Universidad nacional autónoma de México.

Facultad de ingeniería.

Proyecto Final.

Materia: Laboratorio. de computación grafica e interacción humano-computadora.

Profesor: Ing. Luis Sergio Valencia Castro.

Integrantes:

- Maceda Patricio Fernando.
- Reyes Avila David.
- Salinas Romero Daniel.
- Vaquero Barajas Alexis.

Semestre: 2022-2.

Fecha de entrega: 12 de abril de 2022.

Proyecto final de laboratorio.

Contenido.

Objetivo.

Los estudiantes realizarán el proyecto final del laboratorio de Computación gráfica e interacción humano-computadora, en el cual aplicará los conceptos vistos a lo largo del curso. El equipo modelará un conjunto habitacional real de manera que este se aproxime a la propuesta de proyecto presentada.

Requerimientos.

- El alumno deberá de realizar un ambiente virtual, el cual representará un área residencial, formado por edificios, casas, áreas verdes y pasos vehiculares. En estos espacios virtuales se deberán colocar vehículos, personas, mascotas, etc.

- El alumno tendrá que utilizar la técnica de modelado geométrico, modelado jerárquico y texturizado para construir los elementos con base en primitivas o con modelos tridimensionales importados según les convenga.

- Se deberá de colocar, al menos, cinco animaciones complejas (con más de 5 estados) utilizando las técnicas de animación que se consideren más pertinentes. Queda prohibido ocupar las animaciones creadas durante las sesiones de laboratorio.

- Los elementos del escenario deberán de contar con texturas aplicadas correctamente.

- Dentro del escenario se deberá poder recorrer mediante el uso correcto de la cámara.

- Incorporará una biblioteca de audio para agregar música de fondo, por lo que el alumno deberá investigar una biblioteca compatible con OpenGL.

- El alumno puede agregar elementos para formar un escenario más grande y complejo. Se otorgarán puntos extra dependiendo de la originalidad de los elementos.

- Se deberá entregar una bitácora con el trabajo realizado y la división de trabajo correspondiente.

Introducción.

Para este proyecto se escogió modelar la unidad habitacional Villa Olímpica, ubicada en la alcaldía Tlalpan de la Ciudad de México, en específico en el predio de nueve hectáreas entre la Avenida Insurgentes Sur y la Avenida Periférico, muy cerca de Ciudad Universitaria.

> El complejo fue construido por el Banco Nacional de Obras y Servicios Públicos para alojar a los Atletas de la Olimpiada de México, consta de veintinueve edificios que reúnen un total de 5,044 habitaciones y 2,572 baños en 904 departamentos, del total de los inmuebles, de ellos 24 se destinaron para los competidores varones y 3 para las mujeres atletas, los restantes dos fueron empleados para la prensa (Edificios de México, (s. f.)).

Se escogió este lugar en principio porque un compañero del equipo vive ahí, por lo cual se facilitaba el modelado de los objetos que se colocaran en el proyecto. Aprovechando que los edificios tienen la misma forma, sólo habría falta el diseño de un edificio y duplicarlo las veces que se desee, además, este lugar nos permite la colocación de diversos elementos que nos permite cubrir los requerimientos impuestos por el profesor. Cabe resaltar que nos dimos la libertad de modificar algunos objetos de manera que pudiera concordar con nuestra visión, cómo reducir el número de edificios, la colocación de una tienda OXXO y una cancha genérica.

Finalmente mostraremos el diseño con la disposición de los elementos que entregamos en la propuesta del proyecto.

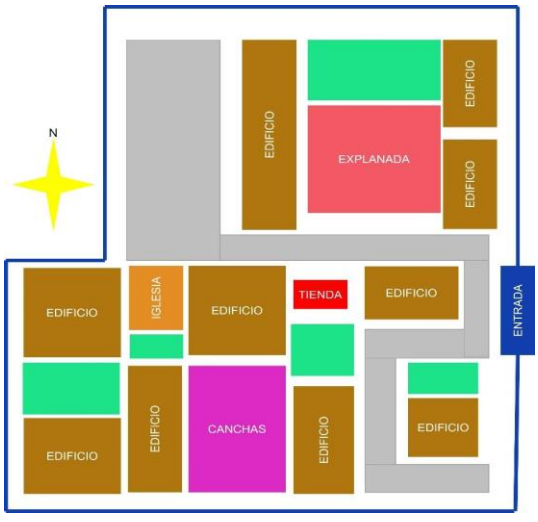


Figura 1. Mapa de la propuesta.

Desarrollo.

Información de los modelos.

A continuación, mostraremos los datos de descarga de los modelos obtenidos de internet y algunos detalles de los modelos realizados por nosotros.

Descargadas de internet:

Modelo: Hombre en chaqueta de pie y Rigged 3D Modelo.

Animación. Persona que conduce el triciclo.

Sitio de internet: open3dmodel.com

El modelo cuenta con una licencia gratis para uso personal para poder ser utilizado.

Figura 2. Hombre de pie.

Modelo: Man 3D Model.

Animación. Persona que conduce una bicicleta.

Sitio de internet: archive3d.net

Dentro de la página de descarga no se encontró algún tipo de restricción o licencia de uso para el modelo, se agregaron las animaciones para las ruedas, pedal y el hombre pedaleando.

Figura 3. Hombre en bicicleta.

Modelo: Pete

Animación. Hombre que camina hacia el oxxo (Left Strafe Walking).

Sitio de internet: Mixamo.

Figura 4. Hombre caminando de lado.

Personaje: Samoyedo Perro Modelo 3D

Animación. Perro que camina por la unidad.

Sitio de internet: open3dmodel.com

Figura 5. Perro.

Personaje: Shannon.

Animación: Persona corriendo al estilo del Fútbol.

Sitio de internet: Mixamo.

Figura 6. Deportista.

Modelo: Sporty Granny

Animación. Persona caminando por la unidad (Female Walk)

Sitio de internet: Mixamo.

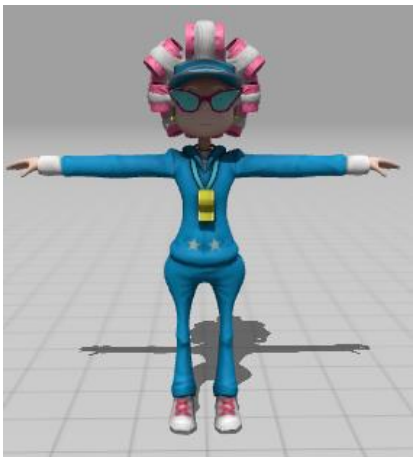


Figura 7. Mujer deportista.

Modelo: Triciclo de reparto 2.

