Account number: 417086611
Group: 12

Computer graphics and human-computer interaction Laboratory.

Final Project.

Index.

Account number: 417086611
Group: 12

Goals:

- The first objective of this project was undoubtedly to experiment, learn, get to know and even have fun throughout the construction of the project.
- To familiarize ourselves with 3D modeling tools such as: Maya, and image editing tools such as: GIMP. For the construction of models and their subsequent texturing.
- To put into practice the concepts learned during the laboratory sessions: basic transformations, camera handling, model loading and drawing, lighting, simple and complex animation, etc.

Gantt diagram:

To keep track of the progress of our project, a simple Gantt diagram was developed.

Gantt diagram									
Duration time									
	Start date: 04/04/2022	April				May			
Activity	Finish date: 12/05/2022	1	2	3	4	1	2	3	4
Reference image	04/04/2022								
Model 1	15/04/2022								
Model 2	15/04/2022								
Model 3	22/04/2022								
Model 4	22/04/2022								
Model 5	29/04/2022								
Model 6	29/04/2022								
Model 7	06/05/2022								
Room	07/05/2022								
Fecade	07/05/2022								
SkyBox	08/05/2022								
Lighting	08/05/2022								
Animation	11/05/2022								
Executable creation	12/05/2022								
Creation of manuals	12/05/20222								

In the project, there were certain peculiarities, personally I think it was a mistake to go making model by model every week, instead of making the fecade and the room as quickly as possible, that took me a lot of time, because I was trying to be as meticulous as possible with my models and when I least thought about it; I had the time on my hands.

Project scope.

This project aims to cover the basic items seen in the lab sessions, from the inclusion of multiple models which using basic transformations: translation, rotation and scaling, are well placed on the stage giving the closest possible resemblance to the reference image. Implementation of the three

types of lighting: spotlight, pointlight and directional light, and a striking effect in which the pointlights oscillate from one color to another, and the inclusion of simple animations mainly rotations and some complex ones giving an extremely striking effect to some models. We did not include circuit type animations, or even more complex ones that have to do with physical phenomenom such as: free fall, parabolic shot, etc, or the inclusion of textures with transparency or Phong type materials, which can show another image when the light hits them.

Limitations.

Personally I consider that it was my little planning in the project, sincerely I thought I had more time but between work and school everything came on top, it is something I regret since I have this challenge of wanting to implement complex animations that have to do with some mathematical equation that describes certain physical phenomenon, to have used more textures with transparencies and thus enable the alpha channel giving more colorful effects to my project.

Code documentation.

The code has a total of 1522 lines of code, so we will not explain line by line but we will cover the most important items: model loading and drawing, lighting, and animation.

Model loading.

```
230    Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
231    Shader lampShader("Shaders/lamp.vs", "Shaders/lamp.frag");
232    Shader SkyBoxshader("Shaders/SkyBox.vs", "Shaders/SkyBox.frag");
233    Shader Anim("Shaders/anim.vs", "Shaders/anim.frag");
234    Shader Anim2("Shaders/anim2.vs", "Shaders/anim2.frag");
235    Shader Anim3("Shaders/anim3.vs", "Shaders/anim2.frag");
236    Shader Anim4("Shaders/anim4.vs", "Shaders/anim2.frag");
237    // Carga de modelos
238
239    Model desk((char*)"Models/MLars/escritoriolars.obj");
240    Model cajon((char*)"Models/MLars/cajon.obj");
241    Model espada1((char*)"Models/MLarsEspada/espada1lars.obj");
242    Model espada2((char*)"Models/MLarsEspada/espada2lars.obj");
243    Model espada3((char*)"Models/MLarsEspada/espada3lars.obj");
244    Model arco((char*)"Models/MLarsEspada/arco.obj");
245    Model escudo1((char*)"Models/MLarsEscudo/escudo1.obj");
246    Model sofa((char*)"Models/MLarsSofa/sofa.obj");
247    Model reloj((char*)"Models/MLarsReloj/reloj.obj");
248    Model pendulo((char*)"Models/MLarsReloj/pendulo.obj");
249    Model piso((char*)"Models/MLarsPiso/piso.obj");
250    Model pisoint((char*)"Models/MLarsPiso/pisoint.obj");
251    Model camino((char*)"Models/MLarsPiso/camino.obj");
252    Model chimenea((char*)"Models/MLarsChimenea/chimenea.obj");
253    Model cuenco((char*)"Models/MLarsChimenea/cuenco.obj");
254    Model flama((char*)"Models/MLarsChimenea/flama.obj");
255    Model letras((char*)"Models/MLarsLetras/letras.obj");
256    Model Room1((char*)"Models/MLarsCuartoint/Kitroom1.obj");
```

