National Autonomous University of Mexico

Faculty of Engineering.

Final Project – Technical Manual.

Subject: Computing graphics and human-machine interaction.

Professor: Roman Balbuena Carlos Aldair.

Student: Vaquero Barajas Alexis.

Semester: 2022-2.

Delivery date: May 27, 2022.

Final Project.

Content

Camera.

To use a synthetic camera, the library camera.h is imported, so that the main program can use it.

```
25   #include <camera.h>
```

Later, to use the functions of this library, an object of this class is created, so the syntax is as follows:

```
// camera
Camera camera(glm::vec3(0.0f, 50.0f, 200.0f));
```

Within this library are the following constants:

```
18   // Default camera values
19   const float YAW         = -90.0f;
20   const float PITCH       =  0.0f;
21   const float SPEED       =  0.3f;
22   const float SENSITIVITY =  0.7f;
23   const float ZOOM        =  45.0f;
```

These constants handle a default value that can be changed depending on the expected result, for example, modifying SPEED involves the speed in how the camera moves, that is, it performs a translation according to the value of SPEED.

```
void my_input(GLFWwindow *window, int key, int scancode, int action, int mo
{
        if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
                glfwSetWindowShouldClose(window, true);
        if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
                camera.ProcessKeyboard(FORWARD, (float)deltaTime);
        if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
                camera.ProcessKeyboard(BACKWARD, (float)deltaTime);
        if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
                camera.ProcessKeyboard(LEFT, (float)deltaTime);
        if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
                camera.ProcessKeyboard(RIGHT, (float)deltaTime);
```

For the camera to move with the keyboard keys, **my_input** function is declared, which will detect when a key is entered in the program. For it to recognize which

key has been entered, we use an IF control structure and through the GLFW library we compare if that key has been pressed, if this condition is met, the operation assigned to it is performed.

In the case of the camera, using FORWARD, BACK, LEFT, and RIGHT, each key specifies which direction the camera will move, as shown below.

Static Objects.

To use static objects in the project, it is necessary to import the models as **.obj**, since the program is designed to use this type of file.

To import a **.obj** file to the project, it is necessary to enter the objects folder, which is located at the following address: **TEO-ProyectoFinal-CGIHC/TEO-ProyectoFinal-CGIHC/resources**.

Once it is in the folder, a folder with the name of the object must be created, inside it **the textures, the .obj and the .mtl** file will be added, in this way the object has been imported correctly. It should be noted that the **.mtl** file must have the correct textures address, otherwise it will not load them into the program.

Here is an example of importing a static object:

Once the file has been successfully imported, it is necessary to declare such an object. In this case, being a static object, the **Static Shader** is used.

The declared objects are shown below:

```
// Modelos Estaticos
// -----------
Model oxxo("resources/objects/oxxo/oxxo.obj");                          // Oxxo
Model entrada("resources/objects/Entrada/Entrada.obj");         //entrada a la unidad
Model Carro("resources/objects/Bocho/Bocho.obj");                       //Carroceria del bocho
Model llanta("resources/objects/Bocho/Rueda.obj");                      //ruedas
Model plano("resources/objects/Plano/planoVilla.obj");          //plano de la unidad
Model edificiosVilla("resources/objects/EdificioVilla/EdificioVilla.obj");//edificios de la unidad
Model iglesia("resources/objects/Iglesia/iglesia.obj");         //iglesia
Model arbusto("resources/objects/arbusto/arbusto.obj");         //arbusto
Model pedales("resources/objects/bicicleta/pedales.obj");       //Bici
Model cuadro("resources/objects/bicicleta/cuadro.obj");
Model rueda("resources/objects/bicicleta/rueda.obj");
Model courtBasket("resources/objects/CanchaBasquet/cancha.obj"); //Cancha Basquetbol
Model cuarto1("resources/objects/Cuarto_1/cuarto1.obj"); //Cuarto 1
Model cuarto2("resources/objects/Cuarto_2/cuarto2.obj"); //Cuarto 2
```

Finally, it only remains to call these objects and perform their corresponding transformations, as well as the **Static Shader** so that it can be displayed in the program.

