UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO-COMPUTADORA

EXTRAORDINARIO

PROFESOR: VALENCIA CASTRO LUIS SERGIO

ALUMNO: CONTRERAS TORRES EDGAR ALAN

Introducción

El siguiente documento es un manual de usuario solicitado en el 3er periodo de extraordinario del año 2025-2. La materia presentada es Computación Gráfica e Interacción Humano-Computadora. Donde se muestra un escenario tridimensional, el cual consta de elementos como prismas, cilindros, toroides y un plano. Se usaron técnicas de modelado geométrico, modelado jerárquico y texturizado.

Requerimientos necesarios del sistema

El software Autodesk 3ds max 2023 requiere las siguientes características mínimas de sistema:

- ➢ Procesador multi núcleo Intel o AMD de 64 bits
- ➢ 4Gb de RAM
- ➢ 9 Gb de espacio en disco
- ➢ Hardware gráfico: NVIDIA GeForce GTX 960 o AMD Radeon HD 7770

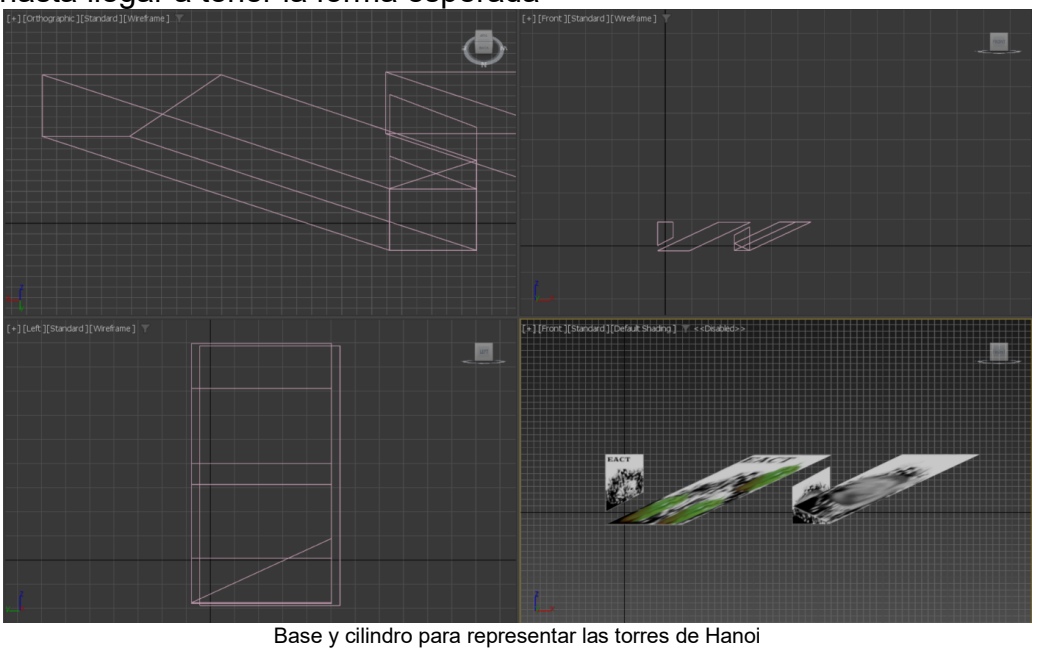
Se utilizará Visual Studio 2022 o superior como entorno de desarrollo, cuyos requisitos son:

- ➢ SO Windows 2010 o superior
- ➢ Procesador x64 (se recomienda de 4 núcleos)
- ➢ Mínimo 4 Gb de RAM
- ➢ La instalación típica requiere entre 20 y 50 Gb de espacio libre. Se recomienda unidad SSD para mejorar el rendimiento.

