Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Semestre 2022 – 2

Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Proyecto Final

Technical documentation

Nombre: Córdoba Gómez Rodrigo

Núm. Cuenta: 3-1611829-9

Profesor: Ing. Carlos Aldair Román Balbuena

Grupo teoría: 2

Grupo laboratorio: 9

Fecha de entrega: 11 de mayo de 2022

Contents

Variable dictionary:

Name	Type	Description	Scope
DoorRot	Float	Defines the angle for Door rotation	Global
recRot	Float	Defines the angle for reception door rotation	Global
chairZ	Float	Defines the chair translation in the Z axis	Global
refRot	Float	Defines the rotation of the soda cup	Global
ballRotX	Float	Defines the bowling ball rotation in the X axis, in degrees	Global
ballRotY	Float	Defines the bowling ball rotation in the Y axis, in degrees	Global
ballRotZ	Float	Defines the bowling ball rotation in the Z axis, in degrees	Global
amX	Float	Defines the increment between time deltas for processor calls for ballRotX	Global
amY	Float	Defines the increment between time deltas for processor calls for ballRotY	Global
amZ	Float	Defines the increment between time deltas for processor calls for ballRotZ	Global
isDoorOpen	Boolean	Controls the door movement direction	Global
doorMoving	Boolean	Defines if the door should be animated	Global
isRecOpen	Boolean	Controls the reception door movement direction	Global

recMoving	Boolean	Defines if the reception door should be animated	Global
isChairOpen	Boolean	Controls the chair position	Global
chairMoving	Float	Defines if the chair should be animated	Global
drawLightCubes	Boolean	Controls drawing the lighting cubes for point lights	Global
ballMoving	Boolean	Defines if the bowling ball shall rotate	Global
refDir	Boolean	Controls the soda cup rotation	Global
servPos	Vector3	Stores the flying napkin position	Global
servRot	Float	Stores the current napkin rotation in Y axis	Global
servRot2	Float	Stores the current napkin rotation in Z axis	Global
ballPosIni	Vector3	Initial ball position for keyframe animation	Global
escobPosIni	Vector3	Initial sweeper position for keyframe animation	Global
pinPosIni	Vector3	Initial pin position for keyframe animation	Global
pinOffset	Vector3	Offset for bowling pin placement	Global
pinRot	Float	Keyframe bowling X rotation	Global
LightP1	Vector3	Controls the color of the ambient and diffuse components of the 4^{th} point light	Global

Objectives

To virtually recreate, using 3D modeling software and OpenGL in the C++ language, a building facade, as well as the interior environment, using the knowledge acquired during the Laboratorio de Computación Gráfica e Interacción Humano-Computadora course. This virtual environment must contain 5 animations, of which 3 must be simple and two others must be complex. Recreate 7 objects that can be visualized within the environment.

Project scope:

The average user, without any computer background, should be able to navigate through this virtual environment and interact with the objects within it using a Microsoft Windows computer, and the redistributable Visual C++ 2019 files. A relatively modern graphics processor capable of running OpenGL, 4 GB of RAM and a 1.0 GHz or better processor should be available. The user must interact and navigate through this virtual environment using a monitor, keyboard and mouse.

The project must be provided in its entirety and for its execution inside a compressed file in ZIP format, which contains the executable file and all the necessary dependencies for the user to run it without problems in his computer. Likewise, a user manual and this document must be provided for consultation.

Limitations

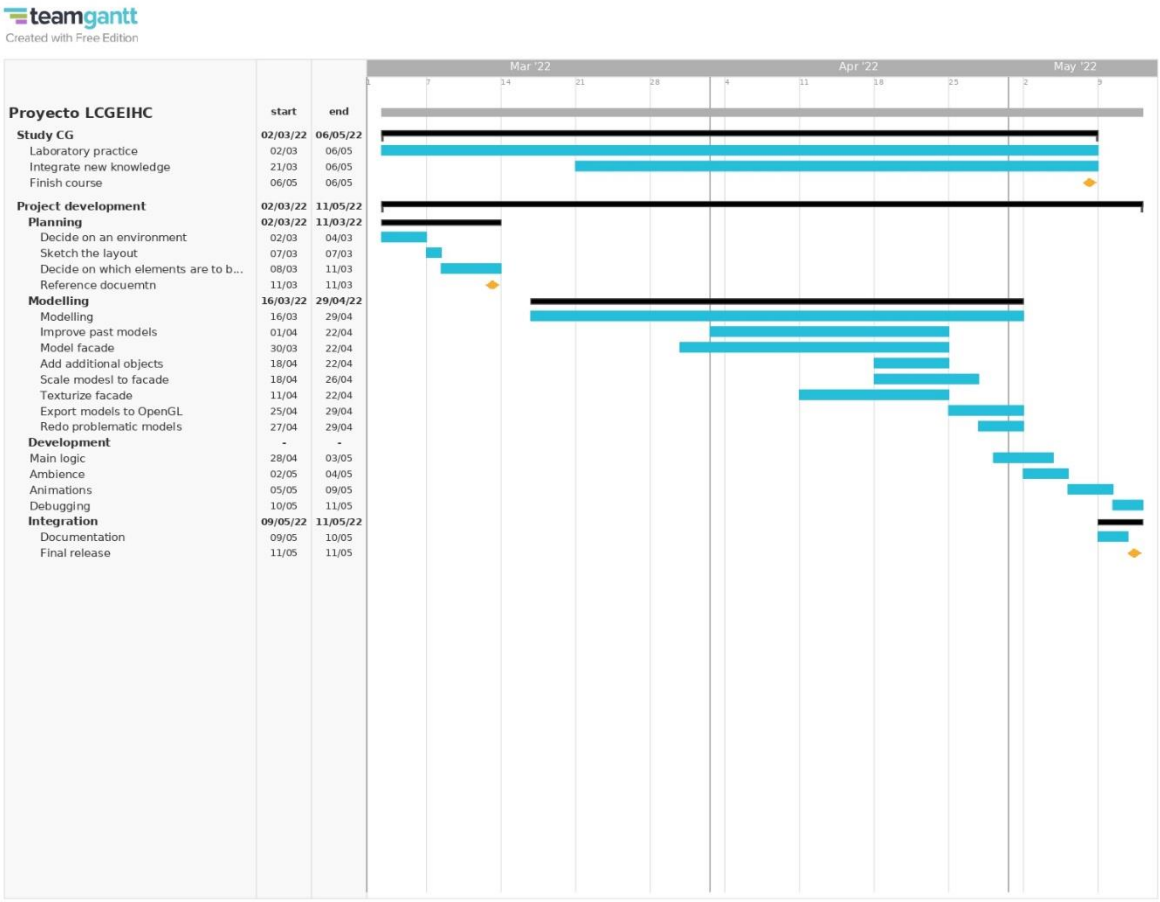
Because the libraries used are based on a 32-bit Microsoft Windows operating system, it is not possible to run this program on MacOS, Linux, or any other operating system without additional software. Compatibility with other operating systems is not supported.

The above specifications may not be sufficient, depending on the user's machine. Due to the lack of hardware, these limitations are not contemplated in the development of the project.

As a virtual recreation of an environment, this project is not completely faithful to the environment, and although it plans to recreate as close to reality as possible, it is not possible to achieve a 1:1 model with all the details that are present in the environment.

Likewise, since it is not a video game, it is not expected that the user interaction will be more than those defined in the user's manual and in the documentation. The recreated interactions are only those found in the original source code, and user actions are not able to change the result of these animations, this is considered out of the scope of the project.