An example is the following:

```
// ------------------------------------------------------------------------------------------------------------
// Escenario (StaticShader)
// ------------------------------------------------------------------------------------------------------------
staticShader.use();
staticShader.setMat4("projection", projection);
staticShader.setMat4("view", view);

model = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, -70.0f));        //proporciones de todo con un plano
model = glm::scale(model, glm::vec3(12.0f));
staticShader.setMat4("model", model);
plano.Draw(staticShader);
```

Dynamic objects.

For these objects, they are imported in the same way as static objects, only now they **use .dae files** and their respective textures, since an **Animate Shader** is used.

To declare a dynamic object is done as follows:

```
// Dog
ModelAnim dog("resources/objects/Dog/doggo.dae");
dog.initShaders(animShader.ID);
```

Subsequently, the object is only called, and its corresponding transformations are carried out in the same way, in addition to using the **Animate Shader**.

```
// -----------------------------------------------------------------------------
// Animaciones (AnimShader)
// -----------------------------------------------------------------------------
//Remember to activate the shader with the animation
animShader.use();
animShader.setMat4("projection", projection);
animShader.setMat4("view", view);

animShader.setVec3("material.specular", glm::vec3(0.5f));
animShader.setFloat("material.shininess", 32.0f);
animShader.setVec3("light.ambient", ambientColor);
animShader.setVec3("light.diffuse", diffuseColor);
animShader.setVec3("light.specular", 0.0f, 0.0f, 0.0f);
animShader.setVec3("light.direction", lightDirection);
animShader.setVec3("viewPos", camera.Position);

//-------------------Bicicleta--------------------------------
model = glm::translate(glm::mat4(1.0f), glm::vec3(movBici_x, 0.0f, movBici_z));
model  = glm::rotate(model, glm::radians(orientaBici), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09));
animShader.setMat4("model", model);
manBici.Draw(animShader);
```

For this object to perform an animation in the environment, it is necessary to make changes to the **Animate function**, such as the ones shown below.

Animation information.

Animation of man on bicycle:

To generate the animation three global variables were created:

```
float movBici_z = -5.0f, // posición inicial en z
      movBici_x = -500.0f,  // posición inicial en x
      orientaBici = 0.0f;   // orientación incial
```

The animation is divided into several objects:

- Man pedaling
- Bicycle frame
- Pedals
- Wheels

The pedaling man animation was created in 3ds Max using the Auto Key tool. When drawing the man in the program, we assign the bike's x and z motion and rotation variables so that when the bike moves, the man moves too.

```
//-------------------Bicicleta--------------------------------
model = glm::translate(glm::mat4(1.0f), glm::vec3(movBici_x, 0.0f, movBici_z));
model  = glm::rotate(model, glm::radians(orientaBici), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09));
animShader.setMat4("model", model);
manBici.Draw(animShader);
```

