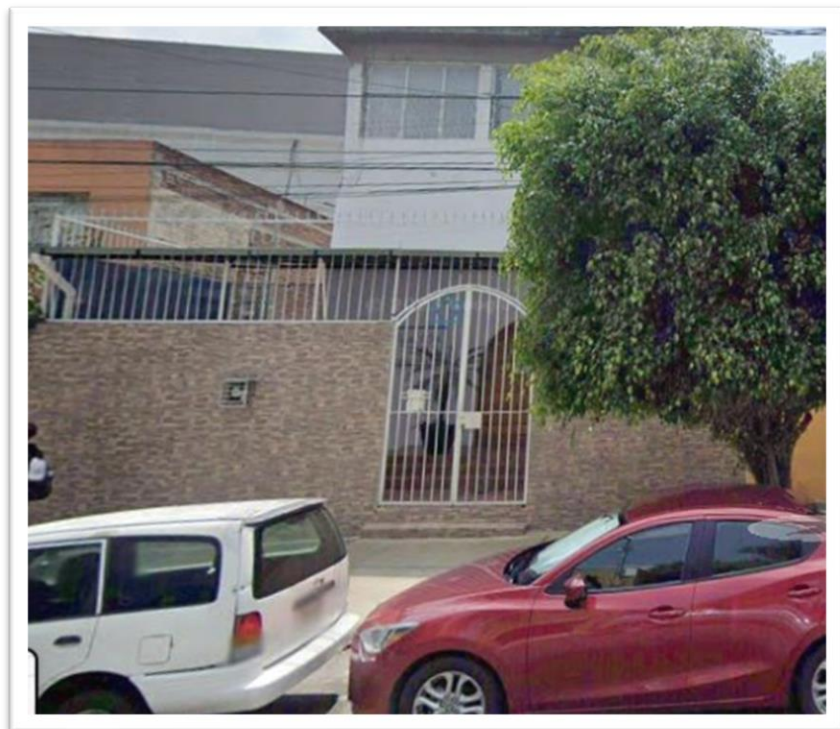
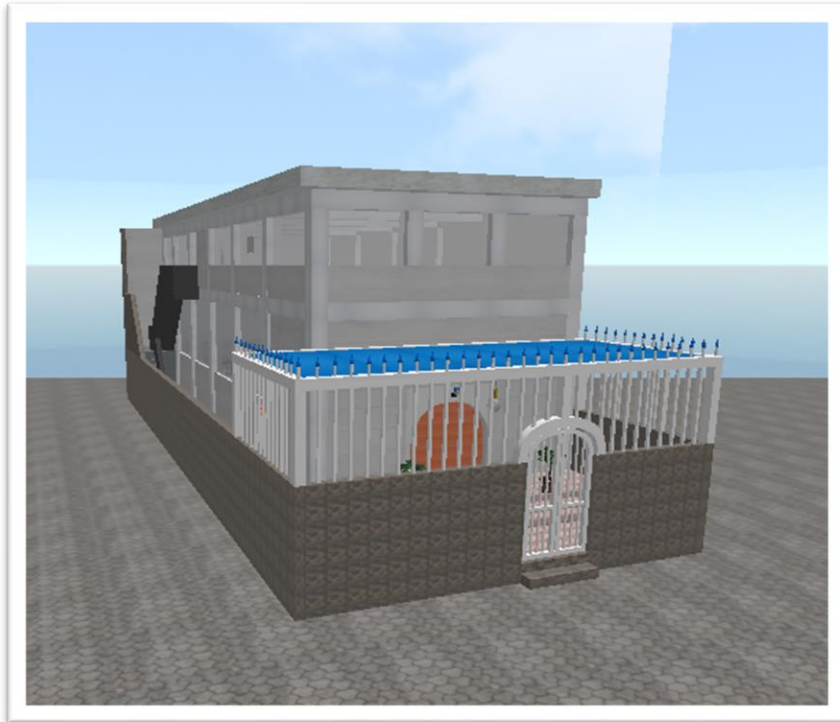


# Manual Técnico



## ***Contenido***

- Objetivos
- Alcance del Proyecto
- Limitantes
- Diccionario de variables
- Animaciones y fundamento matemático
- Conclusiones

## ***Objetivo General***

El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.