- ➢ Tarjeta de vídeo que admita una resolución de pantalla mínima de WXGA (1366 x 768)

Modelado

Se realizaron 4 cajas, las cuales se estuvo jugando con las figuras y modificándolas por sus vértices hasta llegar a tener la forma esperada

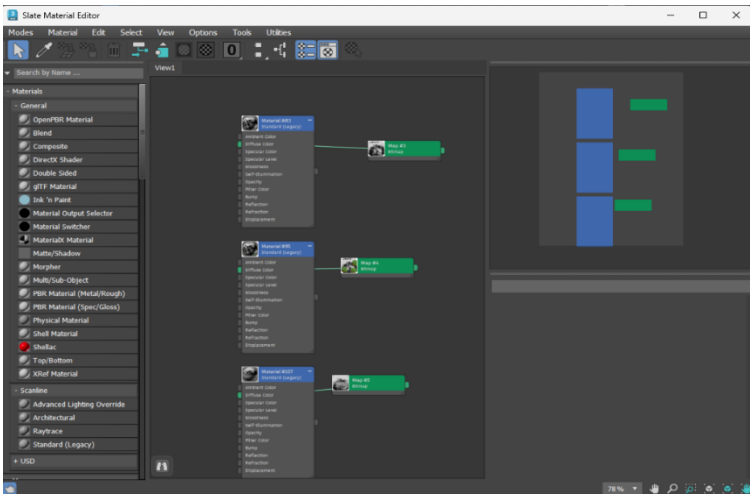


Base y cilindro para representar las torres de Hanoi

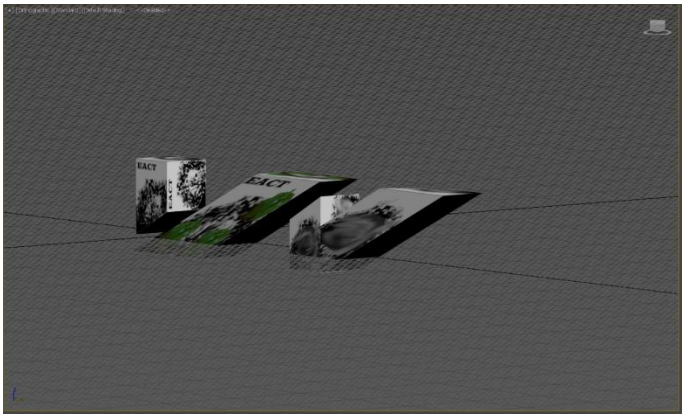
Texturizado

Utilizando la herramienta 3D MAX y la técnica de bitmap, con ello se logró texturizar cada elemento independiente.

Posteriormente, se descargaron en una imagen .jpg en cada caso.



Al final, se descargaron 3 modelos diferentes para poder tener control de cada uno de manera independiente



Animaciones

Se agregaron las siguientes animaciones:

Animación 1

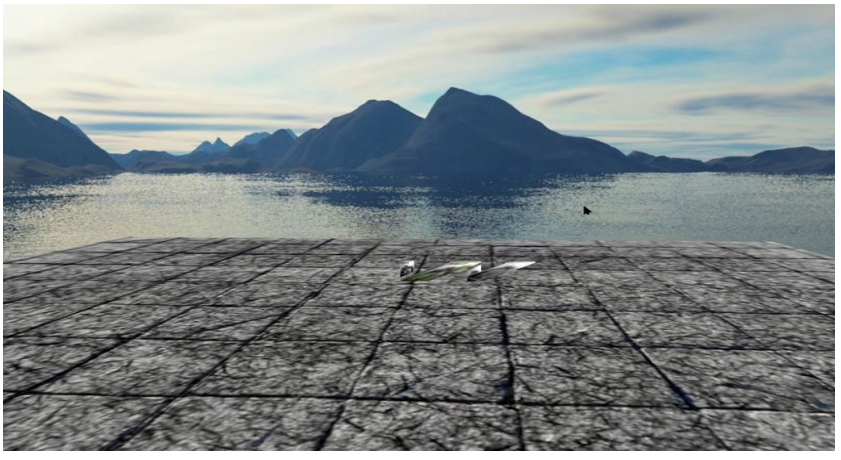
- Las letras están suspendidas en 500.0f en la coordenada y, por lo que se busca su rápida caída con la fórmula 0.5*g*t*t.

```
if (volar == 0) {

		movTor1_y -= 0.5 * gravedad * tiempo *
tiempo;
		movTor2_y -= 0.5 * gravedad * tiempo *
tiempo;
		movTor3_y -= 0.5 * gravedad * tiempo *
tiempo;
		tiempo += dt;
		printf("1movTor3_y: %f \n", movTor3_y);
		if (movTor3_y <= -490.0f) {
			volar = 1;

		}
}
```

