



# **Manual del Proyecto para el Examen Extraordinario de Computación Gráfica e Interacción Humano- Computadora**

FAJARDO TÉLLES CRISTIAN



# En este manual se presenta las especificaciones de la configuración

El enfoque principal estará relacionado con la creación de un escenario interactivo con modelos 3D importados, como los requeridos en el proyecto con simulación, visualización y enseñanza. Gracias a la formación y habilidades técnicas, pude diseñar y ejecutar una animación básica que involucran transformaciones geométricas y manejo de archivos en formatos como .obj, asegurando una integración óptima en entornos de desarrollo en Visual Studio.

El *expertise* también incluye la implementación de un control interactivo, como la repetición de animación mediante teclado, y la gestión de recursos a través del repositorio público en la plataforma GitHub, garantizando transparencia en el proyecto, como validación del laboratorio para presentar el Examen Extraordinario.



# Índice

## Introducción

03 Conceptos Básicos

04 Objetivo del Proyecto

## Sistemas Operativos

05 Windows 11

## Bibliotecas Adicionales

06 Listado de Bibliotecas Implementadas

## Configuración

07 Explicación

10 Pasos para ver la Animación



# Introducción

# Concepto Básico

Las **Torres de Hanoi** (a veces escrito como "Hanoi") es un rompecabezas matemático clásico que ilustra conceptos fundamentales de recursividad, estructuras de datos y algoritmos.

El objetivo del juego es mover todos los discos desde la torre de origen hasta la torre de destino, usando una torre auxiliar, siguiendo ciertas reglas. En cuanto a los elementos del problema requeridos, son:

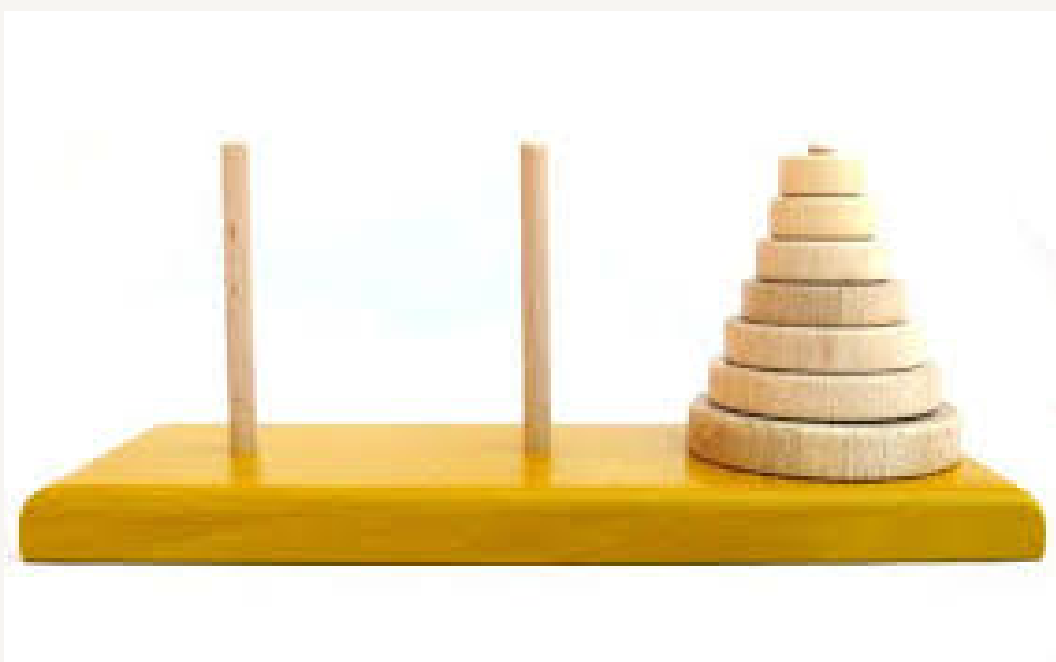
- **Tres torres o postes** (A, B y C).
- **Varios discos de diferente tamaño**, apilados inicialmente en una torre (por lo general en la torre A) en orden decreciente: el disco más grande en la base y el más pequeño arriba.

