS NUEVAS QUE LE PASAMOS COMO VALORES.

## EJEMPLO: CAMARA EN (3,2,1)

LOOKAT (3,3,3) ==> CAMARA EN (3,3,3)

TRANSLATE (3,3,3) ==> CAMARA EN (6,5,4)

###

View = glm::rotate(View, -angleY, glm::vec3(0, 1, 0)); //

View = glm::rotate(View, -angleX, glm::vec3(1, 0, 0)); //

glUniformMatrix4fv (viewLoc, 1, GL\_FALSE, &View[0][0]); //

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCION CREATEBUFFERS()::::::

VAO == TODO LO QUE HAYA DENTRO DEL VAO, SE MODIFICARA SI MODIFICAMOS UNA, SI HAY 3 FIGURAS, SE MODIFICARAN LAS 3. UN VAO PUEDE TENER VARIOS VBO.

NO SE PUEDE NO PONER UN VAO DENTRO DE OTRO VAO.

\*\*\*\*\*\*\*\*\*\*

GENERAR VAO:

--> glGenVertexArrays("NUMERO DE VAO QUE CREAMOS (1 = 1 SOLO VAO, +1 ES UNA ARRAY DE VAO)", "NOMBRE DEL VAO"); ## NO HACE FALTA PONER LOS ""

\*\*\*\*\*\*\*\*\*\*

--> glBindVertexArray(nombre del VAO);

TODAS LAS FUNCIONES (LAS DE GL) Y TODO LO QUE MODIFIQUE AHORA, SERA SOBRE EL VAO QUE HAYA INDICADO EN ESTA FUNCION.

--> glBindVertexArray(0); --> sirve para acabar de modificar VAO

\*\*\*\*\*\*\*\*\*\*

GENERAR VBO:

--> glGenBuffer("NUMERO DE VBO QUE CREAMOS (1 = 1 SOLO VBO, +1 ES UNA ARRAY DE VBO)", "NOMBRE DEL VBO")

\*\*\*\*\*\*\*\*\*\*

--> glBindBuffer(nombre del VBO);

TODAS LAS FUNCIONES (LAS DE GL) Y TODO LO QUE MODIFIQUE AHORA, SERA SOBRE EL VBO QUE HAYA INDICADO EN ESTA FUNCION. (¡¡¡ATENCIÓN!!! TODO LO DE ESTE VBO SIEMPRE ESTARA DENTRO DEL VAO INDICADO ORIGINALMENTE).

glBufferData(GL\_ARRAY\_BUFFER, toda la info de la figura en la posicion, colores, FUNCION PARA PINTAR);

\*\*\*\*EXTRA

---------------------------------------------- Configuracion de Material --------------------------

//Propiedades del suelo

// Definim el material del terra

glm::vec3 amb(0.2,0,0.2); // Set luz

glm::vec3 diff(0.8,0,0.8); // color deseado en RGB

glm::vec3 spec(0,0,0); // Set Cuanto brilla

float shin = 100; // Set brillo

-----------------------------------------------NOTAS-----------------------------------------------

\*\*\*\*EXTRA: patri.load() == CARGAMOS EL PATRICIO

\*\*\*\* AMB = ES EL COLOR DEL AMBIENTE (ES EL COLOR DEL OBJETO SUELO)

\*\*\*\* SHIN = CUANTO VA A BRILLAR EL OBJETO SUELO

\*\*\*\* ESPEC = PUNTO EN EL CUAL EL OBJETO VA A BRILLAR MAS, TODOS LOS FOCOS APUNTAN A ESE PUNTO

\*\*\*\* DIFF = RGB

----------------------------------------------BLOQUE 3-------------------------------------------

//Esta version es para que se vea iluminado desde la posicion inicial.