Animación. Un triciclo que simula un puesto de tamales.

Sitio de internet: 3dwarehouse.sketchup.com.

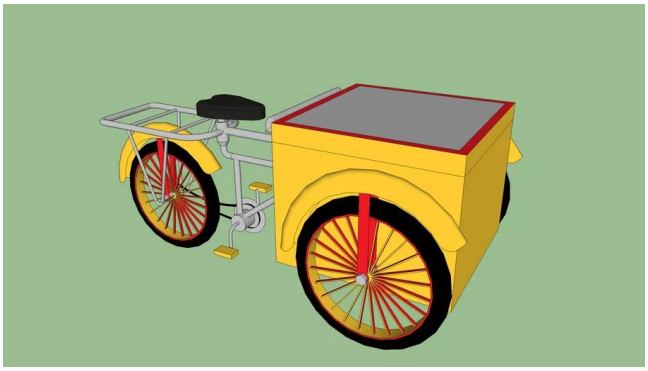


Figura 8. Triciclo.

Al modelo se le agregó una textura en la cara frontal para acabar con su modelo. La textura utilizada fue "un mundo de tamal" obtenida de la página "unmundodetamal.wordpress.com".

Figura 9. Imagen de un mundo de tamal.

Modelo: Vintage Vw Volkswagen Beetle Car Modelo 3D.

Animación. Carro que simula conducirse por la unidad.

Sitio de internet: Open3DModel.

Se decidió cambiar el color al modelo original debido a un problema con las texturas. Las llantas se exportaron por separado para que se pueda animar de manera correcta su rotación.

Figura 10. Bocho.

Realizadas por el equipo:

Modelo: Cancha.

Aplicación de diseño: Blender.

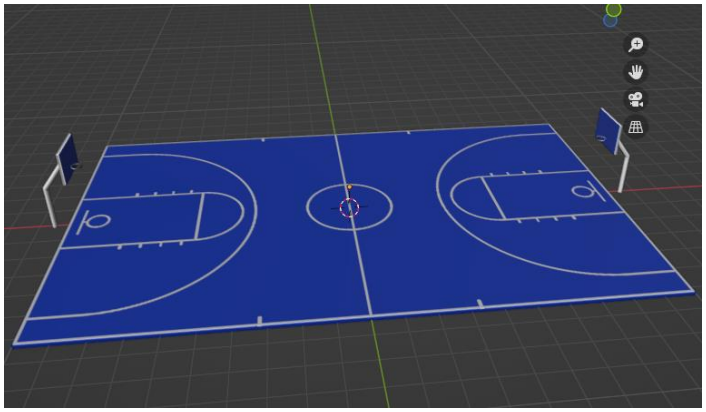


Figura 11. Cancha.

Modelo: Edificio.

Aplicación de diseño: Blender.

El modelo corresponde a los edificios ubicados en la unidad. Para modelarlo se utilizaron planos y cubos. Con sencillas transformaciones y extrusiones se pudo lograr el acabado similar a nuestras referencias originales.

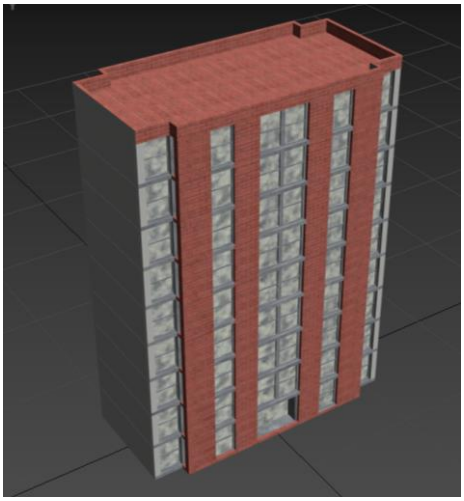


Figura 12. Edificio.

Modelo: Entrada.

Aplicación de diseño: Autodesk 3ds Max.

Este modelo corresponde a la entrada a la unidad. Solo se utilizaron figuras primitivas y un "TexPlus" para los números que indican la dirección.

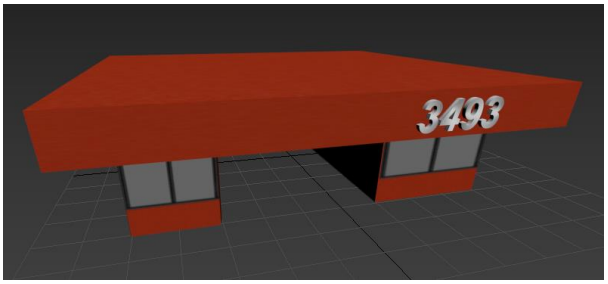


Figura 13. Entrada.

Fachada real de la entrada.

Como se ve en la imagen de referencia colocada en la propuesta, aunque para el modelo se ocupó una textura de un color más oscuro.

Modelo: Iglesia.

Aplicación de diseño: Blender.

El modelo de la iglesia se modeló usando también figuras simples como cubos a los cuales se les aplicaron transformaciones de escala y traslación. Para el vitral únicamente se colocó una textura sacada de una fotografía tomada a la iglesia física.

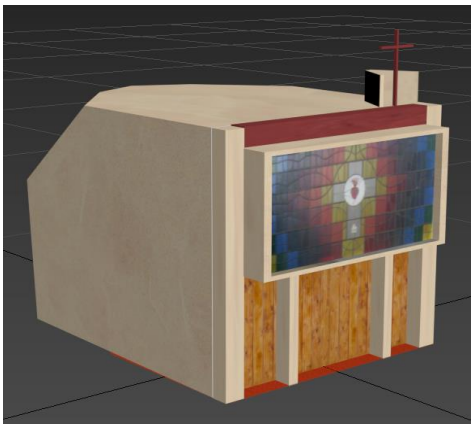


Figura 14. Iglesia.

Modelo: OXXO.

Aplicación de diseño: Autodesk 3ds Max.

El modelo corresponde a una sucursal de las tiendas Oxxo dentro de la unidad. Para el modelado se utilizaron planos, cajas y cilindros además de utilizar sus colores característicos para la textura.

Figura 15. OXXO.