## ***Objetivos del proyecto***

Utilizar las herramientas de software especializadas para el manejo de 3D.

Aplicar los principios de modelado, texturalización, iluminación y animación adquiridos durante el curso de laboratorio, para aplicarlo en la recreación de un edificio de la vida real.

Hacer la recreación exclusivamente de la estructura de la vida real de una iglesia ubicado en la zona poniente de la Ciudad de México.

## ***Alcance del proyecto***

Este proyecto tiene el propósito de realizar una recreación 3 d de una iglesia de la vida real. Para ello hoy es necesario conocer las dimensiones y características físicas y estéticas de dicha edificación para otorgar el realismo más cercano en 3D.

Desde que este proyecto tiene como principales ejes temáticos la aplicación de los conocimientos para el modelado en 3 d, la texturalización, el manejo del ambientación con la iluminación, así como las animaciones sencillas, hoy es necesario conocer cada detalle de la estructura de la iglesia, hoy en la parte estética revisar el tipo de mosaicos, losetas, fachada, pisos, muros, techos, tipo de estructura, los principales objetos que identifican y conforman a esta estructura así como las dimensiones en lo que respecta a la longitud y volumen.

Con ello es necesario acudir presencialmente a este edificio para realizar la toma de fotografías y mediciones de cada uno de estos elementos, de tal manera que esas fotografías o imágenes pueden ser editadas para aplicarlas como texturas y tener referencias de los objetos para su modelado y aplicar las transformaciones correspondientes a cada uno de estos objetos y de ser posible aplicar en algunos de ellos animaciones sencillas.

En la parte de modelado se espera utilizar el uso de primitivas geométricas y cuerpos volumétricos sencillos para aplicarles transformaciones para obtener el objeto deseado. En esta misma parte del modelado a cada uno de estos objetos se le aplicaran las texturas correspondientes, definiendo los parámetros de opacidad, iluminación y estética. Dado que en la estructura se cuentan con múltiples texturas para cada uno de los elementos físicos, sólo se realizará la toma de fotografías para su edición de imágenes de aquellos elementos estéticos que sean más importantes y significativos de la estructura para otorgar el realismo más cercano posible en 3D. Para la parte de las animaciones se espera utilizar el uso de la rotación y la traslación en combinación para generar movimiento en alguno de los objetos de la estructura virtual.

Al finalizar el desarrollo a nivel de código, hoy se constituirán los manuales técnicos y de usuario para la descripción del funcionamiento de este proyecto. El proyecto final de vista del estudio así como el archivo ejecutable se entregarán exclusivamente en un repositorio de github.

## ***Limitantes***

La fecha definida para este proyecto es el día 10 de mayo sin oportunidad de prórroga.

Los recursos con los que se cuentan para realizar este proyecto solamente es una laptop dell que no cuenta con tarjeta gráfica, hoy por lo que es la ejecución y depuración del código requerirá de más tiempo.

Solamente se puede utilizar el software de modelado maya y blender para la creación de los objetos y modelos tridimensionales.

El proyecto se debe realizar en el software de Microsoft visual studio 2022, trabajar con alguna versión previa puede generar errores de compilación para el proyecto.

## *Diccionario de variables añadidas*

- VARIABLES

**float tiempo;:**

Esta variable se utiliza para almacenar el tiempo transcurrido desde el inicio del programa. Se actualiza en cada iteración del bucle principal para controlar el tiempo en la animación o en alguna otra función relacionada con el tiempo.

**float rot = 0.0f; y float rot2 = 0.0f;:**

Estas variables se utilizan para almacenar los ángulos de rotación de algún objeto o elementos gráficos en la escena.

**bool anim = false; y bool anim2 = false;**

Estas variables booleanas se utilizan para controlar el estado de la animación de uno o más objetos en la escena. Cuando anim o anim2 están establecidas como true, indica que la animación está en curso, mientras que false indica que la animación no está activa.

**float speed = 1.0f;**

Esta variable se utiliza para controlar la velocidad de la animación o el movimiento de algún objeto en la escena. Un valor de speed igual a 1.0f indica una velocidad normal. Si se aumenta este valor, la animación o el movimiento se acelerarán, y si se disminuye, se ralentizarán.