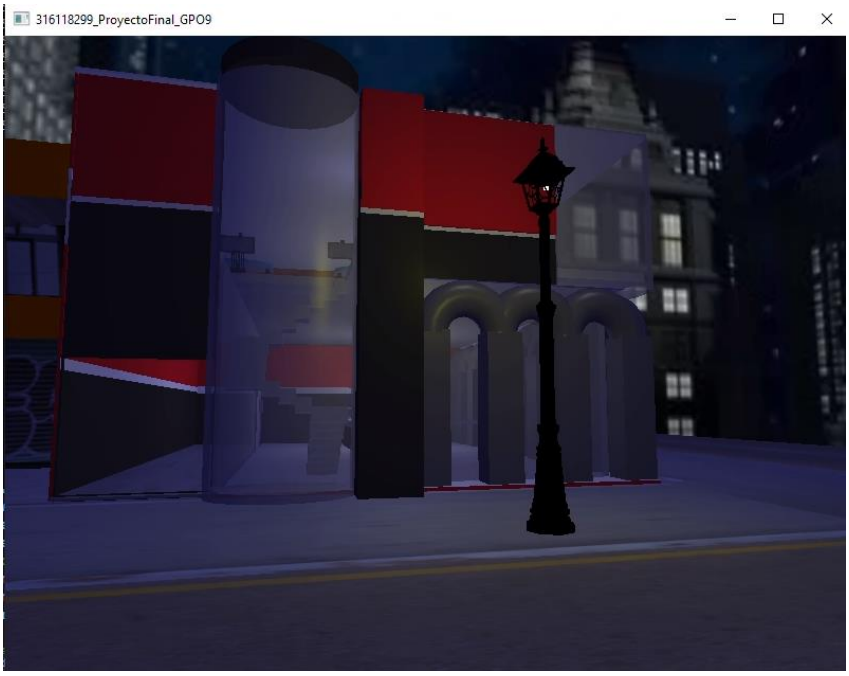
Gantt Diagram



Ambientation

Building exterior

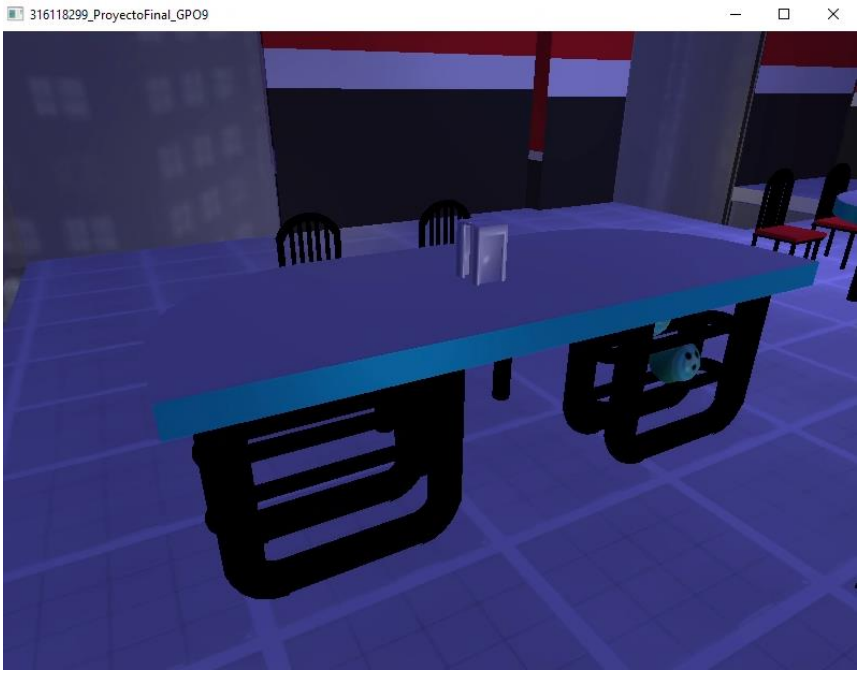
The chosen building was the Bol Montevideo. To increase the realism of the environment, the facade of the adjacent building was modeled, and a skybox was placed to represent the environment in which the building is located. We chose to set the project at night to better showcase the lighting.



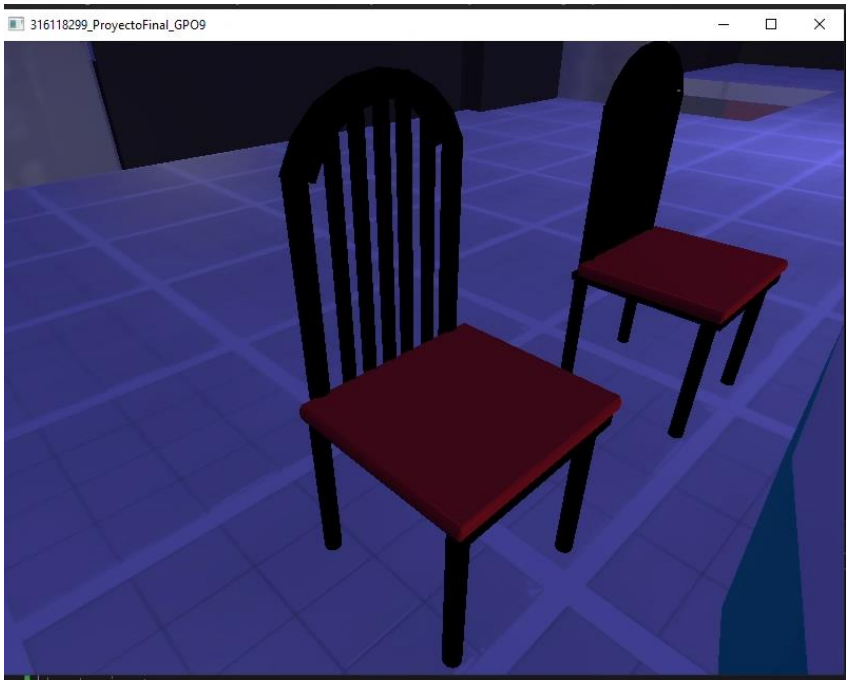
Ambient objects:

As stipulated in the project requirements, 7 objects from the environment must be recreated as 3D models that can be imported into OpenGL.