Las reglas que conforman el juego son:

1. Solo se puede mover un disco a la vez,
2. Solo el disco superior de cualquier torre puede moverse, y
3. Nunca se puede colocar un disco más grande sobre uno más pequeño.

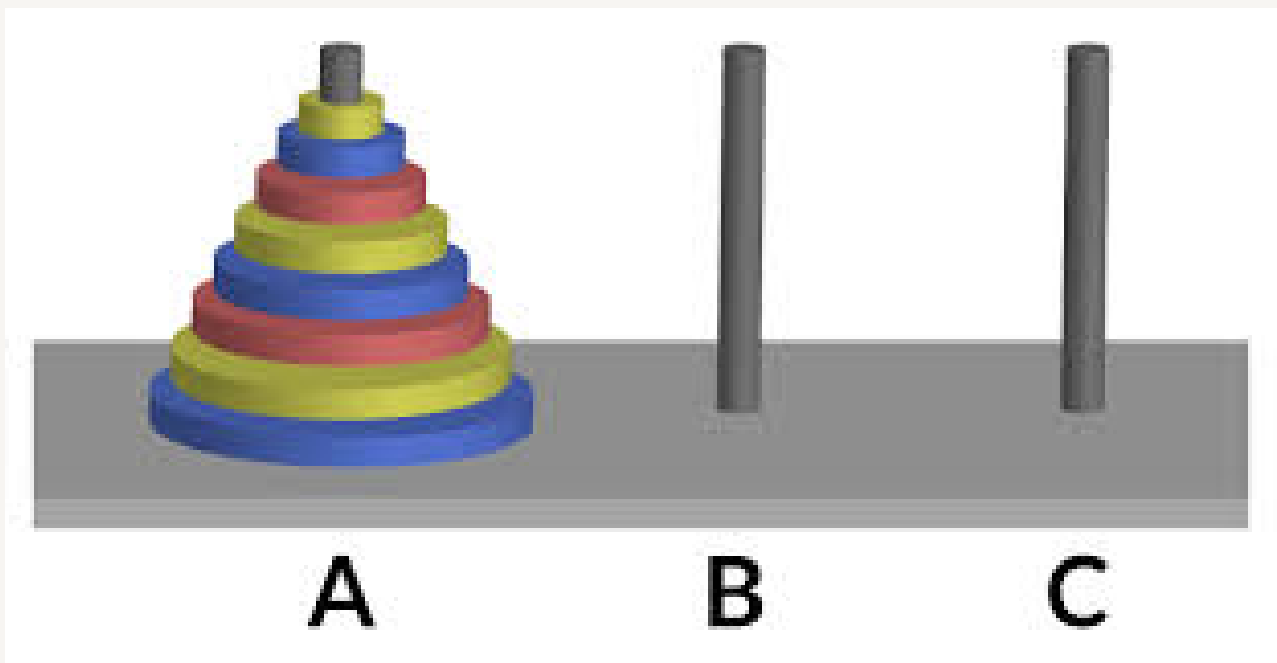
Y por último, este juego es usado para:

- Enseñanza de la **recursión** en programación.
- Ejemplos en **estructuras de datos** (como pilas o stacks).
- Modelos de problemas de **división y conquista**.
- Reflexiones filosóficas o matemáticas (leyenda de los monjes en Hanói).



# Objetivo del Proyecto

Adquirir y aplicar conocimientos en la programación de gráficos por computadora, mediante la creación de un escenario interactivo en **OpenGL 3.1** (o superior), que represente las **Torres de Hanoy**. Este escenario estará compuesto por modelos 3D en formato .obj, incluyendo cilindros, toroides y prismas, importados al programa, y contará con una animación que demuestre dominio en transformaciones geométricas básicas, siendo explicada y grabada en un video.\*



*\*La imagen de esta página es sólo una ilustración de lo que se verá en grabación.*



