Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en Computación

Semestre 2022 – 2

Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Proyecto Final

Documentación

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

Contenido

Diccionario de variables:

Nombre	Tipo	Descripción	Nivel de acceso
DoorRot	Punto flotante		Global
recRot	Punto flotante		Global
chairZ	Punto flotante		Global
refRot	Punto flotante		Global
ballRotX	Punto flotante	Define la rotación, en grados angulares, de la bola de bolos en el eje X.	Global
ballRotY	Punto flotante	Define la rotación, en grados angulares, de la bola de bolos en el eje Y.	Global
ballRotZ	Punto flotante	Define la rotación, en grados angulares, de la bola de bolos en el eje Z.	Global
amX	Punto flotante	Define qué tanto incrementa el valor de ballRotX en cada delta de tiempo del procesador	Global
amY	Punto flotante	Define qué tanto incrementa el valor de ballRotY en cada delta de tiempo del procesador	Global
amZ	Punto flotante	Define qué tanto incrementa el valor de ballRotZ en cada delta de tiempo del procesador	Global
isDoorOpen	Booleana	Controla la dirección en la que debe de moverse la puerta	Global
doorMoving	Booleana	Controla si debe o no animarse la apertura de la puerta	Global
isRecOpen	Booleana	Controla la dirección en la que debe de moverse la puerta de la recepción	Global
recMoving	Booleana	Controla si debe o no animarse la apertura	Global

		de la puerta de la recepción	
isChairOpen	Booleana	Controla el movimiento de la puerta	Global
chairMoving	Punto flotante	El valor de esta variable define la traslación en el eje Z de la silla móvil	Global
drawLightCubes	Booleana	Controla si deben o no dibujarse los cubos que represnetan la posición de las luces puntuales	Global
ballMoving	Booleana	Controla si debe o no rotar la bola de boliche	Global
refDir	Booleana	Controla la rotación del vaso vacío	Global
servPos	Vector3	Aloja y modifica la posición de la servilleta mientras vuela	Global
servRot	Flotante	Modifica la rotación de la servilleta en el eje Y	Global
servRot2	Flotante	Modifica la rotación de la servilleta en el eje Y	Global
ballPosIni	Vector3	Define la posición inicial de la bola para la animación por keyframes	Global
escobPosIni	Vector3	Define la posición inicial de la escobilla para la animación por keyframes	Global
pinPosIni	Vector3	Define la posición inicial de los bolos para la animación por keyframes	Global
pinOffset	Vector3	Traslada los bolos para acomodarlos con respecto al centro del mundo	Global

pinRot	Punto flotante	Define la rotación de los bolos en X para la animación por keyframes	Global
LightP1	Vector3	Controla los valores de las componentes ambiental y difusa de la luz puntual 4	Global

Objetivos

Recrear virtualmente, utilizando software de modelado 3D y OpenGL en el lenguaje C++, una fachada, así como el entorno interior, utilizando los conocimientos adquiridos durante el curso de Laboratorio de Computación Gráfica e Interacción Humano-Computadora. Este entorno virtual debe de contener 5 animaciones, de las cuáles 3 deben ser sencillas y otras dos deben de ser complejas. Recrear 7 objetos que puedan visualizarse dentro del entorno.

Alcance del proyecto:

El usuario promedio, sin ningún trasfondo de computación, debe de ser capaz de navegar a través de este ambiente virtual e interactuar con los objetos que se encuentran dentro del mismo utilizando una computadora con Microsoft Windows, y los archivos redistribuibles de Visual C++ 2019. Se debe de contar con un procesador de gráficos relativamente moderno capaz de ejecutar OpenGL, 4 GB de memoria RAM y un procesador de 1.0 GHz o mejor. El usuario debe de interactuar y desplazarse a través de este entorno virtual utilizando un monitor, un teclado y un mouse.

El proyecto debe de proporcionarse en su totalidad y para su ejecución dentro de un archivo comprimido en formato ZIP, el cuál contiene el archivo ejecutable y todas las dependencias necesarias para que el usuario lo ejecute sin problemas en su computadora. Asimismo, debe de proporcionarse un manual de usuario y el presente documento para su consulta.

Limitaciones

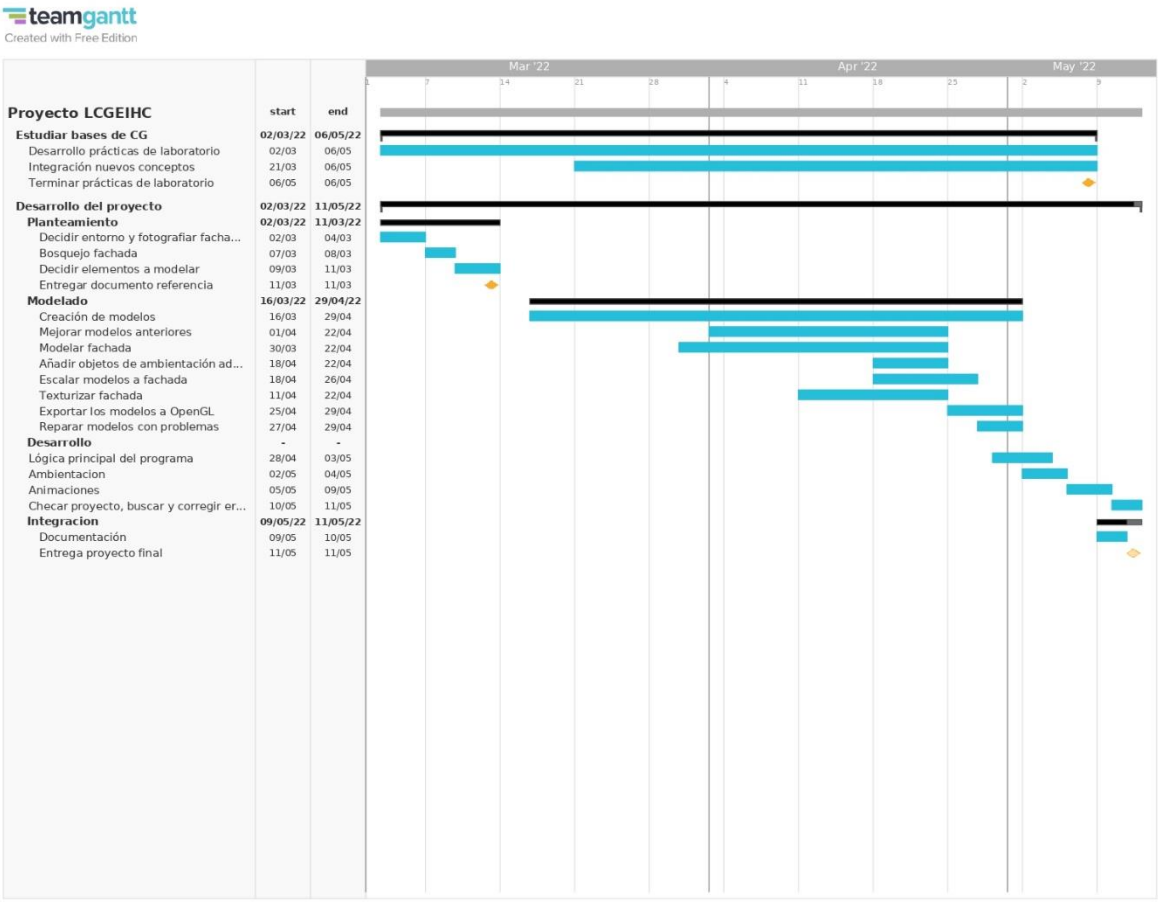
Debido a que las librerías empleadas están basadas en un equipo con sistema operativo Microsoft Windows de 32 bits, no es posible ejecutar este programa en MacOS, Linux, o cualquier otro sistema operativo sin software adicional. La compatibilidad con otros sistemas operativos no está soportada.

Es posible que las especificaciones mencionadas anteriormente no sean suficientes, dependiendo de la máquina del usuario. Debido a la falta de hardware, estas limitaciones no están contempladas en el desarrollo del proyecto.

En su calidad de recreación virtual de un entorno, este proyecto no es completamente fiel al entorno, y aunque busca la representación más apegada a la realidad, no es posible obtener un modelo 1:1 con todos los detalles que en el entorno están presentes.

Igualmente, al no tratarse de un videojuego, no se espera que la interacción del usuario se apegue más que a las definidas en el manual de usuario y en la documentación. Las interacciones recreadas son solamente las que se encuentran en el código fuente original, y las acciones del usuario no son capaces de cambiar el resultado de estas animaciones, esto se considera fuera del alcance del proyecto.