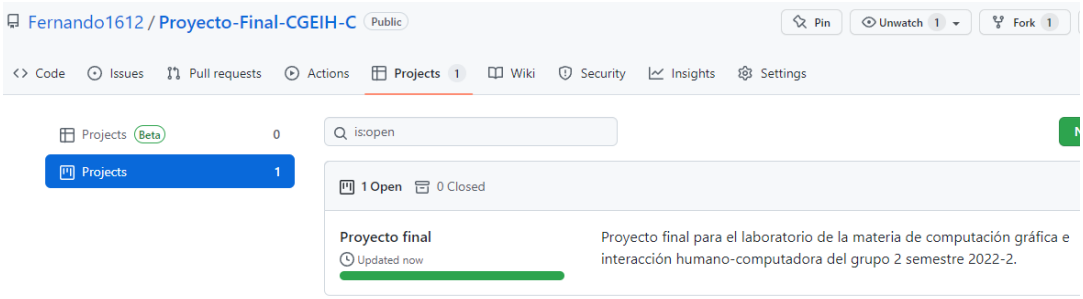
Para la fachada se utilizó la imagen “Fachada de Tienda OXXO” descargada de la página “flickr.com” y subida por la cuenta “FEMSA_Corporativo”.

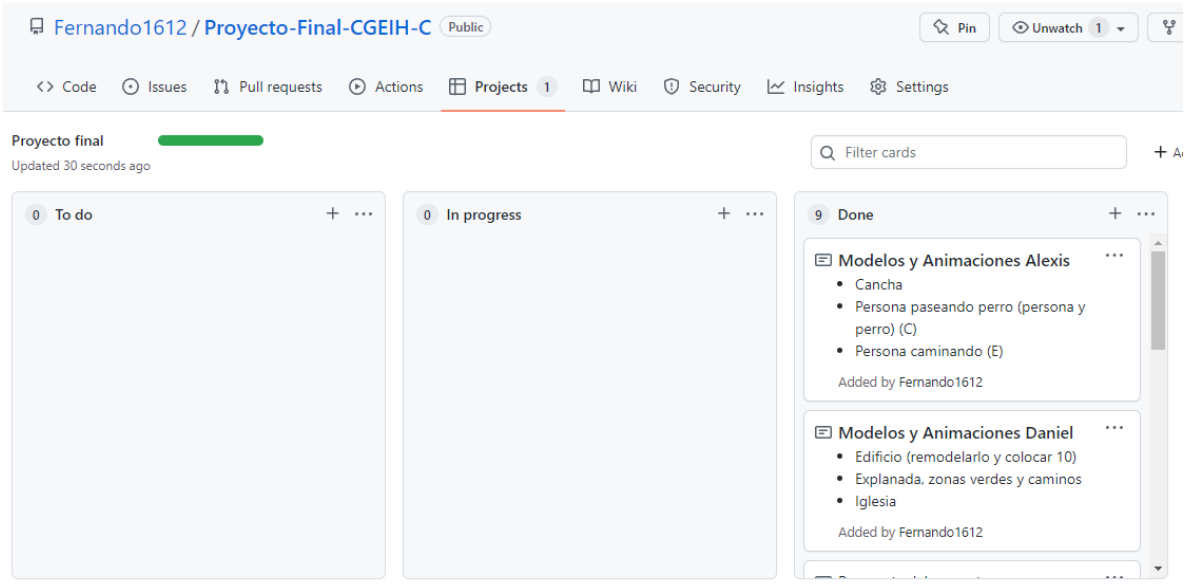
Fachada Oxxo.

Cronograma de actividades.

Actividades	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6
Propuesta del proyecto.						
Tienda (Oxxo). Triciclo-Persona. Persona en bicicleta.						
Entrada. Automóviles. Persona-Cancha.						
Edificio (10). Explanada, caminos y zona verde. Iglesia.						
Cancha. Persona-Perro. Persona caminando.						
Manual de ejecución.						
Manual de usuario.						
Créditos.						
Documentación						

Tablero kanban.





Información de las animaciones.

Animación de hombre en bici:

Para generar la animación se crearon tres variables globales:

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

La animación está dividida en varios objetos:

- Hombre pedaleando
- Cuadro de la bicicleta
- Pedales
- Ruedas

La animación del hombre pedaleando fue creado con 3ds Max utilizando la herramienta de Auto key. al momento de dibujar al hombre en el programa le asignamos las variables de movimiento en x y z de la bicicleta y de rotación para que al momento que la bicicleta se mueva, el hombre también lo haga.

```
//-------------------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);
```

Dibujamos la bicicleta con sus valores iniciales y creamos un modelo temporal. Para la animación de los pedales y las ruedas agregamos una rotación del eje x:

```
// ------------------------------------------------------------------------------------------------------------------------
// 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
```

Para la animación de la bici se asignó una ruta dentro del mapa en donde se utilizarán 6 estados del 0 al 5 utilizando la estructura de control Switch/Case, la bicicleta va a cambiar su posición en el eje x y el eje z. Esta animación se repite infinitamente ya que al llegar al estado 5 la bicicleta se encuentra en las coordenadas de origen y regresa al estado 0 tampoco se necesita presionar alguna tecla para que comience, la animación inicia junto con el programa.

```
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;
}
```

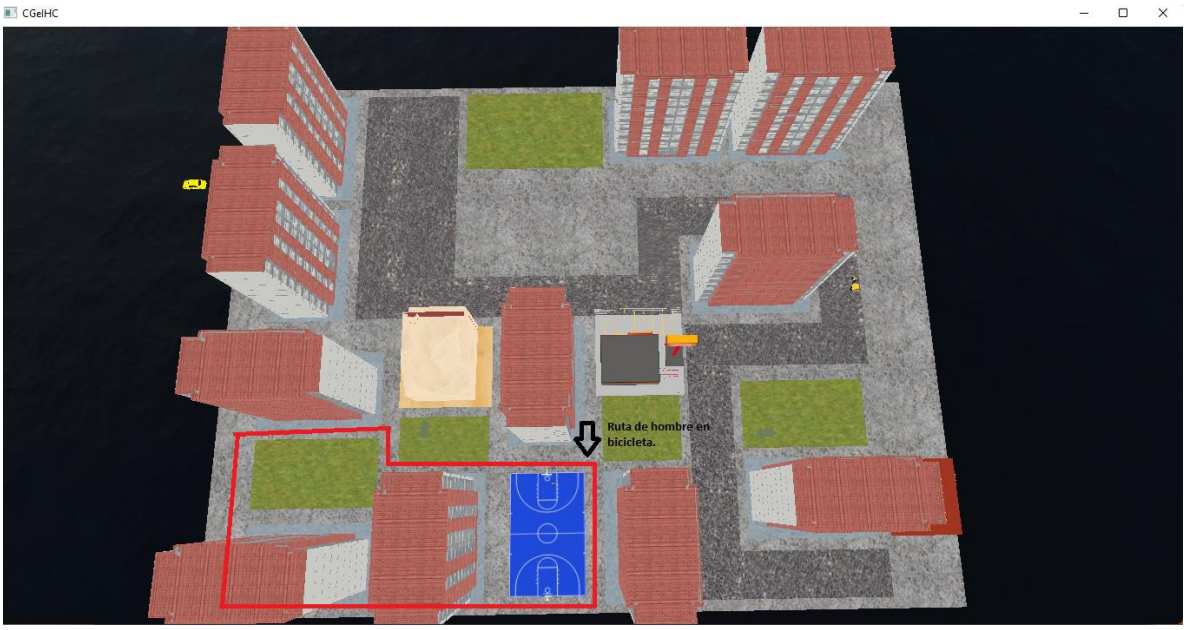


Figura 16. Ruta de animación de bicicleta.