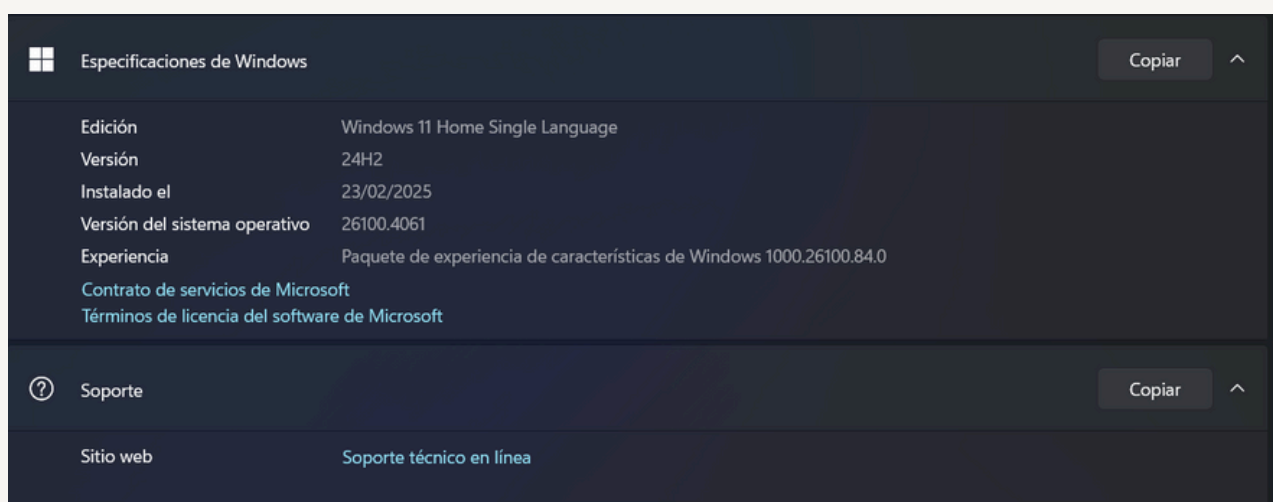
# Sistemas Operativos

# Windows 11

Además de ser el Sistema Operativo (SO) que más se ha manejado en el transcurso de la asignatura, dado que se recomienda el uso de Visual Studio (en mi caso, 2022)—entorno de desarrollo que se integra de forma nativa y optimizada en Windows—este sistema operativo resulta la opción más directa y conveniente para el proyecto.

Esto se debe a que Windows permite aprovechar la amplia compatibilidad con controladores gráficos de OpenGL (al menos la versión 3.1) y facilita la gestión y configuración de bibliotecas adicionales a través de instaladores y gestores propios. Es prudente decir que, Windows ha demostrado ser una plataforma robusta para proyectos los académicos de computación gráfica, vistos en el curso, donde las herramientas de depuración y las facilidades para la integración con el repositorio público en GitHub son esenciales para una implementación más que exitosa.

En la siguiente captura, está las especificaciones del SO que usé para hacer funcionar el proyecto (como mínimas).







# **Bibliotecas Adicionales**

# Listado de Bibliotecas Implementadas

## Estándar y Matemáticas

- `<iostream>` Sirve para entradas y salidas por consola, permitiendo el manejo de flujos de datos (por ejemplo, con `std::cout` y `std::cin`), lo cual es fundamental para depuración o interacción básica en programas que no poseen una interfaz gráfica elaborada.
- `<cmath>` Provee un conjunto completo de funciones matemáticas (como `sin`, `cos`, `pow`, entre otras) que son esenciales para los cálculos geométricos y de transformación en gráficos computarizados.

## Manejo de OpenGL

- `<GL/glew.h>` ([GLEW – OpenGL Extension Wrangler Library](#)). En otras palabras, simplifica el manejo de punteros a funciones de OpenGL; tras crear un contexto de OpenGL (por ejemplo, con GLFW), se invoca `glewInit()` para asegurarse de que todas las funciones de OpenGL que necesites estén disponibles.
- `<GLFW/glfw3.h>` ([GLFW – Graphics Library Framework](#)). Facilita la creación de ventanas, contextos OpenGL y el manejo de entrada (teclado, ratón, etc.) de una manera multiplataforma.

## Carga y Manipulación de Imágenes

- `"stb_image.h"`. Permite cargar formatos como JPEG, PNG, BMP, etc., devolviendo los datos de la imagen en memoria para que puedan ser usados como texturas en OpenGL.
- `"SOIL2/SOIL2.h"`. Similar en su propósito a `stb_image`, SOIL2 (Simple OpenGL Image Library 2) se orienta a la carga y procesamiento de imágenes, especialmente para la generación de texturas.