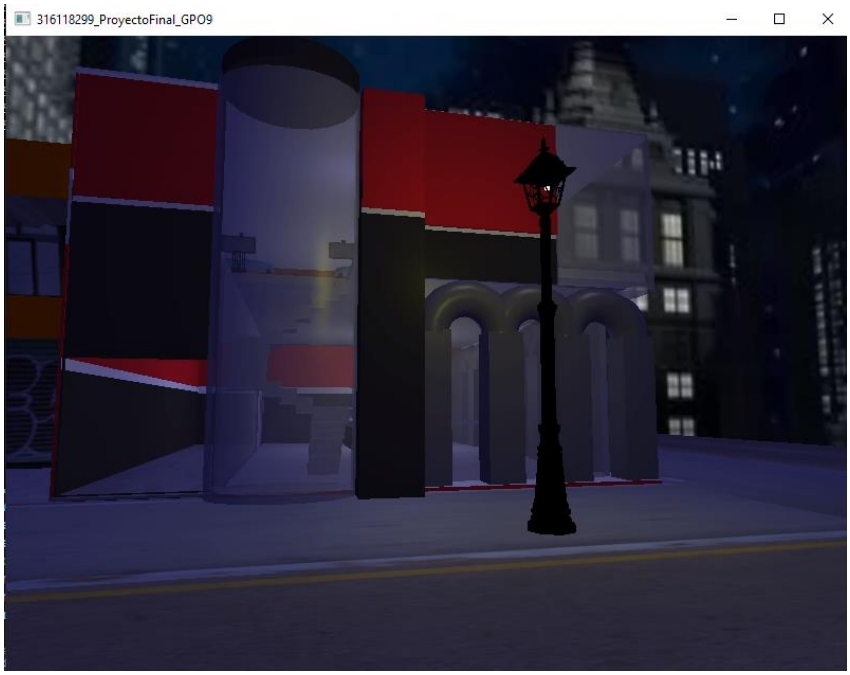
Diagrama de Gantt



Ambientación

Fachada

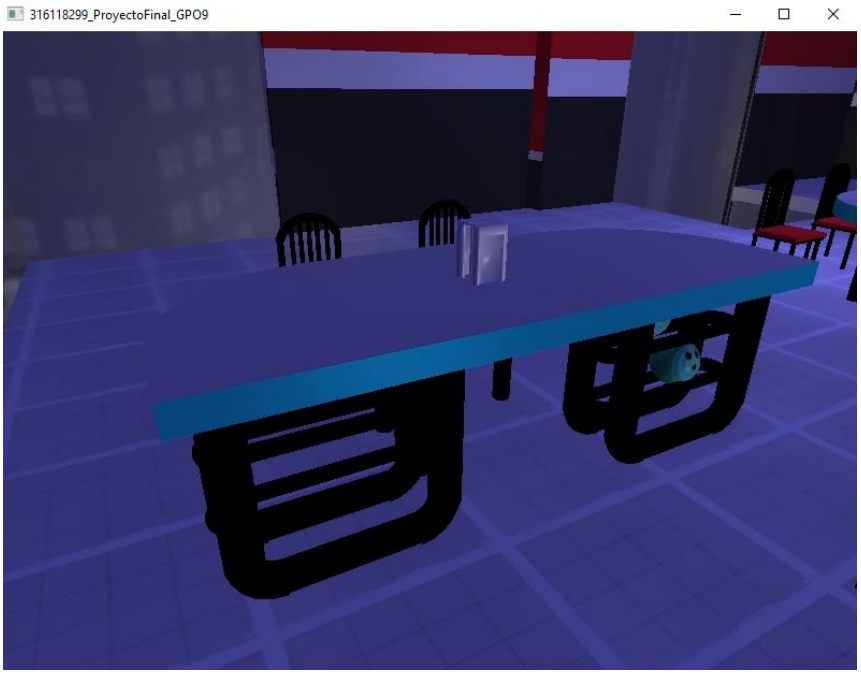
Se diseñó la fachada del Bol Montevideo, perteneciente a Boliches AMF. Para aumentar el realismo del ambiente, se modeló la fachada del edificio adyacente y se colocó un skybox que represente el entorno en el que se encuentra el edificio. Se optó por ambientar el proyecto de noche para mostrar de mejor manera la iluminación.



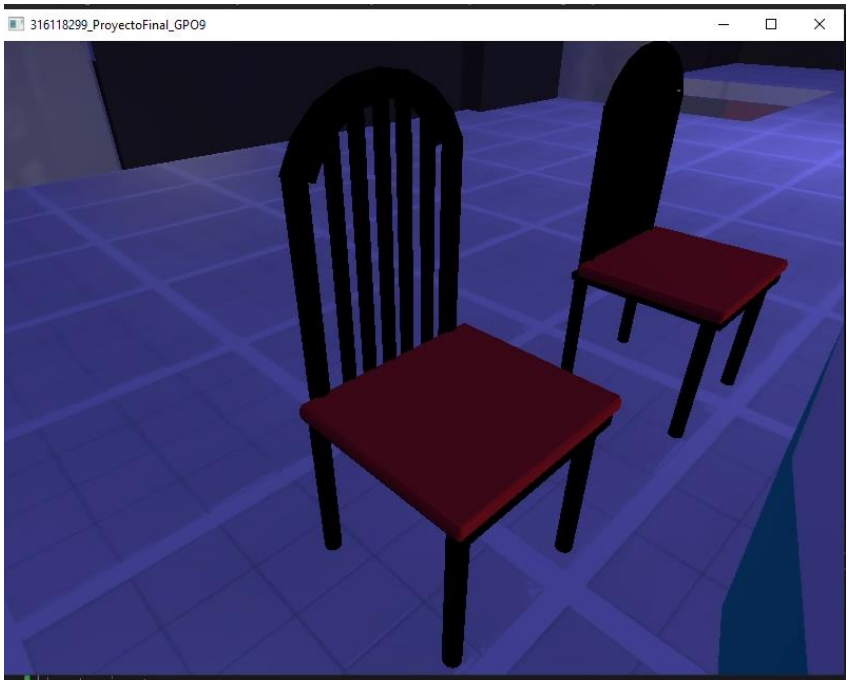
Objetos recreados:

Tal como se estipula en los requisitos del proyecto, se deben de recrear 7 objetos del entorno como modelos 3D que puedan ser importados a OpenGL.