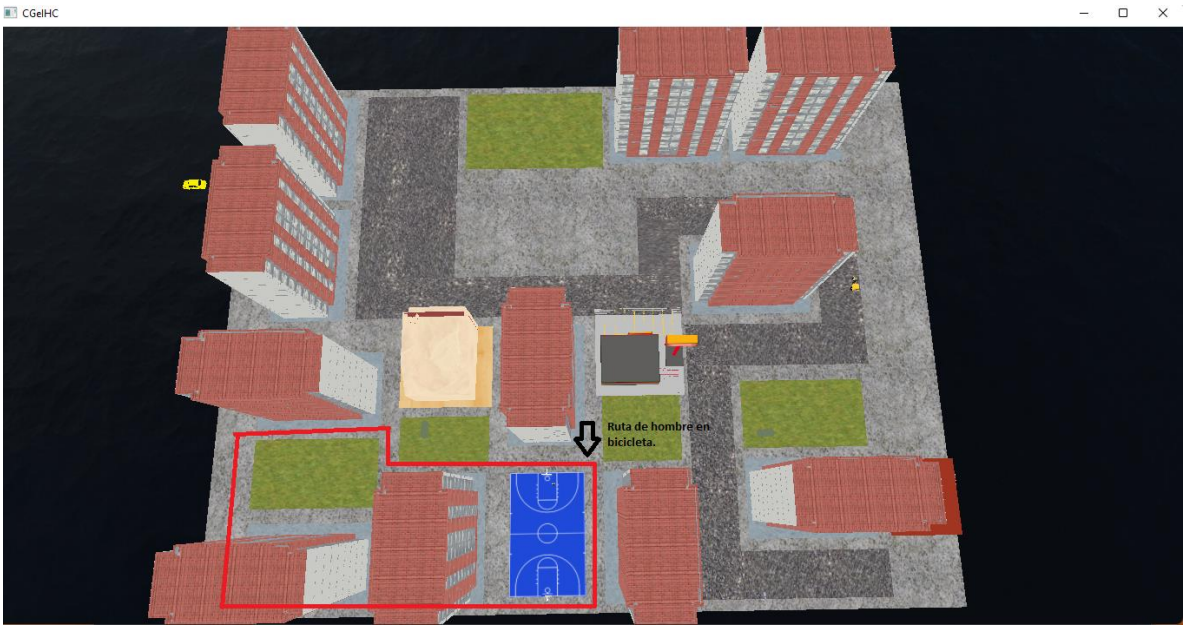
We draw the bike with its initial values and create a temporary model. For the animation of the pedals and the wheels we add a rotation of the x axis:

```
// ------------------------------------------------------------------------------------------------------------------------
// Persona en bici
// ------------------------------------------------------------------------------------------------------------------------
tmp = model = glm::translate(glm::mat4(1.0f), glm::vec3(movBici_x, -0.5f, movBici_z));
tmp = model = glm::rotate(model, glm::radians(orientaBici), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));
staticShader.setMat4("model", model);
cuadro.Draw(staticShader);

model = glm::translate(tmp, glm::vec3(0.0f, 2.75f, 0.5f));
model = model = glm::rotate(model, glm::radians(giroPedales), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));
staticShader.setMat4("model", model);
pedales.Draw(staticShader);

model = glm::translate(tmp, glm::vec3(-0.2f, 2.85f, 5.85f));
model = model = glm::rotate(model, glm::radians(giroPedales), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));
staticShader.setMat4("model", model);
rueda.Draw(staticShader); // Adelante

model = glm::translate(tmp, glm::vec3(-0.2f, 2.85f, -3.25f));
model = model = glm::rotate(model, glm::radians(giroPedales), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));
staticShader.setMat4("model", model);
rueda.Draw(staticShader); // Atras
```

For the animation of the bicycle, a route was assigned within the map where 6 states from 0 to 5 will be used using the Switch/Case control structure, the bicycle will change position on the x axis and the z axis.

This animation is repeated infinitely since when reaching state 5 the bicycle is at the origin coordinates and returns to state 0, it is not necessary to press any key to start it, the animation starts along with the program.

```
switch (estado_bici)
{
case 0:
    if (movBici_z <= 255.0f) {
        movBici_z += 2.0f;
        orientaBici = 0.0;
    }
    else {
        estado_bici = 1;
    }
    break;
case 1:

    if (movBici_x <= 55.0f) {
        orientaBici = 90.0f;
        movBici_x += 2.0;
    }
    else {
        estado_bici = 2;
    }
    break;
case 2:
    if (movBici_z >= 50.0f)
    {
        orientaBici = 180.0f;
        movBici_z -= 2.0f;
    }
    else {
        estado_bici = 3;
    }
    break;
case 3:
    if (movBici_x >= -250.0f) {
        orientaBici = 270.0f;
        movBici_x -= 2.0f;
    }
    else {
        estado_bici = 4;
    }
    break;
case 4:
    if (movBici_z >= -5.0f) {
        orientaBici = 180.0f;
        movBici_z -= 2.0f;
    }
    else {
        estado_bici = 5;
    }
    break;
case 5:
    if (movBici_x >= -500.0f) {
        orientaBici = 270.0f;
        movBici_x -= 2.0f;
    }
    else
    {
        estado_bici = 0;
    }
    break;
default:
    break;
}
```



Animation of man doing sports on the Basketball court:

To generate the animation four global variables were created:

```
//para mov de deportista
float movShan = 0.0f, //con esto se movera el deportista
    IdaRegresoShan = 1.0f; //para que avance y regrese
int estadoShan = 1.0f;
float orienShan = 0.0f; // para que gire a diferentes posiciones
```

The animation is composed of a single object, the character Shannon with the Running animation, downloaded from the Mixamo page. At the time of download it was indicated that the animation should be in one place, so that the scrolling would be generated based on the code.

To draw the object, the variable that will allow it to move on the Z axis had to be placed, taking advantage of the fact that it will not move on the X and Y axes; in conjunction with a variable that allows its rotation about the Y axis to simulate its change of direction.