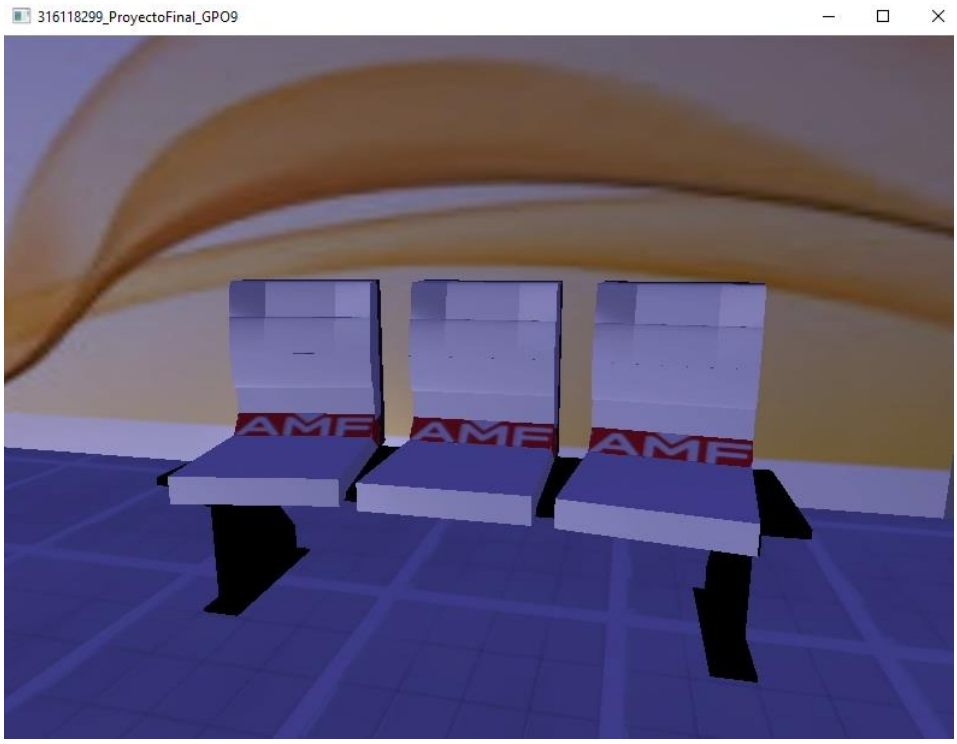
Table



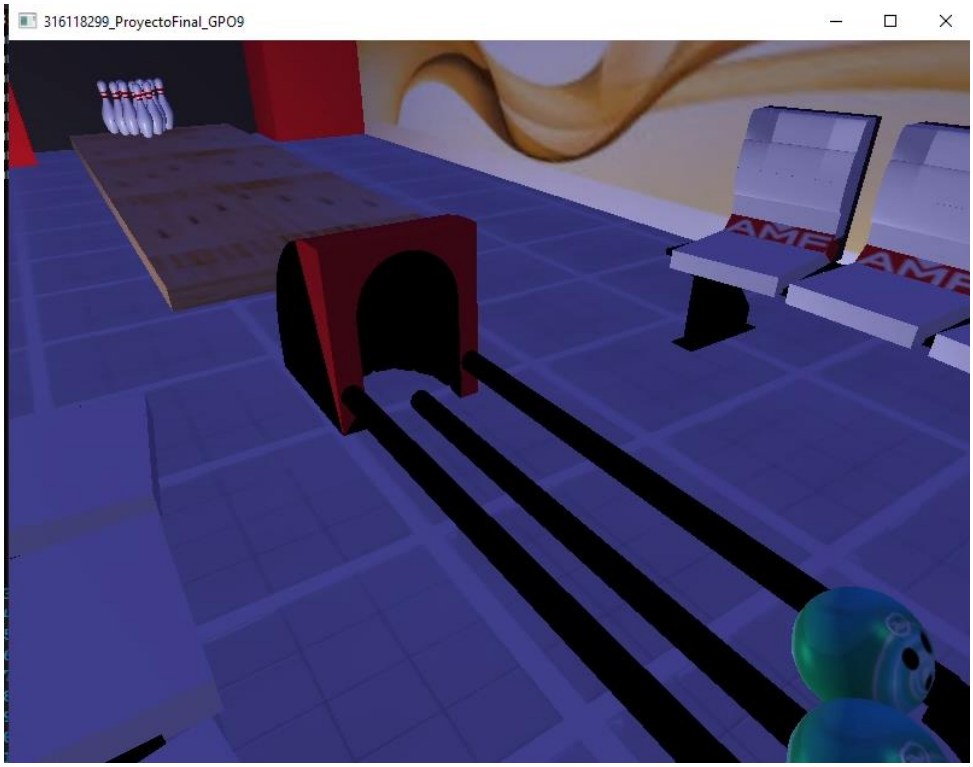
Chairs



Triple chairs



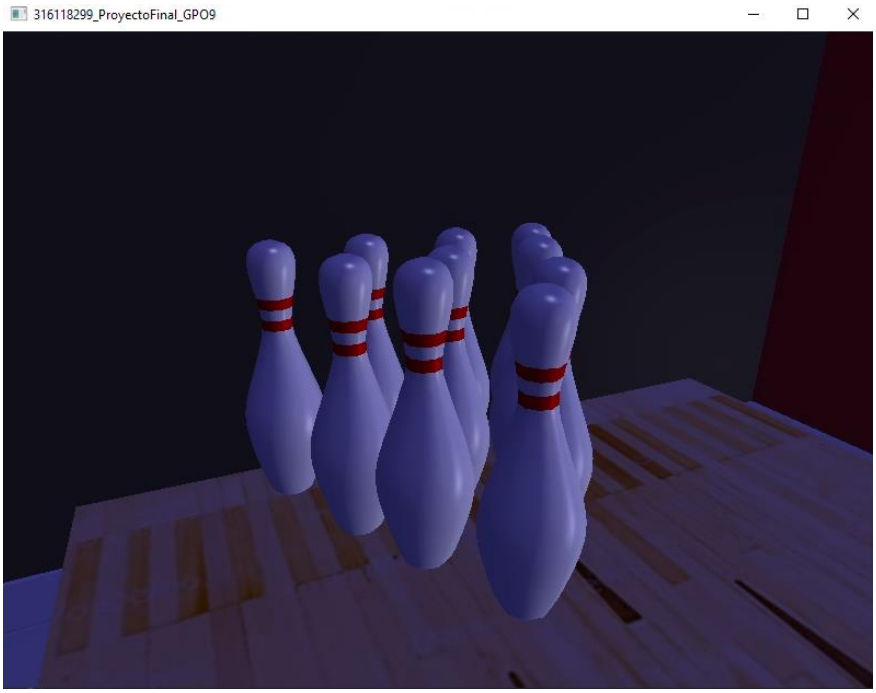
Ball Collector



Bowling Ball

Scoreboard

Bowling Pin

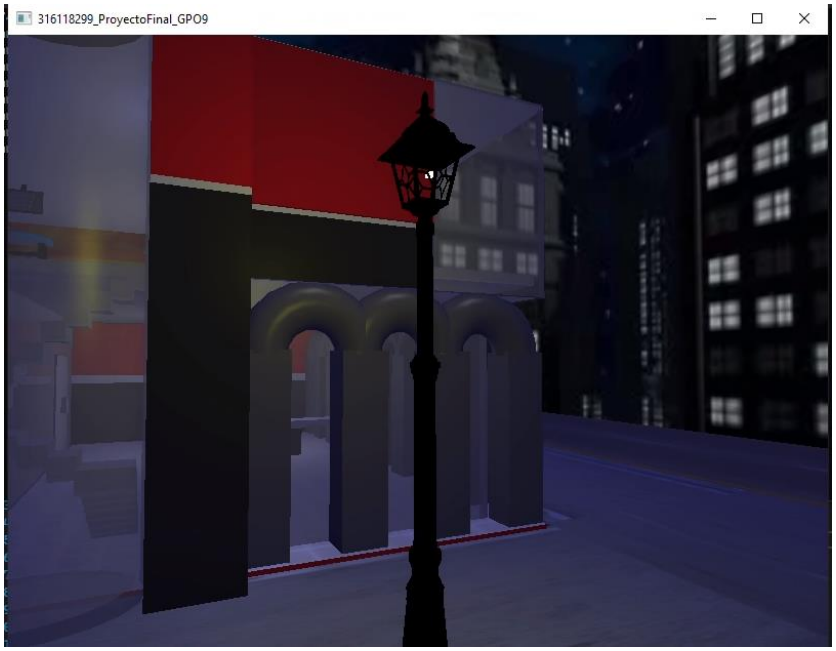


Lane Control Panel

Additional objects:

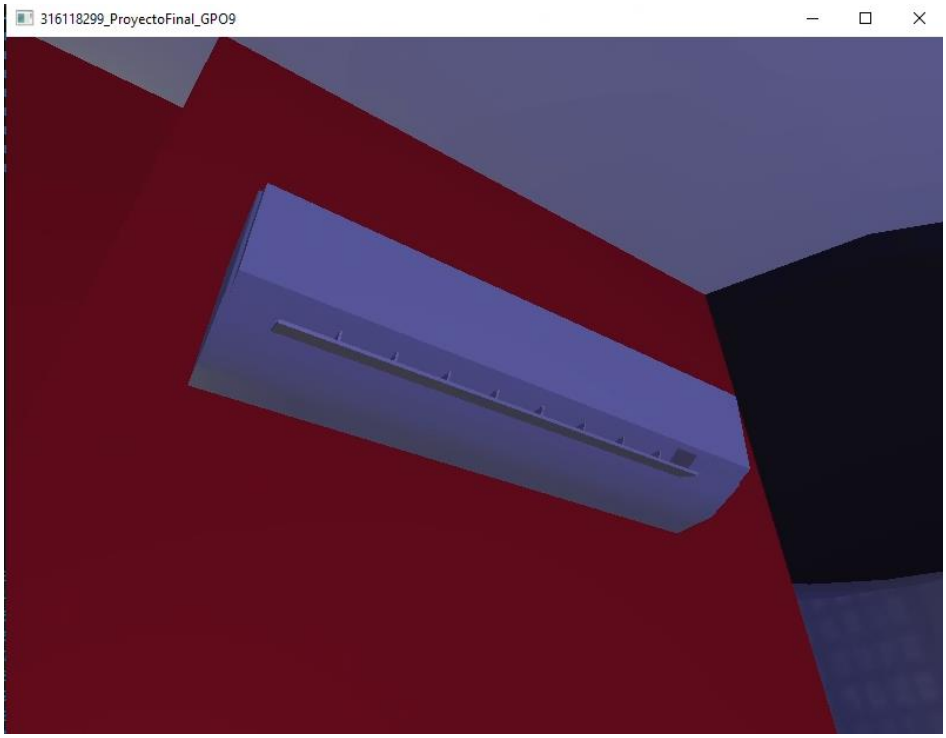
Some additional objects, which aren´t detailed on the reference document, were used to increase the realism of the scene. Most of these objects were obtained from the Internet, from https://www.turbosquid.com/, and are licensed for personal non-commercial use.

Lightpost

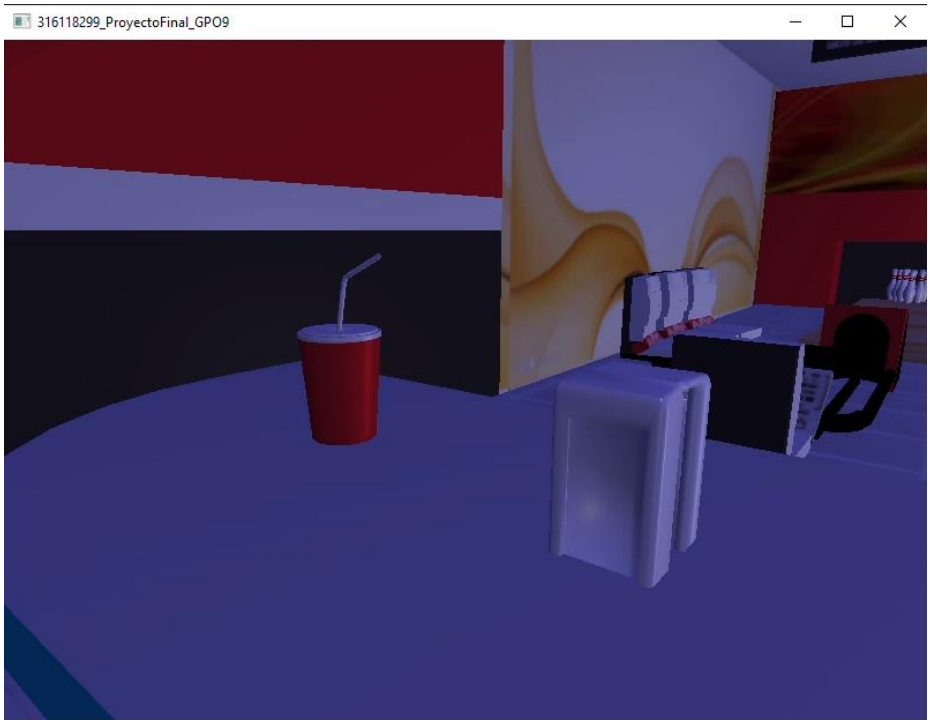


Computers

AC



Soda cup and napkin dispenser



Animations

Simple animations

Simple animations 1: Main doors

Both doors that make up the main entrance perform a rotational movement on the Y-axis ranging from 180° (gate closed) to 30° (gate open), due to the orientation of the model and the pivot. This animation is triggered using two Boolean flags in the KeyCallback() function. The first flag, doorMoving, controls the movement of the doors. The second flag, isDoorOpen, controls their direction depending on the current state (open or closed). This event is controlled by pressing the P key.

```
if (keys[GLFW_KEY_P])
{
    doorMoving = true;
    if (isDoorOpen)
    {
        isDoorOpen = false;
    }
    else
    {
        isDoorOpen = true;
    }
}
```