Animación 2

- Una vez realizada la caída, este rebotara aumentando poco a poco hasta llegar a cierto valor y posteriormente caerá de nuevo

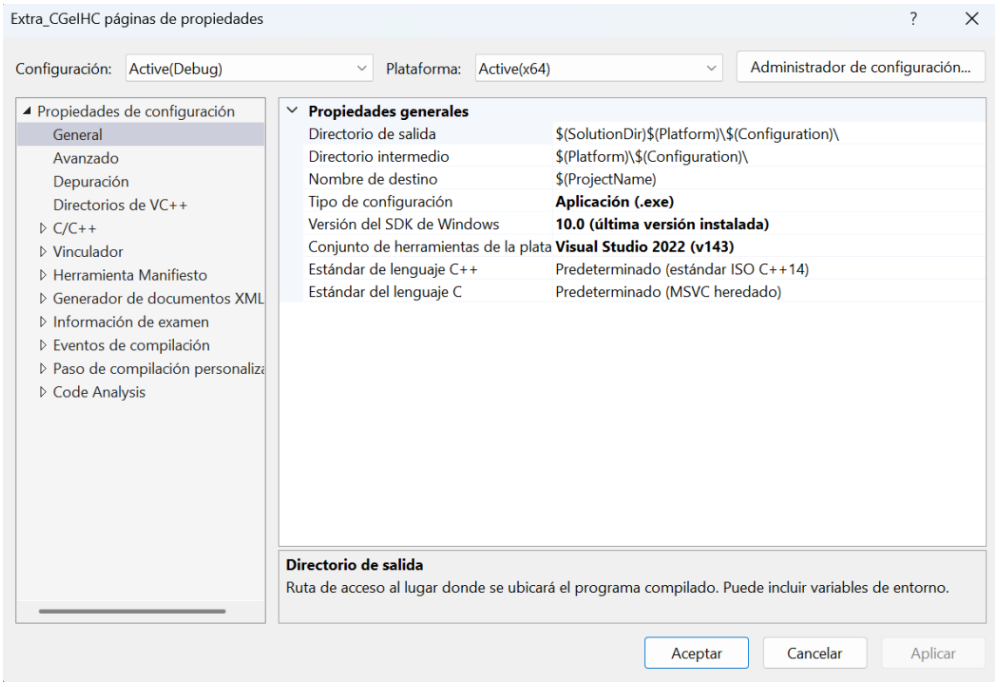
```
if (volar == 1) {
		movTor1_y += 0.5f;
		movTor2_y += 0.5f;
		movTor3_y += 0.5f;
		printf("2movTor3_y: %f \n", movTor3_y);
		if (movTor3_y >= -480.0f) {
			volar = 2;
				//animacion = true;
		}
if (volar == 2) {
		movTor1_y -= 0.5f;
		movTor2_y -= 0.5f;
		movTor3_y -= 0.5f;
		printf("3movTor3_y: %f \n", movTor3_y);
		if (movTor3_y >= -500.0f) {
				volar = 3;
				//animacion = true;
			}
		}
```