**float angle\_offset:**

Esta variable se utiliza para establecer un desplazamiento inicial del ángulo de rotación para la animación de la puerta izquierda. El ángulo de rotación se calculará a partir de este desplazamiento.

**float angular\_speed:**

Esta variable controla la velocidad de rotación para la animación de la puerta izquierda. Un valor mayor de angular\_speed resultará en una rotación más rápida, mientras que un valor menor hará que la rotación sea más lenta.

**float angle\_offset2, float angular\_speed2:**

Funcionan de manera similar a angle\_offset y angular\_speed, pero se aplican a la animación de la puerta derecha.

**float angle\_offset3, float angular\_speed3:**

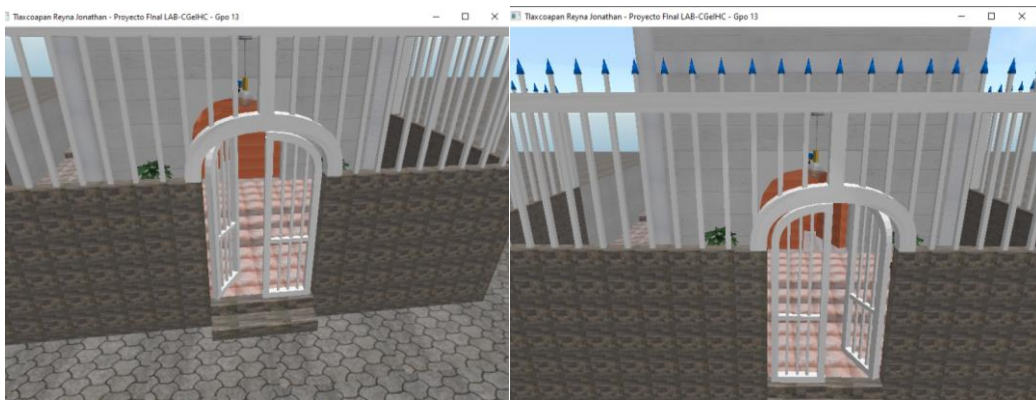
Estas variables controlan la animación de la cámara de seguridad. Al igual que las anteriores, angle\_offset3 establece un desplazamiento inicial del ángulo de rotación, mientras que angular\_speed3 controla la velocidad de rotación.

`float angle_offset4, float angular_speed4:`

Estas variables se utilizan para la animación compleja, probablemente involucrando múltiples objetos o movimientos simultáneos. Al igual que las otras variables, `angle_offset4` establece un desplazamiento inicial del ángulo de rotación, mientras que `angular_speed4` controla la velocidad de rotación.

## ***Animaciones y fundamento matemático en el código***

- ANIMACIONES SIMPLES 1 Y 2: Apertura y cierre de puertas



Para estas líneas de código se calculan y aplican la rotación de las puertas basándose en el tiempo y en parámetros como la velocidad de rotación y el ángulo máximo de oscilación, permitiendo que las puertas se abran y cierren de manera animada.

`float oscillation_angle = sin(glfwGetTime() * angular_speed) * rot;;`

Esta línea calcula un ángulo de oscilación basado en el tiempo utilizando la función seno. `glfwGetTime()` proporciona el tiempo actual, multiplicado por `angular_speed`, que controla la velocidad de la oscilación. El resultado se multiplica por `rot`, que representa el ángulo máximo de oscilación.

`float rotation_angle = oscillation_angle + angle_offset;;`

Aquí se aplica el desplazamiento inicial `angle_offset` al ángulo de rotación calculado anteriormente. Esto permite ajustar la posición inicial de la puerta.

