Universidad Nacional de Ingeniería

DACTIC

Ingeniería en Computación

The Virtual Gallery

Integrantes

Alaniz Herrera Roger Antonio 2023-0625U

Bran Ramos Nazareth de los Angles 2023-0863U

Flores Mendoza Lester Nahum 2023-0632U

Grupo: 3T2-COM

Docente: Danny Oswaldo Chavez Miranda

Lunes 23 de junio del 2025

Introducción

Nuestro proyecto es una creación de un entorno 3D que recrea un museo virtual, este museo virtual trata de pinturas y esculturas clásicas, estas obras fueron creadas por grandes artistas que marcaron un hito en la historia, estos artistas son; Leonardo da Vinci, Miguel Ángel, Botticelli, Fidias, Policleto, Praxiteles y Vincent van Gogh.

Este museo virtual contendrá estatuas y pinturas que han sido creadas por estos artistas. El propósito de esto es que el jugador va a poder apreciar e interactuar con los modelos. Al colisionar con el modelo, el programa mostrara una pequeña información acerca del mismo.

Este museo virtual tiene un valor importante como herramienta educativa y cultura. Esta plataforma puede ser utilizada por instituciones educativas para apoyar clases de historia del arte, humanidades o cultura general, permitiendo a los estudiantes interactuar con el contenido de una forma dinámica y memorable.

Este proyecto ofrece facilidad para el usuario o jugador para comenzar el recorrido en el interior del museo. Algunos modelos implementados en este proyecto fueron La estatua de David de Miguel Ángel, Piedad de Miguel Ángel, La Mona Lisa de Leonardo da Vinci, el retrato de una joven de Botticelli, entre otros.

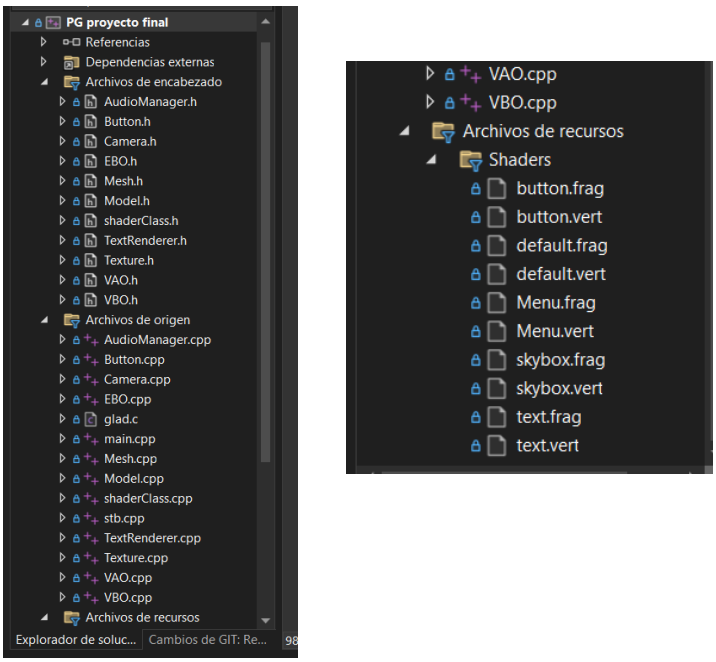
Desarrollo

The Virtual Gallery se desarrolló en Visual Community con C++ y OpenGL, usando las librerías glad, GLFW, glm, json, KHR, stb y freetype. Para el manejo del contexto de la ventana y la entrada del usuario, se utilizó la GLFW, glad se usó como cargador de funciones modernas, GLM fue para las operaciones matemáticas requeridas en el espacio tridimensional, stb_image se encarga de las texturas e imágenes.

Para la estructura del proyecto fue desarrollado con diferentes archivos, lo cual cada archivo tiene distintas funciones para el proyecto. En el main.cpp es donde se implementa la creación de la escena 3D de este proyecto utilizando las diferentes funciones creadas en los archivos de nuestro proyecto

Figura 1

Estructura del proyecto



Nota. Estructura del proyecto y como esta los archivos ubicados. [Captura] Microsoft Visual Studio Community 2022, 2025.

Figura 2

Funciones del mouse

```
// Mouse callback functions
void mouse_callback(GLFWwindow* window, double xpos, double ypos) {
    mousePos = glm::vec2(xpos, ypos);
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
    if (button == GLFW_MOUSE_BUTTON_LEFT) {
        mousePressed = (action == GLFW_PRESS);

        if (mousePressed) {
            if (menu && !showHelp && !showHelpPage2 && !showCredits) {
                for (auto& btn : menuButtons) {
                    if (btn.IsMouseOver(mousePos)) {
                        btn.OnClick(mousePos);
                    }
                }
            }
            else if (showHelp) {
                if (backButtonHelp.IsMouseOver(mousePos)) {
                    backButtonHelp.OnClick(mousePos);
                }
                if (nextButtonHelp.IsMouseOver(mousePos)) {
                    nextButtonHelp.OnClick(mousePos);
                }
            }
            else if (showHelpPage2) {
                if (prevButtonHelp.IsMouseOver(mousePos)) {
                    prevButtonHelp.OnClick(mousePos);
                }
            }
            else if (showCredits) {
                if (backButtonCredits.IsMouseOver(mousePos)) {
                    backButtonCredits.OnClick(mousePos);
                }
            }
        }
    }
}

// Keyboard callback function
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods) {
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS) {
        if (!menu && !showHelp && !showHelpPage2 && !showCredits) {
            menu = true;
        }
        else if (showHelp || showHelpPage2) {
            showHelp = false;
            showHelpPage2 = false;
        }
        else if (showCredits) {
            showCredits = false;
        }
    }
}
```

Nota. Creación de funciones para el manejo de interacciones mediante mouse y teclado. [captura] Microsoft Visual Studio Community 2022, 2025.