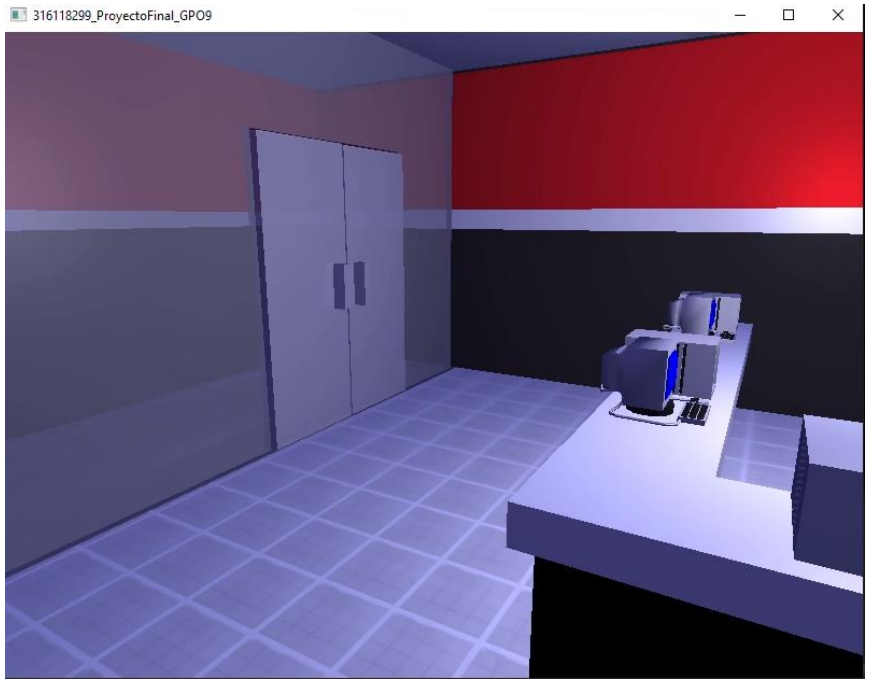
Within the DoMovement function, we include the necessary logic to perform the door rotation movement, using the DoorRot variable, which controls the rotation angle on the Y axis for each door. The direction of this increment (or decrement, depending on the current door state) is defined by the isDoorOpen flag. When this flag is true, the movement is performed by incrementing DoorRot up to 180°, otherwise it is decremented until it reaches 30°.

```
void DoMovement()
{
    //Puertas
    //DoorRot = 30 - abierto
    //        180 - cerrado
    if (doorMoving)
    {
        if (isDoorOpen)
        {
            if (DoorRot >= 180.0f)
            {
                DoorRot = 180.0f;
                doorMoving = false;
            }
            else
            {
                DoorRot = DoorRot + 2.0f;
            }
        }
        else
        {
            if (DoorRot <= 30.0f)
            {
                DoorRot = 30.0f;
                doorMoving = false;
            }
            else
            {
                DoorRot = DoorRot - 2.0f;
            }
        }
    }
```

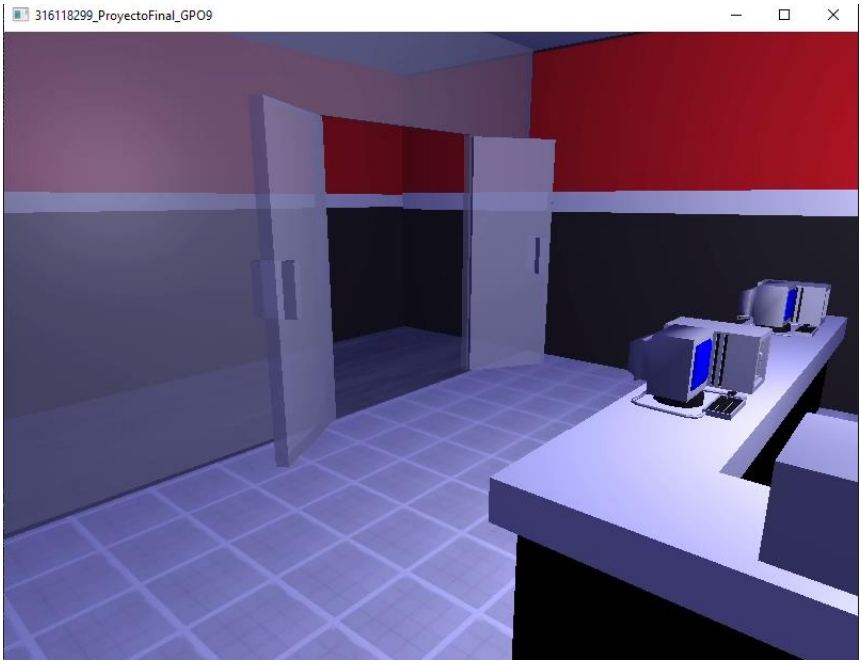
Inside the main function, both doors perform a translation to accommodate the model in the desired position, and a Y axis rotation for each door. The direction in which each door rotates is reversed for the left gate, so as to use the same rotation variable and reduce the computational load of the program.

```
//Puertas
//Animacion sencilla 1
tmp = model = modelPos;
model = glm::translate(model, glm::vec3(-12.5f, -1.2f, -30.7f));
model = glm::rotate(model, glm::radians(-DoorRot), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 1.0);
PuertaIzq.Draw(lightingShader);
model = tmp;
model = glm::translate(model, glm::vec3(-12.5f, -1.2f, -20.7f));
model = glm::rotate(model, glm::radians(DoorRot), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 1.0);
PuertaDer.Draw(lightingShader);
```

Animation state 1: Doors closed

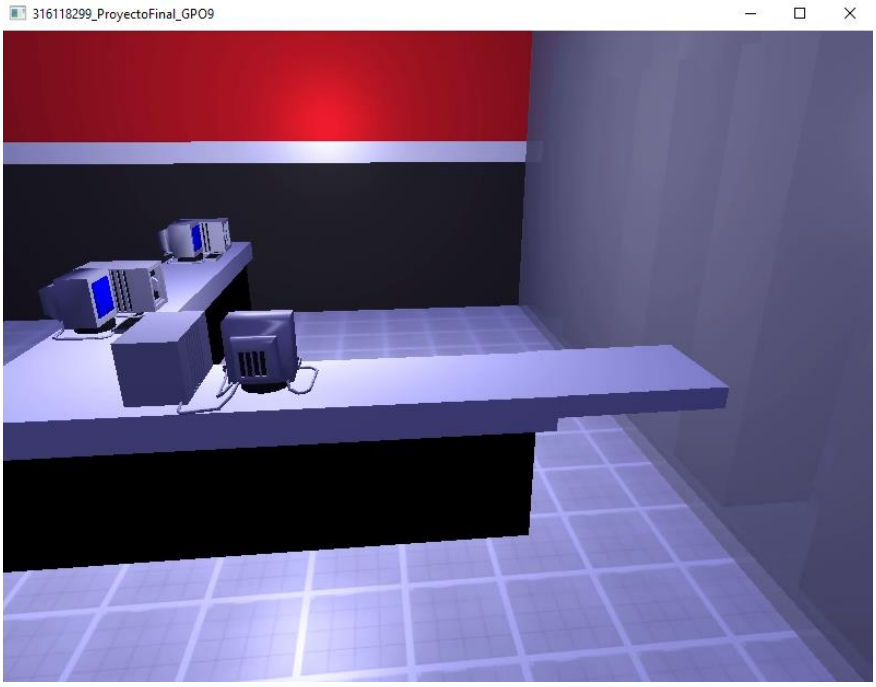


Animation state 2: Doors are open



Simple animation 2: Reception door

To perform the movement of the reception door, a process similar to the one defined in Simple animations 1: Main doors. We use two Boolean flags to determine if it should move, and the direction of rotation, so as to directly modify this rotation in the model transformation.