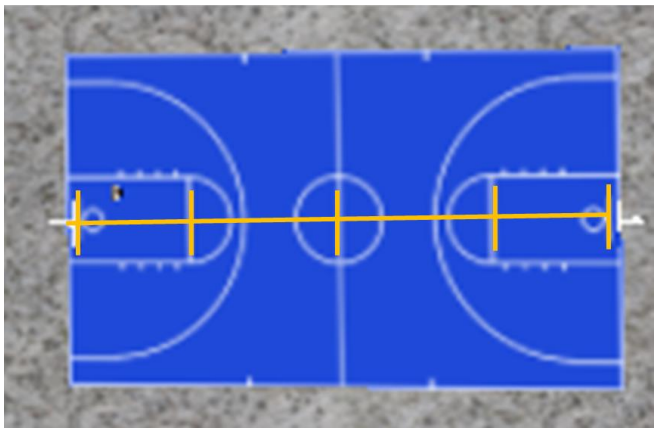
```
//Deportista
model = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 70.0f + movShan));
model = glm::rotate(model, glm::radians(orienShan), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));//escala
animShader.setMat4("model", model);
shannon.Draw(animShader);
```

By occupying a SWITCH, one chooses between the different states, which represent how far the object must move away from the field. Inside each CASE is an IF statement that checks if the object is moving forward (towards higher values) or backwards (toward smaller values). minors). If the condition that it reaches a certain distance is met, the object will make a 180° turn and make the next move by changing state.

```
switch (estadoShan){//deportista: va a hacer Suicidios
case 1://lega a 20
    if (IdaRegresoShan == 1)
    {
        movShan += 0.5f;//con esto el deportista sale hacia adelante
        if (movShan >= 20) {
            IdaRegresoShan = 0;//cambio de estado
            orienShan = 180.0f;//giro para el regreso
        }
    }
    else {
        movShan -= 0.5f;//con esto el deportista sale hacia atras
        if (movShan <= 0) {
            orienShan = 0.0f;//giro hacia enfrente
            IdaRegresoShan = 1;//cambio de estado
            estadoShan = 2;//cambio de estado del case
        }
    }
    break;
case 2://llega a 80
    if (IdaRegresoShan == 1 && estadoShan == 2)
    {
        movShan += 0.5f;//con esto el deportista sale hacia adelante
        if (movShan >= 80) {
            IdaRegresoShan = 0;//cambio de estado
            orienShan = 180.0f;//giro para el regreso
        }
    }
    else {
        movShan -= 0.5f;//con esto el deportista sale hacia atras
        if (movShan <= 0) {
            orienShan = 0.0f;//giro hacia enfrente
            IdaRegresoShan = 1;//cambio de estado
            estadoShan = 3;//cambio de estado del case
        }
    }
```

```
case 3://llega a 120
    if (IdaRegresoShan == 1)
    {
        movShan += 0.5f;//con esto el deportista sale hacia adelante
        if (movShan >= 120) {
            IdaRegresoShan = 0;//cambio de estado
            orienShan = 180.0f;//giro para el regreso
        }
    }
    else {
        movShan -= 0.5f;//con esto el deportista sale hacia atras
        if (movShan <= 0) {
            orienShan = 0.0f;//giro hacia enfrente
            IdaRegresoShan = 1;//cambio de estado
            estadoShan = 4;//cambio de estado del case
        }
    }
    break;
case 4://llega a 160
    if (IdaRegresoShan == 1)
    {
        movShan += 0.5f;//con esto el deportista sale hacia adelante
        if (movShan >= 160) {
            IdaRegresoShan = 0;//cambio de estado
            orienShan = 180.0f;//giro para el regreso
        }
    }
    else {
        movShan -= 0.5f;//con esto el deportista sale hacia atras
        if (movShan <= 0) {
            orienShan = 0.0f;//giro hacia enfrente
            IdaRegresoShan = 1;//cambio de estado
            estadoShan = 1;//cambio de estado del case (regreso)
        }
    }
```

For the animation to result in the character running to each mark and back to the origin repeatedly, as the animation was not conditional on user input from a key press.