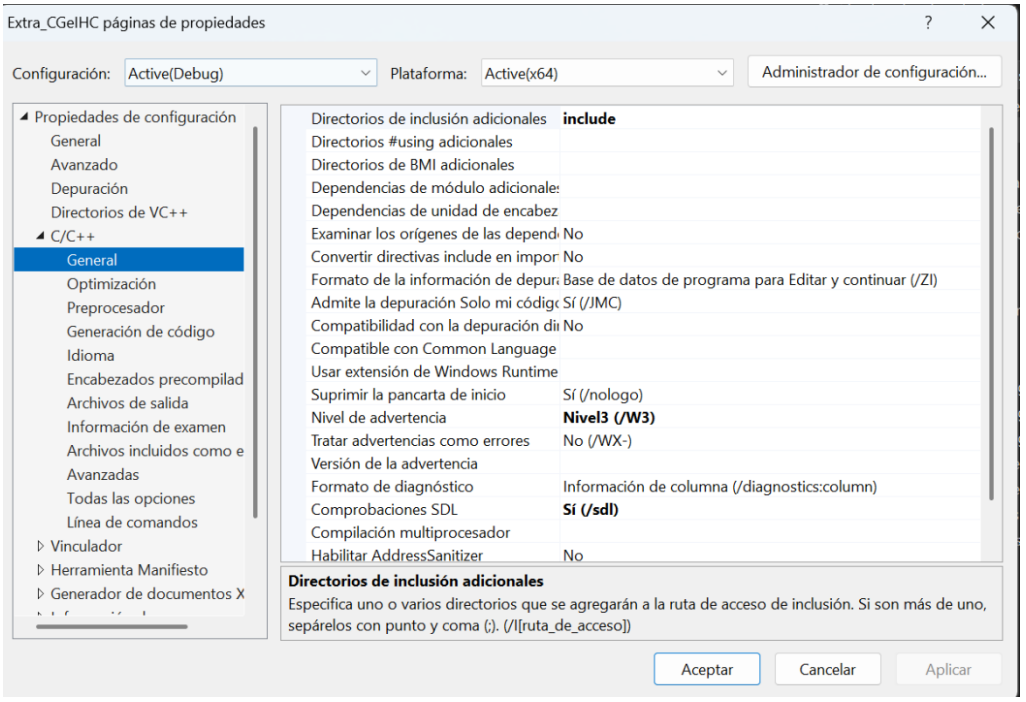
Configuración

El proyecto funciona para la plataforma Visual Studio 2022, se deberán realizar las configuraciones correspondientes en el apartado

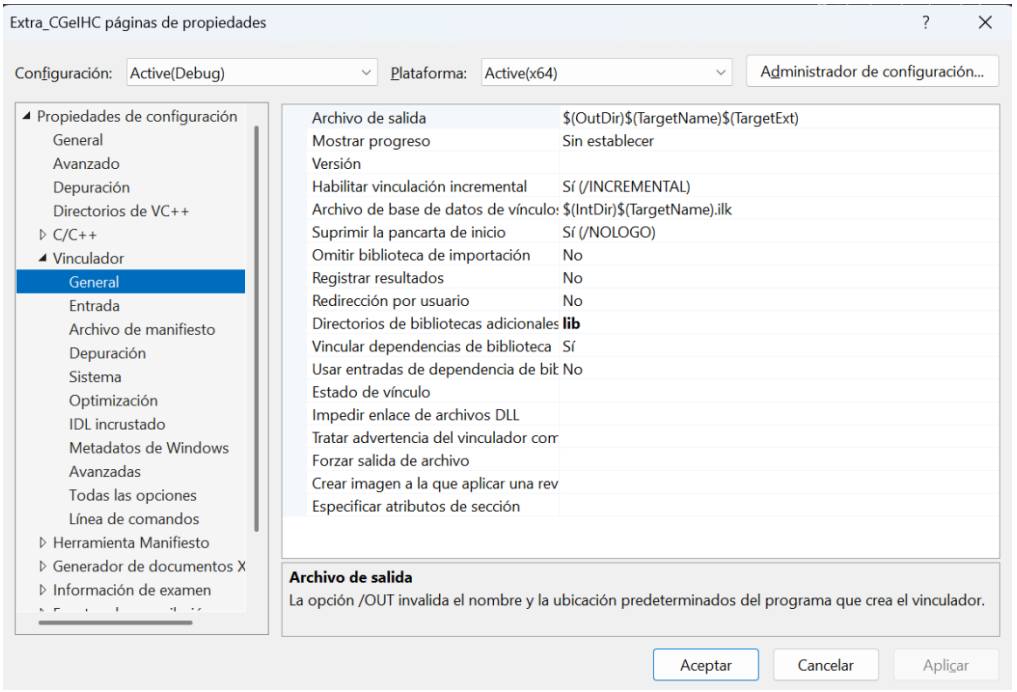
Propiedades > *General*, en las secciones *Versión del SDK de Windows* y *Conjunto de herramientas de la plataforma.*



En la parte de C/C++, se incluirá en directores de inclusión adicionales la palabra include.