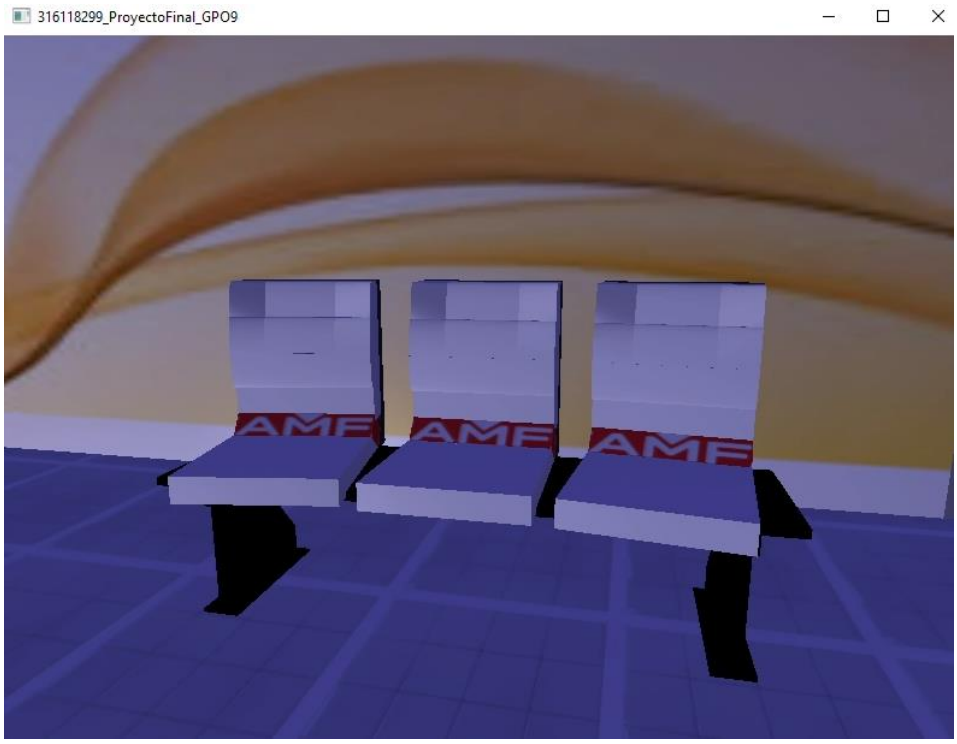
Mesa



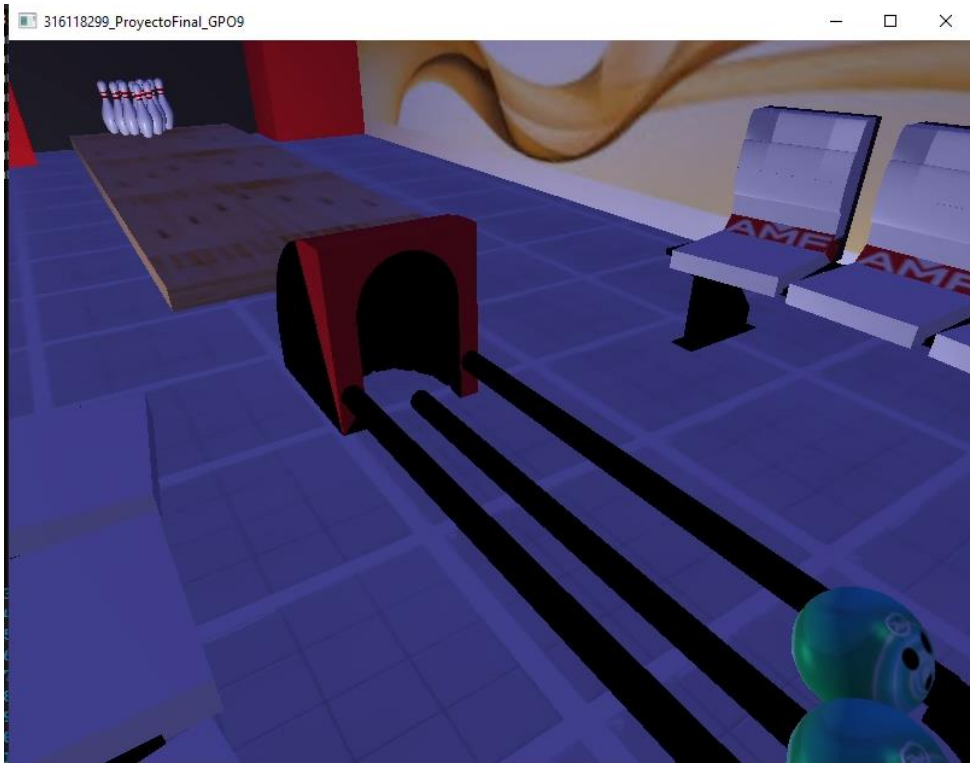
Sillas móviles



Sillas empotradas



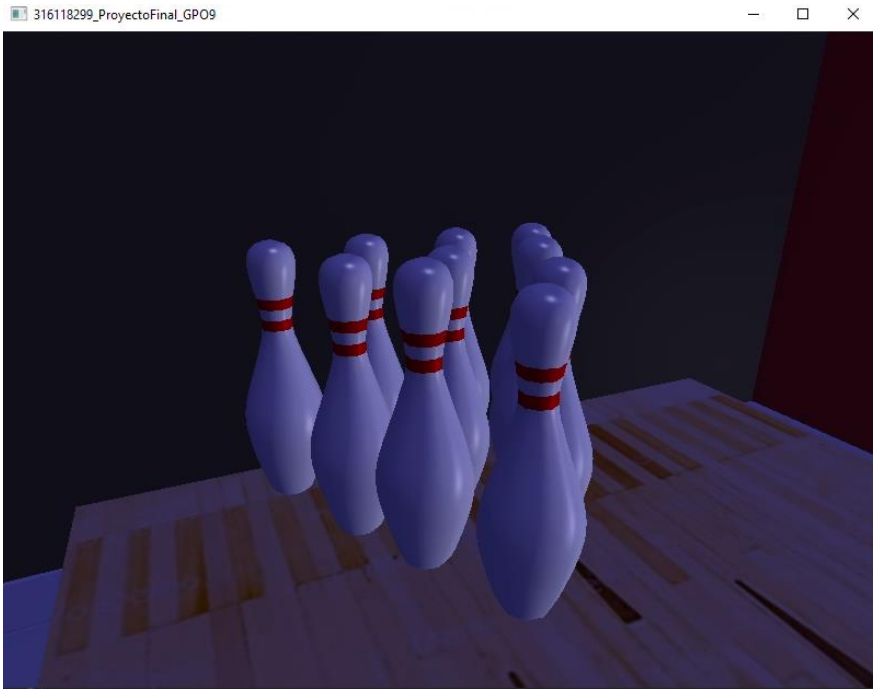
Maquina colectora



Bolas de bolos

Scoreboard

Bolos

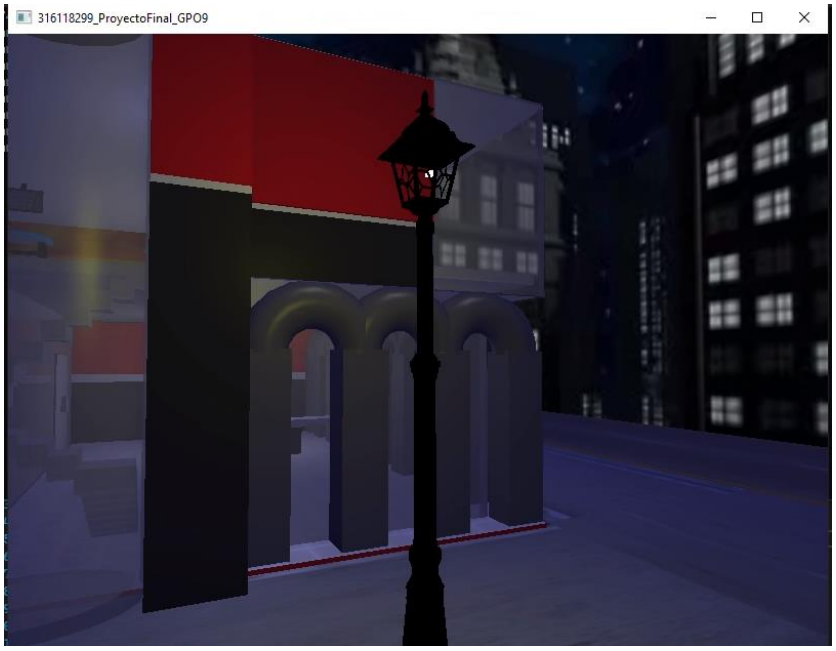


Panel de control

Objetos adicionales:

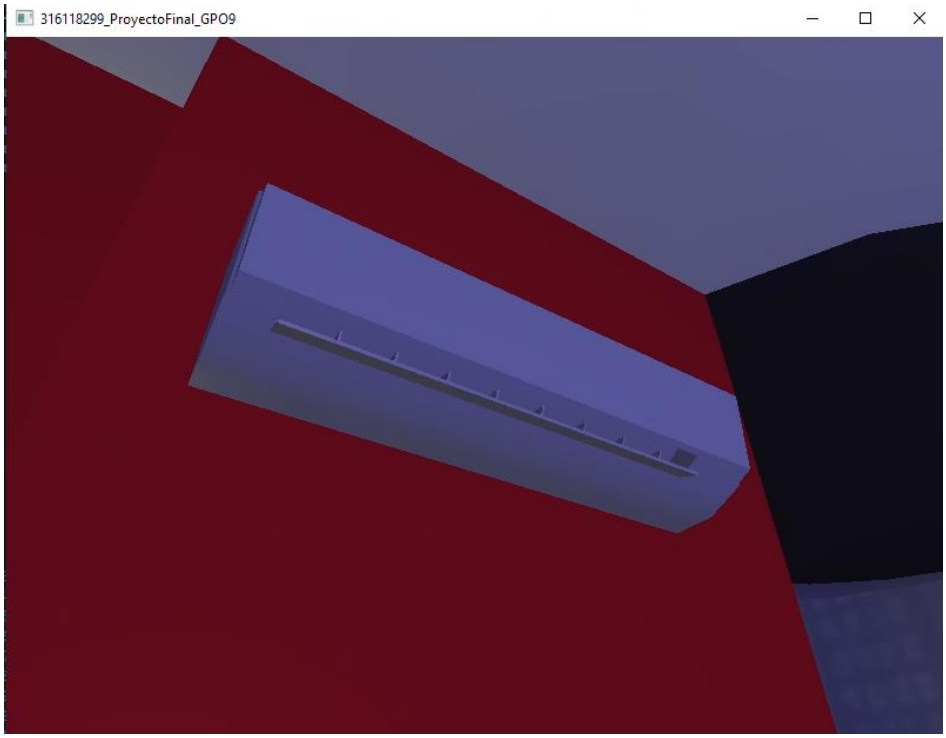
Se utilizaron algunos objetos adicionales que no están descritos en la entrega de la imagen de referencia. La mayoría de estos objetos se obtuvieron de internet, de https://www.turbosquid.com/, y cuentan con una licencia para uso personal no comercial. Estos objetos se incluyeron para añadir ambientación.