## Matemáticas Avanzadas y Transformaciones

- `<glm/glm.hpp>` ([OpenGL Mathematics](#)). Es una biblioteca header-only que provee herramientas matemáticas optimizadas para gráficos 3D.
- `<glm/gtc/matrix_transform.hpp>` Este módulo específico de GLM facilita la construcción de matrices de transformación (por ejemplo, matrices de proyección, vista y modelo), muy útiles para manipular objetos en el espacio 3D.
- `<glm/gtc/type_ptr.hpp>` Este header ofrece funciones que permiten la conversión de tipos de datos GLM a punteros nativos, lo que resulta esencial al pasar estas estructuras directamente a las funciones de OpenGL (como, al enviar matrices de transformación al shader).



# Configuración

# Explicación

## Estructura e Integración de Bibliotecas

*Inicialización y Contexto de Ventana.* El código comienza inicializando GLFW, que se encarga de crear y gestionar la ventana de la aplicación. Se configura el contexto OpenGL (en este caso, se comentan algunas líneas relacionadas con la versión y el perfil, lo que puede depender de la compatibilidad requerida para el proyecto). Una vez creada la ventana con dimensiones predefinidas (800×600), se asigna el contexto y se obtiene el tamaño del framebuffer, asegurando que la configuración de la vista coincida con la resolución del dispositivo.

### *Bibliotecas para Extensiones y Cálculos Matemáticos*

- **GLEW (OpenGL Extension Wrangler Library):** Se activa el modo experimental para obtener punteros a funciones modernas de OpenGL, lo que es fundamental para utilizar características avanzadas del API.
- **GLM (OpenGL Mathematics):** Facilita los cálculos matriciales y vectoriales necesarios para transformaciones y proyecciones en 3D.
- **stb\_image y SOIL2:** Estas bibliotecas se utilizan para cargar texturas e imágenes, lo que permite aplicar materiales realistas a los modelos.

*Gestión de Shaders.* Se cargan dos programas de sombreado (shaders): uno para la iluminación general de la escena ("lighting.vs" y "lighting.frag") y otro para representar fuentes de luz, como la lámpara ("lamp.vs" y "lamp.frag"). Estos shaders son esenciales para calcular la iluminación, el color, y los reflejos, aplicando uniformes que controlan la interacción de la luz con las superficies de los modelos.

## Carga y Representación de Modelos

El código aprovecha una clase Model para cargar objetos 3D desde archivos externos en formato OBJ. Entre los modelos, se incluyen:

- **Prismas:** Cuatro modelos desde la carpeta "Models/Prismas".
- **Toroides:** Tres modelos que representan los componentes animados de la escena, denominados Toroide1, Toroide2 y Toroide3.
- **Cilindros y Plano:** Otros elementos geométricos que componen la escena.

Cada modelo se dibuja en la escena aplicando transformaciones mediante matrices (usualmente identidad o traducciones específicas, como en el caso de los toroides), lo que permite posicionarlos en el espacio 3D de forma ordenada y coherente.

Además, se configuran buffers de vértices (VAO y VBO) para gestionar primitivas geométricas básicas, lo que es práctico cuando se requieren objetos de bajo nivel en la escena.

### Sistema de Iluminación y Cámara

El apartado de iluminación es bastante completo:

- **Luz Direccional:** Simula una fuente de luz lejana (como la luz solar) con componentes ambiental, difusa y especular definidos.
- **Punto de Luz:** Su intensidad se modula dinámicamente usando funciones seno que dependen del tiempo, lo que añade un efecto de cambio periódico en el ambiente.
- **Spotlight (Foco):** Sigue la posición y dirección de la cámara, ofreciendo un efecto de linterna que resalta lo que "mira" el usuario en cada momento.
- La cámara se maneja a través de una clase Camera, que gestiona la posición, la dirección y la configuración de la proyección. Se generan matrices de vista y proyección que se pasan a los shaders para alinear correctamente la perspectiva de la escena.
- **Incrementos Graduales:** Las variables `Toroidex1`, `Toroidey1`, etc., se actualizan en pequeños incrementos (0.001 unidades por frame) para conseguir movimientos suaves y secuenciales.
- **Secuenciación y Reinicio:** Cuando se alcanza un cierto umbral (por ejemplo, cuando `Toroidey1` o `Toroidex1` llegan a un límite determinado), el código cambia la bandera correspondiente para iniciar la siguiente fase o, en su caso, reinicia la animación mediante la función `reinicio()`. Esto permite que la animación se ejecute de forma cíclica o controlada según un patrón predefinido.