Animación de hombre vendiendo tamales :

Para generar la animación se crearon tres variables globales:

```
float movTrici_z = -500.0f, // posición inicial en Z
      movTrici_x = -175.0f, // posición inicial en x
      orientaTrici = 0.0f;  // orinetación inical
```

La animación está dividida en varios objetos:

- Hombre pedaleando
- Cuadro del triciclo
- Pedales
- Ruedas

La animación del hombre pedaleando fue creado con 3ds Max utilizando la herramienta de Auto key. al momento de dibujar al hombre en el programa le asignamos las variables de movimiento en x y z del triciclo y de rotación para que al momento que la bicicleta se mueva, el hombre también lo haga. Al momento de cargar el modelo tenía una rotación incorrecta por lo que se hicieron dos rotaciones, una al inicio para acomodarlo en el triciclo y la segunda que va ir cambiando conforme el triciclo avance.

```
//-------------------Triciclo---------------------------------
model = glm::translate(glm::mat4(1.0f), glm::vec3(movTrici_x, 1.0f, movTrici_z));
model = glm::scale(model, glm::vec3(0.09));
tmp = model = glm::rotate(model, glm::radians(90.0f), glm::vec3(-1.0f, 0.0f, 0.0f));
model = glm::rotate(tmp, glm::radians(orientaTrici), glm::vec3(0.0f, 0.0f, 1.0f));
animShader.setMat4("model", model);
manTricycle.Draw(animShader);
```

Dibujamos el triciclo con sus valores iniciales y creamos un modelo temporal. Para la animación de los pedales y las ruedas agregamos una rotación del eje x:

```
// ------------------------------------------------------------------------------------------
// Persona en Triciclo
// ------------------------------------------------------------------------------------------
tmp = model = glm::translate(glm::mat4(1.0f), glm::vec3(movTrici_x, 1.0f, movTrici_z));
tmp = model = glm::rotate(model, glm::radians(orientaTrici), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.09f));
staticShader.setMat4("model", model);
triciclo.Draw(staticShader);

model = glm::translate(tmp, glm::vec3(0.0f, 3.25f, -3.0f));
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);
pedalesTriciclo.Draw(staticShader);

model = glm::translate(tmp, glm::vec3(-0.45f, 4.0f, -8.5f));
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);
ruedaTriciclo.Draw(staticShader); // rueda trasera

model = glm::translate(tmp, glm::vec3(-5.0f, 4.0f, 8.0f));
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);
ruedaTriciclo.Draw(staticShader); // rueda delantera der

model = glm::translate(tmp, glm::vec3(5.0f, 4.0f, 8.0f));
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);
ruedaTriciclo.Draw(staticShader); // rueda delantera izq
```

Para la animación del triciclo se asignó una ruta dentro del mapa en donde se recorre todo el conjunto habitacional utilizando 16 estados del 0 al 15, utilizando la estructura de control Switch/Case para los cambios en el eje x y z. Esta animación se repite infinitamente ya que al llegar al estado 15 el triciclo se encuentra en las coordenadas de origen y se regresa al estado 0, tampoco se necesita presionar alguna tecla para que comience, la animación inicia junto con el programa.

```
switch (estado_trici)
{
case 0:
    if (movTrici_z <= -250.0f){
        orientaTrici = 0.0f;
        movTrici_z += 0.5f;
    }else {
        estado_trici = 1;
    }
    break;
case 1:
    if (movTrici_x <= 150.0f) {
        orientaTrici = 90.0f;
        movTrici_x += 0.5f;
    }
    else {
        estado_trici = 2;
    }
    break;
case 2:
    if (movTrici_z >= -385.0f) {
        movTrici_z -= 0.5f;
        orientaTrici = 180.0f;
    }
    else {
        estado_trici = 3;
    }
    break;
case 3:
    if (movTrici_x <= 490) {
        movTrici_x += 0.5f;
        orientaTrici = 90.0f;
    }
    else {
        estado_trici = 4;
    }
    break;
case 4:
    if (movTrici_z <= -110.0f) {
        movTrici_z += 0.5f;
        orientaTrici = 0.0f;
    }
    else {
        estado_trici = 5;
    }
    break;
case 5:
    if (movTrici_x >= 270.0f) {
        movTrici_x -= 0.5f;
        orientaTrici = 270.0f;
    }
    else {
        estado_trici = 6;
    }
    break;
case 6:
    if (movTrici_z <= 190.0f) {
        movTrici_z += 0.5f;
        orientaTrici = 0.0f;
    }
    else {
        estado_trici = 7;
    }
    break;
case 7:
    if (movTrici_x <= 500.0f) {
        orientaTrici = 90.0f;
        movTrici_x += 0.5f;
    }
    else {
        estado_trici = 8;
    }
    break;
case 8:
    if (movTrici_x >= 270.0f) {
        orientaTrici = 270.0f;
        movTrici_x -= 0.5f;
    }
    else {
        estado_trici = 9;
    }
    break;
case 9:
    if (movTrici_z >= -110.0f) {
        orientaTrici = 180.0f;
        movTrici_z -= 0.5f;
    }
    else {
        estado_trici = 10;
    }
    break;
case 10:
    if (movTrici_x <= 490.0f) {
        orientaTrici = 90.0f;
        movTrici_x += 0.5f;
    }
    else {
        estado_trici = 11;
    }
    break;
case 11:
    if (movTrici_z >= -385.0f) {
        orientaTrici = 180.0f;
        movTrici_z -= 0.5f;
    }
    else {
        estado_trici = 12;
    }
    break;
case 12:
    if (movTrici_x >= 150.0f) {
        orientaTrici = 270.0f;
        movTrici_x -= 0.5f;
    }
    else {
        estado_trici = 13;
    }
    break;
case 13:
    if (movTrici_z <= -250.0f) {
        movTrici_z += 0.5f;
        orientaTrici = 0.0f;
    }else {
        estado_trici = 14;
    }
    break;
case 14:
    if (movTrici_x >= -175.0f) {
        orientaTrici = 270.0f;
        movTrici_x -= 0.5f;
    }
    else {
        estado_trici = 15;
    }
    break;
case 15:
    if (movTrici_z >= -500.0f) {
        movTrici_z -= 0.5f;
        orientaTrici = 180.0f;
    }
    else {
        estado_trici = 0;
    }
    break;
default:
    break;
}
```

Figura 17. Ruta del señor de los tamales.