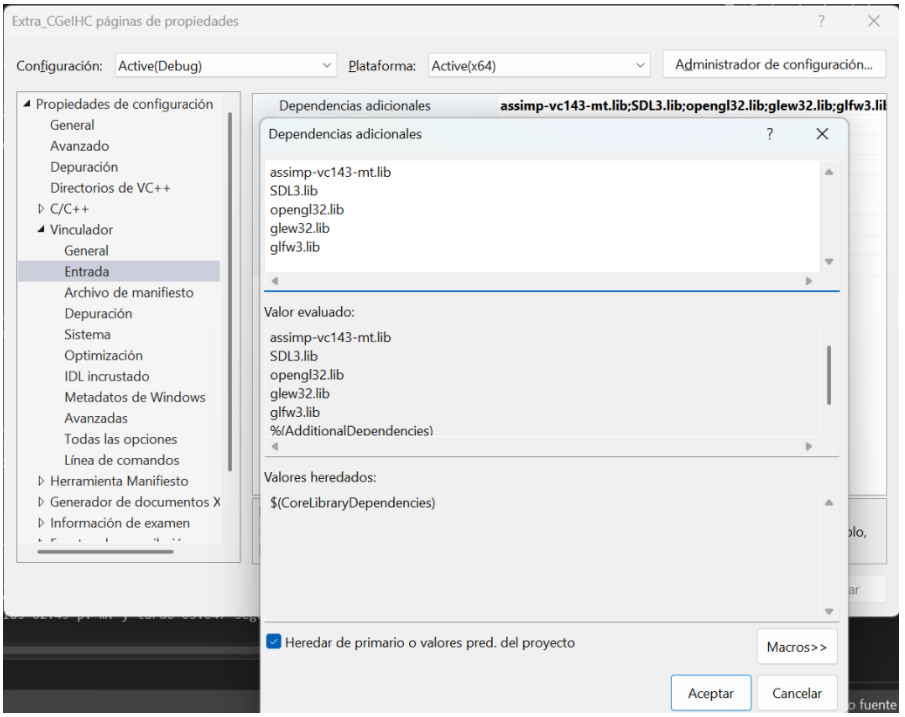


En la parte de Vinculador en general, se incluirá en directores de bibliotecas adicionales la palabra lib.



En la parte de Vinculador en entrada, se incluirán las siguientes dependencias adicionales. Cabe destacar que, si ya hay algunas incluidas, no debemos borrarlas.

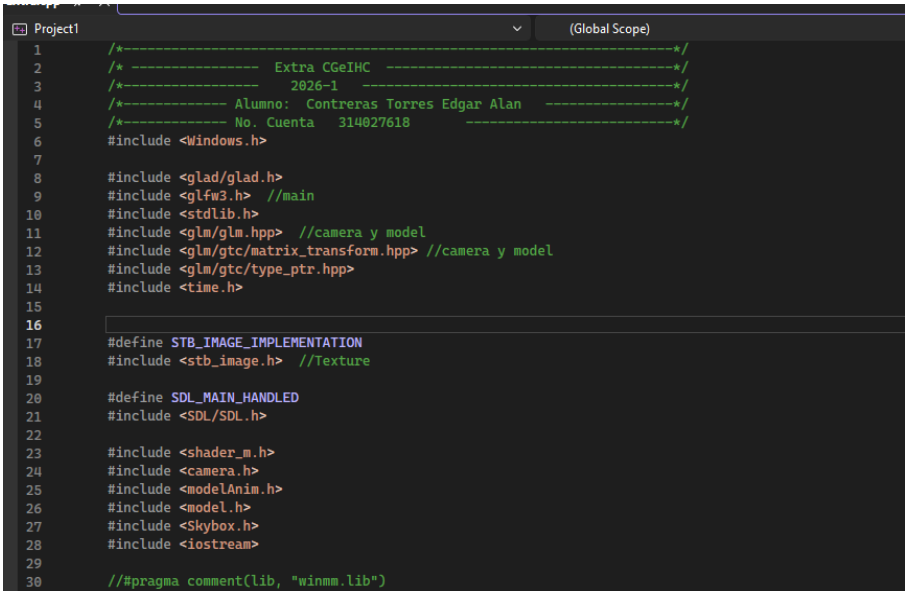
assimp-vc143-mt.lib;SDL3.lib;opengl32.lib;glew32.lib;glfw3.lib;