Se implementaron tres funciones para el manejo de entradas de usuario.

Función mouse_callback registra continuamente la posición bidimensional del cursor mediante parámetros de entrada xpos(eje x) ypos(eje y) y actualiza las variables globales mousePos utilizando glm::vec2 que almacena coordenadas en formato de vector bidimensional.

Función mouse_Button_callback gestiona los eventos del mouse con la ayuda de GLW_MOUSE_BUTTON_LEFT actualizando la variable mousePressed

Key_callback Implementa un sistema de gestión de estados para eventos de teclado.

Figura 3

Renderizado de letras

```
// Model information display functions
void showModelInfo(const std::string& title, const std::string& description) {
    showModelInfoFlag = true;
    currentModelTitle = title;
    currentModelDescription = description;
}

void hideModelInfo() {
    showModelInfoFlag = false;
}

void renderModelInfo(TextRenderer& textRenderer, Shader& textShader, int width, int height) {
    if (!showModelInfoFlag) return;

    // Set up rendering state for 2D overlay
    glDisable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    // Panel dimensions and position
    float panelWidth = 1850.0f;
    float panelHeight = 200.0f;
    float margin = 20.0f;

    // Render semi-transparent background panel
    Shader panelShader("panel.vert", "panel.frag");
    panelShader.Activate();
    glm::mat4 projection = glm::ortho(0.0f, (float)width, (float)height, 0.0f);
    glUniformMatrix4fv(glGetUniformLocation(panelShader.ID, "projection"), 1, GL_FALSE, glm::value_ptr(projection));
    glUniform4f(glGetUniformLocation(panelShader.ID, "backgroundColor"), 0.1f, 0.1f, 0.1f, 0.9f);

    // Panel vertex data
    float panelX = margin;
    float panelY = height - panelHeight - margin;
    float vertices[] = {
        panelX, panelY + panelHeight, 0.0f, 0.0f, 1.0f,
        panelX, panelY, 0.0f, 0.0f, 0.0f,
        panelX + panelWidth, panelY, 0.0f, 1.0f, 0.0f,
        panelX, panelY + panelHeight, 0.0f, 0.0f, 1.0f,
        panelX + panelWidth, panelY, 0.0f, 1.0f, 0.0f,
        panelX + panelWidth, panelY + panelHeight, 0.0f, 1.0f, 1.0f
    };

    // Create and render panel
    unsigned int VAO, VBO;
    glGenVertexArrays(1, &VAO);
    glGenBuffers(1, &VBO);
    glBindVertexArray(VAO);
    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(1);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (void*)(3 * sizeof(float)));
```

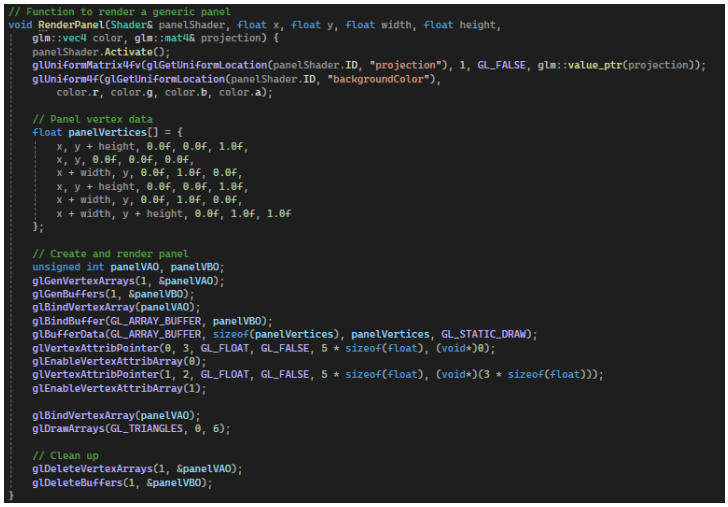
Nota. Creación de funciones para el renderizado de información. [captura] Microsoft Visual Studio Community 2022, 2025.

Este código gestiona la visualización de información sobre modelos 3D mediante funciones especializadas. Las funciones showModelInfo y hideModelInfo controlan cuándo mostrar el panel,

almacenando el título y la descripción correspondientes. La función principal, renderModelInfo, se encarga de dibujar un panel en la parte inferior de la pantalla para una mejor visualización de la información, utilizando shaders para su renderizado, y posteriormente muestra el texto sobre este panel.