Animación de hombre haciendo deporte en la cancha:

Para generar la animación se crearon cuatro variables globales:

```
//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
```

La animación está formada por un único objeto, el personaje Shannon con la animación Running, descargada de la página Miaxamo. Al momento de descargar se indicó que la animación debía ser en un solo lugar, para que a base de código se generará el desplazamiento. Para poder dibujar el objeto se debió colocar la variable que le permitirá desplazarse en el eje de las Z, esto tomando ventaja de que no se moverá en los ejes X y Y; en conjunto con una variable que permita su rotación sobre el eje Y para que simule su cambio de dirección.

```
//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);
```

Ocupando un SWITCH se escoge entre los diferentes estados, que representan hasta qué distancia de la cancha debe llegar el objeto, dentro de cada CASE se encuentra una sentencia IF que verifica si el objeto se encuentra avanzando (hacia valores mayores) o regresando (hacia valores menores). En caso de que se cumpla la condición de que llegue hasta cierta distancia, el objeto realizará un giro de 180° y realizará el movimiento siguiente cambiando de estado.

```
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)
        }
    }
```

Para que resulte la animación de que el personaje corra hasta cada marca y regrese hasta el origen una y otra vez, ya que no se condiciono la animación a una entrada del usuario al presionar una tecla.

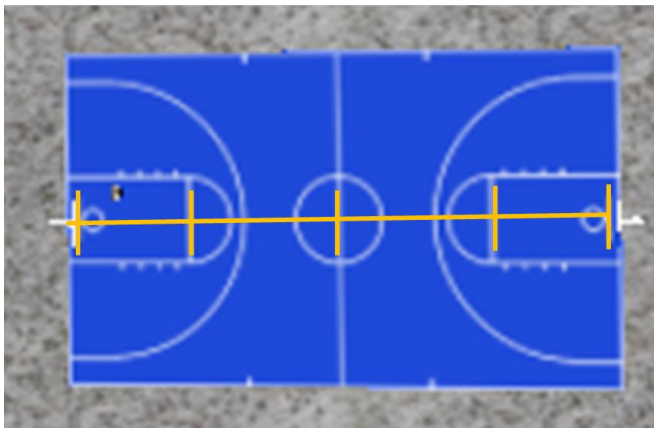


Figura 18. Cancha mostrando distancias que recorre el personaje.

Animación de Carro:

Para generar la animación se crearon cuatro variables globales:

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

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

La animación está formada por cinco objetos, la carrocería y cuatro ruedas, considero que lo más complicado de colocar los objetos fue las proporciones y colocar las ruedas en la posición correcta para que no se vean más grandes de lo que deberían. En la carrocería se colocaron las variables que permiten su movimiento en X y Z y una para realizar el giro sobre Y; para las ruedas solo se ocupa una variable para su giro sobre 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
```

Para que el auto realice sus movimientos se necesita que el usuario presione la tecla SPACE para que avance o se detenga. Para el recorrido se decidió que empiece al lado del objeto de entrada y termine estacionado. Lo primero que se debe hacer es verificar en qué estado se encuentra el auto para que avance y giren las llantas hacia una dirección, después se pregunta si ya llego hasta una posición que seleccionamos para que realice un giro, cambie de estado y siga avanzando; las

posiciones dependen de la distribución del asfalto. Considero que eso fue lo más complicado de la animación fue ver hasta qué punto debe girar y no atraviesa objetos que no deseamos.

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

Animación Hombre que camina:

Para generar la animación se crearon cuatro variables globales:

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

Para esta animación solo se ocupó un objeto descargado de Mixamo. El personaje se colocó debajo de la entrada y con dos variables para su movimiento en X y Z se desplazará hasta llegar al OXXO y con una variable en su rotación girara sobre el eje Y.

```
// 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);
```

Dentro de la función animate, se colocó un IF para que el usuario presione la tecla 1 y así el personaje empiece su recorrido. Posteriormente se verifica en qué estado se

encuentra para que avance en el sentido correcto y al llegar hasta cierta ubicación, éste cambia de orientación y de estado. Cabe aclarar que en el estado cinco, el personaje no avanzará si es que el automóvil va a cruzar por el mismo sitio, esto verificando que el estado del auto sea diferente a dos; a su vez el personaje solo atraviesa el objeto OXXO, no se abren las puertas para que entre.

```
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;
        }
    }
}
```

Animación Persona paseando perro.

Para realizar la animación correspondiente, primero se definió una constante, la cuál indicará la velocidad en la que los personajes además de 5 variables flotantes, donde estas nos ayudaran a trasladar los objetos en el eje X, Z y finalmente rotar estos mismos.

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

Otra variable que también se declara, es la del estado de la animación, la cuál será la que definirá el cambio de estado por medio de un switch-case.

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

Para que los objetos puedan realizar la animación, es importante que donde dibujamos estos mismos, agreguemos las variables correspondientes en la traslación (X, Z), donde estas se sumarán con su posición definida, de manera que la posición inicial no será modificada. De igual manera, para que los objetos puedan realizar una rotación, se agrega la variable correspondiente en esta.

```
// 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);
```

Para realizar los estados de la animación se utiliza la estructura de control Switch-Case, de manera que cuando el resultado esperado se cumpla, pase a otro caso y continúe la animación.

Para cada caso se encuentra un if-else el cuál va a comprobar que el movimiento en determinado eje no supere al que determinó el desarrollador, lo que consecuencia los objetos aumentarán o disminuirán su posición hasta este límite además de rotar los grados establecidos. Finalmente, una vez la condición no se cumpla, el perro aumentará o disminuirá su posición de manera que este logre que el perro esté al lado izquierdo de la persona y se pase al siguiente caso (estado de la animación con un máximo de 5 estados).