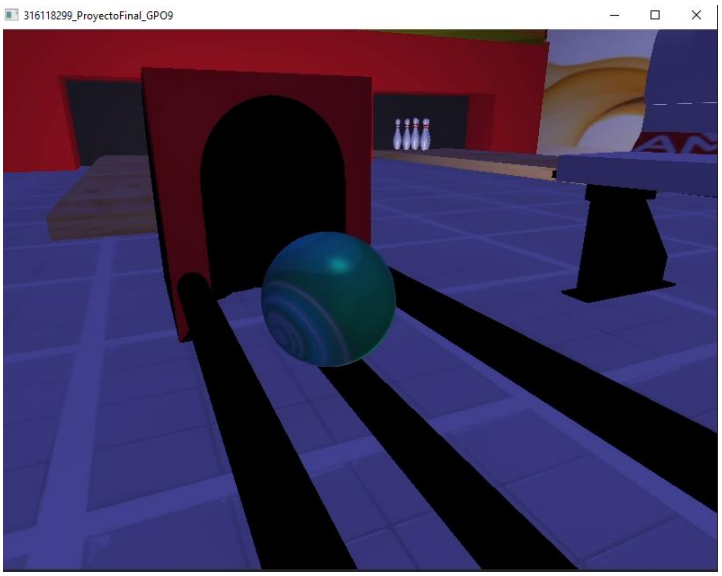


```
//Puerta recepcion - Animacion sencilla 2
view = camera.GetViewMatrix();
model = modelPos;
model = glm::translate(model, glm::vec3(15.8f, -3.9f, -15.5f));
model = glm::rotate(model, glm::radians(recRot), glm::vec3(0.0f, 0.0f, 1));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
Recepcion.Draw(lightingShader);
```

```
if (keys[GLFW_KEY_R])
{
    recMoving = true;
    if (isRecOpen)
    {
        isRecOpen = false;
    }
    else
    {
        isRecOpen = true;
    }
}
```

Simple animation 3: Ball rotation

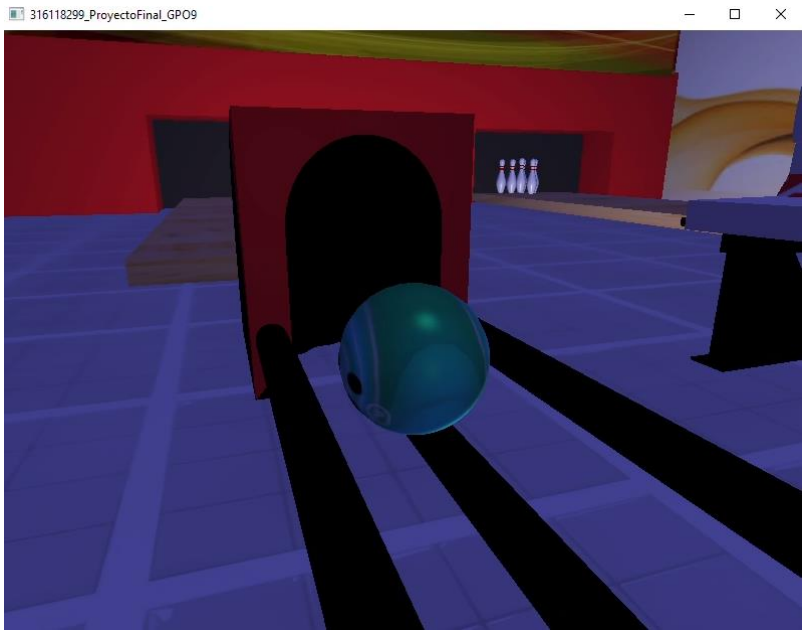
Due to the spherical shape of the bowling ball, when it rolls, it will rarely do so on a single axis, due to the smoothness of the waxed lane, or the initial position on the ball collector. Because of this effect, a rand() function was implemented to determine the increment of rotation and the direction of rotation on each of the axes.



As this object is part of the keyframe animation, we use this opportunity to set the rotation increment for each axis. Inside the animation function, the following code fragment allows us to obtain the rotation speed on each axis using pseudo-random numbers.

```
if ((playIndex == 1) || (playIndex==2))
{
    ballMoving = false;
}
else
{
    ballMoving = true;
}
amX = rand() % 20;
amY = rand() % 20;
amZ = rand() % 20;
if ((rand() % 4) >= 2)
{
    amX = -amX;
}
if ((rand() % 4) >= 2)
{
    amY = -amY;
}
if ((rand() % 4) >= 2)
{
    amZ = -amZ;
}
```

The flag isBallMoving allows us to stop or start the rotation movement of the ball. Due to the way the keyframe animation was set up, which will be detailed in later sections in Complex animation 1: Bowling, the ball is not always rolling, because there are certain frames in which the ball remains static (when it rises and settles on the lane), so we can deactivate it using this boolean flag.



Inside the DoMovement() function, we simply evaluate the value of this flag.

```
//Bowling Ball
if (ballMoving)
{
    ballRotX += amX;
    ballRotY += amY;
    ballRotZ += amZ;
}
```

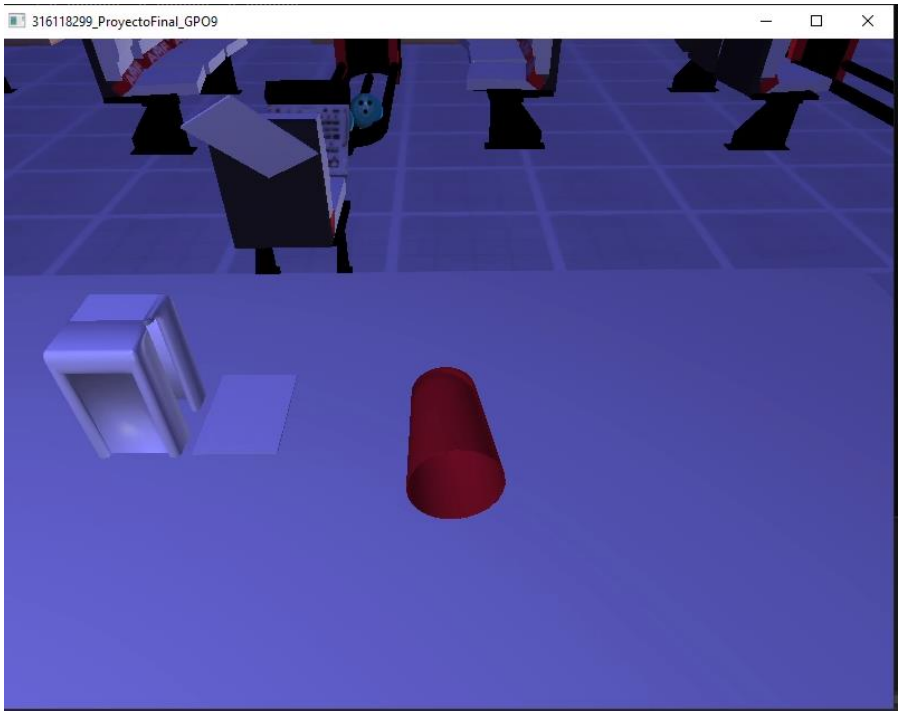
Finally, we modify the model matrix to accommodate a rotation on each axis.

```
//Bola Bolos
view = camera.GetViewMatrix();
model = modelPos;
model = glm::translate(model, glm::vec3(ballPosX,ballPosY,ballPosZ));
model = glm::rotate(model,glm::radians(ballRotX), glm::vec3(1, 0, 0));
model = glm::rotate(model, glm::radians(ballRotY), glm::vec3(0, 1, 0));
model = glm::rotate(model, glm::radians(ballRotZ), glm::vec3(0, 0, 1));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
BowlingBall.Draw(lightingShader);
```

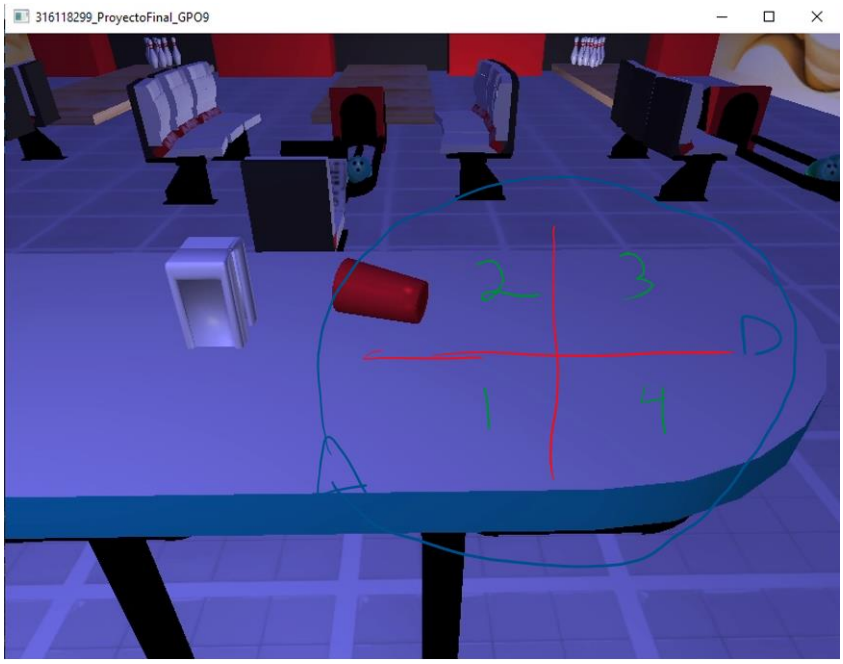
Simple animation 4: Empty soda cup

For both this animation and Complex animation 2: Flying napkins, we imagine a wind current flowing from the AC unit on the wall.