Car Animation:

To generate the animation four global variables were created:

```
//para el movimiento del automovil
float   movAuto_x = 0.0f,
        movAuto_z = 0.0f,

        orienta = 180.0f,
        girollantas = 0.0f,
        estadoAuto = 1.0f;
```

The animation is made up of five objects, the body and four wheels, I think the most complicated part of placing the objects was the proportions and placing the wheels in the correct position so that they do not look bigger than they should.

The variables that allow its movement in X and Z and one to make the turn in Y were placed in the bodywork; for the wheels only one variable is used for their rotation around X.

```
// ------------------------------------------------------------------------------------------------------
// Carro
// ------------------------------------------------------------------------------------------------------
model = glm::rotate(glm::mat4(1.0f), glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(-225.0f + movAuto_x, 1.5f, 500.0f + movAuto_z));
tmp = model = glm::rotate(model, glm::radians(orienta), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.005f, 0.005f, 0.005f));
staticShader.setMat4("model", model);
Carro.Draw(staticShader);

model = glm::translate(tmp, glm::vec3(6.7f, 1.0f, 15.0f));
model = glm::rotate(model, glm::radians(girollantas), glm::vec3(1.0f, 0.0f, 0.0f));//giro de las llantas
model = glm::scale(model, glm::vec3(0.0044f, 0.0044f, 0.0044f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
staticShader.setMat4("model", model);
llanta.Draw(staticShader);  //Izq delantera

model = glm::translate(tmp, glm::vec3(-6.7f, 1.0f, 15.0f));
model = glm::rotate(model, glm::radians(girollantas), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.0044f, 0.0044f, 0.0044f));
staticShader.setMat4("model", model);
llanta.Draw(staticShader);  //Der delantera

model = glm::translate(tmp, glm::vec3(-6.7f, 1.0f, -10.0f));
model = glm::rotate(model, glm::radians(girollantas), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.0044f, 0.0044f, 0.0044f));
staticShader.setMat4("model", model);
llanta.Draw(staticShader);  //Der trasera

model = glm::translate(tmp, glm::vec3(6.7f, 1.0f, -10.0f));
model = glm::rotate(model, glm::radians(girollantas), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.0044f, 0.0044f, 0.0044f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
staticShader.setMat4("model", model);
llanta.Draw(staticShader);  //Izq trase
```

For the car to go through its motions, the user must press the SPACE key to go forward or stop. For the path, it was decided to start next to the input object and finish stationary.

The first thing to do is check the state of the car so that it moves forward, and the wheels turn in one direction, then it asks if it has already reached a position that we select so that it turns, changes state, and continues moving. Forward, the positions depend on the distribution of the asphalt. I think the hardest thing about the animation was seeing how much it must rotate and not go through objects we don't want.

```
//Vehículo (se estaciona)
if (animacion)
{
    if (estadoAuto == 1) {
        movAuto_z -= 3.0f;
        girollantas += 3.0f;
        if (movAuto_z <= -260) {
            orienta = 90.0f;
            estadoAuto = 2.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 2) {
        movAuto_x += 3.0f;
        girollantas += 3.0f;
        if (movAuto_x >= 360) {
            orienta = 0.0f;
            estadoAuto = 3.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 3) {
        movAuto_z += 3.0f;
        girollantas += 3.0f;
        if (movAuto_z >= -30) {
            orienta = 90.0f;
            estadoAuto = 4.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 4) {
        movAuto_x += 3.0f;
        girollantas += 3.0f;
        if (movAuto_x >= 580) {
            orienta = 180.0f;
            estadoAuto = 5.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 5) {
        movAuto_z -= 3.0f;
        girollantas += 3.0f;
        if (movAuto_z <= -330) {
            orienta = 270.0f;
            estadoAuto = 6.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 6) {
        movAuto_x -= 3.0f;
        girollantas += 3.0f;
        if (movAuto_x <= 450) {
            orienta = 180.0f;
            estadoAuto = 7.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 7) {
        movAuto_z -= 3.0f;
        girollantas += 3.0f;
        if (movAuto_z <= -750) {
            orienta = 90.0f;
            estadoAuto = 8.0f;
            //animacion = FALSE;
        }
    }
    if (estadoAuto == 8) {
        movAuto_x += 3.0f;
        girollantas += 3.0f;
        if (movAuto_x >= 650) {
            orienta = 180.0f;
            estadoAuto = 9.0f;
            //animacion = FALSE;//no debe pararse
        }
    }
    if (estadoAuto == 9) {
        movAuto_z -= 3.0f;
        girollantas += 3.0f;
        if (movAuto_z <= -800) {
            animacion = FALSE;
        }
    }
}
```