Figura 4

Renderizado de panel informativo



```
// Function to render a generic panel
void RenderPanel(Shader& panelShader, float x, float y, float width, float height,
    glm::vec4 color, glm::mat4& projection) {
    panelShader.Activate();
    glUniformMatrix4fv(glGetUniformLocation(panelShader.ID, "projection"), 1, GL_FALSE, glm::value_ptr(projection));
    glUniform4f(glGetUniformLocation(panelShader.ID, "backgroundColor"),
        color.r, color.g, color.b, color.a);

    // Panel vertex data
    float panelVertices[] = {
        x, y + height, 0.0f, 0.0f, 1.0f,
        x, y, 0.0f, 0.0f, 0.0f,
        x + width, y, 0.0f, 1.0f, 0.0f,
        x, y + height, 0.0f, 0.0f, 1.0f,
        x + width, y, 0.0f, 1.0f, 0.0f,
        x + width, y + height, 0.0f, 1.0f, 1.0f
    };

    // Create and render panel
    unsigned int panelVAO, panelVBO;
    glGenVertexArrays(1, &panelVAO);
    glGenBuffers(1, &panelVBO);
    glBindVertexArray(panelVAO);
    glBindBuffer(GL_ARRAY_BUFFER, panelVBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(panelVertices), panelVertices, GL_STATIC_DRAW);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 5 * sizeof(float), (void*)(3 * sizeof(float)));
    glEnableVertexAttribArray(1);

    glBindVertexArray(panelVAO);
    glDrawArrays(GL_TRIANGLES, 0, 6);

    // Clean up
    glDeleteVertexArrays(1, &panelVAO);
    glDeleteBuffers(1, &panelVBO);
}
```

Nota. Renderizado de panel para mejor visualización. [captura] Microsoft Visual Studio Community 2022, 2025.

Esta función dibuja paneles rectangulares personalizables para interfaces 2D. Recibe parámetros de posición (x, y), tamaño (width, height), color (RGBA) y matriz de proyección, generando dinámicamente la geometría necesaria (dos triángulos formando un rectángulo). Utiliza un shader específico para aplicar las propiedades visuales y maneja automáticamente los buffers de vértices (creación y eliminación en cada ejecución), optimizando el rendimiento sin dejar residuos en memoria

Figura 5

Sonido del entorno

```
//Música y Sonido
std::vector<std::string> envSongs = {
    //"Sound/environment.mp3",
    "Sound/env1.mp3",
    "Sound/env2.mp3",
    "Sound/env3.mp3"
};

std::string pickRandomSong(const std::vector<std::string>& songs) {
    if (songs.empty()) return "";
    int index = rand() % songs.size();
    return songs[index];
}
```

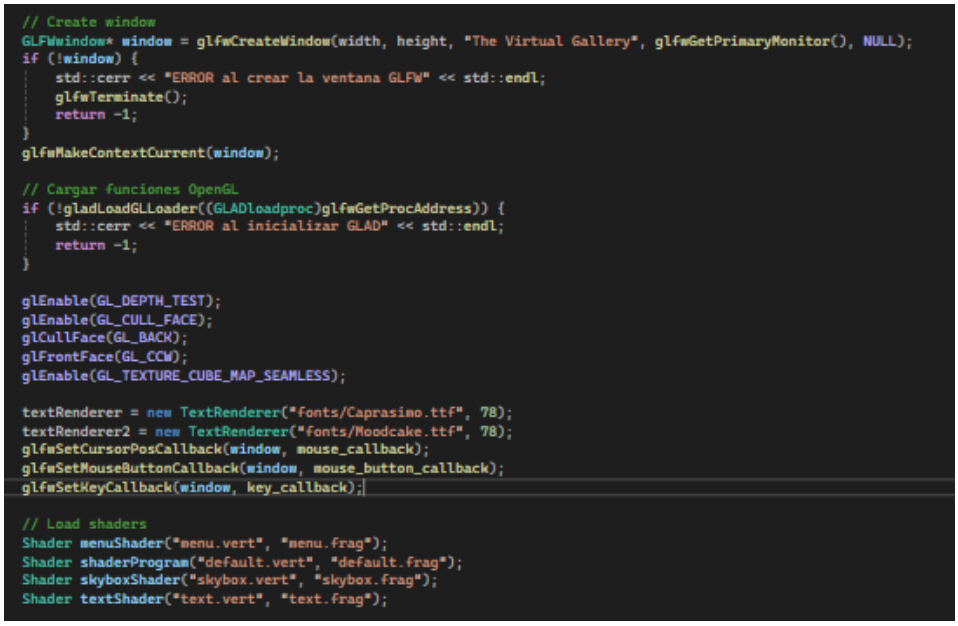
Nota. Implementación de múltiples sonidos para el entorno [captura] Microsoft Visual Studio Community 2022, 2025.

Creamos un arreglo de la dirección en carpeta de cada una de las canciones que queremos que se reproduzcan en nuestro entorno para luego poder utilizar la siguiente función.