Farola

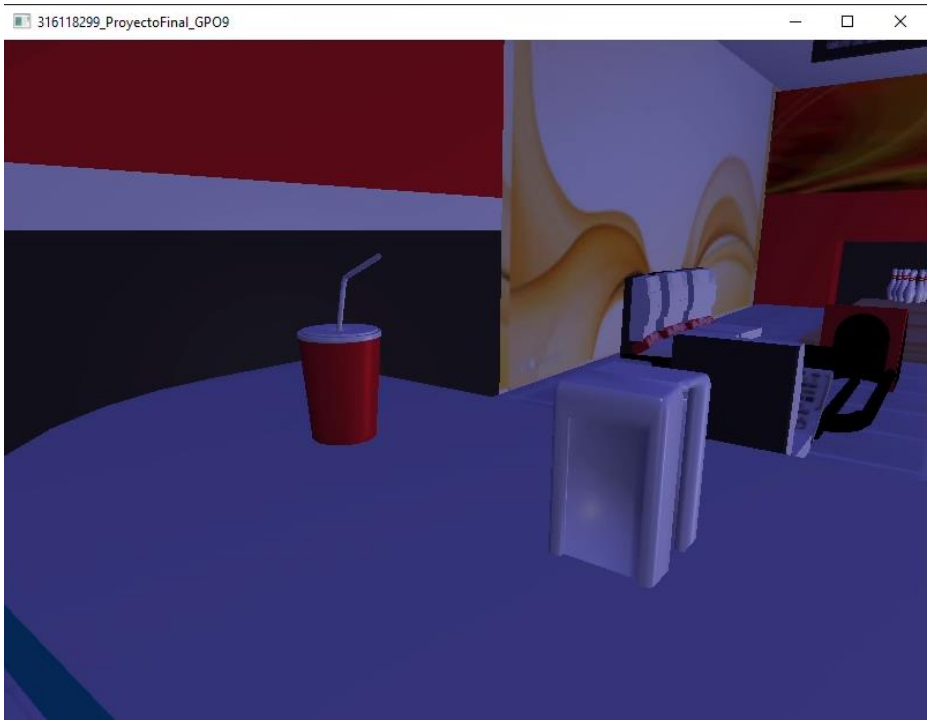


Computadoras

Aire acondicionado



Vaso de refresco y servilletero



Animaciones

Animaciones simples

Animación sencilla 1: Puertas

Las puertas que conforman la entrada principal realizan un movimiento de rotación en el eje Y que va desde 180° (puerta cerrada) hasta 30° (puerta abierta) debido a la orientación del modelo y del pivote. Esta animación se activa utilizando dos banderas booleanas en la función KeyCallback(). La primera bandera, doorMoving, controla el movimiento de las puertas. La segunda bandera, isDoorOpen, controla su dirección dependiendo del estado actual (abierta o cerrada). Este evento se controla presionando la tecla P.

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

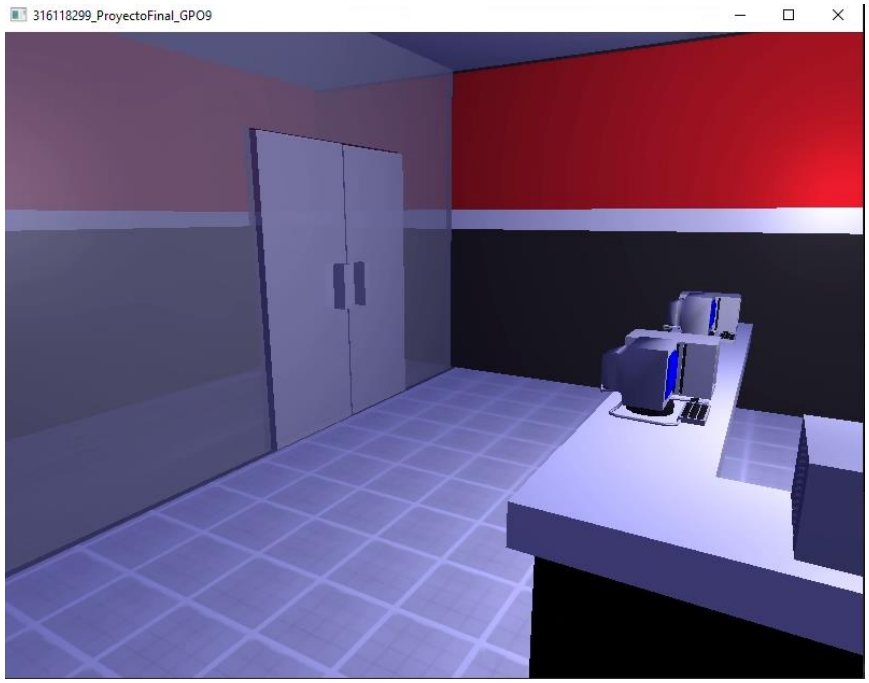
Dentro de la función DoMovement, se encuentra la lógica necesaria para realizar el movimiento de rotación de las puertas mediante la variable DoorRot, que controla el ángulo de rotación en el eje Y de cada puerta. La dirección de este incremento (o decremento según sea el caso) está definida por la bandera isDoorOpen. Cuando esta bandera es verdadera, el movimiento se realiza incrementando DoorRot hasta 180°, en caso contrario, se decrementa hasta llegar a 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;
            }
        }
    }
```

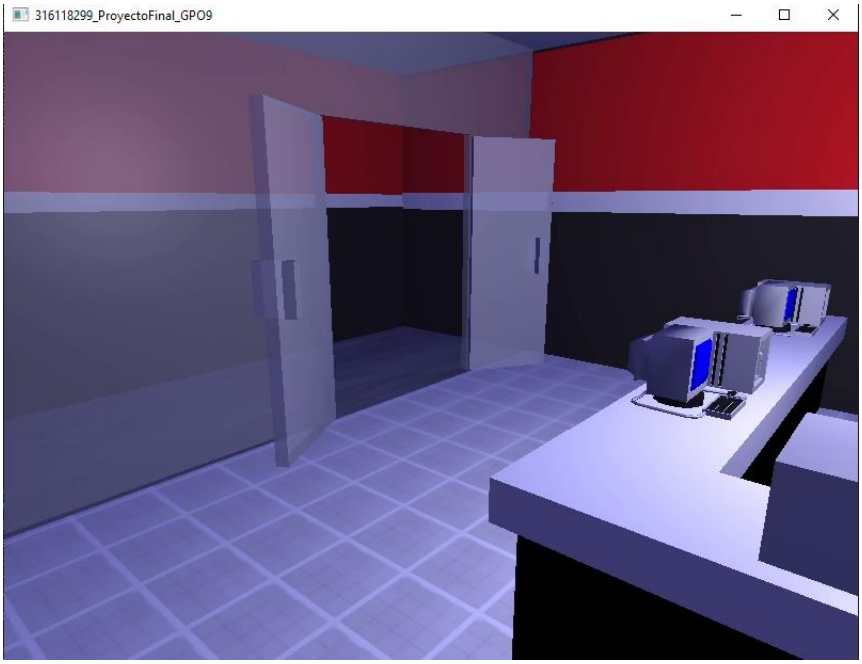
En la función principal, el movimiento de la puerta contempla una traslación para acomodar el modelo en la posición deseada, y una rotación en Y para cada puerta. El sentido de la rotación se invierte en la puerta izquierda para utilizar la misma variable de rotación y reducir la carga computacional del programa.