This airflow affects both the napkins lying on the table and the empty soda cup lying beside it.



If we look at the direction of the airflow, we realize that we can divide the movement of the cup into 4 sections or quadrants, depending on the impact that the air flow has on its speed.



In the first quadrant, the air flow is direct and hits directly inside the cup. Therefore, the velocity of the movement must be higher. In quadrant 2, the flow is still direct, but there is a greater distance between the cup and the AC. In quadrant 3, there is no more airflow, but the velocity reached in quadrants 2 and 3 should have an impact on the current speed, although it's still decreasing. Finally, the fourth quadrant, the speed is minimal, until the flow is again directly hitting the cup's interior

Simple animation 5: Chair

Same as with Simple animations 1: Main doors and Simple animation 2: Reception door, we use two flags, but instead of rotating, we translate the model matrix.

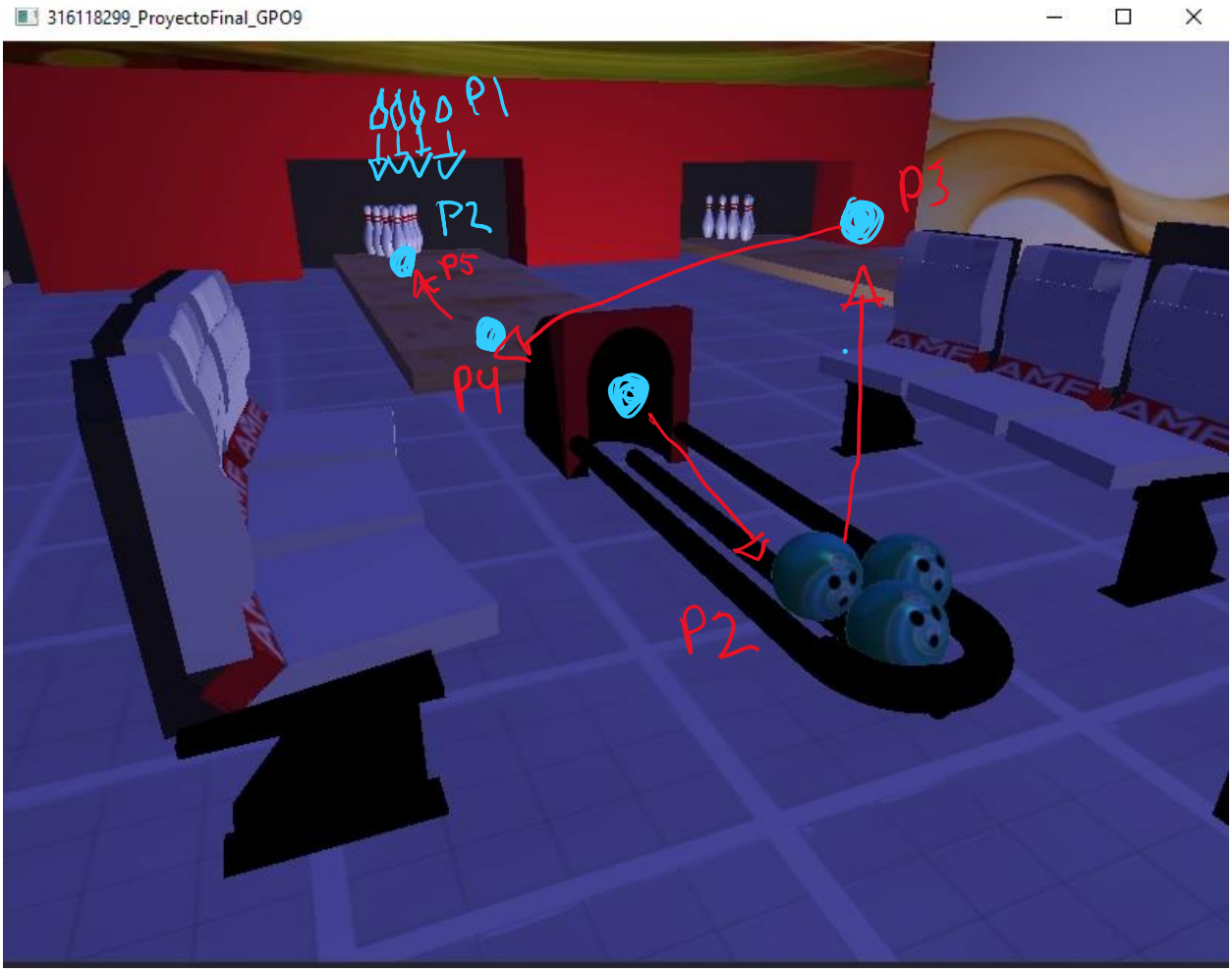
Animation state 1

Animation state 2

Complex animations

Complex animation 1: Bowling

To animate the bowling game, keyframe animation will be used. For this we need to define a series of points that define the most significant positions prior to interpolation.



To achieve a looping animation that represents the movement of the objects on the bowling game, we start with the ball inside the collector, as if it were the first time it is returned from the lane. The pins are placed on the pin setter (not visible). We call this the initial frame, or P1.

Balls:

P1: (0.75, 11.725, -13)

Pins:

P1: (0, 10, 0)

Subsequently, frame P2 is when the bowling ball reaches the end of the collector and is placed alongside the rest of the bowling balls. The pins descend from the pin setter and are placed in the game position.

Balls:

P2: (0.4, 11.725, -13)

Pins:

P2: (0, 0, 0)

Frame P4 represents the bowler's throw, so the only moving object is the ball, which starts at the beginning of the lane. To reach this position, we require an additional frame, P3, so that the movement does not involve crossing the collector as a ghastly ball. We raise the ball, as if the bowler was lifting it. For the next frame, we place the ball at the beginning of the lane.

Ball:

P3: (0.75, 15, -7-5) ->

P4: (0.75, 11.725, -19)

Frame P5 has the ball speeding through the lane to the spot just before hitting the pins.

Ball:

P5: (0.75, 11.725, -34)

For frame P6, we want the pins to rotate just as the ball hits them, while the ball itself disappears behind the lane. The sweeper descends from its position to clean the lane.

Ball:

P6: (0.75, 11.725, -40)

Pins:

P6: (0, -0.5, 0)

Pin rotation:

P6: 90°

Sweeper:

P6: (0, -4, 0)

On frame P7, the sweeper moves the pins behind the lane and disappears as well.

Pins:

P7: (0, 0, -10)

Sweeper:

P7: (0, -4, -5)

Frame P8 sees the sweeper returning to its initial position

P6: (0, 0, -5)

Finally, at frame P9 we reset the initial values and the animation starts again. This frame was added because the objects seemed to "pop-in", that is, at the moment of restarting the animation, they were immediately transported to the position of P1, therefore a smoother animation or transition was sought.

The additional effects for this animation are those detailed in Simple animation 3: Ball rotation. Since we want the animation to stop between P3 and P4 (which is when the ball is lifted prior to a throw), we can turn off the motion using the boolean flag of the ball's movement during the aforementioned frames. We do this inside the animation() function, inside the conditional increment to playIndex.

```
if ((playIndex == 1) || (playIndex==2))
{
    ballMoving = false;
}
```

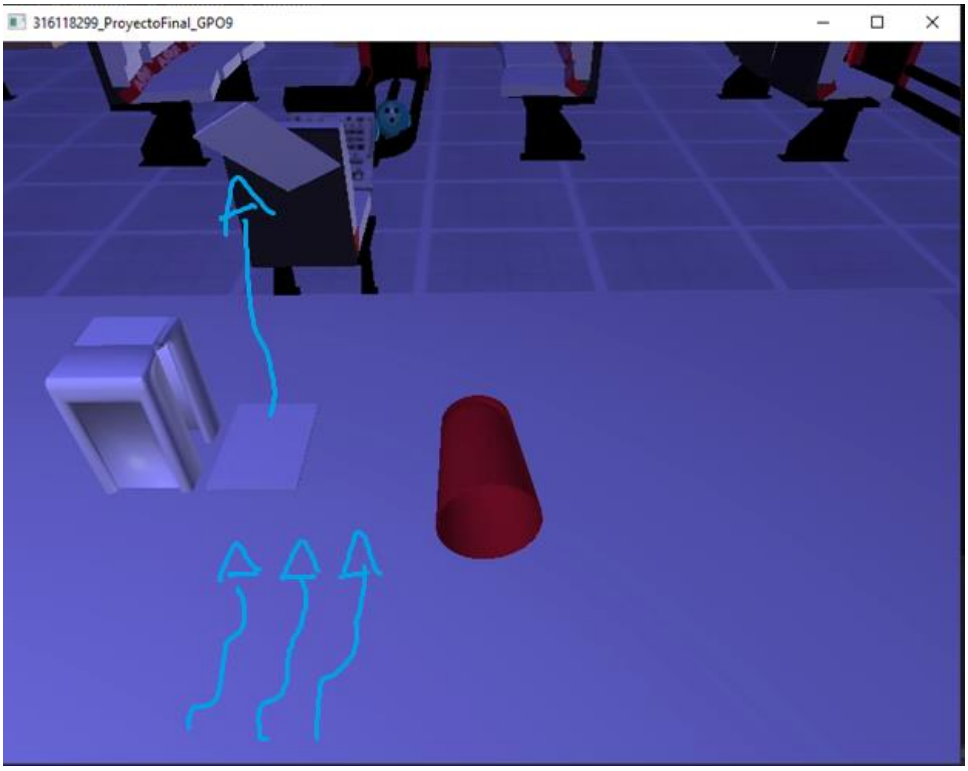
As we want to start rotating the ball just before it exits the collector, we add the code from Simple animation 3: Ball rotation. Inside the KeyCallback() function we paste the same snippet just as we call the keyframe animation.