Se creo la función pickRandomSong() para poder escoger de manera al azar uno dee los 3 sonidos predeterminados para el entrno, consiste en un random que verifica la canción que sonaba antes para no repetir la misma canción cuando deje de sonar

Figura 6

Creacion de la ventana y Shaders



```
// Create window
GLFWwindow* window = glfwCreateWindow(width, height, "The Virtual Gallery", glfwGetPrimaryMonitor(), NULL);
if (!window) {
    std::cerr << "ERROR al crear la ventana GLFW" << std::endl;
    glfwTerminate();
    return -1;
}
glfwMakeContextCurrent(window);

// Cargar funciones OpenGL
if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)) {
    std::cerr << "ERROR al inicializar GLAD" << std::endl;
    return -1;
}

glEnable(GL_DEPTH_TEST);
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);
glFrontFace(GL_CCW);
glEnable(GL_TEXTURE_CUBE_MAP_SEAMLESS);

textRenderer = new TextRenderer("fonts/Caprasimo.ttf", 78);
textRenderer2 = new TextRenderer("fonts/Moodcake.ttf", 78);
glfwSetCursorPosCallback(window, mouse_callback);
glfwSetMouseButtonCallback(window, mouse_button_callback);
glfwSetKeyCallback(window, key_callback);

// Load shaders
Shader menuShader("menu.vert", "menu.frag");
Shader shaderProgram("default.vert", "default.frag");
Shader skyboxShader("skybox.vert", "skybox.frag");
Shader textShader("text.vert", "text.frag");
```

Nota. Creación de la ventana, carga las funciones y los shaders [captura] Microsoft Visual Studio Community 2022, 2025.

En el main.cpp en esa imagen, es donde se crea la ventana del proyecto y después manda a llamar las funciones de la estructura de dicho programa. También se realiza la configuración inicial del entorno gráfico 3D y del sistema de entrada del usuario. Esto habilita las características clave del OpenGL que brindan un buen rendimiento y realista de la escena, como el manejo de profundidad, la optimización en el dibujo de objetos y el soporte para entornos cúbicos como el skybox. Se carga los shaders del menu.vert, menu.frag, default.vert, default.frag, skybox.vert skybox.frag, text.vert y text.frag.

Figura 7

Inicializando botones y pantallas 2d

```
// Configuracion de botones del menu
menuButtons.emplace_back(
    glm::vec2(width / 2 - 150, height / 2),
    glm::vec2(300, 60),
    "ENTRAR",
    [&]() { menu = false; }
);

menuButtons.emplace_back(
    glm::vec2(width / 2 - 150, height / 2 + 80),
    glm::vec2(300, 60),
    "AYUDA",
    [&]() { showHelp = true; }
);

menuButtons.emplace_back(
    glm::vec2(width / 2 - 150, height / 2 + 160),
    glm::vec2(300, 60),
    "CREDITOS",
    [&]() { showCredits = true; }
);

menuButtons.emplace_back(
    glm::vec2(width / 2 - 150, height / 2 + 240),
    glm::vec2(300, 60),
    "SALIR",
    [window]() { glfwSetWindowShouldClose(window, true); }
);

// Configuracion de geometria para menu y pantallas 2D
GLuint quadVAO, quadVBO;
glGenVertexArrays(1, &quadVAO);
glGenBuffers(1, &quadVBO);

float vertices[] = {
    // Posiciones   // Coord. Textura
   -1.0f,  1.0f,  0.0f, 1.0f,  // Superior izquierda
   -1.0f, -1.0f,  0.0f, 0.0f,  // Inferior izquierda
    1.0f, -1.0f,  1.0f, 0.0f,  // Inferior derecha

   -1.0f,  1.0f,  0.0f, 1.0f,  // Superior izquierda
    1.0f, -1.0f,  1.0f, 0.0f,  // Inferior derecha
    1.0f,  1.0f,  1.0f, 1.0f   // Superior derecha
};

// Light configuration
glm::vec4 lightColor = glm::vec4(1.0f, 1.0f, 1.0f, 1.0f);
glm::vec3 lightPos = glm::vec3(0.5f, 0.5f, 0.5f);

shaderProgram.Activate();
glUniform4f(glGetUniformLocation(shaderProgram.ID, "lightColor"), lightColor.x, lightColor.y, lightColor.z, lightColor.w);
glUniform3f(glGetUniformLocation(shaderProgram.ID, "lightPos"), lightPos.x, lightPos.y, lightPos.z);
skyboxShader.Activate();
glUniform1i(glGetUniformLocation(skyboxShader.ID, "skybox"), 0);
```

Nota. Inicialización de botones. [captura] Microsoft Visual Studio Community 2022, 2025.

Este código inicializa los botones interactivos del menú principal del museo virtual, posicionándolos verticalmente en el centro de la pantalla. Cada botón se define mediante parámetros específicos: posición (posX, posY), dimensiones (ancho y alto), texto descriptivo y una función lambda que determina la acción al hacer clic.