Cabe destacar que la configuración se impartió en la clase de laboratorio, por lo que se opta por no cambiar ni excluir alguna.

Bibliotecas

Se agregan bibliotecas como glad, glfw3, stdlib, entre otras. A continuación, se mostrarán sus funcionamientos.



Biblioteca	Funcionamiento
glad/glad.h	Cargar punteros de funciones de OpenGL durante el tiempo de ejecución.
glfw3.h	Crear ventanas y gestionar el contexto OpenGL.
glm.h	Biblioteca de matemáticas para OpenGL (vectores, matrices, transformaciones, etc.).
Time.h	Manipulación del tiempo.
Stb_image.h	Cargar imágenes para usarlas como texturas.
Sdl3/sdl.h	Simple DirectMedia Layer, utilizado para manejar entradas de usuario, gráficos y sonido.
shader_m.h	Gestión de shaders en OpenGL.
camera.h	Implementación de una cámara para OpenGL.
modelAnim.h	Modelo animado para OpenGL.
model.h	Modelo estático para OpenGL.
Skybox.h	Implementación de un Skybox para OpenGL.

Manual de uso

A continuación, podrá encontrar una lista de teclas y la descripción de sus eventos que podrá utilizar durante la ejecución del proyecto

Botones

Tecla	Acción	Descripción
R	Animación	Se utiliza poner los valores iniciales.
Barra espaciadora	Acción	Activa la variable de animación para poder iniciar, pausar o reanudar
1	Acción	Se utiliza poner la gravedad en 3.0
2	Acción	Se utiliza poner la gravedad en 2.0
3	Acción	Se utiliza poner la gravedad en 1.0
4	Acción	Se utiliza poner la gravedad en 0.5

Manejo de la cámara

Tecla	Acción	Descripción
W	Adelante	Mueve la cámara en la dirección indicada
S	Atrás	Mueve la cámara en la dirección indicada
A	Izquierda	Mueve la cámara en la dirección indicada
D	Derecha	Mueve la cámara en la dirección indicada

Conclusiones

La caída libre es un ejercicio muy práctico para juntar cálculos y animación al mismo tiempo. Por ello, se me hizo interactivo y fácil de cierta forma.

Lo mas desafiante fue la texturización de esta, ya que usando 3D Max tuve problemas, sin embargo, se resolvió viendo videos autodidactamente. Cabe destacar que la cámara no tiene una gran velocidad de movimiento, pese a que le cambie el valor hasta 10.0f. Además, tuve un impedimento al momento de ajustar el límite de la caída ya que yo lo suelo calcular con el movimiento en Y de los objetos. Pero por la rápida acelaración, puede que se pase de valor