Inside the animation() function itself, since we want the motion to be different for each of the frames (it makes no sense for the ball to rotate the same way at the speed of the collector as when the bowler's force is applied to knock down the pins), we simply recalculate the rotation on each axis with our pseudo-random number functions.

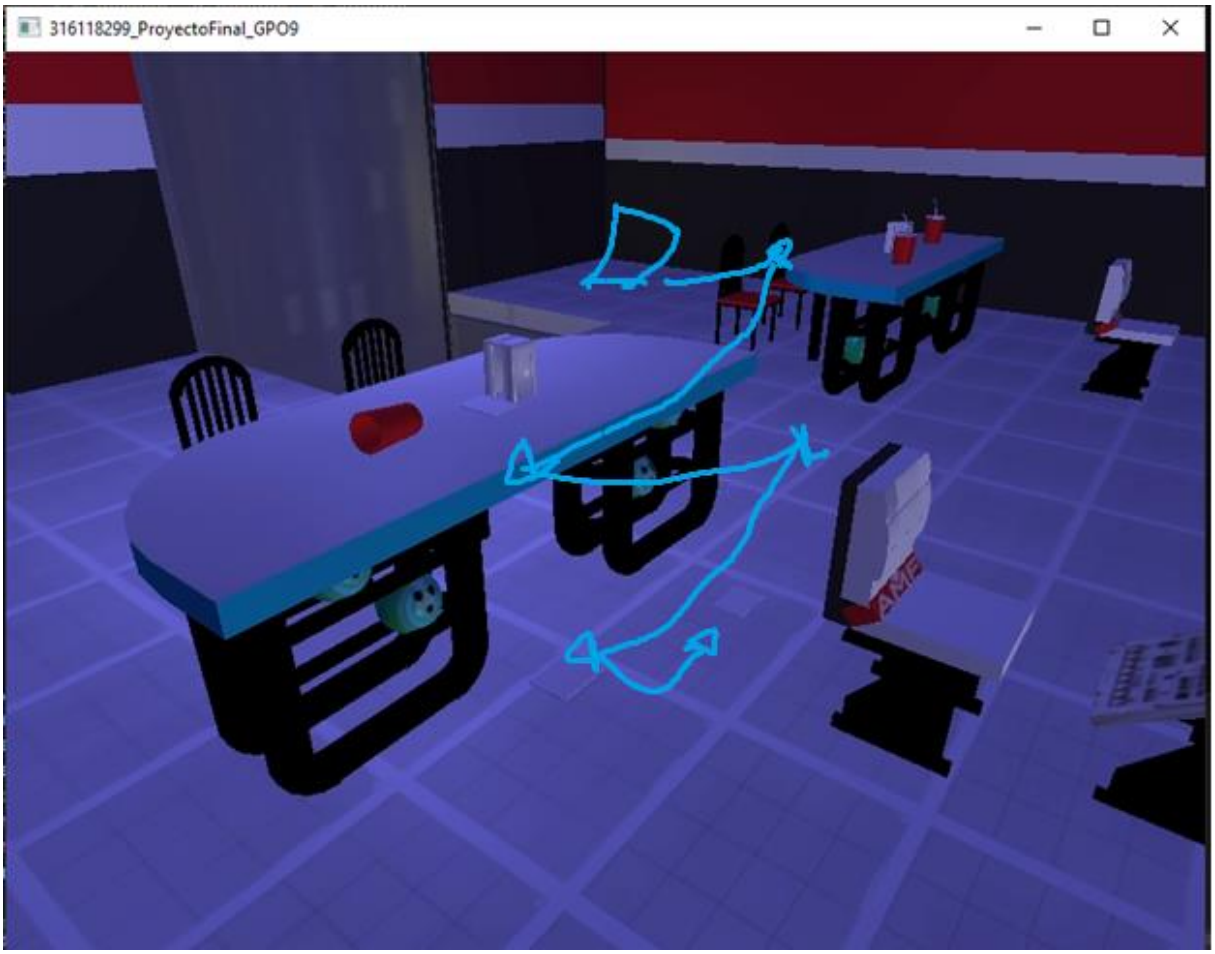
```
        printf("termina anim\n");
        playIndex = 0;
        play = false;
        LightP1 = glm::vec3(0.0f, 0.0f, 0.0f);
    }
    else //Next frame interpolations
    {
        i_curr_steps = 0; //Reset counter
                          //Interpolation
        interpolation();

        LightP1 = glm::vec3(1.0f, 1.0f, 1.0f);


        if ((playIndex == 1) || (playIndex==2))
        {
            ballMoving = false;
        }
        else
        {
            ballMoving = true;
        }
        amX = rand() % 20;
        amY = rand() % 20;
        amZ = rand() % 20;
        if ((rand() % 4) >= 2)
        {
            amX = -amX;
        }
        if ((rand() % 4) >= 2)
        {
            amY = -amY;
        }
        if ((rand() % 4) >= 2)
        {
            amZ = -amZ;
        }
    }
}
```

Complex animation 2: Flying napkins

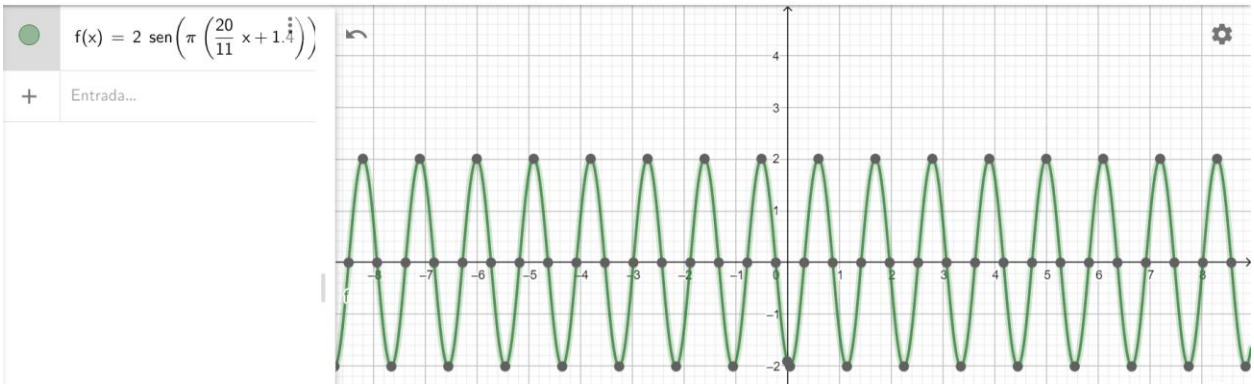
As was mentioned earlier in Simple animation 4: Empty soda cup, the AC unit is emitting an airflow that acts on the lighter objects on top of the central table.



Napkins get carried by the airflow and afterwards, the descend to the floor oscillating between 2 positions, due to the air resistance on the paper sheet. For this animation, we use an state machine.



For the oscillation effect, we will use the trigonometric functions. Since our transformation starts at 0 (origin), we use the sine function.



We determine the amplitude value to obtain the range of motion in the X-axis. Then, we multiply the argument by π to normalize the oscillation, i.e., that the discrete times we sample is in the time domain. Finally, we look for a value of the period that allows us to repeat the period as many times as necessary to obtain the desired effect.

```
//Maquina estados servilleta
double count = 0;
if (napkinAttached)
{
    servPos = glm::vec3(0.0f, 0.0f, 0.0f);
    servRot = 0;
    count = 0;
    napkinFlew = true;
    napkinAttached = false;
}

if (napkinFlew)
{
    if ((servPos.y >= 2.5f) && (servRot >= 360.0f) && (servPos.z <= -3.0f) && (servRot2 >= 270))
    {
        napkinFlew = false;
        napkinFlying = true;
    }
    else
    {
        if (servPos.z <= -3)
        {
            servPos.z = -3;
        }
        else
        {
            servPos.z -= 0.05f;
        }

        if (servPos.y >= 2.5)
        {
            servPos.y = 2.5;
        }
        else
        {
            servPos.y += 0.05f;
        }

        if (servRot >= 360)
        {
            servRot = 360;
        }
        else
        {
            servRot += 5;
        }

        if (servRot2 >= 270)
        {
            servRot2 = 270;
        }
        else
        {
            servRot2 += 5;
        }
    }
}
```

Ilumination:

Directional light

Como se mencionó en el apartado de Ambientación, la meta principal es la de representar la fachada en cuestión con una luz nocturna, es decir que la luz direccional, la que compone la principal componente de iluminación de la escena, debe de irradiar una luz muy tenue, y con un tinte azulado, como la luz que irradia la Luna sobre la Tierra. El Skybox refleja la misma ambientación de una ciudad por la noche. Estas son las horas de mayor afluencia del negocio representado, y también la recreación anecdótica en la que se basa esta fachada.