`rotation_angle = glm::clamp(rotation_angle, 0.0f, 120.0f);`

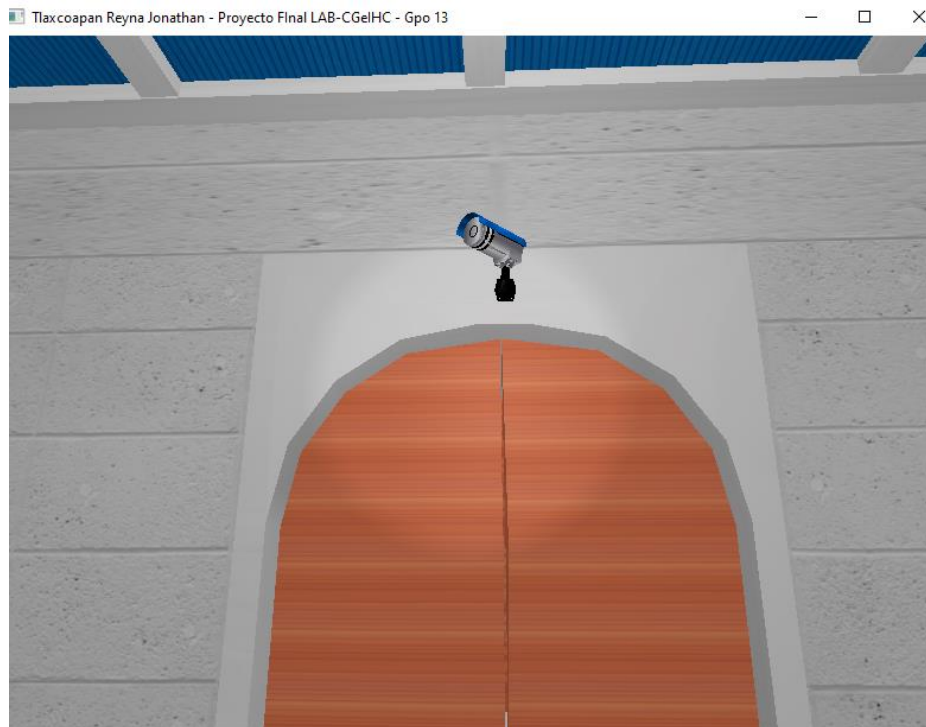
Esta línea limita el ángulo de rotación dentro de un rango específico. En este caso, el ángulo se limita entre 0 y 120 grados para una de las puertas, asegurando que la puerta no rote más allá de esos límites.



```
if (rotation_angle >= 0.0f || rotation_angle <= 120.0f) { angle_offset = -angle_offset; };
```

Estas condiciones comprueban si el ángulo de rotación está en los límites definidos (0 y 120 grados). Si lo está, se invierte la dirección de la oscilación multiplicando `angle_offset` por -1, lo que cambia la dirección de la animación de la puerta. El código restante aplica las transformaciones de traslación y rotación a las matrices de modelo (`model_Puerta1` y `model_Puerta2`) y luego renderiza las puertas (`P1izq`, `P1der`, `P2izq`, `P2der`) en la escena.

- Animacion simple # 3: Rotacion automatica de camara y Fundamento matematico



Para estas líneas de código calculan y aplican la rotación de una cámara de seguridad basándose en el tiempo y en parámetros como la velocidad de rotación y el ángulo máximo de oscilación, permitiendo que la cámara se mueva de manera animada dentro de ciertos límites de ángulo.

```
float oscillation_angle3 = sin glfwGetTime() * angular_speed3 * 45.0f;;
```

Similar al caso anterior, esta línea calcula un ángulo de oscilación basado en el tiempo utilizando la función seno. `glfwGetTime()` proporciona el tiempo actual, multiplicado por `angular_speed3`, que controla la velocidad de la oscilación. El resultado se multiplica por `45.0f`, lo que determina el ángulo máximo de oscilación de la cámara.

```
float rotation_angle3 = oscillation_angle3 + angle_offset;;
```

Al igual que antes, se aplica un desplazamiento inicial (`angle_offset`) al ángulo de rotación calculado anteriormente.

```
rotation_angle3 = glm::clamp(rotation_angle3, -45.0f, 45.0f);
```

Esta línea limita el ángulo de rotación dentro del rango de -45 a 45 grados, asegurando que la cámara no rote más allá de esos límites.

```
if (rotation_angle3 >= 45.0f || rotation_angle3 <= -45.0f) { angle_offset3 = -angle_offset3; };
```

Al igual que antes, estas condiciones comprueban si el ángulo de rotación está en los límites definidos (-45 y 45 grados). Si lo está, se invierte la dirección de la oscilación multiplicando `angle_offset3` por -1.