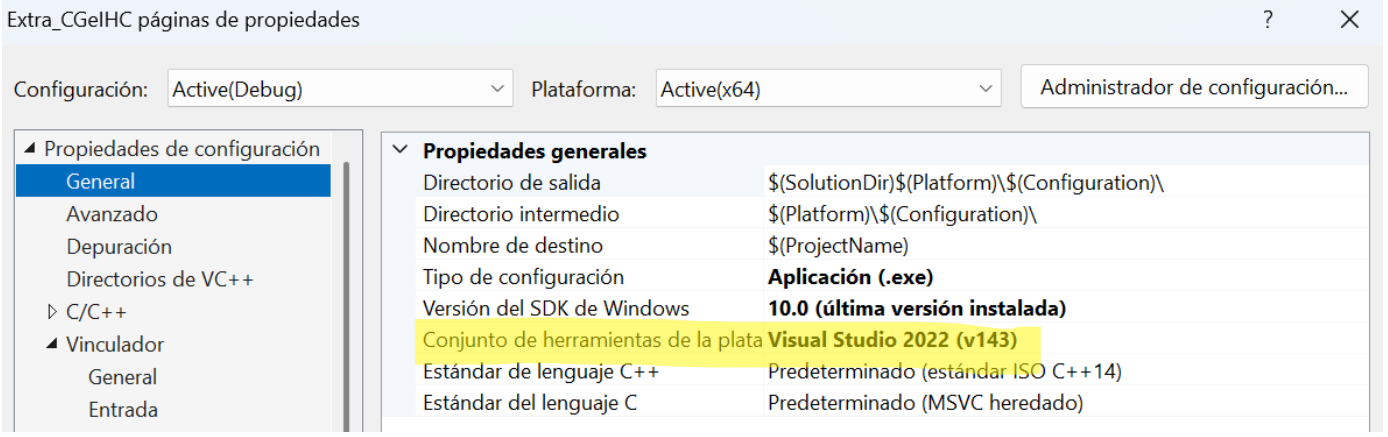
```
// camera
Camera camera(glm::vec3(0.0f, 10.0f, 90.0f));
float MovementSpeed = 10.0f;
float lastX = SCR_WIDTH / 2.0f;
float lastY = SCR_HEIGHT / 2.0f;
bool firstMouse = true;
```

```
//Vehiculo
if (animacion)
{
    // -------------1-----------------
    if (volar == 0) {

        movTor1_y -= 0.5 * gravedad * tiempo * tiempo;
        movTor2_y -= 0.5 * gravedad * tiempo * tiempo;
        movTor3_y -= 0.5 * gravedad * tiempo * tiempo;
        tiempo += dt;
        printf("1movTor3_y: %f \n", movTor3_y);
        if (movTor3_y <= -490.0f) {
            volar = 1;

        }
     }
    if (volar == 1) {
        movTor1_y += 0.5f;
        movTor2_y += 0.5f;
        movTor3_y += 0.5f;
        printf("2movTor3_y: %f \n", movTor3_y);
        if (movTor3_y >= -480.0f) {
            volar = 2;
            //animacion = true;
        }
    if (volar == 2) {
        movTor1_y -= 0.5f;
        movTor2_y -= 0.5f;
        movTor3_y -= 0.5f;
        printf("3movTor3_y: %f \n", movTor3_y);
        if (movTor3_y >= -500.0f) {
                volar = 3;
                //animacion = true;
```

El movimiento de esta misma consta de 4 teclas. Como fue detallado en clase no se tuvo ningún inconveniente

```
void my_input(GLFWwindow *window, int key, int scancode, int action, int mode)
{
    if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);
    if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
        camera.ProcessKeyboard(FORWARD, (float)deltaTime);
    if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
        camera.ProcessKeyboard(BACKWARD, (float)deltaTime);
    if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
        camera.ProcessKeyboard(LEFT, (float)deltaTime);
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
        camera.ProcessKeyboard(RIGHT, (float)deltaTime);
    //To Configure Model
```

El proyecto fue sencillo, pero abarca la mayoría de temas vistos en el laboratorio. Para mí, fue crucial el tener actualizado los .dll y las bibliotecas para que el programa corra con normalidad.
No es lo mismo correr el programa en VSC 2017 que en 2022, debido a que el conjunto de herramientas de la plataforma influye en la compilación y ejecución.



Referencias

ASSIMP_Team. (28 de 05 de 2025). *assimp-vc143-mt.dll : Free Download*. Obtenido de https://www.dllme.com/dll/files/assimp-vc143-mt

ColorAbout. (28 de 05 de 2025). *ColorAbout* . Obtenido de https://www.colorabout.com/color/rgb/255,0,0/

Perez, A. (28 de 05 de 2025). *Youtube*. Obtenido de Materiales y Texturas: https://www.youtube.com/watch?v=sy6u2_H4VOc

Stuehle, T. (28 de 05 de 2019). *Developer Community*. Obtenido de Download Visual Studio 2017: https://developercommunity.visualstudio.com/t/download-visual-studio-2017/585777