```
//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);
```

Estado 1 animación: Puertas cerradas.

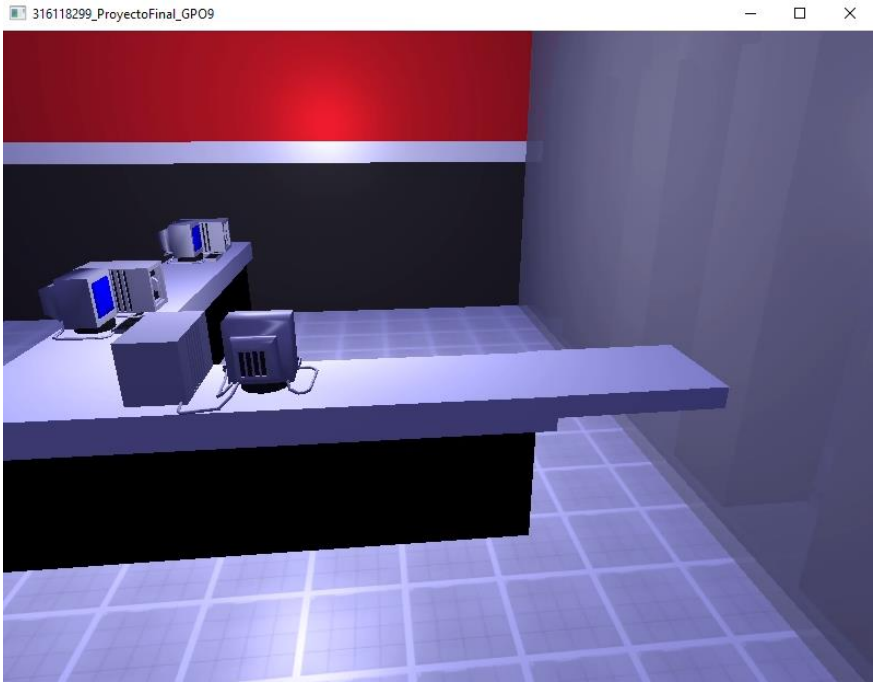


Estado 2 animación: Puertas abiertas.



Animación sencilla 2: Puerta recepción

Para realizar el movimiento de la puerta de la recepción, se realiza un proceso similar a las puertas de la sección Animación sencilla 1: Puertas. Utilizamos dos banderas booleanas para determinar si debe de moverse, y la dirección de la rotación, y se modifica directamente esta rotación en la transformación del modelo.

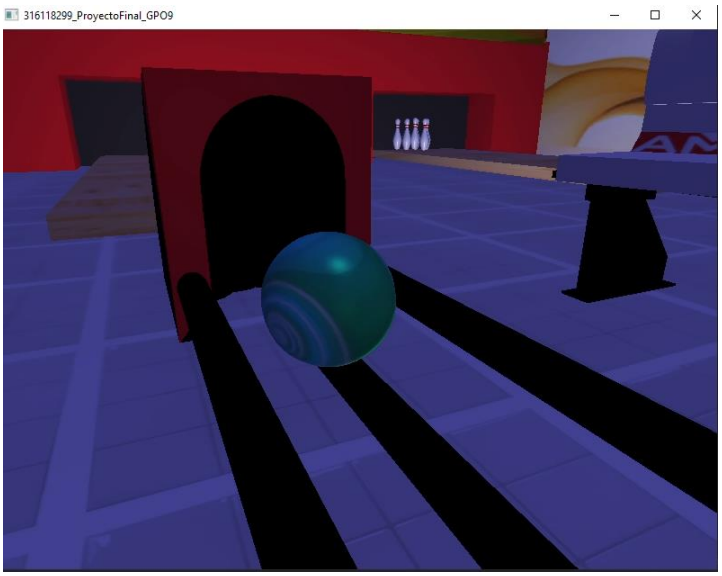


```
//Puerta recepcion - Animacion sencilla 2
view = camera.GetViewMatrix();
model = modelPos;
model = glm::translate(model, glm::vec3(15.8f, -3.9f, -15.5f));
model = glm::rotate(model, glm::radians(recRot), glm::vec3(0.0f, 0.0f, 1));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));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

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

Animación sencilla 3: Rotación bola bolos

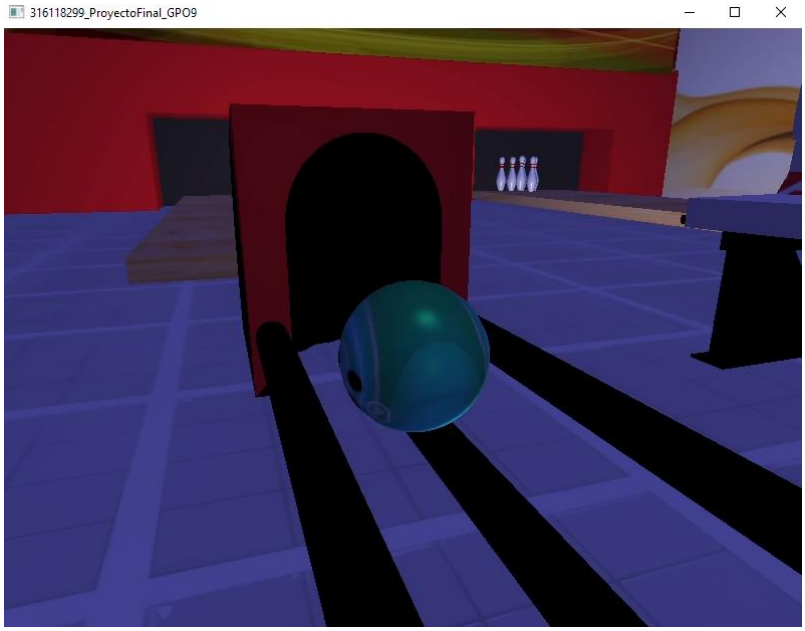
Debido a la forma esférica de la bola de boliche, al momento de rodar, raramente lo hará sobre un solo eje, debido al suelo liso del carril, o a la posición inicial que tiene en el colector. Debido a este efecto, se implementó una función rand() para determinar el incremento de la rotación y la dirección de rotación en cada uno de los ejes.



Como este objeto es parte de la animación por keyframes, usamos esa oportunidad para configurar el aumento de la rotación en cada uno de los ejes. En la función de animación, el siguiente fragmento de código nos permite obtener la velocidad de rotación en cada eje utilizando números pseudoaleatorios. Como se mencionó anteriormente, el movimiento de una bola de bolos rara vez concierne a un único eje.

```
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;
}
```

La variable booleana isBallMoving nos permite detener o iniciar el movimiento de rotación de la bola. Debido a cómo se configuró la animación por keyframes, la cuál se detallará en secciones posteriores en Animación compleja 1: Juego de bolos, no siempre se activa el movimiento de la bola, pues existen ciertos frames en los cuáles la bola se mantiene estática (cuando se levanta y se acomoda en el carril), entonces podemos desactivarla usando esta bandera booleana.



En la función DoMovement(), simplemente necesitamos evaluar el valor actual de la variable booleana.

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

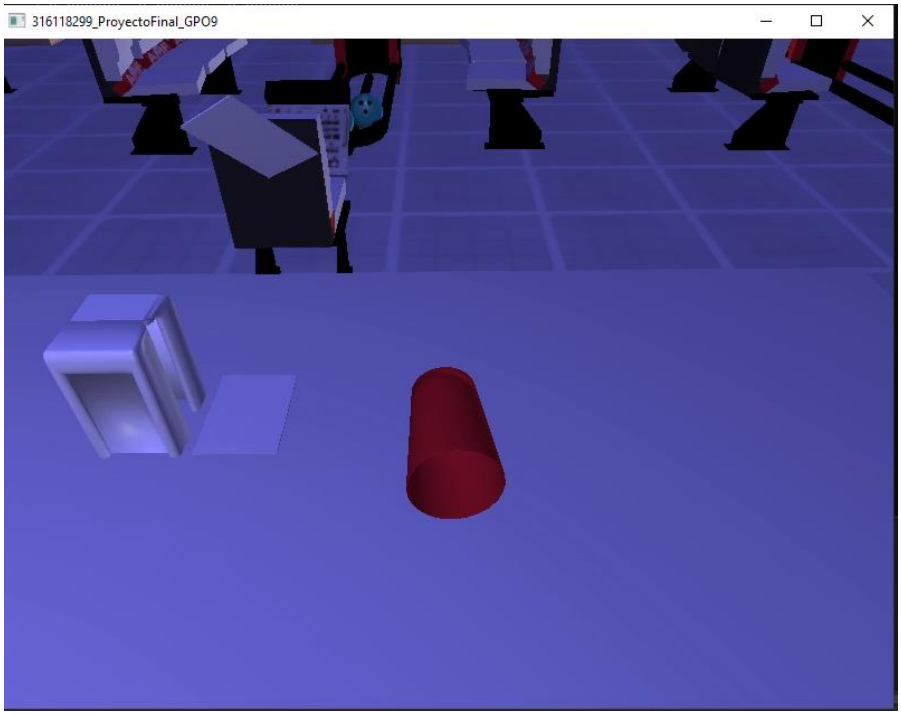
Para lograr el efecto de rotación, basta solamente modificar la matriz de modelo con el respectivo valor de rotación en cada uno de los ejes.

```
//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);
```

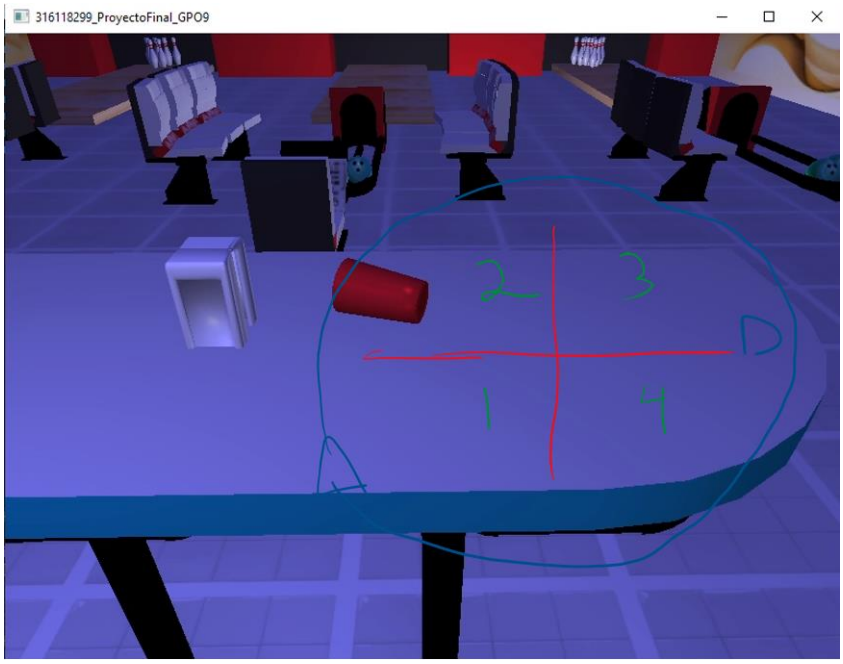
Animación sencilla 4: Vaso de refresco vacío

Para desarrollar esta animación, se asume que existe una corriente de aire, que viene del objeto del aire acondicionado en la pared frontal del edificio.