Figura 8

Carga de modelos

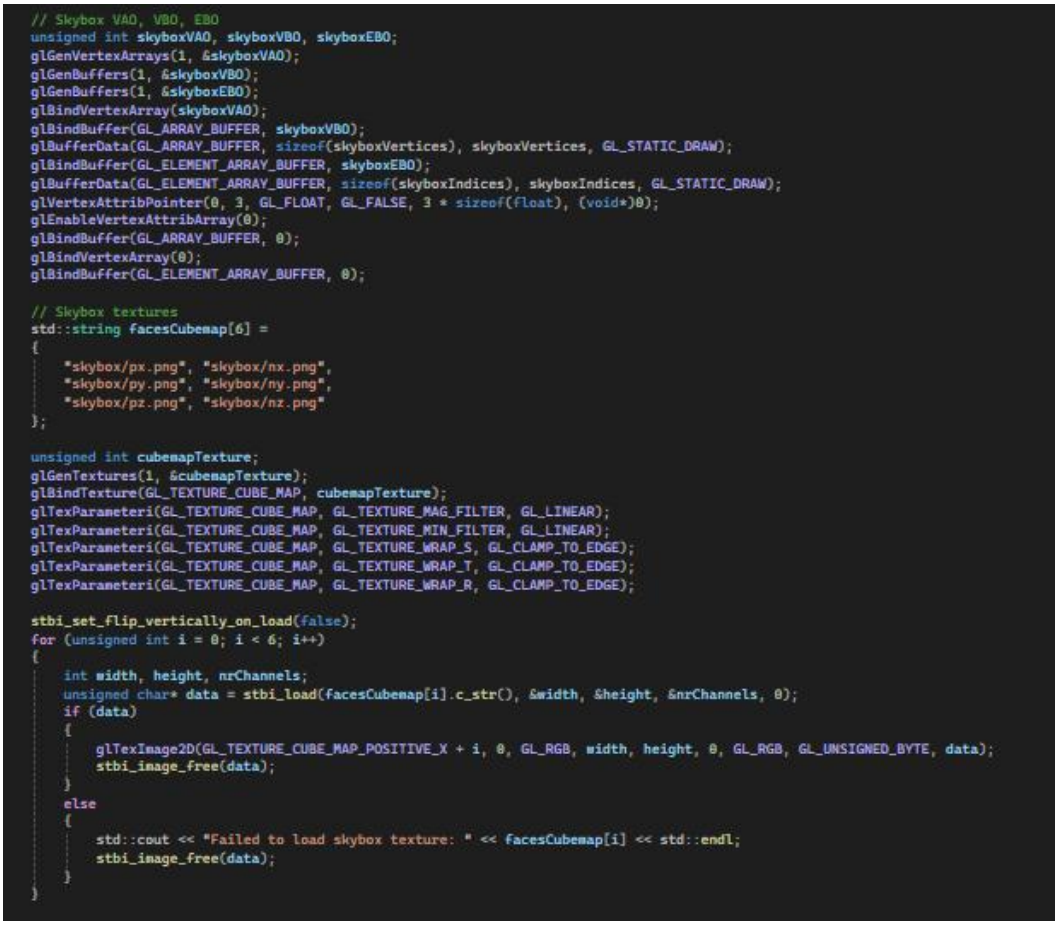
```
// Create camera
Camera camera(width, height, glm::vec3(0.0f, 2.0f, 20.0f));
camera.setAudioManager(&Sound);

stbi_set_flip_vertically_on_load(false); // Important for 3D models

// Load models
Model model("modelos/piso/scene.gltf", glm::vec3(2.0f), glm::vec3(0.0f, 13.0f, 0.0f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f)); // Escala normal
Model pit("modelos/da vinci/mona_lisa/scene.gltf", glm::vec3(0.8f), glm::vec3(3.0f, 16.5f, -4.0f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model escul("modelos/miguel ang/david2/scene.gltf", glm::vec3(0.6f), glm::vec3(-0.7f, 31.0f, 5.4f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model escul2("modelos/miguel ang/pieta/scene.gltf", glm::vec3(1.3f), glm::vec3(13.0f, 18.0f, 0.8f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));

Model escul3("modelos/miguel ang/moises/scene.gltf", glm::vec3(0.6f), glm::vec3(8.0f, 27.0f, -8.5f), glm::quat(0.0f, 0.0f, 1.0f, 0.0f));

Model escul4("modelos/Botticelli/joven/scene.gltf", glm::vec3(2.1f), glm::vec3(0.5f, 4.35f, -1.3f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model room("modelos/room/scene.gltf", glm::vec3(1.2f), glm::vec3(-1.0f, 3.5f, -4.0f), glm::quat(0.0f, 0.0f, 0.0f, 1.0f));
Model room2("modelos/room2/scene.gltf", glm::vec3(1.3f), glm::vec3(0.0f, 20.0f, 0.5f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model escul5("modelos/fidias/atena/scene.gltf", glm::vec3(0.3f), glm::vec3(-11.0f, 5.0f, 18.9f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model escul6("modelos/policleto/dori/scene.gltf", glm::vec3(0.31f), glm::vec3(-20.0f, 58.0f, 28.5f), glm::quat(0.0f, 0.0f, 1.0f, 0.0f));
Model escul7("modelos/praxi/afrodita/scene.gltf", glm::vec3(0.7f), glm::vec3(10.0f, -12.0f, -22.5f), glm::quat(0.0f, 0.0f, 0.0f, 1.0f));
Model pit2("modelos/van gogh/noche_estrella/scene.gltf", glm::vec3(0.9f), glm::vec3(-3.0f, 16.2f, -4.0f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));

Model escul8("modelos/torso/scene.gltf", glm::vec3(1.5f), glm::vec3(3.8f, 26.0f, 1.5f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));

Model pilar("modelos/pilar/scene.gltf", glm::vec3(0.09f), glm::vec3(-16.0f, 28.0f, 0.9f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model pilar2("modelos/pilar/scene.gltf", glm::vec3(0.09f), glm::vec3(-16.0f, 42.0f, 0.9f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model pilar3("modelos/pilar/scene.gltf", glm::vec3(0.09f), glm::vec3(16.0f, 28.0f, 0.9f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model pilar4("modelos/pilar/scene.gltf", glm::vec3(0.09f), glm::vec3(16.0f, 42.0f, 0.9f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model vase("modelos/vase/rosa1/scene.gltf", glm::vec3(1.8f), glm::vec3(-3.0f, 10.5f, -1.6f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model vase2("modelos/vase/rosa2/scene.gltf", glm::vec3(0.7f), glm::vec3(-8.0f, 51.3f, -12.4f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model vase3("modelos/vase/rosa3/scene.gltf", glm::vec3(1.5f), glm::vec3(-15.5f, 26.4f, 4.1f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
Model vase4("modelos/vase/rosa4/scene.gltf", glm::vec3(1.5f), glm::vec3(3.7f, 22.0f, -2.3f), glm::quat(0.0f, 1.0f, 0.0f, 0.0f));
```