El código restante aplica las transformaciones de traslación y rotación a la matriz de modelo `model` y luego renderiza la cámara (Cam) en la escena utilizando OpenGL.

- Animacion Compleja: Fundamento matematico



Para esta animación se realiza el movimiento de un dron siguiendo una trayectoria de media circunferencia dónde asciende y desciende. Se emplea un shader para efectuar la traslacion.

```
tiempo = glfwGetTime() * speed;
```

Aquí, `tiempo` se establece como el tiempo actual multiplicado por la velocidad (`speed`). El tiempo se está utilizando para controlar la animación del objeto, y la velocidad determina qué tan rápido avanza la animación.

```
float oscillation_angle4 = sin(glfwGetTime() * angular_speed4) * 45.0f;
```

Similar a los casos anteriores, esta línea calcula un ángulo de oscilación basado en el tiempo utilizando la función seno. `glfwGetTime()` proporciona el tiempo actual, multiplicado por `angular_speed4`, que controla la velocidad de la oscilación. El resultado se multiplica por 45.0f, lo que determina el ángulo máximo de oscilación del objeto.

```
float rotation_angle4 = oscillation_angle4 + angle_offset;;
```

Se aplica un desplazamiento inicial (angle\_offset) al ángulo de rotación calculado anteriormente, al igual que en los casos anteriores.

```
rotation_angle4 = glm::clamp(rotation_angle4, -270.0f, 270.0f);;
```

Esta línea limita el ángulo de rotación dentro del rango de -270 a 270 grados, asegurando que el objeto no rote más allá de esos límites.

```
glUniform1f(glGetUniformLocation(Anim.Program, "time"), tiempo);;
```

Aquí se envía la variable tiempo al shader como una variable uniforme llamada "time", para controlar la animación en el shader.

## **Conclusiones**

Hoy en base al objetivo general y a los objetivos planteados a nivel de desarrollo se cumplieron gran parte de los aprendizajes adquiridos durante el curso. Sin embargo la parte de la documentación quedó inconclusa.


Este proyecto implicó mucho tiempo de desarrollo, en particular en la sección de modelado. Fue necesario acudir a la edificación de la vida real para realizar la toma de medidas respecto a las longitudes y volúmenes de los objetos seleccionados para el proyecto. Al contar con esta referencia, se procedió a realizar el modelado de cada uno de estos objetos, pisos, muros, y texturas. Dado que fue la primera vez que se empleó el software de modelado en 3D, llevéal menos 2 meses familiarizarse con este software para aplicar las texturas y ajustes de iluminación del especular para poder realizar la exportación apropiada de los objetos en opengl. En lo que concierne a los principios de las transformaciones y modelado se satisface el aprendizaje, ya que la carga de estos modelos en opengl se cargaron sin problema alguno.

Hoy el desarrollo de la iluminación llevo tiempo de 3 semanas de desarrollo, debido a que fue necesario entender las 3 interacciones fundamentales de la luz para otorgar la iluminación apropiada, ya que de lo contrario, los errores frecuentes que se presentaron fueron que los objetos se veían muy opacos, había ausencia del especular en los objetos previamente modelados en maya, exceso de brillo, ajuste de los arreglos para las point lights y el uso apropiado de las componentes ambiental y difusa.

Las animaciones fue el principal foco de dificultad para este proyecto, dado que en el curso aprendimos los principios de animación utilizando un vertex shader y combinando las transformaciones geométricas, la dificultad radicó en encontrar las ecuaciones lineales y paramétricas apropiadas para efectuar el movimiento en los objetos seleccionados.

Este es uno de los proyectos más largos que he desarrollado durante la carrera, sin duda disfruté de acudir presencialmente a la estructura para realizar la toma de medidas y poder otorgar el realismo más cercano para este proyecto, y sin duda alguna mi nulo conocimiento inicial respecto al modelado en maya implicó mucho tiempo de desarrollo y de realizar los ajustes en el código del proyecto de visual studio, hoy pero el resultado del desarrollo considero que es muy satisfactorio, pero la parte de la documentación al dejarla





al final implicó que no cumpliera con los requisitos fundamentales para completar el manual técnico.