Esta corriente de aire afecta tanto al vaso de refresco que esta vacío, como a la pila de servilletas que está sobre la mesa.



Si observamos la dirección de la corriente de viento, nos damos cuenta de que podemos dividir el movimiento del vaso en 4 secciones o cuadrantes, dependiendo del impacto que tiene el flujo del aire en su velocidad de movimiento.



En el cuadrante 1, el flujo de aire es directo y golpea directamente en el interior del vaso. Por lo tanto, la velocidad del movimiento debe de ser mayor. En el cuadrante 2, el flujo sigue siendo directo, pero hay mayor distancia entre el vaso y el AC. En el cuadrante 3, ya no hay flujo de aire, pero la velocidad que alcanzó en cuadrante 2 y 3 deben de tener impacto en la velocidad actual, aunque disminuye. Finalmente, en el cuadrante 4 la velocidad es mínima, hasta que el flujo vuelve a ser directo con el interior del vaso en el cuadrante 1.

Animación sencilla 5: Silla

El procedimiento es el mismo que en Animación sencilla 1: Puertas, y en Animación sencilla 2: Puerta recepción, pero en lugar de rotar el objeto, simplemente se traslada sobre el eje Z.

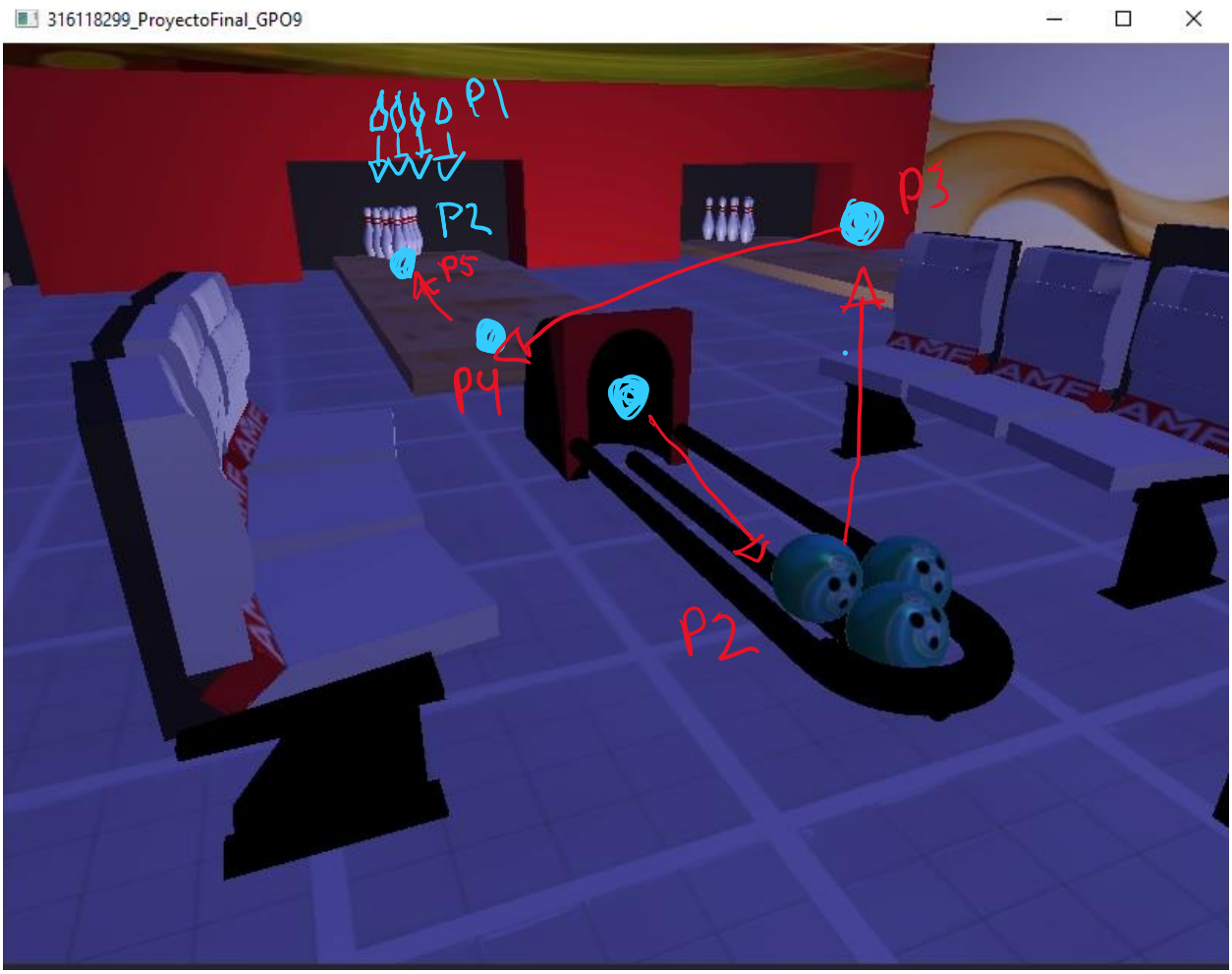
Estado de animación 1

Estado de animación 2

Animaciones complejas

Animación compleja 1: Juego de bolos

Para animar el juego de bolos, se utilizará la animación por keyframes. Para ello necesitamos definir una serie de puntos que definen las posiciones más significativas previo a la interpolación.



Para lograr una animación en bucle que simbolice el movimiento de los objetos en el juego de los bolos, comenzamos con la bola dentro del colector, como si fuera la primera vez que se retorna del carril. Los pinos se encuentran en el acomodador (no visibles). Llamamos este el cuadro inicial, o P1.

Bola:

P1: (0.75, 11.725, -13)

Bolos:

P1: (0, 10, 0)

Posteriormente, el cuadro P2 es cuando la bola de bolos llega al final del colector y se acomoda junto con el resto de las bolas de bolos. Los bolos descienden desde el acomodador y se colocan en la posición del juego.

Bola:

P2: (0.4, 11.725, -13)

Bolos:

P2: (0, 0, 0)

En el cuadro P4, el único objeto en movimiento es la bola, que se acomoda al inicio del carril. Para llegar a esta posición, requerimos de un cuadro adicional, que corresponde a P3, para que el movimiento no involucre atravesar directamente al colector. Elevamos la bola, como si una persona la levantara, por encima del colector. Para el siguiente cuadro, acomodamos la bola al inicio del carril.

Bola:

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

P4: (0.75, 11.725, -19)

En el cuadro P5, recorremos la bola de bolos hasta el punto donde inician los bolos, justo antes de impactarlos.

Bola:

P5: (0.75, 11.725, -34)

Para el cuadro P6, queremos que los bolos giren como si hubiesen sido impactados por la bola, mientras que la bola continua su trayectoria hasta el final del carril y desaparece tras la textura. La escobilla comienza a descender.

Bola:

P6: (0.75, 11.725, -40)

Bolos:

P6: (0, -0.5, 0)

Rotación:

P6: 90°

Escobilla:

P6: (0, -4, 0)

En el cuadro P7, la escobilla se mueve hacia la parte trasera junto con los bolos, como si los limpiara del carril.

Bolos:

P7: (0, 0, -10)

Escobilla:

P7: (0, -4, -5)

El cuadro P8 corresponde a levantar la escobilla

P6: (0, 0, -5)

Finalmente, en el cuadro P9 reiniciamos los valores iniciales y comienza nuevamente la animación. Este cuadro se agregó debido a que los objetos parecían hacer “pop-in”, es decir, que al momento de reiniciar la animación, se transportaban inmediatamente a la posición de P1, por lo tanto se busca una animación o transición más suave.