As mentioned on the Ambientation section, the main goal is to represent the building in question at nighttime, so the directional light, which makes up the main lighting component of the scene, should radiate a very dim light, and with a bluish tint. The Skybox reflects the same ambience of a city at night. These are the peak business hours represented, and also the anecdotal recreation on which this scene is based.

Point lights

Point light 1:

La primera luz puntual se encuentra ubicada en el centro de la recepción, e irradia una luz blanca con una intensidad de 200 unidades. Con esta luz puntual se denota que el interior de la fachada está en funcionamiento, y provee iluminación a las áreas de trabajo y a los clientes que ingresan al establecimiento.

The first point light is located at the center of the reception area and radiates a white light with an intensity of 200 units. This point light indicates that the building is in operation, and provides illumination to the work areas and customers entering the establishment.

```
// Point light 1
//PB
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y, pointLightPositions[0].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), 1, 1, 1);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), 0.5, 0.5, 0.5);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.022f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 0.0019f);
```

Point light 2:

The second point light is the one located at the top, on the second floor in the bowling area.

```
// Point light 2
//1F
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].position"), pointLightPositions[1].x, pointLightPositions[1].y, pointLightPositions[1].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].diffuse"), 0.5f, 0.5f, 1.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].specular"), 0.1f, 0.1f, 0.2f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].linear"), 0.07f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].quadratic"), 0.017f);
```

Again, a blue-tinted light with an intensity of 65 units was chosen to illuminate the tables, where the food sold by the establishment is consumed. This lighting is necessary so as not to affect the bowling lanes area, which have a dimmer style of lighting, a remnant of the neon wave of the 80s.

Point light 3:

La tercera luz puntual es la que ilumina la parte exterior de la fachada. Se optó por una luz con tinte amarillo, como la mayoría de las luminarias de la CDMX, donde se ubica el establecimiento. La iluminación irradia principalmente en la fachada, pero también hacia la banqueta y el farol. Esta es la única instancia de luz puntual en la que se mantuvo el dibujado de un cubo que representa la posición de la luz utilizando el lampShader, de modo que represente el foco dentro de la luminaria.

The third point light is the one that illuminates the outside of the building. A yellow-tinted light was chosen, like most of the luminaires in Mexico City, where the establishment is located. The lighting radiates mainly on the building, but also towards the sidewalk and the streetlight. This is the only instance of point light in which the drawing of a cube representing the position of the light using the lampShader was maintained, so that it represents the bulb inside the luminaire.

Point light 4:

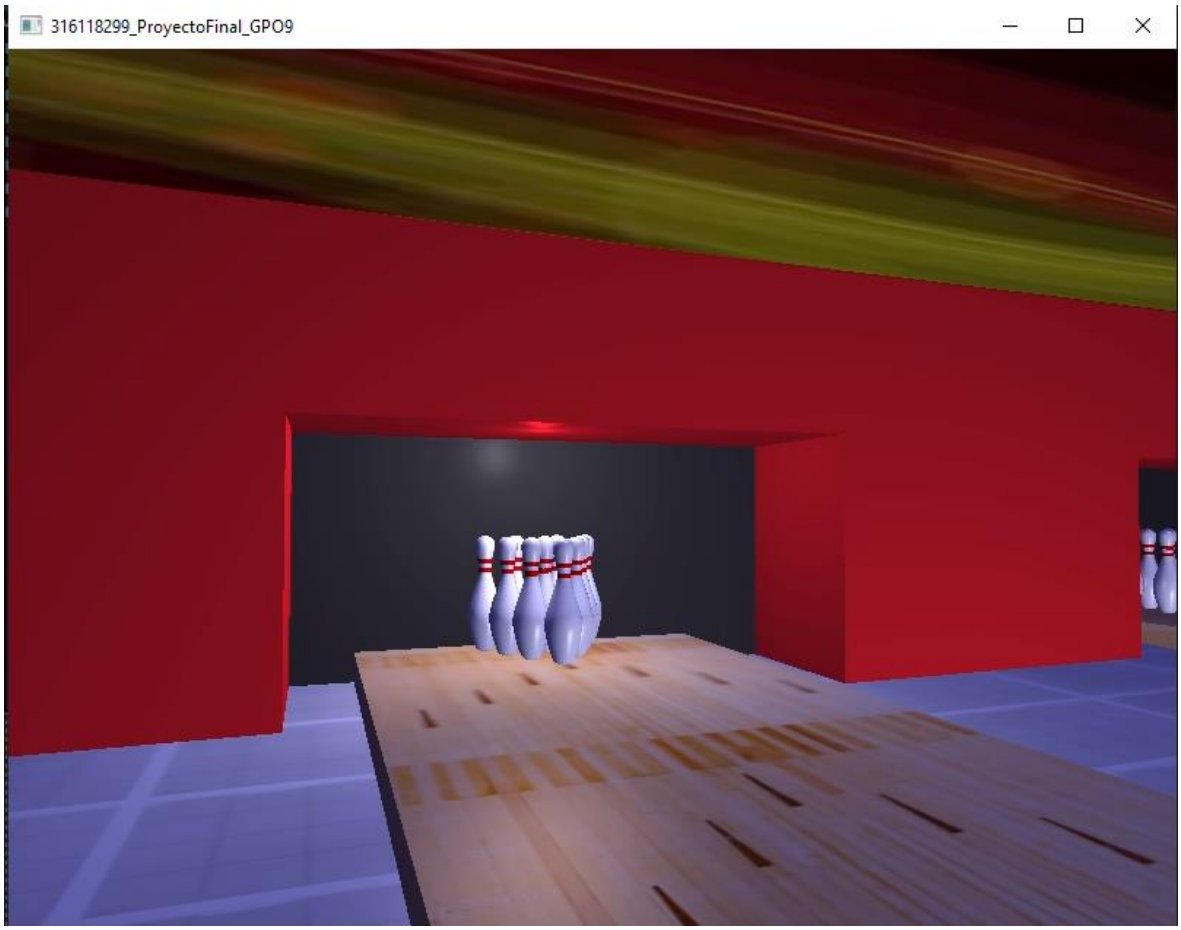
Finalmente, la última posición de las luces puntuales es la que se encuentra encima del carril activo, en este caso, encima de los bolos del carril central. Estas luces simbolizan los carriles activos, y permiten al jugador visualizar los bolos. Solamente las líneas activas se encuentran iluminadas. Esta luz puntual se activa o desactiva dependiendo del estado de la animación, la cual simboliza un juego de un único tiro, por razones de ambientación. Al iniciar la animación detallada en Animación compleja 1: Juego de bolos y proceder a P2, la luz se activa, y al finalizar la animación, la luz vuelve a su estado apagado.

Finally, the last of the point lights is the one above the active lane, in this case, above the center lane pins. These lights represent the active lanes and allow the player to visualize the pins. Only the active lanes are illuminated. This point light is activated or deactivated depending on the state of the animation, which symbolizes a single-shot game, for ambience reasons. When starting the animation detailed in Complex animation 1: Bowling and after transitioning to P2, the light is activated. Then at the end of the animation, the light is turned off.

```
// Point light 4
//Bolera
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].position"), pointLightPositions[3].x, pointLightPositions[3].y, pointLightPositions[3].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].diffuse"), LightP1.x, LightP1.y, LightP1.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].specular"), LightP1.x, LightP1.y, LightP1.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].quadratic"), 0.032f);
```

Activation is done at the animacion() function

```
if (play)
{
    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;
            LightP1 = glm::vec3(0.0f, 0.0f, 0.0f);
        }
        else //Next frame interpolations
        {
            i_curr_steps = 0; //Reset counter
                              //Interpolation
            interpolation();

            LightP1 = glm::vec3(1.0f, 1.0f, 1.0f);
```



Project conclusions

Working on the present project, it is easy to realize that the field of computer graphics is one of high complexity, and requires a great mathematical and computational background, and takes inspiration from the other fields of science, not to mention the imagination, creative thinking and anecdotal memory of the development team. In addition, all knowledge acquired in this field is incremental, that is, on top of the basics, we keep adding knowledge to perform more complex tasks. The final result is perhaps what users without previous knowledge of computing find most interesting, as it allows a high level of interaction and visualization by the user. It is not a simple field, but the results obtained are of great value and usefulness for many professions, ranging from marketing to simulation, medicine and education. Therefore, it is considered that the objectives proposed in Laboratorio de Computación Gráfica E Interacción Humano-Computadora were achieved, and that the knowledge acquired is of great value as the basis on which we can generate and build new knowledge for high-level projects.