```
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;
    }
```

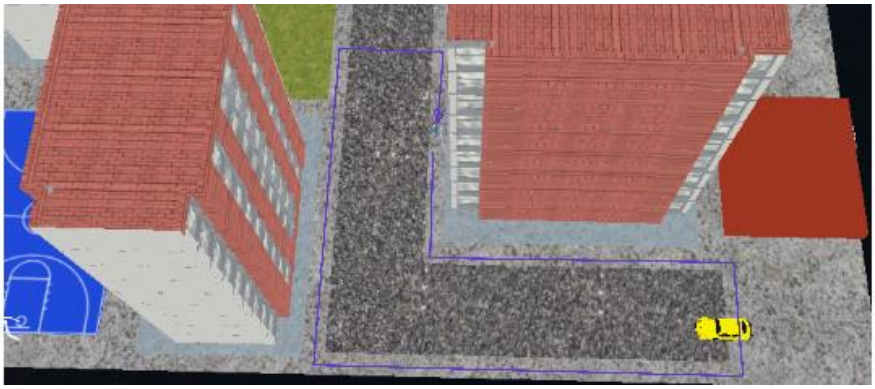


Figura 19. Ruta de la persona paseando un perro.

Estudio técnico para la determinación del costo del proyecto:

Para el costo de este proyecto se realizó el procedimiento aprendido en la materia de Finanzas en la ingeniería en computación ubicada en el séptimo semestre del plan de estudios de la carrera. Ahí se revisó que el costo del proyecto se divide en dos grandes rubros, el costo directo que está en función de la duración total del proyecto y el costo indirecto que depende de todos los gastos asociados al proyecto.

Los cálculos se muestran en las siguientes tablas:

Costo directo	Proyecto final de laboratorio CGeIHC									
	Cantidad	Personal	Rol	No. de proyectos en los que trabaja	Tarifa/h	Disposición	Horas/Mes	Horas/Proyecto	Salario/Mes	Salario/Proyecto
	1	Líder de proyecto	X	5	$700.00	0.2	32	48	$22,400.00	$33,600.00
	3	Desarrollador	Jr.	5	$350.00	0.2	32	48	$11,200.00	$16,800.00
				6		0.16666667	26.6666667	40	$9,333.33	$14,000.00
				7		0.14285714	22.8571429	34.2857143	$8,000.00	$12,000.00
							Total	170.285714		$76,400.00

Costo indirecto	Cantidad	Concepto	Costo dólares unitario	Costo pesos unitario	Costo total por concepto
	1	Modelo de automóvil	$0.00	$0.00	$0.00
	5	Modelo de persona	$0.00	$0.00	$0.00
	1	Modelo de perro	$0.00	$0.00	$0.00
	1	Modelo de bicicleta	$0.00	$0.00	$0.00
	0	Servidor	$0.00	$0.00	$0.00
	0	Sistema de almacenamiento	$0.00	$0.00	$0.00
	0	Licencias	$0.00	$0.00	$0.00
	1	Renta de local/Mes	X	$15,000.00	$4,500.00
	X	Pago de servicios/Mes	X	$10,000.00	$3,000.00
				Total	$7,500.00

Costo directo:

Para el cálculo del costo directo se consideró una jornada laboral de 8 horas cinco días a la semana, lo que resulta a 40 horas laborales de trabajo; teniendo en cuenta que el proyecto duró 6 semanas, equivalentes a 240 horas.

Duración	horas
1 día	8
1 semana	40
1 mes	160
Proyecto (6S)	240

Dado que somos cuatro integrantes en el equipo, se designó a un miembro como líder del proyecto y el resto con el rol de desarrollador Jr.; cabe aclarar que las tarifas reflejadas en las tablas fueron recuperadas de un caso de estudio en la materia de finanzas. A su vez, como todos los integrantes del equipo llevan más de una materia, su disponibilidad no sería al cien por ciento, por lo cual el rubro de "número de proyectos en lo que se participa" es equivalente a "número de materias que está cursando", por lo tanto, la disposición es una simple división del tiempo entre ese valor. Las columnas siguientes son multiplicaciones hasta el cálculo del total de horas hombre trabajadas aproximado a 170, lo que implica un costo directo de $76,400.00.

Costo indirecto:

En el costo indirecto se debe considerar todos los elementos utilizados a lo largo del proyecto, como los modelos descargados de internet, los cuales por fortuna no costaron nada, ahí consideramos el rubro de su precio en dólares americanos ya que en las plataformas de descarga su valor se encuentra en este tipo de moneda.

También se debe considerar todos los gastos implícitos, como es la renta de un local y el pago de los servicios que se necesitan (internet, electricidad, agua, etc.). Estos gastos de la misma manera se deben distribuir entre todos los proyectos que se realizan por los miembros del equipo como se muestra a continuación, cabe aclarar que se multiplicó entre la disposición de un integrante más grande.

$$\$15{,}000.00 \cdot 0.2 = \$4{,}500.00$$

$$\$10{,}000.00 \cdot 0.2 = \$3{,}000.00$$

Lo que resulta en un costo indirecto total de $7,500.00.

Costo total:

El costo total del proyecto corresponde a la suma de ambos valores totales, el del costo directo y el del indirecto, por lo tanto, se concluye que el costo total del proyecto es igual a $83,900.00.

Mapa de navegación:

Proyecto-Final-CGEIH-C/README.md at main · Fernando1612/Proyecto-Final-CGEIH-C (github.com)

Video explicativo:

https://youtu.be/Yrzgm1TxxlU

Explicación individual de actividades (en inglés):

Fernando Maceda Patricio.

Once the activities were distributed with the whole team, my main activities were:

- Store (Oxxo)
- Animated man on bicycle
- Animated man on tricycle with tamales

I started with the store model based on the following image:

Using the 3ds Max program I started creating simple figures, for the parking lot I used a plane and some cylinders, to create the advertisement I used a cylinder and a box adding and to create the store and the logo I used two boxes, with this I started adding the textures, create a UV map template and use the GIMP program to edit the store images and put them in their respective places.

Finally, the result is the following:

The next step was the model of the man on a bicycle, for this I downloaded a model made by Fima on the archive3d.net page (the model doesn't have any use license), I imported it into 3ds Max and it already had all the textures, only I had to add the animation.

First separate the model by objects to have a better control of these, the objects are:

- Man
- Bicycle frame
- Wheels
- Pedals

The bike frame is a static object that only moves in the Z axis, so I didn't do any more modifications, for the wheels I added a rotation in the code to make it feel like it's moving, I did the same for the wheels pedals with this, the animation of the bike moving around the plane was complete.

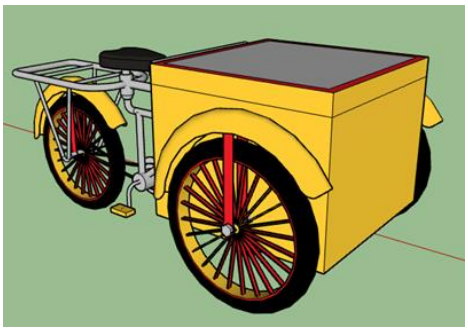
For the animation of the man I had many problems since I only had the skin of the man, when I tried to move it the model was distorted a lot, this is where I discovered the rigging technique, which consists of putting bones on the model and the texture changes of position, with this technique I animated the man pedaling in 3d max using the Auto Key tool, once the animation was done I exported the model with .dae format as it was done in class and finally I just placed the character on the bicycle, the result was the following:

For the last part, I downloaded the model of the tricycle from the page 3dwarehouse.sketchup.com made by Luis Alberto Videira. The model has the General Model License which allows us to use the model for free.

I divided the model into three parts:

- Tricycle frame
- Wheels
- Pedals

Like in the previous animation the tricycle frame is a static object that only moves in the z axis and I added rotations through the code for the pedals and wheels.



Then I added the image "un mundo de tamal" and I put the tricycle as a texture on the front face.

For the man model I downloaded “Hombre en chaqueta de pie y Rigged 3D Modelo” from the open3dmodel.com page, at the moment the user who uploaded the model is not shown, but it has a free license for personal use.

To create the animation I did the same steps as the previous animation with the advantage that this model already had the bones included, using the 3ds max auto key tool I animated the model and exported it to the project, the result of everything was as follows:

With this I finished my main activities.

Extra activities

For extra activities I added the winmm library to play music in our program creating the music function that is activated when the Z key is pressed and using the PlaySound function it plays “TamalesOaxaqueños.wav”.