Los efectos adicionales que se consiguen en esta animación son los que involucran la rotación de la bola de boliche, detallado en Animación sencilla 3: Rotación bola bolos. Como queremos que la animación se detenga entre P3 y P4 (que es cuando se levanta la bola previa a un lanzamiento), podemos apagar el movimiento utilizando la bandera booleana del movimiento de la bola durante los frames involucrados. Esto lo realizamos en la función animacion(), justamente en la lógica que incrementa la variable playIndex.

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

Pero queremos iniciar la rotación justamente en el momento en que la bola sale del colector, por lo que añadimos el código visto en Animación sencilla 3: Rotación bola bolos. En la función KeyCallback añadimos un cálculo de las rotaciones de los ejes de la bola antes de llamar a la animación por keyframes.

```
if (keys[GLFW_KEY_B])
{
    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;
    }
    if (play == false && (FrameIndex > 1))
    {
        resetElements();
        //First Interpolation
        interpolation();

        play = true;
        playIndex = 0;
        i_curr_steps = 0;
    }
    else
    {
        play = false;
    }

}
```

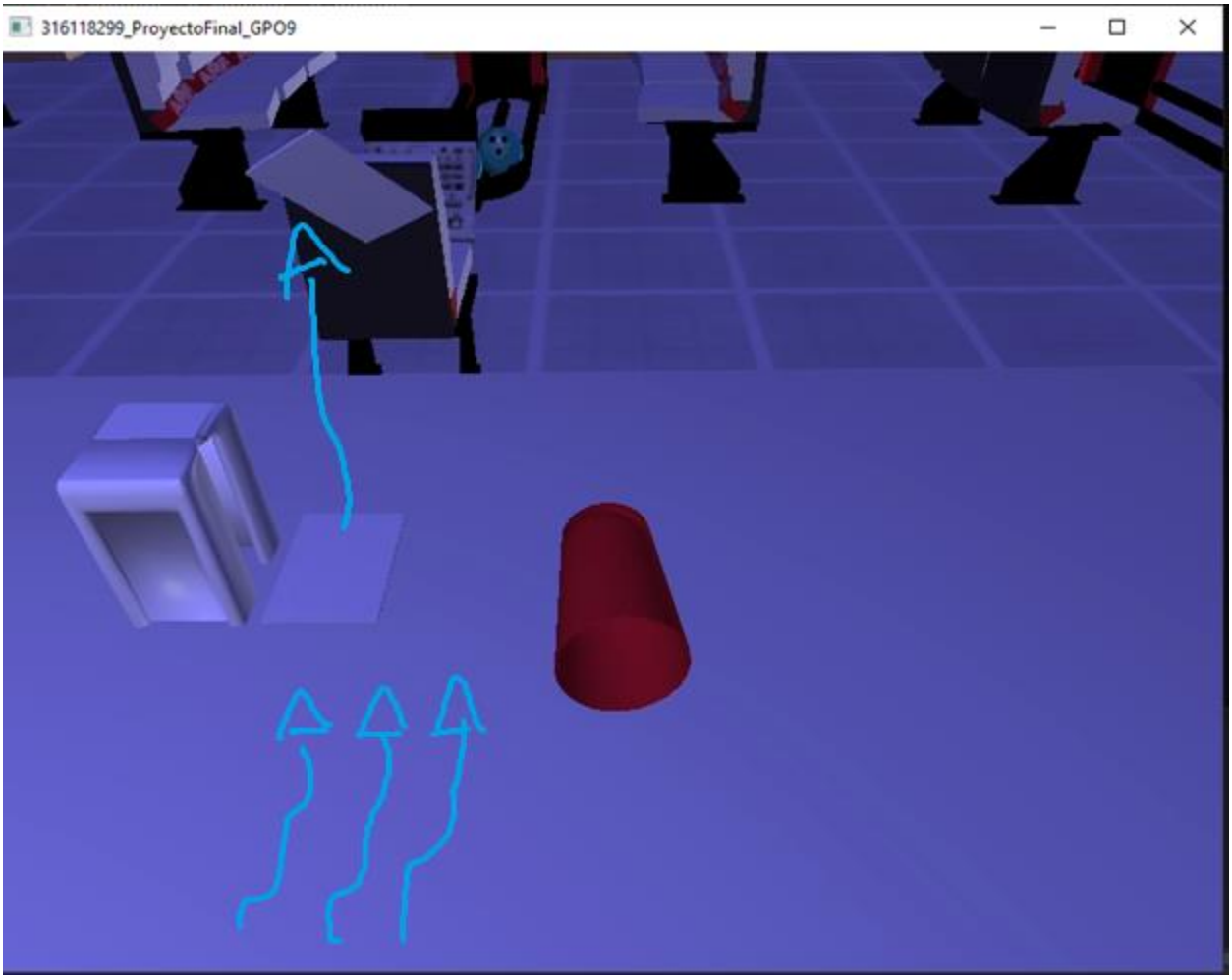
Y en la propia función de animación(), como queremos que el movimiento sea distinto en cada uno de los frames (no tiene sentido que la bola gire de la misma forma a la velocidad del colector que cuando se le aplica la fuerza del jugador para derribar los pinos), basta con que volvamos a calcular la rotación en cada eje con nuestras funciones de números pseudoaleatorios.

```
        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;
        }
    }
}
```

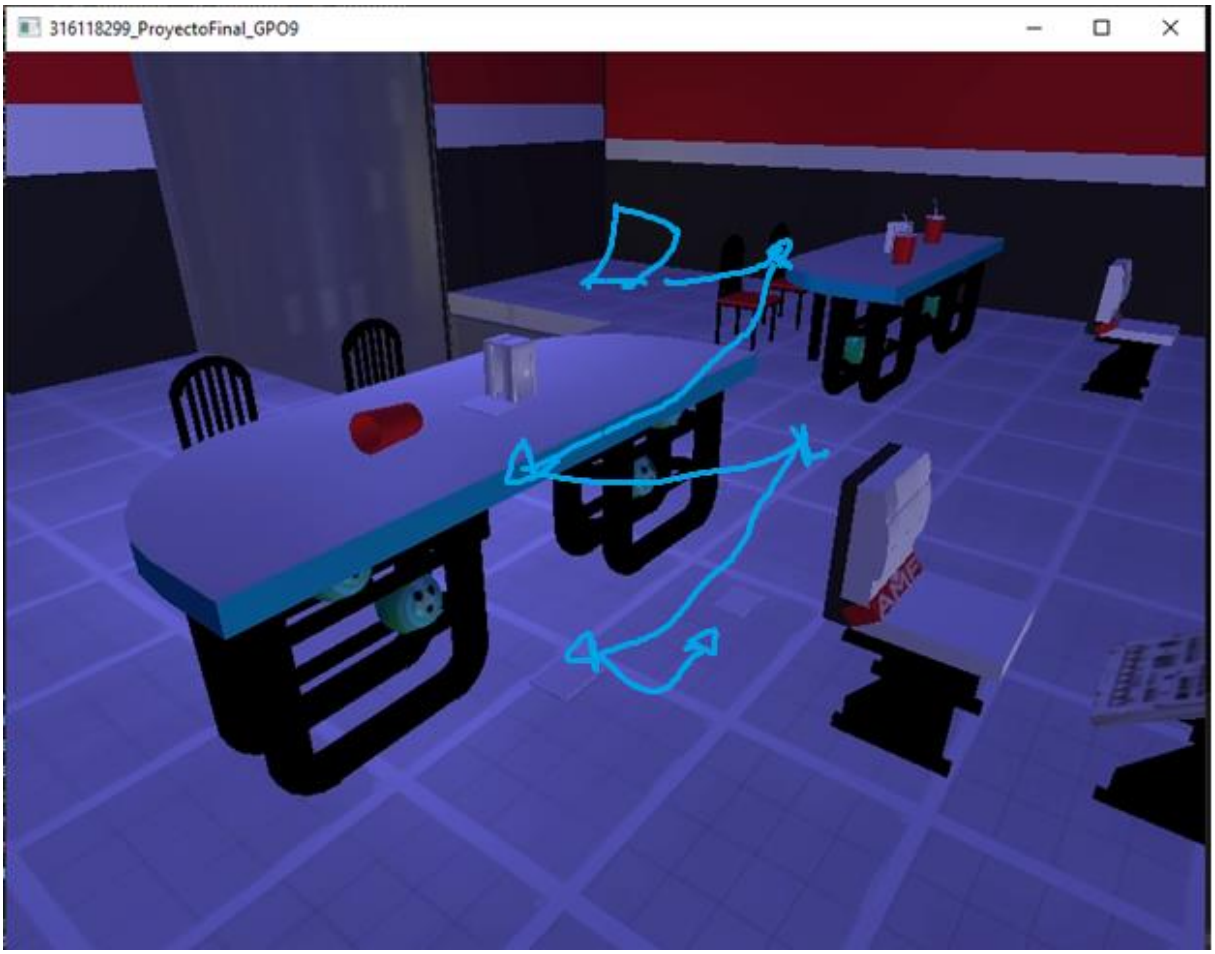
Animación compleja 2: Pila de servilletas

Tal como se mencionó en el apartado de Animación sencilla 4: Vaso de refresco vacío, la unidad de aire acondicionado que se encuentra en la pared emite un flujo de aire que actúa sobre los objetos livianos que se encuentran sobre la mesa central.