En paralelo, el manejo de entrada se integra de manera precisa:


- **Eventos de Teclado:** Se utiliza `KeyCallback` para registrar las pulsaciones de teclas y actualizar el estado del arreglo `keys`. Esto facilita movimientos de cámara (usando WASD o flechas) y la activación de acciones como reiniciar la animación o alternar la iluminación.
- **Eventos de Ratón:** `MouseButtonCallback` captura el movimiento del ratón para modificar la orientación de la cámara, creando una experiencia inmersiva en primera persona.

### Animación y Control de Eventos

El código incluye un módulo de animación que modifica progresivamente la posición de los modelos tipo toroide. Se utilizan múltiples banderas (por ejemplo, `Anim1`, `Anim11`, etc.) para definir diferentes fases en la animación:

### Ciclo de Renderizado y Flujo de Ejecución

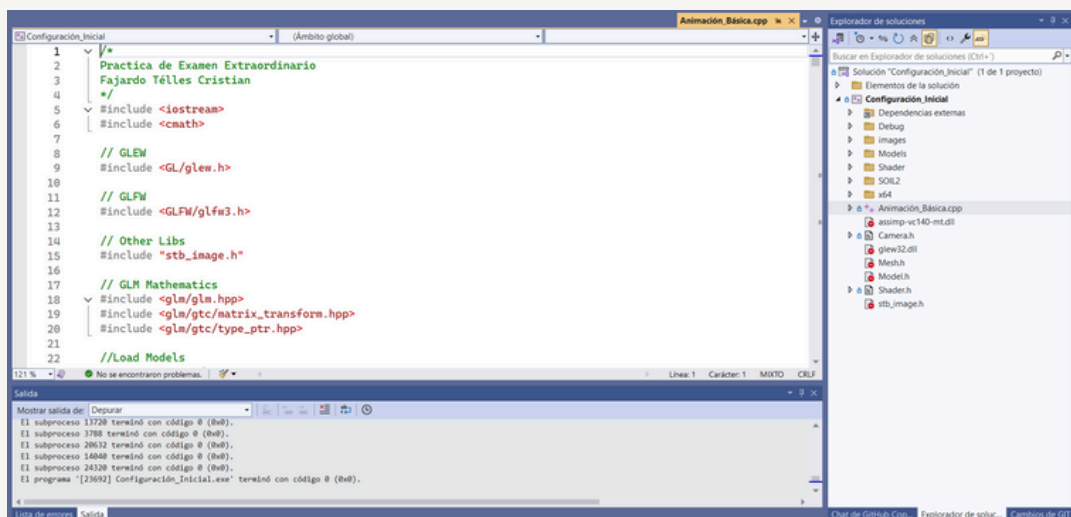
Dentro del "game loop", el código sigue estos pasos de forma reiterada:

- 
- *Cálculo de Delta Time:* Se determina el tiempo transcurrido entre frames para garantizar animaciones y movimientos a velocidad constante, sin depender de la tasa de refresco del sistema.
  - *Procesamiento de Entrada y Actualización de Animaciones:* Se gestionan los movimientos de la cámara y se actualizan las variables de animación según la secuencia lógica definida.
  - *Configuración del Renderizado:* Se limpia el buffer de color y de profundidad, se activa el test de profundidad y se establecen las transformaciones (view, projection, model) a través de uniformes en los shaders.
  - *Dibujo de los Modelos y Fuentes de Luz:* Cada modelo se renderiza con su transformación específica. El uso de shaders para la iluminación asegura que tanto los materiales como la dinámica de la luz se apliquen correctamente.
  - *Actualización de la Ventana:* Finalmente, se intercambian los buffers para presentar la imagen renderizada en pantalla.