The models are loaded in the main function (), in the Model class all the models to be used are added, a local variable is assigned to them, which can be called according to your choice, and a pointer is used indicating the path where the model is located, very important; the model has to be in obj format. Once the models are loaded, we proceed to draw them.

Account number: 417086611
Group: 12

Model drawing.

```
//Carga de modelo
//Personaje
view = camera.GetViewMatrix();
glm::mat4 model(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::scale(model, glm::vec3(2.75f, 0.0f, 2.0f));
model = glm::translate(model, glm::vec3(0.0f, -5.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
piso.Draw(lightingShader);
glBindVertexArray(0);

model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0, 0.0f));
model = glm::scale(model, glm::vec3(0.70f, 1.0f, 1.0f));
model = glm::translate(model, glm::vec3(-27.25f, 0.1f, 7.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
camino.Draw(lightingShader);
glBindVertexArray(0);
```

The first model to load must declare the use of the view matrix and model, the model matrix is always set, and using tmp places us in the center of the stage, you can make the transformations that are but always at the end you have to tell the model matrix the changes that were made either: a rotation, translation or scaling. We draw the model with the variable name that we have chosen and very important the Draw function receives as parameter a Shader, in our case it will be the lightingShader. We repeat the same with each model with the exception of some models.

Lighting.

```
// Directional light
//Creación de la luz direccional
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 0.5f, 0.5f, 0.5f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.1f, 0.1f, 0.1f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"), 1.0f, 1.0f, 1.0f);
```

Directional light is a light that affects the entire stage and is composed of the components: ambient, diffuse and specular.

```
// Point light 0
glm::vec3 lightColor;
lightColor.x = abs(sin(glfwGetTime() * Light1.x));
lightColor.y = abs(sin(glfwGetTime() * Light1.y));
lightColor.z = sin(glfwGetTime() * Light1.z);


glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y, pointLightPositions[0].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), lightColor.x, lightColor.y, lightColor.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), lightColor.x, lightColor.y, lightColor.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), 1.0f, 1.0f, 0.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.07f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 1.8f);
```

The pointlight has a special variable called lightColor, that variable will allow us to give the effect that the light varies its color. The points of light receive the position, the ambient, diffuse, specular

component and the constant, linear and quadratic values. Depending on how these values change, the light can be dimmer or brighter, cover a wider range, etc.

```
// SpotLight
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.position"), camera.GetPosition().x, camera.GetPosition().y, camera.GetPosition().z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.direction"), camera.GetFront().x, camera.GetFront().y, camera.GetFront().z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.ambient"), 1.0f, 1.0f, 1.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.diffuse"), 1.0f, 1.0f, 1.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.specular"), 2.0f, 2.0f, 2.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.linear"), 0.35f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.quadratic"), 0.44f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.cutOff"), glm::cos(glm::radians(12.5f)));
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.outerCutOff"), glm::cos(glm::radians(15.0f)));
```

The spotlight is similar to the pointlight only that it is limited by a specific angle, the first two are important because they will give the impression that we have a flashlight, that is to say, as we move the spotlight will also move. It receives almost the same parameters as the pointlight only has two new ones: cutOff and outerCutOff.

Animations.

Door rotation.

```
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(7.35f, 0.085f, -1.35f));
model = glm::rotate(model, glm::radians(rot), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Puerta.Draw(lightingShader);
glBindVertexArray(0);
```

We draw the door that will rotate certain radians (rot) on the x-axis, it is extremely important in this type of animations to place the pivot very well before exporting the model.