Nota. Creación de la camara, y carga de modelos mandado a llamar de la carpeta [captura] Microsoft Visual Studio Community 2022, 2025.

Aquí se crea inicialmente la cámara. Esto es por los archivos camera.h y camera.cpp que mediante las funciones podemos manejar la cámara de manera eficiente, después se manda a cargar los modelos que se ocupan en el proyecto, esto es por los dos archivos model.h y model.cpp. El modelo se manda a llamar por la dirección donde está guardado el modelo que es en la carpeta modelos que están dentro carpeta general del proyecto, cada modelo que manda a llamar se le agrega cuanto va a ser su escalar (el tamaño), la translación del modelo (ubicación del modelo) y la rotación del modelo, esto se hizo mediante el archivo model.cpp ya que tienes esas tres funciones que recibe esos valores.

Figura 9

Inicializando VAO VBO y EBO del Skybox



```
// Skybox VAO, VBO, EBO
unsigned int skyboxVAO, skyboxVBO, skyboxEBO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1, &skyboxVBO);
glGenBuffers(1, &skyboxEBO);
glBindVertexArray(skyboxVAO);
glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices), skyboxVertices, GL_STATIC_DRAW);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, skyboxEBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(skyboxIndices), skyboxIndices, GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);

// Skybox textures
std::string facesCubemap[6] =
{
    "skybox/px.png", "skybox/nx.png",
    "skybox/py.png", "skybox/ny.png",
    "skybox/pz.png", "skybox/nz.png"
};

unsigned int cubemapTexture;
glGenTextures(1, &cubemapTexture);
glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

stbi_set_flip_vertically_on_load(false);
for (unsigned int i = 0; i < 6; i++)
{
    int width, height, nrChannels;
    unsigned char* data = stbi_load(facesCubemap[i].c_str(), &width, &height, &nrChannels, 0);
    if (data)
    {
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
        stbi_image_free(data);
    }
    else
    {
        std::cout << "Failed to load skybox texture: " << facesCubemap[i] << std::endl;
        stbi_image_free(data);
    }
}
```

Nota. Se inicializa y asignan las partes del cubemap [captura] Microsoft Visual Studio Community 2022, 2025.

Se inicializan los buffers necesarios para manejar el skybox (Vertex Array Buffer, Vertex Buffer Object, Element Buffer Object). Se carga la información de los vértices (skyboxVertices) y los índices (skyboxIndices) en memoria de GPU utilizando glBufferData.

Se especifican las rutas a las imágenes que formaran las seis caras del cubo px, nx, py, ny, pz, nz. Representando derecha, izquierda, arriba, abajo, delante y detrás.

Luego se carga cada imagen usando la librería stb_image.h, y se asigna a la cara correspondiente del GL_TEXTURE_CUBE_MAP.

Estos parámetros configuran el comportamiento del cubemap: Filtros lineales para minimizar y magnificar y Clamp a los bordes para evitar costuras entre las caras.

Figura 11

Asignacion de colisiones

```
// Render de la escena 3D
camera.Inputs(window);
camera.updateMatrix(60.0f, 0.1f, 100.0f);

// Aplicar límites a la cámara
camera.Position.x = limits(camera.Position.x, limit_min.x, limit_max.x);
camera.Position.y = limits(camera.Position.y, limit_min.y, limit_max.y);
camera.Position.z = limits(camera.Position.z, limit_min.z, limit_max.z);

// Colisiones (ajusta los valores según tus modelos y necesidades)
camera.AddCollider(glm::vec3(0.3201, 3.97696, 32.9264), 3.0f, "David", "Escultura de Miguel angel (1501-1504). Marmol blanco de 5.17 metros.");
camera.AddCollider(glm::vec3(-13.9183, 2.28625, 18.5213), 3.0f, "La Piedad", "Escultura de Miguel angel (1498-1499). Marmol, Basilica de San Pedro.");
camera.AddCollider(glm::vec3(16.4938, 3.65387, 16.5488), 2.0f, "Atenea Partenos", "Escultura de Fidias (siglo V a.C.). Replica moderna.");
camera.AddCollider(glm::vec3(14.8127, 3.80944, 49.5978), 2.0f, "Doriforo", "Escultura de Policleto (450-440 a.C.). Copia romana en marmol.");
camera.AddCollider(glm::vec3(8.51457, 2.15766, 49.942), 2.0f, "Afrodita de Cnido", "Escultura de Praxoteles (siglo IV a.C.). Copia romana.");
camera.AddCollider(glm::vec3(-8.46455, 1.67231, 50.2564), 2.0f, "Moises", "Escultura de Miguel angel (1513-1515). Marmol, tumba del Papa Julio II.");
camera.AddCollider(glm::vec3(-16.9746, 2.62017, 45.6913), 2.0f, "Torso de Belvedere", "Escultura helenestica (siglo I a.C.). Marmol, Museos Vaticanos.");
camera.AddCollider(glm::vec3(-17.1125, 2.5, 25.5523), 1.0f);
camera.AddCollider(glm::vec3(-17.1308, 2.5, 38.5671), 1.0f);
camera.AddCollider(glm::vec3(17.507, 2.5, 25.7276), 1.0f);
camera.AddCollider(glm::vec3(17.1492, 2.5, 38.6279), 1.0f);

// Verificar proximidad a modelos
checkModelProximity(camera, exhibitModels);
// Render Skybox
glDepthFunc(GL_LEQUAL);
skyboxShader.Activate();
```