//VertexShader Main

vec4 VertexSCO = view\* TG \* vec4(vertex,1.0);

vec3 normalSCO = normalize(inverse(transpose(mat3(view\*TG))) \* normal);

vec4 posFocusSCO = view \* vec4(posFocus,1.0);

// vec4 posFocusSCO = vec4(posFocus,1.0);

//Podemos dejar el PosFocus sin multiplicar por la view cosa que hara que por todos los

// lados que veamos se vea iluminado

vec3 L = normalize(vec3(posFocusSCO - VertexSCO));

fcolor = Phong(normalSCO,L,VertexSCO);

gl\_Position = proj \* view \* TG \* vec4 (vertex, 1.0);

// MyGLWidget.cpp

void MyGLWidget::FocusTransform(){

glUniform3fv (posFocusLoc, 1, &posFocus[0]);

glUniform3fv (colFocusLoc, 1, &colFocus[0]);

glUniform3fv (llumAmbientLoc, 1, &llumAmbient[0]);

}

// !!!!! IMPORTANTE !!!!!! //

// Anadir llamada a esta funcion cuando cargamos todo porque sino no le pasara lso valores inciales que tienen los atributos de la camara y saldra la escena negra.

//MyGL Widget.h

//Declaracion de la funcion anterior

void FocusTransform();

//Variables

GLuint PosFocusLoc,ColorinFocus,LuzFocus;

glm::vec3 colFocus = glm::vec3(0.8, 0.8, 0.8);

glm::vec3 llumAmbient = glm::vec3(0.2, 0.2, 0.2);

glm::vec3 posFocus = glm::vec3(1, 0, 1); // en SCA

// Si queremos mover la camara por el eje de las x

// En la funcion de : void MyGLWidget::keyPressEvent(QKeyEvent\* event)

//Anadimos un caso para cambiar el valor de posFocus.

case Qt::Key\_T: { // canvia òptica entre perspectiva i axonomètrica

posFocus[0] += 0.5; // Para el caso de querer moverlo para +x

posFocus[0] -= 0.5; // moverlo para -x.

FocusTransform ();

break;

}

---------------------------------- TEMA ILUMINACION

mat3 nMatrix = inverse(transpose(mat3(view \* TG)));  
vec3 NormSCO = normalize(nMatrix \* normal);  
vec4 vertexSCO = view \* TG \* vec4(vertex, 1.0);  
vec4 focusSCO = view \* vec4(posFocus, 1.0);  
vec4 L = focusSCO - vertexSCO;  
vec3 Lxyz = normalize(L.xyz);  
fcolor = Lambert (NormSCO, Lxyz); (LAMBERT O PHONG, EL QUE USEMOS)

FUNCIONES PHONG Y LAMBERT CALCULAN EL FOCO DE LUZ, SE LE TIENEN QUE PASAR COORDENADAS DEL OBSERVADOR

// Valors per als components que necessitem dels focus de llum

vec3 colFocus = vec3(1, 1, 1); // COLOR DEL FOCO

vec3 llumAmbient = vec3(0.2, 0.2, 0.2); // LUZ AMBIENTE

vec3 posFocus = vec3(5, 3, 5); // en SCA // POSICION DONDE ESTA LA LUZ

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCIONES DE LAMBERT Y DE PHONG (EL LEÑADOR Y EL CHINO)

vec3 Lambert (vec3 NormSCO, vec3 L)

{

// S'assumeix que els vectors que es reben com a paràmetres estan normalitzats

// Inicialitzem color a component ambient

vec3 colRes = llumAmbient \* matamb;

// Afegim component difusa, si n'hi ha

if (dot (L, NormSCO) > 0)

colRes = colRes + colFocus \* matdiff \* dot (L, NormSCO);

return (colRes);

}

vec3 Phong (vec3 NormSCO, vec3 L, vec4 vertSCO)

{

// Els vectors estan normalitzats

// Inicialitzem color a Lambert

vec3 colRes = Lambert (NormSCO, L);

// Calculem R i V

if (dot(NormSCO,L) < 0)

return colRes; // no hi ha component especular

vec3 R = reflect(-L, NormSCO); // equival a: normalize (2.0\*dot(NormSCO,L)\*NormSCO - L);

vec3 V = normalize(-vertSCO.xyz);

if ((dot(R, V) < 0) || (matshin == 0))

return colRes; // no hi ha component especular

// Afegim la component especular

float shine = pow(max(0.0, dot(R, V)), matshin);

return (colRes + matspec \* colFocus \* shine);

}