For the animation part, we need a variable rot of type float, which will be the rotation value of the door, it starts at 0.0f, and the second variable of type bool, which will allow me to manage the animation.

```
float rot = 0.0f;
bool abierto = false;
```

```
if (keys[GLFW_KEY_O])
{
    abierto = !abierto;
}
if (abierto) {
    if (rot < 30.0f)
        rot += deltaTime * 25;
}
else {
    if (rot > 0.0f)
        rot -= deltaTime * 25;
}
```

That part is the most important, each time the O key is pressed, the door will rotate up to 30 degrees on the x-axis, in a time deltaTime * 25, that is, it will rotate 25 times faster. When it reaches that degree of rotation, then pressing the O key again the door will return to its original position, the open variable allows me to know when it reached that limit.

Picture rotation.

```
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.4f, 1.5f, -13.75f));
model = glm::rotate(model, glm::radians(rot1), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Cuadro.Draw(lightingShader);
glBindVertexArray(0);
```

t is very similar to the animation of the door, we draw, the frame that will rotate certain radians (rot) on the x-axis, it is extremely important in this type of animations to place very well the pivot before exporting the model.

For the animation part, we need a variable rot of type float, which will be the value of rotation of the door, it starts at 0.0f, and the second variable of type bool, will allow me to manage the animation.

```
float rot = 0.0f;
float rot1 = 0.0f;

bool abierto = false;
bool abierto1 = false;
```

```
if (keys[GLFW_KEY_P])
{
    abierto1 = !abierto1;
}
if (abierto1) {
    if (rot1 < 60.0f)
        rot1 += deltaTime * 25;
}
else {
    if (rot1 > 0.0f)
        rot1 -= deltaTime * 25;

}
```

That part is the most important, each time the P key is pressed, the door will rotate up to 30 degrees on the x-axis, in a time deltaTime * 25, that is, it will rotate 25 times faster. When it reaches that degree of rotation, then by pressing the P key again the door will return to its original position, the open variable lets me know when it reached that limit. This animation hides a big surprise.

Account number: 417086611
Group: 12

Complex animations.

Flame.

```
//Cargando el Shader de Anim
Anim.Use();
tiempo = glfwGetTime();
modelLoc = glGetUniformLocation(Anim.Program, "model");
viewLoc = glGetUniformLocation(Anim.Program, "view");
projLoc = glGetUniformLocation(Anim.Program, "projection");
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(7.5f, 0.5f, -9.5f));
model = glm::scale(model, glm::vec3(2.5f, 4.0f, 2.5f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(Anim.Program, "time"), tiempo);
flama.Draw(Anim);
```

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

const float amplitude = 0.7;
const float frequency = 3.0;
const float PI = 3.14159;
out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float time;

void main()
{
  float distance = length(aPos);
  float effect = amplitude*cos(-PI*distance*frequency+time);
  gl_Position = projection*view*model*vec4(aPos.x,aPos.y, aPos.z,1);
  TexCoords=vec2(aTexCoords.x+effect,aTexCoords.y+effect);

}
```

This animation is complex and we do it on the shader itself, giving us an effect on the texture of the model, which simulates that there is a flame. The effect is given by an equation and the animation being on the shader itself is automatic, there is no need to be pressing a key, since it is executed in each clock cycle.

Pendulum.

```
Anim2.Use();
tiempo = glfwGetTime();
modelLoc = glGetUniformLocation(Anim.Program, "model");
viewLoc = glGetUniformLocation(Anim.Program, "view");
projLoc = glGetUniformLocation(Anim.Program, "projection");
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(-4.0f, 0.1f, -0.05f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(Anim.Program, "time"), tiempo);
pendulo.Draw(Anim2);
glBindVertexArray(0);
```

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

const float amplitude = 0.25;
const float frequency = 1.0;
const float PI = 3.14159;
out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float time;

void main()
{
  float distance = length(aPos);
  float effect = amplitude*cos(time);
  gl_Position = projection*view*model*vec4(aPos.x+effect,aPos.y, aPos.z,1); //modelo
  TexCoords=vec2(aTexCoords.x,aTexCoords.y); //Textura

}
```

This animation is complex and we do it on the shader itself, giving us an effect on the vertices of the model, which simulates a movement on the x-axis of the pendulum. The effect is given by an equation and the animation being on the shader itself is automatic, there is no need to be pressing a key, since it is executed in each clock cycle.