Nota. Se añaden las colisiones al entorno [captura] Microsoft Visual Studio Community 2022, 2025.

La cámara responde a la entrada del teclado y del mouse. Se actualiza su matriz de vista/proyección con: FOV de 60 grados, Near plane: 0.1f y Far plane: 100.0f

Para evitar que la cámara salga del espacio permitido, se aplican límites en los ejes X, Y y Z mediante la función Limits.

Se añaden colisionadores esféricos a la escena. Cada uno representa una escultura o punto de interés con: Posición 3D. Radio (definido internamente). Descripción (como nombre, época y autor de la obra). Estas colisiones impiden que el usuario atraviese los modelos.

Figura 12

Render escena

```
    // Render modelos 3D
    shaderProgram.Activate();
    camera.Matrix(shaderProgram, "camMatrix");

    model.Draw(shaderProgram, camera);
    pit.Draw(shaderProgram, camera);
    escul.Draw(shaderProgram, camera);
    escul2.Draw(shaderProgram, camera);
    escul3.Draw(shaderProgram, camera);
    escul4.Draw(shaderProgram, camera);
    //room.Draw(shaderProgram, camera);
    room2.Draw(shaderProgram, camera);
    escul5.Draw(shaderProgram, camera);
    escul6.Draw(shaderProgram, camera);
    escul7.Draw(shaderProgram, camera);
    pit2.Draw(shaderProgram, camera);
    escul8.Draw(shaderProgram, camera);

    pilar.Draw(shaderProgram, camera);
    pilar2.Draw(shaderProgram, camera);
    pilar3.Draw(shaderProgram, camera);
    pilar4.Draw(shaderProgram, camera);
    vase.Draw(shaderProgram, camera);
    vase2.Draw(shaderProgram, camera);
    vase3.Draw(shaderProgram, camera);
    vase4.Draw(shaderProgram, camera);

    // Renderizar informacion del modelo
    renderModelInfo(*textRenderer2, textShader, width, height);
}

glfwSwapBuffers(window);
glfwPollEvents();
```

Nota. Dibujamos todos los modelos 3d para la escena final [captura] Microsoft Visual Studio Community 2022, 2025.

Dentro del bucle principal, en ese apartado se realiza el dibujo de todos los modelos 3D que hemos mandado a llamar y que conforman la escena del museo virtual. Cada modelo que mandamos a llamar es renderizado utilizando el mismo programa de sombreado y la cámara configurada previamente.

También se renderiza la información que conforma cada modelo. Este apartado es muy importante para la visualización del contenido artístico dentro del museo.

Conclusión

El proyecto The Virtual Gallery logra su objetivo principal de crear un museo virtual en 3D interactivo, ofreciendo una experiencia inmersiva para explorar obras de arte clásicas de artistas como Da Vinci, Miguel Ángel y Van Gogh. La implementación de tecnologías como OpenGL, GLFW y sistemas de renderizado avanzado permite una navegación fluida y una visualización detallada de las obras, junto con información accesible para el usuario.

Además, el proyecto destaca por su potencial educativo, ya que puede ser utilizado como herramienta complementaria en clases de historia del arte, humanidades o cultura general. La interacción dinámica con los modelos 3D y el diseño intuitivo de la interfaz mejoran la experiencia del usuario, haciendo que el aprendizaje sea más atractivo y memorable.

En el aspecto técnico, el desarrollo demuestra un manejo sólido de conceptos clave, como la gestión de cámaras, colisiones, sonido ambiental y shaders. Sin embargo, hay oportunidades de mejora, como la optimización del rendimiento, la inclusión de más obras artísticas y la expansión de funcionalidades interactivas.

En resumen, The Virtual Gallery no solo cumple con su propósito inicial, sino que también sienta las bases para futuras actualizaciones, consolidándose como un proyecto innovador en el ámbito de los entornos virtuales educativos.