Man Walking Animation:

To generate the animation four global variables were created:

```
//para el movimiento del constructor
float movConstX = 0.0f,
    movConstZ = 0.0f,
    OrientaConst = 180.0f,
    estadoConst = 1.0f;
```

For this animation, only an object downloaded from Mixamo was used. The character was placed under the entrance and with two variables for its movement in X and Z it will move until it reaches the OXXO and with a variable in its rotation it will rotate on the Y axis.

```
// Persona Caminando
model = glm::translate(glm::mat4(1.0f), glm::vec3(580.0f + movConstX, 0.0f, 120.0f + movConstZ));
model = glm::scale(model, glm::vec3(0.09f));
model = glm::rotate(model, glm::radians(OrientaConst), glm::vec3(0.0f, 1.0f, 0.0f));
animShader.setMat4("model", model);
manWalk.Draw(animShader);
```

Within the animate function, an IF was placed so that the user presses the 1 key and thus the character begins its journey. Subsequently, the state in which it is found is verified so that it advances in the correct direction and when it reaches a certain place, it changes orientation and state.

It should be noted that in state five, the character will not advance if the car is going to cross the same place, proving that the state of the car is different from state two; in turn, the character only goes through the OXXO object, the doors do not open for him to enter.

```
if (animacionConst) {//presionar 1 para que el ocnstructor camine
    if (estadoConst == 1) {
        movConstX -= 0.6f;
        if (movConstX <= -90) {
            OrientaConst = 90.0f;
            estadoConst = 2.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 2) {
        movConstZ -= 0.6f;
        if (movConstZ <= -85) {
            OrientaConst = 180.0f;
            estadoConst = 3.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 3) {
        movConstX -= 0.6f;
        if (movConstX <= -290) {
            OrientaConst = 90.0f;
            estadoConst = 4.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 4) {
        movConstZ -= 0.6f;
        if (movConstZ <= -200) {
            OrientaConst = 180.0f;
            estadoConst = 5.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 5 && estadoAuto != 2) {//solo cruza si es que el carro no esta
        movConstX -= 0.6f;
        if (movConstX <= -370) {
            OrientaConst = 90.0f;
            estadoConst = 6.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 6) {
        movConstZ -= 0.6f;
        if (movConstZ <= -275) {
            OrientaConst = 180.0f;
            estadoConst = 7.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 7) {
        movConstX -= 0.6f;
        if (movConstX <= -440) {
            OrientaConst = -90.0f;
            estadoConst = 8.0f;
            //animacionConst = FALSE;//no debe pararse
        }
    }
    if (estadoConst == 8) {
        movConstZ += 0.6f;
        if (movConstZ >= -260) {
            animacionConst = FALSE;
        }
    }
}
```

Person walking the dog Animation.

To carry out the corresponding animation, a constant was first defined which will indicate the speed at which the characters move as well as 5 floating variables where these will help us move the objects on the X, Z axis and finally rotate these same ones.

```
#define SPEED_MOV 0.2f
float   rotDogPerson = 180.0f,
        movPersonX = 0.0f,
        movPersonZ = 0.0f,
        movDogX = 0.0f,
        movDogZ = 0.0f;
```

Another variable that is also declared is the state of the animation, which will be the one that will define the change of state through a switch-case.

```
int estado_trici = 0,
    estado_dogPerson = 0,
    estado_bici = 0;
```

For the objects to perform the animation, it is important that where we draw them, we add the corresponding variables in the translation (X, Z), where these will be added to their defined position, so that the initial position will not be modified. In the same way, so that the objects can perform a rotation, the corresponding variable is added in this.

```
// DOG
model = glm::translate(glm::mat4(1.0f), glm::vec3(290.0f + movDogX, 0.0f, 0.0f + movDogZ));
model = glm::scale(model, glm::vec3(0.3f));
model = glm::rotate(model, glm::radians(rotDogPerson), glm::vec3(0.0f, 1.0f, 0.0f));
animShader.setMat4("model", model);
dog.Draw(animShader);

// Persona Paseando
model = glm::translate(glm::mat4(1.0f), glm::vec3(285.0 + movPersonX, 0.0f, 0.0f + movPersonZ));
model = glm::scale(model, glm::vec3(0.09f));
model = glm::rotate(model, glm::radians(rotDogPerson), glm::vec3(0.0f, 1.0f, 0.0f));
animShader.setMat4("model", model);
womanWalk.Draw(animShader);
```