Account number: 417086611
Group: 12

Box.

```
Anim3.Use();
tiempo = glfwGetTime();
modelLoc = glGetUniformLocation(Anim.Program, "model");
viewLoc = glGetUniformLocation(Anim.Program, "view");
projLoc = glGetUniformLocation(Anim.Program, "projection");
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::scale(model, glm::vec3(2.5f, 2.0f, 2.5f));
model = glm::rotate(model, glm::radians(0.0f), glm::vec3(0.0f, 1.0, 0.0f));
model = glm::translate(model, glm::vec3(2.50f, 0.1f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(Anim.Program, "time"), tiempo);
cajon.Draw(Anim3);
glBindVertexArray(0);
```

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

const float amplitude = 0.3;
const float frequency = 1.0;
const float PI = 3.14159;
out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float time;

void main()
{
  float distance = length(aPos);
  float effect = amplitude*cos(time);
  gl_Position = projection*view*model*vec4(aPos.x+effect,aPos.y, aPos.z,1); //modelo
  TexCoords=vec2(aTexCoords.x,aTexCoords.y); //Textura

}
```

This animation is complex and we do it on the shader itself, giving us an effect on the vertices of the model, which simulates a movement on the x-axis of the drawer. The effect is given by an equation and the animation being on the shader itself is automatic, there is no need to be pressing a key, since it is executed in each clock cycle. This animation hides a secret.

Sword and bow.

```
Anim4.Use();
tiempo = glfwGetTime();
modelLoc = glGetUniformLocation(Anim.Program, "model");
viewLoc = glGetUniformLocation(Anim.Program, "view");
projLoc = glGetUniformLocation(Anim.Program, "projection");
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
/*model = glm::translate(model, PosIni + glm::vec3(movKitX, movKitY, 0));*/
model = glm::translate(model, glm::vec3(14.0f, 0.2f, -2.5f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(Anim4.Program, "time"), tiempo);
espada3.Draw(Anim4);
model = glm::mat4(1);
tmp = model = glm::translate(model, glm::vec3(0, 1, 0));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
model = glm::translate(model, glm::vec3(-1.5f, 2.5f, -8.530f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
arco.Draw(Anim4);
glBindVertexArray(0);
```

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

const float amplitude = 0.7;
const float frequency = 3.0;
const float PI = 3.14159;
out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float time;

void main()
{
  float distance = length(aPos);
  float effect = amplitude*cos(-PI*distance*frequency+time);
  gl_Position = projection*view*model*vec4(aPos.x,aPos.y, aPos.z,1);
  TexCoords=vec2(aTexCoords.x,aTexCoords.y+effect);

}
```

This animation is complex and we do it on the shader itself, giving us an effect on the texture of the model, which simulates a magic sword as well as the arc. The effect is given by an equation and the animation being on the shader itself is automatic, there is no need to be pressing a key, since it is executed in each clock cycle.

Account number: 417086611
Group: 12

Color lights.

```
if (keys[GLFW_KEY_SPACE])
{
    active = !active;
    if (active)
    {
        Light1 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light2 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light3 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light4 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light5 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light6 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light7 = glm::vec3(1.0f, 0.0f, 0.0f);
        Light8 = glm::vec3(1.0f, 1.0f, 1.0f);

    }
    else
    {
        Light1 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light2 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light3 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light4 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light5 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light6 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light7 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
        Light8 = glm::vec3(0);//Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
    }
}
```

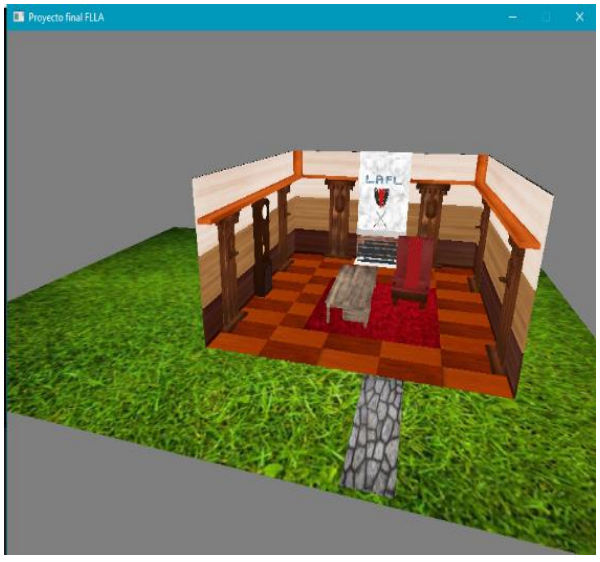
```
// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(-97.5f,7.55f, -38.65f),
    glm::vec3(-97.5f,7.55f, -52.65f),
    glm::vec3(-77.90f,7.55f,  -38.65f),
    glm::vec3(-77.90f,7.55f, -52.65f),
    glm::vec3(-82.15f, 7.55f, -56.9f),
    glm::vec3(-92.85f, 7.55f, -56.9f),
    glm::vec3(-87.45f,3.35f,-56.9f),
    glm::vec3(-87.65f,3.30f,-47.75f),

};
```