Bibliografía

“Klockwork_Animation”. (2022, septiembre 30). *Low Res David* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/low-res-david-66c3406d65d74c97951a56f8fc2d002e

“andrea.notarstefano”. (2019, octubre 20). *Muqarnas Pillar* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/muqarnas-pillar-81f9be447fa14c66b7b384ca32953fc0

“Minneapolis Institute of Art”. (2022, abril 8). *Ghirlandaios: Portrait of a Young Woman* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/ghirlandaios-portrait-of-a-young-woman-dc8e21c7fb4b4d64b822737d8cf94f53

“3Dst8”. (2014, julio 17). *Athena 3Dst8* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/athena-3dst8-6e77248670c64c478ac556ba64f250aa

“Michael Douglass”. (2022, febrero 6). *Flowers in Vase* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/flowers-in-vase-b1047276fc7f4421b5f695ad9ff59e72

“Harryfolwell”. (2023, noviembre 20). *Flower in Flower Pot* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/flower-in-flower-pot-cae5718e1fe3483a859eba50ff01aa07

“IESFloridablanca”. (2023, noviembre 1). *Moisés de Miguel Ángel* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/moises-de-miguel-angel-4d1a12147dd54bfc86cada1892a8485bfc86cada1892a8485b

“Geoffrey Marchal”. (2016, septiembre 7). *Venus Medici* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/venus-medici-708a4166439c45d799f4eaa40d0db70e

“Owlish Media”. (2017, agosto 14). *Spear Bearer (Low-Poly)* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/spear-bearer-low-poly-849dc96336b94ef099ec704a290db02a

“mikesira”. (2022, diciembre 7). *Belvedere Torso* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/belvedere-torso-b73aa07872074d57b895eb8b6cf7d1a4

“SUJESH NAIR”. (2023, octubre 24). *Pietà: a 3D tribute to Michelangelo’s masterpiece* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/pieta-a-3d-tribute-to-michelangelos-masterpiece-b94d8c93e9e947cbb29a5ebeb1dd09c4

“flippednormal”. (2022, abril 18). *Flower Bouquet* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/flower-bouquet-bee4f7b602904024826d6e49ddb95ad5

“Advanced Visualization Lab - Indiana University”. (2022, Agosto 2). *Zinnias in a Decorated Tile Vase* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/zinnias-in-a-decorated-tile-vase-ad30d051cc274224b29c11f371cf5b40

“voxinbain”. (2023, octubre 5). *The Mona Lisa* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/the-mona-lisa-9823e411861d49a998fce9cee0a63947

“voxinbain”. (2023, octubre 5). *The Starry Night* [Modelo 3D]. Sketchfab. https://sketchfab.com/3d-models/the-starry-night-bbe287c595a34b8ea76bf9cc9e8f0053

Not From Nothing. (n.d.). *stb single-file public domain libraries for C/C++* [Repositorio en GitHub]. GitHub. https://github.com/nothings/stb

G-Truc. (2014). *GLM (OpenGL Mathematics) 0.9.9 Documentation* [Documentación]. https://glm.g-truc.net/0.9.9/index.html

David Eberly. (n.d.). *GLAD – Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator* [Sitio web]. https://glad.dav1d.de/

GLFW contributors. (n.d.). *GLFW – Tiny C library for creating windows with OpenGL contexts* [Sitio web]. https://www.glfw.org/

Niels Lohmann. (n.d.). *JSON for Modern C++* [Repositorio en GitHub]. GitHub. https://github.com/nlohmann/json

Microsoft Corporation. (n.d.). *Visual Studio Community* [Software]. Microsoft Visual Studio. https://visualstudio.microsoft.com/es/vs/community/

Luke Faulkner-Topic. (2022, abril 7.). *[Trois gymnopédies: No. 1, Lent et douloreux]* [Video]. YouTube. https://www.youtube.com/watch?v=wN193eERvtg&ab_channel=LukeFaulkner-Topic

Victor Gordan. (2021, junio 18.). *[OpenGL Tutorial 19 - Cubemaps & Skyboxes] [Video]. YouTube. https://www.youtube.com/watch?v=8sVvxeKI9Pk&ab_channel=VictorGordan*

SMORT. (2011, marzo 4). *[C418 - Mice on Venus - Minecraft Volume Alpha] [Video]. YouTube. https://www.youtube.com/watch?v=DZ47H84Bc_Q&ab_channel=SMORT*

SMORT. (2011, marzo 4). *[C C418 - Minecraft - Minecraft Volume Alpha] [Video]. YouTube.* https://www.youtube.com/watch?v=qq-RGFyaq0U&ab_channel=SMORT

SMORT. (2011, marzo 4). *[C418 - Sweden - Minecraft Volume Alpha] [Video]. YouTube. https://youtu.be/aBkTkxKDduc?si=wGhdG2hBc_BmkYQz*

Victor Gordan. (2021, abril 2). ***[OpenGL Tutorial 13 - Model Loading]*** [Video]. YouTube. https://www.youtube.com/watch?v=AB_f4slL2H0&list=PLPaoO-vpZnumdcb4tZc4x5Q-v7CkrQ6M-&index=14

Pinterest user. (n.d.). *[Imagen sin título] [Imagen]. Pinterest. https://es.pinterest.com/pin/621496817352763907/*

Poly Haven. (n.d.). *Spruit Sunrise [HDRI environment]. https://polyhaven.com/a/spruit_sunrise*