```
// Pragma para musica
#pragma comment(lib, "winmm.lib")
```

```
void music() {
    if (sound) {
        bool played = PlaySound(L"TamalesOaxaquenos.wav", NULL, SND_LOOP | SND_ASYNC);
        sound = false;
    }
}
```

And I helped with the project documentation and the user manual.

Reyes Avila David:

As soon as que activities was divided I started with the ones I had to do, which are:

- Entrance.
- Car.
- Someone playing in the court.

The first thing I accomplished was the donwload of a character with an animation from Mixamo. I chose one with a texture which seemed like a soccer player. Then I settled him in the court, created somo variables to move him through the plane and made the logic to move him and rotate him.

Then I began the creation of the building, I made it in 3ds Max, well before I had to watch some tutorials on YouTube because I had no idea of how I was supposed to begin. While I was creating the entrance I had some serious problems with the exportacion of the model to OpenGL. I still don't know what was wrong, but fortunately a teammate helped me to solve those problems.

So the object results like this, it's somo darker than the original one, (you can see it on the right), also I think it's a little bit larger.

Finally I realized the animacion of the car. I used a model from the Internet, it's a VolksWagen. There were few problems with the textures, so I had to change them. To pass it to OpenGL I exported the bodywork and the wheel separated.

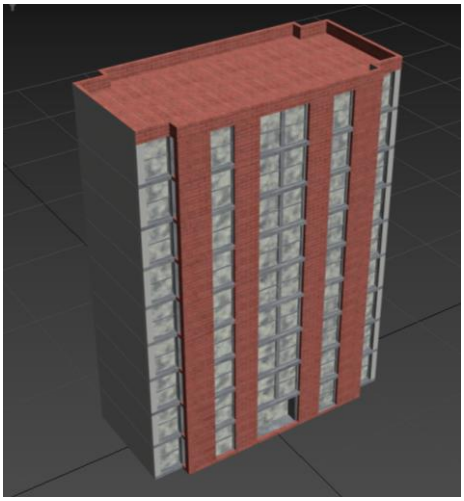
Then I colocated them next to the entrance so that I could make an animation of the movement of the car. The animacion of the car needs the user type the space key to begin, the animacion if formed of eight states, in the last one it paks and it's not possible to move it again.

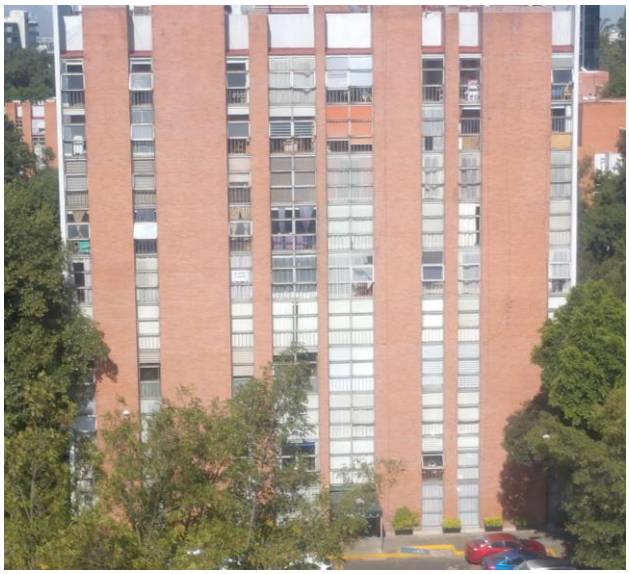
Salinas Romero Daniel:

Once the tasks of the project have been distributed. I decided to use a different software than 3DS MAX. I used Blender because I know modeling and texturing tools a little better. Because the references for the building and church models were close to me, it was easy to get every angle of view of every object. The reason I didn't use 3DS MAX is because it seemed a bit unintuitive when it came to loading textures, and there are plenty of tutorials out there to help.

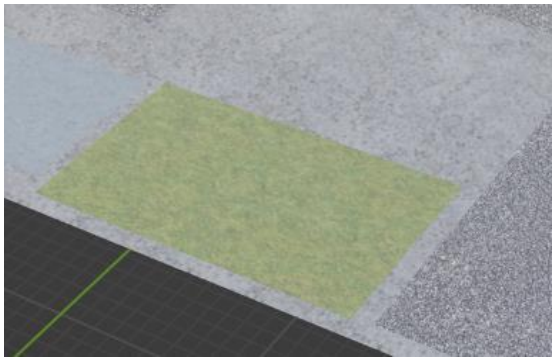
The tasks to be performed were modeling and placing:

- **Buildings.**

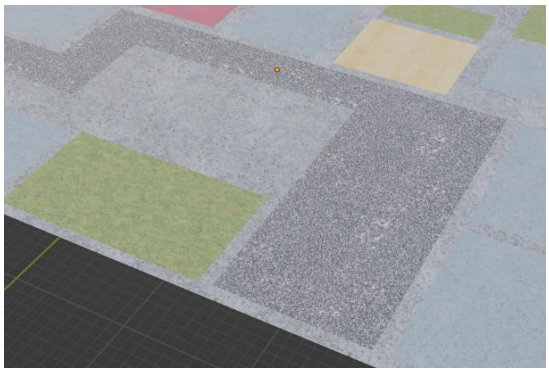


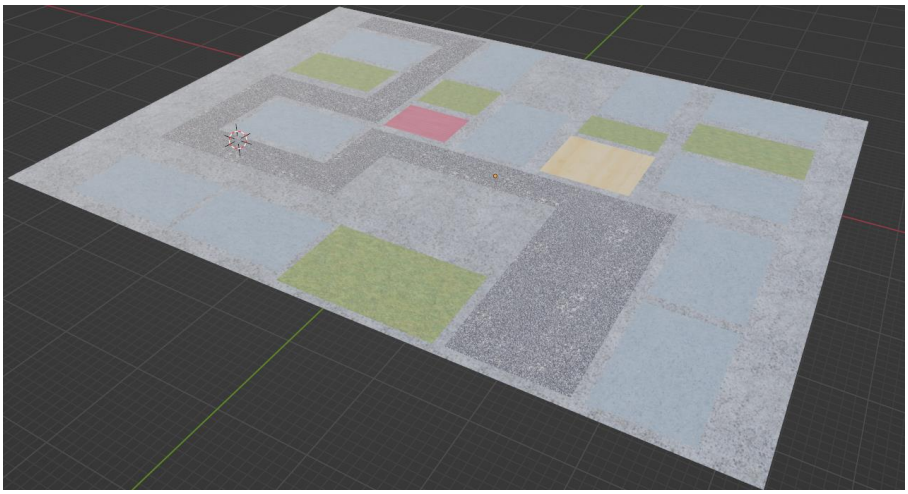


- **Green areas.**

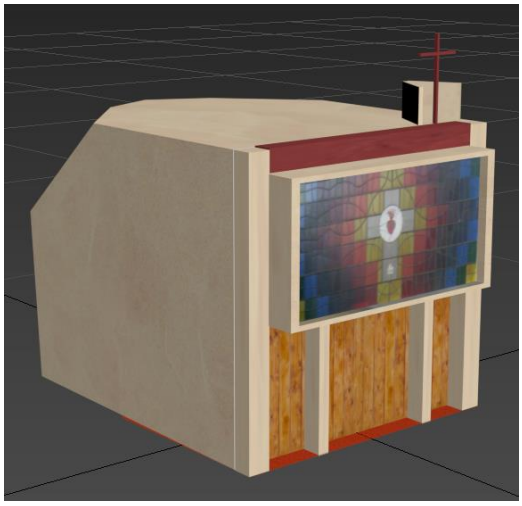


- **Streets.**





- **Church.**



I must say that I had no problem modeling any of the above points. Each model was built from simple figures such as cubes or planes. So there was no need to search for any free models on the internet. The only inconvenience I had was finding an adequate proportion and placing each building following the plan that we proposed in the design. However, reference photos were added to the design proposal.

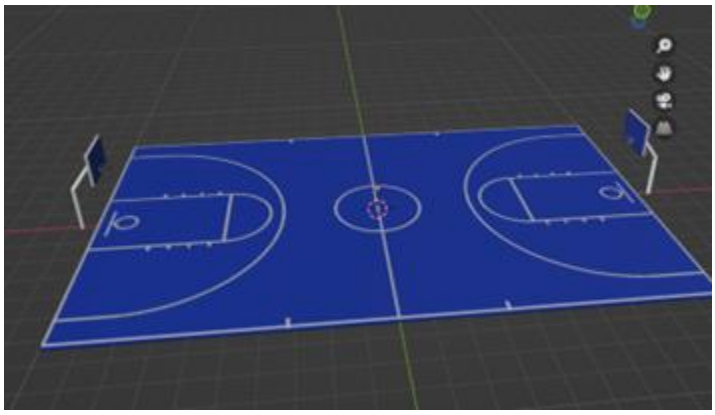
Vaquero Barajas Alexis:

The activities that I had to do were.

- Basketball court.
- Person walking a dog.
- Person walking.

For the basketball court, what I did was create the texture in Photoshop, where I used an image that had just the outlines of the court and then I added the blue background color to it so I could use it in the model. This one was created in Blender software, where the court was made with a cube, the baskets with cylinders and cubes, to finally be textured with the image.

The result:



For the person walking the dog, I used a model from Mixamo, which is a lady in sportswear doing a walking animation. Otherwise, I got the dog model from open3dmodel and to make its animation I used the "biped" tool in 3ds Max. I used the keyframe animation to each paw move in a 30-frame period.

Finally, for the person who walks, I exported a model from Mixamo that is a working person and performs an animation of walking from the left side.

Conclusiones.

The realization of this project was very interesting, but a lot of work from modeling the objects, adding textures, animating them and finally adding them to the project, personally it took me more time to learn the different software for modeling, editing images and animating in addition to obviously programming I consider that a project like this cannot be carried out only if you want to obtain a good result, but rather a team of specialists in modeling, UX/UI, programmers, etc. is needed. This project helped me realize all the work and investment behind projects like video games and what skills are needed to dedicate oneself to that area.

- Fernando Maceda Patricio.

My first impression of the project was that it was extremely big, it was a lot of work to do; but I think that something that helped us was the fact that we divided very well the tasks to accomplish and I'm so grateful with my partners because the did what they had to do in time and showing a huge disposition. In particular, I had some troubles during the project, errors when I exported the models, some difficulties to locate the objects and give them the correct size and especially to make an animation which fulfilled the objectives. Finally I consider that we

satisfied the goal of the project, we put in practice the topics which were studied along the course. ***- David Reyes Avila.***

The proposed objectives for this project were concluded. Using all the concepts seen throughout the semester. From the basics of geometric transformations to modeling, loading models, using procedural animations, keyframes and texturing. Although the lighting methods were studied, they are not part of the project objectives due to the computational cost that they represent. I consider satisfactory the result of teamwork resulting from a good distribution of work. The time we were given to complete the project was fair. In general, there were no complications to be able to integrate the work of each member within the complete project.

Personally, I had no problems being able to carry out my activities, as I mentioned before, the use of Blender is easier for me and due to the permissions that the teacher gave us, the result was satisfactory. I greatly appreciate that Blender is free software and I also thank my classmates for their time and availability, as well as the professor for his availability to answer our questions.

- Daniel Salinas Romero.

According to the project conducted, I can conclude that the work perfectly summarizes everything seen in the laboratory, since designing models as well as importing, contains topics such as: Texturing, Model Hierarchy, Translation, Rotation, Scale, and others.

Doing this project was interesting to me as well as a heavy workload. However, this made me admire more the work that goes into making much larger projects (such as movies, art designs, and video games) as it involves more work than expected one thinks as a spectator. ***- Vaquero Barajas Alexis.***

Referencias.

- Edificios de México (s.f.). Villa Olímpica Miguel Hidalgo [mensaje en un blog]. Recuperado de https://www.edemx.com/site/villa-olimpica-miguel-hidalgo/
- MIXAMO. (2022). Recuperado de https://www.mixamo.com/#/
- Open3DModel. (2022). Recuperado de https://open3dmodel.com/es/