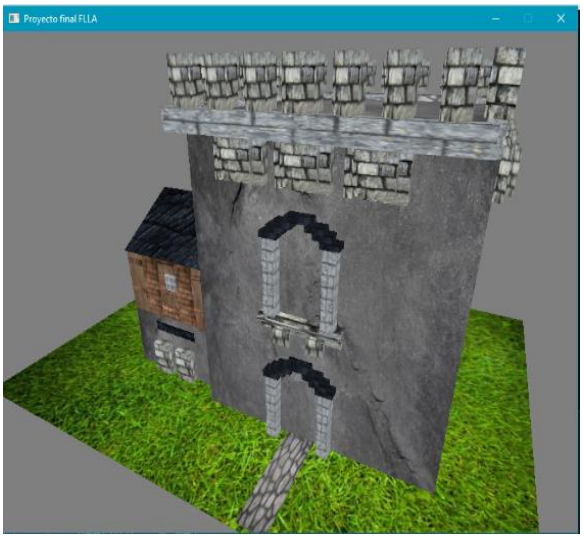
The pointlights are placed in specific points of our scenery, and from the variable lightcolor we oscillate from one color to another when we press the space key the effect is activated, when we press again we stop the effect. It is an extremely colorful effect and gives a good presentation to our scenery.

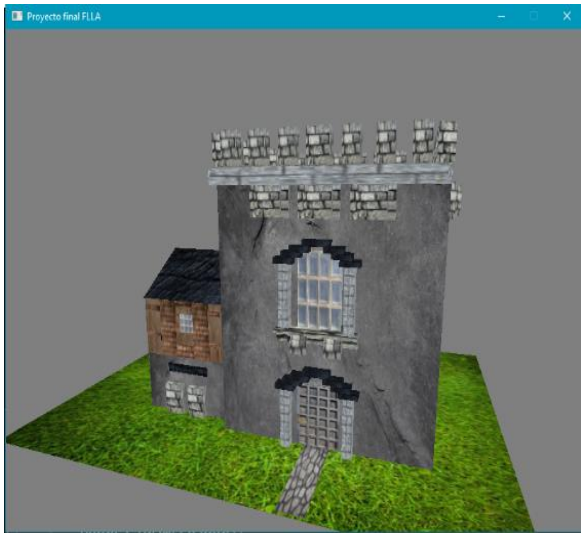
Images.

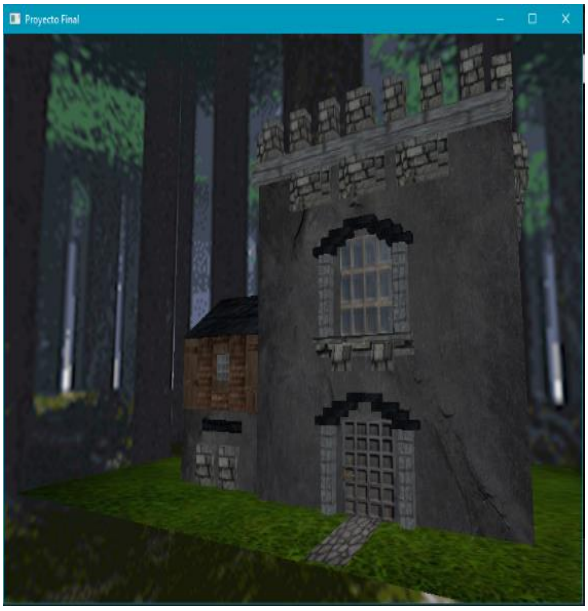
This is the final result! :,) With pictures of the whole process.