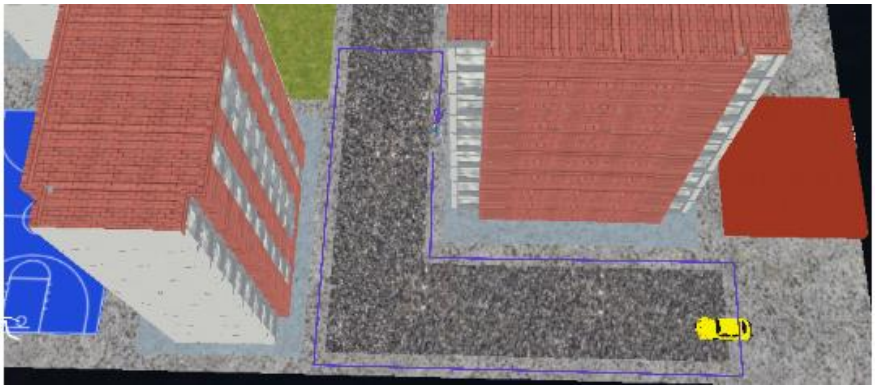
To carry out the states of the animation, the Switch-Case control structure is used, so that when the expected result is met, another case is passed, and the animation continues.

For each case there is an if-else that will check that the movement on a certain axis does not exceed that determined by the developer, so the objects will increase or

decrease their position up to this limit in addition to rotating the established degrees.

Finally, once the condition is not met, the dog will increase or decrease its position so that it takes it to the person's left side and moves on to the next case (animation state with a maximum of 5 states).

```
switch (estado_dogPerson) {
        case 0:
            if (movPersonZ <= 170.0f) {
                rotDogPerson = 0.0f;
                movDogZ += SPEED_MOV;
                movPersonZ += SPEED_MOV;
            } else {
                movDogZ = movPersonZ - 5.0f;
                estado_dogPerson = 1;
            }
            break;
        case 1:
            if (movPersonX <= 220.0f) {
                rotDogPerson = 90.0f;
                movDogX += SPEED_MOV;
                movPersonX += SPEED_MOV;
            } else {
                movDogX = movPersonX + 5.0f;
                estado_dogPerson = 2;
            }
            break;
        case 2:
            if (movPersonZ <= 250.0f) {
                rotDogPerson = 0.0f;
                movDogZ += SPEED_MOV;
                movPersonZ += SPEED_MOV;
            }
            else {
                movDogZ = movPersonZ + 10.0f;
                estado_dogPerson = 3;
            }
            break;
        case 3:
            if (movPersonX >= -80.0f) {
                rotDogPerson = 270.0f;
                movDogX -= SPEED_MOV;
                movPersonX -= SPEED_MOV;
            }
            else {
                movDogX = movPersonX - 10.0f;
                estado_dogPerson = 4;
            }
            break;
        case 4:
            if (movPersonZ >= 0.0f) {
                rotDogPerson = 180.0f;
                movDogZ -= SPEED_MOV;
                movPersonZ -= SPEED_MOV;
            }
            else {
                movDogZ = movPersonZ - 5.0f;
                estado_dogPerson = 5;
            }
            break;

        case 5:
            if (movPersonX <= 5.0f) {
                rotDogPerson = 90.0f;
                movDogX += SPEED_MOV;
                movPersonX += SPEED_MOV;
            }
            else {
                movDogZ = movPersonX + 5.0f;
                estado_dogPerson = 0;
                movPersonZ = movDogZ = 0.0f;
                movPersonX = movDogX = 0.0f;
            }
            break;
    }
```



References.

- Edificios de México (s.f.). Villa Olympic Miguel Hidalgo [mensaje en un blog]. Recuperate from https://www.edemx.com/site/villa-olimpica-miguel-hidalgo/

- MIXAMO. (2022). Recuperate from https://www.mixamo.com/#/

- Open3DModel. (2022). Recuperate from https://open3dmodel.com/es/