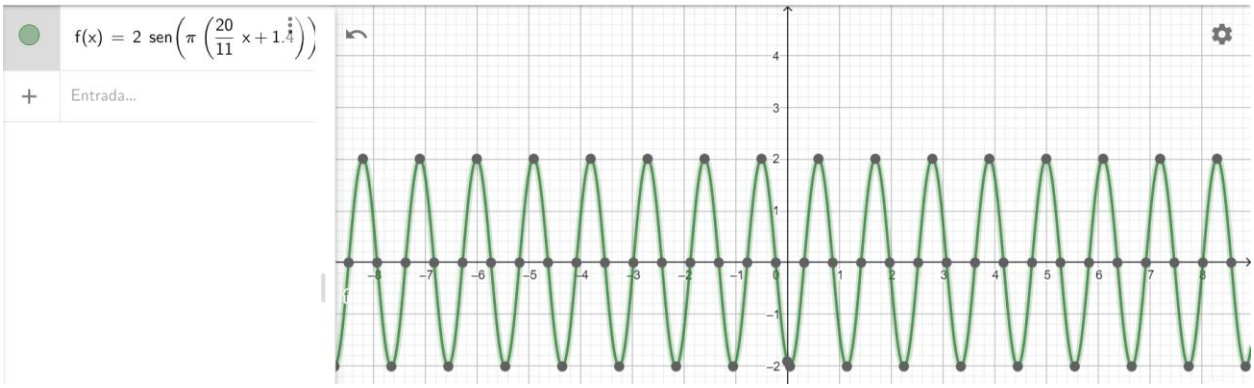
En este diagrama se ejemplifica el efecto que tendría este flujo de aire sobre la pila de servilletas que se encuentran al lado del servilletero y del vaso.



Las servilletas vuelan debido a la acción del aire, y más adelante descienden lentamente mientras oscilan, debido a la resistencia del pliego de papel al aire. Para implementar esta animación, se utilizó una máquina de estados.



Para el efecto de oscilación, utilizaremos las funciones trigonométricas. Como nuestra transformación comienza en 0 (origen), usamos la función seno.



Determinamos el valor de amplitud para obtener el rango de movimiento en el eje X. posteriormente, multiplicamos el argumento por π para normalizar la oscilación, es decir, que los tiempos discretos que muestreamos esté en el dominio del tiempo. Finalmente, buscamos un valor del periodo que permita repetir el periodo las veces necesarias para obtener el efecto deseado.

```
//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;
        }
    }
}
```

Iluminación:

Luz direccional

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.

Luces puntuales

Luz puntual 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.

```
// 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);
```

Luz puntual 2:

La segunda luz puntual es la que se encuentra en la parte superior, en el segundo piso en el área de boliche.

```
// 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);
```

Nuevamente, se eligió una luz con tintes azules y con una intensidad de 65 unidades que alumbra las mesas, en las que se consumen los alimentos que vende el establecimiento. Esta iluminación es la necesaria para que no se vea afectado el área de carriles para boliche, que cuentan con un estilo de iluminación más tenue, remanente de la onda neón de los 80s.

Luz puntual 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.

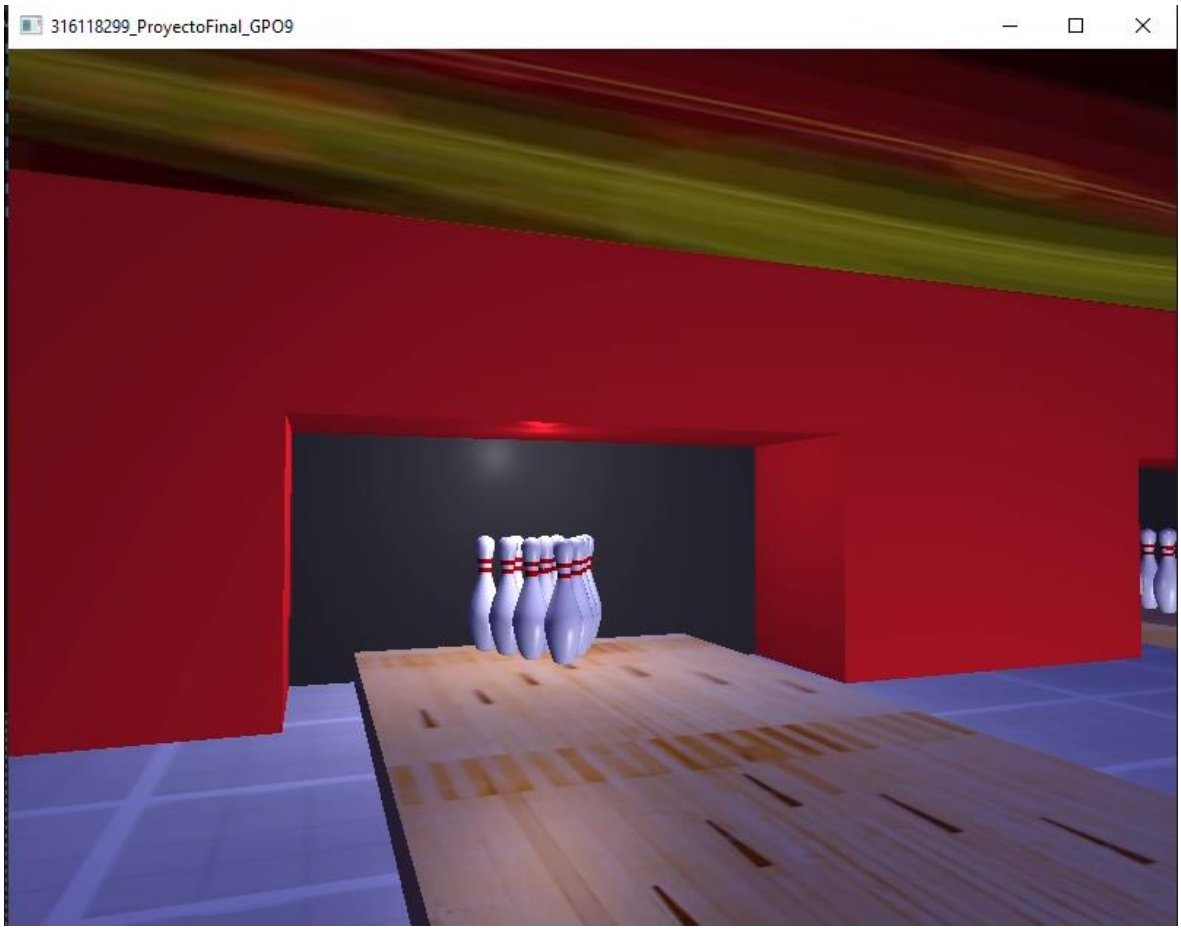
Luz puntual 4:

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.

```
// 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);
```

Y la activación se muestra en la función de animacion()

```
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);
```



Conclusión acerca del proyecto

Al realizar el presente trabajo, es fácil darse cuenta de que el campo de la computación gráfica es uno de alta complejidad, y requiere de un gran trasfondo matemático, computacional, y también toma inspiración en las demás ciencias, sin mencionar la imaginación, el pensamiento creativo y la memoria anecdótica del equipo de desarrollo. Además, todo conocimiento adquirido en este campo es incremental, es decir, que sobre las bases vamos añadiendo conocimiento para realizar labores más complejas. El resultado final es quizás la parte que los usuarios sin conocimientos previos de la computación encuentran más interesante, pues nos permite un alto nivel de interacción y de visualización por parte del usuario. No es un campo sencillo, pero los resultados obtenidos son de gran valor y utilidad en una gran cantidad de profesiones, para fines que van desde el marketing hasta la simulación, la medicina y la educación. Por lo tanto, se considera que se lograron los objetivos propuestos en el plan de estudios para el Laboratorio de Computación Gráfica E Interacción Humano-Computadora, y que los conocimientos adquiridos son de gran valor como las bases sobre las cuáles podemos generar y construir nuevo conocimiento para proyectos de alto nivel.