Account number: 417086611
Group: 12







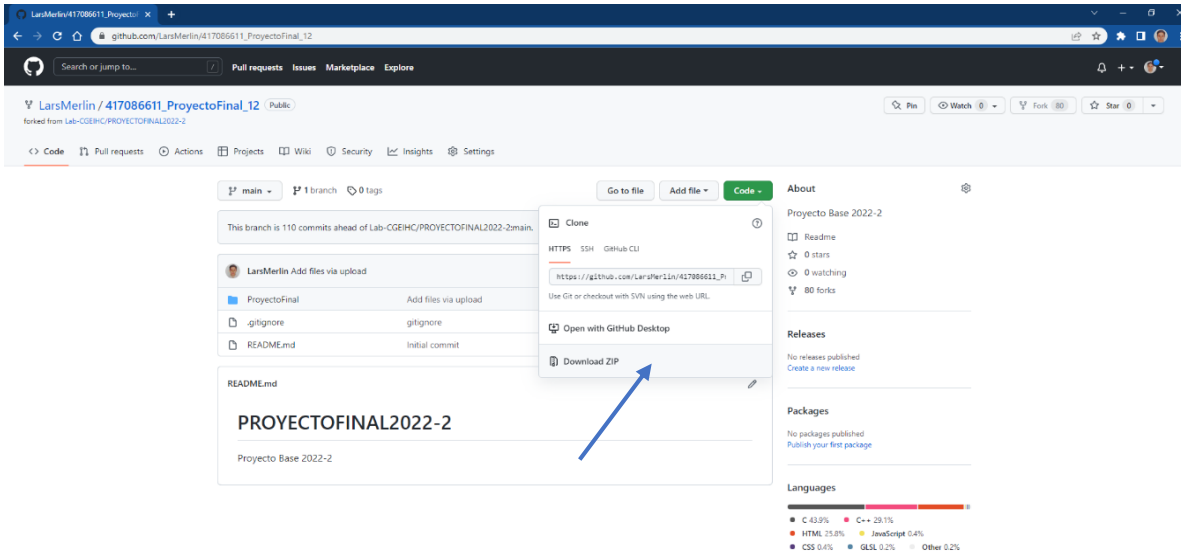
Account number: 417086611
Group: 12

User manual.

This is the user manual for my final project of the Computer Graphics and Human-Computer Interaction Lab. To view and interact with the project, follow the steps below:

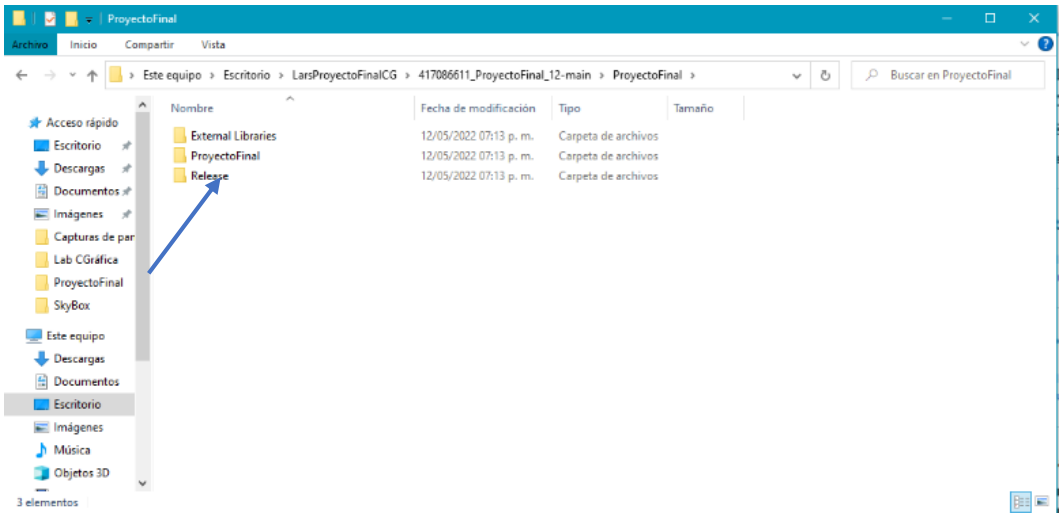
1. Download the project from the GitHub repository.