Al final, es una combinación de múltiples conceptos en gráficos por computadora: gestión de contextos OpenGL, carga de modelos, iluminación dinámica y animación secuencial. Esto ayudó a mantener el código organizado y escalable, facilitando futuras mejoras o la integración de nuevos elementos para la configuración modular y el uso de funciones dedicadas para cada tarea (como el procesamiento de entrada, actualización de animaciones y renderizado).

# Pasos para ver la Animación

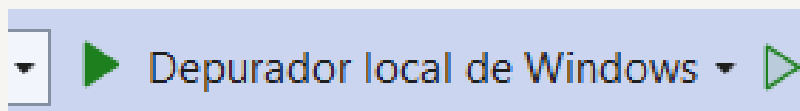
**Paso 1:** Al abrir Configuración\_Inicial.sln en Visual Studio 2022, haga clic en el archivo Animación\_Básica.cpp en el *Explorador de Soluciones* para ver el Código Principal del Proyecto.



**Paso 2:** Cambiar las opciones en el IDE a Debug y x86.

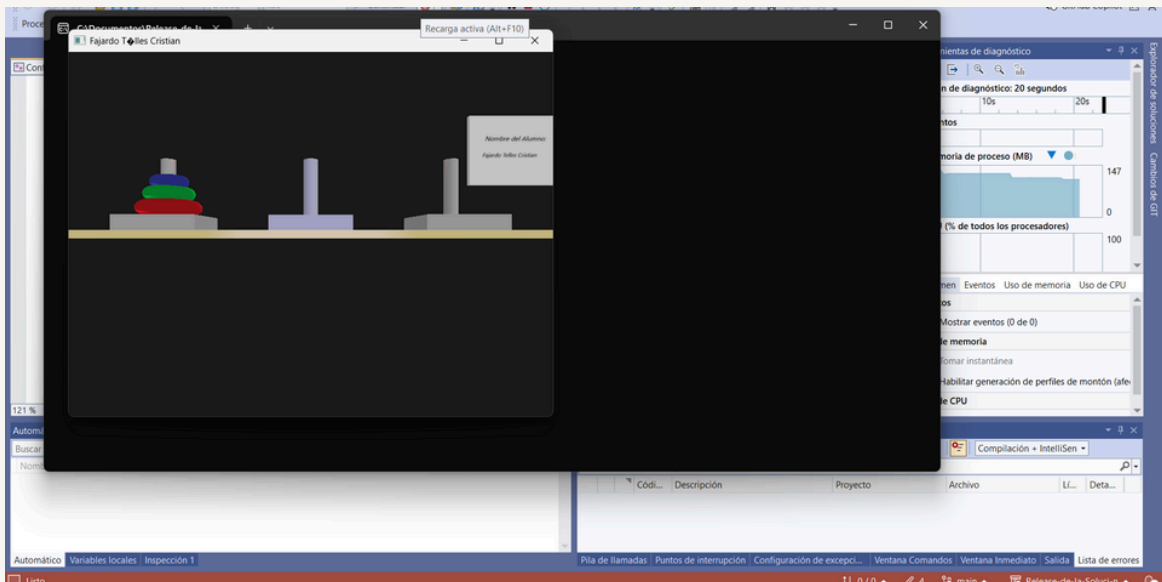


**Paso 3:** Haga clic en Depurador Local de Windows para ejecutar el proyecto.



# Pasos para ver la Animación

**Paso 4:** Al aparecer la siguiente ventana, podrá interactuar con la animación por medio de los siguientes botones:



*Cursor/Mouse:* Mover la visión de la cámara fija (1ra persona)

*Tecla W:* Agrandar (zoom)

*Tecla A:* Mover la cámara a la izquierda

*Tecla S:* Disminuir (zoom)

*Tecla D:* Mover la cámara a la derecha

*Tecla R:* Reiniciar la animación

*Tecla N:* Iniciar la animación

**Paso 5 (Final):** Para cerrar la animación, haga clic en el botón **Cerrar (X)** de las ventanillas, y **Cerrar (X)** en Visual Studio.

**GRACIAS POR SU CONSIDERACIÓN. ¡QUE TENGA UN BUEN DÍA!**