   https://github.com/LarsMerlin/417086611_ProyectoFinal_12.git



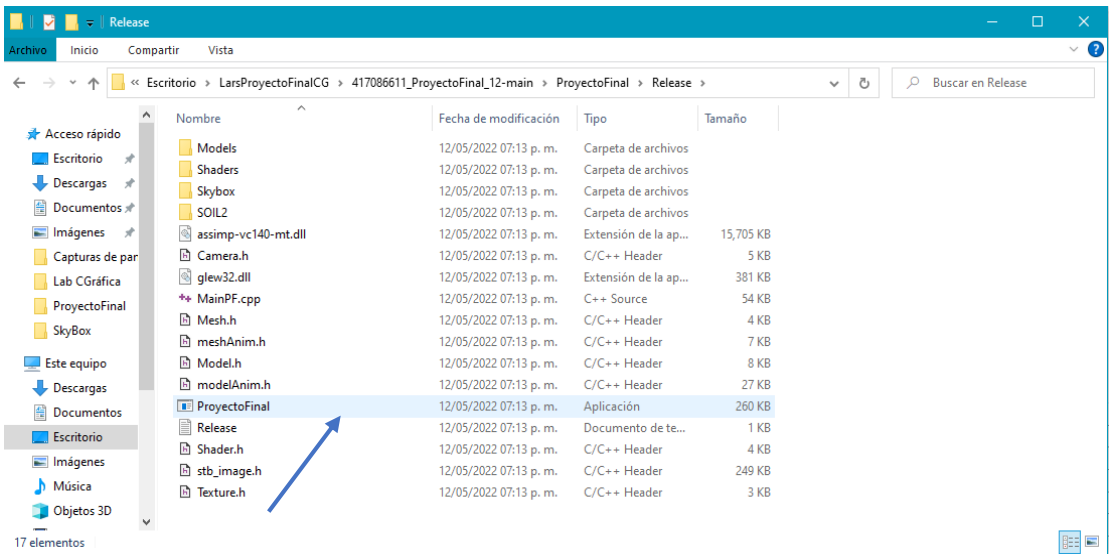
2. Once the project is downloaded, unzip and save it.

3. You will see a folder called Release, click on it.



4. In Release you will see a file called ProyectoFinal.exe, click on it and wait for a moment.

Account number: 417086611
Group: 12



Once inside the project, you can move with the mouse to move the direction of the camera and with the keys W, A, S and D you can move as follows:

- W you move upwards. ↑
- S you move down. ↓
- A you move to the left. ←
- D you move to the right. →
- With the O key you open the door and with the same O key you can close it.
- With the P key you move the frame and with the same P key you can return to its original position.
- With the SPACE key or the space bar you activate the lights and with the same key you deactivate the lights.
- With the ESC key you can close the project.

Conclusion.

This project represented a challenge, time, effort, amazement, in some moments it was tedious, but despite everything I feel happy and at the same time with a bittersweet feeling of wanting to have more time to learn more, to learn more, I really enjoyed the subject and I feel a little sad about the end, but I am grateful to the teacher and my classmates for the fact that I could relive that dream away by certain situations of life; wanting to create video games.

Really, I don't believe it, I see the reference image and that the project resembles a little makes me feel happy, I included things that I liked and were not necessarily in the image but I was confident that they would give a good look and so it was. Each practice was surprising, it was extremely gratifying to be able to solve the exercises, when for the first time we saw modeling or lighting and shading, I felt excited, I thought that this world was very distant, that it was very difficult and after the practices it was not so. I also have to thank the patience and commitment that the teacher has for the subject, without it I would not have been able to resolve many doubts I had throughout the course and during the project.

It is a basic project, it complies with the established, unfortunately for the time and other circumstances it is not as neat as I would like, but it has a great margin of improvement as: in the creation of the materials, that they are no longer of type Lambert and are of type Phong, play more with the lights, include characters and animate them by KeyFrames, include more complex animations that allow to interact to the characters with the scene, include music or sounds, change the keyboard by some control of PlayStation or Xbox, etc.